

Sınıf Üyeleri 3: Özellikler (Property)

Sınıf üyelerinden alanların bilinçsiz ya da kötü niyetli kullanımlarına karşı önlem almak gerekir. Benzeri şekilde veri bütünlüğü ve doğruluğu açısından alanların(field) sınıf dışına public olarak açılması önerilmemektedir. Örneğin integer tipinden tanımlanmış bir yas değişkenine negatif bir değer atanabilir. Ya da verilerin belirli bir değer aralığında girilmesi istenebilir. Bu noktada NDP, veri değerinin kabulünden sonra verinin şartları sağlayıp sağlamaması kontrolünden daha ziyade daha verinin kabulü esnasında verinin gerekli şartları sağlayıp sağlamadığının kontrolünü tavsiye eder. Şartlar sağlandığı takdirde kabul edilmesi aksi takdirde sistemin zarar görmesini engeller.

Bazen sınıflarda private bölgelere ulaşılması gerekmektedir. Birçok sınıfta sarmalama gereği bu alanlara ulaşılmaması gerekirken, kalıtım olmayan ya da paket içerisinde yer alamayan bir sınıfın bu değere bağlı işlemler yapıyor durumu ile karşılaşılmaktadır.

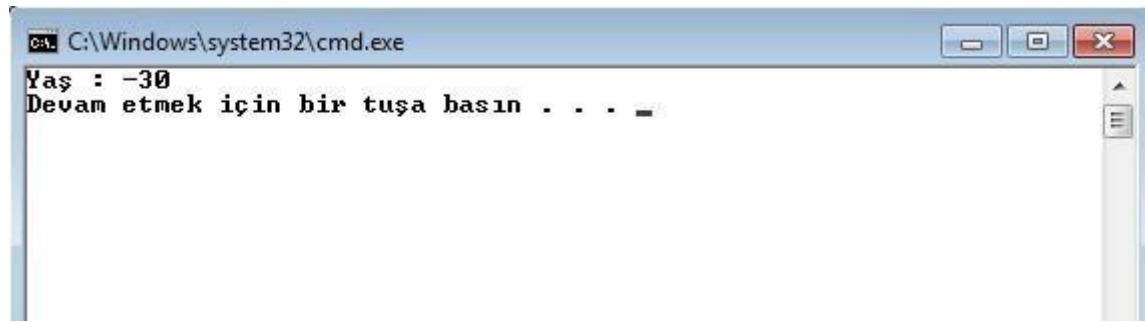
Programların bazı bölümlerinde bir sınıfa ait alanların sadece okunabilme olanağı ya da yalnızca yazılabilir okunamama özellikleri sağlamak gerektiğinde özellikleri kullanabiliriz.

Örnek 1: (Veri doğrulama)

```
class Ornek
{
    public int yas;
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Ornek o = new Ornek();
        o.yas = -30;

        Console.WriteLine("Yaş : {0}", o.yas);
    }
}
```



```
C:\Windows\system32\cmd.exe
Yaş : -30
Devam etmek için bir tuşa basın . . .
```

Burada büyük bir hata vardır. Yaş hiç bir zaman negatif bir değer olamaz. Ancak yas değişkeni sınıf içerisinde int olarak tanımlandığından negatif bir değer girilebilmektedir.

Örnek 2: (Veri bütünlüğü) Araçların trafiğe çıkış yılı baz alan bir banka 2-9 yaş araçlara 2. el kredisi vermektedir.

```
class Banka
{
    public DateTime trafigeCikisTarihi;

    public int TasitYasiHesapla()
    {
        return Convert.ToInt32(DateTime.Now.Year - trafigeCikisTarihi.Year);
    }
}
```

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

        Console.Write("Araç trafiğe çıkış tarihini giriniz :");
        ban.trafigeCikisTarihi = DateTime.Parse(Console.ReadLine());
        Console.WriteLine("Araç Yaşı: " + ban.TasitYasiHesapla());
    }
}
```

```
C:\Windows\system32\cmd.exe
Araçın trafiğe çıkış tarihini giriniz :20/10/2030
Araç Yaşı: -20
Devam etmek için bir tuşa basın . . . _
```

Örnek 1 deki gibi benzeri bir hata oluşmuş girilen tarih bugünün tarihinden büyük girilmiş ve hesaplamalar sonucu ekrana yazdırılmıştır. Oysaki burada bir kontrol yapılmalı bugünün tarihinden büyük değerler girildiğinde program hata mesajı üretmeliydi. Benzeri şekilde araç yaşı belirtilen 2-9 yaş aralığında ise kredinin verilebileceği aksi durumda ise kredinin verilemeyeceği şeklinde sonuçlara ulaşabilmeliydi.

2. örneğimizi aşağıdaki şekilde düzenleyelim.

```
class Banka
{
    private DateTime trafigeCikisTarihi=DateTime.Now;

    public DateTime GetTrafigeCikisTarihi()
    {
        return trafigeCikisTarihi;
    }

    public void SetTrafigeCikisTarihi(DateTime tarih)
    {
        if (tarih > DateTime.Now)
            Console.WriteLine("Hatalı Tarih Girişi");
        else
            trafigeCikisTarihi = tarih;
    }

    public int TasitYasiHesapla()
    {
        return Convert.ToInt32(DateTime.Now.Year - trafigeCikisTarihi.Year);
    }
}
```

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

```

{
    Banka ban = new Banka();

    Console.Write("Aracın trafiğe çıkış tarihini giriniz :");
    ban.SetTrafikeCikisTarihi(DateTime.Parse(Console.ReadLine()));
    Console.WriteLine("Araç Yaşı: " + ban.TasitYasiHesapla());
}
}

```

```

C:\Windows\system32\cmd.exe
Aracın trafiğe çıkış tarihini giriniz :20/10/2030
Hatalı Tarih Girişi
Araç Yaşı: 0
Devam etmek için bir tuşa basın . . .

```

6.2. C#'ta Özellikler (Property)

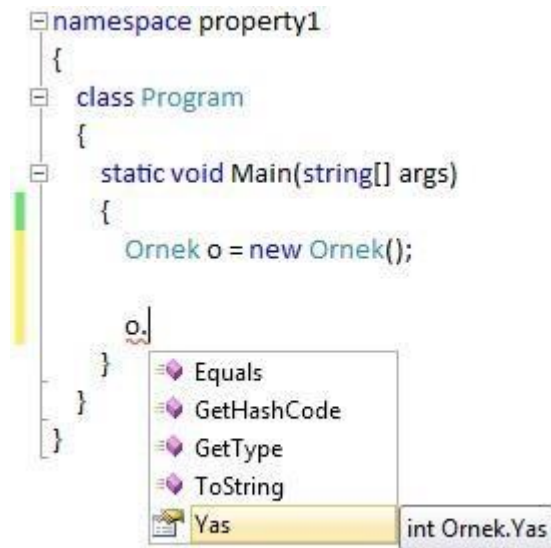
Veri doğrula, veri bütünlüğü ve public alanların güvenliği için C# özel yapıları sınıf üyesi olarak kabul etmiştir. Okunabilme (get) ve yazılabilme(set) yapıları özel bir blok içerisinde gösterilmektedir.

```

class Ornek
{
    private int yas;

    public int Yas
    {
        get // okunabilme özelliği
        {
            return yas;
        }
        set // değer atanabilme(yazılabilme özelliği)
        {
            yas = value;
        }
    }
}

```



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

        o.Yas = 30;

        Console.WriteLine(o.Yas);
    }
}
```

get bloğu içerisinde okuma, set bloğu içerisinde yazma işlemi yapılmaktadır. C# 3.0 ile gelen bir başka özellik ise default şekilde yazılan get-set bloğunun doldurulmasına gerek olmadan otomatik olarak tanımlanabilmesidir.

Burada private değişken tanımlamaya da gerek kalmamıştır.

```
class Yeni
{
    public string Adi
    {
        get;
        set;
    }
}
```

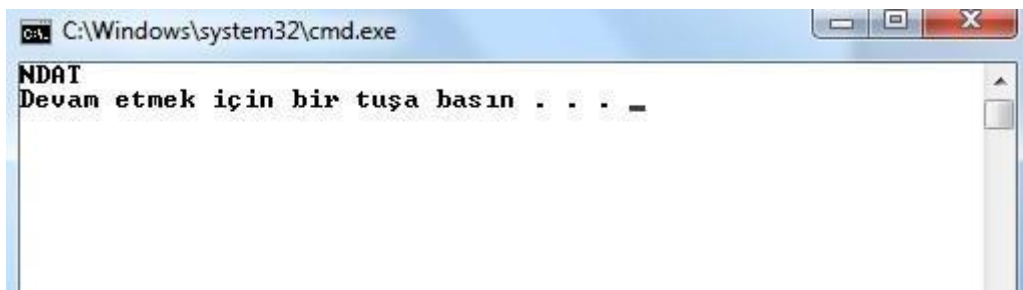
```

class Program
{
    static void Main(string[] args)
    {
        Yeni y = new Yeni();

        y.Adi = "NDAT";

        Console.WriteLine(y.Adi);
    }
}

```



6.2.1. Sadece Okunabilir Özellikler (Property)

NDP'de sarmalama amacıyla değerinin değiştirilmesi güvenlik açısından yasaklanmış sadece okunabilir alanlar oluşturulama istendiğinde get-set bloğunda sade get tarafı yazılmalıdır.

```

class getProperty
{
    private int yaricap = 20;

    public int Yaricap
    {
        get
        {
            return yaricap;
        }
    }
}

```

```

class Program
{
    static void Main(string[] args)
    {
        getProperty gp = new getProperty();
        gp.Yaricap = 100; // hata sadece değeri okunabilinir.

        Console.WriteLine(gp.Yaricap);
    }
}

```

Error List		
1 Error	0 Warnings	0 Messages
	Description	File
1	Property or indexer 'property1.getProperty.Yaricap' cannot be assigned to -- it is read only	Program.cs

6.2.1. Sadece Yazılabilir Özellikler (Property)

Değerinin değiştirilmesi sadece yazılabilir ancak okunamayabilir özellikler oluşturulmak istendiğinde get-set bloğunda sadece set tarafı yazılmalıdır.

```

class setProperty
{
    private int m_A;

    public int A
    {
        set
        {
            m_A = value;
        }
    }
}

```

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

        sp.A = 200;

        Console.WriteLine(sp.A); // hata sadece değeri atabilinir.
    }
}
```

Error List

1 Error 0 Warnings 0 Messages		
	Description	File
1	The property or indexer 'property1.SetProperty.A' cannot be used in this context because it lacks the get accessor	Program.cs

Örnek: Alan hesaplama programının devamı.

```
class Yamuk
{
    private double _taban;
    public double Taban
    {
        get
        {
            return _taban;
        }
        set
        {
            if(value < 0)
            {
                Console.WriteLine("Alan sıfırdan küçük olamaz...!");
                _taban = 0;
            }
            else
            {
                _taban = value;
            }
        }
    }

    private double _tavan;

    public double Tavan
    {
        get
        {
            return _tavan;
        }
        set
        {
            if(value < 0)
            {
                Console.WriteLine("Tavan 0 dan küçük olamaz...!");
                _tavan = 0;
            }
            else
            {
                _tavan = value;
            }
        }
    }
}
```

```

        private double _yukseklık;

        public double Yukseklik
        {
            get
            {
                return _yukseklık;
            }

            set
            {
                if(value<0)
                {
                    Console.WriteLine("Yükseklik sıfırdan küçük olamaz
...!");
                    _yukseklık = 0;
                }
                else
                    _yukseklık = value;
            }
        }

        public Yamuk(double taban, double tavan, double yukseklik)
        {
            this.Taban = taban;
            this.Tavan = tavan;
            this.Yukseklık = yukseklik;
        }

        private double AlanHesapla()
        {
            return (this.Taban + this.Tavan) / 2 * this.Yukseklık;
        }

        public void EkranaYazdır()
        {
            Console.WriteLine("Yamuğun Alanı: " + this.AlanHesapla());
        }
    }

    class Nokta
    {
        public Nokta()
        {

        }

        public double AlanHesapla()

```

```

    {
        return 0;
    }

    public void EkranaYazdir()
    {
        Console.WriteLine("Noktanın alanı: " + this.AlanHesapla());
    }
}

class Dikdortgen
{
    private double _kenarKisa;
    public double KenarKisa
    {
        get
        {
            return _kenarKisa;
        }
        set
        {
            if (value < 0)
            {
                Console.WriteLine("Alan sıfırdan küçük olamaz...!");
                _kenarKisa = 0;
            }
            else
            {
                _kenarKisa = value;
            }
        }
    }

    private double _kenarUzun;
    public double KenarUzun
    {
        get
        {
            return _kenarUzun;
        }

        set
        {
            if (value < 0)
            {
                Console.WriteLine("Tavan 0 dan küçük olamaz...!");
                _kenarUzun = 0;
            }
            else
            {
                _kenarUzun = value;
            }
        }
    }
}

```

```

    }
}

public Dikdortgen(double kısaKenar, double uzunKenar)
{
    this.KenarKisa = kısaKenar;
    this.KenarUzun = uzunKenar;
}

public double AlanHesapla()
{
    return this.KenarUzun * this.KenarKisa;
}

public void EkranaYazdir()
{
    Console.WriteLine("Dikdörtgenin Alanı: " + this.AlanHesapla());
}
}

class Program
{
    static void Main(string[] args)
    {
        Yamuk y = new Yamuk(3,4,5);

        y.EkranaYazdir();

        Nokta n = new Nokta();
        n.EkranaYazdir();

        Dikdortgen d = new Dikdortgen(5, 7);
        d.EkranaYazdir();
    }
}

```