Федеральное агентство связи ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М. А. БОНЧ-БРУЕВИЧА» (СПбГУТ)

Факультет инфокоммуникационных сетей и систем

Кафедра программной инженерии и вычислительной техники

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине «Разработка Java-приложений управления телекоммуникациями»

Выполнил: студент 3-го курса дневного отделения группы ИКПИ-05

Кузнецов Максим Сергеевич

Преподаватель: Березин А. Ю.

Санкт-Петербург

2022

РЕФЕРАТ

Отчет 11 с., 1 кн., 3 рис., 50 источн., 2 прил.

Объектом разработки является программа, написанная на языке Java.

Цель работы - Написать программу, которая будет принимать в качестве аргумента имя текстового файла, и выводить CSV файл с колонками:

1. Слово.

2. Частота.

3. Частота (в %).

CSV файл должен быть упорядочен по убыванию частоты, то есть самые частые слова должны идти в начале. Разделителями считать все символы кроме букв и цифр.

В процессе работы создан класс, принимающий на вход текстовой файл, который проходит обработку. После обработки создается csv файл, в который записываются слова, встретившиеся в текстовом файле, а также частота появления этих слов.

СОДЕРЖАНИЕ

Оглавление

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 5](#_Toc117426592)

[ВВЕДЕНИЕ 6](#_Toc117426593)

[ОСНОВНАЯ ЧАСТЬ 7](#_Toc117426594)

[ЗАКЛЮЧЕНИЕ 9](#_Toc117426595)

[ПРИЛОЖЕНИЕ 10](#_Toc117426596)

# ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящем отчете применяются следующие термины и определения:

Java – используемый для разработки язык программирования.

Консольное приложение – компьютерная программа, предназначенная для использования через текстовый компьютерный интерфейс, такой как текстовый терминал, интерфейс командной строки.

# ВВЕДЕНИЕ

На языке Java написан класс позволяющий обработать текстовой файл и подсчитать частоту появления слов в этом файле. Основной класс называется CSVClass, в нем содержатся методы чтения текстового файла (readFileWords) и создания текстового файла(writeFileWords). Найденные слова хранятся в хеш-словаре mapWords, слова на вывод хранятся в списке listWords.

# ОСНОВНАЯ ЧАСТЬ

Для проверки работоспособности класса создан текстовой файл формата txt с произвольным текстом. Класс CSVClass получает на вход имя файла и осуществляет посимвольное чтение до конца слова, затем слово записывается в хеш-словарь с помощью метода addMap. После чтения всех слов вызывается метод sortedMap для сортировки хеш-таблицы, отсортированные данные попадают в список listWords и выводятся в csv файл. Содержимое csv файла можно увидеть на Рисунке 1.

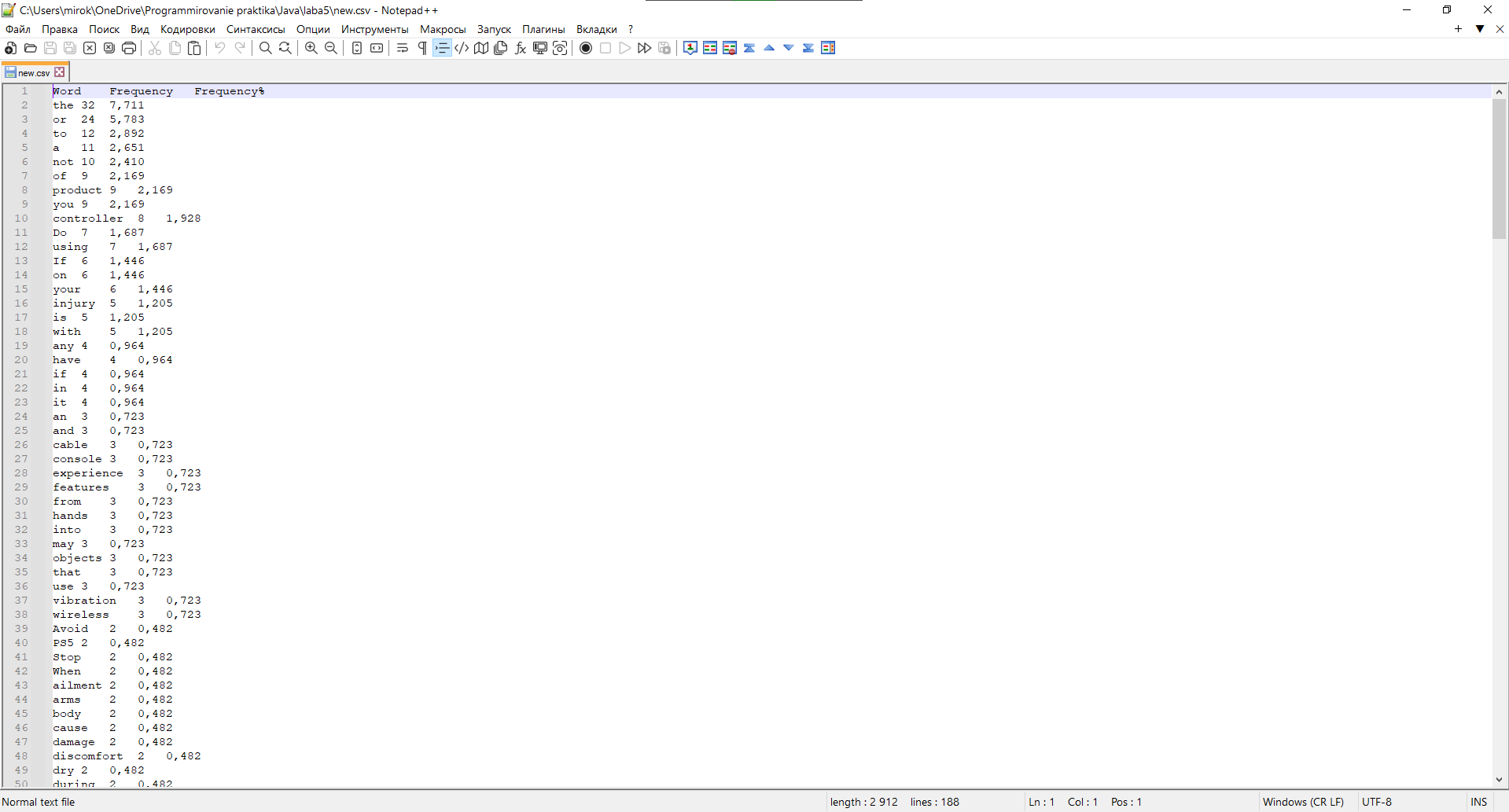


Рисунок 1 - Содержимое csv файла

Для удобства чтения и последующего редактирования csv файл можно импортировать в Microsoft Excel (Рисунок 2).

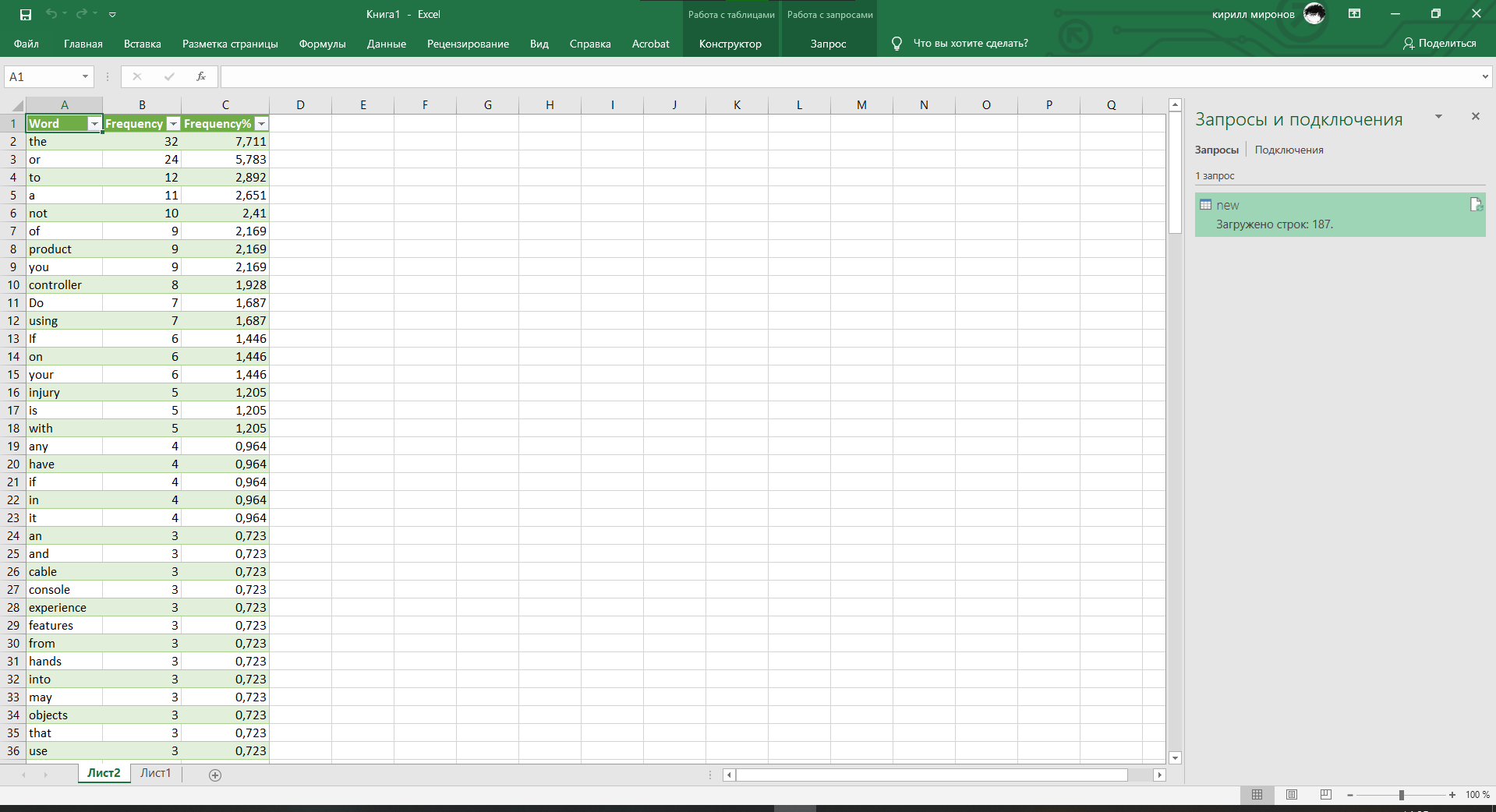


Рисунок 2 - Файл импортированный в Excel

Также класс может вывести обработанные данные в консоль в обход файла, результат показан на Рисунке 3.

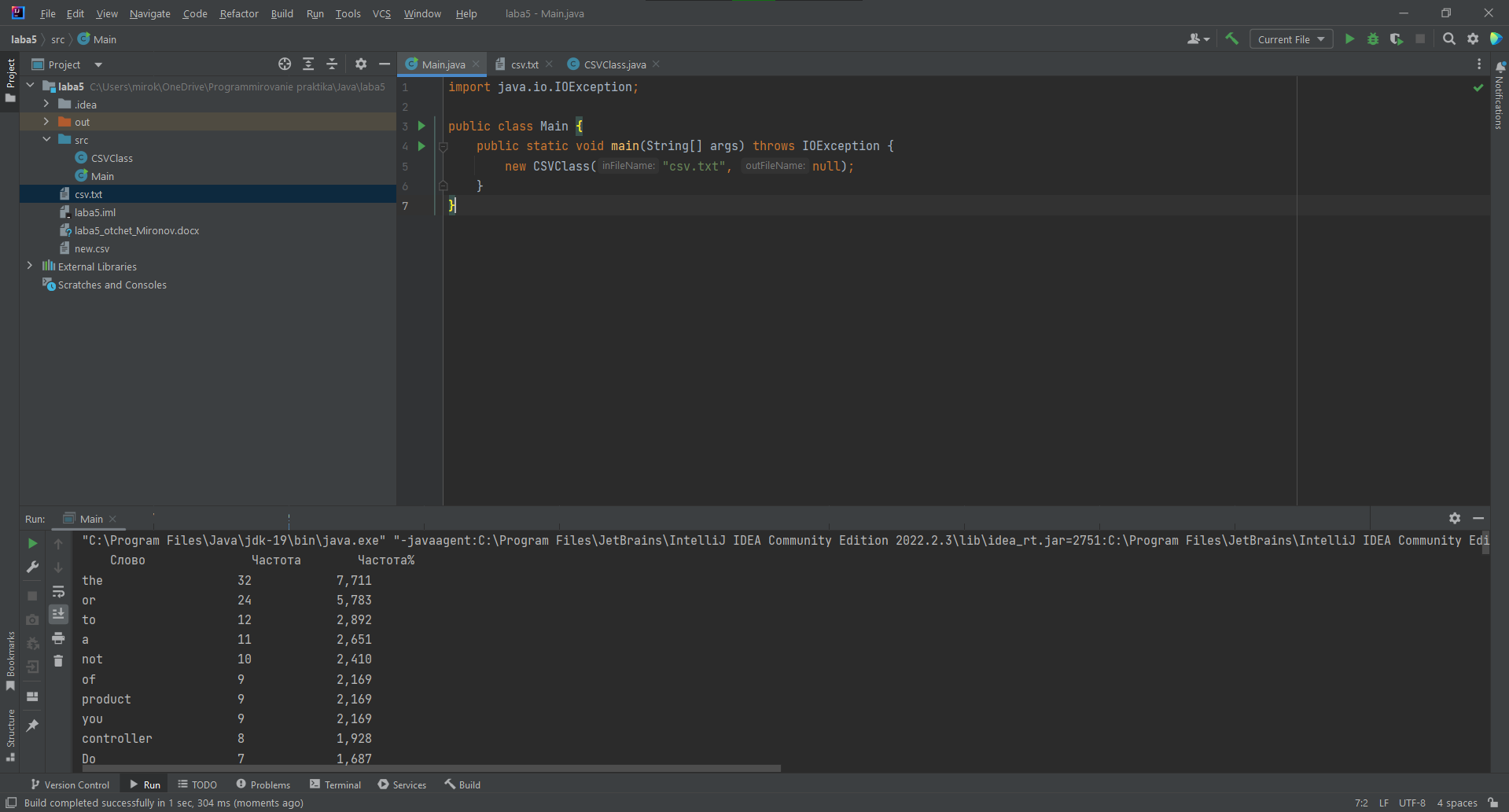


Рисунок 3 - Вывод в консоль

# ЗАКЛЮЧЕНИЕ

В данной работе реализован весь запрошенный в задании функционал, работоспособность каждого метода в классе успешно проверена с помощью тестовой программы.

# ПРИЛОЖЕНИЕ

Файл main.java

import java.io.IOException;  
  
public class Main {  
 public static void main(String[] args) throws IOException {  
 new CSVClass("csv.txt", "new.csv");  
 }  
}

Файл CSVClass.java

import java.io.\*;  
import java.util.\*;  
  
class CSVClass {  
  
 private static final String *SYMBOL\_CSV* = "\t";  
 private Map<String, Integer> mapWords = new HashMap<>();  
 private List<Map.Entry<String, Integer>> listWords;  
 private int count;  
  
 CSVClass(String inFileName, String outFileName) throws IOException {  
  
 if (inFileName == null) {  
 System.*out*.println("\n\tФайл не найден!");  
 return;  
 } else {  
 readFileWords(inFileName);  
 }  
  
 if (outFileName != null) {  
 File file = new File(outFileName);  
 try (PrintWriter writer = new PrintWriter(file.getAbsoluteFile())) {  
 writeFileWords(listWords, count, writer);  
 System.*out*.println("\n\tФайл успешно записан!");  
 }  
 } else {  
 showWords(listWords, count);  
 }  
 }  
  
 private void readFileWords(String inFileName) throws IOException {  
  
 try (Reader reader = new InputStreamReader(new BufferedInputStream(new FileInputStream(inFileName)))) {  
 StringBuilder builder = new StringBuilder();  
 int currentSymbol;  
  
 while ((currentSymbol = reader.read()) != -1) {  
 char charSymbol = (char) currentSymbol;  
  
 if (Character.*isLetterOrDigit*(charSymbol) || charSymbol == '\'') {  
 builder.append(charSymbol);  
 } else if (builder.length() > 0) {  
 addMap(mapWords, builder.toString());  
 count++;  
 builder.delete(0, builder.length());  
 }  
 }  
  
 if (builder.length() > 0) {  
 addMap(mapWords, builder.toString());  
 count++;  
 }  
  
 } catch (FileNotFoundException e) {  
 System.*out*.println("\n\tФайл не найден!");  
 }  
  
 listWords = sortedMap(mapWords);  
 }  
  
 private void addMap(Map<String, Integer> map, String word) {  
 int pass = 1;  
 Integer curPass = map.get(word);  
 if (curPass != null) {  
 pass += curPass;  
 }  
 map.put(word, pass);  
 }  
  
 private List<Map.Entry<String, Integer>> sortedMap(Map<String, Integer> map) {  
 List<Map.Entry<String, Integer>> list = new ArrayList<>(map.entrySet());  
 Collections.*sort*(list, (o1, o2) -> {  
 int rs = o2.getValue().compareTo(o1.getValue());  
 if (rs == 0) {  
 rs = o1.getKey().compareTo(o2.getKey());  
 }  
 return rs;  
 });  
 return list;  
 }  
  
 private void writeFileWords(List<Map.Entry<String, Integer>> list, int numbersOfLines, PrintWriter writer) {  
  
 writer.printf("%s" + *SYMBOL\_CSV* + "%s" + *SYMBOL\_CSV* + "%s%n", "Word", "Frequency", "Frequency%");  
 for (Map.Entry<String, Integer> aList : list) {  
 writer.printf("%s" + *SYMBOL\_CSV* + "%s" + *SYMBOL\_CSV* + "%.3f%n", aList.getKey(), aList.getValue(), (100.0f \* aList.getValue() / numbersOfLines));  
 }  
 writer.printf("%n%s" + *SYMBOL\_CSV* + "%,d", "Total:", numbersOfLines);  
 }  
  
 private void showWords(List<Map.Entry<String, Integer>> list, int numbersOfLines) {  
  
 System.*out*.printf("\t%-20s%-15s%s%n", "Слово", "Частота", "Частота%");  
 for (Map.Entry<String, Integer> aList : list) {  
 System.*out*.printf("\t\b%-22s%-14s%.3f%n", aList.getKey(), aList.getValue(), (100.0f \* aList.getValue() / numbersOfLines));  
 }  
 System.*out*.printf("\n\tОбщее количество строк:" + numbersOfLines);  
 }  
}