EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering

Jnanaprabha, Virgo Nagar Post, Bengaluru-560049

Academic Year: 2023-24

LABORATORY MANUAL

SEMESTER : III

SUBJECT : OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY

SUBCODE : BCS306A

NAME : ____________________

USN : ____________________

SECTION: ____________________

PROGRAM OUTCOMES

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activitieswith an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and Team Work: Function effectively as an individual and as a member or leader in diverse teams, and in multi – disciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the Engineering and management principles and apply these to one’s own work, as a member andleader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life -long learning in the broadest context of technological change.

EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering

INSTITUTE VISION AND MISSION

VISION

The East Point College of Engineering and Technology aspires to be a globally acclaimed institution, recognized for excellence in engineering education, applied research and nurturing students for holistic development.

MISSION

M1: To create engineering graduates through quality education and to nurtureinnovation, creativity and excellence in teaching, learning and research

M2: To serve the technical, scientific, economic and societal developmental needsof our communities

M3: To induce integrity, teamwork, critical thinking, personality development andethics in students and to lay the foundation for lifelong learning

EAST POINT COLLEGE OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering

DEPARTMENT VISION AND MISSION

VISION

The department aspires to be a Centre of excellence in Computer Science & Engineering to develop competent professionals through holistic development.

MISSION

M1: To create successful Computer Science Engineering graduates through effective pedagogies, the latest tools and technologies, and excellence in teaching and learning.

M2: To augment experiential learning skills to serve technical, scientific, economic, and social developmental needs.

M3: To instil integrity, critical thinking, personality development, and ethics in students for a successful career in Industries, Research, and Entrepreneurship.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO 1: To produce graduates who can perform technical roles to contribute effectively in software industries and R&D Centre

PEO 2: To produce graduates having the ability to adapt and contribute in key domains of computer science and engineering to develop competent solutions.

PEO 3: To produce graduates who can provide socially and ethically responsible solutions while adapting to new trends in the domain to carve a successful career in the industry

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: To conceptualize, model, design, simulate, analyse, develop, test, and validate computing systems and solve technical problems arising in the field of computer science & engineering.

PSO2: To specialize in the sub-areas of computer science & engineering systems such as cloud computing, Robotic Process Automation, cyber security, big data analytics, user interface design, and IOT to meet industry requirements.

PSO3: To build innovative solutions to meet the demands of the industry using appropriate tools and techniques.

COURSE LEARNING OBJECTIVES

CLO 1: To learn primitive constructs Java programming language.

CLO 2: To understand Object Oriented Programming features of Java.

CLO 3: To gain knowledge on: packages, multithreaded programming and exceptions.

COURSE OUTCOMES

At the end of the course the student will be able to:

CO1: Demonstrate proficiency in writing simple programs involving branching and looping structures.

CO2: Design a class involving data members and methods for the given scenario.

CO3: Apply the concepts of inheritance and interfaces in solving real world problems.

CO4: Use the concept of packages and exception handling in solving complex problem.

CO5: Apply concepts of multithreading, autoboxing and enumerations in program development.

OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY

(Effective from the academic year 2023 -2024)

SEMESTER - III

Credits – 3

Course Learning Objectives: This course (BCS306A) will enable students to:

- To learn primitive constructs Java programming language.
- To understand Object Oriented Programming features of Java.
- To gain knowledge on: packages, multithreaded programming and exceptions.

Descriptions (if any):

- Design, develop, and implement the specified algorithms for the following problems using Java language under LINUX /Windows environment. Netbeans / Eclipse or IntellijIdea Community Edition IDE tool can be used for development and demonstration.
- Installation procedure of the required software must be demonstrated, carried out ingroups and documented in the journal.

Programs List:

1	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	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.
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 Trianngle 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 and implement it in a suitable class.
11	Write a java 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.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE)is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures not less than 35% (18 Marks out of 50) in the semester-end examination (SEE), and a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

Continuous Internal Evaluation (CIE) for the practical component of the IPCC:

- 15 marks of the conduction of the experiment and preparation of laboratory record, and **10 marks** for the test to be conducted after the completion of all the laboratory sessions.
- On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voce and marks shall be awarded on the same day.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to **15 marks**.
- The laboratory test **(duration 02/03 hours)** after completion of all the experiments shall be conducted for 50 marks and scaled down to **10 marks.**
- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for **25 marks**.
- The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

Semester End Evaluation (SEE):

Theory SEE will be conducted by University as per the scheduled timetable, with commonquestion papers for the course (duration 03 hours).

Index

	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 Trianngle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.	CO3	PO1, PO2, PO3, PO5, PO9, PO12, PSO1,2,3	L3	30
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.	CO3	PO1, PO2, PO3, PO5, PO9, PO12, PSO1,2,3	L2	34
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.	CO3	PO1, PO2, PO3, PO5, PO9, PO12, PSO1,2,3	L2	38
9	Develop a java program to raise a custom exception (user defined exception) for DivisionByZero using try, catch, throw and finally.	CO4	PO1, PO2, PO3, PO5, PO9, PO12, PSO1,2,3	L2	41
10	Develop a java program to create a package named mypack and import and implement it in a suitable class.	CO4	PO1, PO2, PO3, PO5, PO9, PO12, PSO1,2,3	L2	45
11	Write a java 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).	CO5	PO1, PO2, PO3, PO5, PO12, PSO1,2,3	L1	47
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.	CO5	PO1, PO2, PO3, PO5, PO12, PSO1,2,3	L3	50
13	Develop a java program that demonstrates the usage of autoboxing and unboxing.	CO5	PO1, PO2, PO3, PO5, PO12, PSO1,2,3	L3	53

Course Articulation Matrix

COs	**POs**												**PSOs**		
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1	3	3	2	-	3	-	-	-	-	-	-	2	3	1	1
CO2	3	3	3	-	3	-	-	-	-	-	-	2	3	1	1
CO3	3	3	3	-	3	-	-	-	2	-	-	2	3	1	1
CO4	3	3	3	-	3	-	-	-	2	-	-	2	3	1	1
CO5	3	3	3	-	3	-	-	-	-	-	-	2	3	1	1

3 - High Correlation **2 - Medium Correlation** **1 – Low Correlation**

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).

Program:

```
public class MatrixAddition {
    public static void main(String[] args) {
        // Check if the user has provided the value of N as a command line argument
        if (args.length != 1) {
            System.out.println("Please provide the value of N as a command line argument.");
            return;
        }
        try {
            // Parse the value of N from the command line argument
            int N = Integer.parseInt(args[0]);
            // Create two random matrices of size N x N
            int[][] matrix1 = generateRandomMatrix(N);
            int[][] matrix2 = generateRandomMatrix(N);
            // Print the original 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);
        } catch (NumberFormatException e) {
            System.out.println("Invalid input. Please provide a valid integer for N.");
        }
    }
    // Generate a random matrix of size N x N
    public static int[][] generateRandomMatrix(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() * 100); // Fill with random integers (adjust the range)
            }
        }
        return matrix;
    }
    // Add two matrices
    public 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;
    }
    // Print a matrix
    public static void printMatrix(int[][] matrix) {
        int N = matrix.length;
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                System.out.print(matrix[i][j] + "  ");
            }
            System.out.println();
        }
    }
}
```

Output:

```
java MatrixAddition 2
Matrix 1:
99  54
52  82
Matrix 2:
71  26
73  25
Resultant Matrix:
170  80
125  107

java MatrixAddition 3
Matrix 1:
0  30  60
29  83  5
41  39  21
Matrix 2:
69  41  89
42  56  46
14  66  90
Resultant Matrix:
69  71  149
71  139  51
55  105  111
```

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.

Program:

```
//This class defines an integer stack that can hold 10 values.
class Stack
{
    int stck[] = new int[10];
    int tos;
    // Initialize top-of-stack
    Stack()
    {
        tos = -1;
    }
    //push an item onto the stack
    void push(int item)
    {
        if(tos==9)
            System.out.println("Stack is full");
        else
            stck[++tos] = item;
    }
    //Pop an item from the stack
    int pop()
    {
        if(tos<0)
        {
            System.out.println("Stack underflow");
            return 0;
        }
        else
            return stck[tos--];
    }
}
public class TestStack
{
    public static void main(String [] args)
    {
        Stack mystack1 = new Stack();
        Stack mystack2 = new Stack();

        //push some numbers onto the stack
        for(int i=0; i<10; i++)
            mystack1.push(i);
        for(int i=10; i<20;i++)
```

```
                mystack2.push(i);

            //Pop those numbers off the stack
            System.out.println("Stack in mystack1:");
            for(int i=0; i<10; i++)
                System.out.println(mystack1.pop());
            System.out.println("Stack in mystack2:");
            for(int i=0; i<10; i++)
                System.out.println(mystack2.pop());
    }
}
```

Output:

```
java TestStack
Stack in mystack1:
9
8
7
6
5
4
3
2
1
0
Stack in mystack2:
19
18
17
16
15
14
13
12
11
10
```

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.

Program:

```
public class Employee {
    private int id;
    private String name;
    private double salary;
    public Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }
    public void raiseSalary(double percent) {
        if (percent > 0) {
            double increase = salary * (percent / 100.0);
            salary += increase;
            System.out.println(name + "'s salary has been increased by " + percent + "%. New salary: $" +
salary);
        } else {
            System.out.println("Invalid percentage. Salary remains unchanged.");
        }
    }
    public void displayInfo() {
        System.out.println("Employee ID: " + id);
        System.out.println("Employee Name: " + name);
        System.out.println("Employee Salary: $" + salary);
    }
    public static void main(String[] args) {
        // Create an Employee object
        Employee employee1 = new Employee(101, "John Doe", 50000.0);
        // Display employee information
        System.out.println("Employee Information (Before Raise):");
        employee1.displayInfo();
        // Raise employee's salary by 10%
        employee1.raiseSalary(10);
        // Display updated employee information
        System.out.println("\nEmployee Information (After Raise):");
        employee1.displayInfo();
    }
}
```

Output:

```
java Employee
Employee Information (Before Raise):
Employee ID: 101
Employee Name: John Doe
Employee Salary: $50000.0
John Doe's salary has been increased by 10.0%. New salary: $55000.0

Employee Information (After Raise):
Employee ID: 101
Employee Name: John Doe
Employee Salary: $55000.0
```

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.

<u>MyPoint.java</u>

```
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;

}

// Setter method to set both x and y

public void setXY(int x, int y) {

    this.x = x;

    this.y = y;

}

// Getter method to return x and y in a 2-element int array

public int[] getXY() {

    int[] coordinates = {x, y};

    return coordinates;

}

// Calculate the 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);

}

// Calculate the distance from this point to another MyPoint instance (another)

public double distance(MyPoint another) {

    int xDiff = this.x - another.x;

    int yDiff = this.y - another.y;

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

```
  }
  // Calculate the distance from this point to the origin (0,0)
  public double distance() {
      return Math.sqrt(x * x + y * y);
  }
  // toString method to return a string description of the instance in the format "(x,y)"
 //Override
  public String toString() {
      return "(" + x + "," + y + ")";
  }
}
```

TestMyPoint.java

```
public class TestMyPoint {
    public static void main(String[] args) {
        MyPoint point1 = new MyPoint(3, 4);
        MyPoint point2 = new MyPoint(6, 8);
        // Testing setXY and getXY methods
        point1.setXY(5, 7);
        int[] coordinates = point1.getXY();
        System.out.println("Point 1 coordinates: (" + coordinates[0] + "," + coordinates[1] + ")");
        // Testing toString method
        System.out.println("Point 2 coordinates: " + point2.toString());
        // Testing distance methods
        double distance1 = point1.distance(0, 0);
        double distance2 = point1.distance(point2);

        double distance3 = point2.distance();

        System.out.println("Distance from Point 1 to (0,0): " + distance1);

        System.out.println("Distance from Point 1 to Point 2: " + distance2);

        System.out.println("Distance from Point 2 to the origin: " + distance3);

    }

}
```

Output:

```
F:\Shammi\Java>javac MyPoint.java

F:\Shammi\Java>javac TestMyPoint.java

F:\Shammi\Java>java TestMyPoint

Point 1 coordinates: (5,7)

Point 2 coordinates: (6,8)

Distance from Point 1 to (0,0): 8.602325267042627

Distance from Point 1 to Point 2: 1.4142135623730951

Distance from Point 2 to the origin: 10.0
```

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.

Program:

```
class Shape {

    public void draw() {

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

    }

    public void erase() {

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

    }

}

class Circle extends Shape {

   //Override

    public void draw() {

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

    }

   //Override

    public void erase() {

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

    }

}

class Triangle extends Shape {

   //Override
```

```
    public void draw() {
        System.out.println("Drawing a triangle.");
    }
   //Override
    public void erase() {
        System.out.println("Erasing a triangle.");
    }
}
class Square extends Shape {
   //Override
    public void draw() {
        System.out.println("Drawing a square.");
    }
   //Override
    public void erase() {
        System.out.println("Erasing a square.");
    }
}
public class ShapeDemo {
    public static void main(String[] args) {
        Shape[] shapes = new Shape[3];
        shapes[0] = new Circle();
        shapes[1] = new Triangle();
        shapes[2] = new Square();
        for (Shape shape : shapes) {
            shape.draw();
            shape.erase();
            System.out.println();
        }
    }
}
```

Output:

```
F:\Shammi\Java>java ShapeDemo
Drawing a circle.
Erasing a circle.

Drawing a triangle.
Erasing a triangle.

Drawing a square.
Erasing a square.
```

Program 6: Develop a java program to create an abstract class Shape with abstract methods calculateArea() and calculatePerimeter(). Create subclasses Circle and Trianngle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.

ShapeDemoCT.java

```
// Abstract Shape class

abstract class Shape {

    // Abstract method to calculate the area (must be implemented by subclasses)

    public abstract double calculateArea();

    // Abstract method to calculate the perimeter (must be implemented by subclasses)

    public abstract double calculatePerimeter();

}

// Subclass Circle

class Circle extends Shape {

    private double radius;

    public Circle(double radius) {

        this.radius = radius;

    }

   //Override

    public double calculateArea() {

        return Math.PI * radius * radius;

    }

   //Override

    public double calculatePerimeter() {

        return 2 * Math.PI * radius;
    }
}
// Subclass Triangle
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;
    }
   //Override
    public double calculateArea() {
        // Using Heron's formula to calculate the area of a triangle
        double s = (side1 + side2 + side3) / 2.0;
        return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
    }
   //Override
    public double calculatePerimeter() {
        return side1 + side2 + side3;
    }
}
public class ShapeDemoCT {
    public static void main(String[] args) {
        // Create a Circle and calculate its area and perimeter
        Circle circle = new Circle(5.0);
        System.out.println("Circle - Area: " + circle.calculateArea());
        System.out.println("Circle - Perimeter: " + circle.calculatePerimeter());
        // Create a Triangle and calculate its area and perimeter
        Triangle triangle = new Triangle(3.0, 4.0, 5.0);
        System.out.println("Triangle - Area: " + triangle.calculateArea());
        System.out.println("Triangle - Perimeter: " + triangle.calculatePerimeter());
    }
}
```

Output:

```
F:\Shammi\Java>java ShapeDemoCT
Circle - Area: 78.53981633974483
Circle - Perimeter: 31.41592653589793
Triangle - Area: 6.0
Triangle - 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.

Program:

```
// Resizable interface

interface Resizable {

    void resizeWidth(int width);

    void resizeHeight(int height);

}

// Rectangle class implementing Resizable

class Rectangle implements Resizable {

    private int width;

    private int height;

    public Rectangle(int width, int height) {

        this.width = width;

        this.height = height;

    }

    public int getWidth() {

        return width;

    }

    public int getHeight() {

        return height;

    }

   //Override
```

```
    public void resizeWidth(int width) {
        if (width > 0) {
            this.width = width;
            System.out.println("Resized width to " + width);
        } else {
            System.out.println("Invalid width. Width remains unchanged.");
        }
    }
   //Override
    public void resizeHeight(int height) {
        if (height > 0) {
            this.height = height;
            System.out.println("Resized height to " + height);
        } else {
            System.out.println("Invalid height. Height remains unchanged.");
        }
    }
   //Override
    public String toString() {
        return "Rectangle (Width: " + width + ", Height: " + height + ")";
    }
}
public class ResizeDemo {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(5, 7);

        System.out.println("Original Rectangle: " + rectangle);

        // Resize the rectangle

        rectangle.resizeWidth(8);

        rectangle.resizeHeight(10);

        System.out.println("Resized Rectangle: " + rectangle);

    }

}
```

Output:

```
F:\Shammi\Java>java ResizeDemo

Original Rectangle: Rectangle (Width: 5, Height: 7)

Resized width to 8

Resized height to 10

Resized Rectangle: Rectangle (Width: 8, Height: 10)
```

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.

Program:

```
// Outer class
class Outer {
    // Outer class method
    void display() {
        System.out.println("Outer class display method.");
    }
  // Inner class
    class Inner {
        // Inner class method
        void display() {
            System.out.println("Inner class display method.");
        }
    }
}
public class Main {
    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 within the Outer class)

        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.
```

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

Program:

```
// Custom exception class for DivisionByZero

class DivisionByZeroException extends Exception {

    public DivisionByZeroException(String message) {

        super(message);

    }

}

public class CustomExceptionDemo {

    public static void main(String[] args) {

        int numerator = 10;

        int denominator = 0; // Attempting to divide by zero

        try {

            int result = divide(numerator, denominator);

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

        } catch (DivisionByZeroException e) {

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

        } finally {

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

        }

    }

    // Method to perform division and throw custom exception if denominator is zero

    public static int divide(int numerator, int denominator) throws DivisionByZeroException {
        if (denominator == 0) {
            throw new DivisionByZeroException("Division by zero is not allowed.");
        }
        return numerator / denominator;
    }
}
```

Output:

```
java CustomExceptionDemo
Custom Exception: Division by zero is not allowed.
Finally block executed.
```

Program 10: Develop a java program to create a package named mypack and import and implement it in a suitable class.

To create a package named ‘mypack’and import and implement it in a suitable class, follow the steps:

1. Create the Package:

Create a directory named ‘mypack’ in your project's directory structure. Inside this directory, you can place your Java source files. Make sure that the ‘mypack’directory contains a ‘mypack’subdirectory that will hold your Java source files. The structure should look like this:

```
your_project_directory/
├── mypack/
│   └── MyClass.java
└── OtherProjectFiles...
```

2. Create a Java Class in the Package:

Inside the ‘mypack’directory, create a Java class. For example, `MyClass.java`. Here's a simple example of `MyClass`:

```
package mypack; // Package declaration

public class MyClass {

    public void displayMessage() {

        System.out.println("Hello from mypack.MyClass!");

    }

}
```

3. Create a Class to Import and Use the Package:

Create another Java class in a different directory (not within the ‘mypack’package). In this class, import and use the `mypack.MyClass` class from the ‘mypack’package:

```
import mypack.MyClass; // Import the class from the package
```

```
public class MainClass {

    public static void main(String[] args) {

        MyClass myObj = new MyClass(); // Create an instance of MyClass

        myObj.displayMessage(); // Call the method from MyClass

    }

}
```

4. Compile and Run the Program:

Compile both `MyClass.java` and `MainClass.java` using the `javac` command. Make sure you are in the directory containing the ‘mypack’directory when compiling:

```
javac mypack/MyClass.java

javac MainClass.java
```

Then, run the program using the `java` command:

```
java MainClass
```

This will execute the `main` method in `MainClass`, which imports and uses the `mypack.MyClass` class from the ‘mypack’package. The message "Hello from mypack.MyClass!" printed to the console.

Program 11: Write a java 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).

Program:

```
class MyRunnable implements Runnable {
    private String threadName;
    public MyRunnable(String name) {
        this.threadName = name;
    }
   //Override
    public void run() {
        try {
            System.out.println(threadName + " is starting.");
            Thread.sleep(500); // Sleep for 500 milliseconds (0.5 seconds)
            System.out.println(threadName + " is done.");
        } catch (InterruptedException e) {
            System.err.println(threadName + " was interrupted.");
        }
    }
}
public class RunnableThreadDemo {
    public static void main(String[] args) {
        // Create and start multiple threads using the MyRunnable class
        Thread thread1 = new Thread(new MyRunnable("Thread 1"));
        Thread thread2 = new Thread(new MyRunnable("Thread 2"));

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

        thread1.start();

        thread2.start();

        thread3.start();

    }

}
```

Output:

```
Thread 1 is starting.

Thread 2 is starting.

Thread 3 is starting.

Thread 1 is done.

Thread 2 is done.

Thread 3 is done.
```

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.

Program:

```
class MyThread extends Thread {

    public MyThread(String name) {

        super(name); // Call the Thread class constructor with the given name

        start(); // Start the thread

    }

   //Override

    public void run() {

        // Code to be executed in the child thread

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

            System.out.println("Child Thread: " + getName() + " - Count: " + i);

            try {

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

            } catch (InterruptedException e) {

                System.err.println("Child thread interrupted.");

            }

        }

    }

}

public class ThreadDemo {

    public static void main(String[] args) {
        // Create an instance of MyThread and start it

        MyThread myThread = new MyThread("MyThread");

        // Code to be executed in the main thread

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

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

            try {

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

            } catch (InterruptedException e) {

                System.err.println("Main thread interrupted.");

            }

        }

    }

}
```

Output:

```
Main Thread - Count: 1

Child Thread: MyThread - Count: 1

Child Thread: MyThread - Count: 2

Main Thread - Count: 2

Child Thread: MyThread - Count: 3

Main Thread - Count: 3

Child Thread: MyThread - Count: 4

Main Thread - Count: 4

Child Thread: MyThread - Count: 5

Main Thread - Count: 5
```

Program 13: Develop a java program that demonstrates the usage of autoboxing and unboxing.

Program:

```
public class AutoboxingUnboxingDemo {

   public static void main(String[] args) {

      // Autoboxing: Converting primitive types to their corresponding wrapper classes

      Integer num1 = 42; // int to Integer

      Double num2 = 3.14; // double to Double

      Boolean flag = true; // boolean to Boolean

      // Unboxing: Converting wrapper classes back to primitive types

      int intValue = num1; // Integer to int

      double doubleValue = num2; // Double to double

      boolean booleanValue = flag; // Boolean to boolean

      // Performing operations with autoboxed values

      System.out.println("Sum: " + (num1 + num2));

      // Using autoboxing with collections (e.g., ArrayList)

      java.util.ArrayList<Integer> numbers = new java.util.ArrayList<>();

      numbers.add(10);

      numbers.add(20);

      numbers.add(30);

      // Iterating through the list and unboxing the values

      for (Integer number : numbers) {

         int value = number; // Unboxing

         System.out.println("Value: " + value);
```

```
        }
    }
}
```

Output:

```
Sum: 45.14
Value: 10
Value: 20
Value: 30
```

Program 14: Develop a Java program that demonstrates the usage of primitive types and automatic type promotion in expressions.

Program:

```
public class TypePromotionDemo {

    public static void main(String[] args) {

        // Primitive data types

        int intValue = 10;

        long longValue = 20L;

        float floatValue = 3.14f;

        double doubleValue = 2.71828;

        // Automatic type promotion in expressions

        double result1 = intValue + longValue; // int is promoted to long, and the result is a long

        float result2 = floatValue + longValue; // float is promoted to long, and the result is a long

        double result3 = floatValue + doubleValue; // float is promoted to double, and the result is a double

        // Display the results

        System.out.println("Result 1: " + result1);

        System.out.println("Result 2: " + result2);

        System.out.println("Result 3: " + result3);

    }

}
```

Output:

```
Result 1: 30.0

Result 2: 23.14

Result 3: 5.858279999999999
```

Viva Question and Answer

1. List any five features of Java?
   Some features include Object Oriented, Platform Independent, Robust, Interpreted and Multi-threaded.

2. Why is Java Architectural Neutral?

   Its compiler generates an architecture-neutral object file format, which makes the compiled code to be executable on many processors, with the presence of Java runtime system.

3. What is Java Virtual Machine and how it is considered in context of Java's platform independent feature?

   When Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.

4. What do you mean by Object?

   Object is a runtime entity and it's state is stored in fields and behavior is shown via methods. Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication.

5. Define class?

   A class is a blue print from which individual objects are created. A class can contain fields and methods to describe the behavior of an object.

6. What kind of variables a class can consist of?
   A class consist of Local variable, instance variables and class variables.

7. What is a Local Variable?

   Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and it will be destroyed when the method has completed.

8. What is a Instance Variable?

   Instance variables are variables within a class but outside any method. These variables are instantiated when the class is loaded.

9. What is a Class Variable?

   These are variables declared with in a class, outside any method, with the static keyword. Class variables are also known as static variables.

10. What do you mean by Constructor?

    Constructor gets invoked when a new object is created. Every class has a constructor. If we do not explicitly write a constructor for a class the java compiler builds a default constructor for thatclass.

11. What is the default value of float and double datatype in Java?
    Default value of float and double datatype in different as compared to C/C++. For float its 0.0f and for double it is 0.0d

12. When parseInt() method can be used?
    This method is used to get the primitive data type of a certain String.

13. Why main method is static?
    Because object is not required to call static method if It were non-static method, JVM creates object first then call main() method that will lead to the problem of extra memory allocation.

14. What is the use of bufferedReader class?
    BufferedReader class can be used to read data line by line by readLine() method.

15. What is Array?
    An array is a container object that holds a fixed number of values of a single type. Thelength of an array is established when the array is created.

16. Define Inheritance?
    It is the process where one object acquires the properties of another. With the use of inheritance, the information is made manageable in a hierarchic al order.

17. When super keyword is used?
    If the method overrides one of its superclass's methods, overridden method can be invokedthrough the use of the keyword super. It can be also used to refer to a hidden field.

18. What is Polymorphism?
    Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object

19. What is Abstraction?
    It refers to the ability to make a class abstract in OOP. It helps to reduce the complexity and also improves the maintainability of the system.

20. What is Abstract class?
    These classes cannot be instantiated and are either partially implemented or not at all implemented. This class contains one or more abstract methods which are simply method declarations without a body.

21. When Abstract methods are used?

    If you want a class to contain a particular method but you want the actual implementation of that method to be determined by child classes, you can declare the method in the parent class as abstract.

22. What is Encapsulation?
    It is the technique of making the fields in a class private and providing access to the fields via public methods. If a field is declared private, it cannot be accessed by anyone outside the class, thereby hiding the fields within the class. Therefore, encapsulation is also referred to as data hiding.

23. What is an Interface?
    An interface is a collection of abstract methods. A class implements an interface, therebyinheriting the abstract methods of the interface.

24. Explain the use of subclass in a Java program?
    Subclass inherits all the public and protected methods and the implementation. It also inherits all the default modifier methods and their implementation.

25. What is String Tokenizer in Java?
    It is a pre-defined class in **java.util** package can be used to split the given string into tokens (parts) based on delimiters (any special symbols or spaces).

26. Explain access modifier
    The access modifiers in java specifies accessibility (scope) of a data member, method, constructor or class. There are 4 types of java access modifiers: private, default, protected and public.

27. What are the ways to read data from the keyboard?
    InputStreamReader, Console, Scanner, DataInputStreametc.

28. What is the meaning of import in Java?
    The **import** statement in **Java** allows to refer to classes which are declared in otherpackages to be accessed without referring to the full package name.

29. What is the use of **extend** Keyword.
    Extend is the keyword used to inherit the properties of a class.

30. What is an Exception?
    An exception is a problem that arises during the execution of a program. Exceptions are caughtby handlers positioned along the thread's method invocation stack.

31. What do you mean by Checked Exceptions?
    It is an exception that is typically a user error or a problem that cannot be foreseen by the programmer. For example, if a file is to be opened, but the file cannot be found, an exception occurs. These exceptions cannot simply be ignored at the time of compilation.

32. Explain Runtime Exceptions?
    It is an exception that occurs that probably could have been avoided by the programmer. As opposed to checked exceptions, runtime exceptions are ignored at the time of compilation.

33. Which are the two subclasses under Exception class?
    The Exception class has two main subclasses: IOException class and RuntimeException Class.

34. When throws keyword is used?
    If a method does not handle a checked exception, the method must declare it using the throws keyword. The throws keyword appears at the end of a method's signature.

35. How finally used under Exception Handling?
    The finally keyword is used to create a block of code that follows a try block. A finally block of code always executes, whether or not an exception has occurred.

36. What do you mean by Multithreaded program?
    A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread, and each thread defines a separate path of execution.

37. What are the two ways in which Thread can be created?
    Thread can be created by: implementing Runnable interface, extending the Thread class.

38. How does multi-threading take place on a computer with a single CPU?
    The operating system's task scheduler allocates execution time to multiple tasks. By quickly switching between executing tasks, it creates the impression that tasks execute sequentially.

39. What invokes a thread's run() method?
    After a thread is started, via its start() method of the Thread class, the JVM invokes the thread's run() method when the thread is initially executed.