Syllabus

Handout 1
CSCI 334: Spring, 2012

Principles of Programming Languages

Instructor	Prof. Stephen Freund
Office	TPL 302, 597-4260
Email	freund@cs.williams.edu
Office Hours	MW 2:30 – 4:00
TAs	Antal Spector-Zabusky
Lectures	TR 9:55–11:10 in TCL 206
Web Page	http://www.cs.williams.edu/~freund/cs334/index.html

Texts

We will be using the following text book:

- **(Required)** *Concepts in Programming Languages*, John C. Mitchell.

Additional readings will be posted on the web site.

Course Objectives

A programming language is a programmer's principle interface to the computer. As such, the choice of an appropriate language can make a large difference in a programmer's productivity. A major goal of this course is to present a comprehensive introduction to the principle features and overall design of both traditional and modern programming languages. You will examine language features both in isolation and in the context of more complete language descriptions. The material will enable you to:

1. Quickly learn programming languages, and how to apply them to effectively solve programming problems.
2. Rigorously specify, analyze, and reason about the behavior of a software system using a formally defined model of the system's behavior.
3. Realize a precisely specified model by correctly implementing it as a program, set of program components, or a programming language.

We will examine features of a large variety of languages, though we will not study many of languages themselves extensively. Like other CS courses, we will discuss alternate ways of solving problems, looking at the pros and cons. Because programming languages are so tied up (and motivated by) programming problems, we will not only investigate language features, but also the software engineering problems that spawned them.

At the end of this course you will have a more thorough understanding of why certain programming language features provide better support for the production of reliable programs, while others are fraught with ambiguity or other problems. Since programming languages mediate between the programmer and the raw machine, we will also gain a deeper understanding of how programming languages are compiled, what actually happens when a program is executed on a computer, and how the programming language design affects these issues. As an example, by the end of the course, you should be able to understand why Java has replaced C++ language of choice for many projects and to recognize where language design is likely to head in the future.

An important feature of this course is the discussion of programming language paradigms (in particular, languages supporting new ways of thinking about implementing algorithms). We will investigate both the new features themselves and the software engineering problems which spawned these developments.

This course will involve extensive reading on your part, both in the text and in outside sources. The segments of the course that introduce new programming language paradigms will also feature some programming in languages representative of the functional and object-oriented paradigms (Lisp, ML, Scala, and possibly others).

Lectures

Lectures are mandatory. I expect you to attend and participate.

Tentative Schedule

This will undoubtedly change as we begin to explore these topics. Additional reading will be assigned from other sources. The web page will always contain an up-to-date list of topics and readings for each lecture.

Week	Date	Topic
Week 0	2/1	Intro, Halting Problem
Week 1	2/7–2/9	Lisp and Functional Programming
Week 2	2/14–2/17	PL Foundations
Week 3	2/21–2/23	More Foundations and ML
Week 4	2/28–3/1	ML and types
Week 5	3/6–3/8	Stacks and Scope
Week 6	3/13–3/15	Control Flow, Exceptions
Week 7	4/3–4/5	Modularity, Abstraction, OOP
Week 8	4/10–4/12	OOP
Week 9	4/17–4/19	C++
Week 10	4/24–4/26	Java and Scala
Week 11	5/1–5/3	Security, Concurrency
Week 12	5/8–5/10	Concurrency

Homework

Problems involving analysis of programming language features will be assigned and due most weeks during the term. Homework will generally be due on Tuesdays. Each student may use a maximum of *three late days* during the course of the semester. Once those late days are used up, late homework will not be accepted. There will be midterm and final exams covering both lectures and readings. The midterm will occur the week before Spring Break.

There will be small programming assignments as part of the homeworks. These will primarily reinforce conceptual ideas from lecture and expose you to different programming paradigms. We will use the Computer Science Department's UNIX computers for the programming problems. If you are not familiar with the UNIX computing environment, talk to me or the TA as soon as possible so we can bring you up to speed on what you need to know.

Grades will be determined roughly as follows:

Midterm: 15–20%
Final Exam: 25–30%
Homework and programs: 40%
Other (class participation, attendance, quizzes, etc.): 10%

Honor Code

Homework is to be the sole work of each student unless the assignment explicitly states otherwise. Students may collaborate or receive help from each other on an occasional basis as long as all parties contributing or assisting are given explicit credit for their contributions to the homework. In particular, I hope you will help each other in learning the mechanics of how to compile programs in new languages. I will inform students if I believe they are collaborating too much. If in doubt as to what is appropriate, ask me. Uncredited collaborations will be considered a violation of the honor code and will be handled appropriately. The complete computer science honor code may be read at `http://www.cs.williams.edu/~freund/honor.html`.