**CYCLE 1: Basic programs using data types, operators and control statement in java**

|  |
| --- |
| **Program # 1 Date : 20/08/2023** |
| **Write a Java program to check whether a string is palindrome or not.** |

**Source Code :**

import java.io.\*;

public class PalindromeChecker {

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

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

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

String input = br.readLine();

if (isPalindrome(input))

System.out.println("The string is a palindrome.");

else

System.out.println("The string is not a palindrome.");

}

// Function to check if a string is palindrome or not

public static boolean isPalindrome(String str) {

int left = 0;

int right = str.length() - 1;

while (left < right) {

if (str.charAt(left) != str.charAt(right))

return false;

left++;

right--;

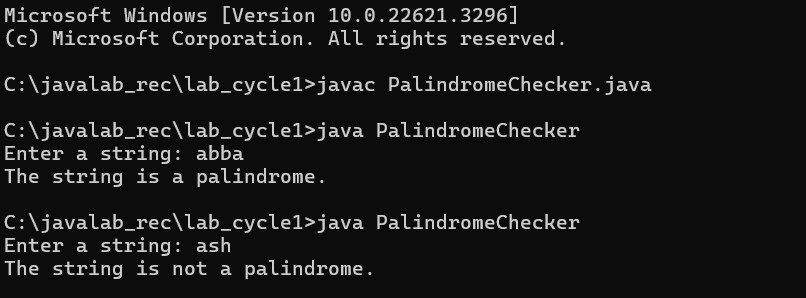
}

return true;

}

}

**Output :**



|  |
| --- |
| **Program # 2 Date : 20/08/2023** |
| **Write a Java program to multiply two matrices** |

**Source Code :**

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class MatrixMultiplication {

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

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the number of rows and columns for the first matrix:");

int rows1 = Integer.parseInt(br.readLine());

int cols1 = Integer.parseInt(br.readLine());

System.out.println("Enter the number of rows and columns for the second matrix:");

int rows2 = Integer.parseInt(br.readLine());

int cols2 = Integer.parseInt(br.readLine());

if (cols1 != rows2) {

System.out.println("Matrices cannot be multiplied.");

return;

}

int[][] firstMatrix = new int[rows1][cols1];

int[][] secondMatrix = new int[rows2][cols2];

System.out.println("Enter elements of the first matrix:");

for (int i = 0; i < rows1; i++) {

for (int j = 0; j < cols1; j++) {

firstMatrix[i][j] = Integer.parseInt(br.readLine());

}

}

System.out.println("Enter elements of the second matrix:");

for (int i = 0; i < rows2; i++) {

for (int j = 0; j < cols2; j++) {

secondMatrix[i][j] = Integer.parseInt(br.readLine());

}

}

int[][] resultMatrix = multiplyMatrices(firstMatrix, secondMatrix);

System.out.println("Resultant matrix after multiplication:");

for (int i = 0; i < rows1; i++) {

for (int j = 0; j < cols2; j++) {

System.out.print(resultMatrix[i][j] + " ");

}

System.out.println();

}

}

public static int[][] multiplyMatrices(int[][] firstMatrix, int[][] secondMatrix) {

int rows1 = firstMatrix.length;

int cols1 = firstMatrix[0].length;

int cols2 = secondMatrix[0].length;

int[][] resultMatrix = new int[rows1][cols2];

for (int i = 0; i < rows1; i++) {

for (int j = 0; j < cols2; j++) {

for (int k = 0; k < cols1; k++) {

resultMatrix[i][j] += firstMatrix[i][k] \* secondMatrix[k][j];

}

}

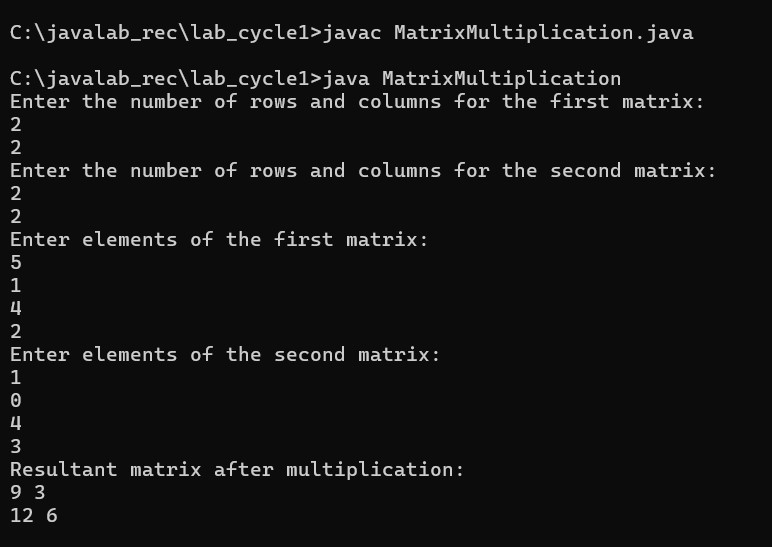
}

return resultMatrix;

}

}

**Output :**



|  |
| --- |
| **Program # 3 Date : 20/08/2023** |
| **Write a Java program to find the transpose of a matrix.** |

**Source Code :**

import java.util.Scanner;

public class MatrixTranspose {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input the matrix dimensions

System.out.print("Enter the number of rows: ");

int rows = scanner.nextInt();

System.out.print("Enter the number of columns: ");

int cols = scanner.nextInt();

// Input the matrix elements

int[][] matrix = new int[rows][cols];

System.out.println("Enter the elements of the matrix:");

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

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

matrix[i][j] = scanner.nextInt();

}

}

// Finding the transpose

int[][] transpose = new int[cols][rows];

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

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

transpose[i][j] = matrix[j][i];

}

}

// Displaying the original matrix

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

displayMatrix(matrix);

// Displaying the transpose

System.out.println("Transpose of the Matrix:");

displayMatrix(transpose);

scanner.close();

}

// Method to display the matrix

public static void displayMatrix(int[][] matrix) {

for (int[] row : matrix) {

for (int element : row) {

System.out.print(element + " ");

}

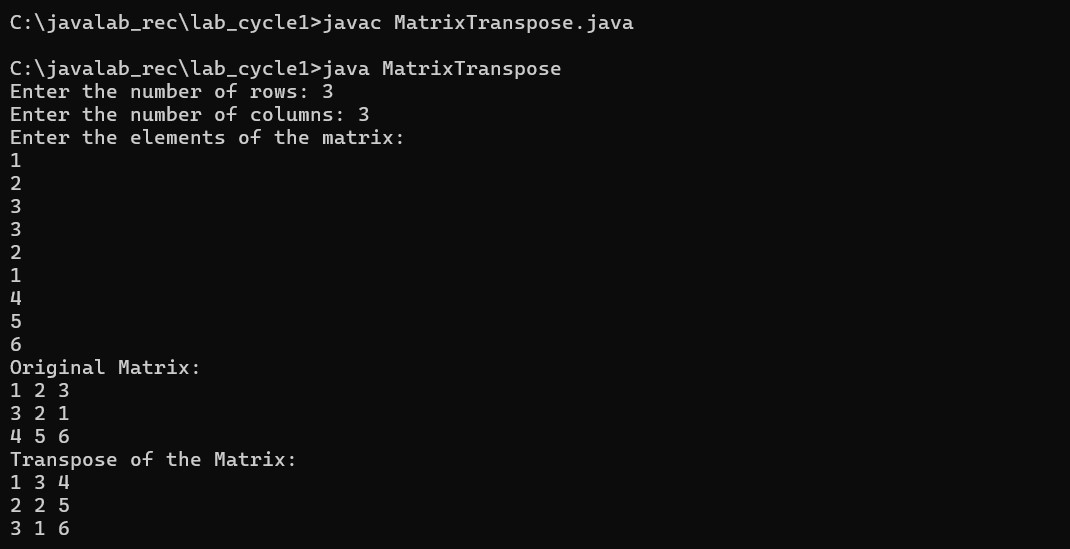
System.out.println();

}

}

}

**Output :**



|  |
| --- |
| **Program # 4 Date : 20/08/2023** |
| **Write a Java program to find the second smallest element in an array.** |

**Source Code :**

import java.util.Scanner;

public class SecondSmallestElement {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input the size of the array

System.out.print("Enter the number of elements in the array: ");

int size = scanner.nextInt();

// Check if the array size is less than 2

if (size < 2) {

System.out.println("Please enter at least two elements.");

return;

}

// Input the elements of the array

int[] array = new int[size];

System.out.println("Enter the elements of the array:");

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

array[i] = scanner.nextInt();

}

// Find the second smallest element

int secondSmallest = findSecondSmallest(array);

// Display the result

if (secondSmallest != Integer.MAX\_VALUE) {

System.out.println("The second smallest element in the array is: " + secondSmallest);

} else {

System.out.println("There is no second smallest element in the array.");

}

scanner.close();

}

public static int findSecondSmallest(int[] array) {

int smallest = Integer.MAX\_VALUE;

int secondSmallest = Integer.MAX\_VALUE;

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

if (array[i] < smallest) {

secondSmallest = smallest;

smallest = array[i];

} else if (array[i] < secondSmallest && array[i] != smallest) {

secondSmallest = array[i];

}

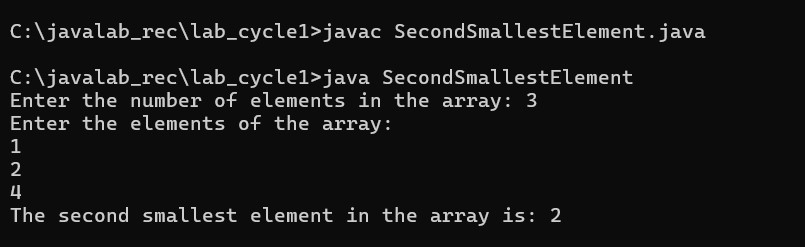
}

return secondSmallest;

}

}

**Output :**



|  |
| --- |
| **Program # 5 Date : 20/08/2023** |
| **Write a Java program to check whether a number is prime or not.** |

**Source Code :**

import java.util.Scanner;

public class PrimeChecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input the number to check

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

int number = scanner.nextInt();

boolean isPrime = checkPrime(number);

if (isPrime) {

System.out.println(number + " is a prime number.");

} else {

System.out.println(number + " is not a prime number.");

}

scanner.close();

}

public static boolean checkPrime(int number) {

if (number <= 1) {

return false; // 1 and numbers less than or equal to 1 are not prime

}

// Check for factors from 2 to the square root of the number

for (int i = 2; i \* i <= number; i++) {

if (number % i == 0) {

return false; // If a factor is found, the number is not prime

}

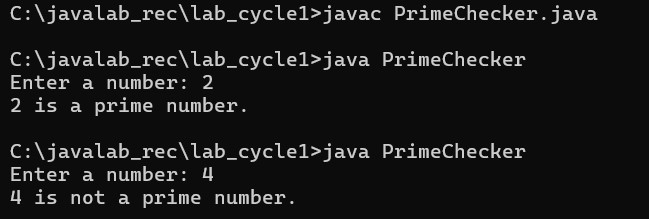
}

return true; // If no factor is found, the number is prime

}

}

**Output :**



|  |
| --- |
| **Program # 6 Date : 20/08/2023** |
| **Write a java program to demonstrate Bitwise logical operators, left shift and right shift operators.** |

**Source Code :**

import java.util.Scanner;

public class BitwiseOperatorsDemo {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the value for 'a': ");

int a = scanner.nextInt();

System.out.print("Enter the value for 'b': ");

int b = scanner.nextInt();

// Bitwise AND operator (&)

int bitwiseAnd = a & b;

System.out.println("Bitwise AND: " + bitwiseAnd);

// Bitwise OR operator (|)

int bitwiseOr = a | b;

System.out.println("Bitwise OR: " + bitwiseOr);

// Bitwise XOR operator (^)

int bitwiseXor = a ^ b;

System.out.println("Bitwise XOR: " + bitwiseXor);

// Bitwise NOT operator (~)

int bitwiseNotA = ~a;

System.out.println("Bitwise NOT of a: " + bitwiseNotA);

// Left shift operator (<<)

int leftShifted = a << 2;

System.out.println("Left shift of a: " + leftShifted);

// Right shift operator (>>)

int rightShifted = a >> 1;

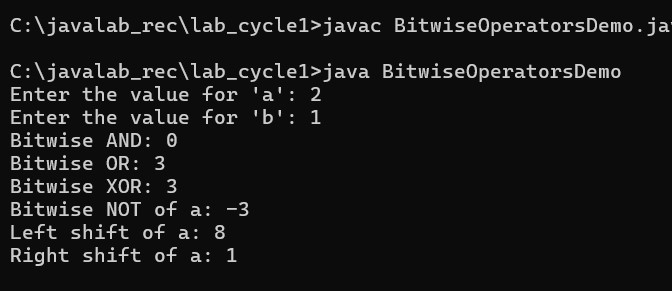
System.out.println("Right shift of a: " + rightShifted);

scanner.close();

}

}

**Output :**



|  |
| --- |
| **Program # 7 Date : 20/08/2023** |
| **Write a java program to find the roots of a quadratic equation.** |

**Source Code :**

import java.util.Scanner;

public class QuadraticEquationRoots {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the coefficients of the quadratic equation (a, b, c):");

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

double a = scanner.nextDouble();

System.out.print("b: ");

double b = scanner.nextDouble();

System.out.print("c: ");

double c = scanner.nextDouble();

double discriminant = b \* b - 4 \* a \* c;

if (discriminant > 0) {

double root1 = (-b + Math.sqrt(discriminant)) / (2 \* a);

double root2 = (-b - Math.sqrt(discriminant)) / (2 \* a);

System.out.println("Roots are real and different.");

System.out.println("Root 1: " + root1);

System.out.println("Root 2: " + root2);

} else if (discriminant == 0) {

double root = -b / (2 \* a);

System.out.println("Roots are real and same.");

System.out.println("Root: " + root);

} else {

double realPart = -b / (2 \* a);

double imaginaryPart = Math.sqrt(-discriminant) / (2 \* a);

System.out.println("Roots are complex and different.");

System.out.println("Root 1: " + realPart + " + " + imaginaryPart + "i");

System.out.println("Root 2: " + realPart + " - " + imaginaryPart + "i");

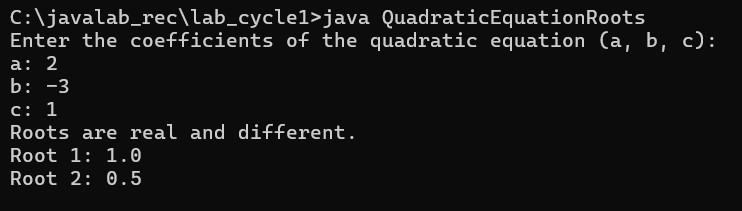
}

scanner.close();

}

}

**Output :**



**CYCLE 2: Object Oriented Concepts**

|  |
| --- |
| **Program # 1 Date : 20/08/2023** |
| **Write a Java program to calculate the area of different shapes namely circle, rectangle, trapezoid and triangle. (Use the concepts of JAVA like this keyword, constructor overloading and method overloading)** |

**Source Code :**

Import java.util.Scanner;

public class AreaCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Choose a shape (circle, rectangle, trapezoid, triangle):");

String shape = scanner.nextLine();

switch (shape.toLowerCase()) {

case "circle":

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

double radius = scanner.nextDouble();

Circle circle = new Circle(radius);

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

break;

case "rectangle":

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

double length = scanner.nextDouble();

System.out.print("Enter the width of the rectangle: ");

double width = scanner.nextDouble();

Rectangle rectangle = new Rectangle(length, width);

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

break;

case "trapezoid":

System.out.print("Enter the lengths of the bases of the trapezoid: ");

double base1 = scanner.nextDouble();

double base2 = scanner.nextDouble();

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

double height = scanner.nextDouble();

Trapezoid trapezoid = new Trapezoid(base1, base2, height);

System.out.println("Area of the trapezoid: " + trapezoid.calculateArea());

break;

case "triangle":

System.out.print("Enter the base length of the triangle: ");

double baseLength = scanner.nextDouble();

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

double triangleHeight = scanner.nextDouble();

Triangle triangle = new Triangle(baseLength, triangleHeight);

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

break;

default:

System.out.println("Invalid shape entered!");

}

scanner.close();

}

}

class Circle {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

public double calculateArea() {

return Math.PI \* radius \* radius;

}

}

class Rectangle {

private double length;

private double width;

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

public double calculateArea() {

return length \* width;

}

}

class Trapezoid {

private double base1;

private double base2;

private double height;

public Trapezoid(double base1, double base2, double height) {

this.base1 = base1;

this.base2 = base2;

this.height = height;

}

public double calculateArea() {

return (base1 + base2) \* height / 2;

}

}

class Triangle {

private double baseLength;

private double height;

public Triangle(double baseLength, double height) {

this.baseLength = baseLength;

this.height = height;

}

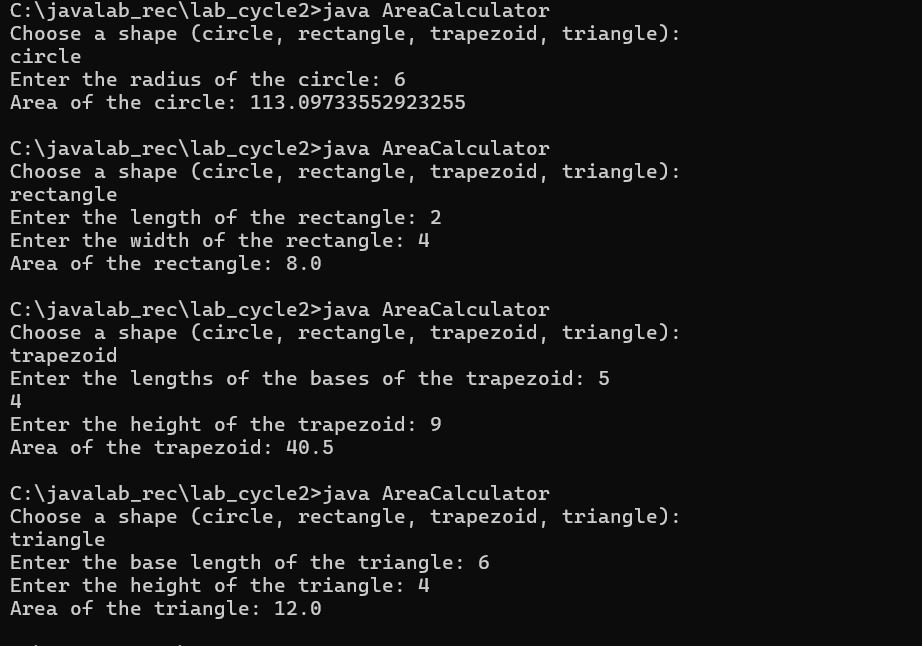
public double calculateArea() {

return baseLength \* height / 2;

}

}

**Output :**



|  |
| --- |
| **Program # 2 Date : 20/08/2023** |
| **Define a class called Rectangle with member variables length and width. Use appropriate member functions to calculate the perimeter and area of the rectangle. Define another member function int sameArea(Rectangle) that has one parameter of type Rectangle. sameArea returns 1 if the two Rectangles have the same area, and returns 0 if they dont. Use appropriate constructors to initialize the member variables(Use both default and parameterized constructor)** |

**Source Code :**

import java.util.Scanner;

public class Rectangle {

private double length;

private double width;

// Default constructor

public Rectangle() {

this.length = 0.0;

this.width = 0.0;

}

// Parameterized constructor

public Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

// Getter methods

public double getLength() {

return length;

}

public double getWidth() {

return width;

}

// Setter methods

public void setLength(double length) {

this.length = length;

}

public void setWidth(double width) {

this.width = width;

}

// Method to calculate the area of the rectangle

public double calculateArea() {

return length \* width;

}

// Method to calculate the perimeter of the rectangle

public double calculatePerimeter() {

return 2 \* (length + width);

}

// Method to check if two rectangles have the same area

public int sameArea(Rectangle otherRectangle) {

if (this.calculateArea() == otherRectangle.calculateArea()) {

return 1;

} else {

return 0;

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Get dimensions for rectangle1

System.out.println("Enter the length and width of rectangle1:");

double length1 = scanner.nextDouble();

double width1 = scanner.nextDouble();

Rectangle rectangle1 = new Rectangle(length1, width1);

// Get dimensions for rectangle2

System.out.println("Enter the length and width of rectangle2:");

double length2 = scanner.nextDouble();

double width2 = scanner.nextDouble();

Rectangle rectangle2 = new Rectangle(length2, width2);

// Display area and perimeter of rectangles

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

System.out.println("Area: " + rectangle1.calculateArea());

System.out.println("Perimeter: " + rectangle1.calculatePerimeter());

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

System.out.println("Area: " + rectangle2.calculateArea());

System.out.println("Perimeter: " + rectangle2.calculatePerimeter());

// Check if the rectangles have the same area

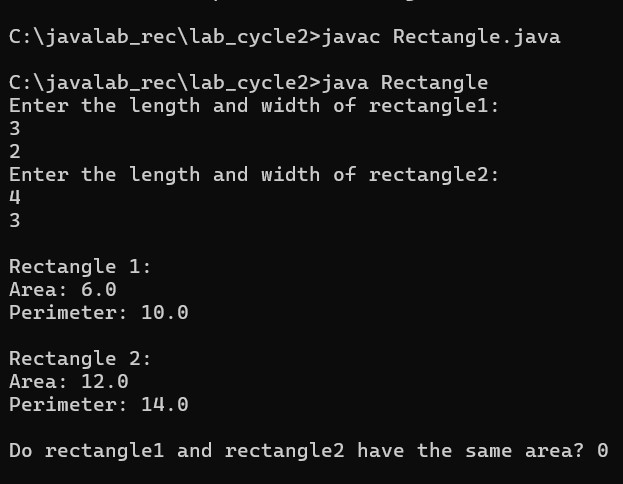
System.out.println("\nDo rectangle1 and rectangle2 have the same area? " + rectangle1.sameArea(rectangle2));

scanner.close();

}

}

**Output :**



|  |
| --- |
| **Program # 3 Date : 20/08/2023** |
| **Write a main function to create two rectangle objects and display its area and perimeter. Check whether the two Rectangles have the same area and print a message indicating the result. (Use the concept of this pointer too)** |

**Source Code :**

import java.util.Scanner;

public class RectanglePerimeter {

private double length;

private double width;

// Default constructor

public RectanglePerimeter() {

this.length = 0.0;

this.width = 0.0;

}

// Parameterized constructor

public RectanglePerimeter(double length, double width) {

this.length = length;

this.width = width;

}

// Getter methods

public double getLength() {

return length;

}

public double getWidth() {

return width;

}

// Setter methods

public void setLength(double length) {

this.length = length;

}

public void setWidth(double width) {

this.width = width;

}

// Method to calculate the area of the rectangle

public double calculateArea() {

return length \* width;

}

// Method to calculate the perimeter of the rectangle

public double calculatePerimeter() {

return 2 \* (length + width);

}

// Method to check if two rectangles have the same area

public boolean sameArea(RectanglePerimeter otherRectangle) {

return this.calculateArea() == otherRectangle.calculateArea();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Get dimensions for rectangle1

System.out.println("Enter the length and width of rectangle1:");

double length1 = scanner.nextDouble();

double width1 = scanner.nextDouble();

RectanglePerimeter rectangle1 = new RectanglePerimeter(length1, width1);

// Get dimensions for rectangle2

System.out.println("Enter the length and width of rectangle2:");

double length2 = scanner.nextDouble();

double width2 = scanner.nextDouble();

RectanglePerimeter rectangle2 = new RectanglePerimeter(length2, width2);

scanner.close();

// Display area and perimeter of rectangles

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

System.out.println("Area: " + rectangle1.calculateArea());

System.out.println("Perimeter: " + rectangle1.calculatePerimeter());

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

System.out.println("Area: " + rectangle2.calculateArea());

System.out.println("Perimeter: " + rectangle2.calculatePerimeter());

// Check if the rectangles have the same area

if (rectangle1.sameArea(rectangle2)) {

System.out.println("\nThe rectangles have the same area.");

} else {

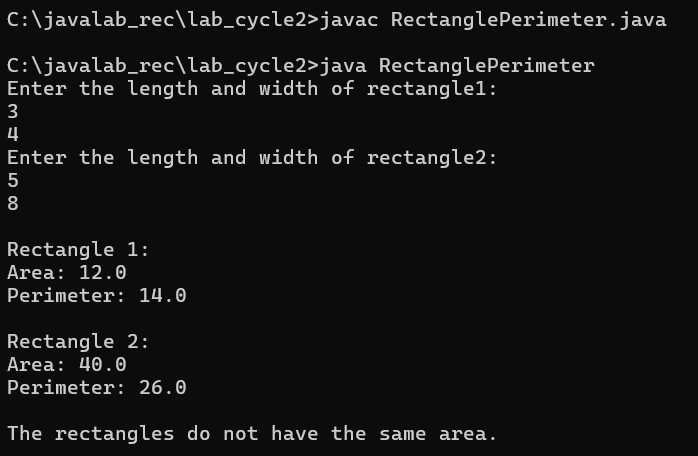
System.out.println("\nThe rectangles do not have the same area.");

}

}

}

**Output :**



|  |
| --- |
| **Program # 4 Date : 20/08/2023** |
| **Write the definition for a class called Complex that has floating point data members for storing real and imaginary parts. Define a function Complex sum(Complex) to add two complex numbers** |

**Source Code :**

import java.util.Scanner;

public class Complex {

private double real;

private double imaginary;

// Default constructor

public Complex() {

this.real = 0.0;

this.imaginary = 0.0;

}

// Parameterized constructor

public Complex(double real, double imaginary) {

this.real = real;

this.imaginary = imaginary;

}

// Getter methods

public double getReal() {

return real;

}

public double getImaginary() {

return imaginary;

}

// Setter methods

public void setReal(double real) {

this.real = real;

}

public void setImaginary(double imaginary) {

this.imaginary = imaginary;

}

// Method to add two complex numbers

public Complex sum(Complex other) {

double sumReal = this.real + other.real;

double sumImaginary = this.imaginary + other.imaginary;

return new Complex(sumReal, sumImaginary);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Get real and imaginary parts for complex number 1

System.out.println("Enter real and imaginary parts for complex number 1:");

double real1 = scanner.nextDouble();

double imaginary1 = scanner.nextDouble();

Complex complex1 = new Complex(real1, imaginary1);

// Get real and imaginary parts for complex number 2

System.out.println("Enter real and imaginary parts for complex number 2:");

double real2 = scanner.nextDouble();

double imaginary2 = scanner.nextDouble();

Complex complex2 = new Complex(real2, imaginary2);

scanner.close();

// Calculate the sum of the two complex numbers

Complex sum = complex1.sum(complex2);

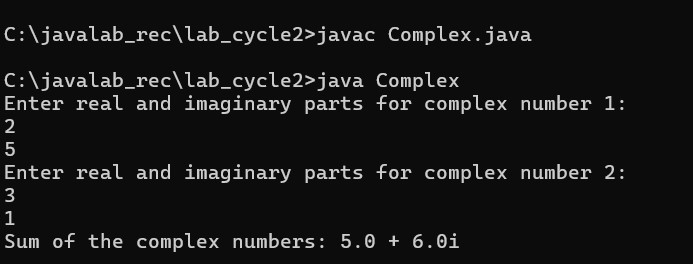
// Display the result

System.out.println("Sum of the complex numbers: " + sum.getReal() + " + " + sum.getImaginary() + "i");

}

}

**Output :**



|  |
| --- |
| **Program # 5 Date : 20/08/2023** |
| **& return complex number. Write main function to create three complex number objects. Set the value in two objects and call sum() to calculate sum and assign it in third object. Display all complex numbers. (Use the concept of this pointer too.)** |

**Source Code :**

import java.util.Scanner;

public class ComplexNumber {

private double real;

private double imaginary;

// Default constructor

public ComplexNumber() {

this.real = 0.0;

this.imaginary = 0.0;

}

// Parameterized constructor

public ComplexNumber(double real, double imaginary) {

this.real = real;

this.imaginary = imaginary;

}

// Getter methods

public double getReal() {

return real;

}

public double getImaginary() {

return imaginary;

}

// Setter methods

public void setReal(double real) {

this.real = real;

}

public void setImaginary(double imaginary) {

this.imaginary = imaginary;

}

// Method to add two complex numbers

public ComplexNumber sum(ComplexNumber other) {

double sumReal = this.real + other.real;

double sumImaginary = this.imaginary + other.imaginary;

return new ComplexNumber(sumReal, sumImaginary);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Get real and imaginary parts for complex number 1

System.out.println("Enter real and imaginary parts for complex number 1:");

double real1 = scanner.nextDouble();

double imaginary1 = scanner.nextDouble();

ComplexNumber c1 = new ComplexNumber(real1, imaginary1);

// Get real and imaginary parts for complex number 2

System.out.println("Enter real and imaginary parts for complex number 2:");

double real2 = scanner.nextDouble();

double imaginary2 = scanner.nextDouble();

ComplexNumber c2 = new ComplexNumber(real2, imaginary2);

scanner.close();

// Displaying c1 and c2

System.out.println("Complex Number 1: " + c1.getReal() + " + " + c1.getImaginary() + "i");

System.out.println("Complex Number 2: " + c2.getReal() + " + " + c2.getImaginary() + "i");

// Calculating sum of c1 and c2

ComplexNumber sum = c1.sum(c2);

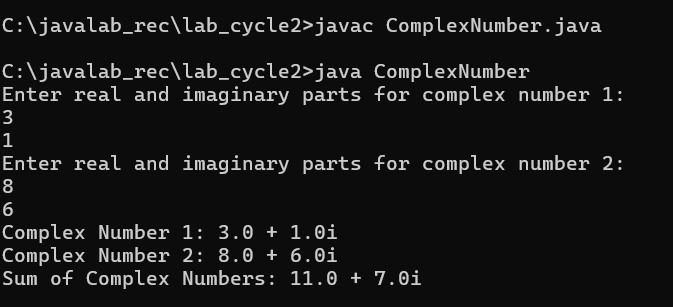
// Displaying the sum

System.out.println("Sum of Complex Numbers: " + sum.getReal() + " + " + sum.getImaginary() + "i");

}

}

**Output :**



|  |
| --- |
| **Program # 6 Date : 20/08/2023** |
| **Define a class called Time that has hours and minutes as integer. The class has the following member function: Time sum(Time) to sum two time object &amp; return time a. Use the concept of constructor overloading to initialize the time**  **6.1 Write the definitions for each of the above member functions.**  **6.2 Write main function to create three time objects. Set the value in two objects and call sum() to calculate sum and assign it in third object. Display all time objects. (Use theconcept of this pointer too)** |

**Source Code :**

import java.util.Scanner;

public class Time {

private int hours;

private int minutes;

// Default constructor

public Time() {

this.hours = 0;

this.minutes = 0;

}

// Parameterized constructor 1

public Time(int hours) {

this.hours = hours;

this.minutes = 0;

}

// Parameterized constructor 2

public Time(int hours, int minutes) {

this.hours = hours;

this.minutes = minutes;

}

// Getter methods

public int getHours() {

return hours;

}

public int getMinutes() {

return minutes;

}

// Setter methods

public void setHours(int hours) {

this.hours = hours;

}

public void setMinutes(int minutes) {

this.minutes = minutes;

}

// Method to sum two time objects

public Time sum(Time other) {

int totalHours = this.hours + other.hours;

int totalMinutes = this.minutes + other.minutes;

if (totalMinutes >= 60) {

totalHours += totalMinutes / 60;

totalMinutes %= 60;

}

return new Time(totalHours, totalMinutes);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Get hours and minutes for time object 1

System.out.println("Enter hours and minutes for time object 1:");

int hours1 = scanner.nextInt();

int minutes1 = scanner.nextInt();

Time t1 = new Time(hours1, minutes1);

// Get hours and minutes for time object 2

System.out.println("Enter hours and minutes for time object 2:");

int hours2 = scanner.nextInt();

int minutes2 = scanner.nextInt();

Time t2 = new Time(hours2, minutes2);

scanner.close();

// Displaying t1 and t2

System.out.println("Time Object 1: " + t1.getHours() + " hours " + t1.getMinutes() + " minutes");

System.out.println("Time Object 2: " + t2.getHours() + " hours " + t2.getMinutes() + " minutes");

// Calculating sum of t1 and t2

Time sum = t1.sum(t2);

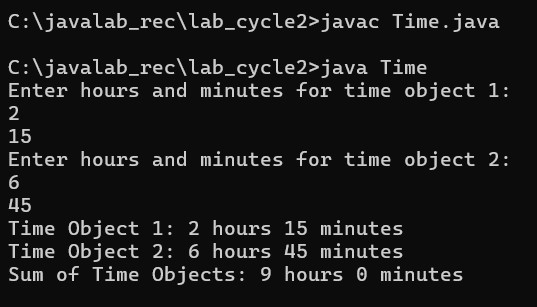
// Displaying the sum

System.out.println("Sum of Time Objects: " + sum.getHours() + " hours " + sum.getMinutes() + " minutes");

}

}

**Output :**



|  |
| --- |
| **Program # 7 Date : 20/08/2023** |
| **Create a class ‘Account’ with two overloaded constructors. The first constructor is used for initializing the name of account holder, the account number and the initial amount in the account. The second constructor is used for initializing the name of the account holder, the account number, the addresses, the type of account and the current balance. The Account class is having methods Deposit (), Withdraw (), and Get\_Balance(). Make the necessary assumption for data members and return types of the methods. Create objects of Account class and use them.** |

**Source Code :**

import java.util.Scanner;

public class Account {

private String accountHolderName;

private int accountNumber;

private String address;

private String accountType;

private double balance;

// First constructor with three parameters

public Account(String accountHolderName, int accountNumber, double initialAmount) {

this.accountHolderName = accountHolderName;

this.accountNumber = accountNumber;

this.balance = initialAmount;

}

// Second constructor with five parameters

public Account(String accountHolderName, int accountNumber, String address, String accountType, double balance) {

this.accountHolderName = accountHolderName;

this.accountNumber = accountNumber;

this.address = address;

this.accountType = accountType;

this.balance = balance;

}

// Method to deposit money into the account

public void deposit(double amount) {

balance += amount;

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

}

// Method to withdraw money from the account

public void withdraw(double amount) {

if (amount > balance) {

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

} else {

balance -= amount;

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

}

}

// Method to get the current balance

public double getBalance() {

return balance;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Getting user input for account 1

System.out.println("Enter account holder name, account number, and initial amount for account 1:");

String name1 = scanner.nextLine();

int number1 = scanner.nextInt();

double initialAmount1 = scanner.nextDouble();

scanner.nextLine(); // Consume newline

// Creating account 1 object

Account account1 = new Account(name1, number1, initialAmount1);

// Getting user input for account 2

System.out.println("Enter account holder name, account number, address, account type, and balance for account 2:");

String name2 = scanner.nextLine();

int number2 = scanner.nextInt();

scanner.nextLine(); // Consume newline

String address2 = scanner.nextLine();

String type2 = scanner.nextLine();

double balance2 = scanner.nextDouble();

// Creating account 2 object

Account account2 = new Account(name2, number2, address2, type2, balance2);

// Performing operations on accounts

account1.deposit(500.0);

account2.withdraw(300.0);

// Displaying the current balances of accounts

System.out.println("Account 1 balance: $" + account1.getBalance());

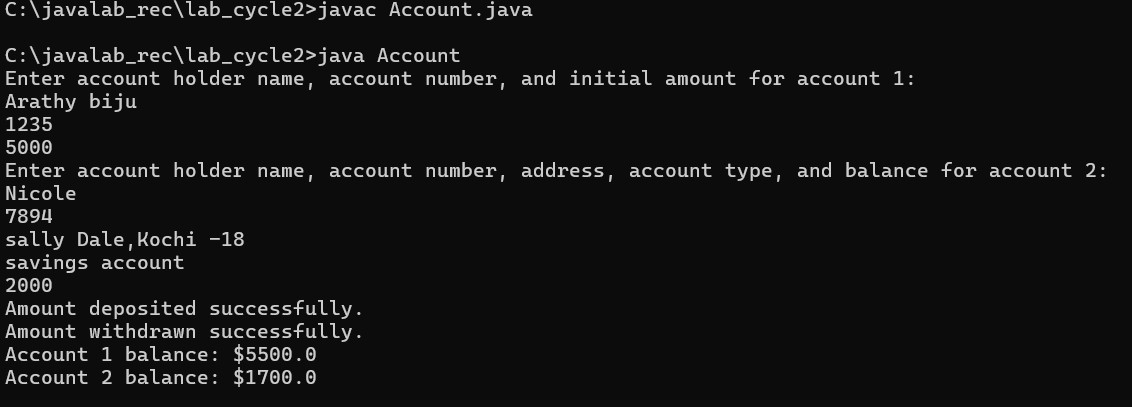
System.out.println("Account 2 balance: $" + account2.getBalance());

scanner.close();

}

}

**Output :**



**CYCLE 3: Inheritance, method overloading and overriding, Polymorphism (Due by: 28-01-**

**2024)**

|  |
| --- |
| **Program # 1 Date : 20/08/2023** |
| **Write a Java program which creates a class named Employee having the following**  **members: Name, Age, Phone number, Address, Salary. It also has a method named print-Salary( ) which prints the salary of the Employee. Two classes Officer and Manager inherits the Employee class. The Officer and Manager classes have data members specialization and department respectively. Now, assign name, age, phone number,address and salary to an officer and a manager by making an object of both of theseclasses and print the same.** |

**Source Code :**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input for Officer

System.out.println("Enter Officer details:");

System.out.print("Name: ");

String officerName = scanner.nextLine();

System.out.print("Age: ");

int officerAge = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Phone Number: ");

String officerPhoneNumber = scanner.nextLine();

System.out.print("Address: ");

String officerAddress = scanner.nextLine();

System.out.print("Salary: ");

double officerSalary = scanner.nextDouble();

scanner.nextLine(); // Consume newline

System.out.print("Specialization: ");

String officerSpecialization = scanner.nextLine();

// Creating Officer object

Officer officer = new Officer(officerName, officerAge, officerPhoneNumber, officerAddress, officerSalary, officerSpecialization);

// Input for Manager

System.out.println("\nEnter Manager details:");

System.out.print("Name: ");

String managerName = scanner.nextLine();

System.out.print("Age: ");

int managerAge = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Phone Number: ");

String managerPhoneNumber = scanner.nextLine();

System.out.print("Address: ");

String managerAddress = scanner.nextLine();

System.out.print("Salary: ");

double managerSalary = scanner.nextDouble();

scanner.nextLine(); // Consume newline

System.out.print("Department: ");

String managerDepartment = scanner.nextLine();

// Creating Manager object

Manager manager = new Manager(managerName, managerAge, managerPhoneNumber, managerAddress, managerSalary, managerDepartment);

// Printing details

System.out.println("\nOfficer details:");

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

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

System.out.println("Phone Number: " + officer.phoneNumber);

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

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

System.out.println("Specialization: " + officer.specialization);

officer.printSalary();

System.out.println("\nManager details:");

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

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

System.out.println("Phone Number: " + manager.phoneNumber);

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

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

System.out.println("Department: " + manager.department);

manager.printSalary();

scanner.close();

}

}

class Employee {

String name;

int age;

String phoneNumber;

String address;

double salary;

// Constructor

public Employee(String name, int age, String phoneNumber, String address, double salary) {

this.name = name;

this.age = age;

this.phoneNumber = phoneNumber;

this.address = address;

this.salary = salary;

}

// Method to print salary

public void printSalary() {

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

}

}

class Officer extends Employee {

String specialization;

// Constructor

public Officer(String name, int age, String phoneNumber, String address, double salary, String specialization) {

super(name, age, phoneNumber, address, salary);

this.specialization = specialization;

}

}

class Manager extends Employee {

String department;

// Constructor

public Manager(String name, int age, String phoneNumber, String address, double salary, String department) {

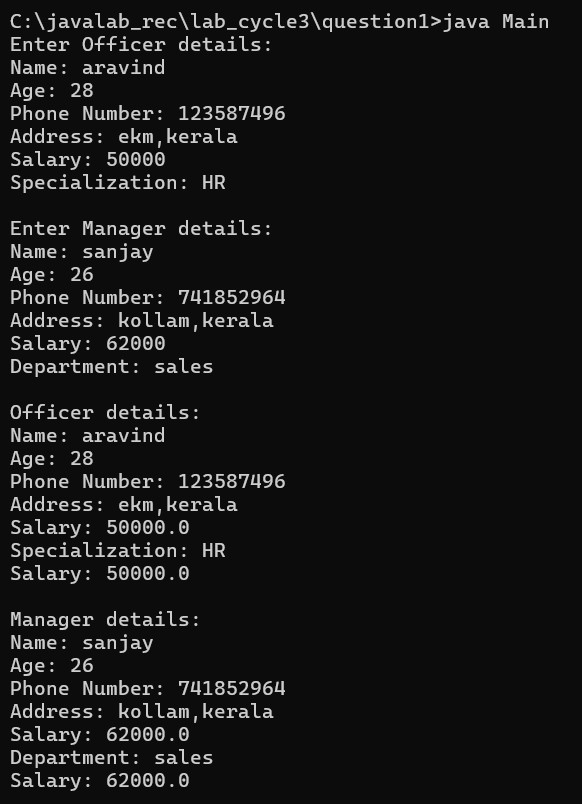
super(name, age, phoneNumber, address, salary);

this.department = department;

}

}

**Output :**



|  |
| --- |
| **Program # 2 Date : 20/08/2023** |
| **Write two Java classes Employee and Engineer. Engineer should inherit from**  **Employee class. Employee class to have two methods display() and calcSalary(). Write a program to display the engineer salary and to display from Employee class using a single object instantiation (i.e., only one object creation is allowed).**  **• display() only prints the name of the class and does not return any value. Ex. “ Name of class is Employee.”**  **• calcSalary() in Employee displays “Salary of employee is 10000” and calcSalary() in Engineer displays “Salary of employee is 20000.”** |

**Source Code :**

import java.util.Scanner;

class Employee {

public void display() {

System.out.println("Name of class is Employee.");

}

public void calcSalary() {

System.out.println("Salary of employee is 10000.");

}

}

class Engineer extends Employee {

@Override

public void calcSalary() {

System.out.println("Salary of employee is 20000.");

}

}

public class Main {

public static void main(String[] args) {

Employee emp = new Engineer(); // Polymorphism: Employee reference, Engineer object

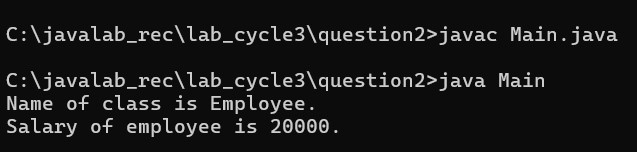
emp.display(); // Calls display method of Employee class

emp.calcSalary(); // Calls calcSalary method of Engineer class

}

}

**Output :**



|  |
| --- |
| **Program # 3 Date :20/08/2023** |
| **Write a Java program to implement the following level of inheritance.**  C:\Users\shari\AppData\Local\Packages\Microsoft.Windows.Photos_8wekyb3d8bbwe\TempState\ShareServiceTempFolder\Screenshot (56).jpeg |

**Source Code :**

import java.util.\*;

// Staff class

class Staff {

int code;

String name;

public Staff(int code, String name) {

this.code = code;

this.name = name;

}

}

// Faculty class

class Faculty extends Staff {

String department;

String subjectTaken;

String researchArea;

public Faculty(int code, String name, String department, String subjectTaken, String researchArea) {

super(code, name);

this.department = department;

this.subjectTaken = subjectTaken;

this.researchArea = researchArea;

}

}

// Typist class

class Typist extends Staff {

String office;

int speed;

public Typist(int code, String name, String office, int speed) {

super(code, name);

this.office = office;

this.speed = speed;

}

}

// Permanent class

class Permanent extends Typist {

double salary;

public Permanent(int code, String name, String office, int speed, double salary) {

super(code, name, office, speed);

this.salary = salary;

}

}

// Casual class

class Casual extends Typist {

double dailyWages;

public Casual(int code, String name, String office, int speed, double dailyWages) {

super(code, name, office, speed);

this.dailyWages = dailyWages;

}

}

// Officer class

class Officer extends Staff {

String rank;

String grade;

public Officer(int code, String name, String rank, String grade) {

super(code, name);

this.rank = rank;

this.grade = grade;

}

}

public class Main {

public static void main(String[] args) {

// Example usage

Faculty faculty = new Faculty(101, "John Doe", "Computer Science", "Data Structures", "Machine Learning");

System.out.println("Faculty: " + faculty.name);

Permanent permanentTypist = new Permanent(201, "Alice Smith", "Administration", 60, 50000.0);

System.out.println("Permanent Typist: " + permanentTypist.name);

Casual casualTypist = new Casual(301, "Bob Johnson", "Accounts", 50, 1000.0);

System.out.println("Casual Typist: " + casualTypist.name);

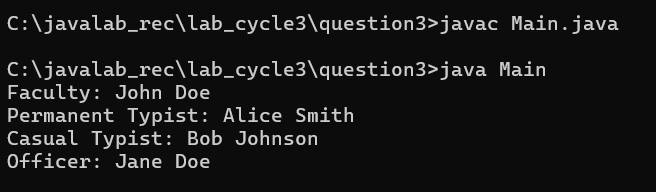
Officer officer = new Officer(401, "Jane Doe", "Commander", "A");

System.out.println("Officer: " + officer.name);

}

}

**Output :**



|  |
| --- |
| **Program # 4 Date : 20/08/2023** |
| **Write a java program to create an abstract class named Shape that contains an**  **empty method named numberOfSides(). Provide three classes named Rectangle, Triangle and Hexagon such that each one of the classes extends the class Shape. Each one of the classes contains only the method numberOfSides( ) that shows the number of sides in the given geometrical structures.** |

**Source Code :**

import java.util.Scanner;

// Abstract Shape class

abstract class Shape {

// Abstract method to get the number of sides

abstract void numberOfSides();

}

// Rectangle class

class Rectangle extends Shape {

// Implementation of the numberOfSides method for Rectangle

@Override

void numberOfSides() {

System.out.println("A rectangle has 4 sides.");

}

}

// Triangle class

class Triangle extends Shape {

// Implementation of the numberOfSides method for Triangle

@Override

void numberOfSides() {

System.out.println("A triangle has 3 sides.");

}

}

// Hexagon class

class Hexagon extends Shape {

// Implementation of the numberOfSides method for Hexagon

@Override

void numberOfSides() {

System.out.println("A hexagon has 6 sides.");

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Choose a shape (1. Rectangle, 2. Triangle, 3. Hexagon):");

int choice = scanner.nextInt();

Shape shape;

switch (choice) {

case 1:

shape = new Rectangle();

break;

case 2:

shape = new Triangle();

break;

case 3:

shape = new Hexagon();

break;

default:

System.out.println("Invalid choice.");

return;

}

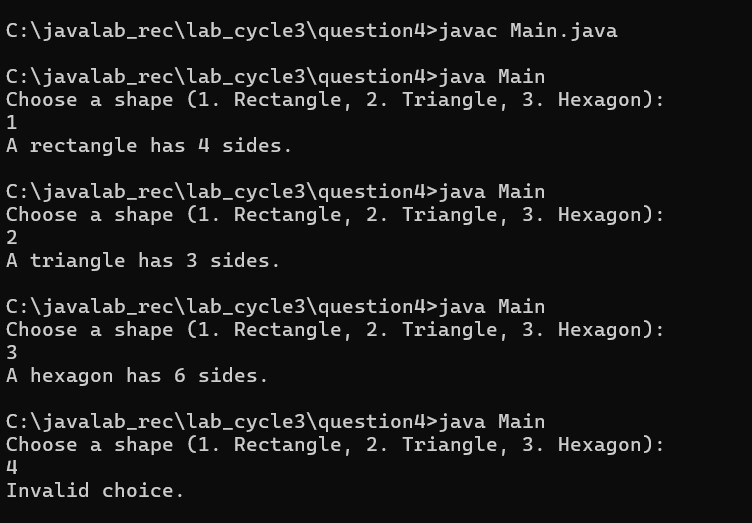
shape.numberOfSides();

scanner.close();

}

}

**Output :**



|  |
| --- |
| **Program # 5 Date : 20/08/2023** |
| **write a program to represent geometric shapes and some operations that can be**  **performed on them. The idea here is that shapes in higher dimensions inherit data from lower dimensional shapes. For example a cube is a three dimensional square. A sphere is a three dimensional circle and a glome is a four dimensional circle. A cylinder is another kind of three dimensional circle. The circle, sphere, cylinder, and glome all share the attribute radius. The square and cube share the attribute side length. There are various ways to use inheritance to relate these shapes but please follow the inheritance described in the table below.**  **All shapes inherit getName() from the superclass Shape.**  **Specification:**  **Your program will consist of the following classes: Shape, Circle, Square, Cube, Sphere, Cylinder,and Glome and two interfaces Area and Volume**  **Your classes may only have the class variable specified in the table below and the methods defined in the two interfaces Area and Volume. You will implement the methods specified in the Area and Volume interfaces and have them return the appropriate value for each shape. Class Shape will have a single public method called getName that returns a string.**  C:\Users\shari\AppData\Local\Packages\Microsoft.Windows.Photos_8wekyb3d8bbwe\TempState\ShareServiceTempFolder\Screenshot (59).jpeg |

**Source Code :**

import java.util.Scanner;

// Area interface

interface Area {

double calculateArea();

}

// Volume interface

interface Volume {

double calculateVolume();

}

// Shape class

class Shape {

String name;

public Shape() {

this.name = "Generic Shape";

}

public String getName() {

return name;

}

}

// Circle class

class Circle extends Shape implements Area {

double radius;

public Circle(double r, String n) {

this.radius = r;

this.name = n;

}

@Override

public double calculateArea() {

return Math.PI \* radius \* radius;

}

}

// Square class

class Square extends Shape implements Area {

double side;

public Square(double s, String n) {

this.side = s;

this.name = n;

}

@Override

public double calculateArea() {

return side \* side;

}

}

// Cylinder class

class Cylinder extends Circle implements Volume {

double height;

public Cylinder(double h, double r, String n) {

super(r, n);

this.height = h;

}

@Override

public double calculateVolume() {

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

}

}

// Sphere class

class Sphere extends Circle implements Volume {

public Sphere(double r, String n) {

super(r, n);

}

@Override

public double calculateVolume() {

return (4.0 / 3.0) \* Math.PI \* Math.pow(radius, 3);

}

}

// Cube class

class Cube extends Square implements Volume {

public Cube(double s, String n) {

super(s, n);

}

@Override

public double calculateVolume() {

return side \* side \* side;

}

}

// Glome class

class Glome extends Sphere implements Volume {

public Glome(double r, String n) {

super(r, n);

}

@Override

public double calculateVolume() {

return 0.5 \* Math.PI \* Math.PI \* radius \* radius \* radius \* radius;

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input for creating Circle

System.out.println("Enter radius for Circle:");

double radius = scanner.nextDouble();

Circle circle = new Circle(radius, "Circle");

System.out.println(circle.getName() + " - Area: " + circle.calculateArea());

// Input for creating Square

System.out.println("Enter side length for Square:");

double side = scanner.nextDouble();

Square square = new Square(side, "Square");

System.out.println(square.getName() + " - Area: " + square.calculateArea());

// Input for creating Cylinder

System.out.println("Enter radius for Cylinder:");

radius = scanner.nextDouble();

System.out.println("Enter height for Cylinder:");

double height = scanner.nextDouble();

Cylinder cylinder = new Cylinder(height, radius, "Cylinder");

System.out.println(cylinder.getName() + " - Area: " + cylinder.calculateArea() + ", Volume: " + cylinder.calculateVolume());

// Input for creating Sphere

System.out.println("Enter radius for Sphere:");

radius = scanner.nextDouble();

Sphere sphere = new Sphere(radius, "Sphere");

System.out.println(sphere.getName() + " - Area: " + sphere.calculateArea() + ", Volume: " + sphere.calculateVolume());

// Input for creating Cube

System.out.println("Enter side length for Cube:");

side = scanner.nextDouble();

Cube cube = new Cube(side, "Cube");

System.out.println(cube.getName() + " - Area: " + cube.calculateArea() + ", Volume: " + cube.calculateVolume());

// Input for creating Glome

System.out.println("Enter radius for Glome:");

radius = scanner.nextDouble();

Glome glome = new Glome(radius, "Glome");

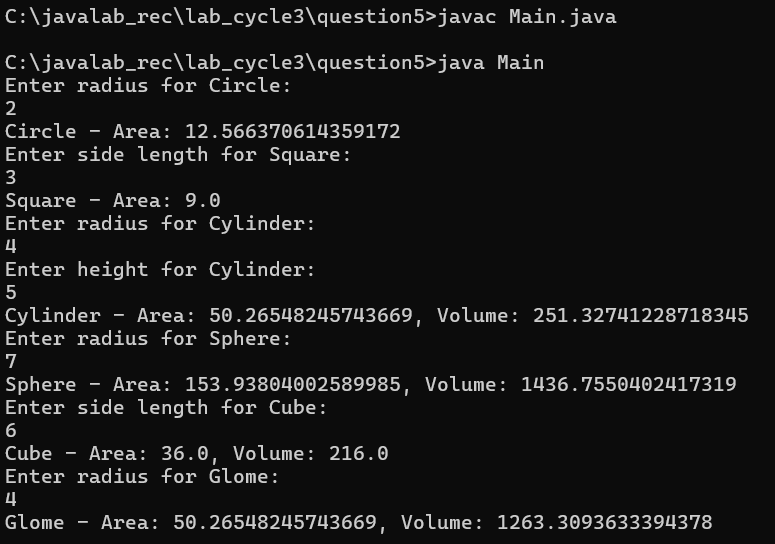
System.out.println(glome.getName() + " - Area: " + glome.calculateArea() + ", Volume: " + glome.calculateVolume());

scanner.close();

}

}

**Output :**



|  |
| --- |
| **Program 6 Date : 20/08/2023** |
| **Define an interface “Operations” which has method area(), volume(). Define a**  **constant PI having value 3.14. Create class a Cylinder( with member variable height) which implements this interface. Create one object and calculate area and volume. Add Required Constructors.** |

**Source Code :**

import java.util.Scanner;

// Define interface Operations

interface Operations {

double PI = 3.14; // constant

double area(); // method to calculate area

double volume(); // method to calculate volume

}

// Implementing class Cylinder

class Cylinder implements Operations {

double height; // member variable

// Constructor

public Cylinder(double height) {

this.height = height;

}

// Method to calculate area

public double area() {

return 2 \* PI \* height;

}

// Method to calculate volume

public double volume() {

return PI \* height \* height;

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

double height = scanner.nextDouble();

Cylinder cylinder = new Cylinder(height); // Creating an object of Cylinder

// Calculating area and volume

double area = cylinder.area();

double volume = cylinder.volume();

// Displaying the results

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

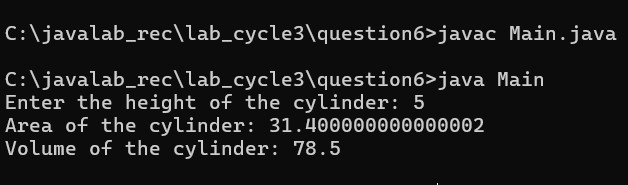
System.out.println("Volume of the cylinder: " + volume);

scanner.close();

}

}

**Output :**



|  |
| --- |
| **Program # 7 Date : 20/08/2023** |
| **Write a program that illustrates interface inheritance. Interface P is extended by P1**  **and P2. Interface P12 inherits from both P1 and P2.Each interface declares one constant and one method. class Q implements P12.Instantiate Q and invoke each of its methods. Each method displays one of the constants.** |

**Source Code :**

// Import statements

import java.util.\*;

// Define interface P

interface P {

void methodP(); // method declaration

}

// Interface P1 extends P

interface P1 extends P {

void methodP1(); // method declaration

}

// Interface P2 extends P

interface P2 extends P {

void methodP2(); // method declaration

}

// Interface P12 inherits from both P1 and P2

interface P12 extends P1, P2 {

// No additional constants or methods here

}

// Class Q implements P12

class Q implements P12 {

// Member variables to store constants

private int constantP;

private int constantP1;

private int constantP2;

// Constructor to receive constants

public Q(int constantP, int constantP1, int constantP2) {

this.constantP = constantP;

this.constantP1 = constantP1;

this.constantP2 = constantP2;

}

// Implementing method from P1 interface

public void methodP1() {

System.out.println("Constant from P1: " + constantP1);

}

// Implementing method from P2 interface

public void methodP2() {

System.out.println("Constant from P2: " + constantP2);

}

// Implementing method from P interface

public void methodP() {

System.out.println("Constant from P: " + constantP);

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Taking user input for constants

System.out.print("Enter constant value for P: ");

int constantP = scanner.nextInt();

System.out.print("Enter constant value for P1: ");

int constantP1 = scanner.nextInt();

System.out.print("Enter constant value for P2: ");

int constantP2 = scanner.nextInt();

// Instantiating Q with constants passed to its constructor

Q q = new Q(constantP, constantP1, constantP2);

// Invoking methods from Q

q.methodP();

q.methodP1();

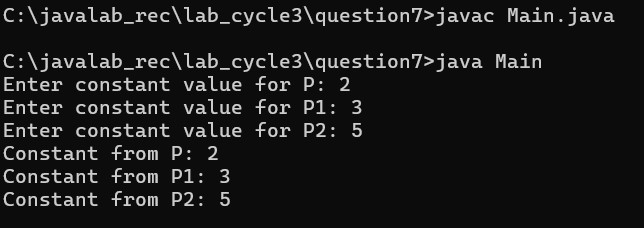
q.methodP2();

scanner.close();

}

}

**Output :**



**Cycle 4: Multithreading (Due by: 05-02-2024)**

|  |
| --- |
| **Program # 1 Date : 20/08/2023** |
| **Write a Java program to create two threads: One for displaying all odd number**  **between 1 and 100 and second thread for displaying all even numbers between 1 and 100. Create a multithreaded program by creating a subclass of Thread and then creating, initializing, and starting two Thread objects from your class. The threads will execute concurrently Main thread should wait until all the other thread terminates its execution(using join()).** |

**Source Code :**

import java.util.Scanner;

class OddThread extends Thread {

public void run() {

for (int i = 1; i <= 100; i += 2) {

System.out.println("Odd Thread: " + i);

try {

Thread.sleep(100); // Just for demonstration purposes

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

class EvenThread extends Thread {

public void run() {

for (int i = 2; i <= 100; i += 2) {

System.out.println("Even Thread: " + i);

try {

Thread.sleep(100); // Just for demonstration purposes

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Press Enter to start displaying odd and even numbers...");

scanner.nextLine();

OddThread oddThread = new OddThread();

EvenThread evenThread = new EvenThread();

oddThread.start();

evenThread.start();

try {

oddThread.join();

evenThread.join();

} catch (InterruptedException e) {

e.printStackTrace();

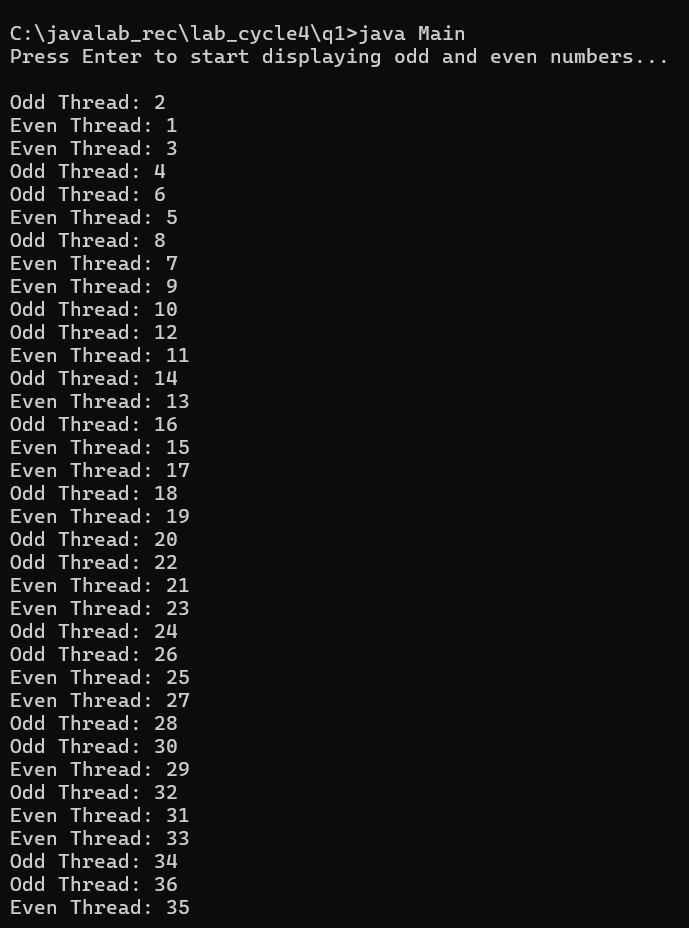
}

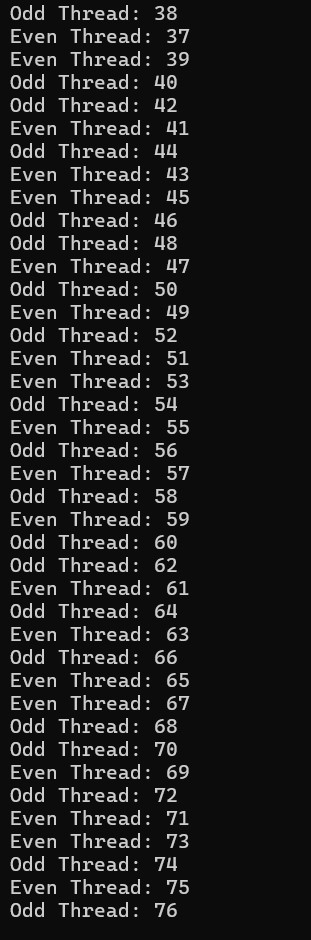
System.out.println("Main thread exiting...");

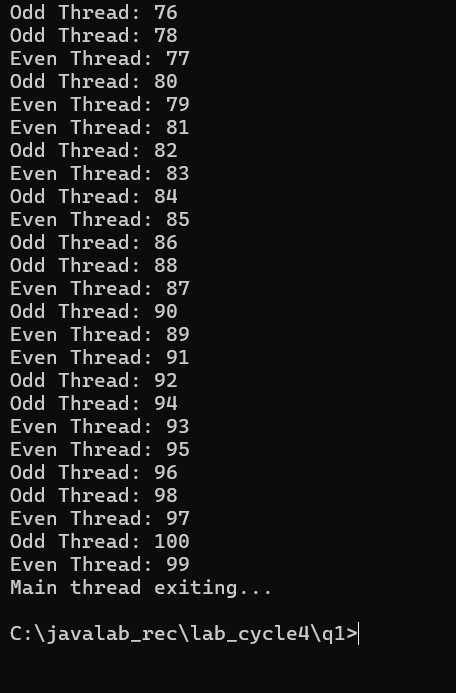
}

}

**Output :**







|  |
| --- |
| **Program # 2 Date : 20/08/2023** |
| **Write a Java program that set thread priorities and display the priority.** |

**Source Code :**

import java.lang.Thread;

import java.util.Scanner;

class PriorityThread extends Thread {

public PriorityThread(String name) {

super(name);

}

public void run() {

System.out.println("Thread Name: " + Thread.currentThread().getName() +

", Priority: " + Thread.currentThread().getPriority());

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Take user input for priorities

System.out.print("Enter priority for Thread 1 (1-10): ");

int priority1 = scanner.nextInt();

System.out.print("Enter priority for Thread 2 (1-10): ");

int priority2 = scanner.nextInt();

System.out.print("Enter priority for Thread 3 (1-10): ");

int priority3 = scanner.nextInt();

PriorityThread thread1 = new PriorityThread("Thread 1");

PriorityThread thread2 = new PriorityThread("Thread 2");

PriorityThread thread3 = new PriorityThread("Thread 3");

// Set priorities

thread1.setPriority(priority1);

thread2.setPriority(priority2);

thread3.setPriority(priority3);

// Start threads

thread1.start();

thread2.start();

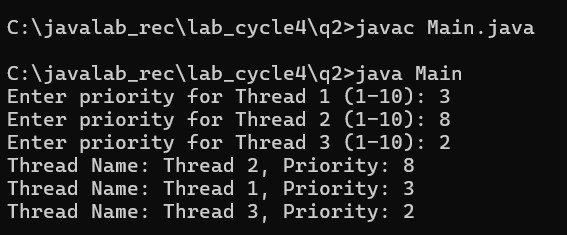
thread3.start();

scanner.close();

}

}

**Output :**



|  |
| --- |
| **Program # 3 Date : 20/08/2023** |
| **Write a java program that implements a multi-thread application that has three**  **threads. The first thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number** |

**Source Code :**

import java.util.Random;

// RandomNumberGenerator class generates random numbers every second

class RandomNumberGenerator extends Thread {

private boolean running; // Flag to control the running state of the thread

// Constructor initializes the running flag to true

public RandomNumberGenerator() {

this.running = true;

}

// Method to stop the thread by setting running flag to false

public void stopGenerating() {

this.running = false;

}

// Overridden run method that generates random numbers and starts appropriate threads

@Override

public void run() {

Random random = new Random();

while (running) { // Loop until running is true

try {

Thread.sleep(1000); // Sleep for 1 second

int number = random.nextInt(100); // Generate random number between 0 and 99

System.out.println("Generated number: " + number); // Print the generated number

if (number % 2 == 0) { // Check if the number is even

SquareThread squareThread = new SquareThread(number); // Create SquareThread

squareThread.start(); // Start SquareThread

}

else { // Number is odd

CubeThread cubeThread = new CubeThread(number); // Create CubeThread

cubeThread.start(); // Start CubeThread

}

} catch (InterruptedException e) {

e.printStackTrace(); // Print stack trace if interrupted

}

}

}

}

// SquareThread class computes and prints the square of a number

class SquareThread extends Thread {

private int number; // Number for which square is to be computed

// Constructor initializes the number

public SquareThread(int number) {

this.number = number;

}

// Overridden run method that computes and prints the square

@Override

public void run() {

System.out.println("Square of " + number + ": " + (number \* number)); // Print square of the number

}

}

// CubeThread class computes and prints the cube of a number

class CubeThread extends Thread {

private int number; // Number for which cube is to be computed

// Constructor initializes the number

public CubeThread(int number) {

this.number = number;

}

// Overridden run method that computes and prints the cube

@Override

public void run() {

System.out.println("Cube of " + number + ": " + (number \* number \* number)); // Print cube of the number

}

}

// Main class to test the program

public class Main {

public static void main(String[] args) {

RandomNumberGenerator randomNumberGenerator = new RandomNumberGenerator(); // Create RandomNumberGenerator

randomNumberGenerator.start(); // Start RandomNumberGenerator

try {

Thread.sleep(10000); // Sleep for 10 seconds to allow the program to run

} catch (InterruptedException e) {

e.printStackTrace(); // Print stack trace if interrupted

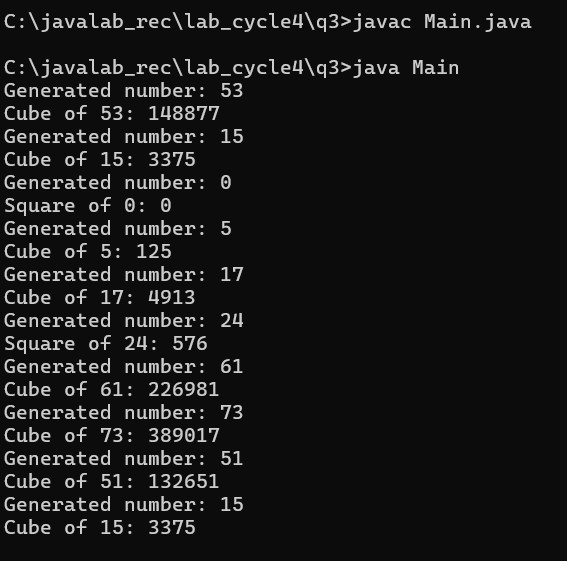
}

randomNumberGenerator.stopGenerating(); // Stop generating random numbers

}

}

**Output :**



|  |
| --- |
| **Program # 4 Date : 20/08/2023** |
| **Write a program to illustrate creation of threads using runnable interface. (start**  **method start each of the newly created thread. Inside the run method there is sleep() for suspend the thread for 500 milliseconds). Main thread should wait until all the other thread terminates its execution (using join()).** |

**Source Code :**

// Importing Scanner class to take user input

import java.util.Scanner;

// Runnable implementation

class MyRunnable implements Runnable {

// Run method that will be executed when the thread starts

public void run() {

try {

// Print the name of the current thread running

System.out.println(Thread.currentThread().getName() + " is running.");

// Pause the thread for 500 milliseconds

Thread.sleep(500);

} catch (InterruptedException e) {

// Print stack trace if the thread is interrupted

e.printStackTrace();

}

}

}

// Main class

public class Main {

public static void main(String[] args) {

// Creating a Scanner object to take user input

Scanner scanner = new Scanner(System.in);

// Prompting the user to enter the number of threads

System.out.print("Enter the number of threads: ");

// Reading the number of threads entered by the user

int numThreads = scanner.nextInt();

// Closing the scanner object

scanner.close();

// Creating an array to hold the threads

Thread[] threads = new Thread[numThreads];

// Creating and starting each thread

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

// Creating a new thread with MyRunnable instance as target

threads[i] = new Thread(new MyRunnable());

// Starting the thread

threads[i].start();

}

// Waiting for each thread to finish

for (Thread thread : threads) {

try {

// Waiting for the thread to finish its execution

thread.join();

} catch (InterruptedException e) {

// Print stack trace if the thread is interrupted while waiting

e.printStackTrace();

}

}

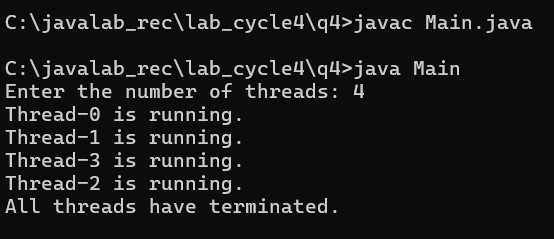
// Printing message indicating all threads have terminated

System.out.println("All threads have terminated.");

}

}

**Output :**



|  |
| --- |
| **Program # 5 Date : 20/08/2023** |
| **Write a java program showing a typical invocation of banking operations via**  **multiple threads. Create three threads and 2 methods deposit and withdraw methods to add the amount to the account and withdraw an amount from the account respectively. As the threads concurrently run the method, avoid the unpredictable behavior. (Use synchronization).** |

**Source Code :**

import java.util.Scanner; // Importing Scanner class to take user input

class BankAccount {

private int balance;

// Constructor to initialize balance

public BankAccount(int initialBalance) {

balance = initialBalance;

}

// Synchronized method to deposit amount into the account

public synchronized void deposit(int amount) {

System.out.println(Thread.currentThread().getName() + " is depositing $" + amount);

balance += amount;

System.out.println("New balance after deposit by " + Thread.currentThread().getName() + ": $" + balance);

}

// Synchronized method to withdraw amount from the account

public synchronized void withdraw(int amount) {

System.out.println(Thread.currentThread().getName() + " is withdrawing $" + amount);

if (balance >= amount) {

balance -= amount;

System.out.println("New balance after withdrawal by " + Thread.currentThread().getName() + ": $" + balance);

} else {

System.out.println("Insufficient balance for withdrawal by " + Thread.currentThread().getName());

}

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); // Creating a Scanner object to take user input

System.out.print("Enter initial balance: ");

int initialBalance = scanner.nextInt(); // Reading initial balance entered by the user

// Creating a bank account with initial balance entered by the user

BankAccount account = new BankAccount(initialBalance);

// Taking user input for deposit and withdrawal amounts

System.out.print("Enter deposit amount: ");

int depositAmount = scanner.nextInt();

System.out.print("Enter withdrawal amount: ");

int withdrawalAmount = scanner.nextInt();

scanner.close();

// Creating three threads performing deposit and withdrawal operations

Thread thread1 = new Thread(() -> {

account.deposit(depositAmount);

}, "Thread-1");

Thread thread2 = new Thread(() -> {

account.withdraw(withdrawalAmount);

}, "Thread-2");

Thread thread3 = new Thread(() -> {

account.deposit(depositAmount);

}, "Thread-3");

// Starting all threads

thread1.start();

thread2.start();

thread3.start();

}

}

**Output :**

