Unit 9 and Unit 10: Programming
Implementing an Array-Based Binary Search Tree

Summary: In this assignment, you will complete the implementation of a BinarySearchTree using a layered design.

1 Background

In order to practice non-linear collections, you practice implementing a binary search tree. Similar to the unit on Lists, this ADT is designed using a layered approached. There are several classes that implement a list (ArrayList and ArrayUnorderedList), followed a binary tree built with a list (LinkedBinaryTree), and then a binary search tree (LinkedBinarySearchTree) built from the basic binary tree.

An UML overview of this system in shown in Figure 1 for the list ADT, and Figure 2for the tree ADT. Although we already spent quite a bit of time on ADTs, there are still a couple of interesting features to notice. First, LinkedBinaryTree represents a non-linear collection, and yet it can be explored in terms of ListADT by its iterator. A list is a linear structure. So, we can see that even a non-linear structure like a binary search tree can be projected down to a linear structure, provided we have a well defined tree traversal algorithm. Second, for the most part, the LinkedBinarySearchTree class doesn't need any changes to work. If it inherits from LinkedBinaryTree, then all methods will function properly. Quite literately, a binary search tree is a binary tree. However, the class won't have optimal implementations for its operations. The advantage of a BST is we can make an additional assumption (how children are ordered) in the algorithms for manipulating a tree, and so speed up the ADT.

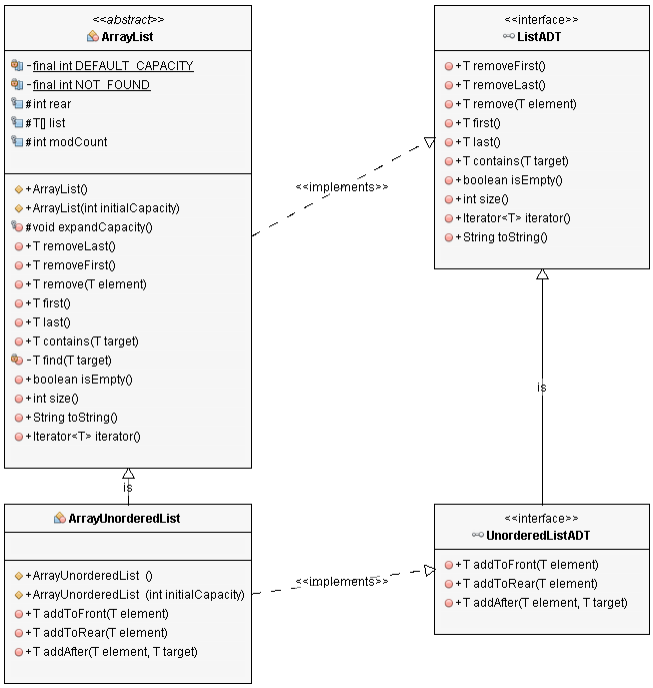

Figure 1: List UML Overview

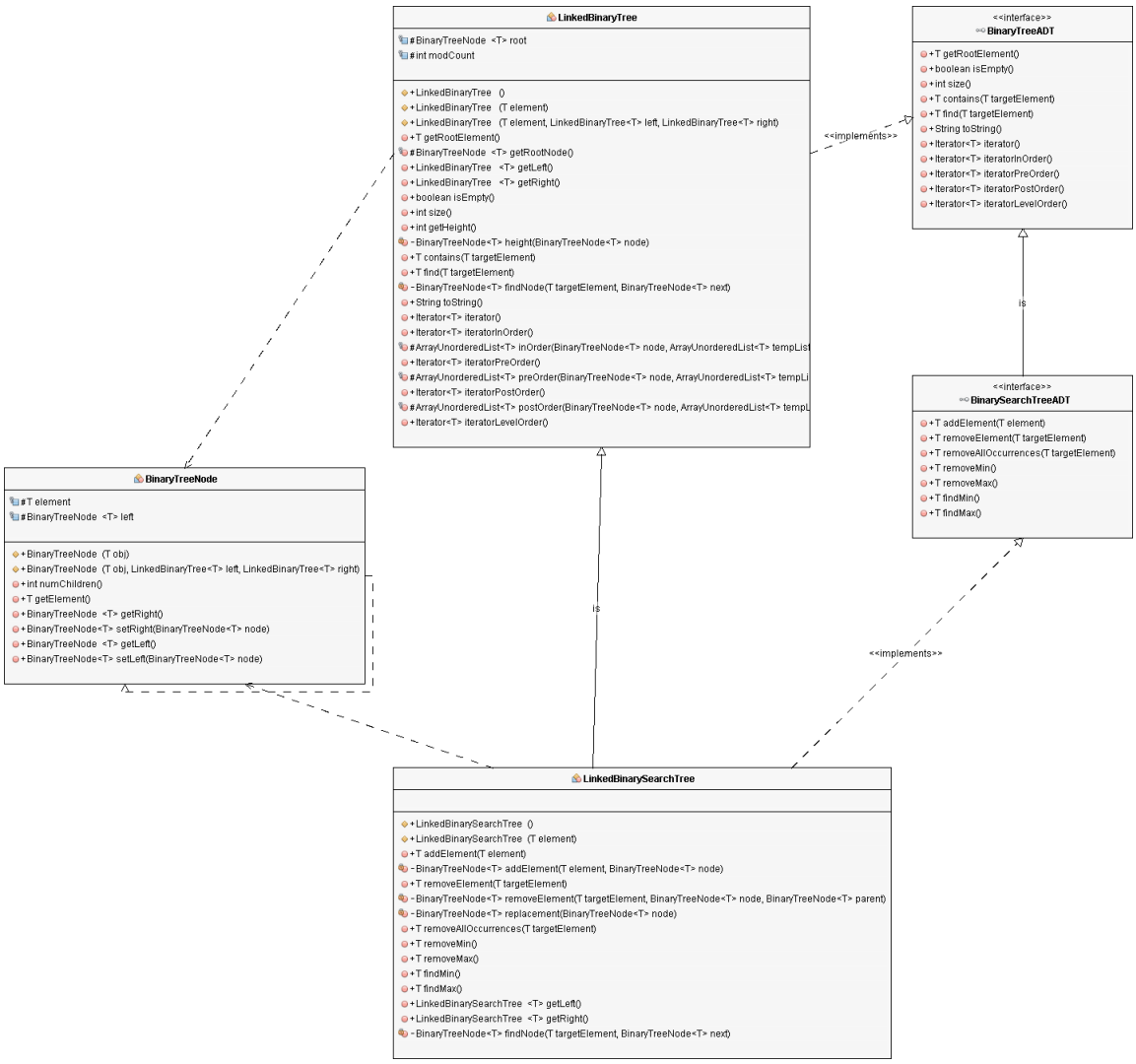

Figure 2: Tree UML Overview

This document is separated into four sections: Background, Requirements, Testing, and Submission. You have almost finished reading the Background section already. In Requirements, we will discuss what is expected of you in this homework. In Testing, we give some basic suggestions on how the maze algorithm additions should be tested. Lastly, Submission discusses how your source code and writeup should be submitted on BlackBoard.

2 Requirements [50 points]

In this programming project you will practice the implementation of binary search trees. Download the attached ZIP file for a starting place; it includes several interfaces and partial implementations that are base for this ADT. You would only need to change three files: ArrayUnorderedList, LinkedBinaryTree, and LinkedBinarySearchTree (should contain testing code).

- Complete the implementation of ArrayList. Specifically, implement these methods: removeFirst and removeLast. Don't neglect support for the iterator in ArrayList. Do not import any packages other than those already imported. [8 points]

- Complete the implementation of ArrayUnorderedList. Specifically, implement these methods: addToFront and addToRear. Don't neglect support for the iterator in ArrayList. Do not import any packages other than those already imported. [8 points]
- Complete the implementation of LinkedBinaryTree. Specifically, implement these methods: getRootElement(), getRootNode(), getLeft(), getRight(), size(), getHeight(), height(), contains(), toString(), and iteratorPreOrder(). Review the comments in LinkedBinaryTree and BinaryTreeADT for implementation requirements. [12 points]
- [LC PP 11.2 modified] The LinkedBinarySearchTree class is currently using the find and contains methods of the LinkedBinaryTree class via inheritance. This is not optimal! Implement these two methods for the LinkedBinarySearchTree class, find() and contains(), so that they will be more efficient by making use of the ordering property of a binary search tree. Notice that five other methods in LinkedBinarySearchTree are not yet implemented: removeMax(), findMin(), findMax(), getLeft(), getRight(), findNode(). If any are needed by your algorithms, implement them, otherwise you may omit them. [12 points]
- Lastly, write appropriate testing documentation - see next section. [10 points]

3 Testing

For this assignment, you will need to describe how you tested your code.Your aim is to demonstrate that you have anticipated things that might go wrong and done appropriate tests to verify that they do not go wrong. Before giving your specific tests, you should define the purpose of your testing methodology. If you used unit testing, or if you test using sample input and expected-output files, please include the contents of the files. You should include both a discussion of how you tested (e.g., which operations), as well as a screen shot of the output of running your program. Organize this into subsections for different test cases if applicable. The writeup should be clear and well written. Quality over quantity.

When you set about writing your own tests, try to focus on testing the methods in terms of the integrity of the tree and the correctness of performing operations over it. For example, you should test that elements don't disappear when a find is run, iterators contain the correct results, and so on. Another consideration might be benchmarking the differences between a plain binary tree and a binary search tree.

4 Submission

The submission for this assignment has two parts: a testing write up and a source code submission. Both should be attached to the submission link on BlackBoard.

Writeup: Submit a PDF that discusses how you tested your program. Include a header that contains your name, the class, and the assignment. We not are expecting anything longer than a single page. (If it makes your writeup more clear, you may use additional pages.)

Source Code: Please zip all of your source code files together as "LastNameBST.zip" (e.g. "AcunaBST.zip"). It should contain only four files: ArrayList.java, ArrayUnorderedList.java, LinkedBinaryTree.java, and LinkedBinarySearchTree.java. Remember to change the class name as well. Be sure that you use the correct compression format - if you do not use the right format, we may be unable to your submission. Do not include your project files, or leave code in your files that is specific to your IDE/project.