Name: Jemil Patel

Instructor: Gary Thai

CMSC 204 Project-5 Fall 2022

**Approach, Design & Algorithm**

I approached the project by first reading the given word file. It helped me get an overview of what I needed to do and how the program works. I then started created the TreeNode class which was quite easy. Next, I moved towards the MorseCodeTree class that implemented a provided interface. Besides, I also followed the given Javadoc files to further understand how each class works.

At first, I was a bit overwhelmed by the number of methods the MorseCodeTree class uses. However, as I went on coding I slowly started getting where all this went to. Implementing the add and fetch methods were the core of this project and so I made sure that they worked correctly without any errors. Apart from that, the remaining methods were comparatively easier to implement.

Finally, I moved on to the MorseCodeConverter class. Here I needed to make sure that I do not add any extra spaces after the conversion to English or else the JUnit tests would fail. In the convertToEnglish method I utilized String class’s split method to separate letters and words, so that I could convert the morse code to English alphabet and produce a meaningful sentence.

After completing all three classes I tested them a couple of times using different types of input. Nonetheless, I also tested them using the GUI, which worked perfectly fine.

**Test Cases**

**Test Case #1: Valid input**

Graphical user interface, text, application, email

Description automatically generated

**Test Case #2: Valid input**

**Graphical user interface, text, application

Description automatically generated**

**Test Case #3: Invalid input**

**Graphical user interface, text, application

Description automatically generated**

**Test Case #4: Invalid input**

**Graphical user interface, text, application

Description automatically generated**

**Test Case #5: Invalid input**

**Graphical user interface, text, application

Description automatically generated**

**Test Case #6: Daisy.txt**

**Graphical user interface, text, application

Description automatically generated**

**Test Case #7: DaisyDaisy.txt**

**Graphical user interface, text, application

Description automatically generated**

**Test Case #8: howDoILoveThee.txt**

**Graphical user interface, text, application

Description automatically generated**

**Test Case #9: LoveLooksNot.txt**

**Graphical user interface, text, application, email

Description automatically generated**

**JUnit Tests**

**MorseCodeConverter\_GFA\_Test.java**

**Graphical user interface, text, application, email

Description automatically generated**

**MorseCodeConverterTestPublic.java**

**Graphical user interface, text, application, email

Description automatically generated**

**Learning Experience**

Project-5 has been a great learning experience for me. Through it I got a good understanding of binary tree and binary tree traversals. Since the project involved working with morse code, it never felt boring. Furthermore, I got a good experience working on recursive methods to get desired results. In order to pass an updated code every time to the method I used the substring method of String class to remove the first character from the code and pass the rest of it during each recursive call.

I did not face any difficulty or challenge in completing the project. Thus, I think that I would not change anything if I had a chance to. All in all, this was a very interesting and unique project that helped me strengthen my knowledge of binary tree, tree traversals, and recursion.

**Assumptions:**

* Any invalid morse code input will result in displaying a message, “Invalid morse code!”