

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут
імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 5 з дисципліни
«Розробка програмного забезпечення на платформі Java»

«Відношення між класами в мові програмування Java»

Варіант 5

Виконав студент

ІІ-15, Буяло Дмитро Олександрович
(шифр, прізвище, ім'я, по батькові)

Перевірив

Ковальчук Олександр Миронович
(прізвище, ім'я, по батькові)

Київ 2023

Лабораторна робота 5

Відношення між класами в мові програмування Java

Мета – ознайомитися з відношеннями між класами в мові програмування Java.

Здобуття навичок у використанні відношень між класів в мові програмування Java.

Індивідуальне завдання

Варіант 5

Завдання

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.

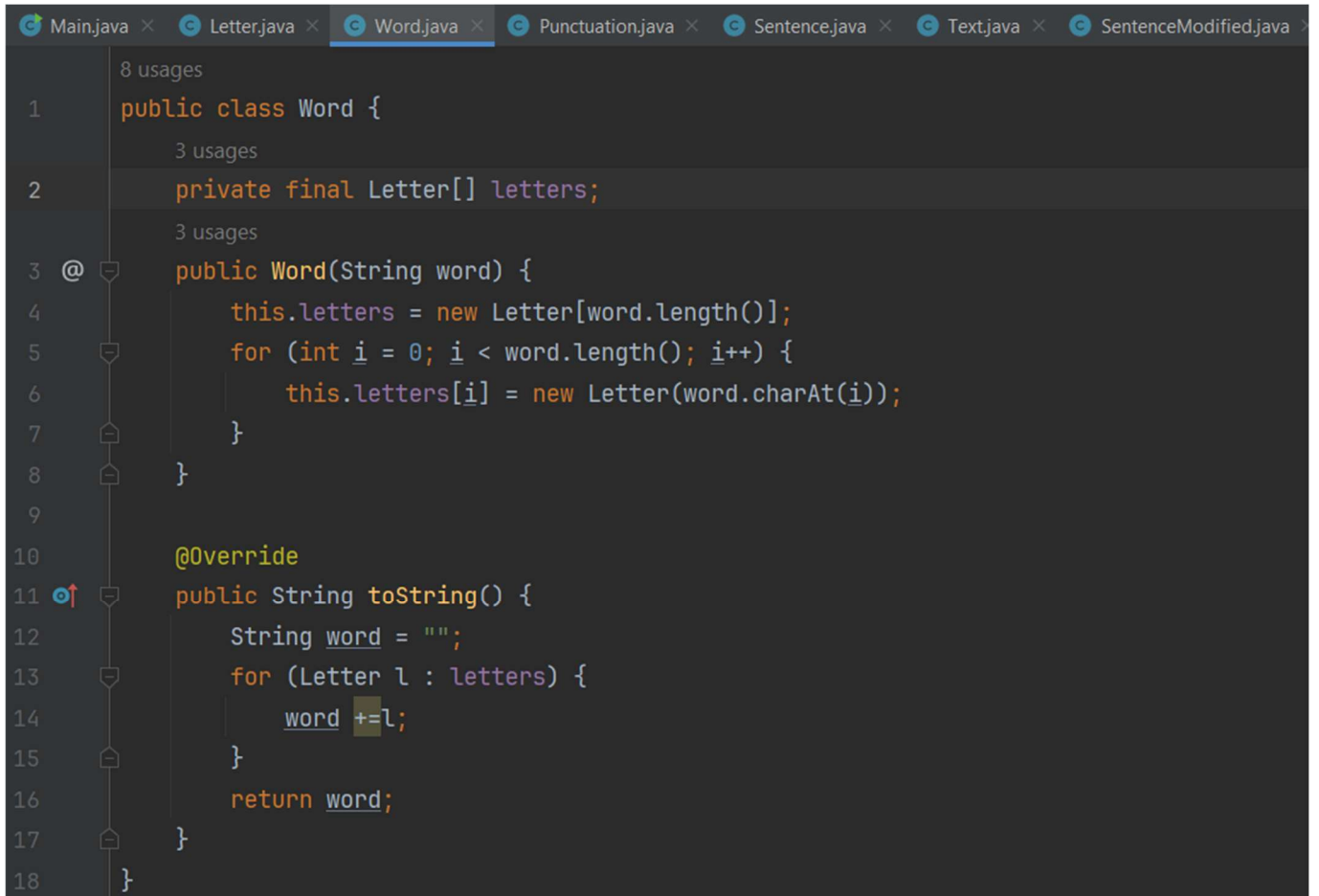
Створити клас, який складається з виконавчого методу, що **друкує слова без повторень заданого тексту в алфавітному порядку за першою літерою**, тип якого – **String**, але в якості типів використовує створені класи. Необхідно обробити всі виключні ситуації, що можуть виникнути під час виконання програмного коду. Всі змінні повинні бути описані та значення їх задані у виконавчому методі. Код повинен відповідати стандартам JCS та бути детально задокументований.

Програмна реалізація

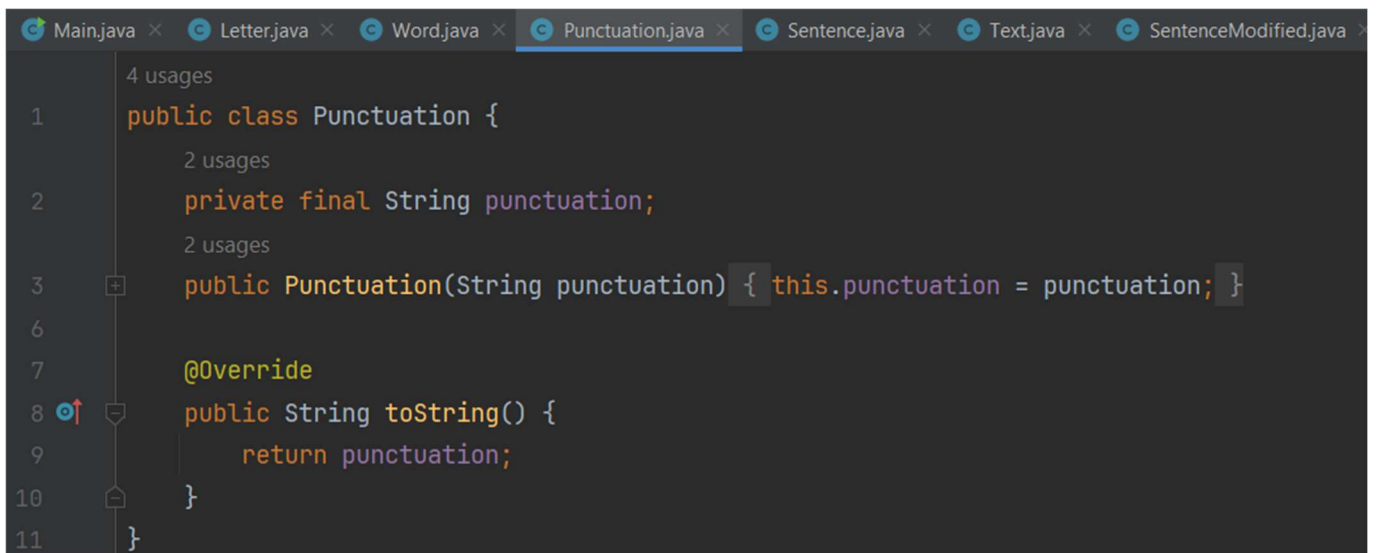


```
4 usages
1  public class Letter {
2      2 usages
3      private final char character;
4      1 usage
5      public Letter(char character) {
6          this.character = character;
7      }
8      @Override
9      public String toString() {
10         return Character.toString(character);
11     }
12 }
```

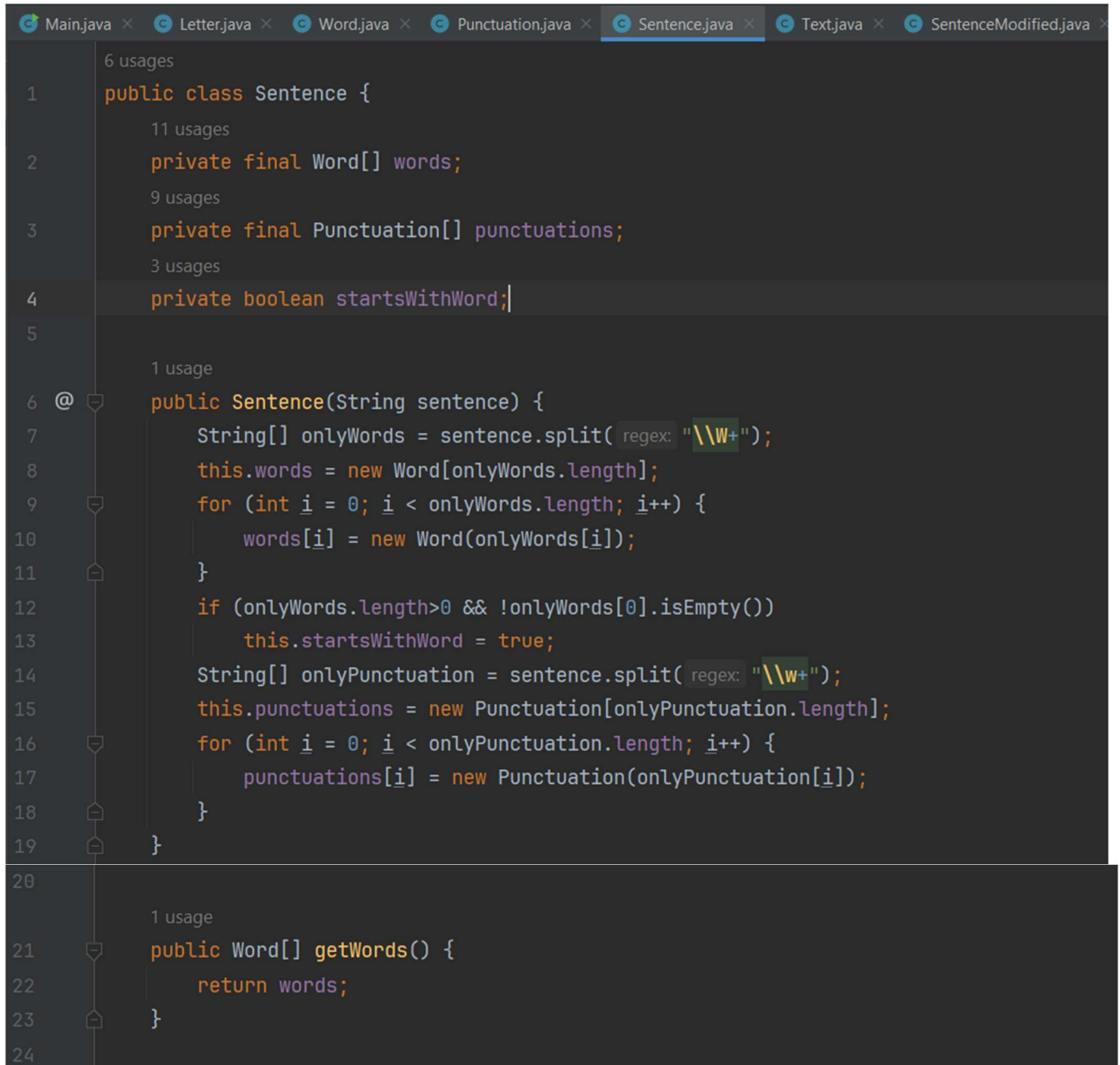
Розробка програмного забезпечення на платформі Java



```
1 8 usages
1 public class Word {
2     3 usages
3     private final Letter[] letters;
4     3 usages
5     @
6     public Word(String word) {
7         this.letters = new Letter[word.length()];
8         for (int i = 0; i < word.length(); i++) {
9             this.letters[i] = new Letter(word.charAt(i));
10        }
11    }
12
13    @Override
14    public String toString() {
15        String word = "";
16        for (Letter l : letters) {
17            word += l;
18        }
19        return word;
20    }
21 }
```

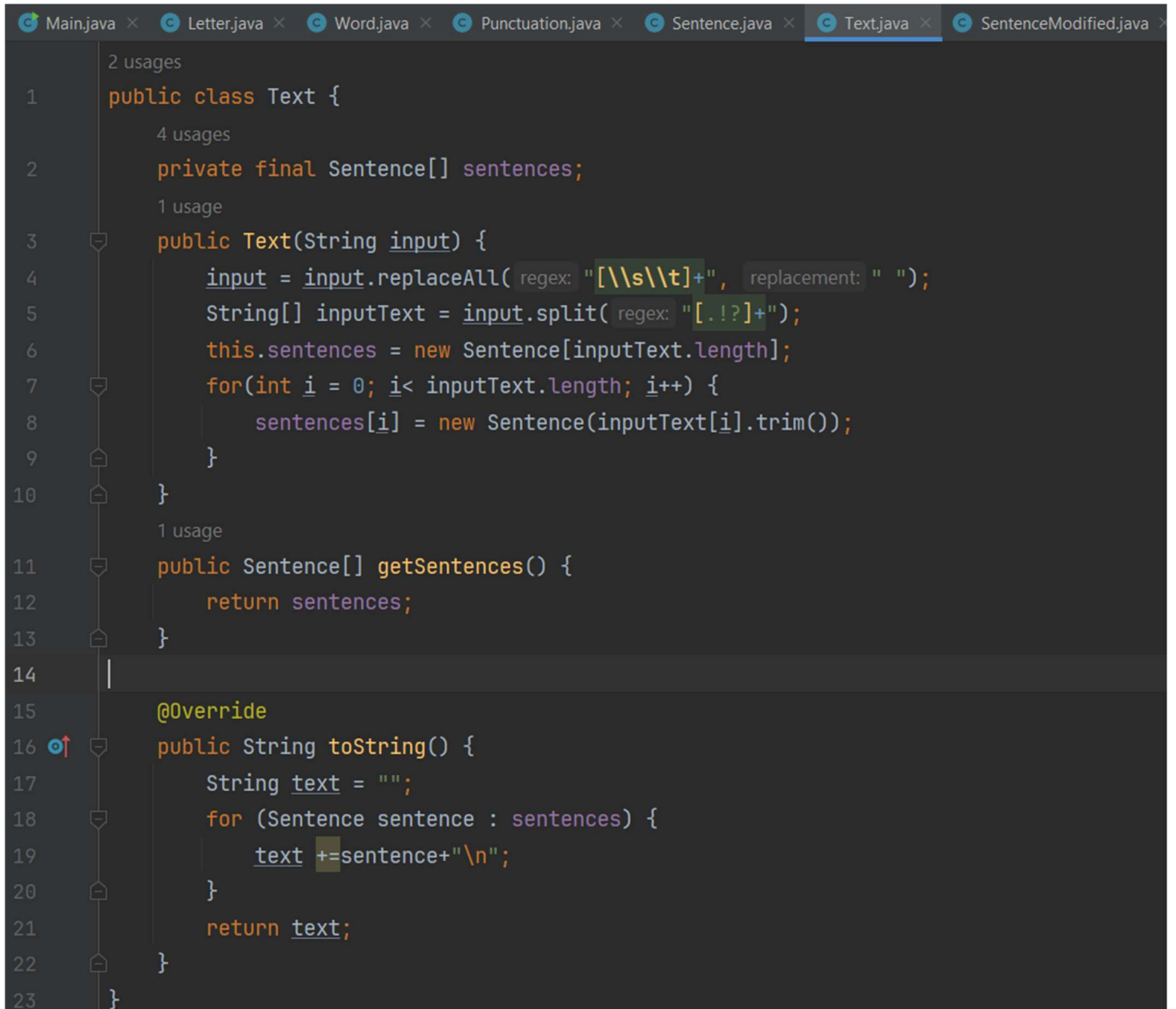


```
1 4 usages
1 public class Punctuation {
2     2 usages
3     private final String punctuation;
4     2 usages
5     public Punctuation(String punctuation) { this.punctuation = punctuation; }
6
7     @Override
8     public String toString() {
9         return punctuation;
10    }
11 }
```



```
1 6 usages
2 public class Sentence {
3     11 usages
4     private final Word[] words;
5     9 usages
6     private final Punctuation[] punctuations;
7     3 usages
8     private boolean startsWithWord;
9
10     1 usage
11     @ public Sentence(String sentence) {
12         String[] onlyWords = sentence.split(regex: "\\W+");
13         this.words = new Word[onlyWords.length];
14         for (int i = 0; i < onlyWords.length; i++) {
15             words[i] = new Word(onlyWords[i]);
16         }
17         if (onlyWords.length>0 && !onlyWords[0].isEmpty())
18             this.startsWithWord = true;
19         String[] onlyPunctuation = sentence.split(regex: "\\W+");
20         this.punctuations = new Punctuation[onlyPunctuation.length];
21         for (int i = 0; i < onlyPunctuation.length; i++) {
22             punctuations[i] = new Punctuation(onlyPunctuation[i]);
23         }
24     }
25
26     1 usage
27     public Word[] getWords() {
28         return words;
29     }
30 }
```

```
Main.java × Letter.java × Word.java × Punctuation.java × Sentence.java × Text.java × SentenceModified.java ×
25  @Override
26  public String toString() {
27      String sentence = "";
28      if (punctuations.length == 0) return words[0].toString();
29      if (words.length == 0) return punctuations[0].toString();
30      if (startsWithWord && punctuations.length == words.length) {
31          for (int i = 0; i < words.length; i++) {
32              sentence += punctuations[i];
33              sentence += words[i];
34          }
35          return sentence;
36      }
37      if (startsWithWord) {
38          for (int i = 1; i < punctuations.length; i++) {
39              sentence += words[i - 1];
40              sentence += punctuations[i];
41          }
42          return sentence;
43      }
44      for (int i = 1; i < words.length; i++) {
45          sentence += punctuations[i - 1];
46          sentence += words[i];
47      }
48      return sentence;
49  }
50 }
```



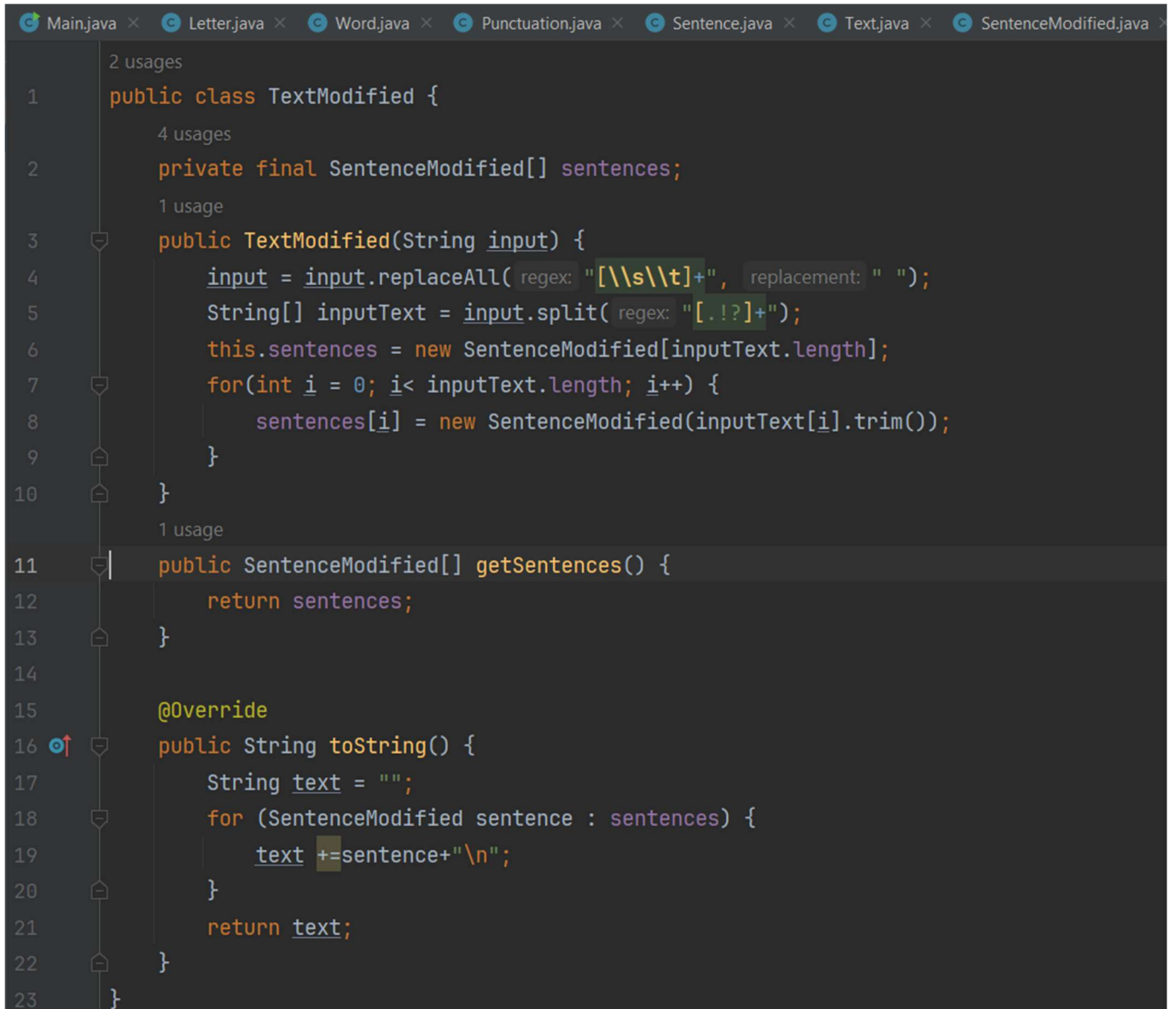
The screenshot shows an IDE with several tabs: Main.java, Letter.java, Word.java, Punctuation.java, Sentence.java, Text.java (selected), and SentenceModified.java. The Text.java file contains the following code:

```
1  public class Text {
2      private final Sentence[] sentences;
3      public Text(String input) {
4          input = input.replaceAll( regex: "[\\s\\t]+", replacement: " ");
5          String[] inputText = input.split( regex: "[.!?]+" );
6          this.sentences = new Sentence[inputText.length];
7          for(int i = 0; i< inputText.length; i++) {
8              sentences[i] = new Sentence(inputText[i].trim());
9          }
10     }
11     public Sentence[] getSentences() {
12         return sentences;
13     }
14
15     @Override
16     public String toString() {
17         String text = "";
18         for (Sentence sentence : sentences) {
19             text += sentence + "\n";
20         }
21         return text;
22     }
23 }
```

The code defines a `Text` class that takes a string `input` and splits it into sentences based on punctuation marks (`[.!?]`). It then returns an array of `Sentence` objects. The `toString` method overrides the default and concatenates the sentences with newline characters.

Розробка програмного забезпечення на платформі Java

```
Main.java x Letter.java x Word.java x Punctuation.java x Sentence.java x Text.java x SentenceModified.java x TextModif
1  import java.util.ArrayList;
2
3  6 usages
4  public class SentenceModified {
5
6  6 usages
7      private final ArrayList<Object> objects;
8
9  1 usage
10 @ public SentenceModified(String sentence) {
11     objects = new ArrayList<>();
12     String word = "";
13     for (int i = 0; i < sentence.length(); i++) {
14         if (Character.isLetterOrDigit(sentence.charAt(i))) word += sentence.charAt(i);
15         else {
16             if (!word.isEmpty()) objects.add(new Word(word));
17             if (!Character.isSpaceChar(sentence.charAt(i)))
18                 objects.add(new Punctuation(Character.toString(sentence.charAt(i))));
19             word = "";
20         }
21     }
22     if (!word.isEmpty()) objects.add(new Word(word));
23 }
24
25 1 usage
26 public ArrayList<Object> getObjects(){
27     return objects;
28 }
29
30 @Override
31 public String toString() {
32     String sentence = "";
33     for (Object s : objects) {
34         sentence += s + " ";
35     }
36     return sentence;
37 }
38 }
```

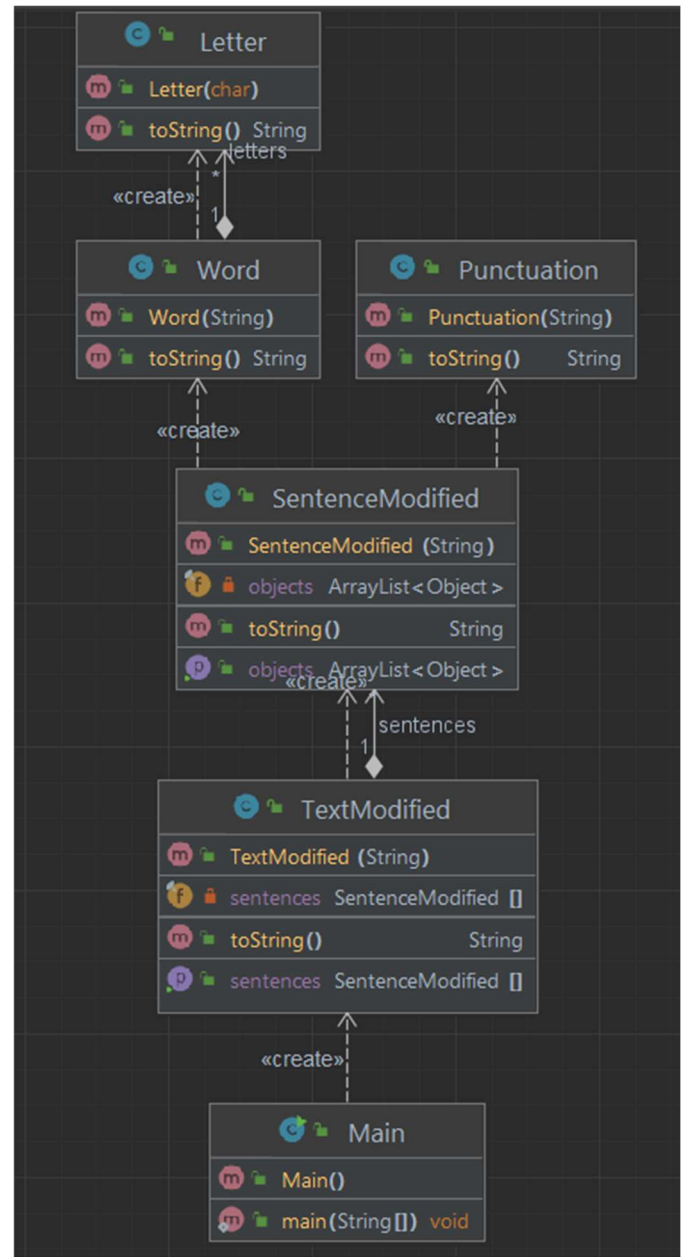
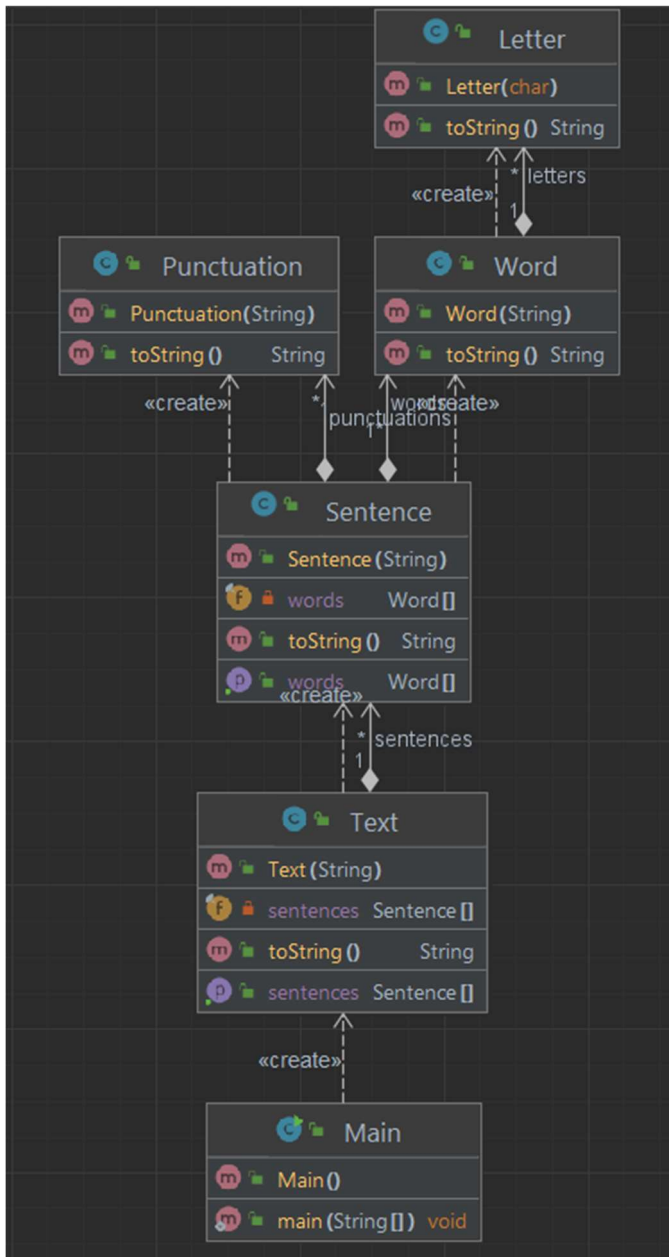


```
1 2 usages
1  public class TextModified {
2     4 usages
3     private final SentenceModified[] sentences;
4     1 usage
5     public TextModified(String input) {
6         input = input.replaceAll( regex: "[\\s\\t]+", replacement: " ");
7         String[] inputText = input.split( regex: "[.!?]+" );
8         this.sentences = new SentenceModified[inputText.length];
9         for(int i = 0; i< inputText.length; i++) {
10             sentences[i] = new SentenceModified(inputText[i].trim());
11         }
12     }
13     1 usage
14     public SentenceModified[] getSences() {
15         return sentences;
16     }
17
18     @Override
19     public String toString() {
20         String text = "";
21         for (SentenceModified sentence : sentences) {
22             text +=sentence+"\n";
23         }
24         return text;
25     }
26 }
```

The screenshot shows an IDE window with several tabs: Main.java, Letter.java, Word.java, Punctuation.java, Sentence.java, Text.java, and SentenceModified.java. The active file is TextModified.java. The code defines a class TextModified with a private final array of SentenceModified objects. The constructor takes a String input, removes all whitespace, splits the string into sentences based on punctuation, and initializes the sentences array. The getSences() method returns the sentences array. The toString() method overrides the default and returns a string representation of the sentences, each on a new line. The code is annotated with usage counts: 2 usages for the class, 4 usages for the sentences array, 1 usage for the constructor, and 1 usage for the getSences() method. The toString() method is annotated with @Override.


```
Main.java × Letter.java × Word.java × Punctuation.java × Sentence.java × Text.java × SentenceModified.java >
1  import java.util.Scanner;
2  import java.util.Set;
3  import java.util.TreeSet;
4
5  public class Main {
6  public static void main(String[] args) {
7      Scanner in = new Scanner(System.in);
8      System.out.print("Enter text: ");
9      String input = in.nextLine();
10
11     Text inputText = new Text(input);
12     System.out.println("\nYour entered text:");
13     System.out.println(inputText);
14
15     Set<String> uniqueWords = new TreeSet<>();
16     for (Sentence sentence : inputText.getSentences()) {
17         for (Word word : sentence.getWords()) {
18             if(!word.toString().isEmpty()) uniqueWords.add(word.toString());
19         }
20     }
21     System.out.print("In alphabet order:");
22     System.out.println(uniqueWords);
23
24     System.out.println("\nModified mode (spaces are not punctuation)");
25     TextModified inputTextModified = new TextModified(input);
26
27     System.out.println("\nYour entered text:");
28     System.out.println(inputTextModified);
29
30     Set<String> uniqueWordsModified = new TreeSet<>();
31     for (SentenceModified sentence : inputTextModified.getSentences()) {
32         for (Object object : sentence.getObjects()) {
33             if(object instanceof Word)
34                 uniqueWordsModified.add(object.toString());
35         }
36     }
37     System.out.print("In alphabet order:");
38     System.out.println(uniqueWordsModified);
39 }
40 }
```

UML діаграма класів



Приклад виконання коду

```
Enter text: r buyeho, r, r frf./ve khv\kw          khv\kw. e.gge. e.et. te !!! btr ! ll

Your entered text:
r buyeho, r, r frf
/ve khv\kw khv\kw
e
gge
e
et
te
btr
ll

In alphabet order:[btr, buyeho, e, et, frf, gge, khv\kw, ll, r, te, ve]
```

Modified mode (spaces are not punctuation)

```
Your entered text:
r buyeho , r , r frf
/ ve khv\kw khv\kw
e
gge
e
et
te
btr
ll

In alphabet order:[btr, buyeho, e, et, frf, gge, khv\kw, ll, r, te, ve]

Process finished with exit code 0
```

Висновок

Під час виконання п'ятої лабораторної роботи, розглянули на практиці роботу з класами та відношеннями між ними. В результаті була створена програма, яка відповідно до варіанту, друкує неповторюваний набір слів з тексту в алфавітному порядку двома варіантами виконання.