Important

There are a few guidelines you must follow in this homework. If you fail to follow any of the following guidelines you will receive a **0** for the entire assignment.

1. All submitted code must compile under **JDK 7**. This includes unused code, 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 7)
2. Don't include any package declarations in your classes.
3. Don't change any *existing* class headers or method signatures. (It is fine to add extra methods and classes)
4. Don't import anything that would trivialize the assignment. (e.g. don't import `java.util.LinkedList` for a Linked List assignment. Ask if you are unsure.)
5. You must submit your source code, the `.java` files, not the compiled `.class` files.

After you submit your files redownload them and run them to make sure they are what you intended to submit. We are not responsible if you submit the wrong files.

Assignment

In this assignment you will be coding a binary search tree. There are two ways ways this is usually implemented, iteratively, or recursively. I *strongly* recommend coding this recursively, it will help you in future assignments. This is also the implementation I will be describing in this document.

It is okay to code the iterative solution, you will not be penalized for it.

Fields

You should have a root node, and a size field.

Add (Recursive)

In order to do this recursively you need to have a private helper method. The recommended signature being:

```
private Node<T> add(Node<T> curr, T item) {...}
```

This method adds `item` to the subtree with root `curr`, then returns the potentially new root of the subtree.

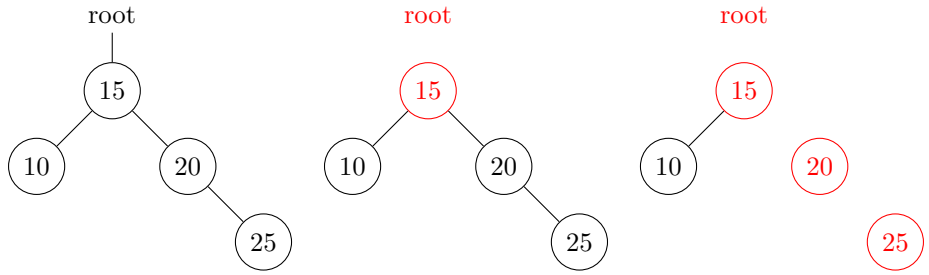
Additionally when recursively traversing the tree inside this method, update the left or right of `curr` to what is returned from the next recursive call of add. (This concept of rebuilding the tree with what is returned from the recursive calls is important in simplifying the code and base cases)

```
curr.setRight(add(curr.getRight(), item));
```

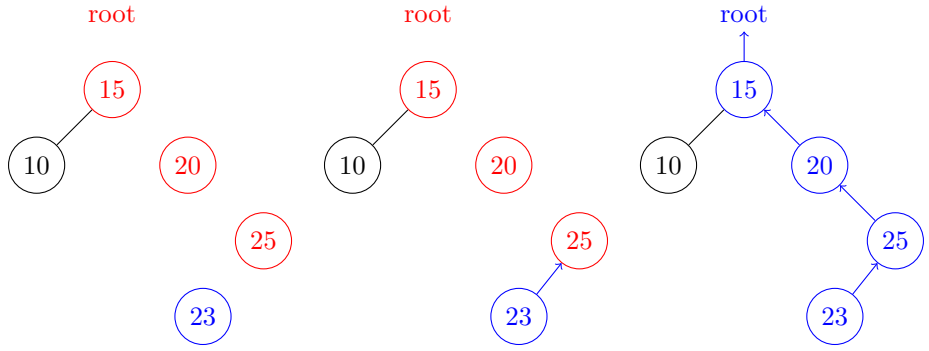
The base case for this recursive method is when `curr` is null, in that case the new root of the tree would be a new node containing `item`, so that's what should be returned.

Here is a simple diagram giving an overview of what this method does.

Initial Tree | add(23) | recurse down the tree



Base case, create node | return new node | return up the tree



Traversals

You will be coding 3 different traversals in your BST. They will each need their own private helper method. Here's a method signature that would work:

```
private List<T> preOrder(List<T> list, Node<T> curr) {...}
```

Assumptions

1. You may assume no parameter passed into a method will ever be null.
2. You may assume that no duplicates will be added to the tree.

Java-docs

You are always required to javadoc methods in assignments; however, unlike previous assignments we did not provide any java-docs for the methods in the assignment. It is your job to add them to every method in BinarySearchTree. (Don't add them to SearchTree.java)

Deliverables

You must submit all of the following files.

1. `BinarySearchTree.java`

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