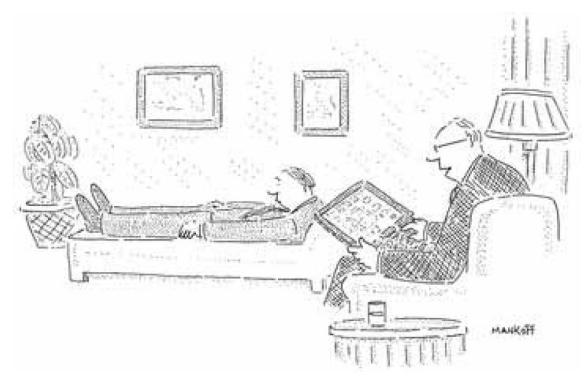
#### CS 32 LECTURE 1: 00PS



"Oops! I just deleted all your files. Can you repeat everything you've ever told me?"

#### Textbooks

- Problem Solving in C++
  - (CS 16) Chapters 10-18
- Data Structures with C++
  - (CS 24) Chapters 12-14
- Reader
  - SBPrinter at UCen

### Grading

- Labs 20%
- Programming Assignments 20%
- $3 \times$  "thirdterm" exams, each 20%

### OOP

- Stands for Object-Oriented Programming
  - As if you didn't know
- A way of factoring code
- A way of encapsulating code

### CS 16

- Have seen arrays and strings
  - And maybe vectors
- Right up to structs
- Next come objects

#### CS 24

- Have already been using classes
- And maybe templates
- Did you get as far as tree-traversal?
- Next come objects

#### Boundaries

- One of the basic ways we structure our code is with boundaries
- Could call them interfaces
  - It worked for living organisms
    - Several times

#### Access

- Your object is the combination of
  - state (values)
  - behavior (functions)

# Why Centralize?

- Easier to debug
  - Responsibility (aka blame) is easier to locate
- Easier to test
  - Individual objects are very amenable to unit testing

# Encapsulation

- Why does the tortoise grow its shell?
- You want to be similarly defensive with your code
- You manage access to the internal state

### But There's More

- "I got my money the old-fashioned way. I inherited it."
- Things and their features (state and functions) are not completely disjoint
  - Car has many similar features as Truck
  - Should a Car be a kind of Truck?
  - Should a Truck be a kind of Car?

### DRY

- And there is always the desire to not repeat mechanical patterns but to factor them
  - Abstraction vs Specialization
- OOP is driven as much by code reuse as it is by encapsulation

#### Snake in the Garden

- Hierarchical types need hierarchical type-checking
- An object's type at runtime may be different (a descendant) from the one declared to the compiler
- Opens up a whole can o' worms

# History

- The idea was in the air in the 60s and 70s
- Smalltalk at Xerox PARC was the first to go all-in
- In the 80s came C++ and Object Pascal
- Java, C#
- JavaScript?
- Objective-C, Swift

### Top Down

• Tree-based inheritance structure

# Subtypes

- Inheritance is isomorphic to containment
- Subtypes are isomorphic to subsets
- It all seems so simple
- But...

#### Tension

- The services an object provides can only <u>expand</u> as it gets more specialized
  - It still has to provide all the services of its parent classes

#### Contravariance

- Two opposing forces
  - Subclassing narrows the type
  - but expands the services

### Another Snake

- This led to unexpected problems in language evolution
- Eiffel
  - Santa Barbara grown!
  - Clear clean Pascal-like language

#### Eiffel

```
class
    POINT
inherit
   ANY
create
    make, make_origin
feature -- Initialization
    make (a_x, a_y: INTEGER)
    make_origin
feature -- Access
    x: INTEGER assign set_x -- Horizontal axis coordinate
    y: INTEGER assign set_y -- Vertical axis coordinate
feature -- Element change
    set_x (a_x: INTEGER)
    set_y (a_y: INTEGER)
end
```

#### Programming Contract

- API between separate objects is a contract
- So is the inheritance path in an OO design
- This led to type-safety issues in Eiffel

# Multiple Inheritance

- Indirectly, maybe inadvertently
- Is repeated inheritance idempotent?
- In C++, issues with virtual vs. nonvirtual
- People got tired of it

# Single Inheritance

- Class can inherit <u>implementation</u> from only one parent
- Can implement any number of interfaces/protocols/abstract classes

### LSP

- Not a new drug
- Liskov Substitution Principle
  - Named after Barbara Liskov
- Subtype must always be OK in context expecting higher type
  - Defining "OK" takes work

### **Co-Contravariance**

- Descendant methods must require less, guarantee more
  - Preconditions weaker
  - Postconditions stronger

#### Next Gen



- Comes with VM also
- Lacks generics
- No value types

#### **C**#

- Started out same time as Java
- Accused of copying
- Java and C# have diverged

## JavaScript

- What's the big deal with top-down analysis anyway?
- Just take what exists, and modify it
  - <u>Self</u>
  - JavaScript
- Prototype-Based Programming

#### Add-Ons

- Some languages have added OOP features
  - Perl
  - Python
  - JavaScript
  - PHP
  - MATLAB
  - Lua

#### New Built-In



- <u>Dart</u>
- <u>Scala</u>



#### Read!

- Reader #2 (<u>Not</u> #1)
  - (most important)
- (CS 16) Problem Solving Chapter 10
  - Less important
- (CS 24) Data Structures Chapter 12
  - Even less (but don't skip!)