**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**B. P. PODDAR INSTITUTE OF MANAGEMENT & TECHNOLOGY**

**MAKAUT INTERNAL LAB EXAMINATION QUESTION PAPER ODD SEMESTER 2022-23**

**OBJECT ORIENTED PROGRAMMING LAB (PCC-CS593)**

1. Create a class Employee having emp\_id,name,salary,designation.

Use constructor overloading for designing 3 types of employees--

(a) Freshers (name should be user given and other parameters should be fixed)

(b) Executive (Name , salary and designation should be user given)

(c) Temporary member(All the parameter having fixed values)

Create an employee array and create and store 3 Employee objects and print the detail.

CODE :

import java.util.\*;

class Employee

{

private int empid;

private String name,designation;

private double salary;

// Freshers

Employee(String n)

{

name=n;

empid=51;

salary=55000.00;

designation="Freshers";

}

Employee(String n, int i, double s, String d)

{

//Executive

name=n;

empid=i;

salary=s;

designation=d;

}

// Temporary member

Employee()

{

name ="Bruh";

empid=95;

salary=99000.00;

designation="Temporary Member";

}

public void getData()

{

System.out.println("Name : "+name);

System.out.println("Id : "+empid);

System.out.println("Salary : "+ salary);

System.out.println("Designation : "+designation);

System.out.println();

}

}

public class Main

{

public static void main(String[] args)

{

Employee E[]=new Employee[3];

E[0]=new Employee();

E[1]=new Employee("Chandler");

E[2]=new Employee("Anakin",69,750000,"Executive");

System.out.println(":: Employees :: \n");

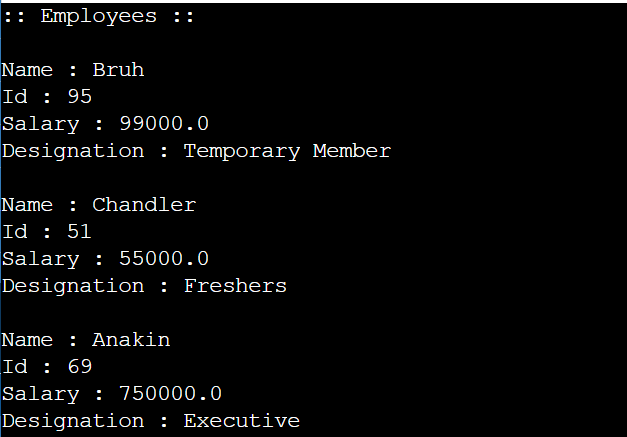
for(int i=0;i<3;i++)

E[i].getData();

}

}

OUTPUT:



1. Write a Java Program to illustrate Scanner, BufferReader and System class for input and output from terminal.Write a java program to illustrate - “Java uses pass by value”

**CODE :**

import java.util.\*;

class Employee{

private String name;

private int employee\_id;

private int salary;

private String designation;

static int emp\_count;

Employee(String name)

{

this.name=name;

this.employee\_id= ++emp\_count;

this.salary=26100;

this.designation="trainee";

}

Employee(String name,int salary,String designation)

{

this.name=name;

this.employee\_id= ++emp\_count;

this.salary=salary;

this.designation=designation;

}

Employee()

{

this.name="Eriksen" ;

this.employee\_id= ++emp\_count;

this.salary=5000;

this.designation="part\_time\_employee";

}

public static void swap(Employee a, Employee b){

Employee temp = a;

a = b;

b = temp;

}

public void display()

{

System.out.println("Name : "+name);

System.out.println("Employee id : "+employee\_id);

System.out.println();

}

}

class Swap{

public static void main(String args[])

{

Employee a = new Employee("Bruh");

a.display();

Employee b = new Employee("Lisandro",36200,"executive officer");

b.display();

System.out.println("After swap : ");

a.swap(a,b);

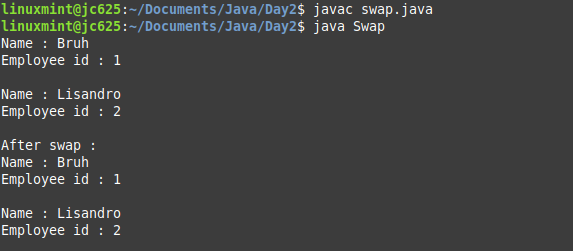
a.display();

b.display();

}

}

OUTPUT:



1. Write a Java Program to implement simple calculator using command line arguments.

**CODE :**

import java.util.\*;

class calculator {

public static void main(String args[]){

int val1 = Integer.parseInt(args[0]);

int val2 = Integer.parseInt(args[2]);

int result;

switch(args[1]){

case "+":

result= val1 + val2;

System.out.println("Sum : "+result);

break;

case "-":

result= val1 - val2;

System.out.println("Difference : "+result);

break;

case "x":

result = val1 \* val2;

System.out.println("Product : "+result);

break;

case "/":

result =val1 / val2;

System.out.println("Quotient : "+result);

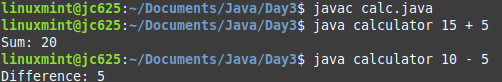
break;

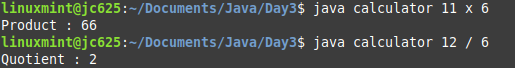
default:

System.out.println("Try again");

}}}

**OUTPUT :**





1. (a) In class Employee created in reference to Question 1 add static instance field

emp\_count. Use emp\_count for getting emp\_id in proper sequence. Also include appropiate accessor method for the instance field emp\_count

(b) Write a static method in Employee class.

(c) Write methods to compare two Employees based upon their salary and return object having higher salary.

(d) Write two overloading methods in your Employee class.

**CODE :**

import java.util.\*;

class Employee{

private String name;

private int employee\_id;

private int salary;

private String designation;

static int emp\_count;

public final String comp = "TCS";

public static Employee maxsal(Employee e1, Employee e2){

if (e1.ret\_sal() >= e2.ret\_sal() ) return e1;

else return e2;

}

public static Employee maxsal(Employee e1, Employee e2, Employee e3){

if (e1.ret\_sal() >= e2.ret\_sal() && e1.ret\_sal() >= e3.ret\_sal()) return e1;

if (e2.ret\_sal() >= e1.ret\_sal() && e2.ret\_sal() >= e3.ret\_sal()) return e2;

if (e3.ret\_sal() >= e1.ret\_sal() && e3.ret\_sal() >= e2.ret\_sal()) return e3;

return e1;

}

Employee(String name)

{

this.name=name;

this.employee\_id= ++emp\_count;

this.salary=26100;

this.designation="trainee";

}

Employee(String name,int salary,String designation)

{

this.name=name;

this.employee\_id= ++emp\_count;

this.salary=salary;

this.designation=designation;

}

Employee()

{

this.name="Eriksen" ;

this.employee\_id= ++emp\_count;

this.salary=5000;

this.designation="part\_time\_employee";

}

public int ret\_sal(){

return this.salary;

}

public String ret\_name(){

return this.name;

}

public void display()

{

System.out.println("Name : "+name);

System.out.println("Employee id : "+employee\_id);

System.out.println("Salary : " +salary);

System.out.println("Designation : "+designation);

System.out.println("Company Name : "+comp);

System.out.println();

}

}

class Grogu{

public static void main(String args[])

{

Employee fresher=new Employee("Bruh");

fresher.display();

Employee executive=new Employee("Lisandro",36200,"executive officer");

executive.display();

Employee part\_time\_member=new Employee();

part\_time\_member.display();

System.out.println("Comparing salary of employee 1 and 3 :");

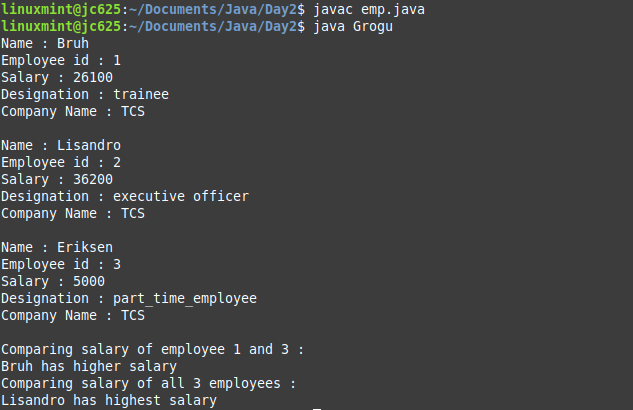
System.out.println(fresher.maxsal(fresher, part\_time\_member).ret\_name() + " has higher salary");

System.out.println("Comparing salary of all 3 employees :");

System.out.println(fresher.maxsal(fresher, executive, part\_time\_member).ret\_name() + " has highest salary");

}

}



1. Write a java program to illustrate following String API methods.

charAt() , compareTo(), equals(), equalsIgnoreCase(), indexOf(), length() , substring(), toCharArray() , toLowerCase(), toString(), toUpperCase() , trim() , valueOf(). Write a java program for explaining the concept of mutable and immutable string.

PART I :

import java.util.\*;

class Strtest{

public static void main(String args[]) throws Exception{

Scanner inp = new Scanner(System.in);

String str1 = "I have become comfortably numb";

String str2 = "There is no pain you are receding";

System.out.println("First string str1 : " + str1);

System.out.println("First string str2 : " + str2);

System.out.print("Enter the index to view character : ");

int n = inp.nextInt();

System.out.println("Demonstrating charAt()");

System.out.println("Character at index " + n + " is " + str1.charAt(n));

System.out.println("---------------------------------------------");

System.out.println("Demonstrating compareTo()");

System.out.println("Comparing str1 and str2 : " + str1.compareTo(str2));

System.out.println("---------------------------------------------");

System.out.println("Demonstrating equals()");

System.out.println("Checking if str1 and str2 are equal : " + str1.equals(str2));

System.out.println("---------------------------------------------");

System.out.println("Demonstrating equalsIgnoreCase()");

System.out.println("Checking if str1 and str2 are equal ( case irrespective) : " + str1.equalsIgnoreCase(str2));

System.out.println("---------------------------------------------");

System.out.println("Demonstrating indexOf()");

System.out.print("Enter the substring to view index : ");

String nstr = inp.nextLine();

System.out.println("using indexOf() on substring : " + str1.equalsIgnoreCase(nstr));

System.out.println("---------------------------------------------");

System.out.println("Demonstrating length()");

System.out.println("Length of str1 is : " + str1.length());

System.out.println("Length of str2 is : " + str2.length());

System.out.println("---------------------------------------------");

System.out.println("Demonstrating substring()");

System.out.print("Enter starting index : ");

n = inp.nextInt();

System.out.print("Enter ending index : ");

int n2 = inp.nextInt();

System.out.println("Substring of index between " + n + " & " + n2 + "is: "+ str1.substring(n,n2) + " , " + str2.substring(n,n2));

System.out.println("---------------------------------------------");

System.out.println("Demonstrating toCharArray()");

System.out.println("Using equals to show that a string and a string converted to character array are different : ");

System.out.println("str1.equals(str1.toCharArray()) : " + str1.equals(str1.toCharArray()));

System.out.println("---------------------------------------------");

System.out.println("Demonstrating toUpperCase() and toLowerCase()");

System.out.println("Uppercase of str1 : " + str1.toUpperCase());

System.out.println("Lowercase of str1 : " + str1.toLowerCase());

System.out.println("---------------------------------------------");

System.out.println("Demonstrating toString()");

System.out.print("Enter a number : ");

Integer n3 = inp.nextInt();

System.out.println("Getting runtime class after using toString() : " + n3.toString().getClass() );

System.out.println("---------------------------------------------");

System.out.println("Demonstrating trim()");

String str = " Hello World !! ";

System.out.println("String is : " + str);

System.out.println("String after usng trim() : " + str.trim());

System.out.println("---------------------------------------------");

System.out.println("Demonstrating valueOf()");

System.out.print("Enter a number : ");

n2 = inp.nextInt();

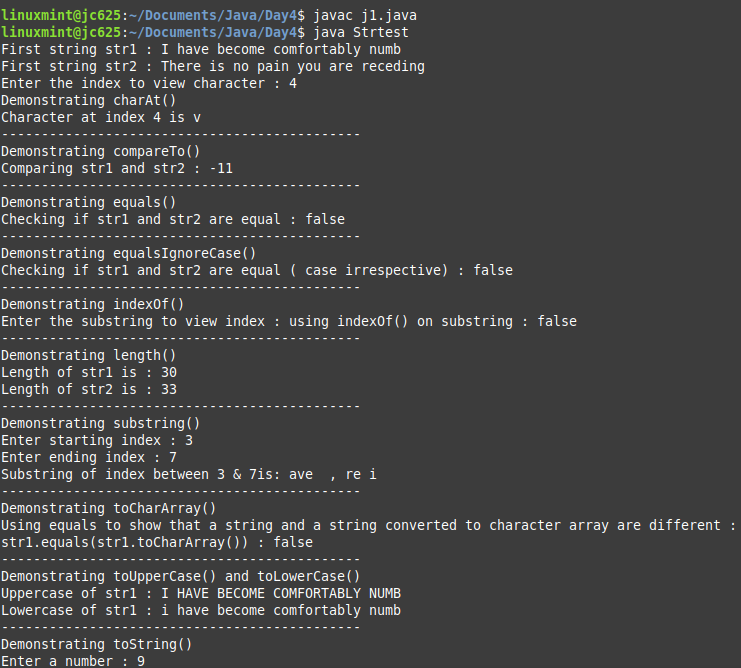
System.out.println("Converting number to string and concatenatng : " + String.valueOf(n2) + 10);

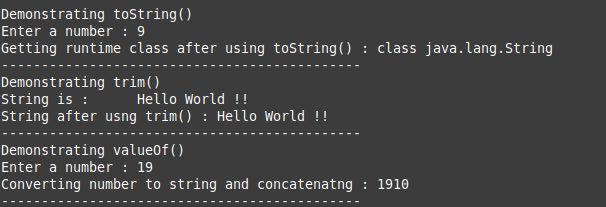
System.out.println("---------------------------------------------");

}

}

OUTPUT:





PART II:

CODE :

class MutImmut{

public static void main(String args[]){

String s="Hello";

s.concat(" Hi");

System.out.println("Immutable : "+s);

StringBuffer s1=new StringBuffer("Hello");

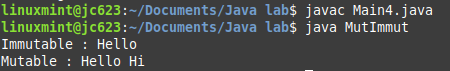
s1.append(" Hi");

System.out.println("Mutable : "+s1);

}

}

OUTPUT:



1. Write a java program to illustrate following StringBuffer API methods.

append(), capacity(), charAt(), delete(), deleteCharAt(), ensureCapacity(), getChars(),

indexOf(), insert(), length(), setCharAt(), setLength(), substring(), toString() . Write a java program for explaining the concept of mutable and immutable string.

class StringBufferExample{

public static void main(String args[]){

Scanner sc=new Scanner(System.in);

System.out.println("Enter the String: ");

String s=sc.nextLine();

StringBuffer s1 = new StringBuffer(s);

System.out.println("append(): "+s1.append("Hello"));

System.out.println("capacity(): "+s1.capacity());

System.out.println("charAt(): "+s1.charAt(0));

System.out.println("delete(): "+s1.delete(1,2));

System.out.println("deleteCharAt(): "+s1.deleteCharAt(0));

s1.ensureCapacity(100);

System.out.println("ensureCapacity(): "+s1.capacity());

System.out.println("indexOf(): "+s1.indexOf("w"));

System.out.println("insert(): "+s1.insert(5,"Hello"));

System.out.println("length(): "+s1.length());

s1.setCharAt(2,'e');

System.out.println("setCharAt(): "+s1);

s1.setLength(30);

System.out.println("setLength(): "+s1.length());

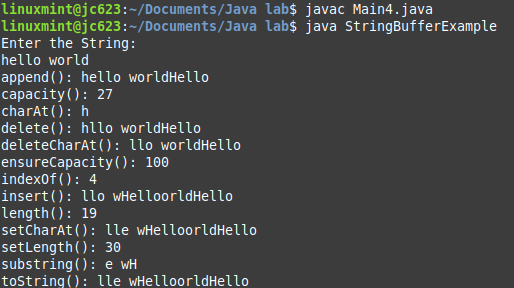
System.out.println("substring(): "+s1.substring(2,6));

System.out.println("toString(): "+s1.toString());

}

}

OUTPUT:



Kanamurgi, immutable ta ager question e a6e !!

1. Create a Student class having roll no. , name, dept., marks. Use array of objects to store details of 5 students. List the name of the student (a) having highest marks (b) lowest marks (c) Marks more than avarage. Also implement a sorting algorithm.

PART I:

import java.util.\*;

class Student implements Comparable<Student>{

int roll;

String name;

String dept;

double marks;

Student(int roll, String name, String dept, double marks){

this.roll = roll;

this.name = name;

this.dept = dept;

this.marks = marks;

}

@Override

public int compareTo(Student s){

if ( this.marks > s.marks) return 1;

else if (this.marks < s.marks) return -1;

else return 0;

}

}

class Test1{

public static double returnAverage(Student arr[]){

double avg = 0;

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

avg+= arr[i].marks;

}

return avg/arr.length;

}

public static void main(String[] args) {

Student arr[] = new Student[5];

arr[0] = new Student(3,"Chintu","CSE",85.56);

arr[1] = new Student(5,"Romelu","CSE",85.45);

arr[2] = new Student(7,"Virat","ECE",87.56);

arr[3] = new Student(9,"Lisandro","IT",91.77);

arr[4] = new Student(11,"Raphael","EE",94.14);

System.out.printf("Roll Name Dept. Marks\n");

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

System.out.printf("%3d %10s %6s %4f\n",arr[i].roll,arr[i].name,arr[i].dept,arr[i].marks);

}

Arrays.sort(arr);

System.out.println("Highest marks is " + arr[4].marks + " received by " + arr[4].name );

System.out.println("Lowest marks is " + arr[0].marks + " received by " + arr[0].name );

double avg = returnAverage(arr);

System.out.println("Average marks is : " + avg);

System.out.println("Marks greater than average : ");

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

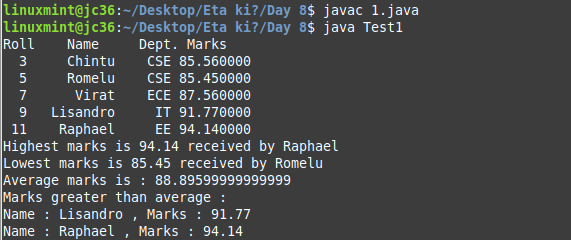
if (arr[i].marks > avg) System.out.println("Name : " + arr[i].name + " , Marks : " + arr[i].marks);

}

}

}

OUTPUT:



PART II:

CODE :

import java.util.\*;

class Student{

int roll;

String name;

String dept;

double marks;

Student(int roll, String name, String dept, double marks){

this.roll = roll;

this.name = name;

this.dept = dept;

this.marks = marks;

}

}

public class Main{

public static void insertionSort(Student arr[]){

for(int i=1; i < arr.length; i++){

Student key = arr[i];

int j = i-1;

while (j >=0 && arr[j].roll > key.roll) {

arr[j+1] = arr[j];

j = j - 1;

} arr[j+ 1] = key;

}

}

static void swap(Student arr[], int i, int j){

Student temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

static int partition(Student[] arr, int low, int high){

double pivot = arr[high].marks;

int i = (low - 1);

for(int j = low; j <= high - 1; j++){

if (arr[j].marks > pivot){

i++;

swap(arr, i, j);

}

}

swap(arr, i + 1, high);

return (i + 1);

}

static void quickSort(Student arr[], int low, int high){

if (low < high){

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

public static void main(String[] args) {

Student arr[] = new Student[5];

arr[0] = new Student(3,"Chintu","CSE",85.56);

arr[1] = new Student(5,"Romelu","CSE",85.45);

arr[2] = new Student(7,"Virat","ECE",87.56);

arr[3] = new Student(9,"Lisandro","IT",91.77);

arr[4] = new Student(11,"Raphael","EE",94.14);

System.out.printf("Roll Name Dept. Marks\n");

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

System.out.printf("%3d %10s %6s %4f\n",arr[i].roll,arr[i].name,arr[i].dept,arr[i].marks);

}

insertionSort(arr);

System.out.println("Sorted array (according to roll) after applying Insertion Sort :");

System.out.printf("Roll Name Dept. Marks\n");

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

System.out.printf("%3d %10s %6s %4f\n",arr[i].roll,arr[i].name,arr[i].dept,arr[i].marks);

}

System.out.println("Reverse sorted array (according to marks) after applying Quick Sort :");

quickSort(arr,0, arr.length - 1);

System.out.printf("Roll Name Dept. Marks\n");

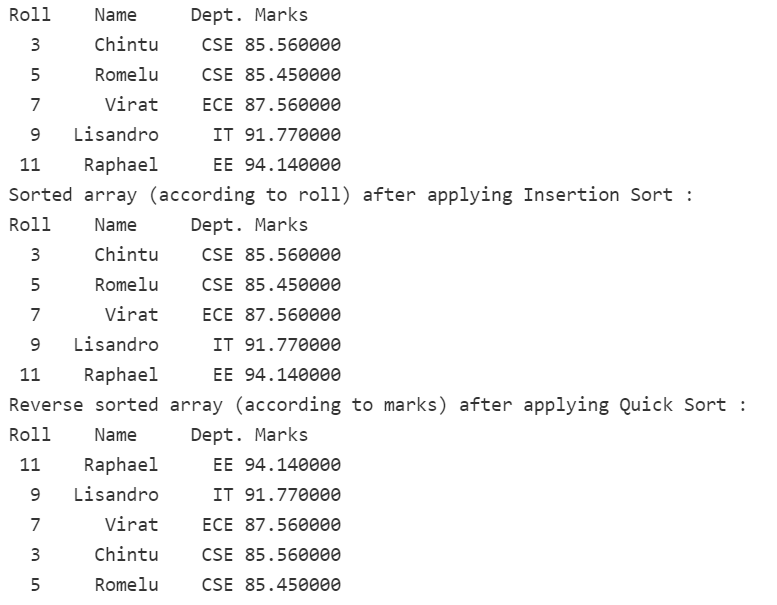
for (int i=0;i < arr.length ;i++ ) {

System.out.printf("%3d %10s %6s %4f\n",arr[i].roll,arr[i].name,arr[i].dept,arr[i].marks);

}

}

}



1. (a) Write a program to implement the concept of Exception Handling using predefined exception.

(b) Write a program to implement the concept of Exception Handling by creating user defined exceptions.

NOTE: Use throws, throw, try, catch and finally keywords in your program

(a)

CODE:

**CODE:**

import java.util.\*;

import java.lang.\*;

class Exception1 {

public static void main(String[] args) {

try {

int arr[] = new int[2];

System.out.println("Access third element : " + arr[3]);

} catch(Exception e) {

System.out.println("Exception thrown : " + e);

}finally{

System.out.println("This block is executed.");

}try {

int a = 5/ 0;

}catch(Exception e) {

System.out.println("Exception thrown : " + e);

}finally{

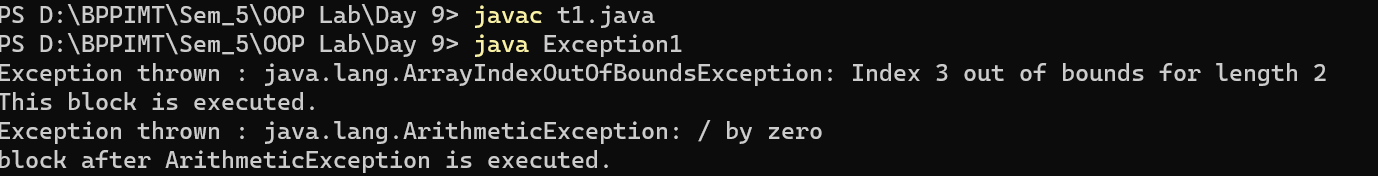
System.out.println("block after ArithmeticException is executed.");

}

}

}

**OUTPUT:**



(b)**throw keyword :**

**CODE:**

class UserDefinedException extends Exception{

public UserDefinedException(String str){

// Calling constructor of parent Exception

super(str);

}

}

class Excep2{

public static void main(String args[]){

try{

// throw an object of user defined exception

throw new UserDefinedException("This is user-defined exception");

}catch (UserDefinedException ude){

System.out.println("Caught the exception");

// Print the message from MyException object

System.out.println(ude.getMessage());

}finally{

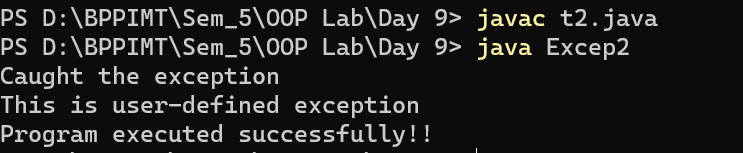
System.out.println("Program executed successfully!!");

}

}

}

**OUTPUT:**



**throws keyword :**

**CODE:**

import java.util.\*;

class myex extends Exception{

myex(String s){

super(s);

}

public String toString(){

return "sorry!! your age is less than required";

}

}

class Excep3{

public static void main (String args[]) throws myex{

int age=13;

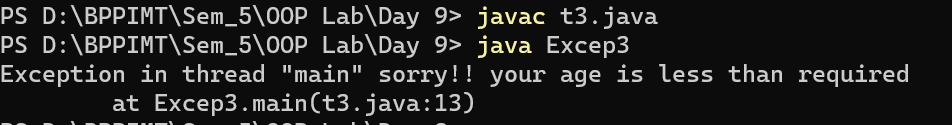
if(age<18) throw new myex("not for under 18");

else System.out.println("welcome to vote");

}

}

**OUTPUT:**



1. Create a class Shape having at most two dimensions. Define two subclasses: circle and rectangle of Shape.

(a) Override a method area.

(b) Show compile time and run time polymorphism(Dynamic method dispatch).

(c) Use a final method for display.

(d) use super keyword.

**CODE:**

import java.util.\*;

class Shape{

void pshape(){

System.out.println("Shape not specified.");

}

final void cp(){

System.out.println("This is the OG class.");

}

}

class Circle extends Shape{

void pshape(String shape){

System.out.println("Original method : ");

super.pshape();

System.out.println("Shape : " + shape);

}

public static void main(String args[]){

Circle c = new Circle();

System.out.println("Method overloading : shown in the method pshape()");

c.cp();

c.pshape("Circle");

}

}

class Rectangle extends Shape{

void pshape(){

System.out.println("Shape : Rectangle");

}

public static void main(String args[]){

Shape s = new Shape();

Rectangle r = new Rectangle();

s.pshape();

r.pshape();

s = r;

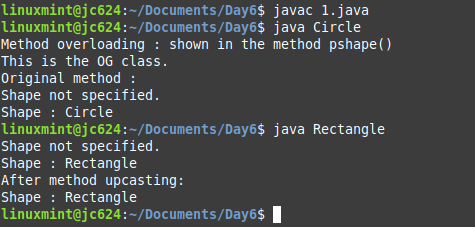
System.out.println("After method upcasting:");

s.pshape();

}

}

**OUTPUT:**



1. (a) Define a abstract class figure. Define the area and volume method in the child classes. Use dynamic method dispatch.

(b) Implement the following design with suitable example classes.

Interface A

Interface B

abstract class C

class D

Solution:

(a)CODE:

import java.util.\*;

abstract class Figure {

public abstract void area();

public abstract void volume();

}

class Sphere extends Figure {

int r;

public void getRadius(){

Scanner my\_input = new Scanner(System.in);

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

r = my\_input.nextInt();

}

public void area(){

System.out.println("Area of Sphere is : " + (4\*Math.PI\*r\*r));

}

public void volume(){

System.out.println("volume of Sphere is : " + (4\*Math.PI\*r\*r\*r/3));

}

}

class Cube extends Figure{

int a;

public void getEdge(){

Scanner my\_input = new Scanner(System.in);

System.out.print("Enter edge of cube : ");

a = my\_input.nextInt();

}

public void area(){

System.out.println("Area of Cube is : " + (6\*a\*a));

}

public void volume(){

System.out.println("volume of Cube is : " + (a\*a\*a));

}

}

class Test{

public static void main(String[] args) {

Sphere s = new Sphere();

s.getRadius();

s.area();

s.volume();

Cube c = new Cube();

c.getEdge();

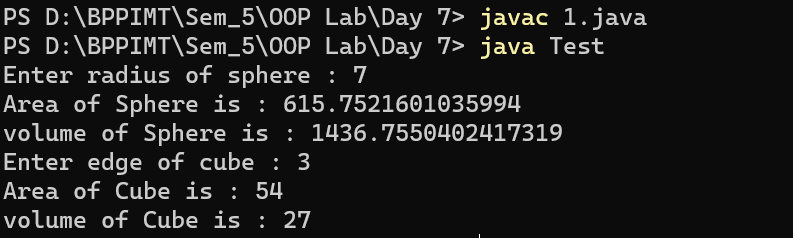
c.area();

c.volume();

}

}

OUTPUT:



(b)CODE:

import java.util.\*;

interface FindArea{

public void area();

}

interface FindVolume{

public void volume();

}

abstract class FindRadius{

public abstract void getRadius();

}

class Sphere extends FindRadius implements FindArea, FindVolume {

int r;

public void getRadius(){

Scanner my\_input = new Scanner(System.in);

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

r = my\_input.nextInt();

}

public void area(){

System.out.println("Area of Sphere is : " + (4\*Math.PI\*r\*r));

}

public void volume(){

System.out.println("volume of Sphere is : " + (4\*Math.PI\*r\*r\*r/3));

}

}

class Test2{

public static void main(String[] args) {

Sphere s = new Sphere();

s.getRadius();

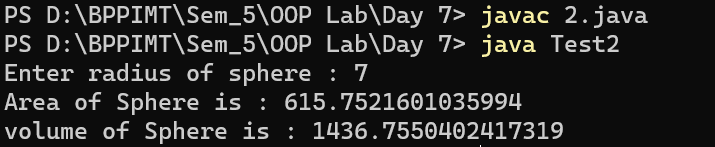
s.area();

s.volume();

}

}

OUTPUT:



1. (a) Define a abstract class figure. Define the area and volume method in the child classes. Use dynamic method dispatch.

(b) Implement the following design with suitable example classes.

Interface A

Interface B

abstract class C

class D

Question repeat kore6e dek6i ..

1. Write a Java program to implement the concept of importing classes from user defined package and creating packages. Write a java program to explain the use of access specifiers - Public,Protected, Default, Private w.r.t. instance fields, methods and classes

PART I:

**CODE:**

import AddPack.\*;

import java.util.\*;

class trial{

public static void main(String args[]){

AddPack.add t = new AddPack.add();

t.addition();

}

}

**Package contents :**

package AddPack;

import java.util.\*;

public class add {

public void addition(){

Scanner n = new Scanner(System.in);

System.out.print("Enter 1st number : ");

int num1 = n.nextInt();

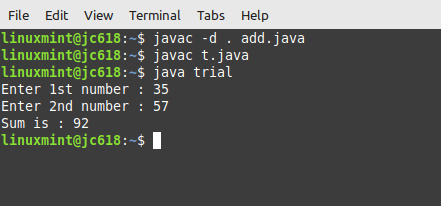
System.out.print("Enter 2nd number : ");

int num2 = n.nextInt();

System.out.println("Sum is : " + (num1 + num2));

}

}



PART II :

package AllPack;

import AddPack.\*;

import java.util.\*;

class publicAccess{

static void division(){

Scanner n = new Scanner(System.in);

System.out.print("Enter 1st number : ");

int num1 = n.nextInt();

System.out.print("Enter 2nd number : ");

int num2 = n.nextInt();

System.out.println("Quotient is : " + (num1 / num2));

}

public static void main(String args[]){

AddPack.add t = new AddPack.add();

t.addition();

}

}

// protected methods are accessible within the same package or subclasses in different packages

class protectedAccess extends AddPack.add{

public static void main(String[] args) {

protectedAccess pro = new protectedAccess();

pro.subtraction();

}

}

// methods with default access are accessible within the same package

class DefaultAccess{

public static void main(String[] args) {

publicAccess t = new publicAccess();

t.division();

}

}

class PrivateAccess{

private void product(){

Scanner n = new Scanner(System.in);

System.out.print("Enter 1st number : ");

int num1 = n.nextInt();

System.out.print("Enter 2nd number : ");

int num2 = n.nextInt();

System.out.println("Product is : " + (num1 \* num2));

}

public static void main(String[] args) {

System.out.println("As the method 'product' is private it can only be accessible from this class");

PrivateAccess t = new PrivateAccess();

t.product();

}

}

**Package Contents:**

package AddPack;

import java.util.\*;

public class add {

public void addition(){

Scanner n = new Scanner(System.in);

System.out.print("Enter 1st number : ");

int num1 = n.nextInt();

System.out.print("Enter 2nd number : ");

int num2 = n.nextInt();

System.out.println("Sum is : " + (num1 + num2));

}

protected void subtraction(){

Scanner n = new Scanner(System.in);

System.out.print("Enter 1st number : ");

int num1 = n.nextInt();

System.out.print("Enter 2nd number : ");

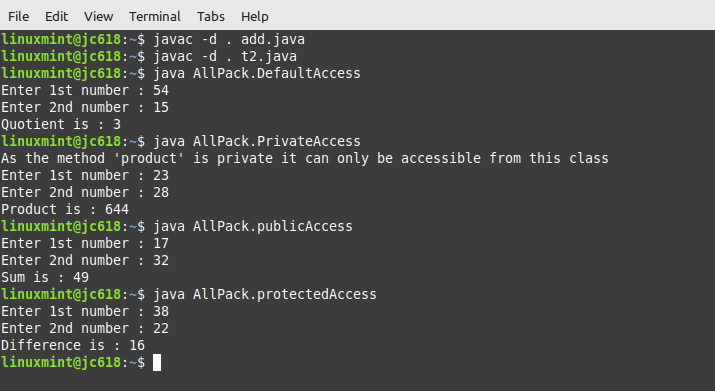
int num2 = n.nextInt();

System.out.println("Difference is : " + (num1 - num2));

}

}

**OUTPUT:**



1. (a) Write a Java program for printing your name ,roll no section by clicking a button using Swing. or

(b)Write a Java Program to change the background color of an applet to Red and Blue using respective Red and Blue buttons.

(a)

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

class MyWindow{

public static void main(String[] args) {

String s="Kanamurgi";

String r="11500120003";

final JFrame frame = new JFrame();

frame.setSize(400, 200);

JPanel panel = new JPanel();

JButton button1 = new JButton("display info");

final JLabel l1 = new JLabel(s);

final JLabel l2 = new JLabel(r);

l1.setVisible(false);

l2.setVisible(false);

frame.add(panel);

panel.add(button1);

panel.add(l1);

panel.add(l2);

frame.setVisible(true);

button1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

l1.setVisible(true);

l2.setVisible(true);

}

});

}

}

(b)import java.awt.\*;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

class Buttons{

Buttons(){

JFrame f = new JFrame("Button Assignment");

f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

f.setSize(400, 200);

JPanel panelMain = new JPanel();

final JButton red\_button = retButton("RED");

final JButton blue\_button = retButton("BLUE");

final JButton reset\_button = retButton("RESET");

final Color defaultColor = panelMain.getBackground();

ActionListener buttonListener = new ActionListener(){

@Override

public void actionPerformed(ActionEvent ae){

Object o = ae.getSource();

if(o == red\_button) panelMain.setBackground(Color.RED);

else if(o == blue\_button) panelMain.setBackground(Color.BLUE);

else if(o == reset\_button) panelMain.setBackground(defaultColor);

}

};

red\_button.addActionListener(buttonListener);

blue\_button.addActionListener(buttonListener);

reset\_button.addActionListener(buttonListener);

panelMain.add(red\_button);

panelMain.add(blue\_button);

panelMain.add(reset\_button);

f.add(panelMain);

f.setVisible(true);

}

public static JButton retButton(String address){

JButton button = new JButton(address);

return button;

}

public static void main(String[] args) {

new Buttons();

}

}

OUTPUT:

// resets on clicking reset…

1. (a) Write a program to create multiple threads. (b) Use join(), isAlive(), getPriority(), SetPriority() methods.

class A extends Thread{

public void run(){

for(int i=0; i < 3; i++){

System.out.println("Hi " + Thread.currentThread().getPriority());

try { Thread.sleep(1000); } catch (Exception e){}

}

}

}

class B extends Thread{

public void run(){

for(int i=0; i < 3; i++){

System.out.println("Hello " + Thread.currentThread().getPriority());

try { Thread.sleep(1000); } catch (Exception e){}

}

}

}

class ThreadDemo2{

public static void main(String[] args) throws Exception{

A t1 = new A();

B t2 = new B();

//setting priority of thread

t1.setPriority(Thread.MIN\_PRIORITY);

t2.setPriority(Thread.MAX\_PRIORITY);

//getting priority of thread .. initially assigned default priority of 5 or Thread.NORM\_PRIORITY

System.out.println("priority of t1 : " + t1.getPriority());

System.out.println("priority of t2 : " + t2.getPriority());

t1.start();

t2.start();

// calling isAlive before join()

System.out.println("Checking whether t1 is alive (before join): " + t1.isAlive());

// thread on joining throws Exception, so we have to declare 'throws Exception' at main

t1.join();

t2.join();

// calling isAlive after join()

System.out.println("Checking whether t1 is alive (after join): " + t1.isAlive());

//t1 will de dead after the program execeutes, so we will get false

System.out.println("Bye !!");

}

}

**OUTPUT:**

