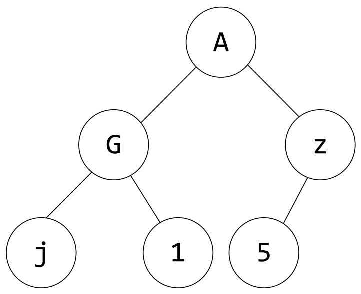
CMSC 350 Project 3

The third programming project involves writing a program that allows the user to enter a binary tree in a parenthesized prefix format and then allows it to be categorized and allows various features of that tree to be displayed. An example of a tree written in the input format is the following:

`(A(G(j)(1))(z(5)))`

In the above example, data in each node of the tree contains a single alphanumeric character. No spaces are permitted. Each tree is enclosed in parentheses. Inside those parentheses, after the single character are either zero, one or two subtrees also enclosed in parentheses. When only one subtree is present, it is the left subtree and when two are present, they represent the left and right subtrees. So the above string represents the following binary tree:



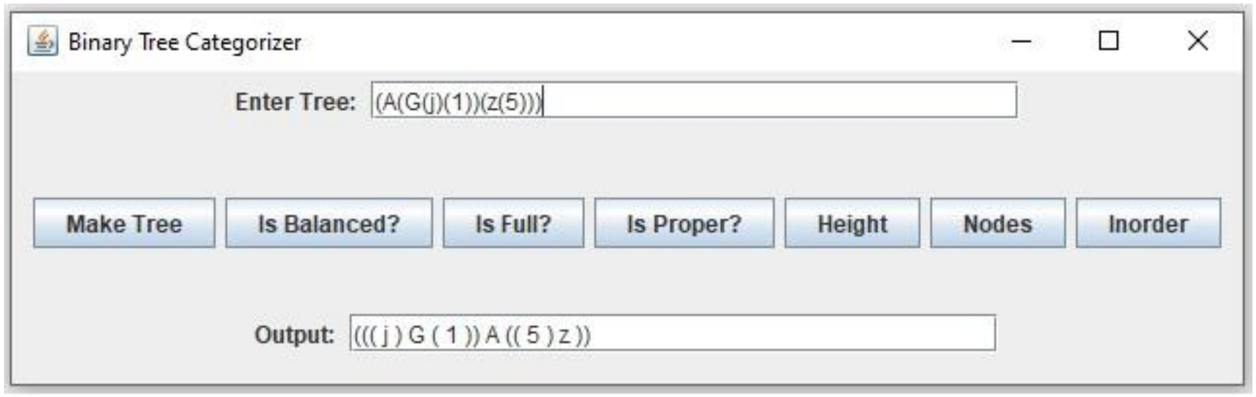
The various categorizations include the following:

1. Whether the binary tree is balanced, which means for each node in the tree, the absolute difference between the height of its left and right subtrees is at most 1. The above binary tree is balanced.
2. Whether the binary tree is full. A full binary tree has the maximum number of nodes for a tree of its height. The above tree is not full because a tree of that height can contain 7 nodes, but the above tree only has 6.
3. Whether the binary tree is proper. In a proper binary tree, every node has either 0 or 2 children. The above tree is not proper because the node containing `z` has only one child.

In addition, the program should allow the user to request that each of the following features of the tree be displayed:

1. The height of the tree. The height of a tree is the maximum level of all of its nodes. The root node containing `A` is at the level 0. Because all three leaf nodes in the above tree are at level 2, its height is 2.
2. The number of nodes in the tree. As previously mentioned, the above tree has 6 nodes.
3. An fully parenthesized inorder traversal of the tree. The following should be displayed as the inorder traversal of the above tree: `((( j ) G ( 1 )) A (( 5 ) z ))`

This project should consist of three classes. The main class should create a GUI that allows the user to input a tree in the above described format and then construct the tree once the *Make Tree* button is clicked. The GUI should look as follows:



Enter Tree: (A(G(j)(1))(z(5)))

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

The GUI must be generated by code that you write. You may not use a drag-and-drop GUI generator.

The second class should be the `BinaryTree` class, which should contain a static nested class to define the nodes of the binary tree, together with a constructor that is called when the *Make Tree* button is clicked and is supplied the string representation of the tree and constructs the actual tree from that string. In addition, it should have public methods that are called when each of the remaining six buttons are clicked. All of those public methods should have corresponding private methods that accomplish their tasks using recursion.

The third class should be named `InvalidTreeSyntax`, which defines a checked exception. It should be thrown when a invalid string is supplied and the *Make Tree* button is clicked. It should be caught in the main class and a `JOptionPane` window should be displayed that describes the reason for the invalid syntax.

You are to submit two files.

1. The first is a `.zip` file that contains all the source code for the project. The `.zip` file should contain only source code and nothing else, which means only the `.java` files. If you elect to use a package the `.java` files should be in a folder whose name is the package name. Every outer class should be in a separate `.java` file with the same name as the class name. Each file should include a comment block at the top containing your name, the project name, the date, and a short description of the class contained in that file.
2. The second is a Word document (PDF or RTF is also acceptable) that contains the documentation for the project, which should include the following:
   a. A UML class diagram that includes all classes you wrote. Do not include predefined classes. You need only include the class name for each individual class, not the variables or methods
   b. A test plan that includes test cases that you have created indicating what aspects of the program each one is testing
   c. A short paragraph on lessons learned from the project

Grading Rubric

Criteria	Meets	Does Not Meet
Design	**20 points**	**0 points**
	GUI is hand coded and matches required design (5)	GUI is generated by a GUI generator or does not match required design (0)
	BinaryTree class contains a static nested class to define the nodes and all the required methods (5)	BinaryTree class does not contain a static nested class to define the nodes and all the required methods (0)
	InvalidTreeSyntax class defines a checked exception as specified (5)	InvalidTreeSyntax class does not define a checked exception as specified (0)
	Recursion is used to accomplish each binary tree operation (5)	Recursion is not used to accomplish each binary tree operation (0)
Functionality	**60 points**	**0 points**
	Correctly builds a binary tree from its parenthesized representation (10)	Does not correctly build a binary tree from its parenthesized representation (0)
	Correctly detects syntax errors in the parenthesized representation, and throws an exception, catches it and displays an error (10)	Does not correctly detect syntax errors in the parenthesized representation, or does not throw an exception, catch it and display an error (0)
	Correctly determines whether a binary tree is balanced (10)	Does not correctly determine whether a binary tree is balanced (0)
	Correctly determines whether a binary tree is full or proper (10)	Does not correctly determine whether a binary tree is full or proper (0)
	Correctly calculates the height and number of nodes in a binary tree (10)	Does not correctly calculate the height or number of nodes in a binary tree (0)
	Correctly displays the binary tree with an inorder representation (10)	Does not correctly display the binary tree with an inorder representation (0)
Test Plan	**10 points**	**0 points**
	Test cases include balanced and unbalanced binary trees (3)	Test cases do not include balanced and unbalanced binary trees (0)
	Test cases include binary trees that are full and trees that are not (2)	Test cases do not include binary trees that are full and trees that are not (0)
	Test cases include binary trees that are proper and trees that are not (2)	Test cases do not include binary trees that are proper and trees that are not (0)
	Test cases include a variety of input strings with syntax errors (3)	Test cases do not include a variety of input strings with syntax errors (0)
Documentation	**10 points**	**0 points**

	Correct UML diagram included (3)	Correct UML diagram not included (0)
	Lessons learned included (4)	Lessons learned not included (0)
	Comment blocks included with each Java file (3)	Comment blocks not included with each Java file(0)