**Assignment-3**

1. **Create a class Publication with data members title(String) and price(int). From this class derive two classes Book and CD. Class Book adds pages(int) and CD adds Size(int). Each of these classes should have constructors and display(). Write a java program to implement this using super, this and method overriding concepts.**

class Publication

{

String title;

int price;

public Publication(String title,int price)

{

this.title=title;

this.price=price;

}

void display()

{

System.out.println(“Title: “+title+” Price:”+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(“In book class”);

System.out.println(“Title: “+title+”Price: “+price+” Pages:”+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(“In CD class”);

System.out.println(“Title:”+title+”Price:”+price+” Pages:”+pages+”Size:”+size);

}

}

public class InheritResult

{

public static void main(String args[])

{

String title=args[0];

int price=Integer.parseInt(args[1]);

int pages=Integer.parseInt(args[2]);

int size=Integer.parseInt(args[3]);

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

cd.display();

}

}

**Output:**

Spectrum (title)

2000 (price)

5000 (pages)

30 (size)

Title: spectrum Price: 2000

In book class

Title: spectrum Price: 2000 Pages: 5000

In CD class

Title: spectrum Price: 2000 Pages: 5000 Size: 30

**2).Write a simple java program to demonstrate method overriding.**

class vechicle{

public void move() {

System.out.println("vechicles can move from one placee to other ");

}

}

class bike extends vechicle {

public void move() {

System.out.println("bike is vechicle");

}

}

public class transport {

public static void main(String args[]) {

vechicle a = new vechice();

vechicle b = new bike();

a.move();

b.move();

}

}

**Output:**

vechicles can move from one placee to other

bike is vechicle

**3)Write a java program to create an interface called Shape with CalculateArea(). Create three classes namely Square,Circle,Triangle which implements Shape.**

**interface** Shape{

public double **CalculateArea();**

}

class **Circle** implements **Shape**

**{**

double radius;

public **Circle**(double r){

radius = r;

}

public double **CalculateArea()**{

return 3.14 \* radius \* radius;

}

**}**

class **triangle** implements **Shape**

**{**

double base;

double height;

public **triangle**(double b, double h){

base=b;

height = h;

}

public double **CalculateArea()**{

return (base \* height)/2;

}

**}**

class **square** implements **Shape**

**{**

double side;

public **square**(double s){

side = s;

}

public double **CalculateArea()**{

return side\*side;

}

**}**

class **Interfaceshape**{

public static void main(String args[]){

Circle c = new Circle(3);

System.out.println("Area of circle: " + c.CalculateArea());

triangle t = new triangle(2.0, 5.0);

System.out.println("Area of triangle: " + t.CalculateArea());

square sq = new square(3);

System.out.println("Area of square: " + sq.CalculateArea());

}

}

**Output:**

Area of circle: 28.259999999999998

Area of triangle: 5.0

Area of square: 9.0

**4).** **Create two packages p1 and p2. The package p1 contains class A which contains one display(). Create class B in package p2. The main method of class B invoke A’s display(). Write a java program to do this.**

A.java:

package p1;

public class A

{

public void display()

{

System.out.println(“I am in package p1”);

}

}

B.java:

package p2;

import p1.A;

class B

{

public static void main(String args[])

{

A a=new A();

a.display();

}

}

Output:

I am in package p1

**1. What is Inheritance?**

Inheritance can be defined as the process where one class acquires the properties of another. With the use of inheritance the information is made manageable in a hierarchical order.

The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class)

**2. What is Multiple Inheritance?**

Multiple Inheritance is a feature of object oriented concept, where a class can inherit properties of more than one parent class. The problem occurs when there exist methods with same signature in both the super classes and subclass.

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

The super keyword refers to superclass (parent) objects. It is used to call superclass methods, and to access the superclass constructor. The most common use of the super keyword is to eliminate the confusion between superclasses and subclasses that have methods with the same name.

**4. What is abstract method?**

A method without body (no implementation) is known as abstract method. Sometimes, we require just method declaration in super-classes. This can be achieve by specifying the abstract type modifier.

**5. What is abstract class?**

A class that is declared using “**abstract**” keyword is known as abstract class. It can have abstract methods as well as regular methods with body. An abstract class can not be instantiated, which means you are not allowed to create an object of it.

**6. 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 subclassed and often sensitive classes.

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

Interface looks like a class but it is not a class. An interface can have methods and variables just like the class but the methods declared in interface are by default abstract.

**Syntax:**

**interface** MyInterface

{

/\* All the methods are public abstract by default

\* As you see they have no body

\*/

public void method1();

public void method2();

}

**8. What is package?**

A package in Java is used to group related classes. We use packages to avoid name conflicts, and to write a better maintainable code. Packages are divided into two categories:

1.Built-in Packages (packages from the Java API)

2.User-defined Packages (create your own packages)

**9. What is exception?**

When an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an *exception object*, contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called throwing an exception.

**10. 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.