Data Structure Lab6 : Doubly Linked List 2022-2023

Topics

1. Implement Node Class

```
public class Node<E> {
    E element;
    Node<E> next;
    Node<E> prev;

    public Node(E element, Node<E> prev, Node<E> next) {
        this.element = element;
        this.prev = prev;
        this.next = next;
    }
}
```

2. Implement DoublyLinkedList Class
3. Implement Basic Methods of DoublyLinkedList
    - isEmpty()

```
public boolean isEmpty() {
    return size == 0;
}
```

    - size()

```
public int size() {
    return size;
}
```

    - first()

```
public E first() {
    if (isEmpty()) return null;
    return header.next.element;
}
```

    - last()

```
public E last() {
    if (isEmpty()) return null;
```

```
    return trailer.prev.element;
}
```

- addFirst()

```
public void addFirst(E element) {
    addBetween(element, header, header.next);
}
```

- addLast()

```
public void addLast(E element) {
    addBetween(element, trailer.prev, trailer);
}
```

- removeFirst()

```
public E removeFirst() {
    if (isEmpty()) return null;
    return remove(header.next);
}
```

- removeLast()

```
public E removeLast() {
    if (isEmpty()) return null;
    return remove(trailer.prev);
}
```

Homework

1. Describe a method for finding the middle node of a doubly linked list with header and trailer sentinels by “link hopping,” and without relying on explicit knowledge of the size of the list. In the case of an even number of nodes, report the node slightly left of center as the “middle.”

```
public Node<E> findMiddle() {
    Node<E> forward = header.next;
    Node<E> backward = trailer.prev;

    while (forward != backward && forward.prev != backward) {
```

```
        forward = forward.next;
        backward = backward.prev;
    }
    return forward; // العقدة الوسطى
}
```

2. Give an implementation of the size( ) method for the DoublyLinkedList class, assuming that we did not maintain size as an instance variable.

```
public int size() {
    int count = 0;
    Node<E> current = header.next;
    while (current != trailer) {
        count++;
        current = current.next;
    }
    return count;
}
```

3. Implement the equals( ) method for the DoublyLinkedList class.

```
@Override
public boolean equals(Object o) {
    if (o == this) return true;
    if (!(o instanceof DoublyLinkedList)) return false;

    DoublyLinkedList<?> other = (DoublyLinkedList<?>) o;
    if (this.size() != other.size()) return false;

    Node<E> current1 = this.header.next;
    Node<?> current2 = other.header.next;

    while (current1 != trailer) {
        if (!current1.element.equals(current2.element)) return false;
        current1 = current1.next;
        current2 = current2.next;
```

```
    }
    return true;
}
```

4. Give an algorithm for concatenating two doubly linked lists L and M, with header and trailer sentinel nodes, into a single list L′.

```
public static <E> DoublyLinkedList<E> concatenate(DoublyLinkedList<E>
L, DoublyLinkedList<E> M) {
    if (L.isEmpty()) return M;
    if (M.isEmpty()) return L;

    L.trailer.prev.next = M.header.next;
    M.header.next.prev = L.trailer.prev;

    L.trailer = M.trailer;
    return L;
}
```

5. Our implementation of a doubly linked list relies on two sentinel nodes, header and trailer, but a single sentinel node that guards both ends of the list should suffice. Reimplement the DoublyLinkedList class using only one sentinel node.

```
public class DoublyLinkedList<E> {
    private Node<E> sentinel; // عقدة وحيدة لحماية الطرفين
    private int size = 0;

    public DoublyLinkedList() {
        sentinel = new Node<>(null, null, null);
        sentinel.next = sentinel;
        sentinel.prev = sentinel;
    }
}
```

6. Implement a circular version of a doubly linked list, without any sentinels, that supports all the public behaviors of the original as well as two new update methods, rotate( ) and rotateBackward.

Data Structure Lab6 : Doubly Linked List 2022-2023

```
public void rotate() {
    if (!isEmpty()) {
        trailer = trailer.next; // دوران للأمام
        header = header.next;
    }
}

public void rotateBackward() {
    if (!isEmpty()) {
        trailer = trailer.prev; // دوران للخلف
        header = header.prev;
    }
}
```

7. Implement the clone( ) method for the DoublyLinkedList class.

```
@Override
public DoublyLinkedList<E> clone() {
    DoublyLinkedList<E> cloned = new DoublyLinkedList<>();
    Node<E> current = this.header.next;
    while (current != trailer) {
        cloned.addLast(current.element); // نسخ كل عنصر
        current = current.next;
    }
    return cloned;
}
```