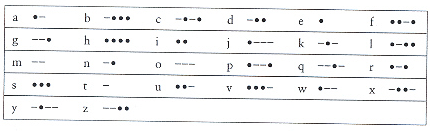
**Project 5**

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](http://en.wikipedia.org/wiki/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).



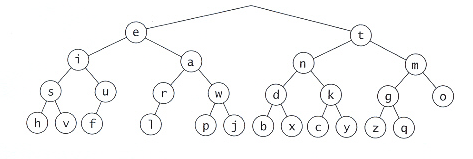
Your assignment is to write a generic TreeNode class, a MorseCodeTree class and a MorseCodeConverter class. There is no GUI required for this assignment. Your classes will be tested with Junit tests.

**Specifications**

**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.

**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 “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.

**Building the Data Structure (buildTree) -** Your MorseCodeTree is a tree 4 levels. 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 Data Structure**

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 ‘/’.

**MorseCodeConverter – Utility Class**

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.

**The JUnit Test Class**

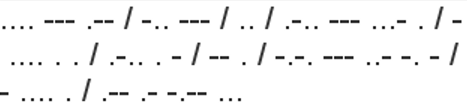
You must add at least 1 test for MorseCodeConverter.convertToEnglish(String) and at least 1 test for MorseCodeConverter.convertToEnglish(File) to the MorseCodeConverterTest class. Include your test file with your code files.

**Test Cases:**

Hello World



How do I love thee let me count the ways



Some suggestions:

1. There is a morse code translator at:

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

it will help you make test cases

**Concepts Utilized in Project**

* Linked Trees
* Building a Tree for conversion purposes
* Generic Classes
* Utility Class (all static methods)

**Write-up & Submission Requirements**

Review the provided rubric (xls file) to understand project expectations, including the documentation, CMSC204, and programming requirements.

In addition to completing the Java project, a write-up is required. At a minimum, the write-up should address -

* Approach, Design & Algorithm
  + DO NOT start coding your project immediately! Come up with a high level design of the project
    - What’s your game plan to tackle the project?
    - Break the project into smallest modules
  + Each student is welcome to expand on the design, if it makes sense. Students will not be penalized for going “above and beyond” the specifications of the project
  + **Complete this step first, then write your code**
* Test Plan & Test Cases
  + Which test cases did you test your program against? (Did you run your program for different type of employees?)
  + What did you do with your program, as far as ensuring that it’s working?
  + I want to see your “thinking,” as to how you are testing your program
  + **Ensure that your project can successfully pass the provided “public test cases.” Your instructor will test your project using a set of “private test cases.”**
  + In theory, each submitted project should be a “rock solid” working program with “near zero” bugs
  + Capture screenshots of most of your test runs in your write-up
* Any assumptions that you are making for this project
* 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
* If the project consists of several applications, include several paragraphs to discuss each application in details in a single write-up. Submit one write-up (Word document or PDF file) for the (entire) project

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

* Source code – one or more **project folder(s)** of all of the code that you have written while working on this project
* Write-up (in Word or PDF – one write-up per project)

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)
* Review provided instructions on how to submit the project carefully (don’t assume anything)
* Double check your submission, as I can only grade what’s being submitted
* I MUST BE able to compile, run and test every submitted project on my computer
* A set of public and private test cases will be test against your project

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

Starting working on each project immediately so that we can discuss any concerns or questions you have!

**Academic Honesty Policy Reminder** – Each submitted project will be compared against other submissions from current and previous semesters.

**Sample Test Runs & Outputs**

None.