

# C# EĞİTİM KİTABI

---

MURAT YÜCEDAĞ

# İÇİNDEKİLER

<b>BÖLÜM 1: GİRİŞ</b>	<b>1</b>
Visual Studio	4
İndirme ve Kurulum	4
C# Nedir?	6
C# ile Neler Yapılabilir?	7
Neler Öğrendik?	8
<b>BÖLÜM 2: TEMEL ARAÇLAR VE KULLANIMLARI</b>	<b>11</b>
Merhaba Dünya!	12
Araçlar	15
Button Aracı	16
Label Aracı	17
TextBox Aracı	19
ComboBox Aracı	22
ListBox Aracı	25
MaskedTextBox Aracı	27
PictureBox Aracı	29
GroupBox Aracı	30
Panel Aracı	31
RichTextBox Aracı	32
Neler Öğrendik?	34

**BÖLÜM 3: DEĞİŞKENLER 37**

String Değişkenler	38
Int Değişkenler	43
Klavyeden Değer Alma ve Dönüşümler	48
Double Değişkenler	51
Char Değişkenler	54
Byte Değişkenler	56
Sbyte Değişkenler	57
Short Değişkenler	59
Ushort Değişkenler	60
Float Değişkenler	60
Decimal Değişkenler	61
Bool Değişkenler	62
Neler Öğrendik?	64

**BÖLÜM 4: OPERATÖRLER 67**

Operatör Nedir?	68
Aritmetik Operatörler	68
Toplama Operatörü (+)	68
Çıkarma Operatörü (-)	69
Çarpma Operatörü (*)	70
Bölme Operatörü (/)	70
Arttırma Operatörü (++)	71
Azaltma Operatörü (--)	71
Mod Alma Operatörü (%)	72
Karşılaştırma Operatörleri	73
Eşitlik Operatörü (==)	73
Küçüktür Operatörü (<)	74

Büyükdür Operatörü (>)	75
Küçük Eşittir Operatörü (<=)	75
Büyük Eşittir Operatörü (>=)	76
Eşit Değildir Operatörü (!=)	77
Bitsel Operatörler	77
ve Operatörü (&)	77
Değil operatörü (~)	78
Veya Operatörü ( )	78
Özel Veya Operatörü (^)	78
Mantıksal Operatörler	78
ve Operatörü (&&)	78
veya Operatörü (  )	79
Değil Operatörü (!)	80
Atama ve İşlemler Atama Operatörleri	80
Atama Operatörü (=)	81
İşlemler Atama Operatörü (+= -=)	81
Özel Amaçlı Operatörler	82
Koşul Operatörü (?:)	82
Tür Değiştirme Operatörü (())	83
Dizi Operatörü ([ ])	83
Neler Öğrendik?	83
<b>BÖLÜM 5: KARAR YAPILARI</b>	<b>85</b>
Karar Yapıları Nedir?	86
Eğer (if) Komutu	86
CheckBox	97
RadioButton	98
Switch - Case	102
Neler Öğrendik?	107

**BÖLÜM 6: DİZİLER 109**

Dizi Nedir?	110
Dizi Oluşturma	110
Dizi Metotları	121
Length Metodu	121
Sort Metodu	122
Reverse Metodu	123
IndexOf Metodu	124
Max-Min Metodu	125
İki Boyutlu Diziler	126
Foreach Döngüsü	127
Neler Öğrendik?	131

**BÖLÜM 7: METOTLAR 133**

Metot Nedir?	134
Geriye Değer Döndürmeyen Metotlar	134
Geriye Değer Döndüren Metotlar	140
Erişim Belirleyiciler	143
Rekürsif (Yinelen) Metotlar	145
Metotları Aşırı Yükleme	146
Neler Öğrendik?	147

**BÖLÜM 8: SINIFLAR 149**

Sınıf Nedir?	150
Nesne Nedir?	150
Kapsülleme	155
Kalıtım (Miras)	158
Çok Biçimlilik	161
Neler Öğrendik?	164

**BÖLÜM 9: EKSTRA ARAÇLAR 167**

Menu Strip Kullanımı	168
Web Browser Kullanımı	172
Context Menu Strip	173
Timer (Zamanlayıcı)	175
Chart	179
Progress Bar	181
Neler Öğrendik?	182

**BÖLÜM 10: EKSTRA YAPILAR 185**

Random Sınıfı	186
Formlar Arası Geçiş İşlemleri	189
Matematik Fonksiyonları	192
String (Metin) Fonksiyonları	196
Concat Fonksiyonu	197
Length Fonksiyonu	197
IndexOf Fonksiyonu	198
StarWith Fonksiyonu	198
Trim Fonksiyonu	199
ToUpper Fonksiyonu	199
ToLower Fonksiyonu	200
Remove Fonksiyonu	200
Replace Fonksiyonu	201
Substring Fonksiyonu	201
Dinamik Araçlar	202
Point Yapısı	204
Tarih Zaman Fonksiyonları	206
Neler Öğrendik?	208

<b>BÖLÜM 11: DOSYA GİRİŞ/ÇIKIŞ İŞLEMLERİ</b>	<b>211</b>
Giriş Çıkış İşlemleri Nedir?	212
Folder Browser Dialog	212
Open File Dialog	214
Save File Dialog	216
Neler Öğrendik?	220
<b>BÖLÜM 12: SQL VERİTABANI</b>	<b>223</b>
SQL Nedir?	224
SQL Kurulumu	224
Neler Öğrendik?	231
<b>BÖLÜM 13: SQL VERİTABANI, VERİ TİPLERİ VE TABLOLAR</b>	<b>233</b>
Veritabanı Kavramı	234
Veri Tipleri	236
Neler Öğrendik?	241
<b>BÖLÜM 14: SORGULARLA SQL</b>	<b>243</b>
Temel SQL Sorguları	244
Select	244
Insert	247
Delete	248
Update	248
Operatörler	249
Alt Sorgular	253
Gruplandırılmalar	254
İlişkili Tablolar	257
Birleştirmeler	263
Prosedürler	264
Tetikleyiciler	265
Neler Öğrendik?	267

**BÖLÜM 15: C# FORM İLE SQL 269**

Kitaplık Projesi	270
Form Ara Yüzünün Oluşturulması	271
Datagridview Aracı	273
Bağlantı Adresinin Alınması	273
Verilerin Listelenmesi (Select)	278
Yeni Kitap Ekleme (Insert)	282
Verileri Araçlara Taşıma	284
Verileri Silme	286
Verileri Güncelleme (Update)	287
Neler Öğrendik?	289

**BÖLÜM 16: RAPORLAMA 291**

Raporlama Nedir?	292
Report Viewer Aracı	292
Neler Öğrendik?	300

**BÖLÜM 17: EVENTS VE DELEGATE 303**

Delegate	304
Events (Alt Başlık)	307
Neler Öğrendik?	310

**BÖLÜM 18: HATA KONTROLLERİ 313**

Hatalar	314
Try - Catch Bloğu	314
Özel Durum Formatları	317
Finally	320
Neler Öğrendik?	321



**BÖLÜM 19: KOLEKSİYONLAR 323**

Koleksiyon Sınıfları	324
Arraylist	324
Sorted List	325
HashTable	326
Stack	327
Kuyruk	328
List	329
Neler Öğrendik?	331

**BÖLÜM 20: ÖĞRENCİ NOT KAYIT SİSTEMİ 333**

Proje Amacı	334
Veritabanı ve Tabloların Oluşturulması	334
Tetikleyicilerin Oluşturulması	339
Prosedürlerin Oluşturulması	342
Giriş Formunun Tasarlanması	343
Bağlantı Sınıfının Oluşturulması ve Giriş Formu Kodları	346
Öğretmen Formu Tasarımı	355
Formlar Arası Veri Taşıma İşlemiyle Numaranın Taşınması	357
Taşınan Numaraya Göre Öğretmenin Ad Soyad Bilgisini Getirme	359
Öğrencileri Listeleme	361
Notları Listeleme	362
Yeni Öğrenci Kaydı	364
Öğrenci Bilgilerini Araçlara Taşıma	366
Öğrenci Bilgilerini Silme	368
Öğrenci Bilgilerini Güncelleme	368

Duyuru İşlemleri Oluşturma Formu	373
Duyuru Listesi Formu	380
Mesajlar Formu	383
Öğrenci Formu	391
Öğrenci Formu Tasarımı	392
Öğrenci Formuna Giriş ve Numaranın Taşınması	392
Numaraya Göre İsim Bilgisinin Getirilmesi	394
Numaraya Göre Sınav Notlarının Getirilmesi	395
Mesajlar Formu	396
Hesap Makinesi	398
Duyurular Formu	398
Çıkış	399
Neler Öğrendik?	400
Son Söz	401
<b>Dizin</b>	<b>402</b>

# 1

## GİRİŞ

### BU BÖLÜMDE

Visual Studio	4
C# Nedir?	6
Neler Öğrendik?	8

Bu bölümde programlamanın ilk ve en önemli adımı olan algoritma kavramını öğrenerek başlayıp; kod, yazılım, derleyici gibi teknik terimlere değineceğiz. Projelerimizi kodlayacağımız program olan Visual Studio hakkında bilgi sahibi olacak ve kitabımızın özünü oluşturan C# programlama dili ile neler yapılabileceğine genel hatları ile değineceğiz.

Nasıl başlayacağım konusunda hep kararsız kalmışımdır. İlk cümleleri bulmak benim için zordu ve zor olmaya devam ediyor. Ama ilk cümleleri kurduktan sonra devamı çorap söküğü gibi geliyor. 21. yy'da 3 yaşında ki çocuktan 60 küsur yaşında ki amca ve teyzelere kadar herkesin elinde telefonların tabletlerin olduğu bir dönem yaşıyoruz. Bundan birkaç sene öncesine kadar **o telefonu bırak eline yapıştı** diyen ebeveynler bugün sosyal medya kullanıyorlar. İnsanlar için zaman kavramı çok büyük önem taşıyor. Markete gitmek yerine market size gelsin ilkesi yaşamamızın bir parçası oldu. Her şeyin sadece bir tık ötede olduğu küresel bir dünyada yaşıyoruz.

Dünyanın en gelişmiş ülkelerini bu kadar güçlü kılan şey **teknoloji** ve **yazılım** alanlarında yaptıkları yatırımlar ve attıkları adımlardır. İlkokullarda dahi kodlama dersi zorunlu hale getirildi. Kodlama ve yazılım artık her yerde! Kendini bu alanda geliştirmek, sistemin tüketen değil üreten bir parçası olmak isteyenler için hazırlamış olduğum kitabımda sizlerden ricam acele etmeden istikrarlı bir şekilde tüm bölümleri okuyup bölüm içinde ki uygulamaları yaparak pratiğinizi arttırıp kodlama dünyasına büyük bir adım atmanızdır.

Büyük sporcuların, en hızlı koşucu atletlerin dahi bu işe ilk önce adım atarak başladığını sakın unutmayın.

Günlük hayatta karşımıza çok sık çıkan bir kavram vardır algoritma. Peki, nedir bu algoritma kısaca tanımlayacak olursak; "bir problemin çözümündeki adımlar bütünüdür." Yaşadığımız her anda verdiğimiz bütün kararlar bir algoritma sonucudur. Algoritmanın en önemli kurallarından birisi adımlara ayrılmış olmasıdır. Misal çok acıktınız ve makarna yapmanız gerekiyor. Öyleyse bu işi algoritmaya dökelim.

**Problem:** Acıkmış olmamız.

**Çözüm:** Karnımızı doyurmak.

**Yöntem:** Makarna yapmak.

### Algoritma Adımları

1. Tencereyi çıkaralım.
2. Yarisından biraz fazlaca su koyalım.
3. Ocağımızı yakalım.
4. Tencereyi ocağın üstüne koyalım.
5. Su biraz ısınmaya başlayınca içerisine makarnamızı koyalım.

6. Yapışmaması için tenceremize tuz ve yağ ekleyebiliriz.
7. Makarna yumuşayana kadar haşlanmaya devam etsin.
8. Haşlanma işlemi bitince ocağı kapatalım.
9. Tenceremizde ki makarnayı süzgece boşaltalım.
10. Hamurlaşmaması için biraz soğuk suya turalım.
11. Tenceremizi yeniden ocağa koyup ocağı yakalım.
12. İçerisine biraz saçla ve yağ ekleyip karıştıralım.
13. Süzgeçte ki makarnayı tencereye boşaltalım.
14. Saçla eriyene kadar karıştıralım.
15. Ocağı kapatıp tencerede ki makarnayı tabaklara servis edelim.
16. Afiyet olsun :)

İşte bu adımları takip eden birisi makarna yapabilecektir. Algoritmalar çok göreceli kavramlardır. **Örneğin;** siz makarna içine kekik ve pul biber eklemek isterseniz algoritma adımları arasına ekleme yaparak işlemleri düzeltebilirsiniz.

Algoritma kavramının zihinlerde yavaş yavaş oturmaya başladığını düşünüyorum. Peki, “yazılım” nedir? Vikipedi’de şöyle güzel bir tanım mevcut; “değişik ve çeşitli görevler yapma amaçlı tasarlanmış elektronik aygıtların birbirleriyle haberleşebilmesini ve uyumunu sağlayarak görevlerini ya da kullanılabilirliklerini geliştirmeye yarayan makine komutlarıdır. Yazılım, elektronik aygıtların belirli bir işi yapmasını sağlayan programların tümüne verilen isimdir.”

Günlük hayatımızın her alanında var olan insan yaşamını kolaylaştıran tüm teknolojik unsurlar birer yazılım ürünüdür. Televizyondan bilgisayarlara, Mp3 çalarlardan telefonlara kadar hepsi yazılımın elektronik ile birleşmesiyle ortaya çıkan icatlardır.

Kod dediğimiz kavramda kısaca yazılımların oluşması için programcılar tarafından ortaya konulan bilgisayarın anlayacağı komutlar bütünüdür. Programlama dillerini de tıpkı İngilizce, Almanca veya Türkçe gibi birer dil olarak düşünelim. Nasıl ki bir turist ile konuşup anlaşmak için karşılıklı olarak birbirimizin dillerini bilmemiz gerekiyorsa kodlamada da o kodun dilini bilip bilgisayarın anlayacağı formata çevirerek anlamlı yapıları haline getirmemiz lazım. Yazılım ve kodlamada başarı için 3 temel nokta çok önemli;

Pratik + İstikrar + Zaman

Ve tabi biraz matematik, biraz İngilizce çokça hayal gücü gerekir. Sonuçta yazılım sektöründeki insanlar müşterilere hayal güçlerinin somut hallerini satmak-

# 3

## DEĞİŞKENLER

### BU BÖLÜMDE

String Değişkenler	38
Int Değişkenler	43
Double Değişkenler	51
Char Değişkenler	54
Byte Değişkenler	56
Sbyte Değişkenler	57
Short Değişkenler	59
Ushort Değişkenler	60
Float Değişkenler	60
Decimal Değişkenler	61
Bool Değişkenler	62
Neler Öğrendik?	64

Bu bölümde değişken yapısını, değişkenlere neden ihtiyaç duyulduğunu, temel değişkenlerin neler olduğunu, aritmetik işlemleri, dönüşüm işlemlerini String, Int, Double, Char-Byte, Float, Bool gibi temel değişken türlerini ve neden değerlerin bir tek başlık altında toplanmayıp değişkenler adı altında farklı dallara ayrıldığını öğreneceğiz.

Günlük hayattan bir örnek ile başlayalım yeni konumuza. Düşününki elinizde bir mantar pano tahtanız var ve küçük not kâğıtlarını tahtaya bir şekilde yerleştirmeniz gerekmekte. Bunun için aşağıda belirttiğim veya sizin aklınıza gelen farklı yöntemler tercih edilebilir.

- » Mantar pano iğnesi kullanılabilir,
- » Raptiye yardımı ile tutturulabilir,
- » Sıvı yapıştırıcı kullanılabilir,
- » Tahtanın arası sıkıştırılabilir,
- » Ve başka yöntemler...

Peki, soru şu, en uygun yöntem hangisi?

Pek tabiidir ki mantar pano iğnesi kullanmak. Değişken kavramını; verileri bellekte en uygun şekilde tutmak için gerekli olan komutlardır, şeklinde bir ifadeyle açıklayabiliriz. Programlamada en çok kullanılan değişkenleri tanıyarak başlayalım. Bu bölümde 4 ana değişken kullanacağız ilerleyen derslerde burada ki alanlara ek olarak birkaç değişkene daha değiniyor olacağız. Kullanacağımız değişkenleri sıralayacak olursak;

- » String Değişkenler,
- » Int Değişkenler,
- » Double Değişkenler,
- » Char Değişkenler,

## STRING DEĞİŞKENLER

4 temel değişkenimizden ilki olan string değişkenler ile konumuza devam edelim. String değişkenler bellekte alfabetik verilerin tutulması amacıyla kullanılan değişkenlerdir. Peki, değişkenlerimize isim verirken nelere dikkat etmeliyiz?

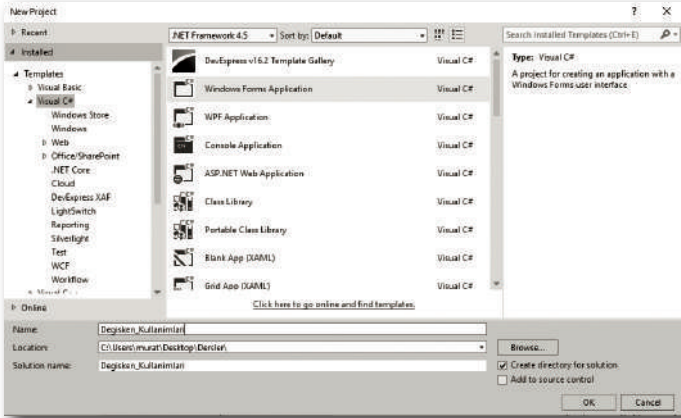
- » Değişken adları kullanılacağı değerle ilgili olmalıdır.
- » 2 kelimedenden oluşan değişken adları ya birleşik yazılmalı ya da alt tire gibi bir semboller birleştirilmelidir.
- » Değişkenlerde 2 kelime olması durumunda ayrı ayrı yazmak hataya sebep olacaktır.
- » Çok uzun değişken isimlerinden kaçınılmalıdır.
- » Oldukça net ve anlaşılır değişken isimleri tercih edilmelidir.
- » Değişkenler sayı değerleri ile başlamaz.
- » Değişkenler içinde semboller kullanılmamalıdır.

Değişken tanımlaması yapılırken önce değişkenin türü belirtilir sonra değişkenin adı yazılır. Değişkenlere bazı özel komutlar haricinde istediğimiz isimleri verebiliriz.

**NOT** C#, JAVA, C, C++ gibi dillerde komut satırları noktalı virgül ile sonlanır.

Öyleyse yeni bir tane proje açalım ve formumuza birer tane button ve label ekleyerek uygulamalarımıza başlayalım. Şekilde ki gibi **New Project** diyoruz ve proje adını veriyoruz.

**NOT** Visual Studio Türkçe karakteri desteklemektedir ancak olabildiğince Türkçe karakterlerden uzak durmaya çalışın.



## Uygulama 1

String olarak tanımlanan değişkende ki veriyi label'a yazdıran programı kodlayalım.

```
private void button1_Click(object sender, EventArgs e)
{
    string metin;
    metin = "Merhaba Dünya Bugün Hava Soğuk";
    label1.Text = metin;
}
```

Metin isminde bir değişken oluşturduk ve bu değişkene bir yazı yazdırıp bunu da Labelde gösterdik. Oluşturduğumuz değişkenimiz **String** türündedir. Değişkenimizin ismi **metin** olup siz burada **metin** kelimesi yerine istediğiniz ismi



Int değişkenler özellikle aritmetik işlemlerin olmazsa olmazıdır. Basit toplama işleminden karenin alan hesaplamasına, stok azaltmadan ücret hesaplamaya kadar binlerce hatta daha fazla aritmetik işlem int aracılığı ile kolayca gerçekleştirilir. Öyleyse örnek uygulamalar üzerinde biraz daha somut olarak görelim int değişken kullanımını.

### Uygulama 6

Kod kısmından girilen bir sayıyı label'a yazdıran programı kodlayalım.

Burada dikkat etmemiz ve yeni öğreneceğimiz önemli bir nokta var. String değişkenlerde ilgili değişkeni olduğu gibi label, TextBox veya kullanacağımız araç hangisi ise ona yazdırabiliyorduk. String haricinde ki değişkenlerde ise durum biraz farklı. C#'da araçların büyük çoğunluğu string değişkenlere göre tanımlanmıştır. Bundan dolayı string ifadeler dışında ki değişkenleri yazdırırken sonuna `.toString()` ifadesi yazılması gerekiyor. `To` kelimesi İngilizce'de **yön belirteci** olarak kullanılmakta. String zaten bizim değişken türümüz. Yani ifadeye string'e anlamı veriyoruz. `ToString` ifadesinin sonunda parantez açılıp kapanmasının sebebi bunun bir metod oluşudur. Metotlar kısmında bu noktaya detaylı şekilde değineceğiz. Öyleyse kodlarımızı yazalım.

```
private void button1_Click(object sender, EventArgs e)
{
    int sayi;
    sayi = 24;
    label1.Text = sayi.ToString();
}
```

String değişkenlerdeki tanımlama yönteminin aynısını int değişkenler içinde kullandık. Sayı isminde bir değişken tanımladık. Değişkenimizin sayı olarak yazmamızın sebebi Türkçe karakter kullanmaktan kaçınmamızdır. C# dilinin Türkçe karakter konusunda bir problemi yoktur. Ancak yazılım dünyasında evrensel dil İngilizce olduğundan dolayı Türkçe karakterlerden elimizden geldiğince uzak duralım. Tanımladığımız sayı değişkenimize 24 değerini verdik. Değişkenimiz alfabetik bir veri olmadığı için çift tırnak sembolü kullanmadık. Label aracımıza değişkenimizi yazdırmak için `label1.text=sayi` ifadesinden sonra bir dönüşüm komutu uyguladık.

Burada ki dönüşüm komutunun amacı şudur; C# formda kullanılan araçların neredeyse yüzde 90'ından fazlası string formatta değerler alır. Bizim değişkenimiz tam sayı türünde bir değişken olduğundan Label aracımıza değerimizi

yazdırmak için ToString metodunu kullandık. Komutumuzun sonunda parantez açıp kapatmamızın sebebi ilgili komutun metot olduğundan kaynaklanıyor. İlerleyen bölümlerde parantezlerimizin içini dolduracağız.



### Uygulama 7

Kod kısmından girilen iki sayıyı toplayıp sonucu label'a yazdıran programı kodlayalım.

**NOT**

Dönüşümlerden önce Convert komutu kullanılıp sonra dönüşüm yapılacak değer seçilir.

```
private void button1_Click(object sender, EventArgs e)
{
    int sayi;
    sayi = Convert.ToInt16(textBox1.Text);
    label1.Text = sayi.ToString();
}
```

Öncelikle ihtiyaçlarımızı belirleyelim. 2 sayıyı toplayacaksa 3 tane değişkenimiz olması gerekiyor. Bunlar sayı1, sayı2, sonuç olayı tetiklemek için bir button ve sonucu yazdırmak için bir label. Öyleyse kodlarımızı yazalım.

```
private void button1_Click(object sender, EventArgs e)
{
    int sayi1, sayi2, sonuc;
    sayi1 = 24;
    sayi2 = 9;
    sonuc = sayi1 + sayi2;
    label1.Text = sonuc.ToString();
}
```

# 6

## DİZİLER

### BU BÖLÜMDE

Dizi Nedir?	110
Dizi Metotları	121
İki Boyutlu Diziler	126
Foreach Döngüsü	127
Neler Öğrendik?	131

Bu bölümde dizi kavramını, dizilerin nerelerde kullanıldığı ve programlama açısından önemini, foreach döngüsünü, array sınıfına bağlı metotları, çok boyutlu dizi kavramını, metotlar ile nasıl hazır işlemler yapıldığını öğreneceğiz.

## Dizi Nedir?

Aynı veri tipindeki çok sayıda veriyi bir arada tutmak için kullanılan yapılardır. Yaptığımız uygulamalarda birbiriyle ortak paydada bulunan değerler olabilir. Bu verilere birkaç örnek verecek olursak; "çift sayılar dizisi, renkler dizisi, kişiler dizisi, bir sınıfta bulunan öğrenciler dizisi, şehirler dizisi, a harfi ile başlayan meyveler dizisi..." şeklinde örnekler arttırılabilir. Diziler programlama literatüründe karşımıza **array** başlığıyla da çıkabilir. Array yapısı yani diziler aslında bir sınıftır. Bu bölümde sınıf kavramına kısaca değiniyor olacağız. Detaylı anlatımını ilerleyen sayfalarda ki sınıf bölümümüzde bulabilirsiniz.

## Dizi OLUŞTURMA

Dizi oluşturma işlemini birden fazla şekilde yapabiliriz. Bunlardan birincisi sınıf nesnesi türeterek oluşturmaktadır. Öncelikle sınıf kavramına değinelim; Sınıflar belli kod bloklarını bir arada tutan ve özellikle kod tekrarının önlenmesinde büyük önem arz eden yapılardır. Sınıfların özelliklerini kullanabilmek için ilgili sınıftan nesne türetilir. Somut bir örnek verecek olursak; apartmanı bir sınıf olarak düşünelim. Apartmanın içerisinde ki daireler birer nesne, dairenin iç özellikleri: **renk, kat, fiyat** vs. bunlarda bizim özelliklerimizdir. Sınıflarda çatı kısmı sabittir. Değişen şey özellikler ne niteliklerdir. Yani 10 katlı bir apartman düşünürsek ve her katta bir tane daire olduğunu varsayarsak bu dairelerin tamamında oda sayısı, banyo ve tuvalet yerleri, balkon kısımları ve mutfakları aynı yerededir. Ancak birinci kattaki dairede çocuk odası olarak kullanılan yeri bir üst kattaki aile yatak odası olarak kullanabilir. Yani nesne tanımlandıktan sonra ilgili değer in alacağı nitelikler farklı olabilir. Sınıflardan nesne tanımlaması yapılırken **new** anahtar sözcüğü kullanılır. Bu başlıklarla ilgili daha detaylı bilgiyi sınıflar bölümünde veriyor olacağız. Sınıf tanımlama işlemi şu şekilde yapılır;

---

```
SınıfAdı NesneAdı = new SınıfAdı
```

---

Önce sınıfın ismi yazılır, sonra o sınıftan türetilecek nesne için herhangi bir ad verilir. Bu ad tıpkı değişken tanımlamasında olduğu gibi istenilen isimler seçilerek verilebilir. Sonra **eşittir** sembolü yazılır ve **new** anahtar sözcüğü kullanılıp sonrasında yeniden sınıfın ismi yazılır. Böylece o sınıftan ilgili nesneyi türetmiş oluruz. Bir yapının dizi olduğunu göstermek için **[]** sembolleri kullanılır. Dizi tanımlamaları yapılırken dikkat edilmesi gereken önemli bir nokta; dizi adları olabildiğince doğru ve anlamlı isimlerden seçilmelidir.

Dizi tanımlaması şu şekilde yapılır.

```
Değişken türü [ ] Dizi Adı = new Değişken Türü [Eleman Sayısı]
```

Dikkat edilmesi gereken bir diğer nokta da ikinci kez değişken türünü yazdıktan sonra süslü parantez bloğu içine bu dizinin kaç elemandan oluşacağını bildirmesidir. Dizi verileri bellekte tutulur ve eleman sayısı önceden yazılarak bellekte buna göre yer tahsisi yapılır. Dizilerde index isminde bir kavram vardır. Index dizinin bellekte kaçınıcı sırada tutulacağını belirler.

**NOT**

Dizilerde index değeri 0'dan başlar. Çünkü programlamada sayma işlemleri 1'den değil 0'dan başlamaktadır.

```
public static bool KartKontrol(string KartNumara)
{
    int toplam = 0;
    for (int i = 0; i < 16; i++)
    {
        int sayi = Convert.ToInt32(KartNumara[i].ToString());

        if (i % 2 == 0)
        {
            sayi = sayi * 2;
            if (sayi.ToString().Length == 2)
                sayi = Convert.ToInt32(sayi.ToString().Substring(0, 1)) + Convert.ToInt32(sayi.ToString().Substring(1, 1));
        }

        toplam += sayi;
    }

    if (toplam % 10 == 0)
        return true;
    else
        return false;
}

static void Main(string[] args)
```

Şekilde 1'de 10 elemandan oluşan bir dizi boş dizi bulunmaktadır. Bellekte bu boş dizi için 10 elemanlık yer ayrılmıştır. İlk index değeri 0'dan başlamış, son index değeri 9'da bitmiştir.

### Uygulama 1

Şekil 2'de ki diziyi çözümlayelim.

```
        return false;
    }
    static void Main(string[] args)
    {
        Console.WriteLine(KartKontrol("4388576018410707"));
        Console.Read();
    }
}
```

- » Dizimiz 6 elemandan oluşmaktadır.
- » Dizimiz kişi adlarını tutan bir “kişiler” dizisi olabilir.
- » Dizimizin ilk indexinde “Yunus” son indexinde “Sinem” değerleri vardır.
- » Elemanlarımız alfabetik veriler olduğundan dizimiz string değişken türünde tanımlanıp oluşturulmuştur.
- » Dizimizin son index numarası 5’tir.
- » Dizilerde son indexin kaç olduğu dizi uzunluğundan 1 çıkarılarak bulunabilir.

## Uygulama 2

5 Elemandan oluşan bir renkler dizisi tanımlayıp bu diziyeye veri girişi yapalım ve dizinin 2. index’inde bulunan elemanı yazdıralım.

```
private void button1_Click(object sender, EventArgs e)
{
    string[] renkler = new string[5];
    renkler[0] = "sarı";
    renkler[1] = "mavi";
    renkler[2] = "beyaz";
    renkler[3] = "turuncu";
    renkler[4] = "kırmızı";
    label1.Text = renkler[2];
}
```



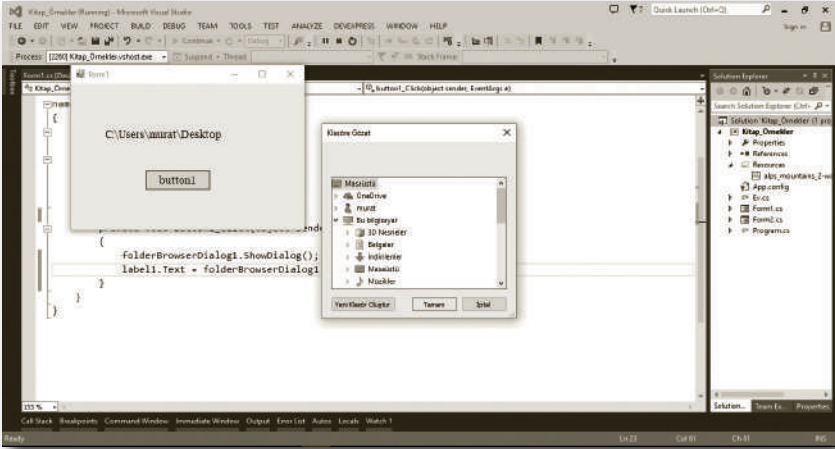
renkler isminde string bir dizi oluşturduk ve dizimizin boyutunu 5 olarak belirledik. renkler[0] ifadesiyle renkler dizimizin ilk index’ine değer atamamızı yaptık. Bu işlemi sırasıyla diğer elemanlarımız içinde yapıp dizimizin değer atama işlemini bitirdikten sonra Label aracımıza dizimizin 2. index’inde bulunan değeri yazdırdık. Dizi string olarak oluşturulduğu için değerlerimizi de string formatta yani çift tırnak sembolü arasına yazdık.

# DOSYA GİRİŞ/ÇIKIŞ İŞLEMLERİ

## BU BÖLÜMDE

Giriş Çıkış İşlemleri Nedir?	212
Folder Browser Dialog	212
Open File Dialog	214
Save File Dialog	216
Neler Öğrendik?	220

Bu bölümde giriş çıkış işlemlerini, dosya oluşturup bunlara erişim sağlamayı, form üzerinden metin dosyalarına müdahale etmeyi, FolderBrowserDialog, SaveFileDialog ve OpenFileDialog araçlarını, System IO kütüphanesinin kullanmayı öğreneceğiz.



1. uygulamamıza göre yalnızca bir satır fark olan örneğimizde Label aracımıza FolderBrowserDialog aracının SelectedPath yani seçilen yolunu metin olarak yazdırdık. İlerleyen örneklerimizde SelectedPath özelliği dosyalarımızı kaydetmek istediğimiz konumlar için kullanacağız.

## OPEN FILE DIALOG

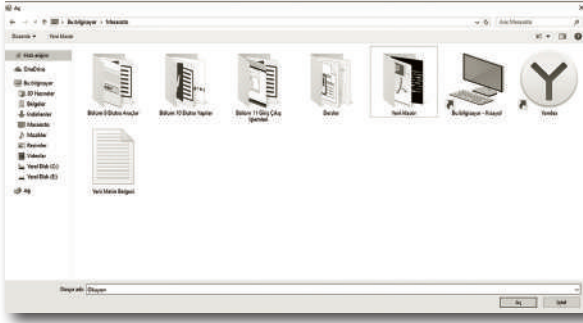
Dosya diyalog/iletişim kutusu aç anlamına gelen, FolderBrowserDialog aracından farklı olarak sadece klasörleri değil, klasör dizinleri içerisinde bulunan dosyaları da gösteren, listeleyen araçtır. Örneğin; form içerisine bir resim dosyası, müzik dosyası ve ya başka herhangi bir dosyanın göz at seçeneği ile eklenmesi istenmektedir. İşte böyle bir durumda OpenFileDialog aracı devreye girecektir. Tıpkı FolderBrowserDialog aracında olduğu gibi karşımıza bir diyalog/iletişim penceresi açarak istediğimiz dosyayı seçip işlemimizi yapabileceğiz. Öyleyse kullanım şeklini inceleyelim.

### Uygulama 3

Buttona tıkladığımız zaman OpenFileDialog aracı ile iletişim kutusunu açan kod bloğunu yazalım.

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.ShowDialog();
}
```



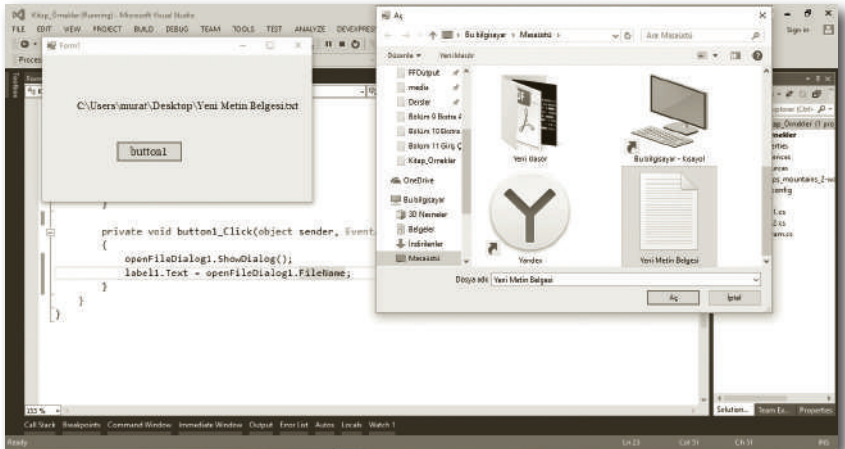


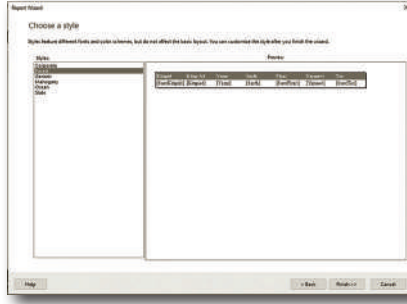
FolderBrowserDialog aracımda olduğu gibi Butonumuza tıklayıp ilgili diyalog penceresini açtık. Ancak bu kez sadece klasörleri ve konumlar değil, dizinler içerisinde bulunan dosyaları da gösterdi. Böylece FolderBrowserDialog aracı ile OpenFileDialog aracı arasındaki fark daha net bir şekilde anlaşılmıştır. Şimdi örneklerimize devam edelim.

#### Uygulama 4

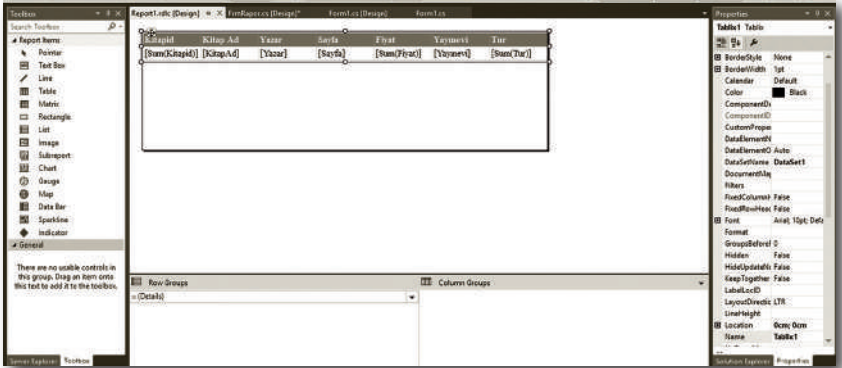
OpenFileDialog penceresinden seçilen herhangi bir dosyanın yolunu Label'e yazdıran programı kodlayalım.

```
private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.ShowDialog();
    label1.Text = openFileDialog1.FileName;
}
```

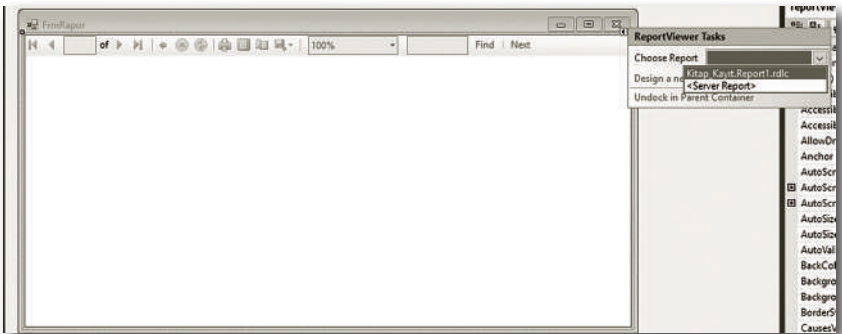




Karşımıza bu şekilde boş bir rapor penceresi açıldı. Bu penceredeki alan aslında raporumuzun uzantısı sonucunda oluşan alandır. Özellikler penceresinden font, renk, yazı stili ve benzeri pek çok nitelik değiştirilebilir.



Raporlama formumuza geri dönelim. Report Viewer aracımızın üst kısmında bulunan simgeye yeniden tıklayalım. Burada **Choose Report** yani rapor seç yazan alandan oluşturmuş olduğumuz raporu seçiyoruz.



Şimdi projemizin ilk formuna geri dönelim ve en alta **raporlar** isminde bir buton ekleyelim. Bu butona tıkladığımız zaman raporlar formumuz açılacaktır.

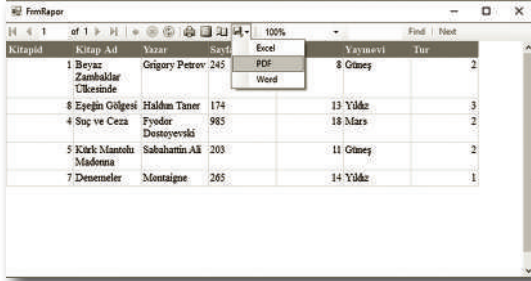
Butonumuza çift tıklayıp formlar arası geçiş kodumuzu yazalım.

```
private void BtnRaporlar_Click(object sender, EventArgs e)
{
    FrmRapor frm = new FrmRapor();
    frm.Show();
}
```

Ve formumuzu çalıştıralım.

KitapId	Kitap Ad	Yazar	Sayfa	Fiyat	Yayınevi	Tür
1	Beyaz Zambaklar Ülkesinde	Grişey Petrov	245		8 Güneş	2
8	Eşejin Gölgesi	Halhan Taner	174		13 Yıldız	3
4	Suç ve Ceza	Fyodor Dostoyevski	985		18 Mars	2
5	Kirek Mantolu Madonna	Sabahattin Ali	203		11 Güneş	2
7	Denemeler	Monjaige	285		14 Yıldız	1

Böylece raporlamamızı bitirmiş olduk. Bize hazır olarak sunulan üst menüden arama yapabilir, ya da aşağıdaki resimde görüldüğü gibi raporumuzu çeşitli formatlarda kaydedebiliriz.



Sizler raporlarınıza daha fazla alan ekleyerek ve daha büyük tablolar üzerinde çalışarak pratiğinizi arttırabilirsiniz. Böylece bölümümüzün sonuna geldik.

## NELER ÖĞRENDİK?

Bu bölümde raporlama kavramını, raporlamanın neden bu kadar önemli olduğunu, raporlamanın nasıl yapılacağını, Report Viewer aracımızı ve çalışma mantığını, raporlama sihirbazını ve kullanımını, raporların farklı formatlarda nasıl çıktı alınacağını, raporlama aracımız üzerinde tasarım biçimlendirmelerinin nasıl yapılacağını öğrenmiş olduk.

# 17

## EVENTS VE DELEGATE

### BU BÖLÜMDE

Delegate	304
Events (Alt Başlık)	307
Neler Öğrendik?	310

Bu bölümde events ve delegate kavramını, delege yani temsilcilerin nasıl oluşturulduğunu, metotlarla temsilcilerin ilişkisini, temsilci parametreleri yapısını, olayları, olaylara neden ihtiyaç duyduğumuzu, Button, TextBox gibi araçların bazı olaylarını kullanmayı öğreneceğiz.

## DELEGATE

Temsilciler C# programlama dilinde bir veya birden fazla metodu gösteren referans türünden nesnelerdir. Metot tanımlamaları ile aynı şekilde yapılırlar. Bir temsilci yani delegate tanımlandıktan sonra bir veya birden fazla metot, temsilciye atanarak kullanılır. Temsilcilerin tam olarak ne işe yaradığını görebilmek için birden fazla aynı formatta metot tanımlaması yapmamız gerekmektedir. Ne işe yarar sorusunun cevabı olarak metotların adresini hafızada tutup istenildiği zaman çağıştırabilir diyebiliriz. Pratik bir uygulama üzerinde görelim.

### Uygulama 1

İlk olarak aritmetik 4 işlem yapan 4 farklı metot oluşturup bunları form içerisinde butonumuza tıkladığımız zaman çağıralım.

---

```
void toplam(int s1, int s2)
{
    listBox1.Items.Add(s1 + s2);
}

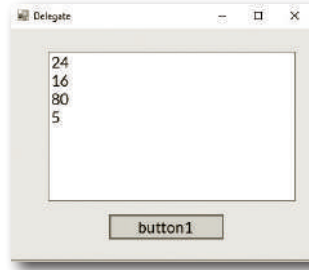
void fark(int x1, int x2)
{
    listBox1.Items.Add(x1 - x2);
}

void carpim(int a1, int a2)
{
    listBox1.Items.Add(a1 * a2);
}

void bolum(int b1, int b2)
{
    listBox1.Items.Add(b1 / b2);
}

private void button1_Click(object sender, EventArgs e)
{
    toplam(20, 4);
    fark(20, 4);
    carpim(20, 4);
    bolum(20, 4);
}
```

---



Global alanda toplam, fark, çarpım ve bölüm olmak üzere 4 farklı metod oluşturduk. Metodlarımız işlemleri ve yazdırmayı kendi içerisinde yapacağı için yazdırma komutunu da metodlarımızın içerisinde tanımladık. ListBox aracımızı içinde direkt olarak aritmetik işlemi de yaptığımız için metodlarımızı void türünde tanımladık. Daha sonra butonumuzun click olayı içerisinde metodlarımızı çağırıp içerisine 20 ve 4 değerlerini vererek 4 farklı metodumuzu çalıştırmış olduk. Peki, delegate bunun neresinde dersiniz şimdi 2. örneğimizde delegate başlığını göreceğiz.

## Uygulama 2

Bir önceki örneğimizde bu kez işlem isminde bir delegate yani temsilci oluşturup uygulamamızı yapalım.

```
delegate void islemler(int p1, int p2);

void toplam(int s1, int s2)
{
    listBox1.Items.Add(s1 + s2);
}

void fark(int x1, int x2)
{
    listBox1.Items.Add(x1 - x2);
}

void carpim(int a1, int a2)
{
    listBox1.Items.Add(a1 * a2);
}

void bolum(int b1, int b2)
{
    listBox1.Items.Add(b1 / b2);
}
```

# 19

## KOLEKSİYONLAR

### BU BÖLÜMDE

Koleksiyon Sınıfları	324
Neler Öğrendik?	331

Bu bölümde koleksiyon sınıfını, koleksiyon yapısının ne olduğunu, listeleri, ArrayList, List, SortedList, HashTable, Kuyruk ve Stack yapılarını, FIFO ve LIFO kavramlarını system.collections kütüphanesini pratik örnekler üzerinde öğreneceğiz.



## KOLEKSİYON SINIFLARI

Kitabımızın ilk bölümlerinde yer alan ve programlama dünyasında neredeyse tüm programlama dilleri için çok büyük bir önem arz eden diziler ya da programcılık literatürü ile array yapısı sayesinde çok sayıda aynı türde veri tanımlanıp çağrılarak üzerinde işlemler yapılabilir. Diziler bu yönüyle oldukça kullanışlı bir formattır fakat ciddi iki sorunla bizi karşı karşıya getiriyor. İlk dizilerde veri tiplerinin aynı olması zorunluluğu vardır. Bir dizi tanımlanırken int olarak tanımlandıysa tüm veriler tam sayı formatında olmak zorundadır. İkincisi dizinin boyutu önceden belli olmak zorundadır. Peki, biz farklı türde verileri bir arada kullanmak istersek ya da boyutu önceden belli olmayan verileri ele almak istediğimizde ne yapmalıyız? İşte bu noktada devreye koleksiyon veri tipleri giriyor. İlk olarak ön tanımlı koleksiyonları inceleyelim.

**System.Collections** kütüphanesi başlığında temel koleksiyon türleri bulunur. Verileri saklamak ve üzerinde işlem yapmak için kullanılan koleksiyonlar genel koleksiyonlarda olduğu gibi farklı türde verileri üzerinde barındırması ve boyutunun önceden belirlenme zorunluluğu olmaması dolayısıyla dizilerden bir adım daha öndedir. Bazı koleksiyonlarımız şunlardır;

- » Arraylist
- » SortedList
- » HashTable
- » Kuyruk
- » Stack
- » List

### ARRAYLIST

Farklı türde verileri bünyesinde varındırıp boyutu da istenildiği gibi büyüyüp küçültülebilen koleksiyon yapısıdır. Dizilere çok benzer. Dizilerle koleksiyon sınıfları arasında belirtmiş olduğumuz farklar ArrayList için de geçerlidir.

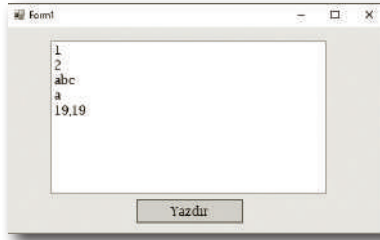
#### NOT

ArrayList komutunu kullanabilmek için System.Collections kütüphanesinin eklenmesi gerekmektedir.

**Uygulama 1**

5 elemanlı bir arraylist oluşturup bu listeyi ListBox aracımıza yazdıralım.

```
private void button1_Click(object sender, EventArgs e)
{
    ArrayList liste = new ArrayList();
    liste.Add(1);
    liste.Add(2);
    liste.Add("abc");
    liste.Add('a');
    liste.Add(19.19);
    foreach (var lst in liste)
    {
        listBox1.Items.Add(lst);
    }
}
```



ArrayList sınıfımızdan `liste` isminde bir nesne türeterek kodlarımızı yazmaya başladık. Liste nesnemize `add` metodu ile eklemeler gerçekleştirdik. Sırasıyla `int`, `int`, `string`, `char`, ve `double` türlerinde veri tanımlamaları yaptık. Böylece dizilerde olduğu gibi tek tip veri kullanmak zorunda kalmadık. Ayrıca `add` metodu ile ekleme yapmadan önce herhangi bir boyut belirleme gereği de duymadık. İstedığımız kadar değeri bir boyut sınırına takılmadan ekleyebiliriz. Eklemeleri yaptıktan sonra dizilerde olduğu gibi elemanların tümünü listelemek için `foreach` döngüsü kullandık.

**SORTED LIST**

Anahtar ve değer olmak üzere iki türlü parametre alan Sorted List yani sıralı listeler genellikle verilerin sıralanması için kullanılır. Sıralama işlemi anahtar değerine göre yapılır. Anahtar ve değer parametreleri birbirinden virgülle ayrılır. Pratik bir örnek üzerinde görelim.

# 20

## ÖĞRENCİ NOT KAYIT SİSTEMİ

### BU BÖLÜMDE

Proje Amacı	334
Veritabanı ve Tabloların Oluşturulması	334
Tetikleyicilerin Oluşturulması	339
Prosedürlerin Oluşturulması	342
Giriş Formunun Tasarlanması	343
Bağlantı Sınıfının Oluşturulması ve Giriş Formu Kodları	346
Öğretmen Formu Tasarımı	355
Duyuru İşlemleri Oluşturma Formu	373
Duyuru Listesi Formu	380
Mesajlar Formu	383
Öğrenci Formu	391
Neler Öğrendik?	400
Son Söz	401
Dizin	402

Bu bölümde C# form ile ADO.NET çerçevesinde temel SQL sorguları olan Ekleme, Silme, Güncelleme ve Listeleme başlıklarını bu 4 temel başlığa ek olarak prosedür, tetikleyici, alt sorgu ve birleştirme işlemlerini bir proje altında birleştireceğiz.

Tamamıyla SQL alt yapısında gerçekleştireceğimiz projemizde kullanıcılar arası mesajlaşma, sisteme giriş yapan kullanıcının bilgilerini getirme işlemini yani biraz daha terim biçiminde ifade edecek olursak “User Login Management” (Kullanıcı Giriş Yönetimi) sistemini, duyuru oluşturmayı, alınan sınav notlarına göre öğrencinin durumunu SQL üzerinde değiştirme gibi pek çok farklı başlığı modüler ve geliştirmeye çok müsait bir proje çerçevesinde öğreneceğiz.

## PROJE AMACI

İlk olarak projemizde neler yapacağımızla başlamak istedim. Bu projemizde amacımız kitabımızın 19 bölümünün tamamında kullandığımız başlıkları derleyip harmanlayarak ortaya somut ve daha kapsamlı ayrıca geliştirmeye çok müsait bir veritabanı uygulaması koyabilmektir. Projemiz içerisinde kişilerin kendi verilerini görebilecekleri, mesajlaşabilecekleri, profil bilgilerini düzenleyebileceği alanlara da yer vermek istedim. Bunu beraber kolaylıkla yapabiliriz. Amacımız öğrencilerin tıpkı e-okul gibi kendi numara ve şifreleri ile sisteme giriş yaptıkları zaman notlarını görüntülediği Fotoğraf, Ad, Soyad, Şifre gibi bilgilerini düzenleyebildikleri, diğer öğrencilere ve öğretmenlere mesaj atabildikleri bir sistem oluşturmak. Projemizde tablo olarak;

- » Öğrenci Bilgileri Tablosu
- » Öğretmen Bilgileri Tablosu
- » Sınav Notları Tablosu
- » Mesajlar Tablosu
- » Duyurular Tablosu

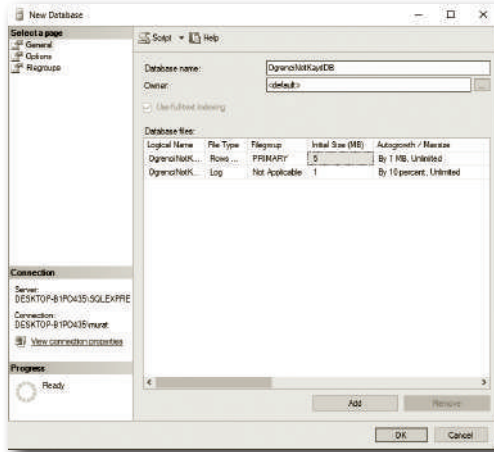
Olmak üzere 5 temel tablo kullanacağız. Form ara yüzü olarak da;

- » Giriş Formu
- » Öğrenci Kayıt Formu
- » Öğrenci Bilgi Detayları Formu
- » Öğretmen Bilgi Detayları Formu
- » Öğrenci Listesi Formu
- » Mesajlar Formu
- » Duyurular Formu

Olmak üzere 7 temel form kullanacağız. Dilerseniz proje bittikten sonra yeni tablo ve veritabanları ile içeriğinizi zenginleştirebilirsiniz.

## VERİTABANI VE TABLOLARIN OLUŞTURULMASI

SQL Server'ı açalım ve yeni bir veritabanı oluşturarak işlemlerimize başlayalım. Projemizin ve veritabanımızın isimlerini birbirine paralel olarak adlandıralım. Veritabanı adı olarak **OğrenciNotKayıtDB** verebiliriz.



Şimdi ilk tablomuz olan öğrenci bilgilerini tutacağımız tablomuzu oluşturarak başlayalım. Sırasıyla tablolarımızın alanlarını yazalım içeriğini detaylandıralım.

### Öğrenci Tablosu

- » ID
- » Ad
- » Soyad
- » Numara
- » Şifre
- » Fotoğraf

Column Name	Data Type	Allow Nulls
ID	smallint	<input type="checkbox"/>
AD	varchar(20)	<input checked="" type="checkbox"/>
SOYAD	varchar(20)	<input checked="" type="checkbox"/>
NUMARA	char(4)	<input checked="" type="checkbox"/>
SIFRE	varchar(10)	<input checked="" type="checkbox"/>
FOTOGRAF	varchar(100)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

ID alanımızı notlar tablosunda ki değerlerle ilişkilendireceğimiz için birincil anahtar olarak belirledik. Ayrıca yine ID alanımızın değerini otomatik artan yaptık. Otomatik artan işlemini önceki bölümlerde anlatmıştık. Numara için char(4) ifadesi kullanmamızın sebebi öğrencilerimizin numarasının 4 karakterden oluşacak olmasındadır. Siz derseniz daha geniş kapsamlı bir proje için daha geniş aralıklar kullanabilirsiniz. Şifre alanını da en fazla 10 karakter olacak

şekilde ayarladık. Fotoğraf kısmı için Image kullanmadık. Çünkü Image veri tipi veritabanı boyutunu çok fazla arttırdığından bunun yerine resim yolunu tutacağımız bir string ifade kullanmanın daha optimal bir çözüm olacağını düşündük. Tablomuzun ismini **TbOgrenci** olarak belirleyip kaydederek ilk tablomuzun işlemlerini tamamlamış olduk.

### Öğretmen Tablosu

- » ID » Numara
- » Ad » Şifre
- » Soyad

Column Name	Data Type	Allow Nulls
ID	tinyint	<input type="checkbox"/>
AD	varchar(20)	<input checked="" type="checkbox"/>
SOYAD	varchar(20)	<input checked="" type="checkbox"/>
NUMARA	char(4)	<input checked="" type="checkbox"/>
SIFRE	varchar(5)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

5 temel alanımız var. Tıpkı öğrenci tablosunda olduğu gibi her öğretmenin de birer ID değeri, Adı, Soyadı, Numarası ve Şifresi bulunacak. Ancak fotoğrafa ihtiyaç duymadık. Çünkü fotoğraf alanı öğretmenin öğrenciyi hatırlaması için mevcuttur. Numarayı yine 4 karakter olarak belirleyip şifreyi bu kez daha kısa tuttuk. Öğretmen ve öğrenci tablolarına girişler farklı şekilde olacak. Bu detayları projemizin form kısmına geçtiğimiz zaman göreceğiz.

### Notlar Tablosu

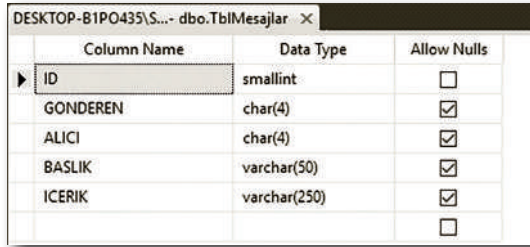
- » ID » Proje
- » Sınav1 » Ortalama
- » Sınav2 » Durum
- » Sınav3

Column Name	Data Type	Allow Nulls
OGRID	smallint	<input type="checkbox"/>
SINAV1	tinyint	<input checked="" type="checkbox"/>
SINAV2	tinyint	<input checked="" type="checkbox"/>
SINAV3	tinyint	<input checked="" type="checkbox"/>
PROJE	tinyint	<input checked="" type="checkbox"/>
ORTALAMA	decimal(18, 2)	<input checked="" type="checkbox"/>
DURUM	bit	<input checked="" type="checkbox"/>

Notlar tablomuzda ID alanı dışında Sınav notlarını gireceğimiz 1, 2 ve 3. sınavlar, proje notunu gireceğimiz proje sütunu, notların toplanıp bölünmesiyle hesaplanacak ortalama değeri ve öğrencinin geçti mi, yoksa kaldı mı? sonucunun yazılacağı durum sütunları bulunmaktadır. Sınav notları 0-100 arasında birer tam sayıdan oluşacağı için **tinyint** veri tipini kullandık. Ortalama değerleri ondalıklı çıkabileceğinden **decimal** veri tipini virgülden sonra en fazla iki basamak alacak şekilde tercih ettik. Durumun iki sonucu olduğundan bu veri tipi için de bit kullanarak tablomuzu oluşturduk.

### Mesajlar Tablosu

- » ID
- » Gönderen
- » Alıcı
- » Başlık
- » İçerik

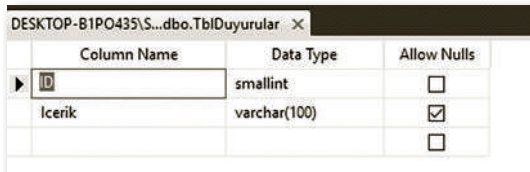


Column Name	Data Type	Allow Nulls
ID	smallint	<input type="checkbox"/>
GONDEREN	char(4)	<input checked="" type="checkbox"/>
ALICI	char(4)	<input checked="" type="checkbox"/>
BASLIK	varchar(50)	<input checked="" type="checkbox"/>
ICERIK	varchar(250)	<input checked="" type="checkbox"/>

Her mesajın bir ID numarası olsun istediğimizden ilk olarak ID sütununu ekledik ve sütun özellikleri penceresinden bu alanımızı otomatik artan yaptık. Gönderici ve alıcı değerleri numaraya göre yapılacağı için bu başlıkları **char(4)** olarak belirledik. Mesajımızın bir başlığı ya da konusu olsun istediğimiz için başlık son olarak da mesajın içeriğinin tutulacağı içerik sütunumuzu da ekleyerek tablomuzu tamamladık.

### Duyurular Tablosu

- » Duyuru ID
- » Duyuru İçerik



Column Name	Data Type	Allow Nulls
ID	smallint	<input type="checkbox"/>
icerik	varchar(100)	<input checked="" type="checkbox"/>

## MURAT YÜCEDAĞ

---

*muratyucedag.wordpress.com*



*facebook.com/murattyucedag*



*twitter.com/murattyucedag*



*linkedin.com/in/murat-yücedağ-186933149*



*udemy.com/user/murat-yucedag-3*



*youtube.com/user/YazilimHerYerde*



*github.com/MuratYucedag/testproje*



*yucedagmurat23@gmail.com*

