

Olasunkanmi Olayinka – SEC01 (NUID 001512266)

Big Data System Engineering with Scala

Fall 2022

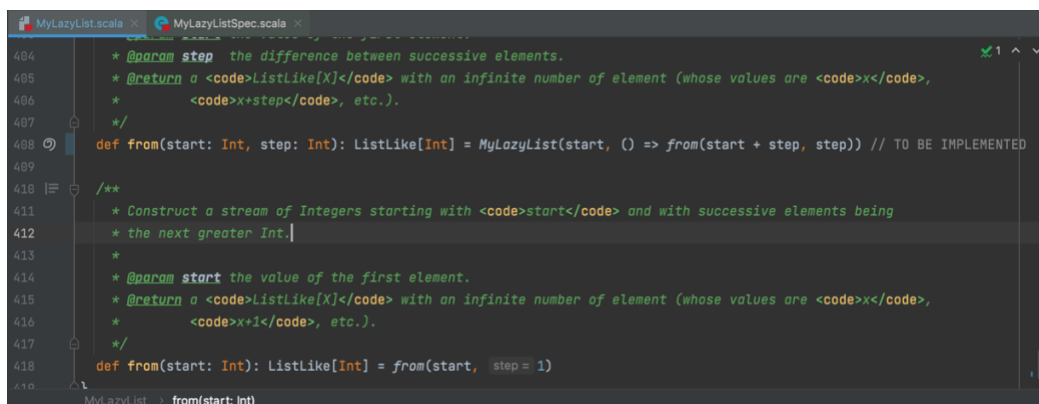
Assignment No. 2



-List of Tasks Implemented

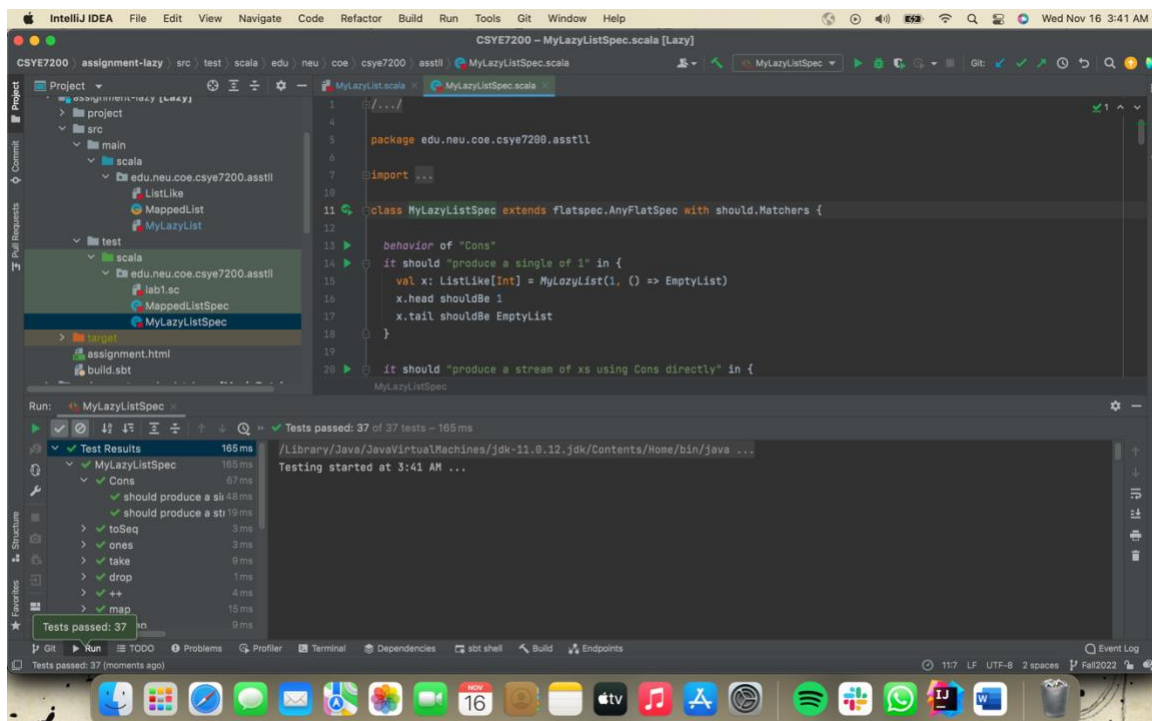
- Implement the *from* method in the companion object of MyLazyList
- Explain the difference in which MyLazyList differs from LazyList and why I think there is a difference
- Explain what this code does and why it is needed
- List all the recursive calls that are in MyLazyList
- List all the mutable variables and collections in MyLazyList
- Explain the purpose of the zip method
- Explain why there is no length/size method for MyLazyList

-Code



```
484      * @param step the difference between successive elements.
485      * @return a <code>ListLike[X]</code> with an infinite number of element (whose values are <code>x</code>,
486      *       <code>x+step</code>, etc.).
487      */
488      def from(start: Int, step: Int): ListLike[Int] = MyLazyList(start, () => from(start + step, step)) // TO BE IMPLEMENTED
489
490      /**
491      * Construct a stream of Integers starting with <code>start</code> and with successive elements being
492      * the next greater Int.
493      *
494      * @param start the value of the first element.
495      * @return a <code>ListLike[X]</code> with an infinite number of element (whose values are <code>x</code>,
496      *       <code>x+1</code>, etc.).
497      */
498      def from(start: Int): ListLike[Int] = from(start, step = 1)
499
```

-Unit tests



```
1 //...
2
3 package edu.neu.coe.csye7200.asstll
4
5 import ...
6
7 class MyLazyListSpec extends FlatSpec with should.Matchers {
8
9   behavior of "Cons"
10
11   it should "produce a single of 1" in {
12     val x: ListLike[Int] = MyLazyList(1, () => EmptyList)
13     x.head shouldBe 1
14     x.tail shouldBe EmptyList
15   }
16
17   it should "produce a stream of xs using Cons directly" in {
18     // ...
19   }
20 }
```

Run: MyLazyListSpec

Tests passed: 37 of 37 tests - 165 ms

Test Results

Test	Time
MyLazyListSpec	165 ms
Cons	97 ms
should produce a single of 1	48 ms
should produce a stream of xs using Cons directly	10 ms
toSeq	3 ms
ones	3 ms
take	0 ms
drop	1 ms
++	4 ms
map	15 ms

Tests passed: 37

- Result

- MyLazyList extends LazyListLike abstract class so when getting a tail of MyLazyList you get a ListLike obj, but in LazList you get another LazyList.
- It yields the tail of a MyLazyList stream
- 43, 70, 116, 131
- There are no “var” in MyLazyList
- It returns a tuple of ListLike objects of the corresponding element else it returns an EmptyList
- The elements are only computed when needed, hence we don't know the length, if it's not empty means the head has been computed and the tail is another MyLazyList