Bowen Jiang (NUID: 001582174)

CSYE 7200

Big-Data Sys Engr Using Scala

Sprint 2021

Assignment 6(Web Crawler)

Task:

Implement the primitive web crawler that is partly complete in the crawler package on our class repo.

There are five *TO BE IMPLEMENTED* to complete, with a total point value of 53. You may also earn up to 10 bonus points for suggestions on how to improve the web crawler (detailed code not required but you do need to explain in words what you would do). Three of these are in *WebCrawler.scala*. The other two are in *MonadOps.scala* as follows:

```scala
def sequence[X](xfs: Seq[Future[X]])(implicit executor: ExecutionContext): Seq[Future[Either[Throwable, X]]] = ??? // TO BE IMPLEMENTED
def sequence[X](xe: Either[Throwable, X]): Option[X] = ??? // TO BE IMPLEMENTED
```

Please ensure that you pull the latest versions of *WebCrawler.scala*, *HTMLParser.scala* and *MonadOps.scala*. You can get these from the class repo (see Course Material/Resources/Class Repository), the module name for this assignment is **assignment-web-crawler**.
For the processing of a Node, we will need to refer back to the way Scala processes XML documents which we covered in Serialization. Hint: you can get all of the anchor, viz. the "a" nodes using

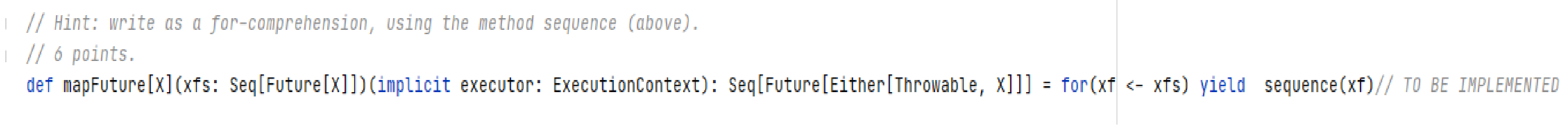
ns \\ "a"

You can get the "href" property from these nodes using "\" and "@href".

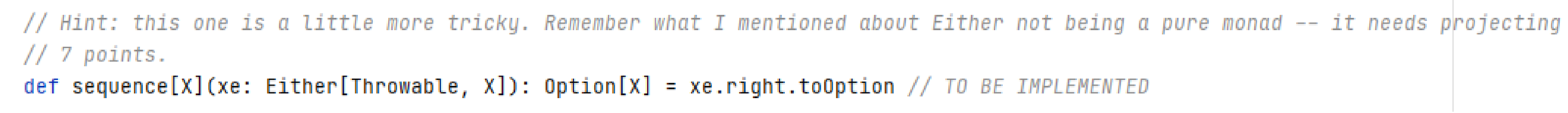
There is also the main program but if you run that, you will need to provide Program arguments consisting of URL(s) at which to start crawling.

Function Implement

MonadOps.scala

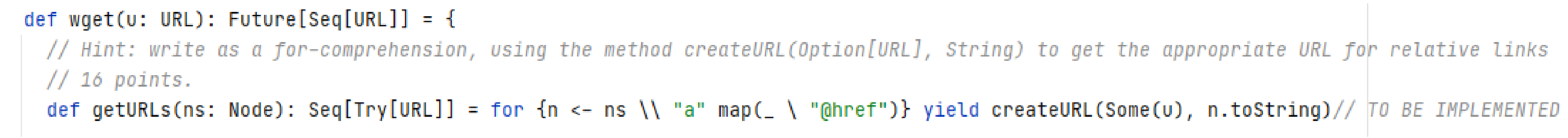


```scala
// Hint: write as a for-comprehension, using the method sequence (above).
// 6 points.
def mapFuture[X](xfs: Seq[Future[X]])(implicit executor: ExecutionContext): Seq[Future[Either[Throwable, X]]] = for(xf <- xfs) yield  sequence(xf)// TO BE IMPLEMENTED
```

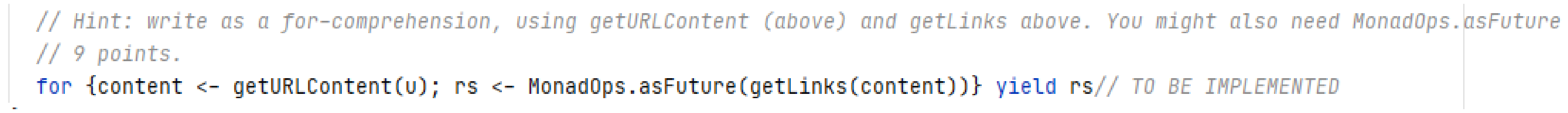


```scala
// Hint: this one is a little more tricky. Remember what I mentioned about Either not being a pure monad -- it needs projecting
// 7 points.
def sequence[X](xe: Either[Throwable, X]): Option[X] = xe.right.toOption // TO BE IMPLEMENTED
```

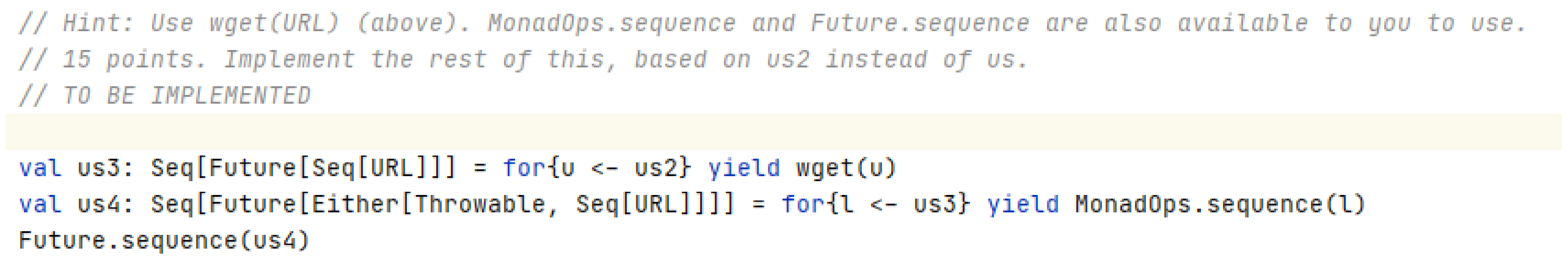
WebCrawler.scala



```scala
def wget(u: URL): Future[Seq[URL]] = {
  // Hint: write as a for-comprehension, using the method createURL(Option[URL], String) to get the appropriate URL for relative links
  // 16 points.
  def getURLs(ns: Node): Seq[Try[URL]] = for {n <- ns \\ "a" map(_ \ "@href")} yield createURL(Some(u), n.toString)// TO BE IMPLEMENTED
```



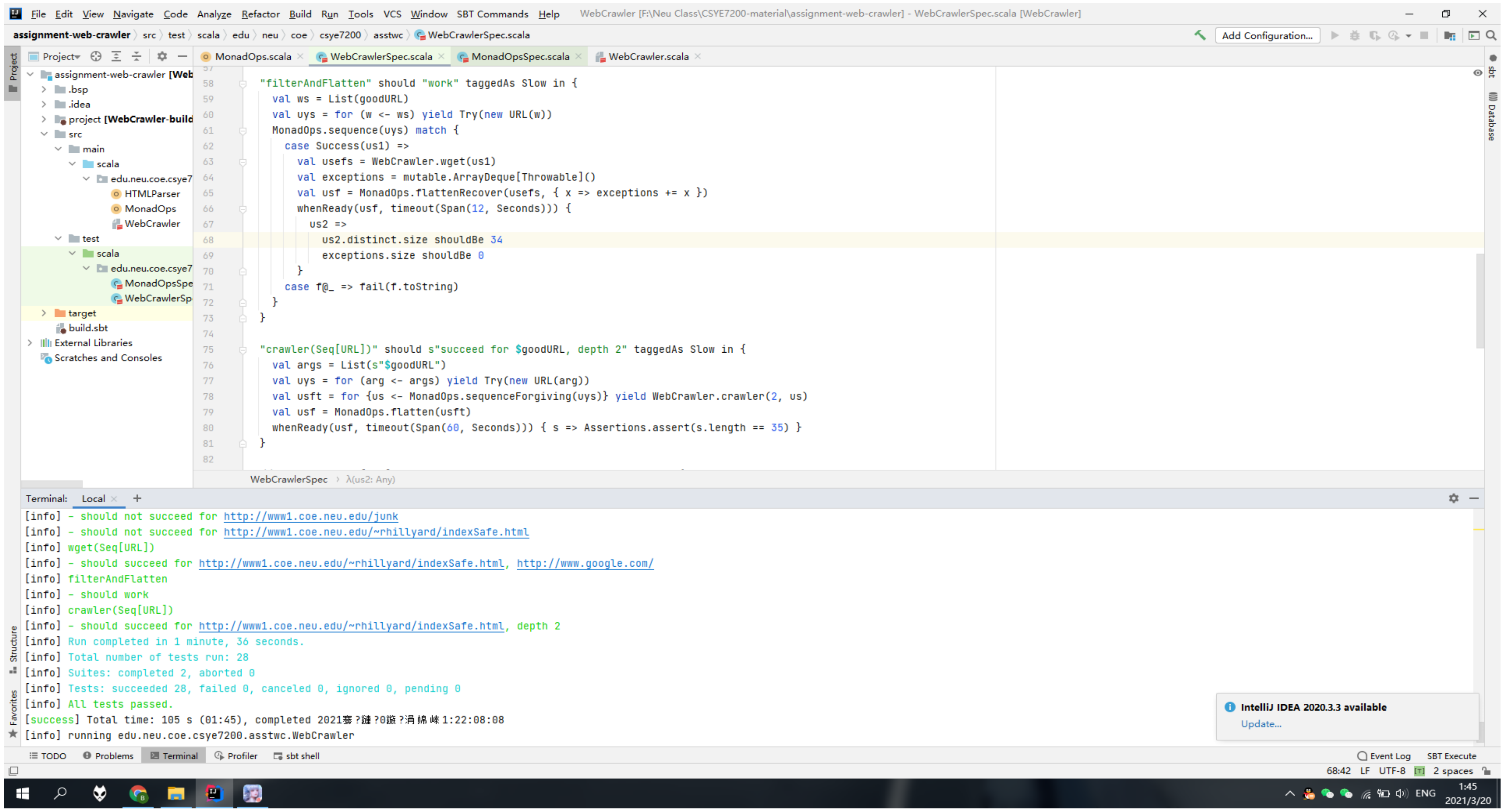
```scala
  // Hint: write as a for-comprehension, using getURLContent (above) and getLinks above. You might also need MonadOps.asFuture
  // 9 points.
  for {content <- getURLContent(u); rs <- MonadOps.asFuture(getLinks(content))} yield rs// TO BE IMPLEMENTED
```



```scala
// Hint: Use wget(URL) (above). MonadOps.sequence and Future.sequence are also available to you to use.
// 15 points. Implement the rest of this, based on us2 instead of us.
// TO BE IMPLEMENTED

val us3: Seq[Future[Seq[URL]]] = for{u <- us2} yield wget(u)
val us4: Seq[Future[Either[Throwable, Seq[URL]]]] = for{l <- us3} yield MonadOps.sequence(l)
Future.sequence(us4)
```

Unit Test Screenshot



WebCrawler [F:\Neu Class\CSYE7200-material\assignment-web-crawler] - WebCrawlerSpec.scala [WebCrawler]
assignment-web-crawler › src › test › scala › edu › neu › coe › csye7200 › asstwc › WebCrawlerSpec.scala
MonadOps.scala
WebCrawlerSpec.scala
MonadOpsSpec.scala
WebCrawler.scala
WebCrawlerSpec › λ(us2: Any)
IntelliJ IDEA 2020.3.3 available
Update...

```scala
  "filterAndFlatten" should "work" taggedAs Slow in {
    val ws = List(goodURL)
    val uys = for (w <- ws) yield Try(new URL(w))
    MonadOps.sequence(uys) match {
      case Success(us1) =>
        val usefs = WebCrawler.wget(us1)
        val exceptions = mutable.ArrayDeque[Throwable]()
        val usf = MonadOps.flattenRecover(usefs, { x => exceptions += x })
        whenReady(usf, timeout(Span(12, Seconds))) {
          us2 =>
            us2.distinct.size shouldBe 34
            exceptions.size shouldBe 0
        }
      case f@_ => fail(f.toString)
    }
  }

  "crawler(Seq[URL])" should s"succeed for $goodURL, depth 2" taggedAs Slow in {
    val args = List(s"$goodURL")
    val uys = for (arg <- args) yield Try(new URL(arg))
    val usft = for {us <- MonadOps.sequenceForgiving(uys)} yield WebCrawler.crawler(2, us)
    val usf = MonadOps.flatten(usft)
    whenReady(usf, timeout(Span(60, Seconds))) { s => Assertions.assert(s.length == 35) }
  }
```

```
[info] - should not succeed for http://www1.coe.neu.edu/junk
[info] - should not succeed for http://www1.coe.neu.edu/~rhillyard/indexSafe.html
[info] wget(Seq[URL])
[info] - should succeed for http://www1.coe.neu.edu/~rhillyard/indexSafe.html, http://www.google.com/
[info] filterAndFlatten
[info] - should work
[info] crawler(Seq[URL])
[info] - should succeed for http://www1.coe.neu.edu/~rhillyard/indexSafe.html, depth 2
[info] Run completed in 1 minute, 36 seconds.
[info] Total number of tests run: 28
[info] Suites: completed 2, aborted 0
[info] Tests: succeeded 28, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[success] Total time: 105 s (01:45), completed 2021骞?鏈?0鏃?涓嬪崍1:22:08:08
[info] running edu.neu.coe.csye7200.asstwc.WebCrawler
```