Class 20: Objects

I invented the term *Object-Oriented*, and I can tell you I did not have C++ in mind.
— Alan Kay

CS150: Computer Science
University of Virginia
Computer Science

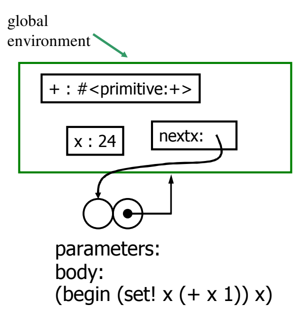
David Evans
http://www.cs.virginia.edu/evans

Menu

- A better counter
- Finishing Fish
- Programming with Objects

nextx from Class 18

```
(define x 0)
(define (nextx)
  (set! x (+ x 1))
  x)
> (nextx)
1
> (set! x 23)
> (next x)
24
```



A Better Counter

- The place that keeps track of the count would be part of the counter, not part of the global environment
- Can we do this?

Recall from Lecture 19:

Application

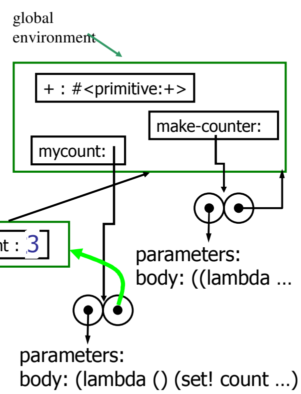
1. Construct a new frame, enclosed in the environment of this procedure
2. Make places in that frame with the names of each parameter
3. Put the values of the parameters in those places
4. Evaluate the body in the new environment

A Better Counter

```
(define (make-counter)
   ((lambda (count)
      (lambda ()
        (set! count (+ 1 count))
        count))
      0))
```

```
(define (make-counter)
  ((lambda (count)
     (lambda ()
       (set! count (+ 1 count))
       count))
   0))
```

```
> (define mycount
     (make-counter))
> (mycount)
1
> (mycount)
2
> (mycount)
3
```



An Even Better Counter

```
(define (make-ocounter)
  ((lambda (count)
     (lambda (message)
       (if (eq? message 'reset) (set! count 0)
          (if (eq? message 'next)
             (set! count (+ 1 count))
             (if (eq? message 'how-many)
                count)))))
   0))
```


Using Counter

```
> (define bcounter (make-ocounter))
> (bcounter 'next)
> (bcounter 'next)
> (bcounter 'next)
> (bcounter 'how-many)
3
> (bcounter 'reset)
> (bcounter 'how-many)
0
```


Objects

- When we package state and procedures together we have an *object*
- Programming with objects is *object-oriented programming*

Finishing Fish

Recap (through class 17)

- May 1941: Nazis start using Lorenz cipher to communicate between conquered European capitals
  - Allies know Baudot code, 2 sets of 5 wheels from test messages
- August 1941: Operator retransmits message (with abbreviations)
  - Allies learn one 4000-character key by XORing intercepted messages and then guessing possible plaintexts to find a pair that makes sense
- ~Feb 1942: Bill Tutte determines structure of Lorenz machine by analyzing key

Double Delta

- Combine two channels:

$\Delta Z_{1,i}$ XOR $\Delta Z_{2,I}$ =

$\Delta M_{1,i}$ XOR $\Delta M_{2,i}$ $> ½$

XOR $\Delta X_{1,i}$ XOR $\Delta X_{2,i}$ $= ½$ (key)

XOR $\Delta S_{1,i}$ XOR $\Delta S_{2,i}$ $> ½$

Message is in German, more likely following letter is a repetition than random

S-wheels only turn some of the time (when M-wheel is 1)

Prob[$\Delta Z_{1,i}$ XOR $\Delta Z_{2,i}$ XOR $\Delta X_{1,i}$ XOR Δ X2,i = 0] = 0.55

So, if guess of initial configuration is correct, generated X will have this property and we will see more 0s than 1s

Using the Advantage

- If the guess of X is correct, should see higher than ½ of the double deltas are 0
- Try guessing different configurations to find highest number of 0 double deltas
- Problem:

  \# of double delta operations to try one config

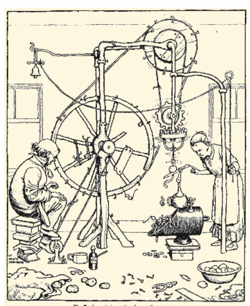
  = length of Z * length of X

  = for 10,000 letter message = 12 M for each setting * 7 XOR per double delta

  = 89 M XOR operations

Heath Robinson

- Dec 1942: Decide to build a machine to do these XORs quickly, due June 1943
- Apr 1943: first Heath Robinson machine is delivered!
- Intercepted ciphertext on tape:
  - 2000 characters per second (12 miles per hour)
  - Needed to perform 7 XOR operations each ½ ms

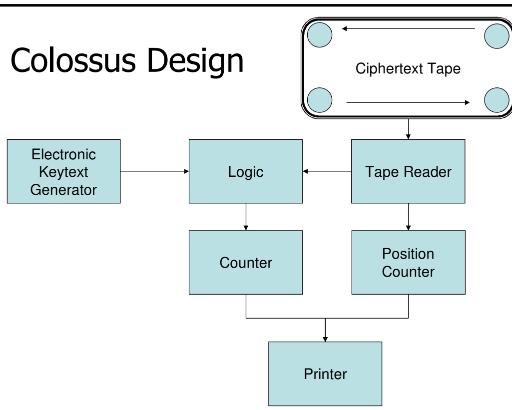


Heath Robinson, British Cartoonist (1872-1944)

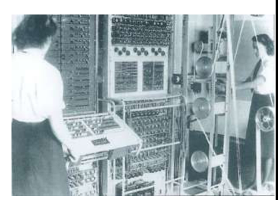
Colossus

- Heath Robinson machines were too slow
- Colossus designed and first built in Jan 1944
- Replaced keytext tape loop with electronic keytext generator
- Speed up ciphertext tape:
  - 5,000 chars per second = 30 mph
  - Perform 5 double deltas simultaneously
  - Speedup = 2.5X for faster tape * 5X for parallelism

Colossus Design



Colossus



- 10 Colossi machines operating by end of WWII
- Decoded messages (63M letters total) that enabled Allies to know German troop locations to plan D-Day
- Destroyed after war, kept secret until 1970s, documents released in late 90s

Object-Oriented Programming

Simula

- Considered the first "object-oriented" programming language
- Language designed for *simula*tion by Kristen Nygaard and Ole-Johan Dahl (Norway, 1962)
- Had special syntax for defining classes that packages state and procedures together

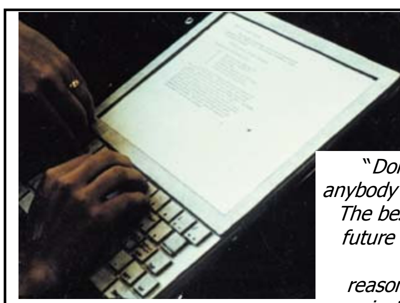
Counter in Simula

```
class counter;
  integer count;
  begin
    procedure reset(); count := 0; end;
    procedure next();
        count := count + 1; end;
   integer procedure how-many();
        how-many := count; end;
  end
```


XEROX Palo Alto Research Center (PARC)

1970s:

- Bitmapped display
- Graphical User Interface
  - Steve Jobs paid $1M to visit and PARC, and returned to make Apple Lisa/Mac
- Ethernet
- First personal computer (Alto)
- PostScript Printers
- **Object-Oriented Programming**



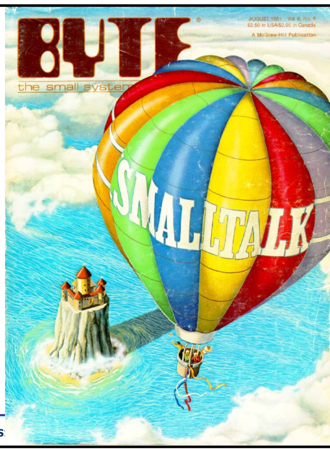
Dynabook, 1972

(Just a model)

"Don't worry about what anybody else is going to do… The best way to predict the future is to invent it. Really smart people with reasonable funding can do just about anything that doesn't violate too many of Newton's Laws!"
— Alan Kay, 1971

Dynabook 1972

- Tablet computer
- Intended as tool for learning
- Kay wanted children to be able to program it also
- Hallway argument, Kay claims you could define "the most powerful language in the world in a page of code"
- Proof: Smalltalk
  - Scheme is as powerful, but takes two pages



BYTE Magazine, August 1981

Smalltalk

- Everything is an *object*
- Objects communicate by sending and receiving *messages*
- Objects have their own state (which may contain other objects)
- How do you do 3 + 4?

  send the object **3** the message **"+ 4"**

Counter in Smalltalk

class name counter
  instance variable names count
  new count <- 0
  next count <- count + 1
  how-many ^ count

Counter in Scheme

```
(define (make-ocounter)
  ((lambda (count)
     (lambda (message)
       (if (eq? message 'reset) (set! count 0)
           (if (eq? message 'next)
               (set! count (+ 1 count))
               (if (eq? message 'how-many)
                   count)))))
   0))
```


Counter in Scheme using let

```
(define (make-ocounter)
  (let ((count 0))
    (lambda (message)
      (if (eq? message 'reset) (set! count 0)
          (if (eq? message 'next)
              (set! count (+ 1 count))
              (if (eq? message 'how-many)
                  count))))))
```


Defining **ask**

(ask *Object Method*)

```
> (ask bcounter 'how-many)
0
> (ask bcounter 'next)
> (ask bcounter 'how-many)
1
```

```
(define (ask object message)
  (object message))
```

Who was the first object-oriented programmer?

By the word operation, we mean any process which alters the mutual relation of two or more things, be this relation of what kind it may. This is the most general definition, and would include all subjects in the universe. Again, it might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine... Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent.

Ada, Countess of Lovelace, around 1830

Charge

- Alan Kay talk at UVa!
  - "Children as Digital Scholars"
  - Wednesday, 10am
  - Newcomb Hall South Meeting Room
- Wednesday (and PS6):
  - Programming with objects
  - Inheritance