**Program : 1**

class Publication

{

String title;

int price;

public Publication(String title,int price)

{

this.title=title;

this.price=price;

//System.out.println(&quot;In publication constructor&quot;);

}

void display()

{

System.out.println(&quot;In publication,display method&quot;);

System.out.println(&quot;Title: &quot;+title+&quot; Price: &quot;+price);

}

}

class Book extends Publication

{

int pages;

Book(String title,int price,int pages)

{

super(title,price);

this.pages=pages;

}

void display()

{

super.display();

System.out.println(&quot;In book class&quot;);

System.out.println(&quot;Title: &quot;+title+&quot; Price: &quot;+price+&quot; Pages:

&quot;+pages);

}

}

class CD extends Book

{

int size;

CD(String title,int price,int pages,int size)

{

super(title,price,pages);

this.size=size;

}

void display()

{

super.display();

System.out.println(&quot;In CD class&quot;);

System.out.println(&quot;Title: &quot;+title+&quot; Price: &quot;+price+&quot; Pages:

&quot;+pages+&quot; Size: &quot;+size);

}

}

public class InheritResult

{

public static void main(String args[])

{

String title=”Harry Potter Book”];

int price=5000;

int pages=100;

int size=120;

CD cd=new CD(title,price,pages,size);

cd.display();

}

}

OUTPUT :

In publication,display method

Title: Harry Potter Price: 5000

In book class

Title: Harry Potter: Price 5000 Pages: 100

In CD class

Title: Harry Potter: Price 5000 Pages: 100 Size: 120

**Program :2**

class Vehicle{

void run(){System.out.println("Vehicle is running");}

}

class Bike extends Vehicle{

public static void main(String args[]){

Bike obj = new Bike();

obj.run();

}

}

OUTPUT : Vehicle is running

**Program : 3**

interface Shape

{

double pi = 3.14;

double calc(double x,double y);

}

class triangle implements Shape

{

public double calc(double x,double y)

{

return(x\*y);

}

}

class square implements Shape

{

public double calc(double x,double y)

{

return(x\*x);

}

}

class circle implements Shape

{

public double calc(double x,double y)

{

return(pi\*x\*x);

}

}

class CalculateArea

{

public static void main(String arg[])

{

triangle t = new triangle();

circle c = new circle();

square s = new square();

Shape a;

a = t;

System.out.println("\nArea of Triangle is : " +a.calc(10,20));

a = s;

System.out.println("\nArea of Square is : " +a.calc(10,10));

a = c;

System.out.println("\nArea of Circle is : " +a.calc(15,15));

}

}

OUTPUT : Area of Triangle is : 200.0

Area of Square is : 100.0

Area of Circle is : 706.5

**Program : 4**

package p1;

public class A{

public void msg(){System.out.println("Hello");}

}

package p2;

import pack.\*;

class B{

public static void main(String args[]){

A obj = new A();

obj.msg();

**}**

**}**

**OUTPUT :** Hello

**Program : 5**

import java.util.\*;

class CharNum

{

public static void main(String args[])

{

String s=args[0];

try

{

int digit=0;

int character=0;

for(int i=0;i&lt;s.length();i++)

{

if(Character.isLetter(s.charAt(i)))

character++;

else

digit++;

}

System.out.println(&quot;no.of characters are: &quot;+character+&quot;\nno of digits

are: &quot;+digit);

}

catch(Exception e)

{

System.out.println(e);

}

}

}

OUTPUT : Abcde12345

no.of characters are: 5

no of digits are: 5

1. **What is Inheritance?**

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system).

The syntax of inheritance:

class Subclass-name extends Superclass-name

{

Methods and fields

}

1. **What is Multiple Inheritance?**

Object Oriented Programming provides a user the feature of multiple inheritance, wherein a class can inherit the properties of more than a single parent class. In simpler terms, multiple inheritance means a class extending more than one class.

1. **What is the use of Super keyword?**

The super keyword refers to the objects of immediate parent class.

The use of super keyword

* To access the data members of parent class when both parent and child class have member with same name.
* To explicitly call the no-arg and parameterized constructor of parent class.
* To access the method of parent class when child class has overridden that method.

1. **What is abstract method?**

A method without body (no implementation) is known as abstract method. A method must always be declared in an abstract class, or in other words you can say that if a class has an abstract method, it should be declared abstract as well.

1. **What is abstract class?**

A class that is declared using “**abstract**” keyword is known as abstract class. It can have abstract methods (methods without body) as well as concrete methods (regular methods with body).

An abstract class can not be **instantiated**, which means you are not allowed to create an **object** of it.

1. **What is the use of final modifier?**

The final is a modifier in Java, which can be applied to a variable, a method or a class.

When a final modifier is used with a class then the class cannot be extended further. This is one way to protect your class from being sub classed and often sensitive classes are made final due to security reason.

1. **What is interface? Write the syntax interface.**

An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

SYNTAX : interface < interface\_name > {

// declare constant fields

// declare methods that are abstract.

// by defaults.

}

1. **What is package?**

A **java package** is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.

There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

1. **What is exception?**

An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions. When an error occurs within a method, the method creates an object and hands it off to the runtime system.

1. **What is the use of finally block?**

**Java finally block** is a block that is used to execute important code such as closing connection, stream etc. Java finally block is always executed whether exception is handled or not. Java finally block follows try or catch block.