* Abstract method is a method which is not having any body or implementation
* Eg: abstract void sum(int x,int y);
* Null body method is a method which is having body but not having implementation
  + Eg : int sum(int x,int y){ }
* Every null body method is a non abstract method,But evey non abstract method is not null body method

When to use abstract methods ?

Whenever 2 or more sub classes required to full fill same roll through different implementation

What is Abstract Method ?

Abstract method always tells about what a method can do is , but doesn’t tell about how a method a implemented

Syn: [modifiers] <abstract> <class> <ClassName>

{

Fields

Abstract Methods

Non abstract Methods

}

Note: if any class extended by an abstract class , then corresponding class must override all abstract methods of it’s super class ,Other wise the corresponding class must be defined with an abstract

* Creating an Object for an abstract class is nothing but creating an object for an of its concrete class
* Concrete class is class which is overridden all abstract methods of its superclass
* Every concrete class is subclass but every sub class is not a concrete class

abstract class Shapes{

float dim1,dim2;

void setShapes(float dim1,float dim2)

{ this.dim1=dim1; this.dim2=dim2; }

abstract float findArea();

}

class Rect extends Shapes{

float findArea()

{ return (dim1\*dim2); }

}

class Triangle extends Shapes

{ float findArea()

{ return (0.5f\*dim1\*dim2); }

}

class Abstract\_main{

public static void main(String args[]){

Shapes s=new Rect();

s.setShapes(4.0f,4.0f);

float ar=s.findArea();

System.out.println("Area of Rect : "+ar);

s=new Triangle();

s.setShapes(5.0f,5.0f);

float at=s.findArea();

System.out.println("Area of Tri : "+at);

}

}

Interface :

* Pure abstract class
* Standard and public way of defining classes
* Java doesn’t support multiple inheritance and its related classes using classes, but all types of inheritances are possible by using through interfaces

**Difference Between Class | Abstract Class | Interface ?**

|  |  |  |
| --- | --- | --- |
| **Class** | **Abstract Class** | **Interface** |
| 1.we can declare all types of fields  (static | non static | final ) | We can declare all types of fields | Only final static fields |
| 2.we can define static blocks | We can define static blocks | No static blocks |
| 3.we can define constructors | We can defined constructors | No constructors |
| 4.Defined non abstract methods | Defined non abstract methods or  Declare abstract methods | We can declare only abstract methods (Jdk1.7) |
| 5.Optional to extend | Abstract class need to be extended | Need to be implemented |
| Class can be referenced and instantiated | Abstract Class can be referenced but can’t instantiated | interface can be referenced but can’t instantiated |

* By default all the fields in the interface is “public final static “
* By default all the methods in interface is “public abstract “

Syn: [modifiers] <interface> <InterfaceName>

{

Public final static fields

Public abstract methods

}

\*\* A marker interface is an interface which is not having abstract methods | or it is an empty interface

* Java.io.Serializable
* Java.lang.Cloneable
* Java.util.RandomAccess

\*\*Functional interface is an interface which is having only abstract method

* Java.lang.Runnable
* Java.lang.Comparable

Note : if any class is implemented by an interface then all abstract methods of an interface must be overridden. Other wise the class must be define with abstract

Creating an Object of interface is nothing but creating an object of any of its implemented class

Ex:1

interface IA{

public abstract void method1(); }

class SubB implements IA{

public void method1()

{ System.out.println("OR m1 of IA"); }

public static void main(String args[])

{ IA ia=new SubB();

ia.method1(); }

}

Eg:2

interface IA{ void method1(); }

interface IB{ void method2(); }

class SubB implements IA,IB{

public void method1()

{ System.out.println("OR m1 of IA "); }

public void method2()

{System.out.println("OR m2 of IB "); }

public static void main(String args[])

{ IA ia=new SubB();

ia.method1();

// ia.method2();

IB ib=new SubB();

ib.method2();

// ib.method1();

}

}