Task 4

1. Print a 2D Array

Code:

```
public class Print2DArray {
   public static void main(String[] args) {
      int[][] array = {
         {1, 2, 3},
         {4, 5, 6},
         {7, 8, 9}
      };

      for (int[] row : array) {
         for (int element : row) {
            System.out.print(element + " ");
         }
         System.out.println();
      }
   }
}
```

Explanation:

This program prints all elements of a given 2D array. It uses a nested for-each loop to iterate through the array. The outer loop iterates over each row, and the inner loop iterates over each element in the row. Each element is printed, followed by a space. After printing all elements in a row, a new line is started to ensure the output is formatted as a 2D grid.

2. Sum of Elements

Code:

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

```
        };

        int sum = 0;
        for (int[] row : array) {
            for (int element : row) {
                sum += element;
            }
        }

        System.out.println("Sum of all elements: " + sum);
    }
}
```

Explanation:

This program calculates the sum of all elements in a 2D array. It initializes a variable `sum` to 0 and uses a nested `for-each` loop to iterate through the array. Each element is added to the `sum` variable. After iterating through all elements, the total sum is printed.

3. Find Maximum and Minimum

Code:

```
public class MaxMinIn2DArray {
    public static void main(String[] args) {
        int[][] array = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        int max = array[0][0];
        int min = array[0][0];

        for (int[] row : array) {
            for (int element : row) {
                if (element > max) max = element;
                if (element < min) min = element;
            }
        }
        System.out.println("Maximum element: " + max);
        System.out.println("Minimum element: " + min);
    }
}
```

<u>Explanation:</u>

This program finds the largest and smallest elements in a 2D array. It initializes two variables, max and min, with the first element of the array. A nested for-each loop iterates through the array, comparing each element with max and min. If an element is greater than max, it updates max. If an element is smaller than min, it updates min. Finally, the maximum and minimum values are printed.

4. Row-wise and Column-wise Sum

<u>Code:</u>

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

        int rows = array.length;
        int cols = array[0].length;

        // Row-wise sum
        for (int i = 0; i < rows; i++) {
            int rowSum = 0;
            for (int j = 0; j < cols; j++) {
                rowSum += array[i][j];
            }
            System.out.println("Sum of row " + (i + 1) + ": " + rowSum);
        }

        // Column-wise sum
        for (int j = 0; j < cols; j++) {
            int colSum = 0;
            for (int i = 0; i < rows; i++) {
                colSum += array[i][j];
            }
            System.out.println("Sum of column " + (j + 1) + ": " + colSum);
        }
    }
}
```

Explanation:

This program calculates the sum of each row and each column in a 2D array. It first calculates the row-wise sum by iterating through each row and summing its elements. The sum of each row is stored in `rowSum` and printed. Next, it calculates the column-wise sum by iterating through each column and summing its elements. The sum of each column is stored in `colSum` and printed.

5. Transpose of a Matrix

Code:

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

        int rows = array.length;
        int cols = array[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] = array[i][j];
            }
        }

        // Print transpose matrix
        for (int[] row : transpose) {
            for (int element : row) {
                System.out.print(element + " ");
            }
            System.out.println();
        }
    }
}
```

Explanation:

This program finds the transpose of a matrix, where rows become columns and vice versa. It creates a new 2D array `transpose` with dimensions swapped. A nested loop iterates through the original array, and each element `array[i][j]` is assigned to `transpose[j][i]`. Finally, the transposed matrix is printed.

6. Matrix Addition

Code:

```
public class MatrixAddition {
    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];
            }
        }
        // Print result matrix
        for (int[] row : result) {
            for (int element : row) {
                System.out.print(element + " ");
            }
            System.out.println();
        }
    }
}
```

Explanation:

This program adds two matrices and stores the result in another 2D array. It defines two matrices, matrix1 and matrix2, and creates a new 2D array result to store the sum. A nested loop iterates through the matrices, and the corresponding elements are added and stored in result. The resulting matrix is printed.

7. Matrix Multiplication

Code:

```
public class MatrixMultiplication {
    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];
            }
        }
    }

    // Print result matrix
    for (int[] row : result) {
        for (int element : row) {
            System.out.print(element + " ");

        }
        System.out.println();
    }
  }
}
```

<u>**Explanation:**</u>

This program multiplies two matrices and stores the result in another 2D array. It defines two matrices, matrix1 and matrix2, and creates a new 2D array result to store the product. A triple nested loop is used to perform the multiplication. The outer loop iterates over the rows of matrix1, the middle loop iterates over the columns of matrix2, and the inner loop performs the dot product of the row from matrix1 and the column from matrix2. The resulting matrix is printed.

8. Search an Element in 2D Array

<u>**Code:**</u>

```
public class SearchIn2DArray {
  public static void main(String[] args) {
    int[][] array = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    int target = 5;
    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("Element found at position: (" + i + ", " + j + ")");
                    found = true;
                    break;
                }
            }
            if (found) break;
        }

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

Explanation:

This program searches for a given number in a 2D array and prints its position. It defines a target number and uses a nested loop to iterate through the array. If the target is found, its position (i, j) is printed, and the loop exits. If the target is not found, a message is printed.

10. Diagonal Sum

Code:

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

        int mainDiagonalSum = 0;
        int secondaryDiagonalSum = 0;

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

        System.out.println("Sum of main diagonal: " + mainDiagonalSum);
        System.out.println("Sum of secondary diagonal: " + secondaryDiagonalSum);
    }
}
```

Explanation:

This program calculates the sum of the main diagonal and secondary diagonal of a square matrix. It initializes two variables, mainDiagonalSum and secondaryDiagonalSum, to 0. A loop iterates through the array, adding the main diagonal elements (array[i][i]) to mainDiagonalSum and the secondary diagonal elements (array[i][n-1-i]) to secondaryDiagonalSum. The sums are printed.

13. Check Identity Matrix

Code:

```
public class IdentityMatrix {
    public static void main(String[] args) {
        int[][] array = {
            {1, 0, 0},
            {0, 1, 0},
            {0, 0, 1}
        };

        boolean isIdentity = true;

        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                if (i == j && array[i][j] != 1) {
                    isIdentity = false;
                    break;
                } else if (i != j && array[i][j] != 0) {
                    isIdentity = false;
                    break;
                }
            }
            if (!isIdentity) break;
        }
    if (isIdentity) {
      System.out.println("The matrix is an identity matrix.");
    } else {
      System.out.println("The matrix is not an identity matrix.");
    }
  }
}
```

Explanation:

This program checks if a matrix is an identity matrix, where diagonal elements are 1 and all other elements are 0. It initializes a boolean variable isIdentity to true. A nested loop iterates through the array. If any diagonal element is not 1 or any non-diagonal element is not 0, isIdentity is set to false. The result is printed based on the value of isIdentity.