

---

## Principles of Programming Languages

---

|              |   |
|--------------|---|
| Instructor   | Prof. Stephen Freund  |
| Office       | TPL 302, 597-4260   |
| Email        | <a href="mailto:freund@cs.williams.edu">freund@cs.williams.edu</a>  |
| Office Hours | MW 2:30 – 4:00  |
| TAs          | Catalin Iordan and Ben Wood   |
| Lectures     | TR 9:55–11:10 in TCL 206  |
| Web Page     | <a href="http://www.cs.williams.edu/~freund/cs334/index.html">http://www.cs.williams.edu/~freund/cs334/index.html</a> |

---

## Texts

---

We will be using the following two text books:

- **(Required)** *Concepts in Programming Languages*, John C. Mitchell.
- **(Recommended)** *Elements of ML Programming: ML '97 Edition (2nd edition)*, Jeffrey D. Ullman.

Both books are available at the bookstore. Additional reading will be posted on the web site.

---

## Course Objectives

---

A programming language is a programmer's principle interface with 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 principal 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.

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++ as the hot language and 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 which support 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, Smallalk, and possibly others).

---

## Lectures

---

Lectures are mandatory, and I expect you to attend and participate in the class.

## Tentative Schedule

This will undoubtedly change as we begin to explore these topics. Additional reading will be assigned from other sources. Check the web page for updates to the schedule.

| Date          | Tues   | Thurs   |
|---------------|--|---|
| Jan 31        |  | Intro<br>Halting Problem                                  |
| Feb 5–Feb 7   | Lisp<br><i>Mitchell 3</i>                            | Lisp<br><i>Mitchell 3</i>                                 |
| Feb 12–Feb 14 | Foundations<br><i>Mitchell 4.1-4.2</i>               | Foundations   |
| Feb 19–Feb 21 | Foundations<br><i>Mitchell 4.4</i>                   | Algol and ML<br><i>Mitchell 5, 6; Ullman as needed</i>    |
| Feb 26–Feb 28 | ML: types and polymorphism                           | ML: type inference<br><i>Mitchell 7</i>                   |
| Mar 4–Mar 6   | Stacks and Scope<br><i>Mitchell 8</i>                | Stacks and Scope  |
| Mar 11–Mar 13 | Exceptions   | To Be Determined  |
| Mar 18–Mar 20 |  |   |
| Mar 25–Mar 26 |  |   |
| Apr 1–Apr 3   | Modularity and Data Abstraction<br><i>Mitchell 9</i> | <b>Midterm</b> (tentative)                                |
| Apr 8–Apr 10  | Object-oriented Languages<br><i>Mitchell 10</i>      | Simula: the beginnings<br><i>Mitchell 11</i>              |
| Apr 15–Apr 17 | Smalltalk: subtyping/inheritance                     | C++: mult. inheritance / efficiency<br><i>Mitchell 12</i> |
| Apr 22–Apr 24 | C++  | Java: interfaces / virtual machines<br><i>Mitchell 13</i> |
| Apr 29–May 1  | Java: safety, exceptions, etc.                       | GJ, C#, Security  |
| May 6–May 8   | Languages and Security                               | Wrap up   |

The relevant chapters from Ullman's book are 2–3, 5.1, 5.3, 6.1–6.2, plus whatever else is needed for the programming assignments.

## 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.

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>.