CSE 505, Autumn 1996
D. Notkin
Introductory Handout
9/30/96

CSE505 will cover material on basic programming language paradigms and central issues of programming language theory and semantics. About 70% of the course will focus on three non-imperative programming paradigms: functional programming, logic/constraint programming, and object-oriented programming. About 10% will cover approaches to defining language semantics, and another 10% will cover issues about types in programming languages. The final 10% will mostly likely consider in some depth a small collection of interesting programming languages, put in the perspective of the earlier material in the quarter.

Personnel

Instructor	Teaching Assistant
David Notkin	*Kurt Partridge*
Sieg 414, 685-3798	
notkin@cs.washington.edu	kepart@cs.washington.edu
www.cs.washington.edu/homes/notkin	www.cs.washington.edu/homes/kepart

For office hours, etc., see the course web page:

`http://www.cs.washington.edu/education/courses/505/CurrentQuarter/`

Instructional objectives

The following are instructional objectives of the course; they are not by no means complete, but they characterize the kind of understanding that I hope you will develop over the quarter.

At the end of the course students should:

1. Be able to distinguish strong typing from static typing.
2. Be able to distinguish imperative, functional, object-oriented, logic, and constraint programming paradigms.
3. Be able to write at least modest (100+ lines of code) programs in languages representative of each paradigm.
4. Be able to write significant (1000+ lines of code) programs in a language representative of a paradigm that the student was not experienced in before the course.
5. Be able to demonstrate the fundamental relationships between logic and programming languages, including identifying language design decisions in which this relationship plays a role.
6. Be able to define formal semantic definitions of simple languages and constructs.
7. Be able to distinguish and motivate different object-oriented typing policies, such as contravariance and covariance.
8. Be able to assess existing and proposed programming languages in terms of a modern understanding of effective language design.
9. Be able to assess the consequences of selecting a particular language for the development and maintenance of a given software system.
10. Be able to relate programming language issues to implementation phase such as lexical analysis, syntactic analysis, compile-time, run-time, link-time, etc.
11. Be able to write a one page report that describes and assesses 90% of programming languages papers and colloquia.
12. Be able to design (parts of) programming languages or similar systems and justify your design decisions.

13. Be able to explain and answer questions about the major features of modern programming languages and their semantics.
14. Be able to explain and answer questions about type systems and how they relate to language design and logic.
15. Be able to read and write formal semantic descriptions for small programming languages, and use semantic description techniques to evaluate language designs.
16. Be able to solve problems using the functional and object-oriented paradigms.
17. Be able to implement the unification algorithm.
18. Be able to precisely distinguish logic from constraint programming.

Assigned work and grading

The basic work in the course consists of:

- Reading of papers from the literature (there is no assigned book for the course).[1]

    - Not all the material in papers will be discussed in class. The cse505 mailing list[2] will be used for a set of ongoing discussions about the papers and other pertinent programming language issues. In some cases, the TA and I will start these discussions; you should feel free to do the same. There is a threaded archive of these discussions on the course's web page, allowing you to follow specific threads, etc. Although participation in these discussions is not required, it is strongly encouraged (and may contribute to intangibles in assigning final grades).

- Assignments including both written and programming components; these will comprise 50% of the total grade. All assignments may be done in pairs, with both members receiving identical grades. You may choose to work in pairs, or not, independently on each assignment; you may also change partners for each assignment. I strongly encourage people to study together in this course, whether or not they are working in pairs; if you're about whether something crosses the line to cheating, feel free to run it by me first (in general, the simple rule is to make sure not to represent someone else's work as your own).

- A final examination; this will comprise 20% of the total grade.

- A final project, which can be done in pairs; this will comprise the remaining 30% of the total grade. These projects can range from significant programming projects to written term papers. They are due on Thursday December 12 at 10AM (the final is the next morning). This date cannot be extended due to the timing of the final, etc. Each person or pair must negotiate specifics of their project with me in advance; no later than the end of the sixth week of the quarter (that is, by November 8th).

[1] To defray copying costs, please give me a check for $25 made out to the University of Washington by the end of the first week of classes.
[2] To sign up for the mailing list, send a message to `majordomo@cs` with a body (not subject field) that says `subscribe`.

Tentative lecture schedule

September 30 *Introduction*	October 2 *Basic Principles*	October 4 (Hudak guest lecture) *Functional Programming*
October 7 *Functional Programming*	October 9 *Functional Programming*	October 11 *Functional Programming*
October 14 *Functional Programming*	October 16 (FSE, Notkin gone) *Functional Programming*	October 18 (Chambers guest lecture) *Functional Programming*
October 21 *Logic/Constraint Programming*	October 23 *Logic/Constraint Programming*	October 25 *Logic/Constraint Programming*
October 28 *Logic/Constraint Programming*	October 30 *Logic/Constraint Programming*	November 1 (P. Lee guest lecturer) *TBA*
November 4 *Object-oriented Programming*	November 6 *Object-oriented Programming*	November 8 *Object-oriented Programming*
November 11 (Veteran's Day) *No class*	November 13 *Object-oriented Programming*	November 15 *Object-oriented Programming*
November 18 *Semantics*	November 20 (Leavens guest lecture) *Semantics*	November 22 (Leavens guest lecture) *Semantics*
November 25 *Semantics*	November 27 *TBA*	November 29 (Thanksgiving) *No class*
December 2 *Semantics*	December 4 *Semantics*	December 6 *Some language*
December 9 *Some other language*	December 11 *Yet another language*	December 13 *Final* 8:30AM-10:20AM

A few specific notes about the schedule. First, we have several guest lecturers, including Paul Hudak (Yale), who will give the overview of functional languages, and Peter Lee (CMU), who will give a presentation on some work on using programming language techniques to define safe kernel code. Gary Leavens will lecture for me twice in November, when I am at a program committee meeting; Craig Chambers will lecture once in October, when I am at a conference (and I still have one day on that trip left to cover). The Wednesday before Thanksgiving will be an informal class, so it's not critical that you attend.