Data Structures and Algorithms with Java CS2105

Lab: Circular and Doubly Linked Lists [1 Mark]

Task 1 - Circular Singly Linked List

You are provided with the complete *CircularSinglyLinkedList<E>* class. Read the provided code carefully before starting. Make sure you understand how the circular linked list is implemented, then complete the required methods below.

Do not modify the existing methods.

Implement only the following two methods:

1. **traverse()** - Return type: **void**
   a. Start from the first node in the list.
   b. Print each element and move through the next reference.
   c. Stop when the traversal reaches the first node again.
   d. Correctly handle an empty list.
2. **rotate()** - Return type: **void**
   a. Move the first node logically to the end of the circular list.
   b. Update the tail reference appropriately.
   c. Do not create or delete any node.

Note: The circular linked list uses a single tail reference. The first node can be accessed using `tail.getNext()`.

Test Program

Create a separate file: TestCircularSinglyLinkedList.java

1. Create an empty circular linked list.
2. Insert several elements using the provided addLast() method.
3. Print the list using traverse().
4. Apply rotate() and print the list again.
5. Apply rotate() more than once and observe how the order changes.

Sample Output

```
1) Traverse empty list:
List is empty.

2) addLast: 10, 20, 30, 40
List: 10 -> 20 -> 30 -> 40

3) addFirst: 5, 3
List: 3 -> 5 -> 10 -> 20 -> 30 -> 40

4) removeFirst()
Removed: 3
List: 5 -> 10 -> 20 -> 30 -> 40

5) rotate() x3 (print after each rotation)
After rotate 1: 10 -> 20 -> 30 -> 40 -> 5
After rotate 2: 20 -> 30 -> 40 -> 5 -> 10
After rotate 3: 30 -> 40 -> 5 -> 10 -> 20
```

Task 2 - Doubly Linked List

You are provided with the complete DoublyLinkedList<E> class. **Read the provided code carefully before starting. Make sure you understand how the doubly linked list is implemented, including the header and trailer sentinel nodes, then complete the required methods below.**

Do not modify the existing implementation.

Implement only the following two methods:

1. **forwardTraversal()** - Return type: **void**
    a. Start from the first actual node.
    b. Print each element in forward order.
    c. Move using the next reference.
    d. Stop before reaching the trailer sentinel node.
2. **backwardTraversal()** - Return type: **void**
    a. Start from the last actual node.
    b. Print each element in reverse order.
    c. Move using the prev reference.
    d. Stop before reaching the header sentinel node.

Note: The provided doubly linked list uses **header** and **trailer** sentinel nodes.

Test Program

Create a separate file: TestDoublyLinkedList.java

1. Create a DoublyLinkedList<Integer>.
2. Insert 100, 200, 300, and 400 using the provided methods.
3. Print the list using forwardTraversal().
4. Print the list using backwardTraversal().
5. Create another list of type String and test both traversal methods.

Sample Output

```
Integer List - Forward:
100 -> 200 -> 300 -> 400
Integer List - Backward:
400 -> 300 -> 200 -> 100

String List - Forward:
ABC -> DEF -> GHI -> JKL
String List - Backward:
JKL -> GHI -> DEF -> ABC
```

Deliverables

- CircularSinglyLinkedList.java
- TestCircularSinglyLinkedList.java
- DoublyLinkedList.java
- TestDoublyLinkedList.java
- A screenshot showing the program output.

Important: Half a mark will be deducted for late submission, incomplete submission, or absence without an excuse.

Good luck