Spring 2015, CMPE 160, Project 3

Cagatay Yildiz, Can Tunca, Haluk Bingol

In this project, you are going to implement a Binary Search Tree (BST).

Some Definitions:

A **tree** is an abstract data structure that is represented by a root and subtrees of this root.
A **binary tree** is a special type of tree in which each node in the tree can have at most 2 children, left and right children.
A **binary search tree** is a binary tree data structure where each node has a comparable key (and an associated value) and satisfies the restriction that the key in any node is larger than the keys in all nodes in that node's left sub-tree and smaller than the keys in all nodes in that node's right sub-tree (Definition from Wikipedia (http://en.wikipedia.org/wiki/Binary_search_tree)). Note that in this project, we do not have values in nodes but just keys. More specifically, each node stores only one integer and nothing else (except for pointers of children) so that you can focus on coding BST and are not bothered with details regarding to data(value) stored.

Representation of BSTs:

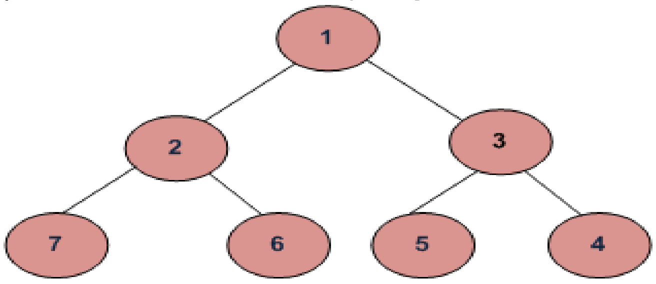
BSTs are represented by a root node, which stores an integer as data in this project, and left and right children. Because children are also nodes, they maintain the same structure. That is, child nodes are also trees, called subtrees conventionally. See that this is a recursive definition. Try to have an intuition of this definition because you are going to use recursion a lot in your implementation.

Details of The Given Code:

- `BadTestExample.java` : This is an example of a bad test design. I have added this to the project so that you check if your code works fine or not. As you implement the project, run the main method in this class and fix errors in your code if you have. Of course, there will be bunch of errors reported by this class when you get started but this is totally normal. Feel free to make changes in this class, it will not affect your grade in any way.
  Hint: Looking at this class, you should be able to understand how a binary search tree is defined, what types of operations can be done on it, what are the expected outputs of these operations, etc. Also, it is a good idea to draw tree with a pen and paper and run your algorithms on the paper first so that you do not think of abstract things all the time.
- `BSTInterface.java` : This interface contains signatures of all methods that you are expected to implement. See the javadoc for more detail.
- `BinarySearchTree.java:` This is the main file of the project, where you represent BST. It contains an inner class, which is already implemented, and implements `BSTInterface.java` interface. Initially, this class produces errors just because it does not implement methods that it promises to.
  - `private class Node` : The implementation of the node concept explained above. It is an inner class. If it is the first time you encounter an inner class, do not worry. Inner classes basically allow you to encapsulate data within a class. So, you can safely access all the fields of `Node` class within `BinarySearchTree.java` . See here (http://www.javaworld.com/article/2077411/core-java/inner-classes.html) for more details

regarding inner classes.
Important Remark: Do not remove anything from `Node` class but feel free to make (reasonable) additions to the class.

- The rest of the class will be methods to be implemented. Their definitions are I think clear.
If you have difficulty understanding how to traverse the tree, here is a good resource to read for in order traversal (http://www.math.ucla.edu/~wittman/10b.1.10w/Lectures/Lec18.pdf).
Also, I slightly modified spiral traversal algorithm. For the following tree, the order of visiting of your spiral traversal algorithm should be 1-2-3-7-6-5-4. That is, you visit all the nodes in a level consecutively, you first visit the root and get deeper levels, and you start traversal from the very left end of a particular level and go to the right. Make sure that you pass the spiral traversal test provided in `BadTestExample.java` .



- Make sure that your `insert` and `delete` methods work properly. I will use those while building your tree. So, for example, if they do not work well but `isEmpty()` method is okay, you are very likely not to receive any points from test cases for `isEmpty()` .

Details of Submission:

- The way your project is going to be graded is as follows: I will take `BinarySearchTree.java` that you implemented, put it into a separate project and run test cases on this project. So, all the code implemented in different .java files will be basically lost. Recall this while implementing. Note that this is not to say you are allowed to break OOP hierarchy, use meaningless methods, fields, inner classes, etc.
- Do not use any Turkish character in variable/field/method/class/package naming. This is a very bad practice. One of the most common naming conventions for Java programming states
  - Names of variables start with lower-case letters, and use upper-case letters at the start of each word in the name ("camel-style"): e.g., **dartsInCircle**.
  - Names of classes start with upper-case letters,and use use other upper-case letters at the start of each word in the name: e.g., **StringTokenizer**.
  - Names of public static final fields are written in all upper-case, and use underscores to separate each word in the name: e.g., **SPEED_OF_LIGHT**.
  - See here (https://www.cs.cmu.edu/~pattis/15-1XX/15-200/lectures/style/lecture.html) for more details regarding proper style of coding.

  Guys, you all know that "Agac yas iken egilir", so today is the right day to get a good naming attitude!
- For each method in `BSTInterface.java` , you are expected to devise at least one test case. As I did in the lab, you will use JUnit for testing purposes (Remember, we had a Runner class that contains a main method in which test cases are run, and another class that contains all the test cases). If you have difficulty remembering what we did in the lab, you may follow this (http://www.vogella.com/tutorials/JUnit/article.html) very nice tutorial. By the way, this guy prepares some really good tutorials, recommended to read in case you need.
- Project is due **May 20, 23.59 PM**. Submission via plugin.

Final Note: This is going to be a *dirty* project in the sense that you are likely to get `NullPointerException` very frequently. Even I got this error a number of times while implementing the project. In my case, many of the errors happened because I tried to reach uninitialized left/right child of a node. So, keep this in mind while coding:)

Good Luck!