Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

Кафедра моделювання систем і технологій

**ЛАБОРАТОРНА РОБОТА №2**

з дисципліни «Об’єктно-орієнтоване програмування»

Тема: Робота з текстовими файлами та обробка даних строкового типу

Виконав студент 2 курсу

Групи КС-21

Клочко Андрій Володимирович

Перевірив:

к.т.н., доц. Нагорний К.А.

Харків – 2020

# ВСТУП

**Тема:** Розробка додатку для обробки строк тексту отриманого із текстового файлу.

**Мета роботи:** Набути базові навики роботи з текстовими файлами, такі як: читання вмісту файлу, запис обробленого тексту до файлу. Набути навики обробки даних строкового типу.

**Завдання:** дано два текстових фрагменти, що містяться у двох текстових файлах. Реалізувати зчитування тесту із вхідних файлів, обробку отриманого тексту, та запис результатів обробки у вихідні файли.

Формат вихідних файлів – HEADER – BODY. Де HEADER містить опис строкових перетворень. BODY містить в собі фрагменти тексту (ів), отриманих в результаті модифікації строк вхідного тексту.

# РОЗДІЛ 1 РЕЗУЛЬТАТИ РОБОТИ

## 1.1 UML – діаграма класів

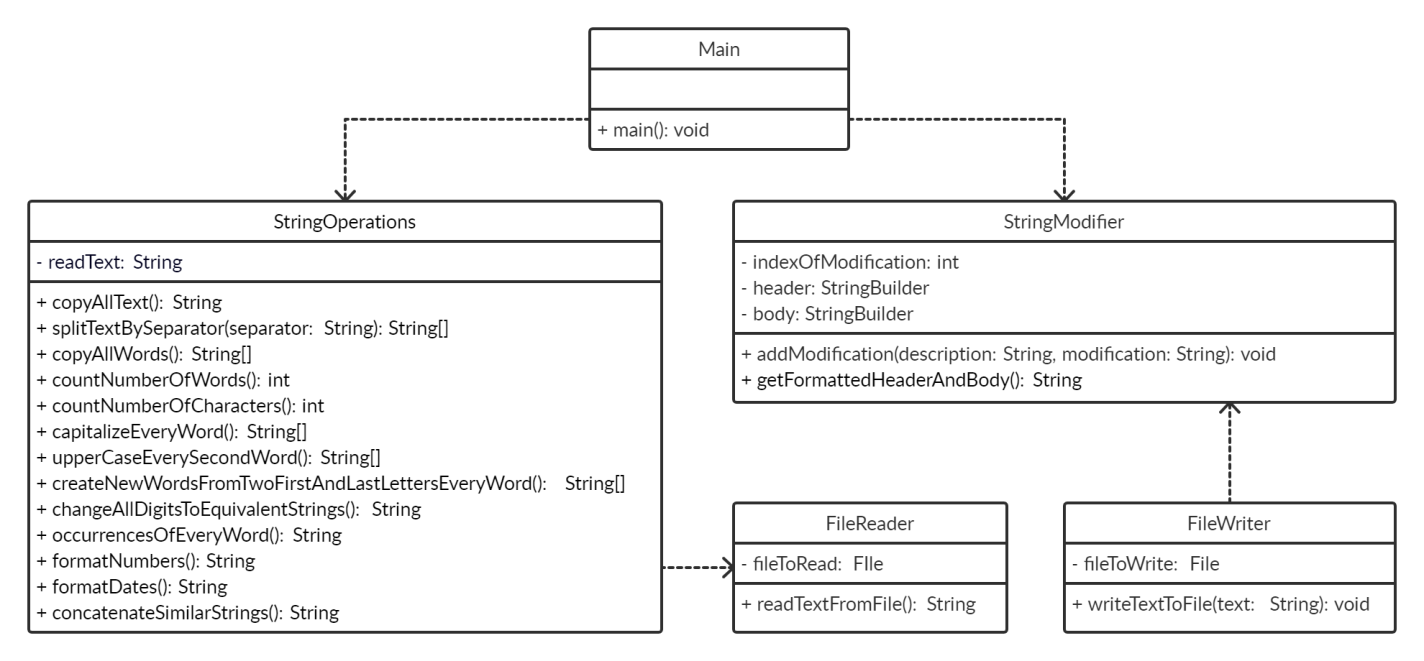


Рис. 1 – UML-діаграма класів

1.2 Лістинг вихідного коду

Лістинг класу «StringOperations»

**import** org.apache.commons.lang3.CharUtils;  
**import** org.apache.commons.lang3.StringUtils;  
**import** org.apache.commons.lang3.math.NumberUtils;  
  
**import** java.text.DecimalFormat;  
**import** java.text.NumberFormat;  
**import** java.time.LocalDate;  
**import** java.time.format.DateTimeFormatter;  
**import** java.util.\*;  
  
**public class** StringOperations {  
  
 **private final** String **text**;  
  
 **public** StringOperations(String text){  
 **this**.**text** = text;  
 }  
  
 **public** String copyAllText(){  
 **return text**;  
 }  
  
 **public** String[] splitTextBySeparator(String separator){  
 **return** StringUtils.*split*(**text**, separator);  
 }  
  
 **public** String[] copyAllWords(){  
 **return** getAllWordsFromTextWithoutSymbols(**text**);  
 }  
  
 **public int** countNumberOfWords(){  
 **return** getAllWordsFromTextWithoutWhitespace().**length**;  
 }  
  
 **public int** countNumberOfCharacters(){  
 **return text**.length();  
 }  
  
 **public** String[] capitalizeEveryWord(){  
 String[] wordsWithoutWhitespace = getAllWordsFromTextWithoutWhitespace();  
  
 **for** (**int** i=0 ; i<wordsWithoutWhitespace.**length** ; i++) {  
 wordsWithoutWhitespace[i] = StringUtils.*capitalize*(wordsWithoutWhitespace[i]);  
 }  
 **return** wordsWithoutWhitespace;  
 }  
  
 **public** String[] upperCaseEverySecondWord(){  
 String[] wordsWithoutWhitespace = getAllWordsFromTextWithoutWhitespace();  
  
 **for** (**int** i = 1; i <= wordsWithoutWhitespace.**length**; i++) {  
 **if**(i%2 == 0){  
 wordsWithoutWhitespace[i-1] = wordsWithoutWhitespace[i-1].toUpperCase();  
 }**else**{  
 wordsWithoutWhitespace[i-1] = wordsWithoutWhitespace[i-1];  
 }  
 }  
 **return** wordsWithoutWhitespace;  
 }  
  
 **public** String[] createNewWordsFromTwoFirstAndLastLettersEveryWord(){  
 String[] wordsWithoutWhitespace = getAllWordsFromTextWithoutWhitespace();  
  
 **for** (**int** i = 0; i < wordsWithoutWhitespace.**length**; i++) {  
 String word = getStringWithoutSymbols(wordsWithoutWhitespace[i]);  
 **if**(word.length() > 4){  
 String newWord = word.substring(0,2) + word.substring(word.length()-2);  
 wordsWithoutWhitespace[i] = StringUtils.*replace*(wordsWithoutWhitespace[i], word, newWord);  
 }  
 }  
 **return** wordsWithoutWhitespace;  
 }  
  
 **public** String ChangeAllDigitsToEquivalentStrings(){  
 String allText = **text**;  
 **for** (**int** i = 0; i <= 9; i++) {  
 allText = StringUtils.*replace*(allText, String.*valueOf*(i), getDigitInStringForm(i));  
 }  
 **return** allText;  
 }  
  
 **public** String occurrencesOfEveryWord(){  
 Map<String, Integer> occurrences = **new** HashMap<>();  
 String[] allWordsWithoutSymbols = getAllWordsFromTextWithoutSymbols(**text**);  
 **for** (String s : allWordsWithoutSymbols) {  
 **if**(!occurrences.containsKey(s)){  
 occurrences.put(s,getOccurrenceOfWord(allWordsWithoutSymbols,s));  
 }  
 }  
 **return** occurrences.entrySet().toString();  
 }  
  
 **public** String formatNumbers(){  
 String[] words = getAllWordsFromTextWithoutWhitespace();  
 **for** (**int** i = 0; i < words.**length**; i++) {  
 **if**(NumberUtils.*isDigits*(words[i])){  
 words[i] = getFormattedString(Integer.*parseInt*(words[i]));  
 }  
 }  
 **return** Arrays.*toString*(words);  
 }  
  
 **public** String formatDates(){  
 String[] allWords = getAllWordsFromTextBySeparator(**"\t"**);  
 **for** (**int** i = 0 ; i<allWords.**length** ; i++) {  
 **try**{  
 LocalDate date = LocalDate.*parse*(allWords[i], DateTimeFormatter.*ofPattern*(**"MMMM d, yyyy"**, Locale.***US***));  
 allWords[i] = date.format(DateTimeFormatter.*ofPattern*(**"dd.MM.yyyy"**));  
 }**catch** (Exception ignored){}  
 }  
 **return** Arrays.*toString*(allWords);  
 }  
  
 **public** String concatenateSimilarStrings(){  
 String[] rows = getAllWordsFromTextBySeparator(System.*lineSeparator*());  
 List<String> titles = **new** ArrayList<>();  
  
 **for** (**int** i = 2; i < rows.**length**; i++) {  
 titles.add(StringUtils.*split*(rows[i], **"\t"**)[0]);  
 titles.set(i-2, StringUtils.*replace*(titles.get(i-2), **" "**, **""**));  
 }  
  
 List<String> concatenatedTitles = **new** ArrayList<>();  
  
 **while**(titles.size() > 0){  
 String string = titles.remove(0);  
 StringBuilder concatenatedBuffer = **new** StringBuilder(string);  
 **for** (**int** i = 0; i < titles.size(); i++) {  
 **if**(getStringWithoutNumbers(string).equals(getStringWithoutNumbers(titles.get(i)))){  
 concatenatedBuffer.append(**"AND"**).append(titles.get(i));  
 titles.remove(i);  
 i--;  
 }  
 }  
 concatenatedTitles.add(concatenatedBuffer.toString());  
 }  
  
 **return** concatenatedTitles.toString();  
 }  
  
 **private** String[] getAllWordsFromTextWithoutWhitespace(){  
 **return** StringUtils.*split*(**text**, **" \t"**);  
 }  
  
 **private** String[] getAllWordsFromTextWithoutSymbols(String text){  
 StringBuilder separator = **new** StringBuilder();  
 **for** (**int** i = 0; i < text.length() ; i++) {  
 **if**(!CharUtils.*isAsciiAlpha*(text.charAt(i)) && !CharUtils.*isAsciiNumeric*(**this**.**text**.toString().charAt(i))){  
 **if**(!separator.toString().contains(String.*valueOf*(text.charAt(i)))){  
 separator.append(text.charAt(i));  
 }  
 }  
 }  
 **return** StringUtils.*split*(text, separator.toString());  
 }  
  
 **private** String getStringWithoutSymbols(String string){  
 StringBuilder newWord = **new** StringBuilder();  
 **for** (**int** i = 0; i < string.length(); i++) {  
 **if**(Character.*isLetter*(string.charAt(i))){  
 newWord.append(string.charAt(i));  
 }  
 }  
 **return** newWord.toString();  
 }  
  
 **private** String getDigitInStringForm(**int** number){  
 String[] numbersInTextForm = {**"ZERO"**, **"ONE"**, **"TWO"**, **"THREE"**, **"FOUR"**, **"FIVE"**, **"SIX"**, **"SEVEN"**, **"EIGHT"**, **"NINE"**};  
 **return** numbersInTextForm[number];  
 }  
  
 **private** Integer getOccurrenceOfWord(String[] text, String word){  
 Integer occurrence = 0;  
 **for** (String s: text) {  
 **if**(word.equals(s)){  
 occurrence++;  
 }  
 }  
 **return** occurrence;  
 }  
  
 **private** String getFormattedString(**int** number){  
 NumberFormat numberFormat = NumberFormat.*getNumberInstance*(Locale.***US***);  
 DecimalFormat decimalFormat = (DecimalFormat)numberFormat;  
 decimalFormat.applyPattern(**"###,###"**);  
 **return** decimalFormat.format(number);  
 }  
 **private** String[] getAllWordsFromTextBySeparator(String separator){  
 **return** StringUtils.*split*(**text**.toString(), separator);  
 }  
 **private** String getStringWithoutNumbers(String string){  
 StringBuilder stringWithoutNumbers = **new** StringBuilder();  
 **for** (**int** i = 0; i < string.length(); i++) {  
 **if**(!Character.*isDigit*(string.charAt(i))){  
 stringWithoutNumbers.append(string.charAt(i));  
 }  
 }  
 **return** stringWithoutNumbers.toString();  
 }  
}

Лістинг класу «StringModifier»

**import** FileWriter.FileWriter;  
**public class** StringModifier {  
  
 **private static final** String ***NEXT\_LINE*** = System.*lineSeparator*();  
  
 **private int indexOfModification** = 0;  
  
 **private final** StringBuilder **header** = **new** StringBuilder();  
 **private final** StringBuilder **body** = **new** StringBuilder();  
  
 **public void** addModification(String description, String modification){  
 **indexOfModification**++;  
 addHeaderToFileForWriting(**indexOfModification** +**". "**+ description);  
 addBodyToFileForWriting(**indexOfModification** +**". "**+ modification);  
 }  
  
 **private void** addHeaderToFileForWriting(String header){  
 **this**.**header**.append(header).append(***NEXT\_LINE***);  
 }  
  
 **private void** addBodyToFileForWriting(String body){  
 **this**.**body**.append(body).append(***NEXT\_LINE***);  
 }  
  
 **public** String getHeader(){  
 **return header**.toString();  
 }  
  
 **public** String getBody(){  
 **return body**.toString();  
 }  
  
 **public** String getFormattedHeaderAndBody(){  
 **return header**.toString() +***NEXT\_LINE***+ **body**.toString();  
 }  
}

Лістинг класу «FileReader»

**package** FileReader;  
  
**import** java.io.File;  
**import** java.io.BufferedReader;  
**import** java.io.IOException;  
  
**public class** FileReader {  
  
 **private final** File **fileToRead**;  
  
 **private static final** String ***NEXT\_LINE*** = System.*lineSeparator*();  
  
 **public** FileReader(String filePathToRead){  
 **fileToRead** = **new** File(filePathToRead);  
 }  
  
 **public** String readTextFromFile(){  
  
 StringBuilder allTextFromFile = **new** StringBuilder();  
  
 **try**(BufferedReader br = **new** BufferedReader(**new** java.io.FileReader(**fileToRead**))){  
 String line;  
 **while**((line = br.readLine()) != **null**){  
 allTextFromFile.append(line).append(***NEXT\_LINE***);  
 }  
 } **catch** (IOException exception) {  
 System.***out***.println(**"PROBLEM WITH READING FROM FILE"**);  
 }  
  
 **return** allTextFromFile.toString();  
 }  
}

Лістинг класу «FileWriter»

**package** FileWriter;  
  
**import** java.io.BufferedWriter;  
**import** java.io.File;  
**import** java.io.IOException;  
  
**public class** FileWriter {  
  
 **private final** File **fileToWrite**;  
  
 **public** FileWriter(String filePathToWrite){  
 **fileToWrite** = **new** File(filePathToWrite);  
 }  
  
 **public void** writeTextToFile(String text){  
 **try**(BufferedWriter bw = **new** BufferedWriter(**new** java.io.FileWriter(**fileToWrite**))){  
 bw.write(text);  
 }**catch** (IOException exception){  
 System.***out***.println(**"PROBLEM WITH WRITING TO FILE"**);  
 }  
 }  
}

## 1.3 Результати роботи

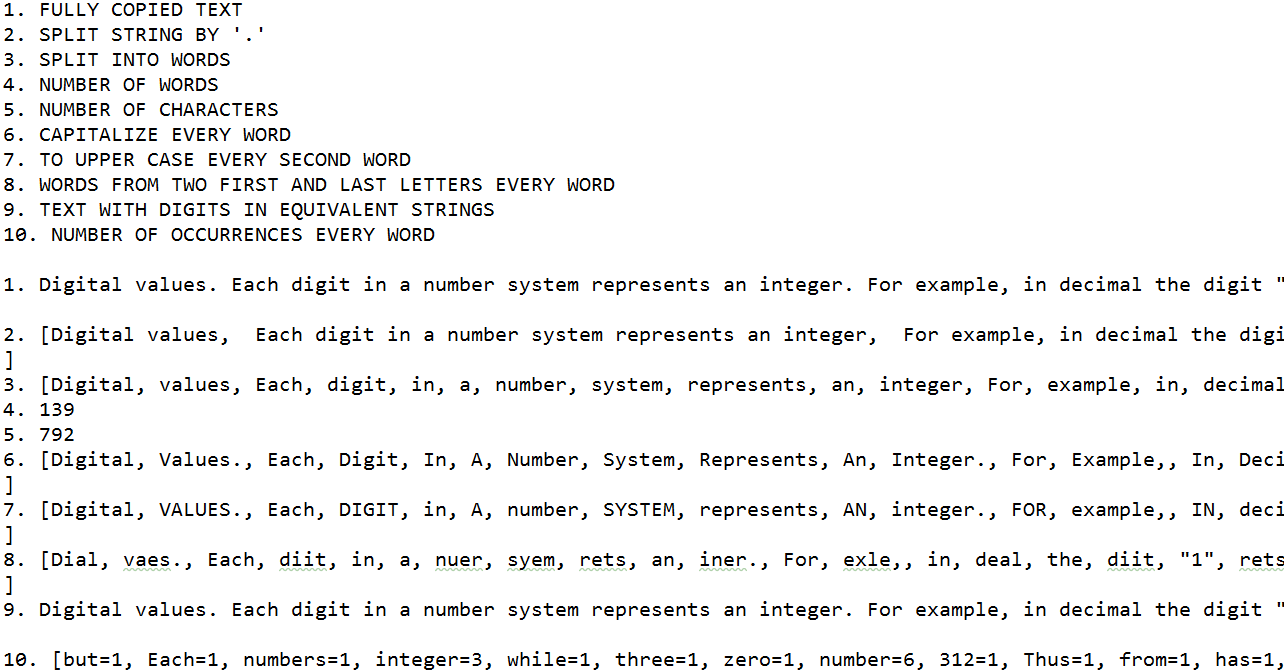


Рис. 2 – Оброблений файл Digital values.txt

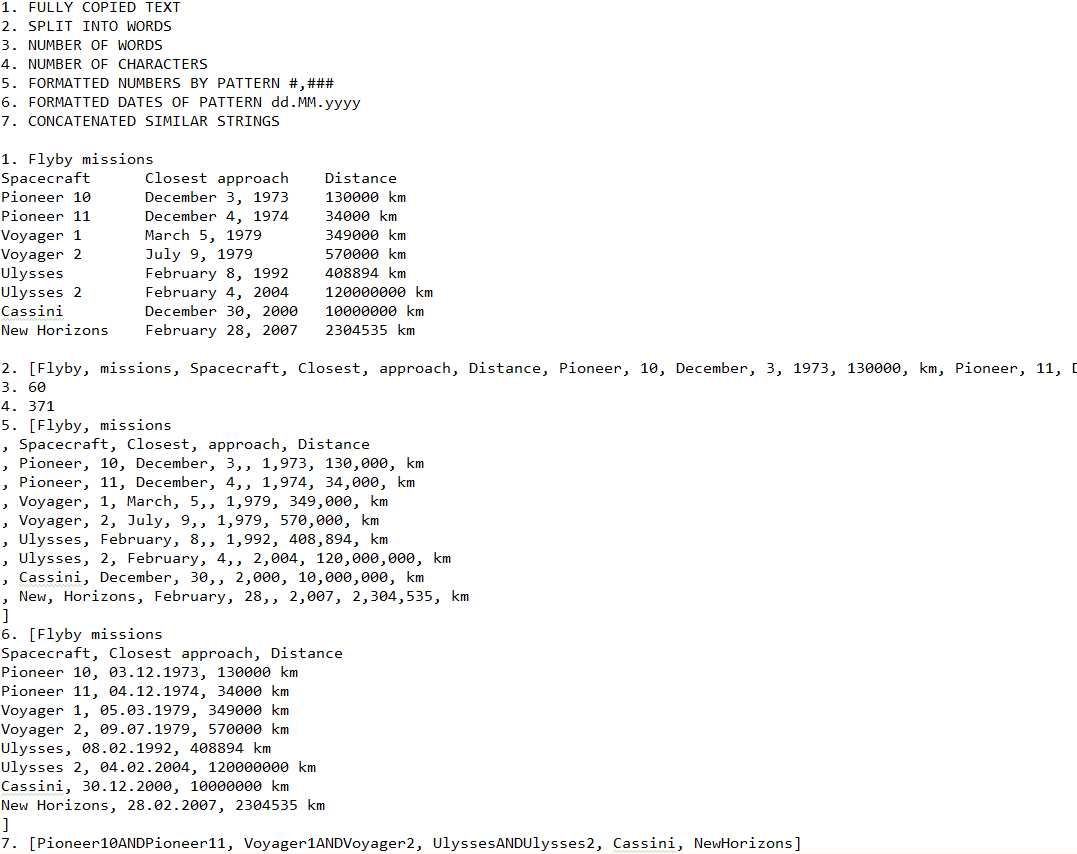


Рис. 3 – Оброблений файл Flyby missions.txt