Lab: Linked List
CSC 207, “Algorithms and Object-Oriented Design”
Department of Computer Science
Grinnell College
February 24, 2021

Goal

In this lab, you will develop a singly linked list to implement queues. You will work with the following files:

- QueueNode.java stores queue items.
- Queue207.java defines an interface for queue operations.
- Queue207Implementation.java implements the Queue207 interface.

Create an Eclipse project called queues. Take QueueNode and Queue207 class files from the following path in MathLAN and import them to the Eclipse project:

/home/hajamini/courses/CSC207/code/

QueueNode

A Queue can be implemented by a singly linked list of QueueNodes. The QueueNode.java class has the following structure:

```java
public class QueueNode<AnyType> {
    private AnyType data; // data is stored inside a node.
    private QueueNode<AnyType> next; // this field references the next node.

    public QueueNode () {
        data = null;
        next = null;
    }

    public QueueNode( AnyType origData, QueueNode<AnyType> origNext ) { // this field references the next node.
        data = origData;
        next = origNext;
    }
}
```

The data field stores a reference to some actual data. The next field stores a reference to the next QueueNode of the list. This class has two overloaded constructors: one that sets the data and next fields to null and the other sets them to parameters passed to the constructor.

By considering the above structure, the Queue207Implementation class contains three private fields: numberItems, head, and tail. numberItems specifies the number of items in the queue. head and tail references mark the two ends of the queue as shown in the following figure:
Items are deleted from the head reference and inserted at the tail reference. This indicates that the items are placed into the queue in the order that they are inserted there. In other words, the item, referenced by head, is the first item inserted in the queue while the item, referenced by tail, is the last item inserted in the queue.

The implementation details of important methods in Queue207Implementation class are explained below:

- **Queue207Implementation** constructor: initializes three fields: numberItems, head, and tail. numberItems is initialized to zero. head and tail are initialized to null.
- **add** method: instantiates a new QueueNode that contains the specified data and null in the next field.
  - The next field of the QueueNode, referenced by the old tail of the queue, is updated to reference the new QueueNode.
  - The tail field of the queue is updated to reference the new QueueNode.
  - If the new QueueNode is the first QueueNode of the queue, the head and tail fields are updated to reference the new QueueNode.
- **remove** method: extracts data from a QueueNode referenced by the head field.
  - The head field is updated to reference the next QueueNode of the queue.
  - If data is extracted from the last QueueNode of the queue, the tail field is updated to reference null.

**Task**

Write the Queue207Implementation class that implements the Queue207 interface. Provide implementations for the constructor, add, remove, element, and size methods.

(6 points) Write Queue207ImplementationTester class to test all of the methods. As a test case, create a queue and call the following methods in-order: add, add, remove, element, size, add, size, remove, remove, size.

Write a JUnit test case that calls these methods in-order and checks whether they are correctly implemented.

**Acknowledgment**

This lab is based on a material made by Henry Walker.