Data Element - **TreeNode class**

This generic class is used in the MorseCodeTree classes. The class consists of a reference to the data and a reference to the left and right child. Follow the Javadoc that is provided. The Javadoc only lists those public methods that are required to pass the Junit tests. You may add any private methods you need for your design.

Data Structure - **MorseCodeTree class**

A generic linked binary tree which inherits from the LinkedConverterTreeInterface. The class uses an external generic TreeNode class parameterized as a String: TreeNode<String>. This class uses the private member of root. Nodes are added based on their morse code value. A '.' (dot) means to traverse left and a '-' (dash) means to traverse right. The constructor will call the method to "build the tree". Follow the Javadoc that is provided. The Javadoc only lists those public methods that are required to pass the Junit tests. You may add any private methods you need for your design.

Utility class - **MorseCodeConverter**

The MorseCodeConverter contains a static MorseCodeTree object and constructs (calls the constructor for) the MorseCodeTree.

This class has two static methods *convertToEnglish* to convert from morse code to English. One method is passed a string object (".-.. --- ...- . / .-.. --- --- -.- ..."). The other method is passed a file to be converted. These static methods use the MorseCodeTree to convert from morse code to English characters. Each method returns a string object of English characters.

There is also a static printTree method that is used for testing purposes – to make sure the tree for MorseCodeTree was built properly.

Use the Javadoc provided to make sure that your MorseCodeConverter class follows the method headers so that the MorseCodeConverterTest will run correctly.

MorseCodeTree is a 4 levels tree. Insert a mapping for every letter of the alphabet into the tree map. The root is a TreeNode with an empty string. The left node at level 1 stores letter 'e' (code '.') and the right node stores letter 't' (code '-'). The 4 nodes at level 2 are 'i', 'a', 'n', 'm' (code '..', '.-', '-.', '—'). **Insert** into the tree by tree level from left to right. A '.' will take the branch to the left and a '-' will take the branch to the right. This is the structure of the tree.