Object Oriented Analysis and Design Using the UML

Cont: Object Oriented Analysis

Goals of OO analysis

- What are the two main goals of OO analysis?
  1) Understand the customer's requirements
  2) Describe problem domain as a set of classes and relationships

OO domain modeling with UML class diagrams and CRC cards

What is a Domain Model?

- Illustrates meaningful conceptual classes in problem domain
- Represents real-world concepts, not software components
- A diagram (or set of diagrams) which represents real world *domain objects*
  - '*conceptual classes*'
- *Not a set of diagrams describing software classes, or* software objects with responsibilities

What Domain Model should it show?

objects, classes
relation between classes

- conceptual classes
- associations between conceptual classes
- attributes of conceptual classes

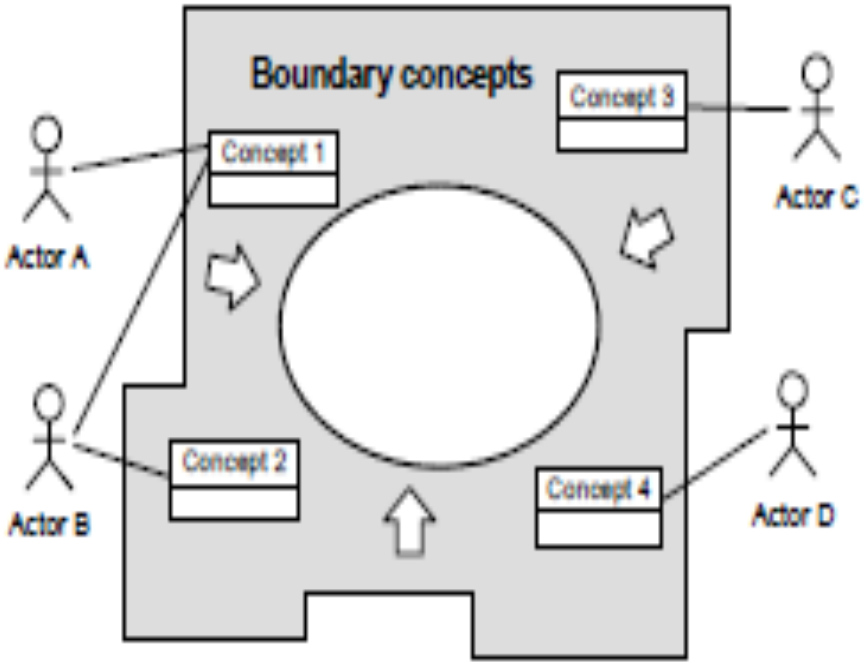
Building the Domain Model

A useful strategy for building a domain model is to start with:

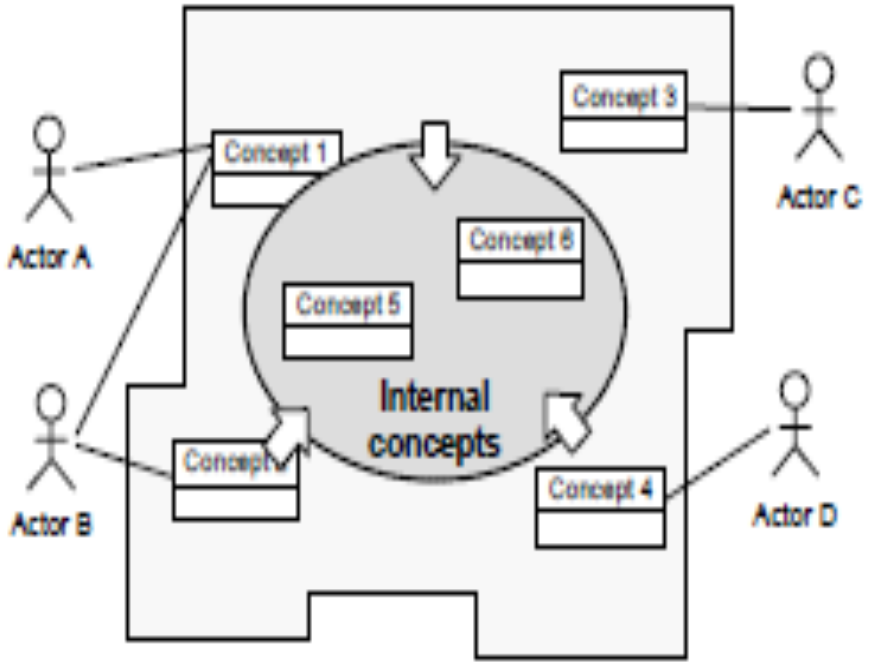
- the "boundary" concepts that interact directly with the actors
- and then identify the internal concepts

قبل ما نقرب بيه بينه

Step 1: Identifying the boundary concepts



Step 2: Identifying the internal concepts



Steps to create a Domain Model

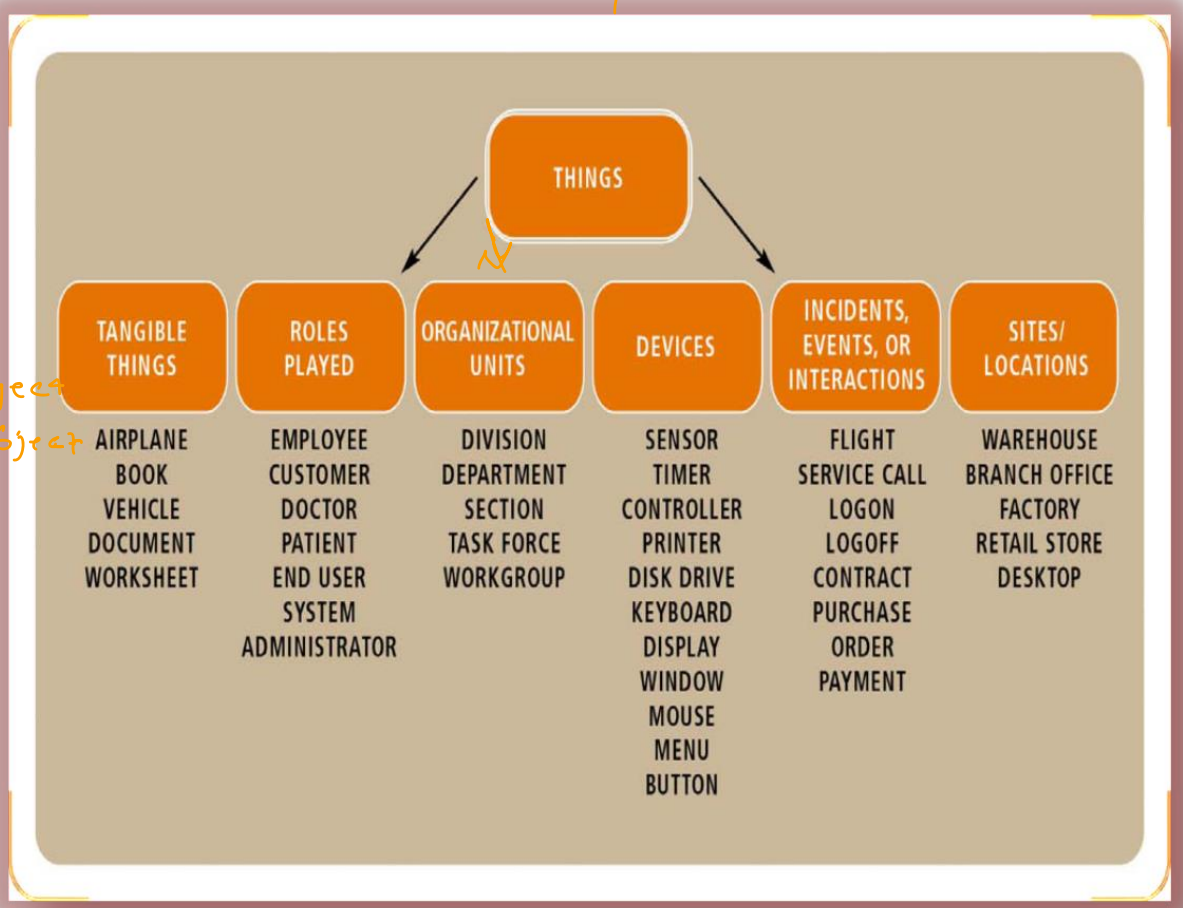
- Identify Candidate Conceptual classes
- Draw them in a Domain Model
- Add associations necessary to record the relationships that must be retained
- Add attributes necessary for information to be preserved

Identify conceptual classes

- Three strategies to find conceptual classes
  - Reuse or modify existing models
    - There are published, well-crafted domain models and data models for common domains: inventory, finance, health..
    - Books: *Analysis patterns* by Martin Fowler, *Data Model Patterns* by David Hay, *Data Model Resource Book* by Len Silverston
  - Use a category list
  - Identify noun phrases

Use a category list

- Finding concepts using the concept category list :
  - Physical objects: register, airplane, blood pressure monitor
  - Places: airport, hospital
  - Catalogs: Product Catalog
  - Transactions: Sale, Payment, reservation



Identify conceptual classes from noun phrases

- Finding concepts using Noun Phrase identification in the textual description of the domain :

  - Noun Phrase Identification [Abbot 83]
    - Analyze ***textual description*** of the domain
    - Identify nouns and noun phrases (indicate candidate classes or attributes)
    - Caveats:
      - Automatic mapping isn't possible
      - Textual descriptions are ambiguous! (different words may refer to the same class)

- Noun phrases may also be attributes or parameters rather than classes:

  - If it stores state information or it has multiple behaviors, then it's a class
  - If it's just a number or a string, then it's probably an attribute

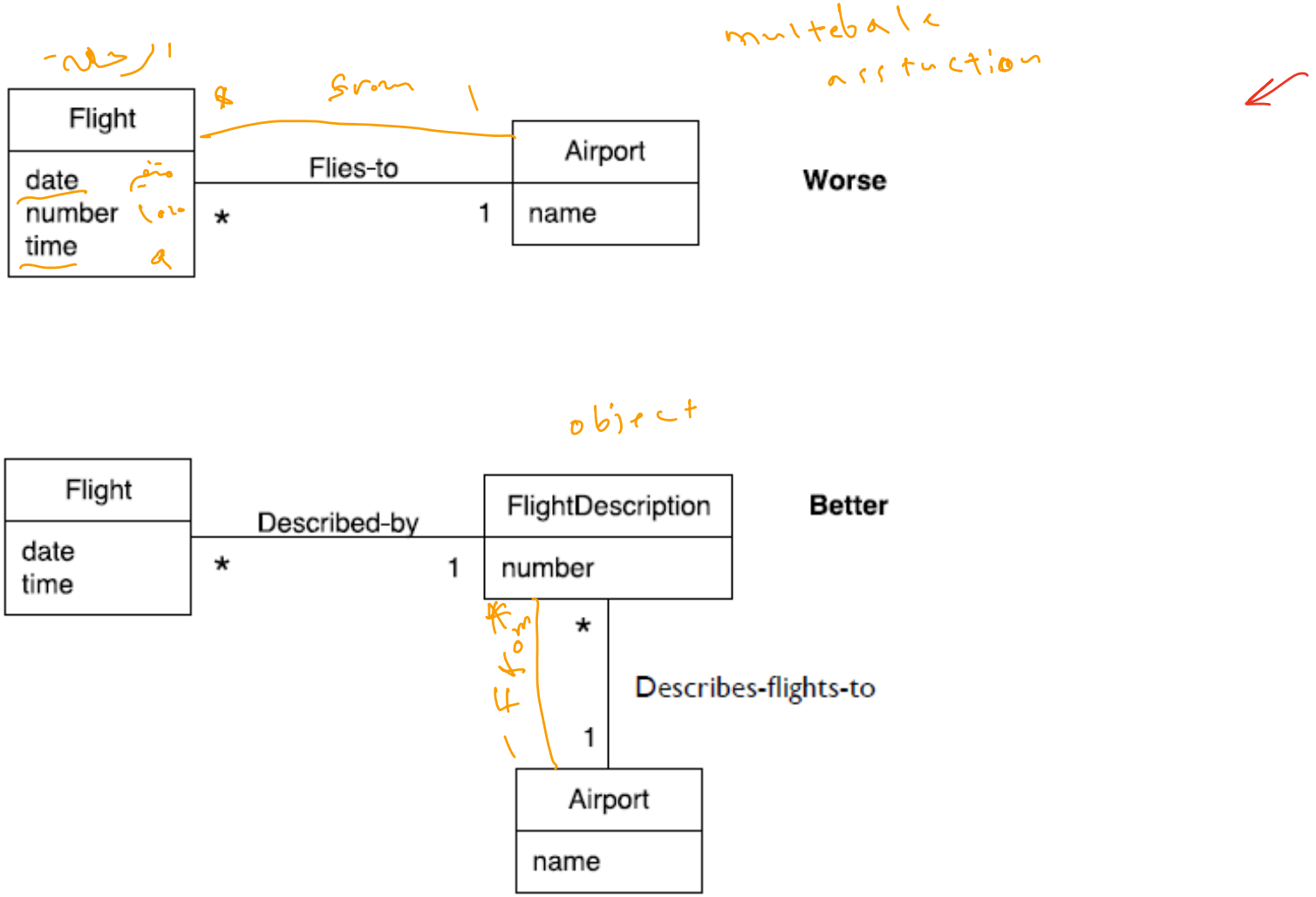
Identifying objects

- Look for **nouns** in the SRS (System Requirements Specifications) document
- Look for **NOUNS** in use cases descriptions
- A **NOUN** may be
  - Object
  - Attribute of an object

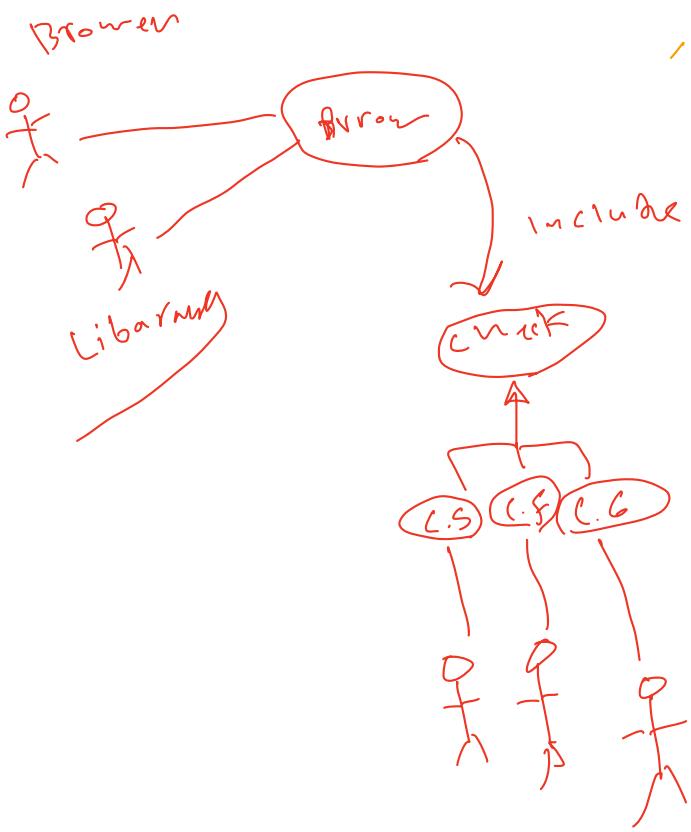
Identifying Operations 'methods'

- Look for verbs in the SRS (System Requirements Specifications) document
- Look for **VERBS** in use cases descriptions
- A **VERB** may be
  - translated to an **operation** or set of operations
  - A method is the code implementation of an operation.

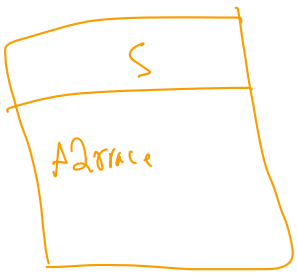
Example



from Ch 9 *Applying UML & Patterns (Larman 2004)*

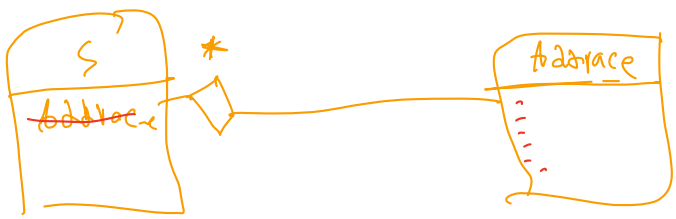


- student
- staff
- Guest



50,000 = Riyadh

100,000 bites for Riyadh

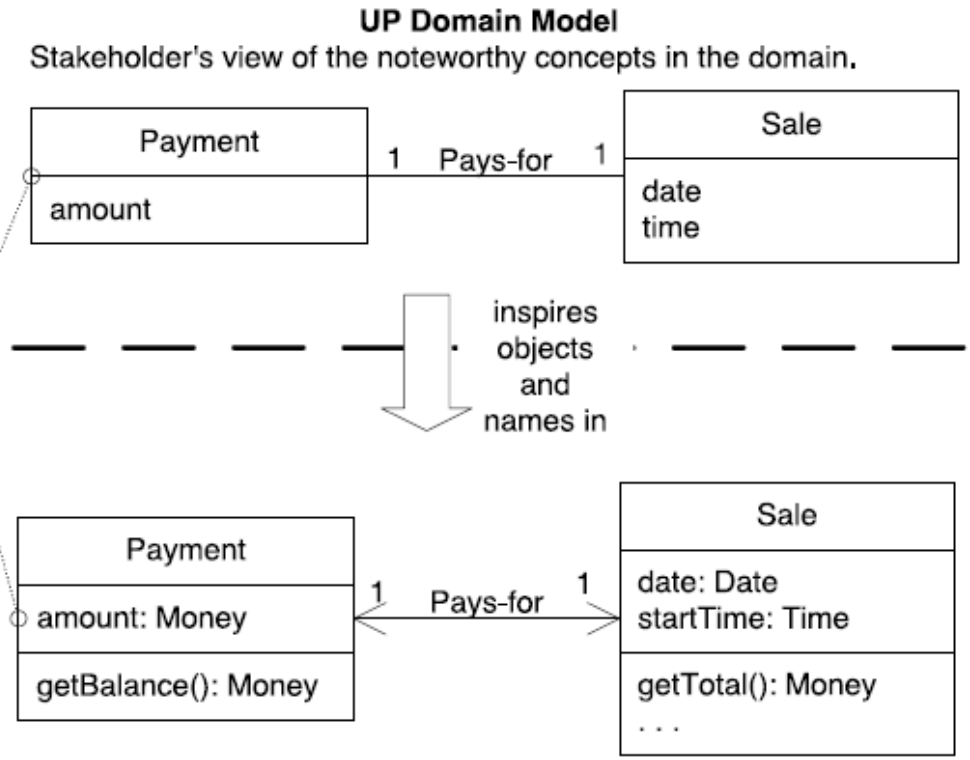


Domain versus Design Models

A Payment in the Domain Model is a concept, but a Payment in the Design Model is a software class. They are not the same thing, but the former *inspired* the naming and definition of the latter.

This reduces the representational gap.

This is one of the big ideas in object technology.



The object-oriented developer has taken inspiration from the real world domain in creating software classes.

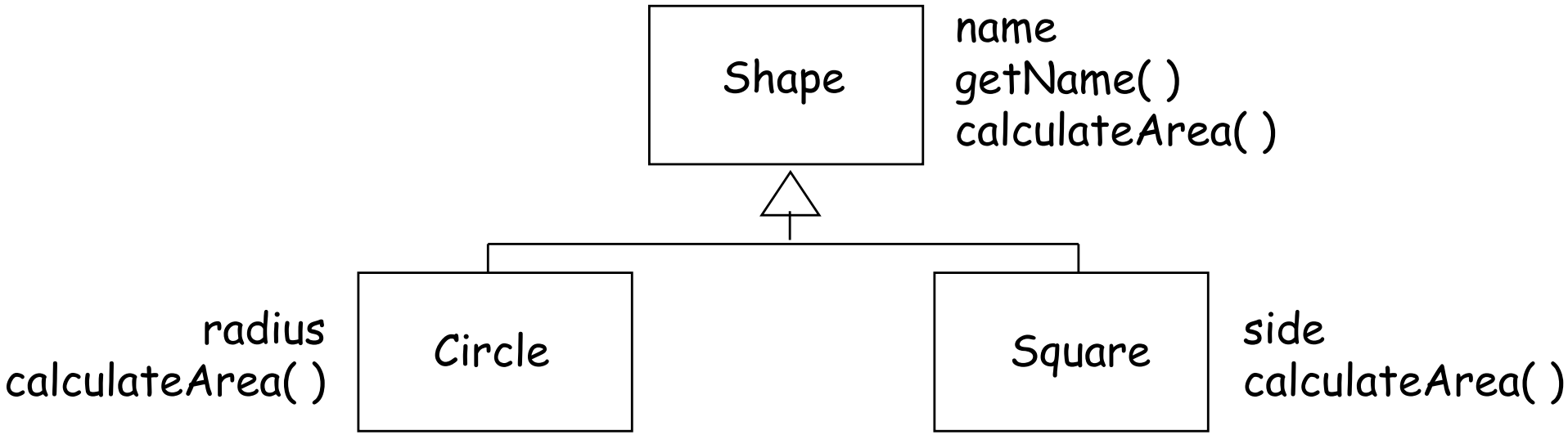
Therefore, the representational gap between how stakeholders conceive the domain, and its representation in software, has been lowered.

from Ch 9 *Applying UML & Patterns (Larman 2004)*

Basic Concepts of Object Orientation

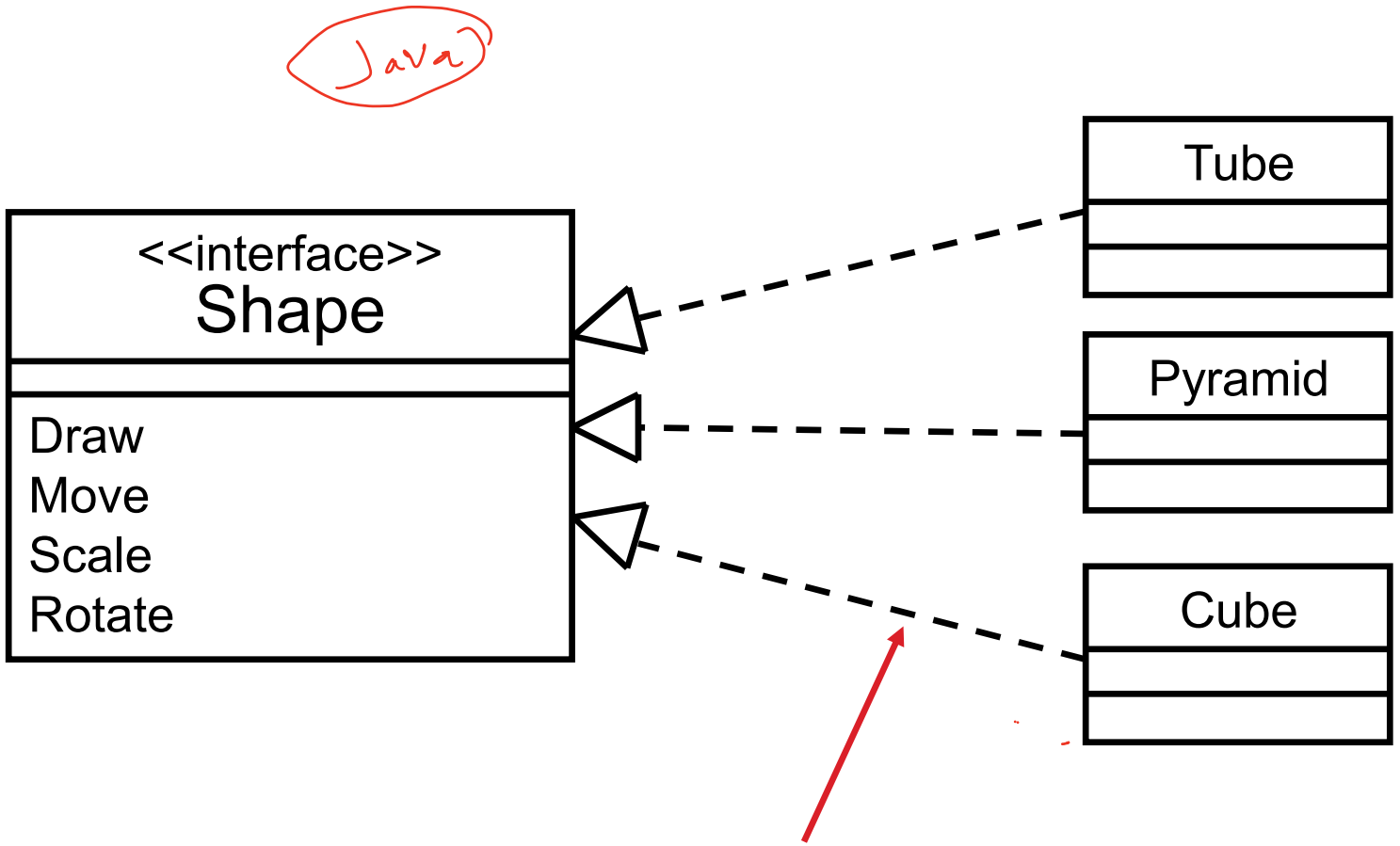
- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Relationships

What is Polymorphism?



What is an Interface?

- Interfaces formalize polymorphism



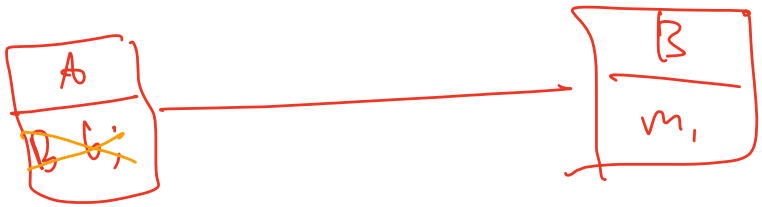
(stay tuned for realization relationships)

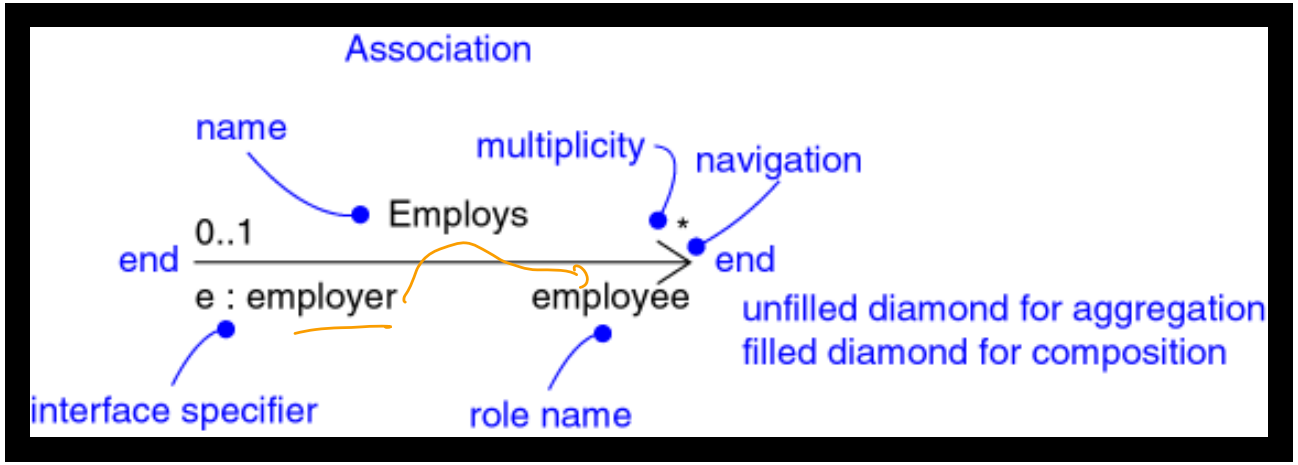
Basic Concepts of Object Orientation

- Object
- Class
- Attribute
- Operation
- Interface (Polymorphism)
- Relationships

Relationships

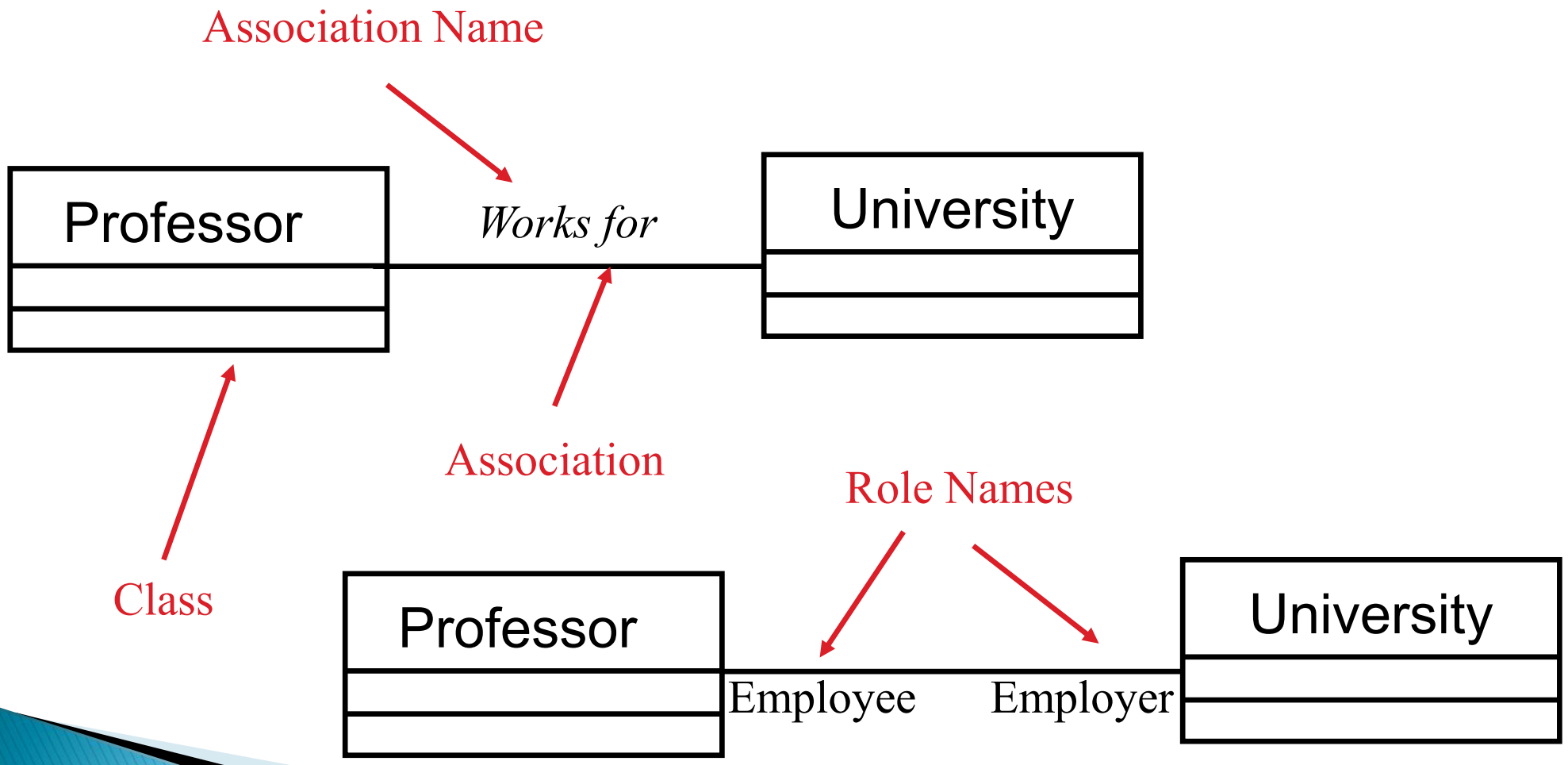
- Association
  - Aggregation
  - Composition
- Dependency
- Generalization
- Realization





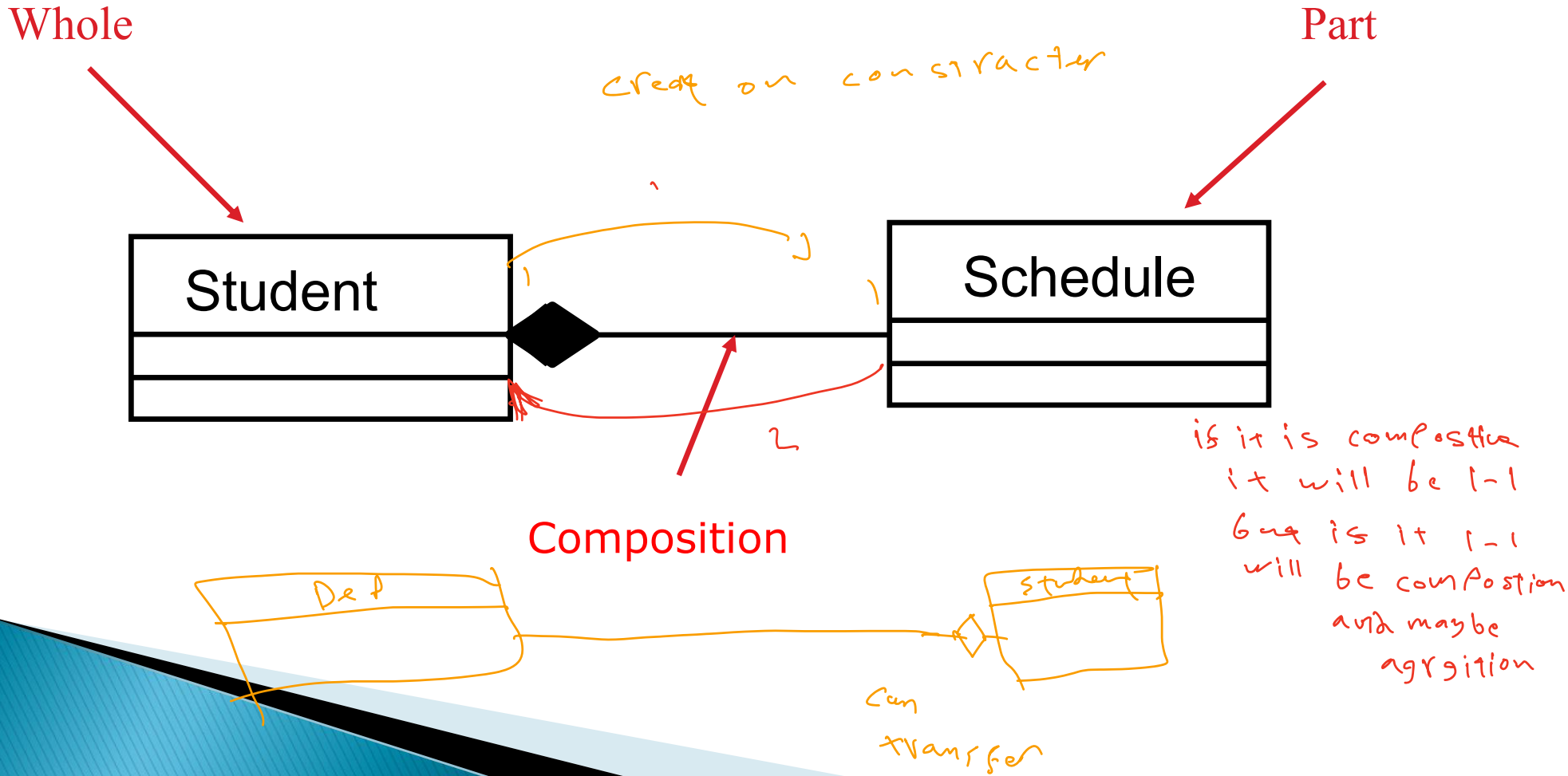
Relationships: Association

- Models a semantic connection among classes



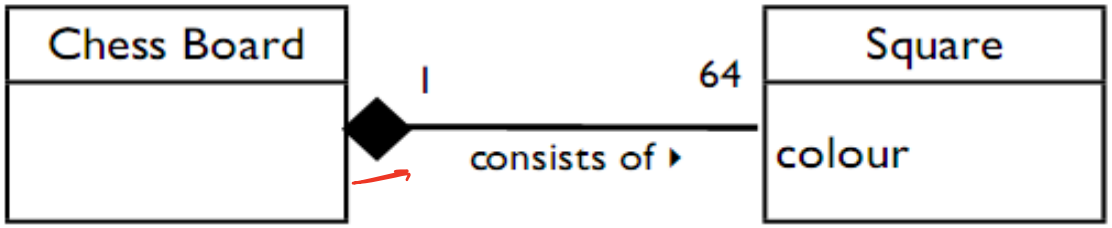
Relationships: Composition

- A special form of association that models a whole–part relationship between an aggregate (the whole) and its parts



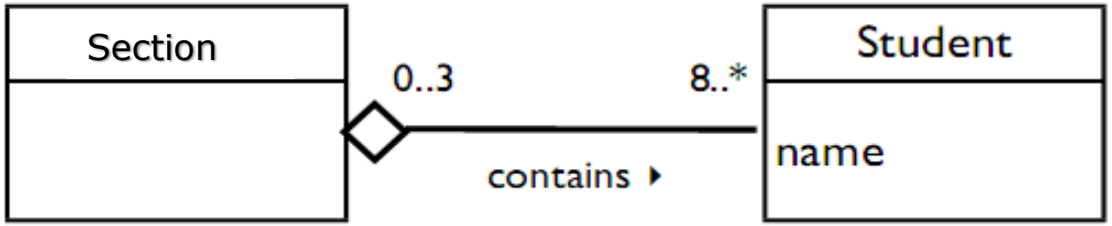
Relationships: Composition

Composition:



without the chess board, the square wouldn't exist...

Aggregation:



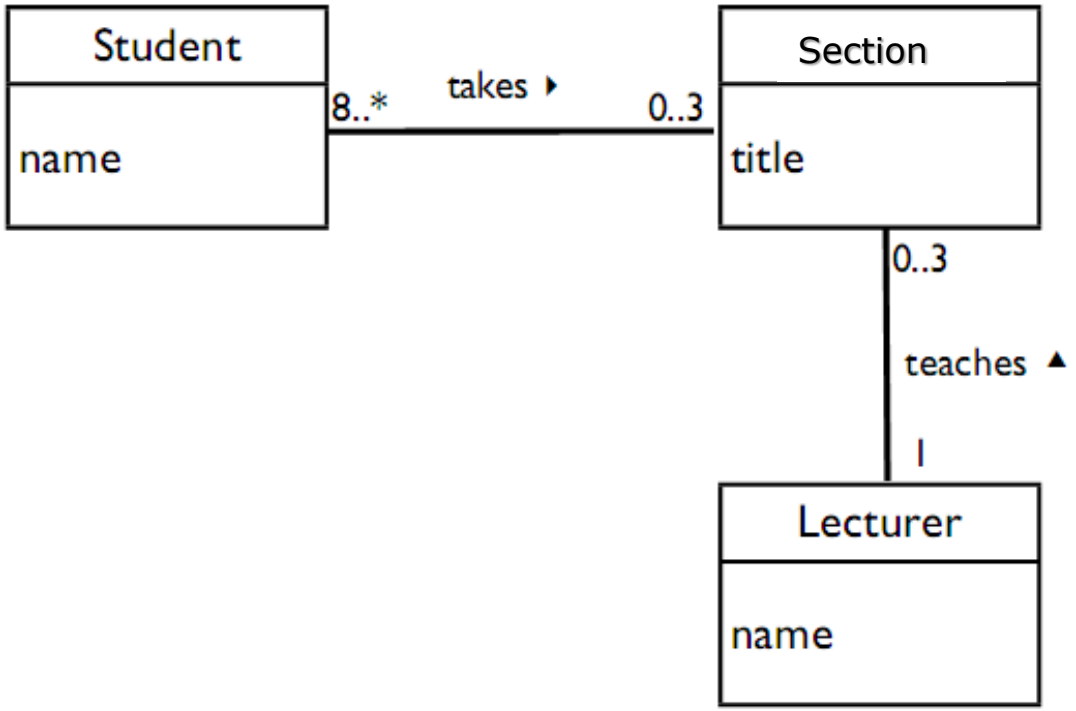
...but without the class list the student would

Association: Multiplicity and Navigation

- Multiplicity defines how many objects participate in a relationships
  - The number of instances of one class related to ONE instance of the other class
  - Specified for each end of the association
- Associations and aggregations are bi-directional by default, but it is often desirable to restrict navigation to one direction
  - If navigation is restricted, an arrowhead is added to indicate the direction of the navigation

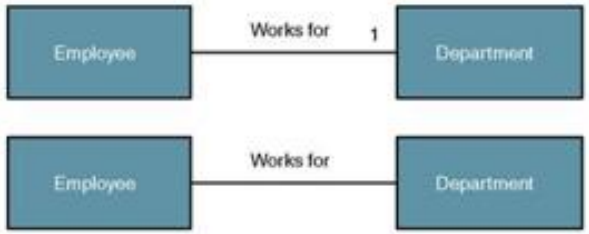
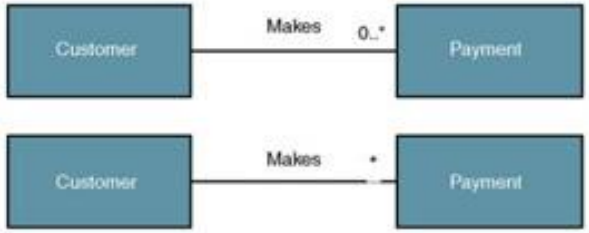
Association: Multiplicity

- how many instances of class A can be associated with a single class B *at a particular point in time*



Type of Multiplicity

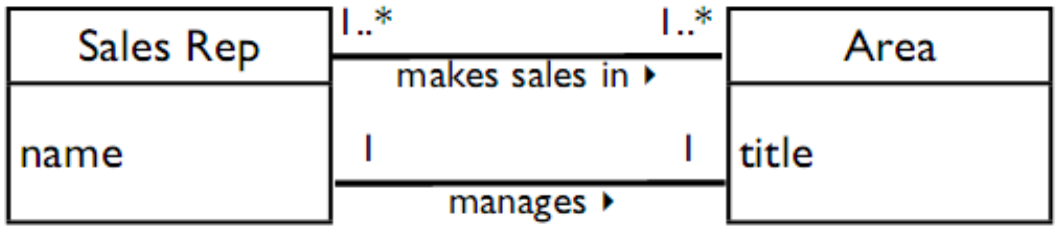
Multiplicity – the minimum and maximum number of occurrences of one object/class for a single occurrence of the related object/class.

Multiplicity	UML Multiplicity Notation	Association with Multiplicity	Association Meaning
Exactly 1	1 or *leave blank*		An employee works for one and only one department.
Zero or 1	0..1		An employee has either one or no spouse.
Zero or more	0..* or *		A customer can make no payment up to many payments.
1 or more	1..*		A university offers at least 1 course up to many courses.
Specific range	7..9		A team has either 7, 8, or 9 games scheduled

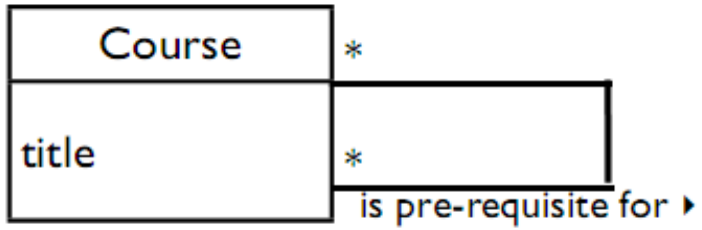
Multiple & Reflexive Associations

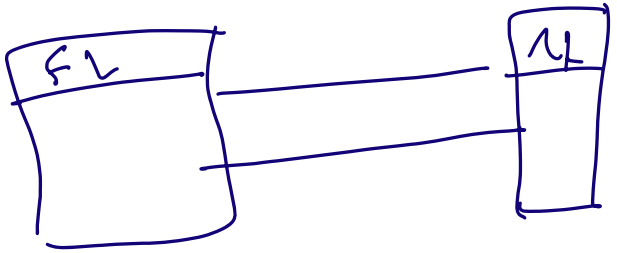
- can two conceptual classes have multiple associations with each other, and can a class associate with itself?

yes,

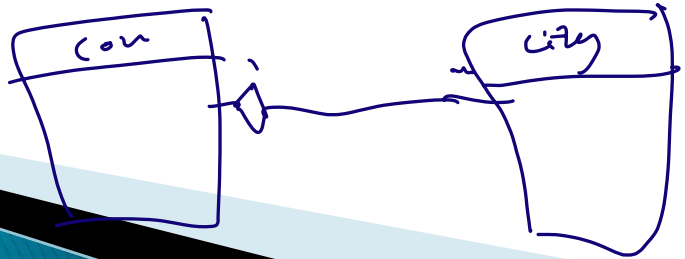


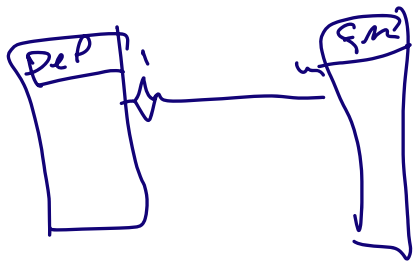
and yes

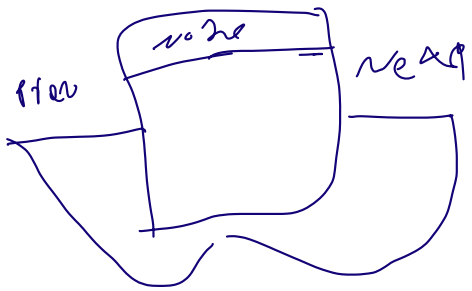


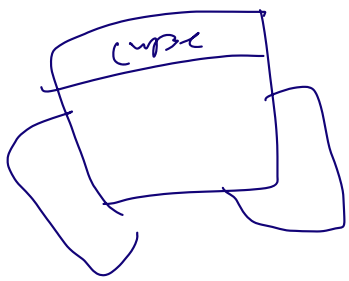


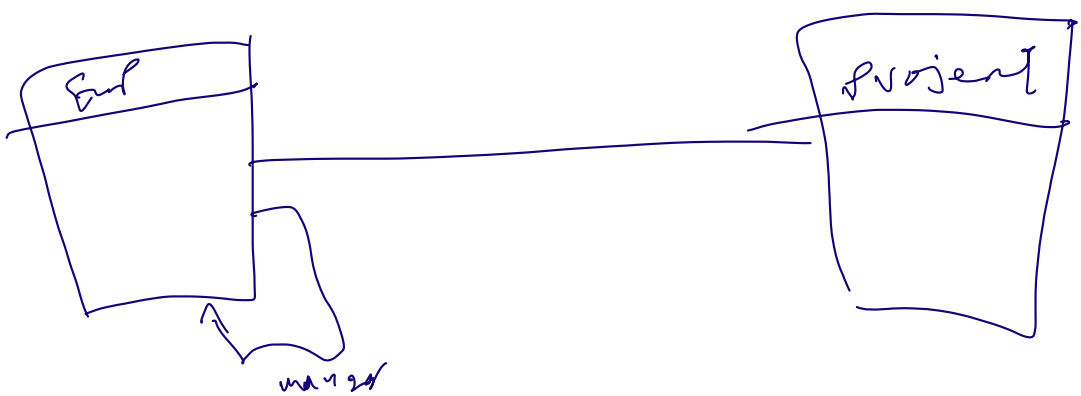
but in each case it might be better to use generalisation and/or to add further conceptual classes to the model






node
prev
next


Employee

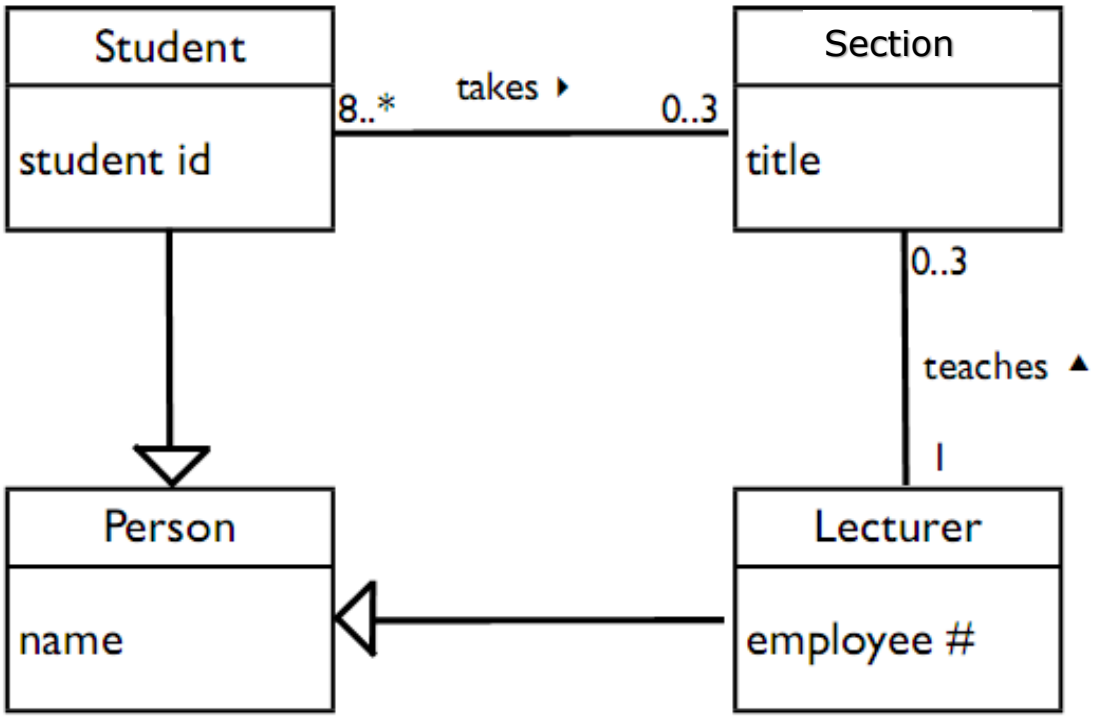

Emp
Project
manages

Relationships: Generalization

- A relationship among classes where one class shares the structure and/or behavior of one or more classes
- Defines a hierarchy of abstractions in which a subclass inherits from one or more superclasses
  - Single inheritance
  - Multiple inheritance
- Generalization is an “is-a-kind of” relationship

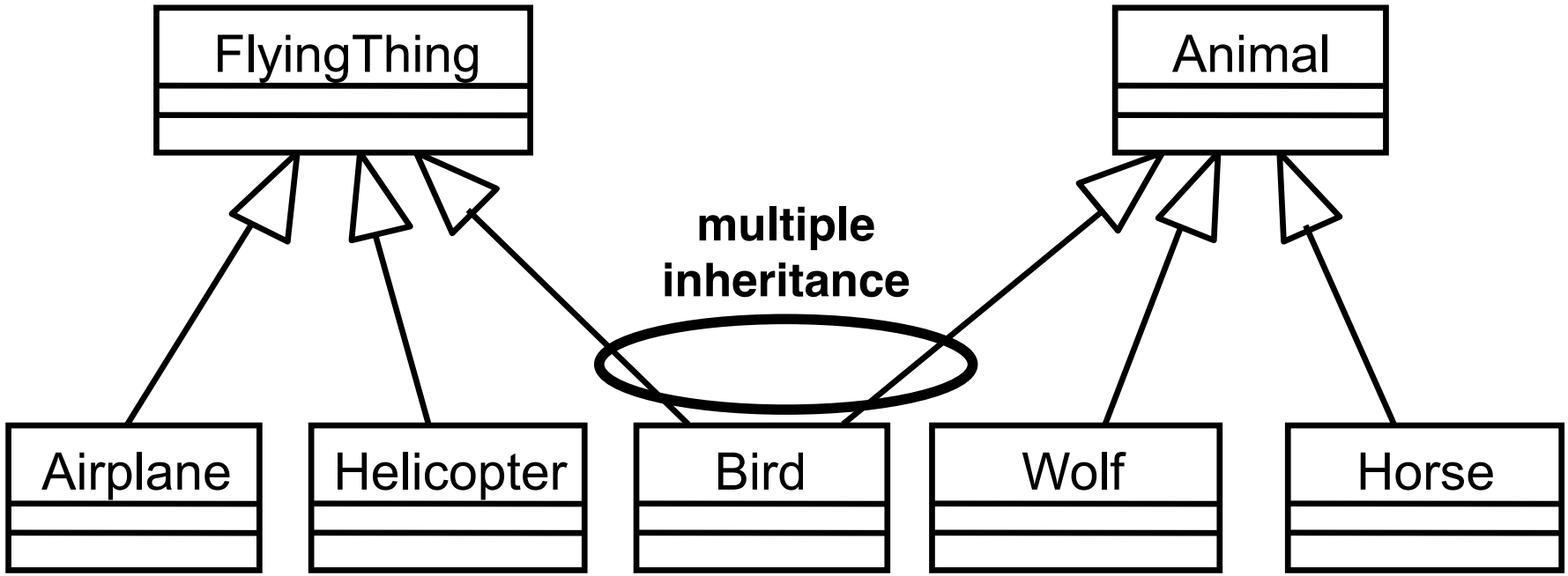
Example: Single Inheritance

- sometimes conceptual classes are (sub) types of another class:



Example: Multiple Inheritance

A class can inherit from several other classes



Use multiple inheritance only when needed, and always with caution !

Associations

- Shows relationship between classes
- A class diagram may show:

Relationship	
Generalization (inheritance)	 "is a" "is a kind of"
Association (dependency)	does ▶ "Who does What" "uses"
Aggregation	 "has"
Composition: Strong aggregation	 "composed of"

Example: Library System

- Consider the world of libraries. A library has books, videos, and CDs that it loans to its users. All library material has a id# and a title. In addition, books have one or more authors, videos have one producer and one or more actors, while CDs have one or more entertainers. The library maintains one or more copies of each library item (book, video or CD). Copies of all library material can be loaned to users. Reference-only material is loaned for 2hrs and can't be removed from the library. Other material can be loaned for 2 weeks. For every loan, the library records the user, the loan date and time, and the return date and time. For users, the library maintains their name, address and phone number.

- Define the two main actors.
- Identify use cases by providing the actors, use case names. Draw the us e case diagram.
- Create the conceptual class diagram.

Example: Digital Music players

Draw a UML Class Diagram representing the following elements from the problem domain for digital music players: An artist is either a band or a musician, where a band consists of two or more musicians. Each song has an artist who wrote it, and an artist who performed it, and a title. Therefore, each song is performed by exactly one artist, and written by exactly one artist. An album is composed of a number of tracks, each of which contains exactly one song. A song can be used in any number of tracks, because it could appear on more than one album (or even more than once on the same album!). A track has bitrate and duration. Because the order of the tracks on an album is important, the system will need to know, for any given track, what the next track is, and what the previous track is.

Draw a class diagram for this information, and be sure to label all the associations (relationships) with appropriate multiplicities.

References & Further Reading

- *Applying UML & Patterns (Larman 2007), Chapters 6, 9.*
- *Object-Oriented Systems Analysis and Design (Bennett et* al, Third Edition, 2006), Chapter 6, 7.