**CMSC204**

**Assignment #5**

Samuel F. B. Morse produced the first working telegraph set in 1836. This made transmission possible over any distance. The first Morse Code message, "What hath God wrought?", was sent from Washington to Baltimore.

Morse code was extensively used for early radio communication beginning in the 1890s.

In the early part of the twentieth century, the majority of high-speed international communication was conducted in Morse code, using telegraph lines, undersea cables, and radio circuits.

Morse code can also be transmitted using light which sometimes happens between ships at sea. It is used in emergencies to transmit distress signals when no other form of communication is available. The standard international distress signal is •••---••• (SOS).

**Assignment Description**

Write the classes required to create a Morse Code Converter Utility. Your Morse Code Converter Utility will be using a generic linked binary tree with generic TreeNodes to convert Morse Code into English. There is no GUI requirement for this assignment. You are supplied a GUI for testing purposes.

**Academic Honesty Policy Reminder** | **Do your own work** – each submitted project will be compared against other submissions from current and previous semesters.

**Concepts tested by this assignment**

Generic Classes

Utility Class (all static methods)

Linked Trees

Building a Tree for conversion purposes

**Classes**

**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.

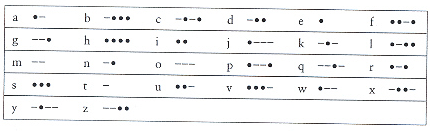
Use the Javadoc provided to make sure that your MorseCodeConverter class follows the method headers so that the MorseCodeConverterTest will run correctly.

**Testing - JUnit Test Classes**

You must add at least 1 test for MorseCodeConverter.convertToEnglish(String) and at least 1 test for MorseCodeConverter.convertToEnglish(File) to the MorseCodeConverterTest class. You must create a JUnit test for your MorseCodeTree class. Include your test files with your code files.

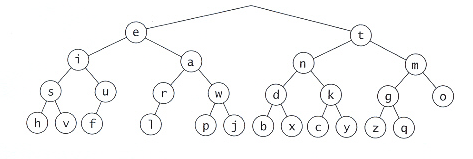
**Assignment Details**

This is a table for the conversion from Morse Code to alpha letters.



**Building the MorseCodeTree (method buildTree)**

Your 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.



**Using the MorseCodeTree**

Use the MorseCodeTree to convert Morse Code to English by taking the code and finding it’s corresponding English letter by traversing the MorseCodeTree, ‘.’ branches to the left and ‘-‘ branches to the right. The code ‘.--.’ would branch to the left, then to the right, then to the right, then to the left to **Fetch** the letter ‘p’. Each letter is delimited by a space (‘ ‘). Each word is delimited by a ‘/’.

Some suggestions:

1. There is a morse code translator at:

<http://morsecode.scphillips.com/jtranslator.html>

1. Morse code (Wikipedia)

https://en.wikipedia.org/wiki/Morse\_code

This will help you build files and test cases for your JUnit Tests.

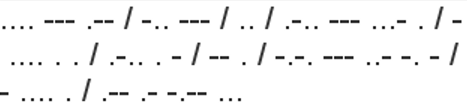
**Examples**

**Test Cases:**

Hello World



How do I love thee let me count the ways



**Write-up & Submission Requirements**

Review the provided rubric to understand project expectations, including documentation, CMSC204, and programming requirements. Two artifacts MUST be submitted for this project: Java source code and a write-up.

At a minimum, the write-up needs to address -

* Approach, design & algorithm
  + **DO NOT** start coding your project immediately! Come up with a high level design for the project first
    - What’s your game plan to complete the project?
    - Break the project into smallest modules where applicable
    - UML diagrams will go a long way
  + Each student is welcome to expand on the design, if it makes sense. Students will not be penalized for going “above and beyond” specifications of the project
  + **Complete this step first, then write your code**
* Test Plan & Test Cases
  + Ensure that your project can successfully pass all provided “public test cases.” What other test cases have you attempted?
    - Students should think about potential private tests – what would they be?
  + **Your instructor will test your project using an additional set of private test cases as well as the provided public test cases**
  + I want to see your “thinking,” in particular, how you are testing your program
  + Each submission should be rock solid, with no bugs
  + Capture screenshots of your test runs in your write-up as you run them
* Highlight your learning experience and lessons learned
  + **I am very interested to learn about what you have done, how you did, etc.**
* Anything else that you want to share with me
* Take a screen shot of the repo with the directory and files
* If your project is not working as expected, **submit it as-is but clearly articulate your situation in your write-up in order to potentially receive partial credit**

Each student must submit one compressed (.zip) file back to the Assignment (link) with the following deliverables:

* Your entire Eclipse project folder (Source code included)
* Write-up as a Word or PDF file, one write-up per project
* **I WILL compile, run and test every submission using a set of public and private test cases – this is how I test to ensure that your project is working correctly**
* Name the compressed file (zip format) as <lastname>\_project\_x
  + where x is the project number and your last name (e.g. Thai\_Project\_1.zip)
* Double check your submission, as I can only grade what’s being submitted (don’t assume anything)
* DO NOT email your submission! It won’t be accepted

**Not clear? That’s okay, but do ask your questions. “I did not know” or “I did not understand” is not good enough.**

Start working on each project immediately so that we can discuss any concerns or questions you have. **Reminder – I am unavailable on Saturdays and have limited availability on Sundays.**