Interface

- We have already learnt that Java does not support multiple inheritance. That is a definition like- class A extends B extends C {…} is not permitted in Java.
- However a large number of real-life applications require the use of multiple inheritance.
- Java provides an alternate approach known as interface to support the concept of multiple inheritance.
- Although a Java class cannot be a subclass of more than one superclass, it can implement more than one interface, thereby enabling us to create class that build upon other classes without the problems created by multiple inheritance.
- An interface is basically a kind of class containing only abstract methods and final fields. This means that interfaces do not specify any code to implement, these methods and data fields contain only constants.
- Therefore it is the responsibility of the class that implements an interface to define the code for implementation of these methods.
- The general form of an interface definition is:

  interface Interfacename
  {
  variable declaration;
  method declaration;
  }
- All variables are declared as constants -

  static final type variable name;
- Methods declaration will contain only a list of methods without any body statements-

  return-type methodname (parameter list);

- **Extending interfaces-**

- Like classes interfaces can also be extended. The new subinterface will inherit all the members of the subinterface in the manner similar to subclasses. Example-

  interface name2 **extends** name1
  {
  Body o f name2
  }
- Subinterfaces cannot define the methods declared in the superinterfaces. It is the responsibility of the class that implements the derived interface to define all the methods.
- When an interface extends two or more interfaces they are separated by commas. Example- **interface** x **extends** y, z {…}
- An interface cannot extend classes. Since this would violate the rule that an interface can have only abstract methods and constants.

- **Implementing interfaces-**

- Interfaces used as *superclasses* whose properties are inherited by classes.
- Example- **class** classname **implements** interfacename
  {
  Body of the classname
  }

- A class can extend another class while implementing interfaces.
- Example- **class** classname **extends** superclass **implements** interface1,interface2
  {
  Body of the classname
  }
- When a class implements more than one interface they are separated by a comma.
- Any number of dissimilar classes can implement an interface. However, to implement the methods, we need to refer to the class objects as types of the interface rather than types of their respective classes.
- If a class that implements an interface does not implement all the methods of the interface, then the class becomes an abstract class and cannot be instantiated.
- Consider the following program implementing interface-

```
interface Area
{
  final static float pi=3.14F;
  float calculate(float a, float b);
}
class Rectangle implements Area
{
 public float calculate(float a, float b)
  {
   return(a*b);
  }
}
class Circle implements Area
{
  public float calculate(float a, float b)
  {
   return(pi*a*b);
  }
}

class InterfaceExample
{
  public static void main(String args[])
  {
   Rectangle rect= new Rectangle();
   Circle cir= new Circle();
   Area area;            //interface object
   area=rect;
   System.out.println("Area of Rectangle="+area.calculate(25,30));
   area=cir;
   System.out.println("Area of Circle="+area.calculate(25,30));
  }
}
```

❖ Consider the following program which illustrates the implementation of the concept of multiple inheritance using interfaces-

```
class Student
{
  int roll;
  void getnum(int num)
  {
    roll=n;
  }
  void putnum()
  {
    System.out.println("Roll No: "+roll);
  }
 }

class Exam extends Student
{
  float sem1,sem2;
  void getMarks(float s1, float s2)
  {
    sem1=s1;
    sem2=s2;
   }
  void putMarks()
  {
    System.out.println("Marks obtained");
    System.out.println("First Semester="+sem1);
    System.out.println("Second Semester="+sem2);
   }
 }

interface Sports
{
  final static  float sportwt=5.0F;
  void putwt();
}

class Results extends Exam implements Sports
 {
   float total;
   public void putwt()
    {
      System.out.println("Sports wt= "+sportwt);
     }
```

```
    void display()
    {
      total=sem1+sem2+sportwt;
      putnum();
      putMarks();
      putwt();
      System.out.println("Total Score= " +total);
    }
  }

class  Hybrid
{
    public static void main(String args[])
     {
        Results ob1=new Results();
        ob1.getnum(101);
        ob1.getMarks(35.5F,40.5F);
        ob1.display();
     }
 }
```