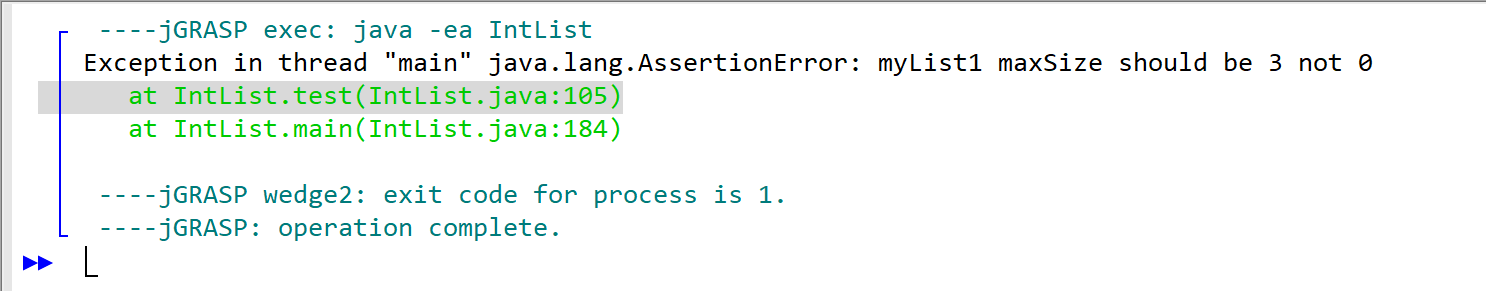
Ian Cowan

CSC-270 Lab 2 Report

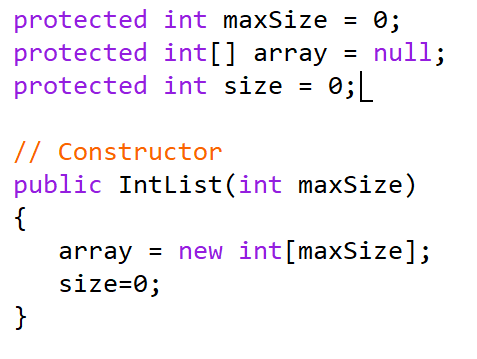
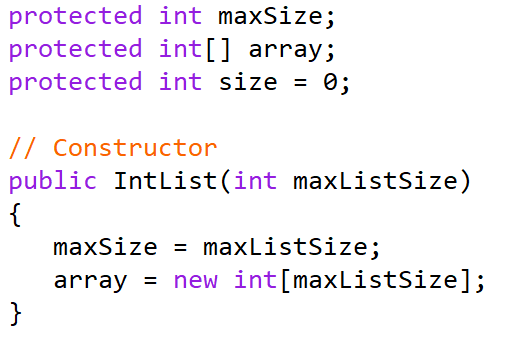
September 15, 2019

To begin this lab, we were given a stub file for an array based list of integers. The first part of this lab was to write tests to test the code of each method of the class. After completing all of these tests, we continued onto checking Timmy’s written code with our tests and then correcting those errors. There were five bugs found in Timmy’s code and four of them were found using tests and the last one was caught while investigating another bug.

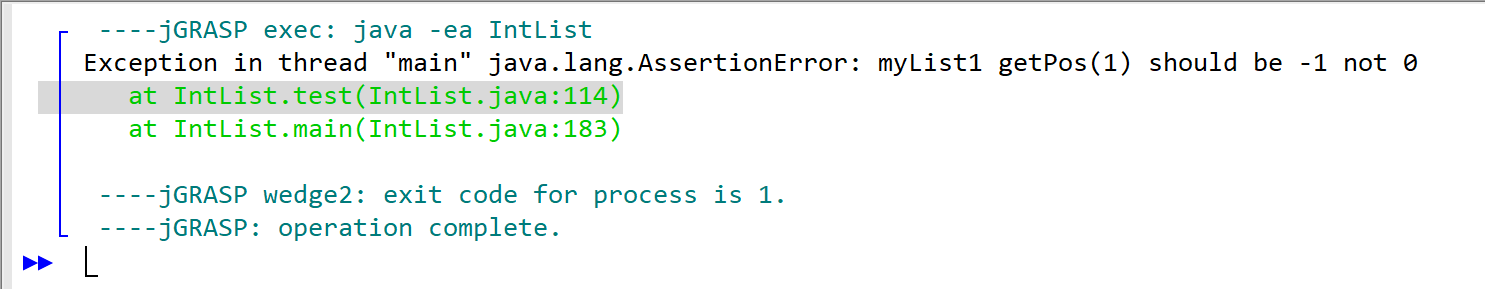
The first of these bugs was the fact that max size was being set to 0 by default. This is incorrect because the user should be able to specify the max size when creating a new IntList. I found this with one of my initial tests that check the maxSize variable after creating a new list.



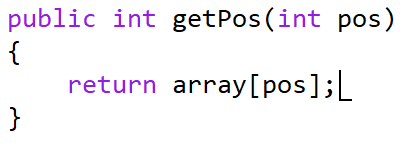
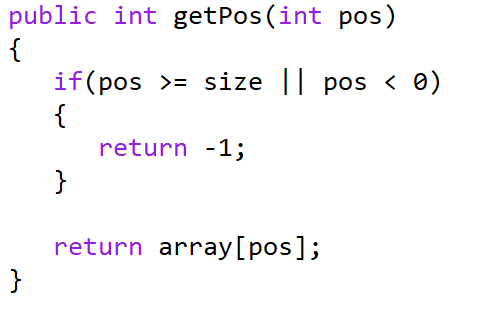
I corrected this bug by assigning the maxSize variable within the constructor, rather than when the attribute is initialized. When fixing this bug, there were a couple of other errors that didn’t break the code but reduced the efficiency of the code when creating a new IntList that I changed.

 ———> 

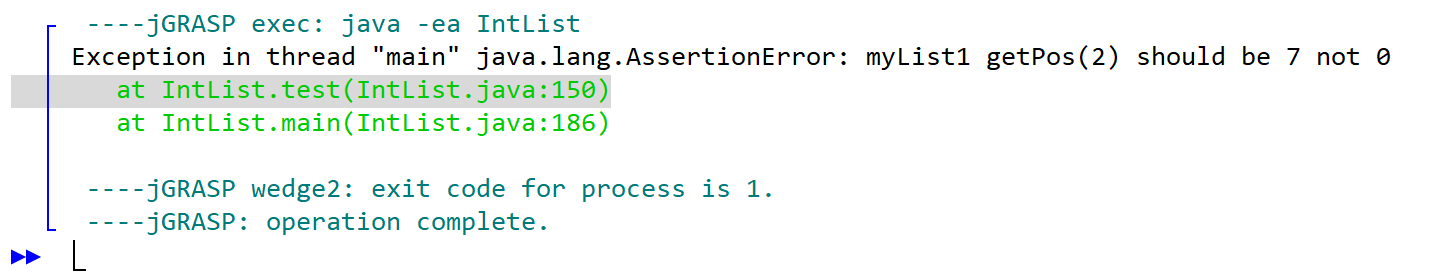
The next bug was causing the getPos() method to not return -1 if the position does not exist in the list. In this case, it was returning 0 which is an incorrect response. I found this bug within the initial tests that attempted to get the position of an index that should be less than the size, because the size after creating an IntList should be 0.



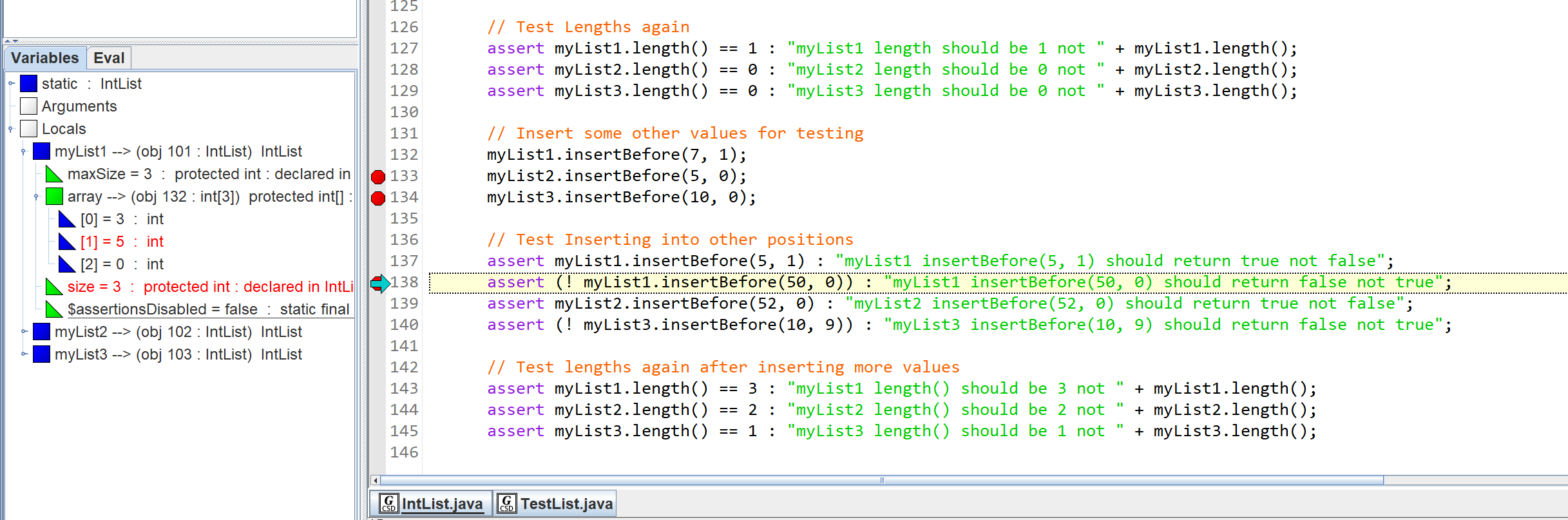
In order to fix this bug, I added an if statement that checks and makes sure the position is greater than or equal to size or if the position is less than 0. If this results in true, it returns a -1 but if not, it just continues on through the getPos method, returning the value of that position.

 ————>

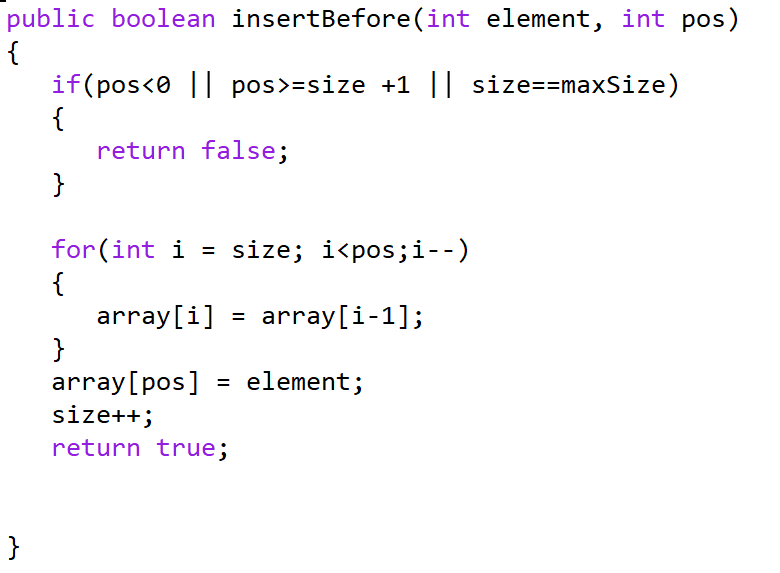
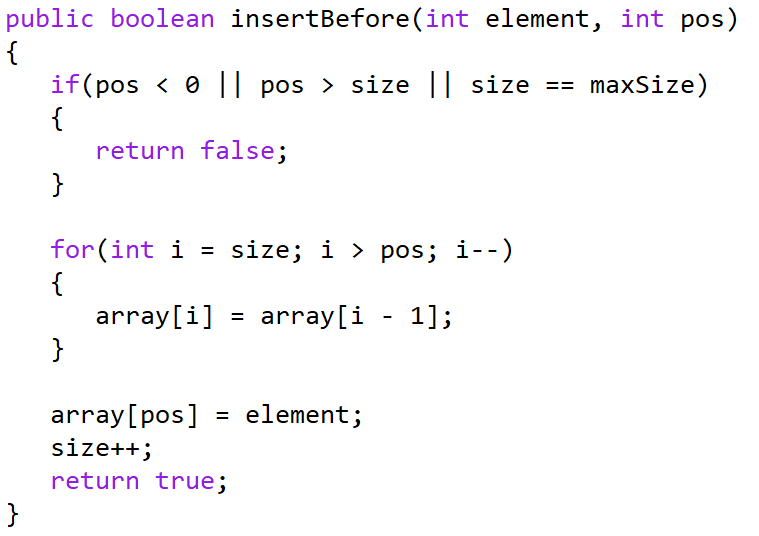
The third bug was a little more complicated than the first two. This bug was causing the getPos() method to return an incorrect value directly after inserting another value into a position other than at the end of the list. This was again discovered by one of my unit tests.

****

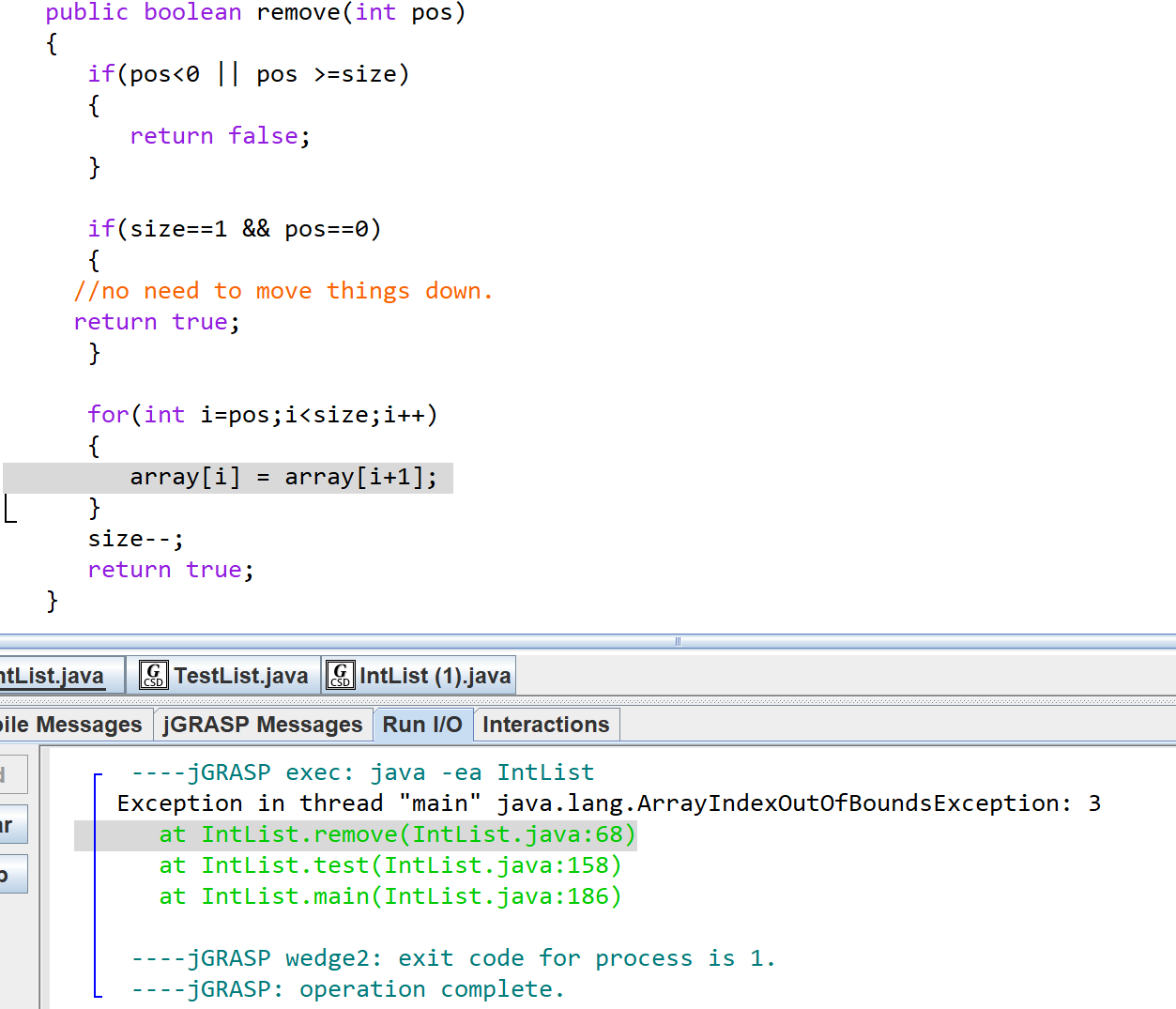
I used an educated guess to guess that this bug was happening because of an issue with the insertBefore() method. I used the debugger to observe the array attribute of myList1, which is an IntList, and I discovered that when inserting into a position that already has a value, the value gets inserted correctly but the other elements of the list were not being moved correctly. In the example below, I was inserting 5 into position 1 which already had a value of 7. This should have moved 7 to the 2 position, but instead just overwrote the 7 with a 5 in position 1.

****

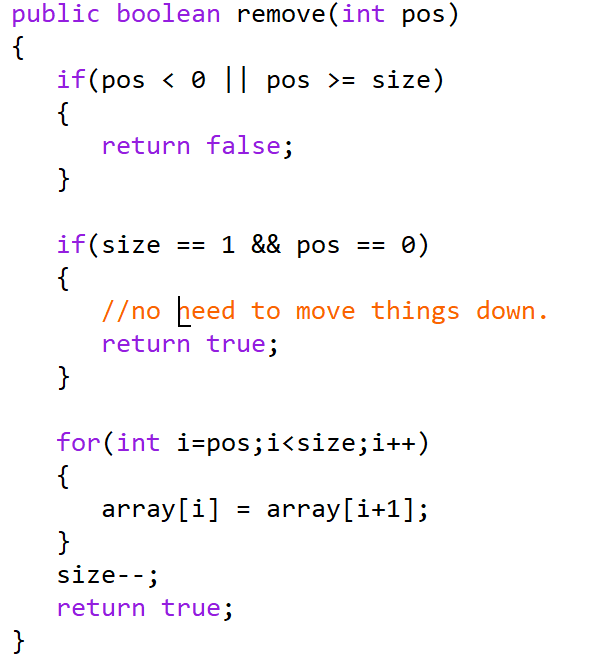
It took me a couple of tries to find the issue with this one, but it ended up being an issue with the position and index comparison in the for loop. I was able to correct this by simply changing “pos < 0” to “pos > 0”. Note that I also changed “pos >= size + 1” to “pos > size”, but this did not change the action of this method. I just changed this while investigating the bug.

** ——>**

When removing an element, there was an out of bounds exception that was arising within the loop for remove(). I discovered this error while running one of the tests, and there wasn’t an assertion error but an actual error when running the loop.



While investigating this bug, I noticed that if the size is 1 and the position that is being removed is 0, it returns true but the size doesn’t actually decrement, thus causing an error. I corrected this bug, but the bug I was investigating still persisted. I fixed the bug I was investigating by subtracting 1 in the for loop condition. This allows the loop to terminate before the index goes out of range for moving the elements within the list.

 ———>