Qinyun Lin – SEC01 (NUID 001582464)

Big Data System Engineering with Scala
Spring 2022
Assignment No. 2

Task

1. **Implement the “from()” method in the “MyLazyList”**
2. **Answer 6 questions related to the “MyLazyList” implementation**

1. (a) what is the chief way by which *MyLazyList* differs from *LazyList* (the built-in Scala class that does the same thing). Don't mention the methods that *MyLazyList* does or doesn't implement--I want to know what is the *structural* difference.
   (b) Why do you think there is this difference?
2. Explain what the following code actually does and why is it needed?

```
def tail = lazyTail()
```

3. List all of the recursive calls that you can find in *MyLazyList* (give line numbers).
4. List all of the mutable variables and mutable collections that you can find in *MyLazyList* (give line numbers).
5. What is the purpose of the *zip* method?
6. Why is there no *length* (or *size*) method for *MyLazyList*?

Solution

1. Implement the “from()” method in the “MyLazyList”

```
def from(start: Int, step: Int): ListLike[Int] =
  new MyLazyList[Int](start, () => from(start+step, step))
```

2. Answer 6 questions related to the “MyLazyList” implementation

(1) Main difference between “MyLazyList” and scala built-in LazyList, and the reason

a. In MyLazyList, we pass the head by “value” and pass the tail by ”name”(by function), so the head in MyLazyList is not “lazy”.
   While in LazyList, we pass head and tail together with a State Object, which is passing by “name.”
   In addition, MyLazyList is sub class of “IterableOnce”, while LazyList is sub class of “Serializable”.

b. The reason for passing head and tail with an State Object by “name” is to make head also “lazy”.
   As for the reason for using “IterableOnce” in MyLazyList is that it is lighter than another one.

(2) It defines tail as a function, which will generate next element when it is invoked. With this code, the list can truly “Lazy”, because it knows how to generate but it won’t generate anything until someone calls it.

(3) Recursive calls:

Line 26: ++ function
Line 43: in flatMap
Line 69: in filter
Line 82: in zip
Line 98: in take
Line 116: in drop
Line 131: in toSeq, inner
Line 408 & 419: from

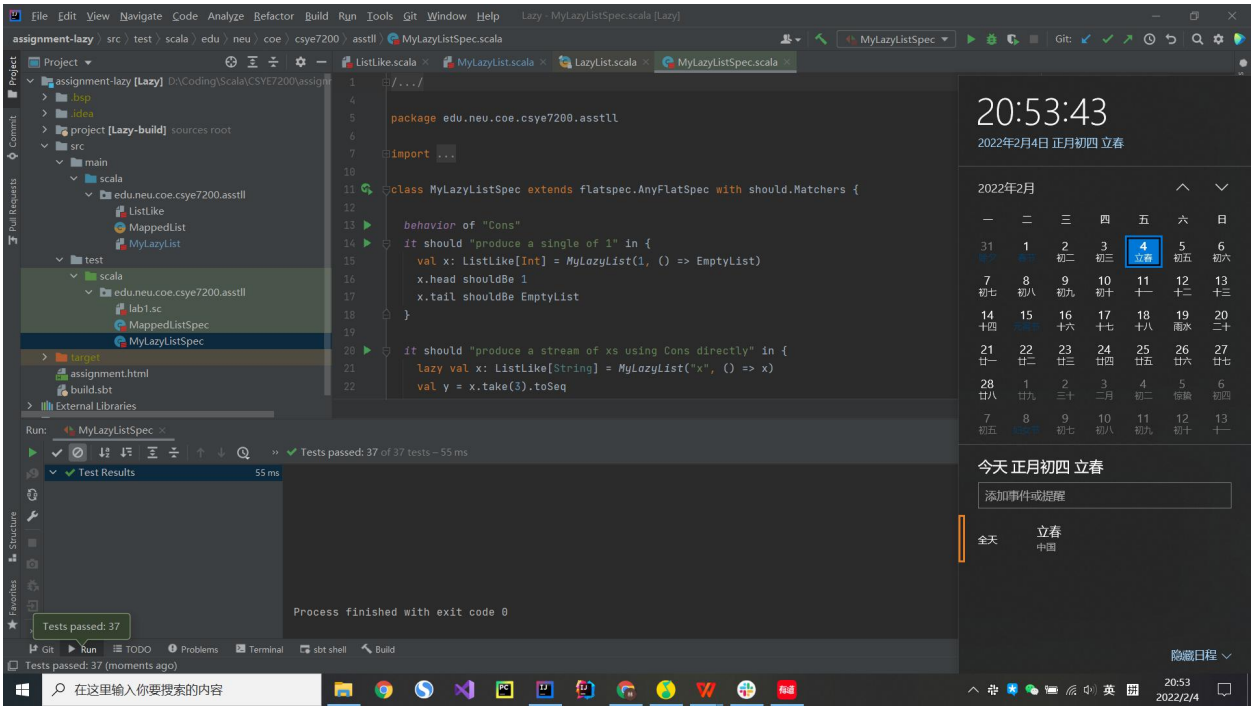
(4) Mutable variable and collections:

Line 42: val y
Line 69: val tailFunc
Line 388: lazy val ones

(5) Zip function let us can deal with 2 lazy list as pairs, and it can automatically ignore remaining elements if one list is longer. Besides, it will preserve the laziness because the pairs will only be processed when needed.

(6) Because MyLazyList is infinite

Unit Test Screenshot



Project Source

https://github.com/MrNiro/CSYE7200/tree/Spring2022/assignment-lazy/src/main/scala/edu/neu/coe/csye7200/asstll