**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего образования «Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

Кафедра «Электронные вычислительные машины»

**Отчет о лабораторной работе № 7**

по теме

«Строки»

**Выполнили:**

Студенты гр. 045

Бригады №10

Анохин В.А.

Вашкулатов Н.А.

**Проверил:**

Асс. Бастрычкин А.С.

Цель работы:изучение классов Java, поддерживающих хранение строк, приобретение навыков использования строк в Java-программах.

Ход работы

Задание 1

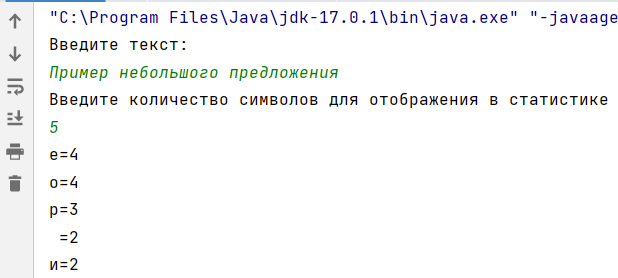
В тексте найти и напечатать n символов (и их количество), встречающихся наиболее часто.

Код программы:

Основная программа

import java.util.HashMap;  
import java.util.Map;  
import java.util.Scanner;  
  
public class App {  
 public static void main(String[] args) {  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Введите текст: ");  
 String text = scan.nextLine();  
 System.out.println("Введите количество символов для отображения в статистике");  
 long limit = scan.nextLong();  
 scan.close();  
 printStatistic(text, limit);  
 }  
  
 public static void printStatistic(String text, long limit){  
 Map<Character, Integer> statistic = new HashMap<>();  
 for(int i=0; i<text.length();++i){  
 Integer n = statistic.get(text.charAt(i));  
 if(n == null){  
 statistic.put(text.charAt(i), 1);  
 }  
 else{  
 statistic.put(text.charAt(i), ++n);  
 }  
 }  
 statistic.entrySet().stream()  
 .sorted(Map.Entry.<Character, Integer>comparingByValue().reversed())  
 .limit(limit)  
 .forEach(System.out::println);  
 }  
}

Окно вывода:



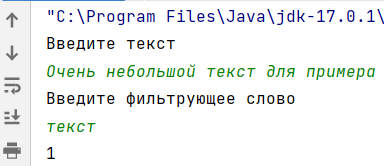
Задание 2

Определить количество вхождений заданного слова в текст, без учета регистра, считая буквы «е»/«ё» и «и»/«й» одинаковыми.

Код программы:

import java.util.\*;  
  
public class App {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.println("Введите текст");  
 String text = scanner.nextLine();  
 System.out.println("Введите фильтрующее слово");  
 String filterWord = scanner.nextLine();  
 scanner.close();  
 text = text.toLowerCase(Locale.ROOT)  
 .replace("ё", "е")  
 .replace("й","и");  
 String[] splitedText = text.split(" ");  
 List<String> words = Arrays.stream(splitedText).toList();  
 words = words.stream().filter(s -> s.equals(filterWord)).toList();  
 System.out.println(words.size());  
 }  
}

Окно вывода:



Задание 3

Создать программу обработки текста учебника по программированию с использованием классов: *Символ, Слово, Предложение, Абзац, Лексема, Листинг, Знак препинания* и др. Во всех задачах с формированием текста заменять табуляции и последовательности пробелов одним пробелом.

Существует текст и список слов. Для каждого слова из списка найти, сколько раз оно встречается в каждом предложении, рассортировать слова по убыванию общего количества вхождений.

**Код программы:**

**Класс Token**

public abstract class Token {  
 private String value;  
  
 Token(String val){  
 value = val;  
 }  
  
 public String getValue() {  
 return value;  
 }  
  
 @Override  
 public String toString() {  
 return value;  
 }  
}

Класс Word

public class Word extends Token{  
 Word(String word){  
 super(word);  
 }  
}

Класс Punctuation

public class Punctuation extends Token{  
 Punctuation(String mark){  
 super(mark);  
 }  
  
 public boolean isEndMark(){  
 return switch (toString()){  
 case "," -> false;  
 case ":" -> false;  
 case ";" -> false;  
 case "\n" -> false;  
 default -> true;  
 };  
 }  
}

Класс Sentence

import java.util.ArrayList;  
import java.util.HashMap;;  
  
public class Sentence {  
 private ArrayList<Token> tokens;  
  
 Sentence(ArrayList<Token> tokens){  
 this.tokens = tokens;  
 }  
  
 public ArrayList<Token> getTokens() {  
 return tokens;  
 }  
  
 public HashMap<String, Integer> findTokens(ArrayList<String> lexemes){  
 HashMap<String, Integer> result = new HashMap<>();  
 lexemes.forEach(word -> result.put(word, 0));  
 for(Token word: tokens){  
 String value = word.getValue().toLowerCase();  
 if(lexemes.contains(value)){  
 Integer n = result.get(value);  
 result.put(value, n+1);  
 }  
 }  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder result = new StringBuilder();  
 result.append(tokens.get(0));  
 for(int i = 1; i<tokens.size(); ++i){  
 if(tokens.get(i) instanceof Word){  
 result.append(" ");  
 }  
 result.append(tokens.get(i));  
 }  
 return result.toString();  
 }  
}

Класс Paragraph

import java.util.ArrayList;  
import java.util.stream.Collectors;  
  
public class Paragraph {  
 private ArrayList<Sentence> sentences;  
  
 Paragraph(ArrayList<Sentence> sentences){  
 this.sentences = sentences;  
 }  
  
 public ArrayList<Sentence> getSentences() {  
 return sentences;  
 }  
  
 @Override  
 public String toString() {  
 return sentences.stream()  
 .map(Sentence::toString)  
 .collect(Collectors.joining(" "));  
 }  
}

Класс Text

import java.util.ArrayList;  
import java.util.stream.Collectors;  
  
public class Text {  
 private ArrayList<Paragraph> paragraphs;  
  
 Text(ArrayList<Paragraph> paragraphs){  
 this.paragraphs = paragraphs;  
 }  
  
 public ArrayList<Paragraph> getParagraphs() {  
 return paragraphs;  
 }  
  
 @Override  
 public String toString() {  
 return paragraphs.stream()  
 .map(Paragraph::toString)  
 .collect(Collectors.joining("\n"));  
 }  
}

Класс Lexer

import java.util.HashSet;  
  
public class Lexer {  
 private int pos;  
 private char currentChar;  
 private String text;  
 private final HashSet<Character> punctuations;  
  
 Lexer(){  
 punctuations = new HashSet<>();  
 punctuations.add('.');  
 punctuations.add('!');  
 punctuations.add('?');  
 punctuations.add(',');  
 punctuations.add(';');  
 punctuations.add(':');  
 }  
  
 private void advance(){  
 pos++;  
 if (pos < text.length()){  
 currentChar = text.charAt(pos);  
 }  
 else currentChar = '\0';  
 }  
  
 private char next\_char(int next){  
 int nextPos = pos + next;  
 if (nextPos < text.length()){  
 return text.charAt(nextPos);  
 }  
 else return '\0';  
 }  
  
 private Word getWord(){  
 StringBuilder result = new StringBuilder();  
 while (Character.isLetter(currentChar) && currentChar != '\0'){  
 result.append(currentChar);  
 advance();  
 }  
 return new Word(result.toString());  
 }  
  
 public Token getNextToken(){  
 while (currentChar != '\0'){  
 if (currentChar == '\n'){  
 advance();  
 return new Punctuation("\n");  
 }  
 if(Character.isWhitespace(currentChar)){  
 advance();  
 continue;  
 }  
 if(Character.isLetter(currentChar)){  
 return getWord();  
 }  
 if(currentChar == '.' && next\_char(1) == '.' && next\_char(2) == '.'){  
 advance(); advance(); advance();  
 return new Punctuation("...");  
 }  
 if(punctuations.contains(currentChar)){  
 String mark = Character.toString(currentChar);  
 advance();  
 return new Punctuation(mark);  
 }  
 throw new RuntimeException("Лексическая ошибка: неизвестный символ '" + currentChar + "'");  
 }  
 return null;  
 }  
  
 public void setText(String text) {  
 this.text = text;  
 pos = 0;  
 currentChar = text.charAt(0);  
 }  
}

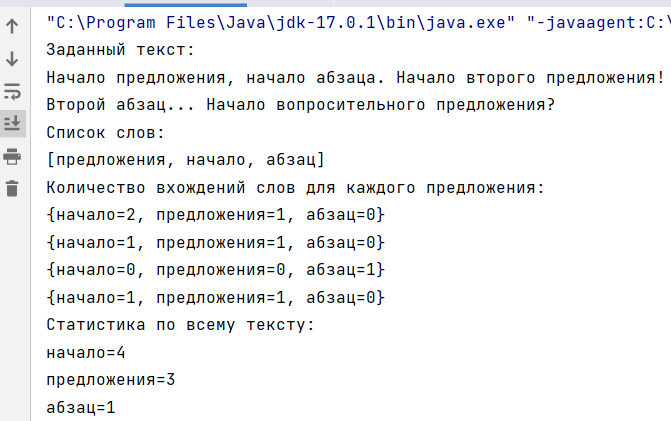
Класс Parser

import java.util.ArrayList;  
  
public class Parser {  
 private final Lexer lexer;  
 Token currentToken;  
 boolean notEOF;  
  
 Parser(){  
 lexer = new Lexer();  
 }  
  
 private void advance(){  
 currentToken = lexer.getNextToken();  
 if(currentToken == null){  
 notEOF = false;  
 }  
 }  
  
 private Sentence getSentence(){  
 ArrayList<Token> tokens = new ArrayList<>();  
 while (notEOF){  
 tokens.add(currentToken);  
 if(currentToken instanceof Punctuation){  
 if(((Punctuation) currentToken).isEndMark()){  
 advance();  
 break;  
 }  
 }  
 advance();  
 }  
 return new Sentence(tokens);  
 }  
  
 private Paragraph getParagraph(){  
 ArrayList<Sentence> sentences = new ArrayList<>();  
 while (notEOF){  
 sentences.add(getSentence());  
 if(notEOF && currentToken.getValue().equals("\n")){  
 advance();  
 break;  
 }  
 }  
 return new Paragraph(sentences);  
 }  
  
 public Text parseText(String text){  
 lexer.setText(text);  
 notEOF = true;  
 advance();  
 ArrayList<Paragraph> paragraphs = new ArrayList<>();  
 while (notEOF){  
 paragraphs.add(getParagraph());  
 }  
 return new Text(paragraphs);  
 }  
}

Основная программа

import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.Map;  
  
public class App {  
 public static void main(String[] args) {  
 Parser parser = new Parser();  
 String text = """  
 Начало предложения, начало абзаца. Начало второго предложения!  
 Второй абзац... Начало вопросительного предложения?""";  
 Text parsedText = parser.parseText(text);  
 System.out.println("Заданный текст:");  
 System.out.println(parsedText);  
 ArrayList<String> words = new ArrayList<>();  
 words.add("предложения");  
 words.add("начало");  
 words.add("абзац");  
 System.out.println("Список слов:");  
 System.out.println(words);  
 System.out.println("Количество вхождений слов для каждого предложения: ");  
 HashMap<String, Integer> statistics = new HashMap<>();  
 words.forEach(word -> statistics.put(word, 0));  
 for(Paragraph paragraph: parsedText.getParagraphs()){  
 for(Sentence sentence: paragraph.getSentences()){  
 HashMap<String, Integer> currentStat = sentence.findTokens(words);  
 System.out.println(currentStat);  
 currentStat.forEach((key, val)-> statistics.merge(key, val, Integer::sum));  
 }  
 }  
 System.out.println("Статистика по всему тексту:");  
 statistics.entrySet().stream()  
 .sorted(Map.Entry.<String, Integer>comparingByValue().reversed())  
 .forEach(System.out::println);  
 }

Окно вывода:



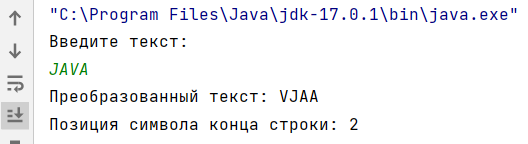
Задание 4

Алгоритм Барроуза-Уиллера для сжатия текстов основывается на преобразовании: для слова рассматриваются все его циклические сдвиги, которые затем сортируются в алфавитном порядке, после чего формируется слово из последних символов отсортированных циклических сдвигов. К примеру, для слова JAVA циклические сдвиги — это JAVA, AVAJ, VAJA, AJAV. После сортировки по алфавиту получим AJAV, AVAJ, JAVA, VAJA. Значит, результат преобразования — слово VJAA. Реализовать программно преобразование Барроуза-Уиллера для данного слова.

Код программы:

import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Scanner;  
  
public class App {  
 public static void main(String[] args) {  
 Scanner scan = new Scanner(System.in);  
 System.out.println("Введите текст: ");  
 StringBuilder text = new StringBuilder(scan.nextLine());  
 scan.close();  
 ArrayList<String> cyclicShifts = new ArrayList<>();  
 for (int i = 0; i<text.length(); ++i){  
 cyclicShifts.add(getCyclicShifts(text));  
 }  
 Collections.sort(cyclicShifts);  
 int eol = cyclicShifts.indexOf(text.toString()); //Необходимо для дешифрации  
 StringBuilder result = new StringBuilder();  
 cyclicShifts.stream()  
 .sorted()  
 .map(str -> str.charAt(str.length()-1))  
 .forEach(result::append);  
 System.out.println("Преобразованный текст: " + result);  
 System.out.println("Позиция символа конца строки: " + eol);  
 }  
  
 public static String getCyclicShifts(StringBuilder str){  
 return str.append(str.charAt(0))  
 .delete(0, 1)  
 .toString();  
 }  
}

Окно вывода:



**Вывод:** мы изучили классы Java, поддерживающие хранение строк, приобрели навыки использования строк в Java-программах.