To start with, I must say that much of the sample input and expected output testing for this program was conducted while the code was unfinished, or more precisely, incorrectly written. Before I managed to complete the program, I ran into a few different instances where leaving a nodes previous reference “hanging” caused tremendous problems for me. The last correction to be made, required that a previous link be set when a node was removed from the middle of a list. When the previous link was left hanging, the result required that the removeLast() method be called a minimum of two times before the current last element of the list was removed. See the photos…

```
boolean found = false;
DoubleNode<T> previous = null;
DoubleNode<T> current = head;
DoubleNode<T> temp = null;

while(current != null && !found)
    if(targetElement.equals(current.getElement()))
        found = true;
    else{
        previous = current;
        current = current.getNext();
    }

if(!found)
    throw new ElementNotFoundException("Linked List");

if(size() == 1)
    head = tail = null;
else if(current.equals(head))
    head = current.getNext();
else if(current.equals(tail)){
    tail = previous;
    tail.setNext(null);
}else{
    previous.setNext(current.getNext());
    /*
     * The following two comments were added to complete the link
     * between the next node and the previous node. When they are
     * not implemented, the removeLast()would have to be called twice.
     */
    //temp = current.getNext();
    //temp.setPrev(current.getPrev());
}
```

```
27
28        System.out.println(list);
29
30        list.remove(7);
31        list.removeFirst();
32        list.remove(17);
33        list.removeLast();
34        list.remove(14);
35        //list.removeLast();   // This method is not called, take notice of
36                               // of the final numbers 16 and 23
37
38
39        System.out.println(list);
40
41
42
43
44        /* Test Results:
45            1 3 7 9 13 14 16 17 23 24
46            3 9 13 16
47        */
48    }
```

Problems Javadoc Declaration Console

```
<terminated> Driver [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (Oct 30, 2016, 7:52:20 PM)
1 3 7 9 13 14 16 17 23 24
3 9 13 16 23
```

```
28        System.out.println(list);
29
30        list.remove(7);
31        list.removeFirst();
32        list.remove(17);
33        list.removeLast();
34        list.remove(14);
35        list.removeLast();   // This method is called, and has to be called
36        list.removeLast();   // a second time for the last element to be removed
37
38
39        System.out.println(list);
40
41
42
43
44        /* Test Results:
45            1 3 7 9 13 14 16 17 23 24
46            3 9 13 16
47        */
48    }
```

Problems Javadoc Declaration Console

```
<terminated> Driver [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (Oct 30, 2016, 8:00:39 PM)
1 3 7 9 13 14 16 17 23 24
3 9 13 16
```

It is worth noting that noting that one of the most important aspects of a list is maintaining the link between the nodes. This is imperative for both previous and next references for doubly linked lists as well. Many of the methods need to be able to traverse through the list both forward and backwards through the list.

Further testing of the ordered list which is not pictured, includes trying to call a remove method for specific elements not in the list and trying to remove first and last elements when the list is empty. Appropriate exceptions were thrown in those cases. Final tests included random integers being added, making sure they were added in the correct position and removing said integers from the list.

Once the previous reference is properly set for a middle node in the remove() method the removeLast() method will only have to be called once to eliminate an element from the end of the list.