Shangqing Hu – SEC01 (NUID 001374342)

# Big Data System Engineering with Scala
# Spring 2023
# Assignment No.4(RandomState)

**-List of Tasks Implemented**

We have talked about the importance of having a service of some kind able to create a new state rather than mutate the existing state. Here you will create a random number service that is modeled as a state. Please see my blog on this subject for background: Working with mutable stateLinks to an external site..

We need to create a trait called RandomState which will have two obvious methods: next and get. Of course, we don't really know what the type of the result of get will be, so let's make it parametric, thus: RandomState[T].

But once we have a RandomState[T], we will want to be able to map it into a RandomState[U] so we'll need to implement map. While we're at it, we might as well implement flatMap too. Technically, this will mean that it's a "monad" but we haven't talked about those yet -- but they are important.

There's one other convenience method that we should probably implement and that is toStream which will return a LazyList[T]. As usual, I have provided the basic framework and a specification for your work: src/main/scala/edu/neu/coe/csye7200/asstrs/RandomState.scala and the corresponding RandomStateSpec in the test directory. All you have to do is to implement the 6 TO BE IMPLEMENTED and run the tests. When it's all green, you're done. You can get these from the class repo (see Course Material/Resources/Class Repository), the module name for this assignment is assignment-random-state.

However, we are going to be getting a little deeper into functional programming in this assignment and some of the concepts will be new to you. You will have to stretch your minds (again) and think out of the box a little. I don't want you to do this assignment in pairs. It's sufficiently important that I want each of you to work on it alone. You will benefit from that when it comes to the mid-term exam.

I want to commend to you another one of my blogs which we talked a little about already. Read it. I do believe it will help you materially in this assignment. It is here: Simple, obvious, elegantLinks to an external site..

Also, there are a couple of useful methods defined on Function1: compose and andThen. Keep that in mind as you work on your implementations.

One further piece of advice: try the easier implementations first (the number of points available is more or less proportional to the degree of difficulty). You will get the hang of it as you go.

You don't need to create a new project, just clone/download class repo, and import assignment-random-state, you may follow the video demo under Canvas / Assignments / Video Demo for Assignment Import and Submit. Remember to follow the Canvas / Assignments / Standard procedure for submitting assignments.

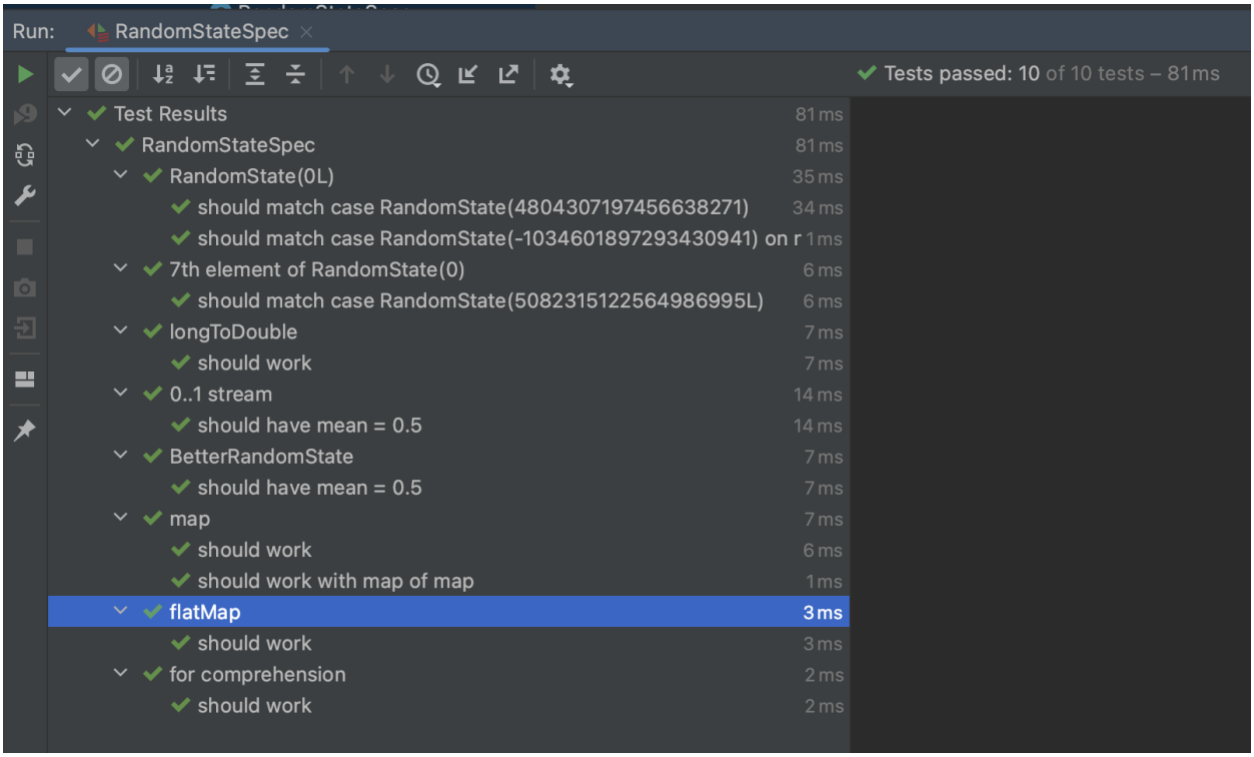
## -Code

```scala
  def flatMap[U](f: T => RandomState[U]): RandomState[U] = f(get) // TO BE IMPLEMENTED

  /**
    * @return a stream of T values
    */
  // Hint: This a recursively method and it concatenate current element with following elements.
  // 12 points
  new *
  def toStream: LazyList[T] = get #:: next.toStream // TO BE IMPLEMENTED
```

```scala
case class JavaRandomState[T](n: Long, g: Long => T) extends RandomState[T] {
  // Hint: Remember to use the "seed" to generate next RandomState.
  // 7 points
  new *
  def next: RandomState[T] = JavaRandomState(new Random(n).nextLong(), g) // TO BE IMPLEMENTED
  // Hint: Think of the input and output.
  // 5 points
  new *
  def get: T = g(n) // TO BE IMPLEMENTED
  // Hint: This one need function composition.
  // 13 points
  new *
  def map[U](f: T => U): RandomState[U] = JavaRandomState[U](n, g.andThen(f)) // TO BE IMPLEMENTED
}
```

```scala
  def apply(): RandomState[Long] = apply(System.currentTimeMillis)

  // Hint: This is a easy one, remember that it not only convert a Long to a Double but also scale down the number to -1 ~ 1.
  // 4 points
  val longToDouble: Long => Double = x => 2.0 * (x.toDouble - Long.MinValue.toDouble) / (Long.MaxValue.toDouble - Long.MinValue.toDouble) - 1.0 // TO BE IMPLEMENTED
  val doubleToUniformDouble: Double => UniformDouble = { x => UniformDouble((x + 1) / 2) }
}
```

**-Unit tests**


Run:
RandomStateSpec
Tests passed: 10 of 10 tests – 81ms
Test Results 81 ms
RandomStateSpec 81 ms
RandomState(0L) 35 ms
should match case RandomState(4804307197456638271) 34 ms
should match case RandomState(-1034601897293430941) on r 1 ms
7th element of RandomState(0) 6 ms
should match case RandomState(5082315122564986995L) 6 ms
longToDouble 7 ms
should work 7 ms
0..1 stream 14 ms
should have mean = 0.5 14 ms
BetterRandomState 7 ms
should have mean = 0.5 7 ms
map 7 ms
should work 6 ms
should work with map of map 1 ms
flatMap 3 ms
should work 3 ms
for comprehension 2 ms
should work 2 ms