

CSCI 230 HW#6

Collaboration policy: **Individual Assignment**

Total Points: **100**

Source Code

The Java classes provided in the zip file attached to the assignment Dropbox are:

- `BinaryNode.java`
- `BinarySearchTree.java`
- `EmptyBSTException.java`
- `NullBinaryNodeException.java`
- `DuplicateElementException.java`

Under no circumstances are you allowed to modify or create new `BinaryNode`, `DuplicateElementException`, `NullBinaryNodeException`, or `EmptyBSTException` classes. You must use these files **as is**.

You **may only** modify the `BinarySearchTree` class. In particular, in this class you **may only** modify the methods listed in Part 1, and under no circumstances are you allowed to remove, add, or modify any other line of code in this class (this include instance variables, class variables, constants, etc.).

Lastly, you **may not** change the package structure! Specifically, `edu.cofc.csci230` cannot be removed or modified. If a solution is submitted with a different package structure, it will not be graded, no exceptions.

Part 1

In the `BinarySearchTree` class please fully implement the methods listed below:

- `public void add(AnyType element) throws DuplicateElementException`
- `private void add(BinaryNode<AnyType> node, AnyType element) throws DuplicateElementException`
- `private boolean hasElement(BinaryNode<AnyType> node, AnyType element)`
- `private BinaryNode<AnyType> findMin(BinaryNode<AnyType> node)`
- `private BinaryNode<AnyType> findMax(BinaryNode<AnyType> node)`

In each method listed above you will see a `TODO` comment, this is where you add your coding solution. In the provided source code, numerous comments are given; please ensure you read them carefully. Additionally, in the course textbook binary trees and the binary

search tree (including BST operations) examples are provided, and of course you can use your course notes.

Part 2

The provided `BinarySearchTree` class has a main method. In the main please add test cases that demonstrate you have fully evaluated the operational correctness of the methods you implemented in Part 1. To receive full credit, these test cases **must** be included.

Submission

Create a zip file that **only** includes the completed `BinarySearchTree.java` file. If you have any questions about the submission policy, you must resolve before the due date. Lastly, please plan appropriately, asking questions the day the assignment is due (within 12 hours) is too late. Please try to resolve any questions at least 2 days before the due date.

The name of the zip file must be your last name. For example, *ritchie.zip* would be correct if the original co-developer of UNIX (Dennis Ritchie) submitted the assignment. Only assignments submitted in the correct format will be accepted (no exceptions).

Please submit the zip file (via OAKS) to the Dropbox setup for this assignment by the due date. You may resubmit the zip file as many times as you like, Dropbox will only keep the newest submission. Per the syllabus, late assignments will not be accepted – no exceptions. Please do not email Hassam or I your assignment after the due date, we will not accept it.

Grading Rubric

BinarySearchTree Compiles	15 points
Thoroughness of your test cases	5 points
Instructor test cases (8 cases 10 points each)	80 points
	100 points

In particular, each data structure will be graded as follows. If the submitted solution

- Does not compile: 0 of 100 points
- Compiles but does not run: 15 of 100 points
- Thoroughness of your test cases: 20 of 100 points
- Passes all 7 test cases developed by instructor: 100 of 100 points.