Lab07 - implements List161<T>

Due: Fri May 15, 2015

Let's code up some generic stuff, ala Ch 17 Generics. This lab covers:

- ❏ Chapter 19 Array lists - build your own
- ❏ Chapter 20 Linked lists - build your own
- ❏ Chapter 17 Generics
- ❏ StringBuilder

Grab the files in my lab07 folder and GO!
We'll focus primarily on the `List161` generic interface. In Lab07, we'll implement this list as 1) an array list, and 2) a linked list. Here's the interface with all my comments stripped out.

```
public interface List161<T> {
    boolean isEmpty();
    int size();
    void add( T element);   // add to end
    T get( int index);
    T remove();
    void clear();
}
```

Your list implementations will also have: 1) a ctor so you can create it, and 2) a `toString()` method so you can print your lists.

STEP 1 - Test the BogusList

I have an implementation of `List161<T>` ready to go. It's bogus because it uses a real `ArrayList` to do its thing. Let's create some tests to make sure it works. Mine looks like this:

```
public class ProfBillTester {

   public static String runTest( List161<String> list) {
      // test code here, use StringBuilder!
   }

}
```

Step 1, change the class to your name and pick a class other than `String`. In `runTest()`, add some objects, remove some, and print the results to a return string. In your `main()`, create a `BogusList` and pass it into your tester. Print the results.
When you're done, let me know… so we can share testers!

STEP 2 - Code up your array list

Now, we start the heavy lifting. implement `List161` using a resizable array. Use your notes, but not your book, please. Let's see:

- What are your class variables?
- This is a generic class, and you can't create a generic array. So, create an array of objects and cast it. If your generic type is T, then: `(T) new Object[size]`
- Create two constants in your class to control your resizing. We're forcing a small initial array size so that we get resizing with just a few elements in your list.

```
/** init size of your array */
private static final int INIT_ARRAY_SIZE = 2;
/** multiply array size by this factor with each resize */
private static final int ARRAY_RESIZE_MULTIPLIER = 2;
```

Once you're ready, create an array-based list back in `main()` and call your test method.

STEP 3 - Code up your linked list

Implement `List161` using a linked list.
Details:

- What are your class variables?
- Make your `Node<T>` class private inside your linked list class. Make your class variables in `Node public`, so you can just access them directly,
- Work out on a piece of paper: what's the empty list case? what's the non-empty case?

Just like the array-based list, test your linked list in `main()`.

STEP 4 - Bigger, better testing

Last step… let's share tester classes and run all the tests on our new code: 1, 2, 3.

1. Run the tester on the BogusList
2. Run the tester on your array-based list
3. Compare the test result strings… if they match, then you're golden.

Rince and repeat for your linked list. Done!

This automation of testing is common in software. You make a change, and then run an automated test suite to see if everything still works.
thanks… yow, bill