Sayeed Ahmed – SEC01 (NUID 002191535)

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

Implementation:

Movie.scala:

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

```scala
//Hint: Serialize the input to Json format and deserialize back to Object,
Sraw *
def testSerializationAndDeserialization(ms: Seq[Movie]): Boolean = {
  // 5 points
  // TO BE IMPLEMENTED
  import MoviesProtocol._
  ms.forall { m =>
    val json = m.toJson
    val deserialized = json.convertTo[Movie]
    deserialized == m
  }
}
```

Function.scala:

```scala
new *
def map2[T1, T2, R](t1y: Try[T1], t2y: Try[T2])(f: (T1, T2) => R): Try[R] = for (x <- t1y; y <- t2y) yield  f(x,y)
```

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

```scala
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:
                                        (f: (T1, T2, T3, T4, T5, T6, T7) => R): Try[R] =
  for (x <- t1y; y <- t2y; z <- t3y; a <- t4y; b <- t5y; c <- t6y; d <- t7y) yield  f(x,y,z,a,b,c,d) // TO BE IMPLEMENTED
```

```scala
new *
def lift[T, R](f: T => R): Try[T] => Try[R] = _ map f
```

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

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

```scala
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)
```

```scala
def invert2[T1, T2, R](f: T1 => T2 => R): T2 => T1 => R = t2 => t1 => f(t1)(t2)
```

```scala
def invert3[T1, T2, T3, R](f: T1 => T2 => T3 => R): T3 => T2 => T1 => R = t3 => t2 => t1 => f(t1)(t2)(t3)
```

```scala
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
```

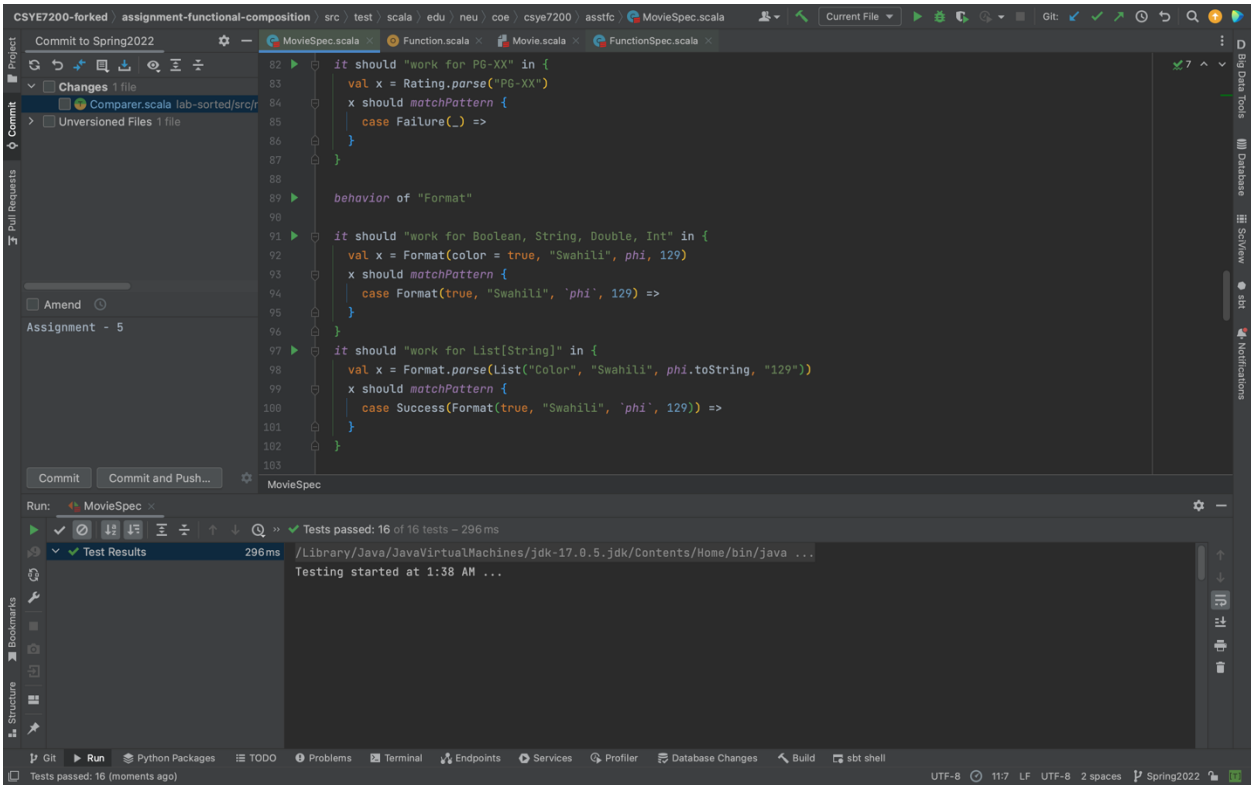
```scala
def uncurried2[T1, T2, T3, R](f: T1 => T2 => T3 => R): (T1, T2) => T3 => R = (t1: T1, t2: T2) => (t3: T3) => f(t1)(t2)(t3)
```

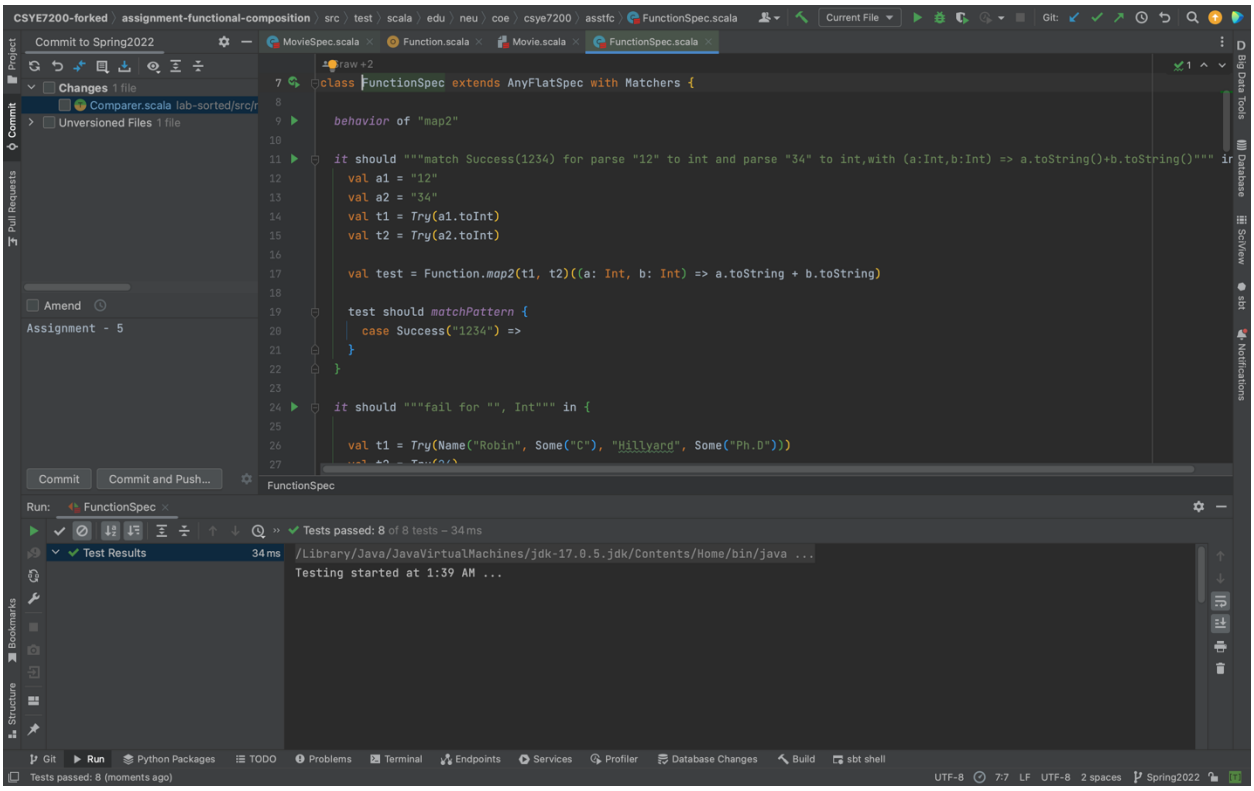
```scala
def uncurried3[T1, T2, T3, T4, R](f: T1 => T2 => T3 => T4 => R): (T1, T2, T3) => T4 => R =
  (t1: T1, t2: T2, t3: T3) => (t4: T4) => f(t1)(t2)(t3)(t4) // TO BE IMPLEMENTED
```

```scala
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,
  (t1: T1, t2: T2, t3: T3, t4: T4, t5: T5, t6: T6, t7: T7) => (t8: T8) => f(t1)(t2)(t3)(t4)(t5)(t6)(t7)(t8)// TO BE IMPLEMENTED
```

github link: https://github.com/sayeedahmed01/CSYE7200/tree/Spring2022/assignment-functional-composition/src/main/scala/edu/neu/coe/csye7200/asstfc

Test:


CSYE7200-forked › assignment-functional-composition › src › test › scala › edu › neu › coe › csye7200 › asstfc › MovieSpec.scala
Commit to Spring2022
MovieSpec.scala
Function.scala
Movie.scala
FunctionSpec.scala
Changes 1 file
Comparer.scala lab-sorted/src/r
Unversioned Files 1 file
Amend
Assignment - 5
Commit
Commit and Push...
it should "work for PG-XX" in {
val x = Rating.parse("PG-XX")
x should matchPattern {
case Failure(_) =>
}
}
behavior of "Format"
it should "work for Boolean, String, Double, Int" in {
val x = Format(color = true, "Swahili", phi, 129)
x should matchPattern {
case Format(true, "Swahili", `phi`, 129) =>
}
}
it should "work for List[String]" in {
val x = Format.parse(List("Color", "Swahili", phi.toString, "129"))
x should matchPattern {
case Success(Format(true, "Swahili", `phi`, 129)) =>
}
}
MovieSpec
Run: MovieSpec
Tests passed: 16 of 16 tests – 296 ms
Test Results 296 ms
/Library/Java/JavaVirtualMachines/jdk-17.0.5.jdk/Contents/Home/bin/java ...
Testing started at 1:38 AM ...
Git
Run
Python Packages
TODO
Problems
Terminal
Endpoints
Services
Profiler
Database Changes
Build
sbt shell
Tests passed: 16 (moments ago)
UTF-8
11:7
LF
UTF-8
2 spaces
Spring2022


CSYE7200-forked › assignment-functional-composition › src › test › scala › edu › neu › coe › csye7200 › asstfc › FunctionSpec.scala
Commit to Spring2022
MovieSpec.scala
Function.scala
Movie.scala
FunctionSpec.scala
Changes 1 file
Comparer.scala lab-sorted/src/r
Unversioned Files 1 file
Amend
Assignment - 5
Commit
Commit and Push...
class FunctionSpec extends AnyFlatSpec with Matchers {
behavior of "map2"
it should """match Success(1234) for parse "12" to int and parse "34" to int,with (a:Int,b:Int) => a.toString()+b.toString()""" in {
val a1 = "12"
val a2 = "34"
val t1 = Try(a1.toInt)
val t2 = Try(a2.toInt)
val test = Function.map2(t1, t2)((a: Int, b: Int) => a.toString + b.toString)
test should matchPattern {
case Success("1234") =>
}
}
it should """fail for "", Int""" in {
val t1 = Try(Name("Robin", Some("C"), "Hillyard", Some("Ph.D")))
FunctionSpec
Run: FunctionSpec
Tests passed: 8 of 8 tests – 34 ms
Test Results 34 ms
/Library/Java/JavaVirtualMachines/jdk-17.0.5.jdk/Contents/Home/bin/java ...
Testing started at 1:39 AM ...
Git
Run
Python Packages
TODO
Problems
Terminal
Endpoints
Services
Profiler
Database Changes
Build
sbt shell
Tests passed: 8 (moments ago)
UTF-8
7:7
LF
UTF-8
2 spaces
Spring2022