Programming Languages for CS 1
Virtual Machines & Programming Languages

- Virtual machines for programming languages:
  - Pascal virtual machine
  - Java virtual machine
  - Smalltalk virtual machine
  - ML or Scheme virtual machine
- Creates abstract computer for programmers
Needed: A Teaching Language

Kölling (1999) JOOP

- Clean concepts
- Pure object-orientation
- Safety
- High level
- Simple object/execution model
- Readable syntax

- No redundancy
- Small
- Easy transition to other languages
- Support for correctness assurance
- Suitable environment
Java strengths for CS 1

- Purely object-oriented
- Objects uniformly represented as references
  - but primitives different (Java 5 tries to fix)
- Garbage collection
- Interfaces separate from classes
  - though under-utilized
- Support for assertions (preconditions, etc.)
Java 5 Improvements

- Simpler I/O
- Implicit conversion between primitive and wrapper types.
- for(-each) with explicit & implicit iterators
- Type parameters
  - unfortunately not supported in JVM
Future of Java

- With Java 5, probably seen most important improvements to language.
- Java’s use will peak in next few years.
- Replacement language in five to ten years.
Java Problems For CS 1

- Many can be overcome w/ libraries
  - Graphics, event-driven programming, GUI
- Still problems w/Java 5 additions
- Others more difficult. Examples:
  - Syntax
  - Static vs Instance Methods in applications
  - Overloading vs. overriding confusion
Syntax

- Syntax is user interface to programmer.
- Inherited from C. Pretty awful for novices!
- Designed for experts by smart people who hated to type.
Inconsistent, Confusing Syntax

- Inconsistent lexical conventions:
  - `=` vs `==`
  - `&` vs `&&`
  - `+` vs `++`

- Array notation not O-O

- Dangling else
Static vs Instance

- Static methods in class cannot access instance variables or methods.
- All applications start with
  - `public static void main(String[] args);`
- Disciplined o-o programmers only construct an object & (optionally) send it a message in main.
- Why not just invoke a constructor at command line (or equivalent)?
Overloading vs Overriding

- Method overloading resolved statically
  - at compile time
- Method overriding resolved dynamically
  - at run time
Can You Tell the Difference?

class C { ...
    boolean eq(C other) {...} (1)
}
class SC of C { ...
    boolean eq(C other) {...} (2) override
    boolean eq(SC other) {...} (3) overload
}

C c,c'; SC sc;

c = new C();   c' = new SC();   sc = new SC();
c.eq(c);       c.eq(c');      c.eq(sc);
c'.eq(c);      c'.eq(c');     c'.eq(sc);
sc.eq(c);      sc.eq(c');     sc.eq(sc);
Moral

Sometimes two perfectly reasonable features interact in very confusing ways.

As instructors, our job is to find a way of avoiding these problems.

Find better language, avoid combining features, or teach students dangers of interaction.