Problem: In HW1 you will deliver the first version of Text Editor that performs some basic functions such as open and read a text file, count number of words, search and replace strings. In this version, you will only read a text file, and will not write anything back into it. This will be a CUI program with just two classes.

Design: The UML class diagram shown in Figure1 shows that there are two classes for this program. TextEditor is the main class which uses the FileUtilities class to perform basic file operations.

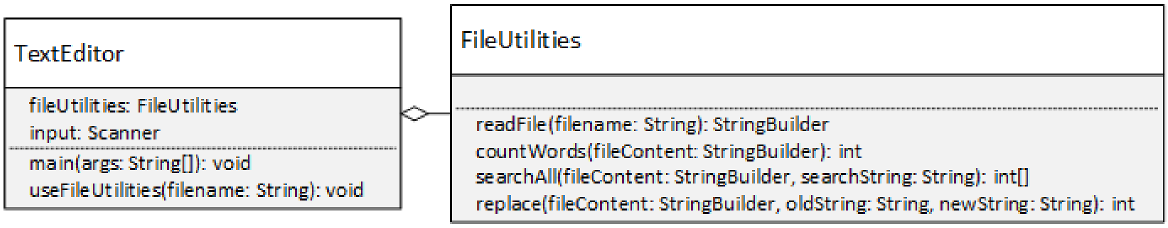

TextEditor
fileUtilities: FileUtilities
input: Scanner
main(args: String[]): void
useFileUtilities(filename: String): void
FileUtilities
readFile(filename: String): StringBuilder
countWords(fileContent: StringBuilder): int
searchAll(fileContent: StringBuilder, searchString: String): int[]
replace(fileContent: StringBuilder, oldString: String, newString: String): int

Figure 1: Class Diagram for HW1

Instructions:

- Create a project named **HW**.
- Create a package named **hw1** in this project
- Download and import three java files **TextEditor.java**, **FileUtilities.java**, and **TestTextEditor**.java into hw1 package
- Download and import **sample.txt** in HW project

You need to write your code only in FileUtilities.java. You will fill in the method-bodies for the signatures provided. You can add more methods, but you cannot change any of the method signatures.

You will test your program in two ways:

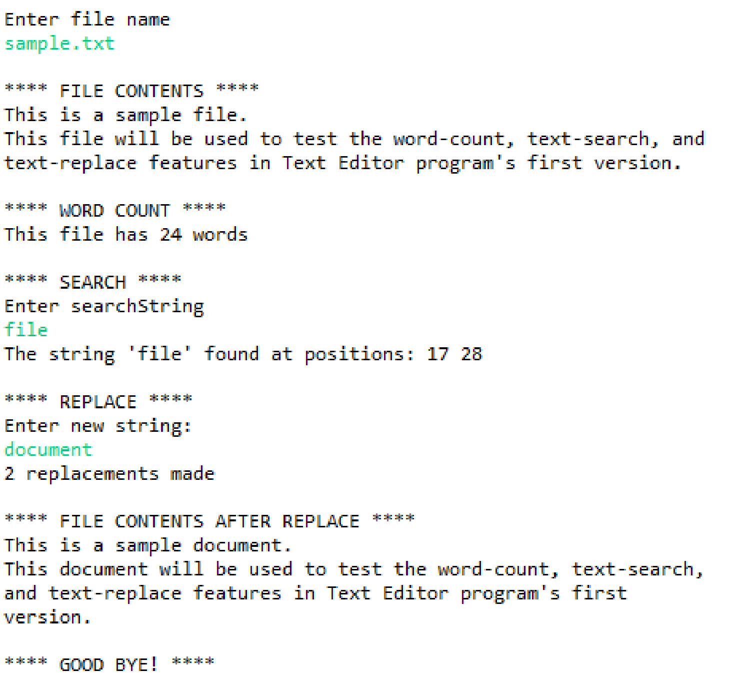
- Run TestTextEditor.java. DO NOT change any of the test cases. The test-cases use sample.txt as test-data.
- Run TextEditor.java that interacts through console I/O. Use sample.txt for this as well. Refer to the last page in this document for some screenshots.

Passing all the test is the basic requirement that will get you 75% of the points. Other criteria that will be applied to evaluate your program are:

1. Documentation (5%): Your code should be well-commented, i.e. neither too many comments, nor too few. Name your variables in a self-explanatory way so that you don’t have to explain what they are. Write your name and Andrew id at the top in the comments. Indent your code properly (In Eclipse, press Ctrl-A to select all your code and then Ctrl-I to indent)
2. Code quality (15%): Use of loops, if-else, should be wisely made. Remove all unused variables and libraries.
3. Submission instructions (5%): Do not submit folders, class files, test file, text files, and rest of your kitchen sink! Submit only FileUtilities.java.

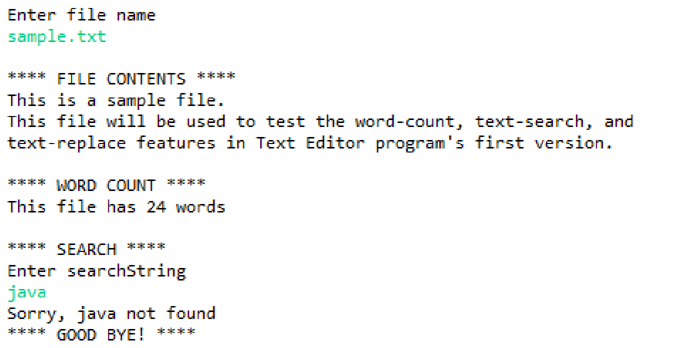
NO LATE SUBMISSIONS PLEASE! If you are unable to submit by due date and time, you lose all the points. So please avoid last minute submissions because sometimes Canvas may decide to quit on you! Learn to trust technology only to the extent you should! I will have no way to know what happened so do not take that risk. Submit well before time!

Following screenshots will help you understand the expected program-flow and different scenarios. Note that the screenshots were taken from Eclipse Console with its 'word-wrap' feature enabled.



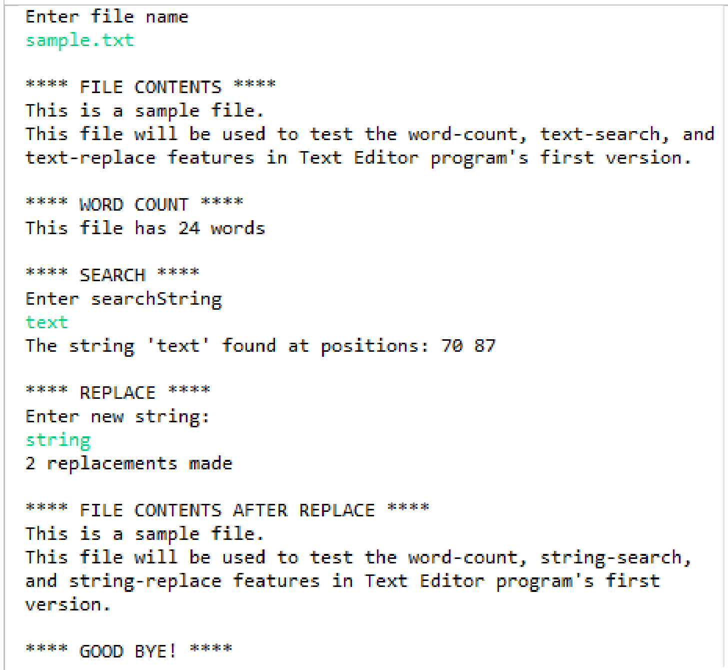
```
Enter file name
sample.txt

**** FILE CONTENTS ****
This is a sample file.
This file will be used to test the word-count, text-search, and
text-replace features in Text Editor program's first version.

**** WORD COUNT ****
This file has 24 words

**** SEARCH ****
Enter searchString
file
The string 'file' found at positions: 17 28

**** REPLACE ****
Enter new string:
document
2 replacements made

**** FILE CONTENTS AFTER REPLACE ****
This is a sample document.
This document will be used to test the word-count, text-search,
and text-replace features in Text Editor program's first
version.

**** GOOD BYE! ****
```

Figure 2: Console I/O with sample.txt data. Search string found and replaced



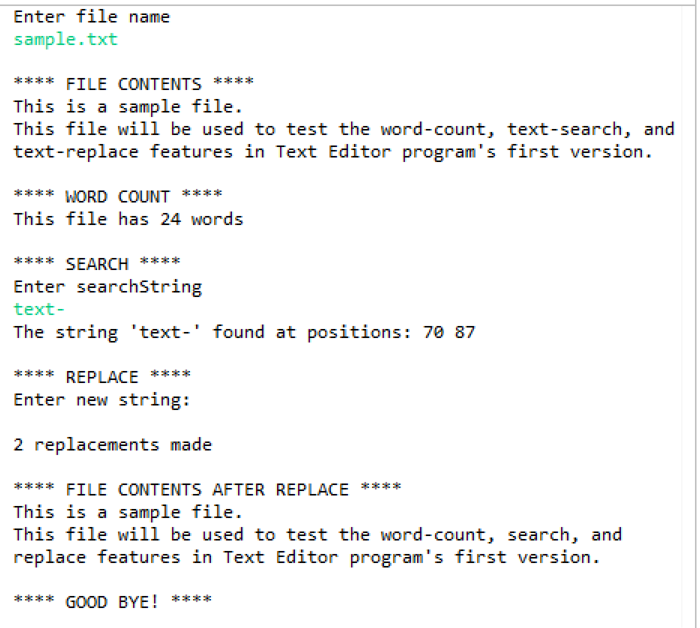
```
Enter file name
sample.txt

**** FILE CONTENTS ****
This is a sample file.
This file will be used to test the word-count, text-search, and
text-replace features in Text Editor program's first version.

**** WORD COUNT ****
This file has 24 words

**** SEARCH ****
Enter searchString
java
Sorry, java not found
**** GOOD BYE! ****
```

Figure 3: Console I/O with sample.txt. Search string not found



```
Enter file name
sample.txt

**** FILE CONTENTS ****
This is a sample file.
This file will be used to test the word-count, text-search, and
text-replace features in Text Editor program's first version.

**** WORD COUNT ****
This file has 24 words

**** SEARCH ****
Enter searchString
text
The string 'text' found at positions: 70 87

**** REPLACE ****
Enter new string:
string
2 replacements made

**** FILE CONTENTS AFTER REPLACE ****
This is a sample file.
This file will be used to test the word-count, string-search,
and string-replace features in Text Editor program's first
version.

**** GOOD BYE! ****
```

Figure 4: Case-sensitive search and replace. 'text' replaced with 'string'. 'Text' not replaced.



```
Enter file name
sample.txt

**** FILE CONTENTS ****
This is a sample file.
This file will be used to test the word-count, text-search, and
text-replace features in Text Editor program's first version.

**** WORD COUNT ****
This file has 24 words

**** SEARCH ****
Enter searchString
text-
The string 'text-' found at positions: 70 87

**** REPLACE ****
Enter new string:

2 replacements made

**** FILE CONTENTS AFTER REPLACE ****
This is a sample file.
This file will be used to test the word-count, search, and
replace features in Text Editor program's first version.

**** GOOD BYE! ****
```

Figure 5: Search string found and replaced with empty string.

Enjoy!!