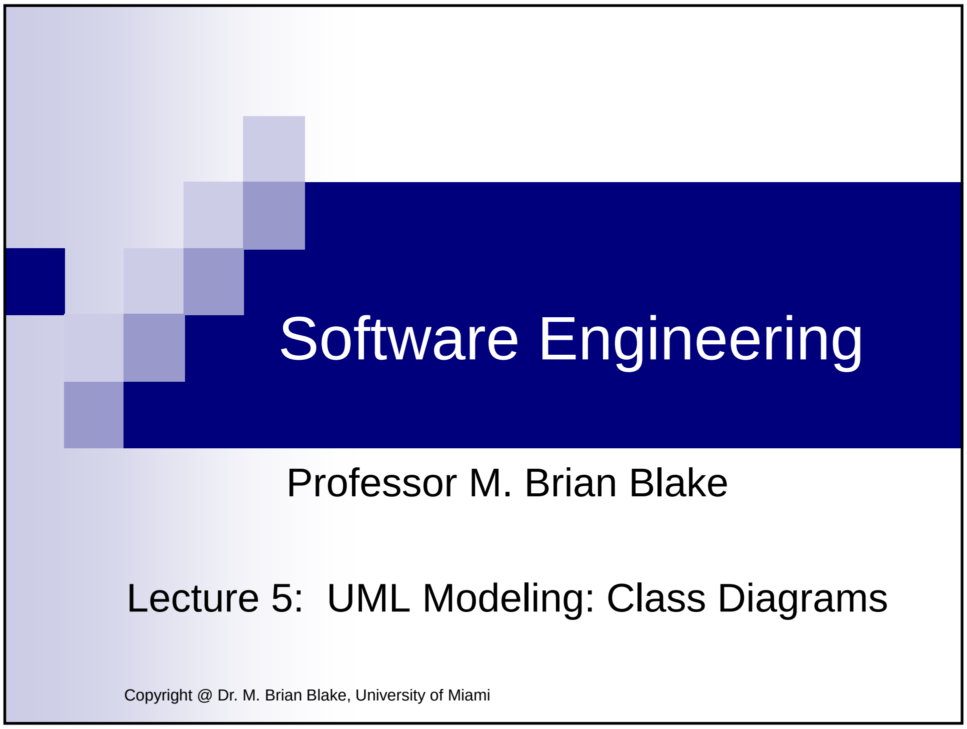
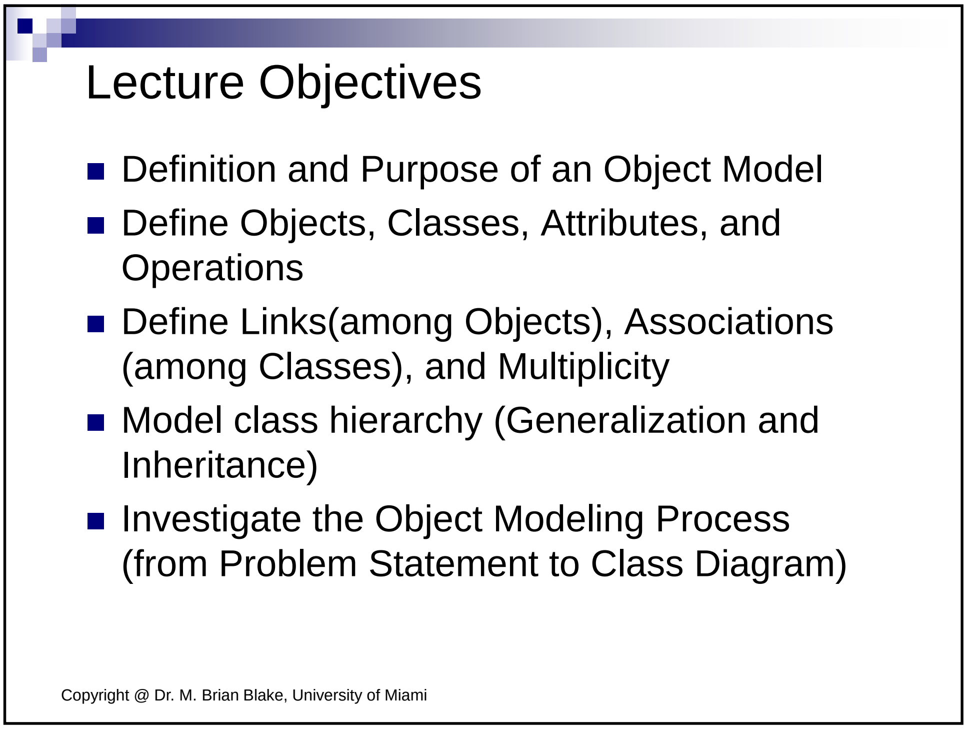


Software Engineering

Professor M. Brian Blake

Lecture 5: UML Modeling: Class Diagrams



Lecture Objectives

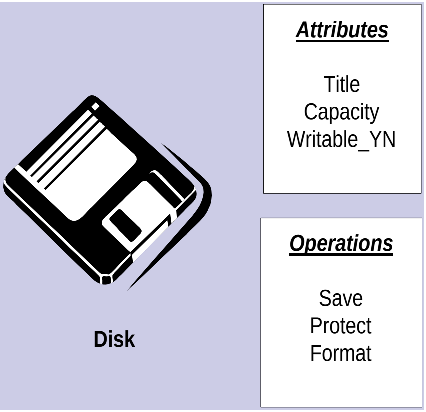
- Definition and Purpose of an Object Model
- Define Objects, Classes, Attributes, and Operations
- Define Links(among Objects), Associations (among Classes), and Multiplicity
- Model class hierarchy (Generalization and Inheritance)
- Investigate the Object Modeling Process (from Problem Statement to Class Diagram)

Lets Review Definitions

- What is Class ?
  - A group of objects with similar properties (attributes), common behavior (operations), common relationships to other objects (associations), and common semantics.

Object Characteristics

- An object has ***structure*** ~ attributes
  - An object must be an entity ~ a thing that can have properties and not be a property itself.
- An object has ***behavior*** ~ operations



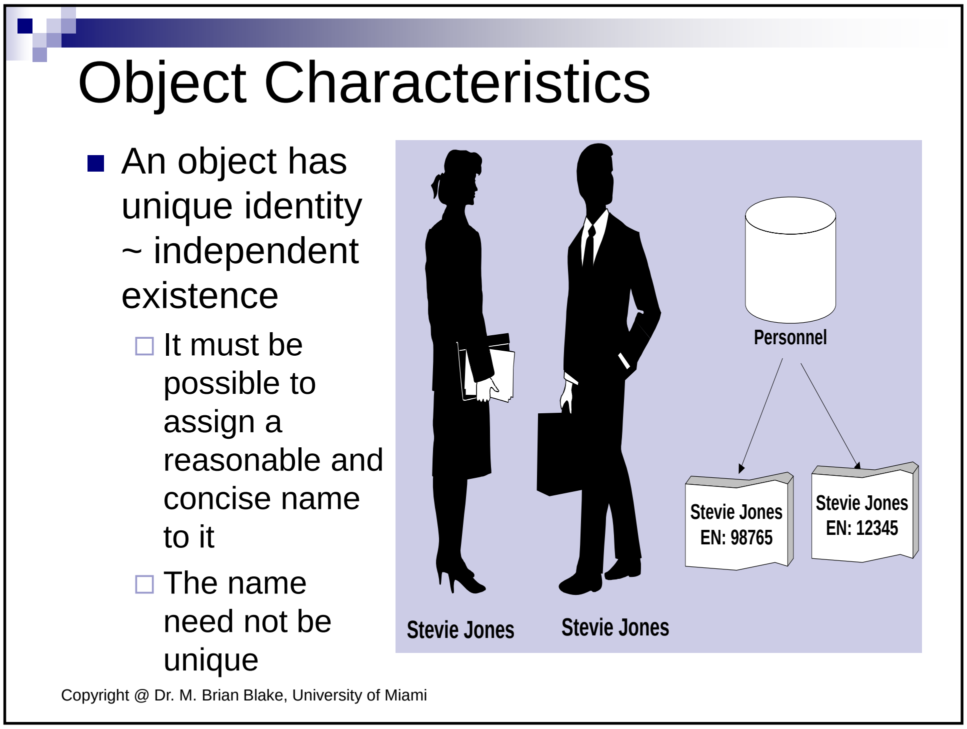
Disk

Attributes

Title
Capacity
Writable_YN

Operations

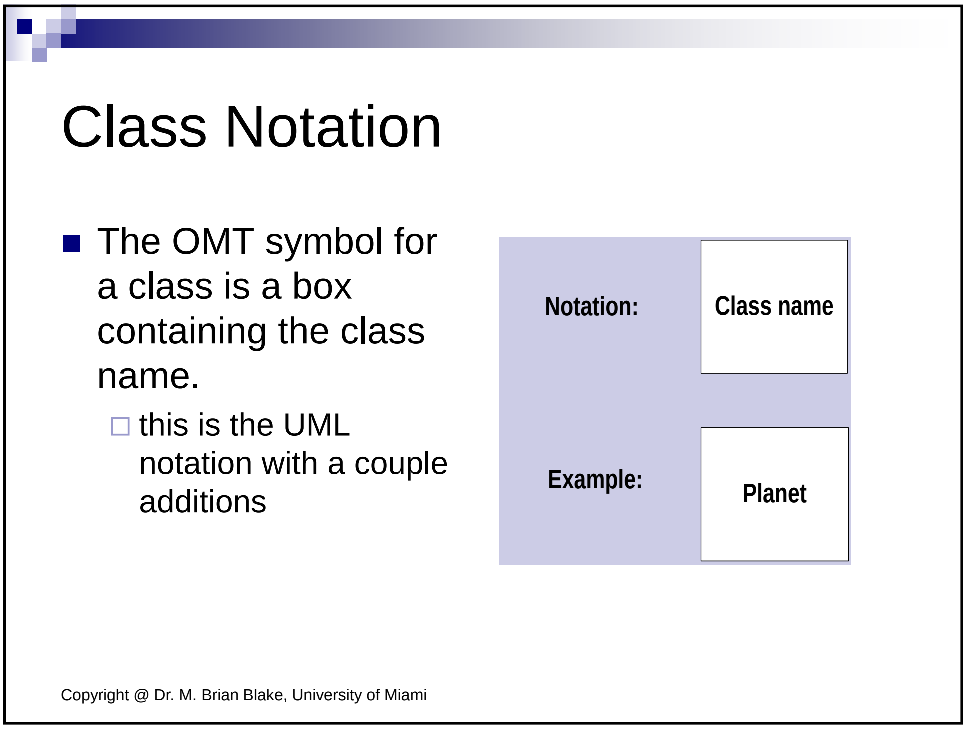
Save
Protect
Format



Object Characteristics

- An object has unique identity ~ independent existence
  - □ It must be possible to assign a reasonable and concise name to it
  - □ The name need not be unique

Personnel
Stevie Jones
EN: 98765
Stevie Jones
EN: 12345
Stevie Jones
Stevie Jones



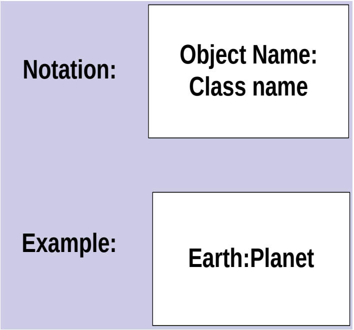
Class Notation

- The OMT symbol for a class is a box containing the class name.
  - □ this is the UML notation with a couple additions

Notation:
Class name
Example:
Planet

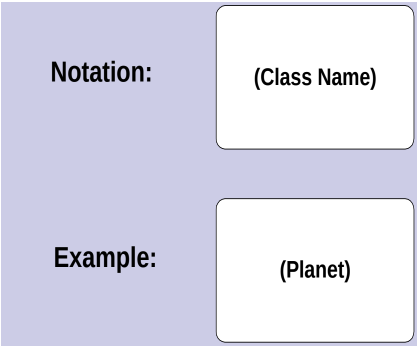
Object Notation

- The OMT symbol for an object is a rectangle containing the object name followed by a (:) and the class name at the top of the box



Alternative Object Notation

- The alternate OMT symbol for an object is a rounded box containing the class name in at the top of the box .
- This is sometimes referred to as an *instance*

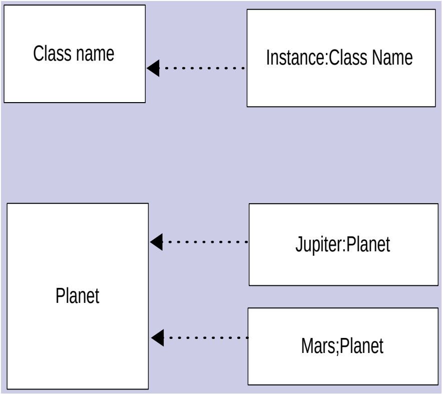


Class and Object Diagrams

- Typically, class diagrams are used more often
- Object diagrams are mainly used to:
  - □ Clarify a class diagram
  - □ Explore the nature of the object
  - □ Give management presentations
  - □ Show objects running in a particular system

Instantiation Notation

- When shown together, an object is often connected to its class with a dotted arrow

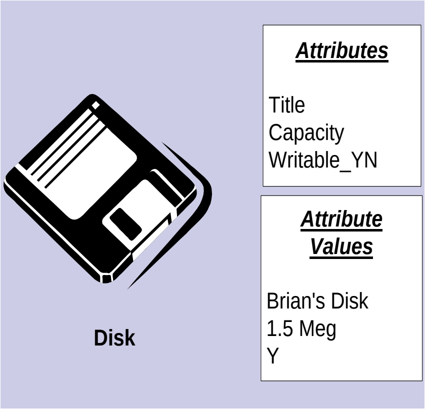


Choices of Objects During Analysis

- ***Domain objects*** ~ found in real-world problem domain carry semantics of the problem
  - □ Identified in early analyses
- ***Application objects*** ~ represent the intersection of the application and the user, visible to the user, and dependent upon the application (e.g. controllers, interface, devices, views)
  - □ Identified in later analyses
- ***Implementation objects*** ~ are implementation dependent and invisible to the user (e.g. stacks, buffers, queues)
  - □ Identified in later analyses

Attributes

- Attributes define the structural properties of classes
- Each object has is own attribute values
- Generally stated as a noun
- All objects of a class have the same attributes, maybe different values

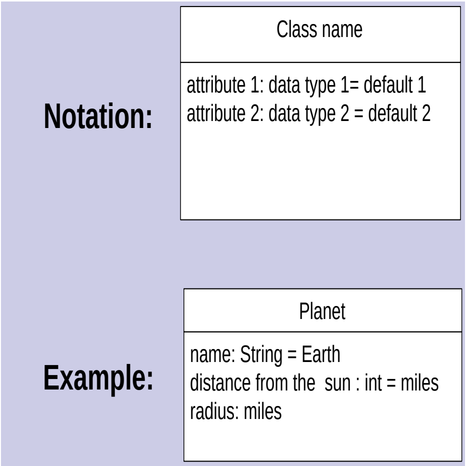


Exercise:Define the Attributes

- What are some of the attributes of the (previously identified) objects?
  - A convertible (e.g., light, tire, radio objects)
  - An airline (e.g., airplane, schedule, flight attendant objects)
  - A computer network (e.g., file, workstation, printer objects)

Class with Attribute Notation

- Attributes are contained in the class box by including them under a dividing line following the class name
- Attribute values are equated with an (=), date type precedes value



Notation:

Class name

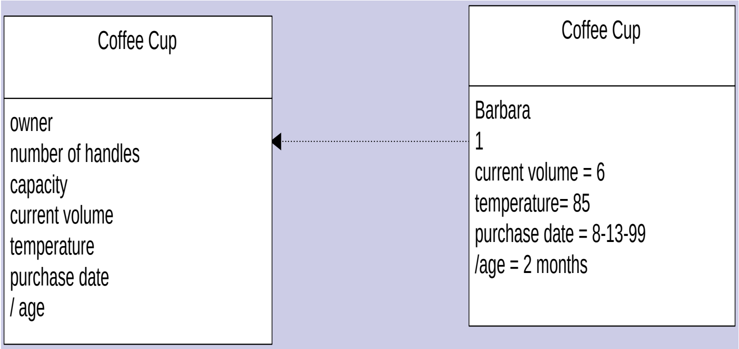
attribute 1: data type 1= default 1
attribute 2: data type 2 = default 2

Example:

Planet

name: String = Earth
distance from the sun : int = miles
radius: miles

Objects and Classes with Attributes



Coffee Cup

owner
number of handles
capacity
current volume
temperature
purchase date
/ age

Coffee Cup

Barbara
1
current volume = 6
temperature= 85
purchase date = 8-13-99
/age = 2 months

* Attibutes following an omitted one must have a type and value

* Derived attributes are precede by a slash (OMT)

Operations

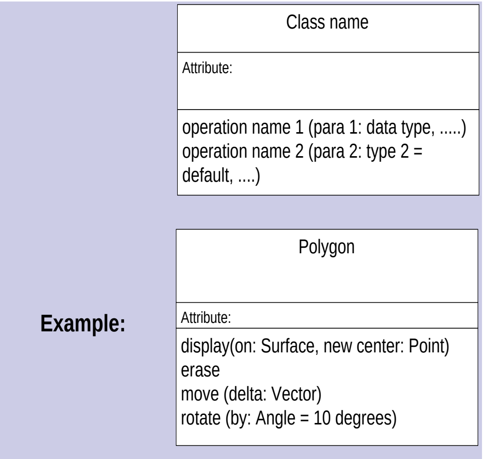
- An operation is an action performed by or on a object
- Operations are available to all instances of the class
- More than one class may have the same operation
- Think in the first person as the object and ask “What do I do?”

Exercise: Define the Operations

- What are some of the operations of the (previously identified) objects?
  - □ A convertible (e.g., light, tire, radio objects)
  - □ An airline (e.g., airplane, schedule, flight attendant objects)
  - □ A computer network (e.g., file, workstation, printer objects)

Operations Notation with Arguments

- Operations may be further detailed by following the operation name with an argument and default value.
- Operations are not shown on object diagrams



Class name

Attribute:

operation name 1 (para 1: data type, ...)
operation name 2 (para 2: type 2 = default, ...)

Example:

Polygon

Attribute:

display(on: Surface, new center: Point)
erase
move (delta: Vector)
rotate (by: Angle = 10 degrees)

Methods

- A ***method*** is the specific implementation for a class.
- For example: You can ***make a call*** (operation) for both a push button and wall crank phone, but the implementation (methods) may be different.
  - □ Push buttons() vs. crank numbers()

Polymorphism

- An operation that may have move than one method is call ***polymorphic***
- An operation is said to be polymorphic if it behaves differently on different classes.
- Every object knows how to perform its own operations
- Examples ??

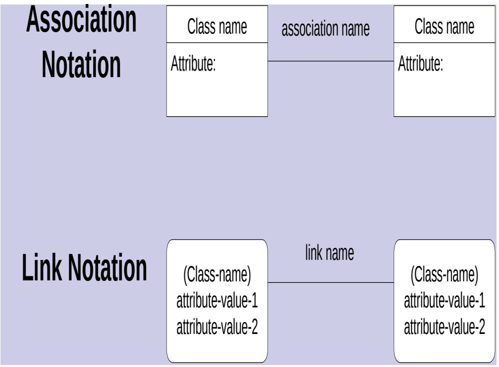
Polymorphic Operations

Polygon
Attribute:
draw erase move(new center: Point) rotate (angle: Degrees)

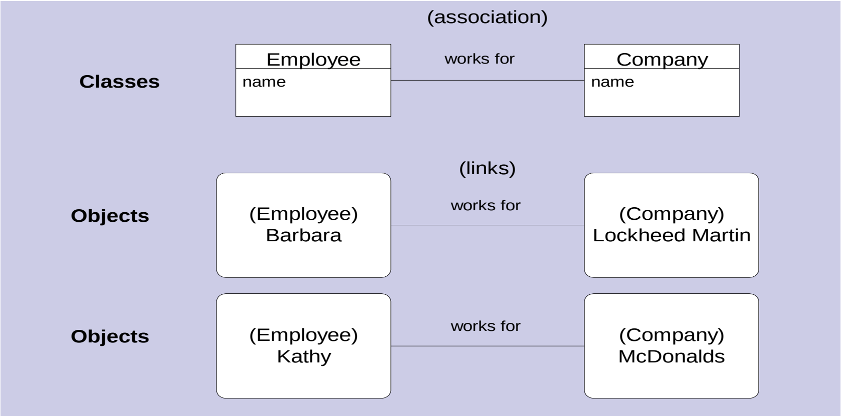
Circle
Attribute:
draw erase move(new center: Point)

Links and Associations

- Associations/links are shown by connection two or more classes/objects with a line.
- The name is usually shown above the line



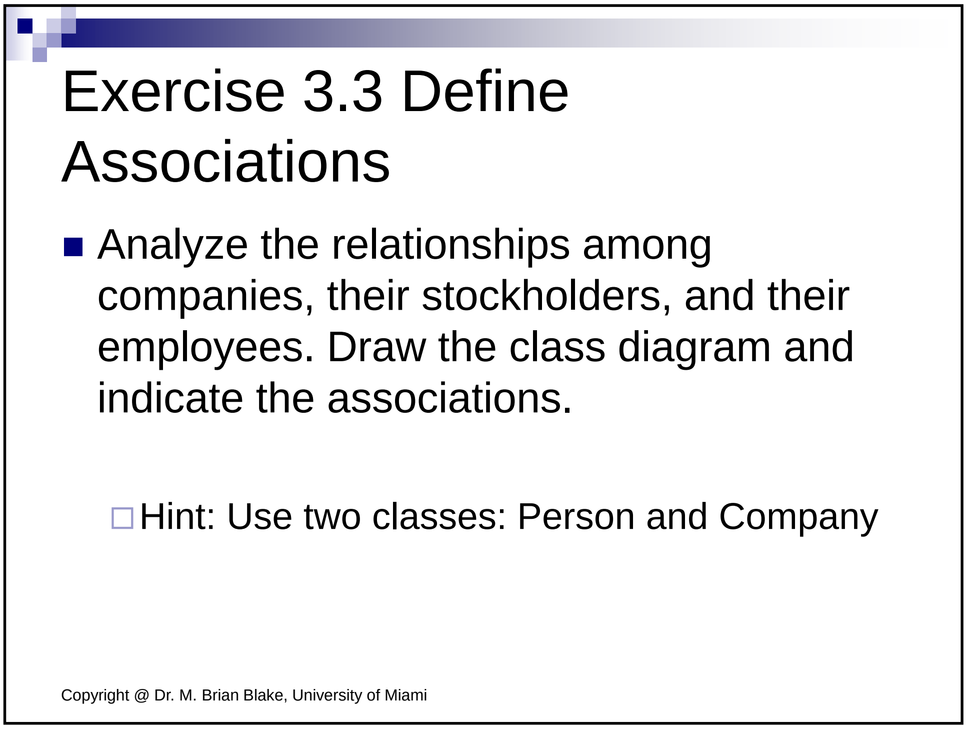
Example of Links and Associations



***** Links are instances of an association just like objects are instances of a class ****

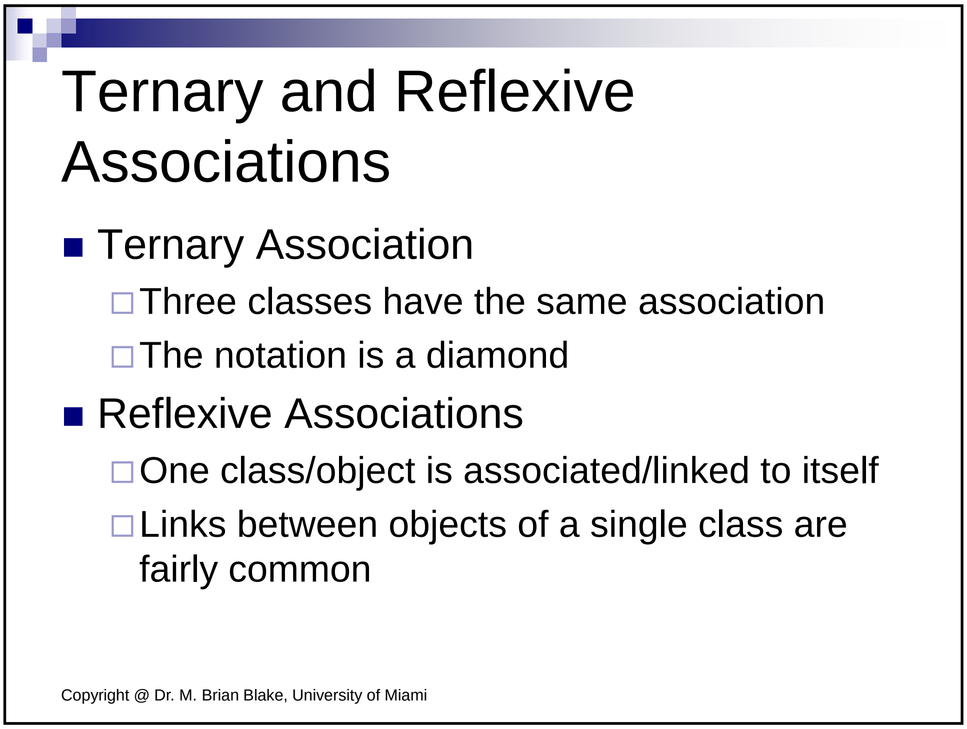
Naming Links and Associations

- Are inherently bi-directional
- Written to imply a direction
  - Can be restated to show the reverse direction
- Usually action verbs
- Read from left to right or top to bottom where possible
- An arrow can be added for clarification above the association/link name



Exercise 3.3 Define Associations

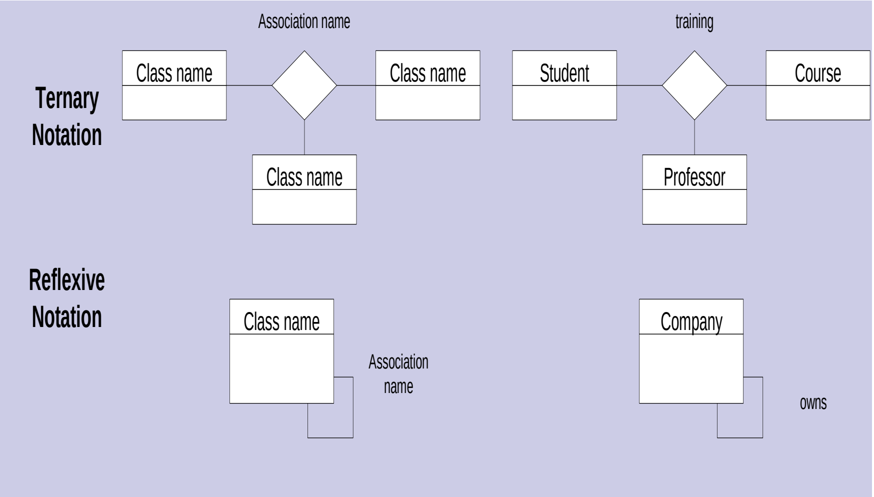
- Analyze the relationships among companies, their stockholders, and their employees. Draw the class diagram and indicate the associations.
  - □ Hint: Use two classes: Person and Company



Ternary and Reflexive Associations

- Ternary Association
  - □ Three classes have the same association
  - □ The notation is a diamond
- Reflexive Associations
  - □ One class/object is associated/linked to itself
  - □ Links between objects of a single class are fairly common

Examples of Ternary and Reflexive Links/Associations



Ternary Notation

Association name

Class name

Class name

Class name

training

Student

Course

Professor

Reflexive Notation

Class name

Association name

Company

owns

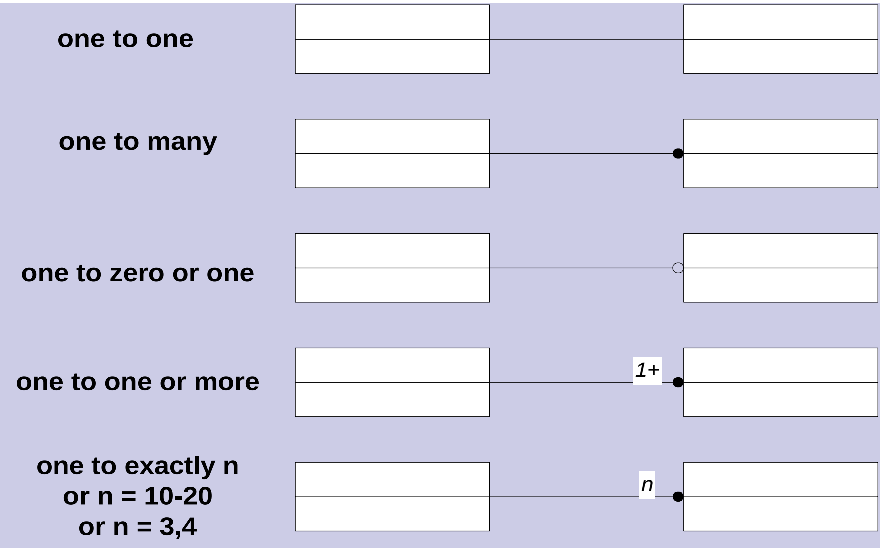
Exercise:Define Reflexive Association

- Prepare a class diagram for the following
  - □ An employee works for a company
  - □ An employee manages other employees

Multiplicity

- The number of objects of one class that relate to a single object of an associated class
- For binary associations, multiplicity decisions must be made at the end of each association

Multiplicity Notation



one to one

one to many

one to zero or one

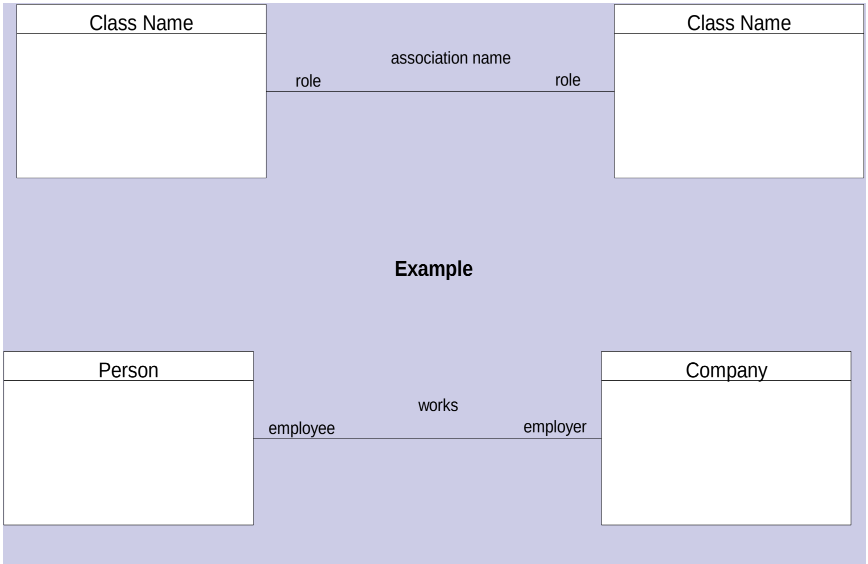
one to one or more

one to exactly n
or n = 10-20
or n = 3,4

Exercise:Multiplicity

- Analyze the relationship between people as (biological) parents and children. Draw the class diagram and indicate the associations.
- Analyze the relationships among courses, instructors, and students. A course is canceled if there are less than 5 or more than 35 students. Draw the diagram w/ associations
- Analyze the relationship between people and social security numbers. Draw the class diagram

Association Roles



Class Name

association name

role

role

Class Name

Example

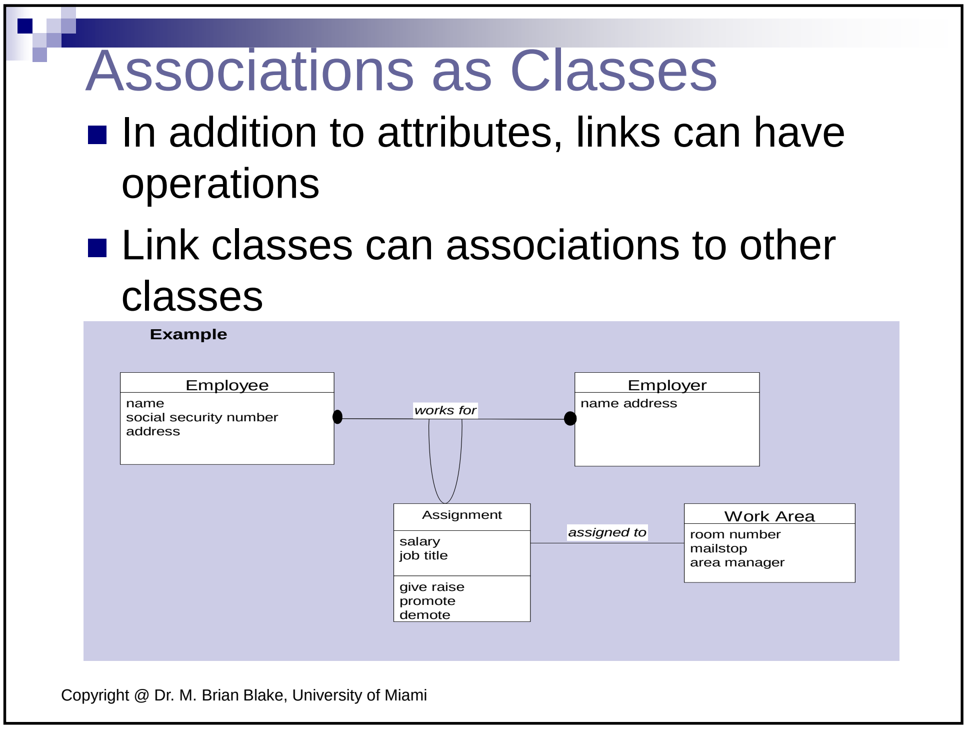
Person

works

employee

employer

Company



Associations as Classes

- In addition to attributes, links can have operations
- Link classes can associations to other classes

Example

Employee
name
social security number
address

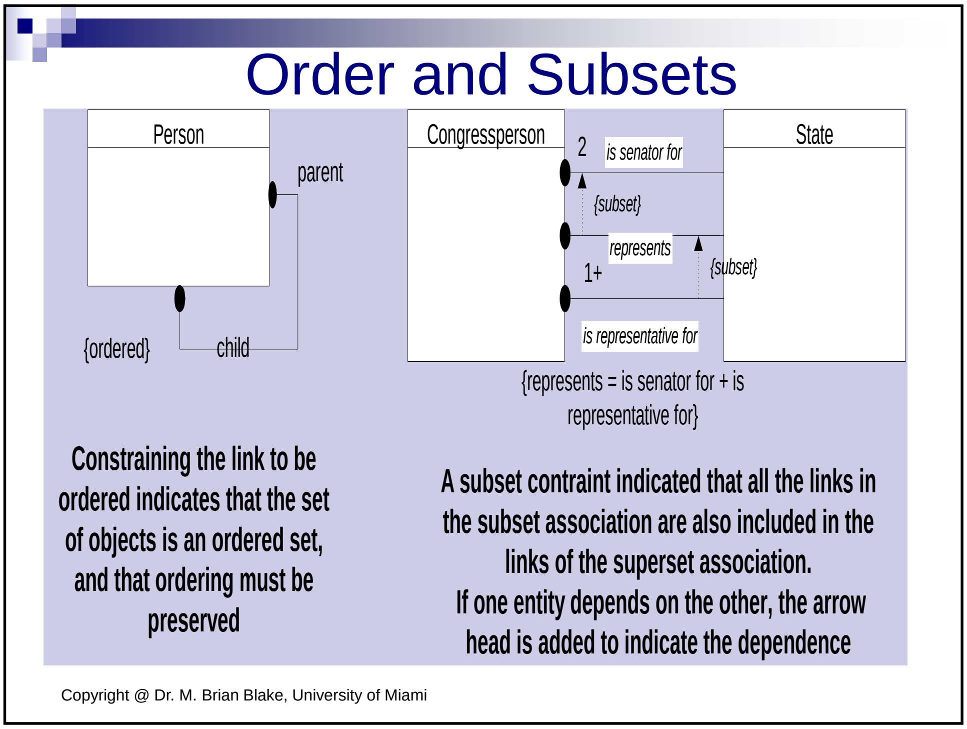
works for

Employer
name address

Assignment
salary
job title
give raise
promote
demote

assigned to

Work Area
room number
mailstop
area manager



Order and Subsets

Person

parent

{ordered}

child

Congressperson

2

is senator for

{subset}

represents

1+

{subset}

is representative for

State

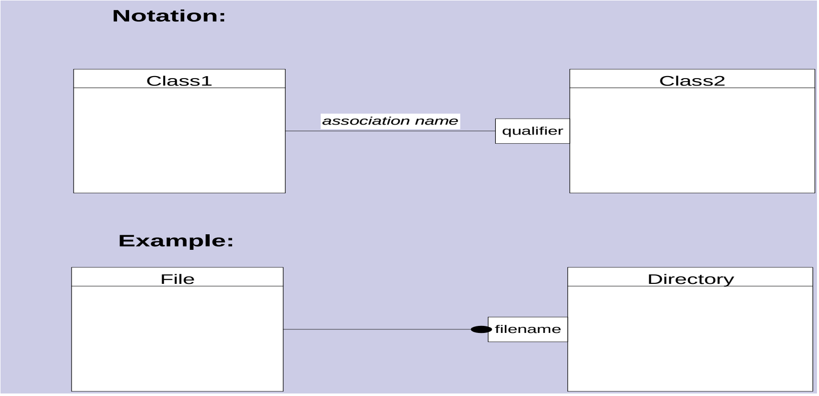
{represents = is senator for + is representative for}

Constraining the link to be ordered indicates that the set of objects is an ordered set, and that ordering must be preserved

A subset contraint indicated that all the links in the subset association are also included in the links of the superset association.
If one entity depends on the other, the arrow head is added to indicate the dependence

Qualified Associations

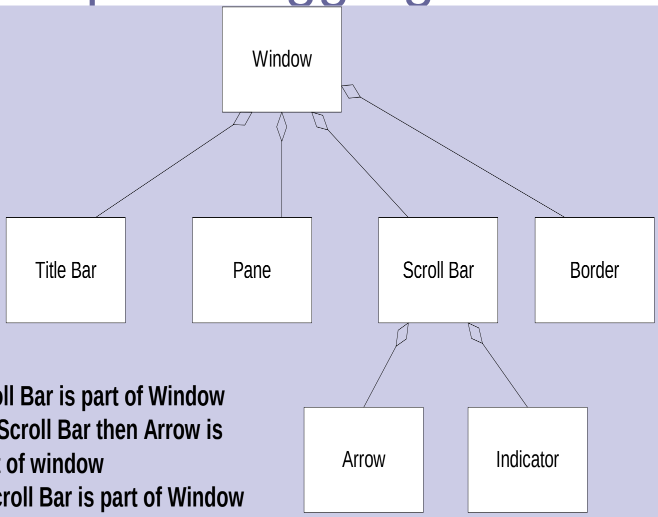
- A qualified association relates a class and a qualifier to another class.
- A qualifier is a link attribute with a special property.



Aggregation

- Aggregation ~ a special form of association, between a whole and its parts, in which the whole is composed of parts
- Aggregation is transitive
- Aggregation is asymmetric
- Usually if two classes have a “has a” relationship

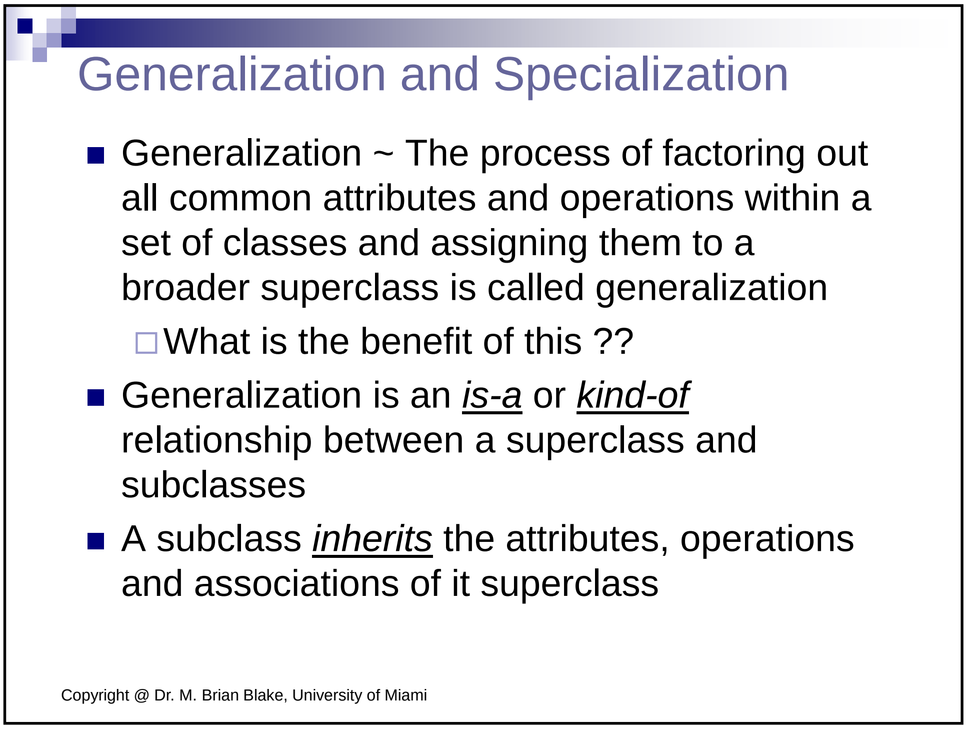
Example of Aggregation



Transitive ~ Scroll Bar is part of Window Arrow is part of Scroll Bar then Arrow is part of window
Asymmetric ~ Scroll Bar is part of Window therefore Window is not a part of Scroll Bar

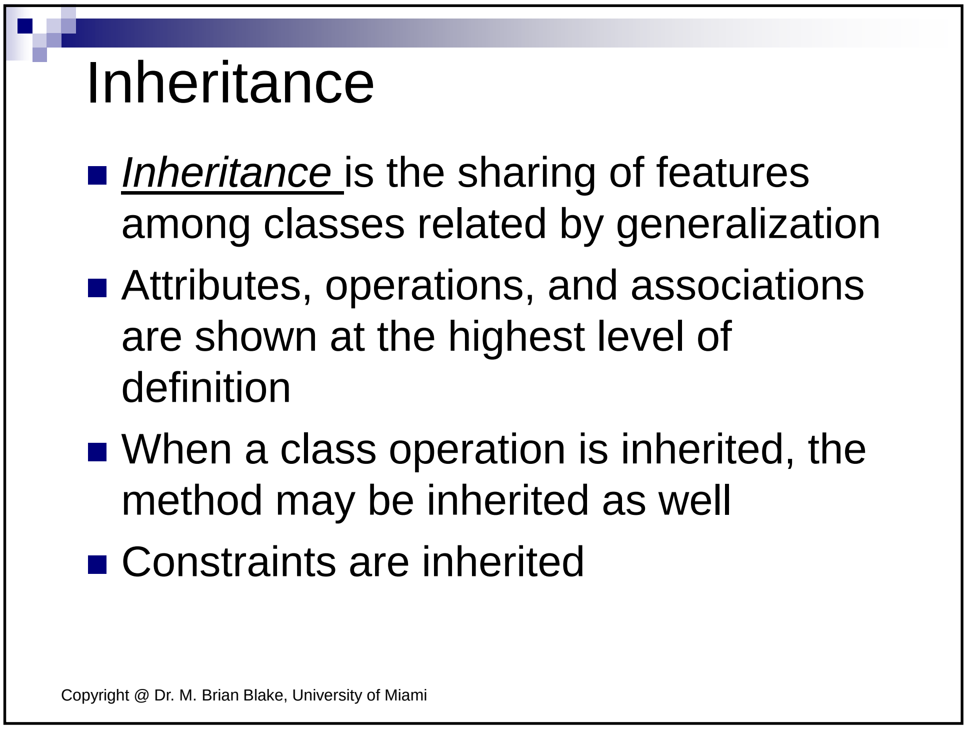
Exercise: Aggregation Problem

- Draw a class diagram for an automobile using at least two levels of aggregation.
- Draw a class diagram for a file directory structure using aggregation



Generalization and Specialization

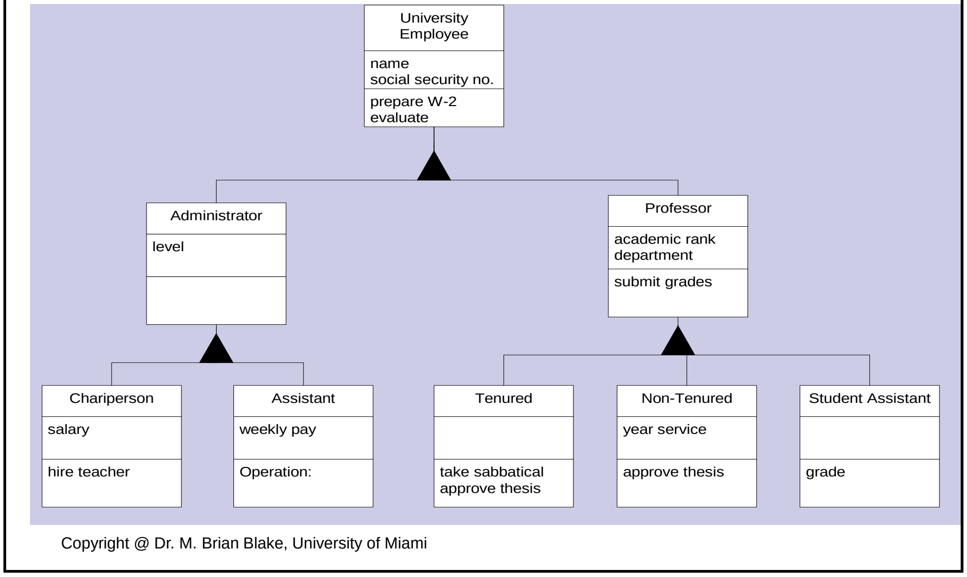
- Generalization ~ The process of factoring out all common attributes and operations within a set of classes and assigning them to a broader superclass is called generalization
  - □What is the benefit of this ??
- Generalization is an *is-a* or *kind-of* relationship between a superclass and subclasses
- A subclass *inherits* the attributes, operations and associations of it superclass



Inheritance

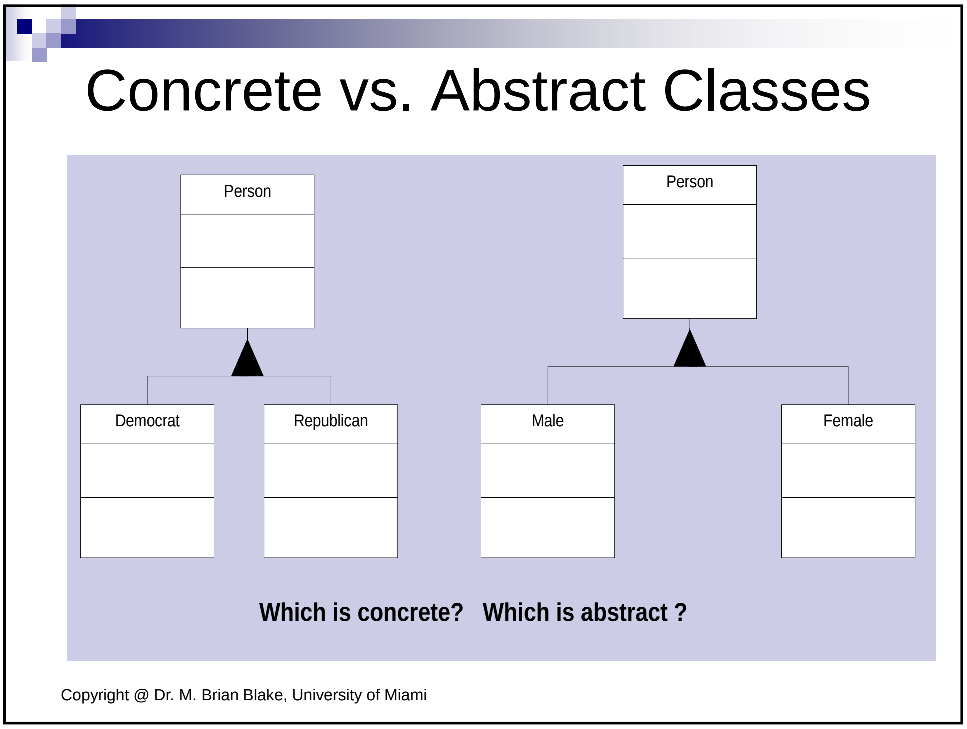
- *Inheritance* is the sharing of features among classes related by generalization
- Attributes, operations, and associations are shown at the highest level of definition
- When a class operation is inherited, the method may be inherited as well
- Constraints are inherited

Example of Multi-level Hierarchy



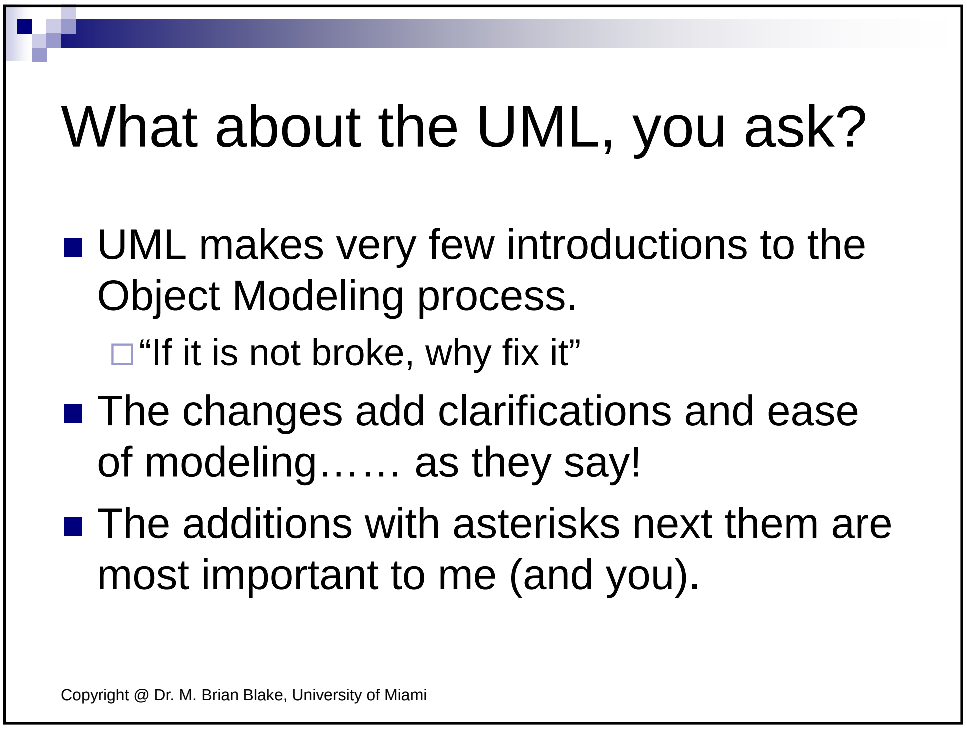
Abstract and Concrete Classes

- Abstract vs. Concrete Classes
  - □ Abstract Classes ~ when a superclass is divided into subclasses by a discriminator and a subclass exists for every possible value of the discriminator
  - □ In implementation, Abstract classes can not be instantiated
  - □ Classes that have instances are concrete classes
  - □ The lowest node in a class hierarchy must be concrete



Concrete vs. Abstract Classes

Which is concrete? Which is abstract ?



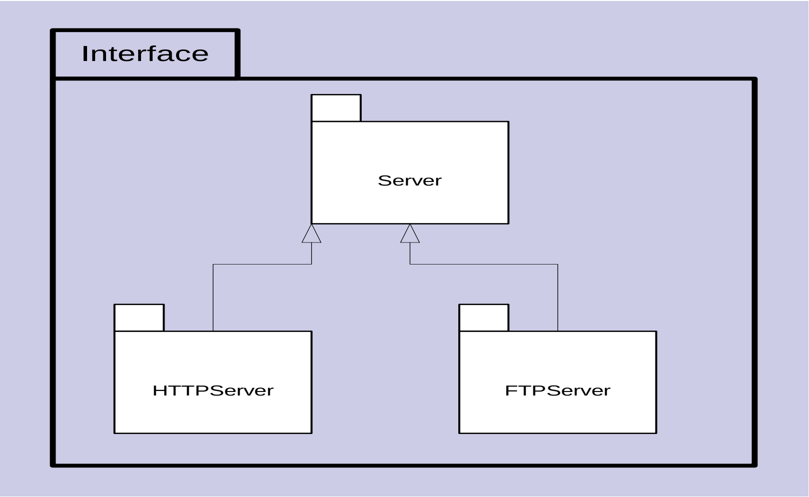
What about the UML, you ask?

- UML makes very few introductions to the Object Modeling process.
  - □“If it is not broke, why fix it”
- The changes add clarifications and ease of modeling…… as they say!
- The additions with asterisks next them are most important to me (and you).

Packages

- Package ~ a group of model elements
- Packages allow groups of like classes to be grouped and give a full system view
- Packages themselves may be nested
- Packages can be used for other models in addition to the class diagram

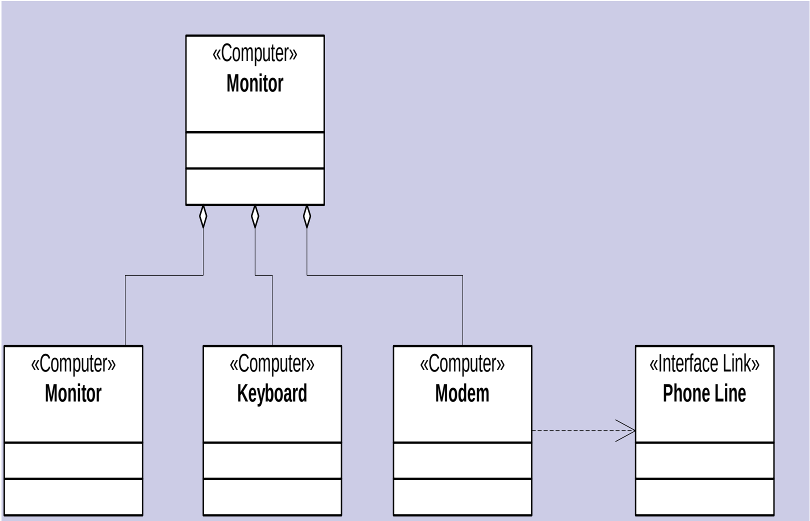
Package Notation

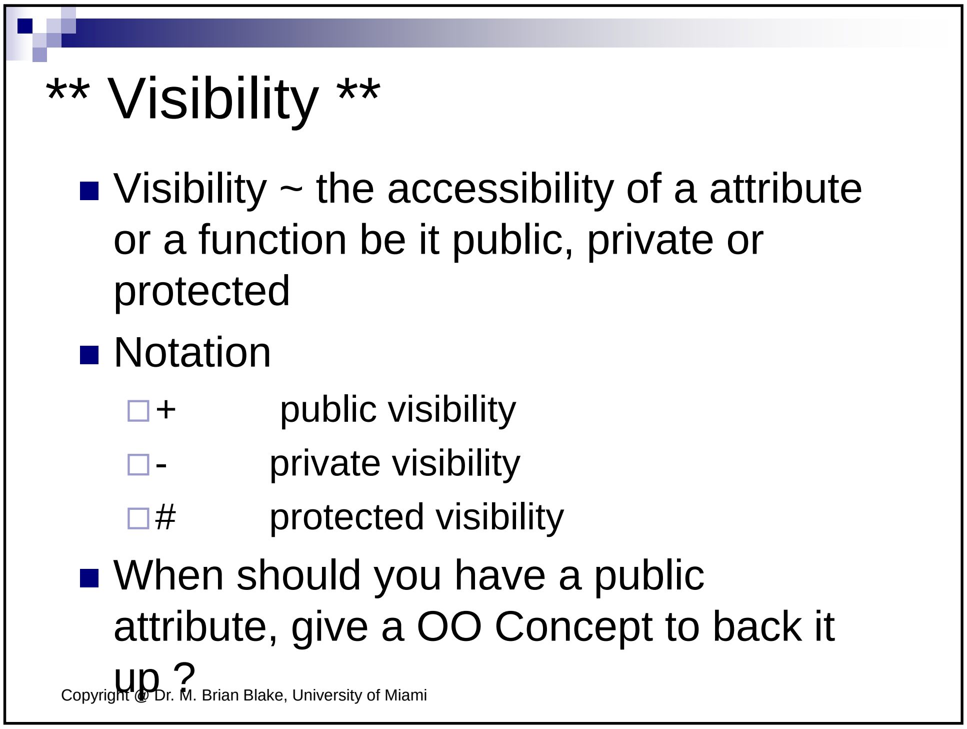


** Stereotypes **

- Stereotypes ~ notation depicted in class diagram that allow designers to specify the type of object
- Good way to specify domain, application and implementation objects
- In some cases classes can be stereotype with the name of the package that contain them

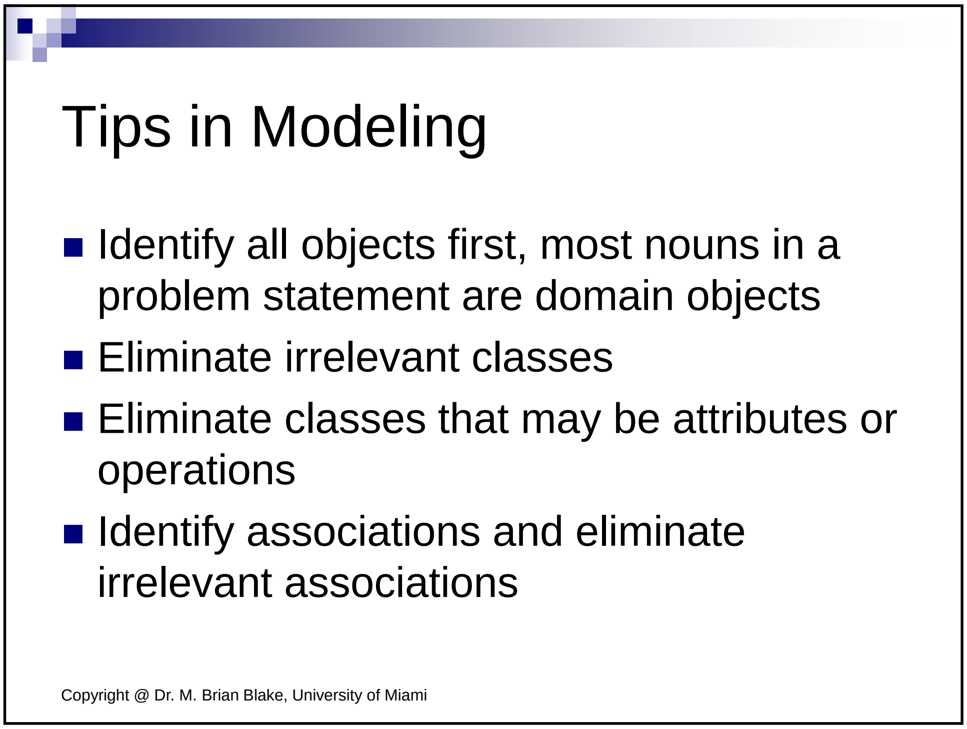
Stereotype Notation





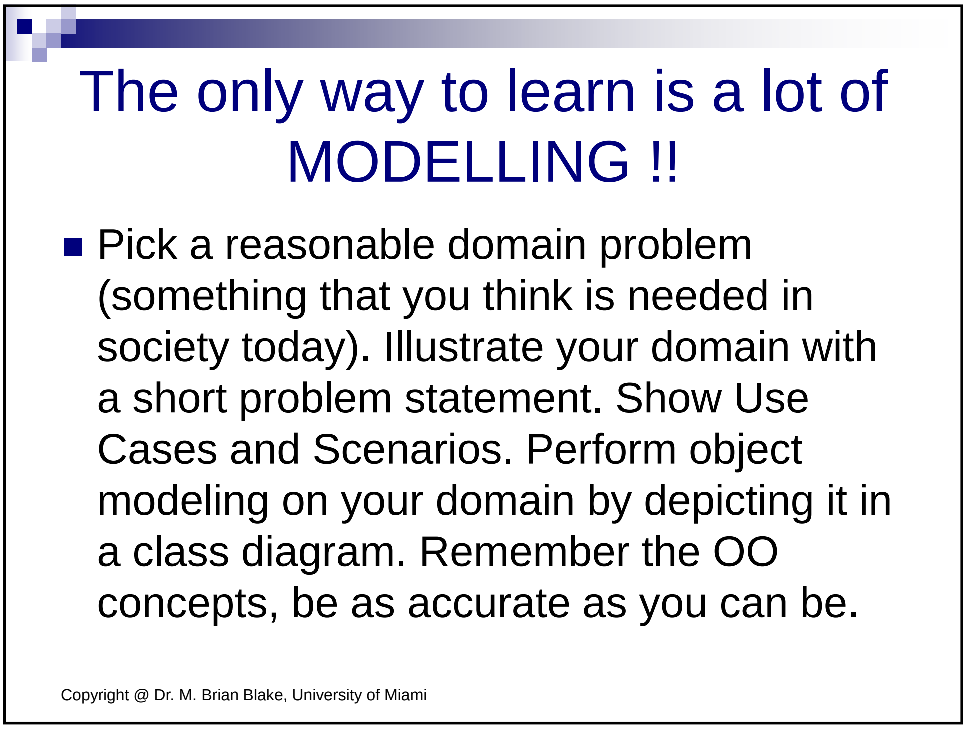
** Visibility **

- Visibility ~ the accessibility of a attribute or a function be it public, private or protected
- Notation
  - □+ public visibility
  - □- private visibility
  - □# protected visibility
- When should you have a public attribute, give a OO Concept to back it up ?



Tips in Modeling

- Identify all objects first, most nouns in a problem statement are domain objects
- Eliminate irrelevant classes
- Eliminate classes that may be attributes or operations
- Identify associations and eliminate irrelevant associations



The only way to learn is a lot of MODELLING !!

- Pick a reasonable domain problem (something that you think is needed in society today). Illustrate your domain with a short problem statement. Show Use Cases and Scenarios. Perform object modeling on your domain by depicting it in a class diagram. Remember the OO concepts, be as accurate as you can be.