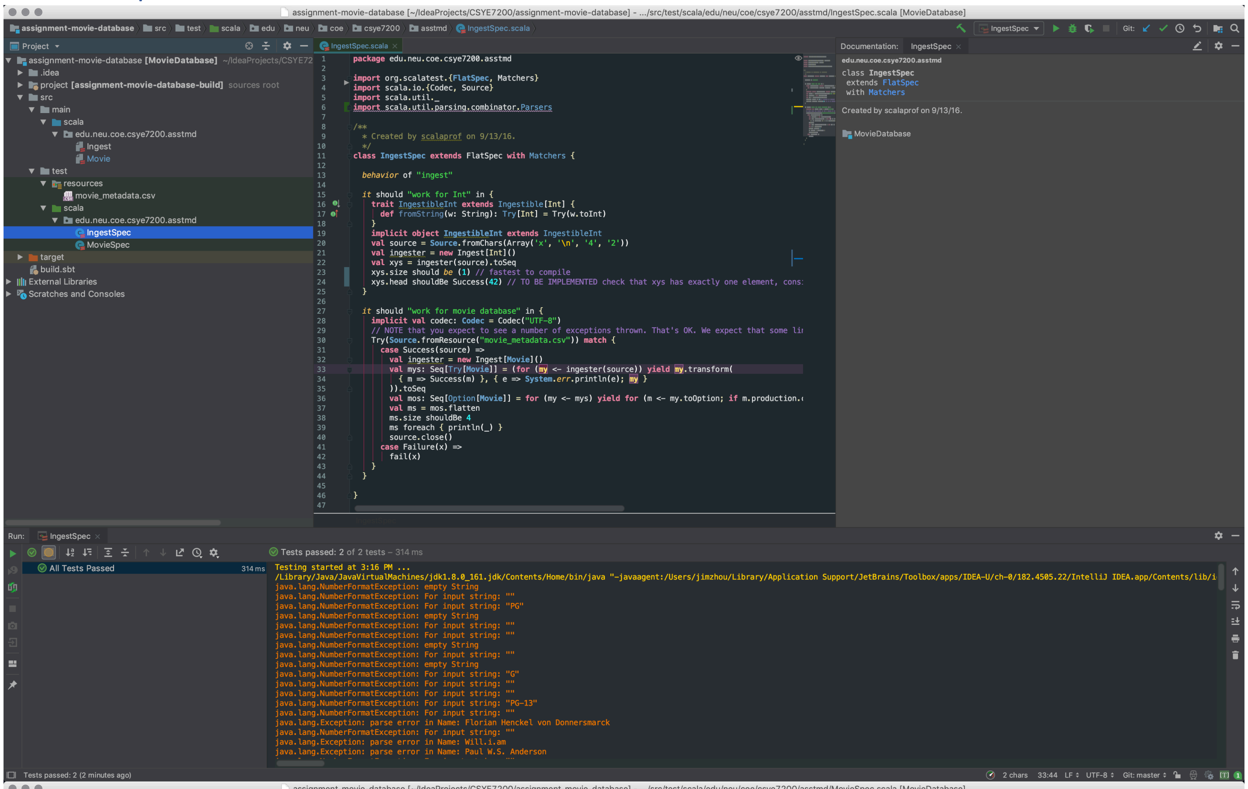
Assignment 3: Movie Database (Part 2)
Name: Qixiang Zhou
NUID: 001822974

The snapshot of unit test:



assignment-movie-database [~/IdeaProjects/CSYE7200/assignment-movie-database] - .../src/test/scala/edu/neu/coe/csye7200/asstmd/IngestSpec.scala [MovieDatabase]

```
assignment-movie-database [MovieDatabase] ~/IdeaProjects/CSYE72
  .idea
  project [assignment-movie-database-build] sources root
  src
    main
      scala
        edu.neu.coe.csye7200.asstmd
          Ingest
          Movie
    test
      resources
        movie_metadata.csv
      scala
        edu.neu.coe.csye7200.asstmd
          IngestSpec
          MovieSpec
  target
  build.sbt
External Libraries
Scratches and Consoles
```

```scala
package edu.neu.coe.csye7200.asstmd

import org.scalatest.{FlatSpec, Matchers}
import scala.io.{Codec, Source}
import scala.util._
import scala.util.parsing.combinator.Parsers

/**
  * Created by scalaprof on 9/13/16.
  */
class IngestSpec extends FlatSpec with Matchers {

  behavior of "ingest"

  it should "work for Int" in {
    trait IngestibleInt extends Ingestible[Int] {
      def fromString(w: String): Try[Int] = Try(w.toInt)
    }
    implicit object IngestibleInt extends IngestibleInt
    val source = Source.fromChars(Array('x', '\n', '4', '2'))
    val ingester = new Ingest[Int]()
    val xys = ingester(source).toSeq
    xys.size should be (1) // fastest to compile
    xys.head shouldBe Success(42) // TO BE IMPLEMENTED check that xys has exactly one element, cons:
  }

  it should "work for movie database" in {
    implicit val codec: Codec = Codec("UTF-8")
    // NOTE that you expect to see a number of exceptions thrown. That's OK. We expect that some lir
    Try(Source.fromResource("movie_metadata.csv")) match {
      case Success(source) =>
        val ingester = new Ingest[Movie]()
        val mys: Seq[Try[Movie]] = (for (my <- ingester(source)) yield my.transform(
          { m => Success(m) }, { e => System.err.println(e); my }
        )).toSeq
        val mos: Seq[Option[Movie]] = for (my <- mys) yield for (m <- my.toOption; if m.production.(
        val ms = mos.flatten
        ms.size shouldBe 4
        ms foreach { println(_) }
        source.close()
      case Failure(x) =>
        fail(x)
    }
  }

}
```

Documentation: IngestSpec

```
edu.neu.coe.csye7200.asstmd
class IngestSpec
 extends FlatSpec
 with Matchers
```

Created by scalaprof on 9/13/16.

MovieDatabase

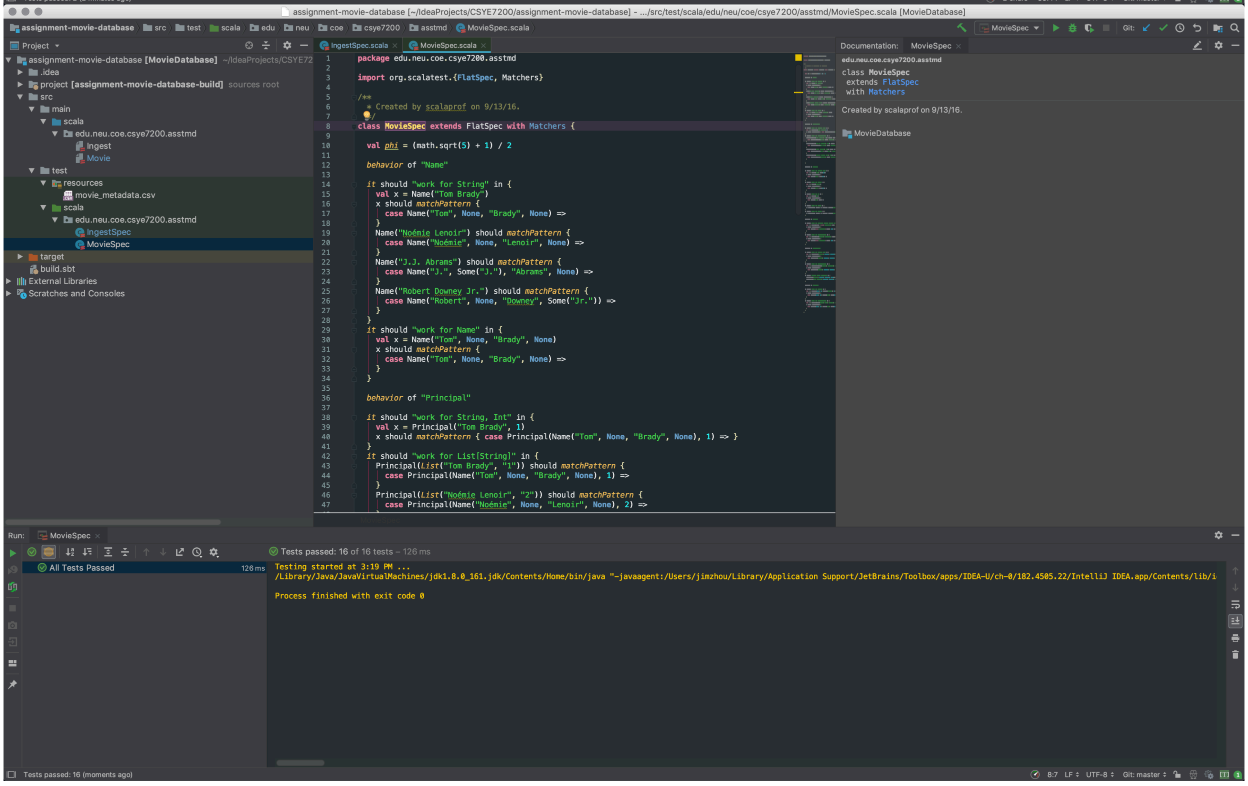
Run: IngestSpec

All Tests Passed 314 ms

Tests passed: 2 of 2 tests – 314 ms

```
Testing started at 3:16 PM ...
/Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java "-javaagent:/Users/jimzhou/Library/Application Support/JetBrains/Toolbox/apps/IDEA-U/ch-0/182.4505.22/IntelliJ IDEA.app/Contents/lib/i
java.lang.NumberFormatException: empty String
java.lang.NumberFormatException: For input string: ""
java.lang.NumberFormatException: For input string: "PG"
java.lang.NumberFormatException: empty String
java.lang.NumberFormatException: For input string: ""
java.lang.NumberFormatException: For input string: ""
java.lang.NumberFormatException: empty String
java.lang.NumberFormatException: For input string: ""
java.lang.NumberFormatException: empty String
java.lang.NumberFormatException: For input string: "G"
java.lang.NumberFormatException: For input string: ""
java.lang.NumberFormatException: For input string: ""
java.lang.NumberFormatException: For input string: "PG-13"
java.lang.NumberFormatException: For input string: ""
java.lang.Exception: parse error in Name: Florian Henckel von Donnersmarck
java.lang.NumberFormatException: For input string: ""
java.lang.Exception: parse error in Name: Will.i.am
java.lang.Exception: parse error in Name: Paul W.S. Anderson
```

Tests passed: 2 (2 minutes ago)



assignment-movie-database [~/IdeaProjects/CSYE7200/assignment-movie-database] - .../src/test/scala/edu/neu/coe/csye7200/asstmd/MovieSpec.scala [MovieDatabase]

```scala
package edu.neu.coe.csye7200.asstmd

import org.scalatest.{FlatSpec, Matchers}

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

  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) =>
    }
    Name("Robert Downey Jr.") should matchPattern {
      case Name("Robert", None, "Downey", Some("Jr.")) =>
    }
  }
  it should "work for Name" in {
    val x = Name("Tom", None, "Brady", None)
    x should matchPattern {
      case Name("Tom", None, "Brady", None) =>
    }
  }

  behavior of "Principal"

  it should "work for String, Int" in {
    val x = Principal("Tom Brady", 1)
    x should matchPattern { case Principal(Name("Tom", None, "Brady", None), 1) => }
  }
  it should "work for List[String]" in {
    Principal(List("Tom Brady", "1")) should matchPattern {
      case Principal(Name("Tom", None, "Brady", None), 1) =>
    }
    Principal(List("Noémie Lenoir", "2")) should matchPattern {
      case Principal(Name("Noémie", None, "Lenoir", None), 2) =>
```

Documentation: MovieSpec

```
edu.neu.coe.csye7200.asstmd
class MovieSpec
 extends FlatSpec
 with Matchers
```

Created by scalaprof on 9/13/16.

MovieDatabase

Run: MovieSpec

All Tests Passed 126 ms

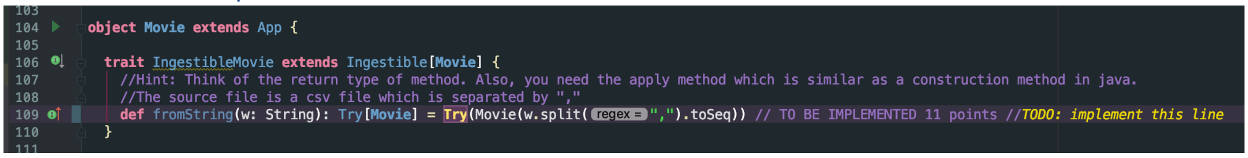
Tests passed: 16 of 16 tests – 126 ms

```
Testing started at 3:19 PM ...
/Library/Java/JavaVirtualMachines/jdk1.8.0_161.jdk/Contents/Home/bin/java "-javaagent:/Users/jimzhou/Library/Application Support/JetBrains/Toolbox/apps/IDEA-U/ch-0/182.4505.22/IntelliJ IDEA.app/Contents/lib/i

Process finished with exit code 0
```

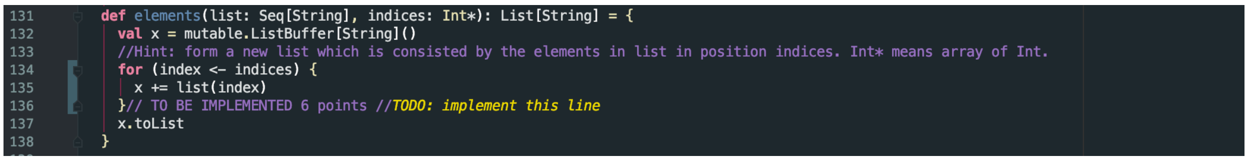
Tests passed: 16 (moments ago)

The code I implemented:

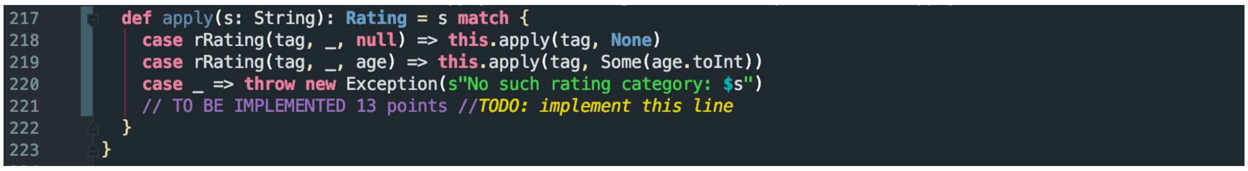


```scala
object Movie extends App {

  trait IngestibleMovie extends Ingestible[Movie] {
    //Hint: Think of the return type of method. Also, you need the apply method which is similar as a construction method in java.
    //The source file is a csv file which is separated by ","
    def fromString(w: String): Try[Movie] = Try(Movie(w.split(regex = ",").toSeq)) // TO BE IMPLEMENTED 11 points //TODO: implement this line
  }
```



```scala
  def elements(list: Seq[String], indices: Int*): List[String] = {
    val x = mutable.ListBuffer[String]()
    //Hint: form a new list which is consisted by the elements in list in position indices. Int* means array of Int.
    for (index <- indices) {
      x += list(index)
    }// TO BE IMPLEMENTED 6 points //TODO: implement this line
    x.toList
  }
```



```scala
  def apply(s: String): Rating = s match {
    case rRating(tag, _, null) => this.apply(tag, None)
    case rRating(tag, _, age) => this.apply(tag, Some(age.toInt))
    case _ => throw new Exception(s"No such rating category: $s")
    // TO BE IMPLEMENTED 13 points //TODO: implement this line
  }
}
```