CS 314 Final Review — Merge Two Sorted Lists — Solution

Linked Lists

Write an instance method for the LinkedList314 class which will add all of the elements of another LinkedList314 in sorted order. Both lists will already be in ascended order before this method is called. No elements of either list are null. Your method will not return a list, instead it shall alter the calling list.

You may use the .compareTo() method. If two objects are equal, the one in this list comes first. The E extends Comparable<...> syntax ensures that all the elements in the Linked List implement the Comparable interface. This ensures that compareTo() can be safely called on the elements of the list.

/* Pre: other != null
 * Post: This list will contain all of the elements in this list and other.
 *       This list will be in sorted order.
 */
public void merge(LinkedList314<E> other);

Examples of calls to merge() (the values shown are Integer objects). The result shown is the new state of this.

- [1].merge([2]) => [1, 2]
- [0, 2, 4].merge([1, 3, 5]) => [0, 1, 2, 3, 4, 5]
- [1, 2, 7, 9, 10].merge([3, 5, 8, 12]) => [1, 2, 3, 5, 7, 8, 9, 10, 12]
- [].merge([1, 2, 3]) => [1, 2, 3]

Recall the LinkedList314 implementation from Lecture.

public class LinkedList314<E extends Comparable<? super E>> {
    private Node<E> first;
    private Node<E> last;
    private int size;

    private static class Node<E> {
        // The nested Node class.
        private E data;
        private Node<E> next;
        public Node(E d) { data = d; }
    }
}

You may not create any new Nodes or other data structures.
You may not use any other methods in the LinkedList class unless you implement them yourself. Do not use any other Java classes or methods.
/* Pre: other != null; this, other are sorted; no elements are null
 * Post: This list will contain all of the elements in this list and other.
 *        This list will be in sorted order.
 */
public void merge(LinkedList314<E> other) {
    //Update size instance variable
    size += other.size;

    Node<E> thisNode = first;
    Node<E> otherNode = other.first;
    while(thisNode != null && otherNode != null){
        int comp = thisNode.data.compareTo(otherNode.data);
        if(comp <= 0){
            thisNode = thisNode.next;
        } else{
            Node<E> temp = thisNode.next;
            thisNode.next = otherNode;
            otherNode.next = temp;
            otherNode = otherNode.next;
        }
    }

    Node<E> restOfList;
    if(thisNode == null && otherNode != null)
        restOfList = otherNode;
    else
        return;

    //Advance last reference
    last.next = otherNode;
    while(last.next != null)
        last = last.next;
}