# Assignment 3: Documentation

Alfie Hippisley – Assignment 3 – 03/03/2018

Introduction :

The problem of this week’s assignment is to create a program that will use the ‘Binary Tree’ data structure to store data about multiple students. For this assignment Java library classes cannot be used, and all coding must be done from first principles.

Requirements :

1 – The user of the program must be able to add a student and their corresponding data into the binary tree data structure to be stored.

2 – The user must be able to request a complete print out of the binary tree data in numerical order of the student id.

3 – The user must be able to enter a student id to be searched for in the tree. Once found the students exam mark should be printed. If the student is not found then the user must be informed.

4 – The user must be able to enter a student id so that the corresponding student can be removed from the binary tree. If the student is not found then the user must be informed.

5 – (Optional) The program must be able to write the binary tree to a external text file. When the program then runs again it must be able to read in the saved data.

6 – (Optional) The program is able to perform different types of traversal with the binary tree such as preorder and postorder traversal.

7 – (Optional) The program can balance the binary tree.

Response to the requirements:

1 – This requirement has been fully implemented.

2 – This requirement has been fully implemented.

3 – This requirement has been fully implemented.

4 – This requirement has not been implemented.

5 – This requirement has not been implemented.

6 – This requirement has not been implemented.

(Next Page = Class Design)

Test Plan :

|  |  |  |
| --- | --- | --- |
| Test | Expected Result | Actual Result |
| Print An Empty Tree | Displays Message Saying Tree Is Empty | Displays Message Saying Tree Is Empty |
| Print A Tree With 8 Nodes Of Data | Correctly Displays All Nodes In Ascending Order | Correctly Displays All Nodes In Ascending Order |
| Print Tree After Node Added | Displays New Addition | Displays New Addition |
| Print Tree After Node Removed | Displays Tree With Node Removed | Displays Tree With Node Removed |
| Add Node To Empty Tree  {4343,56%,John Smith} | Node Added Correctly | Node Added Correctly |
| Add Node To Tree With Existing Nodes  {4890,63%,Amy Black} | Node Added Correctly In Order | Node Added Correctly In Order |
| Add Node With Invalid Data {A,A,A} | Error Message Saying Why Its Invalid. Returned To Menu | Error Message Saying Why Its Invalid. Returned To Menu |
| Add Node With Out Of Range Data {1,1,A} | Error Message Saying The Data Is Out Of Range And To Try Again. | Error Message Saying The Data Is Out Of Range And To Try Again. |
| Attempt To Find A Node In A Tree With No Nodes | Message Saying Nothing Was Found | Message Saying Nothing Was Found |
| Attempt To Find A Node In A Tree With Nodes | Message Saying Node Was Found Then Shows Students Data | Message Saying Node Was Found Then Shows Students Data |
| Attempt To Enter Out Of Range Student ID | Error Message Saying The Data Is Out Of Range And To Try Again. | Error Message Saying The Data Is Out Of Range And To Try Again. |
| Attempt To Enter Invalid Data Type For Student ID | Error Message Saying The Data Is Invalid And Returns User To Menu | Error Message Saying The Data Is Invalid And Returns User To Menu |
|  |  |  |
|  |  |  |
|  |  |  |

Required actions from testing:

None, this testing went very well. Think I tested most situations using min, max and extreme data to do so.

Assignment Evaluation :

I think this assignment went quite well compared to my last assignment. I experienced little to no problems with anything I tried to implement. The multiple players made me think a little bit as I missed that requirement when I initially designed my program, but I was easily able to adapt the design I had made to accommodate this change.

When I first started to code my assignment I started first with the MySet class. I found that this was the best way to start as I could first focus on getting the sets working then the lottery functions. This worked well as I experienced no logic errors when implementing the lottery part of the program. Resulting in far less time debugging. It also happened to be an optional extra. I found that programming this assignment and using the lecture slides for reference was a great way to develop a very good understanding of library classes and interfaces. I also found working with library classes to be enjoyable and hassle free.

The most challenging part of this assignment was implementing the settings menu to configure the lottery the way the user wanted it. I did not want to ask the user at the start of each lottery run as this would be irritating if you are doing repeated draws. With the same settings. I think the final settings menu works quite well.

Overall, I was quite happy with how this assignment went, however I did forget to add exception errors for ‘userInput’ because if a letter is entered the program will crash. I was unable to implement this as I remembered too late.

Code References :

All lectures slide’s from myDundee and the assignment brief were used to form the basic foundation of my code. Some code was also taken from previous submissions