Programming Laboratory 9
CSci 1913: Introduction to Algorithms, Data Structures, and Program Development
November 6–7, 2018

0. Introduction.

Two queue classes were discussed in the lectures. One implemented a queue using a linear linked list of nodes. The other implemented a queue using a circular array. For this lab assignment, you must implement an iterator for the queue class that used a circular array.

1. Theory.

Suppose that we want to visit the elements stored in a sequence, like a stack or a queue. Also suppose that we are not allowed to modify the sequence to visit its elements. Then we can visit a sequence's elements by using an *iterator*. An iterator is class whose instances can visit the elements of a sequence. Each iterator typically has a method called `hasNext` that tests if there are more elements to be visited. It also has a method called `next` that returns the next element to be visited, and advances to the following element. An iterator that visits the elements of a linked stack was discussed in the lectures.

We can simplify an iterator's design by assuming that the sequence will not change while we visit its elements. For example, if we use an iterator to visit the elements of a stack, then we assume that the stack will not be `push`'ed or `pop`'ped. Similarly, if we use an iterator to visit the elements of a queue, then we assume that the queue will not be `enqueue`'d or `dequeue`'d. If a sequence changes while an iterator visits its elements, then the actions of the iterator become *undefined*—which means they don't have to work correctly if that happens.

2. Implementation.

You must add the following members to the class `ArrayQueue`, whose Java source code is available on Moodle. These members implement an iterator for `ArrayQueue`. You are not allowed to modify `ArrayQueue` except to add these additional members.

```
public class Iterator
```

This class must be nested inside `ArrayQueue`. An instance of this class may be used to visit the current elements of an instance of `ArrayQueue`. It must have one or more private variables that let it "know" which elements of `ArrayQueue` are to be visited next. You must decide what those private variables are.

```
private Iterator(...)
```

This is `Iterator`'s constructor. Of course it must be inside `Iterator`. It must set `Iterator`'s private variables to the values of its parameters. You must decide what these parameters are.

```
public boolean hasNext()
```

This method must be inside `Iterator`. It must return `true` if there are more elements of `ArrayQueue` that remain to be visited. It must return `false` otherwise. This method must use `Iterator`'s private variables only. Hint: use ideas from `ArrayQueue`'s method `isEmpty`.

```
public Base next()
```

This method must be inside `Iterator`. It must return the next `Base` element to be visited from `ArrayQueue`. If no more elements remain to be visited, then it must throw an `IllegalStateException`. This method must use `Iterator`'s private variables only. Hint: use ideas from `ArrayQueue`'s method `dequeue`.

```
public Iterator iterator()
```

This method must be inside `ArrayQueue`. It must call `Iterator`'s constructor to make a new instance of `Iterator`. It must then return the new instance.

Be careful to put these members in the right places. For example, the class `Iterator` must be nested inside the clas `ArrayQueue`, and the method `next` must be inside the class `Iterator`, etc. If these things are in the wrong places, then the iterator will not work.

This gives us a slightly different kind of iterator from the one discussed in the lecture, and also different from the ones provided by Java. It does not have a `remove` method, and it is not accessed using an interface.

The file **`tests.java`** on Moodle contains Java code that performs a series of tests. The tests call methods from the `ArrayQueue` class and your `Iterator` class. Some of them print what those methods return. Each test is also followed by a comment that tells how many points it is worth, and optionally what must be printed if it works correctly.

3. Deliverables.

Run the tests, then turn in the Java source code for the modified `ArrayQueue` class with your `Iterator` class in it. Your lab TA will tell you how and where to turn it in. If your lab is on **Tuesday, November 6, 2018,** then your work must be turned in by **11:55 PM** on **Tuesday, November 13, 2018.** If your lab is on **Wednesday, November 7, 2018,** then your work must be turned in by **11:55 PM** on **Wednesday, November 14, 2018.** *To avoid late penalties, do not confuse these dates!*