**First of all Remembered this point(imp point:2)**

# **1. Abstract classes can extend** other at most one **abstract** or concrete **class** and implement several interfaces.

2. Any **class** that **does** not implement all the **abstract** methods of it's super **class** has to be an **abstract class** itself.

3. A class cannot implement two interfaces that have methods with same name but different return type.

4."X extends Y" is correct if X and Y are either both classes or both interfaces

**->An interface is not extended by a class;**

->An interface can extend multiple interfaces;

5. For the interface method ; only **public and abstracts** are permitted.

**Example:**

The abstract class can also be used to provide some implementation of the [interface](https://www.javatpoint.com/interface-in-java). In such case, the end user may not be forced to override all the methods of the interface.

**interface** A{

**void** a();

**void** b();

**void** c();

**void** d();

}

**abstract** **class** B **implements** A{

**public** **void** c(){System.out.println("I am c");}

}

**class** M **extends** B{

**public** **void** a(){System.out.println("I am a");}

**public** **void** b(){System.out.println("I am b");}

**public** **void** d(){System.out.println("I am d");}

}

**class** Test5{

**public** **static** **void** main(String args[]){

A a=**new** M();

a.a();

a.b();

a.c();

a.d();

}}

Output:I am a

I am b

I am c

I am d

The abstract class can also be used to provide some implementation of the [interface](https://www.javatpoint.com/interface-in-java). In such case, the end user may not be forced to override all the methods of the interface.

**Interface:**

Why constructor is not use in interface?

Data member interface by default-**public static final ;**

Constructor is called constructor because it constructs the values at the time of object creation.it is used to initialize the variable but in interface variable is by default **public** **static** **final** so we could not change the value of final variable once it is initialize.

Difference between **Static** and **Static final(interface data member is static final).**

* Declaring variables only as static it values can be change or reinitialize by one or more instances of a class in which it is declared.
* Declaring them as static final then we create it as a **CONSTANT**. Only one copy of variable exists which can’t be reinitialize.
* If the **static** variable declared as **final**, then we have to perform initialization explicitly otherwise CT error.
* It is compulsory that we should perform initialization before class loading completion.

There are two ways to initialize static final variable.

1. We can initialize a final static variable at the time of declaration.

class MyClass {

    final static int x = 10;// Compulsary

    public static void main(String[] args)

    {

        System.out.println(x);

    }

}

2. We can also initialize a final static variable inside a static block because we should initialize a final static variable before class and we know that static block is executed before main() method.

class MyClass {

    final static int x;

    static

    {

        x = 10; // can initialize inside the static block

    }

    public static void main(String[] args)

    {

        System.out.println(x);

    }

}

**Apart from the above mentioned methods, if we try to initialize a final static variable anywhere else then we will get compile time error.**