Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Об’єктно-орієнтовне програмування
Лабораторна робота №5

Виконала:
студентка групи ІО-25
Антоненко В.С.
Залікова книжка №2501

Перевірив
Алещенко О.В.

Київ 2023р.

Лабораторна робота №5

Тема: Відношення між класами в мові програмування Java.

Мета: Ознайомлення з відношеннями між класами в мові програмування Java. Здобуття навичок у використанні відношень між класів в мові програмування Java.

Завдання

1. Модифікувати лабораторну роботу №3 наступним чином: для літер, слів, речень, розділових знаків та тексту створити окремі класи. Слово повинно складатися з масиву літер, речення з масиву слів та розділових знаків, текст з масиву речень. Замінити послідовність табуляцій та пробілів одним пробілом.

2. Створити клас, який складається з виконавчого методу, що виконує описану дію з лабораторної роботи №3, але в якості типів використовує створені класи. Необхідно обробити всі виключні ситуації, що можуть виникнути під час виконання програмного коду. Всі змінні повинні бути описані та значення їх задані у виконавчому методі. Код повинен відповідати стандартам JCC та бути детально задокументований.

Роздруківка коду:

```
import java.util.HashSet;

public class Main {

    public static void main(String[] args) {
        String inputText = "Who are you? What is your name? Where are you
from?"; //вход даних, текст
        int targetLength = 3; //змінна, кількість літер в слові

        try {
            TextProcessor textProcessor = new TextProcessor();
            HashSet<Word> uniqueWords =
textProcessor.findUniqueWordsWithLength(inputText, targetLength);
            System.out.println("Unique words of length " + targetLength + "
in the input text are: " + uniqueWords);// вивод повідомлення, що містить
довжину та унікальні слова змінної
        } catch (Exception e) {
            System.out.println("An error occurred: " +
e.getMessage());//вивиод помилки
        }
    }
}

class Letter {
    private char value;

    public Letter(char value) {
        this.value = value;
    }

    public char getValue() {
        return value;
```

```
    }

    public void setValue(char value) {
        this.value = value;
    }
}

class Word {
    private Letter[] letters;

    public Word(Letter[] letters) {
        this.letters = letters;
    }

    public Letter[] getLetters() {
        return letters;
    }

    public void setLetters(Letter[] letters) {
        this.letters = letters;
    }

    public int getLength() {
        return letters.length;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        for (Letter letter : letters) {
            sb.append(letter.getValue());
        }
        return sb.toString();
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null || getClass() != obj.getClass()) {
            return false;
        }
        Word other = (Word) obj;
        return this.toString().equals(other.toString());
    }

    @Override
    public int hashCode() {
        return this.toString().hashCode();
    }
}

class Sentence {
    private Word[] words;
    private String punctuation;

    public Sentence(Word[] words, String punctuation) {
        this.words = words;
        this.punctuation = punctuation;
    }

    public Word[] getWords() {
        return words;
```

```
    }

    public void setWords(Word[] words) {
        this.words = words;
    }

    public String getPunctuation() {
        return punctuation;
    }

    public void setPunctuation(String punctuation) {
        this.punctuation = punctuation;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        for (Word word : words) {
            sb.append(word.toString()).append(" ");
        }
        sb.append(punctuation);
        return sb.toString();
    }
}

class Text {
    private Sentence[] sentences;

    public Text(Sentence[] sentences) {
        this.sentences = sentences;
    }

    public Sentence[] getSentences() {
        return sentences;
    }

    public void setSentences(Sentence[] sentences) {
        this.sentences = sentences;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        for (Sentence sentence : sentences) {
            sb.append(sentence.toString());
        }
        return sb.toString();
    }
}

class TextProcessor {
    public HashSet<Word> findUniqueWordsWithLength(String inputText, int
targetLength) {
        HashSet<Word> uniqueWords = new HashSet<>();
        String[] sentenceStrings = inputText.split("[?]");
        for (String sentenceString : sentenceStrings) {
            String[] wordStrings = sentenceString.trim().split("\\s+");
            for (String wordString : wordStrings) {
                wordString = wordString.replaceAll("[\\t\\s]+", " "); //
Заміна послідовності табуляцій та пробілів одним пробілом
                wordString = wordString.replaceAll("[^\\p{L}\\s]", ""); //
Вилучення розділових знаків
                wordString = wordString.toLowerCase(); // Перетворення на
нижній регістр
```

```
                wordString = wordString.replace("'", ""); // Вилучення
апострофів

                if (wordString.length() == targetLength) {
                    Letter[] letters = new Letter[wordString.length()];
                    for (int i = 0; i < wordString.length(); i++) {
                        letters[i] = new Letter(wordString.charAt(i));
                    }
                    Word word = new Word(letters);
                    uniqueWords.add(word);
                }
            }
        }
        return uniqueWords;
    }
}
```

Приклади роботи програми:

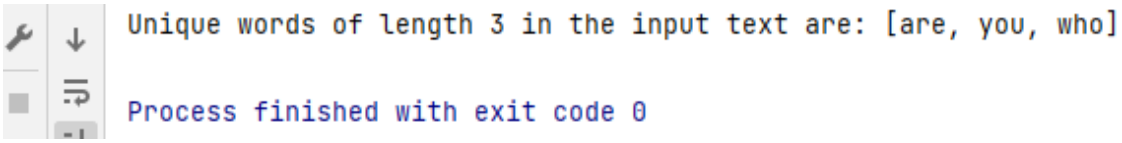

Рис.1

```
public static void main(String[] args) {
    String inputText = "Раз два три, літо прийди! По ООП сто балів захвати. ім'я запише без апострофу, ех";
    int targetLength = 3; //змінна, кількість літер в слові
```

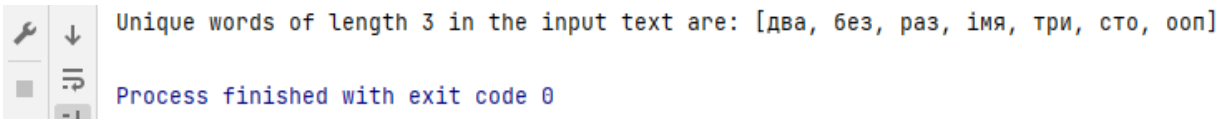

Рис.2

Алгоритм роботи програми:

1. Визначення вхідних даних:
    - Задання тексту у змінній *inputText*.
    - Задання кількості літер у слові у змінній *targetLength*.
2. Створення екземпляра *TextProcessor*.
3. Виклик методу *findUniqueWordsWithLength* з передачею тексту і кількості літер у якості аргументів:
    - Створення порожнього множини *uniqueWords* для зберігання унікальних слів заданої довжини.
    - Розбиття вхідного тексту на речення за допомогою роздільника "?". Результат зберігається у масиві *sentenceStrings*.
    - Перебір кожного речення у *sentenceStrings*:
    - Розбиття речення на слова за допомогою пробілів як роздільника. Результат зберігається у масиві *wordStrings*.
    - Обробка кожного слова у *wordStrings*:
    - Заміна послідовності табуляцій та пробілів одним пробілом.
    - Вилучення розділових знаків.
    - Перетворення на нижній регістр.
    - Вилучення апострофів.
    - Перевірка, чи довжина слова дорівнює *targetLength*.

- Створення масиву *letters* для зберігання об'єктів *Letter*, що представляють кожну літеру слова.
- Створення об'єкта Word на основі масиву letters.
- Додавання унікального слова до множини *uniqueWords*.
- Повернення множини *uniqueWords*.

4. Виведення результату:
   Виведення повідомлення, яке містить кількість літер та унікальні слова змінної *targetLength*.
5. Обробка виняткових ситуацій:

Якщо сталася помилка, виведення повідомлення про помилку.

Цей алгоритм використовує класи Letter, Word, Sentence і Text, щоб представити літери, слова, речення і текст відповідно. Клас TextProcessor містить метод findUniqueWordsWithLength, який виконує обробку тексту і повертає множину унікальних слів заданої довжини.

Висновок: під час виконання даної лабораторної роботи я ознайомилась з відношеннями між класами в мові програмування Java. Здобула навичок у використанні відношень між класів в мові програмування Java.