Welcome

Amer Diwan

---

Goals of the course

- This course will prepare you to
  - learn new languages easily
  - better understand the strengths and weaknesses of current languages
  - compare suitability of different languages for a task
Prerequisites

- **ECEN 2120: Computer as components**
  - Familiarity with some instruction set architecture
- **CSCI 2270: Data structures**
  - Good understanding of data structures: trees etc.
  - Recursion
  - Programming experience in an O-O language
  - Familiarity with UNIX
- **Please take prerequisite before taking this course**

Course requirements

- Three projects (45%)
- Quizzes (25%)
- Final exam (20%)
- Questions on readings (10%)
- See class web page for more details:
  www.cs.colorado.edu/~diwan/3155-01
Projects

- Preparation
- Use concepts
- Implement concepts

Prepare for implementation: may or may not involve writing code
Use a modern programming language to implement simple versions of concepts
Discuss, design, and extend the implementation for a more elaborate version of concepts

Quizzes

- **Good news:** No midterms
- **Bad news:** Every alternate recitation you will have a 20 minute quiz
  - Will cover all material covered before the quiz
Daily questions

• Before each class submit a question via PEP on the day’s readings
  – The question should be specific (e.g., I don’t understand this reading, could you explain it? is not acceptable)
• We will answer these via email
• First question due next class!
• Readings will cover material not covered in lectures

Introduction

Amer Diwan
Readings

• Chapter 1

Why programming languages

• PL+OS give a high-level interface to programmers
  – High-level abstractions
    • Closer to the problem domain than the hardware
    • Objects instead of raw bytes
  – Hardware independence
    • Portability
  – Performance
    • Compiler can tailor code for underlying hardware
Aspects of programming language features

low level

Implementation in compiler and run-time system
Concept implementation in language
Concepts

“Naive” view  “Ideal” view  “Realistic” view

Examples

<table>
<thead>
<tr>
<th>References and V-Tables</th>
<th>“Boxing”, and type inference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclassing, method invocations in C++</td>
<td>Type variables in SML</td>
</tr>
</tbody>
</table>

Polymorphism
Why study concepts?

- Concepts are not burdened by language design or implementation considerations
- So it is easy to learn new languages
- So we are in a better position to understand the strengths and limitations of a language

Why study concept implementation in languages

- So we know how to make the best use of languages.
- So we understand the interaction between different language features.
Why study impact on compilers and run-time systems?

- It is part of the motivation behind many language designs
- So we understand the implication of using language features
- So we know what pitfalls to avoid when designing a language

Relationship of CSCI 3155 to other courses

<table>
<thead>
<tr>
<th>Concepts</th>
<th>CSCI 4555</th>
<th>CSCI 5535</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept implementation in language</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation in compiler and run-time system</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CSCI 3155
What makes one language “better” than another?

• Some criteria:
  – Expressiveness
  – Simplicity
  – Ease of implementation
  – Taste
  – (and many other factors that will come up over the course of the term...)

Expressiveness

• Which language/feature allows you to write clear, concise, and maintainable code for your application?

• C++ versus C
  – Can use classes to write ADTs more easily in C++ than in C
  – Others?
Simplicity

• Which language/feature is simpler to understand and use?
  • C++ versus Smalltalk
    – Everything is an object (Smalltalk) versus some things are arrays, some are structs, some are unions, some are objects...
  • Uniformity in a language often leads to simplicity

Ease of implementation

• What language/feature is simpler to implement?
  • C++ versus Java
    – Multiple inheritance of C++ needs significant more implementation effort than single inheritance in Java
Taste

- What if two languages/features have different strengths (e.g., simplicity versus ease of implementation)? **Boils down to personal judgment and taste**
- C++ versus Java
  - Java’s single inheritance is simpler but less expressive than C++’s multiple inheritance

Next topic: Syntax

- Readings: Scott: 2.1, 2.2 (excl 2.2.4, “Locally Least-Cost error recovery”, 2.2.6)
- Note: when I list a section (e.g., 2.1) it includes all subsection of 2.1 (e.g., 2.1.1)