1. Write a Java program to accept IA marks obtained by five students in three subjects. The program should accept marks obtained by each student and display the total marks and the average marks. The average marks is computed using a method as the average of best two marks obtained.

import java.util.Scanner;

public class TW1 {

public static void main(String[] args){

int[][] marks=new int[5][3];

int[] total = {0,0,0,0,0};

int[] avg = new int[5];

Scanner in = new Scanner(System.in);

for(int i=0; i<5; i++){

System.out.println("Enter the marks of student "+(i+1));

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

marks[i][j]=in.nextInt();

total[i]=total[i]+marks[i][j];

}

}

for(int i=0;i<5;i++)

avg[i] = computeAvg(marks[i][0], marks[i][1],marks[i][2]);

System.out.println("Student results:");

for(int i=0;i<5;i++){

System.out.println("For student " +(i+1));

System.out.println("Total marks:" +total[i]);

System.out.println("Avreage marks:" +avg[i]);

}

}

static int computeAvg(int m1,int m2,int m3){

int min = m1;

if(m2<min)

min = m2;

if(m3<min)

min = m3;

int total = (m1+m2+m3-min);

return(int)Math.ceil(total/2.0);

}

}

**2)** A joint family consisting of 10 households lives in the same compound. Due to mounting electricity bills, the head (Mr. X) of the joint family decides to analyze the consumption pattern (in terms of the billed amount) of each household for a year. Mr. X needs access to the following information for his analysis. Write a function to find:

1) The total expenditure on electricity consumption by each household in a year.

2) The maximum and minimum electricity consumption of each household in a year.

3) The amount by which each household exceeded the average consumption (+/-) of all households in the month of June.

4) The maximum, minimum and average electricity consumption of all households in a year.

Demonstrate how you would use a two dimensional matrix to help Mr. X.

import java.util.Scanner;

class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("How many households? ");

int householdCount = sc.nextInt();

float[][] data = new float[householdCount][];

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

System.out.println("Enter the data for Household " + (i + 1));

float[] householdData = new float[12 \* 2];

for (int j = 0; j < 12; j++) {

System.out.print("Enter the electricity consumption and bill for month " + (j + 1) + ": ");

float electricityConsumption = sc.nextFloat();

float bill = sc.nextFloat();

householdData[j \* 2] = electricityConsumption;

householdData[j \* 2 + 1] = bill;

}

data[i] = householdData;

}

float[] totalExpenditure = getTotalExpenditure(data);

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

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

System.out.println(" For household " + (i + 1) + ": " + totalExpenditure[i]);

}

float[][] maxAndMinElectricityConsumption = getMaxAndMinElectricityConsumption(data);

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

System.out.println(" For household " + (i + 1));

System.out.println("Max. electricity consumption: " + maxAndMinElectricityConsumption[i][0]);

System.out.println("Min. electricity consumption: " + maxAndMinElectricityConsumption[i][1]);

}

float[] avgDiffOfElectricityConsumptionInJune = getAvgDiffOfElectricityConsumptionInJune(data);

System.out.println("---Average Difference of Electricity Consumption in June---");

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

System.out.println(" For household " + (i + 1) + ": " + avgDiffOfElectricityConsumptionInJune[i]);

}

float[][] maxMinAndAvgElectricityConsumption = getMaxMinAndAvgElectricityConsumption(data);

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

System.out.println(" For household " + (i + 1));

System.out.println("Max. electricity consumption: " + maxMinAndAvgElectricityConsumption[i][0]);

System.out.println("Min. electricity consumption: " + maxMinAndAvgElectricityConsumption[i][1]);

System.out.println("Avg. electricity consumption: " + maxMinAndAvgElectricityConsumption[i][2]);

}

}

// "data" parameter structure

// for every household -> [ electricity consumption of first month, bill of first month, ...., electricity consumption of last month, bill of last month ]

public static float[] getTotalExpenditure(float[][] data) {

float[] generatedData = new float[data.length];

// loop through every household data

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

float annualBill = 0;

// loop through i th household's data (i.e, bill of every month, odd intervals (1, 3, 5, 7, etc..))

for (int j = 1; j < data[i].length; j += 2)

annualBill += data[i][j];

// store annualBill in a seperate array

generatedData[i] = annualBill;

}

return generatedData;

}

public static float[][] getMaxAndMinElectricityConsumption(float[][] data) {

// "generatedData" structure

// for every household -> [ max electricity consumption, min electricity consumption ]

float[][] generatedData = new float[data.length][];

// loop through every household

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

float max = 0, min = 0;

// get the largest and smallest electricity consumption for that household

for (int j = 0; j < data[i].length; j += 2) {

if (j == 0)

max = min = data[i][j];

else if (max < data[i][j])

max = data[i][j];

else if (min > data[i][j])

min = data[i][j];

}

// store the largest and smallest electricity consumption in the generatedData

generatedData[i] = new float[] { max, min };

}

return generatedData;

}

public static float[] getAvgDiffOfElectricityConsumptionInJune(float[][] data) {

float avgElectricityConsumptionInJune = 0;

// firstly loop through every household and compute the average for june (june month index is 5, but since every month takes two elements, june month data is present at 5 \* 2 = 10)

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

avgElectricityConsumptionInJune += data[i][5 \* 2] / data[i].length \* 2;

}

float[] generatedData = new float[data.length];

// loop through every household

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

// get difference between its i th electricity consumption and the calculated avg

generatedData[i] = data[i][5 \* 2] - avgElectricityConsumptionInJune;

}

return generatedData;

}

public static float[][] getMaxMinAndAvgElectricityConsumption(float[][] data) {

float[] avgData = new float[data.length];

// loop through every household

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

float avg = 0;

// get avg electricity consumption by looping through data of that household's electricity consumption

for (int j = 0; j < data[i].length; j += 2) {

avg += data[i][j] / data[i].length \* 2;

}

avgData[i] = avg;

}

// get max and min electricity consumption from func

float[][] maxAndMinElectricityConsumption = getMaxAndMinElectricityConsumption(data);

// "generatedData" structure

// for every household -> [ max electricity consumption, min electricity consumption, avg electricity consumption ]

float[][] generatedData = new float[data.length][];

// merging avgData with maxAndMinElectricityConsumption

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

generatedData[i] = new float[] { maxAndMinElectricityConsumption[i][0], maxAndMinElectricityConsumption[i][1], avgData[i] }; // [max, min, avg]

}

return generatedData;

}

}

3) A company has 10 zonal sales offices in four zones namely, North, East, West and South. The company wants to organize the sales data of each of the office in each zone and find answers to queries such as,

1. Which office has performed the highest sales in each zone?

2. What is the average sales done by all the offices in each zone?

3. Which office among each zone is the poorly office?

import java.io.\*;

import java.lang.\*;

import java.util.\*;

class TW1{

static int maximum(int a[]){

int m=0,pos=0;

for(int i=0;i<10;i++){

if(a[i]>m){

m=a[i];

pos=i+1;

}

}

return pos;

}

static int minimum(int a[]){

int m=99999,pos=0;

for(int i=0;i<10;i++){

if(a[i]<m){

m=a[i];

pos=i+1;

}

}

return pos;

}

public static void main(String[]args){

Scanner venki = new Scanner(System.in);

int data[][]=new int[4][10];

int i,j,max[]=new int[4];

int min[]=new int[4];

float avg[]=new float[4];

String zones[]={"North","South","East","West"};

System.out.println("Enter the data of the zones : ");

for(i=0;i<4;i++){

System.out.println("\n"+zones[i]+" : ");

int sum=0;

for(j=0;j<10;j++){

System.out.println("Office "+(j+1)+" : ");

data[i][j]=venki.nextInt();

sum+=data[i][j];

}

avg[i]=sum/10;

max[i] = maximum(data[i]);

min[i] = minimum(data[i]);

}

System.out.println("\nOffice Number with highest sales zone wise : ");

for(i=0;i<4;i++)

System.out.println(zones[i]+" : "+max[i]);

System.out.println("\nOffice with poorest sales zone wise : ");

for(i=0;i<4;i++)

System.out.println(zones

[i]+" : "+min[i]);

System.out.println("\nAverage sales zone wise : ");

for(i=0;i<4;i++)

System.out.println(zones[i]+" : "+avg[i]);

}

}

4) Design a class by name myTriangle to model a triangle geometrical object with three sides. Include functions to:

* • Initialize the three sides of triangle.
* • Determine the type of triangle represented by the three sides (Equilateral/ Isosceles/ Scalene triangle**).**
* • Compute and return the area of the triangle.

import static java.lang.System.exit;

import java.util.Scanner;

class Triangle {

double a,b,c;

void getSides(){

Scanner in=new Scanner(System.in);

System.out.println("Enter 3 sides of a triangle:");

a=in.nextDouble();

b=in.nextDouble();

c=in.nextDouble();

}

void checkTriangle(){

if((a+b)>c && (b+c)>a && (a+c)>b){

if(a==b && b==c && c==a)

System.out.println("Triangle is equilateral");

else if(a==b || b==c || c==a)

System.out.println("Triangle is isosceles");

else

System.out.println("Triangle is scalene");

}

else{

System.out.println("Triangle cannot be formed");

exit(0);

}

}

double computeArea(){

double s=(a+b+c)/2;

double area=Math.sqrt(s\*(s-a)\*(s-b)\*(s-c));

return area;

}

}

public class TW1 {

public static void main(String[] args){

Triangle t=new Triangle();

t.getSides();

t.checkTriangle();

if((t.computeArea())!=0){

System.out.println("Area is "+t.computeArea());

}

}

}

5) Design a class by name myCircle to model Circle geometrical object with its center and radius that enables:

1. Initializing the center, radius and

2. Compute area, perimeter, and diameter of the circle object/s.

class Circle

{

double x,y,area,perimeter,diameter,radius; Circle()

{

x=0; y=0;

area=0; perimeter=0; diameter=0; radius=0;

}

Circle(float X,float Y,float R)

{

x=X; y=Y;

radius=R; diameter=2\*radius;

}

double comp\_area()

{

area=3.142\*radius\*radius; return area;

}

double comp\_perimeter()

{

perimeter=2\*3.142\*radius; return perimeter;

}

void display()

{

System.out.println("Center of circle:"+"("+x+","+y+")"); System.out.println("Radius of circle:"+radius); System.out.println("Diameter of circle:"+diameter); System.out.println("Area of circle:"+comp\_area()); System.out.println("Perimeter of circle:"+comp\_area());

}

}

class myCircle

{

public static void main(String args[])

{

Circle c1=new Circle(2,2,4); c1.display();

Circle c2=new Circle(3,6,8); c2.display();

}

}

6) Define a class to represent the student details such as name, roll number, marks obtained in three internal assessment tests.

a) Identify type and declare the instance variables

b) Identify and develop the constructors to initialize the instance variables

c) Write a method computeAverage() to compute the average as the average of best two marks

d) Write a method to display the student details

Write the corresponding Driver class to instantiate an array of student objects and demonstrate the working of the application by invoking appropriate methods.

import java.util.Scanner;

class student\_def

{

String name; int rollno;

int m1,m2,m3; student\_def()

{

Scanner s=new Scanner(System.in); System.out.println("Enter the student name:"); String name=new String();

name=s.next();

System.out.println("Enter the student roll number:"); rollno=s.nextInt();

System.out.println("Enter the student marks in three subjects:"); m1=s.nextInt();

m2=s.nextInt(); m3=s.nextInt();

}

student\_def(String N,int R,int M1,int M2,int M3)

{

name=N; rollno=R; m1=M1; m2=M2; m3=M3;

}

double computeAverage()

{

int min=m1;

if(m2<min) min=m2; if(m3<min)

min=m3;

double average=(float)((m1+m2+m3-min)/2); return average;

}

void display()

{

System.out.println("Student details are:");

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

System.out.println("Student roll number:"+rollno);

System.out.println("Student marks:"+m1+","+m2+","+m3);

System.out.println("Total marks:"+(m1+m2+m3));

System.out.println("Average of best two marks:"+computeAverage());

}

}

class student

{

public static void main(String args[])

{

student\_def s1=new student\_def(); s1.display();

student\_def s2=new student\_def("Navjeet",68,27,29,26); s2.display();

}

}

7) A certain small bank intends to automate few of its banking operations for its customers. Design a class by name mybankAccount to store the customer data having following details:

1.accountNumber 2. acctType 3. Name 4. Address 5. AccountBalance

The class must have both default and parameterized constructors. Write appropriate method to compute interest accrued on accountBalance based on accountType and time in years. Assume 5% for S/B account 6.5% for RD account and 7.65 for FD account. Further, add two methods withdrawAmount/depositAmount with amount as input to withdraw and deposit respectively. The withdrawAmount method must report in-sufficient balance if accountBalance falls below Rs. 1000.

Build the class with appropriate member variables, constructors and methods. Write a function to display all the three customer details in a tabular form with appropriate column headings. Instantiate three objects of above type and perform different operations on the same.

import static java.lang.System.exit;

import java.util.Scanner;

class Mybankaccount{

int accno;

String acctype,name,address;

double balance;

static int count=1;

Mybankaccount(){

Scanner s = new Scanner(System.in);

accno = count++;

System.out.println("Enter the account type :");

acctype = s.nextLine();

System.out.println("Enter the account holders name :");

name = s.nextLine();

System.out.println("Enter the account holders address :");

address = s.nextLine();

System.out.println("Enter the account holders balance :");

balance = s.nextDouble();

}

Mybankaccount(String type,String name,String address,double balance){

accno = count++;

this.acctype = type;

this.name = name;

this.address = address;

this.balance = balance;

}

void computeInterest()

{

double interest = 0;

switch(acctype)

{

case "SB" :

interest = balance \*0.05;

balance+=interest;

break;

case "RD" :

interest = balance \* (6.5/100);

balance+=interest;

break;

case "FD" :

interest = balance \* (7.65/100);

balance+=interest;

break;

}

System.out.println("the interest gained is :"+ interest);

}

void withdraw(double amt)

{

if(acctype.equals("SB"))

{

if(balance - amt<1000)

{

System.out.println("Insuffient amount in balance ");

exit(0);

}

else

balance = balance - amt;

}

else

System.out.println("amount Cant be withdrawn from this type of account");

}

void deposit(double amt)

{

balance = balance + amt;

}

void dispDetails()

{

System.out.println(String.format("%-5s|%-5d|%-10s|%-15s|%-6f|",acctype,accno,name,address,balance));

}

}

public class Tw3Bank {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

Mybankaccount a1 = new Mybankaccount("SB","Samuel","Bhagyanagar",5000);

Mybankaccount a2 = new Mybankaccount("RD","Rahul","Ganesh nagar",6000);

Mybankaccount a3 = new Mybankaccount("FD","Ajay","vilas nagar",8000);

a1.dispDetails();

a2.dispDetails();

a3.dispDetails();

Mybankaccount a4 = new Mybankaccount();

a4.dispDetails();

a4.computeInterest();

System.out.println("Enter the amount to be deposited :");

double amt = s.nextDouble();

a4.deposit(amt);

System.out.println("Enter the amount to be withdrawn :");

double amt1 = s.nextDouble();

a4.withdraw(amt1);

a4.dispDetails();

}

}

8) Write a Java program to represent a Complex number. Include member functions to:

1. Initialize a complex number to a default value of zero (default constructor)

2. Initialize a complex number to a user defined value (parameterized constructor)

3. Add two complex numbers and return the result. (Parameterized method)

4. Subtract two complex numbers and return the result. (Parameterized method)

5. Display a complex number. (non-parameterized method)

class complex\_def

{

double real,img; complex\_def()

{

real=0; img=0;

}

complex\_def(double R,double I)

{

real=R; img=I;

}

complex\_def add(complex\_def c1)

{

complex\_def t=new complex\_def(); t.real=real+c1.real; t.img=img+c1.img;

return t;

}

complex\_def sub(complex\_def c1)

{

complex\_def t=new complex\_def(); t.real=real-c1.real;

t.img=img-c1.img; return t;

}

void display()

{

System.out.println("Real part="+real); System.out.println("Imaginary part="+img);

}

}

class complex

{

public static void main(String args[])

{

complex\_def c1=new complex\_def(10,20); complex\_def c2=new complex\_def(1,2); complex\_def c3=new complex\_def(); System.out.println("Complex number 1:"); c1.display();

System.out.println("Complex number 2:"); c2.display(); System.out.println("Addition:"); c3=c1.add(c2);

c3.display(); System.out.println("Subtraction:"); c3=c1.sub(c2);

c3.display();

}

}

9) Create a IndMoney class with two integer instance variables rupees and paise. Add a constructor with two parameters for initializing a IndMoney object. The constructor should check the paise value is between 0 and 99 and, if not, transfer some of the paise to the rupees variable to make it between 0 and 99. Add a plus method to the class that takes a IndMoney

object as parameter. It creates and returns a new IndMoney object representing the sum of the object whose plus() method is being invoked and the parameter. It does not modify the values of the two existing objects. It should also ensure that the value of the paise instance variable of the new object is between 0 and 99. For example, if x is an IndMoney object with 12 rupees and 80 paise, and if y is an IndMoney object with 8 rupees and 90 paise, then x.plus(y) will return a new IndMoney object with 21 rupees and 70 paise. Also, create a IndMoneyDemo driver class that tests the IndMoney class.

class indMoney

{

int rupees,paise;

public indMoney(int rupees, int paise)

{

if(paise > 99)

{

int tmprupees = paise/100;

paise = paise % 100;

this.paise = paise;

this.rupees = rupees+tmprupees;

}

else

{

this.paise = paise;

this.rupees = rupees;

}

}

public indMoney plus(indMoney money)

{

money.rupees = this.rupees+money.rupees;

money.paise = this.paise+money.paise;

if(money.paise > 99)

{

int tmprupees = money.paise/100;

money.paise = money.paise % 100;

money.rupees = money.rupees+tmprupees;

}

return money;

}

public void display()

{

System.out.println("Rupees: "+this.rupees);

System.out.println("Paise: "+this.paise);

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

}

}

class Termwork34

{

public static void main(String args[])

{

indMoney m1 = new indMoney(35,650);

System.out.println("currency m1");

m1.display();

indMoney m2 = new indMoney(75,660);

System.out.println("currency m2");

m2.display();

indMoney m3 = new indMoney(0,0);

m3 = m2.plus(m1);

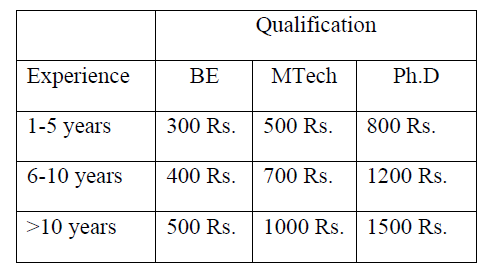
System.out.println("currency m3");

m3.display();

}

}

10) A company has two types of employees – FullTime and Partime. The company records for each employee his/her name, age, address, salary and gender. Given the basic salary of the FullTime employee the components of his/her gross salary are: Dearness allowance – 75% of basic salary, HRA – 7.5% of basic salary, IT – 10% of basic. The salary of a Partime employee is dependent on the qualification, experience, number of working hours and the rate per hour, as below:

}

Model this as a problem of hierarchical inheritance. Test the program by creating objects of the classes that are so identified.

class Employ{

String name,address,gender;

int age;

double sal;

Employ(String name,int age,String address,String gender){

this.name=name;

this.age=age;

this.address=address;

this.gender=gender;

}

void show(){

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

System.out.println("Age:"+age);

System.out.println("Address:"+address);

System.out.println("Gender:"+gender);

System.out.println("Salay:"+sal);

}

}

class FTEmploy extends Employ{

int basSal;

FTEmploy(String name,int age,String address,String gender,int basSal){

super(name,age,address,gender);

this.basSal=basSal;

}

void calSal(){

sal=(basSal+basSal\*0.75+basSal\*0.075-basSal\*0.1);

}

}

class PTEmploy extends Employ{

String qual;

int exp,numHour;

PTEmploy(String name,int age,String address,String gender,String qual,int exp,int numHour){

super(name,age,address,gender);

this.qual=qual;

this.exp=exp;

this.numHour=numHour;

}

void calSal(){

switch(qual){

case "BE":

if(exp<=5) sal=numHour\*300;

else if(exp<=10) sal=numHour\*400;

else sal=numHour\*500;

break;

case "MTech":

if(exp<=5) sal=numHour\*500;

else if(exp<=10) sal=numHour\*700;

else sal=numHour\*1000;

break;

case "PhD":

if(exp<=5) sal=numHour\*800;

else if(exp<=10) sal=numHour\*1200;

else sal=numHour\*1500;

break;

}

}

}

class TW1{

public static void main(String []args){

FTEmploy f1=new FTEmploy("Arun",25,"Vijayapur","Male",10000);

f1.calSal();

System.out.println("Details of Full time Employ");

f1.show();

PTEmploy e1=new PTEmploy("Rohit",30,"Belgum","Male","BE",6,10);

e1.calSal();

System.out.println("\nDetails of Part time Employ 1:");

e1.show();

PTEmploy e2=new PTEmploy("Rohini",26,"Mysore","Female","PhD",10,8);

e2.calSal();

System.out.println("\nDetails of Part time Employ 2:");

e2.show();

}

}

11) Design a base class called Employee who work for a Hospital and this class would have name, dob, address, salary and designation as the attributes. Add a constructor to initialize all these data members. This class would have reportForDuty method to display reporting time and date with a “Welcome” message to the employee. Devise two derived classes Doctor and Nurse to have expertise and yearsofExperience as data members respectively. Devise a method performDuty in each of these derived classes to print appropriate message depending on expertise of doctor and years of experience. For instance if the expertise of the Doctor is Surgeon and yearsofExperience >10 then print “Perform Heart Operation” else perform “Perform minor Surgery”. If his expertise is orthopedic and experience is more than 5 years “Perform surgery and Plastering” else “Perform Plastering”. If the nurse has more than 3 years experience in performDuty method print “Check BP, Sugar” and “Administer medicine” else print “Look after the patient”. Create a Hospital Class that has main method, and instantiate objects of Doctor and Nurse and Perform reportForDuty and performDuty and record the output. Add a static method generateReport(Employee e) that accepts object of employee type and prints in a tabular form, Name, dob, birthday salary and the designation.

import java.util.Calendar;

class employee

{

protected String name, address, designation;

protected int d,m,y;

protected double salary;

public employee(String name, String address, String designation, int d, int m, int y, double salary)

{

this.name = name;

this.address = address;

this.designation = designation;

this.d = d;

this.m = m;

this.y = y;

this.salary = salary;

}

public void reportForDuty()

{

Calendar cal = Calendar.getInstance();

System.out.println("Welcome");

System.out.println("Reporting time: "+cal.getTime());

System.out.println("Date: "+cal);

}

public static void generateReport(employee e)

{

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

System.out.println("Address: "+e.address);

System.out.println("Date of birth: "+e.d+"/"+e.m+"/"+e.y);

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

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

}

}

class doctor extends employee

{

String expertise;

int yearsOfExperience;

public doctor(String name, String address, String designation, int d, int m, int y, double salary, String expertise, int yearsOfExperience)

{

super(name,address,designation,d,m,y,salary);

this.expertise = expertise;

this.yearsOfExperience = yearsOfExperience;

}

public void performDuty()

{

if(expertise.equalsIgnoreCase("Surgeon") && yearsOfExperience > 10)

{

System.out.println("Expertise: "+"Perform Heart Operation");

}

else if(expertise.equalsIgnoreCase("Surgeon") && yearsOfExperience < 10)

{

System.out.println("Expertise: "+"Perform minor Operation");

}

else if(expertise.equalsIgnoreCase("orthopedic") && yearsOfExperience > 5)

{

System.out.println("Expertise: "+"Perform surgery and Plastering");

}

else if(expertise.equalsIgnoreCase("orthopedic") && yearsOfExperience < 5)

{

System.out.println("Expertise: "+"Perform Plastering");

}

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

}

}

class nurse extends employee

{

String expertise;

int yearsOfExperience;

public nurse(String name, String address, String designation, int d, int m, int y, double salary, String expertise, int yearsOfExperience)

{

super(name,address,designation,d,m,y,salary);

this.expertise = expertise;

this.yearsOfExperience = yearsOfExperience;

}

public void performDuty()

{

if(yearsOfExperience > 3)

{

System.out.println("Check BP, Sugar and Administer medicine");

}

else

{

System.out.println("Look after the patient");

}

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

}

}

class Termwork42

{

public static void main(String args[])

{

doctor doc1 = new doctor("paul ","C.T.S 10919 1st main 2nd cross, sadashiv nagar.","Senior surgeon",12,9,2001,1500000,"Surgeon",15);

employee.generateReport(doc1);

doc1.performDuty();

doctor doc2 = new doctor("ian","C.T.S 20935 5st main 10nd cross, Shastri nagar","Junior surgeon",1,1,2011,100000,"Surgeon",9);

employee.generateReport(doc2);

doc2.performDuty();

doctor doc3 = new doctor("joseph","C.T.S 45678 10st main 3nd cross, Shahpur.","Senior orthopedic",15,9,2002,1000000,"Orthopedic",5);

employee.generateReport(doc3);

doc3.performDuty();

doctor doc4 = new doctor("darren","C.T.S 30547 8st main 15nd cross, Azam nagar","Junior orthopedic",13,12,2009,100000,"Orthopedic",2);

employee.generateReport(doc4);

doc4.performDuty();

nurse n1 = new nurse("matt","C.T.S 50897 8st main 5nd cross, Ambedkar nagar","Senior Nurse",18,9,2001,10000,"Nurse",4);

employee.generateReport(n1);

n1.performDuty();

nurse n2 = new nurse("daniel","C.T.S 58005 12st main 25nd cross, shau nagar","Junior Nurse",13,10,2001,10000,"Nurse",2);

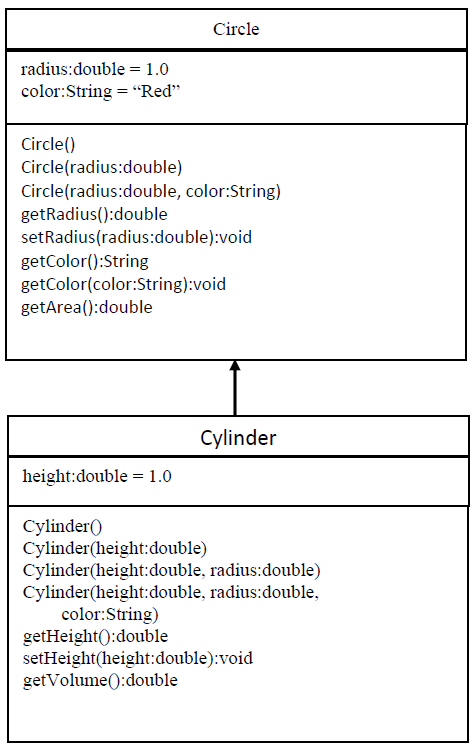
employee.generateReport(n2);

n2.performDuty();

}

}

12) The class Cylinder inherits all the instance variables (radius and color) and methods (getRadius(), getArea(), among others) from its superclass Circle. It further defines a variable called height, three methods getHeight(), setHeight() and getVolume() and its own constructors. Implement the hierarchy as shown below:



package tw4;

//superclass

class Circle{

double radius;

String color;

Circle(){

radius = 1.0;

color="Red";

}

Circle(double radius){

this.radius=radius;

color="Red";

}

Circle(double radius, String color){

this.radius=radius;

this.color=color;

}

double getRadius() {

return radius;

}

void setRadius(double radius) {

this.radius=radius;

}

String getColor() {

return color;

}

void setColor(String color) {

this.color=color;

}

double getArea() {

return (Math.PI\*radius\*radius);

}

}

//subclass

class Cylinder extends Circle{

double height;

Cylinder(){

super();

height=1.0;

}

Cylinder(double height){

super();

this.height=height;

}

Cylinder(double height, double radius){

super(radius);

this.height=height;

}

Cylinder(double height, double radius, String color){

super(radius,color);

this.height=height;

}

double getHeight() {

return height;

}

void setHeight(double height) {

this.height=height;

}

double getVolume() {

return (Math.PI\*radius\*radius\*height);

}

}

class TW1 {

public static void main(String[] args) {

Circle c=new Circle(3.0,"Violet");

System.out.println("Radius of circle = "+c.getRadius()+"\nColor of Circle = "+c.getColor());

c.setColor("Blue");

System.out.println("Changed Color of the cirlce : "+c.getColor()+(String.format("\nThe Area of the circle : %.2f",c.getArea())));

Cylinder c1=new Cylinder(3.0,4.0,"Green");

System.out.println("Radius of the Cylinder : "+c1.getRadius()+"\nHeight of the Cylinder : "+c1.getHeight()+"\nColor of the Cylinder : "+c1.getColor()+String.format("\nVolume of the Cylinder : %.2f",c1.getVolume()));

}

}

13) Create a Stack class having an integer array say elem and top\_of\_stack as instance variables. Define three overloaded methods having the following signatures:

a. initStack(int size) to create an array of specified size and initialize the top\_of\_stack

b. initStack(Stack another) to intialize the Stack object with state of the Stack object "another"

c. initStack(int [] a) to initialize contents of a[] to the instance variable elem.

Write following methods:

a. push(): Pushes the element onto the stack,

b. pop(): Returns the element on the top of the stack, removing it in the process, and

c. peek(): Returns the element on the top of the stack, but does not remove it.

Also write methods that check whether stack is full and stack is empty and return boolean value true or false appropriately.

class Stack{

int[] ele;

int top;

void initStack(int size){

ele=new int[size];

top=-1;

}

void initStack(Stack another){

ele=new int[another.ele.length];

top=-1;

for(int item:another.ele)

push(item);

}

void initStack(int[] a){

ele=new int[a.length];

top=-1;

for(int item:a)

push(item);

}

void push(int item){

if(top<ele.length){

ele[++top]=item;

System.out.println("Pushed element is "+item);

}

else

System.out.println("Stack overflow");

}

int pop(){

if(top==-1){

System.out.println("Stack underflow");

return -1;

}

else{

int item=ele[top--];

return item;

}

}

int peek(){

return ele[top];

}

}

public class TW5a {

public static void main(String[] args) {

Stack s1=new Stack();

Stack s2=new Stack();

s1.initStack(5);

s1.push(10);

s1.push(20);

s1.push(30);

s1.push(40);

s1.push(50);

s2.initStack(s1);

int[] array={1,2,3,4};

Stack s3=new Stack();

s3.initStack(array);

System.out.println("Popped element in S1 object is "+s1.pop());

System.out.println("Element on top of the stack of object s1 is "+s1.peek());

System.out.println("Element on top of the stack of object s2 is "+s2.peek());

}

}

14) Design a Vehicle class to include as data members, the vehicle’s initial velocity (u), final velocity (v) and acceleration (a). The class must a parameterized constructor to initialize these data members**.**

Design a class LawsofMotion and write two overloaded methods, **computeDistanceTravelled** and **computeAcceleration** to return the distance travelled by the vehicle and acceleration attained respectively, given the different parameter values. The class must have appropriate data members for the following methods to work.

computeDistanceTravelled(float t)- computes and returns the distance travelled as: u\*t+0.5f\*a\*t\*t

computeDistanceTravelled() - computes and returns the distance travelled as (( v\*v –u\*u)/(2\*a));

computeAcceleration(float mass, float force) - computes and returns the acceleration as a=mass/force;

computeAcceleration(float mass, float v, float u,float t) - computes and returns the acceleration as (m\*v-m\*u)/t;

Demonstrate the working by instantiating objects of the above class and verify the working of the overloaded methods.

class LawsofMotion

{

double u,v,a;

LawsofMotion()

{

u=0;

v=0;

a=0;

}

LawsofMotion(double U,double V,double A)

{

u=U;

v=V;

a=A;

}

double computeDistanceTravelled(float t)

{

return(u\*t+0.5\*a\*t\*t);

}

double computeDistanceTravelled()

{

return((v\*v-u\*u)/(2\*a));

}

double computeAcceleration(float mass, float force)

{

return(mass/force);

}

double computeAcceleration(float mass, float v, float u,float t)

{

return((mass\*v-mass\*u)/t);

}

}

class Vehicle

{

public static void main(String args[])

{

double result;

LawsofMotion l1=new LawsofMotion(10.5,20,5);

result=l1.computeDistanceTravelled(10);

System.out.println("Distance travelled using the formula (u\*t+0.5\*a\*t\*t):"+result);

result=l1.computeDistanceTravelled();

System.out.println("Distance travelled using the formula ((v\*v-u\*u)/(2\*a)):"+result);

result=l1.computeAcceleration(50,25);

System.out.println("Acceleration using the formula (mass/force):"+result);

result=l1.computeAcceleration(50,20,10,5);

System.out.println("Acceleration using the formula (m\*v-m\*u)/t):"+result);

}

}

**15)** Implement a sort function that illustrates overloading methods. Sort method sorts the array in the default ordering, sorts the array into the specified order, sorts array elements ranging from fromIndex to toIndex in the specified order.

import java.util.Arrays;

import java.util.Collections;

import java.util.Scanner;

class sort{

    void sorting(int []a){

        Arrays.sort(a);

        System.out.println("Default Ordering");

    }

    void sorting(int []a,int n){

        System.out.println("\nSpecified Ordering\n1:Ascending\n2:Descending");

        Scanner in=new Scanner(System.in);

        System.out.println("Enter your choice: ");

        int c=in.nextInt();

        switch(c){

            case 1:  Arrays.sort(a);

                     disp(a,n);

                     break;

            case 2:   Arrays.sort(a);

                      System.out.println("Sorted Array: ");

                      for(int i=n-1;i>=0;i--){

                          System.out.println(a[i]);

                      }

        }

    }

    void sorting(int []a,int n,int x,int y){

        int p=Math.min(x, y);

        int q=Math.max(x, y);

        int []temp=new int[(q-p)+1];

        int j=0;

        for(int i=p;i<=q;i++){

            temp[j]=a[i];

            j++;

        }

        Arrays.sort(temp);

        j=0;

        for(int i=p;i<=q;i++){

            a[i]=temp[j];

            j++;

        }

        System.out.println("fromIndex to toIndex");

        disp(a, n);

    }

    void disp(int []a,int n){

        System.out.println("Sorted Array: ");

                for(int i=0;i<n;i++){

            System.out.println(a[i]);

        }

    }

}

public class SortFunction {

    public static void main(String[] args) {

        int []a={7,8,4,5,2};

        int n=a.length;

        int x=1,y=4;

        sort s1=new sort();

        s1.sorting(a, n, x, y);

        sort s=new sort();

        s.sorting(a);

        s.disp(a, n);

        sort s2=new sort();

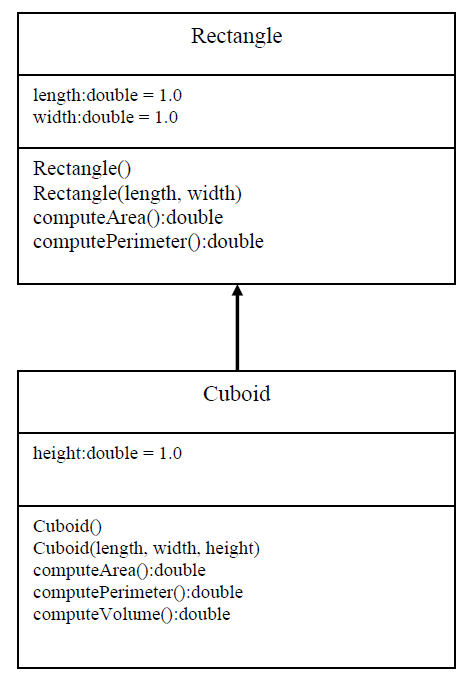
        s2.sorting(a,n);

    }

}

16) Implement the following class hierarchy. In the Cuboid class, override the method computeArea() and computePerimeter() of Rectangle class to compute the surface area and perimeter of a rectangle cuboid. Add a method computeVolume() in Cuboid class to compute volume of the cuboid. Assuming length, width and height as l, w and h respectively,

* • formula to find the surface area = 2(lw) + 2(hl) + 2(hw)
* • formula to find the perimeter = 2l + 2w
* • formula to find the volume = l x w x h



class Rectangle {

double length;

double width;

Rectangle() {

length = 1.0;

width = 1.0;

}

Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

double computeArea() {

return length \* width;

}

double computePerimeter() {

return 2 \* (length + width);

}

}

class Cuboid extends Rectangle {

double height;

Cuboid() {

super();

height = 1.0;

}

Cuboid(double length, double width, double height) {

super(length, width);

this.height = height;

}

@Override

double computeArea() {

return 2 \* ((length \* width) + (width \* height) + (length \* height));

}

@Override

double computePerimeter() {

return 4 \* (length + width + height);

}

double computeVolume() {

return length \* width \* height;

}

}

public class TW5b{

public static void main(String[] args) {

Rectangle r1 = new Rectangle();

System.out.println("Rectangle 1:");

System.out.println("Area:" + r1.computeArea());

System.out.println("Perimeter:" + r1.computePerimeter());

Rectangle r2 = new Rectangle(10,30);

System.out.println("\nRectangle 2:");

System.out.println("Area:" + r2.computeArea());

System.out.println("Perimeter:" + r2.computePerimeter());

Cuboid c1=new Cuboid();

System.out.println("\nCuboid 1:");

System.out.println("Area:" + c1.computeArea());

System.out.println("Perimeter:" + c1.computePerimeter());

System.out.println("Volume:" + c1.computeVolume());

Cuboid c2=new Cuboid(10,30,40);

System.out.println("\nCuboid 2:");

System.out.println("Surface area:" + c2.computeArea());

System.out.println("Perimeter:" + c2.computePerimeter());

System.out.println("Volume:" + c2.computeVolume());

}

}

17) Design a base class ArrayStack that uses array to hold the elements and has 3 methods namely, push, pop and display. Derive a class LinkedStack that overrides these 3 methods and uses linked list to implement stack. Demonstrate the working of both the classes by performing push, pop and display operations on the objects of the above to classes.

import java.io.\*;

import java.lang.\*;

import java.util.\*;

class ArrayStack{

int top,size;

int stack[];

ArrayStack(int s){

top=-1;

size=s;

stack=new int[size];

}

void push(int item){

if(top==size-1){

System.out.println("Stack Overflow!");

return;

}

stack[++top]=item;

System.out.println("Item Added : "+item);

}

void pop(){

if(top==-1){

System.out.println("Stack Underflow!");

return;

}

System.out.println("Item popped : "+stack[top--]);

}

void display(){

if(top==-1){

System.out.println("Stack Empty!");

return;

}

System.out.println("Stack items");

for(int i=top;i>=0;i--){

System.out.println("|"+stack[i]+"|");

}

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

}

}

class LinkedStack{

private class Node{

int data; // holds data

Node link; // reference variable Node Type

}

Node top; // Head Node

LinkedStack(){

this.top=null;

}

void push(int item){

Node temp = new Node();

temp.data = item;

temp.link = top;

top = temp;

System.out.println("Item Pushed : "+item);

}

void pop(){

if(top==null){

System.out.printf("\nStack Underflow");

return;

}

System.out.println("Item popped : "+top.data);

top = (top).link;

}

void display(){

if(top==null){

System.out.printf("\nStack Underflow");

return;

}

Node temp=top;

while(temp!=null){

System.out.println("|"+temp.data+"|");

temp = temp.link;

}

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

}

}

class TW1{

public static void main(String[]args){

ArrayStack s = new ArrayStack(10);

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

s.push(10);

s.push(14);

s.push(13);

s.push(74);

s.push(51);

s.display();

s.pop();

s.pop();

s.display();

LinkedStack c = new LinkedStack();

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

c.push(11);

c.push(21);

c.push(11);

c.push(74);

c.push(15);

c.display();

c.pop();

c.pop();

c.display();

}

}

18) Design an abstract class Car to have carName, chassiNum, modelName as member variables and add two abstract methods, startCar and operateSteering . Inherit MarutiCar and BmwCar from Car class and override the two abstract methods in their own unique way. Design a driver class to have driver name, gender and age as data members and add a method driveCar with abstract class reference variable as argument and invoke the two basic operations namely, startCar and operateStearing and demonstrate run-time polymorphism.

abstract class Car{

String carName,modelName;

int chassiNum;

Car(String carName,int chassiNum,String modelName){

this.carName=carName;

this.chassiNum=chassiNum;

this.modelName=modelName;

}

abstract void startCar();

abstract void operateSteering();

void display(){

System.out.println("Car Name:"+carName);

System.out.println("Chassi number:"+chassiNum);

System.out.println("Model Name:"+modelName);

}

}

class MarutiCar extends Car{

MarutiCar(String carName, int chassiNum, String modelName) {

super(carName, chassiNum, modelName);

}

void startCar(){

System.out.println("Starting a Maruti car....");

}

void operateSteering(){

System.out.println("This car is manually steered.....");

}

}

class BmwCar extends Car{

BmwCar(String carName, int chassiNum, String modelName) {

super(carName, chassiNum, modelName);

}

void startCar(){

System.out.println("Starting a BMW car....");

}

void operateSteering(){

System.out.println("This car is automatically steered.....");

}

}

class Driver{

String name,gender;

int age;

Driver(String name,int age,String gender){

this.name=name;

this.age=age;

this.gender=gender;

}

void driveCar(Car obj){

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

System.out.println("Age:"+age);

System.out.println("Gender:"+gender);

obj.display();

obj.startCar();

obj.operateSteering();

}

}

public class Tw6 {

public static void main(String []args){

MarutiCar m=new MarutiCar("Suzuki",1253,"A21s");

BmwCar b=new BmwCar("BMW5",4596,"S5");

Driver d1=new Driver("Vishal",25,"Male");

d1.driveCar(m);

System.out.println();

Driver d2=new Driver("Priya",23,"Female");

d2.driveCar(b);

}

}

19) Write a Java program that defines an abstract class called Account and accepts the following customer account information: **1) Customer Name 2) Account Number 3) Balance** and provides below operations on customer account: **1) Deposit 2) Withdraw 3) Display Balance 4) Display full account details**

There are two types of accounts – Savings and Current. The Current account has an overdraft facility limited to Rs. 75,000 per account. The following constraints hold on these accounts:

Savings Account:

1) The total number of deposits for a Savings account cannot exceed three per day.

2) The amount deposited into a savings account cannot exceed Rs.5000 in each transaction.

3) The maximum withdrawal amount is Rs.25,000 per transaction.

4) The minimum balance to be maintained is Rs. 10,000.

Current account:

1) The amount withdrawn cannot exceed the overdraft limit once the account balance is zero.

2) Maximum number of withdrawals is two.

3) No limit on the number of deposits.

4) Each deposit cannot exceed Rs. 25,000.

Test the program by creating objects of the Savings and Current Accounts

abstract class Account {

protected String cusName,acNo;

protected float bal;

public abstract void Deposit(float dep);

public abstract void Withdraw(float wdw);

public void dispBal() {

System.out.println("Balance="+bal);

}

public void disp() {

System.out.println("Customer details are:");

System.out.println("Customer Name:"+cusName+"\nAccount No.:"+acNo);

System.out.println("Balance:"+bal);

}

}

class Savings extends Account {

private int n=0;

public Savings(String c,String ac,float b) {

cusName=c;

acNo=ac;

bal=b;

}

public void Deposit(float dep) {

n++;

if(n<=3) {

if(dep<=5000) {

bal=bal+dep;

System.out.println("Amount has been deposited");

dispBal();

}

else

System.out.println("Amount exceeds the limit");

}

else

System.out.println("You've reached maximum limit of deposition");

}

public void Withdraw(float wdw) {

if((bal-wdw)<10000)

System.out.println("Insufficient balance");

else {

if(wdw>25000)

System.out.println("Amount exceeds maximum withdrawal limit");

else {

bal=bal-wdw;

System.out.println("Amount successfully withdrawn");

dispBal();

}

}

}

}

class Current extends Account {

private int n=0;

public Current(String c,String ac,float b) {

cusName=c;

acNo=ac;

bal=b;

}

public void Deposit(float dep) {

bal=bal+dep;

System.out.println("Amount has been deposited");

dispBal();

}

public void Withdraw(float wdw) {

n++;

if(n<=2) {

if((bal-wdw)<-75000) {

n--;

System.out.println("Amount exceeds overdraft amount");

}

else {

bal=bal-wdw;

System.out.println("Amount successfully withdrawn");

dispBal();

}

}

else

System.out.println("You've exceeded withdrawal limit");

}

}

class tw6point3 {

public static void main(String[] args) {

Savings s=new Savings("Finneas","221B",28000);

s.disp();

System.out.println("Account Type:Savings");

s.Deposit(2000);

s.Deposit(1000);

s.Deposit(2500);

s.Deposit(3000);

s.Withdraw(1000);

s.disp();

System.out.println();

Current c=new Current("Elton","241B",0);

c.disp();

System.out.println("Account Type:Current");

c.Withdraw(20000);

c.Withdraw(56000);

c.Withdraw(1000);

c.Withdraw(1000);

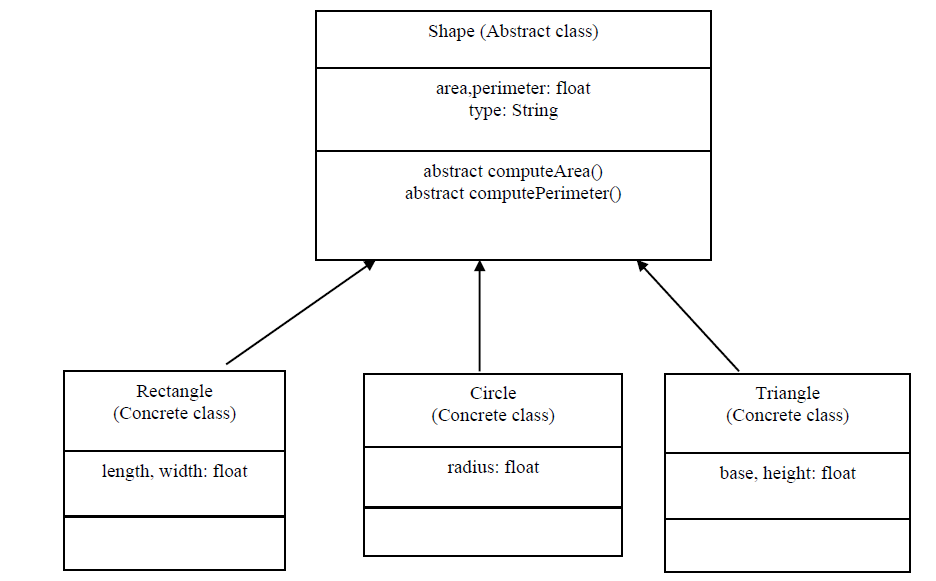
c.Deposit(30000);

c.disp();

}

}

20) Implement the following inheritance hierarchy.



import java.util.Scanner;

abstract class Shape{

double area;

double perimeter;

String type;

abstract void computeArea();

abstract void computePerimeter();

}

class Rectangle extends Shape{

double length, width;

Rectangle(){

Scanner in = new Scanner(System.in);

System.out.println("Enter the length and width: ");

length = in.nextDouble();

width = in.nextDouble();

}

void computeArea() {

area = length \* width;

System.out.println("Area of the rectangle is: " + area);

}

void computePerimeter() {

perimeter = 2 \* (length + width);

System.out.println("Perimeter of the rectangle is: " + perimeter);

}

}

class Circle extends Shape{

double radius;

Circle(){

Scanner in = new Scanner(System.in);

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

radius = in.nextDouble();

}

void computeArea() {

area = Math.PI \* radius \* radius;

System.out.println("Area of the circle is: " + area);

}

void computePerimeter() {

perimeter = 2 \* Math.PI \* radius;

System.out.println("Perimeter of the circle is: " + perimeter);

}

}

class Triangle extends Shape{

double a, b, c, s;

Triangle(){

Scanner in = new Scanner(System.in);

System.out.println("Enter the 3 sides: ");

a = in.nextDouble();

b = in.nextDouble();

c = in.nextDouble();

}

void computeArea() {

s = (a + b + c) / 2.0;

area = Math.sqrt(s \* (s-a) \* (s-b) \* (s-c));

System.out.println("Area of the triangle is: " + area);

}

void computePerimeter() {

perimeter = a + b + c;

System.out.println("Perimeter of the triangle is: " + perimeter);

}

}

public class TW1 {

public static void main(String[] args) {

Rectangle r = new Rectangle();

r.computeArea();

r.computePerimeter();

Circle c = new Circle();

c.computeArea();

c.computePerimeter();

Triangle t = new Triangle();

t.computeArea();

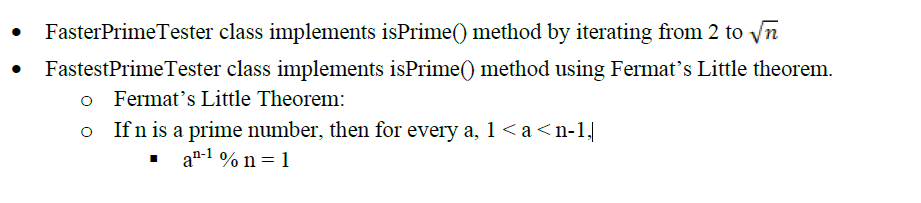
t.computePerimeter();

}

}

21) Write a Java application to implement the following:

* • PrimeTester class implements isPrime() method by iterating from 2 to n-1 for a given number n
* • ImprPrimeTester class implements isPrime() method by iterating from 2 to n/2



interface IsPrime{

boolean isPrime(int n);

}

class PrimeTester implements IsPrime{

@Override

public boolean isPrime(int n){

boolean flag=true;

for(int i=2; i<=n-1; i++){

if((n%i)==0)

{

flag=false;

break;

}

}

return flag;

}

}

class ImprPrimeTester implements IsPrime{

@Override

public boolean isPrime(int n){

boolean flag=true;

for(int i=2; i<=n/2; i++){

if((n%i)==0)

{

flag=false;

break;

}

}

return flag;

}

}

class FasterPrimeTester implements IsPrime{

@Override

public boolean isPrime(int n){

boolean flag=true;

for(int i=2; i<Math.sqrt(n); i++){

if((n%i)==0)

{

flag=false;

break;

}

}

return flag;

}

}

class FastestPrimeTester implements IsPrime{

@Override

public boolean isPrime(int n){

int a=2;

return Math.pow(a,n-1)%n==1;

}

}

public class TW7 {

public static void main(String[] args) {

PrimeTester p1=new PrimeTester();

ImprPrimeTester p2=new ImprPrimeTester();

FasterPrimeTester p3=new FasterPrimeTester();

FastestPrimeTester p4=new FastestPrimeTester();

System.out.println("32 is Prime? "+p1.isPrime(32));

System.out.println("17 is Prime? "+p1.isPrime(17));

System.out.println("32 is Prime? "+p2.isPrime(32));

System.out.println("17 is Prime? "+p2.isPrime(17));

System.out.println("32 is Prime? "+p3.isPrime(32));

System.out.println("17 is Prime? "+p3.isPrime(17));

System.out.println("32 is Prime? "+p4.isPrime(32));

System.out.println("17 is Prime? "+p4.isPrime(17));

}

}

**22)** Write a JAVA program which has: i. An Interface class for Stack Operations (viz., push(), pop(), peek(),display()) ii. A Class that implements the Stack Interface and creates a fixed length Stack. iii. A Class that implements the Stack Interface and creates a Dynamic Length Stack. iv. A Class that uses both the above Stacks through Interface reference and does the Stack operations that demonstrates the runtime binding.

interface stackop

{

void push(int item);

int pop();

}

class FixedStack implements stackop

{

private int stk[ ];

private int top;

FixedStack(int size)

{

stk=new int[size];

top=-1;

}

public void push(int item)

{

if(top==stk.length-1)

{

System.out.println("Stack Overflows");

int t[ ]=new int[stk.length \* 2];

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

t[i]=stk[i];

stk=t;

stk[++top]=item;

}

else stk[++top]=item;

}

public int pop()

{

if(top<0)

{

System.out.println("Stack Underflows");

return 0;

}

else return stk[top--];

}

}

class DynStack implements stackop

{

private int stk[ ];

private int top;

DynStack(int size)

{

stk=new int[size];

top=-1;

}

public void push(int item)

{

if(top==stk.length-1)

{

System.out.println("Stack Overflows.");

int t[ ]=new int[stk.length \* 2];

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

t[i]=stk[i];

stk=t;

stk[++top]=item;

}

else

stk[++top]=item;

}

public int pop()

{

if(top<0)

{

System.out.println("Stack Underflows.");

return 0;

}

else return stk[top--];

}

}

class StackTest

{

public static void main(String args[ ])

{

FixedStack fs=new FixedStack(3);

DynStack ds=new DynStack(5);

stackop mystk;

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

fs.push(i);

System.out.println("Fixed length Stack Contents are.");

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

System.out.println(fs.pop());

for(int i=0;i<7;i++)

ds.push(i);

System.out.println("Dynamic length Stack Contents are");

for(int i=0;i<7;i++)

System.out.println(ds.pop());

mystk=fs;

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

mystk.push(i);

System.out.println("Fixed length Stack Contents using interface reference");

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

System.out.println(mystk.pop());

mystk=ds;

for(int i=0;i<7;i++)

mystk.push(i);

System.out.println("Dynamic length Stack Contents using interface reference");

for(int i=0;i<7;i++)

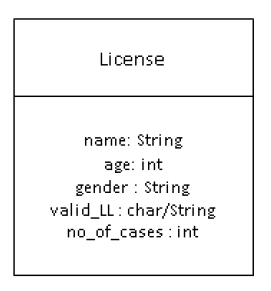
System.out.println(mystk.pop());

}

}

23) Assume that you have received a request from the transport authority for automating the task of issuing the permanent license for two wheelers. The mandatory condition to issue the license are: 1) the applicant must over 18 years of age and 2) holder of a valid learner’s license and 3) no accident cases in the last one year.

Write a Java program that reads user details as required (use the Scanner class). Create user defined exceptions to check for the three conditions imposed by the transport authority. Based on the inputs entered by the user, decide and display whether or not a license has to be issued or an error message as defined by the user exception.



import java.util.Scanner;

class UnderAgeException extends Exception{

UnderAgeException(String s){

super(s);

}

@Override

public String toString(){

return "Sorry. You are too young for the license";

}

}

class ValidLLR extends Exception{

ValidLLR(String s){

super(s);

}

@Override

public String toString(){

return "Sorry. You do not hold a valid LLR";

}

}

class NumAccidents extends Exception{

NumAccidents(String s){

super(s);

}

@Override

public String toString(){

return "Sorry. There are accidents in last one year";

}

}

class License{

String name;

int age, no\_of\_cases;

char gender;

char validLLR;

void readData(){

Scanner in=new Scanner(System.in);

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

name = in.nextLine();

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

age = in.nextInt();

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

gender = in.next().charAt(0);

System.out.println("Do you have Valid LLR (Y/N)? ");

validLLR = in.next().charAt(0);

System.out.println("How many number of cases in past one year? ");

no\_of\_cases = in.nextInt();

}

}

public class TW8 {

public static void main(String[] args) {

License applicant =new License();

applicant.readData();

validateApplicant(applicant);

}

static void validateApplicant(License a){

try{

if(a.age<18)

throw new UnderAgeException("Underageexception:");

if(a.validLLR!='Y')

throw new ValidLLR("ValidLLRexception:");

if(a.no\_of\_cases>0)

throw new NumAccidents("Numberofaccidentsexception:");

System.out.println("Congrats!! Your license is being posted");

}

catch(UnderAgeException e){

System.out.println(e.getMessage()+e);

}

catch(ValidLLR e){

System.out.println(e.getMessage()+e);

}

catch(NumAccidents e){

System.out.println(e.getMessage()+e);

}

catch(Exception e){

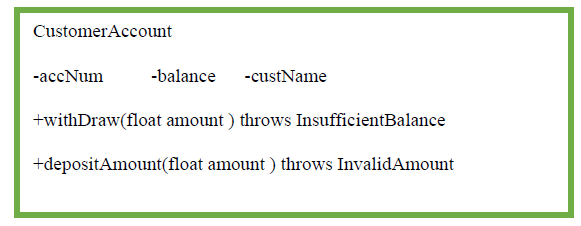
System.out.println(e.getMessage()+e);

}

}

}

24) Design a class CustomerAccount that has acctNum, custName and balance as member variables and a constructor to initialize these. Implement withDraw and depositAmount methods that accepts amount as argument and it must throw an user defined exception called InsufficientBalalance/InvalidAmount exception when amount is greater than balance/ amount is negative respectively. Design two classes InsufficientBalance and InvalidAmount that extend the Exception class and override toString method. Demonstrate the working of the user defined exceptions by instantiating an object of customerAccount class and invoking withDraw and depoistAmount in try… catch... finally block.



import java.util.Scanner;

class CustomerAccount{

int accNo;

String custName;

double balance;

CustomerAccount(int accNo,String custName,double balance){

this.accNo = accNo;

this.custName = custName;

this.balance = balance;

}

void withdrawAmt(double Amt)

{

try{

if(balance-Amt <0)

throw new InsufficientBalException("Insufficient Balance");

balance-=Amt;

System.out.println("Balance in the account is "+ balance);

}

catch (InsufficientBalException e)

{

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

System.out.println(e);

System.out.println();

}

}

void depositAmt(double Amt)

{

try

{

if(Amt<0)

throw new InvalidAmtException("Invalid amount");

balance+=Amt;

System.out.println("Balance in the account is "+ balance);

}

catch( InvalidAmtException e)

{

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

System.out.println(e);

System.out.println();

}

}

}

class InsufficientBalException extends Exception{

InsufficientBalException(String msg)

{

super(msg);

}

public String toString()

{

return "Amount can't be withdrawm as there is insufficient Balance";

}

}

class InvalidAmtException extends Exception{

InvalidAmtException(String msg)

{

super(msg);

}

public String toString()

{

return "Can't deposit as amount is negative";

}

}

class Main {

public static void main(String[] args) {

CustomerAccount c =new CustomerAccount(101,"Harry Styles",1000);

c.depositAmt(-500);

c.withdrawAmt(1200);

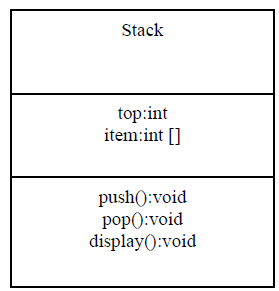
c.depositAmt(1500);

c.withdrawAmt(1000);

}

}

25) Design a class to implement Stack data structure. The method push() throws a custom exception called OverflowException if the item cannot by pushed on to the stack. The method pop() throws a custom exception called UnderflowException if the stack is empty. Design two classes OverflowException and UnderflowException that extend the Exception class and override the getMessage() method. Develop a driver class to demonstrate the working of the custom exceptions by instantiating an object of Stack class and performing push() and pop() operations.



package termworks;

class myStack{

int data[],top;

myStack(int n)

{

data = new int[n];

top = -1;

}

void push(int x)

{

try {

if(isFull())

{

throw new OverFlowException("Stack Overflow");

}

} catch (OverFlowException e) {

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

System.out.println(e + "\n");

}

System.out.println("Element pushed is: " + x);

data[++top] = x;

}

int pop()

{

try {

if(isEmpty())

{

throw new UnderFlowException("Stack Underflow");

}

} catch (UnderFlowException e) {

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

System.out.println(e + "\n");

}

return data[top--];

}

boolean isEmpty()

{

return top == -1;

}

boolean isFull()

{

return top == data.length-1;

}

}

class OverFlowException extends Exception{

OverFlowException(String e)

{

super(e);

}

public String toString()

{

return "OverFlow Exception";

}

}

class UnderFlowException extends Exception{

UnderFlowException(String e)

{

super(e);

}

public String toString()

{

return "UnderFlow Excwption";

}

}

class tw25 {

public static void main(String[] args) {

myStack s = new myStack(4);

s.push(10);

s.push(20);

s.push(30);

s.push(40);

s.push(60);

System.out.println("The popped element is: " + s.pop());

System.out.println("The popped element is: " + s.pop());

System.out.println("The popped element is: " + s.pop());

System.out.println("The popped element is: " + s.pop());

System.out.println("The popped element is: " + s.pop());

}

}

26) Read a string containing 3\_4 words using Scanner class object. Split it into words and for each word check if it's palindrome by writing a function isPalindrome(String the myWord, int s, int e) which return true if its palindrome else return false. Where s is start index and e is end index of the input myWord. Print it in uppercase if it is palindrome else reverse the string and print it in lowercase. Use appropriate string functions to implement the above problem statement.

import java.util.\*;

public class TW1 {

public static void main(String[] args) {

String inputLine; String[] allWords;

Scanner sc=new Scanner(System.in);

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

inputLine=sc.nextLine();

allWords=inputLine.split(" ");

for(String s : allWords){

if(isPalindrome(s,0,s.length()-1)){

System.out.println(s.toUpperCase());

System.out.println("It is Palindrome");

}

else{

System.out.println(reverseString(s).toLowerCase());

System.out.println("It is not Palindrome");

}

}

}

public static boolean isPalindrome(String myWord, int s, int t ){

if(myWord.charAt(s)== myWord.charAt(t)) {

if(s<t)

return isPalindrome(myWord,s+1,t-1);

else

if(s==t)

return true;

}

return false;

}

public static String reverseString(String s){

String rS="";

for(int i=s.length()-1;i>=0;i--)

rS=rS+s.charAt(i);

return rS;

}

}

27) Two strings will be anagram to each other if and only if they contain the same number of characters (order of the characters doesn't matter). That is, If the two strings are anagram to each other, then one string can be rearranged to form the other string. For Example: creative and reactive are anagrams. Write a Java program to test whether two strings are anagrams or not. (listen and silent, stressed and desserts, dusty and study)

import java.util.Arrays;

public class TW1 {

public static void main(String[] args) {

String s1 = "dusty";

String s2 = "study";

checkAnagram(s1, s2);

}

static void checkAnagram(String s1, String s2) {

char [] c1 = s1.toCharArray();

char [] c2 = s2.toCharArray();

if(c1.length != c2.length) {

System.out.println(s1 + " and " + s2 + " are " + "not Anagrams");

}

else {

Arrays.sort(c1);

Arrays.sort(c2);

if(Arrays.equals(c1, c2)) {

System.out.println(s1 + " and " + s2 + " are " + "Anagrams");

}

else {

System.out.println(s1 + " and " + s2 + " are " + "not Anagrams");

}

}

}

}

28) Write a Java program that creates a simple book database (use an array of N objects). Each book is represented with a ID, title, author (First Name & last name), Genre (category –technical, Sci Fi, Fiction, Comedy etc) and a Publisher name. Define methods to perform the following tasks:

1. Given a title, returns a status to indicate whether or not the book exists in database.

2. Given a string “str”, lists the details of all the books whose title contains str.

3. Given a genre, lists publishers who have published books in that genre. Q2

4. Given a character “c”, lists all authors who name starts with “c”.

class book{

int id;

String title, author, genre, publisher;

book(int id, String title, String author, String genre, String publisher ){

this.id = id;

this.title = title;

this.author = author;

this.genre = genre;

this.publisher = publisher;

}

}

class TW9 {

public static void main(String[] args){

book[] b = new book[4];

b[0] = new book(101, "Java Programming", "james Gosling", "Technical", "PHI");

b[1] = new book(102, "Java Made Easy", "Dennis Ritchie", "Technical", "MIT");

b[2] = new book(103, "Programming with C", "Balaguruswamy", "Technical", "VTU");

b[3] = new book(104, "Programming with C++", "Balaguruswamy", "Technical", "PHI");

if(checkTitle(b,"Fun with Python") == true)

System.out.println("The title exits in the database");

else

System.out.println("The title does not exits in the database");

listTitles(b, "Java");

checkGenre(b, "Comedy");

listAuthors(b, 'D');

}

static boolean checkTitle(book[] b , String title){

for(book b1: b)

if(b1.title.equals(title) == true)

return true;

return false;

}

static void listTitles(book[] b , String str){

int i=0;

System.out.println(str + " appears in the following titles: ");

for(book b1: b){

if(b1.title.indexOf(str) != -1){

System.out.println(b1.title);

i++;

}

}

if(i==0)

System.out.println("Not Found");

}

static void checkGenre(book[] b , String genre){

int i=0;

System.out.println(" Following publishers have published books on " + genre + ":");

for(book b1: b){

if(b1.genre.equals(genre) == true){

System.out.println(b1.publisher);

i++;

}

}

if(i==0)

System.out.println("Not Found");

}

static void listAuthors(book[] b , char c){

int i=0;

System.out.println(" Auhtors whose names start with " + c + ":");

for(book b1: b){

if(b1.author.charAt(0) == c){

System.out.println(b1.author);

i++;

}

}

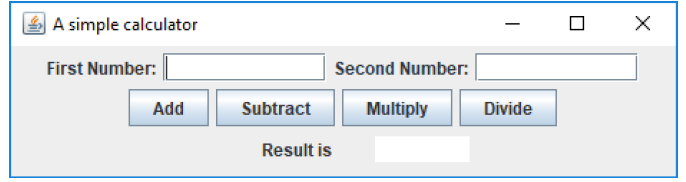
if(i==0)

System.out.println("Not Found");

}

}

29) Design and develop a GUI application as shown below. Assume the two numbers to be integers. The application must check for invalid division condition and throw an appropriate exception.



import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JFrame;

import static javax.swing.JFrame.EXIT\_ON\_CLOSE;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class Tw10 extends JFrame implements ActionListener{

JLabel lb1,lb2,lb3;

JTextField num1,num2,num3;

JButton b1,b2,b3,b4;

Tw10(){

this.setTitle("My first GUI");

this.setSize(400,400);

this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

this.setLayout(null);

lb1=new JLabel("First Number");

lb1.setBounds(60, 100, 110, 30);

this.add(lb1);

num1=new JTextField();

num1.setBounds(160, 100, 110, 30);

this.add(num1);

lb2=new JLabel("Second Number");

lb2.setBounds(60, 140, 110, 30);

this.add(lb2);

num2=new JTextField();

num2.setBounds(160, 140, 110, 30);

this.add(num2);

b1=new JButton("ADD");

b1.setBounds(100, 200, 100, 30);

b1.addActionListener(this);

this.add(b1);

b2=new JButton("SUBTRACT");

b2.setBounds(210, 200, 100, 30);

b2.addActionListener(this);

this.add(b2);

b3=new JButton("MULTIPLY");

b3.setBounds(100, 250, 100, 30);

b3.addActionListener(this);

this.add(b3);

b4=new JButton("DIVIDE");

b4.setBounds(210, 250, 100, 30);

b4.addActionListener(this);

this.add(b4);

lb3=new JLabel("Result");

lb3.setBounds(60,300, 110, 30);

this.add(lb3);

num3=new JTextField();

num3.setBounds(160, 300, 110, 30);

this.add(num3);

this.setVisible(true);

}

public static void main(String[] args){

new Tw10();

}

@Override

public void actionPerformed(ActionEvent ae) {

int res;

int a=Integer.parseInt(num1.getText());

int b=Integer.parseInt(num2.getText());

if(ae.getActionCommand()=="ADD"){

res=a+b;

num3.setText(""+res);

}

else if(ae.getActionCommand()=="SUBTRACT"){

res=a-b;

num3.setText(""+res);

}

else if(ae.getActionCommand()=="MULTIPLY"){

res=a\*b;

num3.setText(""+res);

}

else if(ae.getActionCommand()=="DIVIDE"){

try{

res=a/b;

num3.setText(""+res);

}catch(Exception e){

num3.setText("Division by Zero error");

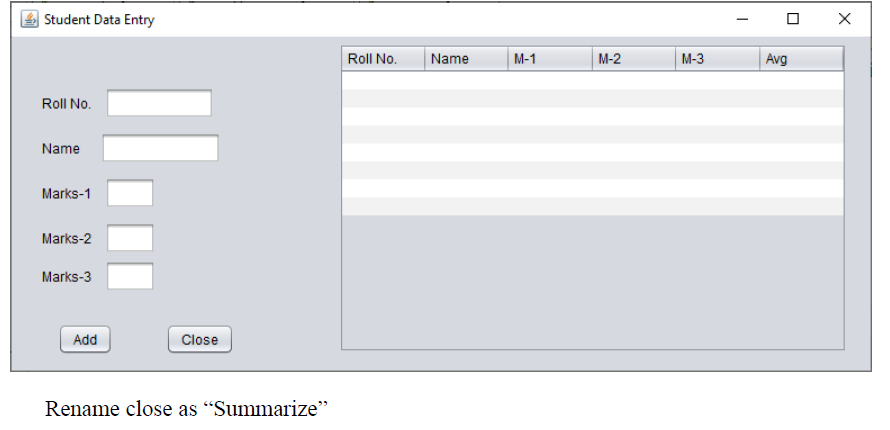
}

}

}

}

30) Design and develop a GUI application to accepts student details as shown in the GUI below. Allow the user to add records one after the other. Once the user clicks on close, compute the average score and display the details using JTable component.



package termwork\_10\_pp1; import java.util.ArrayList; import java.awt.EventQueue; import javax.swing.JFrame; import java.awt.FlowLayout; import javax.swing.JLabel; import javax.swing.JTextField; import javax.swing.JTable;

import javax.swing.table.DefaultTableModel; import javax.swing.JButton;

import java.awt.event.ActionListener; import java.awt.event.ActionEvent;

class GUI3 extends javax.swing.JFrame{

class StudentRecord{

String name;

int rollNo, m1, m2, m3;

StudentRecord(int rollNo, String name, int m1, int m2, int m3){

this.name = name;

this.rollNo = rollNo;

this.m1 = m1;

this.m2 = m2;

this.m3 = m3;

}

}

ArrayList<StudentRecord> records = new ArrayList<StudentRecord>();

private JFrame frame;

private JTextField textField;

private JTextField textField\_1;

private JTextField textField\_2;

private JTextField textField\_3;

private JTextField textField\_4;

private JTable table;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

GUI3 window = new GUI3();

window.frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the application.

\*/

public GUI3() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

frame = new JFrame();

frame.setBounds(100, 100, 450, 300);

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

frame.getContentPane().setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));

JLabel jbl1 = new JLabel("Roll No");

frame.getContentPane().add(jbl1);

textField = new JTextField();

frame.getContentPane().add(textField);

textField.setColumns(10);

JLabel jbl2 = new JLabel("Name");

frame.getContentPane().add(jbl2);

textField\_1 = new JTextField();

frame.getContentPane().add(textField\_1);

textField\_1.setColumns(10);

JLabel label = new JLabel("");

frame.getContentPane().add(label);

JLabel jbl3 = new JLabel("Marks-1");

frame.getContentPane().add(jbl3);

JLabel label\_1 = new JLabel("");

frame.getContentPane().add(label\_1);

textField\_2 = new JTextField();

frame.getContentPane().add(textField\_2);

textField\_2.setColumns(10);

JLabel label\_2 = new JLabel("");

frame.getContentPane().add(label\_2);

JLabel jbl4 = new JLabel("Marks-2");

frame.getContentPane().add(jbl4);

textField\_3 = new JTextField();

frame.getContentPane().add(textField\_3);

textField\_3.setColumns(10);

JLabel jbl5 = new JLabel("Marks-3");

frame.getContentPane().add(jbl5);

JLabel label\_3 = new JLabel("");

frame.getContentPane().add(label\_3);

textField\_4 = new JTextField();

frame.getContentPane().add(textField\_4);

textField\_4.setColumns(10);

JButton b1 = new JButton("Add");

b1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

StudentRecord s1 = new StudentRecord(Integer.parseInt(textField.getText()), textField\_1.getText(), Integer.parseInt(textField\_2.getText()), Integer.parseInt(textField\_3.getText()), Integer.parseInt(textField\_4.getText()));

records.add(s1);

textField.setText("");

textField\_1.setText("");

textField\_2.setText("");

textField\_3.setText("");

textField\_4.setText("");

}

});

JLabel label\_4 = new JLabel("");

frame.getContentPane().add(label\_4);

frame.getContentPane().add(b1);

JButton b2 = new JButton("Summarize");

b2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int rowCount = 0;

for(StudentRecord s : records) {

table.setValueAt(s.rollNo, rowCount, 0);

table.setValueAt(s.name, rowCount, 1);

table.setValueAt(s.m1, rowCount, 2);

table.setValueAt(s.m2, rowCount, 3);

table.setValueAt(s.m3, rowCount, 4);

float avg = (s.m1 + s.m2 + s.m3) / 3.0f;

table.setValueAt(String.format("%.2f", avg), rowCount, 5);

rowCount++;

}

}

});

JLabel label\_5 = new JLabel("");

frame.getContentPane().add(label\_5);

frame.getContentPane().add(b2);

table = new JTable();

table.setModel(new DefaultTableModel(

new Object[][] {

{null, null, null, null, null, null},

{null, null, null, null, null, null},

{null, null, null, null, null, null},

{null, null, null, null, null, null},

{null, null, null, null, null, null},

{null, null, null, null, null, null},

{null, null, null, null, null, null},

{null, null, null, null, null, null},

},

new String[] {

"Roll No", "Name", "Marks 1", "Marks 2", "Marks 3", "Average"

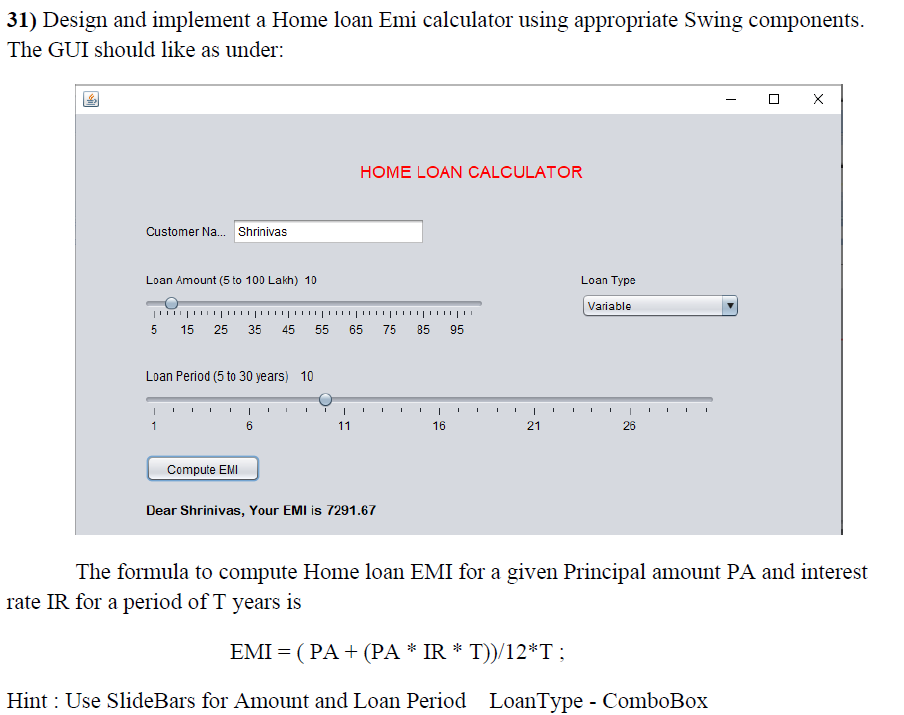
}

));

frame.getContentPane().add(table);

}

}



import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

class emiCalculator extends JFrame {

private JTextField jtfInvestmentAmount;

private JTextField jtfAnnualInterestRate;

private JTextField jtfNumberOfYears;

private JTextField jtfFutureValue;

private JButton jbtCompute;

private JButton jbtReset;

public emiCalculator() {

setTitle("Loan Calculator");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(500,300);

setLocationRelativeTo(null);

setLayout(new GridLayout(5,2,5,5));

JLabel jlInvestmentAmount = new JLabel(" Investment Amount");

JLabel jlNumberOfYears = new JLabel(" Number of Years");

JLabel jlAnnualInterestRate = new JLabel(" Annual Interest Rate");

JLabel jlFutureValue = new JLabel(" Future Value");

jtfInvestmentAmount = new JTextField();

jtfNumberOfYears = new JTextField();

jtfAnnualInterestRate = new JTextField();

jtfFutureValue = new JTextField();

jtfFutureValue.setEditable(false);

jbtCompute = new JButton("Compute");

jbtReset = new JButton("Reset");

add (jlInvestmentAmount);

add (jtfInvestmentAmount);

add (jlNumberOfYears);

add (jtfNumberOfYears);

add (jlAnnualInterestRate);

add (jtfAnnualInterestRate);

add (jlFutureValue);

add (jtfFutureValue);

add (jbtCompute);

add (jbtReset);

ListenerClass listener = new ListenerClass();

jbtCompute.addActionListener(listener);

jbtReset.addActionListener(listener);

setVisible(true);

}

public static void main(String[] args) {

new emiCalculator();

}

private void computeValue() {

try {

double annualInterestRate = Double.parseDouble(jtfAnnualInterestRate.getText());

double monthlyInterestRate = annualInterestRate / 1200.0;

int NumberOfYears = Integer.parseInt(jtfNumberOfYears.getText());

double investmentAmount = Double.parseDouble(jtfInvestmentAmount.getText());

double futureValue = investmentAmount \* Math.pow(1.0 + monthlyInterestRate, NumberOfYears \* 12);

jtfFutureValue.setText(String.format("%.2f", futureValue));

} catch (Exception e) {

JOptionPane.showMessageDialog(null, " Please enter numeric values.");

}

}

private void resetForm() {

jtfInvestmentAmount.setText("");

jtfAnnualInterestRate.setText("");

jtfNumberOfYears.setText("");

jtfFutureValue.setText("");

}

private class ListenerClass implements ActionListener {

public void actionPerformed(ActionEvent e) {

if (e.getSource() == jbtCompute) {

computeValue();

}

if (e.getSource() == jbtReset) {

resetForm();

}

}

}

}