CS2110. Recitation on interfaces Iterator and Iterable Spring 2017

The main task in this recitation is for you to fix a version of A3, circularly linked lists, so that it implements interfaces Iterator and Iterable. But first a few problems for you to do.

1. Suppose you had to explain to a friend the difference between interfaces Iterator and Iterable. What would you say?

A. Write down in a sentence or two what interface Iterator is for.

Iterator over some collection provides methods that help you write a loop to enumerate and process the elements in that collection.

B. Write down in a sentence or two what interface Iterable is for.

Implementing this interface allows an object to be the target of foreach loop statement. Implementing this interface requires us to declare the method iterator.

2. Below is some code that is used to enumerate and print the Strings in some collection. To the right, write down what is wrong with this code:

```
Iterator<String> it=  …;
…
while (it.next()) {
   System.out.println(it.next())
}
```

The while loop condition is incorrect. Condition statement has to type boolean, but it.next() returns a String.

3. Below is some code that is used to enumerate and print the Strings in some collection. To the right, write down a requirement on the collection for this to actually work and also state when it doesn't work.

```
Iterator<String> it=  …;
…
while (it.hasNext()) {
   System.out.println(it.next());
   System.out.println(it.next());

}
```

This case might not work in a list consisting of odd number of elements, because it.next() returns the next element from from the last referenced element. Since it.next() is called twice within the while loop body, the condition it.hasNext() is checked in every other element. So, this would throw an NoSuchElementException for a list of odd number of elements. The only way to not run into an exception is to have even number of elements in the list.

4. This problem concerns implementing Iterator and Iterable in your solution to A3, class CLL. Do the following steps to prepare for this —you can use either your solution to A3, if you *know* it is correct, or our solution, which will be on the Lecture Notes page, on the row for this week's recitation.

A. Start a new Eclipse project, call it something like a3Iterable.

B. Put the correct files into the project:

   a. Download CLLIterable.zip from the lecture notes page and unzip it. Copy our CLLTest.java into the new project. You can do this by selecting the file, doing a copy, selecting the new project's src directory, and doing a paste.

   b. To use OUR CLL.java: Copy CLL.java from CLLIterable.zip, into your project as above.

   c. To use YOUR CLL.java: Copy your CLL.java from your old a3 project to the new one.

C. If JUnit4 is not available in the new project, Insert a new JUnit test-case class and then delete it in the usual way.

D. Look at the code at the bottom of class CLLTest. Look at how it tests interfaces Iterator and Iterable. Of course, there are errors because Iterator and Iterable have not yet been implemented by class CLL.

E. Write inner class CLLIterator with this specification and header:

   /** An instance is an iterator over the elements of this list. */
   private class CLLIterator **implements** Iterator

   (Hint: You'll need keep track of whether you have already returned the head or not to make sure the iterator doesn't accidentally go around the list twice.)

F. Write method iterator with this specification and header:

   /** Return an Iterator over the elements of this list. */
   public Iterator<E> iterator()

G. Change the class of declaration of CLL to implement Iterable<E>.

H. If you did the above correctly, both classes should have no syntax errors. And if you Run class CLLTest and made no mistakes in the above, the test should run without errors.