**Software Development Document**

1. **Software Specifications**
   1. **Functions of Data Structures**

The Software implements a total of 5 Data Structures:

* + 1. **Unsorted Array**

The unsorted array structure has the following Functions:

1. **Add**: adds the element at the end of the array
2. **Remove**: finds the target and removes if found, false otherwise
3. **Find**: private method that finds target using Sequential Search
4. **Contains**: returns true if target is found, false otherwise
5. **Get**: finds the target and returns a copy of the target if found, null otherwise
6. **isEmpty**: returns true if array is empty, false otherwise
7. **isFull**: returns true if array is full, false otherwise
8. **Size**: returns the number of elements in the array
9. **Print**: prints all the elements in the array
   * 1. **Sorted Array**

The Sorted array structure has the following Functions:

1. **Add**: adds the element at the end of the array
2. **QuickSort**: sorts the array using quicksort algorithm
3. **Find**: private method that finds target using Sequential Search
4. **Contains**: returns true if target is found, false otherwise
5. **Get**: finds the target and returns a copy of the target if found, null otherwise
6. **BinarySearch**: finds target in the sorted array using Binary Search Algorithm
7. **isEmpty**: returns true if array is empty, false otherwise
8. **isFull**: returns true if array is full, false otherwise
9. **Size**: returns the number of elements in the array
10. **Print**: prints all the elements in the array
    * 1. **Unsorted Linked list**

The unsorted linked list structure has the following Functions:

1. **Add**: adds the element at the end of the linked list
2. **Remove**: finds the target and removes if found, false otherwise
3. **Find**: private method that finds target using Sequential Search
4. **Contains**: returns true if target is found, false otherwise
5. **Get**: finds the target and returns a copy of the target if found, null otherwise
6. **isEmpty**: returns true if linked list is empty, false otherwise
7. **Size**: returns the number of elements in the linked list
8. **Print**: prints all the elements in the linked list
   * 1. **Sorted Linked list**

The Sorted linked list structure has the following Functions:

1. **Add**: adds the element in sorted order in the linked list
2. **toAdd**: finds the appropriate place in the linked list to add
3. **Remove**: finds the target and removes if found, false otherwise
4. **Find**: private method that finds target using Sequential Search
5. **Contains**: returns true if target is found, false otherwise
6. **Get**: finds the target and returns a copy of the target if found, null otherwise
7. **isEmpty**: returns true if linked list is empty, false otherwise
8. **Size**: returns the number of elements in the linked list
9. **Print**: prints all the elements in the linked list
   * 1. **Binary Search Tree**

The Binary Search Tree structure has the following Functions:

1. Add: adds the element in BST according to value property
2. Search: finds the target in BST returns true if found, false otherwise
3. Get: finds the target in BST returns a copy if found, null otherwise
4. isEmpty: returns true if BST is empty, false otherwise
5. **Size**: returns the number of elements in the BST
6. **inOrder**: prints all the elements in BST by in-order traversal
   1. **Input Data Sets**

The program offers to input 2 types of data:

* + 1. **Customer Data File**

The Customer Data File is a string data file ***“Book1.txt”*** that contains 3 attributes, ***First Name***, ***Second Name*** and ***Customer ID***. All sorting is maintained in ascending order of customer ID, all searching algorithms take ***Target ID*** as an input and find a customer with ID. The Customer IDs are ***unique*** however customer names can be duplicates.

* + 1. **Random Generated Integers**

**2000** Random numbers are generated using ***Java Random Class***, these are stored in Integer Instantiated objects of the data structure. This data set can have duplicates, sorting is done in ascending order. Searching algorithm finds only the 1st duplicate and returns.

* 1. **Functions for the Client**

Each data structure has a number of methods, however not all these methods have been made accessible for the client. Only the requirements of the project have been made available for the client, these available functions are as follows:

* + 1. **Menu 1:** Client can choose the data structure
    2. **Menu 2:** Client has the option to choose the type of Input
    3. **Menu 3:** Client can enter ID/Number to search in the data structure
  1. **Searching Algorithms**
     1. **Searching Algorithms**

The unsorted array, sorted array, unsorted linked list and sorted linked list data structure use ***Sequential Search.*** The sorted array also implements ***Binary Search*** Algorithm, whereas Binary Search Tree uses binary tree value property to search in a BST data structure. All searching algorithms search by ***Customer ID/Integer Number***.

* + 1. **Sorting Algorithms**

The sorted array structure uses ***Quick Sort*** Algorithm to sort data, whereas the sorted linked list uses ***Sorting by Insertion*** to maintain a sorted linked list. All sorting is maintained in ascending order of ***Customer ID/Integer Number***.

* 1. **Complexity Analysis**

The program uses ***System Runtime Measurement*** to display running complexity of searching algorithms, when the user searches for a Customer ID/Number in the Data Structure.

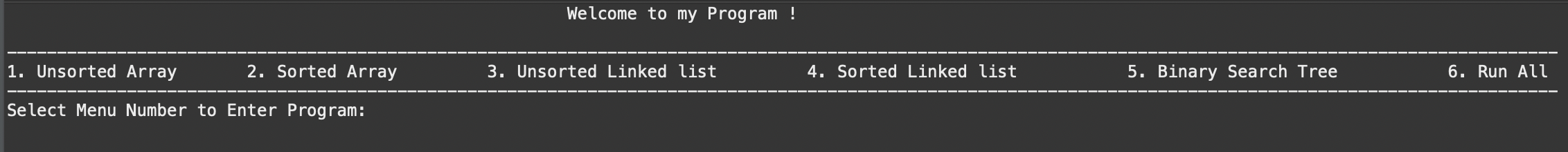
The program calculates Complexity Analysis in accordance with the theory of these topics, however sometimes the Runtime Measurement output differs from theoretical concepts because of ***Eclipse* *Cache***. The complexity of the structure that’s run later, displays considerably less Runtime, this is because the first time a structure is searched it is stored in Eclipse’ Cache Memory. To avoid this a search is called before Measuring its Runtime to try to measure both searching algorithm at the same level. A better solution would be to clear cache memory before calling each searching algorithm, however doing that from Eclipse is not possible in mac (as per my knowledge).

1. **Design Diagram**

****

1. **Operational Document**

The first Menu that appears, displays all the data structures to choose to run from or run all. The input value must be an ***integer***.

****

Upon selecting any option, another Menu appears that asks to choose input data type – ***Customer Data File*** or ***Random Generated Numbers***. The input value must be an ***integer***.

Text

Description automatically generated

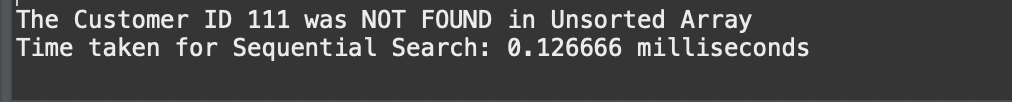
If Customer Data File is selected (Input = 1), then program prints the list and asks to enter ***Customer ID*** to search from the data structure. The input value must be an ***integer***.

Timeline

Description automatically generated

****

Based on the data structure selected the program outputs if the input Customer ID was found and System Runtime Measurement if found.



Graphical user interface, text

Description automatically generated

If Random Generated Numbers is selected (Input = 2), then program prints the list and asks to enter ***Number*** to search from the data structure. The input value must be an ***integer***.

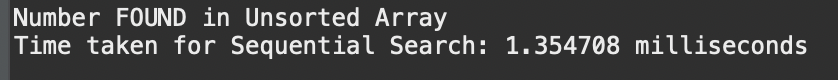
**Text

Description automatically generated**

****

Based on the data structure selected the program outputs if the input Customer ID was found and System Runtime Measurement if found.





For Run All option for Customer Data Set:

Text

Description automatically generated

Text

Description automatically generated

For Run All option for Random Generated Numbers:

Text

Description automatically generated

Text

Description automatically generated

1. **README**

Instructions to build and execute Project:

1. Assume all files in one folder
2. Assume ***“Book1.txt”*** file in same Eclipse Project Folder
3. javac \*.java
4. java Main.java
5. All inputs by user must be ***integer***
6. **Data**

The Customer Data File is a string data file ***“Book1.txt”*** that contains 3 attributes, ***First Name***, ***Second Name*** and ***Customer ID***. Random numbers are generated using ***Java Random Class.***

1. **Project Management Schedule**

|  |  |  |
| --- | --- | --- |
|  | Complete by | *Each BOLD word represents a page in your Research Report and Board.* |
| 1 | 5 Days | Understand the Project requirements, 1hr each day. |
| 2 | 2 Days | Determine the deliverables, 1hr each day. |
| 3 | 2 Days | Specify what the Software must do, 1hr per day. |
| 4 | 5 Days | Design and implement code, 3hr per day. |
| 5 | 3 Days | Testing and Verification, 3hr per day. |
| 6 | 2 Days | Prepare SDLC based Documentation, 3hr per day. |

1. **Complexity Analysis**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Unsorted Array | Sorted Array | Unsorted Linked list | Sorted Linked list | Binary Search Tree |
| Sequential  Search | **O(N)** | **O(N)** | **O(N)** | **O(N)** | **-** |
| Binary Search | **-** | **O(Log2N)** | **-** | **-** | **-** |
| BST Search | **-** | **-** | **-** | **-** | **O(Log2N)** |