

SHREE DEVI INSTITUTE OF TECHNOLOGY

Kenjar Mangalore

**DEPARTMENT OF INFORMATION SCIENCE AND
ENGINEERING**

**OBJECT ORIENTED PROGRAMMING WITH
JAVA INTEGRATED LAB MANUAL**

BCS306A

List of Experiments

1	Implement Three nodes point – to – point network with duplex links between them Develop a JAVA program to add TWO matrices of suitable order N (The value of N should be read from command line arguments).
2	Develop a stack class to hold a maximum of 10 integers with suitable methods. Develop a JAVA main method to illustrate Stack operations.
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.
4	<p>A class called MyPoint, which models a 2D point with x and y coordinates, is designed as follows:</p> <ul style="list-style-type: none"> • 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 getX() which returns the x 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) <p>Develop the code for the class MyPoint. Also develop a JAVA program (called TestMyPoint) to test all the methods defined in the class</p>
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.

6	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.
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
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.
9	Develop a JAVA program to raise a custom exception (user defined exception) for DivisionByZero using try, catch, throw and finally.
10	Develop a JAVA program to create a package named mypack and import & implement it in a suitable class.
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).
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.

1. Implement Three nodes point – to – point network with duplex links between them
Develop a JAVA program to add TWO matrices of suitable order N (The value of N should be read from command line arguments).

Command: gedit MatrixAddition.java

Javac MatrixAddition.java

Java MatrixAddition 2

Program:

```
public class MatrixAddition {
    public static void main(String[] args) {
        // Check if the number of command line arguments is correct
        if (args.length != 1) {
            System.out.println("Usage: java MatrixAddition <order N>");
            return;
        }
        // Parse the order N from the command line argument
        int N = Integer.parseInt(args[0]);

        // Check if the order N is valid
        if (N <= 0) {
            System.out.println("Invalid order N. Please enter a positive integer.");
            return;
        }
        // Create two matrices of order N
        int[][] matrix1 = createMatrix(N);
        int[][] matrix2 = createMatrix(N);
        // Print the matrices
        System.out.println("Matrix 1:");
        printMatrix(matrix1);

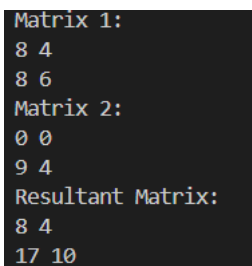
        System.out.println("Matrix 2:");
        printMatrix(matrix2);
        // Add the matrices
        int[][] result = addMatrices(matrix1, matrix2);

        // Print the result
        System.out.println("Resultant Matrix:");
        printMatrix(result);
    }
    // Method to create a matrix of order N with random values
    private static int[][] createMatrix(int N) {
        int[][] matrix = new int[N][N];
```

```
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                matrix[i][j] = (int) (Math.random() * 10); // Assign random values between 0 and 9
            }
        }
        return matrix;
    }
    // Method to add two matrices
    private static int[][] addMatrices(int[][] matrix1, int[][] matrix2) {
        int N = matrix1.length;
        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];
            }
        }
        return result;
    }

    // Method to print a matrix
    private static void printMatrix(int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Output :



```
Matrix 1:
8 4
8 6
Matrix 2:
0 0
9 4
Resultant Matrix:
8 4
17 10
```

2. Develop a stack class to hold a maximum of 10 integers with suitable methods.

Develop a JAVA main method to illustrate Stack operations.

Command : gedit Main.java

Javac Main.java

Java Main

```
import java.util.Scanner;

// Stack class to represent a stack of integers
class Stack {
    // Constants for stack size
    private static final int MAX_SIZE = 10;

    // Variables to track the top of the stack and the stack array
    private int top;
    private int[] stackArray;

    // Constructor to initialize the stack
    public Stack() {
        top = -1;
        stackArray = new int[MAX_SIZE];
    }

    // Method to check if the stack is empty
    public boolean isEmpty() {
        return top == -1;
    }

    // Method to check if the stack is full
    public boolean isFull() {
        return top == MAX_SIZE - 1;
    }

    // Method to push an element onto the stack
    public void push(int value) {
        if (isFull()) {
            System.out.println("Stack overflow. Cannot push element " + value);
        } else {
            stackArray[++top] = value;
            System.out.println(value + " pushed to the stack.");
        }
    }
}
```

```
// Method to pop an element from the stack
public void pop() {
    if (isEmpty()) {
        System.out.println("Stack underflow. Cannot pop from an empty stack.");
    } else {
        int poppedValue = stackArray[top--];
        System.out.println("Popped value: " + poppedValue);
    }
}

// Method to display the elements in the stack
public void display() {
    if (isEmpty()) {
        System.out.println("Stack is empty.");
    } else {
        System.out.print("Stack: ");
        for (int i = 0; i <= top; i++) {
            System.out.print(stackArray[i] + " ");
        }
        System.out.println();
    }
}

// Main class to illustrate stack operations
public class Main {
    public static void main(String[] args) {
        // Create an instance of the Stack class
        Stack stack = new Stack();

        // Create a Scanner object for user input
        Scanner scanner = new Scanner(System.in);

        // Variables for user choice
        int choice;

        // Menu-driven loop for stack operations
        do {
            // Display menu options
            System.out.println("\nStack Operations:");
            System.out.println("1. Push");
            System.out.println("2. Pop");
            System.out.println("3. Display");
            System.out.println("4. Exit");
        }
```

```
System.out.print("Enter your choice: ");

try {
    // Read user input as an integer
    choice = Integer.parseInt(scanner.nextLine());
} catch (NumberFormatException e) {
    // Handle the case where input is not an integer
    choice = 0;
}

// Perform stack operation based on user choice
switch (choice) {
    case 1:
        // Push operation
        System.out.print("Enter the value to push: ");
        try {
            // Read the value to push as an integer
            int valueToPush = Integer.parseInt(scanner.nextLine());
            stack.push(valueToPush);
        } catch (NumberFormatException e) {
            // Handle the case where input is not an integer
            System.out.println("Invalid input. Please enter a valid integer.");
        }
        break;
    case 2:
        // Pop operation
        stack.pop();
        break;
    case 3:
        // Display operation
        stack.display();
        break;
    case 4:
        // Exit the program
        System.out.println("Exiting the program.");
        break;
    default:
        // Handle the case of an invalid choice
        System.out.println("Invalid choice. Please enter a valid option.");
    }
} while (choice != 4);
scanner.close();
}
```


Output :

Stack Operations:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

Enter the value to push: 10

10 pushed to the stack.

Stack Operations:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

Enter the value to push: 20

20 pushed to the stack.

Stack Operations:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

Enter the value to push: 30

30 pushed to the stack.

Stack Operations:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

Enter the value to push: 40

40 pushed to the stack.

Stack Operations:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

Enter the value to push: 50

50 pushed to the stack.

Stack Operations:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 1

Enter the value to push: 60

60 pushed to the stack.

Stack Operations:

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter the value to push: 70

70 pushed to the stack.

Stack Operations:

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter the value to push: 80

80 pushed to the stack.

Stack Operations:

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter the value to push: 90

90 pushed to the stack.

Stack Operations:

1. Push

2. Pop

3. Display

4. Exit

Enter your choice: 1

Enter the value to push: 100

100 pushed to the stack.

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

Enter your choice: 1

Enter the value to push: 110

Stack overflow. Cannot push element 110

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
Enter your choice: 1
Enter the value to push: i
Invalid input. Please enter a valid integer.
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
Enter your choice: 2
Popped value: 100
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Popped value: 90
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Popped value: 80
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack: 10 20 30 40 50 60 70

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2

Popped value: 70
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Popped value: 60
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Popped value: 50
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Popped value: 40
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Popped value: 30
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Popped value: 20
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack: 10
Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
Popped value: 10

Stack Operations:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 2

Stack underflow. Cannot pop from an empty stack.

Stack Operations:

1. Push
2. Pop
3. Display
4. Exit

Enter your choice: 4

Exiting the 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 id;
    private String name;
    private double salary;

    // Constructor
    public Employee(int id, String name, double salary)
    {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }

    // Getter methods
    public int getId()
    {
        return id;
    }

    public String getName()
    {
        return name;
    }

    public double getSalary()
    {
        return salary;
    }

    // Method to raise salary by a given percentage
    public void raiseSalary(double percent)
    {
        if (percent > 0)
        {
            salary += salary * (percent / 100.0);
            System.out.println(name + " 's salary has been increased by " + percent + "%.");
        }
    }
}
```

```
    }  
    else  
    {  
        System.out.println("Invalid percentage. Salary remains unchanged.");  
    }  
}
```

// Main method for demonstration

```
public static void main(String[] args)
```

```
{  
    Scanner sc = new Scanner(System.in);
```

// Get employee details from the user

```
    System.out.print("Enter Employee ID: ");  
    int id = sc.nextInt();
```

```
    System.out.print("Enter Employee Name: ");  
    sc.nextLine(); // Consume the newline character left by nextInt()  
    String name = sc.nextLine();
```

```
    System.out.print("Enter Employee Salary: ");  
    double salary = sc.nextDouble();
```

```
    Employee employee1 = new Employee(id, name, salary);
```

// Displaying initial details

```
    System.out.println("\nInitial Details:");  
    System.out.println("Employee ID: " + employee1.getId());  
    System.out.println("Employee Name: " + employee1.getName());  
    System.out.println("Employee Salary: " + employee1.getSalary());
```

```
    System.out.print("\nEnter the percentage increase: ");  
    double percentageIncrease = sc.nextDouble();
```

// Raising salary by the user-provided percentage

```
    employee1.raiseSalary(percentageIncrease);
```

```
    System.out.println("\nDetails after Salary Increase:");  
    System.out.println("Employee ID: " + employee1.getId());  
    System.out.println("Employee Name: " + employee1.getName());  
    System.out.println("Employee Salary: " + employee1.getSalary());
```

```
    sc.close();  
} }
```

Output :

Enter Employee ID: 101

Enter Employee Name: AMG

Enter Employee Salary: 45000

Initial Details:

Employee ID: 101

Employee Name: AMG

Employee Salary: 45000.0

Enter the percentage increase: 10

AMG's salary has been increased by 10.0%.

Details after Salary Increase:

Employee ID: 101

Employee Name: AMG

Employee Salary: 49500.0

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 getX() which returns the x 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.

gedit Mypoint.java

- write the program which has **MyPoint** as a class name, then save the program and close it.

gedit TestMyPoint.java

- write the program which has **TestMyPoint** as a class name, then save the program and close it.

Javac MyPoint.java

Javac TestMyPoint.java

Java TestMyPoint

MyPoint.java

```
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;
}

        // Set both x and y
public void setXY(int x, int y) {
    this.x = x;
    this.y = y;
}

        // Get x and y in a 2-element int array
public int[] getXY() {
    return new int[]{x, y};
}

        // Return a string description of the instance in the format "(x, y)"
public String toString() {
    return "(" + x + ", " + y + ")";
}

        // Calculate distance from this point to another point at (x, y) coordinates
public double distance(int x, int y) {
    int xDiff = this.x - x;
    int yDiff = this.y - y;
    return Math.sqrt(xDiff * xDiff + yDiff * yDiff);
}

        // Calculate distance from this point to another MyPoint instance (another)
public double distance(MyPoint another) {
    return distance(another.x, another.y);
}

        // Calculate distance from this point to the origin (0,0)
public double distance() {
    return distance(0, 0);
}
}
```

TestMyPoint.java

```
public class TestMyPoint {
    public static void main(String[] args) {
        // Creating MyPoint objects using different constructors
        MyPoint point1 = new MyPoint();
        MyPoint point2 = new MyPoint(3, 4);

        // Testing setXY and getXY methods
        point1.setXY(1, 2);
        System.out.println("Point1 coordinates after setXY: " + point1.getXY()[0] + ", " +
point1.getXY()[1]);

        // Testing toString method
        System.out.println("Point2 coordinates: " + point2.toString());
        System.out.println("Distance from Point1 to Point2: " + point1.distance(point2));
        System.out.println("Distance from Point2 to Origin: " + point2.distance());
    }
}
```

O/P :

```
java TestMyPoint.java
```

```
Point1 coordinates after setXY: 1, 2
Point2 coordinates: (3, 4)
Distance from Point1 to Point2: 2.8284271247461903
Distance from Point2 to Origin: 5.0
```

Program 05 : Inheritance & Polymorphism – Shape Class.

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 ()

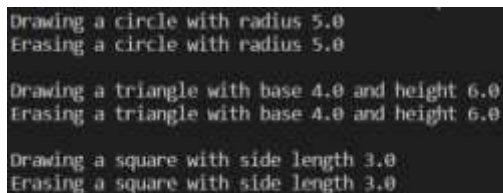
Demonstrate polymorphism concepts by developing suitable methods, defining member data and main program.(**ShapeDemo.java**)

```
class Shape {
    protected String name;
    public Shape(String name) {
        this.name = name;
    }
    public void draw() {
        System.out.println("Drawing a " + name);
    }
    public void erase() {
        System.out.println("Erasing a " + name);
    }
}
class Circle extends Shape {
    private double radius;
    public Circle(String name, double radius) {
        super(name);
        this.radius = radius;
    }
    public void draw() {
        System.out.println("Drawing a circle with radius " + radius);
    }

    public void erase() {
        System.out.println("Erasing a circle with radius " + radius);
    }
}
class Triangle extends Shape {
    private double base;
    private double height;
    public Triangle(String name, double base, double height) {
        super(name);
        this.base = base;
        this.height = height;
    }
    public void draw() {
        System.out.println("Drawing a triangle with base " + base + " and height " + height);
    }
}
```

```
    }  
    public void erase() {  
        System.out.println("Erasing a triangle with base " + base + " and height " + height);  
    }  
}  
class Square extends Shape {  
    private double side;  
  
    public Square(String name, double side) {  
        super(name);  
        this.side = side;  
    }  
    public void draw() {  
        System.out.println("Drawing a square with side length " + side);  
    }  
    public void erase() {  
        System.out.println("Erasing a square with side length " + side);  
    }  
}  
public class ShapeDemo {  
    public static void main(String[] args) {  
        Shape[] shapes = new Shape[3];  
  
        shapes[0] = new Circle("Circle", 5.0);  
        shapes[1] = new Triangle("Triangle", 4.0, 6.0);  
        shapes[2] = new Square("Square", 3.0);  
  
        for (Shape shape : shapes) {  
            shape.draw();  
            shape.erase();  
            System.out.println();  
        }  
    }  
}
```

Output :



```
Drawing a circle with radius 5.0  
Erasing a circle with radius 5.0  
  
Drawing a triangle with base 4.0 and height 6.0  
Erasing a triangle with base 4.0 and height 6.0  
  
Drawing a square with side length 3.0  
Erasing a square with side length 3.0
```

Program 06 : Abstract Class (AbstractShapeDemo.java)

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 class Shape {
    abstract double calculateArea();
    abstract double calculatePerimeter();
}

class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
        this.radius = radius;
    }

    double calculateArea() {
        return Math.PI * radius * radius;
    }

    double calculatePerimeter() {
        return 2 * Math.PI * radius;
    }
}

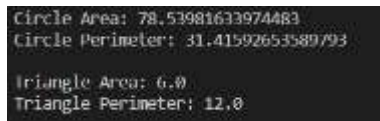
class Triangle extends Shape {
    private double side1;
    private double side2;
    private double side3;

    public Triangle(double side1, double side2, double side3) {
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }

    double calculateArea() {
        // Using Heron's formula to calculate the area of a triangle
        double s = (side1 + side2 + side3) / 2;
        return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
    }
}
```

```
double calculatePerimeter() {  
    return side1 + side2 + side3;  
}  
}  
  
public class ShapeDemo {  
    public static void main(String[] args) {  
        // Creating Circle and Triangle objects  
        Circle circle = new Circle(5.0);  
        Triangle triangle = new Triangle(3.0, 4.0, 5.0);  
  
        // Calculating and displaying area and perimeter  
        System.out.println("Circle Area: " + circle.calculateArea());  
        System.out.println("Circle Perimeter: " + circle.calculatePerimeter());  
  
        System.out.println("\nTriangle Area: " + triangle.calculateArea());  
        System.out.println("Triangle Perimeter: " + triangle.calculatePerimeter());  
    }  
}
```

Output :

A screenshot of a terminal window showing the output of the Java program. The output consists of four lines: "Circle Area: 78.53981633974483", "Circle Perimeter: 31.41592653589793", "Triangle Area: 6.0", and "Triangle Perimeter: 12.0".

```
Circle Area: 78.53981633974483  
Circle Perimeter: 31.41592653589793  
  
Triangle Area: 6.0  
Triangle Perimeter: 12.0
```

Prog 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.

Gedit ResizeDemo.java

```
// 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;

    public Rectangle(int width, int height) {
        this.width = width;
        this.height = height;
    }

    // Implementation of Resizable interface
    // @Override
    public void resizeWidth(int width) {
        this.width = width;
        System.out.println("Resized width to: " + width);
    }

    // @Override
    public void resizeHeight(int height) {
        this.height = height;
        System.out.println("Resized height to: " + height);
    }

    // Additional methods for Rectangle class
    public int getWidth() {
        return width;
    }

    public int getHeight() {
        return height;
    }
}
```



```
public void displayInfo() {
    System.out.println("Rectangle: Width = " + width + ", Height = " + height);
}

// Main class to test the implementation
public class ResizeDemo {
    public static void main(String[] args) {
        // Creating a Rectangle object
        Rectangle rectangle = new Rectangle(10, 5);

        // Displaying the original information
        System.out.println("Original Rectangle Info:");
        rectangle.displayInfo();

        // Resizing the rectangle
        rectangle.resizeWidth(15);
        rectangle.resizeHeight(8);

        // Displaying the updated information
        System.out.println("\nUpdated Rectangle Info:");
        rectangle.displayInfo();
    }
}
```

OUTPUT :

```
Original Rectangle Info:
Rectangle: Width = 10, Height = 5
Resized width to: 15
Resized height to: 8

Updated Rectangle Info:
Rectangle: Width = 15, Height = 8
```

Prog 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. (OuterInnerDemo.java)

```
class Outer {
    void display() {
        System.out.println("Outer class display method");
    }

    class Inner {
        void display() {
            System.out.println("Inner class display method");
        }
    }
}

public class OuterInnerDemo {
    public static void main(String[] args) {
        // Create an instance of the Outer class
        Outer outer = new Outer();

        // Call the display method of the Outer class
        outer.display();

        // Create an instance of the Inner class (nested inside Outer)
        Outer.Inner inner = outer.new Inner();

        // Call the display method of the Inner class
        inner.display();
    }
}
```

OUTPUT :

```
Outer class display method
Inner class display method
```

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

```
import java.util.Scanner;

// Custom exception class
class DivisionByZeroException extends Exception {
    public DivisionByZeroException(String message) {
        super(message);
    }
}

public class CustomExceptionDemo {
    // Method to perform division and throw custom exception if denominator is zero
    static double divide(int numerator, int denominator) throws DivisionByZeroException {
        if (denominator == 0) {
            throw new DivisionByZeroException("Cannot divide by zero!");
        }
        return (double) numerator / denominator;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the numerator: ");
        int numerator = scanner.nextInt();
        System.out.print("Enter the denominator: ");
        int denominator = scanner.nextInt();
        try {
            double result = divide(numerator, denominator);
            System.out.println("Result of division: " + result);
        } catch (DivisionByZeroException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }
}
```

```
    } finally {  
        System.out.println("Finally block executed");  
        scanner.close();  
    }  
}  
}
```

Output :

```
Enter the numerator: 2  
Enter the denominator: 0  
Exception caught: Cannot divide by zero!  
Finally block executed
```

Output :

```
Enter the numerator: 2  
Enter the denominator: 2  
Result of division: 1.0  
Finally block executed
```

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

MyPackageClass.java

```
package mypack;
public class MyPackageClass
{
    // method to add two numbers and return the same
    public static int addNumbers(int a, int b)
    {
        System.out.println("Hello from MyPackageClass in mypack package!");
        return a + b;
    }
}
```

PackageDemo.java

```
// Main program outside the mypack folder
import mypack.MyPackageClass;
// import mypack.*;
public class PackageDemo {
    public static void main(String[] args)
    {
        // Creating an instance of MyPackageClass from the mypack package
        MyPackageClass PackageObject = new MyPackageClass();

        // Using the utility method addNumbers from MyPackageClass
        int result = MyPackageClass.addNumbers(5, 3);
        System.out.println("Result of adding numbers: " + result);
    }
}
```

Output :

Hello from MyPackageClass in mypack package!
Result of adding numbers: 8

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).

```
class MyRunnable implements Runnable {  
    private volatile boolean running = true;  
    public void run() {  
        while (running) {  
            try {  
                // Suppress deprecation warning for Thread.sleep()  
                Thread.sleep(500);  
                System.out.println("Thread ID: " + Thread.currentThread().getId() + " is running.");  
            } catch (InterruptedException e) {  
                System.out.println("Thread interrupted.");  
            }  
        }  
    }  
    public void stopThread() {  
        running = false;  
    }  
}  
  
public class RunnableThreadClass {  
    public static void main(String[] args) {  
        // Create five instances of MyRunnable  
        MyRunnable myRunnable1 = new MyRunnable();  
        MyRunnable myRunnable2 = new MyRunnable();  
        MyRunnable myRunnable3 = new MyRunnable();  
        MyRunnable myRunnable4 = new MyRunnable();  
        MyRunnable myRunnable5 = new MyRunnable();  
    }  
}
```

```
// Create five threads and associate them with MyRunnable instances
Thread thread1 = new Thread(myRunnable1);
Thread thread2 = new Thread(myRunnable2);
Thread thread3 = new Thread(myRunnable3);
Thread thread4 = new Thread(myRunnable4);
Thread thread5 = new Thread(myRunnable5);

// Start the threads
thread1.start();
thread2.start();
thread3.start();
thread4.start();
thread5.start();

// Sleep for a while to allow the threads to run
try {
    Thread.sleep(500);
} catch (InterruptedException e) {
    e.printStackTrace();
}

// Stop the threads gracefully
myRunnable1.stopThread();
myRunnable2.stopThread();
myRunnable3.stopThread();
myRunnable4.stopThread();
myRunnable5.stopThread();
}
}
```

Output :

Thread ID: 23 is running.

Thread ID: 24 is running.

Thread ID: 22 is running.

Thread ID: 25 is running.

Thread ID: 21 is running.

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.

```
class MyThread extends Thread {
    // Constructor calling base class constructor using super
    public MyThread(String name) {
        super(name);
        start(); // Start the thread in the constructor
    }

    // The run method that will be executed when the thread starts
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getName() + " Count: " + i);
            try {
                Thread.sleep(500); // Sleep for 500 milliseconds
            } catch (InterruptedException e) {
                System.out.println(Thread.currentThread().getName() + " Thread interrupted.");
            }
        }
    }
}

public class ThreadConcurrentExample {
    public static void main(String[] args) {
        // Create an instance of MyThread
        MyThread myThread = new MyThread("Child Thread");

        // Main thread
        for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getName() + " Thread Count: " + i);
            try {
                Thread.sleep(500); // Sleep for 500 milliseconds
            } catch (InterruptedException e) {
                System.out.println(Thread.currentThread().getName() + " Thread interrupted.");
            }
        }
    }
}
```

Output :

Child Thread Count: 1

main Thread Count: 1

main Thread Count: 2

Child Thread Count: 2

Child Thread Count: 3

main Thread Count: 3

main Thread Count: 4

Child Thread Count: 4

main Thread Count: 5

Child Thread Count: 5

