UNIVERSITY OF VIRGINIA, DEPARTMENT OF COMPUTER SCIENCE

Linked Lists - Linked List Implementation

Dr. Mark R. Floryan

August 16, 2019

1 SUMMARY

For this homework, you will be writing a generic doubly-linked list data structure in Java. Your summary:

1. Download the starter code and import the project into Eclipse
2. Implement the LinkedList.java class
3. Verify your implementation using the provided tester class
4. **FILES TO DOWNLOAD:** linkedLists.zip
5. **FILES TO SUBMIT:** LinkedList.zip

1.1 LISTITERATOR.JAVA

Once you've imported the provided code into Eclipse, you may want to start on ListIterator.java (though you don't have to). A ListIterator is an object that points to one element in your linked list, and provides methods to grab the element at that index, move the iterator forward one position, backward one position, etc. The provided tester uses this iterator class

to test your code, and so it much be implemented correctly. The methods you will be asked to implement are:

```
    /**
     * These two methods tell us if the iterator has run off
     * the list on either side
     */
    public boolean isPastEnd();
    public boolean isPastBeginning();

    /**
     * Get the data at the current iterator position
     */
    public T value();

    /**
     * These two methods move the cursor of the iterator
     * forward / backward one position
     */
    public void moveForward();
    public void moveBackward();
```

1.2 LinkedList.java

Next, implement all of the methods in LinkedList.java. This class will implement the provided List interface, which is duplicated for your convenience here. Your task is to implement each of these methods.

```
public interface List<T> {

    /**
     * Returns the size of this list, i.e., the number
     * of nodes currently between the head and tail
     * @return
     */
    public int size();

    /**
     * Clears out the entire list
     */
    public void clear() ;

    /**
```

```
16          * Inserts new data at the end of the
            * list (i.e., just before the dummy tail node)
18          * @param data
            */
20      public void insertAtTail(T data);

22      /**
            * Inserts data at the front of the
24          * list (i.e., just after the dummy head node
            * @param data
26          */
        public void insertAtHead(T data);
28
        /**
30          * Inserts node such that index becomes the
            * position of the newly inserted data
32          * @param data
            * @param index
34          */
        public void insertAt(int index, T data);
36
        public T removeAtTail();
38
        public T removeAtHead();
40
        /**
42          * Returns index of first occurrence of
            * the data in the list, or -1 if not present
44          * @param data
            * @return
46          */
        public int find(T data);
48
        /**
50          * Returns the data at the given index, null if
            * anything goes wrong (index out of bounds, empty list, etc.)
52          * @param index
            * @return
54          */
        public T get(int index);
56  }
```

In addition, there are a few Linked List specific methods that you need to implement. They

are all of the methods that involve a ListIterator in some way. The are enumerated below:

```
/**
 * Inserts data after the node pointed to by iterator
 */
public void insert(ListIterator<T> it, T data);

/**
 * Remove based on Iterator position
 * Sets the iterator to the node AFTER the one removed
 */
public T remove(ListIterator<T> it);

/* Return iterators at front and end of list */
public ListIterator<T> front();
public ListIterator<T> back();
```

1.3 Testing Your Implementation

Once you implement this methods, you can test them by running the main method in the provider ListTester.java class. This simple tester will execute the methods in your list and compare them to those in Java's built in LinkedList to make sure the results are as expected.

One strategy might be to implement your linked list methods in the order that they are tested. That way, you can see the tester say "this method is correct" before moving on to the next one.

Notice that we expect your Linked List to be a generic class, meaning any type of Object can be stored within your Linked List.