Inheritance

1. Inheritance

- **The process of object A getting the properties from object B is called A inherits B.**
  - ⇨ **Object B belongs to a Super class or Base Class.**
  - ⇨ **Object A belongs to Subclass or Derived Class**
- **Derived**
  - ⇨ **The class derived from its Base class.**

Inheritance

Super class

Object name: Car

Property: fuel/ weight/ speed

Method: accelerate/ brake

Inheritance

Subclass

Object name: RaceCar

Property: fuel/ weight/ speed

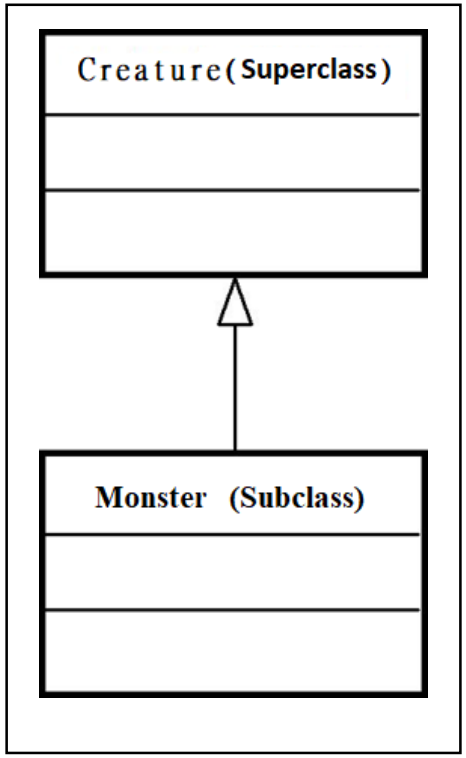
Method: accelerate/ brake/ Turbo

Inheritance and Class Diagram: Description

- Class inheritance allows us to establish a Class Hierarchy.
- In fact, inheritance is the reuse of objects. When a category is defined, other classes can inherit the data and methods of this class to add or replace the data and methods of inherited class.

Inheritance and Class Diagram

- If a class inherits from another class, it is called "Subclass“; and the inherited class is called "Superclass".
- For example, the class Monster inherits from the class Creature. Their inheritance relationship is shown in the right figure:



1-1 Class inheritance: Superclass

- C# language does not support multiple inheritance, that is, one class can have only one superclass, and cannot have multiple superclasses. The main purpose of inheritance is to expand the functionality of existing classes.

Class inheritance: Subclass(Syntax)

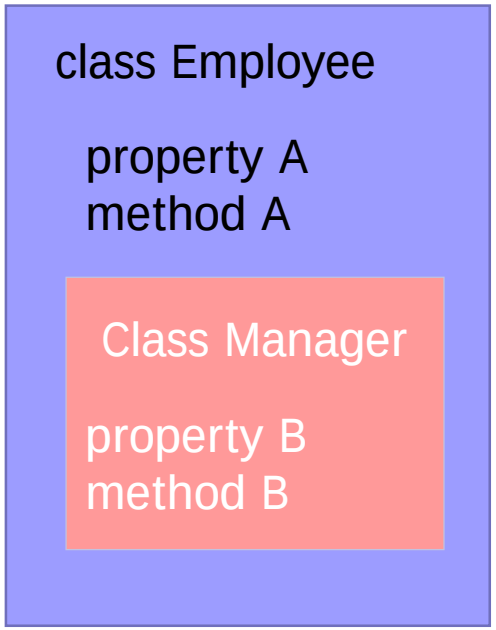
- In C# language you can create a new class to inherit class that already exists. Basic syntax is as follows:

```
Access modifier Subclass : Superclass
{
  // new attributes, fields, and methods
}
```

The symbol ":" declared in the above class is followed by the superclass.

Class Employee includes property A and method A. Class Manager includes property B and method B. Class Manager inherits from class employee, and they are defined as the follows:

```
//define class Employee
Class Employee
{
    property A
    method A
}
//Class Manager inherits from class Employee
Class Manager : Employee
{
    property B
    method B
}
```



2 Class inheritance: Access to Members of the Superclass

- **Subclass has no access to the private members that has been declared by its superclass; while others can be accessed.**

How to create properties

There are two ways to create properties:

1. **Directly declare public variables in class**
2. **Use get and set accessors**

How to use public variables to create object properties

Example: ConsoleProperty1

```
01 namespace ConsoleProperty1
02 {
03     class Car
04     {
05         public int Speed;        // 宣告 Speed 為 public 公用變數
06     }
07     class Program
08     {
09         static void Main(string[] args)
10         {
11             Car Benz = new Car();
12             Benz.Speed = 100;     // 物件建立之後可直接使用「.」存取該屬性
13             Console.WriteLine("Benz.Speed = {0} ",Benz.Speed);
14             Console.Read();
15         }
16     }
17 }
```

The Potential Shortcomings of Public Variables

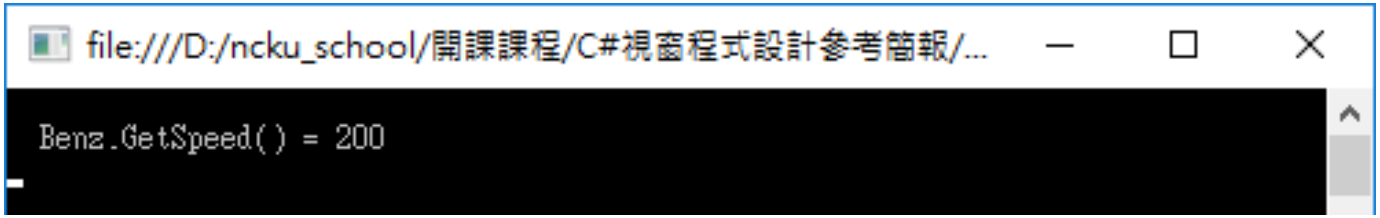
```
class Car
{
   public int Speed ;
}
class Program
{
      static void Main(string[] args)
      {
         Car Benz = new Car() ;
         Benz.Speed = 500 ;   // 屬性值超過 200
      }
}
```

How to use methods or accessors to create properties

Example: ConsoleAccessor1

```
01 namespace ConsoleAccseesor1
02 {
03    class Car
04    {
05     // 宣告_speed為私有變數，表示該變數只能在Car類別內使用
06     private int _speed;
07     // 定義GetSpeed()方法用來傳回_speed
08     public int GetSpeed()
09     {
10        return _speed;
11     }
12     // 定義SetSpeed()方法用來設定_speed
13     public void SetSpeed(int vSpeed)
14     {
15        if (vSpeed < 0) vSpeed = 0;              // 設定速度不得低於 0
16        if (vSpeed > 200) vSpeed = 200; // 設定速度不得高於 200
17            _speed = vSpeed;
18     }
```

```
20  class Program
21  {
22    static void Main(string[] args)
23    {
24        Car Benz = new Car();
25        Benz.SetSpeed(500);  // 速度值超過 200
26        // 顯示速度最大值 200
27        Console.WriteLine("Benz.GetSpeed() = {0}",Benz.GetSpeed());
27        Console.Read();
28    }
29  }
30 }
```



How to use methods or accessors to create properties

- C# provides get and set accessors to define properties.
- Get accessor acquires the property value of an object; set accessor set the property value of an object.
- Use get to get the _speed property value; Use set to set the _speed property value between 0 to 200.

Syntax:

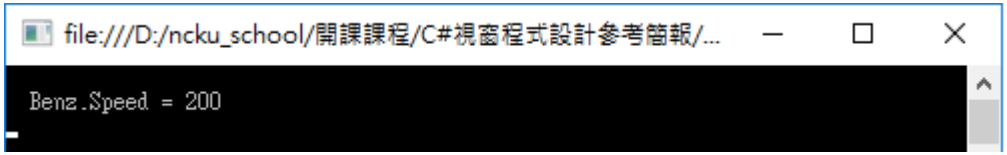
```
public int Speed
{
    get
    {
        return _speed;
    }
    set
    {
        if (value < 0) value = 0;          // 速度不得低於 0
        if (value > 200) value = 200;      // 速度不得高於 200
        _speed = value ;                   // 設定屬性值
    }
}
```

How to use the get and set accessors statement to modify the results of the previous example:

Example: ConsoleProperty2

```
class Car        // 定義Car類別
 {
        // 宣告_speed為私有變數，表示該變數只能在Car類別內使用
        private int _speed;
         // 宣告Speed為公開屬性
         2 個參考
         public int Speed
          {
                  get     // get存取子可傳回屬性值
                   {
                        return _speed;            // 傳回屬性值
                   }
                   set     // set存取子可設定屬性值
                   {
                          if (value< 0) value = 0;   // 速度不得低於 0
                          if (value > 200) value = 200;  // 速度不得高於 200
                         _speed = value;          // 設定屬性值
                    }
            }
 }
```

```
class Program
{
    0 個參考
    static void Main(string[] args)
    {
        Car Benz = new Car();
        Benz.Speed = 500;          // 速度值超過 200
        Console.WriteLine("\n  Benz.Speed = {0}", Benz.Speed);
        Console.Read();
    }
}
```



How To Create Read-Only properties

- **Create read-only properties**

  ⇨ Add no code to the set {...} section of the Speed property above.

  ⇨ Standard prctice:

  There can only the get {...} section in the property definition. set {...} section can not be added.

Example: ConsoleProperty3

```
03    class Car        // 定義 Car 類別
04    {
05        private int _angle = 10; // 私有_angle 變數初值為 10
06        public int Angle         // 定義 Angle 唯讀屬性
07        {                        // Angle 屬性只有 get 區段沒有 set 區段
08            get
09            {
10                return _angle;
11            }
12        }
13    }
```

```
14  class Program
15  {
16    static void Main(string[] args)
17    {
18        Car Benz = new Car();
19        Console.WriteLine("Benz.Angle = {0}",Benz.Angle); //Angle 能讀不能寫
20        Console.Read();
21     }
22  }
```

```
Benz.Angle = 10
```

How To Create Write-Only properties

Write-only properties (can only be modified but can not be read) is to set {...} in the config section; get{...} section cannot be added:

Example: ConsoleProperty4

```
8   namespace ConsoleProperty4
9   {
        2 個參考
10      class Car   // 定義Car類別
11      {
12          private int _turbo = 3;
            2 個參考
13          public int Turbo        // 定義Turbo唯寫屬性
14          {                                   // Turbo屬性只有set區段沒有get區段
15              set
16              {
17                  _turbo = value;
18              }
19          }
20      }
21
        0 個參考
22      class Program
23      {
            0 個參考
24          static void Main(string[] args)
25          {
26              Car Benz = new Car();
27              Benz.Turbo = 5;  // Turbo屬性只能寫不能讀
28          Console.WriteLine("\n Benz.Turbo = {0}", Benz.Turbo);
29          Console.Read();
30          }
31  }
32  }
33
```

誤清單

個方案 | 1 錯誤 | 0 警告 | 0 訊息 | 組建 + IntelliSense | 搜尋錯誤清單

	程式碼	說明	專案	檔案	行	隱藏項目...
✕	CS0154	屬性或索引子 'Car.Turbo' 無法用在此內容中，因為它缺少 get 存取子。	ConsoleProperty 4	Program.cs	28	作用中

The End