Putting the fun into functional programming:

Exploring Scala for teaching functional programming and writing securely

Keoni D'Souza
921231

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Bachelor of Science

Department of Computer Science
Swansea University

Friday 15th May 2020

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed  (candidate)

Date Friday 15th May 2020

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed  (candidate)

Date Friday 15th May 2020

Statement 2

I hereby give my consent for my thesis, if accepted, to be made available for photocopying and inter-library loan, and for the title and summary to be made available to outside organisations.

Signed  (candidate)

Date Friday 15th May 2020

Abstract

We are often told that anyone can program – but they are not always so specific. Some ways of programming are easier than others – it is common to focus on just one paradigm, such as object-oriented programming, crafting your code to align to the concept of so-called *objects* containing data (as attributes) and methods. However, other ways exist, some that can encourage the programmer to think more deeply about *how* they code.

Functional programming uses functions solely through application and composition: favoured are returned values, rather than changing program state. Scala has embraced this side, whilst still maintaining ties with Java's object-oriented style. Adjusting one's mindset can be difficult when stuck in one's ways, so it might well be useful to suggest *evolution* – it should be posited – as opposed to *revolution*.

The multi-paradigm nature of Scala allows for a gentler transition between the two schools of thought (at least in my experience) so this project will look at the ability of Scala in communicating the concepts of functional programming most effectively to a new, student audience. I will analyse previous efforts in teaching Scala, highlight notable functional approaches and, in addition, consider its viability in verifying programs.

I will then use a lecture and practical session I conducted to introduce the basic concepts and transitionary notions and find out if students were more receptive to this language over Haskell when approaching the functional programming paradigm.

With active developers highlighting features such as static typing and pattern matching, and many expressing that it solves concurrency in a "safer way" and "makes [them] a better engineer", [1] *Scala really puts the fun into functional programming…*[I]

- Keoni D'Souza, Gregynog Colloquium, November 2019

[I] Whilst I will not claim here to have originated this phrase, this was from where and when it was first uttered – for the purposes of this project – from my very lips to a(n) (mandatorily captive) audience of my peers.

Acknowledgements

Thank you to my family for letting me get on with it.

Thanks to Dad for proofreading.

Thanks to my friends for supporting my Scala crusade.

And thank you to my supervisor, Monika, for making sure this didn't end up as a four-page project.

Dedications

For Mr & Mrs Padilla. However this ends up, it's a shame I can't share this with you.

Typographic notes

*Paragraph text typeset in **Merriweather Light**, 12pt.*

*Headers typeset in **Merriweather Sans**, 18pt, 16pt, 14pt, 13pt and 12pt.*

*Code fragments typeset in **Fira Code Light**, 11pt, 10.5pt and 10pt.*

Table of Contents

Chapter 1 / Introduction

Scala is a language noticeably still within its infancy, not coming into fruition until after the turn of the century. That should not mean that its presence should be diminished though, especially when comparing it to long relied-upon languages (like C++) or the popular (such as Python). Scala sought to address criticisms of Java and act as a better, more concise alternative and this project seeks to find out what is great about the language, whether we should be encouraging students to become proficient in it, and if it is safe enough to be considered for program verification.

1.1/ Project motivations

Two years ago, I was introduced to the functional programming universe through the CS-205 module on declarative programming at Swansea University. It covered two languages – Haskell and Prolog. At the start of my computer science degree, we were introduced to Java and so I became used to an object-oriented way of thinking (I also looked at OOP techniques in Python at GCSE and A Level). The transition was, therefore, quite a difficult one for me as it involved a more mathematical approach and, for me at least, proved to be quite a high barrier for me to breach. The purely functional nature of Haskell failed to provide a suitable level of assistance.

I was able to leave the module (and the often traumatic[II] world of functional programming) behind – that is until I became accustomed to Scala during my placement at ITV last year.

Although multi-paradigm, ITV use functional Scala to program their backend microservices: after my first six months looking at front-end development, I was exposed to functional programming yet again – but my experience was noticeably different this time. Perhaps it was because I had some prior knowledge on the subject, but I found it easier to understand all the functional concepts, almost feeling like I was learning them again as after my exam on the declarative programming module, I tucked away all that knowledge deep somewhere out of my consciousness.

[II] At least, to me, I should clarify – though a not inconsiderable number of people with whom I have spoken have shared similar feelings.

I realised that, most likely, its appeal was because of Scala's Java-like syntax. I had already two years of Java experimentation, so the initial barrier of entry was not too high, especially in comparison to Haskell. It wasn't long before things became much more complicated through the use of *Cats* – a library providing abstractions for functional programming [2] – but I ended up in a position with a comparatively greater understanding than when I approached the more challenging concepts a year prior in Haskell.

The idea that others, maybe in a similar position, were possibly looking for an easier transition to functional programming entered my mind: what if, I thought to myself, students were able to look at the functional side of Scala first and use it as a gateway language to progress onto the purer Haskell? My experience proved to humour this, and so this experiment was considered.

This is the motivation for this project: I plan to carry out a feasibility study, somewhat, to determine if touching upon Scala is an effective way to improve the understanding of the ideas of functional programming.

1.2/ Project aims

It might be the case that, to this student, Scala appears to be a clean language for many purposes, one that should be a cause for consideration, but contextual concerns should be studied: how well does it perform when teaching functional programming and when verifying programs? Thus, this project seeks to achieve the following aims:

- To research and present the programming language Scala:
  - What can it do?
- To look at Scala from a teaching perspective:
  - How can it be introduced to newcomers in an accessible fashion?
  - Will the language add to students' understanding of functional programming?
  - Which is the best order to learn the languages Java, Scala, and Haskell?
  - Could Scala be added to or replace Haskell in the CS-205 module?
- To explore the language's security:
  - Can it be used to write safe programs?

These aims will be expanded upon later in *3.1/ General considerations*.

Chapter 2 / Background & Related work

2.1/ Scala at a glance

Scala, a portmanteau originating from scalable language, [3] is a programming language designed to appeal to a wide range of tasks, capable of providing a suitable environment that can be scaled from small-scale scripts to larger system applications. Though not an extension, it maintains an interoperability with Java, sharing "basic operators, data types and control structures" [4] and combines the worlds of functional and object-oriented programming, producing code that generally matches Java's efficiency in compilation, whilst allowing for a codebase with a significantly decreased verbosity.

2.2/ History of the language

Originating from Martin Odersky, Scala was designed in 2001, along with his peers at EPFL (École Polytechnique Fédérale de Lausanne, Swiss Federal Institute of Technology in Lausanne). Odersky revealed an interest with functional programming towards the end of his stay at ETH Zurich (Eidgenössische Technische Hochschule Zürich, Swiss Federal Institute of Technology in Zurich) in the late 1980s, where he undertook his PhD with the inventor of Pascal, Niklaus Wirth.

Remaining in the research sphere, he became a university professor in Karlsruhe, Germany and started off working on the theoretical side of programming with Phil Wadler of the University of Glasgow, where he was informed by an assistant that there was a new language coming out that was portable, had bytecode, ran on the web and had garbage collection. [5]

What followed was Pizza, a language that drew generics, higher-order functions and pattern matching from the functional programming space and implemented these features on the Java Virtual Machine, or JVM. It came out in 1996 – the year after Java's initial release – to moderate success, proving that functional programming could have a future alongside the object-oriented world of Java.

The problem with Java, they found, was that it had firm limitations, and Odersky thought that there were better ways of doing things – the right way, so to speak, at least in his mind. So, instead of trying to make Java better, he and his team sought to produce an alternative; it still had to be compatible with the existing Java infrastructure, like the JVM and its associated libraries, as creating a language from scratch would be impractical.

Funnel was the result – but its purity necessitated a steep learning curve and was both practically inaccessible to beginners and tedious to experts. They decided to start over again and find a middle ground between Funnel and Generic Java, something that Odersky worked on in the late 1990s and which formed the basis of generics in Java 5 six years later. The language that they started to design in 2001 became known as Scala, with its first public release in 2003. [6] A moderately extensive redesign followed in early 2006 as Scala v2.0.

Scala has grown in popularity ever since. It is currently the twelfth most active programming language on GitHub and the fourth most highly paid programming language globally according to the latest annual developer survey by Stack Overflow. [7]

2.3/ Significant features of Scala

Scala was designed essentially, it could be seen, as a better version of Java. Indeed, its syntax resembles Java and classes in Scala are able to call methods and objects from Java, as well as inherit from its classes and implement interfaces. [8] There are numerous other features of Scala that are worth delving into, many of which are detailed below.

2.3.1/ A multi-paradigm approach

Scala is an object-oriented language and can be considered pure, as every value is treated as an object. But not only that, it is also functional, because each function is considered a value (by transitivity, every function is also an object since all values are, too). Thus, its multi-paradigm nature can build upon these skills, as well as knowledge of imperative and logical programming. [9] This does, however, mean that there are many more ways of doing the same thing, but this will always be the case in any language, and as long as one way is followed (as should always be true anyway) coding can be performed in an effective manner without too many complications arising. It is for this reason that newcomers are misled into thinking that Scala is a difficult language with which to overcome.

2.3.1.1/ Object-oriented programming

Borrowing from Java, Scala can provide an environment for object-oriented programming. As an example, were we to design a class that models a rational number, one might introduce a couple of entities:

```
class Rational(x: Int, y: Int) {
  def numerator   = x
  def denominator = y
}
```

Figure 2.1, OOP in Scala: Rational is a type and a constructor that creates elements of type Rational[III]

2.3.1.2/ Functional programming

Scala delivers when programming functionally with tail recursion, for example, where a function calls itself as the last execution. A summing function can be expressed using an accumulator that calculates a running total:

```
import scala.annotation.tailrec

def sum(list: List[Int]): Int = {
  @tailrec
  def sumWithAccumulator(list: List[Int], currentSum: Int): Int = {
    list match {
      case Nil => { // if there are no more numbers to add
        ???
      }
      case x :: xs => sumWithAccumulator(xs, currentSum + x)
    }
  }
  sumWithAccumulator(list, 0) // Initiates the function, starting at 0
}
```

Figure 2.2, FP in Scala: The @tailrec annotation tells the compiler that it should only run if the following function is tail recursive.[IV]

The compiler also supports an optimisation technique called tail call recursion, making Scala code compile faster than Java.

[III] Based on the example here: "Scala Tutorial | Object Oriented Programming" [https://www.scala-exercises.org/scala_tutorial/object_oriented_programming]. Accessed October 2019.

[IV] Modified from the original here: Alexander, A. "Tail-Recursive Algorithms in Scala" [https://alvinalexander.com/scala/fp-book/tail-recursive-algorithms]. Accessed October 2019.

2.3.2/ Aesthetics

It has been argued as an "elegant" language – Scala has first-class functions which not only allow for defining and invoking, but also for passing around as values when expressed as unnamed literals.[V] With this approach, functions can be used all around the codebase, not restricting itself to a specific context or type. Similarly, anonymous functions are supported.

Scala is much cleaner than Java, and its precise syntax means that programs tend to be comparatively shorter. As less time may be devoted to development, it suggests that there is more available for those unfamiliar to comprehend the code.

2.3.3/ Pattern matching[VI]

Alluded to before, pattern matching is a construct where a value is checked against a pattern; in successful cases, it can be deconstructed. It is an alternative to `if` statements and provides a compelling alternative to Java's `switch` statements.

In more depth, the `match` keyword succeeds a value and should offer at least one case to compare against.

```
val matchTest: String = (x: Int) => x match {
  case 1 => "one"
  case 2 => "two"
  case _ => "other"
}
```

Figure 2.3, Pattern matching: An integer x can be passed into the matchTest() function and its value is compared with each case – if there is not a successful match, the compiler progresses onto the next case. The final underscore (_) case catches all other possible values in the Int class. Passing in matchTest(3) would return "other".

The mechanism can be extended further, matching on case classes as well as type. Pattern matching is not available in Java.[VII]

[V] Source: "Scala Intro for Spark, v3" [http://mse-bda.s3-website-eu-west-1.amazonaws.com/lectures/BDA%20Lc06%20ScalaIntroForSpark-Part2.pdf]. Accessed October 2019.

[VI] The example in this section is adapted from here: "Pattern Matching | Tour of Scala | Scala Documentation" [https://docs.scala-lang.org/tour/pattern-matching.html]. Accessed April 2020.

[VII] It should not be confused with matching regular expressions against a text in Java, where the `Pattern` class derives the term pattern matching, though in a completely different context. Consult here: "Java Regex - Pattern" [http://tutorials.jenkov.com/java-regex/pattern.html]. Accessed April 2020.

2.3.4/ Algebraic data types[VIII]

Algebraic data types complement pattern matching and structure data so that illegal states cannot be represented. They come in two basic types: product and sum.

```
final case class Foo(b1: Boolean, b2: Boolean)
```

Figure 2.4, Product ADT: The Boolean type (with arity 2) composed onto another Boolean produces four possible outcomes of Foo(true, true) through to Foo(false, false) (and an arity of 4).

Product types can act like a tuple, collecting together multiple values in one wrapper. Case classes, where comparisons can be made on structure,[IX] as opposed to reference, are often used to implement this type. The *product* in the name refers to computing its arity (the values it is possible to possess) by calculating the product of its composing types. Sum types calculate arity according to the sum of the composing type's arities.

```
sealed abstract class Command extends Product with Serializable

object Command {
  final case class Move(meters: Int) extends Command
  final case class Rotate(degrees: Int) extends Command

  def print(cmd: Command) = cmd match {
    case Command.Move(dist)    => println(s"Moving by ${dist}m")
    case Command.Rotate(angle) => println(s"Rotating by ${angle}°")
  }
}
```

Figure 2.5, Applying algebraic data types to an example: Say you wanted to model an object that can only move forward a specific number of metres and rotate to a specified degree, describing it with case classes only allows for these two types of movement. It also lends itself to easy pattern matching.

Algebraic data types are not directly representable in Java, though experiments have modelled lists through interfaces.[X]

[VIII] Examples obtained and definitions constructed from here: "Scala Best Practices – Algebraic Data Types" [https://nrinaudo.github.io/scala-best-practices/definitions/adt.html]. Accessed April 2020.

[IX] Source: "Case Classes | Tour of Scala | Scala Documentation" [https://docs.scala-lang.org/tour/case-classes.html]. Accessed April 2020.

[X] See here for more information: "Functional Programming for Java Developers, Part 2 - Algebraic Data Types" Source: [https://openhome.cc/eGossip/Blog/FunctionalProgrammingforJavaDevelopers2.html]. Accessed April 2020.

2.3.5/ Existence alongside Java

As it runs on the JVM (Java Virtual Machine), Scala has access to Java's extensive collection of libraries and frameworks – and the interoperability between the languages means that Scala developers can integrate their code with Java, and vice versa. Say, one wanted to import the `LocalTime` library in Java, all that is required is this:

```
import java.time.LocalTime

val currentTime: LocalTime = LocalTime.now()
```

Figure 2.6, Importing Java libraries: it is very easy to import Java libraries as they are fully integrated.

2.3.6/ Static typing and a powerful compiler

Scala is statically-typed – this means that variable types are ascertained at compile time, with the compiler determining if a given action is valid.[XI] Statically typed languages are beneficial to programmers, as it can help to reduce the number of mistakes and enable them to write "proper" code. Debugging is easier, too – as it is not a dynamic language (i.e. it does not utilise run-time operations) errors are flagged up earlier (before the program is run), and debugging is easier. The compiler is powerful at referencing and Scala's type inference for variables and functions is much better than Java – in this regard, you can avoid many of the errors associated with the latter.

```
val eggs: Int = 4
val nuts = 5
val pineapples: Short = 3
val minimumPineapples: Int = pineapples
```

Figure 2.7, Types: Because of type inference, the compiler knows that eggs is an integer, so the Int type annotation is unnecessary, as evidenced by the value nuts. The compiler can convert between values as long as there is no loss of precision. An error will always halt the program if are any issues.

[XI] Definition adapted from here: "What is Statically Typed? – Definition from Techopedia." [https://www.techopedia.com/definition/22321/statically-typed]. Accessed October 2019.

2.3.7/ `Option` handles `null` effectively

The `null` runtime exception can be completely omitted if Scala's design choices are taken advantage of. The container `Option` can be used when returning a value that could be `null` – instead of an object being returned if the associated function passes – and `null` if not – an instance of `Option` is sent, either of the `Some` or `None` Scala class. One does not have to worry at all about `null`s, as the function signature simply declares that an `Option` of some type `T` is being returned. It also has the advantage of providing more meaningful information to the consumer.

```
def optionExample(param: S): Option[T] = {
  try {
    // Some mutation of param
  } catch {
    case e NumberFormatException => None
  }
}

optionExample(valueReturned) match {
  case Some(v) => println(v)
  case None => "I am disappointed."
}
```

Figure 2.8, Option: The types S and T can be anything, for example String or Int. The second fragment pattern matches on the value returned by optionExample() to determine which block to follow.

2.4/ Motivations for teaching

Whilst Scala might appear to perform in a multi-faceted nature, why should it be taught to students?

Scala is a capable language that can at least attempt to ease the transition between Java and Haskell and, thus, the gap between object-oriented and functional programming. The fact that it is multi-paradigm means that it occupies a middle ground between the two coding styles – it is not, for example, purely functional like Haskell, and so there is some leeway allowed that will no doubt be encouraging for students new to the paradigm.

Its "beauty and elegance" [10] allows for great expression – Scala can be used to encourage budding developers to code comparatively beautiful and well-designed applications. With access to - and interoperability with – Java, the most popular programming language (TIOBE Index, October 2019 [11]), prospective learners will have access to a wide range of resources and be able to program better code that still is extensively compatible on a large number of devices.

Scala developers are also highly sought after in the workplace due to their rarity and it is the second-highest paid programming language in the UK, with average salaries reaching £85,000. [12] Businesses – like Twitter, LinkedIn and Intel – depend on the language for their expansive mission critical systems and students should be heartened to discover that there is potentially a great career ahead of them with a technological company of considerable scale. [1]

2.5/ Related work

2.5.1/ Teaching Scala to younger students

Kojo, an open source IDE supporting the Scala language, was the first Scala-based environment for young learners to be used as an alternative to Scratch. [13] Targeting, in Sweden, children as young as seven, and, in India, girls between 11 and 13 years of age, through programming challenges and three to five hour-long classes carried out weekly, respectively, Regnell and Pant noted that the visual aid of the turtle was a natural way to start. With just a limited subset of the language, it made concepts easier to understand: its "orthogonality" – only one article of interest is changed, the rest is unaffected – supports this idea, and verifies its capacity to provide an easy starting point. Translating this to the context of a university student would not require the visual element, but this experiment still proved that even Scala provides some sort of accessibility to a younger audience.

2.5.2/ Scala in higher education

Scala has already been used in higher education: Lewis uses Scala in the *Principles of Computer Science* modules at the Department of Computer Science in Trinity University, San Antonio, Texas. He indicates that though educational resources are currently "thin" [14], there are "well over a million" questions answered on programmer question and answer site Stack Overflow. This, indeed, is another motivation for contributing further to the landscape, mentioning that the language offered sound footing covering the concepts needed for an introductory course at early undergraduate level.

Since 2013, it has "worked well" [15] at Aalto University in Espoo, Finland, [16] where they make full use of the (declarative) functional and (the imperative) object-oriented paradigms in their first year *O1* module. They mention that they want to offer "something fresh and a little bit different", and that as well as, perhaps most importantly for this project, it is possible to design "pedagogically effective courses" that are capable of approaching the harder topics incrementally and carefully, the staff teaching the introductory programming are fond of the language. This will undoubtedly diffuse onto the students and be likelier to encourage a rounded, more enthusiastic, and enjoyable learning experience in the most ideal of cases. Lukkarinen indicates that it is "possible" to teach concepts from both paradigms using just a single language, thanks to a "relatively low" learning threshold for those acquainted with Java's syntax, and easily applicable concepts from other languages.

2.5.3/ Describing general concepts with Scala

WebLab, a "learning management system" [17], was used by van der Lippe et al. in teaching concepts of coding languages through Scala. Through definitional interpreters, values can be computed from programs expressed as abstract syntax trees to explicate the "mechanisms" underpinning the described concepts. The lightweight approach, requiring an unexpansive toolset, was deemed to be convenient, recognising its functional style with algebraic data types and pattern matching and its well-developed nature allowing for reusability.

Chapter 3 / Specification and planning

3.1/ General considerations

Scala can be approached in many ways. Closely mirroring the aims, we can hone into a short collection of concepts to be taught that would provide a sound introduction to, and simultaneously foundation for, further study in the language. From each viewpoint, the project seeks to answer the accompanying questions.

- Scala as a programming language
  - What is it?
  - What can it do?
  - What makes it preferable over Haskell?
- Scala as a multi-paradigm language
  - How does it present object-oriented programming (OOP)?
    - How does this compare to Java?
  - How does it present functional programming (FP)?
    - How does this compare to Haskell?
    - Can it be approached in a transitionary way from Java?
    - Can the same be said from Scala into Haskell?
    - Does it effectively communicate FP concepts in a language not as strict as Haskell?
- Scala as a teaching language
  - How can it be introduced to newcomers in an accessible fashion?
  - Will the language add to students' understanding of functional programming?
  - Which is the best order to learn the languages Java, Scala, and Haskell?
  - Could Scala be added to or replace Haskell in the CS-205 module?
- Scala as a secure language
  - What security does it afford?
  - Can it be used to write safe programs?

3.2/ Case study outline

The case study is composed of two parts, of which there are also two sections.

An introductory lecture was planned to cover the fundamental points of functional Scala, as well as easing the transition from Java. An accompanying lab session would allow students to look further into the topics explored in the lecture and try their hand at basic Scala.

A second lecture made available for students interested in looking at more challenging topics was aimed at those who are more adept at programming. Continuing with higher order function applications, implicits and interaction with the functional *Cats* library were looked at, as well as introducing the idea of using Scala as a secure language. Again, another lab session was created to study the topics practically.

The case study was planned to be seen as a dynamic process, with a general specification outlining what is thought to be needed at first, developing much further, later on in the process, with a full teaching package created.

3.3/ Evaluation criteria

To be able to draw any conclusions from the case study and the background research, the data needed to be analysed. This would determine whether or not the project enabled the aims to be satisfied.

A questionnaire was planned to be distributed after both streams (i.e. the single and double lecture/lab session combinations) to consider some of the aims, as well as more tailored questions. The questions were aimed to address some immediately considered thoughts:

- How do you feel about Scala?
- Do you understand the functional concepts now?
- How does Scala compare to Haskell in introducing these concepts?

It was envisaged that one distribution would involve filling out the questionnaire straight after the session, and the other pausing a short while after to allow the students to form opinions after careful deliberation.

3.4/ Planning the project

Every project requires planning – and this one is no exception to the rule. I sought to consider the extent of the project, expressing the basis of the motivations and detailing the proposed structure of task execution, as well

as mapping out where one should be by the project's end – and indeed when that might be the case.

3.2.1/ Steps and milestones

The parts were extracted from the specification and consolidated into distinct groups containing milestones, significant stages in the project by which progress could be measured.

3.2.1.1/ Part I: Background research

The first part involved collecting information to help inform the latter sections of the project. It consisted predominantly of secondary data, using journals, books and the internet to highlight good reasons for choosing this discourse.

Milestone 1: Completion of Background Research	
The following topics were investigated:	
I.I/	How to program in Scala in an object-oriented way
I.II/	How to program in Scala according to the functional programming paradigm
I.III/	The advantages and disadvantages of Scala
I.IV/	Is the language used for teaching? If not, can it be?
I.V/	Verification: Scala as a secure language

3.2.1.2/ Part II: Setting up and monitoring the project

This part pertained to the necessary tasks that would help initiate the project proceedings. Such was its nature, it was completed in tandem with the previous part and in no particular order.

Milestone 2: Completion of the Initial Document	
These activities completed the second milestone:	
II.I/	Discussions with project supervisor: occurring continuously throughout the project as a control to ensure that the project was being kept on track.
II.II/	Writing the *Initial Document*: this document allowed me to see where I was and where I was going. Delivered in November 2019, it provided the opportunity to plan out the whole project, detailing every part of the process and giving the first qualified picture of how it would look at the end.

3.2.1.3/ Part III: Planning the case study

The third part pulled together the gathered information to plan a case study to collect primary data about the researched topics. The planning

sought to frame the information into deliverables that helped comprehend it and prepare for analysis.

Milestone 3: Completion of Case Study-Planning

This section led to completion of the third milestone:

III.I/ Determining the format of the case study: I asked myself how it would be organised to ensure an efficient utilisation of the conducted research and what needed to be recorded.

III.II/ Preparation of the initial lecture: what actions needed to be collected before the lecture; what materials had to be created; and what needed to be considered to progress onto the accompanying lab session properly?

III.III/ Preparation of the first lab session: how would the session follow on from the initial lecture and effectively communicate the lectured topics in a practical manner?

III.IV/ Preparation of the advanced lecture: again, as the step before last, but this time tailored to more advanced programmers and keen learners.

III.V/ Preparation of the second lab session: how could it follow on nicely from the lecture and provide a challenging enough series of lab tasks to keep the students engaged?

III.VI/ Definition of the evaluation criteria: what points required reviewing to best assess the case study?

III.VII/ Designing of the evaluation collection: how would the case study be evaluated? What kind of questions should be asked in a supplementary questionnaire distributed to the students?

The steps completed so far also fed into the next milestone,

Milestone 4: Gregynog Presentation

where I had to prepare a short presentation in front of a group of fellow students, as well as a question and answer section.

3.2.1.4/ Part IV: Carrying out the first part of the case study

The logistical preparations needed to be thought about following on from the planning before the initial part of the case study could be conducted, i.e. the first lecture and lab session:

Milestone 5: Completion of the Initial Lecture and First Lab Session

These steps led to completion of the fifth milestone:

IV.I/ Booking of the venue(s): finding out where the lectures and labs would take place, looking for available rooms with appropriate facilities and the best time to conduct it.

IV.II/	Publicising of the lectures and lab sessions: advertising through various media to generate interest.
IV.III/	Conducting the first part of the case study: delivering the initial lecture.
IV.IV/	Conducting the first part of the case study: delivering the first lab session.
IV.V/	Student evaluation: distributing the evaluation resources, allowing the students to review the lectures and labs.

3.2.1.5/ Part V: Carrying out the second part of the case study

This part was approached dynamically as there were additional factors to consider ensuring the second part was conducted effectively.

V.I/	Brief self-assessment of the initial lecture: improving step II.IV further by appraising the initial lecture and looking for things that did not go as well as they could have and amending the advanced lecture to reduce the chances of something similar happening again.
V.II/	Brief self-assessment of the first lab session: similarly, bettering step II.V by adjusting the lab sheet if students struggled with any of the lab tasks and reducing ambiguity if things were unclear and could be rewritten.
V.III/	Assessing student feedback: using what students had to say about the first part and modifying, if necessary, the advanced lecture and final lab with their constructive comments from the evaluation resource.

The following parts could not be completed because unforeseen issues, including those relating to the coronavirus pandemic, meant that time was significantly negatively impacted.[XII]

V.IV/	Conducting the second part of the case study: delivering the advanced lecture.
V.V/	Conducting the second part of the case study: delivering the final lab session.
V.VI/	Student evaluation: distributing the evaluation resources to allow the students to review the lecture and lab.

Completion of those tasks would have led to the sixth milestone which, ultimately, was only part-delivered:

Milestone 6: Completion of the Advanced Lecture and Final Lab Session

[XII] See *4.4/ The coronavirus effect* for more details.

3.2.1.6/ Part VI: Evaluating the collected data

Once the case study was realised, I was able to critique and look at the whole project.

VI.I/	Self-evaluation of the case study: what did I think was done well and what could I perhaps have done better?
VI.II/	Further evaluation of the case study: using the student evaluations to assess the pros and cons of how the lectures and lab sessions went.

These case study evaluation exercises meant that the seventh milestone could be achieved –

Milestone 7: Completion of the Case Study Evaluation

– and form the basis for my primary data analysis. The eighth milestone,

Milestone 8: Completion of the Project

could then be completed with the next step:

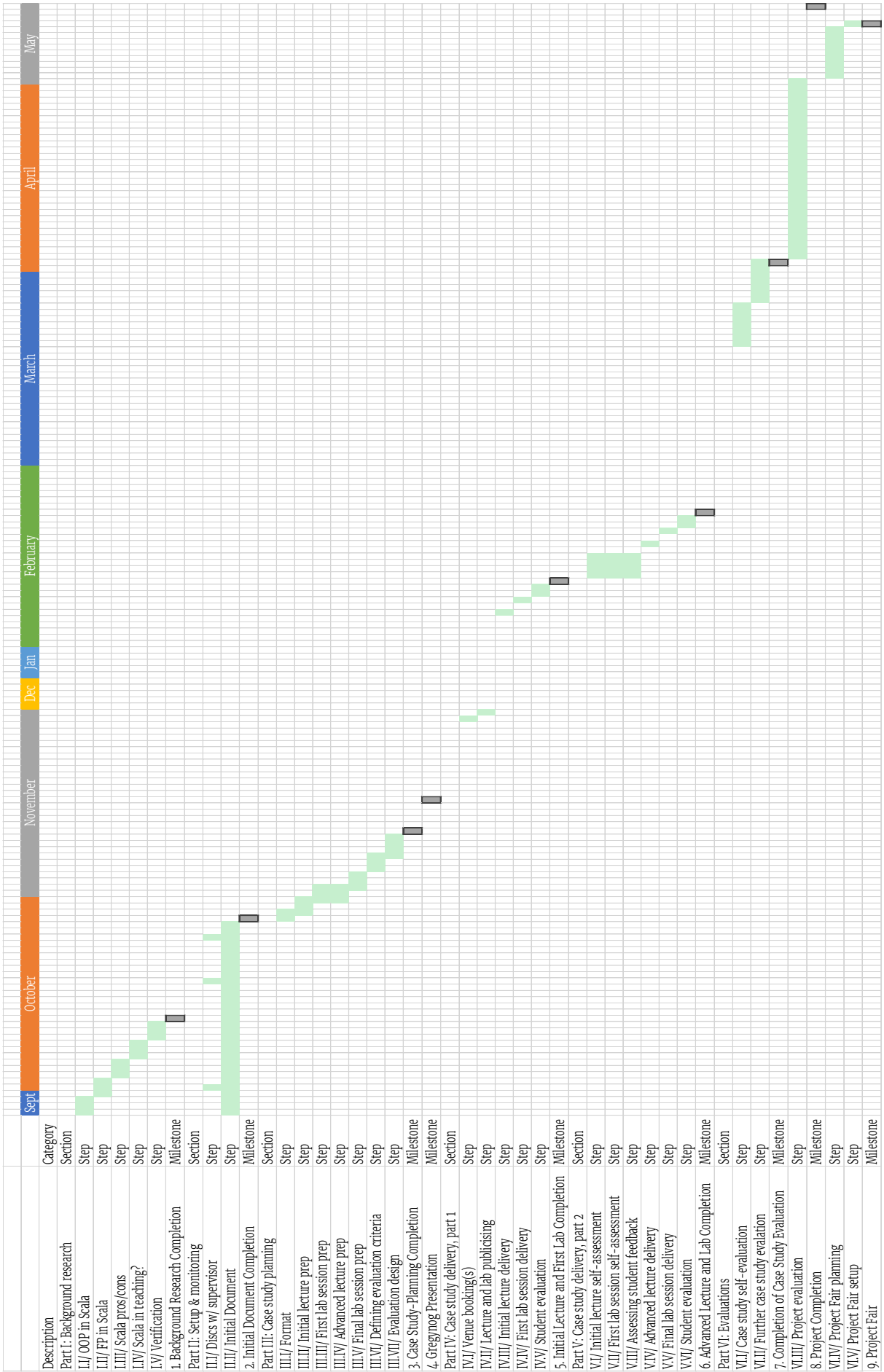
VI.III/	Evaluation of the project: for this, I asked myself: *how did the whole project go? Had the aims been addressed, and did I come to an acceptable conclusion?*

3.5/ Mapping out the project

The milestones and abundant steps have been expressed but can also be framed around a more concrete timetable: the project schedule can be illustrated through a Gantt chart, displaying the activities against time.[XIII] [16] The y-axis on the left details the tasks that need to be completed. The x-axis at the bottom comprises the dates that the project will cover. Each rectangular bar signifies an activity, and its shape and positioning can tell us:

- when the activity starts;
- how long it is estimated to take;
- its positioning amongst the other activities (is there an overlap?); and
- when the activity should end.

[XIII] See: "What is a Gantt chart? Gantt Chart Software, Information, and History" [https://www.gantt.com/]. Accessed October 2019.



Description	Category
Part I: Background research	Section
I.I/ OOP in Scala	Step
I.II/ FP in Scala	Step
I.IIII/ Scala pros/cons	Step
I.IV/ Scala in teaching?	Step
I.V/ Verification	Step
1. Background Research Completion	Milestone
Part II: Setup & monitoring	Section
II.I/ Discs w/ supervisor	Step
II.II/ Initial Document	Step
2. Initial Document Completion	Milestone
Part III: Case study planning	Section
III.I/ Format	Step
III.II/ Initial lecture prep	Step
III.IIII/ First lab session prep	Step
III.IV/ Advanced lecture prep	Step
III.V/ Final lab session prep	Step
III.VI/ Defining evaluation criteria	Step
III.VII/ Evaluation design	Step
3. Case Study-Planning Completion	Milestone
4. Gregynog Presentation	Milestone
Part IV: Case study delivery, part 1	Section
IV.I/ Venue booking(s)	Step
IV.II/ Lecture and lab publicising	Step
IV.III/ Initial lecture delivery	Step
IV.IV/ First lab session delivery	Step
IV.V/ Student evaluation	Step
5. Initial Lecture and First Lab Completion	Milestone
Part V: Case study delivery, part 2	Section
V.I/ Initial lecture self-assessment	Step
V.II/ First lab session self-assessment	Step
V.IIII/ Assessing student feedback	Step
V.IV/ Advanced lecture delivery	Step
V.V/ Final lab session delivery	Step
V.VI/ Student evaluation	Step
6. Advanced Lecture and Lab Completion	Milestone
Part VI: Evaluations	Section
VI.I/ Case study self-evaluation	Step
VI.II/ Further case study evalation	Step
7. Completion of Case Study Evaluation	Milestone
VI.III/ Project evaluation	Step
8. Project Completion	Milestone
VI.IV/ Project Fair planning	Step
VI.V/ Project Fair setup	Step
9. Project Fair	Milestone

3.6/ Risk analysis

As with all projects, there were possible risks (the likelihood that a hazard, something which is harmful, will negatively impact upon someone) that might have appeared. These hazards, and associated risks, require understanding and planning for to reduce the chances of the project being impaired. [18]

3.6.1/ List of hazards

The hazards that could have been encountered are outlined as follows:

I/ Students fail to understand the points trying to be communicated, thus no meaningful evaluations can be drawn.

II/ I might find the topic of verification with Scala too difficult to comprehend, thus reducing both the amount that can be taught and aims that can be addressed.

III/ People do not turn up to the initial lecture, restricting my research further.

IV/ Scala may not be available on the machines in the PC lab in the lecture – students are therefore unable to explore the language practically and interactively.

V/ Parts of the lecture – or indeed lab – overrun, meaning that students either miss out on information or are late for subsequent engagements. People scheduled to use the same room will also bear the knock-on effects.

VI/ I come unprepared to deliver the lectures or labs – information is either communicated poorly or not at all, and peoples' time is wasted.

3.6.2/ Risk assessment

With proper planning, the above hazards could, of course, be dealt with and necessary measures put in place. Therefore, for each possible hazard, it was assigned:

- a possible contractor – the person(s) at risk [XIV]
- a likelihood level – how probable the hazard could occur, on a scale of 1 (unlikely) to 5 (very likely)
- a severity level – how much it could impact the study, on a scale of 1 to 5 (low to high)
- a series of control measures, [XV] to diminish the effects of the hazards, categorised as:
  - avoidance, (A), making sure it does not happen
  - minimisation or mitigation, (M), of the risk and/or the hazard, reducing the effects when it does happen
  - exploitation, (E), taking advantage of the impacts when they are positive
- a subsequent likelihood level – the probability of the hazard occurring with (a)[XVI] control measure(s)[XVI] in place.

[XIV] For clarity, I shall be referred to as the co-ordinator, i.e. the individual carrying out the case study.

[XV] As well as those listed above, there are other ways to manage risks, albeit less common. One could accept the risk if it is considered as too unlikely to even waste time planning for, or if the impact is not that great. A risk could also be transferred to another party, although this only tends to be the case when the project is of a multi-compositional nature, i.e. there are other groups involved. This could range from transferral to another team to letting an insurance company assume responsibility. [26]

[XVI]

Hazard	Possible contractor	Likelihood	Severity	Control measure(s)	Subsequent likelihood
I	Students	4 (quite likely)	5 (high severity)	(A) The co-ordinator can endeavour to carry out detailed research not only into the topics to be taught, but also in effective ways to communicate the information. This research into teaching can compose of secondary data, through literature and the internet, and primary data, perhaps consulting lecturers to consider the best way to spread the information.	2 (not very)
II	Co-ordinator	3 (fairly likely)	5 (high severity)	(A) Detailed would certainly be very helpful. (M) One could take a moment aside to check literature or other resources to fully comprehend the difficult topic.	2 (not very)
III	Co-ordinator	4 (quite likely)	5 (high severity)	(A) Try to publicise better and create engaging promotional materials.	3 (fairly likely)
IV	All	5 (very likely)	5 (high severity)	(A) Check if the machines have Scala installed. If not, attempt to make contact with technical staff so that students will be able to program in the lab sessions.	1 (not likely)
V	Co-ordinator	4 (quite likely)	4 (quite severe)	(A) Plan well and undertake practice run-throughs to get a sense of timings. (M) Highlight topics not as important and reduce or remove them to compensate.	2 (not very)
VI	Students	4 (quite likely)	5 (high severity)	(A) Prepare more than is necessary to ensure that there is always something to do.	2 (not very)

Chapter 4 / Implementation

The approach of this project meant that a whole new application did not need to be written from scratch. Instead, the practical nature meant that existing code could be used and repurposed for the project's use.

The lecture and lab sessions were the deliverables, with code fragments specially written for each. However, the bank account example, for instance, was taken from an existing example (see `figure 4.4`).

All the produced resources can be found in a *GitHub* repository.[XVII]

4.1/ Introductory Lecture and Practical Session

4.1.1/ Summary

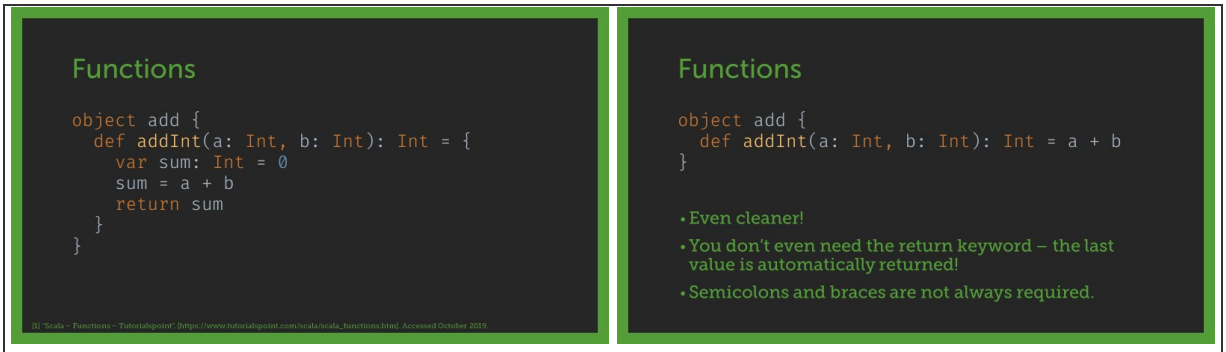
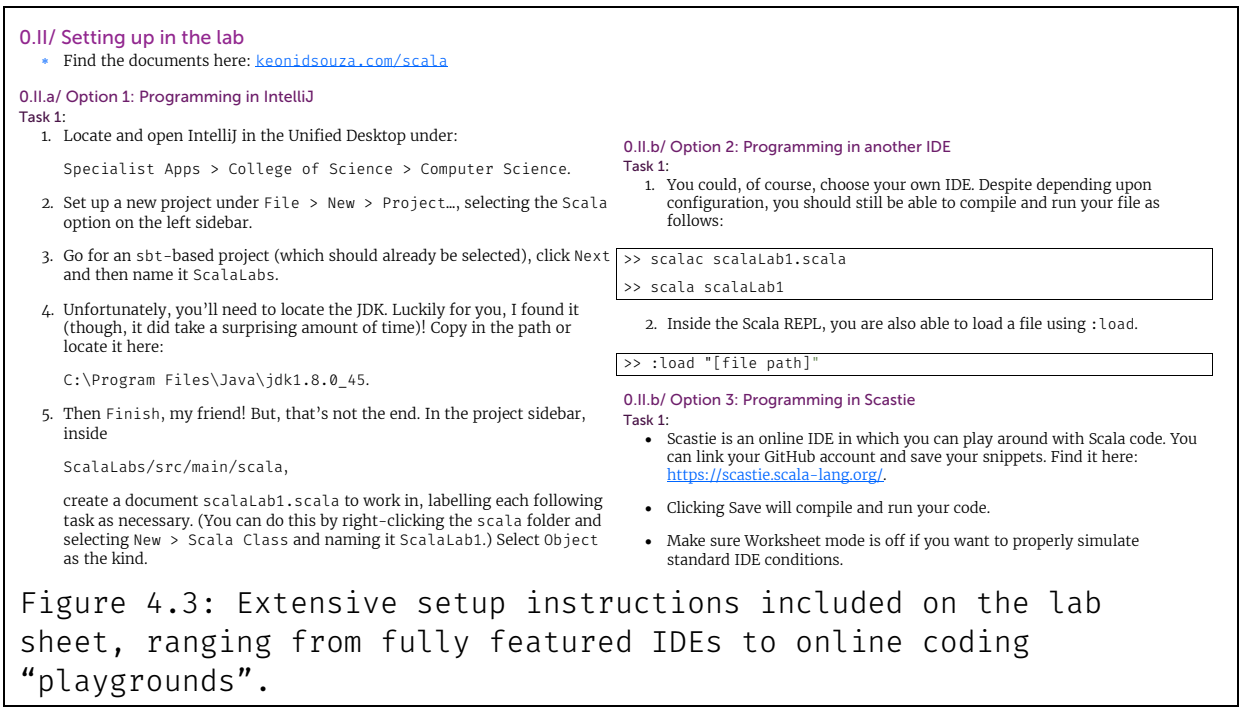


Figure 4.1: Slide extracts demonstrating how taciturn Scala can be. [19] The original presentation animated the gradual process that could be taken to reduce the character use. This is, of course, an extreme example, as there is a lot of unnecessary code in the former, but it is there solely for comparison.

Figure 4.2: Above, the higher order list functions map and filter were covered in the lecture, as well as flatten and flatmap.

[XVII] LegoKeoni/ScalaLabs on *GitHub*: https://github.com/LegoKeoni/ScalaLabs

The first lecture introduced functions in Scala, and basic I/O from the command line, progressing onto mutable and immutable lists and the higher order operations `map`, `filter`, `flatten` and `flatmap`. The object was to negotiate the transition from object-oriented to functional programming, so the interplay between both paradigms was explored in the accompanying practical session. Split into three, the first covered writing and reading, the second lists and higher order functions, and the third writing Scala functions to interact with existing Java code. This let the students apply the newly learnt ideas independently.



0.II/ Setting up in the lab

- Find the documents here: keonidsouza.com/scala

0.II.a/ Option 1: Programming in IntelliJ

Task 1:

1. Locate and open IntelliJ in the Unified Desktop under:

   `Specialist Apps > College of Science > Computer Science`.

2. Set up a new project under `File > New > Project…`, selecting the `Scala` option on the left sidebar.

3. Go for an `sbt`-based project (which should already be selected), click `Next` and then name it `ScalaLabs`.

4. Unfortunately, you'll need to locate the JDK. Luckily for you, I found it (though, it did take a surprising amount of time)! Copy in the path or locate it here:

   `C:\Program Files\Java\jdk1.8.0_45`.

5. Then `Finish`, my friend! But, that's not the end. In the project sidebar, inside

   `ScalaLabs/src/main/scala`,

   create a document `scalaLab1.scala` to work in, labelling each following task as necessary. (You can do this by right-clicking the `scala` folder and selecting `New > Scala Class` and naming it `ScalaLab1`.) Select `Object` as the kind.

0.II.b/ Option 2: Programming in another IDE

Task 1:

1. You could, of course, choose your own IDE. Despite depending upon configuration, you should still be able to compile and run your file as follows:

```
>> scalac scalaLab1.scala
>> scala scalaLab1
```

2. Inside the Scala REPL, you are also able to load a file using `:load`.

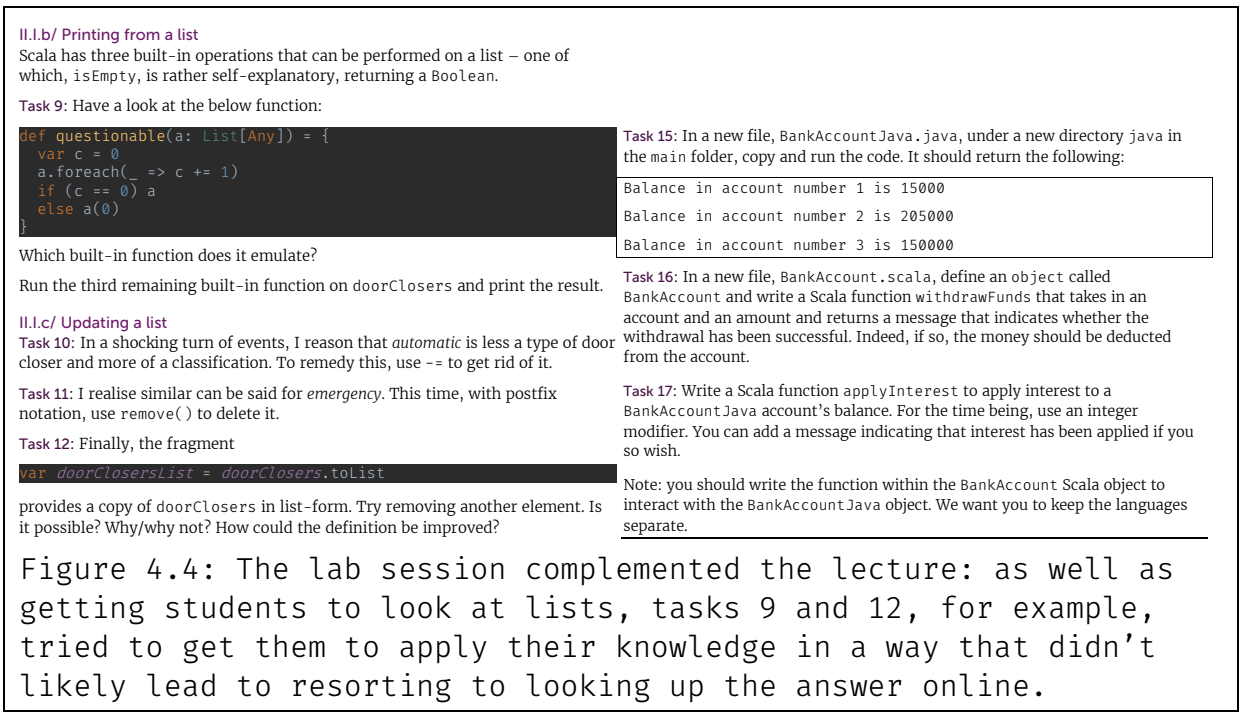
```
>> :load "[file path]"
```

0.II.b/ Option 3: Programming in Scastie

Task 1:

- Scastie is an online IDE in which you can play around with Scala code. You can link your GitHub account and save your snippets. Find it here: https://scastie.scala-lang.org/.
- Clicking Save will compile and run your code.
- Make sure Worksheet mode is off if you want to properly simulate standard IDE conditions.

Figure 4.3: Extensive setup instructions included on the lab sheet, ranging from fully featured IDEs to online coding "playgrounds".



II.I.b/ Printing from a list

Scala has three built-in operations that can be performed on a list – one of which, `isEmpty`, is rather self-explanatory, returning a `Boolean`.

Task 9: Have a look at the below function:

```
def questionable(a: List[Any]) = {
  var c = 0
  a.foreach(_ => c += 1)
  if (c == 0) a
  else a(0)
}
```

Which built-in function does it emulate?

Run the third remaining built-in function on `doorClosers` and print the result.

II.I.c/ Updating a list

Task 10: In a shocking turn of events, I reason that *automatic* is less a type of door closer and more of a classification. To remedy this, use `-=` to get rid of it.

Task 11: I realise similar can be said for *emergency*. This time, with postfix notation, use `remove()` to delete it.

Task 12: Finally, the fragment

```
var doorClosersList = doorClosers.toList
```

provides a copy of `doorClosers` in list-form. Try removing another element. Is it possible? Why/why not? How could the definition be improved?

Task 15: In a new file, `BankAccountJava.java`, under a new directory `java` in the `main` folder, copy and run the code. It should return the following:

```
Balance in account number 1 is 15000
Balance in account number 2 is 205000
Balance in account number 3 is 150000
```

Task 16: In a new file, `BankAccount.scala`, define an `object` called `BankAccount` and write a Scala function `withdrawFunds` that takes in an account and an amount and returns a message that indicates whether the withdrawal has been successful. Indeed, if so, the money should be deducted from the account.

Task 17: Write a Scala function `applyInterest` to apply interest to a `BankAccountJava` account's balance. For the time being, use an integer modifier. You can add a message indicating that interest has been applied if you so wish.

Note: you should write the function within the `BankAccount` Scala object to interact with the `BankAccountJava` object. We want you to keep the languages separate.

Figure 4.4: The lab session complemented the lecture: as well as getting students to look at lists, tasks 9 and 12, for example, tried to get them to apply their knowledge in a way that didn't likely lead to resorting to looking up the answer online.

The sessions both lasted sixty minutes – one after the other – and ran to the planned time. Seven participants joined the first run-through, with a further ten attending the second lecture.

4.1.2/ Qualification of content

It was decided that the most important starting point was to look at function declaration and definition, considering its focal role in the paradigm. The power function is a simple enough example that demonstrates the language's minimal expression and shows that its description can be streamlined through built-in functions.[XVIII] Introducing how to accept user input intended to provide positive comparisons with Java's interpretation and indicate towards Scala's lightweight approach.

Immutability – promoting unchangeable objects after creation – is a core aspect of functional programming, but mutable objects had to be discussed when coming from the world of Java. Thus, both types of list were introduced in the lecture: mutable `ListBuffers` and immutable `Lists`. Higher order functions – those which also pass in functions – were discussed through the four aforementioned list operations.

4.2/ Advanced Lecture and Practical Session

4.2.1/ Summary

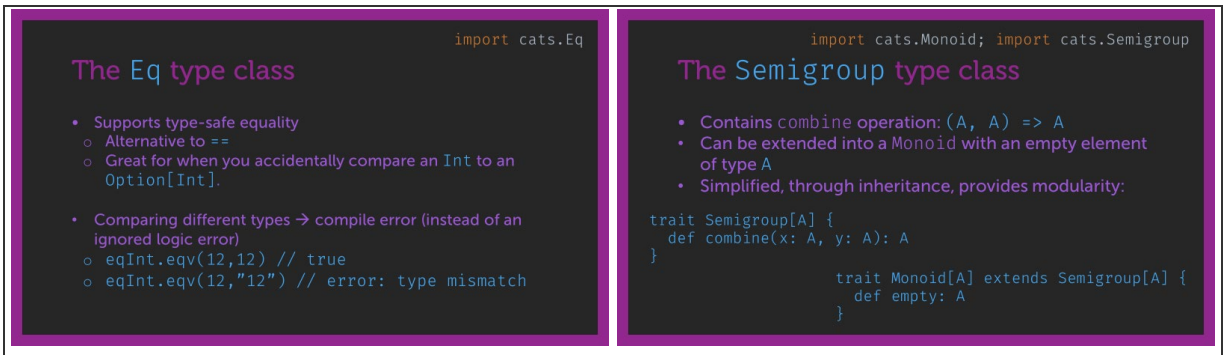


Figure 4.5: Slide extracts introducing the Eq type class, providing a safer alternative when comparing types, and Semigroups, effectively half a monoid. [20]

In the second series, the *Cats* library is explored, using type classes – which originated from Haskell – to provide new functionality to existing libraries, extending them without changing the original source and bypassing inheritance. Pattern matching, algebraic data types (ADTs) and

[XVIII] Here, `scala.math.pow(n,2)` as an alternative to `def square(n: Int): Int = n*n` removes the necessity of defining a new function.

monads were also touched upon, presenting the ideas through a library attempting to increase Scala's functional capacity.

4.2.2/ Qualification of content

The topics were chosen as they have already been studied in the computer science and software engineering degree programmes at the university, so students would be aware of them. Using *Cats* promotes the functional paradigm, trying not to tempt them to approach tasks in an object-oriented fashion.

4.3/ Data collection

The process by which to obtain the mood of the participants was through a questionnaire. The design for the first sessions was split into five sections, with the majority of the questions requiring a response on a five-point Likert scale (a commonly used question format where people respond to a statement using a defined scale). It ranged, predominantly, from "I strongly disagree" to "I very much agree" on a five-point metre, asking them how much they agreed with the provided statement. Each question provided an opportunity for elaboration if they so wished.[XIX]

#	Question Text	Response alphabet
Section 1: Initial thoughts on Scala		
1	What's your initial response to the language Scala?	5-point scale (1: I don't like it → 5: I really like it)
2	Had you heard of Scala before?	Yes, No, Don't know
3	How do you feel about Haskell?	5-point scale (1: I don't like it → 5: I really like it)

The responses for the following questions all corresponded to a five-point scale, with 1 being "I strongly disagree" and 5 representing "I very much agree".

#	Question Text
Section 2: The lecture and lab session	
4	"The lecture presented the material in a clear fashion and was easy to follow."
5	"The lecture presumed the right amount of difficulty for someone who has had recent experience of functional programming in Haskell."

[XIX] The questionnaire is available to view here: https://keonidsouza.com/scala

#	Question Text
6	"The lecture was enjoyable."
7	"The lab session provided clear, easy-to-follow instructions."
8	"The lab session presented tasks over increasing levels of difficulty, sufficiently challenging me."
9	"The lab session did not feel overly long."
10	"The lab session nicely complemented the lecture material."
11	"When required, I was given a suitable amount of assistance in the lab session."
12	"I think another lecture and lab session would be great to delve into more functional programming concepts."
Section 3: Scala and Java	
13	"Scala offers a nice transition from Java to Scala."
14	"The functional and object-oriented concepts sit pleasingly side-by-side in Scala."
15	"I like how I can interact Scala code with Java code without having to rewrite anything (or, at the least, very little)."
16	"The transition from programming in Java to programming in Scala is good."
Section 4: Scala vs Haskell	
17	"Having used Haskell in CS-205, I feel Scala is easier to understand than Haskell from what I looked at in the lecture/lab."
18	"Scala is easier to write than Haskell."
19	"Scala would act as a great transition language between the world of object-oriented programming in Java and functional programming in Haskell."

The final section provided the last opportunity to give feedback.

#	Question Text	Response alphabet
Section 5: Bringing this to an end		
20	"I think that Scala should be taught at the university."	5-point scale (1: I strongly disagree → 5: I very much agree)
-	This time, please elaborate on your final answer.	Longer response text box
21	Is there any other feedback you'd like to impart?	

I also spoke to all participants in the lab session to gain an immediate response to my lecture and provided tasks. Students were very open, and I took note of all their feedback. A questionnaire for the second part of the case study was not produced due to obstacles which shall be elaborated on in the next section.

4.4/ The coronavirus effect

Two lectures and two practical sessions were planned, however only the first of each could be delivered. This was down to water damage to the computer laboratory, and the inability to find a replacement venue – it was planned that the first session would be run again as availability and interest was split amongst two dates. Preceding a brief run-through of the practical element, eventually the second attempt at the first lecture was conducted on the video conferencing application *Zoom* due to the Covid-19 pandemic necessitating government-enforced social distancing measures – this was all after a few weeks of assessment, where I analysed how my lecturers were conducting their lectures in the move to distance learning, and applying the best of their techniques to my setting to optimise deliverability.

It was judged that there would be little time to carry out the second lecture and lab session, so the materials exist alongside the former, but have not been employed.

Chapter 5 / Conclusions

5.1/ Achievements

5.1.1/ Introducing Scala into higher education

This project sought to introduce Scala to the higher education curriculum at the university's computer science department – currently, to the best of my knowledge, the only other institution in the United Kingdom using the language in this fashion is the University of Oxford in their first-year *Imperative Programming* course, complementing Haskell in the module concerning the functional equivalent. [21] With encouraging feedback to my introduction to the language, I hope to have provided a crucial starting point for further development and integration into the computer science courses offered by the department.

5.1.2/ Producing an educational resource

Drawing from a teaching module I took in the first semester, I was able to design a lecture series that asked if the content was engaging enough. The lecture series assumes a certain level of competence – with many agreeing that "[the] lecture presumed the right amount of difficulty for someone who has had recent experience of functional programming in Haskell" – and treads well, according to feedback, between varying difficulties. It was not easy to produce a resource that satisfied these criteria, though the outcome is proof of its competence.

5.1.3/ Getting students to consider functional programming

Conversing with my peers, they generally had a similar response to the functional way of programming in Haskell to mine, i.e. one of getting through the declarative programming module and leaving it behind. Upon further thought, it seemed to me to be a wasted opportunity, so I attempted to frame the paradigm within a context that would be more accessible and convincing. With Scala, compared to Haskell, there was a greater level of interest in the functional concepts and they appreciated the light-hearted delivery and possible career prospects.

5.2/ Analysis

5.2.1/ Addressing the aims

This project has gone through Scala as a programming language. It is a multi-paradigm language that allows for a different way into functional programming: as noted by a questionnaire respondent, they welcomed the "functional concepts with a readable and intuitive syntax." This, along with increased employability opportunities, made it preferable to Haskell.

Someone I asked also agreed "[the] functional and object-oriented concepts sit pleasingly side-by-side in Scala." The simple integration of Java libraries and invocation of their associated functions provided a nice bridge between the languages, whilst the functional approach could generally derive agreeably from Haskell: the functional elements translate well and its decreased purity against it does not impede in educating the concepts in Scala. Its lightweight syntax still supports higher order functions and currying.

With a background in Java afforded due to the university's curriculum, introducing the language was more accessible than in other contexts, and this was helped with a year's experience coding in Java. Some argued that Scala should replace Haskell in the CS-205 module, and another thought it should follow Haskell, occupying a "good sweet spot" between object-oriented and functional programming. Otherwise, using Scala as a transitionary language was largely popular, with over three-quarters of those I questioned agreeing with the statement.

Scala does not enjoy the same purely functional qualification as Haskell. But there are ways in which it still maintains a high security level: its statically typed nature means that – at compile-time – abstractions are "used in a safe and coherent manner", [22] ensuring type safety. Due to the interoperability sustained with Java, Scala has to use an "explicit strong reference type" [23] when dealing with `null` references, though this can be remedied with an `Option`, transforming the partial function into a total function. `Option` is a powerful type that can provide a backup if one is unsure if a value will be returned, thus making the program safer when this side-effect is not disregarded.

It has a "good default security", as found by Dwarampudi et al. [24]: in a comprehensive comparison with nine other languages, it matched Haskell in terms of best default security, alongside Java and VB.NET. Scala has automated garbage collection, and its exception handling is of an implicit nature, affording additional security through the JVM's Security Manager. Usage in web applications is popular as performance is generally unsacrificed

at the expense of "security and robustness", and though reflection is supported, where code can inspect itself, Scala still maintains a high level of security.

5.2.2/ User interaction

During the practical exercise, I sat with the students and observed and discussed their accessibility with the new interface. As many noted the familiarity with already-observed languages (compare with Java's syntax and Haskell's functional nature), they were able to get started in a short timeframe. A couple of subjects struggled with setting up the lab using the instructions on the guidance sheet, but this was later attributed to misreading the text. Clarity was improved in later versions to minimise this kind of issue, and alternative coding environments were provided, including the lightweight online IDE *Scastie* which lets the users start work immediately – as long as they have a good internet connection.

5.2.3/ Problems with multi-paradigm support

Providing concessions for object-oriented and functional programming, whilst beneficial for exploring different paradigms in a common language without having to learn another, does mean that a choice has to be made, both consciously and subconsciously, by the programmer.

When using Scala in a coding assignment within a module concerned only with functional programming, there is the temptation to blend into the other referenced paradigm, and when evaluating such submissions, marks would have to be deducted, as appropriate, to penalise the incorrect approach (within the context of the module). This is something that is difficult to prevent, but the risks can nonetheless be minimised: if students are educated expansively on the differences between the paradigms, they should be of a capable mind to determine which they are using. Lab classes could provide a good opportunity to exercise the subconscious into telling the paradigms apart and reduce the chances of this happening.

Although this would require more provisions to be made from the point of the module co-ordinator(s), what is sacrificed through circumventing a purely functional language like Haskell – where you can only code functionally – reaps rewards in other ways. Students can become more aware in their programming – if they are able to distinguish between paradigms, they can become better programmers through avoiding the temptation of imperative programming. There are also many more job opportunities available: searches for functional programming roles in industry return far more Scala results –there were 1522 permanent jobs citing Scala as a development language according to insights firm IT Jobs Watch, compared

to 69 in the six months leading up to May 2020.[XX] If purity is preferred over employability, this is not as convincing a point to students wanting to pursue a career in software development: Scala provides a respectable middle ground, with companies such as Netflix (in their search algorithms and recommendations) and Airbnb (due to its scalability, fault tolerance and fast process times) implementing the language without having to sacrifice too much in terms of performance. [25]

5.2.4/ Obstacles and their management

The scale of the project, at times, I underestimated and, for which, I did not allow enough time, in retrospective: it took me longer than I thought to construct, suitably, the introductory sessions so that it was the right balance between accommodating those who struggled with the ideas surrounding functional programming, and satisfying enough those who would have found it too easy. My supervisor helped devise an apposite template, which I then described in ways I hoped were engaging to students whilst still applying and framing the concepts in useful ways. With light doses of humour, and an appropriate level of enthusiasm, I aimed to replicate the ostensibly successful – for seven years – approach such is that found at Aalto University[XXI].

It was the first they had seen of Scala for the vast majority of participants and it was pleasing for students to take my introduction so well. Quoting from informal conversations I had, they liked Scala's "familiar" approach and preferred this experience over that of Haskell – they were able to "get up and running" fairly quickly and one, confirming a general consensus, stated that Scala should "replace one of the languages in [the] declarative programming [module]".[XXII]

Whilst I was happy with the physical response – many were keen to express their apparent delight at this language new to them – there was little engagement with the questionnaire. In seeking to address criticisms of "only [measuring] immediate reactions to learning a new language"[XXIII], I added a delay to the first cohort, allowing them a week before the questionnaire was online to gather their thoughts and think more deeply about the comparisons with Haskell. This, it is now obvious, did not prove to be a successful endeavour. Despite communicating via university email, the response was

[XX] Across all permanent jobs advertised in the UK, Scala has a more than 180% advantage over Haskell. [27] [28]

[XXI] cf. "something fresh and a little bit different" [15]

[XXII] At least four students opted for Scala's inclusion, when prompted.

[XXIII] Comments made in response to the preliminary survey design in my *Initial Document*, a paper produced in November 2019 to record progress before an accompanying presentation at the third-year retreat, the *Gregynog Colloquium* in mid Wales.

paltry. In hindsight, the survey could have been designed better: spread across five parts, though it mainly included Likert scale responses, the sheer extent of the whole may well have proved to be a reason to retreat for prospective respondents, with one telling me that it was "too long". Nonetheless, the more interactive approach of communicating directly through the medium of voice afforded me all I, personally, needed to construct an ample argument – though, perhaps, not with adequate academic rigour, one should admit.

Problems with the conducting venue – there was a water leak in one of the lab rooms – meant that there was difficulty in holding the second run-through of the first lecture and practical. Having to postpone it due to conflicting arrangements led to ambiguous email communication and, both interest in and the likelihood of, holding the sessions became increasingly unlikely.

The state of the world provided an unlikely second obstacle in the project, through the coronavirus pandemic, however, once all students become acclimatised to the new normal of online lectures I was able to deliver the lecture again a couple of months after the first. Though this was not without some modifications, as the practical element could not be as optimally delivered as it would have been if I were in a singular room with the students working away independently on separate, but geographically close, screens. On reflection, perhaps adapting the lab sheet into a quiz using a service like *Kahoot!* would have been more productive and a more reliable indicator of understanding.

Finally, and this is constant regret, I wish I had been more thorough in my research and completed this part earlier in the process. It was only towards the latter portion of the project that I found most of my scientific references, as these were severely lacking at the start, and these ended up being considerably more helpful than the web resources.

5.3/ Further work

Were this project fortunate to have been afforded more time, I would have carried out the second part of the course and observed first-hand students' responses to more difficult concepts in Scala. A third part might also have been useful in including parts that I had to omit for brevity and reforming it all into an interactive tutorial series to host on, say, YouTube would have been an aspirational remark, enabling the inclusion of students with less or indeed no prior knowledge of the functional paradigm.[XXIV]

Integrating into the lecture series, on an experimental basis, some Scala alternatives into the existing declarative programming module would have been a great thermometer test to gauge students' understanding, comparing and contrasting responses.

Having enrolled onto a placement module where I was seconded at a primary school to educate pupils about computer science, were the setting a secondary school it would have been an interesting test to see if the paradigm enforced a comprehensibility at an early stage in their computing education and if they became better programmers as a result (though this would be the foundation for a longer-form case study that would have required exponentially greater planning and arrangements).

I only explored the *Cats* library in my project as it seemed appropriately comprehensive and there was a lot of supporting documentation to aid my research. I would also have liked to look at *scalaz*, an alternative functional programming library, if the time allowed.

[XXIV] This, of course, does not exclude the possibility of doing so, regardless.

Chapter 6 / Bibliography

[1] "Scala – Reviews, Pros & Cons | Companies using Scala," [Online]. Available: https://stackshare.io/scala. [Accessed October 2019].

[2] "Cats: Lightweight, modular, and extensible library for functional programming," [Online]. Available: https://typelevel.org/cats. [Accessed October 2019].

[3] M. Odersky, L. Spoon and B. Venners, Programming in Scala, California: Artima Press, 2008.

[4] M. Odersky, P. Altherr, V. Cremet, B. Emir, S. Maneth, S. Micheloud, N. Mihaylov, M. Schinz, E. Stenman and M. Zenger, "An Overview of the Scala Programming Language, 2e.," 2015.

[5] B. Venners and F. Sommers, "The Origins of Scala: A Conversation with Martin Odersky, Part I," 4 May 2009. [Online]. Available: https://www.artima.com/scalazine/articles/origins_of_scala.html. [Accessed October 2019].

[6] M. Odersky, "Scala's Prehistory," 2008. [Online]. Available: https://www.scala-lang.org/old/node/239.html. [Accessed October 2019].

[7] R. S. Aggarwal, "10 top Programming Languages in 2019 for Businesses," 2019. [Online]. Available: https://codeburst.io/10-top-programming-languages-in-2019-for-developers-a2921798d652. [Accessed October 2019].

[8] M. Odersky, P. Altherr, V. Cremet, G. Dubochet, B. Emir, P. Haller, S. Micheloud, N. Mihaylov, A. Moors, L. Rytz, M. Schinz, E. Stenman and M. Zenger, "Scala Language Specification," 2006. [Online]. Available: https://www.scala-lang.org/files/archive/spec/2.13. [Accessed October 2019].

[9] "Uses of Scala | Top 10 Useful Uses Of Scala In Real World," [Online]. Available: https://www.educba.com/uses-of-scala/. [Accessed October 2019].

[10] A. Devlin, “"I was reminded of how I fell in love with Scala’s beauty and elegance" - Signify Technology,” [Online]. Available: https://www.signifytechnology.com/blog/2019/06/i-was-reminded-of-how-i-fell-in-love-with-scalas-beauty-and-elegance. [Accessed April 2020].

[11] “TIOBE Index | TIOBE – The Software Quality Company,” [Online]. Available: https://www.tiobe.com/tiobe-index. [Accessed October 2019].

[12] S. Butcher, “The UK's best-paying technology jobs and coding languages,” [Online]. Available: https://news.efinancialcareers.com/fi-en/3002150/pay-for-developers-uk. [Accessed October 2019].

[13] B. Regnell and L. Pant, “Teaching programming to young learners,” 2014.

[14] M. C. Lewis, D. Blank, K. Bruce and P. Osera, “Uncommon Teaching Languages,” 2016.

[15] “Frequently Asked Questions | Ohjelmointi 1 | A+,” 2018. [Online]. Available: https://plus.cs.aalto.fi/o1/2018/wNN/faq/#the-scala-language. [Accessed April 2020].

[16] A. Lukkarinen, “Supporting Media Computation in Programming Education,” 2016.

[17] T. van der Lippe, T. Smith, D. Pelsmaeker and E. Visser, “A Scalable Infrastructure for Teaching Concepts,” ACM, 2016.

[18] “What is the difference between a ‘hazard’ and a ‘risk’?,” [Online]. Available: https://worksmart.org.uk/health-advice/health-and-safety/hazards-and-risks/what-difference-between-hazard-and-risk. [Accessed October 2019].

[19] “Scala – Functions – Tutorialspoint,” [Online]. Available: https://www.tutorialspoint.com/scala/scala_functions.htm. [Accessed October 2019].

[20] N. Welsh and D. Gurnell, Scala with Cats, Brighton: Underscore Consulting LLP, 2017.

[21] "Imperative Programming Parts 1 and 2," [Online]. Available: https://www.cs.ox.ac.uk/teaching/courses/2019-2020/imperativeprogramming1/. [Accessed April 2020].

[22] "Introduction | Tour of Scala | Scala Documentation," [Online]. Available: https://docs.scala-lang.org/tour/tour-of-scala.html. [Accessed April 2020].

[23] Anler, "Type safe Scala — Don't use null - Anler - Medium," 3 June 2017. [Online]. Available: https://medium.com/@anler/type-safe-scala-dont-use-null-3a1420b2a9d8. [Accessed April 2020].

[24] V. Dwarampudi, S. S. Dhillon, J. Shah, N. J. Sebastian and N. S. Kanigicharla, "Comparative study of the Pros and Cons of Programming languages," 2010.

[25] A. Periel, "How Tech Giants Use Scala — SysGears," 23 September 2019. [Online]. Available: https://sysgears.com/articles/how-tech-giants-use-scala/. [Accessed April 2020].

[26] "5 Ways To Manage Risk," [Online]. Available: http://www.dbpmanagement.com/15/5-ways-to-manage-risk. [Accessed October 2019].

[27] "Scala jobs, average salaries and trends for Scala skills | IT Jobs Watch," [Online]. Available: https://www.itjobswatch.co.uk/jobs/uk/scala.do. [Accessed May 2020].

[28] "Haskell jobs, average salaries and trends for Haskell skills | IT Jobs Watch," [Online]. Available: https://www.itjobswatch.co.uk/jobs/uk/haskell.do. [Accessed May 2020].

A final word

Many thanks for taking the time to read this report. I hope it was an acceptable balance between interesting and uninteresting.