**Abstraction in Java**

Abstraction is a process of hiding the implementation details and showing only functionality to the user.It shows only important things to the user and hides the internal details.

**Abstract class in Java**

A class that is declared with abstract keyword, is known as abstract class in java.Java abstractclasses are used to declare common chararcteristics of sub classes.It can only be used as a superclass for other classes that extend the abstract class.It can have abstract and non-abstract methods (method with body). It needs to be extended and its method implemented.We can make reference variable of abstract class,but can not create object of abstract class. It cannot be instantiated.

**abstract method**

A method that is declared as abstract and does not have implementation is known as abstract method.If a class has any abstract method whether declared or inherited, then that class must be abstract class.

**Example of abstract class that has abstract method:**

abstract class Bike

{

abstract void run();

}

class Honda56 extends Bike

{

void run()

{

System.out.println("running safely..");

}

public static void main(String args[])

{

Bike obj = new Honda56();

obj.run();

}

}

**Another Example**

abstract class Shape{

abstract void draw();

}

class Rectangle extends Shape

{

void draw()

{

System.out.println("drawing rectangle");

}

}

class Circle1 extends Shape{

void draw()

{

System.out.println("drawing circle");

}

}

class TestAbstraction1{

public static void main(String args[]){

Shape s=new Circle1();

s.draw();

}

}

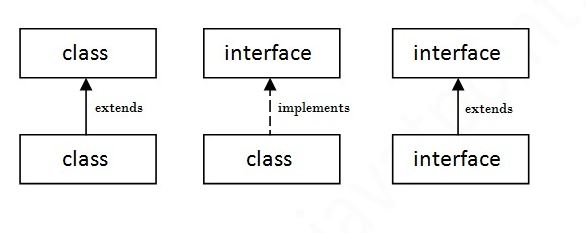
================================================================================================

**Interface in Java**

An interface in java is a blueprint of a class. It has static constants and abstract methods.There can be only abstract methods in the java interface not method body. It is used to achieve abstraction and multiple inheritance in Java. Interface do not have constructors.

**It adds public, static and final keywords before data members.**

**Relationship between classes and interfaces**



interface printable{

void print();

}

class A6 implements printable{

public void print()

{

System.out.println("Hello");

}

public static void main(String args[]){

A6 obj = new A6();

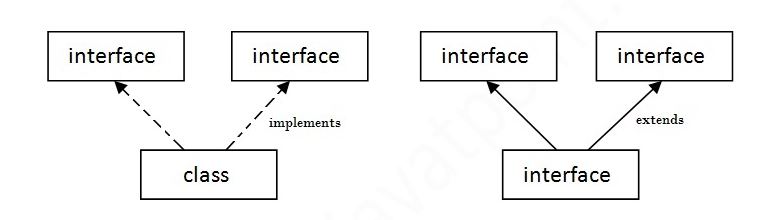
obj.print();

}

}

====================================================================================

**Multiple inheritance in Java by interface**

****

interface Printable

{

void print();

}

interface Showable

{

void show();

}

class A7 implements Printable,Showable{

public void print()

{

System.out.println("Hello");

}

public void show(){

System.out.println("Welcome");

}

public static void main(String args[]){

A7 obj = new A7();

obj.print();

obj.show();

}

}

=====================================================================================

**Interface inheritance**

interface Printable

{

void print();

}

interface Showable extends Printable

{

void show();

}

class TestInterface4 implements Showable

{

public void print()

{

System.out.println("Hello");

}

public void show()

{

System.out.println("Welcome");

}

public static void main(String args[]){

TestInterface4 obj = new TestInterface4();

obj.print();

obj.show();

}

}

====================================================================================