Assignment 3

CSE 214

1. The binary tree node interface

Define a *generic* interface called `BinaryTreeNode`. It must consist of

1. `E getData();`
2. `void setData();`
3. `BinaryTreeNode<E> getParent();`
4. `void setParent();`
5. `BinaryTreeNode<E> getLeft();`
6. `BinaryTreeNode<E> getRight();`
7. `void setLeft(BinaryTreeNode<E> node);`
8. `void setRight(BinaryTreeNode<E> node);`
9. `void removeFromParent();` To remove this node and all its descendants from whatever larger tree this node is in.

Note: The visibility is not mentioned for these methods because Java interface methods are implicitly, and by default, public.

2. The binary tree node class

Implement a *generic* class called `BinaryTreeNodeImpl` that implements the above interface. This class should have nodes that directly link to left child, right child, and parent.

3. The binary tree interface

Define a *generic* interface called `BinaryTree` consisting of the methods

1. `void add(E element)`
2. `void remove(E element)`
3. `boolean contains(E element)`
4. `E min()`
5. `E max()`

4. The binary search tree class

Implement a *generic* class called `BinarySearchTree`. This class should have a private comparator variable. The implementation must have the following behavior:

1. A constructor that takes a comparator as its only argument, and constructs an empty tree.
2. `public void add(E element)`, to add a single element to the binary search tree. This method should implement the insert method studied in class. If the element already exists in the tree, do not create a duplicate node with the same value!
3. `public void remove(E element)`, to remove a single element from the binary search tree. This method should implement the remove method studied in class. It should do nothing if the element is not present in the tree.
4. `public boolean contains(E element)`, to return true/false depending whether or not the tree contains a node with this element. This method should take $\Theta(log_2 n)$ time on an average.
5. `public E min()`, to retrieve and return the minimum element in this tree in $\Theta(log_2 n)$ time.
6. `public E max()`, to retrieve and return the maximum element in this tree in $\Theta(log_2 n)$ time.

Notes:
1. Any other methods in the binary search tree class must not be public.
2. Keep in mind that any two elements of the generic type must have the ability to be compared against each other (i.e., the generic type must have a *total order*). We have learned in class how to do this.

5. The main method

Your main method (in the binary search tree class) for this assignment should ask the user to provide comma-separated strings in a single line. It should construct a binary search tree comprising of these strings. Then, it should ask the user to provide strings to search for (one per line).
If the string x is found, print "String x found in N steps", where N is the exact number of steps down the binary search tree (starting at the root) that were needed to reach x. Similarly, if the string is not in the tree, it should print "String x not found after N steps".
If the user provides "quit" as the string to search, then terminate the program. Otherwise, keep asking for more strings to search.
Note: Of course, given a list of strings, there are other ways of searching for a specific string among them. But that defeats the whole point of this assignment. **If your code is doing anything other than using your own binary search tree class to return the search results, you will receive 0 out of 25 points in section 4.** So, if your search algorithm is not working, but other things are fine, please just print out something like "search algorithm not working".

Points Distribution	**Total: 50 points**
1. Defining the binary tree node interface	5 points
2. Implementing the binary tree node class	10 points
3. Defining the binary tree interface	5 points
4. Implementing the binary search tree class	25 points
5. Main method	5 points

Submission Guidelines

1. Submit a single `.zip` file consisting of your `.java` files.
2. Remember to include the proper documentation in your comments in all files!
3. The `.zip` file that you submit should be named as `<SBUID>_<NETID>.cse214.hw3.zip`. For example, Mr. John Doe with student ID number 123456789 will submit the file: `123456789_jdoe.cse214.hw3.zip`.
4. Please make sure that your code compiles and can be run from the command line.
   Code that does not compile will not be graded.
5. The `public static void main(String[] args)` method is the *entry point* for a user. If you do not implement it, we cannot run and use your code.
   If this method is missing or does not compile, your code will not be graded.
6. Assignments will **not be regraded if you submit the wrong files**.
   So, **double/triple-check what you are submitting!**

Submission Deadline

The due date for this assignment is midnight of **Tuesday, Nov 3, 2015**.