PRogrAmming III AT2.2

Andrew Tuitupou

A picture containing graphical user interface

Description automatically generated

Programming III

27/04/21

Design specification

Analysis

1. Purpose (Context)

I have been hired as the new programmer by the Jupiter Mining Corporation to produce a set of programs for their staff. I will be expected to consider my lecturer a key stakeholder and may ask questions to elicit information, this can then be documented and added to the specification document. The developers must create a program that includes a mechanical parts list which is to be sorted alphabetically. The balanced binary search tree for a dictionary of no less than 10 words. Where the program must be capable of searching the list, deleting from the list and adding to the list.

1. Stakeholders (Scope)

The key Stakeholder within this project includes my lecturer who will act as the project manager, sales representative, software developers, technical writers, systems analysts, Jupiter mining corporations management who will be requested for permission to perform a survey asking Jupiter staff about their current systems critical feedback response. The survey will not be mandatory. Client meetings with Jupiter mining will be organised by the project manager which will determine the suitable functionality of the future program addressing the companies needs.

2. Design Procedure (Methodology)

Design patterns include previous projects developed by software developers across companies. Patterns will determine the nature of the design as well as the software tools used too develop the program. Software development companies use a specific software development lifecycle. For this project the most modern software development lifecycle will be chosen which is the agile methodology. The steps within the agile lifecycle are Planning, analysis, Design, Implementation, Testing and Maintenance. The cycle’s steps can be completed multiple times. This keeps the cycle up to date with client expectations.

1. Software Tools

Interviews with Jupiter mining staff and management (lecturer) have revealed that the agile methodology is the best methodology for this project. The software tools that will be used for the development of this project are used for the steps within the agile development methodology which are Planning, analysis, Design, Implementation, Testing and Maintenance. Listed below in bullet points are the software tools which will be used too develop the program

* Microsoft Word (2021)
* MS Project (2021)
* Visual Studio C# WPF .NET (2019)
* Google Chrome (research)
* Website Draw.io

1. Core design

Diagram

Description automatically generated

1. Payroll Functionality

The main Functionality of the program is a mechanical parts list which uses binary trees the java program has been selected as the main program. According to the purpose statement that program must “Be able to display a parts name-ordering list. I need to create a balanced binary search tree for a dictionary of no less than 10 mechanical parts. I must be able to search the list, add and remove from the list”. The list must also be in alphabetical order.

1. Implementation

The target system is any computer capable of running windows XP at least, the recommended system is a system that can run windows10. The implementation plan for the project is to create an immediate prototype labled Apha1.0. Once the alpha version has been released any bugs will be reported and fixed releasing versions. Once the program has become error free a new version Beta will be released with added features and so forth until the final version of the program can be released. Please see GitHub for more of this plan, using the name **Atuitupou2- Search this name on GitHub for the valid repository**

Testing

|  |  |  |
| --- | --- | --- |
| ADD | DELETE | SEARCH |
| **Strings A (10 parts)** | **After ADD Delete Strings A** | **After ADD Search Strings A** |
| **Strings B (10 parts)** | **After ADD Delete Strings B** | **After ADD Search Strings B** |
| **Strings C (10 parts)** | **After ADD Delete Strings C** | **After ADD Search Strings C** |
| **Strings D (10 parts)** | After ADD Delete Strings D | **After ADD Search Strings D** |

\*10 names are displayed in differing orders after each add test.

|  |  |  |
| --- | --- | --- |
| ADD (SUCCESS) | DELETE (SUCCESS) | SEARCH (SUCCESS) |
| **CHECK Appendix A0** | **CHECK Appendix A1** | **CHECK Appendix A0** |
| **CHECK Appendix B0** | **CHECK Appendix B1** | **CHECK Appendix B0** |
| **CHECK Appendix C0** | **CHECK Appendix C1** | **CHECK Appendix C0** |
| **CHECK Appendix D0** | CHECK Appendix D1 | **CHECK Appendix D0** |

Debugging

|  |  |  |
| --- | --- | --- |
| ADD/SEARCH | ADD results | WORKNG (SUCCESS) |
| **Strings A (10 names)** | **Appendix A0** | **CHECK** |
| **Strings B (10 names)** | **Appendix B0** | **CHECK** |
| **Strings C (10 names)** | **Appendix C0** | **CHECK** |
| **Strings D (10 names)** | **Appendix D0** | **CHECK** |

\*The add function to binary tree is automatically tested if the search function is successful.

Appendix A0

|  |
| --- |
| Text  Description automatically generated |
| Graphical user interface, text, application  Description automatically generatedGraphical user interface, application  Description automatically generated |

Appendix B0

|  |
| --- |
| Text  Description automatically generated |
| Graphical user interface, text, application, email  Description automatically generatedGraphical user interface, text, application  Description automatically generated |

Appendix C0

|  |
| --- |
| Graphical user interface, application  Description automatically generated |
| Graphical user interface, text, application, email  Description automatically generatedGraphical user interface, application  Description automatically generated |

Appendix D0

|  |
| --- |
| Graphical user interface  Description automatically generated with medium confidence |
| Graphical user interface, text, application  Description automatically generatedGraphical user interface, application  Description automatically generated |

Appendix A1

|  |
| --- |
| Graphical user interface, text, application  Description automatically generatedGraphical user interface, text, application, Word  Description automatically generated |

Appendix B1

|  |
| --- |
| Graphical user interface, text, application  Description automatically generatedGraphical user interface, text, application  Description automatically generated |

Appendix C1

|  |
| --- |
| Graphical user interface, text, application, email  Description automatically generatedGraphical user interface, text, application, email  Description automatically generated |

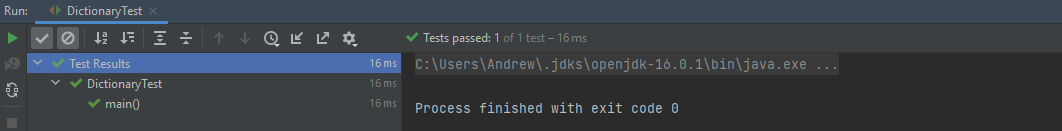
Appendix D1

|  |
| --- |
| Graphical user interface, text, application  Description automatically generatedGraphical user interface, text, application  Description automatically generated |

Junit testing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SEARCH | StringA | StringB | StringB | StringC | StringD | SUCCESS |
| ADD | StringE | StringF | StringG | StringH | StringZ | SUCCESS |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SEARCH | StringA1 | StringB2 | StringB3 | StringC4 | StringD5 | SUCCESS |
| ADD | StringE1 | StringF2 | StringG3 | StringH4 | StringZ5 | SUCCESS |



# References

AfterAcademy. (2021, 04 27). *Sort Array To Balanced Binary Tree*. Retrieved from AfterAcademy: https://afteracademy.com/blog/sorted-array-to-balanced-bst

kc70. (2021, April 27). *Binary Search Tree Implemented in C#*. Retrieved from Youtube: https://www.youtube.com/watch?v=pN1RWeX47tg&t=206s