Lab 4 (List ADT)

CSC 172 (Data Structures and Algorithms)
Fall 2022
University of Rochester
Due Date: Monday, October 3 @ 8:00 am

Introduction

The labs in CSC172 will follow a pair programming paradigm. Every student is encouraged (but not strictly required) to have a lab partner. Every student must hand in their own work but also list the name of their lab partner if any on all labs.

In this lab, you will work on List ADT (Abstract data type). You are **required** to implement the `URLinkedList` class. You are **not** required to implement the `URArrayList` class, but you may do so for extra credit. We provide you the following URList interface and all you have to do is create two classes `URArrayList` and `URLinkedList` those implement `URList` interface. Moreover, both `URArrayList` and `URLinkedList` must implement additional functionality as described below.

```
import java.util.Collection;
import java.util.Iterator;

// URList class ADT. Generalize the element type using Java Generics.
public interface URList<E> extends Iterable<E>{ // URList class ADT

// Appends the specified element to the end of this list
boolean add(E e);

// Inserts the specified element at the specified position in this list
void add(int index, E element);

// Appends all of the elements in the specified collection to the end of this list,
// in the order that they are returned by the specified collection's iterator
boolean   addAll(Collection<? extends E> c);

// Inserts all of the elements in the specified collection into this list
// at the specified position
boolean addAll(int index, Collection<? extends E> c);

// Removes all of the elements from this list
void clear();

// Returns true if this list contains the specified element.
boolean   contains(Object o);

// Returns true if this list contains all of the elements of the specified collection
boolean   containsAll(Collection<?> c);

// Compares the specified object with this list for equality.
// Returns true if both contain the same elements. Ignore capacity
boolean   equals(Object o);
// Returns the element at the specified position in this list.
E get(int index);

// Returns the index of the first occurrence of the specified element in this list,
// or -1 if this list does not contain the element.
int indexOf(Object o);

// Returns true if this list contains no elements.
boolean   isEmpty();

// Returns an iterator over the elements in this list in proper sequence.
Iterator<E> iterator();

// Removes the element at the specified position in this list
E remove(int index);

// Removes the first occurrence of the specified element from this list,
// if it is present
boolean   remove(Object o);

// Removes from this list all of its elements that are contained
//  in the specified collection
boolean   removeAll(Collection<?> c);

// Replaces the element at the specified position in this list
// with the specified element
E set(int index, E element);

// Returns the number of elements in this list.
int size();

// Returns a view of the portion of this list
// between the specified fromIndex, inclusive, and toIndex, exclusive.
URList<E> subList(int fromIndex, int toIndex);


// Returns an array containing all of the elements in this list
//  in proper sequence (from first to the last element).
Object[] toArray();


}
```

Methods specific for only `URArrayList` class:

```
// Increases the capacity of this ArrayList instance, if necessary,
// to ensure that it can hold at least the number of elements specified
// by the minimum capacity argument.
void ensureCapacity(int minCapacity)

// Returns the current capacity of the list
int getCapacity()
```

Methods specific for only `URLinkedList` class:

```
// Inserts the specified element at the beginning of this list.
void addFirst(E e)

// Appends the specified element to the end of this list.
void addLast(E e)

// Retrieves, but does not remove, the first element of this list, or returns null if
     this list is empty.
E peekFirst()

// Retrieves, but does not remove, the last element of this list, or returns null if
    this list is empty.
E peekLast()

// Retrieves and removes the first element of this list, or returns null if this list
     is empty.
E pollFirst()

// Retrieves and removes the last element of this list, or returns null if this list
    is empty.
E pollLast()
```

You must use the following `URNode` objects for your `URLinkedList` implementation.

```
class URNode<E> { // Doubly linked list node

private E e; // Value for this node
private URNode<E> n; // Reference to next node in list
private URNode<E> p; // Reference to previous node

// Constructors
URNode(E it, URNode<E> inp, URNode<E> inn) { e = it; p = inp; n = inn; }
URNode(URNode<E> inp, URNode<E> inn) { p = inp; n = inn; }

// Get and set methods for the data members
public E element() { return e; } // Return the value
public E setElement(E it) { return e = it; } // Set element value
public URNode<E> next() { return n; } // Return next link
public URNode<E> setNext(URNode<E> nextval) { return n = nextval; } // Set next link
public URNode<E> prev() { return p; } // Return prev link
public URNode<E> setPrev(URNode<E> prevval) { return p = prevval; } // Set prev link

}
```

Before you start

You can download URList.java and URNode.java from:
`http://www.cs.rochester.edu/courses/172/spring2018/labs/URList.java` and
`http://www.cs.rochester.edu/courses/172/spring2018/labs/URNode.java` respectively.

Note:

- You MUST use but NOT modify `URList.java` and `URNode.java` file.
- You MUST NOT use java.util.ArrayList and java.util.LinkedList.
- You MUST implement `Iterable<T>` interface for the classes you implement.
- Please write any assumption made. For example: describe what the return value means.
- Your methods must handle all the corner cases gracefully — for example, throwing exceptions with detailed explanations or returning values indicating the error in case the operation is not permitted. The comments should clearly state the issues and the remedies involved. In short, no illegal operation should be permitted and the list and all its parameters should be in a valid state.

Submission

Submit a single zip file `Lab4.zip` containing `URLinkedList.java`, `URArrayList.java` (optional extra credit), a unit test file containing a main method, a README file, and your class files. Your README should describe how to run your unit tests. Upload this file at the appropriate location on the Blackboard system at `learn.rochester.edu`.

1. Two Java source code files `URLinkedList.java` and `URArrayList.java` (optional extra credit), representing the work accomplished. All source code files should contain author and partner identification in the comments at the top of the file.
2. A plain text file named README that includes your contact information, your partner's name, a brief explanation of the lab (a one-paragraph synopsis. Include information identifying what class and lab number your files represent.), and one sentence explaining the contents of any other files you hand in. You can also share testcases those you ran.

Grading

Total: 100 pts

- 10 pts for README file, proper commenting, and error handling.
- (19 * 3 =) 57 pts for 19 common methods for URArrayList and URLinkedList.
- (6 * 3 =) 18 pts for the siz `URLinkedList.java` methods.
- 15 pts for your main method containing your unit testing class (main method, tests, etc)
- 20 pts Extra Credit for Implementing `URArrayList.java`