1. Abstract class

abstract class Oper // adds flexibility

{

int a,b;

Oper(int a,int b)

{

this.a=a;

this.b=b;

}

abstract int result(); // Abstract static can't be used together As abstract method must be implemented by sub class object

}// class Oper // Abstract final can't be used together as contradictory

class Addition extends Oper

{

Addition(int a,int b)

{

super(a,b);

}

int result()

{

return (a+b);

}

}//class Addition

class Subtract extends Oper

{

Subtract(int a,int b)

{

super(a,b);

}

int result()

{

return (a-b);

}

}//clas subtract

public class OperDemo

{

public static void main(String args[])

{

// Oper obj=new Oper(); // Illegal as abstract class can't be instantiated as it is incomplete in nature.

Oper ref;

Addition obj1=new Addition(10,20);

ref=obj1;

int ad=ref.result();// delayed decision making during runtime ...dynamic method dispatch..overridden methods are called at runtime

System.out.println("Added val="+ ad);

}}

2 . Abstract class

abstract class Shape1

{

final double pi=3.14f;

abstract double volume(double x ,double y);

}

class Sphere1 extends Shape1

{

public double volume(double a,double b)

{

return 4/3\*pi\*a\*a\*a;

}

}

class Cone1 extends Shape1

{

public double volume(double a,double b)

{

return ((1.0/3)\*pi\*a\*a\*b);

}

}

class abstractShapeDemo1

{

public static void main(String s[])

{

Shape1 ref=new Sphere1();

System.out.println("volume of sphere ="+ ref.volume(3,0));

ref=new Cone1();

System.out.println("volume of Cone ="+ ref.volume(3,4));

}

}

3. Interface implemented by class

interface Shape

{

double pi=3.14f;

double volume(double x ,double y);

}

class Sphere implements Shape

{

public double volume(double a,double b)

{

return 4/3\*pi\*a\*a\*a;

}

}

class Cone implements Shape

{

public double volume(double a,double b)

{

return ((1.0/3)\*pi\*a\*a\*b);

}

}

class InterfaceShapeDemo

{

public static void main(String s[])

{

Shape ref=new Sphere();

System.out.println("volume of sphere ="+ ref.volume(3,0));

ref=new Cone();

System.out.println("volume of Cone ="+ ref.volume(3,4));

}

}

4. Interface extended

interface one

{void print1();

void print2();

}

interface two extends one

{

void print3();

}

class Three implements two

{

public void print1()

{System.out.println("..one");

}

public void print2()

{System.out.println("..two");

}

public void print3()

{System.out.println("..three");

}

}

public class InterfaceDemo

{

public static void main(String s[])

{

Three p=new Three();

p.print1();

p.print2();

p.print3();

}}

5. extending class and implementing interface

class Library

{int bookno;

void setBook(int b)

{

bookno=b;

}

void display1()

{

System.out.println("The book no is:" +bookno);

}

}

class Book extends Library

{

String author,title;

void setinfo(String a, String t)

{

author=a;title=t;

}

void display2()

{System.out.println("The Title is:" + title);

System.out.println("The author name is:" +author);

}

}

interface Language

{

String lang="ENGLISH";

void display3();

}

class Issue extends Book implements Language

{

public void display3()

{

System.out.println("The book is written in language:" +lang);

}

void display4()

{

System.out.println("Issuing book...");

display1();

display2();

}

}

class LibraryTest

{

public static void main(String s[])

{

Issue b1=new Issue();

b1.setBook(111);

b1.setinfo("R N Tagore", "The Geetanjali..");

b1.display4();

b1.display3();

}

}