Doubly Circular LinkedList

Doubly LinkedList

Circular Doubly Linked List has properties of both doubly linked list and circular linked list in which two consecutive elements are linked or connected by previous and next pointer and the last node points to first node by next pointer and also the first node points to last node by previous pointer.

```
+---------+    +---------+    +---------+    +---------+    +---------+
|         |--->|         |--->|         |--->|         |--->|         |
| Dummy   |    | node 0  |    | node 1  |    | node 2  |    |  Dummy  |
|         |<---|         |<---|         |<---|         |<---|         |
+---------+    +---------+    +---------+    +---------+    +---------+
```

Dummy node initial state:

```
Node<T> dummy = new Node();
dummy.prev = dummy;
dummy.next = dummy;
dummy.data = (T) dummy;     // unsafe cast
```

Exercise

How to implement `clear()` method?

Why do we need a dummy node?

How we could safely iterate through the list with dummy node?

Concerning `wellformed` method in ADT

1. dummy node is not null. Its data should be itself, cast (unsafely) `data = (T)this;`
2. each link must be correctly double linked.
3. size is number of nodes in list, other than the dummy.
4. the list must cycle back to the dummy node.

Concerning `wellformed` method in Iterator

0. The outer invariant holds, and versions match
1. precursor is never `null`
2. precursor is in the list
3. if precursor is before the dummy, hasCurrent must be `false`

Structure

We have to manage `_previous` pointer

```
public class DoublyLinkedList<T> {
    Node _head, _tail;

    private static class Node {
        T _data;
        Node _previous, _next;

        Node(T p, Node previous, Node next) {
            _data = data;
            _previous = previous;
            _next = next;
        }
    }
}
```

Exercise: `add(T value)`

```java
public void add(T value) {
    // TODO: add the node at the end of list given only dummy node
    //
    //
}
```

Solution: `add(T value)`

```java
public void add(T value) {
    dummy.prev = dummy.prev.next = new Node<E>(x, dummy.prev, dummy);
}
```

Exercise: `remove(T value)`

```
// Return false if remove failed!
public void remove() {
    // TODO: assume we only have precursor variable and we want to remove cursor
    //
    //
}
```

Solution: `remove(T value)`

```java
// Return false if remove failed!
public boolean remove(T data) {
    precursor.next.next.prev = precursor;
    precursor.next = precursor.next.next;
}
```

Selection sort

```
function select(list[1..n], k)
    for i from 1 to k
        minIndex = i
        minValue = list[i]
        for j from i+1 to n
            if list[j] < minValue
                minIndex = j
                minValue = list[j]
                swap list[i] and list[minIndex]
    return list[k]
```

Lab assignment #6: