CMPT 280

Topic 4: Cursors

Mark G. Eramian

University of Saskatchewan

References

- Textbook, Chapter 4

Cursor Interface

Do you remember what the methods in the cursor interface do?

- `itemExists`
- `item`
- `goFirst`
- `goForth`
- `goLast`
- `goBefore`
- `goAfter`
- `before`
- `after`

Exercise 1

- Write a java interface for the cursor methods (don't worry about the javadoc for now).

```
1  public interface Cursor<I> {
2
3      /**
4       * Checks whether the cursor is positioned at an
5       * element in the collection.
6       *
7       * @return true if the cursor is positioned at an
8       *         element, false otherwise.
9       */
10     boolean itemExists();
11
12     ...
13 }
```

Exercise 2

Write the methods that we need for iterating all of the elements in a linked list. We need:

- `itemExists`
- `item`
- `goFirst`
- `goForth`

Do we need any additional instance variables to support these methods?

Note: While we don't, strictly speaking, need to, the implementation of `goForth` is a lot easier if you've already implemented `after` and `before`.

Additional Pracatice

On your own, implement the remaining cursor methods. You can find the method stubs in the `LinkedList.java` file in the Lecture03 exercise solutions.

- `goLast`
- `goBefore`
- `goAfter`

You could also practice regression testing by writing the regression tests for all of the cursor methods.

Exercise 3

- Using the `LinkedList` class, starting with an empty list, store 5 random numbers in the list, and then iterate over the list, and print out each number.
- Hint: `Math.random()` returns a random `Double` between 0.0 and 1.0.

Exercise 4

- How would we record a cursor position for an `ArrayedList`?
- How could we represent the "before" and "after" positions?
- Add the required instance variables to `ArrayedList`.

Exercise 5

- Implement the cursor methods for `ArrayedList`
- Additional Practice: On your own, write regression tests for the cursor methods of `ArrayedList`.

Next Class

- We'll be exploring an augmented linked list data structure: the doubly-linked list. This will allow us to build a much more feature-rich and robust list object.
- Next class reading: Chapter 5: Doubly Linked Lists.