Andrew Lawler – COMP122 - Assignment 4 - 201210893

Introduction

This assignment tasked us with learning how to use Java I/O. This assignment wanted us to take input from any amount of files using java command line arguments. We then had to take the text and concatenate them into one large string which we then performed tasks on. Overall, I enjoyed this assignment as it allowed me to learn more about Java and more importantly HashMap.

Part A & B

Part A and B was more about the actual code so I don’t have to write much about them. I have fully documented my design using in line comments and Javadoc comments to make it easy to follow.

Part C

Part C asks us to answer some questions about our code to learn what it would output when we input certain things.

No filename is given as an argument?

My program simply checks and see’s that there has been no input. It then responds by sending the following message.

A close up of a black background

Description automatically generated

There is only one argument "X" but no file of that name exists?

A close up of a sign

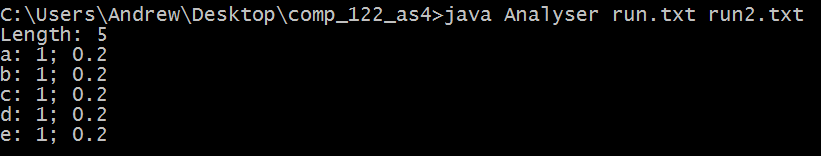
Description automatically generatedMy program would check if the file exists using the file.exists() function and then when it see’s that it doesn’t. It would print out that the file does not exist.

All given filenames point to existing, but empty text files?

A screenshot of a social media post

Description automatically generatedMy code would open the files and then using an if statement, see that they are in fact empty. It would then print out that the file is empty

Two filenames are given, and the corresponding text files contain the strings "abc" and "de", respectively?

This is a good example. My code would concatenate the two strings and then get one string of abcde. It would then calculate the frequencies for each letter and store them inside the HashMap. It would then print them out in the correct format as 0.2 for every character.

A screenshot of a social media post

Description automatically generated

Part D

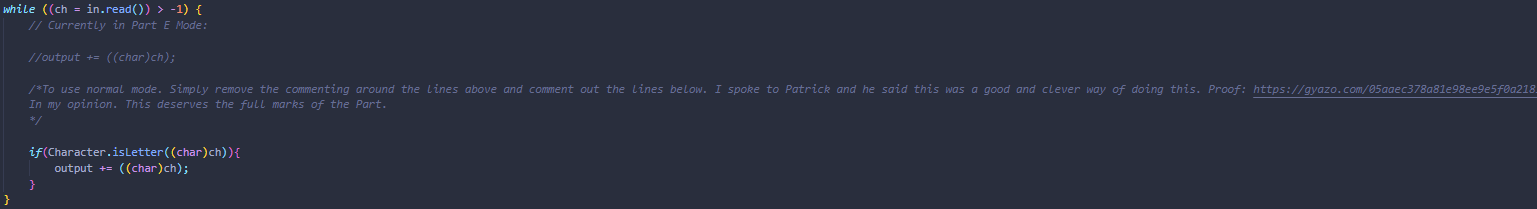
For Part D I decided to keep it simple. I followed the advice of the slides and did lots of tests for each method. This proves that my methods work properly and do what they should do. I also included Javadoc comments and in-line comments in this file so you can look for yourself.

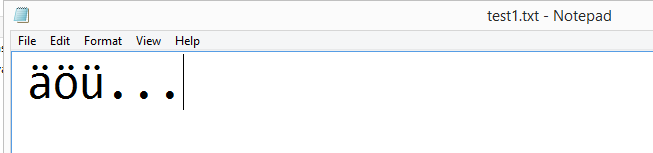
I also included the txt files I used to test my program. One error I found, if the .txt file is saved as UTF-8. Sometimes you get a ? in the input which in turn adds this ? to the string and included it as a character. According to official java documentation this occurs when a character is not recognized by your command prompt. Slightly odd. To make sure this wouldn’t happen I have put my program in Part E mode. Basically, when a ? is generated, it will remove it. Leaving just the characters. I also used official java source code for the reading of the files. This made sure that the error was not something on my part. I have included an example below of someone else getting the ?.

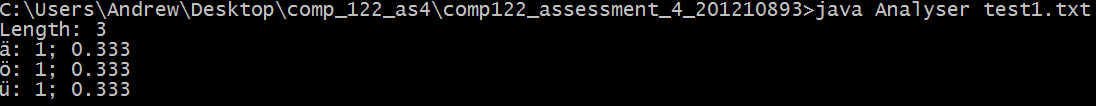
Example of someone else getting ? too: <https://www.mkyong.com/java/how-to-read-utf-8-encoded-data-from-a-file-java/>

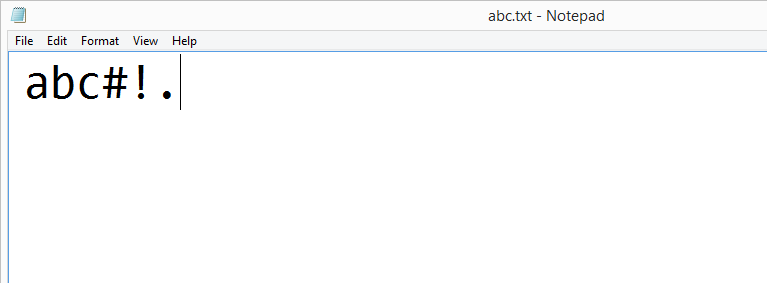
I included this just so you are aware. If you disable my Part E mode and then run the file you will get this issue. I have no idea why. If you save the txt file as ANSI, this error also does not occur. Strange.

Part E

For Part E I decided to go for a completely different method of implementation. For Part E we got the hint to use a whitelist and then connect that via a second constructor. Instead, I decided to implement directly into my first file and then just add some code. For Part E, if you look in main and scroll to the file reader, there is some commented out code with instructions. Basically, you just uncomment the code and comment the line that says output += c. I have used isLetter which is something you can use with Char’s. This will simply exclude all of the punctuation and spacing for me, this is much better as it means it will physically exclude any symbol that is not a letter. Below I will include the source code and two simple tests for this.

Test 1 (Using Umlaut’s)



Test 2 (Using English)

