Assignment 3 – Binary Trees – Documentation (Max Fyall)

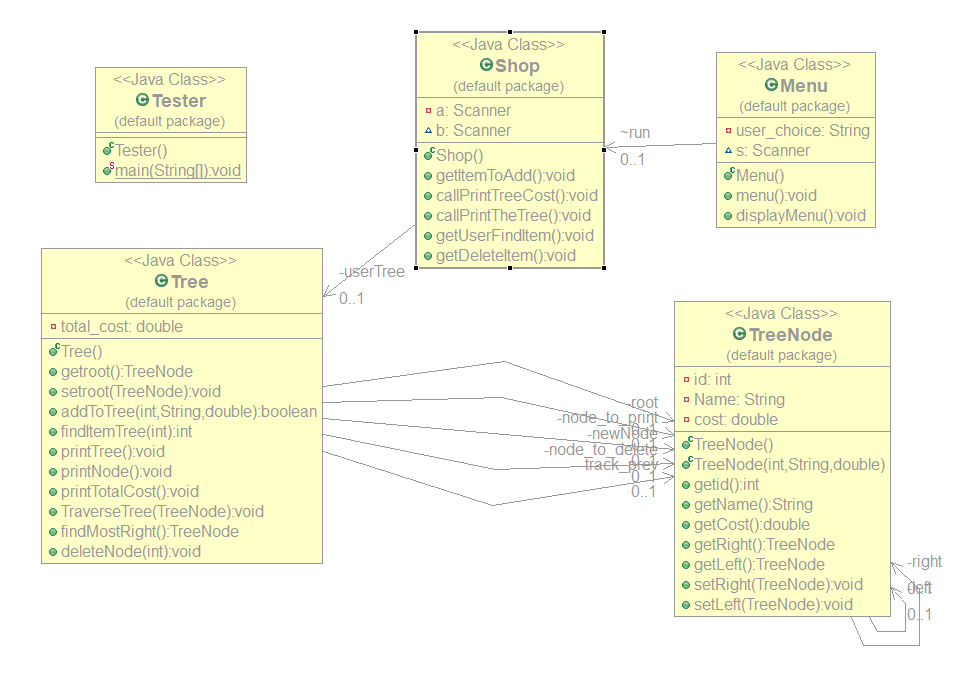
Introduction to the problem:

The aim of this assignment is to program a binary search tree data structure to store shop item information, without using the java collection classes (Implemented through first principles).

Requirements:

The mandatory requirements were completed. The optional requirements were not completed due time constraints and the level of difficultly of the requirements. The implementation of the 5 mandatory requirements was successful. It must be noted that some optional requirements were attempted but were deleted from the final code because they were incomplete and may have caused problems in the program.

Class Design:



Pseudo Code:

getItemToAdd()

* Create Boolean found
* Get item id from keyboard
* Get item name from keyboard
* Get item cost from keyboard
* Found = addToTree(itemID, itemName, itemCost)
* If found is true
  + Item is already in tree
* Else
  + Confirm item was added

getUserFindItem()

* Get the id of the item to find from the keyboard
* Data = findItemTree(findID)
* If findID id not equal to data
  + Then item is not in the list
* Else
  + Item found
  + Print item details

getDeleteItem()

* Get id to delete from the user
* Call the deleteNode method

addToTree()

* Initialise variables
* Create new Node and pass in values from getItemToAdd method
* Add value of cost to global variable total cost to store the total cost of the tree
* If root is null
  + Set new node to root
  + Set found = false
  + Return found
* Else
  + If id of new node is already in tree
    - Set found to true
    - Return found
  + Else
    - Start loop, while current is not null
    - Set previous to current
    - If id of new node is less than the id of current node
      * Set current to left of current
    - Else
      * Set current to right of current
    - End loop
    - If id of new node is less than id of previous
      * Set left of previous to newNode
    - Else
      * Set right of previous to newNode
    - Return found = false

findItemTree()

* Initialise variables
* Set findNode to root
* Start loop, while findNode is not null
* Set data to id of findNode
* If data = id of newNode or data = findID (passed in from getFindItem method)
  + Set node\_to\_print to findNode
  + Set node\_to\_delete to findNode
  + Return data
* Else
  + If findID = 0 (method not called from getFindItem)
    - If id of newNode is less than data
      * Set findNode to left of findNode
    - Else
      * Set findNode to right of findNode
  + Else
    - If findID is less than data
      * Set track\_prev to findNode
      * Set findNode to left of findNode
    - Else
      * Set track\_prev to findNode
      * Set findNode to left of findNode
* Return 0

TraverseTree()

* if current is not null
  + if left of current is not null
    - Call TraverseTree(pass in left of current)
  + Print the details of the node
  + If right of current is not null
    - Call TraverseTree(pass in right of current)

findMostRight()

* Initialise variables
* Set find to left of node\_to\_delete
* Start loop, while find is not null
  + Set second previous to previous
  + Set previous to find
  + Set find to right of find
* End loop
* Set find to previous
* Set previous to second previous
* Return find

deleteNode()

* Initialise variables
* Set info to findItemTree()
* Set previous to track\_prev
* Set current to node\_to\_delete
* If info is not equal to deleteID
  + Output Error message
* Else
  + If left of current is null and right of current is null
    - If previous is null
      * Set root to null
    - Else
      * If left of previous = current
        + Set left of previous to null
      * Else
        + Set right of previous to null
    - Print deleted node details
  + If right of current is null and left of current is not null
    - If left of previous = currents
      * Set left of previous to left of current
    - Else
      * Set right of previous to left of current
    - Print deleted node details
  + If right of current is not null and left of current is null
    - If left of previous = currents
      * Set left of previous to right of current
    - Else
      * Set right of previous to right of current
    - Print deleted node details
  + If right and left of current are not null
    - Set previousToDelete to track\_prev
    - Set replace to findMostRight()
    - If left of replace is null and right of replace is null
      * If left of previous = current
        + Set left of previous to null
      * Else
        + Set right of previous to null
    - If right of replace is null and left of replace is not null
      * If left of previous = replace
        + Set left of previous to left of replace
      * Else
        + Set right of previous to left of replace
    - If right of replace is not null and left of replace is null
      * If left of previous = replace
        + Set left of previous to right of replace
      * Else
        + Set right of previous to right of replace
    - Print deleted node details
    - Set left of replace to left of current
    - Set right of replace to right of current
    - If previousToDelete is null
      * Set root to replace
    - Else
      * Set left of previousToDelete to replace.

Test Plan:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Test Data** | **Expected result** | **Worked?** |
| Add an item to empty tree | Id = 5, name = sweets, cost = 1 | Item is added to the tree as root | y |
| Add item onto tree with root as 10 | Id = 4, name = chocolate, cost = 1 | Item is added to the left side of the tree | y |
| Add item onto tree with root as 10 | Id 15, name = coke, cost = 0.60 | Item is added to the right side of the tree | y |
| Print the whole tree in numerical order of id | Three items with ids:  2,6,10 | Items are printed out in numerical order 2,6,10 with the details of the items alongside them | y |
| Print entire tree again | Three items with ids:  10,15,20 | Items are printed in numerical order | y |
| Add item to tree already in | Set root to id 5 and try add item with same id | Error message should appear, and user should be returned to menu | y |
| Add item already in tree | Set root to id 6 and add a node with id 8, try and add 8 again | Error message should appear, and user should be returned to the menu. | y |
| Find item in empty tree | Try and find an item with id 5 in a empty tree | Error message appears, and menu re-appears | y |
| Find item in tree | Set root as item with id 5 and then try find the id 5 | Message appears that says the item has been found and the details of the item are displayed | y |
| Item not found | Set root to 10 and try find item with id 6 | Error message appears, and menu re-appears | y |
| Calculate the total cost of the tree | Input items with cost, 0.50, 3.00, 10.00 and 0.25 | Calculation should output 13.75 as total cost | y |
| Delete an item from the tree | Tree with root id 5 and two connected nodes (2 and 10), enter 2 to delete | Program should delete the node with id 2 | y |
| Delete an item from the tree | Tree with root id 5 and two connected nodes (2 and 10), enter 10 to delete | Program should delete the node with id 101 | y |
| Delete root from tree | Tree with root id 5, 2 connected nodes (3, and 10) and node with id 4 connected to 3. Delete id 5 | Program should delete the root and replace it with the right most on the left sub tree i.e. 4 | y |
| Delete from an empty tree | Enter id 5 to be deleted from empty tree | Error message should display and menu should re-appear upon hitting enter | y |

Evaluation:

I found this assignment to be one of the more challenging assignments I have done. The complexity of the code was very confusing at times and slowed down my progress significantly. However, this can be a positive because I now know how to code a basic binary tree from first principles which may come in handy for the duration of my programming life. Most of the problems I encountered where fixed with ease. Most of these errors were null pointer exceptions and some stack overflows as well. I make good use of the debugger in eclipse to help me find out what was causing my errors. Due to my slow progress during this assignment, I was only able to complete the mandatory requirements. If I had put more time into implementing the assignment, I believe I would have achieved more than what I have.