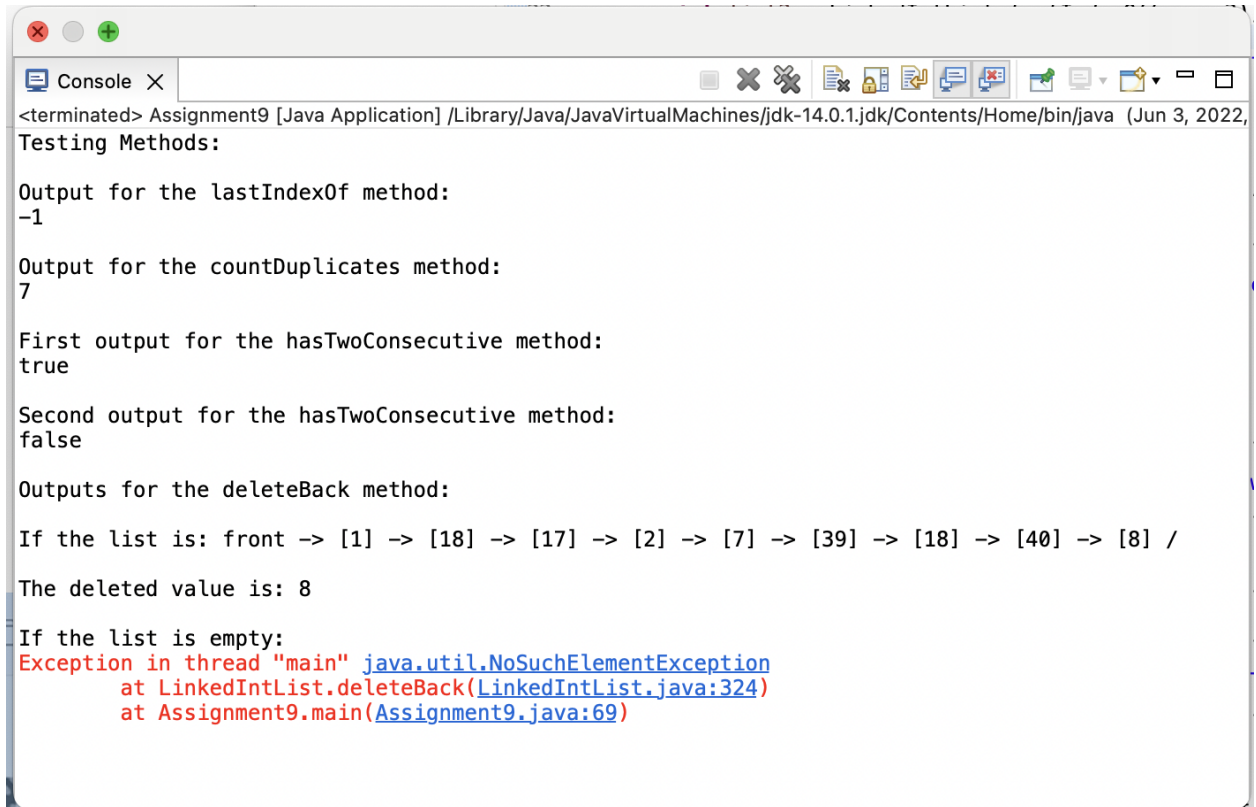


Student Name: **Natalia Reeck Zanini** Course: **CS 211**

Instructor: **Craig Niiyama**

QA Document for Assignment #9

Console Screenshot - Testing Methods:



```
<terminated> Assignment9 [Java Application] /Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home/bin/java (Jun 3, 2022,
Testing Methods:

Output for the lastIndexOf method:
-1

Output for the countDuplicates method:
7

First output for the hasTwoConsecutive method:
true

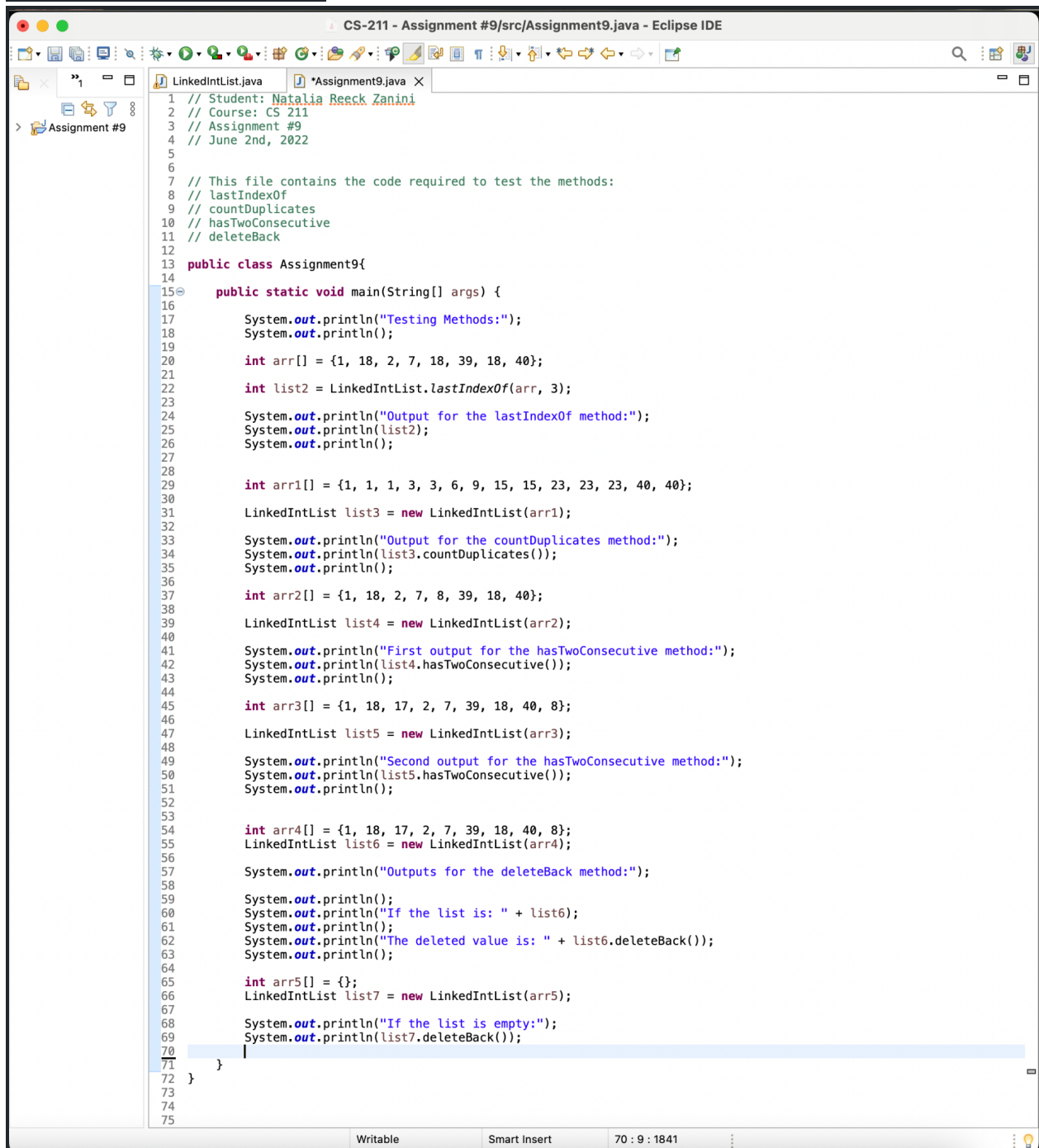
Second output for the hasTwoConsecutive method:
false

Outputs for the deleteBack method:

If the list is: front -> [1] -> [18] -> [17] -> [2] -> [7] -> [39] -> [18] -> [40] -> [8] /
The deleted value is: 8

If the list is empty:
Exception in thread "main" java.util.NoSuchElementException
    at LinkedList.deleteBack(LinkedList.java:324)
    at Assignment9.main(Assignment9.java:69)
```

Assignment9.java code:

The image is a screenshot of the Eclipse IDE interface. The title bar at the top reads "CS-211 - Assignment #9/src/Assignment9.java - Eclipse IDE". The left sidebar shows a project named "Assignment #9". The main editor window displays the code for "Assignment9.java". The code includes comments at the top identifying the student as Natalia Reeck Zanini, the course as CS 211, and the date as June 2nd, 2022. It also lists the methods being tested: lastIndexOf, countDuplicates, hasTwoConsecutive, and deleteBack. The main method contains several test cases using the LinkedList class, including array initialization, list creation, and method calls with output printing. The status bar at the bottom indicates "Writable", "Smart Insert", and a cursor position of "70 : 9 : 1841".

```
1 // Student: Natalia Reeck Zanini
2 // Course: CS 211
3 // Assignment #9
4 // June 2nd, 2022
5
6
7 // This file contains the code required to test the methods:
8 // lastIndexOf
9 // countDuplicates
10 // hasTwoConsecutive
11 // deleteBack
12
13 public class Assignment9{
14
15     public static void main(String[] args) {
16
17         System.out.println("Testing Methods:");
18         System.out.println();
19
20         int arr[] = {1, 18, 2, 7, 18, 39, 18, 40};
21
22         int list2 = LinkedList.lastIndexOf(arr, 3);
23
24         System.out.println("Output for the lastIndexOf method:");
25         System.out.println(list2);
26         System.out.println();
27
28         int arr1[] = {1, 1, 1, 3, 3, 6, 9, 15, 15, 23, 23, 23, 40, 40};
29
30         LinkedList list3 = new LinkedList(arr1);
31
32         System.out.println("Output for the countDuplicates method:");
33         System.out.println(list3.countDuplicates());
34         System.out.println();
35
36         int arr2[] = {1, 18, 2, 7, 8, 39, 18, 40};
37
38         LinkedList list4 = new LinkedList(arr2);
39
40         System.out.println("First output for the hasTwoConsecutive method:");
41         System.out.println(list4.hasTwoConsecutive());
42         System.out.println();
43
44         int arr3[] = {1, 18, 17, 2, 7, 39, 18, 40, 8};
45
46         LinkedList list5 = new LinkedList(arr3);
47
48         System.out.println("Second output for the hasTwoConsecutive method:");
49         System.out.println(list5.hasTwoConsecutive());
50         System.out.println();
51
52         int arr4[] = {1, 18, 17, 2, 7, 39, 18, 40, 8};
53         LinkedList list6 = new LinkedList(arr4);
54
55         System.out.println("Outputs for the deleteBack method:");
56
57         System.out.println();
58         System.out.println("If the list is: " + list6);
59         System.out.println();
60         System.out.println("The deleted value is: " + list6.deleteBack());
61         System.out.println();
62
63         int arr5[] = {};
64         LinkedList list7 = new LinkedList(arr5);
65
66         System.out.println("If the list is empty:");
67         System.out.println(list7.deleteBack());
68
69     }
70 }
71
72
73
74
75
```

LinkedList.java - Methods code section:

```
CS-211 - Assignment #9/src/LinkedList.java - Eclipse IDE

LinkedList.java x Assignment9.java

> Assignment #9

243 // YOUR CODE GOES HERE
244
245 // #1) This method returns the index in the list of the last occurrence
246 // of that value, or -1 if the value is not found in the list:
247
248
249
250 public static int lastIndexOf(int[] arr, int num) {
251     // Initiating index to -1:
252     int index = -1;
253
254     for (int i = 0; i < arr.length; i++) {
255         if (arr[i] == num) {
256             index = i;
257         }
258     }
259     return index;
260 }
261
262 // #2) This method returns the number of duplicates in a sorted list:
263
264 public int countDuplicates() {
265     // If the list is empty, returns zero:
266
267     if (front == null) {
268         return 0;
269     }
270
271     int count = 0;
272     int curr = front.data;
273
274     // Looping through remaining nodes:
275     for (ListNode n = front.next; n != null; n = n.next) {
276         if (n.data == curr) {
277             count++;
278         }
279         else {
280             curr = n.data;
281         }
282     }
283
284     return count;
285 }
286
287
288
289 // #3) This method will return whether or not a list of integers has two
290 // adjacent numbers that are consecutive integers:
291
292 public boolean hasTwoConsecutive() {
293     // Here if the list is empty/has only one element it will return false:
294     if (front == null || front.next == null) {
295         return false;
296     }
297
298     ListNode n = front;
299     // The loop will continue until n.next is null.
300     while (n.next != null) {
301         // Comparing n and n.next
302         if (n.data == (n.next.data - 1)) {
303             // If consecutive is found:
304             return true;
305         }
306
307         // If not, it will advance to next node:
308         n = n.next;
309     }
310
311     // In case it is not found:
312     return false;
313 }
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
```

```
316 }
317
318 // #4) This method deletes and returns the last value.
319 // If the list is empty, then it throws NoSuchElementException error.
320
321 public int deleteBack() {
322     if (front == null)
323         // If list is empty throws NoSuchElementException.
324         throw new NoSuchElementException();
325
326     int data;
327     // If front is the last node it will remove the front.
328     if (front.next == null) {
329         data = front.data;
330         front = null;
331         // Will return the value.
332         return data;
333     }
334
335     // The loop will continue until n.next is the last node.
336     ListNode n = front;
337     while (n.next.next != null)
338         n = n.next;
339     data = n.next.data;
340     // removing n.next
341     n.next = null;
342     return data;
343 }
344
345
346
```