# Important

There are general homework guidelines you must always follow. If you fail to follow any of the following guidelines you risk receiving a **0** for the entire assignment.

1. All submitted code must compile under **JDK 8**. This includes unused code, so don't submit extra files that don't compile. (Java is backwards compatabile so if it compiles under JDK 6 it *should* compile under JDK 8.)
2. Do not include any package declarations in your classes.
3. Do not change any existing class headers, constructors, or method signatures.
4. Do not add additional public methods when implementing an interface.
5. Do not use anything that would trivialize the assignment. (e.g. don't import/use `java.util.LinkedList` for a Linked List assignment. Ask if you are unsure.)
6. You must submit your source code, the `.java` files, not the compiled `.class` files.
7. After you submit your files redownload them and run them to make sure they are what you intended to submit. You are responsible if you submit the wrong files.

# AVL Tree

You are to code an AVL tree. Like a BST, and AVL Tree is a collection of nodes organized such that left child nodes (and all of its children) are less than the data and the right child nodes and all of its children are greater than the data. Unlike the BST class, the AVL tree must be self-balancing, so that methods have a worst case O(logn) performance instead of O(n).

Your AVL tree implementation will implement the AVL interface provided. It will have the public default constructor only - the one with no parameters. You will not write the default constructor, as it is automatically provided by Java. So do not write any constructors.

## Nodes

The AVL consists of nodes. The Node class will be given to you; do not modify it. Each node has two new instance variables that were not present in the BST homework: bf (balance factor) and height. Height represents the height of the node; this is calculated using the same formula as the height for the tree. Balance factor is equal to left.height - right.height (use -1 for null children). These instance variables need to be updated accordingly, but be careful to make sure everything still has O(logn) performance rather than O(n)

## Methods

You will implement all standard methods for a java data structure (add, remove, etc.) See the interface for details.

## Traversals

You will implement 4 different ways of traversing a tree: pre-order traversal, in-order traversal, post-order traversal, and level-order traversal. The first 3 MUST be implemented recursively; level-order may be implemented iteratively. You may import Java's LinkedList/ArrayList classes as approriate for these methods (but they may only be used for these methods).

### Height

You will implement a method to calculate the height of the tree. The height of any given node is max(child nodes height)+1. A leaf node has a height of 0.

## Style and Formatting

It is important that your code is not only functional but is also written clearly and with good style. We will be checking your code against a style checker that we are providing. It is located in resources along with instructions on how to use it. We will take off a point for every style error that occurs. If you feel like what you wrote is in accordance with good style but still sets off the style checker please email Jonathan Jemson (jonathanjemson@gatech.edu) with the subject header of "CheckStyle XML". **Please make sure that you are using the CheckStyle XML file named CS1332-checkstyle-v2.xml. All homework assignments will use this CheckStyle.**

### Javadocs

Javadoc any helper methods you create in a style similar to the Javadocs for the methods in the interface.

## Provided

The following file(s) have been provided to you. There are several, but you will only edit four of them.

1. `AVLInterface.java` This is the interface you will implement. All instructions for what the methods should do are in the javadocs. **Do not alter this file.**
2. `Gradable.java` This is another interface you will implement. All instructions for what the methods should do are in the javadocs. **Do not alter this file.**
3. `AVL.java` This is the class in which you will actually implement the interfaces. Feel free to add private helpers but **do not add any new public methods.**
4. `Node.java` This class encapsulates the data and the left/right references. **Do not alter this file.**
5. `AVLTestStudent.java` This is the test class that contains a set of tests covering the basic operations on the AVL. It is not intended to be exhaustive nor guarantee any type of grade. **Write your own tests to ensure you cover all edge cases.**

## Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder, copy over the interfaces, recompile, and run. It is your responsibility to re-test your submission and discover editing oddities, upload issues, etc.

1. `AVL.java`

You may attach each file individually, or submit them in a zip archive.