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: NANDANA RAMACHANDRAN

Branch: Master of Computer Applications

Semester: 2

Roll No: 39

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

Source Code

```java
import java.util.Scanner;
public class Oddoreven{
        public static void main(String arg[]){
                Scanner s=new Scanner(System.in);
                System.out.print("Enter the number:");
                int num=s.nextInt();
                if(num%2==0){
                        System.out.print("Even");
                }
                else{
                        System.out.print("Odd");
                }
        }
}
```

Result

The program was executed successfully.

```
Enter the number:5
Odd

Enter the number:6
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. Start
2. Take an integer n as input from the user.
3. Use either a for loop or a while loop to compute the sum.
4. Print the result.
5. Stop
```

Source Code

```java
import java.util.Scanner;
public class naturalnos
{
        public static void main(String arg[]){
                Scanner s=new Scanner(System.in);
                System.out.print("Enter n:");
                int n=s.nextInt();
                int sum=0;
                for(int i=1;i<=n;i++)
                        sum=sum+i;
                System.out.print("Sum="+sum);
        }
}
```

Result

The program was executed successfully.

```
Enter n:6
Sum=21
```

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. Start
2. Take an integer as input from the user.
3. Compute the factorial using either a for loop or a while loop.
4. Print the result.
5. Stop
```

Source Code

```java
import java.util.Scanner;
public class factorial
{
	public static void main(String arg[]){
		Scanner s=new Scanner(System.in);
		System.out.print("Enter the number:");
		int n=s.nextInt();
		int fact=1;
		for(int i=1;i<=n;i++)
			fact=fact*i;
		System.out.print("Factorial="+fact);
	}
}
```

Result

The program was executed successfully.

```
Enter the number:5
Factorial=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. Start
2. Take a numeric score (0{100) as input from the user.
3. 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
4. Print the assigned grade.
5. Stop
```

Source Code

```java
import java.util.Scanner;
public class grade{
        public static void main(String arg[]){
                Scanner s=new Scanner(System.in);
                System.out.print("Enter the mark:");
                int m=s.nextInt();
                if(m>=90){
                        System.out.print("Grade A");
                }
                else if(m>=80){
                        System.out.print("Grade B");
                }
                else if(m>=70){
                        System.out.print("Grade C");
                }
                else if(m>=60){
                        System.out.print("Grade D");
                }
                else{
                        System.out.print("Grade F");
                }
        }
}
```

Result

The program was executed successfully.

```
Enter the mark:68
Grade D
Enter the mark:75
Grade C
Enter the mark:88
Grade B
Enter the mark:95
Grade A
Enter the mark:45
Grade F
```

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. 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

```java
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) +":") ;
```

```java
                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

Write a 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

```java
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

Write a 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

```java
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. Create an array to store employee details.
3. Get the number of employees and their details from user.
4. Prompt the user to enter an employee number to search.
5. Search for the employee in the array.
6. If found, display the employee details.
7. Stop
```

Source Code

```java
import java.util.Scanner;
class Employee {
    int empNo;
    String name;
    double salary;
    Employee(int empNo, String name, double salary) {
        this.empNo = empNo;
        this.name = name;
        this.salary = salary;
    }
    void display() {
        System.out.println("Employee Number: " + empNo);
        System.out.println("Name: " + name);
        System.out.println("Salary: " + salary);
    }
}
public class EmployeeSearch {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter number of employees: ");
        int n = scanner.nextInt();
        Employee[] employees = new Employee[n];
        for (int i = 0; i < n; i++) {
            System.out.print("Enter Employee Number: ");
            int empNo = scanner.nextInt();
            scanner.nextLine();
            System.out.print("Enter Name: ");
            String name = scanner.nextLine();
            System.out.print("Enter Salary: ");
            double salary = scanner.nextDouble();
            employees[i] = new Employee(empNo, name, salary);
        }
        System.out.print("Enter Employee Number to search: ");
        int searchNo = scanner.nextInt();
        boolean found = false;
        for (Employee emp : employees) {
            if (emp.empNo == searchNo) {
                System.out.println("Employee Found:");
                emp.display();
                found = true;
                break;
            }
        }
        if (!found) {
            System.out.println("Employee not found.");
        }
    }
}
```

Result

The program was executed successfully.

```
Enter number of employees: 3
Enter Employee Number: 101
Enter Name: Akshay
Enter Salary: 50000
Enter Employee Number: 102
Enter Name: Nandana
Enter Salary: 45000
Enter Employee Number: 103
Enter Name: Yash
Enter Salary: 10000
Enter Employee Number to search: 106
Employee not found.

Enter number of employees: 3
Enter Employee Number: 101
Enter Name: Akshay
Enter Salary: 50000
Enter Employee Number: 102
Enter Name: Nandana
Enter Salary: 45000
Enter Employee Number: 103
Enter Name: Yash
Enter Salary: 10000
Enter Employee Number to search: 102
Employee Found:
Employee Number: 102
Name: Nandana
Salary: 45000.0
```

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

String Search in an Array

Aim

Create an object-oriented 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. Get the number of strings and store them in an array.
3. Prompt the user to enter the string to search.
4. Search for the string in the array.
5. If found, print its index; otherwise, 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 number of strings: ");
        int n = scanner.nextInt();
        scanner.nextLine();
        String[] strings = new String[n];
        for (int i = 0; i < n; i++) {
            System.out.print("Enter string " + (i + 1) + ": ");
            strings[i] = scanner.nextLine();
        }
        System.out.print("Enter string to search: ");
        String searchStr = scanner.nextLine();
        boolean found = false;
        for (int i = 0; i < n; i++) {
            if (strings[i].equals(searchStr)) {
                System.out.println("String found at index: " + i);
                found = true;
                break;
            }
        }
        if (!found) {
            System.out.println("String not found.");
        }
    }
}
```

Result

The program was executed successfully.

```
Enter number of strings: 3
Enter string 1: Hi
Enter string 2: Hello
Enter string 3: World!
Enter string to search: hello
String not found.

Enter number of strings: 3
Enter string 1: Hi
Enter string 2: Hello
Enter string 3: World!
Enter string to search: Hello
String found at index: 1
```

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. Get a string from the user.
3. Find and print the length of string.
4. Convert and print the string in uppercase and lowercase.
5. Extract and print specific characters and substrings.
6. Reverse and print the string.
7. Stop
```

Source Code

```java
import java.util.Scanner;
public class StringManipulation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String str = scanner.nextLine();
        System.out.println("Length: " + str.length());
        System.out.println("Uppercase: " + str.toUpperCase());
        System.out.println("Lowercase: " + str.toLowerCase());
        if (str.length() > 2) {
            System.out.println("First character: " + str.charAt(0));
            System.out.println("Substring (first 3 characters): " + str.substring(0, 3));
        }
        String reversed = new StringBuilder(str).reverse().toString();
        System.out.println("Reversed: " + reversed);
    }
}
```

Result

The program was executed successfully.

```
Enter a string: Nandana
Length: 7
Uppercase: NANDANA
Lowercase: nandana
First character: N
```

```
Substring (first 3 characters): Nan
Reversed: anadnaN
```

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 base class Publisher with publisher details.
3. Create a derived class Book that extends Publisher and adds book.
4. Create two subclasses Literature and Fiction that extends Book.
5. Implement methods to read and display book details in each class.
6. Get user input to enter deetails for books in both categories.
7. Display the entered book details.
8. Stop
```

Source Code

```java
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 author;
    double price;
    Book(String publisherName, String bookTitle, String author, double
    price) {
        super(publisherName);
        this.bookTitle = bookTitle;
        this.author = author;
        this.price = price;
    }
    void displayBookDetails() {
        displayPublisher();
        System.out.println("Book Title: " + bookTitle);
        System.out.println("Author: " + author);
        System.out.println("Price: $" + price);
    }
}
class Literature extends Book {
    Literature(String publisherName, String bookTitle, String author, double price) {
        super(publisherName, bookTitle, author, price);
    }
    void display() {
        System.out.println("\n**Literature Book Details**");
        displayBookDetails();
    }
}
class Fiction extends Book {
    Fiction(String publisherName, String bookTitle, String author, double price) {
        super(publisherName, bookTitle, author, price);
    }
    void display() {
        System.out.println("\n**Fiction Book Details**");
        displayBookDetails();
    }
}
public class BookManagementSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter Literature Book Details:");
        System.out.print("Publisher: ");
        String litPublisher = scanner.nextLine();
        System.out.print("Title: ");
        String litTitle = scanner.nextLine();
        System.out.print("Author: ");
        String litAuthor = scanner.nextLine();
        System.out.print("Price: ");
        double litPrice = scanner.nextDouble();
        scanner.nextLine();
        Literature literature = new Literature(litPublisher, litTitle, litAuthor, litPrice);
        System.out.println("\nEnter Fiction Book Details:");
        System.out.print("Publisher: ");
        String ficPublisher = scanner.nextLine();
        System.out.print("Title: ");
        String ficTitle = scanner.nextLine();
        System.out.print("Author: ");
        String ficAuthor = scanner.nextLine();
        System.out.print("Price: ");
        double ficPrice = scanner.nextDouble();
        Fiction fiction = new Fiction(ficPublisher, ficTitle, ficAuthor, ficPrice);
        literature.display();
        fiction.display();
    }
}
```

Result

The program was executed successfully.

```
Enter Literature Book Details:
Publisher: ABC
```

```
Title: WINGS OF FIRE
Author: APJ Abdul Kalam
Price: 652

Enter Fiction Book Details:
Publisher: CBA
Title: JAVA
Author: author
Price: 183

**Literature Book Details**
Publisher: ABC
Book Title: WINGS OF FIRE
Author: APJ Abdul Kalam
Price: $652.0

**Fiction Book Details**
Publisher: CBA
Book Title: JAVA
Author: author
Price: $183.0
```

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. Define an interface Shape with two methods area() and perimeter().
3. Create a class circle that implements Shape:
    3.1:Define a variable radius to store the circle radius.
    3.2:Implement the area() method to return pi*radius^2.
    3.3:Implement the perimeter() to return 2*pi*radius.
4. Create a class rectangle that implements Shape:
    4.1:Define variable length and width to store dimensions.
    4.2:Implement the area() method to return length*width.
    4.3:Implement the perimeter() to return 2*(length+width).
5. In the main program:
    5.1:Display a menu to choose between circle and rectangle.
    5.2:If user selects circle,create a circle object and
        calculate area and perimeter.
    5.3:If user selects rectangle,create a rectangle object
        and calculate area and perimeter.
    5.4:Display the calculated values.
    5.5:Repeat the menu until the user choose to 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;
    }
}
class Rectangle implements Shape {
    double length, width;
    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    public double area() {
        return length * width;
    }
    public double perimeter() {
        return 2 * (length + width);
    }
}
public class AreaPerimeterCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int choice;
        do {
            System.out.println("\n1. Circle\n2. Rectangle\n3. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter radius: ");
                    double r = scanner.nextDouble();
                    Circle circle = new Circle(r);
                    System.out.println("Area: " + circle.area());
                    System.out.println("Perimeter: " + circle.perimeter());
                    break;
                case 2:
                    System.out.print("Enter length: ");
                    double l = scanner.nextDouble();
                    System.out.print("Enter width: ");
                    double w = scanner.nextDouble();
                    Rectangle rectangle = new Rectangle(l, w);
                    System.out.println("Area: " + rectangle.area());
                    System.out.println("Perimeter: " + rectangle.perimeter());
                    break;
                case 3:System.out.println("Exiting...");
                       break;
                default:System.out.println("Invalid choice!");
            }
        } while (choice != 3);
    }
}
```

Result

The program was executed successfully.

```
1. Circle
2. Rectangle
3. Exit
```

```
Enter your choice: 1
Enter radius: 5
Area: 78.53981633974483
Perimeter: 31.41592653589793

1. Circle
2. Rectangle
3. Exit
Enter your choice: 2
Enter length: 7
Enter width: 3
Area: 21.0
Perimeter: 20.0

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 name,salary and abstract
   method calculateSalary().
3. Create subclasses Manager and Developer that override
   calculateSalary().
4. Implement the interface in Manager and Developer.
5. Store multiple employeesin a list using polymorphism.
6. Overload a method manager for project assignment.
7. Display employee details and salary.
8. Stop
```

Source Code

```java
import java.util.ArrayList;
import java.util.Scanner;
abstract class Employee {
    String name;
    double salary;
    Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }
    abstract double calculateSalary();
    void displayDetails() {
        System.out.println("\nName: " + name);
        System.out.println("Salary: " + calculateSalary());
    }
}
interface Benefits {
    double calculateBenefits();
}
class Manager extends Employee implements Benefits {
    double bonus;
    Manager(String name, double salary, double bonus) {
        super(name, salary);
        this.bonus = bonus;
    }
    double calculateSalary() {
        return salary + bonus;
    }
    public double calculateBenefits() {
        return 5000;
    }
    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);
    }
}
class Developer extends Employee implements Benefits {
    int experience;
    Developer(String name, double salary, int experience) {
        super(name, salary);
        this.experience = experience;
    }
    double calculateSalary() {
        return salary + (experience * 1000);
    }
    public double calculateBenefits() {
        return experience * 500;
    }
}
public class EmployeeManagement {
    public static void main(String[] args) {
        ArrayList<Employee> employees = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter number of employees: ");
        int numEmployees = scanner.nextInt();
        scanner.nextLine();
        for (int i = 0; i < numEmployees; i++) {
            System.out.println("\nEnter details for Employee " + (i +
 1) + ":");
            System.out.print("Enter name: ");
            String name = scanner.nextLine();
            System.out.print("Enter salary: ");
            double salary = scanner.nextDouble();
            scanner.nextLine();
            System.out.print("Enter type (Manager/Developer): ");
            String type = scanner.nextLine();
            if (type.equalsIgnoreCase("Manager")) {
                System.out.print("Enter bonus amount: ");
                double bonus = scanner.nextDouble();
                scanner.nextLine();
                employees.add(new Manager(name, salary, bonus));
            } else if (type.equalsIgnoreCase("Developer")) {
                System.out.print("Enter experience (years): ");
                int experience = scanner.nextInt();
                scanner.nextLine();
                employees.add(new Developer(name, salary, experience));
            } else {
                System.out.println("Invalid entry.");
            }
        }
        System.out.println("\nEmployee Details");
        for (Employee emp : employees) {
            emp.displayDetails();
            if (emp instanceof Benefits) {
                System.out.println("Benefits: " + ((Benefits) emp).calculateBenefits());
            }
        }
        scanner.close();
    }
}
```

Result

The program was executed successfully.

```
Enter number of employees: 2

Enter details for Employee 1:
Enter name: Akshay
Enter salary: 50000
Enter type (Manager/Developer): Manager
Enter bonus amount: 10000

Enter details for Employee 2:
Enter name: Nandana
Enter salary: 45000
Enter type (Manager/Developer): Developer
Enter experience (years): 6

Employee Details

Name: Akshay
Salary: 60000.0
Benefits: 5000.0

Name: Nandana
Salary: 51000.0
Benefits: 3000.0
```

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. Create a package Graphics.
3. Inside this package define an interface Shape with a method area().
4. Create class Rectangle,Triangle,Square,Circle in the package.
5. In the main program import graphics package.
6. Stop
```

Source Code

```java
//TestShapes.java
import java.util.Scanner;
import Graphics.*;
public class TestShapes {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter length of Rectangle: ");
        double rectLength = scanner.nextDouble();
        System.out.print("Enter width of Rectangle: ");
        double rectWidth = scanner.nextDouble();
        Rectangle rectangle = new Rectangle(rectLength, rectWidth);
      System.out.println("\nArea of Rectangle: " + rectangle.area());
        System.out.print("\nEnter base of Triangle: ");
        double triBase = scanner.nextDouble();
        System.out.print("Enter height of Triangle: ");
        double triHeight = scanner.nextDouble();
        Triangle triangle = new Triangle(triBase, triHeight);
      System.out.println("\nArea of Triangle: " + triangle.area());
        System.out.print("\nEnter side of Square: ");
        double squareSide = scanner.nextDouble();
        Square square = new Square(squareSide);
      System.out.println("\nArea of Square: " + square.area());
        System.out.print("\nEnter radius of Circle: ");
        double circleRadius = scanner.nextDouble();
        Circle circle = new Circle(circleRadius);
      System.out.println("\nArea of Circle: " + circle.area());
        scanner.close();
  }
}

//Shape.java
package Graphics;
```

```java

public interface Shape {
    double area();
}

//Rectangle.java
package Graphics;
public class Rectangle implements Shape {
    private double length;
    private double width;
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    @Override
    public double area() {
        return length * width;
    }
}

//Triangle.java
package Graphics;
public class Triangle implements Shape {
    private double base;
    private double 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;
public class Square extends Rectangle {
    public Square(double side) {
        super(side, side);
    }
}

//Cirlce.java
package Graphics;
public class Circle implements Shape {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }

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

Result

The program was executed successfully.

```
Enter length of Rectangle: 5
Enter width of Rectangle: 7

Area of Rectangle: 35.0

Enter base of Triangle: 3
Enter height of Triangle: 9

Area of Triangle: 13.5

Enter side of Square: 25

Area of Square: 625.0

Enter radius of Circle: 81

Area of Circle: 20611.98940020263
```

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. Create a menu for the file operations.
3. Get user input for the operation.
4. Perform the selected operation.
5. If the user chooses to write a file,create a file and write data.
6. If the user chooses to read a file,check if the file exists,
   then read and display its contents.
7. Ithe user chooses to append to a file,open the file in append mode
   and write additional data.
8. Handle exceptions such as file not found and IO exception.
9. Close the  file and display success message.
10. Stop
```

Source Code

```java
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.

```
Choose an option:
1. Write
2. Read
3. Append
1
Enter filename: example
Enter data to write: Welcome
Data written to file successfully.

Choose an option:
1. Write
2. Read
```

```
3. Append
2
Enter filename: example
File contents:
Welcome

Choose an option:
1. Write
2. Read
3. Append
2
Enter filename: abc
Error: File not found.

Choose an option:
1. Write
2. Read
3. Append
3
Enter filename: example
Enter data to append:  to the world!
Data appended to file successfully.
```

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. Prompts the user to enter a filename.
3. Call the read file method.
4. If the file is found, read and print its contents.
5. If the file not found, catch file not found exception and display it.
6. If other IO errors occur,catch IO exception and display an error
   message.
7. Prompt the user to enter a username and password.
8. Call the authenticate method,if the credentials match
   predefined values authentication is successful.
9. If authentication fails, throw an authentication exception.
10. Catch and handle exceptions,display appropriate message.
11. Stop
```

Source Code

```java
import java.io.*;
import java.util.Scanner;
class AuthenticationException extends Exception {
    public AuthenticationException(String message) {
        super(message);
    }
}
public class ExceptionHandling {
    public static void readFile(String filename) throws IOException {
        File file = new File(filename);
        if (!file.exists()) {
            throw new FileNotFoundException("File not found.");
        }
    }
    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);
        } catch (FileNotFoundException e) {
            System.out.println("Error: " + e.getMessage());
            scanner.close();
            return;
        } 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);
            System.out.println("Authentication successful.");
            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: example
Enter username: admin
Enter password: admin123
```

```
Authentication successful.
File contents:
Welcome to the world!

Enter filename: abc3
Error: File not found.

Enter filename: example
Enter username: example
Enter password: example
Authentication Failed: Invalid username or password.
```

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

Multithreading

Aim

Write a Java program that defines two classes: one for generating and displaying the multiplication 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. Define a class MultiplicationTable that extends Thread.
3. Override the run() method to generate and display the multiplication
   table of number.
4. Define class Primenumbers that implements runnable.
5. Override the run() method to print the first n prime numbers.
6. Create an instance of multiplication table and start the thread.
7. Create an instance of prime numbers wrap in a thread object and
   start the thread.
8. Ensure proper thread management using synchronization if needed.
9. Stop
```

Source Code

```java
import java.util.Scanner;
class SharedResource {
    boolean printMultiplication = true;
}
class MultiplicationTable extends Thread {
    private final SharedResource resource;
    private int N;
    public MultiplicationTable(SharedResource resource, int N) {
        this.resource = resource;
  this.N=N;
    }
    public void run() {
        synchronized (resource) {
            for (int i = 1; i <= N; i++) {
                while (!resource.printMultiplication) {
                    try {
                        resource.wait();
                    } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                    }
                }
                System.out.println(i + " x 5 = " + (5 * i));
                resource.printMultiplication = false;
                resource.notify();
                try {
                    Thread.sleep(500);
                } 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) {
                    try {
                        resource.wait();
                    } catch (InterruptedException e) {
                        System.out.println(e.getMessage());
                    }
                }
                if (isPrime(num)) {
                    System.out.println("Prime: " + num);
                    count++;
                    resource.printMultiplication = true;
                    resource.notify();
                    try {
                        Thread.sleep(300);
                    } 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,N);
        PrimeNumbers primeThread = new PrimeNumbers(resource, N);
        tableThread.start();
        primeThread.start();
        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: 7
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
Prime: 13
7 x 5 = 35
Prime: 17
Multithreading demonstration completed.
```