BANGALORE INSTITUTE OF TECHNOLOGY

K.R. ROAD, V.V PURAM, BANGALORE – 560 004

(AFFILIATED TO VTU, BELAGAVI)

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

COURSE CODE: BCS306A

OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY

As per Choice Based Credit System Scheme (CBCS)

FOR III SEMESTER ISE AS PRESCRIBED BY VTU

Academic Year 2023-24

Prepared By:

Priya N V

Assistant Professor

Dept. of ISE, BIT

BANGALORE INSTITUTE OF TECHNOLOGY

VISION:

Establish and develop the Institute as the Centre of higher learning, ever abreast with expanding horizon of knowledge in the field of Engineering and Technology with entrepreneurial thinking, leadership excellence for life-long success and solve societal problems.

MISSION:

- Provide high quality education in the Engineering disciplines from the undergraduate through doctoral levels with creative academic and professional programs.
- Develop the Institute as a leader in Science, Engineering, Technology, Management and Research and apply knowledge for the benefit of society.
- Establish mutual beneficial partnerships with Industry, Alumni, Local, State and Central Governments by Public Service Assistance and Collaborative Research.
- Inculcate personality development through sports, cultural and extracurricular activities and engage in social, economic and professional challenges.

Bangalore Institute of Technology

K. R. Road, V. V. Pura, Bengaluru- 560004

Department of Information Science and Engineering

VISION:

Empower every student to be innovative, creative and productive in the field of Information Technology by imparting quality technical education, developing skills and inculcating human values.

MISSION:

M1	To evolve continually as a Centre of Excellence in offering quality Information Technology **Education**.
M2	To nurture the students to meet the global competency in industry for **Employment**.
M3	To promote collaboration with industry and academia for constructive interaction to empower **Entrepreneurship.**
M4	To provide reliable, contemporary and integrated technology to support and facilitate **Teaching, Learning, Research and Service.**

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

PEO-1	Uplift the students through Information Technology **Education.**
PEO-2	Provide exposure to emerging technologies and train them to **Employable** in multi-disciplinary industries.
PEO-3	Motivate them to become good professional Engineers and **Entrepreneur.**
PEO-4	Inspire them to prepare for **Higher Learning and Research.**

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO-1	To provide our graduates **with Core Competence in Information Processing and Management.**
PSO-2	To provide our graduates with Higher Learning in **Computing Skills.**

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **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.
4. **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.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **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.
7. **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.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **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.
11. **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 and leader in a team, to manage projects and in multidisciplinary environments.
12. **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.

Bangalore Institute of Technology

K. R. Road, V.V. Pura, Bengaluru 560004

Department of Information Science and Engineering

OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY (BCS306A)

PRE-REQUISITES:

Basic C/C++ programming concepts.

COURSE LEARNING OBJECTIVES (CLO)

This laboratory course enables students to get practical experience in design, develop, implement, analyze and evaluation/testing of

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 programing and exceptions.

COURSE OUTCOMES (CO)

On the completion of this laboratory course, the students will be able to:

CO 1: Demonstrate in writing simple programs involving branching, looping structures and a class involving data members and methods for the given scenario.

CO 2: Apply the concepts of inheritance and interfaces in solving real world problems.

CO 3: Analyze the concept of packages, exception handling, multithreading, autoboxing and enumerations in program development.

CO 4: Develop a java program in solvin real world problems.

		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
BCS306A	CO1	2											1
	CO2	2	2										1
	CO3	2	2										1
	CO4	2	2										1

		PSO1	PSO2
BCS306A	CO1	2	
	CO2	2	2
	CO3	2	2
	CO4	2	2

OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY

Subject Code: BCS306A **CIE Marks: 50**
Hours/Week: 0:0:2:0 **Exam Hours: 03**
Total Hours: 24

List of Programs

Sl. No.	Name of Experiment (Part-A)
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 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.

OBJECT ORIENTED PROGRAMMING WITH JAVA LABORATORY

Subject Code: BCS358D **CIE Marks: 50**
Hours/Week: 0:0:2:0 **Total Hours: 24**

Lesson Planning / Schedule of Experiments

Sl. No	Name of Experiment	WEEK
1	Sample Programs	Week1
2	Develop a JAVA program to add TWO matrices of suitable order N (The value of N should be read from command line arguments).	Week2
3	Develop a stack class to hold a maximum of 10 integers with suitable methods Develop a JAVA main method to illustrate Stack operations	Week3
4	A class called Employee, which models an employee with an ID, 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.	Week4
5	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.	Week5
6	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.	Week6

7	**LAB TEST1**	**Week7**
8	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.	**Week8**
9	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	**Week9**
10	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.	**Week10**
11	Develop a JAVA program to raise a custom exception (user defined exception) for DivisionByZero using try, catch, throw and finally.	**Week11**
12	Develop a JAVA program to create a package named mypack and import & implement it in a suitable class.	**Week12**
13	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).	**Week13**
14	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.	**Week14**
15	**LAB TEST 2**	**Week15**

Conduction of Practical Examination:

1. All laboratory experiments are to be included for practical examination.
2. Students are allowed to pick one experiment from Part A with lot.
3. Marks distribution: **Procedure (15%) + Execution (70%) + Viva (15%)**

   Part A: (15+70+15=100)

   Final Exam Marks:_____/50 Marks (100 scaled down to 50)

4. Change of experiment is allowed only once and marks allotted to the write up part to be made Zero

INSTRUCTIONS TO THE CANDIDATES

1. Students should come with thorough preparation for the experiment to be conducted.
2. Students will not be permitted to attend the laboratory unless they bring the practical record fully completed in all respects pertaining to the experiment conducted in the previous class.
3. Practical record should be neatly maintained.
4. They should obtain the signatures of the staff in-charge in the observation book after completing each experiment.
5. Theory regarding each experiment should be written in the practical record before procedure in your own words.
6. Ask lab technician for assistance if you have any problem.
7. Save your class work, assignments in system.
8. Do not download or install software without the assistance of the laboratory technician.
9. Do not alter the configuration of the system.
10. Turn off the systems after use.

PROGRAM-1

Add two matrices of suitable order N [The value of N should be read from command line argument]

```
package Programs;
import java.util.Scanner;
public class MatrixA {

	public static void main (String[] args)
	{
		// TODO Auto-generated method stub

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

		int i,j;

		int[ ][ ] matrix1 = new int[n][n];
		int[ ][ ] matrix2 = new int[n][n];
		int[ ][ ] sum = new int[n][n];
		Scanner sc=new Scanner(System.in);

		// Initialize matrices with some values, for example, i+j
		System.out.println("Enter the elements in the matrix1:");

		for ( i = 0; i < n; i++)
		{
			for (j = 0; j < n; j++)
			{
				matrix1[i][j] = sc.nextInt();


				}
			}

		System.out.println("Enter the elements in the matrix2:");

		for (i = 0; i < n; i++)
		{
			for (j = 0; j < n; j++)
			{
				matrix2[i][j] = sc.nextInt();


				}
			}
```

```
                    // Add the matrices
                    for (i = 0; i < n; i++)
                    {
                            for (j = 0; j < n; j++)
                      {
                                    sum[i][j] = matrix1[i][j] + matrix2[i][j];

                      }
                    }

                    // Print the result
                    System.out.println("Sum of matrices is: ");
                    for ( i = 0; i < n; i++)
                    {
                            for ( j = 0; j < n; j++)
                            {
                                    System.out.print(" " +sum[i][j] );
                            }

                    System.out.println();
                    }
            }
    }
```

OUTPUT:

```
Enter the elements in the matrix1:
1 2
3 4
Enter the elements in the matrix2:
1 5
4 8
Sum of matrices is:
 2 7
 7 12
```

PROGRAM-2

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

```
package Programs;
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 sc = new Scanner(System.in);
         System.out.print("Enter your Choice: ");
        int choice = sc.nextInt();

          switch(choice)
          {
            case 1:  System.out.println("Enter Number to push: ");
                    int num = sc.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
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: 3
Stack: 10 20
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: 3
Stack: 10
Stack Operations
1. Push
2. Pop
3. Display
4. Exit
Enter your Choice: 4
```

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.

```
package Programs;
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. STHIRA", 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: Dr. STHIRA
Salary: Rs.50000.00
Enter the percentage of salary to raise:
25

Employee information after raise:
Employee ID: 1
Name: Dr. STHIRA
Salary: Rs.62500.00

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.

```
package Programs;

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(1,2);//9 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 (1,2): " + point1.distance(1,2));
        System.out.println("Distance between Point 1 and Point 2: " + point1.distance(point2));
        System.out.println("Distance from Point 2 to origin: " + point1.distance());
    }
}
```

OUTPUT:

Point 1: (0, 0)
Point 2: (3, 4)
Point 1 after setXY (): (1, 2)
Point 2 coordinates: (3, 4)
Distance between Point 1 and (1,2): 0.0
Distance between Point 1 and Point 2: 2.8284271247461903
Distance from Point 2 to origin: 2.23606797749979

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.

```
package Programs;


//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");
 }
}
```

```
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-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.

```
package Programs;


// 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

```
package Programs;

// 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.

```
package Programs;

// 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 Excep {
    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.");
        }
    }
}
```

OUTPUT:

```
Exception caught: Division by zero is not allowed.
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!");
    }
}


package Programs; // Package declaration

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

public class Main {
    public static void main(String[] args) {
        MyPackageClass myPackageObj = new MyPackageClass();
        myPackageObj.display(); // Calling the display method from MyPackageClass
    }
}
```

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 14: 2
Thread 15: 2
Thread 14: 3
Thread 15: 3
Thread 14: 4
Thread 15: 4
Thread 14: 5
Thread 15: 5
```

PROGRAM-11

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.

```
package Programs;


// 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(1000); // 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

VIVA QUESTIONS

1. What is The Java Features ?
   1.Compiled and Interpreted 2.Platform-Independent and Portable 3. Object -Oriented 4.Robust and Secure 5.Distributed 6.Simple, Small and Familiar 7.Multithreaded and interactive 8.High Performance 9.Dynamic and Extensible

2. How Java Differs From C ?
   Java does not include the C unique statement keywords goto, size of, and type def. 2.Java does not contain the data type struct,union and enum. 3.Java does not define the type modifiers keywords auto,extern register,signed,and unsigned. 4.Java does not support an explicit pointer type.

3. How Java Differs From C ++ ?
   java does not support operator overloading. 2. Java does not have template classes as in C++.
   3.Java does not multiple inheritance of classes.This is accomplished using a new feature called "interface".

4. What is The Java Components?
   1.Object and Classes 2.Data Abstraction and Encapsulation 3.Inheritance 4.Polymorephism 5.Dynamic Binding 6.Message Communication.

5. What is the Variables ?
   A variable is an identifier that denotes a storage location used to store a data value.

6. What are Class?
   Class is a template for multiple objects with similar features and it is a blue print for objects. It defines a type of object according to the data the object can hold and the operations the object can perform.

7. What are Primitive data types?
   Primitive data types are 8 types and they are: byte, short, int, long, float, double, boolean, char.

8. What is the Constructor?
   Constructor is a special member function of class call automatically when object is created.

9. What is method overloading?
   Function with same name but different argument perform different task known as method overloading.

10. What is parameterized constructor?
    The constructors that can take argument are called parameterized constructors

11. What is OOPs?

Object oriented programming organizes a program around its data, i. e. , objects and a set of well defined interfaces to that data. An object-oriented program can be characterized as data controlling access to code.

12. What are Encapsulation?

Encapsulation is the mechanism that binds together code and data it manipulates and keeps both safe from outside interference and misuse

13. What are Polymorphism

Polymorphism is the feature that allows one interface to be used for general class actions.

14. What is an Object and how do you allocate memory to it?

Object is an instance of a class and it is a software unit that combines a structured set of data with a set of operations for inspecting and manipulating that data. When an object is created using new operator, memory is allocated to it

15. What is the difference between constructor and method

Constructor will be automatically invoked when an object is created whereas method has to be called explicitly.

16. What are methods and how are they defined?

Methods are functions that operate on instances of classes in which they are defined.