Міністерство освіти і науки України

Національний технічний університет України „КПІ”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки

інформації та управління

**ЗВІТ**

до лабораторної роботи № 4

з предмету:

„Основи технологій програмування”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Виконав**  **студент** |  | *ІП-61 Кушка Михайло Олександрович, 2-й курс, ІП-6116* |  |  |
|  |  | (№ групи, прізвище, ім’я, по батькові, курс, номер залікової книжки) |  |  |
|  |  |  |  |  |
| **Прийняв** |  | *Подрубайло О.О.* |  |  |
|  |  | (посада, прізвище, ім’я, по батькові ) |  |  |

Київ 2018

ЗМІСТ

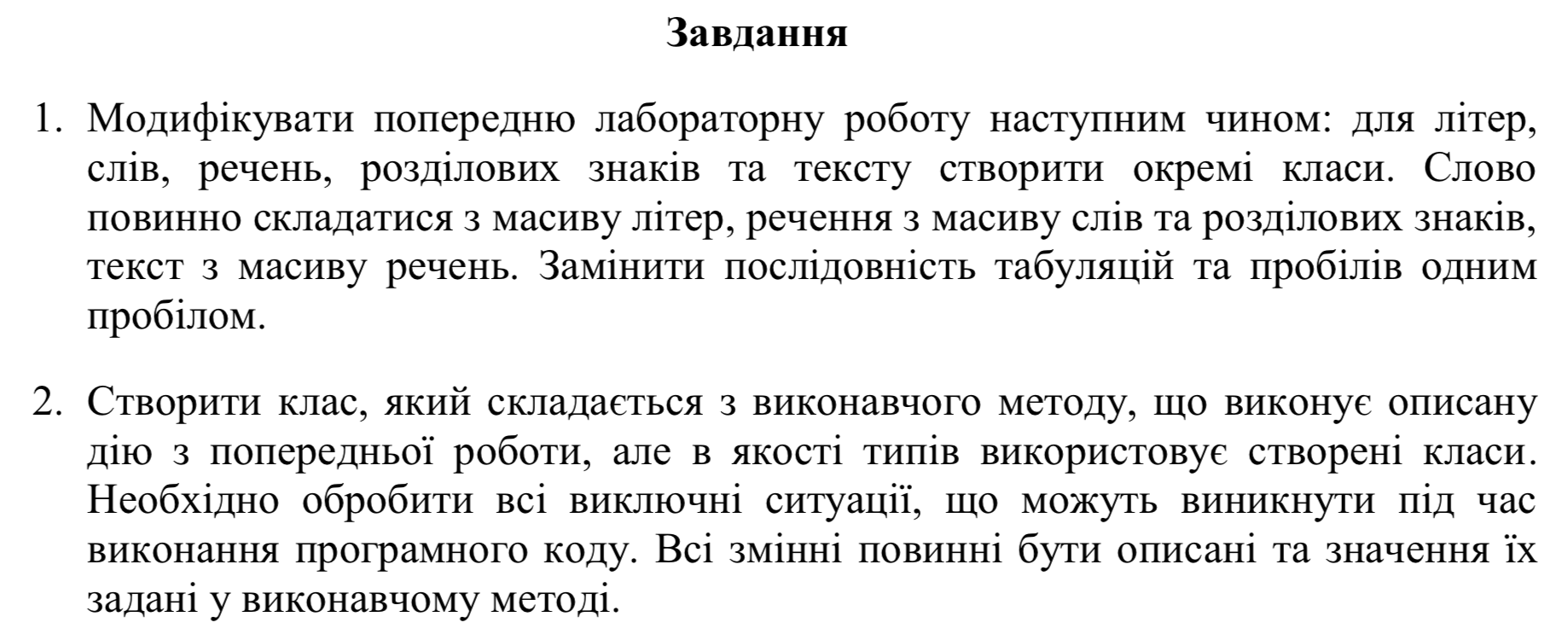
[1. постановка задачі 3](#_Toc508480230)

[2. Діаграма класів 4](#_Toc508480231)

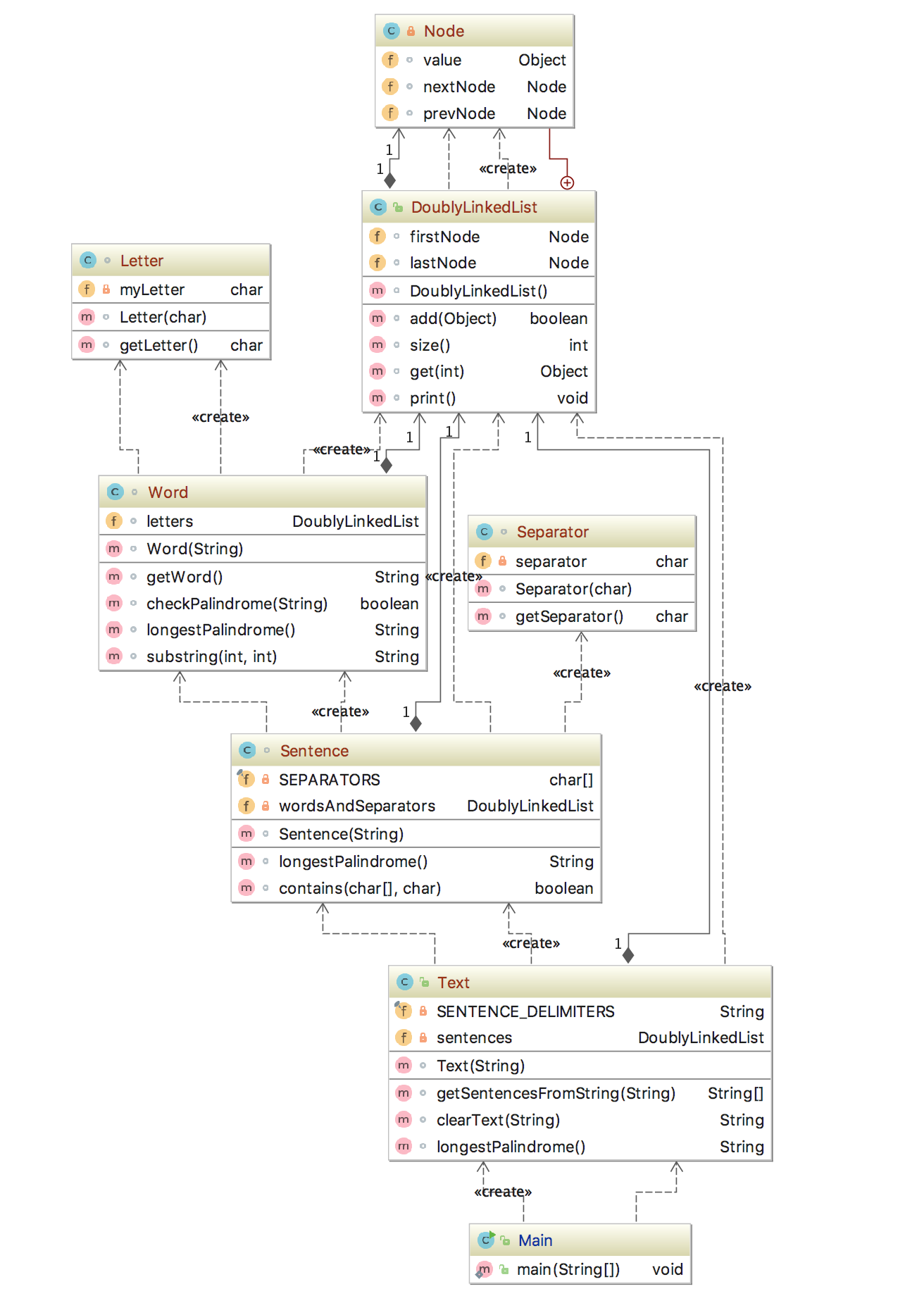
[3. Висновок 5](#_Toc508480232)

[4. Код програми 6](#_Toc508480233)

# постановка задачі



# Діаграма класів



# Висновок

Основою проблемою в даній лабораторній роботі було правильно розробити взаємозв’язок класів між собою та вирішити, як саме вони будуть реалізовані для подальшої коректної обробки тексту.

# Код програми

*/\*\*  
 \* Java labs - Lab4  
 \** ***@version*** *1.2 2018-03-10  
 \** ***@author*** *Misha Kushka  
 \*/  
  
/\*\*  
 \* Simple letter  
 \*/***class** Letter {  
 **private char myLetter**;  
  
 Letter(**char** newLetter) {  
 **myLetter** = newLetter;  
 }  
  
 *// Returns single letter* **char** getLetter() {  
 **return myLetter**;  
 }  
}  
  
*/\*\*  
 \* Separator in sentence  
 \*/***class** Separator {  
 **private char separator**;  
  
 Separator(**char** newSeparator) {  
 **separator** = newSeparator;  
 }  
  
 *// Returns separator* **char** getSeparator() {  
 **return separator**;  
 }  
}  
  
*/\*\*  
 \* Word, made from letters  
 \*/***class** Word {  
 DoublyLinkedList **letters** = **new** DoublyLinkedList();  
  
 Word(String s) {  
 **for** (**char** letter : s.toCharArray()) {  
 **letters**.add(**new** Letter(letter));  
 }  
 }  
  
 *// Returns whole word* String getWord() {  
 String result = **""**;  
  
 **for** (**int** i = 0; i < **letters**.size(); ++i) {  
 result = result + ((Letter)**letters**.get(i)).getLetter();  
 }  
  
 **return** result;  
 }  
  
 *// Checks is current string a palindrome* **boolean** checkPalindrome(String s) {  
 **int** n = s.length();  
 **for** (**int** i = 0; i < n / 2; ++i) {  
 **if** (s.charAt(i) != s.charAt(n-i-1))  
 **return false**;  
 }  
  
 **return true**;  
 }  
  
 *// Get the longest palindrome in the word* String longestPalindrome() {  
 **int** left = 0;  
 **final int** right = **letters**.size();  
 **int** j = right;  
 String temp = **""**;  
 String theLongest = **""**;  
  
 **while** (left < right-1) {  
 temp = substring(left, j);  
 j -= 1;  
  
 **if** (**this**.checkPalindrome(temp) && theLongest.length() < temp.length()) {  
 theLongest = temp;  
 }  
  
 **if** (j < left+2) {  
 left += 1;  
 j = right;  
 }  
 }  
  
 **return** theLongest;  
 }  
  
 *// Get substring from list of Letters objects* String substring(**int** a, **int** b) {  
 String result = **""**;  
  
 **for** (**int** i = a; i < b; ++i) {  
 result = result + ((Letter)**letters**.get(i)).getLetter();  
 }  
  
 **return** result;  
 }  
}  
  
*/\*\*  
 \* Sentence with words and separators  
 \*/***class** Sentence {  
 **private final char**[] **SEPARATORS** = {**','**, **';'**, **'-'**};  
 **private** DoublyLinkedList **wordsAndSeparators** = **new** DoublyLinkedList();  
  
 *// Put all words in the Word class and all separators in the Separator class* Sentence(String s) {  
 String tempWord = **""**;  
  
 **for** (**char** symbol : s.toCharArray()) {  
 **if** (contains(**SEPARATORS**, symbol)) {  
 **wordsAndSeparators**.add(**new** Separator(symbol));  
 **if** (tempWord.length() != 0) {  
 **wordsAndSeparators**.add(**new** Word(tempWord));  
 tempWord = **""**;  
 }  
 } **else** {  
 tempWord = tempWord + symbol;  
 }  
 }  
  
 **if** (tempWord.length() != 0) {  
 **wordsAndSeparators**.add(**new** Word(tempWord));  
 }  
 }  
  
 *// Get the longest palindrome in the sentence* String longestPalindrome() {  
 String theLongest = **""**;  
  
 **for** (**int** i = 0; i < **wordsAndSeparators**.size(); ++i) {  
 **if** (**wordsAndSeparators**.get(i).getClass().equals(Word.**class**)) {  
 Word temp = (Word)**wordsAndSeparators**.get(i);  
 **if** (temp.longestPalindrome().length() > theLongest.length()) {  
 theLongest = temp.longestPalindrome();  
 }  
 }  
 }  
  
 **return** theLongest;  
 }  
  
 *// Check is array contains an element* **boolean** contains(**char**[] arr, **char** element) {  
 **for** (**char** a : arr) {  
 **if** (a == element)  
 **return true**;  
 }  
  
 **return false**;  
 }  
}  
  
*/\*\*  
 \* Stores and processes the text  
 \*/***class** Text {  
 **private final** String **SENTENCE\_DELIMITERS** = **"\\. |\\! |\\? |\\.|\\!|\\?"**;  
 **private** DoublyLinkedList **sentences** = **new** DoublyLinkedList();  
  
 Text(String text) {  
 *// Check whether the input string is empty* **if** (text.length() == 0) {  
 System.***err***.println(**"Input text is empty"**);  
 System.*exit*(2);  
 }  
  
 String[] stringSentences = getSentencesFromString(text);  
  
 **for** (String sentence : stringSentences) {  
 System.***out***.println(**"Sentence: "** + sentence);  
 **sentences**.add(**new** Sentence(sentence));  
 }  
 }  
  
 *// Split string by separators* String[] getSentencesFromString(String s) {  
 s = clearText(s);  
 String[] data = s.split(**SENTENCE\_DELIMITERS**);  
  
 **return** data;  
 }  
  
 *// Clear the text from tabs and spaces duplicates* String clearText(String s) {  
 s = s.replaceAll(**"\t"**, **" "**);  
 s = s.replaceAll(**" +"**, **" "**);  
  
 **return** s;  
 }  
  
 *// Get the longest palindrome in the text* String longestPalindrome() {  
 String theLongest = **""**;  
  
 **for** (**int** i = 0; i < **sentences**.size(); ++i) {  
 **if** (((Sentence)**sentences**.get(i)).longestPalindrome().length() > theLongest.length()) {  
 theLongest = ((Sentence)**sentences**.get(i)).longestPalindrome();  
 }  
 }  
  
 **return** theLongest;  
 }  
}  
  
*/\*\*  
 \* Implementation of Doubly linked list  
 \* to store objects of different types  
 \*/***class** DoublyLinkedList {  
  
 *// Subsidiary class to implement Doubly linked list* **private class** Node {  
 Object **value**;  
 Node **nextNode**;  
 Node **prevNode**;  
 };  
  
 Node **firstNode**;  
 Node **lastNode**;  
  
 DoublyLinkedList() {  
 **firstNode** = **null**;  
 **lastNode** = **null**;  
 }  
  
 *// Add element to the end of the list* **boolean** add(Object value) {  
 Node currNode = **lastNode**;  
 **lastNode** = **new** Node();  
 **if** (**lastNode** != **null**) {  
 **if**(**firstNode** == **null**)  
 **firstNode** = **lastNode**;  
 **lastNode**.**value** = value;  
 **lastNode**.**nextNode** = **null**;  
 **lastNode**.**prevNode** = currNode;  
 **if**(currNode != **null**)  
 currNode.**nextNode** = **lastNode**;  
 **return true**;  
 }  
  
 **return false**;  
 }  
  
 *// Get size of the list* **int** size() {  
 **int** size = 0;  
 Node currNode = **firstNode**;  
  
 **while** (currNode != **null**) {  
 size++;  
 currNode = currNode.**nextNode**;  
 }  
  
 **return** size;  
 }  
  
 *// Get i-th element from the list* Object get(**int** i) {  
 **int** position = 0;  
 Node currNode = **firstNode**;  
  
 **if** (i >= **this**.size()) {  
 System.***err***.println(**"Too big index. Try smaller one."**);  
 System.*exit*(1);  
 }  
  
 **while** (currNode != **null**) {  
 **if** (position == i) {  
 **return** currNode.**value**;  
 }  
 currNode = currNode.**nextNode**;  
 position++;  
 }  
  
 **return** currNode.**value**;  
 }  
  
 *// Prints list on the screen* **void** print() {  
 Node currNode = **firstNode**;  
  
 **while** (currNode != **null**) {  
 System.***out***.println(currNode.**value**);  
 currNode = currNode.**nextNode**;  
 }  
 }  
};  
  
**public class** Main {  
  
 **public static void** main(String[] args) {  
  
 String s = **"abba, cool sool;helloljleh. What? Blue space!bcb?"**;  
 Text text = **new** Text(s);  
 String theLongest = text.longestPalindrome();  
  
 System.***out***.println(**"The longest palindrome: "** + theLongest);  
  
 }  
}