Software Development Document

1. Software Specifications

1.1. Functions of Data Structures

The Software implements a total of 5 Data Structures:

i) **Unsorted Array**

The unsorted array structure has the following Functions:

a) **Add**: adds the element at the end of the array

b) **Remove**: finds the target and removes if found, false otherwise

c) **Find**: private method that finds target using Sequential Search

d) **Contains**: returns true if target is found, false otherwise

e) **Get**: finds the target and returns a copy of the target if found, null otherwise

f) **isEmpty**: returns true if array is empty, false otherwise

g) **isFull**: returns true if array is full, false otherwise

h) **Size**: returns the number of elements in the array

i) **Print**: prints all the elements in the array

ii) **Sorted Array**

The Sorted array structure has the following Functions:

a) **Add**: adds the element at the end of the array

b) **QuickSort**: sorts the array using quicksort algorithm

c) **Find**: private method that finds target using Sequential Search

d) **Contains**: returns true if target is found, false otherwise

e) **Get**: finds the target and returns a copy of the target if found, null otherwise

f) **BinarySearch**: finds target in the sorted array using Binary Search Algorithm

g) **isEmpty**: returns true if array is empty, false otherwise

h) **isFull**: returns true if array is full, false otherwise

i) **Size**: returns the number of elements in the array

j) **Print**: prints all the elements in the array

iii) **Unsorted Linked list**

The unsorted linked list structure has the following Functions:

a) **Add**: adds the element at the end of the linked list

b) **Remove**: finds the target and removes if found, false otherwise

c) **Find**: private method that finds target using Sequential Search

d) **Contains**: returns true if target is found, false otherwise

e) **Get**: finds the target and returns a copy of the target if found, null otherwise

f) **isEmpty**: returns true if linked list is empty, false otherwise

g) **Size**: returns the number of elements in the linked list

h) **Print**: prints all the elements in the linked list

iv) **Sorted Linked list**

The Sorted linked list structure has the following Functions:

a) **Add**: adds the element in sorted order in the linked list

b) **toAdd**: finds the appropriate place in the linked list to add

c) **Remove**: finds the target and removes if found, false otherwise

d) **Find**: private method that finds target using Sequential Search

e) **Contains**: returns true if target is found, false otherwise

f) **Get**: finds the target and returns a copy of the target if found, null otherwise

g) **isEmpty**: returns true if linked list is empty, false otherwise

h) **Size**: returns the number of elements in the linked list

i) **Print**: prints all the elements in the linked list

v) **Binary Search Tree**

The Binary Search Tree structure has the following Functions:

a) Add: adds the element in BST according to value property

b) Search: finds the target in BST returns true if found, false otherwise

c) Get: finds the target in BST returns a copy if found, null otherwise

d) isEmpty: returns true if BST is empty, false otherwise

e) **Size**: returns the number of elements in the BST

f) **inOrder**: prints all the elements in BST by in-order traversal

1.2. Input Data Sets

The program offers to input 2 types of data:

i) 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.

ii) 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.

1.3. 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:

i) **Menu 1:** Client can choose the data structure

ii) **Menu 2:** Client has the option to choose the type of Input

iii) **Menu 3:** Client can enter ID/Number to search in the data structure

1.4. Searching Algorithms

i) 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***.

ii) Sorting Algorithms

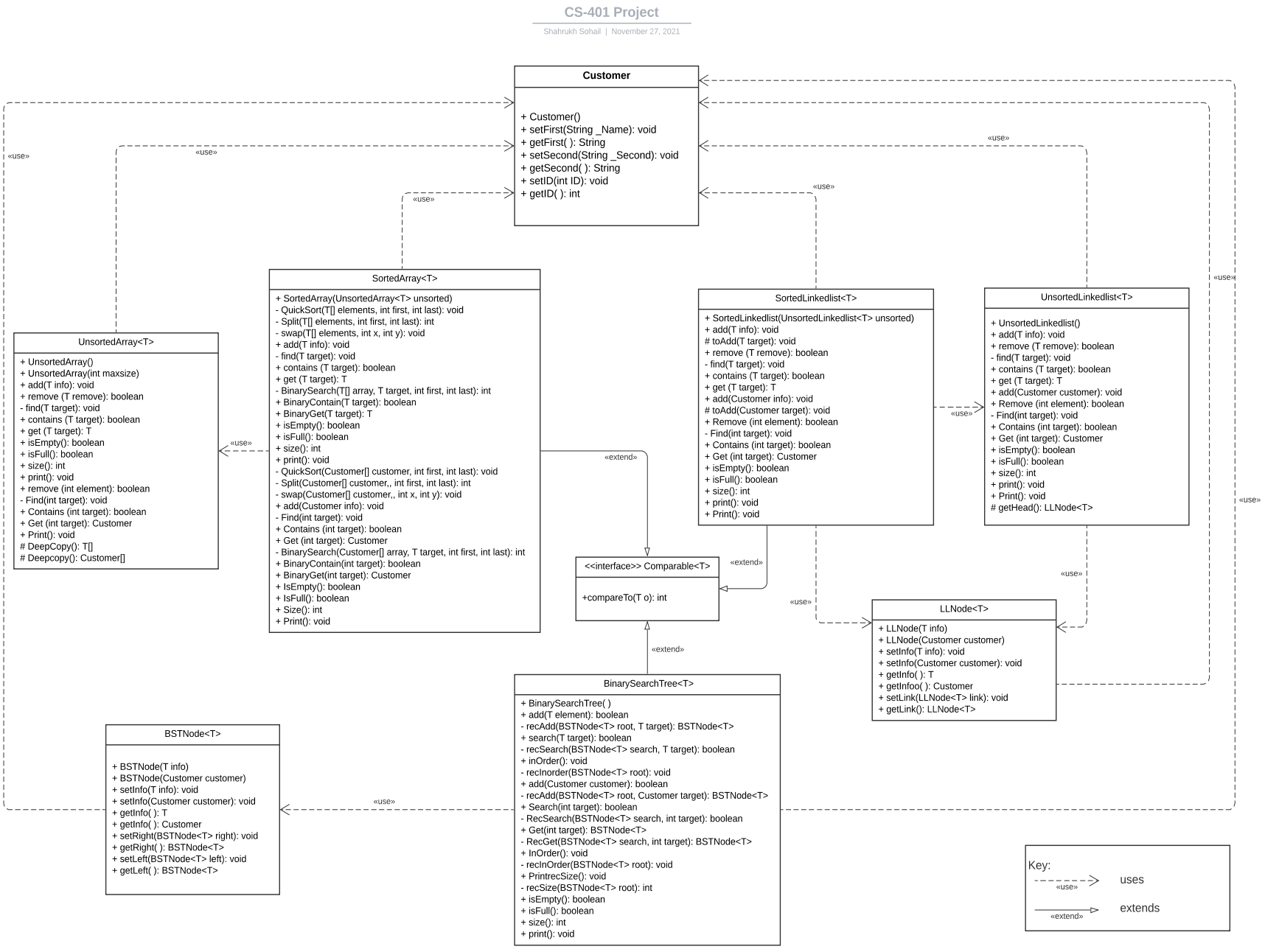
The sorted array structure uses ***Quick Sort*** Algorithm to sort data, whereas the sorted linked list uses ***Sorting by Insertion*** to maintain a sorted linked list. All sorting is maintained in ascending order of ***Customer ID/Integer Number***.

1.5. 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).

2. Design Diagram



3. 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***.

```
                                        Welcome to my Program !
-------------------------------------------------------------------------------------------------------------------------------
1. Unsorted Array       2. Sorted Array        3. Unsorted Linked list        4. Sorted Linked list          5. Binary Search Tree          6. Run All
-------------------------------------------------------------------------------------------------------------------------------
Select Menu Number to Enter Program:
```

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***.

```
                        Unsorted Array
-------------------------------------------------------------------------------
1. Input Customer Data File             2. Input Random Generated Numers
-------------------------------------------------------------------------------
Select Menu Number to Enter Program:
```

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***.

```
-----------------------------------------------------
        Adding Customer Data File in Unsorted Array
-----------------------------------------------------
100577          Emily           Burns
3001146         Lindsay         Shagiari
4003656         Helen           Andreada
5004063         Dave            Brooks
7004063         Dianna          Wilson
9001918         Ben             Peterman
700290          Adam            Bellavance
6001713         Julia           Dunbar
400240          Darrin          Van
300577          Sean            O'Donnell
2001487         Brosina         Hoffman
1002365         Andrew          Allen
600246          Ken             Black
101010          Shahrukh                Sohail
7004977         Gene            Hale
6002986         Steve           Nguyen
5004093         Linda           Cazamias
3003282         Erin            Smith
1001713         Patrick         O'Donnell
1002167         Lena            Hernandez
2004410         Darren          Powers
400604          Ted             Butterfield
5004633         Kunst           Miller
```

```
Enter Customer ID to Search:
```

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

```
The Customer ID 111 was NOT FOUND in Unsorted Array
Time taken for Sequential Search: 0.126666 milliseconds
```

```
Customer ID FOUND in Unsorted Array
Customer Name: Lan Nguyen
Time taken for Sequential Search: 0.122667 milliseconds
```

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***.

```
------------------------------------------------------------
        Adding Random Generated Numers in Unsorted Array
------------------------------------------------------------
2799
1950
32829
32157
36517
65890
43039
47794
96564
53875
89281
25112
71979
96369
33249
99291
85654
57475
45049
70153
73070
66346
39057
51002
89807
87526
26789
```

```
Enter Number to Search:
```

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

```
The Number 111 was NOT FOUND in Sorted Array
```

```
Number FOUND in Unsorted Array
Time taken for Sequential Search: 1.354708 milliseconds
```

For Run All option for Customer Data Set:

```
Enter Customer ID to Search: 111

Customer ID NOT FOUND
Time taken for Sequential Search in Unsorted Array:            66583.0 nanoseconds
Time taken for Sequential Search  in Sorted Array:             10250.0 nanoseconds
Time taken for Binary Search  in Sorted Array:                 10125.0 nanoseconds
Time taken for Sequential Search in Unsorted Linked list:      107500.0 nanoseconds
Time taken for Sequential Search in Sorted Linked list:        26042.0 nanoseconds
Time taken for Binary Tree Search:                             27417.0 nanoseconds
```

```
Enter Customer ID to Search: 202020

Customer ID FOUND
Customer Name: Lan Nguyen
Time taken for Sequential Search in Unsorted Array:             46875.0 nanoseconds
Time taken for Sequential Search  in Sorted Array:              2334.0 nanoseconds
Time taken for Binary Search  in Sorted Array:                  9375.0 nanoseconds
Time taken for Sequential Search in Unsorted Linked list:       92375.0 nanoseconds
Time taken for Sequential Search in Sorted Linked list:         7083.0 nanoseconds
Time taken for Binary Tree Search:                              8083.0 nanoseconds
```

For Run All option for Random Generated Numbers:

```
Enter Number to Search: 111

Number NOT FOUND
Time taken for Sequential Search in Unsorted Array:            232167.0 nanoseconds
Time taken for Sequential Search  in Sorted Array:             185792.0 nanoseconds
Time taken for Binary Search  in Sorted Array:                 16750.0 nanoseconds
Time taken for Sequential Search in Unsorted Linked list:      322291.0 nanoseconds
Time taken for Sequential Search in Sorted Linked list:        606917.0 nanoseconds
Time taken for Binary Tree Search:                             189416.0 nanoseconds
```

```
Enter Number to Search: 9657

Number FOUND
Time taken for Sequential Search in Unsorted Array:            287250.0 nanoseconds
Time taken for Sequential Search  in Sorted Array:             27250.0 nanoseconds
Time taken for Binary Search  in Sorted Array:                 20125.0 nanoseconds
Time taken for Sequential Search in Unsorted Linked list:      496542.0 nanoseconds
Time taken for Sequential Search in Sorted Linked list:        68500.0 nanoseconds
Time taken for Binary Tree Search:                             1342042.0 nanoseconds
```

4. 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***

5. 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.***

6. 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.

7. 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	-	**$O(Log_2N)$**	-	-	-
BST Search	-	-	-	-	**$O(Log_2N)$**