CS 334
Lecture 13
Modularity & Abstraction

Design Methodology
- often hierarchical
- Top-Down Design.
- Bottom-Up Design.

Modularity - changing one module does not require changing others.
Component
- Individual program unit
- function, class, package, library, ...

1. Interface: types & ops defined in component that are publicly visible.


Abstraction
- hide details.
- procedural abstraction.
- data abstraction
- representation info is hidden.

ADT - Abstract Data Type.
- hides internal rep.
- type system to enforce hiding.
- representation independence.
OOP

Execution Model
- Object = hidden data + public methods.
- send messages / call methods
  obj. to String()

OO Methodology

Water
- SimLoop
- Graph
- Mapper
  |    
  v
Create Fish
  ... Plot Data
  Draw Fish
  Top
  Down
Organize around objects.

Roles & Interactions.
1) Dynamic Lookup.

- ML: fun toString (v : int) = ..
  fun toString (v : 'a) = ..
  toString (value).

- Java:
  
  `v.toString();`

  1) meaning depends on receiver.

  2) different objects respond differently.
Encapsulation

- Hide State in objects.
- Designer: detailed view of object internals.
- Client: abstract view via public methods.

- Compare to ADTs:
  + separate internal impl./interface
  + data abstraction.

- ADTs are not extensible.
- ADTs do not support substitutivity.
3. Subtyping

- Interface is external of object
- Relationship between interfaces

\( A \leq B \) if \( A \) contains all methods contained in \( B \)

\textbf{Filled Rect} \( \leq \textbf{Rect} \).

- Substitutivity: If \( A \leq B \) then any expression of type \( A \) can be used in any context requiring an expression of type \( B \).
1. Inheritance
   - implementation reuse.

Subtyping - property used by clients to operate on related objects uniformly.

Inheritance - property used by implementer to reuse code.
Inheritance vs Subtyping in Java

class A extends B  =>  A <: B
class A implements I  =>  A <: I

MouseListener  
Applet
\ <: / <:
Stock Ticker