Class-Based Modeling

- A class-based model contains
  - **Objects**
    - What the system will manipulate
  - **Operations** (methods or services)
    - What to be applied to the objects to effect the manipulation
  - **Relationships** (some hierarchical) between objects
  - **Collaborations** between the classes that are defined

Identifying Classes

6 selection characteristics

1. Retained information

The potential class will be useful during analysis ***only if information about it must be remembered*** so that the system can function.

2. Needed services

The potential class ***must have a set of identifiable operations*** that can change the value of its attributes in some way.

Identifying Classes

6 selection characteristics

3. Multiple attributes

During requirement analysis, the focus should be on “major” information.

A class with a single attribute may, in fact, be useful during design, but is probably better represented as an attribute of another class during the analysis activity.

4. Common attributes

A set of attributes can be defined for the potential class and these attributes apply to all instances of the class.

Identifying Classes

6 selection characteristics

5. Common operations

A set of operations can be defined for the potential class and these operations apply to all instances of the class.

6. Essential requirements

External entities that appear in the problem space and ***produce or consume information essential to the operation*** of any solution for the system will almost always be defined as classes in the requirements model.

Attributes

- Attributes define a class
  - For the same class, attributes can be very *different* in different contexts
  - *What data items fully define this class in the problem context?*

The **Player** class for professional baseball players

- Playing statistics software
  - Name, position, batting average, fielding percentage, years played, and games played, …
- Pension fund software
  - Name, average salary, pension plan options chosen, mailing address, …

Operations

- Operations define the behavior of an object:
  - Manipulate data
    - e.g., adding, deleting, reformatting, selecting
  - Perform a computation
  - Inquire the state of an object
  - Monitor an object for the occurrence of a controlling event
    - e.g., communications between objects

Case Study: Safe Home

The SafeHome security function *enables* the homeowner to *configure* the security system when it's *installed, monitors* all sensors *connected* to the security system, and *interacts* with the homeowner through the Internet, a PC, or a control panel.

Potential Class	Type

Case Study: Safe Home

The SafeHome security function *enables* the homeowner to *configure* the security system when it is *installed, monitors* all sensors *connected* to the security system, and *interacts* with the homeowner through the Internet, a PC, or a control panel.

Potential Class	Type
homeowner	role
(security) system	thing
sensor	external entity
control panel	external entity

Case Study: Safe Home

During installation, the SafeHome PC is used to *program* and *configure* the system. Each sensor is assigned a number and type, a master password is programmed for *arming* and *disarming* the system, and telephone number(s) are *input* for *dialing* when a sensor event occurs.

Potential Class	Type
homeowner	role
system	thing
sensor	external entity
control panel	external entity

Case Study: Safe Home

During installation, the SafeHome PC is used to *program* and *configure* the system. Each sensor is assigned a number and type, a master password is programmed for *arming* and *disarming* the system, and telephone number(s) are *input* for *dialing* when a sensor event occurs.

Potential Class	Type
homeowner	role
system	thing
sensor	external entity
control panel	external entity
Installation	event
number, type	thing
master password	thing
telephone number	thing
sensor event	event

Case Study: Safe Home

When a sensor event is *recognized*, the software *invokes* an audible alarm attached to the system. After a delay time that is *specified* by the homeowner during system configuration activities, the software dials a telephone number of a monitoring service, *provides* information about the location, *reporting* the nature of the event that has been detected. The telephone number will be *redialed* every 20 seconds until telephone connection is *obtained.*

Potential Class	Type
homeowner	role
system	thing
sensor	external entity
control panel	external entity
Installation	event
number, type	thing
master password	thing
telephone number	thing
sensor event	event

Case Study: Safe Home

When a sensor event is *recognized*, the software *invokes* an audible alarm attached to the system. After a delay time that is *specified* by the homeowner during system configuration activities, the software dials a telephone number of a monitoring service, *provides* information about the location, *reporting* the nature of the event that has been detected. The telephone number will be *redialed* every 20 seconds until telephone connection is *obtained.*

Potential Class	Type
homeowner	role
system	thing
sensor	external entity
control panel	external entity
Installation	event
number, type	thing
master password	thing
telephone number	thing
sensor event	event
audible alarm	external entity
delay time	thing
monitoring service	external entity

Recap: Identifying Classes

6 selection characteristics

1. Retained information
2. Needed services
3. Multiple attributes
4. Common attributes
5. Common operations
6. Essential requirements

Potential Class	Type
homeowner	role
system	thing
sensor	external entity
control panel	external entity
Installation	event
number, type	thing
master password	thing
telephone number	thing
sensor event	event
audible alarm	external entity
delay time	thing
monitoring service	external entity

Class	Selection Characteristics

Potential Class	Type
homeowner	role
system	thing
sensor	external entity
control panel	external entity
Installation	event
number, type	thing
master password	thing
telephone number	thing
sensor event	event
audible alarm	external entity
delay time	thing
monitoring service	external entity

Class	Selection Characteristics
No	Does not retain info
Yes	All 6 apply
Yes	All 6 apply
Yes	All 6 apply
No	None applies
No	Attributes of sensor class
No	Does not have multiple attributes
No	Does not have multiple attributes
Yes	All 6 apply
Yes	All 6 apply
No	Attribute of system class
No	So far it does not need service

Analyzing Class Elements (1/2)

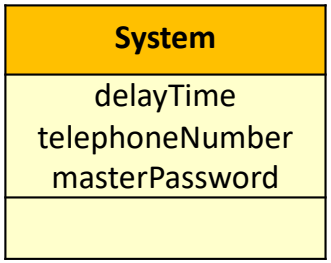
- Consider more fine-grained element types
  - **Roles** and **external entities** (actors)
    - Roles played by people who interact with the system
    - External entities that produce or consume information
  - **Organizational units** that are relevant to an application
    - e.g., team, group, division
  - **Structures**
    - Define a class of objects or related classes of objects
      e.g., sensors, computers, four-wheeled vehicles

Analyzing Class Elements (2/2)

- Consider more fine-grained element types
  - **Things**
    - Part of the information domain for the problem
    - e.g., reports, displays, letters, signals
  - **Occurrences**
    - Events occur within the context of system operation
  - **Places**
    - The context of the problem and the overall function

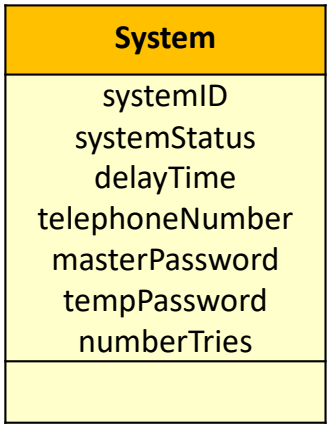
Case Study: Safe Home

- Let's look at the **system** class
  - Alarm response information
    - delayTime, telephoneNumber
  - Activation/deactivation
    - masterPassword
    - number of tries
    - temporary password
  - Identification
    - system ID, status



Case Study: Safe Home

- Let's look at the **system** class
  - Alarm response information
    - delayTime, telephoneNumber
  - Activation/deactivation
    - masterPassword
    - number of tries
    - temporary password
  - Identification
    - system ID, status



UML Class Modeling

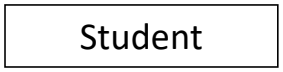
- An object-oriented modeling language developed in 1997
  - Models structure (static) and behavioral (dynamic) aspects of a system
  - Semi-formal: UML 2.0 added much more formality
  - Process-independent: can be used with a variety software development process models
  - Customizable and extensible

Abstraction Levels

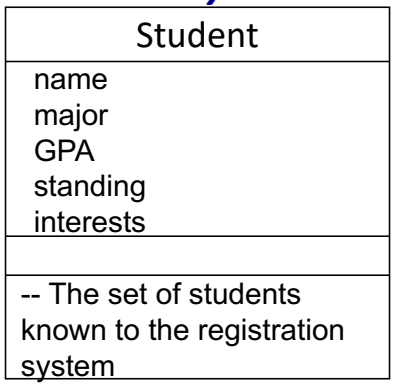
- Three perspectives for class models
  - Analysis
    - Represents concepts in the domain
    - Drawn with no regard for implementation (language independent)
  - Specification
    - Focus on interfaces not on how implementation is broken into classes
  - Implementation
    - A blue-print for coding
    - Direct code implementation of each class in the diagram

Student Records Management System

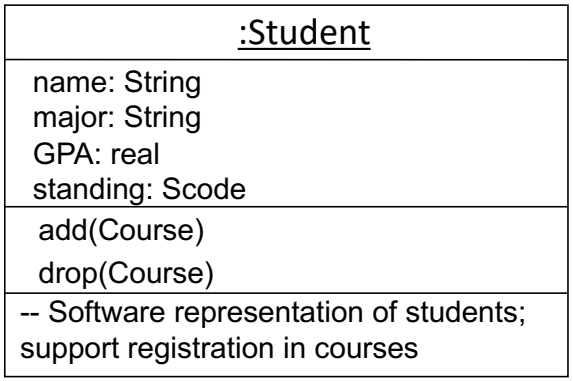
{Joe, Sue, Mary, Frank, Tim, …}



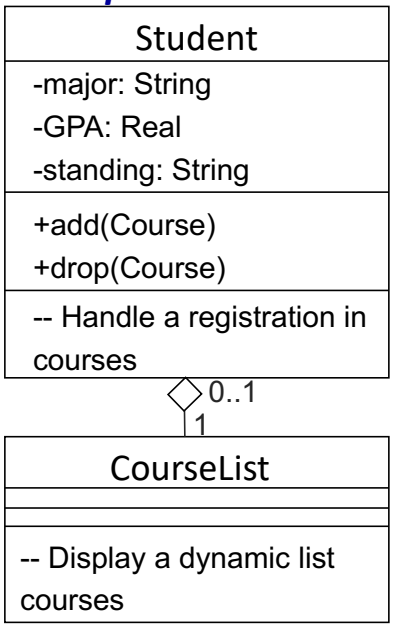
Analysis



Specification

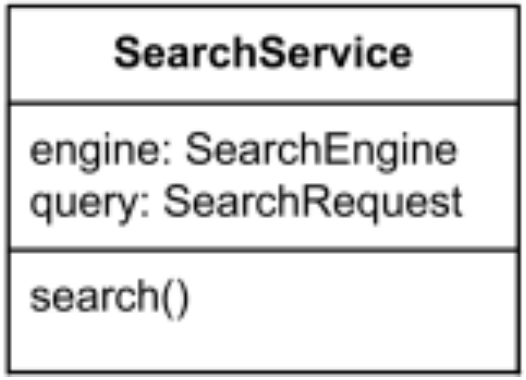


Implementation



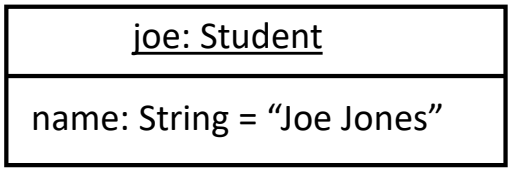
Classes in UML Diagrams

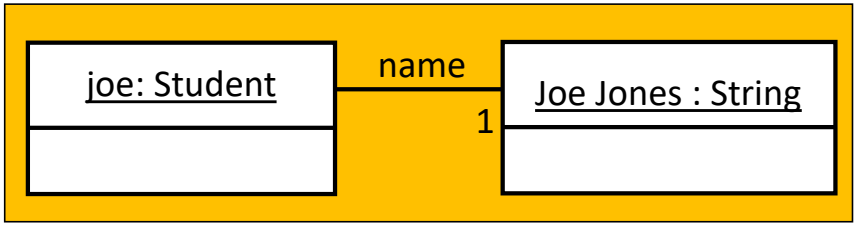
- An abstraction which describes a collection of objects sharing some commonalties
- Syntax
  - Name: noun, singular
    - centered, bold, first letter capitalized
  - Attribute
    - left justified, lower cases
  - Operations
  - Visibility
    - \+ public
    - \- private
    - # protected



Attributes

- An attribute can be defined for individual objects or a class of objects
  - Static: if defined for a class, every object in the class has that attribute (place holder)
- An attribute relates an object to some other object

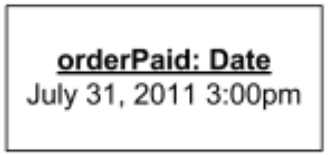


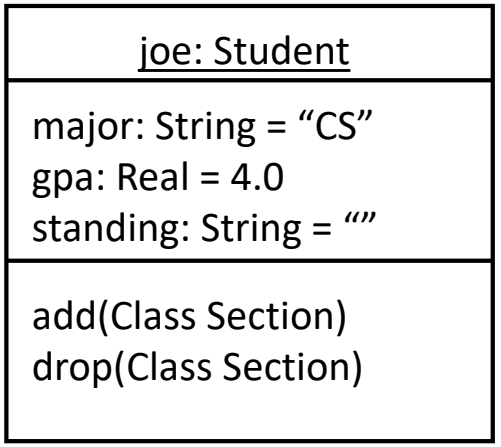


Objects

- Object is an instance of a class
  - Fundamental building blocks of object-oriented systems
  - Instance name and class path are separated by a ":"
  - Operation syntax:
    name (params) : return type
- An instance may have some value
  - Instance orderPaid of the Date class has

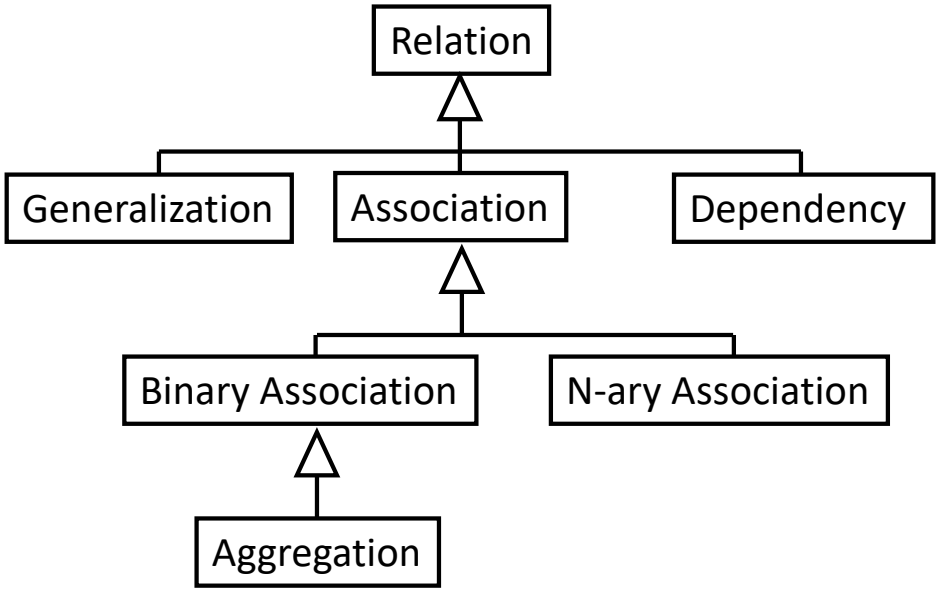
  the value July 31, 2011 3:00 pm





Type of Relationships in Class Diagrams

- Class diagrams show relationships between classes.



Associations

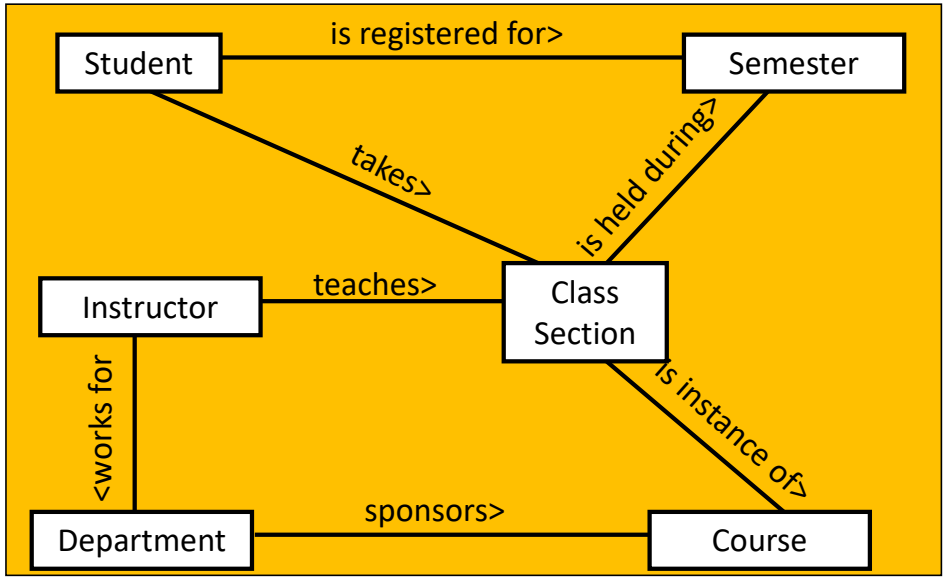
- An association is a *structural relationship* that specifies a connection between classes
- Classes A and B are associated if:
  - An object of class A sends a message to an object of B
  - An object of class A creates an instance of class B
  - An object of class A has an attribute of type B or collections of objects of type B
  - An object of class A receives a message with an argument that is an instance of B (maybe…)
    - Depends whether it “uses” that argument

Associations

- Associations
  - Links are instances of associations
  - Association names are typically verb phrases (in lower case)
  - The name should include an arrow indicating the direction in which the name should be read
  - Often interaction diagrams are useful for modeling objects

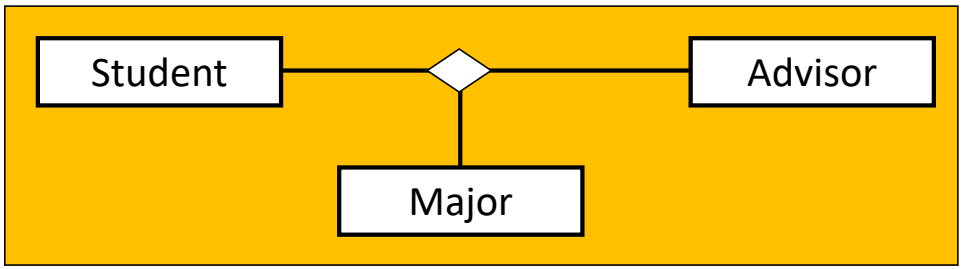
Associations

- A solid line connecting two classes



N-ary Associations

- Associations can connect more than one class

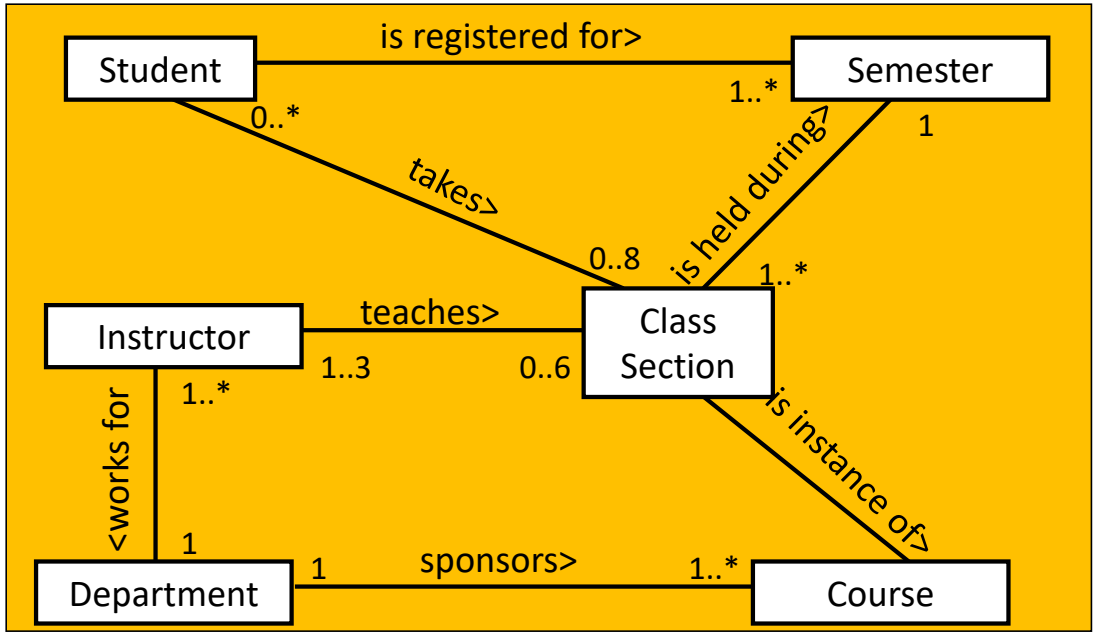


Multiplicity

- How many objects from two classes are linked?
  - An exact number: indicated by the number
  - A range: two dots between a pair of numbers
  - An arbitrary number: indicated by * symbol
  - (Rare) A comma-separated list of ranges

  - e.g., 1 1..2 0..* 1..* * (same as 0..*)
  - Implementing associations depends on multiplicity

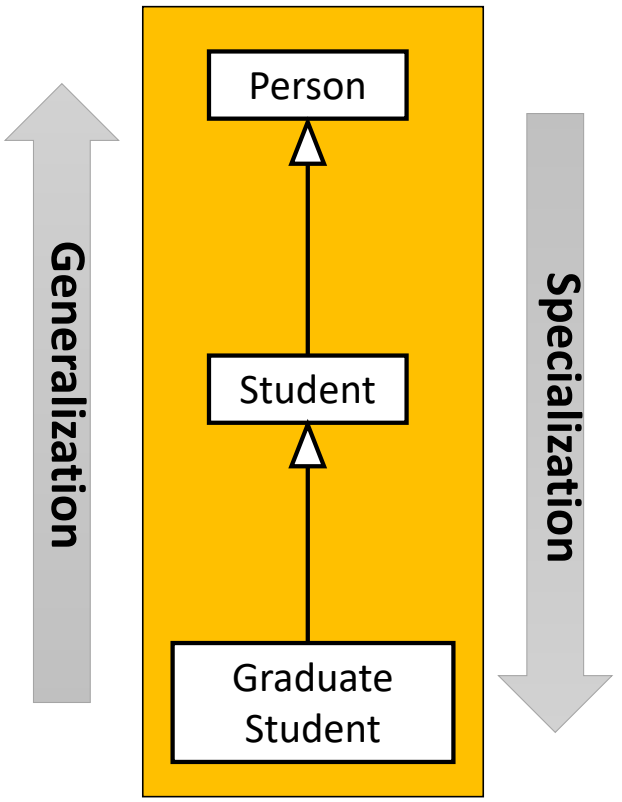
Multiplicity



Generalization

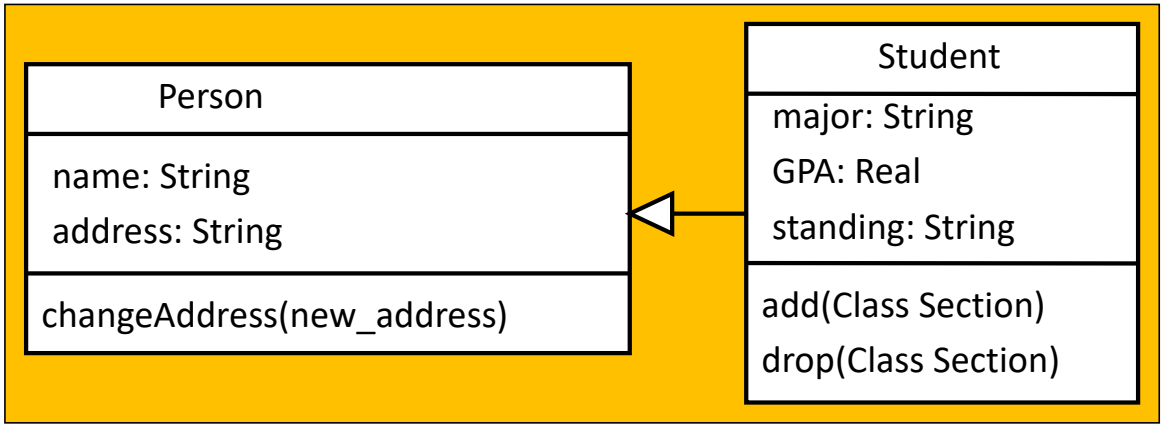
- Generalization is an association between classes
  - A subclass is connected to a superclass by an arrow with a solid line with a hollow arrowhead.

- From an analysis perspective, it represents generalization/specialization:
  - Specialization is a subset of the generalization



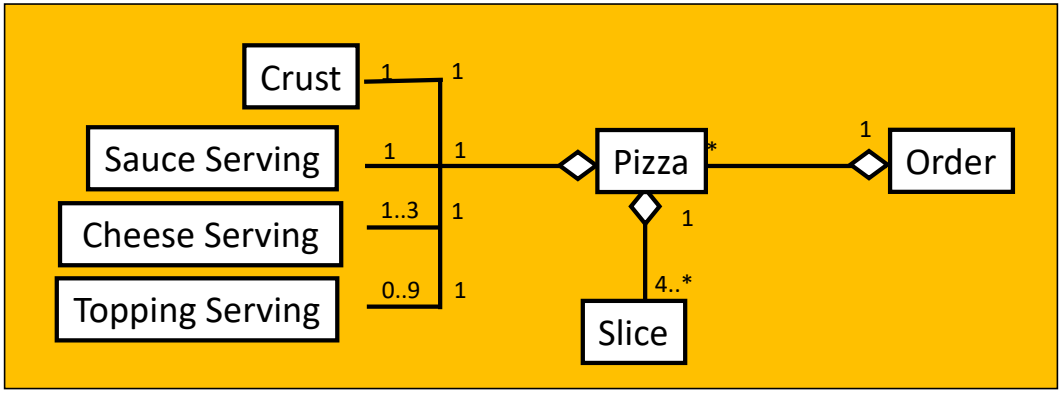
Generalization

- Generalization represents implementation inheritance
  - You model “inheritance” early, but not implement it at the conceptual level



Aggregation

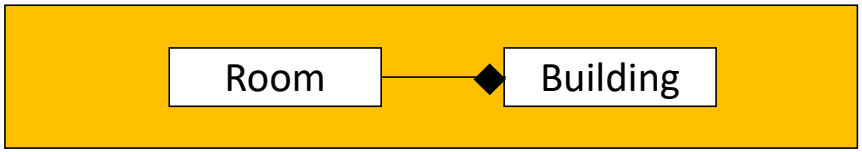
- Aggregation: is a special kind of association that means “part of”
- Aggregations should focus on a single type of composition (physical, organization, etc.)



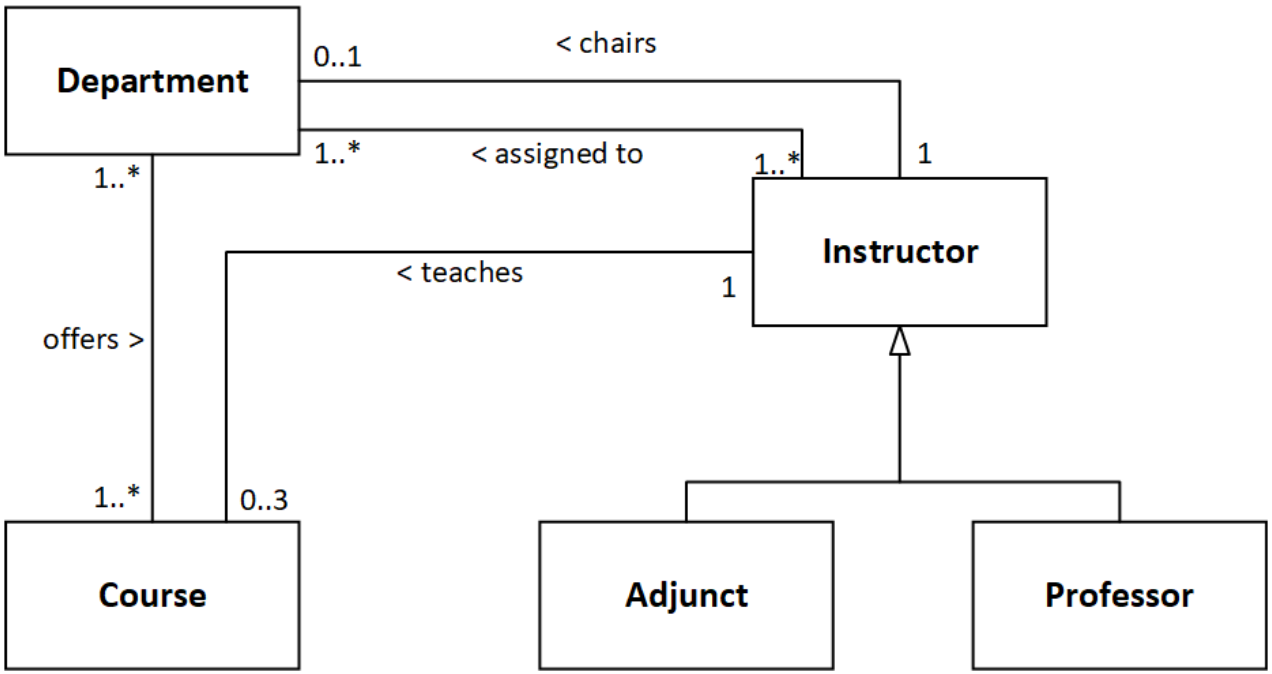
Coming up: Composition (very similar to aggregation)

Composition

- Very similar to aggregation:
  - Think of composition as a stronger form of aggregation
  - **Composition means something is a part of the whole, but cannot survive on it's own**



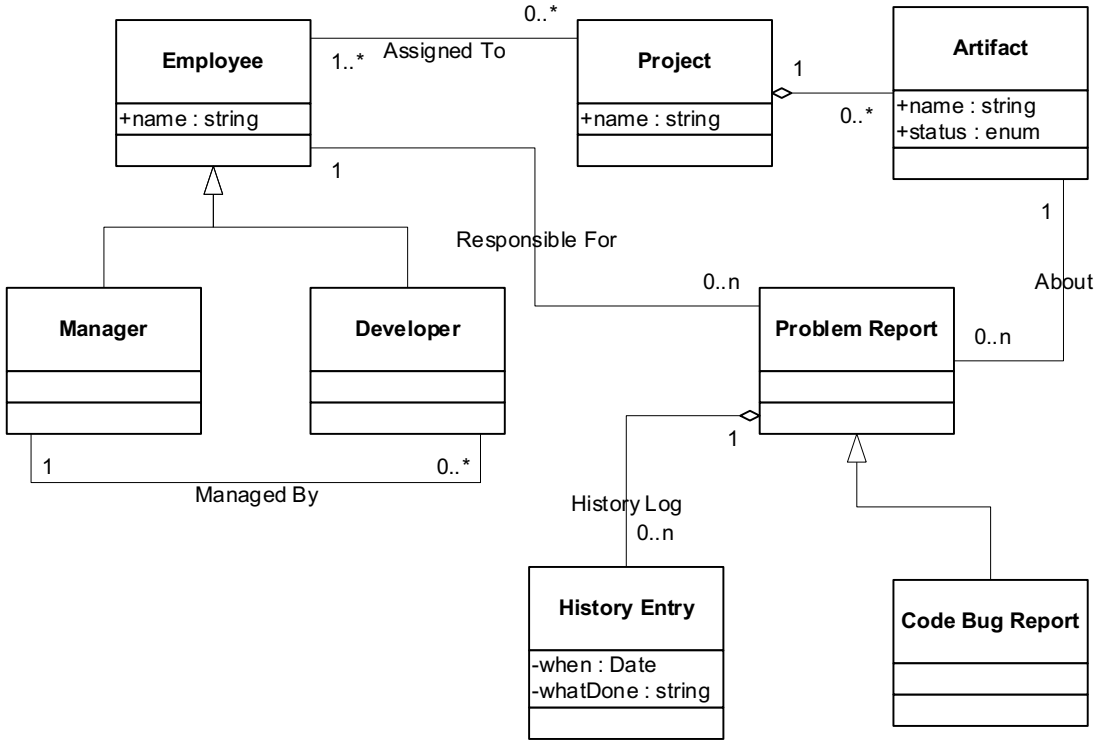
Class Diagram



Another Example

- Problem Reporting Tool: a CASE tool for storing and tracking problem reports
  - Employees are assigned to a project
  - A manager may add new artifacts and assign problem reports to developers
  - Each report contains a problem description and a status
  - Each problem can be assigned to someone
  - Problem reports are made in one of the “artifacts” of a project

Class Diagram



References

- Prof. Fengjun Li's EECS 448 Fall 2015 slides

- This slide set has been extracted and updated from the slides designed to accompany *Software Engineering: A Practitioner's Approach, 8/e* (McGraw-Hill 2014) by Roger Pressman