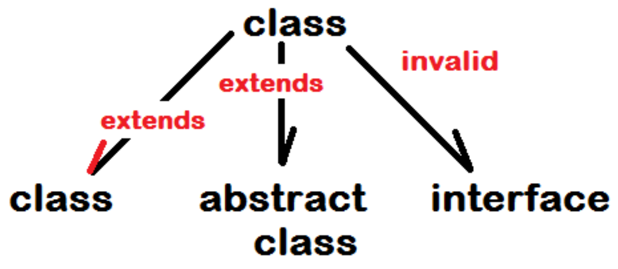
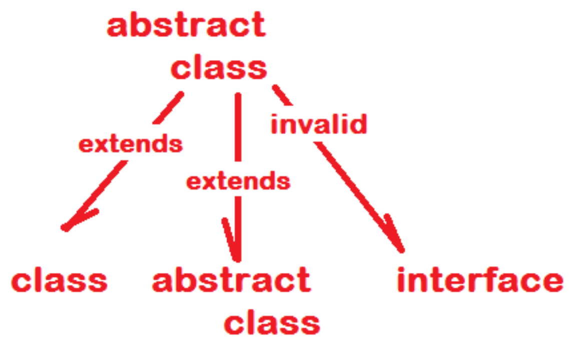
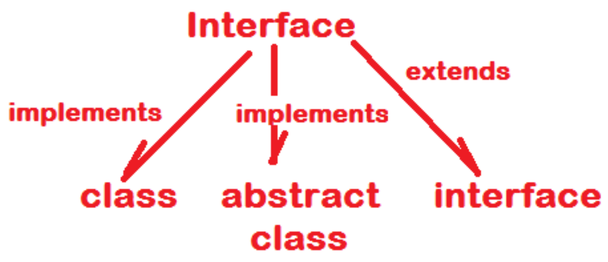






Example:

```
interface IA{
        void method1();  //public abstract
}

interface IB extends  IA{
        void method2();
}

class SubB implements IB
{  @Override
        public void method1()
        { System.out.println("OR M1 of IA "); }

        @Override
        public void method2()
        { System.out.println("OR m2 of IB "); }
}

class InterfaceDemo
{
        public static void main(String args[])
        {
                IA ia=new SubB();
                ia.method1();
                    // ia.method2(); CE

              IB ib=new SubB( );
                ib.method2();
                    ib.method1();
        }
}
```

Note:

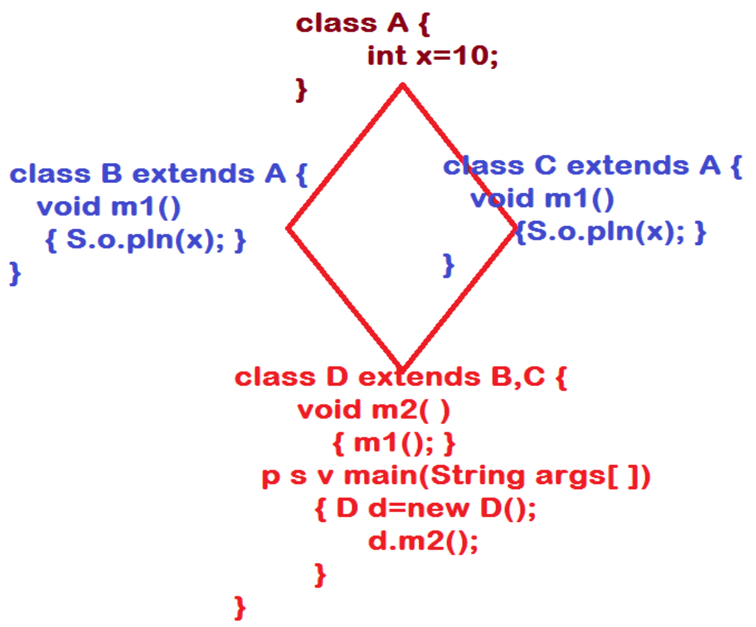
- A class can be extends by a class
- A class can't be extended by more than one class
- A class can be implemented by an interface
- A class can be implemented by more than one interface
- A class can be extends by class and can be implemented by one or more interface but in this case extends to be used first

```
class A{   }
class B{ }
interface C{ }
interface E{ }

1.class D extends A{ }  //valid
2.class D extends A,B{ }//invalid
3.class D extends A implements C{ } //valid
4.class D implements C extends A{ } // invalid
5.class D implements C extends A,B{ }//invalid
6.class D implements C,E extends A{ }//invalid
7.class D extends A implements C,E{  }//valid
```

Q:Why Java doesn't support multiple inheritance using classes .

Ans : Diamond problem



```
                    class A {
                         int x=10;
                    }

class B extends A {                  class C extends A {
   void m1()                            void m1()
    { S.o.pln(x); }                        {S.o.pln(x); }
}                                    }

                  class D extends B,C {
                       void m2( )
                         { m1(); }
                    p s v main(String args[ ])
                       { D d=new D();
                           d.m2();
                       }
                  }
```

Q: Why java doesn't support to create an object for abstract class or an interface ?

Ans : Just bcz abstract and interface will have abstract methods.

- **Abstract methods will not have body or implementation**
- **If we call abstract methods can do nothing that's way abstract methods also known as do nothing method**
- **If any class is defined with abstract methods then those classes are restricted to create an object for them**

Final Modifiers:

- **In c and c++ in order to make any variable value as constant then we have to make use of "const" keyword**
  - **const int x=10;**
    x=x+10; //Error
- **In Java there is not keyword "const" rather we have to user "final"**
  - **final int X=10;**

```
        X=X+10;  //Error
```

With the help of final modifier then we declare

1.Declare Variables
2.We can Define Methods
3.We can Define Classes

Final Methods :

- if we define any method with final modifier then those methods are not overridden ,but those are inherited.

Final Classes:

- Defining a class with "final" modifier.
- Final classes are not inherited . but we can an Object and we access the members of the classes using object possible .

Example on final Methods:

```
class Test
{
        final void method1()
         {System.out.println("M1 of Test "); }
}

class Testing extends Test
{
        public static void main(String args[])
        {
                Testing t=new Testing();
                        t.method1();
        }
}
```

Example 2:

```
class Test
{
        final void method1()
          {System.out.println("M1 of Test "); }
}
class Testing extends Test
{
  @Override
   void method1()
        { System.out.println("OR m1 of Test"); }
}
```

Example on final Class:

```
final class A
{
        int x=10;
        void method1()
        { System.out.println("M1 of A"); }
}

class B
{
        public static void main(String args[])
        {
                A a=new A();
                  System.out.println("x val is : "+a.x);
```

```
                    a.method1();
        }
}
```

Example 2:

```
final class A
{ }
class B extends A{ }
E:\Java_Online11\INTERFACE>javac FinalClass2.java
FinalClass2.java:3: error: cannot inherit from final A
class B extends A{ }
```

```
//TQ2.java
abstract class TestIQ
{
        final abstract void method1();
        abstract void method2();
}
```

Note: illegal combination of modifiers: abstract and final

```
//TQ.java
final abstract class TestIQ
{       abstract void method2(); }
```

Note: illegal combination of modifiers:
abstract and final,
abstract and static
private and abstract
protected and abstract

```
//TQ3.java
interface IA
{
        static abstract void method1();
        final abstract void method2();
}
```

```
//TQ4.java
interface IA
{
        public void method1(); //valid
        private abstract void method2(); //invalid
        protected abstract void method3(); //invalid
}
```