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: Лабораторная работа №5

Дисциплина: Языки программирования для работы с большими данными

Студент	ИУ6-22М		М.И. Замула
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2023 г.

Вариант: 1

Выполнить задания на основе варианта 1 лабораторной работы 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т.д.

Задание 8

Выполнение

```
package lab5;

import lab3.Complex;

public class Lab5_1_8 {
    public static void main(String[] args) {
        int n = 5;
        Complex[] vector01 =  new Complex[n];
        Complex[] vector02 =  new Complex[n];

        try {
            for (int i = 0; i < n; i++) {
                vector01[i] = new Complex(i,i+1);
                vector02[i] = new Complex(i + 1, i);
            }
        } catch (Exception e) {
            System.err.println("Not enough memory!");
            e.printStackTrace();
        }
        Complex[] result = new Complex[n];
        for (int i = 0; i < n; i++) {
            result[i] = vector01[i].addition(vector02[i]);
        }
        System.out.println("Vector 01: ");
        for (int i = 0; i < n; i++) {
            System.out.println(vector01[i].toString());
        }
        System.out.println("Vector 02: ");
        for (int i = 0; i < n; i++) {
            System.out.println(vector02[i].toString());
        }
        System.out.println("Result: ");
        for (int i = 0; i < n; i++) {
            System.out.println(result[i].toString());
        }
    }
}
```

Результаты

```
Vector 01:
0.0 + 1.0i
1.0 + 2.0i
2.0 + 3.0i
3.0 + 4.0i
4.0 + 5.0i
Vector 02:
1.0 + 0.0i
2.0 + 1.0i
3.0 + 2.0i
4.0 + 3.0i
5.0 + 4.0i
Result:
1.0 + 1.0i
3.0 + 3.0i
5.0 + 5.0i
7.0 + 7.0i
9.0 + 9.0i
```

Задание 9

Выполнение

```java
package lab5;

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

public class Lab5_1_9 {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.print("Введите количество уравнений: ");
        int n;

        try {
            n = in.nextInt();
        } catch (InputMismatchException e) {
            System.out.println("Необходимо ввести число типа int");
            return;
        }
```

```
        long maxMemory = Runtime.getRuntime().maxMemory();
        long totalMemory = Runtime.getRuntime().totalMemory();
        long freeMemory = Runtime.getRuntime().freeMemory();

        if ((long) n * 16 > maxMemory - totalMemory + freeMemory) {
            System.out.println("Недостаточно памяти");
            return;
        }

        QuadraticEquation[] array;

        try {
            array = new QuadraticEquation[n];
        } catch (NegativeArraySizeException e) {
            System.out.println("Количество уравнений должно быть
положительным");
            return;
        }


        for (int i = 0; i < n; i++) {
            System.out.println("Уравнение #" + (i + 1));
            System.out.print("Введите коэффициент a: ");
            double a;
            try {
                a = in.nextDouble();
            } catch (InputMismatchException e) {
                System.out.println("Коэффициент a должен быть числом");
                return;
            }

            System.out.print("Введите коэффициент b: ");
            double b;
            try {
                b = in.nextDouble();
            } catch (InputMismatchException e) {
                System.out.println("Коэффициент b должен быть числом");
                return;
            }

            System.out.print("Введите коэффициент c: ");
            double c;
            try {
                c = in.nextDouble();
            } catch (InputMismatchException e) {
                System.out.println("Коэффициент c должен быть числом");
                return;
            }

            QuadraticEquation q = new QuadraticEquation(a, b, c);
            array[i] = q;

            double[] roots = q.findRoot();
            double maxRoot = Double.NEGATIVE_INFINITY;
            double minRoot = Double.POSITIVE_INFINITY;

            for (double root : roots) {
                if (root > maxRoot) {
                    maxRoot = root;
                }
                if (root < minRoot) {
                    minRoot = root;
                }
```

```
            }

            System.out.println("Максимальный корень = " +
String.format("%.4f", maxRoot));
            System.out.println("Минимальный корень = " +
String.format("%.4f", minRoot));
        }
    }
}
```

Результаты

```
Введите количество уравнений: 2
Уравнение #1
Введите коэффициент a: 4
Введите коэффициент b: -9
Введите коэффициент c: 2
Максимальный корень = 2,0000
Минимальный корень = 0,2500
Уравнение #2
Введите коэффициент a: 1
Введите коэффициент b: 0
Введите коэффициент c: -2
Максимальный корень = 1,4142
Минимальный корень = -1,4142
```

Вариант: 2

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

Задание 8

Выполнение

```
package lab5;

import lab3.Car;

import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.List;

public class Lab5_2_8 {
    public static void main(String[] args) {

        Car[] cars = new Car[]{
                new Car(1, "Ford", "Galaxy", 2010, "Blue", 1.6, "A123BC77"),
                new Car(2, "Renault", "Logan", 2010, "Red", 0.3, "E456HK77"),
                new Car(3, "Toyota", "Camry", 2008, "Black", 1.2,
"M007AB77"),
                new Car(4, "Ford", "Focus", 2015, "Grey", 1.3, "C223OT77"),
                new Car(5, "Ford", "Explorer", 2021, "Black", 3.5,
"A777AA77"),
                new Car(6, "Toyota", "Camry", 2010, "White", 1.7,
"P482CX77"),
                new Car(7, "Toyota", "Camry", 2010, "Green", 1.9,
"K911EC77"),
        };

        List<Car> fords;
        try {
            fords = listMarks(cars, "Ford");
            System.out.println("Автомобили марки Ford: ");
            for (Car ford : fords) {
                System.out.println(ford);
            }

System.out.println("**********************************************************
*********");
        } catch (IllegalArgumentException e) {
            System.out.println("Ошибка: " + e.getMessage());
        }

        List<Car> camry2010;
        try {
            camry2010 = listModelAndYear(cars, "Camry", 13);
            System.out.println("Автомобили модели Camry, младше 13 лет: ");
            for (Car car : camry2010) {
                System.out.println(car);
            }

System.out.println("**********************************************************
```

```
*********");
        } catch (IllegalArgumentException e) {
            System.out.println("Ошибка: " + e.getMessage());
        }

        List<Car> expensiveCars;
        try {
            expensiveCars = listYearAndPrice(cars, 2010, 1.5);
            System.out.println("Автомобили 2010 года, дороже 1.5 млн: ");
            for (Car car : expensiveCars) {
                System.out.println(car);
            }
        } catch (InputMismatchException | IllegalArgumentException e) {
            System.out.println("Ошибка: " + e.getMessage());
        }
    }


    public static List<Car> listMarks(Car[] cars, String mark){
        List<Car> result = new ArrayList<>();
        for (Car car : cars) {
            if (car.isMark(mark)) {
                result.add(car);
            }
        }
        return result;
    }

    public static List<Car> listModelAndYear(Car[] cars, String model, int
nYear){
        List<Car> result = new ArrayList<>();
        for (Car car : cars) {
            if (car.isModelAndNYear(model, nYear)) {
                result.add(car);
            }
        }
        return result;
    }

    public static List<Car> listYearAndPrice(Car[] cars, int year, double
price){
        List<Car> result = new ArrayList<>();
        for (Car car : cars) {
            if (car.isYearAndPrice(year, price)) {
                result.add(car);
            }
        }
        return result;
    }
}
```

Результат

```
Автомобили марки Ford:
Автомобиль: id=1, mark='Ford', model='Galaxy', year=2010, color='Blue', price=1.6, number=A123BC77
Автомобиль: id=4, mark='Ford', model='Focus', year=2015, color='Grey', price=1.3, number=C223OT77
Автомобиль: id=5, mark='Ford', model='Explorer', year=2021, color='Black', price=3.5, number=A777AA77
************************************************************
Автомобили модели Camry, младше 13 лет:
Автомобиль: id=6, mark='Toyota', model='Camry', year=2010, color='White', price=1.7, number=P482CX77
Автомобиль: id=7, mark='Toyota', model='Camry', year=2010, color='Green', price=1.9, number=K911EC77
************************************************************
Автомобили 2010 года, дороже 1.5 млн:
Автомобиль: id=1, mark='Ford', model='Galaxy', year=2010, color='Blue', price=1.6, number=A123BC77
Автомобиль: id=6, mark='Toyota', model='Camry', year=2010, color='White', price=1.7, number=P482CX77
Автомобиль: id=7, mark='Toyota', model='Camry', year=2010, color='Green', price=1.9, number=K911EC77
```

Задание 9

Выполнение

```java
package lab5;

import lab3.Product;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class Lab5_2_9 {
    public static void main(String[] args) {
        Product[] products = new Product[]{
                new Product(1, "lipstick", "12345600005", "MAC", 1.7, 1.5,
148),
                new Product(2, "mascara", "21735600006", "INGLOT", 2.2, 1.0,
11),
                new Product(3, "blush", "086320600007", "MAC", 1.9, 3.3,
221),
                new Product(4, "cream", "002638600003", "PAYOT", 6.0, 1.4,
93),
                new Product(5, "shadow", "015338600009", "MAC", 4.5, 2.5,
160),
                new Product(6, "lipstick", "008437200003", "SCINIC", 0.7,
1.5, 20),
                new Product(7, "lipstick", "076649900004", "MAC", 1.4, 2.0,
45),
        };

        List<Product> lipsticks;
        List<Product> cheapLipsticks;
        List<Product> shelfLifeProducts;

        try {
            lipsticks = listName(products, "lipstick");
            System.out.println("Lipsticks: ");
            for (Product lipstick : lipsticks) {
                System.out.println(lipstick);
            }

System.out.println("************************************************************
*********");
```

```
            cheapLipsticks = listNameAndPrice(products, "lipstick", 1.5);
            System.out.println("Lipsticks cheaper 1.5: ");
            for (Product product : cheapLipsticks) {
                System.out.println(product);
            }

System.out.println("**********************************************************
*********");

            shelfLifeProducts = listShelfLife(products, 1.6);
            System.out.println("Products with a shelf life of more than 1.6
years: ");
            for (Product product : shelfLifeProducts) {
                System.out.println(product);
            }
        } catch (Exception e) {
            System.out.println("Unexpected error: " + e.getMessage());
        }

    }

    public static List<Product> listName(Product[] products, String name)
throws IOException {
        List<Product> result = new ArrayList<>();
        for (Product product : products) {
            if (product.isName(name)) {
                result.add(product);
            }
        }
        if (result.isEmpty()) {
            throw new IOException("No products with this name were found.");
        }
        return result;
    }
    public static List<Product> listNameAndPrice(Product[] products, String
name, double price) throws IOException {
        List<Product> result = new ArrayList<>();
        for (Product product : products) {
            if (product.isNameAndPrice(name, price)) {
                result.add(product);
            }
        }
        if (result.isEmpty()) {
            throw new IOException("No products with this name and price were
found.");
        }
        return result;
    }

    public static List<Product> listShelfLife(Product[] products, double
shelfLife) throws IOException {
        List<Product> result = new ArrayList<>();
        for (Product product : products) {
            if (product.isShelfLife(shelfLife)) {
                result.add(product);
            }
        }
        if (result.isEmpty()) {
            throw new IOException("No products with this name and price were
found.");
        }
        return result;
```

```
    }

}
```

Результаты

```
Lipsticks:
Product: id=1, name='lipstick', upc=12345600005, manufacturer='MAC', price=1.7, shelfLife=1.5, quantify=148}
Product: id=6, name='lipstick', upc=008437200003, manufacturer='SCINIC', price=0.7, shelfLife=1.5, quantify=20}
Product: id=7, name='lipstick', upc=076649900004, manufacturer='MAC', price=1.4, shelfLife=2.0, quantify=45}
**************************************************************
Lipsticks cheaper 1.5:
Product: id=6, name='lipstick', upc=008437200003, manufacturer='SCINIC', price=0.7, shelfLife=1.5, quantify=20}
Product: id=7, name='lipstick', upc=076649900004, manufacturer='MAC', price=1.4, shelfLife=2.0, quantify=45}
**************************************************************
Products with a shelf life of more than 1.6 years:
Product: id=3, name='blush', upc=086320600007, manufacturer='MAC', price=1.9, shelfLife=3.3, quantify=221}
Product: id=5, name='shadow', upc=015338600009, manufacturer='MAC', price=4.5, shelfLife=2.5, quantify=160}
Product: id=7, name='lipstick', upc=076649900004, manufacturer='MAC', price=1.4, shelfLife=2.0, quantify=45}
```

Вариант: 3

В следующих заданиях требуется ввести последовательность строк из текстового потока и выполнить указанные действия. При этом могут рассматриваться два варианта:

- каждая строка состоит из одного слова;
- каждая строка состоит из нескольких слов.

Имена входного и выходного файлов, а также абсолютный путь к ним могут быть введены как параметры командной строки или храниться в файле.

Задание 8

Определить частоту повторяемости букв и слов в стихотворении Александра Пушкина.

Выполнение

```
package lab5;

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

```java
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class Lab5_3_8 {

    public static void main(String[] args) {
        String inputFilePath = "poem.txt";
        String outputFilePath = "result.txt";

        // Создаем мапы для подсчета частоты повторяемости букв и слов
        Map<Character, Integer> charFrequencies = new HashMap<>();
        Map<String, Integer> wordFrequencies = new HashMap<>();

        try (BufferedReader br = new BufferedReader(new
FileReader(inputFilePath))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] words = line.split("\\s+");

                // Подсчитываем частоту повторяемости букв
                for (char c : line.toCharArray()) {
                    if (charFrequencies.containsKey(c)) {
                        charFrequencies.put(c, charFrequencies.get(c) + 1);
                    } else {
                        charFrequencies.put(c, 1);
                    }
                }

                // Подсчитываем частоту повторяемости слов
                for (String word : words) {
                    if (wordFrequencies.containsKey(word)) {
                        wordFrequencies.put(word, wordFrequencies.get(word) +
1);
                    } else {
                        wordFrequencies.put(word, 1);
                    }
                }
            }
        } catch (IOException e) {
            System.err.format("Ошибка чтения файла: %s%n", e);
        }

        // Записываем результат в файл
        try (BufferedWriter bw = new BufferedWriter(new
FileWriter(outputFilePath))) {
            bw.write("Частота повторяемости букв:\n");
            for (Map.Entry<Character, Integer> entry :
charFrequencies.entrySet()) {
                bw.write(entry.getKey() + ": " + entry.getValue() + "\n");
            }
            bw.write("\nЧастота повторяемости слов:\n");
            for (Map.Entry<String, Integer> entry :
wordFrequencies.entrySet()) {
                bw.write(entry.getKey() + ": " + entry.getValue() + "\n");
            }
        } catch (IOException e) {
            System.err.format("Ошибка записи файла: %s%n", e);
        }

        System.out.println("Результат записан в файл " + outputFilePath);
    }
```

```
}
```

Poem.txt

```
Духовной жаждою томим,
В пустыне мрачной я влачился, —
И шестикрылый серафим
На перепутье мне явился.
Перстами легкими как сон
Моих зениц коснулся он.
Отверзлись вещие зеницы,
Как у испуганной орлицы.
Моих ушей коснулся он, —
И их наполнил шум и звон:
И внял я неба содроганье,
И горний ангелов полет,
И гад морских подводный ход,
И дольней лозы прозябанье.
И он к устам моим приник,
И вырвал грешный мой язык,
И празднословный и лукавый,
И жало мудрыя змеи
В уста замершие мои
Вложил десницею кровавой.
И он мне грудь рассек мечом,
И сердце трепетное вынул,
И угль, пылающий огнем,
Во грудь отверстую водвинул.
Как труп в пустыне я лежал,
И бога глас ко мне воззвал:
«Восстань, пророк, и виждь, и внемли,
Исполнись волею моей,
И, обходя моря и земли,
Глаголом жги сердца людей».
```

Result.txt

```
Частота повторяемости букв:
р: 29
с: 30
т: 16
у: 21
ф: 1
х: 7
ц: 6
ч: 3
ш: 5
щ: 2
ы: 16
ь: 11
ю: 6
я: 14
В: 5
```

```
Г: 1
Д: 1
—: 2
И: 16
К: 2
М: 2
Н: 1
О: 1
П: 1
 : 104
«: 1
,: 22
.: 7
а: 33
б: 4
в: 25
г: 15
д: 18
е: 46
ж: 7
з: 13
и: 40
й: 16
к: 17
:: 2
л: 37
»: 1
м: 27
н: 41
о: 59
п: 16

Частота повторяемости слов:
водвинул.: 1
жги: 1
прозябанье.: 1
уста: 1
их: 1
шестикрылый: 1
морских: 1
Вложил: 1
В: 2
земли,: 1
—: 2
И,: 1
И: 14
ангелов: 1
сердце: 1
сердца: 1
легкими: 1
пылающий: 1
замершие: 1
десницею: 1
рассек: 1
зениц: 1
в: 1
Перстами: 1
и: 5
к: 1
мудрыя: 1
у: 1
ко: 1
язык,: 1
```

```
неба: 1
жаждою: 1
отверстую: 1
я: 3
перепутье: 1
пророк,: 1
внемли,: 1
трепетное: 1
он,: 1
он.: 1
вынул,: 1
вырвал: 1
огнем,: 1
«Восстань,: 1
гад: 1
жало: 1
обходя: 1
лозы: 1
Во: 1
волею: 1
влачился,: 1
глас: 1
зеницы,: 1
испуганной: 1
коснулся: 2
наполнил: 1
угль,: 1
мечом,: 1
Духовной: 1
горний: 1
приник,: 1
Глаголом: 1
Исполнись: 1
грудь: 2
Отверзлись: 1
Как: 2
подводный: 1
внял: 1
шум: 1
воззвал:: 1
змеи: 1
бога: 1
На: 1
людей».: 1
лежал,: 1
содроганье,: 1
мрачной: 1
устам: 1
он: 2
пустыне: 2
Моих: 2
мне: 3
грешный: 1
виждь,: 1
дольней: 1
ушей: 1
полет,: 1
кровавой.: 1
сон: 1
ход,: 1
как: 1
лукавый,: 1
серафим: 1
```

```
празднословный: 1
явился.: 1
труп: 1
мои: 1
мой: 1
орлицы.: 1
моря: 1
томим,: 1
моей,: 1
вещие: 1
моим: 1
звон:: 1
```

Задание 9

Входной файл содержит совокупность строк. Строка файла содержит строку квадратной матрицы. Ввести матрицу в двумерный массив (размер матрицы найти). Вывести исходную матрицу и результат ее транспонирования.

Выполнение

```java
package lab5;

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

public class Lab5_3_9 {
    public static void main(String[] args) {
        String inputFilePath = "matrix.txt";
        String outputFilePath = "result02.txt";

        // Check if input and output file paths were provided as command line
arguments
        if (args.length >= 2) {
            inputFilePath = args[0];
            outputFilePath = args[1];
        }

        // Read the input matrix from file
        int[][] matrix = readMatrixFromFile(inputFilePath);

        // Print the original matrix
        System.out.println("Original matrix:");
        printMatrix(matrix);

        // Rotate the matrix
        int[][] rotatedMatrix = rotateMatrix(matrix);

        // Print the rotated matrix
        System.out.println("Rotated matrix:");
        printMatrix(rotatedMatrix);
```

```
        // Write the rotated matrix to file
        writeMatrixToFile(rotatedMatrix, outputFilePath);
        System.out.println("Rotated matrix saved to " + outputFilePath);
    }

    // Reads a matrix from a file and returns it as a 2D array of integers
    private static int[][] readMatrixFromFile(String filePath) {
        int[][] matrix = null;

        try (BufferedReader reader = new BufferedReader(new
FileReader(filePath))) {
            // Determine the size of the matrix (assuming it is square)
            String line = reader.readLine();
            int size = line.split("\\s+").length;

            // Initialize the matrix
            matrix = new int[size][size];
            int row = 0;

            // Read the matrix from the file
            do {
                String[] values = line.split("\\s+");
                for (int col = 0; col < size; col++) {
                    matrix[row][col] = Integer.parseInt(values[col]);
                }
                row++;
            } while ((line = reader.readLine()) != null);

        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
            System.exit(1);
        }

        return matrix;
    }

    // Rotates a matrix by 90 degrees clockwise and returns the result as a
new 2D array
    private static int[][] rotateMatrix(int[][] matrix) {
        int size = matrix.length;
        int[][] rotatedMatrix = new int[size][size];

        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                rotatedMatrix[j][size - i - 1] = matrix[i][j];
            }
        }

        return rotatedMatrix;
    }

    // Prints a matrix to the console
    private static void printMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int value : row) {
                System.out.print(value + " ");
            }
            System.out.println();
        }
    }

    // Writes a matrix to a file
```

```
    private static void writeMatrixToFile(int[][] matrix, String filePath) {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(filePath))) {
            for (int[] row : matrix) {
                for (int value : row) {
                    writer.write(value + " ");
                }
                writer.newLine();
            }
        } catch (IOException e) {
            System.err.println("Error writing file: " + e.getMessage());
            System.exit(1);
        }
    }
}
```

Matrix.txt

```
12 35 88 4 1 7 21 19 45 2
6 9 11 26 75 62 13 15 31 50
57 22 42 90 17 23 68 72 84 98
44 33 29 16 99 81 70 54 40 25
51 77 67 92 46 30 37 95 61 83
41 55 8 64 28 38 58 65 79 97
10 80 78 89 94 47 60 3 24 69
76 91 14 85 87 71 34 63 36 5
27 52 66 96 32 48 18 43 73 53
39 56 49 20 86 59 82 93 74 100
```

Result02.txt

```
39 27 76 10 41 51 44 57 6 12
56 52 91 80 55 77 33 22 9 35
49 66 14 78 8 67 29 42 11 88
20 96 85 89 64 92 16 90 26 4
86 32 87 94 28 46 99 17 75 1
59 48 71 47 38 30 81 23 62 7
82 18 34 60 58 37 70 68 13 21
93 43 63 3 65 95 54 72 15 19
74 73 36 24 79 61 40 84 31 45
100 53 5 69 97 83 25 98 50 2
```

Вариант: 4

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

Задание 8

Из текста Java-программы удалить все виды комментариев.

Выполнение

```
package lab5;

import java.io.*;

public class Lab5_4_8 {
    public static void main(String[] args) {
        File inputFile = new File("input.java");
        File outputFile = new File(inputFile.getParent(), "output.java");

        try (BufferedReader reader = new BufferedReader(new
FileReader(inputFile));
             BufferedWriter writer = new BufferedWriter(new
FileWriter(outputFile))) {

            String line;
            boolean isMultiLineComment = false;

            while ((line = reader.readLine()) != null) {
                line = line.trim();

                // Check for multi-line comments
                if (isMultiLineComment) {
                    int endIndex = line.indexOf("*/");

                    if (endIndex != -1) {
                        // End of multi-line comment found
                        line = line.substring(endIndex + 2);
                        isMultiLineComment = false;
                    } else {
                        // Still inside multi-line comment
                        continue;
                    }
                }

                // Check for single-line comments
                int commentIndex = line.indexOf("//");

                if (commentIndex != -1) {
                    // Single-line comment found
                    line = line.substring(0, commentIndex);
                } else {
                    // Check for start of multi-line comment
                    commentIndex = line.indexOf("/*");

                    if (commentIndex != -1) {
                        // Start of multi-line comment found
                        int endIndex = line.indexOf("*/", commentIndex);

                        if (endIndex != -1) {
                            // End of multi-line comment found on the same
line
                            line = line.substring(0, commentIndex) +
line.substring(endIndex + 2);
                        } else {
                            // End of multi-line comment not found on the
same line
                            line = line.substring(0, commentIndex);
```

```
                            isMultiLineComment = true;
                        }
                    }
                }

                // Write the line to the output file
                writer.write(line);
                writer.newLine();
            }

            System.out.println("Comments removed successfully!");
        } catch (IOException e) {
            System.out.println("An error occurred while removing comments: "
+ e.getMessage());
        }
    }
}
```

Input.txt

```
public class Main {
    /*
     * Это многострочный комментарий.
     * Здесь может быть любой текст.
     */
    public static void main(String[] args) {
        // Это однострочный комментарий
        System.out.println("Hello, World!"); // Это тоже однострочный
комментарий
    }
}
```

Output.txt

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Задание 9

Прочитать строки из файла и поменять местами первое и последнее слова в каждой строке.

Выполнение

```
package lab5;

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

public class Lab5_4_9 {
    public static void main(String[] args) {
        File inputFile = new File("input_5_4_9.txt");
        File outputFile = new File("output_5_4_9.txt");

        try (BufferedReader br = new BufferedReader(new
FileReader(inputFile));
             BufferedWriter bw = new BufferedWriter(new
FileWriter(outputFile))) {

            String line;
            while ((line = br.readLine()) != null) {
                String[] words = line.split(" ");
                String first = words[0];
                String last = words[words.length - 1];

                words[0] = last;
                words[words.length - 1] = first;

                String newLine = String.join(" ", words);
                bw.write(newLine);
                bw.newLine();
            }
        } catch (IOException e) {
            System.out.println("Error reading/writing file");
            e.printStackTrace();
        }

    }
}
```

Input_5_4_9.txt

```
This is the first line
The second line is here
And finally, the third line
```

Output_5_4_9.txt

```
line is the first This
here second line is The
line finally, the third And
```