COMP 2140 Lab 3 — A Queue for a Bank Simulation

Helen Cameron

Week of October 31, 2016

Objective

To use a queue to simulate a line-up at a bank.

Exercise

File `Lab03.java` contains a nearly-complete application that simulates customers being served by two bank tellers. A queue implementation is used to simulate the line up to wait for a free bank teller. You also need the input file, `bankSimulationData.txt`. **You do not need to read or understand any of the (many) classes in `Lab03.java` except the last two classes:**

- The second-last class is the generic-type `Node` class from the linked list course notes. The `Node` class is used in another class, not just the `Queue` class — and the two classes that use it store different types of items. Therefore, it is convenient to use generics, rather than creating two different node classes, each one storing a different type of item.
- The last class is an incomplete `Queue` class that implements the queue as a **circular linked list** with no dummy nodes, and using just an `end` pointer pointing to the last node in the list. Of course, because it's a circular linked list, that last node doesn't have `null` in its `next` pointer — instead, it points at the first node.

The methods you must write: You must add appropriate code to complete the `enter`, `leave`, and `isEmpty` methods in the `Queue` class. Note that, if the `Queue` is empty, `leave` should return `null`.

Using generics changes how you write your code only slightly. The `Queue` class declares that it is storing items of type `E`:

- If you declare a `Node` variable, you have to say that the `Node`s that the variable can point at store an item of type `E`. For example:

  ```
  Node<E> end;
  ```

- Similarly, if you create a new `Node` instance, you have to say that the new `Node` will store an item of type `E`. For example:

  ```
  end = new Node<E>( someItem, null ); // someItem must be of type E.
  ```

Note that the `Node` method is outside of the `Queue` class. Therefore, you will have to use the `Node` getters and setters to look at or modify a `Node`. For example, to change `next` pointer of `Node` `curr` to `null`:

```
curr.setNext( null );
```

Reminder: Each method should have only ONE `return` statement.