OBJECT-ORIENTED ETUDES TOO

Curt Clifton
Rose-Hulman Institute of Technology

A WARM-UP: BOOLEANS SANS BOOLEANS

- Implement a set of classes to model booleans
- The classes must support:
  - *and*, *or*, and *not*
  - branching
- The implementation must not use any conditional expressions or statements!

Challenge: How could we make these short-circuiting?

NATURALLY

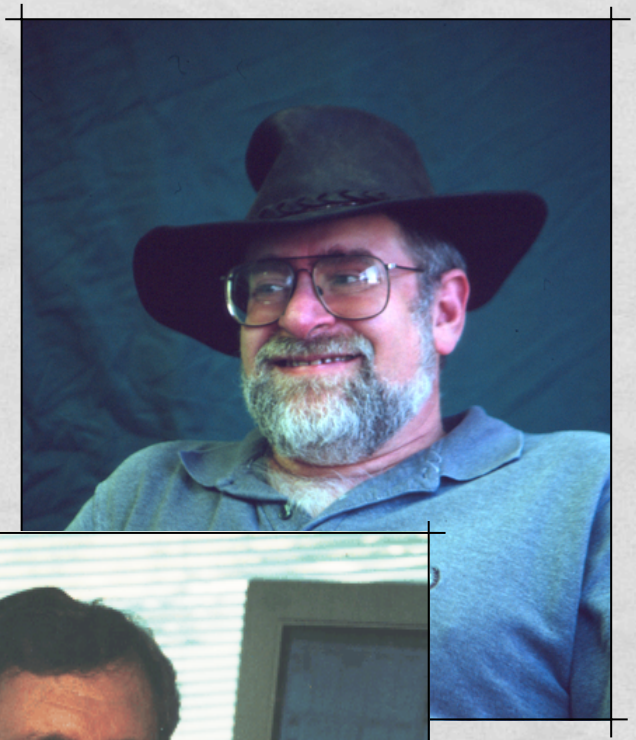
- Implement a set of classes to model natural numbers
- The classes must support:
  - addition
  - comparisons (returning Boolean instances)
- The implementation must not use any existing numeric types!

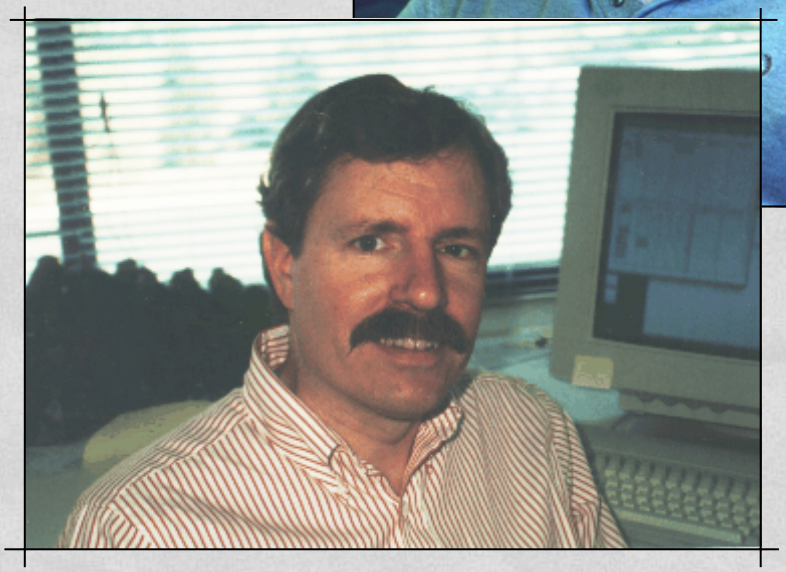
LINKED LIST

- Implement a linked list
  - add_head, add_tail
  - forward and reverse iterators
- Use polymorphic dispatch for all branching/decisions
- Don't use Python lists

SELF: THE POWER OF SIMPLICITY

- David Ungar and Randall B. Smith
- Original paper, OOPSLA 1987
- Lisp and Symb. Comp. paper, 1991





PROTOTYPES

Q1,2

NAMED SLOTS INSTEAD OF VARIABLES

Q3

METHODS ARE OBJECTS

Q4,5

NO CONTROL STRUCTURES

Q6

- Monday — Self-ish ideas in Python
- No class tomorrow — project work day
  - Team meetings with me start **next** week