Elizabeth Perez

CMSC 204

Professor Monshi

12/3/2020

Design Document

|  |
| --- |
| TreeNode<T> |
| T data  TreeNode<T> leftChild  TreeNode<T> rightChild |
| TreeNode(T dataNode)  Set data to dataNode  Set leftChild and rightChild to null  TreeNode(TreeNode<T> node)  Set data to node’s data  Set leftChild to node’s leftChild  Set rightChild to node’s rightChild  getData()  Return data  GetLeftChild()  Return leftChild  GetRightChild()  Return rightChild  SetLeftChild(TreeNode<T> child)  Set leftChild to child  SetRightChild(TreeNode<T> child)  Set rightChild to child |

|  |
| --- |
| MorseCodeTree implements LinkedConverterTreeInterface<String> |
| TreeNode<String> root |
| MorseCodeTree()  Call buildTree()  getRoot()  Return root  setRoot(TreeNode<String> newNode)  Set root to newNode  insert(String code, String result)  Call addNode(root, code, result)  addNode(TreeNode<String> root, String code, String letter)  Declare and set a TreeNode<String> called parent to root  Declare and set a TreeNode<String> called child to new TreeNode with letter  For i less than code length-1  If character in code at index i is ‘.’  Set parent to node’s leftChild  If character in code at index i is ‘-’  Set parent to node’s rightChild  If the last character of the code is ‘.’  Set left child of parent to child  Else  Set right child of parent to child  fetch(String code)  Return fetchNode(root, code)  fetchNode(TreeNode<String> root, String code)  Declare and set a TreeNode<String> called node to root  For i less than code length  If character in code at index i is ‘.’  Set node to node’s leftChild  If character in code at index i is ‘-’  Set node to node’s rightChild  Return node’s data  delete(String data)  Throw UnsuportedOperationException  update()  Throw UnsuportedOperationException  buildTree()  Call insert(“.”, “e”)  Call insert(“-”, “t”)  Call insert(“..”, “i”)  Call insert (“.-”, “a”)  Call insert (“-.”, “n”)  Call insert (“--”, “m”)  Call insert (“...”, “s”)  Call insert (“..-”, “u”)  Call insert (“.-.”, “r”)  Call insert (“.--”, “w”)  Call insert (“-..”, “d”)  Call insert (“-.-”, “k”)  Call insert (“--.”, “g”)  Call insert (“---”, “o”)  Call insert (“....”, “h”)  Call insert (“...-”, “v”)  Call insert (“..-.”, “f”)  Call insert (“.-..”, “l”)  Call insert (“.--.”, “p”)  Call insert (“.---”, “j”)  Call insert (“-...”, “b”)  Call insert (“-.-.”, “c”)  Call insert (“-..-”, “x”)  Call insert (“-.--”, “y”)  Call insert (“--.-”, “q”)  Call insert (“--..”, “z”)  toArrayList()  Declare and initialize an ArrayList<String> called list  Call LNRoutputTraversal(root, list)  Return list  LNRoutputTraversal(TreeNode<String> root, ArrayList<String> list)  If root’s left child is not null  Call LNRoutputTraversal(root’s left child, list)  Add rootNode’s data to list  If root’s right child is not null  Call LNRoutputTraversal(root’s right child, list) |

|  |
| --- |
| MorseCodeConverter |
| MorseCodeTree tree |
| MorseCodeConverter()  Initialize tree to new MorseCodeTree()  convertToEnglish(File codeFile)  Create a Scanner for codeFile called input  Create a String called output and initialize to “”  Create a String called word and initialize to “”  While input hasNext is true  Declare a String called code and initialize to input.next()  If code is ‘/’  Add word to output  Add a space to output  Else  Add tree.fetch(code) to word  Return output  convertToEnglish(String code)  Create a String called output and initialize to “”  Create a String called word and initialize to “”  Declare a String called letter and initialize to “”  For i less than code length  Declare a char called c and set to code’s char at index i  If c is ‘ ’  Add tree.fetch(letter) to word  Set letter to blank  Else if c is ‘/’  Add word to output plus a space  Set word to blank  Else  Add c to letter  Add word to output  Return output  printTree()  Declare an ArrayList<String> called list and set to tree.toArrayList()  Declare a String called s and initialize to “”  For i less than list size  Add list at index i to s  Return s |