



Object Oriented Programming

INHERITANCE

Son i am
Base Class



father

Dad i am
Derive Class



child

Inheritance is the process of acquiring properties and behaviors from one object to another object.

```
class child : father
```

Inheritance is the process by which objects of one class acquired the properties of objects of another classes.

Allows programmers to create new classes based on an existing class.

Methods and attributes from the parent class are inherited by the newly-created class

New methods and attributes can be created in the new class, but don't affect the parent class's definition

It supports the concept of **hierarchical classification** : each derived class shares common characteristics with the class from which it is derived

A subclass is also called a **derived class** and the class from which it is derived (parent class) is called **superclass** or **base class**.

- **Derived Class** (child) - the class that inherits from another class
- **Base Class** (parent) - the class being inherited from

C# and .NET support *single inheritance* only. That is, a class can only inherit from a single class. However, inheritance is *transitive*, which allows you to define an inheritance hierarchy for a set of types. In other words, type D can inherit from type C, which inherits from type B, which inherits from the base class type A. Because inheritance is transitive, the members of type A are available to type D.

In inheritance, we derive a new class from the existing class.

the new class acquires the properties and behaviors from the existing class.

The parent class is also known as **base class** or **super class**.

The child class is also known as derived class or sub class.

To inherit from a class,

use the symbol **:**

class child : mother



Inheritance

- Inheritance allows child classes to inherit the characteristics of existing parent class
 - Attributes (fields and properties)
 - Operations (methods)
- Child class can extend the parent class
 - Add new fields and methods
 - Redefine methods (modify existing behavior)
- A class can implement an interface by providing implementation for all its methods

A class can be derived from more than one classes, which means it can inherit data and functions from **multiple** base classes.

Class hierarchy

The parent-child relationship between classes can be represented in a hierarchical view often called **class tree view**.

The class tree view starts with a general class called **superclass** (sometimes referred to as **base class**, **parent class**, **ancestor class**, **mother class** or **father class**),

Derived classes (**child class** or **subclass**).

Single Inheritance

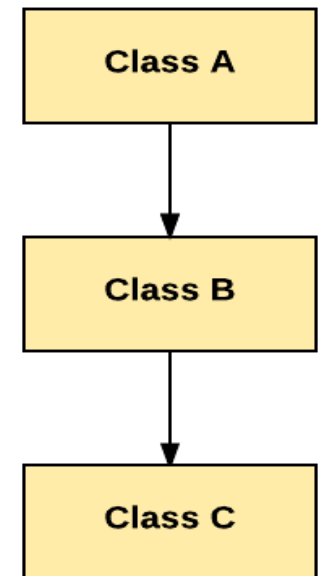
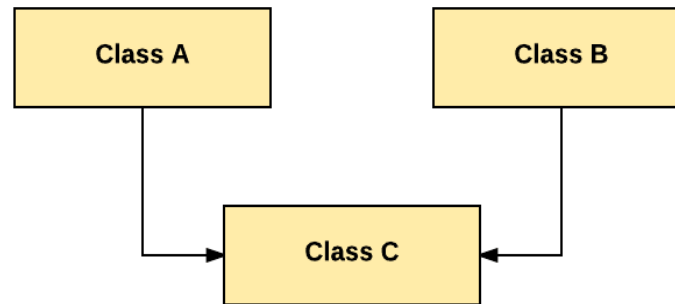
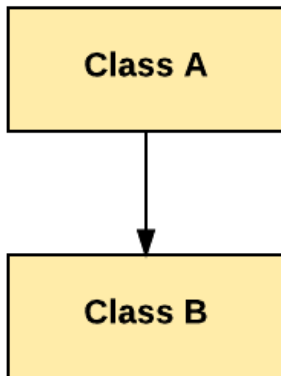
In Single Inheritance one class extends another class (one class only).

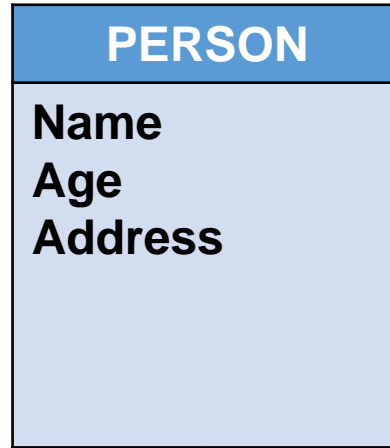
Multiple inheritance

Some object oriented languages, such as C++ allow multiple inheritance, meaning that one class can inherit attributes from two superclasses. This method can be used to group attributes and methods from several classes into one single class.

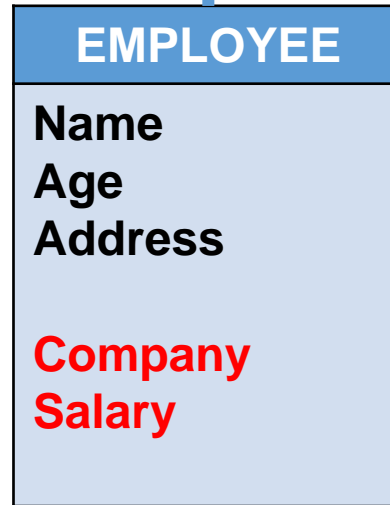
Hierarchical Inheritance

In Hierarchical Inheritance, one class is inherited by many sub classes. Class B, C, and D inherit the same class A.





Base Class



Derived Class



```
using System;  
namespace inheritance
```

```
{  
class Program
```

```
{  
class person  
{  
string name;  
string address;  
}
```

```
class student : person  
{  
string school;  
string teacher;  
}
```

```
static void Main(string[] args)  
{  
student stu = new student();  
employee emp = new employee();  
  
stu.name="Ahmad";  
stu.address = "Baghdad";  
stu.school = "Mustansiria";  
stu.teacher = "Mohammad";  
  
emp.name = "Ali";  
emp.address = "Basra";  
emp.company = "Top";  
emp.salary = 1000;  
  
Console.ReadLine();  
}  
}  
}
```

(in Inheritance) : The members (attributes) of the class should be **protected**

so they can be accessed within that class or its subclass.

```
using System;  
namespace inheritance
```

```
{  
class Program  
{  
    class person  
    {  
        protected string name;  
        protected string address;  
    }  
}
```

```
class student : person  
{  
    private string school;  
    private string teacher;  
}
```

```
class employee : person  
{  
    private string company;  
    private double salary;  
}
```

```
static void Main(string[] args)  
{  
    student stu = new student();  
    employee emp = new employee();  
  
    stu.name="Ahmad";  
    stu.address = "Baghdad";  
    stu.school = "Mustansiria";  
    stu.teacher = "Mohammad";  
  
    emp.name = "Ali";  
    emp.address = "Basra";  
    emp.company = "Top";  
    emp.salary = 1000;  
  
    Console.ReadLine();  
}
```

```
using System;
namespace inheritance
{
    class Program
    {
```

```
        class student
        {
            protected string name;
            protected int age;
            public void readinfo()
            {
                name = "Ahmad";
                age = 22;
            }
        }
    }
```

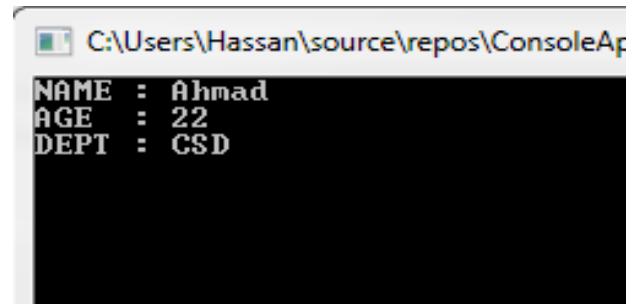
```
        class person : student
        {
            private string dept="CSD";
            public void printinfo()
            {
                Console.WriteLine("NAME : " + name);
                Console.WriteLine("AGE : " + age);
                Console.WriteLine("DEPT : " + dept);
            }
        }
    }
```

```
        static void Main(string[] args)
        {
            person s = new person();

            s.readinfo();

            s.printinfo();

            Console.ReadLine();
        }
    }
}
```



```
C:\Users\Hassan\source\repos\ConsoleAp
NAME : Ahmad
AGE : 22
DEPT : CSD
```

(Association) : attributes and methods as you need

```
using System;  
namespace inheritance  
{  
    class Program  
    {
```

```
        class student  
        {  
            private string name;  
            private string school;  
            private string teacher;  
        }
```

```
        class employee : student  
        {  
            private string name;  
            private string company;  
            private double salary;  
        }
```

```
        private string name;  
        private string school;  
        private string teacher;  
  
        private string name;  
        private string company;  
        private double salary;
```

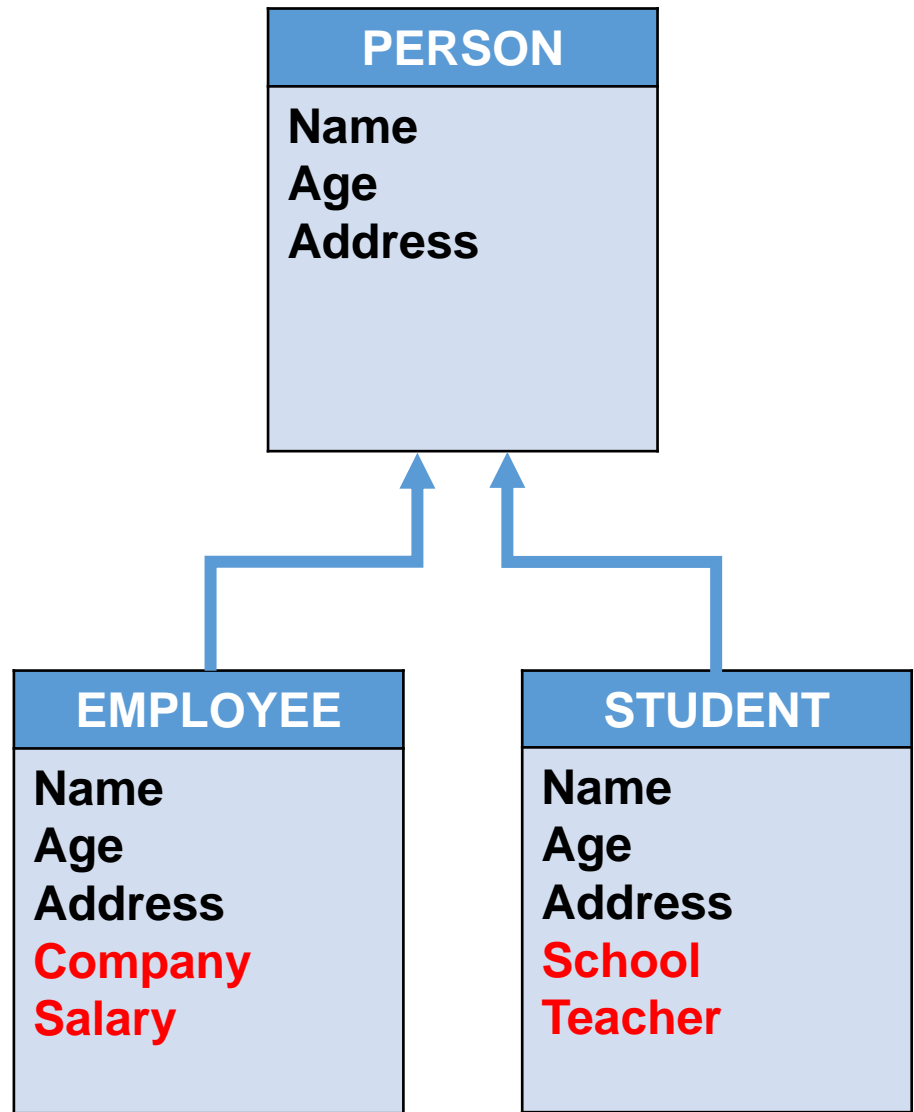
```
using System;  
namespace inheritance
```

```
{  
  class Program
```

```
{  
  class person  
  {  
    string name;  
    string address;  
  }  
}
```

```
class student : person  
{  
  string school;  
  string teacher;  
}
```

```
class employee : person  
{  
  string company;  
  double salary;  
}
```



The **sealed** Keyword

If you don't want other classes to inherit from a class, use the **sealed** keyword:

If you try to access a **sealed** class, C# will generate an error
sealed keyword is used to **restrict** a class from being derived.

We can also use **sealed** keyword with methods to prevent them for being overridden.

```
class Vehicle  
{  
    ...  
}
```

```
class Car : Vehicle  
{  
    ...  
}
```

```
sealed class Vehicle  
{  
    ...  
}
```

```
class Car : Vehicle  
{  
    ...  
}
```



```
using System;
namespace inheritance
{
    class Program
```

```
        sealed class student
        {
            protected string name;
            protected int age;
            public void readinfo()
            {
                name = "Ahmad";
                age = 22;
            }
        }
```



```
        class person : student
        {
            private string dept="CSD";
            public void printinfo()
            {
                Console.WriteLine("NAME : " + name);
                Console.WriteLine("AGE : " + age);
                Console.WriteLine("DEPT : " + dept);
            }
        }
```

```
        static void Main(string[] args)
        {
            person s = new person();

            s.readinfo();

            s.printinfo();

            Console.ReadLine();
        }
    }
```

ERROR:
cannot derive from sealed type

Why And When To Use "Inheritance"?

It is useful for **code reusability**:

reuse fields and methods of an existing class when you create a new class.

```
using System;
namespace inheritance
{
    class Program
    {
        class student
        {
            protected string name;
            protected int age;
            public void readinfo()
            {
                name = console.readline();
                age = console.readline();
            }
        }
    }
}
```

```
class person : student
{
    private string dept="CSD";
    public void printinfo()
    {
        Console.WriteLine("NAME : " + name);
        Console.WriteLine("AGE : " + age);
        Console.WriteLine("DEPT : " + dept);
    }
}
```

```
string name;
int age;
void readinfo()
{
    name = console.readline();
    age = console.readline();
}

string dept="CSD";
void printinfo()
{
    Console.WriteLine("NAME : " + name);
    Console.WriteLine("AGE : " + age);
    Console.WriteLine("DEPT : " + dept);
}
```

QUESTION



Google Classroom :

OOP 2020-2021

البرمجة الكيانية - المرحلة الثانية مسائي - د. حسن قاسم

5riqxy7

