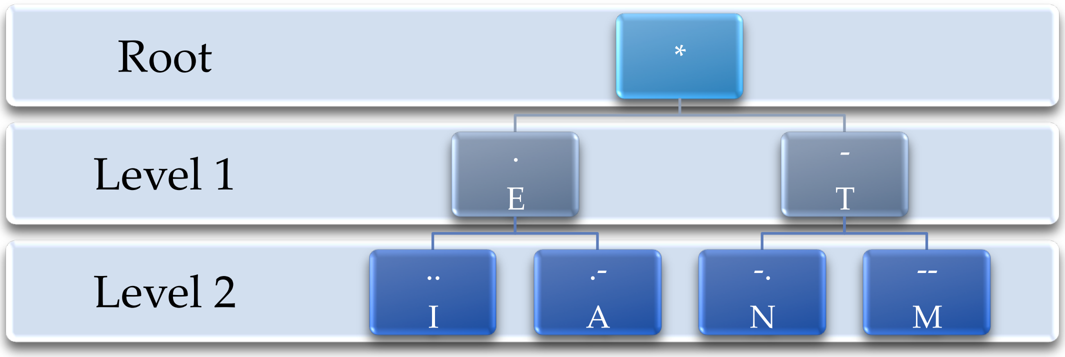
Binary Tree Project

Write a program to represent the Morse code as a binary tree. The dot symbol (.) should cause a branch to the left and the dash symbol (-) a branch to the right. The data field of each node should be the letter represented by the corresponding code. The top part of the tree will look like this with the root represented by a `'*'`. You can use any classes that have been provided to understand trees.



The input of the program should consist of a letter, and the code string. After the complete tree has been built, from a file allow a message in Morse code to be entered. Then print the message in English.

Data files: `MORSECODE` contains the chart below, use to load your binary tree with the Morse code. `TESTMORSECODE` contains some Morse code to decode you should test your program with this file, add your own phrases to the file. (add at least one phrase to the file and turn in with project)

When entering your data to convert: letters are separated by a space, words are separated by a /. Make sure to separate your words in your output.

Use the following chart for the Morse code.

Letter	Code Length	Code
E	1	.
T	1	-
I	2	. .
A	2	. -
N	2	- .
M	2	- -
S	3	. . .
U	3	. . -
R	3	. - .
W	3	. - -
D	3	- . .
K	3	- . -
G	3	- - .

Letter	Code Length	Code
O	3	- - -
H	4	. . . .
V	4	. . . -
F	4	. . - .
L	4	. - . .
P	4	. - - .
J	4	. - - -
B	4	- . . .
X	4	- . . -
C	4	- . - .
Y	4	- . - -
Z	4	- - . .
Q	4	- - . -

Suggested Algorithm

-create a static BinarySearchTree to hold the morsecode (Characters)

-main method

- To use file input you have import java.io.File; and include throws IOException on main method heading.
- Create a Scanner object (remember import) that uses a new File object as it's constructor parameter. The new File object should construct with the file name, MORSECODE.
- Add the root to start the tree ('*')
- While loop that reads until end of file has been reached (use hasNext method of Scanner class)
    - Use next method of Scanner to read each individual String
    - Store the character in a Character object by getting charAt(0)
    - Add to morsecode tree using overloaded add method that you added(pass 0 to first)
- Close Scanner object
- Create Scanner object with new File object with TESTMORSECODE as file name.
- While loop that reads until end of file has been reached (use hasNext method of Scanner class)
    - Read in first line of file
    - Output code to decode
    - Output decoded sentence (call static method to complete task)
- Close Scanner object

-decodeSentence method(@return String decodedSentence, @param String sentence)

- Declare String to hold decoded sentence to return.
- Declare int to determine the number of words (use a method to complete task)
- Declare a String array to hold the words to decode, size should be the int you declared to hold the number of words.
- Use a method to separate the words in the sentence and store in array.
- Loop for the number of words in the sentence
    - Count spaces to determine how many letters in word.
    - Declare String array to hold the codes for each letter
    - Use method to separate the letters
    - Use method to decode each word, store result in String to return (remember to add space to word to store as a sentence)
- Return decoded sentence

-decodeWord method(@return String word, @param String[] codes)

- Declare empty String to store word to return
- Loop for the length of the code
  - o Call decodeLetter for current code, add on to String to return
- Return word

-decodeLetter method(@return Character letter, @param String code)

- Declare empty Character to store letter to return
- get the root of morsecode tree and store in a BinarySearchTreeNode object
- loop for the length of the code
  - o if the charAt current spot of code is a '.' Then go left (getLeft()) store in root
  - o if the charAt current spot of code is a '-' Then go right(getRight()) store in root
- store the letter of the code (getInfo())
- return letter

-separateLetters(void, @param String[] letters, String word)

- Declare empty String to hold the letter code
- Declare a counting variable set to 0
- Loop for the length of word
  - o If word current character is not space(' ') then add character to code
  - o Else store code in array (index counting variable), empty code string and add to counting variable
- Store last code in array

-separateWords(void, @param String[] words, String sentence)

- Declare empty String to hold the word
- Declare a counting variable set to 0
- Loop for the length of sentence
  - o If sentence current character is not slash('/ ') then add character to word
  - o Else store word in array (index counting variable), empty word string and add to counting variable
- Store last word in array

Some support methods that I used:

```
public static int countSpaces(String phrase)
{
          int count = 0;
          for(int i = 0; i < phrase.length(); i++)
          {
               if(phrase.charAt(i) == ' ')
                    count++;
          }
          return count+1; //add one more to count the last letter
}

public static int countWords(String phrase)
{
          int count = 0;
          for(int i = 0; i < phrase.length(); i++)
          {
               if(phrase.charAt(i) == '/')
                    count++;
          }
          return count+1; //add one more for the last word
}
```

Make the following changes to classes provided:

-BinarySearchTree class add the following method to overload the add method

```
/*
 * Specific add for morse code since it uses code to store element
 *
 */
public void add(T element, int first, String code)
{
     BinarySearchTreeNode<T> newNode = new
                                BinarySearchTreeNode<T>(element);
     if(root == null)
          root = newNode;
     else
          root.addNode(newNode, first, code);
}
```

-BinarySearchTreeNode class add the following method(Look at current addNode method for guidance.) Make sure to overload the current addNode

```
/*
 * Insert a new node based on if . or - as a descendant of this node
 * @param newNode the node to insert, start index of string, code
 */
public void addNode(BinarySearchTreeNode<T> newNode,int start,String
code)
```

- Declare char variable to hold start character
- If char is equal to '.' Go left
    - If left is null then set left to newNode
    - Else left.addNode(newNode, start+1, code)
- If char is equal to '-' Go right
    - If right is null then set right to newNode
    - Else right.addNode(newNode, start+1, code)