Harshil Shah – SEC01 (NUID 002780887)

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

- GitHub Repo URL - https://github.com/harshilshahneu/CSYE7200-Harshil-Shah

---

- List of Tasks Implemented

- 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).

- Code

Movie.Scala

```scala
implicit object ParsableMovie extends Parsable[Movie] {
  /**
    * Method to yield a Try[Movie] from a String representing a line of input of the movie database file.
    *
    * TODO 11 points.
    *
    * @param w a line of input.
    * @return a Try[Movie]
    */
  def parse(w: String): Try[Movie] = Try(Movie(w.split(",")))
}
```

```scala
/**
  * Form a list from the elements explicitly specified (by position) from the given list
  *
  * @param list    a list of Strings
  * @param indices a variable number of index values for the desired elements
  * @return a list of Strings containing the specified elements in order
  */
def elements(list: Seq[String], indices: Int*): List[String] = {
  // Hint: form a new list which is consisted by the elements in list in position indices. Int* means array of Int.
  // 6 points
  val result: Seq[String] = for (i <- indices) yield list(i)
  result.toList
}
```

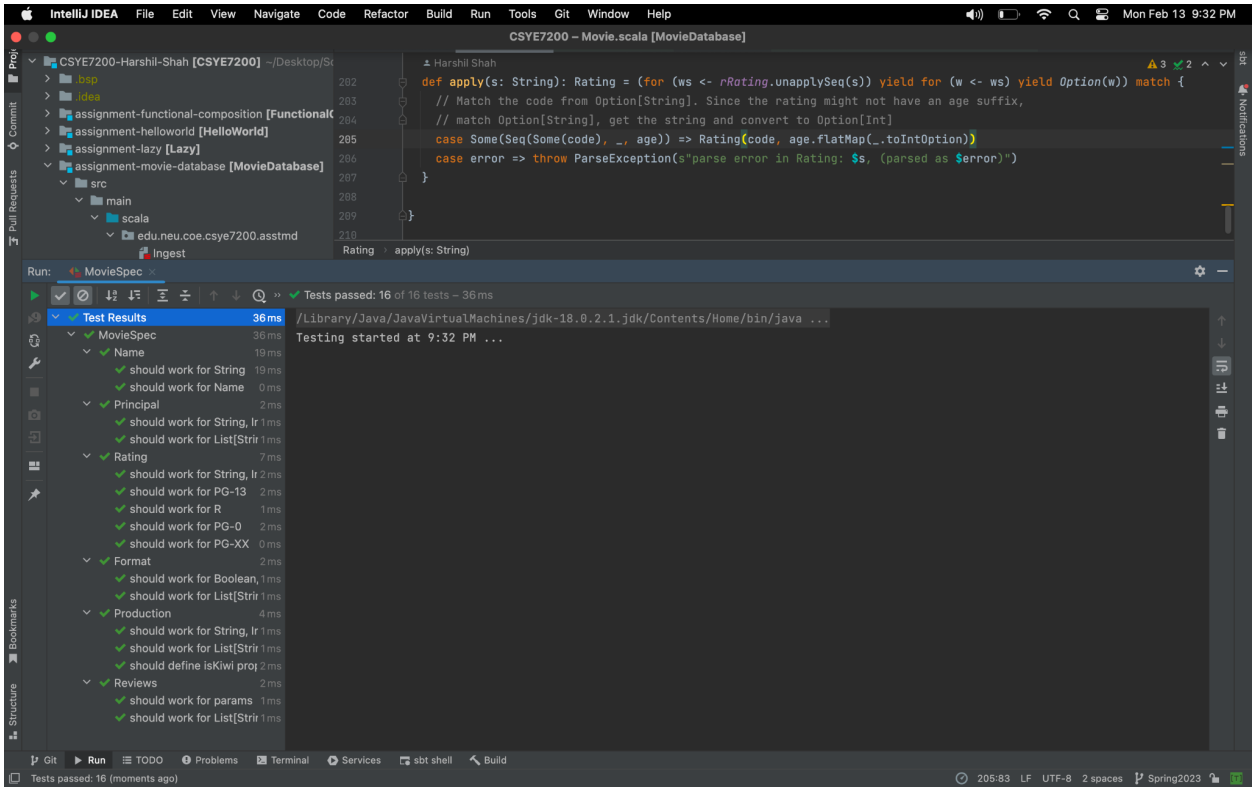
```scala
object Rating {
  // Hint: This regex matches three patterns: (\w*), (-(\d\d)), (\d\d), for example "PG-13", the first one matches "PG", second one "-13", third one "13".
  private val rRating = """^(\w*)(-(\d\d))?$""".r

  /**
   * Alternative apply method for the Rating class such that a single String is decoded
   *
   * @param s a String made up of a code, optionally followed by a dash and a number, e.g. "R" or "PG-13"
   * @return a Rating
   */
  // Hint: This should similar to apply method in Object Name. The parameter of apply in case match should be same as case class Rating
  // 13 points
  def apply(s: String): Rating = (for (ws <- rRating.unapplySeq(s)) yield for (w <- ws) yield Option(w)) match {
    // Match the code from Option[String]. Since the rating might not have an age suffix,
    // match Option[String], get the string and convert to Option[Int]
    case Some(Seq(Some(code), _, age)) => Rating(code, age.flatMap(_.toIntOption))
    case error => throw ParseException(s"parse error in Rating: $s, (parsed as $error)")
  }

}
```

IngestSpec.scala

```scala
  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
    xys.length shouldBe 1
    xys(0) shouldBe Success(42)
  }
```

- Unit tests

MovieSpec.scala



IngestSpec.scala