Assignment #2 - Index List, Positional List

Read through the assignment to get a basic idea of the requirements and then start working through the assignment in the recommended order. With the code example in the text book in Chapter 6 and 7, the actual assignment is simple, but you need to be aware of the different components that will be required for all assignments.

Objectives:

The main objectives of this assignment are:

- Implement abstract data structure List with array.
- Implement abstract data structure Positional List with DOUBLY linked list.
- Use JUnit test to debug and test each method
- Use good programming style and write good comments.
- Use Doubly Linked List to keep track of recent customers. This is similar to the LRU Cache replacement algorithm

Reference Resources

1. Array Lists:
    - Section 7.2 Array Lists (page 260-261)
    - Section 7.2.1 Dynamic Arrays (page 263-265)
    - Section 7.4.2 Implementing Iterators: iterations with the ArrayList class (page 284 - 285)
2. Positional Lists:
    - Section 7.3 Positional Lists (page 270-280)
    - Section 7.4.2 Implementing Iterators: iterations with the linkedPositionalListclass (page 286 - 287)

Allowed imports:

import net.datastructures.List;
import net.datastructures.Position;
import net.datastructures.PositionalList;
import java.util.Iterator;

The Assignment:

1. Download and import the zip file: `Assignment2_List.zip` For Eclipse:
    1) Open Eclipse

    2) Choose "File → Import".
    3) Select the "General" category.
    4) Pick "Existing Projects into Workspace"
    5) Click on "Next"
    6) Select the button for "Select Archive File" and browse to/select the zip file.
    7) Click on "Finish"
    8) A new project `List` has been added to the workspace:
2. Study the new project.
    1) First, study the `net.datastructures` package. You will find all the interfaces that we will cover this semester there. For this programming assignment, examine Queue.java, Position.java, PositionalList.java and List.java in detail. Read comments and JavaDoc carefully and understand what each method does clearly.
    2) Then, study the `cs2321` package. You will see an almost empty DoublyLinkedList.java and ArrayList.java. These are the two java classes that you need to implement correctly.
    3) Then, study the `Customer.java`. Read the comments at the beginning of the file carefully. Then implement it using a field of DoublyLinkedList.
    4) Last, but very important, study the **JUNIT test source code under folder `tests/ArrayList`, `tests/DoublyLinkedList`, `tests/Customer`** Make sure you understand the test setup and how each test case is written. You should make sure that your code pass all these test cases. You are encouraged to write more test cases when ever you find a situation that the existing test cases does not cover.
3. Implement ArrayList.java.
4. Implement DoublyLinkedList.java.

    Please note a few methods/classes have been implemented already. Take some time to understand how they were being implemented.

5. Implement CUSTOMER.java. Read the file block comment carefully to understand the requirement.

Important requirement and information

1. Write meaningful comments while you are implementing the code. See the example below. Please note you don't need to copy the Java doc from the interface file. I copied the header comment here so you can understand method remove(int).

```
/**
  * Removes and returns the element at the given index, shifting all subsequent
  * elements in the list one position closer to the front.
  * @param  i   the index of the element to be removed
  * @return the element that had be stored at the given index
  * @throws IndexOutOfBoundsException if the index is negative or greater than size()-1
  */

public E remove(int i) throws IndexOutOfBoundsException {

      // Method checkIndex will check if index i is within range [0..size-1],
  // and throw exception if not
      checkIndex(i,size-1);

      // Save the old data at index i before it is overwritten
```

```
        E old = data[i];

        // Shift all the elements starting from index i+1 to the last data at index size-1
        // to its left
        for (int k = i; k <= size-2; k++) {
                data[k]=data[k+1];
        }

        //reduce the data size
        size --;


        return old;
}
```

2. Test, Test, Test!!! Test your code for each method separately! Remember many test cases have been provided to you. See the section "Study the new project". Identify the related test cases under folder tests, and run them every time you thought you completed a method. You are encouraged to create more Junit test cases or use simply create `main` method and/or other methods to test your code.

3. NEVER, ever, ever modify ANY classes in the `net.datastructures` package. Don't create new files, don't remove file from net.datastructures. We overwrite this package with the originals when we test your code.

4. DO NOT change the class and method signature for DoublyLinkedList and ArrayList class. They need to stay as-is. Don't change the number of the arguments, the type of each argument, the order of the arguments. Don't change the return type. Don't change access modifiers.

5. DO NOT import java.util or other packages unless you have permission to do so

6. You may create your own class files, main method and other methods for implementation or testing purpose. If you create constructor, make sure it is PUBLIC, otherwise your program may break when we test it.

7. DELETE System.out.print() in all the methods that will be tested. You should clean up your code before you submit, especially you need to remove all your debugging System.out.print statements in the methods that we ask you to implement. Those print statments will cause your program takes much longer to finish. They will affect the performance when we test the efficiency of your code, and they may even cause the testing code to timeout. But you may keep your main methods and testing methods in place. It is totally fine to have print statements in your main methods or testing methods, because we don't run them when we grader your code.

Submission:

First, look at the following checklist:

1. Did you `import` any thing from `java.util` other than `Iterator`? . If so, remove them immediately.
2. Is the indentation easily readable? You can have Eclipse correct indentation by highlighting all code and select "Source → Correct Indentation".
3. Are comments well organized and concise?

If you have reviewed your code and all the criteria on the checklist are acceptable, follow the following procedure to export the files.
For Eclipse:

4. Open Eclipse
5. Choose "File → Export".
6. Select General -> Archive File
7. Click on "Next"
8. Click the down arrow for project "List", check the box next to src.
9. Browse for the destination/enter the archive file name as **prog2.zip**
10. Click on "Finish"
11. file prog2.zip will be created under the destination of your choice.

Double check the zip file has the correct folder structure and ALL the java source code files are under the correct folders. Missing any source file or directory could cause compilation failure. When the project does not compile, you will get 0 point.

Submit your file:
Login to Canvas and load your progN.zip to correct the assignment. You may submit many times. We will grade the latest submission only. When you choose the zip file to submit, make sure to submit the correct file. The timestamp on the file is a good indicator.

Grading Criteria:

- `ArrayList` and `DoublyLinkedList` that correctly implements all the required interfaces methods: 65 Points.
- `CUSTOMER` that correctly implements the List: 15 Points.
- `ArrayList` and `DoublyLinkedList` have been implemented efficiently: 10 points.
- Clear concise code with consistent style and good comments: 10 points