DEPARTMENT OF COMPUTER APPLICATION

TKM COLLEGE OF ENGINEERING

KOLLAM – 691005

23MCAP204 – Object Oriented Programming (JAVA)

LAB RECORD

Second Semester MCA

2024-2025

Submitted by

FATHIMA SHEEN K S

TKM24MCA-2034

DEPARTMENT OF COMPUTER APPLICATION

TKM COLLEGE OF ENGINEERING

KOLLAM – 691005

Certificate

This is a bonafide record of the work done by FATHIMA SHEEN K S (TKM24MCA-2034) in the Second Semester for the lab course Object Oriented Programming(JAVA) (23MCAP204) towards the partial fulfillment of the degree of Master of Computer Applications during the academic year 2024-2025.

Staff Member in-charge   Examiner

…………………………………………   ………………………………………..

INDEX

20.	Create an interface Shape with an abstract method calculateArea().		
21.	Write a Java program that throws an exception if the password is less than 8 characters or does not contain a number and do not have at least one special character{$#&).		
22.	Write a Java program that throws an exception if the phone number does not have exactly 10 digits.		
23.	Write a Java program that checks if a student's grade is valid.		
24.	Define 2 classes; one for generating Fibonacci numbers and other for displaying even numbers in a given range. Implement using threads.		
25.	Implement Producer/Consumer using ITC.		

1. Write a program to find the area and perimeter of a circle.

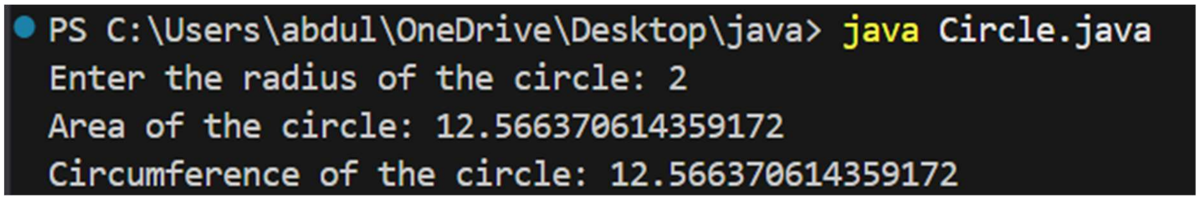
ALGORITHM:

```
I. Start
II. Input the radius of the circle.
III. Calculate the area using the formula: Area = π × radius × radius
IV. Calculate the perimeter (circumference) using the formula:
        Perimeter = 2 × π × radius
V. Display the area and perimeter.
VI. End
```

CODE:

```java
import java.util.Scanner;
public class Circle {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the radius of the circle: ");
      double radius = scanner.nextDouble();
      double area = Math.PI * radius * radius;
      double perimeter = 2 * Math.PI * radius;
      System.out.println("Area of the circle: " + area);
      System.out.println("Circumference of the circle: " + perimeter);
      scanner.close();
   }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java Circle.java
Enter the radius of the circle: 2
Area of the circle: 12.566370614359172
Circumference of the circle: 12.566370614359172
```

2. Write a program to find the area and perimeter of a rectangle.

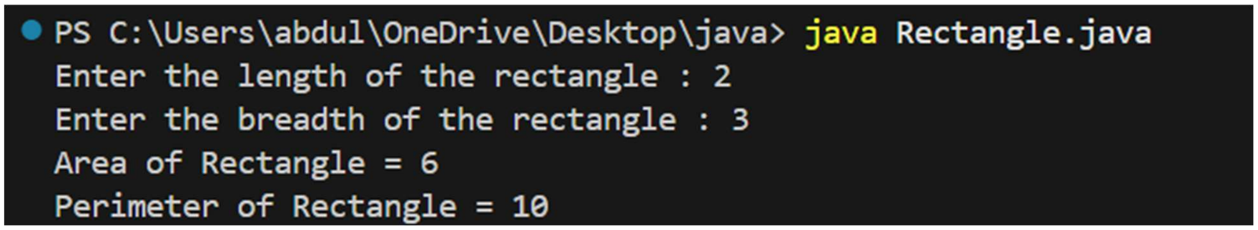
ALGORITHM:

```
I. Start
II. Input the length of the rectangle.
III. Input the breadth (width) of the rectangle.
IV. Calculate the area using the formula: Area = length × breadth
V. Calculate the perimeter using the formula: Perimeter = 2 × (length + breadth)
VI. Display the area and perimeter.
VII. End
```

CODE:

```
import java.util.Scanner;
class Rectangle{
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the length of the rectangle : ");
      int l=scanner.nextInt();
      System.out.print("Enter the breadth of the rectangle : ");
      int b=scanner.nextInt();
      int area=l*b;
      int peri=2*(l+b);
      System.out.println("Area of Rectangle = "+area);
      System.out.print("Perimeter of Rectangle = "+peri);
   }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java Rectangle.java
Enter the length of the rectangle : 2
Enter the breadth of the rectangle : 3
Area of Rectangle = 6
Perimeter of Rectangle = 10
```

3. Write a program to find eligibility of a student :
   a) Input marks for Math, Physics, and Chemistry.
   b) Check eligibility based on:
      Total marks of Math, Physics, and Chemistry.
      1. Math >= 60 and physics >=50 and chemistry>=40 and total >=200
      2. Total marks of Math and Physics >= 150

ALGORITHM:

I. Start

II. Input marks for Math.

III. Input marks for Physics.

IV. Input marks for Chemistry.

V. Calculate the total marks:Total = Math + Physics + Chemistry

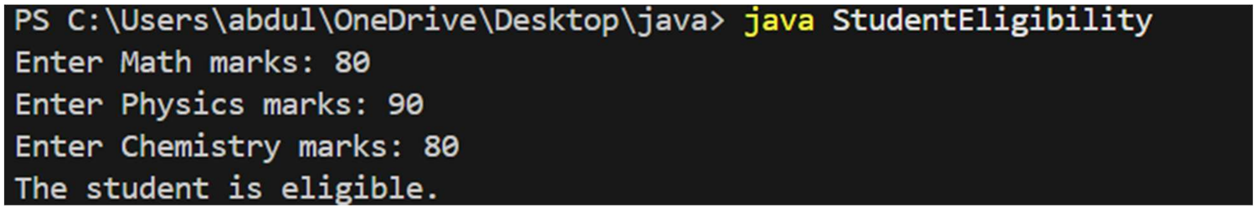
```
VI. Calculate the total of Math and Physics:
MathPhysicsTotal = Math + Physics

VII. Check the eligibility:
If (Math ≥ 60 and Physics ≥ 50 and Chemistry ≥ 40 and Total ≥ 200)
or (MathPhysicsTotal ≥ 150)
then
Display "The student is eligible."
else
Display "The student is not eligible."

VIII. End
```

CODE:

```
import java.util.Scanner;

public class StudentEligibility {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input marks

System.out.print("Enter Math marks: ");

int math = scanner.nextInt();

System.out.print("Enter Physics marks: ");

int physics = scanner.nextInt();

System.out.print("Enter Chemistry marks: ");

int chemistry = scanner.nextInt();

// Calculate totals

int total = math + physics + chemistry;

int mathPhysicsTotal = math + physics;

// Check eligibility

if ((math >= 60 && physics >= 50 && chemistry >= 40 && total >= 200)

|| (mathPhysicsTotal >= 150)) {

System.out.println("The student is eligible.");
```

```
        } else {
            System.out.println("The student is not eligible.");
        }
scanner.close();
    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java StudentEligibility
Enter Math marks: 80
Enter Physics marks: 90
Enter Chemistry marks: 80
The student is eligible.
```

4. Write a program to find salary of an employee:
   Manager = 50,000
   Developer = 30,000
   Intern =15,000

   b) Apply experience-based bonus:
   3 to 5 years = 10% bonus
   More than 5 years = 20% bonus
   Less than 3 years = No bonus

ALGORITHM:

I. Start

II. Input the role of the employee (Manager, Developer, or Intern).

III. Input the years of experience.

IV. Set the base salary according to the role:
If role is Manager, salary = 50000
If role is Developer, salary = 30000
If role is Intern, salary = 15000

V. Apply experience-based bonus:
If experience is between 3 and 5 years (inclusive), bonus = 10% of salary
If experience is more than 5 years, bonus = 20% of salary
If experience is less than 3 years, bonus = 0

VI. Calculate the total salary:
Total salary = salary + bonus

VII. Display the total salary.

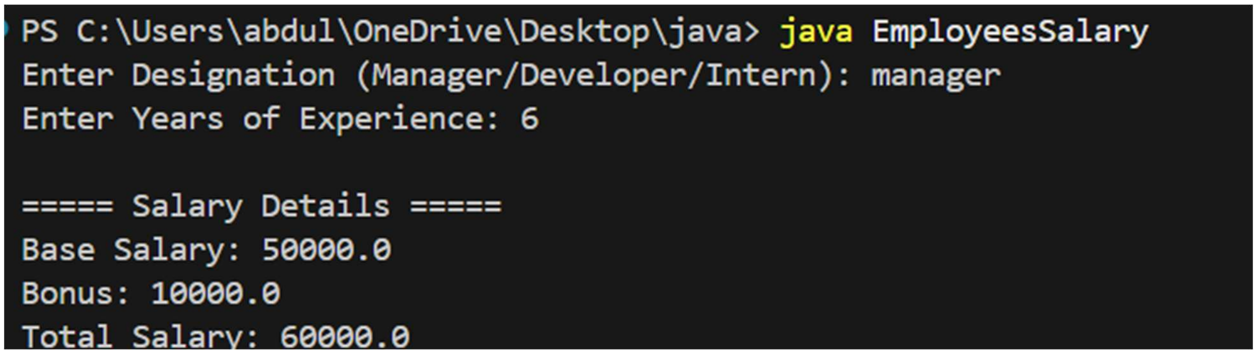
VIII. End

CODE:

```
import java.util.Scanner;

public class EmployeesSalary {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter Designation (Manager/Developer/Intern): ");

String designation = scanner.nextLine().toLowerCase();

System.out.print("Enter Years of Experience: ");

int experience = scanner.nextInt();

double salary = 0;

 double bonus = 0;

switch (designation) {

case "manager":

   salary = 50000;

   break;

case "developer":

   salary = 30000;

   break;

case "intern":

   salary = 15000;

   break;

default:

   System.out.println("Invalid Designation!");
```

```
            return;
        }
        if (experience >= 3 && experience <= 5) {
            bonus = 0.10 * salary;
        } else if (experience > 5) {
            bonus = 0.20 * salary;        }
        double totalSalary = salary + bonus;
        System.out.println("\n===== Salary Details =====");
        System.out.println("Base Salary: " + salary);
        System.out.println("Bonus: " + bonus);
        System.out.println("Total Salary: " + totalSalary);
        scanner.close();
    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java EmployeesSalary
Enter Designation (Manager/Developer/Intern): manager
Enter Years of Experience: 6

===== Salary Details =====
Base Salary: 50000.0
Bonus: 10000.0
Total Salary: 60000.0
```

5. Write a program to display grades
    A (90% and above) = "Excellent work!"
    B (80% to 89%) = "Good job!"
    C (70% to 79%) = "You can do better"
    D (60% to 69%) = "Work harder"
    F (Below 60%) = "Failed. Try again"

ALGORITHM:

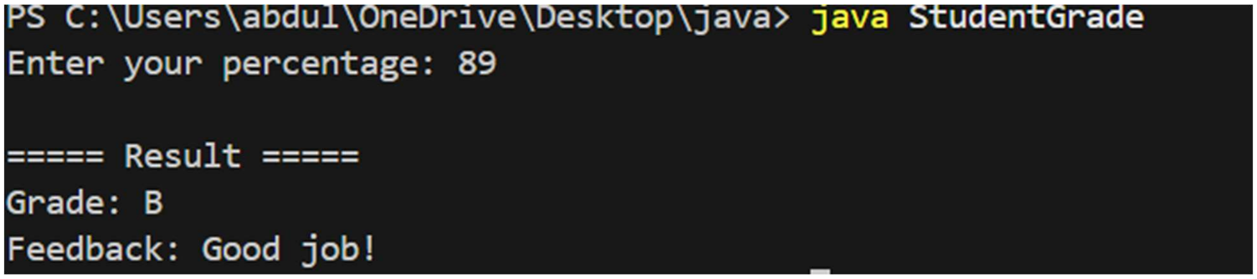
```
I. Start
II. Input the percentage of the student.
III. Check the grade according to the percentage:
If percentage ≥ 90
Display "Grade A: Excellent work!"
Else if percentage is between 80 and 89
Display "Grade B: Good job!"
Else if percentage is between 70 and 79
Display "Grade C: You can do better"
Else if percentage is between 60 and 69
Display "Grade D: Work harder"
Else
Display "Grade F: Failed. Try again"
IV. End
```

CODE:

```
import java.util.Scanner;

public class StudentGrade {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter your percentage: ");

 double percentage = scanner.nextDouble();

char grade;

String message;

if (percentage >= 90) {

grade = 'A';

message = "Excellent work!";

} else if (percentage >= 80) {

grade = 'B';

message = "Good job!";
```

```
        } else if (percentage >= 70) {
            grade = 'C';
            message = "You can do better.";
        } else if (percentage >= 60) {
            grade = 'D';
            message = "Work harder.";
        } else {
            grade = 'F';
            message = "Failed. Try again.";
        }
        System.out.println("\n===== Result =====");
        System.out.println("Grade: " + grade);
        System.out.println("Feedback: " + message);
        scanner.close();
    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java StudentGrade
Enter your percentage: 89

===== Result =====
Grade: B
Feedback: Good job!
```

6. Write a program to check if a number is a palindrome.

ALGORITHM:

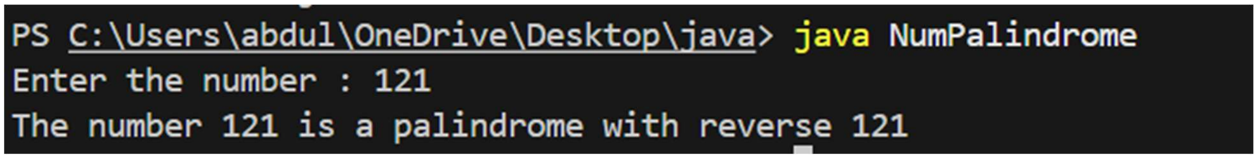
```
I. Start
II. Input the number.
III. Store the original number in a temporary variable.
IV. Reverse the number:
a. Initialize reverse = 0
b. Repeat while number > 0:
```

```
i. Get the last digit (remainder) by number % 10
ii. Multiply reverse by 10 and add the remainder
iii. Divide the number by 10
V. Compare the reversed number with the original number:
If they are equal
Display "The number
VI. End
```

CODE:

```java
import java.util.Scanner;
class NumPalindrome{
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      int rev=0;
      System.out.print("Enter the number : ");
      int num=scanner.nextInt();
      int numo=num;
      while(num!=0){
         int rem=num%10;
         rev=(rev*10)+rem;
         num = num/10;
      }
     if(rev == numo){
      System.out.print("The number "+numo+" is a palindrome with reverse "+rev);
     }
     else{
      System.out.print("The number "+numo+" is a not palindrome with reverse "+rev);
     }
   }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java NumPalindrome
Enter the number : 121
The number 121 is a palindrome with reverse 121
```

7. Write a program to check if a number is an Armstrong number.

ALGORITHM:

```
I. Start
II. Input the number.
III. Store the original number in a temporary variable.
IV. Find the number of digits in the number (let this be n).
```

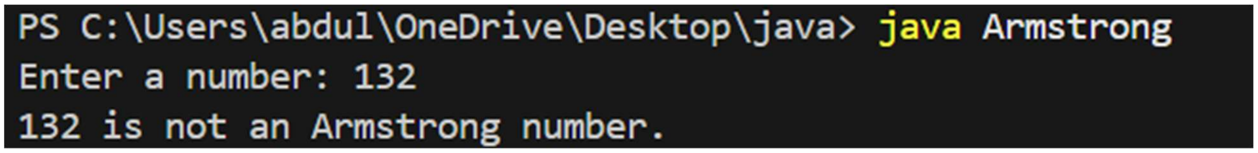
```
V. Initialize a variable sum to 0.
VI. Repeat while the number is greater than 0:
a. Find the last digit of the number using:
lastDigit = number % 10
b. Calculate the power of the last digit raised to n:
power = lastDigit^n
c. Add the result to sum:
sum = sum + power
d. Remove the last digit from the number:
number = number / 10
VII. Compare the sum with the original number:
If sum equals the original number
Display "The number is an Armstrong number."
Else
Display "The number is not an Armstrong number."
VIII. End
```

CODE:

```java
import java.util.Scanner;
class Armstrong {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter a number: ");
      int number = scanner.nextInt();
      int originalNum = number, sum = 0, digits = 0;
      // Counting the number of digits
      while (number > 0) {
         number=number/10;
         digits++;
      }
      number=originalNum;
      while (number > 0) {
         int digit = number % 10;
         sum += Math.pow(digit, digits);
         number =number/10;
      }
      if (sum == originalNum) {
         System.out.println(originalNum + " is an Armstrong number.");
      } else {
         System.out.println(originalNum + " is not an Armstrong number.");
      }
```

```
        scanner.close();
    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java Armstrong
Enter a number: 132
132 is not an Armstrong number.
```

8. Print the Fibonacci sequence up to a given limit.

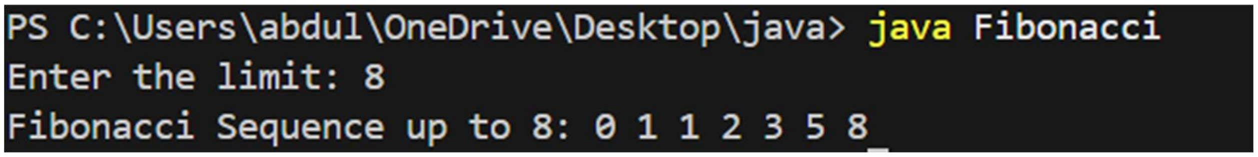
ALGORITHM:

```
I. Start
II. Input the limit (the maximum number up to which the Fibonacci sequence should be printed).
III. Initialize the first two numbers in the Fibonacci sequence:
a. Set first = 0
b. Set second = 1
IV. Print the first number (first).
V. Print the second number (second).
VI. Initialize a variable next to 0.
VII. Repeat the following steps while next is less than or equal to the limit:
a. Set next = first + second
b. If next is less than or equal to the limit, print next.
c. Update first = second and second = next.
VIII. End
```

CODE:

```
import java.util.Scanner;
class Fibonacci{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the limit: ");
        int limit = scanner.nextInt();
        int f1 = 0, f2 = 1;
        System.out.print("Fibonacci Sequence up to " + limit + ": ");
        while (f1 <= limit) {
            System.out.print(f1 + " ");
            int f3 = f1 + f2;
            f1 = f2;
            f2 = f3;
        }       scanner.close();    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java Fibonacci
Enter the limit: 8
Fibonacci Sequence up to 8: 0 1 1 2 3 5 8
```

9. Create a BankAccount class with the following:
   Fields:
   accountNumber (String), accountHolder (String), balance (double)
   A constructor to initialize all fields.
   Methods:
   deposit(double amount): Adds the amount to the balance.
   withdraw(double amount): Deducts the amount if sufficient funds exist; otherwise, prints an error message.
   displayAccountInfo(): Displays account details.
   Test:
   Create a bank account object and perform deposit, withdrawal, and display operations.
   ALGORITHM:

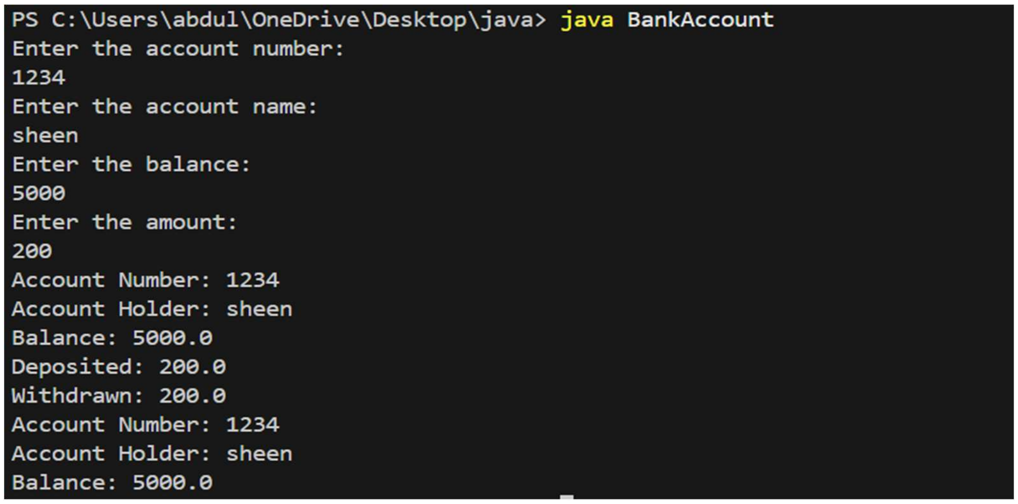
```
I. Start
II. Define the BankAccount class with the following fields:
a. accountNumber (String)
b. accountHolder (String)
c. balance (double)
III. Create a constructor that initializes all fields (accountNumber, accountHolder,
and balance).
IV. Create the methods:
a. deposit(double amount):
i. Add the amount to the balance.
b. withdraw(double amount):
i. Check if the balance is greater than or equal to the amount to withdraw.
ii. If yes, deduct the amount from the balance.
iii. If no, print an error message: "Insufficient funds."
c. displayAccountInfo():
i. Display the account number, account holder, and current balance.
V. In the main method:
a. Create an instance of the BankAccount class with initial account details.
b. Perform the following operations:
i. Call deposit() to add funds to the account.
ii. Call withdraw() to deduct funds from the account.
iii. Call displayAccountInfo() to show the account details.
```

VI. End

CODE:

```
import java.util.Scanner;
 class BankAccount {
     String accountNumber;
     String accountHolder;
     double balance;
    BankAccount(String accountNumber, String accountHolder, double balance) {
        this.accountNumber = accountNumber;
        this.accountHolder = accountHolder;
        this.balance = balance;
    }
     void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: " + amount);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }
    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient funds or invalid amount.");
        }
    }
    public void displayAccountInfo() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Account Holder: " + accountHolder);
        System.out.println("Balance: " + balance);
    }
    public static void main(String[] args) {
        Scanner scanner=new Scanner(System.in);
        System.out.println("Enter the account number:");
        String actnum = scanner.nextLine();
        System.out.println("Enter the account name:");
        String acthold = scanner.nextLine();
        System.out.println("Enter the balance:");
        double bal = scanner.nextDouble();
        System.out.println("Enter the amount:");
        int amt=scanner.nextInt();
        BankAccount account = new BankAccount(actnum,acthold,bal);
        account.displayAccountInfo();
        account.deposit(amt);
        account.withdraw(amt);
        account.displayAccountInfo();
    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java BankAccount
Enter the account number:
1234
Enter the account name:
sheen
Enter the balance:
5000
Enter the amount:
200
Account Number: 1234
Account Holder: sheen
Balance: 5000.0
Deposited: 200.0
Withdrawn: 200.0
Account Number: 1234
Account Holder: sheen
Balance: 5000.0
```

10. Create a Rectangle class with:
    Fields:
    length and width (both double).
    Constructors:
    A default constructor setting length and width to 1.
    A parameterized constructor to initialize given values.
    Methods:
    calculateArea(): Returns area (length * width).
    calculatePerimeter(): Returns perimeter (2 * (length + width)).
    Test:
    Create different rectangles and print their area and perimeter.
    ALGORITHM:

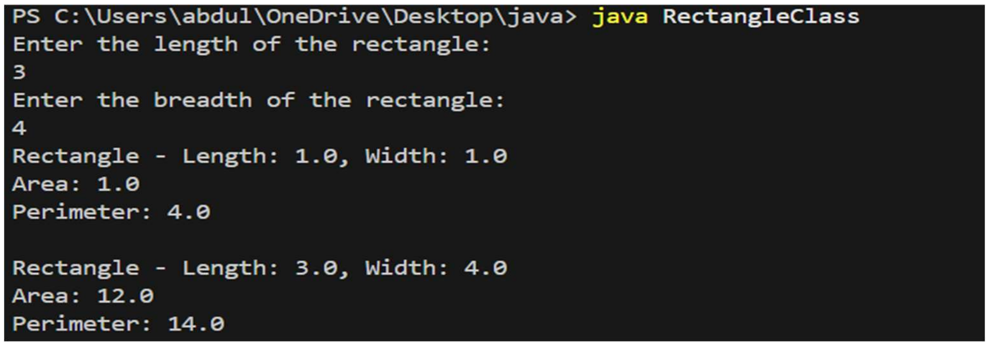
```
I. Start
II. Define the Rectangle class with the following fields:
a. length (double)
b. width (double)
```

```
III. Create constructors:
a. Default constructor:
i. Set length = 1
ii. Set width = 1
b. Parameterized constructor:
i. Initialize length and width with given values.
IV. Create the methods:
a. calculateArea():
i. Return the area calculated as length * width.
b. calculatePerimeter():
i. Return the perimeter calculated as 2 * (length + width).
V. In the main method:
a. Create different Rectangle objects:
i. One with the default constructor.
ii. One with the parameterized constructor (with specific length and width).
b. For each rectangle, perform the following operations:
i. Call calculateArea() to get the area.
ii. Call calculatePerimeter() to get the perimeter.
iii. Print the area and perimeter of the rectangle.
VI. End
```

CODE:

```java
import java.util.Scanner;
class RectangleClass {
    private double length;
    private double width;

    // Default constructor
    public RectangleClass() {
        this.length = 1.0;
        this.width = 1.0;
    }
//parameterised constructor
    public RectangleClass(double length, double width) {
        this.length = length;
        this.width = width;
    }

    public double calculateArea() {
        return length * width;
    }
    public double calculatePerimeter() {
        return 2 * (length + width);
    }

    public void display() {
        System.out.println("Rectangle - Length: " + length + ", Width: " + width);
        System.out.println("Area: " + calculateArea());
        System.out.println("Perimeter: " + calculatePerimeter());
        System.out.println();
    }

    public static void main(String[] args) {
        Scanner scanner=new Scanner(System.in);
        RectangleClass rect1 = new RectangleClass(); // Default constructor
        System.out.println("Enter the length of the rectangle:");
        double l=scanner.nextDouble();
        System.out.println("Enter the breadth of the rectangle:");
        double b=scanner.nextDouble();
        RectangleClass rect2 = new RectangleClass(l,b); // Parameterized constructor

        rect1.display();
        rect2.display();

    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java RectangleClass
Enter the length of the rectangle:
3
Enter the breadth of the rectangle:
4
Rectangle - Length: 1.0, Width: 1.0
Area: 1.0
Perimeter: 4.0

Rectangle - Length: 3.0, Width: 4.0
Area: 12.0
Perimeter: 14.0
```

11. Create a Student class with:
    Fields:
    studentId (int), name (String), marks (double)
    A constructor to initialize all fields.

Methods:
displayDetails(): Displays student details.
isPassed(): Returns true if marks are 40 or above, else false.
Test:
Create multiple students and check their pass status.

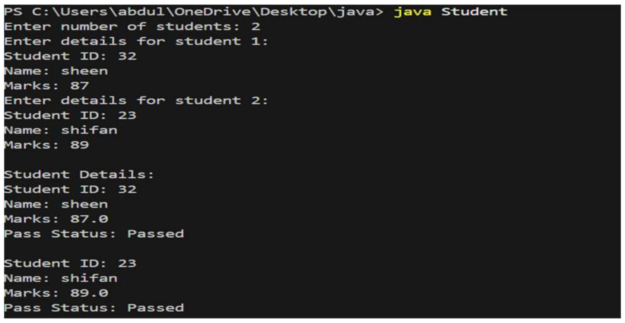
ALGORITHM:

```
I. Start
II. Define the Student class with the following fields:
a. studentId (int)
b. name (String)
c. marks (double)
III. Create a constructor:
a. Initialize studentId, name, and marks with the given values.
IV. Create the methods:
a. displayDetails():
i. Display the student's ID, name, and marks.
b. isPassed():
i. If marks are greater than or equal to 40, return true.
ii. Otherwise, return false.
V. In the main method:
a. Create multiple Student objects with different details.
b. For each student, perform the following operations:
i. Call displayDetails() to show student information.
ii. Call isPassed() to check pass or fail status.
iii. Display whether the student has passed or failed based on the result.
VI. End
```

CODE:

```
import java.util.Scanner;
class Student {
    private int studentId;
    private String name;
    private double marks;
    public Student(int studentId, String name, double marks) {
        this.studentId = studentId;
        this.name = name;
        this.marks = marks;
    }
    public void displayDetails() {
        System.out.println("Student ID: " + studentId);
        System.out.println("Name: " + name);
        System.out.println("Marks: " + marks);
        System.out.println("Pass Status: " + (isPassed() ? "Passed" : "Failed"));
        System.out.println();
    }
    public boolean isPassed() {
        return marks >= 40;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter number of students: ");
        int numStudents = scanner.nextInt();
        scanner.nextLine();
        Student[] students = new Student[numStudents];
        for (int i = 0; i < numStudents; i++) {
            System.out.println("Enter details for student " + (i + 1) + ":");
            System.out.print("Student ID: ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Name: ");
            String name = scanner.nextLine();
            System.out.print("Marks: ");
            double marks = scanner.nextDouble();
            students[i] = new Student(id, name, marks);
        }
        System.out.println("\nStudent Details:");
        for (int i = 0; i < students.length; i++) {
            students[i].displayDetails();
        }
        scanner.close();
    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java Student
Enter number of students: 2
Enter details for student 1:
Student ID: 32
Name: sheen
Marks: 87
Enter details for student 2:
Student ID: 23
Name: shifan
Marks: 89

Student Details:
Student ID: 32
Name: sheen
Marks: 87.0
Pass Status: Passed

Student ID: 23
Name: shifan
Marks: 89.0
Pass Status: Passed
```

12. Create an abstract class Employee with attributes name and id, and an abstract method calculateSalary(). Create two subclasses:
    - FullTimeEmployee (has monthlySalary).
    - PartTimeEmployee (has hourlyRate and hoursWorked).

ALGORITHM:

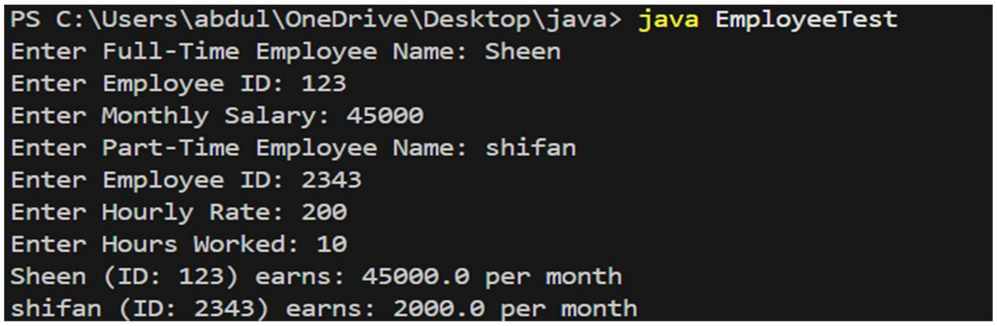
```
I. Start.
II. Define an abstract class Employee with attributes name (String) and id (int),
and an abstract method calculateSalary().
III. Create a subclass FullTimeEmployee extending Employee, with an attribute
monthlySalary (double).
a. Initialize name, id, and monthlySalary using a constructor.
b. Implement calculateSalary() to return monthlySalary.
IV. Create another subclass PartTimeEmployee extending Employee, with
attributes hourlyRate (double) and hoursWorked (int).
a. Initialize name, id, hourlyRate, and hoursWorked using a constructor.
b. Implement calculateSalary() to return hourlyRate * hoursWorked.
V. In the main method, create objects of FullTimeEmployee and
PartTimeEmployee.
a. Call calculateSalary() on both objects.
b. Display their salaries and details.
VI. End.
```

CODE:

```
import java.util.Scanner;
abstract class Employee {
    String name;
    int id;
    public Employee(String name, int id) {
        this.name = name;
        this.id = id;
    }
    public abstract double calculateSalary();
}
class FullTimeEmployee extends Employee {
    private double monthlySalary;
    public FullTimeEmployee(String name, int id, double monthlySalary) {
        super(name, id);
        this.monthlySalary = monthlySalary;
    }
    public double calculateSalary() {
        return monthlySalary;
    }
}
class PartTimeEmployee extends Employee {
    double hourlyRate;
    int hoursWorked;
    public PartTimeEmployee(String name, int id, double hourlyRate, int hoursWorked) {
        super(name, id);
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
    }
    public double calculateSalary() {
        return hourlyRate * hoursWorked;
    }
}
public class EmployeeTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Input for Full-Time Employee
        System.out.print("Enter Full-Time Employee Name: ");
        String fullName = scanner.nextLine();
        System.out.print("Enter Employee ID: ");
        int fullId = scanner.nextInt();
        System.out.print("Enter Monthly Salary: ");
        double monthlySalary = scanner.nextDouble();
        FullTimeEmployee fullTimeEmp = new FullTimeEmployee(fullName, fullId,
monthlySalary);
        // Input for Part-Time Employee
        scanner.nextLine();
        System.out.print("Enter Part-Time Employee Name: ");
        String partName = scanner.nextLine();
        System.out.print("Enter Employee ID: ");
        int partId = scanner.nextInt();
        System.out.print("Enter Hourly Rate: ");
        double hourlyRate = scanner.nextDouble();
        System.out.print("Enter Hours Worked: ");
        int hoursWorked = scanner.nextInt();
        PartTimeEmployee partTimeEmp = new PartTimeEmployee(partName, partId,
hourlyRate, hoursWorked);
        System.out.println(fullTimeEmp.name + " (ID: " + fullTimeEmp.id + ") earns: " +
fullTimeEmp.calculateSalary() + " per month");
```

```
        System.out.println(partTimeEmp.name + " (ID: " + partTimeEmp.id + ") earns: " +
partTimeEmp.calculateSalary() + " per month");
        scanner.close();
    }}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java EmployeeTest
Enter Full-Time Employee Name: Sheen
Enter Employee ID: 123
Enter Monthly Salary: 45000
Enter Part-Time Employee Name: shifan
Enter Employee ID: 2343
Enter Hourly Rate: 200
Enter Hours Worked: 10
Sheen (ID: 123) earns: 45000.0 per month
shifan (ID: 2343) earns: 2000.0 per month
```

13. Create a class ‘Person’ with data members Name, Gender, Address, Age and a constructor to initialize the data members and another class ‘Employee’ that inherits the properties of class Person and also contains its own data members like Empid, Company_name, Qualification, Salary and its own constructor. Create another class ‘Teacher’ that inherits the properties of class Employee and contains its own data members like Subject, Department, Teacherid and also contain constructors and methods to display the data members. Use array of objects to display details of N teachers.

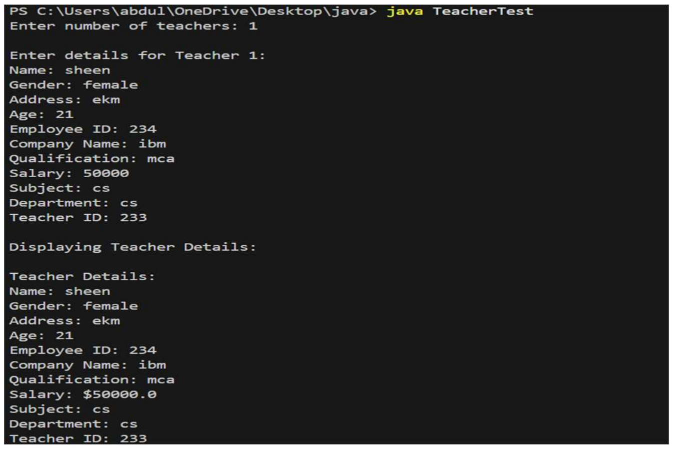
ALGORITHM:

```
I. Start
II. Define the class Person with the following fields:
a. Name (String)
b. Gender (String)
c. Address (String)
d. Age (int)
III. Create a constructor in Person to initialize all its fields.
IV. Define the class Employee that inherits from Person with additional fields:
a. Empid (int)
b. Company_name (String)
c. Qualification (String)
d. Salary (double)
V. Create a constructor in Employee that:
a. Calls the parent class (Person) constructor to initialize inherited fields.
b. Initializes Employee’s own fields.
VI. Define the class Teacher that inherits from Employee with additional fields:
a. Subject (String)
b. Department (String)
c. Teacherid (int)
```

VII. Create a constructor in Teacher that:
a. Calls the parent class (Employee) constructor to initialize inherited fields.
b. Initializes Teacher's own fields.
VIII. Create a method in Teacher called displayDetails() to display all the fields (including inherited fields).
IX. In the main method:
a. Input the number of teachers (N).
b. Create an array of Teacher objects of size N.
c. For each Teacher object:
i. Take input for all fields.
ii. Create and initialize the Teacher object using the constructor.
d. After storing all teachers, loop through the array and call displayDetails() for each teacher to display their information.
X. End

CODE:
```
import java.util.Scanner;
class Person {
    String name, gender, address;
    int age;
    public Person(String name, String gender, String address, int age) {
        this.name = name;
        this.gender = gender;
        this.address = address;
        this.age = age;
    }
}
class Employee extends Person {
    int empId;
    String companyName, qualification;
    double salary;
    public Employee(String name, String gender, String address, int age, int empId, String
companyName, String qualification, double salary) {
        super(name, gender, address, age);
        this.empId = empId;
        this.companyName = companyName;
        this.qualification = qualification;
        this.salary = salary;
    }
}
class Teacher extends Employee {
    String subject, department;
    int teacherId;
     public Teacher(String name, String gender, String address, int age, int empId, String
companyName, String qualification, double salary, String subject, String department, int
teacherId) {
        super(name, gender, address, age, empId, companyName, qualification, salary);
        this.subject = subject;
        this.department = department;
        this.teacherId = teacherId;
    }
    public void display() {
        System.out.println("\nTeacher Details:");
        System.out.println("Name: " + name);
        System.out.println("Gender: " + gender);
        System.out.println("Address: " + address);
        System.out.println("Age: " + age);
        System.out.println("Employee ID: " + empId);
        System.out.println("Company Name: " + companyName);
        System.out.println("Qualification: " + qualification);
        System.out.println("Salary: $" + salary);
        System.out.println("Subject: " + subject);
        System.out.println("Department: " + department);
        System.out.println("Teacher ID: " + teacherId);
    }
}
public class TeacherTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter number of teachers: ");
        int n = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        Teacher[] teachers = new Teacher[n];
        for (int i = 0; i < n; i++) {
            System.out.println("\nEnter details for Teacher " + (i + 1) + ":");
            System.out.print("Name: ");
            String name = scanner.nextLine();
            System.out.print("Gender: ");
            String gender = scanner.nextLine();
            System.out.print("Address: ");
            String address = scanner.nextLine();
            System.out.print("Age: ");
            int age = scanner.nextInt();
            System.out.print("Employee ID: ");
            int empId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Company Name: ");
            String companyName = scanner.nextLine();
            System.out.print("Qualification: ");
            String qualification = scanner.nextLine();
            System.out.print("Salary: ");
            double salary = scanner.nextDouble();
            scanner.nextLine(); // Consume newline
            System.out.print("Subject: ");
            String subject = scanner.nextLine();
            System.out.print("Department: ");
            String department = scanner.nextLine();
            System.out.print("Teacher ID: ");
            int teacherId = scanner.nextInt();
            scanner.nextLine(); // Consume newline
              teachers[i] = new Teacher(name, gender, address, age, empId, companyName,
qualification, salary, subject, department, teacherId);
        }
        System.out.println("\nDisplaying Teacher Details:");
        for (Teacher teacher : teachers) {
            teacher.display();
        }
        scanner.close();
    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java TeacherTest
Enter number of teachers: 1

Enter details for Teacher 1:
Name: sheen
Gender: female
Address: ekm
Age: 21
Employee ID: 234
Company Name: ibm
Qualification: mca
Salary: 50000
Subject: cs
Department: cs
Teacher ID: 233

Displaying Teacher Details:

Teacher Details:
Name: sheen
Gender: female
Address: ekm
Age: 21
Employee ID: 234
Company Name: ibm
Qualification: mca
Salary: $50000.0
Subject: cs
Department: cs
Teacher ID: 233
```

14. Check whether a matrix is symmetric.

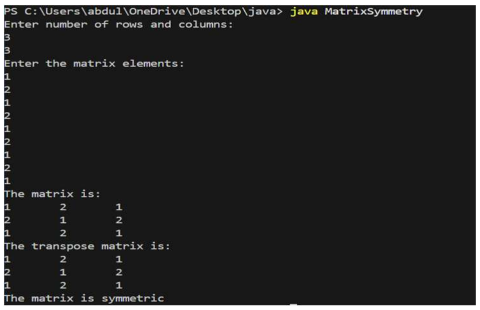
ALGORITHM:

```
I. Start
II. Input the number of rows and columns of the matrix
III. If number of rows ≠ number of columns
a) Print "Matrix is not symmetric"
b) Exit
IV. Input all elements of the matrix
V. For each element at position (i, j)
a) Compare it with element at position (j, i)
b) If any mismatch is found, set a flag to false and break
VI. If flag is true
a) Print "Matrix is symmetric"
Else
a) Print "Matrix is not symmetric"
VII. End
```

CODE:

```
import java.util.*;
class MatrixSymmetry {
   public static void main(String[] args) {
      Scanner s = new Scanner(System.in);
      int r, c;
      System.out.println("Enter number of rows and columns:");
      r = s.nextInt();
      c = s.nextInt();
      if (r != c) {
         System.out.println("The matrix is not symmetric (must be square)");
         return;
      }
      int[][] a = new int[r][c];
      int[][] b= new int[r][c];
      System.out.println("Enter the matrix elements:");
      for (int i = 0; i < r; i++) {
         for (int j = 0; j < c; j++) {
            a[i][j] = s.nextInt();
         }
      }
      System.out.println("The matrix is:");
      for (int i = 0; i < r; i++) {
         for (int j = 0; j < c; j++) {
                System.out.print(a[i][j] + "\t");
            }
            System.out.println();
        }
                for ( int i=0;i<r;i++){
            for(int j=0;j<c;j++){
                b[i][j]=a[j][i];
            }
        }
        System.out.println("The transpose matrix is:");
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                System.out.print(b[i][j] + "\t");
            }
            System.out.println();
        }
        int flag = 1;
        for (int i = 0; i < r; i++) {
            for (int j = i + 1; j < c; j++) {
                if (a[i][j] != a[j][i]) {
                    flag = 0;
                    break;
                }
            }
            if (flag == 0) break;
        }
        if (flag == 1)
            System.out.println("The matrix is symmetric");
        else
            System.out.println("The matrix is not symmetric");
    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java MatrixSymmetry
Enter number of rows and columns:
3
3
Enter the matrix elements:
1
2
1
2
1
2
1
2
1
The matrix is:
1       2       1
2       1       2
1       2       1
The transpose matrix is:
1       2       1
2       1       2
1       2       1
The matrix is symmetric
```

15. Calculate the sum of rows and columns of a matrix.

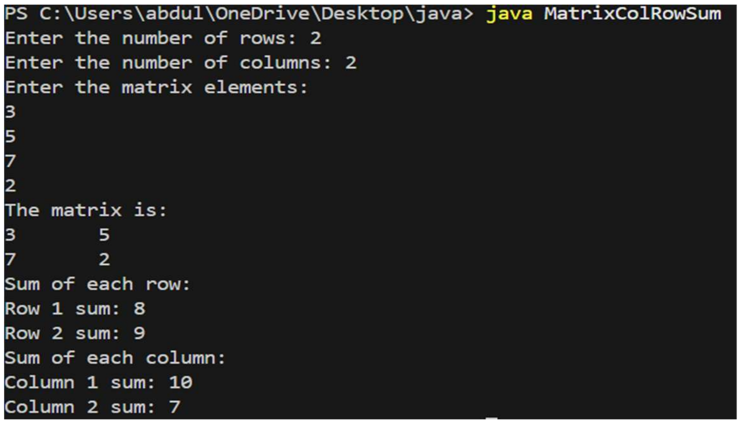
ALGORITHM:

```
I. Start
II. Input the number of rows and columns of the matrix
III. Input all elements of the matrix
IV. For each row (i from 0 to rows - 1)
a) Initialize rowSum = 0
b) For each column (j from 0 to columns - 1)
i) Add matrix[i][j] to rowSum
c) Print the sum of the current row
V. For each column (j from 0 to columns - 1)
a) Initialize colSum = 0
b) For each row (i from 0 to rows - 1)
i) Add matrix[i][j] to colSum
c) Print the sum of the current column
VI. End
```

CODE:

```
import java.util.Scanner;
class MatrixColRowSum {
    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[][] matrix = new int[rows][cols];
        System.out.println("Enter the matrix elements:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }
        System.out.println("The matrix is:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print(matrix[i][j] + "\t");
            }
            System.out.println();
        }
         System.out.println("Sum of each row:");
        for (int i = 0; i < rows; i++) {
            int rowSum = 0;
            for (int j = 0; j < cols; j++) {
                rowSum += matrix[i][j];
            }
            System.out.println("Row " + (i + 1) + " sum: " + rowSum);
        }
        System.out.println("Sum of each column:");
        for (int j = 0; j < cols; j++) {
            int colSum = 0;
            for (int i = 0; i < rows; i++) {
                colSum += matrix[i][j];
            }
            System.out.println("Column " + (j + 1) + " sum: " + colSum);
        }
        scanner.close();
    }
}
```

OUTPUT:

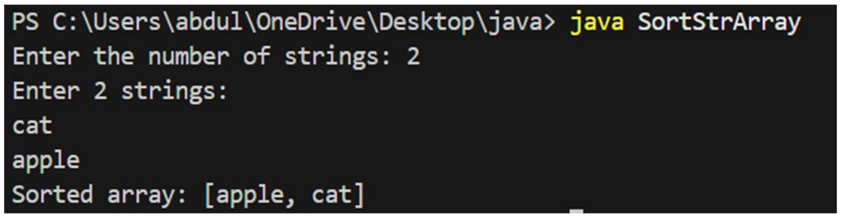


```
PS C:\Users\abdul\OneDrive\Desktop\java> java MatrixColRowSum
Enter the number of rows: 2
Enter the number of columns: 2
Enter the matrix elements:
3
5
7
2
The matrix is:
3       5
7       2
Sum of each row:
Row 1 sum: 8
Row 2 sum: 9
Sum of each column:
Column 1 sum: 10
Column 2 sum: 7
```

16. Sort an array of strings.

ALGORITHM:

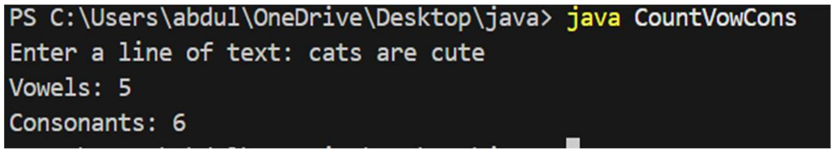
```
I. Start
II. Input the number of strings (n)
III. Input all the strings into an array
IV. For i = 0 to n-2
a) For j = i+1 to n-1
i) If array[i] > array[j] (compare alphabetically)
- Swap array[i] and array[j]
V. Display the sorted array
VI. End
```

CODE:
```
import java.util.Arrays;
import java.util.Scanner;
 class SortStrArray {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of strings: ");
        int n = scanner.nextInt();
        scanner.nextLine();
        String[] words = new String[n];
        System.out.println("Enter " + n + " strings:");
        for (int i = 0; i < n; i++) {
            words[i] = scanner.nextLine();
        }
        Arrays.sort(words);
        System.out.println("Sorted array: " + Arrays.toString(words));
        scanner.close();
    }
}
```
OUTPUT:

```
PS C:\Users\abdul\OneDrive\Desktop\java> java SortStrArray
Enter the number of strings: 2
Enter 2 strings:
cat
apple
Sorted array: [apple, cat]
```
17. Count the number of vowels and consonants in a line of text.

ALGORITHM:
```
I. Start
II. Input a line of text
III. Initialize vowelCount and consonantCount to 0
IV. Convert the text to lowercase
V. For each character in the text:
a) If the character is a letter:
i) If it is a vowel (a, e, i, o, u), increment vowelCount
ii) Else, increment consonantCount
VI. Display vowelCount and consonantCount
VII. End
```
CODE:
```
 import java.util.Scanner;
```

```
public class CountVowCons {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter a line of text: ");
      String text = scanner.nextLine().toLowerCase();
      int vowels = 0, consonants = 0;
      for (char ch : text.toCharArray()) {
         if ("aeiou".indexOf(ch) != -1) {
            vowels++;
         } else if (Character.isLetter(ch)) {
            consonants++;
         }
      }
      System.out.println("Vowels: " + vowels);
      System.out.println("Consonants: " + consonants);
      scanner.close();
   }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java CountVowCons
Enter a line of text: cats are cute
Vowels: 5
Consonants: 6
```

18. Check if a string is a palindrome.

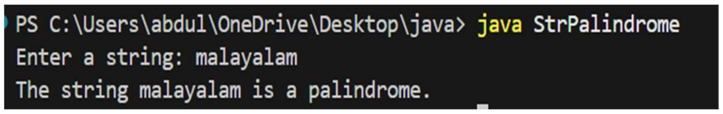
ALGORITHM:

```
I. Start
II. Input the string
III. Convert the string to lowercase (optional, to ignore case)
IV. Initialize two pointers:
a) left = 0
b) right = length of string - 1
V. While left < right:
a) If character at left is not equal to character at right:
i) Print "Not a palindrome"
ii) Exit
b) Increment left by 1
c) Decrement right by 1
VI. If the loop completes without mismatch, print "It is a palindrome"
VII. End
```

CODE:

```
import java.util.Scanner;
 class StrPalindrome {
   public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);
       System.out.print("Enter a string: ");
       String str = scanner.nextLine();
       int length = str.length();
       int flag = 1;
       for (int i = 0; i < length / 2; i++) {
           if (str.charAt(i) != str.charAt(length - i - 1)) {
               flag = 0;
               break;
           }
       }
       if (flag == 1) {
           System.out.println("The string "+str+" is a palindrome.");
       } else {
           System.out.println("The string "+str+" is not a palindrome.");
       }
       scanner.close();
   }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java StrPalindrome
Enter a string: malayalam
The string malayalam is a palindrome.
```

19. Search for a pattern in a string and replace it with another string.

ALGORITHM:

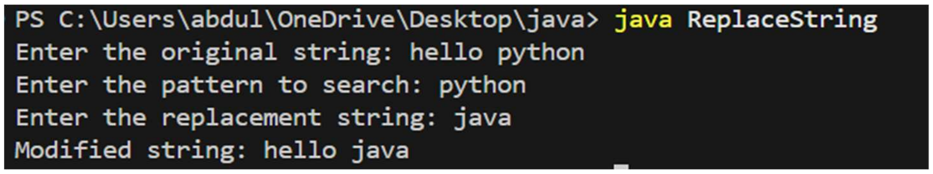
```
I. Start
II. Input the string (text), the pattern to search for, and the replacement string
III. Use a function or method to search for the pattern in the text
IV. If the pattern is found:
a) Replace the found pattern with the replacement string
b) Print the modified string
V. If the pattern is not found, print "Pattern not found"
VI. End
```

CODE:

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

```
class ReplaceString {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the original string: ");
      String original = scanner.nextLine();
      System.out.print("Enter the pattern to search: ");
      String pattern = scanner.nextLine();
      System.out.print("Enter the replacement string: ");
      String replacement = scanner.nextLine();
      String modifiedString = original.replace(pattern, replacement);
      System.out.println("Modified string: " + modifiedString);
      scanner.close();
   }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java ReplaceString
Enter the original string: hello python
Enter the pattern to search: python
Enter the replacement string: java
Modified string: hello java
```

20. Create an interface Shape with an abstract method calculateArea(). Implement Shape in Triangle and Rectangle classes. Use constructors in Triangle and Rectangle to initialize dimensions. Override calculateArea() in both classes to compute their respective areas. Demonstrate polymorphism by calling calculateArea() on Shape references.

ALGORITHM:

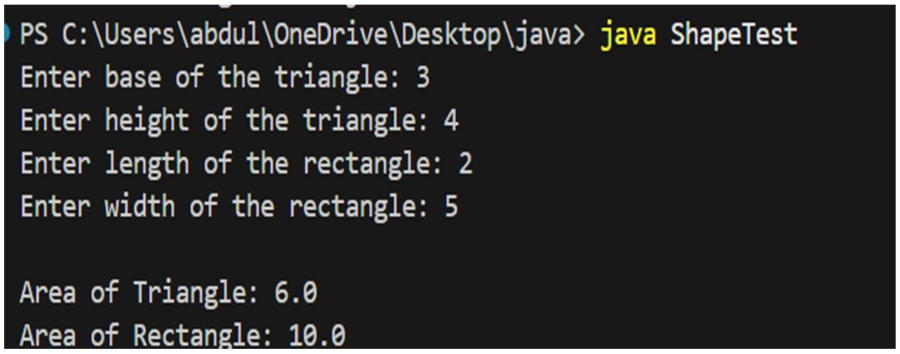
```
I. Start
II. Define an interface Shape
a) Declare an abstract method calculateArea() in the interface.
III. Define the Triangle class:
a) Implement the Shape interface
b) Declare fields for the dimensions (e.g., base, height)
c) Create a constructor to initialize the dimensions
d) Override the calculateArea() method to compute the area of the triangle.
IV. Define the Rectangle class:
a) Implement the Shape interface
b) Declare fields for the dimensions (e.g., length, width)
c) Create a constructor to initialize the dimensions
d) Override the calculateArea() method to compute the area of the rectangle.
V. In the main program:
a) Create references of type Shape
b) Instantiate objects of Triangle and Rectangle
```

c) Use polymorphism to call calculateArea() on the Shape references
i) Print the area of both shapes.
VI. End

CODE:

```
import java.util.Scanner;
interface Shape {
    double calculateArea();
}
class Triangle implements Shape {
    private double base, height;
    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }
    public double calculateArea() {
        return 0.5 * base * height;
    }
}
class Rectangle implements Shape {
    private double length, width;
    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }
    public double calculateArea() {
        return length * width;
    }
}
public class ShapeTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter base of the triangle: ");
        double base = scanner.nextDouble();
        System.out.print("Enter height of the triangle: ");
        double height = scanner.nextDouble();
        Shape triangle = new Triangle(base, height);
        System.out.print("Enter length of the rectangle: ");
        double length = scanner.nextDouble();
        System.out.print("Enter width of the rectangle: ");
        double width = scanner.nextDouble();
        Shape rectangle = new Rectangle(length, width);
        System.out.println("\nArea of Triangle: " + triangle.calculateArea());
        System.out.println("Area of Rectangle: " + rectangle.calculateArea());
        scanner.close();
    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java ShapeTest
Enter base of the triangle: 3
Enter height of the triangle: 4
Enter length of the rectangle: 2
Enter width of the rectangle: 5

Area of Triangle: 6.0
Area of Rectangle: 10.0
```

21. Write a Java program that throws an exception if the password is less than 8 characters or does not contain a number and do not have at least one special character{$#&).

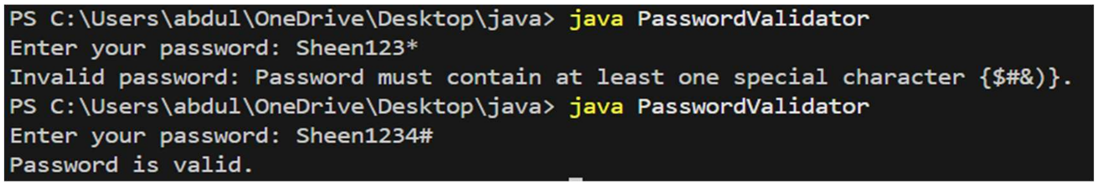
ALGORITHM:

```
I. Start
II. Input the password from the user
III. Check the length of the password:
a) If the length is less than 8 characters, throw an exception with an appropriate error message.
IV. Check if the password contains at least one number:
a) If no number is found, throw an exception with an appropriate error message.
V. Check if the password contains at least one special character from the set {$#&}:
a) If no special character is found, throw an exception with an appropriate error message.
VI. If all checks pass, print "Password is valid"
VII. Catch the exceptions and display the error messages for invalid password criteria.
VIII. End
```

CODE:

```
import java.util.Scanner;
class InvalidPasswordException extends Exception {
    public InvalidPasswordException(String message) {
        super(message);
    }
}
public class PasswordValidator {
   public static void validatePassword(String password) throws
InvalidPasswordException {
      if (password.length() < 8) {
         throw new InvalidPasswordException("Password must be at least 8 characters
long.");
      }
      if (!password.matches(".*\\d.*")) {
         throw new InvalidPasswordException("Password must contain at least one
number.");
      }
      if (!password.matches(".*[#$&)].*")) {
         throw new InvalidPasswordException("Password must contain at least one special
character {$#&)}.");
      }
   }
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter your password: ");
      String password = scanner.nextLine();
      try {
         validatePassword(password);
         System.out.println("Password is valid.");
      } catch (InvalidPasswordException e) {
         System.out.println("Invalid password: " + e.getMessage());
      }
      scanner.close();
   }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java PasswordValidator
Enter your password: Sheen123*
Invalid password: Password must contain at least one special character {$#&)}.
PS C:\Users\abdul\OneDrive\Desktop\java> java PasswordValidator
Enter your password: Sheen1234#
Password is valid.
```

22. Write a Java program that throws an exception if the phone number does not have exactly 10 digits.

ALGORITHM:

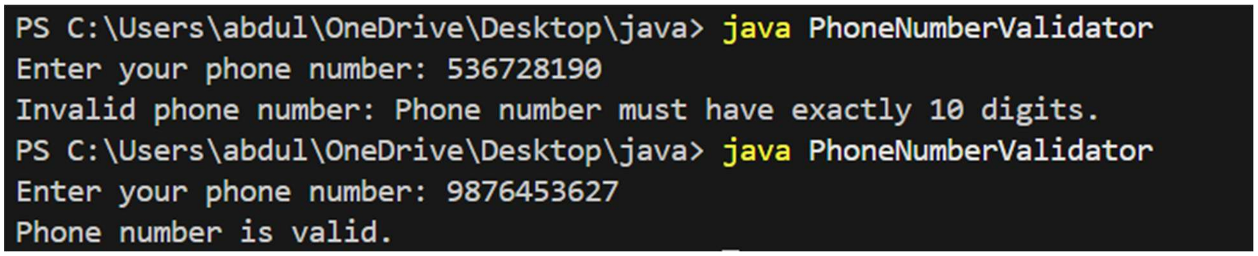
I. Start

```
II. Input the phone number as a string from the user
III. Check if the phone number contains exactly 10 digits:
a) If the length of the phone number is not 10, throw an exception with an
appropriate error message.
IV. Check if all characters in the phone number are digits:
a) If any character is not a digit, throw an exception with an appropriate error
message.
V. If both conditions pass, print "Phone number is valid"
VI. Catch the exception and display the error message if the conditions are not
met.
VII. End
```

CODE:

```
 import java.util.Scanner;
class InvalidPhoneNumberException extends Exception {
   public InvalidPhoneNumberException(String message) {
      super(message);
   }
}
public class PhoneNumberValidator {
   public static void validatePhoneNumber(String phoneNumber) throws
InvalidPhoneNumberException {
      if (!phoneNumber.matches("\\d{10}")) {
         throw new InvalidPhoneNumberException("Phone number must have exactly 10
digits.");
      }
   }
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter your phone number: ");
      String phoneNumber = scanner.nextLine();
      try {
         validatePhoneNumber(phoneNumber);
         System.out.println("Phone number is valid.");
      } catch (InvalidPhoneNumberException e) {
         System.out.println("Invalid phone number: " + e.getMessage());
      }
      scanner.close();
   }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java PhoneNumberValidator
Enter your phone number: 536728190
Invalid phone number: Phone number must have exactly 10 digits.
PS C:\Users\abdul\OneDrive\Desktop\java> java PhoneNumberValidator
Enter your phone number: 9876453627
Phone number is valid.
```

23. Write a Java program that checks if a student's grade is valid. If the grade is not between 'A' and 'F', throw a custom exception InvalidGradeException.

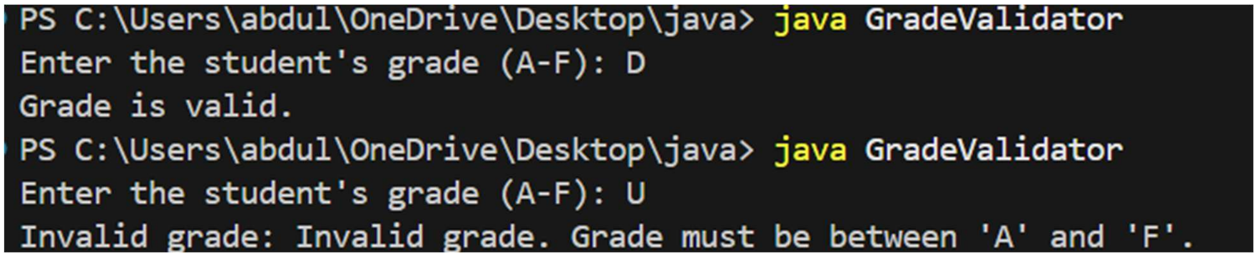
ALGORITHM:

```
I. Start
II. Define a custom exception class InvalidGradeException that extends the Exception class.
a) Provide a constructor to accept a custom error message.
III. Input the grade from the user.
IV. Check if the grade is between 'A' and 'F':
a) If the grade is not in the range 'A' to 'F', throw the custom exception InvalidGradeException.
V. If the grade is valid, print "Valid grade".
VI. Catch the InvalidGradeException and display the error message (e.g., "Invalid grade. Please enter a grade between 'A' and 'F'").
VII. End
```

CODE:

```
import java.util.Scanner;
class InvalidGradeException extends Exception {
    public InvalidGradeException(String message) {
        super(message);
    }
}
public class GradeValidator {
    public static void validateGrade(char grade) throws InvalidGradeException {
        if (grade < 'A' || grade > 'F') {
            throw new InvalidGradeException("Invalid grade. Grade must be between 'A' and
'F'.");
        }
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the student's grade (A-F): ");
        char grade = scanner.next().charAt(0);
        try {
            validateGrade(grade);
            System.out.println("Grade is valid.");
        } catch (InvalidGradeException e) {
            System.out.println("Invalid grade: " + e.getMessage());
        }
        scanner.close();
    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java GradeValidator
Enter the student's grade (A-F): D
Grade is valid.
PS C:\Users\abdul\OneDrive\Desktop\java> java GradeValidator
Enter the student's grade (A-F): U
Invalid grade: Invalid grade. Grade must be between 'A' and 'F'.
```

24. Define 2 classes; one for generating Fibonacci numbers and other for displaying even numbers in a given range. Implement using threads

ALGORITHM:

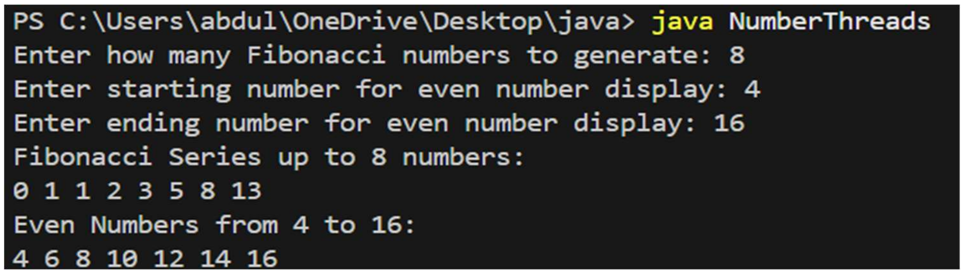
```
I. Start
II. Define the class FibonacciGenerator:
a) Implement a method generateFibonacci() to generate Fibonacci numbers up to
a certain limit.
b) Use a loop to print Fibonacci numbers.
III. Define the class EvenNumberPrinter:
a) Implement a method printEvenNumbers() to print even numbers in a given
range.
b) Use a loop to print even numbers between a start and end value.
IV. Create a class Main to execute the program.
a) In the main() method, create objects of FibonacciGenerator and
EvenNumberPrinter.
b) Create threads to execute both generateFibonacci() and printEvenNumbers()
concurrently.
V. Start both threads:
a) Use Thread.start() to initiate both the FibonacciGenerator and
EvenNumberPrinter threads.
```

```
VI. Wait for both threads to finish:
a) Use Thread.join() to ensure that the main program waits until both threads
finish executing.
VII. End
```

CODE:

```java
import java.util.Scanner;
// Class to generate Fibonacci numbers
class FibonacciGenerator extends Thread {
    private int count;
    public FibonacciGenerator(int count) {
        this.count = count;
    }
    @Override
    public void run() {
        int a = 0, b = 1;
        System.out.println("Fibonacci Series up to " + count + " numbers:");
        for (int i = 0; i < count; i++) {
            System.out.print(a + " ");
            int next = a + b;
            a = b;
            b = next;
        }
        System.out.println();
    }
}
// Class to display even numbers within a range
class EvenNumberDisplayer extends Thread {
    private int start, end;
    public EvenNumberDisplayer(int start, int end) {
        this.start = start;
        this.end = end;
    }
    @Override
    public void run() {
        System.out.println("Even Numbers from " + start + " to " + end + ":");
        for (int i = start; i <= end; i++) {
            if (i % 2 == 0) {
                System.out.print(i + " ");
            }
        }
        System.out.println();
    }
}
// Main class
public class NumberThreads {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Input for Fibonacci
        System.out.print("Enter how many Fibonacci numbers to generate: ");
        int fibCount = scanner.nextInt();
        // Input for Even number range
        System.out.print("Enter starting number for even number display: ");
        int start = scanner.nextInt();
        System.out.print("Enter ending number for even number display: ");
        int end = scanner.nextInt();
        // Create objects with user input
        FibonacciGenerator fibThread = new FibonacciGenerator(fibCount);
        EvenNumberDisplayer evenThread = new EvenNumberDisplayer(start, end);
        // Start both threads
        fibThread.start();
        evenThread.start();
        scanner.close();
    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java NumberThreads
Enter how many Fibonacci numbers to generate: 8
Enter starting number for even number display: 4
Enter ending number for even number display: 16
Fibonacci Series up to 8 numbers:
0 1 1 2 3 5 8 13
Even Numbers from 4 to 16:
4 6 8 10 12 14 16
```

25. Implement Producer/Consumer using ITC

ALGORITHM:

I. Start

II. Define the class SharedBuffer: a) Create a shared buffer (e.g., a Queue or an array) that both the producer and consumer will use. b) Implement methods for adding (producing) and removing (consuming) items from the buffer. c) Use wait() and notify() for synchronization:

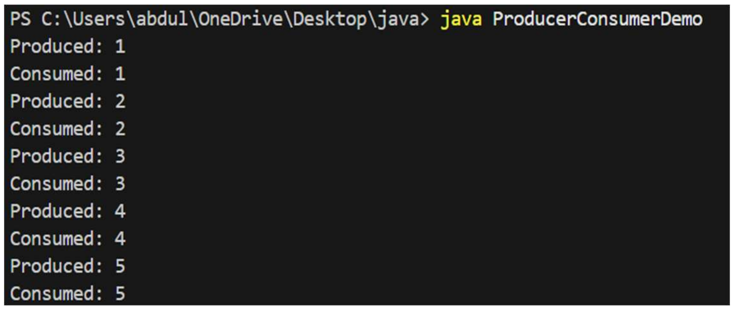
i) wait() to pause a thread if the buffer is full or empty.

ii) notify() to wake up a thread when space is available or when new items are added.
III. Define the class Producer that implements Runnable: a) Create a method produce() to add items to the buffer.
b) Use synchronized block to ensure thread safety while adding items.
c) If the buffer is full, call wait() to pause the thread until the consumer consumes an item.
IV. Define the class Consumer that implements Runnable: a) Create a method consume() to remove items from the buffer.
b) Use synchronized block to ensure thread safety while removing items.
c) If the buffer is empty, call wait() to pause the thread until the producer produces an item.
V. In the main() method, initialize the shared buffer, and create threads for the producer and consumer: a) Create an instance of SharedBuffer. b) Create instances of Producer and Consumer with the shared buffer as an argument. c) Start both threads using Thread.start().
VI. Run the producer and consumer threads: a) The producer thread will produce items and add them to the shared buffer. b) The consumer thread will consume items from the shared buffer.
VII. Ensure that the producer and consumer threads communicate and synchronize properly using wait() and notify().
VIII. End

CODE:

```
// Shared class for Producer and Consumer
class SharedData {
    int data;
    boolean isProduced = false;
    // Producer method
    public synchronized void produce(int value) {
        while (isProduced) {
            try {
                wait(); // wait if data is already produced
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        data = value;
        isProduced = true;
        System.out.println("Produced: " + data);
        notify(); // notify the consumer
    }
    // Consumer method
    public synchronized void consume() {
        while (!isProduced) {
            try {
                wait(); // wait if no data produced yet
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println("Consumed: " + data);
        isProduced = false;
        notify(); // notify the producer
    }
}
// Producer class
class Producer extends Thread {
    SharedData shared;
    public Producer(SharedData shared) {
        this.shared = shared;
    }
    @Override
    public void run() {
        int value = 1;
        for (int i = 0; i < 5; i++) { // producing 5 items
            shared.produce(value++);
            try {
                Thread.sleep(500); // simulate some delay
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
// Consumer class
class Consumer extends Thread {
    SharedData shared;
    public Consumer(SharedData shared) {
        this.shared = shared;
    }
    @Override
    public void run() {
        for (int i = 0; i < 5; i++) { // consuming 5 items
            shared.consume();
            try {
                Thread.sleep(500); // simulate some delay
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
// Main class
public class ProducerConsumerDemo {
    public static void main(String[] args) {
        SharedData shared = new SharedData();
        Producer producer = new Producer(shared);
        Consumer consumer = new Consumer(shared);
        producer.start();
        consumer.start();
    }
}
```

OUTPUT:



```
PS C:\Users\abdul\OneDrive\Desktop\java> java ProducerConsumerDemo
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5
```