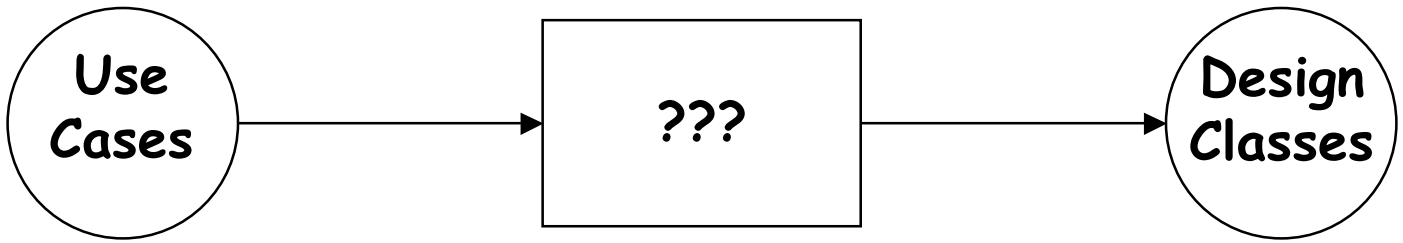
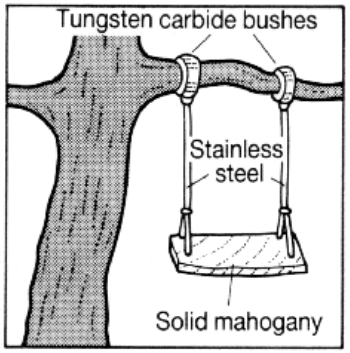
From Requirements Toward Design

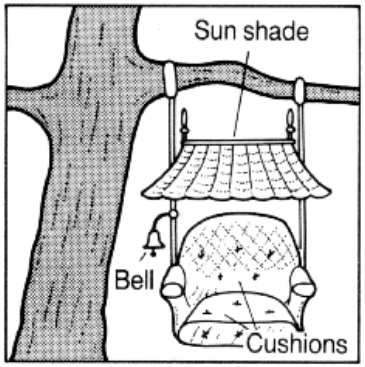
Building an Object-Oriented Model for the Software Product



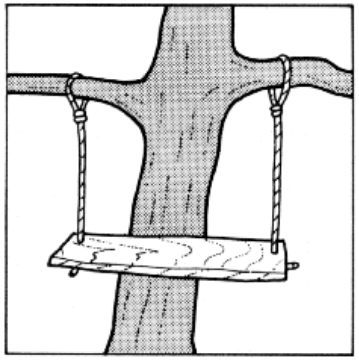
Avoid the possibility of a requirements/design disconnect

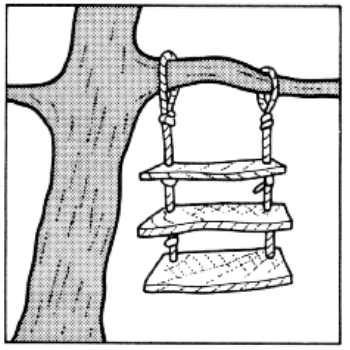


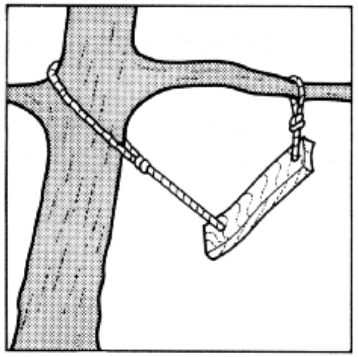
What Product Marketing specified

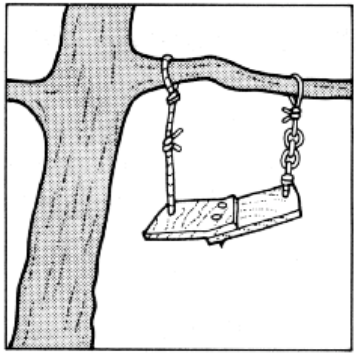


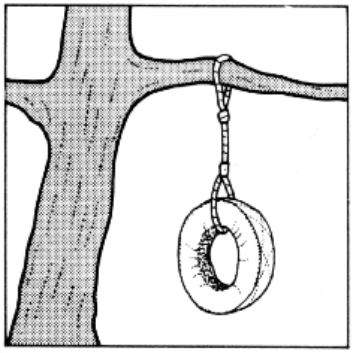
What the salesman promised


Design group's initial design


Corp. Product Architecture's modified design


Pre-release version


General release version


What the customer actually wanted

Domain Modeling

What is Domain Modeling?

- Modeling objects and concepts that actually exist or make sense in the environment or enterprise of the application.

What is the Purpose of Domain Modeling?

- A software product usually relates to some real-world domain.
- The domain model expresses **concepts** and their **interconnection**.
- By modeling the domain, we have a base from which to
  - relate to the user, and
  - conduct further design.

Exhibiting a Domain Model

- Use UML class diagrams
  - Provide a "visual glossary"
  - Related to "data dictionary"
- Don't infer that these classes necessarily refer to ultimate implementation classes :
  - Some might, others might not.
  - Therefore these are called Domain Classes, or Conceptual classes.

UML Modeling

- UML is a graphical language for modeling
- UML is not a programming language
- UML is standardized by the OMG

OMG =
"Object Management Group",
a consortium, not a company

http://www.omg.org/

UML

- Unified Modeling Language
- unifies the approaches of the "three amigos":
  - Grady Booch
  - Ivar Jacobson
  - James Rumbaugh
- Includes a version of E-R (Entity-Relationship) diagrams from the database world.

Earlier Work on Modeling

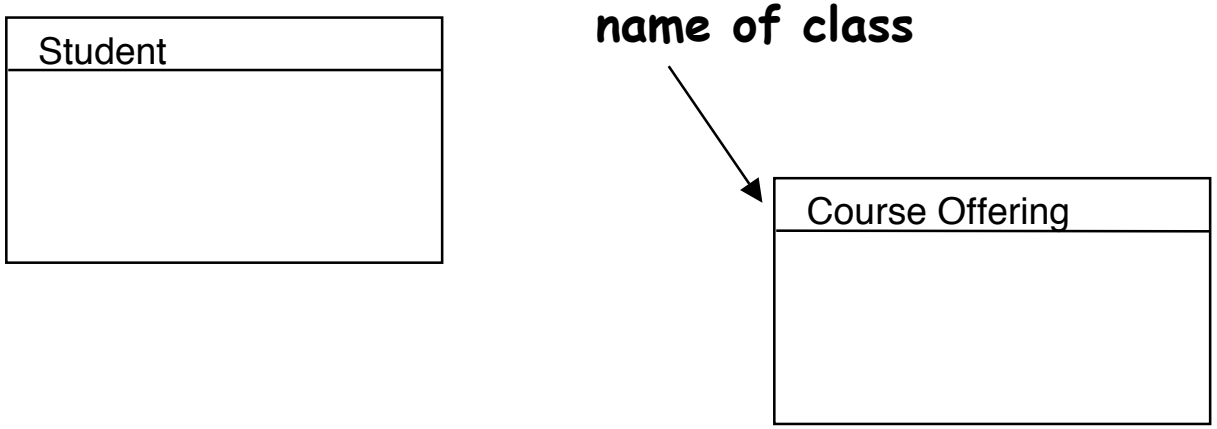
- A lot of these ideas are traceable to celebrated work on **database modeling**, particularly the E-R (Entity-Relationship) Model originally proposed by Peter Chen in 1976.

Prof. Peter Chen,
Louisiana State Univ.

- Entity = UML Class,
  Relationship = UML Association

- UML provides a significant set of extensions to those early ideas.

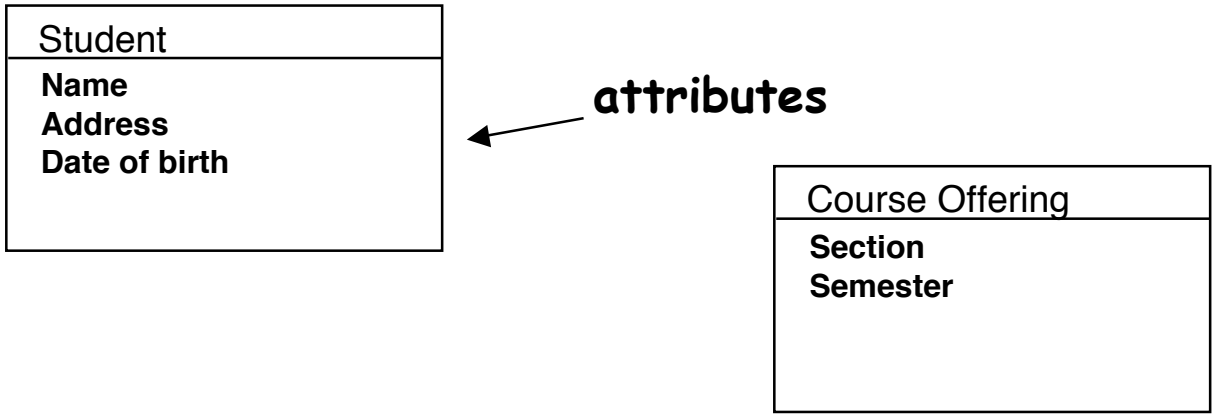
UML: Classes are shown by boxes



Classes, not actual objects

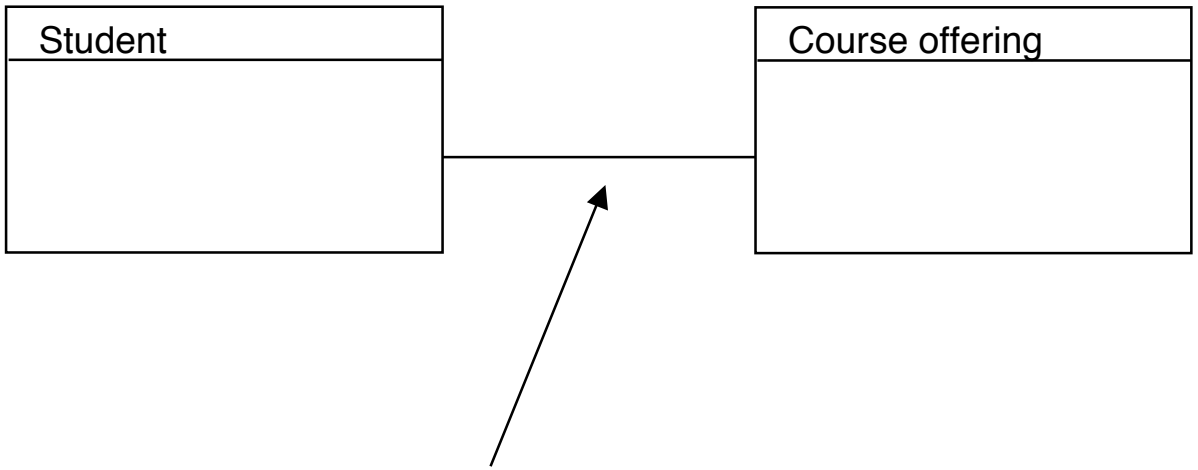
(Objects can also be shown by boxes;
For objects, names are always *underlined*.)

Attributes may be listed



- In *domain* modeling:
  - Methods of classes are not listed.
  - These are not software classes, although some of them may become so.

Associations are shown by lines connecting boxes



association:

perhaps meaning that a student is enrolled in a course offering

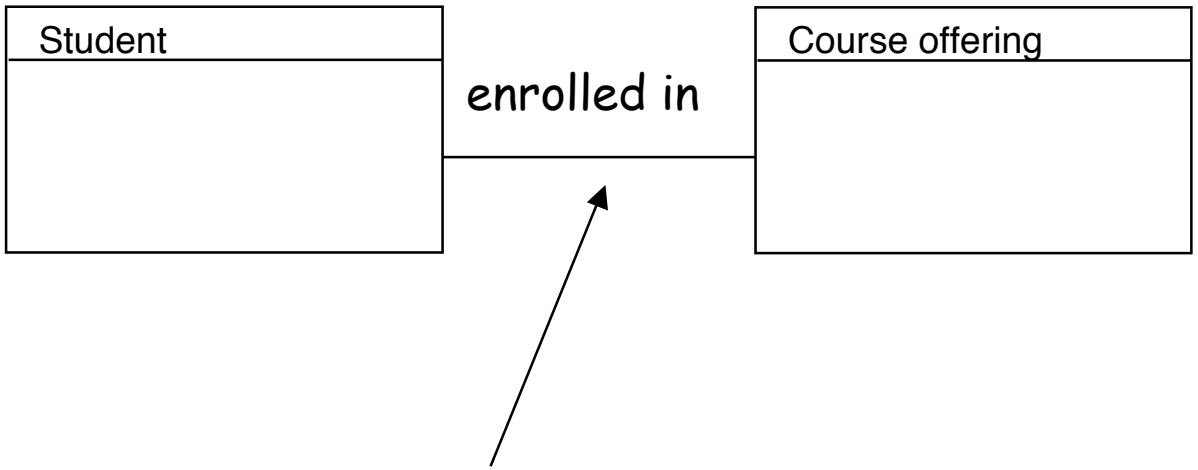
Possible Associations

- Member-of
- Part-of
- Contained-in
- Owned-by
- Director-of
- Manages
- Adjacent-to
- Linked-to
- Controls
- Follows
- Describes
- Communicates-with

Associations Show *Static* Structure

- Unlike Use Cases, which describe flow of events.
- A class diagram, with associations, only shows static relationships.
- Dynamic relationships, such as message-passing, may be *suggested* by associations.

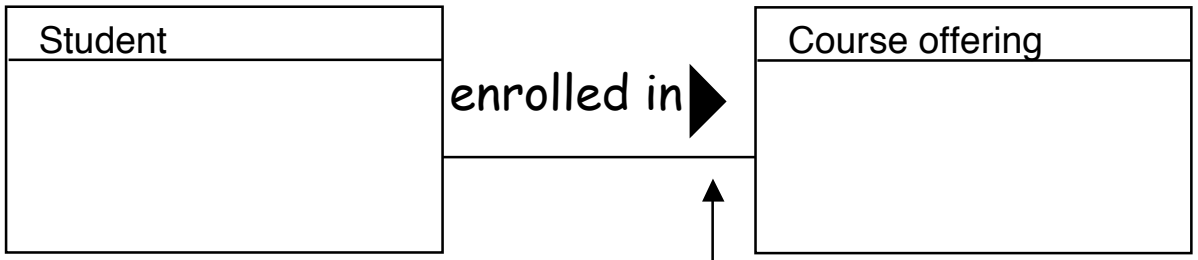
Naming Associations



association:

perhaps meaning that a student is enrolled in a course offering

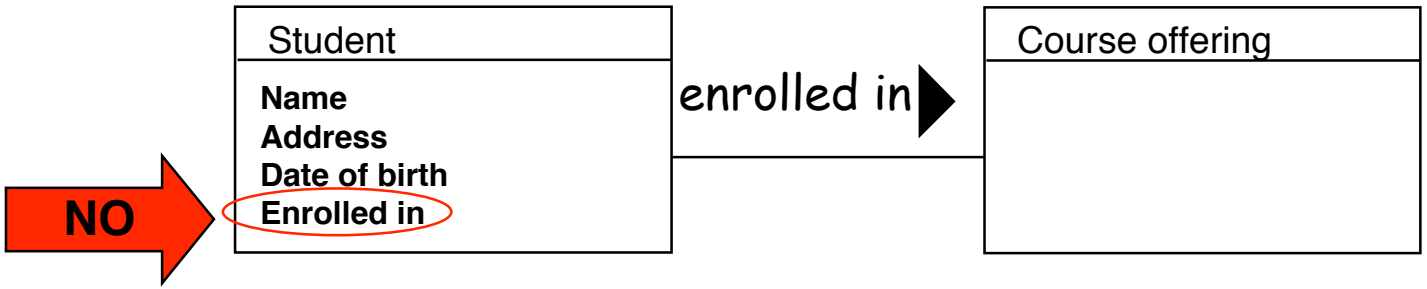
Directionality of Association Names



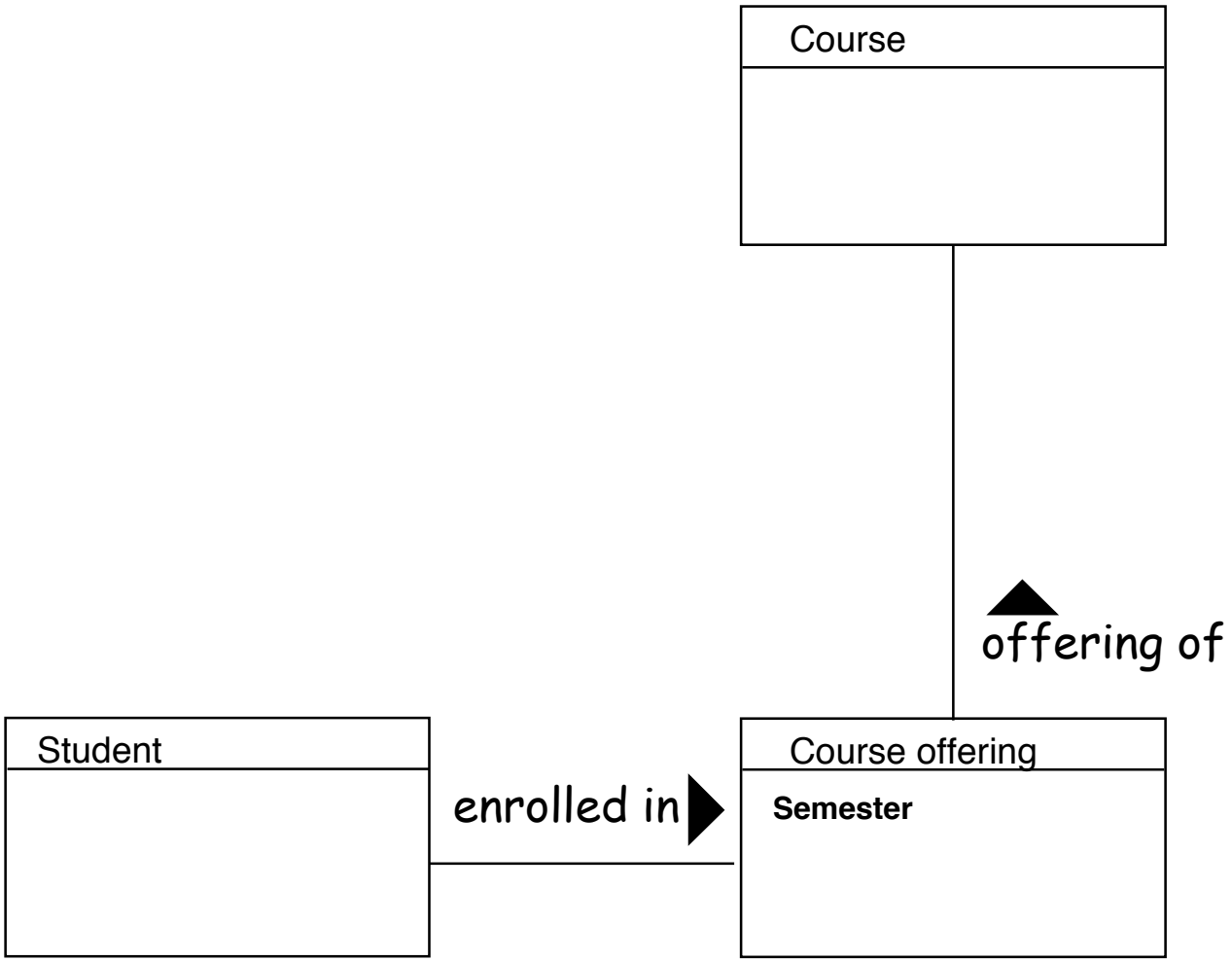
Arrowhead shows direction of ***reading*** the name of the association,

e.g. "student **enrolled in** course offering".

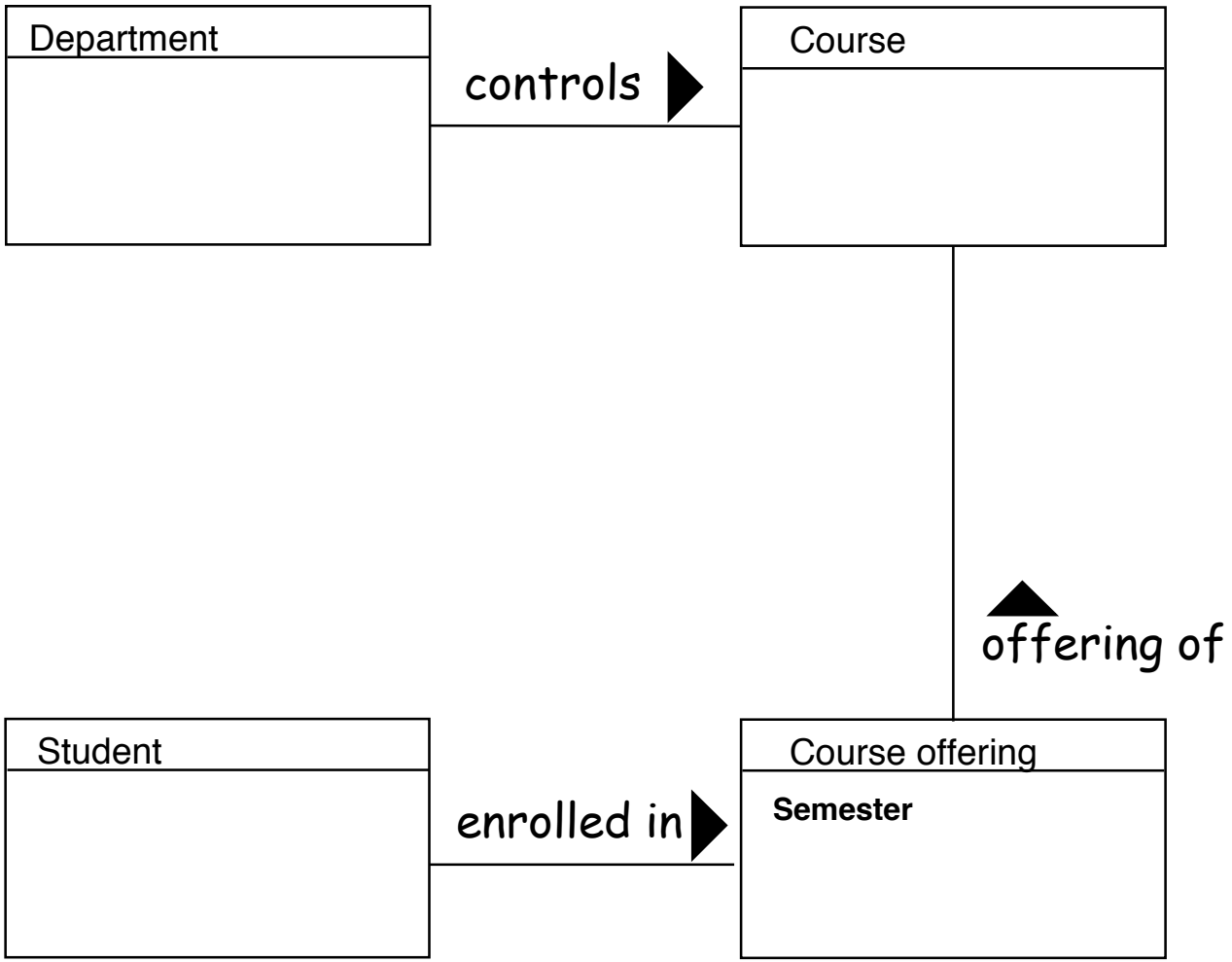
Don't count Associations as Attributes



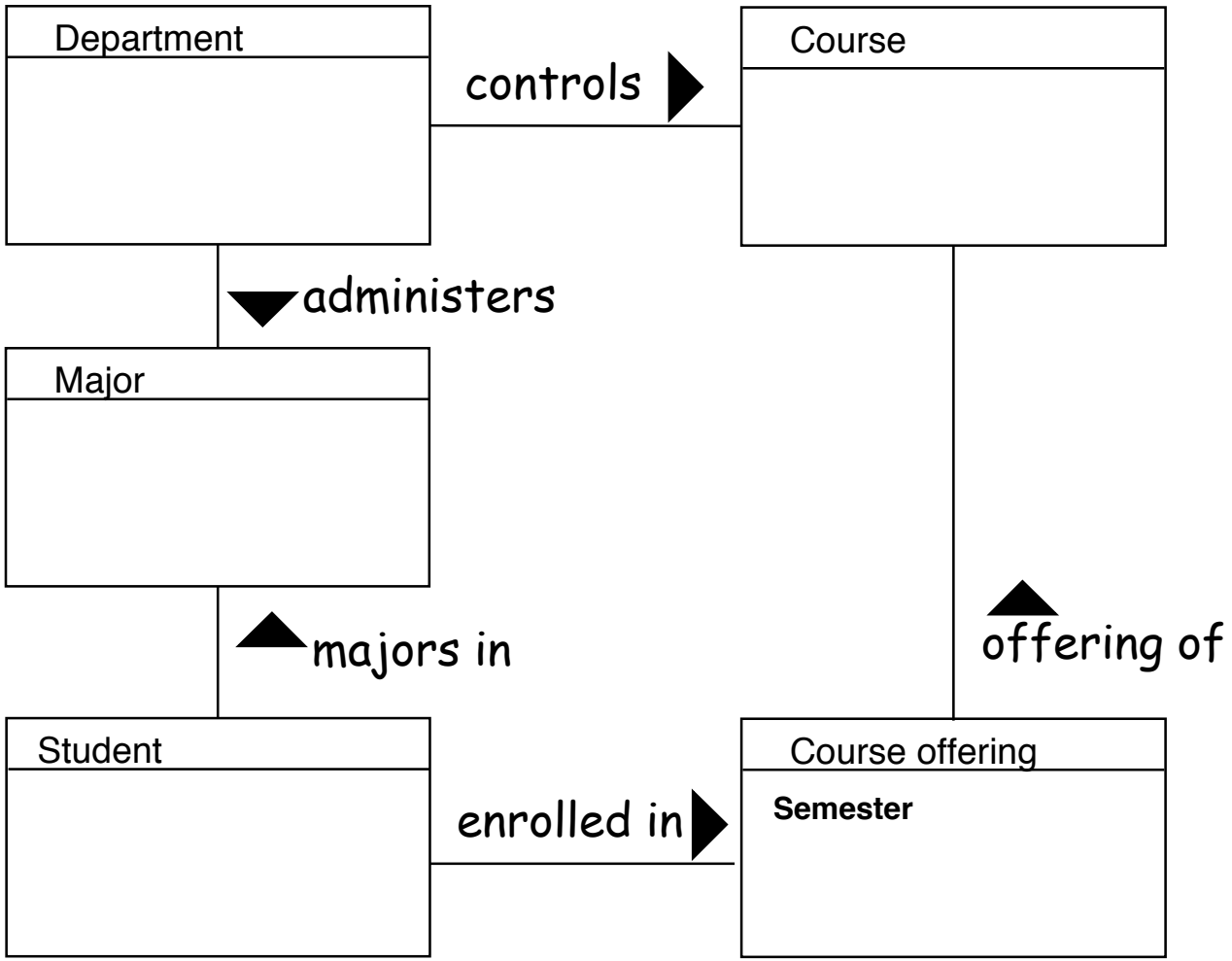
Extending the Domain Model



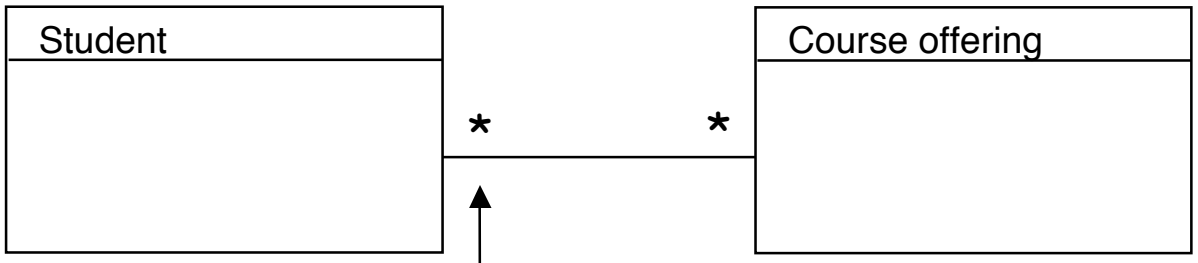
Extending the Domain Model



Extending the Domain Model



Multiplicities of Associations

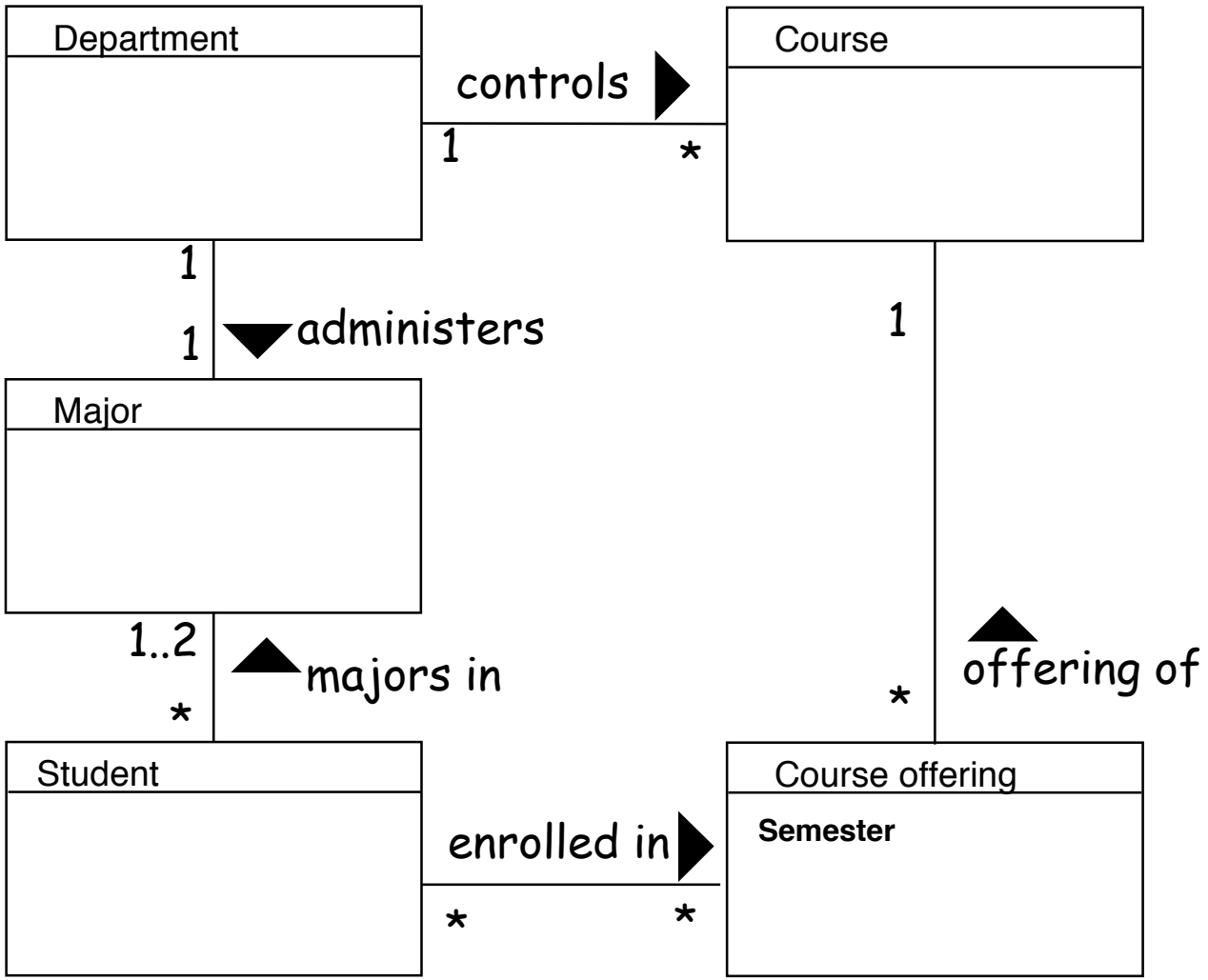


Multiplicity Indicator:

* means that a course offering has **0 or more** students.
Other options:

1	exactly 1 default
n	exactly n
m..n	between m and n inclusive
m..	m or more

Possible Multiplicities

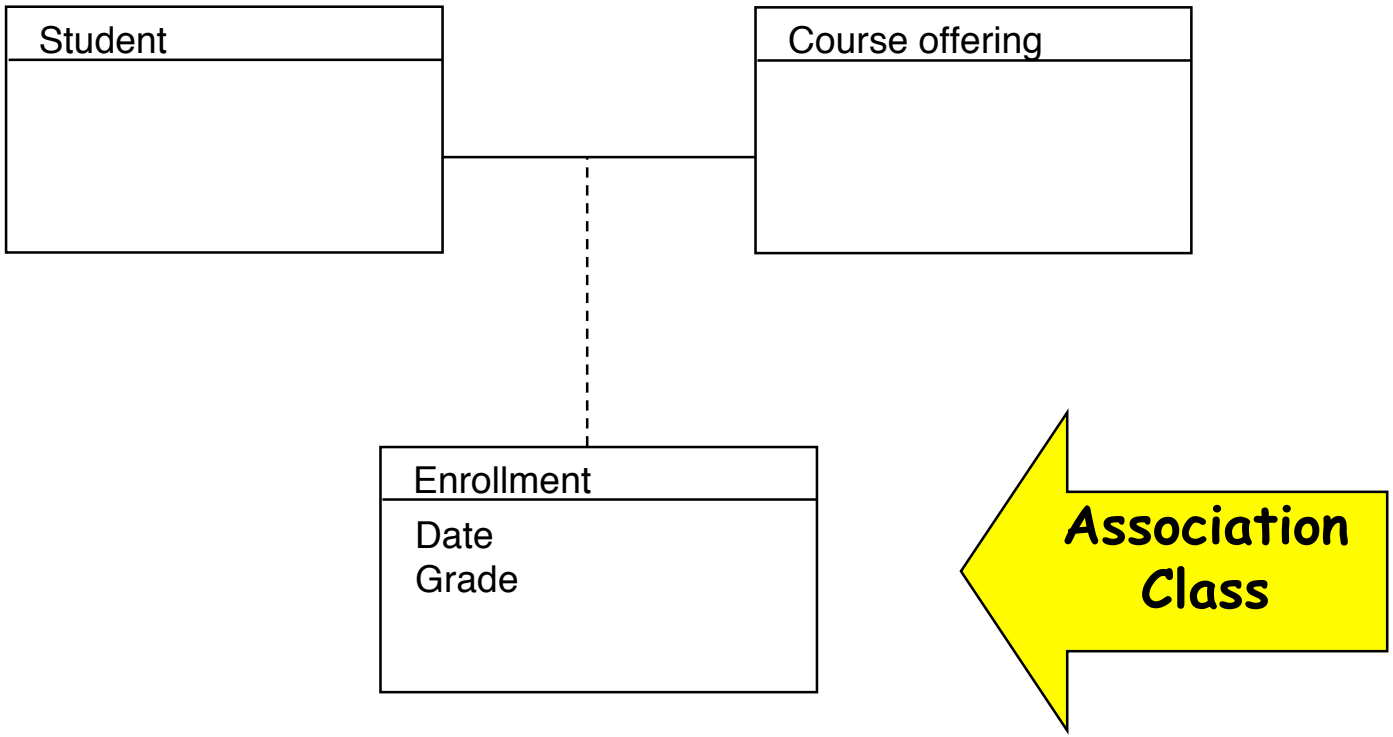


Notes on Multiplicity

- Consideration of multiplicity can be deferred; it should not dominate early discovery processes.

Association Classes

An association may have properties of its own.



Exercise

- Form a Domain Model based on the following suggestive names:
  - Airline
  - Airport
  - Crew Member
  - Flight-Segment Instance
  - Flight-Segment (i.e. leg of a flight)
  - Flight
  - Passenger
  - Scheduled Arrival
  - Scheduled Departure

Extracting Domain Model from A Verbal Description

- Nouns and noun-phrases often suggest classes *or* attributes.
- Attributes are distinguished from classes:
  - Attributes have an ephemeral **value**.
  - The exact **identity** of the value is not important in the overall design.
  - Examples: date, time, price, name, boolean attributes, enumerated types, ...

Example: Identify Classes and Attributes

Door Security Card System

The administrator can register a new card at the central unit or at a door terminal. After identifying and authenticating herself, the administrator may register a new card to the system. The new card is classified and information about the user is added. The information added is the social security number or another identification number.

Once a card is registered to the system, access control can be added. The administrator can register which doors the card should have access to. This can be done both from a door terminal and from the central unit.

The administrator is also allowed to unregister a card and withdraw access privileges. This is done in the same manner as the registering and granting of access, i.e. both from a door terminal and from the central unit.

When the user has been granted access to a room, she can open the door by inserting her card in the card reader. The door terminal asks her for her Personal Identification Number (PIN). The PIN is validated with the card, and if it is correct, the door is opened. If the user enters an erroneous code three times, the access privileges are withdrawn.

Factoring

- When classes appear to be getting too large (too many attributes or associations), consider the possibility of factoring the class into smaller classes.

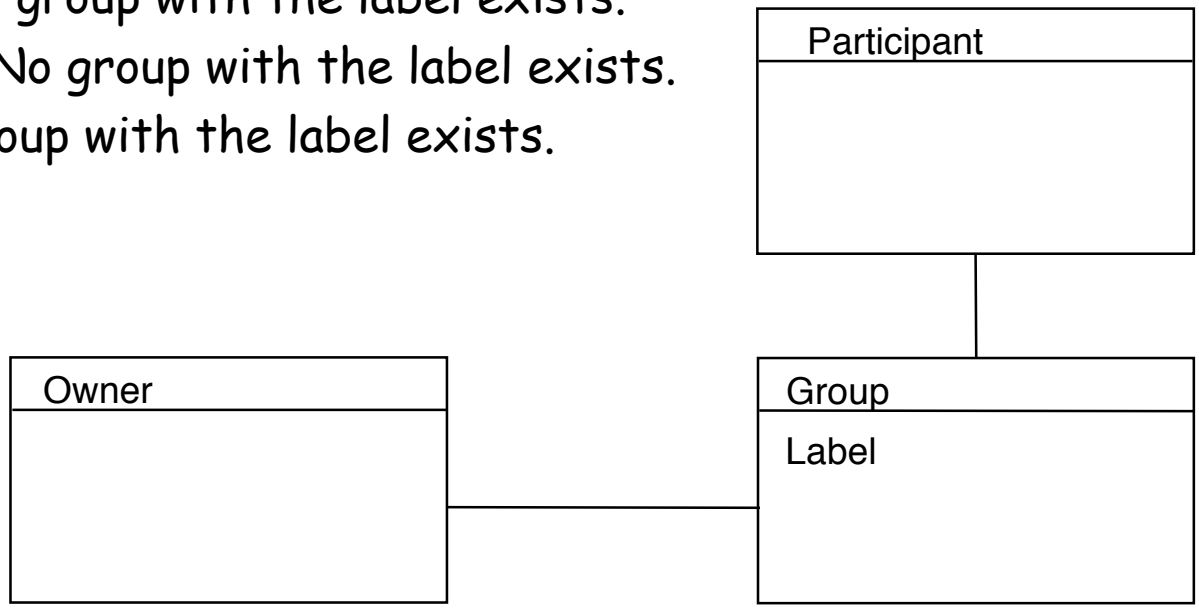
Combining with Use Cases

- Use-Cases: Dynamic
- Domain Classes: Static
- Eventually these need to be related to each other:
  - Names of objects mentioned in use cases should be identified with classes in the domain.

Example Use Case from a Meeting Scheduler:

DisbandGroup: Disband a group

- Goal: To enable a group owner to disband a group.
- Actors: Group owner, group memebers
- Initiator: Group owner
- Description: The group owner dissolves an existing group. Members of the group are notified.
- Pre-conditions: A group with the label exists.
- Post-conditions: No group with the label exists.
- Exception: No group with the label exists.
- Options
- Scenario



Other Possible Domains for Consideration

- Library
- Transit system
- Asset management system
- Supermarket check-out system
- Consortium of colleges/universities
- Athletic events management system
- Automobile service station pump system
- Subversive organization