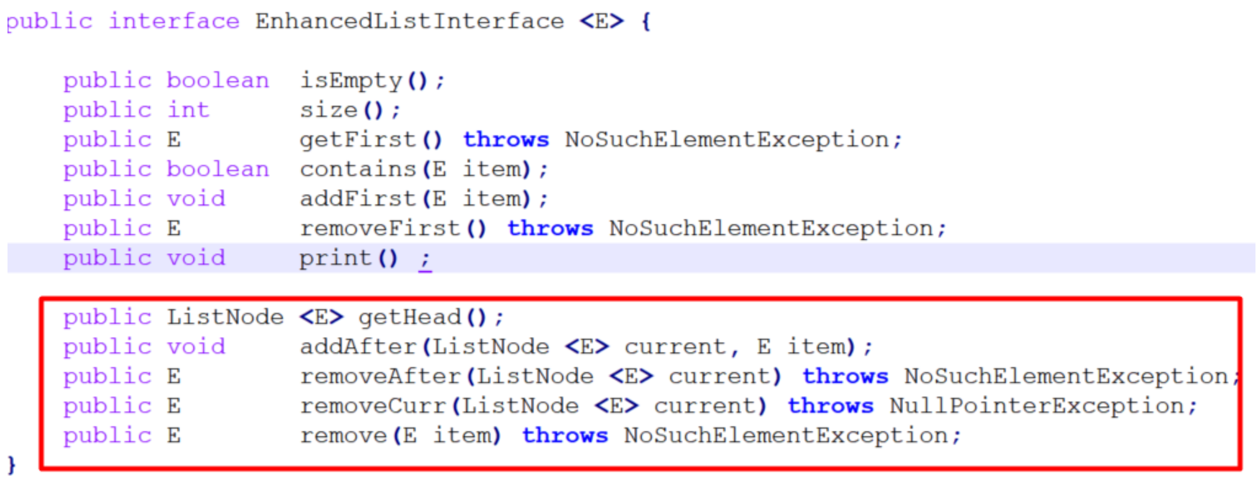
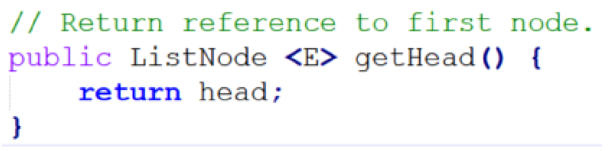
1. **Add new abstract method to the Interface**

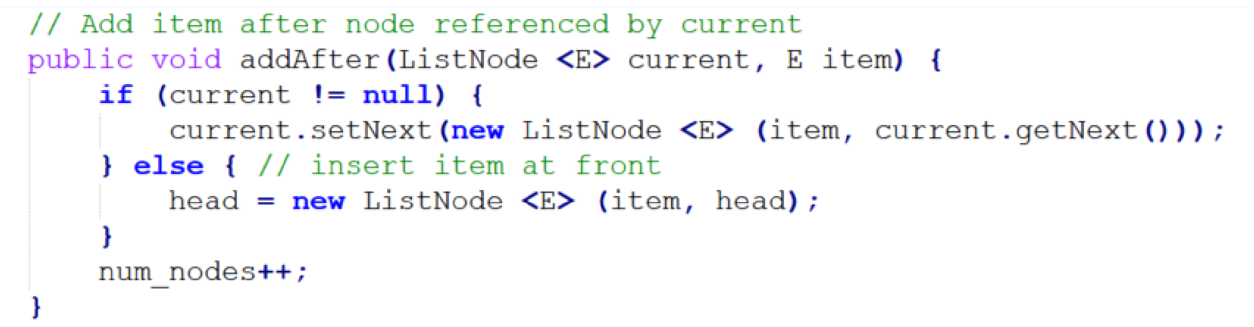


```
public interface EnhancedListInterface <E> {

    public boolean  isEmpty();
    public int      size();
    public E        getFirst() throws NoSuchElementException;
    public boolean  contains(E item);
    public void     addFirst(E item);
    public E        removeFirst() throws NoSuchElementException;
    public void     print() ;

    public ListNode <E> getHead();
    public void     addAfter(ListNode <E> current, E item);
    public E        removeAfter(ListNode <E> current) throws NoSuchElementException;
    public E        removeCurr(ListNode <E> current) throws NullPointerException;
    public E        remove(E item) throws NoSuchElementException;
}
```

2. **From ListNote.java, add a new method named “getHead”, “addAfter”,”removeAfter”, and “removeCurr” to BasicLinkedList.java to implement the operation of removing the current node, after node and add after.**



```
// Return reference to first node.
public ListNode <E> getHead() {
    return head;
}
```

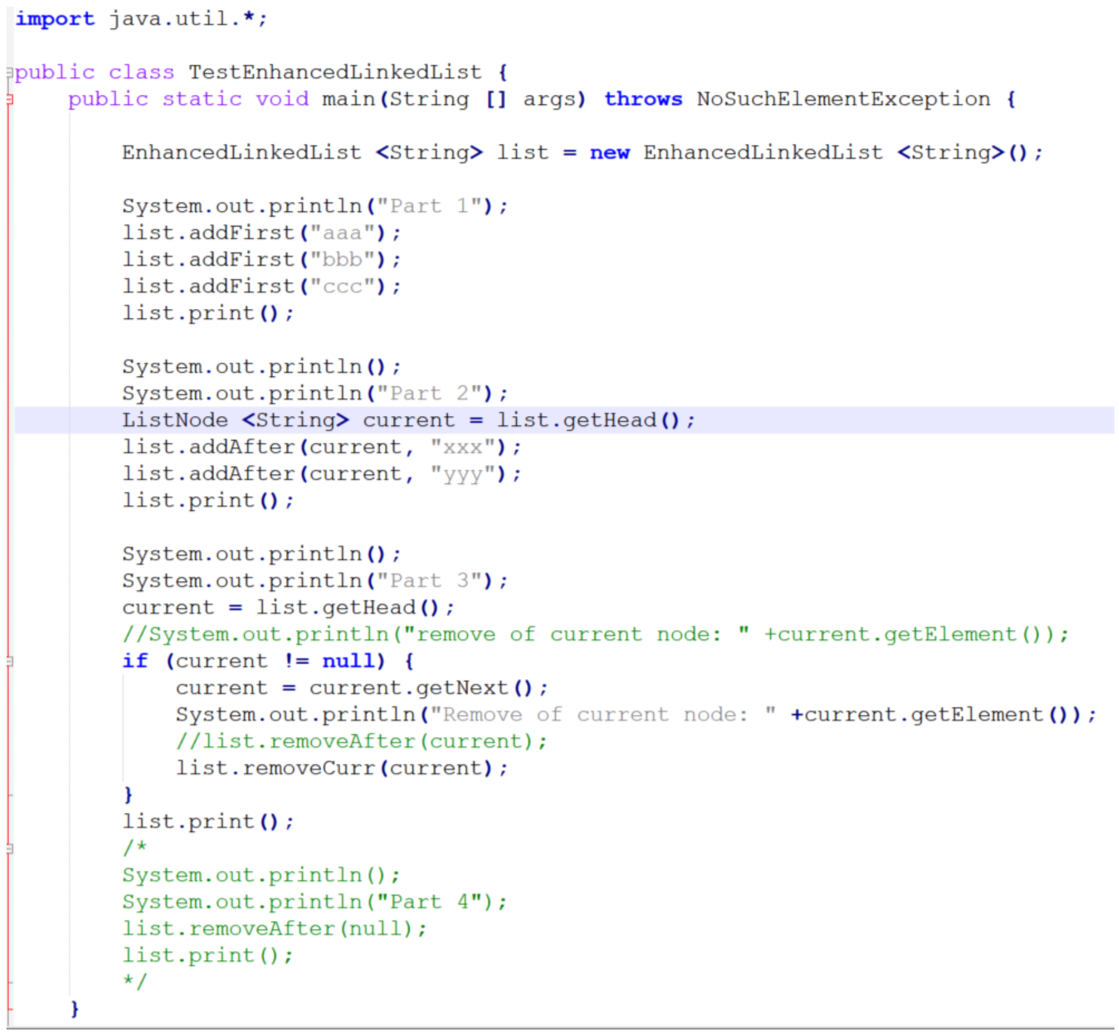


```
// Add item after node referenced by current
public void addAfter(ListNode <E> current, E item) {
    if (current != null) {
        current.setNext(new ListNode <E> (item, current.getNext()));
    } else { // insert item at front
        head = new ListNode <E> (item, head);
    }
    num_nodes++;
}
```

```
// Remove node after node referenced by current
public E removeAfter(ListNode <E> current) throws NoSuchElementException {
    E temp;
    if (current != null) {
        ListNode <E> nextPtr = current.getNext();
        if (nextPtr != null) {
            temp = nextPtr.getElement();
            current.setNext(nextPtr.getNext());
            num_nodes--;
            return temp;
        } else throw new NoSuchElementException("No next node to remove");
    } else { // if current is null, we want to remove head
        if (head != null) {
            temp = head.getElement();
            head = head.getNext();
            num_nodes--;
            return temp;
        } else throw new NoSuchElementException("No next node to remove");
    }
}
```

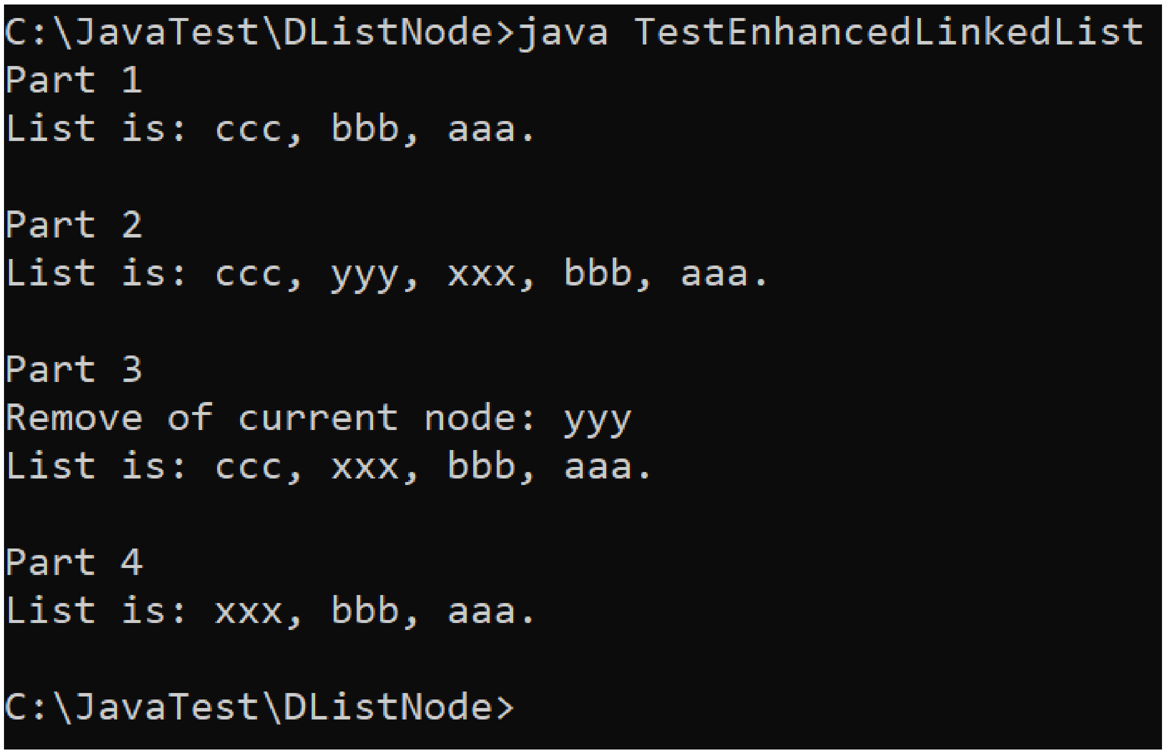
```
//Remove current node
public E removeCurr(ListNode <E> current) throws NullPointerException{
    if(head.getElement().equals(current.getElement()))
        head=head.getNext();
    ListNode <E> p=head;
    ListNode <E> q=head;
    while(p!=null)
    {
        if(p.getElement().equals(current.getElement()))
            break;
        q=p;
        p=p.getNext();
    }
    if(p!=null){
        q.setNext(p.getNext());
        num_nodes--;
    } else throw new NullPointerException("Null pointer");
    return current.getElement();
}
```

3. **Test your program**



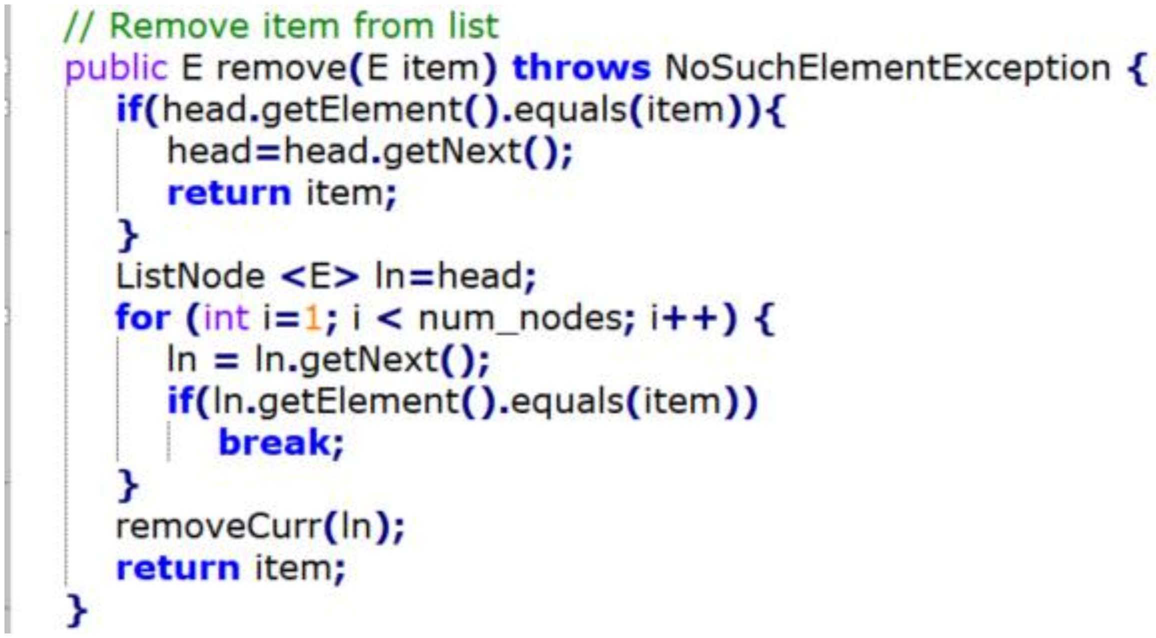
```java
import java.util.*;

public class TestEnhancedLinkedList {
    public static void main(String [] args) throws NoSuchElementException {

        EnhancedLinkedList <String> list = new EnhancedLinkedList <String>();

        System.out.println("Part 1");
        list.addFirst("aaa");
        list.addFirst("bbb");
        list.addFirst("ccc");
        list.print();

        System.out.println();
        System.out.println("Part 2");
        ListNode <String> current = list.getHead();
        list.addAfter(current, "xxx");
        list.addAfter(current, "yyy");
        list.print();

        System.out.println();
        System.out.println("Part 3");
        current = list.getHead();
        //System.out.println("remove of current node: " +current.getElement());
        if (current != null) {
            current = current.getNext();
            System.out.println("Remove of current node: " +current.getElement());
            //list.removeAfter(current);
            list.removeCurr(current);
        }
        list.print();
        /*
        System.out.println();
        System.out.println("Part 4");
        list.removeAfter(null);
        list.print();
        */
    }
```

Output:



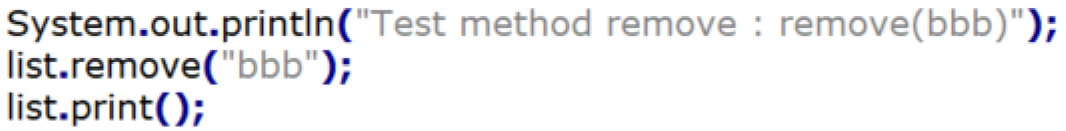
```
C:\JavaTest\DListNode>java TestEnhancedLinkedList
Part 1
List is: ccc, bbb, aaa.

Part 2
List is: ccc, yyy, xxx, bbb, aaa.

Part 3
Remove of current node: yyy
List is: ccc, xxx, bbb, aaa.

Part 4
List is: xxx, bbb, aaa.

C:\JavaTest\DListNode>
```

4. Remove item from list



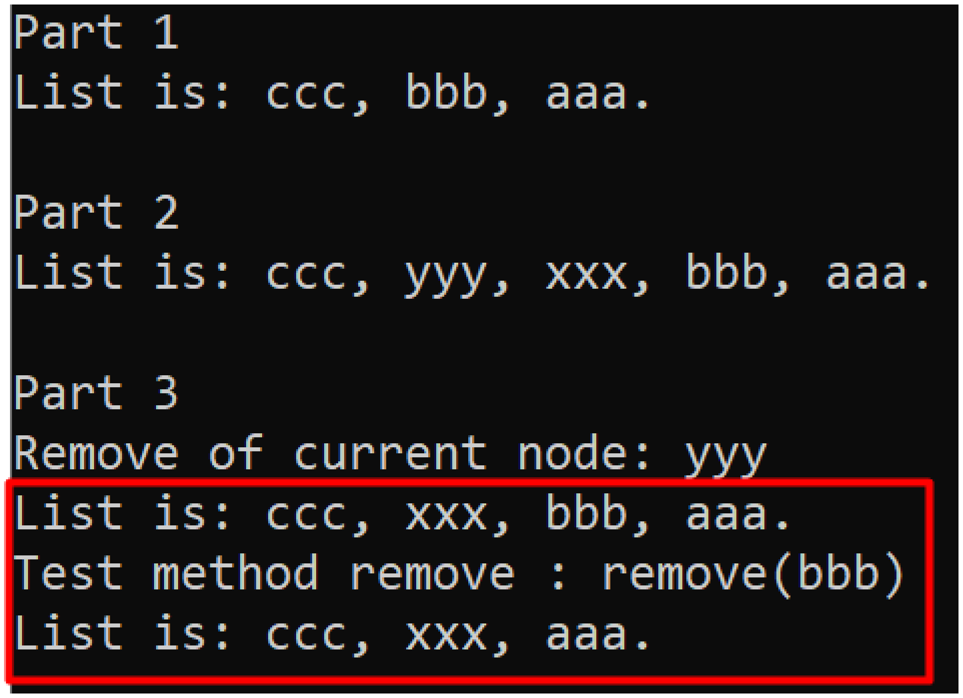
```
// Remove item from list
public E remove(E item) throws NoSuchElementException {
    if(head.getElement().equals(item)){
        head=head.getNext();
        return item;
    }
    ListNode <E> ln=head;
    for (int i=1; i < num_nodes; i++) {
        ln = ln.getNext();
        if(ln.getElement().equals(item))
            break;
    }
    removeCurr(ln);
    return item;
}
```

Call remove method to test your program:



```
System.out.println("Test method remove : remove(bbb)");
list.remove("bbb");
list.print();
```

Output



```
Part 1
List is: ccc, bbb, aaa.

Part 2
List is: ccc, yyy, xxx, bbb, aaa.

Part 3
Remove of current node: yyy
List is: ccc, xxx, bbb, aaa.
Test method remove : remove(bbb)
List is: ccc, xxx, aaa.
```