1

# Object-Oriented Programming

2

# Object-Oriented Programming

- Motivation
  why is object-oriented programming useful?
- Object-Oriented Programming in Turing and CLOS
  classes and objects, inheritance, genericity in object-oriented Turing (OOT) and the common Lisp object system (CLOS)
- A Comparative Review of Object-Oriented Programming Languages
  Simula, Smalltalk, C++, Object Pascal

3

## Motivation

- Consider general tasks
  - Lifting object X
  - Computing area of X
  - Buying object X

*How do we do it? It depends on X!*

- Consider
  - Lifting a pebble vs lifting a boulder
  - Computing the area of a circle vs that of a square
  - Buying candy vs buying a car

- Approach: Have object X know about its operations
  - To perform action A on X, "send message A to X"

e.g., send "lift me" to boulder B
- send "compute my area" to circle C
- send "buy me" to house H


©1993 John Mylopoulos

4

## Objects and Classes

An object is anything we may want to talk about or operate on,
- e.g., boulder B (a particular boulder),
- circle C and house H

When we describe how to perform tasks, we do so for classes of objects, e.g., the class of all boulder objects, circle objects or house objects

Each description of a class consists of:

- a description of the class structure, i.e., what do instances of the class look like,
  - e.g., a circle consists of a radius and a centre point

- a description of the class operations (or, methods, or, class behaviours), i.e., how to perform operations on the instances of the class (or, what behaviours can instances of the class perform in response to external messages)
  - e.g., to calculate the area of a circle use formula...


©1993 John Mylopoulos

5



## An Example

Window
x-position
y-position
width
height

class

attributes/
slots/
instance
variables

instanceOf

instanceOf

*window1*
x-position 27
y-position 12
width 1200
height 1400

objects/
instances

attribute/
slot/
instance var
values

*window2*
x-position 100
y-position 33
width 50
height 50

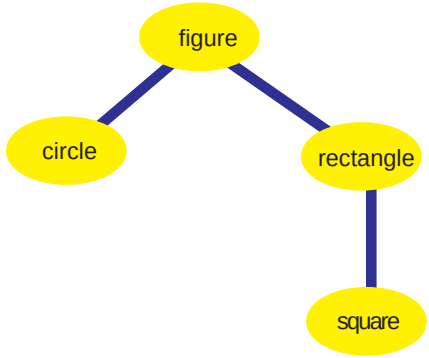

©1993 John Mylopoulos

6



## Subclasses

We conceptualize the world in terms of thousands of concepts. In an object-oriented program, these concepts are represented by classes. It is useful to organize all these classes into taxonomies, where each class is classified above or below others according to its generality/specificity



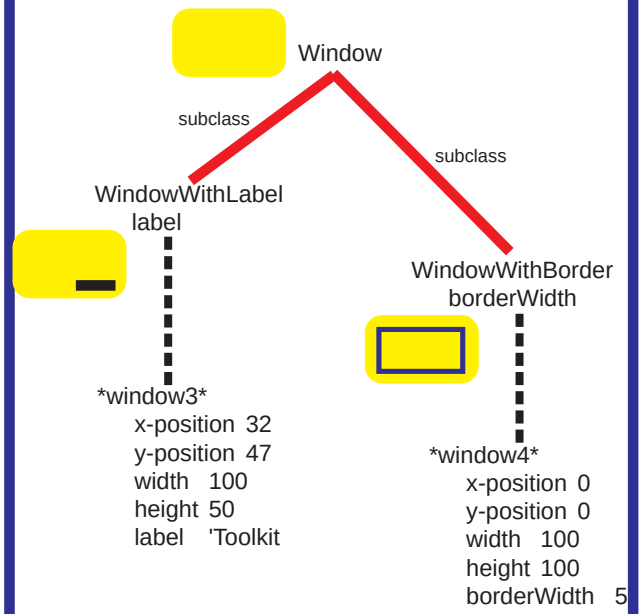


©1993 John Mylopoulos

7



## The Window Example Revisited






8



## Operations on Objects: Computing the Area of X

- Approach A
  Define a function which computes the area of an arbitrary object

```
(defun area (X)
  (cond   ((triangleP X) (* 0.5 (base X) (height X)))
          ((rectangleP X) (* (width X) (height X)))
          ((circleP X) (* pi (expt radius X) 2))) ))
```

*What if there are 100 classes?*

- Approach B
  Write separate subroutines

```
(defun area (X)
   (cond  ((triangleP X) (triArea X))
          ((rectangleP X) (recArea X))
          ((circleP X) (cirArea X)) ))

(defun triArea (X) (* 0.5 (base X) (height X)))
(defun recArea (X) (* (width X) (height X)))
(defun cirArea (X) (* pi (expt (radius X) 2)))
```

9



## Computing the Area of X

- Define separate classes for the different kinds of objects your program will deal with
  `Triangle, Rectangle, Circle`

- Associate methods with each class

| Shape | How to compute area |
|---|---|
| `Triangle` | area of triangle |
| `Rectangle` | area of rectangle |
| `Circle` | area of circle |

- Make X an instance of a class
  e.g., `c1` an instance of `Circle`

- To compute `A` of `X`: Call method `A` on `X`
  Get method from the `X`'s class

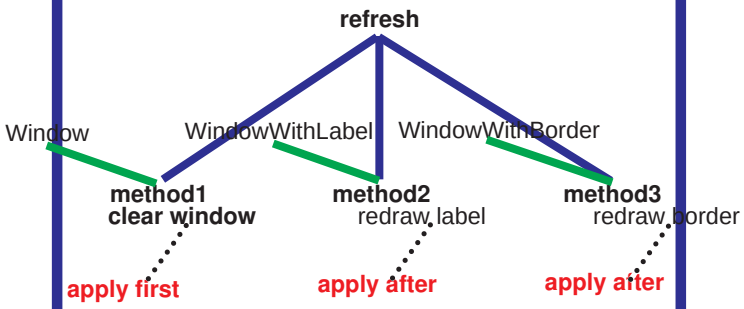
  e.g., call method `area` on `c1`
  get method from `c1`'s class, `Circle`
  apply `Circle`'s area method to `c1`

©1993 John Mylopoulos


10



## Methods Define Generic Functions



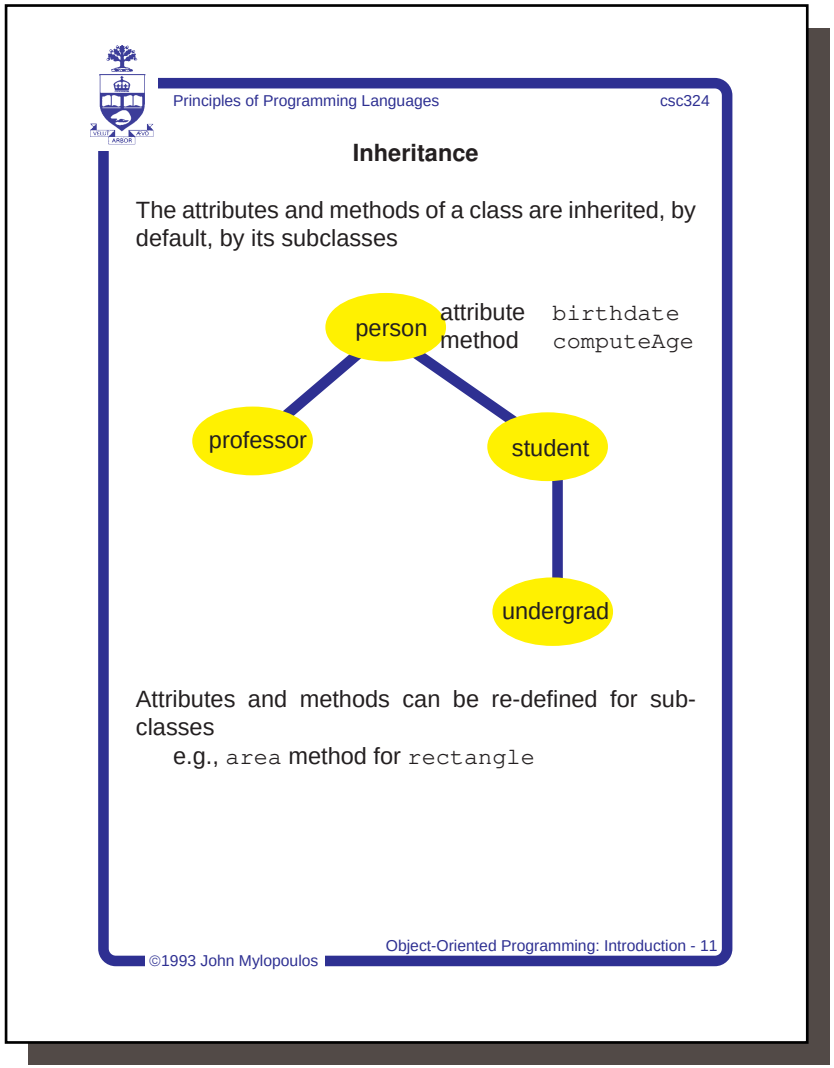

- A generic function is defined in terms of all methods with the same name (e.g., refresh, area).

- Each generic function behaves differently depending on its argument

*Generic functions are defined in a piecemeal fashion, along with the definition of new classes*

©1993 John Mylopoulos

11


Principles of Programming Languages
csc324
Inheritance
The attributes and methods of a class are inherited, by default, by its subclasses
person
attribute birthdate
method computeAge
professor
student
undergrad
Attributes and methods can be re-defined for sub-classes
e.g., area method for rectangle
Object-Oriented Programming: Introduction - 11
©1993 John Mylopoulos