Michael Wells

06/30/19

Assignment 5 Design

Morse Code Translator

My main method calls the **translator** method. I did this to make it easier to add a loop for asking the user if they want to run the translator again but I did not have time to add this feature. The translator method prints a header and promts the user to enter either 1 (to translate Morse code to English) or 2 (to translate English to Morse Code). It then calls either the **morse2english** method or the **english2morse** method. Both overall prompt the user to enter text to be translated, then translate the text and output the translated text. I wanted to keep these separate functions to make the design simpler and more reusable.

Both of these methods call the **searchTranslationArray** which takes in an english string, and morse code string, and direction integers that specifies which way to translate. I designed it like this so that I didn’t have to create multiple reference arrays. The both input strings for the **searchTranslationArray** methodrepresent a single English letter. This method takes the string to be translated and loops throw the reference array until it finds the matching string. Once it finds a match it uses the same index in the other reference array and outputs this string as the translated text.

The only difference **morse2english** and **english2morse** is really how they separate the input text into tokens that can be translated. In the **english2morse** this is very straight forward as I just loop through each letter, change the character letter to a string and call the **searchTranslationArray**  method. For **morse2english** I used the split method that is apart of base java to break up the input text into a string array from every white space, then looped through the length of the array and called the **searchTranslationArray**  method every iteration.