600.325/425
Declarative Methods

Prof. Jason Eisner
MWF 3-4pm (sometimes 3-4:15)
What is this course about?

- What do you learn in a programming course?
  - How to use a language (e.g., Java) to solve problems
  - How the computer actually executes that language
    - (Why do you need to know this?)

- Ok, this is a programming course
  - We’ll survey several languages
  - But they aren’t normal languages!
Low-level vs. high-level languages

One common language
Can do anything
(Turing-complete)

Lots of different tasks

Assembler

C

FORTRAN

Programmer has freedom to write anything.
Hopefully has a lot of free time, too.

Compiler has limited freedom because you told it exactly what to do

Low-level: Long, detailed programs written by anal-retentive programming gurus
Low-level vs. high-level languages

More abstract classes
Interface is closer to how you think about the task

As before, but better organized
Reusable building blocks
(objects or modules)

Java, C++, OCaml ...

Lots of different tasks

Useful basic classes
Written by other people

Building up high-level objects from low-level ones
But language and compiler are still low-level

Compiler has limited freedom because someone told it exactly what to do
Low-level vs. high-level languages

A higher-level language that can also do anything (Turing-complete)

English and its obscure variants (Hindi, Chinese, German, …)

Boy, wouldn’t you like to write this optimizing compiler? (take 600.465 NLP) (and 600.463 Algorithms)

Really high level! Programming for the masses!
Low-level vs. high-level languages

A higher-level language that can also do anything (Turing-complete)

Could we make a formal English-like language?

“On each line, replace every third word by x’s, then sort the words by length ...” It would have to know an awful lot of concepts (line, word, third, sort, length). Maybe just make a big library of specialized objects for those concepts? Some of those objects would need to have pretty powerful methods: “Schedule the classes to minimize time conflicts.”
A lot of the same kinds of stuff over and over, actually
So maybe it is good to build some powerful, general, reusable objects to handle cases that are either common or hard
Then you don’t waste your time doing the same kind of thing again & again
And you don’t waste your time figuring out how to do something new & hard
Low-level vs. high-level languages

Several specialized high-level languages “Tools for the job”

A lot of the same kinds of stuff over and over, actually
So maybe it is good to build some powerful, general, reusable objects to handle cases that are either common or hard
Then you don’t waste your time doing the same kind of thing again & again
And you don’t waste your time figuring out how to do something new & hard

SQL or Datalog
Regexp patterns
Constraint programs

Solvers: general algorithms (sometimes can choose among several wildly different solvers)
Low-level vs. high-level languages

Several specialized high-level languages "Tools for the job"

- SQL or Datalog
- Regexp patterns
- Constraint programs

To tell the database about your problem, use SQL (standard query language). More expressive than calling a database method!

Query optimization – might even compile your query into machine code before running it

Solvers: general algorithms
I can teach you how to bottle fame, brew glory, even stopper death.

If you aren't as big a bunch of dunderheads as I usually teach.
Structure of this course

- **Intro material**
  - What are languages? What do they look like?
  - What’s a declarative language? What’s a solver?
  - Encoding a problem in a language
  - Reducing one language to another; NP-hardness

- **Several actual languages. For each:**
  - **Week 1:** How does this language let me encode interesting problems?
  - **Homework:** Encode a real problem and run a solver.
  - **Week 2:** What strategies does the solver use to solve arbitrary problems written in the language?

- **Project of your choice (for 425 students)**