COMPX201-24A & COMPX241-24A Assignment Two
Binary Search Trees

Due: Friday 19th of April 2024 11:59pm.

This assignment is intended to give you experience building, maintaining, and using a binary search tree (BST). You must write your own BST class and all its required operations. Part Two involves updating your BST to work as a Computer Science dictionary.

Part One (50%):

You must write your own BST class and all its required operations. Your program should follow the specifications given below.

1. **BST:** define a class called `StrBST` in a file called `StrBST.java`. This class is to implement an unbalanced BST using self-referential nodes. Your solution should support the following public methods using **recursion** where applicable:
   - `insert(String s)` - add the value to the BST, maintain ordering alphabetically by using the `compareTo` method (see https://www.w3schools.com/java/ref_string_compareto.asp for more details) and assume that duplicates are not allowed.
   - `remove(String s)` - remove the specified value from the BST.
   - `search(String s)` - search the tree to find the specified value and return a boolean true if value is found, false otherwise.
   - `print()` - a method which prints out the tree following an in-order traversal with each value on a separate line. For example:

     ```
     Root: A | Left: null | Right: null
     Root: B | Left: A | Right: null
     Root: C | Left: B | Right: E
     Root: D | Left: null | Right: null
     Root: E | Left: D | Right: null
     ```
2. **The Node:** define a class called `Node` for the nodes in your `StrBST`. It can be either an external class in a separate file called `Node.java` or an inner class of `StrBST`. It should have the following:
   - A public member variable to hold the value of node as a string.
   - A public member variable to hold a link to the left subtree.
   - A public member variable to hold a link to the right subtree.
   - A constructor that takes a value as a string argument and copies that value into the node's member variable. The constructor should also initialize the left and right subtree links to null.
3. **Debugging:** Write a program class that creates one or more of your `StrBST` objects and tests that all your methods work as per the specification. Example output from one such file will be provided to you. Your test program will not be marked, but is for your own solution development process.

Part Two:

Make a copy of your `StrBST.java` and `Node.java` files and rename them to `DictionaryBST.java` and `DictionaryNode.java`. Update these files as follows.

- Add a public member variable to your `DictionaryNode` class that holds the definition of a word/phrase as a string.
- Update your `DictionaryBST` class so that a String value and String definition are added when a Node is inserted into the BST.
- Add a `printDictionaryItem(String s)` method to your BST that prints the value and definition of a given word/phrase (String s).
- Add a `printDictionary()` method to your BST that prints the value and definition of each node in alphabetical order (printing only the current node, and not the left and right subtrees). For example:

```
ASCII
Pronounced ass-key. Stands for the American Standard Code for
Information Interchange. It is a 7bit code for characters. For
example: the letter 'x' is represented as 1111000. Because it is
only 7 bits long, an ASCII character can be stored in a byte.

Abstraction
Removing detail of something to focus on a subset of the
features without confusing details.

Address
A location in memory or identifying a particular piece of
memory. b) An identification that allows access to something
such as an internet address which allows one to access a device
on the internet.
```

Define a new class called `DictionaryLookup`. This class should have a `main` method and a collection of supporting methods. Your `DictionaryLookup` class should read in a dictionary file, process it, and store each item from the file as a `DictionaryNode` in your `DictionaryBST`. The dictionary file and an example output file will be provided to you.

Your `DictionaryLookup` class should then allow the user to enter instructions via the command line to do the following:

1. Search for a word/phrase in the dictionary
2. Print a given word/phrase and it's definition
3. Add a word/phrase and definition to the dictionary
4. Remove a word/phrase and definition from the dictionary
5. Print all of the words/phrases and their definitions, in alphabetical order

Assessment:
Completing Part One can earn up to a B grade, but to be eligible for an A+ you must also implement Part Two. Your solution will be marked on the basis of how well it satisfies the specification, how well you have approached the problem (Solution Quality), how well-formatted and easy to read your code is (Code Quality), and whether each class and public method has at least some comment explaining what it does, what it's for, and what any of its arguments are (i.e. documentation). Your code should compile and run as a console program from the command-line (i.e. no GUI or IDE).

Submission:
Create an empty directory (i.e. folder) using your student ID number as the directory name. Place copies of your **source code (.java files) in this directory**. If you wish to communicate with the marker any additional information then you may include a plain text README file, **but nothing else (e.g. no compiled code (.class files) or IDE folders or files)**. Compress and upload this directory through the Moodle submission page for this assignment.