Practical – 6 Abstract class, Interface,Object class

Abstract class in Java
A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body)

- An abstract class must be declared with an abstract keyword. It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

.
Example of Abstract class that has an abstract method

```
abstract class Bike{
     abstract void run();
}
  class Honda4 extends Bike{
     void run()
          {    System.out.println("running  safely");
           }
public static void main(String args[]){
   Bike obj = new Honda4();
   obj.run()
   }
}
```

Another example of Abstract class

```
abstract class Bank{
       abstract int getRateOfInterest();
}
class SBI extends Bank{
       int getRateOfInterest(){return 7;}
}
class PNB extends Bank{
        int getRateOfInterest(){return 8;}
}
class TestBank{
```

```
    public static void main(String args[]){
      Bank b =new SBI();
    System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
    PNB b=new PNB();
      System.out.println("Rate of Interest is: "+b.getRateOfInterest()+" %");
      }
}
```

Interface

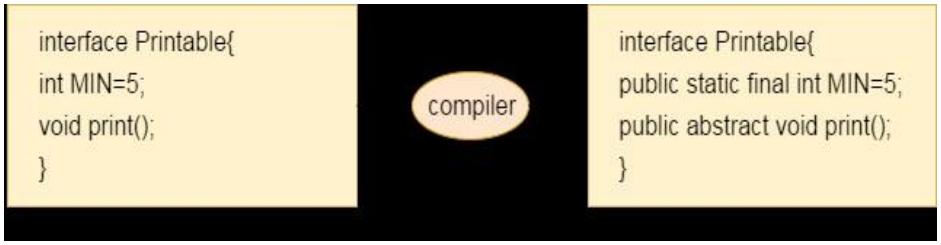
An interface in Java is a blueprint of a class. It has static constants and abstract methods

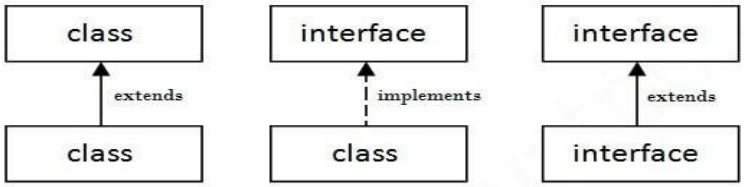
interfaces can have abstract methods and variables.

It cannot have a method body.

It cannot be instantiated just like the abstract class

- When the methods defined represent a small portion of a class When the subclass needs to inherit from another class
- When you cannot reasonably implement any of the methods





Java Interface Example

```
interface Operation
{
    public int one=1;
    public int two=2;
    int add();
     int sub();
}
```

```
class Test implements Operation
{
    int a,b;
    Test(int x,int y){
        a=x; b=y;
    }
    public int add(){
     return a+b;
    }
    public int sub(){
        return a-b;
    }
}

class InterfaceDemo
 {
    public static void main(String arg[]){
        Test t = new Test(5,10);
        int sum = t.add();
        int val = t.sub();
        System.out.println("Addition is:"+sum);
        System.out.println("Subtraction is:"+val);
        t.one =10;  // error
    }
 }
```

MutipleInterface Implementation

```
interface A {
  void showA();
}
interface B {
  void showB();
}
interface C {
  void showC();
}

public class MultiInterface implements A,B,C {
    public void showA()
    { System.out.println("Print A");}
    public void showB()
    { System.out.println("Print B");}
    public void showC()
    { System.out.println("Print C");}
    public static void main(String[] arg){
        MultiInterface obj = new MultiInterface();
        obj.showA();
        obj.showB();
```

```
    }
}
```

Interface Example 2 :

```
interface A1 {
  void showA1();
}
interface B1 extends A1{
    void showB1();
}
class Test
{
    void disp(){
        System.out.println("inside Test");
    }
    void showA1(){ System.out.println("inside Test"); }
}
public class Multioperation1 extends Test implements B1  {
    public void showA1(){
     super.showA1();
 System.out.println("Print A");}
    public void showB1()
    { System.out.println("Print B");}

    public static void main(String[] arg){
        Multioperation1 obj = new Multioperation1();
        obj.showA1();
        obj.showB1();
        obj.disp();

    }
}
```

Object Universal Class

Object is a universal superclass in java

- The Object class is the parent class of all the classes in java by default.

- public Boolean equals(Object obj)
- Object class has no data members that define its state, So, it simply performs comparison.

- The default implementation of this method in Object class simply checks if two object references x and y refer to the same object.
- i.e. It checks if x == y.

```
public class JavaObjectequalsExample1
{
   static int a =10, b=20;
   int c;
   JavaObjectequalsExample1(){
    c = a+b;
    System.out.println("Answer : " +c);
    }

    public static void main(String arg[]){
    System.out.println("first object");
    JavaObjectequalsExample1 obj1 = new JavaObjectequalsExample1();
      System.out.println("Second object");
      JavaObjectequalsExample1 obj2 = new JavaObjectequalsExample1();
      System.out.println(" obj1.equals(obj2) : " + obj1.equals(obj2));
```

```
      JavaObjectequalsExample1 obj3= obj2;
      if(obj2==obj3)
      {  System.out.println("referece same");
         System.out.println("Object are equals " + obj2.equals(obj3));
      }
```

```
    }
}
```

Output

first object
Answer : 30
Second object
Answer : 30
obj1.equals(obj2) : false
referece same
Object are equals true

- public int hashCode()

This method returns the hash code value for the object on which this method is invoked.
This method must be overridden in every class that overrides the equals method.

- public string toString() Method

The toString() method returns the String representation of the object.

If you print any object, Java compiler internally invokes the toString() method on the object.

```
//declare a class
class Password{
  //declare attributes
  private String password;
  private String retypedpassword;

  //setters and getters
  Password(String x){
    this.password = x;
  }
  public String getpassword()
  {
    return this.password;
  }
  //Override the predefined hashCode function
  @Override
  public int hashCode(){
    return (int) password.hashCode();
  }
  //Override the predefined equals function
  @Override
  public boolean equals(Object x){
    if (x == null)
      return false;
    Password y = (Password) x;
    return y.getpassword() == this.getpassword() ;
  }

}
```

```java
//declare a separate class to compare two objects
class hashes{

  public static void main(String args[])
  {
    //declare two objects
    Password p1 = new Password("ABC");
    Password p2 = new Password("DEF");
    //compare and print
    System.out.println("Hash for password 1: ");
    System.out.println(p1.hashCode());
    System.out.println("Hash for password 2: ");
    System.out.println(p2.hashCode());

    System.out.println("Equal? ");
    System.out.println(p1.equals(p2));
  }
}
```

```
Hash for password 1:
64578
Hash for password 2:
67557
Equal?
false
Press any key to continue . . .
```

Exercise

1	Create a java program for the following Create abstract class shape. Having a abstract method double area(). create class Circle inherits the shape. Take necessary properties, constructor and area() method. Create Rectengle class with necessary properties, constructor , area() method ,display(). Create class Triangle with necessary properties, constructor ,area() method. Create objects of circle,rectangle and triangle ,calculate area of each and display it. Also display other properties.
2	Create java program for the following Class Employee abstract Properties : name,age Methods : constructor, detail() , cal_salary()abstract method returns the salary. cal_bonus()abstract method return bonus. Take necessary arguments in all methods. Class Developer inherits Employee Properties : salary, bonus

	Method : constructor, cal_salary() return the salary (salary = basic + HRA+da) cal_bonus () return bonus (if salary > 25,000 bonus is 20% else bonus is 15%) display() : display all the details of Developer as well as name and age . Class Worker inherits Employee Properties : rateperhours, noofhoursworked Method: constructor cal_salary() return the salary (salary = rateperhours*noofhoursworked) cal_bonus() returns bonus (bonus=salary + 2000) display() : display all the details of worker as well as name and age . create two objects of developer and two object of worker. Display details. Calculate bonus and salary and display it.
3	Create java program for following Interface Arrayoperation Methods : boolean search() , void update(int index); Create class IntArray which implements interface ArrayOperation It should define the int array , store value into array . and implements methods of interface. Search method takes the value to be search from user and find into array. If element found then return true other wise false Update method updates the element of array of given index as parameter Create class StrArray which implements interface . Store the string value in String . Search method will take the string from user and check given string is there in source string or not Update method find the character at given index and it replace that character with @.
4	Create class Student having name, and marks properties. Create constructor to set the initial values. Override the method Boolean equals(object s) , int hashcode() and String toString() method.

	if two students having same name then equals method should return true. And it should generate same hashcode. Override toString() method which returns the name and marks as string.