**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 5

**З дисципліни:** *“Кросплатформне програмування”*

**На тему:** *“* *Використання регулярних виразів.”*

**Лектор:**

доц. каф. ПЗ

Дяконюк Л.М.

**Виконав:**

ст. гр. ПЗ-36

Балій М. І.

**Прийняв:**

ст. викл. каф. ПЗ

Баштовий А.В.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2023 р.

∑= \_\_\_\_\_

Львів – 2023

**Тема роботи:** Використання регулярних виразів.

**Мета роботи:** Навчитись використовувати регулярні вирази.

**TЕОРЕТИЧНІ ВІДОМОСТІ**

Регулярні вирази (regex) - це потужний механізм для роботи з рядками в текстових даних. В Java вони реалізовані через пакет java.util.regex. З допомогою регулярних виразів ви можете шукати, визначати та витягувати певні частини тексту за допомогою певних правил.

Основні елементи регулярних виразів в Java:

1. **Метасимволи:**
   * **.**: Позначає будь-який символ, окрім нового рядка.
   * **\***: Позначає нуль чи більше повторень попереднього символу.
   * **+**: Позначає один чи більше повторень попереднього символу.
   * **?**: Позначає нуль чи одне повторення попереднього символу.
2. **Класи символів:**
   * **[abc]**: Позначає будь-який символ з набору a, b або c.
   * **[^abc]**: Позначає будь-який символ, крім a, b або c.
3. **Кількість повторень:**
   * **{n}**: Точно n повторень попереднього елементу.
   * **{n,}**: Принаймні n повторень.
   * **{n,m}**: Від n до m повторень.
4. **Групи і зіставлення:**
   * **(...)**: Групує частину виразу.
   * **\n**: Зіставлення з n-ю групою.
5. **Предвизначені класи символів:**
   * **\d**: Десяткова цифра.
   * **\w**: Буквено-цифровий символ або підкреслення.
   * **\s**: Пробіл, табуляція, новий рядок.

**JUnit:**

JUnit - це фреймворк для тестування програм на мові Java. Він дозволяє розробникам створювати тестові набори для перевірки правильності роботи свого коду. Основні поняття в JUnit:

1. **Тестовий клас:**
   * Кожен клас тестів є підкласом junit.framework.TestCase.
   * Методи в тестовому класі мають починатися з префіксу "test", і вони викликаються фреймворком JUnit під час тестування.
2. **Анотації:**
   * **@Test**: Позначає, що метод є тестовим.
   * **@Before**: Позначає метод, який буде викликаний перед кожним тестом.
   * **@After**: Позначає метод, який буде викликаний після кожного тесту.
   * **@BeforeClass**: Позначає метод, який викликається один раз перед запуском тестового класу.
   * **@AfterClass**: Позначає метод, який викликається один раз після завершення тестового класу.
3. **Асерти:**
   * **assertEquals(expected, actual)**: Перевіряє, чи співпадають очікуване і фактичне значення.
   * **assertTrue(condition)**: Перевіряє, чи вираз є істинним.
   * **assertFalse(condition)**: Перевіряє, чи вираз є хибним.
4. **Тестові набори:**
   * Групування тестів в тестові набори для легшого виконання.
5. **Життєвий цикл тесту:**
   * **@Before** -> **@Test** -> **@After**: Цей цикл виконується для кожного тесту.

JUnit дозволяє автоматизувати тестування, забезпечуючи швидке виявлення помилок і збільшуючи впевненість в правильності коду.

**ЗАВДАННЯ**

Дано текст, який складається з багатьох стрічок. Речення можуть бути в кількох стрічках. Вважати, що відсутні перенесення слова.

1. Написати код до завдання з таблиці

2. Перевірку працездатності коду слід зробити з-за допомогою тестів. Систему тестів, які перекривають перевірку правильності виконання завдання, студент розробляє самостійно та демонструє в звіті лабораторної роботи.

3. Програма повинна передбачати можливість опрацювання ситуації, коли текст порожній, текст не містить лексем, запропонованих в завданнях та виведення результатів в зручній формі.

4. При реалізації завдання слід використати регулярні вирази

Варіант 3.

У всіх запитальних реченнях введеного тексту знайти і надрукувати без повторів слова заданої довжини. Текст і довжину можна ввести з консолі.

**ХІД ВИКОНАННЯ**

**Main.java**

package org.example;  
  
import java.util.HashSet;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Введіть текст:");  
 String inputText = scanner.nextLine();  
  
 System.*out*.println("Введіть довжину слів, які потрібно знайти:");  
 int wordLength = scanner.nextInt();  
 scanner.close();  
  
 TextAnalyzer textAnalyzer = new TextAnalyzer(inputText);  
 HashSet<String> questionSentences = textAnalyzer.findQuestionSentences();  
  
 WordFinder wordFinder = new WordFinder(wordLength);  
 HashSet<String> foundWords = wordFinder.findWordsInQuestions(questionSentences);  
  
 System.*out*.println("Знайдені слова заданої довжини в питальних реченнях без повторів:");  
 for (String foundWord : foundWords) {  
 System.*out*.println(foundWord);  
 }  
  
 }  
}

**TextAnalyzer.java**

package org.example;  
  
import java.util.HashSet;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
public class TextAnalyzer {  
 private final String inputText;  
  
 public TextAnalyzer(String inputText) {  
 this.inputText = inputText;  
 }  
  
 public HashSet<String> findQuestionSentences() {  
  
 String questionSentencePattern = "[A-Za-z\\s]\*\\?";  
 Pattern questionPattern = Pattern.*compile*(questionSentencePattern);  
  
 HashSet<String> questionSentences = new HashSet<>();  
 Matcher questionMatcher = questionPattern.matcher(inputText);  
  
 while (questionMatcher.find()) {  
 questionSentences.add(questionMatcher.group().trim());  
 }  
  
 return questionSentences;  
 }  
}

**WordFinder.java**

public class WordFinder {  
 private final int wordLength;  
  
 public WordFinder(int wordLength) {  
 this.wordLength = wordLength;  
 }  
  
 public HashSet<String> findWordsInQuestions(HashSet<String> questionSentences) {  
  
 String wordPattern = "\\b\\w{" + wordLength + "}\\b";  
 Pattern wordPatternRegex = Pattern.*compile*(wordPattern);  
  
 HashSet<String> foundWords = new HashSet<>();  
  
 for (String questionSentence : questionSentences) {  
 Matcher wordMatcher = wordPatternRegex.matcher(questionSentence);  
  
 while (wordMatcher.find()) {  
 foundWords.add(wordMatcher.group());  
 }  
 }  
  
 return foundWords;  
 }  
}

**TextAnalyzerTest.java**

**​**

import org.example.TextAnalyzer;  
import org.junit.Before;  
import org.junit.Test;  
  
import java.util.HashSet;  
  
import static org.junit.Assert.*assertEquals*;  
import static org.junit.Assert.*assertTrue*;  
  
public class TextAnalyzerTest {  
 private TextAnalyzer textAnalyzer;  
  
 @Before  
 public void setUp() {  
 textAnalyzer = new TextAnalyzer("This is a sample text. Is it working? Yes, it is.");  
 }  
  
 @Test  
 public void testFindQuestionSentences() {  
 HashSet<String> questionSentences = textAnalyzer.findQuestionSentences();  
  
 *assertEquals*(1, questionSentences.size());  
 *assertTrue*(questionSentences.contains("Is it working?"));  
 }  
  
 @Test  
 public void testFindQuestionSentences\_SingleQuestionSentence() {  
 TextAnalyzer analyzer = new TextAnalyzer("Is this a question sentence?");  
 HashSet<String> result = analyzer.findQuestionSentences();  
  
 HashSet<String> expected = new HashSet<>();  
 expected.add("Is this a question sentence?");  
  
 *assertEquals*(expected, result);  
 }  
  
 @Test  
 public void testFindQuestionSentences\_MultipleQuestionSentences() {  
 TextAnalyzer analyzer = new TextAnalyzer("What is this? How are you? Why is the sky blue?");  
 HashSet<String> result = analyzer.findQuestionSentences();  
  
 HashSet<String> expected = new HashSet<>();  
 expected.add("What is this?");  
 expected.add("How are you?");  
 expected.add("Why is the sky blue?");  
  
 *assertEquals*(expected, result);  
 }  
  
 @Test  
 public void testFindQuestionSentences\_NoQuestionSentences() {  
 TextAnalyzer analyzer = new TextAnalyzer("This is not a question. Neither is this.");  
 HashSet<String> result = analyzer.findQuestionSentences();  
  
 *assertTrue*(result.isEmpty());  
 }  
  
 @Test  
 public void testFindQuestionSentences\_EmptyInput() {  
 TextAnalyzer analyzer = new TextAnalyzer("");  
 HashSet<String> result = analyzer.findQuestionSentences();  
  
 *assertTrue*(result.isEmpty());  
 }  
  
 @Test  
 public void testFindQuestionSentences\_WithMultipleSpaces() {  
 TextAnalyzer analyzer = new TextAnalyzer(" Is this a question sentence? ");  
 HashSet<String> result = analyzer.findQuestionSentences();  
  
 HashSet<String> expected = new HashSet<>();  
 expected.add("Is this a question sentence?");  
  
 *assertEquals*(expected, result);  
 }  
  
 @Test  
 public void testFindQuestionSentences\_WithSpecialCharacters() {  
 TextAnalyzer analyzer = new TextAnalyzer("What is this?! How are you? This is not a question.");  
 HashSet<String> result = analyzer.findQuestionSentences();  
  
 HashSet<String> expected = new HashSet<>();  
 expected.add("What is this?");  
 expected.add("How are you?");  
  
 *assertEquals*(expected, result);  
 }  
  
 @Test  
 public void testFindQuestionSentences\_MixedCase() {  
 TextAnalyzer analyzer = new TextAnalyzer("Is this a QuEsTiOn SeNtEnCe? Why NoT? hOw AbOuT tHiS?");  
 HashSet<String> result = analyzer.findQuestionSentences();  
  
 HashSet<String> expected = new HashSet<>();  
 expected.add("Is this a QuEsTiOn SeNtEnCe?");  
 expected.add("Why NoT?");  
 expected.add("hOw AbOuT tHiS?");  
  
 *assertEquals*(expected, result);  
 }  
  
}

**WordFinderTest.java**

import org.example.TextAnalyzer;  
import org.example.WordFinder;  
import org.junit.Before;  
import org.junit.Test;  
  
import java.util.HashSet;  
  
import static org.junit.Assert.*assertEquals*;  
import static org.junit.Assert.*assertTrue*;  
  
public class WordFinderTest {  
 private WordFinder wordFinder;  
 private TextAnalyzer textAnalyzer;  
  
 @Before  
 public void setUp() {  
 textAnalyzer = new TextAnalyzer("This is a sample text. Is it working? Yes, it is.");  
 wordFinder = new WordFinder(2);  
 }  
  
 @Test  
 public void testFindWords() {  
 HashSet<String> questionSentences = textAnalyzer.findQuestionSentences();  
  
 HashSet<String> foundWords = wordFinder.findWordsInQuestions(questionSentences);  
  
 *assertEquals*(2, foundWords.size());  
 *assertTrue*(foundWords.contains("Is"));  
 *assertTrue*(foundWords.contains("it"));  
 }  
  
 @Test  
 public void testFindWordsInQuestions\_SingleWordMatch() {  
 HashSet<String> questionSentences = new HashSet<>();  
 questionSentences.add("What is your name?");  
 questionSentences.add("How old are you?");  
  
 WordFinder wordFinder = new WordFinder(2);  
 HashSet<String> result = wordFinder.findWordsInQuestions(questionSentences);  
  
 HashSet<String> expected = new HashSet<>();  
 expected.add("is");  
  
 *assertEquals*(expected, result);  
 }  
  
 @Test  
 public void testFindWordsInQuestions\_MultipleWordMatches() {  
 HashSet<String> questionSentences = new HashSet<>();  
 questionSentences.add("What is your name?");  
 questionSentences.add("How old are you?");  
 questionSentences.add("Where do you live?");  
  
 WordFinder wordFinder = new WordFinder(3);  
 HashSet<String> result = wordFinder.findWordsInQuestions(questionSentences);  
  
 HashSet<String> expected = new HashSet<>();  
 expected.add("How");  
 expected.add("are");  
 expected.add("old");  
 expected.add("you");  
  
 *assertEquals*(expected, result);  
 }  
  
 @Test  
 public void testFindWordsInQuestions\_NoWordMatches() {  
 HashSet<String> questionSentences = new HashSet<>();  
 questionSentences.add("Who is she?");  
 questionSentences.add("Where am I?");  
  
 WordFinder wordFinder = new WordFinder(6);  
 HashSet<String> result = wordFinder.findWordsInQuestions(questionSentences);  
  
 *assertTrue*(result.isEmpty());  
 }  
  
}

**РЕЗУЛЬТАТИ**

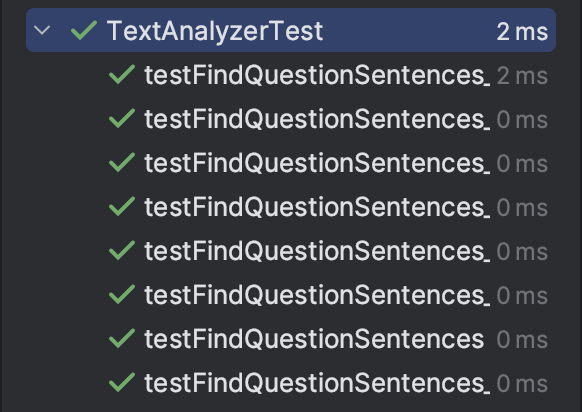


Рис. 1. Результат виконання тестів

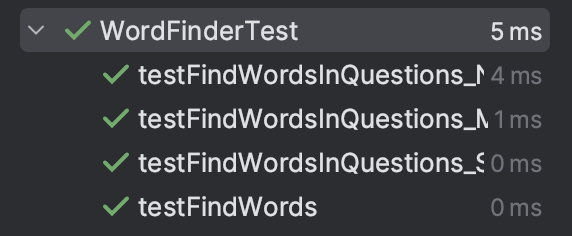


Рис. 2. Результат виконання тестів

**ВИСНОВКИ**

Під час виконання цієї лабораторної роботи я здобув навички роботи з текстовою інформацією, регулярними виразами, та пошуком лексем в тексті. Також навчився встановлювати пакети за допомогою maven. Написав різні варіанти тестів для пошуку питальних речень та пошуку слів заданої довжини в цих реченнях.