**Abstract class vs Interface**

**Abstract class**

An abstract class is a class which has abstract methods. When we have an abstract method inside a class the class must be declared as abstract. We can maintain an abstract class without any abstract methods. Abstract class cannot be instantiated but they can be sub classed.

An abstract method is a method that is declared without an implementation (without braces, and followed by a semicolon), like this:

|  |
| --- |
| Public abstract class Hello{  // declare fields  // declare non abstract methods  public abstract void work();  } |

**Examples:**

Try this **……**

|  |
| --- |
| public abstract class Worker {  static int salary=2000;    public Worker(){  System.out.println("I am working in worker abstract class");  }  public abstract void work();  public void getLeave() {  System.out.println("10 days Leave");  }  public void getHike() {  System.out.println("Annually");    }  public static void main(String[] args) {  //Worker obj=new Worker(); abstract class cannot be instantiated    }  } |

|  |
| --- |
| public class Subworker extends Worker{  public Subworker(int a) {  //super  System.out.println(a);  }  public static void main(String[] args) {    Subworker obj=new Subworker(10);  obj.work();  obj.getLeave();  obj.getHike();  int a=Worker.salary=20;  System.out.println(a);  }  public void work() {  System.out.println("working");  }  } |

**Interface**

Interface is a set of rules or contract. In java we can create Interface by using interface keyword. once interface is created, we can have variables and methods inside an interface. All variables are final and static in an interface. All methods are by default abstract.

We don't need to give abstract keyword in method signature in interface. Interface is implemented using ‘Implements' keyword. A java class can implement any number of Interface. If we implement interface in a class, we should give definition for all methods.

In interface by default all methods are by default public. PRIVATE, PROTECTED modifiers are not allowed. We do not reduce visibility of the methods We cannot create constructor for interface as like as abstract class. Like abstract class we cannot create object for interface.

**Example:**

|  |
| --- |
| public interface EmployeeRules {  int salary=25000;// by default they are final and static  int leaves=10;  public void maintainHours();//by default they are abstract    public void relocate();  public void report();    } |

|  |
| --- |
| public class ABCEmployee implements EmployeeRules {  public void maintainHours() {  System.out.println("48 hours");  }  public void relocate() {  System.out.println("other city");  }  public void report() {  System.out.println("manager");  }    public static void main(String[] args) {  // TODO Auto-generated method stub  ABCEmployee abc=new ABCEmployee();  abc.maintainHours();  abc.relocate();  abc.report();    System.out.println(EmployeeRules.salary);  //salary is static so accessed by class name itself  System.out.println(EmployeeRules.leaves);  //leaves is static so accessed by class name itself  }  } |