design intro
Processes

- Requirements
- Design
- Implementation
- Testing
Design

practices, principles, patterns
Design

practices, principles, patterns

e.g. diagrammatic modeling
Design

practices, principles, patterns

  e.g. “no forgery”
Design

practices, principles, patterns

e.g. singleton
Input File

```
: square, red, (0,0), (1,0), (1,1), (0,1)
: sphere, blue, (-1,0), 1
: ...
```

“draw red square ...”

Shapes: square, circle
Colors: red, green, blue

graphics hardware
rule

• draw red circles
• draw green squares
• draw blue squares
• ignore everything else
pseudo code

Open file
While not at end of file
    read shape, color
    if shape is square then read four vertices
        if color is blue then “draw blue square...”
        if color is green then “draw green square...”
    else if shape is circle then read center and radius
        if color is red then then “draw red circle...”
    else skip this line of input
Close file
whoops ... I meant

- Shapes: square, circle, triangle
- Colors: red, blue, green, purple
- Rule:
  - Draw blue and purple squares
  - Draw red and green circles
  - Draw every triangle
  - Ignore everything else
Thank god for cut and paste.

hacker joe
pseudo code

Open file
While not at end of file
    read shape, color
    if shape is square then read four vertices
        if color is blue then “draw blue square…”
        if color is purple then “draw green square…”
    else if shape is circle then read center and radius
        if color is red then “draw red circle…”
        if color is green then then “draw red circle…”
    else if shape is triangle then read four vertices
        if color is blue then “draw blue square…”
        if color is purple then “draw green square…”
        if color is red then “draw red circle…”
        if color is green then then “draw red circle…”
    else skip this line of input
Close file
pseudo code

Open file
While not at end of file
    read shape, color
    if shape=square then read four vertices
    if color is blue then “draw blue square…”
    if color is purple then “draw green square…”
    else if shape=circle then read center and radius
    if color is red then “draw red circle…”
    if color is green then then “draw red circle…”
    else if shape=triangle then read four vertices
    if color is blue then “draw blue square…”
    if color is purple then “draw green square…”
    if color is red then “draw red circle…”
    if color is green then then “draw red circle…”
    else skip this line of input

Close file
whoops ... I meant
McAlister

I'm Jill
I want to be an engineer

I'm Raju
I want to be a physicist

HMC pre-frosh

new task

Bio: TG105

CS: Pryne
where are the CS pre-frosh?

somewhere in west

I want to sleep.
new task

McAlister

I'm Jill
I want to be an engineer

I'm Raju
I want to be a physicist

HMC pre-frosh

Bio: TG105

CS: Pryne

::
procedure

• post classroom assignments
• post campus map
• tell students to
  - refer to the posted notices
  - go where they belong
whoops

CS has been moved to pepsi room
procedure

• post classroom assignments
• post campus map
• tell students to
  - refer to the posted notices
  - go where they belong
strategies

functional ←→ object oriented

NOTE: this has nothing to do with “functional languages”
strategies

- functional
  - step by step procedure
- object oriented
  - delegate responsibility
strategies

functional approach
- control is concrete, specific
- rigid, hard to change

object oriented approach
- control is abstract, general
- adaptable, easy to change

change is inevitable!
History: functional → OO

• modularization
module

Open file
While not at end of file
  read shape, color
    if shape is square then read vertices
      testAndDrawSquare(color, vertices)
    if shape is circle then read center and radius
      testAndDrawCircle(color, center, radius)
  Close file
functional → OO

- modularization
- user-defined data types
user define data types

Open file
While not at end of file
  read shape, color
    if shape=square then read vertices and create theSquare
    test AndDrawSquare(theSquare)
    if shape=circle then read center, radius and create theCircle
    test AndDrawCircle(theCircle)
Close file
functional → OO

- modularization
- user-defined data types
- union (abstract data types)
abstract data types

Open file
While not at end of file
   theShape = readNextShape()
   testAndDrawShape(theShape)
Close file
functional → OO

- modularization
- user-defined data types
- union (abstract data types)
- encapsulation
encapsulation

Open file
While not at end of file
    theShape = readNextShape()
    theShape.testAndDraw()
Close file