Lab 6 (Binary Search Tree)

CSC 172 (Data Structures and Algorithms)
Fall 2023
University of Rochester
Due Date: Sunday, October 29th @ end of day

Introduction

The labs in CSC172 will follow a pair programming paradigm. Every student is encouraged (but not strictly required) to have a lab partner. Every student must hand in their own work but also list the name of their lab partner if any on all labs.

In this lab, you will work on the Binary Search Tree ADT. Start with the `UR_BST` class code below.[1] Make a concrete class. HINT: There are obvious helper methods that can be used to facilitate recursion. Follow commented instructions regarding exception handling. Include exception handling in your test cases.

```
abstract public class UR_BST<Key extends Comparable<Key>, Value> {
    private UR_Node root;          // root of BST

    private class UR_Node {
        private Key key;         // sorted by key
        private Value val;       // associated data
        private UR_Node left, right; // left and right subtrees
        private int size;        // number of nodes in subtree
    }

    abstract public boolean isEmpty() ;

    abstract public int size() ;

    /**
     * @return {@code true} if this symbol table contains {@code key} and
     *         {@code false} otherwise
     * @throws IllegalArgumentException if {@code key} is {@code null}
     */
    abstract public boolean contains(Key key);

    /**  @throws IllegalArgumentException if {@code key} is {@code null} */
    abstract public Value get(Key key);

    /** @throws IllegalArgumentException if {@code key} is {@code null} */
    abstract public void put(Key key, Value val) ;


    /** @throws NoSuchElementException if the symbol table is empty */
    abstract public void deleteMin() ;

    /** @throws NoSuchElementException if the symbol table is empty */
    abstract public void deleteMax() ;
```

[1] This code is derrived from Segewick & Wayne 2002-2022

```
    /**  @throws IllegalArgumentException if {@code key} is {@code null} */
    abstract public void delete(Key key) ;

    abstract public Iterable<Key> keys();

    abstract public int height() ;

    /**
     * Returns the keys in the BST in level order (for debugging).
     * @return the keys in the BST in level order traversal
     * usually requires a supplemental Queue
     * include this in your test case
     */
    abstract public Iterable<Key> levelOrder() ;

}
```

Before you start

Note:

- Be sure to include a Unit Test program that demonstrates how your program works.
- You MUST demonstrate level by level printing in your test case.
- You MUST implement `Iterable<T>` interface for the Key set in the class you implement.
- Please write any assumption made. For example: describe what the return value means.
- Your methods must handle all the corner cases gracefully — for example, throwing exceptions with detailed explanations or returning values indicating the error in case the operation is not permitted. The comments should clearly state the issues and the remedies involved. In short, no illegal operation should be permitted and the list and all its parameters should be in a valid state.

Submission

Submit a single zip file `Lab6.zip` containing the class you implement, a unit test file containing a main method, a README file, and your class files. Your README should describe how to run your unit tests. Upload this file at the appropriate location on the Blackboard system at `learn.rochester.edu`.

1. All source code files should contain author and partner identification in the comments at the top of the file.
2. All source code should include links to any internet resources you used outside of course materials.
3. A plain text file named README that includes your contact information, your partner's name, a brief explanation of the lab (a one-paragraph synopsis. Include information identifying what class and lab number your files represent.), and one sentence explaining the contents of any other files you hand in. You can also share testcases those you ran.

Grading

Total: 100 pts

- 10 pts for README file, proper commenting, and error handling.
- (11 * 7 =) 77 pts for 12 methods shown in the sample code.
- 13 pts for your main method containing your unit testing class (main method, tests, etc)