Bu kitaba sığmayan daha neler var!

2023 Eğitim Vizyonu

Karekodu okut, bu kitapla ilgili EBA içeriklerine ulaş!

BU DERS KİTABI MİLLÎ EĞİTİM BAKANLIĞINCA ÜCRETSİZ OLARAK VERİLMİŞTİR.
PARA İLE SATILAMAZ.

Bandrol Uygulamasına İlişkin Usul ve Esaslar Hakkında Yönetmeliğin Beşinci Maddesinin İkinci Fıkrası Çerçevesinde Bandrol Taşıması Zorunlu Değildir.

NESNE TABANLI PROGRAMLAMA 10. SINIF DERS KİTABI

MESLEKİ VE TEKNİK ANADOLU LİSESİ

BİLİŞİM TEKNOLOJİLERİ ALANI

NESNE TABANLI PROGRAMLAMA

10

DERS KİTABI

MESLEKİ VE TEKNİK ANADOLU LİSESİ
BİLİŞİM TEKNOLOJİLERİ ALANI

NESNE TABANLI PROGRAMLAMA

10

DERS KİTABI

YAZARLAR

ABDULLAH HOCAOĞLU
DEVRİM ALTINKURT
MURAT İMSİYATOĞLU
MUSTAFA NACAR
YASEMİN AKPINAR

T.C. MİLLÎ EĞİTİM BAKANLIĞI

MİLLÎ EĞİTİM BAKANLIĞI YAYINLARI 7555
YARDIMCI VE KAYNAK KİTAPLAR DİZİSİ 1595

HAZIRLAYANLAR

DİL UZMANI
Melek DEMİR

PROGRAM GELİŞTİRME UZMANI
Esra YAVUZ

ÖLÇME DEĞERLENDİRME UZMANI
Arzu DURSUN URGUN

REHBERLİK UZMANI
Gülşen YALIN

GÖRSEL TASARIM UZMANLARI
Fırat DOĞAN
Nalan OFLAS GÜLER

ISBN:978-975-11-5702-7
"Millî Eğitim Bakanlığının 24.12.2020 gün ve 18433886 sayılı oluru ile
Meslekî ve Teknik Eğitim Genel Müdürlüğünce öğretim materyali olarak hazırlanmıştır."

İSTİKLÂL MARŞI

Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak;
O benimdir, o benim milletimindir ancak.

Çatma, kurban olayım, çehreni ey nazlı hilâl!
Kahraman ırkıma bir gül! Ne bu şiddet, bu celâl?
Sana olmaz dökülen kanlarımız sonra helâl.
Hakkıdır Hakk'a tapan milletimin istiklâl.

Ben ezelden beridir hür yaşadım, hür yaşarım.
Hangi çılgın bana zincir vuracakmış? Şaşarım!
Kükremiş sel gibiyim, bendimi çiğner, aşarım.
Yırtarım dağları, enginlere sığmam, taşarım.

Garbın âfâkını sarmışsa çelik zırhlı duvar,
Benim iman dolu göğsüm gibi serhaddim var.
Ulusun, korkma! Nasıl böyle bir imanı boğar,
Medeniyyet dediğin tek dişi kalmış canavar?

Arkadaş, yurduma alçakları uğratma sakın;
Siper et gövdeni, dursun bu hayâsızca akın.
Doğacaktır sana va'dettiği günler Hakk'ın;
Kim bilir, belki yarın, belki yarından da yakın.

Bastığın yerleri toprak diyerek geçme, tanı:
Düşün altındaki binlerce kefensiz yatanı.
Sen şehit oğlusun, incitme, yazıktır, atanı:
Verme, dünyaları alsan da bu cennet vatanı.

Kim bu cennet vatanın uğruna olmaz ki feda?
Şüheda fışkıracak toprağı sıksan, şüheda!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüda.

Ruhumun senden İlâhî, şudur ancak emeli:
Değmesin mabedimin göğsüne nâmahrem eli.
Bu ezanlar -ki şehadetleri dinin temeli-
Ebedî yurdumun üstünde benim inlemeli.

O zaman vecd ile bin secde eder -varsa- taşım,
Her cerîhamdan İlâhî, boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerret gibi yerden na'şım;
O zaman yükselerek arşa değer belki başım.

Dalgalan sen de şafaklar gibi ey şanlı hilâl!
Olsun artık dökülen kanlarımın hepsi helâl.
Ebediyyen sana yok, ırkıma yok izmihlâl;
Hakkıdır hür yaşamış bayrağımın hürriyyet;
Hakkıdır Hakk'a tapan milletimin istiklâl!

Mehmet Âkif Ersoy

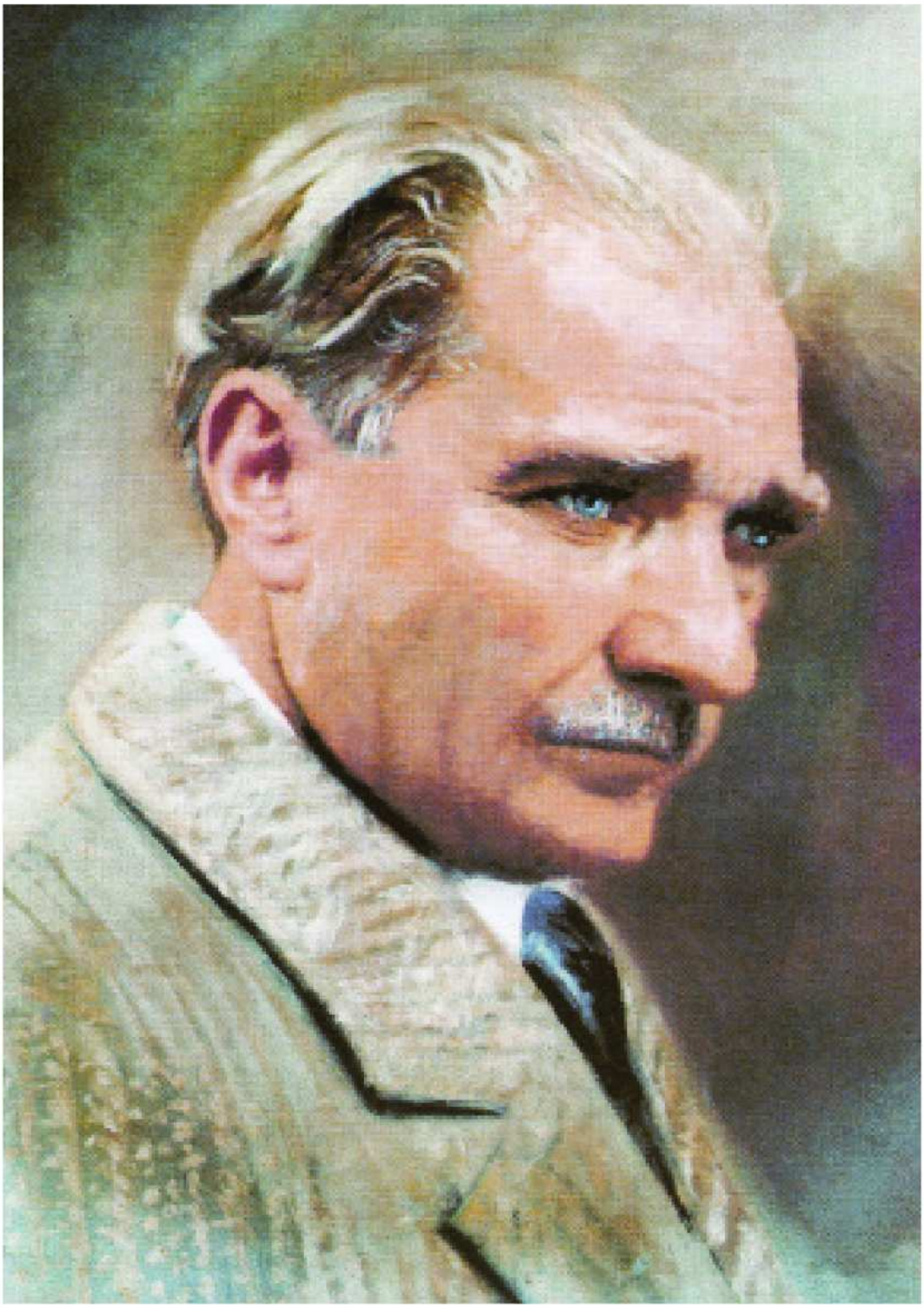
GENÇLİĞE HİTABE

Ey Türk gençliği! Birinci vazifen, Türk istiklâlini, Türk Cumhuriyetini, ilelebet muhafaza ve müdafaa etmektir.

Mevcudiyetinin ve istikbalinin yegâne temeli budur. Bu temel, senin en kıymetli hazinendir. İstikbalde dahi, seni bu hazineden mahrum etmek isteyecek dâhilî ve hâricî bedhahların olacaktır. Bir gün, istiklâl ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için, içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin! Bu imkân ve şerait, çok namüsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elîm ve daha vahim olmak üzere, memleketin dâhilinde iktidara sahip olanlar gaflet ve dalâlet ve hattâ hıyanet içinde bulunabilirler. Hattâ bu iktidar sahipleri şahsî menfaatlerini, müstevlîlerin siyasî emelleriyle tevhit edebilirler. Millet, fakr u zaruret içinde harap ve bîtap düşmüş olabilir.

Ey Türk istikbalinin evlâdı! İşte, bu ahval ve şerait içinde dahi vazifen, Türk istiklâl ve cumhuriyetini kurtarmaktır. Muhtaç olduğun kudret, damarlarındaki asil kanda mevcuttur.

Mustafa Kemal ATATÜRK



MUSTAFA KEMAL ATATÜRK

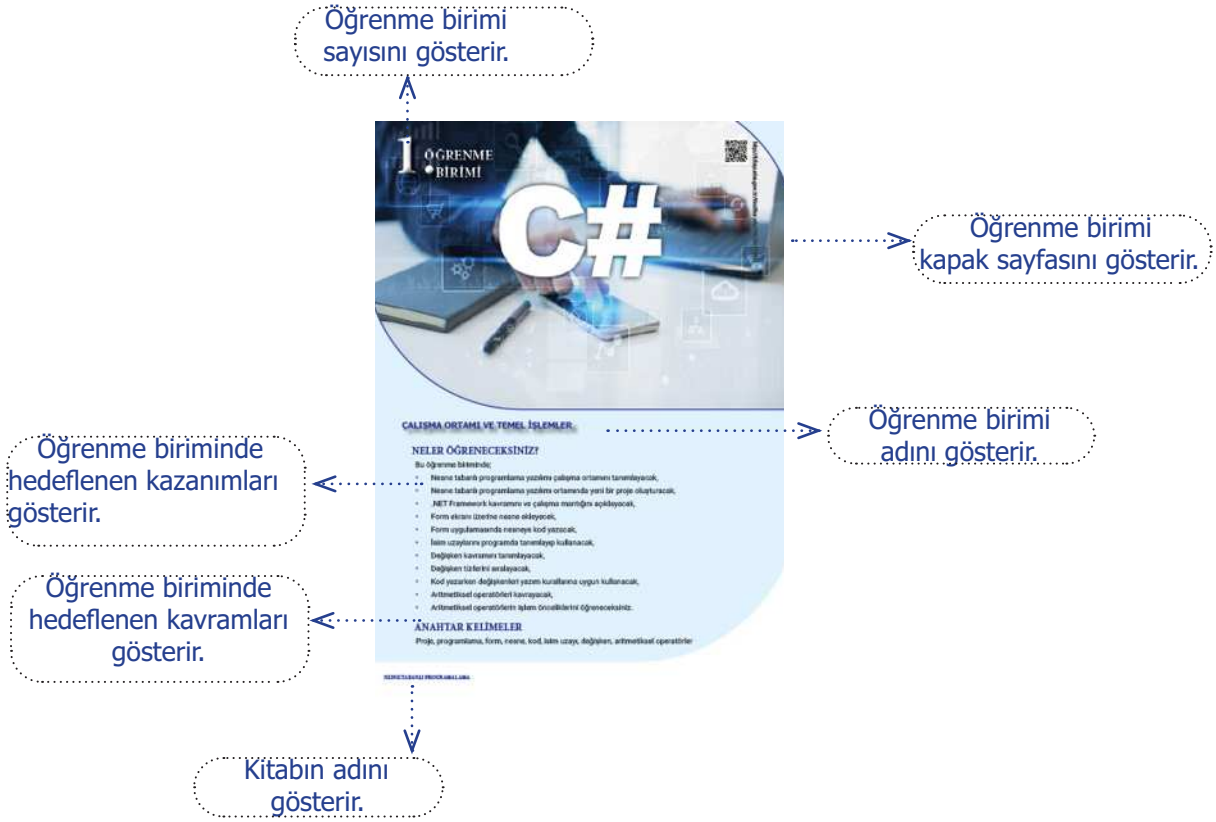
İÇİNDEKİLER

4. ÖĞRENME BİRİMİ

5. ÖĞRENME BİRİMİ

6. ÖĞRENME BİRİMİ

KİTAP TANITIMI

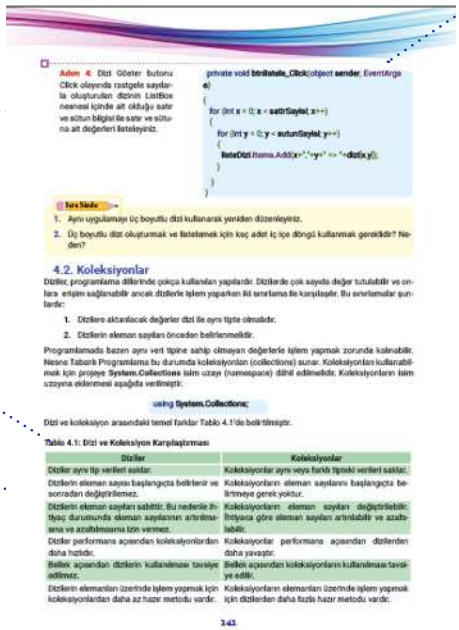


Öğrenme birimine ait görselin numarasını ve görselin ismini gösterir.

Öğrenme biriminin alt başlıklarını gösterir.

Kazanımları destekleyen konu metinlerini gösterir.

Programlama kodlarını gösterir.

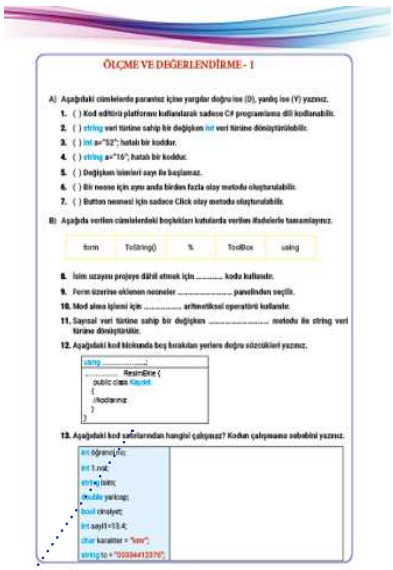


Konu içeriğini destekleyen ve konu ile ilgili bilgi veren tabloyu gösterir.

Konu içeriğini destekleyen ve konu ile ilgili bilgi veren tablonun numarasını ve başlığını gösterir.

Öğrenme biriminde dikkat edilmesi gereken önemli bilgileri gösterir.

Konu ile ilgili örnekleri gösterir.



Her öğrenme biriminin sonunda yer alan ölçme ve değerlendirme sorularını gösterir.

Öğrencilerin yapması istenen çalışmaları gösterir.

1 ÖĞRENME BİRİMİ

http://kitap.eba.gov.tr/KodSor.php?KOD=11895

ÇALIŞMA ORTAMI VE TEMEL İŞLEMLER

NELER ÖĞRENECEKSİNİZ?

Bu öğrenme biriminde;

- Nesne tabanlı programlama yazılımı çalışma ortamını tanımlayacak,
- Nesne tabanlı programlama yazılımı ortamında yeni bir proje oluşturacak,
- .NET Framework kavramını ve çalışma mantığını açıklayacak,
- Form ekranı üzerine nesne ekleyecek,
- Form uygulamasında nesneye kod yazacak,
- İsim uzaylarını programda tanımlayıp kullanacak,
- Değişken kavramını tanımlayacak,
- Değişken türlerini sıralayacak,
- Kod yazarken değişkenleri yazım kurallarına uygun kullanacak,
- Aritmetiksel operatörleri kavrayacak,
- Aritmetiksel operatörlerin işlem önceliklerini öğreneceksiniz.

ANAHTAR KELİMELER

Proje, programlama, form, nesne, kod, isim uzayı, değişken, aritmetiksel operatörler

HAZIRLIK ÇALIŞMALARI

1. Programlama dili size ne ifade ediyor?
2. Hangi programlar C# dili kullanılarak yapılmış olabilir?

1.1. Nesne Tabanlı Programlama Çalışma Ortamı

Program, herhangi bir elektronik cihaza bir işlem yaptırmak için yazılan komutlar dizisidir. Programlar; bilgisayar, cep telefonu, tablet, elektronik ev eşyaları, araba ve daha birçok yerde kullanılır.

Elektronik cihazlara bilgisayar, cep telefonu, akıllı saat, akıllı televizyon vb. örnek olarak verilebilir (Görsel 1.1). Program yazmak denilince akla ilk gelen bilgisayarlar olsa da günümüzde pek çok elektronik cihaza kod yazılabilir. Bilgisayar ve diğer elektronik cihazlar çalışma prensibi olarak programları kullanıp önce girilen bilgiyi alır, ardından bu bilgiyi işler ve en sonunda ortaya bir sonuç çıkarır. Bu sonuç bazen bir mesaj bazen de bir işlemi gerçekleştirme şeklindedir. Bu durum, insanların birbiriyle iletişimi gibi düşünülebilir. İnsanlar da konuşmalarında önce karşı tarafın ifadelerini algılar, algıladıklarını işler ve son olarak karşı tarafa bir cevap verir.

Görsel 1.1: Nesne tabanlı programlama çalışma ortamı

Programlama Temelleri dersinde de bahsedildiği gibi programlar önceden "1" ve "0" lardan oluşan kod blokları ile makine dilinde yazılırdı. Makine dilinde kod yazılması ve yazılan kodun anlaşılması oldukça zordu. Bu nedenle daha okunaklı ve kolay kod yazılabilen Assembly dilleri geliştirildi. İlerleyen zamanlarda daha çok anlaşılır, konuşma diline daha yakın ifadelerle kod yazma imkânı sağlayan C, C++, Turbo Pascal, Visual Basic, C#, Python gibi programlama dilleri kullanıldı.

Programlama; bilgisayar programlarının yazılması, test edilmesi ve bakımının yapılması sürecine verilen isimdir. Bu sürecin daha verimli geçmesi için amaca yönelik bir programlama dili seçilmelidir. Nesne Tabanlı Programlama ders kitabının içeriğinde C# programlama dilinin kullanımı anlatılacaktır.

1.2. C# Programlama Dili

C# programlama dili, nesne tabanlı olarak geliştirilmiş bir dildir. C# programlama dili ile;

- Mobil uygulamalar,
- Konsol uygulamaları,
- Web servisleri,
- Dinamik kütüphaneler (DLL),
- Oyun tasarımı,
- Form uygulamaları yapılabilir.

Günlük hayatın birçok alanında büyük küçük pek çok şirket C# ile geliştirilmiş programlar kullanmaktadır (Görsel 1.2). C# programlama dilinin çok tercih edilmesinin sebeplerinden bazıları şunlardır:

- Yazılması ve anlaşılması kolay kod yapısına sahiptir.
- Yeni teknolojileri destekler.
- Kullanışlıdır.
- Ekip çalışmasına elverişlidir.
- Kullanıcıyla etkileşimlidir.
- Grafik arayüzlü tasarımlar yapılabilir.
- Ağ üzerinden birbiriyle uyumlu çalışabilir.
- Çevrimiçi veya çevrimdışı kullanılabilir.
- QR kod okuyucu, kamera, yazıcı vb. cihazlarla etkileşimlidir.
- Verileri depolayıp işleyerek analiz yapabilir.
- Sosyal medya platformları ile etkileşimlidir.
- Cep telefonu uygulamaları ile etkileşimlidir.
- Yapay zekâ teknolojisi kullanılarak yüz tanıma, nesne tanıma, ses tanıma işlemleri yapılabilir.
- Birçok özelliğe sahip farklı programlar geliştirebilme imkânı sağlar.

Görsel 1.2: Program çalışma süreci

1.3. .NET Framework

Framework kelimesi **geliştirme çatısı** anlamına gelmektedir. .NET Framework, çoğu kişi tarafından bir programlama dili sanılmaktadır fakat programlama dillerinden bağımsız ve farklı programlama dilleri ile ortak çalışma imkânı sağlayan bir uygulama geliştirme platformudur. .NET logosu Görsel 1.3'te görülmektedir. Farklı dilleri bilen programcılar ortak bir projeyi yürütebilir. .NET Framework çalışma mantığı bunun için uygundur. Yaygın kullanılan çoğu dil (C#, Visual Basic, Visual C++, Visual F#, Python) .NET Framework desteklidir. Ayrıca .NET Framework altyapısında kullanıma sunulmuş hazır kod kütüphaneleri ile kod yazma çok daha hızlı ve verimli olmaktadır. .NET Framework kütüphaneleri tüm .NET dillerinde ortak kullanılır. Bu yüzden .NET çok güçlü bir Framework'tür.

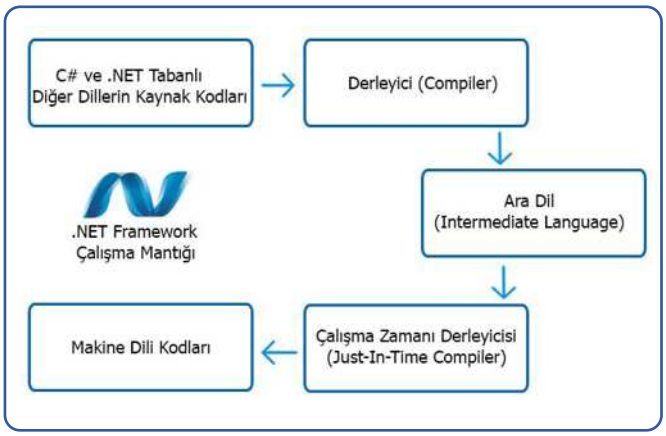
Görsel 1.3: .NET Framework

1.3.1. C# ve .NET Framework İlişkisi

C# bir programlama dilidir, .NET Framework ise C# dili ve birçok dilin kütüphanelerinin yüklü olduğu bir uygulama geliştirme platformudur. C# dilinde kullanılan kütüphanelerin tümü .NET Framework kütüphaneleridir.

1.3.2. .NET Framework Çalışma Mantığı

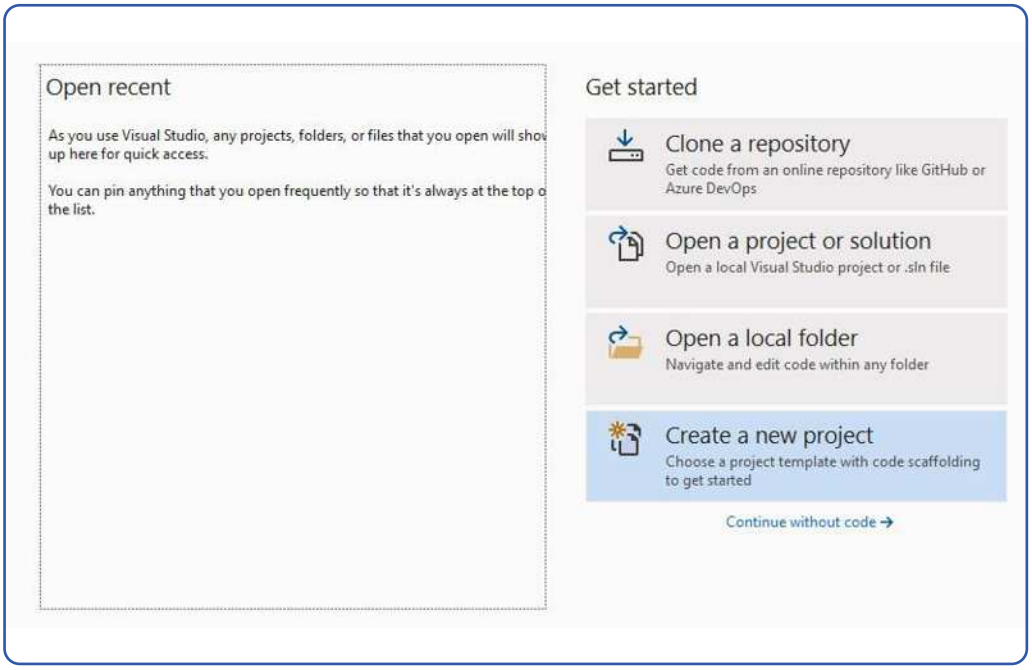
Programlama dilleri ile yazılan kodlar makine için anlamlı değildir, kodların makine dilinde yazılması veya makine diline çevrilmesi gerekir. Program yazarken kullanılan kodlar derlendiğinde doğrudan makine diline çevrilmez. Görsel 1.4'te görüldüğü gibi kodlar önce **ara dil** (Intermediate Language) koduna, ardından da **çalışma zamanı derleyicisi** (Just-In-Time Compiler) tarafından makine diline çevrilir. Böylelikle kodlar sorunsuz bir şekilde çalışır. Bu işlemlerin yapılmasını .NET Framework altyapısı sağlar.



Görsel 1.4: .NET Framework çalışma mantığı

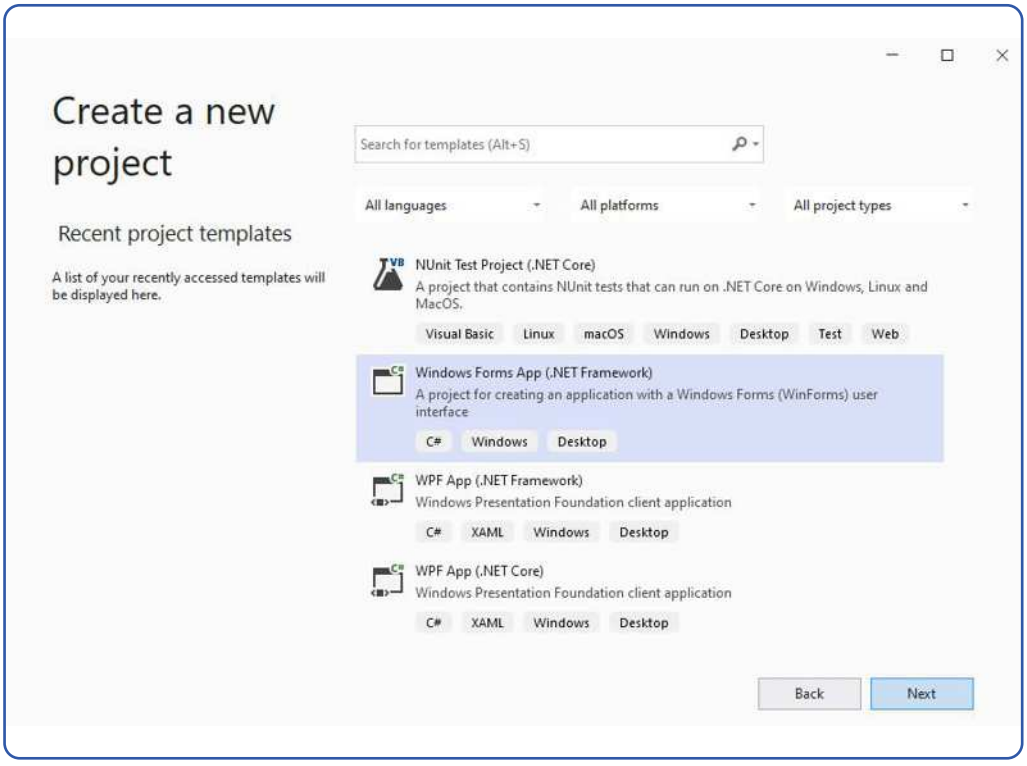
Uygulama-1

Adım 1: Kod editörü arayüzünü açarak Görsel 1.5'te görülen başlangıç ekranından Create a new project'i seçiniz ve yeni bir proje oluşturunuz.



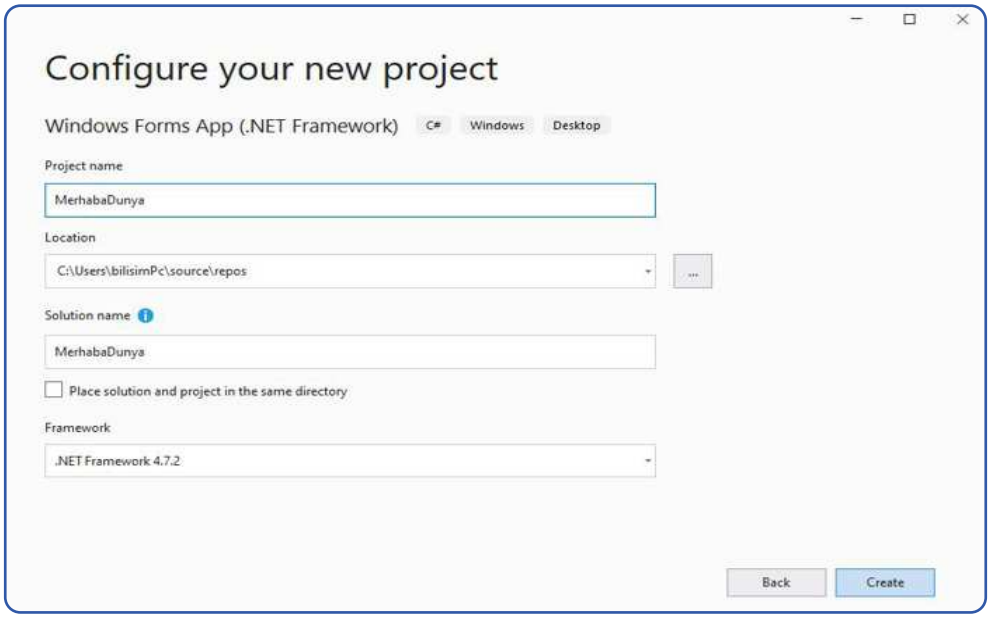
Görsel 1.5: Yeni proje oluşturma

Adım 2: Görsel 1.6'da görülen ekrandan Form Uygulamalarını (.NET Framework) seçiniz.



Görsel 1.6: Programlama dili seçimi

Adım 3: Projeye isim veriniz ve projenin kayıt yerini belirleyiniz (Görsel 1.7).



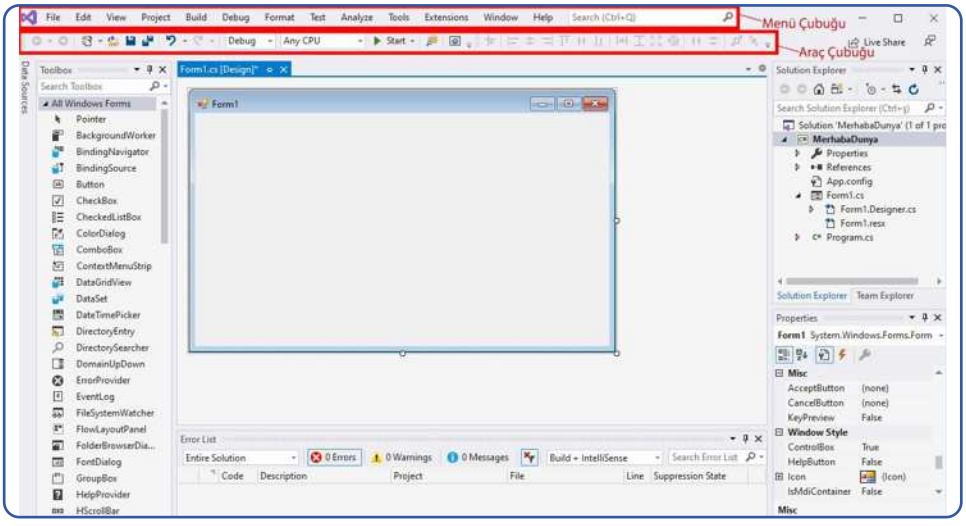
Görsel 1.7: Proje adı belirleme

Sıra Sizde

1. "MerhabaDunya" isimli bir Form uygulaması oluşturunuz.
2. Oluşturduğunuz "MerhabaDunya" isimli projeyi kaydedip kapattıktan sonra **var olan projeyi açma** seçeneği ile yeniden açınız.

1.4. Kod Editörü Arayüz Ekranı

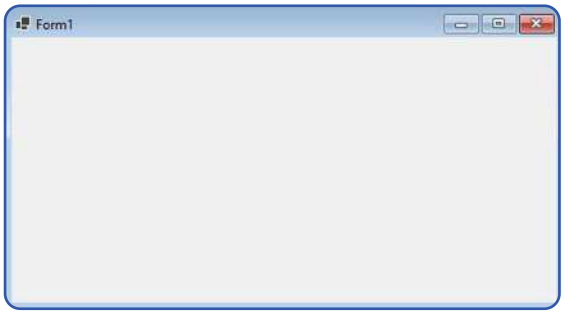
Görsel 1.8'de görüldüğü gibi kod editörü arayüz ekranı açıldığında ekranın üst tarafında **Menü Çubuğu** ve **Araç Çubuğu**, ekranın orta kısmında **Form Ekranı** ve form ekranının etrafında **4 adet panel** bulunmaktadır. Bu panellerin yerleri isteğe bağlı olarak sabitlenebilir, sürükle bırak yöntemi ile yerleri değiştirilebilir veya paneller tamamen kaldırılabilir.



Görsel 1.8: Kod editörü arayüz ekranı

1.4.1. Form Ekranı

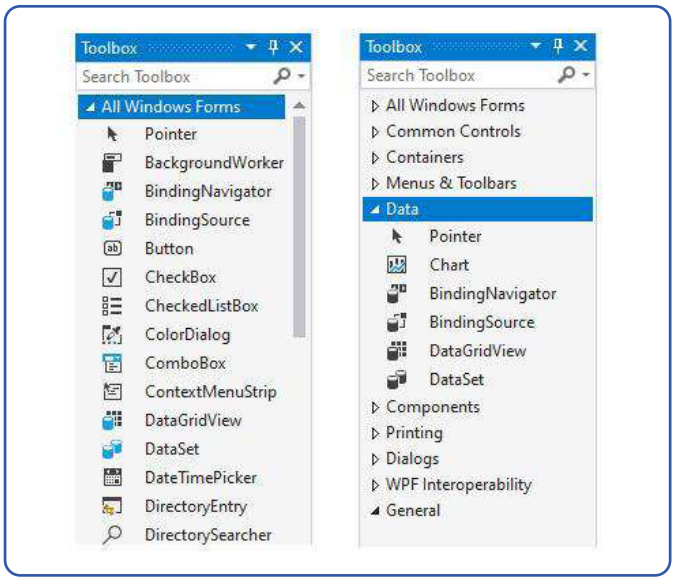
Form ekranı, programın görsel tasarımının yapıldığı yerdir (Görsel 1.9). **Araç Kutusu (Toolbox)** panelinde bulunan nesneler form üzerinde istenilen pozisyona yerleştirilerek programın tasarımı yapılır.



Görsel 1.9: Form ekranı

1.4.2. Araç Kutusu (Toolbox)

Form üzerinde tasarım için kullanılabilecek nesneler bu panelde bulunur. Görsel 1.10'da görüldüğü gibi tüm nesneler aynı anda veya işlevlerine göre çeşitli kategorilerde listelenebilir. Örneğin, Data bölümüne tıklandığında **Data (Veri)** ile ilgili nesneler listelenir. Ayrıca **Search Toolbox (Arama Çubuğu)** ile istenilen nesnenin ismi yazılarak da nesne listelenebilir.



Görsel 1.10: Araç kutusu

Toolbox paneli kullanılarak form üzerine nesne ekleme işlemi iki farklı şekilde yapılabilir:

1. Eklenecek nesne çift tıklanır.
2. Eklenecek nesne sürüklenip formun üzerinde herhangi bir pozisyona bırakılır.

Sıra Sizde

1. Toolbox panelinde **Label** ve **TextBox** nesnelerini çift tıklayarak bu nesneleri formun üzerine yerleştiriniz.
2. **Label** ve **TextBox** nesnelerini formun ortasına yerleştiriniz.

En Çok Tercih Edilen Toolbox Nesneleri

Toolbox'ta 50'den fazla nesne bulunmaktadır. Bu nesnelerin büyük çoğunluğu Görsel 1.11'de ve Görsel 1.12'de görülmektedir. Projelerde en çok kullanılan nesneler aşağıda listelenmiştir. Button: Programlarda bazı kodları çalıştırmak için kullanılan komut düğmeleridir. Button nesnesine tıklandığında içeri doğru basma efekti gerçekleştiği için tıklama (Click) olayları için vazgeçilmez bir nesnedir.

CheckBox: Kullanıcıya bir veya aynı anda birden çok seçeneği işaretleme imkânı sağlayan nesnedir.

ComboBox: Açılır liste ile açılan seçenekler arasından seçim yapılmasına olanak sağlayan araçtır. Listeye yeni eleman ekleme ve çıkarma işlemleri, tasarım ekranından veya program çalışırken kod ile yapılabilir.

DateTimePicker: Tarih ve saat seçme işlemlerine olanak sağlayan nesnedir.

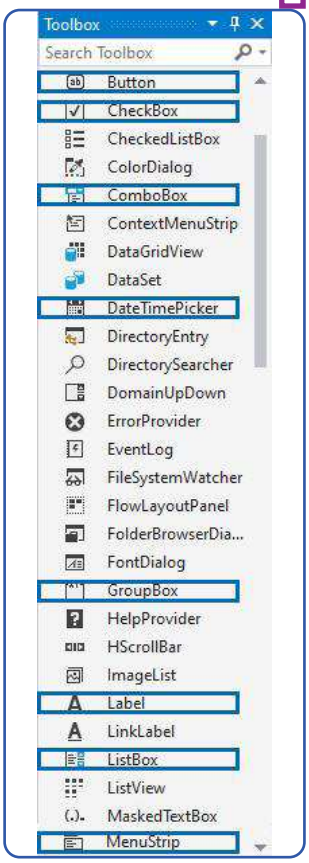
GroupBox: Form elemanlarını kendi aralarında gruplamak için kullanılan nesnedir.

Nesneler gruplar hâlinde olduğu için daha anlaşılır tasarımlar yapılabilir.

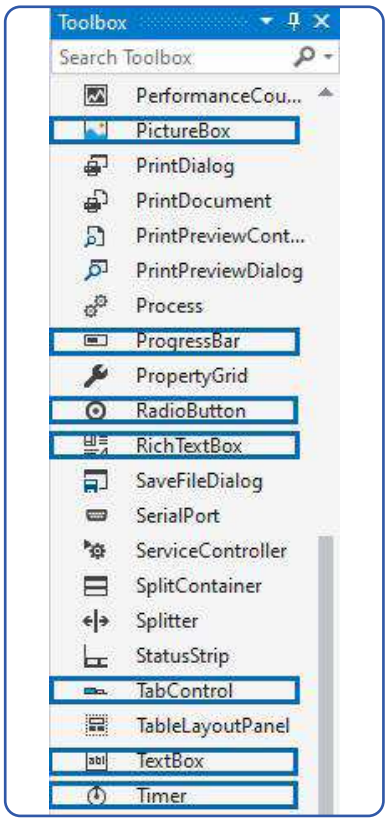
Label: Form üzerinde bilgi vermek için kullanılan nesnedir.

ListBox: Sunulacak seçeneklerin açık bir liste hâlinde gösterildiği nesnedir.

MenuStrip: Programda menü başlıkları ve alt başlıkları oluşturmak için kullanılan nesnedir.



Görsel 1.11: En çok tercih edilen nesneler-1



Görsel 1.12: En çok tercih edilen nesneler-2

PictureBox: Form üzerinde resim göstermek için kullanılan nesnedir.

ProgressBar: Yapılan bir işlemin ne kadarının tamamlandığını göstermek için kullanılan nesnedir.

RadioButton: CheckBox nesnesinden farklı olarak birçok seçenek içinden sadece birinin seçilmesine imkân sağlayan nesnedir.

RichTextBox: Birden çok satır içine metin girişi yapılabilen nesnedir.

TabControl: Form elemanlarının gruplara ayrıldığı, grupların içindeki elemanları görmek için sekmelerin kullanıldığı nesnedir. Her sekme, bir grubu temsil eder.

TextBox: İçine tek satır metin girişi yapılabilen nesnedir. Bilgi girişi için en çok tercih edilen nesnedir.

Timer: Kodların zamanlanarak çalışmasını sağlayan nesnedir.

Sıra Sizde

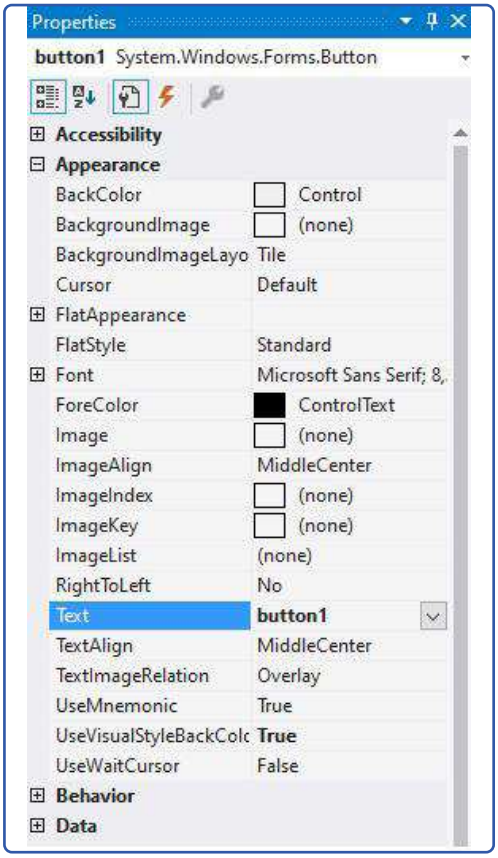
En çok tercih edilen nesnelerin özelliklerini araştırınız ve bulduğunuz özellikleri uygulama içinde kullanınız. Çalışmanızı yaparken öğretmeninizden destek alınız.

1.4.3. Özellikler (Properties)

Form nesnesinin ve diğer tüm nesnelerin özelliklerinin listelendiği, değiştirildiği ve ayrıca nesnelere ait **olayların (events)** listelendiği paneldir.

Her nesnenin kendine ait benzersiz özellikleri olduğu gibi diğer nesneler ile ortak özellikleri de bulunmaktadır. Tüm nesnelerin en temel ortak özelliği, isim (name) özelliğidir. Örneğin, PictureBox nesnesinin genişlik ve yükseklik özelliği vardır fakat CheckBox nesnesinin yoktur. Button nesnesinin yazı rengi özelliği vardır fakat PictureBox nesnesinin yoktur. DateTimePicker nesnesinin tarih belirleme özelliği vardır fakat TextBox nesnesinin yoktur.

Görsel1.13'te seçili button1 nesnesine ait bazı özellikle aşağıda verilmiştir.



Görsel 1.13: Özellikler sekmesi

BackColor: Nesnenin arka plan rengini değiştirir.

BackgroundImage: Nesnenin arka planına resim ekler.

Font: Nesnenin yazı tipini, boyutunu ve kalınlığını değiştirir.

ForeColor: Nesnenin yazı rengini değiştirir.

Text: Nesnenin yazı metnini değiştirir.

TextAlign: Nesnenin yazısını hizalar.

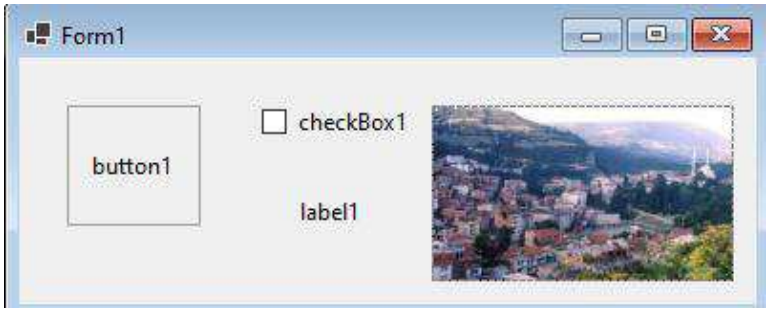
Sıra Sizde

Sizler de form üzerine 2 adet farklı nesne ekleyip her nesnenin en az 7 özelliğini değiştiriniz.

Uygulama-2

Adım 1: İki kişilik gruplar hâlinde eşleşiniz. Yanınızdaki veya öğretmeninizin belirlediği bir arkadaşınız ile grup oluşturabilirsiniz.

Adım 2: Görsel 1.14'te görüldüğü gibi Form üzerine Button, CheckBox, Label ve PictureBox nesnelerini ekleyiniz. Toolbox panelinden sürükle bırak yöntemi ile nesneleri form üzerine yerleştiriniz.



Görsel 1.14: Özellikler paneli uygulama-1

Adım 3: Bu nesnelerin ortak ve farklı özelliklerini belirleyip defterinize not alınız. Form üzerindeki nesnelerin hepsi fare ile aynı anda seçildiğinde properties panelinde nesnelerin sadece ortak özellikleri listelenmektedir.

1.4.4. Olaylar (Events)

Her nesnenin form üzerinde bir görevi bulunmaktadır. Bazı nesneler sadece programın işlevi ile ilgili bilgiyi ve görseli yansıtmak için kullanılır. Bazı nesneler ise belli durumlarda (üzerine tıklandığında, bir tuşa basıldığında vs.) kod parçacıklarını çalıştırmak için kullanılır. Nesneler, kullanıcı ile etkileşimi olaylar sayesinde sağlamaktadır. Günlük hayatta sosyal medya uygulamalarında, web sitelerinde, oyunlarda ve daha birçok alanda nesnelere tanımlanmış olay metotları çalışarak kullanıcıyla etkileşim sağlanır. Olay metotları tanımlanarak nesnelerin hangi durumda, nasıl kodlar çalıştırabileceği belirlenir. Örneğin; butona tıklandığında şifre kontrolünün yapılması, klavyeden sağ ok tuşuna basıldığında bir sonraki resmin gösterilmesi, PictureBox nesnesinin üzerine çift tıklama yapıldığında resme ait ilgili bilgilerin MessageBox ile göstermesi vb. Olaylar için metot tanımlanırken istenilen olayın adının hemen yanındaki boş beyaz kutucuğa çift tıklanır ve otomatik olarak aşağıdaki gibi bir metot oluşturulur.

```
private void button1_Click(object sender, EventArgs e)
    { listBox1.Items.Add(textBox1.Text); }
```

Görsel 1.15'te seçili **button1** nesnesine ait bazı olaylar aşağıda verilmiştir.

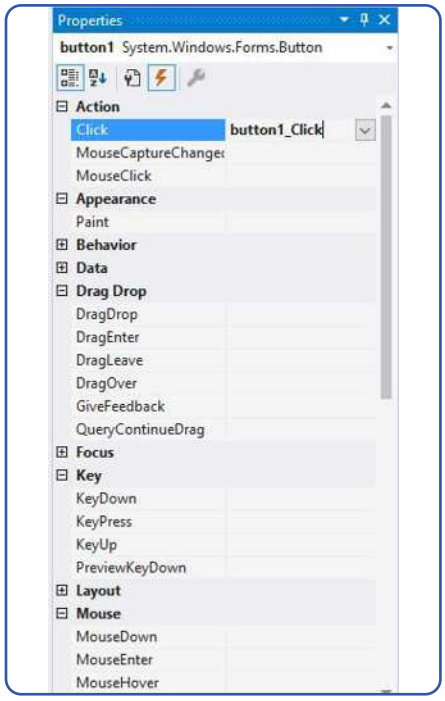
Click: Nesneye fare ile tıklanması veya fare nesne üzerindeyken enter tuşuna basılmasıyla devreye giren olaydır.

MouseClick: Click gibi nesnenin fare ile tıklanması olayıdır fakat MouseClick enter tuşundan etkilenmez. Ayrıca MouseClick, fareye ait koordinatlar gibi özel bilgileri de verir.

KeyDown: Klavyeden bir tuşa basılması olayıdır.

KeyUp: Klavyede basılan tuşun bırakılması olayıdır.

MouseDown: Farenin tuşuna basılması olayıdır.



Görsel 1.15: Olaylar sekmesi

Sıra Sizde

Form nesnesinin Load, Enter, Click olaylarına ait metotları oluşturunuz.

Uygulama-3

Adım 1: Form üzerine TextBox ve Button nesnelerini ekleyiniz.

Adım 2: Eklediğiniz nesnelerin özeliklerini Görsel 1.16'daki gibi tasarıma uygun hâle getiriniz.

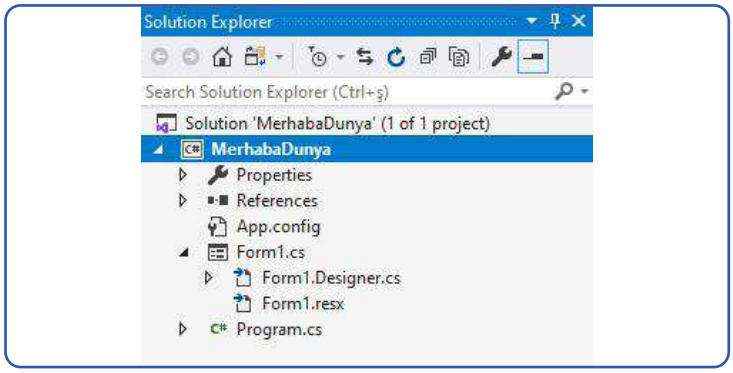
Adım 3: Form üzerindeki nesnelerin ortak olan olaylarını belirleyip sınıf arkadaşlarınızla paylaşınız. Form üzerindeki nesnelerin hepsi fare ile aynı anda seçildiğinde properties panelinde nesnelerin sadece ortak olayları listelenmektedir.

Görsel 1.16: Özellikler paneli uygulama-2

1.4.5. Çözüm Penceresi (Solution Explorer)

Proje ile ilgili tüm dosya ve klasörlerin listelenerek silme, kopyalama, taşıma, isim değiştirme işlemlerinin yapılabildiği paneldir (Görsel 1.17). Projenin detaylı bir haritası gibi düşünülebilir. Çözüm penceresi ile projeye yeni sınıf (class), form ve başka ögeler eklenebilir. Veri tabanı dosyası, resim, müzik, video dosyaları projeye dâhil edilebilir.

Projeye eklenen her form, çözüm penceresinde ayrı ayrı listelenmektedir. Listedeki Form1.cs ifadesine fare sağ tuşu ile tıklandığında açılan listeden View Code tıklanır ise Form1'e ait kod ekranı, View Designer tıklanır ise Form1'e ait tasarım ekranı açılır. Doğrudan Form1.cs ifadesine çift tıklanır ise tasarım ekranı açılır.



Görsel 1.17: Çözüm penceresi

Kod veya tasarım ekranını göstermenin diğer yolları şunlardır:

1. Form ekranı üzerinde herhangi bir noktada fare sağ tuşu kullanılarak açılan listede **View Code (Kodu Göster)** tıklanarak kod ekranı, **View Designer (Tasarımı Göster)** tıklanarak tasarım ekranı görüntülenir.
2. Klavyeden **F7** tuşuna basıldığında kod ekranı açılır. Klavyeden **Shift+F7** tuşuna basıldığında tasarım ekranı açılır.

1.4.6. Hata Listesi (Error List)

Kod yazarken, kod derlenirken veya kod çalışırken oluşan hataların ve uyarıların listelendiği Görsel 1.19'da görülen paneldir.

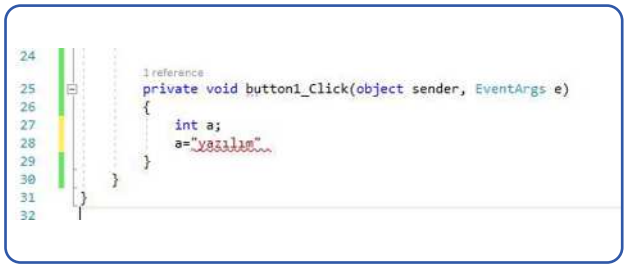
a) Errors (Hatalar) Bölümü: Çok kritik ve programın çalışmasını engelleyen hatalardır. Örneğin;

değişkeni tanımlamadan bir kod bloku içinde kullanmak, kod satırının sonunda noktalı virgül koymamak, kod yazarken açılan parantezin kapatılmaması vb.

b) Warnings (Uyarılar) Bölümü: Programın çalışmasını engellemeyecek düzeydeki iletilerdir.

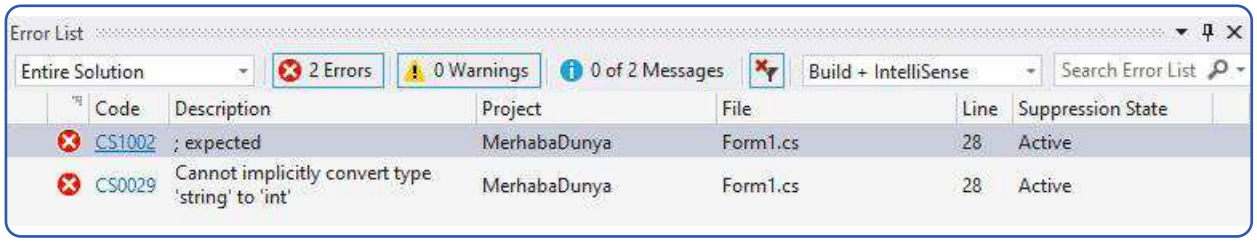
Örneğin, değişkenin tanımlanıp hiçbir zaman kullanılmaması vb.

Görsel 1.18'de hatalı bir kod bloku verilmiştir.



Görsel 1.18: Hatalı kod bloku

Birinci hata, 28. satırda bulunan kodun sonunda noktalı virgül olmamasıdır. İkinci hata ise 28. satırdaki kodda bulunan "a" isimli değişkenin tipi sayısal olmasına rağmen değişkene metinsel ifade atanmış olmasıdır. Hatalar kritik seviyede olduğu için program çalışmaz. Görsel 1.19'da görüldüğü gibi program derleyicisi bu iki hatayı yakalar ve Error List panelinde listeler.



Görsel 1.19: Hata listesi

Uygulama-4

Adım 1: Form üzerine Görsel 1.20'de görüldüğü gibi bir tane Button nesnesi ekleyiniz.

Adım 2: Özellikler penceresinden formun Text özelliğini "Merhaba Dünya" yapınız.

Adım 3: Özellikler listesinden button1'in Text özelliğini mesaj göster yapınız.

Adım 4: Olaylar listesinden button1 için Click olayı metodunu button1'in üzerine çift tıklayarak oluşturunuz.

Adım 5: Oluşturduğunuz metodun içine MessageBox.Show ("Merhaba Dünya"); kodunu yazınız.

Adım 6: Araç Çubuğunda bulunan ▶ Başlat butonu ile programı çalıştırınız.

Görsel 1.20: Merhaba Dünya

```
private void button1_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Merhaba Dünya");
}
```

Örnek uygulamada en önemli nokta, Click olayının metodunu oluşturmaktır. Click, programcıların en çok kullandığı olaydır ve bu olay için metot oluşturma işlemi en pratik şekilde seçili nesneye çift tıklanarak gerçekleştirilir. Uygulamada diğer olaylar için metot tanımlama işlemi istenilseydi Görsel 1.15'teki gibi button1'e ait olayların listesinden istenilen olayın adının hemen yanındaki boş kutucuğa çift tıklanarak metot oluşturulurdu.

Not: MessageBox adından da anlaşılacağı gibi ekrana mesaj verdiren sınıfın adıdır. Programcılar tarafından çok fazla kullanılır. Farklı kullanım şekilleri vardır. En temel kullanımı, örnek kodda verildiği gibi sadece tek bir mesajı gösterme şeklindedir. MessageBox sınıfının bazı kullanım şekilleri aşağıda verilmiştir.

1. MessageBox.Show("mesaj metni","mesaj başlığı");
2. MessageBox.Show("mesaj metni","mesaj başlığı",MessageBoxButtons.YesNoCancel);
3. MessageBox.Show("mesaj metni","mesaj başlığı",MessageBoxButtons.OKCancel);

Sıra Sizde

MessageBox sınıfının farklı kullanım şekilleri neler olabilir? Araştırıp bulduklarınızı program içinde kullanınız ve arkadaşlarınızla paylaşınız.

Uygulama-5

Adım 1: Form üzerine Görsel 1.21'de görüldüğü gibi birer tane Button ve TextBox nesnesi ekleyiniz.

Adım 2: Button1'e tıklandığında textBox1 nesnesinin içine Hello World yazdırınız.

Görsel 1.21: Hello World mesaj uygulaması

```
private void button1_Click(object sender, EventArgs e)
{
textBox1.Text = "Hello World";
}
```

Yukarıdaki örnekte mesaj, bir önceki uygulamadan farklı olarak TextBox nesnesi üzerinden verilmiştir. **textBox1.Text="Hello World";** kod parçasına bakılırsa nesnenin bir özelliğine kod ile müdahale edildiği görülür. Nesnelerin özellikleri sadece Properties panelinden değiştirilmez. Programın çalışma zamanında kodla da değiştirilebilir. Örnek kodlar aşağıda verilmiştir.

```
textBox1.ForeColor = Color.Red; //Yazı rengini kırmızı yapar.
textBox1.Enabled = false;  //Nesneyi pasifleştirir. Artık metin girişi yapılamaz.
textBox1.Visible = false;  //Nesneyi görünmez hâle getirir.
textBox1.Font    = new Font("Broadway", 16);  //Yazı tipi ve boyutu değişir.
```

Uygulama-6

Adım 1: Form üzerine Görsel 1.22'de görüldüğü gibi iki tane GroupBox nesnesi ekleyiniz.

Adım 2: Üst tarafta bulunan GroupBox nesnesinin içine bir tane Label nesnesi ekleyiniz.

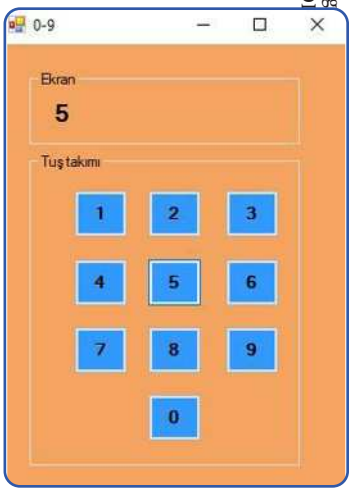
Adım 3: Alt tarafta bulunan GroupBox nesnesinin içine on tane Button nesnesi ekleyiniz ve bunları numaralandırınız.

Adım 4: Numaraların yazılı olduğu Button nesnelerinin arka plan rengini "MenuHighlight" yapınız.

Adım 5: Button nesnelerinin ve Label nesnesinin yazı tipi stilini "kalın", yazı boyutunu "10" yapınız.

Adım 6: Form nesnesinin başlığını "0-9" ve Form nesnesinin arka plan rengini "SandyBrown" yapınız.

Adım 7:Tıklanan Button nesnesine ait sayıyı Label nesnesinin Text özelliğine aktaran programı yazınız.



Görsel 1.22: Tuş takımı

```
private void button1_Click(object sender, EventArgs e)
    {     label1.Text = "1";     }

private void button2_Click(object sender, EventArgs e)
    {     label1.Text = "2";     }

private void button3_Click(object sender, EventArgs e)
    {     label1.Text = "3";     }

private void button4_Click(object sender, EventArgs e)
    {     label1.Text = "4";     }

private void button5_Click(object sender, EventArgs e)
    {     label1.Text = "5";     }

private void button6_Click(object sender, EventArgs e)
    {     label1.Text = "6";     }

private void button7_Click(object sender, EventArgs e)
    {     label1.Text = "7";     }

private void button8_Click(object sender, EventArgs e)
    {     label1.Text = "8";     }

private void button9_Click(object sender, EventArgs e)
    {     label1.Text = "9";     }

private void button10_Click(object sender, EventArgs e)
    {     label1.Text = "0";
```

Sıra Sizde

1. Form üzerine 2 adet **Button** nesnesi ve 1 adet **PictureBox** nesnesi ekleyiniz.
2. Button1'e tıklandığında PictureBox nesnesini görünmez hâle getiren, button2'ye tıklandığında PictureBox nesnesini görünür hâle getiren programı yazınız.

Uygulama-7

Sepete Ekle butonuna tıklandığında TextBox nesnesindeki değeri ListBox nesnesine aktaran Görsel 1.23'teki gibi bir tasarıma sahip programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
    {
        listBox1.Items.Add(textBox1.Text);
    }
```

Görsel 1.23: Manav sepeti uygulaması

Sıra Sizde

1. Görsel 1.23'teki program tasarımına **Sepeti Temizle** butonu ekleyiniz.
2. Yeni eklediğiniz butona tıklandığında ListBox nesnesinin içindekileri tamamen temizleyen programı yazınız.

Sıra Sizde

1. Form üzerine 2 adet **Button** nesnesi ekleyiniz.
2. Button1'e tıklandığında **"Button1'e tıkladınız.",** button2'ye tıklandığında **"Button2'ye tıkladı-**
3. **nız."** mesajlarını verdiriniz.
4. Oluşturduğunuz kodları aşağıdaki kutucuğa yazınız.

Uygulama-8

Not: Bir nesne için sadece bir tane olay metodu oluşturulmaz. Nesne için tanımlanabilecek ne kadar olay varsa o kadar da olay metodu oluşturulabilir.

Mouse simgesi ile forma eklenen bir Button nesnesinin üzerine gelindiğinde **"Mouse şimdi üzerimdedir."** mesajını, mouse simgesi Button nesnesinin üzerinden kaldırıldığında **"Mouse artık üzerimde değildir."** mesajını verdiriniz.

```
private void button1_MouseMove(object sender, EventArgs e)
    {
      MessageBox.Show ("Mouse Üzerimdedir.");
    }
private void button1_MouseLeave(object sender, EventArgs e)
    {
      MessageBox.Show ("Mouse Artık Üzerimde Değildir.");
    }
```

Sıra Sizde

1. Form üzerine 3 adet **Button** nesnesi ekleyiniz.
2. Form nesnesinin arka plan rengini turuncu yapınız.
3. Button'ların hepsinin arka plan ve yazı renklerini birbiriyle uyumlu olacak şekilde tasarlayınız.
4.
   - Mouse simgesi ile button1'in üzerine gelindiğinde **"Mouse benim üzerimdedir."**,
   - mouse simgesi ile button2'nin üzerine çift tıklandığında **"Mouse iki kere tıklandı."**,
   - button3'ün üzerinden mouse simgesi kaldırıldığında **"Mouse üzerinde değil."**

   mesajlarını verdiğiniz ve oluşturduğunuz kodları aşağıdaki kutucuğa yazınız.

Sıra Sizde

Bir Button nesnesine fare ile tıklandığında kaç farklı olay gerçekleşir? Araştırınız, gerçekleşen olayları not ediniz ve notlarınızı arkadaşlarınızla paylaşınız.

Sıra Sizde

Bir TextBox nesnesine yazı yazıldığı andan itibaren kaç farklı olay gerçekleşir? Oluşturacağınız küçük gruplarla araştırınız, sonuçlarınızı diğer gruplarla karşılaştırınız.

Uygulama-9

Adım 1: Form üzerine Görsel 1.24'te görüldüğü gibi GroupBox nesnesi içine 4 adet Button nesnesi ekleyiniz.

Adım 2: Form nesnesinin arka plan rengini beyaz yapınız.

Adım 3: Kırmızı yazılı Button nesnesine tıklandığında Form nesnesinin arka plan rengini kırmızı, yeşil yazılı Button nesnesine tıklandığında Form nesnesinin arka plan rengini yeşil, mavi yazılı Button nesnesine tıklandığında Form nesnesinin arka plan rengini mavi, gri yazılı Button nesnesine tıklandığında Form nesnesinin arka plan rengini gri yapan programı yazınız.

Görsel 1.24: Form boyama

```
private void button1_Click(object sender, EventArgs e)
    {  this.BackColor = Color.Red;  }
private void button2_Click(object sender, EventArgs e)
    {  this.BackColor = Color.Green;  }
private void button3_Click(object sender, EventArgs e)
    {  this.BackColor = Color.Blue;  }
private void button4_Click(object sender, EventArgs e)
    {  this.BackColor = Color.Gray;  }
```

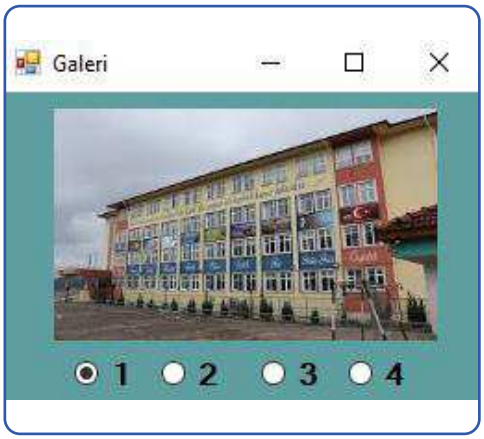
Uygulama-10

Adım 1: Form üzerine Görsel 1.25'te görüldüğü gibi dört tane PictureBox nesnesini üst üste olacak şekilde yerleştiriniz.

Adım 2: PictureBox nesnelerinin "Visible" özelliğini "false" yapınız (Visible özelliği ile nesnenin görünürlüğü belirlenir.).

Adım 3: Dört tane RadioButton nesnesi ekleyiniz ve bunları numaralandırınız.

Adım 4: radioButton1 nesnesi işaretli ise sadece pictureBox1 nesnesini görünür hâle getirecek, radioButton2 nesnesi işaretli ise sadece pictureBox2 nesnesini görünür hâle getirecek, radioButton3 nesnesi işaretli ise sadece pictureBox3 nesnesini görünür hâle getirecek, radioButton4 nesnesi işaretli ise sadece pictureBox4 nesnesini görünür hâle getirecektir.



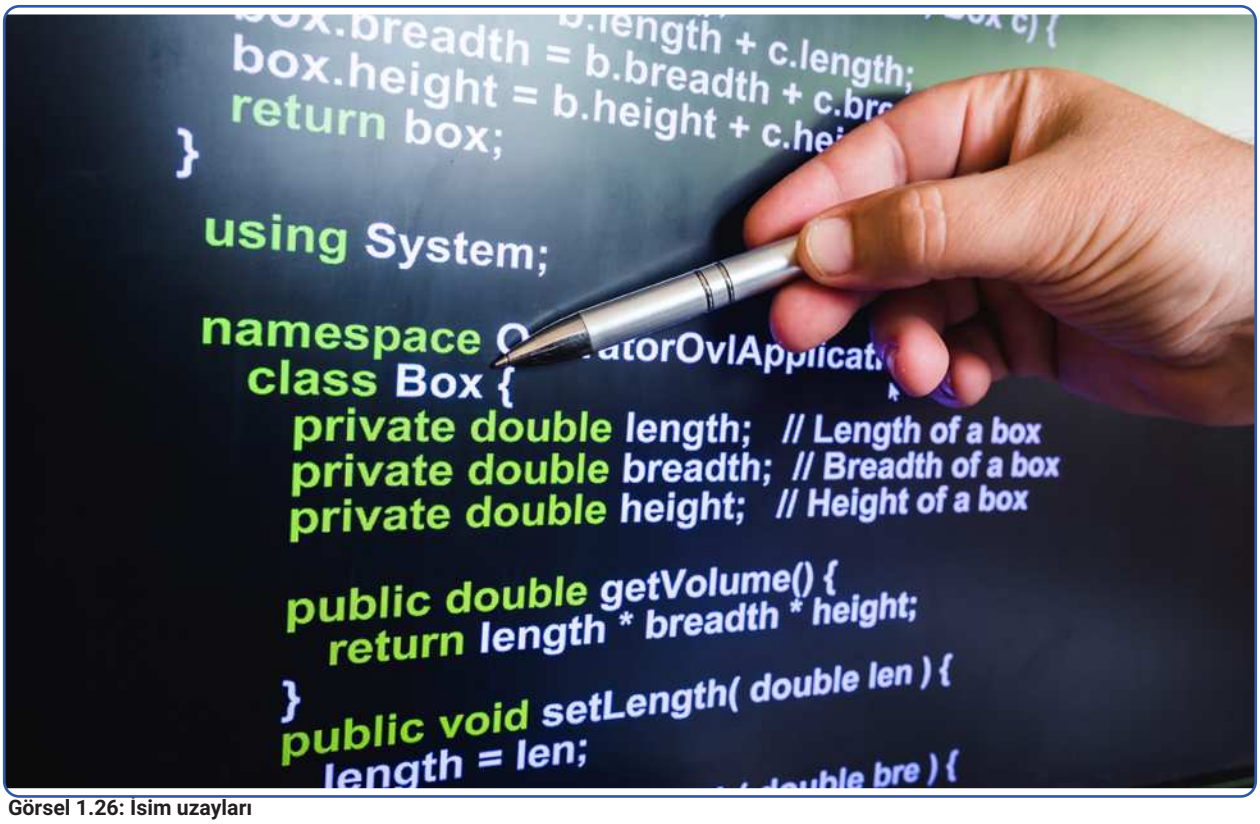
Görsel 1.25: Resim galerisif

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
    { pictureBox1.Visible = radioButton1.Checked; }
private void radioButton2_CheckedChanged(object sender, EventArgs e)
    { pictureBox2.Visible = radioButton2.Checked; }
private void radioButton3_CheckedChanged(object sender, EventArgs e)
    { pictureBox3.Visible = radioButton3.Checked; }
private void radioButton4_CheckedChanged(object sender, EventArgs e)
    { pictureBox4.Visible = radioButton4.Checked; }
```

Not: RadioButton nesnesi işaretlenince veya RadioButton nesnesinin işareti kaldırılınca CheckedChanged olayı devreye girer.

1.5. İsim Uzayları (Namespace)

İsim uzayları; program yazımı esnasında kullanılan metot, sınıf, değişken, sabit gibi yapıları mantıksal olarak kategorize etme sistemidir. İsim uzayları, proje büyüdükçe kod yapısının karmaşıklığını engeller. Örneğin, aile fotoğraflarına bakmak isteyen biri sabit diskindeki dosyaların hepsini tek klasörde tutuyorsa aradığı fotoğrafa ulaşması çok zor ve karmaşık olacaktır. Dosyalarını müzik, video, fotoğraf klasörleri şeklinde düzenleyip hatta fotoğraflar klasörünün içine de aile, okul, doğa diye ayrı ayrı kategorilerde klasörler oluşturursa karmaşa ortadan kalkacaktır (Görsel 1.26).



Görsel 1.26: İsim uzayları

.NET Framework ile gelen standart isim uzayları kullanılabileceği gibi proje yazımı esnasında sonradan oluşturulan isim uzayları da kullanılabilir.

İsim uzayını projeye dâhil etmek için **using** kodu kullanılır. Varsayılan olarak **using** System; kod parçacığı projelerde bulunur. Bu kod parçacığındaki using ifadesi "kullanılıyor" anlamına gelir, System ifadesi ise kullanılan isim uzayını temsil eder.

```
using System.IO;
using System.Data;
using System.Windows.Forms;
```

Kodlara yukarıdaki gibi isim uzayları eklenebilir.

Not: Bu kütüphanelerin hepsi System içinde mevcuttur. Bu kütüphaneler eklendiğinde kodlar daha kısa yazılır. Örneğin "System.IO.File.WriteAllText" diye bir kod kullanılacaksa using System.IO;" isim uzayı eklendiği için kodu sadece "File.WriteAllText" şeklinde yazmak yeterli olacaktır.

Programda kişisel bir isim uzayı tanımlanarak kodların organize olması sağlanabilir.

Veri tabanına kayıt yapma işlemlerini içeren bir kütüphane geliştirilirken Kaydet isimli bir sınıf var ise bu kodlara ait isim uzayını tanımlamak için "namespace" anahtar sözcüğünün kullanımı aşağıdaki örnekte verilmiştir.

```
namespace  VeriTabani {
  public class Kaydet
  {
  //kodlarınız
  }
}
```

Yukarıdaki Kaydet sınıfını kullanabilmek için programa using ifadesi ile isim uzayı eklenmelidir. Bu işlem aşağıda verilmiştir.

```
using VeriTabani
```

Sıra Sizde

Siz de isim uzayı yazma ve projeye dâhil etme kurallarına dikkat ederek kendi isim uzaylarınızı tanımlayıp projenize ekleyiniz.

1.6. Değişkenler ve Temel Veri Türleri

Değişkenler, programlamanın en temel kavramıdır. Program çalışması sırasında çeşitli türlerde verileri hafızada saklayan değişkenler, ihtiyaca göre tekrar tekrar kullanılan veri tutuculardır. Program çalıştığı sürece değişkenler RAM bellekte bulunur, program durdurulduğunda RAM bellekten silinir.

Değişkenler; aynı anda içinde sadece tek ürün taşınabilen, farklı taşıma kapasitelerine sahip ve üzerlerinde seri numaraları yazılı **alışveriş sepetleri** gibi düşünülebilir (Görsel 1.27). Bu sepetlerden seçilen herhangi biri ile ihtiyaca göre ürün taşınabilir. Taşınacak ürünün boyutuna ve ağırlığına en uygun sepet seçimi yapılmalıdır. Büyük ebatta kutuya sahip bir ürün taşınacaksa büyük bir sepet seçilmeli, küçük ebatlı kutuya sahip bir ürün taşınacaksa çok büyük bir sepet seçilmemeli ki en verimli şekilde taşıma işlemi yapılabilsin. Aynı seri numaralı sepetle gün içinde farklı ürünler de taşınabilir. Sepetin içeriği her taşımada sürekli değişebilir. Örneğin bir saat önce SP00052 seri numaralı sepete bakıldığında sepetin içinde parfüm olduğu görülüyorken on dakika önce muz, şimdi ekmek, sonrasında saatlerce sepetin boş bir şekilde kaldığı da görülebilir.

Yukarıdaki örnekte sepetin boy ve ağırlık kapasitesi, **değişkenin veri türü**; "SP00052" ifadesi, **değişkenin ismi;** sepetin içindeki parfüm ise **değişkenin değeri** olarak düşünülebilir.

Görsel 1.27: Değişkenler

1.6.1. Temel Veri Türleri

Değişkenlerin içine aktarılacak verilerin türleri çeşitlilik gösterebilir. Her veri türünün RAM bellekte kapladığı alan ve değişkenin türüne göre verinin değer aralığı farklıdır. Tablo 1.1, Tablo 1.2, Tablo 1.3 ve Tablo 1.4'te değişkenlerin veri türlerine göre kapladığı alan ve değer aralığı verilmiştir.

Değişkenler, RAM bellekte yer kaplar ve programda kullanılan değişken sayısı arttıkça bu durum RAM belleğin kullanılabilir hafıza kapasitesini düşürür. Örneğin öğrenci notlarının girileceği bir değişkenin veri türü, değer aralığı 100 sayısına en yakın olan **byte** veya **sbyte** olarak tanımlanabilir. Değer aralığı 100 sayısından çok daha fazla olan **short**, **ushort** gibi veri türlerinde tanımlanmamalıdır. Bu, 1 kişinin 100 odalı bir evde yaşamayı seçmesi gibi çok verimsiz, masraflı ve gereksiz bir eylemdir.

Tablo 1.1: Tam Sayı Veri Türleri

Tam Sayı Türü	Kapladığı Alan	Değer Aralığı
byte	1 Bayt	0,...,255
sbyte	1 Bayt	-128,...,127
short	2 Bayt	-32768,...,32767
ushort	2 Bayt	0,...65535
int	4 Bayt	-2147483648,...,2147483647
uint	4 Bayt	0,...,4294967295
long	8 Bayt	-9223372036854775808
ulong	8 Bayt	0,...,18446744073709551615

Tablo 1.2: Ondalık Sayı Veri Türleri

Ondalık Sayı Türü	Kapladığı Alan	Değer Aralığı
float	4 Bayt	±1.5*10-45,...,±3.4*1038
double	8 Bayt	±5.0*10-324,...,±1.7*10308
decimal	16 Bayt	±1.5*10-28,...,±7.9*1028

Tablo 1.3: Metinsel Veri Türleri

Metinsel Veri Türü	Kapladığı Alan	Değer Aralığı
string	Sınırsız	Metinsel ifade tutar.
char	2 Bayt	Tek bir karakter tutar.

Tablo 1.4: Mantıksal Veri Türleri

Mantıksal Veri Türü	Kapladığı Alan	Değer Aralığı
bool	1 bit	True-False veya 1-0

1.6.2. Değişken Tanımlama

(Değişkenin Veri Türü) (Değişen Adı)

```
int sayi
double pi
string soru
char karakter
bool cevap
```

Not: Aynı veri türündeki değişkenler, aynı kod satırında ve aralarına virgül konularak tanımlanabilir.

Örnek:

```
int sayi,deger,sonuç;
string isim1,isim2,soyad;
```

Değişken isimleri belirli yazım standartlarına göre verilmelidir. Bu standartlara uymak zorunlu değildir fakat kodların okunabilirliği ve rahat anlaşılması açısından çok önemlidir. Programın tüm kodlarında aynı standarda göre değişken isimleri verilmelidir.

Programda en çok tercih edilen yazım standartları; camel case (deve gösterimi), snake case (yılan gösterimi), pascal case'dir.

a) Camel Case: Bu yazım standardına göre değişkenin ismindeki ilk kelimenin baş harfi küçük, diğer kelimelerin baş harfleri büyük olur.

Örnek: `string kullaniciAdi; int toplamHesapTutari;`

b) Snake Case: Bu yazım standardına göre değişkenin ismindeki kelimelerin arasına _ (alt çizgi) kullanılır.

Örnek: string kullanici_adi; int toplam_hesap_tutari; string boy_Uzunlugu; string urun_Fiyati;

c) Pascal Case: Bu yazım standardına göre değişkenin ismindeki tüm kelimelerin baş harfleri büyük olur.

Örnek: string KullaniciAdi; int ToplamHesapTutari; string BoyUzunlugu; string UrunFiyati;

1.6.3. Değişkene Değer Atama

(Değişkenin Adı) = (Değişkenin Değeri);

```
sayi = 52
PiSayisi = 3.14159
Soru = "Ülkemizde hangi deniz en kuzeydedir?"
karakter = 'A'
cevap=true
adi_soyadi = "Ali YOLCU";
sinav Not Ortalamasi = 76
```

Not: Değişken tanımlanırken de değer atama yapılabilir.

Örnek: int sayi=52;

1.6.4. Değişken İsimlendirme Kuralları

1. Değişken isimlerinde boşluk kullanılmaz. Boşluk yerine alt çizgi (_) kullanılabilir.

Örnek: string ad soyad; şeklinde kullanım yanlıştır.

string ad_soyad; veya string adsoyad; şeklinde kullanım doğrudur.

2. ?, !, :, %, +, -, . gibi özel karakterler kullanılmaz.

Örnek: string soru?; şeklinde kullanım yanlıştır.

string soru; şeklinde kullanım doğrudur.

3. Değişken isimleri sayı ile başlamaz.

Örnek: byte 1not; şeklinde kullanım yanlıştır.

byte not1; şeklinde kullanım doğrudur.

4. Değişken isimleri büyük ve küçük harfe duyarlıdır.

Örnek: ulong toplamTutar; şeklinde tanımlanan değişken ile ulong toplamtutar; veya ulong ToplamTUTAR şeklinde tanımlanan değişken aynı değildir.

5. Herhangi bir kodla aynı isimde değişken tanımlanamaz. Değişkenlerde if, else, random gibi programa ait ifadeler isim olarak kullanılmaz.

6. Zorunluluk yoktur fakat Türkçe karakterlerin (ç, ö, ü, ğ, ş vb.) kullanılması tavsiye edilmez.

Sıra Sizde

Aşağıdaki değişken tanımlamalarını doğru ya da yanlış olarak değerlendiriniz. Yanlış olan değişken tanımlamalarının sebebini açıklamalar bölümüne yazınız.

Değişken Adı	Doğru	Yanlış	Açıklamalar
Musteri_Tc			
Müsteri_adı			
1.isim			
toplam			
textBox			
Yüzde%18			
seri no			
Yuzde10			

1.6.5. Değişken Veri Türü Dönüştürme (Convert) İşlemleri

Değişkenlerin veri türlerini bazen değiştirmek gerekebilir. Sayısal bir ifade, bir nesnenin Text özelliğine aktarılmak istendiğinde program hata verecektir. İçeriği tamamen sayı olsa da metinsel bir ifadeyi sayısal veri türüne sahip bir değişkene aktarırken program yine hata verecektir. Bu tip durumlarda değişkenlerin veri türlerini dönüştürmek gerekir. Tür dönüşümünü sağlayacak hazır metotlar şunlardır:

ToString() >>>> Her türden değişkeni string türüne dönüştürür. ToString(), en sık kullanılan dönüştürme metodudur.

```
Convert.ToByte(metin)      >>>>>   Byte'a çevirir.
Convert.ToInt16(metin)     >>>>>   Short'a çevirir.
Convert.ToInt32(metin)     >>>>>   Int'e çevirir.
Convert.ToInt64(metin)     >>>>>   Long'a çevirir.
Convert.ToSingle(metin)    >>>>>   Float'a çevirir.
Convert.ToDouble(metin)    >>>>>   Double'a çevirir.
Convert.ToDecimal(metin)   >>>>>   Decimal'a çevirir.
Convert.ToChar(metin)      >>>>>   Char'a çevirir.
Convert.ToBoolean(metin)   >>>>>   Bool'a çevirir.
```

Örnek:

```
int sayi1=100;
string deger;
deger=sayi1; //Program bu noktada convert type 'int' to 'string' şeklinde hata verecektir.
```

Örnek:

```
int sayi1=100;
 string deger;
 deger=sayi1.ToString(); //Program artık hata vermeyecektir çünkü veri türleri uyumludur.
```

Uygulama-11

Adım 1: Form üzerine Görsel 1.28'de görüldüğü gibi bir CheckBox nesnesi ekleyiniz.

Adım 2: İki adet Label nesnesi ekleyiniz.

Adım 3: CheckBox nesnesi işaretli ise label2 nesnesine "True" değerini, CheckBox nesnesi işaretli değil ise label2 nesnesine "False" değerini gösteren programı yazınız.

Görsel 1.28: Lamba kontrol

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
    {
       bool secim;
       secim = checkBox1.Checked;  //Checked özelliği True veya False değerleri alır.
       label2.Text = secim.ToString();
    }
```

Sıra Sizde

Bir değişken tanımlayıp içine 500 sayı değerini atayınız. Butona tıklandığında bu değişkendeki değeri ekrana mesaj olarak verdiriniz.

1.7. Aritmetiksel Operatörler

Aritmetiksel operatörler, matematiksel işlemlerde kullanılan özel karakterlerdir (Tablo 1.5).

Tablo 1.5: Aritmetiksel Operatörler

Operatör Adı	Sembolü	Örnek
Toplama	+	2+8
Çıkarma	-	8-2
Çarpma	*	8*2
Bölme	/	8/2
Mod alma	%	8%2

Not: Bir sayı başka bir sayıya bölündüğünde kalan sayıya **mod** denir.
Örnek: 8%2=0 iken 8%5=3 sonucunu verir.

Uygulama-12

Girilen iki sayıyı toplayan programı Görsel 1.29'daki gibi tasarlayıp yazınız.

Görsel 1.29: Toplama uygulaması

```
private void button1_Click(object sender, EventArgs e)        {
        int sayi1, sayi2, toplam;
        sayi1=Convert.ToInt16(textBox1.Text);
        sayi2 = Convert.ToInt16(textBox2.Text);
        toplam = sayi1 + sayi2;
        textBox3.Text = toplam.ToString();
```

Not: TextBox nesnesinin içine sayısal bir değer girilmiş olsa bile matematiksel işlemler yapılamaz çünkü TextBox nesnesi metinsel veri türüne sahip değerler alabilen bir nesnedir. Bu tip durumlarda Convert kodu ile veri türü dönüşümü yapılmalıdır. Ayrıca TextBox nesnesine sayısal veri türün de bir değer yazmak için ToString() metodu ile veri türü dönüşümü yapılmalıdır.

Uygulama-13

Aşağıdaki örnek kodları çalıştırınız.

1.Örnek Kodlar	2.Örnek Kodlar
string ad, soyad,topla; ad = "Zeynep"; soyad = "Sare"; topla = ad +" "+ soyad; MessageBox.Show(topla);	textBox1.Text="25"; textBox2.Text="2" textBox3.Text= textBox1.Text + textBox2.Text

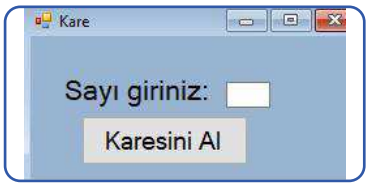
Not: Programlama dillerinde toplama operatörü sadece sayıları toplamak için kullanılmaz. Metinsel veri türüne sahip değişkenlerin arasına **artı (+)** operatörü konularak string değerlerini birleştirmek için de toplama operatörü kullanılır. Yukarıdaki 1. örnek kodlar çalıştığında ekrana **"Zeynep Sare"** mesajı gelecektir. 2. örnek kodlar çalıştığında ise textBox3'ün değeri **"252"** olur.yazmak için ToString() metodu ile veri türü dönüşümü yapılmalıdır.

Sıra Sizde

Girilen iki sayının toplama, çıkarma, çarpma ve bölme işlemlerini yapan programı tasarlayıp yazınız.

Sıra Sizde

Button nesnesine çift tıklandığında TextBox nesnesine girilen sayının karesini alıp mesaj verdiren programı Görsel 1.30'da görüldüğü gibi tasarlayıp yazınız.



Görsel 1.30: Kare alma uygulaması

1.7.1. İşlem Önceliği

Matematiksel bir ifade yazılan kod satırında birden fazla aritmetiksel operatör kullanılabilir. Böyle durumlarda sonuçların doğru çıkması için operatörler arasındaki işlem önceliklerine dikkat edilerek kod yazılmalıdır.

Kodlar çalıştırılırken öncelikle varsa parantez içindeki işlemler, daha sonra parantez dışındaki işlemler yapılır. Çarpma, bölme ve mod alma işlemleri kendi aralarında, toplama ve çıkarma işlemleri de kendi aralarında aynı önceliğe sahiptir. Kendi aralarında aynı önceliğe sahip operatörlerin olduğu ifadelerde ise sırasıyla soldan sağa doğru işlem yapılır.

Örnek: 4*3+15/3-1 işlemini yapınız.

Çözüm:
=12+5-1 -->Önce çarpma sonra bölme işlemi yapılır.
=17-1 -->Toplama işlemi ve sonrasında çıkarma işlemi yapılır.
=16

Örnek: (3+4-2)*(10/5*2)-4 işlemini yapınız.

Çözüm:
=5*4-4 -->Önce parantez içindeki işlemler yapılır.
=20-4 -->Çarpma işlemi toplama işleminden öncelikli olarak yapılır.
=16

Not: Kod yazımı sırasında karmaşayı ortadan kaldırmak ve en doğru sonuca ulaşmak için öncelikli olan ifadeleri parantez içine alabilirsiniz.

Sıra Sizde

Siz de ikili gruplar hâlinde içinde aritmetiksel operatörler geçen 10 adet ifade yazınız ve kendi sonuçlarınızla grup arkadaşınızın sonuçlarını karşılaştırınız.

Sıra Sizde

Aşağıdaki işlemlerin sonucunu yazınız.

7*5-3		(5*4)-(3+9)	
20+30-4*10		(5*8/4)+3-5*2	
9/3+2*2-5+1			

Uygulama-14

İki sayıyı toplayıp mesaj verdiren programı yazınız (Değişkenlere istenilen değerler atanabilir.).

```
private void button1_Click(object sender, EventArgs e)
{  int sayi1;
   int sayi2;
   int toplam;
   sayi1 = 5;
   sayi2 = 20;
   toplam = sayi1 + sayi2;
   MessageBox.Show(toplam.ToString());
}
```

Sıra Sizde

İki sayıyı toplayıp 2 ile çarparak çıkan sonuca 5 ekleyen ve oluşan değeri mesaj verdiren programı işlem önceliklerine dikkat ederek yazınız (Değişkenlere istenilen değerler atanabilir.).

```
private void button1_Click(object sender, EventArgs e)
{




}
```

Uygulama-15

http://kitap.eba.gov.tr/KodSor.php?KOD=21071

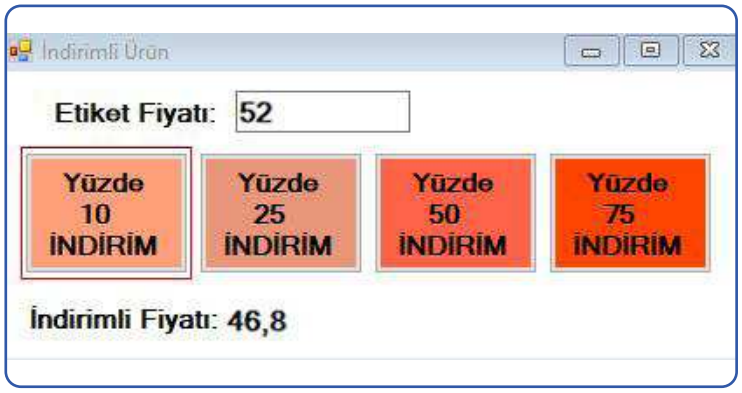
Adım 1: Form üzerine Görsel 1.31'de görüldüğü gibi etiket fiyatı girişi için TextBox nesnesi ekleyiniz.

Adım 2: Dört adet Button nesnesi ekleyiniz.

Adım 3: Button nesnelerinin arka plan renklerini değiştiriniz.

Adım 4: İndirim oranlarını Button nesnelerinin üzerine yazınız.

Adım 5: TextBox nesnesine etiket fiyatı girilen ürünün seçilen indirim oranına göre indirimli fiyatını bulup Label nesnesinde gösteren programı yazınız.



Görsel 1.31: İndirimli ürün

```
private void button1_Click(object sender, EventArgs e)
    {
        int etiketFiyati;
        double indirimliFiyat;
        etiketFiyati = Convert.ToInt32(textBox1.Text);
        indirimliFiyat = etiketFiyati-etiketFiyati * 0.10; //Yüzde 10 indirim
        label3.Text = indirimliFiyat.ToString();
    }
private void button2_Click(object sender, EventArgs e)
    {
        int etiketFiyati;
        double indirimliFiyat;
        etiketFiyati = Convert.ToInt32(textBox1.Text);
        indirimliFiyat = etiketFiyati-etiketFiyati * 0.25; //Yüzde 25 indirim
        label3.Text = indirimliFiyat.ToString();
    }
private void button3_Click(object sender, EventArgs e)
    {
        int etiketFiyati;
        double indirimliFiyat;
        etiketFiyati = Convert.ToInt32(textBox1.Text);
        indirimliFiyat = etiketFiyati-etiketFiyati * 0.50; //Yüzde 50 indirim
        label3.Text = indirimliFiyat.ToString();
    }
private void button4_Click(object sender, EventArgs e)
    {
        int etiketFiyati;
        double indirimliFiyat;
        etiketFiyati = Convert.ToInt32(textBox1.Text);
        indirimliFiyat = etiketFiyati-etiketFiyati * 0.75; //Yüzde 75 indirim
        label3.Text = indirimliFiyat.ToString();
    }
```

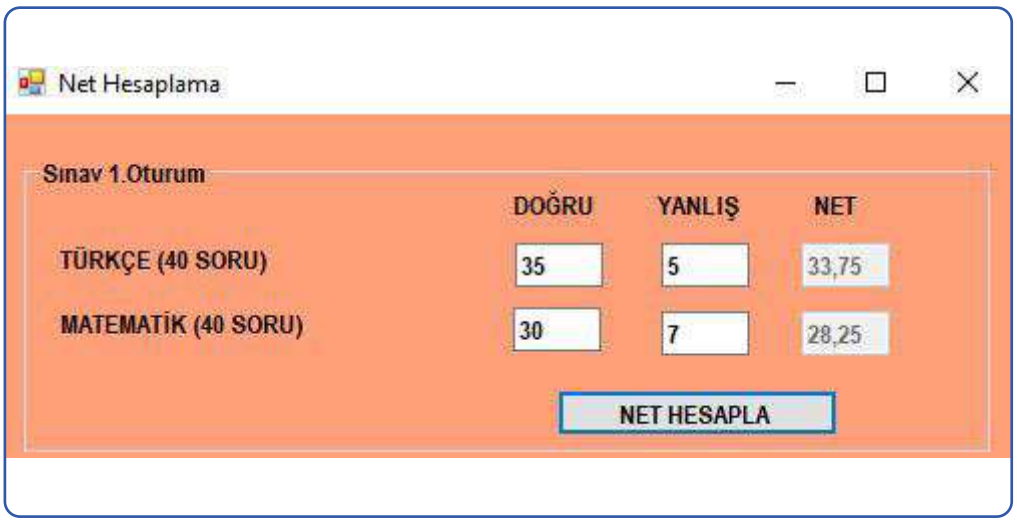
Not: "etiketFiyati - etiketFiyati * 0.10" ifadesinde işlem önceliği çarpmada olduğu için önce çarpma işlemi sonra çıkarma işlemi yapılacaktır. Aynı ifade "etiketFiyati - (etiketFiyati * 0.10)" şeklinde yazılarak da işlem yapılabilir.

Uygulama-16

Adım 1: Form üzerine Görsel 1.32'de görüldüğü gibi 1 adet GroupBox, 1 adet Button, 5 adet Label, 6 adet TextBox nesnesi ekleyiniz.

Adım 2: Türkçe ve Matematik netlerinin gösterileceği TextBox nesnelerinin Enabled özelliğini bilgi girişini engellemek için false yapınız.

Adım 3: Button nesnesi tıklandığında ilgili derslere ait net sayısını hesaplayan programı yazınız (Net sayısı, doğru sayısından yanlış sayısının dörtte biri çıkarılarak hesaplanır.).



Görsel 1.32: Net hesaplama

```
private void button1_Click(object sender, EventArgs e)
    {
        double turkceDogru, matDogru;
        double turkceYanlis, matYanlis;
        double turkceNet, matNet;
        turkceDogru = Convert.ToDouble(textBox1.Text);
        turkceYanlis = Convert.ToDouble(textBox2.Text);
        turkceNet = (turkceDogru - (turkceYanlis / 4));
        textBox3.Text = turkceNet.ToString();
        matDogru = Convert.ToDouble(textBox4.Text);
        matYanlis = Convert.ToDouble(textBox5.Text);
        matNet = (matDogru - (matYanlis / 4));
        textBox6.Text = matNet.ToString();
    }
```

ÖLÇME VE DEĞERLENDİRME - 1

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise (D), yanlış ise (Y) yazınız.

1. **( ) Kod editörü platformu kullanılarak sadece C# programlama dili kodlanabilir.**
2. **( ) string veri türüne sahip bir değişken int veri türüne dönüştürülebilir.**
3. **( ) int a=”52”; hatalı bir koddur.**
4. **( ) string a=”16”; hatalı bir koddur.**
5. **( ) Değişken isimleri sayı ile başlamaz.**
6. **( ) Bir nesne için aynı anda birden fazla olay metodu oluşturulabilir.**
7. **( ) Button nesnesi için sadece Click olay metodu oluşturulabilir.**

B) Aşağıda verilen cümlelerdeki boşlukları kutularda verilen ifadelerle tamamlayınız.

form	ToString()	%	ToolBox	using

8. **İsim uzayını projeye dâhil etmek için …… kodu kullanılır.**
9. **Form üzerine eklenen nesneler …… panelinden seçilir.**
10. **Mod alma işlemi için …… aritmetiksel operatörü kullanılır.**
11. **Sayısal veri türüne sahip bir değişken …… metodu ile string veri türüne dönüştürülür.**
12. **Aşağıdaki kod blokunda boş bırakılan yerlere doğru sözcükleri yazınız.**

```
using ……..;
…….. ResimEkle {
    public class Kaydet
   {
    //kodlarınız
   }
}
```

13. **Aşağıdaki kod satırlarından hangisi çalışmaz? Kodun çalışmama sebebini yazınız.**

```
int öğrenci no;
int 1.not;
string isim;
double yaricap;
bool cinsiyet;
int sayi1=13.4;
char karakter = “kmr”;
string tc = “00334412376”;
```

14. Aşağıdaki kodlar çalıştırıldığında ekranda görülecek sayı kaçtır?

```
double a, b, c, d, islem;
a = 1;
b = 6;
c = 5;
d = 2;
islem = a * b / c + d;
MessageBox.Show(islem.ToString());
```

15. TextBox nesnesine girilen sayının %18'ini bulup mesaj verdiren programı yazınız.

16. Yarıçapı girilen dairenin alanını ve çevresini hesaplayan programı yazınız.

2. ÖĞRENME BİRİMİ

http://kitap.eba.gov.tr/KodSor.php?KOD=11896

KARAR VE DÖNGÜ YAPILARI

NELER ÖĞRENECEKSİNİZ?

Bu öğrenme biriminde;

- Karşılaştırma operatörlerini sıralayacak,
- Karar ifadelerinin kullanım mantığını kavrayacak,
- if, if-else, else if ifadelerini kullanacak,
- İç içe karar ifadelerini kullanacak,
- Mantıksal operatörleri sıralayacak,
- Döngü çalışma mantığını kavrayacak,
- For, while, do-while döngülerinin kullanımını öğreneceksiniz.

ANAHTAR KELİMELER

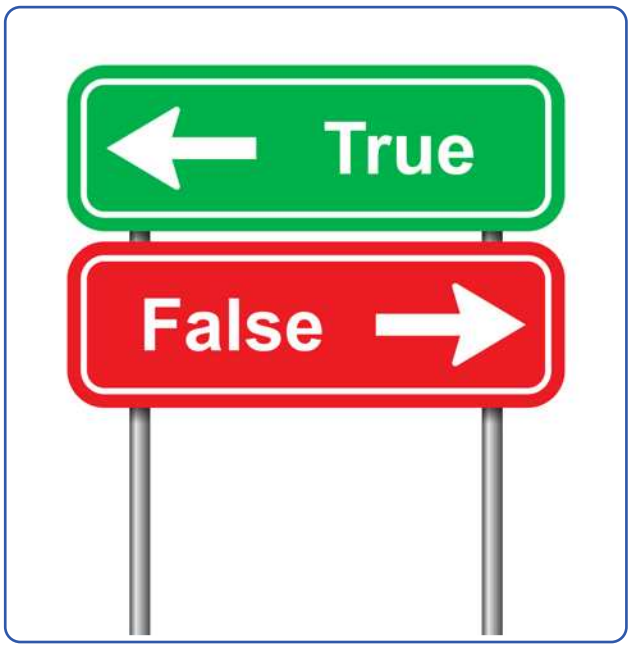
Karar ifadeleri, döngü, if, else, for, while, hata ayıklama

HAZIRLIK ÇALIŞMALARI

1. Karar ifadelerinin neler olabileceğini arkadaşlarınızla tartışınız ve günlük hayattaki kullanım alanlarına örnekler veriniz.
2. Program yazarken döngü yapılarının sağladığı kolaylıklar neler olabilir? Açıklayınız.

2.1. Karar İfadeleri

Karar ifadeleri; tüm programlama dillerinde bulunan, koşula veya koşullara bağlı bir şekilde programın nasıl ilerleyeceğini belirleyen yapılardır. Programın yol ayrımı noktalarını karar ifadeleri temsil eder. Bu yapı sayesinde programlar daha dinamik ve etkileşimlidir.



Görsel 2.1: Karar ifadeleri

Karar ifadeleri, programlamanın vazgeçilmez yapılarıdır. Örneğin; profil girişi için gerekli olan **kullanıcı adı ve şifre** bilgileri, karar ifadeleri kullanılarak kontrol edilir ve program girişe izin verir ya da izin vermez. Bu durum, programın ilerleyeceği yönü belirler (Görsel 2.1). TextBox nesnesine girilebilecek en büyük sayının 100 olması isteniyorsa karar ifadeleri ile TextBox nesnesinin değerinin kontrol edilmesi ve veri girişinin sağlanması da bu konuya bir başka örnektir.

2.1.1. Karşılaştırma Operatörleri

Karşılaştırma operatörleri, karar ifadelerinde iki değeri birbiriyle karşılaştırmak için kullanılan operatörlerdir (Tablo 2.1).

Tablo 2.1: Karşılaştırma Operatörleri

Operatör	Anlamı
<	Küçükse
>	Büyükse
==	Eşitse
<=	Küçük veya Eşitse
>=	Büyük veya Eşitse
!=	Eşit Değilse

2.1.2. if Yapısı

if, kelime olarak eğer anlamına gelmektedir. Sadece şart sağlandığında çalışması istenen kodlar için kullanılır. Şart ifadesi sağlandığında true, sağlanmadığında false değeri oluşur. if(true) olduğunda if yapısına bağlı kodlar çalışır, if(false) olduğunda kodlar çalışmaz.

```
if(şart ifadesi)
    {
     // Şart ifadesi sağlanıyorsa çalışacak kodlarınız
    }
```

Uygulama-1

Adım 1: Form üzerine Button nesnesi ekleyiniz.

Adım 2: Button nesnesinin Click olay metoduna aşağıdaki kodları yazınız.

```
byte skor1,skor2;
      skor1 = 4;
      skor2 = 1;
      if (skor1 > skor2)
      {
         MessageBox.Show("1. Takım Kazandı.");
      }
```

Not: Yukarıdaki uygulamada (**skor1 > skor2**) şart ifadesidir. Şart ifadesi sağlandığında çalışması istenilen kodlar küme parantezlerinin içine yazılır. Şart sağlandığı için parantez içindeki kod çalışacaktır. Şart sağlanmadığında parantez içindeki kod çalışmayacak, parantezden sonra hangi kod varsa onlar işletilecektir.

Uygulama-2

Adım 1: Form üzerine Button ve TextBox nesnesi ekleyiniz.

Adım 2: Button nesnesi tıklandığında TextBox nesnesine girilen sayı 17'den büyük ise "Ehliyet başvurusunda bulunabilirsiniz." şeklinde mesaj veren programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
    {
      byte yas;
      yas = Convert.ToByte(textBox1.Text);
      if (yas > 17)
      {
         MessageBox.Show("Ehliyet başvurusunda bulunabilirsiniz.");
      }
    }
```

Not: Yukarıdaki kod blokunda şart ifadesinde kullanılan "yas" değişkeni yerine "Convert.ToByte(textBox1.Text)" ifadesi de kullanılabilir.

Uygulama-3

Adım 1: Form üzerine 2 adet Label, 2 adet TextBox ve 1 adet Button nesnesi ekleyiniz.

Adım 2: Programı Görsel 2.2'de görüldüğü gibi tasarlayınız.

Adım 3: Button nesnesi tıklandığında textBox1'deki değer ile textBox2'deki değeri büyüklük, küçüklük ve eşitlik bakımından karşılaştıran ve sonucu ekrana mesaj olarak veren programı yazınız.

Görsel 2.2: Sayıları karşılaştırma uygulaması

```
private void button1_Click(object sender, EventArgs e)
{
    byte sayi1, sayi2;
    sayi1 = Convert.ToByte(textBox1.Text);
    sayi2 = Convert.ToByte(textBox2.Text);
    if (sayi1 > sayi2)
    {
    MessageBox.Show("1.sayı 2.sayıdan büyüktür.");
    }
    if (sayi1 == sayi2)
   {
   MessageBox.Show("Sayılar birbirine eşittir.");
   }
   if (sayi1 < sayi2)
   {
   MessageBox.Show("2.sayı 1.sayıdan büyüktür.");
   }
 }
```

Uygulama-4

Adım 1: Form üzerine Button ve TextBox nesnesi ekleyiniz.

Adım 2: Button nesnesi içine tıklandığında TextBox nesnesine girilen sayının tek mi, çift mi olduğunu bulan programı yazınız.

```
 int sayi;
sayi =Convert.ToInt32(textBox1.Text);
 if(sayi % 2 == 0)
{
MessageBox.Show("Bu bir çift sayıdır.");
 }
if(sayi % 2 ==1)
{
 MessageBox.Show("Bu bir tek sayıdır.");
}
```

Not: Bir sayının 2'ye bölümünden kalan sayı 0 ise bölünen sayı çifttir, kalan sayı 1 ise bölünen sayı tektir.

2.1.3. if-else Yapısı

else; kelime olarak **değilse, aksi durumda** anlamına gelmektedir. Şart ifadesi sağlandığında if kod bloku içindeki kodlar çalışır, şart ifadesi sağlanmadığında ise else kod bloku içindeki kodlar çalışır.

```
if(şart ifadesi)
  {
    // Şart ifadesi sağlanıyorsa çalışacak kodlarınız
  }
else
  {
    // Şart ifadesi sağlanmadığında çalışacak kodlarınız
  }
```

Uygulama-5

Adım 1: Form üzerine TextBox ve Button nesnesi ekleyiniz.

Adım 2: TextBox nesnesine girilen kullanıcı adı milliegitim@meb.k12.tr'ye eşit ise "Kullanıcı sisteme kayıtlıdır." mesajını, kullanıcı adı milliegitim@meb.k12.tr'ye eşit değil ise "Kullanıcı adınız yanlıştır." mesajını veren programı yazınız.

```
string kullanici_adi;
kullanici_adi=textBox1.Text;
if (kullanici_adi == "milliegitim@meb.k12.tr")
{
 MessageBox.Show("Kullanıcı sisteme kayıtlıdır.");
 }
 else
 {
 MessageBox.Show("Kullanıcı adınız yanlıştır.");
 }
```

Not: Yukarıdaki kod blokunda (**kullanici_adi == "milliegitim@meb.k12.tr"**) şart ifadesi sağlanmadığında **else** kod blokundaki kodlar çalışır.

Sıra Sizde

TextBox nesnesine girilen sayının tek mi, çift mi olduğunu bulan programı if-else yapısını kullanarak yazınız.

Uygulama-6

Adım 1: Form üzerine 2 adet GroupBox, 2 adet CheckBox, 1 adet Button ve 1 adet ListBox nesnesi ekleyiniz.

Adım 2: Form ve GroupBox nesnelerinin renklerini beyaz yapınız.

Adım 3: Button nesnesine tıklandığında "Lamba Aç/Kapa" işaretli ise "Lambalar Açık" mesajını, "Lamba Aç/Kapa" işaretli değil ise "Lambalar Kapalı" mesajını, "Kombi Aç/Kapa" işaretli ise "Kombi Açık" mesajını, "Kombi Aç/Kapa" işaretli değil ise "Kombi Kapalı" mesajını ListBox nesnesine ekleyen programı Görsel 2.3'teki gibi tasarlayıp yazınız.

```
if (checkBox1.Checked == true)
 {
  listBox1.Items.Add("Lambalar Açık");
 }
 else
 {
 listBox1.Items.Add("Lambalar Kapalı");
 }
 if (checkBox2.Checked == true)
 {
  listBox1.Items.Add("Kombi Açık");
 }
else
   {
   lis tBox1.Items.Add("Kombi Kapalı");
   }
```

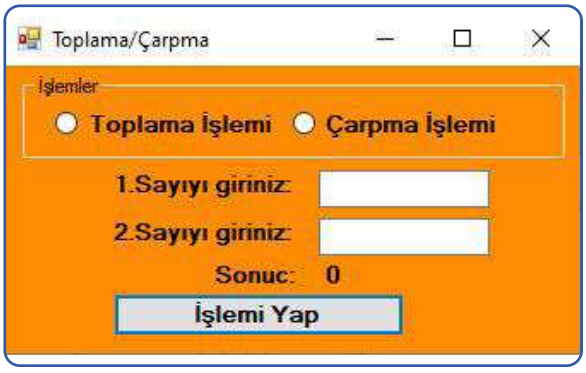
Görsel 2.3: Akıllı ev

Uygulama-7

Adım 1: Form üzerine 2 adet RadioButton, 2 adet TextBox, 4 adet Label, 1 adet Button ve 1 adet GroupBox nesnesi ekleyiniz.

Adım 2: Programı Görsel 2.4'te görüldüğü gibi tasarlayınız.

Adım 3: Button nesnesine tıklanıp RadioButton nesnelerinden "Toplama İşlemi" işaretlendiğinde sayıları toplayan, "Çarpma İşlemi" işaretlendiğinde sayıları çarpan ve sonucu ekrandaki Label nesnesinde gösteren programı if-else yapısını kullanarak yazınız.



Görsel 2.4: Toplama / Çarpma uygulaması

```
private void button1_Click(object sender, EventArgs e)
{
    int deger1 = Convert.ToInt32(textBox1.Text);
    int deger2 = Convert.ToInt32(textBox2.Text);
    int sonuc;
     if (radioButton1.Checked == true)
     {
      label4.Text = (deger1 + deger2).ToString();
       }
      else
      {
      label4.Text = (deger1 - deger2).ToString();
      }
}
```

2.1.4. else if Yapısı

Şartın sağlanmadığı durumlarda else if kullanılarak yeni bir şart ifadesi daha yazılabilir. else if ifadesinden sonra tekrar else if ifadesi veya else ifadesi kullanılabilir.

```
if(şart ifadesi)
{
 // if şart ifadesi sağlanıyorsa çalışacak kodlar
 }
else if(şart ifadesi)
{
// else if şart ifadesi sağlanıyorsa çalışacak kodlar
 }
```

Uygulama-8

http://kitap.eba.gov.tr/KodSor.php?KOD=21073

Adım 1: Form üzerine TextBox ve Button nesnesi ekleyiniz.

Adım 2: Button nesnesine tıklandığında TextBox nesnesinin içine yüzlük sistemde girilen bir notu beşlik sisteme çeviren programı yazınız.

```
int sayi;
 sayi = Convert.ToInt32(textBox1.Text);
 if (sayi < 0)
 { MessageBox.Show("0'dan büyük bir sayı giriniz!"); }
 else if (sayi < 25)
 {
 MessageBox.Show("Notunuz 0");
 }
 else if (sayi < 45)
 {
 MessageBox.Show("Notunuz 1");
 }
 else if (sayi < 55)
 {
 MessageBox.Show("Notunuz 2");
 }
 else if (sayi < 70)
 {
 MessageBox.Show("Notunuz 3");
 }
 else if (sayi < 85)
 {
 MessageBox.Show("Notunuz 4");
 }
else if (sayi <= 100)
{
 MessageBox.Show("Notunuz 5");
}
 else
{
 MessageBox.Show("Hatalı giriş yaptınız!");
 }
```

Sıra Sizde

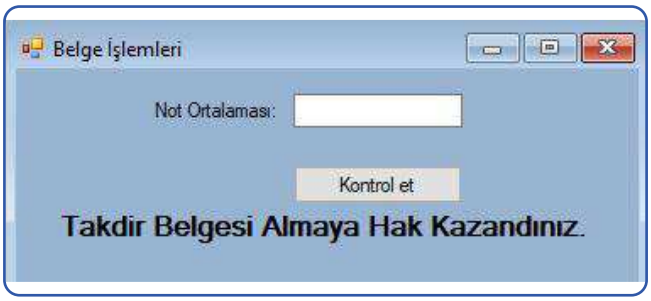
Hava sıcaklığı 10 derecenin altında ise "Hava soğuk" mesajını, 10-25 derece arasında ise "Hava hafif sıcak" mesajını, 25 derecenin üstünde ise "Hava sıcak" mesajını veren programı yazınız.

2.1.5. İç İçe Şart İfadeleri

İç içe şart ifadeleri, birbirini izleyen birden çok şart ifadesinin kontrolünü gerçekleştirmek için kullanılır.

```
if(şart ifadesi)
{
    if(şart ifadesi)
    {
      if(şart ifadesi)
        {

        }
        else if(şart ifadesi)
        {

        }
    }
}
```

Not: Programın mantığına göre şart ifadeleri istenildiği kadar birbiri içinde yazılabilir. İç içe şart ifadeleri kullanılırken karmaşık bir kod yazımı olmaması için küme parantezlerini kendi aralarında hizalamaya dikkat edilmelidir.



Görsel 2.5: Belge işlemleri uygulaması

Uygulama-9

Adım 1: Form nesnesi üzerine 2 adet Label, 1 adet TextBox ve 1 adet Button nesnesi ekleyiniz.

Adım 2: Programı Görsel 2.5'teki gibi tasarlayınız.

Adım 3: Kontrol et butonuna tıklandığında not ortalaması girilen kişinin notu 85 veya üzeri ise "Takdir Belgesi Almaya Hak Kazandınız." mesajını, not ortalaması girilen kişinin notu 70 veya üzeri ise "Teşekkür Belgesi Almaya Hak Kazandınız." mesajını, not ortalaması girilen kişinin notu 50'nin altında ise "Sınıf Geçmek İçin Yeterli Not Alamadınız." mesajını veren programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
{
    byte ortalama;
    ortalama = Convert.ToByte(textBox1.Text);
    if (ortalama >= 50)
    {
    if (ortalama >= 85)
    {
     label1.Text = "Takdir Belgesi Almaya Hak Kazandınız.";
     }
     else if (ortalama >= 70)
     {
     label1.Text = "Teşekkür Belgesi Almaya Hak Kazandınız.";
     }
     else
    { label1.Text = "Belge Almadan Sınıf Geçtiniz.";  }
    }
    else
   { label1.Text = "Sınıf Geçmek İçin Yeterli Not Alamadınız."; }
 }
```

2.1.6. Switch-Case

Switch-case, bir ifadenin aldığı değere bağlı olarak program için birçok farklı çalışma yolu belirleyen bir komuttur. Switch ifadesindeki değer hangi durumun değeri ile eşleşiyorsa o duruma ait kodlar çalışır. Hiçbir durum ile eşleşme olmaz ise default ifadesinde belirtilen kodlar çalışır. "break;" komutu, kodlar çalıştıktan sonra switch-case ifadesinden çıkmayı sağlar. Switch ifadesi içine yazılan değerin veri türü ile case ifadelerindeki değerin veri türleri aynı olmalıdır. Case ifadelerindeki değerler için değişken kullanılmaz.

```
switch (sayısal değer){
   case 1:
      //kodlar
      break;
   case 2:
      //kodlar
      break;
   case 3:
      //kodlar
      break;
   default:
      //kodlar
      break;
}
```

```
switch (karakter değer){
   case 'X':
      //kodlar
      break;
   case 'Y':
      //kodlar
      break;
   case 'Z':
      //kodlar
      break;
   default:
      //kodlar
      break;
}
```

```
switch (metinsel değer){
   case "kırmızı":
      //kodlar
      break;
   case "yeşil":
      //kodlar
      break;
   case "mavi":
      //kodlar
      break;
   default:
      //kodlar
      break;
}
```

Not: Switch-case ile yapılabilecek tüm işlemler if, if-else, else if yapıları kullanılarak da yapılabilir. Kod karmaşıklığını ortadan kaldırmak için uygun durumlarda switch-case kullanılabilir.

Uygulama-10

Adım 1: Form üzerine Button nesnesi ekleyiniz.

Adım 2: Button nesnesine tıklandığında bilgisayarın tarih bilgisine göre haftanın hangi gününde olduğunuzu bulan programı yazınız.

Not: DateTime.Now.DayOfWeek kodu ile haftanın kaçıncı gününde olunduğu bilgisi elde edilir.

```
int gun = Convert.ToInt32(DateTime.Now.DayOfWeek);
switch (gun)
{
case 1:
MessageBox.Show ("Pazartesi");
break;
case 2:
MessageBox.Show ("Salı");
break;
case 3:
MessageBox.Show ("Çarşamba");
break;
case 4:
MessageBox.Show ("Perşembe");
break;
case 5:
MessageBox.Show ("Cuma");
break;
case 6:
MessageBox.Show ("Cumartesi");
break;
case 0:
MessageBox. Show ("Pazar");
break;
default:
MessageBox.Show("Hata oluştu.");
break;
}
```

Uygulama-11

Adım 1: Form üzerine Button nesnesi ekleyiniz.

Adım 2: Button nesnesine tıklandığında bilgisayarın tarih bilgisine göre o günün hafta içi mi, hafta sonu mu olduğunu bulan programı yazınız.

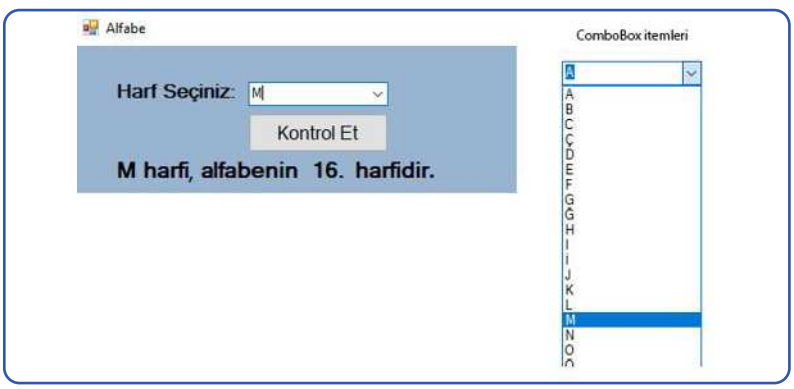
```
int gun = Convert.ToInt32(DateTime.Now.DayOfWeek);
switch (gun)
{
case 1:
case 2:
case 3:
case 4:
case 5:
MessageBox.Show("Hafta içi");
break;
case 6:
case 0:
MessageBox.Show("Hafta sonu");
break;
default:
MessageBox.Show("Hata oluştu.");
break;
}
```

Not: Yukarıdaki uygulamada görüldüğü gibi birden fazla durum için aynı kodlar çalıştırılabilir.

Sıra Sizde

ComboBox nesnesi içinden seçilen bir harfin alfabenin kaçıncı harfi olduğunu bulan programı

switch-case kullanarak Görsel 2.6'da görüldüğü gibi tasarlayıp yazınız.



Görsel 2.6: Alfabe uygulaması

Sıra Sizde

ComboBox nesnesi içinden seçilen bir harfin sesli harf mi, sessiz harf mi olduğunu bulan programı yazınız.

2.2. Mantıksal Operatörler

Kod blokları, if yapılarında mantıksal operatörler kullanılmadığı zaman tek bir şart ifadesinin sonucuna bağlı olarak çalışmaktadır. Bazen birden fazla şart ifadesinin kullanılması gerekebilir. Örneğin, kullanıcı doğrulama işlemi için kullanıcı adının ve şifre bilgilerinin aynı şart ifadesinde kontrol edilmesi gerekmektedir (Görsel 2.7). Bu durumda Tablo 2.2'deki mantıksal operatörlere ihtiyaç vardır.

Görsel 2.7: Mantıksal operatörler

Tablo 2.2: Mantıksal Operatörler

Operatör Adı	Sembolü	Örnek
AND (ve)	&&	(A<B) && (A<C)
OR (veya)	||	(A<B) || (A<C)
NOT (değil)	!	!(A<B)

Not: İç içe if yapıları kullanılarak birden fazla şart ifadesi kontrol edilebilir fakat bu durum kod yapısını karmaşık hâle getirebilir. Örneğin, 4 tane şart ifadesi için iç içe if yapısı kullanılırsa kodu okumak ve yazmak zorlaşacaktır.

2.2.1. AND (&&) Operatörü

Şart ifadelerinin hepsinin sağlanması gerektiğinde "and" operatörü kullanılır. Şart ifadelerinden herhangi biri sağlanmadığında if yapısına bağlı kod bloku çalışmaz.

```
if(user_name=="admin@meb.k12.tr" && password == "12345")
{
// Kodlarınız
}
```

```
if(user_name==”admin@meb.k12.tr” && password == “12345” && yas >= 15)
{
 // Kodlarınız
}
```

Not: Yeni bir && operatörü kullanılarak şart ifadelerinin sayısı artırılabilir.

Uygulama-12

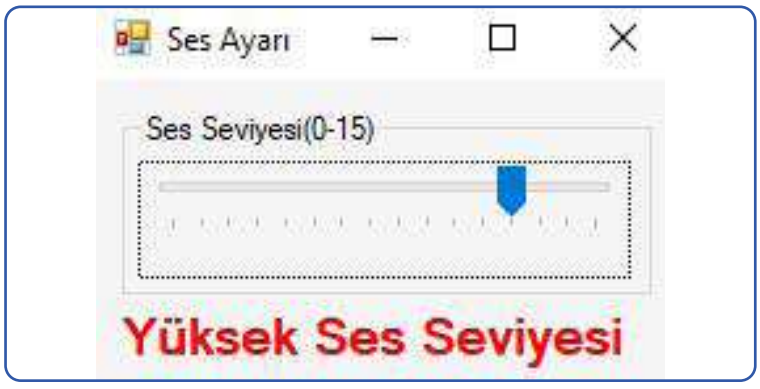
Adım 1: Form üzerine 1 adet GroupBox, 1 adet TrackBar, 1 adet Label nesnesi ekleyiniz.

Adım 2: TrackBar nesnesi için ”Scroll” olay metodu oluşturunuz (TrackBar nesnesi kaydırıldığında Scroll olayı tetiklenir.).

Adım 3: TrackBar nesnesinin Minimum özelliğini “0”, Maximum özelliğini “15” yapınız.

Adım 4: TrackBar nesnesinin Value özelliği 0 ise siyah renkli yazı ile Label nesnesine “Ses Yok”, TrackBar nesnesinin Value özelliği 1 ile 10 arasında ise yeşil renkli yazı ile Label nesnesine “Normal Ses Seviyesi”, TrackBar nesnesinin Value özelliği 11 ile 15 arasında ise kırmızı renkli yazı ile Label nesnesine “Yüksek Ses Seviyesi” mesajlarını veren programı Görsel 2.8’deki gibi tasarlayıp yazınız (Value özelliği, TrackBar nesnesinin hangi değere sahip olduğu bilgisini verir.).

```
private void trackBar1_Scroll(object sender, EventArgs e)
{  int ses = trackBar1.Value;
        if (ses == 0)
        {
        label1.Text = “Ses Yok”;      label1.ForeColor = Color.Black;
         }
        if (ses > 0 && ses <= 10)
        {
        label1.Text = “Normal Ses Seviyesi”;     label1.ForeColor = Color.Green;
         }
        if (ses >= 11)
        {
        label1.Text = “Yüksek Ses Seviyesi”;      label1.ForeColor = Color.Red;
        }
}
```



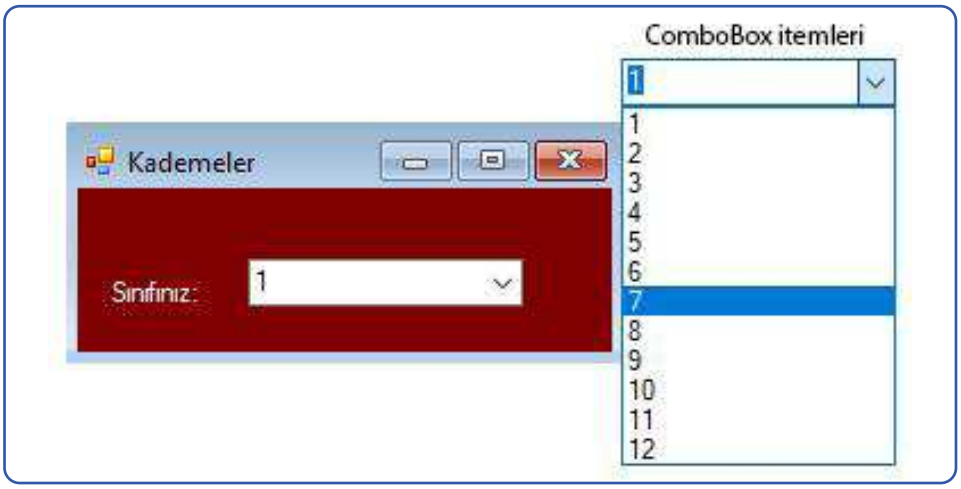
Görsel 2.8: Ses ayarı

Uygulama-13

Adım 1: Form üzerine Label ve ComboBox nesnesi ekleyiniz.

Adım 2: ComboBox nesnesinin içine 1 ile 12 arasındaki sayıları ekleyiniz.

Adım 3: ComboBox nesnesi içinden sınıf bilgisi seçildiğinde, seçilen sınıfın hangi eğitim-öğretim kademesine ait olduğunu mesaj veren programı Görsel 2.9'da görüldüğü gibi tasarlayıp yazınız.



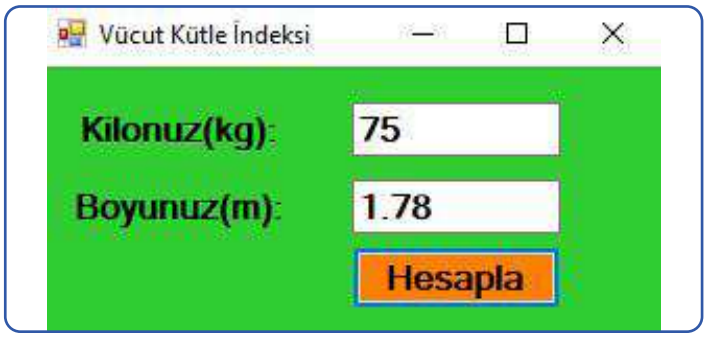
Görsel 2.9: Eğitim-öğretim kademeleri uygulaması

```
private void comboBox1_SelectedValueChanged(object sender,    EventArgs e)
{
   byte sinif;
   sinif = Convert.ToByte(comboBox1.Text);
   if (sinif > 0 && sinif < 5)
   {
   MessageBox.Show("İlkokul kademesi");
   }
   else if (sinif > 4 && sinif < 9)
   {
   MessageBox.Show("Ortaokul kademesi");
   }
   else if (sinif > 8 && sinif < 13)
   {
   MessageBox.Show("Lise kademesi");
    }
}
```

Not: ComboBox nesnesinin değeri değiştiğinde kodların çalışması için SelectedValueChanged olay metodu oluşturulur.

Sıra Sizde

TextBox nesnelerine kilo ve boy bilgileri girilen kişinin vücut kütle indeksini hesaplayıp çıkan sonuca göre vücut kütle indeksinin hangi kategoride olduğunu mesaj veren programı Görsel 2.10'da görüldüğü gibi tasarlayıp yazınız.



Görsel 2.10: Vücut kütle indeksi uygulaması

Tablo 2.3: Vücut Kütle İndeksi

Kategori	Vücut Kütle İndeksi (kg/m²)
İdeal Kilonun Altı	< 18,49
İdeal Kilo	18,5 - 24,99
İdeal Kilonun Üzeri	25 - 29,99
İdeal Kilonun Çok Üzeri	> 30

Not: Vücut kütle indeksi; bir kişinin kilogram cinsinden ağırlığının, metre cinsinden boy uzunluğunun karesine bölünmesiyle hesaplanır. Örneğin, ağırlığı 78 kg ve boyu 1.78 m olan biri için **vücut kütle indeksi=kg/m²** formülünden 23,4 kg/m² olarak hesaplanacaktır. Tablo 2.3'e göre 23,4 kg/m² indeksine sahip kişi ideal kilosundadır.

Sıra Sizde

TextBox nesnelerine girilen üç sayıdan hangisinin en büyük sayı, hangisinin en küçük sayı olduğunu bulan programı yazınız.

2.2.2. OR(||) Operatörü

Şart ifadelerinin en az bir tanesinin sağlanması gerektiğinde "or" operatörü kullanılır. Şart ifadelerinin hiçbiri sağlanmadığında if yapısına bağlı kod bloku çalışmaz.

```
if(yas > 65 || engel_durumu == true)
{
// kodlarınız
}
```

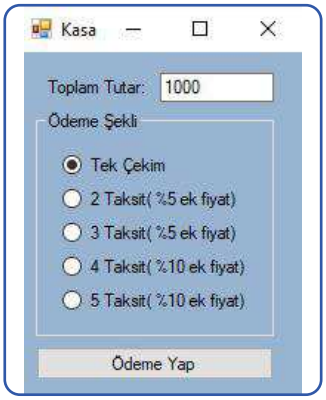
Not: Yeni bir | | operatörü kullanılarak şart ifadelerinin sayısı artırılabilir.

Uygulama-14

Adım 1: Form üzerine 1 adet TextBox, 1 adet Label, 1 adet Button, 1 adet GroupBox ve 5 adet RadioButton nesnesini ekleyiniz.

Adım 2: Form nesnesinin rengini "ActiveCaption" yapınız.

Adım 3: TextBox nesnesine toplam tutarı girilen bir alışverişin RadioButton nesneleri ile işaretlenen ödeme şekli 2 taksit veya 3 taksit ise ödenecek toplam tutara %5 ek fiyat, ödeme şekli 4 taksit veya 5 taksit ise ödenecek toplam tutara %10 ek fiyat ilave eden programı Görsel 2.11'de görüldüğü gibi tasarlayıp yazınız.



Görsel 2.11: Kasa uygulaması

```
private void button1_Click(object sender, EventArgs e)
   {
   double tutar;
   tutar = Convert.ToDouble(textBox1.Text);
   if(radioButton2.Checked==true || radioButton3.Checked == true)
   {
   tutar = tutar + (tutar * 0.05);
   }
   if (radioButton4.Checked == true || radioButton5.Checked == true)
  {
  tutar = tutar + (tutar * 0.10);
   }
  MessageBox.Show("Ödenecek Toplam Tutar:" + tutar.ToString() + "TL");
}
```

Uygulama-15

Adım 1: Form üzerine 1 adet ListBox ve 1 adet Button nesnesini ekleyiniz.

Adım 2: ListBox nesnesinin Items özelliğine "Mouse, Yazıcı, Klavye, Hoparlör, Kamera, Tarayıcı, Projeksiyon" değerlerini ekleyiniz.

Adım 3: ListBox nesnesi içinden seçilen bir bilgisayar parçasının giriş donanım birimi mi, çıkış donanım birimi mi olduğunu mesaj veren programı Görsel 2.12'de görüldüğü gibi tasarlayıp yazınız.

Görsel 2.12: Bilgisayar donanım birimleri uygulaması

```
private void button1_Click(object sender, EventArgs e)
{
    string secim;
    secim = listBox1.SelectedItem.ToString();
    if(secim=="Mouse" || secim=="Klavye" || secim == "Kamera" || secim=="Tarayıcı")
    {
     MessageBox.Show("Bu parça, giriş birimidir.");
     }
    if (secim == "Yazıcı" || secim == "Hoparlör" || secim == "Projeksiyon")
    {
    MessageBox.Show("Bu parça, çıkış birimidir.");
     }
}
```

2.2.3. Mantıksal Operatör Önceliği

Karar ifadelerinde ve(&&) operatörü veya(||) operatörüne göre daha öncelikli işleme alınır.

Uygulama-16

Adım 1: Form üzerine 2 adet GroupBox, 2 adet RadioButton, 1 adet ListBox ve 1 adet Button nesnesi ekleyiniz.

Adım 2: ListBox nesnesinin Items özelliğine "MP4, JPG, MOV, PNG" değerlerini ekleyiniz.

Adım 3: ListBox nesnesi içinden seçilen dosya uzantısı ile RadioButton nesnelerinden seçilen dosya türü bilgisinin eşleşmesini kontrol eden programı Görsel 2.13'te görüldüğü gibi tasarlayıp yazınız.

Görsel 2.13: Dosya uzantısı ve türünü eşleştirme uygulaması

```
private void button1_Click(object sender, EventArgs e)
{
    string secim;
    bool cevap1, cevap2;
    secim = listBox1.SelectedItem.ToString();
    cevap1 = radioButton1.Checked;
    cevap2 = radioButton2.Checked;
     if(cevap1==true && (secim=="MP4" || secim=="MOV" ))
     {
     MessageBox.Show("Cevabınız Doğru");
      }
     if (cevap1 == false && (secim == "MP4" || secim == "MOV"))
     {

     MessageBox.Show("Cevabınız Yanlış");
     }
     if (cevap2 == true && (secim == "JPG" || secim == "PNG"))
     {
     MessageBox.Show("Cevabınız Doğru");
     }
    if (cevap2 == false && (secim == "JPG" || secim == "PNG"))
    {
     MessageBox.Show("Cevabınız Yanlış");
      }
}
```

Not: Yukarıdaki uygulamada ve(&&) operatörü öncelikli olarak işleme alınacağı için veya(| |) operatörleri ile oluşturulan şart ifadeleri ayrı bir parantez içine alınmıştır. Aksi durumda program hatalı çalışacaktır.

Sıra Sizde

Yukarıdaki örnek uygulamayı veya(| |) operatörlerini ayrı bir parantez içine almadan çalıştırınız, oluşacak hatalı mesajları arkadaşlarınızla paylaşınız.

2.2.4. NOT(!) Operatörü

Bu operatör; şart ifadesinin alacağı sonuç true ise false, şart ifadesinin alacağı sonuç false ise true şeklinde değiştirir. Şart ifadesinin önüne değil(!) operatörü konularak kullanılır.

```
if (!radioButton1.Checked == true)
 {
 // kodlarınız
 }
```

Not: Yukarıdaki örnek kodda değil(!) operatörü kullanıldığı için radioButton1 nesnesi işaretli değilken kod çalışacaktır.

```
if (!(yas>=15 && yas<=65))
 {
 // kodlarınız
 }
```

Not: Yukarıdaki örnek kodda **yas** değişkeni şart ifadesinde belirtilen aralığın dışında ise kodlar çalışacaktır (Örneğin; yas=32 olduğunda kodlarınız çalışmayacak, yas=70 olduğunda kodlarınız çalışacaktır.).

Uygulama-17

Adım 1: Form üzerine 2 adet RadioButton nesnesi ve 1 adet Button nesnesi ekleyiniz.

Adım 2: Form nesnesinin arka plan rengini gri yapınız.

Adım 3: Button nesnesine tıklandığında üstteki RadioButton işaretliyse Form nesnesinin arka plan rengini siyah, alttaki RadioButton işaretliyse arka plan rengini beyaz yapan programı Görsel 2.14'te görüldüğü gibi tasarlayıp yazınız.

Görsel 2.14: Arka plan ayarları

```
private void button1_Click(object sender, EventArgs e)
{
    if (radioButton1.Checked == true)
    {
    this.BackColor = Color.Black;
    radioButton1.ForeColor = Color.White;
    radioButton2.ForeColor = Color.White;
    }
    if (!radioButton1.Checked == true)
    {
    this.BackColor = Color.White;
    radioButton1.ForeColor = Color.Black;
    radioButton2.ForeColor = Color.Black;
    }
}
```

Not: Yukarıdaki uygulamada if (!radioButton1.Checked == true) kodu yerine if (radioButton1.Checked == false) kodu veya else yapısı da kullanılabilir.

Sıra Sizde

Yanınızdaki arkadaşınızla birlikte aşağıdaki şart ifadelerini yeni bir proje dosyası açarak deneyiniz ve çıkan sonuçlara göre kodların hangi durumlarda çalışacağını açıklama kısmına yazınız.

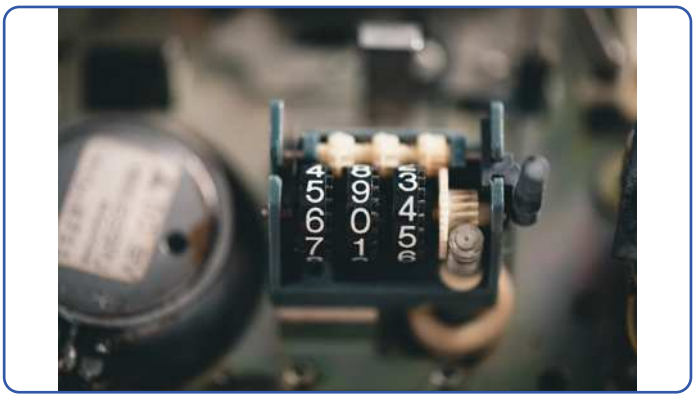
Şart İfadesi	Açıklama
if (!(Stok>=10))	Stok değişkeninin değeri 10'dan küçük olduğunda çalışır.
if (!(sayi > 0 && sayi <10))	
if (!(cinsiyet=="erkek"))	
if (!(numara%2==1))	
if (!(renk=="sarı") && numara%10==2)	
if (textBox1.Text != "Murat")	

2.3. Döngüler

Programda belirli kodların tekrar tekrar çalıştırılmasını sağlayan yapılara **döngü** denir. Duruma göre aynı kod 2 kez çalıştırılabileceği gibi 200 kez hatta 2000 kez de çalıştırılabilir. Böyle durumlarda program yazmak zorlaşacak, zaman alacak ve programın kod yapısı karmaşıklaşacaktır. Örneğin 1000 adet ürün arasından girilen barkod numarasına ait ürünün bilgilerini getirmek için 1000 adet if komutu kullanmak yerine döngü ifadesi içinde sadece bir tane if komutu kullanılabilir.

2.3.1. Sayaçlar

Bir değişkene bağlı değeri farklı aralıklarla artırmak, azaltmak, katlamak veya bölmek gerekebilir. Böyle durumlarda sayaç adı verilen değişkene değer atama yöntemleri kullanılır (Görsel 2.15). Tablo 2.4'te sayaçların kullanımına örnekler verilmiştir.



Görsel 2.15: Döngüler

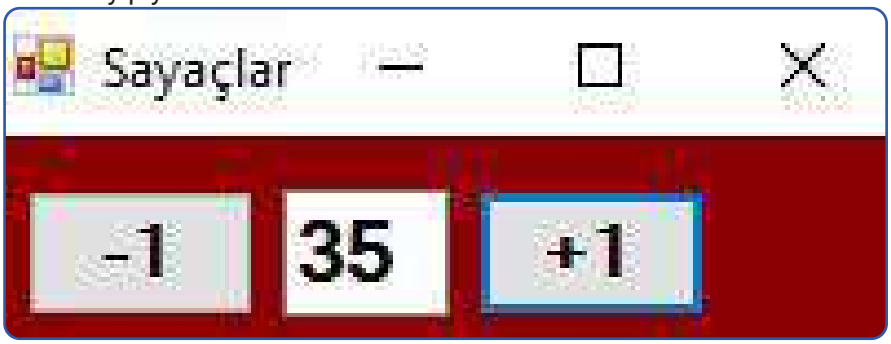
Tablo 2.4: Sayaçlar

Basit Atama ile Sayaç İfadesi	Bileşik Atama ile Açıklama
x=x+1;	x+=1;
x=x-2;	x-=2;
x=x*3;	x*=3
x=x/5;	x/=5;

Uygulama-18

Adım 1: Form üzerine 2 adet Button ve 1 adet TextBox nesnesi ekleyiniz.

Adım 2: Eksi bir yazılı Button nesnesine tıklandığında TextBox nesnesindeki sayıyı 1 azaltan, artı bir yazılı Button nesnesine tıklandığında TextBox nesnesindeki sayıyı 1 artıran programı Görsel 2.16'daki gibi tasarlayıp yazınız.



Görsel 2.16: Sayaçlar

```
private void button1_Click(object sender, EventArgs e)
{
int sayi;
sayi = Convert.ToInt32(textBox1.Text);
sayi = sayi - 1;
textBox1.Text = sayi.ToString();
}
private void button2_Click(object sender, EventArgs e)
{
int sayi;
sayi = Convert.ToInt32(textBox1.Text);
sayi = sayi + 1;
textBox1.Text = sayi.ToString();
}
```

2.3.2. Artırma ve Azaltma Operatörleri

Artırma ve azaltma operatörleri döngü yapılarında çok sık kullanılmaktadır. Döngü değişkeninin değeri birer birer artıyor veya birer birer azalıyorsa sayaç yerine pratik bir kullanıma sahip olan **artırma ve azaltma** operatörleri tercih edilebilir (Tablo 2.5).

Tablo 2.5: Artırma ve Azaltma Operatörleri

Operatör	Açıklama
x++;	Değişkenin değerini 1 artırır.
x--;	Değişkenin değerini 1 azaltır.

Uygulama-19

Adım 1: Form üzerine 2 adet PictureBox ve 4 adet Button nesnesi ekleyiniz.

Adım 2: PictureBox nesnelerine at resmi yerleştiriniz.

Adım 3: Normal yazılı Button nesnelerine tıklandığında PictureBox nesnesinin Left özelliğini 10 artırarak, Hızlı yazılı Button nesnesine tıklandığında PictureBox nesnesinin Left özelliğini 25 artırarak PictureBox nesnelerini hareket ettiren programı Görsel 2.17'deki gibi tasarlayıp yazınız (Herhangi bir nesne, Left özelliğinin değeri değiştirilerek Form nesnesi üzerinde sola veya sağa kaydırılabilir.).

Görsel 2.17: Artırma operatörü

```
private void button1_Click(object sender, EventArgs e)
{
pictureBox1.Left = pictureBox1.Left + 10;
}
private void button2_Click(object sender, EventArgs e)
{
pictureBox1.Left = pictureBox1.Left + 25;
}
private void button3_Click(object sender, EventArgs e)
{
pictureBox2.Left = pictureBox2.Left + 10;
}
private void button4_Click(object sender, EventArgs e)
{
pictureBox2.Left = pictureBox2.Left + 25;
}
```

2.3.3. For Döngüsü

Genellikle kodların tekrar sayısı belli olduğunda for döngüsü kullanılır. Döngü için tanımlanan şart ifadesi her sağlandığında döngüdeki kodlar tekrar çalışır. For döngüsünün kaç kez çalışacağını belirlemek oldukça basittir.

```
for (Döngü değişkeni başlangıç değeri; Döngü şart ifadesi; Döngü değişkeni sayacı)
{
//kodlarınız
}
```

```
for (int i = 0; i < 10; i++)
{
//kodlarınız
}
```

Yandaki örnek kodda "i" değişkeninin alacağı her değer için aynı kodlar çalışır. "i" değişkenindeki değer artışı "i++" sayacı (artış operatörü) ile sağlanır. "i" değişkeni sırasıyla 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 değerlerini alır ve böylece döngü içindeki kodlar toplam 10 kez çalışır. "i" değişkeninin değeri 10 olduğunda "i<10" şart ifadesi sağlanamadığı için döngü durur (Döngü değişkeni ikişer ikişer artmış olsaydı döngü içindeki kodlar 5 kez çalışacaktı.).

Uygulama-20

Adım 1: Form üzerine 1 adet ListBox nesnesi ekleyiniz.

Adım 2: ListBox nesnesinin içine 7 tane "Bilişim Teknolojileri" metni ekleyen programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
{
    for (int i = 0; i < 7; i++)
    {
    listBox1.Items.Add("Bilişim Teknolojileri");
    }
}
```

Not: For döngüsü ile pratik bir şekilde ListBox nesnesine elemanlar eklendi.

```
private void button1_Click(object sender, EventArgs e)
{
listBox1.Items.Add("Bilişim Teknolojileri");
listBox1.Items.Add("Bilişim Teknolojileri");
listBox1.Items.Add("Bilişim Teknolojileri");
listBox1.Items.Add("Bilişim Teknolojileri");
listBox1.Items.Add("Bilişim Teknolojileri");
listBox1.Items.Add("Bilişim Teknolojileri");
listBox1.Items.Add("Bilişim Teknolojileri");
}
```

Not: Döngü kullanılmadan uygulama yapıldığında ListBox'a eklenmek istenen elemanların hepsini tek tek yazmak gereklidir.

Uygulama-21

Adım 1: Form üzerine 1 adet ListBox nesnesi ekleyiniz.

Adım 2: ListBox nesnesinin içine 1'den 10'a kadar olan sayıları ekleyen programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
{
   for (int i = 1; i <= 10; i++)
   {
   listBox1.Items.Add(i);
   }
}
```

Not: Döngü değişkeninin başlangıç değeri "1", bitiş değeri (şart ifadesindeki değeri) "10", artış miktarı da "1"dir. Böylece "i" değişkeni sırasıyla 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 değerlerini alır. Bunlar, ListBox nesnesine eklenecek değerlerdir.

Uygulama-22

Adım 1: Form üzerine 1 adet ComboBox nesnesi ekleyiniz.

Adım 2: ComboBox nesnesinin içine 6'dan 16'ya kadar olan sayıları ekleyen programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
{
    for (int i = 6; i <=16; i++)
    {
    comboBox1.Items.Add(i);
    }
}
```

Not: Döngü değişkeninin başlangıç değeri "6", bitiş değeri (şart ifadesindeki değeri) "16", artış miktarı da "1"dir. Böylece "i" değişkeni sırasıyla 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 değerlerini alır.

Uygulama-23

Adım 1: Form üzerine 1 adet ComboBox nesnesi ekleyiniz.

Adım 2: ComboBox nesnesinin içine 100 ile 150 arasında 5'in katı olan sayıları ekleyen programı yazınız.

```
private void button1_Click(object sender, EventArgs e)
{
    for (int i = 100; i <=150; i=i+5)
    {
    comboBox1.Items.Add(i);
    }
}
```

Not: Döngü değişkeninin başlangıç değeri "100", bitiş değeri (şart ifadesindeki değeri) "150", artış miktarı da "5"tir. Böylece "i" değişkeni sırasıyla 100, 105, 110, 115, 120, 125, 130, 135, 140, 145, 150 değerlerini alır.

Sıra Sizde

ListBox nesnesinin içine sırasıyla 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110 sayılarını ekleyen programı yazınız.

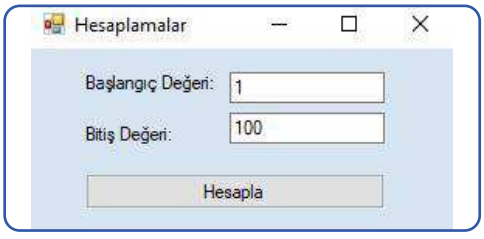
Sıra Sizde

ListBox nesnesinin içine sırasıyla 110, 100, 90, 80, 70, 60, 50, 40, 30, 20, 10 sayılarını ekleyen programı yazınız.

Uygulama-24

Adım 1: Form üzerine 2 adet Label, 2 adet TextBox ve 1 adet Button nesnesi ekleyiniz.

Adım 2: Button nesnesine tıklandığında TextBox nesneleri içine girilen başlangıç ve bitiş değerleri arasındaki sayıları toplayarak sonucu mesaj veren programı Görsel 2.18'de görüldüğü gibi tasarlayıp yazınız.



Görsel 2.18: Hesaplamalar

```
private void button1_Click(object sender, EventArgs e)
{
int sayi1, sayi2, toplam;
sayi1 = Convert.ToInt32(textBox1.Text);
sayi2 = Convert.ToInt32(textBox2.Text);
toplam = 0;
for (int i = sayi1; i <=sayi2 ; i++)
{
toplam = toplam + i;
}
MessageBox.Show("Sayıların Toplamı=" + toplam.ToString());
}
```

Not: Yukarıdaki uygulamada döngünün başlangıç ve bitiş değerleri için değişkenler kullanılmıştır.

Uygulama-26

Adım 1: Form üzerine 1 adet ListBox ve 1 adet Button nesnesi ekleyiniz.

Adım 2: 10 sayısının 0'dan 4'e kadar olan kuvvetlerini ListBox nesnesi içine ekleyen programı Görsel 2.19'da görüldüğü gibi tasarlayıp yazınız.

Görsel 2.19: Kuvvet alma

```
private void button1_Click(object sender, EventArgs e)
{
    double kuvvet;
    for (int i = 0; i < 5; i++)
   {
   kuvvet = Math.Pow(10, i);
   listBox1.Items.Add(kuvvet);
   }
}
```

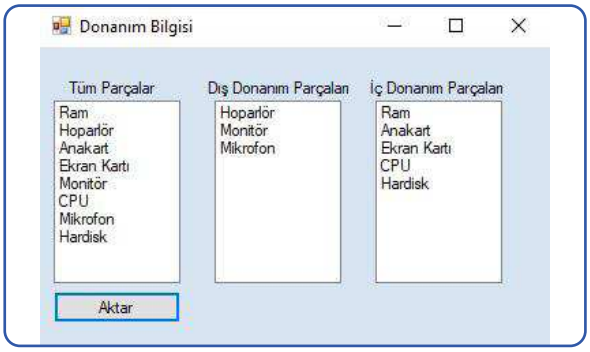
Not: Math.Pow fonksiyonu bir sayının istenilen kuvvetini almak için kullanılır. Bu fonksiyon, double veri türünde geriye değer döndürür.

Uygulama-26

Adım 1: Form üzerine 3 adet ListBox, 3 adet Label ve 1 adet Button nesnesi ekleyiniz.

Adım 2: İlk ListBox nesnesinin Items özelliğine "Ram, Hoparlör, Anakart, Ekran Kartı, Monitör, CPU, Mikrofon, Harddisk" değerlerini ekleyiniz.

Adım 3: Button nesnesine tıklandığında listBox1 nesnesi içinde karışık bir şekilde listelenmiş bilgisayar parçalarını dış donanım ve iç donanım birimleri olarak ayırıp listBox2 ve listBox3 nesnelerine aktaran programı Görsel 2.20'de görüldüğü gibi tasarlayıp yazınız.



Görsel 2.20: Donanım bilgisi

```
private void button1_Click(object sender, EventArgs e)
{
    for (int i = 0; i < listBox1.Items.Count; i++)
    {
        if(listBox1.Items[i].ToString() == "Hoparlör" ||
         listBox1.Items[i].ToString() == "Mikrofon" ||
         listBox1.Items[i].ToString() == "Monitör")
         {
         listBox2.Items.Add(listBox1.Items[i]);
         }
         else
         {
         listBox3.Items.Add(listBox1.Items[i]);
         }
     }
}
```

Not: listBox1.Items.Count kodu listBox nesnesinin içindeki item sayısını verir. Yukarıdaki uygulamada listBox1 nesnesinde 8 tane parça ismi ekli olduğu için listBox1.Items.Count kodu ile 8 sayısı elde edilir.

Sıra Sizde

Rastgele 20 adet sayıyı listBox1 nesnesi içine ekleyiniz. Butona tıkladığınızda listBox1 nesnesindeki tek sayıları listBox2 nesnesine, çift sayıları ise listBox3 nesnesine aktaran programı yazınız.

2.3.4. While Döngüsü

Döngü için tanımlanan şart ifadesi sağlanıyorsa döngü çalışmaya başlar. Şart ifadesi sağlandığı sürece while döngüsü çalışmaya devam eder.

```
while (Şart ifadesi);
{
//kodlar
}
```

Uygulama-27

Adım 1: Form üzerine ListBox nesnesi ekleyiniz.

Adım 2: ListBox nesnesi içine 1'den 10'a kadar olan sayıları ekleyen programı yazınız.

```
int say = 1;
while (say<=10)
{
listBox1.Items.Add(say);
say++;
}
```

Not: **While** ve **do-while** döngülerinde döngü değişkeni, for döngüsünden farklı olarak döngünün dışında tanımlanır ve değişkenin değeri döngünün içinde değiştirilir.

Sıra Sizde

ListBox nesnesi içine 20 tane farklı sayı ekleyiniz. Button nesnesine tıklandığında ListBox nesnesindeki ilk sayıdan başlayarak sayıları toplayan programı while döngüsü kullanarak yazınız.

Uygulama-28

0'dan başlayarak sırasıyla sayıları toplayan ve sayıların toplamı 1000'den fazla olduğunda döngünün kaç kez çalıştığını bulan programı yazınız.

```
int dongu_say = 1;
int toplam = 0;
while (toplam<=1000)
{
toplam = toplam + dongu_say;
dongu_say += 1;
}
MessageBox.Show("Döngü toplam " +
dongu_say.ToString() + " kez çalıştı");
```

2.3.5. Do-while Döngüsü

Döngü için tanımlanan şart ifadesi sağlanmasa da do-while döngüsü en az bir kez çalışır çünkü while döngüsünde şart ifadesi döngünün başlangıcındayken do-while döngüsünde şart ifadesi döngünün sonundadır. Do-while döngüsü de diğer döngüler gibi şart sağlandığı sürece çalışmaya devam eder.

```
do{//kodlar
}while (Şart ifadesi);
```

Uygulama-28

Adım 1: Form üzerine ListBox nesnesi ekleyiniz.

Adım 2: ListBox nesnesi içine 1'den 10'a kadar olan sayıları ekleyen programı yazınız.

```
int say = 1;
do
{
listBox1.Items.Add(say);
say++;
}while(say<=10);
```

Do-while döngüsü şart sağlanmasa bile içindeki kodları bir kez çalıştırır.

do-while	while

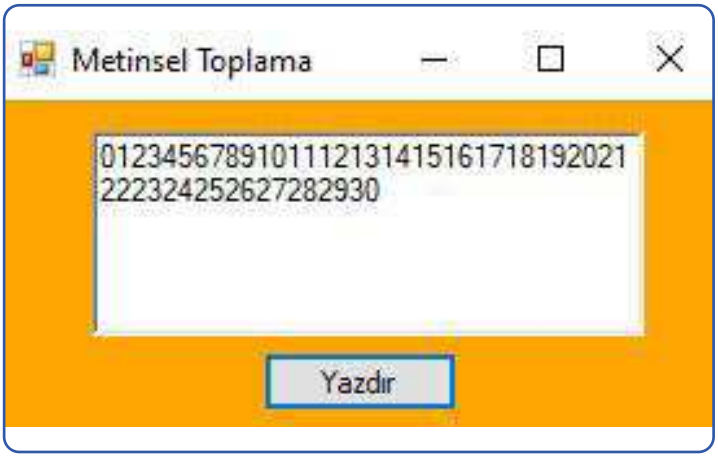
int say = 1
do
{
listBox1.Items.Add(say);
say++;
} while (say > 5);
	int say = 1;
	while (say > 5)
	{
	listBox1.Items.Add(say);
	say++;
	}

Not: While döngüsünde şart ifadesi sağlanmadığı için listBox1 nesnesine hiçbir sayı eklenmeyecektir. Do-while döngüsünde şart sağlanmasa bile listBox1 nesnesine bir tane eleman eklenecektir.

Uygulama-29

Adım 1: Form üzerine 1 adet RichTextBox, 1 adet Button nesnesi ekleyiniz.

Adım 2: Button nesnesine tıklandığında 0'dan 30'a kadar olan sayıları yan yana RichTextBox nesnesine aktaran programı do-while döngüsü kullanarak Görsel 2.21'deki gibi tasarlayıp yazınız.



Görsel 2.21: Metinsel toplama

```
private void button1_Click(object sender, EventArgs e)
{
  int say=0;
  do{

  richTextBox1.Text = richTextBox1.Text + say.ToString();
  say++;
  } while (say<=30);
}
```

2.3.6. Döngüyü Kesme (Durdurma)

Döngüyü kesme komutu "break;" döngü tamamlanmadan döngüden çıkmak için kullanılır. "break;" komutundan sonra döngüye ait hiçbir kod çalışmaz ve program, döngü ifadesinden sonraki kod satırı ile çalışmaya devam eder.

For Döngüsü

```
for (int a = 0; a < 10; a++)
{
   MessageBox.Show("Döngü Çalışıyor.");
   if (a == 5)
   {
   MessageBox.Show("Döngü Durdu.");
   break;
   }
}
```

While Döngüsü

```
int a = 0;
while (a<10)
 {
  MessageBox.Show("Döngü Çalışıyor.");
  if (a == 5)
    {
     MessageBox.Show("Döngü Durdu.");
     break;
    }
  a++;
  }
```

Do-While Döngüsü

```
int a = 0;
do
{
    MessageBox.Show("Döngü Çalışıyor.");
    if (a == 5)
    {
    MessageBox.Show("Döngü Durdu.");
     break;
     }
     a++;
}while (a<10);
```

Not: Döngü türleri için ayrı ayrı verilen örnek kodlara göre "a" döngü değişkeni en son 5 değerini alır, "break;" kesme komutu kullanıldığı için döngü tamamlanmadan durur ve döngü 10 kez çalışması gerekirken 6 kez çalışır.

Uygulama-31

Adım 1: Form nesnesi üzerine 2 adet Label, 1 adet ListBox, 1 adet TextBox ve 1 adet Button nesnesi ekleyiniz.

Adım 2: ListBox nesnesinin Items özelliğine 10 adetten fazla isim giriniz.

Adım 3: TextBox nesnesi içine girilen müşteri adını ListBox nesnesi içinde arayarak eşleşen ad varsa "Aradığınız müşteri bulundu!" şeklinde mesaj veren programı Görsel 2.22'de görüldüğü gibi tasarlayıp yazınız.

Görsel 2.22: Müşteri arama

```
string ad = textBox1.Text;

for (int i = 0; i < listBox1.Items.Count; i++)
{
if (listBox1.Items[i].ToString() == ad)
 {
MessageBox.Show(listBox1.Items[i].ToString()  + " adlı müşteri bulundu!");
break;
  }
}
```

Not: Uygulamada "....adlı müşteri bulundu!" mesajı verildikten sonra "break;" komutu çalıştığı için döngü kesintiye uğrar ve sonlanır. Döngü içinde "break;" komutu kullanılmasaydı müşterinin adı bulunduktan sonra döngüde yapılan tüm işlemler gereksiz olacaktı. Bu gereksiz işlemler, programın çalışma performansını etkilemektedir. 3000 tane müşteri adının kayıtlı olduğu bir listede aranan müşteri listenin 15. sırasında bulunmuş ise geri kalan 2985 tane kaydı kontrol etmek hem gereksizdir hem de programın performans kaybına sebep olur.

Sıra Sizde

Yukarıdaki uygulamada for döngüsü ile düzenlenmiş döngü kodlarını **while** ve **do-while** döngülerini kullanarak ayrı ayrı yazınız.

Sıra Sizde

1 ile 100 arasında 3'ün katı olan sayıları toplarken sayıların toplamı 200'ü geçince sadece bir kez "Limit aşıldı." mesajı veren programı yazınız.

```
int .....= 0;
for(int say = 0; say < 100; say = say +.....)
{
    topla = topla + say;
    if (.....)
    {
    MessageBox.....
     .....
     }
}
```

2.3.7. Döngüyü Devam Ettirme

Döngü içinde "continue;" komutu çalıştıktan sonra diğer kodlar döngünün o andaki adımı için çalışmaz ve döngü bir sonraki adıma geçer. Böylece döngü isteğe bağlı bir şekilde belli adımları atlayarak çalışır.

```
for (int a = 1; a < 15; a++)
{
  if (a==5 || a==10 )
  {
  continue;
  }
  //kodlar
}
```

```
int a = 1;
while (a<15)
{
   //kodlar
   if (a==5 || a==10)
   {
   continue;
   }
   a++;
}
```

```
int a = 1;
do{
//kodlar
   if (a==5 || a==10)
   {
   continue;
   }
   a++;
}while (a<15);
```

Yukarıdaki döngü ifadelerinde "a" değişkeninin değeri 5'e veya 10'a eşit olduğunda döngü bir sonraki değerini alır.

Sıra Sizde

Aşağıdaki kodlara göre listBox1 nesnesi içine hangi sayılar eklenir?

```
for (int i = 1; i <= 10; i++)
{
  if (i < 7)
  {
  continue;
  }
   listBox1.Items.Add(i);
}
```

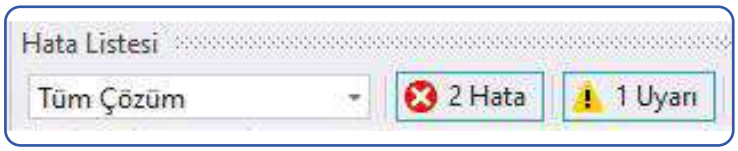
Cevap:

```
for (int i = 1; i <= 10; i=i+2)
{
 listBox1.Items.Add(i);
  if (i >= 7)
  {
  continue;
  }
}
```

Cevap:

2.4. Hata Ayıklama

Kod editörü derleyicisi, programın kodları yazım kurallarına uygun olmadığında Hata Listesi panelinde hatalı kodları göstermektedir (Görsel 2.23). Bazı hatalar program çalıştıktan sonra oluşmaktadır ve programı durdurmaktadır. Örneğin, programda TextBox nesnesine girilen sayı ile işlem yapılıyorsa fakat TextBox nesnesine sayı yerine harf girilmişse program hata verip duracaktır. Bu hata, kullanıcının yanlış veri girişinden kaynaklandığı için derleyici bu tip hataları yakalayıp Error List panelinde gösteremez. Bu tip hataların programı durdurmaması için önlem olarak **try-catch-finally** hata ayıklama blokları kullanılmalıdır.



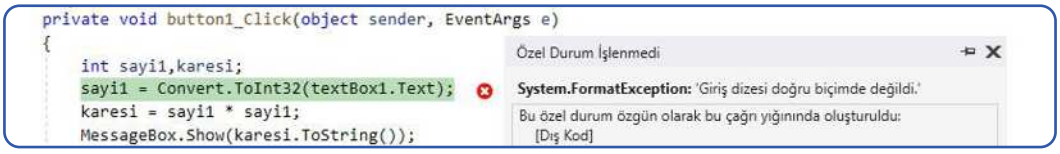
Görsel 2.23: Hata ayıklama

Yukarıdaki örnek kodlar çalıştırıldığında textBox1 nesnesine sayı yerine harf girilmişse program Görsel 2.24'teki gibi hata verecek ve durdurulacaktır. Bu tip hatalar özel durumlarda oluşan hatalardır ve program çalıştırılmadan önce derleyici tarafından tespit edilemez.

2.4.1. Try-Catch-Finally Bloku

Program çalıştırıldığında hata meydana gelme olasılığı olan kodlar **try** bloku içinde yazılır. Try bloku içine yazılan kodlarda hata meydana gelirse program **try** blokundan çıkarak **catch** bloku içindeki kodları çalıştırır. Böylece program hata vermez ve çalışmaya devam eder.

Try bloku içine yazılan kodlarda bir hata meydana gelmezse **catch** blokundaki kodlar çalışmaz fakat **try** blokunda hata meydana gelse de gelmese de **finally** blokundaki kodlar çalışır. **Finally** blokunu kullanmak zorunlu değildir, tercihe bağlıdır.



Görsel 2.24: Hatalı kod-1

Uygulama-32

Adım 1: Form üzerine ButtonTextBox nesnesi ekleyiniz.

Adım 2: Button nesnesinin Click olay metodunu oluşturunuz.

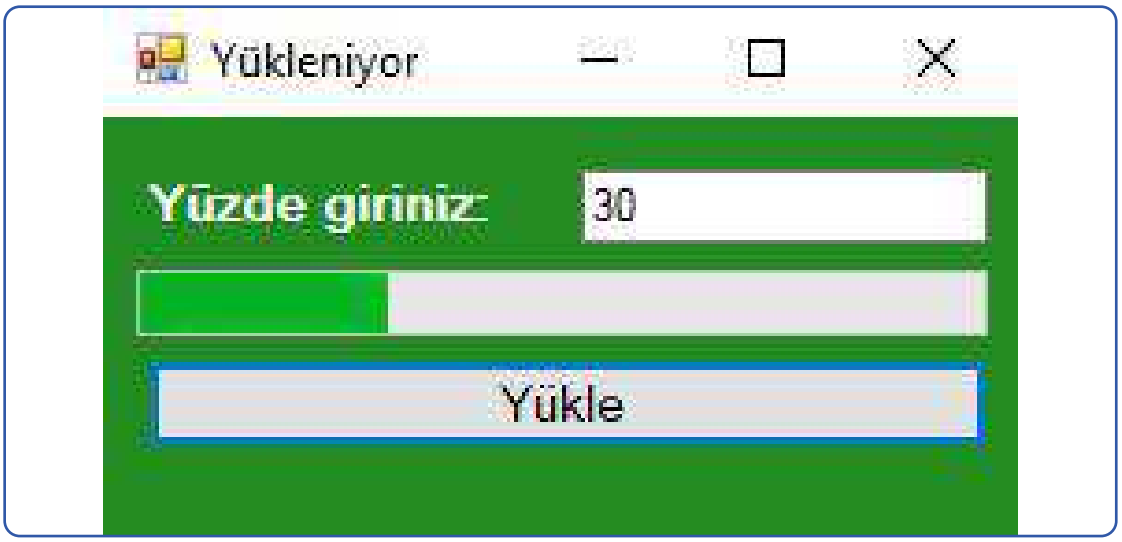
Adım 3: Aşağıdaki kodları Click olay metodunun içine yazınız.

```
int sayi1,sayinin_karesi;
try
{
sayi1 = Convert.ToInt16(textBox1.Text);
sayinin_karesi = sayi1 * sayi1;
MessageBox.Show(sayinin_karesi.ToString());
}
catch
{
MessageBox.Show("Hatalı giriş yaptınız!");
}
finally
{
// Hata olsa da olmasa da çalışacak kodlar
}
```

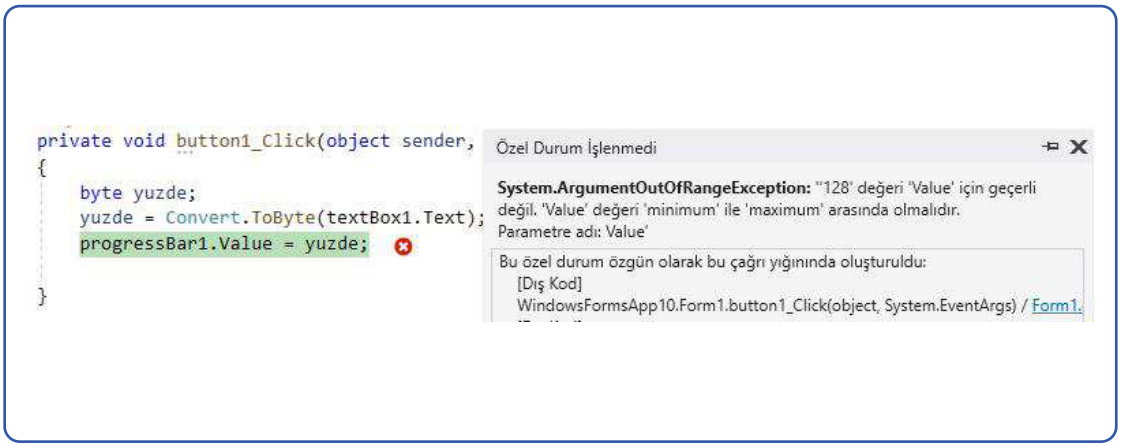
Not: Kullanıcı, yukarıdaki kodlar çalıştırıldıktan sonra textBox1 nesnesine sayı dışında bir değer girerse program hata verir veya sayinin_karesi değişkeninin alacağı değer int veri tipinin kapasite sınırlarının dışında olursa program yine hata verir. Her iki hata durumunda da catch blokundaki kodlar çalışacaktır.

Sıra Sizde

TextBox nesnesi içine girilen yüzde değerine göre ProgressBar nesnesinin değerini değiştiren programı hata ayıklama kodlarını kullanarak Görsel 2.25'te görüldüğü gibi tasarlayıp yazınız.



Görsel 2.25: Yüzde uygulaması



Görsel 2.26: Hatalı kod-2

Not: ProgressBar nesnesi 0 ile 100 arasındaki değerleri alır. Program, hata ayıklama kodları olmadan yazılırsa Görsel 2.26'daki gibi hata meydana gelecek ve program duracaktır.

ÖLÇME VE DEĞERLENDİRME - 2

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise (D), yanlış ise (Y) yazınız.

1. **( ) <,>,==,<=,>=,!= karakterleri mantıksal operatörlerdir.**
2. **( ) Üçten fazla if ifadesi iç içe kullanılmaz.**
3. **( ) Şart ifadesi sağlanmadığında çalışacak kodlar "else" blokunun içine yazılabilir.**
4. **( ) Karar ifadelerinde ve(&&) operatörü veya(||) operatöründen öncelikli işleme alınır.**
5. **( ) Döngü şart ifadesi sağlanmasa da for döngüsü en az bir kez çalışır.**

B) Aşağıda verilen cümlelerdeki boşlukları uygun ifadelerle tamamlayınız.

6. **Eğer "a" değişkeni "b" ve "c" değişkeninden büyükse ifadesini kod olarak yazınız.**

7. **Eğer "a" değişkeni "b" veya "c" değişkeninden küçükse ifadesini kod olarak yazınız.**

C) Aşağıda verilen cümlelerdeki boşlukları uygun ifadelerle tamamlayınız

8. **Aşağıdaki kodlar çalıştırıldığında comboBox1 nesnesine eklenecek değerler nelerdir?**

```
for (int i = 1; i <= 20; i = i + 1)
        {
           if (i % 3 == 0)
              {comboBox1.Items.Add(i); }
           if (i == 17)
              {comboBox1.Items.Add(i); }
        }
```

9. **Aşağıdaki kodlar çalıştırıldığında comboBox1 nesnesine eklenecek değerler nelerdir?**

```
for (int i = 1; i <= 15; i = i + 1)
        {
          if (i % 3 == 0)
        { continue; }
        comboBox1.Items.Add(i);
        }
```

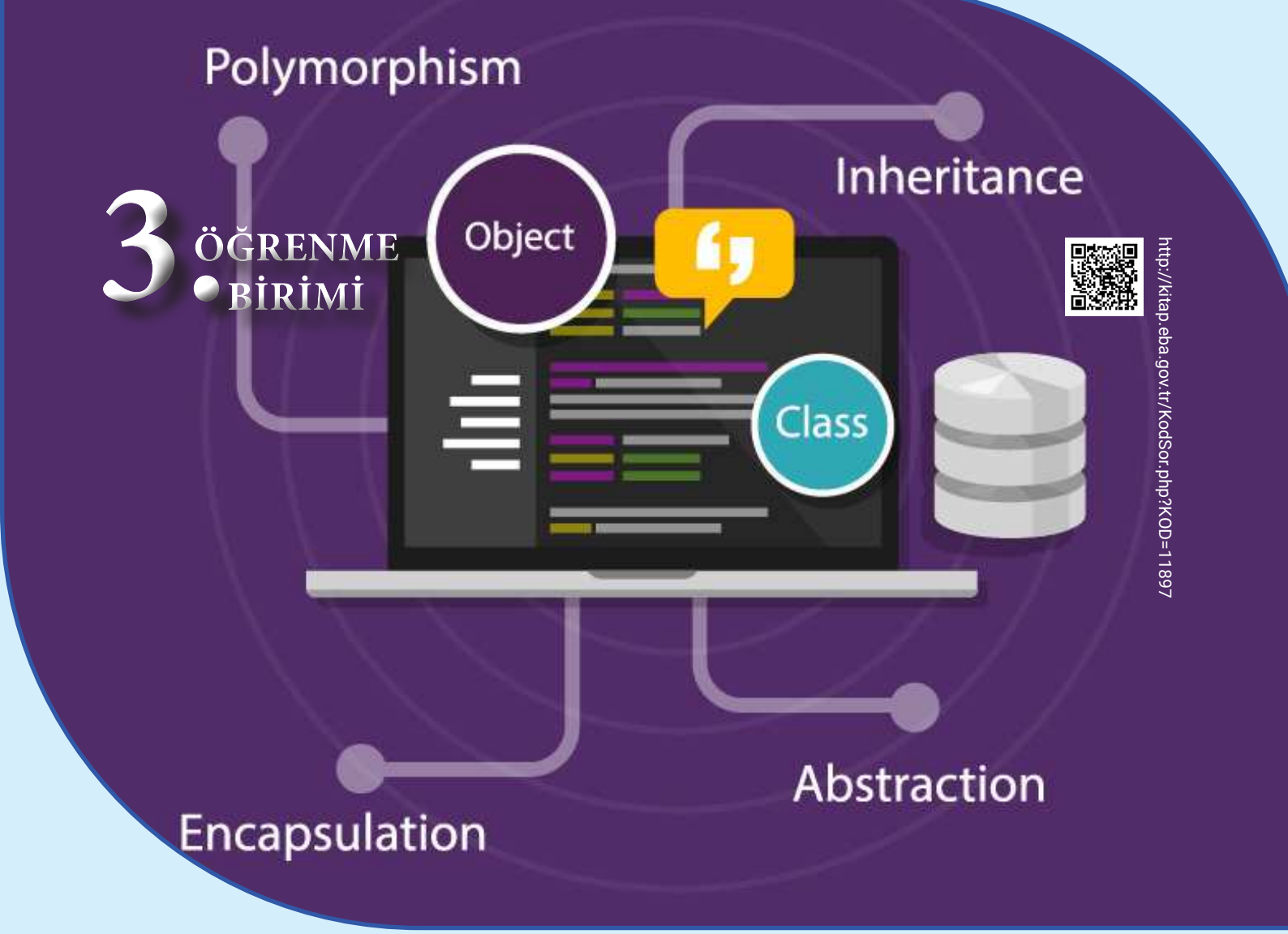
10. Aşağıdaki kodlar çalıştırıldığında oluşacak mesajları sırasıyla yazınız.

```
int sayac = 0,sonuc=1;
while (sayac<6)
{
sayac = sayac+1;
sonuc = sonuc * sayac;
MessageBox.Show(sayac.ToString()+ ".tur sonuç=" + sonuc.ToString());
}
```

11. Aşağıdaki kodlar çalıştırıldığında program nasıl bir yol izler?

```
try
{
int sayi;
sayi = textBox1.Text;
}
catch
{
MessageBox.Show("Hata bulundu!");
}
```

12. Üç farklı TextBox nesnesine girilen sayılardan hangisinin en büyük sayı olduğunu bulan programı yazınız.



3 ÖĞRENME BİRİMİ

http://kitap.eba.gov.tr/KodSor.php?KOD=11897

SINIFLAR (CLASSES)

NELER ÖĞRENECEKSİNİZ?

Bu öğrenme biriminde;

- Nesne tabanlı programlama mantığını kavrayacak,
- Nesne tabanlı programlamanın temel prensiplerini açıklayacak,
- Sınıf ve nesneler oluşturacak,
- Sınıflarda alan, özellik ve metot ögelerini kullanacak,
- Erişim belirleyicilerini sıralayacak,
- Metotlar yazacak ve kullanacak,
- Değer ve referans kavramlarını açıklayacak,
- Sınıflarda kalıtım özelliklerini kullanacak,
- Soyut sınıf, arayüz ve çok biçimlilik kavramlarını açıklayacak,
- Statik, isimsiz, mühürlü ve parçalı sınıfları kullanacak,
- Numaralandırma mantığını öğreneceksiniz.

ANAHTAR KELİMELER

Sınıf, nesne, alan, özellik, kapsülleme, metot, kalıtım, soyut sınıf, arayüz, çok biçimlilik, statik sınıf, isimsiz sınıf, mühürlü sınıf, parçalı sınıf, enums

HAZIRLIK ÇALIŞMALARI

1. Yapısal (prosedürel) programlama mantığı ile ilgili neler biliyorsunuz?
2. Nesne tabanlı programlamanın en çok tercih edilen yazılım geliştirme yaklaşımı olmasının nedeni neler olabilir?

3.1. Nesne Tabanlı Programlamaya Giriş

1960'lı yılların sonunda ortaya çıkan Nesne Tabanlı Programlama (NTP) kavramı, günümüzde yazılım geliştirmek, tasarlamak, analiz ve test etmek için kullanılır. NTP, öğrenilmesi gereken en önemli kavramlardan biridir. NTP bazı kaynaklarda "Nesne Yaklaşımlı Programlama", "Nesneye Yönelik Programlama", "Nesne Yönelimli Yaklaşım" olarak da kullanılmaktadır [(OOP) Object Oriented Programming-(OOA) Object Oriented Approach].

NTP, yazılımların boyutları ve karmaşıklığı arttığı için bazı yazılım gereksinimlerini karşılamak amacıyla doğmuştur. NTP, yazılan kodların bakımını ve aynı yazılım üzerinde birden fazla yazılımcının çalışmasını kolaylaştırmaktadır. Günümüzde geniş çaplı yazılım projelerinde yaygın olarak NTP kullanılmaktadır. NTP, genel olarak kullanılan fonksiyonel veya yapısal programlama tekniğine yeni bir bakış açısı getirmiştir. Günümüzde iyi bir yazılımcı olmak isteyen herkes NTP mantığını ve prensiplerini detaylı bir şekilde öğrenmelidir. Ayrıca bilinmesi gerekir ki şu an dünyada kullanılan -neredeyse- tüm programlama dilleri NTP'yi desteklemektedir.

3.1.1. NTP Öncesi

NTP öncesinde fonksiyonel programlama yaklaşımı mevcuttu. Fonksiyonel programlama, 1950'li yıllarda ilk üst düzey dillerin ortaya çıkışından bu yana kullanılagelmektedir. Hem yapısal programlama hem de NTP genel amaçlı tüm programlama dillerinde uygulanabilmektedir.

Yapısal programlama, büyük programları küçük parçalara bölerek çözümleme yöntemidir. Modüller ve alt programlar sıralı bir şekilde çalışmaktadır. Bu çalışma tarzı, ek bir programcılık yükü getirmekte ve kodların işleyişinin takibini zorlaştırmaktadır.

3.1.2. NTP Temel Prensipleri

NTP genel olarak dört prensip üzerine kurulmuştur. Herhangi bir programlama dilinin nesne tabanlı olduğundan söz edebilmek için asgari düzeyde şu prensipleri desteklemesi gerekmektedir:

a) Soyutlama (Abstraction): Karmaşıklığın azaltılması anlamına gelmektedir. Örneğin otomobillerde gaz pedalına basıldığında otomobil hızlanır ancak arka planda olan bitenler çoğu kişi için önemsizdir.

b) Sarmalama veya Kapsülleme (Encapsulation): Sadece istenilen bilgilerin dış dünyaya açılması, hassas veya özel bilgilerin gizlenmesi anlamına gelmektedir. "Banka hesabına para yatır." komutu verildikten sonra T.C. kimlik numarası ve şifre bilgilerinin gizlenmesi buna örnek verilebilir.

c) Kalıtım (Inheritance): Var olan özelliklerin aktarılması anlamına gelmektedir. Örneğin, bütün kedi türlerinin dört ayaklı olması ortak bir özelliktir. Bir Van kedisi, kedilerin tüm özelliklerini taşırken ayrıca kendine has özellikleri de barındırır.

ç)Çok biçimlilik (Polymorphism): Farklı tiplere ait olan ortak özellikleri tanımlama işlemidir. Örneğin, farklı hayvan türleri farklı sesler çıkarır zira "ses çıkarma" eylemi ortak bir özelliktir.

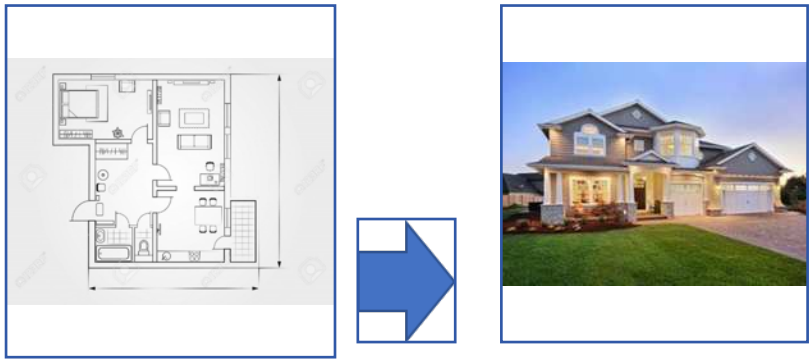
3.2. Sınıflar ve Nesneler

Çevremiz ve dünyamız incelendiğinde her şeyin (cisimler, canlılar vb.) belirli **özelliklerinin** ve **evlerinin** olduğu görülür. Her öğrencinin bir numarası, adı soyadı, aldığı dersler gibi **özellikleri** ve okula gitme, sınava girme gibi **işlevleri** vardır. Benzer şekilde yine bir cep telefonunun rengi, boyutları, markası, adı gibi özellikleri ve çağrı başlatma, mesaj gönderme, uygulama açma gibi işlevleri bulunmaktadır.

NTP, dünyada var olan her şeyin yazılım içinde modellenmesini amaçlamaktadır. Sınıf (class), NTP'nin en önemli kavramıdır. Sınıf, nesnelerin özelliklerini ve işlevlerini (davranışlarını) tanımlamak için kullanılan bir taslaktır. Bu taslak aracılığıyla **nesneler** (objects) oluşturulmaktadır.

Görsel 3.1'de bir ev planı görülmektedir. Programlamada bu ev planına sınıf, ev planından yola çıkılarak yapılan gerçek eve ise nesne adı verilebilir.

Çevrenizde gördüğünüz nesnelerden en az beş tanesinin özelliklerini ve işlevlerini yazınız. Yazdıklarınızı sınıf arkadaşlarınızla paylaşınız.



Görsel 3.1: Sınıf ve nesne ilişkisi

3.2.1. Sınıf Tanımlama

C#'ta sınıf tanımında class anahtar kelimesi kullanılmaktadır. Aşağıda en basit sınıf tanımı verilmiştir.

```
class SinifAdi
{
}
```

Not: public ve private kavramları 3.4. Erişim Belirleyicileri konusunda açıklanacaktır.

Aşağıdaki örnekte bir dikdörtgen sınıfı tanımı yapılmaktadır.

```
class Dikdortgen
{
private int a, b;
public Dikdortgen(int a, int b)
{
this.a = a;
this.b = b;
}
public int AlanHesapla()
{
return a * b;
}
public int CevreHesapla()
{
return 2 * (a + b);
}
}
```

Bir dikdörtgenin iki kenar uzunluğu bilgisi bulunmaktadır. Ayrıca çevre ve alan bilgilerinin hesabı da söz konusudur. Yukarıdaki sınıf tanımında a ve b değişkenleri dikdörtgenin kenar uzunluklarını saklamak için, **AlanHesapla()** ve **CevreHesapla()** işlevleri de dikdörtgenin alan ve çevre hesabının yapılması için tanımlanmıştır.

Not: C#'ta işlevleri gerçekleştiren kod bloklarına "metot" denilmektedir. Dolayısıyla bundan sonra işlev kelimesi yerine metot kelimesi kullanılacaktır.

Yukarıdaki kod blokunda aşağıdaki gibi bir sınıf tanımlaması yapılmıştır.

```
class Dikdortgen
```

Sınıfın adı "Dikdortgen"dir. Isimlendirmelerde genel prensip olarak "ı, I, ü, Ü, ö, Ö, ğ, Ğ, ş, Ş" gibi alfabemize özel harflerin kullanılmaması uygundur.

Daha genel yazılacak olursa bir sınıf aşağıdaki gibi tanımlanmaktadır.

```
«Erişim belirleyici» class  «Sınıf adı»
{
}
```

Sıra Sizde

Dikdortgen sınıfına benzer şekilde Daire sınıfını oluşturunuz.

3.2.2. Nesne Oluşturma

Programlarda sınıfların kullanılabilmesi için bu sınıftan oluşturulan nesnelere (object) gereksinim duyulur. Bu türetme işlemine örnek oluşturma (instance) adı da verilir.

C#'ta önceden tanımlanan bir sınıftan **nesne** türetmek için **new** anahtar kelimesi kullanılmaktadır. Daha önceden oluşturulan Dikdortgen sınıfından bir nesne türetmek ve bu nesnenin ögelerini (özellikler ve metotlar) kullanmak için aşağıdaki gibi bir konsol uygulaması yazılabilir.

```
private static void Main(string[] args)
{
Dikdortgen d = new Dikdortgen(3, 4);
Console.WriteLine("Dikdörtgenin alanı: {0}", d.AlanHesapla());
Console.WriteLine("Dikdörtgenin çevresi: {0}", d.CevreHesapla());
}
```

Oluşturulan nesnenin adı "**d**"dir. Kenar uzunlukları 3 ve 4 olarak belirlenmiştir. Nesnenin ögelerine erişmek için . (nokta) karakterinin kullanıldığı görülmektedir.

Genel olarak yazılacaksa bir nesne aşağıdaki gibi tanımlanmaktadır.

```
«Sınıf adı» «Nesne adı» = new «Sınıf adı»(««Parametre listesi»»);
```

Sıra Sizde

Dikdortgen nesnesine benzer şekilde yarıçapları farklı iki adet Daire nesnesi oluşturup metotlarını kullanınız.

3.3. Kapsülleme, Alanlar ve Özellikler (Encapsulation, Fields, Properties)

Erişim belirleyicileri ile NTP'nin temel prensiplerinden olan **"kapsülleme",** alanlar ve özellikler aracılığıyla programlarda uygulanabilmektedir. Kapsülleme, hassas veya özel bilgilerin gizlenmesi anlamına gelmektedir.

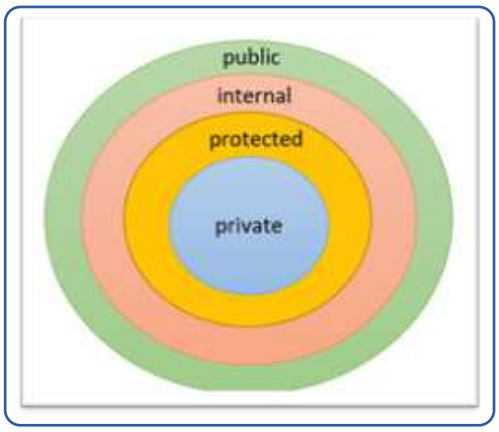
3.4. Erişim Belirleyiciler (Access Modifiers)

.NET platformunda oluşturulan uygulamalarda güvenliği artırmak amacıyla sınıflara ve/veya sınıf içinde kullanılan ögelere erişimin kısıtlanması gerekmektedir. Dolayısıyla koda dışarıdan erişimin sınırlarını belirlemek amacıyla erişim belirleyicileri kullanılır.

C# programlama dilinde kullanılan erişim belirleyicileri şunlardır:

- **public (Genel):** public olarak tanımlanan ögeler üzerinde herhangi bir kısıtlama yoktur. Her yerden erişilebilirdir.
- **private (Gizli):** En katı erişim belirleyicidir. Ögeler sadece tanımlandığı sınıf içinde erişilebilirdir. Bir başka deyişle ögeler sadece tanımlandığı küme parantezleri arasında kullanılabilir.
- **protected (Korunumlu):** Ögeler, bulunduğu sınıf içinde ya da bu sınıftan türeyen diğer sınıflarda erişilebilirdir.
- **internal (Dâhilî):** internal olarak tanımlanan ögelere sadece aynı program içinden erişilebilir.
- **protected internal (Dâhilî+Korumalı):** Ögeler hem protected hem de internal erişim belirleyicisine sahip olarak değerlendirilir. Türetilen sınıfın farklı program içinde olması sorun teşkil etmemektedir.

Gizliden genele doğru erişim belirleyicileri Görsel 3.2'deki hiyerarşiye sahiptir.



Görsel 3.2: Erişim belirleyicileri hiyerarşisi

Not: Bir ögeye herhangi bir erişim belirleyicisi tanımlaması yapılmazsa varsayılan olarak private olduğu kabul edilir.

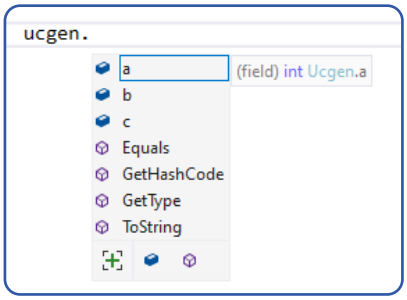
3.5. Alanlar (Fields)

Bir alan, sınıf içinde tanımlanmış herhangi türden (int, string vb.) bir değişkendir. Aşağıda Ucgen sınıfı ve bu sınıfa ait üç adet alan tanımlanmıştır.

```
public class Ucgen
{
  public int a;
  public int b;
  public int c;
  }
  internal class Program
  {
   private static void Main(string[] args)
   {
   Ucgen ucgen = new Ucgen();
   ucgen.a = 3;
   ucgen.b = 4;
   ucgen.c = 5;
   Console.WriteLine("Üçgenin a kenar uzunluğu: {0}", ucgen.a);
   Console.WriteLine("Üçgenin b kenar uzunluğu: {0}", ucgen.b);
   Console.WriteLine("Üçgenin c kenar uzunluğu: {0}", ucgen.c);
   }
}
```

Bu kod blokunda Ucgen sınıfından ucgen adında bir nesne türetilmiştir (C#'ın büyük / küçük harf duyarlı bir dil olduğu unutulmamalıdır. Dolayısıyla "U" ile "u" farklı karakterlerdir.).

Nesnenin alanlarına erişim için Görsel 3.3'te görüldüğü üzere . (nokta) karakteri kullanılmaktadır. Kod editöründe nesnenin adı yazıldıktan sonra . (nokta) karakterine basıldığında sınıfa ait kullanılabilir ögelerin listesi gelmektedir.



Görsel 3.3: Sınıf ögelerine erişim

Sınıfa ait alanlar tanımlanırken başına "public" erişim belirleyicisi yazılmıştır. Bu erişim belirleyicisi, alan bilgisine sınıf dışından erişim için gereklidir. Alanlar yalnızca özel ve gizli kalması gereken değişkenler için kullanılmalıdır. Sınıf içinde tanımlanmış bir değişkenin başına yazılan "public" erişim belirleyicisi ile alanı dış dünyaya açmak uygun değildir. Bu şekilde yapıldığında değişkene değer atama ya da değişkenin değerinin okunması işlemlerinde kontrol mekanizması işletilememektedir. Ucgen sınıfına ait bir değişkene yandaki değer ataması kolaylıkla yapılabilir.

```
Ucgen ucgen = new Ucgen();
ucgen.a = -3;
ucgen.b = 0;
ucgen.c = -12345;
```

Yukarıda belirtilen kenar uzunluklarına sahip bir üçgeni çizmek mümkün değildir. Bu da programın doğru çalışmayacağı anlamına gelir. Bu durumun önüne geçmek için **özellikler** kullanılır.

Sıra Sizde

1. Öğrenci sınıfını ve alanlarını oluşturunuz.
2. Bilgisayar sınıfını ve alanlarını oluşturunuz.

3.6. Özellikler (Propertles)

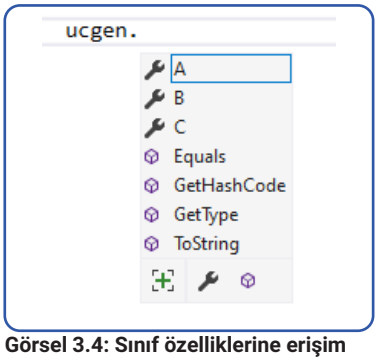
Bir değişkeni dış dünyaya açmak (diğer sınıflardan, programlardan vb. erişmek) için bu değişkene ait bir özellik eklenir. Bu sayece NTP'nin temel prensiplerinden "kapsülleme" prensibi sınıfa uygulanmış olur.

Aşağıda Ucgen sınıfının NTP prensiplerine daha uygun hazırlanmış örneği verilmiştir.

```
public class Ucgen
{
    int a;
    int b;
    int c;

    public int A
    {
        get { return a; }
        set
        {
            if (value <= 0)
                Console.WriteLine("Hatalı bilgi");
            else
                a = value;
        }
    }

    public int B
    {
        get { return b; }
        set
        {
            if (value <= 0)
                Console.WriteLine("Hatalı bilgi");
            else
                b = value;
        }
    }

    public int C
    {
        get { return c; }
        set
        {
            if (value <= 0)
                Console.WriteLine("Hatalı bilgi");
        else
                c = value;
        }
    }
}
```

Not: Değişkenlerin başında erişim belirleyicisinin olmadığına dikkat edilmelidir.

Görsel 3.4'teki gibi sınıf adı yazılıp . (nokta) karakterine basıldığında alanlar değil, özellikler görüntülenmektedir.



Görsel 3.4: Sınıf özelliklerine erişim

Not: Alan ve özellik simgelerinin farklı olduğuna dikkat edilmelidir.

Get (almak, elde etmek) ve **set** (düzenlemek, ayarlamak) şeklinde iki ayrı alt metodu bulunur.

- **get Metodu:** Bir değer döndürmek için kullanılır. Özelliklerin get metodunda **return** anahtar kelimesi kullanılarak "return...;" ile bir değerin döndürüleceği belirtilir.
- **set Metodu:** Değişkene değer atama işlemleri için kullanılır. Burada görülen **value** anahtar kelimesi dışarıdan bu özelliğe gönderilen değeri temsil etmektedir.

```
if (value <= 0)
Console.WriteLine("Hatalı bilgi");
```

Yukarıdaki kod satırları ile dışarıdan alınan bilginin kontrolü gerçekleştirilmektedir, sıfır veya negatif bir değer gönderildiğinde kullanıcıya hata mesajı gösterilmektedir.

Özellikler kullanılarak Ucgen sınıfının alanlarına değer atamak için aşağıdaki kod bloku yazılır.

```
Ucgen ucgen = new Ucgen();
ucgen.A = 3;
ucgen.B = 4;
ucgen.C = 5;
Console.WriteLine("Üçgenin a kenar uzunluğu: {0}", ucgen.A);
Console.WriteLine("Üçgenin b kenar uzunluğu: {0}", ucgen.B);
Console.WriteLine("Üçgenin c kenar uzunluğu: {0}", ucgen.C);
```

A kenarına negatif bir değer atanmak istenildiğinde kullanıcıya bir uyarı mesajı gösterilecektir.

```
private static void Main(string[] args)
{
   Ucgen ucgen = new Ucgen();
   ucgen.A = -3;
}

// Ekran çıktısı:
Hatalı bilgi
```

Programlarda özellikler sıklıkla kullanıldığı için kısa yoldan dâhilî bir değişken ve bu değişkene ait özellik tanımlanabilir.

```
public class Ucgen
{
   public int A { get; set; }
   public int B { get; set; }
   public int C { get; set; }
}
```

Nesne oluştururken ilk değer ataması da kısa yoldan yapılabilir.

```
static void Main(string[] args)
{
   Ucgen u = new Ucgen
   {
      A = 3,
      B = 4,
      C = 5
   };
}
```

Yapıcı metotları aşırı yüklemeye (Aşırı yükleme konusu 3.7.5'te incelenecektir.) gerek kalmadan doğrudan özelliklere değer ataması gerçekleştirilebilmektedir. Özelliklerin programcılara sunduğu bazı avantajlar vardır. Her özellik;

- Sadece **get** metoduna,
- Sadece **set** metoduna,
- Hem **get** hem **set** metoduna sahip olabilir.

Sıra Sizde

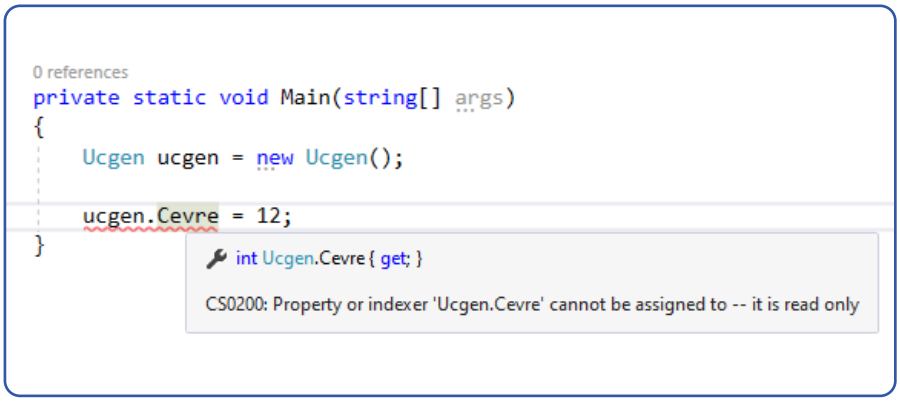
1. Öğrenci sınıfını ve özelliklerini oluşturunuz.
2. Bilgisayar sınıfını ve özelliklerini oluşturunuz.

3.6.1. Sadece Okunabilir Özellikler

Bir özellikte sadece **get** metodu kullanılırsa dışarıdan bu özelliğe değer ataması gerçekleştirilemez. Bu özellik = (eşittir) karakterinin solunda kullanılamaz. Bu özellik "sadece okunabilir" (readonly) bir özellik olur. Aşağıda sadece okunabilir bir özellik tanımı yapılmıştır.

```
public class Ucgen
{
   // ...

   public int Cevre
   {
      get
      {
         return a + b + c;
      }
   }
}
internal class Program
{
   private static void Main(string[] args)
   {
      Ucgen ucgen = new Ucgen();

      ucgen.A = 3;
      ucgen.B = 4;
      ucgen.C = 5;

      Console.WriteLine("Üçgenin çevresi: {0}", ucgen.Cevre);
   }
}
// Ekran çıktısı:
Üçgenin çevresi: 12
```

Buna karşın Cevre özelliğine değer atanmak istenirse bir hata oluşacaktır.



Görsel 3.5: Salt okunur özelliğe değer atamaya çalışma

Görsel 3.5'te görüldüğü gibi ucgen nesnesinin Cevre özelliğine değer atama işlemi gerçekleştirilememektedir.

3.6.2. Sadece Yazılabilir Özellikler

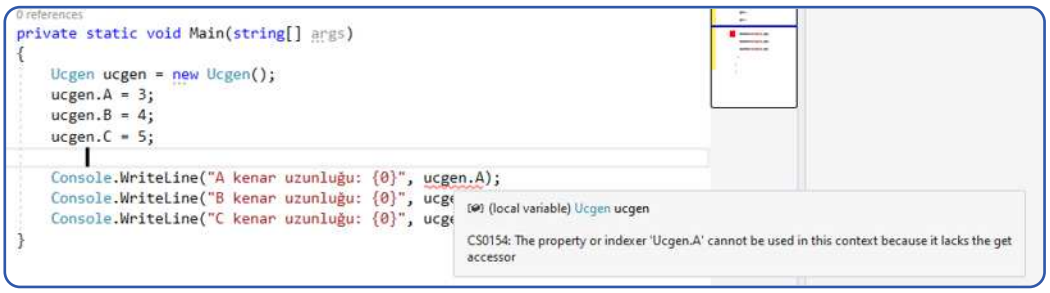
Bir özellik sadece **set** metoduna sahipse bu özelliğe "sadece yazılabilir" özellik denir. Bu özellik = (eşittir) karakterinin sağında kullanılamaz. Dolayısıyla bu tür özellikler değer döndürmez, sadece dışarıdan değer alabilir.

Daha önceden yazılan Ucgen sınıfının bir özelliği sadece yazılabilir özellik yapılmak istenirse bu özellik aşağıdaki gibi tanımlanmalıdır.

```
public class Ucgen
{
    int a;
    int b;
    int c;
    public int A
    {
        set
        {
            if (value <= 0)
                Console.WriteLine("Hatalı bilgi");
            else
                a = value;
        }
    }
    // ...
}

internal class Program
{
    private static void Main(string[] args)
    {
        Ucgen ucgen = new Ucgen();
        ucgen.A = 3;
        ucgen.B = 4;
        ucgen.C = 5;

        Console.WriteLine("A kenar uzunluğu: {0}", ucgen.A);
    }
}
```

Yukarıdaki kod parçasında üçgenin a, b ve c kenarlarına özelliklerin set metodu üzerinden değer ataması yapılabildiği ancak ekrana yazdırma esnasında get metodu olmayan A özelliğinden bir değer döndürülemediği görülmektedir.

Kod editöründe bu hata mesajı Görsel 3.6'daki gibi görülmektedir.



Görsel 3.6: Sadece yazılabilir özelliği okumaya çalışma

3.7. Metotlar (Methods)

Bir metot, yalnızca çağrıldığında çalışan ve bir dizi ifade içeren kod blokudur. Yazılım dünyasında bir metot, sınıf içinde yapılacak işlerin veya operasyonların tanımlanmasını sağlar. Metotlara parametreler aracılığıyla ana programdan veriler gönderilebilir ve metotlar çalışmasını bitirdikten sonra ana programa değer döndürülebilir.

Genel olarak bir metot aşağıdaki şekilde tanımlanmaktadır.

```
«Erişim belirleyici» «Dönüş tipi» «Metodun adı» («Parametre listesi»)
{
}
```

- **Erişim Belirleyici:** Metoda nerelerden erişilebileceğini tanımlar.
- **Dönüş Tipi:** Metotlar değer döndürebilir. Dönüş tipi, metodun döndüreceği değerin tipini tanımlar (int, string vb.). Değer döndürmek için **return** anahtar kelimesi kullanılır. Metot değer döndürmeyecekse dönüş tipi olarak **void** anahtar kelimesi kullanılır.
- **Metot Adı:** Metodun adını tanımlamada kullanılır.
- **Parametre Listesi:** Parantez içinde verilen parametreler, bir metoda değer göndermek veya metottan değer almak için kullanılır. Parametrelerin türü, sayısı ve sırası parametre listesi olarak adlandırılır.

Bunlardan metodun adı ve parametre listesi **metodun imzası**, küme parantezi { } arasına yazılan kodlar da **metodun gövdesi** olarak adlandırılır. Metot imzaları, bir sınıf içinde her metotta farklı olmak zorundadır.

Metot adı + Parametre listesi = Metodun imzası

Aşağıda örnek metot tanımları verilmiştir.

```
public void ProgramiKapat() { ... }
public void DaireCiz(double x, double y, double cap) { ... }
public int KareKokHesapla(int sayi) { ... }
public void SMSGonder(string cepNo, string mesaj) { ... }
public decimal MaasHesapla(int gunSayisi, decimal gundelikUcret) { ... }
```

Bir metodu ana programdan çağırmak için metodun adı ile ( ) parantez açma ve kapatma karakterleri kullanılmalıdır.

```
nesneAdi.ProgramiKapat();
nesneAdi.DaireCiz(3.2, 2.4, 3);
int kareKok = nesneAdi.KareKokHesapla(9);
nesneAdi.SMSGonder("5051234567", "Merhaba, nasılsın?");
decimal maas = nesneAdi.MaasHesapla(24, 90.25M);
```

Verilen iki sayıdan büyük olanını bulan bir metot yanda yazılmıştır.

```
class SayiBulucu
{
    public int BuyukOlaniBul(int sayi1, int sayi2)
    {
        int sonuc;
        if (sayi1 < sayi2)
            sonuc = sayi2;
        else
            sonuc = sayi1;
        return sonuc;
    }
}
class Program
{
    static void Main(string[] args)
    {
        SayiBulucu sb = new SayiBulucu();
        int a = 100;
        int b = 25;
        int sonuc = sb.BuyukOlaniBul(a, b);
        Console.WriteLine("Büyük olan sayı: {0}", sonuc);
    }
}
```

SayiBulucu adlı sınıfın içindeki metotta;

- Erişim belirleyicisi **"public"**,
- Dönüş tipi **"int"**,
- Metot adı **"BuyukOlaniBul"**,
- Parametre listesi **"(int sayi1, int sayi2)"**,
- Metot imzası **"BuyukOlaniBul(int sayi1, int sayi2)"** olduğu söylenebilir.

Metottan değer döndürmek için **return** anahtar kelimesi kullanılır.

Sıra Sizde

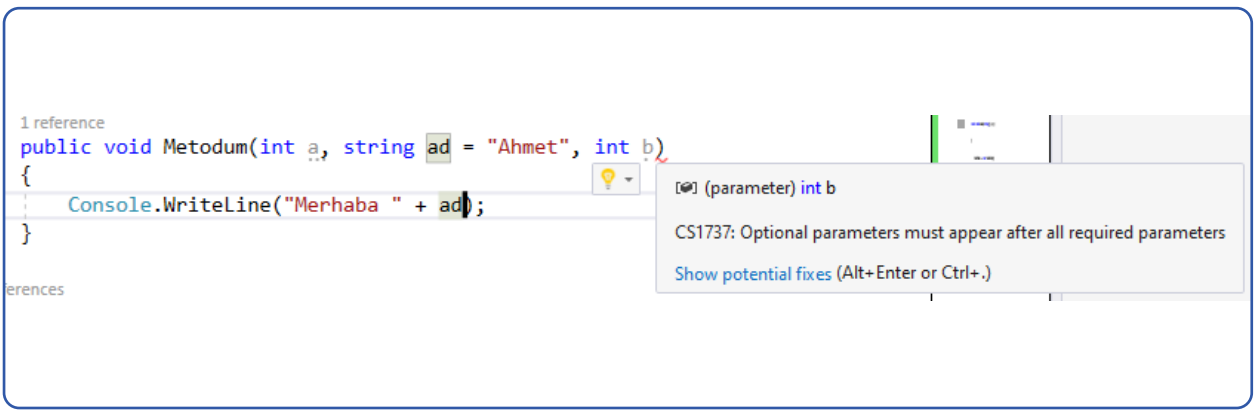
1. Parametresinde verilen sayının değeri tek ise true, çift ise false döndüren metodu yazınız.
2. Klavyeden 0 (sıfır) girilene kadar bu değerleri toplayıp döndüren metodu yazınız.

3.7.1. Varsayılan Değerli Parametreler (Optional Parameters)

Metot parametreleri tanımlanırken istendiğinde bunlara "varsayılan değerler" atanabilmektedir. Metot çağrılırken bu parametrelere değer ataması yapılmazsa varsayılan değerler kullanılır.

```
class Sinifim
{
   public void Selamla(string ad = "Emre")
   {
      Console.WriteLine("Merhaba " + ad);
   }
}

class Program
{
   static void Main(string[] args)
   {
      Sinifim s = new Sinifim();
      s.Selamla();
      s.Selamla("Defne");
   }
}

// Ekran çıktısı:
Merhaba Emre
Merhaba Defne
```

Birden fazla parametre kullanıldığında varsayılan değere sahip parametreler, parametre listesinin en sonunda yer almalıdır. Aksi takdirde derleyici hatası oluşur (Görsel 3.7).



Görsel 3.7: Varsayılan parametrelerin hatalı kullanımı

3.7.2. İsimlendirilmiş Parametreler (Named Parameters)

Bir metot oluşturulurken tanımlanan parametrelere değer atamak için ana programda parametreler sırasıyla yazılmalıdır. İstenilirse parametre isimleri kullanılarak bu sıralamaya uyulmayabilir.

```
class SayiIslemleri
{

   public int Topla(int sayi1, int sayi2, int sayi3)
   {
      return sayi1 + sayi2 + sayi3;
   }
}

class Program
{
   static void Main(string[] args)
   {

      SayiIslemleri si = new SayiIslemleri();
      // int toplam = si.Topla(5, 10, 15);
   int toplam = si.Topla(sayi2: 10, sayi3: 15, sayi1: 5);
      Console.WriteLine("Toplam: {0}", toplam);
   }
}
```

Yukarıdaki kodda metot çağrılırken parametre adının ardından : (iki nokta) karakteri kullanılarak parametre sırasına uymadan atama yapıldığı görülmektedir.

3.7.3. Parametre Dizileri

Parametre, metodun parametre sayısının bilinmediği durumlarda **params** anahtar kelimesi ile tanımlanır.

```
class SayiIslemleri
{
   public int Toplam(params int[] sayilar)
   {
      int toplam = 0;
      foreach (var s in sayilar)
      {
         toplam += s;
      }
      return toplam;
   }
}

class Program
{
   static void Main(string[] args)
   {
      SayiIslemleri si = new SayiIslemleri();

      Console.WriteLine("Toplam: {0}", si.Toplam(3));
      Console.WriteLine("Toplam: {0}", si.Toplam(3, 4, 5));
      Console.WriteLine("Toplam: {0}", si.Toplam(5, 1, 7, 3, 4, 5));
   }
}
// Ekran çıktısı:
Toplam: 3
Toplam: 12

Toplam: 25
```

params anahtar kelimesi ile bir parametre tanımlanacaksa bu parametre, metodun en son parametresi olmalıdır. Aksi takdirde derleyici hata verecektir. Ayrıca metotlarda sadece bir adet params tipinde parametre tanımlanabilir.

3.7.4. Metodu Sonlandırma

Dönüş tipi **void** olan bir metodun çalıştırılması, istenildiği an sonlandırılabilir. Bunun için **return** anahtar kelimesi kullanılır.

```
class EkranIslem
{
    public void EkranaYaz(params int[] sayilar)
    {
        if (sayilar.Length == 0)
        {
            Console.WriteLine("Parametre olmadığı için metottan çıkılıyor.");
            return;
        }
        Console.WriteLine("Parametreden gelen değerler:");
        foreach (var s in sayilar)
        {
            Console.WriteLine(s);
        }
    }
}

class Program
{
    static void Main(string[] args)
    {
        EkranIslem ei = new EkranIslem();
        ei.EkranaYaz(3, 4, 5);
        Console.WriteLine("========================");
        ei.EkranaYaz();
    }
}

// Ekran çıktısı:
Parametreden gelen değerler:
3
4
5
========================
Parametre olmadığı için metottan çıkılıyor.
```

Yukarıdaki kod blokunda **EkranaYaz** metodunun içinde **return** ifadesi ile metodun çalıştırılması sonlandırılmıştır.

Sıra Sizde

Yukarıdaki kod blokunda EkranaYaz metodunda verilen parametrelerden herhangi biri 0 (sıfır) ise programın çalışmasını sonlandıran değişikliği yapınız.

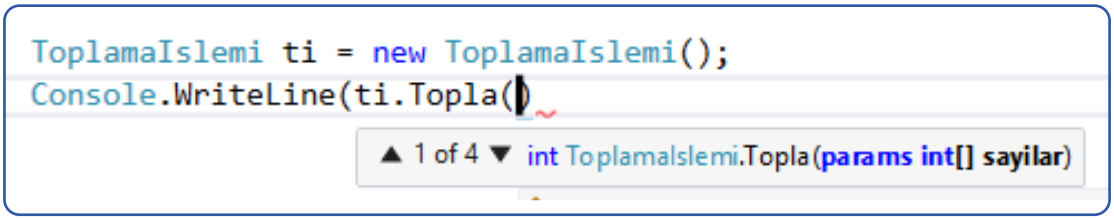
3.7.5. Metot Aşırı Yüklemeleri (Method Overloads)

Metodun adı aynı kalmak şartıyla parametre tipleri ve/veya sayısı değiştirilerek farklı metot imzaları oluşturulabilir. Bu durumda aynı isimli birden fazla metot oluşacaktır. Aşağıdaki sınıfta farklı türden parametreler alan **Topla** isimli metodun birden fazla kez oluşturulduğu görülmektedir.

```
class ToplamaIslemi
{
    public int Topla(int a, int b)
    {
        Console.WriteLine("int parametreli metot çağrılıyor.");
        return a + b;
    }

    public int Topla(params int[] sayilar)
    {
        Console.WriteLine("params parametreli metot çağrılıyor.");
        int toplam = 0;
        foreach (var s in sayilar)
        {
            toplam += s;
        }
        return toplam;
    }

    public double Topla(double a, double b)
    {
        Console.WriteLine("double parametreli metot çağrılıyor.");
        return a + b;
    }

    public string Topla(string a, string b)
    {
        Console.WriteLine("string parametreli metot çağrılıyor.");
        return a + b;
    }
}
class Program
{
    static void Main(string[] args)
    {
    ToplamaIslemi ti = new ToplamaIslemi();
        Console.WriteLine(ti.Topla(2, 5));
        Console.WriteLine("==============================");
        Console.WriteLine(ti.Topla(3.3, 5.1));
        Console.WriteLine("==============================");
        Console.WriteLine(ti.Topla("Sağlıcakla ", "kalınız."));
        Console.WriteLine("==============================");
        Console.WriteLine(ti.Topla(3, 8, 3, 7, 12, 33, 11, 4));
    }
}

// Ekran çıktısı:
int parametreli metot çağrılıyor.
7
==============================
double parametreli metot çağrılıyor.
8,4
==============================
string parametreli metot çağrılıyor.
Sağlıcakla kalınız.
==============================
params parametreli metot çağrılıyor.
81
```

Her bir Topla metodunun imzası farklı olduğu için derleyici hata vermemekte ve program başarılı bir şekilde çalıştırılmaktadır.

Not: Görsel 3.8'deki kod editöründe metot adından sonra kullanılabilir parametre tipleri listelenmiştir.



Görsel 3.8: Metot aşırı yüklemelerini kullanma

Burada Topla metodunun dört farklı aşırı yüklenmiş hâli olduğu belirtilmektedir. Klavyeden yukarı ve aşağı tuşları ile parametre tipleri incelenebilir.

Sıra Sizde

Yaş hesaplayan bir metodu DateTime (doğum tarihi) ve int (doğum yılı) parametrelerini alacak şekilde aşırı yükleyerek gerçekleştiriniz.

3.8. Yapıcı ve Yıkıcı Metotlar

Nesneler oluşturulduğunda ve yok edilme anında otomatik olarak çalıştırılan metotlar bulunmaktadır.

Nesneler oluşturulurken otomatik olarak çalıştırılan metotlara **yapıcı metot** (constructor), nesnelerin yok edildiği anda otomatik olarak çalıştırılan metotlara **yıkıcı metot** (destructors) denilmektedir.

3.8.1. Yapıcı Metotlar (Constructors)

Yapıcı metotlar, nesnelerin ilk oluşturulduğu anda otomatik olarak çalıştırılır. Yapıcı metotlar genellikle sınıf içinde tanımlanan yerel değişkenlerin ilk değerlerini düzenlemek için kullanılır.

Bir metodun yapıcı metot olabilmesi için şu şartları taşıması gerekir:

- Metot adı, sınıfın adı ile aynı olmalıdır.
- Geri dönüş tipi olmamalıdır (void ya da int gibi).
- Nesneleri oluşturmak için **new** operatörü kullanıldığı anda yapıcı metotlar otomatik olarak çalıştırılır. Nesne oluşturulduktan sonra yapıcı metotlar bir daha çağrılamaz.

Not: Sınıf içinde bir yapıcı metot tanımlanmamışsa derleyici arka planda boş bir varsayılan yapıcı metot oluşturmaktadır.

Not: Yapıcı metotlar bazı kaynaklarda "kurucu metot" veya "oluşturucu metot" olarak da geçmektedir.

Aşağıda Kisi sınıfında tanımlanan iki adet yerel değişkene yapıcı metot içinde değer ataması yapılmıştır.

```
class Kisi
{
    int yas;
    string ad;
    public Kisi()
    {
        yas = 19;
        ad = "Ahmet";
        Console.WriteLine("Yapıcı metot çalıştı.");
    }

    public int Yas
    {
        get
        {
            return yas;
        }
    }

    public string Ad
    {
        get
        {
            return ad;
        }
    }

}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Program başladı.");
        Kisi k = new Kisi();
        Console.WriteLine("Adı: {0}, Yaşı: {1}", k.Ad, k.Yas);
        Console.WriteLine("Program bitti.");
    }
}

// Ekran çıktısı:
Program başladı.
Yapıcı metot çalıştı.
Adı: Ahmet, Yaşı: 19
Program bitti.
```

Yukarıdaki kod parçasında **new Kisi()** komutu işletildiği anda yapıcı metodun çalıştırıldığı görülmektedir. Yapıcı metotları da aşırı yüklemek mümkündür.

Yukarıdaki **Kisi** sınıfı şu şekilde de tanımlanabilir.

```
class Kisi
{
   int yas = 0;
   string ad = "";

   public Kisi()
   {
      yas = 19;
      ad = "Ahmet";
      Console.WriteLine("Yapıcı metot çalıştı.");
   }
   public Kisi(int yas)
   {
      this.yas = yas;
      ad = "Ahmet";
      Console.WriteLine("int parametreli yapıcı metot çalıştı.");
   }

   public Kisi(string ad)
   {
      yas = 19;
      this.ad = ad;
      Console.WriteLine("string parametreli yapıcı metot çalıştı.");
   }

   public Kisi(int yas, string ad)
   {
      this.yas = yas;
      this.ad = ad;
      Console.WriteLine("İki parametreli yapıcı metot çalıştı.");
   }
}

class Program
{
   static void Main(string[] args)
   {
      Console.WriteLine("Program başladı.");
      Kisi k1 = new Kisi();
      Kisi k2 = new Kisi(23);
      Kisi k3 = new Kisi("Filiz");
      Kisi k4 = new Kisi(25, "Süleyman");
      Console.WriteLine("Program bitti.");
   }
}
```

```
// Ekran çıktısı:
Program başladı.
Yapıcı metot çalıştı.
int parametreli yapıcı metot çalıştı.
string parametreli yapıcı metot çalıştı.
İki parametreli yapıcı metot çalıştı.
Program bitti.
```

Yapıcı metot içinde kullanılan **this** anahtar kelimesi, bu sınıftan türeyen nesneyi temsil etmektedir. Dolayısıyla parametre adı ile sınıf değişkeninin adlarının aynı olması durumunda **this** anahtar kelimesi ile bu sınıftan türetilen nesnenin ilgili değişkenini kullanmak mümkündür.

```
public Kisi(int yas)
{
   this.yas = yas;
}
```

Parametreden gelen **yas** bilgisi, **"this.yas"** ifadesi ile bu sınıftan türetilen nesnenin **yas** alanına atanmaktadır.

Sıra Sizde

Daire sınıfını yapıcı metodu ile beraber tanımlayınız (Yarıçap değerini almalıdır.).

Sıra Sizde

Ev sınıfını yapıcı metodu ile beraber tanımlayınız (Oda sayısı ve m^2 değerlerini almalıdır.).

3.8.2. Yıkıcı Metotlar (Destructors)

Nesne hafızadan atıldığı anda otomatik olarak çalışan yıkıcı metotlar, tıpkı yapıcı metotlar gibi özel metotlardır ve aşağıdaki şu şartları taşımalıdır:

- Metot adı, sınıfın adı ile aynı olmalıdır.
- Metot adının başında ~ (Tilde) karakteri olmalıdır.
- Bir sınıfın yalnızca bir tane yıkıcı metodu olabilir.
- Yıkıcı metotlar aşırı yüklenemez.
- Yıkıcı metotlar parametre alamaz.
- Yıkıcı metotların erişim belirleyicisi olamaz.

Programcının yıkıcı metot üzerinde bir kontrolü bulunmamaktadır. Yıkıcı metotlar genellikle sınıf içinde kullanılan kaynakların (veri tabanı, dosya vb.) kapatılması ve hafızadan atılması amacıyla .**NET Framework** içindeki **Garbage Collector** (Çöp Toplayıcısı) tarafından gerekli görüldüğü zaman çalıştırılır.

Aşağıda yıkıcı metoda sahip bir sınıf örneği verilmiştir.

```
class Otomobil
{
    string marka = "";
    string renk = "";
    public Otomobil()
    {
        marka = "TOGG";
        renk = "kırmızı";
        Console.WriteLine("Yapıcı metot çalıştı.");
    }

    ~Otomobil()
    {
        Console.WriteLine("Nesne hafızadan atıldı.");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Program başladı.");
        Otomobil oto = new Otomobil();
      Console.WriteLine("Program bitti.");
    }
}
// Ekran çıktısı:
Program başladı.
Yapıcı metot çalıştı.
Program bitti.
Nesne hafızadan atıldı.
```

Yıkıcı metot, ekran çıktısından da görüleceği gibi programın sona ermesinden sonra çalıştırılmaktadır.

Sıra Sizde

Yıkıcı metotların kullanım yerlerini araştırınız, edindiğiniz bilgileri sınıf arkadaşlarınızla paylaşınız.

3.9. Değer ve Referans Tipler

.NET platformunda hafıza yönetiminin nasıl işlediğinin bilinmesi, programcılar için oldukça önemlidir. .NET'te hafıza, **yığın** (stack) ve **öbek** (heap) olmak üzere iki bölgeye ayrılmıştır.

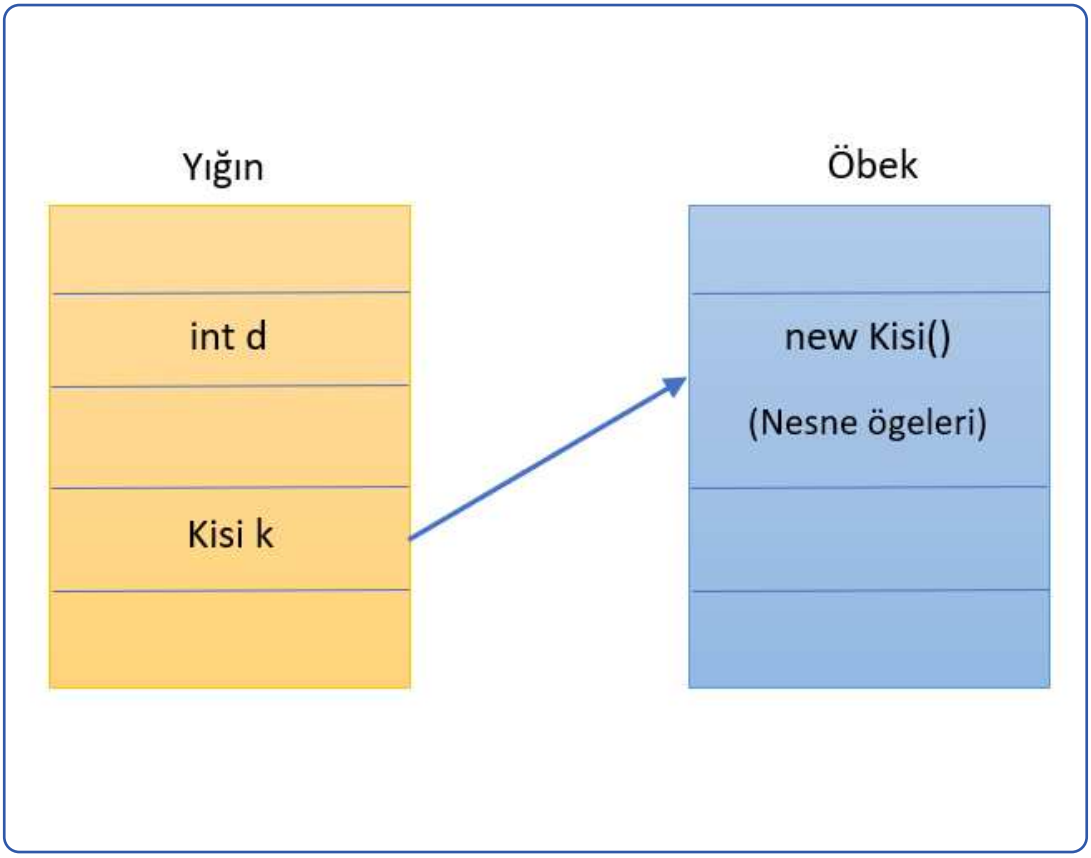
Değişkenler, veri tipine göre atanan değerleri taşıyan veri tutuculardır. .NET platformunda kullanılan her bir veri tipi **değer tipli** ve **referans tipli** olarak ikiye ayrılmaktadır. Bu veri tiplerinden değer veri tipleri hafızanın yığın bölgesinde, referans veri tipleri de hafızanın öbek bölgesinde tutulur.

Değer Tipleri: int, long, float, double, decimal, char, bool, byte, short, struct, enum

Referans Tipleri: string, object, class, interface, array, delegate

Not: Bunlardan **string** veri tipi teorikte referans tipli olmasına rağmen program içinde değer tipli olarak işlem görmektedir.

Değer tipleri veriyi bizzat barındıran türlerdir, referans tipleri ise veri yerine verinin bellekteki adresini tutan türlerdir (Görsel 3.9).



Görsel 3.9: Yığın ve öbek ilişkisi

Değer tiplerinden biri kullanılarak bir değişken tanımlandığında değişkenin değeri yığın bellek bölgesinde tutulur. Referans tipte bir değişken tanımlandığında ise değişkenin değeri öbek hafıza bölgesinde tutulur ve bu bölgenin adresini tutan bilgi de yığında saklanır. Yığında öbek bölgesini işaret eden bir "işaretçi" (pointer) oluşturulur.

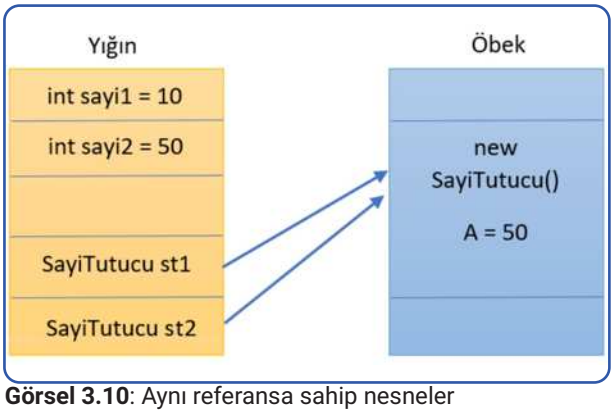
Aşağıdaki örnekte değer ve referans tiplerin çalışma mantıkları verilmiştir.

```
class SayiTutucu
{
   public int A { get; set; }
}
class Program
{
   static void Main(string[] args)
   {
      int sayi1 = 10;
      int sayi2 = sayi1;
      sayi2 = 50;
      Console.WriteLine("sayi1 = {0}", sayi1);
      Console.WriteLine("sayi2 = {0}", sayi2);
      Console.WriteLine("====================");
      SayiTutucu st1 = new SayiTutucu();
      st1.A = 10;
      SayiTutucu st2 = st1;
      st2.A = 50;
      Console.WriteLine("st1.A değeri: {0}", st1.A);
      Console.WriteLine("st2.A değeri: {0}", st2.A);
   }
}
// Ekran çıktısı:
sayi1 = 10
sayi2 = 50
====================
st1.A değeri: 50
st2.A değeri: 50
```

Main metodu çalıştırıldığında sırasıyla şu işlemler gerçekleşir:

- **int sayi1 = 10;** => sayi1 için yığında alan ayrılır ve buraya 10 değeri yazılır.
- **int sayi2 = sayi1;** => sayi2 için yığında alan ayrılır ve bu alana sayi1'in değeri 10 yazılır.
- **sayi2 = 50;** => sayi2 için ayrılan alana 50 değeri yazılır. sayi1 değişkeninin değeri değişmez.
- **SayiTutucu st1 = new SayiTutucu();** => Oluşturulan nesne için öbekte bir alan ayrılır ve nesnenin ögeleri burada saklanır. Bu alanın adresi st1 değişkeninde tutulur. Bu değişken için de yığında bir yer ayrılır.
- **st1.A = 10;** => st1'in gösterdiği öbek alanındaki nesnenin A özelliğinin değeri 10 olarak güncellenir.
- **SayiTutucu st2 = st1;** => st2 için yığında ayrı bir yer ayrılır ve buraya st1 değişkeninin tuttuğu
- adres bilgisi yazılır. Dolayısıyla st1 ve st2 aynı öbek alanının adresini tutmaktadır.
- **st2.A = 50;** => st2'nin gösterdiği öbek alanındaki nesnenin A özelliğinin değeri 50 olarak güncellenir.

st1 ve st2 aynı öbek alanının adresini tuttuğu için hangi değişken üzerinden olduğu fark etmeksizin aynı nesnenin A özelliğinin değeri ekrana yazdırılacaktır (Görsel 3.10).



Görsel 3.10: Aynı referansa sahip nesneler

3.9.1. Metotlarda ref ve out Kullanımı

Değer tipli değişkenler metotlara parametre olarak gönderildiğinde bu değişkenin değeri için yığında farklı bir bellek alanı ayrılır. Dolayısıyla metot içinde bu değişkenin değeri değiştirilse bile değişiklik ana programdan gönderilen değişkeni etkilemeyecektir. Aşağıdaki kod parçasında bu durum verilmiştir.

```
class Matematik
{
    public void Artir(int x)
    {
        x++;
    }
}

class Program
{
    static void Main(string[] args)
    {
        Matematik m = new Matematik();
        int a = 100;
        m.Artir(a);
        Console.WriteLine("a değeri = {0}", a);
    }
}
// Ekran çıktısı:
a değeri = 100
```

Artir metodunun içinde x değişkeninin değeri 1 artırılmasına rağmen bu değişikliğin ana metot içinde kullanılan a değişkeninin değerine bir etkisi olmamaktadır çünkü ana metot içindeki a değişkeni ile Artir metodunda kullanılan x değişkeni için yığında farklı hafıza alanları ayrılır ve bunlar birbirlerinden tamamen ayrı iki değişken olarak düşünülmelidir.

Parametre olarak gönderilen değişkenin değeri Artir metodunun içinde değiştirilmek istenirse bu durumda **ref** veya **out** anahtar kelimeleri kullanılmalıdır.

Bir önceki kod parçacığı ref anahtar kelimesi kullanılarak tekrar yazılırsa ana metot içindeki değişkenin değerinin değiştiği görülecektir.

```
class Matematik
{
    public void Artir(ref int x)
    {
        x++;
    }
}

class Program
{
    static void Main(string[] args)
    {
    Matematik m = new Matematik();
        int a = 100;
    m.Artir(ref a);
        Console.WriteLine("a değeri = {0}", a);
    }
}
// Ekran çıktısı:
a değeri = 101
```

Yukarıdaki kodda Artir metodu yazılırken ve bu metot çağrılırken ref anahtar kelimesinin kullanıldığına dikkat edilmelidir. ref veya out anahtar kelimeleri ile metoda parametre gönderildiğinde aslında değişkenin değeri değil, değişkenin yığında bulunduğu hafıza adresi gönderilmektedir. Dolayısıyla metot içinde değişken üzerinde yapılan değişiklikler aslında ana metotta kullanılan değişken üzerinde gerçekleşmektedir.

out anahtar kelimesinin kullanımı, ref anahtar kelimesinin kullanımından biraz farklıdır. ref kullanılmak istendiğinde parametre olarak göndermeden önce değişkene mutlaka bir değer ataması yapılmalıdır. out kullanıldığında ise değişkenin değeri artırma veya azaltma gibi bir işleme tabi tutulmadan önce değişkene mutlaka değer ataması yapılmalıdır.

Yukarıdaki kod parçası out anahtar kelimesi ile tekrar yazılmıştır.

```
class Matematik
{
   public void Artir(out int x)
   {
      x = 123;
   }
}

class Program
{
   static void Main(string[] args)
   {
      Matematik m = new Matematik();
      int a;
      m.Artir(out a);
      Console.WriteLine("a değeri = {0}", a);
   }
}
// Ekran çıktısı:
a değeri = 123
```

Tablo 3.1'de ref ve out anahtar kelimeleri arasındaki farklar listelenmiştir.

Tablo 3.1: ref ile out Arasındaki Farklar

ref	out
Metodu tanımlarken parametrenin önüne "ref" yazılmalıdır.	Metodu tanımlarken parametrenin önüne "out" yazılmalıdır.
Metodu çağırırken değişkenin önüne "ref" yazılmalıdır.	Metodu çağırırken değişkenin önüne "out" yazılmalıdır.
Metoda göndermeden önce değişken başlangıç değeri almak zorundadır.	Metoda göndermeden önce değişken başlangıç değeri almak zorunda değildir.
Metot içinde istenildiği gibi kullanılabilir.	Metot içinde mutlaka bir değer ataması gerçekleştirilmelidir.

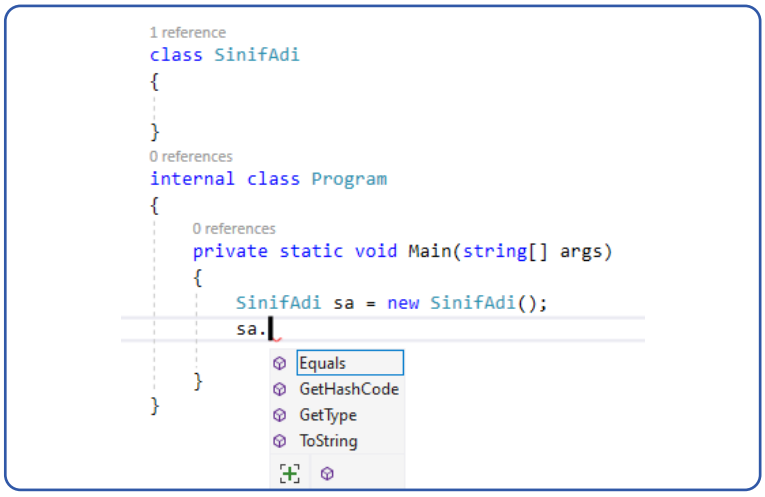
Sıra Sizde

Siz de ref veya out kullanarak ad, soyad ve yaş bilgilerini döndüren metodu yazınız.

3.10. Kalıtım (Inheritance)

Kalıtım, NTP'deki en önemli kavramlardan biridir ve bir sınıfın özelliklerinin farklı sınıflar tarafından da kullanılabilmesini sağlar. Buna "miras alma" da denilmektedir. Bu durumda miras alınan sınıfa üst veya temel (parent) sınıf, miras alan sınıfa da türetilmiş (derived) sınıf denir. C#'ta bir sınıf sadece bir sınıftan türetilebilir.

Bu durumun tek istisnası **Object** sınıfıdır. Bir sınıf, başka bir sınıftan türesin veya türemesin, varsayılan olarak **Object** sınıfından türer. Buna "örtük devralma" denir. Boş bir sınıftan oluşturulan nesnelerin sahip olduğu metotlar **Object** sınıfından gelmektedir (Görsel 3.11).



Görsel 3.11: Object sınıfından miras alınan ögeler

Üst sınıf ile türetilmiş sınıf arasında bir üst / alt ilişkisi vardır. Türetilmiş sınıf, üst sınıf ögelerine erişebilir ancak bu durumun tersi doğru değildir.

Bir sınıfı bir başka sınıftan türetmek için : (iki nokta) karakteri kullanılır.

```
«Erişim belirleyici» class «Sınıf adı» : «Üst sınıf adı»
{
}
```

Aşağıda **OkulPersoneli** sınıfı ve bu sınıftan türetilen bir **Ogretmen** sınıfı tanımlanmıştır.

```
public class OkulPersoneli
{
   public string Ad { get; set; }
   public string Soyad { get; set; }
}

public class Ogretmen : OkulPersoneli
{
   public string Brans { get; set; }

}
class Program
{
   static void Main(string[] args)
   {
      Ogretmen ogr = new Ogretmen
      {
         Ad = "Ahmet",
         Soyad = "Öz",
         Brans = "Matematik"
      };
      // ..
   }
}
```

Bu kod parçasında Ad ve Soyad özellikleri Ogretmen sınıfına aktarılmıştır. Ogretmen sınıfı, Ad ve Soyad bilgilerini OkulPersoneli sınıfından miras almıştır.

Miras alma işlemlerinde "Her ... bir ... dır." mantığı bulunmaktadır. Yukarıdaki kod parçası için şu ifade kurulabilir: "Her öğretmen bir okul personelidir."

Her öğretmen, bir okul personeli olduğuna göre aşağıdaki kod satırı hata vermeyecektir.

```
Ogretmen ogr = new Ogretmen
{
    Ad = "Ahmet",
    Soyad = "Öz",
    Brans = "Matematik"
};
OkulPersoneli per = ogr; // !!!
Console.WriteLine(per.Ad);

// Ekran çıktısı:
Ahmet
```

Hafızanın öbek bölgesinde Ogretmen nesnesi bulunmasına rağmen per değişkeni üzerinden sadece Ad ve Soyad özelliklerine erişim mümkündür (Görsel 3.12).

Görsel 3.12: Referansın tipine bağlı nesneye erişim

Sıra Sizde

Çevrenizdeki nesneleri kalıtım açısından inceleyiniz ve en az üç tane kalıtım örneği gerçekleştiriniz.

3.10.1. Hiyerarşik Kalıtım

Bir sınıf, türetilmiş sınıflardan kalıtım yoluyla aynı şekilde türetilebilir. Bir anlamda hiyerarşik kalıtım mümkündür.

```
public class Canli
{
  //...
}
public class Hayvan : Canli
{
  //...
}
public class Kopek : Hayvan
{
  //...
}
public class Kangal : Kopek
{
  //...
}
```

Yukarıdaki sınıf tanımlamaları geçerlidir ve hiyerarşik kalıtıma bir örnektir. "Her ... bir ... dır." mantığı her bir sınıf için geçerlidir.

3.10.2. new Operatörüyle Metot Gölgeleme (Shadowing)

Bir sınıftan başka bir sınıf türetildiğinde özel bir durum ortaya çıkar. Üst sınıfta bulunan bir metot, alt sınıfta da tanımlanmışsa derleyici bir "uyarı" verir. Bu durumda üst sınıfta yer alan metot gölgelenerek erişilemez duruma gelir.

```
public class OkulPersoneli
{
   public string Ad { get; set; }
   public string Soyad { get; set; }

   public void AdSoyadYazdir()
   {
      Console.WriteLine("Benim adım soyadım : " + Ad + " " +
Soyad);
   }
}

public class Ogretmen : OkulPersoneli
{
   public string Brans { get; set; }

   public void AdSoyadYazdir()
   {
      Console.WriteLine("Benim adım soyadım : " + Ad + " " +
Soyad);
   }
}
```

Yukarıdaki kod parçasında **Ogretmen** sınıfındaki **AdSoyadYazdir()** metodu, üst sınıftaki aynı isimli metodu gölgelemektedir. Hem üst sınıfta hem de türetilmiş sınıfta aynı isimli metot bulunduğu için derleyici, alt sınıftaki metodun **new** operatörü ile tanımlanması gerektiğini belirten bir "uyarı" (hata değil) verir (Görsel 3.13).

Görsel 3.13: Derleyici uyarı mesajı

Bu bir hata olmadığı için program çalışacaktır. Ancak bu uyarı mesajını ortadan kaldırmak için bu metodun kasıtlı olarak yazıldığını derleyiciye bildirmek gerekmektedir. Bunun için **new** anahtar kelimesi kullanılır.

```
public class Ogretmen : OkulPersoneli
{
   public string Brans { get; set; }

   public new void AdSoyadYazdir()
   {
      Console.WriteLine("Benim adım soyadım : " + Ad + " " + Soyad);
   }
}
```

Bu durumda derleyici herhangi bir uyarı ya da hata mesajı vermeyecektir. Ancak bu kullanımın NTP prensiplerine uygun olduğu söylenemez. O nedenle bu gibi durumlarda "sanal metot"ların kullanılması uygun olacaktır.

3.10.3. Sanal Metotlar (Virtual Methods)

Temel sınıftan türetilmiş alt sınıflara aktarılan metotlar her zaman olduğu gibi kullanılmayabilir. İstenilen metotlar alt sınıflarda tekrardan yazılabilir. Böyle bir durumda bu metotlar, üst sınıfta **virtual** (sanal), alt sınıflarda da **override** (geçersiz kılma veya ezme) anahtar kelimeleri kullanılarak tanımlanmalıdır. Sanal olarak tanımlanan metotlar, alt sınıflarda geçersiz kılınmak zorunda değildir. Sanal metotlar geçersiz kılınmazsa metodun kendisi, sanal metotlar geçersiz kılınırsa alt sınıfın metodu çağrılır.

```
public class Sekil
{
   public const double pi = 3.14;
   protected double x, y;
   public Sekil()
   {
   }

   public Sekil(double x, double y)
   {
      this.x = x;
      this.y = y;
   }
public virtual double AlanHesapla()
   {
       return x * y;
   }

   public virtual void BilgiYazdir()
   {
      Console.WriteLine("x= " + x + " ve y= " + y);
   }
}

public class Daire : Sekil
{
   public Daire(double r) : base(r, 0)
   {
   }

   public override double AlanHesapla()
   {
      return pi * x * x;
   }
}

class Program
{
   static void Main(string[] args)
   {

      Sekil s = new Sekil(3, 4);
      s.BilgiYazdir();
      Console.WriteLine("Şekil alanı: " + s.AlanHesapla());

      Console.WriteLine("======================");

      Daire d = new Daire(1.3);
      d.BilgiYazdir();
      Console.WriteLine("Daire alanı: {0:N2}", d.AlanHesapla());
   }
}
```

```
// Ekran çıktısı:
x= 3 ve y = 4
Şekil alanı: 12
======================
x= 1,3 ve y= 0
Daire alanı: 5,31
```

Daire sınıfının alan hesabı, Sekil sınıfının alan hesabından farklıdır. Yukarıdaki programda da görüldüğü gibi AlanHesapla() metodu alt sınıfta geçersiz kılınmış ancak BilgiYazdir() metodu alt sınıfta geçersiz kılınmamıştır.

Not:
```
public Daire(double r) : base(r, 0)
        {
        }
```
Yukarıdaki kod blokunda Daire sınıfının yapıcı metodu tanımlanırken nesne oluşturulduğu anda aynı zamanda üst sınıfın yapıcı metodunun da çağrılması sağlanmıştır. Bunun için base anahtar kelimesi kullanılmaktadır. this anahtar kelimesi sınıfı, base anahtar kelimesi ise üst sınıfı temsil etmektedir.

Not: Sınıf içinden üst sınıfın ögelerine **base** anahtar kelimesi ile erişim mümkündür. Yukarıdaki örnek için **"base.x"** ifadesi ile Sekil sınıfının x alanına erişilebilir.

Sıra Sizde

Kitap, dergi ve ansiklopediler için bir üst sınıf yazarak Oku() sanal metotlarını ihtiva eden sınıflar tanımlayınız.

3.11. Soyut Sınıflar (Abstract Classes)

NTP'nin bir diğer önemli kavramı da soyutlamadır. Soyutlama genellikle ortak özellikleri olan sınıfları bir çatı altında toplamak için kullanılır. Soyut sınıfların klasik sınıflardan en önemli farkı, new anahtar kelimesi ile nesnelerinin oluşturulamamasıdır.

Soyut sınıflar **abstract** anahtar kelimesi ile tanımlanmalı ve en az bir tane **abstract** ile tanımlanmış metodu olmalıdır. Bu metodun sadece imzası bulunur, gövdesi bulunmaz. Ayrıca soyut olarak tanımlanmış metotlar, bu sınıftan türeyen alt sınıflarda mutlaka geçersiz kılınmalıdır (override).

Not: Sanal metotlar geçersiz kılınmak zorunda değildir. Buna karşın soyut metotlar mutlaka geçersiz kılınmalıdır.

```
public abstract class MotorluArac
{
   public int MotorHacmi { get; set; }
   public int ModelYili { get; set; }

   public abstract void Calis();
   public abstract void Dur();
}
public class Otomobil : MotorluArac
{
   public bool OtomatikVites { get; set; }

   public override void Calis()
   {
      Console.WriteLine("Otomobil çalıştı.");
   }
   public override void Dur()
   {
      Console.WriteLine("Otomobil durdu.");
   }
}
class Program
{
   static void Main(string[] args)
{
      // ** Alttaki satır hata verir.
      // ** Sanal sınıflardan nesne türetilemez.
      // MotorluArac ma = new MotorluArac();

   Otomobil oto = new Otomobil
      {
         ModelYili = 2020,
         MotorHacmi = 1600,
         OtomatikVites = true
      };
      oto.Calis();
      oto.Dur();
   }
}
// Ekran çıktısı:
Otomobil çalıştı.
Otomobil durdu.
```

Her motorlu araçta bulunan ortak özellikler MotorluAraclar sınıfında, otomobillere özgü özellikler de Otomobil sınıfında tanımlanmıştır.

MotorluAraclar sınıfındaki iki soyut metot, Otomobil sınıfında geçersiz kılınmıştır (Geçersiz kılınmak zorundadır.).

Not: Sanal metotların sadece imzaları vardır, gövdeleri yoktur.

```
public abstract void Calis();
public abstract void Dur();
```

"Her ... bir ... dır." mantığı burada da geçerlidir. "Her otomobil bir motorlu araçtır." ifadesi doğru olduğuna göre aşağıdaki kod hata vermeyecektir.

```
Otomobil oto = new Otomobil
{
    ModelYili = 2020,
    MotorHacmi = 1600,
    OtomatikVites = true
};

MotorluArac ma = oto;
ma.Calis();
ma.Dur();
```

Sıra Sizde

Kitap, dergi ve ansiklopediler için bir soyut üst sınıf yazarak Oku() metotlarını ihtiva etmek zorunda olan sınıfları tanımlayınız.

3.12. Arayüzler (Interfaces)

NTP'de soyutlamanın bir başka yolu da **arayüzler** (interfaces) aracılığıyla mümkündür. Bir arayüz, tüm ögeleri soyut olan bir sınıfa benzetilebilir ancak burada arayüzlerin ve sınıfların farklı kavramlar olması önemli bir husustur. Bir sınıf sadece bir sınıftan türetilebiliyor iken birden fazla arayüzden türetilebilir.

Arayüzün içinde tanımlanan metotların sadece imzaları bulunur, gövdeleri bulunmaz. Ayrıca arayüzde bulunan tüm metotlar varsayılan olarak soyuttur (abstract) ve genel (public) erişim belirleyicisine sahiptir.

Arayüz bir sınıf türü olmadığından içinde kod bloku bulunamaz. Arayüzde tanımlanan ögeler, kendisinden türetilen sınıfta mutlaka uygulanmak (implement) zorundadır.

Not: Soyut sınıflarda "geçersiz kılma" (override) kavramı, arayüzler için de "uygulamak" (implement) kavramı kullanılır.

Not: Arayüz isimlerinin sınıf olmadığını belirtmek için "I" (büyük ı) harfi ile başlatılması gelenektir.

Aşağıda türetildiği arayüzleri uygulayan bir sınıf tanımı görülmektedir.

```
interface IHayvan
{
   void SesCikar();
}

interface IBeslen
{
   void Beslen();
}

public class Kedi : IHayvan, IBeslen
{
   public void SesCikar()
   {
      Console.WriteLine("Kedi: miyav");
   }

   public void Beslen()
   {
      Console.WriteLine("Kedi süt içti.");
   }
}

public class Kopek : IHayvan, IBeslen
{
   public void SesCikar()
   {
      Console.WriteLine("Köpek: hav hav");
   }

   public void Beslen()
   {
      Console.WriteLine("Köpek et yedi.");
   }
}
class Program
{
   static void Main(string[] args)
   {
      Kedi kedi = new Kedi();
      kedi.SesCikar();
      kedi.Beslen();

      Kopek kopek = new Kopek();
      kopek.SesCikar();
      kopek.Beslen();

      Console.WriteLine("==================");
      IHayvan hayvan1 = kedi;
      IHayvan hayvan2 = kopek;
      hayvan1.SesCikar();
      hayvan2.SesCikar();
      Console.WriteLine("==================");
      IBeslen beslen1 = kedi;
      IBeslen beslen2 = kopek;
      beslen1.Beslen();
      beslen2.Beslen();
    }
}
```

```
// Ekran çıktısı:
Kedi: miyav
Kedi süt içti.
Köpek: hav hav
Köpek et yedi.
==================
Kedi: miyav
Köpek: hav hav
==================
Kedi süt içti.
Köpek et yedi.
```

Hem Kedi hem de Kopek sınıfları, IHayvan ve IBeslen arayüzlerinden türetilmiştir. Dolayısıyla her iki sınıf da arayüzlerin içindeki metotları uygulamak zorundadır. Arayüzden türetilen sınıflar için soyut sınıflarda olduğu gibi "Her ... bir ... dır." ifadesi yine kullanılmaktadır ("Her kedi bir hayvandır.", "Her köpek bir hayvandır."). Tablo 3.2'de arayüzler ve soyut sınıflar arasındaki farklar listelenmiştir.

Tablo 3.2: Arayüzler ve Soyut Sınıflar Arasındaki Farklar

Arayüzler	Soyut Sınıflar
Bir sınıf birden fazla arayüzden türetilebilir.	Bir sınıf sadece tek bir soyut sınıftan türetilebilir.
Sadece boş (gövdesi olmayan) metotlar tanımlanabilir.	Hem normal metot hem de boş metotlar tanımlanabilir.
Çoklu kalıtım özelliği sağlar.	Çoklu kalıtım özelliği sağlamaz.
Tüm ögeler public olarak kabul edilir.	Ögeler public olmak zorunda değildir.
Yapıcı metot içeremez.	Yapıcı metot içerebilir.
Statik ögeler barındıramaz.	Statik ögeler barındırabilir.

Sıra Sizde

Kitap, dergi ve ansiklopediler için bir arayüz aracılığıyla Oku() metotlarını ihtiva etmek zorunda olan sınıfları tanımlayınız.

3.13. Çok Biçimlilik (Polymorphism)

Çok biçimlilik NTP'deki bir diğer önemli prensiptir. Çok biçimlilik, aynı temel sınıftan veya arayüzden türetilmiş alt sınıflardaki metotların farklı şekillerde davranabilmesidir.Türetilen alt sınıflarda aşağıdaki durumlardan birinin sağlanması gerekir:

- Üst sınıftaki sanal ögeler geçersiz kılınır (virtual / override).
- Soyut sınıflarda soyut tanımlanan ögeler geçersiz kılınır (abstract / override).
- Arayüzlerdeki ögeler uygulanır (implementation).

Her üç yolla da çok biçimlilik sağlanabilir.

Aşağıdaki sınıflar için üç yolla da çok biçimliliğin sağlandığı görülmektedir.

virtual / override

```
class Sekil
{
   public virtual void Ciz()
   {
      Console.WriteLine("Şekil çizildi.");
   }
}
class Kare : Sekil
{
   public override void Ciz()
   {
      Console.WriteLine("Kare çizildi.");
   }
}
class Daire : Sekil
{
   public override void Ciz()
   {
     Console.WriteLine("Daire çizildi.");
   }
}
class Ucgen : Sekil
{
   public override void Ciz()
   {
      Console.WriteLine("Üçgen çizildi.");
   }
}
```

abstract / override

```
abstract class Sekil
{
   public abstract void Ciz();
}

class Kare : Sekil
{
   public override void Ciz()
   {
      Console.WriteLine("Kare çizildi.");
   }
}

class Daire : Sekil
{
   public override void Ciz()
   {
      Console.WriteLine("Daire çizildi.");
   }
}

class Ucgen : Sekil
{
   public override void Ciz()
   {
      Console.WriteLine("Üçgen çizildi.");
   }
}
```

interface

```
interface Sekil
{
   void Ciz();
}

class Kare : Sekil
{
   public void Ciz()
   {
      Console.WriteLine("Kare çizildi.");
   }
}
class Daire : Sekil
{
   public void Ciz()
   {
      Console.WriteLine("Daire çizildi.");
   }
}
class Ucgen : Sekil
{
   public void Ciz()
   {
      Console.WriteLine("Üçgen çizildi.");
   }
 }
```

Aşağıdaki programda çok biçimlilik uygulanmıştır.

```
class Program
{
   static void Main(string[] args)
   {
      List<Sekil> sekiller = new List<Sekil>
      {
         new Daire(),
         new Kare(),
         new Ucgen()
      };

      foreach (var sekil in sekiller)
      {
         sekil.Ciz();
      }
   }
}

// Ekran çıktısı:
Daire çizildi.
Kare çizildi.
Üçgen çizildi.
```

Burada önemli olan husus, "sekil.Ciz();" ifadesi ile oluşturulan farklı nesnelerin aynı isimli metodunun çağrılarak ilgili nesneye ait işlemlerin gerçekleştirilmesidir.

Sıra Sizde

Kitap, dergi ve ansiklopedileri tutan bir koleksiyon nesnesi oluşturunuz ve bu sınıfların içindeki Oku metodunu döngü kullanarak çağırınız.

3.14. Statik Sınıflar (Static Classes)

Statik (static) sınıflar temel olarak statik olmayan sınıflarla aynıdır ancak statik sınıflardan new anahtar kelimesi ile nesne türetilemez. Nesne türetilemediği için bu sınıfların ögelerine nesne adı üzerinden değil, doğrudan sınıf adı üzerinden erişilir. Sınıfın kendisi ya da içindeki bazı ögeler statik olarak tanımlanabilir. Sınıfın kendisi statik olarak tanımlanırsa sınıf içindeki tüm ögelerin statik olması zorunludur.

Statik ögeler genellikle nesnelerin durumuna göre değişmeyen verileri temsil etmede veya hesaplamaları yapmada kullanılır. Buna en güzel örnek, .NET Sınıf Kütüphanesi'ndeki Math sınıfıdır.

```
Math m1 = new Math(); // Hata
Math m2 = new Math(); // Hata
Console.WriteLine(m1.Sqrt(9));
Console.WriteLine(m2.Sqrt(9));
```

Bir sayının karekökü yukarıdaki kod yazılarak hesaplanmak istendiğinde **m1** ve **m2** nesneleri üzerinden hesaplamada farklılık olmayacağı için **Math** sınıfından bir nesne türetilmesine gerek yoktur. Dolayısıyla **Math** sınıfı **static** olarak tanımlanmıştır ve bu sınıftan nesne türetilemez. Bu yüzden yukarıdaki kod parçası hata verecektir. Karekök hesaplamak için kullanılan **Sqrt()** metoduna nesne oluşturmadan sınıf adı üzerinden erişilmektedir.

```
double sayi = Math.Sqrt(9);
Console.WriteLine(sayi);
```

Statik bir sınıf, herhangi bir ögesi ilk kullanıldığı anda hafızaya yüklenir ve programın çalışması sonlanana kadar hafızada durur. Bu yüzden statik sınıfın dikkatli kullanılması önerilir.

Bir sınıf **static** olarak tanımlanmadan içindeki herhangi bir öge **static** olarak tanımlanabilir. Ayrıca statik bir yapıcı oluşturmak da mümkündür. Aşağıdaki örnekte hem normal bir sınıf yapıcısı hem de statik yapıcı bir arada kullanılmıştır.

```
public class Ogrenci
{
    public int Numara { get; set; }
    public string AdSoyad { get; set; }
}

public class OgrenciIslem
{
    public List<Ogrenci> ogrenciler;
    public static int OgrenciSayisi { get; set; }

    static OgrenciIslem()
    {
        OgrenciSayisi = 0;
        Console.WriteLine("Statik yapıcı çalıştı.");
    }

    public OgrenciIslem()
    {
        ogrenciler = new List<Ogrenci>();
        Console.WriteLine("Yapıcı çalıştı.");
}

    public void OgrenciEkle(Ogrenci ogr)
    {
        ogrenciler.Add(ogr);
        OgrenciSayisi++;
        Console.WriteLine("Öğrenci eklendi.");
}
```

```
    public void OgrenciSil(int numara)
    {
        var ogr = ogrenciler.FirstOrDefault(x => x.Numara == numara);
        if (ogr != null)
        {
            ogrenciler.Remove(ogr);
            OgrenciSayisi--;
            Console.WriteLine("Öğrenci silindi.");
        }
    }
}

internal class Program
{
    private static void Main(string[] args)
    {
        OgrenciIslem oi = new OgrenciIslem();

        Console.WriteLine("1) ==========================");
        oi.OgrenciEkle(new Ogrenci
        {
            Numara = 100,
            AdSoyad = "Nihal Öz"
        });
        Console.WriteLine("Öğrenci sayısı: " + OgrenciIslem.OgrenciSayisi);

        Console.WriteLine("2) ==========================");

        oi.OgrenciEkle(new Ogrenci
        {
            Numara = 200,
            AdSoyad = "İbrahim Yurt"
        });
        Console.WriteLine("Öğrenci sayısı: " + OgrenciIslem.OgrenciSayisi);

        Console.WriteLine("3) ==========================");
        oi.OgrenciSil(100);
        Console.WriteLine("Öğrenci sayısı: " + OgrenciIslem.OgrenciSayisi);
    }
}

// Ekran çıktısı:
Statik yapıcı çalıştı.
Yapıcı çalıştı.
1) ==========================
Öğrenci eklendi.
Öğrenci sayısı: 1
2) ==========================
Öğrenci eklendi.
Öğrenci sayısı: 2
3) ==========================
Öğrenci silindi.
Öğrenci sayısı: 1
```

Sıra Sizde

Bir kütüphane sınıfını aşağıdaki istekleri karşılayacak şekilde oluşturunuz.

1. Eklenen ögelerin adedini tutan gizli bir özelliği olsun.
2. Kitap, dergi veya ansiklopedi ekleme metodu olsun.
3. Eklenen öge adedini ekrana yazdıran bir metodu olsun.

3.15. İsimsiz Sınıflar (Anonymous Classes)

İsimsiz sınıflar yalnızca salt okunur özellikleri içeren ve adı olmayan sınıflardır. Alan, metot gibi diğer ögeleri barındıramaz. Sınıf özelliklerinin veri tipleri, aldığı değere göre otomatik olarak belirlenir.

Bir isimsiz sınıf, **var** anahtar kelimesi ile tanımlanır ve **new** anahtar kelimesi ile oluşturulur.

```
var ogrenci = new
{
   Numara = 35,
   Ad = "Yasin",
   Ortalama = 80.5
};

Console.WriteLine("Öğrencinin adı: " + ogrenci.Ad);

//ogrenci.Ortalama = 90.2; HATA!!!
```

Ayrıca bir isimsiz sınıf içinde bir başka isimsiz sınıf oluşturulabilir.

```
var ogrenci = new
{
   Numara = 35,
   Ad = "Yasin",
   Ortalama = 80.5,
   Adres = new
   {
      Il = "Malatya",
      Ilce = "Yeşilyurt"
   }
};
Console.WriteLine("Öğrencinin yaşadığı il: " + ogrenci.Adres.Il);

//ogrenci.Ortalama = 90.2; HATA!!!
```

İstenirse isimsiz sınıf dizisi de oluşturulabilir.

```
var ogrenciler = new[]
    {
    new { Numara = 100, Ad = "Yasin", Ortalama = 80},
    new { Numara = 200, Ad = "İsmail", Ortalama = 75},
    new { Numara = 300, Ad = "Ömer", Ortalama = 60}
  };
Console.WriteLine("2. öğrencinin adı: " + ogrenciler[1].Ad);
Console.WriteLine("====================");

foreach (var ogrenci in ogrenciler)
{
   Console.WriteLine("Adı: {0}, Ortalaması: {1}", ogrenci.Ad,
ogrenci.Ortalama);
}
```

```
// Ekran çıktısı:
2.öğrencinin adı: İsmail
====================
Adı: Yasin, Ortalaması: 80
Adı: İsmail, Ortalaması: 75
Adı: Ömer, Ortalaması: 60
```

Sıra Sizde

Bir isimsiz sınıfı parametre olarak alan metodu nasıl tanımlayabileceğinizi araştırınız ve edindiğiniz bilgileri sınıf arkadaşlarınızla paylaşınız.

3.16. Mühürlü Sınıflar (Sealed Classes)

Bir sınıftan bir başka sınıf türetilmek istenmediğinde bu sınıfı **sealed** anahtar kelimesiyle mühürlü tanımlamak gerekir.

```
public sealed class UstSinif
{
   // ..
}

public class AltSinif : UstSinif //
HATA !!!
{
   // ...
}
```

Yukarıdaki kod parçası hata verecektir (Görsel 3.14).

Görsel 3.14: Derleyici hata mesajı

3.17. Parçalı Sınıflar (Partial Classes)

Büyük projelerde oluşturulan sınıfları birden fazla dosyaya yaymak mümkündür. Parçalı sınıflar;

Büyük sınıfları parçalamak, okunmasını kolaylaştırmak,

Mantıksal olarak katmanlara ayırmak (veri tabanı işlemlerinin ayrı bir dosyada olması gibi),

Sınıf ögelerini ayrıştırmak (özellikler bir dosyada, metotlar başka bir dosyada vb.),

Aynı sınıf üzerinde birden fazla programcının çalışması gibi durumlar için kullanılabilir.

Parçalı sınıflar oluşturmak için **partial** anahtar kelimesi kullanılır. Parçalı sınıfların isimleri aynı olmalıdır.

```
// ParcaliSinif1.cs
public partial class ParcaliSinif
{
    public int Ozellik1 { get; set; }
    // ...
}

// ParcaliSinif2.cs
public partial class ParcaliSinif
{
    public void Metot1() { }
    //...
}

// Program.cs
internal class Program
{
    private static void Main(string[] args)
    {
        ParcaliSinif ps = new ParcaliSinif();
        Console.WriteLine(ps.Ozellik1);
        ps.Metot1();
    }
}
```

Program derlendiğinde tüm parçalar birleştirilir ve tek bir sınıf tanımlanmış gibi çalıştırılır.

3.18. Enums (Numaralandırmalar)

Bir **enum** (enumerations kelimesinin kısaltması) sadece **int** tipindeki sabitlerden oluşan özel bir sınıftır. Bu değerler sadece okunabilirdir ve değiştirilemez. Enum genellikle programların okunmasını kolaylaştırmak için kullanılır.

Bir numaralandırma oluşturabilmek için enum anahtar kelimesi ve değerleri birbirinden ayırmak için , (virgül) karakteri kullanılır.

```
enum Seviyeler
{
    Çok_Düşük, // 0
    Düşük,     // 1
    Orta,      // 2
    Yüksek,    // 3
    Çok_Yüksek // 4
}
internal class Program
{
    private static void Main(string[] args)
    {
        Console.WriteLine(Seviyeler.Düşük);
        Console.WriteLine((int)Seviyeler.Düşük);
    }
}
// Ekran çıktısı:
Düşük
1
```

enum içindeki değerler 0'dan (sıfır) başlayarak birer birer artar. İstenirse farklı tam sayı değerleri de verilebilir.

```
enum Kategoriler
{
   Bilgisayar = 3,
   Mobilya    = 10,
   Kırtasiye  = 7,
   Hırdavat,   // 8
   Otomobil    // 9
}
```

Yukarıdaki örnekte bilgisayar, mobilya ve kırtasiye kategorilerine istenilen değerler atanmıştır. Değer ataması yapılmayan hırdavat ve otomobil kategorilerine ise 7'den sonra gelen 8 ve 9 değerleri otomatik olarak atanmıştır.

Program içinde örnek kullanım aşağıda verilmiştir.

```
// enum tanımlama
Kategoriler kat = Kategoriler.Kırtasiye;
// değer adının kullanımı
Console.WriteLine(kat);
// değerin sayısal değerinin kullanımı
Console.WriteLine((int)kat);
// enuma sayısal değer atama
kat = (Kategoriler)8;
// if ile kullanımı
if (kat == Kategoriler.Kırtasiye)
   Console.WriteLine("Hırdavat kategorisi");
// switch ile kullanımı
switch (kat)
{
   case Kategoriler.Bilgisayar: // ..
      break;
   case Kategoriler.Mobilya:    // ..
      break;
   case Kategoriler.Kırtasiye:  // ..
      break;
   case Kategoriler.Hırdavat:   // ..
      break;
   case Kategoriler.Otomobil:   // ..
      break;
   default:
      break;
}
```

Sıra Sizde

Ayları 1'den başlayarak numaralandıran tanımlamayı yazınız.

Sıra Sizde

Numaralandırmayı nerelerde kullanabileceğinizi araştırınız ve edindiğiniz bilgileri sınıf arkadaşlarınızla paylaşınız.

ÖLÇME VE DEĞERLENDİRME - 3

1. **"Ses seviyesi", "ekran boyutu" ve "görüntü teknolojisi" alanlarına sahip bir "Televizyon" sınıfı yazınız.**
2. **"RAM bellek kapasitesi", "CPU", "HD kapasitesi" alanlarına sahip bir "Bilgisayar" sınıfı yazınız.**
3. **"Televizyon" sınıfını özellikler kullanarak tekrar yazınız.**
4. **"Bilgisayar" sınıfını özellikler kullanarak tekrar yazınız.**
5. **"Televizyon" sınıfına "Güç aç / kapat", "Kanal değiştir" ,"Ses seviyesi oku" metotlarını ekleyiniz (Gerekli alanları sınıfa ekleyiniz.).**
6. **"Televizyon" sınıfının "Kanal değiştir" metodunu aşağıdaki şekilde güncelleyiniz (Gerekli alanları sınıfa ekleyiniz.). Eklediğiniz metotları program içinde kullanınız.**
   - KanalNoArtir() => Kanal numarasını bir artırmalı.
   - KanalNoArtir(int) => Kanal numarasını parametrede verilen değer kadar artırmalı.
   - KanalNoAzalt() => Kanal numarasını bir azaltmalı.
   - KanalNoAzalt(int) => Kanal numarasını parametrede verilen değer kadar azaltmalı.
7. **"Televizyon" sınıfında kullandığınız alanların, özelliklerin ve metotların erişim türlerini açıklayınız.**
8. **"Televizyon" sınıfından "İşletim sistemi" özelliğine sahip bir "Akıllı televizyon" sınıfını aşağıdaki hususlara dikkat ederek türetiniz**.
   - "Televizyon" sınıfındaki "Güç aç / kapat" metodu bu sınıf içinde tekrar yazılmalıdır.
   - "Televizyon" sınıfındaki "ses seviyesi" bilgisi sadece bu iki sınıf içinde kullanılabilir olmalıdır.
9. **"Televizyon" sınıfından türetilen tüm nesneler için "Marka" bilgisinin aynı olması istenirse bu sınıf üzerinde nasıl bir değişiklik yaparsınız?**
10. **"Televizyon" sınıfından başka bir sınıf türetilemesin istenirse bu sınıf üzerinde nasıl bir değişiklik yaparsınız?**
11. **Sınıf içindeki bir değişkeni dış dünyaya kapatıp sadece sınıf içinde kullanılabilir kılmak için bu özellik …… şeklinde tanımlanmalıdır.**
12. **Sınıf içindeki bir değişkeni dış dünyaya kapatıp sadece sınıf içinde ve bu sınıftan türetilen alt sınıflarda kullanılabilir kılmak için bu özellik…… şeklinde tanımlanmalıdır.**
13. **Sınıf içindeki bir değişkeni her yerden erişilebilir kılmak için bu özellik …… şeklinde tanımlanmalıdır.**

14. Aşağıdaki kodun çıktısı nedir?

```
using System;
class program
{
   static void Main(string[] args)
   {
      int num = 2;
      Fonk1(ref num);
      Console.WriteLine(num);
   }
   static void Fonk1(ref int num)
   {
      num = num * num * num;
   }
}
```

15. Sınıf içinde nesne oluşturulurken ve nesne hafızadan atılırken otomatik olarak çalıştırılan metotlar nasıl adlandırılır? Özellikleri nelerdir?

16. Aşağıdaki kodun çıktısı nedir?

```
static void Main(string[] args)
{
   int sayi = 5;
   int kare = 0, kup = 0;
   Hesapla(sayi, kare, ref kup);
   Console.WriteLine(kare + " & " + kup);
   Console.ReadLine();
}
static void Hesapla(int sayi, int kare, ref int kup)
{
   kare = sayi * sayi ;
   kup = kare * sayi;
}
```

17. Statik sınıflar hangi durumlarda kullanılır?

18. Statik yapıcı metotlar ne zaman çalıştırılır?

19. Soyut sınıf ile arayüzler arasındaki benzerlikler ve farklılıklar nelerdir?

20. "Güç aç" ve "Güç kapat" metotlarını tanımlayan bir arayüz yazarak "Televizyon" ve "Bilgisayar" sınıflarına bu metotları uygulayınız.

DİZİLER (ARRAYS) VE KOLEKSİYONLAR (COLLECTIONS)

NELER ÖĞRENECEKSİNİZ?

Bu öğrenme biriminde;

- Dizi kavramını açıklayacak,
- Dizi tanımlaması yapacak,
- Dizilere değer verme ve dizilerden değer alma işlemlerini yapacak,
- Çok boyutlu dizi kavramını açıklayacak,
- Çok boyutlu dizi tanımlaması yapacak,
- Çok boyutlu dizilere değer verme ve dizilerden değer alma işlemlerini yapacak,
- İhtiyaca uygun olarak dizileri kullanacak,
- Koleksiyon kavramını açıklayacak,
- Koleksiyon tanımlaması yapacak,
- Koleksiyonlara değer verme ve koleksiyonlardan değer alma işlemlerini yapacak,
- İhtiyaca uygun koleksiyon kullanımını öğreneceksiniz.

ANAHTAR KELİMELER

Dizi, index, veri tipi, for döngüsü, foreach döngüsü, koleksiyon

HAZIRLIK ÇALIŞMALARI

1. Projenizde girilen 10 adet sayının ortalamasını değişkenler kullanarak nasıl alırsınız? Sınıf arkadaşlarınızla paylaşınız.
2. Size verilen 100 kişinin isimlerini düzenlemede yapacağınız ilk işlem nedir? Nedeniyle birlikte açıklayınız?

4.1. Diziler

Dizi, aynı tipte birden çok değeri bellek üzerinde tutabilecek yapıdır. Programlama yaparken dizileri kullanmak; dizilerin verdiği avantajlardan yararlanarak değerler üzerinde seçme, silme, değiştirme, sıralama vb. işlemlerin kolayca gerçekleştirilmesini sağlar.

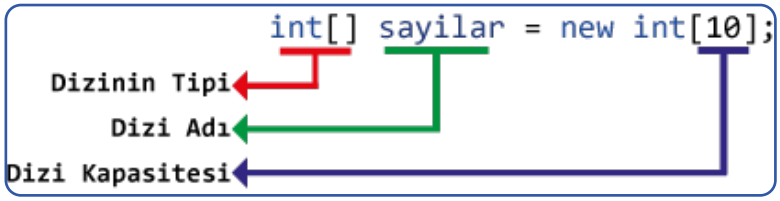
4.1.1. Tek Boyutlu Diziler

Bir boyutlu veya tek boyutlu diziler, verileri saklamak için bir satırdan oluşan dizilerdir. Bu diziler, art arda sıralanmış bellek alanları gibi düşünülebilir. Aynı tipten değerler olmak şartıyla belirlenen adet kadar veri, dizi içinde sıralanmış bellek alanlarında saklanır. Burada dikkat edilmesi gereken noktaların başında diziyi oluştururken içinde kaç adet veri olacağı ve dizilerde hangi tip verilerin saklanacağı (int, string, char, double vb.) gelir. Belirtilen tipin sınırı dışında veya belirtilen adet sayısından daha fazla veri saklamaya çalışıldığında derleyici hata verir.

4.1.2. Bir Boyutlu Dizilerin Oluşturulması

Dizi oluştururken temelde üç noktaya dikkat edilir.

1. **Dizinin Tipi:** Dizide hangi tip verilerin saklanacağı (int, string, char, byte, double vb.)
2. **Dizi Adı:** Dizide saklanacak verilerle anlamlandırılan değişken isimlendirme kurallarına uygun hangi isimlerin diziye verileceği (Anlamlı isimler vermek, yazılan kodların okunabilirliğini artıracağı için her zaman tavsiye edilen bir yöntemdir. Örneğin, okul numaraları saklanacak bir dizi için diziOkulNo kullanılabilir.)
3. **Dizilerin Kapasitesi:** Dizi içinde kaç adet veri saklanacağı Görsel 4.1'de sayilar isminde, integer tipinde 10 adet veri saklama kapasitesine sahip bir dizi tanımlaması yapılmıştır. Derleyicinin bir diziyi tanıması için başlangıçta veri tipi belirtildikten sonra içi boş köşeli parantezler kullanılmalıdır. İçi boş köşeli parantezler, bu ifadenin bir boyutlu dizi olduğu anlamına gelir. Tanımlamadaki ikinci köşeli parantez ise dizide saklanacak değer sayısını belirtmektedir. Aşağıda farklı veri tiplerine sahip dizi tanımlama örnekleri verilmiştir.



Görsel 4.1: Bir boyutlu dizi tanımlaması

```
string[] isimler = new string[5];  // String tipinde 5 elemanlı dizidir.

byte[] siralar = new byte[6];      // Byte tipinde 6 elemanlı dizidir.

bool[] durumlar = new bool[4];     // Boolean tipinde 4 elemanlı dizidir.

float[] uzunluklar = new float[8]; // Float tipinde 8 elemanlı dizidir.
```

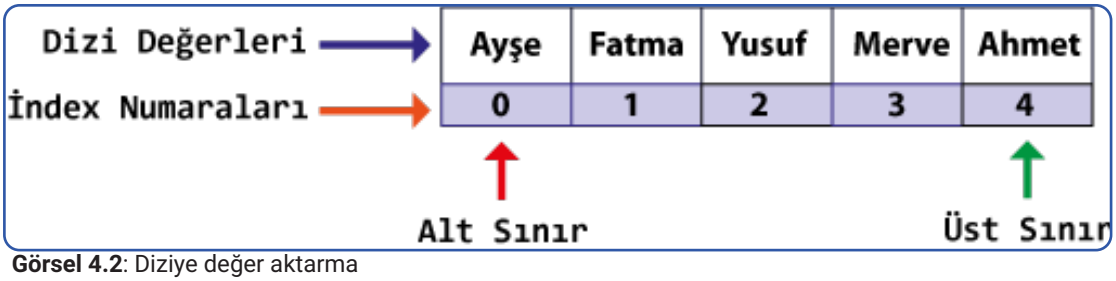
Bir dizi tanımlaması yapıldığında derleyici, dizinin her elemanına temel veri tipleri için varsayılan değerleri ilk değer olarak verir. İlk değerler, dizi içine veri eklenmeden verilir. Bunlar; string tipi için null, sayısal tipler için 0, bool tipi için ise false değerleridir. Verilen bu ilk değerler, dizilere değer aktarımı yapıldıkça yeni değerlerle değiştirilir.

4.1.3. Bir Boyutlu Dizilere Değer Aktarma

Dizilere değer aktarmanın farklı yöntemleri vardır. Dizilere ilk olarak tanımlamasının yapıldığı satırda değer verilebilir.

```
string[] kisiler = new string[5] {"Ayşe","Fatma","Yusuf","Merve","Ahmet"};
```

Yukarıda tanımlanan ve aynı satırda değer aktarımı yapılan dizide kodlar derlendiğinde bellekte **5** elemanlı bir dizi oluşturulur. Oluşturulan bu diziye küme parantezi **{ }** içindeki değerler sırasıyla verilir.



Görsel 4.2: Diziye değer aktarma

Günlük hayatta sıralama işlemlerine hep 1'den başlanır fakat programlama dillerinin çoğunda sıralama 0'dan başlar. Görsel 4.2'de eklenen ilk elemanın sıra numarası 0'dır. Dizilerin her değerinin bir sıra numarası vardır. Sıra numaraları index, indis veya indeks olarak adlandırılır.

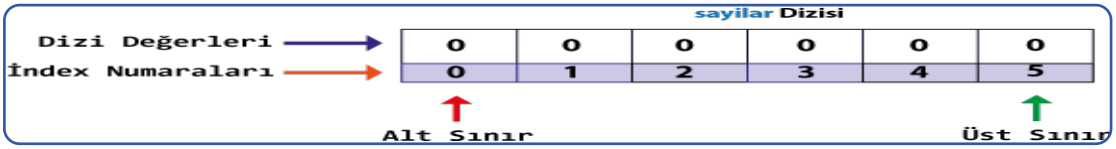
Tanımlandıkları satırda dizilere değer aktarma işlemi farklı şekillerde de yapılabilir. Aşağıdaki örnekte diziye değer aktarım işlemi, dizinin eleman sayısı belirtilmeden veya new sözcüğü kullanılmadan gerçekleştirilmiştir. Bu durumda derleyici hata vermez ve dizinin eleman sayısı derleyici tarafından belirlenir.

```
string[] kisiler = new string[] {"Ayşe","Fatma","Yusuf","Merve","Mehmet"};
string[] kisiler = {"Ayşe","Fatma","Yusuf","Merve","Mehmet"};
```

Dizilere değer aktarımının bir diğer yöntemi, dizinin index numaralarının kullanılarak yapılmasıdır.

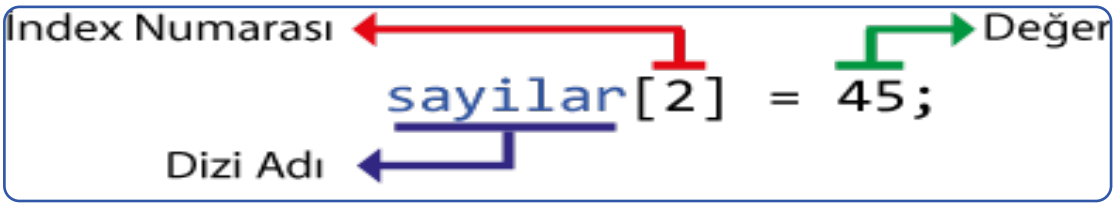
```
int[] sayilar = new int[6];
```

Yukarıdaki kodda **integer** veri tipine sahip, **6** elemanlı, **sayilar** adında bir dizi tanımlandı. Derleyici, bu kod satırına geldiğinde bellekte eleman sayısı kadar yer ayırır ve bu yerlere ilk değer olarak **0** (sıfır) sayısını aktarır.



Görsel 4.3: Dizinin ilk değerlerinin verilmesi

Bellek üzerinde dizi oluşturulduktan sonra index numaraları kullanılarak değer aktarımı gerçekleştirilebilir. Görsel 4.4'te **sayilar** ismindeki dizinin **2** numaralı index elemanına (dizinin üçüncü elemanına) 45 değerinin aktarımı yapılmıştır.



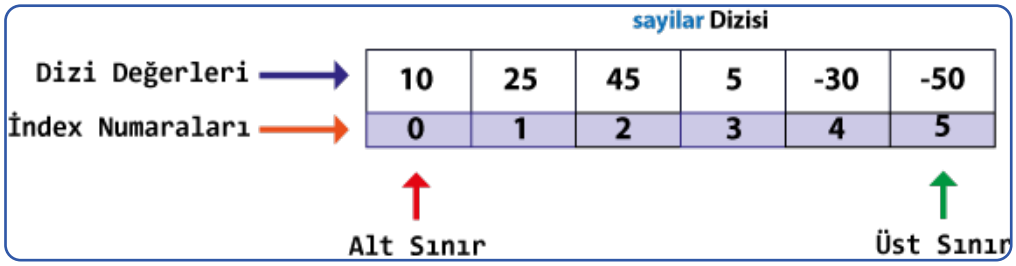
Görsel 4.4: Diziye index numarası ile değer aktarılması

Sıra Sizde

Aşağıdaki kodlamaları yaparak dizi elemanlarına değer aktarma işlemini gerçekleştiriniz.

```
int[] sayilar = new int[6];

sayilar[0] = 10;  //sayilar dizisinin 0 index numaralı elemanı 10 oldu.
sayilar[1] = 25;  //sayilar dizisinin 1 index numaralı elemanı 25 oldu.
sayilar[2] = 45;  //sayilar dizisinin 2 index numaralı elemanı 45 oldu.
sayilar[3] = 5;   //sayilar dizisinin 3 index numaralı elemanı 5 oldu.
sayilar[4] = -30; //sayilar dizisinin 4 index numaralı elemanı -30 oldu.
sayilar[5] = -50; //sayilar dizisinin 5 index numaralı elemanı -50 oldu.
```

Görsel 4.5'te değer aktarma işlemi bittikten sonra derleme işleminde bellek üzerindeki dizinin son hâli verilmiştir.



Görsel 4.5: Değer atandıktan sonra dizinin bellek üzerindeki durumu

Değer aktarım işleminde **index** numaralarına göre dizinin index numarası **0**'dan başlayarak dizideki eleman sayısının bir eksiğine kadar istenilen alana değer aktarılabilir. Tanımlanan dizide **0**'dan küçük ve eleman sayısının bir eksiğinden büyük bir index numarası ile diziye değer aktarmaya veya dizi elemanına erişmeye çalışıldığında derleyici tarafından hata mesajı gönderilir. Hata mesajı, girilen index numarasının dizinin sınırları dışında olduğunu bildirmektedir.

```
sayilar[6] = -30;
```

Yukarıdaki kod yazıldığında derleyici Görsel 4.6'daki hata mesajını vererek kullanıcıyı uyarır.

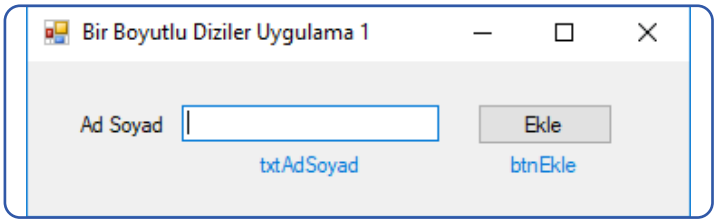
Görsel 4.6: Dizi sınır aşımı hata mesajı

Uygulama-1

Bir Boyutlu Diziler

Dizilere değer aktarım işlemi, kodlama sırasında değil de uygulamanın çalışması esnasında olabilir. Aşağıdaki uygulamada çalışma esnasında dizilere değer aktarım işlemi yapılmıştır.

Adım 1: Görsel 4.7'deki form tasarımını yaptıktan sonra form içindeki kontrollere name değerlerini (mavi yazı ile belirtilen) veriniz.



Görsel 4.7: Dizi uygulamaları form tasarımı-1

Adım 2: Ekle butonu Click olayında butona her tıkladığınızda TextBox içine girilen değerleri **isimler** adındaki **5** elemanlı bir diziye aktaracak kodlamaları yapınız.

```
string[] isimler = new string[5]; //Global Dizi
int index = 0; //Global Değişken
private void btnEkle_Click(object sender, EventArgs e)
{
   isimler[index] = txtAdSoyad.Text;
   index++;
}
```

Sıra Sizde

1. Kodlarda isimler dizisi neden btnEkle_Click içinde değil de global olarak tanımlanmıştır? Arkadaşlarınızla paylaşınız.
2. Adım 2'de verilen kodda tanımlanan index ismindeki değişken neden kullanılmıştır? Açıklayınız.
3. Uygulamada 6. kişi eklenmeye çalışıldığında nasıl bir hatayla karşılaşılır? Karşılaşılan bu hatanın çözümü için neler yapılabileceğini sınıf arkadaşlarınızla paylaşınız.

4.1.4. Bir Boyutlu Dizi Elemanlarına Erişim

Dizinin elemanlarına erişim, dizilere değer aktarmada olduğu gibi index numaraları kullanılarak sağlanır. Erişim sağlanan dizi elemanı; değişkenlere aktarma, hesaplamalar yapma, ekrana yazdırma, nesneye aktarma gibi işlemlerde kullanılır.

```
int[] dizi = new int[5] { 46, 41, 34, -10, 55 };
// 5 elemanlı bir dizi tanımlaması yapıldı. Aynı satırda değerler verildi.

int sayi1 = dizi[0];
// Dizinin 0 index numaralı değeri sayi1 ismindeki int tipindeki değişkene aktarıldı.

int toplam = dizi[0] + dizi[1] + dizi[2];
// Dizinin ilk üç elemanıyla toplama işlemi yapıldı.

Console.WriteLine(dizi[1]);
// Dizinin 1 index numaralı değeri (41 değeri) Console ekranına yazdırıldı.

dizi[2] = dizi[3];
// Dizinin 3 index numaralı değeri 2 index numaralı alana aktarıldı.

label1.Text = dizi[4].ToString();
// Dizinin 4 index numaralı değeri label1 nesnesinin text özelliğine aktarıldı.
```

Uygulama-2

Bir Boyutlu Diziler

Bu işlem, Uygulama 1'deki 5 elemanlı isimler dizisinde yer alan değerleri, Listele butonuna tıklayarak ListBox nesnesinin içinde göstermek olacaktır.

Adım 1: Görsel 4.8'de verilen formun tasarımını yapınız. Form kontrol nesnelerine name değerlerini veriniz.

Görsel 4.8: Dizi uygulamaları form tasarımı-2

Adım 2: Listele butonu Click olayına dizi içindeki her bir elemanı for döngüsü yardımıyla ListBox içinde göstermek için Items.Add() metodu kullanarak listeleme işlemi yapan kodlamaları yazınız.

```
private void btnListele_Click(object sender, EventArgs e)
{
  for (int i = 0; i < isimler.Length; i++)
  {
     lbListe.Items.Add(isimler[i]);
  }
}
```

Sıra Sizde

1. Kodlarda dizi isminden sonra **Length** ifadesi hangi amaç için kullanılmıştır?
2. Uygulamada 5 elemanlı dizinin tamamına değer aktarmadan listele butonuna tıklandığında nasıl bir hata ile karşılaşılır? Karşılaşılan bu hatanın çözümü için ne yapılabilir?
3. Dizi değerlerini listeleme işlemi, **for** döngüsü yerine **while** döngüsü kullanılarak nasıl yapılır?

4.1.5. Dizilerde Foreach Döngüsü Kullanımı

Birçok programlama dili, diziler üzerinde işlem yapılmasını kolaylaştıran bir döngü sunar. Bu döngü, foreach döngüsüdür. Dizilerde kullanılan foreach, dizi elemanlarını ilk elemandan başlayıp dizinin son elemanına kadar her elemanı tek tek dolaşarak belirlenen bir değişkene aktarır. Örneğin, 10 elemanlı bir dizide foreach döngüsü kullanıldığında döngü 10 defa tekrarlama işlemi yapar. Döngü her seferinde dizi içindeki değeri alarak aynı tipte olan bir değişkene aktarır. Döngünün yapısı aşağıda verilmiştir.

```
foreach (Tip Değişken in Dizi)
{
  // Döngü içindeki işlemler
}
```

Foreach döngüsünün yapısındaki ögeler aşağıda sıralanmıştır:

- **Tip:** Dizi içindeki veri tipleri ile aynı olmalıdır (Dizi içindeki değerler string ise Tip de string, double ise Tip de double olmalıdır.). Bazı durumlarda Tip olarak **var** kullanılır. Var tipi, kendisine atanan değer ne ise o değerin tipini alır.
- **Değişken:** Döngü, dizi içindeki değeri her dönme işleminde belirtilen bir değişkene aktarır.
- **in:** Bir anahtar kelimedir, foreach döngülerinde değişken adlarından sonra kullanılır.
- **Dizi:** Üzerinde işlem yapılacak dizinin adıdır.

```
int[] sayilar = { 20, 30, 40, 50 };
foreach (int sayi in sayilar)
{
  Console.WriteLine(sayi);
}
```

```
int[] sayilar = { 20, 30, 40, 50 };
for (int i = 0; i < sayilar.Length; i++)
{
  Console.WriteLine(sayilar[i]);
}
```

Yukarıdaki örneklerde iki döngü de aynı görevi yerine getirmektedir. Dizilerde for döngüsüne göre daha az kod yazarak sonuca ulaşıldığı için genellikle foreach döngüsü tercih edilmektedir. Foreach döngüsünde dizi içindeki değerler döngü tamamlanıncaya kadar sırasıyla sayi ismindeki değişkene aktarılır. For döngüsünde i değişkeni, dizinin index numarası olarak dizi elemanlarına erişim için kullanılır.

Sıra Sizde

Bir Boyutlu Diziler Uygulama 2'deki dizi elemanlarını listele butonuna tıklama olayında ListBox içine foreach döngüsü kullanarak listeleyiniz.

Uygulama-3

Bir Boyutlu Diziler

Bu işlemde Bir Boyutlu Diziler Uygulama 2'deki 5 elemanlı isimler dizisiyle birlikte integer tipindeki değerleri saklayan 5 elemanlı **notlar** isimli dizi kullanılacaktır. **Listele** butonuna tıklandığında her iki dizideki değerlerin ListBox nesnesinin içinde gösterilmesi sağlanacaktır.

Adım 1: Görsel 4.9'da verilen formun tasarımını yapınız. Form kontrol nesnelerine name değerlerini veriniz.

Görsel 4.9: Dizi uygulamaları form tasarımı-3

Adım 2: Kullanılacak dizileri ve index değişkenini global olarak oluşturunuz.

```
string[] isimler = new string[5]; //Global dizi
int[] notlar = new int[5];      //Global dizi
int index = 0;                  //Global değişken
```

Adım 3: Ekle butonunun Click olayına aşağıdaki kodları yazarak dizilere ad, soyad ve ders notu bilgilerini aktarınız.

```
private void btnEkle_Click(object sender, EventArgs e)
{
  if(index<isimler.Length)
  {
     isimler[index] = txtAdSoyad.Text;
     notlar.[index] = int.Parse(txtDersNotu.Text);
     // int.Parse fonksiyonu girilen değeri "int" veri türüne dönüştürür
     index++;
     txtAdSoyad.Text = "";
     txtDersNotu.Text = "";
  }
}
```

Adım 4: Listele butonu Click olayında aşağıdaki kodları yazarak dizideki değerleri ListBox kontrolü içinde gösteriniz.

```
private void btnListele_Click(object sender, EventArgs e)
{
 for (int i = 0; i < isimler.Length; i++)
 {
   if(isimler[i] != null)
   {
     lbListe.Items.Add(isimler[i]+" > "+notlar[i]);
   }
 }
}
```

Sıra Sizde

1. Adım 3'te karşılaştırma ifadesinin kullanım amacı nedir?
2. Adım 4'te karşılaştırma ifadesinin kullanım amacı nedir?
3. Adım 3'te for döngüsü yerine foreach döngüsü kullanılabilir mi? Neden?

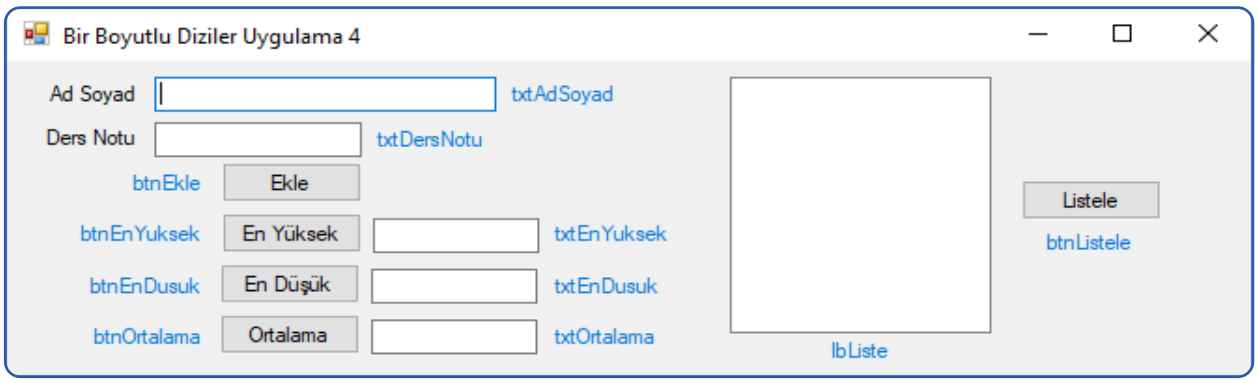
Uygulama-4

http://kitap.eba.gov.tr/KodSor.php?KOD=21077

Bir Boyutlu Diziler

Bu işlem, Bir Boyutlu Diziler Uygulama 3'ün geliştirilmiş hâlidir. Bir önceki uygulamada isimler ve notlar dizilerine değer aktarımı sağlanmıştı. Bu uygulamada ise girilen notlar içinde en yüksek ve en düşük not ile tüm notların ortalamasını hesaplama işlemi yapılacaktır.

Adım 1: Görsel 4.10'da verilen formun tasarımını yapınız. Form kontrol nesnelerine name değerlerini veriniz.



Görsel 4.10: Dizi uygulamaları form tasarımı-4

Adım 2: En Yüksek butonu Click olayında notlar dizisi içindeki en yüksek notu bulduran kodlamayı yapınız.

```
private void btnEnYuksek_Click(object sender, EventArgs e)
{
 int enyuksek = notlar[0];
 for (int i = 0; i < notlar.Length; i++)
 {
    if (notlar[i] > enyuksek)
    {
      enyuksek = notlar[i];
    }
 }
 txtEnYuksek.Text = enyuksek.ToString();
}
```

Adım 3: En Düşük butonu Click olayında notlar dizisi içindeki en düşük notu bulduran kodlamayı yapınız.

```
private void btnEnDusuk_Click(object sender, EventArgs e)
{
  int endusuk = notlar[0];
  for (int i = 0; i < notlar.Length; i++)
  {
    if (notlar[i] < endusuk)
       endusuk = notlar[i];
  }
  txtEnDusuk.Text = endusuk.ToString();
}
```

Adım 4: Ortalama butonu Click olayında notlar dizisi içindeki notların ortalamasını bulduran kodlamayı yapınız.

```
private void btnOrtalama_Click(object sender, EventArgs
e)
{
  int toplam=0;
  double ortalama=0;
  for (int i = 0; i < notlar.Length; i++)
  {
     toplam += notlar[i];
  }
   ortalama = toplam / notlar.Length;
 txtOrtalama.Text = ortalama.ToString();
}
```

Sıra Sizde

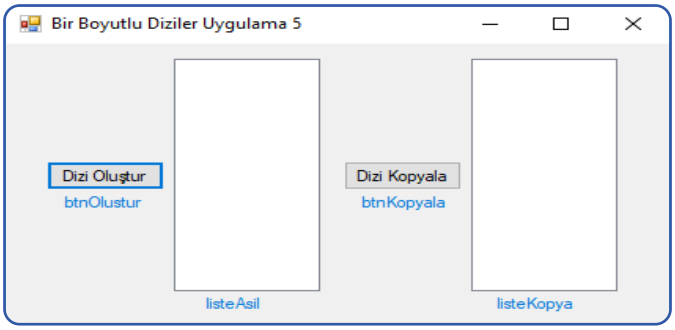
1. Adım 2'nin çalışma mantığının nasıl olduğunu açıklayınız.
2. Adım 2'de kullandığınız kodda neden dizinin ilk elemanı bir değişkene aktarıldı?
3. Adım 2 ve 3'teki karşılaştırma ifadelerinin kullanım amacı nedir?
4. Adım 4'te toplam ve ortalama değişkenlerine neden başlangıç değeri olarak 0 verildi?
5. Adım 4'te ortalama değişkeninin tipi neden double olarak seçildi?

Uygulama-5

Bir Boyutlu Diziler

Bu uygulamada bir dizinin elemanları başka bir diziye kopyalanacaktır.

Adım 1: Görsel 4.11'de verilen formun tasarımını yapınız. Form kontrol nesnelerine name değerlerini veriniz.



Görsel 4.11: Dizi uygulamaları form tasarımı-5

Adım 2: Formda kaynak dizi ve kopyalanacak diziyi 100 elemanlı ve integer tipinde sayıları saklayacak şekilde global olarak oluşturunuz.

```
int[] diziKaynak = new int[100];
int[] diziKopya = new int[100];
```

Adım 3: Dizi Oluştur butonu Click olayında kaynak diziye 0 ile 100 arasında rastgele sayılardan oluşan değerlerin aktarımı yapılacaktır. Bu işlem için Random sınıfından üretilen rastgele isimli nesneye Next metodu ile birlikte 0 ile 100 arasında üretilen sayıları diziye aktaran ve dizi değerlerini listeleyen kodlamayı yapınız.

```
private void btnOlustur_Click(object sender, EventArgs e)
{
  Random rastgele = new Random();
  for (int i = 0; i < diziKaynak.Length; i++)
  {
     diziKaynak[i] = rastgele.Next(0, 101);
}
  for (int i = 0; i < diziKaynak.Length; i++)
  {
     listeAsil.Items.Add(diziKaynak[i]);
  }
}
```

Adım 4: Dizi Kopyala butonu Click olayında ise kaynak dizideki değerleri kopyalanacak diziye aktarım işlemini yaptıktan sonra kopyalanan diziyi listeleyen kodlamaları yapınız.

```
private void btnKopya_Click(object sender, EventArgs e)
{
  for (int i = 0; i < diziKopya.Length; i++)
  {
     diziKopya[i] = diziKaynak[i];
  }
  for (int i = 0; i < diziKopya.Length; i++)
  {
     listeKopya.Items.Add(diziKopya[i]);
  }
}
```

Sıra Sizde

1. Listeleme işlemlerini foreach döngüsü ile gerçekleştiriniz.
2. Dizi kopyalama işleminin daha kısa yolu var mıdır? Oluşturacağınız küçük gruplarla dizi kopyalama işleminin kısa yolunun olup olmadığını tartışınız. Sonuçları sınıfla paylaşınız.

4.1.6. Bir Boyutlu Dizilerde Kullanılan Özellikler ve Metotlar

Dizilerle işlem yaparken işleri kolaylaştıracak bazı metotlar mevcuttur. Bu metotlar, dizilerin daha etkin kullanılabilmesine olanak sağlar.

- **Length Özelliği:** Dizideki eleman sayısını verir. Bu özelliğin kullanımı **diziadi.Length** şeklindedir.
- **Rank Özelliği:** Dizideki boyut sayısını verir. Bu özelliğin kullanımı **diziadi.Rank** şeklindedir.
- **Max Metodu:** Dizideki en büyük sayıyı verir. Bu metodun kullanımı **diziadi.Max()** şeklindedir.
- **Min Metodu:** Dizideki en küçük sayıyı verir. Bu metodun kullanımı **diziadi.Min()** şeklindedir.
- **Sum Metodu:** Dizideki sayıların toplamını verir. Bu metodun kullanımı **diziadi.Sum()** şeklindedir.
- **Average Metodu:** Dizideki sayıların ortalamasını verir. Bu metodun kullanımı **diziadi.Average()** şeklindedir.
- **First Metodu:** Dizideki ilk elemanı verir. Bu metodun kullanımı **diziadi.First()** şeklindedir.
- **Last Metodu:** Dizideki son elemanı verir. Bu metodun kullanımı **diziadi.Last()** şeklindedir.

Sıra Sizde

Bir form tasarımı yapınız. Formun içine bir adet Button ve bir adet ListBox ekleyiniz. Butona tıklandığında oluşturduğunuz sayısal tipteki diziye 0 ile 10 arasında rastgele değerler aktarınız. Değer aktarımı yapılan dizi için hangi metot ve özelliklerin kullanılabileceğini sınıfla paylaşınız. Elde edilen sonuçları ListBox içinde gösteren kodlamayı yapınız.

4.1.7. Çok Boyutlu Diziler

Tek boyutlu diziler, aynı veri tipinden değerler içeren ve tek satırdan oluşan dizilerdir. Çok boyutlu diziler ise tek boyutlu dizilerin genişletilmiş biçimidir. Çok boyutlu diziler genellikle matematiksel hesaplamalar, görüntü işleme ve kayıt işlemlerinde kullanılır. Çok boyutlu diziler içinde en çok iki boyutlu ve üç boyutlu diziler kullanılmaktadır. Bu öğrenme biriminde iki boyutlu diziler açıklanacaktır.

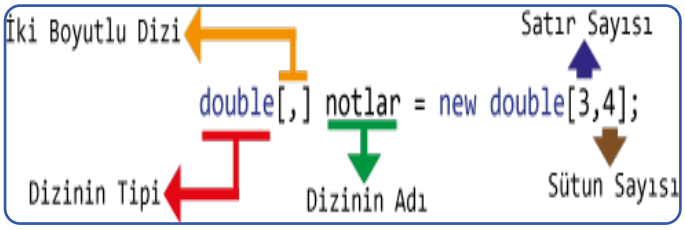
İki boyutlu diziler, tek boyutlu dizilerin birden çok satırdan oluşmuş hâlidir. Satranç tahtası, iki boyutlu dizilere örnek olarak verilebilir. Satranç tahtasında 8 satır ve 8 sütundan oluşan kareler ile her karenin satır ve sütunlarının kesişmesinden oluşan bir adresi vardır. Bu adresler kullanılarak karenin tahta üzerindeki konumu belirlenir. Tahtanın her karesinde bir veri saklanacağı varsayılırsa adresler kullanılarak karelerin içindeki verilerle işlem yapılabilir.

4.1.8. İki Boyutlu Dizi Tanımlama

Çok boyutlu dizilerin tanımlanmasında boyut sayısı, veri tipi belirtildikten sonra köşeli parantez içine yazılan virgül adediyle belirlenir. Örneğin; hiç virgül yoksa tek boyutlu dizi, bir virgül varsa iki boyutlu dizi, iki virgül varsa üç boyutlu dizi tanımlanacaktır.

```
int[,] dizi2d;        // İki boyutlu dizi
int[, ,] dizi3d;      // Üç boyutlu dizi
int[, , ,] dizi4d ;   // Dört boyutlu dizi
int[, , , ,] dizi5d;  // Beş boyutlu dizi
```

Görsel 4.12'de double tipinde sayılar saklayan **notlar** isminde **3** satır ve **4** sütunlu iki boyutlu bir dizi tanımlaması yapılmıştır. Derleyici, kodu derlediğinde bellek üzerinde 3*4=12 adet sayının saklanacağı bir dizi oluşturur.



Görsel 4.12: İki boyutlu dizi tanımlaması

Derleyici, tanımlanan diziyi derlediğinde bellek üzerindeki bu alanlara varsayılan olarak 0 değerini verir. Görsel 4.13'te sağ alt köşede kırmızı ile belirtilen adresler kullanılarak dizideki verilerle işlem yapılır. Tek boyutlu dizilerde olduğu gibi iki boyutlu dizilerde de hem satır hem sütun sıralaması daima 0'dan başlar.

Görsel 4.13: İki boyutlu dizinin satır ve sütunları

4.1.9. İki Boyutlu Diziye Değer Aktarma

İki boyutlu dizilere iki şekilde değer aktarılır.

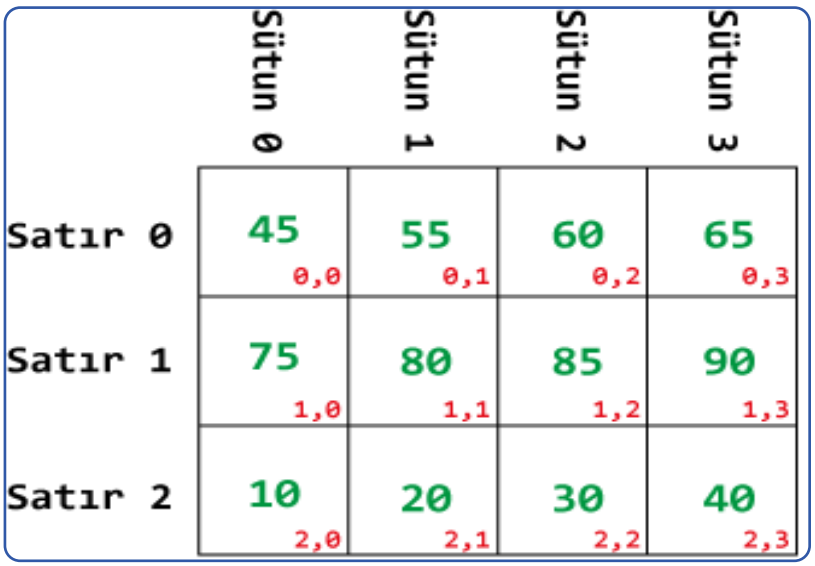
- İki boyutlu diziye tanımlandığı satırda değer aktarımı yapılabilir.

```
double[,] notlar = new double[3,4] {{45,55,60,65},{75,80,85,90},{10,20,30,40}};
```

- Dizinin index numaraları kullanılarak değer aktarımı yapılabilir.

```
double[,] notlar = new double[3, 4];
notlar[0, 0] = 45;
notlar[0, 1] = 55;
notlar[0, 2] = 60;
notlar[0, 3] = 65;
notlar[1, 0] = 75;
notlar[1, 1] = 80;
notlar[1, 2] = 85;
notlar[1, 3] = 90;
notlar[2, 0] = 10;
notlar[2, 1] = 20;
notlar[2, 2] = 30;
notlar[2, 3] = 40;
```

Diziye değer aktarımı yapıldıktan sonra dizinin değerleri Görsel 4.14'teki gibi olur.



Görsel 4.14: İki boyutlu diziye değer aktarma

Sıra Sizde

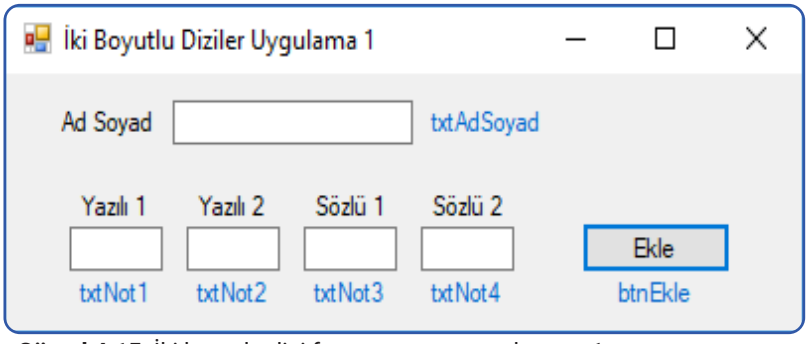
Sınıfınızda üçer kişilik dört takım kuracağınızı düşününüz. İki boyutlu dizi oluşturarak her satırda bir takım bulunacak şekilde arkadaşlarınızın isimlerini bu diziye aktarma işlemini yapınız.

Uygulama-1

İki Boyutlu Diziler

Bu uygulamada üç öğrenciye ait 4 adet ders notu girişi yapılacaktır. Öğrenci adları tek boyutlu diziye, öğrenciye ait ders notları iki boyutlu diziye aktarılacaktır.

Adım 1: Görsel 4.15'te verilen form tasarımını yapınız. Form üzerindeki kontrollerin name değerlerini veriniz.



Görsel 4.15: İki boyutlu dizi form tasarımı uygulaması-1

Adım 2: Öğrenci adı soyadı değerlerinde tek boyutlu dizi, not değerlerinde iki boyutlu dizi kullanılacağı için dizi tanımlamalarını global olarak yapınız.

```
string[] isimler = new string[3]; //Tek boyutlu global dizi
int[,] notlar = new int[3,4];    //İki boyutlu global dizi
int index = 0;                   //Global değişken
```

Adım 3: Ekle butonu Click olayında TextBox'lara girilen değerleri ilgili dizilere aktarma işlemini gerçekleştiriniz.

```
private void btnEkle_Click(object sender, EventArgs e)
{
  isimler[index] = txtAdSoyad.Text;
  notlar[index, 0] = int.Parse(txtNot1.Text);
  notlar[index, 1] = int.Parse(txtNot2.Text);
  notlar[index, 2] = int.Parse(txtNot3.Text);
  notlar[index, 3] = int.Parse(txtNot4.Text);
  index++;
}
```

Sıra Sizde

1. Görsel 4.15'teki uygulamanın çalışma mantığını açıklayınız.
2. Görsel 4.15'teki uygulamada notları girilecek öğrenci sayılarını form içindeki bir TextBox'tan alacak şekilde kodları düzenleyiniz.

4.1.10. İki Boyutlu Dizi Elemanlarına Erişim

Bir boyutlu dizilerde olduğu gibi iki boyutlu dizi elemanlarına erişmek için de index numaraları kullanılır. Erişim sağlanan dizi elemanı; değişkenlere aktarma, hesaplamalar yapma, ekrana yazdırma, nesneye aktarma gibi işlemlerde kullanılır.

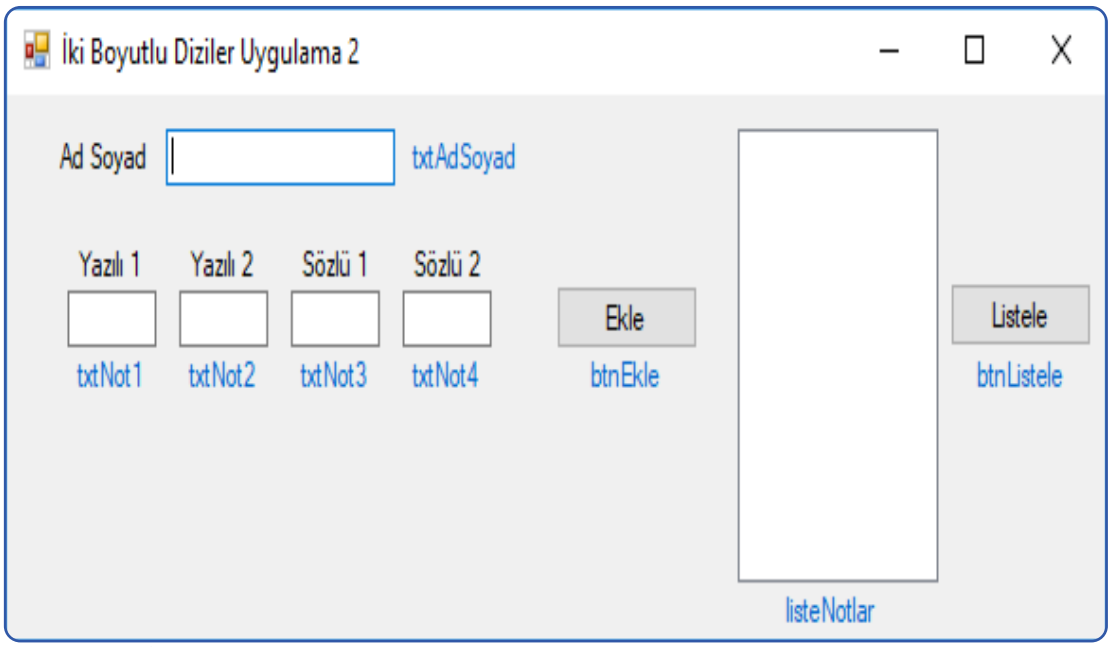
```
double[,] notlar = new double[3,4] {{45,55,60,65},{75,80,85,90},{10,20,30,40}};
// 3x4 elemanlı iki boyutlu dizi tanımlaması yapıldı. Aynı satırda değerler verildi.

int sayi1 = notlar[0,1];
// Dizinin 0,1 index numaralı değeri sayi1 ismindeki integer değişkenine aktarıldı.

int toplam = notlar[1,0] + notlar[1,1] + notlar[1,2] + notlar[1,3];
// Dizisinin 1. index numaralı satırındaki elemanlarla toplama yapıldı.

Console.WriteLine(notlar[2,1]);
// Dizisinin 2,1 index numaralı değeri Console ekranına yazdırıldı.

notlar[2,0] = notlar[1,3];
// Dizisinin 1,3 index numaralı değeri 2,0 index numaralı alanına aktarıldı.

label1.Text = notlar[2,2].ToString();
// Dizinin 2,2 index numaralı değeri label1 nesnesinin text özelliğine aktarıldı.
```

Uygulama-2

İki Boyutlu Diziler

İki Boyutlu Diziler Uygulama 1'de diziye aktarılan not bilgilerinin ortalaması hesaplanarak ListBox içinde gösterilecektir.

Adım 1: Uygulama 1 formuna Görsel 4.16'da olduğu gibi ListBox ve Button kontrollerini ekleyiniz.



Görsel 4.16: İki boyutlu dizi form tasarımı uygulaması-2

Adım 2: Listele butonu Click olayında isimlerle birlikte girilen notların ortalamasını hesaplayarak ListBox içinde gösteren kodlamaları yapınız.

```
private void btnListele_Click(object sender, EventArgs e)
{
  double toplam;
  for (int x = 0; x < 3; x++)
  {
    toplam = 0;
    for (int y = 0; y < 4; y++)
    {
      toplam += notlar[x, y];
    }
    listeNotlar.Items.Add(isimler[x]+" => "+ toplam / 4);
  }
}
```

Sıra Sizde

Adım 2'de yazılan kodların çalışma mantığını açıklayınız.

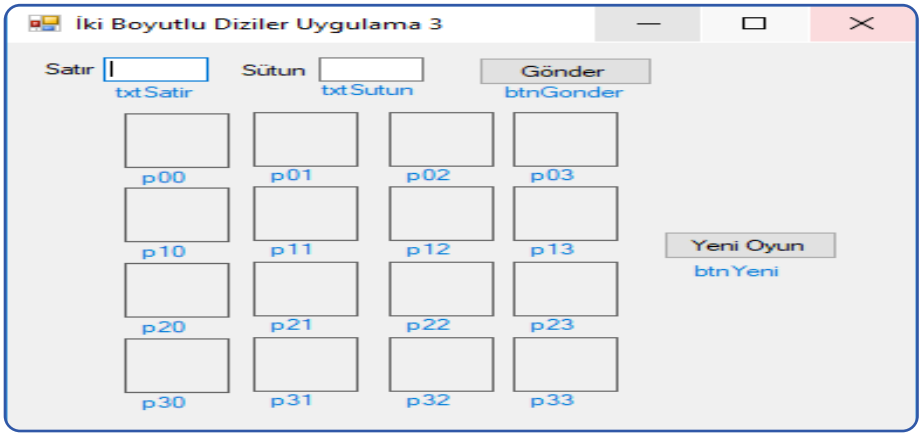
Uygulama-3

http://kitap.eba.gov.tr/KodSor.php?KOD=21078

İki Boyutlu Diziler

Bu uygulamada iki boyutlu diziler kullanılarak basit bir oyun programı yapılacaktır. Oyun programının amacı, 4 satır ve 4 sütundan oluşan iki boyutlu dizide rastgele bir konuma 1 değerini aktarmaktır. Dizinin geri kalan değerleri 0 olarak bırakılacaktır. Satır ve sütun bilgileri, form üzerinde bulunan iki adet TextBox'tan alınacaktır. Alınan satır ve sütun bilgileri dizinin satır ve sütun konumundaki değerle karşılaştırılarak dizi içindeki değer 1 ise oyun kaybedilecek ve ilgili PictureBox'ın rengi kırmızı olacak, dizi içindeki değer 1 değil ise ilgili PictureBox'ın rengi yeşil olacak şekilde kodlama yapınız.

Adım 1: Yeni bir form oluşturarak 16 adet PictureBox, 2 adet TextBox ve 2 adet Button kontrolünü Görsel 4.17'deki gibi ekleyiniz. Kontrollere name değerlerini veriniz.



Görsel 4.17: İki boyutlu dizi form tasarımı uygulaması-3

Adım 2: Byte tipinde 4 satır ve 4 sütundan oluşan iki boyutlu diziyi global olarak tanımlayınız.

```
byte[,] dizi = new byte[4, 4];
```

Adım 3: Yeni Oyun butonu Click olayında dizi içinde rastgele bir konuma 1 değerini aktarmak için Random sınıfından üretilen rastgele isminde bir nesne kullanınız. Random sınıfından üretilen bu nesne Next metodu ile birlikte 0 ile 4 arasında, 4 dâhil olmayacak şekilde bir sayı vermektedir. Üretilen bu sayılar, dizinin herhangi bir konumuna 1 değerini aktarmak için kullanılmaktadır. Dizinin belirtilen konumundaki sayı 1 olurken diğer konumlardaki değerler 0 olarak kalmaktadır.

```
private void btnYeni_Click(object sender, EventArgs e)
{
 Random rastgele = new Random(); // rastgele isminde bir Random nesnesi oluşturdu.
 int satirRastgele = rastgele.Next(4);// 0-4 arası (4 dâhil değil) üretilen sayıdır.
 int sutunRastgele = rastgele.Next(4); // 0-4 arası (4 dâhil değil) üretilen sayıdır.
 dizi[satirRastgele, sutunRastgele] = 1; // Dizi içinde rastgele bir konuma 1 değeri aktarıldı.
}
```

Adım 4: Gönder butonu Click olayında satır ve sütun TextBox'larına girilen bilgileri satir ve sutun isimli değişkenlere aktarınız. Bu değişkenleri kullanarak önce hangi PictureBox'ın seçileceğini belirleyiniz. Örneğin; satır değeri 1, sütun değeri 3 ise formdaki PictureBox kontrollerinden name değeri "p13" seçilecektir. Dizi içinde [1,3] konumundaki değer 1 ise kutu rengi kırmızı, değilse kutu rengi yeşil olmaktadır.

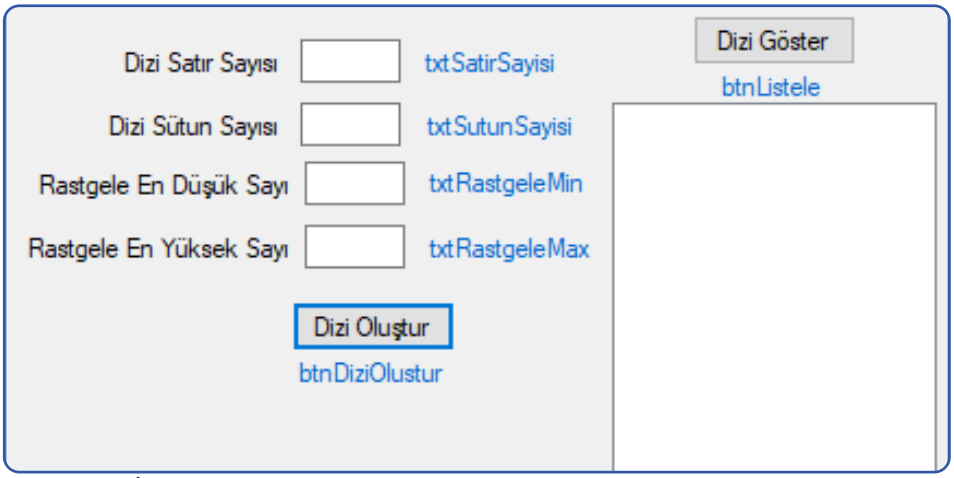
```
private void btnGonder_Click(object sender, EventArgs e)
{
byte satir = byte.Parse(txtSatir.Text);
byte sutun = byte.Parse(txtSutun.Text);
PictureBox kutu=this.Controls.Find("p"+satir+sutun, true)[0]as PictureBox;
// Controls.Find fonksiyonu name değerlerine göre  form içindeki kontrolleri bulmak için kullanılır.
// Bulunan kontroller as operatörü ile  PictureBox nesnelerine dönüştürülür.
byte durum = dizi[satir, sutun];
if (durum == 0)
 {kutu.BackColor = Color.Green; }
 else
 {kutu.BackColor = Color.Red; }
}
```

Uygulama-4

İki Boyutlu Diziler

Bu uygulamada satır ve sütun sayıları kullanıcı tarafından verilen iki boyutlu bir diziye form üzerinden girilen belirli aralıktaki rastgele sayıları aktarma işlemi yapılacaktır. Sonraki işlem, bu dizinin değerlerini ListBox nesnesi içinde göstermektir.

Adım 1: Görsel 4.18'deki form tasarımını yapınız ve form üzerindeki kontrollerin name değerlerini veriniz.



Görsel 4.18: İki boyutlu dizi form tasarımı uygulaması-4

Adım 2: Dizinin satır ve sütun sayıları birden çok yerde kullanılacağı için diziyi global olarak tanımlayınız.

```
int[,] dizi;
int satirSayisi;
int sutunSayisi;
```

Adım 3: Dizi Oluştur butonu Click olayında formdan gelen satır ve sütun bilgileri ile bir dizi oluşturunuz. Oluşturulan bu diziye aktarılacak rastgele sayıların hangi aralıkta olacağı bilgisi kullanılarak iki boyutlu bir dizi elde edilecektir. Bu dizinin her elemanına içi içe döngü kullanılarak rastgele sayılar atanacaktır.

```
private void btnDiziOlustur_Click(object sender, EventArgs e)
{
 int rastgeleMin= int.Parse(txtRastgeleMin.Text);
 int rastgeleMax = int.Parse(txtRastgeleMax.Text);
 satirSayisi = int.Parse(txtSatirSayisi.Text);
 sutunSayisi = int.Parse(txtSutunSayisi.Text);
 dizi = new int[satirSayisi, sutunSayisi];
 Random rastgele = new Random();
        for (int x = 0; x < satirSayisi; x++)
         {
                 for (int y = 0; y < sutunSayisi; y++)
                {
                        dizi[x, y] = rastgele.Next(rastgeleMin, rastgeleMax);
                }
          }
}
```

Adım 4: Dizi Göster butonu Click olayında rastgele sayılarla oluşturulan dizinin ListBox nesnesi içinde ait olduğu satır ve sütun bilgisi ile satır ve sütuna ait değerleri listeleyiniz.

```
private void btnlistele_Click(object sender, EventArgs
e)
{
  for (int x = 0; x < satirSayisi; x++)
  {
     for (int y = 0; y < sutunSayisi; y++)
     {
       listeDizi.Items.Add(x+","+y+" => "+dizi[x,y]);
     }
  }
}
```

Sıra Sizde

1. Aynı uygulamayı üç boyutlu dizi kullanarak yeniden düzenleyiniz.
2. Üç boyutlu dizi oluşturmak ve listelemek için kaç adet iç içe döngü kullanmak gereklidir? Neden?

4.2. Koleksiyonlar

Diziler, programlama dillerinde çokça kullanılan yapılardır. Dizilerde çok sayıda değer tutulabilir ve onlara erişim sağlanabilir ancak dizilerle işlem yaparken iki sınırlama ile karşılaşılır. Bu sınırlamalar şunlardır:

1. Dizilere aktarılacak değerler dizi ile aynı tipte olmalıdır.
2. Dizilerin eleman sayıları önceden belirlenmelidir.

Programlamada bazen aynı veri tipine sahip olmayan değerlerle işlem yapmak zorunda kalınabilir. Nesne Tabanlı Programlama bu durumda koleksiyonları (collections) sunar. Koleksiyonları kullanabilmek için projeye **System.Collections** isim uzayı (namespace) dâhil edilmelidir. Koleksiyonların isim uzayına eklenmesi aşağıda verilmiştir.

```
using System.Collections;
```

Dizi ve koleksiyon arasındaki temel farklar Tablo 4.1'de belirtilmiştir.

Tablo 4.1: Dizi ve Koleksiyon Karşılaştırması

Diziler	Koleksiyonlar
Diziler aynı tip verileri saklar.	Koleksiyonlar aynı veya farklı tipteki verileri saklar.
Dizilerin eleman sayısı başlangıçta belirlenir ve sonradan değiştirilemez.	Koleksiyonların eleman sayılarını başlangıçta belirtmeye gerek yoktur.
Dizilerin eleman sayıları sabittir. Bu nedenle ihtiyaç durumunda eleman sayılarının artırılmasına ve azaltılmasına izin vermez.	Koleksiyonların eleman sayıları değiştirilebilir. İhtiyaca göre eleman sayıları artırılabilir ve azaltılabilir.
Diziler performans açısından koleksiyonlardan daha hızlıdır.	Koleksiyonlar performans açısından dizilerden daha yavaştır.
Bellek açısından dizilerin kullanılması tavsiye edilmez.	Bellek açısından koleksiyonların kullanılması tavsiye edilir.
Dizilerin elemanları üzerinde işlem yapmak için koleksiyonlardan daha az hazır metodu vardır.	Koleksiyonların elemanları üzerinde işlem yapmak için dizilerden daha fazla hazır metodu vardır.

Kodlama yaparken özel isteğe uyarlanmış koleksiyon oluşturulabileceği gibi Nesne Tabanlı Programlama dili ile gelen hazır koleksiyonlar da kullanılabilir. Hazır koleksiyonlardan bazıları şunlardır:

- ArrayList
- List
- Queue-Stack
- Dictionary
- HashTable
- SortedList

4.2.1. Boxing (Kutulama)-Unboxing (Kutu Açma)

Nesne Tabanlı Programlama dilinde **Value Type** (değer tipi) ve **Reference Type** (referans tipi) olmak üzere iki veri tipi vardır. Değer tipleri; int, char, byte, double vb. belleğinin **Stack** adı verilen kısmında tutulur. Referans tipleri ise object, string, class vb. belleğin **Heap** adı verilen kısmında tutulur.

Boxing, herhangi value (değer) tipideki değişkenin object (nesne) tipindeki değişkene dönüştürülmesidir.

```
int x = 1234;
object obj;
obj = x;
```

Unboxing, object (nesne) tipindeki değişkenin value (değer) tipindeki değişkene dönüştürülmesidir. Burada dikkat edilmesi gereken nokta, dönüştürülmek istenen nesne hangi tipte boxing yapılmış ise aynı tipte **Casting** yapılması gerekmektedir. Aksi durumda veri tipleri aynı olmadığı için dönüştürme hatası (InvalidCastException) alınacaktır.

```
int y = (int)obj;
```

4.2.2. ArrayList Koleksiyonu

ArrayList, dizilerde olduğu gibi veri saklama amacıyla kullanılan bir koleksiyondur. Eleman sayıları dinamik olarak değişir ve farklı tiplerde veri saklama imkânı sunar. ArrayList elemanlarına erişim için index numaraları kullanılır. ArrayList tanımlanırken eleman sayısını belirtmeye gerek yoktur. Dinamik yapısı sayesinde kodlama sırasında veya çalışma anında ekleme, silme, araya ekleme, değerleri değiştirme gibi işlemler yapılabilir. ArrayList içine eklenen elemanlar object tipinde olacağı için veri eklerken boxing (kutulama), veri alırken de unboxing (kutu açma) işlemi yapılmaktadır.

Koleksiyon nesnesi ArrayList'in oluşturulması aşağıda verilmiştir.

```
ArrayList liste = new ArrayList();
```

Yukarıdaki komut derlendiğinde **ArrayList** sınıfından **liste** adında bir nesne üretilmiş olur.

- **ArrayList Veri Ekleme**

Koleksiyonlara veri eklemek için **Add** metodu kullanılır. Koleksiyonlar da diziler gibi index numaralarına sahiptir. ArrayList'e ilk eklenen elemanın index numarası 0 olur, diğer elemanlar da 0'dan başlayarak sıralanır.

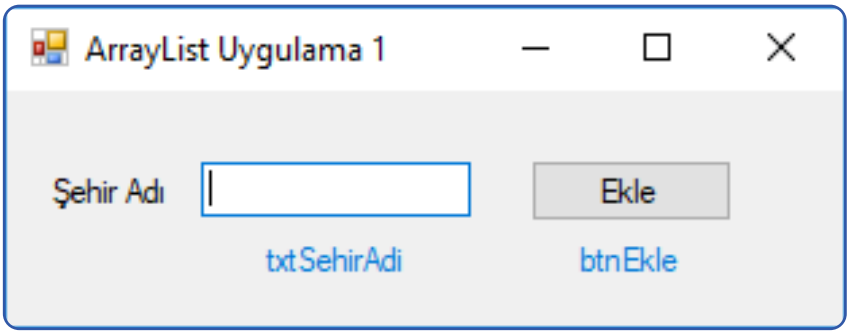
```
ArrayList liste = new ArrayList();
liste.Add("Bilişim"); // Metinsel
liste.Add(100);       // Tam sayı
liste.Add('m');       // Karakter
liste.Add(3.14);      // Ondalık sayı
liste.Add(true);      // Mantıksal
```

Uygulama-1

ArrayList

Şehir isimlerinin saklanacağı bir ArrayList oluşturarak formdan girilen şehir isimlerini tanımlanan ArrayList içine ekleme işlemi yapılacaktır.

Adım 1: Görsel 4.19'da verilen form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.



Görsel 4.19: ArrayList form tasarımı uygulaması-1

Adım 2: Ekle butonu Click olayında global olarak tanımlanan sehirler ismindeki ArrayList'e TextBox'tan gelen verileri aktarınız.

```
ArrayList sehirler = new ArrayList();
private void btnEkle_Click(object sender, EventArgs e)
{
 sehirler.Add(txtSehirAdi.Text);
}
```

- **ArrayList Elemanlarına Erişim**

ArrrayList içindeki elemanlara erişim için dizilerde olduğu gibi index numaraları kullanılır.

```
sehirler[0]  // 0 index numaralı ilk eleman
sehirler[1]  // 1 index numaralı ikinci eleman
sehirler[2]  // 2 index numaralı üçüncü eleman
```

Yukarıda gösterildiği gibi index numaraları 0'dan başlayarak eleman sayısının 1 eksiğine kadar istenilen değerlere erişim sağlanabilir. Dizilerde olduğu gibi ArrayList elemanlarına sınırları dışında bir index numarası ile ulaşılmaya çalışıldığında hata ile karşılaşılır.

```
sehirler[1]="Mardin"; // 1 index numaralı elemana erişim sağlanarak değeri değiştirildi.
```

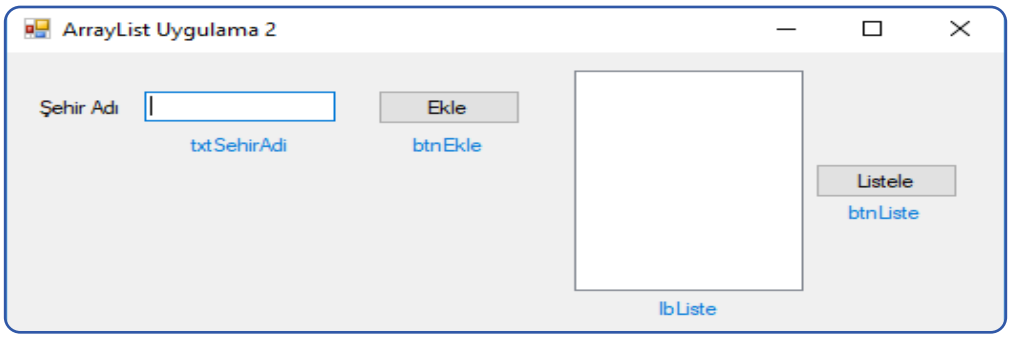
```
Label1.Text=(string)sehirler[1];  // 1 index numaralı elemana erişim sağlanarak unboxing işlemiyle
// Label kontrolü içinde gösterilmiştir.
```

Uygulama-2

ArrayList

ArrayList Uygulama 1 biraz daha geliştirilerek ArrayList nesnesine eklenen veriler index numaraları ile birlikte ListBox kontrolü içinde gösterilecektir.

Adım 1: Görsel 4.20'de verilen form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.



Görsel 4.20: ArrayList form tasarımı uygulaması-2

Adım 2: Listele butonu Click olayında aşağıdaki kodları yazarak listeleme işlemini gerçekleştiriniz.

```
private void btnListe_Click(object sender, EventArgs
e)
{
  for (int i = 0; i < sehirler.Count; i++)
  {
    lbListe.Items.Add(sehirler[i]);
  }
}
```

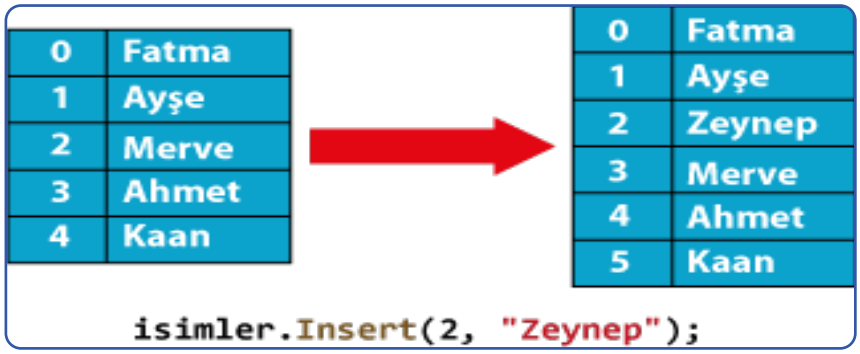
Uygulamada ArrayList koleksiyonu içindeki değerlere ulaşmak için bir döngü kullanıldı. Bu döngüyle 0'dan başlayarak koleksiyonun eleman sayısına kadar dönme işlemi yapıldı. Koleksiyonlarda eleman sayısı Count özelliği ile alınıp döngü değişkeni koleksiyon index numarası olarak kullanıldı ve her elemana erişim sağlandı.

Sıra Sizde

1. Adım 2'de kullanılan for döngüsü yerine foreach döngüsünü kullanarak aynı işlemleri yapınız.
2. Adım 2'de koleksiyon değerlerinin ListBox içinde gösterilmesinde niçin unboxing yapılmamıştır? Açıklayınız.

Insert metodu, koleksiyonlara değer eklemenin bir diğer yoludur. Bu metodun add metodundan farkı, eklenecek değerin hangi sıraya atanacağının belirtilmesidir. Insert metodu kullanımından sonra koleksiyonun bellek üzerindeki durumu Görsel 4.21'de verilmiştir.

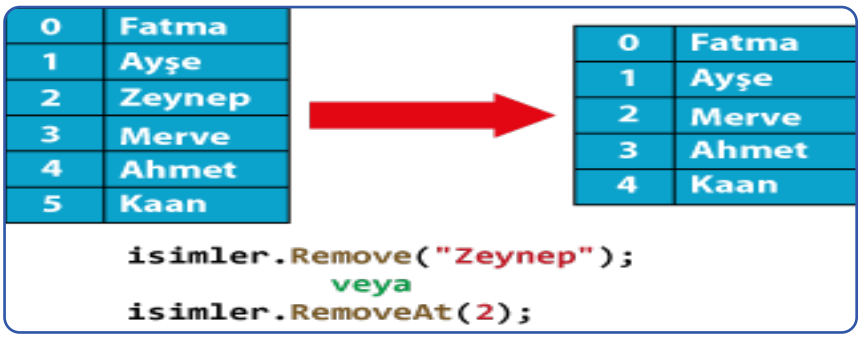
```
ArrayList isimler = new ArrayList();
isimler.Add("Fatma");
isimler.Add("Ayşe");
isimler.Add("Merve");
isimler.Add("Ahmet");
isimler.Add("Kaan");
isimler.Insert(2,"Zeynep");
```



Görsel 4.21: ArrayList insert metodu kullanımı

- **ArrayList Veri Silme**

ArrayList'lerde bir veriyi silmek için **Remove** ve **RemoveAt** metotları vardır. **Remove** metodunda silinecek nesnenin değeri, **RemoveAt** metodunda ise silinecek nesnenin index numarası kullanılır. Veri silme metodlarının kullanımından sonra koleksiyonun bellek üzerindeki durumu Görsel 4.22'de verilmiştir.



Görsel 4.22: ArrayList Remove ve RemoveAt metodu kullanımı

ArrayList içindeki tüm verileri silmek için **Clear()** metodu kullanılır.

Sıra Sizde

ArrayList içinde aynı değerlere sahip birden çok eleman varsa Remove metodu aynı değerlere sahip elemanlardan ilk bulunanı siler. Diğer elemanları da silmek için ne yapılmalıdır?

- **ArrayList Veri Arama**

Bir ArrayList içinde bir verinin aranması için birkaç yöntem bulunmaktadır. Bunlardan bir tanesi **Contains** metodudur. Contains metodu; aranan veri koleksiyonda varsa **true**, yoksa **false** değerini geriye döndürür.

```
if(isimler.Contains("Ahmet"))
{
  label1.Text = "Aranan veri bulundu.";
}
else
{
  label1.Text = "Bulunamadı.";
}
```

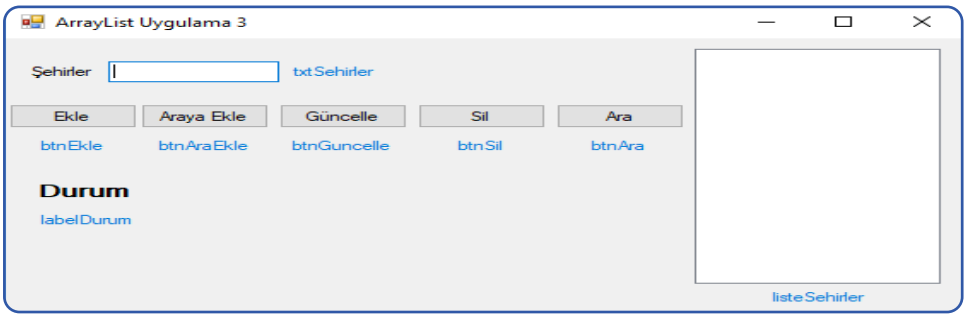
ArrayList içinde bir diğer arama yöntemi **IndexOf** metodudur. Bu metot, **Contains** metodundan farklı olarak koleksiyon içinde aranan veri bulunursa index numarasını, bulunamazsa **-1** değerini geriye döndürür.

```
int durum = isimler.IndexOf("Ahmet");
if (durum != -1)
{
   label1.Text = durum+ "    index numaralı aranan veri bulundu.";
}
else
{
   label1.Text = "Bulunamadı.";
}
```

Uygulama-3

Uygulamada sehirler isminde global tanımlanan bir ArrayList üzerinde ekleme, araya ekleme, güncelleme, silme ve arama işlemleri yapılacaktır.

Adım 1: Görsel 4.23'teki form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.



Görsel 4.23: ArrayList form tasarımı uygulaması-3

Adım 2: Ekle butonu Click olayında global olarak tanımlanan sehirler ismindeki ArrayList'e TextBox'tan gelen verileri aktarınız.

```
ArrayList sehirler = new ArrayList();
private void btnEkle_Click(object sender, EventArgs e)
{
  sehirler.Add(txtSehirler.Text);
  Listele();
}
```

Adım 3: Ekle butonu içinde Listele isminde bir metot kullanınız. Bu metodun görevi; eklenen, silinen veya değeri değiştirilen ArrayList elemanlarını ListBox kontrolü içinde göstermektir.

```
private void Listele()
{
  listeSehirler.Items.Clear();
  foreach (string sehir in sehirler)
  {
     listeSehirler.Items.Add(sehir);
  }
}
```

Adım 4: ListBox içindeki elemanın index numarası, ArrayList elemanları ile aynıdır. Araya Ekle butonu Click olayında ListBox içinden seçilen elemanın index numarası, **SelectedIndex** özelliği kullanılarak bir değişkene aktarılır. Bu değişkeni **insert** metodu içinde kullanarak araya ekleme işlemini gerçekleştiriniz.

```
private void btnAraEkle_Click(object sender, EventArgs e)
{
  int indexNo = listeSehirler.SelectedIndex;
  sehirler.Insert(indexNo, txtSehirler.Text);
  Listele();
}
```

Adım 5: Güncelle butonu Click olayında ListBox içinden seçilen elemanın index numarasını kullanarak ArrayList elemanlarının değerlerini değiştiriniz.

```
private void btnGuncelle_Click(object sender, EventArgs e)
{
  int indexNo = listeSehirler.SelectedIndex;
  sehirler[indexNo] = txtSehirler.Text;
  Listele();
}
```

Adım 6: Sil butonu Click olayında ListBox üzerinden index numarası alınan ArrayList elemanını silme işlemini yapınız.

```
private void btnSil_Click(object sender, EventArgs e)
{
  int indexNo = listeSehirler.SelectedIndex;
  sehirler.RemoveAt(indexNo);
  Listele();
}
```

Adım 7: Ara butonu Click olayında TextBox'a yazılan değeri ArrayList içinde arama işlemini yapınız. Aranan değer bulunursa labelDurum'a "Aranan Değer Bulundu." mesajı verilecek ve **IndexOf** metodu kullanılarak ListBox içindeki konumuna gidecektir. Aranan değer bulunamazsa "Aranan Değer Bulunamadı." mesajı verilecektir.

```
private void btnAra_Click(object sender, EventArgs e)
{
  if (sehirler.Contains(txtSehirler.Text))
  {
    labelDurum.Text = "Aranan Değer Bulundu.";
    listeSehirler.SelectedIndex = sehirler.IndexOf(txtSehirler.Text);
  }
  else
  {
     labelDurum.Text = "Aranan Değer Bulunamadı.";
  }
}
```

- **ArrayList Veri Sıralama**

ArrayList içindeki veriler, eklenme sırasına göre 0 index numarasından başlayarak devam eder. **Reverse** metodu bu sıralamayı tamamen tersine çevirmektedir.

```
sehirler.Reverse();
```

Bir diğer sıralama ise **Sort** metodudur. Sort metodu, ArrayList içindeki değerleri artan bir sıra hâlinde yeniden düzenlemektedir. Sort metodu; koleksiyondaki elemanlar yazı tipinde ise a'dan z'ye, sayısal tipte ise küçükten büyüğe doğru sıralar.

```
sehirler.Sort();
```

Sıra Sizde

ArrayList Uygulama 3'teki forma iki adet buton ekleyiniz. Butonlardan birinin görevi sıralama, diğerinin görevi ise tersten sıralama olsun. Bu butonlara tıklandığında ListBox içinde ArrayList'in sıralanmış veya tersten sıralanmış hâllerini gösteriniz.

4.2.3. List Koleksiyonu

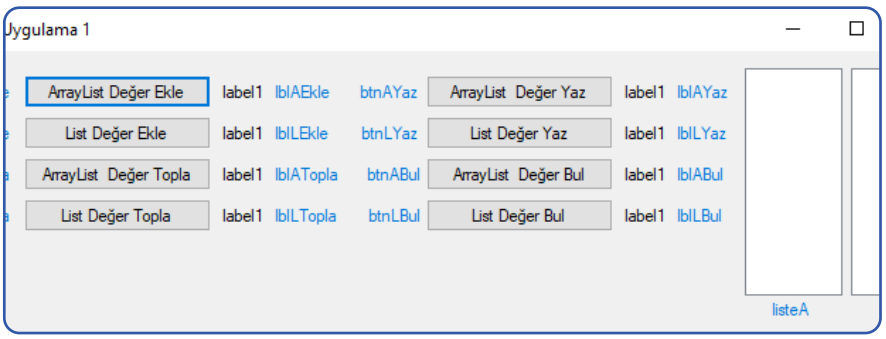
List koleksiyonu ile ArrayList koleksiyonu benzerlik gösterir. Aralarındaki fark; List koleksiyonunun generic, ArrayList koleksiyonunun ise non-generic yapıda olmasıdır. Generic koleksiyonlarda mutlaka içinde saklanacak verinin tipi belirtilmelidir çünkü belirlenen tipin dışında veri saklanmaya çalışıldığında hata ile karşılaşılır. Non-generic koleksiyonlarda ise bu işlem yapılmaz. Non-generic koleksiyonların veri tipi obje olarak belirlenir ve bu koleksiyonlarda her tipten veri saklanabilir. Generic koleksiyonlarda boxing-unboxing işlemleri yapılmazken non-generic koleksiyonlarda yapılır.

Uygulama-1

http://kitap.eba.gov.tr/KodSor.php?KOD=21080

Bu uygulamada ArrayList koleksiyonu ile List koleksiyonu işlem süresi olarak karşılaştırılacaktır.

Adım 1: Görsel 4.24'teki form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.



Görsel 4.24: List form tasarımı uygulaması-1

Adım 2: Bu uygulamada global olarak bir ArrayList, bir List koleksiyonu, her iki koleksiyonun eleman sayılarını belirleyecek integer değişken ve işlemlerde geçen süreyi hesaplamak için Stopwatch nesnesi kullanılmıştır. Stopwatch nesnesi için "using System.Diagnostics" isim uzayını ekleyiniz.

```
ArrayList kolA = new ArrayList();    // ArrayList koleksiyonu
List<int> kolL = new List<int>();  // List koleksiyonu
Stopwatch km = new Stopwatch(); // Geçen süreyi hesaplama için
nesne
int elemanSayisi = 100000;          // Test için eleman sayısı
```

```
private void btnAEkle_Click(object sender, EventArgs e)
{
 km.Start();
 for (int i = 0; i < elemanSayisi; i++)
 {
  kolA.Add(i);
 }
 km.Stop();
 lblAEkle.Text = "= " +
 km.Elapsed.TotalMilliseconds;
 km.Reset();
}
```

```
private void btnLEkle_Click(object sender, EventArgs e)
{
 km.Start();
 for (int i = 0; i < elemanSayisi; i++)
 {
  kolL.Add(i);
 }
 km.Stop();
 lblLEkle.Text = "= " +
 km.Elapsed.TotalMilliseconds;
 km.Reset();
}
```

```
private void btnATopla_Click(object sender, EventArgs e)
{
 km.Start();
 int toplam = 0;
 for (int i = 0; i < kolA.Count; i++)
 {
   toplam +=(int)kolA[i];
 }
 km.Stop();
 lblATopla.Text = "= " +
 km.Elapsed.TotalMilliseconds;
 km.Reset();
}
```

```
private void btnLTopla_Click(object sender, EventArgs e)
{
 km.Start();
 int toplam = 0;
 for (int i = 0; i < kolL.Count; i++)
 {
  toplam += kolL[i];
 }
 km.Stop();
 lblLTopla.Text = "= " +
 km.Elapsed.TotalMilliseconds;
 km.Reset();
}
```

```
private void btnAYaz_Click(object sender, EventArgs
e)
{
 km.Start();
 for (int i = 0; i < elemanSayisi; i++)
 {
  listeA.Items.Add(kolA[i]);
 }
 km.Stop();
 lblAYaz.Text= "= " +
 km.Elapsed.TotalMilliseconds;
 km.Reset();
}
```

```
private void btnLYaz_Click(object sender, EventArgs
e)
{
  km.Start();
  for (int i = 0; i < kolL.Count; i++)
  {
   listeL.Items.Add(kolL[i]);
  }
  km.Stop();
  lblLYaz.Text = "= " +
  km.Elapsed.TotalMilliseconds;
  km.Reset();
}
```

```
private void btnABul_Click(object sender, EventArgs e)
{
 km.Start();
 if(kolA.Contains(9999))
 {
  km.Stop();
 }
 lblABul.Text = "= " +
 km.Elapsed.TotalMilliseconds;
 km.Reset();
}
```

```
private void btnLBul_Click(object sender, EventArgs
e)
{
 km.Start();
 if (kolL.Contains(9999))
 {
   km.Stop();
 }
 lblLBul.Text = "= " +
 km.Elapsed.TotalMilliseconds;
 km.Reset();
}
```

Sıra Sizde

ArrayList ile List arasındaki işlem süresi sonuçlarını karşılaştırıp yorumlayınız.

4.2.4. Queue-Stack Koleksiyonları

Queue kelimesi kuyruk anlamına gelir. İlk giren eleman ilk çıkar işleyişine sahip bir koleksiyondur (FIFO-First In First Out). Koleksiyondan bir eleman çıkarılmak istendiğinde kuyruğa ilk eklenen eleman çıkartılacaktır. Yeni eklenecek eleman ise kuyruğun en sonuna getirilir.

Aslında bu yapıyla günlük hayatta çokça karşılaşılır. Örneğin, sıra numarasına göre işlem yapılan bir işletmede önce bir sıra numarası alınır. Alınan bu sıra numarası kuyruğun en sonundadır. İşlem yapılan numaralar sıradan çıkar ve işlem sırası diğer numaralara gelir.

Queue ekleme ve çıkarma işleminin iki metodu vardır:

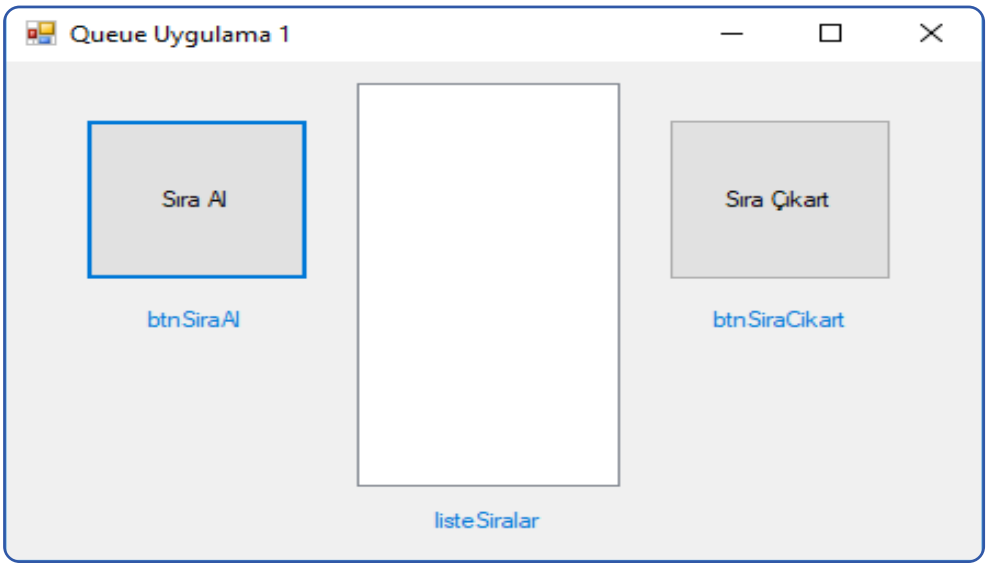
1. **Enqueue()** metodu kuyruğun sonuna bir eleman ekler.
2. **Dequeue()** metodu kuyruğun başındaki elemanı çıkarır.

Uygulama-1

Queue Koleksiyonu

Uygulamada Queue sınıfından üretilen nesne ile iki Button bir ListBox kontrolü kullanılarak kuyruğa alma ve kuyruktan çıkarma işlemleri yapılacaktır.

Adım 1: Görsel 4.25'teki form tasarımını yapınız ve form üzerindeki kontrollerin name değerlerini veriniz.



Görsel 4.25: Queue form tasarımı uygulaması-1

Adım 2: Queue sınıfından üretilecek nesneyi ve bu nesneye aktarılacak sıra numaraları için global değişkenleri oluşturunuz.

```
Queue kuyruk = new Queue();
int sira = 0;
```

Adım 3: Sıra Al butonu Click olayında sira değişkeninin değerini bir artırarak oluşturulan Queue nesnesine aktarınız.

```
private void btnSiraAl_Click(object sender, EventArgs e)
{
  sira++;
  kuyruk.Enqueue(sira);
  Listele();
}
```

Adım 4: Queue nesnesindeki değerleri ListBox içinde göstermek için Listele adında bir metot oluşturarak sıra durumunu bu metot içinde gösteriniz.

```
private void Listele()
{
  listeSiralar.Items.Clear();
  foreach (int sira in kuyruk)
  {
    listeSiralar.Items.Add(sira);
  }
}
```

Adım 5: Sıra Çıkart butonu Click olayında oluşturduğunuz Queue nesnesine verilen ilk değeri kuyruktan çıkarma işlemini gerçekleştiriniz. Listele metodunu kullanarak en güncel kuyruk değerlerini ListBox içinde gösteriniz.

```
private void btnSiraCikart_Click(object sender, EventArgs e)
{
  kuyruk.Dequeue();
  Listele();
}
```

Sıra Sizde

Adım 5'te kuyrukta hiç sıra yokken çıkartma işlemi yapıldığında nasıl bir hata ile karşılaşılır? Bu hatanın çözümü için ne yapılabilir? Önerilerinizi sınıfınızla paylaşınız.

Stack kelimesi yığın anlamına gelir. Son giren eleman ilk çıkar işleyişine sahip bir koleksiyondur (LIFO-Last In First Out). Koleksiyondan bir eleman çıkarılmak istendiğinde yığına son eklenen eleman çıkartılacaktır. Örneğin, ofis uygulamalarında yapılan işlemler uygulama tarafından hafızaya alınır. Uygulamada geri alma işlemi yapıldığında en son işlem gerçekleştirilir. Geri alma işlemine devam edildiğinde sondan başa doğru hafızadaki işlemler gerçekleştirilir.

Stack içine ekleme ve çıkarma işleminin iki metodu vardır:

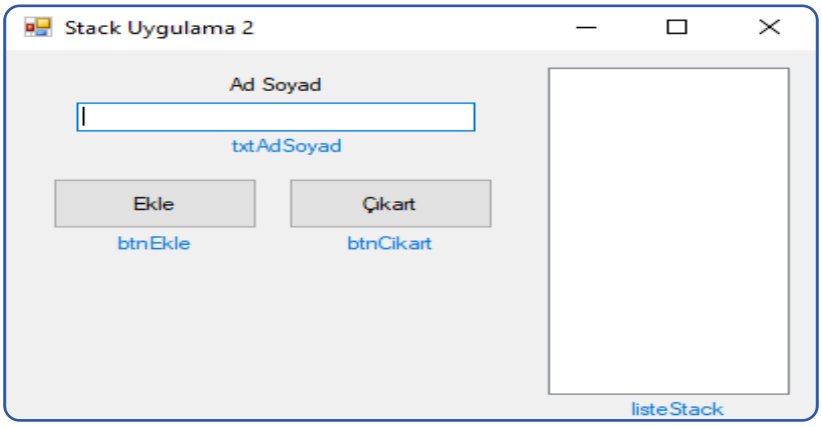
1. **Push()** metodu Stack içine bir değer ekler.
2. **Pop()** metodu Stack içine eklenen son değeri çıkarır.

Uygulama-2

Stack Uygulama

Uygulamada Stack sınıfından üretilen nesne ile bir TexBox, iki Button, bir ListBox kontrolü kullanarak yığına alma ve yığından çıkarma işlemleri yapılacaktır.

Adım 1: Görsel 4.26'daki form tasarımını yapınız ve form üzerindeki kontrollerin name değerlerini veriniz.



Görsel 4.26: Stack form tasarımı uygulaması-2

Adım 2: Stack sınıfından üretilecek nesneyi global olarak oluşturunuz.

```
Stack yigin = new Stack();
```

Adım 3: Ekle butonu Click olayında TextBox içinden alınan değerleri Stack içine aktarınız.

```
private void btnEkle_Click(object sender, EventArgs
e)
{
  yigin.Push(txtAdSoyad.Text);
  Listele();
}
```

Adım 4: Stack içindeki değerleri List-Box içinde göstermek için Listele adında bir metot oluşturarak listeleme işlemlerini yapınız.

```
private void Listele()
{
  listeStack.Items.Clear();
  foreach (string eleman in yigin)
  {
    listeStack.Items.Add(eleman);
  }
}
```

Adım 5: Çıkart butonu Click olayında Stack içinden son değeri çıkarınız. Listele metodunu kullanarak en güncel Stack değerlerini ListBox içinde gösteriniz.

```
private void btnCikart_Click(object sender, EventArgs e)
{
   yigin.Pop();
   Listele();
}
```

4.2.5. Dictionary Koleksiyonu

Dizi ve ArrayList içine eklenen elemanlar bellek üzerine sıralı olarak yerleşir. Dizi ile ArrayList elemanlarına 0'dan başlayarak index numarası verilir ve elemanlara erişim için bu index numaraları kullanılır. Dictionary koleksiyonunda index-değer ilişkisi Key (Anahtar) - Value (Değer) olarak kullanılmaktadır. Key dizilerdeki index numarası, Value ise index numarası ile belirtilen değer olarak düşünülebilir. Key benzersiz olmak şartıyla int, string, byte, object vb. olabilir.

Aşağıda Dictionary koleksiyonunun kullanımı verilmiştir.

```
Dictionary<KeyTipi,ValueTipi> koleksiyon_adı = new Dictionary<KeyTipi,ValueTipi>();
```

Aşağıdaki kodlamalarda Dictionary nesnesi tanımlaması ve değer aktarımında Key-Value tiplerinin farklı kullanımları gösterilmiştir.

```
Dictionary<int,string> sehirler =
new Dictionary<int,string>();
sehirler.Add(1,"Adana");
sehirler.Add(2,"Adıyaman");
```

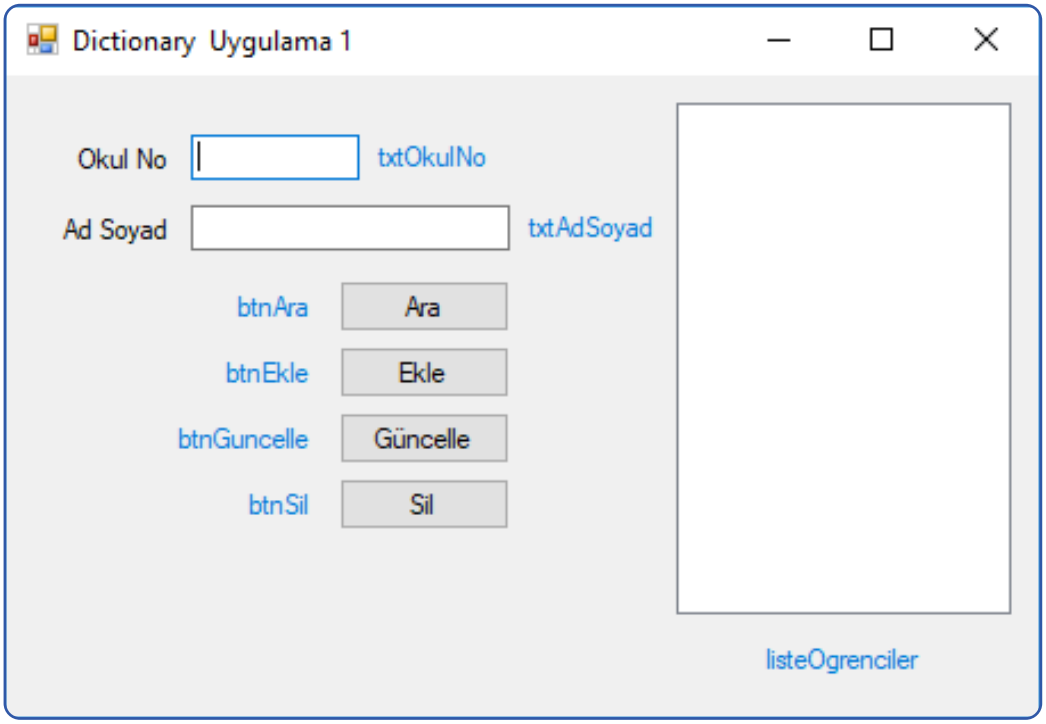
```
Dictionary<string, int> sehirler =
new Dictionary<string, int>();
sehirler.Add("Adana", 1);
sehirler.Add("Adıyaman", 2);
```

Uygulama-1

Dictionary Koleksiyonu

Bu uygulamada okul öğrencilerinin numaralarını ve isimlerini bir Dictionary koleksiyonuna ekleme, silme, güncelleme ve arama işlemleri gerçekleştirilecektir.

Adım 1: Görsel 4.27'deki form tasarımını yapınız ve form üzerindeki kontrollerin name değerlerini veriniz.



Görsel 4.27: Dictionary form tasarımı uygulaması-1

Adım 2: Bu uygulamada Dictionary nesnesinin anahtarını okul numarası ve integer tipinde, değerini ise öğrencilerin adı soyadı ve string tipinde tanımlayınız.

```
Dictionary<int,string> ogrenciler = new Dictionary<int, string>();
int anahtar;
string deger;
```

Adım 3: Ekle butonu Click olayında okul numaralarını Key, öğrencilerin adı soyadını Value olarak koleksiyona ekleyiniz.

```
private void btnEkle_Click(object sender, EventArgs
e)
{
  anahtar = int.Parse(txtOkulNo.Text);
  deger = txtAdSoyad.Text;
  ogrenciler.Add(anahtar,deger);
  Listele();
}
```

Adım 4: Dictionary koleksiyonundaki Key ve Value bilgilerini ListBox içinde göstermek için Listele isminde bir metot kullanınız. Bu metot, foreach döngüsü içinde ogrenciler koleksiyonundaki tüm Key ve Value bilgilerini var tipinde ismi ogrenci olan nesneye sırasıyla aktaracaktır. Bu metotla döngü her döndüğünde ogrenci nesnesini kullanarak ListBox içine Key ve Value bilgileri eklenecektir.

```
private void Listele()
{
 listeOgrenciler.Items.Clear();
 foreach (var ogrenci in ogrenciler)
 {
  listeOgrenciler.Items.Add(ogrenci.Key+"-"+ ogrenci.Value);
 }
}
```

Adım 5: Güncelle butonu Click olayında okul numarasına göre öğrenci isimlerini güncelleme işlemini gerçekleştiriniz.

```
private void btnGuncelle_Click(object sender, EventArgs e)
{
 anahtar = int.Parse(txtOkulNo.Text);
 deger = txtAdSoyad.Text;
 ogrenciler[anahtar] = deger;
 Listele();
}
```

Adım 6: Sil butonu Click olayında okul numarasına göre Dictionary koleksiyonundan öğrenci bilgilerini silme işlemini gerçekleştiriniz.

```
private void btnSil_Click(object sender, EventArgs e)
{
  anahtar = int.Parse(txtOkulNo.Text);
  ogrenciler.Remove(anahtar);
  Listele();
}
```

Adım 7: Ara butonu Click olayında okul numarasının veya öğrenci adının koleksiyon içinde bulunup bulunmadığını kontrol ediniz, okul numarası veya öğrenci adı varsa mesaj olarak gösterme işlemini yapınız.

```
private void btnAra_Click(object sender, EventArgs e)
{
  bool durum=false;
  if(txtOkulNo.Text!="")
  {
    anahtar = int.Parse(txtOkulNo.Text);
    durum = ogrenciler.ContainsKey(anahtar);
  }
  else
  {
    deger = txtAdSoyad.Text;
    durum = ogrenciler.ContainsValue(deger);
  }
  if(durum==true)
  {
    MessageBox.Show("Öğrenci Kayıtlıdır.");
  }
  else
  {
    MessageBox.Show("Öğrenci Kayıtlı Değildir.");
  }
}
```

Sıra Sizde

Dictionary Koleksiyonu Uygulama 1'de koleksiyon tanımlanırken öğrencinin adı soyadı Key olarak kullanılabilir mi? Açıklayınız.

4.2.6. Hashtable Koleksiyonu

Hashtable, non-generic bir koleksiyondur. Dictionary koleksiyonunda olduğu gibi Key-Value tipleri belirtilmez. Hashtable, karışık tablo anlamına gelmektedir. Bu koleksiyona yalnızca bir defa kullanılmak şartıyla Key değeri olarak istenilen herhangi bir değer atanabilir.

Hashtable koleksiyonunda kullanılan metot ve özellikler aşağıda verilmiştir.

1. Hashtable nesnesi oluşturma

```
Hashtable ogrenciler = new Hashtable();
```

2. Hashtable koleksiyonuna veri ekleme

```
ogrenciler.Add(368,"Ahmet");
ogrenciler.Add("Ahmet",368);
```

3. Hashtable koleksiyonundan veri silme

```
ogrenciler.Remove(368);
ogrenciler.Remove("Ahmet");
```

4. Hashtable koleksiyonunda veri güncelleme

```
ogrenciler[368] = "Mehmet";
```

5. Hashtable koleksiyonunda Key içeriğini listeleme

```
foreach (var anahtar in ogrenciler.Keys)
{
 Console.WriteLine(anahtar);
}
```

6. Hashtable koleksiyonunda Value içeriğini listeleme

```
foreach (var deger in ogrenciler.Values)
{
  Console.WriteLine(deger);
}
```

7. Hashtable koleksiyonu Key ve Value içeriğini listelemek için DictionaryEntry isimde bir sınıf kullanılarak koleksiyon içindeki elemanların hem Key hem de Value değerlerine erişim sağlanabilir.

```
foreach (DictionaryEntry eleman in ogrenciler)
{
  Console.WriteLine(eleman.Key + " - " + eleman.
Value);
}
```

Sıra Sizde

Dictionary Koleksiyonu Uygulama 1'de hazırladığınız formu Hashtable koleksiyonunu kullanarak yeniden yapınız.

4.2.7. SortedList Koleksiyonu

SortedList koleksiyonu, Hashtable koleksiyonuna benzer. SortedList koleksiyonunun en önemli özelliği, içindeki verileri anahtarın içeriğine göre sıralamasıdır. Karışık olarak verilen anahtar değerlerini bile kendi içinde artan şekilde bir sıralama yaparak koleksiyon içine ekler.

SortedList koleksiyonunda kullanılan metot ve özellikler aşağıda verilmiştir.

1. SortedList nesnesi oluşturma

```
SortedList ogrenciler = new SortedList();
```

2. SortedList koleksiyonuna veri ekleme

```
ogrenciler.Add(368, "Ahmet");
```

3. SortedList koleksiyonundan veri silme

```
ogrenciler.Remove(368);
```

4. SortedList koleksiyonunda veri güncelleme

```
ogrenciler[368] = "Mehmet";
```

5. SortedList koleksiyonu Key ve Value içeriğini listelemek için DictionaryEntry isimde bir sınıf kullanılarak koleksiyon içindeki elemanların hem Key hem de Value değerlerine erişim sağlanabilir.

```
foreach (DictionaryEntry eleman in ogrenciler)
{
  Console.WriteLine(eleman.Key+" - "+eleman.Value);
}
```

Sıra Sizde

Dictionary Koleksiyonu Uygulama 1'de hazırladığınız formu SortedList koleksiyonunu kullanarak yeniden yapınız.

ÖLÇME VE DEĞERLENDİRME - 4

A)

1. Aşağıdaki kod satırında integer tipinde numaralar isminde bir dizi oluşturmak için boşluklara doğru ifadeleri yazınız.

_____ _____=={10,20,30,40};

2. Aşağıdaki kod satırlarında string tipinde sehirler isminde 81 elemanlı bir dizi oluşturmak için boşluklara doğru ifadeleri yazınız.

_____ _____ = new _____

3. Aşağıda tanımlanan diziye göre x değişkenin değeri ne olur?

```
byte[ ]  sayilar  =   new byte[ ]{ 4,3,2,1};
byte x=sayilar[3];
```

4. int[ ] sayilar = {10, 32, 60, 100, 90, 5}; dizisinde sayilar.Length komutunun değeri aşağıdakilerden hangisidir?

A) 7 **B)** 6 **C)** Hata verir. **D)** 10 **E)** 5

5. Aşağıdakilerden hangisi 4 elemanlı integer tipinde tanımlanmış bir dizidir?

A) int[] dizi = new int[4]; **B)** int[4] dizi; **C)** int[] dizi = 4;
D) int dizi[4]; **E)** int[] dizi=new byte[4];

6. Dizilerle ilgili aşağıdaki ifadelerden hangisi doğrudur?

A) Dizi elemanları sadece integer tipinde olur.
B) Dizinin Rank özelliği dizideki eleman sayısını verir.
C) Dizinin Length özelliği dizinin boyutunu verir.
D) Sayısal dizilerde ön tanımlı değeri 0'dır.
E) Dizilere istenilen tipte değer verilebilir.

7. Aşağıdakilerden hangisi 4 satır ve 5 sütundan oluşan iki boyutlu dizi tanımlamasıdır?

A) int[4, 5] sayilar = new int[,];
B) int [4][5] sayilar = new int[];
C) int [,] sayilar = new int[4, 5];
D) int [,] sayilar = new int[5, 4];
E) int[5,4] sayilar = new int[];

8. Aşağıdakilerden hangisi 2 satır ve 3 sütunlu bir dizinin doğru tanımlanmış hâlidir?

A) int[,] sayi= new int[2, 3]{{1, 2},{10,11},{ 5, 6}};
B) int[,] sayi = new int[2, 3]{};
C) int[,] a = new int[1, 2];
D) int[,] a = new int[1, 2]{{7, 1, 9}, {2, 5, 6}};
E) int[,] sayi= new int[2, 3]{{7, 1, 9},{2, 5, 6}};

9.

```
int[] sayilar = { 1, 4, 8, 2 };

int toplam  = 0;

for ( int index=0; index < sayilar.Length; index++ )
{
 toplam = toplam + sayilar[index] ;
}
```

Yukarıda verilen kodlar çalıştırıldığında toplam değişkenin değeri aşağıdakilerden hangisidir?

A) 15 **B)** 13 **C)** 5 **D)** 1 4 8 2 **E)** 0

10. Aşağıdakilerden hangisi bir ArrayList tanımlamasıdır?

A) ArrayList[] liste=new ArrayList();
B) ArrayList liste=new ArrayList();
C) ArrayList liste=ArrayList();
D) ArrayList liste=new ArrayList[10];
E) ArrayList() liste=new ArrayList();

11. Aşağıdakilerden hangisi bir ArrayList içine veri ekleme metodudur?

A) liste.Add(“abc”) **B)** liste.Set(“abc”) **C)** liste.Size(“abc”)
D) liste.Count(“abc”) **E)** liste.Get(“abc”)

```
ArrayList liste = new ArrayList();

liste.add(“Ali”);

liste.add(“Elif”);

liste.add(“Can”);

liste.add(“Fatih”);
```

12. Yukarıda yer alan kodlarda koleksiyona eklenen değerlerden sonra aşağıdaki listeye benzemesi için hangi işlem yapılmıştır?

Ali

Elif

Mehmet

Can

Fatih

A) liste.Add(“Mehmet”) **B)** liste.Add(2,”Mehmet”) **C)** liste.Add(“Mehmet”,2)
D) liste.Insert(“Mehmet”,2); **E)** liste.Insert(2,”Mehmet”)

```
ArrayList liste = new ArrayList();
liste.add("Ali");
liste.add("Elif");
liste.add("Can");
liste.add("Fatih");
liste.Insert(0,"Mehmet");
```

13. Yukarıda yer alan kodlamalara göre liste[2] değeri aşağıdakilerden hangisidir?

A) Ali **B)** Can **C)** Mehmet

D) Elif **E)** Fatih

```
ArrayList liste = new ArrayList();
liste.add("Ali");
liste.add("Elif");
liste.add("Can");
liste.add("Fatih");
liste.add("Mehmet");
```

14. Yukarıda yer alan kodlamalara göre aşağıdaki kodlardan hangisi Can ismini Ayşe olarak değiştirir?

A) liste("Can","Ayşe"); **B)** liste[2]="Ayşe"; **C)** liste["Can"] = "Ayşe";

D) liste(2)="Ayşe"; **E)** liste.Set("Can","Ayşe");

15.

```
ArrayList liste = new ArrayList();
liste.add("Ali");
liste.add("Elif");
liste.add("Can");
liste.add("Fatih");
liste.add("Mehmet");
```

Yukarıda yer alan kodlamada koleksiyona eklenen değerlerden sonra aşağıdaki listeye benzemesi için hangi işlem yapılmıştır?

Ali
Elif
Fatih
Mehmet

A) liste.RemoveAt(2); **B)** liste.Remove(2,"Can"); **C)** liste.Remove("Can",2)

D) liste.Remove(2); **E)** liste.RemoveAt("Can");

16.

```
ArrayList liste = new ArrayList();
liste.add("Ali");
liste.add("Elif");
liste.add("Can");
liste.add("Fatih");
liste.add("Mehmet");

for (int index = 0; index < liste._____; index++)
{
  lbListe.Items.Add(liste[_____]);
}
```

Yukarıda yer alan kodlamalarda boşluklara gelecek ifadeler aşağıdakilerden hangisinde doğru olarak verilmiştir?

A) Length, index **B)** Count, index **C)** Size, liste **D)** Count, liste **E)** Size, index

17. Queue koleksiyonu için aşağıdaki ifadelerden hangisi doğrudur?

A) Her eklenen eleman koleksiyon tarafından otomatik olarak sıralanır.

B) İlk giren eleman son çıkar.

C) Pop() ve Push() metotları kullanılır.

D) Anahtar (Key)-Değer (Value) ikilisine sahiptir.

E) İlk giren eleman ilk çıkar.

18. Aşağıdakilerden hangisinde Dictionary nesnesinin tanımlaması doğru olarak yapılmıştır?

A) Dictionary[int,string] nesne = new Dictionary[int, string]();

B) Dictionary<int,string> nesne = new Dictionary(int, string);

C) Dictionary(int,string) nesne = new Dictionary<int, string>();

D) Dictionary<int,string> nesne = new Dictionary<int, string>();

E) Dictionary<int,string> nesne = new Dictionary<int, string>;

19. Hashtable koleksiyonu ile Dictionary koleksiyonu arasındaki fark aşağıdaki seçeneklerden hangisinde verilmiştir?

	Hashtable Koleksiyonu	**Dictionary Koleksiyonu**
A)	İlk giren eleman ilk çıkar.	Son giren eleman ilk çıkar.
B)	Sıralama yapar.	Sıralama yapmaz.
C)	Veri tipleri belirtilmez.	Veri tipleri belirtilir.
D)	Anahtar değeri sayı olmalıdır.	Anahtar değeri sayı olmayabilir.
E)	Aynı Key değeri verilebilir.	Aynı Key değeri verilemez.

5 ÖĞRENME BİRİMİ

http://kitap.eba.gov.tr/KodSor.php?KOD=1899

FORM UYGULAMALARI

NELER ÖĞRENECEKSİNİZ?

Bu öğrenme biriminde;

- Form sınıfı ve kontrol sınıfı kavramlarını açıklayacak,
- Form uygulamalarını yapacak,
- Masaüstü uygulamaları için menü oluşturacak,
- Veri doğrulama işlemlerini yapacak,
- Kontrollere veri bağlama işlemlerini öğreneceksiniz.

ANAHTAR KELİMELER

Windows form uygulamaları, form sınıfı, kontrol sınıfı, menüler, veri doğrulama, veri bağlama

HAZIRLIK ÇALIŞMALARI

1. Windows ortamında çalışan uygulamalardan hatırladıklarınızı arkadaşlarınızla paylaşınız.
2. Windows uygulamalarında bulunan nesneler neler olabilir?

5.1. Formlar

Herhangi bir etkileşimli uygulamada en az bir tane kullanıcı arabirimi bulunmaktadır. C# programlama dili ile geliştirilen Windows form uygulamalarında bu arabirimler için Form sınıfından üretilen nesneler kullanılmaktadır. Formlar; TextBox, Button, Label gibi kontrolleri üzerinde tutarak bir arabirim oluşturmak için kullanılır. Kod editörü ile Windows form uygulama projesi hazırlandığında Form1 ismindeki form otomatik olarak oluşturulmaktadır.

Tüm uygulamalar, belirlenen bir noktadan itibaren çalışmaya başlar. Windows form uygulamalarında başlangıç noktası, Solution Explorer penceresinde **Program.cs** dosyası içindeki **Main** isimli fonksiyon ile belirlenmektedir.

```
static void Main()
{
  Application.EnableVisualStyles();
  Application.SetCompatibleTextRenderingDefault(false);
  Application.Run(new Form1());
}
```

Program.cs dosyasında bulunan yukarıdaki kodlamalarda **Application.Run** metodu, uygulamanın hangi form ile başlayacağını belirtmektedir.
Windows form uygulamalarında kullanılan formlar ve kontroller, **Form** sınıfından oluşturulan nesnelerdir. Bu nesnelerin ait oldukları sınıfın özellikleri (properties), olayları (events) ve metotları (methods) uygulama içinde ihtiyaca göre kullanılmaktadır. Form ile kontrollerin bazı özellik, olay ve metotları aynı, bazıları ise farklıdır.

5.1.1. Form Sınıfı

Windows form uygulamalarında System.Windows.Forms isim uzayındaki Form sınıfından üretilen form nesneleri kullanılır. Projeye bir form eklendiğinde Form sınıfının özeliklerini miras alan yeni Form sınıfı, kod editörü tarafından oluşturulmaktadır.

```
public partial class Form1 : Form
```

Oluşturulan form sınıfının içinde sınıf ismiyle aynı olan **Yapıcı Metot** (Contructor) bulunmaktadır. Yapıcı Metot içinden **InitializeComponent** isminde bir metot çağrılmaktadır. InitializeComponent metodu form ile birlikte oluşturulan **Form_İsmi.Designer.cs** dosyası içinde bulunmaktadır. Bu metot, form tasarımı sırasında form içine kontroller eklendiğinde veya kontrollerin özellikleri değiştirildiğinde arka planda kodlar üzerinde güncelleme yapmaktadır. Form açıldığında form ve forma eklenen kontrollerin özellikleri ve olayları bu metot içindeki kodlara göre düzenlenmektedir. Örneğin; formun boyutları görsel olarak değiştirildiğinde, forma bir kontrol eklendiğinde veya özelliklere değer verildiğinde kod editörü tarafından InitializeComponent metodu otomatik olarak güncellenmektedir.

Aşağıda Tablo 5.1, Tablo 5.2 ve Tablo 5.3'te Form sınıfında kullanılan temel özellikler, metotlar ve olaylar verilmiştir.

Tablo 5.1: Form Sınıfının Temel Özellikleri

AcceptButton	Form aktif iken **enter** tuşuna basıldığında belirlenen bir butona tıklama işlemi yapar.
CancelButton	Form aktif iken **esc** tuşuna basıldığında belirlenen bir butona tıklama işlemi yapar.
ControlBox	Formun sağ üst köşesindeki kontrol butonlarını gösterir veya gizler.
FormBorderStyle	Formun kenarlık stilini ayarlar.
MinimizeBox	Formun sağ üst köşesindeki küçültme butonunu aktif veya pasif yapar.
MaximizeBox	Formun sağ üst köşesindeki büyültme butonunu aktif veya pasif yapar.
ShowInTaskbar	Görev çubuğunda formun görünüp görünmeyeceğini belirler.
StartPosition	Çalışma anında formun başlangıç konumunu belirler.
WindowState	Başlangıçta formun nasıl görüntüleneceğini ayarlar.

Tablo 5.2: Form Sınıfının Temel Metotları

Activate()	Form aktif hâle gelir.
Close()	Formu kapatır. Form, başlangıç formu ise uygulamayı sonlandırır.
CenterToScreen()	Formu ekranın merkezine yerleştirir.
Show()	Formu gösterir.
ShowDialog()	Formu diyalog kutusu olarak gösterir. Gösterilen form kapatılmadan diğer forma geçiş yapılamaz.
Hide()	Formu gizler.

Tablo 5.3: Form Sınıfının Temel Olayları

Load	Form açılırken gerçekleşir.
Click	Form üzerine tıklandığı zaman gerçekleşir.
FormClosing	Form kapanmadan hemen önce gerçekleşir.
FormClosed	Form kapandıktan sonra gerçekleşir.

Uygulama-1

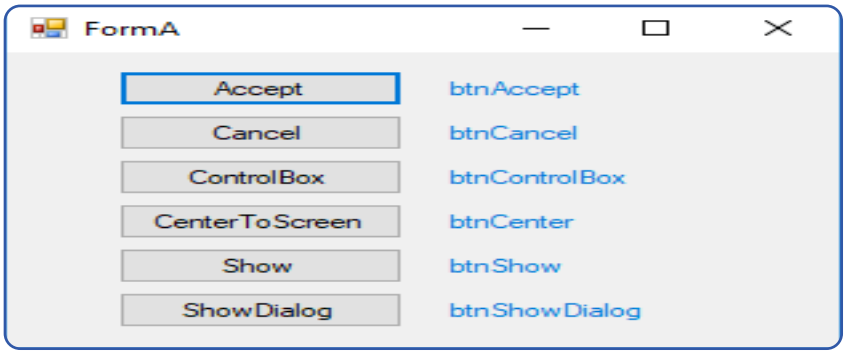
http://kitap.eba.gov.tr/KodSor.php?KOD=21081

Form Sınıfı

Bu uygulamada Form sınıfı için bazı olayların ve özelliklerin kullanılması gösterilecektir.

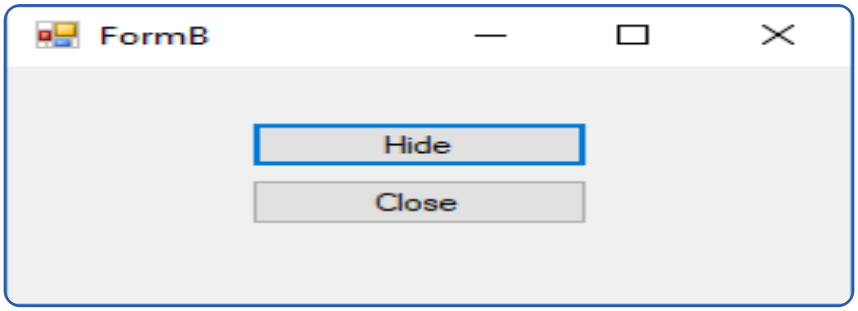
Adım 1: Yeni bir proje oluşturarak FormA ve FormB isminde iki form hazırlayınız.

Adım 2: FormA için Görsel 5.1'deki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.1: Form sınıfı uygulaması birinci form tasarımı

Adım 3: FormB için Görsel 5.2'deki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.2: Form sınıfı uygulaması ikinci form tasarımı

Adım 4: Birinci formdaki Accept ve Cancel butonları Click olaylarına aşağıdaki kodlamaları yapınız. Kodlamaları yaptıktan sonra Properties penceresinden formun AcceptButton ve CancelButton özelliğine ilgili butonları atayınız.

```
private void btnAccept_Click(object sender, EventArgs e)
{
  MessageBox.Show("Enter tuşu çalışıyor.");
}
private void btnCancel_Click(object sender, EventArgs e)
{
   MessageBox.Show("Esc tuşu çalışıyor.");
}
```

Adım 5: ControlBox butonu Click olayına formun sağ üst köşesindeki butonların gösterilmesi veya gizlenmesi işlemini gerçekleştiren kodlamayı yapınız.

```
private void btnControlBox_Click(object sender, EventArgs e)
{
  if (this.ControlBox == true)
    this.ControlBox = false;
  else
    this.ControlBox = true;
}
```

Adım 6: CenterToScreen butonu Click olayına formu ekranın merkezine yerleştirme işlemini gerçekleştiren kodlamayı yapınız.

```
private void btnCenter_Click(object sender, EventArgs e)
{
 this.CenterToScreen();
}
```

Adım 7: Show butonu Click olayına oluşturulan ikinci formun gösterilmesini sağlayan kodlamayı yapınız.

```
private void btnShow_Click(object sender, EventArgs e)
{
 FormB formB = new FormB();
 formB.Show();
}
```

Adım 8: ShowDialog butonu Click olayına oluşturulan ikinci formun diyalog penceresi olarak gösterilmesini sağlayan kodlamayı yapınız.

```
private void btnShowDialog_Click(object sender, EventArgs e)
{
 FormB formB = new FormB();
 formB.ShowDialog();
}
```

Adım 9: İkinci formun Load olayına form yüklendiğinde çalışan kodlamaları yapınız.

```
private void FormB_Load(object sender, EventArgs e)
{
  MessageBox.Show("FormB yükleniyor.");
}
```

Adım 10: İkinci formun FormClosing olayına form kapanırken çalışan kodlamaları yapınız.

```
private void FormB_FormClosing(object sender, FormClosingEventArgs e)
{
 MessageBox.Show("FormB kapanıyor.");
}
```

Adım 11: İkinci formun FormClosed olayına form kapandıktan sonra çalışan kodlamaları yapınız.

```
private void FormB_FormClosed(object sender, FormClosedEventArgs e)
{
  MessageBox.Show("FormB kapandı.");
}
```

Adım 12: İkinci form içindeki Hide butonu Click olayına formun gizlenmesini sağlayan kodlamaları yapınız.

```
private void btnHide_Click(object sender, EventArgs e)
{
  this.Hide();
}
```

Adım 13: İkinci form içindeki Close butonu Click olayına formun kapanmasını sağlayan kodlamaları yapınız.

```
private void btnClose_Click(object sender, EventArgs e)
{
  this.Close();
}
```

Adım 14: İkinci formun Click olayına form üzerinde herhangi bir yere fare ile tıklandığında çalışan kodlamaları yapınız.

```
private void FormB_Click(object sender, EventArgs e)
{
   MessageBox.Show("FormB click olayı çalışıyor.");
}
```

5.1.2. Kontrol Sınıfı

Kontrol sınıfı, System.Windows.Forms isim uzayında yer alan bir sınıftır. Form içine eklenen Button, TextBox, ListBox gibi görsel Windows kontrollerinin temel sınıfıdır. Aşağıda Tablo 5.4 ve Tablo 5.5'te kontrol sınıfında kullanılan temel özellikler ve olaylar verilmiştir.

Tablo 5.4: Kontrollerin Temel Özellikleri

BackColor **ForeColor** **Font** **BackgroundImage** **Cursor**	Kullanıcı arabirim görünüm ayarlarıdır.
Top **Left** **Bottom** **Right** **Bounds** **Height** **Width**	Kontrolün boyutlarını ve konumunu belirler.
Enabled **Visible**	Kontrolün durumunu belirler.
TabIndex **TabStop**	Kontrolün sekme sırasını belirler.
Opacity	Kontrolün şeffaflığını belirler (0.0 Şeffaf -1.0 Mat).
Text	Kontrolde metin gösterilmesini sağlar.

Tablo 5.5: Kontrollerin Temel Olayları

Click **DoubleClick** **MouseEnter** **MouseLeave** **MouseDown** **MouseUp** **MouseMove** **MouseHover** **MouseWheel**	Fare ile etkileşimi gerçekleştirir.
KeyPress **KeyUp** **KeyDown**	Klavye ile etkileşimi gerçekleştirir.
DragDrop **DragEnter** **DragLeave** **DragOver**	Sürükle bırak olaylarını gerçekleştirir.

Uygulama-2

Form Uygulamaları

Bazen form özelliklerine değer verilmesinde veya kontrollerin oluşturulmasında görsel kullanılarak değil, çalışma anında dinamik olarak kodlarla oluşturma ihtiyacı gerekebilir. Bu uygulamada birer adet TextBox, Label ve Button kontrolünü form içine görsel olarak Toolbox penceresinde eklemek yerine çalışma anında dinamik olarak ekleme işlemi yapılacaktır. Dinamik olarak eklenen kontrole özellik değerleri verme ve olay metotları oluşturma gibi işlemler gerçekleştirilecektir.

Adım 1: Yeni bir form uygulama projesi oluşturunuz. Projedeki formu kod görünümünde açınız.

Adım 2: Form içine eklenecek kontrolleri tanımlamak için Form sınıfının başlangıcına aşağıdaki kodlamaları yapınız.

```
public partial class Form1 : Form
{
  Label lblAdSoyad = new Label();
  TextBox txtAdSoyad = new TextBox();
  Button btnGiris = new Button();
```

Adım 3: Form Load olayına form özelliklerini değiştirmek için gereken kodlamaları yazınız.

```
private void Form1_Load(object sender, EventArgs e)
{
  this.Text = "Form Sınıfı Uygulama 2";
  this.BackColor = Color.Green;
  this.ForeColor = Color.Black;
  this.Size = new Size(300, 150);
}
```

Adım 4: Forma eklenecek Label kontrolünün özellikleri için Form Load olayına kodları yazınız.

```
lblAdSoyad.Text = "Adınız";
lblAdSoyad.Location = new Point(10, 10);
lblAdSoyad.Size = new Size(65, 15);
lblAdSoyad.ForeColor = Color.White;
```

Adım 5: Forma eklenecek TextBox kontrolünün özellikleri için Form Load olayına kodları yazınız.

```
txtAdSoyad.Location = new Point(75, 10);
txtAdSoyad.Size = new Size(150, 15);
txtAdSoyad.TabIndex = 1;
```

Adım 6: Forma eklenecek Button kontrolünün özellikleri için Form Load olayına kodları yazınız.

```
btnGiris.Text = "Tıkla";
btnGiris.Location = new Point(100, 60);
btnGiris.Size = new Size(100, 30);
btnGiris.ForeColor = Color.White;
btnGiris.BackColor = Color.Black;
btnGiris.TabIndex = 2;
```

Adım 7: Buton için tıklama olayı ve TextBox için klavyeden bir tuşa basma olayı metotları oluşturulacaktır. Bu işlem için kontrole atanan olaydan sonra += operatörü ile olay metotlarının isimlerini vererek Form Load olayı içine ekleyiniz (+= operatöründen sonra tab tuşuna basıldığında metot otomatik olarak oluşturulacaktır.).

```
txtAdSoyad.KeyPress += TxtAdSoyad_KeyPress;
btnGiris.Click += BtnGiris_Click;
```

Adım 8: Dinamik oluşturulan, özellikleri verilen ve olay metotları ataması yapılan kontrollerin forma eklenmesi için Form Load olayına kodlamaları yapınız.

```
this.Controls.Add(lblAdSoyad);
this.Controls.Add(txtAdSoyad);
this.Controls.Add(btnGiris);
```

Adım 9: TextBox için oluşturulan Keypress olayında klavyeden girilen her harfi büyük harfe çeviren kodlamaları yapınız.

```
private void TxtAdSoyad_KeyPress(object sender, KeyPressEventArgs e)
{
        e.KeyChar = Char.ToUpper(e.KeyChar);
}
```

Adım 10: Buton için oluşturulan Click olayında TextBox içine girilen bilgileri mesaj kutusunda gösteren kodlamaları yapınız.

```
private void BtnGiris_Click(object sender, EventArgs e)
{
  MessageBox.Show("Merhaba " + txtAdSoyad.Text);
}
```

Sıra Sizde

Kodlamalarda neden **this** ifadesi kullanılmıştır? Açıklayınız.

5.1.3. Konteyner Kontrolleri

Bu kontroller, diğer kontroller için özelleştirilebilir konteyner görevi gören özel kontrollerdir. Form uygulaması geliştirilirken eklenen kontrollerin fazlalığı, formun görüntüsünü karmaşık hâle getirebilmektedir. Formun karmaşıklığını önlemek ve birbirleriyle ilişkili kontrolleri bir arada tutmak için konteyner kontroller kullanılır. Bunlar; Toolbox penceresinde Containers sekmesi içinde bulunan Panel, Groupbox, TabControl, FlowLayoutPanel kontrolleridir. Örneğin; öğrenci ve veli bilgileri kaydedilen bir arabirim formu tasarlandığında öğrenci bilgilerine ait kontroller bir konteyner içinde, veli bilgilerine ait kontroller ise başka bir konteyner içinde tutulabilmektedir.

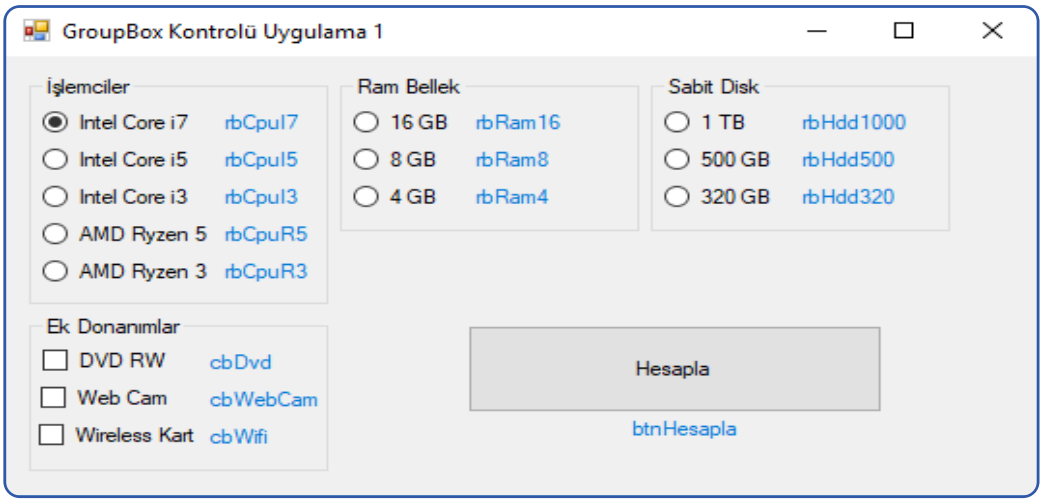
Konteyner kontrolünün belirli özellikleri değiştirildiğinde içinde bulunan kontrollerin özellikleri de bu durumdan etkilenmektedir. Konteyner kontrolün **Enable** veya **Visible** özellikleri değiştirildiğinde konteyner içindeki tüm kontroller aynı şekilde etkilenmektedir.

- **GroupBox Kontrolü:** Formu bölümlere ayırarak ilişkili kontrolleri bir arada tutmak için kullanılan bir konteyner kontrolüdür.

Uygulama-1

Bu uygulamada form üzerindeki ilişkili kontrolleri GroupBox içine alarak basit bir bilgisayar fiyat hesaplama uygulaması yapılacaktır.

Adım 1: Görsel 5.3'teki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.3: GroupBox uygulaması form tasarımı-1

Adım 2: Hesapla butonu Click olayına belirlenen taban fiyata seçilen her donanımın fiyatını ekleyen ve toplam fiyatı hesaplayıp mesaj olarak gösteren kodlamayı yapınız.

```
private void btnHesapla_Click(object sender, EventArgs e)
{
  decimal tabanFiyat = 500;
  // İşlemci fiyat hesaplaması
  decimal cpuFiyat = 0;
  if (rbCpuI7.Checked)
    cpuFiyat = 300;
  else if (rbCpuI5.Checked)
    cpuFiyat = 200;
  else if (rbCpuI3.Checked)
    cpuFiyat = 100;
  else if (rbCpuR5.Checked)
     cpuFiyat = 250;
  else if (rbCpuR3.Checked)
     cpuFiyat = 150;
  tabanFiyat += cpuFiyat;
  // Ram bellek fiyat hesaplaması
  decimal ramFiyat = 0;
  if (rbRam16.Checked)
    ramFiyat = 125;
  else if (rbRam8.Checked)
    ramFiyat = 75;
  else if (rbRam4.Checked)
     ramFiyat = 45;
  tabanFiyat += ramFiyat;
  MessageBox.Show(string.Format("Toplam Fiyat ={0:C}", tabanFi-
yat));
}
```

Sıra Sizde

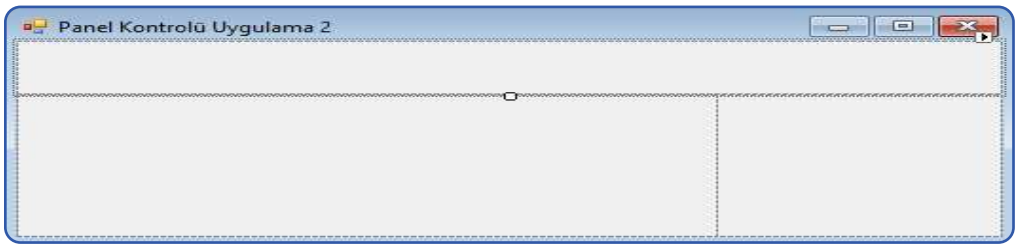
GroupBox Uygulama 1'deki diğer donanım özellikleri için fiyat hesaplamasını yapınız.

- **Panel Kontrolü:** Bu kontrol, GroupBox gibi diğer kontrolleri barındıran formun alt bölümüdür. GroupBox'tan farklı olarak bu kontrolde kaydırma çubukları bulunur. **AutoScroll** özelliği **true** yapıldığında sınırları dışındaki alanlar için yatay veya dikey kaydırma çubukları görünür hâle gelmektedir.

Uygulama-2

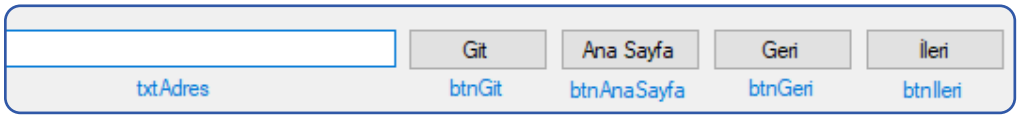
Bu uygulamada kod editörü içindeki Toolbox penceresinde bulunan WebBrowser ve Panel kontrollerini ekleyerek Internet Explorer'a ait özellikleri kullanan web tarayıcı uygulaması yapılacaktır.

Adım 1: Yeni bir Windows form uygulaması oluşturunuz. Projenizin formuna üç adet Panel kontrolü ekleyiniz. Panellerin Dock özelliğini Görsel 5.4'teki gibi Top, Fill ve Right olarak düzenleyiniz.



Görsel 5.4: Panel kontrollerinin form üzerinde yerleşimi

Adım 2: Üst panel için Görsel 5.5'teki tasarımı yapınız.



Görsel 5.5: Web tarayıcı gezinti bölümü panel tasarımı

Adım 3: Toolbox penceresinden WebBrowser kontrolünü ortadaki panel içine sürükleyerek bırakınız. Properties penceresinden ScriptErrorsSuppressed özelliğini true yapınız.

Adım 4: Toolbox penceresinden ListBox kontrolünü sağdaki panel içine sürükleyerek bırakınız. ListBox kontrolünün Dock özelliğini Fill yapınız.

Adım 5: Üst panelde bulunan Git ve Ana Sayfa butonlarının Click olaylarına aşağıdaki kodlamaları yapınız.

```
private void btnGit_Click(object sender, EventArgs e)
{
 webBrowser1.Navigate(txtAdres.Text);
}
private void btnAnaSayfa_Click(object sender, EventArgs e)
{
 webBrowser1.GoHome();
}
```

Adım 6: Gezilen sayfaların listesini ListBox kontrolü içinde göstermek için WebBrowser kontrolünün Navigated olayına aşağıdaki kodlamaları yapınız.

```
private void webBrowser1_Navigated(object sender, WebBrowserNavigatedEventArgs e)
{
 listBox1.Items.Add(webBrowser1.Url);
}
```

Sıra Sizde

1. Geri, İleri ve Yenile butonlarına tıklandığında WebBrowser kontrolünün ziyaret edilen siteler arasında gezinti yapmasını ve mevcut siteyi yenilemesini sağlayan kodlamaları yapınız.
2. Üstteki panel içine Geçmiş adında bir buton ekleyerek ListBox kontrolünün bulunduğu paneli gösterip gizleme işlemini gerçekleştiren kodlamaları yapınız.

- **TabControl Kontrolü:** Bu kontrol, gruplar hâlindeki kontrolleri sekme sayfalarında göstermek için kullanılır. TabControl sekmelerine tıklanarak sekme sayfaları arasında geçiş yapılabilmektedir.

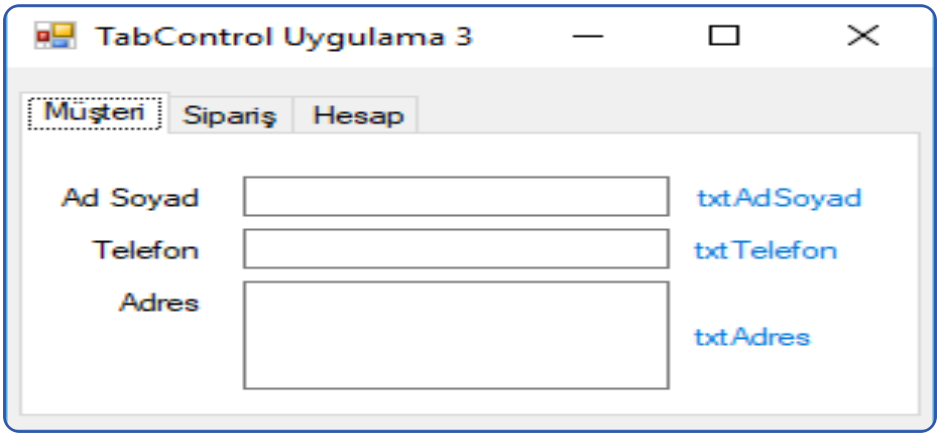
Uygulama-3

Bu uygulamada TabControl kullanılarak müşteri bilgileri girilecek ve verilen siparişlere göre hesap oluşturulacaktır.

Adım 1: Yeni bir form uygulama projesi oluşturunuz. Projenizin formuna TabControl ekleyiniz.

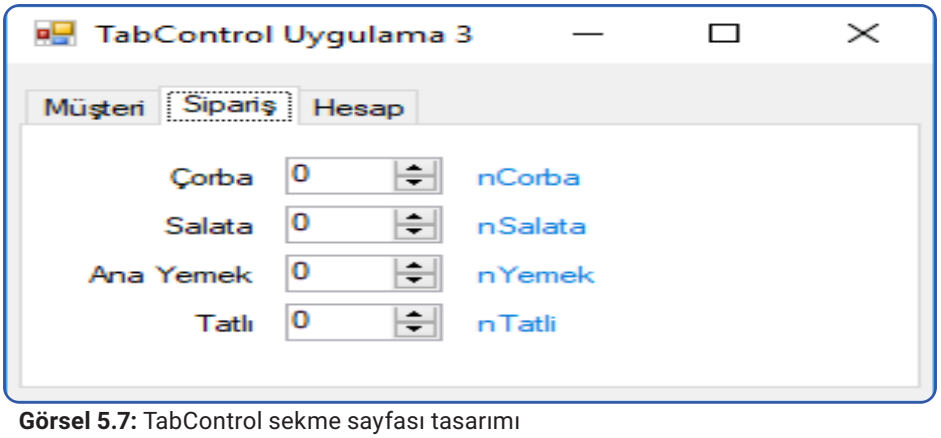
Adım 2: TabControl nesnesini seçtikten sonra Properties penceresinden TabPages özelliğine tıklayarak açılan pencereden üç adet sekme sayfası oluşturunuz. Oluşturulan sekme sayfalarının Text özelliğine sırasıyla Müşteri, Sipariş ve Hesap değerlerini atayınız.

Adım 3: İlk sekme sayfası için Görsel 5.6'daki tasarımı yapınız.



Görsel 5.6: TabControl sekme sayfası tasarımı

Adım 4: İkinci sekme sayfası için Görsel 5.7'deki tasarımı yapınız.



Görsel 5.7: TabControl sekme sayfası tasarımı

Adım 5: Üçüncü sekme sayfası için txtBilgi name değerine sahip bir TextBox ekleyerek Dock özelliğini Fill olarak değiştiriniz.

Adım 6: TabControl nesnesinin SelectedIndexChanged olayına üçüncü sekme sayfası açıldığında diğer sekme sayfalarındaki bilgileri kullanarak hesaplayan kodlamaları yapınız.

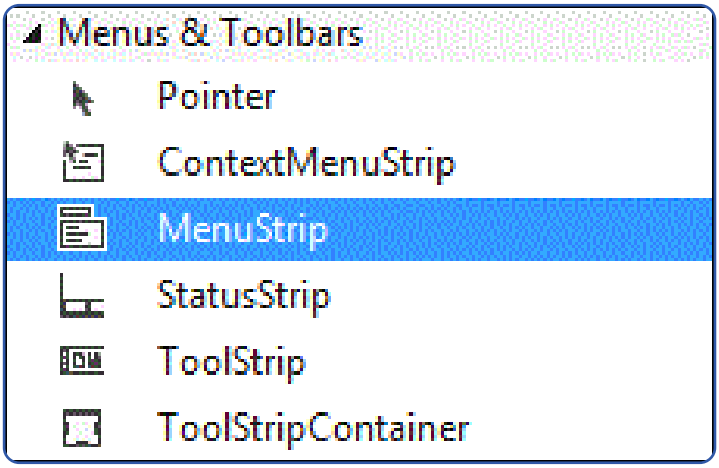
```
private void tabControl1_SelectedIndexChanged(object sender, EventArgs e)
{
  if(tabControl1.SelectedIndex==2)
  {
    txtBilgi.Text = "";
    txtBilgi.Text += txtAdSoyad.Text + "\r\n";
    txtBilgi.Text += txtTelefon.Text + "\r\n";
    txtBilgi.Text += txtAdres.Text + "\r\n";
    decimal hesap = 0;
    if (nCorba.Value > 0)
    {
      hesap += nCorba.Value*12;//Adet fiyatı 12 lira.
      txtBilgi.Text += string.Format("Çorba {0:C}",nCorba.Value * 12) + "\r\n";
    }
    txtBilgi.Text += "----------";
    txtBilgi.Text += string.Format("Toplam {0:C}",hesap);
  }
}
```

Sıra Sizde

Müşterilerin diğer siparişleri için hesaplama kodlamalarını yapınız.

5.2. Menüler

Menüler, bilgisayar uygulamalarında benzer veya ilgili komutları gruplandırmak için kullanılır. Menüler, uygulamanın daha kolay kullanılmasına olanak sağlar. Bu bölümde form içine menü ekleme, menü elemanlarına isim verme, menü elemanlarına tıklandığında ilgili kodları çalıştırma gibi işlemler açıklanacaktır. Bir Windows form uygulamasına menü eklemek için Görsel 5.8'de gösterilen Toolbox penceresinde Menus & Toolbars sekmesinde bulunan menü kontrolleri kullanılır.



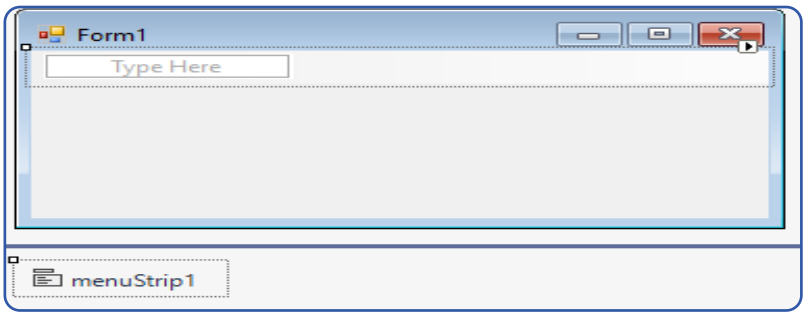
Görsel 5.8: Menüler ve araç çubukları

5.2.1. MenuStrip Kontrolü

MenuStrip kontrolü, menü araçlarının içinde en çok kullanılan kontroldür. Bu kontrol; birçok uygulamada karşılaşılan, genelde uygulamanın üst kısmında yer alan menüler ve alt menülerden oluşur. MenuStrip kontrolü, form tasarım görünümünde iken Toolbox penceresinden çift tıklanarak ya da form üzerine sürüklenerek forma eklenebilir.

MenuStrip kontrolü form içine eklendiğinde menülerin tasarlanacağı alan formun üst kısmında belirir. Bu alanda menüler görsel olarak tasarlanabileceği gibi Properties içinde Items özelliğine tıklanarak açılan "Items Collection Editor" penceresinden de tasarlanabilir.

Menüleri tasarım görünümünde oluşturmak için Görsel 5.9'da gösterilen "Type Here" yazan kısımlara bir kez tıklandığında menü içeriklerinin oluşturulmasını sağlayan kutular belirir. Bu kutulardan menü elemanı, açılır kutu, ayraç veya yazı eklenebilmektedir.



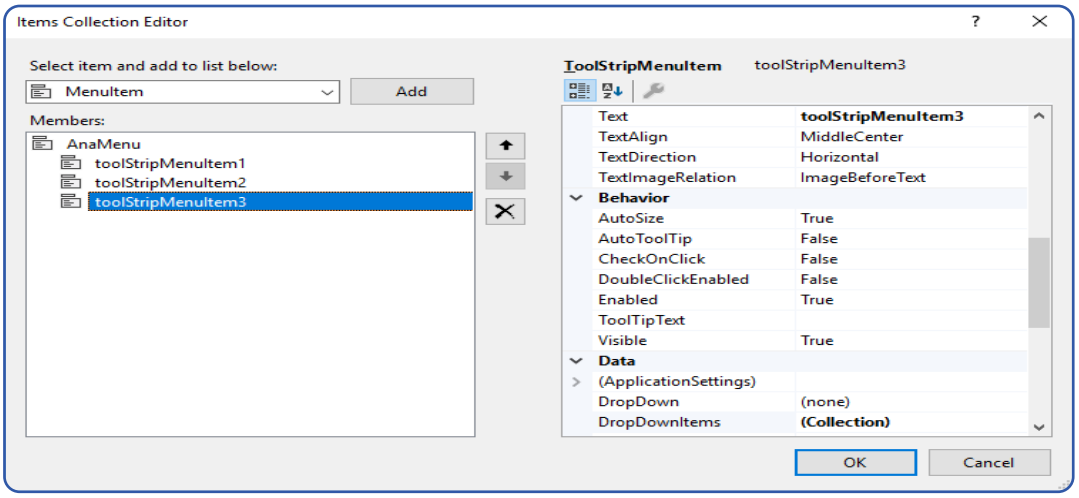
Görsel 5.9: MenuStrip tasarım alanı ve nesnesi

Uygulama-1

Bu uygulamada form içine MenuStrip kontrolü ekleyerek menü elemanlarını oluşturma işlemleri yapılacaktır.

Adım 1: Form üzerine Toolbox içinden MenuStrip kontrolü ekleyiniz. MenuStrip kontrolünün name değerini "AnaMenu" olarak değiştiriniz.

Adım 2: MenuStrip kontrolü Items özelliğine tıklayarak açılan "Items Collection Editor" penceresinden Görsel 5.10'da olduğu gibi üç tane MenuItem ekleyiniz. Eklenen menü elemanlarının Text özelliklerini sırasıyla Dosya, Düzen, Çıkış ve Name özelliklerini de menuDosya, menuDuzen, menuCikis olarak değiştiriniz.



Görsel 5.10: Items Collection Editor penceresi

Not: "Items Collection Editor" kullanılarak hazırlanan menü tasarımı, menü alanı içindeki "Type Here" kutuları seçilerek de yapılabilir.

Menü elemanlarına açılır menü ekleyerek benzer görevleri yerine getiren alt menüler oluşturulabilir. Örneğin; Dosya menüsünde dosya açma, kaydetme, yazıcıya gönderme gibi görevleri yerine getirecek işlemler için açılır menü kullanılır.

Menü elemanlarının altında açılır menü oluşturma, görsel olarak menü elemanlarının altında beliren kutularla yapılabileceği gibi menü elemanı seçildikten sonra Properties penceresi içinden DropDownItems özelliği kullanılarak da yapılabilir.

Uygulama-2

Açılır Menüler

Bu uygulamada menü elemanlarına tıklandığında benzer görevleri yerine getirecek olan açılır menü oluşturulacaktır. Oluşturulan açılır menülerin Text ve Name değerleri Görsel 5.11'de verilmiştir.

Dosya (menuDosya)	Düzen (menuDuzen)	Çıkış (menuCikis)
Yeni (menuYeni)	Kes (menuKes)	
Aç (menuAc)	Kopyala (menuKopya)	
Kaydet (menuKaydet)	Yapıştır (menuYapistir)	
Yazdır (menuYazdir)		

Görsel **5.11:** Açılır menüler

Adım 1: Dosya menü elemanı için açılır menüleri görsel tasarım alanındaki kutuları kullanarak oluşturunuz.

Adım 2: Düzen menü elemanı için açılır menüleri Properties içinde DropDownItems özelliğini kullanarak oluşturunuz.

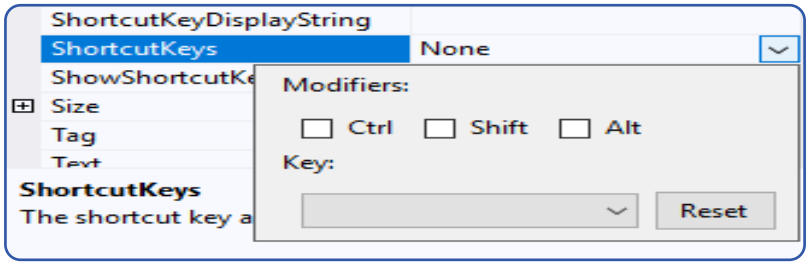
- **Menü Elemanlara Kısayol Tuşlarının Atanması**

Kısayol tuşları, menü elemanlarına tıklanmadan klavye üzerindeki tuş birleşimleri ile menü elemanlarına erişim sağlar. Yaygın olarak kullanılan tuş birleşimlerinden bazıları Tablo 5.6'da verilmiştir.

Tablo 5.6: Yaygın Kullanılan Tuş Birleşimleri

Ctrl+N	Yeni Dosya	Ctrl+C	Kopyala
Ctrl+O	Dosya Açma	Ctrl+X	Kes
Ctrl+S	Kaydetme	Ctrl+V	Yapıştır
Ctrl+P	Yazdır	Alt+F4	Programı Kapat

Menü elemanlarına Properties penceresinden ShortcutKeys özelliğiyle Görsel 5.12'de olduğu gibi kısayol ataması gerçekleştirilebilir.



Görsel 5.12: Menülere kısayol ataması

- **Pasif Menü Elemanları**

Bir menü elemanı soluk renkte pasif hâle Properties penceresinden Enable özelliği false yapılarak getirilir. Örneğin bir metin editörü uygulamasında yapıştırma işlemi, kopyalama veya kesme işlemleri yapılana kadar pasif hâldedir. Kopyalama veya kesme işlemi yapılırsa Yapıştır menü elemanı aktif hâle gelir.

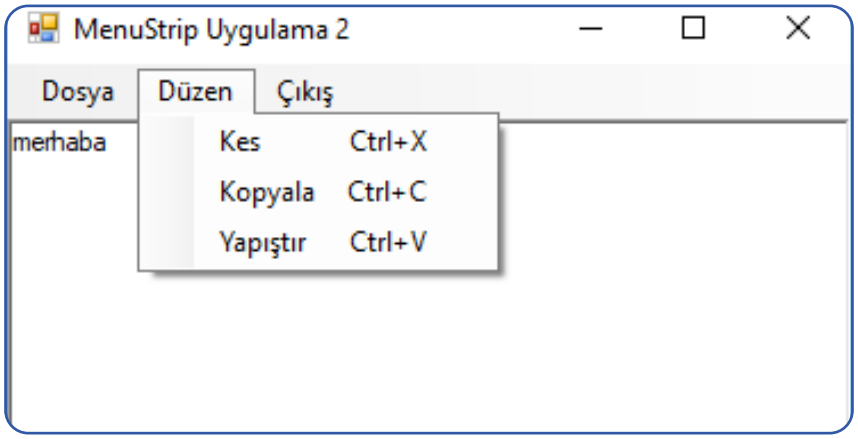
Uygulama-3

http://kitap.eba.gov.tr/KodSor.php?KOD=21082

Editör Uygulaması

Uygulama 2'deki menü tasarımı kullanılarak form içine eklenen RichTextBox kontrolü ile metinleri kesme, kopyalama ve yapıştırma işlemlerini gerçekleştiren bir editör uygulaması yapılacaktır.

Adım 1: Form içine Toolbox penceresinde RichTextBox kontrolü ekleyiniz. Görsel 5.13'teki gibi RichTextBox kontrolünün formun tamamını kaplaması için Properties penceresinde Dock özelliğini Fill olarak değiştiriniz.



Görsel 5.13: Editör uygulaması menü görünümü

Adım 2: Eklenen RichTextBox kontrolünün name değerini txtEditor olarak değiştiriniz. RichTextBox içinden seçilen metnin kesme, kopyalama ve yapıştırma işlemleri için menü elemanları Click olaylarına kodlamaları yapınız

```
private void menuKes_Click(object sender, EventArgs e)
{
  txtEditor.Cut();
}
private void menuKopyala_Click(object sender, EventArgs e)
{
  txtEditor.Copy();
}
private void menuYapistir_Click(object sender, EventArgs e)
{
  txtEditor.Paste();
}
```

Sıra Sizde

Yapıştır menü elemanını pasif hâle getiriniz. Kesme veya kopyalama işlemi gerçekleştirildiyse Yapıştır menü elemanını tekrar aktif hâle getirecek kodlamaları yapınız.

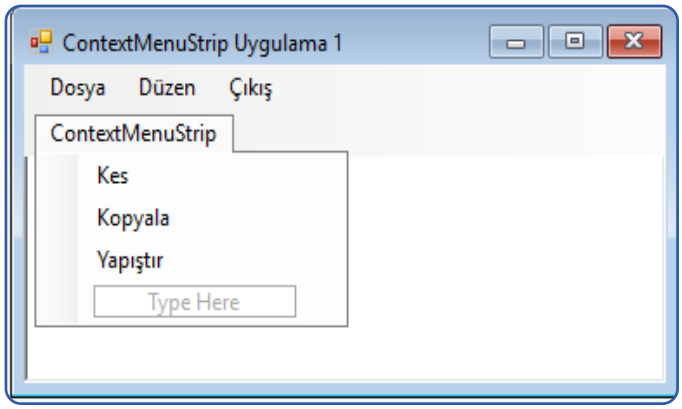
5.2.2. ContextMenuStrip Kontrolü

Birçok uygulamada işlemlerin daha hızlı gerçekleştirilebilmesi için sağ tıklama ile açılır menüler kullanılır. Bu menüleri oluşturmak için ContextMenuStrip kontrolü kullanılır. ContextMenuStrip tasarımı ve menü elemanlarının oluşturulması, MenuStrip kontrolüne benzer. Toolbox içinden Menus & Toolbars sekmesindeki ContextMenuStrip kontrolü çift tıklanarak veya sürüklenerek form içine eklenir. Menüler oluşturulduktan sonra bu menüler formla veya form içindeki herhangi bir kontrolle ilişkilendirilir. Uygulama çalıştırıldığında ilişkilendirilen kontrol üzerindeyken farenin sağ tuşuna basıldığında menü görünür hâle gelmektedir.

Uygulama-1

Editör uygulamasında oluşturulan form üzerine ContextMenuStrip kontrolü ekleyerek seçilen metin üzerinde sağ tıklandığında açılır menüleri gösterme işlemi yapılacaktır. Açılır menüden kesme, kopyalama ve yapıştırma işlemleri gerçekleştirilecektir.

Adım 1: Form içine Toolbox penceresinden Menus & Toolbars sekmesindeki ContextMenuStrip kontrolünü ekleyiniz. ContextMenuStrip kontrolünün name değerini cmDuzen olarak değiştiriniz. Açılır menü tasarımını Görsel 5.14'te olduğu gibi yapınız.



Görsel 5.14: ContextMenuStrip kontrolü menü tasarımı

Adım 2: ContextMenuStrip içindeki menü elemanları Click olayına Uygulama 3'teki (editör uygulaması) düzen menüsü için yapılan olayları seçiniz (Burada seçim işleminin yapılması kod tekrarını önlemek içindir.).

Adım 3: ContextMenuStrip menülerini RichTextBox nesnesi ile ilişkilendirmek için RichTextBox nesnesini seçtikten sonra Properties penceresinden ContextMenuStrip özelliğini tıklayarak açılır menüyü seçiniz.

5.3. İletişim Kutuları (Dialog Boxes)

Dialog Boxes, kullanıcı ile uygulama arasında iletişimi sağlayan pencerelerdir. İletişim kutuları genellikle kullanıcıya bir komutun nasıl işleneceğini ifade eder veya kullanıcıdan bir sorunun cevabı istenildiğinde kullanılır.

İletişim kutuları yeniden boyutlandırılamaz. İletişim kutuları; kullanıcıya bilgi, hata veya uyarı mesajı göndermede, dosya açma veya kaydetme esnasında dosya konumunu belirlemede, belgeyi yazıcıya gönderme esnasında yazdırma ayarlarında, editör uygulamalarında yazının fontunu veya rengini değiştirmede vb. açılan pencerelerdir.

En çok kullanılan iletişim kutuları şunlardır:

- MessageBox
- OpenDialog
- SaveDialog
- FontDialog
- ColorDialog
- PrintDialog

5.3.1. Mesaj İletişim Kutusu (MessageBox)

MessageBox kullanıcıya bir mesaj vermek amacıyla kullanılır. **Show()** metodu ile parantez içine yazılan mesajları gösterir. Ayrıca overload yöntemi ile farklı kullanım şekilleri de bulunmaktadır. Aşağıda MessageBox'ın temel kullanımı verilmiştir.

```
MessageBox.Show("Merhaba Dünya","Bilgi Mesajı");
```

İletişim kutusuna başlık eklemek için mesaj bilgisi verildikten sonra virgül bırakılarak başlık bilgisi yazılır.

```
MessageBox.Show("Merhaba Dünya");
```

İletişim kutusunda standart olarak Tamam butonu bulunur. Tamam butonu dışında buton eklemek istenildiğinde MessageBox butonları kullanılır. MessageBox butonları şunlardır:

- MessageBoxButtons.OK - Tamam
- MessageBoxButtons.AbortRetryIgnore - Durdur, Yeniden Dene, Yoksay
- MessageBoxButtons.OKCancel - Tamam, İptal
- MessageBoxButtons.RetryCancel - Yeniden Dene, İptal
- MessageBoxButtons.YesNo - Evet, Hayır
- MessageBoxButtons.YesNoCancel - Evet, Hayır, İptal

```
MessageBox.Show("Merhaba Dünya","Bilgi Mesajı",MessageBoxButtons.YesNoCancel);
```

İletişim kutusu içinde ikonlar kullanılabilir. Bu ikonlar, verilmek istenen mesajın içeriğiyle ilişkili olmalıdır. MessageBox ikonları şunlardır:

- **MessageBoxIcon.Error** – Hata ikonu
- **MessageBoxIcon.Warning** – Uyarı ikonu
- **MessageBoxIcon. Information** – Bilgilendirme ikonu
- **MessageBoxIcon.Question** – Soru ikonu

```
MessageBox.Show("Merhaba Dünya","Bilgi Mesajı",MessageBoxButtons.YesNoCancel,
MessageBoxIcon.Information);
```

Uygulama içinden kullanıcıya iletişim kutusu gösterildikten sonra kullanıcının hangi butona tıkladığının kontrolü için **DialogResult** kullanılır.

```
DialogResult cevap=MessageBox.Show("Bu dosyayı silmek istediğinize emin
misiniz?", "Dosya Sil",
MessageBoxButtons.YesNo,MessageBoxIcon.Warning);
if(cevap==DialogResult.Yes)
{
  MessageBox.Show("Dosya silindi.");
}
else if(cevap==DialogResult.No)
{
  MessageBox.Show("İşlem iptal edildi.");
}
```

Sıra Sizde

Bir önceki bölümde yaptığınız editör uygulamasında Çıkış menü elemanına tıklandığında MessageBox iletişim kutusu gösterilerek "Çıkmak istediğinize emin misiniz?" mesajı verilecektir. Evet butonuna tıklandığında uygulamayı kapatma işlemini gerçekleştiren kodlamayı yapınız.

5.3.2. Dosya Kaydet İletişim Kutusu (SaveFileDialog)

SaveFileDialog, kullanıcının bilgisayarındaki klasörlere göz atmasını ve dosyayı kaydedebileceği konumu belirlemesini sağlayan iletişim kutusudur. SaveFileDialog dosya kaydetme işlemi yapmaz, sadece kaydedilecek dosyanın konumunun ve isminin belirlenmesini sağlar. Dosya iletişim kutusu, tasarım veya çalışma anında oluşturularak kullanılabilir. Tasarım ekranında iken Toolbox penceresi Dialogs sekmesinden SaveFileDialog form içine sürüklenerek veya çift tıklanarak eklenebilmektedir. Çalışma anında ise SaveFileDialog sınıfından bir nesne oluşturularak iletişim kutusunun özellikleri kontrol edilebilmektedir.

```
SaveFileDialog sfd = new SaveFileDialog();
sfd.ShowDialog();
```

Filter özelliği, SaveFileDialog sınıfından oluşturulan nesne veya Toolbox penceresinde form içine eklenen diyalog, iletişim kutusu kaydedilecek dosyanın uzantısını veya uzantılarını belirlemek için kullanılır.

```
sfd.Filter ="Text Dosyası |*.txt";

// Kaydedilecek dosyanın uzantısı txt olarak belirlenir.
sfd.Filter = "Text Dosyası |*.txt| Word Dosyası |*.docx";

// Kaydedilecek dosyanın uzantısı txt  veya docx olarak belirlenir.
```

SaveFileDialog penceresinde kaydedilecek dosyanın ismi, uzantısı ve konumu belirlendikten sonra Kaydet butonuna tıklandığını kontrol etmek için DialogResult kullanılır.

```
SaveFileDialog sfd = new SaveFileDialog();
sfd.Filter = "Text Dosyası |*.txt| Tüm Dosyalar |*.*";
DialogResult cevap = sfd.ShowDialog();
if(cevap==DialogResult.OK)
{
 MessageBox.Show("Dosya kaydetme işlemi başlıyor.")
}
```

Uygulama-1

SaveFileDialog

Burada Editor uygulamasındaki RichTextBox kontrolünün içinde yer alan metinleri bir metin dosyası olarak kaydetme işlemi gerçekleştirilecektir.

Adım 1: Editör uygulamasında Dosya menüsündeki Kaydet menü elemanı Click olayına SaveFileDialog sınıfından oluşturulan nesne kullanılarak iletişim kutusu gösterilecektir. İletişim kutusundaki Kaydet butonuna tıklandığında RichTextBox içindeki bilgileri belirlenen konuma SaveFile() metodu ile kaydetme işlemini gerçekleştiren kodlamaları yapınız.

```
private void menuKaydet_Click(object sender, EventArgs e)
{
  SaveFileDialog sfd = new SaveFileDialog();
  sfd.Filter = "Text Dosyası |*.txt| Tüm Dosyalar |*.*";
  DialogResult cevap = sfd.ShowDialog();
  if(cevap==DialogResult.OK)
  {
    txtEditor.SaveFile(sfd.FileName,RichTextBoxStreamType.PlainText);
  }
}
```

5.3.3. Dosya Aç İletişim Kutusu (OpenFileDialog)

OpenFileDialog, kullanıcıların kendi bilgisayarına veya ağdaki herhangi bir bilgisayarın klasörlerine göz atmasına ve açmak için bir dosya seçmesine olanak sağlayan iletişim kutusudur. OpenFileDialog sınıfından üretilen nesneyi veya Toolbox penceresi Dialogs sekmesinden eklenen kontrolü kullanarak seçilen dosyanın adı ve yolu elde edilir.

```
OpenFileDialog ofd = new OpenFileDialog();
ofd.ShowDialog();
```

OpenFileDialog, Filter özelliği ile iletişim kutusu içinde gösterilecek dosyaların uzantılarına göre filtreleme yaparak sadece belirlenen uzantıda olan dosyaların gösterilmesini sağlar.

```
ofd.Filter = "Text Dosyaları|*.txt";
// Uzantısı txt olanları gösterir.
ofd.Filter = "Text Dosyaları|*.txt| Word Dosyaları|*.docx";
// Uzantısı txt veya docx olanları gösterir.
ofd.Filter = "Text Dosyaları |*.txt| Word Dosyaları |*.docx| Tüm Dosyalar |*.*";
// Uzantısı txt, docx veya tüm dosyaları gösterir.
```

OpenFileDialog penceresinden dosya seçildikten sonra Dosya Aç butonuna tıklandığını kontrol etmek için DialogResult kullanılır.

```
DialogResult cevap= ofd.ShowDialog();
if(cevap==DialogResult.OK)
{
  MessageBox.Show(ofd.FileName);
}
```

Uygulama-1

OpenFileDialog

Burada Editor uygulaması kullanılarak bir metin dosyasının içeriğini RichTextBox kontrolü içinde gösterme işlemi yapılacaktır.

Adım 1: Editör uygulamasında Dosya menüsündeki Dosya Aç menü elemanı Click olayına OpenFileDialog nesnesi kullanılarak txt uzantılı dosyaların seçilmesi sağlanacaktır. İletişim kutusundaki Dosya Aç butonuna tıklandığında seçilen dosyanın RichTextBox kontrolünün **LoadFile()** metodu ile dosya içeriğini gösteren kodlamaları yapınız.

```
private void menuAc_Click(object sender, EventArgs e)
{
  OpenFileDialog ofd = new OpenFileDialog();
  ofd.Filter = "Text Dosyaları |*.txt";
  DialogResult cevap= ofd.ShowDialog();
  if(cevap==DialogResult.OK)
  {
    txtEditor.LoadFile(ofd.FileName,RichTextBoxStreamType.PlainText);
  }
}
```

5.3.4. Yazdırma İletişim Kutusu (PrintDialog)

PrintDialog; yazıcı seçme, kâğıt boyutlarını ayarlama veya yazdırmayı başlatma gibi işlemlerin yapılmasını sağlayan iletişim kutusudur. PrintDialog sınıfından bir nesne oluşturularak bu nesne ile iletişim kutusunun özellikleri kontrol edilmektedir.

```
PrintDialog pd = new PrintDialog();

pd.ShowDialog();
```

Uygulama-1

PrintDialog

Burada Editor uygulaması kullanılarak Dosya menüsündeki Yazdır menü elemanına tıklandığında iletişim kutusu açılacaktır. İletişim kutusundaki Tamam butonuna tıklandığında RichTextBox kontrolü içindeki metinleri yazdırma işlemi gerçekleştirilecektir.

Adım 1: PrintDialog sınıfından oluşturulan nesne iletişim kutusunu göstererek yazıcı seçimi ve ayarların yapılması sağlanmaktadır. Yazdırma işlemi için PrintDocument nesnesi kullanılacaktır. Tasarım görünümünde iken ToolBox penceresinden PrintDocument kontrolünü form içine ekleyiniz. PrintDocument kontrolünün name değerini belge olarak değiştiriniz.

Adım 2: Dosya menüsündeki Yazdır menü elemanı Click olayında PrintDialog nesnesi kullanılarak RichTextBox içindeki yazıların yazıcıya gönderilmesi işlemini yapan kodlamaları yazınız.

```
private void menuYazdir_Click(object sender, EventArgs e)
{
   PrintDialog pd = new PrintDialog();
   DialogResult cevap = pd.ShowDialog();
   if (cevap == DialogResult.OK)
   {
      belge.Print();
   }
}
```

Adım 3: Yazdırma işlemini başlatmak için form içine eklenen PrintDocument kontrolü PrintPage olayına aşağıdaki kodlamaları yazınız.

```
private void belge_PrintPage(object sender, PrintPageEventArgs e)
{
 e.Graphics.DrawString(txtEditor.Text, txtEditor.Font,
 Brushes.Black,new Point(100,100));
}
```

5.3.5. Yazı Tipi İletişim Kutusu (FontDialog)

FontDialog; yazı tipinin, büyüklüğünün ve renginin seçilmesine olanak sağlayan iletişim kutusudur.

FontDialog sınıfından bir nesne oluşturularak bu nesne ile iletişim kutusunun özellikleri kontrol edilmektedir.

Uygulama-1

FontDialog

Editör uygulamasında FontDialog sınıfında üretilen bir nesne kullanılarak RichTextBox içinde seçilen yazının tipini ve büyüklüğünü değiştirme işlemi yapılacaktır.

Adım 1: Editör uygulaması menülerine Biçim adında menü ekleyiniz. Biçim menüsü içine de Yazı Tipi adında açılır menü elemanını ekleyiniz.

Adım 2: Yazı Tipi açılır menüsünün name değerini menuYaziTipi olarak veriniz. Yazı Tipi alt menüsü Click olayında FontDialog iletişim kutusunu göstererek belirlenen yazı tipini ve boyutunu RichTextBox kontrolüne uygulayan kodlamayı yapınız.

```
private void menuYaziTipi_Click(object sender, EventArgs
e)
{
 FontDialog fd = new FontDialog();
 if (fd.ShowDialog() == DialogResult.OK)
 {
   txtEditor.Font = fd.Font;
 }
}
```

Sıra Sizde

1. Uygulamada ShowDialog metodu neden karşılaştırma ifadesi içinde kullanıldı? Açıklayınız.
2. Uygulamada RichTextBox içindeki metnin sadece seçim yapılan kısımlarının biçimini değiştiren kodlamayı yapınız.

5.3.6. Renk İletişim Kutusu (ColorDialog)

ColorDialog, renk paleti içinden bir renk seçilmesini sağlayan iletişim kutusudur.

Uygulama-1

ColorDialog

Editör uygulamasında ColorDialog sınıfında üretilen bir nesne kullanılarak iletişim kutusu içinden seçilen bir rengin RichTextBox kontrolüne uygulanması işlemi yapılacaktır.

Adım 1: Editör uygulaması Biçim menüsü içine Renk adında açılır menü ekleyiniz. Renk menü elemanının name değerini menuRenk olarak veriniz.

Adım 2: Renk menü elemanı Click olayında RichTextBox içinden seçilen metnin renginin ColorDialog iletişim kutusu ile değiştirilmesini sağlayan kodlamayı yapınız.

```
private void menuRenk_Click(object sender, EventArgs e)
{
 ColorDialog cd = new ColorDialog();
 if (cd.ShowDialog() == DialogResult.OK)
 {
   txtEditor.SelectionColor = cd.Color;
 }
}
```

Sıra Sizde

ColorDialog iletişim kutusunun yaptığı görevi FontDialog iletişim kutusu da yapabilmektedir. FontDialog içinden renk seçme işlemi için gerekli düzenlemeleri yapınız.

5.4. Veri Doğrulama (Validation)

Veri girişi ile işlem yapılan uygulamalarda önemli noktalardan biri de verinin doğru girilmesidir. Doğru veri girilmemesi veya eksik bilgi girişi yapılması durumunda uygulamada hatalar, aksaklıklar ortaya çıkacak veya uygulama çalışamaz hâle gelecektir.

Yazılımcı, bir uygulamayı geliştirirken bu uygulamayı kullanacak kişilerin yapabileceği hataları öngörerek önlem almalıdır. Örneğin, öğrenci not girişleri yapılan uygulamada sınav notu 100 olan bir öğrenciye yanlışlıkla 1000 notu girilirse öğrencinin ortalaması çok yüksek çıkacaktır veya rakam yerine bir harf girilirse uygulama hata verecektir. Uygulama geliştirirken kullanıcıları bilgilendirerek ve hatalara karşı önlem alarak oluşabilecek problemlerin önüne geçilir.

5.4.1. İpucu (ToolTip)

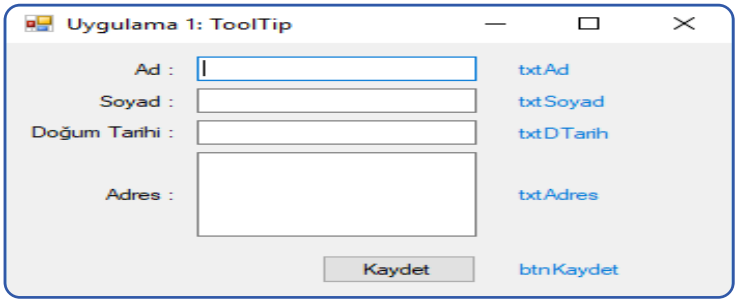
ToolTip, form içindeki nesnelere ait bilgilerin görüntülendiği açıklama pencereleridir. Form içindeki bir nesne üzerine fare ile gelindiğinde açıklama bilgisinin görüntülenmesini ToolTip sağlar.

Uygulama-1

ToolTip

Bu uygulama, **ToolTip** sınıfından üretilen nesne ile form içindeki kontrollerin üzerine gelindiğinde açıklama pencerelerinin gösterilmesidir.

Adım 1: Görsel 5.15'teki form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.



Görsel 5.15: ToolTip uygulaması form tasarımı-1

Adım 2: Formun Load olayında ToolTip sınıfından bir nesne oluşturarak form içindeki kontrollerde açıklama pencerelerini gösteren kodlamayı yapınız.

```
private void Form1_Load(object sender, EventArgs e)
{
 ToolTip tt = new ToolTip();
 tt.SetToolTip(txtAd, "Adınızı giriniz.");
 tt.SetToolTip(txtSoyad, "Soyadınızı giriniz.");
 tt.SetToolTip(txtDTarih, "Doğum tarihinizi gg/aa/yyyy şeklinde giriniz.");
 tt.SetToolTip(txtAdres, "Adresinizi giriniz.");
 tt.SetToolTip(btnKaydet, "Kaydetmek için tıklayınız.");
}
```

5.4.2. Veri Girişi Doğrulama (Input Validation)

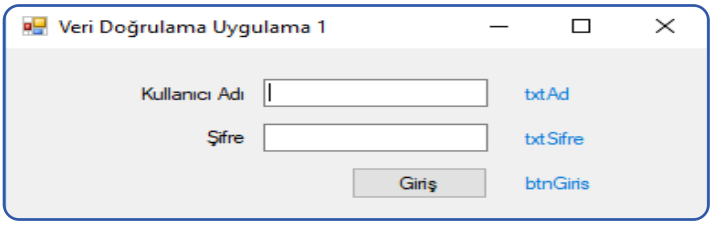
Veri doğrulama, uygulamanın her katmanında yapılabilmektedir. Yazılımcı tarafından belirlenen bazı ölçütlere göre kullanıcıların girdiği bilgilerin doğruluğu kontrol edilmelidir. Kullanıcının girdiği bilgiler ilk olarak form seviyesinden kontrol edilir. Form uygulamalarında veri girişleri en çok TextBox kontrolleri ile gerçekleşmektedir. Yazılımcı TextBox'a girilen verilerin doğruluğunu kullanıcının klavyeden girdiği her karakter esnasında sağlayabileceği gibi **Validating** olayı ile de kontrol edebilir. Girilen veriler uygun biçimde olmadığında uyarı mesajları verilir veya **ErrorProvider** nesnesi kullanılarak bilgilendirme işlemi gerçekleştirilir.

Uygulama-1

Veri Girişi Doğrulama

Burada form içinden veri girişi yapılan alanlara belirlenen ölçütlere göre doğru veri girişi yapılmadığı takdirde uyarı mesajı veren uygulama gerçekleştirilecektir.

Adım 1: Görsel 5.16'daki form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.



Görsel 5.16: Veri doğrulama uygulaması form tasarımı-1

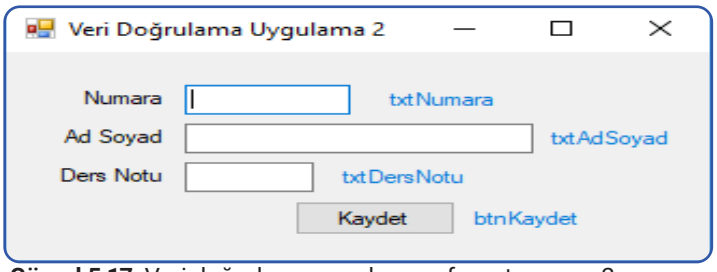
Adım 2: Giriş butonu Click olayında kullanıcı adı ve şifre verileri girilmez ise kullanıcıyı mesaj ile uyaran kodlamaları yapınız.

```
private void btnGiris_Click(object sender, EventArgs e)
{
 if (string.IsNullOrEmpty(txtAd.Text))
 {
    MessageBox.Show("Kullanıcı adını giriniz.", "Uyarı");
 }
 if(string.IsNullOrEmpty(txtSifre.Text))
 {
    MessageBox.Show("Şifreyi giriniz.", "Uyarı");
 }
}
```

Uygulama-2

Bu uygulamada uyarı mesajları MessageBox yerine ErrorProvider nesnesi kullanılarak verilecektir. Doğrulama kontrolü ise TextBox nesnelerinin Validating olayı içinde gerçekleştirilecektir.

Adım 1: Görsel 5.17'deki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.17: Veri doğrulama uygulaması form tasarımı-2

Adım 2: Uygulamada ErrorProvider sınıfından üretilen nesneyi global olarak oluşturunuz.

```
ErrorProvider ep = new ErrorProvider();
```

Adım 3: Form içindeki txtNumara isimli TextBox kontrolüne sayısal bir değer girilmelidir. TextBox içine girilen değerin sayısal olup olmadığını Validating olayında kontrol eden kodlamaları yapınız.

```
private void txtNumara_Validating(object sender, CancelEventArgs e)
{
  if(int.TryParse(txtNumara.Text,out int sonuc))
  {
     ep.SetError(txtNumara, "");
  }
  else
  {
     e.Cancel = true;
     ep.SetError(txtNumara, "Numara girişi hatalı");
  }
}
```

Adım 4: txtAdSoyad isimli TextBox kontrolünün Validating olayında TextBox içine değer girilmemiş ise ErrorProvider nesnesi ile kullanıcıyı uyaran doğrulama kodlarını yazınız.

```
private void txtAdSoyad_Validating(object sender, CancelEventArgs e)
{
  if (txtAdSoyad.Text == "")
  {
     e.Cancel = true;
     ep.SetError(txtAdSoyad, "Adı ve soyadı giriniz.");
  }
  else
  {
     ep.SetError(txtAdSoyad, "");
  }
}
```

Adım 5: txtDersNotu isimli TextBox kontolünün Validating olayında TextBox içine girilen değer sayısal değilse veya sayısal olup 0 ile 100 arasında değilse ErrorProvider nesnesi ile kullanıcıyı uyaran doğrulama kodlarını yazınız.

```
private void txtDersNotu_Validating(object sender, CancelEventArgs e)
{
  int dersNotu;
  if(int.TryParse(txtDersNotu.Text,out dersNotu))
 {
     if(dersNotu<0 || dersNotu>100)
     {
        e.Cancel = true;
        ep.SetError(txtDersNotu, "0-100 arasında değer giriniz.");
     }
     else
     {
        ep.SetError(txtDersNotu, "");
     }
  }
  else
  {
     e.Cancel = true;
     ep.SetError(txtDersNotu, "Sayısal değer giriniz.");
  }
}
```

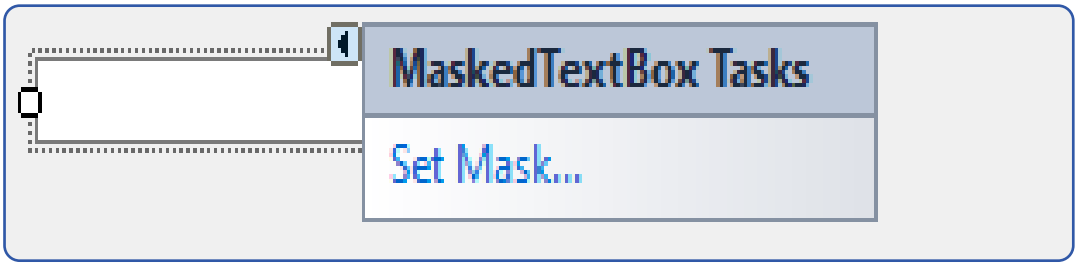
Sıra Sizde

Uygulamada e.Cancel = true ifadesi neden kullanılmıştır? Düşüncelerinizi sınıf arkadaşlarınızla paylaşınız

5.4.3. Veri Girişi Maskeleme (MaskedTextBox)

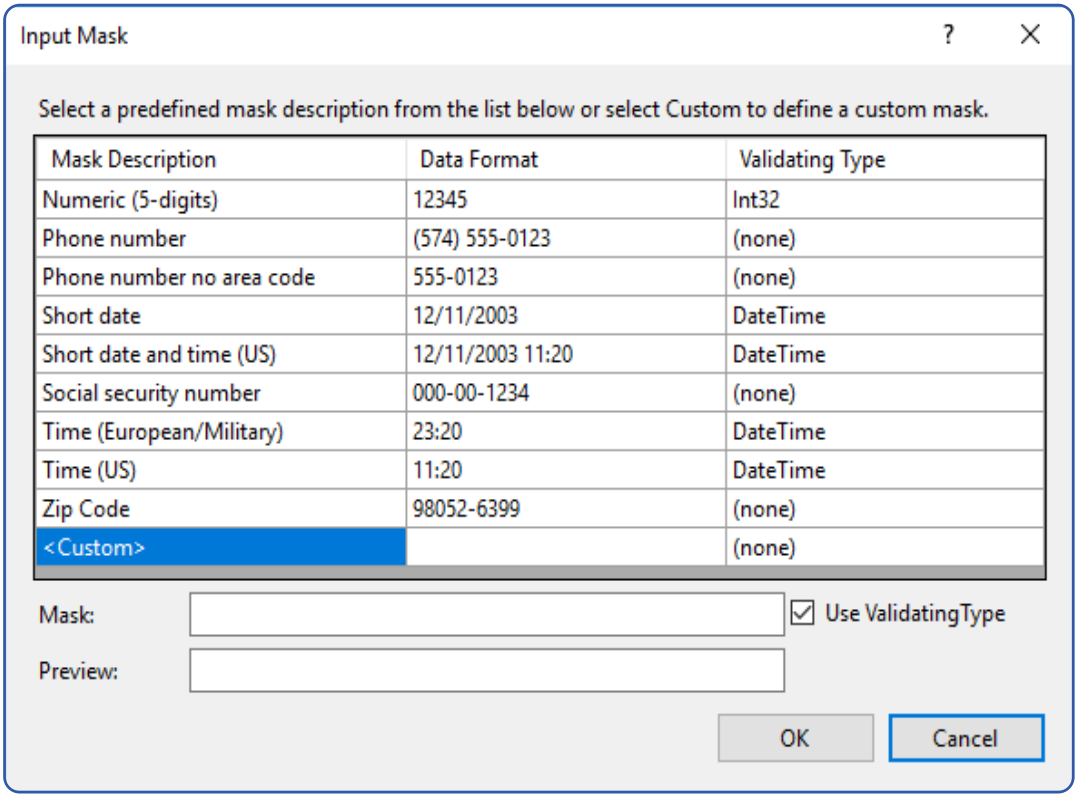
MaskedTextBox, belirlenen formatta veri girişinin yapılmasını sağlayan bir kontroldür. MaskedTextBox kontrolleri, form içinde kullanıcının gireceği verilerin doğrulanmasını sağlar. Örneğin, telefon numarası girişi yapılan bir alana uygun formatta veri girilmesi için MaskedTextBox kontrolü kullanılır.

Forma eklemek için tasarım ekranında iken Toolbox penceresinden MaskedTextBox kontrolü form içine sürüklenerek bırakılabilir. MaskedTextBox kontrolü eklendikten sonra giriş maskesi için Görsel 5.18'de gösterilen **Set Mask** linkine veya Properties penceresinden **Mask** özelliğine tıklanarak açılan diyalog kutusundan maske formatı belirlenebilir.



Görsel 5.18: Giriş maskesi linki

Görsel 5.19'da açılan diyalog kutusundan maske formatlarının listesi görülebilir ve istenilen bir format seçilerek giriş maskesi olarak kullanılabilir.



Görsel 5.19: Giriş maskesi diyalog penceresi

Bazı durumlarda Input Mask diyalog kutusundaki formatlar dışında özel formatlar belirlemek için Mask alanına maskeleme karakterleri girilerek özel maskeler oluşturulabilir. Bu karakterler şunlardır:

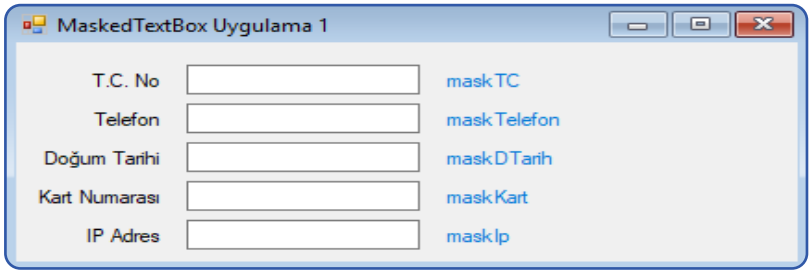
0	-	Rakam (Girilmesi zorunludur.)
9	-	Rakam veya boşluk (Girilmesi isteğe bağlıdır.)
#	-	Rakam veya boşluk (Girilmesi isteğe bağlıdır. Maskede bu konum boşsa özellik de bir boşluk olarak işlenir. Artı ve eksi işaretlerine izin verilir.)
L	-	Harf (Girilmesi zorunludur. A-Z veya a-z arasındaki harflere izin verir.)
?	-	Harf (Girilmesi isteğe bağlıdır. A-Z veya a-z arasındaki harflere izin verir.)
&	-	Karakter (Girilmesi zorunludur.)
C	-	Karakter (Girilmesi isteğe bağlıdır.)
A	-	Hem harf hem de rakam (Girilmesi zorunludur.)
a	-	Hem harf hem de rakam (Girilmesi isteğe bağlıdır.)
.	-	Ondalık ayracı
,	-	Binlik ayracı
:	-	Saat ayracı
/	-	Tarih ayracı
$	-	Para birimi işareti
>	-	Tüm karakterleri küçük harfe çevirir.
<	-	Tüm karakterleri büyük harfe çevirir.

Uygulama-1

MaskedTextBox

Bu uygulamada MaskedTextBox kontrollerine Form Load olayı içinde farklı maskeler verilecektir.

Adım 1: Görsel 5.20'deki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.20: Giriş maskeleme uygulaması-1

Adım 2: Form Load olayında ilgili MaskedTexBox kontrollerine aşağıdaki kodlamaları yaparak maske formatlarını veriniz.

```
private void Form7_Load(object sender, EventArgs e)
{
  maskTC.Mask = "00000000000";
  maskTelefon.Mask = "(999) 000 00 00";
  maskDTarih.Mask = "00/00/0000";
  maskKart.Mask = "0000-0000-0000-0000";
  maskIp.Mask = "###.###.###.###";
}
```

5.5. Veri Bağlama (Data Binding)

Veri bağlama, veri kaynağının form içindeki bir kontrole bağlanmasıdır. Veri bağlama işleminde form kontrolü ile kaynak arasında veri gönderme veya alma yapılabilir. Veri bağlama en çok veri tabanı içinde depolanan verilerle işlem yapmakta kullanılsa da kontrollerle, dizilerle veya koleksiyonlarla da veri bağlama işlemi yapılabilmektedir.

5.5.1. Basit Veri Bağlama (Simple Data Binding)

Basit veri bağlama, bir form kontrolüne tek veri bilgisinin bağlanmasıdır. Basit veri bağlama genellikle veri kümesi içindeki bir veriyi form içindeki kontrole aktarmak için kullanılır. Basit veri bağlama işlemlerinde Binding sınıfında üretilen nesne kullanılır. Bu nesne temelde üç parametre almaktadır. Bu parametreler şunlardır:

```
Binding binding=new Binding(string propertyName, object dataSource, string dataMember )
```

Binding sınıfındaki parametrelerin işlevleri aşağıda verilmiştir.

- propertyName: Bağlanacak olan nesnenin özelliğini belirtir.
- dataSource: Veri kaynağını belirtir.
- dataMember: Veri kaynağının hangi özelliğinin bağlanacağını belirtir.

Uygulama-1

Basit Veri Bağlama

Bu uygulamada form içindeki iki TextBox kontrolünden biri veri kaynağı olarak kullanılacak ve diğer TextBox kontrolüne veri bağlama işlemi gerçekleştirilecektir.

Adım 1: Görsel 5.21'deki form tasarımını yapınız ve kontrollere name değerlerini veriniz.

Görsel 5.21: Basit veri bağlama uygulaması-1

Adım 2: Formun Load olayına veri bağlama işlemleri için kullanılan kodlamaları yapınız

```
private void Form1_Load(object sender, EventArgs e)
{
 Binding bagla = new Binding("Text", txtKaynak, "Text");
 txtHedef.DataBindings.Add(bagla);
}
```

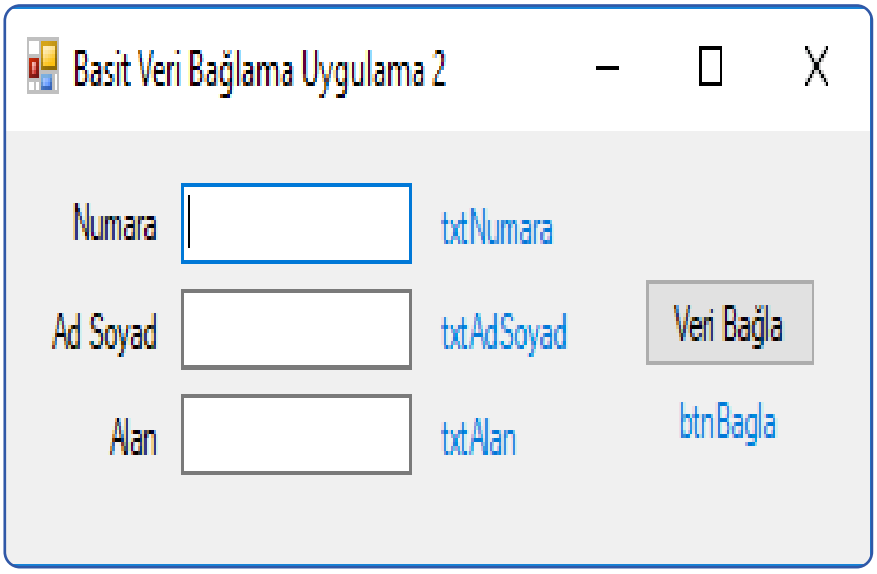
Uygulama-2

http://kitap.eba.gov.tr/KodSor.php?KOD=21083

Basit Veri Bağlama

Bu uygulamada bir sınıf oluşturulacaktır. Sınıftan üretilen nesne ile sınıf özeliklerine değer aktarılarak form içinde kontrollere veri bağlama işlemi gerçekleştirilecektir.

Adım 1: Görsel 5.22'deki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.22: Basit veri bağlama uygulaması-2

Adım 2: Veri kaynağı olarak kullanmak için aşağıda özellikleri verilen sınıfı oluşturunuz

```
class Ogrenciler
{
  public int Numara { get; set; }
  public string AdSoyad { get; set; }
  public string Alan { get; set; }
}
```

Adım 3: Veri Bağla butonu Click olayında sınıf özelliklerine değer aktarıldıktan sonra TextBox kontrollerine veri bağlama işlemini gerçekleştiren kodlamaları yapınız.

```
private void btnBagla_Click(object sender, EventArgs e)
{
Ogrenciler ogrenci = new Ogrenciler();
ogrenci.Numara = 1111;
ogrenci.AdSoyad = "Mehmet";
ogrenci.Alan = "Bilişim Teknolojileri";
txtNumara.DataBindings.Add("Text", ogrenci, "Numara");
txtAdSoyad.DataBindings.Add("Text", ogrenci, "AdSoyad");
txtAlan.DataBindings.Add("Text", ogrenci, "Alan");
}
```

5.5.2. Kompleks Veri Bağlama (Complex Data Binding)

Basit veri bağlama, bir kontrol nesnesine tek bir veri değerinin bağlanması işlemidir. Kompleks veri bağlama işlemi ise bir kontrol nesnesine birden çok veri değerinin bağlanması işlemidir. ListBox, ComboBox ve DataGridView kontrolleri, kompleks veri bağlanmasında en çok kullanılan kontrollerdir.

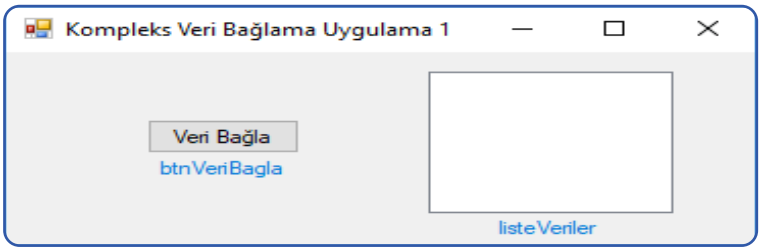
Kompleks bağlamada veri bağlanacak olan kontrolün **DataSource** özelliği ile veri kaynağı belirtilerek bağlama işlemi gerçekleştirilir.

Uygulama-1

Kompleks Veri Bağlama

Bu uygulamada dizideki verileri ListBox kontrolü içinde gösterme işlemi yapılacaktır. Diziler konusunda döngülerle gerçekleştirilen işlem, bu uygulamada veri bağlama yöntemi ile gerçekleştirilecektir.

Adım 1: Görsel 5.23'teki gibi form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.23: Kompleks veri bağlama uygulaması form tasarımı-1

Adım 2: Veri Bağla butonu Click olayında bir dizi oluşturarak dizinin değerlerini ListBox kontrolü içinde göstermek için veri bağlama işleminin kodlarını yazınız.

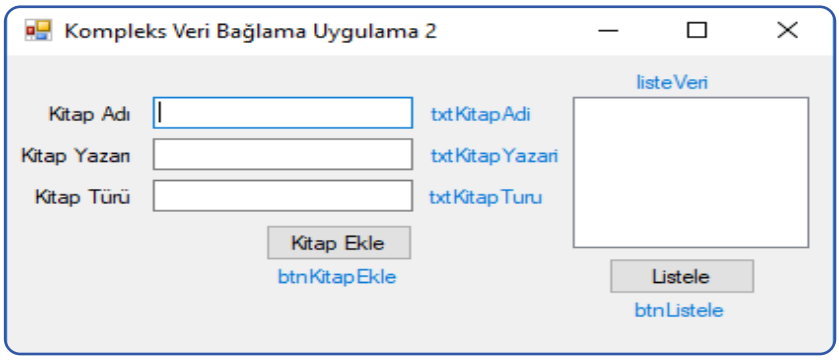
```
private void btnVeriBagla_Click(object sender, EventArgs e)
{
  string[] diller = { "C#", "Java", "Python","Delphi"};
  listeVeriler.DataSource = diller;
}
```

Uygulama-2

Kompleks Veri Bağlama

Bu uygulamada ArrayList koleksiyonu içine sınıf nesnesi eklenerek ListBox içinde gösterilecektir.

Adım 1: Görsel 5.24'teki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.24: Kompleks veri bağlama uygulaması form tasarımı-2

Adım 2: Kitaplar adında bir sınıf oluşturunuz ve bu sınıfın özelliklerini belirleyen kodlamaları yapınız.

```
class Kitaplar
{
  public string KitapAdi { get; set; }
  public string KitapYazari { get; set; }
  public string KitapTuru { get; set; }
}
```

Adım 3: Veri kaynağı olarak kullanılacak ArrayList koleksiyonunu global olarak oluşturunuz.

```
ArrayList kaynakVeri = new ArrayList();
```

Adım 4: Kitap Ekle butonu Click olayında Kitaplar sınıfından üretilen nesne kullanılarak sınıfın özelliklerine değer aktarımını ve nesneyi ArrayList koleksiyonuna ekleme işlemini yapınız.

```
private void btnEkle_Click(object sender, EventArgs e)
{
  Kitaplar kitap = new Kitaplar();
  kitap.KitapAdi =txtKitapAdi.Text;
  kitap.KitapYazari = txtKitapYazari.Text;
  kitap.KitapTuru = txtKitapTuru.Text;
  kaynakVeri.Add(kitap);
}
```

Adım 5: Listele butonu Click olayında ListBox kontrolüne ArrayList nesnesini bağlayınız. ListBox kontrolüne bağlanan veriler içinden KitapAdi özelliğinin gösterilmesini sağlayan kodlamaları yapınız.

```
private void btnListele_Click(object sender, EventArgs e)
{
  listeVeri.DataSource=null;
  listeVeri.DataSource = kaynakVeri;
  listeVeri.DisplayMember = "KitapAdi";
}
```

Sıra Sizde

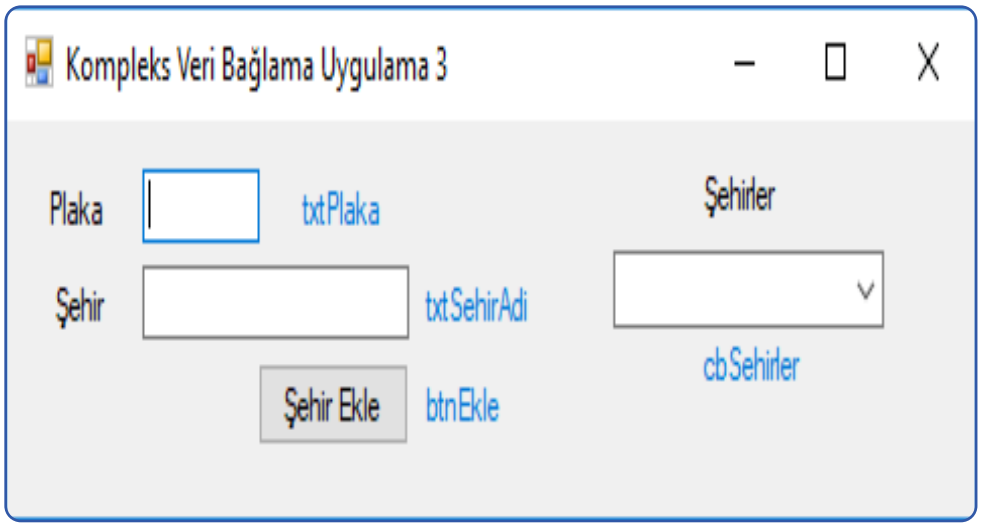
1. ListBox kontrolü içinde veri kaynağının diğer özelliklerinin gösterilmesini sağlayan kodlamaları yapınız.
2. Oluşturacağınız küçük gruplarla Listele butonu Click olayında DataSource özelliğine neden null değerinin aktarıldığını araştırınız. Elde ettiğiniz sonuçları diğer gruplarla paylaşınız.

Uygulama-3

Kompleks Veri Bağlama

Bu uygulamada ComboBox kontrolü ve List koleksiyonu kullanılarak veri bağlama işlemleri gerçekleştirilecektir. List koleksiyonuna plaka numarası ve şehir ismi özelliğine sahip sınıf nesnesi eklenerek ComboBox kontrolü içinde gösterme işlemi yapılacaktır.

Adım 1: Görsel 5.25'teki form tasarımını yapınız ve kontrollere name değerlerini veriniz.



Görsel 5.25: Kompleks veri bağlama uygulaması form tasarımı-3

Adım 2: List koleksiyonuna eklenecek nesneler için sınıf oluşturunuz ve bu sınıfın özelliklerini belirleyen kodlamaları yapınız.

```
class Sehirler
{
  public string Plaka { get; set; }
  public string SehirAdi { get; set; }
}
```

Adım 3: Kullanılacak List koleksiyonunu global olarak oluşturunuz.

```
List<Sehirler> listSehirler = new List<Sehirler>();
```

Adım 4: Şehir Ekle butonu Click olayında Sehirler sınıfından üretilen nesne kullanılarak sınıfın özelliklerine değer aktarımı yapıldıktan sonra List koleksiyonuna ekleme işleminin kodlarını yazınız.

```
private void btnEkle_Click(object sender, EventArgs e)
{
   Sehirler sehir = new Sehirler();
   sehir.Plaka = txtPlaka.Text;
   sehir.SehirAdi = txtSehirAdi.Text;
   listSehirler.Add(sehir);
   Bagla();
}
```

Adım 5: Bagla() ismindeki metot ile List koleksiyonunun ComboBox kontrolüne veri bağlama işlemini gerçekleştiriniz.

```
private void Bagla()
{
  cbSehirler.DataSource = null;
  cbSehirler.DataSource = listSehirler;
  cbSehirler.DisplayMember = "SehirAdi";
  cbSehirler.ValueMember = "Plaka";
}
```

Uygulama-4

Kompleks Veri Bağlama

Bu uygulamada DataGridView kontrolü ve List koleksiyonu kullanılarak veri bağlama işlemleri gerçekleştirilecektir. List koleksiyonuna öğrenci numarası, adı ve sınav notu özelliklerine sahip sınıf nesnesi eklenerek DataGridView kontrolü içinde gösterme işlemi yapılacaktır.

Adım 1: Görsel 5.26'daki form tasarımını yapınız ve form içindeki kontrollere name değerlerini veriniz.

Görsel 5.26: Kompleks veri bağlama uygulaması form tasarımı-4

Adım 2: Form kod görünümüne geçerek Ogreciler isminde bir sınıfın özelliklerini oluşturan kodlamayı yapınız.

```
class Ogrenciler
{
 public int Numara { get; set; }
 public string AdSoyad { get; set; }
 public int DersNotu { get; set; }
}
```

Adım 3: Ogrenciler sınıfından nesneleri saklayacak liste isminde bir List koleksiyonunu global olarak oluşturunuz.

```
List<Ogrenciler> liste = new List<Ogrenciler>();
```

Adım 4: Ekle butonu Click olayına Ogrenciler sınıfından üretilen nesne kullanılarak sınıfın özelliklerine değer aktarıldıktan sonra List koleksiyonuna ekleme işleminin kodlarını yazınız.

```
private void btnEkle_Click(object sender, EventArgs e)
{
  Ogrenciler ogrenci = new Ogrenciler();
  ogrenci.Numara = int.Parse(txtNumara.Text);
  ogrenci.AdSoyad = txtAdSoyad.Text;
  ogrenci.DersNotu =int.Parse(txtDersNotu.Text);
  liste.Add(ogrenci);
  Bagla();
}
```

Adım 5: Bagla() ismindeki metot ile List koleksiyonunu DataGridView kontrolüne veri bağlama işlemini gerçekleştiriniz.

```
private void Bagla()
{
  gridListe.DataSource = null;
  gridListe.DataSource = liste;
}
```

Adım 6: DataGridView kontrolü içindeki hücrelerin Double Click olayında çift tıklanan hücrenin bulunduğu satırdaki bilgileri TextBox kontrollerine aktaran kodlamayı yapınız.

```
private void gridListe_CellDoubleClick(object sender,DataGridViewCellE-
ventArgs e)
{
  txtNumara.Text = gridListe.CurrentRow.Cells[0].Value.ToString();
  txtAdSoyad.Text = gridListe.CurrentRow.Cells[1].Value.ToString();
  txtDersNotu.Text = gridListe.CurrentRow.Cells[2].Value.ToString();
}
```

Sıra Sizde

Form içine Sil isminde bir buton ekleyiniz. Sil butonu Click olayına DataGridView kontrolünün seçili satırındaki bilgileri silme işlemini gerçekleştiren kodlamayı yapınız

ÖLÇME VE DEĞERLENDİRME - 5

A) Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. **Form uygulamaları ile bilgisayar ortamında çalışan kullanıcı etkileşimli ..... geliştirilebilir.**
2. **Bir Windows form projesinde uygulamanın hangi formdan başlayacağı Program.cs dosyasındaki Main fonksiyonu içindeki ..... metodu ile belirlenir.**
3. **Windows form sınıfı ..... isim uzayı içinde bulunur.**

ControlBox– CenterToScreen – Load – AcceptButton – Show – FormClosed – CancelButton – Hide

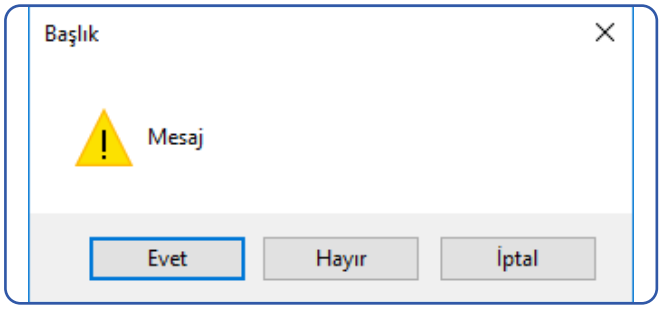
4. **Yukarıda form sınıfı için verilen özellik, metot ve olayların işlevlerini tablodaki boşluklara yazınız.**

	Formun ekranın ortasında açılmasını sağlar.
	Form kapandığında çalışan olaydır.
	Formun üzerindeki büyültme, küçültme ve kapatma butonlarını gösterir veya gizler.
	Form açılırken çalışan olaydır.
	Form aktifken klavyeden enter tuşuna basıldığında belirlenen bir butonun tıklanma olayı gerçekleşir.
	Formu göstermek için kullanılan metottur.

5. **Aşağıdakilerden hangisi Toolbox penceresi içindeki Containers sekmesinde <u>bulunmaz?</u>**
   A) GroupBox **B)** Panel **C)** TabControl
   D) SplitContainer **E)** WebBrowser

6. **Aşağıdakilerden hangisi MenuStrip içine eklenen nesnelerden biri <u>değildir?</u>**
   A) Label **B)** MenuItem **C)** ComboBox
   D) Separator **E)** TextBox

7. **Diyalog pencereleri ile ilgili aşağıda verilen bilgilerden hangisi <u>yanlıştır?</u>**
   A) OpenFileDialog, açılacak olan dosyanın konumunu belirler.
   B) SaveFileDialog, kaydedilecek dosyanın konumunu belirler.
   C) PrintDialog, yazdırma işlemini gerçekleştirir.
   D) ColorDialog, renk paletinden renk seçilmesini sağlar.
   E) FontDialog, yazı tipinin seçilmesini sağlar.



8. Görseldeki gibi bir MessageBox penceresi oluşturmak için aşağıda verilen kodlardan hangisi Show() metodu içinde kullanılmalıdır?

A) "Mesaj", "Başlık", MessageBoxButtons.YesNo, MessageBoxIcon.Error

B) "Mesaj", "Başlık", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Warning

C) "Başlık", "Mesaj", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Warning

D) "Başlık", "Mesaj", MessageBoxButtons.YesNo, MessageBoxIcon.Error

E) "Mesaj", " Başlık ", MessageBoxButtons.OKCancel, MessageBoxIcon. Warning

9. Aşağıda bilgileri verilen MaskedTextBox kontrolünde kullanılan maskeleme karakterleri için doğru sıralama aşağıdakilerden hangisidir?

I. Girilmesi zorunlu rakamlar için kullanılır.

II. Para birimi işareti için kullanılır.

III. Tüm karakterleri büyük harfe çevirir.

IV. Girilmesi zorunlu harfler için kullanılır.

A) #, $, < , L **B)** 9, $, < , A **C)** 0, $, < , a

D) 0, $, > , L **E)** 9, $, > , A

10.

I. Tek veri için basit veri bağlama kullanılır.

II. Birden çok veri için kompleks veri bağlama kullanılır.

III. Basit veri bağlama için Binding sınıfından üretilen nesne kullanılır.

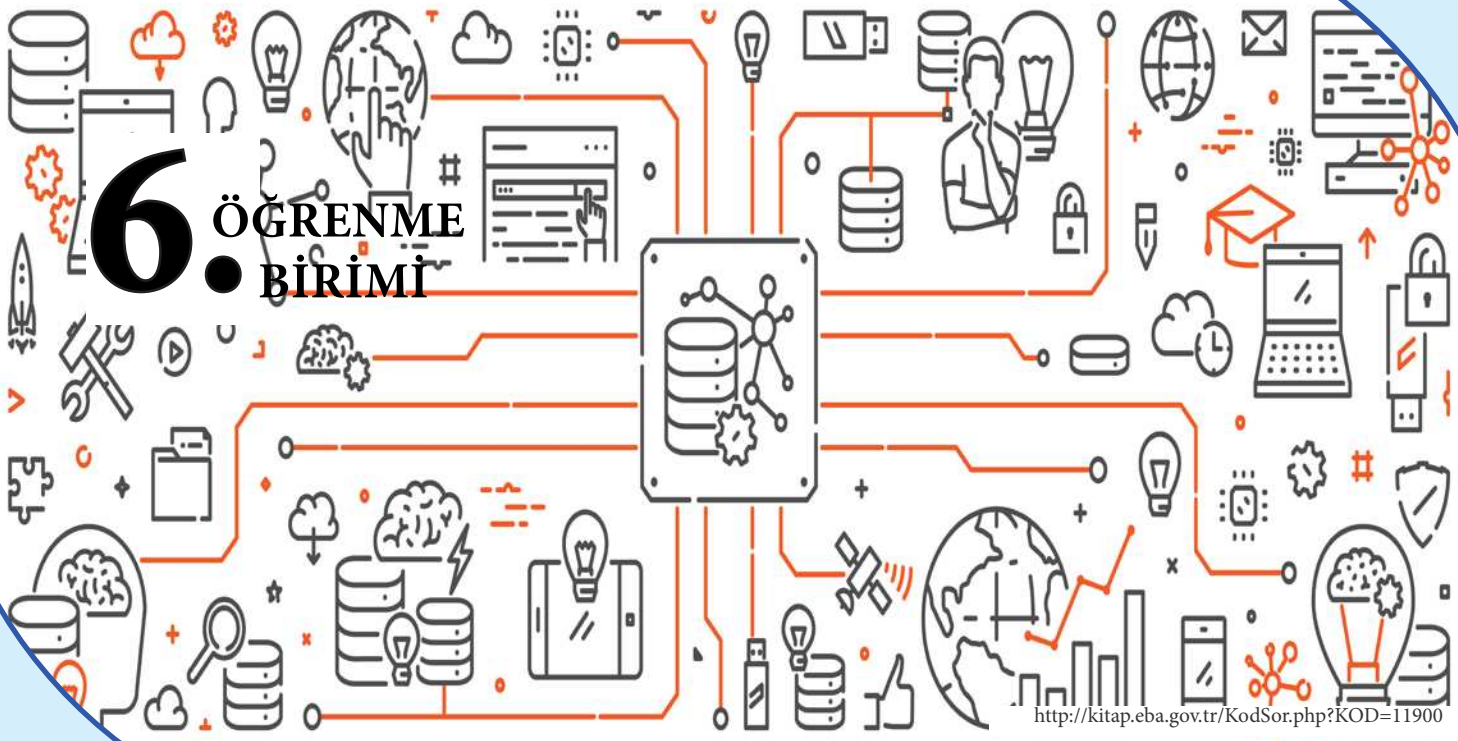
IV. ComboBox kontrolü kompleks veri bağlama için uygun değildir.

V. Kompleks veri bağlamada kontrolün DataSource özelliği kullanılır.

Yukarıdaki bilgilerden hangileri doğrudur?

A) I-II **B)** I-II-III **C)** I-II-V

D) I-II-III-IV **E)** I-II-III-V



6. ÖĞRENME BİRİMİ

http://kitap.eba.gov.tr/KodSor.php?KOD=11900

VERİ TABANI İŞLEMLERİ

NELER ÖĞRENECEKSİNİZ?

Bu öğrenme biriminde;

- Veri tabanının ne olduğunu ve ne işe yaradığını açıklayacak,
- Veri tabanı yönetim yazılımını kuracak,
- Yapılandırılmış sorgu (SQL) dilini öğrenecek,
- Veri tabanı ve tablolar oluşturacak,
- Veri türlerini öğrenecek,
- Tabloları ilişkilendirecek,
- SQL komutlarını kullanacak,
- Veri tabanı ile nesne tabanlı programlama arasında bağlantı kuracak,
- Nesne tabanlı programlama üzerinden SQL komutlarını kullanacak,
- Entity Framework yapısını öğrenecek,
- Projenize Setup hazırlayıp istediğiniz bilgisayara Setup kuracaksınız.

ANAHTAR KELİMELER

Veri tabanı, MySQL, Connection String, SQL komutları, Entity Framework, Setup hazırlama

HAZIRLIK ÇALIŞMALARI

1. Kullandığınız sosyal medya uygulamalarının nasıl çalıştığını düşünüyorsunuz? Fikirlerinizi arkadaşlarınızla paylaşınız.
2. Program, veri ve veri tabanı kavramları size neler ifade ediyor?

6. VERİ TABANI İŞLEMLERİ

Bu öğrenme birimi, adım adım gidilecek basit ve eğlenceli bir proje bölümüdür. Birkaç haftalık bir süre sonunda 350 satır civarı kod yazılacaktır. Bu süreçte sınıf gruplara ayrılarak çalışmalar yapılacaktır. Hem veri tabanı hem de proje dosyalarını taşımak için bir flash disk bulundurulmalıdır.

Görsel 6.1: Veri tabanı işlemleri

6.1. Veri Tabanı Yazılımının Kurulumu

En basit anlamda veri tabanı, belirli bir amaca yönelik düzenlenen kayıt ve dosyaların tümüdür. Bilgisayar sisteminde arkadaş ya da müşterilerin ad ve adresleri toplanabilir. Yazılan tüm mektuplar bilgisayar sisteminde bulundurulabilir ve alıcıya göre mektuplara düzen verilebilir. Ödeme ve tahsilat yapılacak hesaplar ya da çek defterinin borçları ve bakiyesi gibi finansal verilerin toplandığı bir grup dosya da olabilir. Konularına göre düzenlenen sözcük işlemci belgeler, en geniş anlamda bir tür veri tabanıdır. Kullanıma göre düzenlenen elektronik tablo dosyaları da ayrı bir tür veri tabanıdır. Başlat menüsündeki tüm program kısayolları da bir veri tabanıdır. Sık Kullanılanlar klasöründe düzenlenen internet kısayolları da bir veri tabanıdır Görsel 6.1).

Çok düzenli kişiler birkaç yüz elektronik tablo ya da kısayolu klasör ve alt klasörleri kullanarak yönetebilir. Bunu yaptıklarında veri tabanı yöneticisi olurlar. Çözülmeye çalışılan sorunlar büyüdükçe ne yapılabilir? Veriler, çeşitli belgelerde ya da elektronik tablolarda saklanmışken tüm müşterilerle ve onların siparişleriyle ilgili bilgiler nasıl kolayca toplanabilir? Yeni bilgiler girildiğinde dosyalar arasındaki bağlantılar nasıl korunabilir? Verilerin doğru girildiğinden nasıl emin olunabilir? Bilgileri birçok kişiyle paylaşma gereksinimi varsa aynı anda iki kişinin aynı verileri güncelleştirmesi istenilmediğinde ne yapılabilir? Bu tür güçlüklerle karşılaşıldığında bir veri tabanı yönetim sistemine (DBMS-Database Management System) gereksinim vardır.

6.1.1. Veri Tabanı Yönetim Sistemi (Database Management System)

Veri tabanları genellikle veri tabanı yönetim sistemi (DBMS) olarak bilinen kapsamlı bir veri tabanı yazılım programını gerektirir. DBMS, veri tabanı ve son kullanıcılar ya da programlar arasında bir arayüz işlevi görür. Kullanıcıların bilgilerin nasıl organize ve optimize edildiğini yönetmesine, bilgileri almasına ve güncellemesine olanak sağlar. DBMS aynı zamanda veri tabanlarına ilişkin gözetim ve kontrol faaliyetlerini kolaylaştırarak performans izleme, ince ayar yapma, yedekleme ve kurtarma gibi çeşitli yönetim operasyonlarının gerçekleştirilmesini sağlar.

6.1.2. SQL (Structured Query Language)

Yapılandırılmış sorgu dili (SQL-Structured Query Language); veriyi sorgulamak, manipüle etmek, tanımlamak ve veriye erişim kontrolü sağlamak üzere tüm ilişkisel veri tabanlarında kullanılan bir programlama dilidir. SQL ilk olarak 1970'li yıllarda geliştirilmiş, ardından SQL ANSI standardı uygulanmış ve yazılım şirketleri pek çok uzantının temelini atmıştır. SQL günümüzde oldukça yaygın bir şekilde kullanılsa da yeni programlama dilleri de geliştirilmektedir.

6.1.3. Büyük Veri (Big Data) ve Veri Madenciliği

Veri günümüzde paradan daha değerli bir hâle gelmiştir. Bunun sebebi, büyük firmaların stratejilerini bu verilere göre belirlemesidir. Bir arama yapıldığında o aramayla ilgili reklamların gelmesi bu duruma bir örnektir. Dünyanın en büyük firmalarının interneti kullanan milyarlarca kişi için bu işlemi yaptığı düşünülürse verilerin ne kadar önemli olduğu daha iyi anlaşılacaktır.

6.1.3.1. Big Data

Günümüzde adı sıkça duyulan Big Data (Büyük Veri), her ne kadar teknolojinin ilerlemesi ve kullanım alanlarının artması ile ortaya çıkmış bir kelime olarak görülse de yıllardır içinde bulunulan fakat gelişiminden pek haberdar olunmayan bir olgudur. Bu olguya farkında olmadan sürekli destek verilmekte ve "Büyük Veri" olarak isimlendirilen bu ortama sürekli veri akışı sağlanmaktadır.

Büyük Veri, yükselen teknolojiler ve tüketicilerin artan veri kullanım oranı paralelinde daha fazla çeşitlilik içeren veri kümesidir.

Teknolojinin ilerlemesi ve internetin gelişmesi ile birlikte günümüzde bilginin gücü de ön plana çıktı ve bununla beraber internet dünyasındaki birçok olgu "Bilgi Çöplüğü" olarak anılmaya başladı. Bu çöp-

lükten anlamlı verilerin de çıkabileceğini düşünen yazılım şirketleri, AR-GE çalışmalarını bu bağlamda yürüterek Big Data olarak adlandırılan olguyu ortaya çıkarttı.

Big Data diye isimlendirilen bu olgu, diskte çok fazla yer kaplayan veri çağrışımı yapsa da aslında tam olarak böyle değildir. Big Data; sosyal medya paylaşımları, fotoğraf arşivleri, sürekli kayıt alınan "log" dosyaları gibi farklı kaynaklardan elde edilen tüm verilerin anlamlı ve işlenebilir hâle dönüştürülmüş biçimidir. Big Data kullanım alanları ile ilgili örnekler şöyle sıralanabilir:

- Firmaların müşterileri, malzeme tedarikçileri, şirket içindeki her türlü işlem ve ürünleri ile ilgili tril-yonlarca bayt'lık veri toplanmakta ve anlamlı raporlar üretilmektedir.
- Sosyal medya paylaşımları sayesinde her gün milyarlarca kilobayt veri elde edilmektedir.
- Normal bir tüketicinin günlük yaşamında internette yaptığı haberleşme, arama, satın alma ve paylaşma türü işlemlerin yarattığı verilerin tümü saklanmaktadır.
- Büyük sosyal medya sitelerinde günlük işlenen veri boyutu 7 terabyte'a yaklaşmakta ve günden güne artmaktadır.
- Sağlık kuruluşları, hastalarına yönelik bireysel ve kişiselleştirilmiş sağlık hizmetlerini sağlaya-bilmek için bireysel durumdaki verileri kendi sayısal ortamlarında depolamaktadır.
- Bankalar, müşterileriyle ilgili sakladıkları bilgiler aracılığı ile kullanıcılarını tanıyan ve internet şubesinde o gün için hangi hizmeti aldığını bilen, ana sayfayı ve menüleri etkileşimli hâle getiren (kişiselleştirme uygulamaları), müşterilerine hatırlatmalar yapan, kişiselleştirilmiş arayüz deneyi-mi ve zengin içerikle sürekli hizmet sağlayan birer şube hâline geldi.
- İlaç depolarında yüz binlerce ilacın adını ve tam olarak nerede bulunduğunu bilen, herhangi bir ilaca ulaşmak istenildiğinde ilacın yerini doğru bir şekilde tespit eden sistemlerin arkasında büyük veriler yatmaktadır.
- Arama motorları sayesinde milyonlarca sayfa arasında en doğru ve en hızlı veriye ulaşabilmek için yine aynı şekilde arka planda büyük veriler ve bunların anlamlı hâle getirilmesi bulunmaktadır.

Big Data ile yapılabilecek her şey hayal gücü ile sınırlıdır. Sadece ne yapmak ve neyi bilmek istendiği-nin iyi belirlenmesi gerekmektedir.

6.1.3.2. Veri Madenciliği

Veri madenciliği, çok büyük veriden faydalı bilgi ve kalıp çıkarma süreci olarak tanımlanabilir. Veri madenciliği; toplama, çıkarma, analiz ve veri istatistiklerini içerir.

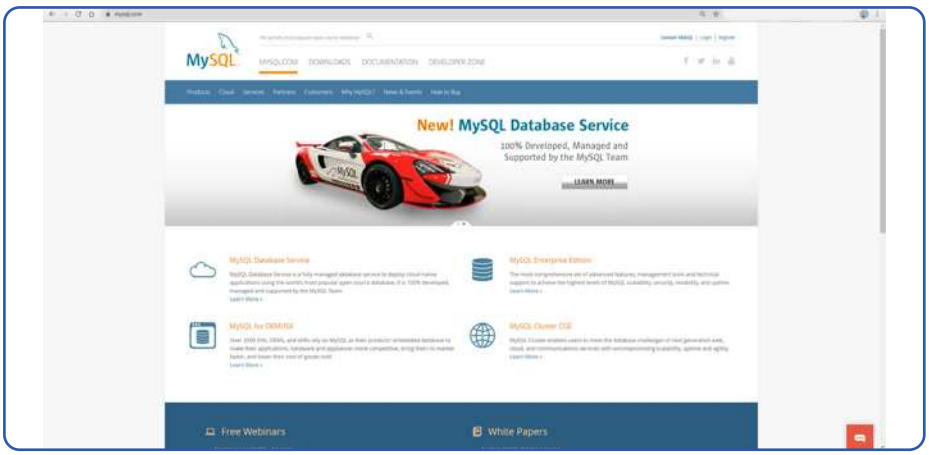
Veri madenciliği; daha iyi pazarlama stratejileri geliştirmek, performansı artırmak veya işletmeyi yö-netme maliyetlerini azaltmak için satış numaraları ile fiyatlar ve müşteriler dâhil olmak üzere ham verilerin incelenmesinde kullanılır. Veri madenciliği, tüketiciler arasında yeni davranış kalıplarını keşfetmeye de hizmet eder.

Veri madenciliğinin faydaları şöyle sıralanabilir:

- Veri madenciliğindeki türetilmiş model, müşteri davranışını daha iyi anlamada yardımcı olur. Bu da daha iyi ve üretken gelecek kararlarına yol açar.
- Veri madenciliği, iş için faydalı olan ancak henüz erişilemeyen pazar fırsatlarını yakalamak için kullanılır. Veri madenciliği ayrıca pazar alanını belirlemek, pazarlama hedeflerine ulaşmak ve iyi bir yatırım getirisi elde etmek için kullanılır.
- Veri madenciliği, potansiyel yatırım alanlarını keşfederek ve tanımlayarak işletme maliyetini düşürmeye yardımcı olur.

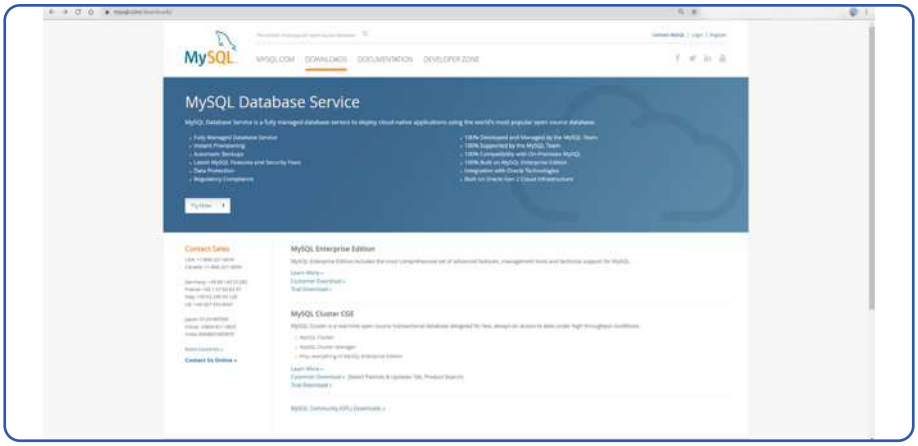
6.1.4. MySQL Veri Tabanı

MySQL şu an dünyada en çok kullanılan veri tabanıdır. MySQL resmi veb sitesine girildiğinde Görsel 6.2'deki ekran gelecektir. Üst tarafta yer alan DOWNLOADS'a tıklanır.



Görsel 6.2: mysql.com sayfası

Görsel 6.3'teki sayfada en altta yer alan **"MySQL Community (GPL) Downloads"** kısmına tıklanır.

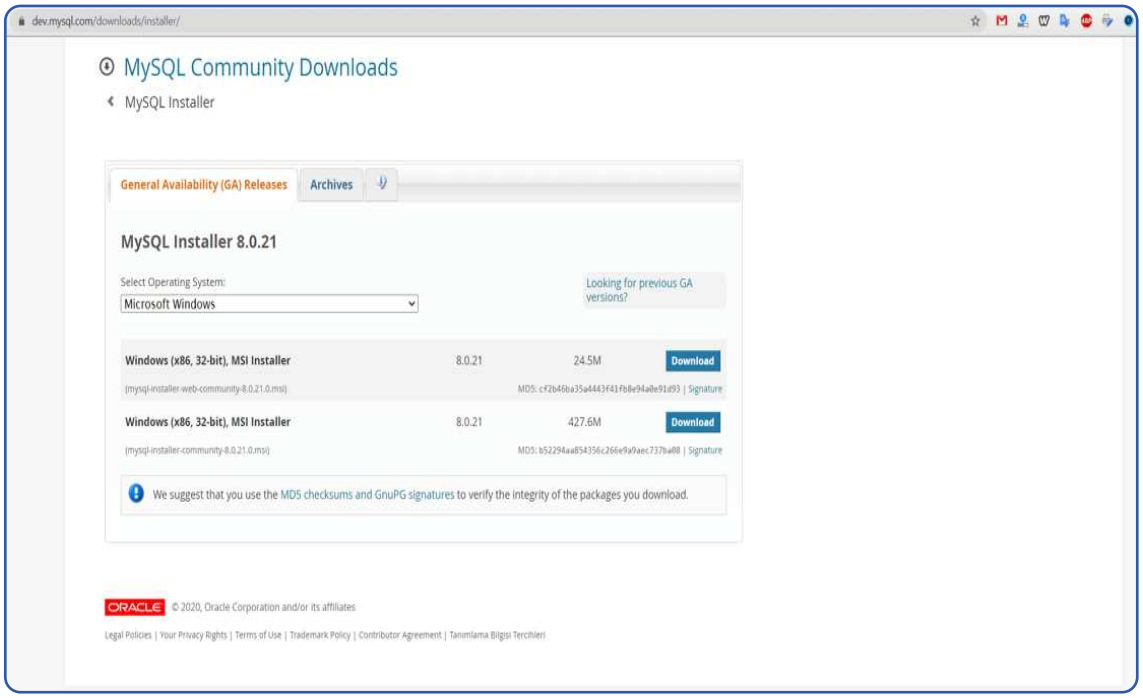


Görsel 6.3: mysql.com downloads sayfası

Görsel 6.4'teki sayfada uygun olan MySQL sürümü seçilir.

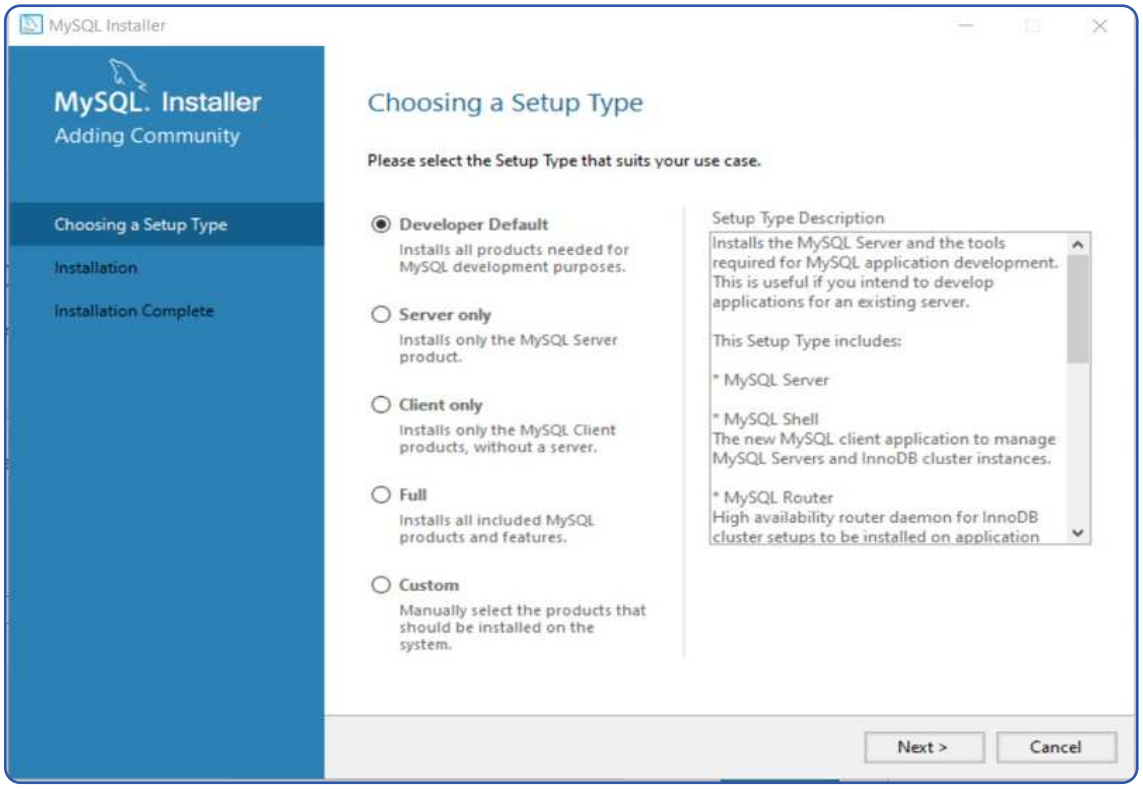
Görsel 6.4: MySQL Community Downloads sayfası

Görsel 6.5'teki sayfada uygun versiyon seçilir. Bu kurulum "**Desktop**" örneği üzerinden devam edecektir. İndirme işlemi bittikten sonra kurulum adımlarına geçilir.



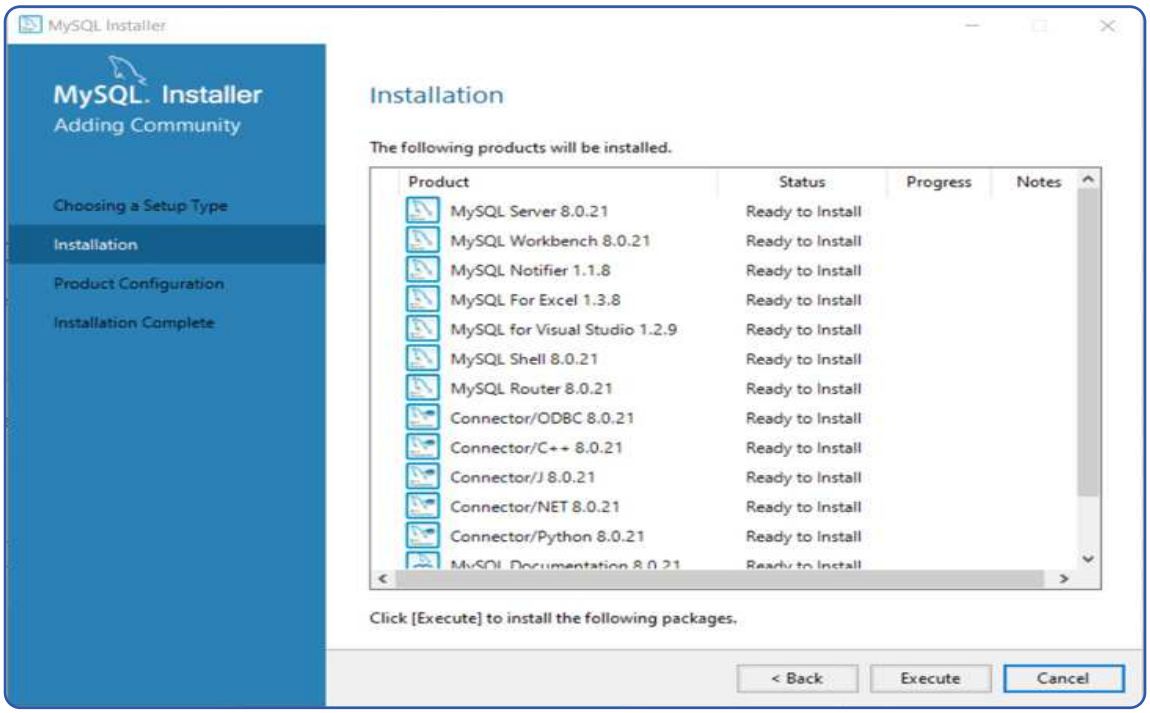
Görsel 6.5: MySQL indirme sayfası

Görsel 6.6'daki ekranda ne tür bir kurulum istenildiğinin seçilmesi gerekmektedir. Developer Default (Geliştirici Standart) kurulum için uygundur. İstenilen seçenek işaretlenerek "**Next**" düğmesine basılır.

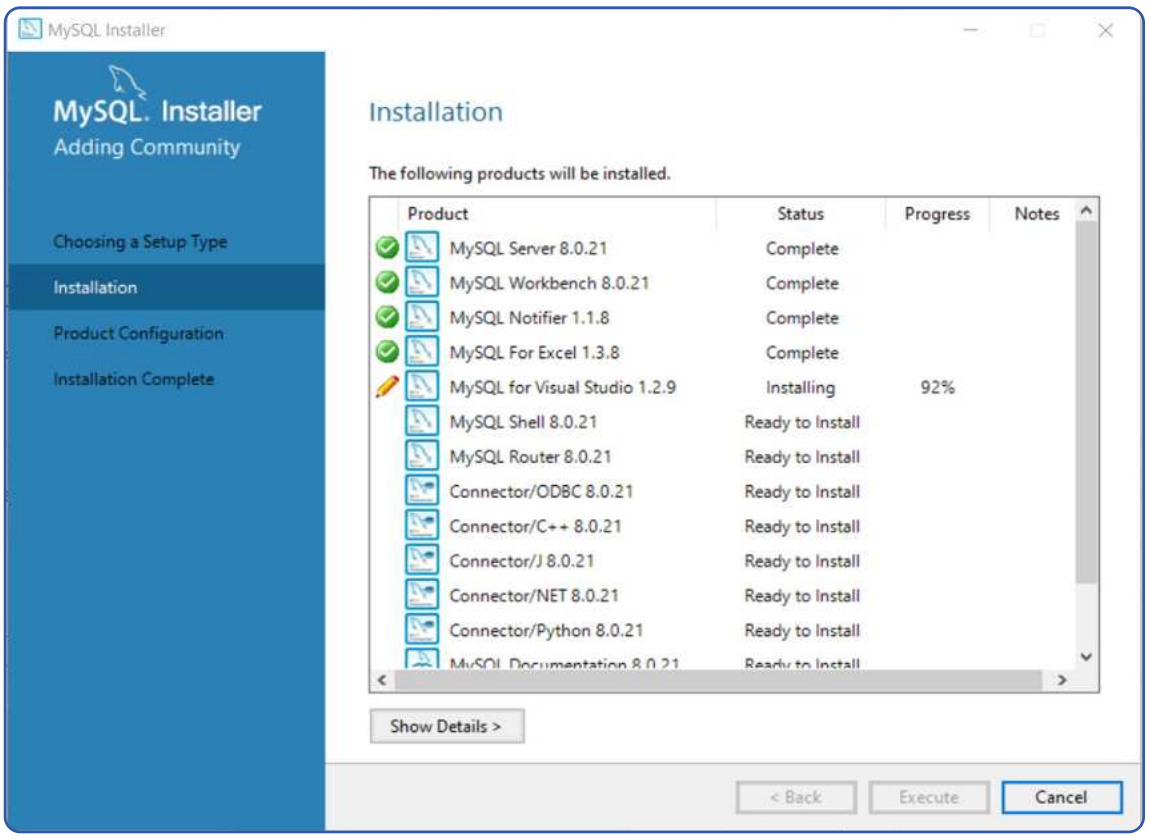


Görsel 6.6: MySQL kurulum türü seçme ekranı

Görsel 6.7'deki ekranda kurulacak bileşenler listelenmiştir. "**Execute**" düğmesine basılır ve kuruluma başlanır (Görsel 6.8).

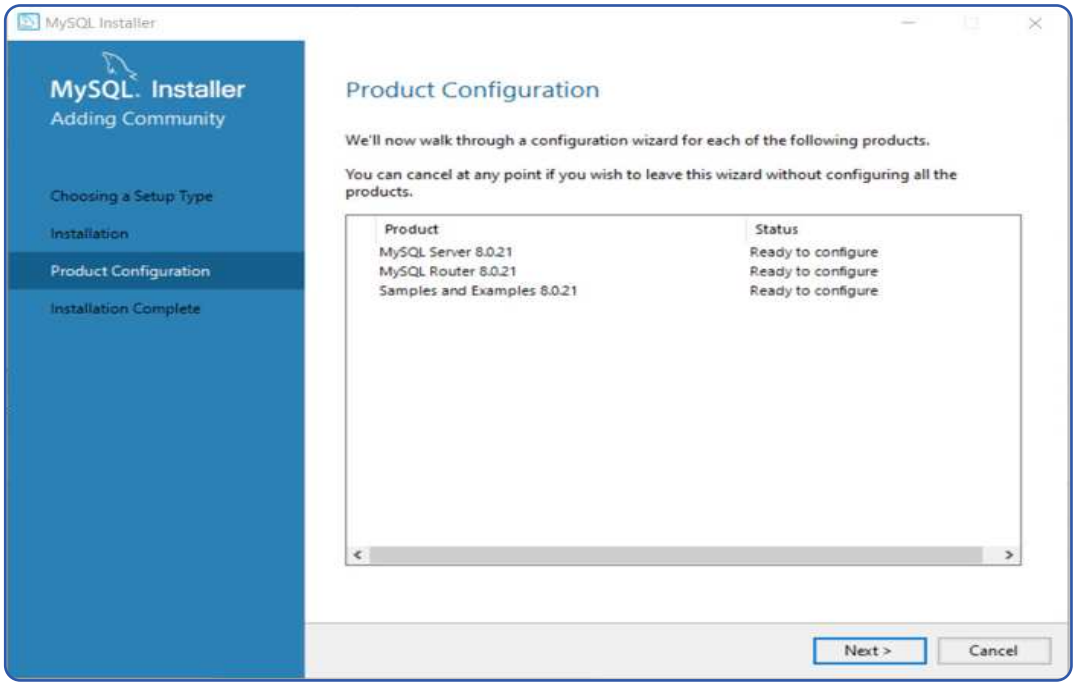


Görsel 6.7: Kurulacak bileşenler



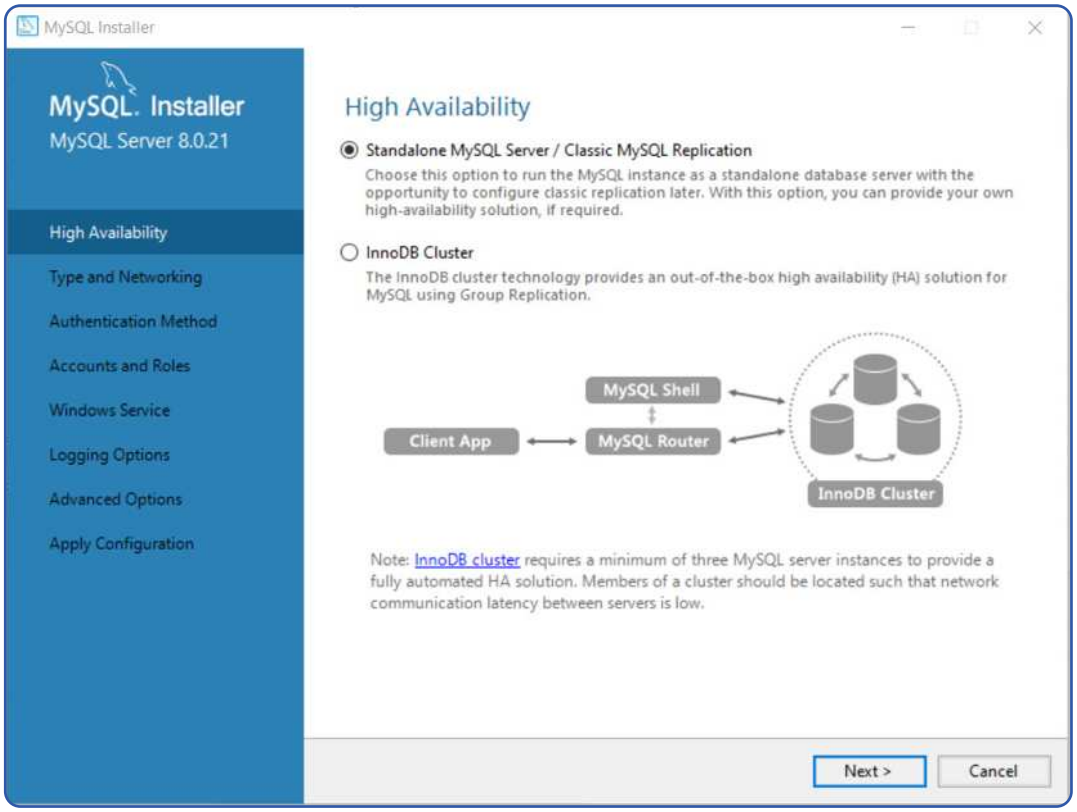
Görsel 6.8: Kurulumun başlaması

Görsel 6.9'daki ekranda kurulan bileşenlerin ayarları yapılabilir. Önemli olan kısım "**MySQL Server 8.0.21**" yazan bileşendir. Bu bileşen, MySQL Server ayarlarının yapılacağını gösterir.



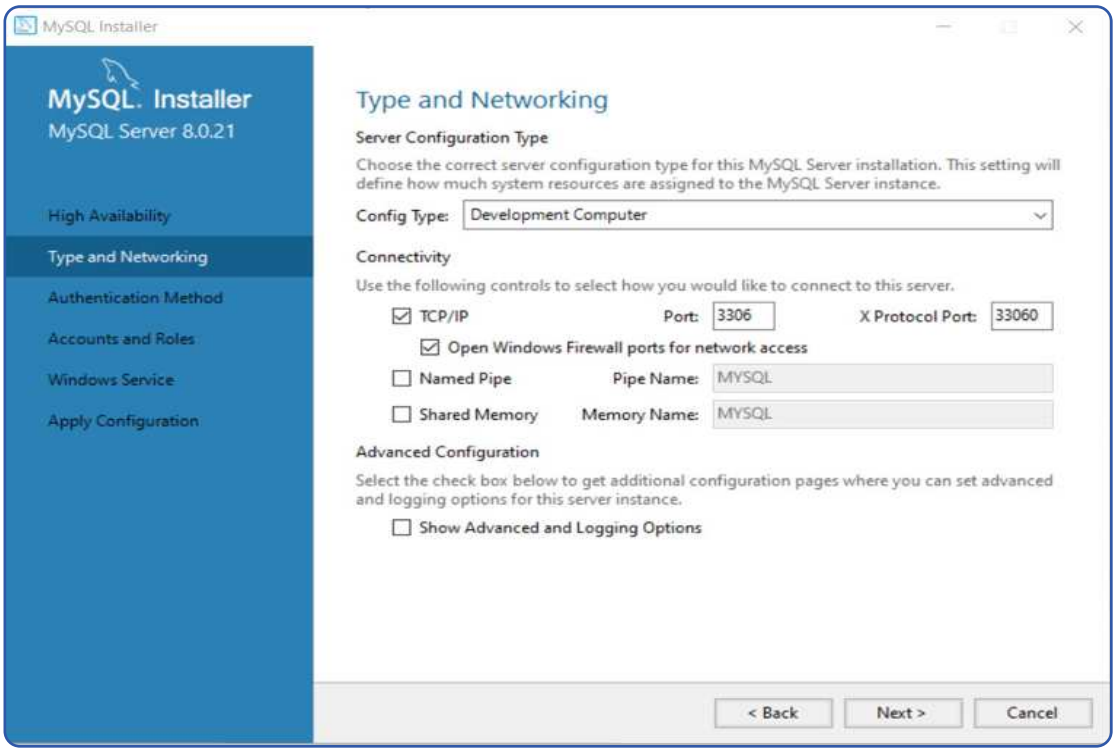
Görsel 6.9: Product Configuration ekranı

Görsel 6.10'daki ekranda standart bir kurulum için herhangi bir ayar yapılmasına gerek yoktur. "**Next**" düğmesine basılır.



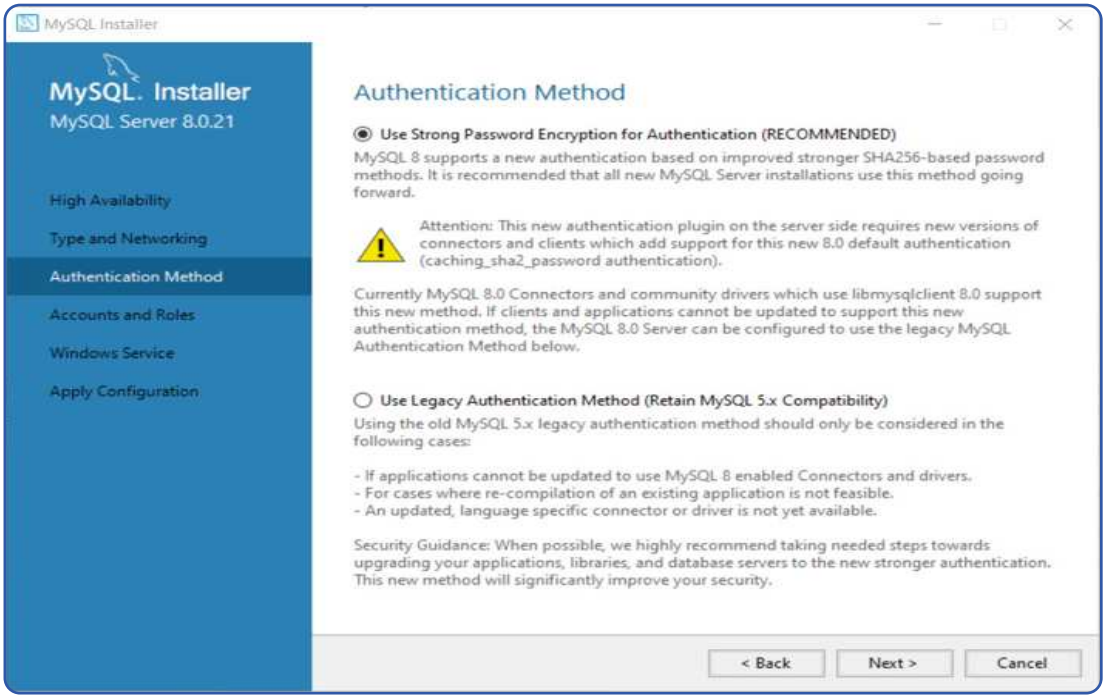
Görsel 6.10: MySQL Server ayarları

Görsel 6.11'deki "**Next**" düğmesine basılır.



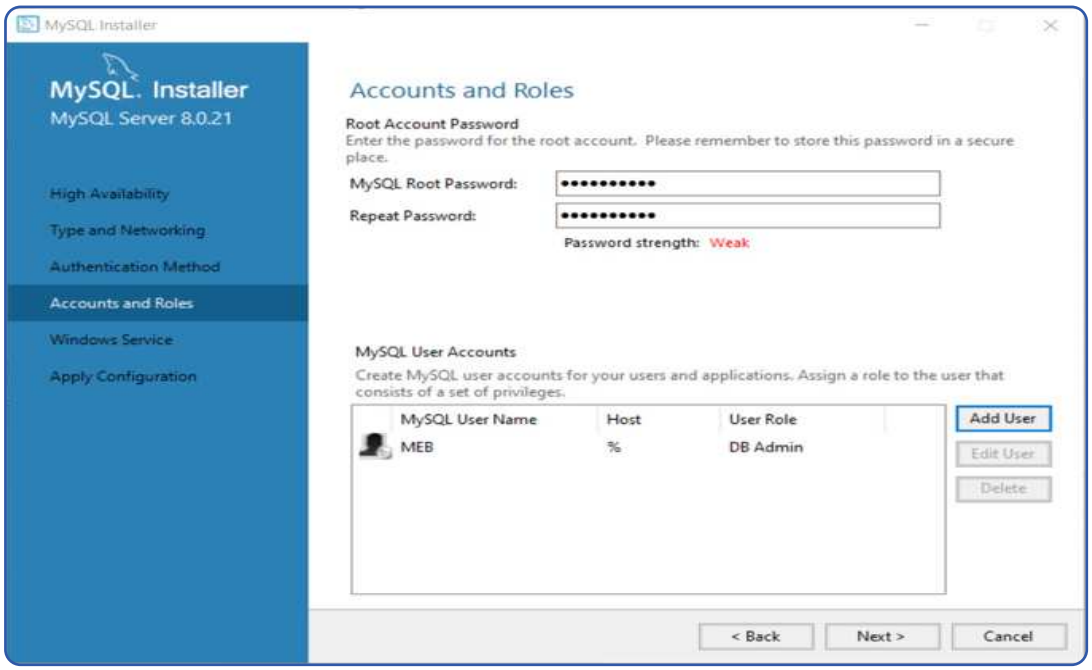
Görsel 6.11: Type and Networking ayarları

Görsel 6.12'deki ekranda MySQL Server erişim güvenliği için seçim yapılması istenmektedir. Günümüzde güvenlik kavramı çok önemlidir çünkü kimse veri tabanına başkalarının girmesini istemez. Bu ekranda "**Use Strong Password Encryption for Authentication**" seçeneği işaretlenerek üst düzey güvenlik ayarları seçilir.

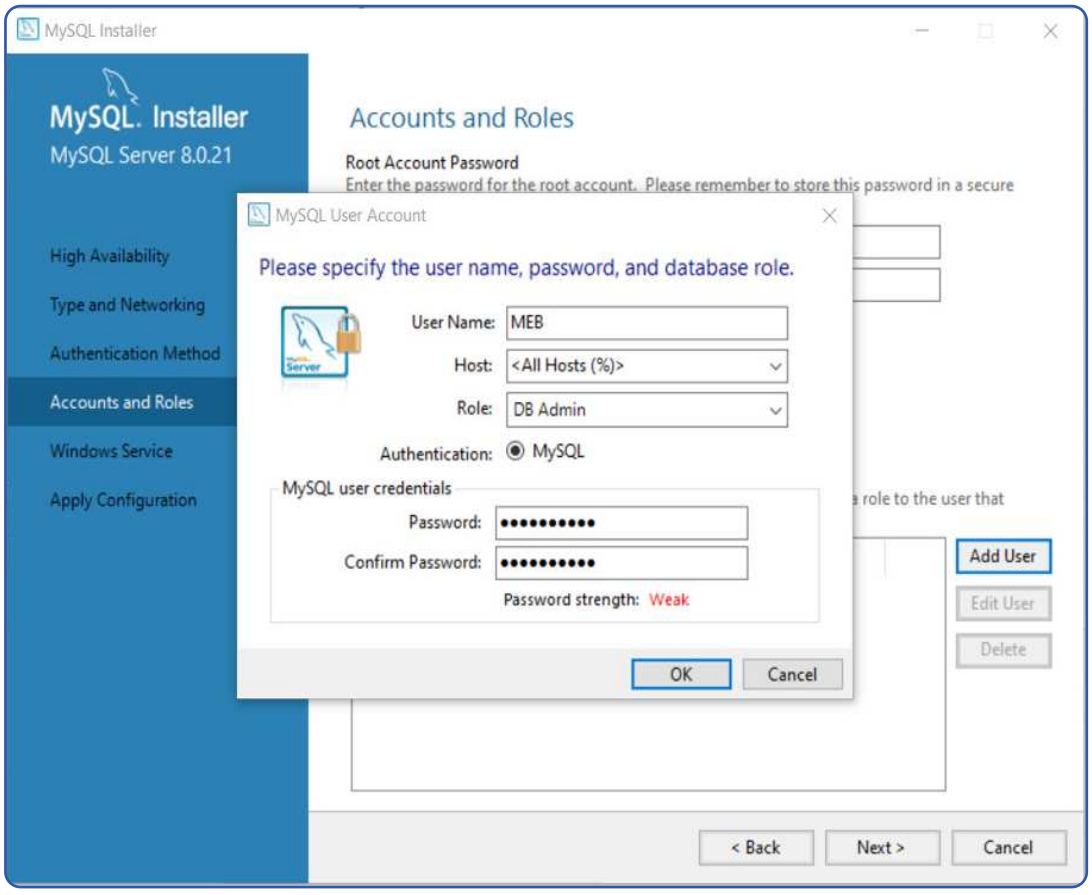


Görsel 6.12: Authentication Method ekranı

Görsel 6.13'teki ekranda MySQL Server erişimi için bir şifre belirlenmesi ve Server'a erişecek başka kullanıcılar varsa onların eklemesi gereklidir. Görsel 6.13'te üst kısımda MySQL Server şifresi belirlenmişken Görsel 6.14'teki **MEB** isminde bir kullanıcı "**Add User**" düğmesi ile eklenmiştir.

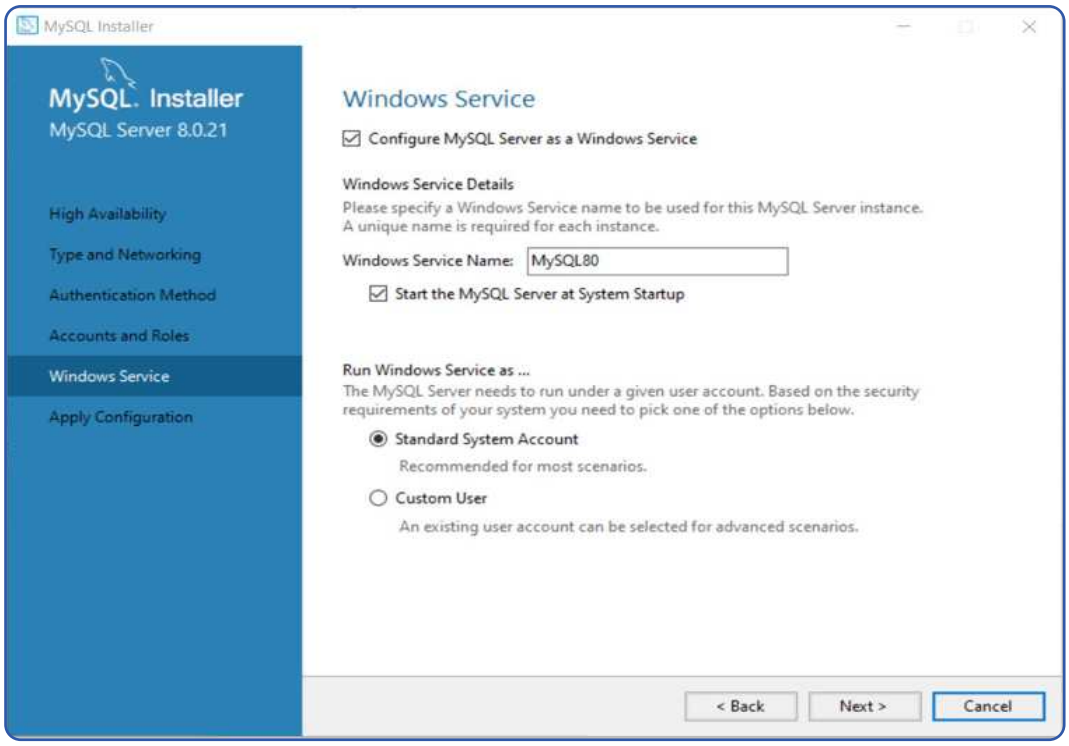


Görsel 6.13: Accounts and Roles ayarları



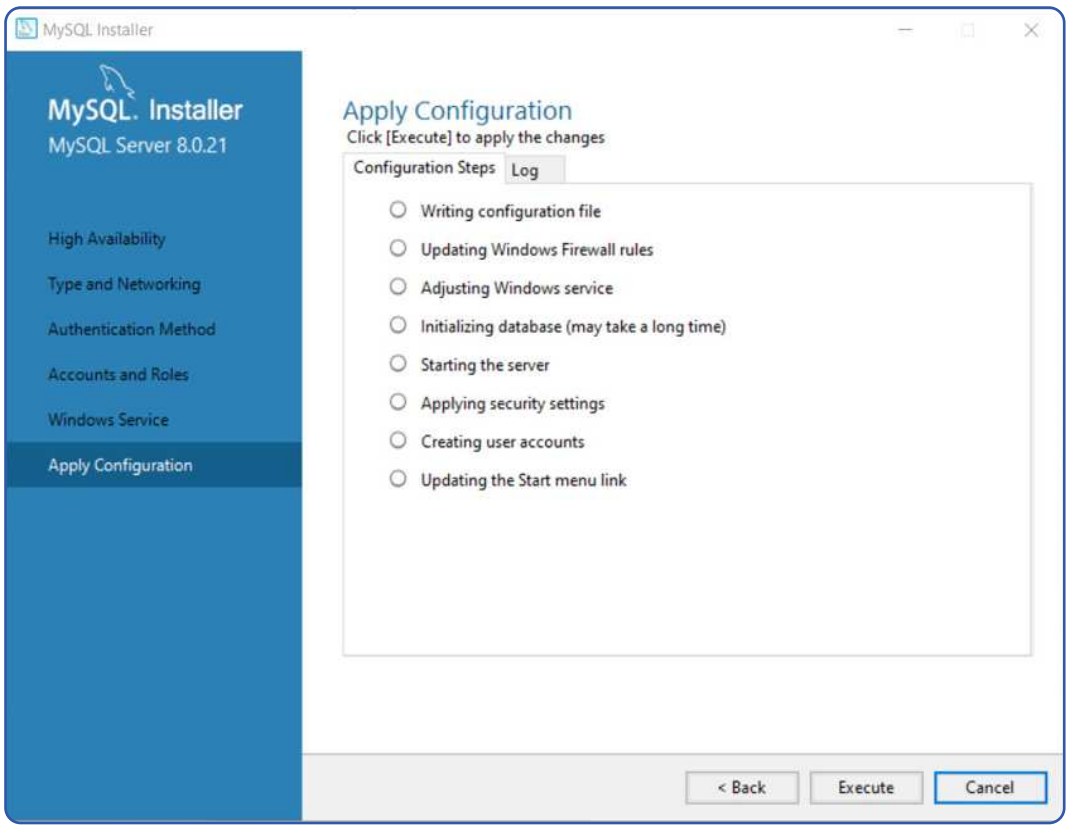
Görsel 6.14: MySQL Server'a kullanıcı eklenmesi

Görsel 6.15'teki "**Next**" düğmesine basılır.

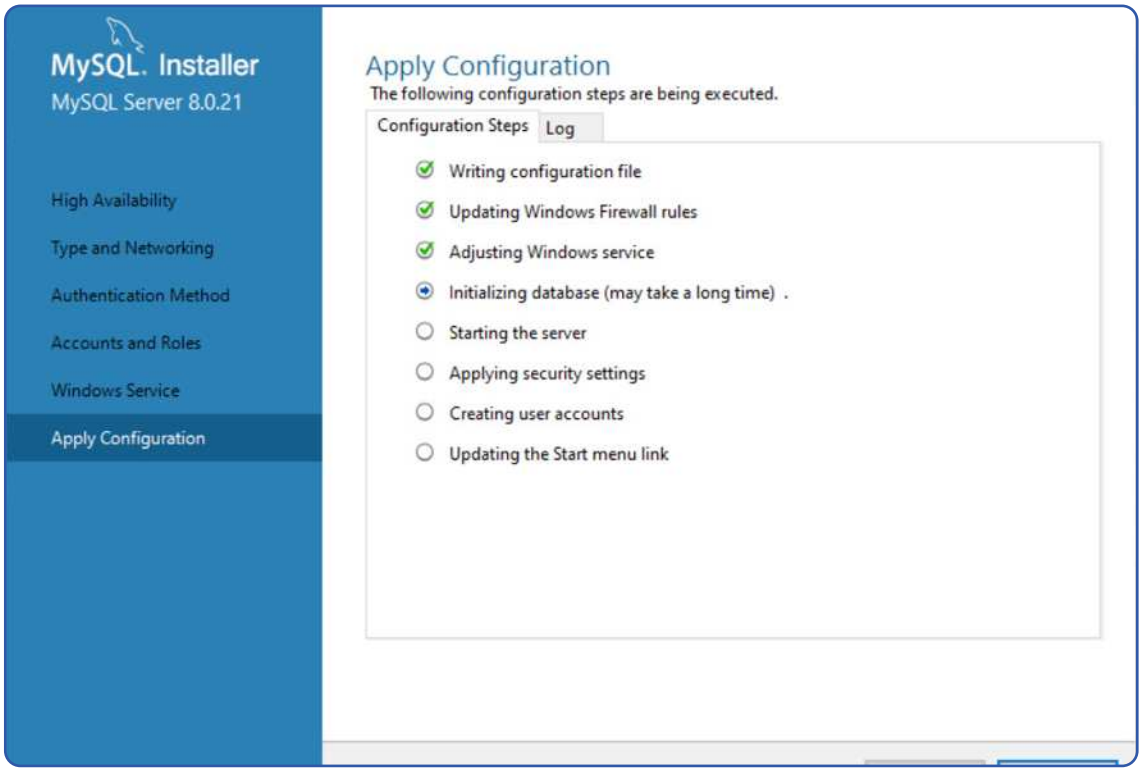


Görsel 6.15: Service ayarları

Görsel 6.16'daki ekran ile MySQL Server ayarlarının uygulanması sağlanır. "**Execute**" düğmesine basılır ve ayarlar uygulanmaya başlanır (Görsel 6.17).

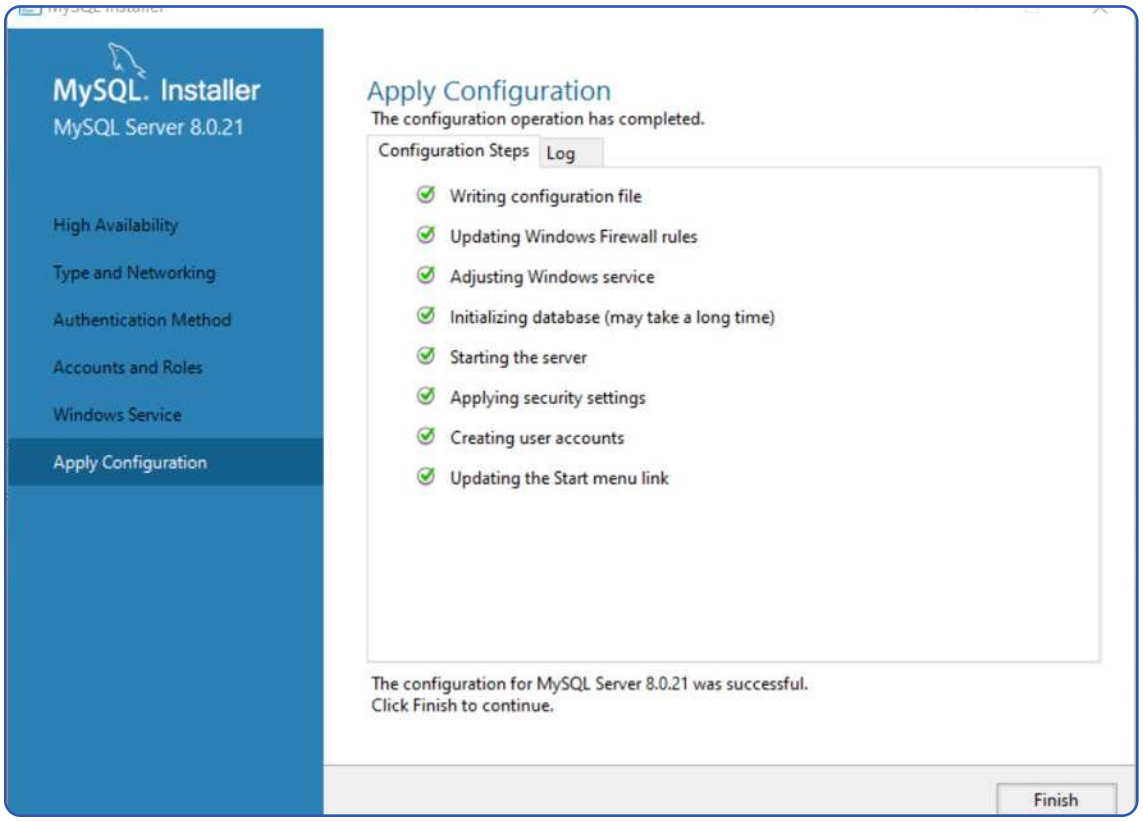


Görsel 6.16: Apply Configuration ekranı



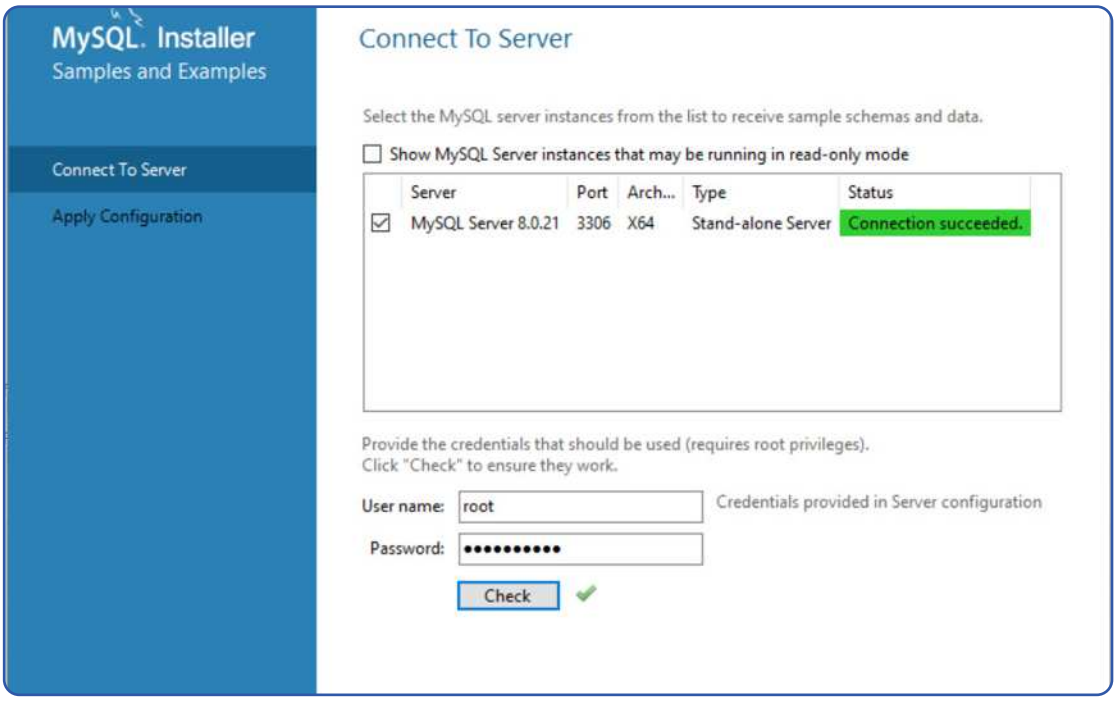
Görsel 6.17: Server ayarları uygulama ekranı

Görsel 6.18'de görüldüğü gibi bütün ayarlar doğru bir şekilde uygulanmış ve MySQL Server kullanıma hazır duruma gelmiştir.

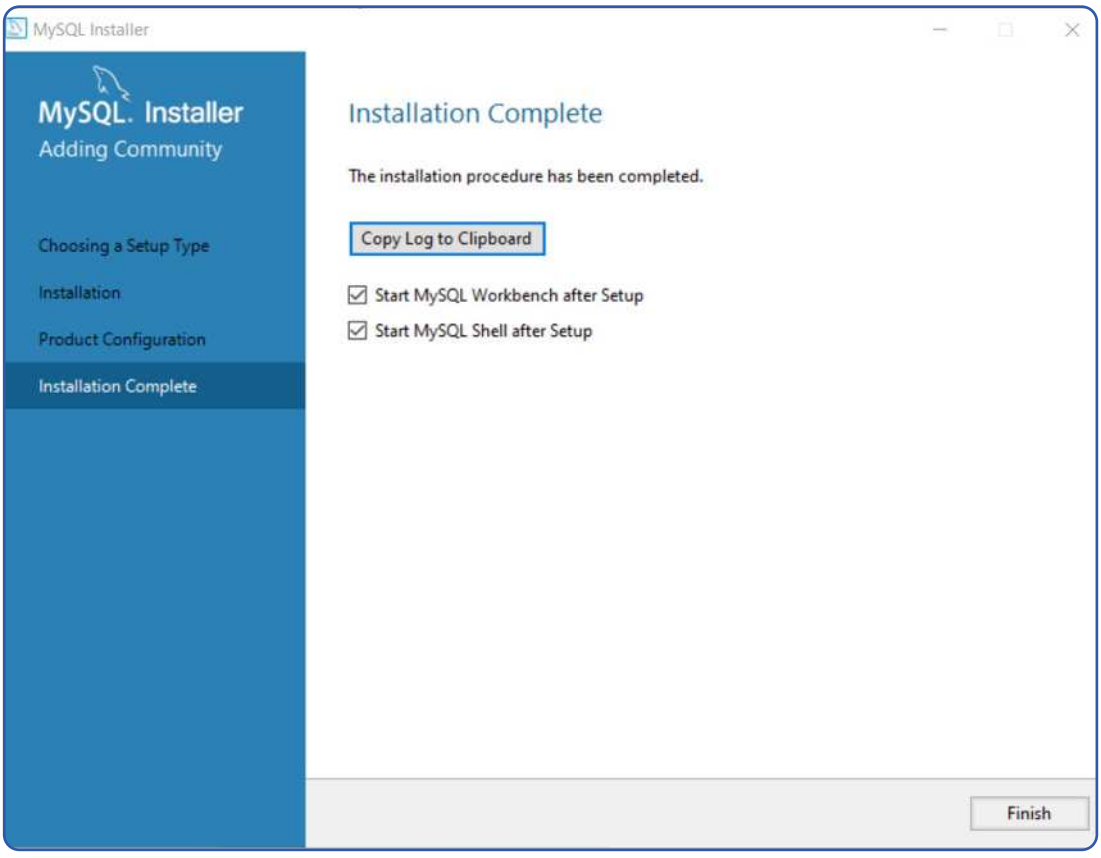


Görsel 6.18: Server ayarlarının tamamlanması

Görsel 6.19'daki ekranda kurulum aşamasında belirlenen şifre ile MySQL Server'ın çalışıp çalışmadığı test edilir. "**Connection Succeeded**" yazısı görülüyorsa MySQL Server ayarları doğru yapılmıştır ve MySQL Server doğru çalışmaktadır. "Next" tuşuna basılarak Görsel 6.20'deki ekrana ulaşılır.



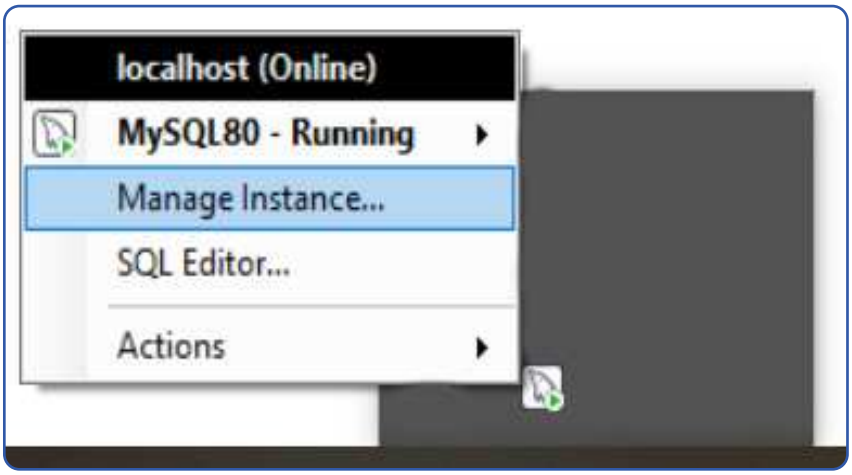
Görsel 6.19: Connect To Server ekranı



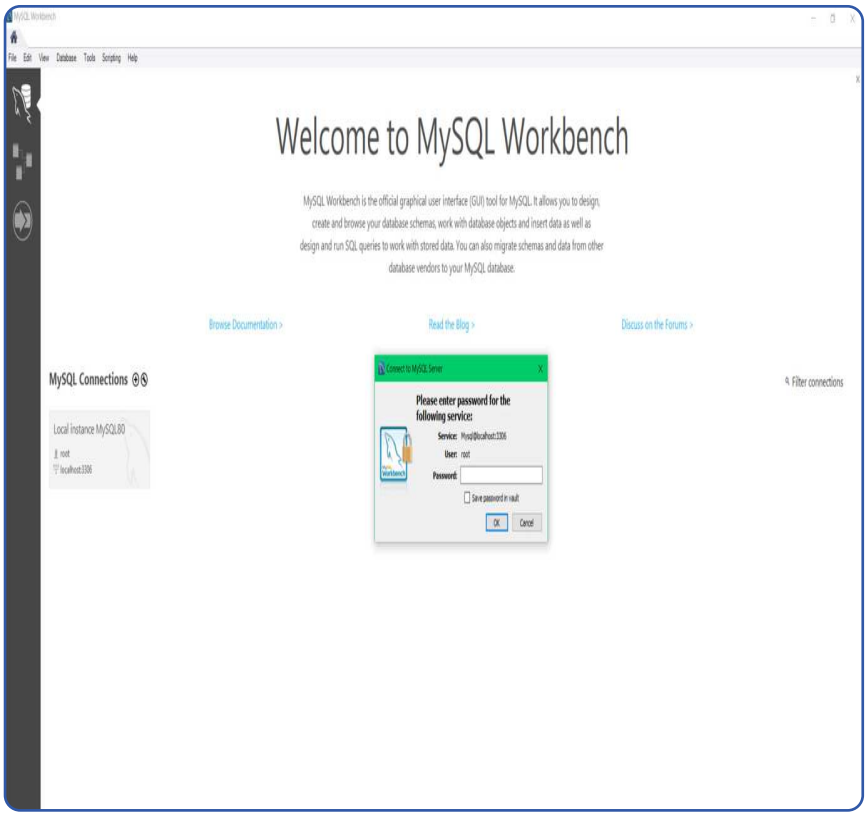
Görsel 6.20: MySQL Server kurulumunun bitiş ekranı

6.2. MySQL Server Arayüz (Workbench) Ekranı

MySQL Server kurulumu bittikten sonra MySQL kurulumunda yüklenen "**Workbench**" özelliği ile MySQL Server'ı yönetecek arayüze ulaşılır. Başlat menüsünden ya da görev çubuğuna gelen MySQL simgesindeki "**Manage Instance…**" kısmından Workbench arayüzü başlatılır (Görsel 6.21).



Görsel 6.21: Görev çubuğundan MySQL Workbench çalıştırma

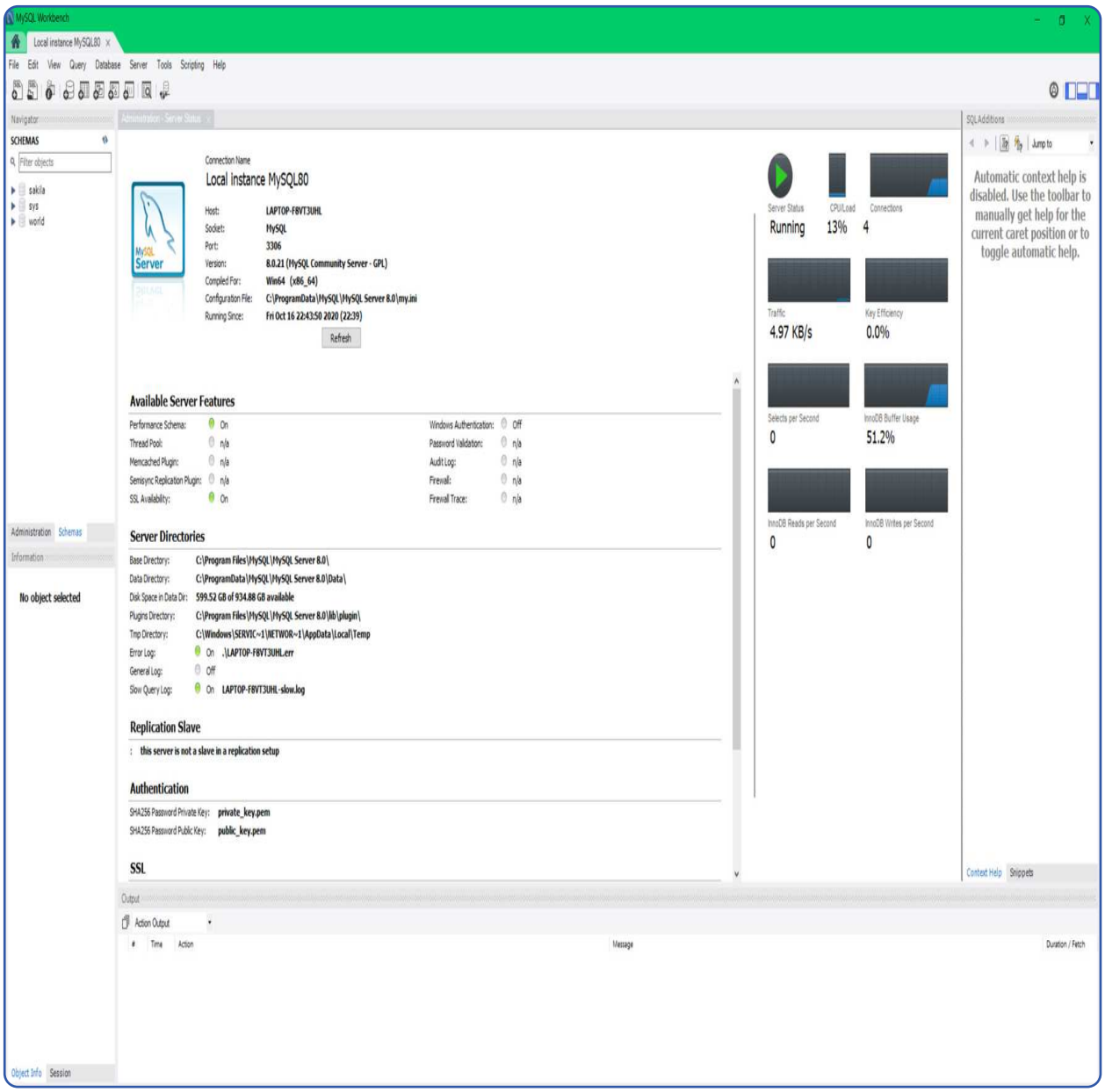


Görsel 6.22: MySQL Workbench arayüz giriş ekranı

Görsel 6.22'deki ekranda kurulum esnasında belirlenen parola girilmelidir. Ekranın alt tarafındaki "**Save password in vault**" seçeneği işaretlenerek bir daha şifre sorulmaması sağlanır.

Şifre girildikten sonra MySQL başlangıç ekranı ile karşılaşılır. Bu ekranda Server ile ilgili bilgiler ve değerler yer almaktadır.

Görsel 6.23'te görülen ekrandaki alanlar şunlardır:



Görsel 6.23: MySQL Workbench Server Status ekranı

- Navigator: Bu penceredeki "Schemas" alanında veri tabanları görünür. MySQL kurulduktan sonra kendi içinde üç örnek veri tabanı görünmektedir.
- Information: Seçilen bileşene ait (veri tabanı, tablo vb.) bilgilerin görülebileceği alandır.
- Output: Yapılan işlemlerin sonuçlarının görülebileceği alandır. İşlem doğru tamamlanmışsa yeşil renkte, işlemde hata oluşmuşsa hata mesajıyla beraber kırmızı renkte görünecektir.
- SQL Additions: Komutlar ve bileşenlerle ilgili yardım alınabilecek alandır.
- Sekmeler: MySQL ile ilgili her işlemin yapıldığı alandır. Orta alanın tamamını kullanır. Erişilen bütün bileşenlerle ilgili işlemler bu alanda açılacak sekmelerin içinde yapılır.

6.3. Veri Türleri

C#'ta değişkenlerin veri tipi belirlendiği gibi SQL veri tabanı tasarımı yapılırken de her kolon (alan) için veri tipi tanımlanır. Bu tanımlamalar yapılırken dikkat edilmesi gereken noktalardan biri, mümkün olduğunca en uygun ve hafızada en az yer kaplayacak veri tipini seçmektir (Tablo 6.1).

Tablo 6.1: Temel Veri Tipleri

Veri Tipi	Türü	Açıklama
char(n)	String	Uzunluğu değişmeyen sabit verileri saklar. n değeri 10 ise daha kısa bir değer girilince kalan boşluğu kendi tamamlar ve öylece saklar.
varchar(n)	String	Değişebilir uzunlukta verileri saklar. En fazla 8.000 karakter alır. n değeri maksimum değerdir. Daha kısa bir değer girilse bile bu değeri olduğu gibi kaydeder.
text	String	Değişebilir uzunlukta karakterleri saklar. En fazla 2 GB metin içerir.
nchar	String	Sabit uzunlukta Unicode karakterleri saklar. char tipinden farkı, çoklu dil ve Unicode desteğinin olmasıdır. En fazla 4.000 karakterdir.
nvarchar	String	Değişebilir uzunlukta verileri saklar. varchar tipinden farkı, çoklu dil ve Unicode desteğinin olmasıdır. En fazla 4.000 karakterdir.
bit	Boolean	0, 1 ve Null değerlerini saklar.
tinyint	Tam sayı	0 ile 255 arasındaki değerleri saklar.
smallint	Tam sayı	-32.768 ile 32.767 arasındaki değerleri saklar.
int	Tam sayı	-2.147.483.648 ile 2.147.483.647 arasındaki değerleri saklar.
bigint	Tam sayı	-9.223.372.036.854.775.808 ile 9.223.372.036.854.775.807 arasındaki değerleri saklar.
datetime	Tarih	1 Ocak 1753–31 Aralık 9999 arasındadır.
smalldatetime	Tarih	1 Ocak 1900–6 Haziran 2079 arasındadır.
date	Tarih	1 Ocak 0001–31 Aralık 9999 arasındadır. Sadece tarih içerir, saati saklamaz.
decimal	Ondalık	Hafızada ondalık sayı türünden veri tutmak için kullanılır. Örneğin 102,35 sayısını hafızada tutmak için DECIMAL (5,2) şeklinde veri türü tanımlanmalıdır.
float	Ondalık	Hafızada ondalık sayı türünden veri tutmak için kullanılır.

Bunlar dışında da veri türleri mevcuttur.

6.4. Veri Tabanı Tasarımı

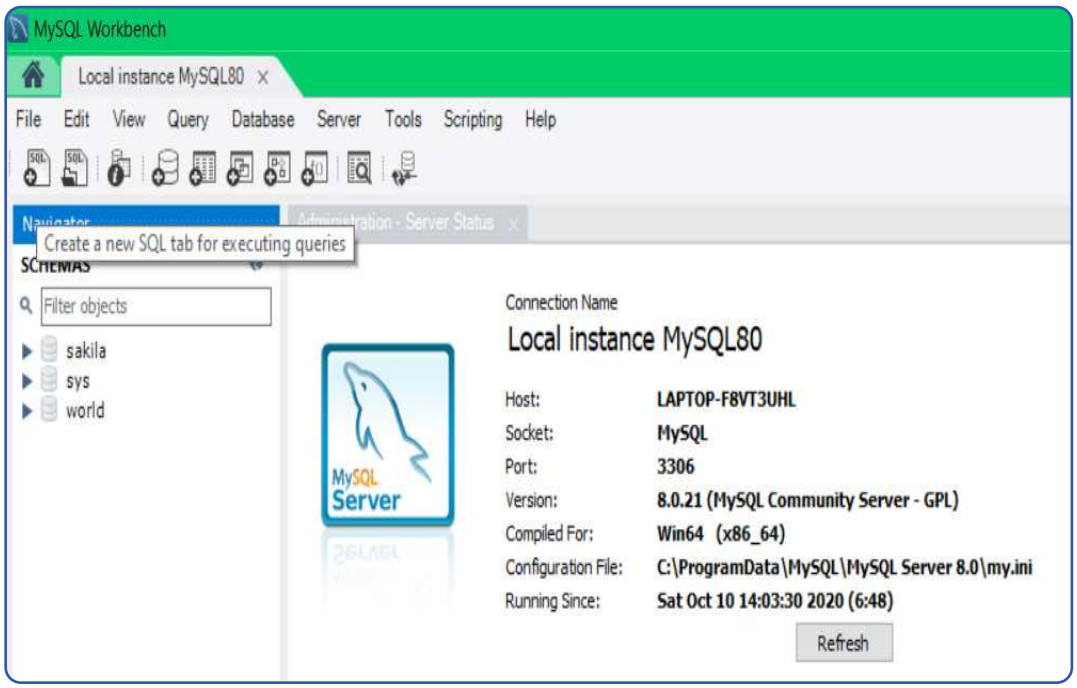
Kodlama öncesi nasıl algoritma yapılıyorsa veri tabanı oluşturmadan önce de bir veri tabanı tasarımı yapmak, kabaca bir taslak çıkarmak önemlidir. Öncelikle veri tabanının hangi amaçla oluşturulacağı belirlenmelidir. Bu amaca yönelik bir isim belirlenmesi de doğru olacaktır. Yapılacak veri tabanı, birbirine bağlı alanların bulunduğu bir yapıda (**ilişkisel**) olmalıdır.

Burada öğrenci bilgilerini, notlarını ve derslerini kayıt altında tutacak bir veri tabanı tasarlanacaktır. Veri tabanında şu an için üç adet tablo olacaktır. Bu tabloların içine gerekli alanlar tanımlanmalıdır. İstenildiği zaman tablolarda ekleme ve çıkarma gibi işlemler yapılabilir.

Not: Veri tabanı tasarımı oluşturulurken Türkçe karakter kullanılmamalıdır.

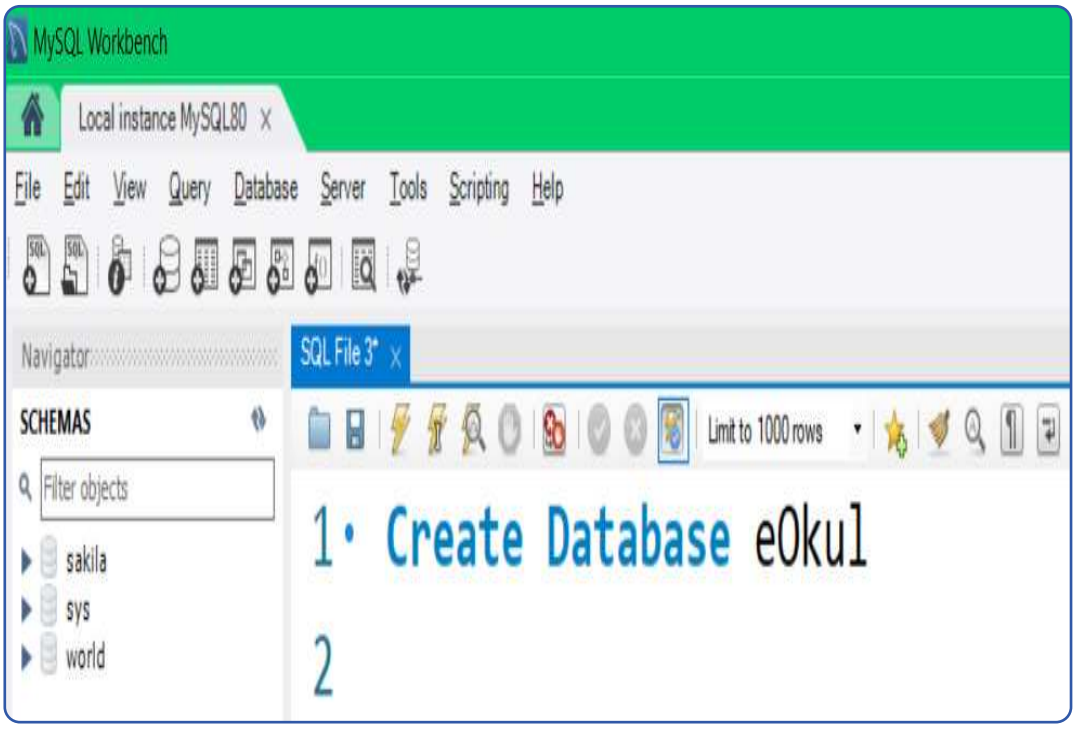
6.4.1. Veri Tabanı Oluşturma

Veri tabanları, SQL komutları ve MySQL arayüzü kullanılarak iki farklı şekilde oluşturulabilir. SQL komutuyla veri tabanı oluşturmak için Görsel 6.24'teki MySQL Workbench arayüzünde bulunan simgesine tıklanır.



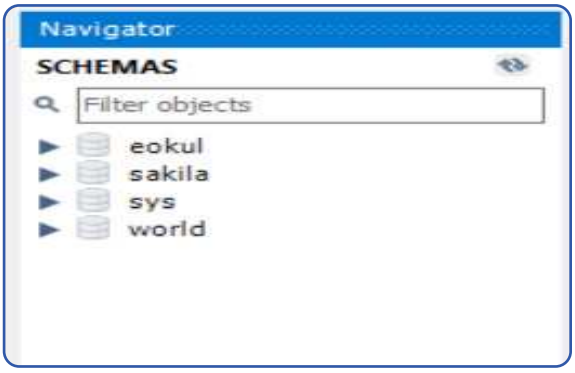
Görsel 6.24: SQL Query sayfası açma

SQL komutuyla veri tabanı oluşturulurken "**Create Database İsim**" komutu ile belirtilen isimde veri tabanı (**database**) elde edilir (Görsel 6.25).



Görsel 6.25: Veri tabanı oluşturma ve komutun doğru çalıştığını gösteren output çıktısı

Sadece tek bir komutla içinde milyonlarca veri tutabilecek bir veri tabanı oluşturuldu (Görsel 6.26).



Görsel 6.26: Veri tabanının MySQL'e eklenmesi

6.4.2. Veri Tabanında Anahtarlar (Keyler)

- **Primary Key (PK-Birincil Anahtar)**

Primary Key (birincil anahtar), bir veri tablosunda yer alan her satır için bir vekil veya tanımlayıcı (**identify**) görevi görür, kısıtlamadır (**constraint**) ve eşsizdir. Satırlara ait değerlerin karışmaması adına bu alana ait bilginin tekrarlanmaması gerekir. Birincil anahtarın temel işlevi, verilerin hangi satıra dizileceğine veya hangi satırda düzenleneceğine karar vermesidir. Çoğunlukla tek bir alan (id, user_id, e_mail, username, national_identification_number vb.) olarak kullanılsa da birden fazla alanın birleşimiyle oluşturulabilir. Sayılar genelde birincil anahtar olarak seçilir ancak bu bir zorunluluk değildir. Birincil anahtar değeri boş geçilemez ve Null değeri alamaz. **İlişkisel veri tabanlarında** (Relational Database Management System) mutlaka birincil anahtar olmalıdır.

- **Foreign Key (FK-Yabancı Anahtar)**

Foreign Key (yabancı anahtar) ikincil anahtar olarak da ifade edilmektedir. Bir veri tablosuna girilebilecek değerleri başka bir veri tablosundaki alanlarla ilişkilendirmeye yarar. Başka bir tablonun birincil anahtarının bir diğer tablo içinde yer almasıdır. **Çoğunlukla bir ana tablo (parent) ile alt tablonun (child) ilişkilendirilmesinde kullanılır.** Bu sayede olası veri tekrarlarının önüne geçilebilmekte ve ilerleyen zamanda söz konusu olabilecek güncellemelerde ilgili verilerin her yerde güncellenmesi sağlanabilmektedir.

- **Unique Key (UQ-Tekil veya Benzersiz Anahtar)**

Unique Key (tekil anahtar), yer aldığı tablo içinde bir değeri sadece bir kere alır. İlgili değerin tekrar atanmasına izin verilmez. **Bu anahtarın işlevi birincil anahtar ile benzerlik taşısa da ikisi arasındaki en önemli farklılık, Unique Key'in Null değeri alabilmesidir.**

PK ile UQ arasındaki farklar şöyle sıralanabilir:

- Her birincil anahtar benzersiz iken her benzersiz anahtar birincil değildir.
- Bir tablo içinde birden fazla alan tek birincil anahtar ile tanımlanabilir ancak birincil anahtar yapısı her tabloda sadece bir tane olabilir. Benzersiz anahtarda ise böyle bir sınırlama yoktur.
- Birincil anahtarda Null kullanımına izin verilmez ancak benzersiz anahtarlarda Null kullanımına izin verilir.
- Birincil anahtar ile veri tablosu üzerinde bir Index tanımı oluşturulur ve her kaydın benzersiz bir tanımı yapılır. Benzersiz anahtar kullanımında ise değerlerin benzersiz olup olmadığına bakılır.

6.4.3. Tablo Oluşturma

Bir veri tabanı içinde onlarca ya da yüzlere tablo yer alabilir. Tablolar, veri tabanları içinde verileri tutan yapılardır. En başta analiz doğru yapılırsa tablolar oluşturulurken yaşanacak sıkıntılar en aza indirilir. Tabloda özel bir durum yoksa mutlaka bir tane primary key bulundurulur. Böylece tablo, diğer tablolarla ilişkilendirilmek istenildiğinde sıkıntı yaşanmaz. Tablo oluştururken komutlar veya arayüz kullanılabilir. Aşağıda bir tablo taslağı görülmektedir.

```
CREATE TABLE tabloadı (
sütunAdı1 veriTipi diğerParametreler ,
sütunAdı2 veriTipi diğerParamatreler ,
...
)
```

Create Table komutu kullanılırken tablonun her alanı (sütunu) için veri tipi ve başka parametreler girilir. Bu parametreler şu şekilde sıralanabilir:

- **Not Null:** Alan için bir değerin mutlaka girilmesi gerektiğini gösterir (Bu parametre daha çok birincil anahtarlar için kullanılır.).
- **Primary Key:** Alanı birincil anahtar olarak belirler.
- **Auto_increment:** Primary key alanına kayıt eklendikçe otomatik olarak bir sayı artacaktır.

```
SID int primary key auto_increment,
```

Görsel 6.27: Örnek bir SQL satırı

Tablonun SQL komutları yazılır (Görsel 6.28).

```
Create Table ogrenciBilgi(
ogrenciNo int primary key,
ogrenciAdi varchar(20) not null,
ogrenciSoyadi varchar (20) not null,
ogrenciBolumu varchar (30) not null,
ogrenciSinifi tinyint not null,
ogrencidogumtarihi date not null
);
```

Görsel 6.28: ogrenciBilgi tablosu için SQL kodları

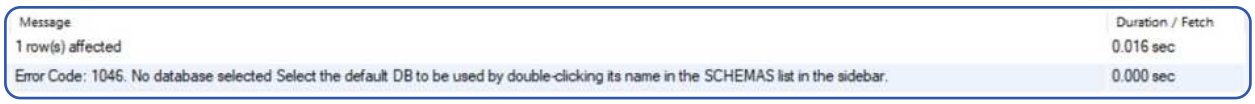
Create Table komutuyla bir tablo oluşturulduktan sonra bu tabloda yer alacak alanlar tek tek belirlenir. **Tablo oluşturulurken birinci alan (ogrenciNo) primary key olarak belirlenir.**

Sıra Sizde

Primary key alanının neden auto increment olarak belirlenmediğini arkadaşlarınızla tartışınız.

Öğrenci numarasından sonra öğrencinin adı, soyadı ve bölümü varchar veri türüyle belirlenir.

Varchar veri türlerinin yanındaki sayılar, o alan için hafızada ne kadar yer ayrıldığını göstermektedir. Daha sonra öğrencinin sınıfının (9, 10, 11, 12) kaydedileceği sınıf alanı ve öğrencinin doğum tarihinin girileceği doğum tarihi alanı belirlenir. Görsel 6.28'deki kodlar yazıldıktan sonra üst taraftaki yıldırım simgesiyle kodlar çalıştırılabilir ve sonuçlar "Output" bölmesinden takip edilebilir (Görsel 6.29, Görsel 6.30).



Görsel 6.29: Sol taraftan veri tabanı seçmeden kodu çalıştırınca ortaya çıkan hata

Görsel 6.30: SQL kodunun düzgün çalıştığına dair output çıktısı

Görsel 6.31: ogrencibilgi tablosunun veri tabanına eklenmesi

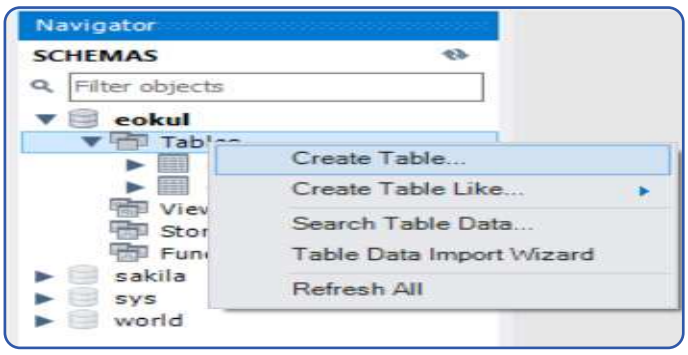
Görsel 6.32'deki kodlarla dersler tablosu oluşturulur.

```
create table dersler(
derskodu varchar(4) primary key,
dersadi varchar(45) not null
);
```

Görsel 6.32: Dersler tablosunun SQL kodları

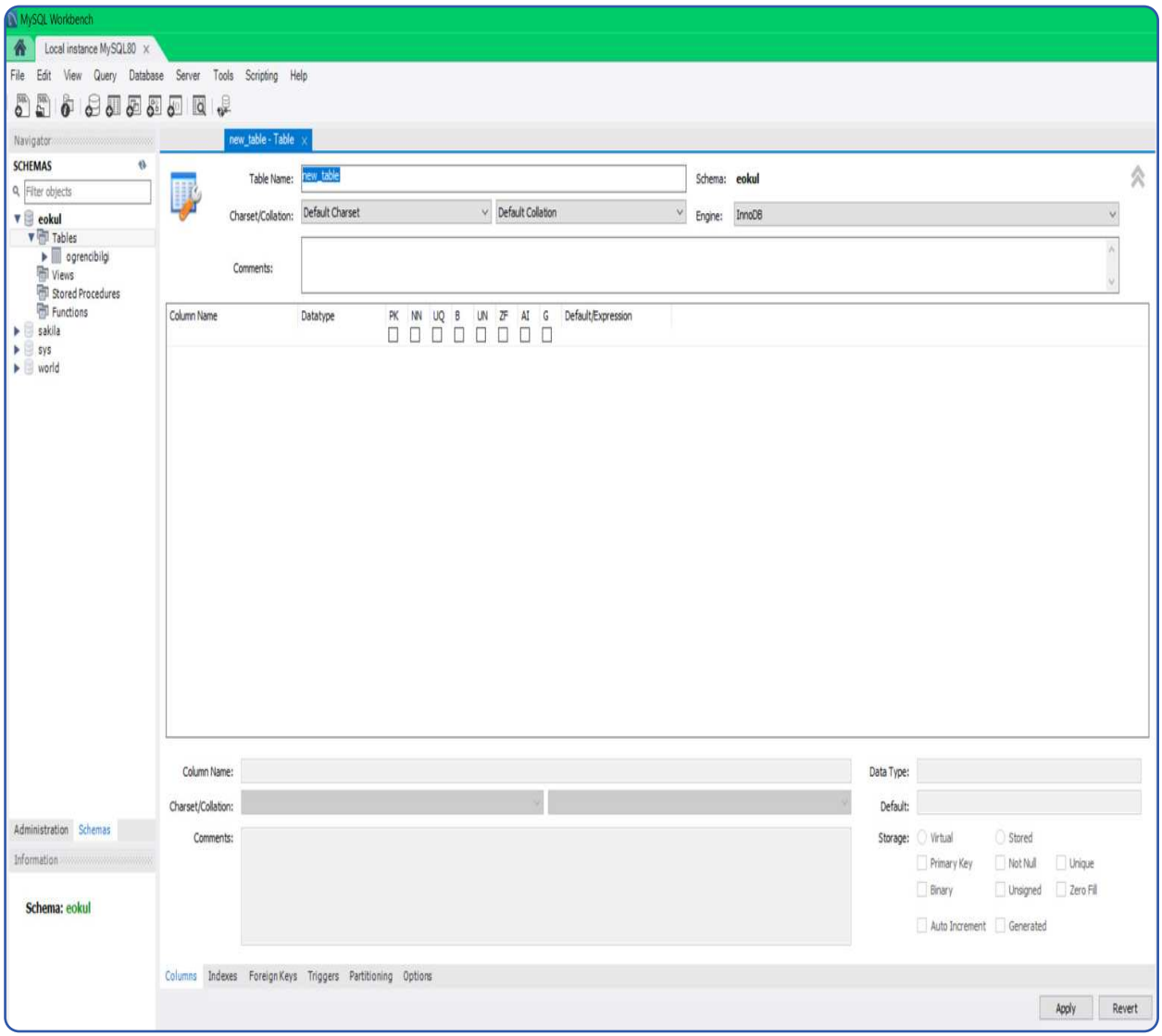
Burada dersler için 4 karakterlik bir primary key alanı ve ders adının kaydedileceği bir alan oluşturulur.

Tıpkı veri tabanı gibi tablolar da hem SQL kodlarıyla hem de MySQL Workbench arayüzüyle oluşturulabilir. Üçüncü tablo, MySQL arayüzü kullanılarak oluşturulur (Görsel 6.33).



Görsel 6.33: Tables'a sağ tıklanarak Create Table seçimi

SQL kodları çalışacak olsa da arayüz sayesinde MySQL bütün kodları otomatik olarak yazmaktadır. Tek fark, işlemlerin tıklanarak yapılmasıdır (Görsel 6.34).

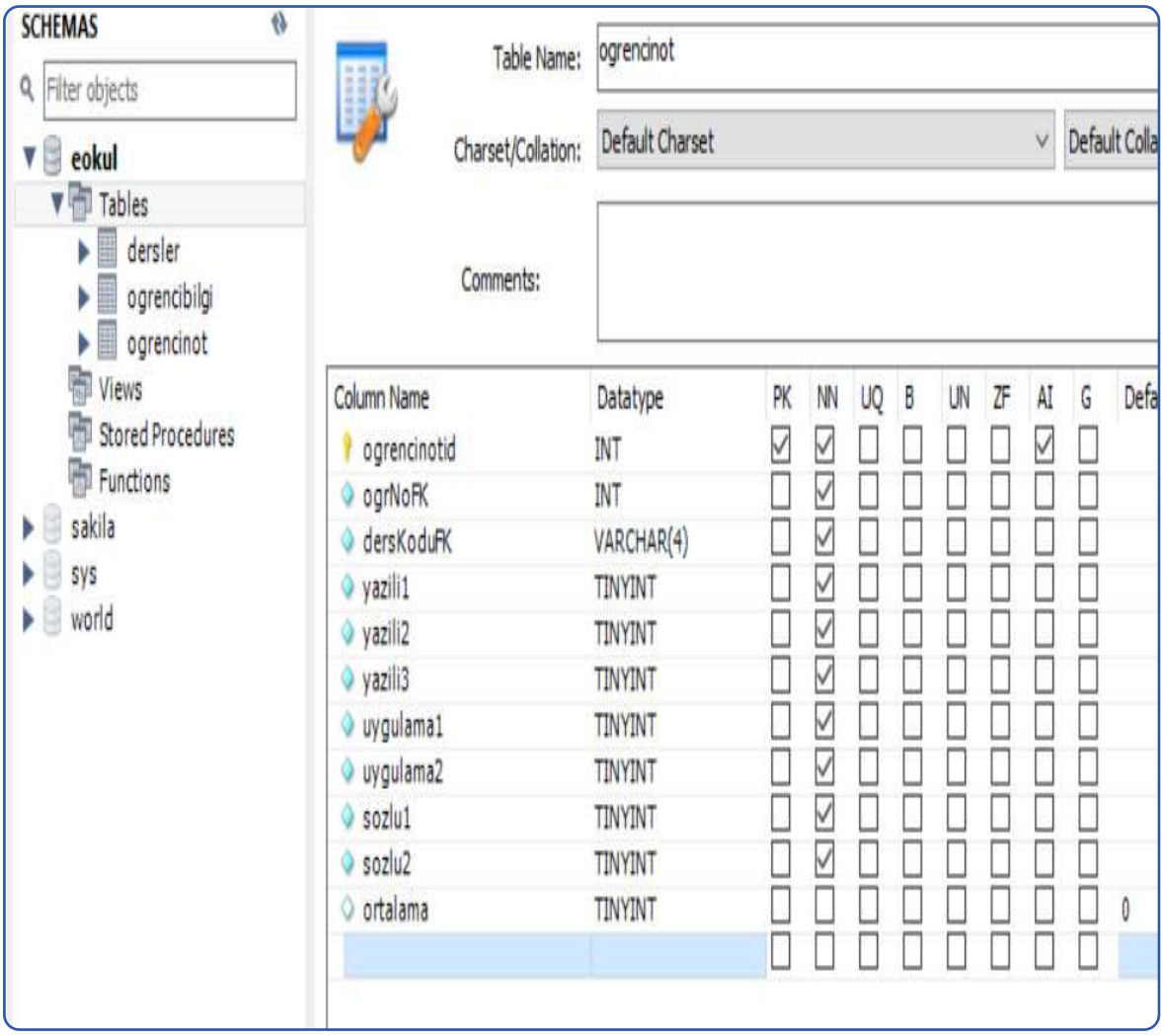


Görsel 6.34: Tablo oluşturma ekranı

Arayüzdeki kısaltmaların açılımları şöyle sıralanabilir:

- PK: Eklenen satırın primary key olup olmamasının seçildiği kutucuktur.
- NN: Not Null, eklenen satırın boş geçilemeyeceğini belirtir.
- BIN: Seçilen satırdaki bilgilerin Binary olarak saklanıp saklanmayacağının seçildiği alandır.
- UN: Unsigned, seçilen alan için sadece pozitif ekleme yapılabileceğini gösteren alandır. Örneğin, bir tinyint alan unsigned olarak işaretlendiğinde 0 ile 255 arasında bir değer alabilir.
- UQ: Seçilen alanın benzersiz olması istenildiğinde kullanılır.
- ZF: Seçilen alanda boşluk kalması hâlinde alanın "0" ile doldurulması istenildiğinde seçilmesi gereken alandır. Örneğin, int(3) alanına 21 bilgisi geldiğinde bu alan 021 olarak eklenecektir.
- G: Bu alanın diğer alanlardan üretildiğini gösterir.
- AI: Auto Increment, bu alandaki değerin bir bir artacağını gösterir. Primary keyler için kullanılır.

Görsel 6.35'teki alanlar incelendiğinde şu bilgilere ulaşılır:



Görsel 6.35: ogrencinot tablosunun son hâli

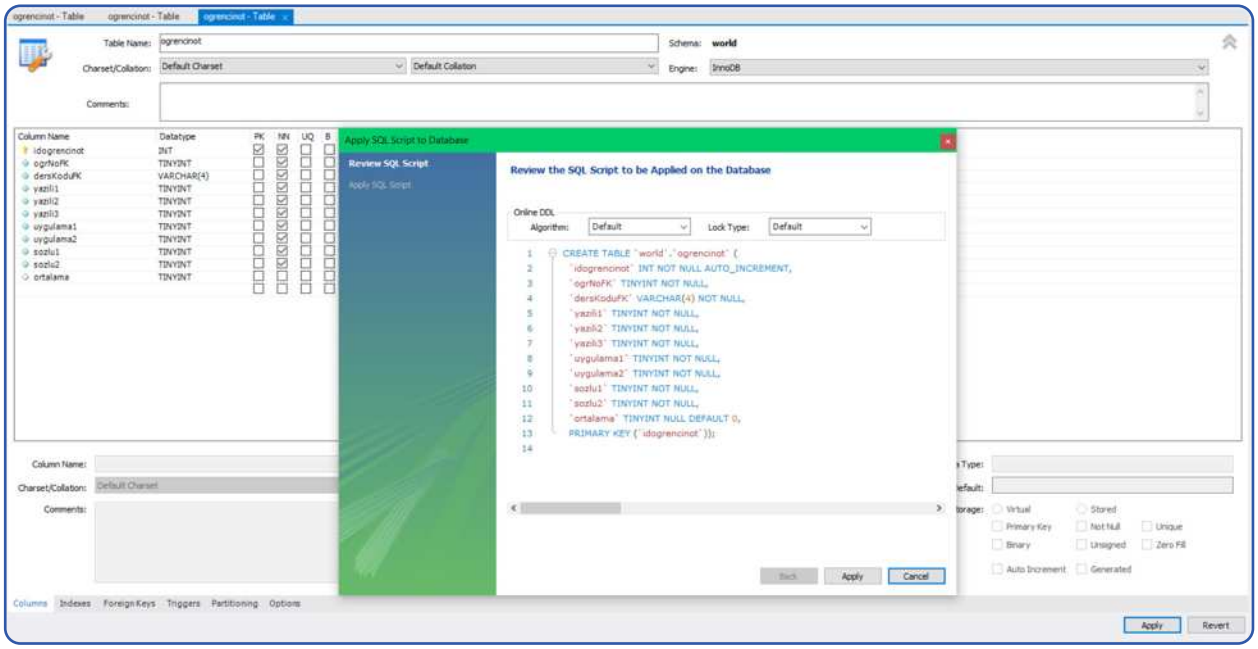
- **ogrencinotid** alanı, bu tablodaki her bir kayıt için benzersiz bir id üretilmesini sağlayacaktır. Buradaki id'ler özel bir anlam ifade etmediği için auto increment alanı da seçilmiştir. Böylece her kayıt eklendiğinde buradaki id alanı otomatik olarak bir artacaktır. ogrencibilgi tablosunda primary key olan öğrenci numarası alanı, kullanıcı gireceği için auto increment yapılmamıştır.
- **ogrNoFK** ilişkisel veri tabanı için ogrencibilgi tablosuyla bağlantı kurulacak alanı temsil etmektedir.

Burası bir Foreign Key alanıdır.

- **dersKoduFK** alanı da dersler tablosu ile bağlantı kurulacak alanı temsil etmektedir. Burası da bir Foreign Key alanıdır.
- Ortalama alanına ise şu an kullanılmamasından dolayı "0" değeri default olarak verilmiştir.
- Diğer alanlar ise not girişi yapılacak alanlardır.

Tablo oluşturulduktan sonra aşağıda yer alan "**Apply**" düğmesine basılır ve gelen SQL komutları incelenir.

Görsel 6.36'daki "**Apply**" düğmesine basıldıktan sonra aşağıdaki mesajın alınması gerekir (Görsel 6.37).



Görsel 6.36: ogrencinot tablosunun SQL komutları

SQL script was successfully applied to the database.

Görsel 6.37: Kodların düzgün çalıştığını belirten mesaj

Bu mesaj gelmediyse "**Output**" penceresinde hatanın sebebi görülebilir. Bu hata internetten araştırılıp çözüm üretilebilir.

6.5. Tabloları İlişkilendirme

İçinde veri olmayan üç adet tablo oluşturuldu. Bu tabloların gerekli alanları birbirine bağlanarak aralarında anlamlı bir ilişki oluşturulmalıdır.

6.5.1. İlişkisel Veri Tabanları

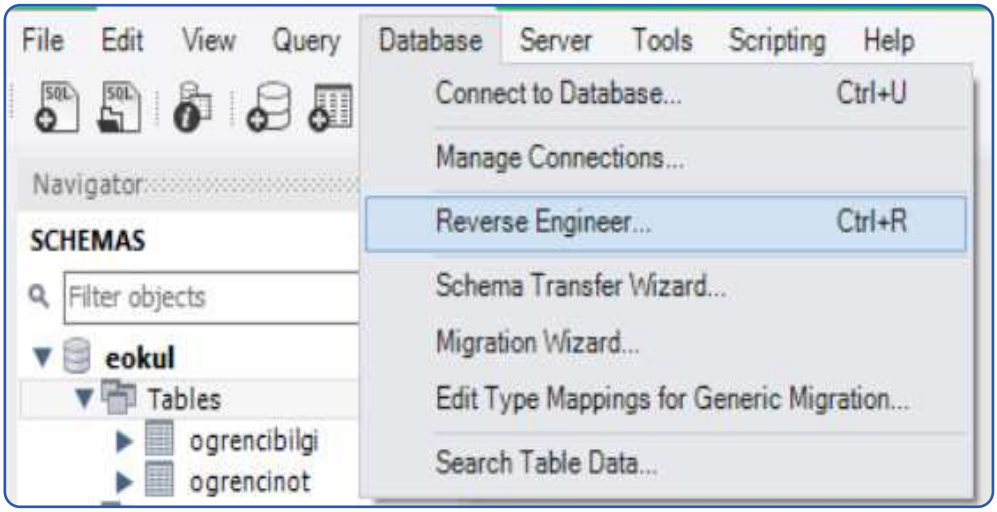
Tüm modern veri tabanı yönetim sistemleri, ilişkisel veri tabanı yönetim modelini kullanarak bilgi saklar ve işler. İlişkisel terimi, veri tabanındaki her kaydın bir tek konu hakkında ve yalnızca o konuyla ilişkili bilgileri içerdiği gerçeğinden ortaya çıkar.

İlişkisel veri tabanı yönetim sisteminde (**RDBMS-Relational Database Management System)** tüm veriler tablolar içinde yönetilir. Ayrıca birden çok tablo ya da sorgudan alınan ilişkili değerler birleştirilebilir. Örneğin; hangi yazarların, hangi kitapları yazdığını öğrenmek için yazar ve kitap bilgileri veya hangi satış elemanının prim hak ettiğini bulmak için çalışan ve sözleşme bilgileri birleştirilebilir.

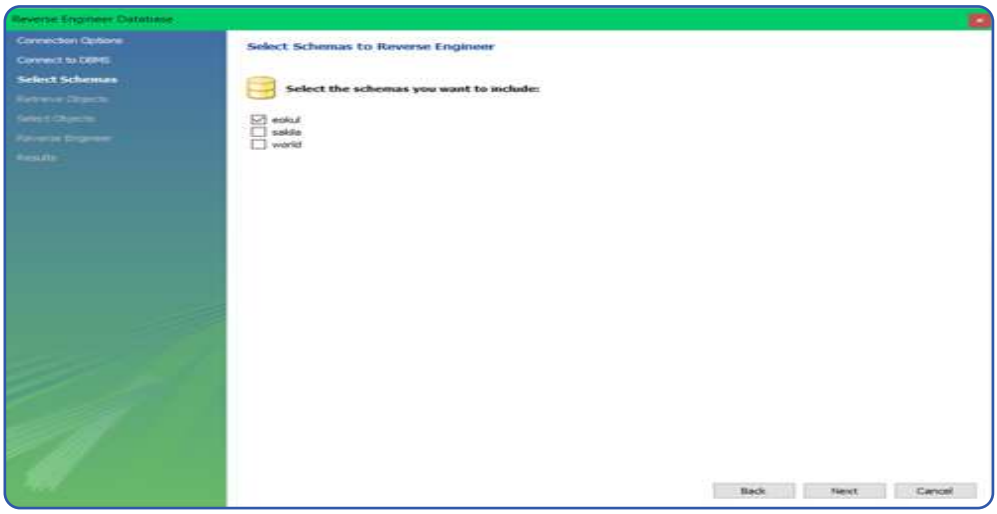
İlişkisel veri tabanı modelinin amacı, yüksek verimliliktir. Örneğin; personel bilgileri bir veri tabanında tutulduğunda personel adı, adresi ve telefonu personelin çalıştığı farklı bölümler için çoğaltılacaktır. Bu gereksiz fazlalık, önemli sayıda kayıtla uğraşan veri tabanları söz konusu olduğunda hem artan disk boşluğu ihtiyacı hem de artan veri ulaşım zamanı sorunlarına yol açar. Veri, ilişkisel modelde iki tabloya ayrılacaktır. Birinci tablo sigorta numarası, kimlik numarası, adres, telefon, yaş gibi kişisel bilgileri içerir. İkinci tablo ise bu bilgileri çoğaltmak yerine personelin çalıştığı bölüm bilgilerini içerir. Bu durum, normalizasyon ile doğrudan ilişkilidir.

- MySQL'de İlişkisel Veri Tabanı Oluşturmak (Reverse Engineer)

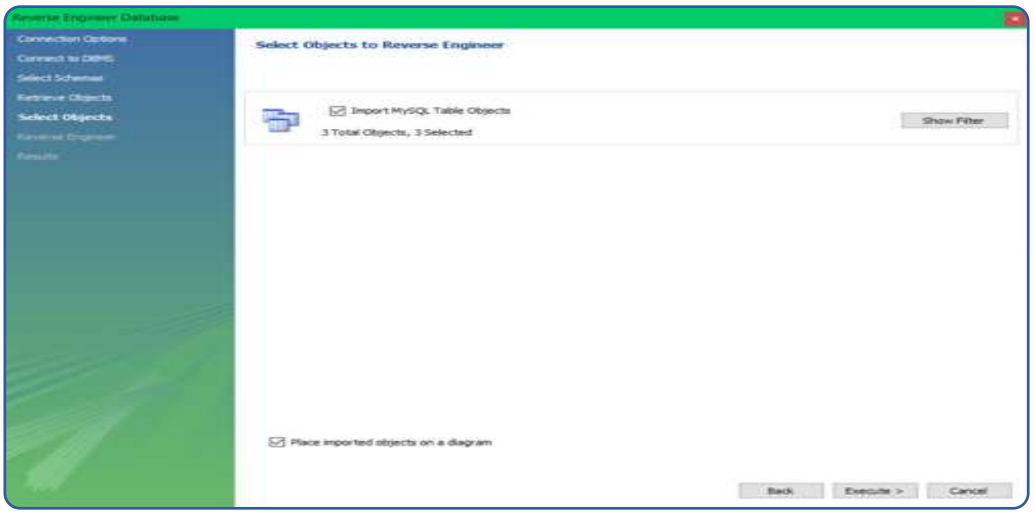
MySQL'de veri tabanları arasında ilişki kurmak için "Database → Reverse Engineer" menüsünden diyagram alanına tabloların aktarımı yapılmalıdır (Görsel 6.38). Görsel 6.39'da eokul veri tabanı seçilmelidir. Görsel 6.40'taki ekranda ise istenen tablolar seçilmelidir.



Görsel 6.38: Reverse Engineer seçimi

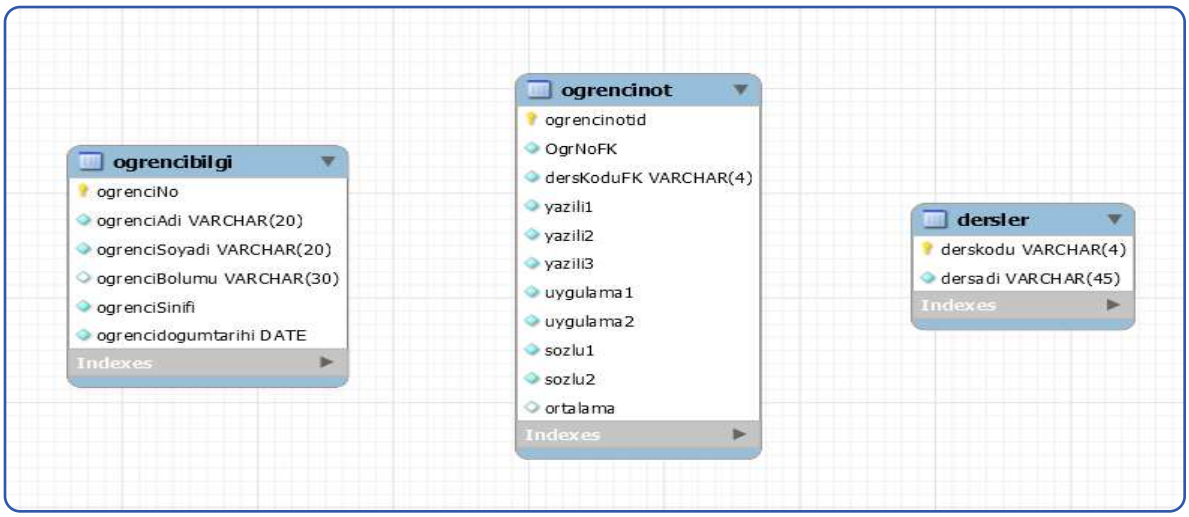


Görsel 6.39: eokul veri tabanı seçimi



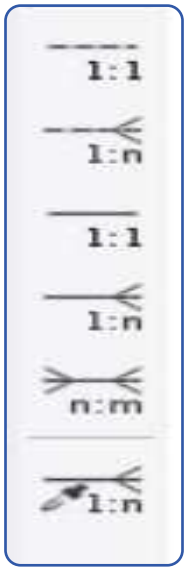
Görsel 6.40: Veri tabanından istenen tabloların seçimi

Görsel 6.41'deki tablolar, diyagram alanına düzgün bir şekilde aktarılmıştır. Tabloların sol tarafında alan ilişki türleri verilmiştir.



Görsel 6.41: Diyagram alanına eklenen tabloların son hâli

Veri tabanları arasındaki bağlantılar; bire çok (one to many), bire bir (one to one), çoka çok (many to many) olmak üzere üç şekilde sağlanmaktadır (Görsel 6.42).



Görsel 6.42: Tablolar arasındaki ilişki türleri

- **Bire Bir Bağlantı Türü (1:1):** Her iki tablodaki bağlantılı alan, her iki tabloda da benzersiz olursa bire bir bağlantı türü kullanılmıştır. Örneğin, öğrenci bilgi tablosundaki öğrenci numarası ile mezuniyet tablosundaki öğrenci numarası arasında 1:1 bağlantı vardır çünkü okuldan o numara ile mezun olabilecek 1 öğrenci vardır ve numara alanları bağlandığında ekrana 1 tane kayıt gelecektir.
- **Bire Çok Bağlantı Türü (1:n):** Bir tabloda primary key olan alanın başka bir tabloda birden çok tekrar etmesi ile ortaya çıkan bağlantı türüdür. Örneğin, öğrenci bilgi tablosundaki öğrenci numarası alanı ile öğrenci not tablosundaki öğrenci numarası alanı ilişkilendirildiğinde bir öğrencinin birden çok ders notu olacağı için öğrenci numarası, öğrenci not tablosunda birden çok tekrarlanacaktır.
- **Çoka Çok Bağlantı Türü (n:m):** İki tablo arasında yapılan bağlantı birden çok kez tekrarlanmışsa bu ilişki türü kullanılmıştır. Örneğin, bir film veri tabanı ve bu veri tabanında film ile kategori tabloları olsun.

Filmin kategorileri ilişkilendirilmek istenildiğinde bir filmin birden çok kategorisi olabileceği için kategori tablosunda hem film id'si birkaç kez görülebilir hem de o kategoride başka filmler bulunabileceği için kategori id'si de aynı tabloda birden çok tekrar edebilir (Görsel 6.43).

film_id	category_id
58	6
58	9
59	3
60	4
61	7
62	6
63	8
64	7
65	11
66	1
66	9
68	3
69	14
70	2

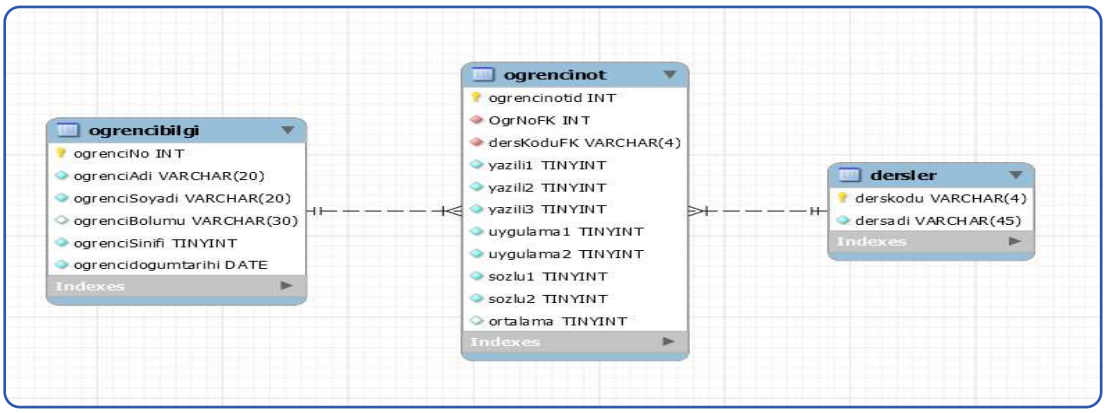
Görsel 6.43: Çoka çok bağlantı türü

Tabloların kesik ve düz çizgi olarak görülmelerinin sebebi, aralarındaki bilginin **tutarlı olup olmamasıyla** ilgilidir. Tablonun bir alanındaki verinin diğer tabloda yer alıp almayacağına karar verilerek bağlantı türü seçilmelidir. Tablonun bir alanındaki verinin diğer tabloda kesinlikle olması gerekiyorsa düz çizgi, verinin diğer tabloda olup olmaması önemli değilse de kesik çizgi seçilmelidir. Örneğin, filmlerle ilgili veri tabanı film tablosunda mutlaka bir yönetmen alanı ve yönetmen bilgilerini tutan bir tane de yönetmen tablosu olacaktır. Tabloya film eklendiğinde yönetmen id eklenmesi istenirse düz çizgili bağlantı, yönetmen bilgisinin sonradan eklenmesinde bir sakınca görülmezse de kesik çizgi bağlantısı seçilir.

6.5.2. Tablolar Arası Bağlantı Yapılması

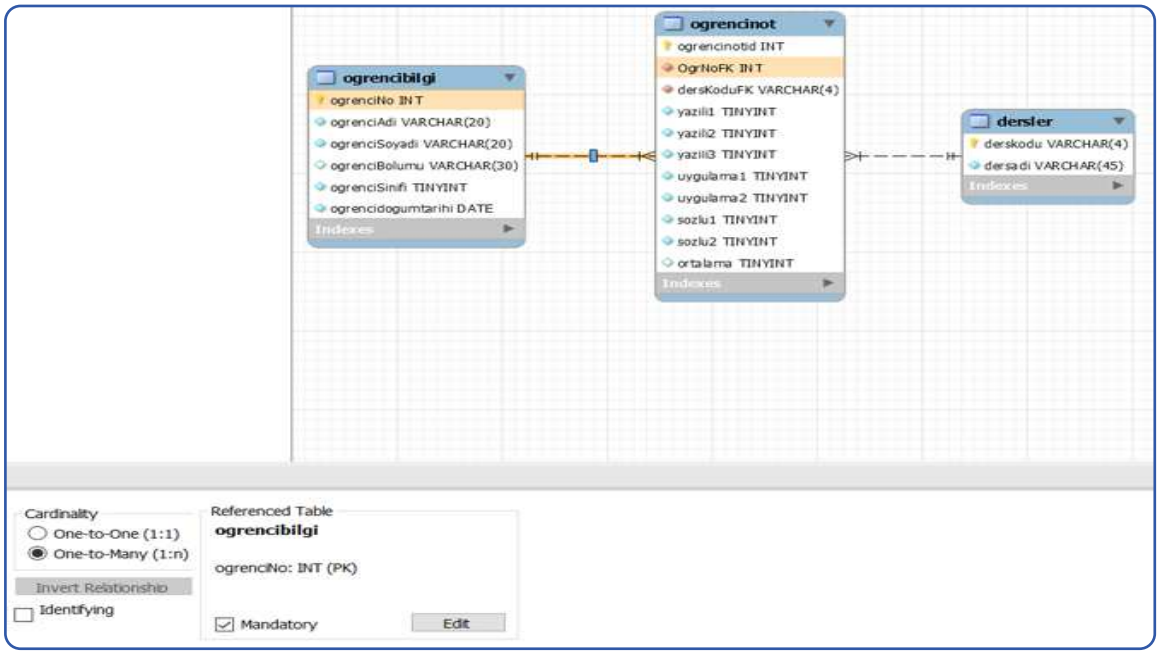
Tablolar hazırlanırken ogrencinot tablosunda gerekli foreign key alanları oluşturulmuştu. O yüzden bağlantı kısmında en altta yer alan ve istenilen alanlar arasındaki bağlantıyı kullanıcının yapmasına olanak sağlayan bire çok  simgesiyle bağlantılar gerçekleştirilir. Bunun için simgeye tıklanarak önce ogrencinot tablosundaki ogrNoFK alanına sonra da ogrencibilgi alanındaki ogrenciNo alanına gelinir. Ardından yine simgeye tıklanarak önce ogrencinot tablosundaki dersKoduFK alanına sonra da dersler tablosundaki derskodu alanına gelinir.

Görsel 6.44'te ogrencibilgi ve dersler tablolarına gelen uçlar tekli, ogrencinot tablosuna gelen uçlar ise çoklu olmuştur. ogrencinot tablosundaki ogrNoFK ve dersKoduFK alanları tekrar edecek şekilde (**Foreign Key**) yapılandırılmış ve ilişkilendirilmiştir.



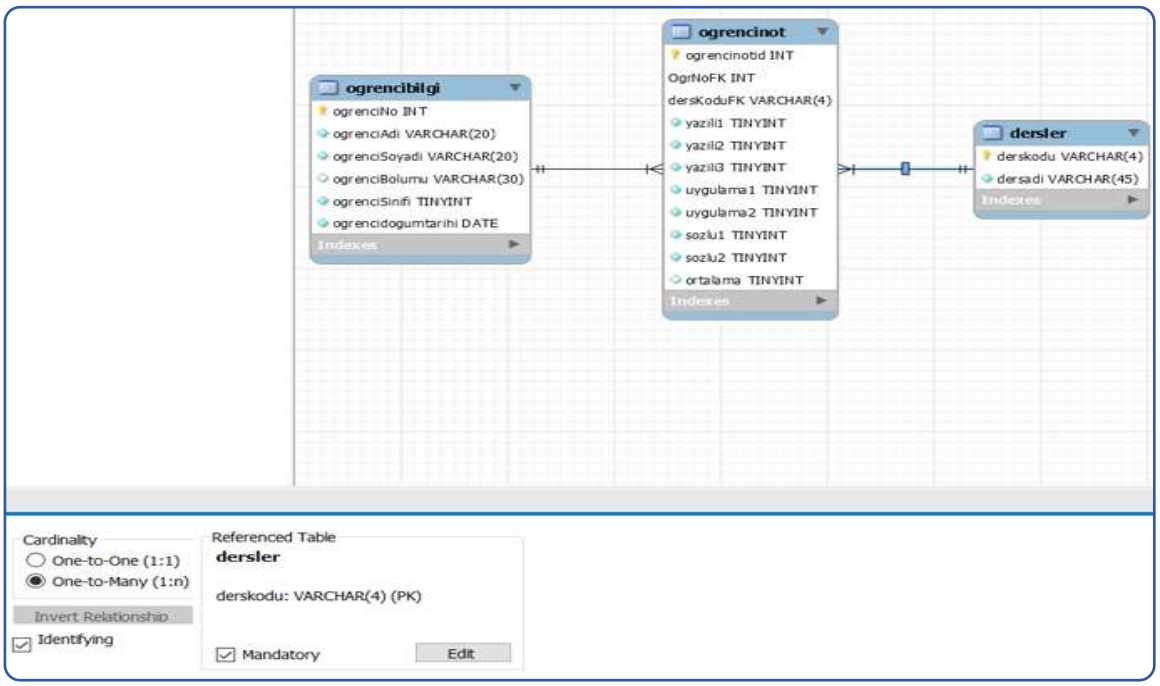
Görsel 6.44: Tablolar arası bağlantı şeması

Bağlantıları tamamen kullanıcı yaptığı için aralarındaki bağlantı kesik çizgi ile oluşturulmuştur. Bu bağlantı türü uygun değildir. Öğrenci numarası ile ders kodu olmadan öğrenci not bilgisi girilemez ve ortalama hesaplanamaz. **Bu yüzden aralarındaki ilişki birbirine bağlantılı yapılmalıdır** (Görsel 6.45).



Görsel 6.45: İlişki türünü değiştirme işlemi

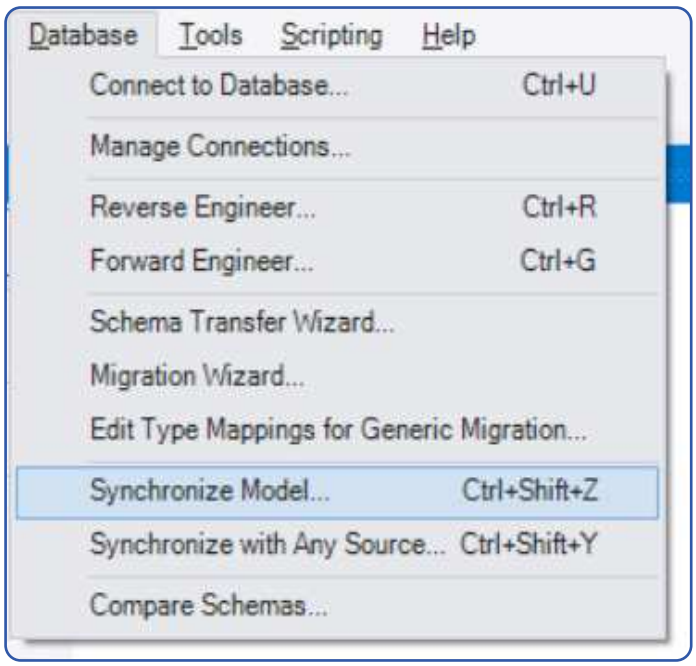
Bunun için oluşturulan bağlantıya çift tıklandıktan sonra aşağı tarafta "**Relationship**" penceresinde "**Foreign Key**" sekmesine geçilir. Daha sonra "**Identifying Relationship**" kutucuğu seçilir. Böylece ogrencibilgi tablosunun ogrencino alanında olmayan bir kayıt, ogrencinot tablosuna eklenemeyecektir. Bu da veri tabanı için istenilen bir durumdur (Görsel 6.46).



Görsel 6.46: Bağlantıların tamamlandığı ekran

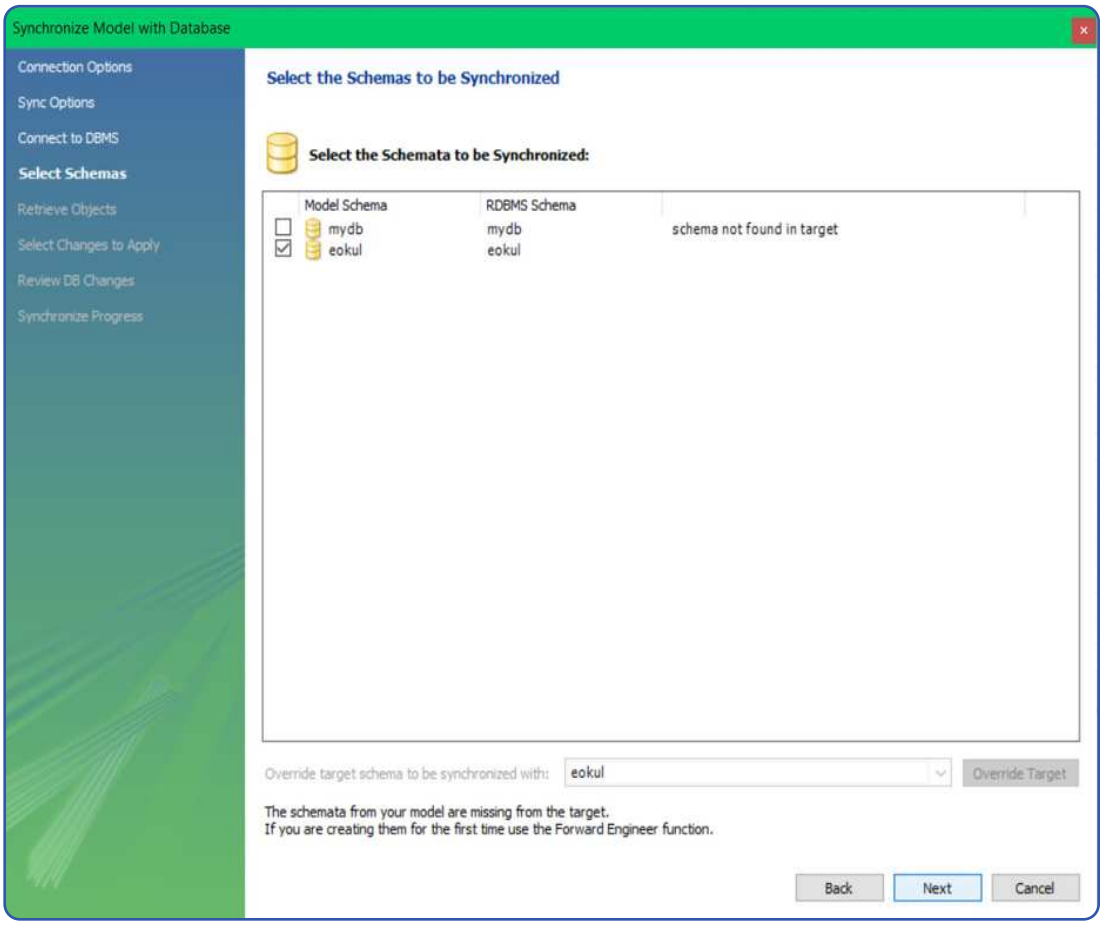
DİKKAT: Bazı durumlarda ekrana veri türleri (INT gibi) gelmemektedir (Görsel 6.41). Gelmemişse tablo başlıklarına çift tıkladıktan sonra aşağıda açılan pencereden tekrar veri türü tanımlanması gerekmektedir.

Yapılan değişiklikleri veri tabanına uygulamak için "**Database**" menüsünden "**Synchronize Model**" seçeneği işaretlenir (Görsel 6.47).



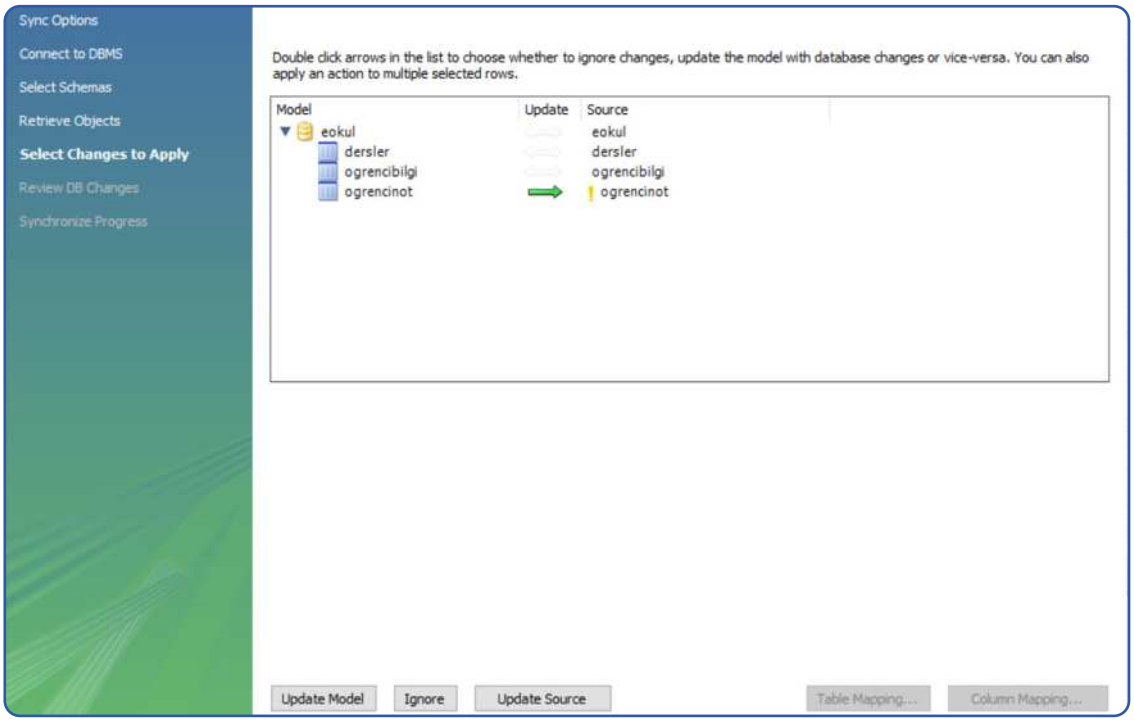
Görsel 6.47: Synchronize Model seçimi

Görsel 6.48'deki ekranda senkronize yapılacak veri tabanı seçilir.

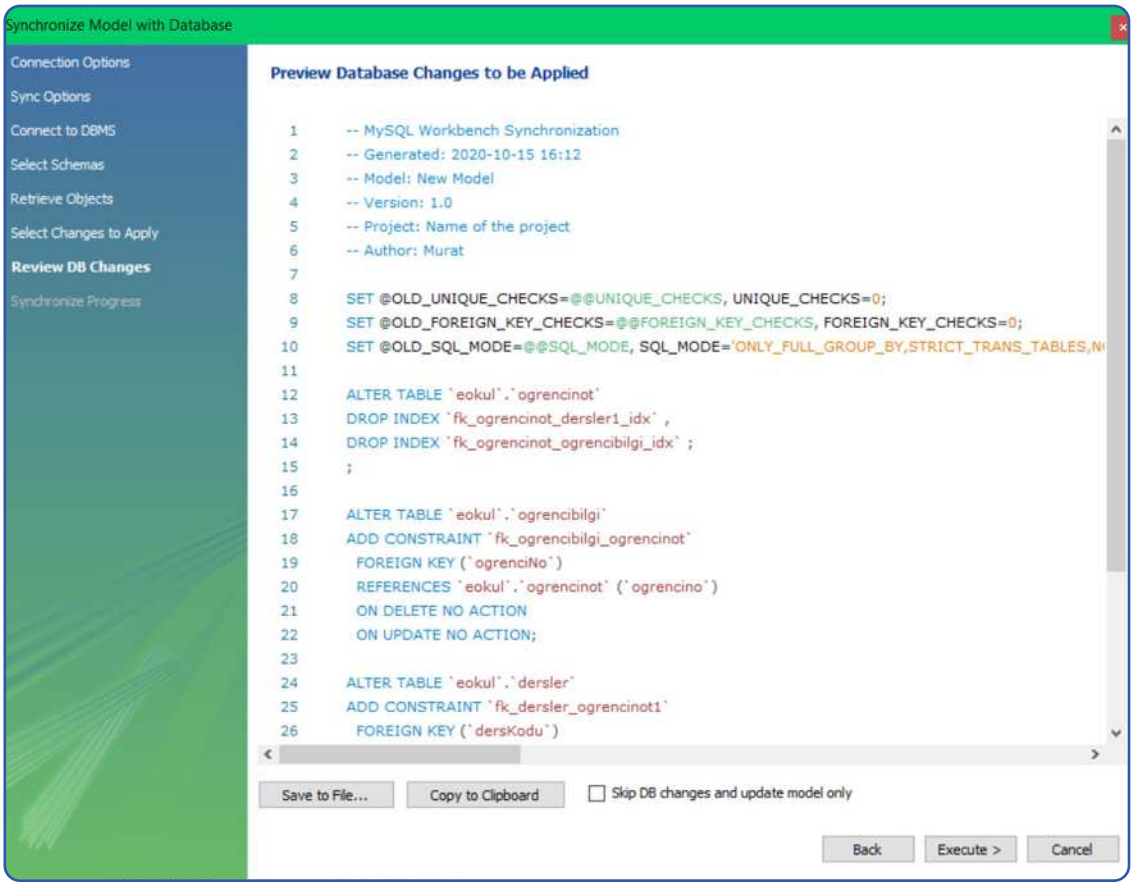


Görsel 6.48: Synchronize Model adımları

Görsel 6.49'daki ekranda yapılacak değişiklikler gösterilir. Görsel 6.50'deki ekranda yapılacak değişiklikler SQL kodlarıyla gösterilir.

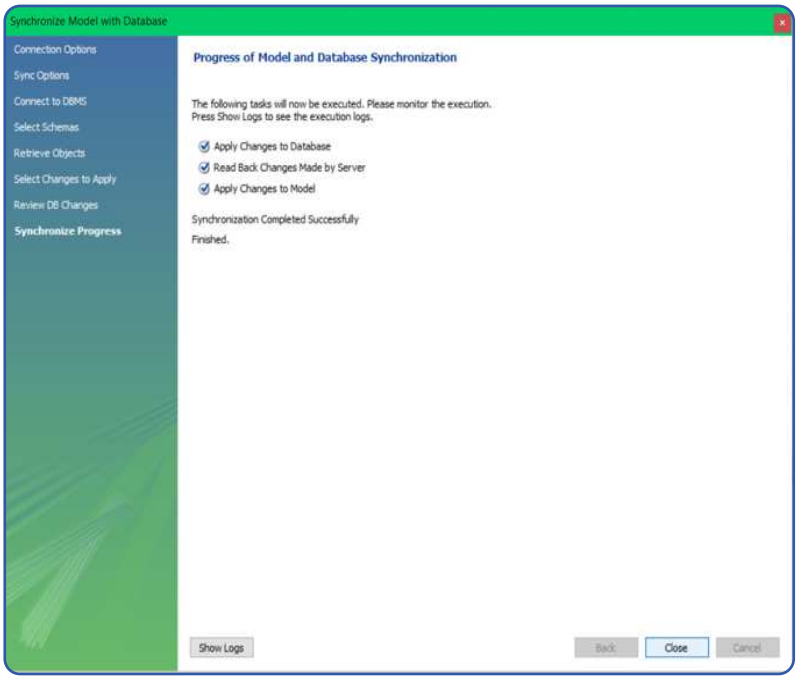


Görsel 6.49: Synchronize Model adımları



Görsel 6.50: Synchronize Model SQL kodları

Görsel 6.51'deki ekran ile işlemlerin düzgün sonlandığı görülür.



Görsel 6.51: Senkronizasyonun sorunsuz uygulanması

6.6. Veri Tabanına Bilgi Girişi

Tablolar oluşturulduğuna ve ilişkileri yapıldığına göre tablolara veri girişine başlanabilir. Tablolara SQL kodları veya MySQL arayüzü kullanılarak iki şekilde veri girişi yapılabilir. Görsel 6.52'de görüldüğü gibi dersler, SQL kodlarıyla dersler tablosuna eklenir:

insert into tabloadı (alan1, alan2, …, alan(n)) VALUES (deger1, deger2, …, deger(n))

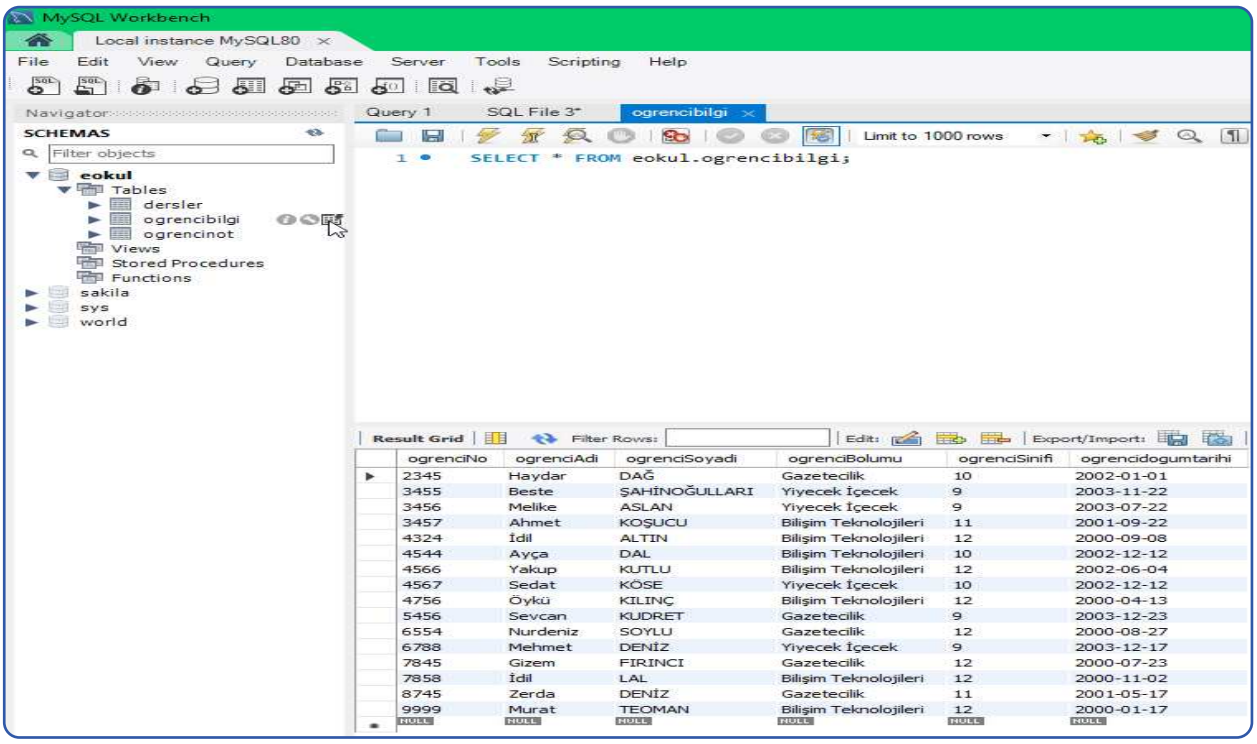
MySQL arayüzü kullanılarak ogrencibilgi tablosuna veri girişi yapmak için tabloya sağ tıklanıp "Select Rows –Limit 100" seçimi yapılır ya da tablonun üzerine gelindiğinde çıkan simgelerden en sağdaki tıklanır (Görsel 6.53)

```
insert into dersler(
derskodu,
dersadi
) values
("BTT", "Bilişim Teknoloji Temelleri"),
("PT", "Programlama Temelleri"),
("NESN", "Nesne Tabanlı Programlama"),
("MAT", "Matematik"),
("EDE", "Edebiyat"),
("FİZ", "Fizik"),
("BDN", "Beden Eğitimi")
```

derskodu	dersadi
BDN	Beden Eğitimi
BTT	Bilişim Teknoloji Temelleri
EDE	Edebiyat
FİZ	Fizik
MAT	Matematik
NESN	Nesne Tabanlı Programlama
PT	Programlama Temelleri
NULL	NULL

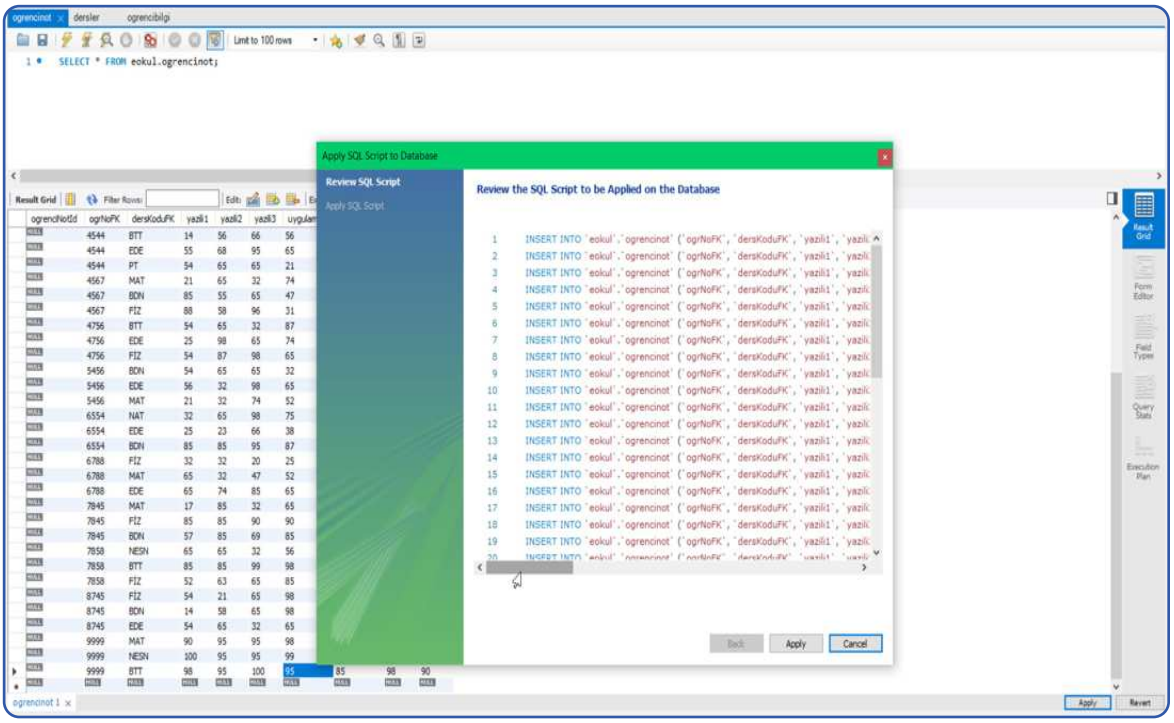
Görsel 6.52: Dersler tablosuna ders ekleme

Görsel 6.53'teki gibi öğrenci numarasını kullanıcı gireceği için auto increment seçilmemesi doğru bir karardır. Veri girişi bittikten sonra aşağıdaki "**Apply**" düğmesine basılır. Bilgiler ogrencinot tablosuna girilerek "**Apply**" düğmesine basılır (Görsel 6.54).



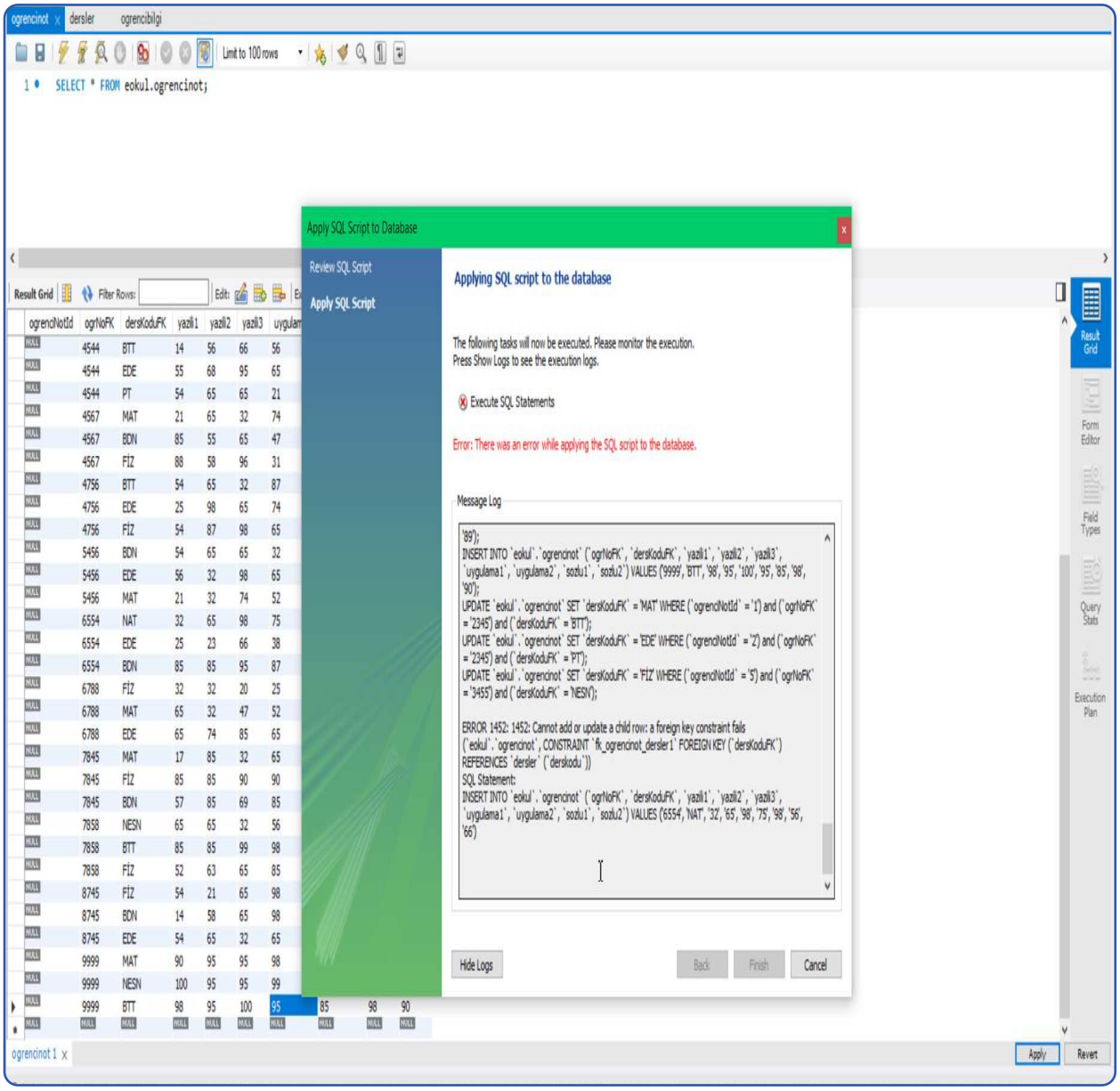
Görsel 6.53: MySQL arayüzü ile veri girişi yapılması

MySQL, kodları **otomatik** olarak oluşturdu. "**Apply**" düğmesine basılarak bir hata yoksa veri girişi tamamlanır. Bu özellik, veri tabanı yönetim sistemlerinin ne kadar önemli olduğunu göstermektedir.



Görsel 6.54: Apply düğmesine basıldıktan sonra gelen ekran

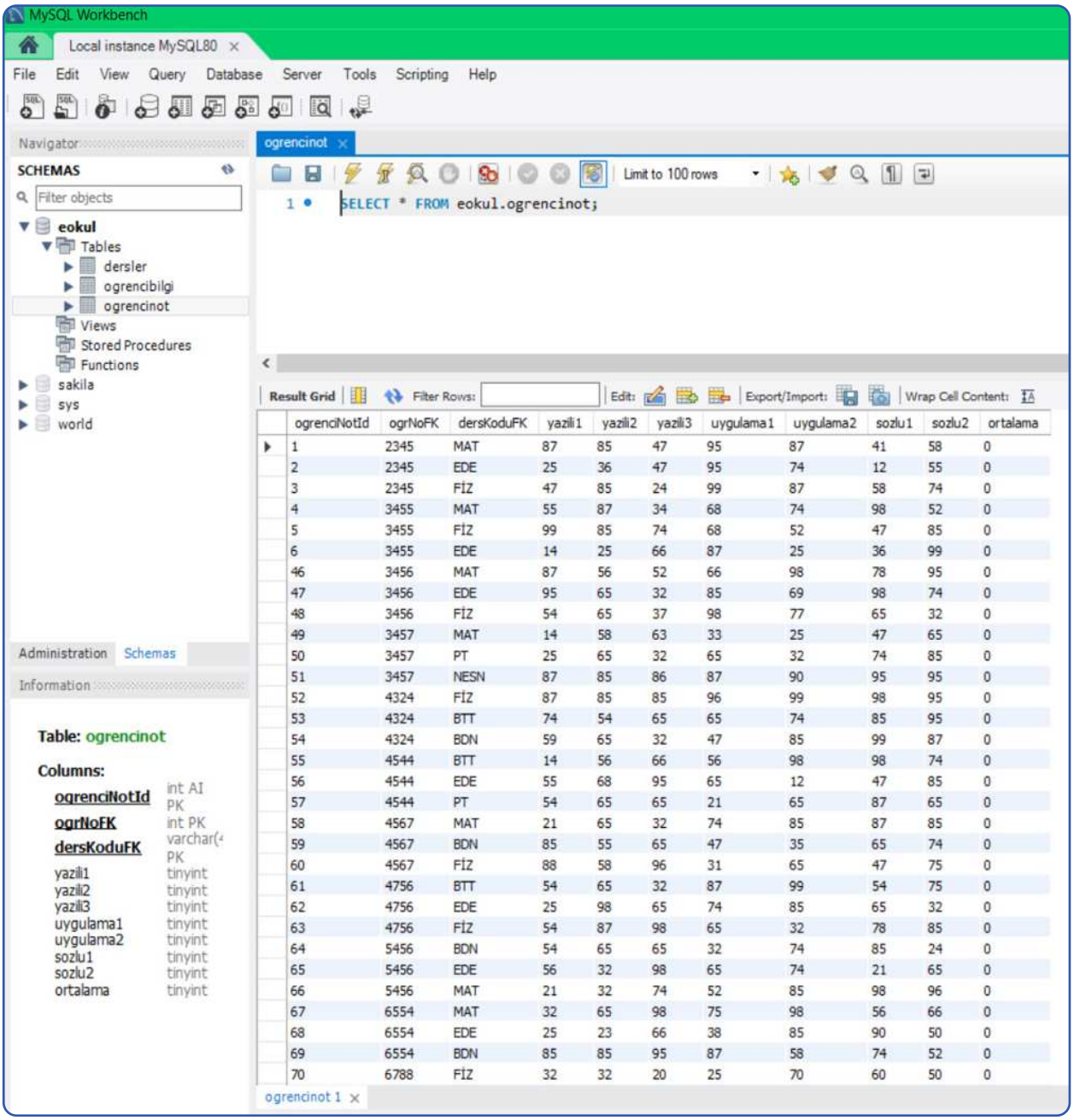
"**Apply**" düğmesine basıldıktan sonra bir hata varsa Görsel 6.55'teki ekran ile karşılaşılır. "**ERROR**" kısmında hatanın sebebi görülebilir ve bu hata düzeltilebilir. Burada gösterilen hata, bir foreign key hatasıdır. Bu da tablolar arasında yapılan ilişkilendirmelerle ilgilidir. İlişkiler oluşturulurken **ogrencinot** tablosundaki **ogrencino** alanı **ogrencibilgi** alanındaki **ogrencino** ile, **ogrencinot** tablosundaki **ders-kodu** alanı da dersler tablosundaki dersler ile ilişkilendirilmiş ve "**Identifying Relationship**" özelliği seçilmişti.



Görsel 6.55: Veri ekleme hatası

Hatanın sebebi budur. Tabloya bakıldığında 6554 numaralı öğrenci için yapılan girişte ders adı "**MAT**" yerine "**NAT**" yazılmıştır. Dersler tablosunda böyle bir kayıt bulunmadığı ve ilişkilendirmeden dolayı dersler tablosunda olmayan bir giriş yapılmaya çalışıldığı için veri tabanı bu girişe izin vermemiştir. Bu ilişkilendirme yapılmasaydı ve "NAT" ders olarak eklenseydi ortalama hesaplanmasına kadar birçok yerde ciddi hatalara yol açabilirdi. Aynı hata, **ogrencibilgi** tablosunda yer alan **ogrencino**'lar dışında bir ogrencino eklenirse de görülür çünkü ogrencibilgi tablosunda yer almayan bir ogrencino'sunun anlamı yoktur. Hata düzeltilerek veri tabanına tekrar kayıt eklenir.

6554 numaralı öğrencinin ders kodu "**MAT**" yapılınca diğer alanlarda hata olmadığı için tabloya sorunsuzca veri eklenmiştir. Tablo incelendiğinde "**ogrenciNotId**" alanının otomatik olarak arttığı görülür. ogrenciNotId'isinin 6'dan 46'ya atlamasının birkaç sebebi bulunabilir. Örneğin, bu aradaki öğrenciler silinmiş olabilir. 46'nın devamında ogrenciNotId'isi bir bir artmaya devam etmiştir. Ortalama alanı ise default değeri olan "0" ile doldurulmuştur Görsel 6.56).



ogrenciNotId	ogrNoFK	dersKoduFK	yazili1	yazili2	yazili3	uygulama1	uygulama2	sozlu1	sozlu2	ortalama
1	2345	MAT	87	85	47	95	87	41	58	0
2	2345	EDE	25	36	47	95	74	12	55	0
3	2345	FİZ	47	85	24	99	87	58	74	0
4	3455	MAT	55	87	34	68	74	98	52	0
5	3455	FİZ	99	85	74	68	52	47	85	0
6	3455	EDE	14	25	66	87	25	36	99	0
46	3456	MAT	87	56	52	66	98	78	95	0
47	3456	EDE	95	65	32	85	69	98	74	0
48	3456	FİZ	54	65	37	98	77	65	32	0
49	3457	MAT	14	58	63	33	25	47	65	0
50	3457	PT	25	65	32	65	32	74	85	0
51	3457	NESN	87	85	86	87	90	95	95	0
52	4324	FİZ	87	85	85	96	99	98	95	0
53	4324	BTT	74	54	65	65	74	85	95	0
54	4324	BDN	59	65	32	47	85	99	87	0
55	4544	BTT	14	56	66	56	98	98	74	0
56	4544	EDE	55	68	95	65	12	47	85	0
57	4544	PT	54	65	65	21	65	87	65	0
58	4567	MAT	21	65	32	74	85	87	85	0
59	4567	BDN	85	55	65	47	35	65	74	0
60	4567	FİZ	88	58	96	31	65	47	75	0
61	4756	BTT	54	65	32	87	99	54	75	0
62	4756	EDE	25	98	65	74	85	65	32	0
63	4756	FİZ	54	87	98	65	32	78	85	0
64	5456	BDN	54	65	65	32	74	85	24	0
65	5456	EDE	56	32	98	65	74	21	65	0
66	5456	MAT	21	32	74	52	85	98	96	0
67	6554	MAT	32	65	98	75	98	56	66	0
68	6554	EDE	25	23	66	38	85	90	50	0
69	6554	BDN	85	85	95	87	58	74	52	0
70	6788	FİZ	32	32	20	25	70	60	50	0

Görsel 6.56: Hatasız veri ekleme ekranı ve mesajı

Sıra Sizde

Siz de 15 öğrenci için 3 dersten birer not girişi yapınız. ogrencinot tablonuzda 45 kayıt olsun. İçinde veri bulunan üç tablolu bir veri tabanına sahipsiniz. En çok kullanılan SQL komutlarını tablolarınıza uygulayınız.

6.7. SQL Komutları Kullanımı

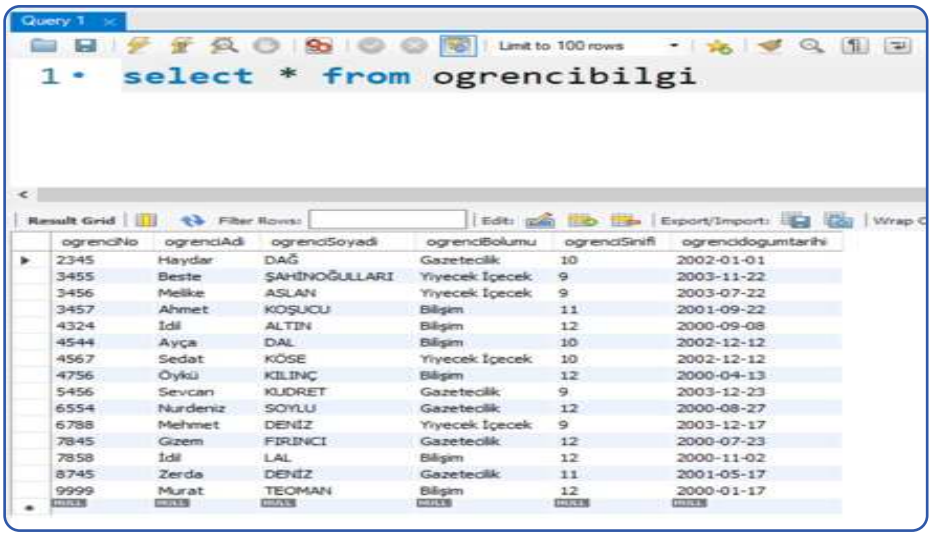
Önce  simgesinden yeni bir SQL sayfası açılır. Tablo isimleri belirtilirken önceden oluşturulan tablo isimleri kullanılır. Her bir SQL sorgusu aksi belirtilmedikçe boş bir SQL sayfasında çalıştırılır.

Aynı sayfada birden çok SQL komutu hata almaya neden olabilir.  soldaki simge bütün komutları çalıştırmak için, sağdaki simge ise sadece imlecin bulunduğu kodu çalıştırmak içindir. Tıpkı kod editöründe olduğu gibi otomatik kod tamamlama özelliği vardır. SQL kodları yazılırken String'ler için tek tırnak işareti kullanılır.

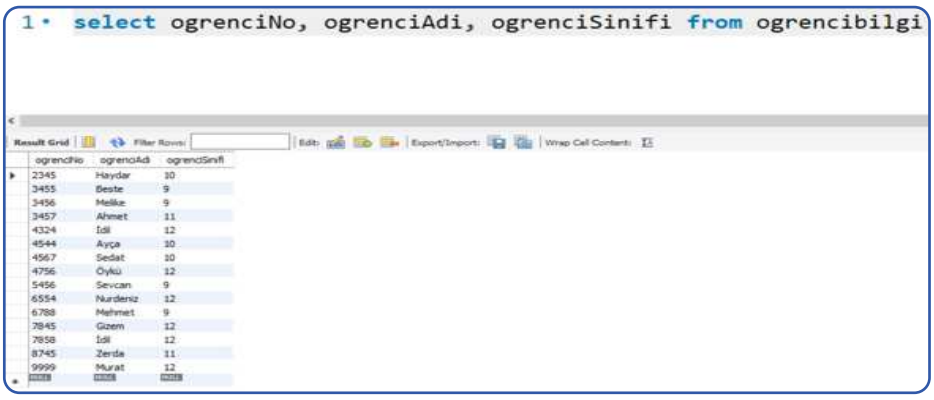
6.7.1. Select Deyimi

SELECT {*, SÜTUN, ...} FROM tabloAdı

SQL' de sorgulama işlemleri Select deyimi kullanılarak yerine getirilir. Tablonun tüm sütunlarını seçmek için * (yıldız) kullanılır. Tabloda istenilen sütunların seçilmesi için sütunların isimleri belirtilmelidir.ogrenci tablosundaki tüm sütunları seçmek için Select deyiminin kullanımı Görsel 6.57'de verilmiştir. Tablodaki isimler rastgele seçilmiştir. Görsel 6.58'de ogrenci tablosundan sadece öğrenci no, ad ve sınıf bilgilerini seçmek için **Select** deyiminin kullanımı verilmiştir.

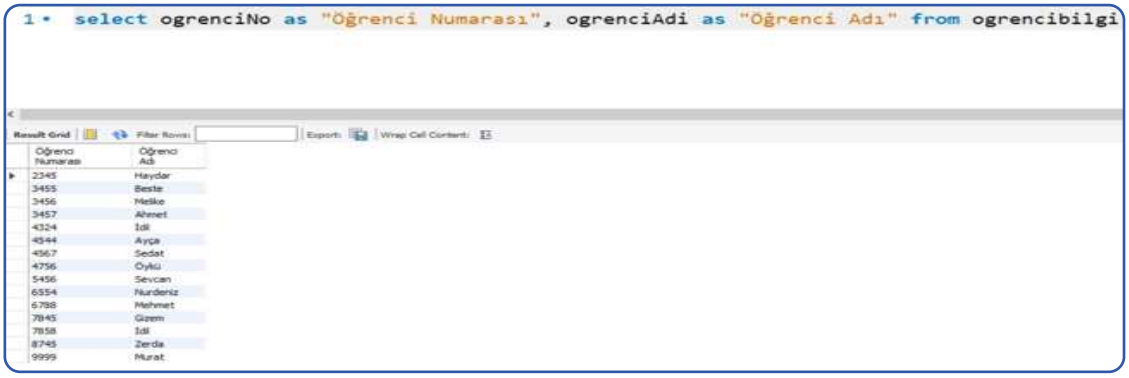


Görsel 6.57: Select * From sorgusunun çalıştırılması



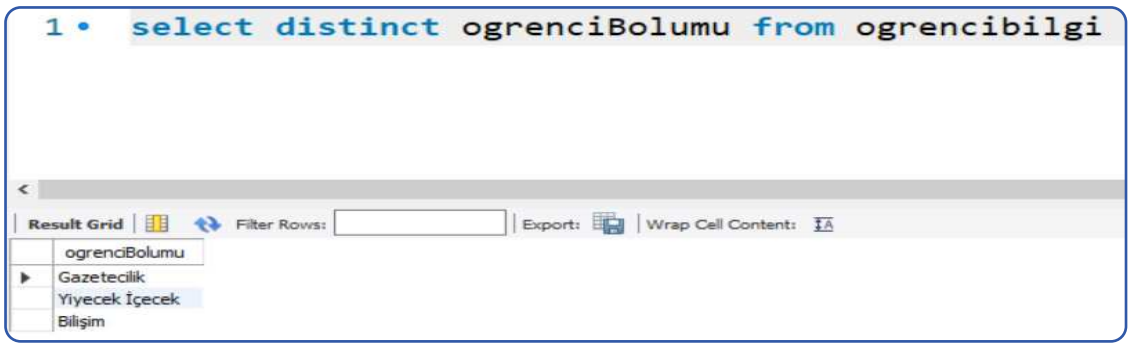
Görsel 6.58: Select sorgusu ile belirli alanları çağırma

Sütunlarda takma isim kullanılması ve başlıkların alan adları dışında bir isimle görüntülenmesi için AS anahtar kelimesi kullanılır (Görsel 6.59).



Görsel 6.59: Select sorgusu ile çağrılan alanları adlandırma

Select deyimi, tekrarlı satırların yalnızca bir tanesinin listelenmesi isteniyorsa **Distinct** anahtar kelimesi ile birlikte kullanılır. Aynı bölüm isimlerinden sadece birer tanesinin listelenmesi istenirse Görsel 6.60'taki gibi bir ekranla karşılaşılır.

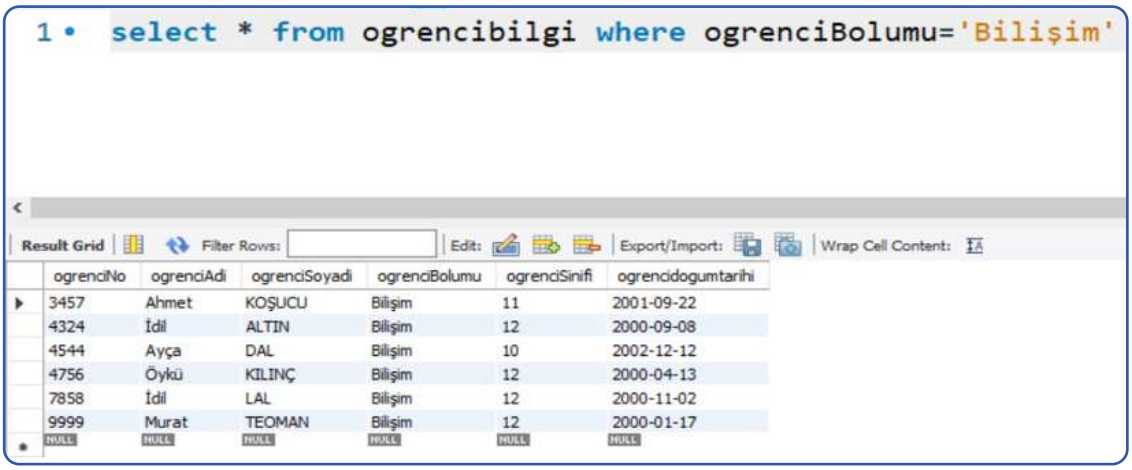


Görsel 6.60: Distinct kullanımı

6.7.2. Where Şart İfadesi

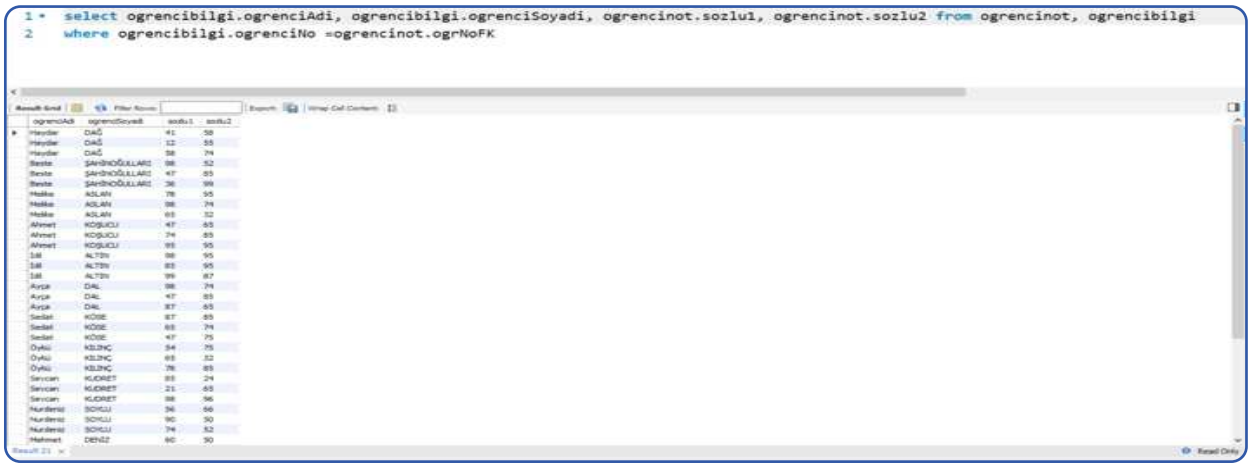
Belirli bir koşulu sağlayan kayıtların süzülmesi için Where şart ifadesi kullanılır. Bütün SQL komutları içinde belki de en önemlisi ve çok kullanılanı bu ifadedir. Where ile yapılacak işlemlerin sadece belirtilen değere göre olması sağlanabilir. **Where ile bir filtreleme yapılmadığı takdirde güncelleme, silme vb. işlemler bütün tabloya uygulanacaktır.**

Görsel 6.61'de bölümü "**Bilişim**" olan öğrencileri listelemek için **Where** kullanılmıştır.



Görsel 6.61: Where kullanımı

Sütun isimleri, özellikle iki tablo ile çalışıldığında tablo ismiyle birlikte kullanılır (Görsel 6.62).



Görsel 6.62: Where kullanarak ilişkili tablolardan veri çekme

Buradaki Select sorgusu, şu ana kadar yapılan her şeyin bir sonucudur. SQL sorgusuna bakıldığında tablo adları belirtilerek (ogrencibilgi ve ogrencinot) iki tablodan görüntülenmek istenilen alanlar çağrılmış ve bu iki tablodaki alanlar Where kısmında öğrenci numarasının eşit olmasına göre listelenmiştir. ogrencinot tablosunda öğrenci adı ve soyadı yerine öğrenci numarası mevcuttur. Daha sonra diyagramlarla tablolar öğrenci numarası üzerinden bağlanmıştır. Burada öğrenci numarası eşitliğinden öğrenci adının ve soyadının görünmesi **bu ilişkilendirmeden** kaynaklanmaktadır. ogrencinot tablosunda yer alan ogrNoFK değeri, ogrencibilgi tablosuna gidip okunan numaradaki değere karşılık gelen satırdan ad ve soyad bilgisini alıp gelmektedir.

6.7.3. Karşılaştırma Operatörleri

Karşılaştırma operatörleri iki değer arasında karşılaştırma yapmak için kullanılan operatörlerdir. Tablolardan gelen değer, sorgulara Where eklenirken bir kıstasa göre filtrelenmelidir. Karşılaştırma operatörleri istenilen verilere ulaşılmasını sağlar.

Tablo 6.2: Karşılaştırma Operatörleri

= Eşit	> Büyük	< Küçük
>= Büyük eşit	<= Küçük eşit	< > Eşit değil

Örnekler:

```
Select ogrenciNo, ogrenciAdi, ogrenciSoyadi from ogrencibilgi Where ogrenciNo=9999;
Select * from ogrencibilgi Where ogrenciBolumu <> 'Bilişim Teknolojileri';
Select ogrenciNo, ogrenciAdi, ogrenciSoyadi from ogrencibilgi Where ogrenciSinifi>10;
```

6.7.4. Mantıksal Operatörler

Birden çok karşılaştırma operatörünün kullanılması durumunda mantıksal operatörler kullanılır.

AND (ve): Her iki şartı da sağlıyorsa true döner.

OR (veya): Şartlardan herhangi birini sağlıyorsa true döner.

Örnekler:

```
Select * from ogrencibilgi Where ogrenciBolumu='Gazetecilik' or ogrenciSinifi>=11;
Select * from ogrencinot Where dersKoduFK='MAT' and ortalama>=50.0;
```

6.7.5. Arama Operatörü

LIKE operatörü, karakter grubu içinde arama yapmak için kullanılır.

Öğrenci adı "Murat" olan kayıtlar:

SELECT * FROM ogrencibilgi WHERE ogrenciAdi LIKE 'Murat';

Öğrenci soyadı "S" ile başlayan kayıtlar:

SELECT * FROM ogrencibilgi WHERE ogrenciSoyadi LIKE 'S%';

Ders adında "temel" geçen kayıtlar:

SELECT * FROM dersler WHERE dersadi LIKE '%temel%';

Adının ikinci harfi "n" olan öğrenciler:

SELECT * FROM ogrencibilgi WHERE ogrenciAdi LIKE '_n%';

6.7.6. Order By Komutu (Sıralama)

Sıralama işlemi için Select ifadesinde **ORDER BY** kullanılır. Bu ifadede ASC kelimesi kullanılırsa sıralama küçükten büyüğe doğru (A-Z), DESC kullanılırsa sıralama büyükten küçüğe doğru (Z-A) yapılır.

Not: Her ikisi de kullanılmazsa sıralama küçükten büyüğe doğru yapılır.

Öğrencileri numarasına göre sıralayan SQL ifadesi:
Select * from ogrenci Order By ogrno ASC
Select * from ogrenci Order By ogrno
Öğrencileri not ortalamasına göre (büyükten küçüğe) sıralama:
Select ogrno, ad, soyad from ogrenci Order By ortalama DESC

Sıra Sizde

Aşağıdaki 5 satıra kendi oluşturduğunuz **Select** ve **Where** içeren sorguları yazınız.

1.	
2.	
3.	
4.	
5.	

6.7.7. Insert Into Komutu (Kayıt Ekleme)

Tabloya veri ekleme işlemi INSERT INTO komutu ile gerçekleştirilir (Görsel 6.63).

insert into tabloadı (alan1, alan2, ..., alan(n)) VALUES (deger1, deger2, ..., deger(n))

```
insert into dersler(
derskodu,
dersadi
) values
("BTT", "Bilişim Teknoloji Temelleri"),
("PT", "Programlama Temelleri"),
("NESN", "Nesne Tabanlı Programlama"),
("MAT", "Matematik"),
("EDE", "Edebiyat"),
("FİZ", "Fizik"),
("BDN", "Beden Eğitimi")
```

Görsel 6.63: Insert into komutu

6.7.8. Update Komutu (Kayıt Güncelleme)

Tablodaki verileri güncelleme işlemi UPDATE SQL deyimiyle gerçekleştirilir (Görsel 6.64).

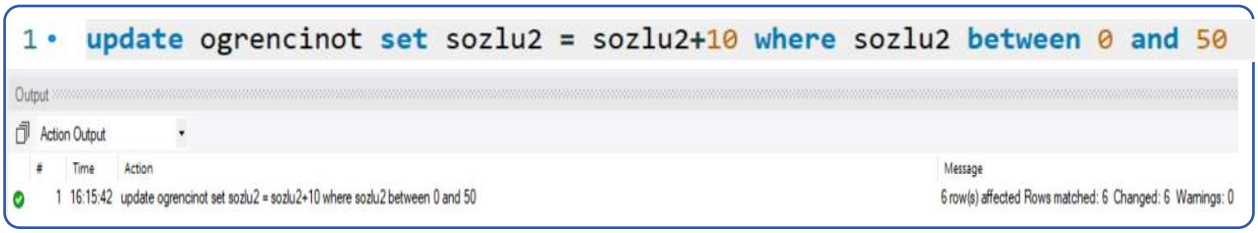
UPDATE tablo **SET** sütun1=değer1, sütun2=değer2,...**WHERE** şart

```
1    update ogrencinot set sozlu1 = sozlu1+5
```

Görsel 6.64: Tüm öğrencilerin sözlüsüne 5 puan ekleyen SQL ifadesi

MySQL, "**Where**" ifadesi primary key alanını sınırlandırmadığından güvenlik sebebiyle buna izin vermeyecektir çünkü burada yapılacak bir hata, 10.000 tane kayıt olsa bile bütün veri tabanını etkileyecektir. Bu bir güvenlik önlemidir. Bu güvenlik önlemi "Edit → Preferences → SQL Editor → Safe Updates" seçeneği kapatılarak kaldırılabilir.

Görsel 6.65'te ogrencinot tablosunda sozlu2 notu 0 ile 50 arasında olan öğrencilere 10 puan ekleyen SQL ifadesi verilmiştir.



Görsel 6.65: Update sorgusu kullanımı

Görüldüğü gibi 6 kayıt güncellenmiştir.

Görsel 6.66'da ogrencibilgi tablosunda "Bilişim" alanını "Bilişim Teknolojileri" yapan SQL kodu verilmiştir.

```
1    update ogrencibilgi set ogrenciBolumu='Bilişim Teknolojileri' where ogrenciBolumu='Bilişim'
```

Görsel 6.66: Update kullanımı

6.7.9. Delete Komutu (Kayıt Silme)

Tablodan satır silme işlemi **DELETE** SQL deyimiyle gerçekleştirilir.

DELETE FROM tablo WHERE şart

Öğrenci tablosunda e-posta adresi boş (Null) olan kayıtları silmek için kullanılan ifadeler aşağıda verilmiştir.

```
DELETE FROM ogrenci
WHERE eposta IS NULL
```

Not: Silinecek satırın başka bir tabloyla yapılmış ilişkilendirmede yer almaması gerekmektedir yoksa hata mesajı alınır. Tablolar arası ilişkilendirme yapılırken bir tabloda olan kaydın diğer tabloya eklenemeyeceği belirtilmiştir (Bağlantı düz çizgi hâline getirilmiştir.). Başka tabloda yer alan bir kayıt silinmek istendiğinde diğer tablodaki ogrencibilgi tablosundan türetilen bilgiler boşa çıkacaktır. Kayıtlar ogrencinot tablosuna ogrencibilgi tablosundaki ogrencino'ya göre eklenmiştir. O yüzden önce ogrencinot tablosundan istenilen numaradaki kayıtlar silinmelidir.

6.7.10. Create

Veri tabanı ya da tablo oluşturmak için kullanılır.
CREATE DATABASE eokul
CREATE TABLE ogrencibilgi

6.7.11. Alter

Var olan bir nesne üzerinde değişiklikler yapmak için kullanılır. Veri alanı eklemek için aşağıdaki kod kullanılabilir.

ALTER TABLE personel **ADD** eposta VARCHAR(40) NULL

Tablolarda bir eksiklik görüldüğünde veri türleri de değiştirilebilir. Örneğin, ogrencinot tablosundaki ortalama alanının veri türü tinyint seçilmiştir fakat bu bir tam sayı değişkeni türüdür. Ortalama hesaplandığında büyük ihtimalle ondalık bir sayı olarak gelecektir. Bu ufak sorun Görsel 6.67'deki sorgu yazılarak çözülür.

```
ALTER TABLE ogrencinot MODIFY ortalama DECIMAL (5,2) DEFAULT 0;
```

Görsel 6.67: Veri türü değiştirme sorgusu

Modify komutuyla ogrencinot tablosundaki ortalama alanı, ondalık bir veri türü olan Decimal ile değiştirildi. 5 rakamı, toplam uzunluğu ve 2 rakamı ise virgülden sonraki rakamları temsil etmektedir. Aradaki değişim ogrencinot tablosundan gözlemlenebilir.

Sıra Sizde

Neden DECIMAL (5,2) ve DEFAULT kullandığınızı açıklayarak arkadaşlarınızla fikirlerinizi paylaşınız.

6.7.12. Drop

Veri tabanındaki herhangi bir nesneyi (tablo veya veri tabanı) kaldırmak için kullanılır. DROP kullanırken çok dikkatli olunmalıdır.

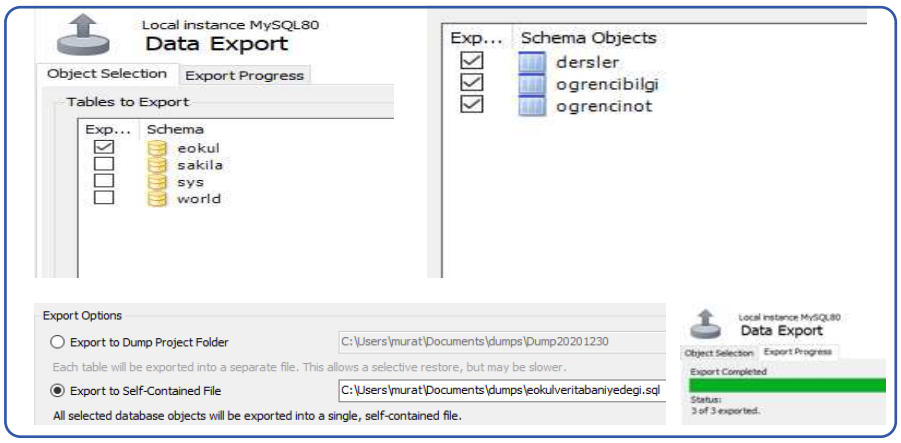
DROP TABLE dersler

6.8. MySQL Veri Tabanı Alma ve Yükleme

Uzun süren projelerde veri tabanlarının taşınması gerekebilir. Proje uzun süreceği için MySQL üzerinden veri tabanının nasıl alınacağı ve MySQL'e nasıl geri yükleneceği çok önemlidir.

Önce Görsel 6.68'deki **Navigator** penceresinin altındaki **Administration** sekmesine geçilir.

Görsel 6.68: Navigator penceresi

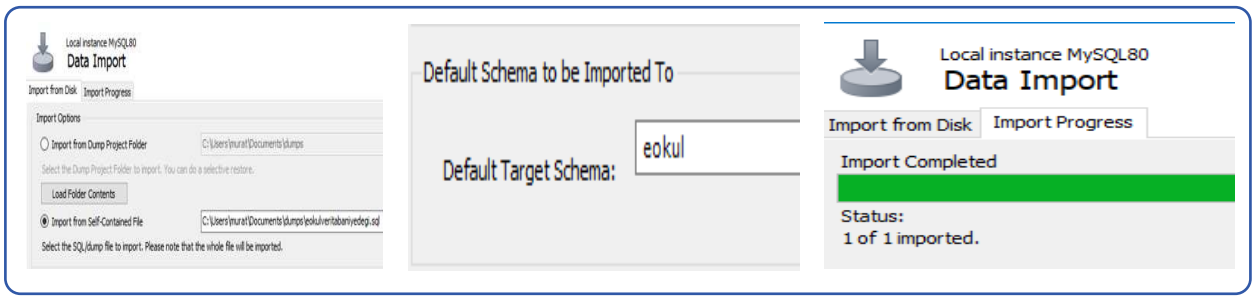


Görsel 6.69: Data Export penceresi

Daha sonra bu pencerede Data Export'a tıklanır.

Gelen pencerede yedeği alınacak veri tabanı seçilir ve alttaki "**Export to Self-Contained File**" seçeneği işaretlendikten sonra yedeğe bir isim verilir. Daha sonra sağ alt taraftaki "**Start Export**" ile veri tabanının yedeği alınır. Veri tabanının yedeği, belirlenen isim ile "**Belgelerim/dumps**" içine aktarılır (Görsel 6.69). Yedeklenen veri tabanını geri yüklemek için **Administration** içindeki **Data Import / Restore** seçeneği işaretlenir. Gelen pencerede "**Import From Self-Contained File**" kısmından yedek seçilir.

Yedeği yüklenecek veri tabanı hazır ise aşağıdaki "**Default Target Schema:**" kısmından seçilir. Hazırda yedeği yüklenecek bir veri tabanı yoksa sağ taraftaki "**New**" düğmesi ile yeni ve boş bir veri tabanı oluşturularak "**Start Import**" ile yedek yüklenebilir (Görsel 6.70).



Görsel 6.70: Data Import penceresi

6.9. SQL ve NTP Bağlantısı

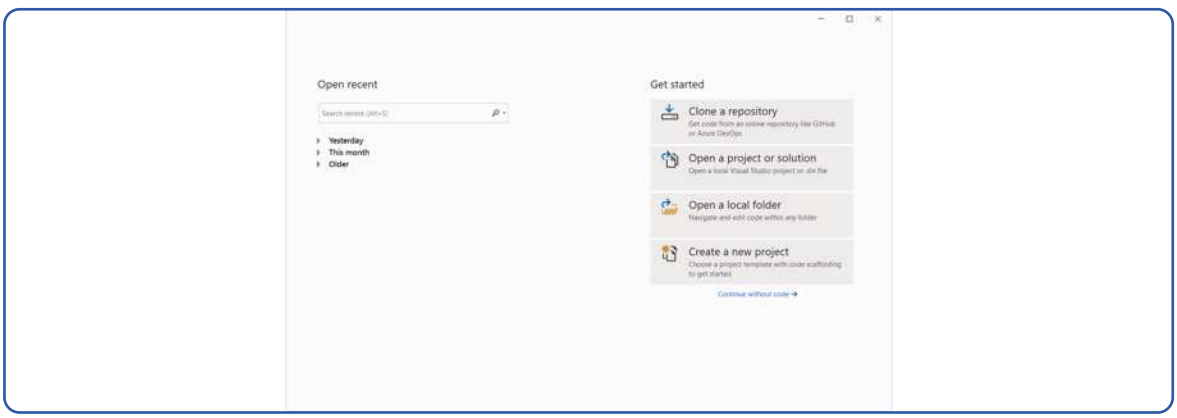
Veri tabanı, içindeki verilerle beraber kullanıma hazır hâle geldi fakat bu kayıtları insanlara veri tabanı arayüzü ile vermek mümkün değildir. E-okul düşünüldüğünde kullanıcılar için hazırlanmış bir arayüze sahip olduğu ve kişilere ait geçmişe yönelik bilgilere bu arayüz sayesinde kolayca erişildiği görülür. Oradaki arayüz olmasa ve kayıtlar veri tabanı üzerinden kullanıcılara sunulsa çok anlamlı sonuçlarla karşılaşılmayacaktır. Üstelik veri tabanı konusunda ön bilgisi olmayan biri, istediği kayıtlara ulaşamayacaktır. Buradaki ihtiyaçtan dolayı veri tabanları arayüzlerle entegre edilir. Böylece kullanıcı sadece istediği bilgiyi seçer ve bu işlemde hiç zorlanmaz.

Bu tür bir projeye başlamadan önce yapılacakların **algoritması** çıkarılmalı ya da yapılacaklar bir plana dökülmelidir. Bu projede ilişkili üç tablo vardır. Veri tabanındaki tablolar üzerinde ekleme, görüntüleme, düzenleme ve silme işlemleri (**CRUD**) yapılmak istenmektedir. Bu noktada form tasarımı tamamen kişiye özgü olabilir. Projede üç ayrı tablo çağrılıp her bir tablo için görüntüleme, ekleme, düzenleme ve silme işlemleri gerçekleştirecektir (ogrencibilgi tablosunda öğrenci bilgileri, ogrencinot tablosunda öğrencilerin notları ve dersler tablosunda dersler ile ilgili işlemler). İşlemlere başlanmadan önce arka plan için birkaç tane resim ve simge paketi indirilirse form tasarlanırken bunlar kullanılabilir.

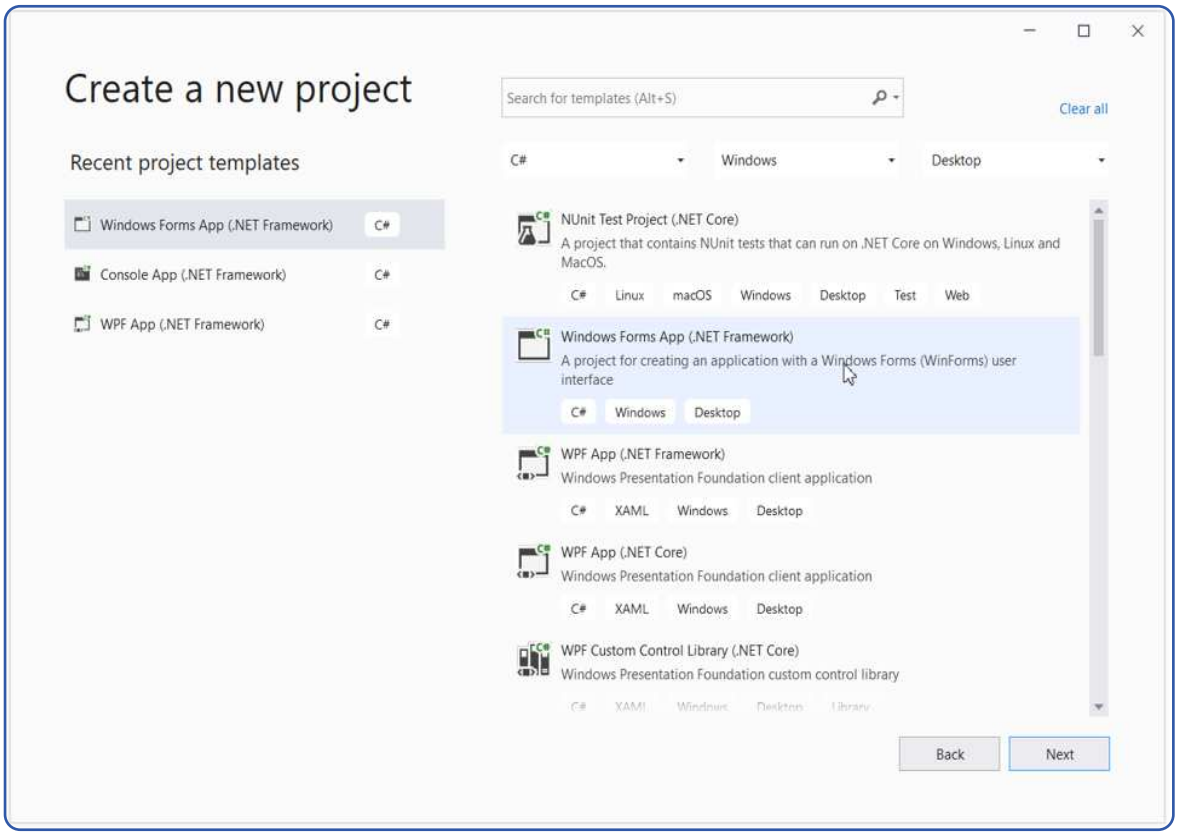
Kod editörü ile veri tabanı bağlantısı yapmak için bir arayüz oluşturulur.

6.9.1. Form Tasarımları

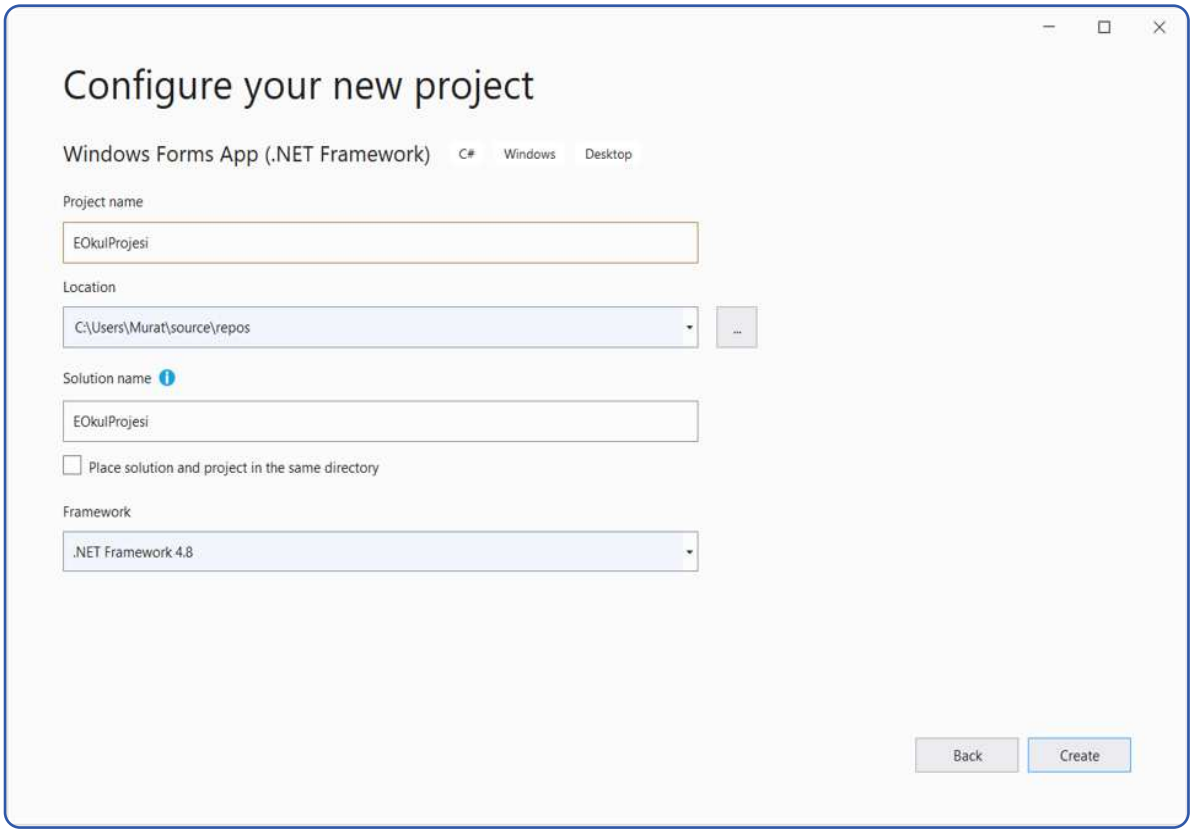
Hazırlanacak form verilerin gösterilmesini, eklenmesini, silinmesini, güncellenmesini sağlayacak şekilde tasarlanmalıdır. Yeni bir proje ve boş bir form açılır (Görsel 6.71, Görsel 6.72, Görsel 6.73, Görsel 6.74).



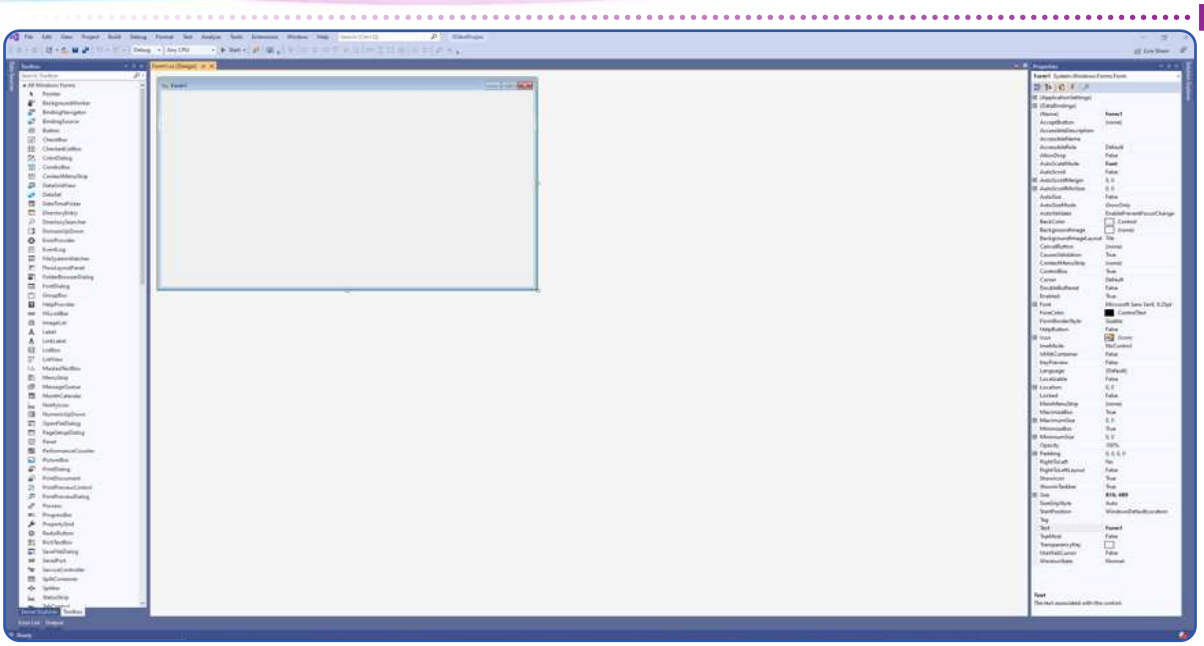
Görsel 6.71: Kod editörü açılış penceresi



Görsel 6.72: Yeni proje oluşturma



Görsel 6.73: Yeni proje ayarları penceresi



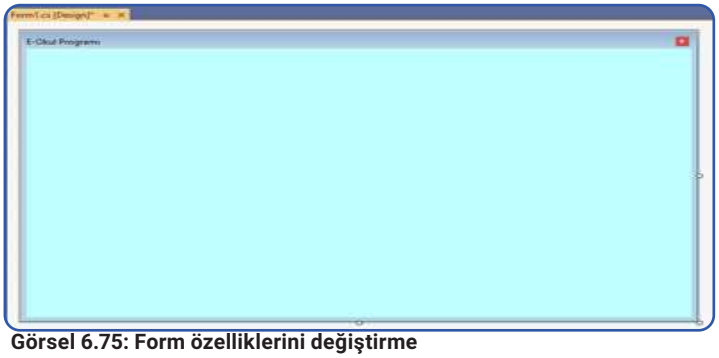
Görsel 6.74: Kod editörü yeni proje form ekranı

6.9.2. Form Özellikleri

Formla ilgili birkaç ayar yapılır. Bu ayarlar şunlardır:

- Formun **Name** özelliği eokulForm yapılır.
- Formun **Text** özelliği E-Okul Programı yapılır.
- Formun **Icon** kısmına internetten bulunan bir ikon dosyası (*.ico) eklenir.
- Formun **StartPosition** özelliği CenterScreen yapılır.
- Formun **BackColor** özelliğine bir renk veya **BackgroundImage** özelliğine bir resim eklenir.
- Formun **FormBorderStyle** özelliği FixedToolWindow yapılır.

Form tasarımı gözden geçirilir (Görsel 6.75).



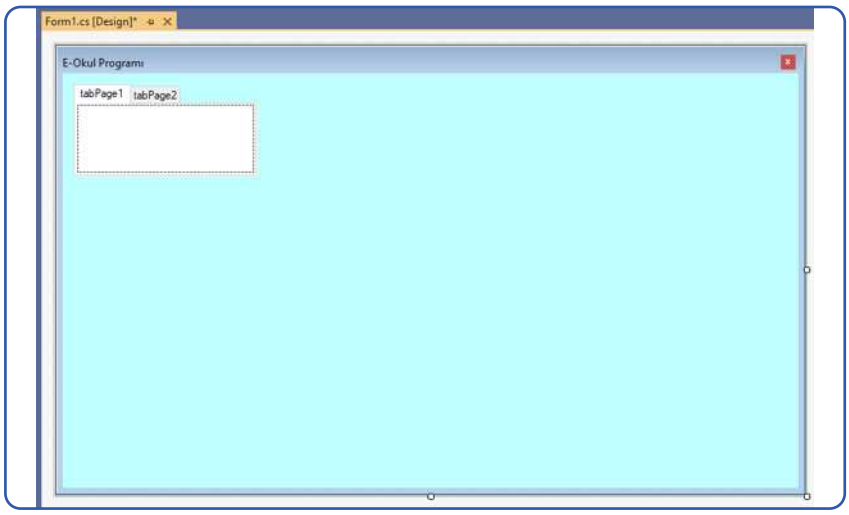
Görsel 6.75: Form özelliklerini değiştirme

Projeye başlanmadan önce bir yol haritası çizilir (Bu yol haritasının kesin olmadığı ve istenildiğinde değiştirebileceği unutulmamalıdır.). Öncelikle yapılacak işlemler belirlenmelidir. Projede bir ana ekranda bütün bilgiler gösterilecektir. Üç tablonun birleşimi sonucu bütün bilgiler ana forma gelecek ve bu ana formun bir kısmında veri ekleme alanı olacaktır. Her bir tablo için bir alan oluşturulup bu alanda ekleme, silme, güncelleme, arama işlemleri yapılacaktır. Ana sayfa ile beraber üç tablo da düşünülürse toplamda dört tane alan olacaktır. Bu yapı istenirse dört ayrı formla istenirse de **TabControl** bileşeniyle yapılabilir.

6.9.3. TabControl Bileşeni

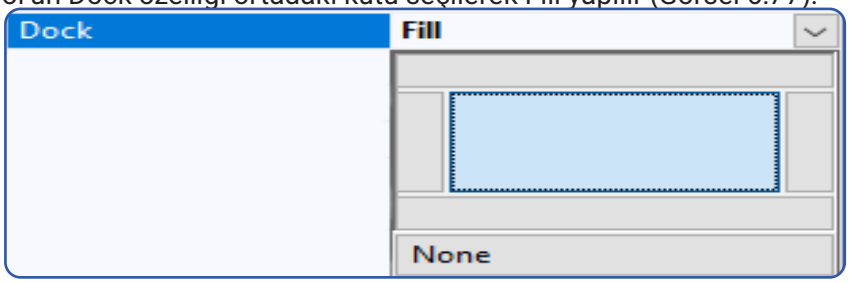
TabControl bileşeninin sekmeli yapısı sayesinde formu değiştirmeden sekmeler arasında geçiş yaparak işlemler gerçekleştirilir. Forma TabControl eklemek için sol taraftaki **ToolBox** penceresinin arama kutusuna TabControl yazılır. Çıkan bileşen tıpkı bir buton gibi forma sürüklenir (Görsel 6.76).

Burada forma tıklandıktan sonra mutlaka properties kısmından hangi bileşende olunduğu kontrol edilmelidir çünkü TabControl, içinde TabPage1, TabPage2 gibi sekmeleri de barındırmakta dır ve bazen bunlar seçim ekranında karışabilmektedir. TabControl bileşeninin özellikleri **properties** kısmından belirlenir.



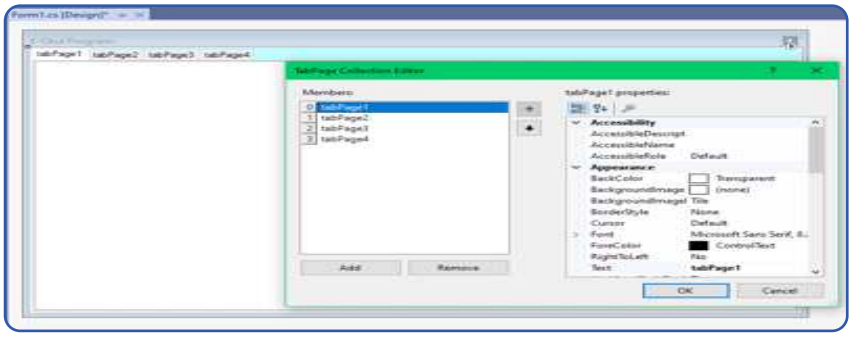
Görsel 6.76: Forma TabControl ekleme

- TabControl'un Name özelliği tabctrlEokul yapılır.
- TabControl'un Font özelliğinden Size değeri 20 yapılır ve istenirse Türkçe karakter destekleyen bir font seçilir.
- TabControl'un Dock özelliği ortadaki kutu seçilerek Fill yapılır (Görsel 6.77).



Görsel 6.77: TabControl Fill özelliği

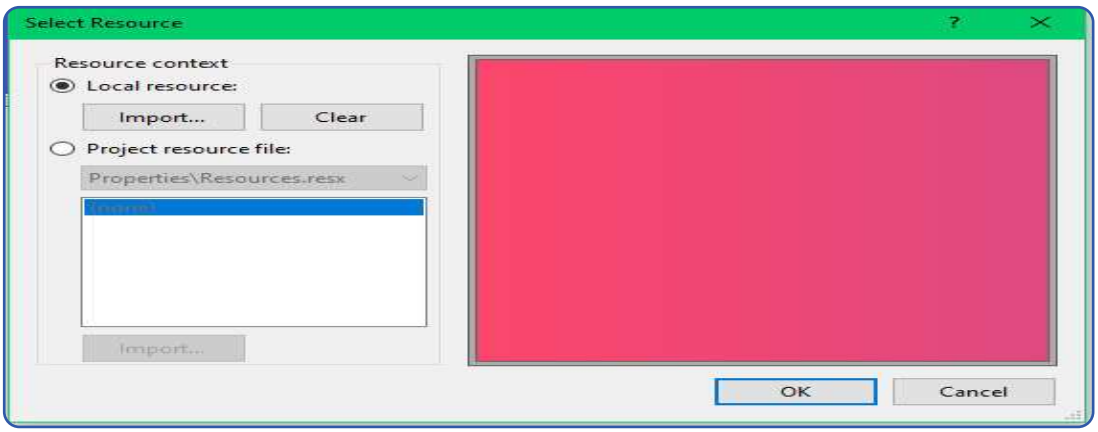
- TabPages (Collection) kısmındaki "..." kutucuğuna basılarak iki tane daha tab eklenir ve ayarlar yapılır. Toplamda dört adet tab olacaktır (Görsel 6.78).



Görsel 6.78: TabControl ayarları penceresi

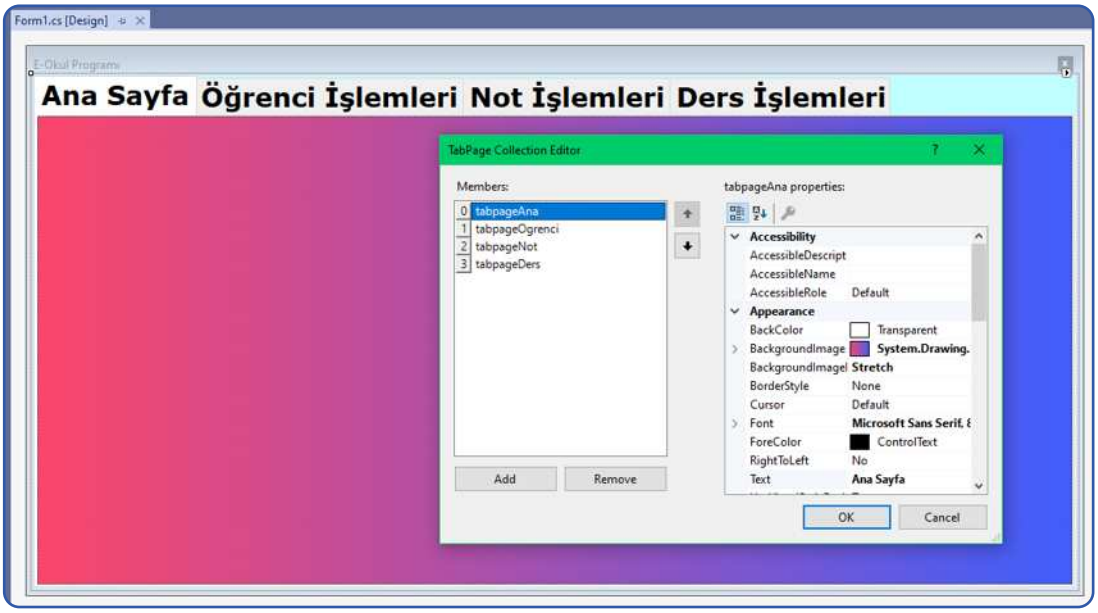
Görsel 6.78'deki ekranda tab'ların ayarları yapılabilir.

- Her bir tab sayfasına sırasıyla "**tabpageAna**", "**tabpageOgrenci**", "**tabpageNot**" ve "**tabpageDers**" olarak name verilir.
- Bulunan resimler her bir tabpage sayfasına **BackgroundImage** olarak eklenir. Bunun için aşağıda gelen ekrandan "**Import**" seçeneği işaretlendikten sonra istenilen resmin seçilmesi yeterlidir (Görsel 6.79).



Görsel 6.79: TabControl arka plana resim ekleme

- Bu ekleme işlemi yapıldıktan sonra tab'lar için sırasıyla "**Ana Sayfa**", "**Öğrenci İşlemleri**", "**Not İşlemleri**" ve "**Ders İşlemleri**" yazılır. Burada yazılan isimler ekranda görünecektir (Görsel 6.80).

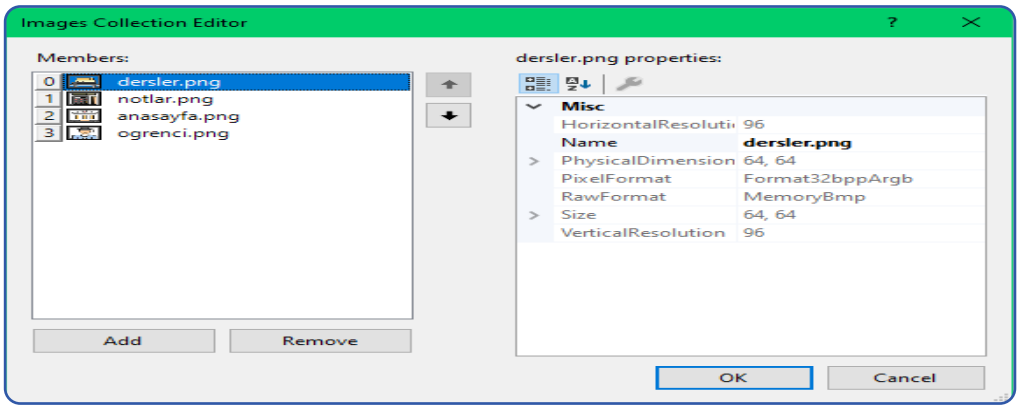


Görsel 6.80: TabControl bileşenin son hâli

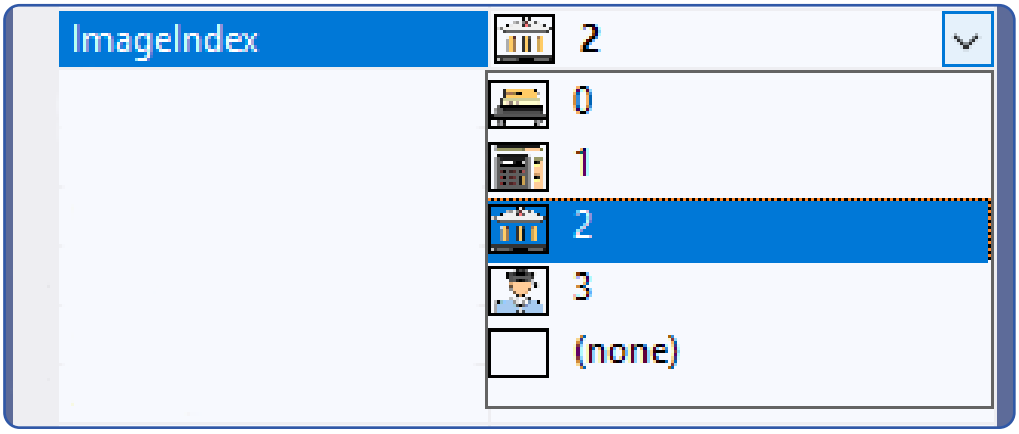
6.9.4. ImageList Bileşeni

Formun görsel kısmını biraz daha güçlendirmek için forma ToolBox penceresinden bir tane "**ImageList**" bileşeni eklenir. ImageList, içine resimler atılabilen bir bileşendir. Daha sonra bu resimler istenilen yerlerde kullanılabilir. Burada da ImageList bileşeni forma eklendikten sonra başlık kısmına yerleştirilecek simgeler, ImageList bileşeninin Images (Collection) özelliğine eklenebilir.

Simgeler eklendikten sonra "ImageList"in "**ImageSize**" özelliği kullanılarak istenilen en ve boy oranı verilebilir. Bu projede **64x64** boyutu kullanılmıştır (Görsel 6.81). TabControl bileşenine tıklanıp ImageList özelliği, eklenen "**imageList1**" olarak seçilebilir. Bu seçim yapıldıktan sonra TabPage'lerde bu simgeler kullanılmaya başlanabilir. TabPage'ler tıklanır ve "**ImageIndex**" veya "**ImageKey**" özelliklerinin birinden istenilen simgeler tek tek seçilir (Görsel 6.82).

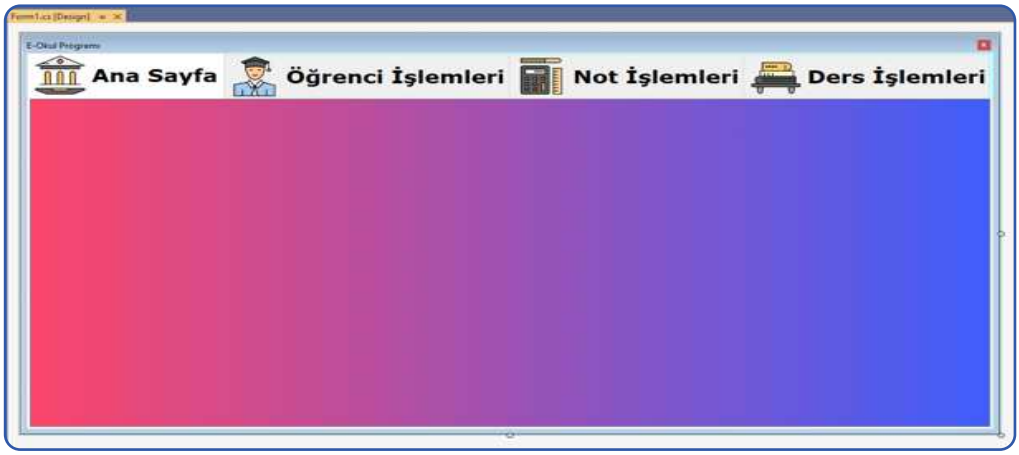


Görsel 6.81: ImageList bileşeni ayarları



Görsel 6.82: ImageList resim seçme

TabControl bileşeni Görsel 6.83'teki gibi görünecektir.



Görsel 6.83: TabControl sekmelerine ikon eklenmiş hâli

Her biri birbirinden farklı 4 adet sekmesi bulunan bir form elde edildi. Bu forma veri tabanı ile ilgili bileşenler eklenebilir.

6.10. ADO.NET

ADO.NET, "ActiveX Data Objects.NET" olarak adlandırılır. Veri tabanı ile uygulamalar arasında köprü görevi görür. ADO.NET ile uygulamada veri tabanına bağlanılabilir, veriler buradan listelenebilir, güncellenebilir, veri eklenebilir veya silinebilir. ADO.NET ile MySQL sorguları ve komutları uygulamada kullanılabilir (Görsel 6.84).

ADO.NET platformu sadece MySQL için değil, diğer veri tabanları için de kullanılabilir. ADO.NET mimarisinde kullanılan temel nesneler; **Connection**, **Command**, **DataReader**, **DataAdapter** ve **DataSet**'tir.

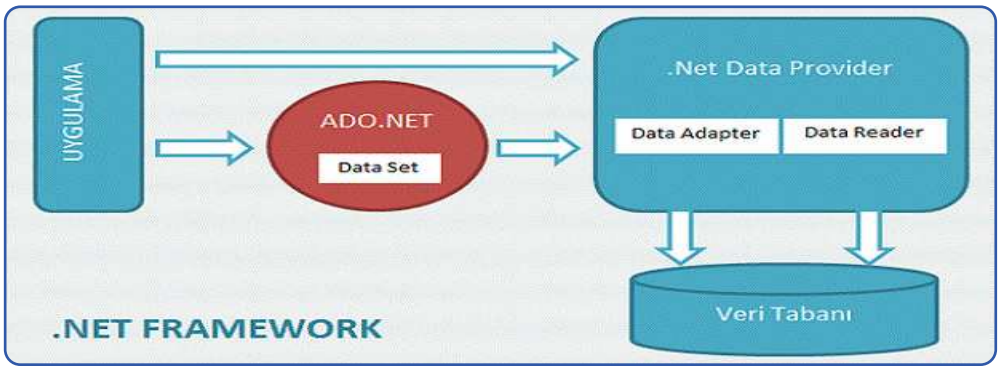
- **Connection Nesnesi:** Projede veri tabanı kullanımı için olmazsa olmaz nesnedir. Connection nesnesi, veri tabanı ile uygulama arasında bağlantı kurmak için kullanılır.
- **Command Nesnesi:** Veri tabanı ile uygulama arasındaki veri akışını iki yönlü olarak kontrol etmeye yarar. Execute.NonQuery, ExecuteScalar ve ExecuteReader olmak üzere üç alt nesnesi vardır.
- **DataReader Nesnesi:** Command nesnesi aracılığı ile getirilen verileri okumak için kullanılır.
- **DataAdapter Nesnesi:** Veri tabanından alınan veriler üzerinde değişiklik yapma, veri tabanına tekrar aktarma gibi işlemler için kullanılır. Bu işlemleri yapabilmek için dört adet alt nesne bulundurur. Bunlar; seçmek için SelectCommand, eklemek için InsertCommand, güncellemek için UpdateCommand ve silmek için de DeleteCommand nesneleridir.
- **DataSet Nesnesi:** ADO.NET teknolojisinin en yenilikçi ve güçlü tarafını DataSet oluşturur çünkü DataSet nesnesi, DataAdapter ile alınan verileri bağlantısız olarak depolayabilir ve yönetebilir.

.NET Framework ile veri tabanından veri çekmek için kullanılan System.Data komutu, ADO.NET mimarisinin temelini içerir.

6.11. Veri Tabanı Bağlantısı ve Bileşenlerin Eklenmesi

Projenin ilk sekmesinde bütün bilgilerin ve arama kutusunun yer aldığı bir alan ve diğer üç sekmede ise her bir tabloya ait verileri görüntüleyecek, güncelleyecek, silecek alanlar oluşturuldu. Böylece ana formda üç tablo birleştirilip sadece görüntüleme işlemi yapılarak karışıklık azaltılmış olur. Diğer sekmelerde de istenilen değişiklikler yapılabilir.

Projeye bir veri tabanı ekleneceğinde yapılması gereken bazı işlemler vardır. Bunlardan ilki ve en önemlisi, proje ve veri tabanı arasında köprü görevi görecek bir **"Connection String"** oluşturmaktır. Connection String'ler, veri tabanına göre değişiklik gösteren ve kalıplar hâlinde bulunan yapılardır. İnternette veri tabanına göre bu kalıplar bulunabilir ve özelleştirilebilir.

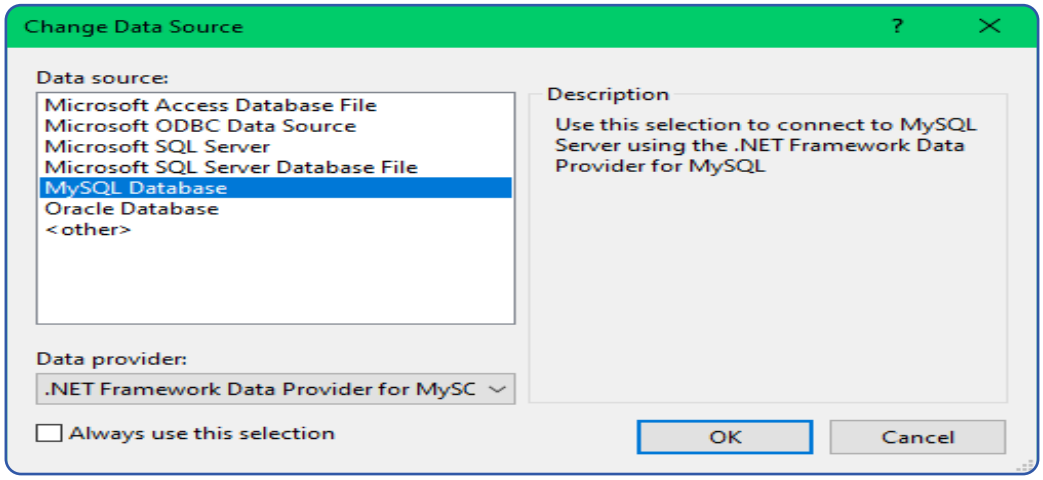


Görsel 6.84: ADO.NET şeması

Not: Connection String'lerdeki en ufak hata, veri tabanı ile bağlantıda soruna yol açacaktır.

6.11.1. MySQL Bağlantı Kontrolü

MySQL veri tabanı kurulmuştu fakat bu kurulumda MySQL ile kod editörü arasında bağlantıyı sağlayacak bileşenler kurulmamış olabilir. Bağlantıyı sağlayacak bileşenlerin kurulup kurulmadığını anlamanın en kolay yolu, menü çubuğunda yer alan "Tools→Connect to Database →Change" penceresinde MySQL veri tabanının görünüp görünmediğini kontrol etmektir.

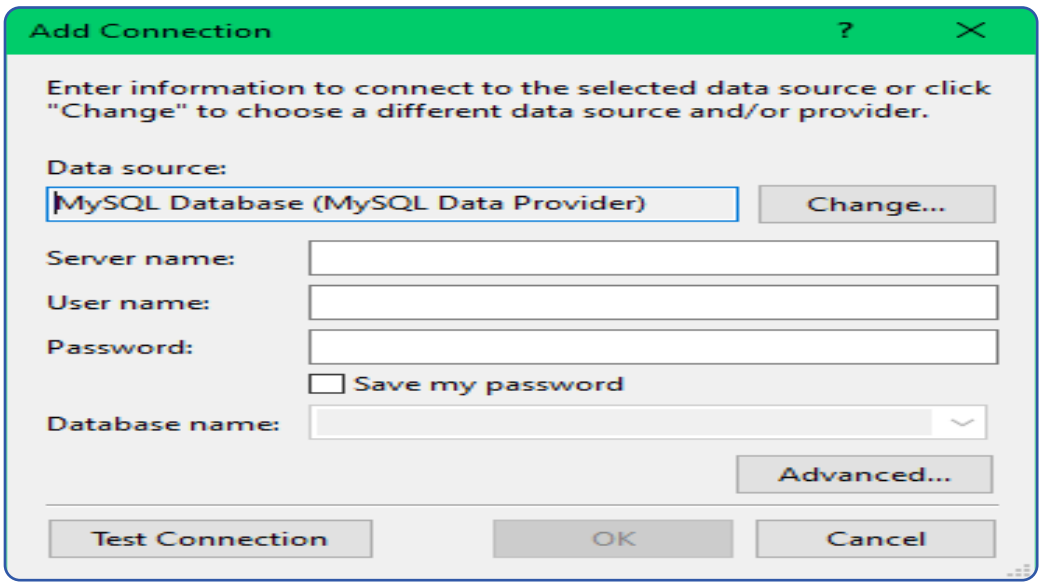


Görsel 6.85: Kod editörü veri tabanı bağlantı ekranı

Görsel 6.85'teki "MySQL Database" görünmüyorsa aşağıdaki iki eklenti indirilip sisteme kurulur.

- https://www.mysql.com/products/connector/ adresinden "ADO.NET Driver for MySQL (Connector/NET)"
- https://dev.mysql.com/downloads/windows/visualstudio/

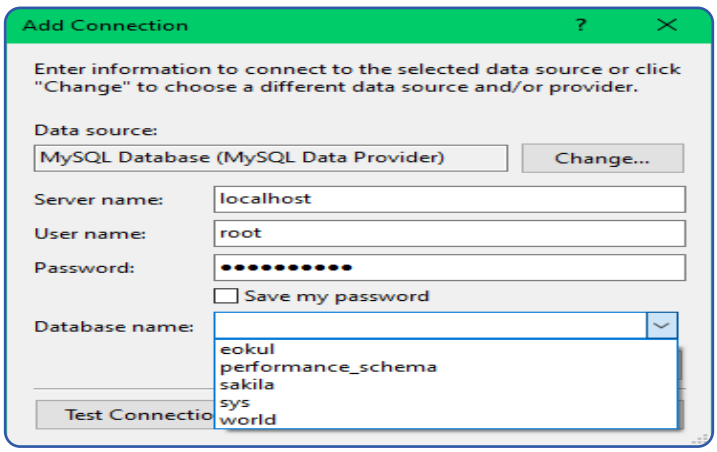
Bu işlemin ardından kod editörü yeniden başlatıldıktan sonra MySQL veri tabanlarına erişim sağlanır. MySQL Database seçildikten sonra bağlantı kontrol edilir.



Görsel 6.86: Kod editörü veri tabanı bağlantısı

- Görsel 6.86'da "**Server name**" kişisel bilgisayarda çalışıldığı için "**localhost**" olmalıdır.
- "User name", MySQL Server' kurulurken kullanılan "**root**" olmalıdır. Başka bir isim belirlendiyse o isim girilmelidir.
- "Password" ise MySQL Server kurulurken belirlenen şifre olmalıdır.

Bilgiler girildikten sonra veri tabanlarının adı geliyorsa ya da "**Test Connection**" düğmesine basıldığında "**Test connection succeeded**" yazıyorsa bağlantı doğru yapılmıştır. Bunlar olmuyorsa işlem adımları tekrar kontrol edilir (Görsel 6.87).



Görsel 6.87: Kod editörüne eklenebilecek veri tabanlarını listeleme

6.11.2. MySQL Connection String

Bağlantı cümlesi aşağıdaki kalıp ifade şeklinde yazılır.

"Server= ;Database= ;Uid= ;Pwd= ;"

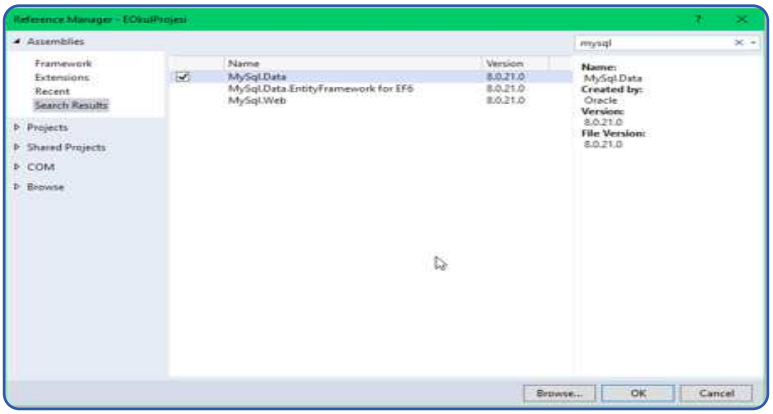
Kod editörü içine String olarak yukarıdaki kalıp eklenir. Buradaki değerlerin düzenlenmesi gerekmektedir. Bu projede aşağıdaki gibi bir bağlantı cümlesi olmalıdır:

"Server=localhost;Database=eokul;Uid=root;Pwd=milliegitimbakanligi;"

Server değeri, yerelde çalışıldığı için localhost; **Database** değeri, kullanılacak veri tabanı olan eokul; kullanıcı adı **(Uid)** ve şifre değerleri **(Pwd)**, MySQL kurarken belirlenen değerler olmalıdır.

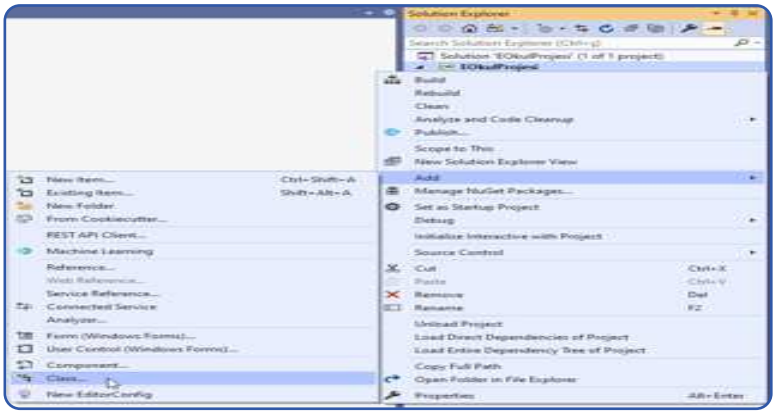
6.11.3. Projeye Giriş

Görsel olarak form düzenlendi, Connection String belirlendi ve MySQL ile bağlantı sağlandı. Projenin en önemli aşaması olan .NET üzerinden veri tabanı bağlantısı yapılabilir. MySQL, references olarak projeye dâhil edilir. Bu referans ile projede bütün MySQL özellikleri kullanılabilir. Bunun için projeye sağ tıklandıktan sonra "Add→References" yolu izlenip gelen pencerede arama yapılarak MySQL Data seçilmelidir (Görsel 6.88). MySQL Data ekrana gelmiyorsa "MySQL Bağlantı Kontrolü" kısmında yazılanlar tekrar uygulanır.



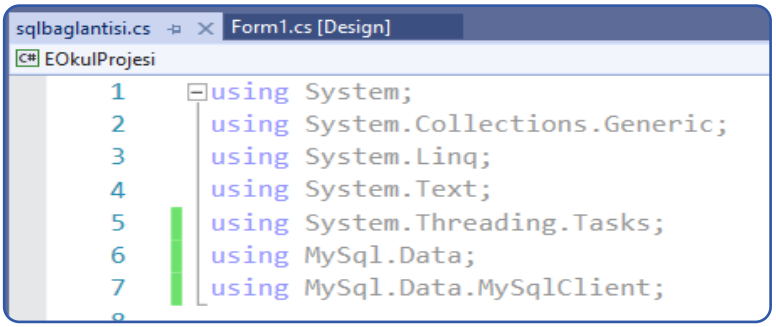
Görsel 6.88: Kod editörü referans ekleme

"**Connection String**" özelliği çok fazla kullanılacağı için bu bir **class** olarak eklenir. Böylece her seferinde bütün kodu tekrar yazmak yerine sadece bu class'tan üretilen nesne ile bağlantı yapılır. Bunun için **"Solution Explorer"** penceresinde projenin adına sağ tıklanır. Daha sonra gelen pencerede "**Add**" ve ardından "Class" seçenekleri işaretlenir (Görsel 6.89).



Görsel 6.89: Kod editörü class ekleme

Class'a "**sqlbaglantisi**" adı verildikten sonra gelen kod ekranında referans olarak "**using MySql.Data;**" ve "**using MySql.Data.MySqlClient;**" kütüphaneleri eklenir (Görsel 6.90).



Görsel 6.90: Projeye gerekli kütüphaneleri ekleme

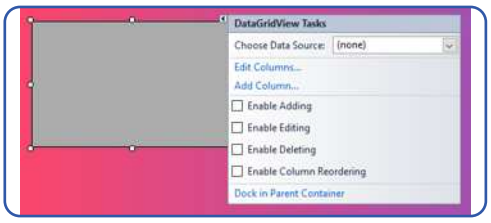
Not: Önceki adımlarda MySQL ile düzgün bağlantı kurulamamışsa burada eklenen satırların altı çizilecektir.

Connection String ile ilgili kodlar class dosyasının içine yazılır.

```
namespace EOkulProjesi
{
  class sqlbaglantisi
  {
    public MySqlConnection baglan() // class içinde baglan adında fonksiyon oluşturuldu.
    {
MySqlConnection baglanti = new MySqlConnection("Server=localhost;Database=eokul;Uid=roo-
t;Pwd=12345;");
//  Bağlantı yolu verildi.
        baglanti.Open();// Bağlantı açıldı.
        MySqlConnection.ClearPool(baglanti);// Bundan önceki bağlantılar temizlendi.
        return (baglanti);// Çağrıldığı yere bağlantı gönderildi.
    }

  }
}
```

6.11.4. DataGridView Bileşeni

Veri gösterimi yapılacağı için bütün sekmelere birer tane "**DataGridView**" nesnesi eklenir. DataGridView eklenirken gelen düzenleme penceresindeki **Enable** özellikler kaldırılır. "Choose Data Source" kısmı da kod ile belirtileceği için bir işlem yapılmasına gerek yoktur (Görsel 6.91).



Görsel 6.91: DataGridView eklenmesi

DataGridView'in Properties kısmı aşağıdaki şekilde düzenlenir:

- Önce **dtgvwAna**, **dtgvwOgrenci**, **dtgvwNot**, **dtgvwDers** şeklinde birer Name verilir.
- **AllowUserToAddRows** true ise false yapılır. Bu özellik en altta boş bir satır oluşmasına yol açar.
- **EditMode** özelliği "**EditProgrammatically**" olarak değiştirilir. Bu değişiklik sağlanmazsa her bir hücreye tıklandığında değişiklik yapılabilir.
- **SelectionMode** özelliği **"FullRowSelect"** yapılır. Böylece Datagridview'e tıklandığında bütün satır seçilir.
- **MultiSelect** özelliği false yapılır yoksa aynı anda birden fazla satır seçilir.
- **AutoSizeColumnsMode** özelliği **AllCells** yapılır. Böylece sütunlar yazı boyutu kadar otomatik olarak genişler.
- **DefaultCellStyle** özelliğinden de yazı tipi ile ilgili ayarlar yapılır. TabControl için yazı tipi büyüklüğü 20 yapılmıştır. Bu değişiklik TabControl içindeki bütün nesneleri etkilediği için DataGridView de yazı tipi boyutunu 20 almak isteyecektir. Burada uygun bir şekilde ayarlama yapılabilir. Standart yazı tipi boyutu 12'dir. Kolon başlıklarında aynı sorun yaşanırsa **ColumnHeadersDefaultCellStyle** özelliğinden font biçimlendirilebilir.

6.11.5. Projenin Kodlamasına Giriş

"**Events**" menüsünde **Load**'a çift tıklanarak ya da formda boş bir yere çift tıklanarak "**Load**" olayı aktifleştirilir, MySQL kütüphaneleri eklenir ve kullanılacak global değişkenler tanımlanır. **Global değişkenler**, metotlarla sınırlı olmayan küme parantezlerinin dışında tanımlanan değişkenlerdir. Böylece global değişkenler bir metotta kullanıldıktan sonra global değişkenler içindeki veri başka metotlarda da kullanılabilir.

```
 using MySql.Data; //Eklenen kütüphaneler
using MySql.Data.MySqlClient;
namespace EOkulProjesi
{
   public partial class eokulForm : Form
   {
      public eokulForm()
      {
         InitializeComponent();
      }
      //Global değişkenler
sqlbaglantisi bag = new sqlbaglantisi();// Bağlantı için oluşturulan class'tan türetilen bir nesne

DataTable table = new DataTable();//Tablolara yüklemek için bir adet DataTable değişkeni tanımlandı.
MySqlCommand kmt = new MySqlCommand();//İşlemler için bir MySqlCommand nesnesi tanımlandı.
      private void eokulForm_Load(object sender, EventArgs e)
      {

      }
```

Veri tabanı DataGridView nesnesine bağlanarak veri tabanındaki bilgiler proje içinde görülür. TabControl bileşeninde sekmeler **"SelectedIndexChanged"** olayı ile ayırt edilebilir. Sekmeler eklenirken her sekmeye 0'dan itibaren bir indeks verilir. Bu indeks değişimleri kontrol edilerek sekmelere erişim sağlanır.

TabControl bileşeninin 

olayına çift tıklanarak girilir. TabControl bileşeninin adı "**tabctrlEokul**" olduğu için C# bu isimde bir metot oluşturur. Hangi sekmeye gelindiğinde ne yapılması istenirse if-else yapısı içinde yazılacak şekilde aşağıdaki yapı oluşturulur.

```
private void tabctrlEokul_SelectedIndexChanged(object sender, EventArgs e)
    {
       if(tabctrlEokul.SelectedIndex==0)//Ana sekmesine girilince ne yapılsın kısmı
       {

       }
  else if (tabctrlEokul.SelectedIndex==1)//Öğrenci sekmesine girilince ne yapılsın kısmı
       {

       }
  else if (tabctrlEokul.SelectedIndex==2)//Not sekmesine girilince ne yapılsın kısmı
       {

       }
       else //Dersler sekmesine girilince ne yapılsın kısmı
       {

       }
    }
```

Sekmelere girildiğinde istenilen durum, veri tabanındaki tablolardan alınan verilerin **DataGridView** üzerinde gösterilmesidir. Bu gösterim işlemleri için birer metot tanımlanır ve bu metotlar **selectedIndexChanged** içinde çağrılır.

6.11.6. Dersler Sekmesi

Dersler sekmesi derslerin gösterilmesi için hazırlanır.

```
private void DersYukle()
    {
       dtgvwDers.RowHeadersVisible = false;//Satır başlıkları görünmez yapılır.
       dtgvwDers.Font = new Font("Verdana", 12);// Datagridview için font ayarlaması yapılır.
       table.Columns.Clear();// Tablonun önceki kullanımından kalan sütunlar temizlenir.
       table.Clear(); //  Tablo temizlenir.
       MySqlDataAdapter adtr = new MySqlDataAdapter("select * from dersler", bag.baglan());// Komut ve bağlantı
yüklenen DataAdapter nesnesi
       adtr.Fill(table);//DataAdapter nesnesine yüklenen bilgiler tabloya aktarılır.
       dtgvwDers.DataSource = table; //Datagridview bileşeni data kaynağı olarak table seçilir.
       dtgvwDers.Columns["derskodu"].HeaderText = "Ders Kodu"; //MySQL'den gelen kolon adı Datagridview'de
istenildiği gibi yazılır.
       dtgvwDers.Columns["dersadi"].HeaderText = "Ders Adı";
    }
//Bu metot, ders sekmesi tıklandığında çağrılacak şekilde yazılır.
private void tabctrlEokul_SelectedIndexChanged(object sender, EventArgs e)
    {
       if(tabctrlEokul.SelectedIndex==0)
       {
       }
 else if (tabctrlEokul.SelectedIndex==1)
       {
       }
       else if (tabctrlEokul.SelectedIndex==2)
       { }
       else //Dersler sekmesine girilince ne yapılsın kısmı
       {
          DersYukle();
       }
```

Dersler sekmesine girildiğinde **DersYukle()** metodu çağrılacak ve metottaki kodlar çalışacaktır. F5 ile kod çalıştırılarak denenir (Görsel 6.92).

Görsel 6.92: DersYukle() metodunun görünüşü

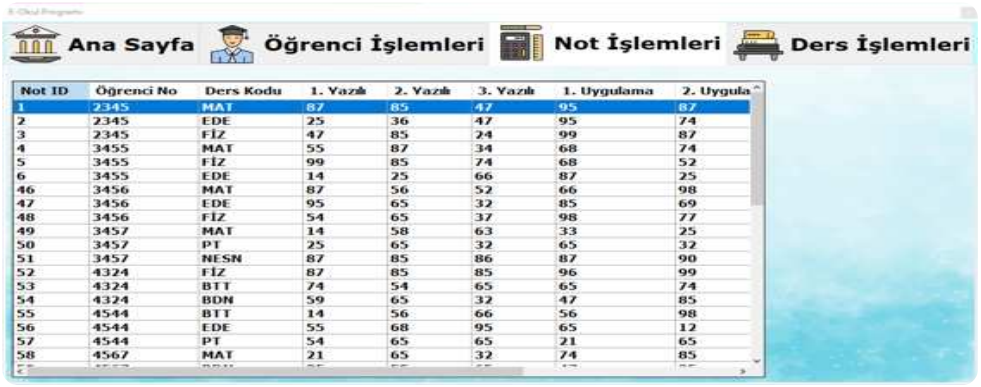
Diğer tablolar için de benzer şekilde metotlar yazılır ve sekmelere gelindiğinde metotların çağrılması sağlanır.

Not: Bu aşamada MySQL ile veri tabanına tekrar erişilerek gerekli alanların yazımları mutlaka kontrol edilir. Buradaki alanlar veri tabanı ile aynı olmalıdır. Tek bir harf hatası, veri tabanının düzgün çalışamamasına neden olacaktır.

6.11.7. Notlar Sekmesi

Notlar sekmesi notların gösterilmesi için hazırlanır.

```
private void NotYukle()
    {
        dtgvwNot.RowHeadersVisible = false;//Satır başlıkları görünmez yapılır.
        dtgvwNot.Font = new Font("Verdana", 12);// Datagridview için font ayarlaması yapılır.

        table.Columns.Clear();//Tablonun önceki kullanımından kalan sütunlar temizlenir.
        table.Clear(); // Tablo temizlenir.

        MySqlDataAdapter adtr = new MySqlDataAdapter("select * from ogrencinot", bag.baglan());//Komut ve bağlantı yüklenen DataAdapter nesnesi

        adtr.Fill(table);//DataAdapter nesnesine yüklenen bilgiler tabloya aktarılır.
        dtgvwNot.DataSource = table; //Datagridview bileşeni data kaynağı olarak table seçilir.
        dtgvwNot.Columns["ogrenciNotId"].HeaderText = "Not ID"; //MySQL'den gelen kolon adı Datagridview'de istenildiği gibi yazılır.
        dtgvwNot.Columns["ogrNoFK"].HeaderText = "Öğrenci No";
        dtgvwNot.Columns["dersKoduFK"].HeaderText = "Ders Kodu";
        dtgvwNot.Columns["yazili1"].HeaderText = "1. Yazılı";
        dtgvwNot.Columns["yazili2"].HeaderText = "2. Yazılı";
        dtgvwNot.Columns["yazili3"].HeaderText = "3. Yazılı";
        dtgvwNot.Columns["uygulama1"].HeaderText = "1. Uygulama";
        dtgvwNot.Columns["uygulama2"].HeaderText = "2. Uygulama";
        dtgvwNot.Columns["sozlu1"].HeaderText = "1. Sözlü";
        dtgvwNot.Columns["sozlu2"].HeaderText = "2. Sözlü";
        dtgvwNot.Columns["ortalama"].HeaderText = "Ortalama";
    }
```



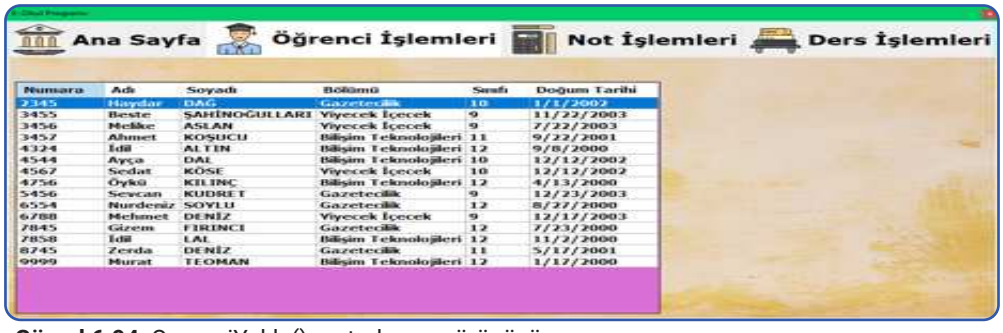
Not ID	Öğrenci No	Ders Kodu	1. Yazılı	2. Yazılı	3. Yazılı	1. Uygulama	2. Uygula
1	2345	MAT	87	85	47	95	87
2	2345	EDE	25	36	47	95	74
3	2345	FİZ	47	85	24	99	87
4	3455	MAT	55	87	34	68	74
5	3455	FİZ	99	85	74	68	52
6	3455	EDE	14	25	66	87	25
46	3456	MAT	87	56	52	66	98
47	3456	EDE	95	65	32	85	69
48	3456	FİZ	54	65	37	98	77
49	3457	MAT	14	58	63	33	25
50	3457	PT	25	65	32	65	32
51	3457	NESN	87	85	86	87	90
52	4324	FİZ	87	85	85	96	99
53	4324	BTT	74	54	65	65	74
54	4324	BDN	59	65	32	47	85
55	4544	BTT	14	56	66	56	98
56	4544	EDE	55	68	95	65	12
57	4544	PT	54	65	65	21	65
58	4567	MAT	21	65	32	74	85

Görsel 6.93: NotYukle() metodunun görünüşü

6.11.8. Öğrenci İşlemleri Sekmesi

Öğrenci işlemleri sekmesi öğrenci bilgilerinin gösterilmesi için hazırlanır.

```
private void OgrenciYukle()
    {
        dtgvwOgrenci.RowHeadersVisible = false;//Satır başlıkları görünmez yapılır.
        dtgvwOgrenci.Font = new Font("Verdana",12);// Datagridview için font ayarlaması yapılır.

        table.Columns.Clear();//Tablonun önceki kullanımından kalan sütunlar temizlenir.
        table.Clear(); // Tablo temizlenir.

        MySqlDataAdapter adtr = new MySqlDataAdapter("select * from ogrencibilgi", bag.baglan());//Komut ve bağlantı yüklenen DataAdapter nesnesi

        adtr.Fill(table);//DataAdapter nesnesine yüklenen bilgiler tabloya aktarılır.
        dtgvwOgrenci.DataSource = table; //Datagridview bileşeni data kaynağı olarak table seçilir.
        dtgvwOgrenci.Columns["ogrenciNo"].HeaderText = "Numara"; //MySQL'den gelen kolon adı Datagridview'de istenildiği gibi yazılır.
        dtgvwOgrenci.Columns["ogrenciAdi"].HeaderText = "Adı";
        dtgvwOgrenci.Columns["ogrenciSoyadi"].HeaderText = "Soyadı";
        dtgvwOgrenci.Columns["ogrenciBolumu"].HeaderText = "Bölümü";
        dtgvwOgrenci.Columns["ogrenciSinifi"].HeaderText = "Sınıfı";
        dtgvwOgrenci.Columns["ogrencidogumtarihi"].HeaderText = "Doğum Tarihi";
    }
```



Numara	Adı	Soyadı	Bölümü	Sınıfı	Doğum Tarihi
2345	Haydar	DAĞ	Gazetecilik	10	1/1/2002
3455	Beste	ŞAHİNOĞULLARI	Yiyecek İçecek	9	11/22/2003
3456	Melike	ASLAN	Yiyecek İçecek	9	7/22/2003
3457	Ahmet	KOŞUCU	Bilişim Teknolojileri	11	9/22/2001
4324	İdil	ALTIN	Bilişim Teknolojileri	12	9/8/2000
4544	Ayça	DAL	Bilişim Teknolojileri	10	12/12/2002
4567	Sedat	KÖSE	Yiyecek İçecek	10	12/12/2002
4756	Öykü	KELİNÇ	Bilişim Teknolojileri	12	4/13/2000
5456	Sevcan	KUDRET	Gazetecilik	9	12/23/2003
6554	Nurdeniz	SOYLU	Gazetecilik	12	8/27/2000
6788	Mehmet	DENİZ	Yiyecek İçecek	9	12/17/2003
7845	Gizem	FIRINCI	Gazetecilik	12	7/23/2000
7858	İdil	LAL	Bilişim Teknolojileri	12	11/2/2000
8745	Zerda	DENİZ	Gazetecilik	11	5/17/2001
9999	Murat	TEOMAN	Bilişim Teknolojileri	12	1/17/2000

Görsel 6.94: OgrenciYukle() metodunun görünüşü

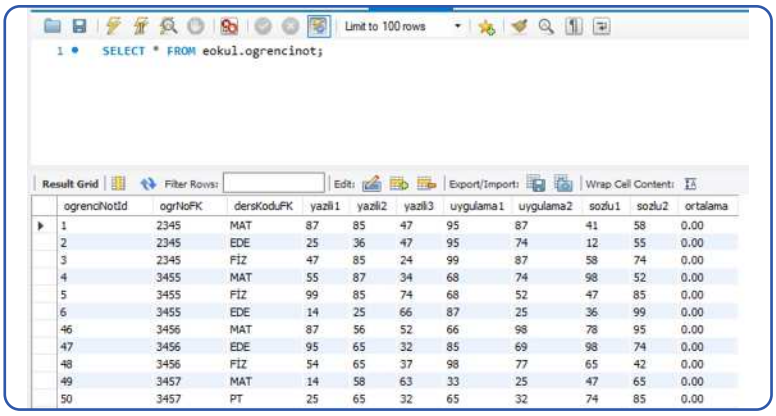
SelectedIndexChanged metodunun son hâli aşağıda verilmiştir.

```
private void tabctrlEokul_SelectedIndexChanged(object sender, EventArgs e)
    {
      if(tabctrlEokul.SelectedIndex==0)
      {

      }
      else if (tabctrlEokul.SelectedIndex==1)

     {
        OgrenciYukle();
      }
      else if (tabctrlEokul.SelectedIndex==2)
      {
        NotYukle();
      }
      else
      {
        DersYukle();
      }
    }
```

Görüldüğü gibi öğrenci, not ve ders sekmesine girildiğinde metotlar çağrılır ve verilerin datagridview içine yüklenmesi sağlanır.

6.11.9. Anasayfa Sekmesinin Doldurulması

Ana sekmede üç tablo birleştirilecektir. SQL komutu yazılırken MySQL arayüzünde çalışmak çok daha iyi olacaktır. MySQL Workbench arayüzü açılıp eokul veri tabanındaki **ogrencibilgi**, **ogrencinot** ve **dersler** tablosunu kapsayacak bir **"inner join"** SQL sorgusu yazılır.



ogrenciNotId	ogrNoFK	dersKoduFK	yazili1	yazili2	yazili3	uygulama1	uygulama2	sozlu1	sozlu2	ortalama
1	2345	MAT	87	85	47	95	87	41	58	0.00
2	2345	EDE	25	36	47	95	74	12	55	0.00
3	2345	FİZ	47	85	24	99	87	58	74	0.00
4	3455	MAT	55	87	34	68	74	98	52	0.00
5	3455	FİZ	99	85	74	68	52	47	85	0.00
6	3455	EDE	14	25	66	87	25	36	99	0.00
46	3456	MAT	87	56	52	66	98	78	95	0.00
47	3456	EDE	95	65	32	85	69	98	74	0.00
48	3456	FİZ	54	65	37	98	77	65	42	0.00
49	3457	MAT	14	58	63	33	25	47	65	0.00
50	3457	PT	25	65	32	65	32	74	85	0.00

Görsel 6.95: MySQL'de ogrencinot tablosunun görünüşü

Normalizasyon kuralları çerçevesinde tablo olabildiğince sade ve sadece sayılardan oluşacak bir hâle getirildi. Görsel 6.95'te ogrNoFK alanı, ogrencibilgi tablosundaki ogrenciNo alanı ile ve dersKoduFK alanı ise dersler tablosundaki derskodu alanı ile ilişkilendirilmişti. Daha önce Where ifadesiyle iki tablo ilişkilendirildiği alanlar üzerinden birleştirildi fakat şu an üç tablodan veri çekilmesi gerekmektedir. Burada devreye **"Inner Join"** komutu girer çünkü ogrNoFK'ya göre öğrencinin adı, soyadı, dersKoduFK ile de dersin adı ve ogrencinot tablosundan notların gösterilmesi gerekmektedir.

Inner Join geniş bir komut olduğu için MySQL içinde yazılması daha doğru olacaktır. Böylece hatalar daha net görülerek onlara müdahale edilebilir (Görsel 6.96).

```
select B.ogrenciNo, B.ogrenciAdi,B. ogrenciSoyadi, D.dersAdi, N.yazili1, N.yazili2,
N.yazili3, N.uygulama1, N.uygulama2, N.sozlu1, N.sozlu2, N.ortalama from ogrencinot as N
inner join ogrencibilgi as B
on
N.ogrNoFK=B.ogrenciNo
inner join dersler as D
on
N.dersKoduFK=D.derskodu
order by B.ogrenciNo DESC
```

Görsel 6.96: MySQL'de Inner Join sorgusunun yazımı

Yazılan SQL kodu incelendiğinde aşağıdaki bilgilere ulaşılır.

Select B.ogrenciNo, B.ogrenciAdi, B.ogrenciSoyadi, D.dersAdi, N.yazili1, N.yazili2, N.yazili3, N.uygulama1, N.uygulama2, N.sozlu1, N.sozlu2, N.ortalama from ogrencinot as N

Bu kısımda kullanılacak alanlar sorguya eklenir. Buradaki harfler, veri tabanlarına sonradan eklenen kısaltmaları temsil etmektedir. Örneğin, **"ogrencinot as N"** ile ogrencinot yerine N harfinin kullanılacağı belirtilmiştir.

Bu kısımda eklenen tablolara harf olarak kısaltmalar verilirken bu tabloların hangi alana göre eşitleneceği de belirtilir. Buradaki eşitlikler, veri tabanı oluşturulurken diyagram alanında yapılan ilişkilendirmelerden kaynaklanır. Öğrenci numaraları ve ders kodu alanları birbiriyle eşitlenmiştir.

Tablo, en son satırdaki SQL kodu ile öğrenci numarası alanına göre büyükten küçüğe doğru sıralanmıştır.

inner join ogrencibilgi as B

on

N.ogrNoFK=B.ogrenciNo

inner join dersler as D

on

N.dersKoduFK=D.derskodu

order by B.ogrenciNo DESC

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ogrenciNo	ogrenciAdi	ogrenciSoyadi	dersAdi	yazili1	yazili2	yazili3	uygulama1	uygulama2	sozlu1	sozlu2
2345	Haydar	DAĞ	Matematik	87	85	47	95	87	41	58
2345	Haydar	DAĞ	Edebiyat	25	36	47	95	74	12	55
2345	Haydar	DAĞ	Fizik	47	85	24	99	87	58	74
3455	Beste	ŞAHİNOĞULLARI	Matematik	55	87	34	68	74	98	52
3455	Beste	ŞAHİNOĞULLARI	Fizik	99	85	74	68	52	47	85
3455	Beste	ŞAHİNOĞULLARI	Edebiyat	14	25	66	87	25	36	99
3456	Melike	ASLAN	Matematik	87	56	52	66	98	78	95
3456	Melike	ASLAN	Edebiyat	95	65	32	85	69	98	74
3456	Melike	ASLAN	Fizik	54	65	37	98	77	65	42
3457	Ahmet	KOŞUCU	Matematik	14	58	63	33	25	47	65
3457	Ahmet	KOŞUCU	Programl...	25	65	32	65	32	74	85
3457	Ahmet	KOŞUCU	Nesne Ta...	87	85	86	87	90	95	95
4324	İdil	ALTIN	Fizik	87	85	85	96	99	98	95
4324	İdil	ALTIN	Bilişim Te...	74	54	65	65	74	85	95
4324	İdil	ALTIN	Beden Eğ...	59	65	32	47	85	99	87
4544	Ayça	DAL	Bilişim Te...	14	56	66	56	98	98	74
4544	Ayça	DAL	Edebiyat	55	68	95	65	12	47	85
4544	Ayça	DAL	Programl...	54	65	65	21	65	87	65
4567	Sedat	KÖSE	Matematik	21	65	32	74	85	87	85
4567	Sedat	KÖSE	Beden Eğ...	85	55	65	47	35	65	74
4567	Sedat	KÖSE	Fizik	88	58	96	31	65	47	75
4756	Öykü	KILINÇ	Bilişim Te...	54	65	32	87	99	54	75
4756	Öykü	KILINÇ	Edebiyat	25	98	65	74	85	65	42
4756	Öykü	KILINÇ	Fizik	54	87	98	65	32	78	85
5456	Sevcan	KUDRET	Beden Eğ...	54	65	65	32	74	85	34
5456	Sevcan	KUDRET	Edebiyat	56	32	98	65	74	21	65
5456	Sevcan	KUDRET	Matematik	21	32	74	52	85	98	96
6554	Nurdeniz	SOYLU	Matematik	32	65	98	75	98	56	66
6554	Nurdeniz	SOYLU	Edebiyat	25	23	66	38	85	90	60
6554	Nurdeniz	SOYLU	Beden Eğ...	85	85	95	87	58	74	52
6788	Mehmet	DENİZ	Fizik	32	32	20	25	70	60	60
6788	Mehmet	DENİZ	Matematik	65	32	47	52	58	58	55
6788	Mehmet	DENİZ	Edebiyat	65	74	85	65	65	78	95
7845	Gizem	FIRINCI	Matematik	17	85	32	65	75	65	65
7845	Gizem	FIRINCI	Fizik	85	85	90	90	98	98	88
7845	Gizem	FIRINCI	Beden Eğ...	57	85	69	85	65	32	75
7858	İdil	LAL	Nesne Ta...	65	65	32	56	58	95	85
7858	İdil	LAL	Bilişim Te...	85	85	99	98	65	65	57
7858	İdil	LAL	Fizik	52	63	65	85	54	55	54

Görsel 6.97: Inner Join komutu sonrası tablonun görünüşü

Uygulama

Tabloyu Görsel 6.95 ile Görsel 6.97'yi kullanarak karşılaştırınız.

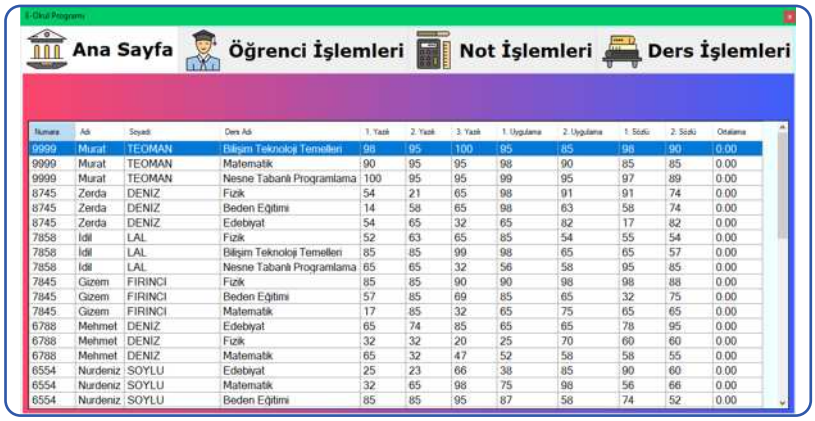
AnaTabloYukle() adında bir metot oluşturulur ve bu metodun içine MySQL'de yazılan "Inner Join" kodu eklenir (Kopyala yapıştır yapılabilir.).

```
private void AnaTabloYukle()
    {
        dtgvwAna.RowHeadersVisible = false;//Satır başlıkları görünmez yapılır.
        dtgvwAna.Font = new Font("Verdana", 12);// Datagridview için font ayarlaması yapılır.
        table.Columns.Clear();//Tablonun önceki kullanımından kalan sütunlar temizlenir.
        table.Clear(); // Tablo temizlenir.
        MySqlDataAdapter adtr = new MySqlDataAdapter("select B.ogrenciNo, B.ogrenciAdi,B. ogrenciSoyadi, D.dersAdi, N.yazili1, N.yazili2, N.yazili3, N.uygulama1, N.uygulama2, N.sozlu1, N.sozlu2, N.ortalama from ogrencinot as N inner join ogrencibilgi as B on N.ogrNoFK = B.ogrenciNo inner join dersler as D on N.dersKoduFK = D.derskodu order by B.ogrenciNo DESC", bag.baglan());//Komut ve bağlantı yüklenen DataAdapter nesnesi
adtr.Fill(table);//DataAdapter nesnesine yüklenen bilgiler tabloya aktarılır.
dtgvwAna.DataSource = table; //Datagridview bileşeni veri kaynağı olarak table seçilir.
dtgvwAna.Columns["ogrenciNo"].HeaderText = "Numara"; //MySQL'den gelen kolon adı Datagridview'de istenildiği gibi yazılır.
        dtgvwAna.Columns["ogrenciAdi"].HeaderText = "Adı";
        dtgvwAna.Columns["ogrenciSoyadi"].HeaderText = "Soyadı";
        dtgvwAna.Columns["dersadi"].HeaderText = "Ders Adı";
        dtgvwAna.Columns["yazili1"].HeaderText = "1. Yazılı";
        dtgvwAna.Columns["yazili2"].HeaderText = "2. Yazılı";
        dtgvwAna.Columns["yazili3"].HeaderText = "3. Yazılı";
        dtgvwAna.Columns["uygulama1"].HeaderText = "1. Uygulama";
        dtgvwAna.Columns["uygulama2"].HeaderText = "2. Uygulama";
        dtgvwAna.Columns["sozlu1"].HeaderText = "1. Sözlü";
        dtgvwAna.Columns["sozlu2"].HeaderText = "2. Sözlü";
        dtgvwAna.Columns["ortalama"].HeaderText = "Ortalama";
    }
```

Kod yazılır ve datagridview'de görünmesi istenilen kolon başlıkları eklenir. Metot, diğerlerinden farklı olarak **FormLoad()** içinde çağrılır. Form açıldığında ilk gelen bu sekme olduğu için form yüklendiği zaman verilerin ekrana gelmesi gerekir. Ayrıca **SelectedIndexChanged** metodu içine de bu metot eklenir.

```
private void tabctrlEokul_SelectedIndexChanged(object sender, EventArgs e)
    {
        if(tabctrlEokul.SelectedIndex==0)
        {
            AnaTabloYukle();
        }
        else if (tabctrlEokul.SelectedIndex==1)
        {
            OgrenciYukle();
        }
        else if (tabctrlEokul.SelectedIndex==2)
        {
            NotYukle();
        }
        else
        {
            DersYukle();
        }
    }
```

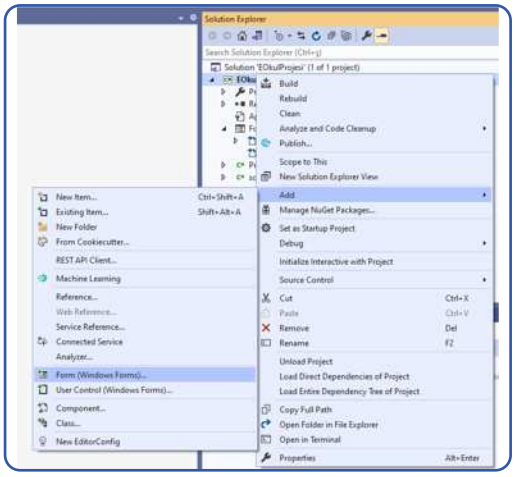
Projenin son hâli Görsel 6.98'deki gibi olacaktır. Artık hangi sekmeye geçilirse veriler tablolardan o sekmeye gelecektir.



Görsel 6.98: Projenin son hâli

6.12. Kayıt Ekleme

Tablolara C# üzerinden veri eklemek için projeye yeni bir form eklenir ve bu form, ana sekme üzerine eklenecek bir düğme yardımıyla çağrılır (Görsel 6.99).



Görsel 6.99: Projeye yeni form ekleme

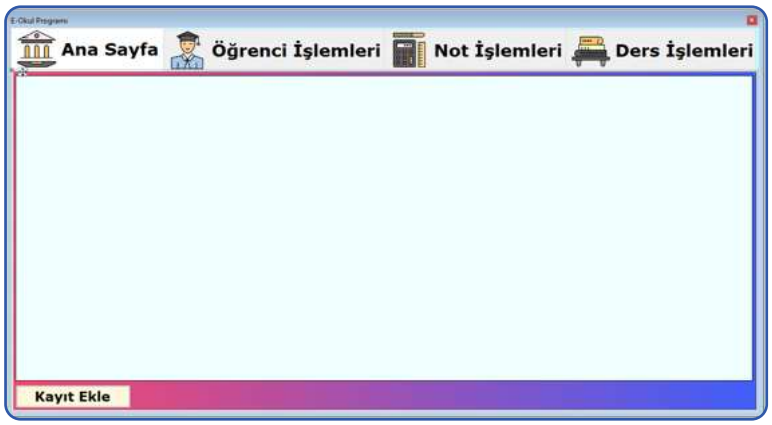
Form eklenirken forma bir ad verilmelidir. Forma **"YeniKayit"** ismi verilir. Boş bir form gelir ve bu form Görsel 6.100'deki gibi düzenlenir. İstenilen şekilde biçimlendirme yapılabilir.

Görsel 6.100: Yeni kayıt formunun tasarımı

İlk formdan ikinci formu çağırmak için ilk forma bir buton eklenir ve **Click** olayına aşağıdaki kodlar yazılır (Görsel 6.101).

```
YeniKayit kayitform = new YeniKayit(); //Formun türünden bir nesne oluşturuldu.
kayitform.Show(); //Bu nesne (form) görünür yapıldı.
```

Burada dikkat edilmesi gereken nokta, öğrenci bilgilerinin **ogrenci-bilgi** tablosundan ve ders bilgisinin de **dersler** tablosundan alınması gerektiğidir. Bu bilgiler, kullanıcıdan girdirilmeye kalkılırsa bu tablolarda olmayan bilgilerde veri tabanı hata verecektir. Bu yüzden bu bilgiler ilgili tablolardan çekilerek kullanıcıya seçtirilir. Formun tasarımı tamamlandıktan sonra veri tabanı bağlantısı için tekrar eklenir.



Görsel 6.101: Kayıt Ekle butonu eklenmesi

```
using MySql.Data;
using MySql.Data.MySqlClient;
//İki adet global değişken de eklenir.
sqlbaglantisi bag = new sqlbaglantisi();//Bağlantı için class oluşturuldu.
MySqlCommand kmt = new MySqlCommand();//İşlemler için komut değişkeni
```

Artık veri tabanıyla bağlantı kurulup veriler alınabilir. Kodlara başlamadan önce öğrenci numarası ile ders kodları veri tabanından çekilir. Seçilen değere göre öğrenci adı ve soyadı lblOgrenci'ye, ders koduna göre dersin adı lblDers'e aktarılır. Ardından alt tarafta notların girişi tamamlandıktan sonra ortalama da hesaplanarak veri tabanına ekleme işlemi gerçekleştirilir. Önce Load anında comboboxlar doldurulur.

```
private void YeniKayit_Load(object sender, EventArgs e) {
        DataTable tableogrenci = new DataTable();//tableogrenci nesnesi oluşturuyoruz
        MySqlDataAdapter adtrogrenci = new MySqlDataAdapter("select ogrenciNo from ogrencibilgi",
bag.baglan());//dataadapter içerisine sql cümlemizden gelen kayıtları yüklüyoruz
        adtrogrenci.Fill(tableogrenci);//veriyi datatable nesnemize yüklüyoruz
        adtrogrenci.Dispose();
        cmbOgrNo.DataSource = tableogrenci;//data kaynağı olarak datatable veriyoruz
        cmbOgrNo.ValueMember = tableogrenci.Columns[0].ToString();
        DataTable tablelers = new DataTable();//tableogrenci nesnesi oluşturuyoruz
        MySqlDataAdapter adtrders = new MySqlDataAdapter("select derskodu from dersler", bag.bag-
lan());//dataadapter içerisine sql cümlemizden gelen kayıtları yüklüyoruz
        adtrders.Fill(tablelers);//veriyi datatable nesnemize yüklüyoruz
          cmbDersKodu.DataSource = tablelers;//data kaynağı olarak datatable veriyoruz
    cmbDersKodu.ValueMember = tablelers.Columns[0].ToString();      }
```

Öğrenci no combobox'una çift tıklanarak **cmbOgrNo_SelectedIndexChanged** metodu oluşturulur. Bu metot ile öğrenci numarası değiştiğinde seçilen öğrencinin adı ve soyadı, altındaki lblOgrenci değişkenine aktarılır.

```
private void cmbOgrNo_SelectedIndexChanged(object sender, EventArgs e)
    {
   kmt.Connection = bag.baglan();//SQL komutu yazılacak değişkene veri tabanı ile bağlantı sağlanır.
   kmt.CommandText = ("select * from ogrencibilgi where ogrenciNo = '" + cmbOgrNo.Text.ToString() +
"'");//Combobox'tan gelen değer SQL komutuna eklenir.
        kmt.ExecuteNonQuery();//SQL komutu çalıştırılır.
        MySqlDataReader oku = kmt.ExecuteReader();//SQL komutu sonucunda okunan bütün değerler
datareader nesnesine aktarılır.
        while (oku.Read())//While döngüsü ile datareader içindeki veriler bitene kadar tek tek okunur. Şu an
için bu veriler öğrenci adı ve soyadıdır.
        {
           lblOgrenci.Text = oku["ogrenciAdi"].ToString() + "  " + oku["ogrenciSoyadi"].ToString();// Gelen
öğrenci ad ve soyadı birleştirilip lblogrenci'ye aktarılır.
        }
        oku.Close();// Veriler bittikten sonra bağlantı kapatılır.
    }
```

Yaklaşık aynı kodlar **cmbDersKodu_SelectedIndexChanged** için yazılır. Bu tür durumlarda copy-paste yapılabilir fakat bileşen ve değişken isimlerine çok dikkat edilmelidir.

```
private void cmbDersKodu_SelectedIndexChanged(object sender, EventArgs e)
    {
        kmt.Connection = bag.baglan();//SQL komutu yazılacak değişkene veri tabanı ile bağlantı sağlanır.

        kmt.CommandText = ("select * from dersler where derskodu = '" + cmbDersKodu.Text.ToString() +
"'");//Combobox'tan gelen değer SQL komutuna eklenir.

        kmt.ExecuteNonQuery();//SQL komutu çalıştırılır.

        MySqlDataReader oku = kmt.ExecuteReader();//SQL komutu sonucunda okunan bütün değerler
datareader nesnesine aktarılır.

        while (oku.Read())//While döngüsü ile datareader içindeki veriler bitene kadar tek tek okunur.Şu an
için bu veri ders adıdır.
        {
           lblDers.Text = oku["dersAdi"].ToString();//Gelen ders adı lblDers'e aktarılır.
        }
        oku.Close();//Veriler bittikten sonra bağlantı kapatılır.
    }
```

Görsel 6.102: Formun son durumu

Son olarak ortalama hesaplanarak bilgiler veri tabanına kaydedilir. Önce int türünde bir metot tanımlanır.

```
private double OrtalamaHesapla()//ortalama veritabanında double türünde olduğu için
        //double türünde bir metot tanımlandı.
    {
        double ortalama=0;
         ortalama = (Convert.ToDouble(txt1Yazili.Text) + Convert.ToDouble(txt2Yazili.Text) + Convert.
ToDouble(txt3Yazili.Text) + Convert.ToDouble(txt1Uyg.Text) + Convert.ToDouble(txt2Uyg.Text) + Con-
vert.ToDouble(txt1Sozlu.Text) + Convert.ToDouble(txt2Sozlu.Text))/7;
        return ortalama;//double türünde değişken döndürüldü.       }
```

Bu metotta bütün textbox içindeki notlar alınarak önce integer'e convert edilmiş sonra da toplanarak sonuç 7'ye bölünmüştür. En sonunda da ortalama değer döndürülmüştür.
Kayıt ekleme işlemine girmeden önce aynı öğrenci için aynı dersin eklenmesinin önüne geçilmelidir. Kontrol için **Boolean** metodu yazılır.

```
private Boolean KayitVarmi()
    {
    kmt.Connection = bag.baglan();//SQL komutu yazılacak değişkene veri tabanı ile bağlantı sağlanır.

    kmt.CommandText = ("select dersKoduFK, ogrNoFK from ogrencinot");//ogrencinot tablosundan
gerekli alanlar çekilir.

     kmt.ExecuteNonQuery();//SQL komutu çalıştırılır.

     MySqlDataReader oku = kmt.ExecuteReader();//SQL komutu sonucunda okunan bütün değerler data-
reader nesnesine aktarılır.

        while (oku.Read())// While döngüsü ile datareader içindeki veriler bitene kadar tek tek okunur.
        {
          if(oku["dersKoduFK"].ToString()== cmbDersKodu.Text.ToString() && oku["ogrNoFK"].ToString()==
cmbOgrNo.Text.ToString())// Veri tabanından okunan dersKoduFK değeri ile ogrNoFK alanları "ve" ile
kontrol edilir. Sonuç true ise ve bu iki değerin olduğu bir kayıt varsa false döndürülür.
          {
            return false;
          }
        }
        oku.Close();//Veriler bittikten sonra bağlantı kapatılır.
        return true;//Bütün veri tabanı kontrol edildikten sonra false dönmemişse bu öğrenci için bu ders
eklenmemiştir sonucuna ulaşılır ve true döndürülür.
    }
```

Bağlantı kısmı aynı kalır. Veri okunduktan sonra gelen değerler if ile kontrol edilerek hem öğrenci no hem de ders kodu varsa bu kaydın önceden eklendiği anlaşılır ve **false** döndürülür. Gelen verinin tamamında aynı kayıt yoksa **true** döndürülür. Bu verileri veri tabanına eklemek için gerekli kodlar butonun **click** olayına yazılır. Bu kodlar istenirse metot olarak da yazılabilir.

```
private void btnKayitEkle_Click(object sender, EventArgs e)
    {
        if(KayitVarmi())//Gelecek true veya false değerine göre if yazılır. true gelirse kayıt eklenecek, true gelmezse hata mesajı verilecektir.
        {
            try
            {
                //Komut cümlesi parametreler ile yazılacak ve aşağıdaki parametrelere de değer atanacaktır.
                string sorgu = "INSERT INTO ogrencinot(ogrNoFK,dersKoduFK,yazili1,yazili2,yazili3,uygulama1, uygulama2, sozlu1, sozlu2, ortalama) VALUES (@no, @derskodu, @yaz1, @yaz2, @yaz3, @uyg1, @uyg2, @soz1, @soz2, @ort)";
                //Parametrelere değer atanması
                kmt = new MySqlCommand(sorgu, bag.baglan());
                kmt.Parameters.AddWithValue("@no", cmbOgrNo.Text.ToString());
                kmt.Parameters.AddWithValue("@derskodu", cmbDersKodu.Text.ToString());
                kmt.Parameters.AddWithValue("@yaz1", txt1Yazili.Text);
                kmt.Parameters.AddWithValue("@yaz2", txt2Yazili.Text);
                kmt.Parameters.AddWithValue("@yaz3", txt3Yazili.Text);
                kmt.Parameters.AddWithValue("@uyg1", txt1Uyg.Text);
                kmt.Parameters.AddWithValue("@uyg2", txt2Uyg.Text);
                kmt.Parameters.AddWithValue("@soz1", txt1Sozlu.Text);
                kmt.Parameters.AddWithValue("@soz2", txt2Sozlu.Text);
                kmt.Parameters.AddWithValue("@ort", Math.Round(OrtalamaHesapla(),2));
                //Virgülden sonra iki basamak
                //Bağlantı yolu verildi.
                kmt.Connection = bag.baglan();
                kmt.ExecuteNonQuery();//Komut çalıştırıldı.
                kmt.Connection.Close();//Bağlantı kapatıldı.
                MessageBox.Show("Bilgiler veritabanına başarı ile eklendi!");//İşlem başarılı mesajı verildi.
            catch(Exception ee)//Hata varsa çalışacak kısım
            {
                MessageBox.Show(ee.ToString());
                MessageBox.Show("Lütfen verilerinizi kontrol ediniz.\nLütfen boş geçmeyiniz.");//Hata olduğu belirtilir.
            }
        }
        else
        {
            MessageBox.Show("Bu ders bu öğrenci için eklenmemiştir!\nBu dersin kaydı vardır.");
        }
    }
```

Veri tabanına kayıt eklerken parametreler kullanılabilir. Parametrelerin kullanımı görsel 6.103'te görülmektedir. Insert into cümlesine değerleri direkt vermek yerine (txtAd.Text gibi) @ad gibi bir parametre ile değeri değişken üzerinden aktarılabilir. Böylece hem kod daha düzenli gözükecektir hem de veri güvenliği açısından daha avantajlı bir kullanım olacaktır. Koda başlarken **KayitVarmi()** metodu kullanılarak öğrenci için bu dersin olup olmadığı kontrol edilir. **True** dönerse kod **try-catch** blokları içinde çalıştırılır, **false** dönerse hata mesajı verdirilir (Görsel 6.103).

Görsel 6.103: Başarılı ve başarısız kayıt ekleme sonuçları

Uygulama

1. Veri tabanına ekleme yapıldıktan sonra bütün alanları temizleyen metodu araştırarak yazınız.
2. Forma girilen değer 0'dan küçük veya 100'den büyükse tekrar değer girilmesini sağlayınız.
3. Forma sadece rakam girilmesini sağlayınız.

Kayıt eklendikten ve kayıt formu kapatıldıktan sonra ana form üzerinde kaydı görmek için bir tane yenile butonu eklenerek **AnaTabloYukle()** metodu çalıştırılır.

```
private void btnYenile_Click(object sender, EventArgs e)
{
  AnaTabloYukle();
}
```

Böylece kayıt eklendikten sonra düğmeye basılarak kaydın görünmesi sağlanabilir. Metot kullanımı sayesinde tek bir satır ile bütün kodlar tekrar çağrılarak kullanılabilir.

6.13. Arama Metodu

Ana formda öğrenci adına göre arama yapan bir metot yazılır. Bunun için ana sekmeye bir textbox eklenip **TextChanged** olayına arama metodu yazılarak çağrılır.

```
private void AnaSekmeAramaYap()
    {
      DataTable Anatable = new DataTable();//Bilgilerin aktarılacağı bir Datatable nesnesi oluşturulur.
      Anatable = table.Copy();//Sekme çağrılırken bilgilerin yüklendiği table nesnesinin bir kopyası içine aktarılır.
       DataView dv = Anatable.DefaultView;//Bu bilgileri görüntülemek için DataView nesnesi oluşturulup table ile bağlanır.
       dv.RowFilter = "ogrenciAdi LIKE '%" + txtAra.Text + "%' OR ogrenciSoyadi LIKE '%" + txtAra.Text + "%'";//Arama cümlesi yazılır.
       dtgvwAna.DataSource = dv;//Gelen arama sonuçları datagridview içine aktarılır.
    }
```

Burada SQL kullanmak işi uzatacağı için doğrudan datagridview nesnesinin **RowFilter** özelliği kullanılmıştır. Bu özelliğe veri tabanı üzerinden çekilen kolon adları baz alınarak (ogrenciAdi ve ogrenciSoyadi) sorgusu yazılır. Bu sorgu, datagridview üzerinde ad ve soyad alanına göre arama yapacaktır. Arama işlemine istenirse diğer alanlar da eklenebilir (Görsel 6.104).

Görsel 6.104: Arama işlemi

6.14. Ekleme, Silme ve Güncelleme İşlemleri

Öğrenci, not ve ders işlemleri sekmelerinde ekleme, silme, güncelleme yapılır. Sekmelerin üçünde de aynı işlemler yapılacağı için ilk seferden sonra bunlar copy-paste ile gerçekleştirilebilir.

6.14.1. Öğrenci Sekmesi İşlemleri

Öğrenci sekmesi içinde **OgrenciYukle();** metodu kullanılır. Böylece veri tabanına bağlanılarak bilgiler çekilir. Veri tabanından çekilen kolon başlıklarına göre işlemler gerçekleştirilir. Önce form tasarlanarak formdaki bileşenlere birer **"Name"** verilir.

Görsel 6.105: Öğrenci sekmesinde kayıt ekleme ve güncelleme için form tasarımı

Görsel 6.105'teki gibi sağ tarafa bir **panel** ve içine de kullanılacak bileşenler eklenir. Burada en üstte **radiobutton** ekleme sebebi, seçilen radiobutton'a göre ekleme veya güncelleme işleminin yapılmasıdır. Daha sonra güncelleme ve ekleme yapılacak her bir alan için bir **label** ve bir **textbox** eklenir. Doğum tarihi için bir adet **datetimepicker,** en alta da butonlar eklenir. Sınıf combobox'ının **DropDownStyle** değeri **DropDownList** yapılır ve **Items** özelliğinden 9, 10, 11, 12 eklenir.

Not: Burada textbox'lara birer name verilmesi çok önemlidir çünkü formda bolca kullanılacaktır.İsimlendirme yapılmaması çok büyük karışıklıklara yol açar.

Tasarım yapıldıktan ve isimlendirmeler gerçekleştirildikten sonra kodlar yazılabilir. Burada seçilen radiobutton'a göre bu düğmede ekle veya güncelle yazacaktır. Bunun için soldaki **rdbEkle** radiobutton'unun **Checked** özelliği true yapıldıktan sonra **CheckedChanged** olayına aşağıdaki kodlar yazılır.

```
private void rdbEkle_CheckedChanged(object sender, EventArgs e)
    {
      if (rdbGuncelle.Checked)
         btnGuncelleEkle.Text = "Güncelle";
      else
         btnGuncelleEkle.Text = "Ekle";
    }
```

Böylece form açıldığında butonda **Ekle** yazarken diğer seçenek seçildiğinde düğme üzerinde **Güncelle** yazacaktır. Ekle için btnGuncelleEkle düğmesine gerekli kodlar yazılır. Burada basit bir **INSERT INTO** cümlesi yazılır ve textbox'lardaki değerler veri tabanına gönderilir.

Not: Ekleme işlemi yapılırken kod yazımına çok dikkat edilmelidir. Kesin bir sonuca varılmak istenirse alanlar birer birer eklenip alanlara tek tek değer girilerek gidilmesi eklemeyi kolaylaştıracaktır.

6.14.1.1. Öğrenci Sekmesi Kayıt Ekleme

İf ile radiobutton'da "Ekle" seçilmişse bu kodlar çalışacaktır.

```
private void btnGuncelleEkle_Click(object sender, EventArgs e)
    {
      if (rdbEkle.Checked)
      {
        try//try başlangıcı
        {
          //Komut cümlesi parametreler ile yazılacak ve aşağıdaki parametrelere değer atanacaktır.
          string sorgu = "INSERT INTO ogrencibilgi(ogrencino, ogrenciAdi, ogrenciSoyadi, ogrenci-
Bolumu, ogrenciSinifi, ogrencidogumtarihi) VALUES(@no, @ad, @soyad, @bolum, @sinif, @dogum)";

          kmt = new MySqlCommand(sorgu, bag.baglan());

          //Parametrelere değer atanması
          kmt.Parameters.AddWithValue("@no", txtOgrNumara.Text);
          kmt.Parameters.AddWithValue("@ad", txtOgrAd.Text);
          kmt.Parameters.AddWithValue("@soyad", txtOgrSoyad.Text);
          kmt.Parameters.AddWithValue("@bolum", txtOgrBolum.Text);
          kmt.Parameters.AddWithValue("@sinif", cmbOgrSinif.Text);
          kmt.Parameters.AddWithValue("@dogum", dateDogum.Value.Date.ToString("yyy-
y-MM-dd"));

          kmt.Connection = bag.baglan();//Bağlantı yolu verildi.

          kmt.ExecuteNonQuery();//Komut çalıştırıldı.

          kmt.Connection.Close();//Bağlantı kapatıldı.
          OgrenciYukle();//Datagridview'e yeni öğrenci listesi yüklendi.

          MessageBox.Show("Bilgiler veritabanına başarı ile eklendi!");//İşlem başarılı mesajı veril-
di.
        }
        catch//Hata yakalama başlangıcı
        {
          MessageBox.Show("Ekleme Başarısız!\nLütfen Alanları Kontrol Ediniz.\nAynı numarada 2
öğrenci olamaz.\nBoş alan olamaz.");//Uyarı mesajı
        }
      }
```

Burada normal bir SQL cümlesi yazılır fakat veri tabanına değerler el ile değil, kullanıcının forma girdiği değerlere göre eklenir. **Kmt** değişkeni, MySqlCommand'dan türetilmiştir ve içine SQL komutları yazılabilir. Burada da kmt içine komut olarak **INSERT INTO** cümlesi yazıldı. Önce veri tabanındaki veri aktarılacak alanlar yazıldı, ardından VALUES ile her bir alana denk gelen bir parametre tanımlandı ve son olarak bu parametrelere değerler atandı. Formdaki veriler, **kmt.ExecuteNonQuery()** çalıştırıldığında bir hata yoksa veri tabanına eklenecektir.

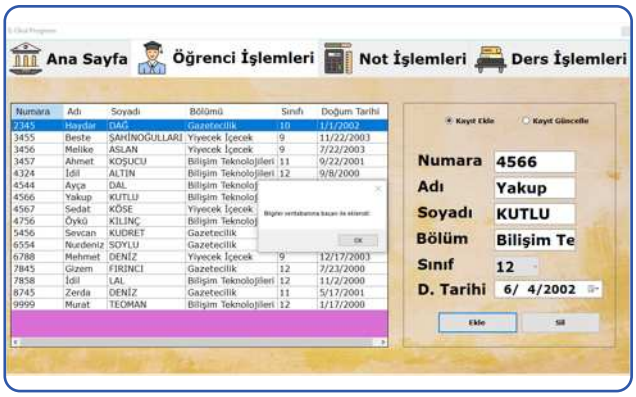
Veriler eklendikten sonra bağlantı kapatılır. Veri eklendiği için tablo **güncellenir**. Daha sonra kayıt eklendiğine dair mesaj, **MessageBox.Show** ile ekrana yazdırılır.

Kodda bir hata olursa bu hatanın giderilmesi için **try-catch-finally** kullanımı önemlidir. Try kodların olduğu alanı, catch bir hata olduğunda verilecek mesajı ve finally de hata olsa da olmasa da çalışacak kodları içerir. Bu yapı ile kodlardan veya kullanıcıdan kaynaklanan bir hata olsa bile program çökmeyecektir. Onun yerine catch kısmında yazılan mesajı veya kodları çalıştıracaktır. Bu projede finally'e gerek olmadığı için kullanılmamıştır. Proje çalıştırılarak denenir.

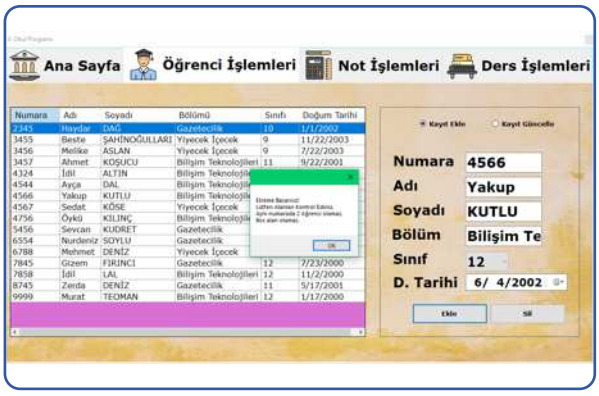
Görsel 6.106'da kayıt eklendi ve tabloya geldi. Hata almak için aynı kayıt tekrar eklenmeye çalışılır.

Sıra Sizde

Görsel 6.107'deki hatanın sebebini açıklayınız.



Görsel 6.106: Öğrenci sekmesine kayıt eklenmesi



Görsel 6.107: Öğrenci sekmesine kayıt ekleme hatası

6.14.1.2. Öğrenci Sekmesi Kayıt Güncelleme

Ekleme kodları bittikten sonra güncelleme kısmına geçilebilir. Güncelleme kodlarına geçmeden önce güncellenecek kaydı seçmek için bir işlem daha yapılmalıdır. Güncelleme işlemi, var olan bir kayıt üzerinde değişiklik yapmaktır. **Datagridview** nesnesine çift tıklandığında güncellenecek kaydı otomatik olarak forma yükleyen kodlar datagridview'in **CellDoubleClick** olayına yazılır.

```
private void dtgvwOgrenci_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
    {
//Burada yapılan işlem textbox'lara veri aktarmaktır.
        txtOgrNumara.Text = dtgvwOgrenci.Rows[e.RowIndex].Cells[0].Value.ToString();
        txtOgrAd.Text = dtgvwOgrenci.Rows[e.RowIndex].Cells[1].Value.ToString();
        txtOgrSoyad.Text = dtgvwOgrenci.Rows[e.RowIndex].Cells[2].Value.ToString();
        txtOgrBolum.Text = dtgvwOgrenci.Rows[e.RowIndex].Cells[3].Value.ToString();
        cmbOgrSinif.Text = dtgvwOgrenci.Rows[e.RowIndex].Cells[4].Value.ToString();
        dateDogum.Text = dtgvwOgrenci.Rows[e.RowIndex].Cells[5].Value.ToString();
    }
```

Bu kodla beraber herhangi bir kayıt çift tıklandığında o satırdaki veriler otomatik olarak textbox'lara yüklenecektir. Burada sağ taraftaki **Cells[0]** kısmı, datagridview'deki kolonları temsil etmektedir ve sol tarafla uyumlu olmak zorundadır. İlk satır baz alınırsa numara textbox'ına datagridview'deki o satırın ilk kolonundaki öğrenci numarası aktarılmaktadır. **Bu uyum çok önemlidir.**

Ekleme kısmı, radiobutton kontrol edilerek yapılmıştır. Ekleme yerine **rdbGuncelle** seçiliyse kodlar güncelleme işlemine göre yazılmalıdır. Bu kodlar önceki if'in devamıdır. Alanlar dolduktan sonra güncelleme kodları yazılabilir.

```
else if (rdbGuncelle.Checked)
        {
          try
          {

//Komut cümlesi parametreler ile yazılacak ve aşağıdaki parametrelere değer atanacaktır.
              string sorgu = "UPDATE ogrencibilgi SET ogrenciNo= @no , ogrenciAdi= @ad ,
ogrenciSoyadi= @soyad , ogrenciBolumu= @bolum , ogrenciSinifi= @sinif , ogrenciDogum-
Tarihi= @dogum  where ogrenciNo= @no ";

              kmt = new MySqlCommand(sorgu, bag.baglan());

              //Parametrelere değer atanması
              kmt.Parameters.AddWithValue("@no", txtOgrNumara.Text);
              kmt.Parameters.AddWithValue("@ad", txtOgrAd.Text);
              kmt.Parameters.AddWithValue("@soyad", txtOgrSoyad.Text);
              kmt.Parameters.AddWithValue("@bolum", txtOgrBolum.Text);
              kmt.Parameters.AddWithValue("@sinif", cmbOgrSinif.Text);
              kmt.Parameters.AddWithValue("@dogum", dateDogum.Value.Date.ToString("yy-
yy-MM-dd"));

              kmt.Connection = bag.baglan();//Bağlantı yolunu verildi.

              kmt.ExecuteNonQuery();//Komut çalıştırıldı.

              kmt.Connection.Close();//Bağlantı kapatıldı.
              OgrenciYukle();//Datagridview'e yeni öğrenci listesi yüklendi.

              MessageBox.Show("Güncelleme İşlemi Başarılı");//işlem başarılı mesajı veriyo-
ruz
          }
          catch//Hata yakalama başlangıcı
          {
              MessageBox.Show("Güncelleme Başarısız!\nLütfen Alanları Kontrol Ediniz.\nAy-
nı numarada 2 öğrenci olamaz.\nBoş alan olamaz.");//Uyarı mesajımız
          }
        }
    }
```

Buradaki yapı da ekleme ile paraleldir sadece kmt.CommandText'indeki SQL sorgu değişmiştir. UPDATE sorgusunda veriler doğrudan alanlara atanır. Ayrıca Where kısmına veri tabanından alınan sütun ismi yazılmalıdır. Burada "**ogrenciNo**" seçilen satırdan okunmuştur.

6.14.1.3. Öğrenci Sekmesi Kayıt Silme

Veri silme de diğer işlemlere benzer şekilde yapılmaktadır fakat silme işlemi geri alınamadığı için öncesinde kodlara bir onay mekanizması eklenmelidir. Burada yanlışlıkla veri silinmesinin önüne geçebilmek için bir uyarı penceresi ile onay alınacaktır.

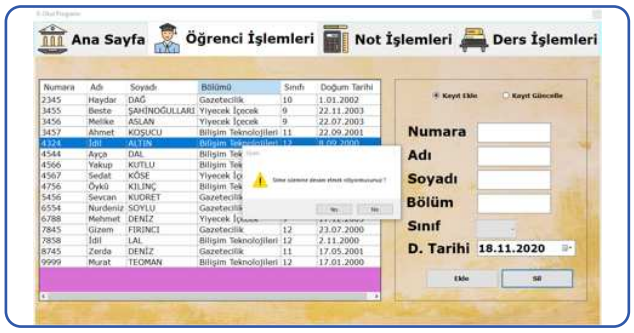
```
private void btnOgrSil_Click(object sender, EventArgs e)
    {
        DialogResult cikis = new DialogResult();//cikis ismiyle nesne türet.
        cikis = MessageBox.Show("Silme işlemine devam etmek istiyor musunuz?", "Uyarı", MessageBox-
Buttons.YesNo, MessageBoxIcon.Warning); //Uyarı penceresi göster.
        if (cikis == DialogResult.Yes) //Evete tıklanmışsa
        {
            try
            {
             kmt.Connection = bag.baglan();//Bağlantı yolu verildi.
             kmt.CommandText = "DELETE FROM ogrencibilgi WHERE ogrenciNo = '" + dtgvwOgrenci.Cur-
rentRow.Cells["ogrenciNo"].Value.ToString() + "'";//SQL cümlesi
                kmt.ExecuteNonQuery();//Komutu çalıştır.
                MessageBox.Show("Silme işlemi başarılı");//Mesaj ver.
                OgrenciYukle();//Yeni öğrenci listesini yükle.
            }
            catch//Hata oluşursa
            {
                MessageBox.Show("Silme gerçekleşmedi!\nSilinecek öğrencinin diğer alanlarda\nkaydının
olmamasına dikkat ediniz."); //Hata mesajı
            }
        }
    }
```

DialogResult türünde tanımlanan cikis değişkeni ile bu onay penceresinden gelen Evet (Yes) veya Hayır (No) ifadelerine göre işlem yapılmalıdır.

Bu onay ekranında "**Evet**" gelirse silme işlemi yapılacaktır. Veri tabanları arasındaki ilişkilerden dolayı silmek istenilen veri başka bir tabloda kullanılmışsa oradaki verilerde boşluk oluşmaması için silme işlemine izin verilmeyecektir. Bir öğrencinin notlar tablosunda kaydı varsa öğrencinin silinmesi istenildiğinde aradaki ilişkiden dolayı notlar tablosunda bu kişi bulunduğu için hata mesajı alınır. Böyle bir yapı olmasa ve öğrenci silinebilse notlar tablosunda artık öyle bir öğrenci olmadığı için çöküş yaşanır. **Bu yüzden silme işlemlerinde kontroller iyi yapılmalıdır.**

Görsel 6.108: DialogResult penceresi görünüşü

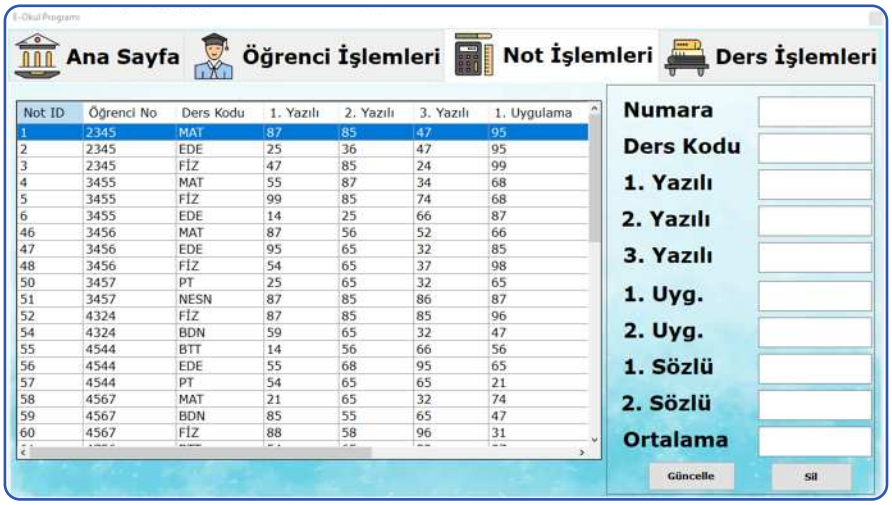
Görsel 6.109'da 4324 numaralı öğrencinin notlar tablosunda notları olduğu için silinmesine izin verilmemiştir.



Görsel 6.109: Kayıt silme hatası

6.14.2. Notlar Sekmesi

Notlar sekmesinde ekleme işlemi ayrı formda yapıldığı için sadece güncelleme işlemine ihtiyaç vardır (Görsel 6.110).



Görsel 6.110: Notlar sekmesi tasarımı

6.14.2.1. Notlar Sekmesi Kayıt Güncelleme

Öğrenci sekmesindeki kodların benzerleri burada da kullanılır. Önce **datagridview**'e çift tıklanarak değerler textbox'lara alınır.

Not: Burada 0. sütun güncellenmeyeceği için alınmamış ve veri alımına 1. sütundan başlanmıştır.

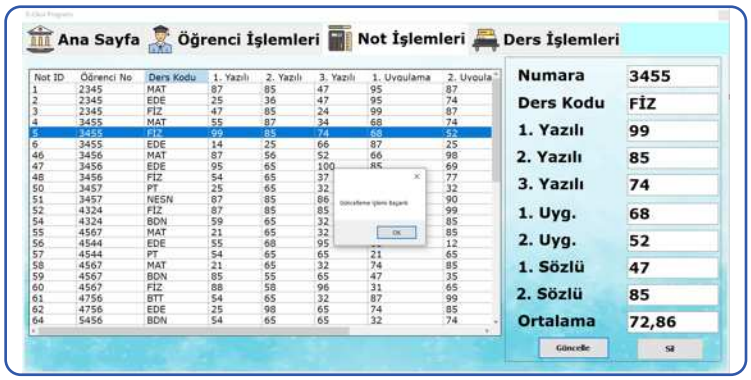
```
private void dtgvwNot_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
    {
  //Buradaki işlem textbox'lara veri aktarmaktır.
      txtNotNo.Text = dtgvwNot.Rows[e.RowIndex].Cells[1].Value.ToString();
      txtNotDers.Text = dtgvwNot.Rows[e.RowIndex].Cells[2].Value.ToString();
      txtNot1Yaz.Text = dtgvwNot.Rows[e.RowIndex].Cells[3].Value.ToString();
      txtNot2Yaz.Text = dtgvwNot.Rows[e.RowIndex].Cells[4].Value.ToString();
      txtNot3Yaz.Text = dtgvwNot.Rows[e.RowIndex].Cells[5].Value.ToString();
      txtNot1Uyg.Text = dtgvwNot.Rows[e.RowIndex].Cells[6].Value.ToString();
      txtNot2Uyg.Text = dtgvwNot.Rows[e.RowIndex].Cells[7].Value.ToString();
      txtNot1Soz.Text = dtgvwNot.Rows[e.RowIndex].Cells[8].Value.ToString();
      txtNot2Soz.Text = dtgvwNot.Rows[e.RowIndex].Cells[9].Value.ToString();
      txtNotOrtalama.Text = dtgvwNot.Rows[e.RowIndex].Cells[10].Value.ToString();
    }
```

Daha sonra **UPDATE** sorgusu yazılır.

```
private void btnNotGuncelle_Click(object sender, EventArgs e)
    {
        double ortalama=0;
        try
        {
//Komut cümlesi parametreler ile yazılacak ve aşağıdaki parametrelere değer atanacaktır.
            ortalama = (Convert.ToDouble(txtNot1Yaz.Text) + Convert.ToDouble(txtNot2Yaz.Text) + Convert.ToDouble(txtNot3Yaz.Text) + Convert.ToDouble(txtNot1Uyg.Text) + Convert.ToDouble(txtNot2Uyg.Text) + Convert.ToDouble(txtNot1Soz.Text) + Convert.ToDouble(txtNot2Soz.Text)) / 7;
            string sorgu = "UPDATE ogrencinot SET ogrNoFK= @no , dersKoduFK= @derskodu , yazili1= @yaz1 , yazili2= @yaz2, yazili3= @yaz3 , uygulama1= @uyg1 , uygulama2= @uyg2 , sozlu1= @soz1 , sozlu2= @soz2 , ortalama= @ort  where ogrenciNotID= @ogrid ";
            kmt = new MySqlCommand(sorgu, bag.baglan());
                    //Parametrelere değer atanması
            kmt.Parameters.AddWithValue("@no", txtNotNo.Text);
            kmt.Parameters.AddWithValue("@derskodu", txtNotDers.Text);
            kmt.Parameters.AddWithValue("@yaz1", txtNot1Yaz.Text);
            kmt.Parameters.AddWithValue("@yaz2", txtNot2Yaz.Text);
            kmt.Parameters.AddWithValue("@yaz3", txtNot3Yaz.Text);
            kmt.Parameters.AddWithValue("@uyg1", txtNot1Uyg.Text);
            kmt.Parameters.AddWithValue("@uyg2", txtNot2Uyg.Text);
            kmt.Parameters.AddWithValue("@soz1", txtNot1Soz.Text);
            kmt.Parameters.AddWithValue("@soz2", txtNot2Soz.Text);
            kmt.Parameters.AddWithValue("@ort", Math.Round(ortalama,2));//Virgülden sonra 2 basamak
            kmt.Parameters.AddWithValue("@ogrid", dtgvwNot.CurrentRow.Cells["ogrenciNotID"].Value.ToString());
            kmt.Connection = bag.baglan();
            kmt.ExecuteNonQuery();
            kmt.Connection.Close();
            MessageBox.Show("Güncelleme İşlemi Başarılı");
            NotYukle();
        }
        catch
        {
            MessageBox.Show("Güncelleme Başarısız!\nLütfen Alanları Kontrol Ediniz.\nAynı numarada 2 öğrenci olamaz.\nBoş alan olamaz.");
        }
    }
```

Not: Güncelleme yapılırken öğrenci numarası veya ders kodu, öğrenci tablosunda ya da ders tablosunda olmayan değerlerden yazılırsa tablolar ilişkisel olmasından dolayı hata verecektir.

Kodlar neredeyse aynıdır sadece bu sekmede notlar olduğu için biraz daha fazla veri aktarımı mevcuttur. Notlar değiştirildiğinde doğal olarak ortalama da değişeceği için ortalama tekrar hesaplanmıştır. MySQL'de veri aktarıldığında ortalama değeri "**Default**" olarak "0" girilmiştir. Güncelleme işlemi yapılarak "0" olan değerlerin gerçek değerleri hesaplatılabilir. Burada try-catch yapısı da bulunmaktadır (Görsel 6.111).



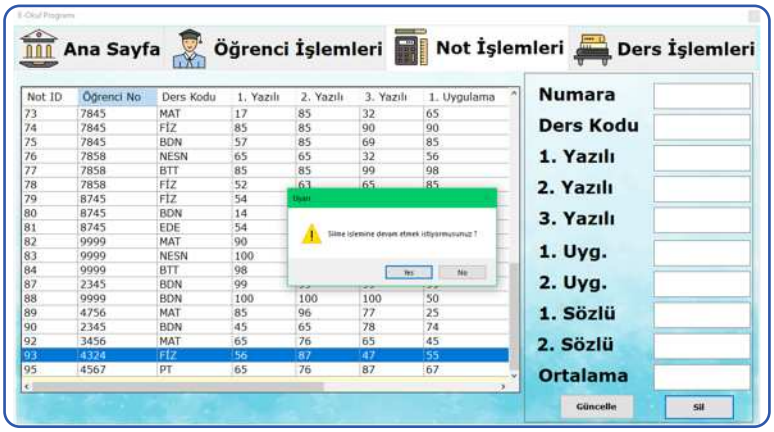
Görsel 6.111: Güncelleme işleminin sonucu

Not: Güncelleme işlemi sonucunda ortalama sütunu kontrol edilir.

6.14.2.2. Notlar Sekmesi Kayıt Silme

Aşağıdaki kodlara göre kayıt silme işlemi gerçekleştirilir.

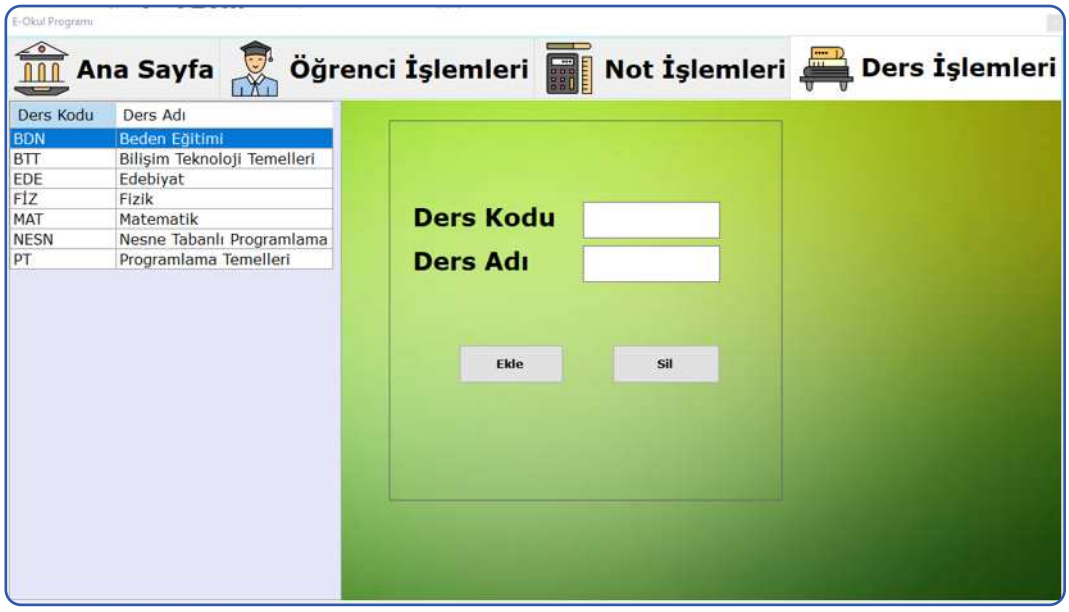
```
private void btnNotSil_Click(object sender, EventArgs e)
    {
        DialogResult cikis = new DialogResult();//cikis adında nesne türetilir.

        cikis = MessageBox.Show("Silme işlemine devam etmek istiyor musunuz?", "Uyarı", MessageBox-
Buttons.YesNo, MessageBoxIcon.Warning); //Uyarı penceresi

        if (cikis == DialogResult.Yes) //Evete tıklanmışsa
        {
            try
            {
                kmt.Connection = bag.baglan();//Bağlantı sağlanır.

                kmt.CommandText = "DELETE FROM ogrencinot WHERE ogrenciNotID = '" + dtgvwNot.Cur-
rentRow.Cells["ogrenciNotID"].Value.ToString() + "'";//Silme SQL komutu

                kmt.ExecuteNonQuery();//Komut çalıştırılır.

                MessageBox.Show("Silme işlemi başarılı");//Başarılı işlem mesajı

                NotYukle();//Notların yeni hâlinin yüklenmesi
            }
            catch//Hata varsa
            {
                MessageBox.Show("Silme gerçekleşmedi!\nLütfen kontrol ediniz.");
            }
        }
    }
```



Görsel 6.112: Notlar sekmesi kayıt silme işlemi

6.14.3. Dersler Sekmesi

Son olarak dersler sekmesine geçilir. Dersler tablosunda iki alan olduğu için işlem çok kısa sürecektir. Sadece iki alan olduğu için güncelleme işlemi yerine ekleme ve silme işlemleride kullanılabilir (Görsel 6.113)



Görsel 6.113: Dersler sekmesi form tasarımı

6.14.3.1. Dersler Sekmesi Kayıt Ekleme

Aşağıdaki kodlara göre ders ekleme işlemi gerçekleştirilir.

```
private void btnDersEkle_Click(object sender, EventArgs e)
    {
        kmt.Connection = bag.baglan();//Bağlantı yolu verildi.
        kmt.CommandText = "INSERT INTO dersler(derskodu, dersadi) VALUES ('" + txtDersKodu.Text + "', '"
+ txtDersAdi.Text + "')";//Kayıt için SQL cümlesi
        kmt.ExecuteNonQuery();//Komut çalıştırıldı.
        kmt.Connection.Close();//Bağlantı kapatılır.
        DersYukle();//Derslerin yeni hâli yüklendi.
        MessageBox.Show("Bilgiler veri tabanına başarı ile eklendi.");//İşlem başarılı mesajı verilir.
    }
```

6.14.3.2.Dersler Sekmesi Kayıt Silme

Aşağıdaki kodlara göre ders silme işlemi gerçekleştirilir.

```
private void BtnDersSil_Click(object sender, EventArgs e)
    {
        DialogResult cikis = new DialogResult();//cikis adında nesne türetilir.
        cikis = MessageBox.Show("Silme işlemine devam etmek istiyor musunuz?", "Uya-
rı", MessageBoxButtons.YesNo, MessageBoxIcon.Warning); //Uyarı penceresi

        if (cikis == DialogResult.Yes) //Evete tıklanmışsa
        {
            try
            {
                kmt.Connection = bag.baglan();//Bağlantı yolu verildi.
                kmt.CommandText = "DELETE FROM dersler WHERE derskodu = '" + dtgvw-
Ders.CurrentRow.Cells["derskodu"].Value.ToString() + "'";//Silme SQL komut cümlesi
                kmt.ExecuteNonQuery();//Komut çalıştırıldı.
                MessageBox.Show("Silme işlemi başarılı");//Başarılı işlem mesajı
                DersYukle();//Derslerin son hâlinin yüklenmesi
            }
            catch//Hata varsa
            {
                MessageBox.Show("Silme gerçekleşmedi!\nLütfen dersin diğer alanlarda\
nKullanılmadığından emin olunuz."); //Hata mesajı
            }
        }
    }
```

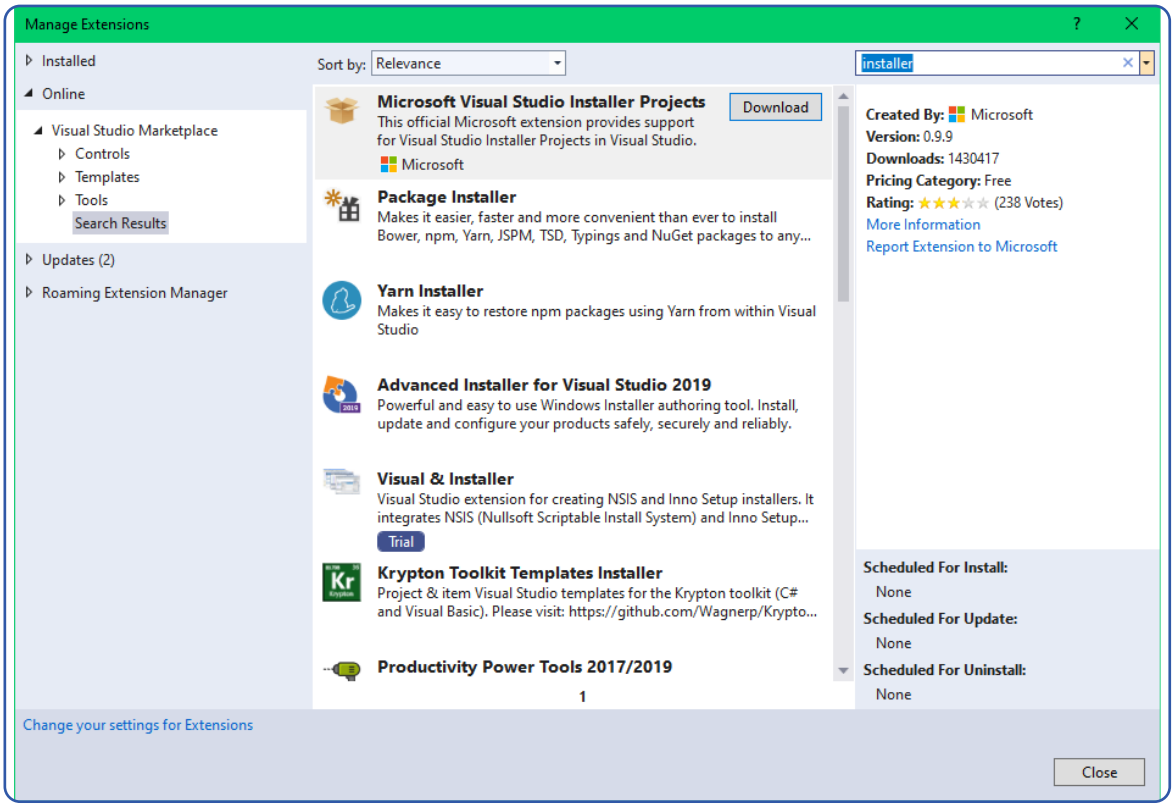
6.15. Kurulum (Setup) Hazırlama

Hazırlanan projenin bir başka bilgisayarda kullanılabilmesi için proje "**Setup**" (**Kurulum**) hâline getirilmelidir. Kod editörü bu imkânı da sağlamaktadır.

Not: Bu tür işlemlerde diğer bilgisayarda da MySQL Server olmalıdır ve projedeki veri tabanı kurulum yapılacak bilgisayarın veri tabanına import edilmelidir. MySQL üzerinden "Export" ve "Import" işlemleri daha önce anlatılmıştır (6.8. MySQL Veri Tabanı Alma ve Yükleme).

Setup hazırlamanın adımlarına geçilir.

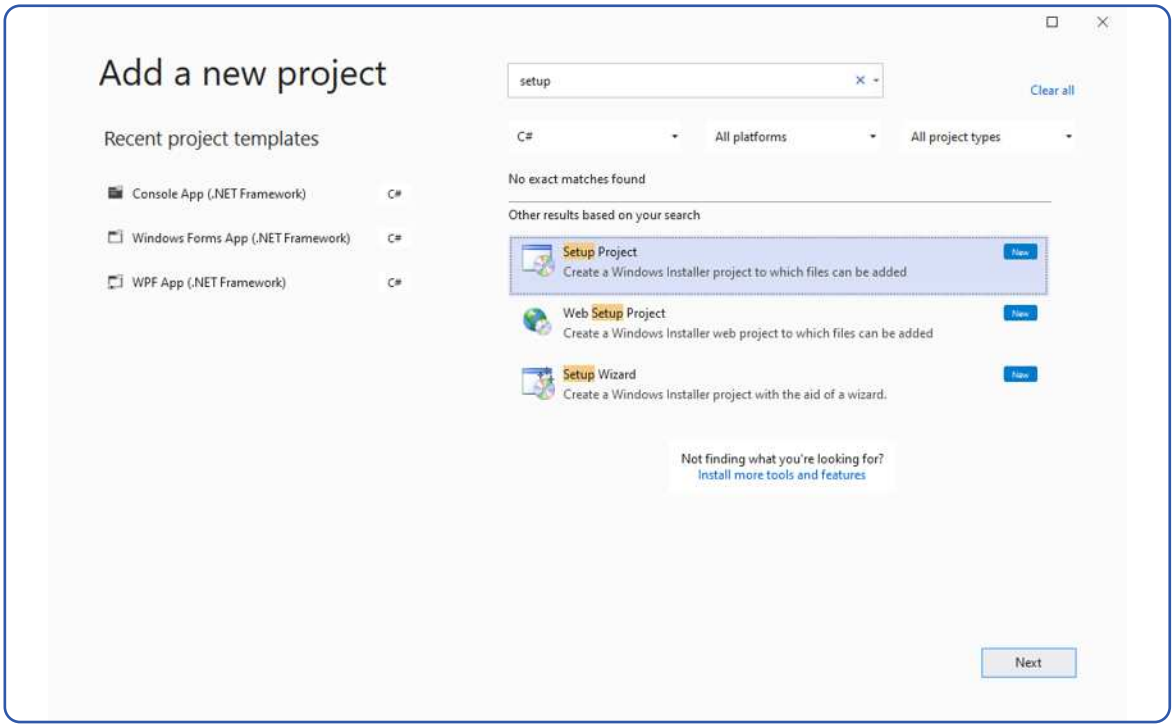
İlk olarak kod editörü programına Setup bileşeni kurulmalıdır. Bunun için üstteki menüden "**Extension → Manage Extensions**" bölümüne girilir. Gelen pencerenin arama kısmına "**Installer**" veya "**Setup**" yazılır. Çıkan sonuçlardan "**Installer Projects**" seçilerek **Download** düğmesine basılır (Görsel 6.114). Kurulum bittikten sonra yüklemenin başlaması için kod editörü programı kapatılır.



Görsel 6.114: Projeye Setup projesi ekleme

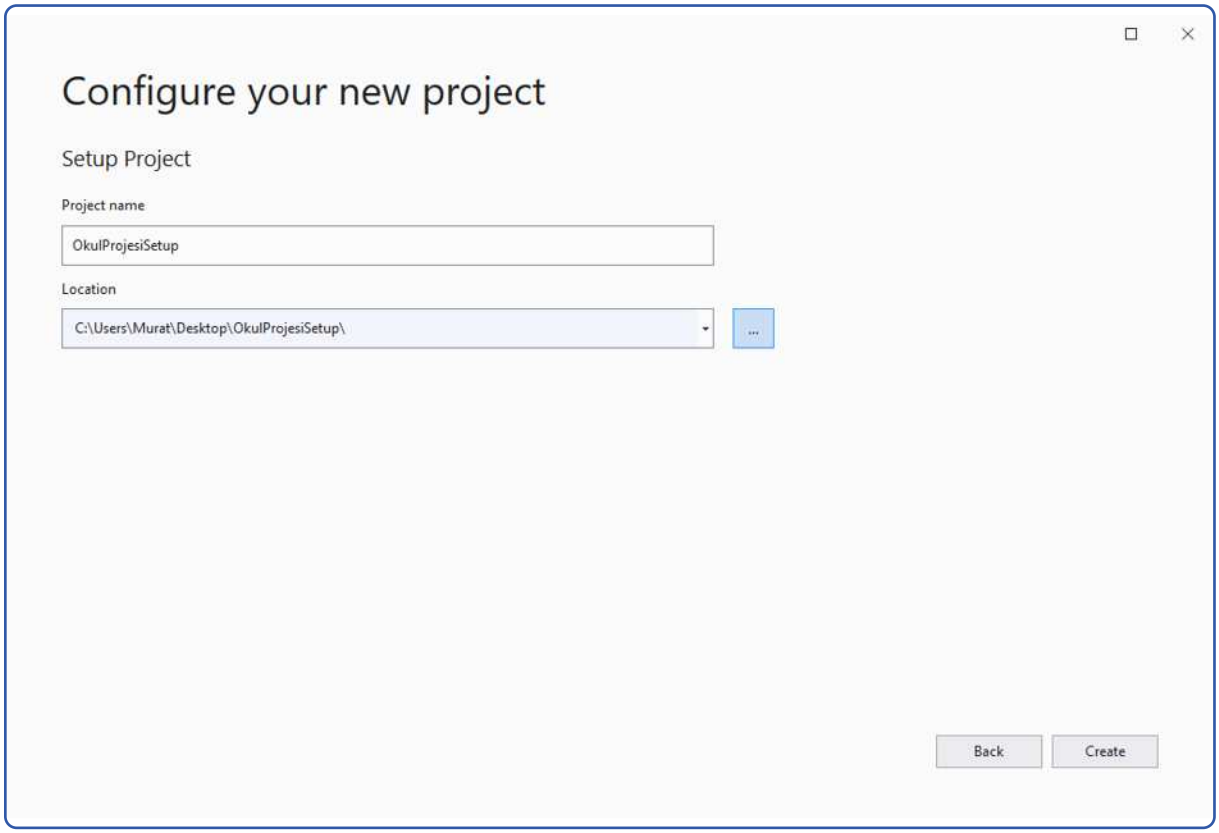
Yükleme bittikten sonra kod editörüne Setup özelliği eklenecektir. Proje açılır ve menüden

"File → Add → New Project" ile projeye eklenecek Setup projesi seçilir. Setup görünmezse yukarıdaki arama çubuğuna "Setup" yazılabilir. Gelen pencereden Setup Project seçilerek ikinci adıma geçilir (Görsel 6.115).



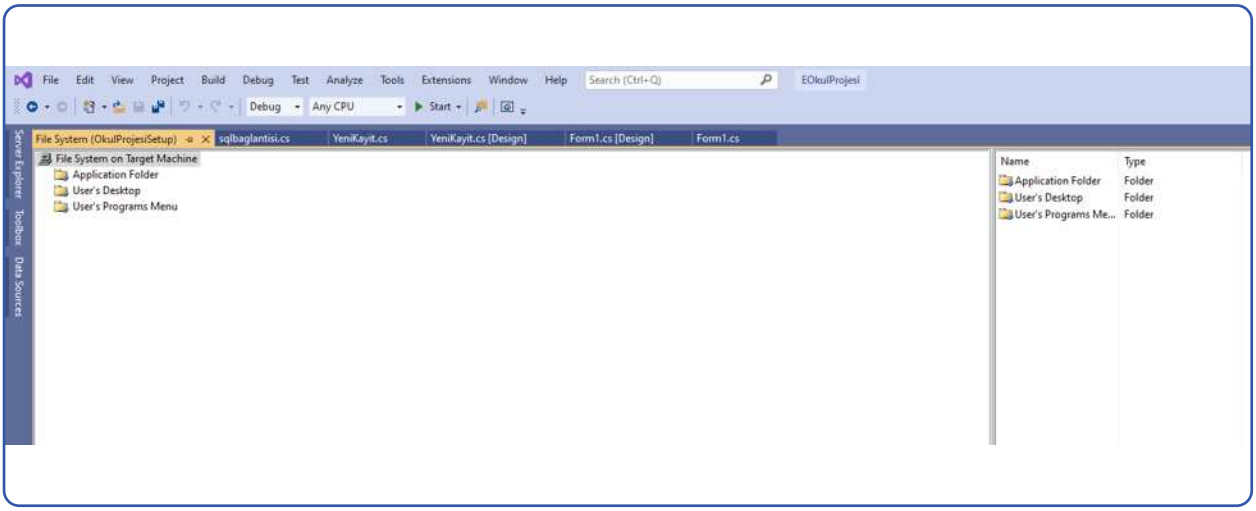
Görsel 6.115: Projeye Setup projesi ekleme

Gelen pencerede projeye eklenecek Setup projesinin ayarları yapılır. Burada istenen bir isim ve yer belirlenebilir. Bu, ana projeden ayrı bir bölümdür. Örneğin, Setup projesi masaüstündeki bir klasöre çıkartılmak istendiğinde bir masaüstü klasörü oluşturulup o seçilir (Görsel 6.116).



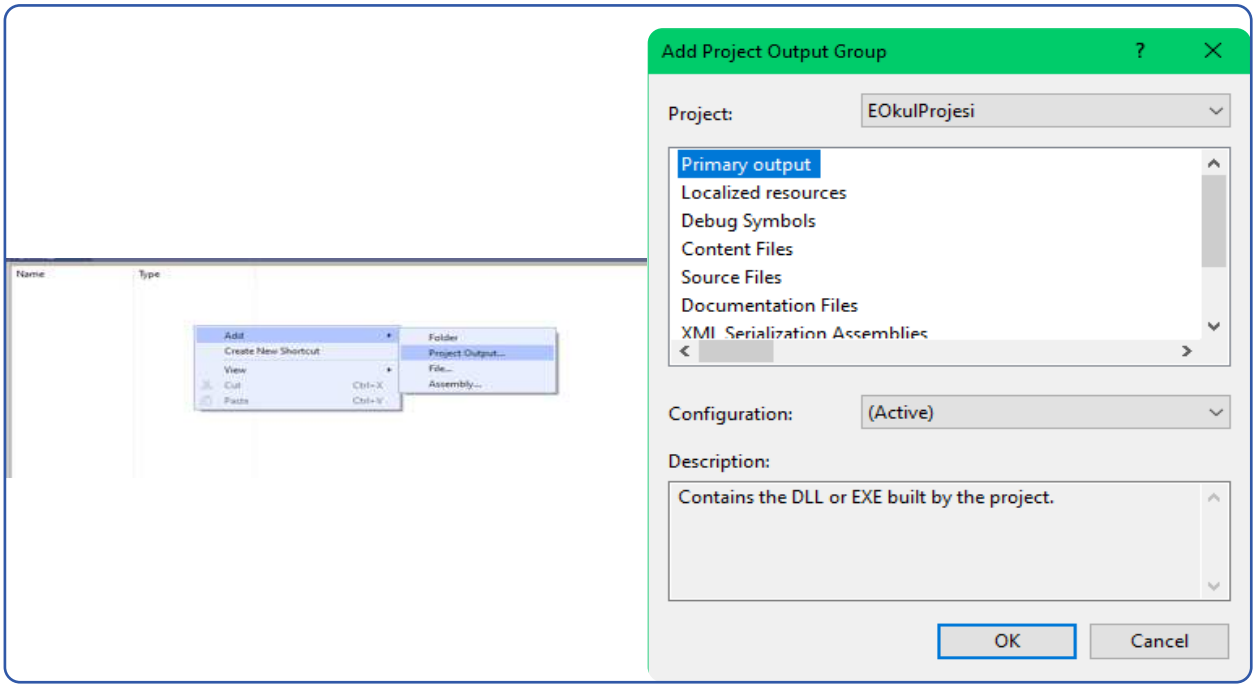
Görsel 6.116: Setup projesi ayarları

Setup projesi, projeye eklendi ve Setup projesinin ayarlarının yapılacağı kısımlar ekrana geldi. Solution Explorer penceresine dikkat edilirse Setup projesinin eklendiği görülür (Görsel 6.117). Projenin ayarlarına geçilebilir.



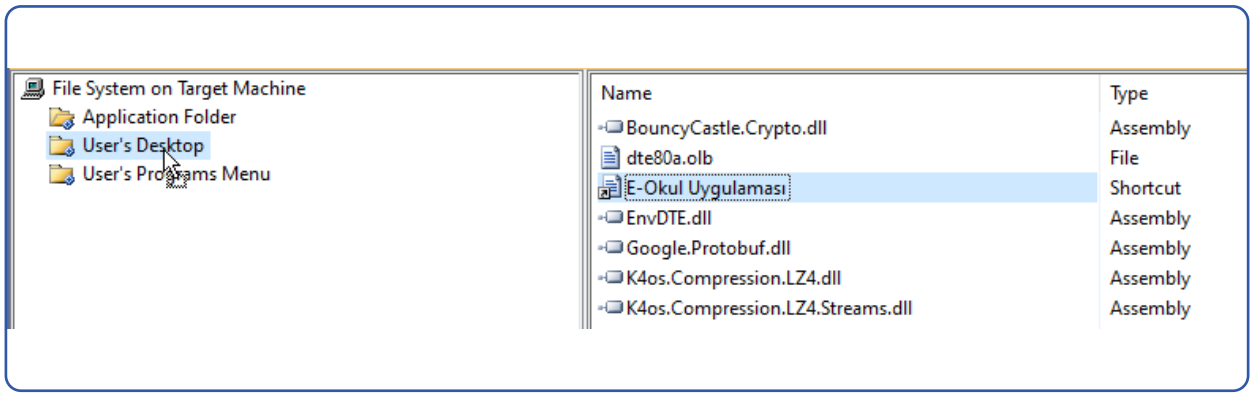
Görsel 6.117: Setup proje işlem penceresi

Burada ilk yapılması gereken işlem, projenin dosyalarını Setup projesine eklemektir. Sağdaki bölümde "**Application Folder**" çift tıklanarak giriş sağlandıktan sonra "Add → Project Output" seçeneği seçilir. Gelen ekranda "**Primary output**" seçilir (Görsel 6.118). Bu işlemden sonra projenin çalışması için yüklenen dosyaların tamamı Setup'a eklenecektir.



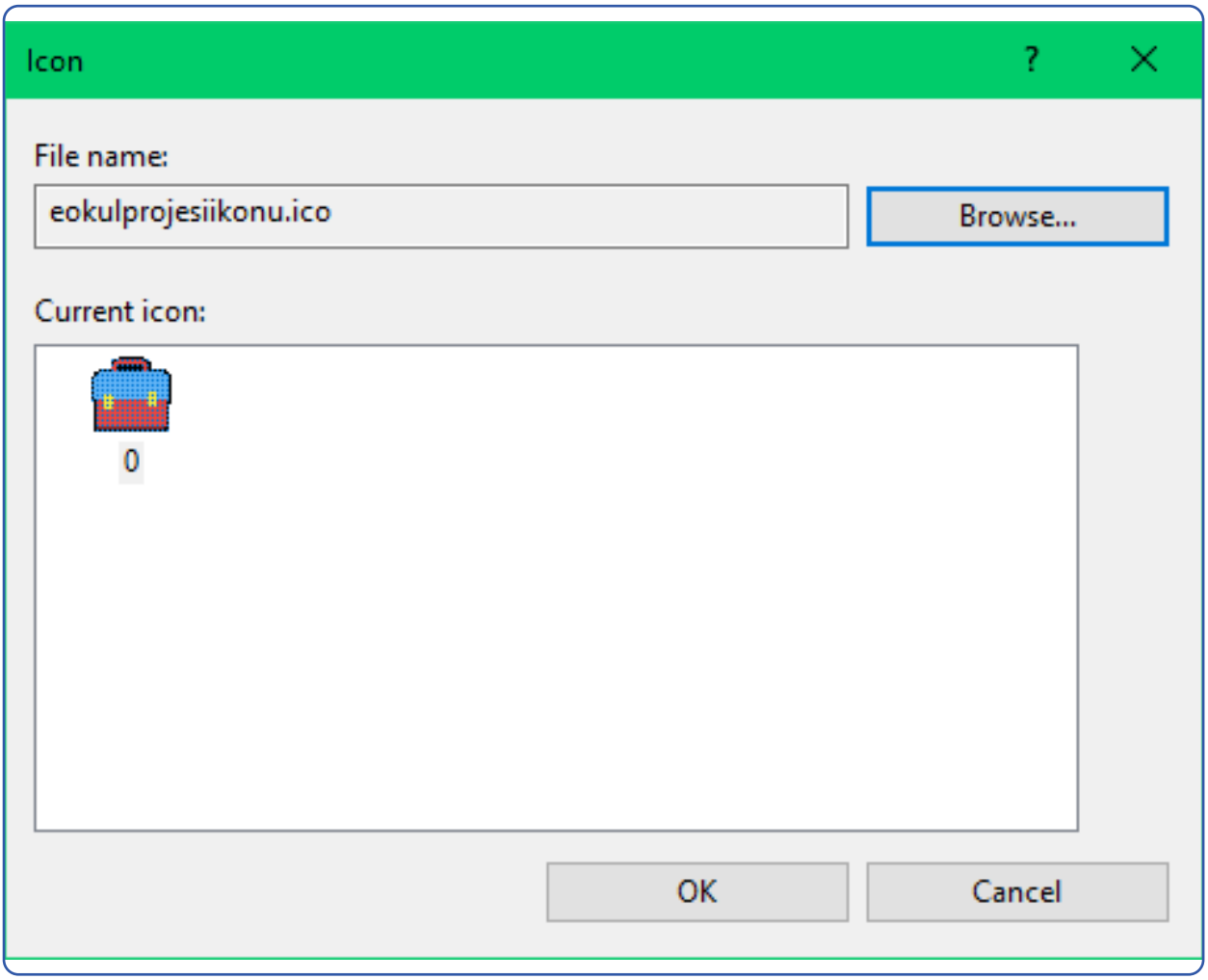
Görsel 6.118: Setup projesine ana projenin bileşenlerini ekleme

Setup'a eklenen ana dosyadan (.exe dosyası) iki kısayol oluşturulup bunlar kurulum yapılan bilgisayarın masaüstü ve programlar menüsüne aktarılır. Bunun için eklenen dosyalardan **"Primary output from EokulProjesi (Active)"** dosyasına sağ tıklanarak "**Create Shortcut ...**" seçeneği ile iki kısayol oluşturulur. Bu kısayollara proje adının verilmesi unutulmamalıdır çünkü kısayollar masaüstüne ve başlat menüsüne eklenecektir. Bu kısayollar, isimler de verildikten sonra sürükle bırak ile "**User's Desktop**" ve "**User's Programs Menu**" içine taşınır (Görsel 6.119).



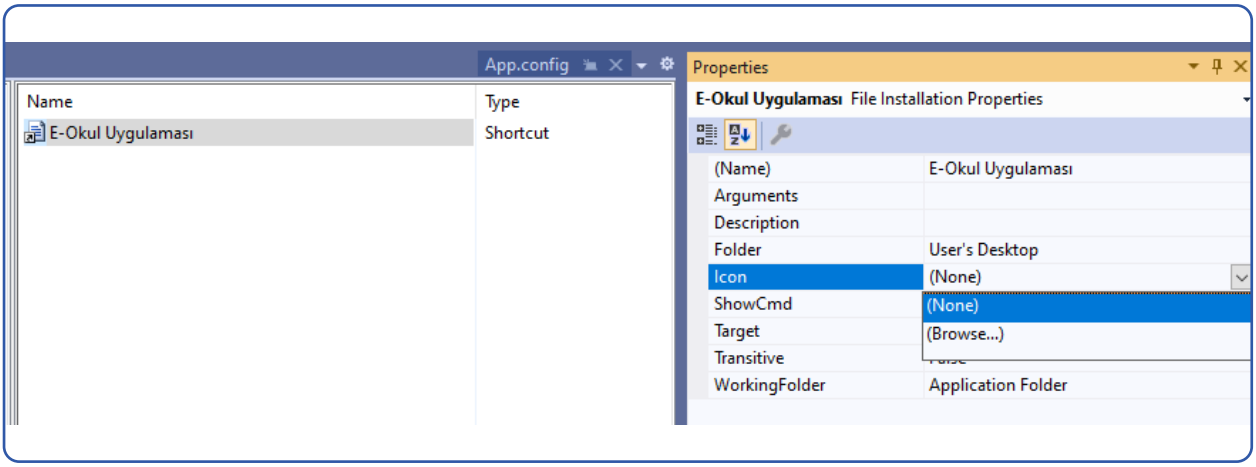
Görsel 6.119: Oluşturulan kısayolların taşınması

Setup'a ikon eklenmesi işleminde öncelikle ikon dosyası da Setup içine alınmalıdır. Bunun için yine "**Application Folder**" içinde sağ tıklandıktan sonra "Add → File" seçeneği ile ikon seçilir ve Setup projesine dâhil edilir (Görsel 6.120).



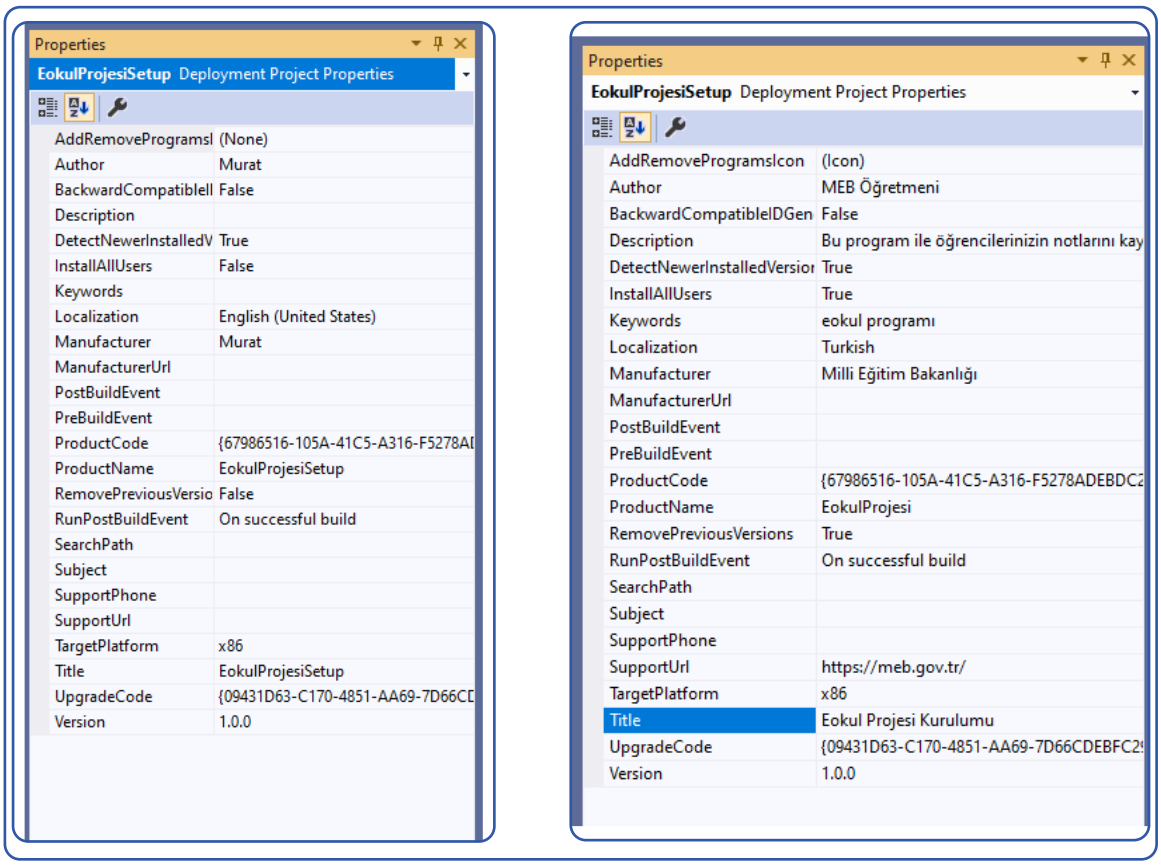
Görsel 6.120: Setup projeye ikon eklenmesi

Belirlenen ikonun eklendiği kısayollar, propertis kısmından Icon seçeneği ile seçilir (Görsel 6.121).



Görsel 6.121: Eklenen ikonun kısayollara atanması

Setup dosyasının ayarları için **Solution Explorer** penceresinden Setup projesi seçilir ve **properties** penceresine geçilir (Görsel 6.122).



Görsel 6.122: Setup projesi bilgiler penceresi öncesi ve sonrası

- Setup projesinin ayarları şu şekilde sıralanabilir:
- **AddRemoveProgramsIcon:** Program ekle / kaldır kısmında görünen simge
- **Author:** Programın kodlayıcısı
- **DetectNewerInstalledVersion:** Programın daha yeni sürümü yüklü mü kontrol et.
- **InstallAllUsers:** Program bütün kullanıcılar için kurulsun mu?
- **Keywords:** Programla ilgili anahtar kelimeler
- **Localization:** Bölgesel ayarlar (Localization, program Türkiye'de kullanacağı için Türkiye seçilir.)
- **Manufacturer:** Projeyi yapan firma adı
- **ManufacturerUrl:** Projeyi yapan firmanın web adresi
- **ProductName:** Projenin adı
- **RemovePreviousVersions:** Eski sürümü varsa kaldır.
- **Subject:** Programın kısa özeti
- **SupportPhone:** Program desteği için telefon
- **SupportUrl:** Program desteği için web adresi
- **TargetPlatform:** Programın desteklediği işlemci mimarisi
- **Title:** Programın başlığı
- **Version:** Programın versiyonu

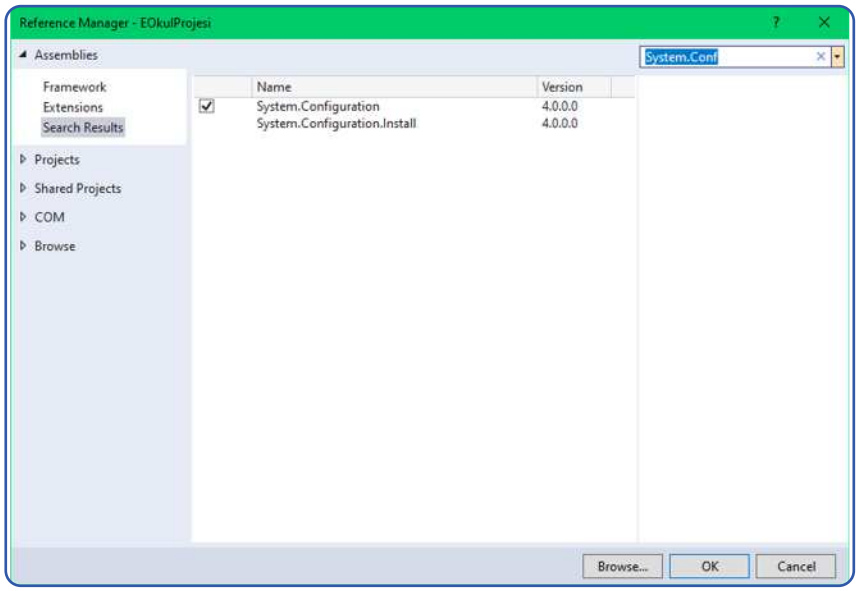
Program başka bilgisayara kurulurken MySQL Server ayarları çok önemlidir. Bu ayarlar bütün bilgisayarlarda aynı değildir. O yüzden program kurulduktan sonra bu bağlantıyı değiştirmek için bazı düzenlemeler yapılmalıdır.

Önce Solution Explorer penceresine gidilip App.config dosyası açılır ve Connection String cümlesi Görsel 6.123'teki gibi **<connectionStrings></connectionStrings>** yazılarak içine eklenir.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <connectionStrings>
    <add name="Eokul_Connection_String"
         connectionString="Server=localhost;Database=eokul;Uid=root;Pwd=milliegitim;"/>
  </connectionStrings>
```

Görsel 6.123: Setup projesine Connection String eklenmesi

Daha sonra projeye referans olarak Görsel 6.124'teki gibi **System.Configuration** eklenir.



Görsel 6.124: System.Configuration referansı ekleme

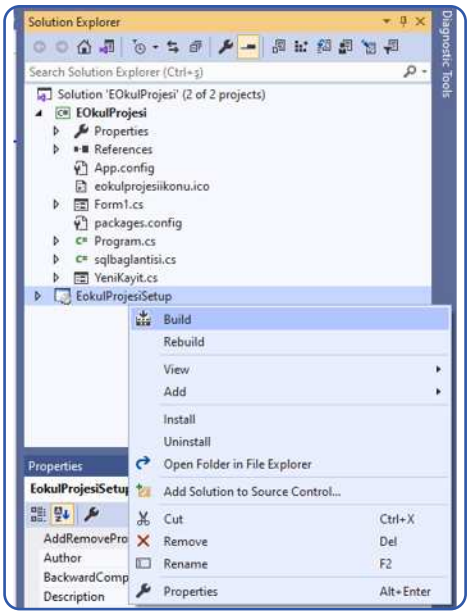
Connection String'in olduğu sqlbaglantisi.cs dosyasına gidilip "**using System.Configuration**" kütüphanesi eklenir, ardından **App.config** içine yazılan **Connection String**'in **Name**'i eklenir.

```
public MySqlConnection baglan() // class içinde baglan adında fonksiyon oluşturuldu.
    {
        MySqlConnection baglanti = new MySqlConnection(ConfigurationManager.ConnectionStrings["Eokul_Connection_String"].ConnectionString);// Bağlantı yolu
        baglanti.Open();//Bağlantı açıldı.
        MySqlConnection.ClearPool(baglanti);//Önceki bağlantılar temizlendi.
        return (baglanti);//Çağrıldığı yere bağlantı gönderildi.
    }
```

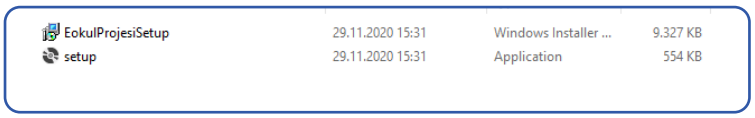
Connection String, kurulan bilgisayarın MySQL özelliklerine göre (ad ve şifre gibi) sonradan değiştirebilir ve bu programın doğru çalışması için çok önemlidir.

Artık Setup projesi derlenebilir. Görsel 6.125'teki gibi Setup projesine gidilerek sağ tıklanır ve Build seçeneği işaretlenir.

Build, en başta seçilen yerde iki tane Setup dosyası oluşturacaktır (Görsel 6.126).

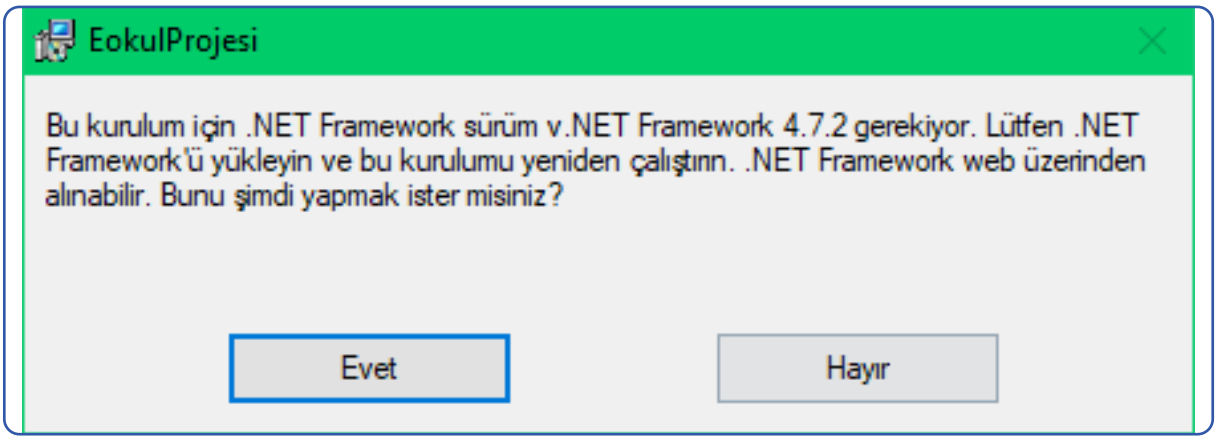


Görsel 6.125: Setup projesinin derlenmesi



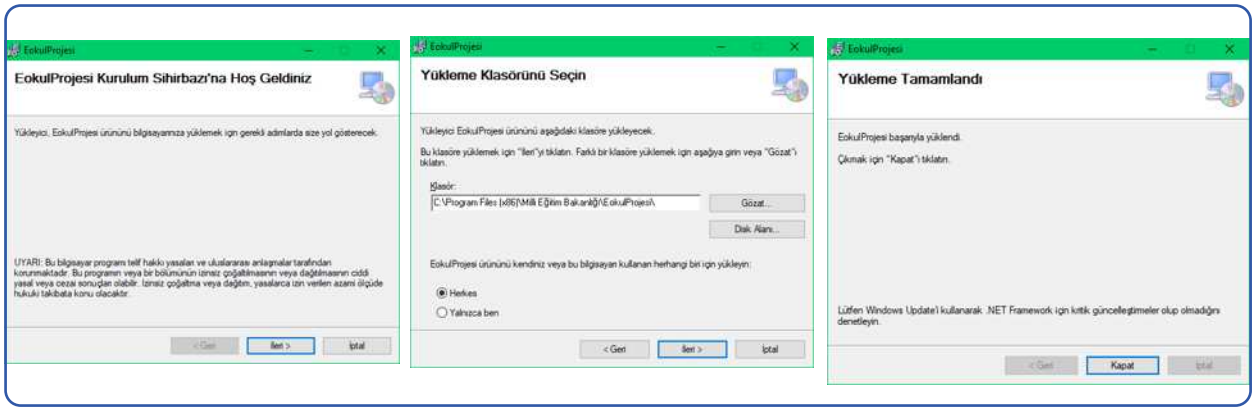
Görsel 6.126: Derleme sonucu oluşan kurulum dosyaları

Kuruluma geçilir. İstenen bir Setup dosyası çalıştırılır ve işlem adımları takip edilir. Gerekli bir bileşen kurulumda eksikse program Görsel 6.127'deki gibi uyarı verir.



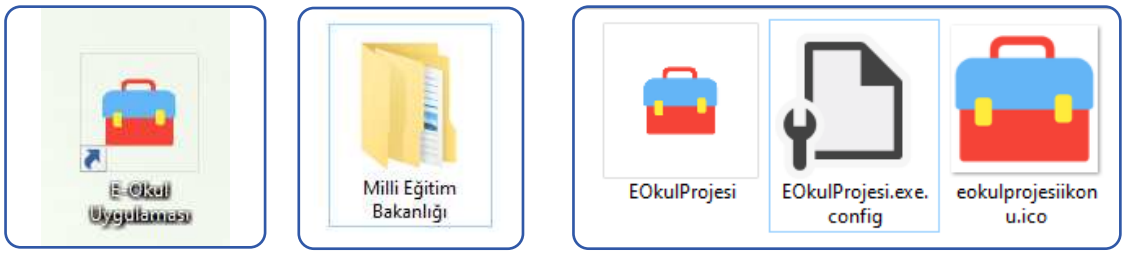
Görsel 6.127: Uygulamanın kurulumunda yapılan kontroller sonucu eksik bileşen yüklenmesi

Gerekli kurulum yapılıp Görsel 6.128'deki gibi adımlara devam edilir.



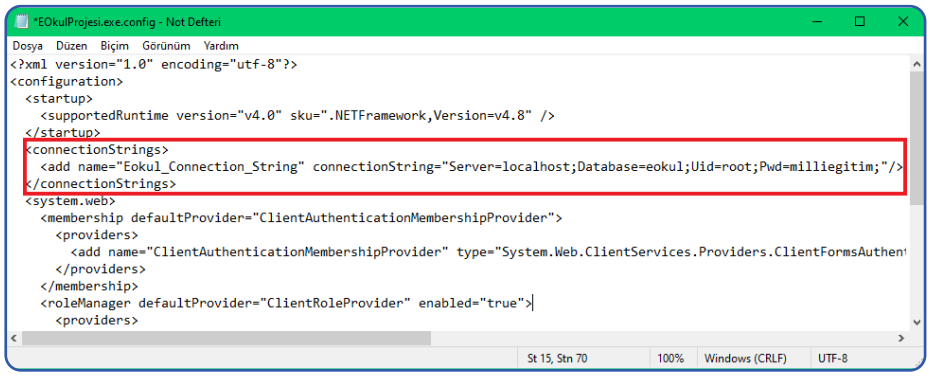
Görsel 6.128: Uygulamanın kurulum adımları

Kurulum başarıyla gerçekleştikten sonra Görsel 6.129'daki gibi simgeler oluşur.



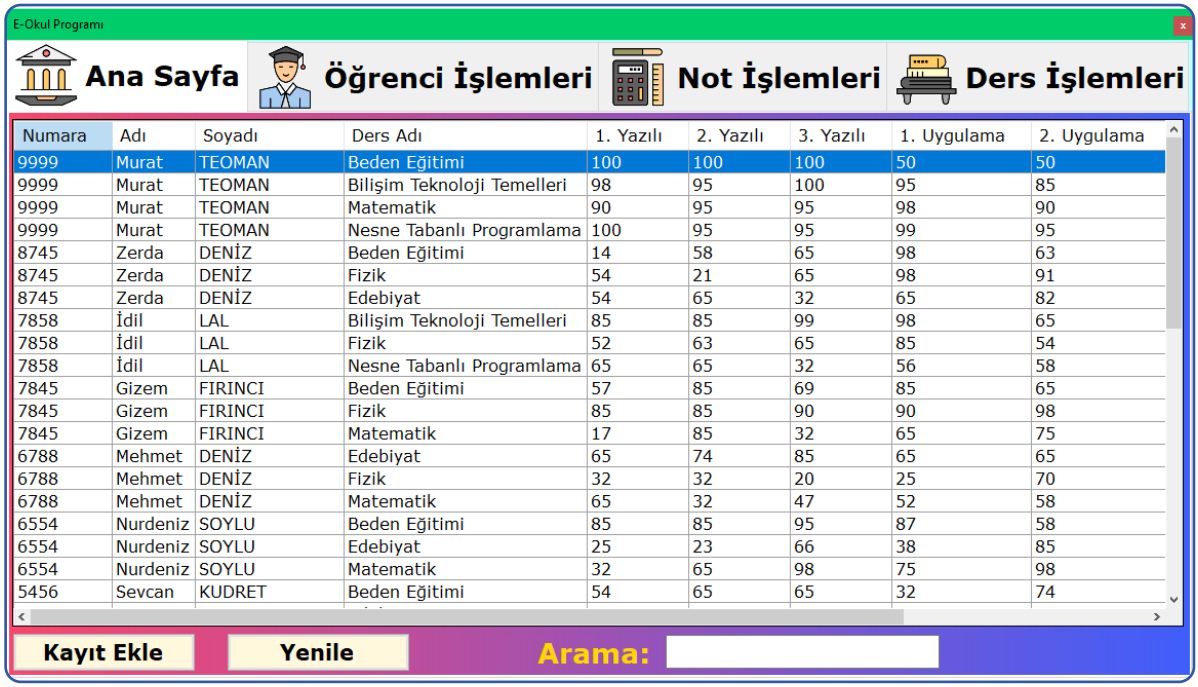
Görsel 6.129: Uygulamanın kurulumu sonucu masaüstü ve program files klasöründe oluşan simgeler ve dosyalar

Bu noktada **EOkulProjesi.exe.config** dosyasının içinde yer alan Connection String'in kurulum yapılan bilgisayarın Server bilgilerine göre (sunucu adı, kullanıcı adı, şifre) güncellenmesi gerekmektedir (Görsel 6.130).



Görsel 6.130: Kurulum sonrası Connection String değiştirme

Masaüstünde oluşan  kısayolu çift tıklanarak proje çalıştırılabilir.



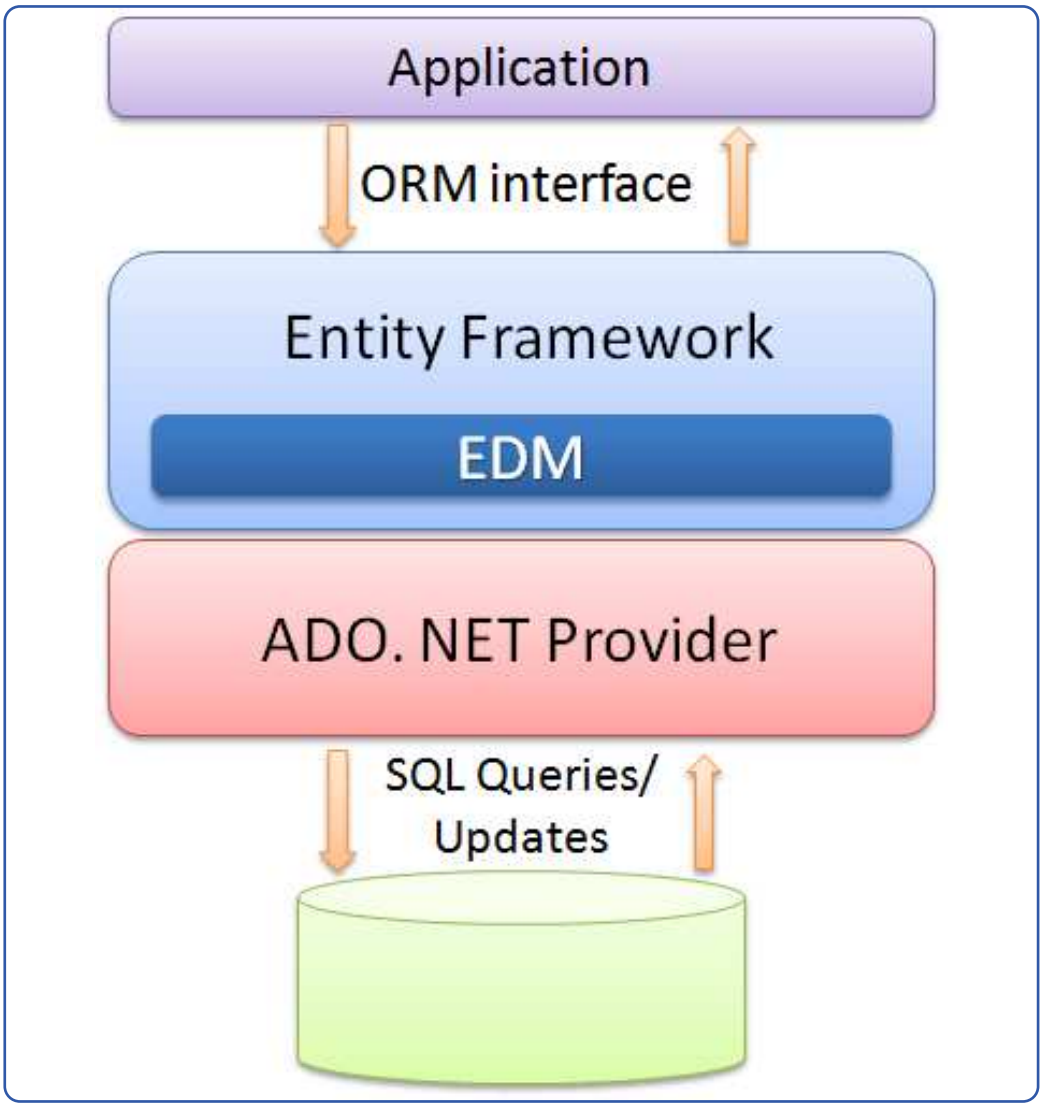
Numara	Adı	Soyadı	Ders Adı	1. Yazılı	2. Yazılı	3. Yazılı	1. Uygulama	2. Uygulama
9999	Murat	TEOMAN	Beden Eğitimi	100	100	100	50	50
9999	Murat	TEOMAN	Bilişim Teknoloji Temelleri	98	95	100	95	85
9999	Murat	TEOMAN	Matematik	90	95	95	98	90
9999	Murat	TEOMAN	Nesne Tabanlı Programlama	100	95	95	99	95
8745	Zerda	DENİZ	Beden Eğitimi	14	58	65	98	63
8745	Zerda	DENİZ	Fizik	54	21	65	98	91
8745	Zerda	DENİZ	Edebiyat	54	65	32	65	82
7858	İdil	LAL	Bilişim Teknoloji Temelleri	85	85	99	98	65
7858	İdil	LAL	Fizik	52	63	65	85	54
7858	İdil	LAL	Nesne Tabanlı Programlama	65	65	32	56	58
7845	Gizem	FIRINCI	Beden Eğitimi	57	85	69	85	65
7845	Gizem	FIRINCI	Fizik	85	85	90	90	98
7845	Gizem	FIRINCI	Matematik	17	85	32	65	75
6788	Mehmet	DENİZ	Edebiyat	65	74	85	65	65
6788	Mehmet	DENİZ	Fizik	32	32	20	25	70
6788	Mehmet	DENİZ	Matematik	65	32	47	52	58
6554	Nurdeniz	SOYLU	Beden Eğitimi	85	85	95	87	58
6554	Nurdeniz	SOYLU	Edebiyat	25	23	66	38	85
6554	Nurdeniz	SOYLU	Matematik	32	65	98	75	98
5456	Sevcan	KUDRET	Beden Eğitimi	54	65	65	32	74

Görsel 6.131: Projenin bitmiş hâli

Görsel 6.131'de kurulumu gerçekleştirilen proje hiçbir hata vermeden çalıştı. Başka bilgisayarlara kurulum yapıldığında hem o bilgisayardaki Server ayarları **config** dosyasına eklenmeli hem de veri tabanları yeni Server'a **import** edilmelidir.

6.16. Entity Framework

Entity Framework (Görsel 6.132), yazılım geliştiricilerin uzun SQL sorguları yazmalarını ortadan kaldırarak bir **Object Relational Mapping** (ORM) imkânı sağlayan framework'tür. ORM ise ilişkisel veri tabanı yönetim sistemlerine doğrudan müdahale yerine nesneler aracılığı ile müdahale edilmesini sağlayan bir köprüdür. Entity Framework ile birlikte **Language Integrated Query** (LINQ) sorguları kullanılarak nesneler üzerinde güçlü bir sorgulama imkânına sahip olunur.

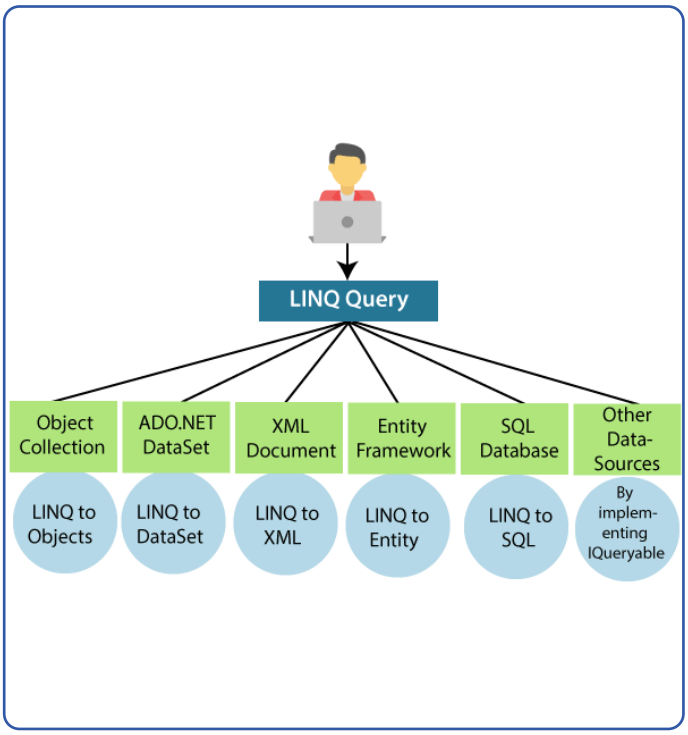


Görsel 6.132: Entity Framework şeması

Entity Framework'ün temel amacı, uygulama geliştiricinin data işlemleri ile çok vakit kaybetmeden uygulama tarafına odaklanmasını sağlamaktır. Örneğin klasik ADO.NET uygulamalarında bir bağlantının açılmasından ve kapatılmasından tamamen geliştiriciler sorumludur ancak Entity Framework kullanıldığında bu tür işlemlere karışılmaz. Sorgu hazırlanır ve Entity Framework aracılığı ile database'e iletilir. Entity Framework teknolojisi, veri tabanlarıyla ilgili birçok konuda işlemleri kullanıcıların yerine yaparak işi kolaylaştırmaktadır. Entity Framework performans olarak ADO.NET teknolojisinden yavaştır. **Kısaca özetlenirse tablolar sınıflara, satırlar nesnelere ve sütunlar değişkenlere dönüşerek projeye entegre edilir.**

Entity Framework yapısını kullanırken LINQ ve Lambda Expressions yapıları da kullanılır.

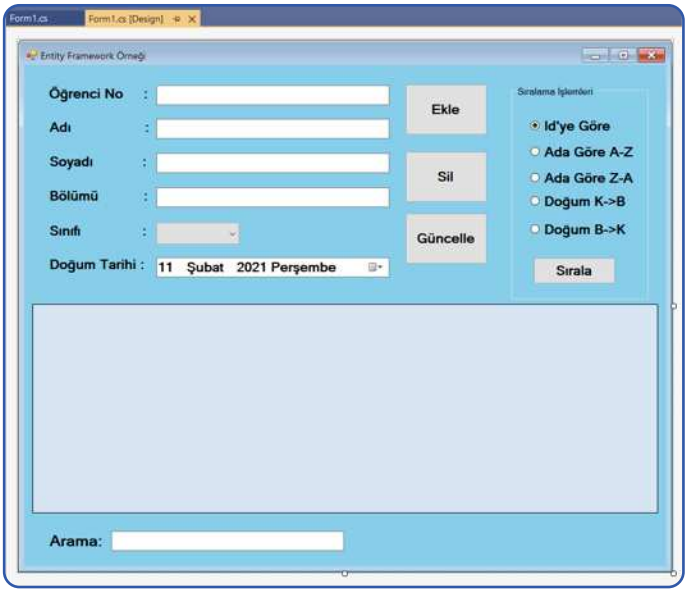
Linq (Language integrated query) dile entegre edilmiş sorgu olarak Türkçeye çevrilebilir. Linq C# ve VB.NET programlama dilleri ile farklı kaynaklardaki veriler üzerinde **CRUD** (Create - Read - Update - Delete) işlemleri yapılmasını sağlar. Böylelikle veri tabanı uyumsuzluklarını ortadan kaldırarak tek bir sorgu arayüzü elde edilmiş olur (Görsel 6.133 Linq şeması). Örneğin olarak SQL dili, veri tabanından veriyi almak, kaydetmek vb. işlemlerde kullanılmak üzere yapılandırılmış bir sorgu dilidir. Linq ise koleksiyonlar, ADO.NET Data Set, XML, Web Servisler, MS SQL Server, Entity Framework ve diğer veri tabanı ve ORM teknolojileri kullanılarak veriler üzerinden CRUD işlemleri yapmak için C# ve VB.NET dilleri için oluşturulmuş bir sorgu söz dizimidir.



Görsel 6.133: LINQ şeması

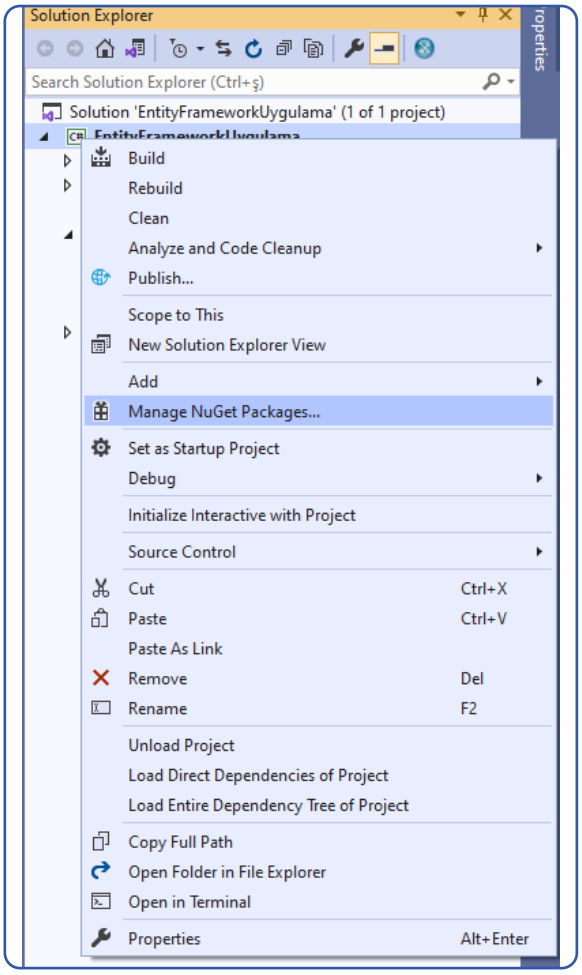
Lambda Expressions (Lambda ifadeleri), yani lambda ifadeleri, verilen filtrelere ve belirlenen kriterlere göre, değişkenlere değer atamaya yarayan fonksiyonlar olarak düşünülebilir. Genellikle diziler veya entity listeleri üzerinde kolayca sorgu yapılmasını sağladığı için, SQL üzerinden yapabilecek pek çok sorgu bu fonksiyonlar sayesinde halledebilir. Önceden oluşturulan "**eokul**" veri tabanını kullanarak yapılacak proje için Görsel 6.134'teki gibi Entity Framework için bir form tasarlanır. Formu tasarlarken "dataGridView" için aşağıdaki işlemler uygulanır.

1. Bütün bileşenlere isim veriniz.
2. Datagridview için şu ayarları yapınız:
   - **SelectionMode** = FullRowSelect
   - **MultiSelect** = False
   - **ReadOnly** = True
   - **EditMode** = EditProgrammatically
   - **AutoSizeColumnsMode** = Fill



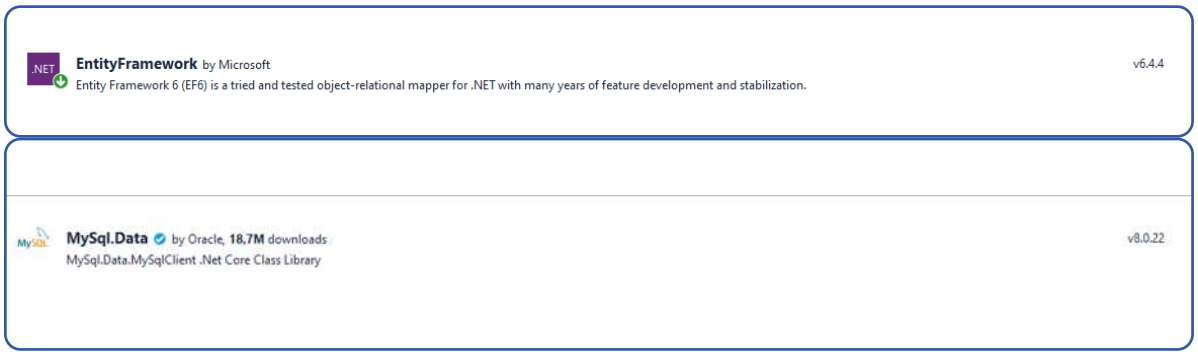
Görsel 6.134: Entity Framework form tasarımı

Form tasarlandıktan sonra projeye gerekli paketler kurulmalıdır. Bu işlem Görsel 6.135'deki gibi Solution Explorer penceresine sağ tıklandıktan sonra "**Manage NuGet Packages**" seçeneği işaretlenerek yapılır.



Görsel 6.135: Nuget paket eklemek

Gelen ekranda iki adet paket kurulumu gerçekleştirilir. Bunun için arama kutusuna sırasıyla "**Entity Framework**" ve "**MySql.Data.EntityFramework**" yazılarak aşağıdaki paketler indirilir (Görsel 6.136).



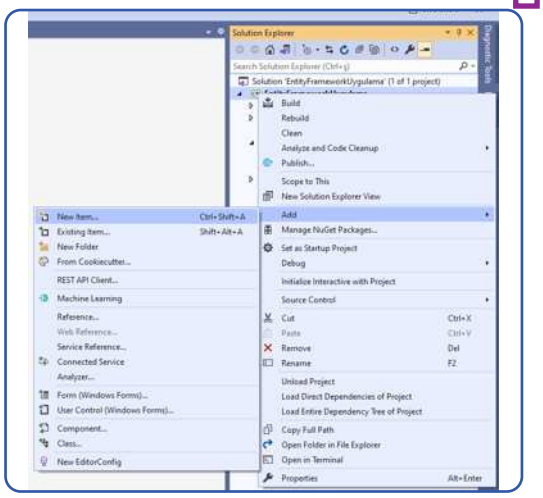
Görsel 6.136: Nuget paketleri kurma

DİKKAT: Benzer isimde birçok paket yer almaktadır. Yükleme yapılırken isimlere dikkat edilmelidir.

DİKKAT: İki paket yüklemesinden kaynaklı uyumsuzluk sorunlarına karşı daha alt sürümler tercih edilebilir. Ekranlar gelmiyorsa ve en başa dönüş oluyorsa sürümler ve MySQL .NET Connector sürümü kontrol edilir. Gerekiyorsa alt sürümler kurulur. Alt sürümler, paketler yüklenmeden önce yanındaki sürüm kutusundan indirilip çıkarılabilir.

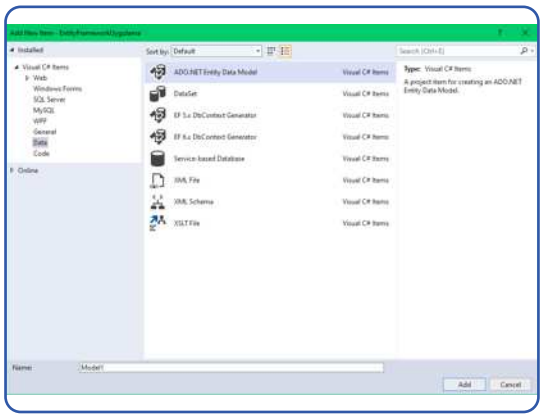
Projeye sağ tıklanarak proje son bir kez "**Rebuild**" edilir.

Forma **Entity Framework** eklenmelidir. Solution Explorer penceresinde projeye sağ tıklanarak "Add → New Item" yolu izlenir (Görsel 6.137).



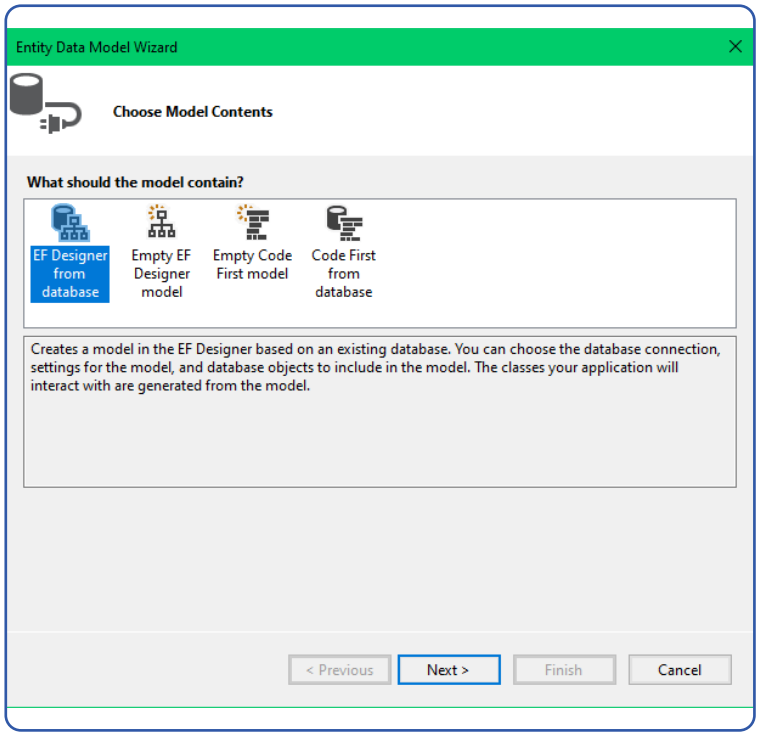
Görsel 6.137: Entity Framework için yeni bileşen ekleme

Gelen pencerede sol taraftan Data ve sağ taraftan "**ADO.NET Entity Data Model**" seçilerek Add düğmesine basılır (Görsel 6.138).



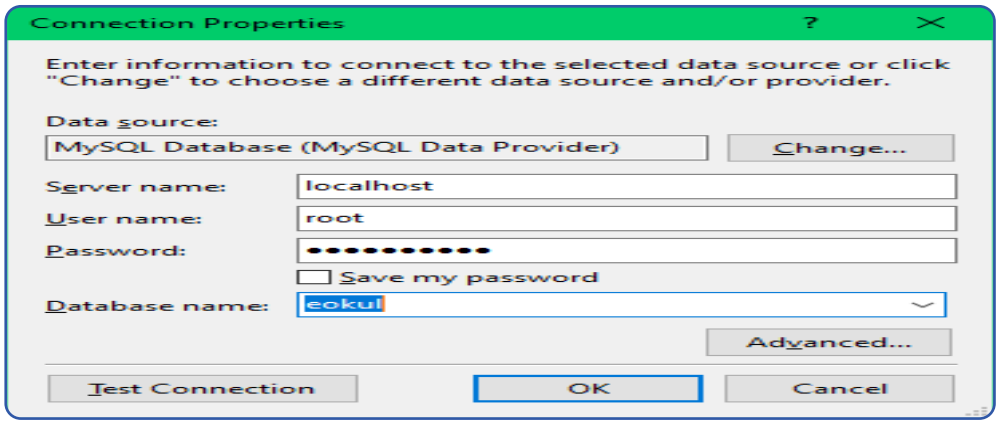
Görsel 6.138: Projeye Entity Framework ekleme

Gelen pencerede "**EF Designer from database**" seçeneği işaretlenerek Next kutucuğuna tıklanır (Görsel 6.139).



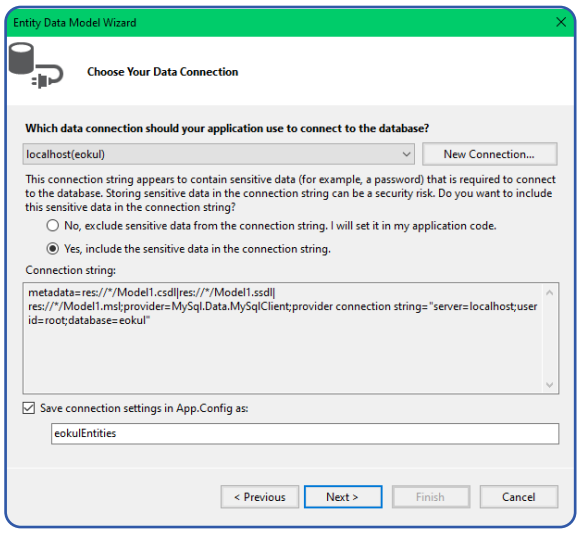
Görsel 6.139: Entity Framework Model seçim penceresi

Gelen ekranda bağlantının tanımlanması gerekmektedir. "**New Connection**" düğmesine basılır. Veri tabanının ayarları girilir, veri tabanı bağlantısı "**Test Connection**" ile test edilir, "**database name**" kısmından kullanılacak veri tabanı seçilerek "**OK**" düğmesine basılır (Görsel 6.140).



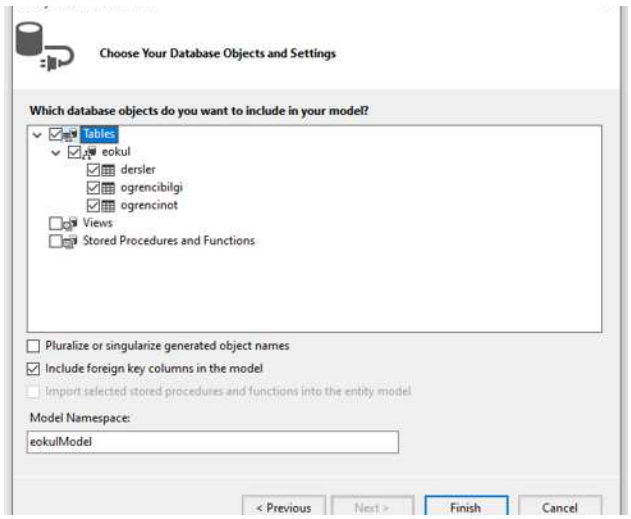
Görsel 6.140: Entity Framework veri tabanı bağlantısı

Gelen ekranda "**Yes, include the sensitive data in the connection string**" seçeneği seçilir. Burada en önemli bölüm, **eokulEntities** kısmıdır. Bu, bağlantı sonucu oluşan bir yapıdır. Projede bu isim ile veri tabanına ulaşılır (Görsel 6.141).



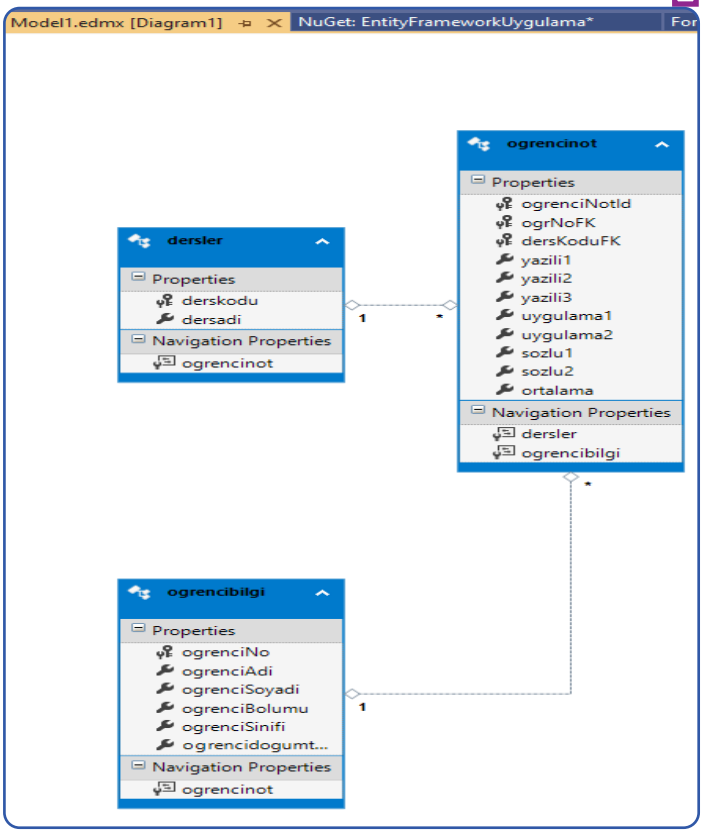
Görsel 6.141: Entity Framework ayarları

Gelen ekranda veri tabanına eklenecek yapılar seçilir. Tablolarla çalışılacağı için tablolar seçilir ve "**Finish**" düğmesine basılır (Görsel 6.142). Bu işlemin tamamlanması biraz zaman alacaktır.



Görsel 6.142: Entity Framework bileşen seçim ekranı

Tabloların ve bağlantıların olduğu diyagram ekranı gelecektir (Görsel 6.143).



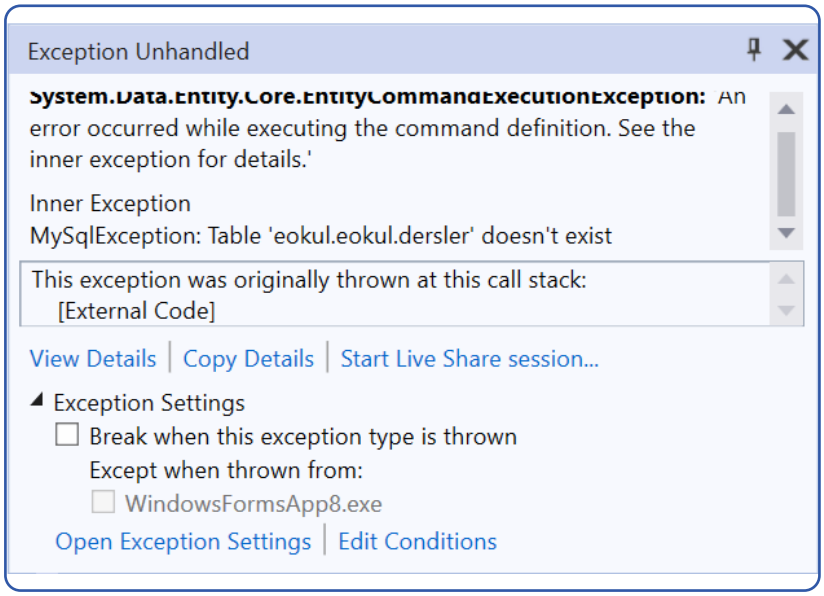
Görsel 6.143: Entity Framework eklendikten sonra gelen diyagram ekranı

Bu diyagram ile işlemler başarılı bir şekilde tamamlanır. Kodlar yazılırken ilk olarak formun **Load()** olayına listele metodu yazılır.

```
eokulEntities ogrenci_ent = new eokulEntities(); //Projeye eklenen Entity türünde bir nesne tanımlanır.
private void Form1_Load(object sender, EventArgs e)
 {
    dataGridView1.DataSource = ogrenci_ent.ogrencibilgi.ToList(); //Oluşturulan nesne ile ogrenci bilgi tablosuna erişim sağlanıp listeleme yapılır.
    dataGridView1.Columns[6].Visible = false;//Datagridview'de görünmesi istenmeyen sütunlar gizlenebilir. Listelemede bir sorun yoksa bu satır gereksizdir.
 }
```

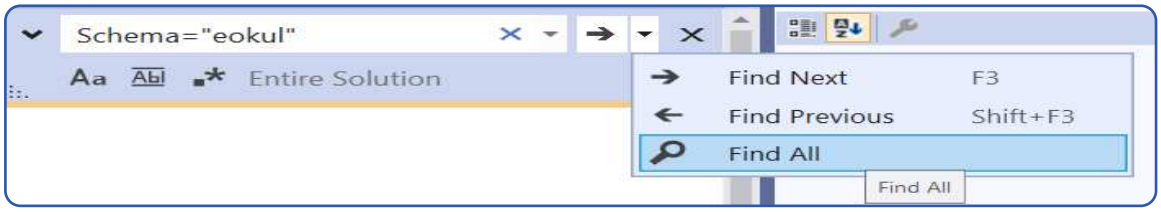
DİKKAT: Bu noktada projeyi çalıştırırken olabilecek bir hataya ve çözümüne ilişkin şu adımlar izlenmelidir:

- Hata mesajında veri tabanı adı istek dışında tekrarlandığı için veri tabanı ile iletişim sağlanamamaktadır. Projedeki veri tabanı adı "eokul" olduğu için burada veri tabanı adının tekrar ettiği görülmektedir (Görsel 6.144 veri tabanı adı tekrarlama hatası).



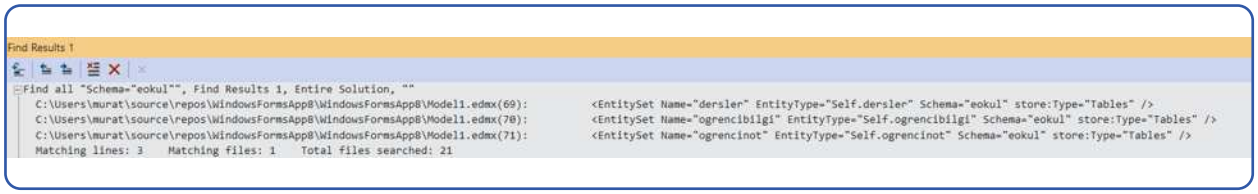
Görsel 6.144: Veri tabanı adı tekrarlama hatası

- Çözüm için ilk olarak projeye geçilir ve **Ctrl + F** yapılır. Arama penceresine "**Schema="eokul""** yazılır, arama yeri olarak "**Entire Solution**" seçilir ve son olarak arama metodu olarak "**Find All**" seçilir ve ok işaretine basılır (Görsel 6.145).



Görsel 6.145: Arama penceresi ayarları

- Böylece tekrara sebep olan kısımlar tespit edilir ve herhangi birine çift tıklanır (Görsel 6.146).



Görsel 6.146: Arama sonucu bulunan araçlar

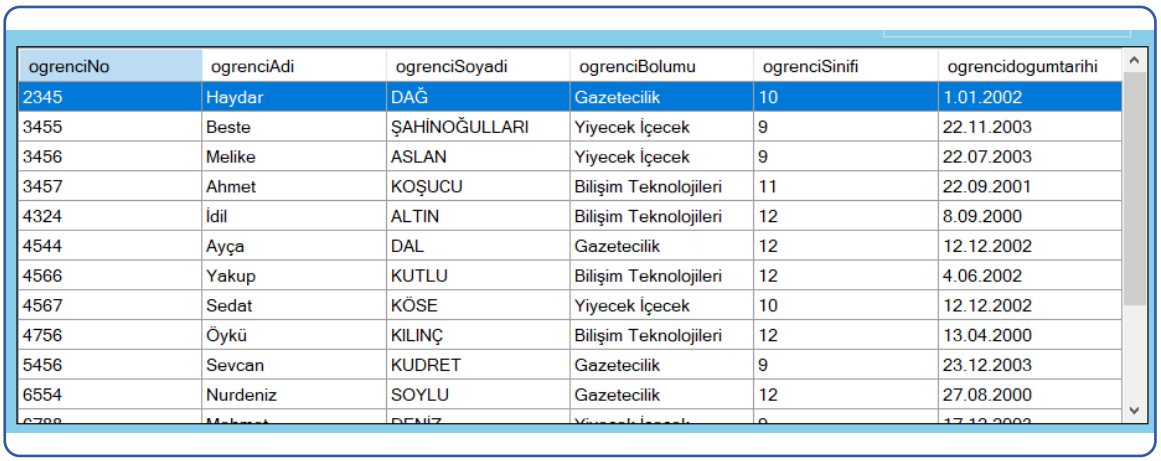
- Böylece "**Model1.edmx**" dosyasının kodlarına erişilir.

```
<EntitySet Name="dersler" EntityType="Self.dersler" Schema="eokul" store:Type="Tables" />
<EntitySet Name="ogrencibilgi" EntityType="Self.ogrencibilgi" Schema="eokul" store:Type="Tables" />
<EntitySet Name="ogrencinot" EntityType="Self.ogrencinot" Schema="eokul" store:Type="Tables" />
```

- Erişim sağladıktan sonra yapılması gereken "**Schema="eokul""** kısmından **eokul** bölümünü silmektir. Son durum şöyle olacaktır:

```
<EntitySet Name="dersler" EntityType="Self.dersler" Schema="" store:Type="Tables" />
<EntitySet Name="ogrencibilgi" EntityType="Self.ogrencibilgi" Schema="" store:Type="Tables" />
<EntitySet Name="ogrencinot" EntityType="Self.ogrencinot" Schema="" store:Type="Tables" />
```

Böylece artık bir sıkıntı olmadan proje çalışacaktır (Görsel 6.147).



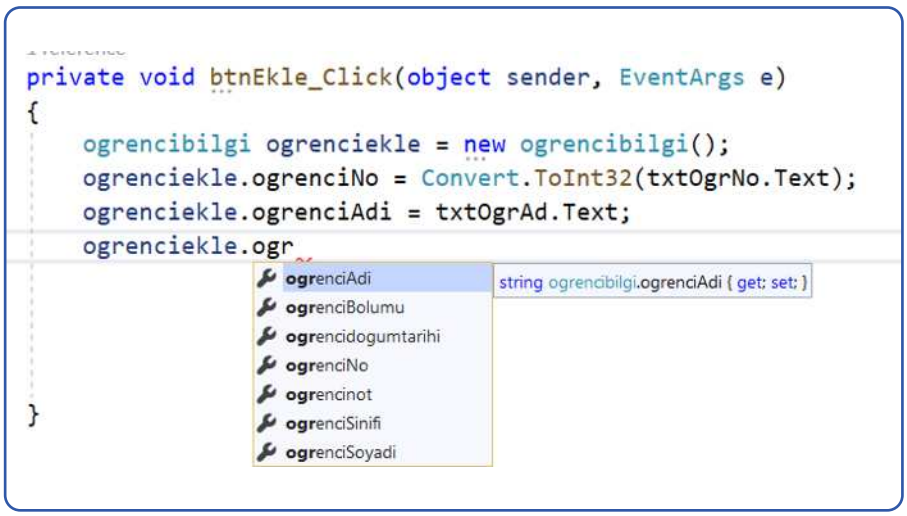
ogrenciNo	ogrenciAdi	ogrenciSoyadi	ogrenciBolumu	ogrenciSinifi	ogrencidogumtarihi
2345	Haydar	DAĞ	Gazetecilik	10	1.01.2002
3455	Beste	ŞAHİNOĞULLARI	Yiyecek İçecek	9	22.11.2003
3456	Melike	ASLAN	Yiyecek İçecek	9	22.07.2003
3457	Ahmet	KOŞUCU	Bilişim Teknolojileri	11	22.09.2001
4324	İdil	ALTIN	Bilişim Teknolojileri	12	8.09.2000
4544	Ayça	DAL	Gazetecilik	12	12.12.2002
4566	Yakup	KUTLU	Bilişim Teknolojileri	12	4.06.2002
4567	Sedat	KÖSE	Yiyecek İçecek	10	12.12.2002
4756	Öykü	KILINÇ	Bilişim Teknolojileri	12	13.04.2000
5456	Sevcan	KUDRET	Gazetecilik	9	23.12.2003
6554	Nurdeniz	SOYLU	Gazetecilik	12	27.08.2000

Görsel 6.147: Entity Framework'ün çalışması

Artık projeye kayıt eklenebilir.

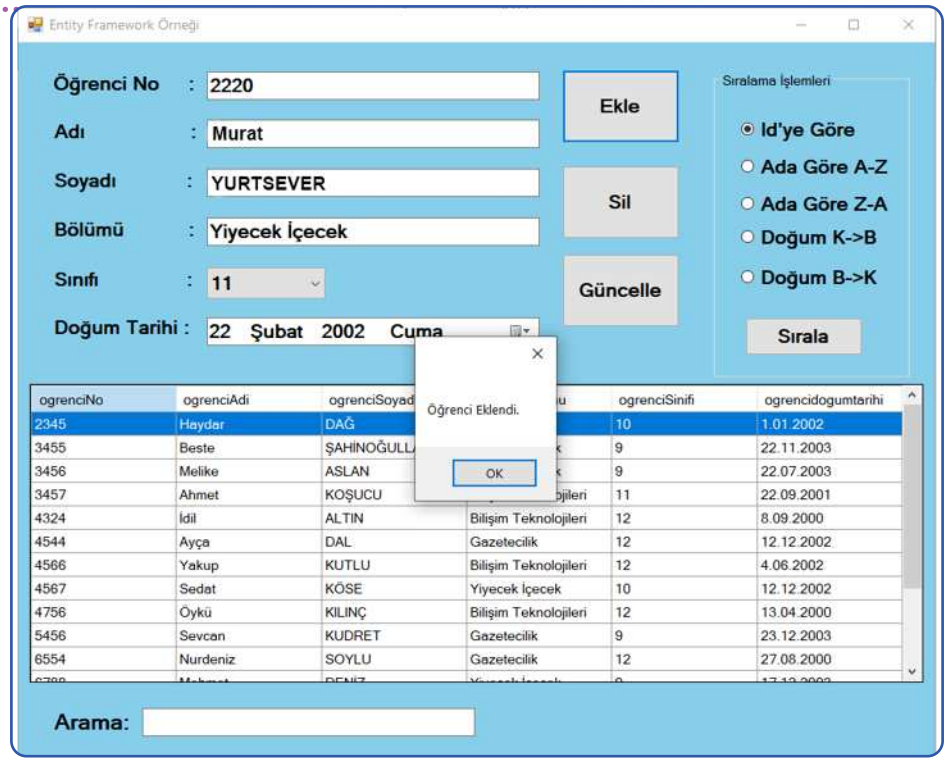
```
private void btnEkle_Click(object sender, EventArgs e)
    {
        try
        {
            //projemize veri tabanını Entity Framework olarak ekledikten sonra
            //tablolar sınıflara dönüşür. ogrencibilgi tablosundan
            //bir nesne türetilir.
            ogrencibilgi ogrenciekle = new ogrencibilgi();
            ogrenciekle.ogrenciNo = Convert.ToInt32(txtOgrNo.Text);
            ogrenciekle.ogrenciAdi = txtOgrAd.Text;
            ogrenciekle.ogrenciSoyadi = txtOgrSoyad.Text;
            ogrenciekle.ogrenciBolumu = txtOgrBolum.Text;
            ogrenciekle.ogrenciSinifi = Convert.ToSByte(cmbSinif.Text);
            ogrenciekle.ogrencidogumtarihi = dateTimePicker1.Value.Date;
            ogrenci_ent.ogrencibilgi.Add(ogrenciekle);//Nesneye değerleri atadıktan sonra tek satır ile veri tabanına eklenir.
            ogrenci_ent.SaveChanges();//Değişiklikler kaydedilir.
            MessageBox.Show("Öğrenci Eklendi.");
            dataGridView1.DataSource = ogrenci_ent.ogrencibilgi.ToList();//Datagrid yenilenir.
        }
        catch
        {
            MessageBox.Show("Lütfen Boş alan geçmeyiniz.\nLütfen doğru değerler giriniz.\nNo benzersiz olmalıdır.");
        }

    }
```

Görüldüğü gibi kayıt eklerken ADO.Net'teki gibi veri tabanı bağlantısı açmak, SQL komutu yazmak ya da diğer işlemlere gerek kalmamıştır. Üstelik **ogrencibilgi** tablosundan türetilen ogrenciekle nesnesi ile direkt olarak tablodaki alanlarla etkileşime geçilebilecektir (Görsel 6.148) (Görsel 6.149).



Görsel 6.148: Entity Framework ile Intellisense (Akıllı Kod Tamamlama) Özelliği

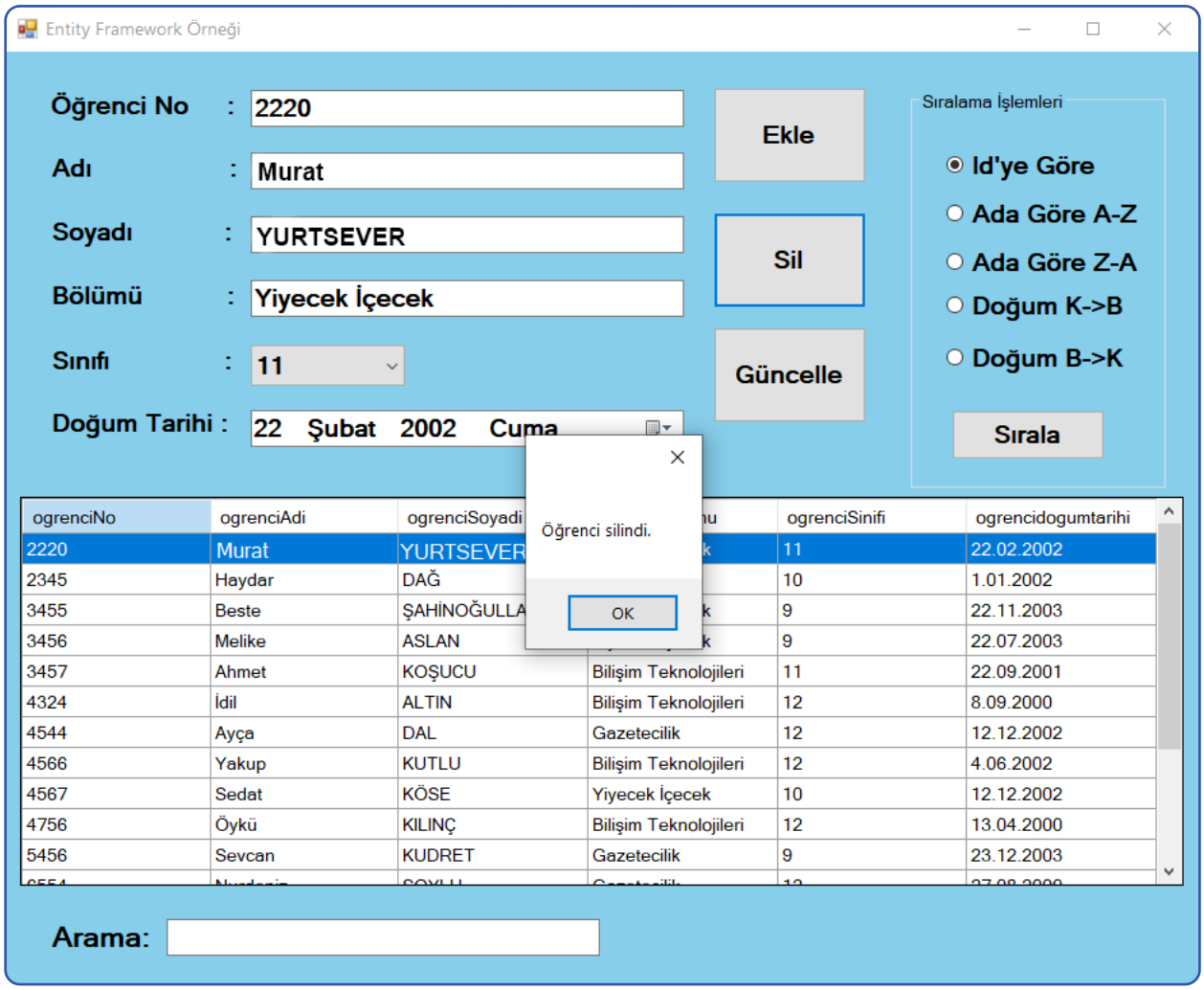
Kayıt silme işlemine geçilir.



Görsel 6.149: Entity Framework ile kayıt ekleme

```
private void btnSil_Click(object sender, EventArgs e)
    {
        try
        {
            if (dataGridView1.SelectedRows.Count != 0)
            {
                int secilen_ogrencino = Convert.ToInt32(dataGridView1.CurrentRow.Cells[0].Value.ToString());
                //var türünde yani türsüz bir lambda kullanılmış değişkenimiz
                var ogrencisil = ogrenci_ent.ogrencibilgi.Where(w => w.ogrenciNo == secilen_ogrencino).Firs-
tOrDefault();//w yapısını burada tanımlayıp veri tabanındaki ogrencibilgi tablosundaki ogrencino alanı ile
ilişkilendirip seçilen satırın öğrenci numarasını kullanarak karşılaştırıyoruz.

                ogrenci_ent.ogrencibilgi.Remove(ogrencisil);//var olarak tanımladığımız değişkenimizi Remove
metodunda kullanıyoruz
                ogrenci_ent.SaveChanges();//değişiklikleri kayıt ediyoruz.
                MessageBox.Show("Öğrenci silindi.");
                dataGridView1.DataSource = ogrenci_ent.ogrencibilgi.ToList();
            }
            else
                MessageBox.Show("Lütfen Silinecek Kaydı Seçiniz...");
        }
        catch
        {
            MessageBox.Show("Başka tabloda ilişkilendirilmiş olan kayıtlar silinemez.");
        }
    }
```

Burada farklı olarak var türünde bir lambda cümlesi görülmektedir. Silme ve güncelleme işlemlerinde mutlaka "**Where**" kullanılarak sınırlandırılma yapılmalıdır. Burada da yapılan işlem budur. Güncelleme işlemine geçilir. Önce **datagridview**'in **CellDoubleClick** olayına çift tıklanıldığı zaman bilgilerin textboxlara girilmesi için kodlar yazılır (Görsel 6.150).

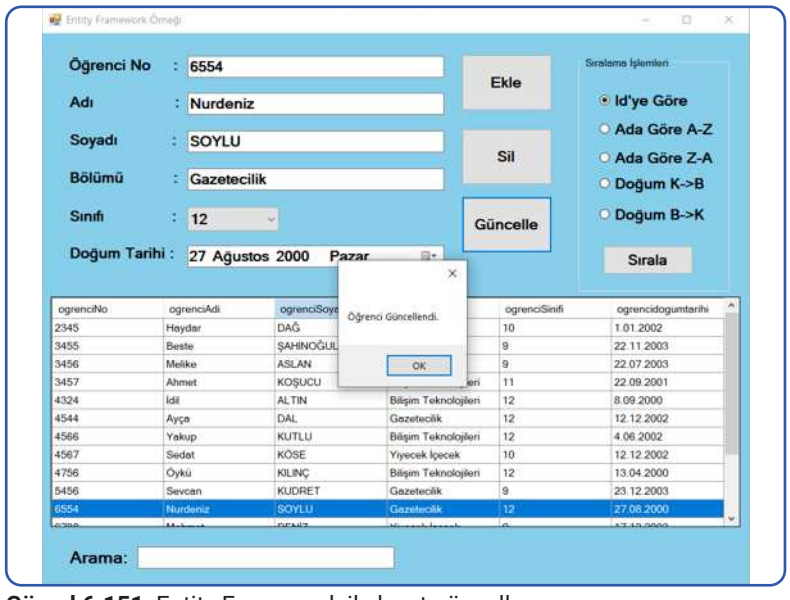


Görsel 6.150: Entity Framework ile öğrenci silme

```
private void dataGridView1_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
    {
        txtOgrNo.Text = dataGridView1.CurrentRow.Cells[0].Value.ToString();
        txtOgrAd.Text = dataGridView1.CurrentRow.Cells[1].Value.ToString();
        txtOgrSoyad.Text = dataGridView1.CurrentRow.Cells[2].Value.ToString();
        txtOgrBolum.Text = dataGridView1.CurrentRow.Cells[3].Value.ToString();
        cmbSinif.Text = dataGridView1.CurrentRow.Cells[4].Value.ToString();
        dateTimePicker1.Value.Date=dataGridView1.CurrentRow.Cells[5].Value.ToString();
```

Şimdi güncelleme kodları yazılır (Görsel 6.151).

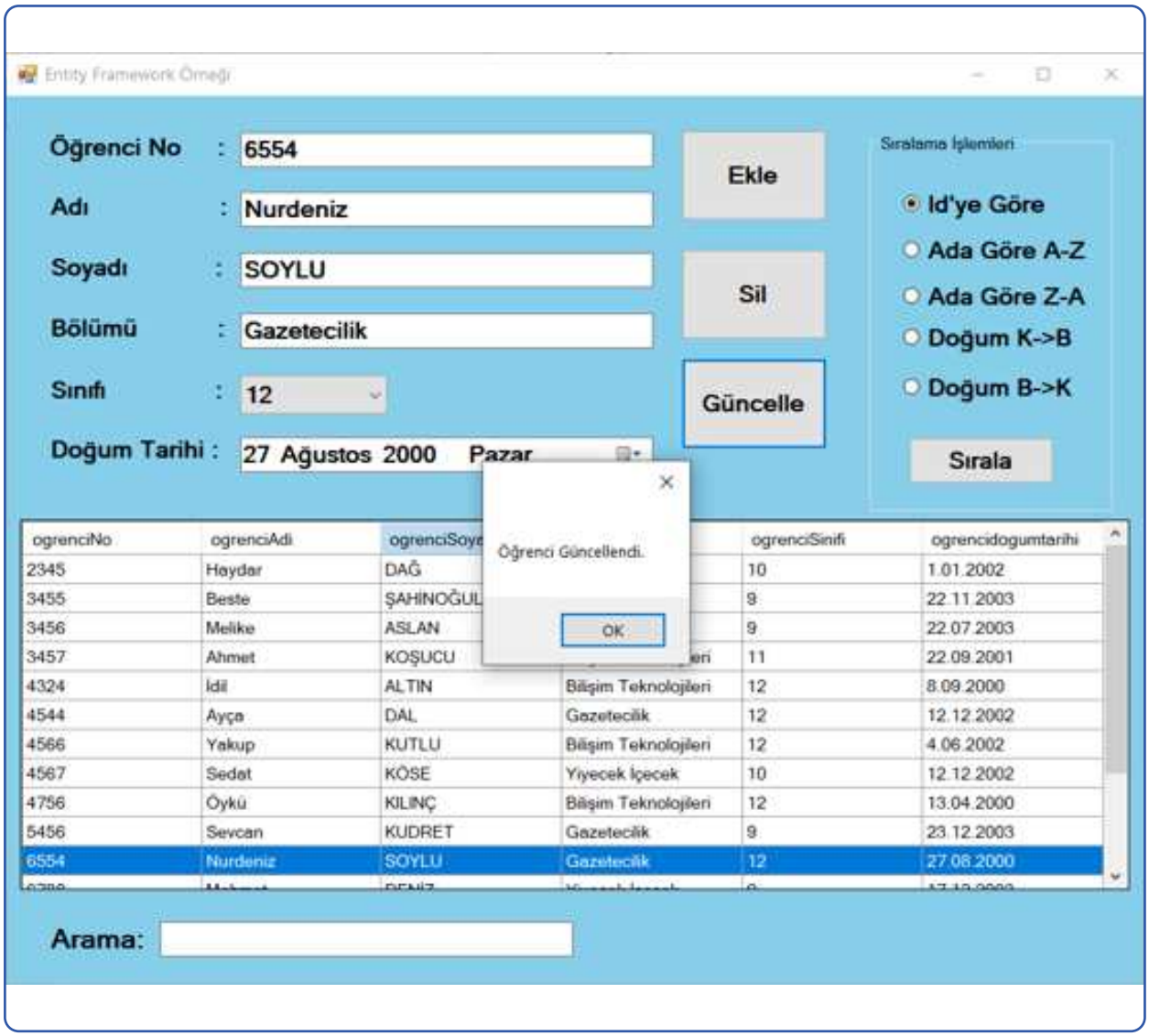
```
private void btnGuncelle_Click(object sender, EventArgs e)
    {
        try
        {
            if (dataGridView1.SelectedRows.Count != 0)
            {
                int secilen_ogrencino = Convert.ToInt32(dataGridView1.CurrentRow.Cells[0].Value.ToString());

                //Lambda yapısı ile sorgu cümlesi yazılır.
                var ogrenciguncelle = ogrenci_ent.ogrencibilgi.Where(w => w.ogrenciNo == secilen_ogrenci-
no).FirstOrDefault();
                ogrenciguncelle.ogrenciNo = Convert.ToInt32(txtOgrNo.Text);
                ogrenciguncelle.ogrenciAdi = txtOgrAd.Text;
                ogrenciguncelle.ogrenciSoyadi = txtOgrSoyad.Text;
                ogrenciguncelle.ogrenciBolumu = txtOgrBolum.Text;
                ogrenciguncelle.ogrenciSinifi = Convert.ToSByte(cmbSinif.Text);
                ogrenciguncelle.ogrencidogumtarihi = dateTimePicker1.Value.Date;
                ogrenci_ent.SaveChanges();//Değişiklikler kaydedilir.
                MessageBox.Show("Öğrenci Güncellendi.");
                dataGridView1.DataSource = ogrenci_ent.ogrencibilgi.ToList();
            }
            else
                MessageBox.Show("Lütfen güncellenecek kayda çift tıklayınız.");
        }
        catch
        {
            MessageBox.Show("Lütfen güncellenecek kayda çift tıklayınız.\nLütfen alanları kontrol ediniz.");
        }
    }
```



Görsel 6.151: Entity Framework ile kayıt güncelleme

Böylece temel işlemler (ekleme, listeleme, güncelleme, silme - CRUD) yapılmış olur.
Arama işlemine geçilir. Bu işlem arama textbox'ının **textchanged** olayına yazılır (Görsel 6.152).

```
private void txtArama_TextChanged(object sender, EventArgs e)
    {
      dataGridView1.DataSource = ogrenci_ent.ogrencibilgi.Where(w => w.ogrenciAdi == txtArama.Text | w.ogrenciSoyadi == txtArama.Text).ToList();
    }
```

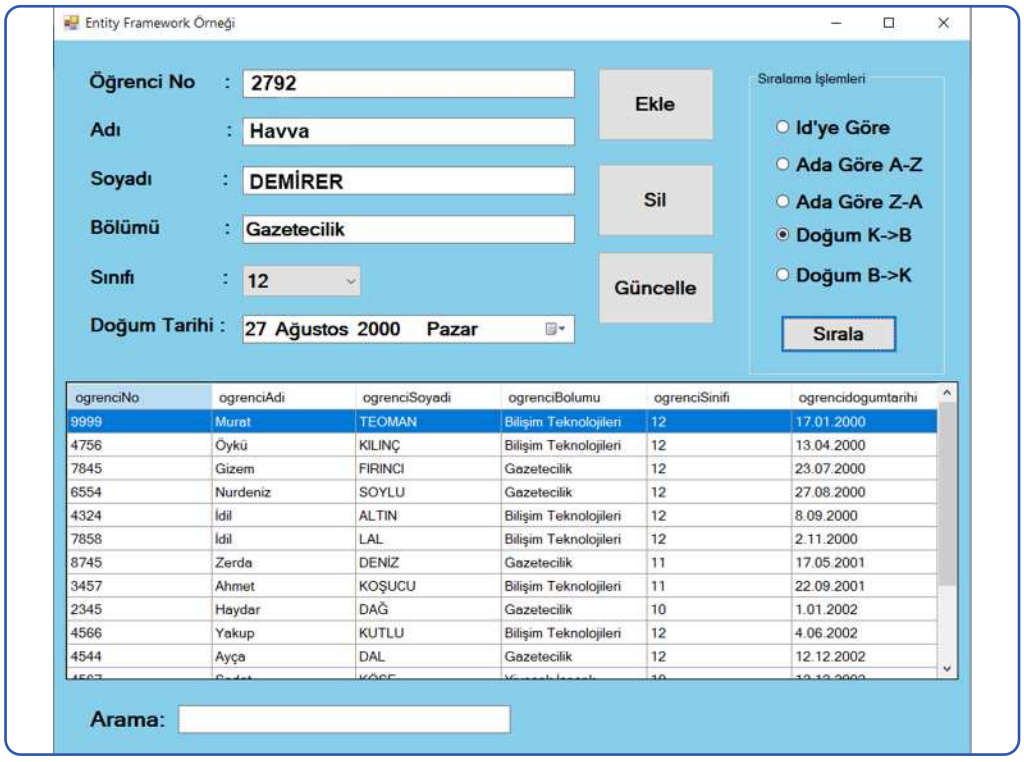


Görsel 6.152: Lambda yapısı ile arama işlemi

Son olarak **LINQ** yapısı kullanılarak sıralama işlemleri yapılır.

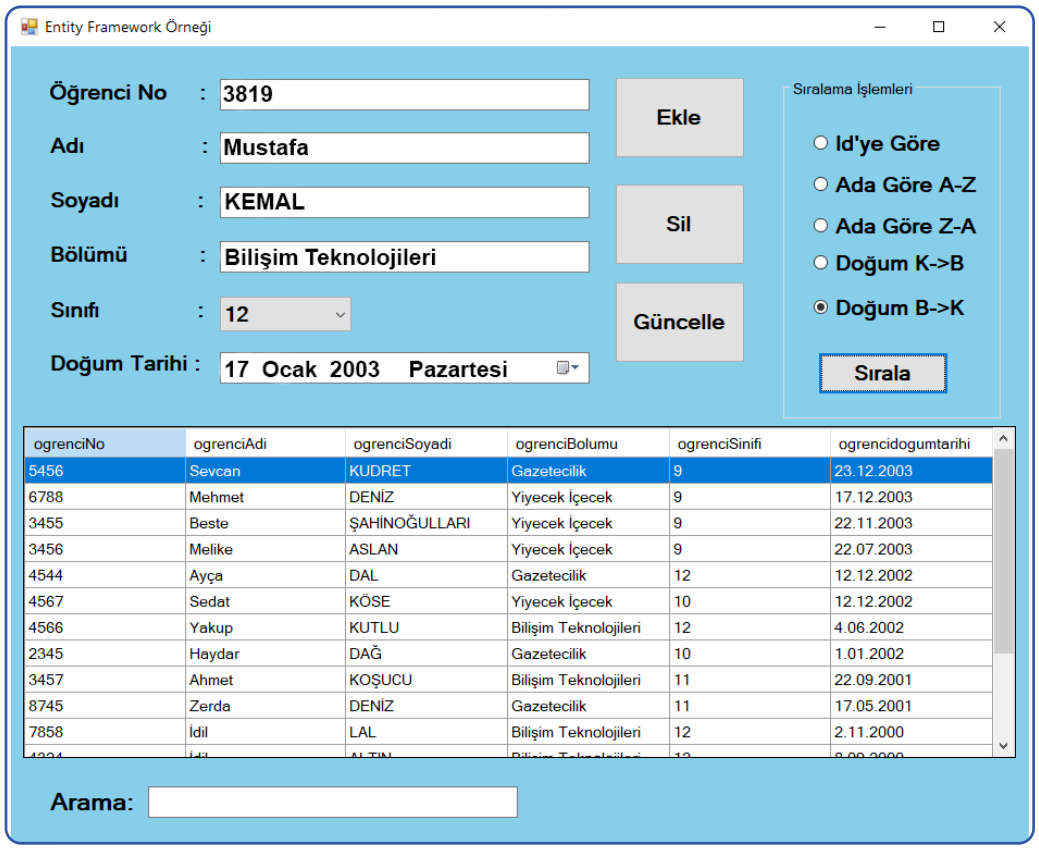
```
private void btnSirala_Click(object sender, EventArgs e)
    {
      if (rdbIdSirala.Checked==true)
      {
        //Numaraya göre sıralama
        List<ogrencibilgi> nosiraliliste = ogrenci_ent.ogrencibilgi.OrderBy(p => p.ogrenciNo).ToList();
        dataGridView1.DataSource = nosiraliliste;
      }
      else if (rdbAZ.Checked==true)
      {
        //Ada göre küçükten büyüğe doğru sıralama
        List<ogrencibilgi> azsiraliliste = ogrenci_ent.ogrencibilgi.OrderBy(p => p.ogrenciAdi).ToList();
        dataGridView1.DataSource = azsiraliliste;
      }
      else if (rdbZA.Checked == true)
      {
        //Ada göre büyükten küçüğe sıralama
        List<ogrencibilgi> zasiraliliste = ogrenci_ent.ogrencibilgi.OrderByDescending(p => p.ogrenciAdi).
ToList();
        dataGridView1.DataSource = zasiraliliste;
      }
      else if (rdbDogum.Checked == true)
      {
        //Doğum tarihine göre küçükten büyüğe doğru sıralama
        List<ogrencibilgi> dogumkb = ogrenci_ent.ogrencibilgi.OrderBy(p => p.ogrencidogumtarihi).
ToList();
        dataGridView1.DataSource = dogumkb;
      }
      else
      {
        //Doğum tarihine göre büyükten küçüğe doğru sıralama
        List<ogrencibilgi> dogumbk = ogrenci_ent.ogrencibilgi.OrderByDescending(p => p.ogrencido-
gumtarihi).ToList();
        dataGridView1.DataSource = dogumbk;
      }
    }
```

Burada seçilen radiobutton'a göre sıralama işlemleri tek satır kod ile yapılmaktadır. ADO.NET'te bu işlemler daha fazla kod ile sağlanmaktadır fakat unutulmaması gereken husus ADO.NET'te kodlarla çalışmadan diğer yapılara düzgün geçiş yapılamayacağıdır (Görsel 6.153) (Görsel 6.154).



ogrenciNo	ogrenciAdi	ogrenciSoyadi	ogrenciBolumu	ogrenciSinifi	ogrencidogumtarihi
9999	Murat	TEOMAN	Bilişim Teknolojileri	12	17.01.2000
4756	Öykü	KILINÇ	Bilişim Teknolojileri	12	13.04.2000
7845	Gizem	FIRINCI	Gazetecilik	12	23.07.2000
6554	Nurdeniz	SOYLU	Gazetecilik	12	27.08.2000
4324	İdil	ALTIN	Bilişim Teknolojileri	12	8.09.2000
7858	İdil	LAL	Bilişim Teknolojileri	12	2.11.2000
8745	Zerda	DENİZ	Gazetecilik	11	17.05.2001
3457	Ahmet	KOŞUCU	Bilişim Teknolojileri	11	22.09.2001
2345	Haydar	DAĞ	Gazetecilik	10	1.01.2002
4566	Yakup	KUTLU	Bilişim Teknolojileri	12	4.06.2002
4544	Ayça	DAL	Gazetecilik	12	12.12.2002

Görsel 6.153: LINQ ile doğum tarihine göre küçükten büyüğe sıralama işlemi



ogrenciNo	ogrenciAdi	ogrenciSoyadi	ogrenciBolumu	ogrenciSinifi	ogrencidogumtarihi
5456	Sevcan	KUDRET	Gazetecilik	9	23.12.2003
6788	Mehmet	DENİZ	Yiyecek İçecek	9	17.12.2003
3455	Beste	ŞAHİNOĞULLARI	Yiyecek İçecek	9	22.11.2003
3456	Melike	ASLAN	Yiyecek İçecek	9	22.07.2003
4544	Ayça	DAL	Gazetecilik	12	12.12.2002
4567	Sedat	KÖSE	Yiyecek İçecek	10	12.12.2002
4566	Yakup	KUTLU	Bilişim Teknolojileri	12	4.06.2002
2345	Haydar	DAĞ	Gazetecilik	10	1.01.2002
3457	Ahmet	KOŞUCU	Bilişim Teknolojileri	11	22.09.2001
8745	Zerda	DENİZ	Gazetecilik	11	17.05.2001
7858	İdil	LAL	Bilişim Teknolojileri	12	2.11.2000

Görsel 6.154: LINQ ile doğum tarihine göre büyükten küçüğe sıralama işlemi

ÖLÇME VE DEĞERLENDİRME - 6

1. Veri tabanları olmasaydı bugün ne tür sorunlar yaşanırdı? Araştırınız.

2. Aşağıdaki SQL komutlarının sonuçlarını yazınız.

A) insert into yazar(yazarad,yazarsoyad) values('İdil','İMSİYAT');

B) select * from kitap where kitapadi like '_p%';

C) select * from ogrenci where sinif='10A' order by ogrno desc;

D) select ograd,ogrsoyad,sinif,cinsiyet from ogrenci where (sinif='9A' and cinsiyet='E') or (sinif='9B' and cinsiyet='K');

E) select * from kitap where sayfasayisi between 50 and 200;

3. Aşağıda belirtilen durumları SQL komutları ile yazınız.

A) Öğrenci tablosunda doğum yılı 1989 olan öğrencileri listeleyiniz.

B) 9A sınıfındaki tüm öğrencileri 10A sınıfına aktarınız.

C) Öğrenci tablosundaki adı 'A' harfi ile başlayan öğrencileri listeleyiniz.

D) Öğrenci tablosunda 10A veya 10B sınıfındaki öğrencilerin adını, soyadını ve sınıfını listeleyiniz.

E) Öğrenci tablosunda 10A sınıfı ve kız öğrencileri listeleyiniz.

4. Bir e-ticaret sitesini inceleyiniz. Ürünler için, müşteriler için, siparişler için tablolar oluşturup bunların bağlantılarını yapınız.

5. Veri tabanına kayıtlar ekleyiniz ve bu kayıtlar üzerinde insert, delete, update, select komutlarını kullanınız.

6. Örnek bir Connection String yazınız.

7. ADO.NET ve Entity Framework yapılarını karşılaştırınız.

ÖĞRENME BİRİMLERİ

ÖLÇME VE DEĞERLENDİRME CEVAP ANATARLARI

ÖĞRENME BİRİMİ 1'İN CEVAP ANAHTARI

1. Y
2. D
3. D
4. Y
5. D
6. D
7. Y
8. using
9. ToolBox veya Araç Kutusu
10. %
11. ToString()
12. Kaydet - namespace
13. Değişken isimlerinde boşluk karakteri kullanılmaz.
    Değişken isimleri sayı ile başlamaz.
    Doğru tanımlanmıştır.
    Doğru tanımlanmıştır.
    Doğru tanımlanmıştır.
    Ondalık sayılar int veri türünde tanımlanamaz.
    char olarak tanımlı değişkenlere tek tırnak içine alınmış sadece bir karakter atanabilir.
    Doğru tanımlanmıştır.
14. 3,2 sayısı mesaj olarak görülür.
15. private void button1_Click(object sender, EventArgs e)
    {
    int sayi;
    double sonuc;
    sayi = Convert.ToInt32(textBox1.Text);
    sonuc = sayi * 0.18;
    MessageBox.Show(sonuc.ToString());
    }
16. private void button1_Click(object sender, EventArgs e)
    {
    int yariCap;
    double alan,cevre,piSayisi;
    piSayisi = 3.14;
    yariCap = Convert.ToInt32(textBox1.Text);
    alan = piSayisi * yariCap * yariCap;
    cevre = 2 * piSayisi * yariCap;
    MessageBox.Show("Dairenin alanı=" + alan.ToString());
    MessageBox.Show("Dairenin çevresi=" + cevre.ToString());
    }

ÖĞRENME BİRİMİ 2 'NİN CEVAP ANAHTARI

1. Y
2. Y
3. D
4. D
5. Y
6. if(a > b && a > c)
7. if(a < b || a < c)
8. 3 – 6 – 9 – 12 – 15 – 17 – 18
9. 1 – 2 – 4 – 5 – 7 – 8 – 10 – 11 – 13 – 14
10. 1. tur sonuç=1
    2. tur sonuç=2
    3. tur sonuç=6
    4. tur sonuç=24
    5. tur sonuç=120
    6. tur sonuç=720
11. Kodlar çalıştırılmaz ve program başlamaz.

    Açıklama: Kodlar derlenirken Hata Listesi panelinde "string türü örtülü olarak int türüne dönüştürülemez."

    mesajı verir ve kodlar çalıştırılmaz çünkü yapılan hata, derleme hatasıdır.

    Bu tip hatalar, programın başlatılmasını engelleyen çok kritik hatalardır.

12.
```
private void button1_Click_1(object sender, EventArgs e)
 {
int sayi1, sayi2, sayi3;
sayi1 = Convert.ToInt32(textBox1.Text);
sayi2 = Convert.ToInt32(textBox2.Text);
sayi3 = Convert.ToInt32(textBox3.Text);
if(sayi1>sayi2 && sayi1 > sayi3)
{
MessageBox.Show("İlk girilen sayı en büyüktür.");
}
 if(sayi2>sayi1 && sayi2 > sayi3)
{
 MessageBox.Show("İkinci girilen sayı en büyüktür.");
}
 if (sayi3 > sayi1 && sayi3 > sayi2)
{
 MessageBox.Show("Son girilen sayı en büyüktür.");
}
 }
```

ÖĞRENME BİRİMİ 3'ÜN CEVAP ANAHTARI

1.

```
class Televizyon
{
    private int sesSeviyesi;
    private double ekranBoyutu;
    private string goruntuTeknolojisi;
    public int SesSeviyesi
    {
        get
        {
            return sesSeviyesi;
        }
        set
        {
            sesSeviyesi = value;
        }
    }

    public double EkranBoyutu
    {
        get
        {
            return ekranBoyutu;
        }
        set
        {
            ekranBoyutu = value;
        }
    }

    public string GoruntuTeknolojisi
    {
        get
        {
            return goruntuTeknolojisi;
        }
        set
        {
            goruntuTeknolojisi = value;
        }
    }
}
```

2.

```
private class Bilgisayar
{
    double ram;
    public double RAMKapasitesi
    {
        get { return ram; }
        set { ram = value; }
    }
    string cpu;
    public string CPU
    {
        get { return cpu; }
        set { cpu = value; }
    }
    double hd;
    public double HDKapasitesi
    {
        get { return hd; }
        set { hd = value; }
    }
}
```

3.

```
class Televizyon
{
    public int SesSeviyesi { get; set; }
    public double EkranBoyutu { get; set; }
    public string GoruntuTeknolojisi { get; set; }
}
```

4.

```
class Bilgisayar
{
    public double RAMKapasitesi { get; set; }
    public string CPU { get; set; }
    public double HDKapasitesi { get; set; }
}
```

5.

```
class Televizyon
{
    public int SesSeviyesi { get; set; }
    public double EkranBoyutu { get; set; }
    public string GoruntuTeknolojisi { get; set; }

    bool gucAcik = false;
    int kanalNo = 1;
    public void GucAc()
    {
        gucAcik = true;
    }
    public void GucKapat()
    {
        gucAcik = false;
    }

    public void KanalDegistir(int kanalNo)
    {
        this.kanalNo = kanalNo;
    }

    public int SesSeviyesiOku()
    {
        return SesSeviyesi;
    }
}
```

6.

```
class Televizyon
{
    public int SesSeviyesi { get; set; }
    public double EkranBoyutu { get; set; }
    public string GoruntuTeknolojisi { get; set; }

    bool gucAcik = false;
    int kanalNo = 1;
    public int KanalNo
    {
        get { return kanalNo; }
    }
    public void GucAc()
    {
        gucAcik = true;
    }
    public void GucKapat()
    {
        gucAcik = false;
    }
    public void KanalDegistir(int kanalNo)
    {
        this.kanalNo = kanalNo;
    }
    public void KanalNoArtir()
    {
        kanalNo++;
    }
    public void KanalNoArtir(int artis)
    {
        kanalNo += artis;
    }
    public void KanalNoAzalt()
    {
        kanalNo--;
    }
    public void KanalNoAzalt(int azalis)
    {
        kanalNo -= azalis;
    }

    public int SesSeviyesiOku()
    {
        return SesSeviyesi;
    }
}
private static void Main(string[] args)
{
    Televizyon tv = new Televizyon();
    tv.KanalDegistir(20);
    System.Console.WriteLine(tv.KanalNo);
    tv.KanalNoArtir();
    System.Console.WriteLine(tv.KanalNo);

    tv.KanalNoArtir(5);
    System.Console.WriteLine(tv.KanalNo);
    tv.KanalNoAzalt();
    System.Console.WriteLine(tv.KanalNo);
    tv.KanalNoAzalt(3);
    System.Console.WriteLine(tv.KanalNo);
}
```

7.

public: Sınıf ögelerine dışarıdan erişim sağlamak için kullanılır.

private: Bu tür ögelere sadece sınıf içinden erişilebilir.

8.

```
abstract class Televizyon
{
    protected int SesSeviyesi { get; set; }
    public double EkranBoyutu { get; set; }
    public string GoruntuTeknolojisi { get; set; }
    bool gucAcik = false;
    int kanalNo = 1;
    public int KanalNo
    {
        get { return kanalNo; }
    }
    public abstract void GucAc();
    public abstract void GucKapat();

    public void KanalDegistir(int kanalNo)
    {
        this.kanalNo = kanalNo;
    }
    public void KanalNoArtir()
    {
        kanalNo++;
    }
    public void KanalNoArtir(int artis)
    {
        kanalNo += artis;
    }
    public void KanalNoAzalt()
    {
        kanalNo--;
    }
    public void KanalNoAzalt(int azalis)
    {
        kanalNo -= azalis;
    }
    public int SesSeviyesiOku()
    {
        return SesSeviyesi;
    }
}
class AkilliTelevizyon : Televizyon
{
    public string IsletimSistemi { get; set; }
    public override void GucAc()
    {
    }
    public override void GucKapat()
    {
    }
}
```

9.

Televizyon sınıfına statik bir alan eklenir.

```
public static string Marka = "...";
```

10.

Televizyon sınıfı sealed olarak tanımlanır.

```
sealed class Televizyon
```

11.

Sınıf içindeki bir değişkeni, dış dünyaya kapatıp sadece sınıf içinde kullanılabilir kılmak için özellik private şeklinde tanımlanmalıdır.

12. Sınıf içindeki bir değişkeni, dış dünyaya kapatıp sadece sınıf içinde ve bu sınıftan türetilen alt sınıflarda kullanılabilir kılmak için özellik protected şeklinde tanımlanmalıdır.

13.

Sınıf içindeki bir değişkeni her yerden erişilebilir kılmak için özellik public şeklinde tanımlanmalıdır.

14. 8

15.

Nesne oluşturulurken otomatik çalışan metot:
Yapıcı metot (Constuctor)

- Sınıf adı ile aynı ada sahip olmalıdır.
- Soyut tanımlanmamalıdır.
- Sadece bir tane statik yapıcı metot olabilir.
- Dönüş tipi olamaz (void dâhil).
- Aşırı yüklenebilir.

Nesne yok edilirken otomatik çalışan metot:
Yıkıcı metot (Destructor)

- Sadece bir tane olabilir (Aşırı yüklenemez).
- Sınıf adı ile aynı ada sahip olmalıdır.
- Adı ~ (tilde) karakteri ile başlamalıdır.
- Dönüş tipi olamaz (void dâhil).

16. 0 & 125

17. Sınıftan nesne oluşturmadan doğrudan sınıf adı ile sınıf ögelerine erişim için kullanılır.

18. Sınıfa ilk erişim sağlandığında çalıştırılır.

19.

ARAYÜZLER	SOYUT SINIFLAR
Bir sınıf birden fazla arayüzden türetilebilir.	Bir sınıf sadece tek bir soyut sınıftan türetilebilir.
Sadece boş (gövdesi olmayan) metotlar tanımlanabilir.	Hem normal metot hem de boş metotlar tanımlanabilir.
Çoklu kalıtım özelliği sağlar.	Çoklu kalıtım özelliği sağlamaz.
Tüm ögeler public olarak kabul edilir.	Ögeler public olmak zorunda değildir.
Yapıcı metot içeremez.	Yapıcı metot içerebilir.
Statik ögeler barındıramaz.	Statik ögeler barındırabilir.

20.

```
interface IGuc
{
   void GucAc();
   void GucKapat();
}
class Televizyon : IGuc
{
   // ...
   public void GucAc()
   {
   }

   public void GucKapat()
   {
   }
}
class Bilgisayar : IGuc
{
   // ...
   public void GucAc()
   {
   }

   public void GucKapat()
   {
   }
}
```

ÖĞRENME BİRİMİ 4'ÜN CEVAP ANAHTARI

1. int[], numaralar
2. string[],sehirler, string[81]
3. 1
4. B
5. A
6. D
7. C
8. E
9. A
10. B
11. A
12. E
13. D
14. B
15. A
16. B
17. E
18. D
19. C

ÖĞRENME BİRİMİ 5'İN CEVAP ANAHTARI

1. Arabirim
2. Application.Run
3. System.Windows.Forms
4.

CenterToScreen
FormClosed
ControlBox
Load
AcceptButton
Show

5. E
6. A
7. C
8. B
9. D
10. E

ÖĞRENME BİRİMİ 6'NIN CEVAP ANAHTARI

2.

A. Yazar isimli veri tabanına yazar ekler.
B. kitap tablosunda kitap adının 2. harfi p olan kayıtları getirir.
C. ogrenci tablosundan sınıfı 10-A olanları okul numarasına göre azalan olarak listeler.
D. öğrenci tablosundan sınıfı 9A ve cinsiyeti erkek olanlar veya sınıfı 9B ve cinsiyeti kız olanları listeler.
E. kitap tablosundan sayfa sayısı 50 ile 100 arasında olanları listeler.

3.

A. select * from ogrenci where dtarih between '01/01/1989' and '12/31/1989';
B. update ogrenci set sinif='10A' where sinif='9A';
C. select * from ogrenci where ograd like 'A%';
D. select ograd, ogrsoyad, sinif from ogrenci where sinif='10A' or sinif='10B';
E. select * from ogrenci where cinsiyet='K' and sinif='10A';

GÖRSEL KAYNAKÇA

KİTAP KAPAĞI GÖRSELLERİ		
Shutterstock	https://www.shutterstock.com/image-vector/digital-code-background-abstract-vector-illustration-270522938	
123rf	https://tr.123rf.com/photo_52422597_abstract-vector-background-blue-and-violet-waved-lines-for-brochure-website-flyer-design-illustratio.html	
Kitap İçinde Kullanılan Görseller		
Sıra Sizde Görseli	https://www.shutterstock.com/tr/image-vector/pencil-write-isolated-icon-1200010768	
Uygulama Görseli	https://tr.123rf.com/photo_16516142_netbook.-doodle-tarz%C4%B1.html	
Not Görseli	https://tr.123rf.com/stok-foto%C4%9Fraf/9043967.html?sti=mlyogs57l6y54x60en|&mediapopup=9043967	
Dikkat Görseli	https://tr.123rf.com/profile_johny007pan?mediapopup=13341637	
Sayfa Numarası Görseli	https://tr.123rf.com/klipart-vekt%C3%B6r/computer.html?alttext=1&oriSearch=sign+icon&sti=lvie4ou39us-kofb73d%7C&mediapopup=30644988	
ÖĞRENME BİRİMİ-1		
Öğrenme Birimi Kapak Resmi	Shutterstock	https://www.shutterstock.com/tr/image-photo/businessman-working-office-pressing-button-on-598371008
Görsel 1.1	Shutterstock	https://www.shutterstock.com/tr/image-vector/desktop-computer-laptop-tablet-smart-phone-221566516
Görsel 1.2	Shutterstock	https://www.shutterstock.com/tr/image-photo/c-sharp-programming-language-web-development-604209716
Görsel 1.3		Yazar tarafından düzenlenmiştir.
Görsel 1.4		Yazar tarafından düzenlenmiştir.
Görsel 1.5		Yazar tarafından düzenlenmiştir.
Görsel 1.6		Yazar tarafından düzenlenmiştir.
Görsel 1.7		Yazar tarafından düzenlenmiştir.
Görsel 1.8		Yazar tarafından düzenlenmiştir.
Görsel 1.9		Yazar tarafından düzenlenmiştir.
Görsel 1.10		Yazar tarafından düzenlenmiştir.
Görsel 1.11		Yazar tarafından düzenlenmiştir.
Görsel 1.12		Yazar tarafından düzenlenmiştir.
Görsel 1.13		Yazar tarafından düzenlenmiştir.
Görsel 1.14		Yazar tarafından düzenlenmiştir.
Görsel 1.15		Yazar tarafından düzenlenmiştir.
Görsel 1.16		Yazar tarafından düzenlenmiştir.
Görsel 1.17		Yazar tarafından düzenlenmiştir.
Görsel 1.18		Yazar tarafından düzenlenmiştir.
Görsel 1.19		Yazar tarafından düzenlenmiştir.
Görsel 1.20		Yazar tarafından düzenlenmiştir.
Görsel 1.21		Yazar tarafından düzenlenmiştir.
Görsel 1.22		Yazar tarafından düzenlenmiştir.
Görsel 1.23		Yazar tarafından düzenlenmiştir.
Görsel 1.24		Yazar tarafından düzenlenmiştir.
Görsel 1.25		Yazar tarafından düzenlenmiştir.
Görsel 1.26	Shutterstock	https://www.shutterstock.com/tr/image-photo/c-code-hand-pencil-points-sharp-1099318907
Görsel 1.27	Shutterstock	https://www.shutterstock.com/tr/image-photo/c-code-hand-pencil-points-sharp-1099318907
Görsel 1.28		Yazar tarafından düzenlenmiştir.
Görsel 1.30		Yazar tarafından düzenlenmiştir.
Görsel 1.31		Yazar tarafından düzenlenmiştir.
Görsel 1.32		Yazar tarafından düzenlenmiştir.

ÖĞRENME BİRİMİ-2		
Öğrenme Birimi Kapak Resmi	Shutters-tock	https://www.shutterstock.com/tr/image-photo/black-shoes-standing-crossroad-making-decisi-on-1179925327
Görsel 2.1	Shutters-tock	https://www.shutterstock.com/tr/image-vector/true-false-green-red-signal-on-307758260
Görsel 2.2	Yazar tarafından düzenlenmiştir.	
Görsel 2.3	Yazar tarafından düzenlenmiştir.	
Görsel 2.4	Yazar tarafından düzenlenmiştir.	
Görsel 2.5	Yazar tarafından düzenlenmiştir.	
Görsel 2.6	Yazar tarafından düzenlenmiştir.	
Görsel 2.7	Shutters-tock	https://www.shutterstock.com/tr/image-vector/man-key-near-computer-account-login-1499141258
Görsel 2.8	Shutters-tock	
Görsel 2.9	Yazar tarafından düzenlenmiştir.	
Görsel 2.10	Yazar tarafından düzenlenmiştir.	
Görsel 2.11	Yazar tarafından düzenlenmiştir.	
Görsel 2.12	Yazar tarafından düzenlenmiştir.	
Görsel 2.13	Yazar tarafından düzenlenmiştir.	
Görsel 2.14	Yazar tarafından düzenlenmiştir.	
Görsel 2.15	Shutters-tock	https://www.shutterstock.com/tr/image-photo/close-counter-cassette-tape-player-repairing-1781199686
Görsel 2.16	Yazar tarafından düzenlenmiştir.	
Görsel 2.17	Yazar tarafından düzenlenmiştir.	
Görsel 2.18	Yazar tarafından düzenlenmiştir.	
Görsel 2.19	Yazar tarafından düzenlenmiştir.	
Görsel 2.20	Yazar tarafından düzenlenmiştir.	
Görsel 2.21	Yazar tarafından düzenlenmiştir.	
Görsel 2.22	Yazar tarafından düzenlenmiştir.	
Görsel 2.23	Yazar tarafından düzenlenmiştir.	
Görsel 2.24	Yazar tarafından düzenlenmiştir.	
Görsel 2.25	Yazar tarafından düzenlenmiştir.	
Görsel 2.26	Yazar tarafından düzenlenmiştir.	

ÖĞRENME BİRİMİ-3		
Öğrenme Birimi Kapak Resmi	Shutters-tock	https://www.shutterstock.com/tr/image-vector/object-oriented-programming-oop-273118595
Görsel 3.1-1	123rf	https://www.123rf.com/stock-photo/55883157.html?sti=nx784oj8xwnh9vplea|
Görsel 3.1-2	123rf	https://www.123rf.com/stock-photo/53600574.html?oriSearch=55883157&sti=od-dw7x1mf2q88i9kfu|
Görsel 3.2	Yazar tarafından düzenlenmiştir.	
Görsel 3.3	Yazar tarafından düzenlenmiştir.	
Görsel 3.4	Yazar tarafından düzenlenmiştir.	
Görsel 3.5	Yazar tarafından düzenlenmiştir.	
Görsel 3.6	Yazar tarafından düzenlenmiştir.	
Görsel 3.7	Yazar tarafından düzenlenmiştir.	
Görsel 3.8	Yazar tarafından düzenlenmiştir.	
Görsel 3.9	Yazar tarafından düzenlenmiştir.	
Görsel 3.10	Yazar tarafından düzenlenmiştir.	

Görsel 3.11	Yazar tarafından düzenlenmiştir.
Görsel 3.12	Yazar tarafından düzenlenmiştir.
Görsel 3.13	Yazar tarafından düzenlenmiştir.
Görsel 3.14	Yazar tarafından düzenlenmiştir.

ÖĞRENME BİRİMİ-4			
Öğrenme Birimi Kapak Resmi	Shutterstock	https://www.shutterstock.com/image-illustration/neon-light-blockchain-technology-information-block-1317288464	
Görsel 4.1	Yazar tarafından düzenlenmiştir.	Görsel 4.15	Yazar tarafından düzenlenmiştir.
Görsel 4.2	Yazar tarafından düzenlenmiştir.	Görsel 4.16	Yazar tarafından düzenlenmiştir.
Görsel 4.3	Yazar tarafından düzenlenmiştir.	Görsel 4.17	Yazar tarafından düzenlenmiştir.
Görsel 4.4	Yazar tarafından düzenlenmiştir.	Görsel 4.18	Yazar tarafından düzenlenmiştir.
Görsel 4.5	Yazar tarafından düzenlenmiştir.	Görsel 4.19	Yazar tarafından düzenlenmiştir.
Görsel 4.6	Yazar tarafından düzenlenmiştir.	Görsel 4.20	Yazar tarafından düzenlenmiştir.
Görsel 4.7	Yazar tarafından düzenlenmiştir.	Görsel 4.21	Yazar tarafından düzenlenmiştir.
Görsel 4.8	Yazar tarafından düzenlenmiştir.	Görsel 4.22	Yazar tarafından düzenlenmiştir.
Görsel 4.9	Yazar tarafından düzenlenmiştir.	Görsel 4.23	Yazar tarafından düzenlenmiştir.
Görsel 4.10	Yazar tarafından düzenlenmiştir.	Görsel 4.24	Yazar tarafından düzenlenmiştir.
Görsel 4.11	Yazar tarafından düzenlenmiştir.	Görsel 4.25	Yazar tarafından düzenlenmiştir.
Görsel 4.12	Yazar tarafından düzenlenmiştir.	Görsel 4.26	Yazar tarafından düzenlenmiştir.
Görsel 4.13	Yazar tarafından düzenlenmiştir.	Görsel 4.27	Yazar tarafından düzenlenmiştir.
Görsel 4.14	Yazar tarafından düzenlenmiştir.		

ÖĞRENME BİRİMİ-5			
Öğrenme Birimi Kapak Resmi	Shutterstock	https://www.shutterstock.com/image-vector/software-web-development-programming-concept-abstract-1122339353	
Görsel 5.1	Yazar tarafından düzenlenmiştir.	Görsel 5.15	Yazar tarafından düzenlenmiştir.
Görsel 5.2	Yazar tarafından düzenlenmiştir.	Görsel 5.16	Yazar tarafından düzenlenmiştir.
Görsel 5.3	Yazar tarafından düzenlenmiştir.	Görsel 5.17	Yazar tarafından düzenlenmiştir.
Görsel 5.4	Yazar tarafından düzenlenmiştir.	Görsel 5.18	Yazar tarafından düzenlenmiştir.
Görsel 5.5	Yazar tarafından düzenlenmiştir.	Görsel 5.19	Yazar tarafından düzenlenmiştir.
Görsel 5.6	Yazar tarafından düzenlenmiştir.	Görsel 5.20	Yazar tarafından düzenlenmiştir.
Görsel 5.7	Yazar tarafından düzenlenmiştir.	Görsel 5.21	Yazar tarafından düzenlenmiştir.
Görsel 5.8	Yazar tarafından düzenlenmiştir.	Görsel 5.22	Yazar tarafından düzenlenmiştir.
Görsel 5.9	Yazar tarafından düzenlenmiştir.	Görsel 5.23	Yazar tarafından düzenlenmiştir.
Görsel 5.10	Yazar tarafından düzenlenmiştir.	Görsel 5.24	Yazar tarafından düzenlenmiştir.
Görsel 5.11	Yazar tarafından düzenlenmiştir.	Görsel 5.25	Yazar tarafından düzenlenmiştir.
Görsel 5.12	Yazar tarafından düzenlenmiştir.	Görsel 5.26	Yazar tarafından düzenlenmiştir.
Görsel 5.13	Yazar tarafından düzenlenmiştir.	Görsel 5.27	Yazar tarafından düzenlenmiştir.
Görsel 5.14	Yazar tarafından düzenlenmiştir.		

ÖĞRENME BİRİMİ-6			
Öğrenme Birimi Kapak Resmi	Shutterstock	https://www.shutterstock.com/tr/image-vector/vector-line-web-banner-big-data-795575680	
Görsel 6.1	Shutterstock	https://www.shutterstock.com/tr/image-photo/database-table-technical-concept-girl-pointing-92126581	
Görsel 6.2	Yazar tarafından düzenlenmiştir.	Görsel 6.43	Yazar tarafından düzenlenmiştir.
Görsel 6.3	Yazar tarafından düzenlenmiştir.	Görsel 6.44	Yazar tarafından düzenlenmiştir.
Görsel 6.4	Yazar tarafından düzenlenmiştir.	Görsel 6.45	Yazar tarafından düzenlenmiştir.
Görsel 6.5	Yazar tarafından düzenlenmiştir.	Görsel 6.46	Yazar tarafından düzenlenmiştir.
Görsel 6.6	Yazar tarafından düzenlenmiştir.	Görsel 6.47	Yazar tarafından düzenlenmiştir.
Görsel 6.7	Yazar tarafından düzenlenmiştir.	Görsel 6.48	Yazar tarafından düzenlenmiştir.
Görsel 6.8	Yazar tarafından düzenlenmiştir.	Görsel 6.49	Yazar tarafından düzenlenmiştir.
Görsel 6.9	Yazar tarafından düzenlenmiştir.	Görsel 6.50	Yazar tarafından düzenlenmiştir.
Görsel 6.10	Yazar tarafından düzenlenmiştir.	Görsel 6.51	Yazar tarafından düzenlenmiştir.
Görsel 6.11	Yazar tarafından düzenlenmiştir.	Görsel 6.52	Yazar tarafından düzenlenmiştir.
Görsel 6.12	Yazar tarafından düzenlenmiştir.	Görsel 6.53	Yazar tarafından düzenlenmiştir.
Görsel 6.13	Yazar tarafından düzenlenmiştir.	Görsel 6.54	Yazar tarafından düzenlenmiştir.
Görsel 6.14	Yazar tarafından düzenlenmiştir.	Görsel 6.55	Yazar tarafından düzenlenmiştir.
Görsel 6.15	Yazar tarafından düzenlenmiştir.	Görsel 6.56	Yazar tarafından düzenlenmiştir.
Görsel 6.16	Yazar tarafından düzenlenmiştir.	Görsel 6.56-1	Yazar tarafından düzenlenmiştir.
Görsel 6.17	Yazar tarafından düzenlenmiştir.	Görsel 6.57	Yazar tarafından düzenlenmiştir.
Görsel 6.18	Yazar tarafından düzenlenmiştir.	Görsel 6.58	Yazar tarafından düzenlenmiştir.
Görsel 6.19	Yazar tarafından düzenlenmiştir.	Görsel 6.59	Yazar tarafından düzenlenmiştir.
Görsel 6.20	Yazar tarafından düzenlenmiştir.	Görsel 6.60	Yazar tarafından düzenlenmiştir.
Görsel 6.21	Yazar tarafından düzenlenmiştir.	Görsel 6.61	Yazar tarafından düzenlenmiştir.
Görsel 6.22	Yazar tarafından düzenlenmiştir.	Görsel 6.62	Yazar tarafından düzenlenmiştir.
Görsel 6.23	Yazar tarafından düzenlenmiştir.	Görsel 6.63	Yazar tarafından düzenlenmiştir.
Görsel 6.24	Yazar tarafından düzenlenmiştir.	Görsel 6.64	Yazar tarafından düzenlenmiştir.
Görsel 6.25	Yazar tarafından düzenlenmiştir.	Görsel 6.65	Yazar tarafından düzenlenmiştir.
Görsel 6.25-1	Yazar tarafından düzenlenmiştir.	Görsel 6.66	Yazar tarafından düzenlenmiştir.
Görsel 6.26	Yazar tarafından düzenlenmiştir.	Görsel 6.67	Yazar tarafından düzenlenmiştir.
Görsel 6.27	Yazar tarafından düzenlenmiştir.	Görsel 6.68	Yazar tarafından düzenlenmiştir.
Görsel 6.28	Yazar tarafından düzenlenmiştir.	Görsel 6.69	Yazar tarafından düzenlenmiştir.
Görsel 6.29	Yazar tarafından düzenlenmiştir.	Görsel 6.69-1	Yazar tarafından düzenlenmiştir.
Görsel 6.30	Yazar tarafından düzenlenmiştir.	Görsel 6.69-2	Yazar tarafından düzenlenmiştir.
Görsel 6.31	Yazar tarafından düzenlenmiştir.	Görsel 6.69-3	Yazar tarafından düzenlenmiştir.
Görsel 6.32	Yazar tarafından düzenlenmiştir.	Görsel 6.70	Yazar tarafından düzenlenmiştir.
Görsel 6.33	Yazar tarafından düzenlenmiştir.	Görsel 6.70-1	Yazar tarafından düzenlenmiştir.
Görsel 6.34	Yazar tarafından düzenlenmiştir.	Görsel 6.70-2	Yazar tarafından düzenlenmiştir.
Görsel 6.35	Yazar tarafından düzenlenmiştir.	Görsel 6.771	Yazar tarafından düzenlenmiştir.
Görsel 6.36	Yazar tarafından düzenlenmiştir.	Görsel 6.72	Yazar tarafından düzenlenmiştir.
Görsel 6.37	Yazar tarafından düzenlenmiştir.	Görsel 6.73	Yazar tarafından düzenlenmiştir.
Görsel 6.38	Yazar tarafından düzenlenmiştir.	Görsel 6.74	Yazar tarafından düzenlenmiştir.
Görsel 6.39	Yazar tarafından düzenlenmiştir.	Görsel 6.75	Yazar tarafından düzenlenmiştir.
Görsel 6.40	Yazar tarafından düzenlenmiştir.	Görsel 6.76	Yazar tarafından düzenlenmiştir.
Görsel 6.41	Yazar tarafından düzenlenmiştir.	Görsel 6.77	Yazar tarafından düzenlenmiştir.
Görsel 6.42	Yazar tarafından düzenlenmiştir.	Görsel 6.78	Yazar tarafından düzenlenmiştir.

Görsel 6.79	Yazar tarafından düzenlenmiştir.	**Görsel 6.119**	Yazar tarafından düzenlenmiştir.
Görsel 6.80	Yazar tarafından düzenlenmiştir.	**Görsel 6.120**	Yazar tarafından düzenlenmiştir.
Görsel 6.81	Yazar tarafından düzenlenmiştir.	**Görsel 6.121**	Yazar tarafından düzenlenmiştir.
Görsel 6.82	Yazar tarafından düzenlenmiştir.	**Görsel 6.121-1**	Yazar tarafından düzenlenmiştir.
Görsel 6.83	Yazar tarafından düzenlenmiştir.	**Görsel 6.122**	Yazar tarafından düzenlenmiştir.
Görsel 6.84	Yazar tarafından düzenlenmiştir.	**Görsel 6.123**	Yazar tarafından düzenlenmiştir.
Görsel 6.85	Yazar tarafından düzenlenmiştir.	**Görsel 6.124**	Yazar tarafından düzenlenmiştir.
Görsel 6.86	Yazar tarafından düzenlenmiştir.	**Görsel 6.125**	Yazar tarafından düzenlenmiştir.
Görsel 6.87	Yazar tarafından düzenlenmiştir.	**Görsel 6.126**	Yazar tarafından düzenlenmiştir.
Görsel 6.88	Yazar tarafından düzenlenmiştir.	**Görsel 6.127**	Yazar tarafından düzenlenmiştir.
Görsel 6.89	Yazar tarafından düzenlenmiştir.	**Görsel 6.128**	Yazar tarafından düzenlenmiştir.
Görsel 6.90	Yazar tarafından düzenlenmiştir.	**Görsel 6.128-1**	Yazar tarafından düzenlenmiştir.
Görsel 6.91	Yazar tarafından düzenlenmiştir.	**Görsel 6.128-2**	Yazar tarafından düzenlenmiştir.
Görsel 6.92	Yazar tarafından düzenlenmiştir.	**Görsel 6.129**	Yazar tarafından düzenlenmiştir.
Görsel 6.93	Yazar tarafından düzenlenmiştir.	**Görsel 6.129-1**	Yazar tarafından düzenlenmiştir.
Görsel 6.94	Yazar tarafından düzenlenmiştir.	**Görsel 6.129-2**	Yazar tarafından düzenlenmiştir.
Görsel 6.95	Yazar tarafından düzenlenmiştir.	**Görsel 6.130**	Yazar tarafından düzenlenmiştir.
Görsel 6.96	Yazar tarafından düzenlenmiştir.	**Görsel 6.131**	Yazar tarafından düzenlenmiştir.
Görsel 6.97	Yazar tarafından düzenlenmiştir.	**Görsel 6.132**	Yazar tarafından düzenlenmiştir.
Görsel 6.98	Yazar tarafından düzenlenmiştir.	**Görsel 6.133**	https://oguzcangenc.com.tr/blog/linq-language-integrated-query-nedir/
Görsel 6.99	Yazar tarafından düzenlenmiştir.	**Görsel 6.134**	Yazar tarafından düzenlenmiştir.
Görsel 6.100	Yazar tarafından düzenlenmiştir.	**Görsel 6.135**	Yazar tarafından düzenlenmiştir.
Görsel 6.101	Yazar tarafından düzenlenmiştir.	**Görsel 6.136**	Yazar tarafından düzenlenmiştir.
Görsel 6.102	Yazar tarafından düzenlenmiştir.	**Görsel 6.136-1**	Yazar tarafından düzenlenmiştir.
Görsel 6.103	Yazar tarafından düzenlenmiştir.	**Görsel 6.137**	Yazar tarafından düzenlenmiştir.
Görsel 6.103-1	Yazar tarafından düzenlenmiştir.	**Görsel 6.138**	Yazar tarafından düzenlenmiştir.
Görsel 6.104	Yazar tarafından düzenlenmiştir.	**Görsel 6.139**	Yazar tarafından düzenlenmiştir.
Görsel 6.105	Yazar tarafından düzenlenmiştir.	**Görsel 6.140**	Yazar tarafından düzenlenmiştir.
Görsel 6.106	Yazar tarafından düzenlenmiştir.	**Görsel 6.141**	Yazar tarafından düzenlenmiştir.
Görsel 6.107	Yazar tarafından düzenlenmiştir.	**Görsel 6.142**	Yazar tarafından düzenlenmiştir.
Görsel 6.108	Yazar tarafından düzenlenmiştir.	**Görsel 6.143**	Yazar tarafından düzenlenmiştir.
Görsel 6.109	Yazar tarafından düzenlenmiştir.	**Görsel 6.144**	Yazar tarafından düzenlenmiştir.
Görsel 6.110	Yazar tarafından düzenlenmiştir.	**Görsel 6.145**	Yazar tarafından düzenlenmiştir.
Görsel 6.111	Yazar tarafından düzenlenmiştir.	**Görsel 6.146**	Yazar tarafından düzenlenmiştir.
Görsel 6.112	Yazar tarafından düzenlenmiştir.	**Görsel 6.147**	Yazar tarafından düzenlenmiştir.
Görsel 6.113	Yazar tarafından düzenlenmiştir.	**Görsel 6.148**	Yazar tarafından düzenlenmiştir.
Görsel 6.114	Yazar tarafından düzenlenmiştir.	**Görsel 6.149**	Yazar tarafından düzenlenmiştir.
Görsel 6.115	Yazar tarafından düzenlenmiştir.	**Görsel 6.150**	Yazar tarafından düzenlenmiştir.
Görsel 6.116	Yazar tarafından düzenlenmiştir.	**Görsel 6.151**	Yazar tarafından düzenlenmiştir.
Görsel 6.117	Yazar tarafından düzenlenmiştir.	**Görsel 6.152**	Yazar tarafından düzenlenmiştir.
Görsel 6.118	Yazar tarafından düzenlenmiştir.	**Görsel 6.153**	Yazar tarafından düzenlenmiştir.
Görsel 6.118-1	Yazar tarafından düzenlenmiştir.	**Görsel 6.154**	Yazar tarafından düzenlenmiştir.

KAYNAKÇA

- **https://medium.com/**@nuriyavuz2.71/sql-101-ve-sql-injection-101-d2b35dff52de - 01.12.2020 - 10.00
- **https://www.oracle.com/tr**/database/what-is-database.html - 02.12.2020 - 12.00
- **https://www.ozcanbayri.com.tr**/big-data-nedir/ - 02.12.2020 - 12.30
- **https://kod5.org**/t-sql-veri-tipleri/ - 02.12.2020 - 13.00
- **https://ceaksan.com/tr/**primary-unique-foreign-key - 03.12.2020 - 10.00
- **https://www.dijitalders.com/**icerik/2378/veritabani_nedir.html - 23.12.2020 - 09.00
- **https://imsiyat.com** - 05.12.2020 - 10.00
- **https://icon-icons.com**/icon/education-school-lunch-box-break-bag-food/133449 - 05.12.2020 - 10.00
- **https://icon-icons.com**/pack/Modern-Education-And-Knowledge-Power/1804 - 05.12.2020 - 10.00
- **https://medium.com**/@kdrcandogan/entity-framework-nedir-6aed4dcf6328 - 07.12.2020 - 12.00
- **https://beylikweb.com/**veri-madenciligi-nedir-tanimi-amaci-ve-teknikleri/ 02.12.2020 - 09.00
- **http://www.sercangulgeze.com.tr**/ado.net-nedir-nasil-kullanilir.html - 06.12.2020 - 09.00