**Program: 1**

**Develop a JAVA program to add TWO matrices of suitable order N (The value of N should be read from command line arguments).**

import java.util.Scanner;

public class MatrixAddition

{

public static void main(String[] args)

{

if (args.length != 1)

{

System.out.println("Please provide the value of N as a command line argument.");

return;

}

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

int[][] matrix1 = new int[N][N];

int[][] matrix2 = new int[N][N];

Scanner scanner = new Scanner(System.in);

// Populate matrix1 and matrix2 with random values for demonstration

// You can modify this part to input values from the user or any other source

// For simplicity, we use random values between 1 and 10

/\* //Reading matrix using random

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

{

for(int j = 0; j < N; j++)

{

matrix1[i][j] = (int) (Math.random() \* 10) + 1;

matrix2[i][j] = (int) (Math.random() \* 10) + 1;

}

}\*/

// Reading the Matrix-1

System.out.println("Enter the Elements of Matrix 1:");

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

{

for(int j = 0; j < N; j++)

{

//matrix1[i][j] = (int) (Math.random() \* 10) + 1;

//matrix2[i][j] = (int) (Math.random() \* 10) + 1;

matrix1[i][j]=scanner.nextInt();

}

}

// Reading the Matrix-2

System.out.println("Enter the Elements of Matrix 2:");

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

{

for(int j = 0; j < N; j++)

{

//matrix1[i][j] = (int) (Math.random() \* 10) + 1;

//matrix2[i][j] = (int) (Math.random() \* 10) + 1;

matrix2[i][j]=scanner.nextInt();

}

}

// Perform matrix addition

int[][] result = new int[N][N];

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

{

for(int j = 0; j < N; j++)

{

result[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

// Display the matrices and their sum

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

printMatrix(matrix1);

System.out.println("Matrix 2:");

printMatrix(matrix2);

System.out.println("Sum of the matrices:");

printMatrix(result);

}

// Helper method to print a matrix

private static void printMatrix(int[][] matrix)

{

for (int[] row : matrix)

{

for (int num : row)

{

System.out.print(num + "\t");

}

System.out.println();

}

System.out.println();

}

}

**OUTPUT**

Enter the Elements of Matrix 1:

1 2 3

3 2 1

4 5 6

Enter the Elements of Matrix 2:

6 5 4

4 5 6

1 2 3

Matrix 1:

1 2 3

3 2 1

4 5 6

Matrix 2:

6 5 4

4 5 6

1 2 3

Sum of the matrices:

7 7 7

7 7 7

5 7 9

**Program: 2**

**Develop a stack class to hold a maximum of 10 integers with suitable methods. Develop a JAVA main method to illustrate Stack operations.**

import java.util.Scanner;

class Stack

{

private int[] elements;

private int top;

public Stack()

{

elements = new int[10];

top = -1;

}

public boolean isEmpty() {

return top == -1;

}

public boolean isFull()

{

return top == 9;

}

public void push(int element)

{

if (isFull())

{

System.out.println("Stack is full. Cannot push more elements.");

}

else

{

elements[++top] = element;

System.out.println("Pushed: " + element);

}

}

public void pop()

{

if (isEmpty())

{

System.out.println("Stack is empty. Cannot pop elements.");

}

else

{

int poppedElement = elements[top--];

System.out.println("Popped: " + poppedElement);

}

}

public void printStack()

{

if (isEmpty())

{

System.out.println("Stack is empty.");

}

else

{

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

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

{

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

}

System.out.println();

}

}

}

public class Main

{

public static void main(String[] args)

{

Stack stack = new Stack();

while(true)

{

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

System.out.println("1. Push");

System.out.println("2. Pop");

System.out.println("3. Display");

System.out.println("4. Exit");

Scanner scanner = new Scanner(System.in);

System.out.print("Enter your Choice: ");

int choice = scanner.nextInt();

switch(choice)

{

case 1: System.out.println("Enter Number to push: ");

int num = scanner.nextInt();

stack.push(num);

break;

case 2: stack.pop();

break;

case 3: stack.printStack();

break;

case 4: System.exit(0);

break;

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

}

}

}

}

**OUTPUT**

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 1

Enter Number to push:

10

Pushed: 10

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 1

Enter Number to push:

20

Pushed: 20

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 1

Enter Number to push:

30

Pushed: 30

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 1

Enter Number to push:

40

Pushed: 40

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 1

Enter Number to push:

50

Pushed: 50

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 1

Enter Number to push:

60

Pushed: 60

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 1

Enter Number to push:

70

Pushed: 70

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 1

Enter Number to push:

80

Pushed: 80

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 1

Enter Number to push:

90

Pushed: 90

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 1

Enter Number to push:

100

Pushed: 100

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 3

Stack: 10 20 30 40 50 60 70 80 90 100

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 1

Enter Number to push:

110

Stack is full. Cannot push more elements.

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 2

Popped: 100

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 2

Popped: 90

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 2

Popped: 80

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 2

Popped: 70

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 2

Popped: 60

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 2

Popped: 50

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 2

Popped: 40

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 2

Popped: 30

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 2

Popped: 20

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 2

Popped: 10

Stack Operations

1. Push

2. Pop

3. Display

4. Exit

Enter your Choice: 2

Stack is empty. Cannot pop elements

**Program: 3**

**A class called Employee, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method raiseSalary (percent) increases the salary by the given percentage. Develop the Employee class and suitable main method for demonstration.**

import java.util.Scanner;

public class Employee

{

private int empId;

private String name;

private double salary;

public Employee(int empId, String name, double salary)

{

this.empId = empId;

this.name = name;

this.salary = salary;

}

public void raiseSalary(double percentage)

{

if (percentage > 0)

{

double raiseAmount = salary \* (percentage / 100);

salary += raiseAmount;

}

}

public void displayInfo()

{

System.out.println("Employee ID: " + empId);

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

System.out.println("Salary: Rs." + String.format("%.2f", salary));

}

public static void main(String[] args)

{

// Creating an Employee object

Employee emp = new Employee(1, "Dr. Harish Kumar B T", 50000.0);

Scanner scanner = new Scanner(System.in);

// Displaying employee information before raise

System.out.println("Employee information before raise:");

emp.displayInfo();

System.out.println("Enter the percentage of salary to raise");

int percentage = scanner.nextInt();

// Raising salary by 10%

emp.raiseSalary(percentage);

// Displaying employee information after raise

System.out.println("\nEmployee information after raise:");

emp.displayInfo();

}

}

**OUTPUT**

Employee information before raise:

Employee ID: 1

Name: ROhit Sharma

Salary: Rs.50000.00

Enter the percentage of salary to raise

**Program: 4**

**A class called MyPoint, which models a 2D point with x and y coordinates, is designed as follows:**

**--Two instance variables x (int) and y (int).**

**-- A default (or "no-arg") constructor that construct a point at the default location of (0, 0).**

**-- A overloaded constructor that constructs a point with the given x and y coordinates.**

**-- A method setXY() to set both x and y.**

**-- A method getXY() which returns the x and y in a 2-element int array.**

**-- A toString() method that returns a string description of the instance in the format "(x, y)".**

**-- A method called distance(int x, int y) that returns the distance from this point to another point at the given (x, y) coordinates.**

**-- An overloaded distance(MyPoint another) that returns the distance from this point to the given MyPoint instance (called another).**

**-- Another overloaded distance() method that returns the distance from this point to the origin (0,0) Develop the code for the class MyPoint. Also develop a JAVA program (called TestMyPoint) to test all the methods defined in the class.**

public class MyPoint

{

private int x;

private int y;

// Default constructor

public MyPoint() {

this.x = 0;

this.y = 0;

}

// Overloaded constructor

public MyPoint(int x, int y)

{

this.x = x;

this.y = y;

}

// Setters for x and y

public void setXY(int x, int y)

{

this.x = x;

this.y = y;

}

// Getter for x and y

public int[] getXY()

{

int[] coordinates = {x, y};

return coordinates;

}

// Returns the string description of the instance in the format "(x, y)"

@Override

public String toString()

{

return "(" + x + ", " + y + ")";

}

// Calculates distance from this point to another point (x, y)

public double distance(int x, int y)

{

int xDiff = this.x - x;

int yDiff = this.y - y;

return Math.sqrt(xDiff \* xDiff + yDiff \* yDiff);

}

// Calculates distance from this point to another MyPoint instance

public double distance(MyPoint another)

{

int xDiff = this.x - another.x;

int yDiff = this.y - another.y;

return Math.sqrt(xDiff \* xDiff + yDiff \* yDiff);

}

// Calculates distance from this point to the origin (0, 0)

public double distance()

{

return Math.sqrt(x \* x + y \* y);

}

public static void main(String[] args)

{

MyPoint point1 = new MyPoint(); // Default constructor (0,0)

System.out.println("Point 1: " + point1);

MyPoint point2 = new MyPoint(3, 4); // Overloaded constructor (3,4)

System.out.println("Point 2: " + point2);

point1.setXY(5, 6); // Set coordinates using setXY() method

System.out.println("Point 1 after setXY(): " + point1);

int[] coordinates = point2.getXY(); // Get coordinates using getXY() method

System.out.println("Point 2 coordinates: (" + coordinates[0] + ", " + coordinates[1] + ")");

System.out.println("Distance between Point 1 and (5, 6): " + point1.distance(5, 6));

System.out.println("Distance between Point 1 and Point 2: " + point1.distance(point2));

System.out.println("Distance from Point 1 to origin: " + point1.distance());

}

}

**OUTPUT**

Point 1: (0, 0)

Point 2: (3, 4)

Point 1 after setXY(): (5, 6)

Point 2 coordinates: (3, 4)

Distance between Point 1 and (5, 6): 0.0

Distance between Point 1 and Point 2: 2.8284271247461903

Distance from Point 1 to origin: 7.810249675906654

**Program: 5**

**Develop a JAVA program to create a class named shape. Create three sub classes namely: circle, triangle and square, each class has two member functions named draw () and erase (). Demonstrate polymorphism concepts by developing suitable methods, defining member data and main program.**

// Shape class (Superclass)

class Shape

{

public void draw()

{

System.out.println("Drawing a shape");

}

public void erase()

{

System.out.println("Erasing a shape");

}

}

// Circle class (Subclass)

class Circle extends Shape

{

@Override

public void draw()

{

System.out.println("Drawing a circle");

}

@Override

public void erase()

{

System.out.println("Erasing a circle");

}

}

// Triangle class (Subclass)

class Triangle extends Shape

{

@Override

public void draw()

{

System.out.println("Drawing a triangle");

}

@Override

public void erase()

{

System.out.println("Erasing a triangle");

}

}

// Square class (Subclass)

class Square extends Shape

{

@Override

public void draw()

{

System.out.println("Drawing a square");

}

@Override

public void erase()

{

System.out.println("Erasing a square");

}

}

// Main class

public class Main

{

public static void main(String[] args)

{

// Polymorphism: Creating objects of different subclasses using the reference of the superclass

Shape shape1 = new Circle();

Shape shape2 = new Triangle();

Shape shape3 = new Square();

// Demonstrating polymorphic behavior

shape1.draw(); // Calls draw() method of Circle class

shape1.erase(); // Calls erase() method of Circle class

shape2.draw(); // Calls draw() method of Triangle class

shape2.erase(); // Calls erase() method of Triangle class

shape3.draw(); // Calls draw() method of Square class

shape3.erase(); // Calls erase() method of Square class

}

}

**OUTPUT**

Drawing a circle

Erasing a circle

Drawing a triangle

Erasing a triangle

Drawing a square

Erasing a square

**Program:**

**Develop a JAVA program to create an abstract class Shape with abstract methods calculateArea() and calculatePerimeter(). Create subclasses Circle and Triangle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.**

// Abstract Shape class

abstract class Shape

{

// Abstract methods to calculate area and perimeter

abstract double calculateArea();

abstract double calculatePerimeter();

}

// Circle class extending Shape

class Circle extends Shape

{

private double radius;

// Constructor for Circle class

public Circle(double radius)

{

this.radius = radius;

}

// Implementation of abstract method to calculate area for Circle

@Override

double calculateArea()

{

return Math.PI \* radius \* radius;

}

// Implementation of abstract method to calculate perimeter (circumference) for Circle

@Override

double calculatePerimeter()

{

return 2 \* Math.PI \* radius;

}

}

// Triangle class extending Shape

class Triangle extends Shape

{

private double side1;

private double side2;

private double side3;

// Constructor for Triangle class

public Triangle(double side1, double side2, double side3)

{

this.side1 = side1;

this.side2 = side2;

this.side3 = side3;

}

// Implementation of abstract method to calculate area for Triangle using Heron's formula

@Override

double calculateArea()

{

double s = (side1 + side2 + side3) / 2;

return Math.sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

}

// Implementation of abstract method to calculate perimeter for Triangle

@Override

double calculatePerimeter()   
{

return side1 + side2 + side3;

}

}

// Main class

public class Main

{

public static void main(String[] args)

{

// Creating Circle and Triangle objects

Circle circle = new Circle(5);

Triangle triangle = new Triangle(3, 4, 5);

// Calculating and displaying area and perimeter for Circle

System.out.println("Circle - Area: " + circle.calculateArea() + ", Perimeter: " + circle.calculatePerimeter());

// Calculating and displaying area and perimeter for Triangle

System.out.println("Triangle - Area: " + triangle.calculateArea() + ", Perimeter: " + triangle.calculatePerimeter());

}

}

**OUTPUT**

Circle - Area: 78.53981633974483, Perimeter: 31.41592653589793

Triangle - Area: 6.0, Perimeter: 12.0

**Program: 7**

**Develop a JAVA program to create an interface Resizable with methods resizeWidth(int width) and resizeHeight(int height) that allow an object to be resized. Create a class Rectangle that implements the Resizable interface and implements the resize methods.**

// Resizable interface

interface Resizable

{

void resizeWidth(int width);

void resizeHeight(int height);

}

// Rectangle class implementing Resizable interface

class Rectangle implements Resizable

{

private int width;

private int height;

// Constructor

public Rectangle(int width, int height)

{

this.width = width;

this.height = height;

}

// Implementation of resizeWidth method from Resizable interface

@Override

public void resizeWidth(int width)

{

this.width = width;

}

// Implementation of resizeHeight method from Resizable interface

@Override

public void resizeHeight(int height)

{

this.height = height;

}

// Method to display the dimensions of the rectangle

public void displayDimensions()

{

System.out.println("Width: " + width + ", Height: " + height);

}

}

// Main class

public class Main

{

public static void main(String[] args)

{

// Creating a Rectangle object

Rectangle rectangle = new Rectangle(10, 20);

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

rectangle.displayDimensions(); // Output: Width: 10, Height: 20

// Resizing the rectangle

rectangle.resizeWidth(15);

rectangle.resizeHeight(25);

System.out.println("Dimensions after resizing:");

rectangle.displayDimensions(); // Output: Width: 15, Height: 25

}

}

**OUTPUT**

Original Dimensions:

Width: 10, Height: 20

Dimensions after resizing:

Width: 15, Height: 25

**Program: 8**

**Develop a JAVA program to create an outer class with a function display. Create another class inside the outer class named inner with a function called display and call the two functions in the main class.**

// OuterClass containing an inner class Inner

class OuterClass

{

// Outer class display method

public void display()

{

System.out.println("OuterClass display method");

}

// Inner class

class Inner

{

// Inner class display method

public void display()

{

System.out.println("InnerClass display method");

}

}

}

// Main class

public class Main

{

public static void main(String[] args)

{

// Creating an object of OuterClass

OuterClass outerObject = new OuterClass();

// Calling display method of OuterClass

outerObject.display(); // Output: OuterClass display method

// Creating an object of Inner class (inside OuterClass)

OuterClass.Inner innerObject = outerObject.new Inner();

// Calling display method of Inner class

innerObject.display(); // Output: InnerClass display method

}

}

**OUTPUT**

OuterClass display method

InnerClass display method

**Program: 9**

**Develop a JAVA program to raise a custom exception (user defined exception) for DivisionByZero using try, catch, throw and finally.**

// Custom exception class (extends Exception)

class DivisionByZeroException extends Exception

{

public DivisionByZeroException(String message)

{

super(message);

}

}

// Main class

public class Main

{

public static void main(String[] args)

{

int numerator = 10;

int denominator = 0;

try

{

// Attempting division

if (denominator == 0)

{

// Throwing custom exception if denominator is zero

throw new DivisionByZeroException("Division by zero is not allowed.");

}

int result = numerator / denominator;

System.out.println("Result of division: " + result);

}

catch (DivisionByZeroException e)

{

// Catching and handling the custom exception

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

}

finally

{

// Code in the finally block will always execute, whether an exception occurs or not

System.out.println("Finally block executed.");

}

}

}

**Create a Java file named MainClass.java in a different package (let's say mainpack).**

**MainClass.java (Inside mainpack package):**

package mainpack; // Package declaration

import mypack.MyPackageClass; // Importing the class from mypack package

public class MainClass

{

public static void main(String[] args)

{

MyPackageClass myPackageObj = new MyPackageClass();

myPackageObj.display(); // Calling the display method from MyPackageClass

}

}

**OUTPUT**

Exception caught: Division by zero is not allowed.

Finally block executed.

Result of division: 5

Finally block executed.

**Program: 10**

**Develop a JAVA program to create a package named mypack and import & implement it in a suitable class.**

**Create a directory named mypack in your project's source folder.**

**Create a Java File inside the mypack Package:**

**Inside the mypack directory, create a Java file named MyPackageClass.java.**

**MyPackageClass.java (Inside mypack package):**

// Class inside the mypack package

package mypack; // Package declaration

public class MyPackageClass

{

public void display()

{

System.out.println("Hello from MyPackageClass in mypack package!");

}

}

**OUTPUT**

Hello from MyPackageClass in mypack package!

**Program: 11**

**Write a program to illustrate creation of threads using runnable class. (start method start each of the newly created thread. Inside the run method there is sleep() for suspend the thread for 500**

**milliseconds).**

// Runnable class implementation

class MyRunnable implements Runnable

{

public void run()

{

for (int i = 1; i <= 5; i++)

{

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

try

{

Thread.sleep(500); // Suspend the thread for 500 milliseconds

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

}

}

// Main class

public class Main

{

public static void main(String[] args)

{

// Creating Runnable objects

MyRunnable myRunnable1 = new MyRunnable();

MyRunnable myRunnable2 = new MyRunnable();

// Creating threads and starting them

Thread thread1 = new Thread(myRunnable1);

Thread thread2 = new Thread(myRunnable2);

// Starting threads using the start() method

thread1.start();

thread2.start();

}

}

**OUTPUT**

Thread 14: 1

Thread 15: 1

Thread 15: 2

Thread 14: 2

Thread 15: 3

Thread 14: 3

Thread 14: 4

Thread 15: 4

Thread 14: 5

Thread 15: 5

**Program: 12**

**Develop a program to create a class MyThread in this class a constructor, call the base class constructor, using super and start the thread. The run method of the class starts after this. It can be observed that both main thread and created child thread are executed concurrently.**

// Custom thread class MyThread extending Thread

class MyThread extends Thread

{

// Constructor calling the base class constructor and starting the thread

public MyThread(String name)

{

super(name);

start(); // Start the thread when the constructor is called

}

// Run method to be executed when the thread starts

public void run()

{

for (int i = 1; i <= 5; i++)

{

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

try

{

Thread.sleep(500); // Suspend the thread for 500 milliseconds

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

}

}

// Main class

public class Main

{

public static void main(String[] args)

{

// Main thread executing concurrently with the MyThread instance

for (int i = 1; i <= 5; i++)

{

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

try

{

Thread.sleep(500); // Suspend the main thread for 500 milliseconds

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

// Creating an instance of MyThread

MyThread myThread = new MyThread("Child Thread");

// Main thread and child thread executing concurrently

}

}

**OUTPUT**

Main Thread: 1

Main Thread: 2

Main Thread: 3

Main Thread: 4

Main Thread: 5

Child Thread: 1

Child Thread: 2

Child Thread: 3

Child Thread: 4

Child Thread: 5