



**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 09.04.01/12 Интеллектуальный анализ больших
данных в системах поддержки принятия решений.**

О Т Ч Е Т

по лабораторной работе № 5

Вариант № 5

Название: Внутренние классы и интерфейсы

Дисциплина: Языки программирования для работы с большими данными

Студент

ИУ6-23М

(Группа)

(Подпись, дата)

А.О.Крейденко

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2024

Цель: изучить работу с исключениями и файлами в java.

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.

Код класса Main:

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

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        Matrix[] matrices = new Matrix[2];
        try {
            for (int i = 0; i < 2; i++) {
                System.out.println("Введите количество строк:");
                int rows = scanner.nextInt();
                System.out.println("Введите количество столбцов");
                int columns = scanner.nextInt();
                scanner.nextLine();
                double[][] arr = new double[rows][columns];
                for (int j = 0; j < rows; j++) {
                    System.out.println("Введите ряд " + (j + 1));
                    String[] row = scanner.nextLine().split(" ");
                    for (int k = 0; k < row.length; k++) {
                        arr[j][k] = Double.parseDouble(row[k]);
                    }
                    matrices[i] = new Matrix(rows, columns, arr);
                }

                // задаем номер столбца для метода перестановки строк
                int k = 0;
                for (int i = 0; i < 2; ++i) {
                    matrices[i].swap_rows(k);
                }
            }
        } catch (InputMismatchException e) {
            System.out.println("Некорректный ввод");
        }
    }
}
```

```

        for (int i = 0; i < 2; ++i) {
            System.out.printf("Matrix № %d\n", i + 1);
            matrices[i].print();
            System.out.println();
        }

        // возведение матрицы в квадрат
        System.out.println("Возведения в квадрат матрицы
1:");

        matrices[0].make_squared();
        matrices[0].print();

        System.out.println("Возведение в квадрат матрицы
2:");

        matrices[1].make_squared();
        matrices[1].print();
    } catch (InputMismatchException e) {
        System.out.println("Введено не числовое значение для
кол-ва строк/столбцов или значений элемента матрицы");
    }
    catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Элементов в введенной строке
больше, чем задано перед этим");
    }
}
}

```

Код класса Matrix:

```

public class Matrix {
    private final int rows;
    private final int columns;
    private double[][] matrix;
    public Matrix(int m, int n) {
        this.rows = m;
        this.columns = n;
        this.matrix = new double[rows][columns];
        for(int i = 0; i < m; ++i) {
            for(int j = 0; j < n; ++j){
                this.matrix[i][j] = 0;
            }
        }
    }

    public Matrix(int m, int n, double[][] array) {
        this.rows = m;
        this.columns = n;
        this.matrix = new double[rows][columns];

        for(int i = 0; i < m; ++i) {
            System.arraycopy(array[i], 0, this.matrix[i], 0, n);
        }
    }
}

```

```

public void swap_rows(int k) {
    double min, max;
    min = max = this.matrix[0][k];
    int min_row = 0, max_row = 0;

    // поиск строк с минимальным и максимальным элементом в
k-столбце
    for(int i = 1; i < this.rows; ++i) {
        if(this.matrix[i][k] > max) {
            max = this.matrix[i][k];
            max_row = i;
        } else if (this.matrix[i][k] < min) {
            min = this.matrix[i][k];
            min_row = i;
        }
    }

    // перестановка строк
    for(int j = 0; j < this.columns; ++j) {
        double temp = this.matrix[min_row][j];
        this.matrix[min_row][j] = this.matrix[max_row][j];
        this.matrix[max_row][j] = temp;
    }
}

public void make_squared() {
    try {
        double[][] temp_matrix = new
double[this.rows][this.columns];
        for (int i = 0; i < this.rows; ++i) {
            for (int j = 0; j < this.columns; ++j) {
                int sum = 0;
                for (int z = 0; z < this.rows; ++z) {
                    sum += this.matrix[i][z] *
this.matrix[z][j];
                }
                temp_matrix[i][j] = sum;
            }
        }
        for(int i = 0; i < this.rows; ++i) {
            System.arraycopy(temp_matrix[i], 0,
this.matrix[i], 0, this.columns);
        }
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Матрицу нельзя возвести в
квадрат, т.к количество строк и столбцов не равно");
    }
}

public void print() {
    for(int i = 0; i < this.rows; ++i) {
        for(int j = 0; j < this.columns; ++j) {
            System.out.printf("%.1f ", this.matrix[i][j]);

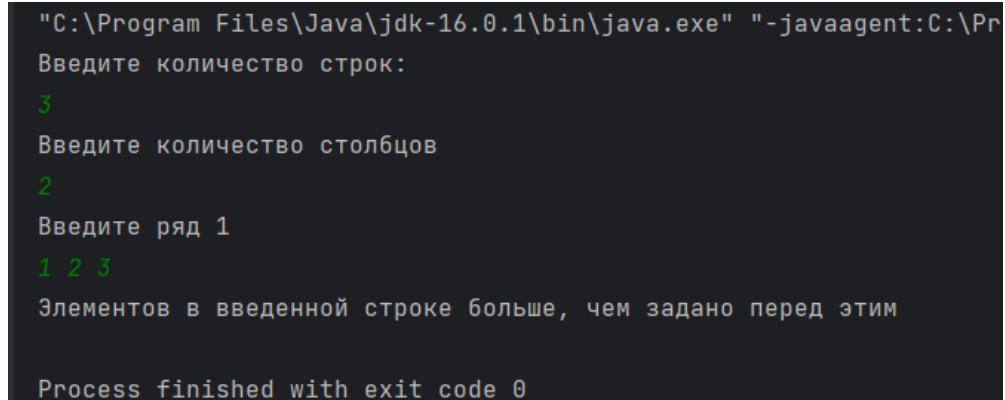
```

```

    }
    System.out.println();
}
}
}

```

Работа программы показана на рисунке 1.



```

"C:\Program Files\Java\jdk-16.0.1\bin\java.exe" "-javaagent:C:\Pr
Введите количество строк:
3
Введите количество столбцов
2
Введите ряд 1
1 2 3
Элементов в введенной строке больше, чем задано перед этим
Process finished with exit code 0

```

Рисунок 1 – Работа программы 1

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.

Код класса ContinuedFraction:

```

import java.util.Scanner;

public class ContinuedFraction {
    private final double constant;
    private final double[] coefficients;
    private final int coef_number;
    public ContinuedFraction(double x, double[] coefs) {
        this.constant = x;
        this.coefficients = new double[coefs.length];
        this.coef_number = coefs.length;
        System.arraycopy(coefs, 0, this.coefficients, 0,
coefs.length);
    }

    public double calculate(int n) {
        try {
            double value = this.coefficients[n - 1];
            for (int i = n - 2; i >= 0; --i) {

```

```

        value = this.coefficients[i] + this.constant
/ value;
    }
    return value;
} catch (ArithmeticException e) {
    System.out.println("Ошибка при вычислении
значения дроби");
    return 0;
}
}

public double sum(ContinuedFraction fraction) {
    try {
        return this.calculate(this.coef_number) +
fraction.calculate(fraction.coef_number);
    } catch (ArithmeticException e) {
        System.out.println("Ошибка при вычислении суммы
дробей");
        return 0;
    }
}

public double diff(ContinuedFraction fraction) {
    try {
        return this.calculate(this.coef_number) -
fraction.calculate(fraction.coef_number);
    } catch (ArithmeticException e) {
        System.out.println("Ошибка при вычислении
разности дробей");
        return 0;
    }
}

public double multiplication(ContinuedFraction fraction)
{
    try {
        return this.calculate(this.coef_number) *
fraction.calculate(fraction.coef_number);
    } catch (ArithmeticException e) {
        System.out.println("Ошибка при вычислении
произведения дробей");
        return 0;
    }
}

public double divison(ContinuedFraction fraction) {
    try {
        return this.calculate(this.coef_number) /
fraction.calculate(fraction.coef_number);
    } catch (ArithmeticException e) {
        System.out.println("Ошибка при вычислении
деления дробей");
        return 0;
    }
}

```

```

    }
}
public static void main(String[] args) {
    try {
        Scanner sc = new Scanner(System.in);
        System.out.println("Введите коэффициенты для
первой дроби: ");
        String[] coefs1_str = sc.nextLine().split(" ");
        double[] coefs1 = new double[coefs1_str.length];
        for (int i = 0; i < coefs1_str.length; i++) {
            coefs1[i] =
Double.parseDouble(coefs1_str[i]);
        }
        //double[] coefs1 = {1, 3, 2, 5};
        ContinuedFraction fraction1 = new
ContinuedFraction(2, coefs1);

        //double[] coefs2 = {3, 4, 1, 2};
        System.out.println("Введите коэффициенты для
второй дроби: ");
        String[] coefs2_str = sc.nextLine().split(" ");
        double[] coefs2 = new double[coefs2_str.length];
        for (int i = 0; i < coefs2_str.length; i++) {
            coefs2[i] =
Double.parseDouble(coefs2_str[i]);
        }
        ContinuedFraction fraction2 = new
ContinuedFraction(2, coefs2);

        System.out.println("Значение дробей:");
        System.out.println(fraction1.calculate(4));
        System.out.println(fraction2.calculate(4));

        System.out.println("Сумма дробей " +
fraction1.sum(fraction2));
        System.out.println("Разность дробей " +
fraction1.diff(fraction2));
        System.out.println("Произведение дробей " +
fraction1.multiplication(fraction2));
        System.out.println("Деление дробей " +
fraction1.divison(fraction2));

    } catch (NumberFormatException e) {
        System.out.println("Введенное значение
коэффициента не является числом");
    }
}
}

```

Работа программы показана на рисунке 2.

```
Введите коэффициенты для первой дроби:
1 3 2 5
Введите коэффициенты для второй дроби:
3 4 1 2
Значение дробей:
1.5217391304347827
3.4
Сумма дробей 4.921739130434783
Разность дробей -1.8782608695652172
Произведение дробей 5.173913043478261
Деление дробей 0.44757033248081846

Process finished with exit code 0
```

Рисунок 2 – Работа программы 2

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

Код класса Main:

```
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        try {
            ArrayList<Book> books = new ArrayList<>();
            books.add(new Book(0, "Энциклопедия", new
String[]{"Иванов"}, "Издательство1",
                2003, 350, 1000, "Твердый"));
            books.add(new Book(1, "Учебник математики", new
String[]{"Петров", "Иванов"}, "Издательство2",
                2007, 200, 570, "Твердый"));
            books.add(new Book(2, "Детектив", new
String[]{"Конан Дойл"}, "Издательство1",
                2010, -167, 400, "Мягкий"));
            books.add(new Book(-3, "Словарь", new
String[]{"Петров"}, "Издательство3",
                2001, 300, 250, "Мягкий"));

            Scanner scan = new Scanner(System.in);
            // вывод книг заданного автора
            System.out.println("Введите имя автора:");
            String author = scan.nextLine();
```



```

        ArrayList<Book> filtered_author_books =
filter_by_author(books, author);
        if (filtered_author_books.size() == 0) {
            System.out.println("Книг заданного автора нет");
        } else {
            for (Book b : filtered_author_books) {
                System.out.println(b.toString());
            }
        }

        // вывод книг заданного издательства
        System.out.println("Введите название
издательства:");
        String publisher = scan.nextLine();
        ArrayList<Book> filtered_publisher_books =
filter_by_publisher(books, publisher);
        if (filtered_publisher_books.size() == 0) {
            System.out.println("Книг, выпущенных заданным
издательством, нет");
        } else {
            for (Book b : filtered_publisher_books) {
                System.out.println(b.toString());
            }
        }

        // вывод книг, выпущенных после заданного года
        System.out.println("Введите минимальный год
выпуска");
        int year = scan.nextInt();
        ArrayList<Book> filtered_year_books =
filter_by_year(books, year);
        if (filtered_author_books.size() == 0) {
            System.out.println("Книг, выпущенных после
заданного года, нет");
        } else {
            for (Book b : filtered_year_books) {
                System.out.println(b.toString());
            }
        }
    } catch (BookInitializationException e) {
        System.out.println("Ошибка инициализации: " +
e.getMessage());
    }
}

    public static ArrayList<Book>
filter_by_author(ArrayList<Book> books, String author) {
        ArrayList<Book> filtered_books = new ArrayList<>();
        for(Book book: books) {
            boolean found_author = false;
            for(String auth: book.getAuthors()) {
                if(auth.equals(author)) {
                    found_author = true;

```

```

        break;
    }
}
if(found_author) {
    filtered_books.add(book);
}
}
return filtered_books;
}

public static ArrayList<Book>
filter_by_publisher(ArrayList<Book> books, String publisher) {
    ArrayList<Book> filtered_books = new ArrayList<>();
    for(Book book: books) {
        if(book.getPublisher().equals(publisher)) {
            filtered_books.add(book);
        }
    }
    return filtered_books;
}

public static ArrayList<Book> filter_by_year(ArrayList<Book>
books, int year) {
    ArrayList<Book> filtered_books = new ArrayList<>();
    for(Book book: books) {
        if(book.getYear_publication() >= year) {
            filtered_books.add(book);
        }
    }
    return filtered_books;
}
}

```

Код класса Book:

```

public class Book {
    private int id;
    private String title;
    private String[] authors;
    private String publisher;
    private int year_publication;
    private int page_number;
    private float price;
    private String cover_type;

    public Book(int id, String title, String[] authors, String
publisher, int year_publication,
        int page_number, float price, String cover_type)
throws BookInitializationException {
        validateInitialization(id, title, authors, publisher,
year_publication,
            page_number, price, cover_type);
        this.id = id;
        this.title = title;
    }
}

```

```

        this.authors = authors;
        this.publisher = publisher;
        this.year_publication = year_publication;
        this.page_number = page_number;
        this.price = price;
        this.cover_type = cover_type;
    }

    @Override
    public String toString() {
        return "Книга: '" + this.title + "', " + String.join(", ", this.authors) +
            ", изд." + this.publisher + ", " +
            this.year_publication + "г., " + this.page_number +
            " стр., переплет: " + this.cover_type + ", " +
            this.price + " руб.";
    }

    public static void validateInitialization(int id, String
title, String[] authors,
                                           String publisher,
int year_publication,
                                           int page_number,
float price,
                                           String
cover_type) throws BookInitializationException {
        validateId(id);
        validateTitle(title);
        validateAuthors(authors);
        validatePublisher(publisher);
        validateYearPublication(year_publication);
        validatePageNumber(page_number);
        validatePrice(price);
        validateCoverType(cover_type);
    }

    public static void validateId(int id) throws
BookInitializationException {
        if (id <= 0) {
            throw new BookInitializationException("Id должно
быть больше нуля");
        }
    }

    public static void validateTitle(String title) throws
BookInitializationException {
        if (title.equals("")) {
            throw new BookInitializationException("Название
книги не должно быть пустой строкой");
        }
    }
}

```

```

        public static void validateAuthors(String[] authors) throws
BookInitializationException {
            for (String author: authors) {
                if (author.equals("")) {
                    throw new BookInitializationException("Имя
автора не должно быть пустой строкой");
                }
            }
        }

        public static void validatePublisher(String publisher)
throws BookInitializationException {
            if (publisher.equals("")) {
                throw new BookInitializationException("Название
издателя не должно быть пустой строкой");
            }
        }

        public static void validateYearPublication(int
year_publication) throws BookInitializationException {
            if (year_publication <=0) {
                throw new BookInitializationException("Год написания
книги должен быть больше нуля");
            }
        }

        public static void validatePageNumber(int page_number)
throws BookInitializationException {
            if (page_number <=0) {
                throw new BookInitializationException("Количество
страниц у книги должно быть больше нуля");
            }
        }

        public static void validatePrice(float price) throws
BookInitializationException {
            if (price <=0) {
                throw new BookInitializationException("Цена у книги
должна быть больше нуля");
            }
        }

        public static void validateCoverType(String cover_type)
throws BookInitializationException {
            if (cover_type.equals("")) {
                throw new BookInitializationException("Название типа
перплета книги не должно быть пустой строкой");
            }
        }

        public int getId() {
            return id;
        }

```

```

public String getTitle() {
    return this.title;
}

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

public String getPublisher() {
    return publisher;
}

public int getYear_publication() {
    return year_publication;
}

public int getPage_number() {
    return page_number;
}

public float getPrice() {
    return price;
}

public String getCover_type() {
    return cover_type;
}

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

public void setTitle(String title) {
    this.title = title;
}

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

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

public void setYear_publication(int year_publication) {
    this.year_publication = year_publication;
}

public void setPage_number(int page_number) {
    this.page_number = page_number;
}

```

```

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

    public void setCover_type(String cover_type) {
        this.cover_type = cover_type;
    }
}

```

Код класса BookInitializationException:

```

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

```

Работа программы показана на рисунке 3.

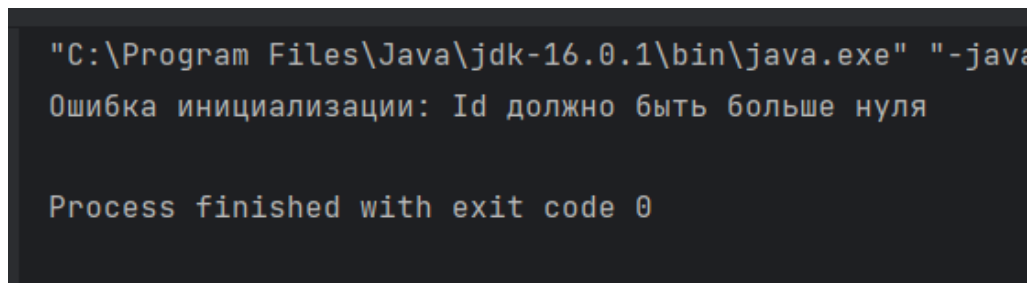


Рисунок 3 – Работа программы 3

Задание 4: выполнить задания из варианта 2 лабораторной работы 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

Код класса Main:

```

import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        try {
            ArrayList<House> houses = new ArrayList<>();
            houses.add(new House(1, 18, 60, 5, 2,
                                "Бауманская", "Тип1", 10));
            houses.add(new House(2, 10, 77, 3, 3,
                                "Тверская", "", 30));
            houses.add(new House(3, 20, -100, 5, 4,
                                "Арбат", "Тип1", 60));
            houses.add(new House(4, 4, 60, 1, 2,
                                "Бауманская", "Тип3", 10));

            Scanner scan = new Scanner(System.in);

```

```

        // вывод списка квартир с заданным числом комнат
        System.out.println("Введите необходимое число
комнат:");
        int num = scan.nextInt();
        ArrayList<House> houses_with_num_rooms =
filter_by_room_number(houses, num);
        if (houses_with_num_rooms.size() == 0) {
            System.out.println("Квартир с заданным числом
комнат нет");
        } else {
            for (House h : houses_with_num_rooms) {
                System.out.println(h.toString());
            }
        }

        // вывод списка квартир с заданным числом комнат и
расположенных на этаже в интервале
        System.out.println("Введите минимальный возможный
этаж:");
        int min_floor = scan.nextInt();
        System.out.println("Введите максимально возможный
этаж:");
        int max_floor = scan.nextInt();
        ArrayList<House> houses_filtered_floor =
filter_by_floor(houses_with_num_rooms, min_floor, max_floor);
        if (houses_filtered_floor.size() == 0) {
            System.out.println("Квартир с заданным числом
комнат и номером этажа в заданном интервале нет");
        } else {
            for (House h : houses_filtered_floor) {
                System.out.println(h.toString());
            }
        }
        // вывод списка квартир, имеющих площадь,
превосходящую заданную
        System.out.println("Введите минимально возможную
площадь:");
        int min_area = scan.nextInt();
        ArrayList<House> houses_filtered_area =
filter_by_area(houses, min_area);
        if (houses_filtered_area.size() == 0) {
            System.out.println("Квартир с площадью,
превосходящую заданную, нет");
        } else {
            for (House h : houses_filtered_area) {
                System.out.println(h.toString());
            }
        }
    } catch (HouseInitializationException e) {
        System.out.println("Ошибка инициализации: " +
e.getMessage());
    }
}

```

```

    }

    public static ArrayList<House>
    filter_by_room_number(ArrayList<House> houses, int n) {
        ArrayList<House> filtered_houses = new ArrayList<>();
        for(House house: houses) {
            if(house.getRooms_number() == n) {
                filtered_houses.add(house);
            }
        }
        return filtered_houses;
    }

    public static ArrayList<House>
    filter_by_floor(ArrayList<House> houses, int min_floor, int
    max_floor) {
        ArrayList<House> filtered_houses = new ArrayList<>();
        for(House house: houses) {
            if(house.getFloor() >= min_floor && house.getFloor()
    <= max_floor) {
                filtered_houses.add(house);
            }
        }
        return filtered_houses;
    }

    public static ArrayList<House>
    filter_by_area(ArrayList<House> houses, int area) {
        ArrayList<House> filtered_houses = new ArrayList<>();
        for(House house: houses) {
            if(house.getArea() >= area) {
                filtered_houses.add(house);
            }
        }
        return filtered_houses;
    }
}

```

Код класса House:

```

public class House {
    private int id;
    private int number;
    private float area;
    private int floor;
    private int rooms_number;
    private String street_name;
    private String building_type;
    private float exploit_period;

    public House(int id, int number, float area, int floor, int
    rooms_number,
                String street_name, String building_type, float
    exploit_period) throws HouseInitializationException {

```



```

        validateInitialization(id, number, area, floor,
rooms_number, street_name, building_type, exploit_period);
        this.id = id;
        this.number = number;
        this.area = area;
        this.floor = floor;
        this.rooms_number = rooms_number;
        this.street_name = street_name;
        this.building_type = building_type;
        this.exploit_period = exploit_period;
    }

    public static void validateInitialization(int id, int
number, float area, int floor, int rooms_number,
                                         String
street_name, String building_type,
                                         float
exploit_period) throws HouseInitializationException {
        validateId(id);
        validateNumber(number);
        validateArea(area);
        validateFloor(floor);
        validateRoomsNumber(rooms_number);
        validateStreetName(street_name);
        validateBuildingType(building_type);
        validateExploitPeriod(exploit_period);
    }

    public static void validateId(int id) throws
HouseInitializationException {
        if (id <= 0) {
            throw new HouseInitializationException("Id должно
быть больше нуля");
        }
    }

    public static void validateNumber(int number) throws
HouseInitializationException {
        if (number <= 0) {
            throw new HouseInitializationException("Номер
квартиры должен быть больше 0");
        }
    }

    public static void validateArea(float area) throws
HouseInitializationException {
        if (area <= 0) {
            throw new HouseInitializationException("Площадь
квартиры должно быть больше 0");
        }
    }

```

```

        public static void validateFloor(int floor) throws
HouseInitializationException {
            if (floor <= 0) {
                throw new HouseInitializationException("Этаж должен
быть больше 0");
            }
        }

        public static void validateRoomsNumber(int rooms_number)
throws HouseInitializationException {
            if (rooms_number <= 0) {
                throw new HouseInitializationException("Количество
комнат квартиры должно быть больше 0");
            }
        }

        public static void validateStreetName(String street_name)
throws HouseInitializationException {
            if (street_name.equals("")) {
                throw new HouseInitializationException("Название
улицы не должно быть пустой строкой");
            }
        }

        public static void validateBuildingType(String
building_type) throws HouseInitializationException {
            if (building_type.equals("")) {
                throw new HouseInitializationException("Тип здания
не должно быть пустой строкой");
            }
        }

        public static void validateExploitPeriod(float
exploit_period) throws HouseInitializationException {
            if (exploit_period <= 0) {
                throw new HouseInitializationException("Период
эксплуатации квартиры должно быть больше 0");
            }
        }

        @Override
        public String toString() {
            return "Квартира № " + this.number + ", площадь " +
this.area + "м^2, " +
                this.floor + " этаж, " + this.rooms_number + "
комнат, ул." + this.street_name +
                ", типа здания: " + this.building_type + ", срок
эксплуатации: " +
                this.exploit_period + " лет.";
        }

        public int getId() {

```

```

        return id;
    }

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

    public int getNumber() {
        return number;
    }

    public void setNumber(int number) {
        this.number = number;
    }

    public float getArea() {
        return area;
    }

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

    public int getFloor() {
        return floor;
    }

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

    public int getRooms_number() {
        return rooms_number;
    }

    public void setRooms_number(int rooms_number) {
        this.rooms_number = rooms_number;
    }

    public String getStreet_name() {
        return street_name;
    }

    public void setStreet_name(String street_name) {
        this.street_name = street_name;
    }

    public String getBuilding_type() {
        return building_type;
    }

    public void setBuilding_type(String building_type) {
        this.building_type = building_type;
    }

```

```

    }

    public float getExploit_period() {
        return exploit_period;
    }

    public void setExploit_period(float exploit_period) {
        this.exploit_period = exploit_period;
    }
}

```

Код класса HouseInitializationException:

```

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

```

Работа программы показана на рисунке 4.

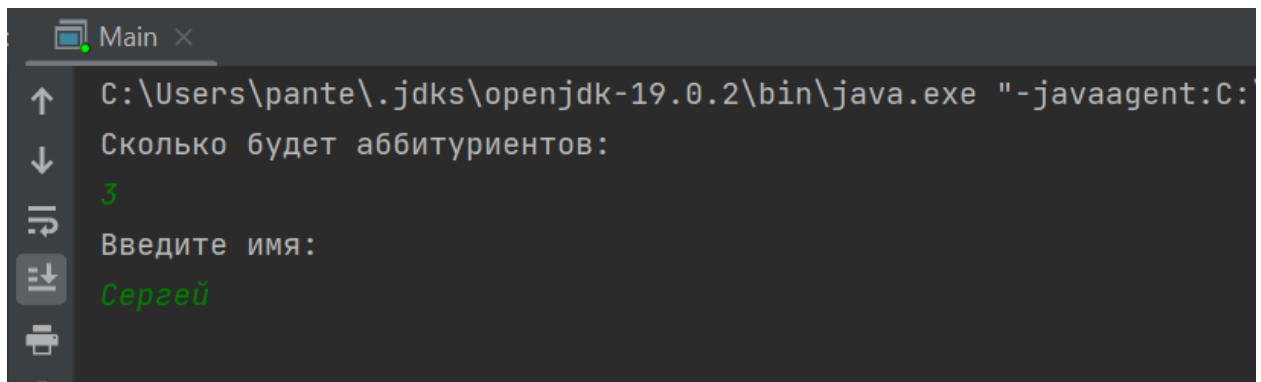


Рисунок 4 – Работа программы 4

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

Код класса Main:

```

Найти в строке наибольшее число цифр, идущих по
import java.io.*;
import java.nio.Buffer;
import java.util.Objects;

```

```

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(new
            FileReader("src/input.txt"));
            BufferedWriter writer = new BufferedWriter(new
            FileWriter("src/output.txt"));
            String str;
            while ((str = reader.readLine()) != null) {
                int count = 0;
                int count_max = 0;
                for (char c : str.toCharArray()) {
                    if (Character.isDigit(c)) {
                        count += 1;
                    } else {
                        if (count > count_max) {
                            count_max = count;
                        }
                        count = 0;
                    }
                }
                writer.write(count_max +
                System.lineSeparator());
            }
            reader.close();
            writer.close();
        } catch (IOException e) {
            System.out.println("Ошибка при чтении/записи в
            файл");
            e.printStackTrace();
        }
    }
}

```

ряд.

Работа программы показана на рисунках 5 – 6.

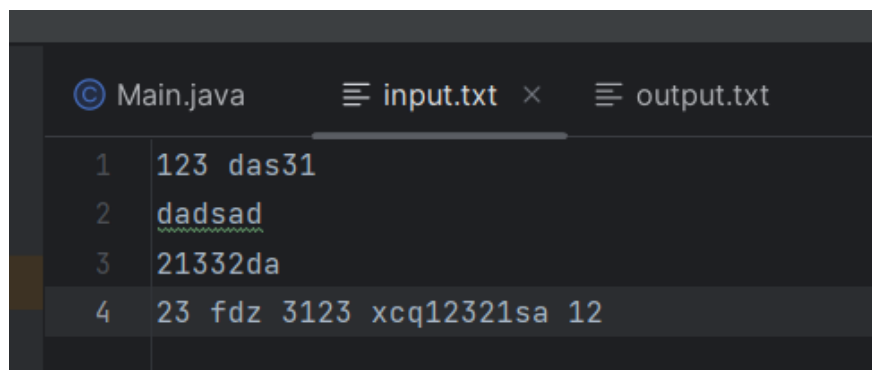


Рисунок 5 – Входной файл

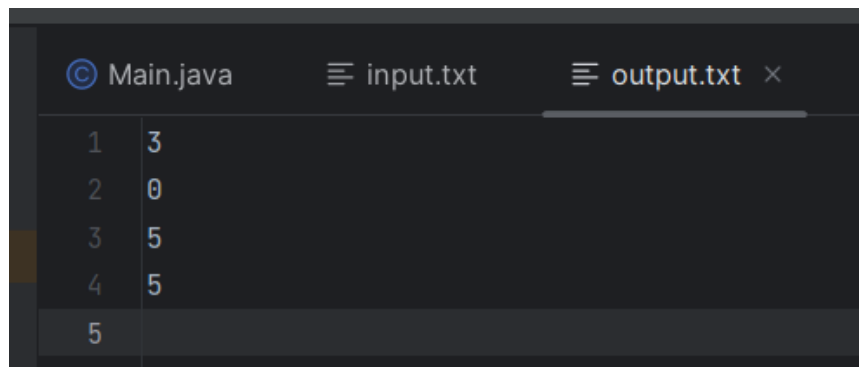


Рисунок 6 – Выходной файл

Задание 6: 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.

Код класса Main:

```
import java.io.*;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import java.util.TreeMap;

public class Main {
    public static void main(String[] args) {
        // ввод заданного списка слов
        HashMap<String, Integer> words_freq = new HashMap<>();
        Scanner scanner = new Scanner(System.in);
        System.out.println("Введите слова для анализа частоты в
нижнем регистре черел пробел");
        String[] words = scanner.nextLine().split(" ");
        for (String w: words) {
            words_freq.put(w, 0);
        }

        try {
```

```

        BufferedReader reader = new BufferedReader(new
FileReader("src/input.txt"));
        BufferedWriter writer = new BufferedWriter(new
FileWriter("src/output.txt"));

        // построчное чтение файла
        String line;
        while ((line = reader.readLine()) != null) {
            for (String word: line.toLowerCase().split(" "))
{
                if (words_freq.get(word) != null) {
                    words_freq.put(word,
words_freq.get(word) + 1);
                }
            }

            // сортируем по значениям частоты слов
            TreeMap<Integer, String> sorted_words = new
TreeMap<>();
            for(Map.Entry<String, Integer> w:
words_freq.entrySet()) {
                sorted_words.put(w.getValue(), w.getKey());
            }

            // вывод слов и их частот в отсортированном виде и
запись в файл
            for(Map.Entry<Integer, String> w:
sorted_words.entrySet()) {
                System.out.println(w.getValue() + " " +
w.getKey());
                writer.write(w.getValue() + " " + w.getKey() +
System.lineSeparator());
            }

            reader.close();
            writer.close();

        } catch (IOException e) {
            System.out.println("Ошибка при чтении/записи в
файл");
            e.printStackTrace();
        }
    }
}

```

Работа программы показана на рисунках 7–8.

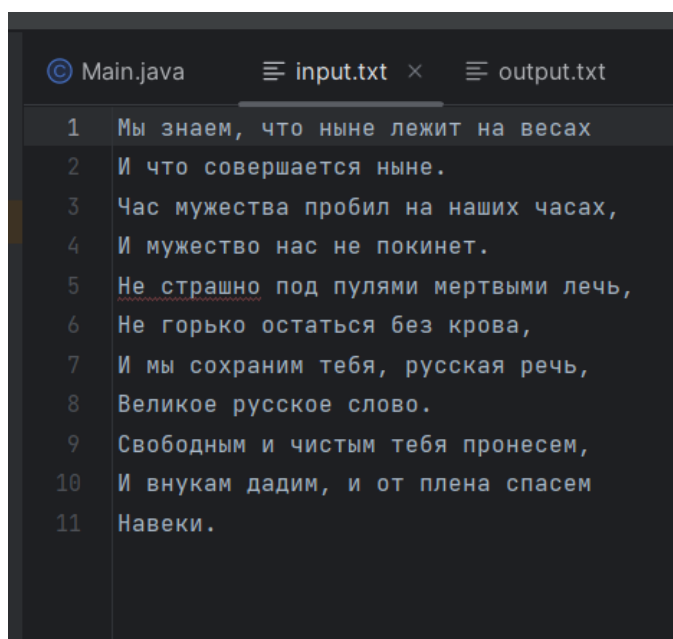


Рисунок 7 – Входной файл

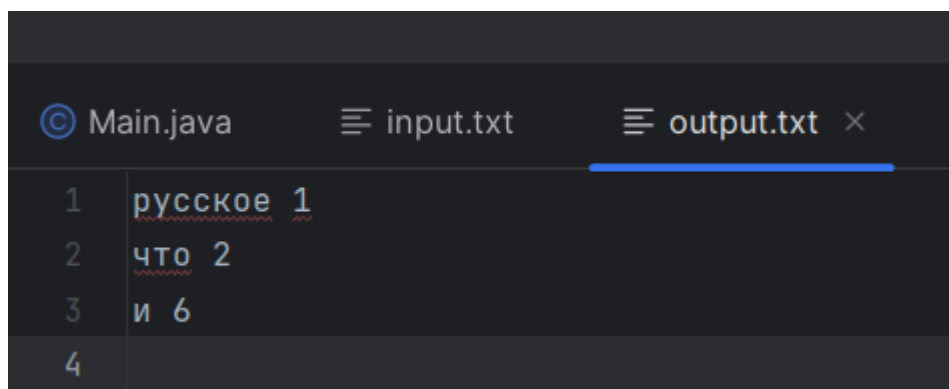


Рисунок 8 – Выходной файл

Задание 7: при выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File. Файл содержит символы, слова, целые числа и числа с плавающей запятой. Определить все данные, тип которых вводится из командной строки.

Код класса Main:

```
import java.io.*;
import java.nio.file.*;
import java.util.*;
import java.util.regex.*;

public class Main {
```



```

public static void main(String[] args) {
    if (args.length != 2) {
        System.out.println("Больше двух аргументов");
        return;
    }

    // определение пути входного файла и типа данных для
    // фильтрации
    String inputFile = args[0];
    String dataType = args[1];
    List<String> data = readFile(inputFile);

    if (data == null) {
        System.out.println("Error reading file.");
        return;
    }

    // Фильтр данных по типу
    List<String> filteredData = filterDataByType(data,
    dataType);

    // Создание директории и файла для вывода результатов
    String outputDirName = "output";
    File outputDir = new File(outputDirName);
    if (!outputDir.exists()) {
        Boolean result = outputDir.mkdir();
    }

    String outputFileName = outputDirName +
    "/filtered_data.txt";
    File outputFile = new File(outputFileName);

    writeFile(outputFile, filteredData);
    System.out.println("Результат фильтрации записан в " +
    outputFileName);
}

private static List<String> readFile(String fileName) {
    List<String> data = new ArrayList<>();
    try (BufferedReader br = new BufferedReader(new
    FileReader(fileName))) {
        String line;
        while ((line = br.readLine()) != null) {
            data.addAll(Arrays.asList(line.split("\\s+")));
        }
    } catch (IOException e) {
        e.printStackTrace();
        return null;
    }
    return data;
}

```

```

        private static List<String> filterDataByType(List<String>
data, String dataType) {
    List<String> filteredData = new ArrayList<>();
    Pattern pattern;
    // фильтрация данных с помощью регулярных выражений
    switch (dataType.toLowerCase()) {
        case "integer":
            pattern = Pattern.compile("^\\d+$");
            break;
        case "float":
            pattern = Pattern.compile("^([+-]?\\d*\\.\\d+$");
            break;
        case "word":
            pattern = Pattern.compile("[a-zA-Z]+$");
            break;
        case "symbol":
            pattern = Pattern.compile("^.$");
            break;
        default:
            System.out.println("Незнакомый тип данных: " +
dataType);
            return filteredData;
    }

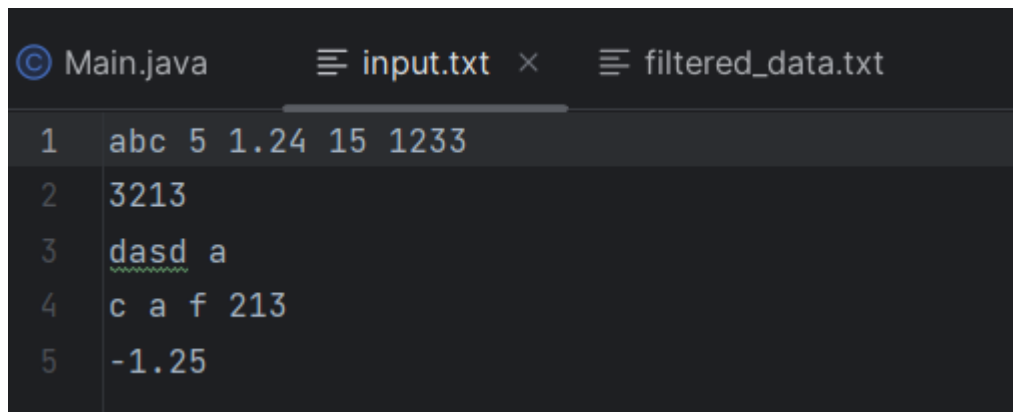
    for (String item : data) {
        if (pattern.matcher(item).matches()) {
            filteredData.add(item);
        }
    }

    return filteredData;
}

private static void writeFile(File file, List<String> data)
{
    try (BufferedWriter writer = new BufferedWriter(new
FileWriter(file))) {
        for (String item : data) {
            writer.write(item);
            writer.newLine();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

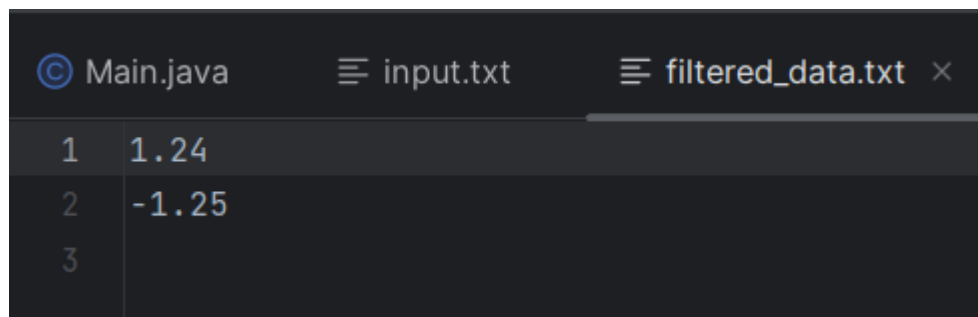
```

Работа программы показана на рисунках 9–10.



```
© Main.java  input.txt  filtered_data.txt
1 abc 5 1.24 15 1233
2 3213
3 dasd a
4 c a f 213
5 -1.25
```

Рисунок 9 – Исходный файл



```
© Main.java  input.txt  filtered_data.txt
1 1.24
2 -1.25
3
```

Рисунок 10 – Выходной файл

Задание 8: 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 класса Main:

```
import java.io.*;
import java.nio.file.*;
import java.util.*;
import java.util.regex.*;

public class Main {

    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Необходимо передать только 1
параметр - путь к входному файлу");
            return;
        }

        // путь входного файла
        String inputFile = args[0];
        List<String> lines = readFile(inputFile);
```

```

// проверка на содержимое файла
if (lines == null) {
    System.out.println("Error reading file.");
    return;
}

// убираем лишние слова из каждой строки
List<String> modifiedLines = new ArrayList<>();
for (String line : lines) {
    modifiedLines.add(removeWords(line));
}

// Создаем новую директорию и файл в ней для вывода
String outputDirName = "output";
File outputDir = new File(outputDirName);
if (!outputDir.exists()) {
    outputDir.mkdir();
}

String outputFileName = outputDirName +
"/modified_data.txt";
File outputFile = new File(outputFileName);

writeFile(outputFile, modifiedLines);
System.out.println("Модифицированные данные записаны в "
+ outputFileName);
}

private static List<String> readFile(String fileName) {
    List<String> lines = new ArrayList<>();
    try (BufferedReader br = new BufferedReader(new
FileReader(fileName))) {
        String line;
        while ((line = br.readLine()) != null) {
            lines.add(line);
        }
    } catch (IOException e) {
        e.printStackTrace();
        return null;
    }
    return lines;
}

private static String removeWords(String line) {
    String[] words = line.split("\\s+");
    // target - слова длиной от 3 до 5 символов
    List<String> targetWords = new ArrayList<>();
    List<String> nonTargetWords = new ArrayList<>();

    // поиск таргетных слов
    for (String word : words) {
        if (word.length() >= 3 && word.length() <= 5) {
            targetWords.add(word);
        }
    }
}

```

```

        } else {
            nonTargetWords.add(word);
        }
    }

    // поиск ближайшего четного числа таргетных слов
    int toRemove = targetWords.size();
    if (toRemove % 2 != 0) {
        toRemove--; // Reduce to even number
    }

    // удаляем слова максимальные по длине
    targetWords.sort(Comparator.comparingInt(String::length).reverse
d());
    for (int i = 0; i < toRemove; i++) {
        targetWords.remove(0);
    }

    List<String> resultWords = new
ArrayList<>(nonTargetWords);
    resultWords.addAll(targetWords);

    return String.join(" ", resultWords);
}

private static void writeFile(File file, List<String> data)
{
    try (BufferedWriter writer = new BufferedWriter(new
FileWriter(file))) {
        for (String line : data) {
            writer.write(line);
            writer.newLine();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

Работа программы показана на рисунках 11 – 12

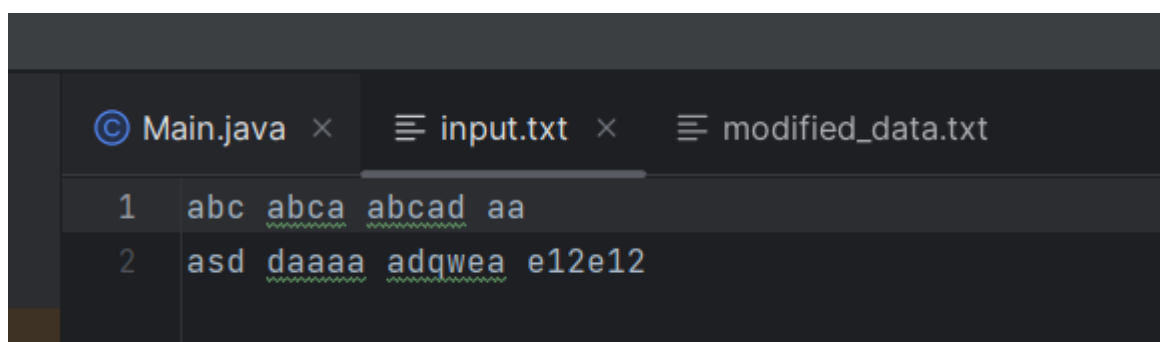
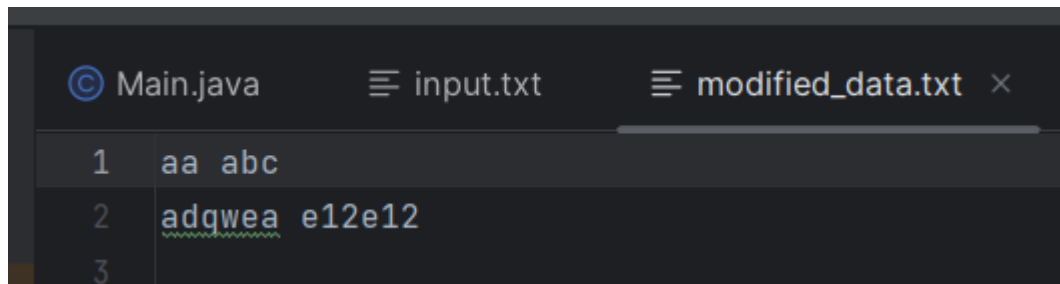


Рисунок 11 – Исходный файл



```
© Main.java  input.txt  modified_data.txt ×
1 aa abc
2 adgwea e12e12
3
```

Рисунок 12 – Выходной файл

Вывод: были изучена работа с исключениями и файлами в java.