Comp 411
Principles of Programming Languages
Lecture 1
Course Overview and Culture

Corky Cartwright

January 13, 2014

Course Facts

- See web page `www.cs.rice.edu/~javaplt/411` and Piazza page `piazza.com/rice/spring2014/comp411`
- Participate in the discussions at Piazza site.
- Coding style matters; testing suites really matter.
- Grade is 50% assignments and 50% tests; there are two segmented take home exams. See the tentative schedule on the course web page.

What is Comp 411?

Anatomy (Syntax) and Physiology (Semantics) of Programming Languages

- What is the anatomy of a programming language
  - Parsing and abstract syntax
  - Lexical nesting and the scope of variables
- What are the conceptual building blocks of programming languages? (common anatomical structures and their functions)
- Use high-level interpreters to define meaning of languages (expression evaluators)
-

What is Comp 411, cont.

- Using anatomy to prevent bugs
  - Type systems
  - Type checking
  - Type inference (reconstruction)
- Mechanisms for language extension
  - Syntax extension (macros)
  - Reflection
  - Custom class loaders
- Sketch how the interpretation process can actually be efficiently implemented by machine instructions
  - CPS transformation
  - Garbage Collection

Subtext of Comp 411

- Teach good software engineering practice in Java.
- You have to write lots of lines of conceptually challenging lines of code in this course. With good software engineering practices, the workload is reasonable.
- With poor software engineering practices, the workload is unreasonable.
- The assignments in this course leverage abstractions that are not explicit in Java but are easily encoded using the proper design patterns (*e.g.*, composite, interpreter, strategy, visitor).
- In putative successors to Java, notably Scala, these abstractions are built-in to the language. Unfortunately, the semantics of Scala are hideously complex. Martin Odersky has assured me that a new edition of Scala with a semantically tractable core subset (called "Dot") is in the pipeline. I am hopeful, but in the meantime, we will use Java, which is a nice language if it used properly.

Good Software Engineering Practice

- Test-driven design
  - Unit tests for each non-trivial method written before any method code is written
  - Unit tests are a permanent part of the code base
- Pair programming
- Continual integration
- Continual refactoring to avoid code duplication
- Conscientious documentation (contracts)
- Avoiding mutation unless there is a compelling reason

Course Culture

- Approximately 8 programming assignments
  - 7 required
  - 1-2 extra credit
- Assignments must be done in either Generic Java (Java 6/7/8 including parameterized types). We encourage you to use DrJava. Both JUnit and javadoc are built-in to DrJava and they are fully compatible with command line compilation, execution, and testing (using ant scripts).
- Late assignments not accepted, but …
  - Every student has 7 slip days to use as he/she sees fit.
  - Advice: save as many slip days as possible until late in the term. The last two assignments are the most time-consuming.

Course Culture, cont.

- Assignments are cumulative.
- Class solutions are provided for the first three assignments within three days after they are due.
- After the third assignment, you are on your own except for skeleton test suites which we will provide. Extensive unit testing is important. In most of the projects, you can reuse previous unit tests on subsequent assignments with no change.

Why Study Programming Languages?

- Programmers must master the programming languages of importance within the domains in which they are working.
- New languages are continually being developed. Who knows what languages may be involved in computing 25 years from now?
- Many software applications involve defining and implementing a programming language.
- A deep knowledge of programming languages expands the range of possible solutions available to a software developer. A program design may involve extending the designated implementation language either explicitly (macros, new/revised translators & custom class loaders) or implicitly (new libraries, hand-translation)
- Some implementation languages are extensible through macros, reflection, or customizable class loaders.

Course Culture, cont.

- My teaching style
  - Encourage you to develop a passion for the subject and personally digest and master the material.
  - Make the course accessible to students who don't aspire to become language researchers (avoid repeating my experience in complex analysis with Andy Gleason)
  - Weaknesses:
    - Tendency to digress
    - Explain concepts at too abstract a level without sufficient examples
    - Redress: remind when I have strayed from the course outline; ask questions about examples and tell me if my explanations are too abstract

Putative Assignment

The web page:

wiki.rice.edu/confluence/display/cswiki/Comp+411+Putative+Assignment

contains a putative assignment showing the OO program design background that is expected for this course. In the past, some students have quickly learned this approach to program design in the first few weeks of the course.

This assignment is not intended to add your workload (a pocket assignment). Do not do the assignment unless you need to do it as a learning experience.