2217 Java Software Dev 2
Project 8 – Enhanced Binary Search Tree

Purpose

Implement behaviors that are appropriate for a binary search tree.

Goal

Write a program that implements new behaviors for a binary search tree.

Project Overview

According to Copilot,

> *"Understanding how to write code for binary search trees (BSTs) is a foundational skill in computer science, as it sharpens students' grasp of algorithms, recursion, and data structure design. BSTs provide an efficient way to store, search, and manage sorted data, and they're widely used in applications like databases, file systems, and syntax trees. Writing methods for a BST—such as insertion, deletion, traversal, and search—requires students to think critically about structure, edge cases, and performance. It also reinforces key object-oriented principles like encapsulation and modular design. By learning to code BSTs, students build the analytical mindset needed to tackle more complex data structures and real-world programming challenges."*

With that thought in mind, we are going to add some new behaviors to the BST class presented in the book.

Program Requirements

Start the project by copying the `BinarySearchTree.java` code from exercise zyDE 17.4.1. Create a class `EnhancedBST` that extends `BinarySearchTree`. All the new behaviors will be added to the class `EnhancedBST`. Submissions that do not meet these criteria will not be accepted.

One problem with extending class `BinarySearchTree` is that the `root` instance variable is `private`, and we will need to access the root in class `EnhancedBST`. Think about the best way to solve this problem. (Hint: Do not change the access specifier of `root` to `public`.) This is the only change you should make to the `BinarySearchTree` class.

Create a `UserInterface` class that has a public method named `go`. The only action needed in the `main` method is to instantiate a `UserInterface` object and call the `go` method.

The `UserInterface` class should control all the input/output. Display a menu with the following options:

```
Welcome to Enhanced BST Tester.


Here's the menu of choices -
  0) Quit
  1) Build a BST from a text file
  2) Print the tree
  3) Add data
  4) Remove data
  5) Show tree height
  6) Show internal path length
  7) Count absent children
  8) Find a path sum
  9) Export a BST to a text file
Enter your choice:
```

Option 1:

Prompt the user for a text file name that contains integers. Read the numbers from the file and add them to the tree.

Option 2:

Override the inherited `print` method. The enhanced version should use recursive methods to display the tree data using three common tree traversals: in-order, preorder, and post-order. *AI Tip:* Research how these traversals work if it is still confusing to you. Given a data file that looks like this:

```
50
30
10
20
15
80
60
70
65
100
```

The output should look like this:

```
  Inorder: 10 15 20 30 50 60 65 70 80 100
 Preorder: 50 30 10 20 15 80 60 70 65 100
Postorder: 15 20 10 30 65 70 60 100 80 50
```

It is recommended that you sketch the shape of this tree for reference, as it will be used in the description of several of the other menu options.

Option 3:

Prompt the user for a number. Then add the number to the tree.

Option 4:

Prompt the user for a number. Then remove the number from the tree.

Option 5:

Write a method for the enhanced BST that uses a recursive helper method to calculate the height of the tree. Use the author's definition of tree height. The height of the tree shown in option 2 above is 5.

Option 6:

Write a method for the enhanced BST that uses a recursive helper method to calculate the internal path length of the tree. The internal path length is the sum of the level numbers of the nodes in the tree. The root is on level 0. The internal path length of the tree shown in option 2 above is 22. *AI Tip:* Look for a visual that illustrates internal path length.

Option 7:

Write a method for the enhanced BST that uses a recursive helper method to calculate the number of absent children in the tree. Absent children are defined as every place in the tree where a child node could be attached. The number of absent children in the tree shown in option 2 above is 11.

Option 8:

Write a method for the enhanced BST that uses a recursive helper method to determine whether a path sum in the tree is equal to a value input by the user. The path sum is the sum of the data in the nodes in a path from the root to a leaf node. Using the tree shown in option 2 above as an example, 125 is a path sum, but 110 is not.

Option 9:

Prompt the user for a text file name. Traverse the tree and output the tree data in the correct order, such that the tree can be rebuilt exactly as it is by using option 1. To accomplish this task, create a method in the enhanced BST class that returns an Iterator. Successive calls of the `next` method of the iterator should return the nodes in the proper order.

AI Tip: Research how to read from and write to a text file, if you need a refresher.

Note that your program should be resilient. That means it should handle common exceptions appropriately.

Expectations

The program is expected to compile successfully, include identification comments at the top, and produce output that exemplifies good grammar and spelling. Failure to meet these standards may result in point deductions.