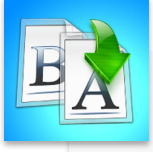


# SINIF YAPISINA GİRİŞ



## İÇİNDEKİLER

- Sınıflar (Class)
  - Sınıf Tanımlama
  - Nesne Tanımlama
  - this Anahtar Sözcüğü
  - Get ve Set Anahtar Sözcükleri



## HEDEFLER

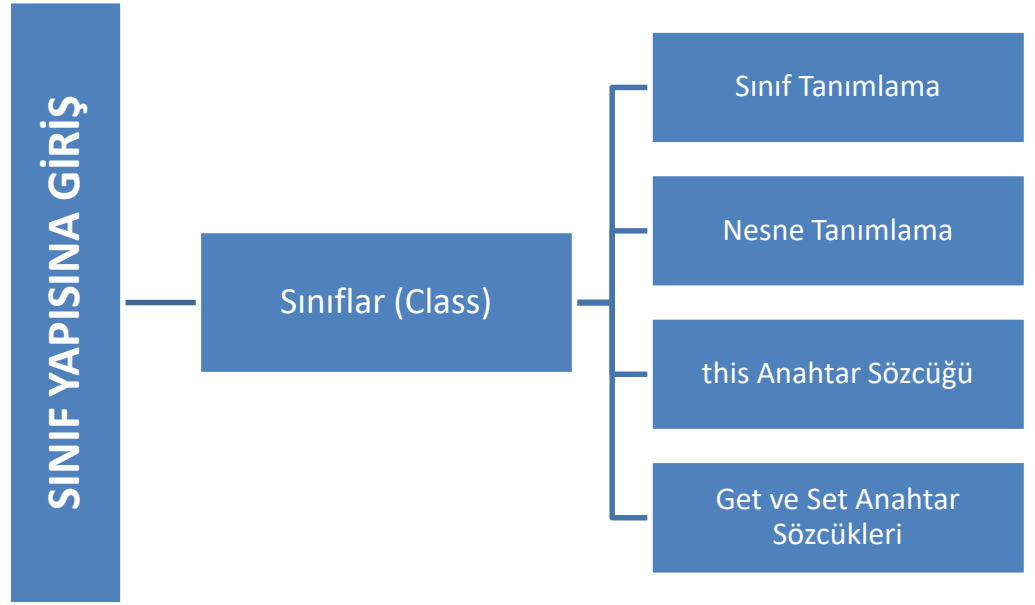
- Bu üniteyi çalıştıktan sonra;
  - Sınıf tanımlayabilecek,
  - Sınıftan nesneler oluşturabilecek,
  - Get ve Set metotlarını kullanabilecek,
  - Kalıtım kavramını öğrenebileceksiniz.



**Atatürk Üniversitesi**  
Açıköğretim Fakültesi

## NESNE TABANLI PROGRAMLAMA I Öğr. Gör. İsa BİNGÖL

# ÜNİTE 2



## GİRİŞ

Nesne yönelimli programlama, kod anlaşılabilirliğini arttıran, yazılım geliştirme sürecini kısaltan ve sistematikleştiren bir teknik ya da paradigma olarak kabul edilmektedir. Nesne yönelimli programlama yaklaşımının en önemli unsurları sınıf, nesne ve metot olarak sıralanabilir. Nesne bir sınıftan oluşturulan değişkenlerdir. Sınıf bir nesnenin özelliklerini barındıran yapılardır. Metot ise bu nesnenin gerçekleştirdiği işlerdir.

Sınıf konusu nesne tabanlı programlamanın en önemli konularından biridir. Bu kavramı iyi bilmek, projelerde nesneler ve metotlar oluşturup kullanabilmek hem yapılan projenin daha profesyonel bir yapıya kavuşmasına hem de kod tekrarının önüne geçmeye imkan tanır. Nesne yönelimli programlama yaklaşımı, gerçek hayatta karşılaşılan birçok şeyin bilgisayar ortamına aktarılabilmesini ilke edinmiştir. Nesne yönelimli programlama kapsamındaki sınıf, nesne veya metot gibi yapılar programcıya karşılaşılan bir problemi olabildiğince esnek şekilde modelleme imkanı sunar. Örneğin; motorlu araçları nesne yönelimli programlama yaklaşımına göre modellememizin istendiğini düşünelim. Motorlu araçlar geniş bir konudur ve içerisinde otomobil, uçak, motosiklet ve kamyon gibi birden fazla motorlu aracı barındırır. İşte burada motorlu araçları sınıf, otomobil, uçak, motosiklet ve kamyon gibi araçları da birer nesne olarak düşünebiliriz. Metot ise bu araçların gerçekleştirebildikleri işleri ifade eder. Örneğin, her araç çalışabilir, durabilir, gidebilir, yolcu (yük) taşıyabilir. Araçların yapmış olduğu bu işlemler metot olarak adlandırılabilir.

Bu bölümde temel olarak sınıf yapısına giriş anlatılacaktır. Bu kapsamda sınıf kavramı, sınıf tanımlama, nesne tanımlama, metot oluşturma, get ve set yapısı, kalıtım gibi konular örnekler üzerinden anlatılacaktır. Anlatım ve örnekler bir nesne tabanlı programlama dili olan C# üzerinden yapılacaktır.



C# nesne tabanlı bir programlama dilidir.

## SINIFLAR (CLASS)

Sınıflar, içerisinde tutulan bilgilerin ve özelliklerin daha düzenli ve sistematik bir yapıda tutulmasını sağlarlar. Aynı zamanda bir program yazarken sınıfları kullanmak, her bir nesne için ayrı ayrı kod yazmaya gerek kalmadan tek merkezden kontrol etmeye imkan tanır. Bu şekilde ilerde kod üzerinde bir değişiklik yapılmak istendiğinde tek bir yerden değişiklik yapmak yeterli olacaktır.

Giriş bölümünde de bahsedildiği üzere sınıflar nesnelerin özelliklerini ve gerçekleştireceği işleri barındıran yapılardır. Nesneler ise sınıflardan tanımlanmış değişkenlerdir. Bir sınıftan birden fazla nesne oluşturulabilir. Oluşturulan her nesnenin ise bir *özelliliği (properties)* ve yapacağı bir *işi yani metodu* olabilir.

Bu kavramları daha da somutlaştırarak yukarıda vermiş olduğumuz motorlu araçlar örneği üzerinden gidelim. Motorlu araçlar genel bir kavramdır ve içerisinde birçok aracı barındırabilir. Dolayısıyla motorlu araçlar bizim sınıfımızdır. Otomobil, otobüs, uçak gibi araçlar motorlu araç sınıfından türediği için bunların her biri motorlu araç sınıfından türeyen nesne gibi düşünebilirler. Her motorlu aracın

kendisine özgü rengi, kapı sayısı, yakıt türü gibi özellikleri vardır. Bu özelliklerin içeriği nesneye (otomobil, uçak, kamyon) değişir. Örneğin her motorlu araç yakıtla hareket eder ama yakıt olarak farklı içerikte maddeler kullanabilirler. Her aracın bir rengi vardır. Son olarak bütün motorlu araçların hareket etmek, yolcu taşımak gibi işlevleri vardır. Bir sınıfın sahip olduğu işlevlere nesne tabanlı programlama çerçevesinde “metot” denmektedir. Kısacası sınıf modellemesi programcının hayal gücüne ve problemi nasıl ele aldığına bağlıdır.



Bireysel Etkinlik

- Sınıf ve nesne kavramlarına çevrenizden nelerin örnek olabileceğini düşününüz.

## Sınıf Tanımlama

C#’da bir sınıf tanımlamak için **class** anahtar sözcüğü kullanılır. Class anahtar sözcüğünden sonra sınıf ismi belirtilir. Ardından küme parantezleri açılarak sınıfın özellikleri eklenir. Aşağıda örnek bir sınıf tanımlaması görülmektedir.



Örnek

```
•class sınıf_adi
{
    .....
}
```

Küme parantezleri arasında sabit, değişken ve metod gibi sınıfın öğeleri eklenir. Tanımlanan sınıf içine bir değişken ve bir metod örneği aşağıda gösterilmektedir;



Örnek

```
•class sınıf_adi
{
    erişim_türü veri_tipi değişken_adi;
    erişim_türü dönüş_tipi metod_adi(parametreler)
    {
        .....
    }
}
```



Varsayılan ayar olarak değişken ve metotlar özel (private)’dir.



Projeye birden fazla yolla sınıf eklenebilir.

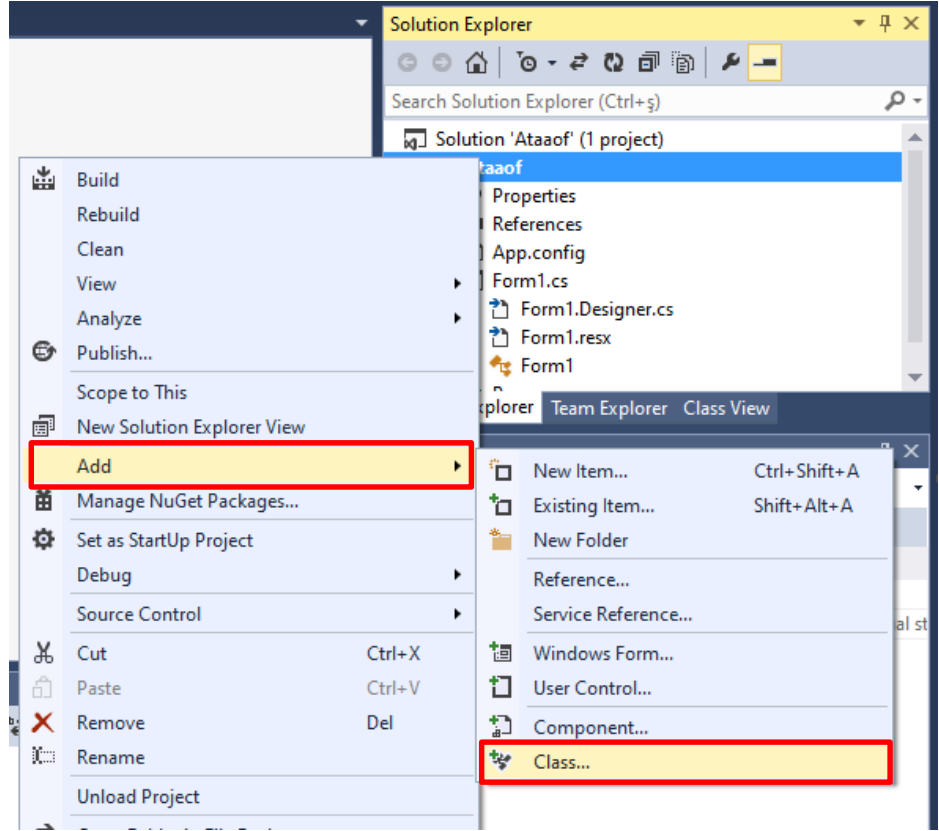
Yukarıda verilen örnek tanımlamaya dikkat edilecek olursa, değişken ve metod tanımlarken ilk başta bu değişken ya da metoda nerelerden erişileceğini belirlediğimiz bir erişim türü ekleriz. Eğer değişken ya da metoda sadece tanımlandığı sınıfın içinden erişilmek isteniyorsa erişim türü *private (özel)* olarak tercih edilmelidir. Değişken ya da metoda sadece tanımlandığı sınıfın içinden değil de sınıfın dışından da erişilmek isteniyorsa erişim türü *public (genel)* olarak tanımlanmalıdır.

Sınıf kavramını daha iyi anlamak için yukarıda bahsettiğimiz örneklerden yola çıkarak gerçek bir uygulama yapalım. Araba adında bir sınıf oluşturalım ve bu sınıftan nesneler oluşturalım. Öncelikle Visual Studio 2019 programımızı açıp aşağıdaki görseldeki gibi bir form tasarlayalım.

Resim 2.1. Örnek Form Tasarımı

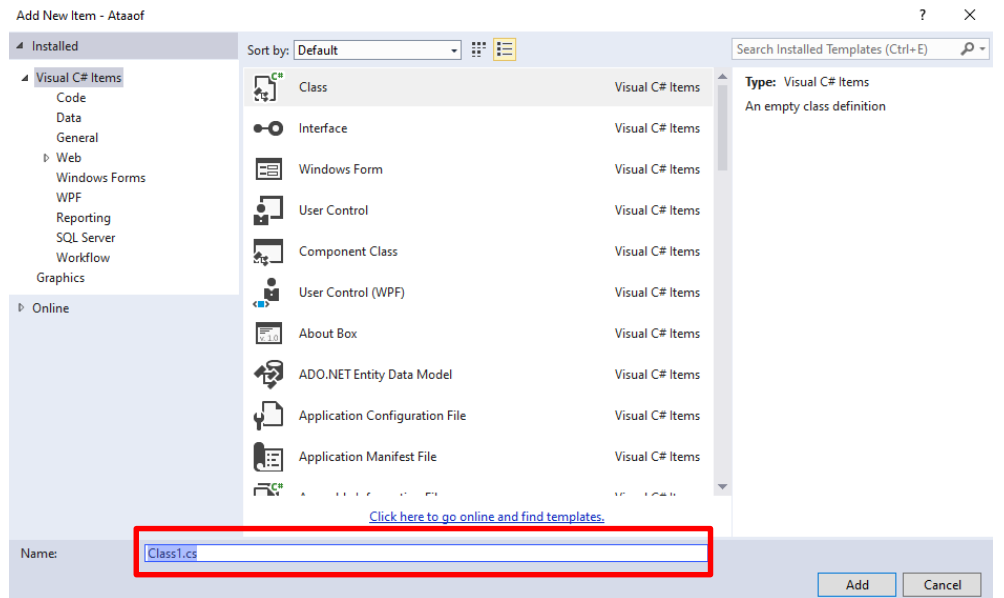
Bu örnekteki temel amacımız araba adında bir sınıf oluşturup, bu sınıftan iki farklı nesne oluşturmaktır. Söz konusu nesneler motorlu araçlar içerisinde tanımlanan araçları temsil edecektir. Daha sonra bu nesnelere tanımlanan özellikleri yukarıda verilen görselde ilgili alanlara yazdıracağız. Resim 2.1’de görseli paylaşılan formdan da anlaşılacağı üzere araba sınıfının özellikleri *marka, model, üretim yılı ve fiyat* olarak belirlenmiştir.

Projeye yeni bir sınıf eklemek için birden fazla yol kullanılabilir. İlk olarak üst menülerde yer alan *Project* menüsü ardından *Add Class* komutu kullanılabilir. Bir diğer yöntem ise *Solution Explorer* penceresinde yer alan proje adına sağ tıklayıp *Add>Class* seçeneğini kullanmaktır (Resim 2.2).



**Resim 2.2.** Projeye Class Ekleme

Bu yollardan herhangi biri tercih edildiğinde Resim 2.3'teki gibi bir pencere açılacaktır.



**Resim 2.3.** Class'a İsim Verme

Bu pencerede oluşturulacak olan sınıfa bir isim girilmesi istenecektir. Dilerseniz ismi hiç değiştirmeden Class1.cs olarak bırakıp Add butonuna tıklayabilirsiniz. Bu örnek kapsamında oluşturulan sınıfa *Araba* adını verelim.

Sınıfımıza adını verdikten sonra projemize yeni bir kod sayfası eklenecek ve içeriği aşağıda verilen kod bloğundaki gibi görünecektir.

```
namespace Ataaof
{
    class Araba
    {
    }
}
```

Burada öne çıkan bir diğer husus ise sınıfımız Ataaof namespace'i içinde tanımlanmış olmasıdır. Bunun anlamı projemizin bu sınıfı kapsadığıdır. Öte yandan sınıfın erişim türü ise gözükmemektedir. Çünkü varsayılan değer private (özel) olarak ayarlanmaktadır. Ancak bir sınıf istenildiği takdirde her yerden erişilebilir yani public (genel) olarak tanımlanabilir.



Sınıf tanımlamada class anahtar sözcüğü kullanılır.



Örnek

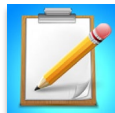
```
•public class Araba
{
}
```

Örnek olarak tanımladığımız araba sınıfının özelliklerini marka, model, üretim yılı ve fiyat olarak belirlemiştik. Şimdi bu özellikleri sınıfımızın içine tanımlayalım. Yapacağımız bütün tanımlamaları süslü parantezler arasında kodlayacağız.

```
public class Araba
{
    public string marka;
    public string model;
    public string uretim_yili;
    public int fiyat;
}
```



Bir sınıftan birden fazla nesne oluşturulabilir.



Bireysel Etkinlik

- İnsan türünde public bir sınıf oluşturup bu sınıfa ait 5 özellik tanımlayınız.

Özelliklerin hepsini public (genel) olarak tanımlayacağız. Marka, model ve üretim yılı üzerinde herhangi bir sayısal işlem yapılmayacağını göz önünde bulundurarak tanımladığımız özelliklerden marka ve modelin veri tipini string, fiyatın veri tipini ise int olarak belirleyelim.



Örnek

```
•Sınıf_Adı Nesne_Adı= new Sınıf_Adı();
```

Bir sınıftan yeni bir nesne oluştururken *new* anahtar sözcüğü kullanılmaktadır. Bu sözcük tanımladığımız sınıfın bir kopyasını hafızada oluşturacaktır. Şimdi tasarladığımız formumuza geri dönelim ve “*Araç Bilgilerini Getir*” butonunun click olayına yukarıdaki yapıyı göz önünde bulundurarak nesnemizi tanımlayalım.

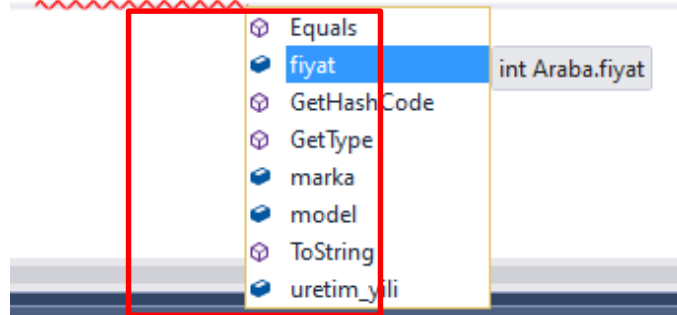


Örnek

```
•Araba otomobil1 = new Araba();
```

Bu örnekte Araba sınıfından otomobil1 adında bir nesne tanımladık. Şimdi de otomobil1 nesnesinin alacağı özelliklere değer atayalım. Kod ekranında nesne adını yazdıktan sonra nokta koyarsak Araba sınıfında tanımladığımız özelliklerin (fiyat, marka, model, üretim\_yili) geldiğini göreceksiniz (Resim 2.4).

```
Araba otomobil1 = new Araba();
otomobil1.
```



Resim 2.4. Sınıftan Nesne Oluşturma

Araba sınıfından oluşturmuş olduğumuz otomobil1 nesnesine bu özellikleri ekleyerek aşağıdaki gibi değer ataması yapalım.



Integer bir ifadeyi string türünde gösterirken tip dönüşümlerine dikkat edilmelidir.

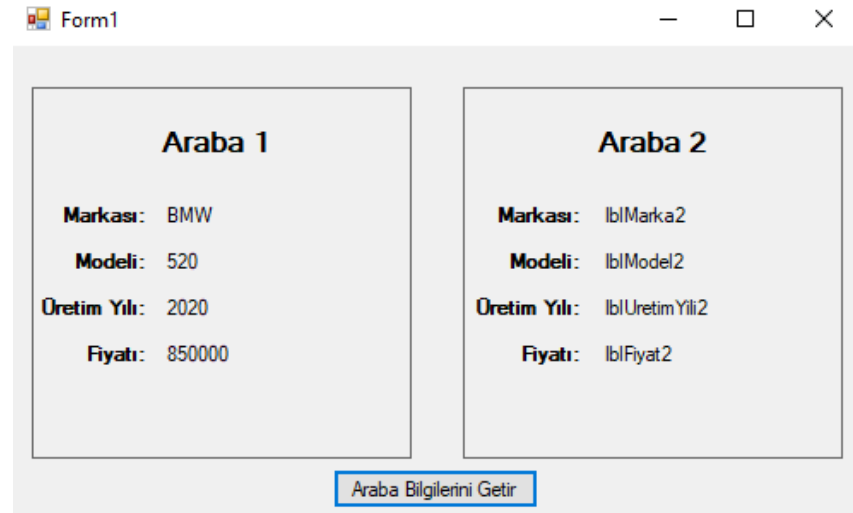


```
Araba otomobil1 = new Araba();  
otomobil1.marka = "BMW";  
otomobil1.model = "520";  
otomobil1.uretim_yili = "2020";  
otomobil1.fiyat = 850000;
```

Şimdi de sınıfın özelliklerine atadığımız bu değerleri tasarladığımız formda yer alan labellere yazdıralım. Bunu yaparken hata ile karşılaşmamız için tip dönüşümlerine dikkat etmemiz gerekmektedir. Örneğin, fiyat özelliğini integer olarak tanımladığımız için bunu label metnine yazdırırken ToString() metodu ile string tipine çevirmemiz gerekmektedir.

```
lblMarka1.Text = otomobil1.marka;  
lblModel1.Text = otomobil1.model;  
lblUretimYili1.Text = otomobil1.uretim_yili;  
lblFiyat1.Text = otomobil1.fiyat.ToString();
```

Form ekranındaki çıktısı Resim 2.5'deki gibi olacaktır.



The screenshot shows a Windows application window titled 'Form1'. Inside the window, there are two side-by-side panels. The left panel is titled 'Araba 1' and contains four lines of text: 'Markası: BMW', 'Modeli: 520', 'Üretim Yılı: 2020', and 'Fiyatı: 850000'. The right panel is titled 'Araba 2' and contains four lines of text: 'Markası: lblMarka2', 'Modeli: lblModel2', 'Üretim Yılı: lblUretimYili2', and 'Fiyatı: lblFiyat2'. At the bottom center of the window, there is a button labeled 'Araba Bilgilerini Getir'.

**Resim 2.5.** Formun Çalıştırılması

Program çalıştırıldığında Form ekranında yer alan Araba 2 alanına herhangi bir değer yazılmadığı görülmektedir. Bunun sebebi Araba sınıfından sadece bir tane nesne türetmemiz ve bunu da form ekranındaki Araba 1 alanında göstermemizdir.

Bir sınıftan istediğiniz kadar nesne oluşturabilirsiniz. Şimdi Araba sınıfından otomobil2 adında bir nesne daha oluşturalım ve bu nesnenin değerlerini de form ekranındaki Araba 2 alanında gösterelim. Bunun için yazmamız gereken kodlar aşağıdaki gibi olacaktır.

```

Araba otomobil2 = new Araba();
otomobil2.marka = "Renault";
otomobil2.model = "Megane";
otomobil2.uretim_yili = "2016";
otomobil2.fiyat = 250000;

lblMarka2.Text = otomobil2.marka;
lblModel2.Text = otomobil2.model;
lblUretimYili2.Text = otomobil2.uretim_yili;
lblFiyat2.Text = otomobil2.fiyat.ToString();

```



Bireysel Etkinlik

- Bitki adında bir sınıf oluşturarak bu sınıfa ait 3 özellik tanımlayınız. Daha sonra bu sınıftan iki adet nesne oluşturunuz.

Resim 2.6. Formun Çalıştırılması

## this Anahtar Sözcüğü

Metotlar, sınıf tarafından tanımlanan veriler üzerinde işlemler yapılmasına olanak tanıyan fonksiyon veya alt programlardır. Dolayısıyla metotlar yardımıyla geliştirilen programda yapılacak olan işlemler tekrar tekrar yazılmak yerine bir kere tanımlanır. Bu şekilde kod kalabalıklığından kurtulmuş olunur. Metot konusu ilerleyen ünitelerde detaylı olarak anlatılacağı için bu bölümde bu kavram üzerinde çok durulmayacaktır.

Bir metot çağrıldığı zaman bu metoda ilgili nesnenin bir referansı otomatik olarak aktarılır. Bu referans ise *this* olarak adlandırılmaktadır. Oluşturmuş



this anahtar sözcüğü sınıfın üye elemanlarını kasteder.

olduğumuz bir sınıfın üye elemanları ile metodumuzun parametreleri aynı isimle tanımlanmış olabilir. Dolayısıyla `this` anahtar sözcüğü bu ikisi arasındaki ayrımı yapmamızda bize yardımcı olabilmektedir. `this` anahtar sözcüğünü daha iyi anlamak için bir örnek üzerinden gidelim;

```
class sinif_yapisi
{
    int en, boy;

    public void alan_hesabi(int en, int boy)
    {

    }
}
```

Yukarıda yer alan koda dikkat edilirse `sinif_yapisi` adında bir sınıf tanımlanmış olup `en` ve `boy` adında `int` veri tipinde iki adet özellik eklenmiştir. Yine `alan_hesabi` adında tanımlanan metodun da `en` ve `boy` adında iki parametresinin olduğu görülmektedir. Peki metod içine `en` veya `boy` yazdığımızda bu sınıfın üye elemanı mı yoksa metodun parametresi mi bunu nasıl anlayabiliriz? Burada `this` anahtar sözcüğünü kullanabiliriz. Aşağıdaki kod incelendiğinde `this.en` ve `this.boy` ifadeleriyle sınıfın üye elemanları çağırılmıştır.

```
class sinif_yapisi
{
    int en, boy, alan;

    public void alan_hesabi(int en, int boy)
    {
        this.en = en;
        this.boy = boy;
    }
}
```

## Get ve Set Anahtar Sözcükleri

Bir sınıf içine eklenen özelliklerin erişim türlerinin `public` (genel) olarak tanımlanması o özelliklere direkt erişebilmeyi mümkün hale getirir. Bu da programlamada çok istediğimiz bir durum değildir. Ama bir metod yazarak bu özelliklere doğrudan erişmek yerine metod değişkenleri aracılığıyla erişebiliriz. Bu şekilde girilen değerlerin belirli aralıktaki olup olmadığını kontrol ederek olası problemlerin önüne geçebiliriz. Bir özelliği görüntülemek için bir metod bu özelliğin değerini değiştirmek için ise başka bir metod yazabiliriz. Ama her özellik için iki ayrı metod yazmak yerine *get ve set* anahtar sözcükleri kullanarak da sanki iki ayrı metod bildirmiş gibi oluruz.

Örneğin kullanıcıdan bir dörtgenin en ve boy değerlerini alarak alan hesaplaması yapacağımızı varsayalım. Kullanıcı en ve boy değerlerine farkında olmadan negatif değer girebilir. Bu tip sorunları ortadan kaldırmak için kullanıcının bu özelliklere direkt erişimini ortadan kaldırıp get ve set yapısıyla kontrol mekanizması kurabiliriz. Get ve Set yapısının temel kullanımı aşağıdaki gibidir.



Değişken tanımlarken büyük küçük harf duyarlılığına dikkat edilmelidir.

Örnek

```
•public veri_tipi nesne_adi
{
    get
    {
        .....
    }
    set
    {
        .....
    }
}
```

Bu yapı incelendiğinde kodların get ve set blokları arasına yazılacağı görülmektedir. Bir özelliğin ya da değişkenin değeri öğrenilmeye çalışıldığında get bloğu kullanılır. Bu blok içine yazılan kodla o değişkenin değerini geri göndermelisiniz. Geri gönderme işlemi *return* komutuyla gerçekleştirilir.

Bir özelliğin ya da değişkenin değeri değiştirilmeye çalışıldığında ise set bloğu kullanılır. Değişkenin değeri *value* isimli özel bir anahtar sözcük ile öğrenilir. Eğer set bloğu tanımlanmazsa o değer sadece okunabilir olur. Yine aynı şekilde eğer get bloğu tanımlanmazsa o değişkene sadece değer atanabilir ama okuma yapılamaz.



return get bloğunda, value set bloğunda kullanılır.

Şimdi get ve set metodu ile ilgili bir örnek yapalım. İnsan türünde bir sınıf tanımlayıp bu sınıftan Ali adında bir insan nesnesi oluşturalım. Daha sonra bu nesnenin yaşını get ve set metodu yardımıyla bir label de gösterelim. Sınıf oluşturma işlemini yukarıda anlatmıştık. Şimdi bu adımları tekrar ederek bir sınıf oluşturalım. Ardından yaş özelliğini tanımlayıp get ve set bloklarımızı oluşturalım

```
class İnsan
{
    private int yas;
    public int Yas
    {
        get
        {
            return yas;
        }
        set
        {
            yas = value;
        }
    }
}
```

Yukarıda yer alan koda dikkat edilirse yaş özelliğimiz private(özel) olarak tanımlandı ki bu özelliğe direk erişimi engelleyelim. Yani bu özelliğe değer atarken sadece bizim kontrolümüzde olmasını istedik. Söz konusu bu kontrol işlemini set bloğu arasında yapabiliriz. Ardından bu sefer get ve set bloklarımızın da içinde yer aldığı Yas adında public bir değişken tanımladık. Değer atama işlemlerinde buradaki Yas değişkeni aracı olarak kullanıldı. Program tarafında ise aşağıdaki kodları yazarak yas değerini lblYas adlı labelde gösterdik.



Örnek

- İnsan Ali = new İnsan();
- Ali.Yas = 25;
- lblYas.Text = Ali.Yas.ToString();

Burada kullanıcının yaş özelliğine değer atarken hata yapma ihtimaline karşı kontrol mekanizması ekleyebiliriz. Diyelim ki kullanıcı yaş özelliğine negatif değer atadı. Bunu şu şekilde kontrol ettirebiliriz.

```
class İnsan
{
    private int yas;
    public int Yas
    {
        get
        {
            return yas;
        }
        set
        {
            if (value < 0)
                yas = 0;
            else
                yas = value;
        }
    }
}
```

Set bloğu içinde, eğer sıfırdan küçük bir değer girildiğinde bunu sıfır olarak göster dedik. Burada bir kontrol mekanizması kurduk. Artık negatif girilen değerler sıfır olarak gösterilecektir.



**Bireysel Etkinlik**

- Get ve Set metotlarını kullanabileceğiniz bir örnek tasarlayınız.

Sınıf ve nesne kavramlarını daha iyi anlayabilmek için bir örnek daha yapalım. Ev adında bir sınıf tanımlayalım ve bu sınıftan nesneler oluşturalım. Bunun için Resim 2.7'deki gibi bir form tasarlayınız.

Resim 2.7. Örnek Form Tasarımı

Bu örnekteki temel amacımız ev adında bir sınıf oluşturup, bu sınıftan farklı nesneler oluşturmaktır. Söz konusu nesneler oluşturacağımız ev sınıfının özelliklerini temsil edecektir. Daha sonra bu nesnelere tanımlanan özellikleri yukarıda verilen Resim 2.7’de ilgili alanlara yazdıracağız. Resim 2.7’de görseli paylaşılan formdan da anlaşılacağı üzere ev sınıfının özellikleri *bina yaşı, kat sayısı, bulunduğu kat, oda sayısı, metrekaresi ve fiyatı* olarak belirlenmiştir.

Proje adına sağ tıklayıp *Add>Class* yolunu takip ederek (Resim 2.2) yeni bir sınıf ekleyelim. Bu örnek kapsamında oluşturacağımız sınıfa *Ev* adını verelim.

```
public class Ev
{
}
```

Örnek olarak tanımladığımız ev sınıfının özelliklerini bina yaşı, kat sayısı, bulunduğu kat, oda sayısı, metrekaresi ve fiyatı olarak belirlemiştik. Şimdi bu özellikleri sınıfımızın içine tanımlayalım.

```
public class Ev
{
    public int BinaYasi;
    public int KatSayisi;
    public int BulunduguKat;
    public int OdaSayisi;
    public double Metrekaresi;
    public int Fiyati;
}
```

Şimdi de form tasarımıımızda yer alan *Ev Bilgileri Göster* butonunun click olayına gelerek, ev sınıfından nesne oluşturarak sınıfın özelliklere değer atayalım.

```
private void button1_Click(object sender, EventArgs e)
{
    Ev ev1 = new Ev();
    ev1.BinaYasi = 3;
    ev1.KatSayisi = 7;
    ev1.BulunduguKat = 2;
    ev1.OdaSayisi = 4;
    ev1.Metrekaresi = 165;
    ev1.Fiyati = 450000;
}
```

Değer atama işleminden sonra bu değerleri tasarladığımız formda ilgili alanlara yazdıralım. Bunun için kodlarımız aşağıdaki gibi olacaktır. İlgili alanlara yazdırırken herhangi bir hata ile karşılaşmamız için tip dönüşümlerine dikkat etmemiz gerekmektedir. Integer olarak tanımladığımız özellikleri label metnine yazdırırken ToString() metodu ile string tipine çevirmeliyiz.

```
lblBinaYasi.Text = ev1.BinaYasi.ToString();
lblKatSayisi.Text = ev1.KatSayisi.ToString();
lblBulunduguKat.Text = ev1.BulunduguKat.ToString();
lblOdaSayisi.Text = ev1.OdaSayisi.ToString();
lblMetrekaresi.Text = ev1.Metrekaresi.ToString()+ " m^2";
lblFiyati.Text = ev1.Fiyati.ToString()+ " TL";
```

Form ekranındaki çıktısı Resim 2.8'deki gibi olacaktır.

Resim 2.8. Formun Çalıştırılması

Sınıfların bir özelliği de bir sınıftan başka bir sınıf türetilmesidir. Yani tanımlanan bir A sınıfından bir B sınıfı türeterek B sınıfına A sınıfından aldığı özelliklerin yanında yeni özellikler de eklenebilmektedir. Bu olaya *kalıtım* (*inheritance*) denilmektedir. A sınıfından bir B sınıfı oluşturulduğunda B sınıfının A



sınıfından türetildiğini göstermek için : işareti kullanılır (B:A). Aşağıda kalıtım kavramının örnek yapısı gösterilmiştir.



Örnek

```
•class A
{
    public int değişken1;
}
•class B : A
{
    public int değişken2;
}
```

Yukarıda A sınıfı değişken1 adlı özelliğe sahiptir. A sınıfından türetilen B sınıfı ise hem değişken1 hem de değişken2 özelliklerine sahip olur.

Kalıtım'a daha somut bir örnek verecek olursak, beyaz eşya adında bir sınıf oluşturduğumuz varsayalım. Bu sınıfa renk, fiyat ve marka adında üç özellik tanımlayalım. Ardından beyaz eşya sınıfından buzdolabı adında bir sınıf daha oluşturalım ve bu yeni sınıfa sadece raf sayısı özelliğini ekleyelim. En son oluşturduğumuz buzdolabı sınıfı beyaz eşya sınıfından türediği için raf sayısı özelliğinin yanında renk, fiyat ve marka özelliklerine de sahip olacaktır. Kalıtım konusu ilerleyen ünitelerde detaylı olarak anlatılacağı için bu bölümde bu kavram üzerinde çok durulmayacaktır.



Bireysel Etkinlik

- Kalıtım konusunun programcılara avantajları neler olabilir düşününüz.
- Elektronik aletler adında bir sınıf oluşturup ardından bu sınıftan 3 adet nesne türetiniz.



## Özet

- Nesne yönelimli programlama (Object oriented programming), bir programda sınıf (class) tanımlamaya ve bu sınıflardan yeni sınıflar ve nesneler (object) oluşmaya olanak tanıyan bir programlama tekniğidir.
- Nesne yönelimli programlama yaklaşımının en önemli unsurları sınıf, nesne ve metot olarak sıralanabilir.
- Nesne yönelimli programlama kapsamındaki sınıf, nesne veya metot gibi yapılar programcıya karşılaşılan bir nesneyi olabildiğince esnek şekilde modellemesini sağlar.
- C# nesne tabanlı bir programlama dilidir.
- Sınıflar, içerisinde tutulan bilgilerin ve özelliklerin daha düzenli ve sistematik bir yapıda tutulmasını sağlarlar.
- Nesne bir sınıftan oluşturulan değişkenlerdir. Sınıf o nesnenin özelliklerini barındıran yapıdır. Metot ise bu nesnenin gerçekleştirdiği işlerdir.
- Bir sınıftan birden fazla nesne oluşturulabilir. Ve her nesnenin bir özelliği (properties) ve yapacağı bir işi (metot) vardır.
- C#'da bir sınıf tanımlamak için class anahtar sözcüğü kullanılır. Class anahtar sözcüğünden sonra sınıf ismi belirtilir.
- Eğer değişken ya da metoda sadece tanımlandığı sınıfın içinden erişilmek isteniyorsa erişim türü private (özel) olarak ayarlanmalıdır. Değişken ya da metoda sadece tanımlandığı sınıfın içinden değil de sınıfın dışından da erişilmek isteniyorsa erişim türü public (genel) olarak tanımlanmalıdır.
- Oluşturulan bir sınıfın üye elemanları ile metodun parametreleri aynı isimle tanımlanmış olabilir. Burada sınıfın üye elemanlarını metodun parametrelerinden ayırmak için this anahtar sözcüğü kullanılabilir.
- Bir sınıf içine eklenen özelliklere doğrudan erişmeyi ortadan kaldırmak için o özelliği private (özel) olarak tanımlayıp o özelliğe get ve set metotları yardımıyla ulaşmak daha sağlıklı olacaktır.
- Bir özelliğin ya da değişkenin değeri öğrenilmeye çalışıldığında get bloğu kullanılır. Bu blok içine yazılan değer geri döndürülmelidir. Geri döndürme işlemi return komutuyla gerçekleştirilir.
- Bir özelliğin ya da değişkenin değeri değiştirilmeye çalışıldığında ise set bloğu kullanılır. Bu blok içinde değişkenin değeri value isimli özel bir anahtar sözcük ile öğrenilir.
- Metotlar sınıf tarafından tanımlanan veriler üzerinde işlemler yapılmasına olanak tanıyan fonksiyon veya alt programlardır.
- Sınıfların bir özelliği de bir sınıftan başka bir sınıf türetilmesidir. Yani tanımlanan bir A sınıfından bir B sınıfı tanımlanarak oluşturulan B sınıfı A sınıfının özelliklerinin yanında yeni özellikler de kazanabilir. Bu olaya kalıtım (inheritance) denilmektedir.

## DEĞERLENDİRME SORULARI

1. Bir metodun geriye değer döndürmesi istenmiyorsa aşağıdaki tiplerden hangisi kullanılmalıdır?
  - a) return
  - b) void
  - c) back
  - d) public
  - e) private
2. İnsan Ali = ..... İnsan (); noktalı yere aşağıdaki anahtar sözcüklerden hangisi gelmelidir?
  - a) class
  - b) object
  - c) method
  - d) new
  - e) public
3. Aşağıdaki yapılardan hangisi sınıf tanımında kullanılmaz?
  - a) Erişim Türü
  - b) Veri Tipi
  - c) Sınıf Adı
  - d) Süslü Parantezler
  - e) new anahtar sözcüğü
4. Aşağıdakilerden hangisi sınıfın üye elemanı ile metodun parametresini birbirinden ayırt etmede kullanılan anahtar sözcüktür?
  - a) this
  - b) value
  - c) return
  - d) new
  - e) class
5. Bir sınıftan başka bir sınıf türetirken sınıf isimleri arasına hangi sembol gelmelidir?
  - a) ~
  - b) :
  - c) ;
  - d) -
  - e) {}

6. Bir B sınıfının A sınıfından türetildiğini göstermek için aşağıdaki yapılardan hangisi kullanılabilir?
- a) B;A
  - b) A;B
  - c) A:B
  - d) B:A
  - e) A-B

```
private int yas;  
public int Yas  
{  
    get  
    {  
        return yas;  
    }  
    set  
    {  
        yas = .....;  
    }  
}
```

7. Yukarıda verilen kodlarda noktalı yere aşağıdakilerden hangisi gelmelidir?
- a) Yas
  - b) return
  - c) value
  - d) void
  - e) 0

```
public int Toplam (..... , .....)  
{  
    int sonuc = sayi1 + sayi2;  
    return sonuc;  
}
```

8. Yukarıda verilen kodlarda noktalı yerlere aşağıdakilerden hangisi gelmelidir?
- a) sayi1,sonuc
  - b) sayi1,sayi2
  - c) sonuc,sayi2
  - d) sayi1
  - e) sayi,sayi2,sonuc

9. Bir metot içerisinde geriye değer döndürmek için hangi anahtar kelime kullanılmalıdır?
- a) break
  - b) void
  - c) return
  - d) get
  - e) set
10. Bir değişken ya da metoda sadece tanımlandığı sınıfın içinden erişilmek isteniyorsa erişim türü ne olarak belirlenmelidir?
- a) public
  - b) properties
  - c) void
  - d) private
  - e) new

**Cevap Anahtarı**

1.b, 2.d, 3.e, 4.a, 5.b, 6.d, 7.c, 8.b, 9.c, 10.d

## YARARLANILAN KAYNAKLAR

Algan, S. (2010), Her Yönüyle C#, İstanbul: Pusula Yayıncılık, İstanbul.

Karagülle, İ. (2004). Visual C# .NET Başlangıç Rehberi. İstanbul: Türkmen Kitabevi.

Schildt, H. (2002). The Complete Reference C#, çev. Duygu Arbatlı Yağcı, Alfa Basım Yayım Dağıtım, İstanbul.