Hrishikesh Sanjay Pawar – SEC01 (NUID 002707307)

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

-List of Tasks Implemented

Movie.scala:

Implemented the method to yield a Try[Movie] from a String representing a line of input of the movie database file.

Implemented the method to form a list from the elements explicitly specified (by position) from the given list.

Implemented an alternative apply method for the Rating class such that a single String is decoded.

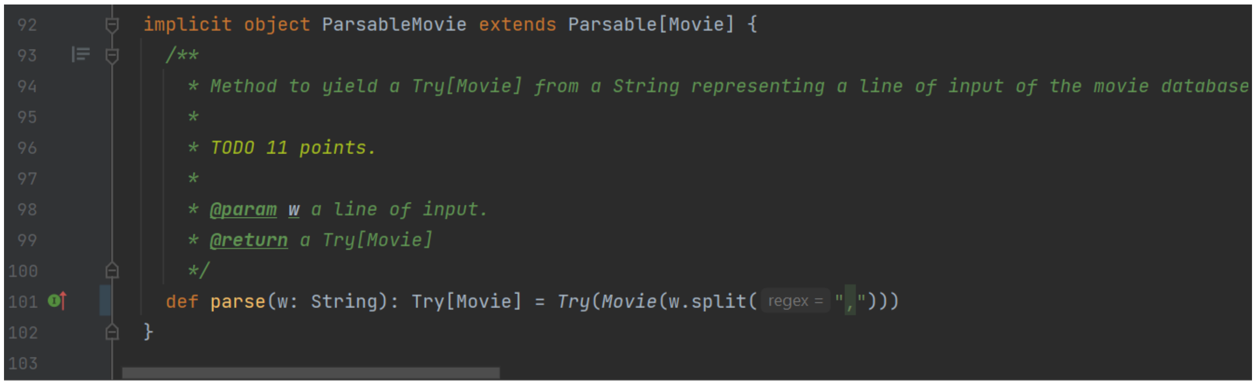
IngestSpec.scala:

Checked whether xys has exactly one element, consisting of Success(42)

-Code

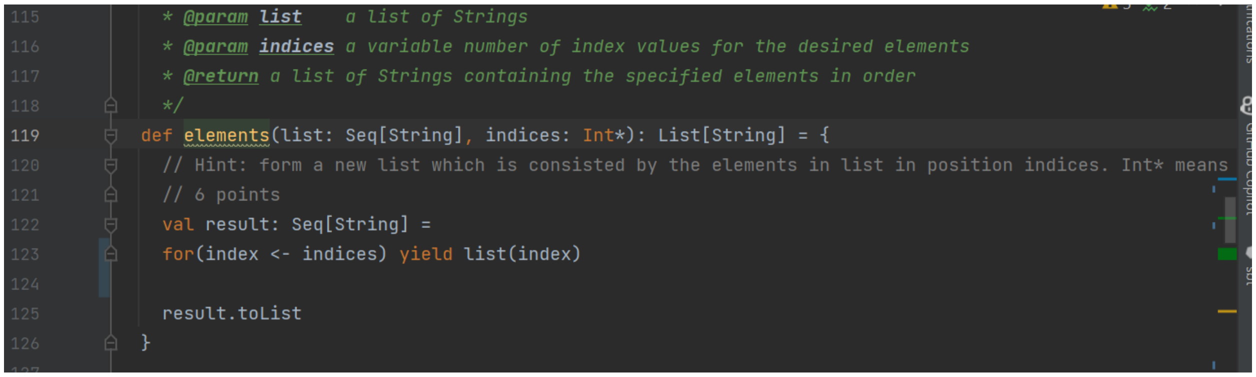
Movie.scala:

Line 101:



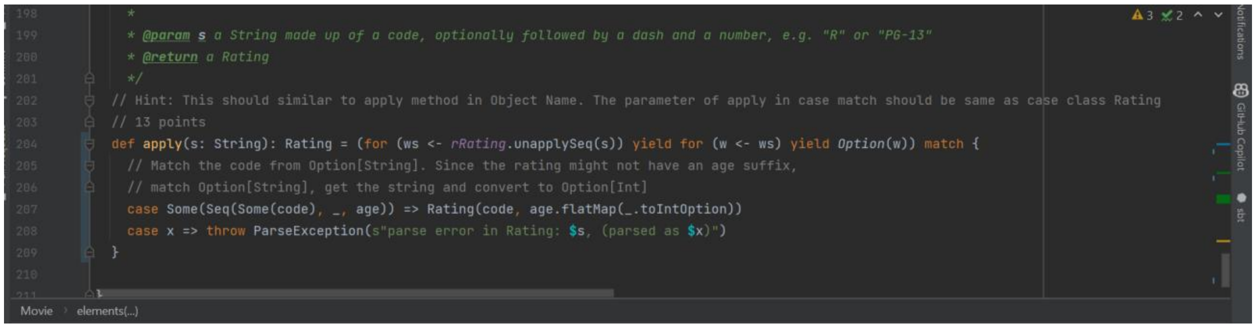
```
92   implicit object ParsableMovie extends Parsable[Movie] {
93     /**
94       * Method to yield a Try[Movie] from a String representing a line of input of the movie database
95       *
96       * TODO 11 points.
97       *
98       * @param w a line of input.
99       * @return a Try[Movie]
100      */
101    def parse(w: String): Try[Movie] = Try(Movie(w.split( regex = ",")))
102  }
103
```

Line 124:



```
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
121    // 6 points
122    val result: Seq[String] =
123    for(index <- indices) yield list(index)
124
125    result.toList
126  }
```

Line 204:

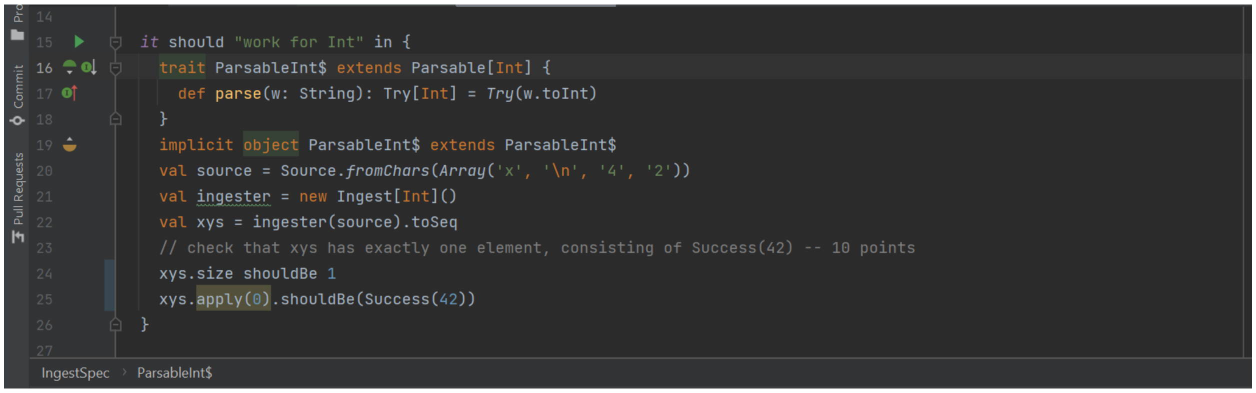


```
198      *
199      * @param s a String made up of a code, optionally followed by a dash and a number, e.g. "R" or "PG-13"
200      * @return a Rating
201      */
202    // Hint: This should similar to apply method in Object Name. The parameter of apply in case match should be same as case class Rating
203    // 13 points
204    def apply(s: String): Rating = (for (ws <- rRating.unapplySeq(s)) yield for (w <- ws) yield Option(w)) match {
205      // Match the code from Option[String]. Since the rating might not have an age suffix,
206      // match Option[String], get the string and convert to Option[Int]
207      case Some(Seq(Some(code), _, age)) => Rating(code, age.flatMap(_.toIntOption))
208      case x => throw ParseException(s"parse error in Rating: $s, (parsed as $x)")
209    }
210
```

Movie › elements(...)

IngestSpec.scala:

Line 24:

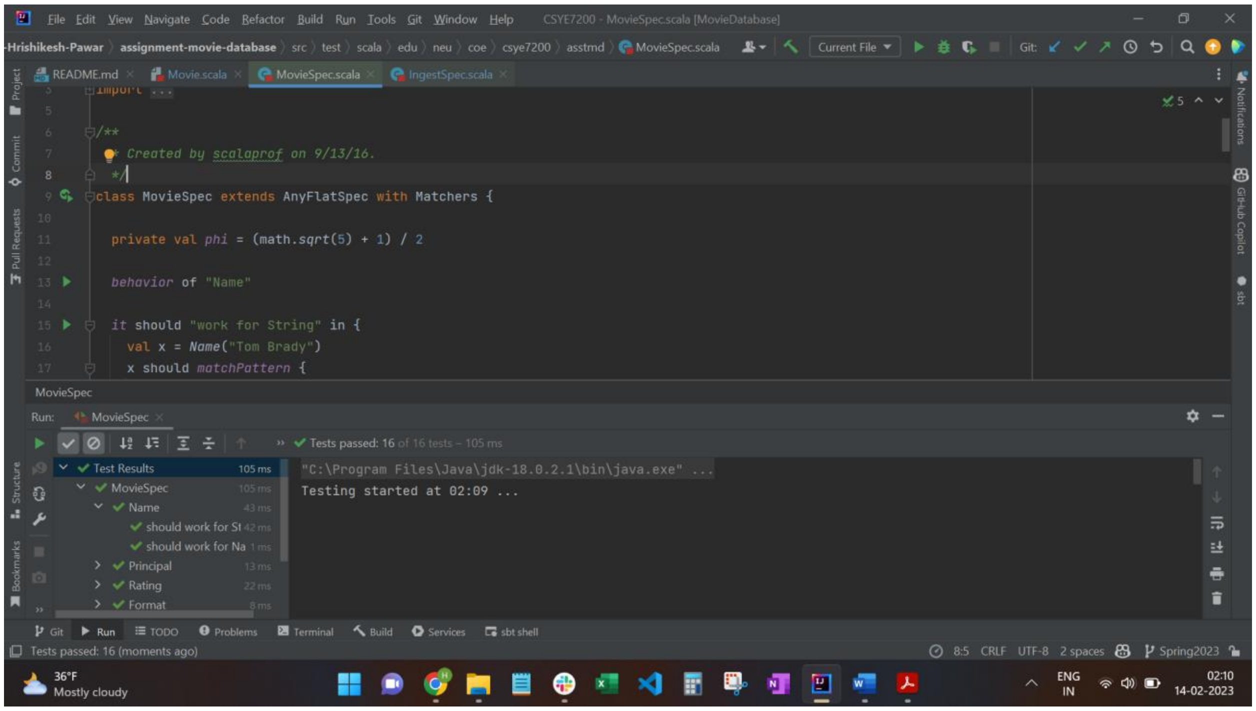


```
14
15    it should "work for Int" in {
16      trait ParsableInt$ extends Parsable[Int] {
17        def parse(w: String): Try[Int] = Try(w.toInt)
18      }
19      implicit object ParsableInt$ extends ParsableInt$
20      val source = Source.fromChars(Array('x', '\n', '4', '2'))
21      val ingester = new Ingest[Int]()
22      val xys = ingester(source).toSeq
23      // check that xys has exactly one element, consisting of Success(42) -- 10 points
24      xys.size shouldBe 1
25      xys.apply(0).shouldBe(Success(42))
26    }
27
```

IngestSpec › ParsableInt$

-Unit tests

MovieSpec.scala:



```
import ...

/**
 * 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 {
```

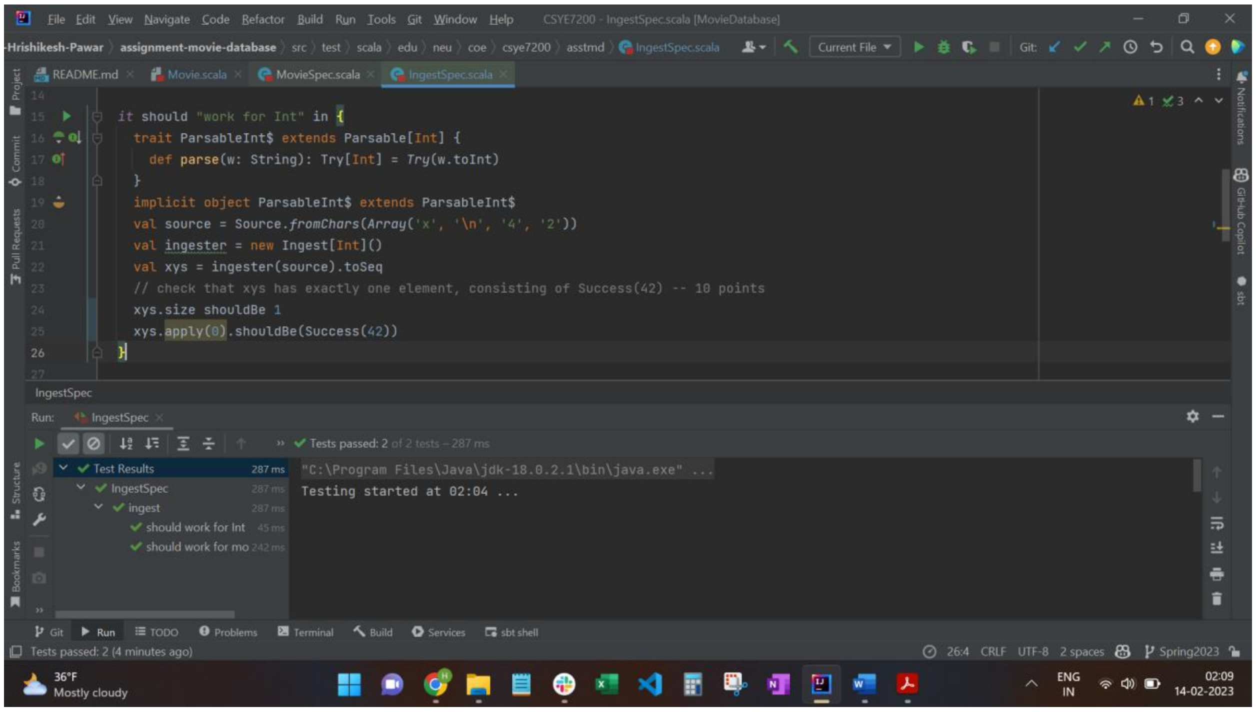
```
Tests passed: 16 of 16 tests – 105 ms

Test Results                105 ms
  MovieSpec                 105 ms
    Name                     43 ms
      should work for St     42 ms
      should work for Na      1 ms
    Principal                13 ms
    Rating                   22 ms
    Format                    8 ms

"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
Testing started at 02:09 ...

Tests passed: 16 (moments ago)
```

IngestSpec.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.size shouldBe 1
    xys.apply(0).shouldBe(Success(42))
  }
```

```
Tests passed: 2 of 2 tests – 287 ms

Test Results                287 ms
  IngestSpec                287 ms
    ingest                  287 ms
      should work for Int    45 ms
      should work for mo    242 ms

"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
Testing started at 02:04 ...

Tests passed: 2 (4 minutes ago)
```

- Result

Successfully implemented all the incomplete methods in Movie.scala and added a test case in IngestSpec.scala.

GitHub Repo URL: https://github.com/hrishikesh-pawar/CSYE7200-Hrishikesh-Pawar