Rules of inheritance:

Rule2: Multi-level Inheritance is permitted in java.

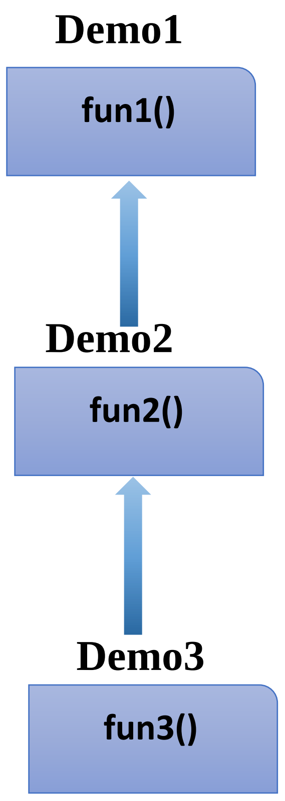
- When multiple classes are involved and their parent-child relation is formed in a chained way then such formation is known as multi-level inheritance.
- In multilevel inheritance, a parent a class has a maximum of one direct child class only.
- In multi-level inheritance, the inheritance linkage is formed in a linear way and minimum 3 classes are involved.

```
class Demo1
{
    void fun1()
    {

    }
}
class Demo2 extends Demo1
{
    void fun2()
    {

    }
}
class Demo3 extends Demo2
{
    void fun3()
    {

    }
}
```



Like any other successful tech entrepreneurs, Bill gates was a college dropout.

Rule3: Multiple Inheritance is not permitted in java as it leads to diamond shaped problem which results in ambiguity.

Multiple Inheritance is a feature of object-oriented concept, where a class can inherit properties of more than one parent class. The problem occurs when there exist methods with same signature in both the super classes and subclass. On calling the method, the compiler cannot determine which class method to be called and even on calling which class method gets the priority.

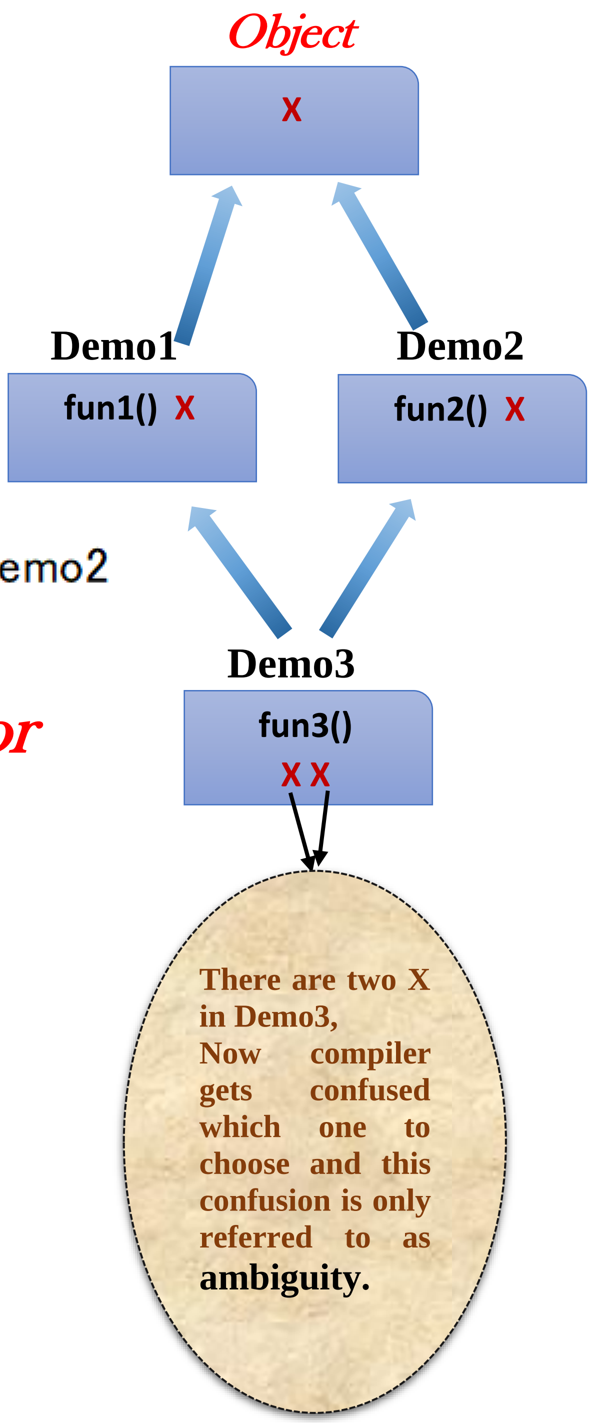
```
class Demo1   extends Object
{
    void fun1()
    {

    }
}
class Demo2   extends Object
{
    void fun2()
    {

    }
}
class Demo3 extends Demo1,Demo2
{
    void fun3()
    {

    }
}
```

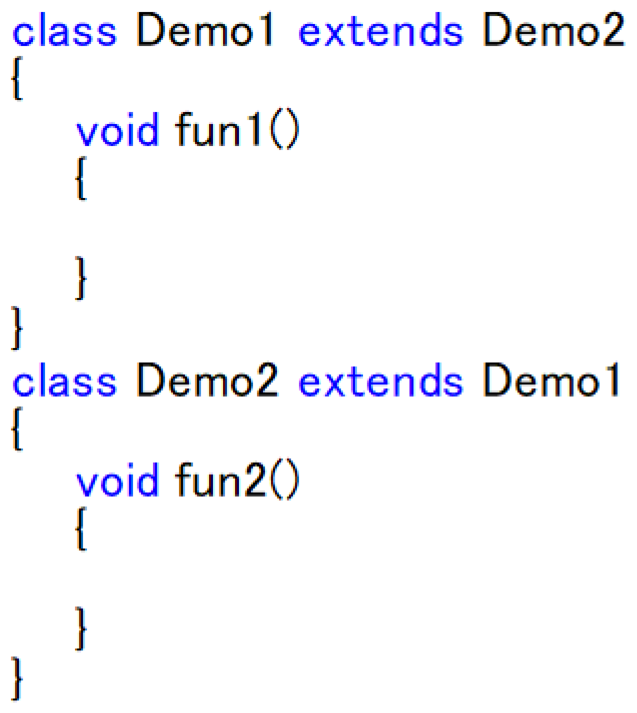
error



There are two X in Demo3, Now compiler gets confused which one to choose and this confusion is only referred to as **ambiguity.**

Rule4: Cyclic Inheritance is not permitted in.

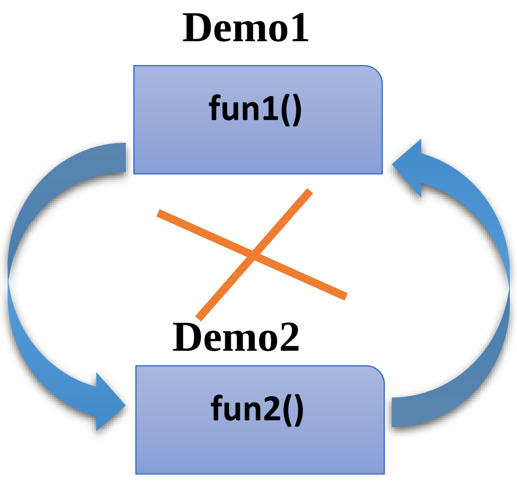
It is a type of inheritance in which a class extends itself and form a loop itself. Now think if a class extends itself or in any way, if it forms cycle within the user-defined classes, then is there any chance of extending the Object class.



```
class Demo1 extends Demo2
{
     void fun1()
     {

     }
}
class Demo2 extends Demo1
{
     void fun2()
     {

     }
}
```



Rule5: Constructors do not participate in inheritance.

In Java, constructor of base class with no argument gets automatically called in derived class constructor by a process called as **constructor chaining.**

Constructor chaining is a process of child class constructor calling its parent class constructor using **super() call.**

The first line is automatically super() call.

Irrespective of whether it is parameterized constructor or zero parameterized constructor of child class, the super() call will automatically take the control to zero parameterized constructor of parent class.

Let us understand constructor chaining in detail by considering few examples as shown below:

```
Class Object
{
        Object( )
        {

        }
}
Class Test1 extends Object
{
     int x,y;
     Test1()
     {
          super();
               x=100;
               y=200;
     }

     Test1(int x, int y)
     {
          this.x=x;
          this.y=y;
     }
}
Class Test2 extends Test1
{
     int a,b;
     Test2()
     {
          super();
          a=300;
          b=400;
     }
     Test2(int a, int b)
     {
          this.a=a;
          this.b=b;
     }
```

```
      void disp()
      {
           System.out.println(x);
           System.out.println(y);
           System.out.println(a);
           System.out.println(b);
      }
 }

Class Demo
{
     public static void main(String[ ] args)
     {
          Test2 t2 = new Test2();
          t2.disp();
     }
}
```

Output:

```
100
200
300
400
```