Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Рязанский государственный радиотехнический университет»  
(ФГБОУ ВО «РГРТУ», РГРТУ)

Кафедра «Вычислительная и прикладная математика» (ВПМ)

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ № 4

по дисциплине

**«ПРОГРАММИРОВАНИЕ НА JAVA»**

Выполнил:

студент группы 643

Паршина Анна Романовна

Проверил:

Пруцков Александр Викторович,

д-р техн. наук, профессор кафедры ВПМ

Рязань 2019

# Задание

Среди всех элементов с минимальными значениями массива определить координаты минимума, находящегося ближе всех к элементу, находящемуся в последней строке последнего столбца. При выводе массива на экран вывести также номера строк и столбцов.

# Основные классы, реализующие задание

## Класс Runner

package ru.rsreu.parshina0413;

/\*\*

\* Class Runner.

\*

\* @autor Parshina Anna Romanovna

\*/

public class Runner {

/\*\*

\* Constructor - creating new object

\*

\* @see Runner#Runner()

\*/

private Runner() {

}

/\*\*

\* Procedure for start application

\*

\* @param args - parameters of command string

\*/

public static void main(String[] args) {

NearestMinimalElementSearcher array = new NearestMinimalElementSearcher(3, 4);

array.fill(1, 12);

StringBuilder result = new StringBuilder();

result.append(Resourcer.getString("message.array")).append("\n").append(array).append("\n")

.append(array.searchCoordinatesNearestMinimalElement());

System.out.println(result);

}

}

## Класс NearestMinimalElementSearcher

package ru.rsreu.parshina0413;

import com.prutzkow.twodimarray.Matrix;

/\*\*

\* Class NearestMinimalElementSearcher.

\*

\* @autor Parshina Anna Romanovna

\*/

public class NearestMinimalElementSearcher extends Matrix {

/\*\*

\* Constructor - create new object

\*

\* @see NearestMinimalElementSearcher#NearestMinimalElementSearcher(int, int)

\*/

public NearestMinimalElementSearcher(int sizeX, int sizeY) throws IllegalArgumentException {

super(sizeX, sizeY);

}

/\*\*

\* Function for find the value minimal array element

\*

\* @return value minimal element

\*/

private int searchMinimalElement() {

int minimal = super.arrayBody[0][0];

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

for (int j = 0; j < super.arrayBody[i].length; j++) {

if (super.arrayBody[i][j] < minimal) {

minimal = super.arrayBody[i][j];

}

}

}

return minimal;

}

/\*\*

\* Procedure for find coordinates nearest minimal element

\*

\* @param minimal - minimal value of array

\*

\* @return coordinates

\*/

private CoordinatesElement searchCoordinatesNearestElement(int minimal) {

int minimalRow = 0;

int minimalColumn = 0;

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

for (int j = 0; j < super.arrayBody[i].length; j++) {

if (super.arrayBody[i][j] == minimal) {

minimalRow = i + 1;

minimalColumn = j + 1;

}

}

}

CoordinatesElement coordinates = new CoordinatesElement(minimalRow, minimalColumn);

return coordinates;

}

/\*\*

\* Procedure for fill the array with random numbers

\*

\* @param lowLine

\* @param upLine

\*/

public void fill(int lowLine, int upLine) {

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

for (int j = 0; j < super.arrayBody[i].length; j++) {

super.arrayBody[i][j] = lowLine + (int) (Math.random() \* upLine);

}

}

}

/\*\*

\* Function for search the coordinates of the nearest minimal element

\*

\* @return string with coordinates

\*/

public StringBuilder searchCoordinatesNearestMinimalElement() {

int minimal = searchMinimalElement();

CoordinatesElement coordinates = this.searchCoordinatesNearestElement(minimal);

StringBuilder result = new StringBuilder();

result.append(String.format(Resourcer.getString("message.format"), minimal, coordinates.getRow(),

coordinates.getColumn())).append(Resourcer.getString("message.nearestElement"));

return result;

}

/\*\*

\* Function for get the count of the first row

\*

\* @return count of the first row

\*/

private int getlengthFirstRow() {

return super.arrayBody[0].length;

}

private String outputArray() {

String s = "";

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

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

s += String.format(Resourcer.getString("message.format"), super.arrayBody[i][j], i + 1, j + 1);

}

s += "\n";

}

return s;

}

/\*\*

\* Override function from Object

\*

\* @return string with array and count of the first row

\*/

@Override

public String toString() {

return this.outputArray() + Resourcer.getString("message.lengthFirstRow") + this.getlengthFirstRow();

}

}

## Класс CoordinatesElement

package ru.rsreu.parshina0413;

/\*\*

\* Class CoordinatesElementGetter with fields row and column.

\*

\* @autor Parshina Anna Romanovna

\*/

public class CoordinatesElement {

private int row;

private int column;

/\*\*

\* Constructor - create new object

\*

\* @see CoordinatesElement#CoordinatesElementGetter(int, int)

\*/

public CoordinatesElement(int row, int column) {

this.row = row;

this.column = column;

}

/\*\*

\* Function for get the value row element

\*

\* @return value row element

\*/

public final int getRow() {

return this.row;

}

/\*\*

\* Function for get the value column element

\*

\* @return value column element

\*/

public final int getColumn() {

return this.column;

}

}