Introduction

- Object-oriented design.
- Unified Modeling Language (UML).

System modeling

- Need languages to describe systems:
  - useful across several levels of abstraction;
  - understandable within and between organizations.
- Block diagrams are a start, but don’t cover everything.

Object-oriented design

- Object-oriented (OO) design: A generalization of object-oriented programming.
- Object = state + methods.
  - State provides each object with its own identity.
  - Methods provide an abstract interface to the object.

OO implementation in C++

```
class display {
  pixels : pixeltype[IMAX,JMAX];
public:
  display() { }
  pixeltype pixel(int i, int j) {
  return pixels[i,j]; }
  void set_pixel(pixeltype val, int
  i, int j) { pixels[i,j] = val; }
}
```

OO implementation in C

```
typedef struct { pixels:
  pixeltype[IMAX,JMAX]; } display;
display d1;
pixeltype pixelval(pixel *px, int i, int j)
  { return px[i,j]; }
```

Objects and classes

- Class: object type.
- Class defines the object's state elements but state values may change over time.
- Class defines the methods used to interact with all objects of that type.
  - Each object has its own state.

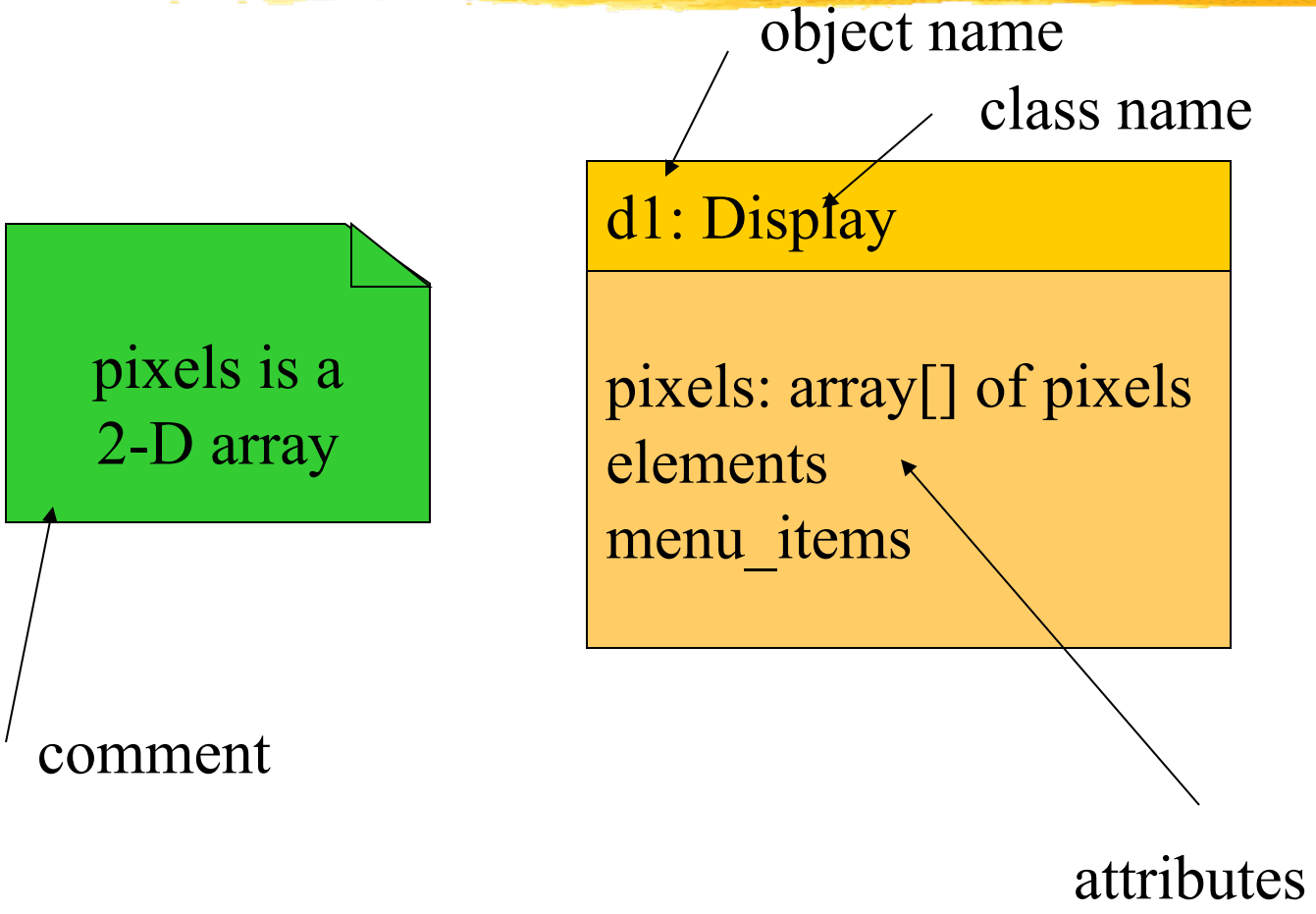
OO design principles

- Some objects will closely correspond to real-world objects.
  - Some objects may be useful only for description or implementation.
- Objects provide interfaces to read/write state, hiding the object's implementation from the rest of the system.

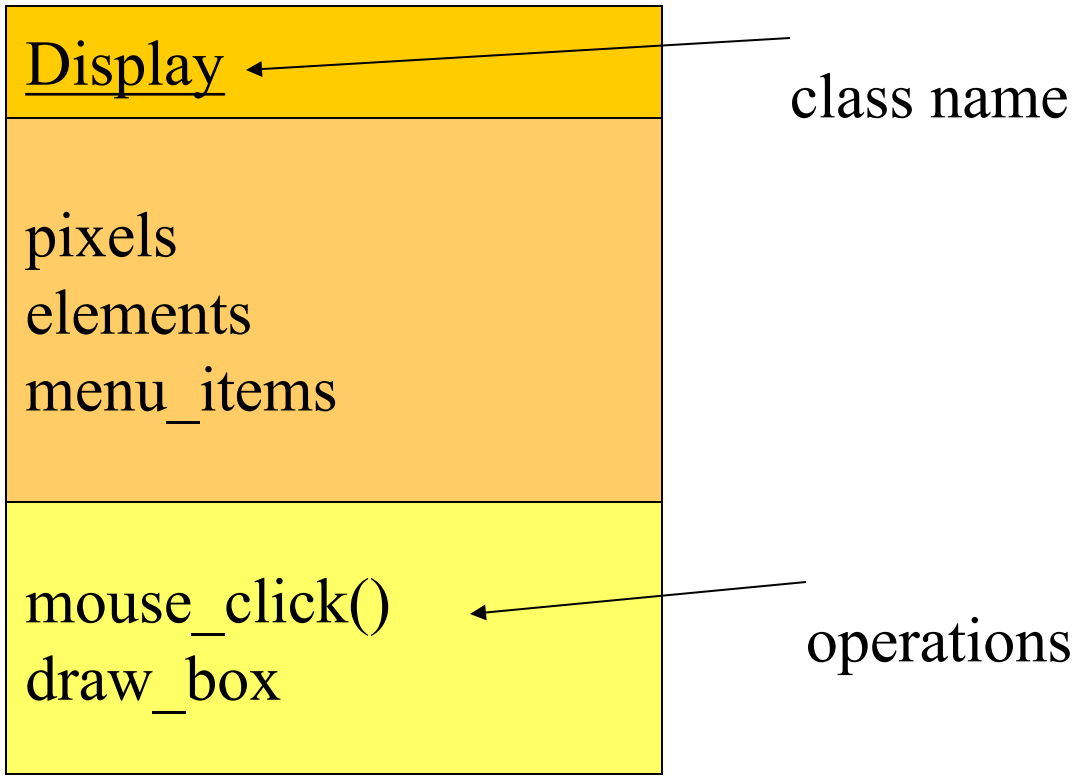
UML

- Developed by Booch et al.
- Goals:
  - object-oriented;
  - visual;
  - useful at many levels of abstraction;
  - usable for all aspects of design.

UML object



UML class



The class interface

- The operations provide the abstract interface between the class's implementation and other classes.
- Operations may have arguments, return values.
- An operation can examine and/or modify the object's state.

Choose your interface properly

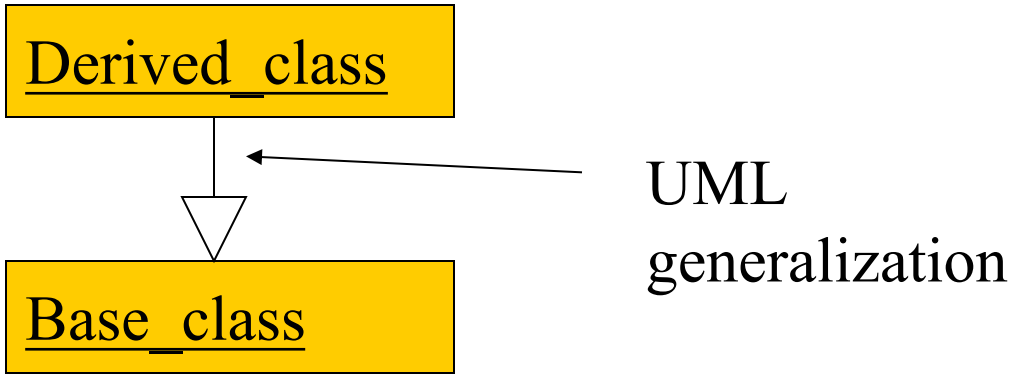
- If the interface is too small/specialized:
  - object is hard to use for even one application;
  - even harder to reuse.
- If the interface is too large:
  - class becomes too cumbersome for designers to understand;
  - implementation may be too slow;
  - spec and implementation are probably buggy.

Relationships between objects and classes

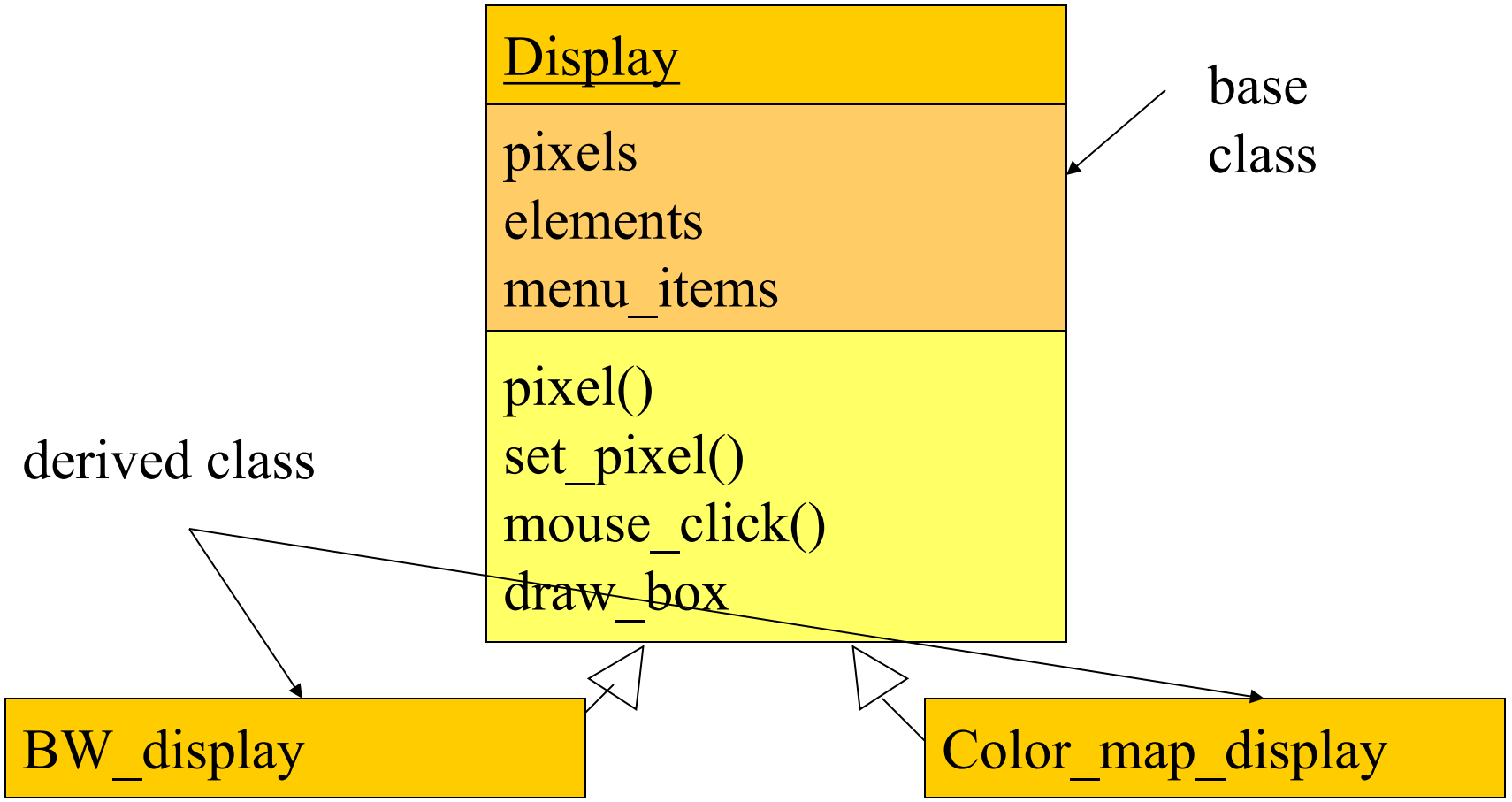
- Association: objects communicate but one does not own the other.
- Aggregation: a complex object is made of several smaller objects.
- Composition: aggregation in which owner does not allow access to its components.
- Generalization: define one class in terms of another.

Class derivation

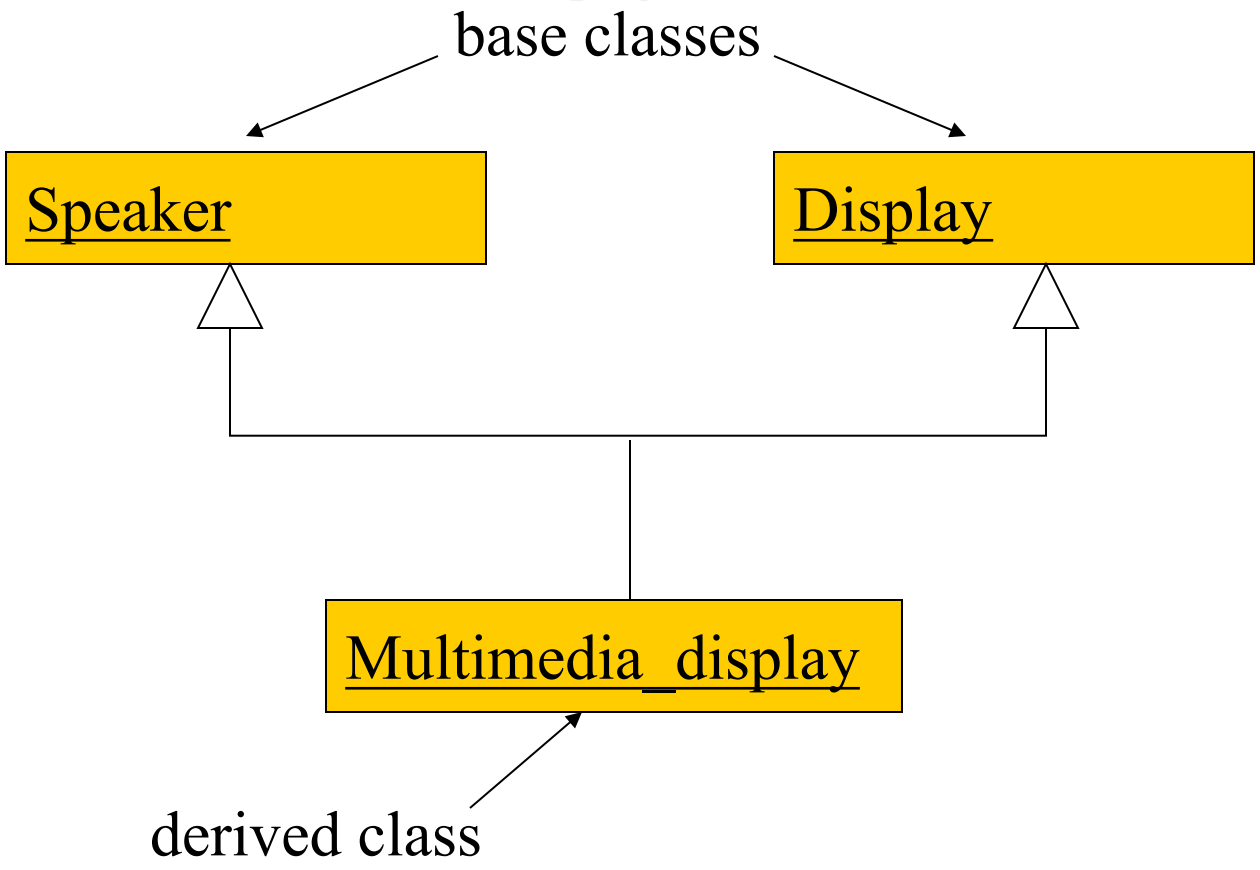
- May want to define one class in terms of another.
  - Derived class inherits attributes, operations of base class.



Class derivation example



Multiple inheritance

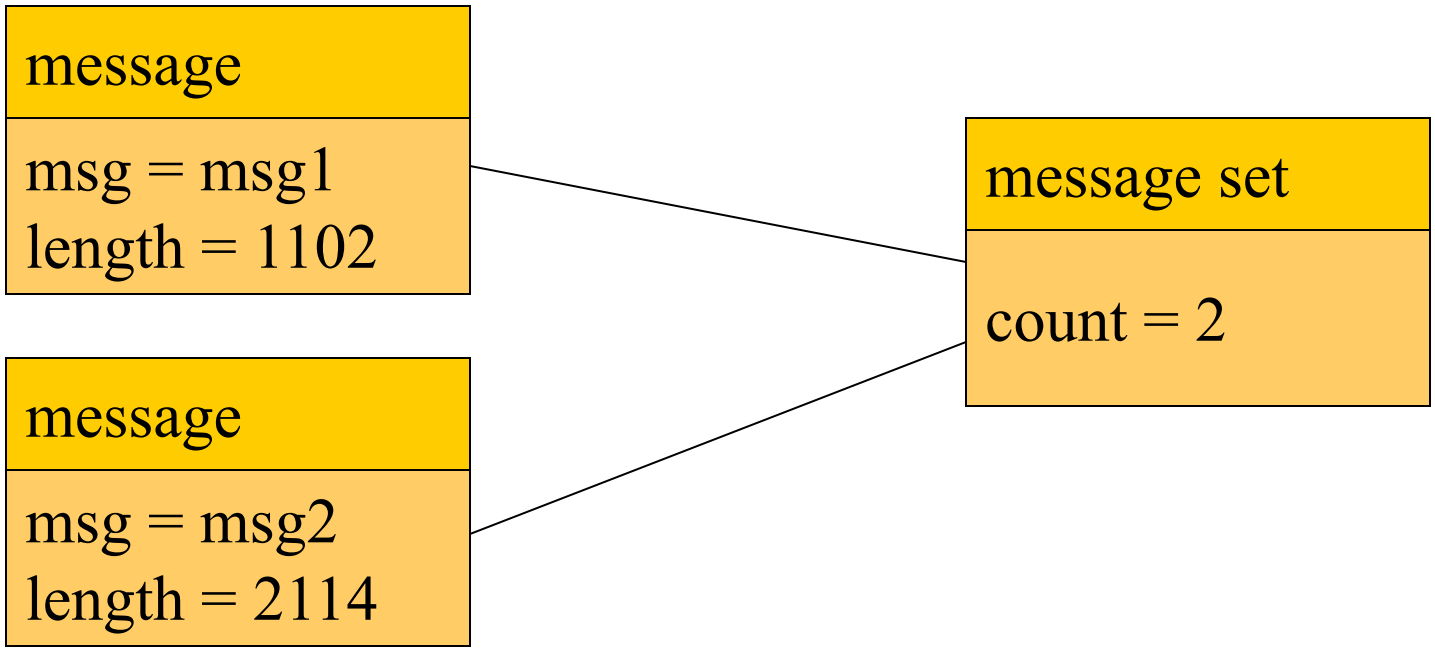


Links and associations

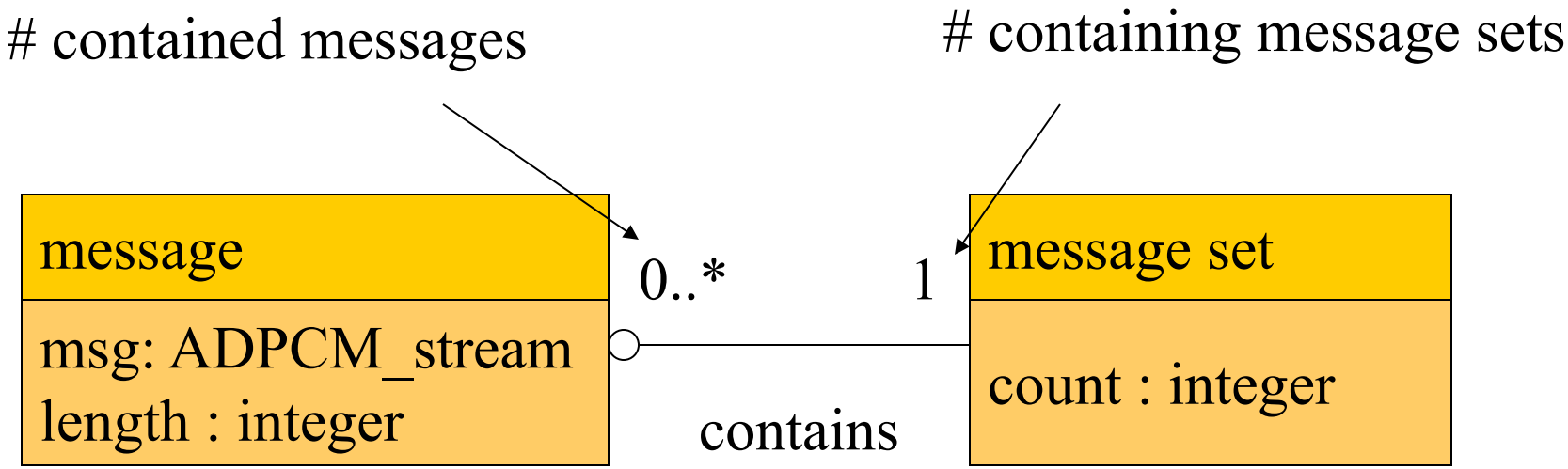
- Link: describes relationships between objects.
- Association: describes relationship between classes.

Link example

- Link defines the contains relationship:



Association example



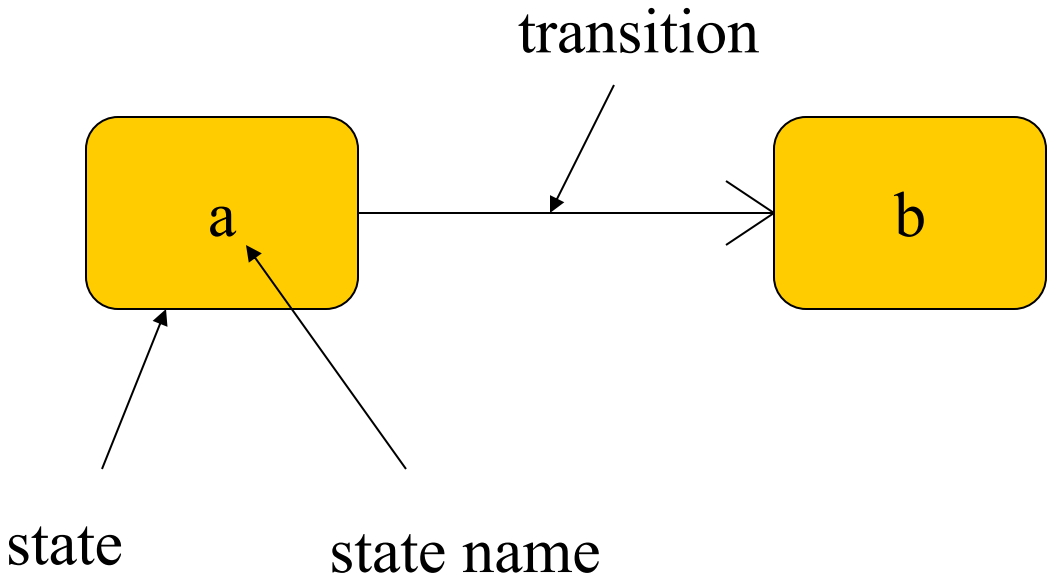
Stereotypes

- Stereotype: recurring combination of elements in an object or class.
- Example:
  - <<foo>>

Behavioral description

- Several ways to describe behavior:
  - internal view;
  - external view.

State machines



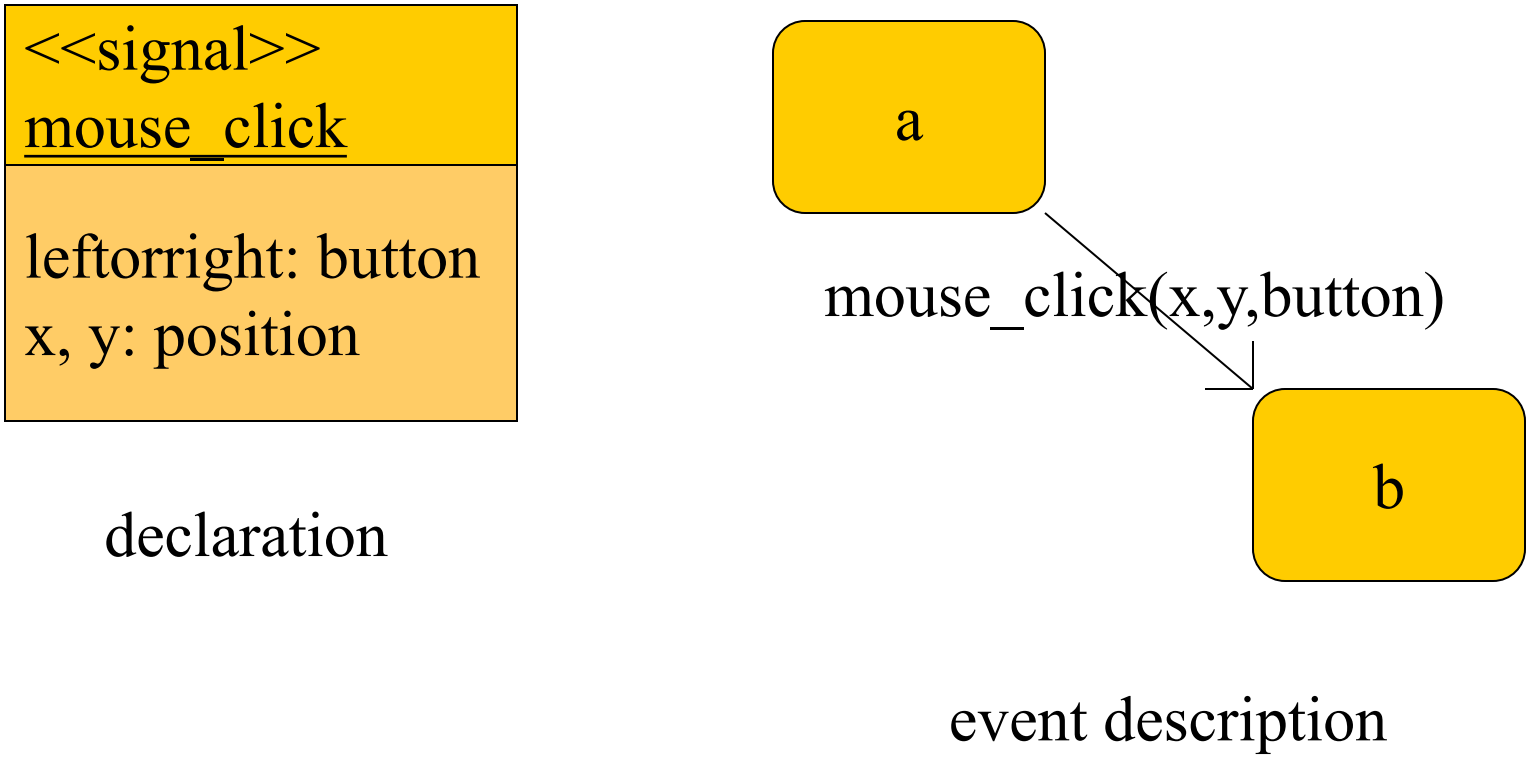
Event-driven state machines

- Behavioral descriptions are written as event-driven state machines.
  - Machine changes state when receiving an input.
- An event may come from inside or outside of the system.

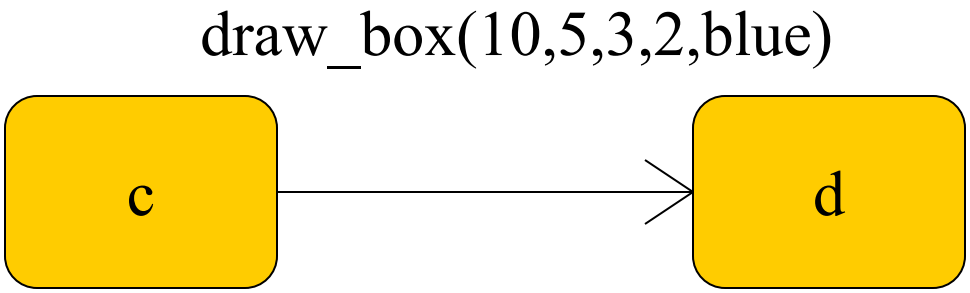
Types of events

- Signal: asynchronous event.
- Call: synchronized communication.
- Timer: activated by time.

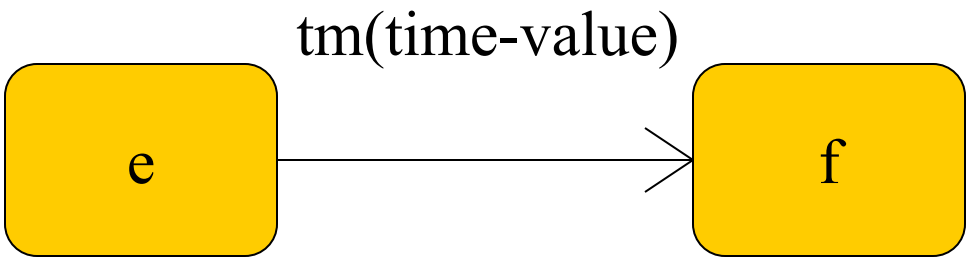
Signal event



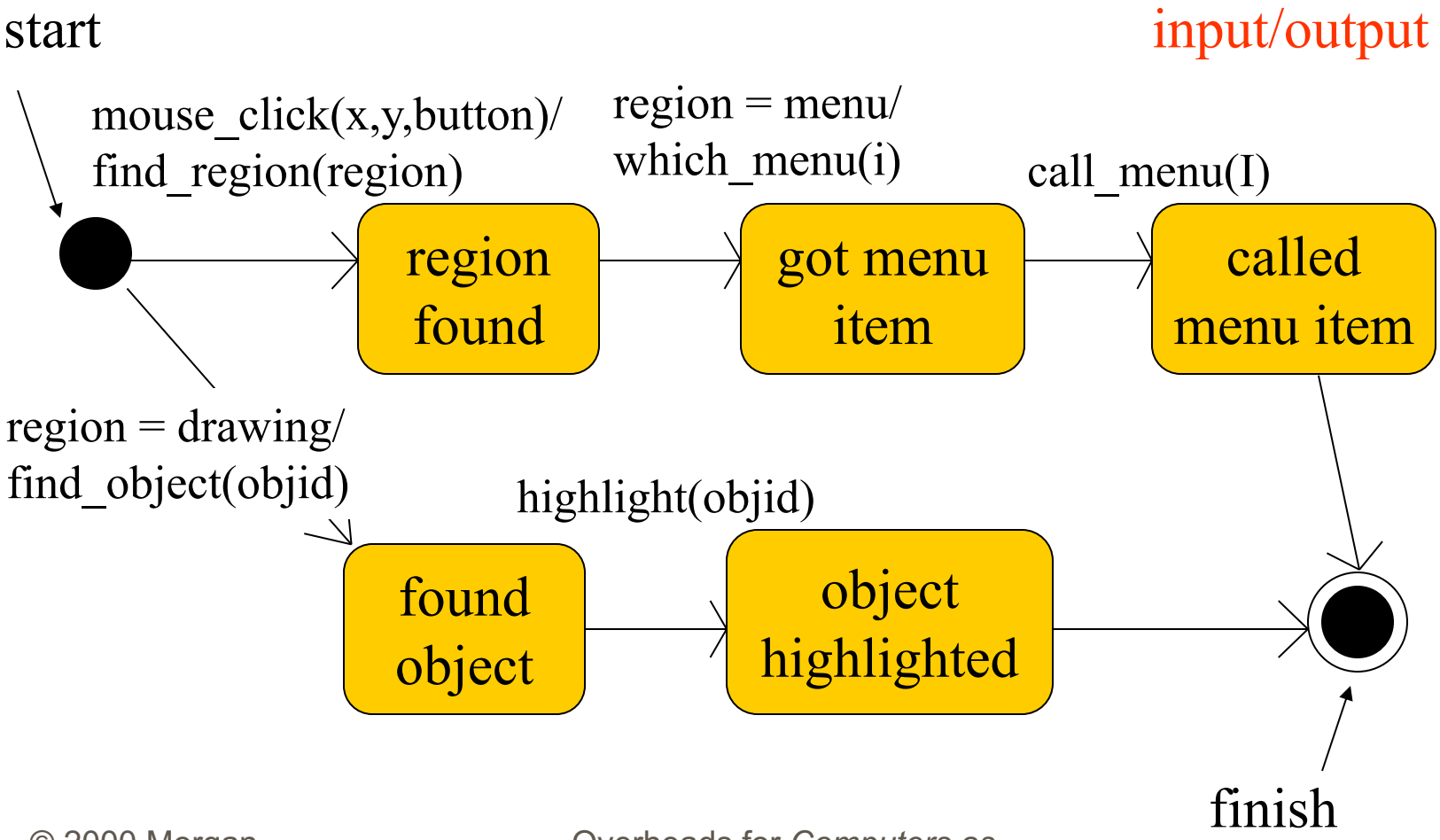
Call event



Timer event



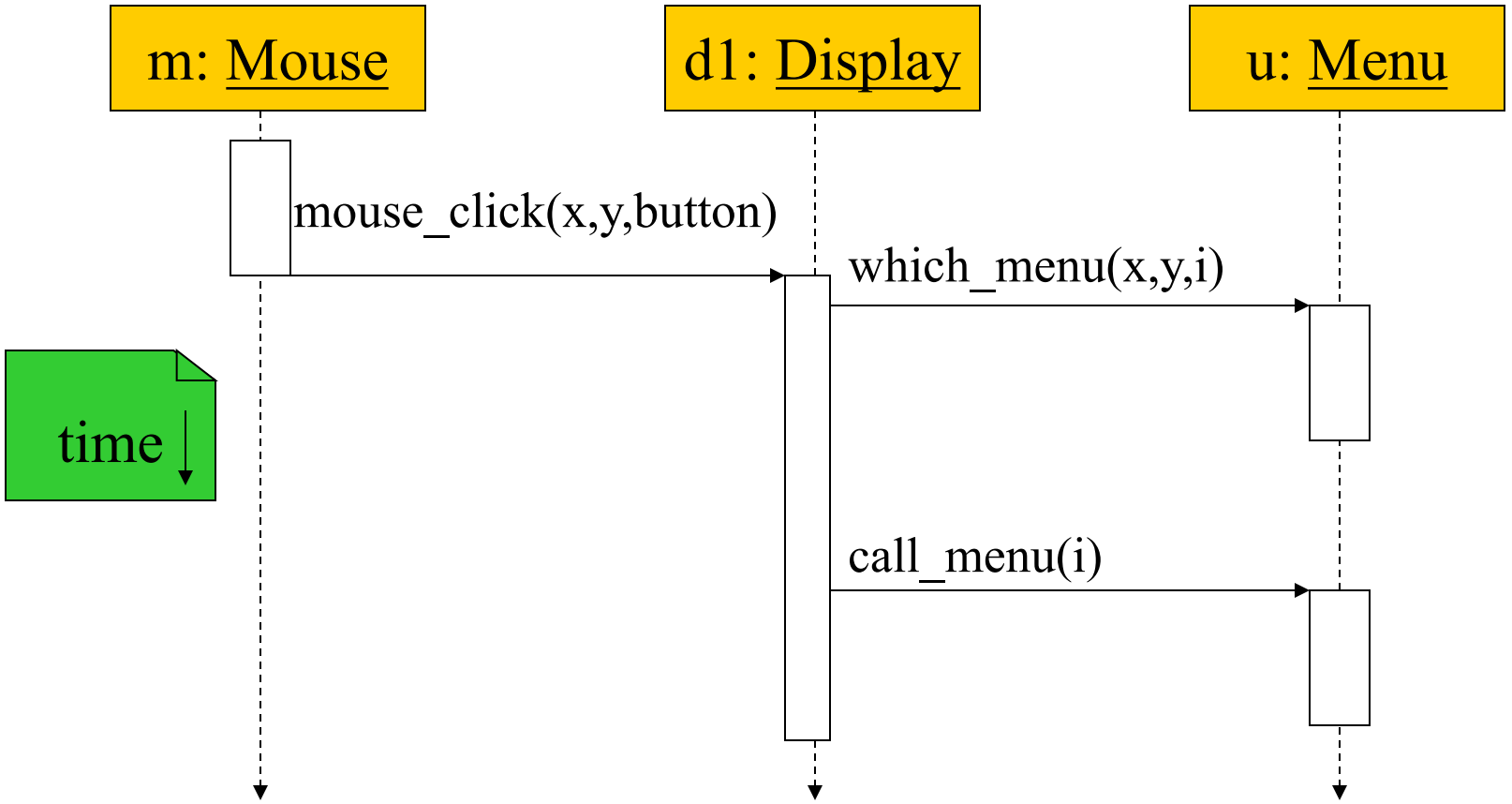
Example state machine



Sequence diagram

- Shows sequence of operations over time.
- Relates behaviors of multiple objects.

Sequence diagram example



Summary

- Object-oriented design helps us organize a design.
- UML is a transportable system design language.
  - Provides structural and behavioral description primitives.