|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

**Отчет**

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

**Название:**

Строки. Регулярные выражения

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

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

**Цель работы** — ознакомление с принципами работы со строками и регулярными выражениями в языке Java.

1. В тексте после буквы Р, если она не последняя в слове, ошибочно напечатана буква А вместо О. Внести исправления в текст.

Часть кода задания приведена в листинге 1, результат выполнения – на рисунке 1.

Листинг 1 – Замена PA на PO

|  |
| --- |
| String text = "some randome PO text with PA APA LAPA PATA word P";  String updatedStr = text.replaceAll("(PA\\b)", "PO");  System.out.println(updatedStr); |



Рисунок 1 – Результат замены PA на PO

1. В тексте слова заданной длины заменить указанной подстрокой, длина которой может не совпадать с длиной слова.

Часть кода задания приведена в листинге 2, результат выполнения – на рисунке 2.

Листинг 2 – Замена подстрокой слов длины 4

|  |
| --- |
| int length = 4;  updatedStr = text.replaceAll("\\b\\w{" + length + "}\\b", "substring");  System.out.println(updatedStr); |



Рисунок 2 – Замена подстрокой

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

Часть кода задания приведена в листинге 3, результат выполнения – на рисунке 3.

Листинг 3 – Нахождение n символов

|  |
| --- |
| public static void findAndPrintNSymbols(String text, int n) {  HashMap<Character, Integer> symbols = new HashMap<>();  for (int i = 0; i < text.length(); i++) {  char c = text.charAt(i);  if (symbols.containsKey(c)) {  symbols.put(c, symbols.get(c) + 1);  } else {  symbols.put(c, 1);  }  }  for (int i = 0; i < n; i++) {  int max = -1;  char maxC = ' ';  for (char c : symbols.keySet()) {  if (symbols.get(c) > max) {  max = symbols.get(c);  maxC = c;  }  }  System.out.println(maxC + ":" + max);  symbols.remove(maxC);  }  } |



Рисунок 3 – Три самых часто встречающихся символа

1. Найти, каких букв, гласных или согласных, больше в каждом предложении текста

Часть кода задания приведена в листинге 4, результат выполнения – на рисунке 4.

Листинг 4 – Три абитуриента с самыми высокими средними оценками

|  |
| --- |
| public static void findVowelConsonant(String text) {  String copy = text.toLowerCase();  String[] sentences = copy.split("[.]");  for (int i = 0; i < sentences.length; i++) {  int vowelCount = 0;  int consonantCount = 0;  for (int j = 0; j < sentences[i].length(); j++) {  Character c = sentences[i].charAt(j);  if (Character.isAlphabetic(c)) {  if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {  vowelCount++;  } else {  consonantCount++;  }  }  }  System.out.println("Vowels: " + vowelCount + " Consonants: " + consonantCount);  if (vowelCount > consonantCount) {  System.out.println("More vowels");  } else if (vowelCount < consonantCount) {  System.out.println("More consolants");  } else {  System.out.println("equal");  }  }  } |

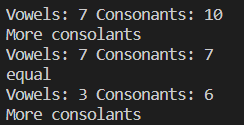


Рисунок 4 – Гласные и согласные в предложении

1. Найти такое слово в первом предложении, которого нет ни в одном из остальных предложений.

Часть кода задания приведена в листинге 5, результат выполнения – на рисунке 5.

Листинг 5 – Нахождение уникального слова

|  |
| --- |
| public static String findUniqueWord(String text) {  String[] sentences = text.split("[.]");  HashSet<String> sentence1Words = new HashSet<>();  String[] words1 = sentences[0].split("\s+");  for (String word : words1) {  sentence1Words.add(word);  }  for (int i = 1; i < sentences.length; i++) {  String[] words = sentences[i].split("\s+");  for (String word : words) {  if (sentence1Words.contains(word)) {  sentence1Words.remove(word);  }  }  }  return sentence1Words.iterator().next();  } |



Рисунок 5 – Уникальные слова

1. Во всех вопросительных предложениях текста найти и напечатать без повторений слова заданной длины.

Часть кода задания приведена в листинге 6, результат выполнения – на рисунке 6.

Листинг 6 – Нахождение слов заданной длины

|  |
| --- |
| public static void findWithLength(String text, int length) {  String[] sentences = text.split("(?<=[!?.])");  HashSet<String> words = new HashSet<>();  for (int i = 0; i < sentences.length; i++) {  if (!sentences[i].endsWith("?")) {  continue;  }  sentences[i] = sentences[i].substring(0, sentences[i].length() - 1);  String[] wordsInSentence = sentences[i].split("\s+");  for (String word : wordsInSentence) {  if (word.length() == length) {  words.add(word);  }  }  }  for (String word : words) {  System.out.println(word);  }  } |



Рисунок 6 – Слова заданной длины без повторений

1. Все слова текста рассортировать в порядке убывания их длин, при этом все слова одинаковой длины рассортировать в порядке возрастания в них количества гласных букв.

Часть кода задания приведена в листинге 7, результат выполнения – на рисунке 7.

Листинг 7 – Сортировка слов

|  |
| --- |
| public static void sortByLengthVowel(String text) {  String[] words = text.split("\s+|[.?!]");  HashMap<String, Integer> wordToLength = new HashMap<>();  for (String word : words) {  wordToLength.put(word, word.length());  }  List<String> wordList = new ArrayList<>(wordToLength.keySet());  Collections.sort(wordList, new Comparator<String>() {  @Override  public int compare(String s1, String s2) {  int length1 = wordToLength.get(s1);  int length2 = wordToLength.get(s2);  if (length1 != length2) {  return Integer.compare(length2, length1);  } else {  int vowelCount1 = countVowels(s1);  int vowelCount2 = countVowels(s2);  return Integer.compare(vowelCount1, vowelCount2);  }  }  });  for (String word : wordList) {  System.out.println(word);  }  } |

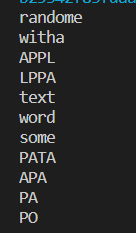


Рисунок 7 – Список отсортированных слов

1. В тексте исключить подстроку максимальной длины, начинающуюся и заканчивающуюся заданными символами.

Часть кода задания приведена в листинге 8, результат выполнения – на рисунке 8.

Листинг 8 – Исключение подстроки

|  |
| --- |
| public static String removeMaxSubstring(String text, char startChar, char endChar) {  int max = 0;  int startIndex = 0;  for (int i = 0; i < text.length(); i++) {  if (text.charAt(i) == startChar) {  for (int j = i + 1; j < text.length(); j++) {  if (text.charAt(j) == endChar && j - i > max) {  max = j - i;  startIndex = i;  }  }  }  }  return (max > 0) ? text.substring(0, startIndex) + text.substring(startIndex + max + 1) : text;  } |



Рисунок 8 – Текст после исключения подстроки

Полный код заданий размещен в репозитории по ссылке – https://github.com/moroz-matros/BDL.

**Вывод** — в ходе работы были получены навыки работы со строками и регулярными выражениями в языке Java.