1.
Matrix Diagonal Sum: https://leetcode.com/problems/matrix-diagonal-sum/

```
public class MatrixDiagonalSum {

    public static int matrixDiagonalSum(int[][] matrix) {
        int sum = 0;
        for (int i = 0; i < matrix.length; i++) {
            sum += matrix[i][i];
        }
        return sum;
    }

    public static void main(String[] args) {
        int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        System.out.println(matrixDiagonalSum(matrix));
    }
}
```

2.
Rotate Image :https://leetcode.com/problems/rotate-image/

```
class RotateImage {
    public static void rotate(int[][] matrix) {
        int n = matrix.length;
        for (int i = 0; i < n / 2; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                int temp = matrix[i][j];
                matrix[i][j] = matrix[n - 1 - j][n - 1 - i];
                matrix[n - 1 - j][n - 1 - i] = matrix[n - 1 - i][n - 1
- j];
                matrix[n - 1 - i][n - 1 - j] = matrix[i][n - 1 - j];
                matrix[i][n - 1 - j] = temp;
            }
        }
    }

    public static void main(String[] args) {
        int[][] matrix = new int[][]{{1, 2, 3}, {4, 5, 6}, {7, 8,
9}};
        rotate(matrix);
        for (int[] row : matrix) {
            for (int element : row) {
```

```
                System.out.print(element + " ");
            }
            System.out.println();
        }
    }
}
```

3.
59. Spiral Matrix II : https://leetcode.com/problems/spiral-matrix-ii/

```
public class SpiralMatrixII {

    public static List<Integer> spiralOrder(int n) {
        List<Integer> result = new ArrayList<>();
        int[][] matrix = new int[n][n];
        int rowStart = 0;
        int rowEnd = n - 1;
        int colStart = 0;
        int colEnd = n - 1;
        int direction = 0;
        for (int i = 0; i < n * n; i++) {
```

```
switch (direction) {
    case 0:
        for (int j = colStart; j <= colEnd; j++) {
            result.add(matrix[rowStart][j]);
        }
        rowStart++;
        break;
    case 1:
        for (int j = rowStart; j <= rowEnd; j++) {
            result.add(matrix[j][colEnd]);
        }
        colEnd--;
        break;
    case 2:
        for (int j = colEnd; j >= colStart; j--) {
            result.add(matrix[rowEnd][j]);
        }
        rowEnd--;
        break;
    case 3:
        for (int j = rowEnd; j >= rowStart; j--) {
            result.add(matrix[j][colStart]);
        }
        colStart++;
        break;
```

```
            }
            direction = (direction + 1) % 4;
        }
        return result;
    }

    public static void main(String[] args) {
        int n = 3;
        List<Integer> result = spiralOrder(n);
        for (int i : result) {
            System.out.print(i + " ");
        }
        System.out.println();
    }
}
```

4.
867. Transpose Matrix : https://leetcode.com/problems/transpose-matrix/

```
public class TransposeMatrix {

    public static int[][] transpose(int[][] matrix) {
```

```
        int n = matrix.length;
        int m = matrix[0].length;
        int[][] transposedMatrix = new int[m][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                transposedMatrix[j][i] = matrix[i][j];
            }
        }
        return transposedMatrix;
    }

    public static void main(String[] args) {
        int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
        int[][] transposedMatrix = transpose(matrix);
        for (int[] row : transposedMatrix) {
            for (int i : row) {
                System.out.print(i + " ");
            }
            System.out.println();
        }
    }
}
```

5
73. Set Matrix Zeroes : https://leetcode.com/problems/set-matrix-zeroes/

```
public class SetMatrixZeroes {

    public static void main(String[] args) {
        int[][] matrix = {{1, 1, 1}, {1, 0, 1}, {1, 1, 1}};
        setMatrixZeroes(matrix);
        for (int[] row : matrix) {
            for (int num : row) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }

    public static void setMatrixZeroes(int[][] matrix) {
        int rows = matrix.length;
        int columns = matrix[0].length;

        boolean[] rowHasZero = new boolean[rows];
        boolean[] columnHasZero = new
```

```
boolean[columns];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                if (matrix[i][j] == 0) {
                    rowHasZero[i] = true;
                    columnHasZero[j] = true;
                }
            }
        }

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                if (rowHasZero[i] || columnHasZero[j]) {
                    matrix[i][j] = 0;
                }
            }
        }
    }
}
```

6.
1424. Diagonal Traverse II : https://leetcode.com/problems/diagonal-traverse-ii/

```
import java.util.*;

public class DiagonalTraverseII {

    public static List<Integer> diagonalTraverse(int[][] nums) {
        List<Integer> result = new ArrayList<>();
        int n = nums.length;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j <= i; j++) {
                result.add(nums[i][j]);
            }
        }
        return result;
    }

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

```
        List<Integer> result = diagonalTraverse(nums);
        System.out.println(result);
    }
}
```

7.
Anti Diagonal
https://www.interviewbit.com/problems/anti-diagonals/

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

```
public class Matrix {

    private int[][] matrix;

    public Matrix(int[][] matrix) {
        this.matrix = matrix;
    }

    public int get(int row, int column) {
        return matrix[row][column];
    }

    public void set(int row, int column, int value) {
        matrix[row][column] = value;
    }

    public int getHeight() {
        return matrix.length;
    }

    public int getWidth() {
        return matrix[0].length;
    }

    public void print() {
```

```
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[0].length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        int[][] matrix = new int[][]{{1, 2, 3}, {4, 5, 6}, {7, 8,
9}};
        Matrix m = new Matrix(matrix);
        m.print();
    }
}
```