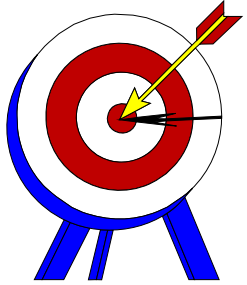


OBJECT-ORIENTED SYSTEMS DEVELOPMENT: USING THE UNIFIED MODELING LANGUAGE



Object Basics

GOALS



- Objects and classes and their differences
- Objects' methods, attributes and how objects respond to messages,
- Class hierarchy inheritance and multiple inheritance
- Polymorphism, Inheritance, data abstraction, encapsulation/information hiding, and protocol,
- Aggregations
- Objects relationships and association,
- Object persistence,
- Static and dynamic binding
- Understand meta-classes.



INTRODUCTION

- Single motivation factor behind object oriented system development is the desire to make software development easier and more natural
 - by raising the level of abstraction to the point where applications can be implemented in the same terms as described by the user.



INTRODUCTION

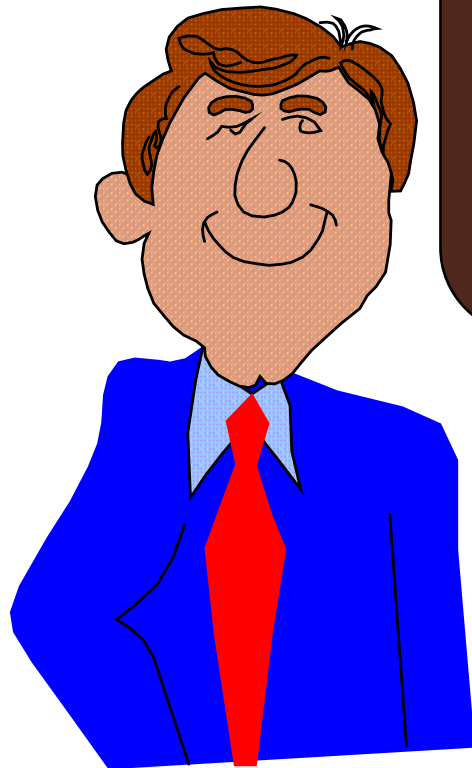
- The fundamental difference between OO systems and their traditional counterparts is the way you approach the problem
- Most traditional methodologies are:
 - Algorithm-centric (think of an algorithm first that can accomplish the task, then build data structures around it)
 - Data-centric (first think how to structure the data, and then build the algorithm around the structure)
- In Object oriented system
 - The algorithm and data structures are packaged together as an object, which has set of attributes or properties.
 - The state of these attributes is reflected in the values stored in the data structures.
 - The object has a collection of procedures or methods- things it can do.

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)

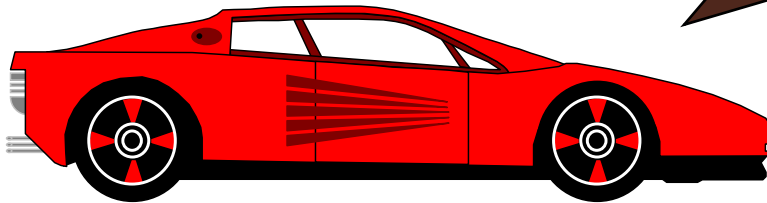


**I am an Employee.
I know my name,
social security number and
my address.**



ATTRIBUTES (CON'T)

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



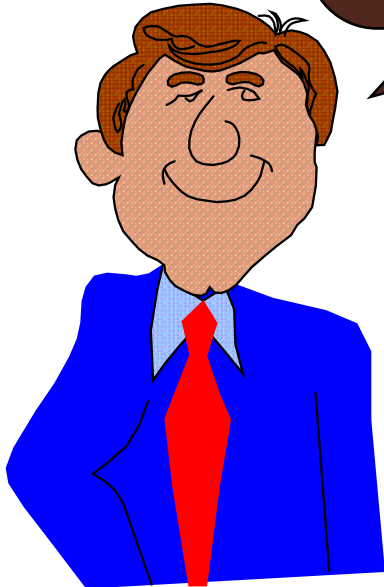
ATTRIBUTES (CON'T)

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



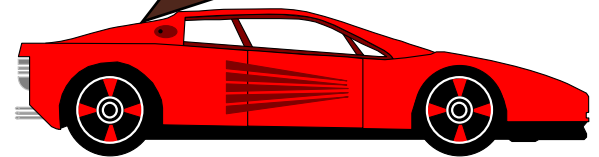
IT DOES THINGS (METHODS)

**I know how to
compute
my payroll.**



METHODS (CON'T)

I know how
to stop.



METHODS (CON'T)

I know how
to cook myself.



WHAT IS AN OBJECT? (CON'T)

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



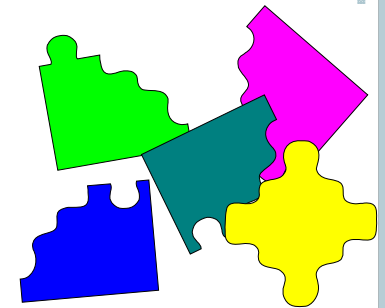
OBJECT IS WHATEVER AN APPLICATION WANTS TO TALK ABOUT

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



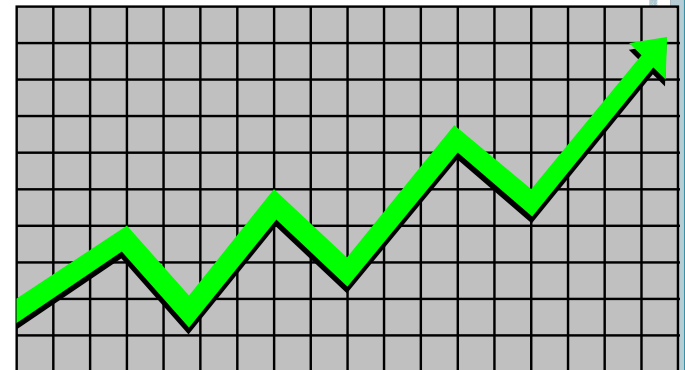
OBJECTS

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

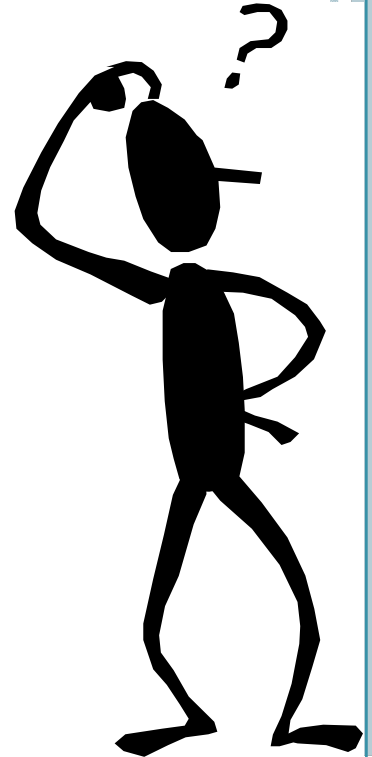
- A window object is responsible for things like opening, sizing, and closing itself.
- A chart object is responsible for things like maintaining its data and labels, and even for drawing itself.



TWO BASIC QUESTIONS

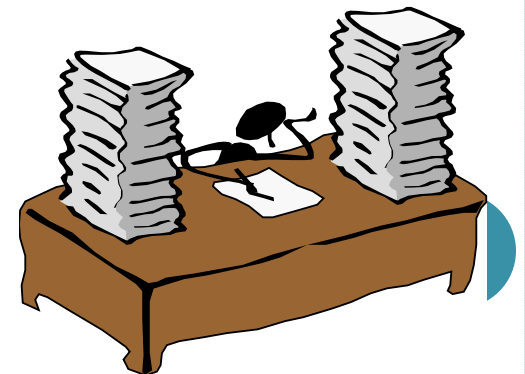
When developing an O-O application, two basic questions always arise.

- 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
- The introduction of new classes of data with different needs requires changing the main logic of the program
- It also may require the addition of new code in many different areas of the application



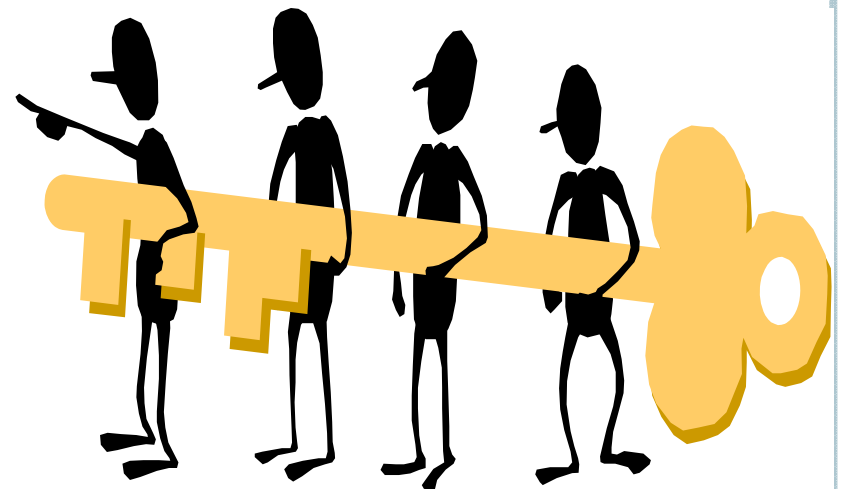
CONT..

- This problem limits the programmers ability to reuse the code, since each function is tied to the data on which it operates
- On the other hand OOP allows the programmer to solve problems by creating logical objects that incorporate both data and functionality along with providing reusability



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.



CONT..

- The OO system development consists of:
 - OO Analysis
 - OO information modeling
 - OO Design
 - Prototyping and Implementation
 - Testing, iteration, and documentation



OBJECTS AND IDENTITY

- Every object has its own unique and immutable identity – which never be confused with another object even if the original object is deleted
- The identity name never changes even if all the properties of the object change – it is independent of the object's state
- In a system it can be implemented through some kind of object identifier (OID) or unique identifier (UID)



STATIC AND DYNAMIC BINDING

- The process of determining at run time which function to invoke is termed dynamic binding
- Making this determination earlier, at compile time, is called static binding
- Static binding occurs when polymorphic calls are issued



OBJECT PERSISTENCE

- Objects have a lifetime, as they are explicitly created and can exist for the time period of the process for which they were created
- A file or a database provide support for objects having longer lifeline – longer than the duration of the process for which they were created
- This characteristic is called object persistence



META CLASSES

- In an OO system, everything is an object : hence a class too. And if so, it must belongs to some class
- Indeed, such class belongs to a class called meta-class. In short all the objects are instances of a class and the classes are instances of a meta-class
- A meta-class is also called an instance of itself



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

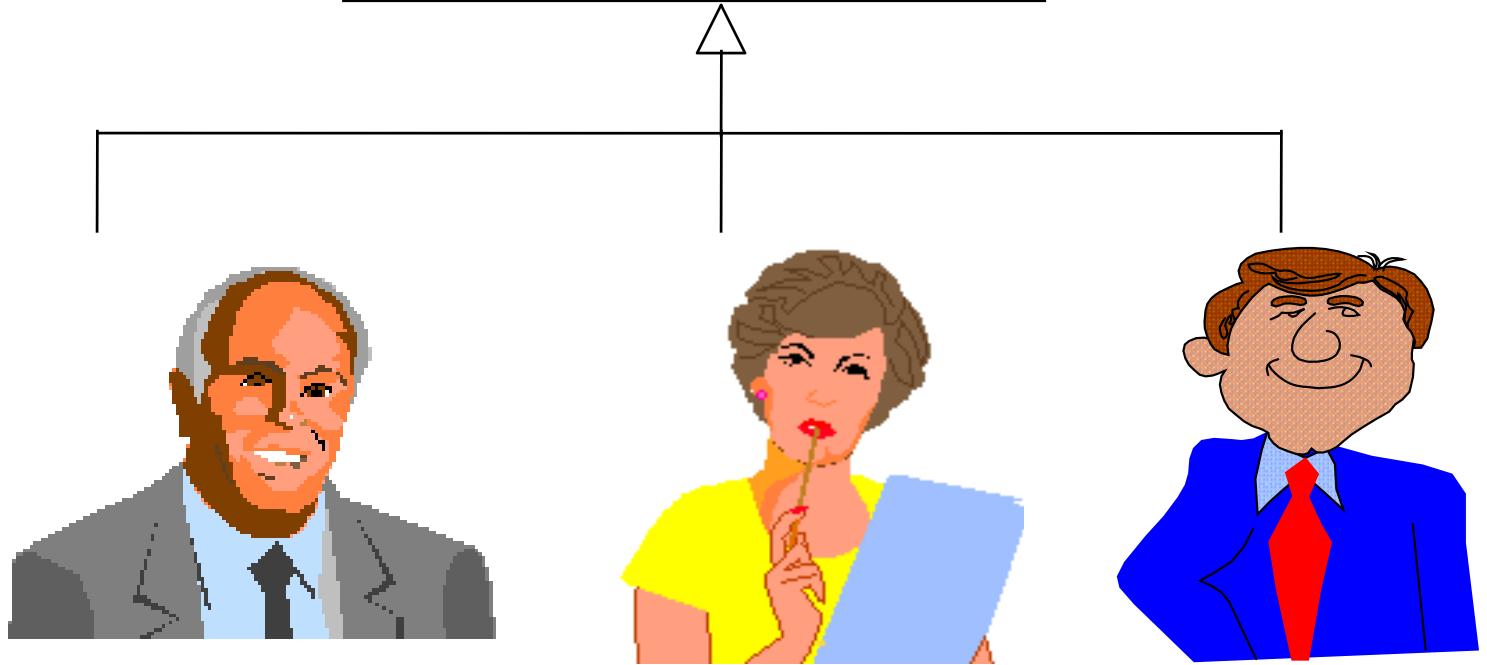


OBJECTS ARE GROUPED IN CLASSES

- The role of a class is to define the attributes and methods (the state and behavior) of its instances
- We distinguish classes from instances.(e.g. eagle or airplane)
- 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

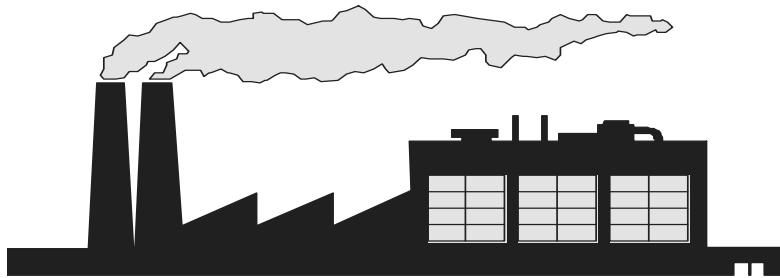


John object

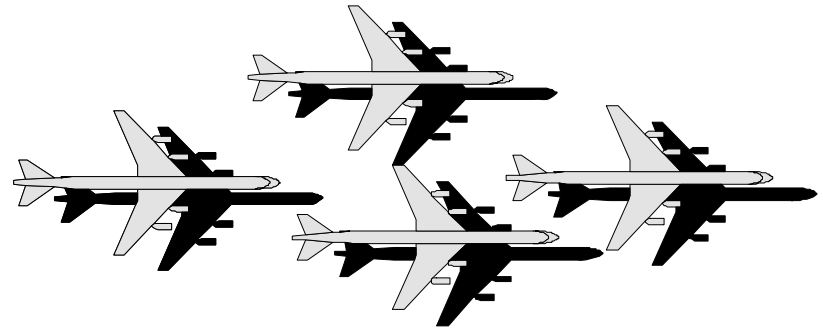
Jane object

Mark object

A CLASS IS AN OBJECT TEMPLATE, OR AN OBJECT FACTORY.



Boeing Factory
(Boeing class)



Boeing Airplane Objects
(Boeing instances)

ATTRIBUTES :OBJECTS STATE AND PROCEDURES

- Properties represent the state of an object.
- Each individual car (object) will have the attributes : cost, color, make, model



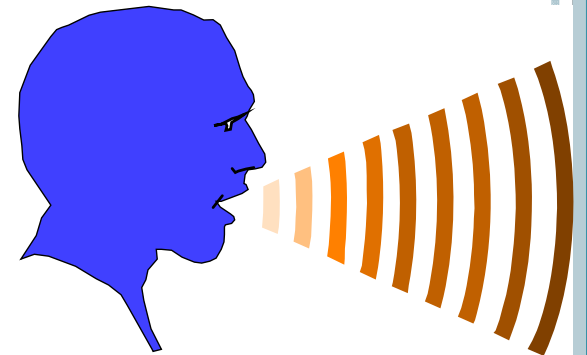
OBJECTS BEHAVIOR AND METHOD

- The method implements the behavior of an object.
(e.g. car and elephant)
- A method is a function or procedure that is defined for a class and typically can access the internal state of an object of that class to perform some operations.
- Behavior denotes the collection of methods that abstractly describes what an object is capable of doing.
- The object is called **receiver**.
- Methods encapsulate the behavior of the object.
- Employee object know how to compute its salary.




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.



OBJECTS RESPOND TO MESSAGE

- Method are equivalent to the functions
 - Message is sent to the object to perform the action.
 - **Messages** essentially are nonspecific function calls.
 - Message is different from a subroutine call, since different objects can respond to the same message in different ways.(e.g. stop message for car and bike).
 - It is the receiver's responsibility to respond to a message in an appropriated manner.
 - It is known as **polymorphism**.
 - Difference between message and method.
 - The message is the instruction and the method is the implementation.
- 

OBJECTS RESPOND TO MESSAGE

- Difference between message and method.
- The message is the instruction and the method is the implementation.
- An object understands a message when it can match the message to method, that has the same name as the message.
- If not found it search in the super class. If not found return an error.
- Message is different from function (function says how to do something and a message says what to do).



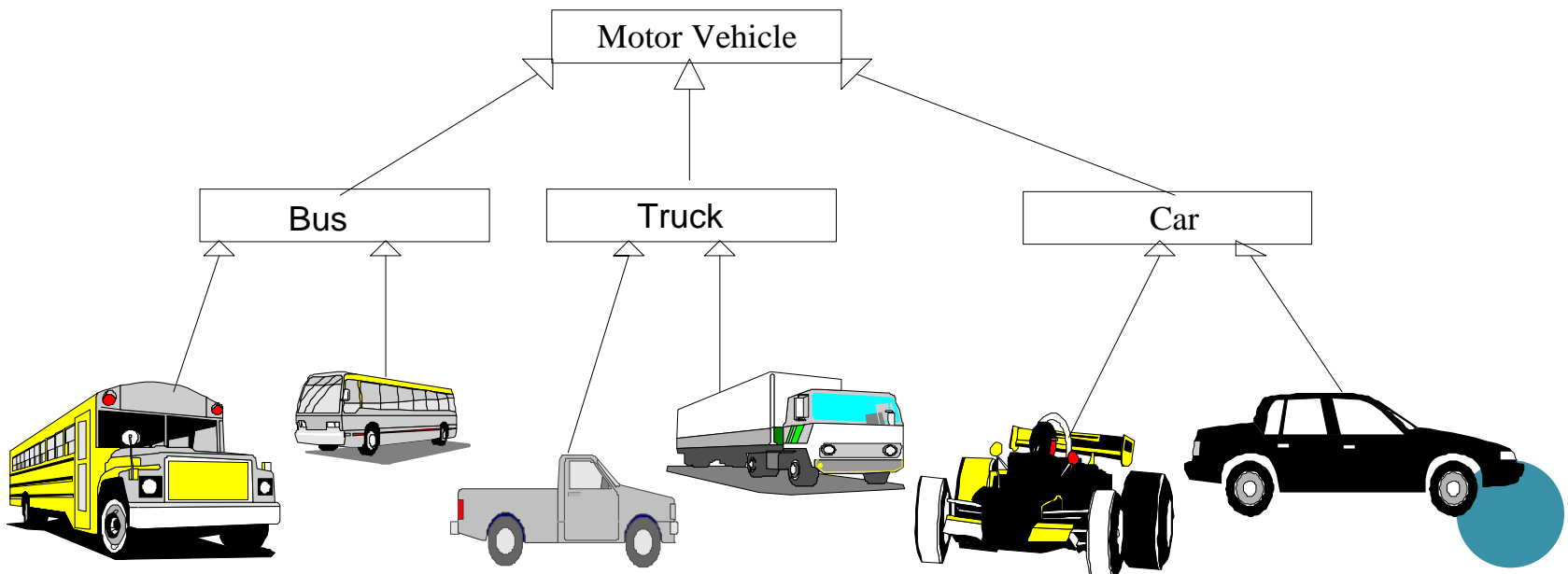
CLASS HIERARCHY

- 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.

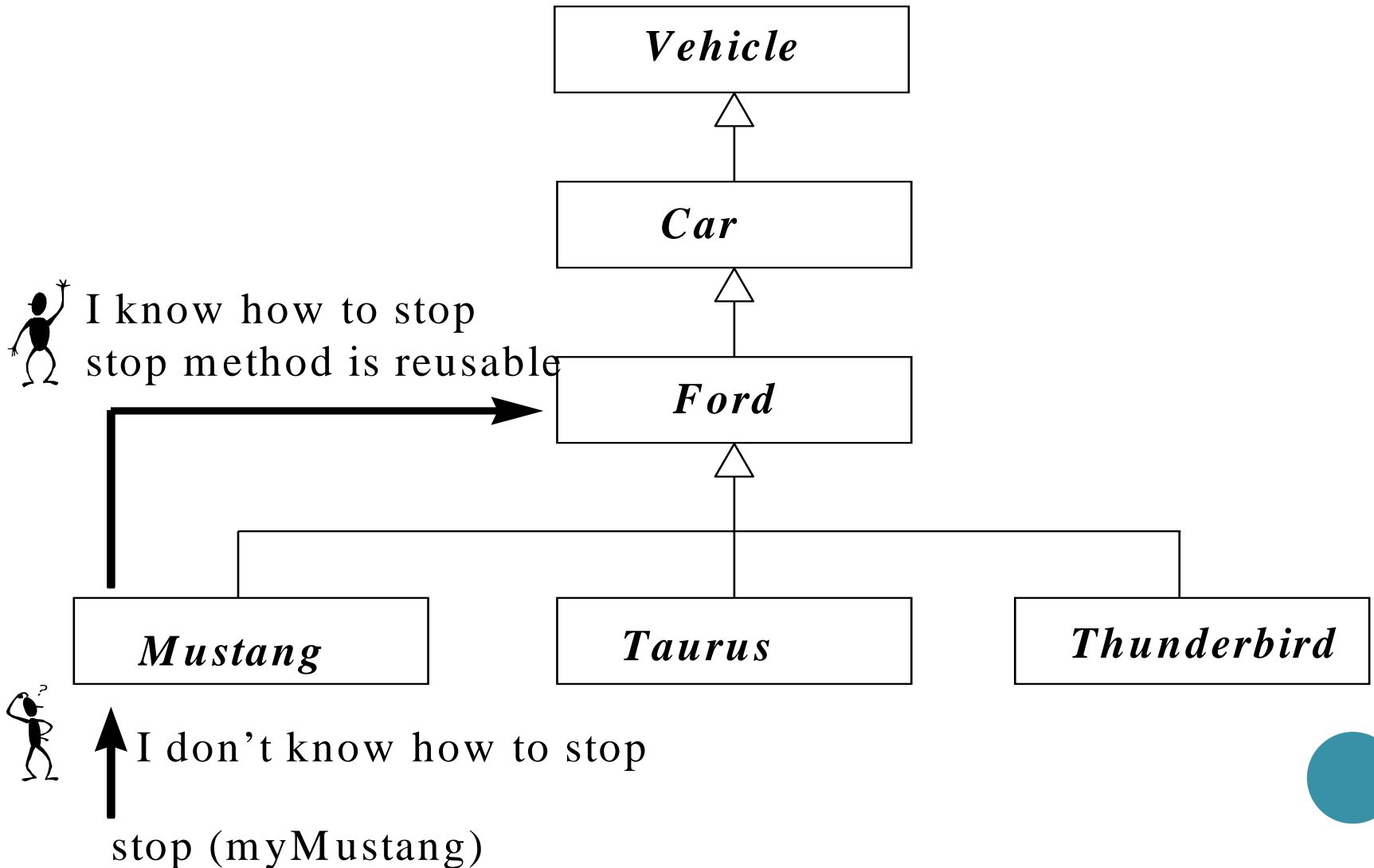


INHERITANCE

(PROGRAMMING BY EXTENSION)

- Inheritance is a relationship between classes where one class is the parent class of another (derived) class.
 - It allows classes to share and reuse behaviors and attributes.
 - The real advantage of inheritance is that we can build upon what we already have and,
 - Reuse what we already have.
 - Dynamic inheritance : refers to the ability to add, delete or change parents from objects (or classes) at run time.
- 

INHERITANCE (CON'T)



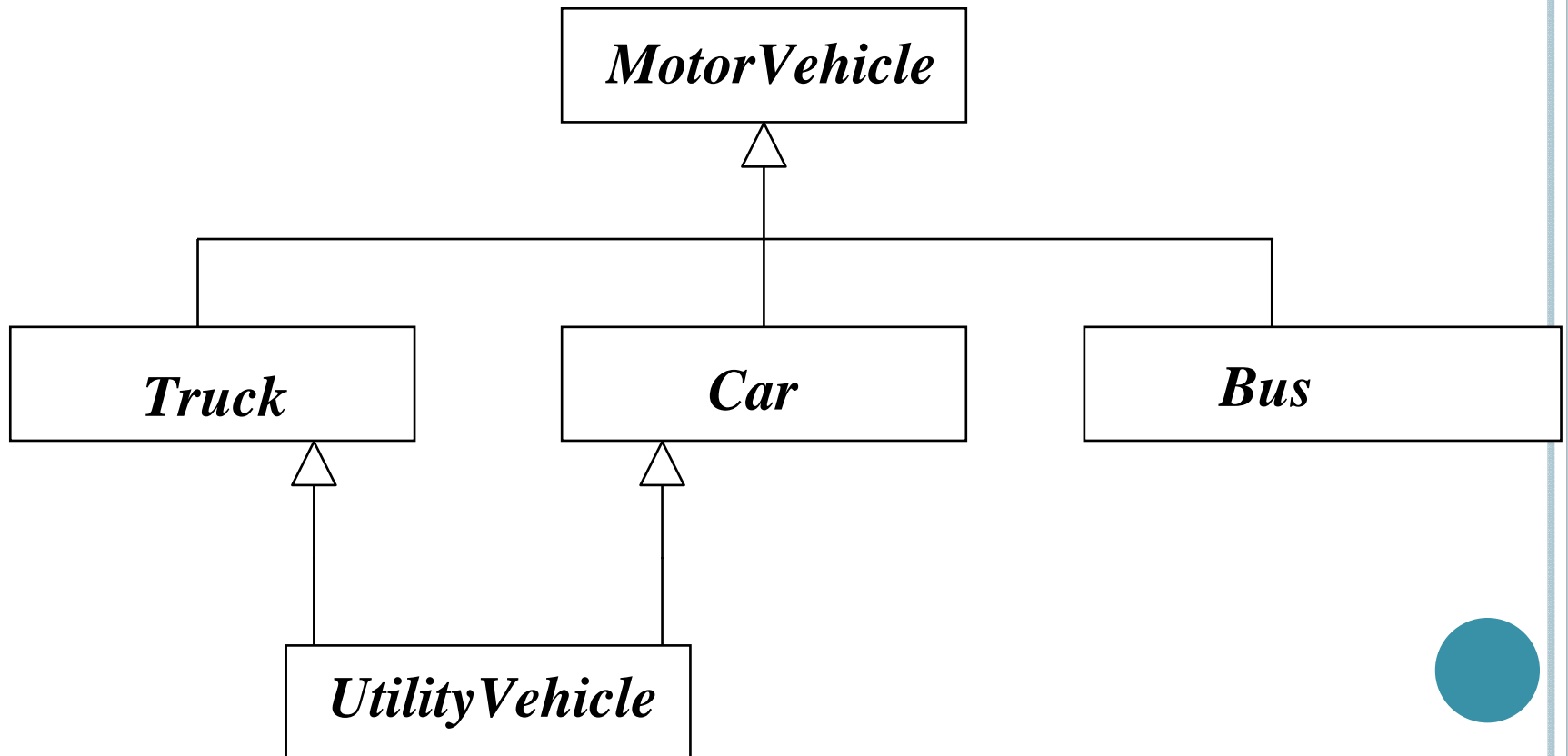
MULTIPLE INHERITANCE

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



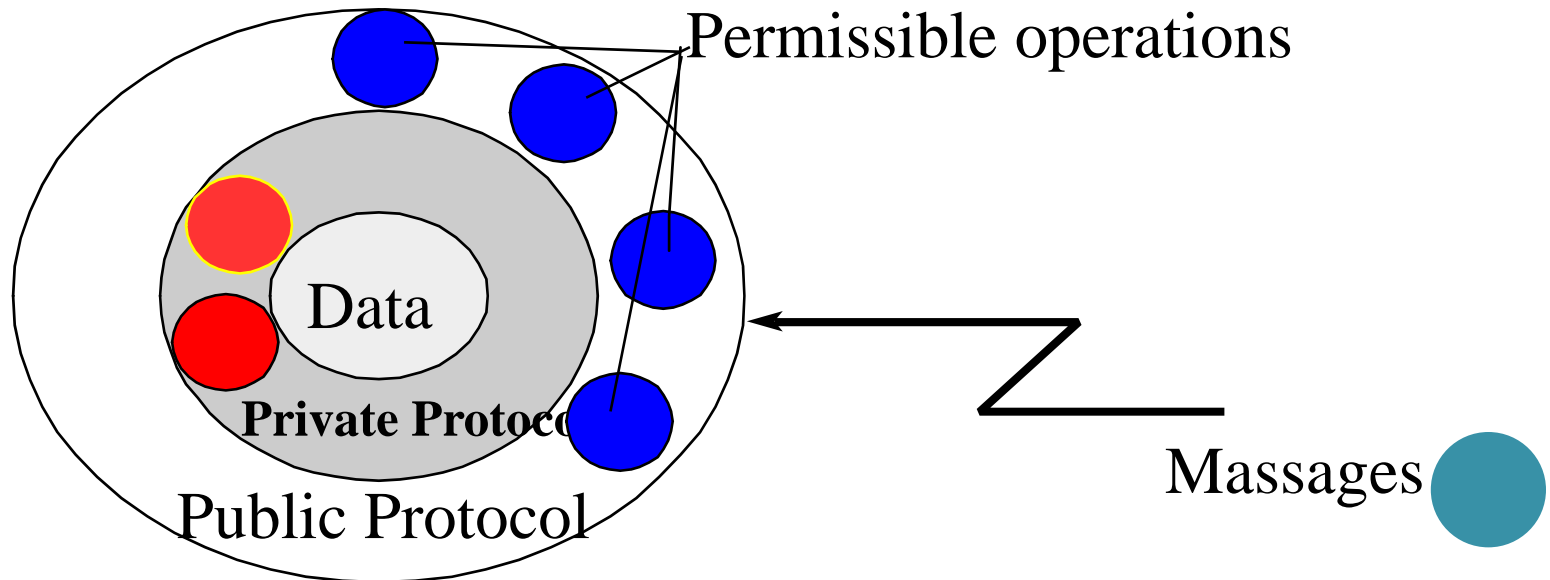
MULTIPLE INHERITANCE (CON'T)

- For example utility vehicle inherent from Car and Truck classes



ENCAPSULATION AND INFORMATION HIDING

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



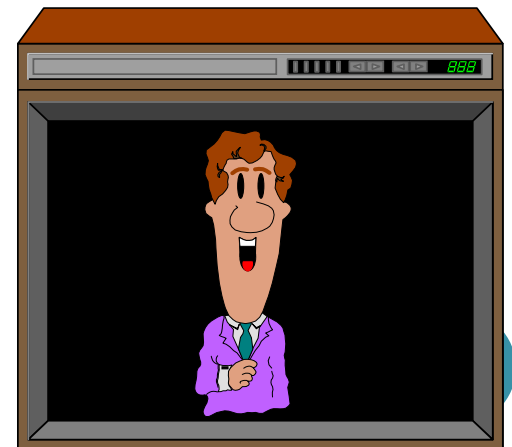
ENCAPSULATION AND INFORMATION HIDING (CON'T)

- 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.



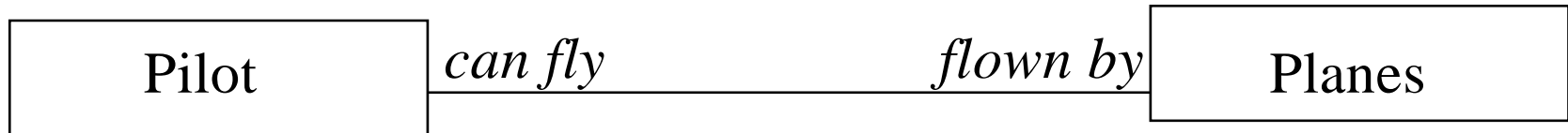
POLYMORPHISM

- 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.

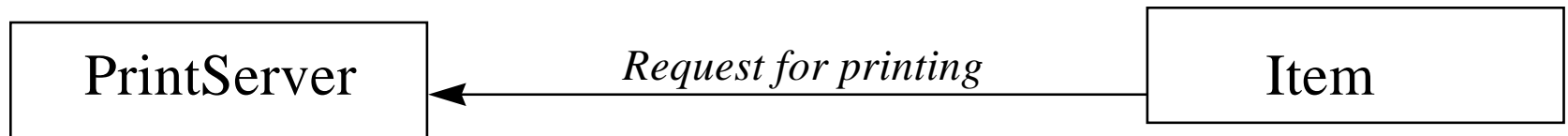


- Above association is bidirectional.
- **Cardinality** which specifies how many instances of one class may relate to a single instance of associated class.[one or many]



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).



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

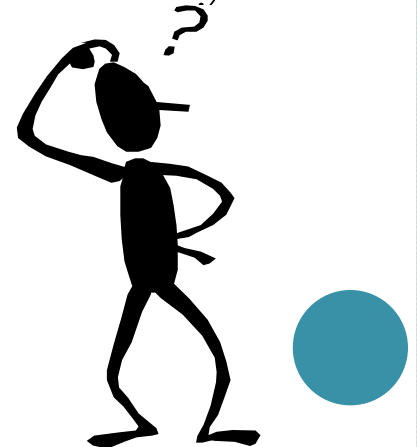
END



AN OBJECT-ORIENTED APPROACH

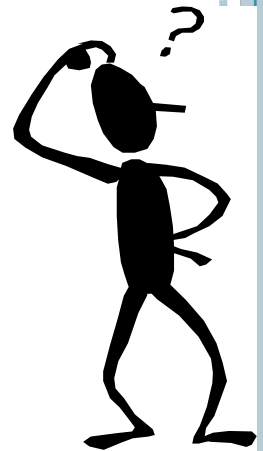
○ *What objects does the application need?*

- 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: what role does the person play in the system.*
- *Places: physical location.*
- *Things or events: who, what, when, where , how or why.*



WHAT ARE SOME OF THE APPLICATION'S CLASSES?

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

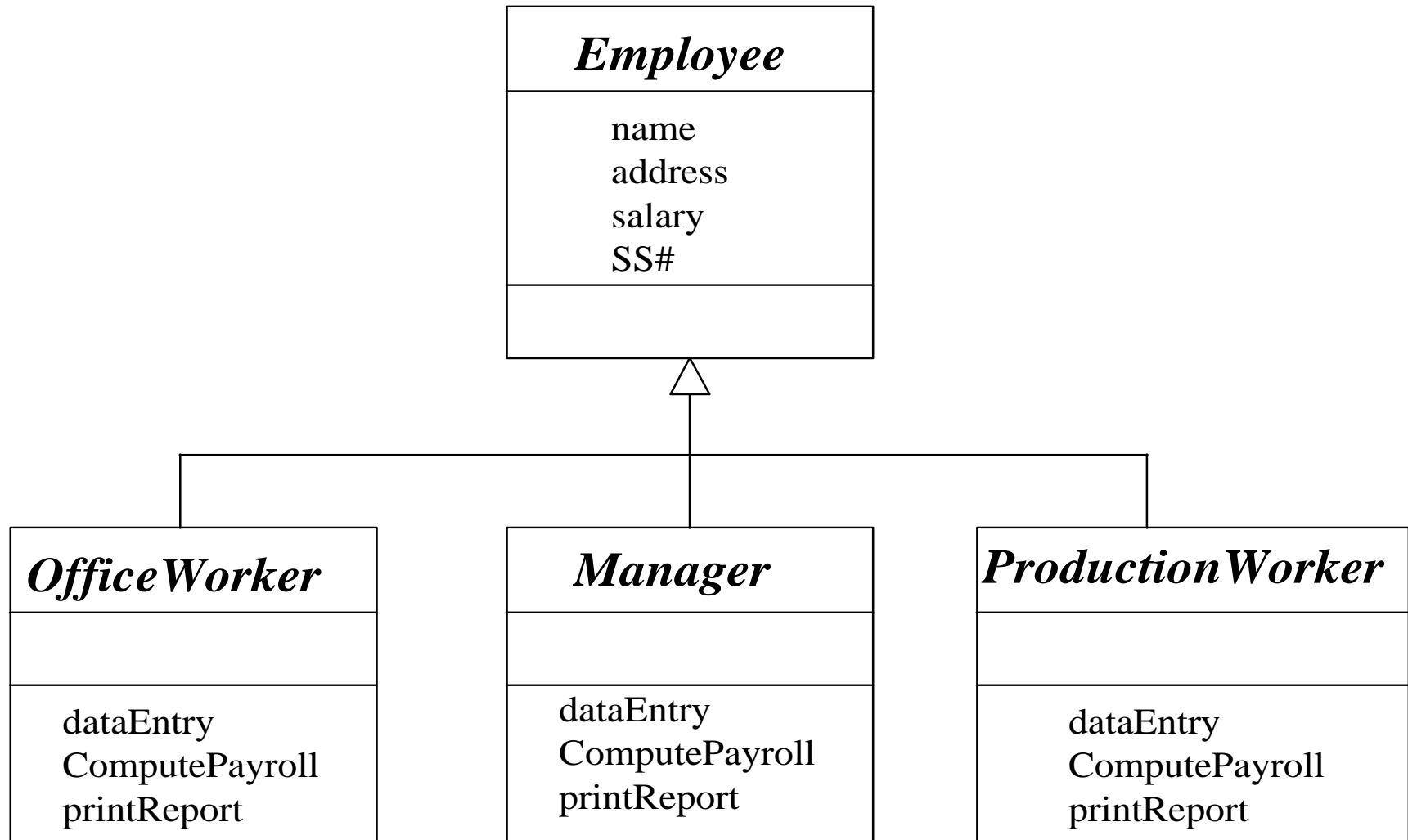


CLASS HIERARCHY

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



CLASS HIERARCHY (CON'T)

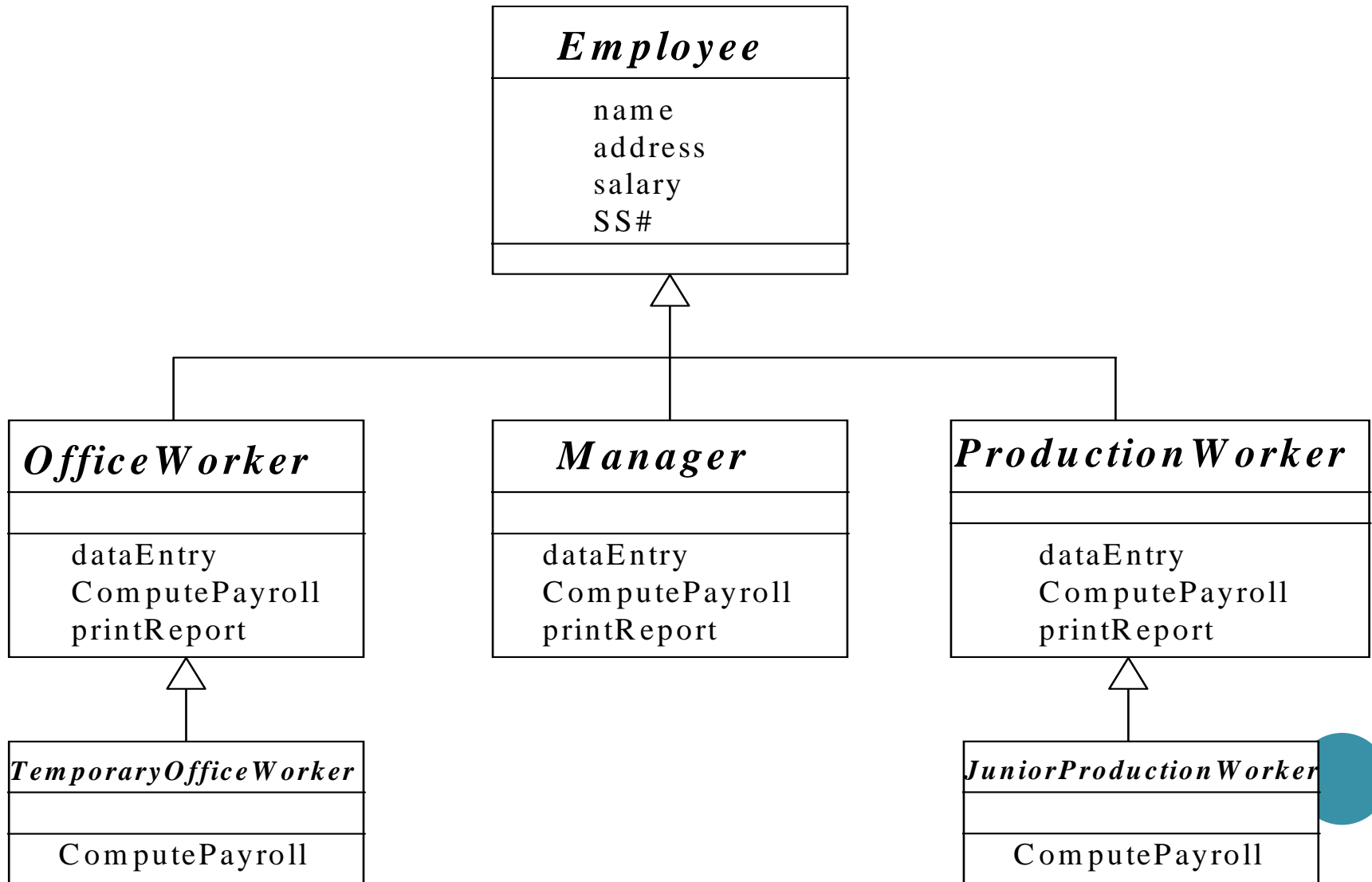


OO APPROACH

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



IF A NEW CLASS OF EMPLOYEE WERE ADDED



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
- **Advantages of object-oriented programming are:**
 - The ability to reuse code,
 - develop more maintainable systems in a shorter amount of time.
 - more resilient to change, and
 - more reliable, since they are built from completely tested and debugged classes.

