



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

О Т Ч Е Т

по лабораторной работе № 5

Вариант № 5

Название лабораторной работы: Исключения. Файлы

Дисциплина: Языки программирования для работы с большими данными

Студент гр. ИУ6-22М

(Подпись, дата)

Р.Г. Гаделия

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2023

Введение

Целью лабораторной работы является приобретение навыков работы с исключениями и с файлами на языке программирования Java.

Практическая часть

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.

Определить класс Цепная дробь

$$A = a_0 + \frac{x}{a_1 + \frac{x}{a_2 + \frac{x}{a_3 + \dots}}}$$

Определить методы сложения, вычитания, умножения, деления.

Вычислить значение для заданного n, x, a[n].

Код написанной программы представлен в листинге 1.

Листинг 1 – Программа для первого задания

```
package com.java.lab;

import java.util.InputMismatchException;
import java.util.Scanner;

// Определить класс Цепная дробь.
// Определить методы сложения, вычитания, умножения, деления.
// Вычислить значение для заданного n, x, a[n].
// 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.

public class Main {
    public static void main(String[] args) {

        try {
            ContinuedFraction firstContinuedFraction =
            makeChainFraction();
            System.out.println("Total value: " +
            firstContinuedFraction.getValue());
        }
    }
}
```

```

        ContinuedFraction secondContinuedFraction =
makeChainFraction();
        System.out.println("Total value: " +
secondContinuedFraction.getValue());

        System.out.println("sum: " +
firstContinuedFraction.add(secondContinuedFraction));
        System.out.println("subtract: " +
firstContinuedFraction.subtract(secondContinuedFraction));
        System.out.println("multiply: " +
firstContinuedFraction.multiply(secondContinuedFraction));
        System.out.println("divide: " +
firstContinuedFraction.divide(secondContinuedFraction));
    } catch (InputMismatchException e) {
        System.out.println("Wrong input" +
e.getLocalizedMessage());
    } catch (OutOfMemoryError e) {
        System.out.println("Memory Error" +
e.getLocalizedMessage());
    } catch (Exception e) {
        System.out.println(e.getLocalizedMessage());
    }
}

public static ContinuedFraction makeChainFraction() {
    Scanner scanner = new Scanner(System.in);

    System.out.print("number of coefficients: ");
    var n = scanner.nextInt();

    System.out.print("continued fraction numerator: ");
    var x = scanner.nextInt();

    var a = new double[n];
    for (var i = 0; i < n; i++) {
        System.out.print("Enter coefficient No. " + (i) + " : ");
        a[i] = scanner.nextInt();
    }

    return new ContinuedFraction(x, n, a);
}

}

class ContinuedFraction {
    private final double[] a;
    private final int n;
    private final int x;

    ContinuedFraction(int x, int n, double[] a) {
        this.x = x;
        this.n = n;
        this.a = a;
    }

    double getValue() {
        var value = a[n-1];
        for (var i = n - 2; i >= 0; i--) {
            value = a[i] + x / value;
        }
    }
}

```

```

        }
        return value;
    }

    double add(ContinuedFraction continuedFraction) {
        try {
            return this.getValue() + continuedFraction.getValue();
        } catch (ArithmeticException e) {
            return 0;
        }
    }

    double subtract(ContinuedFraction continuedFraction) {
        try {
            return this.getValue() - continuedFraction.getValue();
        } catch (ArithmeticException e) {
            return 0;
        }
    }

    double multiply(ContinuedFraction continuedFraction) {
        try {
            return this.getValue() * continuedFraction.getValue();
        } catch (ArithmeticException e) {
            return 0;
        }
    }

    double divide(ContinuedFraction continuedFraction) {
        try {
            return this.getValue() / continuedFraction.getValue();
        } catch (ArithmeticException e) {
            return 0;
        }
    }
}

```

Результат выполнения программы показан на рисунке 1.

```
number of coefficients: 2
continued fraction numerator: 3
Enter coefficient No. 0 : 4
Enter coefficient No. 1 : 5
Total value: 4.6
number of coefficients: 6
continued fraction numerator: 7
Enter coefficient No. 0 : 8
Enter coefficient No. 1 : 7
Enter coefficient No. 2 : 6
Enter coefficient No. 3 : 5
Enter coefficient No. 4 : 4
Enter coefficient No. 5 : 3
Total value: 8.877248677248677
sum: 13.477248677248676
subtract: -4.277248677248677
multiply: 40.835343915343906
divide: 0.5181785671712957
```

Рисунок 1 – Результат выполнения программы

Задание 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.

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.

Код написанной программы представлен в листинге 2.

Листинг 2 – Программа для второго задания

```
package com.java.lab;

import java.util.InputMismatchException;
import java.util.Scanner;

class Fraction {
    int m, n;

    public Fraction() {
        this.m = 0;
        this.n = 1;
    }

    public Fraction(int m, int n) {
        this.m = m;
        this.n = n;
    }

    public Fraction add(Fraction addedFraction) {
        var sum = new Fraction();
        sum.n = this.n * addedFraction.n;
        sum.m = m * addedFraction.n + addedFraction.m * n;
        return sum;
    }

    public Fraction multiply(Fraction multipliedFraction) {
        return new Fraction(
            m * multipliedFraction.m,
            n * multipliedFraction.n
        );
    }

    public Fraction subtract(Fraction subtractedFraction) {
        Fraction invertedFraction = new Fraction(-
            subtractedFraction.m, subtractedFraction.n);
        return add(invertedFraction);
    }

    public Fraction divide(Fraction dividedFraction) {
        Fraction switchedFraction = new Fraction(dividedFraction.n,
            dividedFraction.m);
        try {
            return this.multiply(switchedFraction);
        } catch (ArithmeticException e) {
            return null;
        }
    }
}
```

```

        public String toString() {
            return m + "/" + n;
        }
    }

    // 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.
    // 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.
    public class Main {

        public static void main(String[] args) {
            System.out.println("k: ");
            Fraction[] fractions;
            try {
                Scanner console = new Scanner(System.in);
                var k = console.nextInt();

                fractions = new Fraction[k];

                for (var i = 0; i < k; i++) {
                    System.out.println("input m for Fraction No. " + (i +
1) + ":");
                    var m = console.nextInt();
                    System.out.println("input n for Fraction No. " + (i +
1) + ":");
                    var n = console.nextInt();

                    fractions[i] = new Fraction(m, n);
                }
            } catch (InputMismatchException e) {
                System.out.println("Wrong input" +
e.getLocalizedMessage());
                return;
            } catch (OutOfMemoryError e) {
                System.out.println("Memory Error" +
e.getLocalizedMessage());
                return;
            } catch (Exception e) {
                System.out.println(e.getLocalizedMessage());
                return;
            }

            try {
                print(fractions);
                System.out.println("sum:");

                System.out.println((fractions[0].add(fractions[1]).toString()));
                System.out.println("subtract:");
            }
        }
    }

```



```

System.out.println((fractions[0].subtract(fractions[1]).toString()));
    System.out.println("multiply:");

System.out.println((fractions[0].multiply(fractions[1]).toString()));
    System.out.println("divide:");

System.out.println((fractions[0].divide(fractions[1]).toString()));
    changeFraction(fractions);
    print(fractions);
} catch (Exception e) {
    System.out.println(e.getLocalizedMessage());
}
}

private static void changeFraction(Fraction[] fractions) {
    for (int i = 0; i < fractions.length - 1; i += 2) {
        fractions[i] = fractions[i].add(fractions[i + 1]);
    }
}

private static void print(Fraction[] array) {
    System.out.println("---- ");
    for (var i = 0; i < array.length; i++) {
        System.out.print(array[i].toString());
        if (i != array.length - 1) {
            System.out.print(" ");
        }
    }
    System.out.println("\n---- ");
}
}

```

Результат выполнения программы показан на рисунке 2.

```

k:
4
input m for Fraction No. 1:
1
input n for Fraction No. 1:
2
input m for Fraction No. 2:
3
input n for Fraction No. 2:
4
input m for Fraction No. 3:
5
input n for Fraction No. 3:
6
input m for Fraction No. 4:
7
input n for Fraction No. 4:
8
-----
1/2 3/4 5/6 7/8
-----
sum:
10/8
subtract:
-2/8
multiply:
3/8
divide:
4/6
-----
10/8 3/4 82/48 7/8
-----

```

Рисунок 2 – Результат выполнения программы

Задание 3

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

House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации. Создать массив объектов. Вывести: а) список

квартир, имеющих заданное число комнат; б) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке; с) список квартир, имеющих площадь, превосходящую заданную.

Код написанной программы представлен в листинге 3.

Листинг 3 – Программа для третьего задания

```
package com.java.lab;

import java.util.InputMismatchException;
import java.util.Scanner;

// House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица,
// Тип здания, Срок эксплуатации. Создать массив объектов. Вывести: а)
// список квартир, имеющих заданное число комнат; б) список квартир,
// имеющих заданное число комнат и расположенных на этаже, который
// находится в заданном промежутке; с) список квартир, имеющих площадь,
// превосходящую заданную.
// Выполнить задания из варианта 2 лабораторной работы 3, реализуя
// собственные обработчики исключений и исключения ввода/вывода.
public class Main {

    public static void main(String[] args) {
        var scanner = new Scanner(System.in);
        Flat[] arr;
        int n;
        try {
            System.out.print("n: ");
            n = scanner.nextInt();
            arr = new Flat[n];
            for(var i = 0; i < n; i++) {
                arr[i] = new Flat();
                System.out.print("input id for " + (i+1) + " flat: ");
                arr[i].setId(scanner.nextInt());
                System.out.print("input flatNum for " + (i+1) + "
flat: ");
                arr[i].setFlatNum(scanner.nextInt());
                System.out.print("input S for " + (i+1) + " flat: ");
                arr[i].setS(scanner.nextDouble());
                System.out.print("input flour for " + (i+1) + " flat:
");
                arr[i].setFlour(scanner.nextInt());
                System.out.print("input roomNum for " + (i+1) + "
flat: ");
                arr[i].setRoomNum(scanner.nextInt());
                System.out.print("input street for " + (i+1) + " flat:
");
                arr[i].setStreet(scanner.next());
                System.out.print("input type for " + (i+1) + " flat:
");
                arr[i].setType(scanner.next());
                System.out.print("input duration for " + (i+1) + "
flat: ");
```

```

        arr[i].setDuration(scanner.nextInt());
    }
    } catch (InputMismatchException e) {
        System.out.println("Wrong input " +
e.getLocalizedMessage());
        return;
    } catch (OutOfMemoryError e) {
        System.out.println("Memory Error " +
e.getLocalizedMessage());
        return;
    } catch (InvalidFlatException e) {
        System.out.println("InvalidFlatException " +
e.getLocalizedMessage());
        return;
    } catch (Exception e) {
        System.out.println(e.getLocalizedMessage());
        return;
    }
}

try {
    System.out.print("count roomNum: ");
    var countRoomNum = scanner.nextInt();
    for (var i = 0; i < n; i++) {
        if (arr[i].getRoomNum() == countRoomNum) {
            System.out.println();
            arr[i].printFlat();
            System.out.println();
        }
    }
} catch (InputMismatchException e) {
    System.out.println("Wrong input" +
e.getLocalizedMessage());
    return;
}

try {
    System.out.println();
    System.out.print("count roomNum and range flour");
    var countRoomNum = scanner.nextInt();
    var rangeMin = scanner.nextInt();
    var rangeMax = scanner.nextInt();
    for (var i = 0; i < n; i++) {
        if (arr[i].getRoomNum() == countRoomNum) {
            if (arr[i].getFlour() >= rangeMin &&
arr[i].getFlour() <= rangeMax) {
                System.out.println();
                arr[i].printFlat();
                System.out.println();
            }
        }
    }
} catch (InputMismatchException e) {
    System.out.println("Wrong input" +
e.getLocalizedMessage());
    return;
}

try {
    System.out.print("S: ");

```

```

        var S = scanner.nextDouble();
        for (int i = 0; i < n; i++) {
            if (arr[i].getS() >= S) {
                System.out.println();
                arr[i].printFlat();
                System.out.println();
            }
        }
    } catch (InputMismatchException e) {
        System.out.println("Wrong input" +
e.getLocalizedMessage());
    }
}

}

class Flat {
    private int id;
    private int flatNum;
    private double S;
    private int flour;
    private int roomNum;
    private String street;
    private String type;
    private int duration;

    public Flat() {
        id = 0;
        flatNum = 0;
        S = 0;
        flour = 0;
        roomNum = 0;
        street = "";
        type = "";
        duration = 0;
    }

    public Flat(int id, int flatNum, double S, int flour, int roomNum,
String street, String type, int duration) {
        this.id = id;
        this.flatNum = flatNum;
        this.S = S;
        this.flour = flour;
        this.roomNum = roomNum;
        this.street = street;
        this.type = type;
        this.duration = duration;
    }

    public void printFlat() {
        System.out.println("id: " + id);
        System.out.println("flatNum: " + flatNum);
        System.out.println("S: " + S);
        System.out.println("flour: " + flour);
        System.out.println("roomNum: " + roomNum);
        System.out.println("street: " + street);
        System.out.println("type: " + type);
        System.out.println("duration: " + duration);
    }
}

```

```

    }

    public double getS() {
        return S;
    }

    public void setS(double s) throws Exception {
        if (s <= 0) {
            throw new InvalidFlatException("S mast be > 0");
        }
        S = s;
    }

    public int getFlour() {
        return flour;
    }

    public void setFlour(int flour) {
        this.flour = flour;
    }

    public int getRoomNum() {
        return roomNum;
    }

    public void setRoomNum(int roomNum) {
        this.roomNum = roomNum;
    }

    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public int getDuration() {
        return duration;
    }

    public void setDuration(int duration) {
        this.duration = duration;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) throws Exception {

```

```

        if (id <= 0) {
            throw new InvalidFlatException("ID mast be > 0");
        }

        this.id = id;
    }

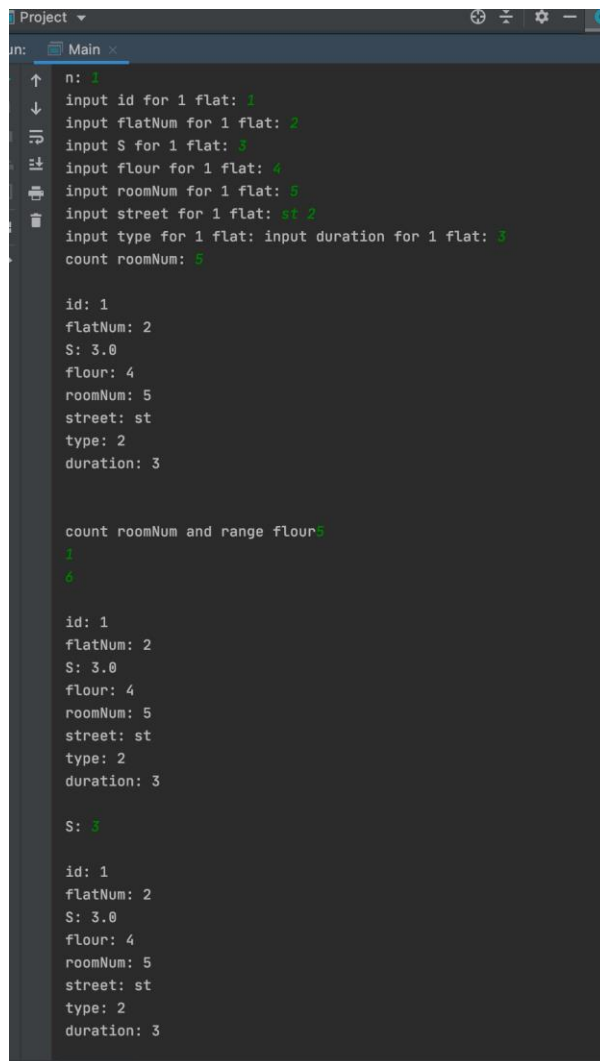
    public int getFlatNum() {
        return flatNum;
    }

    public void setFlatNum(int flatNum) throws Exception {
        if (flatNum <= 0) {
            throw new InvalidFlatException("flatNum mast be > 0");
        }
        this.flatNum = flatNum;
    }
}

class InvalidFlatException extends Exception {
    public InvalidFlatException (String str) {
        super(str);
    }
}

```

Результат выполнения программы показан на рисунке 3.



```
Project
in: Main x
n: 1
input id for 1 flat: 1
input flatNum for 1 flat: 2
input S for 1 flat: 3
input flour for 1 flat: 4
input roomNum for 1 flat: 5
input street for 1 flat: st 2
input type for 1 flat: input duration for 1 flat: 3
count roomNum: 5

id: 1
flatNum: 2
S: 3.0
flour: 4
roomNum: 5
street: st
type: 2
duration: 3

count roomNum and range flour5
1
6

id: 1
flatNum: 2
S: 3.0
flour: 4
roomNum: 5
street: st
type: 2
duration: 3

S: 3

id: 1
flatNum: 2
S: 3.0
flour: 4
roomNum: 5
street: st
type: 2
duration: 3
```

Рисунок 3 – Результат выполнения программы

Задание 4

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

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; с) сведения об абонентах в алфавитном порядке.

Код написанной программы представлен в листинге 4.

Листинг 4 – Программа для четвертого задания

```
package com.java.lab;
```



```

import java.util.Arrays;
import java.util.Comparator;
import java.util.InputMismatchException;
import java.util.Scanner;

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

    public static void main(String[] args) {
        var scanner = new Scanner(System.in);
        System.out.print("n: ");
        int n ;
        Phone[] arr;
        try {
            n = scanner.nextInt();
            arr = new Phone[n];
            for(var i = 0; i < n; i++) {
                arr[i] = new Phone();
                System.out.print("input id for " + (i+1) + " Phone:
");
                arr[i].setId(scanner.nextInt());
                System.out.print("input surname for " + (i+1) + "
Phone: ");
                arr[i].setSurname(scanner.next());
                System.out.print("input name for " + (i+1) + " Phone:
");
                arr[i].setName(scanner.next());
                System.out.print("input lastname for " + (i+1) + "
Phone: ");
                arr[i].setLastname(scanner.next());
                System.out.print("input address for " + (i+1) + "
Phone: ");
                arr[i].setLastname(scanner.next());
                System.out.print("input cardNumber for " + (i+1) + "
Phone: ");
                arr[i].setCardNumber(scanner.nextLong());
                System.out.print("input debit for " + (i+1) + " Phone:
");
                arr[i].setDebit(scanner.nextInt());
                System.out.print("input credit for " + (i+1) + "
Phone: ");
                arr[i].setCredit(scanner.nextInt());
                System.out.print("input cityTime for " + (i+1) + "
Phone: ");
                arr[i].setCityTime(scanner.nextInt());
                System.out.print("input outsideCityTime for " + (i+1)
+ " Phone: ");
                arr[i].setOutsideCityTime(scanner.nextInt());
            }
        } catch (InputMismatchException e) {

```

```

        System.out.println("Wrong input " +
e.getLocalizedMessage());
        return;
    } catch (OutOfMemoryError e) {
        System.out.println("Memory Error " +
e.getLocalizedMessage());
        return;
    } catch (InvalidPhoneException e) {
        System.out.println("InvalidPhoneException " +
e.getLocalizedMessage());
        return;
    } catch (Exception e) {
        System.out.println(e.getLocalizedMessage());
        return;
    }
    System.out.print("direction cityTime >: ");
    var direction = scanner.nextInt();
    for (var i = 0; i < n; i++) {
        if (arr[i].getCityTime() > direction) {
            System.out.println();
            arr[i].printPhone();
            System.out.println();
        }
    }

    System.out.println();
    System.out.print("use outsideCityTime");
    for (var i = 0; i < n; i++) {
        if (arr[i].getOutsideCityTime() > 0) {
            System.out.println();
            arr[i].printPhone();
            System.out.println();
        }
    }

    System.out.print("sorted phones: ");
    Arrays.stream(arr).sorted(new
PhoneComparator()).forEach(Phone::printPhone);
}
}

class PhoneComparator implements Comparator<Phone> {

    public int compare(Phone a, Phone b){

        return
a.getSurname().toUpperCase().compareTo(b.getSurname().toUpperCase());
    }
}

class Phone {
    private int id;
    private String surname;
    private String name;
    private String lastname;
    private String address;
    private long cardNumber;

```

```

private int debit;
private int credit;
private int cityTime;
private int outsideCityTime;

    public Phone(int id, String surname, String name, String lastname,
String address, long cardNumber, int debit, int credit, int cityTime,
int outsideCityTime) {
    this.id = id;
    this.surname = surname;
    this.name = name;
    this.lastname = lastname;
    this.address = address;
    this.cardNumber = cardNumber;
    this.debit = debit;
    this.credit = credit;
    this.cityTime = cityTime;
    this.outsideCityTime = outsideCityTime;
}

public Phone() {
    this.id = 0;
    this.surname = "surname";
    this.name = "name";
    this.lastname = "lastname";
    this.address = "address";
    this.cardNumber = 0;
    this.debit = 0;
    this.credit = 0;
    this.cityTime = 0;
    this.outsideCityTime = 0;
}

public int getId() {
    return id;
}

public void setId(int id) throws InvalidPhoneException {
    if (id <= 0 ) {
        throw new InvalidPhoneException("id must be >0");
    }
    this.id = id;
}

public String getSurname() {
    return surname;
}

public void setSurname(String surname) throws
InvalidPhoneException {
    if (surname.length() <= 0 ) {
        throw new InvalidPhoneException("surname must be >0");
    }
    this.surname = surname;
}

public String getName() {

```

```

        return name;
    }

    public void setName(String name) throws InvalidPhoneException {
        if (name.length() <= 0 ) {
            throw new InvalidPhoneException("name must be >0");
        }
        this.name = name;
    }

    public String getLastName() {
        return lastname;
    }

    public void setLastName(String lastname) {
        this.lastname = lastname;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public long getCardNumber() {
        return cardNumber;
    }

    public void setCardNumber(long cardNumber) {
        this.cardNumber = cardNumber;
    }

    public int getDebit() {
        return debit;
    }

    public void setDebit(int debit) {
        this.debit = debit;
    }

    public int getCredit() {
        return credit;
    }

    public void setCredit(int credit) {
        this.credit = credit;
    }

    public int getCityTime() {
        return cityTime;
    }

    public void setCityTime(int cityTime) {
        this.cityTime = cityTime;
    }
}

```

```

    public int getOutsideCityTime() {
        return outsideCityTime;
    }

    public void setOutsideCityTime(int outsideCityTime) {
        this.outsideCityTime = outsideCityTime;
    }

    public void printPhone() {
        System.out.println("id: " + id);
        System.out.println("surname: " + surname);
        System.out.println("name: " + name);
        System.out.println("lastname: " + lastname);
        System.out.println("address: " + address);
        System.out.println("cardNumber: " + cardNumber);
        System.out.println("debit: " + debit);
        System.out.println("credit: " + credit);
        System.out.println("cityTime: " + cityTime);
        System.out.println("outsideCityTime: " + outsideCityTime);
    }
}

class InvalidPhoneException extends Exception {
    public InvalidPhoneException (String str) {
        super(str);
    }
}

```

Результат выполнения программы показан на рисунке 4.

```
main
n: 1
input id for 1 Phone: 1
input surname for 1 Phone: Gorshkov
input name for 1 Phone: Ivan
input lastname for 1 Phone: Aleksandrovich
input address for 1 Phone: st 1
input cardNumber for 1 Phone: input debit for 1 Phone: 123
input credit for 1 Phone: 213
input cityTime for 1 Phone: 23
input outsideCityTime for 1 Phone: 34
direction cityTime >: 1

id: 1
surname: Gorshkov
name: Ivan
lastname: st
address: address
cardNumber: 1
debit: 123
credit: 213
cityTime: 23
outsideCityTime: 34

use outsideCityTime
id: 1
surname: Gorshkov
name: Ivan
lastname: st
address: address
cardNumber: 1
debit: 123
credit: 213
cityTime: 23
outsideCityTime: 34

sorted phones: id: 1
surname: Gorshkov
name: Ivan
lastname: st
address: address
cardNumber: 1
debit: 123
credit: 213
```

Рисунок 4 – Результат выполнения программы

Задание 5

В каждой строке стихотворения Анны Ахматовой подсчитать частоту повторяемости каждого слова из заданного списка и вывести эти слова в порядке возрастания частоты повторяемости.

Код написанной программы представлен в листинге 5.

Листинг 5 – Программа для пятого задания

```
package com.java.lab;
```

```

import java.io.File;
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.nio.file.StandardOpenOption;
import java.util.*;
import java.util.stream.Collectors;

// 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 каждой строке стихотворения Анны Ахматовой подсчитать частоту
повторяемости каждого слова из заданного списка и вывести эти слова в
порядке возрастания частоты повторяемости.
public class Main {

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

        File file_in = new
File("/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-
12M/Lab5/Z5/in.txt");
        Path path =
Paths.get("/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-
12M/Lab5/Z5/out.txt");
        Scanner in = new Scanner(file_in);
        int k = 1;
        while(in.hasNextLine()) {
            var wordsKeys = new String[] {"ним", "спокойно", "я"};
            String line = in.nextLine();
            String[] words = line.split(" ");
            var lineCountWords = filling(words);
            Map<String, Integer> linkedHashMap = new
LinkedHashMap<>();

            for (Map.Entry<String, Integer> employee :
lineCountWords.entrySet()) {
                if (Arrays.stream(wordsKeys).anyMatch(v ->
v.toLowerCase().contains(employee.getKey().toLowerCase())) &&
employee.getKey().length() != 0){
                    linkedHashMap.put(employee.getKey(),
employee.getValue());
                }
            }
            Map<String, Integer> sortedMap = linkedHashMap.entrySet().
stream().

sorted(Map.Entry.comparingByValue(Comparator.reverseOrder())).
collect(Collectors.toMap(Map.Entry::getKey,
Map.Entry::getValue, (e1, e2) -> e1, LinkedHashMap::new));

            if (k == 1) {

```

```

        Files.write(path,
Collections.singleton(sortedMap.toString()), StandardCharsets.UTF_8);
        } else {
            Files.write(path,
Collections.singleton(sortedMap.toString()), StandardCharsets.UTF_8,
StandardOpenOption.APPEND);
        }
        k += 1;
    }
}

public static Map<String, Integer> filling(String[] words) {
    Map<String, Integer> m = new HashMap<>();
    for (String word : words) {
        m.put(word, m.get(word) == null ? 1 : m.get(word) + 1);
    }
    return m;
}
}

```

Результат выполнения программы показан на рисунке 5.

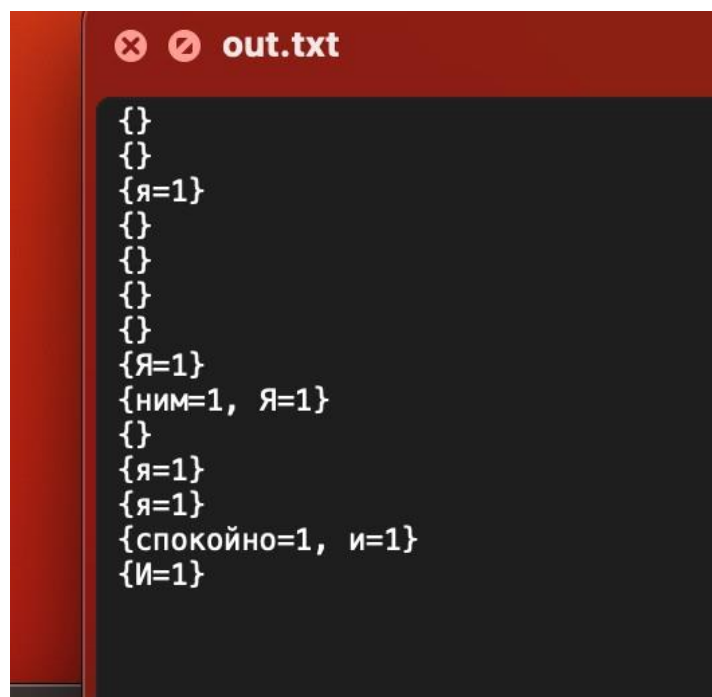


Рисунок 5 – Результат выполнения программы

Задание 6

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

Код написанной программы представлен в листинге 6.

Листинг 6 – Программа для шестого задания

```
package com.java.lab;
```



```

import java.io.File;
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.nio.file.StandardOpenOption;
import java.util.*;

// 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.

public class Main {

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

        File file_in = new
File("/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-
12M/Lab5/Z6/in.txt");
        Path path =
Paths.get("/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-
12M/Lab5/Z6/out.txt");
        Scanner in = new Scanner(file_in);
        int k = 1;
        while(in.hasNextLine()) {
            String line = in.nextLine();
            String[] words = line.split(" ");
            words[0] = capitalize(words[0]);
            if (k == 1) {
                Files.write(path, Collections.singleton(String.join("
", words)), StandardCharsets.UTF_8);
            } else {
                Files.write(path, Collections.singleton(String.join("
", words)), StandardCharsets.UTF_8, StandardOpenOption.APPEND);
            }
            k += 1;
        }
        public static String capitalize(String str)
        {
            if (str == null || str.length() == 0) return str;
            return str.substring(0, 1).toUpperCase() + str.substring(1);
        }
    }
}

```

Результат выполнения программы показан на рисунке 6.

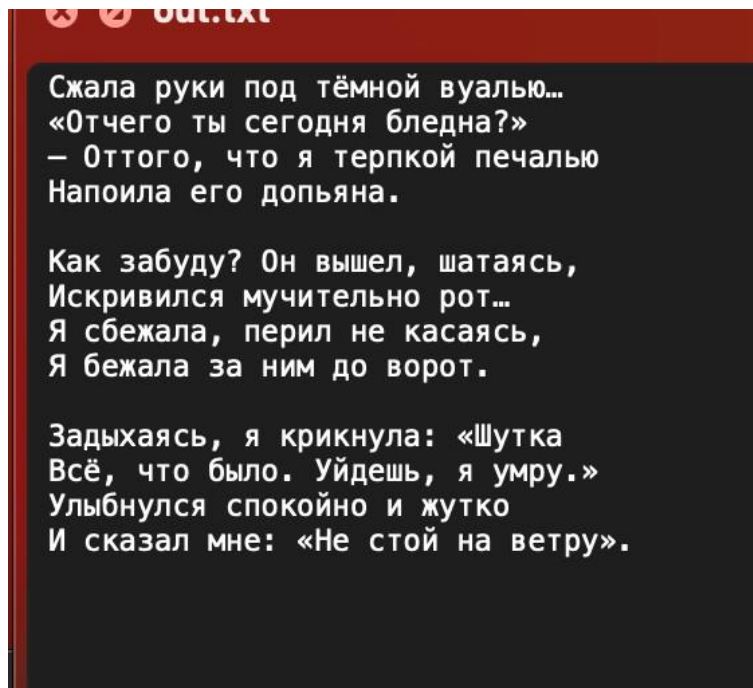


Рисунок 6 – Результат выполнения программы

Задание 7

При выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File. Из файла удалить все слова, содержащие от трех до пяти символов, но при этом из каждой строки должно быть удалено только максимальное четное количество таких слов.

Код написанной программы представлен в листинге 7.

Листинг 7 – Программа для седьмого задания

```
package com.java.lab;
import java.io.File;
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.nio.file.StandardOpenOption;
import java.util.*;

// При выполнении следующих заданий для вывода результатов создавать
// новую директорию и файл средствами класса File
// 6. Из файла удалить все слова, содержащие от трех до пяти символов,
// но при этом из каждой строки должно быть удалено только максимальное
// четное количество таких слов.
public class Main {

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

```

        File file_in = new
File("/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-
12M/Lab5/Z7/in.txt");
        Path path =
Paths.get("/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-
12M/Lab5/Z7/directory/out.txt");
        File theDir = new
File("/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-
12M/Lab5/Z7/directory");
        if (!theDir.exists()){
            theDir.mkdir();
        }

        Scanner in = new Scanner(file_in);
        int k = 1;

        while(in.hasNextLine()) {
            String line = in.nextLine();
            String[] words = line.split(" ");
            var list = new ArrayList<String>();
            var buf = new StringBuilder(line);

            for (String s : words) {
                if ((s.length() >= 3) && (s.length() <= 5)) {
                    list.add(s);
                }
            }

            if ((list.size() % 2) == 1) {
                list.remove(list.size() - 1);
            }

            int index = 0;
            for (String s : list) {
                index = buf.indexOf(s, index);
                buf.delete(index, index + s.length());
            }
            if (k == 1) {
                Files.write(path,
Collections.singleton(buf.toString()), StandardCharsets.UTF_8);
            } else {
                Files.write(path,
Collections.singleton(buf.toString()), StandardCharsets.UTF_8,
StandardOpenOption.APPEND);
            }
            k += 1;
        }
    }
}

```

Результат выполнения программы показан на рисунке 7.

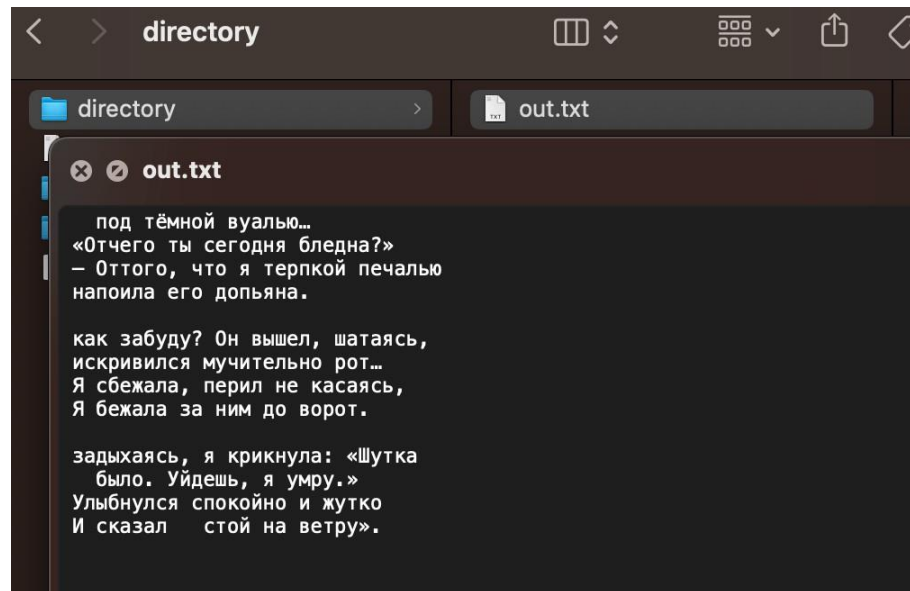


Рисунок 7 – Результат выполнения программы

Задание 8

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.

Код написанной программы представлен в листинге 8.

Листинг 8 – Программа для четвертого задания

```
package com.java.lab;
import java.io.File;
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.nio.file.StandardOpenOption;
import java.util.*;
// При выполнении следующих заданий для вывода результатов создавать
// новую директорию и файл средствами класса File
// 7.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 {

        File file_in = new
File("/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-
12M/Lab5/Z8/Main.java");
        Path path =
Paths.get("/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-
12M/Lab5/Z8/directory/out.java");
```

```

        File theDir = new
File("/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-
12M/Lab5/Z8/directory");
        if (!theDir.exists()){
            theDir.mkdir();
        }

        Scanner in = new Scanner(file_in);
        int k = 1;

        while(in.hasNextLine()) {
            String line = in.nextLine();
            StringBuilder stringBuilder = new StringBuilder();

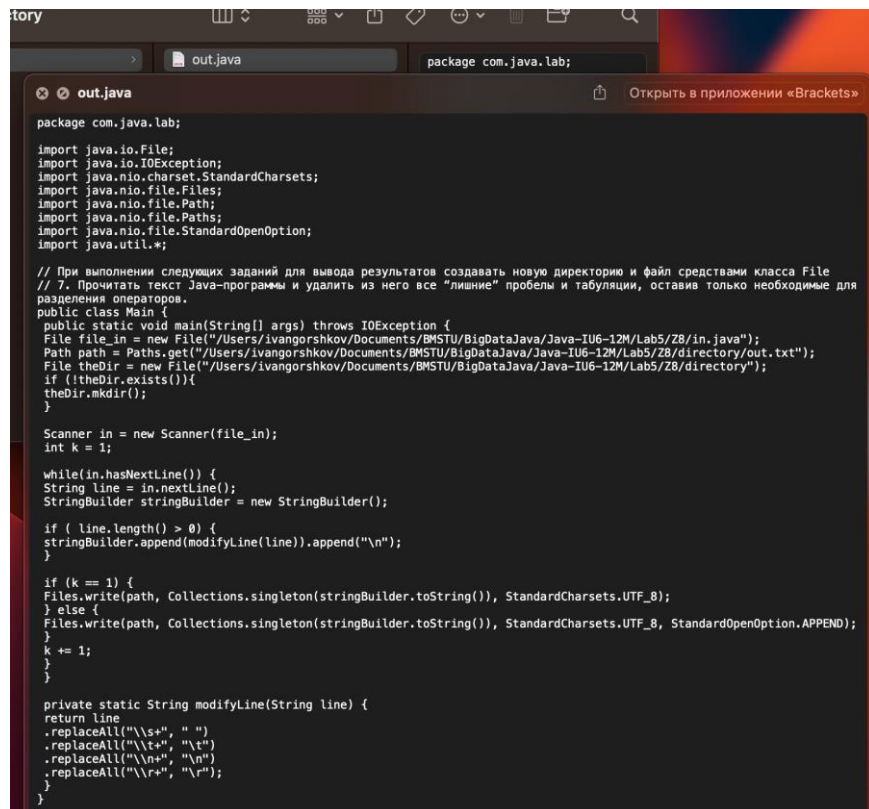
            if (line.length() > 0) {
                stringBuilder.append(modifyLine(line));
            }

            if (k == 1) {
                Files.write(path,
Collections.singleton(stringBuilder.toString()),
StandardCharsets.UTF_8);
            } else {
                Files.write(path,
Collections.singleton(stringBuilder.toString()),
StandardCharsets.UTF_8, StandardOpenOption.APPEND);
            }
            k += 1;
        }
    }

    private static String modifyLine(String line) {
        return line
            .replaceAll("\\s+", " ")
            .replaceAll("\\t+", "\t")
            .replaceAll("\\n+", "\n")
            .replaceAll("\\r+", "\r");
    }
}

```

Результат выполнения программы показан на рисунке 8.



```
package com.java.lab;

import java.io.File;
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.nio.file.StandardOpenOption;
import java.util.*;

// При выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File
// 7. 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 {
        File file_in = new File("C:/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-12M/Lab5/Z8/in.java");
        Path path = Paths.get("C:/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-12M/Lab5/Z8/directory/out.txt");
        File theDir = new File("C:/Users/ivangorshkov/Documents/BMSTU/BigDataJava/Java-IU6-12M/Lab5/Z8/directory");
        if (!theDir.exists()){
            theDir.mkdir();
        }

        Scanner in = new Scanner(file_in);
        int k = 1;

        while(in.hasNextLine()) {
            String line = in.nextLine();
            StringBuilder stringBuilder = new StringBuilder();

            if (line.length() > 0) {
                stringBuilder.append(modifyLine(line)).append("\n");
            }

            if (k == 1) {
                Files.write(path, Collections.singleton(stringBuilder.toString()), StandardCharsets.UTF_8);
            } else {
                Files.write(path, Collections.singleton(stringBuilder.toString()), StandardCharsets.UTF_8, StandardOpenOption.APPEND);
            }
            k += 1;
        }
    }

    private static String modifyLine(String line) {
        return line
            .replaceAll("\\s+", " ")
            .replaceAll("\\t+", "\t")
            .replaceAll("\\n+", "\n")
            .replaceAll("\\r+", "\r");
    }
}
```

Рисунок 8 – Результат выполнения программы

Вывод: В результате выполнения лабораторной работы были приобретены навыки работы с исключениями и файлами на языке программирования Java.