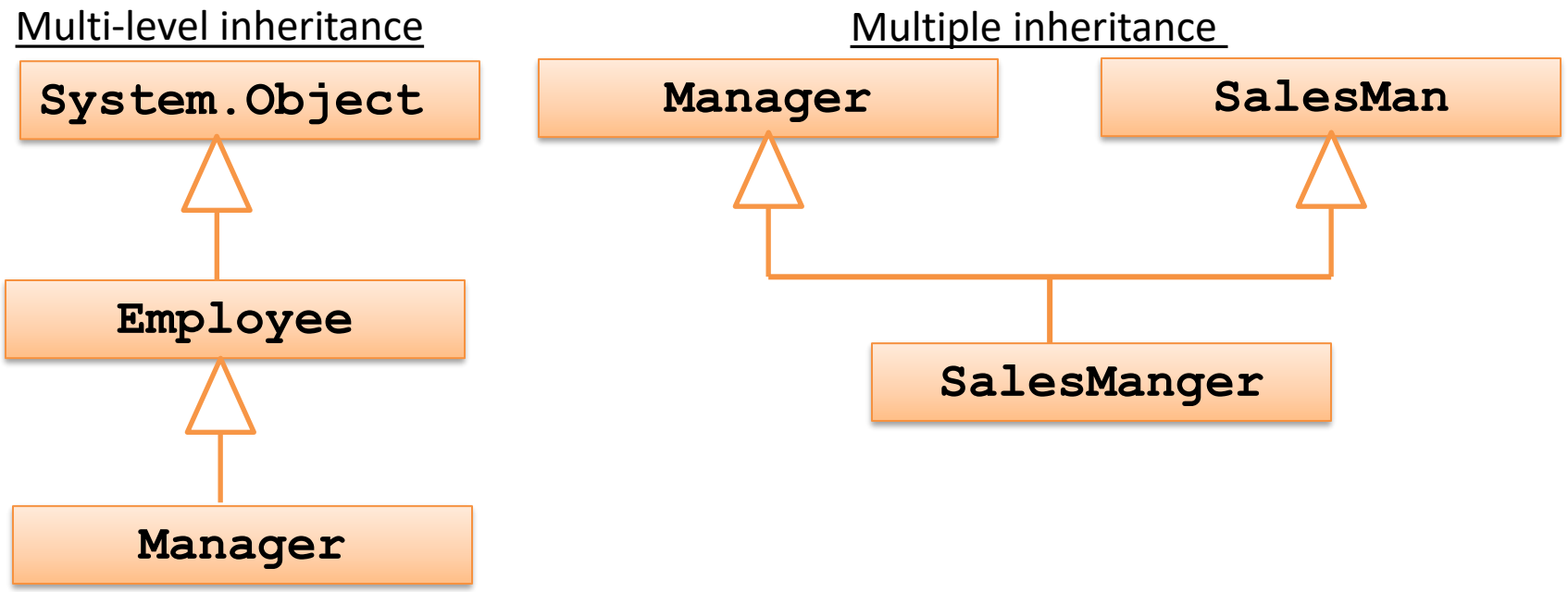
Tell me what

- What about multiple inheritance in C#?

- C# does not support multiple inheritance through classes. It supports only multi-level inheritance.

- To have a class of that can be of more than one type (which is achieved through multiple inheritance, interfaces are used.



What is an Interface ?

- An interface is a special kind of construct like class which contains just the declaration of methods (abstract methods).
- It defines a contract and any class (or struct) that implements this interface must provide implementation for all the methods declared inside the interface.
- Example of an interface that is .NET defined interfaces are **`IEnumerable`**, **`ICloneable`** etc.
- It can be a member of a namespace or a class .
- It can inherit from one or more base interfaces.
- It cannot be instantiated

Interface Members

- Interfaces can contain methods, properties, events, and indexers.
- All interface methods are implicitly **`public`** and **`abstract`**.
- An interface cannot contain constants, fields, operators, instance constructors, destructors, or types, nor can an interface contain static members of any kind.
- When class (struct) implements interfaces it is similar to inheriting from class.
- A class or struct can implement more than one interface unlike class inheritance.

Syntax

- ***modifier*** **interface** ***interface-name*** **{**

  members ***;***

}

- Modifiers allowed when the interface is declared outside a class are **public and internal**.
- Modifiers allowed when the interface is declared inside a class **new, internal, private, public, protected**
- It is a compile-time error for interface member declarations to include any modifiers.

Example: interface

```
public interface IShape
{
    void printSides(string s); // method
    int sides { get; set; } // properties
}
IShape s= new IShape ();→ ERROR!
```

Implementing interface

- A class can implement any number of interfaces.

```
public class Square:IShape
```

- If the class inherits from another class say **`Rect`** and interface as well say **`IShape,`** then syntax requires the class name to appear before the interface list.

```
public class Square:Rect,IShape
```

Interface inheritance

- An interface can inherit from zero or more interfaces, which are called the explicit base interfaces of the interface.

```
public interface Shape2D:IShape{

void draw();}
```

- An interface can declare a member with the same name or signature as an inherited member. In such case, the derived interface member is said to *hide* the base interface member.
- Hiding causes the compiler to issue a warning.
- To suppress the warning, the declaration of the derived interface member must include a **`new`** modifier to indicate that the derived member is intended to hide the base member.

Explicit Interface Implementation

- Explicit interface member implementation allows access to the interface declared method only through interface reference.
- This helps in overcoming the method name clashes if a class inherits from
  - two (or more) interfaces
  - or an interface and a class

```
interface IItem
   {
       void Add(int i);
   }

   interface IPrice
   {
       void Add(double d);
   }

   class Test : IItem, IPrice
   {
       void IItem.Add(int i)
       {
           Console.WriteLine(2 * i);
       }
       void IPrice.Add(double d)
       {
           Console.WriteLine(2 * d);
       }
       static void Main()
       {
           IItem it = new Test();
           it.Add(3);
           IPrice ip = new Test();
           ip.Add(2.3);
           Console.ReadKey();
       }
   }
```