

5 Kalıtım (Inheritance)

Kalıtım ya da miras alma (Inheritance) bir nesnenin özelliklerinin farklı nesneler tarafından da kullanılabilmesine olanak sağlayan NYP özelliğidir.

Yazılan bir sınıf bir başka sınıf tarafından miras alınabilir. Bu işlem yapıldığı zaman temel alınan sınıfın tüm özellikleri yeni sınıfa aktarılır.

Örnek olarak temel veri türleri dediğimiz `byte`, `int`, `uint`, `short`, `float` ve bezerlerinin tamamı `object` sınıfından türetilmiştir. Bu sayede normalde `object` sınıfında bulunan `ToString()` metodu bu türlerdeki nesnelerde de kullanılabilir.

Örnek:

İnsan – Canlı ilişkisinde, insanın Canlı sınıfını miras aldığı söylenebilir. Bu sayede insan sınıfını yazarken canlı özelliklerini tekrar yazmamıza gerek kalmaz. Örneğin nefes al fonksiyonu canlıya yazılır insana tekrar yazılmasına gerek yoktur.

Elinizde bir taşıt sınıfı varsa; otomobil, kamyon, motosiklet gibi alt sınıfları üretmek çok daha az çaba gerektirir. Taşıt sınıfında tekerlek özelliği tanımlanır. Bu özellik otomobil için 4, kamyon için 6 ve motorsiklet için 2 vb... yapılarak, her bir sınıf için tekerlek tanımlamak yerine taşıta tanımlandığımız tekerlek özelliği sayesinde üç sınıfı da yönetebiliriz. Tekerlek özelliği taşıt sınıfından diğer sınıflara miras kalmış olur.

Diyelim ki `Sayı` isminde bir sınıfımız var ve `Deger` isminde bir sınıfa daha ihtiyacımız var. Bu `Deger` sınıfının içinde `Sayı` sınıfındaki özellik ve metotların da bulunması gerekiyor. Böyle bir durumda `Deger` sınıfını `Sayı` sınıfından türetmeniz gerekir. Türetme kalıtım yoluyla olduğu için `Sayı` sınıfının bütün üye elemanları `Deger` sınıfına adeta kopyalanır. Daha sonra `Deger` sınıfının kendine özel üye elemanları eklenebilir.

Örnek:

```
class Sayi
{
    //...
}
class Deger : Sayi
{
    //...
```

}

5.1 Erişim Belirteçleri

public olarak tanımlanan öge, kod bloğunun içinde ve dışında tamamen erişilebilirdir. Yani, hiçbir kısıtlama yoktur. Veri kontrol işlemi gerektiğinde veya güvenlik açısından önemli olan üyeler kesinlikle public tanımlanmamalıdır.

private olarak tanımlanan öge, sadece tanımlandığı class'ın içerisinde erişilebilirdir. En katı erişim belirleyicidir. Sadece kendi sınıfından o üyeye ulaşabiliriz. Sınıf dışından gereksiz müdahalelerin önüne geçmiş oluruz. Ayrıca bir fonksiyon veya property üzerinden bu değişkenleri yönetebiliriz. Bu yüzden kontrol ve güvenlik açısından bize destek sağlar.

protected olarak tanımlanan öge, sadece tanımlandığı class'ın içinde ve o class'tan türetilmiş diğer class'ların içinde erişilebilirdir.

internal olarak tanımlanan öge, bulunduğu assembly'nin (DLL veya EXE dosyası) içinde erişilebilirdir. DLL veya EXE dosyasının içerisinde erişim için kısıtlama yoktur, ama dışarıdan erişilemez.

protected internal erişim belirleyicisi, protected ve internal erişim belirleyicilerinin VEYA (OR) işlemiyle birleştirilmiş halidir. protected internal olarak tanımlanmış öge, tanımlandığı class'ın içinde ve o class'tan türetilmiş diğer class'ların içinde erişilebilir. Ayrıca, aynı assembly içinde olmasalar dahi, tanımlandığı class'tan türetilmiş diğer class'ların içinde de erişilebilirdir.

5.2 Eğer erişim belirteci kullanılmazsa

Class varsayılan olarak internal erişim belirleyicisine sahip olur.

Class içerisinde tanımlı ögeler ise varsayılan olarak private erişim belirleyicisine sahiptir.

5.3 Base ve This Kullanımı

`base`: Kalıtımda temel alınan (kendisinden türetilen) sınıfı temsil eder.

`this`: İçinde bulunulan sınıfı temsil eder.

A sınıfı B sınıfından türetilmişse B sınıfında kullanılan bir `this` ifadesi iki sınıfı da kapsar.

B sınıfında `base` kullanılırsa sadece A sınıfını kapsar.

Örnek:

```
using System;
namespace operatorAsiriYukleme
{
    class Sayi
    {
        public int a;
        private int b; //Alt sınıftan ve main'den ulaşılamaz
        protected int c = 250; //Alt Sınıftan ulaşılır, main ulaşamaz

        public int toplamHesapla()
        {
            return this.a + c; //this Sayi sınıfını işaret eder
        }
    }
    class Deger : Sayi
    {
        public int m;
        protected int n;

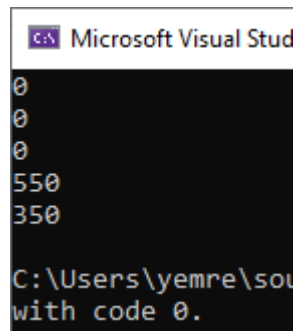
        public int toplamBul()
        {
            return base.a + this.m + c; //base Sayi sınıfını işaret
            eder.
            //this ise hem Sayi hem de
            Deger
            // sınıfına ulaşır
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Sayi x = new Sayi();
            Deger y = new Deger();

            //Ulaşılabilen alanlar:
            Console.WriteLine(x.a);
            Console.WriteLine(y.a);
            Console.WriteLine(y.m);

            y.a = 100;
            y.m = 200;
            Console.WriteLine(y.toplamBul());
            Console.WriteLine(y.toplamHesapla());

            Console.ReadKey();
        }
    }
}
```

```
    }  
}  
}
```



```
C:\Users\yemre\source\code\ Microsoft Visual Stud  
0  
0  
0  
550  
350  
  
C:\Users\yemre\source\code\  
with code 0.
```