# *Programming III (420-B31-HR)*

# *Final Project – The Binary Tree ADT*

Date assigned: Tuesday, November 15, 2016

Date due: Tuesday, December 6, 2016

***Objectives:***

After completing this assignment, the student should be able to:

1. Create and use a BinaryTree object.
2. Use a list ADT.
3. Use a stack ADT.
4. Design unit tests.
5. Use JUnit to test a class.
6. Create a class diagram.

***Problem Specifications:***

The Morse code is a common code that is used to encode message consisting of letters and digits. Each letter consists of a series of dots and dashes; for example, the code for the letter a is .- and the code for the letter b is -... . The complete Morse code table for letters is shown below.

Create a GUI application that will encode a message into Morse code and decode a Morse code string into a text message.

## Morse code for letters

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **a** | **.-** |  | **B** | **-...** |  | **C** | **-.-.** |
| **d** | **-..** |  | **E** | **.** |  | **F** | **..-.** |
| **g** | **--.** |  | **H** | **....** |  | **I** | **..** |
| **j** | **.---** |  | **K** | **-.-** |  | **L** | **.-..** |
| **m** | **--** |  | **N** | **-.** |  | **O** | **---** |
| **p** | **.--.** |  | **Q** | **--.-** |  | **R** | **.-.** |
| **s** | **...** |  | **T** | **-** |  | **U** | **..-** |
| **v** | **...-** |  | **W** | **.--** |  | **X** | **-..-** |
| **y** | **-.--** |  | **Z** | **--..** |  |  |  |

# Design

To store the Morse code table and provide encoding and decoding of text messages, create a **MorseCodeTree** class. You will also create a **CodeVisitor** class that will be used when traversing the tree. The **gray – A04 Classes** folder contains the classes you will need to create and access a binary tree.

The **MorseCodeTree** class will contain a **BinaryTree** object representing the Morse code table. Store each letter of the alphabet in a node of a binary tree of level 5 as shown below. The root node is at level 1 and stores no letter. The left node at level 2 stores the letter e (code is .) and the right node stores the letter t (code is -). The 4 nodes at level 3 store the letters with codes (.., .-,-.--).

To build the tree, read a data file in which each line consists of a letter followed by its code. The letters should be ordered by tree level.

**Encoding a message**: Use a **CodeVisitor** object to create an array of Morse code symbols. Each element of the array represents the code for the letter at that position in the array.

**Decoding a message**: Decode the message using the Morse code tree. Make sure you use a delimiter symbol between coded letters .To find the position for a letter in the tree, scan the code and branch left for a dot and branch right for a dash.

## Morse Code Tree

**The CodeVisitor Class:** This is an implementation of the Visitor interface. The **visit()** method will build the Morse code value of the visited node by travelling back up the path to the root. It will store the code in a String array. A method will be added to get the array of codes. Use a **stack** to build the code.

## Class Diagram

Draw and submit a class diagram that includes all classes involved in your design.

## Pseudocode

Submit the pseudocode for the MorseCodeTree() constructor, which creates the tree from the file, encode(), decode() and the visit() methods.

## Junit Tests

Provide junit test cases:

- to verify that the tree was loaded correctly from the data file

- to thoroughly test the encode() method

- to thoroughly test the decode() method

## The Data file

- each line of the data file consists of a letter followed by its code.

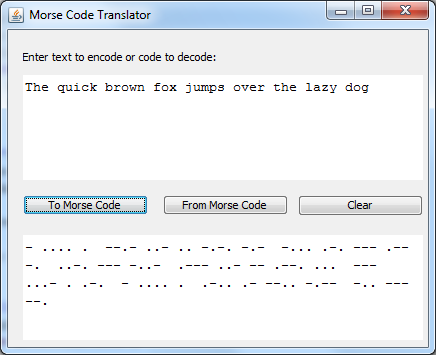
## The GUI Frame

The frame will consist of a prompt, two text areas and three buttons. To enable word wrap in the text areas use the following two **JtextArea** methods.

setLineWrap(**true**)

setWrapStyleWord(**true**)

Your GUI should look similar to:



## Big Oh Analysis

Determine the time complexity for both the encode() method and the decode() method. State the Big Oh notation for these two methods, and explain your reasoning.

***To be uploaded to Moodle:***

The ***username*\_B31\_Final\_Project** folder containing the Junit test cases, the class diagram and the java files for the project.

**Marking Scheme**

|  |  |
| --- | --- |
|  | **Out of** |
| **MorseCodeTree** class  - contains BinaryTree for Morse code table – left branch of each node is for a ‘.’, right branch is for a ‘-‘  - tree read in from a data file  - encode() – uses CodeVisitor  - decode() – traverses tree to find letter | 35 |
| **CodeVisitor** class  - implements Visitor  - visit() – uses stack to build code  - creates an array of Morse code symbols – each element represents the code the letter at that position | 15 |
| **Junit** tests  - verifies that tree was loaded correctly from the data file  - tests encode()  - tests decode() | 15 |
| **GUI frame**  - includes prompt, two text areas, three buttons | 10 |
| Correct Program execution | 10 |
| Class diagram | 5 |
| Pseudocode - MorseCodeTree() constructor, encode(), decode(), visit() | 10 |
| Big Oh Analysis | 5 |
| Organization | 5 |
| **Total** | **110** |

***Organization Marks:***

Marks will be given for organization. This includes:

* naming files and folders according to the department standards
* giving meaningful names to variables, classes, objects and methods
* formatting and indenting Java classes using the Eclipse format tool
* submitting the assignment correctly on **Moodle**
* including all required files in the submitted assignment folder