1. Immutable Collections

In diesem Arbeitsblatt lernen Sie den Umgang mit den immutable Scala Collections. Auf dem AD finden Sie das 10_LE_Scala_Conpr.zip Archiv. Importieren Sie es in Ihre Scala-IDE. Für dieses erste Arbeitsblatt sind die Packages ws.`common` und ws.`collections` relevant.

Im Package ws.`collections` haben wir Ihnen eine realistische Scala Portierung des Evento Systems (Schulverwaltungssoftware an der FHNW) bereitgestellt:

```
object SyncEvento {
  def classMembers(): List[Student] = ...

  def estimateGrade(s: Student): Double = ...
}
```

Das Objekt `SyncEvento` bietet die Methode `classMembers`, die eine Liste der Studiereden Ihrer Klasse zurückgibt. Die Klasse `Student` ist so definiert:

```
class Student(val email: String)
```

Zusätzlich steht die Methode `estimateGrade` zur Verfügung, um für einen Studierenden die MSP Note für das Conpr Modul vorherzusagen.

Sie sollen nun den `SyncEventoClient` vervollständigen, um die Top 10 Studierenden ihrer Klasse auszugeben. Führen Sie als Einstieg die `main` Methode des `SyncEventoClient` aus um die Liste der Studierenden auszugeben. Seien Sie geduldig - `SyncEvento` ist nun mal etwas langsam.

Aufgaben:

a) Erzeugen Sie eine neue Liste `gradedStudents` vom Typ `List[GradedStudent]`, indem Sie für jedes `Student` Element der Liste `students` die `estimateGrade` holen und eine neue Instanz der Klasse `GradedStudent` erzeugen:

```
class GradedStudent(val student: Student, val grade: Double)
```

Hinweise:
- Verwenden Sie die Methode `map`
- Lassen Sie das Programm einfach nochmals laufen, wenn Sie mit der für Sie geschätzten Note nicht einverstanden sind.

b) Suchen Sie in der Liste `gradedStudents` nach den talentierten Studierenden – das sind jene Studierende mit einer Note >= 5.
Hinweis: Verwenden Sie die Methode `filter`

c) [Bubi] Geben Sie die Top 10 Studierenden aus. Wir haben für Sie die Methode `printTopTen` bereits implementiert.

d) [Optional] Studieren Sie das API [1] der Klasse `scala.collection`.immutable.`List`
Finden Sie heraus was der Aufruf von `zipWithIndex` in der Methode `printTopTen` bewirkt?

[1] http://www.scala-lang.org/api/current/scala/collection/immutable/List.html

```scala
/** The most important things to do with Scala's immutable collections! */
object ImmutableCollections extends App {

  /** List operations */
  val list0 = List(1,2,1)                    // List(1,2,1)

  val head = list0.head                      // 1

  val tail = list0.tail                      // List(2,1)

  val list1 = 5 :: list0                     // List(5,1,2,1)

  val list2 = Nil                            // List()

  val list3 = list0.map(i => i + 1)          // List(2,3,2)

  val list4 = list0.filter(i => i > 1)       // List(2)

  val sum = list0.reduce((x,y) => x+y)       // 4

  val list5 = list0.zip(List('A', 'B', 'C')) // List((1,A), (2,B), (1,C))

  val list6 = list0.groupBy(i => i % 2 == 0) // Map(false -> List(1,1), true -> List(2))

  val large = list0.find(i => i > 12)        // None

  val small = list0.find(i => i < 12)        // Some(1)

  list0.foreach(i => print(i + " "))         // 1 2 1


  /** Set operations */
  val set0 = Set(1,2,3,2)            // Set(1,2,3)

  val set1 = set0 + 4                // Set(1,2,3,4)

  val set2 = set0 - 1                // Set(2,3)

  val contains0 = set1(0)            // false

  val set3 = set1.filter(i => i > 2) // Set(3,4)

  val set4 = set1.map(i => i > 2)    // Set(false,true)


  /** Map operations */
  val map0 = Map(1 -> "one", 2 -> "two")  // Map(1 -> "one", 2 -> "two")

  val map1 = map0 + (3 -> "three")        // Map(1 -> "one", 2 -> "two", 3 -> "three")

  val map2 = map0 - 1                     // Map(2 -> "two", 3 -> "three")

  val val1 = map0(1)                      // "one"

  val val0 = map0(0)                      // j.u.NoSuchElementException: key not found: 0

  val optVal0 = map0.get(0)               // None

  val optVal1 = map0.get(1)               // Some(1)

  val res = map1.filter(kv => kv._1 > 2)  // Map(3 -> "three")
}
```

2. Observables

In diesem zweiten Arbeitsblatt programmieren Sie gegen eine asynchrone Variante des Evento Systems. Sie finden den Code im Package ws.`observables`. Hier ist die Definition von `AsyncEvento`:

```
object AsyncEvento {
  def classMembers(): Observable[Student] = ...

  def estimateGrade(s: Student): Double = ...
}
```

Die Methode `classMembers` gibt in dieser Variante ein `Observable[Student]` [1] zurück und erlaubt somit die asynchrone Umsetzung der Aufgabe aus dem ersten Teil des Arbeitsblatts. Führen Sie die main Methode des `AsyncEventoClient` aus. Vergleichen Sie das Verhalten zur `SyncEventoClient` Implementierung. Viel besser oder?

Aufgaben:

a) Erzeugen Sie ein neues Observable `gradedStudents` vom Typ `Observable[GradedStudent]` indem Sie für jedes `Student` Element des Observables `students` die `estimateGrade` holen und eine neue Instanz der Klasse `GradedStudent` erzeugen.

b) Suchen Sie im Observable `gradedStudents` nach den talentierten Studierenden – das sind jene Studierende mit einer Note >= 5.

c) [Bubi] Geben Sie die Top 10 Studierenden aus. Wir haben für Sie die Methode `printTopTen` bereits implementiert.

d) Vergleichen Sie Ihre Implementierung des `AsyncEventoClient` mit derjenigen des `SyncEventoClient`. Was fällt Ihnen auf?

e) Wieso können die Top 10 erst ganz am Ende ausgegeben werden?

[1] http://reactivex.io/rxscala/scaladoc/#rx.lang.scala.Observable