Lab 9

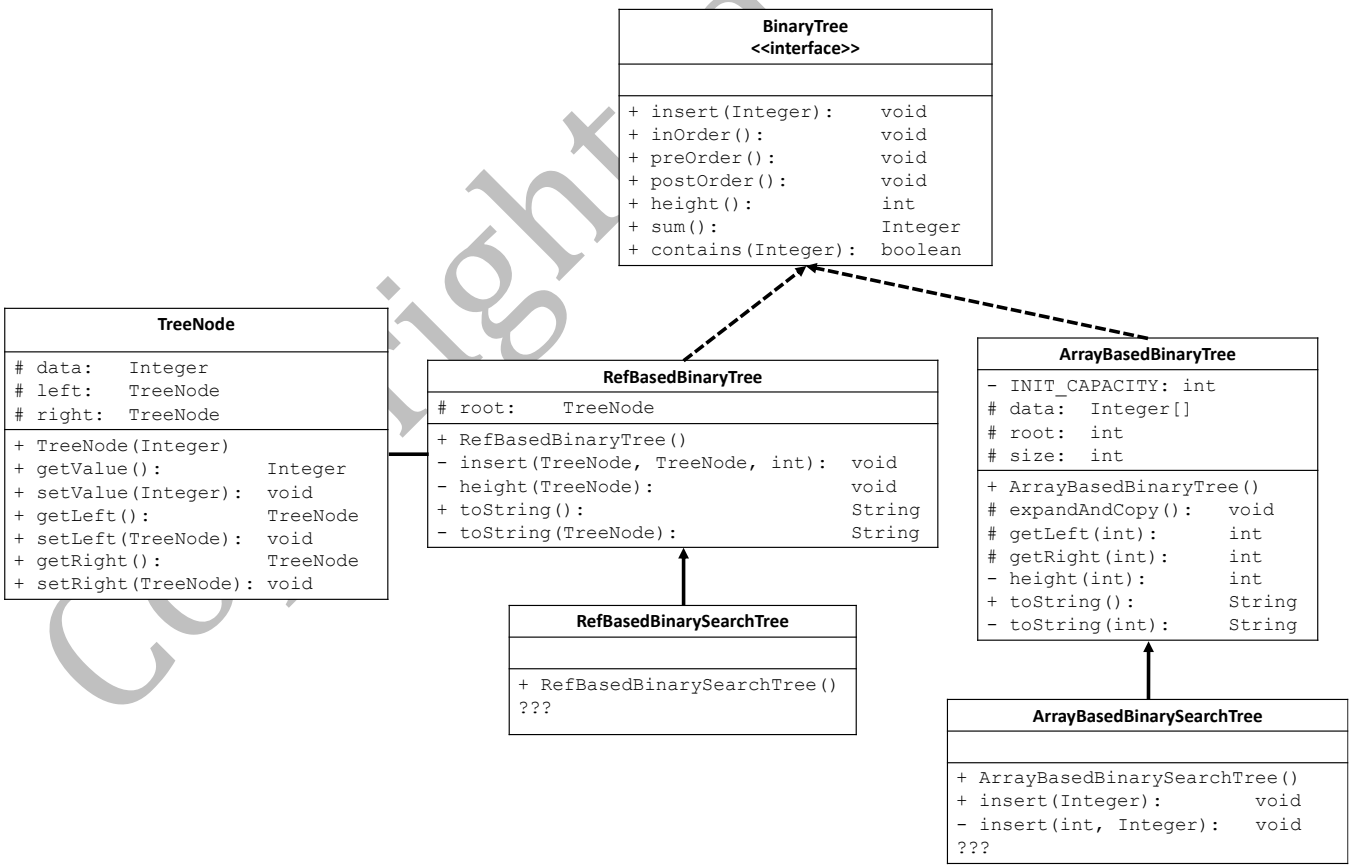
Reminder: Your code is to be designed and written by only you and not to be shared with anyone else. See the Course Outline for details explaining the policies on Academic Integrity. Submissions that violate the Academic Integrity policy will be forwarded directly to the Computer Science Academic Integrity Committee.
All materials provided to you for this work are copyrighted, these and all solutions you create for this work cannot be shared in any form (digital, printed or otherwise). Any violations of this will be investigated and reported to Academic Integrity.

Objectives

- Extending Binary Trees to make Binary Search Trees
- Practice with extending a class and overriding methods

Part I – Extending BinaryTree

In this part of the lab you will implement the `ArrayBasedBinarySearchTree` and `RefBasedBinarySearchTree` classes that will `extend` the `ArrayBasedBinaryTree` and `RefBasedBinaryTree` respectively (as shown in the UML diagram below).
The UML for the `ArrayBasedBinarySearchTree` and `RefBasedBinarySearchTree` classes are intentionally not complete (method lists left as ???) because part of this exercise is to decide which of the methods these classes can inherit from their superclass and which they will need to override!



RECALL: A Binary Search Tree maintains the invariant that for every element in the tree, every element in its left subtree must be smaller than the parent and every element in its right subtree must be larger than the parent. If an element that already exist in the tree is inserted, the tree is updated with the new element.

1. Download all the files provided to you in this lab to your Lab9 folder.
2. You will be implementing the necessary methods in `ArrayBasedBinarySearchTree` that extends `ArrayBasedBinaryTree`
3. Begin by deciding which methods `ArrayBasedBinarySearchTree` can inherit from the superclass and which methods will need to be overridden. Ask yourself, will the algorithm be different given the constraints of a Binary Search Tree. You will want to override the method if the functionality of the superclass implementation is **incorrect** given these constraints **OR** if it can be implemented with a **more efficient** algorithm given these constraints.
   NOTE: we have completed the `insert` method for you in `ArrayBasedBinarySearchTree`
4. Implement the required methods that you must **override** from the superclass.
5. Uncomment the corresponding test method call in the `main` method of `Lab9Tester` to test.

CHECK POINT – get help from your lab TA if you are unable to complete this part.

6. Repeat steps 3-6 for `RefBasedBinarySearchTree`

CHECK POINT – get help from your lab TA if you are unable to complete this part.

Part II – Adding functionality

In this part of the lab you will be adding functionality to your Binary Tree ADT by adding the following two abstract methods:

- `sum` - gets the sum of all the values in the tree
- `contains` - determines whether the tree contains a given value

For each of these two methods do the following:

1. Find the stubs for this method in `ArrayBasedBinaryTree` and `RefBasedBinaryTree` classes. Complete the implementation of the each method.
2. Uncomment the corresponding test method call in the `main` method of `Lab9Tester` to test.
3. Determine whether `ArrayBasedBinarySearchTree` and/or `RefBasedBinarySearchTree` should inherit the implementation from the superclass if it needs to be overridden. Add the method implementation if you believe it should be overridden.
4. Uncomment the corresponding test method call in the `main` method of `Lab9Tester` to test.

CHECK POINT – get help from your lab TA if you are unable to complete this part.