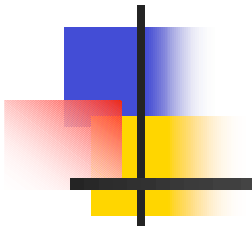


OO Design Retrospective

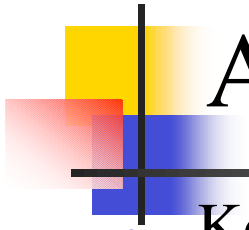
Corky Cartwright
Department of Computer Science
Rice University

Enduring Design Principles

- Form of data determines form of the code
- A repeated code pattern flags a missing abstraction; eliminate it if the language provides a simple abstraction mechanism for expressing the abstraction.
- Brief case study: Acker's laundry
  - Dominant form of data is sequences of garments
    sequence of shirts
    sequence of pants
    sequence of socks
    sequence of garment (shirt | pants | socks)
    sequence of sequence of garment

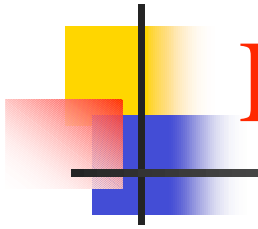
Acker's Laundry cont.

- Interesting technical issues with generics arise if we use narrower (better) types like `BiListI<Shirt>` instead of `BiListI<Garment>`. I ducked those issues by writing the program skeleton for you and consistently using the type `Garment` instead of its subtypes.
- What complication arises if we use types like `BiListI<Shirt>`?
- `BiListI<Shirt>` is not a subtype of `BiListI<Garment>`. The solution requires the use of wildcard types like `BiListI<? Extends Garment>`, which we have avoided for good reason. (Even the JLS makes a mess of them.)



Acker's Laundry cont.

- Key observation: most operations on sequences involve iteration over the sequence: the iterator design pattern. The code base for HW10 included appropriate definitions of sequence data structures and iterators over them. (Java 6.0 has some passable alternatives.)



For Next Class

- Exam over OO material will be distributed on Friday, April 10, and due by 11:59PM on Friday, April 17 in my office.
- Tic-tac-toe homework due Friday. We are demoting the Alpha-Beta pruning part to extra credit. Like the last assignment, you only have to write a modest part of the total application. Have fun.
- Please, please read my notes on OO Design.