GOVERNMENT OF KERALA

DEPARTMENT OF TECHNICAL EDUCATION

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(GOVT. ENGINEERING COLLEGE)

KOTTAYAM - 686501

RECORD BOOK

GOVERNMENT OF KERALA

DEPARTMENT OF TECHNICAL EDUCATION

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(GOVT. ENGINEERING COLLEGE)

KOTTAYAM - 686501

20MCA132

OBJECT ORIENTED PROGRAMMING LAB

Name: MINNA JOBY

Branch: Master of Computer Applications

Semester: 2

Roll No: 35

CERTIFIED BONAFIDE RECORD WORK DONE BY

Reg No. KTE24MCA-2038

STAFF IN CHARGE

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Contents

Experiment No: 1 **Date: 11/02/2025**

Even-Odd Classification

Aim

Write a Java program to check whether an input number is even or odd.

Algorithm

1. Take an integer as input from the user.
2. Use an if-else statement to check if the number is even or odd.
3. Print the result accordingly.

Source Code

```
import java.util.Scanner;
public class EvenOdd {
        public static void main ( String [] args ) {
                Scanner scanner = new Scanner ( System . in ) ;
                System . out . print (" Enter a number : ") ;
                int number = scanner . nextInt () ;
                if ( number % 2 == 0) {
                        System . out . println ( number + " is even .")
    ;
                } else {
                        System . out . println ( number + " is odd .")
   ;
                }
        }
}
```

Result

The program was executed successfully.

```
Enter a Number: 34
34 is even
```

Experiment No: 2 **Date: 11/02/2025**

Sum Of First n Natural Numbers

Aim

Write a Java program to compute the sum of the first n natural numbers.

Algorithm

1. Take an integer n as input from the user.
2. Use either a for loop or a while loop to compute the sum.
3. Print the result.

Source Code

```
import java.util.Scanner;
public class SumOfNumbers {
        public static void main ( String [] args ) {
                Scanner scanner = new Scanner (System.in);
                System.out.print(" Enter a number n: ");
                int n = scanner.nextInt () ;
                int sum = 0;
                for (int i = 1; i <= n ; i ++) {
                        sum += i ;
                }
                System.out.println ("The sum of the first " + n + "
   numbers is: " + sum ) ;
        }
}
```

Result

The program was executed successfully.

```
Enter a number n: 10
The sum of the first 10 numbers is: 55
```

Experiment No: 3 **Date: 11/02/2025**

Factorial of a Number

Aim

Write a Java program to compute the factorial of a given number.

Algorithm

1. Take an integer as input from the user.
2. Compute the factorial using either a for loop or a while loop.
3. Print the result.

Source Code

```
import java . util . Scanner ;
public class Factorial {
        public static void main ( String [] args ) {
                Scanner scanner = new Scanner ( System . in ) ;
                System . out . print (" Enter a number : ") ;
                int number = scanner . nextInt () ;
                int factorial = 1;
                int i = 1;
                while ( i <= number ) {
                        factorial *= i ;
                        i ++;
                }
                System . out . println ("The factorial of " + number +
   " is:" + factorial ) ;
        }
}
```

Result

The program was executed successfully.

```
Enter a number : 5
The factorial of 5 is:120
```

Experiment No: 4 **Date: 11/02/2025**

Assigning Grades Based on Numeric Score

Aim

Write a Java program that assigns a grade based on a numeric score.

Algorithm

1. Take a numeric score (0-100) as input from the user.
2. Use either an if-else if-else structure or a switch-case statement to assign a grade:
   - 90-100 → A
   - 80-89 → B
   - 70-79 → C
   - 60-69 → D
   - Below 60 → F
3. Print the assigned grade.

Source Code

```
import java . util . Scanner ;
public class GradeClassification {
        public static void main ( String [] args ) {
                Scanner scanner = new Scanner ( System . in ) ;
                System . out . print (" Enter the score : ") ;
                int score = scanner . nextInt () ;
                char grade ;
                switch ( score / 10) {
                        case 10:
                        case 9:
                                grade = 'A';
                                break ;
                        case 8:
                                grade = 'B';
                                break ;
                        case 7:
                                grade = 'C';
                                break ;
                        case 6:
                                grade = 'D';
                                break ;
                        default :
                                grade = 'F';
                                break ;
                }
                System . out . println (" Your grade is: " + grade ) ;
}}
```

Result

The program was executed successfully.

```
Enter the score : 89
Your grade is: B
```

Experiment No: 5 **Date: 18/02/2025**

Find Product with Lowest Price

Aim

Create an object-oriented Java program to find product with lowest price.

Algorithm

1. Start
2. Define a class Product with attributes pcode,pname,price.
3. Create a function findLowest to compare product prices and return the lowest.
4. Read details of three products from the user.
5. Call findLowest and display the product with lowest price.
6. Stop

Source Code

```
import java.util.Scanner;

class Product
{
    String pcode, pname;
    double price;

    Product(String pcode, String pname, double price)
    {
        this.pcode = pcode;
        this.pname = pname;
        this.price = price;
    }

    static Product findLowest(Product[] products)
    {
        Product lowest = products[0];
        for (Product p : products)
        {
            if (p.price < lowest.price)
            {
                lowest = p;
            }
        }
        return lowest;
    }

    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        Product[] products = new Product[3];

        for (int i = 0; i < 3; i++)
```

```
        {
            System.out.println("Enter details for product " + (i + 1) +
    ":");
            System.out.print("Pcode: ");
            String pcode = sc.next();
            System.out.print("Pname: ");
            String pname = sc.next();
            System.out.print("Price: ");
            double price = sc.nextDouble();
            products[i] = new Product(pcode, pname, price);
        }
        Product lowest = findLowest(products);
        System.out.println("\nProduct with Lowest Price:");
        System.out.println("Pcode: " + lowest.pcode + ", Pname: " +
    lowest.pname + ", Price: " + lowest.price);
    }
}
```

Result

The program was executed successfully.

```
Enter details for product 1:
Pcode: 101
Pname: chair
Price: 300
Enter details for product 2:
Pcode: 102
Pname: table
Price: 500
Enter details for product 3:
Pcode: 103
Pname: fan
Price: 200

Product with Lowest Price:
Pcode: 103, Pname: fan, Price: 200.0
```

Experiment No: 6 **Date: 18/02/2025**

Complex Number Operations

Aim

Create an object-oriented Java program to perform addition and multiplication of complex numbers,with inputs provided by the user.

Algorithm

1. Start
2. Define a class Complex with attributes real and img.
3. Implement methods add and multiply to perform operations on complex numbers.
4. Read two complex numbers from the user.
5. Compute their sum and product using respective methods.
6. Display the results.
7. Stop

Source Code

```
import java.util.Scanner;

class Complex
{
    double real, imag;

    Complex(double real, double imag)
    {
        this.real = real;
        this.imag = imag;
    }

    Complex add(Complex c)
    {
        return new Complex(this.real + c.real, this.imag + c.imag);
    }

    Complex multiply(Complex c)
    {
        double realPart = (this.real * c.real)- (this.imag * c.imag);
        double imagPart = (this.real * c.imag) + (this.imag * c.real);
        return new Complex(realPart, imagPart);
    }

    public String toString()
    {
        return real + " + " + imag + "i";
    }

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

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

        System.out.print("Enter real and imaginary part of first
    complex number: ");
        Complex c1 = new Complex(sc.nextDouble(), sc.nextDouble());

        System.out.print("Enter real and imaginary part of second
    complex number: ");
        Complex c2 = new Complex(sc.nextDouble(), sc.nextDouble());

        Complex sum = c1.add(c2);
        Complex product = c1.multiply(c2);

        System.out.println("Sum: " + sum);
        System.out.println("Product: " + product);
    }
}
```

Result

The program was executed successfully.

```
Enter real and imaginary part of first complex number: 5
3
Enter real and imaginary part of second complex number: 6
8
Sum: 11.0 + 11.0i
Product: 6.0 + 58.0i
```

Experiment No: 7 **Date: 18/02/2025**

Matrix Addition

Aim

Create an object-oriented Java program to perform matrix addition.

Algorithm

1. Start
2. Read the number of rows and columns of the matrices.
3. Read elements of first matrix.
4. Read elements of second matrix.
5. Perform element wise addition to obtain the sum matrix.
6. Display the sum matrix.
7. Stop

Source Code

```
import java . util . Scanner ;

class MatrixAddition
{
    public static void main ( String [] args )
    {
        Scanner sc = new Scanner ( System . in ) ;

        System . out . print (" Enter number of rows and columns : ") ;
        int rows = sc . nextInt () ;
        int cols = sc . nextInt () ;

        int [][] matrix1 = new int[ rows ][ cols ];
        int [][] matrix2 = new int[ rows ][ cols ];
        int [][] sumMatrix = new int[ rows ][ cols ];

        System . out . println (" Enter elements of first matrix :") ;
        for (int i = 0; i < rows ; i ++)
            for (int j = 0; j < cols ; j ++)
                matrix1 [ i ][ j ] = sc . nextInt () ;

        System . out . println (" Enter elements of second matrix :") ;
        for (int i = 0; i < rows ; i ++)
            for (int j = 0; j < cols ; j ++)
                matrix2 [ i ][ j ] = sc . nextInt () ;

        for (int i = 0; i < rows ; i ++)
            for (int j = 0; j < cols ; j ++)
                sumMatrix [ i ][ j ] = matrix1 [ i ][ j ] + matrix2 [ i
    ][ j ];

        System . out . println ("Sum of matrices :") ;
        for (int i = 0; i < rows ; i ++) {
```

```
            for (int j = 0; j < cols ; j ++)
                System . out . print ( sumMatrix [ i ][ j ] + " ") ;
            System . out . println () ;
        }
    }
}
```

Result

The program was executed successfully.

```
Enter number of rows and columns: 2
2
Enter elements of first matrix:
1 2
3 1
Enter elements of second matrix:
2 4
1 3
Sum of matrices:
3 6
4 4
```

Experiment No: 8 **Date: 25/02/2025**

Employee Search Using an Array of Objects

Aim

Write a Java program to store employee details including employee number, name, and salary, and search for an employee by employee number.

Algorithm

1. Start
2. Input number of employees.
3. For each employee:
   - Read employee details (number, name, salary).
   - Add employee to the list.
4. Input employee number to search.
5. Search the list for the employee number.
   - If found, display details.
   - If not, display "not found."
6. Stop

Source Code

```
import java.util.ArrayList;
import java.util.Scanner;

class Employee {
    int employeeNumber;
    String name;
    double salary;

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

    public void displayEmployeeDetails() {
        System.out.println("Employee Number: " + employeeNumber);
        System.out.println("Employee Name: " + name);
        System.out.println("Employee Salary: " + salary);
    }
}

public class EmployeeSearch {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int numberOfEmployees = 3;
        ArrayList<Employee> employeeList = new ArrayList<>();

        for (int i = 0; i < numberOfEmployees; i++) {
```

```
            System.out.println("Enter details for employee " + (i + 1));

            System.out.print("Enter Employee Number: ");
            int employeeNumber = scanner.nextInt();
            scanner.nextLine();

            System.out.print("Enter Employee Name: ");
            String name = scanner.nextLine();

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

            Employee newEmployee = new Employee(employeeNumber, name, salary);
            employeeList.add(newEmployee);
        }

        System.out.print("\nEnter Employee Number to search: ");
        int searchNumber = scanner.nextInt();

        boolean found = false;
        for (Employee employee : employeeList) {
            if (employee.employeeNumber == searchNumber) {
                employee.displayEmployeeDetails();
                found = true;
                break;
            }
        }

        if (!found) {
            System.out.println("Employee with Employee Number " + searchNumber + " not found.");
        }

        scanner.close();
    }
}
```

Result

The program was executed successfully.

```
Enter details for employee 1
Enter Employee Number: 1
Enter Employee Name: anjaly
Enter Employee Salary: 2000
Enter details for employee 2
Enter Employee Number: 2
Enter Employee Name: anamika
Enter Employee Salary: 3000
Enter details for employee 3
Enter Employee Number: 3
Enter Employee Name: shine
```

```
Enter Employee Salary: 1000

Enter Employee Number to search: 2
Employee Number: 2
Employee Name: anamika
Employee Salary: 3000.0
```

Experiment No: 9 **Date: 25/02/2025**

String Search in an Array

Aim

Write a Java program to store ‘n‘ strings in an array. Search for a given string. If found,print its index; otherwise, display ”String not found.”

Algorithm

1. Start
2. Input the number of strings (’n’).
3. Store ’n’ strings in an array.
4. Input the string to search for.
5. Search the array for the string:
   - If found, print the index and stop.
   - If not found, print "String not found."
6. Stop

Source Code

```java
import java.util.Scanner;

public class StringSearch {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of strings to store: ");
        int n = scanner.nextInt();
        scanner.nextLine();

        String[] strings = new String[n];

        System.out.println("Enter " + n + " strings:");
        for (int i = 0; i < n; i++) {
            strings[i] = scanner.nextLine();
        }
        System.out.print("\nEnter the string to search for: ");
        String searchString = scanner.nextLine();
        boolean found = false;
        for (int i = 0; i < n; i++) {
            if (strings[i].equals(searchString)) {
                System.out.println("String found at index: " + i);
                found = true;
                break;
            }
        }
        if (!found) {
            System.out.println("String not found.");
        }
        scanner.close();
    }
}
```

Result

The program was executed successfully.

```
Enter the number of strings to store: 2
Enter 2 strings:
hello
world

Enter the string to search for: hello
String found at index: 0
```

Experiment No: 10 **Date: 25/02/2025**

String Manipulations

Aim

Write a Java program to perform various string manipulations, including finding the length, converting to uppercase and lowercase, extracting characters and substrings, and reversing the string.

Algorithm

1. Start
2. Input a string from the user.
3. Print the length of the string.
4. Convert and print the string in uppercase.
5. Convert and print the string in lowercase.
6. If the string length is greater than 3:
   - Print the character at position 3.
   - Otherwise, print that the string is too short.
7. If the string length is greater than 5:
   - Print the substring from index 2 to 5.
   - Otherwise, print that the string is too short.
8. Reverse and print the string.
9. Stop

Source Code

```
import java.util.Scanner;

public class StringManipulations {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string: ");
        String inputString = scanner.nextLine();

        int length = inputString.length();
        System.out.println("Length of the string: " + length);

        String upperCaseString = inputString.toUpperCase();
        System.out.println("String in uppercase: " + upperCaseString);

        String lowerCaseString = inputString.toLowerCase();
        System.out.println("String in lowercase: " + lowerCaseString);

        if (length > 3) {
            char charAtPosition = inputString.charAt(3);
            System.out.println("Character at position 3: " +
   charAtPosition);
        } else {
```

```
23              System.out.println("String is too short to extract
    character at position 3.");
24          }
25
26          if (length > 5) {
27              String substring = inputString.substring(2, 5);
28              System.out.println("Substring from index 2 to 5: " +
    substring);
29          } else {
30              System.out.println("String is too short to extract a
    substring from index 2 to 5.");
31          }
32
33          String reversedString = new StringBuilder(inputString).reverse
    ().toString();
34          System.out.println("Reversed string: " + reversedString);
35
36          scanner.close();
37      }
38  }
```

Result

The program was executed successfully.

```
Enter a string: hello world
Length of the string: 11
String in uppercase: HELLO WORLD
String in lowercase: hello world
Character at position 3: l
Substring from index 2 to 5: llo
Reversed string: dlrow olleh
```

Experiment No: 11 **Date: 25/02/2025**

Inheritance in Java

Aim

Write a Java program to implement hierarchical inheritance for a book management system. Define a base class 'Publisher', a derived class 'Book', and two subclasses 'Literature' and 'Fiction'. Include methods to read and display book details and demonstrate the functionality using user input.

Algorithm

1. Start
2. Input publisher details.
3. Input literature book details.
4. Input fiction book details.
5. Display the details of the literature book.
6. Display the details of the fiction book.
7. Stop

Source Code

```
import java.util.Scanner;

class Publisher {
    String publisherName;
    String publisherAddress;

    public void inputPublisherDetails(Scanner scanner) {
        System.out.print("Enter Publisher Name: ");
        publisherName = scanner.nextLine();

        System.out.print("Enter Publisher Address: ");
        publisherAddress = scanner.nextLine();
    }

    public void displayPublisherDetails() {
        System.out.println("Publisher Name: " + publisherName);
        System.out.println("Publisher Address: " + publisherAddress);
    }
}

class Book extends Publisher {
    String bookTitle;
    String authorName;
    double price;

    public void inputBookDetails(Scanner scanner) {
        System.out.print("Enter Book Title: ");
        bookTitle = scanner.nextLine();

        System.out.print("Enter Author Name: ");
```

```
        authorName = scanner.nextLine();

        System.out.print("Enter Book Price: ");
        price = scanner.nextDouble();
        scanner.nextLine();
    }

    public void displayBookDetails() {
        displayPublisherDetails();
        System.out.println("Book Title: " + bookTitle);
        System.out.println("Author Name: " + authorName);
        System.out.println("Book Price: " + price);
    }
}

class Literature extends Book {
    String genre;

    public void inputLiteratureDetails(Scanner scanner) {
        inputBookDetails(scanner);
        System.out.print("Enter Genre (e.g., Poetry, Drama): ");
        genre = scanner.nextLine();
    }

    public void displayLiteratureDetails() {
        displayBookDetails();
        System.out.println("Literature Genre: " + genre);
    }
}

class Fiction extends Book {
    String subGenre;

    public void inputFictionDetails(Scanner scanner) {
        inputBookDetails(scanner);
        System.out.print("Enter Fiction Subgenre (e.g., Mystery,
   Thriller): ");
        subGenre = scanner.nextLine();
    }

    public void displayFictionDetails() {
        displayBookDetails();
        System.out.println("Fiction Subgenre: " + subGenre);
    }
}

public class BookManagementSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        Literature literatureBook = new Literature();
        Fiction fictionBook = new Fiction();

        System.out.println("Enter details for Literature Book:");
        literatureBook.inputPublisherDetails(scanner);
        literatureBook.inputLiteratureDetails(scanner);

        System.out.println("\nEnter details for Fiction Book:");
```

```
88          fictionBook.inputPublisherDetails(scanner);
89          fictionBook.inputFictionDetails(scanner);
90
91          System.out.println("\n--- Literature Book Details ---");
92          literatureBook.displayLiteratureDetails();
93
94          System.out.println("\n--- Fiction Book Details ---");
95          fictionBook.displayFictionDetails();
96
97          scanner.close();
98      }
99  }
```

Result

The program was executed successfully.

```
Enter details for Literature Book:
Enter Publisher Name: anjaly
Enter Publisher Address: India
Enter Book Title: Hello world
Enter Author Name: anamika
Enter Book Price: 2000
Enter Genre (e.g., Poetry, Drama): poetry

Enter details for Fiction Book:
Enter Publisher Name: meril
Enter Publisher Address: usa
Enter Book Title: Little things
Enter Author Name: meril
Enter Book Price: 3000
Enter Fiction Subgenre (e.g., Mystery, Thriller): Mystery

--- Literature Book Details ---
Publisher Name: anjaly
Publisher Address: India
Book Title: Hello world
Author Name: anamika
Book Price: 2000.0
Literature Genre: poetry

--- Fiction Book Details ---
Publisher Name: meril
Publisher Address: usa
Book Title: Little things
Author Name: meril
Book Price: 3000.0
Fiction Subgenre: Mystery
```

Experiment No: 12 **Date: 04/03/2025**

Calculate Area and Perimeter Using Interfaces

Aim

Write a Java Program to create an interface having prototypes of functions 'area()' and 'perimeter()'. Create two classes 'Circle' and 'Rectangle' which implement the above interface. Develop a menu-driven program to find the area and perimeter of these shapes.

Algorithm

1. Start
2. Create an interface Shape with methods area() and perimeter().
3. Implement Shape in Circle and Rectangle classes.
   - Circle: Calculate area as pi × radius² and perimeter as 2×pi×radius.
   - Rectangle: Calculate area as length × width and perimeter as 2×(length+width).
4. In main():
   - Create a Scanner for user input.
   - Use a do-while loop to display a menu:
     1. Circle: Ask for radius, compute area and perimeter.
     2. Rectangle: Ask for length and width, compute area and perimeter.
     3. Exit the program.
   - Validate user choice and handle invalid inputs.
5. Repeat until the user selects Exit.
6. Stop

Source Code

```java
import java.util.Scanner;

interface Shape {
  double area();
  double perimeter();
}

class Circle implements Shape {
  double radius;

  Circle(double radius) {
    this.radius = radius;
  }
  public double area() {
    return Math.PI * radius * radius;
  }
  public double perimeter() {
    return 2 * Math.PI * radius;
  }
}
```

```
21
22 class Rectangle implements Shape {
23   double length, width;
24   Rectangle(double length, double width) {
25     this.length = length;
26     this.width = width;
27   }
28
29 public double area() {
30   return length * width;
31 }
32
33 public double perimeter() {
34   return 2 * (length + width);
35 }
36 }
37 public class AreaPerimeterCalculator {
38   public static void main(String[] args) {
39     Scanner scanner = new Scanner(System.in);
40     int choice;
41     do {
42       System.out.println("\n1. Circle\n2. Rectangle\n3. Exit");
43       System.out.print("Enter your choice: ");
44       choice = scanner.nextInt();
45       switch (choice) {
46         case 1:
47           System.out.print("Enter radius: ");
48           double r = scanner.nextDouble();
49           Circle circle = new Circle(r);
50           System.out.println("Area: " + circle.area());
51           System.out.println("Perimeter: " + circle.perimeter());
52           break;
53         case 2:
54           System.out.print("Enter length: ");
55           double l = scanner.nextDouble();
56           System.out.print("Enter width: ");
57           double w = scanner.nextDouble();
58           Rectangle rectangle = new Rectangle(l, w);
59           System.out.println("Area: " + rectangle.area());
60           System.out.println("Perimeter: " + rectangle.perimeter());
61           break;
62         case 3:
63           System.out.println("Exiting...");
64           break;
65         default:
66           System.out.println("Invalid choice!");
67       }
68     } while (choice != 3);
69   }
70 }
```

Result

The program was executed successfully.

```
1. Circle
2. Rectangle
3. Exit
Enter your choice: 1
Enter radius: 2
Area: 12.566370614359172
Perimeter: 12.566370614359172

1. Circle
2. Rectangle
3. Exit
Enter your choice: 2
Enter length: 3
Enter width: 4
Area: 12.0
Perimeter: 14.0

1. Circle
2. Rectangle
3. Exit
Enter your choice: 3
Exiting...
```

Experiment No: 13 **Date: 04/03/2025**

Program to Manage Employee Collection

Aim

Create a Java program to manage a collection of employees in a company. Implement an abstract class Employee with fields name (String) and salary (double), and an abstract method calculateSalary(). Create two subclasses: Manager (with a bonus field) and Developer (with an experience field), both overriding calculateSalary() to calculate the total salary. Implement an interface Benefits with a method calculateBenefits(), where Manager provides a fixed insurance benefit and Developer provides an allowance based on experience. Use polymorphism to store Employee objects in a list and display employee details and salary. Add method overloading in Manager for project assignment, where one method takes just a project name and the other takes both the project name and the number of team members.

Algorithm

1. Start
2. Create an abstract class Employee with attributes name, salary, and method calculateSalary().
3. Create an interface Benefits with method calculateBenefits().
4. Implement Manager and Developer classes:
   Manager: Includes bonus and assignProject() method (overloaded).
   - Developer: Includes experience and salary based on experience.
5. In main():
   - Create an ArrayList<Employee>.
   - Get the number of employees.
   - For each employee:
     1.Input name, salary, and type (Manager/Developer).
     2.If Manager, input bonus, create object, and add to list.
     3.If Developer, input experience, create object, and add to list.
6. Display details for all employees, including benefits.
7. Stop

Source Code

```
import java.util.ArrayList;
import java.util.Scanner;

abstract class Employee {
    String name;
    double salary;

    Employee(String name, double salary) {
        this.name = name;
```

```
10          this.salary = salary;
11      }
12
13      abstract double calculateSalary();
14
15      void displayDetails() {
16          System.out.println("\nName: " + name);
17          System.out.println("Salary: " + calculateSalary());
18      }
19  }
20
21  interface Benefits {
22      double calculateBenefits();
23  }
24
25  class Manager extends Employee implements Benefits {
26      double bonus;
27
28      Manager(String name, double salary, double bonus) {
29          super(name, salary);
30          this.bonus = bonus;
31      }
32
33      double calculateSalary() {
34          return salary + bonus;
35      }
36
37      public double calculateBenefits() {
38          return 5000;
39      }
40
41      void assignProject(String projectName) {
42          System.out.println(name + " assigned to project: " +
    projectName);
43      }
44
45      void assignProject(String projectName, int teamSize) {
46          System.out.println(name + " assigned to project: " +
    projectName + " with team size: " + teamSize);
47      }
48  }
49
50  class Developer extends Employee implements Benefits {
51      int experience;
52
53      Developer(String name, double salary, int experience) {
54          super(name, salary);
55          this.experience = experience;
56      }
57
58      double calculateSalary() {
59          return salary + (experience * 1000);
60      }
61
62      public double calculateBenefits() {
63          return experience * 500;
64      }
65  }
```

```
66
67 public class EmployeeManagement {
68     public static void main(String[] args) {
69         ArrayList<Employee> employees = new ArrayList<>();
70         Scanner scanner = new Scanner(System.in);
71
72         System.out.print("Enter number of employees: ");
73         int numEmployees = scanner.nextInt();
74         scanner.nextLine();
75
76         for (int i = 0; i < numEmployees; i++) {
77             System.out.println("\nEnter details for Employee " + (i +
    1) + ":");
78             System.out.print("Enter name: ");
79             String name = scanner.nextLine();
80
81             System.out.print("Enter salary: ");
82             double salary = scanner.nextDouble();
83             scanner.nextLine();
84
85             System.out.print("Enter type (Manager/Developer): ");
86             String type = scanner.nextLine();
87
88             if (type.equalsIgnoreCase("Manager")) {
89                 System.out.print("Enter bonus for Manager: ");
90                 double bonus = scanner.nextDouble();
91                 scanner.nextLine();
92                 employees.add(new Manager(name, salary, bonus));
93             } else if (type.equalsIgnoreCase("Developer")) {
94                 System.out.print("Enter experience for Developer: ");
95                 int experience = scanner.nextInt();
96                 scanner.nextLine();
97                 employees.add(new Developer(name, salary, experience));
98             }
99         }
100
101         System.out.println("\nEmployee Details:");
102         for (Employee emp : employees) {
103             emp.displayDetails();
104             if (emp instanceof Benefits) {
105                 System.out.println("Benefits: " + ((Benefits) emp).
    calculateBenefits());
106             }
107         }
108
109         scanner.close();
110     }
111 }
```

Result

The program was executed successfully.

```
Enter number of employees: 2

Enter details for Employee 1:
Enter name: anjali
Enter salary: 2000
Enter type (Manager/Developer): Manager
Enter bonus for Manager: 1000

Enter details for Employee 2:
Enter name: anamika
Enter salary: 3000
Enter type (Manager/Developer): Developer
Enter experience for Developer: 2

Employee Details:

Name: anjali
Salary: 3000.0
Benefits: 5000.0

Name: anamika
Salary: 5000.0
Benefits: 1000.0
```

Experiment No: 14 **Date: 04/03/2025**

Graphics Package for Geometric Figures

Aim

Create a Graphics package that contains classes and interfaces for geometric figures such as 'Rectangle', 'Triangle', 'Square', and 'Circle'. Test the package by finding the area of these figures.

Algorithm

1. Start
2. Define an Interface GeometricFigure:
   - Declare an abstract method calculateArea() that returns a double.
3. Create Rectangle Class:
   - Implement GeometricFigure
   - Store width and height
   - Implement calculateArea() to return width * height.
4. Create Triangle Class:
   - Implement GeometricFigure
   - Store base and height
   - Implement calculateArea() to return 0.5 * base * height.
5. Create Square Class:
   - Inherit from Rectangle.
   - Constructor takes sideLength and calls Rectangle constructor with equal width and height.
6. Create Circle Class:
   - Implement GeometricFigure.
   - Store radius.
   - Implement calculateArea() to return pi * radius^2.
7. Create TestGraphics Class:
   - Instantiate Rectangle, Triangle, Square, and Circle.
   - Print area of each figure using calculateArea().
8. Stop

Source Code

GeometricFigure.java

```java
package graphics;

public interface GeometricFigure {
    double calculateArea();
}
```

Rectangle.java

```java
package graphics;

public class Rectangle implements GeometricFigure {
    private double width;
    private double height;

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

    @Override
    public double calculateArea() {
        return width * height;
    }
}
```

Triangle.java

```java
package graphics;

public class Triangle implements GeometricFigure {
    private double base;
    private double height;

    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    @Override
    public double calculateArea() {
        return 0.5 * base * height;
    }
}
```

Square.java

```java
package graphics;

public class Square extends Rectangle {

    public Square(double sideLength) {
        super(sideLength, sideLength);
    }
}
```

Circle.java

```java
package graphics;

public class Circle implements GeometricFigure {
    private double radius;

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

```
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
}
```

TestGraphics.java

```
package graphics;

public class TestGraphics {
    public static void main(String[] args) {
        GeometricFigure rectangle = new Rectangle(5, 10);
        GeometricFigure triangle = new Triangle(6, 8);
        GeometricFigure square = new Square(4);
        GeometricFigure circle = new Circle(7);

        System.out.println("Area of Rectangle: " + rectangle.
    calculateArea());
        System.out.println("Area of Triangle: " + triangle.
    calculateArea());
        System.out.println("Area of Square: " + square.calculateArea())
    ;
        System.out.println("Area of Circle: " + circle.calculateArea())
    ;
    }
}
```

Result

The program was executed successfully.

```
Area of Rectangle: 50.0
Area of Triangle: 24.0
Area of Square: 16.0
Area of Circle: 153.93804002589985
```

Experiment No: 15 **Date: 18/03/2025**

File Operations in Java

Aim

Write a program that performs various file operations such as reading, writing, and appending data to a file.

Algorithm

1. Start
2. Display menu (Write, Append, Read, Exit)
3. Get user choice
4. Perform action:
   Write → Overwrite file with input data
   Append → Add input data to file
   Read → Display file content
   Exit → Terminate program
5. Repeat until exit
6. Stop

Source Code

```
import java.io.*;
import java.util.Scanner;

public class FileOperations {
    private static final String FILE_NAME = "sample.txt";

    public static void writeToFile(String data) {
        try (FileWriter writer = new FileWriter(FILE_NAME)) {
            writer.write(data);
            System.out.println("Data written to " + FILE_NAME);
        } catch (IOException e) {
            System.out.println("Error writing to file: " + e.getMessage
    ());
        }
    }

    public static void appendToFile(String data) {
        try (FileWriter writer = new FileWriter(FILE_NAME, true)) {
            writer.write(data);
            System.out.println("Data appended to " + FILE_NAME);
        } catch (IOException e) {
            System.out.println("Error appending to file: " + e.
    getMessage());
        }
    }

    public static void readFromFile() {
        try (BufferedReader reader = new BufferedReader(new FileReader(
    FILE_NAME))) {
```

```
            String line;
            System.out.println("Data read from " + FILE_NAME + ":");
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        } catch (FileNotFoundException e) {
            System.out.println("Error: " + FILE_NAME + " not found.");
        } catch (IOException e) {
            System.out.println("Error reading from file: " + e.
    getMessage());
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\nFile Operations Menu:");
            System.out.println("1. Write to file");
            System.out.println("2. Append to file");
            System.out.println("3. Read from file");
            System.out.println("4. Exit");
            System.out.print("Enter your choice (1-4): ");

            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter data to write: ");
                    String writeData = scanner.nextLine();
                    writeToFile(writeData);
                    break;
                case 2:
                    System.out.print("Enter data to append: ");
                    String appendData = scanner.nextLine();
                    appendToFile(appendData);
                    break;
                case 3:
                    readFromFile();
                    break;
                case 4:
                    System.out.println("Exiting program.");
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid choice. Please enter a
    number between 1 and 4.");
            }
        }
    }
}
```

Result

The program was executed successfully.

```
File Operations Menu:
1. Write to file
2. Append to file
3. Read from file
4. Exit
Enter your choice (1-4): 1
Enter data to write: hello world
Data written to sample.txt

File Operations Menu:
1. Write to file
2. Append to file
3. Read from file
4. Exit
Enter your choice (1-4): 2
Enter data to append: This is me
Data appended to sample.txt

File Operations Menu:
1. Write to file
2. Append to file
3. Read from file
4. Exit
Enter your choice (1-4): 3
Data read from sample.txt:
hello worldThis is me

File Operations Menu:
1. Write to file
2. Append to file
3. Read from file
4. Exit
Enter your choice (1-4): 4
Exiting program.
```

Experiment No: 16 **Date: 18/03/2025**

System-Defined and User-Defined Exception for Authentication

Aim

Write a Java program that demonstrates both system-defined exceptions (such as FileNot- FoundException and IOException) and user-defined exceptions for authentication fail- ures. Implement a readFile(String filename) method that attempts to read a file and prints its contents while handling FileNotFoundException if the file does not exist and IOException for other input/output errors. Define a custom exception class AuthenticationException that extends Exception and create an authenticate(String username, String password) method to validate user credentials against predefined values (e.g., "admin" with password "admin123"), throwing an AuthenticationException if authentication fails. In the main method, prompt the user to enter a filename, attempt to read the file, then request login credentials, invoking authenticate() and handling exceptions using try-catch blocks to display appropriate error messages, ensuring meaningful feedback to the user.

Algorithm

1. Start
2. Read filename, attempt to open file
   Handle ‘FileNotFoundException‘ & ‘IOException‘
3. Read username & password, authenticate
   Throw ‘AuthenticationException‘ if invalid
4. Print success or error message
5. Stop

Source Code

```
import java.io.*;
import java.util.Scanner;

class AuthenticationException extends Exception {
    public AuthenticationException(String message) {
        super(message);
    }
}

public class ExceptionHandlingDemo {

    public static void readFile(String filename) {
        try (BufferedReader reader = new BufferedReader(new FileReader(
   filename))) {
            String line;
            System.out.println("File Contents:");
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        } catch (FileNotFoundException e) {
```

```
            System.out.println("Error: File not found - " + filename);
        } catch (IOException e) {
            System.out.println("Error reading the file: " + e.
    getMessage());
        }
    }

    public static void authenticate(String username, String password)
    throws AuthenticationException {
        String validUsername = "admin";
        String validPassword = "admin123";

        if (!username.equals(validUsername) || !password.equals(
    validPassword)) {
            throw new AuthenticationException("Invalid username or
    password!");
        }
        System.out.println("Authentication successful! Welcome, " +
    username + "!");
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter filename to read: ");
        String filename = scanner.nextLine();
        readFile(filename);

        System.out.print("\nEnter username: ");
        String username = scanner.nextLine();
        System.out.print("Enter password: ");
        String password = scanner.nextLine();

        try {
            authenticate(username, password);
        } catch (AuthenticationException e) {
            System.out.println("Authentication Failed: " + e.getMessage
    ());
        }
        scanner.close();
    }
}
```

Result

The program was executed successfully.

```
Enter filename to read: sample.txt
File Contents:
hello worldThis is me

Enter username: admin
Enter password: admin123
Authentication successful! Welcome, admin!
```

Experiment No: 17 **Date: 18/03/2025**

Multithreading

Aim

Write a Java program that defines two classes: one for generating and displaying the mul- tiplication table of 5 and another for printing the first N prime numbers. Implement both classes using multithreading, demonstrating both approaches—by extending the Thread class and implementing the Runnable interface. Ensure proper thread management and synchronization if needed.

Algorithm

1. Start
2. Create a thread for the multiplication table (extends ‘Thread‘)
3. Create a thread for prime numbers (implements ‘Runnable‘)
4. Start both threads
5. Stop

Source Code

```
class MultiplicationTable extends Thread {
    public void run() {
        System.out.println("Multiplication Table of 5:");
        for (int i = 1; i <= 10; i++) {
            System.out.println("5 x " + i + " = " + (5 * i));
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                System.out.println("Thread interrupted: " + e.
   getMessage());
            }
        }
    }
}

class PrimeNumbers implements Runnable {
    private int n;

    public PrimeNumbers(int n) {
        this.n = n;
    }

    public void run() {
        System.out.println("First " + n + " Prime Numbers:");
        int count = 0, num = 2;
        while (count < n) {
            if (isPrime(num)) {
                System.out.print(num + " ");
                count++;
            }
            num++;
```

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

    private boolean isPrime(int num) {
        if (num < 2) return false;
        for (int i = 2; i <= Math.sqrt(num); i++) {
            if (num % i == 0) return false;
        }
        return true;
    }
}

public class MultithreadingDemo {
    public static void main(String[] args) {
        MultiplicationTable tableThread = new MultiplicationTable();
        PrimeNumbers primeRunnable = new PrimeNumbers(10);
        Thread primeThread = new Thread(primeRunnable);

        tableThread.start();
        primeThread.start();
    }
}
```

Result

The program was executed successfully.

```
Multiplication Table of 5:
First 10 Prime Numbers:
2 3 5 7 11 13 17 19 23 29
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```