Binary Search Tree Dictionary Lab

CS 145 – Lab 6

Note before you continue:

You will get extra credit for a successful implementation of a binary search tree with nodes that are alphabetically ordered and that stay alphabetically ordered as more are added. The instructions below make reference to a binary search tree, but that is not required for full credit. Let me know if you have any questions about this.

If you submit a binary search tree, be sure to mention it either in your header comment in all caps or in a Canvas comment with your submission.

Key topics:

Printing, data types, methods, operators, expressions, variables, loops, recursion, parameters, returns, String objects, Scanner objects, and binary search trees

Learning Outcomes:

- Continue to become familiar with setup, design, execution and testing of Java programs
- Design and develop a multi-class, multi-method program in good style
- Demonstrate your understanding of recursion
- Apply tools and techniques introduced in class to create a working program
- Demonstrate your understanding of binary search trees

Task:

Your task is to create a dictionary using a binary search tree to do an employee, customer, or member lookup. You may use either recursion or iteration to implement this data structure. Your BST should give the user the option to use either pre-order, in-order, and post-order searches. You will name the file Dictionary.java. You will be including this in your GitHub portfolio that you will show off to employers, so make it good.

The database will start out empty, and each of the nodes will include first name, last name, street address, city, state, zip, e-mail, and phone number, along with a primary key for doing search, add, modify, and deletes.

Functional Rundown:

Upon executing your program, the console should display a menu (add, delete, modify, lookup, list number of records. You don't need to worry about sorting or balancing the

tree. Based on the user's prompt, it should ask the user for the appropriate information and carry out the assigned tasks.

For Adds, it should prompt the user for the appropriate primary key, along with user data. The program should give the user confirmation about the data added.

For Deletes, it should prompt the user for field to be deleted and confirm that the user wishes to delete this record. It should also confirm that the user wishes to delete the record.

For Lookups, you should give the user option of pre-order, in-order, or post order, and print out the appropriate values.

For Modify, the user should be prompted on which field(s) should be modified along with confirmation.

Implementation Details:

You are free to choose the names of the classes, methods, variables, and constants. Make sure that they follow the standard naming conventions covered in class. Be sure to include two to three classes, along with multiple methods for each class.

I strongly encourage you to tackle this assignment in parts and practice iterative design. Work on a few things and then test them to make sure they work. Then, add some more and test those. Don't forget to turn in your pre-lab and post-lab along with this assignment. If you worked together on this as a team, make sure that you include the names of all the team members on the program in the comment header.

Breaking Down the Assignment:

This is a large project; I highly recommend breaking it down into pieces! Tackling this assignment method by method is a good strategy.

Style Guidelines and Grading:

Follow good general Java style guidelines such as: appropriately using control structures like loops and if/else statements; avoiding redundancy using techniques such as methods, loops, and factoring common code out of if/else statements; properly using indentation, good variable names, and types; and not having any lines of code longer than 100 characters in length.

Functionality: 50%; half of your grade comes from the output of the program; make sure there is communication with the user of your program; if your program has no output, I will dock 50% of your points; you must include a main() method.

Style: 25%; every .java file you submit must have a header comment with at least the following four elements: your name, the date, the project (e.g., CS 145 Lab 2), and a purpose statement (you may also include your instructor's name if it is helpful for you to keep track, but it is not required); every file needs to have all four of these elements; if some are missing, I will dock points from this portion of your grade.

Readability: 25%; follow standard practices for indentation, naming (camelCase for methods and variables, PascalCase for classes, etc.), and line length; keep your lines under 80 characters long or at most 100 characters; if you include a line, even if it is a comment, that has 101 or more characters, I will dock points.