Sayeed Ahmed – SEC01 (NUID 002191535)

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

- **List of Tasks Implemented:**
  1) Implemented the parse method to create Movie objects from a csv string.
  2) Implemented the elements method which mapped the sequence of string to the indices.
  3) Implemented the apply method for the companion object of the Ratings class that creates Rating objects from a string.
  4) Completed the test case to the functionality of class Ingest for parsing integers from a source of characters

- **Github Repo:**
  https://github.com/sayeedahmed01/CSYE7200/tree/Spring2022/assignment-movie-database

- **Code:**

1)

```
object Movie extends App {

  Robin Hillyard *
  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]
      */
    new *
    def parse(w: String): Try[Movie] = Try{
    Movie(w.split( regex = ",").toSeq)
    }
  }
```

2)

```
  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] = indices.map(list(_))
    result.toList
```

3)

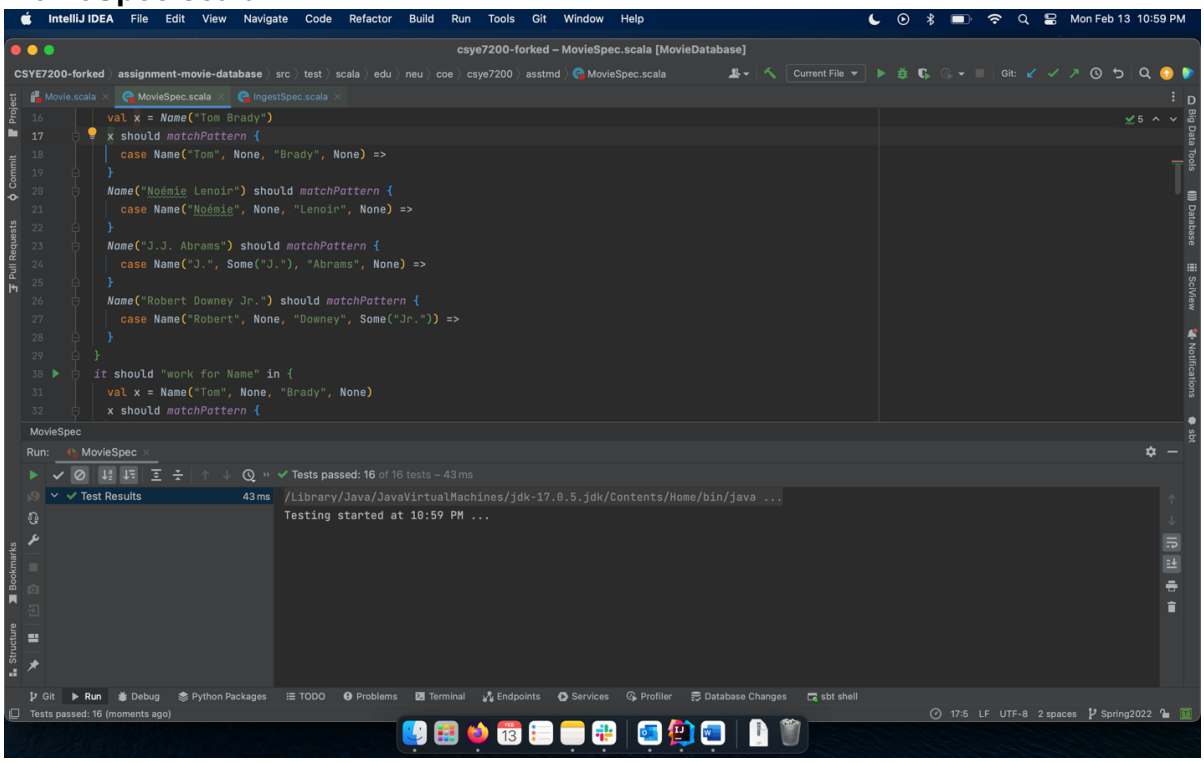
```
object Rating {
  // Hint: This regex matches three patterns: (\w*), (-(\d\d)), (\d\d), for example "PG-13", the first one ma
  private val rRating = """^(\w*)(-(\d\d))?$""".r

  /**
    * 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
    */
  new *
  def apply(s: String): Rating = s match{
    case rRating(code, _, age) => Rating(code, Option(age).map((_.toInt)))
    case rRating(code,_, null) => Rating(code,None)
    case _ => throw ParseException(s"parse error in Rating: $s")
  }
}
```

4)

```
  it should "work for Int" in {
    rohanchoudhary +1
    trait ParsableInt$ extends Parsable[Int] {
      rohanchoudhary
      def parse(w: String): Try[Int] = Try(w.toInt)
    }
    rohanchoudhary
    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
    xys(0) shouldBe Success(42)
  }
```

- **Unit Tests:**
  1) **MovieSpec.scala:**



  2) **IngestSpec.scala:**