Name: Jamie Fergus

Matric number: 1900018054

Lab Title: AC12001 assignment 3: Binary Trees

# Introduction

The aim of this lab is to learn how to design and code a program that uses a binary search tree data structure without using the Java collection classes, i.e. you must code your own binary search tree from first principles and not use a pre-built library tree class. You will also gain further experience of using recursion. For this task you will store shop item information as nodes in a binary tree.

# Requirements

1. Allow the user to enter details for a shop item.

2. Allow the user to request that the whole tree be printed in numerical order of Item ID.

3. Allow the user to enter an Item ID to look for in the tree and show the name and cost for that shop item.

4. Calculate the total cost of all the items in your Tree.

5. Allow the user to request that a particular item is removed from the.

6. [Optional] Write your tree to a text file, and read it in again at the start of the program.

7. [Optional] Implement other forms of traversal for the tree.

8. [Optional] Extend your program to keep track of stock levels of all Items by adding a field in the TreeNode to count the number of Items in stock.

9. [Optional] Read up on how to balance a binary tree and implement a practical solution to

improve the balance of your binary tree.

I tackled every requirement except from the last optional requirement. I was successfully able to complete the requirements I attempted.

# Design

## Class Diagram

TreeNode

Tree

Menu

itemID  
name  
cost  
stock  
left  
right  
parent

TreeNode root

Tree shop

TreeNode()  
increaseStock()  
decreaseStock()  
(get and set methods)

Main()

runMenu()

displayMessage()

displayMenu

checkInt()

Tree()  
getRoot()  
setRoot()  
isTreeEmpty()  
addToTree()  
printTree()  
findInTree()  
calculateTotal()  
deleteFromTree()  
writeToFile()  
ReadFromFile

## Pseudocode

addToTree()

if current is null, add node to tree  
else if itemID < current itemID, traverse left  
else if itemID > current itemID, traverse right  
else, increase stock

deleteFromTree()

if item not found prompt user

else if itemID < current itemID, traverse left  
else if itemID > current itemID, traverse right  
else item found

if stock > 1 decrease stock  
 if leaf node, delete leaf  
 if current node with single child, delete node and link child to parent node  
 if current node has two childs, find right most node in left tree, replace deleted node, reconnect children.

Return

# Test Plan & Results

Test number/date/version: 01/03/20

Test Notes: Tests run on submitted assignment

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Description** | **Test Data** | **Expected result** | **Worked?** |
| Test adding nodes to tree | Items with ids  3,2,1,7,6,9 | Create binary tree sorted with itemIDs | Y |
| Find item | 7 | Find and display items 7 data | Y |
| Find item, (not in tree) | 16 | Display item not found | Y |
| Print items inorder |  | 1,2,3,6,7,9 with data | Y |
| Print items preorder |  | 3,2,1,7,6,9 with data | Y |
| Print items postorder |  | 1,2,6,9,7,3 with data | Y |
| Calculate total |  | Correct total cost of items | Y |
| Add to stock | 1,1,6 | Correct increase of stock | Y |
| Calculate total stock |  | Correct total cost of stock | Y |
| Delete leaf node | 9 | Remove leaf | Y |
| Delete node with 1 child | 7 | Remove node and reconnect child | Y |
| Delete node with 2 childlren | 3 | Delete node without breaking the tree structure with all childs connected | Y |
| Decrease stock | 1 | Correct decrease of stock | Y |
| Write to file | A tree | Correctly written contents of tree written to a file | Y |
| Read from file | A tree | Correctly read all node data from file, and keeping original tree structure | Y |

# Self-Evaluation

I think I did well this assignment. I was able to complete most requirement without to many difficulties. The hardest part was probably getting the deleteFromTree() method working correctly.