

5.9 Soyut (Abstract) Sınıflar

Sınıflar ve metotlar soyut olarak tanımlanabilir.

5.9.1 Soyut sınıf:

- Diğer sınıflar tarafından kullanılmak üzere hazırlanmış sınıflardır.
- Soyut sınıflarda nesne üretilmez: `Cannot create an instance of the abstract type or interface 'Personel'`
- Soyut sınıflar soyut metotları içerebilir.

Örnek:

```
abstract class Personel
{
    public double tabanMaas, maas;
    ...
}
```

5.9.2 Soyut Metot

- Soyut sınıfların içine yazılırlar.
- Devralınan sınıflar tarafından ezilirler (**override**) edilirler.
- Private olmazlar.
- Tanımlanırken sadece dönüş tipi ve parametreleri ile belirtilirler.
- Devralınan sınıflarda override edilirken metot gövdesi yazılır.
- Devralınan sınıflarda override edilirken tipi vs.. değiştirilirse hata alınır:
'YetkiliPersonel.maasHesapla()': return type must be 'void' to match overridden member
'Personel.maasHesapla()'

Örnek:

```
abstract class Personel
{
    public double tabanMaas, maas;
    public abstract void maasHesapla(); //Buraya metot gövdesi
    yazarsak hata alırız!
}
```

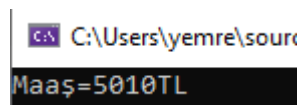
Soyut metotların ezilmesi (override) gerekir:

Örnek:

```
using System;
namespace Kalitim
{
    abstract class Personel
    {
        public double tabanMaas=5000, maas;
        public abstract void maasHesapla();
    }
    class YetkiliPersonel : Personel
    {
        double satisTutari=100;
        public override void maasHesapla()
        {
            maas = tabanMaas + satisTutari * 0.1;
            Console.WriteLine("Maaş=" + maas + "TL");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            YetkiliPersonel def = new YetkiliPersonel();
            def.maasHesapla();

            Console.Read();
        }
    }
}
```

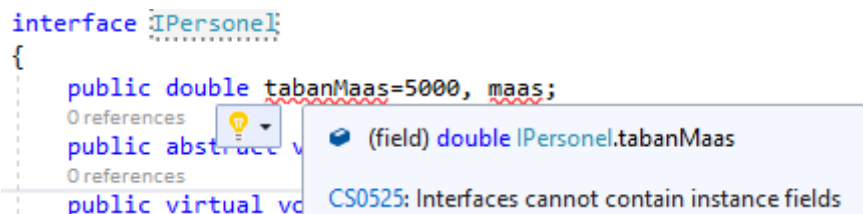
Çıktısı:



C:\Users\yemre\source
Maaş=5010TL

5.10 Arayüzler (Interface)

- İçerisine kod yazılmayan, diğer sınıflara rehber olması için kullanılan yapılardır.
- Interface içerisinde yalnızca metodlar ve propertyler tanımlanır, field tanımlanmaz.



```
interface IPersonel
{
    public double tabanMaas=5000, maas;
    public abstract void maasHesapla();
    public virtual void maasHesapla();
}
```

CS0525: Interfaces cannot contain instance fields

- Bir sınıf birden fazla arayüzü uygulayabilir.
- Bir sınıf uyguladığı arayüzün metotlarını ezme zorundadır.
- Arayüzler `public` olarak tanımlanır.
- Arayüzlerin yapıcı metodu olmaz.

Örnek:

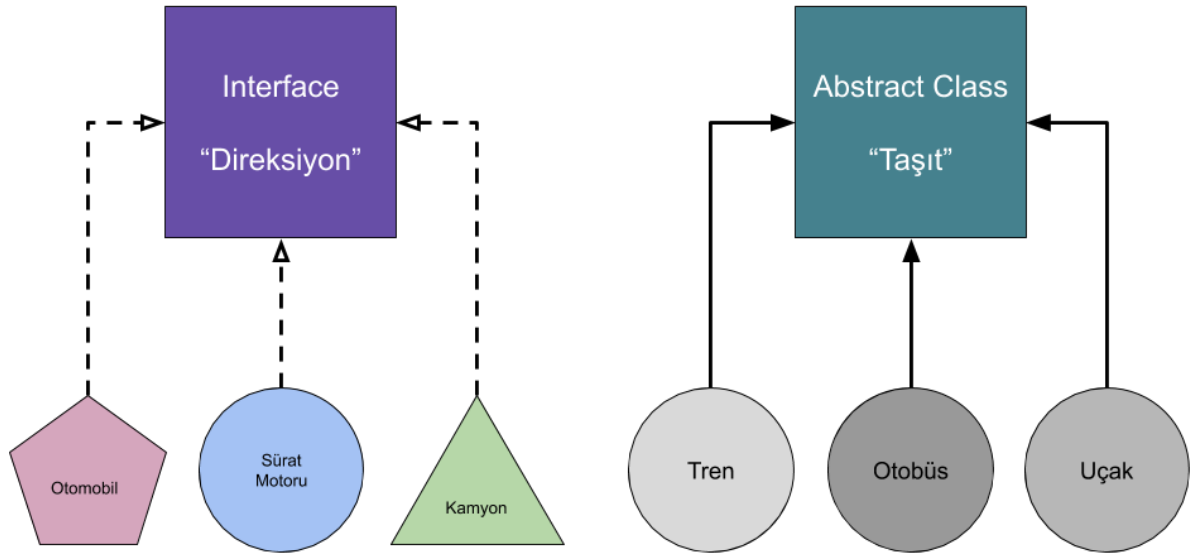
```
using System;

interface IAnimal
{
    void animalSound(); // interface method (gövdesi yok)
}

// Dog sınıfı IAnimal interface'ini "implement" ediyor
class Dog : IAnimal
{
    public void animalSound()
    {
        // animalSound() metodunun gövdesi burada yazılır
        Console.WriteLine("The dog says: bark bark");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Dog myDog = new Dog(); // Bir dog nesnesi üretilir
        myDog.animalSound();
    }
}
```

5.11 Interface ve Abstract Farkı



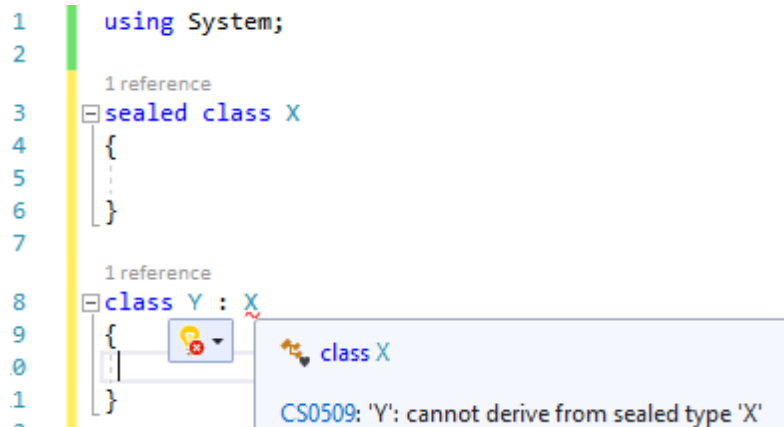
- Abstract class; statik metotlar içerebilir.
- Interface; override edilebilir statik metot içeremez.
- Abstract class'da metotların gövdeleri (yani implementasyonları) olabilir.
- Interface'de metotların ancak imzaları bulunabilir.
- Abstract class; kurucu (constructor) ve yıkıcı (destructor) içerebilir.
- Interface; kurucu veya yıkıcı içeremez. Ancak imzalarını içerebilir.
- Abstract class'ın her access modifier'a (private, protected, public gibi) sahip metodu bulunabilir.
- Interface'in metot imzaları ancak 'public' access modifier'ına sahip olabilir.
- Abstract class -genel olarak- ancak bir sınıftan inheritance alabilir. Veya bir sınıf -genel olarak- ancak bir abstract sınıftan inheritance alabilir.
- Interface birden fazla interface'den inheritance alabilir.
- Abstract class, sınıfın neyden türediğini ifade edebilir. Bir (... is a ...) ilişkisini gösterir.
- Interface, sınıfın hangi yeteneklere sahip olduğunu ifade edebilir. Bir (... can do ...) ilişkisini gösterir.
- Ortak sınıf davranışı kazandırma için abstract class kullanılmalıdır.
- Ortak yetenek metodu kazandırma için interface kullanılmalıdır.
- Abstract class, nesnenin ne yapması gerektiğini belirlemek ile beraber nasıl yapması gerektiğini de belirleyebilir.

- Interface, nesnenin ne yapması gerektiğini belirler ama nasıl yapması gerektiğini belirlemez.

5.12 Sealed Tanımlaması

Sealed, sınıfların kalıtım işlemini engellemek için kullanılan bir anahtar kelimedir.

Sealed anahtar kelimesi bir sınıf için uygulanırsa kalıtımı, bir üye için uygulanırsa üyenin override edilmesini engeller.



6 Numaralandırma (Enumeration)

Numaralandırma (enumeration) kod içerisinde sayısal karşılaştırma veya işlem gerektiren yerlerde yazılımcı için okunabilirliği artıran ve kod karmaşasını azaltan yardımcı bir yapıdır.

Bir enumeration tamsayı sabitlerini adlandıran bir kümedir.

enum anahtar sözcüğü ile bildirilir.

Koda uzun süre sonra tekrar bakıldığında anlaşılmasını kolaylaştırır. Tanımlaması:

```
enum enum_ismi { deger1,deger2,deger3};
```

Örnek tanımlamalar:

```
enum Renkler { Kirmizi, Yesil, Mavi };
enum Departmanlar { Yazilim, Bilgi_Işlem, Muhasebe };
enum Gunler { Pazartesi, Sali, Carsamba, Persembe, Cuma, Cumartesi,
    Pazar };
```

enum tanımlamamıza sayı değerleri de atabiliriz:

```
enum Gun { Pazartesi = 1, Sali = 2, Carsamba = 3, Persembe = 4, Cuma = 5, Cumartesi = 6, Pazar = 7 }; // doğru
```

Örnek:

```
using System;

class Program
{
    enum Gunler { Pazartesi, Sali, Carsamba, Persembe, Cuma, Cumartesi, Pazar };
    enum Gun { Pazartesi = 1, Sali = 2, Carsamba = 3, Persembe = 4, Cuma = 5, Cumartesi = 6, Pazar = 7 }; // doğru

    static void Main(string[] args)
    {
        Gun secilenGun = Gun.Carsamba;

        if (secilenGun == Gun.Cumartesi || secilenGun == Gun.Pazar)
        {
            Console.WriteLine("Hafta sonu seçtiniz.");
        }
        else
        {
            Console.WriteLine("Hafta içi seçtiniz.");
        }
    }
}
```