### Portfolio Function Title: Queue

Version number: 1

### Function Description:

Allows for a user to build queue of objects. Once a queue is built a user can add first (Enqueue), remove last (Dequeue) and get first (LookAtFirst). A user can also find the size of the queue (GetSize) and check whether it is empty (CheckIfEmpty).

### Author:

Nathan Mankovich

### Date Written/Last Modified:

04-27-2015 / 05-01-2015

### Design overview:

Queue relies on the instantiation of the LinkedList.

Each method in Queue.java calls the LinkedList program to fill a singly linked list that was instantiated at the beginning of Queue.java.

The queue is populated in the Enqueue method. Objects are removed from the front and returned using the Dequeue method. The LookAtFirst method gets the object at the top without removing it. The CheckIfEmpty method checks if the queue is empty and returns a true Boolean if it is, otherwise the method returns a false boolean. The GetSize method returns the size of the queue as an integer.

### Feature Specifications:

Enqueue elements onto the top of the queue using AddToFront from LinkedList.

Dequeue elements off the top of the queue using RemoveLast from LinkedList.

LookAtFirst at the first element on the top of the queue using GetLast from LinkedList.

Get the size of the queue as an integer using the CheckIfEmpty from LinkedList.

Determine whether there are elements in the queue using the GetLength from LinkedList.

### Programmer User Interface:

User Interface: Java code and command line only.

### Input and Output Requirements and Restrictions:

* Enqueue:
  + Input: object to be added
  + Output: none
* Dequeue:
  + Input: none
  + Output: removed object
* LookAtFirst:
  + Input: none
  + Output: the first object
* CheckIfEmpty:
  + Input: none
  + Output: a Boolean, true if empty
* GetSize:
  + Input: none
  + Output: an integer that is the length of the queue

### Assumptions and Dependencies:

The queue assumes that a user will not use any removing or getting methods without checking to see that there are elements in the list.

### Known problems and limitations:

A queue is limited to placement at the front and removal from the back of the list of elements. Using queues allow a user to only look at the top element in the list. Yet, all functions in the LinkedList implementation of queue provides fast processing times.

### Use Cases:

Queue scan be used for data sets that need to be outputted in the same order they were inputted. Possibly a good use for queues is to store large amounts of ordered data and move the data through different methods.

Queues are best used for data that needs to be inserted at one end and removed at the other end.

### Testing Methodology:

Be sure to put the tQueue.java and the portfolio.jar file in the same folder.

To compile:

javac -cp .;portfolio.jar Queue.java

To run:

java -cp .;portfolio.jar tQueue > LogQueueTest.txt

All command line outputs will be in the new LogQueueTest.txt file in the folder that contains the tQueue and the portfolio.jar files.

### Modification history:

Version 1: 05-01-2015

### Design detail and/or Diagrams:

When the program starts, a LinkedList is instantiated and all methods in the Queue call methods in the LinkedList class. For more information about the implementation of the queue- specifically the storage of elements in a linked list, see the LinkedListPortfolioDesignDoc in the portfolio folder.