Chapter 3: Part-A
Classes, Objects, and Basic Structural Modeling in UML

H.S. Sarjoughian

CSE 460: Software Analysis and Design

School of Computing, Informatics and Decision Systems Engineering
Fulton Schools of Engineering

Arizona State University, Tempe, AZ, USA

Unified Modeling Language: Structural Diagrams

Diagrams	Purpose
Class	specify a set of classes, interfaces, and their relationships
Object	specify a set of objects and their relationships
Component	specify a set of components and their relationships – a component is defined as a physical, replaceable part of a system that packages implementation and conforms to and provides the realization of a set of interfaces
Deployment	specify a set of nodes and their relationships – a node is defined as a run-time physical object that represents a computational resource, which generally has at least a memory and often processing capability

Unified Modeling Language

- Unified Modeling Language (UML) is a graphical language aimed at
  - specifying
  - constructing
  - visualizing and
  - documenting

  ← syntax and semantics

  the artifacts of a software intensive system

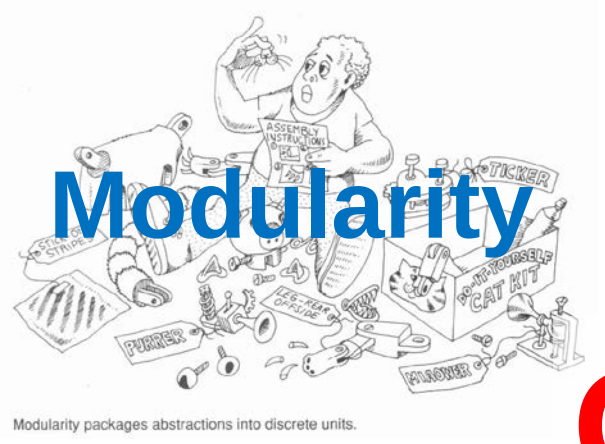
- UML is a standard under the sponsorship of the Object Management Group (OMG)

- UML consists of the best practices in object-oriented modeling

… , UML is based on the fundamentals of the Object Model

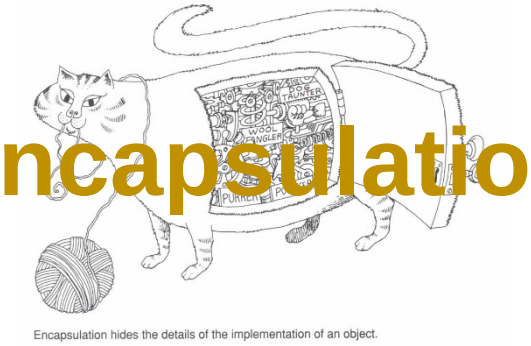
Object Model Basic Elements

Modularity



Modularity packages abstractions into discrete units.

Encapsulation



Encapsulation hides the details of the implementation of an object.

Objects

Abstraction

Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.

Hierarchy

Abstractions form a hierarchy.

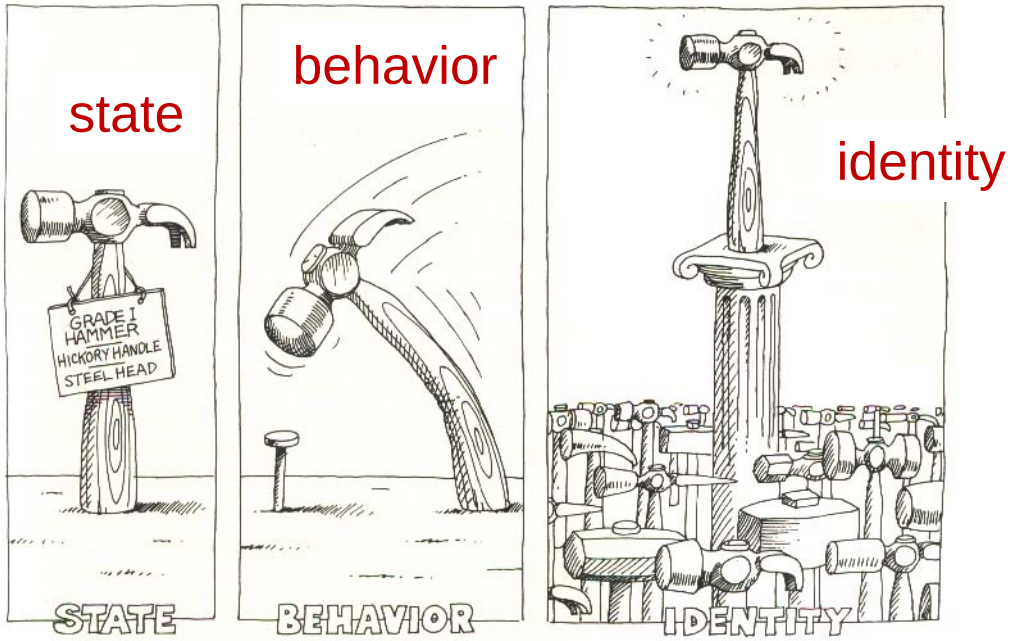
What Are Objects?

- What is an object?
  - A thing we can view or comprehend in the real-world or in the imagined world
  - A thing which can be manipulated either directly or indirectly – possibly by another object and/or by itself (autonomous)
  - A thing that models some part or an aspect of a real-world or imagined entity – static or dynamic
  - A thing that exist in time and space.

an object represents an individual, identifiable item, unit, or entity, either real or abstract, with a **well-defined role** in the **problem and/or solution domain**.

Object Caricature

- The structure and behavior of similar objects are defined in their **common class**



an object has **state**, exhibits some **well-defined behavior**, and has a **unique identity**

Sample Objects

- Hydroponics Farm:
  - Physical motion sensor
  - Model of a motion sensor (e.g., simulation model)
- Mouse: model or physical
  - Mouse Buttons (part)
    - Buttons configuration
      - ❑ right-hand
      - ❑ left-hand

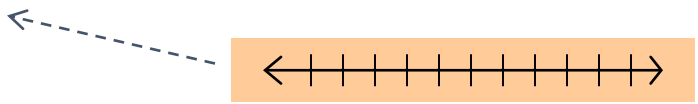
    button Actions

    - Files and Folders
      - ❑ single left click to open a folder, double left click to open a file
    - Double-click speed
      - ❑ slow to fast
  - Mouse Motion (aspect)
    - Speed
      - ❑ slow to fast (discrete choices) – how fast mouse pointer moves

    continuous

    - Snap to default
      - ❑ move pointer to default buttons in dialogue boxes

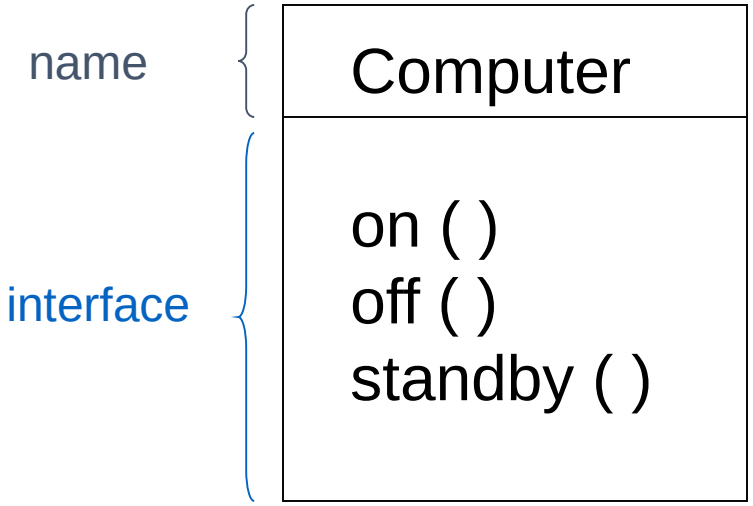
    

    discrete

Types of Objects

- Objects *may or may not have crisp boundaries* and *may or may not exist independent of other objects*
  - bicycles, books, … (crisp boundaries, exist independently of other objects)
  - line between two intersecting shapes (crisp boundary, but cannot exist independent of other objects)
  - crowds of people (fuzzy boundaries, may or may not exist independent of other objects)
- Why should we be concerned about the type of an object's boundary?

An Object from OO Programming Viewpoint




```
class Computer {
   …

   public on ( ) {
      …
   }
   public off ( ) {
      …
   }
   …
}
```

```
Computer myComputer = new Computer( );

void myComputer.standby( );
```

State of an Object

Consider a coin operating vending machine with choices for two kinds of beverages. The following state variables can represent the state (condition) of a vending machine

- # of quarters, # of dimes, # of nickels
  - Type: integer; Values: $\mathbb{N}$ (natural numbers)
- # of beverage brand **A**, # of beverage brand **B**
  - Type: integer; Values: $A = \mathbb{N}$
- amount deposited by a customer
  - Type: integer; Values: $\mathbb{N}$
- total money deposited in coin vault
  - Type: integer; Values: $\mathbb{N}$
- status
  - Type: String, values: ready, make a selection, out of beverage, out of order

note: one out of many possible abstractions!

State of an Object (cont.)

- Objects generally have event- and time-dependent properties. That is,
  - an object's state captures event and temporal aspects; e.g.,
    - coins deposited by all customers and the last customer
    - keep order of events – selection followed by minimum deposit
- An object's state can also capture non-temporal aspects; e.g.,
  - maximum capacity for holding beverages

$\Longrightarrow$ An object's state can be

- **untimed**
- **timed**

the state of an object represents the cumulative results of its behavior at an instance of time – e.g., total # of coins

the state of an object may also represent a time-indexed cumulative behavior – e.g., the last 10 beverages sold

State of an Object (cont.)

- Few attributes of the coin vault (Java)

```
public class CoinVault {

  //Constructor
  public CoinVault(String vaultBrandName) {
    name = vaultBrandName;
  }

//attributes
  String name = “World’s Wonder Water";
  int totalNoQuarCoin, totalNoDimeCoin, totalNoNickCoin;
  int totalNoCoins = 0;
  int customerDeposit = 0;
  String status = “ready";

….
}
```

Behavior of an Object

Behavior is how an object acts and reacts in terms of its state changes and message passing.

Beverage vending machine

- purchase beverage brand **B**
  - insert (deposit) coins – action
  - push beverage A button – action
  - display amount deposited – reaction
  - display “make a selection” – reaction
  - display choice of beverages available – reaction
  - display status – reaction

Behavior of an Object (cont.)

- Few methods for a coin vault (Java)

```
//methods
  public int amountDeposited( ) {
    return customerDeposit;
  }

  /*one coin may be deposited at a time*/

public void coinDeposit(int amt) {
  /*
      if appropriate "preconditions" exist for deposit action then
      e.g., coin slots available for adding certain coin denominations*/

   customerDeposit = customerDeposit + amt;
   if (amt == 25)
     ++totalNoQuarCoin;
   …
   else System.out.println ("invalid amount");
   …
}
```

See note section

Operations of an Object

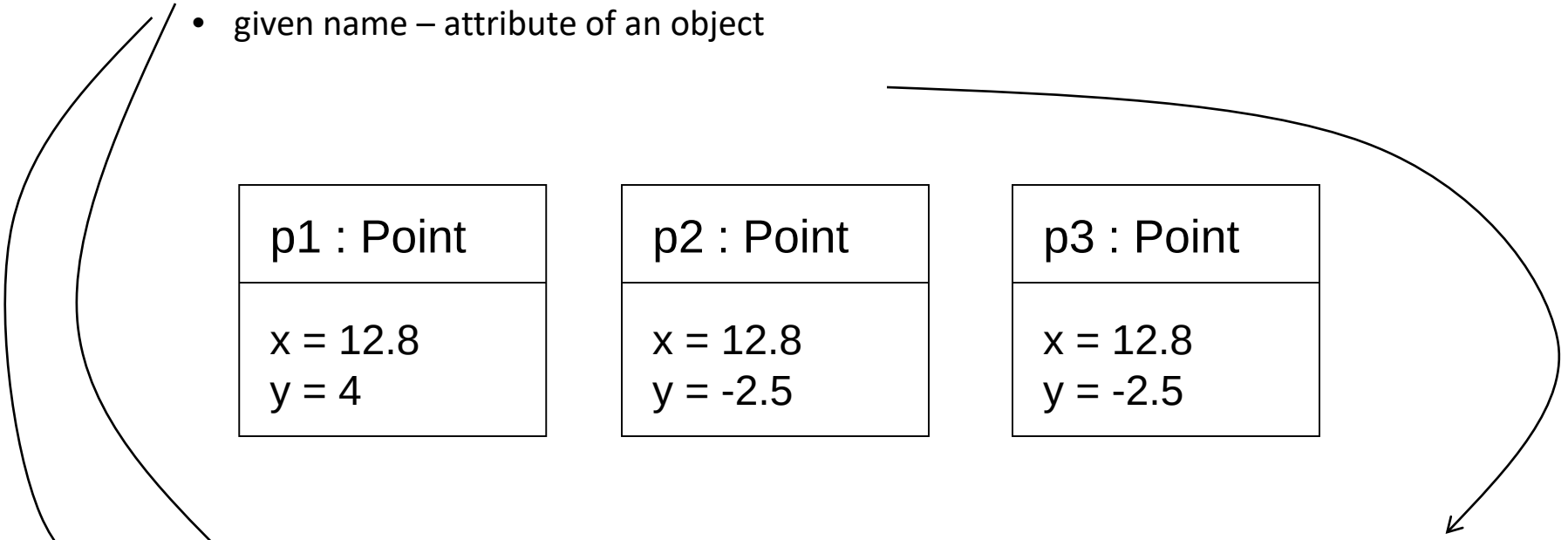
- **Modifier** – an operation that alters the state of an object
- **Selector** – an operation that assesses the state of an object, but does not alter the state
- **Iterator** – an operation that permits all parts of an object to be assessed in some well-defined order
- Constructor – an operation that creates an object and/or initializes its state
- **Destructor** – an operation that frees the state of an object and/or destroys the object itself.

an object's responsibilities:

- maintain knowledge (state of the object)
- support operations (actions it can do, expected to perform)

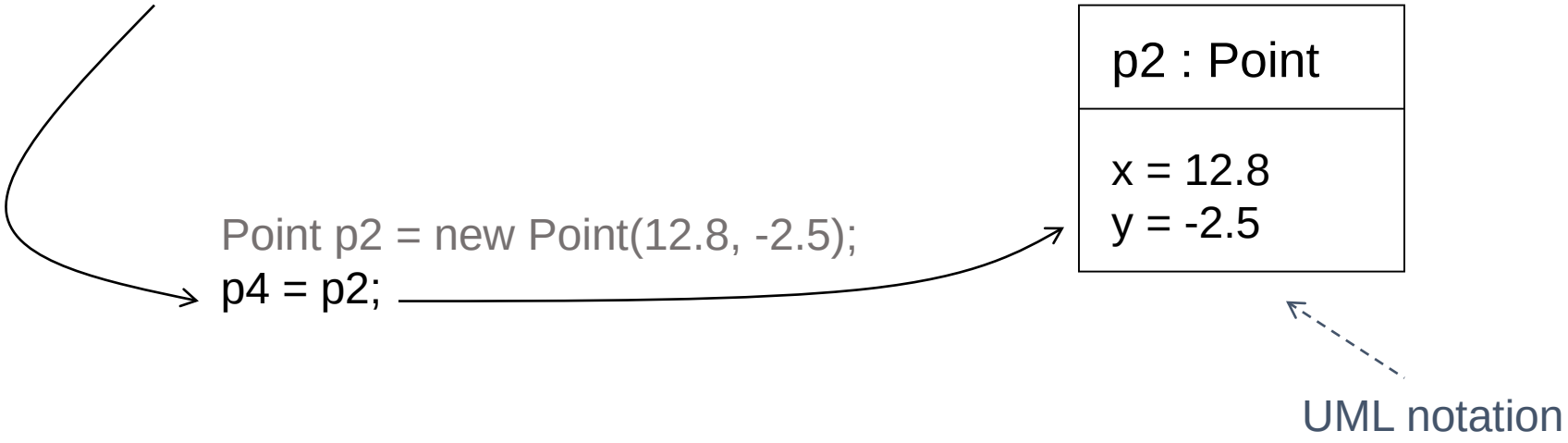
Identity of an Object

- Identity is that property of an object which distinguishes from all other objects
  - name (handle) of an object
  - object itself – resides somewhere in memory such as heap
  - given name – attribute of an object



Identity of an Object (cont.)

- Identity is that property of an object which distinguishes it from all other objects
  - assignment of an object



  - p2 and p4 are two distinct names for the same handle
  - p2 and p4 point to the same object
  - p4 points to object p2 (aliasing – an object is attached to more than one handle)

Identity of an Object (cont.)

- Identity is that property of an object which distinguishes from all other objects
  - copying (cloning) of an object – allows manipulating the copied (cloned) object vs. the original object

- p2 and p4 are distinct objects
- initially p2 and p4 may have the same state
- each object has its own life-cycle (created, lives, and dies)