Homework 4: Doubly Linked List *Due: Wednesday, February 12, 8:00 PM*

READ THIS :)

This represents a huge simplification of this homework. Here are all the changes. Several menu options are no longer included. See below - they are all marked out with a line through them. Also, I added a comment to menu option 7. The comment is in italics. Due to the changes to the menu, I also marked out any output in the example run that are now meaningless.

Important

There are general homework guidelines you must always follow. If you fail to follow any of the following guidelines you risk receiving a **0** for the entire assignment.

1. All submitted code must compile under **JDK 7**. This includes unused code, so don't submit extra files that don't compile. (Java is backwards compatabile so if it compiles under JDK 6 it should compile under JDK 7.)
2. Do not include any package declarations in your classes.
3. Do not change any existing class headers, constructors, or method signatures.
4. Do not add additional public methods when implementing an interface.
5. Do not use anything that would trivialize the assignment. (e.g. don't import/use `java.util.LinkedList` for a Linked List assignment. Ask if you are unsure.)
6. You must submit your source code, the `.java` files, not the compiled `.class` files.
7. After you submit your files redownload them and run them to make sure they are what you intended to submit. You are responsible if you submit the wrong files.

Doubly Linked List

You are to code a doubly linked list with a head and tail reference and several public instance methods. A doubly linked list is similar to a singly linked list except each node in the list has a next reference as well as a previous reference. The purpose of having the previous references is to be able to traverse the list forward and backward, as well as make deleting a specific node easier.

Iterator

Your linked list will also implement Iterable and will need to make a private inner class to make this happen. For more information about this please read through the Java API.

Driver

You will need to implement a Driver class using the main method to create an application to test your List and Iterator. ~~For this driver, you will have a hypothetical 'cursor' that will start at the head and then shift according to the user's input. You should do input checking for edge cases such as a call to next when there is no next element.~~ Your Driver should include the following options:

1. Add - Add a node to the list, front or back based on user input
2. Remove - Remove a node from the list, front or back based on the user input
3. ~~Current - Print the data of the current element in the list~~
4. ~~Next - Print the data of the next node in the list and move the cursor forward~~

5. ~~Previous - Print the data of the previous node in the list and move the cursor backwards~~
6. Reverse - Reverse the list
7. List - prints the contents of the list. This command must use a for-each loop. ~~Notice that using a for-each loop will cause multiple instances of your iterator to be created. Your implementation should be able to handle this.~~ *Note: For a for-each loop to work with this structure, your iterator setup must be working properly. It should just work like magic if the private inner class implementing java.util.Iterator is implemented correctly. Also your outer class must implement java.lang.Iterable. Please see the LinkedList.java file under T-Square Resources for hints - although that file does not use generics (but you must) it is helpful to at least get started. Your code for the "List" menu option will not literally instantiate the iterator. Again look at that example code in Resources-WEEK05-LinkedList.java.*
8. *Exit - quits the program.*

This is an example of what your console should output

```
The list currently has 0 elements :
Enter a command : add
What number would you like to add? 2
Where would you like to add? Back
Enter a command : add
What number would you like to add? 37
Where would you like to add? Back
Enter a command : add
What number would you like to add? 15
Where would you like to add? Front
Enter a command : list
List: 15 2 37
```

~~Enter a command : previous Result: 15 Enter a command : previous Nothing is there! Enter a command : next Result: 15 Enter a command : next Result: 2 Enter a command : next Result: 37 Enter a command : next Nothing is there!~~

```
Enter a command : remove
Where would you like to remove? Back
Enter a command : list
List: 15 2
```

~~Enter a command : next Nothing is there~~

```
Enter a command : foo
Not a Number
Enter a command : exit
```

Style and Formatting

It is important that your code is not only functional but is also written clearly and with good style. We will be checking your code against a style checker that we are providing you with. It is located in resources along with instructions on how to use it. Since this is the first homework, we **WILL** be deducting points for style mistakes. If you feel like what you wrote is in accordance with good style but still sets off the style checker please email Hannah Lau hlau6@gatech.edu and Kush Mansingh kmansingh3@gatech.edu with the subject header of "Style XML".

Javadocs

Javadoc any helper methods you create in a style similar to the Javadocs for the methods in the interface. If a method already has Javadocs using @Override is acceptable.

Provided

The following files have been provided to you:

1. `LinkedListInterface.java`
2. `DoublyLinkedList.java`
3. `Node.java`

Deliverables

You must submit all of the following files. Please make sure the filename matches the filenames below.

1. `DoublyLinkedList.java`
2. `Node.java`
3. `Driver.java`

Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder, copy over the interfaces, recompile, and run. It is your responsibility to re-test your submission and discover editing oddities, upload issues, etc. You may attach each file individually, or submit them in a zip archive.