University of Science &Technology Chittagong
(USTC)
FACULTY of SCIENCE ENGINEERING & TECHNOLOGY (FSET)
Department of Computer Science & Engineering(CSE)

Lab Task - 4

Course Code: CSE 123

Course Title: Object Oriented Programming

Submitted To:

Debabrata Mallick
Lecturer
CSE,FSET,USTC

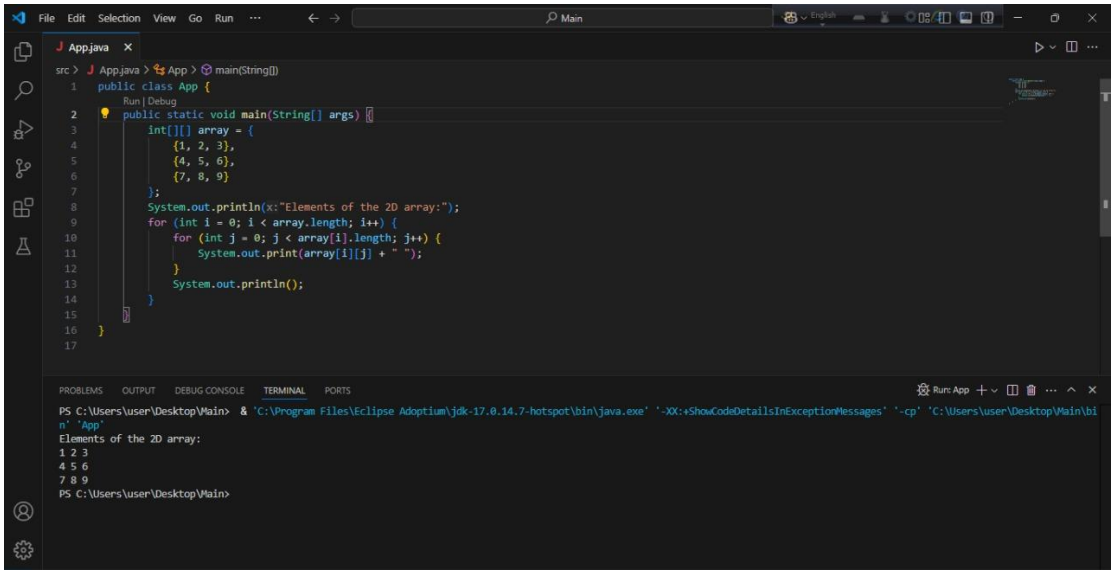
Submitted By:

Name: Afifah Mansur
ID: 0022420005101018
Reg NO: 1169
Dept: CSE
Semester: 2nd

1. 2D Array :

Code :

```
public class App {
    public static void main(String[] args) {
        int[][] array = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };
        System.out.println("Elements of the 2D array:");
        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                System.out.print(array[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```



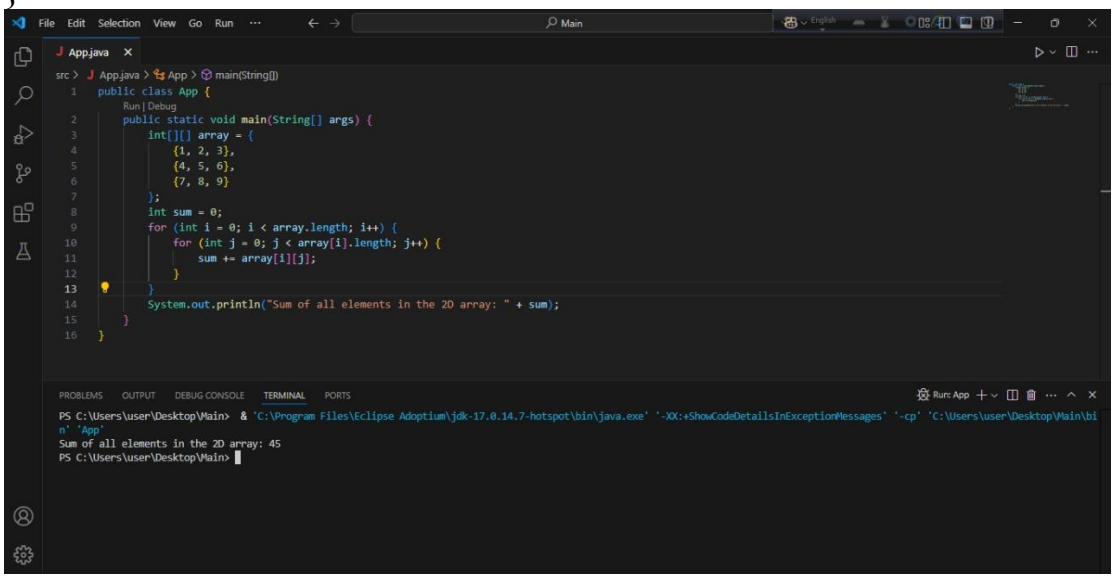
Explanation:

* array.length gives the number of rows.
* array[i].length gives the number of columns in each row.
* Two nested for loops iterate through each element of the array, row by row.

2. Sum of Elements :

Code:

```
public class App {
    public static void main(String[] args) {
        int[][] array = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };
        int sum = 0;
        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                sum += array[i][j];
            }
        }
        System.out.println("Sum of all elements in the 2D array: " + sum);
    }
}
```



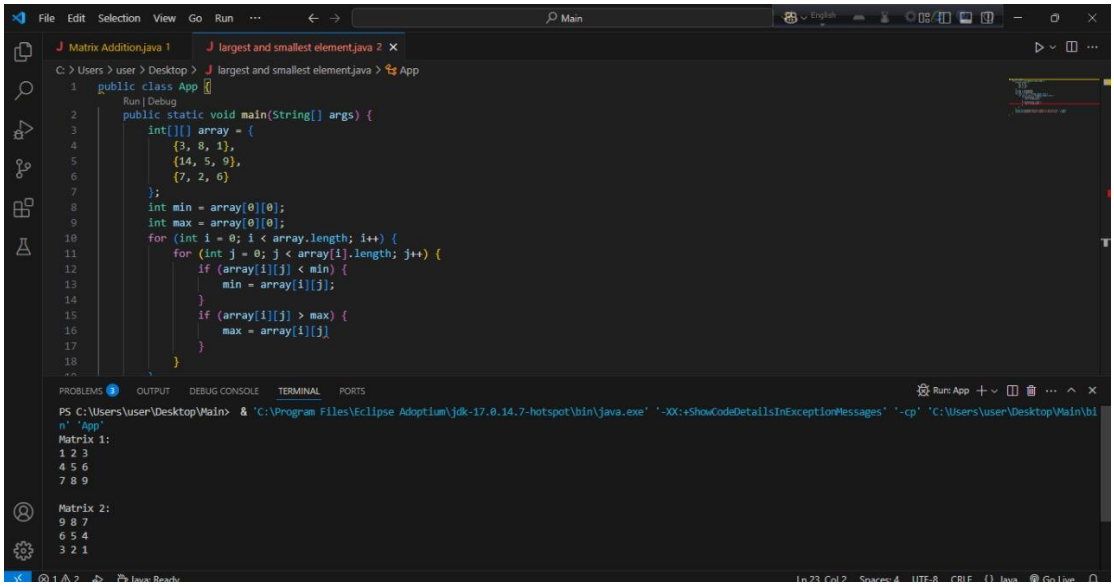
Explanation:

* Array Initialization: The 2D array is declared and initialized with sample values.
* Sum Calculation: Nested for loops iterate through each element.
* The outer loop goes through each row (i index).
* The inner loop goes through each column (j index).
* Sum Accumulation: Each element's value is added to the sum variable.

3. Find Maximum and Minimum:

Code :

```
public class App {
    public static void main(String[] args) {
        int[][] array = {
            {3, 8, 1},
            {14, 5, 9},
            {7, 2, 6}
        };
        int min = array[0][0];
        int max = array[0][0];
        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                if (array[i][j] < min) {
                    min = array[i][j];
                }
                if (array[i][j] > max) {
                    max = array[i][j]
                }
            }
        }
        System.out.println("Smallest element in the 2D array: " + min);
        System.out.println("Largest element in the 2D array: " + max);
    }
}
```

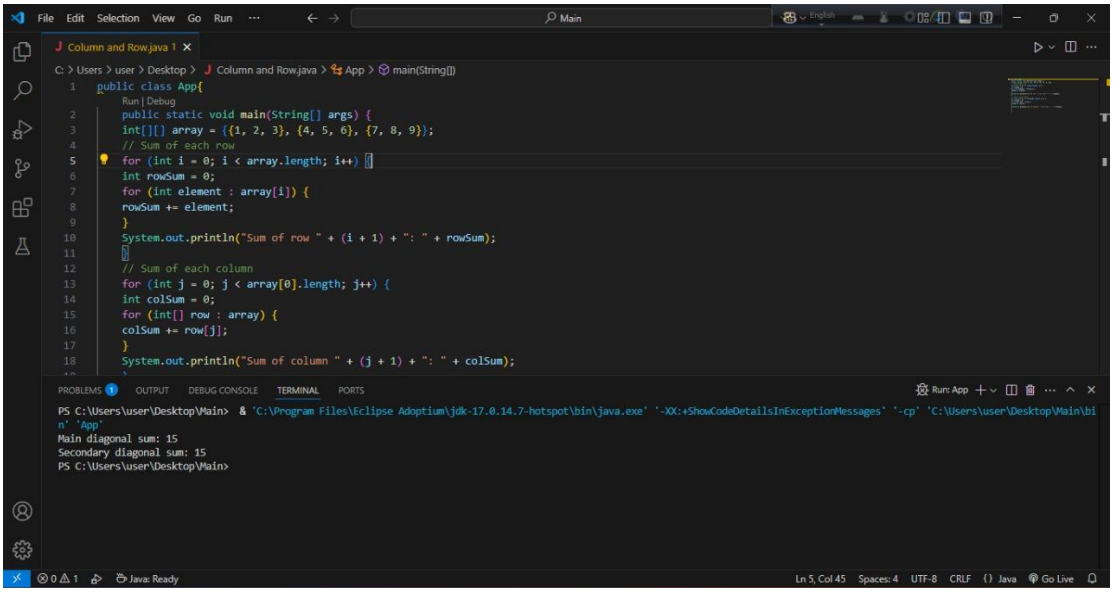


Explanation:
* Array Initialization: A 2D array is declared with some values.
* Min & Max Initialization: The first element of the array is assigned to both min and max.
* Iteration:
* Nested for loops traverse the array.
* Each element is compared to min and max to update them accordingly.

4. Row- wise and Column - wise Sum:

Code:

```
public class App{
    public static void main(String[] args) {
    int[][] array = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

       for (int i = 0; i < array.length; i++) {
    int rowSum = 0;
    for (int element : array[i]) {
    rowSum += element;
      }
    System.out.println("Sum of row " + (i + 1) + ": " + rowSum);
     }

    for (int j = 0; j < array[0].length; j++) {
    int colSum = 0;
    for (int[] row : array) {
    colSum += row[j];
    }
    System.out.println("Sum of column " + (j + 1) + ": " + colSum);
    }
    }
    }
```



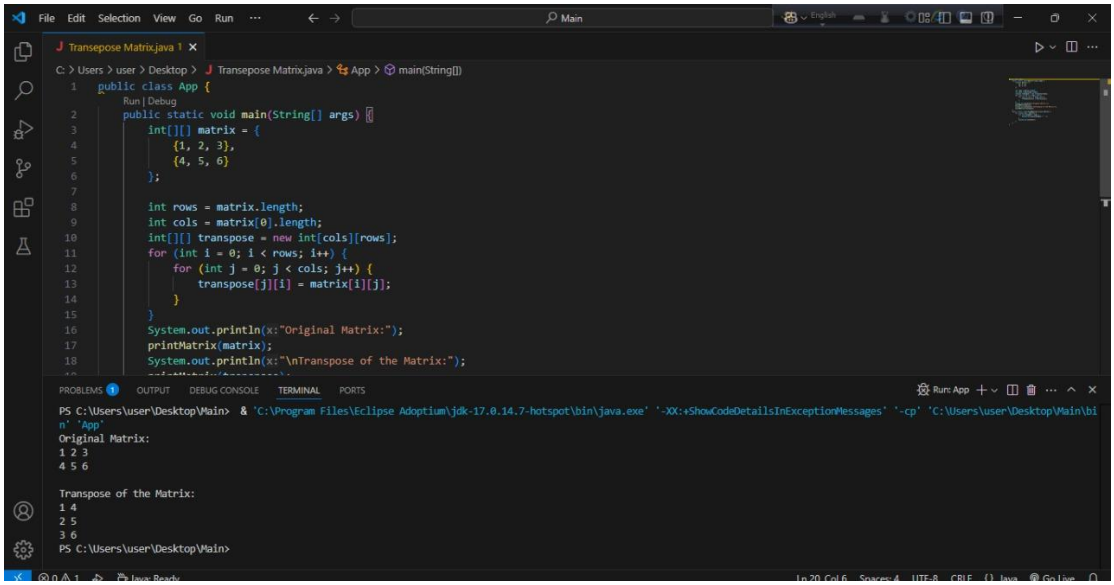
Explanation:

* Array Initialization: A 2D array is declared and initialized with some sample values.
* Row Sum Calculation:
* The outer loop iterates through rows.
* The inner loop adds each element in the row to rowsum.
* Column Sum Calculation:
* The outer loop iterates through columns.
* The inner loop adds each element in the column to colSum.

5. Transpose of a Matrix:

Code:

```
public class App {
    public static void main(String[] args) {
        int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6}
        };
        int rows = matrix.length;
        int cols = matrix[0].length;
        int[][] transpose = new int[cols][rows];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                transpose[j][i] = matrix[i][j];
            }
        }
        System.out.println("Original Matrix:");
        printMatrix(matrix);
        System.out.println("\nTranspose of the Matrix:");
        printMatrix(transpose);
    }
    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int element : row) {
                System.out.print(element + " ");
            }
            System.out.println();
        }
    }
```



Explanation:

* Matrix Initialization:

The given matrix is of size 2x3 (2 rows, 3 columns).

* Transpose Logic:

The element at position (i, j) in the original matrix is moved to (j, 1) in the transposed matrix.

* Matrix Printing:

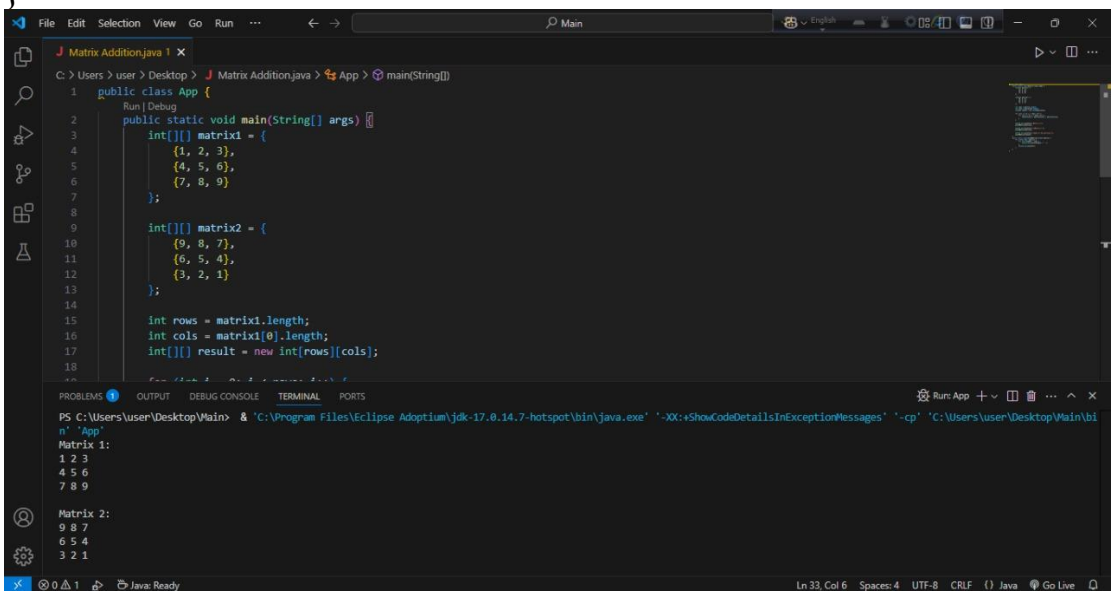
A helper method printMatrix () is used to print both the original and transposed matrices.

6. Matrix Addition:

Code:

```
public class App {
    public static void main(String[] args) {
        int[][] matrix1 = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };
        int[][] matrix2 = {
            {9, 8, 7},
            {6, 5, 4},
            {3, 2, 1}
        };
        int rows = matrix1.length;
        int cols = matrix1[0].length;
        int[][] result = new int[rows][cols];

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

        System.out.println("Matrix 1:");
        printMatrix(matrix1);

        System.out.println("\nMatrix 2:");
        printMatrix(matrix2);

        System.out.println("\nSum of the matrices:");
        printMatrix(result);
    }
    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int element : row) {
                System.out.print(element + " ");
            }
            System.out.println();
        }
    }
```

}



Explanation:

* Matrix Initialization:

Two matrices (matrix1 and matrix2) of the same size are declared and initialized.

* Matrix Addition Logic:

Each element of the resulting matrix is the sum of corresponding elements from matrix1 andmatrix2:

result|ib] - matrixllil] + matrix2 [l

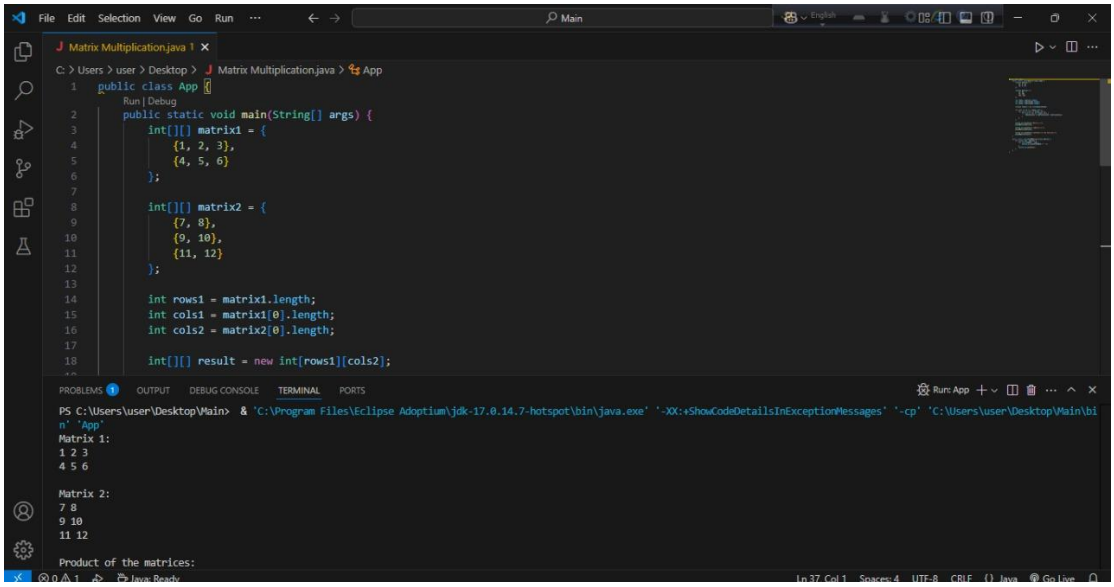
* Matrix Display:

The printMatrix () method prints matrices in a clean format.

7. Matrix Multiplication:

Code:

```
public class App {
    public static void main(String[] args) {
        int[][] matrix1 = {
            {1, 2, 3},
            {4, 5, 6}
        };
        int[][] matrix2 = {
            {7, 8},
            {9, 10},
            {11, 12}
        };
        int rows1 = matrix1.length;
        int cols1 = matrix1[0].length;
        int cols2 = matrix2[0].length;

        int[][] result = new int[rows1][cols2];

        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols2; j++) {
                for (int k = 0; k < cols1; k++) {
                    result[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }
        System.out.println("Matrix 1:");
        printMatrix(matrix1);
        System.out.println("\nMatrix 2:");
        printMatrix(matrix2);
        System.out.println("\nProduct of the matrices:");
        printMatrix(result);
    }
    public static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int element : row) {
                System.out.print(element + " ");
            }
            System.out.println();
        }
    }
}
```



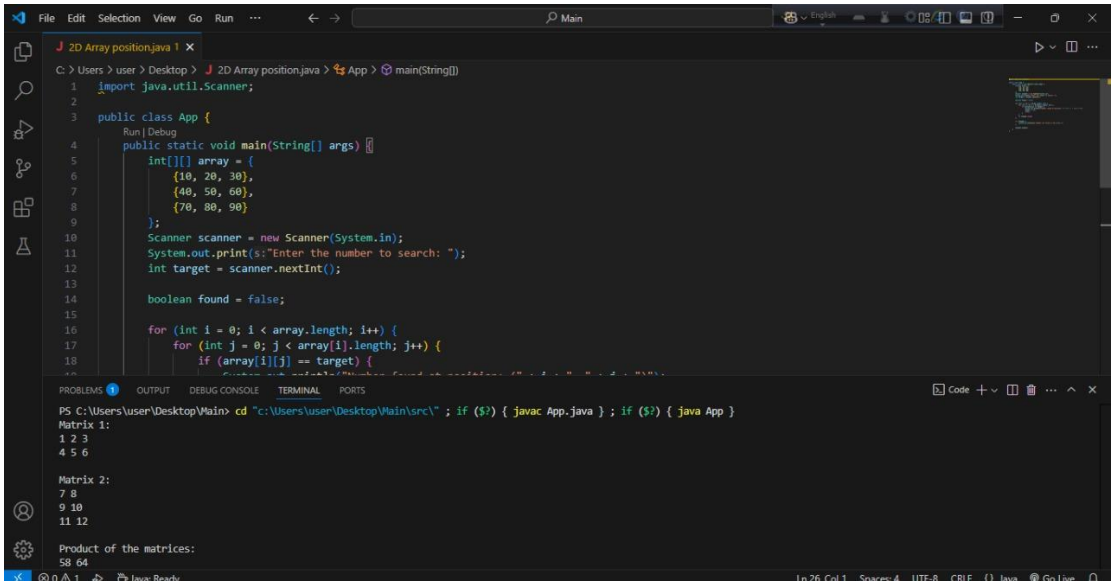
Explanation:

* Matrix dimensions:
* matrix1 is of size 2x3.
* matrix2 is of size 3x2.
* The resulting matrix will have 2 rows and 2 columns.

8. Search an Element in 2D Array :

Code :

```
import java.util.Scanner;

public class App {
    public static void main(String[] args) {
        int[][] array = {
            {10, 20, 30},
            {40, 50, 60},
            {70, 80, 90}
        };
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number to search: ");
        int target = scanner.nextInt();

        boolean found = false;

        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                if (array[i][j] == target) {
                    System.out.println("Number found at position: (" + i + ", " +
j + ")");
                    found = true;
                    break;
                }
            }
            if (found) break;
        }

        if (!found) {
            System.out.println("Number not found in the array.");
        }

        scanner.close();
    }
}
```



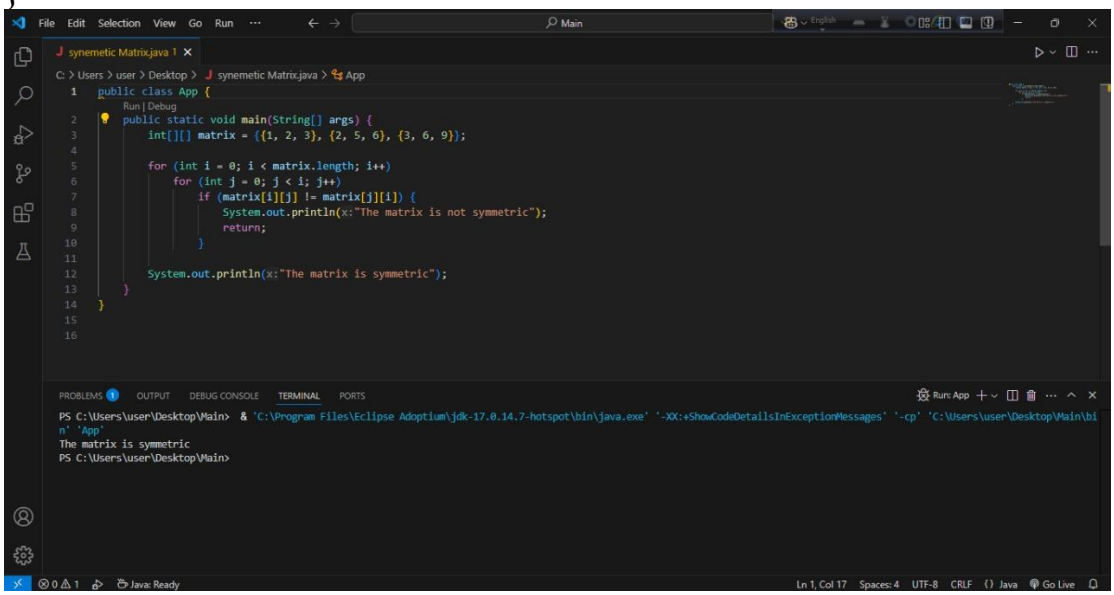
Explanation:

* 2D array: The array is pre-initialized with some values.
* User input: The program takes a number from the user to search.
* Search logic:
* Two nested for loops go through each element.
* If the target number is found, its position (row, column) is printed.
* The break statement exits the loops once the number is found.
* Not found case: If the number isn't found, it prints a relevant message.

9.Check Symmetric Matrix:

Code:

```
public class App {
    public static void main(String[] args) {
        int[][] matrix = {{1, 2, 3}, {2, 5, 6}, {3, 6, 9}};

        for (int i = 0; i < matrix.length; i++)
            for (int j = 0; j < i; j++)
                if (matrix[i][j] != matrix[j][i]) {
                    System.out.println("The matrix is not symmetric");
                    return;
                }

        System.out.println("The matrix is symmetric");
    }
}
```



Explanation:

Matrix Initialization: Initializes a 2D array matrix with predefined values.

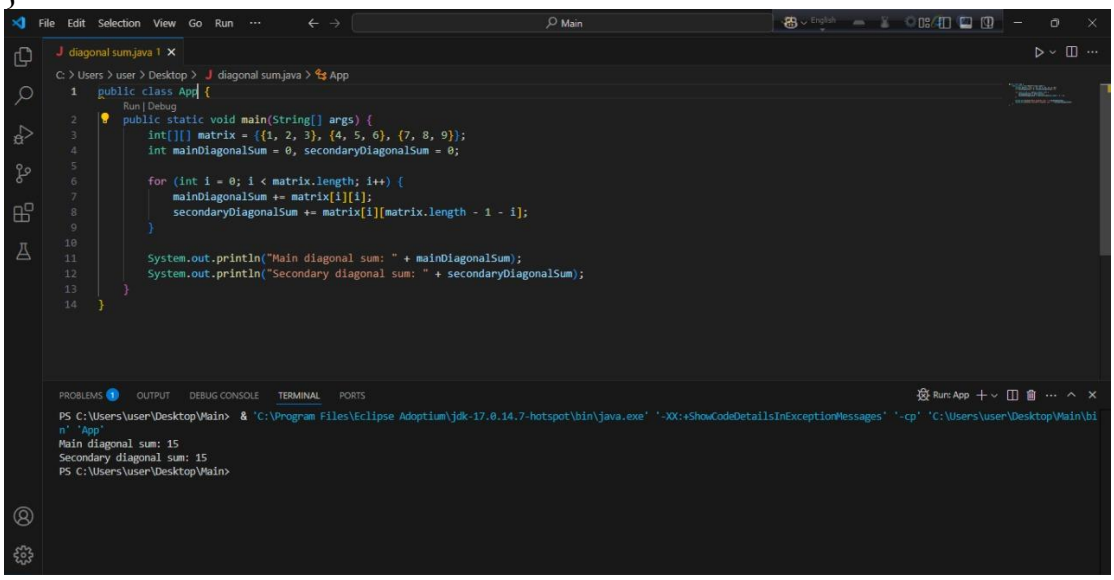
Symmetry Check: Uses nested loops to iterate through the elements of the matrix. Only the lower triangle of the matrix (excluding the diagonal) is compared with the corresponding elements of the upper triangle.

Print Result: If a mismatch is found, it prints "The matrix is not symmetric" and exits. Otherwise, it prints "The matrix is symmetric".

10.Diagonal Sum :

Code:

```
public class App {
    public static void main(String[] args) {
        int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        int mainDiagonalSum = 0, secondaryDiagonalSum = 0;

        for (int i = 0; i < matrix.length; i++) {
            mainDiagonalSum += matrix[i][i];
            secondaryDiagonalSum += matrix[i][matrix.length - 1 - i];
        }

        System.out.println("Main diagonal sum: " + mainDiagonalSum);
        System.out.println("Secondary diagonal sum: " +
secondaryDiagonalSum);
    }
}
```



Explanation:

- Matrix Initialization: Initializes a 2D array matrix with predefined values.
- Diagonal Sum Initialization: Initializes variables mainDiagonalSum and secondaryDiagonalSum to 0.
- Sum Calculation: Iterates through the matrix. For the main diagonal, it adds elements where the row index and column index are the same. For the secondary diagonal, it adds elements where the row index and column index are complementary.
- Print Result: Prints the sums of the main and secondary diagonals.