Programowanie funkcyjne i współbieżne
Lista 11

1. Zdefiniuj funkcję

```
def pairFut[A, B] (fut1: Future[A], fut2: Future[B]): Future[(A, B)] = ???
```

   a) Wykorzystaj metodę zip (wykład 11, str. 18)
   b) Wykorzystaj for (wykład 11, str. 19)

2. Do typu Future[T] dodaj metodę exists:

```
def exists(p: T => Boolean): Future[Boolean] = ???
```

   Wynikowy obiekt Future ma zawierać wartość true wtedy i tylko wtedy, obliczenia obiektu oryginalnego kończą się pomyślnie i predykat p zwraca wartość true, w przeciwnym razie wynikowy obiekt Future ma zawierać wartość false. Wykorzystaj klasę implicytną i mechanizm niejawnych konwersji (wykład 11, str. 22).
   a) Wykorzystaj promesę
   b) Nie korzystaj z promesy (użyj map)

   Przeprowadź trzy testy: kiedy predykat jest spełniony, kiedy predykat nie jest spełniony, kiedy rzucany jest wyjątek.

3. Należy policzyć liczbę słów w każdym pliku tekstowym zadanego folderu i wydrukować wynik w postaci par (nazwa pliku, liczba słów), posortowany niemalejąco względem liczby słów. Możemy założyć dla uproszczenia, że słowa są oddzielone spacjami (wykorzystaj metodę split). Obliczenia należy przeprowadzać asynchronicznie. Program ma być napisany funkcyjnie. Poniżej jest jego schemat.

```
object Zad3 {
  import scala.concurrent._
  import ExecutionContext.Implicits.global
  import scala.util.{Success, Failure}
  import scala.io.Source

   def main(args: Array[String]): Unit = {
//ścieżka do folderu, pobierana z wiersza poleceń, np. "C:/lista11/pliki/" lub "C:\\lista11\\pliki\\"
     val path = args(0)
     val promiseOfFinalResult = Promise[Seq[(String, Int)]]()
 //  Tu oblicz  promiseOfFinalResult

     promiseOfFinalResult.future onComplete {
       case Success(result) => result foreach println
       case Failure(t)          => t.printStackTrace
     }
    Thread.sleep(5000)
  }                                  // koniec metody main

  // Oblicza liczbę słów w każdym pliku z sekwencji wejściowej
  private def processFiles(fileNames: Seq[String]): Future[Seq[(String, Int)]] = ???
      // Wskazówka.  Wykorzystaj Future.sequence(futures)

// Oblicza liczbę słów w podanym pliku i zwraca parę: (nazwa pliku, liczba słów)
  private def processFile(fileName: String): Future[(String, Int)] = ???

// Zwraca sekwencję nazw plików  z folderu docRoot
  private def scanFiles(docRoot: String): Future[Seq[String]] =
     Future { new java.io.File(docRoot).list.toIndexedSeq.map(docRoot + _) }
}
```

Wszystkie zadania umieść w pliku Lista11.scala.