CSE 341: Programming Languages

Winter 2005

Lecture 24— Static Typing for OO Languages

Static Typing for OO

Remember, any sound static type system prevents certain errors.

In ML, we never treated numbers as strings or functions, etc.

For an OO language, what's the most conventional guarantee for a type system?

- Program execution will not send a not-understood message
- Except for `nil`?

Is that it?

- Pretty much; that's all Smalltalk programs do
- And it's not that easy to be sound and useful
- With multimethods or *static overloading* (coming later), we can also seek to prevent "no best match" errors

The plan

- A "from first principles" approach to object-types
  - For objects with just getters and setters
  - The need for subtyping
  - Digression to ML records
    - Immutability makes more things subtypes
  - Considering methods
    - Arguments are "contravariant"
- Next time: continue; connect this up with classes and interfaces

Warning: Lots of jargon, but ideas are very important