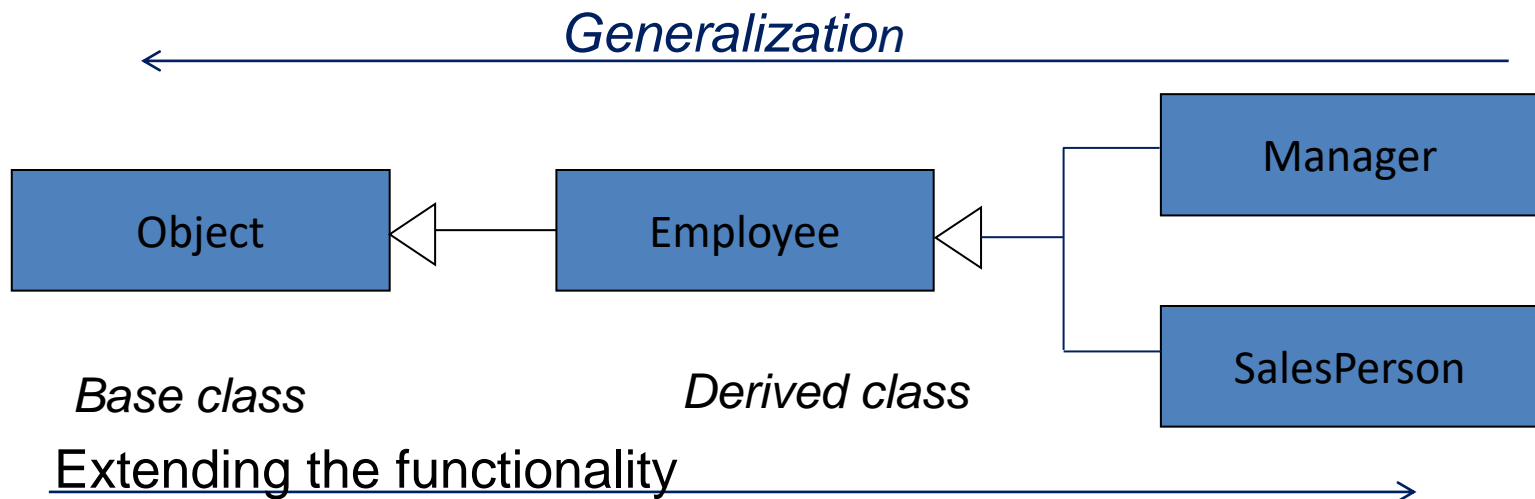


- Inheritance is an one of the three foundational features of Object Oriented Programming.
- Inheritance allows to deriving features from one class into another class.
- Inheritance leads to code reusability[write once use many times]
- The class that is inherited is called super class or base class or parent class and the class that is inheriting is called a subclass or derived class or child class.

- By default all the C# classes automatically inherit from `System.Object` class.
- Syntax: `class DerivedClass: Base class {.. }`
- The `private` members of base class is not accessible by the inherited class.
- Base class should not be less accessible than derived class.



Order of constructor and destructor call

- The simple example demonstrates the order of constructor and destructor call when creating derived class object.

```
using System;

public class Base{
    static Base() {
        Console.WriteLine("Base Static Constructor.");
    }
    public Base() {
        Console.WriteLine("Base Constructor.");    }
    ~Base()    {
        Console.WriteLine("Base destroyed");
    }
}
```

```

public class Der : Base{
    static Der() {
        Console.WriteLine("Der Static Constructor.");
    }
    public Der()    {
        Console.WriteLine("Der Constructor.");
    }
    ~Der()    {
        Console.WriteLine("Der  destroyed");
    }

    public static void Main()    {
        Der child = new Der();
    }
}

```

```

Der Static Constructor.
Base Static Constructor.
Base Constructor.
Der Constructor.
Der  destroyed
Base destroyed
Press any key to continue . . .

```

Calling base class methods

- The **base** keyword can be used to call base class methods from derived class.

```
public class Employee{  
...  
    public void print()    {  
        Console.WriteLine("ID:" + ID + " Name :" +  
Name) ;  
    }  
}
```

```
class Manager: Employee {  
    public void print()  
    {  
        base.print() ;  
        Console.WriteLine("Level " + level) ;  
    }  
}
```

- The protected members of the base class can be accessed only by the base class members as well as the derived class members.

```
public class Employee{
```

```
...
```

```
    protected void print() {
```

```
        Console.WriteLine("ID:" + id + " Name : "+name);
```

```
    }}
```

- A protected member of a base class is accessible in a derived class only if the access takes place through the derived class type.

```
class Manager : Employee{
```

```
void f(){
```

```
    Employee e = new Employee(1, "ABC");
```

```
    e.print(); // error
```

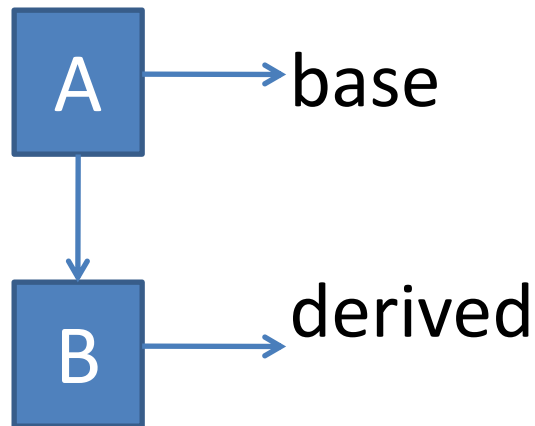
```
    Manager m = new Manager();
```

```
    m.print(); //ok
```

```
    print(); //ok
```

```
}
```

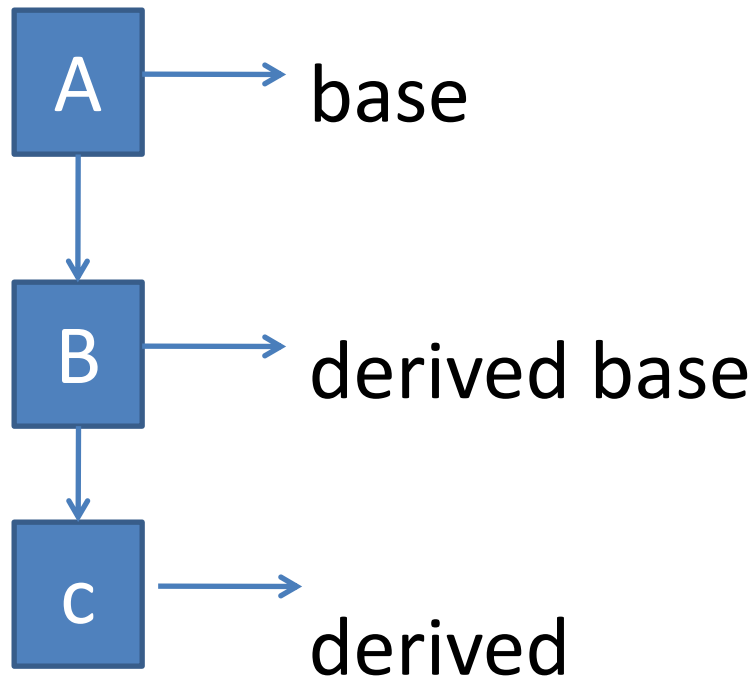
- Types:
- Single Inheritance:



```
Class A
{
}
Class B:A
{
}
```

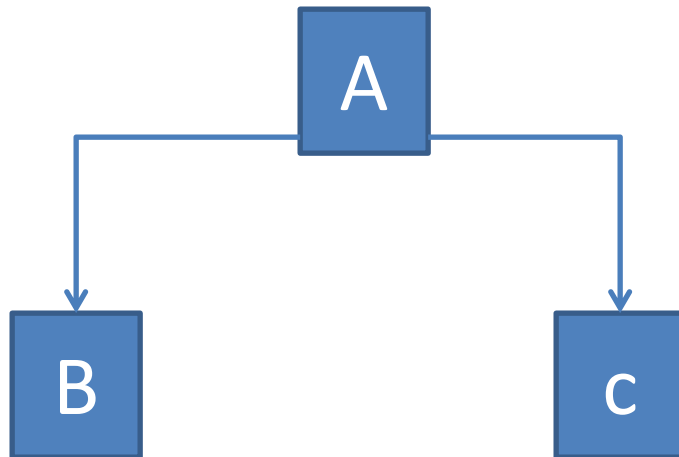
Note: In Inheritance A derived class object Can Access the members of base and derived class

Multilevel Inheritance:



```
Class A
{
}
Class B:A
{
}
Class C:B
{
}
```


Hierarchical Inheritance:



Class A

{
}

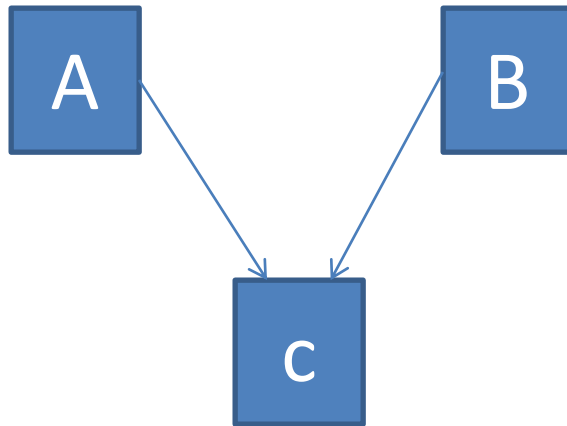
Class B:A

{
}

Class C:A

{
}

Multiple Inheritance:



C# does not support Multiple Inheritance

```
Class A
{
}
Class B
{
}
Class C:A,B
{
}
```