OOP and Data Structures	Discussion 5 — Iterators
CS2110 Summer 2019	Due Friday 7/12 at 5PM

Before starting this discussion, watch the Iterator tutorials on the Java HyperText.

1. Create a file called `written.txt` and answer the following questions:

   (a) Explain the difference between interface `Iterator<E>` and interface `Iterable<E>`

   (b) The following code is intended to enumerate and print the strings in a collection. Indicate what is wrong with the code:

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

   (c) The following code is intended to print the strings in a collection. This code will only work in certain situations. When will it work, and when will it fail?

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

2. In project 2, you implemented a doubly-linked list. Because your implementation extended `AbstractList`, it inherits a default implementation of the `iterator` method.

   This implementation is not very efficient for linked lists, because it is defined in terms of `get` which runs in linear (not constant) time. Repeated calls to `next` will call `get(0)`, `get(1)`, `get(2)`, etc. Later calls to `get` will repeatedly traverse from the beginning of the list.

   Here, you will override this method for your (or ours, if you prefer) `DLinkedList` class.

   - Download the file `DListIteratorTest.java` from CMS. Put it in your project for assignment 2. If you want to use our `DLinkedList` implementation, you can find the solution in CMS under the "Project 2" assignment.
   - There are two JUnit tests in `DListIteratorTest`. Run them. They should pass, because the inherited implementation of `List.iterator()` works. Run them to double check.
   - In class `DLinkedList`, add an `iterator` method that overrides `List.iterator` (the tests should now fail):

     ```
     1 /** @see java.util.List.iterator() */
     2 @Override
     3 public Iterator<E> iterator() {
     4   throw new NotImplementedError();
     5 }
     ```

   - Create a new inner class `DListIterator` inside of your `DLinkedList` class:

     ```
     1 /** An instance iterates over a DLinkedList */
     2 private class DListIterator implements Iterator<E> {
     3 }
     ```

     Ask eclipse to add the unimplemented methods, and make them throw `NotImplementedErrors`. Change the `iterator` method to create and return a new `DListIterator`.

   - Complete the implementation of `DListIterator`. An iterator represents a "position" in a list; for the `DLinkedList`, a reference to a `Node` is a good way to represent a position.

3. For an additional challenge (this is not required), you can change your `DListIterator` class so that it implements the ListIterator interface instead of just `Iterator`. If you do this, you should also override the `listIterator` method in your `DLinkedList` class.