4. Requirements for RedBlackTree Class

The class that implements your RedBlackTree for this assignment MUST:

* be a public and instantiable class named RedBlackTree.
* be defined in its own file RedBlackTree.java.
* be declared in the default package (do not include any package statements in your class).
* extend your BSTRotation class from P102 (the BSTRotation class must extend your BinarySearch tree class from P101, and cannot extend BinarySearchTree\_Placeholder).
* use a bounded generic type parameter T to define the data stored in the tree and to specify the type of BSTRotation that is being extended by your RedBlackTree class.
* only makes use of one field: the protected field root that is inherited from the BinarySearchTree class.
* contains a method with the following signature:

/\*\*

\* Checks if a new red node in the RedBlackTree causes a red property violation

\* by having a red parent. If this is not the case, the method terminates without

\* making any changes to the tree. If a red property violation is detected, then

\* the method repairs this violation and any additional red property violations

\* that are generated as a result of the applied repair operation.

\* @param newRedNode a newly inserted red node, or a node turned red by previous repair

\*/

protected void ensureRedProperty(RBTNode<T> newRedNode) {

// TODO: Implement this method.

}

* contains an insert method that:
  + overrides the insert method inherited from BinarySearchTree.
  + uses BinarySearchTree’s insertHelper method to insert a node with a the new value into the tree.
  + ensures that any node inserted into the tree is a red node of type RBTNode.
  + ensures that ensureRedProperty is called for every newly inserted red node (with the exception of the root node) to identify and repair a potential red property violation.
  + sets the color of the root node to black after the insertion of a new red node and any potential red property repair operations to ensure a black root node. Update 27/09: To set the color of the root node, you need to cast the root node to type RBTNode, e.g. ((RBTNode<T>)this.root).isRed
* be clearly organized and consistently styled.

Your class MAY:

* include additional private helper methods that may only be called from within ensureRedProperty.
* make use of other classes from the java standard library.

**Submit your RedBlackTree class that satisfies the above requirements by the midweek deadline (Saturday, September 28). Note that the ensureRedProperty method can be left empty in your midweek submission. To submit, make sure the “P104.RBT” on your VM contains the latest version of your RedBlackTree class. Then submit by running “cs400 submit” from within that folder.**

5. Requirements for ensureRedProperty Method

Your implementation of the ensureRedProperty method MUST:

* detect a red property violation caused by the red node passed in as a parameter having a red parent.
* repair the detected property violation.
* call the rotate method inherited from BSTRotation for any rotation required by the repair operation.
* call itself recursively to detect and resolve a potential additional red property violation that results from the repair if this is required by the specific repair operation used.
* ensure that the contains and insert methods of a RedBlackTree with N nodes have a worst case runtime complexity of O(log N) by following the specific repair algorithm discussed during lecture.
* be clearly organized, consistently styled, and well commented.

Your method MAY:

* call additional private helper methods.
* make use of classes from the java standard library.

6. Testing Requirements

Add at least three JUnit test methods to your RedBlackTree class that can be run with the JUnit 5 test runner. Those three test methods should:

* each test a different case for inserting into a Red-Black Tree.
* at least one of them should implement an example insertion that you worked through in the Q03.RBTInsert quiz from last week, and you are welcome to use additional examples from the quiz if you’d like.
* check that your ensureRedProperty method correctly follows the repair algorithm discussed during lecture.
* include in-line comments to document the steps of your test code.
* include a JavaDoc header comment that describes what specific case each method is testing, and, if it implements an example from the quiz, indicate which quiz question the example is from.