|  |  |
| --- | --- |
| 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**

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

**обработки и интерпретации больших данных.**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 5 |
| **Вариант №** 8 |  |

**Название:**

Исключения и файлы

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-23М | |  |  | Н.М. Иванюк |
|  | | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  | |  |  |  |  |
| Преподаватель | |  |  |  | П.В. Степанов |
|  | |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

## Цель работы

Получение первичных навыков обработки исключительных ситуаций в языке программирования Java. Изучение основ работы с файлами и файловой системой.

## Ход работы

*Задание 1.*

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.

8. Определить класс Комплекс. Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения, деления, присваивания комплексных чисел. Создать два вектора размерности n из комплексных координат. Передать их в метод, который выполнит их сложение.

Листинг – Код программы класса «Complex» первого задания

public class Complex {

int re, im;

public Complex() {

this.re = 0;

this.im = 0;

}

public Complex(int re) {

this.re = re;

this.im = 0;

}

public Complex(int re, int im) {

this.re = re;

this.im = im;

}

public Complex(Complex complex) {

this.re = complex.re;

this.im = complex.im;

}

// Сложение

public Complex Add(Complex addee) {

// Создаем новый объект класса Complex

Complex sum = new Complex(

this.re + addee.re,

this.im + addee.im

);

return sum;

}

// Вычитание

public Complex Subtract(Complex subtractee) {

Complex sub = new Complex(

this.re - subtractee.re,

this.im - subtractee.im

);

return sub;

}

// Умножение

public Complex Multiply(Complex multee) {

int k1 = multee.re \* (this.re + this.im);

int k2 = this.re \* (multee.im - multee.re);

int k3 = this.im \* (multee.re + multee.im);

int real\_part = k1 - k3;

int imaginary\_part = k1 + k2;

Complex mul = new Complex(real\_part, imaginary\_part);

return mul;

}

// Деление

public Complex Divide(Complex dividee) {

int re\_num = (this.re \* dividee.re) + (this.im \* dividee.im);

int re\_den = (dividee.re \* dividee.re + dividee.im \* dividee.im);

int im\_num = (this.im \* dividee.re) - (this.re \* dividee.im);

int im\_den = (dividee.re \* dividee.re + dividee.im \* dividee.im);

Complex div = new Complex(

re\_num / re\_den,

im\_num / im\_den

);

return div;

}

// Присвоение

public void SetComplex(int re, int im) {

this.re = re;

this.im = im;

}

// Вывод числа

public String toString() {

String complex = re + " + i" + im;

return complex;

}

}

Листинг 2 – Код основной программы задания 1 (Подзадача 1)

import java.util.InputMismatchException;

import java.util.Scanner;

import java.util.Vector;

public class App {

public static void main(String[] args) throws Exception {

System.out.println("Введите n:");

Scanner console = new Scanner(System.in);

try {

int n = console.nextInt();

Vector<Complex> complex\_1 = new Vector<>();

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

System.out.println("Введите re для числа " + (i + 1) + "в векторе 1:");

int re = console.nextInt();

System.out.println("Введите im для числа " + (i + 1) + "в векторе 1:");

int im = console.nextInt();

complex\_1.add(new Complex(re, im));

}

Vector<Complex> complex\_2 = new Vector<>();

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

System.out.println("Введите re для числа " + (i + 1) + "в векторе 2:");

int re = console.nextInt();

System.out.println("Введите im для числа " + (i + 1) + "в векторе 2:");

int im = console.nextInt();

complex\_2.add(new Complex(re, im));

}

PrintVector(complex\_1);

PrintVector(complex\_2);

Vector<Complex> result = SumVectors(complex\_1, complex\_2);

PrintVector(result);

} catch (InputMismatchException e) {

System.out.println("Неверный формат введенных данных: допустимы только целые числа");

} catch (OutOfMemoryError e) {

System.out.println("Не хватает памяти для списка");

} catch (ArithmeticException e) {

System.out.println(e);

} catch (Exception e) {

System.out.println("Что-то пошло не так");

} finally {

console.close();

}

}

private static void PrintVector(Vector<Complex> vector) {

System.out.print("[");

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

System.out.print(vector.get(i).toString());

if (i != vector.size() - 1) System.out.print(", ");

}

System.out.print("]\n");

}

private static Vector<Complex> SumVectors(Vector<Complex> a, Vector<Complex> b) {

Vector<Complex> result = new Vector<>();

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

result.add(a.get(i).Add(b.get(i)));

}

return result;

}

}

Приведем результаты обработки нескольких исключительных ситуаций.

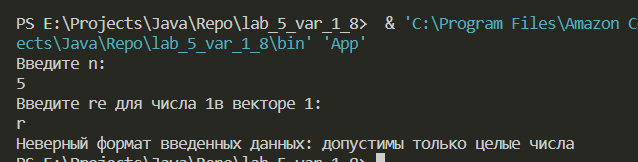


Рисунок – Результаты выполнения варианта задания 1

9. Определить класс Квадратное уравнение. Класс должен содержать несколько конструкторов. Реализовать методы для поиска корней, экстремумов, а также интервалов убывания/возрастания. Создать массив объектов и определить наибольшие и наименьшие по значению корни.

Листинг – Код основной программы

import java.util.ArrayList;

import java.util.Collections;

public class App {

public static void main(String[] args) throws Exception {

try {

SquareEquation one = new SquareEquation(1, 5, 1);

SquareEquation two = new SquareEquation(5, 2, 3);

SquareEquation three = new SquareEquation(6, -17, 12);

ArrayList<Double> roots = new ArrayList<>();

roots.addAll(one.FindRoots());

roots.addAll(two.FindRoots());

roots.addAll(three.FindRoots());

System.out.println(Collections.max(roots));

System.out.println(Collections.min(roots));

} catch (OutOfMemoryError e) {

System.out.println("Не хватает памяти для списка");

} catch (ArithmeticException e) {

System.out.println(e);

} catch (Exception e) {

System.out.println("Что-то пошло не так");

}

}

}

Листинг 4 – Код класса «Complex»

import java.util.ArrayList;

public class SquareEquation {

// ax^2+bx+c=0

int a, b, c;

public SquareEquation() {

a = 1;

b = 1;

c = 0;

}

public SquareEquation(int a, int b, int c) {

if (a == 0) throw new ArithmeticException("a == 0. Equation is not quadratic");

this.a = a;

this.b = b;

this.c = c;

}

public ArrayList<Double> FindRoots() {

double d = b \* b - 4.0 \* a \* c;

ArrayList<Double> result = new ArrayList<>();

if (d> 0.0)

{

double r1 = (-b + Math.pow(d, 0.5)) / (2.0 \* a);

double r2 = (-b - Math.pow(d, 0.5)) / (2.0 \* a);

result.add(r1);

result.add(r2);

System.out.println("The roots are " + r1 + " and " + r2);

}

else if (d == 0.0)

{

double r1 = -b / (2.0 \* a);

result.add(r1);

System.out.println("The root is " + r1);

}

else

{

System.out.println("Roots are not real.");

}

return result;

}

public void Intervals() {

double x0 = - b / (2 \* a);

// Print the values

if (a > 0)

{

// Open upward parabola function

System.out.print("Increase = "

+ "(" + x0 + "; +Inf)\n");

System.out.print("Decrease = "

+ "(-Inf; " + x0 + ")\n");

}

else if (a < 0)

{

System.out.print("Increase = "

+ "(-Inf; " + x0 + ")\n");

// Open upward parabola function

System.out.print("Decrease = "

+ "(" + x0 + "; +Inf)\n");

}

else

{

// If a=0 then it is not a quadratic function

System.out.print("Not a quadratic function\n");

}

}

public ArrayList<Double> Extremum() {

double secondPart = c \* 1.0 - (b \* b / (4.0 \* a));

ArrayList<Double> result = new ArrayList<>();

// Print the values

if (a > 0)

{

// Open upward parabola function

System.out.print("Maxvalue = "

+ "Infinity\n");

result.add(Double.POSITIVE\_INFINITY);

System.out.print("Minvalue = " + secondPart);

result.add(secondPart);

}

else if (a < 0)

{

// Open downward parabola function

System.out.print("Maxvalue = " + secondPart + "\n");

result.add(secondPart);

System.out.print("Minvalue = "

+ "-Infinity");

result.add(Double.POSITIVE\_INFINITY);

}

else

{

// If a=0 then it is not a quadratic function

System.out.print("Not a quadratic function\n");

}

return result;

}

}



Рисунок – Результаты обработки исключений

*Задание 2.*

Выполнить задания из варианта 2 лабораторной работы 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

8. Car: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер. Создать массив объектов. Вывести: a) список автомобилей заданной марки; b) список автомобилей заданной модели, которые эксплуатируются больше n лет; c) список автомобилей заданного года выпуска, цена которых больше указанной.

Листинг – Код измененного класса «Car» с добавленными вызовами исключений

import java.util.regex.Pattern;

public class Car {

int id;

String brand;

String model;

int year;

String color;

int price;

String number;

public void setId(int id) throws Exception {

if (id <= 0) throw new Exception("ID can't be 0 or negative");

this.id = id;

}

public void setBrand(String brand) throws Exception {

checkStringCorrectness(brand);

this.brand = brand;

}

public void setModel(String model) throws Exception {

checkStringCorrectness(model);

this.model = model;

}

public void setColor(String color) throws Exception {

checkStringCorrectness(color);

this.color = color;

}

public void setPrice(int price) throws Exception {

if (price < 0) throw new Exception("Price can't be negative");

this.price = price;

}

public void setNumber(String number) throws Exception {

if (!Pattern.matches("\\D{1}\\d{3}\\D{2}", number)) throw new Exception("Invalid car number format");;

this.number = number;

}

public void setYear(int year) {

this.year = year;

}

public int getId() {

return id;

}

public String getBrand() {

return brand;

}

public String getModel() {

return model;

}

public String getColor() {

return color;

}

public String getNumber() {

return number;

}

public int getPrice() {

return price;

}

public int getYear() {

return year;

}

@Override

public String toString() {

String result = "{ " +

"ID: " + this.id + "\n" +

"Brand: " + this.brand + "\n" +

"Model: " + this.model + "\n" +

"Year: " + this.year + "\n" +

"Color: " + this.color + "\n" +

"Price: " + this.price + "\n" +

"Number: " + this.number + "\n" +

"}";

return result;

}

private void checkStringCorrectness(String value) throws Exception {

if (value.equals("")) throw new Exception("Invalid value");

}

}

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

9. Product: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество. Создать массив объектов. Вывести: a) список товаров для заданного наименования; b) список товаров для заданного наименования, цена которых не превосходит заданную; c) список товаров, срок хранения которых больше заданного.

Листинг – Код измененного класса «Product» с добавленными вызовами исключений

public class Product {

int id;

String title;

int UPC;

String Manufacturer;

int cost;

int valid\_until;

int amount;

public Product(int id, String title, int UPC, String Manufacturer, int cost, int valid\_until, int amount) throws Exception {

checkStringCorrectness(title);

checkStringCorrectness(Manufacturer);

this.id = id;

this.title = title;

this.UPC = UPC;

this.Manufacturer = Manufacturer;

this.cost = cost;

this.valid\_until = valid\_until;

this.amount = amount;

}

public int getId() {

return id;

}

public String getTitle() {

return title;

}

public int getUPC() {

return UPC;

}

public String getManufacturer() {

return Manufacturer;

}

public int getCost() {

return cost;

}

public int getValid\_until() {

return valid\_until;

}

public int getAmount() {

return amount;

}

public void setId(int id) {

this.id = id;

}

public void setTitle(String title) throws Exception {

checkStringCorrectness(title);

this.title = title;

}

public void setUPC(int uPC) {

UPC = uPC;

}

public void setManufacturer(String manufacturer) throws Exception {

checkStringCorrectness(manufacturer);

Manufacturer = manufacturer;

}

public void setCost(int cost) {

this.cost = cost;

}

public void setValid\_until(int valid\_until) {

this.valid\_until = valid\_until;

}

public void setAmount(int amount) {

this.amount = amount;

}

@Override

public String toString() {

return "{" +

" id='" + getId() + "'" +

", title='" + getTitle() + "'" +

", UPC='" + getUPC() + "'" +

", Manufacturer='" + getManufacturer() + "'" +

", cost='" + getCost() + "'" +

", valid\_until='" + getValid\_until() + "'" +

", amount='" + getAmount() + "'" +

"}";

}

private void checkStringCorrectness(String value) throws Exception {

if (value.equals("")) throw new Exception("Invalid value");

}

}

Приведем результаты выполнения работы программы при введенных некорректных данных.

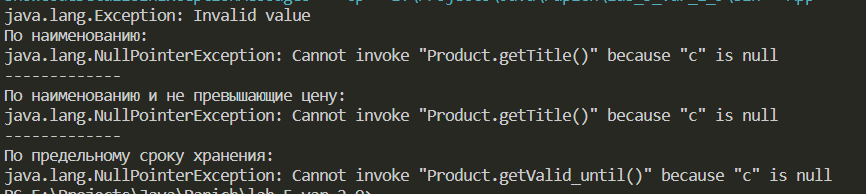


Рисунок – Обработка исключения

*Задание 3.*

В следующих заданиях требуется ввести последовательность строк из текстового потока и выполнить указанные действия. При этом могут рассматриваться два варианта:

– каждая строка состоит из одного слова;

– каждая строка состоит из нескольких слов.

Имена входного и выходного файлов, а также абсолютный путь к ним могут быть введены как параметры командной строки или храниться в файле.

8. Определить частоту повторяемости букв и слов в стихотворении Александра Пушкина.

9. Входной файл содержит совокупность строк. Строка файла содержит строку квадратной матрицы. Ввести матрицу в двумерный массив (размер матрицы найти). Вывести исходную матрицу и результат ее транспонирования.

Листинг 7 – Код основной программы

import java.io.File;

import java.io.FileNotFoundException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;

public class App {

public static void main(String[] args) throws Exception {

// Работаем с Пушкиным

try {

System.out.println("Отрабатываем стихотворение А. Пушкина");

File poem = new File("data\\poem\_pushkin.txt");

HashMap<String, Integer> words = new HashMap<>();

HashMap<Character, Integer> characters = new HashMap<>();

Scanner reader = new Scanner(poem, "UTF-8");

while (reader.hasNextLine()) {

System.out.print(".");

String data = removeAllNonAlphaNumeric(reader.nextLine());

String[] splitted = data.split("\\s");

for (String w:splitted) {

if (w.equals("")) continue;

int count = words.containsKey(w) ? words.get(w) : 0;

words.put(w, count + 1);

for (Character c:w.toCharArray()) {

int c\_count = characters.containsKey(c) ? characters.get(c) : 0;

characters.put(c, c\_count + 1);

}

}

}

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

System.out.println("Стихотворение Пушкина успешно обработано");

System.out.println("Подсчет слов:");

words.entrySet().forEach(entry -> {

System.out.println(entry.getKey() + " " + entry.getValue());

});

System.out.println("Подсчет символов:");

characters.entrySet().forEach(entry -> {

System.out.println(entry.getKey() + " " + entry.getValue());

});

} catch (FileNotFoundException e) {

System.out.println("Файл не найден");

e.printStackTrace();

}

try {

System.out.println("Отрабатываем матрицу");

File poem = new File("data\\matrix.txt");

Scanner reader = new Scanner(poem, "UTF-8");

ArrayList<String> rows = new ArrayList<>();

while (reader.hasNextLine()) {

System.out.print(".");

rows.add(reader.nextLine());

}

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

String[] splitted = rows.get(0).split(" ");

int n = splitted.length;

int[][] matrix = new int[n][n];

int i = 0;

for (String r:rows) {

String[] row = r.split(" ");

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

matrix[i][j] = Integer.parseInt(row[j]);

}

i++;

}

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

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

System.out.print(matrix[k][m] + " ");

}

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

}

reader.close();

} catch (FileNotFoundException e) {

System.out.println("Файл не найден");

e.printStackTrace();

} catch (Exception e) {

}

}

static String removeAllNonAlphaNumeric(String s) {

if (s == null) {

return null;

}

return s.replaceAll("[^\\wА-Яа-я\\s]", "");

}

}

Приведем результаты выполнения разработанной программы. Покажем получившиеся в результате файлы.

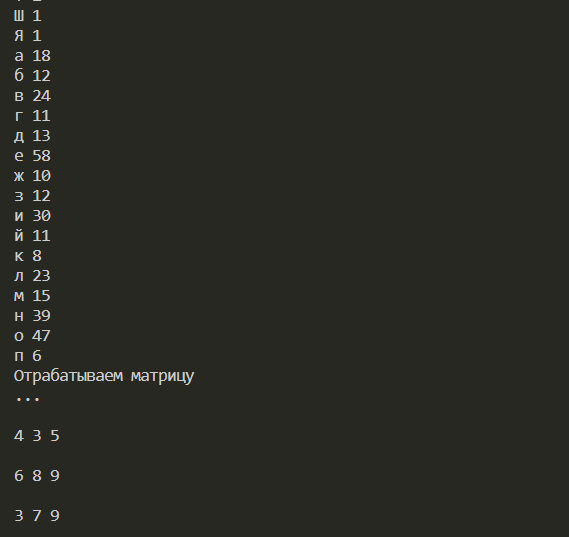


Рисунок – Результаты подсчета слов

*Задание 4.*

При выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File

8. Из текста Java-программы удалить все виды комментариев.

9. Прочитать строки из файла и поменять местами первое и последнее слова в каждой строке.

Листинг 8 – Код программы

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileWriter;

import java.io.IOException;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.Scanner;

public class App {

public static void main(String[] args) throws Exception {

try {

// Получаем путь файла

Path inputPath = Paths.get("data\\Hello.java");

// Читаем все строки файла с кодом программы

byte[] content = Files.readAllBytes(inputPath);

String program = new String(content);

// Обрабатываем строки - удаляем комментарии

String out = program.trim()

.replaceAll("/\\\*([^\*]|[\\r\\n]|(\\\*+([^\*/]|[\\r\\n])))\*\\\*+/", "") // Многострочные

.replaceAll("\\/\\/(.\*)]\*", ""); // Однострочные

// Создаем новую директорию, если не существует

Files.createDirectories(Paths.get("output"));

// Задаем путь для нового файла для вывода

Path outPath = Paths.get("output\\Hello\_linted.java");

// Записываем в новый файл получившийся код

Files.write(outPath, out.getBytes());

} catch (FileNotFoundException e) {

System.out.println("Файл не найден");

e.printStackTrace();

} catch (IOException e) {

System.out.println("Произошла ошибка I/O");

e.printStackTrace();

}

try {

// Получаем путь файла

System.out.println("Отрабатываем слова");

File poem = new File("data\\strings.txt");

Scanner reader = new Scanner(poem, "UTF-8");

ArrayList<String> rows = new ArrayList<>();

while (reader.hasNextLine()) {

System.out.print(".");

rows.add(reader.nextLine());

}

// Создаем новую директорию, если не существует

Files.createDirectories(Paths.get("output"));

// Задаем путь для нового файла для вывода

FileWriter writer = new FileWriter("output\\strings\_ch.txt", StandardCharsets.UTF\_8);

for (String r:rows) {

String[] splitted = r.split(" ");

String first = splitted[0];

splitted[0] = splitted[splitted.length - 1];

splitted[splitted.length - 1] = first;

for (String s:splitted) {

writer.write(s + " ");

}

writer.write("\n");

}

writer.close();

} catch (FileNotFoundException e) {

System.out.println("Файл не найден");

e.printStackTrace();

} catch (IOException e) {

System.out.println("Произошла ошибка I/O");

e.printStackTrace();

}

}

}

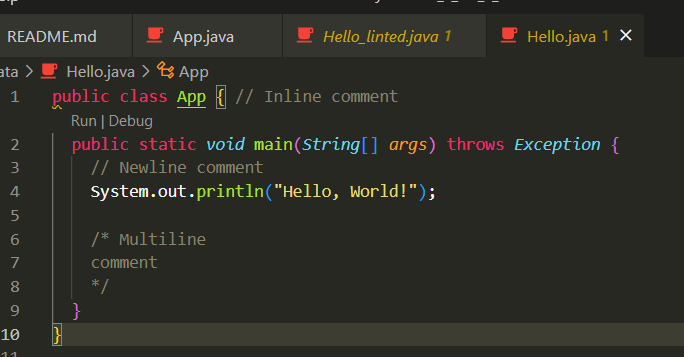


Рисунок – Исходный файл

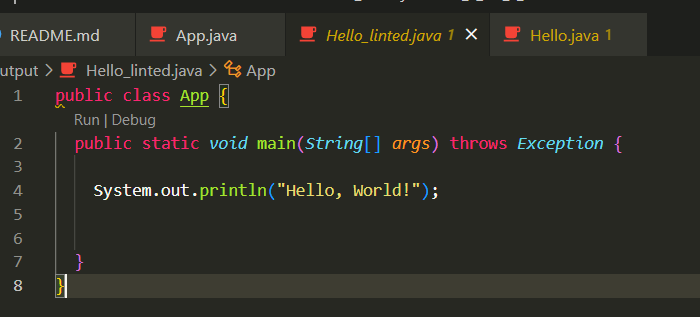


Рисунок – Результирующий файл

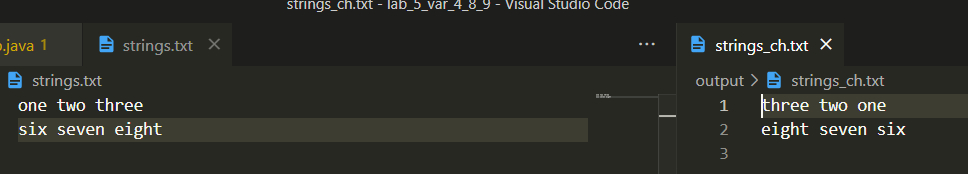


Рисунок – Результат смены слов местами

## Вывод

По итогам выполнения лабораторной работы были получены навыки программирования с использованием исключений в языке Java. Также был получен опыт работы с файлами и файловой системой.