|  |  |
| --- | --- |
| 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 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 |
| **Вариант №** 8 |  |

**Название:**

Строки и регулярные выражения

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-23М | |  |  | Н.М. Иванюк |
|  | | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  | |  |  |  |  |
| Преподаватель | |  |  |  | П.В. Степанов |
|  | |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

## Цель работы

Получение первичных навыков работы со строками и регулярными выражениями языка программирования Java.

## Ход работы

*Задание 1.*

8. Из небольшого текста удалить все символы, кроме пробелов, не являющиеся буквами. Между последовательностями подряд идущих букв оставить хотя бы один пробел.

9. Из текста удалить все слова заданной длины, начинающиеся на согласную букву.

Листинг 1 – Код программы первого задания

public class App {

public static void main(String[] args) throws Exception {

String line = "Helloworld11@ whi1tespa1ces$ owari";

System.out.println(PurgeNonLetters(line));

String line2 = "agor amet solo dev";

System.out.println(RemoveLength(line2, "a", 4));

}

public static String PurgeNonLetters(String input) {

return input.replaceAll("[^a-zA-Z ]", "");

}

public static String RemoveLength(String input, String c, int length) {

String regex = "\\b" + c + "\\w{" + (length - 1) + "}\\b";

return input.replaceAll(regex, "");

}

}

Приведем результаты выполнения данного кода.

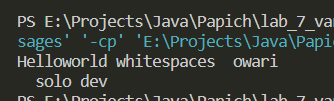


Рисунок 1 – Результат выполнения варианта задания 1

*Задание 2.*

8. Напечатать квитанцию об оплате телеграммы, если стоимость одного слова задана.

9. В стихотворении найти одинаковые буквы, которые встречаются во всех словах.

Листинг 2 – Код выполнения задания 2

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashSet;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class App {

public static void main(String[] args) throws Exception {

String telegram = "Ichi ni san yon go roku shichi";

System.out.println(CalculateCost(telegram, 10));

String poem = """

children sleep,

exchanging dreams her

seraphim

""";

System.out.println(Arrays.toString(FindSameWords(poem).toArray()));

}

public static int CalculateCost(String input, int price) {

int cost = 0;

String[] str = input.split(" +");

cost = str.length \* price;

return cost;

}

public static HashSet<Character> FindSameWords(String input) {

ArrayList<String> allMatches = new ArrayList<String>();

Matcher m = Pattern.compile("\\b\\w+\\b")

.matcher(input);

while (m.find()) {

allMatches.add(m.group());

}

ArrayList<HashSet<Character>> wordletters = new ArrayList<HashSet<Character>>();

for (String w : allMatches) {

HashSet<Character> charset = new HashSet<>();

for(char c : w.toCharArray()) {

charset.add(c);

}

wordletters.add(charset);

System.out.println(Arrays.toString(charset.toArray()));

}

HashSet<Character> cache = new HashSet<>();

for (int i = 0; i < wordletters.size(); i++) {

if (i == 0)

cache = wordletters.get(i);

else

cache.retainAll(wordletters.get(i));

}

return cache;

}

}

Результаты выполнения задания приведем на рисунке далее.

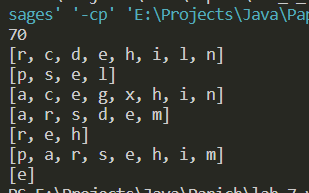


Рисунок 2 – Выполненный код задания 2

*Задание 3.*

– Текст шифруется по следующему правилу: из исходного текста выбирается 1, 4, 7, 10-й и т.д. (до конца текста) символы, затем 2, 5, 8, 11-й и т.д. (до конца текста) символы, затем 3, 6, 9, 12-й и т.д. Зашифровать заданный текст.

8. На основании правила кодирования, описанного в предыдущем примере, расшифровать заданный набор символов.

9. Напечатать слова русского текста в алфавитном порядке по первой букве. Слова, начинающиеся с новой буквы, печатать с красной строки.

Листинг 3 – Код основной программы

import java.util.ArrayList;

import java.util.HashMap;

import java.util.TreeMap;

import java.util.regex.Pattern;

import java.util.regex.Matcher;

public class App {

public static void main(String[] args) throws Exception {

String test = "abcdefgh";

String coded = EncodeString(test);

String decoded = EncodeString(coded);

System.out.println(coded);

System.out.println(decoded);

String poem = "Я достаю из широких штанин паспорт";

SortWords(poem);

}

public static String EncodeString(String input) {

int len = input.length();

String result = "";

int offset = 0;

while (result.length() < len) {

int index = 0 + offset;

while (index < input.length()) {

result += input.charAt(index);

index += 3;

}

offset++;

}

return result;

}

public static void SortWords(String input) {

ArrayList<String> allMatches = new ArrayList<String>();

Matcher m = Pattern.compile("\\b[А-Яа-яЁё]+\\b").matcher(input);

while (m.find()) {

allMatches.add(m.group());

}

TreeMap<Character, ArrayList<String>> words = new TreeMap<>();

for (String w:allMatches) {

ArrayList<String> s = words.get(Character.toLowerCase(w.charAt(0)));

if (s == null) {

words.put(Character.toLowerCase(w.charAt(0)), new ArrayList<>());

words.get(Character.toLowerCase(w.charAt(0))).add(w.toLowerCase());

} else {

s.add(w.toLowerCase());

}

}

for (ArrayList<String> ar : words.values()) {

System.out.println(" " + String.join(", ", ar));

}

}

}

Приведем результат выполнения программы:

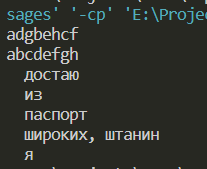


Рисунок 3 – Зашифрованный и расшифрованный текст

*Задание 4.*

8. Подсчитать, сколько раз заданное слово входит в текст.

9. Преобразовать каждое слово в тексте, удалив из него все последующие (предыдущие) вхождения первой (последней) буквы этого слова

Листинг 4 – Код программы

import java.lang.reflect.Array;

import java.util.ArrayList;

import java.util.Arrays;

public class App {

public static void main(String[] args) throws Exception {

String input = "lorem ipsum dolor sit amet consectetur adipiscing elit Praesent blandit concoction amet";

String[] str = input.split(" +");

String word = "amet";

int count = 0;

for (String w : str) {

if (w.toLowerCase().equals(word.toLowerCase())) count++;

}

System.out.println(count);

ArrayList<String> nwords = new ArrayList<>();

for (String w : str) {

String regex = Character.toString(Character.toLowerCase(w.charAt(0)));

String[] parts = w.split(regex, 2);

if (parts.length < 2) continue;

w = parts[0] + regex + parts[1].replaceAll(regex, "");

nwords.add(w);

}

System.out.println(Arrays.toString(nwords.toArray()));

}

}

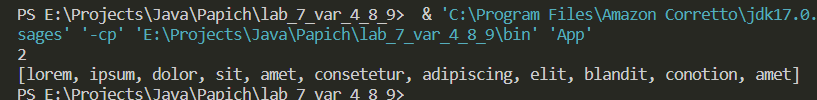


Рисунок 4 – Результат выполнения работы

## Вывод

По итогам выполнения лабораторной работы были получены навыки программирования с использованием строк в языке Java. Также были изучены способы обработки строк при помощи регулярных выражений.