

# **DATA STRUCTURES**

Part – III

Sınıflar, Nesneler ve  
İlgili Programlama Örnekleri

# Sınıf (Class) ve Nesne (Object)

- Sınıflar, verileri (değişkenler, ...) ve onlar üzerinde işlem yapacak kodu (metotları,...) bir çatı altında toplayarak bütünleştirirler.
- Metodları da içeren daha geniş kod blokları oluşturulmasını sağlarlar.
- Nesneleri oluşturmak için bir sınıf şablonu kullanılmaktadır. Bir nesne, oluşturulduğu sınıfın bir örneğidir.
- Sınıflar, nesne yönelimli yazılım geliştirmenin temel yapıtaşlarındandır.

# Örnek 1

## Sınıf Oluşturma ve Kullanımı

```
class Öğrenci
{
    public int numara;
    public string ad, soyad;
}
```

Öğrenci sınıfı (ve aynı zamanda yeni bir veri tipi) oluşturuluyor. Öğrencinin numarasını, ad ve soyadını tutacak değişkenler tanımlanıyor. Bu sınıf metot içermiyor.

```
class Program
{
    static void Main(string[] args)
    {
        int i = 5;
        Öğrenci ogr = new Öğrenci();
        ogr.ad = "Ali"; ogr.soyad = "Yılmaz"; ogr.numara = 14;
    }
}
```

Nesne Oluşturuluyor

Ana metot içerisinde Öğrenci sınıfı kullanılıyor. Değişken tanımlamadan önemli bir farkı, new ile bellekte, Öğrenci sınıfından oluşacak nesneler için yer ayrılması gerekliliği. Öğrenci sınıfından ogr nesnesini oluşturduk.

Sınıf üyelerine, sınıf dışından erişim . ile yapılır.

# Örnek 2

## I - Sınıfa Metot Ekleme ve Çağırma

```
class Öğrenci
{
    public int numara;
    public string ad, soyad;
    public void yazdır()
    { Console.WriteLine(numara+" "+ad+" "+soyad); }
}
```

Öğrenci sınıfına yazdır metodunu ekledik. Sınıf içerisindeki değişkenlerin değerlerini ekrana yazdırıyor.

```
class Program
{
    static void Main(string[] args)
    {
        Öğrenci ogr = new Öğrenci();
        ogr.numara = 14; ogr.ad = "Ali"; ogr.soyad = "Yılmaz";
        ogr.yazdır();
    }
}
```

ogr nesnesinin yazdır metodunu, ana metottan çağırıyoruz.

Ekran Çıktısı :  
14 Ali Yılmaz

# Örnek 2

## II - Sınıfa Üyelerine Dışarıdan Erişim

```
class Öğrenci
{
    public int numara;
    public string ad, soyad;
    public void yazdır()
    { Console.WriteLine(numara+" "+ad+" "+soyad); }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Öğrenci ogr = new Öğrenci();
        ogr.numara = 14; ogr.ad = "Ali"; ogr.soyad = "Yılmaz";
        Console.WriteLine(ogr.numara+" "+ogr.ad + " " +
ogr.soyad + " ");
    }
}
```

Yazdır metodu yerine elle yazılabilir. numara, ad ve soyad, **public** tanımlı oldukları için sınıf dışından erişilebilir. Ancak, bu tür bir kullanım yerine, Öğrenci sınıfına metot eklenmesi önerilir.

# Örnek 2

## III - Dışarıdan Erişimin Engellenmesi

```
class Öğrenci
{
    private int numara;
    public string ad, soyad;
    public void yazdır()
    { Console.WriteLine(numara+" "+ad+" "+soyad); }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Öğrenci ogr = new Öğrenci();
        ogr.numara = 14; ogr.ad = "Ali"; ogr.soyad = "Yılmaz";
        Console.WriteLine(ogr.numara+" "+ogr.ad + " " +
ogr.soyad + " ");
    }
}
```

numara değişkenini **private** tanımlarsak, sınıf dışından erişilemez.  
*Error1'ConsoleApplication11.Öğrenci.numara' is inaccessible due to its protection level* şeklinde 2 adet hata mesajı verir. ad ve soyad değişkenlerinin kullanımında ise sorun çıkmaz.

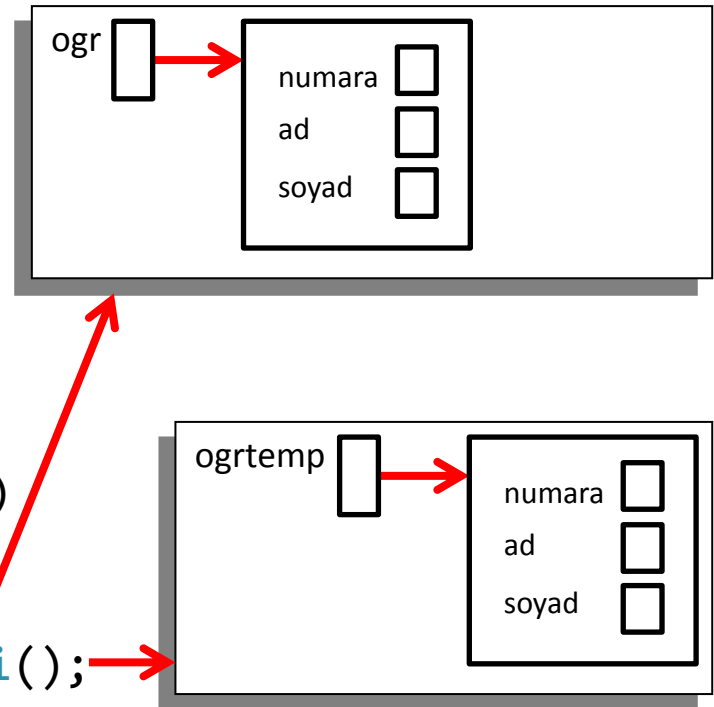
# Örnek 3

## Öğrenci Sınıfından Nesne Oluşturma

```
class Öğrenci
{
    private int numara;
    public string ad, soyad;
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Öğrenci ogr = new Öğrenci();
        Öğrenci ogrtemp = new Öğrenci();
    }
}
```

ogr ve ogrtemp olmak üzere, iki adet nesne oluşturduk.




# Örnek 4

## Bellekte Yer Ayrılmazsa!

```
class Öğrenci
{
    private int numara;
    public string ad, soyad;
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Öğrenci ogr;
        ogr.ad = "Ali";
    }
}
```

ogr   
New ile bellekte yer ayrılmadı!

### Hata Mesajı Verir:

Error1 Use of unassigned local variable 'ogr'  
ogr referansı null içerip bir adresi göstermez. ad sahası için bellekte yer de ayrılmamıştır.



# Örnek 5

## Nesne Dizisi Oluşturma

```
class Program
```

```
{
```

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

```
    {
```

```
        Öğrenci[] öğrenciler = new Öğrenci[10];
```

```
        for (int i = 0; i < öğrenciler.Length; ++i)
```

```
            öğrenciler[i] = new Öğrenci();
```

```
        öğrenciler[3].ad = "Kemal";
```

```
    }
```

```
}
```

öğrenciler



10 adet Öğrenci referansı için yer ayrıldı.

Her bir öğrenci için bellekte yer ayrılması gerekiyor.

öğrenciler



numara

ad

soyad



Kemal

# Örnek 6

## Yapılandırıcı (Metot) - Constructor

```
class Öğrenci
{
    private int numara;
    public string ad, soyad;

    public Öğrenci() { }

    public Öğrenci(string ad1, string soyad1, int no)
    { ad = ad1; soyad = soyad1; numara = no; }

    public void yazdır()
    { Console.WriteLine(numara+" "+ad+" "+soyad); }
}
```

Yapılandırıcı Metot, sınıfla aynı ismi taşır. Bu örnekte, 2 adet constructor (yapılandırıcı) tanımlanmıştır.  
1.'si parametre almamaktadır.  
2.'si 3 adet parametre içermektedir.

```
class Program
{
    static void Main(string[] args)
    {
        Öğrenci ogr1 = new Öğrenci();
        Öğrenci ogr2 = new Öğrenci("Ali", "Yılmaz", 11);
        ogr1.yazdır(); ogr2.yazdır();
    }
}
```

Ekran Çıktısı :  
0  
11 Ali Yılmaz

Yapılandırıcı Metot, ilgili sınıftan bir nesne oluşturulduğunda devreye girer. Örnekte, parametre uyumluluğu dolayısı ile,

ogr1 nesnesi oluşturulurken 1. yapılandırıcı otomatik çağrılır.

ogr2 nesnesi oluşturulurken 2. yapılandırıcı otomatik çağrılır.

2. yapılandırıcı, nesneye doğrudan değer aktarmayı sağlayarak işlemleri kolaylaştırır.

# Nesneye Yönelik Programlama

## Temel Bilgi ve Terminoloji - I

**Sınıf (Class) :** Soyut bir veri tipinin hem verilen tiplerdeki veriler kümesini, hem de bu değerler üzerinde yapılabilecek işlemler kümesini bir araya getirir.

Örnek : "Öğrenci" sınıfı.

**Nesne (Object) :** Sınıfın örneğine (sınıf tipindeki değişkenlere) nesne adı verilir.

Örnek : "ogr1" ve "ogr2" nesneleri.

**Metot (Method) :** Bir eylemi veya işlemi gerçekleştiren sınıf üyesidir.

"Öğrenci()" yapılandırıcıları ve "yazdır()" metodu Öğrenci sınıfının metotlarıdır.

**Sınıf Üyeleri (Class Members) :** Sınıfın elemanlarına üye adı verilir.

Değişkenler, metotlar ...

Örnekler : "ad", "soyad", "numara" değişkenleri; "Öğrenci()" yapılandırıcıları ve "yazdir()" metodu Öğrenci sınıfının üyeleridir.

# Nesneye Yönelik Programlama

## Temel Bilgi ve Terminoloji - II

**Yapılandırıcı metot (Constructor) :** Sınıftan yeni bir nesne oluşturulduğu anda çağrılan metoda yapılandırıcı adı verilir. Yapılandırıcı metot ismi, sınıf ismi ile aynıdır. Sınıfların bu kısmında, ilk değerlerin atanması ve kullanılacak veri yapılarının tanımlanması gibi işlemlere ilişkin kodlar yazılır.

```
Öğrenci ogr2 = new Öğrenci("Ali", "Yılmaz", 11);
```

ogr2 nesnesi, new deyimi ile oluşturulurken, Öğrenci sınıfının sırayla iki tane string ve bir tane tamsayı alan yapılandırıcı metodu devreye girer.

```
// Yapılandırıcı metot
```

```
public Öğrenci(string ad1, string soyad1, int no)
{ ad = ad1; soyad = soyad1; numara = no; }
```

# Nesneye Yönelik Programlama

## Temel Bilgi ve Terminoloji - III

**Çokbiçimlilik (polymorphism)** : Bir metod ismi bulunmakla birlikte, farklı şekillerde tanımlanıp çağrılabilme özelliği bir tür çokbiçimliliktir:

```
public Öğrenci() { }
```

```
public Öğrenci(string adı, string soyadı, int no)  
{ ad = adı; soyad = soyadı; numara = no; }
```

Öğrenci yapılandırıcısının farklı parametrelere sahip iki farklı biçimi.

int, float, double sayılar için tanımlanmış matematiksel fonksiyonlar (topla gibi) da benzeri şekilde örnek olarak verilebilir.

# Örnek 7

## this referansı

**Sınıf değişkenleri**

```
class Öğrenci
{
    private int numara;
    public string ad, soyad;

    public Öğrenci() { }

    public Öğrenci(string ad, string soyad, int no)
    { this.ad = ad; this.soyad = soyad; numara = no; }

    public void yazdır()
    { Console.WriteLine(numara+" "+ad+" "+soyad); }
}
```

2. yapılandırıcıda önceki örneğe göre değişiklik yaptık. Önceki örnekte, parametre bölümündeki değişkenleri tanımlarken, karışıklık yani isim çakışması olmaması için sınıf değişkeni olanlarla farklı isimler kullanmıştık. Bu örnekte ise ad ve soyad parametreleri Sınıf değişkenleri ile aynı ismi taşıyor. this referansı, sınıf değişkenlerine erişmeyi sağlar.

# Örnek 8 – Part I : Öğrenci Sınıfı

## private üyelere dışarıdan erişim

```
class Öğrenci
```

```
{  
    private int numara;  
    private string ad, soyad;
```

Sınıfın tüm değişkenlerini,  
private olarak tanımladık  
(yetkisiz erişimleri engellemek  
için)

```
    public Öğrenci(string ad, string soyad, int no)  
    { this.ad = ad; this.soyad = soyad; numara = no; }
```

```
    public string adAl()  
    { return ad; }
```

adAl metodu ile, nesnenin ad  
sahasını dışarıdan okuyabiliyoruz.

```
    public void adAta(string ad1)  
    { ad = ad1; }
```

adAta metodu ile, nesnenin ad  
sahasına değer atayabiliyoruz.

```
    public void yazdır()  
    { Console.WriteLine(numara+" "+ad+" "+soyad); }
```

```
}
```

# Örnek 8 – Part II : Ana Metot

## private üyelere dışarıdan erişim

```
class Program
{
    static void Main(string[] args)
    {
        Öğrenci ogr = new Öğrenci("Ali", "Yılmaz", 11);
        ogr.yazdır();
        // ogr.ad = "Ali"; inaccessible
        ogr.adAta("Veli");
        ogr.yazdır();
        Console.WriteLine(ogr.adAl());

        Console.ReadKey();
    }
}
```

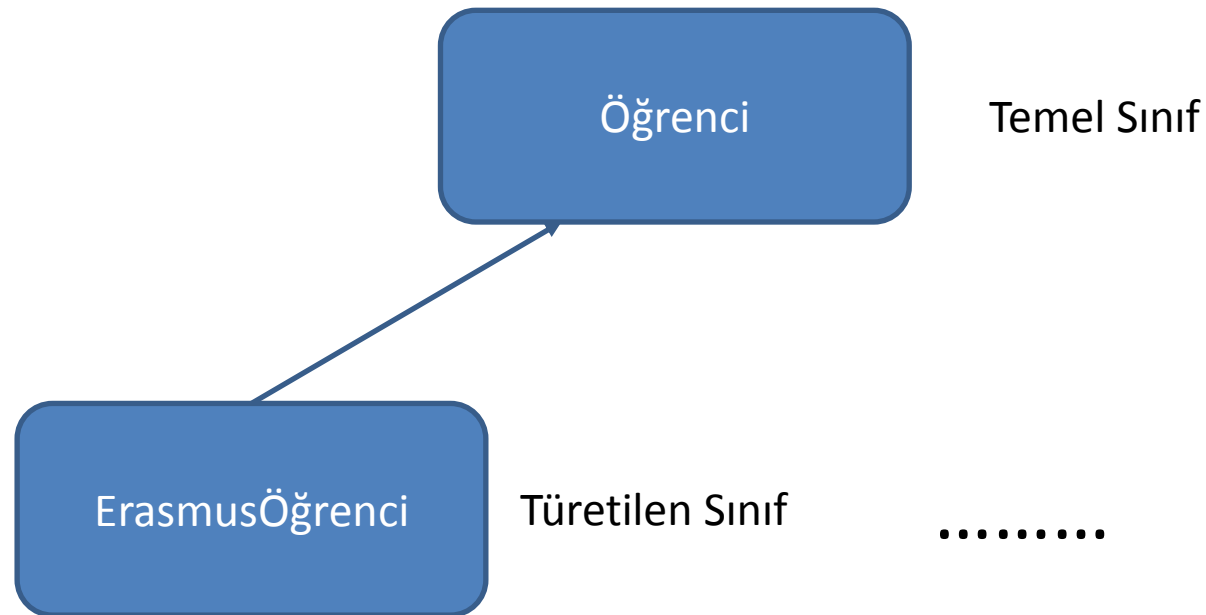
Ekran Çıktısı:  
11 Ali Yılmaz  
11 Veli Yılmaz  
Veli



# Kalıtım (Inheritance)

- Yazılım yeniden kullanılabilirliğinin bir çeşididir.
- Diğer sınıfların özelliklerini devralmasını sağlar.
- Temel sınıftan türetilen bir sınıf için, aynı olan değişken ve metotları yeniden yazmaya gerek kalmaz.
- Farklılıklar üzerine yoğunlaşılabilir.
- Kaynak kod yazım ve hata ayıklama sürelerini de azaltır.

# Temel Sınıf (Base Class) ve Türetilen Sınıf (Derived Class)



# Örnek 9

## Kalıtım – I : Temel Sınıf

```
class Öğrenci
{
    public int numara;
    public string ad, soyad;

    public Öğrenci() { }

    public Öğrenci(string ad1, string soyad1, int no)
    { ad = ad1; soyad = soyad1; numara = no; }

    public void yazdır()
    { Console.WriteLine(numara + " " + ad + " " + soyad); }
}
```

# Örnek 9

## Kalıtım – II : Türetilmiş Sınıf

```
class ErasmusÖğrenci: Öğrenci
{
    public string country;

    public void yazdır()
    { base.yazdır(); Console.WriteLine(country); }
}

class Program
{ static void Main(string[] args)
    { ErasmusÖğrenci ogr = new ErasmusÖğrenci();
      ogr.ad = "John"; ogr.soyad = "Doe";
      ogr.numara = 25; ogr.country = "England"; ogr.yazdır();
    }
}
```

Ekran Çıktısı:  
25 John Doe  
England

# **protected** erişim belirleyicisi (**protected** access modifier)

- Kalıtım içeren durumlarda kullanılır.
- Türetilmiş alt sınıflar tarafından erişilebilmesi hariç private'dır.
- Protected belirleyicisinin kullanıldığı metot veya değişkene tanımlandığı sınıf içerisinden veya bu sınıftan türetilmiş sınıflardan erişilebilir.

# Örnek 10

## Özellikler (Properties) : Getter, Setter

Bir alanı bu alana erişen metot ile birleştirirler. Değişkenler üzerindeki kontrolü artırırılar. Get ve set metotlarına da gerek kalmaz.

```
class Maaş
{
    private int maaş;

    public int maaşÖzellik
    {
        get { return maaş; }
        set
        {
            if (value < 500)
                maaş = 500;
        }
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Maaş m = new Maaş();

        m.maaşÖzellik = 350;

        Console.WriteLine(m.maaşÖzellik);
    }
}
```

# İndeksleyiciler (Indexers)

- Diziler `System.Array` sınıfı türünden birer nesnedirler. `System.Array` sınıfının üyelerinden biri olan `Length` özelliği dizinin eleman sayısını tutar.
- İndeksleyiciler, bir nesnenin bir dizi gibi indekslenebileceği bir mekanizma sağlarlar.

# Örnek 11

## İndeksleyiciler (Indexers)

```
class Öğrler
{
    public string ad;
    public string this[int indeks]
    {
        get
        {
            return ad;
        }
        set
        {
            ad = value;
        }
    }
}
```

```
class Program
{
    static void Main()
    {
        Öğrler ogr = new Öğrler();
        ogr[7] = "Kemal";

        Console.WriteLine(ogr[7]);
    }
}
```



# Değişen Sayıda Argüman Alabilen Metot Tanımlamak

```
static int topla(params int[] sayılar)
{
    int t=0;
    for (int i = 0; i < sayılar.Length; ++i)
        t+=sayılar[i];
    return t;
}
```

params niteleyicisi kullanılır

```
static void Main(string[] args)
{
    int toplam;

    toplam = topla(3,4,5); // 12
    toplam = topla(7, 8, 9, 15); // 39
    Console.WriteLine(toplam);
}
```

# **AD UZAYI**

## **NAMESPACE**

# Ad Uzayı (namespace)

## Ad Uzayı Tanımı:

```
namespace Öğrenci1
{
    class Öğrenci
    { }
}
```

# Ad Uzayı Kullanımı

**using Öğrenci1;**  
şeklinde kullanabiliriz.

Ad Uzayı mekanizması, isim çakışmalarını önler.  
Program ve Sınıfların farklı sürümlerinin  
oluşturulmasını sağlar.

...

# System Ad Uzayı

- System bir ad uzayıdır.
- Math, Console, Convert, String sınıfları, ... System ad uzayı altında yer alır.
- Kod içerisinde System. yazdığımızda System ad uzayı içerisindeki diğer elemanları da görürüz.
- System.String. yazdığımızda string sınıfına ilişkin hangi metotları kullanabileceğimizi görürüz.

# STRING SINIFI

STRING CLASS

# String Sınıfı Metotları

<http://www.c-sharpcorner.com>

Method	Meanings
ToUpper() ToLower() ToTitleCase()	Creates a copy of a given string in upper case or lower case or in title case.
Length()	Returns the length of the string.
Concat()	Returns a new string that is composed of two discrete strings.
Contains()	Determine if the current string object contains a specified string.
Compare() CompareTo() CompareOrdinal()	Compares two strings.
Copy()	Returns a fresh new copy of an existing string.
Format()	Formats string literal using other primitives like numerical data and other strings and the {0} notation we uses.
Insert()	Receives a copy of the current string that contains newly inserted string data.
PadLeft(), PadRight()	Returns copies of the current string that has been padded with specific data.
Remove(), Replace()	Receives copy of string with modifications.
Split()	Separates strings by a specified set of characters and places these strings into an array of strings.
Trim()	Trim method removes white spaces from the beginning and end of a string.
TrimStart()	TrimStart method removes characters specified in an array of characters from the beginning of a string.
TrimEnd()	TrimEnd method removes characters specified in an array of characters from the end of a string.
Substring()	Returns a string that represents a substring of the current string.
ToCharArray()	Returns a character array representing the current string.

# String Sınıfı

string yapılandırıcıları, özellikleri, metotlarına ve diğer ayrıntılarına :

<http://msdn.microsoft.com/en-us/library/system.string.aspx>

bağlantısından da ulaşabilirsiniz.



# String Karşılaştırma

```
string str1 = "abc";  
string str2 = "abz";  
Console.WriteLine(str1.CompareTo(str2)); // -1 döndürür.
```

```
string str1 = "abc";  
string str2 = "aba";  
Console.WriteLine(str1.CompareTo(str2)); // 1 döndürür.
```

```
string str1 = "abc";  
string str2 = "abc";  
Console.WriteLine(str1.CompareTo(str2)); // 0 döndürür.
```

# String Eşitlik Kontrolü

`if(str1 == str2) ...`

Java'da farklı : Referans değerlerini karşılaştırır.

Java'da İçerikleri karşılaştırmaz.

Equals ve `CompareTo==0` şeklinde de eşitlik kontrolü yapılabilir.

# String Split Metodu

Verilen string'i parçalarına ayırır:

```
string str1 = "abc 123 aaab derece";
```

```
String[] dizi = str1.Split();
```

dizi içerisine 4 adet string parçası olarak atar.

0. abc

1. 123

2. aaab

3. derece

<http://msdn.microsoft.com/en-us/library/ms228388.aspx>:

```
char[] delimiterChars = { ' ', ',', ':', '\t' };  
string text = "one\two three:four,five six seven";  
System.Console.WriteLine("Original text: '{0}'", text);  
string[] words = text.Split(delimiterChars);
```

# Strings are immutable

Strings are immutable--the contents of a string object cannot be changed after the object is created, although the syntax makes it appear as if you can do this. For example, when you write this code, the compiler actually creates a new string object to hold the new sequence of characters, and that new object is assigned to `b`. The string `"h"` is then eligible for garbage collection.

- `string b = "h"; b += "ello";`

# String parameter example and ref keyword

```
static void strMethod(string str)
{
    str = "Modified";
}
```

```
static void Main(string[] args)
{
    string str = "Original";
    Console.WriteLine(str);
    strMethod(str);
    Console.WriteLine(str);
    Console.ReadLine();
}
```

Output :  
Original  
Original

```
void strMethod(ref string str)
```

```
strMethod(ref str);
```

Output :  
Original  
Modified

See also out keyword