Cristian Cantoral, Dylan Shane, Tim Clerico
Comp 220
Dr. Toby Dragon
12/18/2017

Book Store

The ADTs used in our Bookstore were Queue and Inventory; both of these were implemented with modified LinkedNode structures. We created two types of nodes, a book node and a person node, each keeping track of their respective information. For our waiting list we used implemented a LinkedQueue which was a subclass of Queue, and our inventory of books was held implemented in a LinkedInventory structure which was a subclass of Inventory.

We used the queue inside the Book class for our waiting list, we believed this was the best choice because in a real life scenario a waitlist should serve as FIFO. This makes it time efficient because we are able to add and remove from our queue at O(1) speed. Adding and removing are the primary methods of Queue, so we decided a LinkedQueue structure would be the best implementation.

Our second ADT is Inventory, which was implemented using LinkedInventory. This structure inserts books in alphabetical order making it an O(n) function since it could have to insert a book to the end of the list. We believe that printing the inventory command will be used more frequently than the add book command, so we decided to insert the books in order to avoid having to sort the entire inventory every time the user requested to list out the inventory. Sorting the inventory takes $O(n^2)$ time; therefore, having to sort the list before listing the inventory would be less efficient than having the inventory already sorted. Like the waitlist, the Inventory uses a linked node structure which we decided on for the purpose of memory efficiency. Though we realize that most computers will be capable of handling our programs memory capacity, our implementation is more space efficient than an array implementation would be due to an array having a finite number of items and having to double its capacity.

MEMORY DIAGRAM: