Objects 101:
An Introduction

- Benefits of Object Technology
- Object Mythology
- Anatomy of an Object
- Anatomy of a Class
- Classes and Instances
- Anthromorphizing Objects
- Definitions of:
  - *Message*
  - *Encapsulation*
  - *Inheritance*
  - *Polymorphism*

Benefits of Object Technology

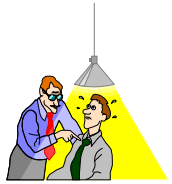
- Faster development
- Reduced cost
  - *Reuse*
- Higher quality systems
- Easier maintenance
- Improved architecture
  - *Adaptability*
  - *Scalability*

↳ OT Mythology

It's not magical!

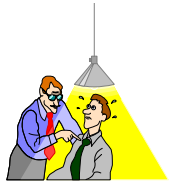
Benefits come from careful analysis, design, and development practices that exploit the OT paradigm!

OT Mythology - 1
Fact vs. Fiction

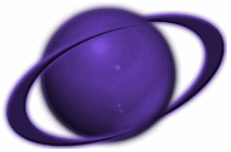


- Faster development
  - *First OO project is usually not faster or as cost-effective as a traditional approach*
  - *Payback comes in subsequent projects, but only if the development team is well seasoned*
  - *Expect approximately 2-3 development projects (each project about 6 months in duration) before seeing a return on investment*
- Reduced cost
  - *Reuse*
    - » *Reuse is not free!*
    - » *Achieving reuse costs time and money*
    - » *Takes effort in analysis, design, implementation, and testing*

OT Mythology - 2
Fact vs. Fiction



- Higher quality
  - *OT does <u>not</u> guarantee quality, people instill quality*
- Easier maintenance
  - *Problems will be localized to a set of objects*
  - *However, inheritance hierarchies and polymorphic operations can present problems when determining errors*
- Improved architecture
  - *OT systems have a natural structure for modular design*
  - *However, nothing in OT compels or requires that the code produced will be modular*



Anatomy of an Object - 1

- An object is an abstraction and has an identity
- An object has a name (which may or may not be unique)
- An object is anything that exhibits structure and behavior
  - *Structure*
    - » *Attributes that define the object's properties*
    - » *Each attribute takes on a single value for a given object*
  - *Behavior*
    - » *Operations performed on or by an object*
    - » *Each operation is implemented via a method*
- Objects can be related or linked to other objects

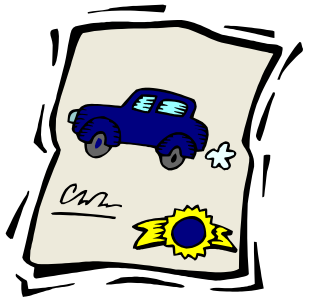
Anatomy of an Object - 2

Object: Bill's Automobile

Attributes:
Color=Blue
Make=Ford
Model=Explorer
Owner=Bill

Operations:
Start
Drive
Park
Hit

Relationships: covered by Bill's auto policy



Object: Bill's AutoPolicy

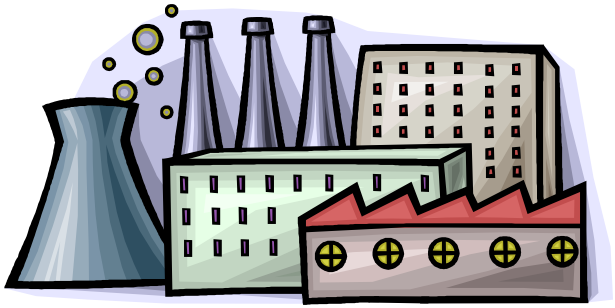
Attributes:
Policy#=83-598-2
Premium=$920.00
Expiration=6/1/06
Holder=Bill

Operations:
Rate
Issue
Cancel
Renew

Relationships: covers Bill's Automobile

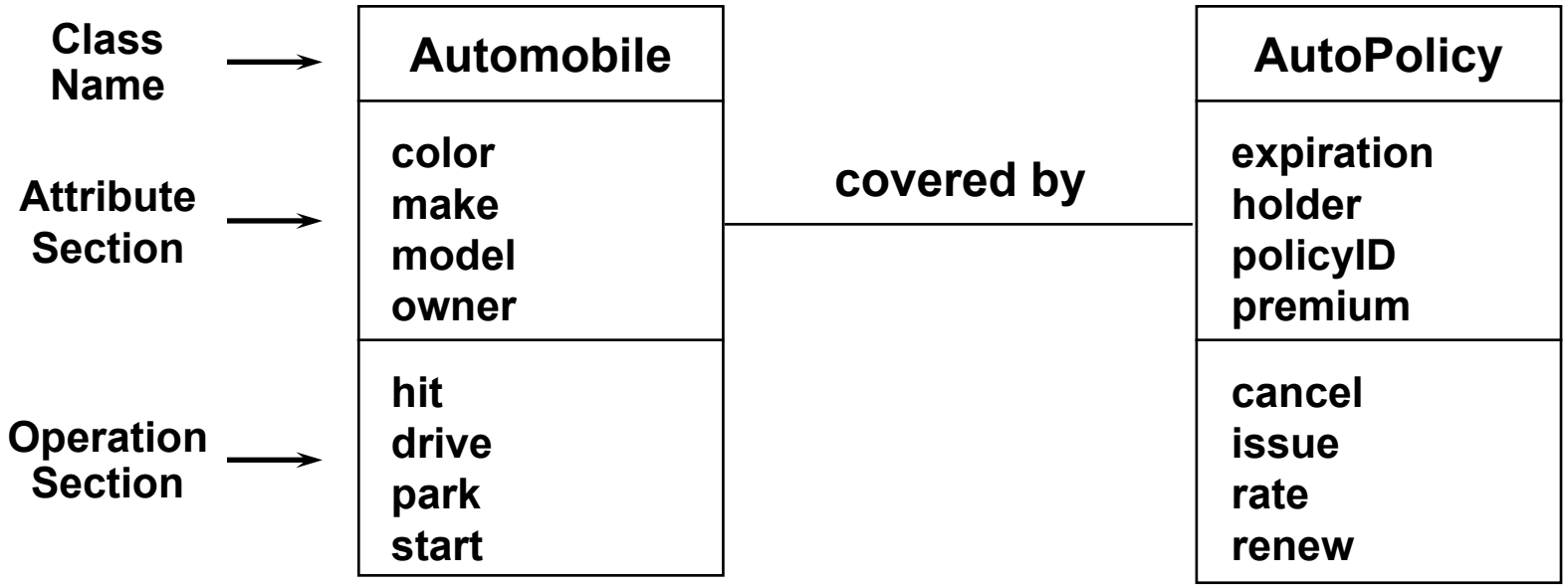
Anatomy of a Class - 1

- A class describes a group of objects with the same
  - *Structure*
  - *Behavior*
  - *Common relationships to other objects*
  - *Common semantics*
- A class may represent one or more objects
  - *It's a "blueprint" for objects*
  - *An "object factory"*
- A class defines
  - *Attributes*
  - *Operations*



- A class may be related or associated to other classes

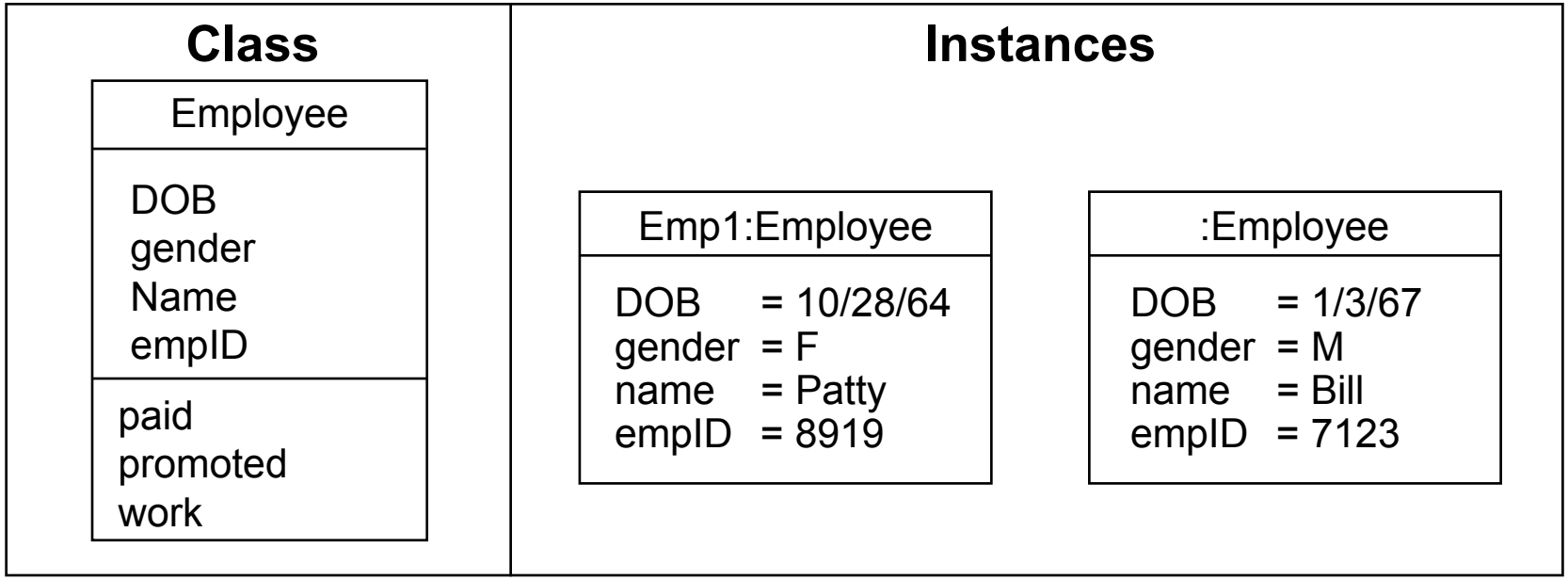
Anatomy of a Class - 2*



* Shown using Unified Modeling Language (UML) notation

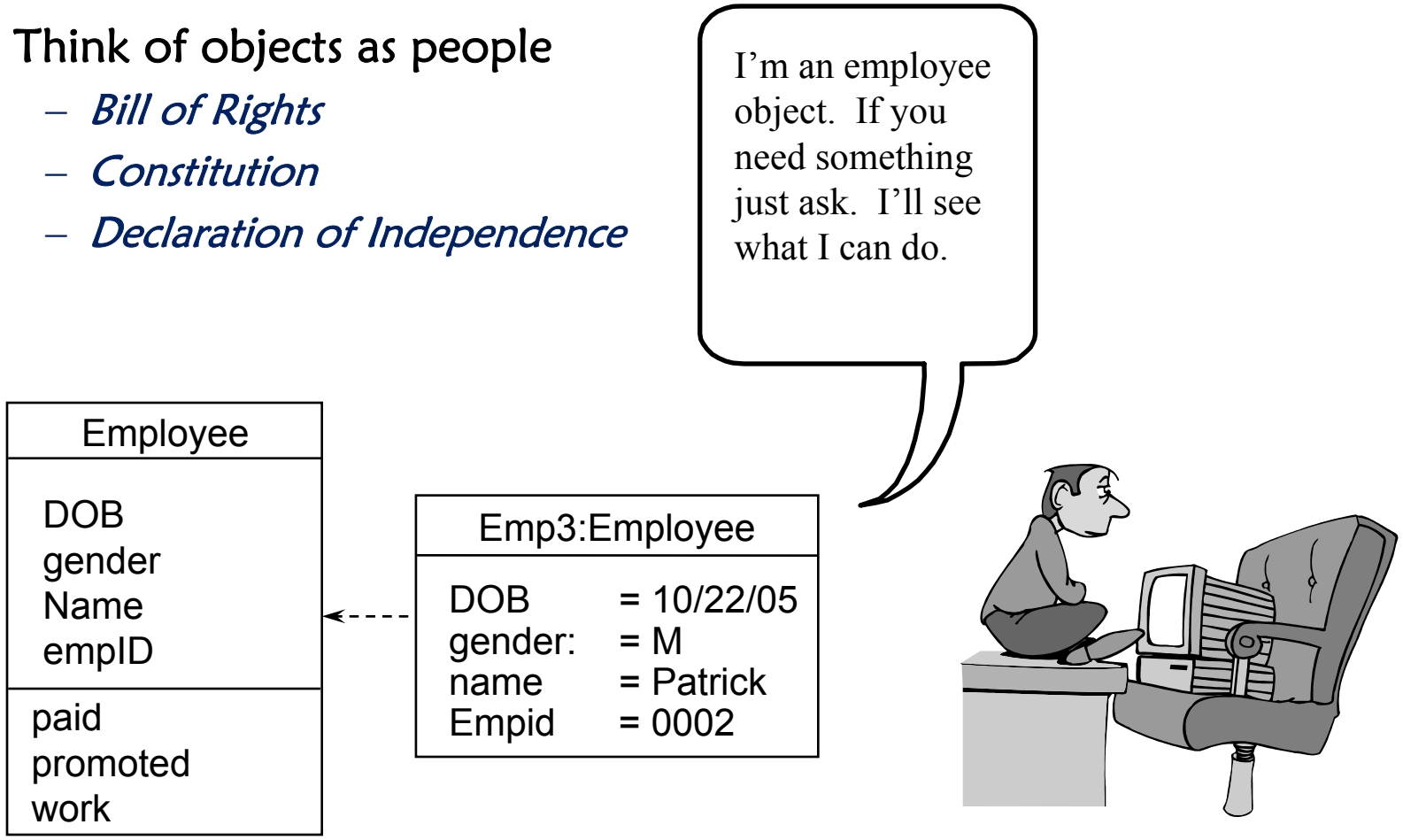
Classes and Instances

- Every object is an instance of a class
- Every instance of a class has the same
  - *Operations*
  - *Attributes (but may have different values for those attributes)*



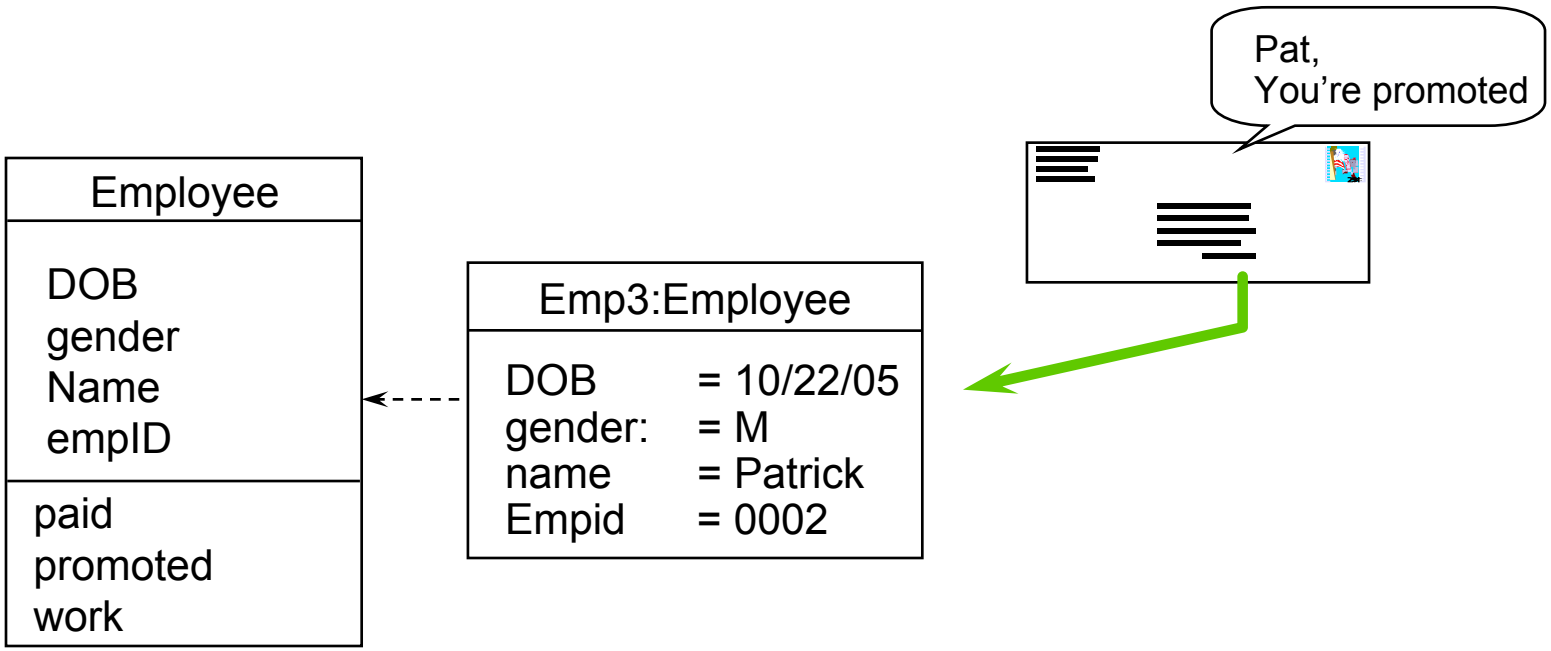
Anthropomorphizing Objects

- Think of objects as people
  - *Bill of Rights*
  - *Constitution*
  - *Declaration of Independence*



Message *Defined*

- Any request for a service that is sent or received by an object

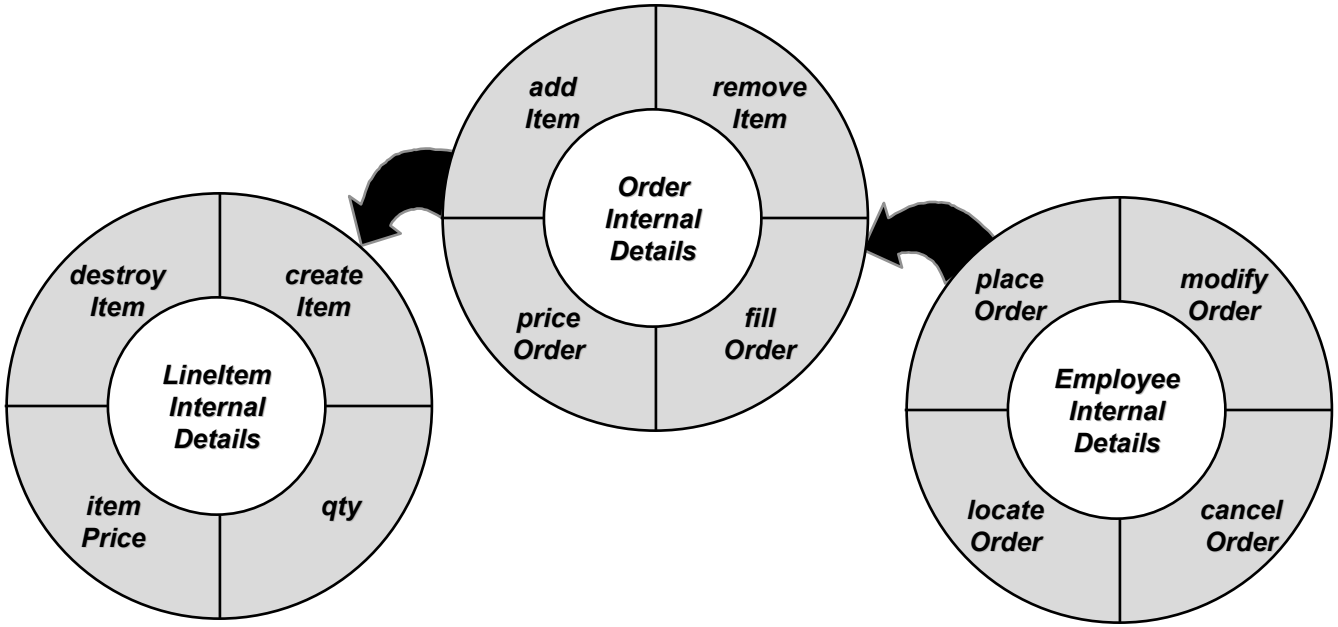


Object Technology
Easy as PIE!

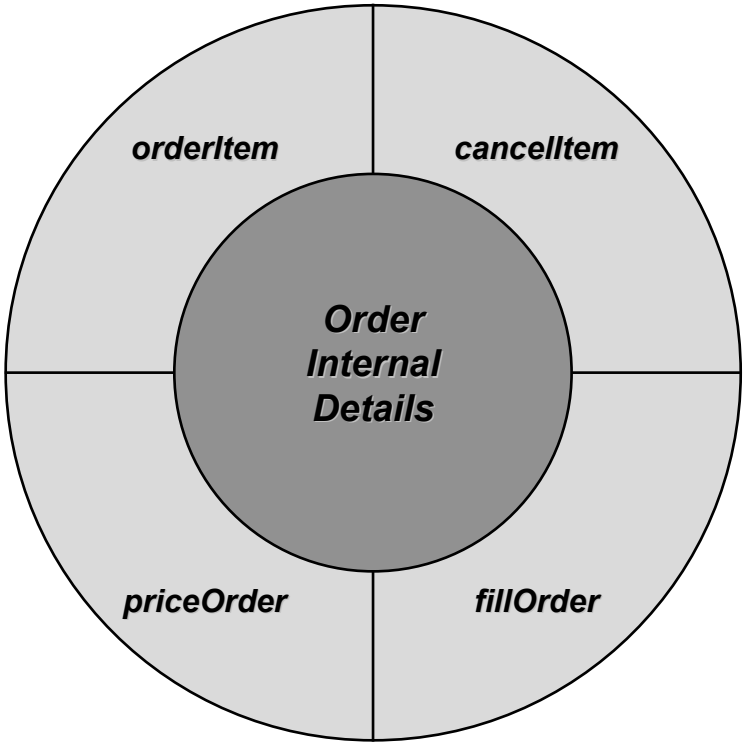
- Polymorphism [3]
- Inheritance and Hierarchies [2]
- Encapsulation [1]

Encapsulation *Defined*

- The combination of attributes and operations that manipulate those attributes into a single unit

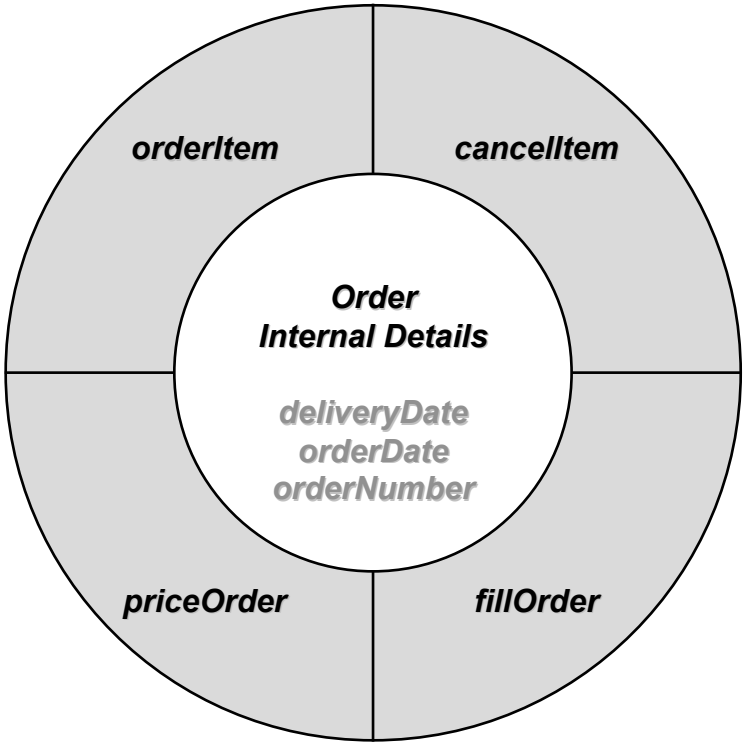


Encapsulation - 1



- Each object is a “black box”
- Its internal details are hidden
- It can only be accessed through its operations
- Information hiding is a mechanism for limiting complexity

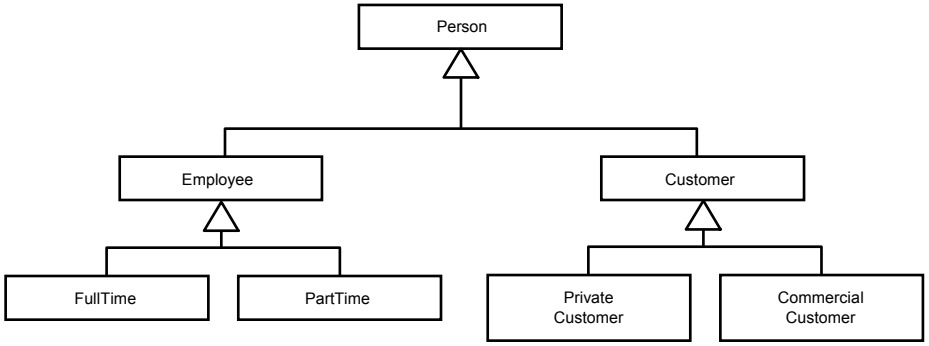
Encapsulation - 2

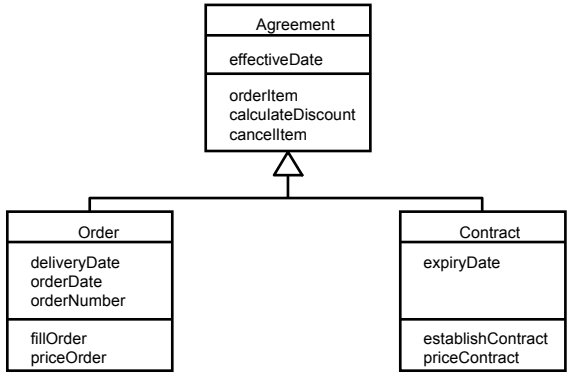


- Changes to the object's internals do not affect the outside world
- Side effects from modifications are localized

Inheritance *Defined*

- A mechanism that permits classes to share attributes and operations based on a generalization relationship

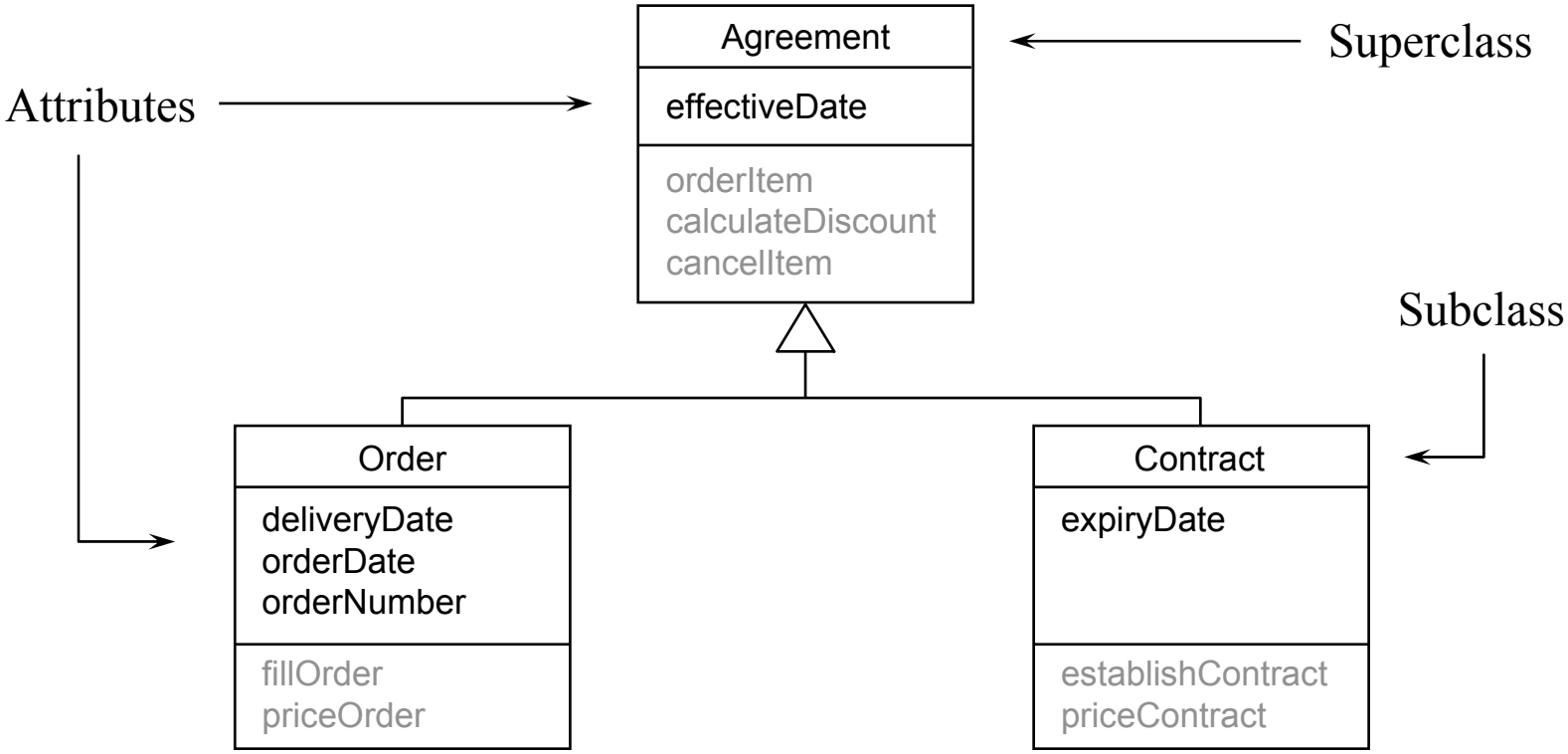




Enables classes to share commonality

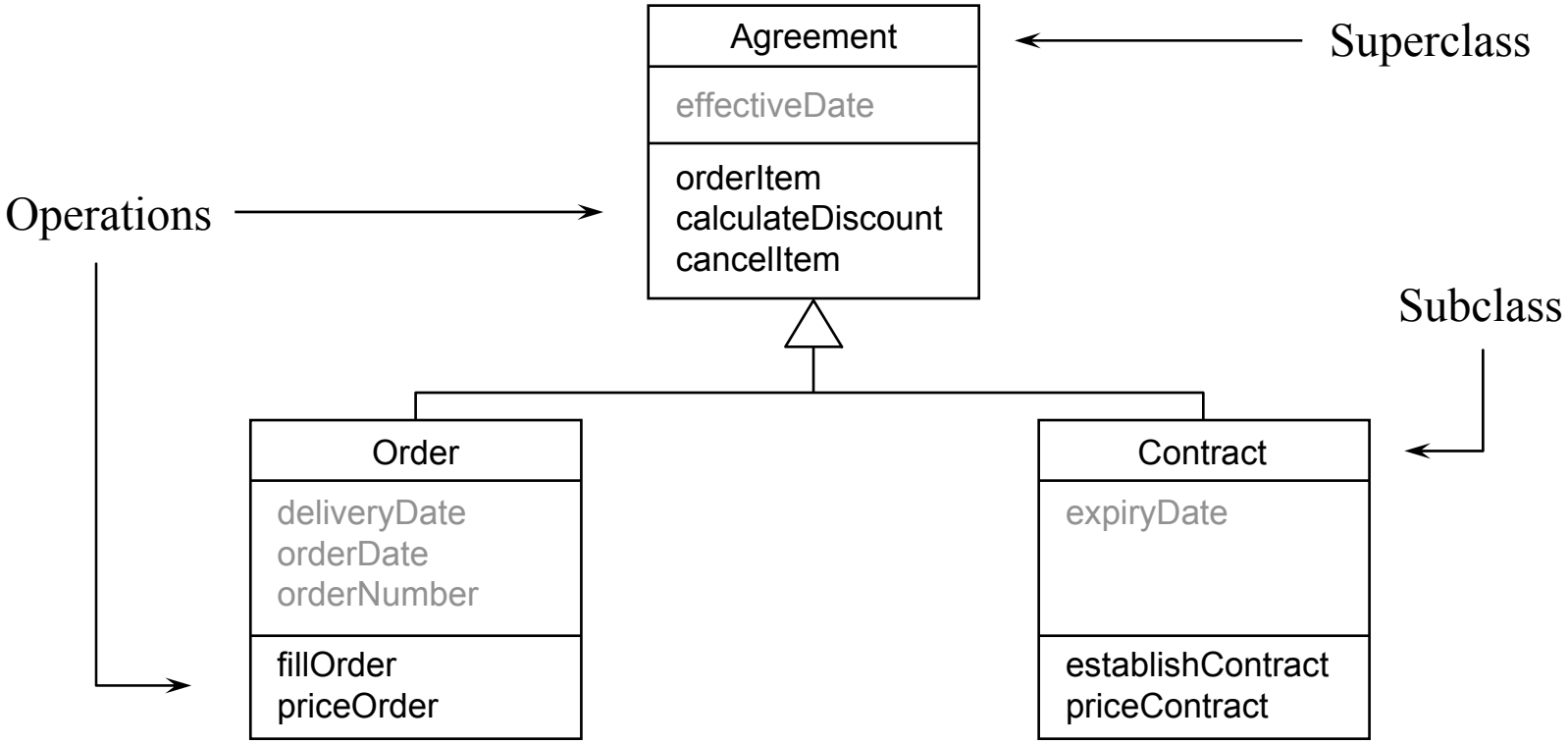
Inheritance
Attributes

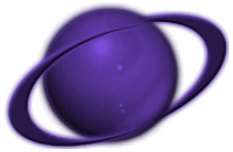
- Common attributes are encapsulated in the superclass
- Specific attributes encapsulated in the subclass



Inheritance
Operations

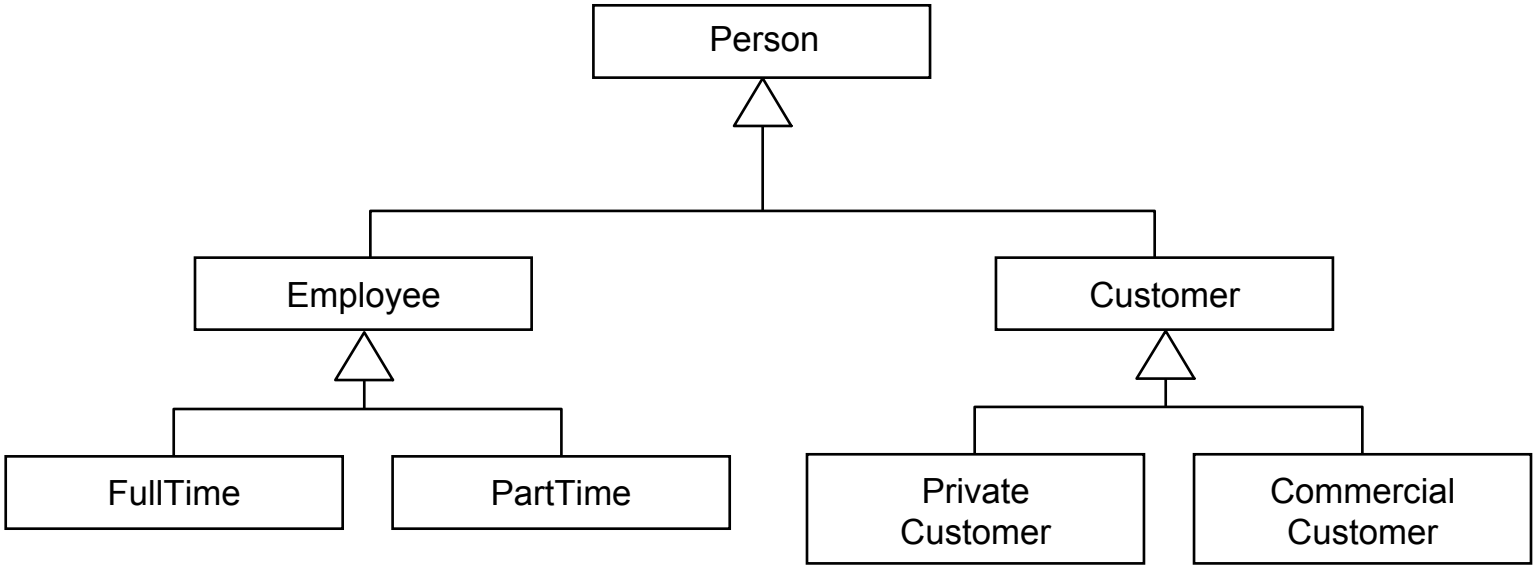
- Common operations are encapsulated in the superclass
- Specific operations are encapsulated in the subclass





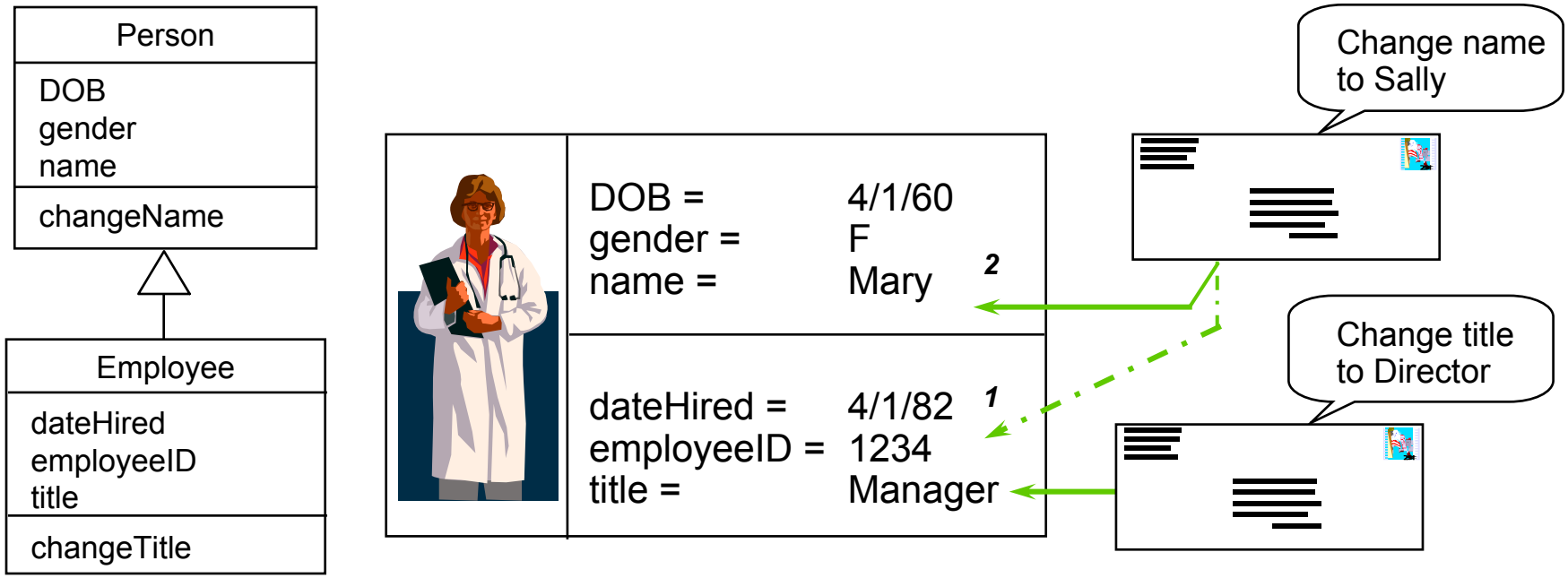
Inheritance
Multiple Levels

- Inheritance hierarchies can be several layers deep



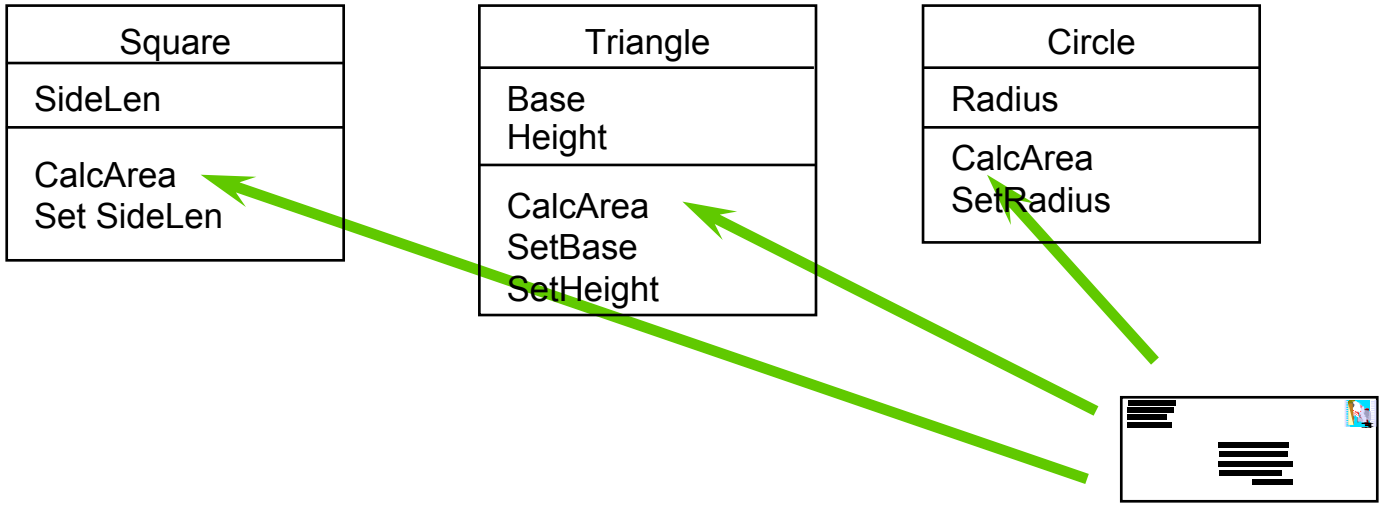
Inheritance
Hierarchy Traversal

- Send the object a message
- Object traverses up its hierarchy until it finds an operation that can respond to the message



Polymorphism *Defined*

- The ability to have the same message evoke different responses in different objects



Greek term meaning "many forms"

Polymorphism
At the Orchestra

Each Instrument Responds to The Same Message Differently

Polymorphism
Operations

What happens when classes implement the same operation differently?

Square
SideLen
CalcArea SetSideLen

Triangle
Base Height
CalcArea SetBase SetHeight

Circle
Radius
CalcArea SetRadius

SideLen: 3

CalcArea
SetSideLen

Base: 4
Height: 2

CalcArea
SetBase
SetHeight

Radius: 2

CalcArea
SetRadius

CalcArea:

Area = SideLen ^{2}

CalcArea:

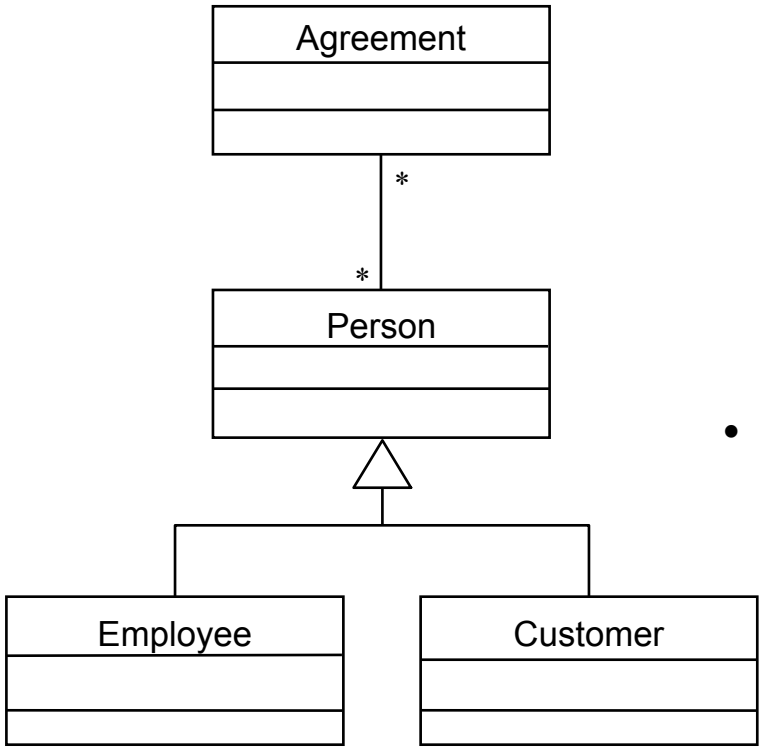
Area = 0.5 * Base * Height

CalcArea:

Area = Pi * Radius ^{2}

This is called polymorphism.

Unified Modeling Language *(UML)*



- History
  - *Combination of existing OO methodologies: Booch, Object Modeling Technique (OMT), and OOSE*
  - *Principals methodologists: Grady Booch, James Rumbaugh, Ivar Jacobson*
  - *UML Release 1.1 ~ 1997*
  - *UML Release 2.0 ~ 2003*
  - *Object Management Group (OMG) standard*
- Major types of diagrams
  - *Structural diagrams*
    - » *What are the objects?*
    - » *How are they related?*
  - *Behavioral diagrams*
    - » *How do objects change state?*
    - » *What are the required sequencing of operations?*
    - » *What are the required collaborations of objects?*
    - » *Who controls the system?*