Marlon Aquino

Professor Dominick Atanasio

Class: CS 2400-03

Project: LinkedBag

Due Date: 9/13/2018

In this project, I needed to implement the BagInterface, but instead of using an array like the project in Exercise 1, I used a singly linked list as discussed in lecture. A linked list is a type of data structure that has a group of nodes in a sequence. Each node has data and contains a reference to the next node. The new class that I needed to create is called LinkedBag which is a class of a bag whose entries are stored in a chain of linked nodes. This bag is never full and the time it takes to iterate for each method depends on the number of entries in the bag.

In order to test my code, I first created a new java class file called Main.java and proceeded to create the Main class. Next I created the main method. Then I made a new LinkedBag object called bag containing strings. I then tested the isEmpty() method as well as the getCurrentSize() method to make sure it returns true and 0 respectively. I then added items to each node using the add() method then used the toArray() method to print out each item. From this step, I tested out the rest of the methods and kept using the toArray() method to make sure each method was altering the LinkedBag correctly.

From this exercise, I gained a stronger visualization of how singly linked lists work. The diagram that Professor Atansio drew on the board especially helped me in programming the exercise. Compared to the last exercise where we implemented the BagInterface using an array, we used a linked list which is a better way to manage memory because the memory in each node can be used for a specific purpose. Also, unlike arrays, the size of a linked list is dynamic.

Perhaps the most difficult part of the project for me was the remove(T anEntry) method. In my understanding of the exercise, I needed to get a reference to the entry that the user wants to delete and then remove one occurrence of that entry by iterating through the linked bag. I knew that I had to compare anEntry and iterate through the bag then remove that entry, but I had

problems with comparing anEntry with a Node object. I first used the equals method then compared it to the current node that the program was on. The problem was however, that I was comparing a T object to a Node. I then remembered that there was a private Node class that had a private T object called item. I then used currentNode.item to compare it with anEntry. I also used a private helper method called getItem(T anEntry) that returns the reference of that entry to make it easier with the remove(T anEntry) method.