Komal Suryan – SEC01 (NUID 002747707)

Big Data System Engineering with Scala
Spring 2023
Assignment No. 5

Function.scala

```
def map2[T1, T2, R](t1y: Try[T1], t2y: Try[T2])(f: (T1, T2) => R): Try[R] =
  for (t1 <- t1y; t2 <- t2y) yield f(t1, t2) // TO BE IMPLEMENTED
```

```
def map3[T1, T2, T3, R](t1y: Try[T1], t2y: Try[T2], t3y: Try[T3])(f: (T1, T2, T3) =>
R): Try[R] =
  for (t1 <- t1y; t2 <- t2y; t3 <- t3y) yield f(t1, t2, t3) // TO BE IMPLEMENTED
```

```
for (t1 <- t1y; t2 <- t2y; t3 <- t3y;
     t4 <- t4y; t5 <- t5y; t6 <- t6y; t7 <- t7y) yield f(t1, t2, t3, t4, t5, t6, t7)
// TO BE IMPLEMENTED
```

```
def lift[T, R](f: T => R): Try[T] => Try[R] = _ map f // TO BE IMPLEMENTED
```

```
def lift2[T1, T2, R](f: (T1, T2) => R): (Try[T1], Try[T2]) => Try[R] = map2(_,_)(f)
// TO BE IMPLEMENTED
```

```
def lift3[T1, T2, T3, R](f: (T1, T2, T3) => R): (Try[T1], Try[T2], Try[T3]) =>
Try[R] = map3(_,_,_)(f)// TO BE IMPLEMENTED
```

```
def lift7[T1, T2, T3, T4, T5, T6, T7, R](f: (T1, T2, T3, T4, T5, T6, T7) => R):
(Try[T1], Try[T2], Try[T3], Try[T4], Try[T5], Try[T6], Try[T7]) => Try[R] =
map7(_,_,_,_,_,_,_)(f) // TO BE IMPLEMENTED
```

```
def invert2[T1, T2, R](f: T1 => T2 => R): T2 => T1 => R =  t2 => t1 => f(t1)(t2) //
TO BE IMPLEMENTED
```

```
/ If you can do invert2, you can do this one too
def invert3[T1, T2, T3, R](f: T1 => T2 => T3 => R): T3 => T2 => T1 => R = t3 => t2
=> t1 => f(t1)(t2)(t3) // TO BE IMPLEMENTED
```

```
// If you can do invert3, you can do this one too
def invert4[T1, T2, T3, T4, R](f: T1 => T2 => T3 => T4 => R): T4 => T3 => T2 => T1
=> R = t4 => t3 => t2 => t1 => f(t1)(t2)(t3)(t4)  // TO BE IMPLEMENTED
```

```
// This one is a bit harder. But again, think in terms of an anonymous function that
is what you want to return
def uncurried2[T1, T2, T3, R](f: T1 => T2 => T3 => R): (T1, T2) => T3 => R = (t1,t2)
=> t3 => f(t1)(t2)(t3) // TO BE IMPLEMENTED
```

```
// If you can do uncurried2, then you can do this one
def uncurried3[T1, T2, T3, T4, R](f: T1 => T2 => T3 => T4 => R): (T1, T2, T3) => T4
=> R = (t1,t2,t3) => t4 => f(t1)(t2)(t3)(t4) // TO BE IMPLEMENTED
```

```
// If you can do uncurried3, then you can do this one
def uncurried7[T1, T2, T3, T4, T5, T6, T7, T8, R](f: T1 => T2 => T3 => T4 => T5 =>
T6 => T7 => T8 => R): (T1, T2, T3, T4, T5, T6, T7) => T8 => R =
  (t1,t2,t3,t4,t5,t6,t7) => t8 => f(t1)(t2)(t3)(t4)(t5)(t6)(t7)(t8) // TO BE
IMPLEMENTED
```

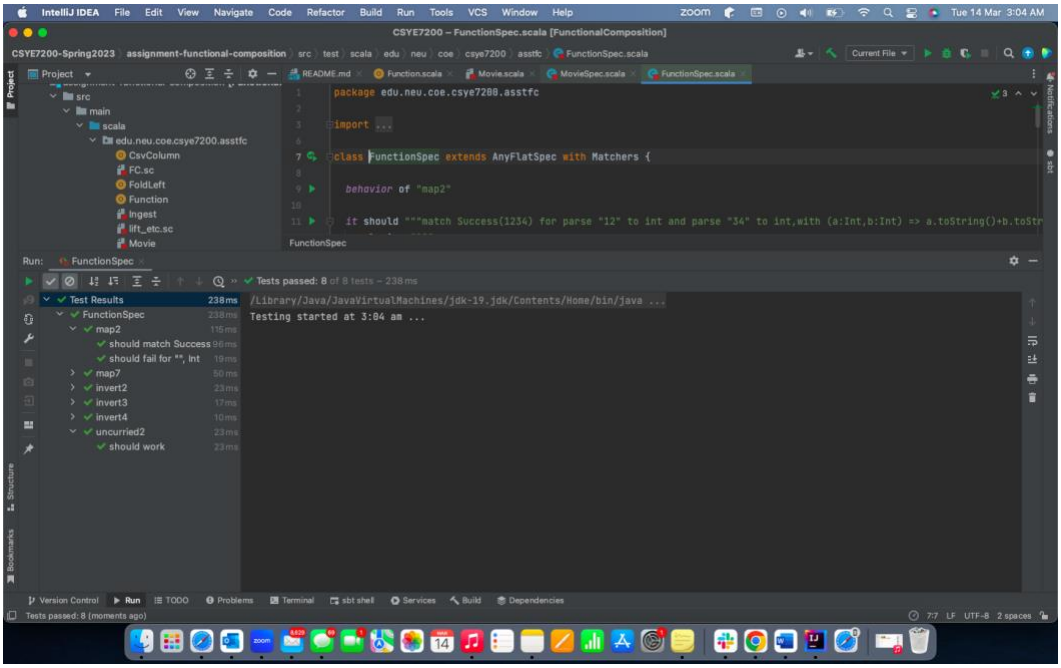
Movie.scala

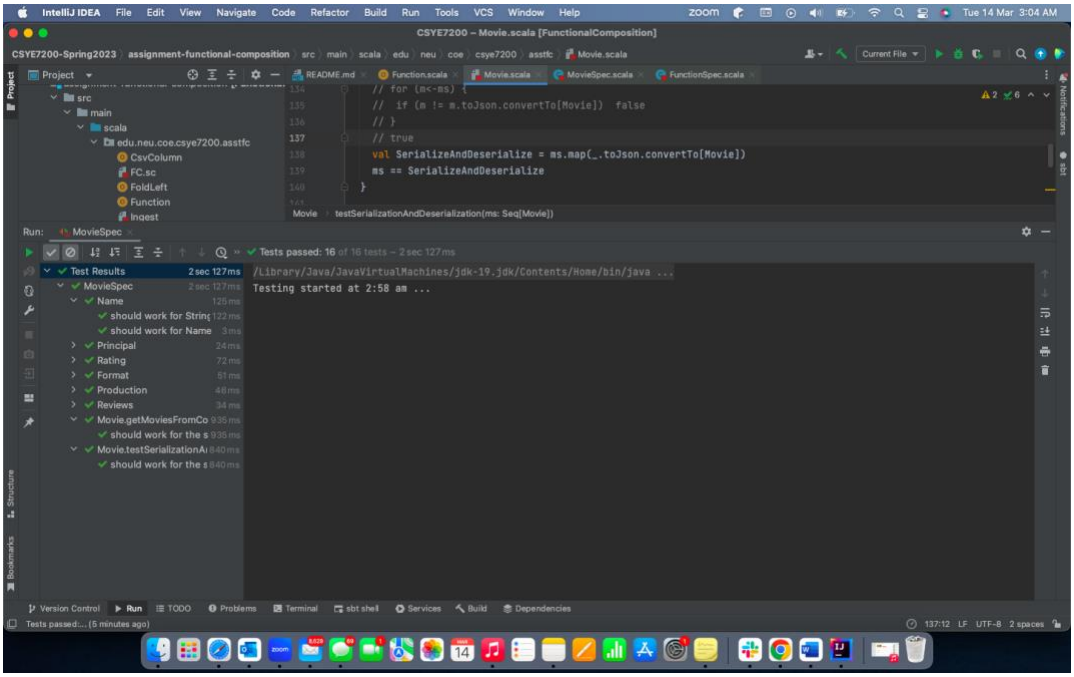
```
//Hint: You may refer to the slides discussed in class for how to serialize object
to json
object MoviesProtocol extends DefaultJsonProtocol {
  // 20 points
  // TO BE IMPLEMENTED
  implicit val formatSERFormat: RootJsonFormat[Format] = jsonFormat4(Format.apply)
  implicit val productionFormat: RootJsonFormat[Production] =
jsonFormat4(Production.apply)
  implicit val ratingFormat: RootJsonFormat[Rating] = jsonFormat2(Rating.apply)
  implicit val reviewsFormat: RootJsonFormat[Reviews] = jsonFormat7(Reviews.apply)
  implicit val nameFormat: RootJsonFormat[Name] = jsonFormat4(Name.apply)
  implicit val principalFormat: RootJsonFormat[Principal] =
jsonFormat2(Principal.apply)
  implicit val movieFormat: RootJsonFormat[Movie] = jsonFormat11(Movie.apply)

}
```

```
//Hint: Serialize the input to Json format and deserialize back to Object, check the
result is still equal to original input.
def testSerializationAndDeserialization(ms: Seq[Movie]): Boolean = {
  // 5 points
  // TO BE IMPLEMENTED
  import MoviesProtocol._
  // for (m<-ms) {
  //  if (m != m.toJson.convertTo[Movie])  false
  // }
  // true
  val SerializeAndDeserialize = ms.map(_.toJson.convertTo[Movie])
  ms == SerializeAndDeserialize
}
```

Unit Tests


CSYE7200 – FunctionSpec.scala [FunctionalComposition]
package edu.neu.coe.csye7200.asstfc
class FunctionSpec extends AnyFlatSpec with Matchers {
behavior of "map2"
Tests passed: 8 of 8 tests – 238 ms
Testing started at 3:04 am ...


CSYE7200 – Movie.scala [FunctionalComposition]
Tests passed: 16 of 16 tests – 2 sec 127 ms
Testing started at 2:58 am ...