```java
/*
    Question 2 and 3
    Creator: Wahid Bawa
    Purpose: To create a doubly linked list
             Adds multiple delete methods
             Adds enqueue and dequeue
*/
public class LList {
    private LNode head, tail;

    public LList() {
        head = null;
        tail = null;
    }
    public void add(int val) { // Adds node to tail of list
        LNode tmp = new LNode(val, tail, null);
        if (tail != null) { // sets the "last" node's next to new node
            tail.setNext(tmp);
        }
        if (head == null) { // sets head to new node if list is empty.
            head = tmp;
        }
        tail = tmp;
    }

    public String toString() { // displays the parts of the linked list
        String ans = "";
        LNode tmp = head;
        while (tmp != null) {
            ans += tmp.getVal() + (tmp.getNext() == null ? "" : "-");
            tmp = tmp.getNext();
        }
        return ans;
    }

    public void enqueue(int val) { // calls add
        add(val);
    }

    public int dequeue() {// removes the head and then returns it
        LNode tmp = head;
        head = tmp.getNext();
        head.setPrev(null);
        return tmp.getVal();
    }
    private void delete(LNode node) { // used for deleting within the class
        if (node == head) { // this is the case for removing head
            head = node.getNext();
        }
        if (node == tail) { // this is the case for removing tails
            tail = node.getPrev();
        }
        if (node.getNext() != null) { // this updates the next node
            node.getNext().setPrev(node.getPrev());
        }
        if (node.getPrev() != null) { // this updates the previous node
            node.getPrev().setNext(node.getNext());
        }
    }
    public void delete(int val) { // loops through all nodes until "val" is found. Then calls the private delete method to get rid of it
        LNode tmp = head;
        while (tmp != null) {
            if (tmp.getVal() == val) {
                delete(tmp);
                break;
            }
            tmp = tmp.getNext();
        }
    }
    public void deleteAt(int index) { // loops through nodes until "index" is reached. then calls the private delete method
        int i = 0;
        LNode tmp = head;
        while (tmp != null) {
            if (i == index) {
                delete(tmp);
                break;
            }
            tmp = tmp.getNext();
            i++;
        }
    }
}
```