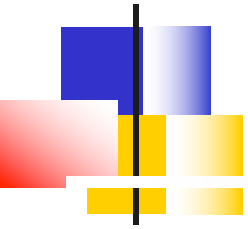
Information Systems Concepts

What Is Object-Orientation?



Roman Kontchakov

Birkbeck, University of London

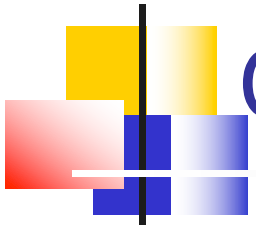
Based on Chapter 4 of Bennett, McRobb and Farmer:
Object Oriented Systems Analysis and Design Using UML, (4th Edition), McGraw Hill, 2010

You'd have to be living face down in a moon crater not to have heard about object-oriented programming.

Tom Swan

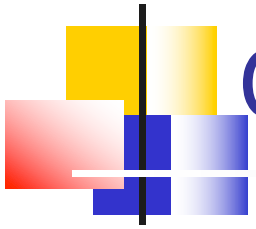
Object-oriented programming is an exceptionally bad idea which could only have originated in California.

Edsger Dijkstra



Outline

- Object-Orientation Concepts
  - Section 4.2 (pp. 91 – 106)
- Object-Orientation Benefits
  - Section 4.3 (pp. 106 – 109)



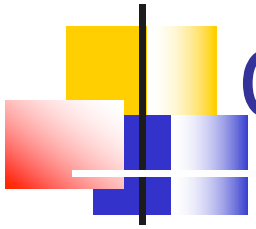
Object

- An object is "an ***abstraction*** of something **in a problem domain**, reflecting the capabilities of the system to keep information about it, interact with it, or both."

  Coad and Yourdon (1990)

- "We define an object as a concept, abstraction, or thing with crisp boundaries and meaning for the problem at hand. Objects serve two purposes: they **promote understanding of the real world** and **provide a practical basis for computer implementation**."

  Rumbaugh et al. (1991)

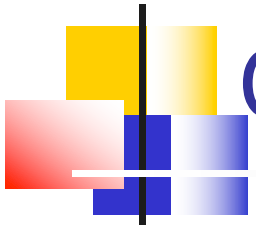


Object

- "Objects have *state*, *behaviour* and *identity*."

  Booch (1994)

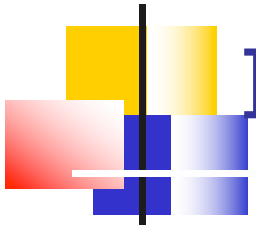
  - Identity (**Who am I?**)
    - each object is unique
  - State (**What do I know?**)
    - the conditions of an object at any moment that affect how it behaves
  - Behaviour (**What can I do?**)
    - the way in which an object responds to messages



Objects

Object	Identity	States	Behaviour
A person	'Hussain Pervez'	Studying Resting Qualified	Speak Walk Read
A shirt	'My favourite button-down white denim shirt'	Pressed Dirty Worn	Shrink Stain Rip
A sale	'Sale no 0015, 15/06/02'	Invoiced Cancelled	Earn loyalty points
A bottle of ketchup	'*This* bottle of ketchup'	Unsold Opened Empty	Spill in transit

A coffee machine object?



Identity != Equality

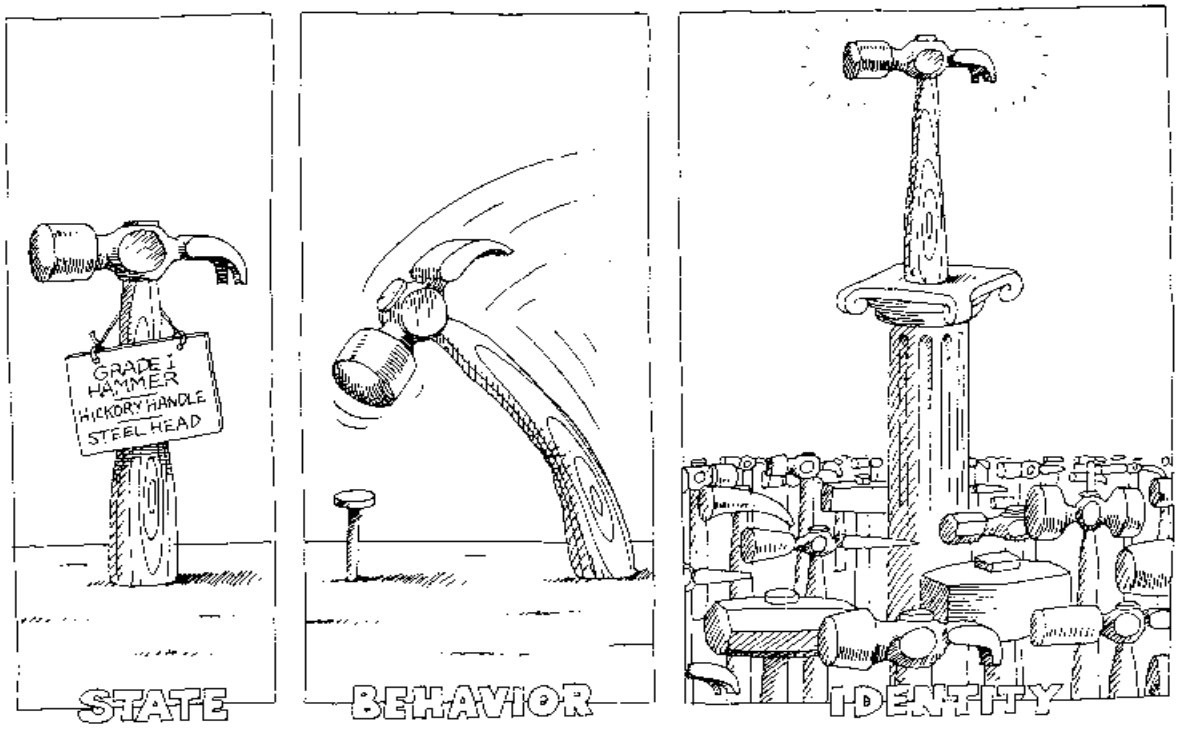
- Different objects must have different identities
- Different objects may have exactly the same state (be equal)
  - e.g., twin brothers, two interchangeable blue pens, etc.

[Java]

`if (obj1 == obj2)` tests **identity**

`if (obj1.equals(obj2))` tests **equality**

Object



Object has

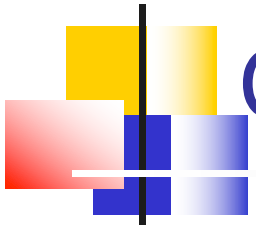
State

Behaviour

Identity

(equal ≠ identical)

BOOCH, G. (1994): *Object Oriented Analysis and Design with Applications*, 2nd ed, The Benjamin/Cummings Publ



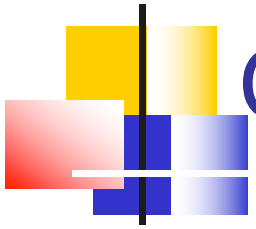
Class

- A class is "***a set of objects*** that share the **same specifications** of features (attributes, operations, links), constraints (e.g. when and whether an object can be instantiated) and semantics"

  OMG (2004)

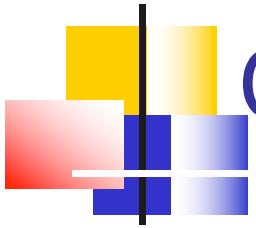
- Moreover, "The purpose of a class is to specify a classification of objects and to specify the features that characterize the **structure** and **behaviour** of those objects"

  OMG (2004)



Class

- An object = An instance of some class
  - Every object must be an instance of some class
- A class = A set of objects that share the same
  - structure
    - what information it holds
    - what links it has to other objects
  - behaviour
    - what things it can do



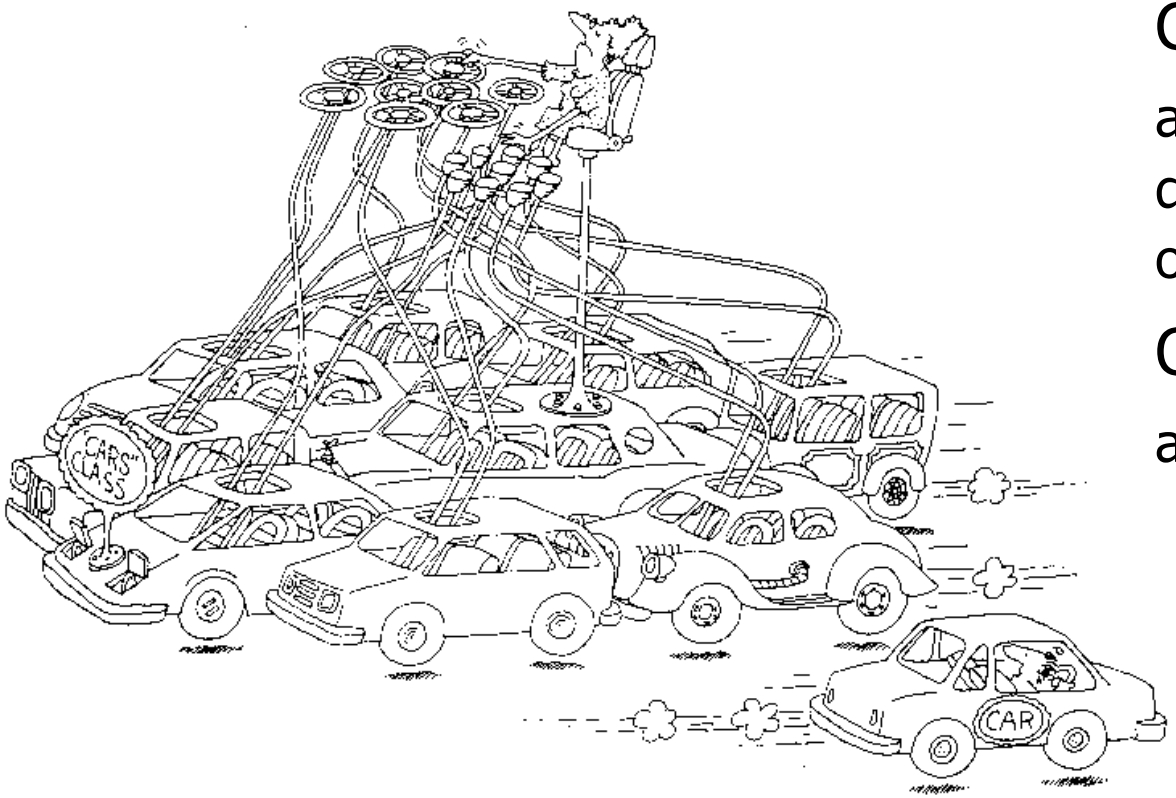
Objects and Classes: terminology

	Object	**Class**	*In C++*	*In Java*
	Identity			
Data	State (values)	**Structure Specification**	Member Variables	Fields (instance variables)
Operations	Behaviour	**Behaviour Specification**	Member Functions	Methods

Ways of thinking about a class

- A **factory** manufacturing objects according to a blueprint
- A **set** that specifies what features its member objects have
- A **template** that allows us to produce any number of objects of a given shape

Class



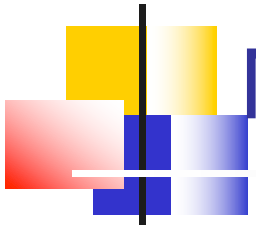
BOOCH, G. (1994): *Object Oriented Analysis and Design with Applications*, 2nd ed, The Benjamin/Cummings Publ

Class

an abstraction (generic description) for a set of objects

Object

an instance of a class

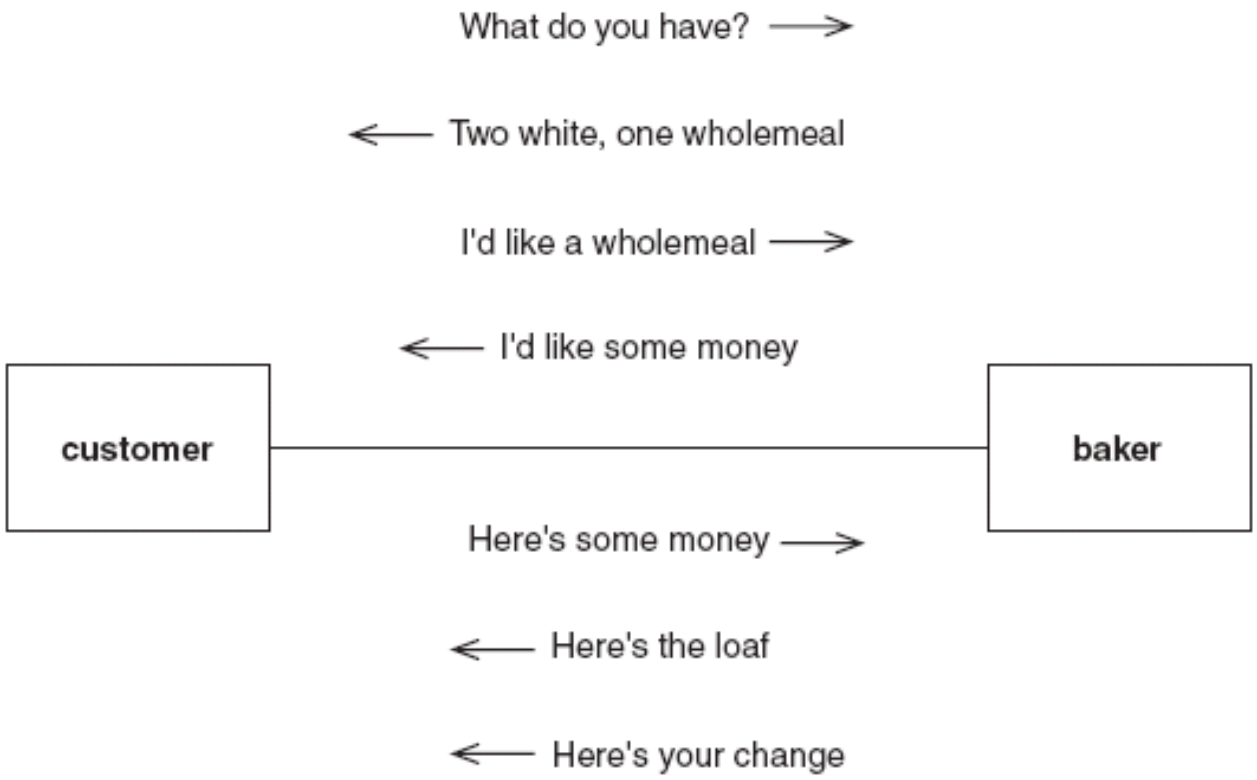


Message Passing

- Objects collaborate to fulfil some system function, and they communicate by sending each other messages:
  - A *question* message asks an object for some information
    - How much is the balance?
  - A *command* message tells an object to do something
    - Withdraw 100 pounds

Message Passing: Example

- Buying a loaf of bread:



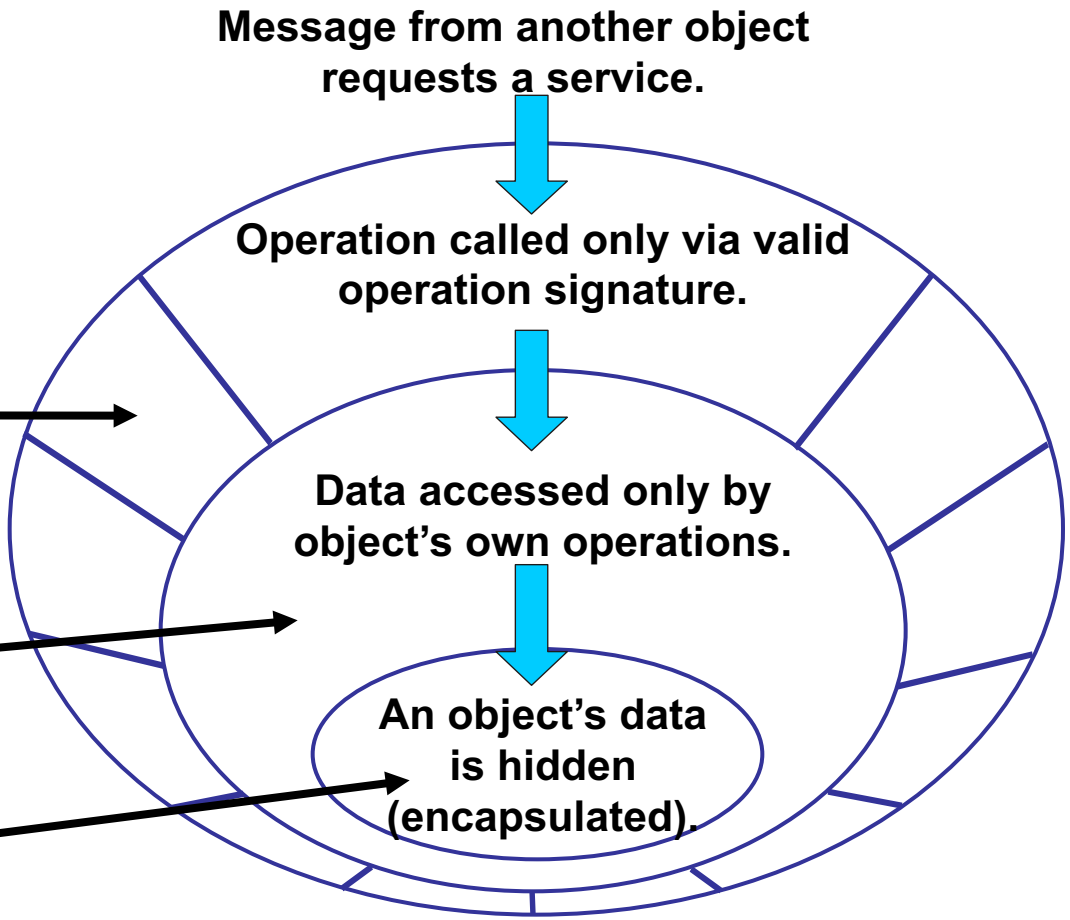
Encapsulation

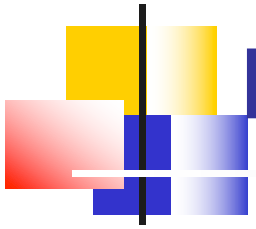
'Layers of an onion' model of an object

An outer layer of operation signatures…

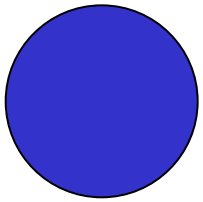
…gives access to middle layer of operations…

…which can access inner core of data





Encapsulation



Consider an object representing a circle.

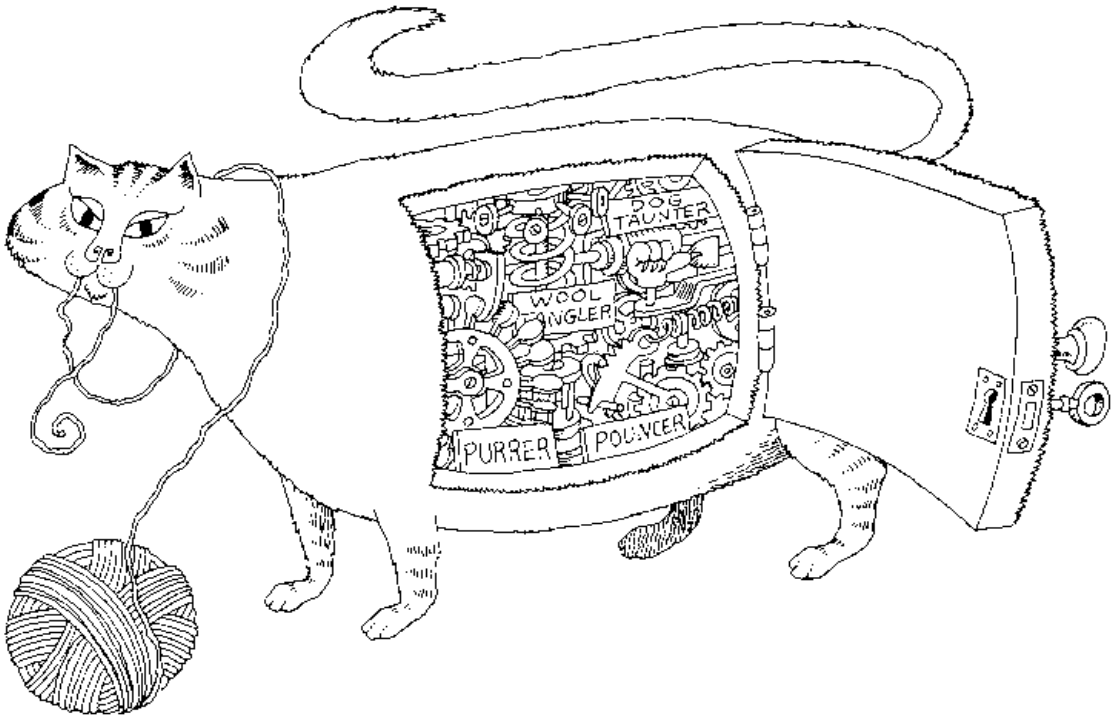
A circle would be likely to have operations allowing us to discover its radius, diameter, area and perimeter. We could store any one of the four attributes and calculate the other three on demand. Let's say we choose to store the ***diameter***.

Without encapsulation, any programmer who was allowed to access the diameter might do so, rather than going via the **getDiameter** operation.

If, for a later version of our software, we decided that we wanted to store the ***radius*** instead, we would have to find all the pieces of code in the system that used direct access to the diameter, so that we could correct them (and we might introduce faults along the way).

With encapsulation, there is no problem.

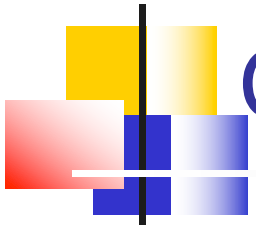
Encapsulation



Object data is hidden

Operations **encapsulate** manipulation of the data

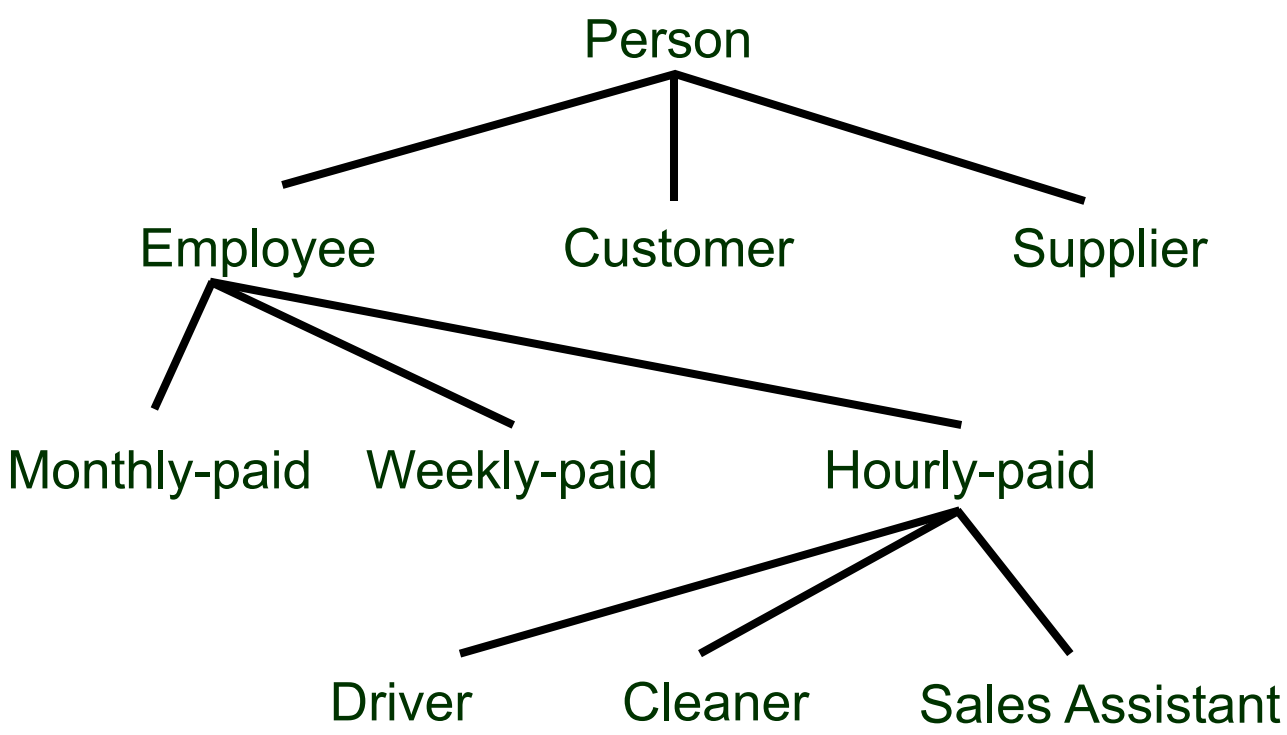
BOOCH, G. (1994): *Object Oriented Analysis and Design with Applications*, 2nd ed, The Benjamin/Cummings Publ



Generalization / Specialization

- Classification is hierarchical in nature
  - A person may be an employee, a customer or a supplier
  - An employee may be paid monthly, weekly or hourly
  - An hourly-paid employee may be a driver, a cleaner or a sales assistant.
- Every instance of the specific class **(subclass)** is also an instance of the more general class **(superclass)**
- A subclass **is a** (kind of) its superclass

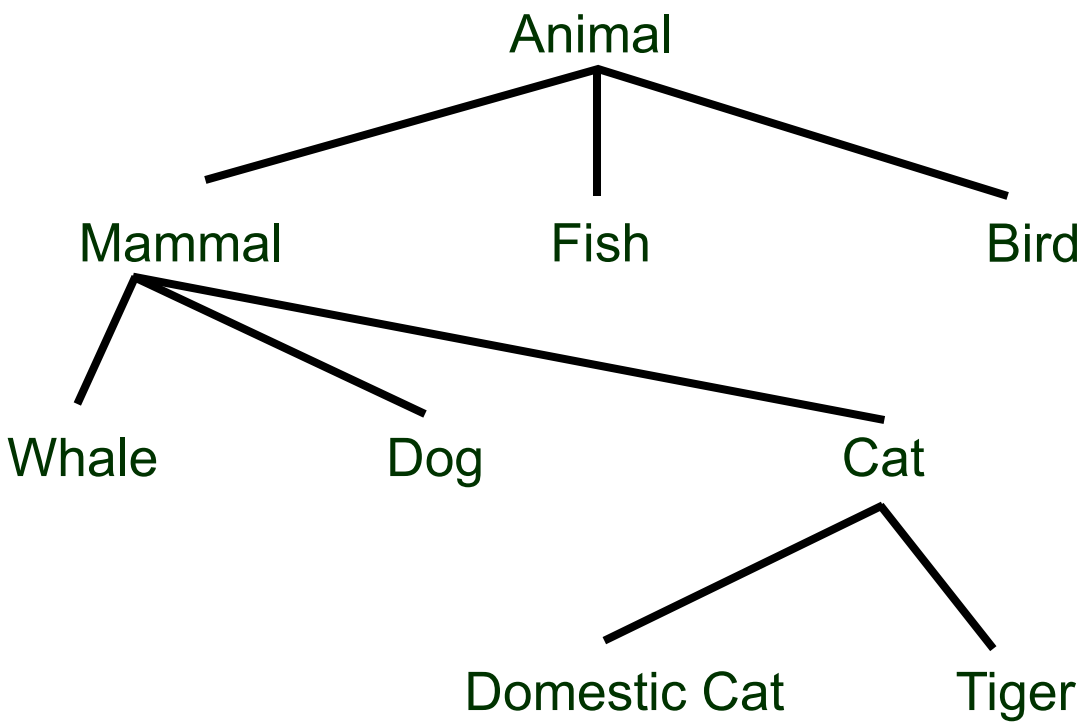
Generalization / Specialization



More general
(superclasses)

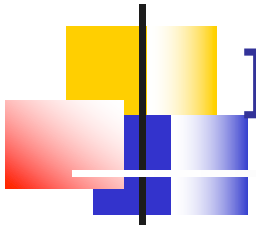
More specific
(subclasses)

Taxonomies



More general
(superclasses)

More specific
(subclasses)

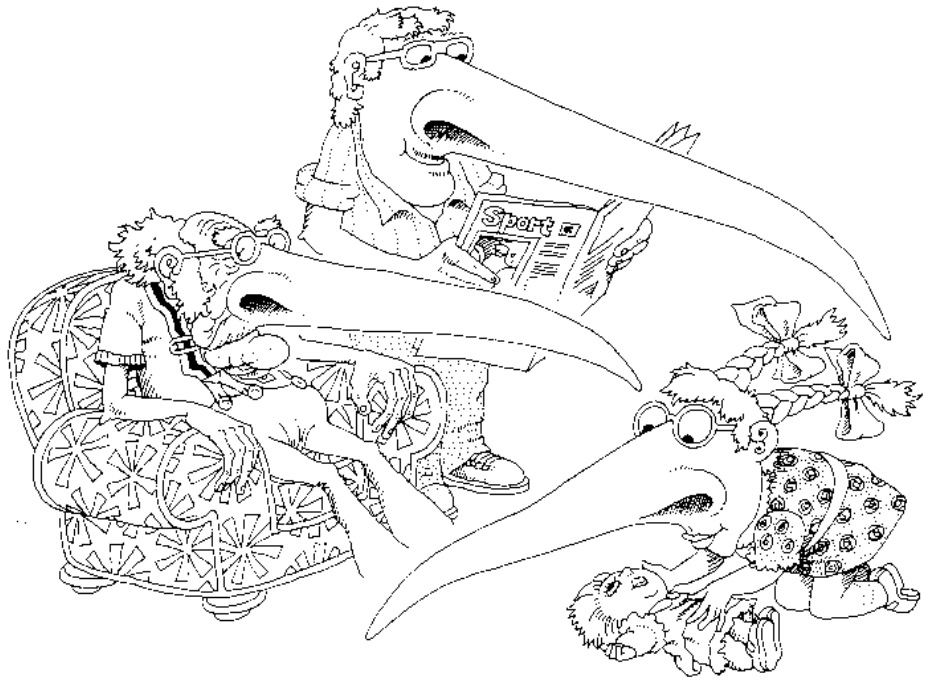


Inheritance

- A subclass always inherits **all** the characteristics (data structure and behaviour) of **all** its superclasses
- The definition of a subclass should always include at least one detail not derived from any of its superclasses

Generalization

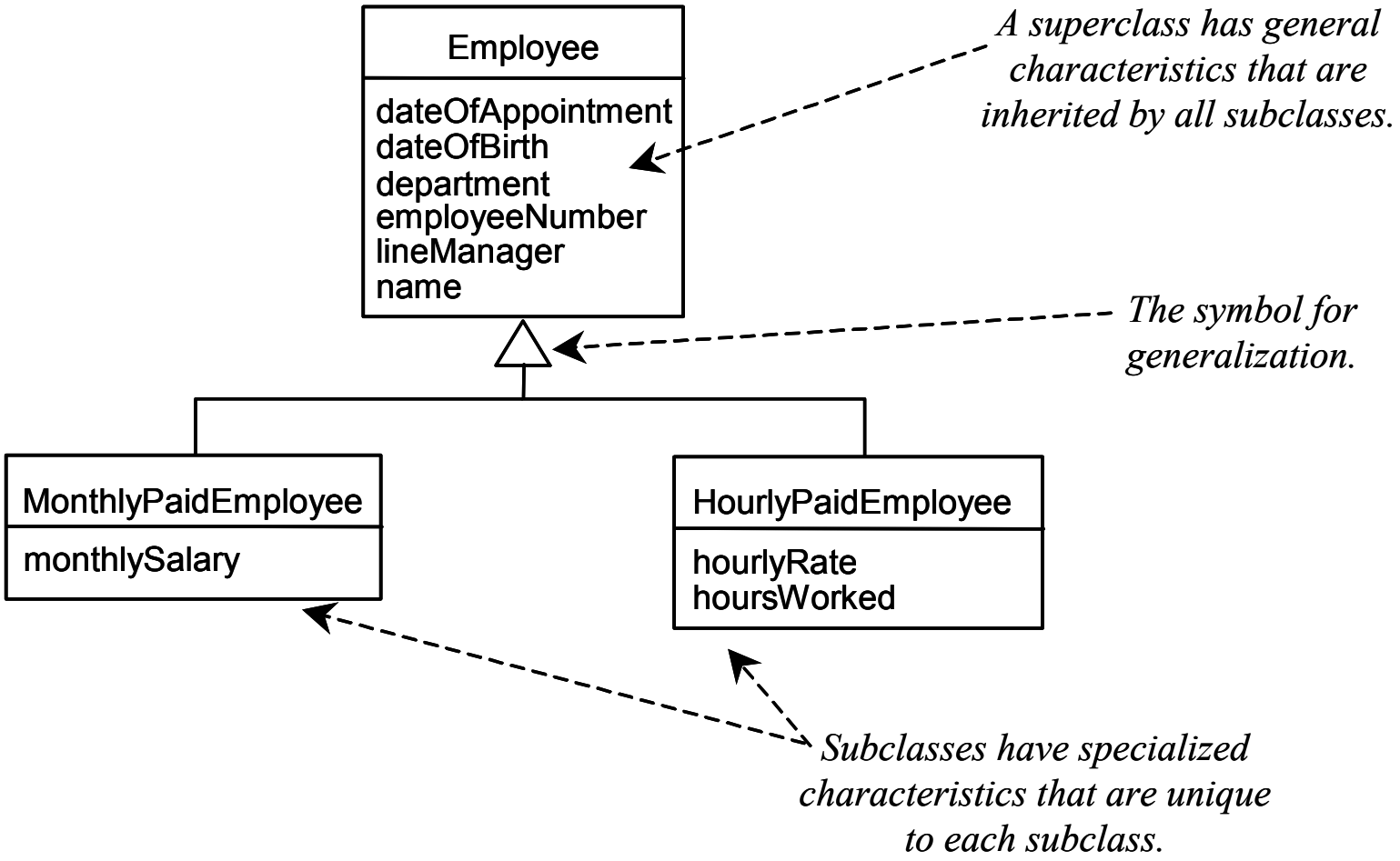
A subclass inherits the structure and behavior of its superclass



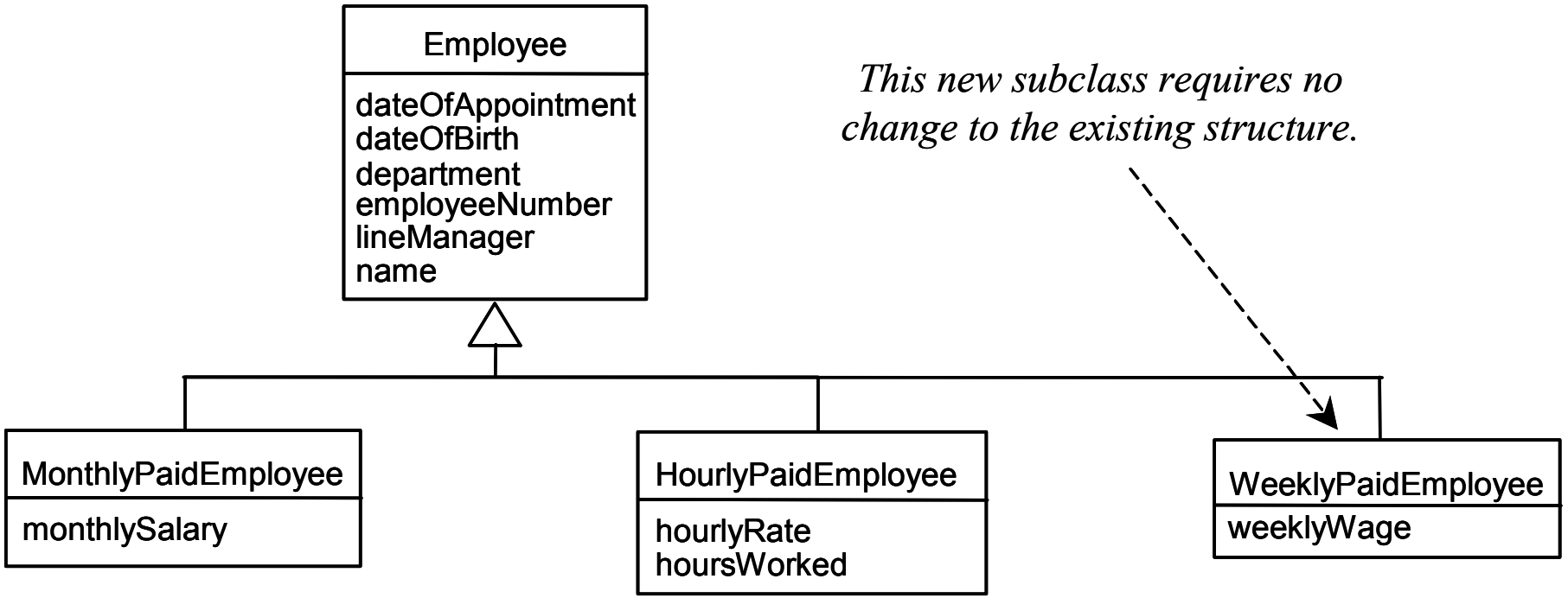
Not a good visualization of generalization, because subclasses inherit types, not values (a nose not a long nose)!

BOOCH, G. (1994): *Object Oriented Analysis and Design with Applications*, 2nd ed, The Benjamin/Cummings Publ

Generalization in UML

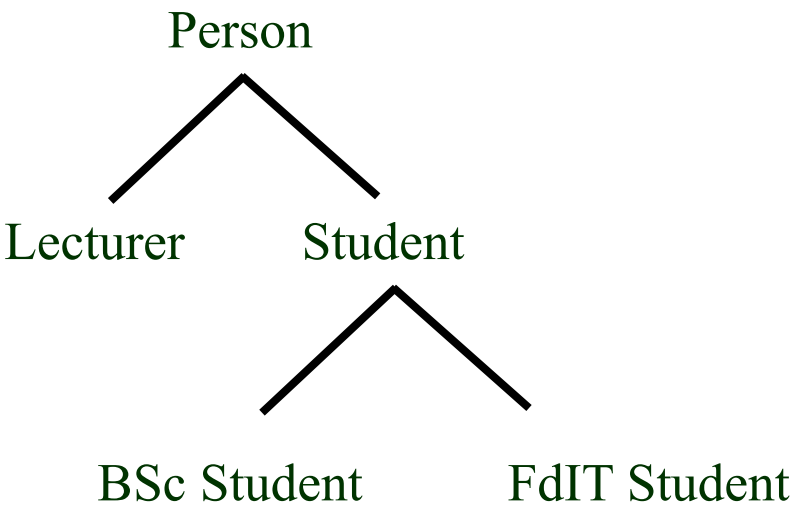


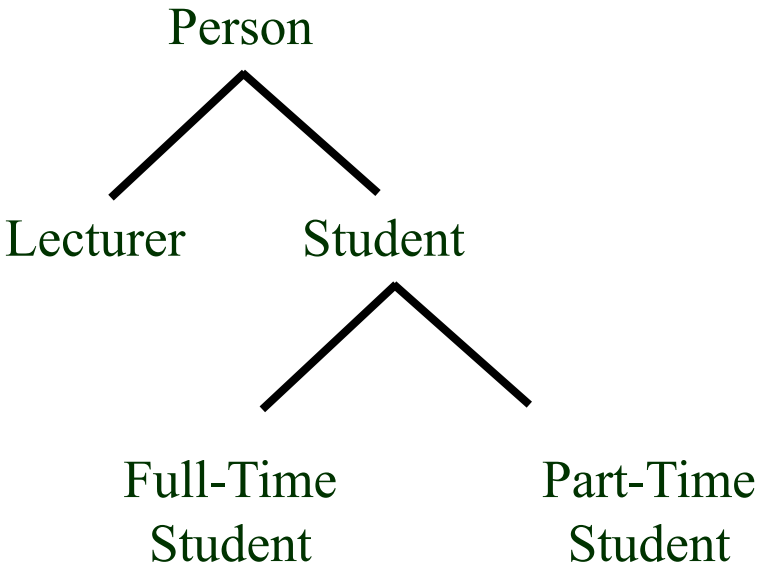
Advantages of using Generalization



Multiple Inheritance

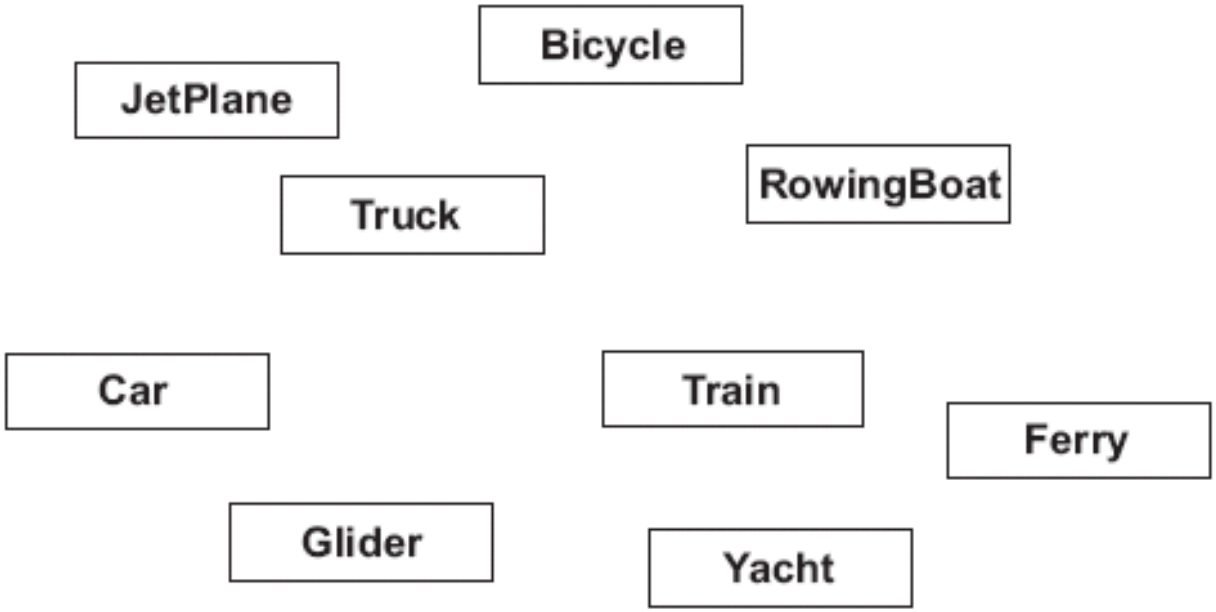
- We may want the 'Part-Time BSc Student' class to be a sub-class of both the 'BSc Student' class and the `Part-Time Student' class.

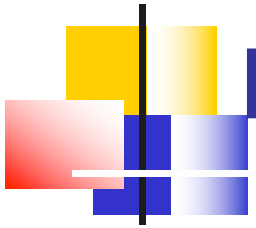




Generalization: Exercise

- How shall we group these classes into a generalization hierarchy?





Polymorphism

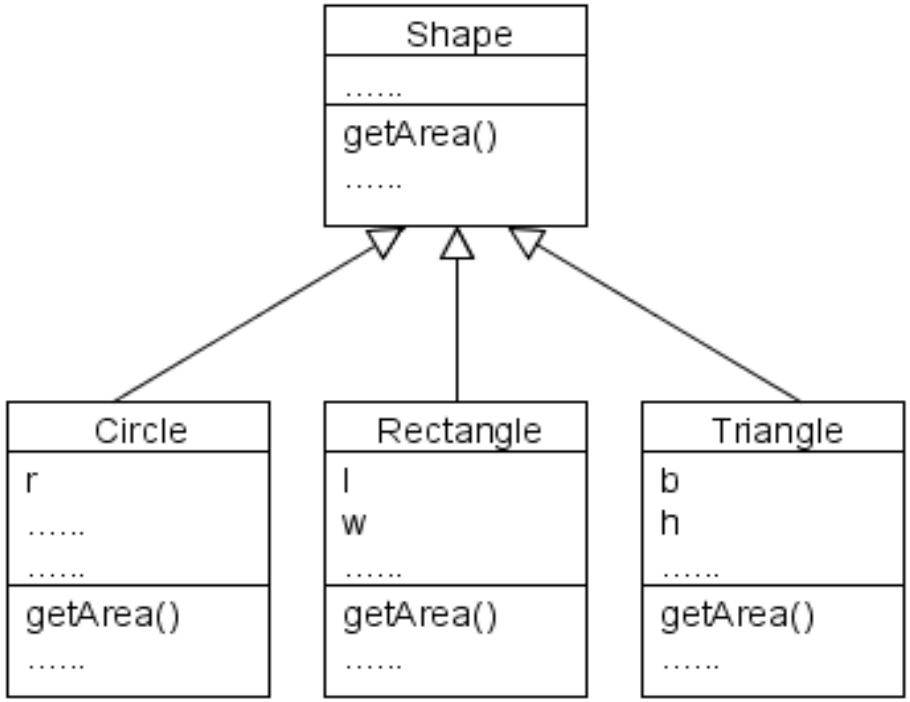
- Polymorphism allows one message to be sent to objects of different classes
- Sending object need not know what kind of object will receive the message
- Each receiving object responds appropriately, i.e., different kinds of objects may respond to the message in different ways

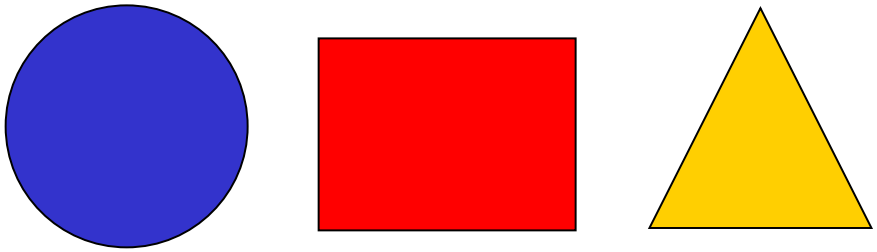
poly morph ic = having many shapes

Polymorphism: Example

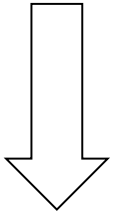
open

Polymorphism: Example





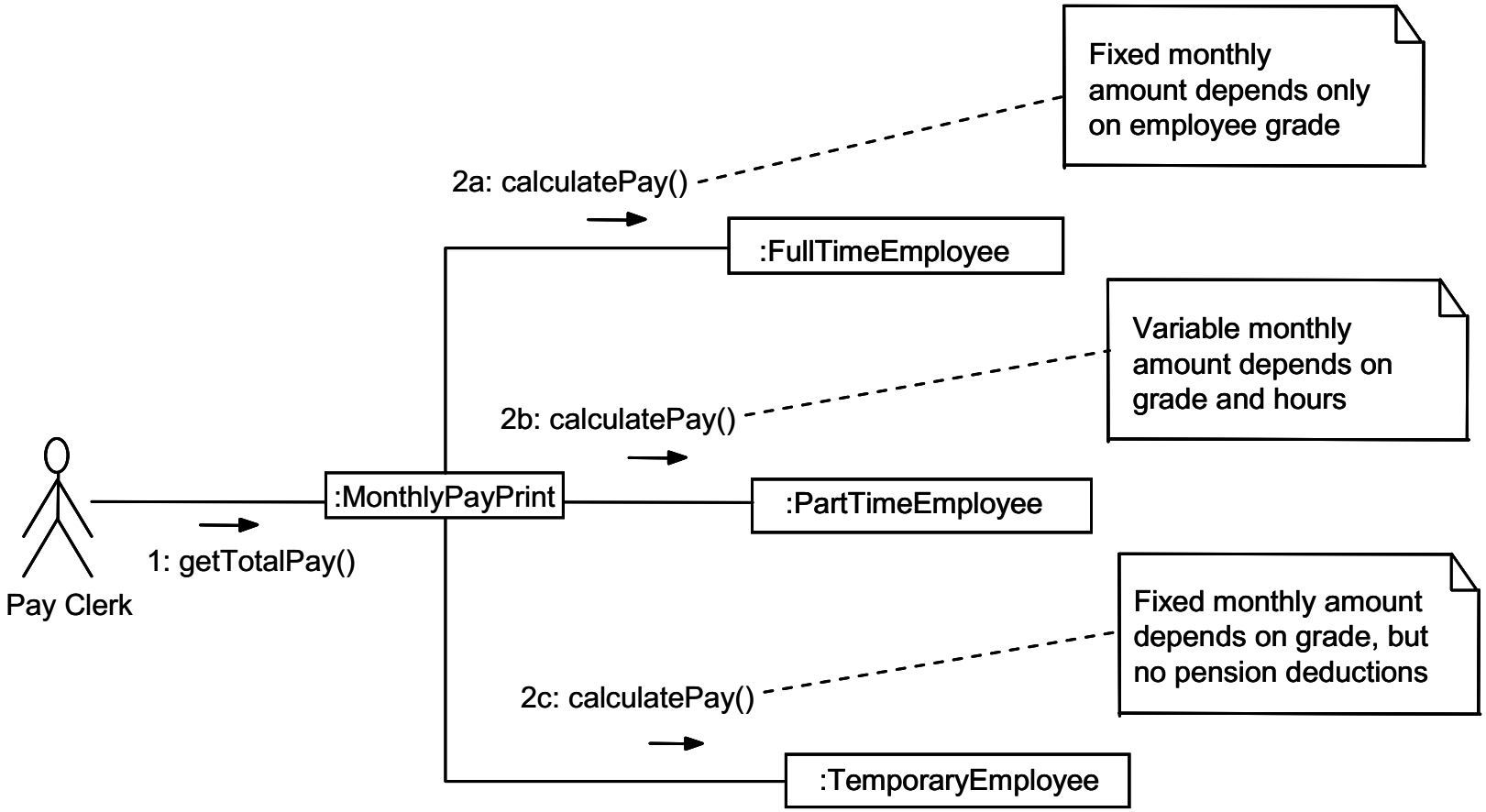
```
if (x is of type 1)
   a = getCircleArea(x.r);
else if (x is of type 2)
   a = getRectangleArea(x.l, x.w);
else if (x is of type 3)
   a = getTriangleArea(x.b, x.h);
```

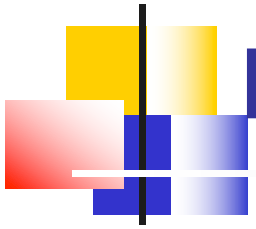


```
a = x.getArea();
```

Polymorphism: Example

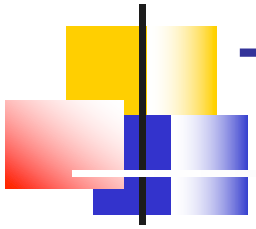
"calculatePay" for different kinds of employees





Benefits of Object-Orientation

- Object-Orientation concepts and techniques improve both software *quality* and software *productivity*
  - Abstraction, Modularity and Reusability
  - Event-Driven Programming and GUI Programming
  - Model Transition and Iterative/Incremental Lifecycle



Take Home Messages

- Object-Orientation Concepts
  - Object and Class
  - Encapsulation
  - Generalization
  - Inheritance
  - Polymorphism
- Object-Orientation Benefits