CS 0445 Spring 2025

Assignment of Iterator

Introduction:

Recently in lecture we discussed iterators and the Iterator<T> interface. An example was discussed in which an iterator was added to the author's LList<T> class, resulting in the LinkedListWithIterator<T> class. Before completing this assignment, review Lecture 18 and carefully look over the code in the author's LinkedListWithIterator<T> class and also in the MyArrayIterable<T> class.

In Assignment of Simple Deque you implemented a primitive deque that satisfied the author's DequeInterface<T>. In this assignment you will take that implementation and add an iterator to it, so that a user will be able to iterate over all of the values of the deque.

Consider the following interface:

```
import java.util.*;
public interface DequeWithIteratorInterface<T>
{
   /** Adds a new entry to the front/back of this deque.
       @param newEntry An object to be added. */
   public void addToFront(T newEntry);
   public void addToBack(T newEntry);

   /** Removes and returns the front/back entry of this deque .
       @return  The object at the front/back of the deque.
       @return  null if the deque is empty before the
                operation . */
   public T removeFront();
   public T removeBack();

   /** Retrieves the front/back entry of this deque .
       @return  The object at the front/back of the deque.
       @return  null if the deque is empty. */
   public T getFront();
   public T getBack();

   /** Detects whether this deque is empty.
       @return  True if the deque is empty, or false otherwise. */
   public boolean isEmpty();

   /*  Removes all entries from this deque. */
   public void clear();

   /*  Create and return a new Iterator<T> over the contents of
       this deque
   public Iterator<T> iterator();
}
```

This interface is the same interface that you saw in Assignment of Simple Deque, with the addition of the iterator() method. You can obtain this interface in file DequeWithIteratorInterface.java.

In this assignment you will add the iterator() method to your SimpleDeque<T> class to yield the SimpleDequeWithIterator<T> class. SimpleDequeWithIterator<T> will implement the DequeWithIteratorInterface<T>, as shown above.

Note: In your iterator you only need to implement the hasNext() and next() methods – don't worry about remove(). The remove() method is declared as "default" within the Iterator<T> interface, which allows the programmer to ignore it while still satisfying the interface.

Recall that the deque ADT organizes data logically from front to back and your iterator should proceed through the data in this way. Thus, your iterator object should be initialized at the "front" of the deque and should proceed to the "back" of the deque with each call to the next() method. As discussed in lecture, the iterator should be an inner class so that it can access the underlying data in the deque object. For an example of an iterator on an array-based data structure, see the MyArrayIterable.java handout.

Once you have implemented your SimpleDequeWithIterator<T> class, you can test it with the main program TestIterator.java. The TestIterator.java program should compile and run with your SimpleDequeWithIterator<T> class, and the output should match that shown in file IteratorOut.txt. Read the program carefully so that you see how your class is being utilized in the program.