Молдавский Государственный Университет

Факультет Математики и Информатики

Департамент Информатики

Лабораторная работа №4

по курсу Java

на тему

“[Обработка строк. Коллекции. Регулярные выражения.](https://moodle.usm.md/course/section.php?id=51735)”

Выполнил: Bogdanov Iurii,

группа I2302

Проверил: Gheorghe Latul

Кишинев, 2024

**Цель работы:**

Целью данной лабораторной работы является изучение и применение методов работы с текстом, строками, регулярными выражениями и коллекциями в Java. В рамках работы требуется разработать программу, которая:

1. Извлекает первое слово из введенного предложения.
2. Находит в предложении все слова, состоящие из тех же букв, что и первое слово.
3. Использует классы String, StringBuffer, коллекции и регулярные выражения для реализации задачи.

**Описание задачи:**

*Задание:*

* Перечислить все слова заданного предложения, которые состоят из тех же букв, что и первое слово предложения. Между словами могут быть произвольные пробелы и знаки препинания. Необходимо вывести предложение и найденные слова.

*Требования к реализации:*

* Использовать классы String и StringBuffer.
* Использовать коллекции и регулярные выражения для обработки текста.
* Избегать низкоуровневой работы с отдельными символами.

**Реализация:**

1. **Создание метода для извлечения первого слова**

Создан метод extractFirstWord, который извлекает первое слово из текста, используя регулярные выражения. Он обрабатывает предложения с любыми знаками препинания и возвращает слово в нижнем регистре.  
private static String extractFirstWord(String text) {  
 Matcher matcher = Pattern.*compile*("\\b[а-яА-Яa-zA-Z]+\\b").matcher(text);  
 return matcher.find() ? matcher.group().toLowerCase() : "";  
}

1. ****Сравнение слов с первым словом****  
   Создан метод findWordsWithSameLetters, который:

* Разделяет предложение на слова с помощью регулярного выражения.
* Сравнивает каждое слово с первым, проверяя совпадение их буквенного состав

private static List<String> findWordsWithSameLetters(String sentence, String sortedFirstWord, String firstWord) {  
 List<String> matchingWords = new ArrayList<>();  
 String[] words = sentence.split("[^а-яА-Яa-zA-Z]+");  
  
 for (String word : words) {  
 if (!word.isEmpty()) {  
 if (!word.equalsIgnoreCase(firstWord)) {  
 String sortedWord = *sortLetters*(word.toLowerCase());  
 if (sortedWord.equals(sortedFirstWord)) {  
 matchingWords.add(word);  
 }  
 }  
 }  
 }  
  
 return matchingWords;  
}

1. **Сортировка букв в словах**

Для определения слов, состоящих из тех же букв, реализован метод sortLetters. Он сортирует символы в слове и возвращает строку с отсортированными буквами.  
private static String sortLetters(String word) {  
 char[] letters = word.toCharArray();  
 Arrays.*sort*(letters);  
 return new String(letters);  
}

**4. **Основной метод программы****  
В классе Main реализован метод main, который:

* Принимает предложение.
* Извлекает первое слово и его буквенный состав.
* Находит и выводит слова, состоящие из тех же букв.

public class Main {  
 public static void main(String[] args) {  
 String sentence = "Тест для проверки работы программы с помощью слов: ттсе тсет сетт";  
 System.*out*.println("Исходное предложение: " + sentence);  
  
 String firstWord = *extractFirstWord*(sentence);  
 if (firstWord.isEmpty()) {  
 System.*out*.println("Первое слово не найдено.");  
 return;  
 }  
  
 String sortedFirstWord = *sortLetters*(firstWord.toLowerCase());  
 List<String> matchingWords = *findWordsWithSameLetters*(sentence, sortedFirstWord, firstWord);  
  
 System.*out*.println("Слова, состоящие из тех же букв, что и первое слово: " + matchingWords);  
 }

**Результат вывода программы:**



**Заключение:**

В ходе выполнения лабораторной работы я:

* Научился использовать регулярные выражения для извлечения и анализа слов в тексте.
* Изучил методы работы с коллекциями для поиска и хранения результатов.
* Реализовал сортировку символов в словах для сравнения их буквенных составов.

Программа успешно выполняет задачу по нахождению слов, состоящих из тех же букв, что и первое слово. Использование регулярных выражений и коллекций позволило упростить работу с текстом и повысить эффективность кода.

Данная работа продемонстрировала важность обработки строк и текста в программировании, а также предоставила опыт создания гибких решений для анализа текстовых данных.