Objects: Why?

- A better way to organize data and operations on it.
- A better way to model:
  - the real world
  - problems to be solved
Objects: Why? II

Better how?

Produce programs that are easier to:
  - understand
  - fix
  - enhance
  - reuse
Objects

- Some Concepts
  - Abstraction
  - Virtual Machines
- What Makes An Object
  - Abstraction / Encapsulation
  - Inheritance
  - Polymorphism
Abstraction

“A means of achieving stepwise refinement by suppressing unnecessary details and accentuating relevant details”—Schach, Classical and Object-Oriented Software Engineering, 4th edition, p. 189

To create a good abstraction, you must recognize what is important.
Abstract Machine a.k.a. Virtual Machine

In any system, at any level, define an interface (data & actions) and hide all the details below it.

This can be hardware, software, firmware, or any other kind of ware.

A very powerful concept, found throughout computer science (and life).

But not always recognized! J
Virtual Machine II

- If you replace a virtual machine with another one that has the same interface, the higher level should never know the difference.

- Original meaning of computer architecture, as used in the IBM System/360 (1964).
Virtual Machine Examples

- Emacs
- Languages
  - ANSI C, Java
- Unix command line
  - Bourne Shell
  - Bourne-Again Shell
  - C-shell, Tcsh
  - Korn Shell
- Unix system calls
- Java Virtual Machine
- Pentium Instruction Set
- UART (serial port chip)
- Web server
  - Protocols
  - Configuration files
- Web browser
What is an Object?

- “An instantiation of an abstract data type”—Schach and just about everyone else.
- Each object is an instance of a class.
  - A class defines features.
  - Each object in that class has those features.
- A class is an abstract data type which supports inheritance.
- An object is not the same as a variable which holds a reference (handle) to it.
Features of an Object

- Abstraction / Encapsulation
- Inheritance
- Polymorphism
Abstraction & Encapsulation

Data and the operations which act on it are bundled together, forming a class.

Details of the data and operations are usually hidden from outside the class. “Data hiding”

This is encapsulation, a way of implementing abstraction.
Inheritance

- **Superclass** = parent class
- **Subclass** = derived class
- Each subclass inherits all attributes and methods of the superclass, but can add to or override them.

**UML:** Open arrow shows superclass, “is-a” relationship.
Polymorphism

- A variable declared with one class can be assigned an object of that class or any subclass, but not vice-versa.
- Automatically, the correct method is dynamically bound (at runtime).
Suppose you need to implement the open() function for files on different types of storage media.

```c
int open(FILE *f) {
    determine_type(f);
    if (f->type == HD)
        open_hd(f);
    else if (f->type == FD)
        open_fd(f);
    else if (f->type == TAPE)
        open_tape(f);
}
```
You automatically get the right method at run time.
Implementing Objects

z Remember how C++ was first implemented.
  y As a preprocessor for the C compiler.
  y The various fields and functions had funny machine-generated names.
Next Time

z An introduction / review of Java