Morse Code\*

Let’s start by watching this [video](https://www.khanacademy.org/computing/computer-science/informationtheory/info-theory/v/morse-code-the-information-age-language-of-coins-8-12) for some background and this [video](http://www.wimp.com/neatexperiment/) for some fun.

In 1838 Samuel Finley Breese Morse (1791-1872) devised a signaling code for use with his electromagnetic telegraph. The code used two basic signaling elements: the “dot,” a short-duration electric current, and the “dash,” a longer-duration signal. The signals lowered an ink pen mounted on a special arm, which left dots (aka dits) and dashes (aka dahs) on the strip of paper moving beneath. Morse’s code gained wide acceptance and, in its international form, is still in use.

In 1858 Queen Victoria sent the first transatlantic telegram of ninety-eight words to congratulate President James Buchanan of the United States. The telegram started a new era of “instant” messaging – took only sixteen and a half hours to transmit via the brand new transatlantic telegraph cable!

The table below shows the Morse Code.

A .- N -. 1 .---- . .-.-.-

B -... O --- 2 ..--- , --..--

C -.-. P .--. 3 ...-- ? ..--..

D -.. Q --.- 4 ....- ( -.--.

E . R .-. 5 ..... ) -.--.-

F ..-. S ... 6 -.... - -....-

G --. T - 7 --... " .-..-.

H .... U ..- 8 ---.. \_ ..--.-

I .. V ...- 9 ----. ' .----.

J .--- W .-- 0 ----- : ---...

K -.- X -..- / -..-. ; -.-.-.

L .-.. Y -.-- + .-.-. $ ...-..-

M -- Z --.. = -...-

In this project, we will simulate a telegraph station that encodes messages from text to Morse code and decodes a Morse code message back to plain text. You are to complete a MorseCode class.

**Instance Variables in the MorseCode class**

**TreeMap<Character, String> toCode**

This data structure will be used to encode messages into Morse Code. The keys are the (upper case) alphabetic characters above, a space, and a carriage return. Their values are the Morse Code strings given above. For space, the value is “\*“. For carriage return, the value is “+”.

**TreeNode<Character> root**

This data structure will be used to decode messages back to plain text. We hope it is obvious to you that you could use a second **TreeMap** to decode messages. **However, using a TreeMap for decoding will result in 0 points for the method.** Instead, our data structure is an explicit tree representation of the codes. A dot means go down the left hand branch. A dash means follow the right hand branch. Below is the resulting binary search tree borrowed from Wikipedia. Note that our tree doesn’t contain all the special characters. Also note that that the tree does not contain the \* and +. These need to be decoded separately.

Both of the above data structures are created by having the constructor read in a file named MorseCode.txt which contains all the letters in the alphabet and their Morse translation. The constructor is all set to read in that file, but make certain that the file is in the project directory.

**Assignment**

Complete the following public methods:

**public void encodeFile(String inputFilename, String outputFilename) throws Exception** – which takes the name of a text file and encodes the entire file writing the output to the other file. Each line of the output file will be the Morse Code string for one character.

Note that Morse Code does not have a way to represent a space or return. Telegraph operators would just leave a small time slice between characters when sending. For our purposes, if the character in the file is a space insert a line containing a \* into the file. If the character is a carriage return insert a line containing a + into the file. Punctuation and other special characters may be ignored.

**public void decodeFile(String inputFilename, String outputFileName) throws Exception** is the second method you need to implement. It should read a code String from the input file, translate it to its corresponding char and write that character to the file. To translate the String to a character, use the Morse Code tree (except for the two special strings, \* and +).

**Sample Output**

|  |  |  |
| --- | --- | --- |
| **sos.txt** | **encoded.txt** | **decoded.txt** |
| SOS  SOS  SOS | ... --- ... + ... --- ... + ... --- ... | SOS  SOS  SOS |

----------------------------------------------------------------------------------------------------------------

\*Original lab developed by Stuart Hansen from the University of Wisconsin-Parkside for their Computer Science II course.