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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ 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

Название: Исключения. Файлы

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-22М (Группа) _____ (Подпись, дата) А.М. Панфилкин (И.О. Фамилия)

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

Москва, 2022 г.

Весь приведенный ниже код также доступен в следующем репозитории:

https://github.com/SilkSlime/iu6plfbd

Задание 1: Определить класс Матрица размерности (m x n). Класс должен содержать несколько конструкторов. Объявить массив объектов. Передать объекты в метод, меняющий местами строки с максимальным и минимальным элементами k-го столбца. Создать метод, который изменяет i-ю матрицу путем возведения ее в квадрат. Выполнить задание 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.

Листинг 1 – Задание 1

```
package l5;

public class e1 {

    /**
     * Из 3й лабы. Вариант 1. Задача 5.
     * Определить класс Матрица размерности (m x n). Класс должен содержать
     * несколько конструкторов. Объявить массив объектов. Передать объекты
     * в метод, меняющий местами строки с максимальным и минимальным элементами
     * k-го столбца. Создать метод, который изменяет i-ю матрицу путем
     * возведения ее в квадрат.
     *
     * Выполнить задание 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 static void main(String[] args) {
        // Define an array of matrixes and fill it using constructors
        Matrix[] matrixes = new Matrix[3];
        matrixes[0] = new Matrix(3, 3);
        matrixes[1] = new Matrix(new int[][] {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}});
        matrixes[2] = new Matrix(new int[][] {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}});

        matrixes[1].print();
        System.out.println();

        // Change rows with max and min elements of k-th column
        matrixes[1].changeRowsMinMaxOfK(1);
        matrixes[1].print();
        System.out.println();
```

```
        // Change i-th matrix by squaring it
        matrixes[1].square(1, matrixes);
        matrixes[1].print();
        System.out.println();

    }
}

// Define class Matrix with size (m x n) and two constructors
class Matrix {
    private int m;
    private int n;
    private int[][] matrix;

    public Matrix(int m, int n) {
        // Выбрасываем исклчение, если размерность матрицы меньше 1
        if (m  1 || n  1) {
            throw new IllegalArgumentException("Matrix size must be greater than 1");
        }

        this.m = m;
        this.n = n;
        matrix = new int[m][n];
    }

    public Matrix(int[][] matrix) {
        this.matrix = matrix;
        this.m = matrix.length;
        this.n = matrix[0].length;
    }

    // Method for printing matrix using System.out.println()
    public void print() {
        for (int i = 0; i  this.n; i++) {
            for (int j = 0; j  this.m; j++) {
                System.out.printf("%4d", this.matrix[i][j]);
            }
            System.out.println();
        }
    }

    // Method for changing rows with max and min elements of k-th column
    public void changeRowsMinMaxOfK(int k) {
        // Выбрасываем исклчение, если k больше размерности матрицы или меньше 0
        if (k > this.n || k  0) {
            throw new IllegalArgumentException("k must be less than matrix size and greater
than 0");
        }
        int max = matrix[0][k];
        int min = matrix[0][k];
        int maxRow = 0;
        int minRow = 0;
```

```
        for (int i = 0; i  m; i++) {
            if (matrix[i][k] > max) {
                max = matrix[i][k];
                maxRow = i;
            }
            if (matrix[i][k]  min) {
                min = matrix[i][k];
                minRow = i;
            }
        }

        int[] temp = matrix[maxRow];
        matrix[maxRow] = matrix[minRow];
        matrix[minRow] = temp;
    }

    // Method for changing i-th matrix by squaring it
    public void square(int i, Matrix[] matrixes) {
        // Выбрасываем исклчение, если i больше размерности массива или меньше 0
        if (i > matrixes.length || i  0) {
            throw new IllegalArgumentException("i must be less than matrixes array size and
greater than 0");
        }
        int[][] temp = matrixes[i].matrix;
        for (int j = 0; j  m; j++) {
            for (int k = 0; k  n; k++) {
                temp[j][k] *= temp[j][k];
            }
        }
        matrixes[i].matrix = temp;
    }

}
```

Задание 2: Определить класс Цепная дробь. Определить методы сложения, вычитания, умножения, деления. Вычислить значение для заданного n, x, a[n]. Выполнить задание 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.

Листинг 2 – Задание 2

```
package l5;

public class e2 {
```

```
    /**
     * Из 3й лабы. Вариант 1. Задача 6.
     * Определить класс Цепная дробь. Определить методы сложения, вычитания,
     * умножения, деления. Вычислить значение для заданного n, x, a[n].
     *
     * Выполнить задание 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 static void main(String[] args) {
        ChainFraction cf1 = new ChainFraction(3, 2, new int[] {1, 2, 3});
        cf1.print();
        System.out.println(cf1.value());

        ChainFraction cf2 = new ChainFraction(1.75, 3, 2);
        cf2.print();
        System.out.println(cf2.value());

        cf1.add(cf2);
        cf1.print();
        System.out.println(cf1.value());


    }
}

class ChainFraction {
    private int n;
    private int x;
    private int[] a;

    public ChainFraction(int n, int x, int[] a) {
        // Выбрасываем исключение, если длина массива не равна n
        if (a.length != n) {
            throw new IllegalArgumentException("Length of array must be equal to n");
        }
        // Выбрасываем исключение, если x равен 0
        if (x == 0) {
            throw new IllegalArgumentException("x must not be equal to 0");
        }
        // Выбрасываем исключение, если n равен 0
        if (n == 0) {
            throw new IllegalArgumentException("n must not be equal to 0");
        }
        // Выбрасываем исключение, если n меньше 0
        if (n  0) {
            throw new IllegalArgumentException("n must not be less than 0");
        }
        this.n = n;
```

```
        this.x = x;
        this.a = a;
    }

    public ChainFraction(double value, int n, int x) {
        // Выбрасываем исключение, если x равен 0
        if (x == 0) {
            throw new IllegalArgumentException("x must not be equal to 0");
        }
        // Выбрасываем исключение, если n равен 0
        if (n == 0) {
            throw new IllegalArgumentException("n must not be equal to 0");
        }
        // Выбрасываем исключение, если n меньше 0
        if (n  0) {
            throw new IllegalArgumentException("n must not be less than 0");
        }
        this.n = n;
        this.x = x;
        this.a = calculateA(value, n, x);
    }

    private int[] calculateA(double value, int n, int x) {
        // Выбрасываем исключение, если x равен 0
        if (x == 0) {
            throw new IllegalArgumentException("x must not be equal to 0");
        }
        // Выбрасываем исключение, если n равен 0
        if (n == 0) {
            throw new IllegalArgumentException("n must not be equal to 0");
        }
        // Выбрасываем исключение, если n меньше 0
        if (n  0) {
            throw new IllegalArgumentException("n must not be less than 0");
        }
        this.a = new int[n];
        a[0] = (int) value;
        for (int i = 1; i  n; i++) {
            value = x / (value - a[i-1]);
            a[i] = (int) value;
        }
        return a;
    }

    public ChainFraction add(ChainFraction cf) {
        this.a = calculateA(this.value() + cf.value(), this.n, this.x);
        return null;
    }

    public ChainFraction subtract(ChainFraction cf) {
        this.a = calculateA(this.value() - cf.value(), this.n, this.x);
        return null;
    }
```

```
    public ChainFraction multiply(ChainFraction cf) {
        this.a = calculateA(this.value() * cf.value(), this.n, this.x);
        return null;
    }

    public ChainFraction divide(ChainFraction cf) {
        this.a = calculateA(this.value() / cf.value(), this.n, this.x);
        return null;
    }

    public double value() {
        double res = a[n-1];
        for (int i = n-2; i >= 0; i--) {
            res = a[i] + x / res;
        }
        return res;
    }

    public void print() {
        System.out.printf("n = %d, x = %d, a = ", this.n, this.x);
        for (int i = 0; i  this.n; i++) {
            System.out.printf("%d ", this.a[i]);
        }
        System.out.println();
    }
}
```

Задание 3: Создать класс Bookk: id, Название, Автор(ы), Издательство, Год издания, Количество страниц, Цена, Переплет. Создать массив объектов. Вывести: a) список книг заданного автора; b) список книг, выпущенных заданным издательством; c) список книг, выпущенных после заданного года. 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. Выполнить задание, реализуя собственные обработчики исключений и исключения ввода/вывода

Листинг 3 – Задание 3

```
package l5;

public class e3 {

    /**
     * Из 3й лабы. Вариант 2. Задача 5.
     * Создать класс
     * Bookk: id, Название, Автор(ы), Издательство, Год издания, Количество страниц, Цена,
Переплет.
     * Создать массив объектов. Вывести:
     * a) список книг заданного автора;
```

```
     * b) список книг, выпущенных заданным издательством;
     * c) список книг, выпущенных после заданного года.
     * 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.
     *
     * Выполнить задание, реализуя собственные обработчики исключений и исключения ввода/вывода
     */

    public static void main(String[] args) {
        // Define an array of 5 books and fill it with different data (differe authors,
publishers, years, etc.)
        Bookk[] books = new Bookk[5];

        try {
            books[0] = new Bookk(1, "Bookk1", new String[] {"Author1", "Author2"},
"Publisher1", 2000, 100, 10.0, "Hardcover");
            books[1] = new Bookk(2, "Bookk2", new String[] {"Author1", "Author3"},
"Publisher2", 2001, 200, 20.0, "Hardcover");
            books[2] = new Bookk(3, "Bookk3", new String[] {"Author2", "Author3"},
"Publisher3", 2002, 300, 30.0, "Hardcover");
            books[3] = new Bookk(4, "Bookk4", new String[] {"Author1", "Author2"},
"Publisher4", 2003, 400, 40.0, "Hardcover");
            books[4] = new Bookk(5, "Bookk5", new String[] {"Author1", "Author3"},
"Publisher5", 2004, 500, 50.0, "Hardcover");


            books[4] = new Bookk(-3, "", new String[] {"Author1", "Author3"}, "Publisher5",
2004, 500, 50.0, "Hardcover");
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
            return;
        }

        // Print all books
        System.out.println("All books:");
        for (Bookk book : books) {
            System.out.println(book);
        }

        // Print books by author
        System.out.println("Bookks by Author1:");
        for (Bookk book : books) {
            for (String author : book.getAuthors()) {
                if (author.equals("Author1")) {
                    System.out.println(book);
                }
            }
        }

        // Print books by publisher
        System.out.println("Bookks by Publisher1:");
        for (Bookk book : books) {
```

```
            if (book.getPublisher().equals("Publisher1")) {
                System.out.println(book);
            }
        }

        // Print books after year
        System.out.println("Bookks after 2002:");
        for (Bookk book : books) {
            if (book.getYear() > 2002) {
                System.out.println(book);
            }
        }

    }
}

class Bookk {
    private int id;
    private String name;
    private String[] authors;
    private String publisher;
    private int year;
    private int pages;
    private double price;
    private String binding;

    public Bookk(int id, String name, String[] authors, String publisher, int year, int pages,
double price, String binding) {
        // Выкидываем исключение, если id меньше 0
        if (id  0) {
            throw new IllegalArgumentException("Id must be greater than 0");
        }
        // Выкидываем исключение, если name пустая строка
        if (name.equals("")) {
            throw new IllegalArgumentException("Name must not be empty");
        }

        this.id = id;
        this.name = name;
        this.authors = authors;
        this.publisher = publisher;
        this.year = year;
        this.pages = pages;
        this.price = price;
        this.binding = binding;
    }

    // method to create array of books with random data and random authors
    public static Bookk[] createRandomBookks(int count) {
        // Выкидываем исключение, если count меньше 1
        if (count  1) {
            throw new IllegalArgumentException("Count must be greater than 0");
        }
```

```
        Bookk[] books = new Bookk[count];
        for (int i = 0; i  count; i++) {
            books[i] = new Bookk(i, "Bookk" + i, new String[] {"Author" + (int) (Math.random()
* 10), "Author" + (int) (Math.random() * 10)}, "Publisher" + (int) (Math.random() * 10), (int)
(Math.random() * 100), (int) (Math.random() * 1000), Math.random() * 100, "Hardcover");
        }
        return books;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String[] getAuthors() {
        return authors;
    }

    public void setAuthors(String[] authors) {
        this.authors = authors;
    }

    public String getPublisher() {
        return publisher;
    }

    public void setPublisher(String publisher) {
        this.publisher = publisher;
    }

    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public int getPages() {
        return pages;
    }
```

```
    public void setPages(int pages) {
        this.pages = pages;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public String getBinding() {
        return binding;
    }

    public void setBinding(String binding) {
        this.binding = binding;
    }

    // Override toString() method to print the object
    @Override
    public String toString() {
        String authors = "";
        // join all authors in one string without last comma
        for (int i = 0; i  this.authors.length; i++) {
            if (i == this.authors.length - 1) {
                authors += this.authors[i];
            } else {
                authors += this.authors[i] + ", ";
            }
        }
        return name + " by " + authors + " published by " + publisher + " in " + year + " with
" + pages + " pages, " + price + " rub. and " + binding + " binding.";
    }

}
```

Задание 4: Создать класс House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации. Создать массив объектов. Вывести: a) список квартир, имеющих заданное число комнат; b) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке; c) список квартир, имеющих площадь, превосходящую заданную. 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. Выполнить задание, реализуя собственные обработчики исключений и исключения ввода/вывода

Листинг 4 – Задание 4

```
package l5;

public class e4 {

    /**
     * Из 3й лабы. Вариант 2. Задача 6.
     * Создать класс
     * House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок
эксплуатации.
     * Создать массив объектов. Вывести:
     * a) список квартир, имеющих заданное число комнат;
     * b) список квартир, имеющих заданное число комнат и расположенных на этаже, который
находится в заданном промежутке;
     * c) список квартир, имеющих площадь, превосходящую заданную.
     * 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.
     *
     * Выполнить задание, реализуя собственные обработчики исключений и исключения ввода/вывода
     */

    public static void main(String[] args) {
        // Create array of houses with random data
        House[] houses = null;

        try {
            houses = House.createHouses(0);
        } catch (IllegalArgumentException e) {
            System.out.println("Error: " + e.getMessage());
            return;
        }

        // Print all houses with 3 rooms
        System.out.println("All houses with 3 rooms:");
        for (House house : houses) {
            if (house.getRooms() == 3) {
                System.out.println(house);
            }
        }

        // Print all houses with 3 rooms and floor between 10 and 20
        System.out.println("All houses with 3 rooms and floor between 10 and 20:");
        for (House house : houses) {
            if (house.getRooms() == 3  house.getFloor() >= 10  house.getFloor() = 20) {
                System.out.println(house);
            }
        }

        // Print all houses with area more than 50
        System.out.println("All houses with area more than 50:");
        for (House house : houses) {
```

```
            if (house.getArea() > 90) {
                System.out.println(house);
            }
        }


    }
}

class House {
    private int id;
    private int apartmentNumber;
    private double area;
    private int floor;
    private int rooms;
    private String street;
    private String buildingType;
    private int exploitationTerm;

    public House(int id, int apartmentNumber, double area, int floor, int rooms, String street,
String buildingType, int exploitationTerm) {
        this.id = id;
        this.apartmentNumber = apartmentNumber;
        this.area = area;
        this.floor = floor;
        this.rooms = rooms;
        this.street = street;
        this.buildingType = buildingType;
        this.exploitationTerm = exploitationTerm;
    }

    // Create array of houses with random data
    public static House[] createHouses(int count) {
        // Выкидывает исключение, если count  0
        if (count = 0) {
            throw new IllegalArgumentException("Count must be more than 0");
        }
        House[] houses = new House[count];
        for (int i = 0; i  count; i++) {
            houses[i] = new House(i, (int) (Math.random() * 100), Math.random() * 100, (int)
(Math.random() * 100), (int) (Math.random() * 10), "Street " + (int) (Math.random() * 100),
"Type " + (int) (Math.random() * 10), (int) (Math.random() * 100));
        }
        return houses;
    }

    // Getters and setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
```

```
    }

    public int getApartmentNumber() {
        return apartmentNumber;
    }

    public void setApartmentNumber(int apartmentNumber) {
        this.apartmentNumber = apartmentNumber;
    }

    public double getArea() {
        return area;
    }

    public void setArea(double area) {
        this.area = area;
    }

    public int getFloor() {
        return floor;
    }

    public void setFloor(int floor) {
        this.floor = floor;
    }

    public int getRooms() {
        return rooms;
    }

    public void setRooms(int rooms) {
        this.rooms = rooms;
    }

    public String getStreet() {
        return street;
    }

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

    public String getBuildingType() {
        return buildingType;
    }

    public void setBuildingType(String buildingType) {
        this.buildingType = buildingType;
    }

    public int getExploitationTerm() {
        return exploitationTerm;
    }
```

```
    public void setExploitationTerm(int exploitationTerm) {
        this.exploitationTerm = exploitationTerm;
    }

    @Override
    public String toString() {
        return "House [id=" + id + ", apartmentNumber=" + apartmentNumber + ", area=" + area +
", floor=" + floor
                + ", rooms=" + rooms + ", street=" + street + ", buildingType=" + buildingType
+ ", exploitationTerm="
                + exploitationTerm + "]";
    }
}
```

Задание 5: 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 Найти в строке наибольшее число цифр, идущих подряд.

Листинг 5 – Задание 5

```
package l5;

import java.util.Scanner;

public class e5 {

    /**
     * Вариант 3. Задача 5.
     * 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
     *
     * Найти в строке наибольшее число цифр, идущих подряд.
     */

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Количество цифр подряд
        int count = 0;
        // Максимальное количество цифр подряд
        int maxCount = 0;

        // Читаем строки из консоли пока не встретим пустую строку
        String line;
        while (!(line = scanner.nextLine()).isEmpty()) {
```

```
            // Разбиваем строку на слова с разделителем - любое количество пробелов
            String[] words = line.split("\\s+");

            for (String word : words) {

                // Для каждого символа в слове
                for (int i = 0; i  word.length(); i++) {
                    // Если символ - цифра, то увеличиваем счетчик
                    if (Character.isDigit(word.charAt(i))) {
                        count++;
                    } else {
                        // Иначе обнуляем счетчик
                        count = 0;
                    }

                    // Если счетчик больше максимального, то обновляем максимальный
                    if (count > maxCount) {
                        maxCount = count;
                    }
                }
            }

            // Выводим максимальное значение
            System.out.println("Max digits len: " + maxCount);
            maxCount = 0;
            count = 0;
        }



    }
}
```

Задание 8: При выполнении 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.

Листинг 8 – Задание 8

```
package l5;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.File;
```

```
public class e8 {

    /**
     * Вариант 4. Задача 6.
     * При выполнении 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.
     */

    public static void main(String[] args) {
        // Входной и выходной файлы
        String inputFileName = "15/e8_in.txt";
        String outputFileName = "15/e8_out.txt";

        try {
            File inputFile = new File(inputFileName);
            File outputFile = new File(outputFileName);

            // Создаем директорию для выходного файла, если она не существует
            File outputDir = outputFile.getParentFile();
            if (!outputDir.exists()) {
                outputDir.mkdirs();
            }

            // Создаем объекты для чтения и записи (буферизованные потоки)
            BufferedReader reader = new BufferedReader(new FileReader(inputFile));
            BufferedWriter writer = new BufferedWriter(new FileWriter(outputFile));

            String line;
            // Читаем построчно
            while ((line = reader.readLine()) != null) {
                // Разбиваем строку на слова по пробелам и табуляциям (или любым другим
пробельным символом)
                String[] words = line.split("\\s+");

                // Считаем количество слов, которые стоит удалить
                int countToDelete = 0;
                for (String word : words) {
                    if (word.length() >= 3  word.length() = 5) {
                        countToDelete++;
                    }
                }
                // Удаляем только максимальное четное количество слов
                countToDelete = countToDelete % 2 == 0 ? countToDelete : countToDelete - 1;

                // Удаляем слова
                int deleted = 0;
                // Проходим по всем словам в строке и удаляем нужное количество слов
                for (int i = 0; i  words.length  deleted  countToDelete; i++) {
```

```
                    String word = words[i];
                    if (word.length() >= 3  word.length() = 5) {
                        // Маркируем слово для удаления (присваиваем null)
                        words[i] = null;
                        deleted++;
                    }
                }

                // Записываем строку в выходной файл
                for (String word : words) {
                    if (word != null) {
                        writer.write(word + " ");
                    }
                }
                writer.write("\n");
            }
            // Закрываем потоки
            reader.close();
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Вывод: В ходе лабораторной работы мы изучили и реализовали задания по работе с исключениями, файлами и классами в Java. Мы создали классы для матрицы, цепных дробей, книг и квартир, а также написали программы для обработки текстовых данных. Также мы практиковались в контроле состояния потоков ввода/вывода и создании новых директорий и файлов. В целом, выполнение лабораторной работы помогло нам укрепить свои навыки программирования на Java и применение ее возможностей в различных задачах.