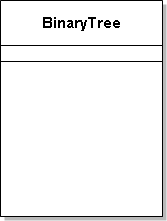
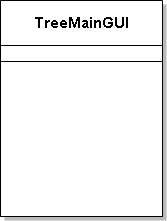
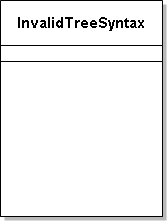
Sairam Soundararajan

CMSC350: Data Structures and Analysis

University of Maryland Global Campus

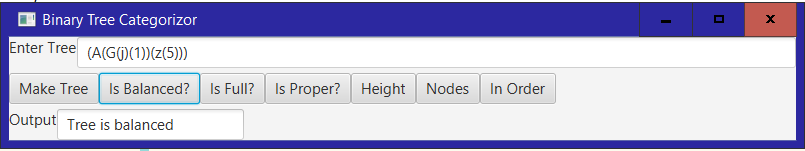
Professor Specioso

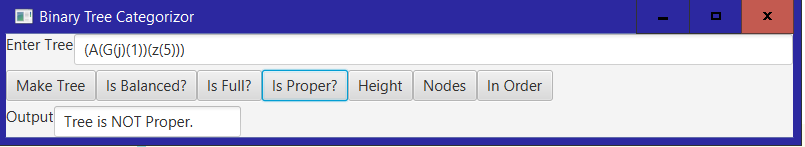
**UML Diagram:**

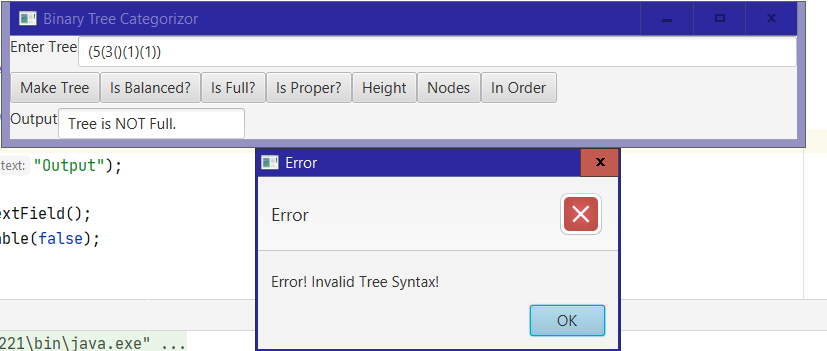
  

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Input | Expected Output | Actual Output | Pass? |
| 1 | Enter Tree: (A(G(j)(1))(z(5)))  Make Tree –  Is Balanced –  Is Full? –  Is Proper? –  Height –  Nodes –  In Order – | Output: Tree Successfully Created  Output: Tree is balanced  Output: Tree is NOT Full.  Output: Tree is NOT Proper.  Output: Max tree height is 2  Output: Number of Nodes in tree is 6  Output: (((j)G(1))A((5)z)) | Output: Tree Successfully Created  Output: Tree is balanced  Output: Tree is NOT Full.  Output: Tree is NOT Proper.  Output: Max tree height is 2  Output: Number of Nodes in tree is 6  Output: (((j)G(1))A((5)z)) | YES |
| 2 | Enter Tree: (4(2(3)(1))(6(5)))  Make Tree –  Is Balanced –  Is Full? –  Is Proper? –  Height –  Nodes –  In Order – | Output: Tree Successfully Created!!  Output: Tree is balanced  Output: Tree is NOT Full.  Output: Tree is NOT Proper.  Output: Max tree height is 2  Output: Number of Nodes in tree is 6  Output: (((3)2(1))4((5)6)) | Output: Tree Successfully Created!!  Output: Tree is balanced  Output: Tree is NOT Full.  Output: Tree is NOT Proper.  Output: Max tree height is 2  Output: Number of Nodes in tree is 6  Output: (((3)2(1))4((5)6)) | Yes |
| 3 | Enter Tree: (5(6(4)(3))(7(2)(1)))  Make Tree –  Is Balanced –  Is Full? –  Is Proper? –  Height –  Nodes –  In Order – | Output: Tree Successfully Created!!  Output: Tree is balanced  Output: Tree is Full!!  Output: Tree is Proper!!  Output: Max tree height is 2  Output: Number of Nodes in tree is 7  Output: (((4)6(3))5((2)7(1))) | Output: Tree Successfully Created!!  Output: Tree is balanced  Output: Tree is Full!!  Output: Tree is Proper!!  Output: Max tree height is 2  Output: Number of Nodes in tree is 7  Output: (((4)6(3))5((2)7(1))) | Yes |

Snapshots of Test Run:







Reflection:

For this project, I reviewed the pictorial concepts of the binary tree and what kinds of traversals are used. For this particular program, I only used Inorder traversal. When it came to coding, working on the BinaryTree class was the lengthiest, as there were many essential functions needed for the binary tree itself to work. The BinaryTree class has a HashSet, recursive method, arraylist, a nested static node class, multiple Boolean, int, and void methods, and a constructor that throws a checked exception. For the GUI, I decided to use JavaFX. By recalling many of the functions in previous GUI’s I’ve made in JavaFX, I was able to get a quicker start on the GUI and took less time to work on it. Last but not least, the InvalidTreeSyntax was the most straight-forward part of working on this project. One thing I have learned while working on this project is that for in-order traversal, the left subtree is visited first, then the root, and then the right subtree.