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. Any compile errors will result in a 0.
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.
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. Always be very conscious of efficiency. Even if your method is to be $O(n)$, traversing the structure multiple times is considered non-efficient unless that is absolutely required (and that case is extremely rare).
7. You must submit your source code, the `.java` files, not the compiled `.class` files.
8. 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.

Binary Search Tree

You are to code a Binary Search Tree. A binary search tree is a collection of nodes, each having a data item and a reference pointing to the left and a reference pointing to the right child nodes. The nodes in the left sub-tree are less than the node in the parent node and the nodes in the right sub-tree are greater than or equal to the node in the parent node. For this assignment, any attempts to add data already present in the tree should be ignored. All elements added to the tree must implement Java's generic `Comparable` interface. **All methods in the BST that are not O(1) must be implemented recursively** (except `levelorder`, since it's best implemented iteratively) .

Your binary search tree implementation will implement the BST interface provided. It will have two constructors: the no-argument constructor (which should initialize an empty tree), and a constructor that initializes a tree using the data in the given collection.

Nodes

A class `BSTNode` is provided to you. The BST consists of nodes having `data` of type `T`, a `BSTNode` reference called `left` for the left child and a `BSTNode` reference called `right` for the right child .

Methods

You will implement all standard methods for a Java data structure (add, remove, get etc.) and some utility methods. See the interface for more 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 is best implemented iteratively. You may import Java's LinkedList/ArrayList classes as appropriate for these methods (but they may only be used for these methods).

Pre-Order Traversal: visits the root before visiting the node in either sub-tree.

In-Order Traversal: visits all the nodes of the tree in sorted order.

Post-Order Traversal: visits the root node after visiting the nodes in either sub-tree.

Lever-Order Traversal: every node at a level is visited before going to a lower level.

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`. The height of the tree is the height of the root node. A leaf node has a height of 0.

A note on JUnits

We have provided a **very basic** set of tests for your code, in `BSTStudentTests.java`. These tests do not guarantee the correctness of your code (by any measure), nor does it guarantee you any grade. You may additionally post your own set of tests for others to use on the Georgia Tech GitHub as a gist. Do **NOT** post your tests on the public GitHub. There will be a link to the Georgia Tech GitHub as well as a list of JUnits other students have posted on the class Piazza (when it comes up).

If you need help on running JUnits, there is a guide, available on T-Square under Resources, to help you run JUnits on the command line or in IntelliJ.

Generics

If available, use the generic type of the class; do **not** use the raw type of the class. For example, use `new BST<Integer>()` instead of `new BST()`. Using the raw type of the class will result in a penalty.

Forbidden Statements

You may not use these in your code at any time in CS 1332. If you use these, we will take off points.

- `break` may only be used in switch-case statements
- `continue`
- `package`
- `System.arraycopy()`
- `clone()`
- `assert()`
- `Arrays` class
- `Array` class
- `Collections` class
- `Collection.toArray()`
- Reflection APIs
- Inner or nested classes

Debug print statements are fine, but nothing should be printed when we run them. We expect clean runs - printing to the console when we're grading will result in a penalty.

Provided

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

1. `BSTInterface.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. `BST.java`

   This is the class in which you will implement the interface. Feel free to add private helper methods but **do not add any new public methods, inner/nested classes, instance variables, or static variables**.

3. `BSTNode.java`

   This class represents a single node in the BST. It encapsulates the `data`, the `left` reference and the `right` reference. **Do not alter this file.**

4. `BSTStudentTests.java`

   This is the test class that contains a set of tests covering the basic operations on the `BST` class. It is not intended to be exhaustive and does not 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. `BST.java`