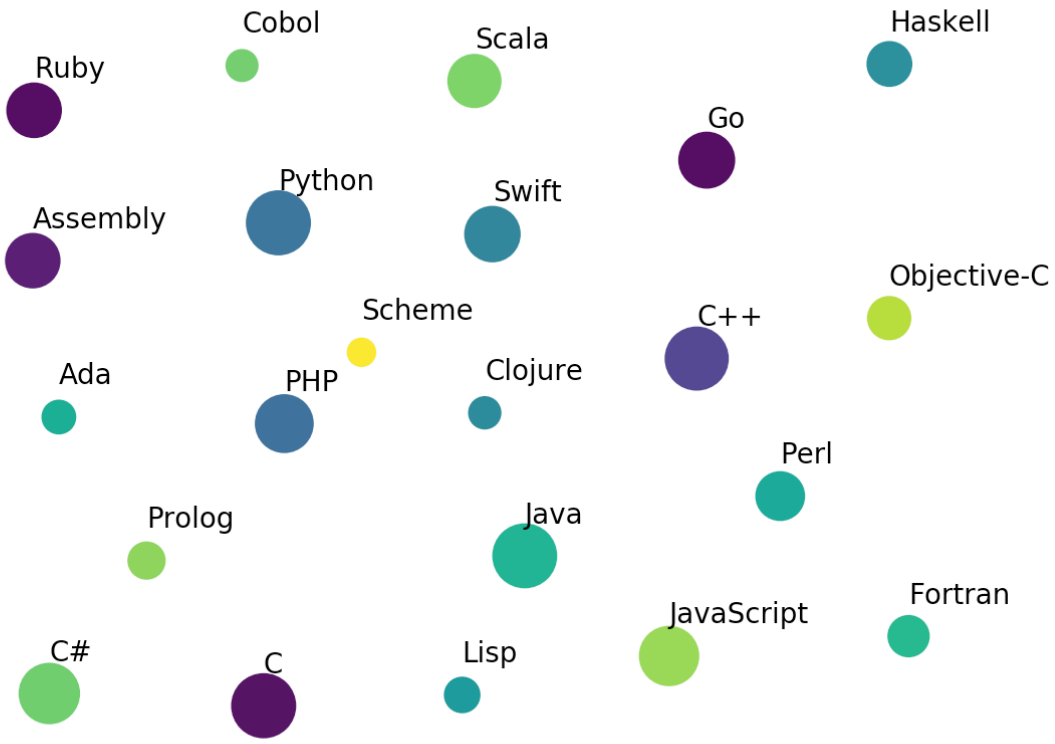
Course Syllabus

Programming Language Paradigms



Semester & Location:	Fall - DIS Copenhagen
Type & Credits:	Elective Course - 3 credits
Major Disciplines:	Computer Science, Mathematics
Faculty Members:	John Rager
Program Director:	Iben de Neergaard idn@dis.dk

Time & Place: Xxxxxx & Xxxxxx, XX.XX - XX.XX

Description of Course

The main purpose of a programming language is to provide a natural way to express algorithms and computational structures. Different meanings of “natural” have produced several distinct styles of languages – these are called paradigms. We will explore some important paradigms. Students will develop practical competency in languages representing distinct paradigms (e.g. Clojure, Prolog, Haskell). They will also be exposed to a selection of other languages. Topics will include object-oriented programming, functional programming, declarative programming, and programming for concurrency and distributed computing.

I imagine this course as consisting of three main parts:

1. Functional Programming (Haskell and/or Clojure)
    - ◊ type induction
    - ◊ lazy evaluation – infinite lists and recursion without base cases
    - ◊ currying - calling a function without all its arguments
    - ◊ higher order functions and functions as first class values
    - ◊ folds

2. Logic Programming (Prolog/ ECL^{i}PS^{e} Constraint Programming Language)
    - ◊ logic programming – programming "without algorithms", programming by telling the computer what is "true"
    - ◊ constraint programming – think Sudoku, but without searching explicitly

3. Object-oriented and Multi-Paradigm Languages (Java 8, Ruby, Scala, C++)
    - ◊ what makes a language OO – encapsulation, inheritance, polymorphism (delegation)
    - ◊ semantic ambiguity in inheritance
    - ◊ functional support in object oriented languages

In addition to these three main parts, we will also discuss general programming language issues as they arise throughout the course. These include:

- ◊ Binding times and flexibility - Binding is the word used to describe the process of associating two things (e.g. a variable and its type, a variable and its value). The timing of binding is perhaps the most important decision in programming language design.

- ◊ Scoping - Given a declaration, where within the program is it valid?

- ◊ First class values - Something that exists in a programming language is said to be a first-class value if you can use it in pretty much any possible context. Deciding what

to include in the first-class values of a language is an important decision.

◊ Abstraction - Abstraction - the separation of specification from implementation - can occur in different realms, including at least data abstraction and procedural abstraction. Languages differ in how they support this.

◊ Typing: Strong vs. Weak, Static vs. Dynamic Typing, Explicit vs. Implicit

◊ Storage Allocation - where and how does memory need to be allocated? Heaps, stacks and static allocation.

◊ Dynamic memory - How is memory allocated and deallocated dynamically? Is there explicit allocation or garbage collection?

Learning Objectives

By the end of this course students will

- Have developed an in-depth understanding of functional, logic, and object-oriented programming paradigms
- Understand the concepts and terms used to describe languages that support the imperative, functional, object-oriented, and logic programming paradigms
- Have written programs in functional and logical programming languages using the features central to those paradigms, thereby obtaining a working knowledge of programming in those paradigms.
-

Prerequisites

One year of computer science at university level. It is assumed that you are comfortable with programming in some language.

Faculty

John Rager is a full professor at Amherst College in Amherst, Massachusetts. He has always been interested in languages, both human and computer. His dissertation was in the field of symbolic natural language processing and subsequent to that his research has shifted to (among other things) natural language processing using machine learning. He has also worked on applying Artificial Intelligence to teaching English to Speakers of Other Languages. This work was motivated by the difficulties faced by English teachers in Moldova, where he was a Fulbright Scholar during the 2003-04 academic year. He has recently become interested in digital humanities and spent part of a recent sabbatical leave working at the Folger Shakespeare Library.

His teaching has often touched on language. For example, he has taught a seminar for first-year students called “Natural and Unnatural Languages.” The material in that course included “traditional” natural language processing as done in artificial intelligence, but also a discussion of rhetorical devices in Shakespeare, a reading of parts of Finnegan’s Wake and a discussion of language evolution. He has also recently taught a course on Digital Textual Analysis. That course discussed the computer science (e.g. topic modeling, Naive Bayes classification) used in papers in digital humanities. The course included both Computer Science and Humanities students, who worked together in groups on projects.

Readings

Scott, Michael. Programming Language Pragmatics. 2015

Thompson, Simon. Haskell: The Craft of Functional Programming. 2011

Bratko, Ivan. Prolog Programming for Artificial Intelligence.

Fogus, Michael. The Joy of Clojure. 2014

Horstmann, Cay. Java SE8 for the Really Impatient: A Short Course on the Basics

Most of the material in this course will be supplied via in-class examples and handouts. I will also supply internet resources for the languages we will be studying.

Field Studies

The Field Studies for this course will include visits to companies that use functional and/or logical programming languages. We will be able to see that these languages are used in the "real world" and hear from people who use them.

•

Guest Lecturers

Practical details about preparation should go in the course calendar.

Approach to Teaching

I have always believed in teaching students, not material, so expect the course to change in response to the needs and interests of the students in it.

Most days I will introduce some material and then we will then an exercise to support understanding the material. There will be lots of examples, and lots of discussion.

Expectations of the Students

1. Come to class. You won't learn much if you do not.

2. Ask and answer questions. You can reply to a question either with an answer or with a clarifying question. I ask lots of questions – it helps us all stay engaged.

Evaluation

There are several kinds of assignments in this course:

1. Many classes will include exercises. Some will be individual, some group. Your solutions to them will be gathered into "portfolios" which will be handed in several times during the semester. Some of these exercises will need to be finished after class.

2. There will be a group project – a modest sized program written in either functional or logical style. What you do is largely up to you and your group.

3. Short in-class quizzes. These are designed to measure how the understanding is going. They do not count as part of your grade.

4. One or two people will be assigned as note-takes for each class session. This way the class together will produce a set of notes of the course.

5. Two graded quizzes, one at the end of each of the first two major paradigms (functional and logical).

Tentative Outline

CLASS 1	Why are there over 4000 programming languages?
CLASS 2	What is Functional Programming?
	(Possible Exercise (PE): functional style programming)
CLASS 3	Functional Programming: Lists and Recursion
	(PE: lists as fundamental data structures)
CLASS 4	Functional Programming: Higher Order Functions
	(PE: higher order functions to avoid recursions)
CLASS 5	Functional Programming: Infinite Lists, Recursion without Bases Cases, Currying
	(PE: infinite lists)
short tour	
CLASS 6	Functional Programming: Folds
	(PE: folds)
CLASS 7	Concurrency and Functional Programming

CLASS 8	Concurrency/Functional Programming WrapUp
CLASS 9	What is Logic Programming? The simplest Prolog program
	(PE: Intro to Prolog)
CLASS 10	Prolog lists and recursion
	(PE: Prolog lists)
CLASS 11	Prolog concluded
Long Tour	
CLASS 12	Constraint Programming and Eclipse
	(PE: Intro to Eclipse and the Sentry problem)
CLASS 13	Constraint Programming
	(PE: Game)
CLASS 14	Constraint Programming
	(PE: Puzzles)
CLASS 15	Procedural Programming: C, Cobol and Fortran
	History of Programming Languages
Long Tour	
CLASS 16	Object-oriented Programming
CLASS 17	Object-oriented Programming : C++
CLASS 18	Object-oriented Programming: Polymorphism Challenge
	(PE: The Polymorphism Challenge)
CLASS 19	Multi-Paradigm Languages: Ruby
	(PE: Intro to Ruby)
CLASS 20	Multi-Paradigm Languages: Java 8
	(PE: Java 8)
Break	
CLASS 21	Project Work
CLASS 22	Project Work
CLASS 23	Project Presentations
End	

Exhibition

Grading

Assignment	Percent
Participation – behavior that promotes learning by you and others	20%
Exercise Portfolios	30%
Note Taking (not graded, you either did them and get credit, or didn't and don't)	10%
Graded Quizzes	15%
Project	25%

Late Assignments

You need to do the assignments in order to learn the material, so I will usually be willing to consider extensions. Please talk to me well before the deadline if you think you are going to have trouble making the deadline.

Use of laptops or phones in class

This is a programming-intensive class. You will need to use your laptop to do the programming. Please restrict your laptop use to working on coursework.

Academic Regulations

Please make sure to read the Academic Regulations on the DIS website. There you will find regulations on:

- Course Enrollment and Grading
- Attendance
- Coursework, Exams, and Final Grade Reports