الاسم : زهرة خالد

تقنية معلومات عربي

Topics

1. Implement Node Class
2. Implement DoublyLinkedList Class
3. Implement Basic Methods of DoublyLinkedList
    - isEmpty()
    - size()
    - first()
    - last()
    - addFirst()
    - addLast()
    - removeFirst()
    - removeLast()

```
public class Node {
    public int data;
    public Node prev;
    public Node next;

    public Node(int data) {
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}

public class DoublyLinkedList {
    private Node head;
    private Node tail;
    private int size;

    public DoublyLinkedList() {
        this.head = null;
        this.tail = null;
        this.size = 0;
    }

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

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

public int first() {
    if (isEmpty()) {
        throw new IllegalStateException("List is empty");
    }
    return head.data;
}

public int last() {
    if (isEmpty()) {
        throw new IllegalStateException("List is empty");
    }
    return tail.data;
}

public void addFirst(int data) {
    Node newNode = new Node(data);
    if (isEmpty()) {
        head = newNode;
        tail = newNode;
    } else {
        newNode.next = head;
        head.prev = newNode;
        head = newNode;
    }
    size++;
}

public void addLast(int data) {
    Node newNode = new Node(data);
    if (isEmpty()) {
        head = newNode;
        tail = newNode;
    } else {
        newNode.prev = tail;
        tail.next = newNode;
        tail = newNode;
    }
    size++;
}
```

الاسم : زهرة خالد

تقنية معلومات عربي

```
    public int removeFirst() {
        if (isEmpty()) {
            throw new IllegalStateException("List is empty");
        }
        int removedData = head.data;
        if (head == tail) {
            head = null;
            tail = null;
        } else {
            head = head.next;
            head.prev = null;
        }
        size--;
        return removedData;
    }

    public int removeLast() {
        if (isEmpty()) {
            throw new IllegalStateException("List is empty");
        }
        int removedData = tail.data;
        if (head == tail) {
            head = null;
            tail = null;
        } else {
            tail = tail.prev;
            tail.next = null;
        }
        size--;
        return removedData;
    }
}
public class Main {
    public static void main(String[] args) {
        DoublyLinkedList dllist = new DoublyLinkedList();

        System.out.println(dllist.isEmpty());

        dllist.addFirst(1);
        dllist.addLast(2);
        dllist.addLast(3);

        System.out.println(dllist.size());
        System.out.println(dllist.first());
```

الاسم : زهرة خالد

تقنية معلومات عربي

```
 System.out.println(dllist.last());

        dllist.removeFirst();
        dllist.removeLast();

        System.out.println(dllist.size());
        System.out.println(dllist.first());
        System.out.println(dllist.last());    }
}
```

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 findMiddleNode() {
    Node slow = header.next;
    Node fast = header.next;

    while (fast != trailer && fast.next != trailer) {
        slow = slow.next;
        fast = fast.next.next;
    }

    return slow;
}
```

الاسم : زهرة خالد

تقنية معلومات عربي

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 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 obj) {
    if (this == obj) {
        return true;
    }

    if (obj == null || getClass() != obj.getClass()) {
        return false;
    }

    DoublyLinkedList other = (DoublyLinkedList) obj;
    if (size() != other.size()) {
        return false;
    }

    Node currentThis = header.next;
    Node currentOther = other.header.next;

    while (currentThis != trailer) {
        if (currentThis.data != currentOther.data) {
            return false;
        }
        currentThis = currentThis.next;
        currentOther = currentOther.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 void concatenate(DoublyLinkedList listM) {
    if (isEmpty()) {
        header.next = listM.header.next;
        trailer.prev = listM.trailer.prev;
    } else if (!listM.isEmpty()) {
        Node lastNodeL = trailer.prev;

        lastNodeL.next = listM.header.next;
        listM.header.next.prev = lastNodeL;

        trailer = listM.trailer;
    }

    size += listM.size();
    listM.clear();
}
```

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.

الاسم : زهرة خالد

تقنية معلومات عربي

```
public class DoublyLinkedList {
    private Node sentinel;

    public DoublyLinkedList() {
        sentinel = new Node();
        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.

```
public class CircularDoublyLinkedList {
    private Node current;

    public CircularDoublyLinkedList() {
        current = null;
    }

    public boolean isEmpty() {
        return current == null;
    }

    public void rotate() {
        if (!isEmpty()) {
            current = current.next;
        }
    }

    public void rotateBackward() {
        if (!isEmpty()) {
            current = current.prev;
        } }}
```

الاسم : زهرة خالد

تقنية معلومات عربي

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

```
@Override
public DoublyLinkedList clone() {
    DoublyLinkedList newList = new DoublyLinkedList();
    Node current = header.next;

    while (current != trailer) {
        newList.addLast(current.data);
        current = current.next;
    }

    return newList;
}
```