**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

Факультет инфокоммуникационных сетей и систем (ИКСС)

Лабораторная работа №4

по курсу

«Разработка Java-приложений управления телекоммуникациями»

Группа: ИКПИ-11

Выполнил студент: Хохлов Т. В.

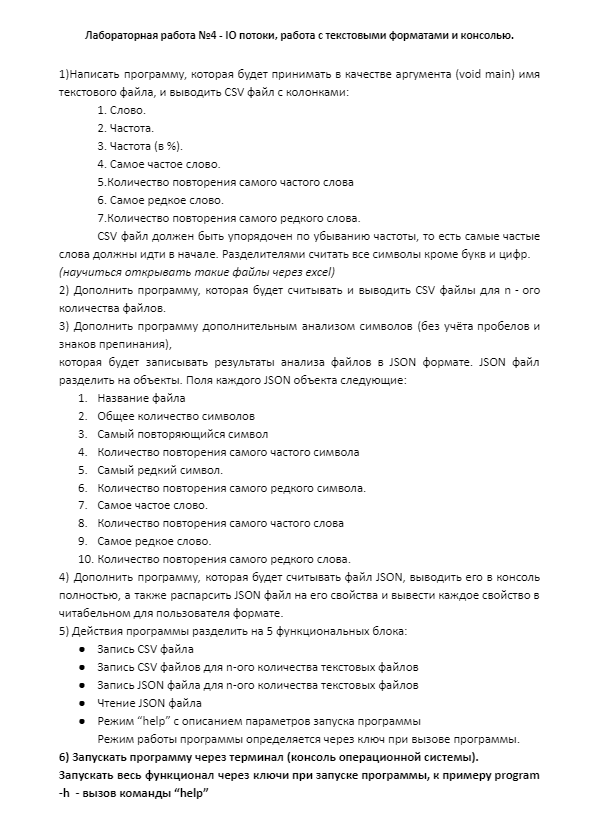
Принял преподаватель: Березин А.Ю.

Место для подписи \_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

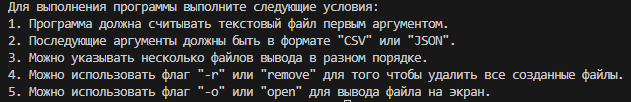
2023 г.

# **Цель работы**



**Описание кода**

Программа запускается из терминала. Весь функционал программы запускается через ключи. Список ключей и функционала выводится на экран при запуске программы с ключом «-h» или «help»:



static float makeHashMaps(String pathTextFile, int amountOfWords) throws IOException – функция создающая HashMap из заданного текстового файла.

static String getMinKeyWord(HashMap<String, Float> map) – функция, которая перебирает HashMap и находит слово с минимальным количеством повторений

static String getMaxKeyWord(HashMap<String, Float> map) - функция, которая перебирает HashMap и находит слово с максимальным количеством повторений

static Character getMinKeyChar(HashMap<Character, Float> map) - функция, которая перебирает HashMap и находит символ с минимальным количеством повторений

static Character getMaxKeyChar(HashMap<Character, Float> map) - функция, которая перебирает HashMap и находит символ с максимальным количеством повторений

static LinkedHashMap<String, Float> makeSortedMap() – функция, создающая сортированную HashMap по убыванию.

static void makeCsvFile(LinkedHashMap<String, Float> sortedMap, int amountOfWords, String File) throws IOException – функция, создающая CSV-файлы.

static void makeJsonFile(LinkedHashMap<String, Float> sortedMap, int amountOfWords, String filename) throws IOException – функция создающая json-файлы.

# **Листинг**

## **Lab4.java**

package Java.lab4;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.Collections;

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.concurrent.TimeUnit;

// CSV -> 1, 2, 3, 4, 5, 6, 7

//word, frequency, frequency%, mostFrequentWord, MostFrequentWordAmount, MostRareWord, MostRareWordAmount

public class lab4 {

    public static HashMap<Character, Float> mapChars;

    public static HashMap<String, Float> mapWords;

    public static float amountOfChars;

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

        if (args.length == 0) {

            System.out.println("Введите минимум один аргумент!");

            return;

        }

        if (args.length > 0 && args[0].indexOf(".txt") != -1) {

            float amountOfWords = 0;

            lab4.mapWords = new HashMap<String, Float>();

            lab4.mapChars = new HashMap<Character, Float>();

            amountOfWords = makeHashMaps(args[0], (int)amountOfWords);

            LinkedHashMap<String, Float> sortedMap = makeSortedMap();

            for (String filename: args) {

                if (filename.indexOf(".txt") == -1) {

                    if (filename.indexOf(".csv") != -1) {

                        makeCsvFile(sortedMap, (int)amountOfWords, filename);

                    }

                    if (filename.indexOf(".json")!= -1) {

                        makeJsonFile(sortedMap, (int)amountOfWords, filename);

                    }

                }

            }

        } else if (args[0].equals("-h") || args[0].equals("help")) {

            System.out.println("Для выполнения программы выполните следующие условия:");

            System.out.println("1. Программа должна считывать текстовый файл первым аргументом.");

            System.out.println("2. Последующие аргументы должны быть в формате \"CSV\" или \"JSON\".");

            System.out.println("3. Можно указывать несколько файлов вывода в разном порядке.");

            System.out.println("4. Можно использовать флаг \"-r\" или \"remove\" для того чтобы удалить все созданные файлы.");

            System.out.println("5. Можно использовать флаг \"-o\" или \"open\" для вывода файла на экран.");

            return;

        } else if (args[0].equals("-r") || args[0].equals("remove")) {

            File file = new File("");

            File folder = new File(file.getAbsolutePath());

            File fList[] = folder.listFiles();

            for (int i = 0; i < fList.length; i++) {

                File pes = fList[i];

                if (pes.getName().endsWith(".csv") || pes.getName().endsWith(".json")) {

                    fList[i].delete();

                }

            }

            System.out.println("Работа программы успешно завершена. Файлы удалены.");

            return;

        } else if (args[0].equals("-o") || args[0].equals("open")) {

            if (args[1] == null) {

                System.out.println("Введите название файла для открытия.");

                return;

            }

            File file = new File(args[1]);

            if (file.exists()) {

                try (BufferedReader br = new BufferedReader(new FileReader(args[1]))) {

                    String line;

                    while ((line = br.readLine()) != null) {

                        System.out.println(line);

                    }

                } catch (FileNotFoundException e) {

                    System.out.println("Данного файла не существует.");

                    return;

                } catch (IOException e) {

                    e.getMessage();

                }

            }

        } else {

            System.out.println(args[0]);

            System.out.println("Неверные аргументы.");

            return;

        }

        System.out.println("Работа программы успешно завершена.");

        return;

    }

    static float makeHashMaps(String pathTextFile, int amountOfWords) throws IOException {

        StringBuilder word = new StringBuilder();

        String text = new String ((Files.readAllBytes(Paths.get(pathTextFile))), "UTF-8");

        for (int i = 0; i < text.length(); i++) {

            lab4.amountOfChars++;

            // if (text.charAt(i) > 47 && text.charAt(i) < 58 || text.charAt(i) > 64 && text.charAt(i) < 91 || text.charAt(i) > 96 && text.charAt(i) < 123 ||

            // text.charAt(i) > 1039 && text.charAt(i) < 1104) {

            if (Character.isLetterOrDigit(text.charAt(i))) {

                if (lab4.mapChars.containsKey(text.charAt(i))) {

                    lab4.mapChars.put(text.charAt(i), lab4.mapChars.get(text.charAt(i)) + 1);

                } else {

                    lab4.mapChars.put(text.charAt(i), 1f);

                }

                word.append(text.charAt(i));

                if (i == text.length() - 1) {

                    if (lab4.mapWords.containsKey(word.toString())) {

                    lab4.mapWords.put(word.toString(), lab4.mapWords.get(word.toString()) + 1);

                    } else {

                        lab4.mapWords.put(word.toString(), 1f);

                    }

                    amountOfWords++;

                }

            } else if (word.length() > 0) {

                if (lab4.mapWords.containsKey(word.toString())) {

                    lab4.mapWords.put(word.toString(), lab4.mapWords.get(word.toString()) + 1);

                } else {

                    lab4.mapWords.put(word.toString(), 1f);

                }

                word = new StringBuilder();

                amountOfWords++;

            }

        }

        return amountOfWords;

    }

    static String getMinKeyWord(HashMap<String, Float> map) {

        String minKey = null;

        for (java.util.Map.Entry<String, Float> entryLocal : map.entrySet()) {

            if (Collections.min(map.values()) == entryLocal.getValue()) {

                minKey = entryLocal.getKey();

            }

        }

        return minKey;

    }

    static String getMaxKeyWord(HashMap<String, Float> map) {

        String maxKey = null;

        for (java.util.Map.Entry<String, Float> entryLocal : map.entrySet()) {

            if (Collections.max(map.values()) == entryLocal.getValue()) {

                maxKey = entryLocal.getKey();

            }

        }

        return maxKey;

    }

    static Character getMinKeyChar(HashMap<Character, Float> map) {

        Character minKey = null;

        for (java.util.Map.Entry<Character, Float> entryLocal : map.entrySet()) {

            if (Collections.min(map.values()) == entryLocal.getValue()) {

                minKey = entryLocal.getKey();

            }

        }

        return minKey;

    }

    static Character getMaxKeyChar(HashMap<Character, Float> map) {

        Character maxKey = null;

        for (java.util.Map.Entry<Character, Float> entryLocal : map.entrySet()) {

            if (Collections.max(map.values()) == entryLocal.getValue()) {

                maxKey = entryLocal.getKey();

            }

        }

        return maxKey;

    }

    static LinkedHashMap<String, Float> makeSortedMap() {

        ArrayList<Float> list = new ArrayList<>();

        LinkedHashMap<String, Float> sortedMap = new LinkedHashMap<>();

        for (Map.Entry<String, Float> entry : lab4.mapWords.entrySet()) {

            list.add(entry.getValue());

        }

        Collections.sort(list);

        for (int i = list.size()-1; i > 0; i--) {

            for (java.util.Map.Entry<String, Float> entry : lab4.mapWords.entrySet()) {

                if (entry.getValue().equals(list.get(i))) {

                    sortedMap.put(entry.getKey(), list.get(i));

                }

            }

        }

        return sortedMap;

    }

    static void makeCsvFile(LinkedHashMap<String, Float> sortedMap, int amountOfWords, String File) throws IOException {

        BufferedWriter csvWriter = new BufferedWriter(new FileWriter(File));

        csvWriter.write("word;frequency;frequency%;mostFrequentWord;MostFrequentWordAmount;MostRareWord;MostRareWordAmount;\n");

        boolean flag = true;

        for (String key : sortedMap.keySet()) {

            if (flag) {

                csvWriter.write(key + ";" + sortedMap.get(key) + ";" + sortedMap.get(key) \* 100 / amountOfWords + ";" + getMaxKeyWord(sortedMap) + ";" +

                Collections.max(sortedMap.values()) + ";" + getMinKeyWord(sortedMap) + ";" + Collections.min(sortedMap.values()) + "\n");

                flag = false;

                continue;

            }

            csvWriter.write(key + ";" + sortedMap.get(key) + ";" + sortedMap.get(key) \* 100 / amountOfWords + "\n");

        }

        csvWriter.close();

        return;

    }

    static void makeJsonFile(LinkedHashMap<String, Float> sortedMap, int amountOfWords, String filename) throws IOException {

        BufferedWriter jsonWriter = new BufferedWriter(new FileWriter(filename));

        String tab = "    ";

        jsonWriter.write("{\n");

        jsonWriter.write(tab + "\"name\": " + "\"" + filename + "\",\n");

        jsonWriter.write(tab + "\"amountOfChars\": " + "\"" + lab4.amountOfChars + "\",\n");

        jsonWriter.write(tab + "\"mostRepeatedChar\": " + "{\n");

        jsonWriter.write(tab + tab + "\"char\": " + "\"" + getMaxKeyChar(mapChars) + "\",\n");

        jsonWriter.write(tab + tab + "\"amount\": " + Collections.max(mapChars.values()) + "\n");

        jsonWriter.write(tab + tab + "},\n");

        jsonWriter.write(tab + "\"rarestChar\": " + "{\n");

        jsonWriter.write(tab + tab + "\"char\": " + "\"" + getMinKeyChar(mapChars) + "\",\n");

        jsonWriter.write(tab + tab + "\"amount\": " + Collections.min(mapChars.values()) + "\n");

        jsonWriter.write(tab + tab + "},\n");

        jsonWriter.write(tab + "\"mostFrequentWord\": " + "{\n");

        jsonWriter.write(tab + tab + "\"word\": " + "\"" + getMaxKeyWord(sortedMap) + "\",\n");

        jsonWriter.write(tab + tab + "\"amount\": " + Collections.max(sortedMap.values()) + "\n");

        jsonWriter.write(tab + tab + "},\n");

        jsonWriter.write(tab + "\"rarestWord\": " + "{\n");

        jsonWriter.write(tab + tab + "\"word\": " + "\"" + getMinKeyWord(sortedMap) + "\",\n");

        jsonWriter.write(tab + tab + "\"amount\": " + Collections.min(sortedMap.values()) + "\n");

        jsonWriter.write(tab + tab + "}\n");

        jsonWriter.write("}");

        jsonWriter.close();

        return;

    }

}

# **Выводы**

В результате выполнения лабораторной работы была написана программа на языке Java, которая включает в себя несколько задач; изучена структура csv и json файлов; изучены такие коллекции как HashSet и HashMap.