Alexander Hernandez

CSCI312

David Krogmann

9/16/2024

Overview

1) This program demonstrates the use of a binary search tree. The program uses multiple functions including adding an element, finding an element, functions that put the tree in a pre, post, and in order, and a size function to give the size of the tree. The program would display simple demonstrations of the working functions.

⦁ PROCESSING LOGIC

1) The program shows three different classes, one class creating a node called the Node class, another class called BTImp which is the class holding all the main functions, and BST which is the main class displaying the output. Node is a short class creating nodes in a tree including a left node, right node, and the data that goes in it. The Node class also incorporates a generic value “T” to take in any data type specified in the main program. BTImp demonstrates each main function of the program. It first creates the “root” value and sets it to null. Each function has a private and public function. The private function takes in some cases takes in values and uses those values in the function to make it work as intended, the public function simply calls the private version of itself to provide an easier way to call the function in the main program. BTImp also uses the generic value “T”.

⦁ DATA (INPUT/OUTPUT)

1) Each main private function takes in the generic type “T” node called myRoot. Other functions take in some other values including getSize taking in an int tSize to return the size of the tree. FindElement and addElement both take in a generic type “T” data including their public functions so that when they are called, you must put in values to add that data or to find that data. BST, the main program, creates a new tree called “tree” and then uses the “String” type. Then it adds the letters A-G into the tree.

⦁ COMPONENTS (SOURCE CODE NAMES, CLASSES, METHODS)

|  |
| --- |
| BTImp |
| -root: Node<T> |
| +BTImp() |
| +addElement(<T>: <T>) |
| -addElement( <T>: <T>, Node<T>) |
| +findElement(<T>: <T>) |
| -findElement(<T>: <T>, Node<T>) |
| +getSize(): int |
| -getSize(int: int, Node<T>) |
| +postOrder() |
| -postOrder(Node<T>) |
| +preOrder() |
| -preOrder(Node<T>) |
| +inOrder() |
| -inOrder(Node<T>) |

⦁ TESTING

1) The expected results are what's displayed in the main program using the main functions. For example, the getSize function should return the size of the tree. In this case, the letters A-G was added to the tree so the value 7 should return to indicate 7 values in the tree. In order, post order, and pre order all should display their respective orders. The find function should display the element if it was in the tree otherwise, it should display null. In this case, I tried to find the letters “F” and “S”. The letter “F” was returned but “S” was returned as null. I also testing using an Integer as the tree’s value and put in several numbers into the tree to test the generic value.