Computer Science Final Project Journal

Planning

Day 1 - Deciding on what to make for my project. I want to keep it in java because I am now familiar with the language and don't want to waste time learning a new language and take time away from development.

Day 2 - I have decided to build a word processor. It will have the functions of a regular word processor. These functions include: writing to a file, reading from a file, deleting a file, and listing the files.

Coding

Day 3 - The main class is called "Processor". It will handle user input and output.

Day 4 - I've decided that the documents will each be their own object type. I've created a new "Document" class that is initialized with the documents name passed as a parameter for the constructor. It will have two ArrayLists, 'files' and 'fileName'. 'Files' will store the actual document objects and 'fileName' will store the Documents' names in the corresponding positions. This is necessary for searching for files by their filenames.

Day 5 - The processor class has a primary method called *command*() that will handle user input and will contain a switch that initializes each different command's function. The switch will have the case help, new, delete, list, and write, and quit. The default case will spit out an error message.

help - displays all available commands and there functions
new - creates a new file with filename.txt
delete - deletes the filename.txt matching input
list - lists all files in fileName ArrayList
write - writes text to an existing file
quit - exits the program

Day 6 - I created the help command that basically spits out the text above to the console. I also created a filePos method that searches for the position of the filename in the fileName ArrayList. The new function first uses filePos to check if the file exists, if it does not, it uses fileWriter to create a new filename.txt in the files/ directory and adds it to both arrays.

Day 7 - I used similar code as the new function and created a delete function that does the opposite. I also created a list function that prints out the values of the fileName ArrayList in a list form. I have decided for writing and reading a Document, the methods would be within the Document class. I created two empty methods: *print()* and *write().*

Day 8 - The *print()* method uses the BufferedReader library to read each line of the text document and print each one line by line. The *write()* method uses the FileWriter library to write to the filename.txt

Day 9 - I also decided to add spellcheck and say functions. The spellcheck function compares each word of a english language wordlist to every word in the sentence. If there is not a match for each word, the incorrect word is printed to the screen. I also created a method in the Document class that converts the file's text to a string and returns it for use with the spellcheck function. I also decided that the say function will convert the file's text to speech. I think I'm going to use the freeTTS API.

Day 10 - I finished configuring the freeTTS API and implemented it into the method *speak*(). For some reason, it prints an error to the console. I'm going to look into this tomorrow.

Day 11 - I was unable to resolve the error so I used the PrintStream library to divert the error message to a dummy PrintStream.

Day 12 - I noticed that like conventional text editors, My text editor didn't save the files in the program after quitting. I created a *scan()* method that searches the files/ directory for existing files and creates a new Document for each and adds it to both ArrayLists on the program startup

Day 13 - Commented everything

Day 14 - I changed the command layout to make the command and user input work on one line. So instead of the command "new" asking for the name input, the format is "new |filename|". This works for every command working with files (Does not include help, list or quit). I then commented the modifications.

Testing and Deployment

Day 15 - Tested the program thoroughly, seems to be working fine. I also finalized the welcome menu and worked out formatting things like spacing.