|  |  |
| --- | --- |
|  | **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 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 |  |  | С.В. Астахов | |
|  | (Группа) |  | (Подпись, дата) | | (И.О. Фамилия) |
|  |  |  |  | |  |
| Преподаватель |  |  |  | | П.В. Степанов |
|  |  |  | (Подпись, дата) | | (И.О. Фамилия) |

Москва, 2024

**Цель работы:** освоить базовые принципы работы со строками и регулярными выражениями на языке Java.

**Вариант: 1.**

**Задание 1:** В каждом слове текста k-ю букву заменить заданным символом. Если k больше длины слова, корректировку не выполнять.

Код решения приведен в листинге 1.

Листинг 1 — реализация решения

|  |
| --- |
| package org.example;  import java.io.\*;  import java.util.\*;  import java.util.stream.Collectors;  import java.util.stream.IntStream;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== Regex work example ===\n\n");  String inputString = "Hello World Java Example a b";  int k = 2;  char replacementChar = '@';  String result = inputString.replaceAll("(\\b\\w{" + (k - 1) + "})(\\w)(\\w\*)", "$1" + replacementChar + "$3");  System.out.println("k: " + k);  System.out.println("ReplacementChar: " + replacementChar);  System.out.println("Input String: " + inputString);  System.out.println("Modified String: " + result);  }  } |

Результат исполнения программы приведен в листинге 2.

Листинг 2 — результат исполнения программы

|  |
| --- |
| === Regex work example ===  k: 2  ReplacementChar: @  Input String: Hello World Java Example a b  Modified String: H@llo W@rld J@va E@ample a b |

**Задание 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.

Код решения приведен в листинге 3.

Листинг 3 — реализация решения

|  |
| --- |
| package org.example;  import java.io.\*;  import java.util.\*;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  import java.util.stream.Collectors;  import java.util.stream.IntStream;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== Regex work example ===\n\n");  String inputString = "Привет, Джава!";  Pattern pattern = Pattern.compile("."); // "\\p{L}"  String result = "";  String inputCopy = "";  Matcher matcher = pattern.matcher(inputString);  while (matcher.find()) {  result += String.format("\\u%04x ", (int) inputString.substring(matcher.start(), matcher.end()).charAt(0));  inputCopy += String.format("%c ", (int) inputString.substring(matcher.start(), matcher.end()).charAt(0));  }  System.out.println("Input String: " + inputCopy);  System.out.println("Modified String: " + result);  }  } |

Результат исполнения программы приведен в листинге 4.

Листинг 4 — результат исполнения программы

|  |
| --- |
| === Regex work example ===  Input String: П р и в е т , Д ж а в а !  Modified String: \u041f \u0440 \u0438 \u0432 \u0435 \u0442 \u002c \u0020 \u0414 \u0436 \u0430 \u0432 \u0430 \u0021 |

**Задание 3:** В тексте найти все пары слов, из которых одно является обращением другого.

Код решения приведен в листинге 5.

Листинг 5 — реализация решения

|  |
| --- |
| package org.example;  import java.io.\*;  import java.util.\*;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  import java.util.stream.Collectors;  import java.util.stream.IntStream;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== Regex work example ===\n\n");  String inputString = "Hello, Java! avaJ abc cba cba";  Pattern pattern = Pattern.compile("\\b\\w+\\b");  Matcher matcher = pattern.matcher(inputString);  System.out.println("Input String: " + inputString);  while (matcher.find()) {  String sub = inputString.substring(matcher.start(), matcher.end());  String sub\_reversed = new StringBuilder(sub).reverse().toString();  Pattern pattern2 = Pattern.compile("\\b" + sub\_reversed + "\\b");  Matcher matcher2 = pattern2.matcher(inputString);  while (matcher2.find()) { // if for single usage + set for no reverse pairs  System.out.printf("%s : %s\n", sub, inputString.substring(matcher2.start(), matcher2.end()));  }  }  }  } |

Результат исполнения программы приведен в листинге 6.

Листинг 6 — результат исполнения программы

|  |
| --- |
| === Regex work example ===  Input String: Hello, Java! avaJ abc cba cba  Java : avaJ  avaJ : Java  abc : cba  abc : cba  cba : abc  cba : abc |

**Задание 4:** Найти и напечатать, сколько раз повторяется в тексте каждое слово, которое встречается в нем.

Код решения приведен в листинге 7.

Листинг 7 — реализация решения

|  |
| --- |
| package org.example;  import java.io.\*;  import java.util.\*;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  import java.util.stream.Collectors;  import java.util.stream.IntStream;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== Regex work example ===\n\n");  String inputString = "Hello, Java! Java Java abc cba cba hello hElLo";  Pattern pattern = Pattern.compile("\\b\\w+\\b");  Matcher matcher = pattern.matcher(inputString);  System.out.println("Input String: " + inputString);  Set<String> words = new HashSet<String>();  while (matcher.find()) {  String sub = inputString.substring(matcher.start(), matcher.end());  if(!words.contains(sub.toLowerCase())) {  words.add(sub.toLowerCase());  int cnt = 0;  Pattern pattern2 = Pattern.compile("\\b" + sub + "\\b", Pattern.CASE\_INSENSITIVE);  Matcher matcher2 = pattern2.matcher(inputString);  while (matcher2.find()) {  cnt++;  }  System.out.printf("%s : %d\n", sub, cnt);  }  }  }  } |

Результат исполнения программы приведен в листинге 8.

Листинг 8 — результат исполнения программы

|  |
| --- |
| === Regex work example ===  Input String: Hello, Java! Java Java abc cba cba hello hElLo  Hello : 3  Java : 3  abc : 1  cba : 2 |

**Задание 5:** 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.

Код решения приведен в листинге 9.

Листинг 9 — реализация решения

|  |
| --- |
| package org.example;  import java.io.\*;  import java.util.\*;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  import java.util.stream.Collectors;  import java.util.stream.IntStream;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== Regex work example ===\n\n");  String inputString = "apple zalgo error nuget redemtion orange";  ArrayList<String> strs = new ArrayList<String>();  Pattern pattern = Pattern.compile("\\b\\w+\\b");  Matcher matcher = pattern.matcher(inputString);  while (matcher.find()) {  String sub = inputString.substring(matcher.start(), matcher.end());  strs.add(sub);  }  strs = buildChains(inputString, strs, 0);  System.out.println("Input String: " + inputString);  System.out.println("Longest String: " + strs.stream().max((s1, s2) -> s1.length() - s2.length()).orElseThrow());  }  private static ArrayList<String> buildChains(String inputString, ArrayList<String> current\_chains, int depth){  ArrayList<String> new\_chains = new ArrayList<String>();  boolean changes\_fl = false;  for(int i = 0; i < current\_chains.size(); i++) {  String end = current\_chains.get(i).substring(current\_chains.get(i).length() - 1);  System.out.printf("=> [%d] '%s' : '%s'\n", depth ,end, current\_chains.get(i));  Pattern pattern = Pattern.compile("\\b" + end + "\\w+\\b");  Matcher matcher = pattern.matcher(inputString);  boolean fl\_new\_appends = false;  while (matcher.find()) {  changes\_fl = true;  fl\_new\_appends = true;  new\_chains.add(current\_chains.get(i) + " " + inputString.substring(matcher.start(), matcher.end()));  }  if(!fl\_new\_appends){  new\_chains.add(current\_chains.get(i));  }  }  if(changes\_fl){  new\_chains = buildChains(inputString, new\_chains, depth + 1);  }  return new\_chains;  }  } |

Результат исполнения программы приведен в листинге 10.

Листинг 10 — результат исполнения программы

|  |
| --- |
| === Regex work example ===  => [0] 'e' : 'apple'  => [0] 'o' : 'zalgo'  => [0] 'r' : 'error'  => [0] 't' : 'nuget'  => [0] 'n' : 'redemtion'  => [0] 'e' : 'orange'  => [1] 'r' : 'apple error'  => [1] 'e' : 'zalgo orange'  => [1] 'n' : 'error redemtion'  => [1] 't' : 'nuget'  => [1] 't' : 'redemtion nuget'  => [1] 'r' : 'orange error'  => [2] 'n' : 'apple error redemtion'  => [2] 'r' : 'zalgo orange error'  => [2] 't' : 'error redemtion nuget'  => [2] 't' : 'nuget'  => [2] 't' : 'redemtion nuget'  => [2] 'n' : 'orange error redemtion'  => [3] 't' : 'apple error redemtion nuget'  => [3] 'n' : 'zalgo orange error redemtion'  => [3] 't' : 'error redemtion nuget'  => [3] 't' : 'nuget'  => [3] 't' : 'redemtion nuget'  => [3] 't' : 'orange error redemtion nuget'  => [4] 't' : 'apple error redemtion nuget'  => [4] 't' : 'zalgo orange error redemtion nuget'  => [4] 't' : 'error redemtion nuget'  => [4] 't' : 'nuget'  => [4] 't' : 'redemtion nuget'  => [4] 't' : 'orange error redemtion nuget'  Input String: apple zalgo error nuget redemtion orange  Longest String: zalgo orange error redemtion nuget |

**Задание 6:** Найти наибольшее количество предложений текста, в которых есть одинаковые слова.

Код решения приведен в листинге 11.

Листинг 11 — реализация решения

|  |
| --- |
| package org.example;  import java.io.\*;  import java.util.\*;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  import java.util.stream.Collectors;  import java.util.stream.IntStream;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== Regex work example ===\n\n");  String inputString = "Hello, Java! Java Java abc cba cba hello hElLo. Hello. Bbnn? Java. Mmmm!";  Pattern pattern = Pattern.compile("([A-Z][^.!?]\*[.!?])");  Matcher matcher = pattern.matcher(inputString);  System.out.println("Input String: " + inputString);  ArrayList<String> sentences = new ArrayList<String>();  while (matcher.find()) {  System.out.println("=> " + inputString.substring(matcher.start(), matcher.end()));  sentences.add(inputString.substring(matcher.start(), matcher.end()));  }  // count max word-connected graph nodes  int max = 0;  for(Integer i=0; i < sentences.size(); i++){  int cnt = 0;  for (Integer j=0; j < sentences.size(); j++) {  Pattern pattern2 = Pattern.compile("(^|\\b)\\w+[ .!?,]", Pattern.CASE\_INSENSITIVE);  Matcher matcher2 = pattern2.matcher(sentences.get(i));  // System.out.println(">>> " + sentences.get(i));  while (matcher2.find()) {  String sub = sentences.get(i).substring(matcher2.start(), matcher2.end() - 1);  // System.out.println(i.toString() + " - " + sub);  Pattern pattern3 = Pattern.compile("(^|\\b)" + sub + "[ .!?,]", Pattern.CASE\_INSENSITIVE);  Matcher matcher3 = pattern3.matcher(sentences.get(j));  if(matcher3.find()){  System.out.println("[" + i.toString() + ":" + j.toString() + "] " + sub + " : " + sentences.get(j).substring(matcher3.start(), matcher3.end() - 1));  cnt++;  break;  }  }  }  if(cnt > max){  max = cnt;  }  }  System.out.println("Max: " + max);  }  } |

Результат исполнения программы приведен в листинге 12.

Листинг 12 — результат исполнения программы

|  |
| --- |
| === Regex work example ===  Input String: Hello, Java! Java Java abc cba cba hello hElLo. Hello. Bbnn? Java. Mmmm!  => Hello, Java!  => Java Java abc cba cba hello hElLo.  => Hello.  => Bbnn?  => Java.  => Mmmm!  [0:0] Hello : Hello  [0:1] Hello : hello  [0:2] Hello : Hello  [0:4] Java : Java  [1:0] Java : Java  [1:1] Java : Java  [1:2] hello : Hello  [1:4] Java : Java  [2:0] Hello : Hello  [2:1] Hello : hello  [2:2] Hello : Hello  [3:3] Bbnn : Bbnn  [4:0] Java : Java  [4:1] Java : Java  [4:4] Java : Java  [5:5] Mmmm : Mmmm  Max: 4 |

**Задание 7:** Все слова английского текста рассортировать по возрастанию количества заданной буквы в слове. Слова с одинаковым количеством расположить в алфавитном порядке.

Код решения приведен в листинге 13.

Листинг 13 — реализация решения

|  |
| --- |
| package org.example;  import java.io.\*;  import java.util.\*;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  import java.util.stream.Collectors;  import java.util.stream.IntStream;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== Regex work example ===\n\n");  String inputString = "Hello, Java! Java Java abc cba cba hello hElLo App AAAAA. Hello. Bbnn? Java. Mmmm!";  String inputChar = "a";  Pattern pattern = Pattern.compile("(^|\\b)\\w+[ .!?,]", Pattern.CASE\_INSENSITIVE);  Matcher matcher = pattern.matcher(inputString);  System.out.println("Input String: " + inputString);  System.out.println("Input Char: " + inputChar);  ArrayList<CountedWord> words = new ArrayList<CountedWord>();  while (matcher.find()) {  String word = inputString.substring(matcher.start(), matcher.end() - 1);  Pattern pattern2 = Pattern.compile(inputChar, Pattern.CASE\_INSENSITIVE);  Matcher matcher2 = pattern2.matcher(word);  words.add(new CountedWord(word, matcher2.results().count()));  }  words.sort(Main::compareWords);  for(CountedWord word : words){  System.out.println(word);  }  }  private static int compareWords(CountedWord cw1, CountedWord cw2){  if (cw1.cnt != cw2.cnt) {  return (int) (cw1.cnt - cw2.cnt);  }  return cw1.word.compareTo(cw2.word);  }  }  class CountedWord {  public String word;  public long cnt;  CountedWord(String word, long cnt){  this.word = word;  this.cnt = cnt;  }  public String toString(){  return String.format("<word> %s [%d]", word, cnt);  }  } |

Результат исполнения программы приведен в листинге 14.

Листинг 14 — результат исполнения программы

|  |
| --- |
| === Regex work example ===  Input String: Hello, Java! Java Java abc cba cba hello hElLo App AAAAA. Hello. Bbnn? Java. Mmmm!  Input Char: a  <word> Bbnn [0]  <word> Hello [0]  <word> Hello [0]  <word> Mmmm [0]  <word> hElLo [0]  <word> hello [0]  <word> App [1]  <word> abc [1]  <word> cba [1]  <word> cba [1]  <word> Java [2]  <word> Java [2]  <word> Java [2]  <word> Java [2]  <word> AAAAA [5] |

**Задание 8:** 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.

Код решения приведен в листинге 15.

Листинг 15 — реализация решения

|  |
| --- |
| package org.example;  import java.io.\*;  import java.util.\*;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  import java.util.stream.Collectors;  import java.util.stream.IntStream;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== Regex work example ===\n\n");  String inputString = "Hello, Java! Java Java abc cba cba hello hElLo App AAAAA. Hello. Bbnn? Java. Mmmm!";  String[] seacrh\_words = new String[]{"java", "hello", "abc"};  System.out.println("Input String: " + inputString);  ArrayList<CountedWord> words = new ArrayList<CountedWord>();  for (String word : seacrh\_words) {  Pattern pattern = Pattern.compile("(^|\\b)" + word + "[ .!?,]", Pattern.CASE\_INSENSITIVE);  Matcher matcher = pattern.matcher(inputString);  words.add(new CountedWord(word, matcher.results().count()));  }  words.sort(CountedWord::compareWordsDesc);  for(CountedWord word : words){  System.out.println(word);  }  }  }  class CountedWord {  public String word;  public long cnt;  CountedWord(String word, long cnt){  this.word = word;  this.cnt = cnt;  }  public String toString(){  return String.format("<word> %s [%d]", word, cnt);  }  public static int compareWordsDesc(CountedWord cw1, CountedWord cw2){  return (int) (- cw1.cnt + cw2.cnt);  }  } |

Результат исполнения программы приведен в листинге 16.

Листинг 16 — результат исполнения программы

|  |
| --- |
| === Regex work example ===  Input String: Hello, Java! Java Java abc cba cba hello hElLo App AAAAA. Hello. Bbnn? Java. Mmmm!  <word> java [4]  <word> hello [4]  <word> abc [1] |

**Вывод:** в ходе лабораторной работы были освоены базовые принципы работы со строками и регулярными выражениями на языке Java.