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: SHREYAS S

Branch: Master of Computer Applications

Semester: 2

Roll No: 49

CERTIFIED BONAFIDE RECORD WORK DONE BY

Reg No. .....

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

```java
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 Number");
        } else {
            System.out.println(number + " is odd Number");
        }
    }
}
```

Result

The program was executed successfully.

```
Enter a number : 6
6 is Even Number
Enter a number : 13
13 is odd Number
```

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

```java
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  : ");
        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  : 8
The sum of the first 8 numbers is : 36
```

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

```java
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 to find its Factorial : ");
        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 to find its Factorial : 6
The factorial of 6 is : 720
```

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

```java
import java.util.Scanner;
public class Grades {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println(" 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 : 75
Your grade is : C
```

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

Find Product with Lowest Price

Aim

Write a Java program to define a class Product with data members pcode, pname, and price. Find and display the product with the lowest price.

Algorithm

1. Start
2. Input the number of products (n).
3. Declare an array items[] of size n to store Product objects.
4. For each product from 0 to n-1:
   4.1 Input the product code (pcode).
   4.2 Input the product name (pname).
   4.3 Input the product price (price).
   4.4 Create a Product object and store it in items[i].
5. Find the product with the lowest price:
   5.1 Initialize lowest as items[0].
   5.2 For each product from 1 to n-1:
       If items[i].price is less than lowest.price,
       update lowest = items[i].
6. Display the product details of lowest.
7. Stop

Source Code

```
import java.util.Scanner;
public class Product{
        String pcode,pname;
        double price;
        public Product(String pcode, String pname, double price){
                this.pcode=pcode;
                this.pname=pname;
                this.price=price;
        }
        public static Product getLowestPrice(Product items[]){
                Product lowest=items[0];
                for(int i=1;i<items.length;i++){
                        if(items[i].price<lowest.price)
                        lowest=items[i];
                }
                return lowest;
        }
        public void display(){
                System.out.println("Product code: "+this.pcode+"\
   nProduct name: "+this.pname+"\nPrice: "+price);
        }
        public static void main(String args[]){
```

```
                Scanner s=new Scanner(System.in);
                System.out.println("Enter number of products: ");
                int n=s.nextInt();
                Product items[]=new Product[n];
                for(int i=0;i<n;i++){
                        System.out.println("Enter product code of
                        product "+i+": ");
                        String pcode=s.next();
                        System.out.println("Enter product name of
                        product "+i+": ");
                        String pname=s.next();
                        System.out.println("Enter price of product
                        "+i+": ");
                        double price=s.nextDouble();
                        items[i]= new Product(pcode,pname,price);
                }
                Product lowest=Product.getLowestPrice(items);
                System.out.println("Item with lowest Price : ");
                lowest.display();
        }
}
```

Result

The program was executed successfully.

```
Enter number of products:
3
Enter product code of product 0:
p01
Enter product name of product 0:
Laptop
Enter price of product 0:
45000
Enter product code of product 1:
p02
Enter product name of product 1:
Smartphone
Enter price of product 1:
14000
Enter product code of product 2:
p02
Enter product name of product 2:
Tablet
Enter price of product 2:
28000

Item with lowest Price :
Product code: p02
Product name: Smartphone
Price: 14000.0
```

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

Complex Number Operations

Aim

Write a Java program to perform addition and multiplication of complex numbers, with inputs provided by the user.

Algorithm

1. Start
2. Input the real and imaginary parts of the first complex number (real1, imaginary1).
3. Input the real and imaginary parts of the second complex number (real2,imaginary2).
4. Create two Complex objects, c1 and c2, using the input values.
5. Perform addition of complex numbers:
   5.1 Compute realPart = c1.real + c2.real.
   5.2 Compute imaginaryPart = c1.imaginary + c2.imaginary.
   5.3 Store the result in a new Complex object (additionResult).
6. Perform multiplication of complex numbers:
   6.1 Compute realPart = c1.real * c2.real - c1.imaginary * c2.imaginary.
   6.2 Compute imaginaryPart = c1.real * c2.imaginary + c1.imaginary * c2.real.c.
   6.3 Store the result in a new Complex object (multiplicationResult).
7. Display the result of the addition.
8. Display the result of the multiplication.
9. Stop

Source Code

```
import java.util.Scanner;

class Complex {
    double real, imaginary;

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

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

    Complex multiply(Complex c) {
        double realPart = this.real * c.real - this.imaginary * c.
        imaginary;

```

```
        double imaginaryPart = this.real * c.imaginary + this.imaginary
        * c.real;
        return new Complex(realPart, imaginaryPart);
    }

    void display() {
        if (imaginary < 0) {
            System.out.println(real +" - "+ Math.abs(imaginary) +"i");
        } else {
            System.out.println(real + " + " + imaginary + "i");
        }
    }
}
public class ComplexNumberOperations {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the real and imaginary parts of the
    first complex number:");
        System.out.print("Real: ");
        double real1 = sc.nextDouble();
        System.out.print("Imaginary: ");
        double imaginary1 = sc.nextDouble();

        System.out.println("Enter the real and imaginary parts of the
    second complex number:");
        System.out.print("Real: ");
        double real2 = sc.nextDouble();
        System.out.print("Imaginary: ");
        double imaginary2 = sc.nextDouble();

        Complex c1 = new Complex(real1, imaginary1);
        Complex c2 = new Complex(real2, imaginary2);

        Complex additionResult = c1.add(c2);
        System.out.print("Addition of the two complex numbers: ");
        additionResult.display();

        Complex multiplicationResult = c1.multiply(c2);
        System.out.print("Multiplication of the two complex numbers:");
        multiplicationResult.display();
    }
}
```

Result

The program was executed successfully.

```
Enter the real and imaginary parts of the first complex number:
Real: 3
Imaginary: 2
Enter the real and imaginary parts of the second complex number:
Real: 1
Imaginary: 4
Addition of the two complex numbers: 4.0 + 6.0i
Multiplication of the two complex numbers: -5.0 + 14.0i
```

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

Matrix Addition

Aim

Write a Java program to perform matrix addition.

Algorithm

1. Start
2. Input the number of rows (rows) and columns (cols).
3. Declare three 2D arrays: matrix1, matrix2, and sumMatrix of size [rows][cols].
4. Read the elements of matrix1 from the user.
5. Read the elements of matrix2 from the user.
6. Perform matrix addition:
   For each row i from 0 to rows - 1
    For each column j from 0 to cols - 1
     sumMatrix[i][j] = matrix1[i][j] + matrix2[i][j]
7. Display the resulting sumMatrix.
8. Stop

Source Code

```java
import java.util.Scanner;
public class MatrixAddition {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of rows: ");
        int rows = scanner.nextInt();
        System.out.print("Enter the number of columns: ");
        int cols = scanner.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] = scanner.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] = scanner.nextInt();
            }
        }
```

```
28
29          for (int i = 0; i < rows; i++) {
30              for (int j = 0; j < cols; j++) {
31                  sumMatrix[i][j] = matrix1[i][j] + matrix2[i][j];
32              }
33          }
34
35          System.out.println("Sum of matrices:");
36          for (int i = 0; i < rows; i++) {
37              for (int j = 0; j < cols; j++) {
38                  System.out.print(sumMatrix[i][j] + " ");
39              }
40              System.out.println();
41          }
42          scanner.close();
43      }
44  }
```

Result

The program was executed successfully.

```
Enter the number of rows: 3
Enter the number of columns: 3
Enter elements of first matrix:
2 3 6
1 7 4
5 9 1

Enter elements of second matrix:
1 7 8
3 9 2
5 4 6

Sum of matrices:
3 10 14
4 16 6
10 13 7
```

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. Create a list to store employee details.
3. Input the number of employees (‘n‘).
4. Repeat for ‘n‘ employees:
   4.1 Input Employee Number, Name, and Salary.
   4.2 Store the details in the list.
5. Input the Employee Number to search (‘empNumberToSearch‘).
6. Search for the employee in the list
   6.1 If found, display employee details.
   6.2 If not found, print "Employee not found."
7. Stop

Source Code

```java
import java.util.ArrayList;
import java.util.Scanner;
class Employee {
  int empNumber;
  String empName;
  double empSalary;
  Employee(int empNumber, String empName, double empSalary) {
    this.empNumber = empNumber;
    this.empName = empName;
    this.empSalary = empSalary;
  }
  void displayEmployeeDetails() {
    System.out.println("Employee Number: " + empNumber);
    System.out.println("Employee Name: " + empName);
    System.out.println("Employee Salary: " + empSalary);
  }
}
public class EmployeeDetails {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    ArrayList<Employee> employeeList = new ArrayList<>();
    System.out.print("Enter the number of employees: ");
    int numberOfEmployees = scanner.nextInt();
    scanner.nextLine();
    for (int i = 0; i < numberOfEmployees; i++) {
      System.out.println("\nEnter details for employee " + (i + 1));
```

```
27        System.out.print("Enter employee number: ");
28        int empNumber = scanner.nextInt();
29        scanner.nextLine();
30        System.out.print("Enter employee name: ");
31        String empName = scanner.nextLine();
32        System.out.print("Enter employee salary: ");
33        double empSalary = scanner.nextDouble();
34        scanner.nextLine();
35        employeeList.add(new Employee(empNumber, empName, empSalary));
36      }
37      System.out.print("\nEnter employee number to search: ");
38      int empNumberToSearch = scanner.nextInt();
39      boolean found = false;
40      for (Employee emp : employeeList) {
41        if (emp.empNumber == empNumberToSearch) {
42          emp.displayEmployeeDetails();
43          found = true;
44          break;
45        }
46      }
47      if (!found) {
48        System.out.println( "Employee not found with
49        employee number: " + empNumberToSearch);
50      }
51      scanner.close();
52    }
53  }
```

Result

The program was executed successfully.

```
Enter the number of employees: 3

Enter details for employee 1
Enter employee number: 121
Enter employee name: Amar
Enter the Salary of the Employee: 25000

Enter details for employee 2
Enter employee number: 122
Enter employee name: Akbar
Enter the Salary of the Employee: 27000

Enter details for employee 3
Enter employee number: 123
Enter employee name: Antony
Enter the Salary of the Employee: 26000
```

```
Enter Employee number to search: 122
Employee Number: 122
Employee Name: Akbar
Employee Salary: 27000.0


Enter Employee number to search: 145
Employee not found with employee number: 145
```

Experiment No: 9 **Date: 18/02/2025**

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. Create an array of size ‘n‘ to store the strings.
4. Repeat for ‘i = 0‘ to ‘n-1‘:
   4.1 Input string and store it in the array.
5. Input the string to search (‘searchString‘).
6. Initialize a flag ‘found = false‘.
7. Search for ‘searchString‘ in the array:
   7.1 If found, print the index and set
   ‘found = true‘, then exit loop.
8. If ‘found == false‘, print "String not found."
9. 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 you want to store:");
    int n = scanner.nextInt();
    scanner.nextLine();
    String[] strings = new String[n];
    System.out.println("Enter the strings:");
    for (int i = 0; i < n; i++) {
      System.out.print("String " + (i + 1) + ": ");
      strings[i] = scanner.nextLine();
    }

    System.out.print("\nEnter the string to search: ");
    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;
      }
    }

```

```
27      if (!found) {
28        System.out.println("String not found.");
29      }
30      scanner.close();
31    }
32  }
```

Result

The program was executed successfully.

```
Enter the number of strings you want to store: 4
Enter the strings:
String 1: Java
String 2: Python
String 3: C
String 4: Rust

Enter the string to search: Rust
String found at index: 3

Enter the string to search: Ruby
String not found.
```

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. Find and display the length of the string.
4. Convert and display the string in uppercase.
5. Convert and display the string in lowercase.
6. Extract and display the first character of the string.
7. Extract and display the substring from index ‘2‘ to ‘5‘.
8. Reverse and display 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);
        char firstChar = inputString.charAt(0);
        System.out.println("First character: " + firstChar);
        String substring = inputString.substring(2, 5);
        System.out.println("Substring from index 2 to 5:"+ substring);
        String reversedString = new StringBuilder(inputString).
        reverse().toString();
        System.out.println("Reversed string: " + reversedString);
        scanner.close();
    }
}
```

Result

The program was executed successfully.

```
Enter a String: Manipulation
Length of the string: 12
String in uppercase: MANIPULATION
String in lowercase: manipulation
First character of the String : M
Substring from index 2 to 5: nip
Reversed string: noitalupinaM
```

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. Define a Publisher class with a name and method to display it.
3. Define a Book class that inherits from Publisher and includes book title and author.
4. Define a Literature class that inherits from Book and includes a genre.
5. Define a Fiction class that inherits from Book and includes a category.
6. In the main method:
   6.1 Take user input for a Literature book (publisher, title, author, genre).
   6.2 Take user input for a Fiction book (publisher, title, author, category).
   6.3 Create objects for Literature and Fiction books.
   6.4 Display the details of both books.
7. Stop

Source Code

```
import java.util.Scanner;

class Publisher {
    String publisherName;
    Publisher(String publisherName) {
        this.publisherName = publisherName;
    }

    void displayPublisher() {
        System.out.println("Publisher: " + publisherName);
    }
}

class Book extends Publisher {
    String bookTitle;
    String authorName;

    Book(String publisherName, String bookTitle, String authorName) {
        super(publisherName);
        this.bookTitle = bookTitle;
        this.authorName = authorName;
    }


```

```
    void displayBook() {
        displayPublisher();
        System.out.println("Book Title: " + bookTitle);
        System.out.println("Author: " + authorName);
    }
}

class Literature extends Book {
    String genre;

    Literature(String publisherName, String bookTitle, String
   authorName, String genre) {
        super(publisherName, bookTitle, authorName);
        this.genre = genre;
    }

    void display() {
        System.out.println("\n---Literature Book Details---");
        displayBook();
        System.out.println("Genre: " + genre);
    }
}

class Fiction extends Book {
    String category;

    Fiction(String publisherName, String bookTitle, String authorName,
   String category) {
        super(publisherName, bookTitle, authorName);
        this.category = category;
    }
    void display() {
        System.out.println("\n---Fiction Book Details---");
        displayBook();
        System.out.println("Category: " + category);
    }
}

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

        System.out.println("Enter details for Literature book:");
        System.out.print("Publisher Name: ");
        String pub1 = sc.nextLine();
        System.out.print("Book Title: ");
        String title1 = sc.nextLine();
        System.out.print("Author Name: ");
        String author1 = sc.nextLine();
        System.out.print("Genre: ");
        String genre = sc.nextLine();

        System.out.println("\nEnter details for Fiction book:");
        System.out.print("Publisher Name: ");
        String pub2 = sc.nextLine();
        System.out.print("Book Title: ");
        String title2 = sc.nextLine();
        System.out.print("Author Name: ");
```

```
        String author2 = sc.nextLine();
        System.out.print("Category: ");
        String category = sc.nextLine();

        Literature litBook = new Literature(pub1, title1,author1,
         genre);
        Fiction ficBook = new Fiction(pub2, title2, author2, category);

        litBook.display();
        ficBook.display();

        sc.close();
    }
}
```

Result

The program was executed successfully.

```
Enter details for Literature book:
Publisher Name: Manorama Books
Book Title: Aadujeevitham
Author Name: Benyamin
Genre: Novel

Enter details for Fiction book:
Publisher Name: H & C
Book Title: The Jungle Book
Author Name: Rudyard Kipling
Category: Fantasy

---Literature Book Details---
Publisher: Manorama Books
Book Title: Aadujeevitham
Author: Benyamin
Genre: Novel

---Fiction Book Details---
Publisher: H & C
Book Title: The Jungle Book
Author: Rudyard Kipling
Category: Fantasy
```

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. Initialize Scanner for user input.
3. Display menu options:
    3.1 "1. Circle"
    3.2 "2. Rectangle"
    3.3 "3. Exit"
4. Loop until the user chooses to exit:
    4.1 Prompt user to enter choice.
    4.2 If choice is 1 (Circle):
        4.2.1 Prompt user to enter the radius.
        4.2.2 Create a Circle object with the given radius.
        4.2.3 Calculate and display the area and perimeter of the circle.
    4.3 If choice is 2 (Rectangle):
        4.3.1 Prompt user to enter the length and width.
        4.3.2 Create a Rectangle object with the given dimensions.
        4.3.3 Calculate and display the area and perimeter of the rectangle.
    4.4 If choice is 3, print "Exiting." and terminate the loop.
    4.5 If choice is invalid, display an error message.
5. Close Scanner.
6. End
```

Source Code

```java
import java.util.Scanner;

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

// Circle class implementing Shape interface
class Circle implements Shape {
double radius;
// Constructor
Circle(double radius) {
    this.radius = radius;
}
```

```
// Implementing area method
public double area() {
    return Math.PI * radius * radius;
}

// Implementing perimeter method
public double perimeter() {
    return 2 * Math.PI * radius;
}
}

// Rectangle class implementing Shape interface
class Rectangle implements Shape {
double length, width;

// Constructor
Rectangle(double length, double width) {
    this.length = length;
    this.width = width;
}

// Implementing area method
public double area() {
    return length * width;
}

// Implementing perimeter method
public double perimeter() {
    return 2 * (length + width);
}
}

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

while (true) {
        System.out.println("\nChoose a shape:");
        System.out.println("1. Circle");
        System.out.println("2. Rectangle");
        System.out.println("3. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();

        if (choice == 1) {
            // Circle input
            System.out.print("Enter radius of the circle: ");
            double radius = scanner.nextDouble();
            Circle circle = new Circle(radius);
            System.out.println("Area: " + circle.area());
            System.out.println("Perimeter: " + circle.perimeter());
        }
        else if (choice == 2) {
            // Rectangle input
            System.out.print("Enter length of the rectangle: ");
            double length = scanner.nextDouble();
            System.out.print("Enter width of the rectangle: ");
```

```
            double width = scanner.nextDouble();
            Rectangle rectangle = new Rectangle(length, width);
            System.out.println("Area: " + rectangle.area());
            System.out.println("Perimeter: " + rectangle.perimeter());
        }
        else if (choice == 3) {
            System.out.println("Exiting...");
            break;
        }
        else {
            System.out.println("Invalid choice. Please try again.");
        }
    }
    scanner.close();
}
}
```

Result

The program was executed successfully.

```
Choose a Shape:
1. Circle
2. Rectangle
3. Exit

Enter your choice: 1
Enter radius of the circle: 7
Area: 153.93804002589985
Perimeter: 43.982297150257104

Enter your choice: 2
Enter length of the rectangle: 12
Enter width of the rectangle: 5
Area: 60.0
Perimeter: 34.0

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. Initialize Scanner for user input.
3. Create an ArrayList to store Employee objects.
4. Take input for Manager:
    4.1 Prompt the user to enter the Manager's name.
    4.2 Prompt the user to enter the Manager's salary.
    4.3 Prompt the user to enter the Manager's bonus.
    4.4 Create a Manager object using the input values.
    4.5 Add the Manager object to the employees list.
5. Take input for Developer:
    5.1 Prompt the user to enter the Developer's name.
    5.2 Prompt the user to enter the Developer's salary.
    5.3 Prompt the user to enter the Developer's experience in years.
    5.4 Create a Developer object using the input values.
    5.5 Add the Developer object to the employees list.
6. Display details of all employees:
    6.1 Loop through each employee in the employees list.
        6.1.1 If the employee is a Manager:
            - Display Manager details.
            - Assign the Manager to "Generative AI" project.
            - Assign the Manager to "IOT" project
              with a team size of 5.
        6.1.2 If the employee is a Developer:
            - Display Developer details.
7. Close Scanner.
8. End
```

Source Code

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

// Abstract Employee class
abstract class Employee {
String name;
double salary;

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

// Abstract method to calculate salary
abstract double calculateSalary();

// Method to display details
void displayDetails() {
    System.out.println("Name: " + name);
    System.out.println("Salary: " + salary);
}
}

// Benefits interface
interface Benefits {
double calculateBenefits();
}

// Manager class
class Manager extends Employee implements Benefits {
double bonus;

Manager(String name, double salary, double bonus) {
    super(name, salary);
    this.bonus = bonus;
}

// Overriding calculateSalary()
@Override
double calculateSalary() {
    return salary + bonus;
}

// Overriding calculateBenefits()
@Override
public double calculateBenefits() {
    return 5000; // Fixed insurance benefit
}

// Method overloading: Assigning projects
void assignProject(String projectName) {
    System.out.println(name + " assigned to project: " + projectName);
}
void assignProject(String projectName, int teamSize) {
    System.out.println(name + " assigned to project: " + projectName +
   " with team size: " + teamSize);
}
```

```
// Display details
void display() {
    displayDetails();
    System.out.println("Bonus: " + bonus);
    System.out.println("Total Salary: " + calculateSalary());
    System.out.println("Benefits: " + calculateBenefits());
}
}

// Developer class
class Developer extends Employee implements Benefits {
int experience;

Developer(String name, double salary, int experience) {
    super(name, salary);
    this.experience = experience;
}

// Overriding calculateSalary()
@Override
double calculateSalary() {
    return salary + (experience * 1000); // Extra 1000 per year of
   experience
}

// Overriding calculateBenefits()
@Override
public double calculateBenefits() {
    return experience * 500; // Allowance based on experience
}

// Display details
void display() {
    displayDetails();
    System.out.println("Experience: " + experience + " years");
    System.out.println("Total Salary: " + calculateSalary());
    System.out.println("Benefits: " + calculateBenefits());
}
}

// Main class
public class ManageEmployee {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
ArrayList<Employee> employees = new ArrayList<>();

// Taking input for a Manager
    System.out.println("Enter Manager details:");
    System.out.print("Name: ");
    String mgrName = scanner.nextLine();
    System.out.print("Salary: ");
    double mgrSalary = scanner.nextDouble();
    System.out.print("Bonus: ");
    double mgrBonus = scanner.nextDouble();
    scanner.nextLine(); // Consume newline
    Manager manager = new Manager(mgrName, mgrSalary, mgrBonus);
    employees.add(manager);

```

```
114     // Taking input for a Developer
115     System.out.println("\nEnter Developer details:");
116     System.out.print("Name: ");
117     String devName = scanner.nextLine();
118     System.out.print("Salary: ");
119     double devSalary = scanner.nextDouble();
120     System.out.print("Experience (years): ");
121     int devExp = scanner.nextInt();
122     Developer developer = new Developer(devName, devSalary, devExp);
123     employees.add(developer);
124 
125     // Display all employees
126     System.out.println("\n--- Employee Details ---");
127     for (Employee emp : employees) {
128         if (emp instanceof Manager) {
129             ((Manager) emp).display();
130             ((Manager) emp).assignProject("Generative AI");
131             ((Manager) emp).assignProject("IOT", 5);
132         } else if (emp instanceof Developer) {
133             ((Developer) emp).display();
134         }
135         System.out.println("----------------------");
136     }
137     scanner.close();
138  }
139 }
```

Result

The program was executed successfully.

```
Enter Manager details:
Name: Amar
Salary: 28000
Bonus: 2000

Enter Developer details:
Name: Philip
Salary: 15000
Experience (years): 2

--- Employee Details ---
Name: Amar
Salary: 28000.0
Bonus: 2000.0
Total Salary: 30000.0
Benefits: 5000.0
Amar assigned to project: IOT
Amar assigned to project: Generative AI with team size: 5
```

```
----------------------

Name: Philip
Salary: 15000.0
Experience: 2 years
Total Salary: 17000.0
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. Initialize Scanner for user input.
3. Take input for Rectangle:
    3.1 Prompt user to enter length and width.
    3.2 Create a Rectangle object with input values.
    3.3 Calculate and display the area of the Rectangle.
4. Take input for Triangle:
    4.1 Prompt user to enter base and height.
    4.2 Create a Triangle object with input values.
    4.3 Calculate and display the area of the Triangle.
5. Take input for Square:
    5.1 Prompt user to enter the side length.
    5.2 Create a Square object with input value.
    5.3 Calculate and display the area of the Square.
6. Take input for Circle:
    6.1 Prompt user to enter the radius.
    6.2 Create a Circle object with input value.
    6.3 Calculate and display the area of the Circle.
7. Close Scanner.
8. End

Source Code

Main.java

```java
// Importing the graphics package
import graphics.*; // Importing the graphics package
import java.util.Scanner;

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

        // Rectangle
        System.out.println("Enter length and width of the Rectangle:");
        double length = scanner.nextDouble();
        double width = scanner.nextDouble();
        Rectangle rect = new Rectangle(length, width);
        System.out.println("Area of Rectangle: " + rect.area());
```

```
15
16          // Triangle
17          System.out.println("\nEnter base and height of the Triangle:");
18          double base = scanner.nextDouble();
19          double height = scanner.nextDouble();
20          Triangle tri = new Triangle(base, height);
21          System.out.println("Area of Triangle: " + tri.area());
22
23          // Square
24          System.out.println("\nEnter side of the Square:");
25          double side = scanner.nextDouble();
26          Square square = new Square(side);
27          System.out.println("Area of Square: " + square.area());
28
29          // Circle
30          System.out.println("\nEnter radius of the Circle:");
31          double radius = scanner.nextDouble();
32          Circle circle = new Circle(radius);
33          System.out.println("Area of Circle: " + circle.area());
34
35          scanner.close();
36      }
37  }
```

Shape.java

```
1  package graphics;
2
3  // Interface for Shapes
4  public interface Shape {
5      double area();
6  }
```

Circle.java

```
1  package graphics;
2  // Circle class implementing Shape
3  public class Circle implements Shape {
4      double radius;
5
6      public Circle(double radius) {
7          this.radius = radius;
8      }
9      @Override
10     public double area() {
11         return Math.PI * radius * radius;
12     }
13 }
```

Rectangle.java

```
1  package graphics;
2
3  // Rectangle class implementing Shape
4  public class Rectangle implements Shape {
5      double length, width;
6
7      public Rectangle(double length, double width) {
8          this.length = length;
9          this.width = width;
10     }
```

```
    @Override
    public double area() {
        return length * width;
    }
}
```

Triangle.java

```
package graphics;

// Triangle class implementing Shape
public class Triangle implements Shape {
    double base, height;

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

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

Square.Java

```
package graphics;

// Square class implementing Shape
public class Square implements Shape {
    double side;

    public Square(double side) {
        this.side = side;
    }

    @Override
    public double area() {
        return side * side;
    }
}
```

Result

The program was executed successfully.

```
Enter length and width of the Rectangle:
12
5
Area of Rectangle: 60.0
```

```
Enter base and height of the Triangle:
```

```
5
14
Area of Triangle: 35.0

Enter side of the Square:
7
Area of Square: 49.0

Enter radius of the Circle:
7
Area of Circle: 153.93804002589985
```

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

File Operations in Java

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. Define ‘writeFile‘ method to:
   a. Create a ‘FileWriter‘ for the specified filename.
   b. Write the provided data to the file.
   c. Close the writer.
   d. Print confirmation message.

3. Define ‘readFile‘ method to:
   a. Check if the file exists.
   b. Use ‘BufferedReader‘ to read the file line by line.
   c. Print the file contents.
   d. Close the reader.

4. Define ‘appendToFile‘ method to:
   a. Open the file in append mode.
   b. Append the provided data.
   c. Close the writer.
   d. Print confirmation message.

5. Define ‘main‘ method to:
   a. Create a ‘Scanner‘ object for user input.
   b. Prompt the user to choose an option:
      i.   1 - Write
      ii.  2 - Read
      iii. 3 - Append
   c. Read the user’s choice.
   d. Prompt the user to enter the filename.
   e. Depending on the choice:
      i.   If choice is 1 (Write):
           - Prompt and read data.
           - Call ‘writeFile(filename, data)‘.
      ii.  If choice is 2 (Read):
           - Call ‘readFile(filename)‘.
      iii. If choice is 3 (Append):
           - Prompt and read data.
           - Call ‘appendToFile(filename, data)‘.
```

iv. Else:
- Print "Invalid choice."
f. Handle any ‘IOException‘ errors.
g. Close the ‘Scanner‘.
6. Stop

Source Code

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

public class FileOperations {
    public static void writeFile(String filename, String data) throws IOException {
        FileWriter writer = new FileWriter(filename);
        writer.write(data);
        writer.close();
        System.out.println("Data written to file successfully.");
    }

    public static void readFile(String filename) throws IOException {
        File file = new File(filename);
        if (!file.exists()) {
            throw new FileNotFoundException("File not found.");
        }
        BufferedReader reader = new BufferedReader(new FileReader(filename));
        String line;
        System.out.println("File contents:");
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        reader.close();
    }

    public static void appendToFile(String filename, String data) throws IOException {
        FileWriter writer = new FileWriter(filename, true);
        writer.write(data);
        writer.close();
        System.out.println("Data appended to file successfully.");
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Choose an option:\n1. Write\n2. Read\n3. Append");
        int choice = scanner.nextInt();
        scanner.nextLine();

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

        try {
```

```
            switch (choice) {
                case 1:
                    System.out.print("Enter data to write: ");
                    String writeData = scanner.nextLine();
                    writeFile(filename, writeData);
                    break;
                case 2:
                    readFile(filename);
                    break;
                case 3:
                    System.out.print("Enter data to append: ");
                    String appendData = scanner.nextLine();
                    appendToFile(filename, appendData);
                    break;
                default:
                    System.out.println("Invalid choice.");
            }
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
        scanner.close();
    }
}
```

Result

The program was executed successfully.

```
 File operations in Java
1. Write
2. Read
3. Append

Choose an Operation :
1
Enter filename: sample.txt
Enter data to write: This is File operations in Java
Data written to file successfully.

Choose an Operation :
2
Enter filename: sample.txt
File contents:
This is File operations in Java

Choose an Operation :
3
Enter filename: sample.txt
Enter data to append: . It supports Exception Handling
Data appended to file successfully.
```

```
Choose an Operation :
2
Enter filename: sample.txt
File contents:
This is File operations in Java. It supports Exception Handling
```

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. Initialize Scanner to take user input.
3. Prompt user to enter the filename.
4. Try to read the file:
    4.1 If the file does not exist, throw a FileNotFoundException.
    4.2 If any other I/O error occurs, throw an IOException.
    4.3 If an exception occurs, display the error message and
    terminate the program.
5. Prompt user to enter username and password.
6. Try to authenticate user:
    6.1 If the username is not "admin" or the password is not "admin123",
    throw an AuthenticationException.
    6.2 If authentication fails, display the error message and
    terminate the process.
7. If authentication is successful, proceed to read the file:
    7.1 Open the file using BufferedReader.
    7.2 Read and display its contents line by line.
    7.3 Handle any IOException that may occur while reading.
8. Close Scanner to prevent resource leaks.
9. End
```

Source Code

```
1 import java.io.*;
2 import java.util.Scanner;
3
```

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

public class ExceptionHandling {

    // Method to read a file
    public static void readFile(String filename) throws IOException {
        File file = new File(filename);
        if (!file.exists()) {
            throw new FileNotFoundException("File not found.");
        }
    }

    // Method to authenticate user
    public static void authenticate(String username, String password)
    throws AuthenticationException {
        if (!username.equals("admin") || !password.equals("admin123"))
    {
            throw new AuthenticationException("Invalid username or
    password.");
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter filename: ");
        String filename = scanner.nextLine();

        try {
            readFile(filename);  // Check if the file exists
        } catch (FileNotFoundException e) {
            System.out.println("Error: " + e.getMessage());
            scanner.close();
            return; // Exit program if file not found
        } catch (IOException e) {
            System.out.println("IO Error: " + e.getMessage());
            scanner.close();
            return;
        }
        System.out.print("Enter username: ");
        String username = scanner.nextLine();
        System.out.print("Enter password: ");
        String password = scanner.nextLine();

        try {
            authenticate(username, password);  // Check authentication
            System.out.println("Authentication successful.");

            // If authentication is successful, read and display file
    content
            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 (IOException e) {
                System.out.println("IO Error while reading file: " + e.
    getMessage());
            }

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

Result

The program was executed successfully.

```
Enter filename: file2.txt
Error: File not found.

Enter filename: file1.txt
Enter username: abc
Enter password: abcd
Authentication Failed: Invalid username or password.

Enter filename: file1.txt
Enter username: abc
Enter password: 123
Authentication Failed: Invalid username or password.

Enter filename: file1.txt
Enter username: admin
Enter password: admin123
Authentication successful.
File contents:
welcome to file operations in Java

Enter filename: file1.txt
Enter username: admin
Enter password: admin123
Authentication successful.
IO Error while reading file: file1.txt (Permission denied)
```

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. Initialize Scanner to take user input.
3. Prompt the user to enter the number of prime numbers to generate.
4. Create a shared resource object to synchronize threads.
5. Create and start the MultiplicationTable thread:
    5.1 Print the multiplication table of 5 (1 to 10).
    5.2 After printing each line, switch the turn to the PrimeNumbers thread.
    5.3 Notify the waiting thread.
6. Create and start the PrimeNumbers thread:
    6.1 Generate the first N prime numbers.
    6.2 After printing each prime number, switch the turn to the MultiplicationTable
    6.3 Notify the waiting thread.
7. Wait for both threads to finish execution.
8. Print completion message.
9. Close the Scanner.
10. End

Source Code

```
import java.util.Scanner;

class SharedResource {
    boolean printMultiplication = true; // Flag to control execution
    order
}

class MultiplicationTable extends Thread {
    private final SharedResource resource;

    public MultiplicationTable(SharedResource resource) {
        this.resource = resource;
    }

    public void run() {
        synchronized (resource) {
            for (int i = 1; i <= 10; i++) {
```

```
                while (!resource.printMultiplication) { // Wait if it's not this thread's turn
                    try {
                        resource.wait();
                    } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                    }
                }
                System.out.println(i + " x 5 = " + (5 * i));
                resource.printMultiplication = false; // Switch turn to PrimeNumbers
                resource.notify(); // Notify the waiting thread
                try {
                    Thread.sleep(500); // Simulating processing delay
                } catch (InterruptedException e) {
                    System.out.println(e.getMessage());
                }
            }
        }
    }
}

class PrimeNumbers extends Thread {
    private final SharedResource resource;
    private int N;

    public PrimeNumbers(SharedResource resource, int N) {
        this.resource = resource;
        this.N = N;
    }

    public void run() {
        synchronized (resource) {
            int count = 0, num = 2;
            while (count < N) {
                while (resource.printMultiplication) { // Wait if it's not this thread's turn
                    try {
                        resource.wait();
                    } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                    }
                }
                if (isPrime(num)) {
                    System.out.println("Prime: " + num);
                    count++;
                    resource.printMultiplication = true; // Switch turn to MultiplicationTable
                    resource.notify(); // Notify the waiting thread
                    try {
                        Thread.sleep(300); // Simulating processing delay
                    } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                    }
                }
                num++;
            }
```

```
        }
    }

    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 MultithreadingExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of prime numbers to generate
    : ");
        int N = scanner.nextInt();

       SharedResource resource = new SharedResource();
        MultiplicationTable tableThread = new MultiplicationTable(
    resource);
        PrimeNumbers primeThread = new PrimeNumbers(resource, N);
        tableThread.start();
        primeThread.start();

        // Wait for both threads to finish
        try {
            tableThread.join();
            primeThread.join();
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
        System.out.println("Multithreading demonstration completed.");
        scanner.close();
    }
}
```

Result

The program was executed successfully.

```
Enter the number of prime numbers to generate: 5

1 x 5 = 5
Prime: 2
2 x 5 = 10
Prime: 3
3 x 5 = 15
Prime: 5
4 x 5 = 20
```

```
Prime: 7
5 x 5 = 25
Prime: 11
6 x 5 = 30
```