Nesne Tabanlı Programlama
Bölüm 6
Kalıtım (inheritance)

Dr. Öğr. Üyesi Murat TAŞYÜREK (kayubmprogramlama1@gmail.com)

6 Kasım 2023

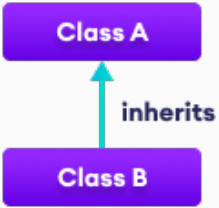
Kayseri Üniversitesi, Bilgisayar Mühendisliği Bölümü

Ders İçeriği

- Kalıtım türleri
- Base kelimesinin kullanımı
- Overloading
- Sealed
- Overrading

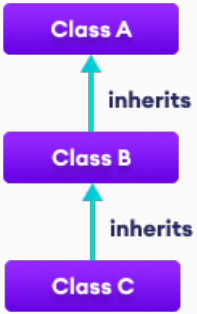
Kalıtım türleri - Tekli kalıtım

- Tekli kalıtım modelinde, tek bir türetilmiş sınıf, tek bir temel sınıftan miras alır.



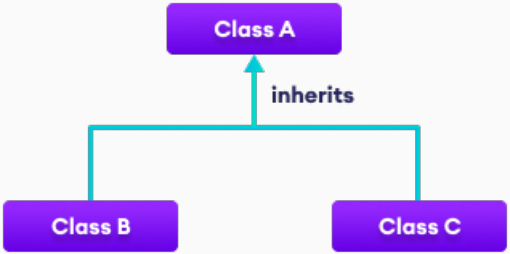
Kalıtım türleri - Çok Düzeyli kalıtım

- Çok düzeyli kalıtım modelinde (Multilevel Inheritance), türetilmiş bir sınıf bir temel sınıftan miras alır ve daha sonra aynı türetilmiş sınıf başka bir sınıf için temel sınıf görevi görür.



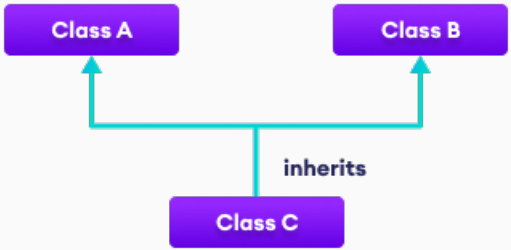
Kalıtım türleri - Hiyerarşik kalıtım

- Hiyerarşik kalıtım modelinde (Hierarchical Inheritance), birden çok türetilmiş sınıf tek bir temel sınıftan miras alır.



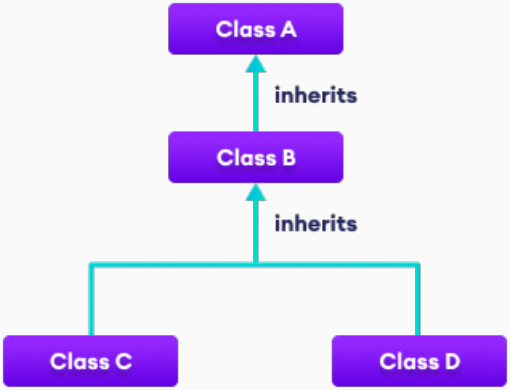
Kalıtım türleri - Çoklu kalıtım

- Çoklu kalıtım modelinde (Multiple Inheritance), tek bir türetilmiş sınıf, birden çok temel sınıftan miras alır. C# çoklu kalıtımı desteklemez. Ancak, arayüzler (interface) aracılığıyla çoklu kalıtım elde edebiliriz.



Kalıtım türleri - Hibrit kalıtım

- Hibrit kalıtım (Hybrid Inheritance), iki veya daha fazla kalıtım türünün birleşimidir. Çok düzeyli ve hiyerarşik kalıtımın birleşimi, Hibrit kalıtımın bir örneğidir.

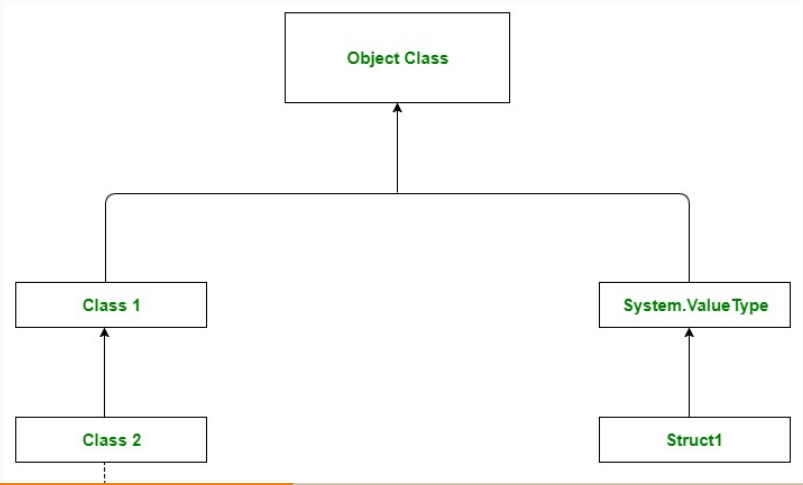


Kalıtım türleri - Gizli kalıtım

- Oluşturulan her yeni sınıf otomatik ve gizli olarak otomatik olarak **Object** sınıfıtan türetilir.
- C# ve Java gibi nesne yönelimli programlama dillerinin çoğunda böyledir.
- **Object** sınıfı C# ve Java programlama dillerinde kullanılan tüm sınıfların atasıdır (base class).
- Object sınıfı, .Net Framework'teki tüm sınıflar için temel sınıftır. System namespace alanında bulunur. C#'da, .NET Base Class Library(BCL), System.Object olarak tam olarak nitelenmiş ada sahip Object sınıfı olan dile özgü bir diğer ada sahiptir.

Kalıtım türleri - Gizli kalıtım

- C#'daki her sınıf, doğrudan veya dolaylı olarak Object sınıfından türetilir. Bir sınıf başka bir sınıfı genişletmiyorsa, Object sınıfının doğrudan alt sınıfıdır ve başka bir sınıfı genişletiyorsa dolaylı olarak türetilir. Bu nedenle, Object sınıfı yöntemleri tüm C# sınıfları tarafından kullanılabilir.



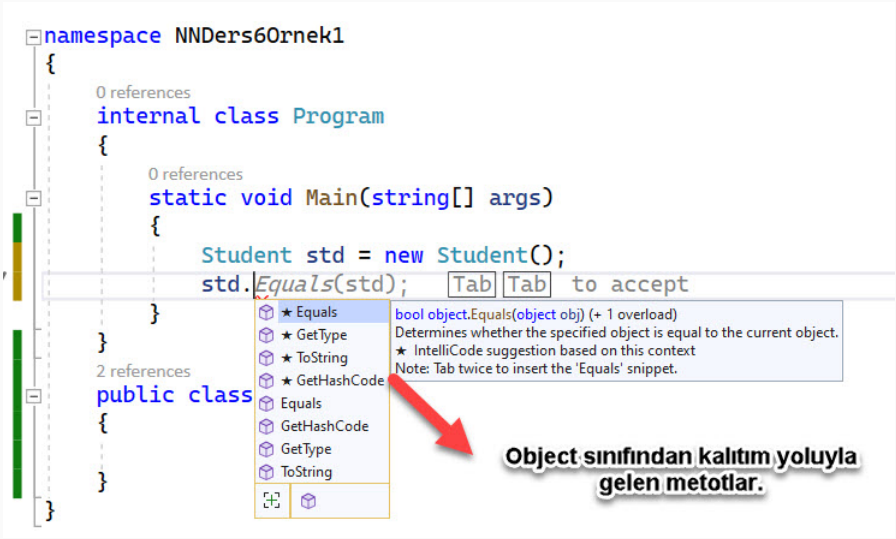
Kalıtım türleri - Gizli kalıtım -Örnek

- C# **Object** sınıfı **Equals, GetType, ToString ve GetHashCode** metotlarına sahiptir.
- Bu her sınıfın otomatik olarak **Object** sınıfından türediği için bu metotlara sahip olacaktır.
- Örneğin bir öğrenci sınıfı tanımlayalım.
- Öğrenci sınıfın hiç bir özelliği, özniteli ve metotu olmasın.
- Daha sonra bu sıfıftan bir nesne üretelim.

Stutdent Class

```
namespace NNDers6Ornek1
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            Student std = new Student();
        }
    }
    2 references
    public class Student
    {

    }
}
```

Stutdent Class Kaltım Yoluyla Gelen Metotlar



Kalıtım- Method Overriding

- Hem temel sınıfta hem de türetilmiş sınıfta aynı yöntem varsa, türetilmiş sınıftaki yöntem, temel sınıftaki yöntemi geçersiz kılar. Buna C#'da yöntem geçersiz kılma denir.
- Bunun için **override** kelimesi kullanılır.
- Bu kelime ile aynı isimli fonskiyon veya procedüre alt sınıftada türetilmiş olur.
- Örneğin Student sınıfı için ToString() metodunu çağırdımızıda geriye öğrencinin adını ve soyadının dönmesini isteyelim.
- ToString() fonksiyonu Object sınıfıtın bir metodudur. Her sınıfda Object sınıfından türediği için override etmemiz gerekir.

Stutdent Class

```
public class Student
{
    private string name, surname;
    1 reference
    public Student(string name, string surname)
    {
        this.name = name;
        this.surname = surname;
    }
    1 reference
    public override string ToString()
    {
        return name + " " + surname;
    }
}
```

Source Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace NNDers6Ornek2
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            Student ogr = new Student("Murat", "Taşyürek");
            string strDegeri=ogr.ToString();
            Console.WriteLine(strDegeri);
        }
    }
    3 references
    public class Student...
}
```

Çıktı

```
6
7   namespace NNDers6Ornek2
8   {
        0 references
9       internal class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              Student ogr = new Student("Murat", "Taşyürek");
14              string strDegeri=ogr.ToString();
15              Console.WriteLine(strDegeri);
16          }
17      }
        3 references
18      public class Student...
31  }
32
```

```
C:\WINDOWS\system32\cmd.exe
Murat Taşyürek
Press any key to continue . . .
```

Kalıtım- Method Overriding

- **Override** edilen yöntemin, override ettiği yöntemden aynı veya daha yüksek bir erişim belirleyecisine sahip olması gerekir.
- Yani override ettiği yöntem main sınıfta protected ise alt sınıfta proteced veya public olmalı. Eğer main sınıfta public ise alt sınıfta sadece public olabilir.
- C#'ta türetidliği sınıfın özelliklerine veya metotlarına erişmek için **base** kelimesi kullanılır.
- Ancak **override** edebilmek için temel sınıfta o yeteneğin değiştirilebileceğinin belirtilmesi gerekir.
- Bunun içinde **virtual** kelimesi daha sonradan değiştirebileceğinin belirtilmesi gerekir.
- The **virtual** keyword is used to modify a method, property, indexer, or event declared in the base class and allow it to be overridden in the derived class.

Örnek

- Animal türünde bir sınıfımızın olduğunu düşünelim.
- Bu sınıfın protected olarak tanımlanmış ve yemek yediğini gösteren bir metodunun olduğunu varsayalım.
- Dog içinde Animal sınıftan kalıtım yoluyla türetilen bir sınıf olduğunu düşünelim ve bunun da yemek yediğini göstermek için aynı isimde bir fonksiyonun olduğunu varsayalım. Ancak bu fonskiyonde öncelikle Animal sınıfındaki yemek metodunun tekiklenmesini sağlayalım.

Animals and Dog Class

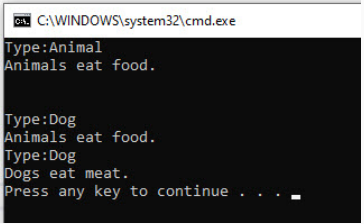
```
3 references
public class Animal
{
    4 references
    public virtual void Eat()
    {
        Console.WriteLine("Type:" + this.GetType().Name);
        Console.WriteLine("Animals eat food.");
    }
}
2 references
public class Dog : Animal
{
    4 references
    public override void Eat()
    {
        Console.WriteLine("\n");
        base.Eat();
        Console.WriteLine("Type:" + this.GetType().Name);
        Console.WriteLine("Dogs eat meat.");
    }
}
```

Source Code

```
namespace NNDers6Ornek3
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            Animal anm = new Animal();
            anm.Eat();

            Dog dg = new Dog();
            dg.Eat();
        }
    }
    3 references
    public class Animal ...
    2 references
    public class Dog ...
}
```

Çıktı

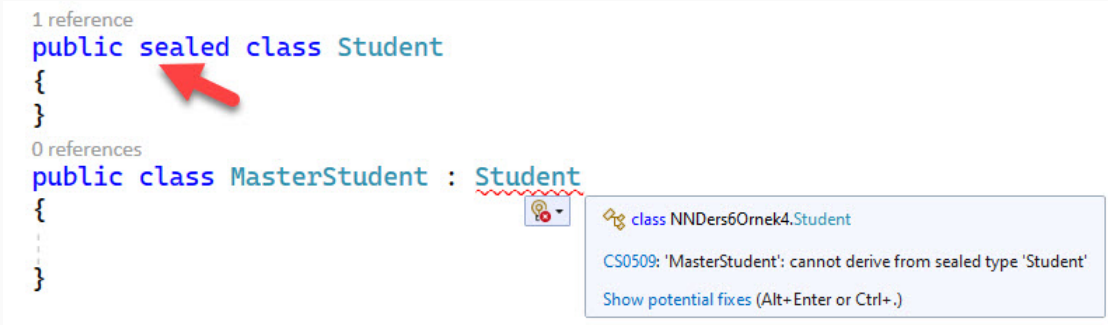
```
namespace NNDers6Ornek3
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
            Animal anm = new Animal();
            anm.Eat();

            Dog dg = new Dog();
            dg.Eat();
        }
    }
    3 references
    public class Animal...
    2 references
    public class Dog ...
}
```



No Kalıtım

- Nense yönelimli programlama ve gerçek hayattada bazen kalıtım olması istenmez.
- Bu durumda sınıfın kalıtım yoluna kapalı olduğunu belirtmek gerekir.
- Bunun için de **sealed** kelimesi kullanılır.
- Bu kelime kullanıldığı zaman sınıftan **kalıtım yapılamaz.**
- Metodu **sealed** kullanımı direk olmaz. Ancak üst sınıftaki bir metod alt sınıfta sealed override ile kapatılabilir. Bu durumda buradan türeyecek sonraki sınıfta metod override edilemez.

Sealed Class



Sealed Method

```
public class SealedMethodDemo
{
    1 reference
    public virtual void Method1()
    {
        Console.Write("Base class Method1");
    }
}
1 reference
public class ChildClass : SealedMethodDemo
{
    1 reference
    public sealed override void Method1()
    {
        Console.Write("Base class Method1");
    }
}
0 references
public class ChildOfChild : ChildClass
{
    public override
}
```

override edilecek metod yok.

Equals(object obj)
GetHashCode()
ToString()

bool object.Equals(object obj)
Determines whether the specified object is equal to the current object.

Overloading

- Aynı isimde farklı parametre türü alan veya farklı sayıda parametre alan metodlara ithiayç duyarız.
- C#'ta aynı sınıfın içinde iki veya daha fazla metot parametre tanımlamaları farklı olması şartı ile aynı ismi taşıyabilir.
- Bu duruma metodun aşırı yüklenmesi (**overloading**) denir.
- Bir metodun aşırı yüklenebilmesi için farklı parametre tipi veya sayısına sahip olması gerekmektedir.

Örnek

- Toplama işlemi için bir sınıf oluşturduğumuz düşünelim.
- Sınıfın topla isminde bir metodu olsun. Bu metot 2 tane parametre alabilsin.
- Ancak bu parametreler int int, string string, int str, str int olabilsin.
- Str gönderildiğde uzunluklarını toplasın integer gönderdiğinde direk toplasın.

Class

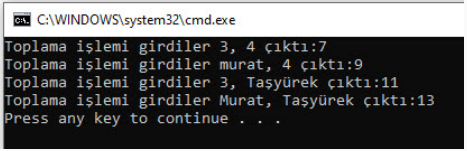
```
public class ToplamaIslemi
{
    1 reference
    public void Topla(int a, int b)
    {
        int sonuc = a + b;
        Console.WriteLine(String.Format("Toplama işlemi girdiler {0}, {1} çıktı:{2}",a,b,sonuc));
    }
    1 reference
    public void Topla(int a, string b)
    {
        int sonuc = a + b.Length;
        Console.WriteLine(String.Format("Toplama işlemi girdiler {0}, {1} çıktı:{2}", a, b, sonuc));
    }
    1 reference
    public void Topla(string a, string b)
    {
        int sonuc = a.Length + b.Length;
        Console.WriteLine(String.Format("Toplama işlemi girdiler {0}, {1} çıktı:{2}", a, b, sonuc));
    }
    1 reference
    public void Topla(string a, int b)
    {
        int sonuc = a.Length + b;
        Console.WriteLine(String.Format("Toplama işlemi girdiler {0}, {1} çıktı:{2}", a, b, sonuc));
    }
}
```

Source Code

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        ToplamaIslemi tt = new ToplamaIslemi();
        tt.Topla(3, 4);

        tt.Topla("murat", 4);

        tt.Topla(3, "Taşyürek");

        tt.Topla("Murat", "Taşyürek");
    }
}
2 references
public class ToplamaIslemi...
```

Çıktı

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        ToplamaIslemi tt = new ToplamaIslemi();
        tt.Topla(3, 4);

        tt.Topla("murat", 4);

        tt.Topla(3, "Taşyürek");

        tt.Topla("Murat", "Taşyürek");
    }
}
2 references
public class ToplamaIslemi...
```



Ödev-Kredi kartı

- Elimizde bir kredi kartımızın olduğunu ve bunun yapılan işlemlere göre alt kartlarının tanımalanbildiğini varsayalım.
- Ulaşım, eğlence, giyim ve yemek harcamları için toplamda 4 adet sanal kart tanımlı olduğumuzu düşünelim. Bu türe girmeyen harcamalar ise ana kart üzerinden yapılmaktadır.
- Yapılan harcama hangi türde ise o sanal kart üzerinden harcama yapmaktadır.
- Kartın limiti toplamda 20.000 TL'dir.
- Alt kartların limitleride ise her biri için 3.500 TL'dir.
- Alt kartlar 800 TL'ye kadar limit aşımına imkan vermektedir. Ancak an kartta limit var ise harcama yapılmalıdır.
- Bir aylık süre sonunda her bir kartta en az 5 harcama ile yapılan alış veriş ektresini nesne yönelimli programlama ile gösteriniz.
- 13 Kasım 06.59'a kadar gönderilenler 8+2 puan, vize tarihine kadar gönderenler 8 puan (kayubmprogramlama1@gmail.com).