Scala First:
Lessons from 3 student generations

Bjorn Regnell

Dept. of Computer Science, Lund University

Who am I

- Björn Regnell (skip the strange dots and just call me Bjorn)
- Professor in Software Engineering
- Dept. of Computer Science, Engineering Faculty LTH, Lund University, Sweden
- Research focus: Software Requirements Engineering
  - **`http://cs.lth.se/bjorn-regnell`**
- Scala programmer since 2010
- Teaching Scala using Kojo IDE in primary school to kids and teachers since 2011
  - **`http://www.kogics.net/kojo-download`**
- Teaching Scala at university level since 2016
  - **`https://github.com/lunduniversity/introprog`**

Acknowledgements

Many thanks to

- hard-working students
- enthusiastic teaching assistants
- supportive colleagues
- contributors to open source course material
  - `https://github.com/lunduniversity/introprog/blob/master/contributors.tex`
- the fantastic Scala community

Scala first lessons

Agenda:

- Why Scala first?
- How did we implement Scala first?
- What did we learn?
- The road ahead

Why Scala first?

About our Scala students at Lund Univesity

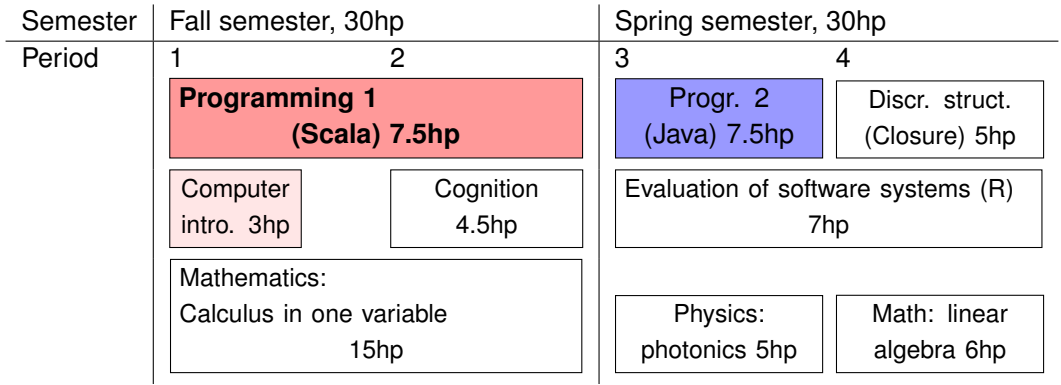
- Enrolled at the 5 year program in Computer Science & Engineering (CSE) at LTH
- Taking their first programming course 7.5 ECTS credits during their first semester
- 100% are fluent in Swedish
- More than 80% are 19–22 years old
- Around 15% are female
- Around 65% have programmed before (C#, Java, Python, Javascript, ...)
- Very big span of pre-knowledge
- Increasingly selective admission: 136 accepted out of 1337 applicants (2018)

History of first languages at Lund University

First languages for CSE students at LTH:

(Algol)	(pre-history using punch cards)
Pascal	1982, CSE program started
Simula	1990
Java	1997
Scala	2016

The first year of the CSE program since 2016



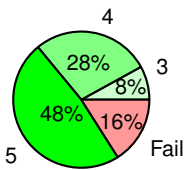
A study year is 60 ECTS higher education points (hp) = 40 weeks of full-time studies.

The Java first situation (2015)

I took over as course head of the first programming course in 2015 and ran it (almost) as usual with Java first, while in parallell heading the development of the Scala first course material as an open source community project with colleagues and senior students.

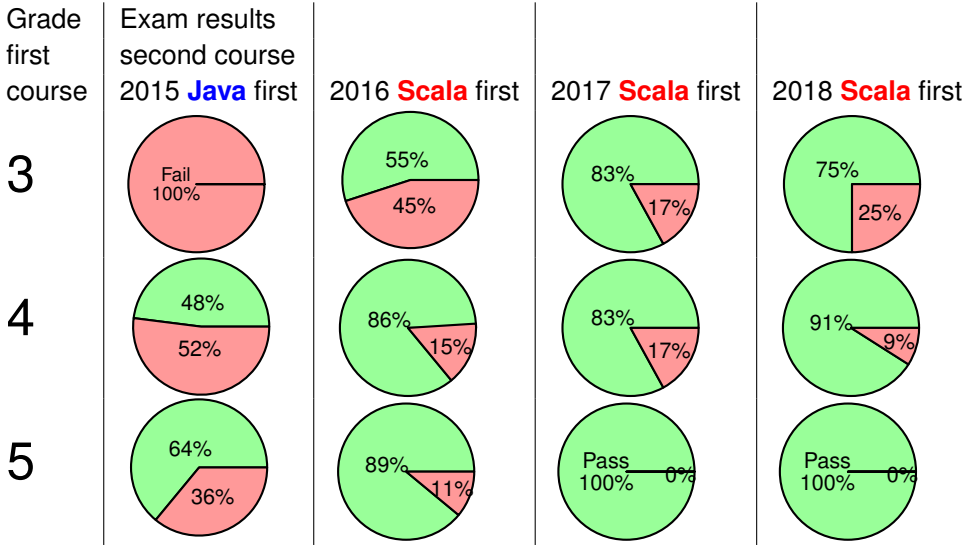
Good:

- Course very popular
- Exam results very good:



Bad:

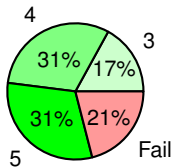
- Competent students not challenged
- Poor results in second course



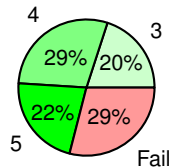
Scala first – a big success! :)

- Exam results in second course much better: increased learning outcome
- Course evaluations are still very positive: increased student engagement
- All students are challenged, independent of pre-knowledge
- Students find Scala interesting and novel
- Exam results:

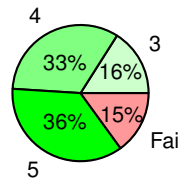
2016



2017



2018

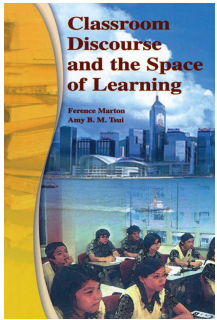


Why Scala first?

My hypotheses supported by my experience since 2011 (but no systematic empiricism):

- Easy for beginners and interesting for non-beginners (depending how you teach...)
- Regular semantics is easier for beginners; e.g. value types are real objects
- Concise (but not too concise), expressive: do interesting things with small programs
- Powerful, simple-to-use standard library: do interesting things with small programs
- Static typing help finding bugs: the compiler is your friend
- Interactive learning in the Scala REPL: understand expressions step by step
- Modern, evolving language: exciting for students and teachers
- Multi-paradigm: excellent for mixing imperative, object-oriented, functional

Pedagical support for multi-paradigm language



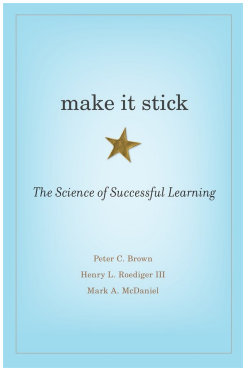
Marton & Tsui (2004)

- Learning is the process of being capable of *doing* something
- Learners construct a language by discerning parts and wholes
- Patterns of variation:
  - **Contrast**: to experience something, you need to experience something else to compare with
  - **Generalisation**: to understand a concept you need to be able to abstract from irrelevant features
  - **Separation**: to experience a certain aspect of something you need to vary that aspect while other aspects remain invariant
  - **Fusion**: to be capable of doing something advanced you need to experience several critical aspects at the same time

Scala enables contrasting, generalisation, separation and fusion across paradigms.

How did we implement Scala first?

Pedagical ideas behind course design



Brown et al. (2014)

What did we learn?

The road ahead

LUND
UNIVERSITY