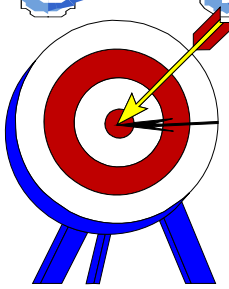
Object-Oriented Systems Development:
Using the Unified Modeling Language

Chapter 2:
Object Basics



Goals

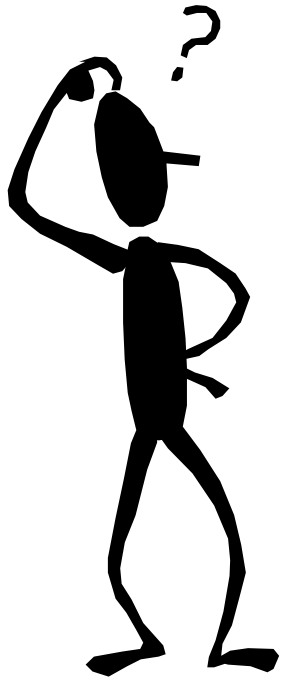
- Define Objects and classes
- Describe objects' methods, attributes and how objects respond to messages,
- Define Polymorphism, Inheritance, data abstraction, encapsulation, and protocol,

Goals (Con't)

- **Describe objects relationships,**
- **Describe object persistence,**
- **Understand meta-classes.**

What is an object?

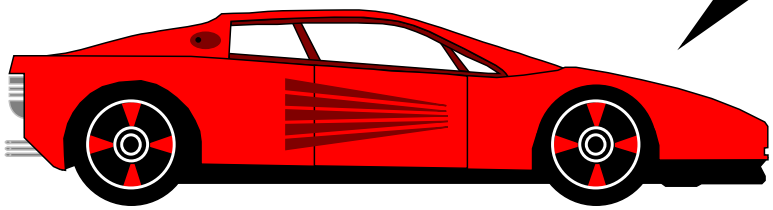
- **The term object was first formally utilized in the Simula language to simulate some aspect of reality.**
- **An object is an entity.**
  - **It knows things (has attributes)**
  - **It does things (provides services or has methods)**



It Knows things (attributes)

Attributes (Con't)

I am a Car.
I know my color, manufacturer, cost, owner and model.

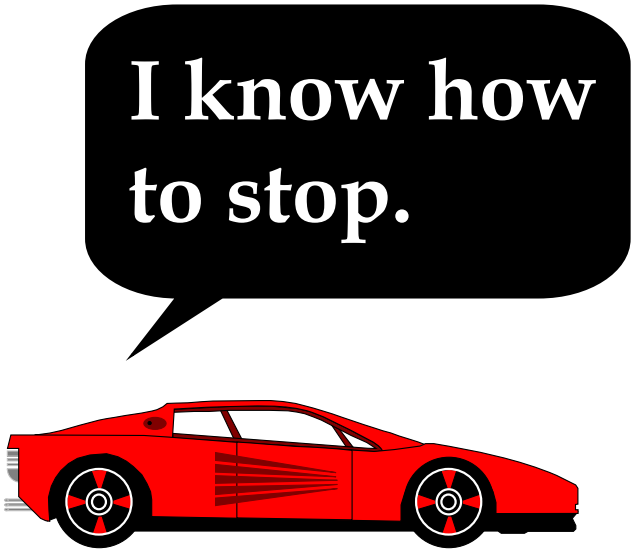


Attributes (Con't)

I am a Fish.
I know my date of arrival and expiration.

It does things (methods)

Methods (Con't)



Methods (Con't)

What is an object? (Con't)

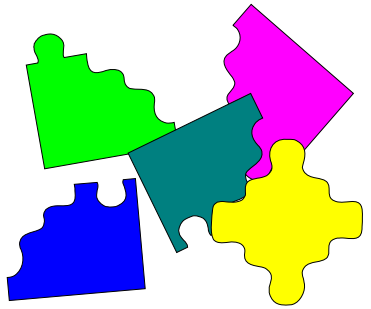
- **Attributes or properties describe object's state (data) and methods define its behavior.**

Object is whatever an application wants to talk about.

- **For example, Parts and assemblies might be objects of bill of material application.**
- **Stocks and bonds might be objects of financial investment applications.**

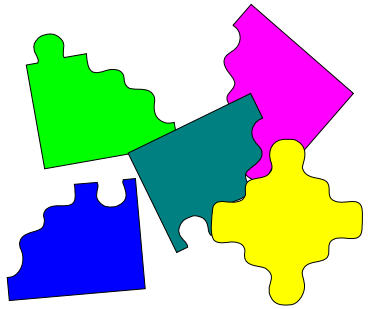
Objects

- **In an object-oriented system, everything is an object: numbers, arrays, records, fields, files, forms, an invoice, etc.**



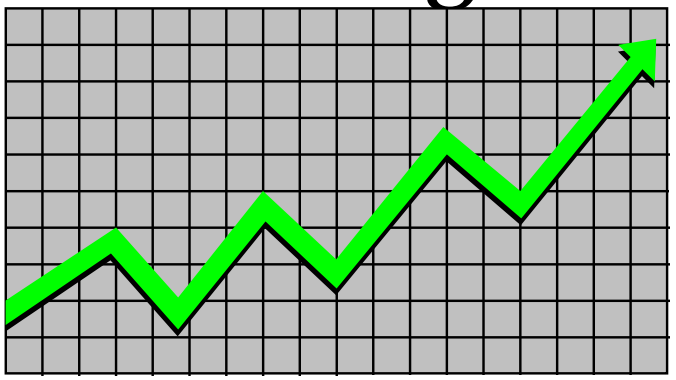
Objects (Con't)

- **An Object is anything, real or abstract, about which we store data and those methods that manipulate the data.**
- **Conceptually, each object is responsible for itself.**



Objects (Con't)

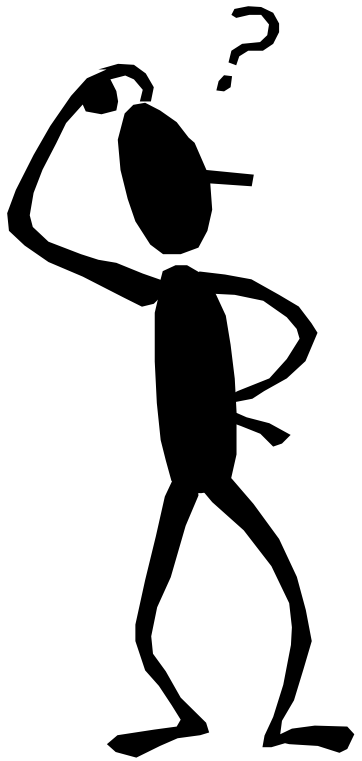
- **A window object is responsible for things like opening, sizing, and closing itself.**
- **A chart object is responsible for things like maintaining its data and labels, and even for drawing itself.**



Two Basic Questions

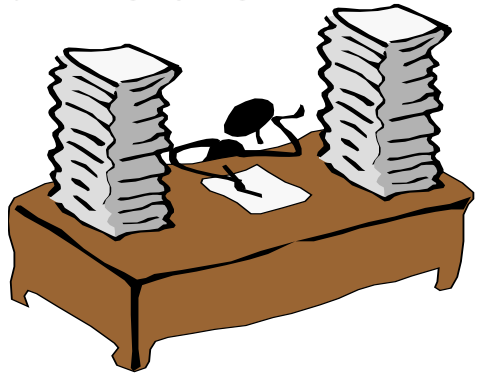
When developing an O-O application, two basic questions always arise.

- **What objects does the application need?**
- **What functionality should those objects have?**



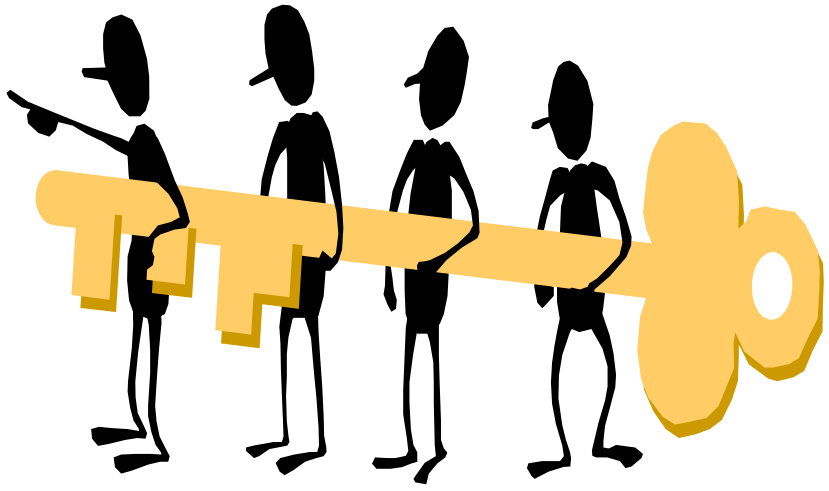
Traditional Approach

- **The traditional approach to software development tends toward writing a lot of code to do all the things that have to be done.**
- **You are the only active entity and the code is just basically a lot of building materials.**



Object-Oriented Approach

- **OO approach is more like creating a lot of helpers that take on an active role, a spirit, that form a community whose interactions become the application.**



Object's Attributes

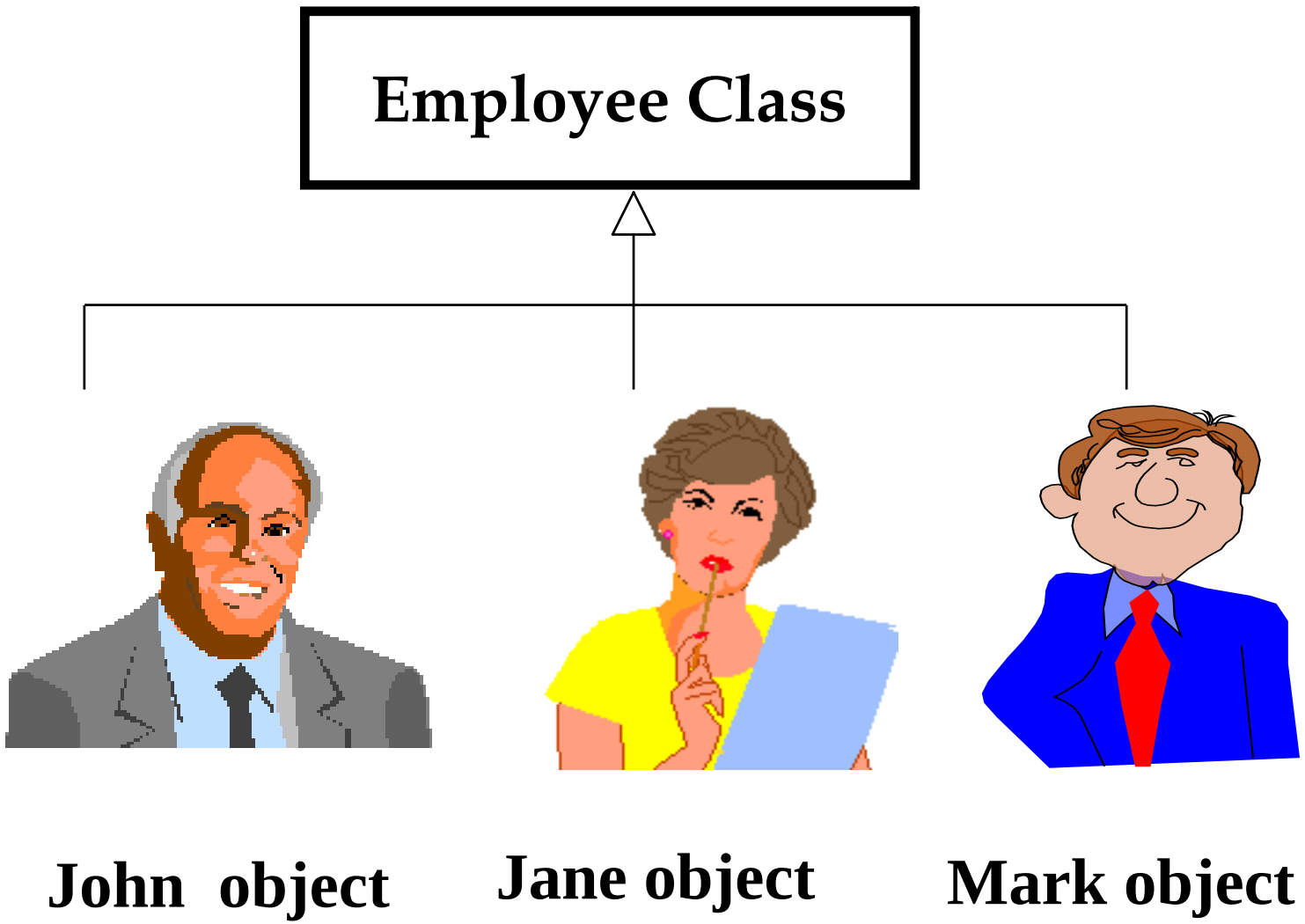
- **Attributes represented by data type.**
- **They describe objects states.**
- **In the Car example the car's attributes are:**
- **color, manufacturer, cost, owner, model, etc.**

Object's Methods

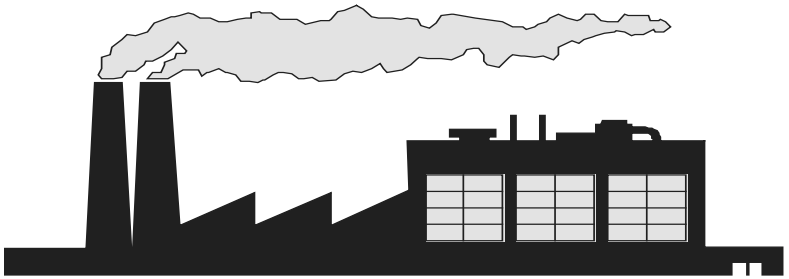
- **Methods define objects behavior and specify the way in which an Object's data are manipulated.**
- **In the Car example the car's methods are:**
- **drive it, lock it, tow it, carry passenger in it.**

Objects are Grouped in Classes

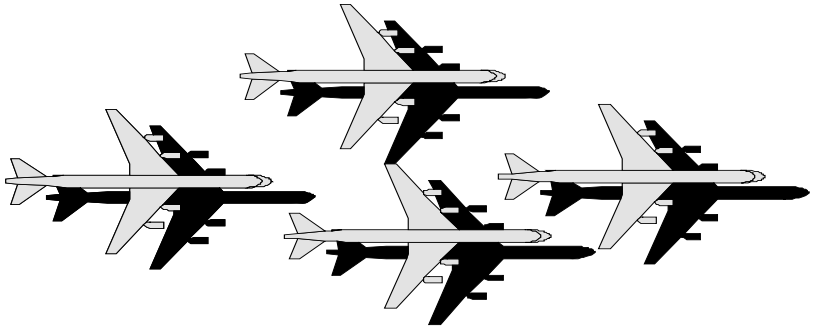
- **The role of a class is to define the attributes and methods (the state and behavior) of its instances.**
- **The class car, for example, defines the property color.**
- **Each individual car (object) will have a value for this property, such as "maroon," "yellow" or "white."**


Employee Class
John object
Jane object
Mark object

A Class is an Object Template, or an Object Factory.



Boeing Factory
(Boeing class)



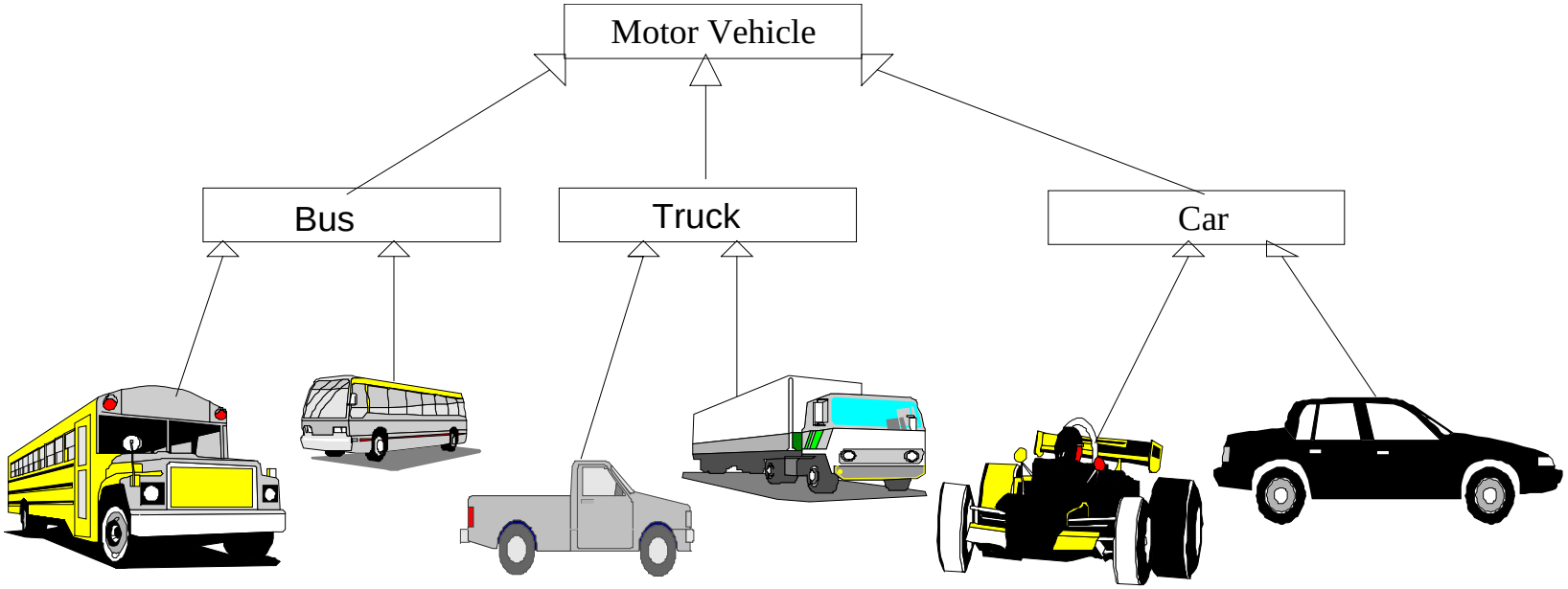
Boeing Airplane Objects
(Boeing instances)

Class Hierarchy

- **An object-oriented system organizes classes into subclass-super hierarchy.**
- **At the top of the hierarchy are the most general classes and at the bottom are the most specific**

Class Hierarchy (Con't)

- **A subclass inherits all of the properties and methods (procedures) defined in its superclass.**



Inheritance (programming by extension)

- **Inheritance is a relationship between classes where one class is the parent class of another (derived) class.**

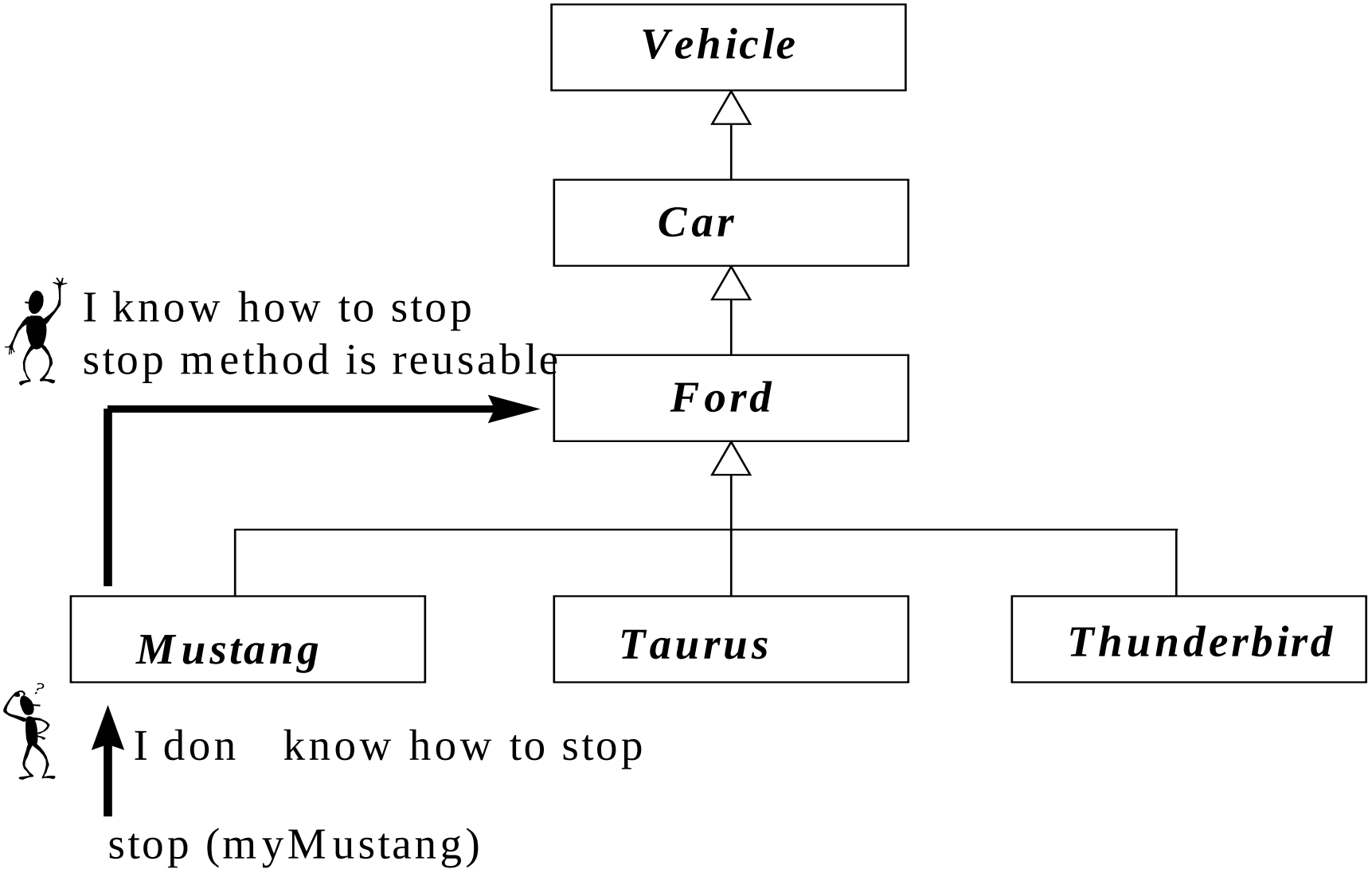
Inheritance (Con't)

- **Inheritance allows classes to share and reuse behaviors and attributes.**

Inheritance (Con't)

- **The real advantage of inheritance is that we can build upon what we already have and,**
- **Reuse what we already have.**

Inheritance (Con't)

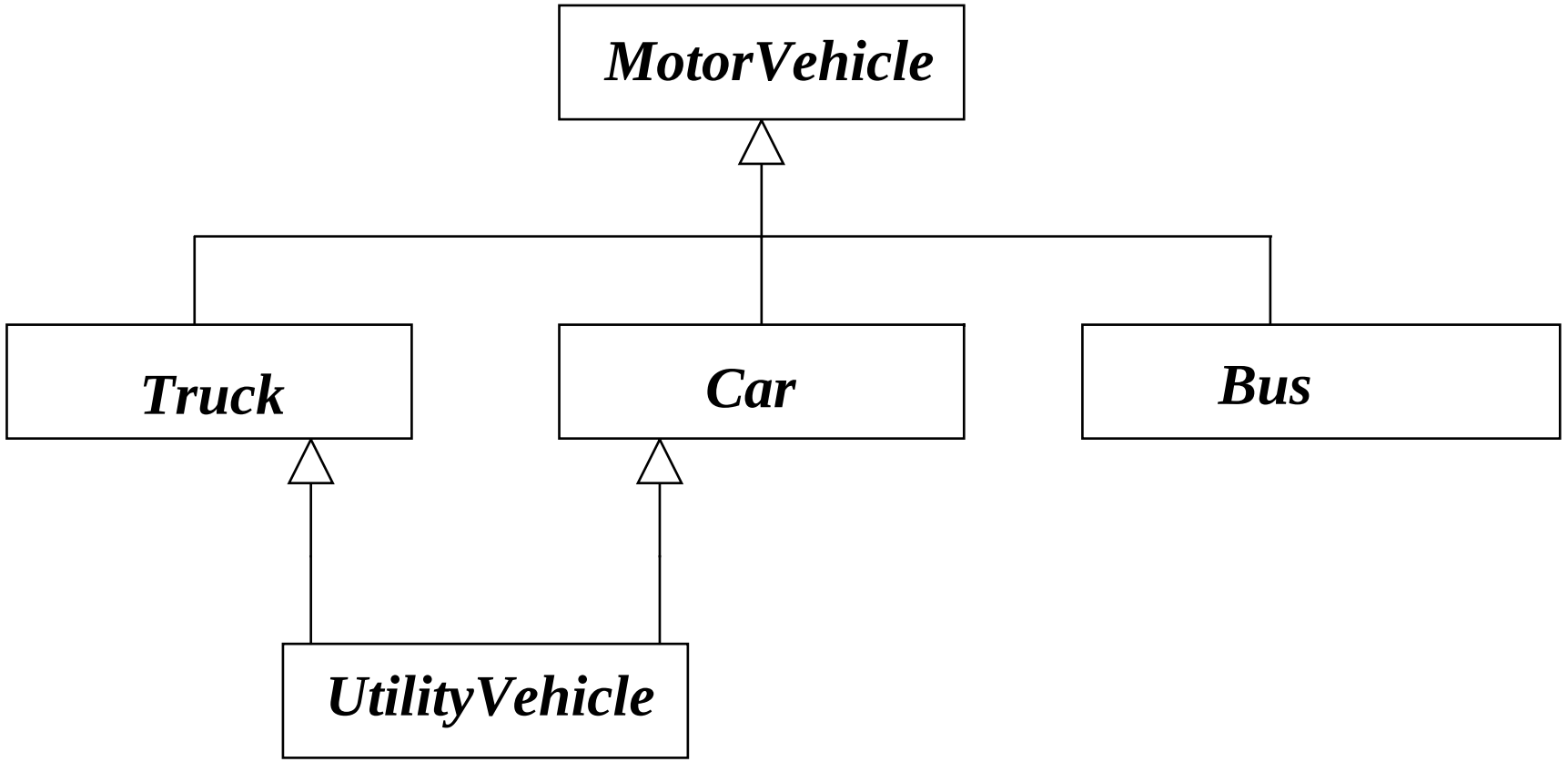


Multiple Inheritance

- **OO systems permit a class to inherit from more than one superclass.**
- **This kind of inheritance is referred to as multiple inheritance.**

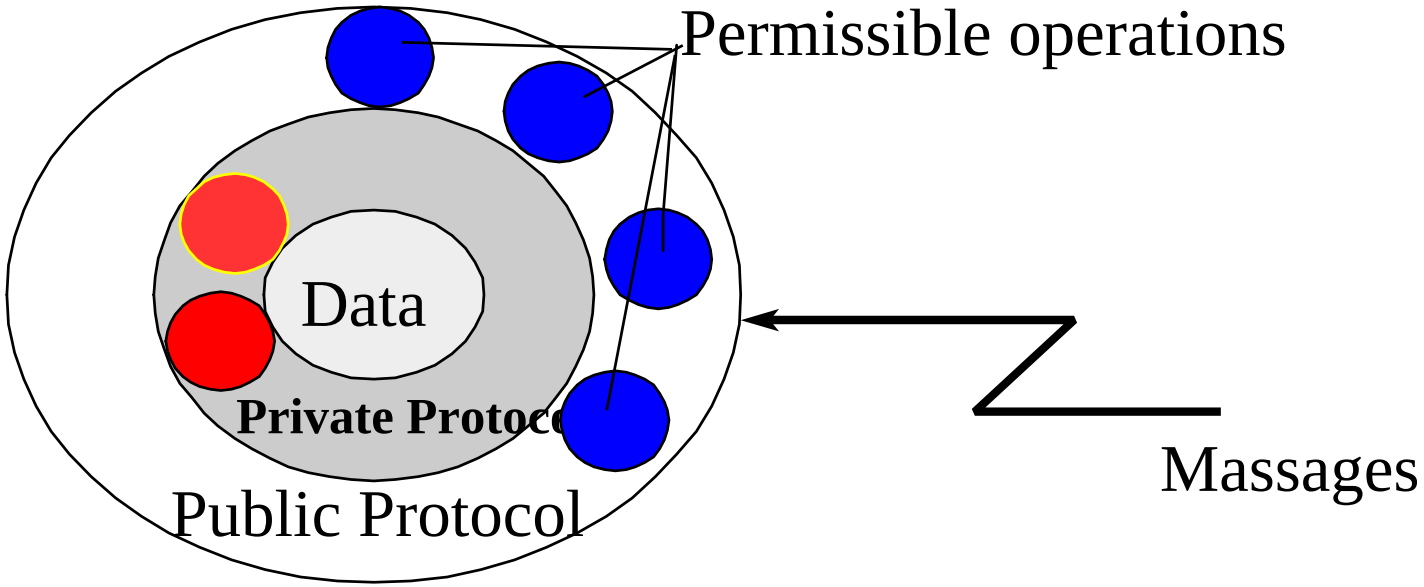
Multiple Inheritance (Con't)

- **For example utility vehicle inherents from Car and Truck classes.**



Encapsulation and Information Hiding

- **Information hiding is a principle of hiding internal data and procedures of an object.**

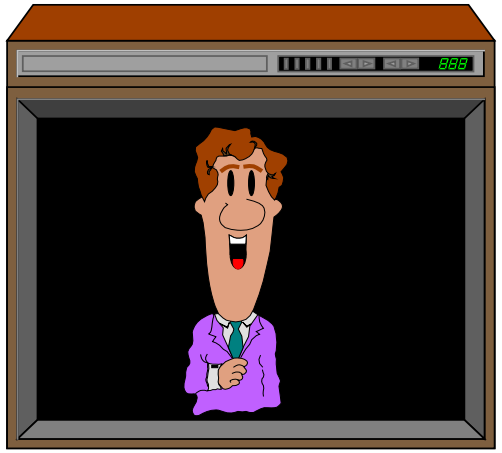


Encapsulation and Information Hiding (Con't)

- **By providing an interface to each object in such a way as to reveal as little as possible about its inner workings.**
- **Encapsulation protects the data from corruption.**

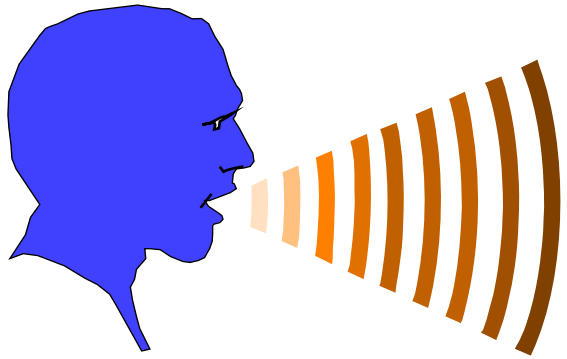
Protocol

- **Protocol is an interface to the object.**
- **TV contains many complex components, but you do not need to know about them to use it.**



Message

- **Objects perform operations in response to messages.**
- **For example, you may communicate with your computer by sending it a message from hand-help controller.**



A Case Study - A Payroll Program

- **Consider a payroll program that processes employee records at a small manufacturing firm. This company has three types of employees:**
- **1. *Managers*: Receive a regular salary.**
- **2. *Office Workers*: Receive an hourly wage and are eligible for overtime after 40 hours.**
- **3. *Production Workers*: Are paid according to a piece rate.**

Structured Approach

```
FOR EVERY EMPLOYEE DO
  BEGIN
   IF employee = manager  THEN
     CALL computeManagerSalary
   IF employee = office worker THEN
          CALL computeOfficeWorkerSalary
  IF employee = production worker
   THEN   CALL
   computeProductionWorkerSalary
 END
```

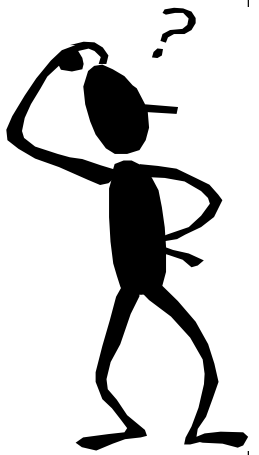
What if we add two new types of employees?

- **Temporary office workers ineligible for overtime,**
- **Junior production workers who receive an hourly wage plus a lower piece rate.**

```
FOR EVERY EMPLOYEE DO
  BEGIN
    IF employee = manager THEN
          CALL computeManagerSalary
    IF employee = office worker THEN
          CALL computeOfficeWorker_salary
    IF employee = production worker THEN
          CALL computeProductionWorker_salary
    IF employee = temporary office worker THEN
          CALL computeTemporaryOfficeWorkerSalary
    IF employee = junior production worker THEN
          CALL computeJuniorProductionWorkerSalary
  END
```

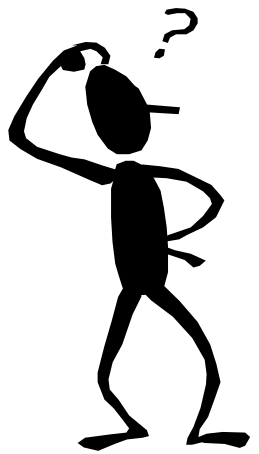
An Object-Oriented Approach

- *What objects does the application need?*
  - **The goal of OO analysis is to identify objects and classes that support the problem domain and system's requirements.**
  - **Some general candidate classes are:**
  - *Persons*
  - *Places*
  - *Things*



What are some of the application's classes?

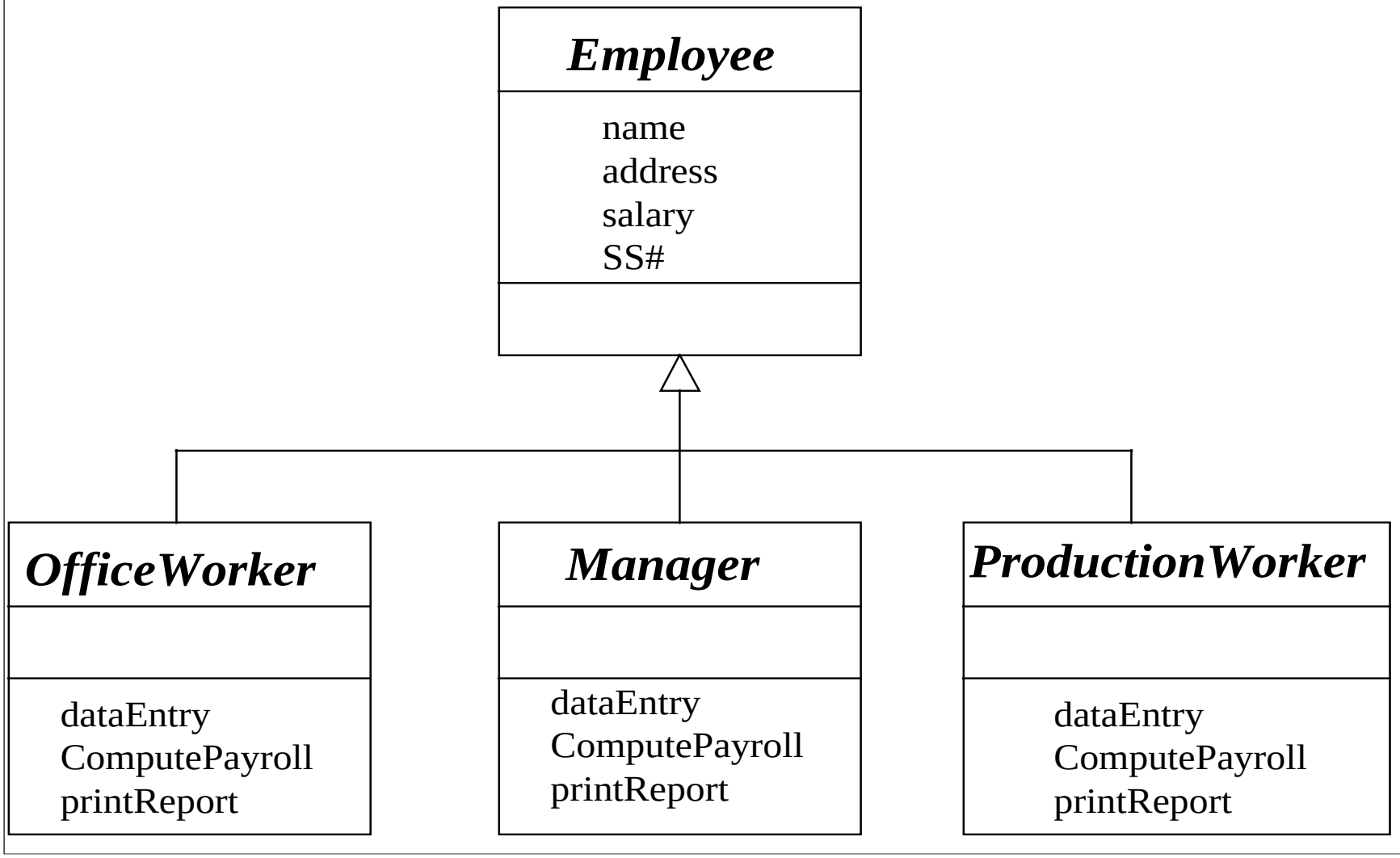
- *Employee*
- *Manager*
- *Office Workers*
- *Production Workers*



Class Hierarchy

- **Identify class hierarchy**
- **Identify commonality among the classes**
- **Draw the general-specific class hierarchy.**

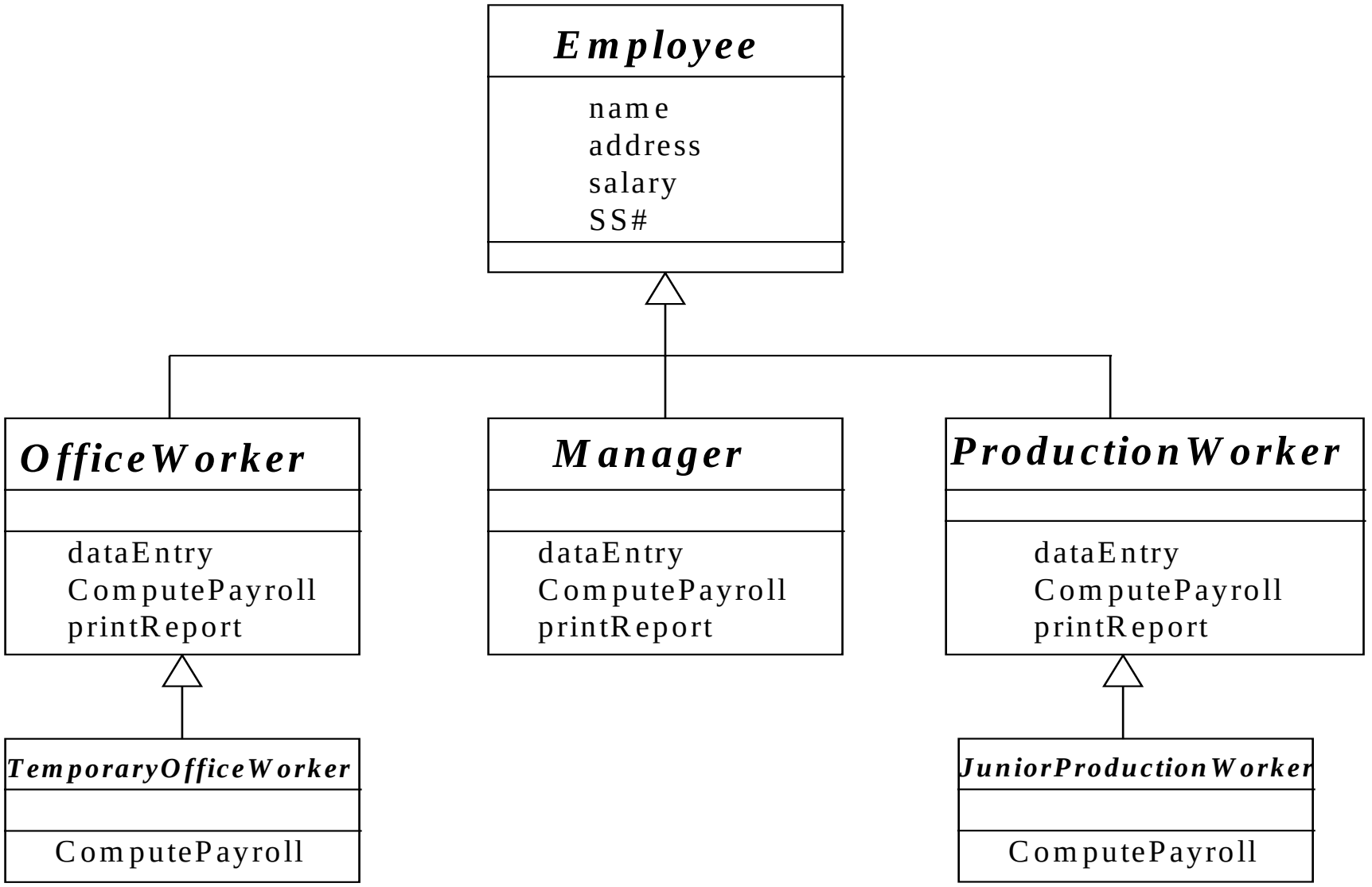
Class Hierarchy (Con't)



OO Approach

```
FOR EVERY EMPLOYEE DO
  BEGIN
    employee computePayroll
END
```

If a new class of employee were



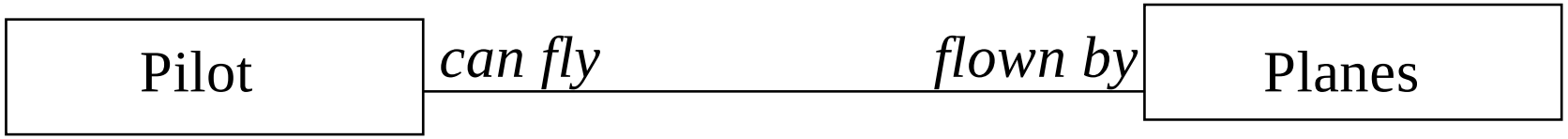
Polymorphism

- **Polymorphism means that the same operation may behave differently on different classes.**
- **Example: *computePayroll***

Associations

- **The concept of association represents relationships between objects and classes.**
- **For example a pilot *can fly* planes.**

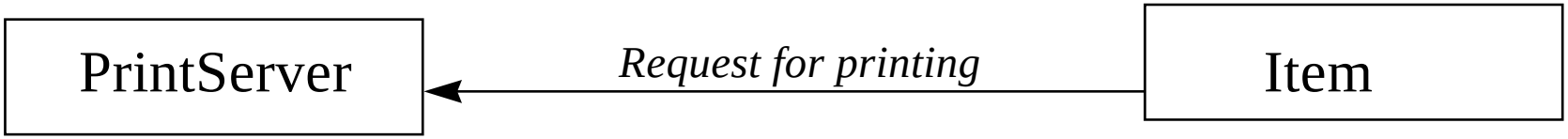
Associations (Con't)



Clients and Servers

- **A special form of association is a client-server relationship.**
- **This relationship can be viewed as one-way interaction: one object (client) requests the service of another object (server).**

Clients and Servers (Con't)



Objects and Persistence

- **Objects have a lifetime.**
- **An object can persist beyond application session boundaries, during which the object is stored in a file or a database, in some file or database form.**

Meta-Classes

- **Everything is an object.**
- **How about a class?**
- **Is a class an object?**
- **Yes, a class is an object! So, if it is an object, it must belong to a class.**
- **Indeed, class belongs to a class called a Meta-Class or a class' class.**

Meta-Classes (Con't)

- **Meta-class used by the compiler. For example, the meta-classes handle messages to classes, such as constructors and "new."**

Summary

- **Rather than treat data and procedures separately, object-oriented programming packages them into "objects."**
- **O-O system provides you with the set of objects that closely reflects the underlying application**

Summary (Con't)

Advantages of object-oriented programming are:

- **The ability to reuse code,**
- **develop more maintainable systems in a shorter amount of time.**

Summary (Con't)

- **more resilient to change, and**
- **more reliable, since they are built from completely tested and debugged classes.**