НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №4**

з дисципліни **«**Основи технологій програмування**»**

Варіант: «6412»

Виконав:

студент 2 курсу

група ІП-64

Лейзьо Сергій Іванович

Перевірив:

Подрубайло Олександр Олександрович

Київ – 2018

**Завдання**

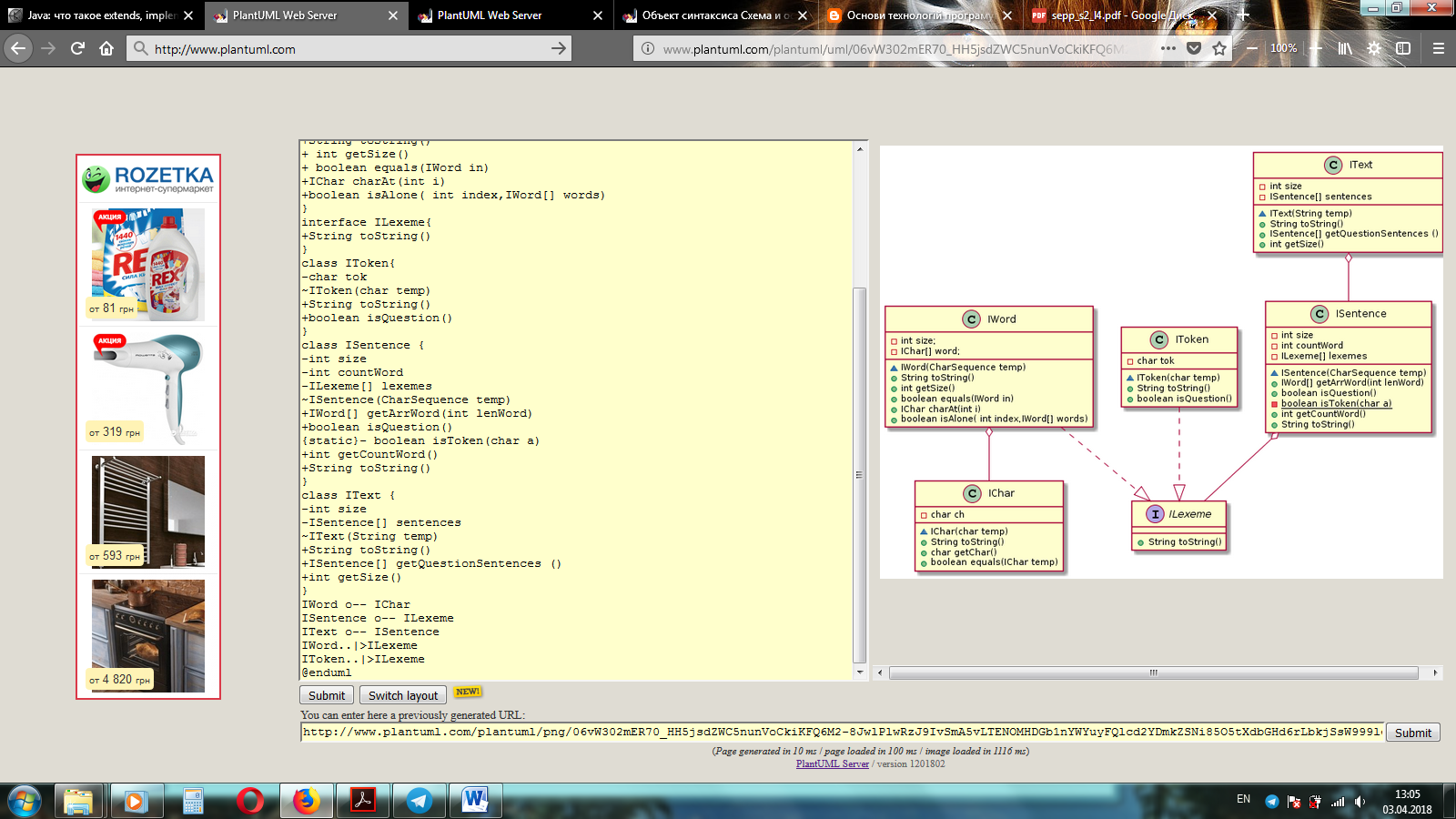
**В усіх питальних реченнях заданого тексту знайти та надрукувати без повторень слова заданої довжини.**

Модифікувати попередню лабораторну роботу 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.

**Код програми**

package com.company;  
  
import java.util.Arrays;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
  
 Scanner in = new Scanner(System.*in*);  
 IText buf = new IText(in.nextLine());  
 ISentence[] sentences = buf.getQuestionSentences();  
 System.*out*.println("Задайте количество букв в слове: ");  
 int wordlenght = in.nextInt();  
 IWord[] mainArr = new IWord[10];  
 int count = 0;  
 for(ISentence sentence: sentences) {  
 IWord[] words = sentence.getArrWord(wordlenght);  
 for(IWord word : words) {  
 mainArr[count] = word;  
 ++count;  
 if (count == words.length - 1) {  
 mainArr = Arrays.*copyOf*(mainArr, 2 \* mainArr.length);  
 }  
 }  
 }  
 IWord[] newArr = new IWord[count];  
 System.*arraycopy*(mainArr,0,newArr,0,count);  
  
 for(int i = 0 ; i < newArr.length; ++i) {  
 if(newArr[i].isAlone(i,newArr)) {  
 System.*out*.println(newArr[i]);  
 }  
  
 }  
 }  
}  
  
class IChar {  
 private char ch;  
 IChar(char temp) {  
 ch = temp;  
 }  
  
 @Override  
 public String toString() {  
 return String.*valueOf*(ch);  
 }  
  
 public char getChar() {  
 return ch;  
 }  
  
  
 public boolean equals(IChar temp) {  
 return ch == temp.getChar();  
 }  
  
}  
  
class IWord implements ILexeme{  
 private int size;  
 private IChar[] word;  
  
 IWord(CharSequence temp) {  
 size = temp.length();  
 word = new IChar[size];  
 for (int i = 0; i < size; i++) {  
 word[i] = new IChar(temp.charAt(i));  
 }  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder temp = new StringBuilder();  
 for (int i = 0; i < size; ++i) {  
 temp.append(word[i].toString());  
 }  
 //System.out.println(temp);  
 return temp.toString();  
 }  
  
 public int getSize() {  
 return size;  
 }  
  
 public boolean equals(IWord in) {  
 int count = 0;  
 for(int i = 0; i < size; ++i) {  
 if(word[i].equals(in.charAt(i))) {  
 ++count;  
 }  
 }  
 return count == size;  
 }  
  
 public IChar charAt(int i){  
 return word[i];  
 }  
  
  
 public boolean isAlone( int index,IWord[] words) {  
 int count = 0;  
 for(int i = index; i < words.length; ++i) {  
 if(words[index].equals(words[i])){  
 count++;  
 }  
 }  
 // System.out.println(count);  
 return count == 1;  
 }  
  
  
}  
  
interface ILexeme {  
 public String toString();  
}  
  
class IToken implements ILexeme {  
 private char tok;  
 IToken(char temp) {  
 tok = temp;  
 }  
  
 public String toString() {  
 return String.*valueOf*(tok) ;  
 }  
  
 public boolean isQuestion() {  
 return tok == '?';  
 }  
}  
  
class ISentence {  
 private int size = 0;  
 private int countWord = 0;  
 private ILexeme[] lexemes;  
  
 ISentence(CharSequence temp) {  
 lexemes = new ILexeme[10];  
 int i = 0;  
 while ( i < temp.length()) {  
 if(*isToken*(temp.charAt(i))) {  
 lexemes[size++] = new IToken(temp.charAt(i));  
 ++i;  
 }  
 else {  
 int offset = 0;  
 while(i+offset < temp.length() &&!*isToken*(temp.charAt(i+offset))) {  
 ++offset;  
 }  
 lexemes[size++] = new IWord(temp.subSequence(i, i+offset));  
 i+= offset;  
 ++countWord;  
 }  
 if(lexemes.length == size) {  
 lexemes = Arrays.*copyOf*(lexemes, 2 \* size);  
 }  
 }  
 ILexeme[] newArr = new ILexeme[size];  
 System.*arraycopy*(lexemes,0,newArr, 0, size);  
 lexemes = newArr;  
 }  
  
 public IWord[] getArrWord(int lenWord) {  
 IWord[] words = new IWord[countWord];  
 int i = 0;  
 for(ILexeme lex: lexemes) {  
 if(lex instanceof IWord) {  
 if(((IWord)lex).getSize() == lenWord ) {  
 words[i] = (IWord) lex;  
 ++i;  
 }  
 }  
 }  
 IWord[] newArr = new IWord[i];  
 System.*arraycopy*(words,0,newArr, 0, i);  
 words = newArr;  
 return words;  
 }  
  
 public boolean isQuestion() {  
 return ((IToken)lexemes[lexemes.length - 1]).isQuestion();  
 }  
  
  
 private static boolean isToken(char a) {  
 return (a == '.' || a == '?' || a == '!' || a == ',' || a == ' ');  
 }  
  
 public int getCountWord() {  
 return countWord;  
 }  
  
  
 public String toString() {  
 StringBuilder out = new StringBuilder();  
 for(ILexeme el : lexemes) {  
 out.append((el.toString()));  
 }  
 return out.toString();  
 }  
}  
  
class IText {  
 private int size = 0;  
 private ISentence[] sentences;  
  
 IText(String temp) {  
  
 for(int i = 0; i < temp.length(); ++i) {  
 if(temp.charAt(i) == '!' || temp.charAt(i) == '?' || temp.charAt(i) == '.' ) {  
 ++size;  
 }  
 }  
 int prevPos = 0;  
 sentences = new ISentence[size];  
 int count = 0;  
 for(int i = 0; i < temp.length(); ++i) {  
  
 if(temp.charAt(i) == '!' || temp.charAt(i) == '?' || temp.charAt(i) == '.' ) {  
  
 sentences[count++] = new ISentence(temp.subSequence(prevPos,i+1));  
 prevPos = i + 1;  
 }  
 }  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder out = new StringBuilder();  
 for(ISentence s: sentences) {  
 out.append(s);  
 }  
 return out.toString();  
 }  
  
 public ISentence[] getQuestionSentences () {  
 int count = 0;  
 for(ISentence sen: sentences) {  
 if(sen.isQuestion()) {  
 ++count;  
 }  
 }  
 ISentence[] arr = new ISentence[count];  
 for (ISentence sen: sentences) {  
 if(sen.isQuestion()) {  
 arr[--count] = sen;  
 }  
 }  
 return arr;  
 }  
  
 public int getSize() {  
 return size;  
 }  
}

**Діаграма класів**



**Висновок**

У готовій лабораторній роботі програма знаходить питальні речення, оброблює їх таким чином: Виводяться слова без повторень, кількість букв яких, вказується на початку. Програма працює коректно, всі слова виводяться правильно (не повторюються).

**Для перевірки використовувався текст:**

Яблуко апельсин мандарин кокос, банан, банан? вишня! арбуз. кокос, батон, флако? "a `n, ;б: в- с? a a?