



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

Название: Исключения. Файлы

Дисциплина: Языки программирования для работы с большими
данными

Студент ИУ6-22М

Д. Р. Григорян

Преподаватель

П.В. Степанов
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Цель работы:

Ознакомиться с языком программирования Java и научиться разрабатывать собственные обработчики исключений и исключения ввода/вывода, работать с текстовым потоком.

Выполнение:

Задача 1.1:

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.

Задание: Определить класс Дробь в виде пары (m,n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения и деления дробей. Объявить массив из k дробей, ввести/вывести значения для массива дробей. Создать массив объектов и передать его в метод, который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива.

Листинг 1 класса Fraction:

```
class Fraction{ // имя

    int m,n; // числитель и знаменатель
    public Fraction() {}

    public Fraction(int m, int n) {
        if ((m==0) || (n==0)) {
            throw new ArithmeticException("Одно из значений дроби равно 0");
        }
        this.m = m;
        this.n = n;
    }

    private static int gcd(int a, int b) // Greatest Common Divisor
    {
        while (b > 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }
}
```

```

private static int gcd(int[] input) { // Greatest Common Divisor
    int result = input[0];
    for (int i = 1; i < input.length; i++)
        result = gcd(result, input[i]);
    return result;
}
private static int lcm(int a,int b){
    return a / gcd(a,b) * b;
}
private static int lcm(int[] input) {
    int result = input[0];
    for (int i = 1; i < input.length; i++) {
        if ((input[i] == 0 || input[0] == 0)
            || (input[i] < 0 || input[0] < 0))
            break;
        result = lcm(result, input[i]);
    }
    return result;
}

void displayInfo(Fraction res,String operation){
    System.out.println(res.m);
    System.out.print(res.n);
    System.out.println(operation);
}

public Fraction sum(Fraction first, Fraction second){
    Fraction result = new Fraction();
    if (first.n==second.n) {
        result.m = first.m + second.m;
        result.n = first.n;
    }
    else {
        int LCM = lcm(first.n,second.n);
        result.m =first.m*(LCM/ first.n)+second.m*(LCM / second.n);
        result.n = LCM;
    }
    displayInfo(result," Sum");
    return result;
}

public void edit(Fraction[] array_for_edit){
    for (int i=0;i<array_for_edit.length-1;i++) {
        if (i % 2 == 0) {
            array_for_edit[i] =
sum(array_for_edit[i],array_for_edit[i+1]);
        }
    }
    //displayInfo(result," Sum");
}

public void sub(Fraction first, Fraction second){
    Fraction result = new Fraction();
    if (first.n==second.n) {
        result.m = first.m - second.m;
        result.n = first.n;
    }
    else {
        int LCM = lcm(first.n,second.n);
        result.m =first.m*(LCM/ first.n)-second.m*(LCM / second.n);
        result.n = LCM;
    }
    displayInfo(result," Sub");
}
}

```

```

    public void mul(Fraction first, Fraction second){
        Fraction result = new Fraction();
        result.n= first.n* second.n;
        result.m= first.m* second.m;
        displayInfo(result," Mul/Div");
    }
    public void div(Fraction first, Fraction second){
        if ((second.m==0) || (second.n==0)) {
            throw new ArithmeticException("Знаменатель второй дроби = 0,
деление невозможно");
        }
        Fraction doubler = new Fraction();
        doubler.m = second.n;
        doubler.n = second.m;
        mul(first,doubler);
    }
}

```

Листинг 2 программы main:

```

import java.util.Scanner;
//7.Определить класс Дробь в виде пары (m,n).
// Класс должен содержать несколько конструкторов. Реализовать методы для
сложения, вычитания, умножения и деления дробей.
// Объявить массив из k дробей, ввести/вывести значения для массива дробей.
// Создать массив объектов и передать его в метод, который изменяет каждый
элемент массива с четным индексом
// путем добавления следующего за ним элемента массива.
public class Main {
    public static void main(String[] args) {
        Fraction tom = new Fraction(1, 4); // вызов второго конструктора с
одним параметром
        Fraction andy = new Fraction(2, 6);
        tom.sum(tom, andy);
        tom.sub(tom, andy);
        tom.mul(tom, andy);
        tom.div(tom, andy);

        Scanner in = new Scanner(System.in);
        int num = 0;
        while (true) {
            System.out.print("Input a number of Fractions for array: ");
            num = in.nextInt();
            if (num > 0) {
                break;
            }
        }
        int nums[] = new int[num * 2];
        System.out.println("Input one at the time int numbers: ");
        for (int i = 0; i < nums.length; i++) {
            nums[i] = in.nextInt();
        }
        Fraction[] arr = new Fraction[num];
        int k = 0;
        System.out.println("The input array of fractions:");
        try {
            for (int i = 0; i < nums.length - 1; i += 2) {
                int numerator = nums[i];
                int determinator = nums[i + 1];
                arr[k] = new Fraction(numerator, determinator);
                System.out.println(arr[k].m);
            }
        }
    }
}

```

```

        System.out.println(arr[k].n);
        System.out.println("---");
        k++;
    }
}
catch (OutOfMemoryError e) {
    System.err.println("Недостаточно памяти");
    e.printStackTrace();
}
Fraction alex = new Fraction();
alex.edit(arr);
System.out.println("The edit array of fractions:");
for (int i=0;i<arr.length;i++){
    System.out.println(arr[i].m);
    System.out.println(arr[i].n);
    System.out.println("---");
}
}
}

```

Результат приведен на рисунке 1:

```

Exception in thread "main" java.lang.ArithmeticException Create breakpoint : Одно из значений дроби равно 0
    at Fraction.<init>(Fraction.java:8)
    at Main.main(Main.java:9)

Process finished with exit code 1

```

Рисунок 1 – Результат выполнения программы

Задача 1.2:

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.

Задание: 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 3 класса Complex :

```

class Complex {
    private double real;
    private double imaginary;
    public Complex() {

```

```

        this.real = 0;
        this.imaginary = 0;
    }
    public Complex(double real1, double imaginary1) {
        this.real = real;
        this.imaginary = imaginary;
    }
    public Complex add(Complex other) {
        return new Complex(this.real + other.real, this.imaginary +
other.imaginary);
    }
    public Complex subtract(Complex other) {
        return new Complex(this.real - other.real, this.imaginary -
other.imaginary);
    }
    public Complex multiply(Complex other) {
        return new Complex(this.real * other.real - this.imaginary *
other.imaginary,
            this.real * other.imaginary + this.imaginary * other.real);
    }
    public Complex divide(Complex other) {
        double denominator = other.real * other.real + other.imaginary *
other.imaginary;
        return new Complex((this.real * other.real + this.imaginary *
other.imaginary) / denominator,
            (this.imaginary * other.real - this.real * other.imaginary) /
denominator);
    }
    public void setReal(double real) {
        this.real = real;
    }
    public void setImaginary(double imaginary) {
        this.imaginary = imaginary;
    }
    public double getReal() {
        return this.real;
    }
    public double getImaginary() {
        return this.imaginary;
    }
    public String toString() {
        if (this.imaginary < 0) {
            return this.real + " - " + Math.abs(this.imaginary) + "i";
        } else {
            return this.real + " + " + this.imaginary + "i";
        }
    }
}

```

Листинг 4 программы main:

```

public class Main {
    public static void main(String[] args) {
        int n = 3;
        Complex[] vector1 = new Complex[n];
        Complex[] vector2 = new Complex[n];
        try {
            for (int i = 0; i < n; i++) {
                vector1[i] = new Complex(i,i+1);
                vector2[i] = new Complex(i + 1, i);
            }
        }
    }
}

```

```

        catch (OutOfMemoryError e) {
            System.err.println("Недостаточно памяти");
            e.printStackTrace();
        }
        Complex[] result = new Complex[n];
        for (int i = 0; i < n; i++) {
            result[i] = vector1[i].add(vector2[i]);
        }
        System.out.println("Vector 1: ");
        for (int i = 0; i < n; i++) {
            System.out.println(vector1[i].toString());
        }
        System.out.println("Vector 2: ");
        for (int i = 0; i < n; i++) {
            System.out.println(vector2[i].toString());
        }
        System.out.println("Result: ");
        for (int i = 0; i < n; i++) {
            System.out.println(result[i].toString());
        }
    }
}

```

Результат приведен на рисунке 2:

```

Vector 1:
0.0 + 2.0i
1.0 + 3.0i
2.0 + 4.0i
Vector 2:
2.0 + 0.0i
3.0 + 1.0i
4.0 + 2.0i
Result:
2.0 + 2.0i
4.0 + 4.0i
6.0 + 6.0i

```

Рисунок 2 – Результат выполнения программы

Задача 2.1:

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

Задание: Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время городских и междугородных разговоров. Создать массив объектов. Вывести: а) сведения об абонентах, у которых время внутригородских разговоров превышает заданное; б) сведения об абонентах, которые пользовались междугородной связью; в) сведения об абонентах в алфавитном порядке.

Листинг 5 класса Company_result:

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

class Company_result {
    private ArrayList<Phone_Operator> list_result = new ArrayList<>();
    void add(Phone_Operator phone) {
        list_result.add(phone);
    }
    void show() throws OperatorDataException, InvalidCredentialsException {
        for (Phone_Operator client : list_result){
            System.out.println(client);
            if (client.getDebet() <= 0 || client.getCredit() <= 0 ) throw new
OperatorDataException("Problem with cards Debet " +
                "or credit");
            if (client.getLast_name() == "" || client.getFirst_name() == "" )
throw new InvalidCredentialsException("Problem with" +
                " your credentials" );
        }
    }
    public ArrayList<Phone_Operator> getSortByName() {
        ArrayList<Phone_Operator> tempList = new ArrayList<Phone_Operator>();
        Comparator<Phone_Operator> countryModelsComparator
            = Comparator.comparing(Phone_Operator::getLast_name);

        Collections.sort(list_result, countryModelsComparator);
        return tempList;
    }
    public ArrayList<Phone_Operator> getClientsLocalLimitsTime(int time) {
        ArrayList<Phone_Operator> tempList = new ArrayList<Phone_Operator>();
        for (Phone_Operator client : list_result){
            if (client.getIntercity_time() > time){
                tempList.add(client);
            }
        }
        return tempList;
    }
    public ArrayList<Phone_Operator> getClientsByInternationTime(int
inter_time) {
        ArrayList<Phone_Operator> tempList = new ArrayList<Phone_Operator>();
        for (Phone_Operator client : list_result){
            if (client.getIntercity_time() > inter_time){
                tempList.add(client);
            }
        }
        return tempList;
    }
}
```

Листинг 6 класса исключения InvalidCredentialsException:

```
class InvalidCredentialsException extends Exception{
    public InvalidCredentialsException(String message){
        super(message);
    }
}
```


Листинг 7 класса исключения OperatorDataException:

```
class OperatorDataException extends Exception{  
  
    public OperatorDataException(String message){  
        super(message);  
    }  
}
```

Результат приведен на рисунке 3:

```
Unsorted list of clients  
Phone_Operator{all_clients=6, id=0, Last_name='Grigoryan', First_name='David', Patron='Rube  
Phone_Operator{all_clients=6, id=1, Last_name='Askerova', First_name='Nargiz', Patron='Alex  
Phone_Operator{all_clients=6, id=2, Last_name='Zamula', First_name='Margo', Patron='Viktoro  
Exception in thread "main" OperatorDataException: Problem with cards Debet or credit  
    at Company_result.show(Company_result.java:13)  
    at Main.main(Main.java:31)  
  
Process finished with exit code 1
```

Рисунок 3 – Результат выполнения программы

Задача 2.2:

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

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

Листинг 8 класса исключения InvalidYearException :

```
public class InvalidYearException extends Exception {  
  
    public InvalidYearException(String message) {  
        super(message);  
    }  
}
```

Листинг 9 класса исключения InvalidDataException:

```
public class InvalidDataException extends Exception{  
    public InvalidDataException(String message) {  
        super(message);  
    }  
}
```

Листинг 10 программы main:

```
public class Main {
    public static void main(String[] args) throws
InvalidDataException, InvalidYearException {
        Car[] cars = {
            new Car(1, "Toyota", "Camry", 2018, "Silver", 25000,
"ABC123"),
            new Car(2, "Honda", "Civic", 2019, "Red", 20000, "DEF456"),
            new Car(3, "", "Corolla", 2017, "Black", 22000, "GHI789"),
            new Car(4, "Ford", "Mustang", 2021, "Yellow", 30000,
"JKL012"),
            new Car(5, "Chevrolet", "Camaro", 2016, "White", 28000,
"MNO345")
        };

        CarManager cm = new CarManager(cars);

        for (Car car : cars) {
            if (car.getYear() >= 2024) throw new InvalidYearException("The
car is older than 2023 year of " +
"production has not yet been released");
            if (car.getBrand().isEmpty()) throw new InvalidDataException("There is
incorrect name of brand");
            if (car.getPrice() <= 0) throw new InvalidDataException("Free car
or negative price? It is not legal...");
        }
        // Get cars by brand
        Car[] toyotas = cm.getCarsByBrand("Toyota");
        System.out.println("Cars by brand:");
        for (Car car : toyotas) {
            System.out.println(car.toString());
        }

        // Get cars by model and age
        Car[] oldCivics = cm.getCarsByModelAndAge("Civic", 3);
        System.out.println("\nOld cars by model:");
        for (Car car : oldCivics) {
            System.out.println(car.toString());
        }

        // Get cars by year and price
        Car[] expensive2016 = cm.getCarsByYearAndPrice(2016, 25000);
        System.out.println("\nExpensive cars by year:");
        for (Car car : expensive2016) {
            System.out.println(car.toString());
        }
    }
}
```

Результат приведен на рисунках 4-5:

```
C:\Users\sonar\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022
Exception in thread "main" InvalidYearException: The car is older than 2023 year of production has not yet been released
    at Main.main(Main.java:15)

Process finished with exit code 1
```

Рисунок 4 – Результат выполнения программы

```
C:\Users\sonar\.jdk\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBra
Exception in thread "main" InvalidDataException: There is incorrect name of brand
    at Main.main(Main.java:17)

Process finished with exit code 1
```

Рисунок 5 – Результат выполнения программы

Задача 3.1:

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.

В каждом слове стихотворения Николая Заболоцкого заменить первую букву слова на прописную

Листинг 11 программы main:

```
// В каждом слове стихотворения Николая Заболоцкого заменить первую букву
// слова на прописную.
import java.io.*;
import java.util.ArrayList;

public class Main {

    public static void main(String[] args) throws Exception {
        File file = new File("Zabolo.txt");

        FileReader fr = new FileReader(file);
        BufferedReader reader = new BufferedReader(fr);
        FileWriter fw = new FileWriter("Result.txt");
        char [] a = new char[256];

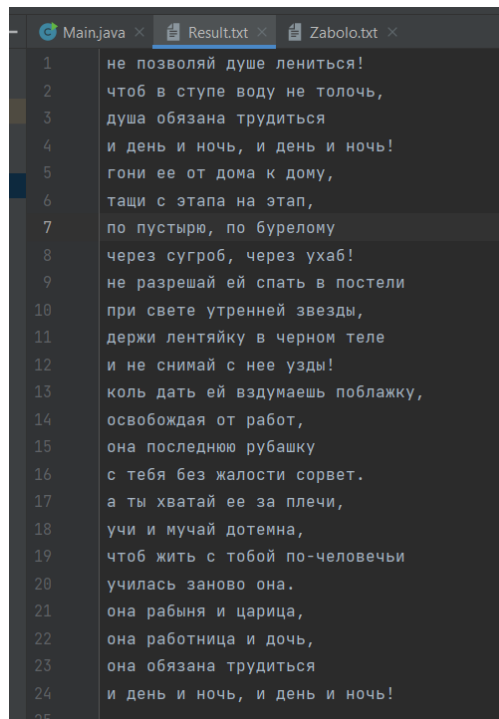
        String line = reader.readLine();
        while (line != null) {
            String words[] = line.split(" ");
            words[0] = words[0].toLowerCase();
            String lineNew = String.join(" ", words);
            lineNew = lineNew.concat("\n");
            fw.write(lineNew);
            line = reader.readLine();
        }
        fr.close();
        fw.close();
    }
}
```

```
}  
}
```

Листинг 12 файла ввода:

```
Не позволяй душе лениться!  
Чтоб в ступе воду не толочь,  
Душа обязана трудиться  
И день и ночь, и день и ночь!  
Гони ее от дома к дому,  
Тащи с этапа на этап,  
По пустырю, по бурелому  
Через сугроб, через ухаб!  
Не разрешай ей спать в постели  
При свете утренней звезды,  
Держи лентяйку в черном теле  
И не снимай с нее узды!  
Коль дать ей вздумаешь поблажку,  
Освобождая от работ,  
Она последнюю рубашку  
С тебя без жалости сорвет.  
А ты хватай ее за плечи,  
Учи и мучай дотемна,  
Чтоб жить с тобой по-человечьи  
Училась заново она.  
Она рабыня и царица,  
Она работница и дочь,  
Она обязана трудиться  
И день и ночь, и день и ночь!
```

Результат приведен на рисунке 6:



```
1  не позволяй душе лениться!  
2  чтоб в ступе воду не толочь,  
3  душа обязана трудиться  
4  и день и ночь, и день и ночь!  
5  гони ее от дома к дому,  
6  тащи с этапа на этап,  
7  по пустырю, по бурелому  
8  через сугроб, через ухаб!  
9  не разрешай ей спать в постели  
10 при свете утренней звезды,  
11 держи лентяйку в черном теле  
12 и не снимай с нее узды!  
13 коль дать ей вздумаешь поблажку,  
14 освобождая от работ,  
15 она последнюю рубашку  
16 с тебя без жалости сорвет.  
17 а ты хватай ее за плечи,  
18 учи и мучай дотемна,  
19 чтоб жить с тобой по-человечьи  
20 училась заново она.  
21 она рабыня и царица,  
22 она работница и дочь,  
23 она обязана трудиться  
24 и день и ночь, и день и ночь!  
25
```

Рисунок 6 – Результат выполнения программы

Задача 3.2:

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.

Определить частоту повторяемости букв и слов в стихотворении Александра Пушкина.

Листинг 13 программы main:

```
// Определить частоту повторяемости букв и слов в стихотворении Александра
Пушкина
import java.io.*;
import java.nio.file.Files;
import java.util.*;

public class Main {

    public static void main(String[] args) throws Exception {
        var file = new File("Pushkin.txt");
        var file2 = new File("Result.txt");

        var fr = new FileReader(file);
        var reader = new BufferedReader(fr);
        var fw = new FileWriter("Result.txt");
        Map<String, Integer> uniqueWords = new HashMap<String, Integer>();
        Map<Character, Integer> uniqueLetters = new HashMap<Character,
Integer>();

        String line = reader.readLine();
        while (line != null) {
            String words[] = line.split(" |, |; " );

            for (String word: words) {
                word = word.toLowerCase();
                if (word=="") {
                    break;
                } else if (!uniqueWords.containsKey(word)) {
                    uniqueWords.put(word,1);
                }
                else {
                    uniqueWords.put(word,uniqueWords.get(word)+1);
                }
                char[] str = word.toCharArray();
                for (int i=0;i<word.length();i++) {
                    if (!uniqueLetters.containsKey(str[i])) {
                        uniqueLetters.put(str[i],1);
                    }
                    else {
                        uniqueLetters.put(str[i],uniqueLetters.get(str[i])+1);
                    }
                }
            }
        }
    }
}
```

```

    }
}

}
line = reader.readLine();
}
BufferedWriter bf = null;
try {
    // create new BufferedWriter for the output file
    bf = new BufferedWriter(new FileWriter(file2));

    // iterate map entries
    for (Map.Entry<String, Integer> entry :
        uniqueWords.entrySet()) {

        // put key and value separated by a colon
        bf.write(entry.getKey() + ":"
            + entry.getValue());

        // new line
        bf.newLine();
    }
    bf.flush();
    bf.newLine();
    for (Map.Entry<Character, Integer> entry :
        uniqueLetters.entrySet()) {

        // put key and value separated by a colon
        bf.write(entry.getKey() + ":"
            + entry.getValue());

        // new line
        bf.newLine();
    }
    bf.flush();
}
catch (IOException e) {
    e.printStackTrace();
}
fr.close();
fw.close();
}
}

```

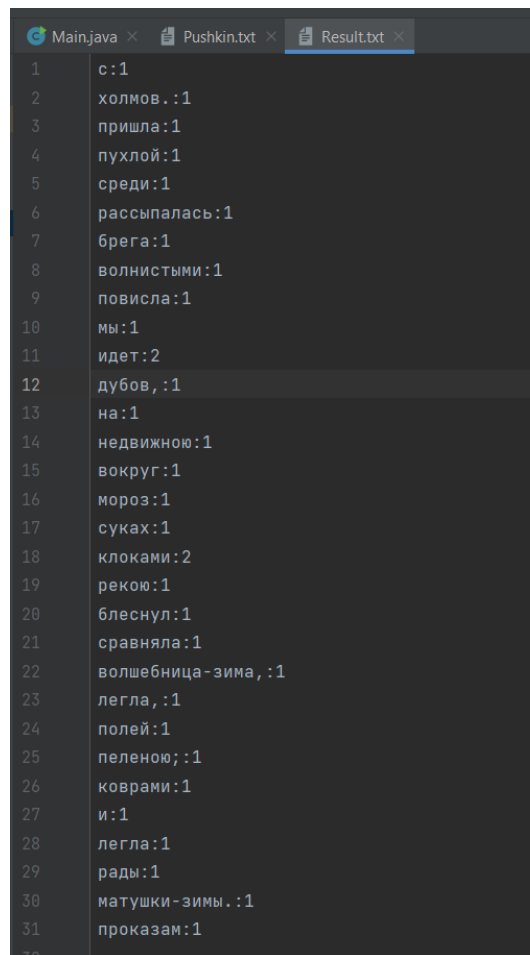
Листинг 14 файла ввода:

```

Идет волшебница-зима,
Пришла, рассыпалась; клоками
Повисла на суках дубов,
Легла волнистыми коврами
Среди полей вокруг холмов.
Берега с недвижною рекою
Идет клоками легла,
Сравняла пухлой пеленою;
Блеснул мороз, и рады мы
Проказам матушки-зимы.

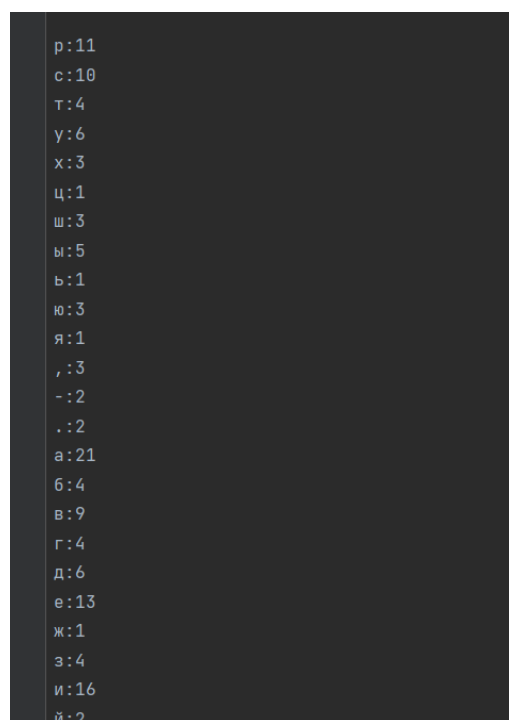
```

Результат приведен на рисунке 7-8:



```
1 с:1
2 холмов.:1
3 пришла:1
4 пухлой:1
5 среди:1
6 рассыпалась:1
7 берега:1
8 волнистыми:1
9 повисла:1
10 мы:1
11 идет:2
12 дубов,:1
13 на:1
14 недвижимую:1
15 вокруг:1
16 мороз:1
17 суках:1
18 клоками:2
19 рекою:1
20 блеснул:1
21 сравнила:1
22 волшебница-зима,:1
23 легла,:1
24 полей:1
25 пеленою,:1
26 коврами:1
27 и:1
28 легла:1
29 рады:1
30 матушки-зимы.:1
31 проказам:1
```

Рисунок 7 – Результат выполнения программы



```
р:11
с:10
т:4
у:6
х:3
ц:1
ш:3
ы:5
ь:1
ю:3
я:1
,:3
-:2
.:2
а:21
б:4
в:9
г:4
д:6
е:13
ж:1
з:4
и:16
й:2
```

Рисунок 8 – Результат выполнения программы

Задача 4.1:

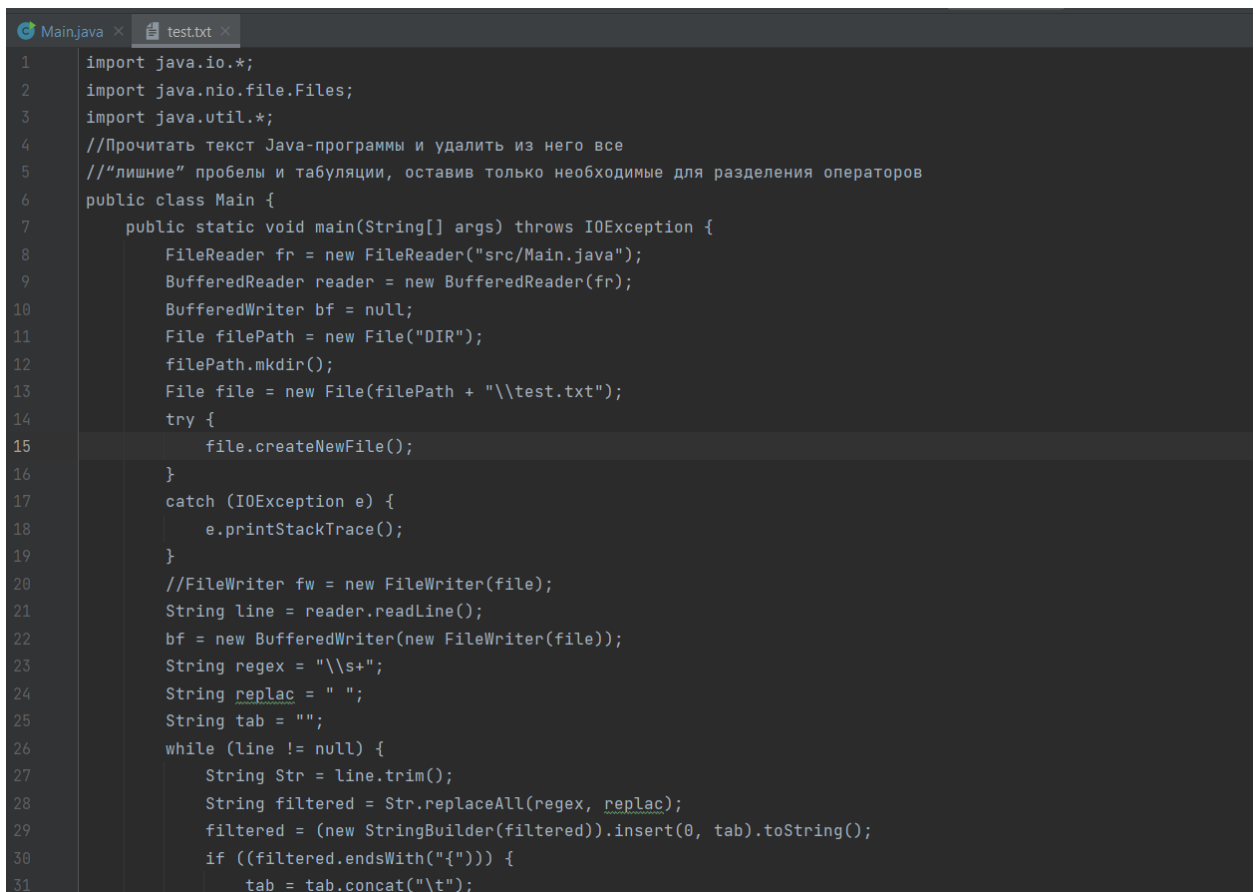
При выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File.

Прочитать текст Java-программы и удалить из него все “лишние” пробелы и табуляции, оставив только необходимые для разделения операторов.

Листинг 15 программы main:

```
import java.io.*;
import java.nio.file.Files;
import java.util.*;
//Прочитать текст Java-программы и удалить из него все
//“лишние” пробелы и табуляции, оставив только необходимые для разделения
операторов
public class Main {
    public static void main(String[] args) throws IOException {
        FileReader fr = new FileReader("src/Main.java");
        BufferedReader reader = new BufferedReader(fr);
        BufferedWriter bf = null;
        File filePath = new File("DIR");
        filePath.mkdir();
        File file = new File(filePath + "\\test.txt");
        try {
            file.createNewFile();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        //FileWriter fw = new FileWriter(file);
        String line = reader.readLine();
        bf = new BufferedWriter(new FileWriter(file));
        String regex = "\\s+";
        String replac = " ";
        String tab = "";
        while (line != null) {
            String Str = line.trim();
            String filtered = Str.replaceAll(regex, replac);
            filtered = (new StringBuilder(filtered)).insert(0,
tab).toString();
            if ((filtered.endsWith("{")) {
                tab = tab.concat("\t");
            }
            else if (filtered.endsWith("}")) {
                filtered = filtered.substring(1);
                tab = tab.substring(1);
            }
            bf.write(filtered);
            bf.newLine();
            bf.flush();
            line = reader.readLine();
        }
    }
}
```

Результат приведен на рисунке 9:



```

1  import java.io.*;
2  import java.nio.file.Files;
3  import java.util.*;
4  //Прочитать текст Java-программы и удалить из него все
5  //“лишние” пробелы и табуляции, оставив только необходимые для разделения операторов
6  public class Main {
7      public static void main(String[] args) throws IOException {
8          FileReader fr = new FileReader("src/Main.java");
9          BufferedReader reader = new BufferedReader(fr);
10         BufferedWriter bf = null;
11         File filePath = new File("DIR");
12         filePath.mkdir();
13         File file = new File(filePath + "\\test.txt");
14         try {
15             file.createNewFile();
16         }
17         catch (IOException e) {
18             e.printStackTrace();
19         }
20         //FileWriter fw = new FileWriter(file);
21         String line = reader.readLine();
22         bf = new BufferedWriter(new FileWriter(file));
23         String regex = "\\s+";
24         String replac = " ";
25         String tab = "";
26         while (line != null) {
27             String Str = line.trim();
28             String filtered = Str.replaceAll(regex, replac);
29             filtered = (new StringBuilder(filtered)).insert(0, tab).toString();
30             if ((filtered.endsWith("{")))) {
31                 tab = tab.concat("\t");

```

Рисунок 9 – Результат выполнения программы

Задача 4.2:

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

Из текста Java-программы удалить все виды комментариев.

Листинг 16 программы main:

```

import java.io.*;
import java.util.*;
//Из текста Java-программы удалить все виды комментариев
public class Main {
    public static void main(String[] args) throws IOException {
        FileReader fr = new FileReader("src/Main.java");
        BufferedReader reader = new BufferedReader(fr);
        BufferedWriter bf;
        File filePath = new File("DIR");
        filePath.mkdir();
        File file = new File(filePath + "\\test5.8.txt");
        /*Из текста Java-программы удалить все виды комментариев
        Из текста Java-программы удалить все виды комментариев
        Из текста Java-программы удалить все виды комментариев
        Из текста Java-программы удалить все виды комментариев
        Из текста Java-программы удалить все виды комментариев*/
        try {
            file.createNewFile();
        }
        catch (IOException e) {

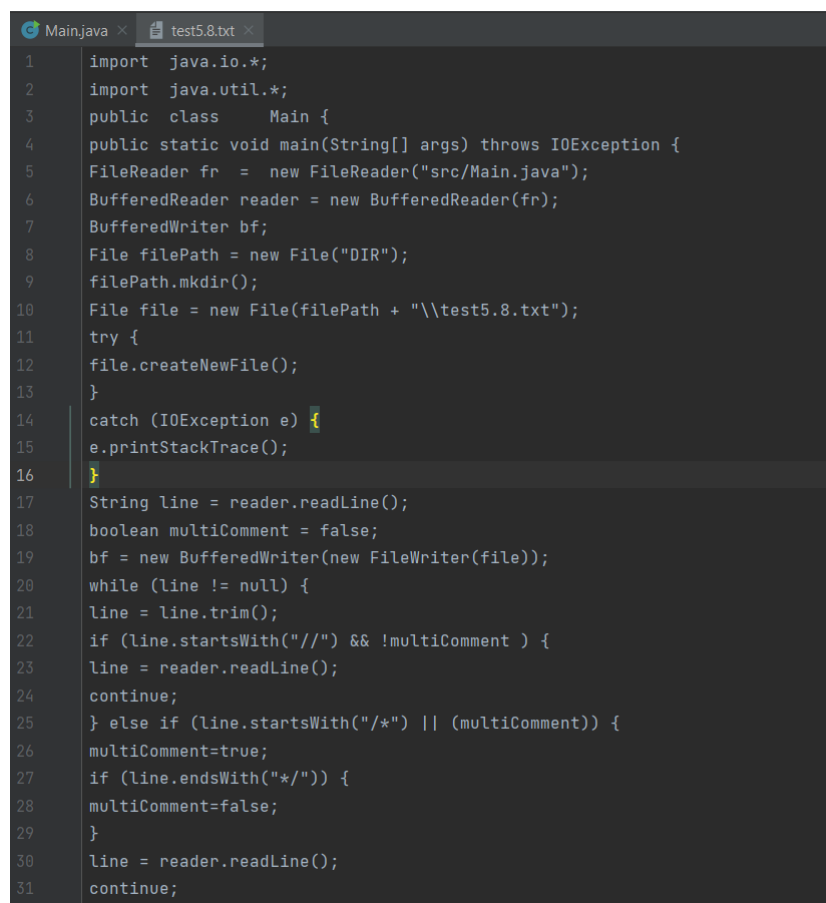
```

```

        e.printStackTrace();
    }
    //FileWriter fw = new FileWriter(file);
    String line = reader.readLine();
    boolean multiComment = false;
    bf = new BufferedWriter(new FileWriter(file));
    while (line != null) {
        line = line.trim();
        if (line.startsWith("//") && !multiComment ) {
            line = reader.readLine();
            continue;
        } else if (line.startsWith("/*") || (multiComment)) {
            multiComment=true;
            if (line.endsWith("*/")) {
                multiComment=false;
            }
            line = reader.readLine();
            continue;
        }
        bf.write(line);
        bf.newLine();
        bf.flush();
        line = reader.readLine();
    }
}
}

```

Результат приведен на рисунке 10:



```

1  import java.io.*;
2  import java.util.*;
3  public class Main {
4  public static void main(String[] args) throws IOException {
5  FileReader fr = new FileReader("src/Main.java");
6  BufferedReader reader = new BufferedReader(fr);
7  BufferedWriter bf;
8  File filePath = new File("DIR");
9  filePath.mkdir();
10 File file = new File(filePath + "\\test5.8.txt");
11 try {
12 file.createNewFile();
13 }
14 catch (IOException e) {
15 e.printStackTrace();
16 }
17 String line = reader.readLine();
18 boolean multiComment = false;
19 bf = new BufferedWriter(new FileWriter(file));
20 while (line != null) {
21 line = line.trim();
22 if (line.startsWith("//") && !multiComment ) {
23 line = reader.readLine();
24 continue;
25 } else if (line.startsWith("/*") || (multiComment)) {
26 multiComment=true;
27 if (line.endsWith("*/")) {
28 multiComment=false;
29 }
30 line = reader.readLine();
31 continue;

```

Рисунок 10 – Результат выполнения программы

Вывод: в ходе выполнения лабораторной работы были написаны программы согласно выданному заданию. Реализованы функции работы с файлами, текстовыми потоками и собственными обработчиками исключений.