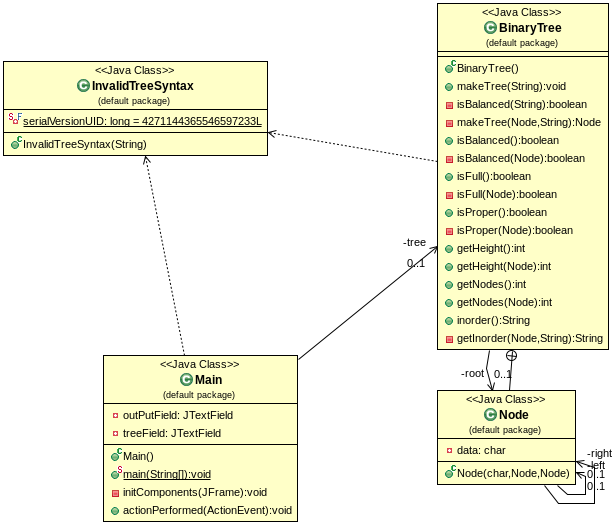
CMSC 350 Project 3

# Assumption

In tree and subtree, the left and right child must be separated by a space.

Ex: (A(G(j) (1)) (z(5)))

# Class Diagram



# Test Cases

1. **Test case 1: Test cases include balanced and unbalanced binary trees**

**Test case** 1a:

1. In the **Enter Tree** field type “(B(D(E) (2)) (R(6)))”
2. Click the **Make Tree** Button
3. After clicking the **Is Balanced?** button it should display the message in **Output** field “Tree is balanced”

**Test case** 1b:

1. In the **Enter Tree** field type “(1(2(3(4)) (5))”
2. Click the **Make Tree** Button
3. An error message in JOptionPane should be shown as “Expression is not balanced

**Test case** 1c:

1. In the **Enter Tree** field type “(A(B(1) (2)) (C(3) (4)))”
2. Click the **Make Tree** Button
3. After clicking the **Is Balanced?** button it should display the message in **Output** field “Tree is balanced”

**Test case** 1d:

1. In the **Enter Tree** field type “(A(G(j(3(6)))))”
2. Click the **Make Tree** Button
3. After clicking the **Is Balanced?** button it should display the message in **Output** field “Tree is not balanced”
4. **Test case 2: Test cases include binary trees that are full and trees that are not**

**Test case** 2a:

1. In the **Enter Tree** field type “(1(2(A) (B)) (3(C) (D)))”
2. Click the **Make Tree** Button
3. After clicking the **Is Full?** button it should display the message in **Output** field “Tree is full”

**Test case** 2b:

1. In the **Enter Tree** field type “(A(G(j) (1)) (z(5)))”
2. Click the **Make Tree** Button
3. After clicking the **Is Full?** button it should display the message in **Output** field “Tree is not full”
4. **Test case 3: Test cases include binary trees that are proper and trees that are not**

**Test case** 3a:

1. In the **Enter Tree** field type “(1(2(A) (B)) (3(C) (D)))”
2. Click the **Make Tree** Button
3. After clicking the **Is Proper?** button it should display the message in **Output** field “Tree is proper”

**Test case** 3b:

1. In the **Enter Tree** field type “(A(G(j) (1)) (z(5)))”
2. Click the **Make Tree** Button
3. After clicking the **Is Proper?** button it should display the message in **Output** field “Tree is not proper”

**Test case 3b:**

1. In the **Enter Tree** field type “(A(G(j) (1)) (z))”
2. Click the **Make Tree** Button
3. After clicking the **Is Proper?** button it should display the message in **Output** field “Tree is proper”
4. **Test case 4: Test cases include a variety of input strings with syntax errors**

`**Test case** 4a:

1. In the **Enter Tree** field type nothing (leave as an empty string)
2. Click the **Make Tree** Button
3. An error message in a JOptionPane should be shown as “Invalid Expression.”

**Test case** 4b:

1. In the **Enter Tree** field type “344/\*”
2. Click the **Make Tree** Button
3. An error message in a JOptionPane should be shown as “Invalid characters in the expression”

**Test case** 4c:

1. In the **Enter Tree** field type “(+(1) (2))”
2. Click the **Make Tree** Button
3. An error message in a JOptionPane should be shown as “Invalid characters in the expression”

**Learning**

In this assignment I learnt about the construction of binary trees, different properties of binary tree and binary tree traversal. This project was useful in understanding how to relate, use, and implement data structures in the context of binary trees. Not only did I learn more about binary trees and data structures, this project built on and expanded upon previously explored programming topics, such as exceptions, GUI, and recursion. This project, a useful application of binary trees, gave me the opportunity to discover more about recursion. Following this project, I have a better working knowledge of how to write recursive methods. Some challenges I encountered were correctly determining the height of the binary tree and whether the binary tree is full or proper, but by effectively using data structures like stacks, I was able to work through the algorithm.