CSCI 230 Data Structures and Algorithms
Problem Set 2 - Data Structures
Student Name

Assignment

This assignment is based on material from the course primary textbook, "Data Structures and Algorithms in Java" by Michael Goodrich, chapters:

- Chapter 6 Stacks, Queues, and Dequeues
- Chapter 7 List and Iterator ADTs
- Chapter 8 Trees

Problem 1. Augment class `LinkedQueue` by adding a definition for `public static <E> void concatenate(LinkedQueue<E> q)` which appends all elements of `q` to the end of the `LinkedQueue`. The operation should run in constant-time and should leave `q` empty.

Hint: Modify the `SinglyLinkedList` class by adding a definition for the `concatenate` method which allows a modification of the list directly (which would be difficult otherwise since the necessary variables are declared as `private`.

Solution.

Code 1: `LinkedQueueOperations`

```
/* Example embedded code */
public class LinkedQueueOperations {
    public static <E> void concatenate(LinkedQueue<E> Q1, LinkedQueue<E> Q2) {
        // Append Q2 to Q1 in constant-time
        assert Q2.isEmpty() : "Error: Q2 should be empty!"
        // See https://stackoverflow.com/questions/5509082/eclipse-enable-assertions to
        // enable assertions
        // Terminal Users: java -ea EntryClass
    }
}
```

□

Problem 2. Modify the `LinkedPositionalList` implementation, as described in Section 3.6 from the course primary textbook, to support the `Cloneable` interface. The class declaraction should now read `public class LinkedPositionalList<E> implements PositionalList<E>, Cloneable`.

Solution.

Code 2: `LinkedPositionalList`

```
public class LinkedPositionalList<E> implements PositionalList<E>, Cloneable {
    ...
}
```

□

Problem 3. Implement a *preorder traversasl* lazy iterator for the `AbstractTree<E>` class, that is, your iterator must step through the elements of the tree in the same order a preorder traversal would. **Hint**: The `AbstractTree<E>` generic class already implements a *snapshot iterator* which you may use as a reference for your solution.

Solution.

Code 3: `AbstractTree`

```
public abstract class AbstractTree<E> implements Tree<E> {
    ...
    private class PreorderIterator implements Iterator<E> {
        ...
    }
    ...
}
```

□

Submission Guidelines

Modify this LaTeX document by inserting your solutions into the `solution` environments above. Submit this document along with any source code files `[*.java]` and archives `[*.jar]` to the course LMS. Finally, comment out the `\input{TexFiles/SubmissionGuidelines.tex}` line in `main.tex` to hide this section.