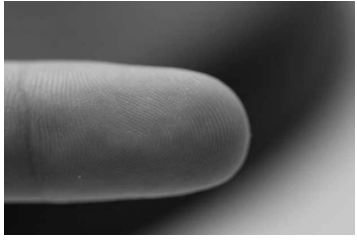
UC Berkeley EECS
Sr Lecturer SOE
Dan Garcia

The Beauty and Joy of Computing

Lecture #5
Programming Paradigms

Your Typing Style Can Give You Away…

Researchers have used a mechanical "intelligent keyboard" that records the force and time between keys to determine who is typing with high accuracy. Could prevent a hacker who gets your password from using the computer!



www.news.gatech.edu/2015/01/22/self-powered-intelligent-keyboard-could-provide-new-layer-security

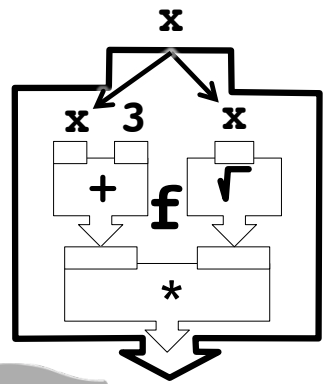
What are Programming Paradigms?

- "The concepts and abstractions used to represent the elements of a program (e.g., objects, functions, variables, constraints, etc.) and the steps that compose a computation (assignation, evaluation, continuations, data flows, etc.)."
- Or, a way to classify the style of programming.

Functions Review

- **Computation is the evaluation of functions**
  - Plugging pipes together
  - Function: ≥ 0 inputs, 1 output
  - Functions can be input!
- **Features**
  - No state
    - E.g., variable assignments
  - No mutation
    - E.g., changing variable values
  - No side effects
    - E.g., nothing else happens
- **Examples (tho not pure)**
  - Schme, Scratch, BYOB, Snap!

$$f(x) = (x+3) * \sqrt{x}$$



Imperative Programming

- **"Sequential" Programming**
- **Computation a series of steps**
  - Assignment allowed
    - Setting variables
  - Mutation allowed
    - Changing variables
- **Like following a recipe. E.g.,**
- **Procedure f(x)**
  - ans = x
  - ans = $\sqrt{ans}$
  - ans = (x+3) * ans
  - return ans
- **Examples: (tho not pure)**
  - Pascal, C

$$f(x) = (x+3) * \sqrt{x}$$

(Cal) Which of the following is true?

a) Functional-style code is actually imperative-style code

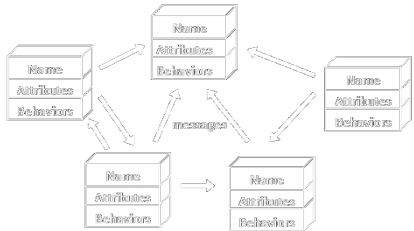
b) Imperative-style code is actually functional-style code

c) Both (a) and (b)

d) Neither (a) nor (b)

Object-Oriented Programming (OOP)

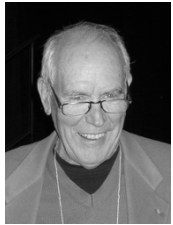
- **Objects as data structures**
  - With methods you ask of them
    - These are the behaviors
  - With local state, to remember
    - These are the attributes
- **Classes & Instances**
  - Instance an example of class
  - E.g., Fluffy is instance of Dog
- **Inheritance saves code**
  - Hierarchical classes
  - E.g., pianist special case of musician, a special case of performer
- **Examples (tho not pure)**
  - Java, C++



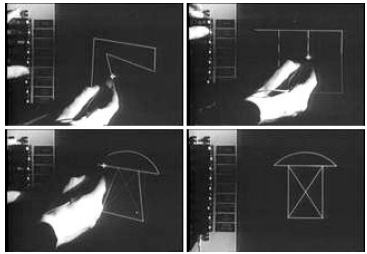
www3.ntu.edu.sg/home/ehchua/programming/java/images/OOP-Objects.gif

OOP Example: SketchPad

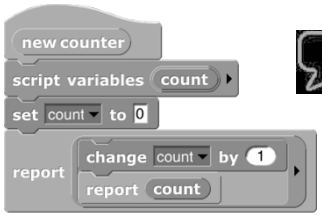
- **Dr. Ivan Sutherland**
  - "Father of Computer Graphics"
  - 1988 Turing Award ("Nobel prize" for CS)
  - Wrote Sketchpad for his foundational 1963 thesis
- **The most impressive software ever written**
- **First…**
  - Object-oriented system
  - Graphical user interface
  - non-procedural language

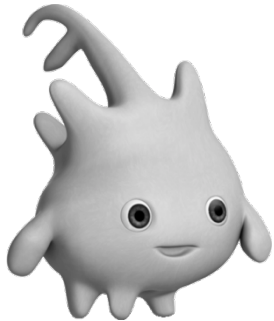


Spent the past few years doing research @ Berkeley in EECS dept!



OOP in Snap!





broadcast Away, animals!

(Cal) Which of the following is true?

a) Objects can only delete other objects

b) Objects can only contain other objects

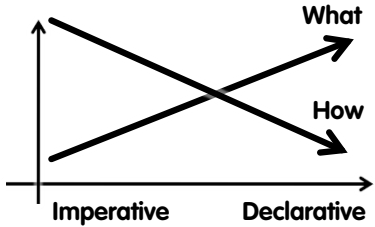
c) Objects can both delete and contain other objects

d) Objects can neither delete or contain other objects, they can only send messages to them.

Declarative Programming

- **Express <u>what</u> computation desired without specifying <u>how</u> it carries it out**
  - Often a series of assertions and queries
  - Feels like magic!
- **Sub-categories**
  - Logic
  - Constraint
    - We saw in Sketchpad!
- **Example: Prolog**

Thatnks to Anders Hejlsberg
"The Future of C#" @ PDC2008
channel9.msdn.com/pdc2008/TL16/



Declarative Programming Example

- **Five schoolgirls sat for an examination. Their parents – so they thought – showed an undue degree of interest in the result. They therefore agreed that, in writing home about the examination, each girl should make one true statement and one untrue one. The following are the relevant passages from their letters:**

- **Betty**
  - Kitty was 2nd
  - I was 3rd
- **Ethel**
  - I was on top
  - Joan was 2nd
- **Joan**
  - I was 3rd
  - Ethel was last
- **Kitty**
  - I came out 2nd
  - Mary was only 4th
- **Mary**
  - I was 4th
  - Betty was 1st

Most Languages are Hybrids!

- **This makes it hard to teach to students, because most languages have facets of several paradigms!**
  - Called "Multi-paradigm" languages
  - Scratch, BYOB, Snap*!* too
- **It's like giving someone a juice drink (with many fruit in it) and asking to taste just one fruit!**

(Cal) Of 4 paradigms, what's the *most* powerful?

a) Functional

b) Imperative

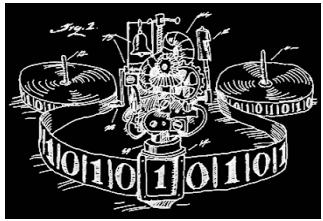
c) OOP

d) Declarative

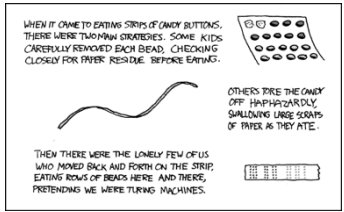
e) All equally powerful

Turing Completeness

- **A Turing Machine has an infinite tape of 1s and 0s and instructions that say whether to move the tape left, right, read, or write it**
  - Can simulate any computer algorithm!
- **A Universal Turing Machine is one that can simulate a Turing machine on any input**
- **A language is considered Turing Complete if it can simulate a Universal Turing Machine**
  - A way to decide that one programming language or paradigm is just as powerful as another



Turing Machine by Tom Dunne



Xkcd comic "Candy Button Paper"

Ways to Remember the Paradigms

- **Functional**
  - Evaluate an expression and use the resulting value for something
- **Object-oriented**
  - Send messages between objects to simulate the temporal evolution of a set of real world phenomena
- **Imperative**
  - First *do this* and next *do that*
- **Declarative**
  - Answer a question via search for a solution

www.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html

Summary

- **Each paradigm has its unique benefits**
  - If a language is Turing complete, it is equally powerful
  - Paradigms vary in efficiency, scalability, overhead, fun, "how" vs "what" to specify, etc.
- **Modern languages usually take the best from all**
  - E.g., Snap!
    - Can be functional
    - Can be imperative
    - Can be object-oriented
    - Can be declarative