ADS2 – Telephone Banking System

Design

When designing the data structures for the telephone banking system I decided that a Singly Linked List would be an appropriate data structure to store the queue of customers; Linked List as there is no random access required – only sequential – and Singly Linked List as there is no need for backwards traversal of the queue items.

My implementation of a Singly Linked List in the CustomerQueue class is fairly minimal – I strived to keep the code simple to achieve faster processing at runtime and an easier to understand structure.

I designed a basic Customer class to store the customer's ID and balance, with some basic methods to manipulate and retrieve these values. I tried to ensure throughout the program that any changes to a Customer object would be performed via these public mutator methods, in order to uphold object responsibilities and the integrity of the data.

The handling of the list is done, for the same reasons, via the CustomerQueue class and its methods wherever possible throughout the program.

The rest of the program code is mainly just method/function calls when required by the finite-state machine, with some user inputs and console outputs when necessary.

Big-O

Adding a new customer

The method for adding a new customer has a Big-O of O(n) because the code is required to traverse the entire linked list and add the new customer at the end.

Removing a customer (from the front of the queue)

The method for getting the next customer (removing the customer at the front of the queue) has a Big-O of O(1) as all the code has to do is get the Customer object out of the first linked node (the head) and assign the next (2nd) node as the new head; the list could be infinitely long and it would still only be these two operations.

Removing a specific customer (by ID)

The method for removing a specific customer has a Big-O of O(n) because the code *may* have to traverse the entire linked list before it finds the ID of the customer requested to be removed. This does not mean it will always take the maximum amount of time, but as the data structure is a singly linked list and also not sorted (it cannot be as the IDs are character strings) a sequential search is the only option.