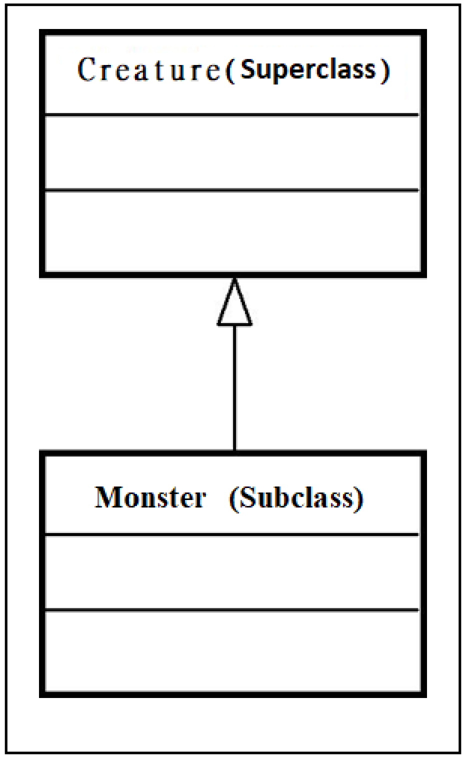
Inheritance

1. 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

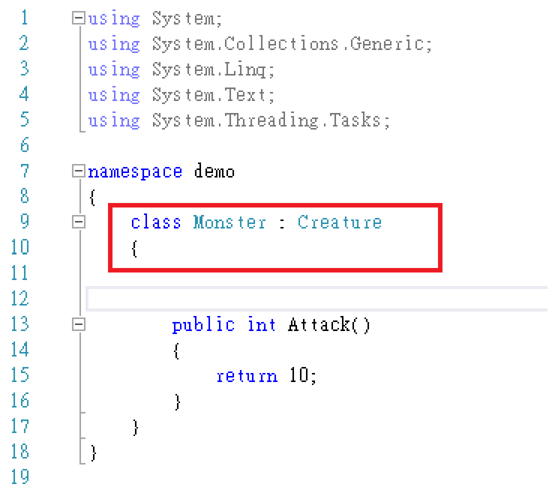
- **C# language does not support multiple inheritance, that is, one class can have only one superclass, and cannot have multiple superclasses. The main purpose of inheritance is to expand the functionality of existing classes.**

2-1 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 : Parent category {
   // new attributes, fields, and methods
}
```

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



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

namespace demo
{
    class Monster : Creature
    {


        public int Attack()
        {
            return 10;
        }
    }
}
```

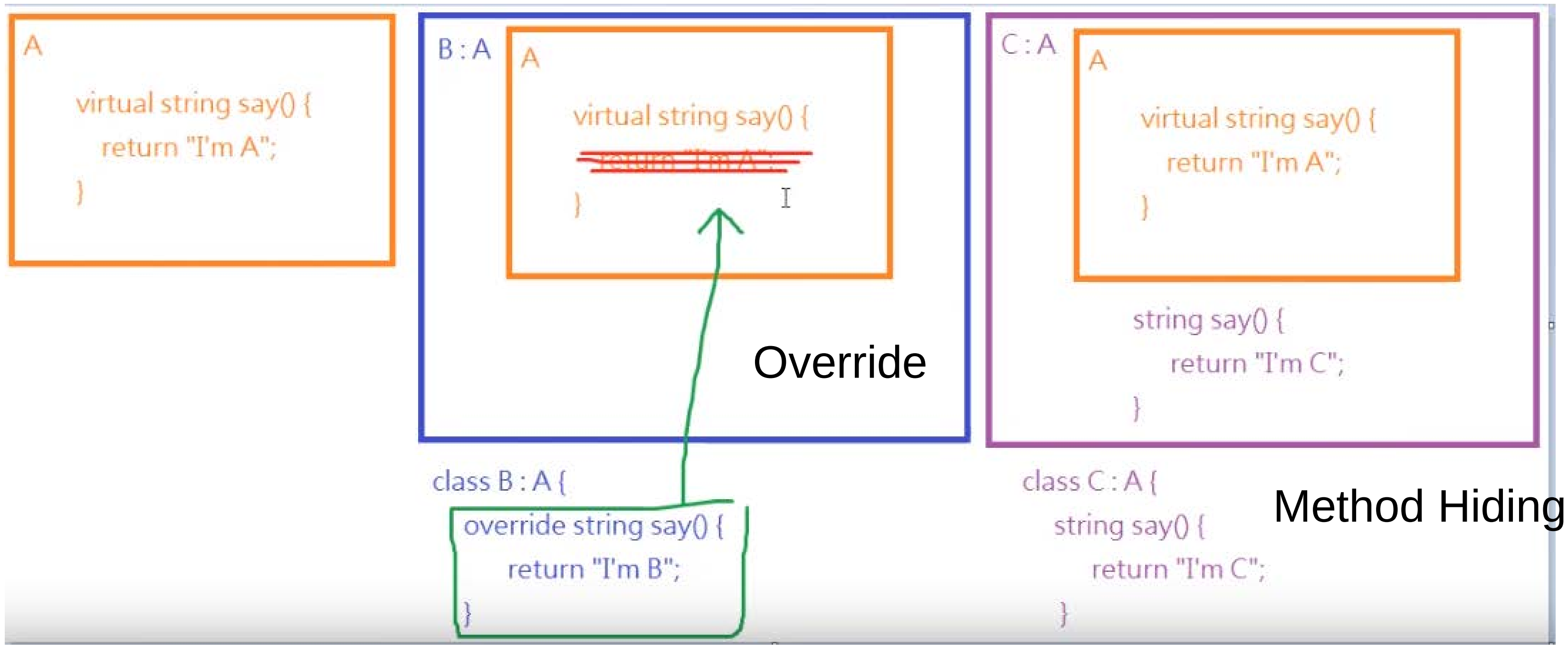
Inheritance:
Able to let objects inherit the other objects' property and behavior.

2-2 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.**

2-3 Hide or overwrite members of the superclass: Keywords new and override

- In the subclass, you can use the new or override keyword to declare a method of the same name to hide or overwrite the superclass. The description is as follows:
  - New keyword: Indicates that this method is not related to the method of the superclass and can be used to "hide" the function of the superclass.
  - Override keyword: A method that represents an object of a subclass that calls a subcategory, rather than a superclass method. It is called an Override superclass method.
- The main difference is that after using new, there are actually two functions (only the superclass is hidden). If use override, only one function actually exists.



```
virtual string say() {
    return "I'm A";
}
```

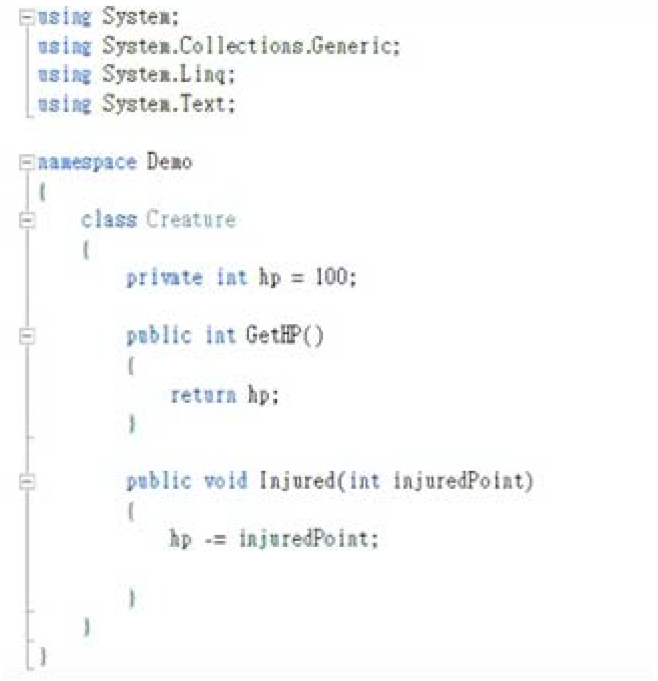
```
virtual string say() {
    ~~return "I'm A";~~
}
```

```
virtual string say() {
    return "I'm A";
}

string say() {
    return "I'm C";
}
```

```
class B : A {
  override string say() {
      return "I'm B";
  }
```

```
class C : A {
  string say() {
      return "I'm C";
  }
```



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

namespace Demo
{
    class Creature
    {
        private int hp = 100;

        public int GetHP()
        {
            return hp;
        }

        public void Injured(int injuredPoint)
        {
            hp -= injuredPoint;

        }
    }
}
```

advantage:
1. Reduce the repetition of code
2. Make it easier to maintain
3. Concise

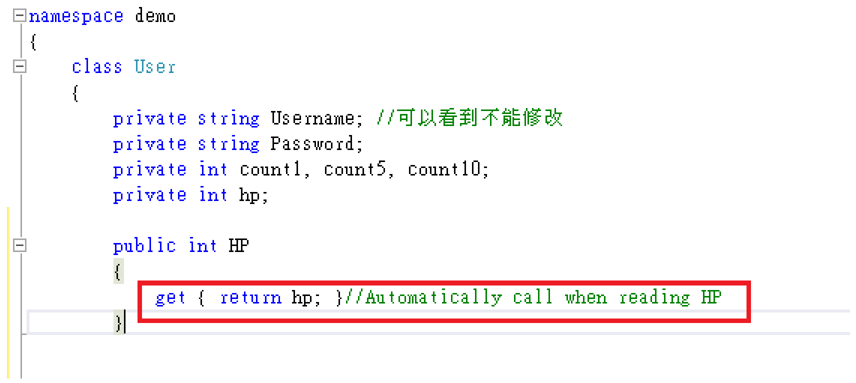
Private:
Means that property or method can only be used inside class

Situation1:
Do not want to show property content to anyone.

Situation2:
Want to set some of the property as read-only

Situation3:
Set limitation to some of the property on the setting

Get&Set

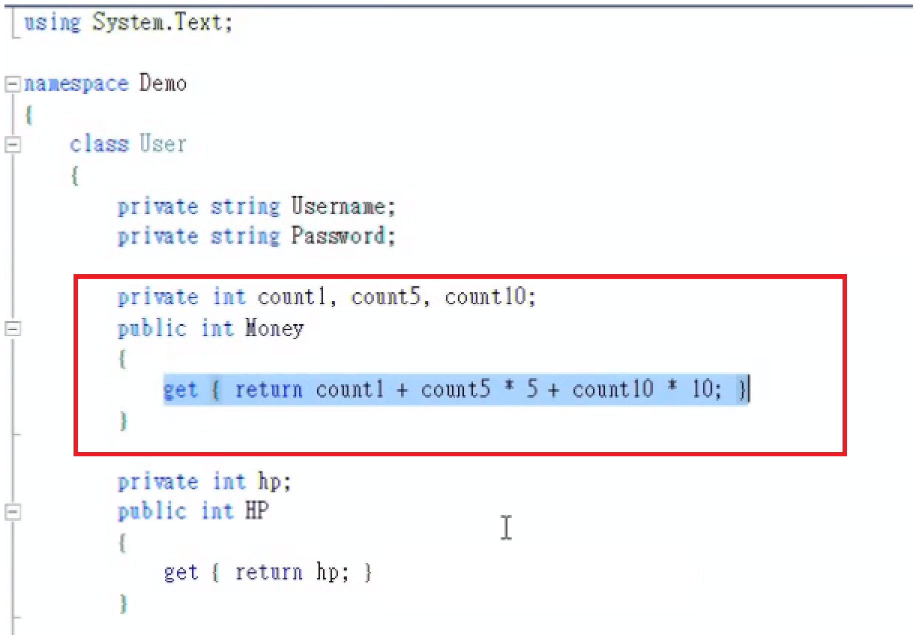


```
namespace demo
{
    class User
    {
        private string Username; //可以看到不能修改
        private string Password;
        private int count1, count5, count10;
        private int hp;

        public int HP
        {
            get { return hp; }//Automatically call when reading HP
        }
```

If we only leave get saver, the variables will become read-only

Get&Set

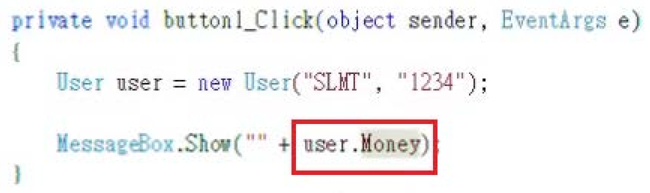


```
using System.Text;

namespace Demo
{
    class User
    {
        private string Username;
        private string Password;

        private int count1, count5, count10;
        public int Money
        {
            get { return count1 + count5 * 5 + count10 * 10; }
        }

        private int hp;
        public int HP
        {
            get { return hp; }
        }
```



```
private void button1_Click(object sender, EventArgs e)
{
    User user = new User("SLMT", "1234");

    MessageBox.Show("" + user.Money);
}
```

Get & Set saver
Be able to let some calculations show as variables.

The End