**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 И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

|  |  |
| --- | --- |
| **по лабораторной работе № 7** |  |

**Дисциплина:** Строки. Регулярные выражения

**Название:** Языки программирования для работы с большими данными

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ6-22М |  | Д.М. Карасёв\_\_ |
|  | (Группа) | (Подпись, дата) | (И.О. Фамилия) |
| Преподаватель |  | П.В. Степанов | |
|  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Цель:** ознакомиться с базовыми принципами языка Java для работы с большими данными

**Вариант 1:**

1. В каждом слове текста k-ю букву заменить заданным символом. Если k больше длины слова, корректировку не выполнять.
2. В русском тексте каждую букву заменить ее порядковым номером в алфавите. При выводе в одной строке печатать текст с двумя пробелами между буквами, в следующей строке внизу под каждой буквой печатать ее номер.

**Решение:**

package lr7\_v1\_task1;  
  
public class Main {  
 public static String replaceKthLetter(String text, int k, char replacement){  
 String[] words = text.split(" ");  
 StringBuilder result = new StringBuilder();  
 for (String word : words){  
 if (word.length() >= k){  
 result.append(word.replace(word.charAt(k-1),replacement)).append(" ");  
 }  
 else {  
 result.append(word).append(" ");  
 }  
 }  
 return result.toString();  
 }  
 public static void main(String[] args) {  
 String text = "This is a test text";  
 int k = 3;  
 char replacement = '\*';  
 String replacedText = replaceKthLetter(text, k, replacement);  
 System.out.println(replacedText);  
 }  
}

На рисунке 1 представлен результат выполнения.



Рисунок 1- Результат выполнения

package lr7\_v1\_task2;  
  
public class Main {  
 public static void main(String[] args) {  
 String text = "Добрый день, ребята";  
 StringBuilder result = new StringBuilder();  
 StringBuilder numbers = new StringBuilder();  
  
 for (int i = 0; i < text.length(); i++){  
 char currentChar = text.charAt(i);  
 if (Character.*isLetter*(currentChar)){  
 int num = Character.*toLowerCase*(currentChar) - 'а' + 1;  
 result.append(currentChar).append(" ");  
 numbers.append(num).append(" ");  
 }  
 }  
  
 System.*out*.println(result);  
 System.*out*.println(numbers);  
 }  
}

На рисунке 2 представлен результат выполнения.



Рисунок 2 – Результат выполнения

**Вариант 2:**

1. В тексте найти все пары слов, из которых одно является обращением другого.

2. Найти и напечатать, сколько раз повторяется в тексте каждое слово, которое встречается в нем.

**Решение:**

package lr7\_v2\_task1;  
  
public class Main {  
 public static void main(String[] args) {  
 String text = "мама радар лотс дарар стол амам";  
 StringBuilder result = new StringBuilder();  
  
 String[] words = text.split("\\s");  
  
 for (String word : words){  
 String word1 = new StringBuilder(word).reverse().toString();  
 for (String word2 : words){  
 if(word1.equals(word2)){  
 result.append(word + " - " + word2 + "\n");  
 }  
 }  
 }  
  
 System.*out*.println(result);  
 }  
}

На рисунке 3 представлен результат выполнения.

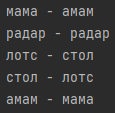


Рисунок 3 – Результат выполнения

package lr7\_v2\_task2;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class Main {  
 public static void main(String[] args) {  
 String text = "кот кот собака мышь мышь мышь лошадь лошадь лошадь лошадь";  
 Map<String , Integer> wordCount = new HashMap<>();  
  
 String[] words = text.split("\\s");  
  
 for(String word : words){  
 if(wordCount.containsKey(word)){  
 wordCount.put(word, wordCount.get(word)+1);  
 }  
 else {  
 wordCount.put(word, 1);  
 }  
 }  
  
 wordCount.forEach((key, value) -> {  
 System.*out*.println(key + " - " + value + " раз(а)");  
 });  
 }  
}

На рисунке 4 представлен результат выполнения.

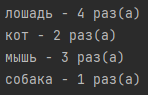


Рисунок 4 – Результат выполнения

**Вариант 3:**

1. В тексте нет слов, начинающихся одинаковыми буквами. Напечатать слова текста в таком порядке, чтобы последняя буква каждого слова совпадала с первой буквой последующего слова. Если все слова нельзя напечатать в таком порядке, найти такую цепочку, состоящую из наибольшего количества слов.
2. Найти наибольшее количество предложений текста, в которых есть одинаковые слова.

**Решение:**

package lr7\_v3\_task1;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
  
public class Main {  
 public static void main(String[] args) {  
 String text = "собака лампа авто фонарь облако";  
 ArrayList<String> words = new ArrayList<String>(Arrays.*asList*(text.split(" ")));  
 ArrayList<String> MaxChains = new ArrayList<String>();  
 int maxsize = 0;  
  
 for (int i = 0; i < words.size(); i++){  
 ArrayList<String> copyWords = new ArrayList<String>();  
 copyWords.addAll(words);  
 String currentWordOfChains = copyWords.get(i);  
  
 ArrayList<String> currentChains = new ArrayList<String>();  
 currentChains.add(currentWordOfChains);  
 copyWords.remove(currentWordOfChains);  
  
 int currentWordNumber = 0;  
 while (currentWordNumber < copyWords.size()){  
 String currentWord = copyWords.get(currentWordNumber);  
 if(currentWordOfChains.charAt(currentWordOfChains.length()-1) == currentWord.charAt(0)){  
 copyWords.remove(currentWord);  
 currentWordOfChains = currentWord;  
 currentChains.add(currentWordOfChains);  
 currentWordNumber = 0;  
 }  
 currentWordNumber++;  
 }  
  
 if (currentChains.size() > maxsize){  
 MaxChains = currentChains;  
 maxsize = currentChains.size();  
 }  
  
 }  
  
 System.*out*.println(MaxChains);  
 }  
}

На рисунке 5 представлен результат выполнения.



Рисунок 5 – Результат выполнения

package lr7\_v3\_task2;  
  
import java.util.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 String text = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent vel ligula eget ipsum blandit ultrices vel vel dolor. Duis porttitor tincidunt lectus, eget tincidunt eros tempus at. Aliquam pharetra, elit vel sollicitudin bibendum, urna enim feugiat leo, vel vestibulum sem nulla sed nulla. Sed vel ex ut orci tincidunt dapibus ut non velit. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Sed tristique turpis et enim pulvinar, vel ornare neque elementum. Vestibulum eu hendrerit arcu. Fusce venenatis elit et augue varius, vel pulvinar dolor eleifend. Nam euismod bibendum diam, non convallis turpis laoreet a.";  
 List<String> sentencesWithRepeat = new ArrayList<String>();  
 int countOfSentences = 0;  
  
 String[] sentences = text.split("[.!?]\\s");  
  
 for (String sentence : sentences){  
 Set<String> uniqueWord = new HashSet<>();  
 List<String> wordSentence = Arrays.*asList*(sentence.replaceAll("[^a-zA-Z ]", "").split(" "));  
 uniqueWord.addAll(wordSentence);  
 if(wordSentence.size() != uniqueWord.size()){  
 sentencesWithRepeat.add(sentence);  
 countOfSentences++;  
 }  
 }  
  
 System.*out*.println("Count of sentences: " + countOfSentences);  
  
 for (String sentence : sentencesWithRepeat){  
 System.*out*.println(sentence);  
 }  
 }  
}

На рисунке 6 представлен результат выполнения.

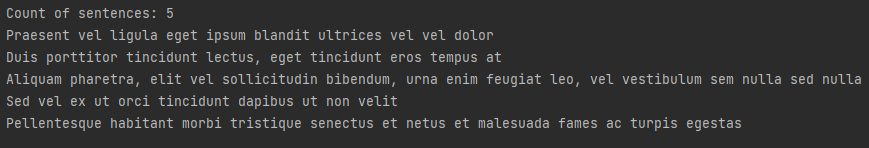


Рисунок 6 – Результат выполнения

**Вариант 4:**

1. Все слова английского текста рассортировать по возрастанию количества заданной буквы в слове. Слова с одинаковым количеством расположить в алфавитном порядке.
2. Ввести текст и список слов. Для каждого слова из заданного списка найти, сколько раз оно встречается в тексте, и рассортировать слова по убыванию количества вхождений.

**Решение:**

package lr7\_v4\_task1;  
  
import java.util.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 String text = "All the world's a stage, and all the men and women merely players. "  
 + "They have their exits and their entrances, and one man in his time plays many parts. "  
 + "As You Like It, Act II, Scene VII.";  
 List<String> words = *getWords*(text);  
 char letter = 'e';  
 *sortByLetter*(words, letter);  
 System.*out*.println(words);  
 }  
  
 private static List<String> getWords(String text) {  
 String[] wordsArr = text.replaceAll("[^a-zA-Z ]", "").toLowerCase().trim().split("\\s+");  
 return new ArrayList<>(Arrays.*asList*(wordsArr));  
 }  
  
 private static int countLetter(String word, char letter) {  
 int count = 0;  
 for (int i = 0; i < word.length(); i++) {  
 if (word.charAt(i) == letter) {  
 count++;  
 }  
 }  
 return count;  
 }  
  
 private static void sortByLetter(List<String> words, char letter) {  
 Comparator<String> letterCountComparator = new Comparator<String>() {  
 @Override  
 public int compare(String word1, String word2) {  
 int count1 = *countLetter*(word1, letter);  
 int count2 = *countLetter*(word2, letter);  
 if (count1 == count2) {  
 return word1.compareTo(word2);  
 }  
 return Integer.*compare*(count1, count2);  
 }  
 };  
 Collections.*sort*(words, letterCountComparator);  
 }  
}

На рисунке 7 представлен результат выполнения.



Рисунок 7 – Результат выполнения

package lr7\_v4\_task2;  
  
import java.util.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 String text = "The quick brown fox jumps over the lazy dog. The dog is lazy. Dog.";  
 String wordsForCount = "fox dog quick lazy";  
 Map<String, Integer> wordCount = new HashMap<>();  
  
 String[] words = wordsForCount.toLowerCase().split("\\s+");  
  
 for (String word : words){  
 wordCount.put(word, 0);  
 }  
  
 String[] textWords = text.toLowerCase().split("[^a-z]");  
  
 for (String word : textWords){  
 if(wordCount.containsKey(word)){  
 wordCount.put(word, wordCount.get(word) + 1);  
 }  
 }  
  
 List<Map.Entry<String, Integer>> mappings = new ArrayList<>(wordCount.entrySet());  
 Collections.*sort*(mappings, new Comparator<Map.Entry<String, Integer>>() {  
 public int compare(Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2) {  
 return o2.getValue().compareTo(o1.getValue());  
 }  
 });  
  
 for(Map.Entry<String, Integer> entry : mappings ){  
 System.*out*.println(entry.getKey() + " - " + entry.getValue());  
 }  
 }  
}

На рисунке 8 представлен результат выполнения.



Рисунок 8 – Результат выполнения

**Вывод:** Были разработаны классы и методы согласно варианту