Минобрнауки России | РГРТУ | Кафедра ВПМ

Курс «ПРОГРАММИРОВАНИЕ НА JAVA»

**Отчет о практической работе №**4

Выполнил:

Вербицкая Ирина Сергеевна

студент группы 143

электронная почта oora.frt@gmail.com

Проверил:

Пруцков Александр Викторович

д-р техн. наук, профессор кафедры ВПМ

Рязань 2023

# Задание

Подсчитать суммы значений, находящихся выше и ниже главной диагонали квадратной матрицы. Вычесть из большего значения суммы меньшее значение. Если суммы значений равны, то вывести соответствующее сообщение.

# Основные классы, реализующие задание

## Класс Runner

package ru.rsreu.verbickaya0405;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

public class Runner {

/\*\*

\* A dimension of matrix.

\*

\* @see #main(String[])

\*/

static final int MATRIX\_SIZE = 4;

/\*\*

\* The bottom border of random generation of matrix elements.

\*

\* @see #main(String[])

\*/

static final int BOTTOM\_BOURDER\_OF\_GENERATION = -10;

/\*\*

\* The bottom border of random generation of matrix elements.

\*

\* @see #main(String[])

\*/

static final int UPPER\_BOURDER\_OF\_GENERATION = 10;

private static Resourcer resourcer = ProjectResourcer.getInstance();

/\*\*

\* The private constructor of utility class.

\*/

private Runner() {

}

/\*\*

\* The beginning of program execution.

\*/

public static void main(String[] args) {

SquareMatrix matrix = new SquareMatrix(MATRIX\_SIZE);

matrix.fill(BOTTOM\_BOURDER\_OF\_GENERATION, UPPER\_BOURDER\_OF\_GENERATION);

StringBuilder resultMessage = new StringBuilder(resourcer.getString("message.here.is.an.origin.matrix") + "\n");

resultMessage.append(matrix.toString() + "\n");

resultMessage.append(resourcer.getString("message.the.result.of.work") + "\n");

int above = matrix.getSumAboveMainDiagonal();

int below = matrix.getSumBelowMainDiagonal();

resultMessage.append(resourcer.getString("message.the.sum.above.diagonal") + above + "\n");

resultMessage.append(resourcer.getString("message.the.sum.below.diagonal") + below + "\n");

int i = matrix.getDifferenceBetweenSumAboveAndBelowMainDiagonal();

if (i == 0) {

resultMessage.append(resourcer.getString("message.the.difference.between.sums.is.null"));

} else {

resultMessage.append(resourcer.getString("message.the.difference.between.sums") + i);

}

System.out.println(resultMessage);

}

}

## Класс SquareMatrix

package ru.rsreu.verbickaya0405;

import java.util.Arrays;

import com.prutzkow.resourcer.ProjectResourcer;

import com.prutzkow.resourcer.Resourcer;

import com.prutzkow.twodimarray.Matrix;

public class SquareMatrix extends Matrix {

private static Resourcer resourcer = ProjectResourcer.getInstance();

/\*\*

\* Constructor of class.

\*

\* @param size A dimension of matrix.

\* @throws IllegalArgumentException

\*

\* @see {@link Runner#main()}

\*/

public SquareMatrix(int size) throws IllegalArgumentException {

super(size, size);

}

/\*\*

\* Fills the matrix by random integer values generated in the range of two

\* numbers.

\*

\* @param bourder1 The first border of random generating

\* @param bourder2 The second border of random generating

\*

\* @see Runner#main()

\*/

public void fill(int bourder1, int bourder2) {

int[] bourders = { bourder1, bourder2 };

Arrays.sort(bourders);

for (int i = 0; i < super.getRowCount(); i++) {

for (int j = 0; j < super.getColCount(); j++) {

super.arrayBody[i][j] = (int) (Math.random() \* (bourders[1] - bourders[0] + 1) + bourders[0]);

}

}

}

/\*\*

\* Forms a matrix output in string format with message about the number of

\* elements in the last row.

\*

\* @return A string of formatted matrix

\*

\* @see Runner#main()

\*/

@Override

public String toString() {

String s = "";

for (int i = 0; i < super.getRowCount(); i++) {

for (int j = 0; j < super.getColCount(); j++) {

s += String.format(super.FORMAT, super.arrayBody[i][j]);

}

s += "\n";

}

s += resourcer.getString("message.a.number.of.elements.in.the.last.row");

s += super.getRowLength(super.getRowCount() - 1);

return s;

}

/\*\*

\* Subtracts the bigger and the less sums of elements below and above main

\* diagonal in the matrix.

\*

\* @return The module of difference between this two sums.

\*

\* @see Runner#main()

\*/

public int getDifferenceBetweenSumAboveAndBelowMainDiagonal() {

int a = this.getSumAboveMainDiagonal();

int b = this.getSumBelowMainDiagonal();

return Math.abs(a - b);

}

/\*\*

\* Calculates sum of elements staying above main diagonal.

\*

\* @return The value of sum

\*

\* @see #getDifferenceBetweenSumAboveAndBelowMainDiagonal()

\* @see Runner#main()

\*/

public int getSumAboveMainDiagonal() {

return this.getSumBehindMainDiagonal(true);

}

/\*\*

\* Calculates sum of elements staying below main diagonal.

\*

\* @return The value of sum

\*

\* @see #getDifferenceBetweenSumAboveAndBelowMainDiagonal()

\* @see Runner#main()

\*/

public int getSumBelowMainDiagonal() {

return this.getSumBehindMainDiagonal(false);

}

/\*\*

\* Calculates sum of matrix elements below or above main diagonal.

\*

\* @param parameterAboveIsTrueOrBelowIsFalse Defines the sum. If

\* {@code parameterAboveIsTrueOrBelowIsFalse = true}

\* it is sum above diagonal. If

\* {@code parameterAboveIsTrueOrBelowIsFalse = false}

\* it is sum below diagonal

\* @return The value of a sum

\*

\* @see #getSumAboveMainDiagonal()

\* @see #getSumBelowMainDiagonal()

\*/

private int getSumBehindMainDiagonal(boolean parameterAboveIsTrueOrBelowIsFalse) {

boolean b = parameterAboveIsTrueOrBelowIsFalse;

int resultSum = 0;

for (int i = 0; i < super.getRowCount(); i++) {

for (int j = 0; j < super.getColCount(); j++) {

if (((b) && (i < j)) || ((!b) && (i > j))) {

resultSum += super.arrayBody[i][j];

}

}

}

return resultSum;

}

}