Index

Practical No 1. wap in java to take input from user and print 2 by 3 matrix.

```
import java.util.Scanner;
public class Matrix
{
   public static void main(String[] args)
   {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the Number of Rows:");
      int rows = sc.nextInt();
      System.out.println("Enter the Number of Column:");
      int columns = sc.nextInt();

      // Create a 2D array to store the matrix
      int[][] matrix = new int[rows][columns];

      System.out.println("Enter the elements of the matrix:");
      for (int i = 0; i < rows; i++) {
         for (int j = 0; j < columns; j++) {
            System.out.print("Enter element at position [" + (i + 1) + "][" + (j + 1) + "]: ");
            matrix[i][j] = sc.nextInt();
         }
      }
      System.out.println("\nThe " +rows+" by "+columns+" matrix is:");
      for (int i = 0; i < rows; i++) {
         for (int j = 0; j < columns; j++) {
            System.out.print(matrix[i][j] + "\t");
         }
         System.out.println();
      }
      sc.close();
   }
}
```

OUTPUT

Enter the Number of Rows:

2

Enter the Number of Column:

3

Enter the elements of the matrix:

Enter element at position [1][1]: 12

Enter element at position [1][2]: 15

Enter element at position [1][3]: 18

Enter element at position [2][1]: 22

Enter element at position [2][2]: 32

Enter element at position [2][3]: 43

The 2 by 3 matrix is:

12 15 18

22 32 43

Practical No 2. wap in java take two 4 by 4 Matrix as input from user and print addition of Matrix

```
import java.util.Scanner;
public class MatrixAdd
 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the Number of Rows:");
      int rows = sc.nextInt();
      System.out.println("Enter the Number of Column:");
      int columns = sc.nextInt();

      // Create two 2D arrays to store the matrices
      int[][] matrix1 = new int[rows][columns];
      int[][] matrix2 = new int[rows][columns];
      int[][] resultMatrix = new int[rows][columns];

      // Take input for the first matrix
      System.out.println("Enter elements for the first matrix:");
      for (int i = 0; i < rows; i++) {
         for (int j = 0; j < columns; j++) {
            System.out.print("Enter element at position [" + (i + 1) + "][" + (j + 1) + "]: ");
            matrix1[i][j] = sc.nextInt();
         }
      }
      // Take input for the second matrix
      System.out.println("\nEnter elements for the second matrix:");
      for (int i = 0; i < rows; i++) {
         for (int j = 0; j < columns; j++) {
            System.out.print("Enter element at position [" + (i + 1) + "][" + (j + 1) + "]: ");
            matrix2[i][j] = sc.nextInt();
         }
      }
```

```
        // Perform matrix addition
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                resultMatrix[i][j] = matrix1[i][j] + matrix2[i][j];
            }
        }
        System.out.println("\nThe result of matrix addition is:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                System.out.print(resultMatrix[i][j] + "\t");
            }
            System.out.println();
        }
        sc.close();
    }
}
```

OUTPUT

```
Enter the Number of Rows:
4
Enter the Number of Column:
4
Enter elements for the first matrix:
Enter element at position [1][1]: 12
Enter element at position [1][2]: 13
Enter element at position [1][3]: 14
Enter element at position [1][4]: 15
Enter element at position [2][1]: 21
Enter element at position [2][2]: 22
Enter element at position [2][3]: 23
Enter element at position [2][4]: 24
```

Enter element at position [3][1]: 31

Enter element at position [3][2]: 32

Enter element at position [3][3]: 33

Enter element at position [3][4]: 34

Enter element at position [4][1]: 41

Enter element at position [4][2]: 42

Enter element at position [4][3]: 43

Enter element at position [4][4]: 44

Enter elements for the second matrix:

Enter element at position [1][1]: 1

Enter element at position [1][2]: 2

Enter element at position [1][3]: 3

Enter element at position [1][4]: 4

Enter element at position [2][1]: 5

Enter element at position [2][2]: 6

Enter element at position [2][3]: 7

Enter element at position [2][4]: 8

Enter element at position [3][1]: 11

Enter element at position [3][2]: 12

Enter element at position [3][3]: 13

Enter element at position [3][4]: 14

Enter element at position [4][1]: 16

Enter element at position [4][2]: 17

Enter element at position [4][3]: 18

Enter element at position [4][4]: 19

The result of matrix addition is:

13 15 17 19

26 28 30 32

42 44 46 48

57 59 61 63

Practical No 3. wap in java which gives implentation of interface and abstract class.

```
// Define an interface
interface Animal {
    void makeSound();
}
// Define an abstract class
abstract class Shape {
    abstract void draw();
}
// Implement the interface and extend the abstract class
class Dog implements Animal {
    @Override
    public void makeSound() {
        System.out.println("Woof! Woof!");
    }
}
class Circle extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a circle");
    }
}

public class interfaceAbstract {
    public static void main(String[] args) {
        // Using the Dog class that implements the Animal interface
        System.out.println("-: Interface :-");
        Dog dog = new Dog();
        dog.makeSound();
```

```
        // Using the Circle class that extends the Shape abstract class
        System.out.println("-: Abstract Class :-");
        Circle circle = new Circle();
        circle.draw();
    }
}
```

OUTPUT

```
-: Interface :-
Woof! Woof!
-: Abstract Class :-
Drawing a circle
```

Practical No 4. Wap to take string as a input and perform all operations of string, stringBuffer and StringTokenizer class on given string like remove(),substring,concat() etc..

```
import java.util.StringTokenizer;
import java.util.Scanner;

public class StringOperations {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter String:");
        String inputString = sc.nextLine();

        // String operations
        System.out.println("Original String: " + inputString);
        System.out.println("Length of the string: " + inputString.length());
        System.out.println("Substring from index 3 to the end: " + inputString.substring(5));
        System.out.println("Concatenation with another string: " + inputString.concat(" How are you?"));
        System.out.println("Uppercase: " + inputString.toUpperCase());
        System.out.println("Lowercase: " + inputString.toLowerCase());
        System.out.println();

        // StringBuffer operations
        StringBuffer stringBuffer = new StringBuffer(inputString);
        System.out.println("Original StringBuffer: " + stringBuffer);
        stringBuffer.append(" How are you?");
        System.out.println("Appending another string: " + stringBuffer);
        stringBuffer.insert(5, " T ");
        System.out.println("Inserting 'T' at index 6: " + stringBuffer);
        stringBuffer.delete(6, 7);
        System.out.println("Deleting characters from index 5 to 6: " + stringBuffer);
        System.out.println();
```

```
        // StringTokenizer operations
        StringTokenizer tokenizer = new StringTokenizer(inputString);
        System.out.println("Original StringTokenizer:"+ inputString);
        System.out.println("Number of tokens: " + tokenizer.countTokens());
        while (tokenizer.hasMoreTokens()) {
            System.out.println("Token: " + tokenizer.nextToken());
        }
    }
}
```

OUTPUT

```
Enter String:
Kunal Mahajan

Original String: Kunal Mahajan
Length of the string: 13
Substring from index 3 to the end:  Mahajan
Concatenation with another string: Kunal Mahajan How are you?
Uppercase: KUNAL MAHAJAN
Lowercase: kunal mahajan

Original StringBuffer: Kunal Mahajan
Appending another string: Kunal Mahajan How are you?
Inserting 'T' at index 6: Kunal T  Mahajan How are you?
Deleting characters from index 5 to 6: Kunal   Mahajan How are you?

Original StringTokenizer:Kunal Mahajan
Number of tokens: 2
Token: Kunal
Token: Mahajan
```

Practical No 5. wap in java by using try catch block which used handle predefined exception like ArrayIndexOutofBounds

```
public class ArrayIndexOutOfBounds {
    public static void main(String[] args) {
        try {
            // Creating an array of size 3
            int[] numbers = {1, 2, 3};
            // Accessing an element at index 3 (which is out of bounds)
            int element = numbers[3];
            System.out.println("Element at index 3: " + element);
        } catch (ArrayIndexOutOfBoundsException e) {
            // Handling the ArrayIndexOutOfBoundsException
            System.out.println("Exception caught: " + e.getMessage());
        } finally {
            // This block will be executed whether an exception is caught or not
            System.out.println("Program completed.");
        }
    }
}
```

OUTPUT

Exception caught: Index 3 out of bounds for length 3

Program completed.

Practical No 6. wap in java by using try catch block ,take a input from user which is age and if age >18 print your eligible for vote otherwise give a exception "sorry you are not eligible to vote!"

```
import java.util.Scanner;


public class VoteEligibility {
   public static void main(String[] args) {
      try {
         // Taking user input for age
         Scanner sc = new Scanner(System.in);
         System.out.print("Enter your age: ");
         int age = sc.nextInt();


         // Checking eligibility
         if (age > 18) {
            System.out.println("You are eligible to vote.");
         } else {
            throw new Exception("Sorry, you are not eligible to vote!");
         }
      } catch (Exception e) {
         // Catching and handling the exception
         System.out.println("Exception: " + e.getMessage());
      }
   }
}
```

OUTPUT

Enter your age: 22

You are eligible to vote.

OR

Enter your age: 15

Exception: Sorry, you are not eligible to vote!

Practical No 7. wap in java , to demonstrate multithreading by extending Thread class.

```
class MyThread extends Thread {
   public void run() {
      for (int i = 1; i <= 5; i++) {
         System.out.println(Thread.currentThread().getId() + " Value " + i);
      }
   }
}
public class MultiThread {
   public static void main(String args[]) {
      // Creating two threads
      MyThread thread1 = new MyThread();
      MyThread thread2 = new MyThread();

      // Starting the threads
      thread1.start();
      thread2.start();
   }
}
```

OUTPUT

```
15 Value 1
16 Value 1
15 Value 2
16 Value 2
15 Value 3
16 Value 3
15 Value 4
16 Value 4
15 Value 5
16 Value 5
```

Practical No 8. wap in java to demonstrate multithreading by using Runnable interface

```
// Create a class that implements the Runnable interface
class MyRunnable implements Runnable {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getId() + " Value " + i);
        }
    }
}
public class MultithreadRunnable {
    public static void main(String args[]) {
        //Create an instance of the class that implements Runnable
        MyRunnable myRunnable = new MyRunnable();

        //Create threads using the Runnable instance
        Thread thread1 = new Thread(myRunnable);
        Thread thread2 = new Thread(myRunnable);
        thread1.start();
        thread2.start();
    }
}
```

OUTPUT

```
15 Value 1
15 Value 2
16 Value 1
16 Value 2
15 Value 3
16 Value 3
15 Value 4
15 Value 5
16 Value 4
16 Value 5
```

Practical No 9. wap in java to show use of all predefined method of linkedlist and vector.

```
import java.util.LinkedList;
import java.util.Vector;

public class ListVector {
    public static void main(String[] args) {
        System.out.println("---- LinkedList Example ----");
        // Creating a LinkedList
        LinkedList<String> linkedList = new LinkedList<>();

        // Adding elements
        linkedList.add("Apple");
        linkedList.add("Banana");
        linkedList.add("Orange");

        // Displaying elements
        System.out.println("LinkedList Elements: " + linkedList);

        // Adding elements at the beginning and end
        linkedList.addFirst("Grapes");
        linkedList.addLast("Pineapple");

        // Displaying elements after modifications
        System.out.println("LinkedList Elements after Adding: " + linkedList);

        // Removing an element
        linkedList.remove("Banana");

        // Displaying elements after removal
        System.out.println("LinkedList Elements after remove: " + linkedList);
```

```
        // Getting the size
        System.out.println("Size of LinkedList: " + linkedList.size());

        System.out.println("\n---- Vector Example ----");
        // Creating a Vector
        Vector<Integer> vector = new Vector<>();

        // Adding elements
        vector.add(10);
        vector.add(20);
        vector.add(30);

        // Displaying elements
        System.out.println("Vector Elements: " + vector);

        // Adding an element at a specific index
        vector.add(1, 15);

        // Displaying elements after modification
        System.out.println("Vector Elements after Adding: " + vector);

        // Removing an element
        vector.removeElement(20);

        // Displaying elements after removal
        System.out.println("Vector Elements after remove: " + vector);

        // Getting the size
        System.out.println("Size of Vector: " + vector.size());
    }
}
```

OUTPUT

---- LinkedList Example ----

LinkedList Elements: [Apple, Banana, Orange]

LinkedList Elements after Adding: [Grapes, Apple, Banana, Orange, Pineapple]

LinkedList Elements after remove: [Grapes, Apple, Orange, Pineapple]

Size of LinkedList: 4

---- Vector Example ----

Vector Elements: [10, 20, 30]

Vector Elements after Adding: [10, 15, 20, 30]

Vector Elements after remove: [10, 15, 30]

Size of Vector: 3

Practical No 10. wap in java using all methods of hashmap

```
import java.util.HashMap;
public class HashMapEx {
    public static void main(String[] args) {
        // Creating a HashMap
        HashMap<Integer, String> hashMap = new HashMap<>();

        // Adding key-value pairs to the HashMap
        hashMap.put(1, "Apple");
        hashMap.put(2, "Banana");
        hashMap.put(3, "Orange");
        hashMap.put(4, "Grapes");

        // Displaying the original HashMap
        System.out.println("Original HashMap: " + hashMap);

        // Accessing a value using a key
        System.out.println("Value at key 2: " + hashMap.get(2));

        // Checking if a key is present
        System.out.println("Is key 3 present? " + hashMap.containsKey(3));

        // Checking if a value is present
        System.out.println("Is value 'Banana' present? " +
hashMap.containsValue("Banana"));

        // Removing a key-value pair
        hashMap.remove(1);
        System.out.println("HashMap after removing key 1: " + hashMap);

        // Getting the size of the HashMap
        System.out.println("Size of HashMap: " + hashMap.size());
```

```
        // Checking if the HashMap is empty
        System.out.println("Is HashMap empty? " + hashMap.isEmpty());

        // Clearing the HashMap
        hashMap.clear();
        System.out.println("HashMap after clearing: " + hashMap);
    }
}
```

OUTPUT

```
Original HashMap: {1=Apple, 2=Banana, 3=Orange, 4=Grapes}
Value at key 2: Banana
Is key 3 present? true
Is value 'Banana' present? true
HashMap after removing key 1: {2=Banana, 3=Orange, 4=Grapes}
Size of HashMap: 3
Is HashMap empty? false
HashMap after clearing: {}
```

Practical No 11. Wap in java create a table student in database by using jdbc student table should contain roll no,name,class,div.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class CreateTableExample {
    // JDBC URL, username, and password of MySQL server
    private static final String JDBC_URL = "jdbc:mysql://localhost:3306/abc";
    private static final String USERNAME = "root";
    private static final String PASSWORD = " ";
   public static void main(String[] args) {
        // JDBC variables
        Connection connection = null;
        Statement statement = null;
    try {
         // Register JDBC driver
          Class.forName("com.mysql.cj.jdbc.Driver");
  // Open a connection
          System.out.println("Connecting to database...");
          connection = DriverManager.getConnection(JDBC_URL,
      USERNAME, PASSWORD);
  // Create a statement
          statement = connection.createStatement();
   // Define SQL query to create the student table
          String createTableQuery = "CREATE TABLE IF NOT EXISTS
      student (" +
              "roll_no INT PRIMARY KEY," +
              "name VARCHAR(255)," +
              "class VARCHAR(50)," +
              "div CHAR(1)" +
              ")";
   // Execute the SQL query to create the table
```

```
statement.executeUpdate(createTableQuery);
System.out.println("Table 'student' created successfully.");
 } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
            } finally {
                try {
                    if (statement != null) {
                        statement.close();
                    }
                    if (connection != null) {
                        connection.close();
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
```

Practical No 12. Write a program in java to convert an Iterator to a list in Java.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;


public class IteratorToListExample {
    public static void main(String[] args) {
        Iterator<String> iterator = createIterator();
// Convert Iterator to List
        List<String> list = convertIteratorToList(iterator);
// Print the elements in the List
        System.out.println("List elements: " + list);
    }
  private static Iterator<String> createIterator() {
        // Create an Iterator with some elements
        List<String> elements = new ArrayList<>();
        elements.add("Java");
        elements.add("Python");
        elements.add("C++");
 return elements.iterator();
    }
    private static <T> List<T> convertIteratorToList(Iterator<T> iterator) {
        // Convert Iterator to List
        List<T> list = new ArrayList<>();
        while (iterator.hasNext()) {
            list.add(iterator.next());
        }
        return list;
    }
}
```

Output:-

```
List elements: [Java, Python, C+]
```

Practical No 13. Write a program using JDBC to perform insert operation in MYSQL database.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.*;


public class JdbcInsertExample {
   // JDBC URL, username, and password of MySQL server
   private static final String JDBC_URL = "jdbc:mysql://localhost:3306/abc";
   private static final String USERNAME = "root";
   private static final String PASSWORD = " ";
 public static void main(String[] args) {
       // JDBC variables
       Connection connection = null;
       Statement statement = null;
       ResultSet resultSet = null;
 try {
           Class.forName("com.mysql.cj.jdbc.Driver");
           System.out.println("Connecting to database...");
           connection = DriverManager.getConnection(JDBC_URL, USERNAME,
PASSWORD);
           statement = connection.createStatement();
// Insert a record into the student table
           String insertQuery = "INSERT INTO student (roll_no, name, class, div) VALUES
(?, ?, ?, ?)";
           try (PreparedStatement preparedStatement =
connection.prepareStatement(insertQuery)) {
               preparedStatement.setInt(1, 101);
               preparedStatement.setString(2, "John Doe");
               preparedStatement.setString(3, "10th");
               preparedStatement.setString(4, "A");
int rowsAffected = preparedStatement.executeUpdate();
               System.out.println(rowsAffected + " row(s) inserted.");
           }
```

```
// Retrieve and print all records from the student table
            String selectQuery = "SELECT * FROM student";
            resultSet = statement.executeQuery(selectQuery);
System.out.println("\nStudent Table after Insert Operation:");
            while (resultSet.next()) {
                int rollNo = resultSet.getInt("roll_no");
                String name = resultSet.getString("name");
                String studentClass = resultSet.getString("class");
                String div = resultSet.getString("div");
System.out.println("Roll No: " + rollNo + ", Name: " + name + ", Class: " + studentClass
+ ", Division: " + div);
            }
} catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        } finally {
            // Close JDBC resources in the finally block to ensure they are closed even if an
exception occurs
            try {
                if (resultSet != null) {
                    resultSet.close();
                }
                if (statement != null) {
                    statement.close();
                }
                if (connection != null) {
                    connection.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Practical No 14. WAP in java to implement Polymorphism.

```
class Animal {
    void makeSound() {
        System.out.println("Some generic sound");
    }
}

class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Bark! Bark!");
    }
}

class Cat extends Animal {
    @Override
    void makeSound() {
        System.out.println("Meow");
    }
}

public class Polymorphism {

    // Method with two integer parameters
    static int add(int a, int b) {
        return a + b;
    }

    // Method with three integer parameters
    static int add(int a, int b, int c) {
        return a + b + c;
```

```
    }
    public static void main(String[] args) {
        // Calling different versions of the add method
        int result1 = add(10, 20);
        int result2 = add(10, 20, 30);

        // Displaying the results
        System.out.println("Static Polymorphism Means Method Overloading");
        System.out.println("Result 1: " + result1);
        System.out.println("Result 2: " + result2);

        System.out.println("Runtime Polymorphism Means Method Overriding");
        Animal myDog = new Dog();
        Animal myCat = new Cat();

        myDog.makeSound();
        myCat.makeSound();
    }
}
```

OUTPUT

```
Static Polymorphism Means Method Overloading
Result 1: 30
Result 2: 60
Runtime Polymorphism Means Method Overriding
Bark! Bark!
Meow
```

Practical No 15. WAP in JAVA to print Right angle star pattern

```
import java.util.Scanner;

public class RightAngleStar {

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

        // Get the number of rows from the user
        System.out.print("Enter the number of rows for the star pattern: ");
        int rows = scanner.nextInt();

        for (int i = 1; i <= rows; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print("* ");
            }
            System.out.println();
        }
        scanner.close();
    }
}
```

OUTPUT

```
Enter the number of rows for the star pattern: 6
*
* *
* * *
* * * *
* * * * *
* * * * * *
```

Practical No 16. WAP in Java to implement Java program using priority.

```
import java.util.PriorityQueue;
public class PriorityQ {

   public static void main(String[] args) {
      PriorityQueue<Integer> priorityQueue = new PriorityQueue<>();

      priorityQueue.add(30);
      priorityQueue.add(20);
      priorityQueue.add(50);
      priorityQueue.add(10);

      System.out.println("Elements in the Priority Queue:");

      while (!priorityQueue.isEmpty()) {
         System.out.println(priorityQueue.poll());
      }
   }
}
```

OUTPUT

```
Elements in the Priority Queue:
10
20
30
50
```

Practical No 17. WAP in JAVA to implement Inheritance.

```
// Base class (superclass)
class Animal {
    void eat() {
        System.out.println("Animal is eating");
    }

    void sleep() {
        System.out.println("Animal is sleeping");
    }
}
// Derived class (subclass)
class Dog extends Animal {
    void bark() {
        System.out.println("Dog is barking");
    }
}
// Derived class (subclass)
class Cat extends Animal {
    void meow() {
        System.out.println("Cat is meowing");
    }
}
public class InheritanceExample {
    public static void main(String[] args) {
        // Create an instance of the Dog class
        Dog myDog = new Dog();
  // Access methods from the base class
        myDog.eat();    // Inherited method
        myDog.sleep();  // Inherited method
 // Access method specific to the Dog class
        myDog.bark();
   System.out.println("\n----------------------\n");
```

```
// Create an instance of the Cat class
        Cat myCat = new Cat();
// Access methods from the base class
        myCat.eat();    // Inherited method
        myCat.sleep();  // Inherited method
// Access method specific to the Cat class
        myCat.meow();
    }
}
```

Output:-

Animal is eating

Animal is sleeping

Dog is barking

Animal is eating

Animal is sleeping

Cat is meowing

Practical No 18. Write a JAVA SERVELET Program to shows the Fibonacci series up to a particular term, while the input is taken from an HTML form.

Create an HTML form (index.html):

```
<!DOCTYPE html>
<html>
<head>
   <title>Fibonacci Series</title>
</head>
<body>
   <h2>Generate Fibonacci Series</h2>
   <form action="FibonacciServlet" method="post">
      Enter the number of terms: <input type="text" name="terms">
      <input type="submit" value="Generate">
   </form>
</body>
</html>
```

```
//Create a Java Servlet (FibonacciServlet.java):
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/FibonacciServlet")
public class FibonacciServlet extends HttpServlet {
   private static final long serialVersionUID = 1L;
protected void doPost(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
                response.setContentType("text/html");
                PrintWriter out = response.getWriter();
               // Get the number of terms from the HTML form
```

```
            String termsStr = request.getParameter("terms");
        try {
                // Parse the input to get the number of terms
                int numTerms = Integer.parseInt(termsStr);
        // Generate and print Fibonacci series up to the specified term
                out.println("<h3>Fibonacci Series:</h3>");
                printFibonacciSeries(out, numTerms);
            } catch (NumberFormatException e) {
                out.println("<p style='color:red;'>Invalid input. Please enter a valid
        number.</p>");
            }
          }
private void printFibonacciSeries(PrintWriter out, int numTerms) {
      int firstTerm = 0;
      int secondTerm = 1;
for (int i = 0; i < numTerms; i++) {
        out.print(firstTerm + ", ");
        int nextTerm = firstTerm + secondTerm;
        firstTerm = secondTerm;
        secondTerm = nextTerm;
      }
   }
}
```

Output:-

Enter the number of terms: 7

0

1

1

2

3

5

8