

Comp 311  
Principles of Programming Languages  
Lectures 14-15  
Review

Corky Cartwright  
September 27-29, 2009

# OO Code Samples

- Show selections from solution to Assignment 2
  - Class hierarchy for Binding union
  - Sample visitor method code
- Discuss some OO design tradeoffs
  - Use of instanceof
  - With composite, visitor implementation of methods is not always mandated. Good idea to “build in” some core operations of a composite using the interpreter pattern. Why? Leaner (in terms of lines of code). Easier to read.

# Good Commenting Conventions

- Javadoc description for every class, field, non-trivial method.
- Method descriptions are informal contracts. Contracts should be as precise as possible. In some cases (e.g., GUI libraries), complete precision may not be feasible.
- Sample solutions could be better commented.

# Answer to Question on Eliminating Lambda

Goal: devise a few combinators that enable us to express all  $\lambda$ -expressions without explicitly using  $\lambda$ .

Notation: let  $\lambda^* x. M$  denote  $\lambda x. M$  converted to a form that eliminates the starred  $\lambda$ . Then

- $\lambda^* x. x \rightarrow \mathbf{I}$  (where  $\mathbf{I} = \lambda x. x$ )
- $\lambda^* x. y \rightarrow \mathbf{K} y$
- $\lambda^* x. M N \rightarrow \mathbf{S} (\lambda^* x. M) (\lambda^* x. N)$  (where  $\mathbf{S} = \lambda x. \lambda y. \lambda z. (y x) (z x)$ )

Strategy: eliminate  $\lambda$ -abstractions from inside out, one-at-a-time. But any order works. Transformation can cause exponential blow-up.

Note:  $\mathbf{I}$  is technically unnecessary since  $\mathbf{SKK} = \mathbf{I}$