CPSC 221

Double-linked List

"The way to succeed is to double your error rate."
--Thomas J. Watson

Objectives

- Create a doubly-linked list implementation of the `IndexedUnsortedList` interface
- Create a fully functional list iterator that implements the `ListIterator` interface
- Use a test suite to ensure correct functionality of a class and add scenarios and test cases focusing on the `ListIterator`.

Tasks

For this programming assignment, you will:

- Create a class called `IUDoubleLinkedList` that implements the `IndexedUnsortedList` interface. The underlying data structure of your class should use a node-based, doubly-linked list. A `BidirectionalNode` class has been provided for you to use and you should use the `ListTester` to confirm your class is working as expected; be sure to uncomment the `doubleLinkedList` case in the `newList()` method and set the `LIST_TO_USE` appropriately.
- Implement a list iterator within your class that supports all `ListIterator` methods, including the `add()`, `remove()`, and `set()` methods.
- Update your `ListTester` class to include `ListIterator` testing:
  - Implement the following change scenarios resulting from modifications to the list from `ListIterator`'s `add()`, `remove()`, and `set()` methods: 57, 60, 66, 71, 87, 90, 96, 99. Ensure when you place these scenarios in the `runTests()` method that you place them within the `SUPPORTS_LIST_ITERATOR` if-block.
  - Implement the test cases for all of the `ListIterator` methods within the appropriate test suites. Code is supplied for the tests for empty and single-element lists which you must be uncommented

in the `SUPPORTS_LIST_ITERATOR` if-block as well as the test case expectations for two- and three-element lists given to you below.

ListIterator Change Scenarios for IndexedUnsortedList

Each scenario listed below is described as a starting state, a change to the list, and the resulting state.
Example list iterator tests are shown for some scenarios.

```
55) [A] -> list-iterator(0),next(),remove() -> []
56) [A,B] -> list-iterator(0),next(),remove() -> [B]
57) [A,B] -> list-iterator(1),next(),remove() -> [A]
  ListIterator Tests:
      listIterator() throws no Exception
      listIterator(), nextIndex() returns 0
      listIterator(), previousIndex() returns -1
      listIterator(-1) throws IndexOutOfBoundsException
      listIterator(0) throws no Exception
      listIterator(1) throws no Exception
      listIterator(2) throws IndexOutOfBoundsException
      listIterator(0), hasPrevious() returns false
      listIterator(1), hasPrevious() returns true
      listIterator(0), hasNext() returns true
      listIterator(1), hasNext() returns false
      listIterator(0), previous() throws NoSuchElementException
      listIterator(1), previous() returns A
      listIterator(0), next() returns A
      listIterator(1), next() throws NoSuchElementException
      listIterator(0), add(X) throws no Exception
      listIterator(1), add(X) throws no Exception
      listIterator(0), nextIndex() returns 0
      listIterator(1), nextIndex() returns 1
      listIterator(0), previousIndex() returns -1
      listIterator(1), previousIndex() returns 0
      listIterator(0), set(X) throws IllegalStateException
      listIterator(0), remove() throws IllegalStateException
      listIterator(0), next(), set(X) throws no Exception
      listIterator(0), next(), remove() throws no Exception
      listIterator(1), previous(), set(X) throws no Exception
      listIterator(1), previous(), remove() throws no Exception
58) [A,B] -> list-iterator(0),next(),remove(),next(),remove() -> []
59) [A,B,C] -> list-iterator(0),next(),remove() -> [B,C]
60) [A,B,C] -> list-iterator(1),next(),remove() -> [A,C]
  ListIterator Tests:
      listIterator() throws no Exception
      listIterator(), nextIndex() returns 0
      listIterator(), previousIndex() returns -1
      listIterator(-1) throws IndexOutOfBoundsException
      listIterator(0) throws no Exception
      listIterator(1) throws no Exception
      listIterator(2) throws no Exception
      listIterator(3) throws IndexOutOfBoundsException
      listIterator(0), hasPrevious() returns false
      listIterator(1), hasPrevious() returns true
      listIterator(2), hasPrevious() returns true
      listIterator(0), hasNext() returns true
      listIterator(1), hasNext() returns true
      listIterator(2), hasNext() returns false
      listIterator(0), previous() throws NoSuchElementException
      listIterator(1), previous() returns A
      listIterator(2), previous() returns C
      listIterator(0), next() returns A
      listIterator(1), next() returns C
      listIterator(2), next() throws NoSuchElementException
      listIterator(0), add(X) throws no Exception
      listIterator(1), add(X) throws no Exception
      listIterator(2), add(X) throws no Exception
      listIterator(0), nextIndex() returns 0
      listIterator(1), nextIndex() returns 1
      listIterator(2), nextIndex() returns 2
      listIterator(0), previousIndex() returns -1
      listIterator(1), previousIndex() returns 0
      listIterator(2), previousIndex() returns 1
      listIterator(0), set(X) throws IllegalStateException
      listIterator(0), remove() throws IllegalStateException
      listIterator(0), next(), set(X) throws no Exception
      listIterator(1), next(), set(X) throws no Exception
      listIterator(0), next(), remove() throws no Exception
      listIterator(1), next(), remove() throws no Exception
      listIterator(1), previous(), set(X) throws no Exception
      listIterator(2), previous(), set(X) throws no Exception
      listIterator(1), previous(), remove() throws no Exception
      listIterator(2), previous(), remove() throws no Exception
61) [A,B,C] -> list-iterator(2),next(),remove() -> [A,B]
62) [A,B,C] -> list-iterator(0),next(),remove(),next(),remove() -> [C]
63) [A,B,C] -> list-iterator(0),next(),remove(),next(),next(),remove() -> [B]
64) [A,B,C] -> list-iterator(1),next(),remove(),next(),remove() -> [A]
65) [A,B,C] ->
   list-iterator(0),next(),remove(),next(),remove(),next(),remove() -> []
66) [A] -> list-iterator(1),previous(),remove() -> []
  ListIterator Tests:
      listIterator() throws no Exception
      listIterator(), nextIndex() returns 0
      listIterator(), previousIndex() returns -1
      listIterator(-1) throws IndexOutOfBoundsException
      listIterator(0) throws no Exception
      listIterator(1) throws IndexOutOfBoundsException
      listIterator(0), hasPrevious() returns false
      listIterator(0), hasNext() returns false
      listIterator(0), previous() throws NoSuchElementException
      listIterator(0), next() throws NoSuchElementException
      listIterator(0), add(X) throws no Exception
      listIterator(0), nextIndex() returns 0
      listIterator(0), previousIndex() returns -1
      listIterator(0), set(X) throws IllegalStateException
      listIterator(0), remove() throws IllegalStateException
67) [A,B] -> list-iterator(1),previous(),remove() -> [B]
68) [A,B] -> list-iterator(2), previous(),remove() -> [A]
69) [A,B] -> list-iterator(2),previous(),remove(),previous(),remove() -> []
70) [A,B,C] -> list-iterator(1),previous(),remove() -> [B,C]
71) [A,B,C] -> list-iterator(2),previous(),remove() -> [A,C]
72) [A,B,C] -> list-iterator(3),previous(),remove() -> [A,B]
73) [A,B,C] ->
   list-iterator(2),previous(),remove(),previous(),remove() -> [C]
74) [A,B,C] ->
   list-iterator(3),previous(),remove(),previous(),previous(),remove() -> [B]
75) [A,B,C] ->
   list-iterator(3),previous(),remove(),previous(),remove() -> [A]
76) [A,B,C] ->
   list-iterator(3),previous(),remove(),previous(),remove(),
   previous(),remove() -> []
77) [] -> list-iterator(0),add(A) -> [A]
78) [A] -> list-iterator(0),add(B) -> [B,A]
79) [A] -> list-iterator(1),add(B) -> [A,B]
80) [A] -> list-iterator(0),next(),add(B) -> [A,B]
81) [A] -> list-iterator(1),previous(),add(B) -> [B,A]
82) [A] -> list-iterator(0),next(),set(B) -> [B]
83) [A] -> list-iterator(1),previous(),set(B) -> [B]
84) [A,B] -> list-iterator(0),add(C) -> [C,A,B]
85) [A,B] -> list-iterator(1),add(C) -> [A,C,B]
86) [A,B] -> list-iterator(2),add(C) -> [A,B,C]
87) [A,B] -> list-iterator(0),next(),add(C) -> [A,C,B]
  ListIterator Tests:
      listIterator() throws no Exception
      listIterator(), nextIndex() returns 0
      listIterator(), previousIndex() returns -1
      listIterator(-1) throws IndexOutOfBoundsException
      listIterator(0) throws no Exception
      listIterator(1) throws no Exception
      listIterator(2) throws no Exception
      listIterator(3) throws no Exception
      listIterator(4) throws IndexOutOfBoundsException
      listIterator(0), hasPrevious() returns false
      listIterator(1), hasPrevious() returns true
      listIterator(2), hasPrevious() returns true
      listIterator(3), hasPrevious() returns true
      listIterator(0), hasNext() returns true
      listIterator(1), hasNext() returns true
      listIterator(2), hasNext() returns true
      listIterator(3), hasNext() returns false
      listIterator(0), previous() throws NoSuchElementException
      listIterator(1), previous() returns A
      listIterator(2), previous() returns C
      listIterator(3), previous() returns B
      listIterator(0), next() returns A
      listIterator(1), next() returns C
      listIterator(2), next() returns B
      listIterator(3), next() throws NoSuchElementException
      listIterator(0), add(X) throws no Exception
      listIterator(1), add(X) throws no Exception
      listIterator(2), add(X) throws no Exception
      listIterator(3), add(X) throws no Exception
      listIterator(0), nextIndex() returns 0
      listIterator(1), nextIndex() returns 1
      listIterator(2), nextIndex() returns 2
      listIterator(3), nextIndex() returns 3
      listIterator(0), previousIndex() returns -1
      listIterator(1), previousIndex() returns 0
      listIterator(2), previousIndex() returns 1
      listIterator(3), previousIndex() returns 2
      listIterator(0), set(X) throws IllegalStateException
      listIterator(0), remove() throws IllegalStateException
      listIterator(0), next(), set(X) throws no Exception
      listIterator(1), next(), set(X) throws no Exception
      listIterator(2), next(), set(X) throws no Exception
      listIterator(0), next(), remove() throws no Exception
      listIterator(1), next(), remove() throws no Exception
      listIterator(2), next(), remove() throws no Exception
      listIterator(1), previous(), set(X) throws no Exception
      listIterator(2), previous(), set(X) throws no Exception
      listIterator(3), previous(), set(X) throws no Exception
      listIterator(1), previous(), remove() throws no Exception
      listIterator(2), previous(), remove() throws no Exception
      listIterator(3), previous(), remove() throws no Exception
88) [A,B] -> list-iterator(1),next(),add(C) -> [A,B,C]
89) [A,B] -> list-iterator(1),previous(),add(C) -> [C,A,B]
90) [A,B] -> list-iterator(2),previous(),add(C) -> [A,C,B]
91) [A,B] -> list-iterator(0),next(),set(C) -> [C,B]
92) [A,B] -> list-iterator(1),next(),set(C) -> [A,C]
93) [A,B] -> list-iterator(1),previous(),set(C) -> [C,B]
94) [A,B] -> list-iterator(2),previous(),set(C) -> [A,C]
95) [A,B,C] -> list-iterator(0),next(),set(D) -> [D,B,C]
96) [A,B,C] -> list-iterator(1),next(),set(D) -> [A,D,C]
97) [A,B,C] -> list-iterator(2),next(),set(D) -> [A,B,D]
98) [A,B,C] -> list-iterator(1),previous(),set(D) -> [D,B,C]
99) [A,B,C] -> list-iterator(2),previous(),set(D) -> [A,D,C]
100) [A,B,C] -> list-iterator(3),previous(),set(D) -> [A,B,D]
```

Files

This project builds on the `ListTester` begun in previous assignments and uses the same IndexedUnsortedList.java.

Grading

Points will be awarded according to the following breakdown:

Tasks	**Points**
ListTester - complete, including `ListIterator` functionality tests	20
`IUDoubleLinkedList` and `ListIterator` Functionality	50
Quality - code formatting, naming conventions, encapsulation, etc.	10

Required Files

Please submit the following files:

- `IUDoubleLinkedList.java`
- `ListTester.java (updated)`