|  |  |
| --- | --- |
| **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 (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

**ОТЧЕТ**

**по лабораторной работе № 5**

|  |  |
| --- | --- |
| **Название:** | Исключения, файлы |
| **Дисциплина:** | Языки программирования для работы с большими данными |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  | 14.04.2023 | И.Б. Нуриддинов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

Цель работы: Освоение и применение концепций исключений и работы с файлами в языке программирования Java.

*Вариант 1. Задание 1.* Определить класс Матрица размерности (n x n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения матриц. Объявить массив объектов. Создать методы, вычисляющие первую и вторую нормы матрицы



Контролировать 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.

public class Main {

public static void main(String[] args) {

Matrix[] matrices = new Matrix[5];

try {

matrices[0] = new Matrix(3);

matrices[1] = new Matrix(4);

matrices[2] = new Matrix(5);

matrices[3] = new Matrix(3, 23);

matrices[4] = new Matrix(6);

} catch (Exception e){

e.printStackTrace();

System.exit(1);

}

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

matrices[i].print();

System.out.println();

}

/\* Инициализация объекта с отрицательным размером

try {

Matrix A = new Matrix(-235);

} catch (Exception e){

e.printStackTrace();

System.exit(1); // завершение программы с кодом ошибки 1

}

\*/

/\* Инициализация объекта с большим значением

try {

Matrix C = new Matrix(100000000);

} catch (OutOfMemoryError e){

System.out.println("Ошибка: недостаточно памяти для создания объекта.");

System.exit(1); // завершение программы с кодом ошибки 1

}

\*/

Matrix A = new Matrix(2);

try {

Matrix tmp1 = new Matrix(2);

Matrix tmp2 = new Matrix(2);

tmp1.add(tmp2);

A = tmp1;

} catch (Exception e){

e.printStackTrace();

}

A.print();

System.out.println();

Matrix B = new Matrix(3);

Matrix C = new Matrix(4);

B.print();

System.out.println();

C.print();

System.out.println();

try{

B.multiply(C);

} catch (IllegalArgumentException e){

e.printStackTrace();

}

B.print();

System.out.println(A.normFirst());

System.out.println(A.normSecond());

int minNormFirstPos = Matrix.minNormFirst(matrices);

int minNormSecondPos = Matrix.minNormSecond(matrices);

System.out.println("Матрица с наименьшей первой нормы:");

matrices[minNormFirstPos].print();

System.out.println("Первая норма: "+ matrices[minNormFirstPos].normFirst());

System.out.println("Матрица с наименьшей второй нормы:");

matrices[minNormSecondPos].print();

System.out.println("Вторая норма: "+ matrices[minNormSecondPos].normSecond());

}

}

*Вариант 3. Задание 1.* Найти и вывести слова текста, для которых последняя буква одного слова совпадает с первой буквой следующего слова.

public class Main {

public static void main(String[] args) {

String inputFile = "input.txt";

String outputFile = "output.txt";

if (args.length == 2) {

inputFile = args[0];

outputFile = args[1];

} else {

try {

BufferedReader reader = new BufferedReader(new FileReader("config.txt"));

String line = reader.readLine();

inputFile = line.split("=")[1];

line = reader.readLine();

outputFile = line.split("=")[1];

reader.close();

} catch (IOException e) {

e.printStackTrace();

}

}

List<String> words = new ArrayList<>();

try {

BufferedReader reader = new BufferedReader(new FileReader(inputFile));

String line = reader.readLine();

while (line != null) {

String[] lineWords = line.split("\\s+");

for (String word : lineWords) {

if (word.length() != 0)

words.add(word);

}

line = reader.readLine();

}

reader.close();

} catch (IOException e) {

e.printStackTrace();

}

List<String> result = new ArrayList<>();

for (int i = 0; i < words.size() - 1; i++) {

String currentWord = words.get(i);

String nextWord = words.get(i + 1);

if (currentWord.charAt(currentWord.length() - 1) == nextWord.charAt(0)) {

result.add(currentWord);

}

}

try {

FileWriter writer = new FileWriter(outputFile);

for (String word : result) {

writer.write(word + "\n");

}

writer.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

Вывод: Лабораторная работа позволила освоить важные концепции и инструменты работы с исключениями и файлами в языке Java. Понимание исключений и их правильная обработка помогают создавать более надежное и безопасное программное обеспечение, обеспечивая корректную реакцию на возможные ошибки и исключительные ситуации.