Principles of Programming Languages

Instructor: Prof. Stephen Freund
Office: TPL 302, 597-4260
Email: freund@cs.williams.edu
Office Hours: See web page

TA Hours: TBD (See web page)
Lectures: TR 9:55–11:10 in SSL 030B

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.

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 0</td>
<td>2/1</td>
<td>Intro, Halting Problem</td>
</tr>
<tr>
<td>Week 1</td>
<td>2/7–2/9</td>
<td>Lisp and Functional Programming</td>
</tr>
<tr>
<td>Week 2</td>
<td>2/14–2/16</td>
<td>PL Foundations</td>
</tr>
<tr>
<td>Week 3</td>
<td>2/21–2/23</td>
<td>More Foundations and ML</td>
</tr>
<tr>
<td>Week 4</td>
<td>2/28–3/2</td>
<td>ML and types</td>
</tr>
<tr>
<td>Week 5</td>
<td>3/7–3/9</td>
<td>Stacks and Scope</td>
</tr>
<tr>
<td>Week 6</td>
<td>3/14–3/16</td>
<td>Control Flow, Exceptions</td>
</tr>
<tr>
<td>Week 7</td>
<td>4/4–4/6</td>
<td>Modularity, Abstraction, OOP</td>
</tr>
<tr>
<td>Week 8</td>
<td>4/11–4/13</td>
<td>OOP</td>
</tr>
<tr>
<td>Week 9</td>
<td>4/17–4/20</td>
<td>C++</td>
</tr>
<tr>
<td>Week 10</td>
<td>4/25–4/27</td>
<td>Java and Scala</td>
</tr>
<tr>
<td>Week 11</td>
<td>5/2–5/4</td>
<td>Concurrency</td>
</tr>
<tr>
<td>Week 12</td>
<td>5/9–5/11</td>
<td>Secrurity</td>
</tr>
</tbody>
</table>

### Homework

Problems involving analysis of programming language features will be assigned weekly and typically on Tuesday. You homework should:

- be turned at the beginning of class on the due date,
- be clearly written or typed,
- include source code print outs for questions involving programming,
- list any students with whom you discussed the problems (see Honor Code below), and
- be stapled, with your answers clearly marked and in the same order as the questions.

You will be asked to fix and resubmit homework not following these conventions. I will not accept homework by email.

Each student may use a maximum of three late days during the course of the semester. A single late day enables you to hand in the homework up to 24 hours after the original due date. Once those late days are used up, late homework will be penalized 20% per day. Late homeworks should be turned into Lauren Vining in TCL 303.

### Programming

The homeworks will often include small programming assignments to reinforce conceptual ideas from lecture and expose you to different programming paradigms. For some of these you will be required to work with a partner. I will provide more details on the first homework including a programming component.
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. You may also see Mary Bailey to get your Unix password if you have forgotten it.

Exams

There will be midterm and final exams covering both lectures and readings. The midterm will occur after Spring Break. The final will be a 24-hour Take Home Exam.

Grading

Grades will be determined roughly as follows:

- Midterm: 25%
- Final Exam: 35%
- Homework and programs: 35%
- Other (class participation, attendance, quizzes, etc.): 5%

Honor Code

Homework is to be the sole work of each student unless the assignment explicitly states otherwise. Students may discuss problems with each other on an occasional basis as long as everyone contributing to the discussion is given explicit credit for their contributions. In particular, I hope you will help each other in learning the mechanics of how to write and compile programs in new languages. All solutions should be written independently. 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](http://www.cs.williams.edu/~freund/honor.html).