Deepika Balasubramanian – SEC01 (NUID 002194564)

Big Data System Engineering with Scala  
Spring 2022   
Assignment No. 5



**-List of Tasks Implemented**

Function.scala:

* map2 function
* map3 function
* map7 function
* lift function
* lift2 function
* lift3 function
* lift7 function
* invert2 function
* invert3 function
* invert4 function
* uncurried2 function
* uncurried3 function
* uncurried7 function

Movie.Scala:

* MoviesProtocol Object
* Def testSerializationandDeserialization

**Code:**

**Code for** map2 function:

def map2[T1, T2, R](t1y: Try[T1], t2y: Try[T2])(f: (T1, T2) => R): Try[R] = *Try*(f(t1y.get, t2y.get))

**Code for** map3 function:

def map3[T1, T2, T3, R](t1y: Try[T1], t2y: Try[T2], t3y: Try[T3])(f: (T1, T2, T3) => R): Try[R] = *Try*(f(t1y.get, t2y.get, t3y.get))

**Code for** map7 function:

def map7[T1, T2, T3, T4, T5, T6, T7, R](t1y: Try[T1], t2y: Try[T2], t3y: Try[T3], t4y: Try[T4], t5y: Try[T5], t6y: Try[T6], t7y: Try[T7])  
 (f: (T1, T2, T3, T4, T5, T6, T7) => R): Try[R] = *Try*(f(t1y.get, t2y.get, t3y.get, t4y.get, t5y.get, t6y.get, t7y.get))

Code for lift function:

def lift[T, R](f: T => R): Try[T] => Try[R] = \_ map f

Code for lift2 function:

def lift2[T1, T2, R](f: (T1, T2) => R): (Try[T1], Try[T2]) => Try[R] = *map2*(\_, \_)(f)

Code for lift3 function:

def lift3[T1, T2, T3, R](f: (T1, T2, T3) => R): (Try[T1], Try[T2], Try[T3]) => Try[R] = *map3*(\_, \_, \_)(f)

Code for lift7 function:

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)

Code for invert2 function:

def invert2[T1, T2, R](f: T1 => T2 => R): T2 => T1 => R = (x:T2) => (y:T1) => f(y)(x)

Code for invert3 function:

def invert3[T1, T2, T3, R](f: T1 => T2 => T3 => R): T3 => T2 => T1 => R = (x:T3)=> (y:T2) => (z:T1) => f(z)(y)(x)

Code for invert4 function:

def invert4[T1, T2, T3, T4, R](f: T1 => T2 => T3 => T4 => R): T4 => T3 => T2 => T1 => R = (w:T4) => (x:T3)=> (y:T2) => (z:T1) => f(z)(y)(x)(w)

Code for uncurried2 function:

def uncurried2[T1, T2, T3, R](f: T1 => T2 => T3 => R): (T1, T2) => T3 => R = (x:T1, y:T2) => (z:T3) => f(x)(y)(z)

Code for uncurried3 function:

def uncurried3[T1, T2, T3, T4, R](f: T1 => T2 => T3 => T4 => R): (T1, T2, T3) => T4 => R = (w:T1, x:T2, y:T3) => (z:T4) => f(w)(x)(y)(z)

Code for uncurried7 function:

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 =  
 (s:T1, t:T2, u:T3, v:T4, w:T5, x:T6, y:T7) => (z:T8) => f(s)(t)(u)(v)(w)(x)(y)(z)

Movie.Scala:

Code for MoviesProtocol Object:

implicit val *nameJsonFormat* = jsonFormat4(Name.apply)  
implicit val *ratingJsonFormat* = jsonFormat2(Rating.apply)  
implicit val *principalJsonFormat* = jsonFormat2(Principal.apply)  
implicit val *reviewsJsonFormat* = jsonFormat7(Reviews.apply)  
implicit val *productionJsonFormat* = jsonFormat4(Production.apply)  
implicit val *formatJsonFormat* = jsonFormat4(Format.apply)  
implicit val *movieJsonFormat* = jsonFormat11(Movie.apply)

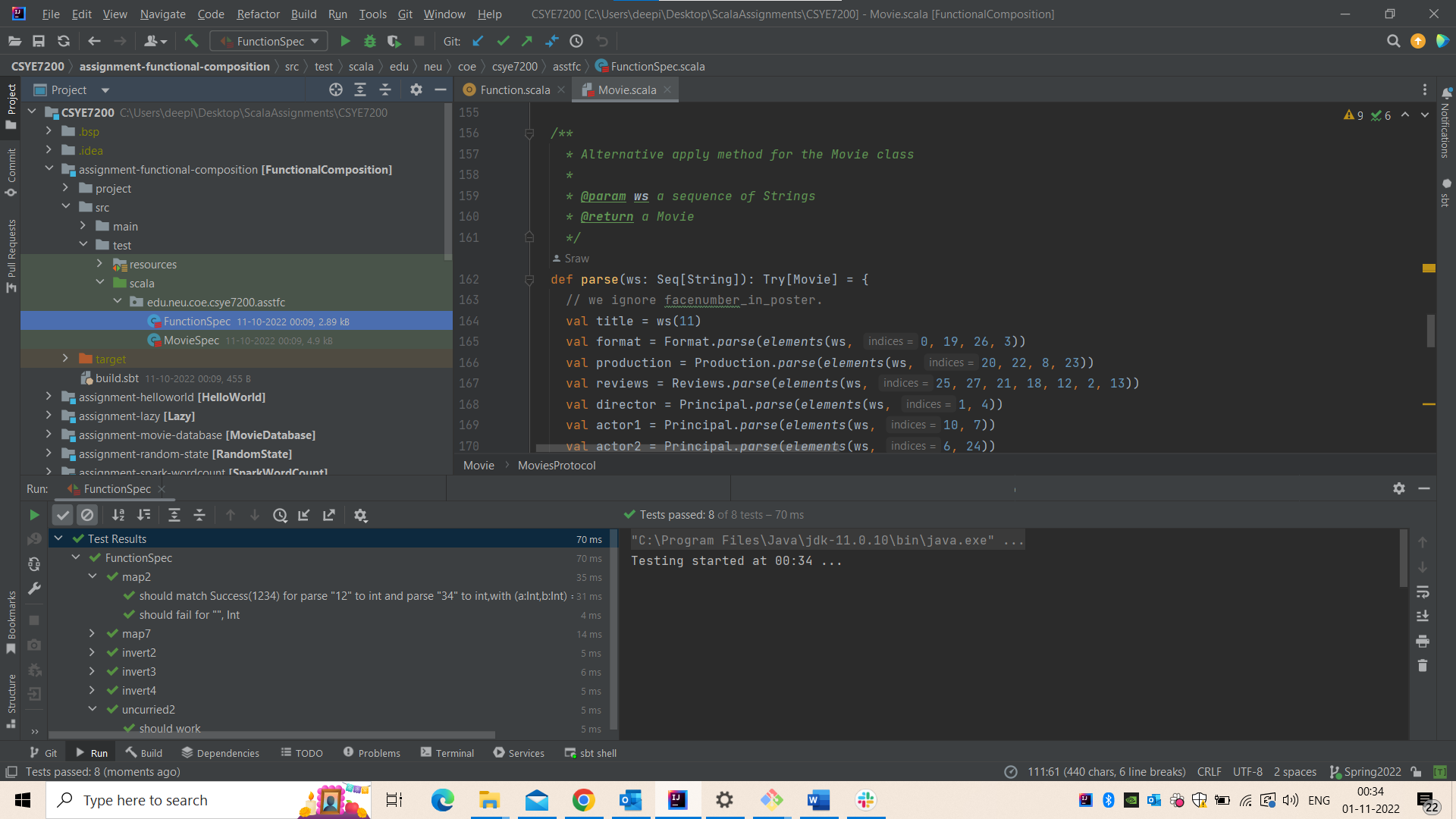
Code for testSerializationAndDeserialization:

ms.toJson.convertTo[Seq[Movie]] equals ms

**Screenshots:**

**Unit Tests Screenshot:**

**FunctionSpec UnitTest run:**

****

**MovieSpec UnitTest run:**

**A screenshot of a computer

Description automatically generated with medium confidence**