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**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

ОТЧЕТ
по лабораторной работе №5

Название: Исключения и файлы

Дисциплина Языки программирования для работы с большими данными

Студент ИУ6–22М (Группа) _____ (Подпись, дата) М.Э.Хабаров (И.О. Фамилия)

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

2024 г

Цель: освоить принципы исключений и файлов на языке Java.

Задание №1

Формулировка задания и код программы представлены в листинге 1:

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

/*
    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
*/

class Fraction {
    private int numerator;
    private int denominator;

    public Fraction() {
        this.numerator = 0;
        this.denominator = 1;
    }

    public Fraction(int numerator, int denominator) {
        this.numerator = numerator;
        if (denominator != 0) {
            this.denominator = denominator;
        } else {
            System.out.println("Знаменатель не может быть равен нулю.
Устанавливаю значение по умолчанию (1).");
            this.denominator = 1;
        }
    }

    public Fraction add(Fraction other) {
        int newNumerator = this.numerator * other.denominator +
other.numerator * this.denominator;
        int newDenominator = this.denominator * other.denominator;
        return new Fraction(newNumerator, newDenominator);
    }

    public Fraction subtract(Fraction other) {
        int newNumerator = this.numerator * other.denominator -
other.numerator * this.denominator;
        int newDenominator = this.denominator * other.denominator;
        return new Fraction(newNumerator, newDenominator);
    }

    public Fraction multiply(Fraction other) {
        int newNumerator = this.numerator * other.numerator;
        int newDenominator = this.denominator * other.denominator;
```

```
        return new Fraction(newNumerator, newDenominator);
    }

    public Fraction divide(Fraction other) {
        if (other.numerator == 0) {
            System.out.println("Деление на ноль. Возвращаю дробь (0/1).");
            return new Fraction(0, 1);
        }

        int newNumerator = this.numerator * other.denominator;
        int newDenominator = this.denominator * other.numerator;
        return new Fraction(newNumerator, newDenominator);
    }

    public String toString() {
        return this.numerator + "/" + this.denominator;
    }
}

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

        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Введите k: ");
            int k = scanner.nextInt(); // Количество дробей в массиве

            Fraction[] fractions = new Fraction[k];

            // Ввод значений для массива дробей
            System.out.println("Введите дроби:");
            for (int i = 0; i < k; i++) {
                System.out.println("Числитель " + i + "-й " + "дроби: ");
                int m = scanner.nextInt();
                System.out.println("Знаменатель " + i + "-й " + "дроби: ");
                int o = scanner.nextInt();
                fractions[i] = new Fraction(m, o); // Пример значений для
дробей
            }

            // Вывод значений массива дробей
            System.out.println("Исходные дроби:");
            for (int i = 0; i < k; i++) {
                System.out.println(fractions[i].toString());
            }

            modifyArray(fractions);

            // Вывод измененных дробей
            System.out.println("\nИзмененные дроби:");
            for (int i = 0; i < k; i++) {
                System.out.println(fractions[i].toString());
            }
        }  catch (InputMismatchException e) {
            System.out.println("Неверный тип данных");
        }  catch (OutOfMemoryError e) {
            System.out.println("Оперативка исчерпана");
        }
    }

    public static void modifyArray(Fraction[] fractions) {
        for (int i = 0; i < fractions.length - 1; i += 2) {
            fractions[i] = fractions[i].add(fractions[i+1]);
```

```
        }
    }
}
```

Задание №2

Формулировка задания и код программы представлены в листинге 2:

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

class QuadraticEquation {
    private double a;
    private double b;
    private double c;

    public QuadraticEquation(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public QuadraticEquation(double a, double b) {
        this(a, b, 0);
    }

    public double[] findRoots() {
        double discriminant = b * b - 4 * a * c;
        if (discriminant > 0) {
            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            return new double[]{root1, root2};
        } else if (discriminant == 0) {
            double root = -b / (2 * a);
            return new double[]{root};
        } else {
            return new double[0];
        }
    }

    public double findExtremePoint() {
        return -b / (2 * a);
    }

    public String findInterval() {
        if (a > 0) {
            return "(-∞, " + findExtremePoint() + "]";
        } else if (a < 0) {
```

```
            return "[" + findExtremePoint() + ", +∞)";
        } else {
            return "The coefficient 'a' must be non-zero.";
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Enter the number of quadratic equations: ");
            int n = scanner.nextInt();
            QuadraticEquation[] equations = new QuadraticEquation[n];

            for (int i = 0; i < n; i++) {
                System.out.print("Enter coefficients for equation " + (i + 1)
+ " (a b c): ");
                double a = scanner.nextDouble();
                double b = scanner.nextDouble();
                double c = scanner.nextDouble();
                equations[i] = new QuadraticEquation(a, b, c);
            }

            double maxRoot = Double.NEGATIVE_INFINITY;
            double minRoot = Double.POSITIVE_INFINITY;

            for (QuadraticEquation equation : equations) {
                double[] roots = equation.findRoots();
                for (double root : roots) {
                    if (!Double.isNaN(root)) {
                        maxRoot = Math.max(maxRoot, root);
                        minRoot = Math.min(minRoot, root);
                    }
                }
            }

            System.out.println("The largest root is: " + maxRoot);
            System.out.println("The smallest root is: " + minRoot);
        } catch (InputMismatchException e) {
            System.out.println("Неверный тип данных");
        } catch (OutOfMemoryError e) {
            System.out.println("Оперативка исчерпана");
        }
    }
}
```

Задание №3

Формулировка задания и код программы представлены в листинге 3:

```
import java.util.*;
/*
    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: a) сведения об абонентах, у которых время внутригородских
разговоров превышает заданное;
    b) сведения об абонентах, которые пользовались междугородной связью;
    c) сведения об абонентах в алфавитном порядке.
```

```
*/

class Phone {
    private int id;
    private String lastName;
    private String firstName;
    private String middleName;
    private String address;
    private String creditCardNumber;
    private double debit;
    private double credit;
    private double localCallsTime;
    private double longDistanceCallsTime;

    public Phone(int id, String lastName, String firstName, String
middleName, String address, String creditCardNumber, double debit, double
credit, double localCallsTime, double longDistanceCallsTime) {
        this.id = id;
        this.lastName = lastName;
        this.firstName = firstName;
        this.middleName = middleName;
        this.address = address;
        this.creditCardNumber = creditCardNumber;
        this.debit = debit;
        this.credit = credit;
        this.localCallsTime = localCallsTime;
        this.longDistanceCallsTime = longDistanceCallsTime;
    }

    public int getId() {
        return id;
    }

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

    public String getLastName() {
        return lastName;
    }

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

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getMiddleName() {
        return middleName;
    }

    public void setMiddleName(String middleName) {
        this.middleName = middleName;
    }

    public String getAddress() {
        return address;
```

```
    }

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

    public String getCreditCardNumber() {
        return creditCardNumber;
    }

    public void setCreditCardNumber(String creditCardNumber) {
        this.creditCardNumber = creditCardNumber;
    }

    public double getDebit() {
        return debit;
    }

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

    public double getCredit() {
        return credit;
    }

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

    public double getLocalCallsTime() {
        return localCallsTime;
    }

    public void setLocalCallsTime(double localCallsTime) {
        this.localCallsTime = localCallsTime;
    }

    public double getLongDistanceCallsTime() {
        return longDistanceCallsTime;
    }

    public void setLongDistanceCallsTime(double longDistanceCallsTime) {
        this.longDistanceCallsTime = longDistanceCallsTime;
    }

    @Override
    public String toString() {
        return "Phone{" +
                "id=" + id +
                ", lastName='" + lastName + '\'' +
                ", firstName='" + firstName + '\'' +
                ", middleName='" + middleName + '\'' +
                ", address='" + address + '\'' +
                ", creditCardNumber=" + creditCardNumber +
                ", debit=" + debit +
                ", credit=" + credit +
                ", localCallsTime=" + localCallsTime +
                ", longDistanceCallsTime=" + longDistanceCallsTime +
                '}';
    }

    public static ArrayList<Phone> filterByLocalCallsTime(ArrayList<Phone>
```

```
phoneList, double time) {
        ArrayList<Phone> result = new ArrayList<>();
        for (Phone phone : phoneList) {
            if (phone.getLocalCallsTime() > time) {
                result.add(phone);
            }
        }
        return result;
    }

    public static ArrayList<Phone> filterByLongDistanceUsage(ArrayList<Phone>
phoneList) {
        ArrayList<Phone> result = new ArrayList<>();
        for (Phone phone : phoneList) {
            if (phone.getLongDistanceCallsTime() > 0) {
                result.add(phone);
            }
        }
        return result;
    }

    public static ArrayList<Phone> sortByLastName(ArrayList<Phone> phoneList)
{
        Collections.sort(phoneList,
Comparator.comparing(Phone::getLastName));
        return phoneList;
    }
}

public class Main {
    public static void main(String[] args) {
        ArrayList<Phone> phoneList = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Введите количество абонентов: ");
            int k = scanner.nextInt();

            for (int i = 0; i < k; ++i) {
                System.out.print("Введите id " + (i + 1) + " абонента: ");
                int number = scanner.nextInt();
                if (number == 666)
                    throw new InvalidPhoneException("Невалидный id");
                scanner.nextLine();
                System.out.print("Введите фамилию " + (i + 1) + " абонента:
");
                String surname = scanner.nextLine();
                if (surname.equals("rrr"))
                    throw new InvalidPhoneException("Невалидная фамилия");
                System.out.print("Введите имя абонента " + (i + 1) + ": ");
                String name = scanner.nextLine();
                System.out.print("Введите отчество абонента " + (i + 1) + ":
");
                String middlename = scanner.nextLine();
                System.out.print("Введите адрес абонента " + (i + 1) + ": ");
                String address = scanner.nextLine();
                System.out.print("Введите дебет абонента " + (i + 1) + ": ");
                double debit = scanner.nextDouble();
                System.out.print("Введите кредит абонента " + (i + 1) + ":
");
                double credit = scanner.nextDouble();
                scanner.nextLine();
                phoneList.add(new Phone(number, surname, name, middlename,
address, "1234 5678 9012 3456", debit, credit, 20.0, 10.0));
```

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

            System.out.println("Введенные значения: ");
            for(Phone phone : phoneList) {
                System.out.println(phone);
            }
            System.out.println();

            // Выводим информацию об абонентах с временем внутригородских
разговоров более 10 минут
            System.out.println("Абоненты с временем внутригородских
разговоров более 10 минут:");
            ArrayList<Phone> filteredByLocalCalls =
Phone.filterByLocalCallsTime(phoneList, 10.0);
            for (Phone phone : filteredByLocalCalls) {
                System.out.println(phone);
            }

            // Выводим информацию об абонентах, которые пользовались
междугородней связью
            System.out.println("\nАбоненты, которые пользовались
междугородней связью:");
            ArrayList<Phone> filteredByLongDistanceUsage =
Phone.filterByLongDistanceUsage(phoneList);
            for (Phone phone : filteredByLongDistanceUsage) {
                System.out.println(phone);
            }

            // Выводим информацию об абонентах в алфавитном порядке
            System.out.println("\nАбоненты в алфавитном порядке:");
            ArrayList<Phone> sortedByLastName =
Phone.sortByLastName(phoneList);
            for (Phone phone : sortedByLastName) {
                System.out.println(phone);
            }
        } catch (InputMismatchException e) {
        System.out.println("Неверный тип данных");
        } catch (OutOfMemoryError e) {
            System.out.println("Оперативка исчерпана");
        } catch (InvalidPhoneException e) {
            System.out.println("InvalidPhoneException " +
e.getLocalizedMessage());
        } catch (Exception e) {
            System.out.println(e.getLocalizedMessage());
        }
    }
}
```

Задание №4

Формулировка задания и код программы представлены в листинге 4:

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

/*
    Выполнить задания из варианта 2 лабораторной работы 3,
    реализуя собственные обработчики исключений и исключения ввода/вывода.
    Car: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер.
    Создать массив объектов. Вывести: а) список автомобилей заданной марки;
```

```
    b) список автомобилей заданной модели, которые эксплуатируются больше n
лет;
    c) список автомобилей заданного года выпуска, цена которых больше
указанной.
*/

class Car {
    private int id;
    private String brand;
    private String model;
    private int year;
    private String color;
    private double price;
    private String regNumber;

    public Car(int id, String brand, String model, int year, String color,
double price, String regNumber) {
        this.id = id;
        this.brand = brand;
        this.model = model;
        this.year = year;
        this.color = color;
        this.price = price;
        this.regNumber = regNumber;
    }

    public int getId() {
        return id;
    }

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

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public int getYear() {
        return year;
    }

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

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
```

```
        this.color = color;
    }

    public double getPrice() {
        return price;
    }

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

    public String getRegNumber() {
        return regNumber;
    }

    public void setRegNumber(String regNumber) {
        this.regNumber = regNumber;
    }

    @Override
    public String toString() {
        return "Car{" +
                "id=" + id +
                ", brand='" + brand + '\'' +
                ", model='" + model + '\'' +
                ", year=" + year +
                ", color='" + color + '\'' +
                ", price=" + price +
                ", regNumber='" + regNumber +
                '}';
    }

    public static ArrayList<Car> filterByBrand(ArrayList<Car> carList, String
brand) {
        ArrayList<Car> result = new ArrayList<>();
        for (Car car : carList) {
            if (car.getBrand().equalsIgnoreCase(brand)) {
                result.add(car);
            }
        }
        return result;
    }

    public static ArrayList<Car> filterByModelAndYears(ArrayList<Car>
carList, String model, int years) {
        ArrayList<Car> result = new ArrayList<>();
        for (Car car : carList) {
            if (car.getModel().equalsIgnoreCase(model) && (2024 -
car.getYear()) > years) {
                result.add(car);
            }
        }
        return result;
    }

    public static ArrayList<Car> filterByYearAndPrice(ArrayList<Car> carList,
int year, double price) {
        ArrayList<Car> result = new ArrayList<>();
        for (Car car : carList) {
            if (car.getYear() == year && car.getPrice() > price) {
                result.add(car);
            }
        }
```

```
        return result;
    }
}

public class Main {
    public static void main(String[] args) {
        ArrayList<Car> carList = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);
        try {
            for (int i = 0; i < 2; ++i) {
                System.out.print("Введите id " + (i + 1) + " автомобиля: ");
                int number = scanner.nextInt();
                if (number == 666)
                    throw new InvalidCarException("Невалидный id");
                carList.add(new Car(number, "Toyota", "Camry", 2018, "Black",
25000.0, "AB1234"));
            }
            // Выводим список автомобилей заданной марки
            System.out.println("Список автомобилей марки Toyota:");
            ArrayList<Car> filteredByBrand = Car.filterByBrand(carList,
"Toyota");
            for (Car car : filteredByBrand) {
                System.out.println(car);
            }

            // Выводим список автомобилей заданной модели, которые
эксплуатируются больше n лет
            System.out.println("\nСписок автомобилей модели Civic, которые
эксплуатируются больше 5 лет:");
            ArrayList<Car> filteredByModelAndYears =
Car.filterByModelAndYears(carList, "Civic", 5);
            for (Car car : filteredByModelAndYears) {
                System.out.println(car);
            }

            // Выводим список автомобилей заданного года выпуска, цена
которых больше указанной
            System.out.println("\nСписок автомобилей 2018 года выпуска с
ценой выше 24000:");
            ArrayList<Car> filteredByYearAndPrice =
Car.filterByYearAndPrice(carList, 2018, 24000.0);
            for (Car car : filteredByYearAndPrice) {
                System.out.println(car);
            }
        } catch (InputMismatchException e) {
            System.out.println("Неверный тип данных");
        } catch (OutOfMemoryError e) {
            System.out.println("Оперативка исчерпана");
        } catch (InvalidCarException e) {
            System.out.println("InvalidPhoneException " +
e.getLocalizedMessage());
        } catch (Exception e) {
            System.out.println(e.getLocalizedMessage());
        }
    }
}
```

Задание №5

Формулировка задания и код программы представлены в листинге 5:

```java
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

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

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

        String inputFilePath = "src/input.txt";
        String outputFilePath = "src/output.txt";

        try (Scanner scanner = new Scanner(new File(inputFilePath));
             FileWriter writer = new FileWriter(outputFilePath)) {

            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                String[] words = line.split("\\s+");

                StringBuilder resultLine = new StringBuilder();
                for (String word : words) {
                    if (word.length() > 0) {
                        char firstChar =
Character.toLowerCase(word.charAt(0));

resultLine.append(firstChar).append(word.substring(1)).append(" ");
                    }
                }

                writer.write(resultLine.toString().trim() + "\n");
            }

            System.out.println("Преобразование завершено. Результат записан в
файл: " + outputFilePath);

        } catch (IOException e) {
            System.out.println("Ошибка при обработке файлов: " +
e.getMessage());
        }
    }
}
```

Задание №6

Формулировка задания и код программы представлены в листинге 6:

```java
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

/*
    Определить частоту повторяемости букв и слов в стихотворении Александра
Пушкина
 */
```

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

        String inputFilePath = "src/input.txt";
        String outputFilePath = "src/output.txt";

        try (Scanner scanner = new Scanner(new File(inputFilePath));
             FileWriter writer = new FileWriter(outputFilePath)) {

            Map<Character, Integer> charFrequency = new HashMap<>();
            Map<String, Integer> wordFrequency = new HashMap<>();

            while (scanner.hasNextLine()) {
                String line = scanner.nextLine().trim();

                // Подсчет частоты повторяемости букв
                for (char c : line.toCharArray()) {
                    charFrequency.put(c, charFrequency.getOrDefault(c, 0) +
1);
                }

                // Подсчет частоты повторяемости слов
                String[] words = line.split("\\s+");
                for (String word : words) {
                    wordFrequency.put(word, wordFrequency.getOrDefault(word,
0) + 1);
                }
            }

            writer.write("Частота повторяемости букв:\n");
            for (Map.Entry<Character, Integer> entry :
charFrequency.entrySet()) {
                writer.write(entry.getKey() + ": " + entry.getValue() +
"\n");
            }

            writer.write("\nЧастота повторяемости слов:\n");
            for (Map.Entry<String, Integer> entry : wordFrequency.entrySet())
{
                writer.write(entry.getKey() + ": " + entry.getValue() +
"\n");
            }

            System.out.println("Программа завершена. Результат записан в
файл: " + outputFilePath);

        } catch (IOException e) {
            System.out.println("Ошибка при обработке файлов: " +
e.getMessage());
        }
    }
}
```

Задание №7

Формулировка задания и код программы представлены в листинге 7:

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
```

```java
/*
    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. Для вывода результатов создавать новую
    директорию и файл средствами класса File
 */

public class Main {
    public static void main(String[] args) {
        try {
            File inputFile = new File("src/input.txt");
            Scanner scanner = new Scanner(inputFile);

            File outputDir = new File("output");
            outputDir.mkdir();
            File outputFile = new File(outputDir, "output.txt");
            FileWriter writer = new FileWriter(outputFile);

            while (scanner.hasNextLine()) {
                String line = scanner.nextLine().trim();
                String[] words = line.split("\\s+");

                int evenCount = 0;
                for (String word : words) {
                    if (word.length() >= 3 && word.length() <= 5) {
                        evenCount++;
                    }
                }

                int maxEvenCount = evenCount % 2 == 0 ? evenCount : evenCount
- 1;

                int removedCount = 0;
                for (String word : words) {
                    if (word.length() >= 3 && word.length() <= 5) {
                        if (removedCount < maxEvenCount) {
                            removedCount++;
                        } else {
                            writer.write(word + " ");
                        }
                    } else {
                        writer.write(word + " ");
                    }
                }
                writer.write("\n");
            }

            scanner.close();
            writer.close();

            System.out.println("Результат записан в файл: " +
outputFile.getAbsolutePath());
        } catch (IOException e) {
            System.out.println("Ошибка при обработке файлов: " +
e.getMessage());
        }
    }
}
```

Задание №8

Формулировка задания и код программы представлены в листинге 8:

```java
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

/*
    Прочитать строки из файла и поменять местами первое
    и последнее слова в каждой строке. Для вывода результатов создавать новую
    директорию и файл средствами класса File
 */

public class Main {
    public static void main(String[] args) {
        try {
            File inputFile = new File("src/input.txt");
            Scanner scanner = new Scanner(inputFile);

            File outputDir = new File("output");
            outputDir.mkdir();
            File outputFile = new File(outputDir, "output.txt");
            FileWriter writer = new FileWriter(outputFile);

            while (scanner.hasNextLine()) {
                String line = scanner.nextLine().trim();
                String[] words = line.split("\\s+");

                if (words.length > 1) {
                    String temp = words[0];
                    words[0] = words[words.length - 1];
                    words[words.length - 1] = temp;
                }

                for (String word : words) {
                    writer.write(word + " ");
                }
                writer.write("\n");
            }

            scanner.close();
            writer.close();

            System.out.println("Результат записан в файл: " +
outputFile.getAbsolutePath());
        } catch (IOException e) {
            System.out.println("Ошибка при обработке файлов: " +
e.getMessage());
        }
    }
}
```

Вывод: в результате выполнения лабораторной работы были освоены основные принципы исключений и файлов на языке Java.