Programming Assignment 3 - DupLists

Introduction:

In this assignment you will extend the OptimisticList, LazyList, and LockFreeList list algorithms described in pages 205, 208, and 213 of the book so that they represent multisets, which are lists that can contain duplicates.

Your list implementations will follow the following specification:

1. add(T item) should increase the number of instances n of item in the list by one, returning True.
2. remove(T item) should return False if n == 0. Otherwise (for n >= 1) decrease the number of instances of item in the list by one and return True.
3. contains(T item) should return True when there are 1 or more instances of item in the list. It should return False otherwise.
4. You are not required to be able to count the number of instances of item in the list. You can support this feature if you really want to but the TAs won’t give you any extra points.

Setup:

The stencil code for this assignment is located at the course's pub directory at /course/cs1760/pub/duplists/ and can be installed with:

```bash
cs1760_install duplists
```

Once you copy the stencil code into your directory, you'll be ready to go! The files you need to modify are:

- LazyDupList.java
- LockFreeDupList.java
- OptimisticDupList.java
- ListTester.java

Since the point of this assignment is to introduce you to the sort of design patterns that arise when creating lock/wait-free implementations of an object’s methods, the stencil code that we give you already consists of interface definitions. You are expected to provide the method definitions in the various List classes that implement the interfaces. Additionally, you are expected to complete the tester class so that you can test your program.
Assignment Tips:

Here are some tips to get you started on the assignment:

1. Most of the code you will need for this assignment will be from the book, so feel free to use it!
2. You must use sentinels in your implementations of these lists. This is required to pass the minimal functionality tests and not including them will probably cause the TA test suite to break as well.

Compiling and Running Your Code:

To compile your code, simply run:

    make

in your command line.

To run and test your code, run:

    make runX

where “x” is replaced with the number of threads you wish to run tests on (the options are 1, 10, 100, 1000).

Testing:

Note that a portion of this assignment’s total score is reserved for testing. Passing basic functionality tests, located in BasicTestLists.java and BasicListTester.java, will reward you with some points; however, we also expect you to write any additional tests you deem necessary.

Failing basic functionality tests does not necessarily imply that you will receive no credit for the assignment. At the same time, the course staff will be unable to conduct a rigorous inspection of non-functional code to award partial credit.

Handin:

When you have completed DupLists, please hand in your assignment on Gradescope.