# Chosen Collections

* HashSet – storing the menu when read in from a txt file.
* HashTree – storing the collection of customers/orders.
* LinkedList – storing the items within an order.

**HashSet**

* Stores each item object and associates them by a ‘key’ that can be searched for. Using the items name as its key all other properties tied to it can be easily found when referenced for an order.
* Only allows unique entries. Stops multiple entries of the same item being added to the menu.

**HashTree**

* Stores each object in a sorted tree and associates them by a ‘key’.
* Using the customer/order ID as the key, a sorted set of customers/orders, from the first down to the last, can be returned.

**LinkedList**

* Stores each entry in insertion order with a pointer to the next entry in the list.
* While the entries cannot be access via index, using a list allows for easy deletion and insertion of entries.
* The LinkedList will be used to store the items in each order. It will allow for easy correction if a customer wants to remove an item that was added earlier to the list.
* LinkedList’s are dynamic, so memory doesn’t have to be pre-allocated before starting the order. So, no number is placed on the number of items that can be ordered at any one time.
* This will also allow duplicate items to be added to the order.

The coffee shops menu will be created from the text file read in at the beginning of the program by instantiating a *menu* object and filling its corresponding hashSet with the reported items. The *item* objects will be stored as the ‘value’ with the ‘key’ being the items name. In doing so this will allow for easy referencing and searching of the available menu items. Another property of the hashSet implementation of hashTables that is beneficial in our application is that they ensure that only unique entities are added. This helps prevent multiple entrees of the same item being added to the menu if there is an error in the text file.

All customer objects that have been created during the running of the application will be stored in a hashTree data structure within the *allCustomers* object which is created at the start of the application. Each customer object will be stored as the ‘value’ and their corresponding customer ID will be used as the ‘key’. By using the tree implementation, a Set of customer objects, sorted using their customer IDs, can be returned. Allowing easy access to the customer object either in order of appearance or by searching by a specific ID.

Each *customer* object created will have to store the items they order within the object. A LinkedList will be used to do this. By storing the customer’s order in its order of insertion will allow for easier and more natural storing and recital. LinkedLists also allow for efficient insertion and deletion, so if a customer wants to remove an item previously added to their order this can be done with minimal disruption. Another property of linkedLists that make them suitable for this application is that they can be dynamically allocated. This means that an order limit doesn’t have to be set. They also allow for duplicate items to be added.