Do you want to look at a language that allows you to implement FP nicely?

Do you want to look at a language that allows you to implement FP nicely?

Do you want to look at a language that has a similar syntax to one you've seen already?

Do you want

Do you wan
at a langua
allows you
implement
nicely?

Do you want to look at a language that companies actually use?

Do you want to look at a language that allows you to implement FP nicely?

Do you (not) like Haskell?

Do you want to look at a language that allows you implemen nicely?

Do you want to look at a language that has a similar

o you ot) like askell?

Well, aren't you in luck?!

Do you want to look at a language that companies actually use?

```
val training_df = sqlContext.createDataFrame(training_pca)
val test_df = sqlContext.createDataFrame(test_pca)



val tokenizer = new Tokenizer()
val hashingTF = new HashingTF()
val lr1 = new LogisticRegression()
val pipeline = new Pipeline()

val model = pipeline.fit(training_df)

mod.transform(test_df)
  .select("id", "text", "probability", "prediction")
  .collect()
  .foreach { case Row(id: Long, text: String, prob: Vector, prediction: Do
  println(s"($id, $text) --> prob=$prob, prediction=$prediction")
}
```

solves concurrency in a "safer way"

"makes me a better programmer"

static typing

pattern matching

Scala

```scala
val training_df = sqlContext.createDataFrame(training_pca)
val test_df = sqlContext.createDataFrame(test_pca)


val tokenizer = new Tokenizer()
val hashingTF = new HashingTF()
val lr1 = new LogisticRegression()
val pipeline = new Pipeline()

val mod = pipeline.fit(training_df)

mod.transform(test_df)
  .select("id", "text", "probability", "prediction")
  .collect()
  .foreach { case Row(id: Long, text: String, prob: Vector, prediction: Do
  println(s"($id, $text) --> prob=$prob, prediction=$prediction")
}
```

```scala
val training_df = sqlContext.createDataFrame(training_pca)
val test_df = sqlContext.createDataFrame(test_pca)



val tokenizer = new Tokenizer()
val hashingTF = new HashingTF()
val lr1 = new LogisticRegression()
val pipeline = new Pipeline()

val mod = pipeline.fit(training_df)

mod.transform(test_df)
  .select("id", "text", "probability", "prediction")
  .collect()
  .foreach { case Row(id: Long, text: String, prob: Vector, prediction: Do
  println(s"($id, $text) --> prob=$prob, prediction=$prediction")
}
```

Scala

putting the fun

Scala

putting the fun into

```
val training_df = sqlContext.createDataFrame(training_pca)
val test_df = sqlContext.createDataFrame(test_pca)



val tokenizer = new Tokenizer()
val hashingTF = new HashingTF()
val lr1 = new LogisticRegression()
val pipeline = new Pipeline()

val mod = pipeline.fit(training_df)

mod.transform(test_df)
  .select("id", "text", "probability", "prediction")
  .collect()
  .foreach { case Row(id: Long, text: String, prob: Vector, prediction: Do
  println(s"($id, $text) --> prob=$prob, prediction=$prediction")
}
```

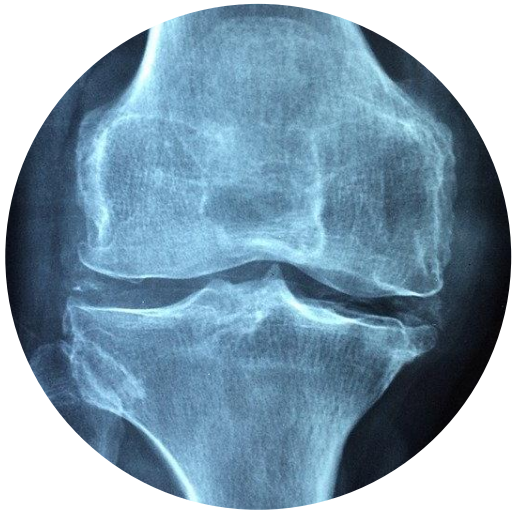
Scala

putting the fun into functional programming

Keoni D'Souza, 921231

w/ Dr Monika Seisenberger

Friday, 31 January 2020



Benvenuto/a!

Willkommen!

Добро пожаловать!

Welcome!

Välkommen!

Bienvenue !

Croeso!

Witaj!

¡Bienvenido/a!

Selamat datang!

Bem-vindo/a!

Hoş geldin!

Benvenuto/a!

Willkommen!

Добро пожаловать!

Välkomm

[APPLAUSE]

venue !

Wi /a!

Selamat datang!

Bem-vindo/a!

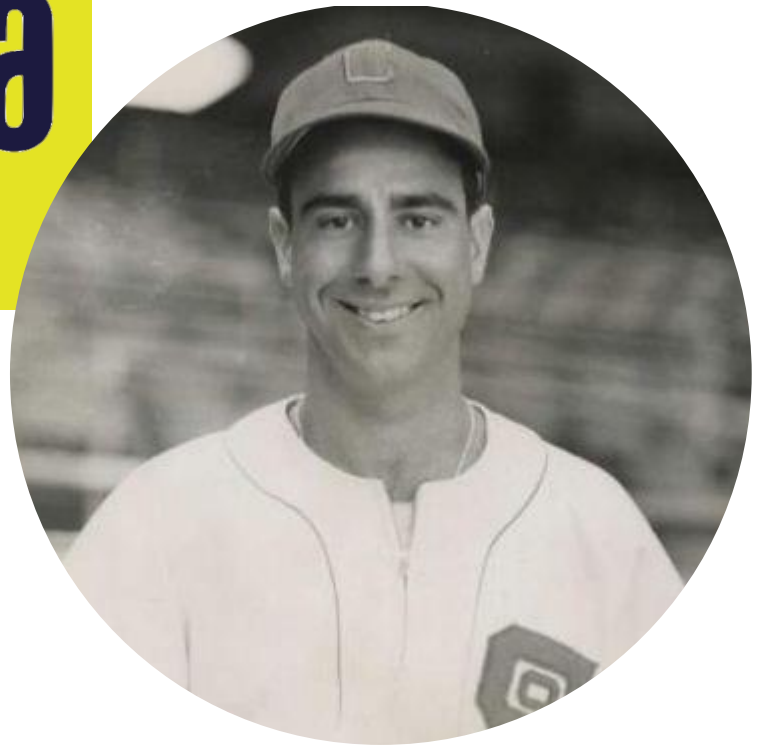
Hoş geldin!

WHAT ARE YOU HERE TO TALK ABOUT TODAY?

What are you here to talk about today?

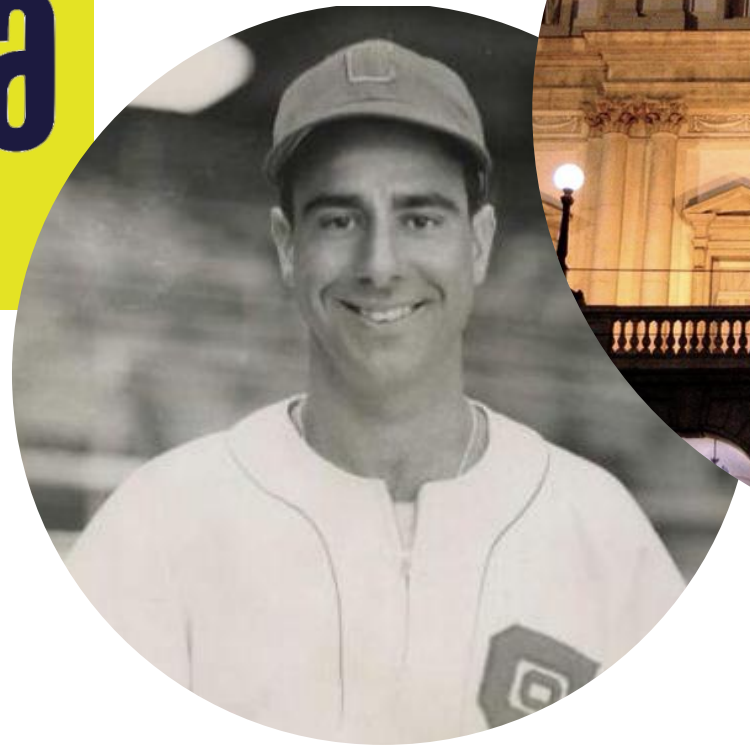
What are you here to talk about today?

scala RADIO



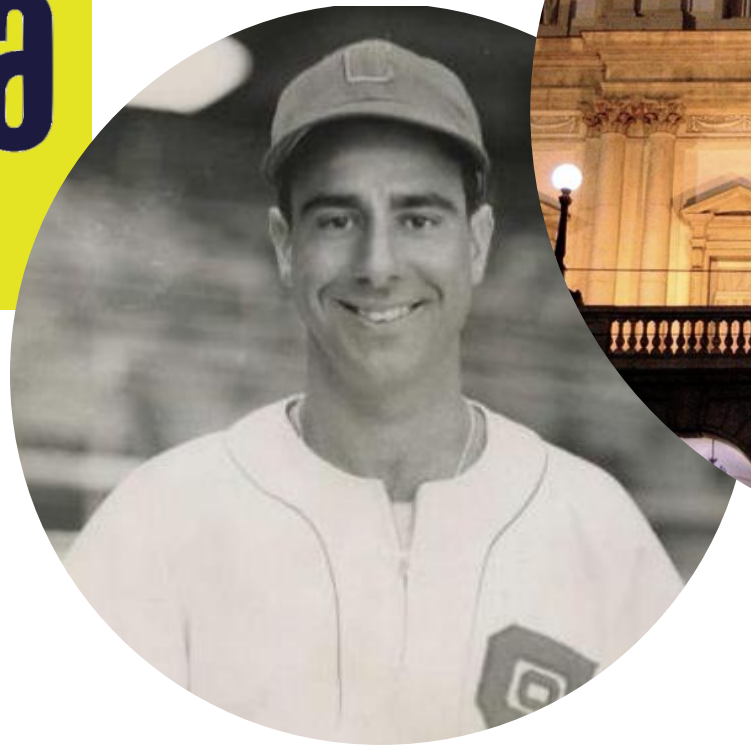
What are you here to talk about today?

scala RADIO



What are you here to talk about today?

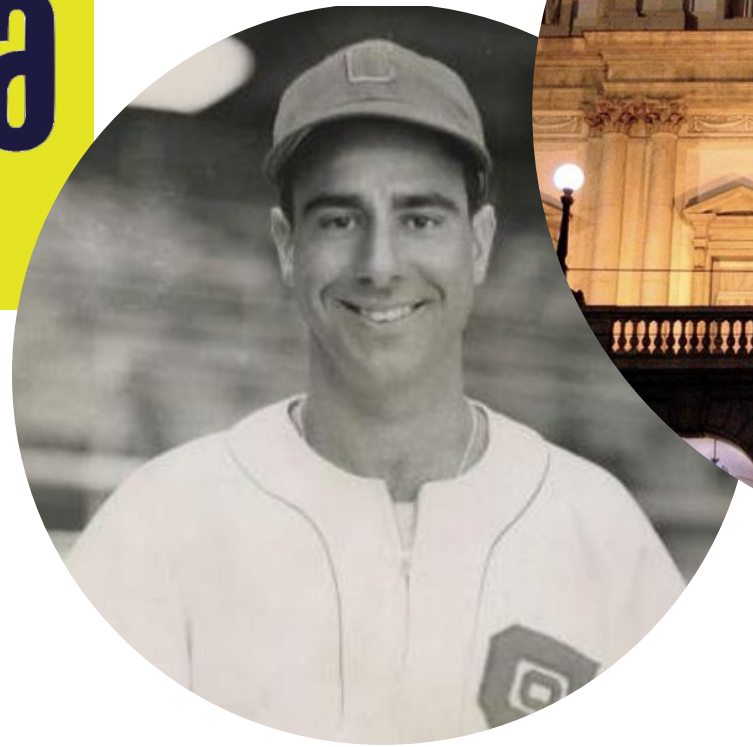
scala RADIO



Scala

What are you here to talk about today?

scala RADIO



Scala

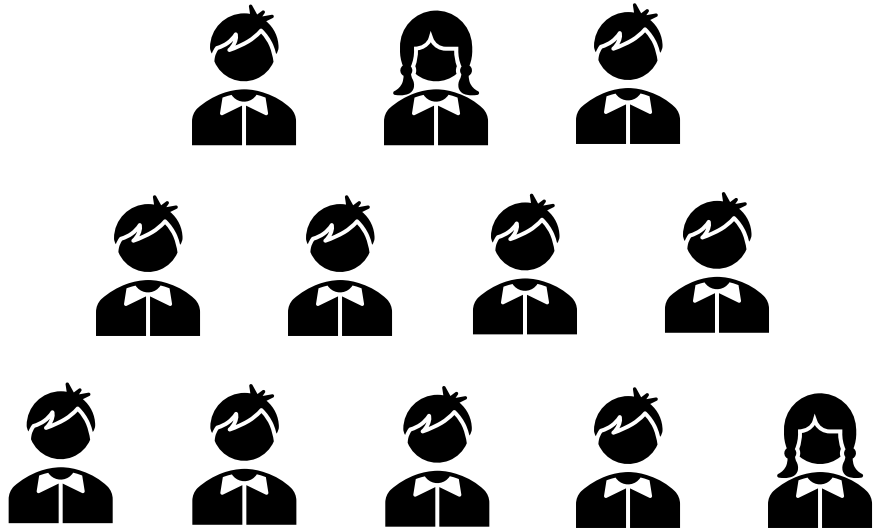
WHY ARE YOU HERE?

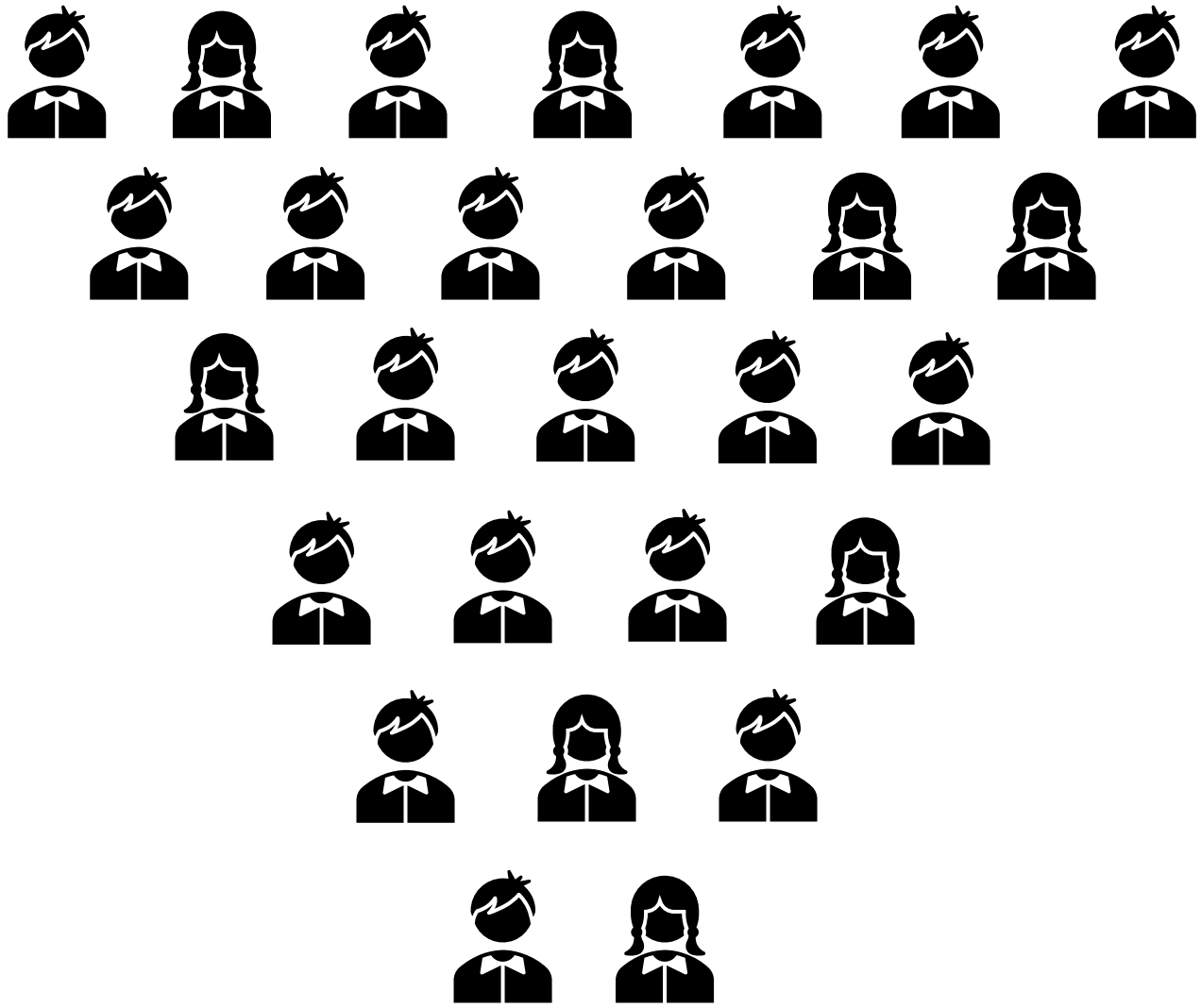
Why are you here?

Project motivations

IMP  RTANT!

Swansea University
Prifysgol Abertawe





WHAT EVEN *IS* SCALA?

WHAT EVEN *IS* SCALA?

Background on the language

What even is Scala?
Background on the language

- Scala (scalable language)
  - small to the big
- Not a Java extension
  - Interoperability
  - Basic operators, data types and control structures shared

What even is Scala?
Background on the language

- Design started in 2001 at EPFL
- Better ways than Java
- Providing an alternative
- Precursors:
  - Pizza – moderately successful
  - Funnel – too academic
- First public release: 2003
- Scala v2.0: 2016

SHOW ME THE GOOD STUFF...

Programming functionally in Scala

Functions

- A function is a group of statements that perform a task.
- Scala has both functions and methods:
  - A method is a part of a class which has a name, a signature, optionally some annotations, and some bytecode
  - A function is a complete object which can be assigned to a variable.
  - In other words, a function, which is defined as a member of some object, is called a method.

Functions

- Functions are declared in the following form:

```
def functionName ([list of parameters]): [return type]
```

Functions

- Functions are defined in the following form:

```
def functionName ([list of parameters]) : [return type] = {
   function body
   return [expr]
}
```

- The return type could be any valid Scala data type and list of parameters will be a list of variables separated by commas (both are optional).

Functions

```
object add {
  def addInt(a: Int, b: Int): Int = {
    var sum: Int = 0
    sum = a + b
    return sum
  }
}
```

Functions

```
object add {
  def addInt(a: Int, b: Int): Int = a + b;
}
```

- Even cleaner!
- You don't even need the return keyword – the last value is automatically returned!
- Semicolons and braces are not always required.

Functions

- How would we write a function to calculate the square of a number?
- The math library has a built-in power function:

```
pow(x: Double, y: Double): Double
```

where:

x is the base

y is the exponent

x^y is returned

that allows you to calculate the square easily:

```
scala.math.pow(n,2).
```

Functions

- But, how would we describe one manually?

```
def square(n: Int): Int = n * n
```

- The type annotations (the parts after the colons) don't necessarily have to be included because of Scala's built-in type inference.

Functions

- Functions that don't return anything are called procedures.
  - It doesn't actually return nothing – it returns a Unit, which is equivalent to Java's void.

```
object Hello {
  def printMe(): Unit = {
    println("Hello, Scala!")
  }
}
```

Reading user input

- The io.StdIn library stands for standard input, allowing users to communicate with the keyboard and interact with Scala functions.
- What does this function do?

```
def greet() = {
  val name=scala.io.StdIn.readInt("Please enter your name: ");
  println("Congratulations, " + name + " – you have been
called to learn Scala!);
}
```

Lists

mutability

you can change stuff

`ListBuffer`

immutability

stuff stays the same

`List`

```
import scala.collection.mutable.ListBuffer;
```

Lists

- Creating a new ListBuffer instance:

```
var breads = new ListBuffer[String]();
```

```
import scala.collection.mutable.ListBuffer;
```

Lists

- Adding to a ListBuffer instance:

```
breads ++= "Bagel"

breads ++= List("Baguette",
                "Boule",
                "Brioche")
```

```
import scala.collection.mutable.ListBuffer;
```

Lists

- Removing from a ListBuffer instance:

```
breads -= “Brioche";
```

```
import scala.collection.mutable.ListBuffer;
```

Lists

- Also:

```
breads remove 0; // the same as breads.remove(0);
```

- Here, we're using postfix notation instead of brackets.
- The number refers to the position in the ListBuffer.

List operations

- `anyList.head` returns the first element
- `anyList.tail` returns the list minus the first element
- `anyList.isEmpty` returns a Boolean asking if the list is empty

List operations

```
scala> val nums = List.range(1,10)
x: List[Int] = List(1, 2, 3, 4, 5, 6, 7, 8,
9)

scala> val nums = List.range(0,10,2)
x: List[Int] = List(0,2,4,6,8)
```

Show them the square function!

```
// square: Int -> Int
def square(x: Int): Int = x*x
def square: x => x*x
```

Scala collection operations

The `map` operation

```
val ints = list.map(s => s.toInt)
```

- The `map` operation takes a predicate and applies it to every element contained within the collection.
- It's part of the `TraversableLike` trait, so will work on all different types of collections.
- In this example, for each `s` it applies the `toInt` function to convert it into an integer.

The `filter` operation

```
val ints = list.map(s => s.toInt).filter(_ % 2 == 0)
```

- The `filter` operation takes a predicate that returns a Boolean.
- If an element evaluates to true, it is returned. Falsely evaluated items are filtered out of the result.
- In this example, even numbers are returned. The underscore (_) symbol (wildcard) represents, in each case, the evaluated element.

The `flatten` operation

```
val couples = List(List("Jim", "Julia"), List("Alex", "Sam"))
val people = couples.flatten
```

- The `flatten` operation takes a collection of n dimensions and squashes it into $n - 1$ dimensions.
- It works, from the lowest degree, with two-dimension collections or higher.
- In this example, the pairs in `couples` are flattened into an array containing all the peoples' names. Compare the 2-dimensional `couples` with the 1-dimensional `people`.

The `flatmap` operation

```
val couples = List(List(“Jim”, “Julia”), List(“Alex”, “Sam”))
val people = couples.flatmap(_ + “ Blanck”)
```

- `flatmap` combines the flatten and map operations.
- It is syntactic sugar for:

```
val people = couples.flatten.map(_  + “ Blanck”)
```

- In this example, for each person it adds the same surname.

WILL THIS WORK FOR PEOPLE LIKE US?

Teaching Scala to students

Will this work for people like us?

Teaching Scala to students

- Multi-paradigm – there's leeway
- Elegant – you can write beautiful code
- You could earn, on average, £85,000
- Big companies use the language

IS THAT IT?

Rounding off the presentation

Image references

[1] https://upload.wikimedia.org/wikipedia/en/a/a2/Scala_Radio.svg

[2] https://i.pinimg.com/originals/df/e8/f3/dfe8f32097ef7086ee02f84064c31925.jpg

[3] https://upload.wikimedia.org/wikipedia/commons/5/52/Milano-scalanotte_e.jpg

[4] https://upload.wikimedia.org/wikipedia/en/8/85/Scala_logo.png

[5] https://upload.wikimedia.org/wikipedia/commons/b/b7/Mark_Odersky_photo_by_Linda_Poeng.jpg

[6] https://upload.wikimedia.org/wikipedia/commons/e/e9/9.13.09DaveRomanByLuigiNovi.jpg

General references

- [1] Aggarwal, R S 2019. 10 top Programming Languages in 2019 for Businesses. [https://codeburst.io/10-top-programming-languages-in-2019-for-developers-a2921798d652]. Accessed October 2019.
- [2] Alexander, A. Tail-Recursive Algorithms in Scala. [https://alvinalexander.com/scala/fp-book/tail-recursive-algorithms]. Accessed October 2019.
- [3] Bahadoor, N 2018. An Overview Of Scala Type Inference. [http://allaboutscala.com/tutorials/chapter-2-learning-basics-scala-programming/scala-tutorial-overview-scala-type-inference/]. Accessed October 2019.
- [4] Butcher, S. The UK's best-paying technology jobs and coding languages. [https://news.efinancialcareers.com/fi-en/3002150/pay-for-developers-uk]. Accessed October 2019.
- [5] Odersky, M; et al. 2015. An Overview of the Scala Programming Language (2. Edition). 24 pp.
- [6] Odersky, M; Spoon, L; and Venners, B 2008. Programming in Scala. Artima Press, California. 754 pp.
- [7] Odersky, M; et al. 2006. Scala Language Specification. [https://www.scala-lang.org/files/archive/spec/2.13/]. Accessed October 2019.
- [8] Odersky, M 2008. Scala's Prehistory. [https://www.scala-lang.org/old/node/239.html]. Accessed October 2019.
- [9] Venners, B; Sommers, F; and Odersky, M 2009. The Origins of Scala; A Conversation with Martin Odersky, Part I. [https://www.artima.com/scalazine/articles/origins_of_scala.html]. Accessed October 2019.
- [10] EDUCBA 2018. Uses of Scala. [https://www.educba.com/uses-of-scala/]. Accessed October 2019.
- [11] The Cats Maintainers. Cats: Lightweight, modular, and extensible library for functional programming [https://typelevel.org/cats/]. Accessed October 2019.
- [12] Scala Intro for Spark, v3. [http://mse-bda.s3-website-eu-west-1.amazonaws.com/lectures/BDA%20Lc06%20ScalaIntroForSpark-Part2.pdf]. Accessed October 2019.
- [13] Scala – Reviews, Pros & Cons | Companies using Scala. [https://stackshare.io/scala]. Accessed October 2019.
- [14] Scala Tutorial | Object Oriented Programming. [https://www.scala-exercises.org/scala_tutorial/object_oriented_programming]. Accessed October 2019.
- [15] TIOBE Index | TIOBE – The Software Quality Company. [https://www.tiobe.com/tiobe-index/]. Accessed October 2019.
- [16] What is a Gantt chart? Gantt Chart Software, Information, and History. [https://www.gantt.com/]. Accessed October 2019.
- [17] What is Statically Typed? – Definition from Techopedia. [https://www.techopedia.com/definition/22321/statically-typed]. Accessed October 2019.
- [18] What is the difference between a 'hazard' and a 'risk'?. [https://worksmart.org.uk/health-advice/health-and-safety/hazards-and-risks/what-difference-between-hazard-and-risk]. Accessed October 2019.
- [19] 5 Ways To Manage Risk. [http://www.dbpmanagement.com/15/5-ways-to-manage-risk]. Accessed October 2019.

That's it
from me!