Assignment 3 Report

MPNTLO002 Tlotliso Mapana

Object Oriented Design

Below is a description of classes created and how they are related.

Classes	Description
IDGenerator	To be run only once, this class takes in a text file which has a list of 100 random names and extracts them. Then generates a random 13-digit ID number and assigns it to a name, and appends that string to a new text file, it does this for all 100 names.
DataChain	This class has an inner class Node which creates a node to be added to a BST. It uses chaining method for collision resolution when adding items to the hash table. This class has methods to add items to the hash table, compute hash values, and to search through the hash table for the desired key value.
UserInterface	This program is the GUI to be used at the voting station to search through the hash table for a desired value. In the main function it reads in the text file of 100 names and ID numbers and searches iteratively through the file as ID numbers are being entered, displays the names of the voter if key is found.

Application

The user interface chosen for this application is a graphical user interface which uses a java program to search through a hash table, continuously.

Below is the chosen hash function, which implements Chaining for collision resolution:

```
public int computehash (String key, int tableSize)
	{
		int hashIndex = 0;
		int temp = 0;
		for (int i = 0; i< key.length(); i++){
			/** Convert string (key) into a natural number **/
			temp = 1 * (temp + (int)key.charAt(i)); /**radix-37
notation**/
		}
		/** compute index in hash table **/
			hashIndex = temp % tableSize; /**Not good if hash table is
large**/
		return hashIndex;
}
```

This hash function was inherited from previous CSC2001F code samples and uses radix-37 notation when calculating the hash value of a key. This is a good implementation as we can ensure that there are no duplicates.

The default size of the hash table array was chosen to be 100 (as the text file of voters has 100 items) but the size can be adjusted to add more hash keys into the array.

Below is the piece of code in the user interface class that enables continuous searching of ID numbers:

```
Scanner keyboard = new Scanner (System.in);
System.out.println("Enter a 13-digit ID number: ");
while (keyboard.hasNextLong())
{
       String id = keyboard.nextLine ();
      // Exit case
      if (Long.parseLong(id) == 0)
        System.exit(0);

      if (id.length() == 13)
      {
        int found = table.findKey(id, tablesize);
        System.out.println("Voter registered!");
        table.display(found);
}

System.out.println("Enter a 13-digit ID number: ");
}
```