Home work. Tasks \*. Arrays

**Task 1\* (ArrayFiller)**

Написать метод заполнения массива array указанным значением filler.

void fill(int[] array, int filler)

**Task 2\* (ArraySumCalculator)**

Написать метод вычисления суммы всех элементов массива.

int sum(int[] array)

Пример работы:

sum(new int[]{2, 1, 3}) == 6

**Task 3\* (ArrayPositiveFinder)**

Написать методы поиска индекса первого и последнего положительного числа. Если положительных чисел нет, вернуть -1.

int findFirstPositive(int[] array)

int findLastPositive(int[] array)

Пример работы:

findFirstPositive(new int[]{-3, 0, -1, 4, -2, 5}) == 3

findLastPositive(new int[]{-3, 0, -1, 4, -2, 5}) == 5

findFirstPositive(new int[]{-3, 0, -1, -4, -2, -5}) == -1

**Task 4\* (ArrayPositivesCounter)**

Написать метод подсчета количества положительных чисел в массиве.

int countPositives(int[] array)

Пример работы:

countPositives(new int[]{-3, 0, -1, 4, -2, 5}) == 2

countPositives(new int[]{-3, 0, -1, -4, -2, -5}) == 0

**Task 5\* (SequenceArrayCreator)**

Написать метод создания массива размера size, заполненного значениями от 0..(size-1). Если size < 0, бросить исключение IllegalArgumentException.

int[] createSequenceArray(int size)

Пример работы:

createSequenceArray(5) => {0, 1, 2, 3, 4}

createSequenceArray(-1) 🡺 IllegalArgumentException

**Task 6\* (MatrixPrinter)**

Написать метод вывода матрицы в консоль. Предполагается, что матрица прямоугольная.

void printMatrix(int[][] matrix)

**Task 7\* (MatrixAverageCalculator)**

Написать метод подсчета среднего арифметического всех элементов матрицы. Предполагается, что матрица прямоугольная.

double calculateAverage(int[][] matrix)

Пример работы:

findMaxSumRow(new int[][]{{0, 3, -2},

{2, 1, 3},

{-1, 5, 2}}) == 1.4444…

**Task 8\* (MatrixMaxSumRowFinder)**

Написать метод поиска индекса строки с максимальной суммой элементов. Если таких строк несколько, вернуть индекс первой. Предполагается, что матрица прямоугольная.

int findMaxSumRow(int[][] matrix)

Пример работы:

findMaxSumRow(new int[][]{{0, 3, -2},

{2, 1, 3},

{-1, 5, 2}}) == 1

**Task 9\* (MatrixTransposer)**

Написать метод транспонирования матрицы. Исходная матрица не должна изменяться. Предполагается, что матрица прямоугольная.

<https://ru.wikipedia.org/wiki/Транспонированная_матрица>

int[][] transpose(int[][] sourceMatrix)

Пример работы:

transpose(new int[][]{ {1, 2, 3, 4},

{5, 6, 7, 8} }) => { {1, 5},

{2, 6},

{3, 7},

{4, 8} }

**Task 10\* (IdentityMatrixChecker)**

Написать метод определения, является ли матрица единичной. Исходная матрица не должна изменяться. Предполагается, что матрица прямоугольная. Если матрица НЕ квадратная, бросить исключение IllegalArgumentException.

[https://ru.wikipedia.org/wiki/Единичная\_матрица](https://ru.wikipedia.org/wiki/Транспонированная_матрица)

boolean isIdentity(int[][] matrix)

Пример работы:

isIdentity(new int[][]{ {1, 0, 0},

{0, 1, 0},

{0, 0, 1} }) == true

isIdentity(new int[][]{ {1, 0, 2},

{0, 1, 0},

{2, 0, 1} }) == false

isIdentity(new int[][]{ {1, 0, 0},

{0, 1, 0},

{0, 0, 1},

{0, 0, 0} }) 🡺 IllegalArgumentException