|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа,**

**обработки и интерпретации больших данных.**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 5 |

**Название:**

Исключения. Файлы.

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | Е.Е. Шморгун |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Вариант 1 Задача 9.** Выполнить задания на основе варианта 1 лабораторной работы 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т.д.

Main.java

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.Arrays;

import java.util.LinkedList;

import java.util.List;

import java.util.stream.Collectors;

public class Main {

public static void main(String[] args) {

Path input\_path = Paths.get("koefs.txt");

BufferedReader reader;

List<QuadraticEquation> arr = new LinkedList<>();

try {

reader = Files.newBufferedReader(input\_path);

while(reader.ready()) {

List<Double> koefs = Arrays.stream(reader

.readLine()

.split(" "))

.map(koef -> Double.parseDouble(koef))

.collect(Collectors.toList());

QuadraticEquation equation = new QuadraticEquation(

koefs.get(0),

koefs.get(1),

koefs.get(2)

);

arr.add(equation);

}

reader.close();

} catch (IOException e) {

System.out.println("Cannot read input file");

return;

} catch (IndexOutOfBoundsException e) {

System.out.println("Each row should contain three numbers");

return;

} catch (Exception e) {

System.out.printf("Unexpected error: %s\n", e);

return;

}

double min\_solution;

try {

min\_solution = arr.get(0).solve()[0];

} catch (QuadraticEquation.SolutionException e) {

System.out.println(e.message);

return;

}

double max\_solution = min\_solution;

try {

for (int i = 0; i < arr.size(); i++) {

System.out.printf("%d: ", i);

double[] solutions = arr.get(i).solve();

for (int j = 0; j < solutions.length; j++) {

System.out.printf("%s ", solutions[j]);

if (solutions[j] > max\_solution) {

max\_solution = solutions[j];

}

if (solutions[j] < min\_solution) {

min\_solution = solutions[j];

}

}

System.out.printf("\n");

}

} catch (QuadraticEquation.SolutionException e) {

System.out.println(e.message);

return;

} catch (Exception e) {

System.out.printf("Unexpected error: %s\n", e);

return;

}

Path output\_path = Paths.get("result.txt");

try {

BufferedWriter writer = Files.newBufferedWriter(output\_path);

writer.write(String.format("Max solution: %s\n", max\_solution));

writer.write(String.format("Min solution: %s\n", min\_solution));

Double[] increase\_interval = arr.get(0).increase\_interval();

Double[] decrease\_interval = arr.get(0).decrease\_interval();

writer.write(String.format("Extreme point for first: %s\n", arr.get(0).find\_extreme()));

writer.write(

String.format(

"Increase interval for first: (%s, %s)\n",

increase\_interval[0], increase\_interval[1]

)

);

writer.write(

String.format(

"Decrease interval for first: (%s, %s)\n",

decrease\_interval[0], decrease\_interval[1]

)

);

writer.close();

} catch (IOException e) {

System.out.println("Error while writing results to file");

return;

} catch (Exception e) {

System.out.printf("Unexpected error: %s\n", e);

return;

}

}

}

QuadraticEquation.java

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.Arrays;

import java.util.LinkedList;

import java.util.List;

import java.util.stream.Collectors;

public class Main {

public static void main(String[] args) {

Path input\_path = Paths.get("koefs.txt");

BufferedReader reader;

List<QuadraticEquation> arr = new LinkedList<>();

try {

reader = Files.newBufferedReader(input\_path);

while(reader.ready()) {

List<Double> koefs = Arrays.stream(reader

.readLine()

.split(" "))

.map(koef -> Double.parseDouble(koef))

.collect(Collectors.toList());

QuadraticEquation equation = new QuadraticEquation(

koefs.get(0),

koefs.get(1),

koefs.get(2)

);

arr.add(equation);

}

reader.close();

} catch (IOException e) {

System.out.println("Cannot read input file");

return;

} catch (IndexOutOfBoundsException e) {

System.out.println("Each row should contain three numbers");

return;

} catch (Exception e) {

System.out.printf("Unexpected error: %s\n", e);

return;

}

double min\_solution;

try {

min\_solution = arr.get(0).solve()[0];

} catch (QuadraticEquation.SolutionException e) {

System.out.println(e.message);

return;

}

double max\_solution = min\_solution;

try {

for (int i = 0; i < arr.size(); i++) {

System.out.printf("%d: ", i);

double[] solutions = arr.get(i).solve();

for (int j = 0; j < solutions.length; j++) {

System.out.printf("%s ", solutions[j]);

if (solutions[j] > max\_solution) {

max\_solution = solutions[j];

}

if (solutions[j] < min\_solution) {

min\_solution = solutions[j];

}

}

System.out.printf("\n");

}

} catch (QuadraticEquation.SolutionException e) {

System.out.println(e.message);

return;

} catch (Exception e) {

System.out.printf("Unexpected error: %s\n", e);

return;

}

Path output\_path = Paths.get("result.txt");

try {

BufferedWriter writer = Files.newBufferedWriter(output\_path);

writer.write(String.format("Max solution: %s\n", max\_solution));

writer.write(String.format("Min solution: %s\n", min\_solution));

Double[] increase\_interval = arr.get(0).increase\_interval();

Double[] decrease\_interval = arr.get(0).decrease\_interval();

writer.write(String.format("Extreme point for first: %s\n", arr.get(0).find\_extreme()));

writer.write(

String.format(

"Increase interval for first: (%s, %s)\n",

increase\_interval[0], increase\_interval[1]

)

);

writer.write(

String.format(

"Decrease interval for first: (%s, %s)\n",

decrease\_interval[0], decrease\_interval[1]

)

);

writer.close();

} catch (IOException e) {

System.out.println("Error while writing results to file");

return;

} catch (Exception e) {

System.out.printf("Unexpected error: %s\n", e);

return;

}

}

}

**Вариант 1 Задача 10.** Выполнить задания на основе варианта 1 лабораторной работы 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т.д.

Main.java

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.Arrays;

import java.util.LinkedList;

import java.util.List;

import java.util.stream.Collectors;

public class Main {

public static void main(String[] args) {

Path input\_path = Paths.get("input.txt");

BufferedReader reader;

List<BoolMatrix> arr = new LinkedList<>();

try {

reader = Files.newBufferedReader(input\_path);

while(reader.ready()) {

List<Integer> sizes = Arrays.stream(reader

.readLine()

.split(" "))

.map(koef -> Integer.parseInt(koef))

.collect(Collectors.toList());

Boolean[][] matrix = new Boolean[sizes.get(0)][sizes.get(1)];

for (int i = 0; i < sizes.get(0); i++) {

List<Boolean> row = Arrays.stream(reader

.readLine()

.split(" "))

.map(koef -> Boolean.parseBoolean(koef))

.collect(Collectors.toList());

matrix[i] = row.toArray(new Boolean[0]);

}

BoolMatrix boolMatrix = new BoolMatrix(matrix);

arr.add(boolMatrix);

}

reader.close();

} catch (IOException e) {

System.out.printf("Cannot read input file. Error: %s\n", e);

return;

} catch (IndexOutOfBoundsException e) {

System.out.println("Each row should contain three numbers");

return;

} catch (Exception e) {

System.out.printf("Unexpected error: %s\n", e);

return;

}

BoolMatrix matrix\_4 = arr.get(0)

.logic\_multiply(arr.get(1))

.logic\_add(arr.get(2).invert());

Path output\_path = Paths.get("result.txt");

try {

BufferedWriter writer = Files.newBufferedWriter(output\_path);

writer.write("Operations result:\n");

writer.write(matrix\_4.toString());

writer.write(String.format("Count: %d\n", matrix\_4.count()));

writer.write("Sort second matrix:\n");

arr.get(1).sort();

writer.write(arr.get(1).toString());

writer.close();

} catch (IOException e) {

System.out.println("Error while writing results to file");

return;

} catch (Exception e) {

System.out.printf("Unexpected error: %s\n", e);

return;

}

}

}

BoolMatrix.java

public class BoolMatrix {

private Boolean[][] matrix;

private int n;

private int m;

public BoolMatrix(Boolean[][] matrix) {

this.matrix = matrix;

this.n = matrix.length;

this.m = matrix[0].length;

}

public BoolMatrix(int n) {

this.matrix = new Boolean[n][n];

this.n = n;

this.m = n;

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

this.matrix[i][j] = false;

}

}

}

public BoolMatrix(int n, int m) {

this.matrix = new Boolean[n][m];

this.n = n;

this.m = m;

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

this.matrix[i][j] = false;

}

}

}

public BoolMatrix invert() {

Boolean[][] inverted = new Boolean[n][m];

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

inverted[i][j] = !matrix[i][j];

}

}

return new BoolMatrix(inverted);

}

public BoolMatrix logic\_add(BoolMatrix second) {

Boolean[][] result = new Boolean[n][m];

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

result[i][j] = matrix[i][j] || second.matrix[i][j];

}

}

return new BoolMatrix(result);

}

public BoolMatrix logic\_multiply(BoolMatrix second) {

Boolean[][] result = new Boolean[n][m];

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

result[i][j] = matrix[i][j] && second.matrix[i][j];

}

}

return new BoolMatrix(result);

}

public int count() {

int counter = 0;

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

if (matrix[i][j]) counter++;

}

}

return counter;

}

public void sort() {

for (int i = 0; i < n; i++) {

int min\_row\_value = row\_value(i);

int min\_row\_index = i;

for (int j = i + 1; j < n; j++) {

if (row\_value(j) < min\_row\_value) {

min\_row\_value = row\_value(j);

min\_row\_index = j;

}

}

if (min\_row\_index != i) {

Boolean[] tmp = matrix[i];

matrix[i] = matrix[min\_row\_index];

matrix[min\_row\_index] = tmp;

}

}

}

public String toString() {

String res = "";

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

res += String.format("%s, ", matrix[i][j]);

}

res += "\n";

}

return res;

}

public int row\_value(int row\_number) {

int result = 0;

for (int i = m - 1; i >= 0; i--) {

if (matrix[row\_number][i]) {

result += Math.pow(2, m - i - 1);

}

}

return result;

}

}

**Вывод:** В данной работе я познакомился с работой с файлами в Java. Изучил синтаксис исключений и их основные принципы.