Sai Sumana Puppala – SEC01 (NUID 002925158)

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



**-List of Tasks Implemented**

Implemented the 15 code changes.

Github Link:

https://github.com/puppala-sumana/CSYE7200/tree/Spring2022/assignment-functional-composition

**-Code**

def map2[T1, T2, R](t1y: Try[T1], t2y: Try[T2])(f: (T1, T2) => R): Try[R] = t1y flatMap (t1 => t2y map (t2 => 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] =  
 t1y flatMap (t1 => t2y flatMap(t2 => t3y map( t3 => f(t1,t2,t3))))// TO BE IMPLEMENTED

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] = t1y flatMap(t1 => t2y flatMap(t2 => t3y flatMap( t3 => t4y flatMap( t4 => t5y flatMap(t5 => t6y flatMap( t6 => t7y map( t7 => 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)  
 //(x,y) => x flatMap( a => y map( b => f(a,b))) // TO BE IMPLEMENTED

def lift3[T1, T2, T3, R](f: (T1, T2, T3) => R): (Try[T1], Try[T2], Try[T3]) => Try[R] = *map3*(\_, \_, \_)(f)  
 //(x,y,z) => x flatMap(a => y flatMap(b => z map(c => f(a,b,c)))) // 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)  
 //(x1, x2, x3, x4, x5, x6, x7) => x1 flatMap(a => x2 flatMap(b => x3 flatMap(c => x4 flatMap(d => x5 flatMap(e => x6 flatMap(g => x7 map(h => f(a,b,c,d,e,g,h)))))))) // TO BE IMPLEMENTE

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

def invert3[T1, T2, T3, R](f: T1 => T2 => T3 => R): T3 => T2 => T1 => R = {  
// val g = invert2((x: T2) => (y: T3) => (r: T1)) ; f(g)  
 (z: T3) => (y: T2) => (x: T1) => f(x)(y)(z)  
 } // TO BE IMPLEMENTED

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

def uncurried2[T1, T2, T3, R](f: T1 => T2 => T3 => R): (T1, T2) => T3 => R = (x:T1, y:T2) => f(x)(y)// TO BE IMPLEMENTED

def uncurried3[T1, T2, T3, T4, R](f: T1 => T2 => T3 => T4 => R): (T1, T2, T3) => T4 => R = (x,y,z) => f(x)(y)(z) // TO BE IMPLEMENTED

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 =  
 (a,b,c,x,y,z,w) => f(a)(b)(c)(x)(y)(z)(w) // TO BE IMPLEMENTED

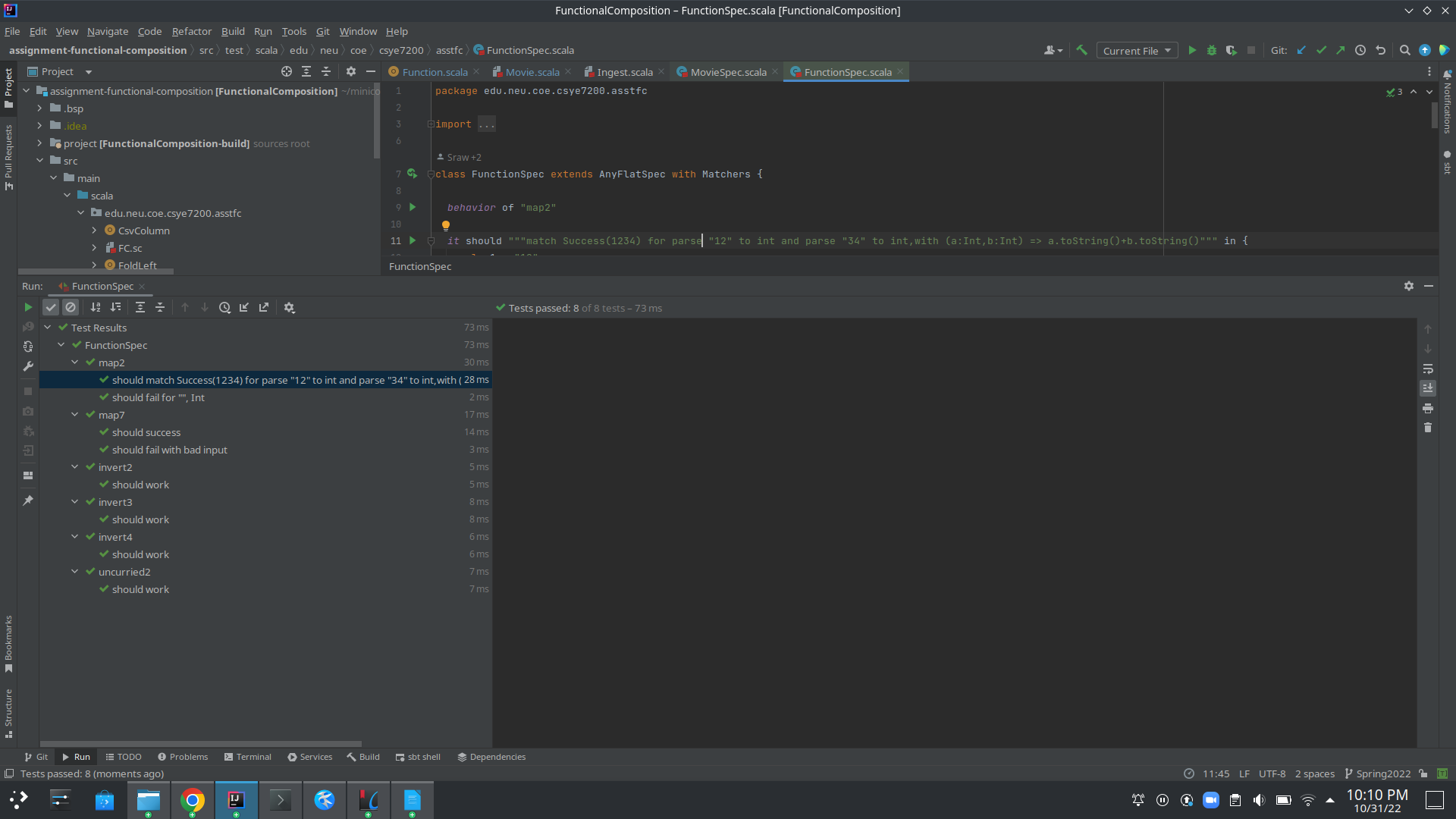
Movie.scala:

object MoviesProtocol extends DefaultJsonProtocol {  
 // 20 points  
 // TO BE IMPLEMENTED  
 implicit val *formatJ*: RootJsonFormat[Format] = jsonFormat4(Format.apply)  
 implicit val *productionJ*: RootJsonFormat[Production] = jsonFormat4(Production.apply)  
 implicit val *ratingsJ*: RootJsonFormat[Rating] = jsonFormat2(Rating.apply)  
 implicit val *reviewsJ*: RootJsonFormat[Reviews] = jsonFormat7(Reviews.apply)  
 implicit val *nameJ*: RootJsonFormat[Name] = jsonFormat4(Name.apply)  
 implicit val *principalJ*: RootJsonFormat[Principal] = jsonFormat2(Principal.apply)  
 implicit val *movieJ*: RootJsonFormat[Movie] = jsonFormat11(Movie.apply)  
 def serializeAndDeserialize(m: Movie) = m.toJson.convertTo[Movie]  
}

def testSerializationAndDeserialization(ms: Seq[Movie]): Boolean = {  
 // 5 points  
 // TO BE IMPLEMENTED  
 for( m <- ms) { val mj = MoviesProtocol.*serializeAndDeserialize*(m)  
 if(!m.equals(`mj`)) false  
 }  
 true  
}

**-Unit tests**

FunctionSpec.scala:



MovieSpec.scala:

