Project 2 – Binary Search Tree

Overview

This project is to create a binary search tree to store and search key/value pairs (what the book calls a "symbol table"). Imagine we have an index of a book (a mapping of a word to the page number it first appears on) and we want to use this to build a search program that, given a word, will tell us which page the word appears on.

Requirements

Your main method should create a binary search tree object. It must then read a list of words and page numbers from a text file (three files are provided) and add all of these words to the binary search tree. The word will be the "key" and the page number will be the "value." After adding all the words, it should display all three tree traversals. Finally, it should enter a loop where the user may input words to search for. If the entered word is found in the tree, it should output the page number of that word. Repeat this until some special terminating word is entered.

Sample Output

Your output doesn't have to match this perfectly, this is what my program looks like using "10words.txt"

```
Please input the filename of the input file: 10words.txt

Pre-order traversal: market:151 brown:410 accurate:292 imported:424 collect:146 incandescent:83
wren:286 oafish:289 mug:540 nondescript:139
In-order traversal: accurate:292 brown:410 collect:146 imported:424 incandescent:83 market:151 mug:540
nondescript:139 oafish:289 wren:286
Post-order traversal: accurate:292 collect:146 incandescent:83 imported:424 brown:410 nondescript:139
mug:540 oafish:289 wren:286 market:151

Please input a word: imported
'imported' was found on page 424!
Please input another word (CTRL+D or "EXIT" to exit): market
'market' was found on page 151!
Please input another word (CTRL+D or "EXIT" to exit): pizza
'pizza' was NOT found!
```

BinarySearchTree Class Requirements

Your binary search tree implementation must be in its own class named `BinarySearchTree`. Your search tree must hold strings as keys and integers as values. It must contain the following public methods. You may add as many private methods as you need.

`BinarySearchTree()` – Default constructor
`addValue(string key, int value)` – Adds a new key/value pair to the tree.
`search(string searchKey)` – Search in the tree. Returns the integer value associated with the key if found, -1 if not found.
`preOrderTraversal()` – Prints the contents of the tree using pre order traversal (root-left-right).
`inOrderTraversal()` – Prints the contents of the tree using in order traversal (left-root-right).
`postOrderTraversal()` – Prints the contents of the tree using post order traversal (left-right-root).

Programming Languages

You may use any programming language as long as you confirm it with me first. This course will be centered on Java, so that is my recommendation.

Dataset

Three dataset files are provided. Each line of the files will have a word (key), then a tab, then a page number (value). There is no guarantee that these lists won't contain duplicates, so make sure your tree can handle duplicate words (just overwrite the existing value). I recommend you look at the dataset files to get a better idea of what they look like.

Hints

- Before you even start working on your binary search tree, write code to read in the input file. Once you get that working then move on to the binary search tree.
- You should make a separate class, Node, which will hold the key, value, and links to a left and right a child node. This will be similar to a node used in a linked list.
- Your `search()`, `addValue()`, and three traversals methods can be done pretty easily using private recursive methods.
- Start off using the smallest dataset and manually draw out the tree. This way you know what the tree should look like and can verify that your program is correctly assembling the tree.
- If you are uncertain about how binary search trees work, review section 3.2 in the textbook.
- I highly recommend using an IDE you're comfortable with using the debugger.
- Start early and ask questions when you get stuck. It's normal to get stuck, don't feel bad.
- Seriously. Ask me for help if you need it. Don't feel ashamed or embarrassed.

Grading

The assignment is worth 100 points. If your code doesn't compile, there is an immediate 30-point penalty. You will be graded on the following criteria:

Header comment block with name, class, date, project	10
Adding key/value pairs to the tree works as it should	20
Searching for a key in the tree works as it should	20
Tree traversal methods work as they should	20
Words and page numbers are correctly read from the text file	10
User is able to input words and search results are displayed	10
Overall organization and quality of code	10
Total	**100**

Submitting your work

Your work should be submitted as an archive (.zip, .7z, .rar) on Moodle. This archive should include all of your source code files. Please do not submit your entire project folder.

This project is due Sunday, March 10th at 11:59pm.