**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.**

**Source Code :**

import java.util.Scanner;

class Publication{

String title;

int price;

Publication(String title){

this(title,100);

}

Publication(String title, int price){

this.title=title;

this.price=price;

}

void display(){

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

System.out.println("Price : Rs."+price);

}

}

class Book extends Publication{

int pages;

Book(String title,int pages){

super(title);

this.pages=pages;

}

void display(){

super.display();

System.out.println("Pages : "+pages);

}

}

class CD extends Publication{

int size;

CD(String title,int size){

super(title);

this.size=size;

}

void display(){

super.display();

System.out.println("size : "+size+"cm");

}

}

class PublicationBook{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

System.out.print("Enter the name of the book : ");

String s=sc.nextLine();

System.out.print("Enter the price of the book : ");

int pr = sc.nextInt();

System.out.print("Enter how many pages in the book : ");

int pa = sc.nextInt();

Publication p;

p= new Book(s,pa);

p.display();

System.out.print("Enter the size of the book : ");

int si=sc.nextInt();

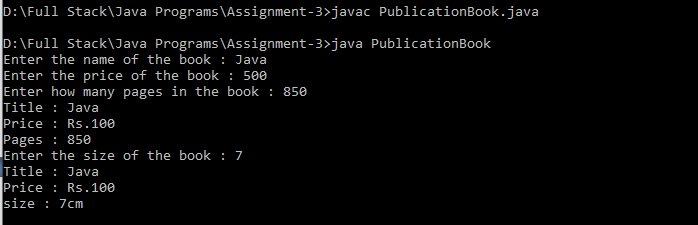
p=new CD(s,si);

p.display();

}

}

**Output :**

****

**2.Write a simple jav aprogram to demonstrate method overriding.**

**Source code :**

class A{

void sp(int a, int b){

System.out.println("Sum of two numbers is : "+(a+b));

}

}

class B extends A{

void sp(int a, int b){

System.out.println("Product of two numbers is : "+(a\*b));

}

}

class OverrideDemo{

public static void main(String args[]){

A a=new A();

B b=new B();

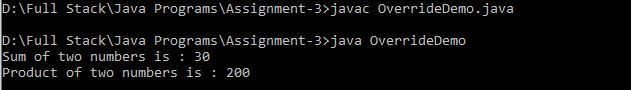
a.sp(10,20);

b.sp(10,20);

}

}

**Output :**

****

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

**Source code :**

import java.util.Scanner;

interface Shape{

public void calculateArea();

}

class Square implements Shape{

public void calculateArea(){

Scanner sc=new Scanner(System.in);

System.out.print("Enter s value : ");

int s=sc.nextInt();

System.out.println("Area of Square is : "+(s\*s));

}

}

class Circle implements Shape{

public void calculateArea(){

Scanner sc=new Scanner(System.in);

System.out.print("Enter value of radius : ");

int r=sc.nextInt();

System.out.println("Area of Circle is : "+(3.14f\*r\*r));

}

}

class Rectangle implements Shape{

public void calculateArea(){

Scanner sc=new Scanner(System.in);

System.out.print("Enter length of Rectangle : ");

int l=sc.nextInt();

System.out.print("Enter height of Rectangle : ");

int h=sc.nextInt();

System.out.println("Area of Rectangle is : "+(l\*h));

}

}

class InterfaceDemo{

public static void main(String args[]){

Square s = new Square();

Circle c = new Circle();

Rectangle re = new Rectangle();

s.calculateArea();

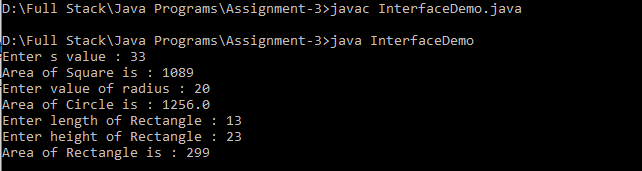
c.calculateArea();

re.calculateArea();

}

}

**Output :**



**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.**

**Source Code :**

**//package pack1**

package pack1;

public class A{

public void display(){

System.out.println("Hello! I am From class A ");

}

}

**//package pack 2**

package pack2;

import pack1.A;

public class B{

public static void main(String args[]){

A a = new A();

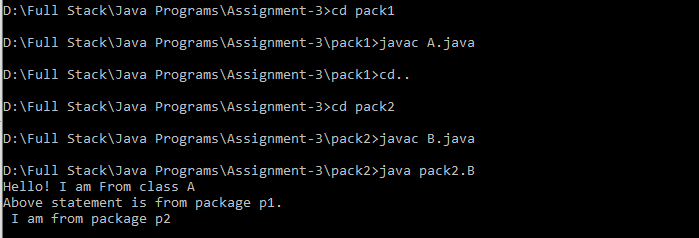
a.display();

System.out.println("Above statement is from package p1.\n I am from package p2");

}

}

**Output :**



**5.Write a java program to count numbers, characters in the command line arguments using Exception handling mechanism.**

**Source Code :**

import java.util.Scanner;

class CountNC{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

int countN=0,countC=0,countS=0;

System.out.print("Enter the sentence you want : ");

try{

String s = sc.nextLine();

for(int i=0;i<s.length();i++){

if(Character.isDigit(s.charAt(i))){

countN++;

}

else if(s.charAt(i)!=' '){

countC++;

}

else if(s.charAt(i)==' '){

countS++;

}

else{

}

}

System.out.println("Total number of character in the given sentence is : "+countC);

System.out.println("Total number of number in the given sentence is : "+countN);

System.out.println("Total number of spaces and special symbols in the given sentence is : "+countS);

}

catch(Exception e){

System.out.println("You entered invalid symbols in a sentence");

}

finally{

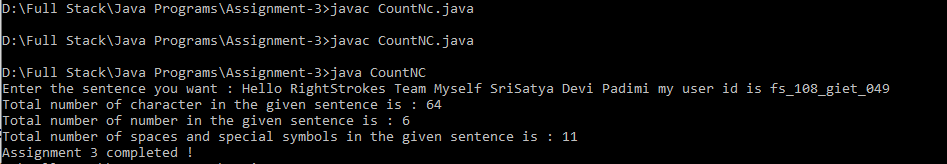
System.out.print("Assignment 3 completed ! ");

}

}

}

**Output :**



**1. What is Inheritance?**

Inheritance can be defined as the process where one class acquires the properties and behavior of other class.

A class i.e., derived from another class is called a sub class (or) derived class (or) extended class (or) child class.

The class from which the subclass is derived is called a super class (or) base class (or) parent class.

* single inheritance
* multi level inheritance
* hybrid inheritance
* hierarchical inheritance
* multiple inhertance

**2. What is Multiple Inheritance?**

Multiple Inheritance can be defined as a class can inherits the properties of more than one parent class (or) super class.

Java doesn't support multiple inheritance directly by classes. but it can achieved indirectly by the use of **interfaces.**

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

Whenever a subclass needs to refer to its immediate super class, it can be done by using super keyword.

It is used to call the super class constructors and members of the super class.

**General form :**

**super(parameters\_list);**

* Used to access super class constructor.
* Used to access super class method.
* Used to access super class variables.

**4. What is abstract method?**

An abstract method contains only method heading (or) prototype (or) declaration but not containing method body (or) definition.

**Syntax :**

abstract return\_type method\_name(parameters\_list);

**example:**

abstract int sum(int,int);

abstract void display();

**5. What is abstract class?**

An abstract class is a collection of defined methods and undefined methods.

Undefined methods are also known as abstract methods (or) unimplemented methods.

**Syntax :**

abstract class <class\_name>{

.........

.........

abstract return\_type method\_name(parameters\_list);

.........

.........

}

**Example :**

abstract class operations{

........

........

abstract void sum(int,int);

abstract int mul(int,int);

abstract void display();

........

}

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

The final modifier is used to prevent the operation as follows :

* A class declared as final it cannot be instantiated.
* A variable declared as final it cannot be reassigned.
* A method declared as final it cannot be overridden.

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

Interface is similar to class. It contains collection of abstract methods and static final constants. we will not create objects for any interface, we can implements an interface.

* An interface does not contain any constructors.
* All of the methods in an interface are abstract.
* An interface cannot contain instance fields. The only fields that can appear in ana interface must be declared both static and final.
* An interface is not extended by a class but it is implemented by a class.

**Syntax :**

Access modifier(public/no modifier) interface interface\_name{

return\_type method-name1(parameter-list);

return\_type method-name2(parameter-list);

type final-variablename1 = value;

type final-variablename2 = value;

.....

......

return\_type method-nameN(parameter-list);

type final-variablenameN= value;

}

**Example :**

public interface Sample{

public void display();

public void sun(int a,int b);

final static int s=0;

}

**8. What is package?**

Package is a group of similar types of classes , interfaces and sub-packages.

Packages are a way of grouping a variety of classes together. The gropind is done according to their functionality. Packages acts as "**Containers "** for classes.

package can be categorized in two forms :

1.built-in packages

2.user-defined packages

**9. What is exception?**

An exception is an object occurs at run time which will describe nature of the message. The nature of the message can be either system error message (or) user friendly error message.

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

Finally is default catch handler. It will be executed whether an exception is thrown or not.

If an exception is thrown, the finally block will execute even if no catch statement matches the exception.