**Project #3**

Name - Herman Mann

Course - CMSC 350

Date - 02/21/2022

Table of Contents

[**UML Diagram** 3](#_Toc65023737)

[**Test Plan** 4](#_Toc65023738)

[**Test Case # 1** 4](#_Toc65023739)

[**Test Case # 2** 5](#_Toc65023740)

[**Test Case # 3** 6](#_Toc65023741)

[**Test Case # 4** 7](#_Toc65023742)

[**Test Case # 5** 8](#_Toc65023743)

[**Test Case # 6** 9](#_Toc65023744)

[**Test Case # 7** 10](#_Toc65023745)

[**Test Case # 8** 11](#_Toc65023746)

[**Test Case # 9** 12](#_Toc65023747)

[**Test Case # 10** 13](#_Toc65023748)

[**Test Case # 11** 14](#_Toc65023749)

[**Test Case # 12** 15](#_Toc65023750)

[**Lesson Learned** 16](#_Toc65023751)

# **UML Diagram**

Diagram

Description automatically generated

# **Test Plan**

|  |  |
| --- | --- |
| **Test Case # 1** | |
| **Description** | Binary tree is balanced |
| **Input** | (A(G(j)(1))(z(5))) |
| **Expected Output** | Print true in the output display text box |
| **Actual Output** | True is printed in output display text box |
| **Pass** | **Pass** |
| **Screenshots** |  |

|  |  |
| --- | --- |
| **Test Case # 2** | |
| **Description** | Binary tree is not balanced |
| **Input** | (A(G(j)(1))) |
| **Expected Output** | Print false in the output display text box |
| **Actual Output** | False is printed in output display text box |
| **Pass** | **Pass** |
| **Screenshots** |  |

|  |  |
| --- | --- |
| **Test Case # 3** | |
| **Description** | Binary tree is not full |
| **Input** | (A(G(j)(1))(z(5))) |
| **Expected Output** | Print false in the output display text box |
| **Actual Output** | False is printed in output display text box |
| **Pass** | **Pass** |
| **Screenshots** |  |

|  |  |
| --- | --- |
| **Test Case # 4** | |
| **Description** | Binary tree is full |
| **Input** | (A(G(j)(1))(z(5)(2))) |
| **Expected Output** | Print true in the output display text box |
| **Actual Output** | True is printed in output display text box |
| **Pass** | **Pass** |
| **Screenshots** |  |

|  |  |
| --- | --- |
| **Test Case # 5** | |
| **Description** | Binary tree is proper |
| **Input** | (A(G(j)(1))(z(5)(2))) |
| **Expected Output** | Print true in the output display text box |
| **Actual Output** | True is printed in output display text box |
| **Pass** | **Pass** |
| **Screenshots** |  |

|  |  |
| --- | --- |
| **Test Case # 6** | |
| **Description** | Binary tree is not proper |
| **Input** | (A(G(j)(1))(z(5))) |
| **Expected Output** | Print false in the output display text box |
| **Actual Output** | False is printed in output display text box |
| **Pass** | **Pass** |
| **Screenshots** |  |

|  |  |
| --- | --- |
| **Test Case # 7** | |
| **Description** | Get Binary tree height |
| **Input** | (A(G(j)(1))(z(5))) |
| **Expected Output** | Print 2 as height in the output display text box |
| **Actual Output** | 2 is printed in output display text box |
| **Pass** | **Pass** |
| **Screenshots** |  |

|  |  |
| --- | --- |
| **Test Case # 8** | |
| **Description** | Get number of nodes of a Binary tree |
| **Input** | (A(G(j)(1))(z(5))) |
| **Expected Output** | Print 6 as number of nodes in the output display text box |
| **Actual Output** | 6 is printed in output display text box |
| **Pass** | **Pass** |
| **Screenshots** |  |

|  |  |
| --- | --- |
| **Test Case # 9** | |
| **Description** | Inorder format of a Binary tree |
| **Input** | (A(G(j)(1))(z(5))) |
| **Expected Output** | (((j)G(1))A((5)z)) |
| **Actual Output** | (((j)G(1))A((5)z)) |
| **Pass** | **Pass** |
| **Screenshots** |  |

|  |  |
| --- | --- |
| **Test Case # 10** | |
| **Description** | Incorrect syntax error message |
| **Input** | (A(G(j)(1))(z(5)) |
| **Expected Output** | Error message with "Input has incorrect syntax. Please enter a valid tree expression" |
| **Actual Output** | Popup Error message with "Input has incorrect syntax. Please enter a valid tree expression" |
| **Pass** | **Pass** |
| **Screenshots** |  |

|  |  |
| --- | --- |
| **Test Case # 11** | |
| **Description** | Binary tree can have only two child nodes error message |
| **Input** | (A(G(j)(1))(z(5))0 |
| **Expected Output** | Error message with " Binary tree can have only two child nodes" |
| **Actual Output** | Popup Error message with " Binary tree can have only two child nodes" |
| **Pass** | **Pass** |
| **Screenshots** |  |

|  |  |
| --- | --- |
| **Test Case # 12** | |
| **Description** | Incorrect syntax error message |
| **Input** | (A(G(j)(1))(z(5)))) |
| **Expected Output** | Error message with " Input has invalid parenthesis. Please enter a valid tree expression" |
| **Actual Output** | Popup Error message with " Input has invalid parenthesis. Please enter a valid tree expression" |
| **Pass** | **Pass** |
| **Screenshots** |  |

# **Lesson Learned**

Upon the completion of this project, I got to learn and grasp a lot about using more advanced JAVA data structures and things in that nature. I learned about creating a binary tree using the JAVA Programming Language. After completing this project, I feel that I am more advanced in the JAVA programming language, and I feel that the skills that I have learnt so far will take me to great ways in my career and life. Also, some more things I learnt after completing this project is about using the stack in different ways for nodes on handling the parent and the child nodes with a nested class being created itself. I learned about creating a more advancer type of GUI using the JAVA programming language, I learned about the various binary tree operations such as using a full tree implementation, the proper tree, a balanced tree using the recursive approach method of the JAVA programming language. Also, I learned about the utilization of the toString method of specifically overriding of the various methods used in the entirety of this project. Also, I got to learn about the designing error dialog message for the project itself. Lastly, I learned about the custom error handling process and using checked exception classes to throw an exception at runtime depending on the error situation at that time.