Programming Assignment 2

Due: 8 November 2024

CSC311, Data Structures

Begin by thoroughly reviewing the assignment to gain a foundational understanding of its requirements. Subsequently, embark on addressing the tasks in the recommended sequence. Given that the textbook provides code examples in Chapters 6 and 7, the practical aspects of the assignment should not pose significant challenges. Nonetheless, it is essential to recognize and account for the distinct components mandated for all assignments. While you are permitted to reference the code examples from the textbook, it is imperative to appropriately acknowledge and cite your source within your project. AI cannot be employed to generate programs, and all methods must be crafted from the ground up without relying on pre-existing methods in Java.

Objectives

The main objectives of this assignment are:

- Implement abstract data structrue List with array.
- Implement abstract data structrue PositionalList with DOUBLY linked list.
- Implement abstract data structrue Queue with CIRCULAR array.
- Use JUnit test to debug and test each method.
- Use good programming style and write good comments.

To complete this assignment, follow the steps listed below:

1. **Prepare the environment**

   Download and import the zip file: PA2.zip for Eclipse:

   (a) Open Eclipse

   (b) Select File, then Import

   (c) Select General category

   (d) Select Existing Projects into Workspace, and then click on Next

   (e) Select the button for "Select Archive File" and browse to/select the zip file.

   (f) Click on Finish. A new project List has been added to the workspace.

2. **Review Programming Assignment 2**

   Now, that you have added the project to your workplace:

   (a) First, study the net.datastructures package. You will find many interfaces. We will use a few of these interfaces for this assignment (PA2), and the next assignment(s). For this programming assignment, you should only examine Queue.java, Position.java, PositionalList.java and List.java in detail. Read comments and JavaDoc carefully and clearly understand what each method does.

   (b) Next, study the csc311 package. You will see the following 3, almost empty, java classes: DoublyLinkedList.java, ArrayList.java and CircularArrayQueue.java. These are the three java classes that you need to implement correctly.

   (c) Last, but very important, study the JUNIT test source code under folder tests/ArrayList and tests/DoublyLinkedList. 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 (not required) 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**

5. **Implement CircularArrayQueue.java**

6. **Write good comments** While you are implementing the above three files, besides typing the source code, you should also Write good comments to document source code. See the comment before the for loop in the example below. Please note you don't need to copy the Java doc from interface. 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()
  */

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 was overwritten
        E old = data[i];

        // Set element from [i .. size-2] to be the next one on its right.
        // In other words, all the elements starting from index i+1 to index size-1 have to shift to its left.
        for (int k = i; k <= size-2; k++) {
                data[k]=data[k+1];
        }

        //reduce the data size
        size --;


        return old;
}
```

7. **Test your code** Test, Test, and Test!!! Test your code. Remember try to implement one method and test it before you implement another method! 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.

   Important Notes:

   - 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.
   - DO NOT change the class and method signature for the CircularArrayQueue, 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.
   - All the given methods/data structures must be implemented from scratch.
   - 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. 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 asked 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.
   - Make sure you understand generic types (the E and <E>) in the source code. Refer to Sun's tutorial: Generics
   - Java Generics can be really picky when it comes to arrays. In order to initialize a generic array to store objects of type E, where E is a formal type, you will need to do:
     myArray = (E[])new Object[numberOfElements];

8. **Submit your project:**

   (a) Before submission, be sure to review the following checklist:

   - Will your project easily import to Eclipse? (Make sure to export your project as a single .zip file exactly like PA2.zip that has been provided to you. See below instruction for more information about exporting your project)
   - Does you programs compile without errors? (Programs that don't compile will not be graded)
   - Is the indentation easily readable? You can have Eclipse correct indentation by highlighting all code and select "Source $\rightarrow$ Correct Indentation".
   - Does the program meet all required interfaces?

- Are comments well organized and concise?

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

i. Open Eclipse, and choose File, then Export

ii. Select General, and then Archive File

iii. Click Next

iv. Click the down arrow for project "List", check the box next to src.

v. Browse for the destination/enter the archive file name as PA2.zip (not .rar)

vi. Click on "Finish"

vii. file PA2.zip List will be created under the destination of your choice.

(c) 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. For example, for the first assignment, the zip file should have folder list/src/csc311 and all the source files under csc311.

(d) For the submission, login to Canvas and upload your PA2.zip to the appropriate assignment. You may submit many times, the latest submission will be graded only.

Grading Criteria

- CircularArrayQueue, ArrayList and DoublyLinkedList that correctly implements all the required interfaces methods: (80 Points).
- Clear concise code with consistent style (10 points).
- Good commenting (10 points).
- When the project does not compile (-100 points)

FAQ

- Is the ArrayList fixed size or is the ArrayList resizable? *It is resizable. The initial size could be set as 16 as the text book suggested.*
- How should I expand the ArraySize? *Double the size once it reaches its capacity.*
- Do I need to shrink the array? *Not needed for this assignment.*
- Does the data in the array has to be stored continuously? *Yes.*
- Can we put NULL to some array cells ? *Yes. ArrayList does not care what data you store in each array cell.*