CPSC 221

Single-linked List

"Strange how paranoia can link up with reality now and then." --Philip K. Dick

Objectives

- Create a singly-linked node implementation of the `IndexedUnsortedList` interface called `IUSingleLinkedList`.
- Create a fully functional `Iterator` for your `IUSingleLinkedList`.
- Expand your test suite to ensure correct functionality of your `IUSingleLinkedList` and `Iterator`.

Tasks

For this lab, you will:

1. Complete your test plan.
    - Implement the remaining change scenarios from your test plan (16, 23, 27, 29, 30, 33, 37, 39). In total, you should have 25 scenarios altogether from your original test plan.
    - Run tests frequently against `GoodList` and `IUArrayList` during development to be sure your scenarios and list classes are working properly.
2. Implement a generic typed list class called `IUSingleLinkedList`.
    - Your `IUSingleLinkedList` class should use a chain of singly-linked nodes to manage list elements and implement all of the methods required by the `IndexedUnsortedList` interface.
    - Use the provided `LinearNode` class in your implementation.
    - Be sure to uncomment the lines in the `newList()` method of `ListTester`, so the class can create `IUSingleLinkedList` objects.
    - Run tests frequently against `IUSingleLinkedList` during development to be sure your list class is working properly.
3. Implement a fully functional `Iterator` for your `IUSingleLinkedList` class.
    - Your iterator should implement the Java `Iterator` interface.

    - `Iterator` only requires you to implement two of its methods, `hasNext()` and `next()`, but you will also implement the optional `remove()` method.
    - The `Iterator` documentation says the result of calling its methods after a change has occurred to the list is undefined, because fail-fast behavior cannot be *guaranteed* for all `Iterator`s. *Your* implementation, however, will be expected to throw a *`ConcurrentModificationException`* if any `Iterator` method is called after the list has been modified by any source other than the current `Iterator`.
4. Include change scenarios resulting from `Iterator`'s `remove()` method in your test suite.
    - See scenarios 44-54 in this expanded set of change scenarios: IteratorTestScenarios.txt.
    - The test cases for the scenarios are the same as the ones you have used universally for empty, 1-element and 2-element list end states.
    - Uncomment the `test_IterConcurrency()` method call in the `runTests()` method.
    - Run tests frequently against `IUSingleLinkedList` during development to be sure your change scenarios and single linked list class are working properly.
5. Add the test cases involving the `Iterator` methods
    - Uncomment the iterator test cases in the empty list and single element list test suites.
    - Using the test cases in the previous test suites as an example, add the test cases listed in IteratorTestScenarios.txt for the two and three element list test suites.

Files

Use the following class to support creation of your `IUSingleLinkedList`:

- `LinearNode.java`

The starter `IUSingleLinkedList.java` includes the basic class structure and a sample method.

`ListTester` is the same test class you've been developing through previous assignments (List Test Part 1, List Test Part 2, and ArrayList) for testing implementations of `IndexedUnsortedList`

Grading

Points will be awarded according to the following breakdown:

Tasks	Points
`ListTester` testing all change scenarios from your test plan	20
`IUSingleLinkedList` class and `Iterator` functionality and quality	20

Required Files

Submit the following files:

- `IUSingleLinkedList.java`
- `ListTester.java` with all tests from your test plan, plus `Iterator` tests cases and change scenarios.