3 Sınıf ve Nesne

3.1 Sınıflar

Sınıf, nesne yönelimli programlama dillerinde nesnelerin özelliklerini, davranışlarını ve başlangıç durumlarını tanımlamak için kullanılan şablonlara verilen addır.

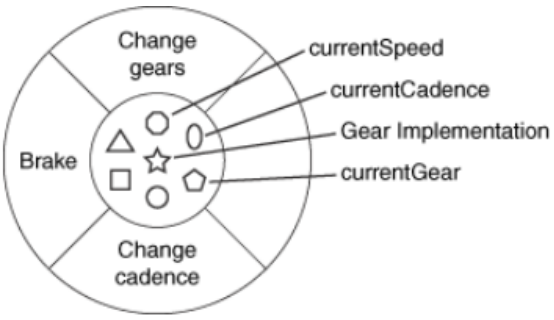
Bir sınıftan türetilmiş bir **nesne** ise o sınıfın örneği olarak tanımlanır.

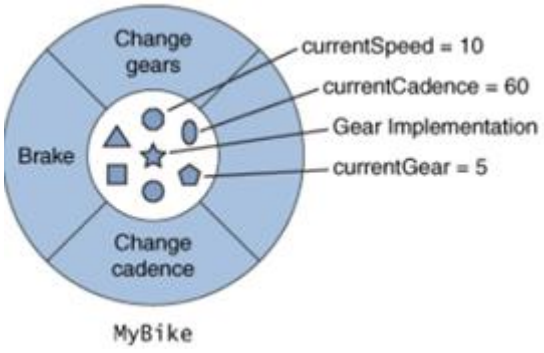
Sınıflar genelde şahıs, yer ya da bir nesnenin ismini temsil ederler.

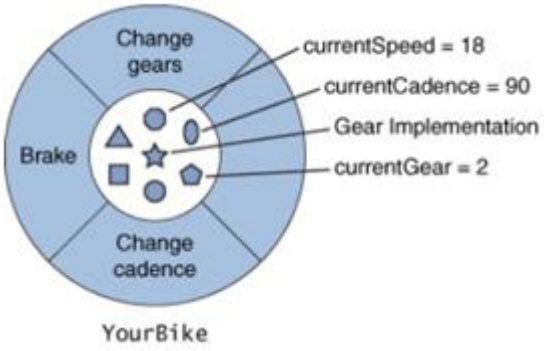
Sınıflar metotları ile nesnelerin davranışlarını, değişkenleri ile ise nesnelerin durumlarını tutar.

Sınıflar hem veri yapısına hem de bir ara yüze sahiptirler. Sınıflar ile nasıl etkileşime girileceği bu ara yüzler sayesinde sağlanır.

Örneğin bir sınıf şablonu ile araba: yakıt ve maksimum hız özelliklerine ve ayrıca depoyu doldur, o anki yakıtı gör, hızı ayarla, hızı göster ve arabayı kullan gibi metotlara sahip olabilir.







Sınıf Kullanılmasının Nedenleri

- Gerçek hayat problemleri sınıf şablonları kullanılarak bilgisayar ortamına daha kolay ve anlaşılabilir bir biçimde aktarılabilir.
- Sınıflar kodlarımızın daha düzenli ve erişilebilir olmasını sağlar.
- Nesne yönelimli programlamada herhangi bir projede kullanılmak üzere yaratılan bir sınıf başka projelerde tekrar kullanılabilir. (Reusability)
- Yeni veri tiplerinin oluşturulabilir olması
- Hata ihtimalinin azalması ve ayıklamanın kolaylaşması
- Gerektiğinde değişikliğin hızlı ve kolayca yapılması
- Takım çalışmasının desteklenmesi

3.1.1 Metotlar ve Özellikler

Metot ifadesi yerine yordam ya da fonksiyon gibi isimlendirmeler yapılabilir. Özellik (property) ifadesi yerine de variable (değişken), veri ve attritube ifadeleri de kullanılabilir.

Metotlar ve özellikler dört farklı erişim kuralına göre tanımlanabilir: `public`, `protected`, `private` ve `internal` olarak adlandırılmıştır.

`Private` (Gizli): Değişkene sadece kendi class'ı içinden ulaşılabileceği anlamına gelmektedir. Program içinde kesinlikle değiştirilmemesi gereken, kritik kodlarda kullanılmaktadır. Ayrıca; private, varsayılan erişim belirleyici tipidir. Örneğin; "int deneme = 0;" gibi bir değişken tanımlandığında program tarafından deneme değeri privateolarak algılanmaktadır.

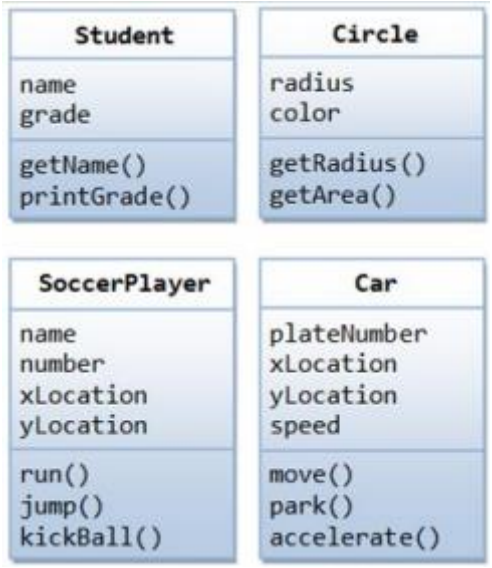
`Public` (Genel): Değerin, kod içinde herhangi bir yerden erişilebilir durumda olmasını sağlamaktadır. Public erişim belirleyici tipinde hiç bir kısıtlama yoktur.

`Protected` (Korunumlu): Değere, bulunduğu class ve ondan türetilen diğer sınıflar içinden erişilebilir olduğunu göstermektedir.

`Internal` (İçsel): Aynı program içerisinden erişilebilir, fakat farklı bir program içerisinden erişilemez durumdadır.

Metotlar bir değer döndürebilir (`return`). Bir metodun geri dönüş değerinin boş olması istendiğinde boş veri türü olan `void` kullanılmaktadır.

3.1.2 Sınıf Tanımlaması



Şekil 3-1. Örnek sınıf tanımlamaları

```
<Erişim_Belirteci> class <Sınıf Adı>
{
     //Özellikler
     //Metodlar
}
```

```
class oyuncu
{
    public string ad;
    public string mevki;
    public int savunma;
    public int atak;
    public int topSurme;
    public double deger;
}
```

Class View: Solution Explorer penceresinden “class view” seçilerek sınıfı ve sınıfın özellik ve metotlarını ağaç gösterimi şeklinde bulabilirsiniz.

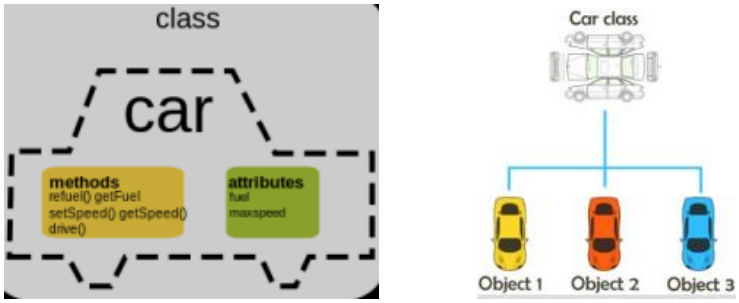
3.2 Nesne

NYP’de nesneler (obje) sınıflardan üretilir.

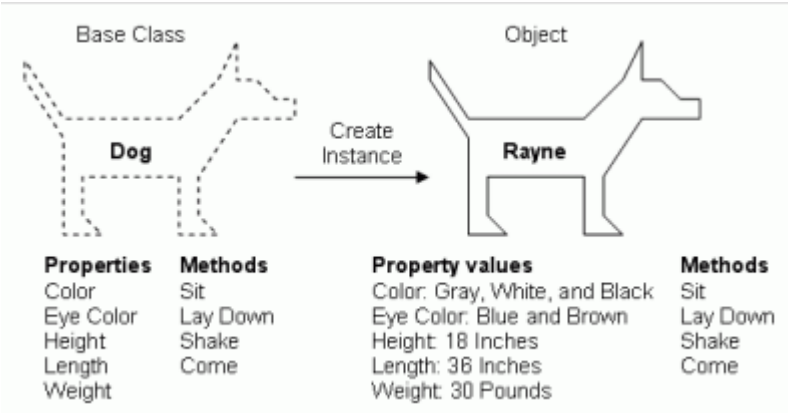
Nesneler, sınıfların aksine canlıdır ve kimlikleri vardır.

Aynı sınıftan üretilmiş iki objenin sahip olduğu değişkenler değişik özelliklere sahiptir.

Örneğin araba sınıfı maxspeed değişkeni araba türüne göre farklılık gösterir.



Şekil 3-2. “car” sınıfı ve bu sınıftan türetilen nesneler



Şekil 3-3. "dog" sınıfı ve bu sınıftan türetilen "Rayne" isimli nesne

3.2.1 Nesne Üretme

<Sınıf Adı> <Nesne Adı> = new <Sınıf Adı>;

Örnek Uygulama: “oyuncu” sınıfı ve bu sınıfı kullanan bir örnek program:

```
using System;
```

```
namespace yEmre
{
    class oyuncu
    {
        public string ad;
        public string mevki;
        public int savunma;
        public int atak;
        public int topSurme;
       public double deger;
     }

    class Futbol
    {
        static void Main(string[] args)
        {
            oyuncu oyuncu1 = new oyuncu(); //1. nesne üretiliyor
            oyuncu oyuncu2 = new oyuncu(); //2. nesne üretiliyor

            // 1. nesne özellikleri giriliyor
            oyuncu1.ad = "Emir";
            oyuncu1.mevki = "Santrafor";
            oyuncu1.savunma = 45;
            oyuncu1.atak = 79;
            oyuncu1.topSurme = 81;

            // 2. nesne özellikleri giriliyor
            oyuncu2.ad = "Kerem";
            oyuncu2.mevki = "Stoper";
            oyuncu2.savunma = 85;
            oyuncu2.atak = 59;
            oyuncu2.topSurme = 55;

            // nesnelerden özellik değerleri okunup ortalama
     hesaplanıyor
            Console.WriteLine("Oyuncuların ortalama top sürme
     kabiliyeti" + (oyuncu1.topSurme + oyuncu2.topSurme) / 2);
            Console.ReadKey();
        }
    }
}
```