|  |  |
| --- | --- |
| **Name** | **Virinchi Sadashiv Shettigar** |
| **UID no.** | **2021300118** |
| **Experiment No.** | 3 |

|  |  |
| --- | --- |
| **AIM:** | Programs on Polymorphism: Implement a Program to demonstrate method overloading and constructor overloading. |
| **Program 1** | |
| **PROBLEM STATEMENT:** | **Create a Date class with data int year, int month, int date,  int hrs, int min, int sec. Create a default, no-argument constructor which sets the default date to January 1, 2000, 00:00:00 Create 3 overloaded setter methods**  **void setDate(int year, int month, int date)**  **void setDate(int year, int month, int date, int hrs, int min)**  **void setDate(int year, int month, int date, int hrs, int min, int sec)**  **Also write a displayDate() method which will display the date depending on the type of date object created** |
| **PROGRAM:** | import java.util.\*;  class Date {  int year;  int month;  int date;  int hrs;  int min;  int sec;  Date() {  year = 2000;  month = 01;  date = 01;  hrs = 00;  min = 00;  sec = 00;  }  void setDate(int y,int m,int d) {  year = y;month = m;date = d;  }  void setDate(int y,int m,int d,int h,int mi) {  year = y;month = m;date = d;hrs = h;min = mi;  }  void setDate(int y,int m,int d,int h,int mi,int s) {  year = y;month = m;date = d;hrs = h;min = mi;sec = s;  }  void displayDate(int n) {  if(n==1) {  System.out.printf("Date: %d/%d/%d\n",date,month,year);  }  else if(n==2) {  System.out.printf("Date: %d/%d/%d, %d:%d\n",date,month,year,hrs,min);  }  else if(n==3) {  System.out.printf("Date: %d/%d/%d, %d:%d:%d\n",date,month,year,hrs,min,sec);  }  }  public static void main( String[] args){  Scanner sc= new Scanner(System.in);  Date d1= new Date();  Date d2= new Date();  Date d3= new Date();  int choice,flag;  int year,month,date,hrs,min,sec;  while(true) {  System.out.println("Welcome to date displayer");  System.out.println("Select any 1 Format\n1 -> DD/MM/YYYY\n2 -> DD/MM/YYYY, 00:00\n3 -> DD/MM/YYYY, 00:00:00");  choice = sc.nextInt();  System.out.println("Enter Year,Month and Date: ");  year = sc.nextInt();  month = sc.nextInt();  date = sc.nextInt();  switch(choice) {  case 1:  d1.setDate(year, month, date);  d1.displayDate(choice);  break;  case 2:  System.out.println("Enter Hours and Minutes: ");  hrs = sc.nextInt();  min = sc.nextInt();  d2.setDate(year, month, date, hrs, min);  d2.displayDate(choice);  break;  case 3:  System.out.println("Enter Hours,Minutes and Seconds: ");  hrs = sc.nextInt();  min = sc.nextInt();  sec = sc.nextInt();  d3.setDate(year, month, date, hrs, min, sec);  d3.displayDate(choice);  break;  default:  System.out.println("Invalid Choice!");  break;  }  System.out.println("Do you want to continue?(yes=1/0=no)");  flag = sc.nextInt();  if(flag==0) {  break;  }  }  }  } |
| **RESULT:**  **1) DD/MM/YY**    **2) DD/MM/YY, Hrs: Min**    **3) ) DD/MM/YY, Hrs: Min: Sec** | |
| **Program 2** | |
| **PROBLEM STATEMENT:** | **Given a class Line with slope, y-intercept, x1, y1, x2, y2 as  attributes, write 3 constructors for equations for the line given Slope-y-intercept, Slope Point and two Point forms**  **Slope-y-intercept:**  **y = mx + c**  **Slope point form:**  **y - y1 = m(x - x1)**  **Two Point form:**  **(y - y1) / (y1 - y2) = m(x - x1) / (x1 - x2)**  **Also, m = (y2 - y1) / (x2 - x1)**  **Each constructor should display the appropriate Line equation and appropriate value of y for given x.** |
| **PROGRAM:** | import java.util.\*;  class line{  double x=1,x1,y1,x2,y2;  double m,c;  line(double m, double c){  this.m=m;  this.c=c;  System.out.printf("y=%.0fx+%.0f",m,c);  System.out.println("");  System.out.printf("Value of y=%.0f at x=%.0f",m\*x+c,x);    }  line(double m, double x1, double y1){  this.m=m;  this.x1=x1;  this.y1=y1;  System.out.printf("Line eq: y-%.0f=%.0f(x - %.0f)\n",y1,m,x1);  System.out.printf("Value of y=%.0f at x=%.0f", (m\*(x-x1))-y1, x);  }  line(double x1, double y1, double x2, double y2){  this.x1=x1;  this.y1=y1;  this.x2=x2;  this.y2=y2;  System.out.printf("Line eq: y - %.0f/(%.0f-%.0f)= (x- %.0f)/( %.0f- %.0f)\n" ,y1,y1,y2,x1,x1,x2);  System.out.printf("Value of y = %.0f at x=%.0f", ((m\*(x-x1)\*(y1-y2))/(x1-x2))+y1,x);  }  public static void main( String[] args){  Scanner scan= new Scanner(System.in);  int option,flag;  double x1,y1,x2,y2,m,c;  while(true) {  System.out.println("\n 1) Slope-y-intercept form: \n 2)Slope point form: \n 3)Two-point form: ");  System.out.print("Enter the option you want: ");  option = scan.nextInt();  switch(option) {  case 1:  System.out.print("Enter slope: ");  m= scan.nextDouble();  System.out.print("Enter constant: ");  c= scan.nextDouble();  line l1= new line(m,c);  break;  case 2:  System.out.print("Enter slope: ");  m= scan.nextDouble();  System.out.print("Enter X1& Y1 Co-ordinate: ");  x1= scan.nextDouble();  y1= scan.nextDouble();  line l2= new line(m,x1,y1);  break;  case 3:  System.out.print("Enter X1& Y1 Co-ordinate: ");  x1= scan.nextDouble();  y1= scan.nextDouble();  System.out.print("Enter X2& Y2 Co-ordinate: ");  x2= scan.nextDouble();  y2= scan.nextDouble();  line l3= new line(x1,y1,x2,y2);  break;    default:  System.out.println("Invalid choice ");  break;  }  System.out.println("");  System.out.println("Do you want to continue?(yes=1/0=no)");  flag = scan.nextInt();  if(flag==0) {  break;  }  }  }  } |
| **RESULT:**  **1) Slope-y-intercept:**    **2) Slope point form:**    **3) Two Point form:** | |
| **Program 3** | |
| **PROBLEM STATEMENT:** | **Create a Test class with data double base, int power, int logBase, int argument.**  **Create a no-argument constructor which sets the default value of all variables to 2.**  **There are 2 overloaded functions:**  **1. double calculate (double base, int power)**  **This function returns the value when \*base\* is raised to \*power\***  **For example: calculate (3.0, 2) returns the value of 3.0 raised to 2 i.e., 9.0**  **2. double calculate (int logBase, int argument)**  **This function returns the value of the log of \*argument\* to the base \*logBase\*.**  **For example: calculate (3, 9) returns log of 9 to the base 3 i.e., 2.0**  **Create a main method in a separate class to call the above functions with the following inputs:**  **1. calculate (2, 4)**  **2. calculate (2.0, 4.0)**  **Create a display() method which displays the output based on the type of Test object created.** |
| **PROGRAM:** | import java.util.\*;  import java.lang.Math;  public class test{  double base;  int power,logBase, argument;    test(){  base = 2;  power = 2;  logBase = 2;  argument = 2;  }  public double calculate(double base, int power)  {  return Math.pow(base,power);  }  public double calculate(int logBase, int argument)  {  return (Math.log(argument)/Math.log(logBase));  }  public static void main(String[] args)  {  test ob1 = new test();  System.out.println("2 to power of 4 is "+ob1.calculate(2.0d,4));  System.out.println("log(4) to the base of 2 is "+ob1.calculate(2,4));  }  } |
| **RESULT:** | |
| **Program 4** | |
| **PROBLEM STATEMENT:** | **Write a menu-driven program to recruit an employee (depending on his performance in various rounds) in some software company using constructor overloading.**  **Selection Criteria for each post is given below:**  **i) Programmer (Minimum total of 80 marks):-**  **Rounds:-**  **(1) Course Work**  **(2) Aptitude Test**  **(3) Technical Test**  **(4) Interview**  **ii) Team Leader (Minimum total of 85 marks ):-**  **Rounds:-**  **(1) Technical Test**  **(2) Interview**  **iii) Project Manager (Minimum score 90 marks)**  **Rounds:-**  **(1) Interview**  **Create a class Posting and write 3 constructors to initialize the object and set the parameters**  **and display the employee post according to selection criteria.**  **Data members:**  **● int courseWork;**  **● int AptTest;**  **● int TechTest;**  **● int interview;**  **Methods:**  **● Posting (int courseWork, int AptTest, int TechTest,int interview)**  **● Posting (int TechTest,int interview)**  **● Posting (int interview)**  **Make use of ‘this’ keyword.** |
| **PROGRAM:** | import java.util.\*;  class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.print("Enter the number of candidates: ");  int n = sc.nextInt();  Posting b[] = new Posting[n];  int pcount = 0;  int lcount = 0;  int mcount = 0;  for (int k = 0; k < n; k++) {  System.out.println("Enter the number for the job you are applying: ");  System.out.println("1. Programmer\n2. Team Leader\n3. Project Manager");  int choice = sc.nextInt();  int c, a, t, i;  float score;  switch (choice) {  case 1:  System.out.println("Enter your scores in Course Work, Aptitude Test, Tech Test and Interview in order:");  c = sc.nextInt();  a = sc.nextInt();  t = sc.nextInt();  i = sc.nextInt();  b[k] = new Posting(c, a, t, i);  score = b[k].getScore();  if (score / 4 >= 80) {  System.out.println("You are recruited for the job");  pcount++;  } else {  System.out.println("Sorry to inform you that you aren't selected for the job.");  }  break;  case 2:  System.out.println("Enter your scores in Tech Test and Interview in order:");  t = sc.nextInt();  i = sc.nextInt();  b[k] = new Posting(t, i);  score = b[k].getScore();  if (score / 2 >= 85) {  System.out.println("You are recruited for the job");  lcount++;  } else {  System.out.println("Sorry to inform you that you aren't selected for the job.");  }  break;  case 3:  System.out.println("Enter your scores in Interview in order:");  i = sc.nextInt();  b[k] = new Posting(i);  score = b[k].getScore();  if (score >= 90) {  System.out.println("You are recruited for the job");  mcount++;  } else {  System.out.println("Sorry to inform you that you aren't selected for the job.");  }  break;  default:  System.out.println("Enter correct choice.");  }  }  System.out.println("Number of Programmers recruited = " + pcount);  System.out.println("Number of Team Leaders recruited = " + lcount);  System.out.println("Number of Project Managers recruited = " + mcount);  }  }  class Posting {  private int courseWork;  private int AptTest;  private int TechTest;  private int interview;  Posting() {  courseWork = 0;  AptTest = 0;  TechTest = 0;  interview = 0;  }  public Posting(int courseWork, int AptTest, int TechTest, int interview) {  this.courseWork = courseWork;  this.AptTest = AptTest;  this.TechTest = TechTest;  this.interview = interview;  }  public Posting(int TechTest, int interview) {  this.TechTest = TechTest;  this.interview = interview;  courseWork = 0;  AptTest = 0;  }  public Posting(int interview) {  this.interview = interview;  courseWork = 0;  AptTest = 0;  TechTest = 0;  }  float getScore() {  return courseWork + AptTest + TechTest + interview;  }  } |
| **RESULT:** | |
| **CONCLUSION:** | In this experiment, method overloading allows having more than one method with the same name in a class. |