| **Program 1** | |
| --- | --- |
| **PROBLEM STATEMENT :** | Enter an array (use Scanner) of distinct n integers (use command line argument) and a sum value. Print all triplets with a sum smaller than the given sum value.  Example:  Input : arr[] = {-2, 0, 1, 3}  sum = 2.  Output : (-2, 0, 1)  (-2, 0, 3)  Explanation : The two triplets have sum less than or equal to 2 |
| **PROGRAM:** | import java.util.Scanner;  class Triplets {  private int[] arr;  private int sum;  public Triplets(int[] arr, int sum) {  this.arr = arr;  this.sum = sum;  }  public void findTriplets() {  int n = arr.length;  for (int i = 0; i < n - 2; i++) {  for (int j = i + 1; j < n - 1; j++) {  for (int k = j + 1; k < n; k++) {  if (arr[i] + arr[j] + arr[k] < sum) {  System.out.println("(" + arr[i] + ", " + arr[j] + ", " + arr[k] + ")");  }  }  }  }  }  }  public class Main {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter the size of the array: ");  int n = scanner.nextInt();  int[] arr = new int[n];  System.out.println("Enter the array elements:");  for (int i = 0; i < n; i++) {  arr[i] = scanner.nextInt();  }  System.out.print("Enter the sum value: ");  int sum = scanner.nextInt();  Triplets triplets = new Triplets(arr, sum);  System.out.println("Triplets with sum smaller than " + sum + " are:");  triplets.findTriplets();  scanner.close();  }  } |
| **Program 2** | |
| **PROBLEM STATEMENT :** | Write a java program to keep a record of heights and weight of 10 different  people, and find out the number of people whose weight is less than 50kg and height is more than 170cms. |
| **PROGRAM:** | import java.util.Scanner;  class Person {  private double height;  private double weight;  public Person(double height, double weight) {  this.height = height;  this.weight = weight;  }  public double getHeight() {  return height;  }  public double getWeight() {  return weight;  }  }  public class Main {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  Person[] people = new Person[10];  // Input loop to get height and weight of 10 people  for (int i = 0; i < 10; i++) {  System.out.println("Enter details for person " + (i + 1) + ":");  System.out.print("Height (in cm): ");  double height = scanner.nextDouble();  System.out.print("Weight (in kg): ");  double weight = scanner.nextDouble();  people[i] = new Person(height, weight);  }  // Counting people with height > 170cm and weight < 50kg  int count = 0;  for (int i = 0; i < 10; i++) {  if (people[i].getHeight() > 170 && people[i].getWeight() < 50) {  count++;  }  }  System.out.println("Number of people with height more than 170cm and weight less than 50kg: " + count);  scanner.close();  }  } |
| **Program 3** | |
| **PROBLEM STATEMENT:** | Create a class 'Employee' with three data members which are name, age and Salary. The constructor of the class assigns default values name as "unknown" , age as '0' and address as "not available". It has SetInfo method assigns name, age and Salary respectively. Print the name, age and Salary of 10 employees sorted according to their age. Hint - Use array of objects |
| **PROGRAM:** | import java.util.Arrays;  import java.util.Comparator;  import java.util.Scanner;  class Employee {  private String name;  private int age;  private double salary;  public Employee() {  this.name = "unknown";  this.age = 0;  this.salary = 0.0;  }  public void setInfo(String name, int age, double salary) {  this.name = name;  this.age = age;  this.salary = salary;  }  public String getName() {  return name;  }  public int getAge() {  return age;  }  public double getSalary() {  return salary;  }  }  public class Main {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  Employee[] employees = new Employee[10];  // Input loop to get details for 10 employees  for (int i = 0; i < 10; i++) {  System.out.println("Enter details for employee " + (i + 1) + ":");  System.out.print("Name: ");  String name = scanner.nextLine();  System.out.print("Age: ");  int age = scanner.nextInt();  System.out.print("Salary: ");  double salary = scanner.nextDouble();  scanner.nextLine(); // Consume newline character  employees[i] = new Employee();  employees[i].setInfo(name, age, salary);  }  // Sorting employees based on age  Arrays.sort(employees, Comparator.comparingInt(Employee::getAge));  // Printing employee information  System.out.println("Employees sorted by age:");  for (Employee emp : employees) {  System.out.println("Name: " + emp.getName() + ", Age: " + emp.getAge() + ", Salary: " + emp.getSalary());  }  scanner.close();  }  } |
| **Program 4** | |
| **PROBLEM STATEMENT:** | A random number is generated between 0-1000 which is the cash inflow to the person playing the game. Two people play the game 5 times and the winner is the one whose total cash is maximum and the program displays the winner. Use the constructor to set the initial amount as zero.  Hint - Use array of objects, import java.lang.Math; and use Math.random() |
| **PROGRAM:** | import java.lang.Math;  class Player {  private int totalCash;  public Player() {  this.totalCash = 0;  }  public void playGame() {  int cashInflow = (int) (Math.random() \* 1000); // Random number between 0 and 1000  totalCash += cashInflow;  }  public int getTotalCash() {  return totalCash;  }  }  public class Main {  public static void main(String[] args) {  Player player1 = new Player();  Player player2 = new Player();  // Play the game 5 times  for (int i = 0; i < 5; i++) {  player1.playGame();  player2.playGame();  }  // Determine the winner  String winner;  if (player1.getTotalCash() > player2.getTotalCash()) {  winner = "Player 1";  } else if (player1.getTotalCash() < player2.getTotalCash()) {  winner = "Player 2";  } else {  winner = "It's a tie!";  }  // Print the winner  System.out.println("The winner is: " + winner);  }  } |

| **Program 5** | |
| --- | --- |
| **PROBLEM STATEMENT :** | A manufacturer is designing a rectangular box of various shapes. He has two assembly lines for the boxes. One is for the small boxes of size(L=2,B=6,H=3) in cm and big boxes of size L=12.50,B=30.00,H=78.25) in cm. When the material is supplied to him, he decides on the capacity of the store which is 50,000 cubic cm which boxes to manufacture. He has to optimize the production to fit in the store everyday. Use method overloading to set the values of L,B and H. |
| **PROGRAM:** | import java.util.Scanner;  class Box {  private double length;  private double breadth;  private double height;  public Box() {  this.length = 0;  this.breadth = 0;  this.height = 0;  }  // Method overloading to set dimensions for small box  public void setDimensions(double length, double breadth, double height) {  this.length = length;  this.breadth = breadth;  this.height = height;  }  // Method overloading to set dimensions for big box  public void setDimensions(double length, double breadth, double height, boolean isBigBox) {  if (isBigBox) {  this.length = length;  this.breadth = breadth;  this.height = height;  } else {  System.out.println("This method is only for setting dimensions of big boxes.");  }  }  // Method to calculate the volume of the box  public double calculateVolume() {  return length \* breadth \* height;  }  // Method to check if the box fits within the given capacity  public boolean fitsInStore(double capacity) {  return calculateVolume() <= capacity;  }  }  public class Main {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  // Input for small box dimensions  System.out.println("Enter dimensions for small box:");  System.out.print("Length: ");  double smallLength = scanner.nextDouble();  System.out.print("Breadth: ");  double smallBreadth = scanner.nextDouble();  System.out.print("Height: ");  double smallHeight = scanner.nextDouble();  // Input for big box dimensions  System.out.println("Enter dimensions for big box:");  System.out.print("Length: ");  double bigLength = scanner.nextDouble();  System.out.print("Breadth: ");  double bigBreadth = scanner.nextDouble();  System.out.print("Height: ");  double bigHeight = scanner.nextDouble();  double storeCapacity = 50000;  Box smallBox = new Box();  smallBox.setDimensions(smallLength, smallBreadth, smallHeight);  Box bigBox = new Box();  bigBox.setDimensions(bigLength, bigBreadth, bigHeight, true);  if (smallBox.fitsInStore(storeCapacity)) {  System.out.println("Small box fits in the store.");  } else {  System.out.println("Small box does not fit in the store.");  }  if (bigBox.fitsInStore(storeCapacity)) {  System.out.println("Big box fits in the store.");  } else {  System.out.println("Big box does not fit in the store.");  }  scanner.close();  }  } |
| **Program 6** | |
| **PROBLEM STATEMENT :** | Find out the area of geometric shapes with the given input attributes like length, breadth, height, radius etc. Design a program using method overloading to print the area of square, rectangle, circle and triangle. |
| **PROGRAM:** | import java.util.Scanner;  public class AreaCalculator {  // Method to calculate area of a square  public static double calculateArea(double side) {  return side \* side;  }  // Method to calculate area of a rectangle  public static double calculateArea(double length, double breadth) {  return length \* breadth;  }  // Method to calculate area of a circle  public static double calculateArea(double radius, boolean isCircle) {  if (isCircle) {  return Math.PI \* radius \* radius;  } else {  System.out.println("This method is only for calculating the area of a circle.");  return -1;  }  }  // Method to calculate area of a triangle  public static double calculateArea(double base, double height, boolean isTriangle) {  if (isTriangle) {  return 0.5 \* base \* height;  } else {  System.out.println("This method is only for calculating the area of a triangle.");  return -1;  }  }  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  // Square  System.out.print("Enter the side length of the square: ");  double side = scanner.nextDouble();  System.out.println("Area of the square: " + calculateArea(side));  // Rectangle  System.out.print("Enter the length of the rectangle: ");  double length = scanner.nextDouble();  System.out.print("Enter the breadth of the rectangle: ");  double breadth = scanner.nextDouble();  System.out.println("Area of the rectangle: " + calculateArea(length, breadth));  // Circle  System.out.print("Enter the radius of the circle: ");  double radius = scanner.nextDouble();  System.out.println("Area of the circle: " + calculateArea(radius, true));  // Triangle  System.out.print("Enter the base of the triangle: ");  double base = scanner.nextDouble();  System.out.print("Enter the height of the triangle: ");  double height = scanner.nextDouble();  System.out.println("Area of the triangle: " + calculateArea(base, height, true));  scanner.close();  }  } |
| **Program 7** | |
| **PROBLEM STATEMENT:** | Write a java program consisting of a class named Complex to add two complex numbers by using add() method by passing the objects as parameter(Complex number 1 and Complex number 2) and display the result using method display(). Initialize the real and imaginary values of the Complex number by writing parameterized Constructor. Use default constructor to initialize default values to real and imaginary values to zero. Use this keyword. |
| **PROGRAM:** | import java.util.\*;  class ComplexNo{  float real, imag;  ComplexNo(){  this.real = 0;  this.imag = 0;  }  ComplexNo(float real, float imag){  this.real = real;  this.imag = imag;  }  void addComplex(ComplexNo c1, ComplexNo c2){  System.out.println("Sum of these two complex numbers is : " + (c1.real + c2.real) + " + i " + (c1.imag + c2.imag));  }  }  public class complex {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.print("Enter real part of first number : ");  float real1 = sc.nextFloat();  System.out.print("Enter imaginary part of first number : ");  float imag1 = sc.nextFloat();  System.out.print("Enter real part of second number : ");  float real2 = sc.nextFloat();  System.out.print("Enter imaginary part of second number : ");  float imag2 = sc.nextFloat();  ComplexNo c1 = new ComplexNo(real1, imag1);  ComplexNo c2 = new ComplexNo(real2, imag2);  c1.addComplex(c1, c2);  }  } |
| **Program 8** | |
| **PROBLEM STATEMENT:** | A class with name LikesToDo have constructors LikesToDo(Girl object), LikesToDo(Boy object), where Boy and Girl are the two different classes. Now, body of the functions :  LikesToDo(Girl object, number of books)  {  //variables String Go to school, Go to Library;  if not reached maximum book limit then takes a book;  else return an old book and get a new book;  }  LikesToDo(Boy object)  {  //variables Sting Go to school, Go to the playground;  If not raining then play outdoor games;  else play indoor games;  }  Use appropriate methods and constructor call. |
| **PROGRAM:** | import java.util.Scanner;  class Girl {  private String name;  private int age;  public Girl(String name, int age) {  this.name = name;  this.age = age;  }  public String getName() {  return name;  }  public int getAge() {  return age;  }  }  class Boy {  private String name;  private int age;  public Boy(String name, int age) {  this.name = name;  this.age = age;  }  public String getName() {  return name;  }  public int getAge() {  return age;  }  }  public class LikesToDo {  private String activity;  // Constructor for Girl object  public LikesToDo(Girl girl, int numberOfBooks) {  this.activity = "Go to school, Go to Library";  if (numberOfBooks < 5) {  System.out.println(girl.getName() + " is taking a new book.");  } else {  System.out.println(girl.getName() + " is returning an old book and getting a new one.");  }  }  // Constructor for Boy object  public LikesToDo(Boy boy) {  this.activity = "Go to school, Go to the playground";  String weather = "sunny"; // Assuming it's sunny for demonstration  if (weather.equals("sunny")) {  System.out.println(boy.getName() + " is playing outdoor games.");  } else {  System.out.println(boy.getName() + " is playing indoor games.");  }  }  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  // Input for girl's details  System.out.println("Enter girl's name:");  String girlName = scanner.nextLine();  System.out.println("Enter girl's age:");  int girlAge = scanner.nextInt();  scanner.nextLine(); // Consume newline character  // Input for number of books the girl has  System.out.println("Enter number of books the girl has:");  int numberOfBooks = scanner.nextInt();  scanner.nextLine(); // Consume newline character  // Creating Girl object  Girl girl = new Girl(girlName, girlAge);  LikesToDo activity1 = new LikesToDo(girl, numberOfBooks);  // Input for boy's details  System.out.println("Enter boy's name:");  String boyName = scanner.nextLine();  System.out.println("Enter boy's age:");  int boyAge = scanner.nextInt();  scanner.nextLine(); // Consume newline character  // Creating Boy object  Boy boy = new Boy(boyName, boyAge);  LikesToDo activity2 = new LikesToDo(boy);  scanner.close();  }  } |

| **Program 9** | |
| --- | --- |
| **PROBLEM STATEMENT :** | To write a Java Program to create employee as a superclass and it include name and  designation. The subclass Salaried is calculating salary on the basis of designation and  experience.  --if experience is less than five years and designation=clerk, then salary=20000+bonus  where bonus=5000. If experience>= 5 and designation=clerk, then salary= 30000+bonus,  bonus= 10000.  --If experience is more than or equal to 5 years and designation=manager then  salary=35000+ bonus where bonus=15000. If experience>= 5 and designation=manager,  then salary= 40000+bonus, bonus= 30000.  --Print employee name, designation, bonus and total salary. Use super(). |
| **PROGRAM:** | import java.util.Scanner;  class Employee {  protected String name;  protected String designation;  public Employee(String name, String designation) {  this.name = name;  this.designation = designation;  }  public void display() {  System.out.println("Name: " + name);  System.out.println("Designation: " + designation);  }  }  class Salaried extends Employee {  private int experience;  private double bonus;  private double salary;  public Salaried(String name, String designation, int experience) {  super(name, designation);  this.experience = experience;  calculateSalary();  }  private void calculateSalary() {  if (designation.equalsIgnoreCase("clerk")) {  if (experience < 5) {  bonus = 5000;  salary = 20000 + bonus;  } else {  bonus = 10000;  salary = 30000 + bonus;  }  } else if (designation.equalsIgnoreCase("manager")) {  if (experience < 5) {  bonus = 15000;  salary = 35000 + bonus;  } else {  bonus = 30000;  salary = 40000 + bonus;  }  }  }  public void display() {  super.display();  System.out.println("Bonus: " + bonus);  System.out.println("Total Salary: " + salary);  }  }  public class Main {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter employee name: ");  String name = scanner.nextLine();  System.out.print("Enter designation (clerk/manager): ");  String designation = scanner.nextLine();  System.out.print("Enter years of experience: ");  int experience = scanner.nextInt();  Salaried employee = new Salaried(name, designation, experience);  employee.display();  scanner.close();  }  } |
| **Program 10** | |
| **PROBLEM STATEMENT :** | Write an inheritance hierarchy for classes Quadrilateral, Trapezoid, parallelogram, rectangle & Square. Use Quadrilateral as the super class of the hierarchy. Specify the instance variable and methods foreachclass.  The private instance variables of Quadrilateral should be the x-y coordinate  pairs of the four end points of quadrilateral. Write a java program that instantiates objects of your classes and outputs each object area( except quadrilateral) |
| **PROGRAM:** | import java.util.Scanner;  class Quadrilateral {  private double x1, y1, x2, y2, x3, y3, x4, y4;  public Quadrilateral(double x1, double y1, double x2, double y2,  double x3, double y3, double x4, double y4) {  this.x1 = x1;  this.y1 = y1;  this.x2 = x2;  this.y2 = y2;  this.x3 = x3;  this.y3 = y3;  this.x4 = x4;  this.y4 = y4;  }  public double getX1() {  return x1;  }  public double getY1() {  return y1;  }  public double getX2() {  return x2;  }  public double getY2() {  return y2;  }  public double getX3() {  return x3;  }  public double getY3() {  return y3;  }  public double getX4() {  return x4;  }  public double getY4() {  return y4;  }  public double distance(double x1, double y1, double x2, double y2) {  return Math.sqrt(Math.pow((x2 - x1), 2) + Math.pow((y2 - y1), 2));  }  public double calculateArea() {  // To be overridden by subclasses  return 0.0;  }  }  class Trapezoid extends Quadrilateral {  private double height;  public Trapezoid(double x1, double y1, double x2, double y2,  double x3, double y3, double x4, double y4, double height) {  super(x1, y1, x2, y2, x3, y3, x4, y4);  this.height = height;  }  @Override  public double calculateArea() {  double base1 = distance(getX1(), getY1(), getX2(), getY2());  double base2 = distance(getX3(), getY3(), getX4(), getY4());  return ((base1 + base2) / 2) \* height;  }  }  class Parallelogram extends Quadrilateral {  private double height;  public Parallelogram(double x1, double y1, double x2, double y2,  double x3, double y3, double x4, double y4, double height) {  super(x1, y1, x2, y2, x3, y3, x4, y4);  this.height = height;  }  @Override  public double calculateArea() {  double base = distance(getX1(), getY1(), getX2(), getY2());  return base \* height;  }  }  class Rectangle extends Parallelogram {  public Rectangle(double x1, double y1, double x2, double y2,  double x3, double y3, double x4, double y4) {  super(x1, y1, x2, y2, x3, y3, x4, y4, distance(x1, y1, x2, y2));  }  }  class Square extends Rectangle {  public Square(double x1, double y1, double x2, double y2,  double x3, double y3, double x4, double y4) {  super(x1, y1, x2, y2, x3, y3, x4, y4);  }  }  public class Main {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.println("Enter coordinates of the Quadrilateral (x1 y1 x2 y2 x3 y3 x4 y4):");  double x1 = scanner.nextDouble();  double y1 = scanner.nextDouble();  double x2 = scanner.nextDouble();  double y2 = scanner.nextDouble();  double x3 = scanner.nextDouble();  double y3 = scanner.nextDouble();  double x4 = scanner.nextDouble();  double y4 = scanner.nextDouble();  System.out.println("Enter height for Trapezoid:");  double heightTrapezoid = scanner.nextDouble();  System.out.println("Enter height for Parallelogram:");  double heightParallelogram = scanner.nextDouble();  Quadrilateral quad = new Quadrilateral(x1, y1, x2, y2, x3, y3, x4, y4);  Trapezoid trapezoid = new Trapezoid(x1, y1, x2, y2, x3, y3, x4, y4, heightTrapezoid);  Parallelogram parallelogram = new Parallelogram(x1, y1, x2, y2, x3, y3, x4, y4, heightParallelogram);  Rectangle rectangle = new Rectangle(x1, y1, x2, y2, x3, y3, x4, y4);  Square square = new Square(x1, y1, x2, y2, x3, y3, x4, y4);  System.out.println("Area of Trapezoid: " + trapezoid.calculateArea());  System.out.println("Area of Parallelogram: " + parallelogram.calculateArea());  System.out.println("Area of Rectangle: " + rectangle.calculateArea());  System.out.println("Area of Square: " + square.calculateArea());  scanner.close();  }  } |
| **Program 11** | |
| **PROBLEM STATEMENT:** | Write a java program Superclass: class Info, with members as pid, branch, year; Display(): should display whether the branch is IT, Computer, Mech. year and pid class FE, char sub, int marks of 3 subjects, average . Display() class SE, char sub, int marks of 3 subjects, average. Display() class TE, char sub, int marks of 3 subjects, average. Display() The user should input id, branch, marks, and subject. O/p:Displays display id, year, subject,  marks, average. The FE, SE, TE classes all extend class info. |
| **PROGRAM:** | import java.util.Scanner;  class Info {  protected int pid;  protected String branch;  protected int year;  public void Display() {  System.out.println("ID: " + pid);  System.out.println("Branch: " + branch);  System.out.println("Year: " + year);  }  }  class FE extends Info {  private char sub;  private int[] marks = new int[3];  private double average;  public void inputDetails() {  Scanner scanner = new Scanner(System.in);  System.out.println("Enter ID: ");  pid = scanner.nextInt();  System.out.println("Enter Branch: ");  branch = scanner.next();  System.out.println("Enter Year: ");  year = scanner.nextInt();  System.out.println("Enter Subject: ");  sub = scanner.next().charAt(0);  System.out.println("Enter Marks for 3 Subjects: ");  for (int i = 0; i < 3; i++) {  marks[i] = scanner.nextInt();  }  }  public void calculateAverage() {  int sum = 0;  for (int mark : marks) {  sum += mark;  }  average = sum / 3.0;  }  public void Display() {  super.Display();  System.out.println("Subject: " + sub);  System.out.println("Marks: ");  for (int i = 0; i < 3; i++) {  System.out.println("Subject " + (i + 1) + ": " + marks[i]);  }  System.out.println("Average: " + average);  }  }  // Similarly define classes SE and TE extending Info  public class Main {  public static void main(String[] args) {  FE fe = new FE();  fe.inputDetails();  fe.calculateAverage();  fe.Display();  }  } |
| **Program 12** | |
| **PROBLEM STATEMENT:** | A boy has his money deposited $1000, $1500 and $2000 in banks-Bank A, Bank B and Bank C respectively. We have to print the money deposited by him in a particular bank. Create a class 'Bank' with a method 'getBalance' which returns 0. Make its three subclasses named 'BankA', 'BankB' and 'BankC' with a method with the same name 'getBalance' which returns the amount deposited in that particular bank. Call the method 'getBalance' by the object of each of the three banks. |
| **PROGRAM:** | class Bank {  public int getBalance() {  return 0; // Default balance for the base class Bank  }  }  class BankA extends Bank {  @Override  public int getBalance() {  return 1000; // Balance in Bank A  }  }  class BankB extends Bank {  @Override  public int getBalance() {  return 1500; // Balance in Bank B  }  }  class BankC extends Bank {  @Override  public int getBalance() {  return 2000; // Balance in Bank C  }  }  public class Main {  public static void main(String[] args) {  BankA bankA = new BankA();  BankB bankB = new BankB();  BankC bankC = new BankC();    // Getting balance from each bank  System.out.println("Balance in Bank A: $" + bankA.getBalance());  System.out.println("Balance in Bank B: $" + bankB.getBalance());  System.out.println("Balance in Bank C: $" + bankC.getBalance());  }  } |

| **Program 13** | |
| --- | --- |
| **PROBLEM STATEMENT:** | Write a program to maintain marks of student.  i) Student is Abstract class and it has Roll no., Name, subjectmark attributes. Showstudentdata() is abstract method. Getstudentdata() is non abstract method.  ii) ISport is an Interface, having attribute sportgracemarks=5. Showsportmark() is a method.  iii) IExServiceMan is an Interface, having attribute ExServiceMangracemarks=10.  ShowExServiceManmark() is a method.  iv) Result is Class and it is inheriting Student, ISport, IExServiceMan.  Totalmarks=subjectmark + sportgracemarks +ExServiceMangracemarks. Showresult() is method of Result class. |
| **PROGRAM:** | import java.util.Scanner;  abstract class Student {  protected int rollNo;  protected String name;  protected int subjectMarks;  // Abstract method  public abstract void showStudentData();  // Non-abstract method  public void getStudentData() {  Scanner scanner = new Scanner(System.in);  System.out.println("Enter Roll No.: ");  rollNo = scanner.nextInt();  System.out.println("Enter Name: ");  name = scanner.next();  System.out.println("Enter Subject Marks: ");  subjectMarks = scanner.nextInt();  }  }  interface ISport {  int sportGraceMarks = 5;  void showSportMarks();  }  interface IExServiceMan {  int exServiceManGraceMarks = 10;  void showExServiceManMarks();  }  class Result extends Student implements ISport, IExServiceMan {  public void showStudentData() {  System.out.println("Roll No.: " + rollNo);  System.out.println("Name: " + name);  System.out.println("Subject Marks: " + subjectMarks);  }  public void showSportMarks() {  System.out.println("Sport Grace Marks: " + sportGraceMarks);  }  public void showExServiceManMarks() {  System.out.println("Ex-Service Man Grace Marks: " + exServiceManGraceMarks);  }  public void showResult() {  int totalMarks = subjectMarks + sportGraceMarks + exServiceManGraceMarks;  System.out.println("Total Marks: " + totalMarks);  }  }  public class Main {  public static void main(String[] args) {  Result result = new Result();  result.getStudentData();  result.showStudentData();  result.showSportMarks();  result.showExServiceManMarks();  result.showResult();  }  } |
| **Program 14** | |
| **PROBLEM STATEMENT:** | You are given an interface Arithmetic which contains a method signature int divisor\_sum(int n). You need to write a class called MyCalculator which implements the interface. divisorSum function just takes an integer as input and return the sum of all its divisors. For example divisors of 6 are 1, 2, 3 and 6, so should return 12. The value of n will be at most 1000. |
| **PROGRAM:** | interface Arithmetic {  int divisor\_sum(int n);  }  class MyCalculator implements Arithmetic {  public int divisor\_sum(int n) {  int sum = 0;  for (int i = 1; i <= n; i++) {  if (n % i == 0) {  sum += i;  }  }  return sum;  }  }  public class Main {  public static void main(String[] args) {  MyCalculator myCalculator = new MyCalculator();  int n = 6; // Example value  System.out.println("Sum of divisors of " + n + ": " + myCalculator.divisor\_sum(n));  }  } |
| **Program 15** | |
| **PROBLEM STATEMENT:** | Anand and Krishna are playing a game. This game starts with two piles of n1 and n2 chips. They play alternatively. In his/her turn a person has to remove one of the piles and split the other pile into two piles, these two new piles need not be of the same size. The person who cannot make a move in his turn loses. Write a program to find the winner. Here interface Piles has  SetPiles(int,int) to set the number of chips in each pile.  Sample Input  31  92  Sample Output  Bob  Alice  Explanation  Test case 1:  Initially, the piles are (3, 1).  Alice takes 1 and splits 3 into (1, 2).  Bob takes 1 and splits 2 into (1, 1).  Alice cannot make her move. Thus Bob wins.  Test case 2:  Initially, the piles are (9, 2).  Alice takes 9 and splits 2 into (1, 1).  Bob cannot make his move. Thus Alice wins. |
| **PROGRAM:** |  |
| **Program 16** | |
| **PROBLEM STATEMENT:** | We have to calculate the percentage of marks obtained in three subjects (each out of 100) by student A and in four subjects (each out of 100) by student B. Create an abstract class 'Marks' with an abstract method 'getPercentage'. It is inherited by two other classes 'A' and 'B' each having a method with the same name which returns the percentage of the students. The constructor of student A takes the marks in three subjects as its parameters and the marks in four subjects as its parameters for student B. Create an object for each of the two classes and print the percentage of marks for both the students |
| **PROGRAM:** | import java.util.Scanner;  abstract class Marks {  public abstract double getPercentage();  }  class A extends Marks {  private int subject1Marks;  private int subject2Marks;  private int subject3Marks;  public A(int subject1Marks, int subject2Marks, int subject3Marks) {  this.subject1Marks = subject1Marks;  this.subject2Marks = subject2Marks;  this.subject3Marks = subject3Marks;  }  public double getPercentage() {  double totalMarks = subject1Marks + subject2Marks + subject3Marks;  return (totalMarks / 300) \* 100; // Total marks for 3 subjects = 300  }  }  class B extends Marks {  private int subject1Marks;  private int subject2Marks;  private int subject3Marks;  private int subject4Marks;  public B(int subject1Marks, int subject2Marks, int subject3Marks, int subject4Marks) {  this.subject1Marks = subject1Marks;  this.subject2Marks = subject2Marks;  this.subject3Marks = subject3Marks;  this.subject4Marks = subject4Marks;  }  public double getPercentage() {  double totalMarks = subject1Marks + subject2Marks + subject3Marks + subject4Marks;  return (totalMarks / 400) \* 100; // Total marks for 4 subjects = 400  }  }  public class Main {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  // Input marks for student A  System.out.println("Enter marks for Student A (Subject 1, Subject 2, Subject 3): ");  int marksA1 = scanner.nextInt();  int marksA2 = scanner.nextInt();  int marksA3 = scanner.nextInt();  A studentA = new A(marksA1, marksA2, marksA3);  double percentageA = studentA.getPercentage();  System.out.println("Percentage of marks for Student A: " + percentageA + "%");  // Input marks for student B  System.out.println("Enter marks for Student B (Subject 1, Subject 2, Subject 3, Subject 4): ");  int marksB1 = scanner.nextInt();  int marksB2 = scanner.nextInt();  int marksB3 = scanner.nextInt();  int marksB4 = scanner.nextInt();  B studentB = new B(marksB1, marksB2, marksB3, marksB4);  double percentageB = studentB.getPercentage();  System.out.println("Percentage of marks for Student B: " + percentageB + "%");  }  } |

| **Program 17** | |
| --- | --- |
| **PROBLEM STATEMENT:** | We have to calculate the area of a rectangle, a square and a circle. Create an abstract class 'Shape' with three abstract methods namely 'RectangleArea' taking two parameters, 'SquareArea' and 'CircleArea' taking one parameter each. The parameters of 'RectangleArea' are its length and breadth, that of 'SquareArea' is its side and that of 'CircleArea' is its radius. Now create another class 'Area' containing all the three methods 'RectangleArea', 'SquareArea' and 'CircleArea' for printing the area of rectangle, square and circle respectively. Create an object of class 'Area' and call all the three methods. Your code check for 0, negative values and other error checks |
| **PROGRAM:** | import java.util.Scanner;  abstract class Shape {  public abstract double rectangleArea(double length, double breadth);  public abstract double squareArea(double side);  public abstract double circleArea(double radius);  }  class Area extends Shape {  public double rectangleArea(double length, double breadth) {  if (length <= 0 || breadth <= 0) {  System.out.println("Error: Invalid dimensions for rectangle");  return -1;  }  return length \* breadth;  }  public double squareArea(double side) {  if (side <= 0) {  System.out.println("Error: Invalid dimension for square");  return -1;  }  return side \* side;  }  public double circleArea(double radius) {  if (radius <= 0) {  System.out.println("Error: Invalid radius for circle");  return -1;  }  return Math.PI \* radius \* radius;  }  }  public class Main {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  Area areaCalculator = new Area();  // Calculate area of rectangle  System.out.println("Enter length and breadth of the rectangle: ");  double length = scanner.nextDouble();  double breadth = scanner.nextDouble();  double rectangleArea = areaCalculator.rectangleArea(length, breadth);  if (rectangleArea >= 0) {  System.out.println("Area of the rectangle: " + rectangleArea);  }  // Calculate area of square  System.out.println("Enter the side of the square: ");  double side = scanner.nextDouble();  double squareArea = areaCalculator.squareArea(side);  if (squareArea >= 0) {  System.out.println("Area of the square: " + squareArea);  }  // Calculate area of circle  System.out.println("Enter the radius of the circle: ");  double radius = scanner.nextDouble();  double circleArea = areaCalculator.circleArea(radius);  if (circleArea >= 0) {  System.out.println("Area of the circle: " + circleArea);  }  }  } |
| **Program 18** | |
| **PROBLEM STATEMENT:** | The GUI displays the account ID and the balance, and let the user deposit to or withdraw from the account. For each transaction, a message is displayed to indicate the status of the transaction: successful or failed. In case of failure, the failure reason is reported. The possible failure are negative-amount-exception (in both deposit and withdraw transaction) and insufficient-amount-exception ( in withdraw transaction). Account.java- private int id, private double balance, use constructor, int getId() ,void setBalance(double balance), double getBalance(), deposit an amount to this accountpublic void deposit(double amount), withdraw an amount from this account public void withdraw(double amount)  AccountGUI.java(Main class), NegativeAmountException.java and  InsufficientAmountException.java handle account transaction |
| **PROGRAM:** | // Account.java  public class Account {  private int id;  private double balance;  public Account(int id, double balance) {  this.id = id;  this.balance = balance;  }  public int getId() {  return id;  }  public void setBalance(double balance) {  this.balance = balance;  }  public double getBalance() {  return balance;  }  public void deposit(double amount) throws NegativeAmountException {  if (amount < 0) {  throw new NegativeAmountException("Deposit amount cannot be negative");  }  balance += amount;  }  public void withdraw(double amount) throws NegativeAmountException, InsufficientAmountException {  if (amount < 0) {  throw new NegativeAmountException("Withdrawal amount cannot be negative");  }  if (balance < amount) {  throw new InsufficientAmountException("Insufficient balance");  }  balance -= amount;  }  }  // AccountGUI.java  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.\*;  public class AccountGUI extends JFrame {  private Account account;  private JLabel idLabel, balanceLabel, amountLabel;  private JTextField amountField;  private JButton depositButton, withdrawButton;  public AccountGUI(Account account) {  this.account = account;  initialize();  }  private void initialize() {  setTitle("Account Transactions");  setSize(300, 200);  setDefaultCloseOperation(EXIT\_ON\_CLOSE);  idLabel = new JLabel("Account ID: " + account.getId());  balanceLabel = new JLabel("Balance: " + account.getBalance());  amountLabel = new JLabel("Enter amount: ");  amountField = new JTextField(10);  depositButton = new JButton("Deposit");  withdrawButton = new JButton("Withdraw");  depositButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  try {  double amount = Double.parseDouble(amountField.getText());  account.deposit(amount);  balanceLabel.setText("Balance: " + account.getBalance());  JOptionPane.showMessageDialog(AccountGUI.this, "Deposit successful");  } catch (NumberFormatException ex) {  JOptionPane.showMessageDialog(AccountGUI.this, "Invalid amount");  } catch (NegativeAmountException ex) {  JOptionPane.showMessageDialog(AccountGUI.this, ex.getMessage());  }  }  });  withdrawButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  try {  double amount = Double.parseDouble(amountField.getText());  account.withdraw(amount);  balanceLabel.setText("Balance: " + account.getBalance());  JOptionPane.showMessageDialog(AccountGUI.this, "Withdrawal successful");  } catch (NumberFormatException ex) {  JOptionPane.showMessageDialog(AccountGUI.this, "Invalid amount");  } catch (NegativeAmountException ex) {  JOptionPane.showMessageDialog(AccountGUI.this, ex.getMessage());  } catch (InsufficientAmountException ex) {  JOptionPane.showMessageDialog(AccountGUI.this, ex.getMessage());  }  }  });  setLayout(new GridLayout(4, 2));  add(idLabel);  add(balanceLabel);  add(amountLabel);  add(amountField);  add(depositButton);  add(withdrawButton);  setVisible(true);  }  public static void main(String[] args) {  Account account = new Account(12345, 1000);  new AccountGUI(account);  }  }  // NegativeAmountException.java  public class NegativeAmountException extends Exception {  public NegativeAmountException(String message) {  super(message);  }  }  // InsufficientAmountException.java  public class InsufficientAmountException extends Exception {  public InsufficientAmountException(String message) {  super(message);  }  } |
| **Program 19** | |
| **PROBLEM STATEMENT:** | Write a java program to throw an exception (checked) for an employee details. a) If an employee name is a number, a name exception must be thrown. b) If an employee age is greater than 50, an age exception must be thrown. c) Or else an object must be created for the entered employee details |
| **PROGRAM:** | import java.util.Scanner;  class NameException extends Exception {  public NameException(String message) {  super(message);  }  }  class AgeException extends Exception {  public AgeException(String message) {  super(message);  }  }  class Employee {  private String name;  private int age;  public void setName(String name) throws NameException {  // Check if name contains a number  if (name.matches(".\*\\d.\*")) {  throw new NameException("Employee name cannot contain numbers");  }  this.name = name;  }  public void setAge(int age) throws AgeException {  // Check if age is greater than 50  if (age > 50) {  throw new AgeException("Employee age cannot be greater than 50");  }  this.age = age;  }  public void displayDetails() {  System.out.println("Employee Details:");  System.out.println("Name: " + name);  System.out.println("Age: " + age);  }  }  public class Main {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  Employee employee = new Employee();  System.out.println("Enter employee name: ");  String name = scanner.nextLine();  System.out.println("Enter employee age: ");  int age = scanner.nextInt();  try {  employee.setName(name);  employee.setAge(age);  employee.displayDetails();  } catch (NameException e) {  System.out.println("Name Exception: " + e.getMessage());  } catch (AgeException e) {  System.out.println("Age Exception: " + e.getMessage());  }  }  } |
| **Program 20** | |
| **PROBLEM STATEMENT:** | Write a program to create a new thread by extending a thread class  1. get current thread name  2. Set to highest priority to the newly created thread  3. pause a thread for 1.5 seconds  4. Check weather the thread is running state or not  5. Check and make sure that your new thread completes before main thread  completes |
| **PROGRAM:** | class NewThread extends Thread {  public void run() {  // Task 1: Get current thread name  System.out.println("Current thread name: " + Thread.currentThread().getName());  // Task 2: Set highest priority to the newly created thread  Thread.currentThread().setPriority(Thread.MAX\_PRIORITY);  // Task 3: Pause the thread for 1.5 seconds  try {  Thread.sleep(1500);  } catch (InterruptedException e) {  System.out.println("Thread interrupted");  }  // Task 4: Check whether the thread is in the running state or not  if (Thread.currentThread().getState() == Thread.State.RUNNABLE) {  System.out.println("Thread is in running state");  } else {  System.out.println("Thread is not in running state");  }  }  }  public class Main {  public static void main(String[] args) {  NewThread newThread = new NewThread();  // Task 5: Start the new thread  newThread.start();  try {  // Wait for the new thread to complete before main thread completes  newThread.join();  } catch (InterruptedException e) {  System.out.println("Main thread interrupted");  }  System.out.println("Main thread completes");  }  } |

| **Program 21** | |
| --- | --- |
| **PROBLEM STATEMENT:** | In a bank there are two operations performed as withdraw and deposit. Either deposit is done or withdraw is done. Initial amount is set to 2500 and amount should not go below 500 while withdrawing money i.e it gives message “less money”. Write code for five transactions using synchronized multithreading. |
| **PROGRAM:** | class Bank {  private int balance = 2500;  // Synchronized method to withdraw money  public synchronized void withdraw(int amount) {  // Check if withdrawal amount is greater than balance  if (amount > balance) {  System.out.println("Insufficient funds for withdrawal");  return;  }  // Check if balance will go below 500 after withdrawal  if (balance - amount < 500) {  System.out.println("Cannot withdraw. Balance will go below 500");  return;  }  // Withdraw money  balance -= amount;  System.out.println("Withdrawal of " + amount + " successful. New balance: " + balance);  }  // Synchronized method to deposit money  public synchronized void deposit(int amount) {  // Deposit money  balance += amount;  System.out.println("Deposit of " + amount + " successful. New balance: " + balance);  }  }  class Transaction extends Thread {  private Bank bank;  private boolean isDeposit;  private int amount;  public Transaction(Bank bank, boolean isDeposit, int amount) {  this.bank = bank;  this.isDeposit = isDeposit;  this.amount = amount;  }  public void run() {  if (isDeposit) {  bank.deposit(amount);  } else {  bank.withdraw(amount);  }  }  }  public class Main {  public static void main(String[] args) {  Bank bank = new Bank();  // Creating 5 transactions (3 deposits, 2 withdrawals)  Transaction[] transactions = new Transaction[5];  transactions[0] = new Transaction(bank, true, 1000);  transactions[1] = new Transaction(bank, false, 800);  transactions[2] = new Transaction(bank, true, 1500);  transactions[3] = new Transaction(bank, false, 2000);  transactions[4] = new Transaction(bank, true, 600);  // Starting the transactions  for (Transaction transaction : transactions) {  transaction.start();  }  // Waiting for all transactions to complete  for (Transaction transaction : transactions) {  try {  transaction.join();  } catch (InterruptedException e) {  e.printStackTrace();  }  }  }  } |