Anirudha Joshi – SEC01 (NUID 002991365)

Big Data System Engineering with Scala
Spring 2023
Assignment No. 3 (Movie Assignment)

GitHub - https://github.com/anirudhajoshi2808/Anirudha_Joshi_CSYE7200

- Tasks:

There are three implementations to create in *Movie.scala* (**lines 101, 124, 204**). This time it is OK to edit the Spec file (but after this assignment, you should not edit any Spec files). But I want to give you some familiarity with Spec files (i.e. ScalaTest with Matchers). The most useful documentation for this sort of thing is here: http://www.scalatest.org/user_guide/using_matchers.

The purpose of this code is to read the movie database file into *Movie* instances.

The module name for this assignment is *assignment-movie-database*. Let me clarify a little: you only need to work with *Ingest.scala, Movie.scala* and *IngestSpec.scala,* in addition, *MovieSpec.scala* is available for you to verify your implementation. These are all in the *asstmd* package (the spec file is under test of course). You will also need *movie_metadata.csv* which is under test/resources of the project (use this version).

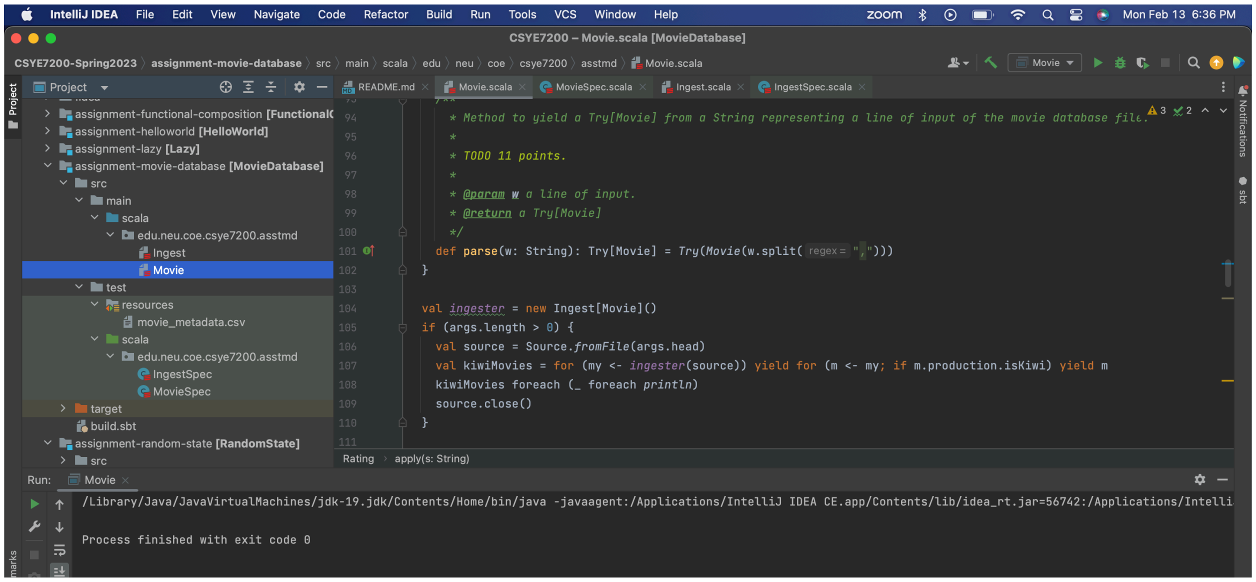
You don't need to create a new project, just clone/download class repo, and import assignment-movie-database, you may follow the video demo under *Canvas / Assignments / Video Demo for Assignment Import and Submit*. Remember the to follow the *Canvas / Assignments / Standard procedure for submitting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

You will have to do some real programming to create these implementations and you may feel that you aren't quite ready for it. But, the compiler is your friend. The REPL is your friend. **You can do it!** And the principle of **Simple, Obvious, Elegant** is your friend too (I particularly commend my blog article Links to an external site. on this).

- Code: Movie.scala

Line 101

def parse(w: String): Try[Movie] = *Try*(*Movie*(w.split(",")))

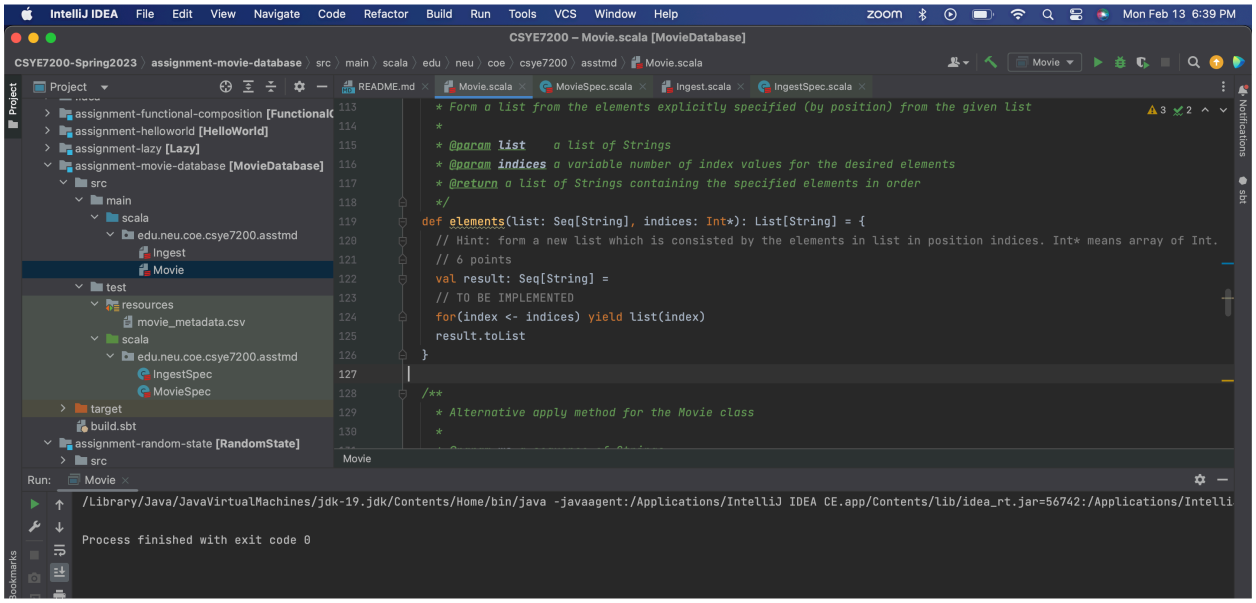


```
94     * Method to yield a Try[Movie] from a String representing a line of input of the movie database file.
95     *
96     * TODO 11 points.
97     *
98     * @param w a line of input.
99     * @return a Try[Movie]
100    */
101   def parse(w: String): Try[Movie] = Try(Movie(w.split(",")))
102  }
103
104  val ingester = new Ingest[Movie]()
105  if (args.length > 0) {
106    val source = Source.fromFile(args.head)
107    val kiwiMovies = for (my <- ingester(source)) yield for (m <- my; if m.production.isKiwi) yield m
108    kiwiMovies foreach (_ foreach println)
109    source.close()
110  }
111
```

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=56742:/Applications/Intelli

Process finished with exit code 0
```

Line 124:

val result: Seq[String] =
for(index <- indices) yield list(index)
result.toList



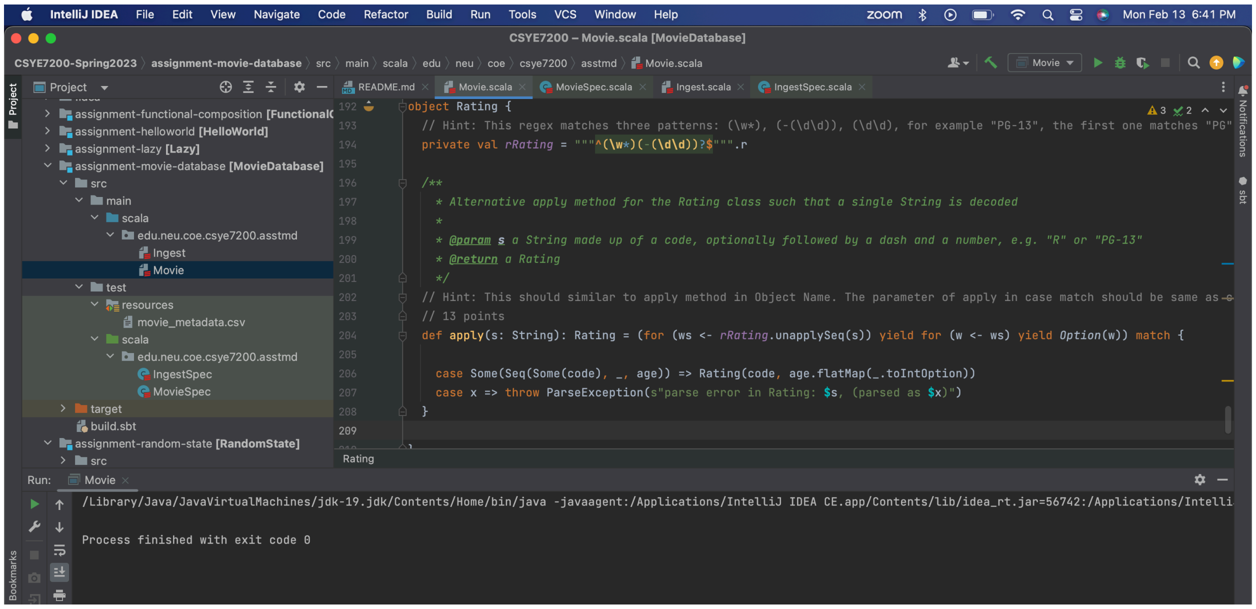
```
113    * Form a list from the elements explicitly specified (by position) from the given list
114    *
115    * @param list    a list of Strings
116    * @param indices a variable number of index values for the desired elements
117    * @return a list of Strings containing the specified elements in order
118    */
119  def elements(list: Seq[String], indices: Int*): List[String] = {
120    // Hint: form a new list which is consisted by the elements in list in position indices. Int* means array of Int.
121    // 6 points
122    val result: Seq[String] =
123    // TO BE IMPLEMENTED
124    for(index <- indices) yield list(index)
125    result.toList
126  }
127
128  /**
129    * Alternative apply method for the Movie class
130    *
```

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=56742:/Applications/Intelli

Process finished with exit code 0
```

Line 204:

def apply(s: String): Rating = (for (ws <- *rRating*.unapplySeq(s)) yield for (w <- ws) yield *Option*(w)) match {

  case Some(Seq(Some(code), _, age)) => Rating(code, age.flatMap(_.toIntOption))
  case x => throw ParseException(s"parse error in Rating: $s, (parsed as $x)")
}

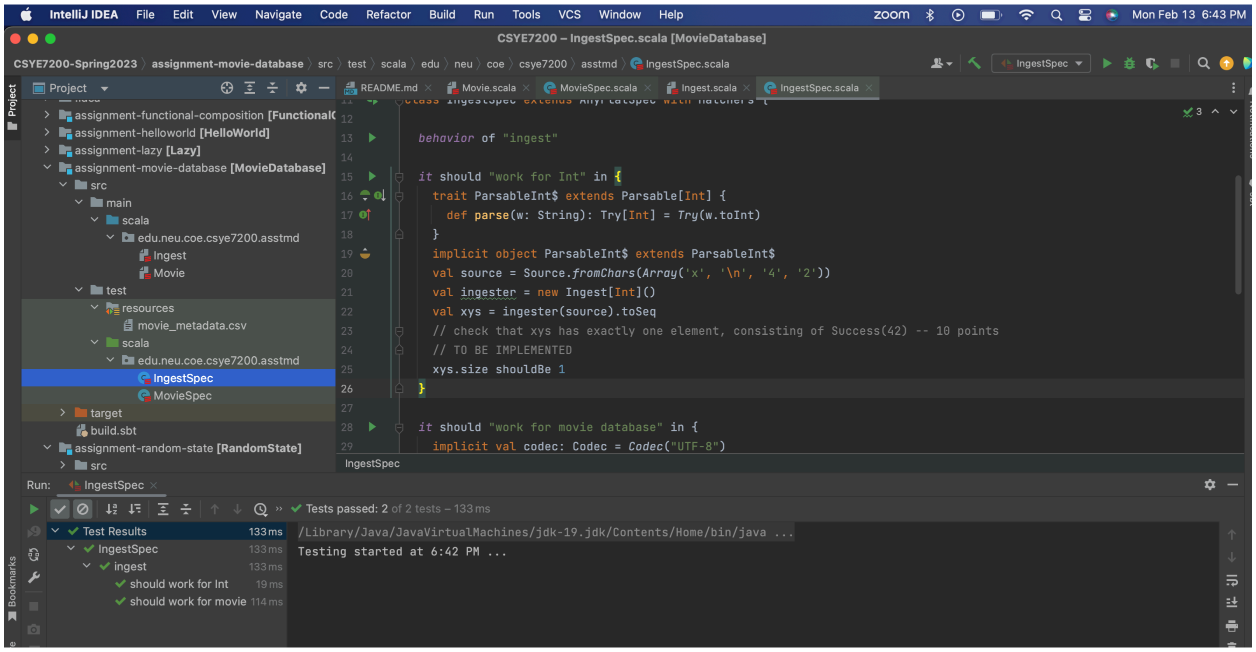


CSYE7200 – Movie.scala [MovieDatabase]

```
192  object Rating {
193    // Hint: This regex matches three patterns: (\w*), (-(\d\d)), (\d\d), for example "PG-13", the first one matches "PG"
194    private val rRating = """^(\w*)(-(\d\d))?$""".r
195
196    /**
197      * Alternative apply method for the Rating class such that a single String is decoded
198      *
199      * @param s a String made up of a code, optionally followed by a dash and a number, e.g. "R" or "PG-13"
200      * @return a Rating
201      */
202    // Hint: This should similar to apply method in Object Name. The parameter of apply in case match should be same as
203    // 13 points
204    def apply(s: String): Rating = (for (ws <- rRating.unapplySeq(s)) yield for (w <- ws) yield Option(w)) match {
205
206      case Some(Seq(Some(code), _, age)) => Rating(code, age.flatMap(_.toIntOption))
207      case x => throw ParseException(s"parse error in Rating: $s, (parsed as $x)")
208    }
209
```

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=56742:/Applications/Intelli

Process finished with exit code 0
```

IngestSpec.scala:

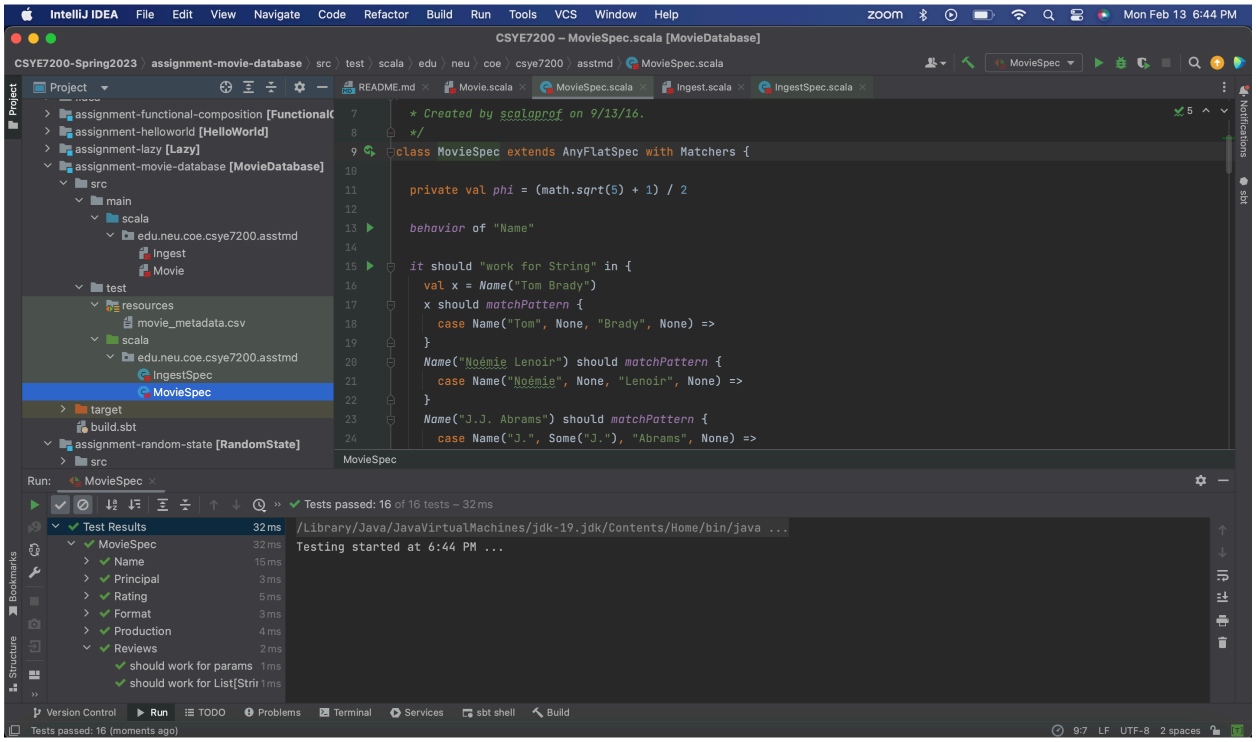


CSYE7200 – IngestSpec.scala [MovieDatabase]

```
  behavior of "ingest"

  it should "work for Int" in {
    trait ParsableInt$ extends Parsable[Int] {
      def parse(w: String): Try[Int] = Try(w.toInt)
    }
    implicit object ParsableInt$ extends ParsableInt$
    val source = Source.fromChars(Array('x', '\n', '4', '2'))
    val ingester = new Ingest[Int]()
    val xys = ingester(source).toSeq
    // check that xys has exactly one element, consisting of Success(42) -- 10 points
    // TO BE IMPLEMENTED
    xys.size shouldBe 1
  }

  it should "work for movie database" in {
    implicit val codec: Codec = Codec("UTF-8")
```

```
Tests passed: 2 of 2 tests – 133 ms

Test Results                    133 ms
  IngestSpec                    133 ms
    ingest                      133 ms
      should work for Int        19 ms
      should work for movie     114 ms

/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java ...
Testing started at 6:42 PM ...
```

- Unit tests



```
  * Created by scalaprof on 9/13/16.
  */
class MovieSpec extends AnyFlatSpec with Matchers {

  private val phi = (math.sqrt(5) + 1) / 2

  behavior of "Name"

  it should "work for String" in {
    val x = Name("Tom Brady")
    x should matchPattern {
      case Name("Tom", None, "Brady", None) =>
    }
    Name("Noémie Lenoir") should matchPattern {
      case Name("Noémie", None, "Lenoir", None) =>
    }
    Name("J.J. Abrams") should matchPattern {
      case Name("J.", Some("J."), "Abrams", None) =>
```

Tests passed: 16 of 16 tests – 32 ms

```
Test Results                     32 ms
  MovieSpec                      32 ms
    Name                         15 ms
    Principal                     3 ms
    Rating                        5 ms
    Format                        3 ms
    Production                    4 ms
    Reviews                       2 ms
      should work for params      1 ms
      should work for List[Stri   1 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java ...
Testing started at 6:44 PM ...
```

Tests passed: 16 (moments ago)