

CS 94SI: Cross-Paradigm Programming with Scala

Spring Quarter Course: 1 unit, C/NC only

Wednesdays 4:15pm – 5:45pm

Gates B12

Course Leaders:

Ben Newman

Jorge Ortiz

Julien Wetterwald

Faculty Sponsor:

Mehran Sahami

Staff E-mail:

cs94si-spr0708-staff@lists.stanford.edu

(Only for questions about course logistics or grading)

Scala Mailing List:

scala-user@listes.epfl.ch

(For questions about Scala. To subscribe, visit: http://listes.epfl.ch/doc_en.cgi?liste=scala-user)

Goal:

By this end of this course, when you are confronted with a software problem, you should be able to:

- analyze it in terms of several different programming paradigms
- evaluate the advantages and disadvantages of each approach

Description:

Programming language design as exemplified by Scala, a new type-safe language that combines functional and object-oriented aspects at the language level and targets the Java Virtual Machine. Scala's type inference system and the promise of "scalable component abstractions" for software systems. Applications to concurrent programming and domain specific languages. Weekly readings.

Prerequisites: academic exposure to at least two programming languages; CS107 recommended.

Assignments:

There will be 4-7 mandatory semi-weekly assignments throughout the course of the quarter.

Assignments are due by e-mail (to the staff mailing list, cs94si-spr0708-staff@stanford.edu) before the start of class the week after the homework is assigned.

Grading Policy:

The course is graded on a C/NC basis only. To receive a passing grade, you must complete at least all but one of the weekly assignments and participate actively in class. (**Important Note:** The course description on Axxess incorrectly states that weekly assignments are optional and that there is a mandatory final project. There will be no final project. Weekly assignments are not optional.)

Collaboration Policy:

You are encouraged to discuss your ideas with others. However, you should work on your programs independently and hand in original code. If you receive significant help from others, you should

attribute them when handing in your assignments.

Class Mailing List:

Signing up for the class on Axxess will automatically subscribe you to the class mailing list. Please make sure to do so as soon as possible.

Auditors:

Auditors are welcome, but please make sure to sign up for the class mailing list manually by visiting: <https://mailman.stanford.edu/mailman/listinfo/cs94si-spr0708-students>

Scala Version:

The class will use the Scala 2.7.0-final release of the language. It can be obtained from: <http://www.scala-lang.org/> Instructions on how to install Scala will be included in the first assignment.

“Scala *lift off*” Conference:

On Saturday, May 10th a Scala/lift conference will be held in San Francisco. The conference is **free** for students (free lunch is provided). This is a fantastic opportunity to meet many of the people in the Scala and lift communities, including Prof. Martin Odersky. Please register for the conference at:

<http://scalaliftoff.com/liftoff/>

References:

Everything you need to know will be covered in class and in the lecture notes, but these documents may be helpful as you explore Scala:

- Scala by Example, by Martin Odersky (2008)
(<http://www.scala-lang.org/docu/files/ScalaByExample.pdf>)
- Programming in Scala (\$22.50), by Martin Odersky, Lex Spoon, and Bill Venners
(<http://www.artima.com/shop/forsale>)
- Scala Language Specification, by Martin Odersky (2008)
(<http://www.scala-lang.org/docu/files/ScalaReference.pdf>)
- Scala Library Documentation
(<http://www.scala-lang.org/docu/files/api/index.html>)
- Scalable Component Abstractions, by Martin Odersky, EPFL, and Matthias Zenger, Google (2005)
(<http://lamp.epfl.ch/~odersky/papers/ScalableComponent.pdf>)
- scala.xml, by Burak Emir
(<http://burak.emir.googlepages.com/scalaxbook.docbk.html>)
- Actors that Unify Threads and Events, by Philipp Haller and Martin Odersky (2007)
(<http://lamp.epfl.ch/~phaller/doc/haller07actorsunify.pdf>)

Week 1: March, 2nd

Introduction (Ben, Jorge, Julien)

Pure functional programming (Ben)

the Scala interpreter, values, expressions, higher-order functions and closures

Week 2: March, 9th

Pure functional programming (Ben)

local type inference, type parameters, operations on lists (map, filter, ...), for-comprehensions

State and object-oriented programming (Julien)

the Scala compiler, vars, classes, methods, nested definitions, objects, traits, mix-in composition

Week 3: March, 16th

State and object-oriented programming (Julien)

companion objects, packages, type members, self-types, bounded polymorphism, definition variance vs use-site variance

Unifying functional and object-oriented programming (Jorge)

algebraic data types, case classes, matching on lists, traversing trees

Week 4: March, 23th

Unifying functional and object-oriented programming (Jorge)

evaluating expressions, apply (functions are objects), extractors

Lazy programming (Jorge)

thunks, lazy values, streams

Week 5: March, 30th

Domain specific languages (Ben)

expression problem, DSLs, implicit definitions, “Pimp My Library” pattern, currying, operator precedence and associativity, matching on XML

Week 6: May, 7th

Software composition (Julien)

function composition (monoids, monads) and for-comprehensions, scalable component abstraction, “Cake” pattern

Week 6: May, 10th – Scala unconference

Week 7: May, 14th

Concurrent programming (Jorge)

actors, remote actors

Week 8: May, 21th – *lift* presentation (guest lecture by David Pollak)

Week 9: May, 28th – Miscellaneous (TBD)

Week 10: June, 4th (last day of classes) – Scala in the real world