3. The Unified Modeling Language(UML)

The Unified Modeling Language

- A graphical language for communicating design specifications
- To meet the special needs of describing object-oriented software design
- The *class* and *activity* diagram types are particularly useful for discussing database design issues
  - UML class diagrams
    - The original creators of UML point out the influence of ER models on the origins of class diagrams
    - Capture the structural aspects found in database schemas
    - UML class diagrams and entity-relationship models are similar in both form and semantics
  - UML activity diagrams
    - Similar in purpose to flowcharts
    - Facilitate discussion on the dynamic processes involved in database design
    - Processes are partitioned into constituent activities along with control flow specifications
- UML 2.0

Class Diagrams

- Class
  - A descriptor for a set of *objects* that share some attributes and/or operations
  - Ex: A car
    - Has attributes, such as vehicle identification number (VIN) and mileage
    - Also has operations, such as accelerate and brake
    - All cars have these attributes and operations
    - Individual cars differ in the details
      - Individual cars are objects that are instances of the class "Car."
- Classes and objects
  - A natural way of conceptualizing the world around us
  - The paradigms that form the foundation of object-oriented programming
- The development of object-oriented programming led to the need for a language to describe object-oriented design, giving rise to UML

Class Diagrams vs ER Diagrams

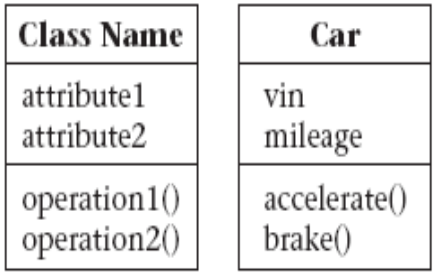
- Classes are analogous to entities
- Database schemas can be diagrammed using UML
  - It is possible to conceptualize a database table as a class
    - The columns in the table are the attributes
    - The rows are objects of that class
- The major difference between Classes and Entities
  - The lack of operations in Entities
    - Stored procedures, functions, triggers, and constraints: Are forms of named behavior that can be defined in databases
    - These are not associated with the behavior of individual rows
      - These behaviors are not stored in the definition of rows within the database
      - There are no operations named "accelerate" or "brake" associated with rows in our "Car" table
  - Operation in UML
    - Refers to the methods inherent in classes

Basic Class Diagram Notation

- Showing both *attributes* and *operations*

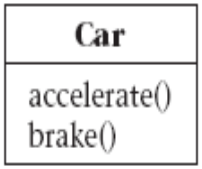
Classes

Notation and Example

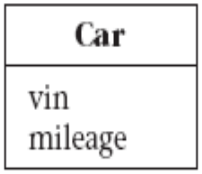


Notational Variations

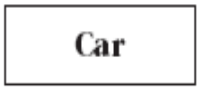
Emphasizing Operations



Emphasizing Attributes



Emphasizing Class



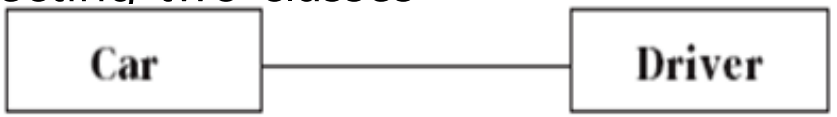
Basic Class Diagram Notation - Classes

- Showing operations and hiding attributes is a very common syntax used by software designers
- Database designers, on the other hand, do not generally deal with class operations
  - The attributes are of paramount importance
  - The needs of the database designer can be met by writing the class with only the class name and attribute compartments showing

Basic Class Diagram Notation - Relationships

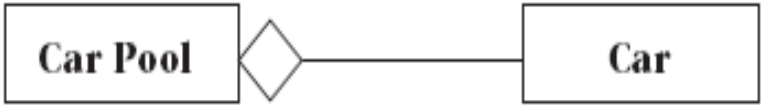
- Various types of relationships may exist between classes
- Association
  - Drawn with a line connecting two classes

Association



- Aggregation
  - Indicates "part-of" association
    - Ex: A Car may be part of a Car Pool. The Car also exists on its own, independent of any Car Pool

Aggregation

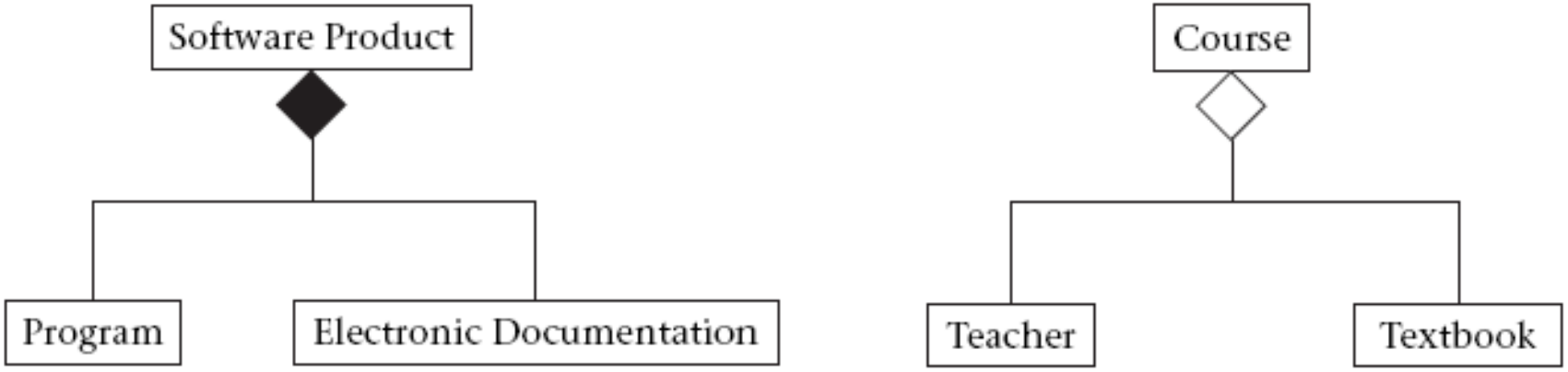


  - The part may be shared among multiple objects
    - Ex: a Car may belong to more than one Car Pool
- Composition
  - Another "part of" association
  - The parts are strictly owned, not shared

Composition

The distinction between Composition and Aggregation

- Sometimes elusive

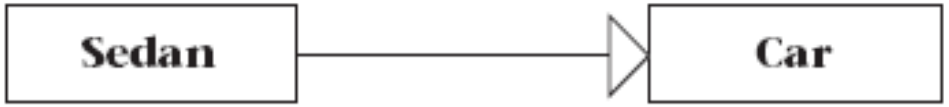


- The composition signifies that the parts do not exist without the Software Product
- A Teacher and a Textbook are aggregated by a course
  - The aggregation signifies that the Teacher and the Textbook are part of the Course, but they also exist separately.
  - If a Course is canceled, the Teacher and the Textbook continue to exist.

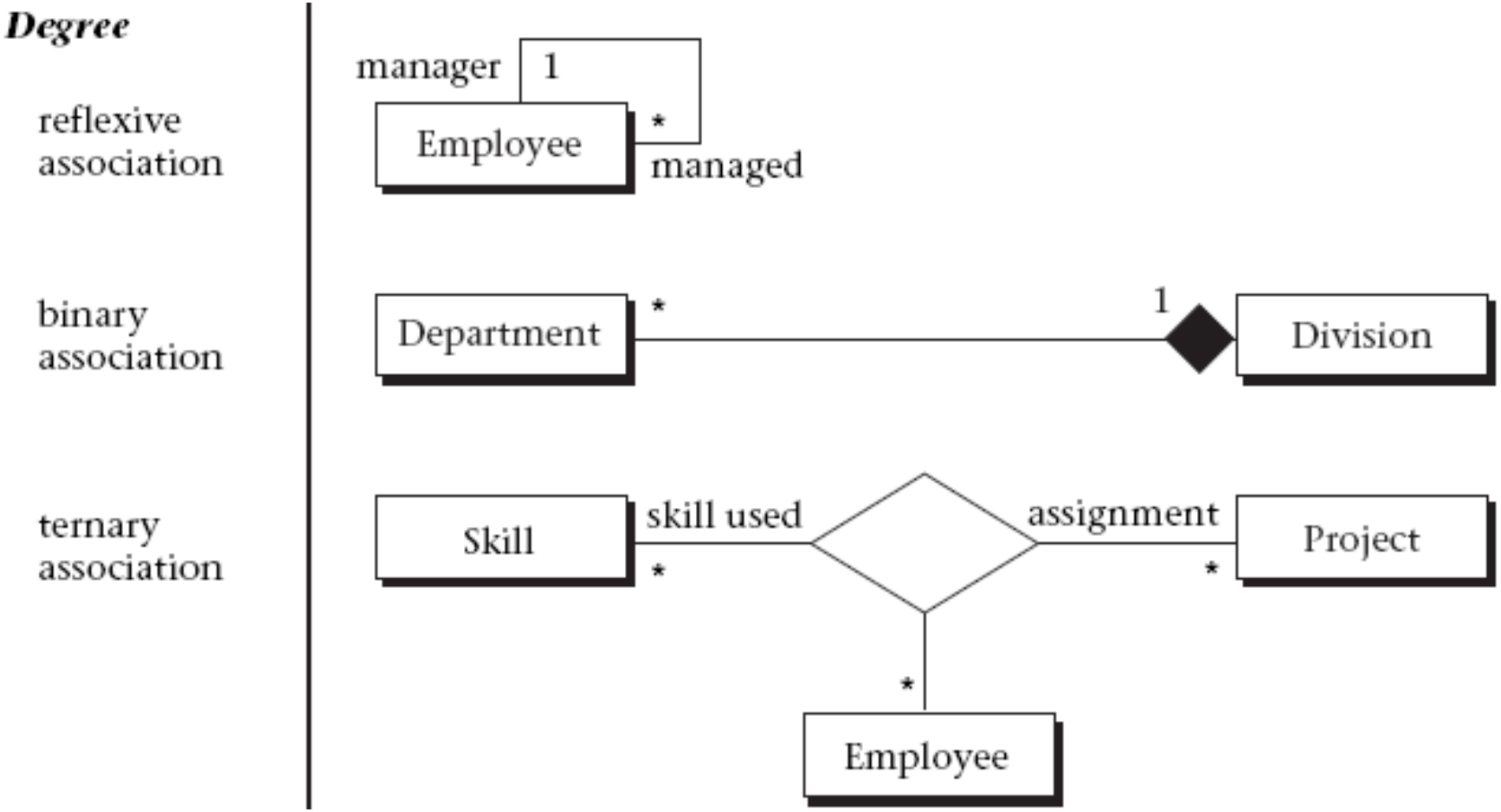
Basic Class Diagram Notation - Relationships

- Generalization
  - Ex: Sedan is a type of car
    - The "Car" class is more general than the "Sedan" class

Generalization



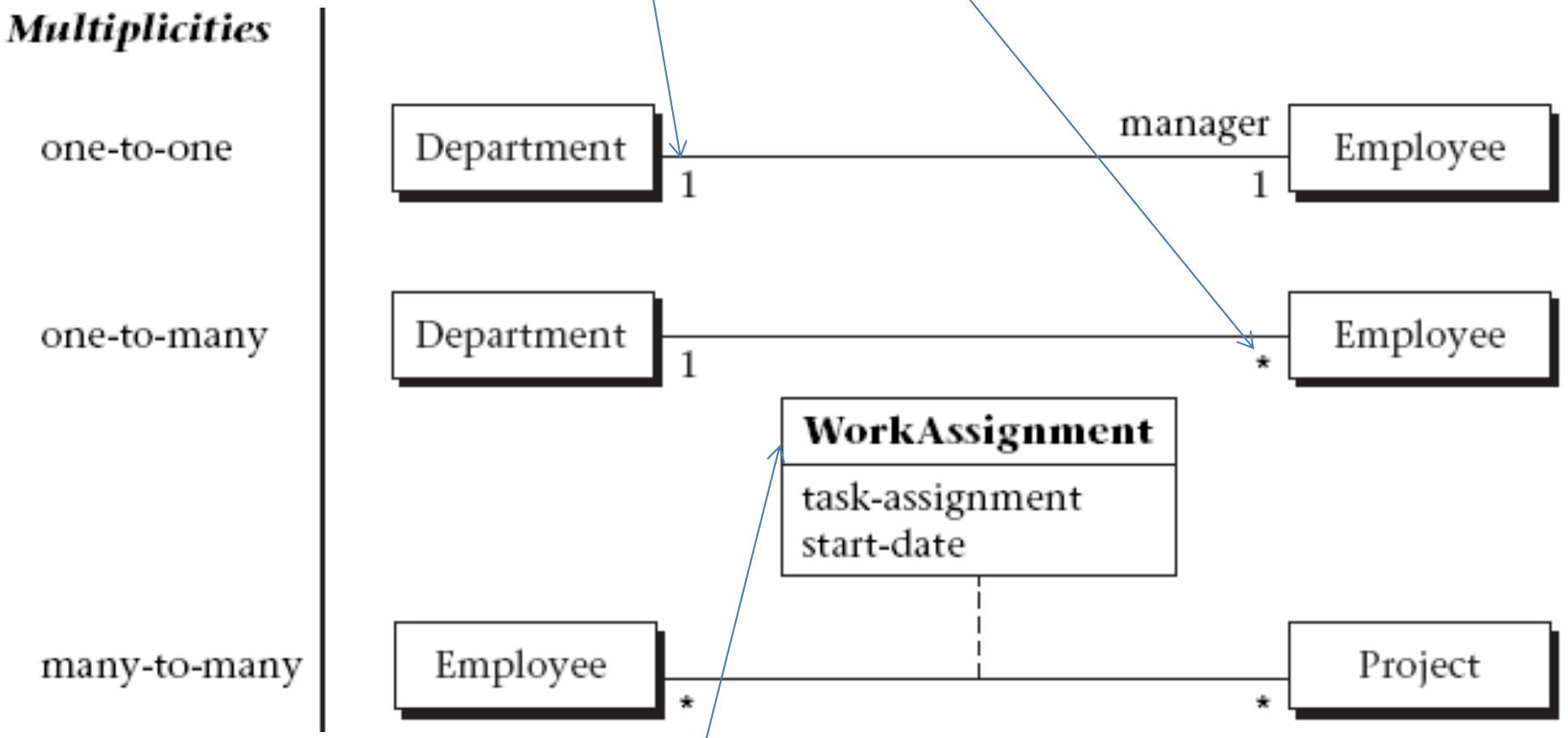
Degrees of Relationships



Multiplicity of Relationships

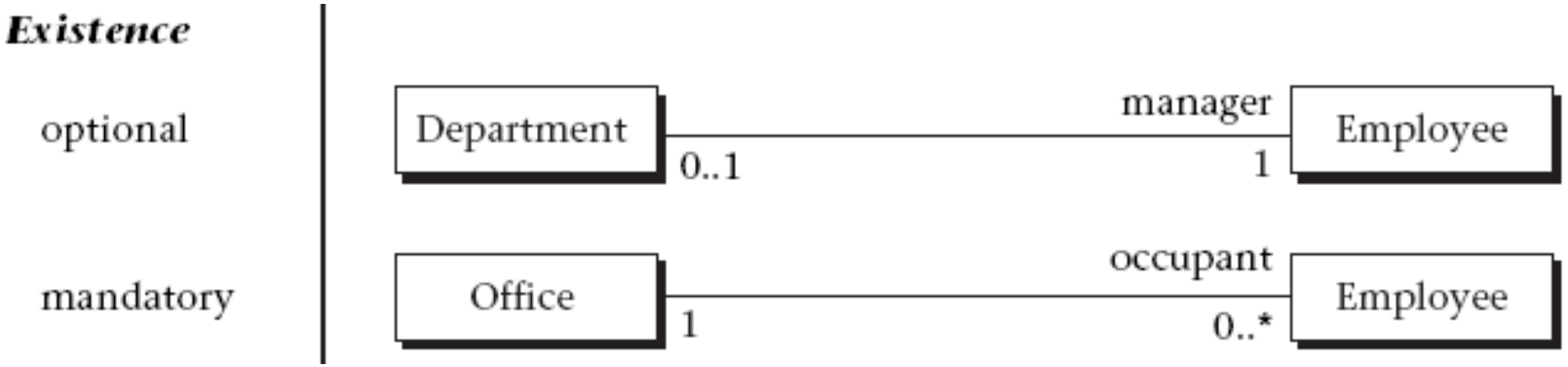
[1 .. 1] means the same as [1]

The asterisk by itself carries the same meaning as the range [0 .. *]



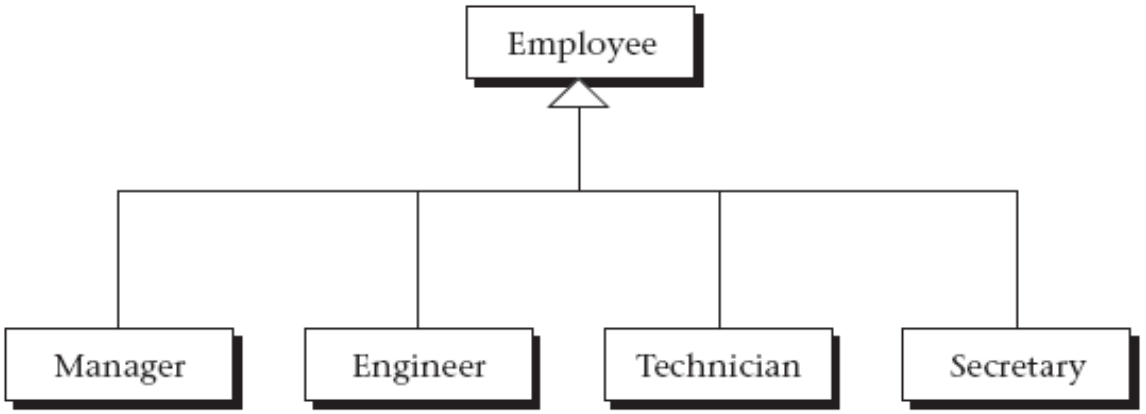
If an association has attributes, these are written in a class that is attached to the association with a dashed line

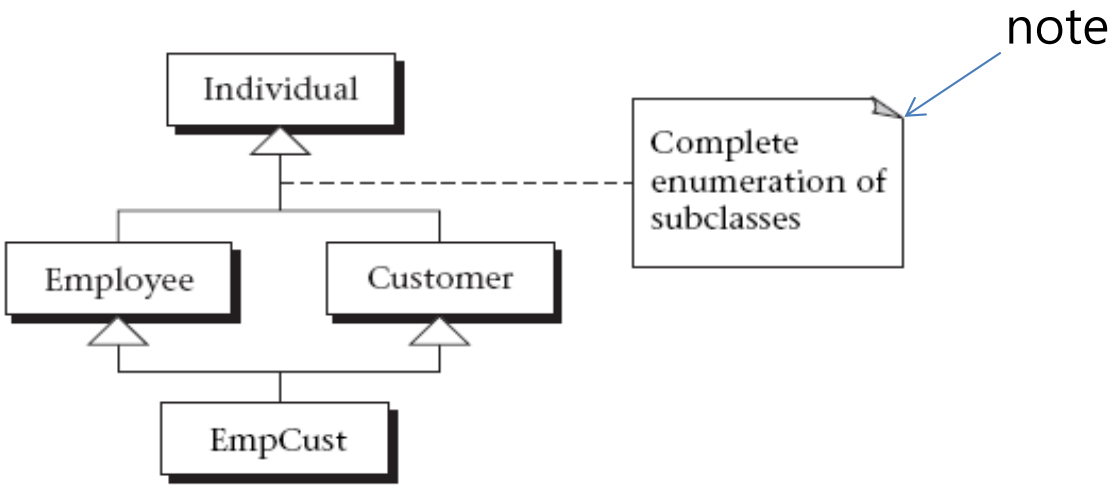
Existence of Relationships



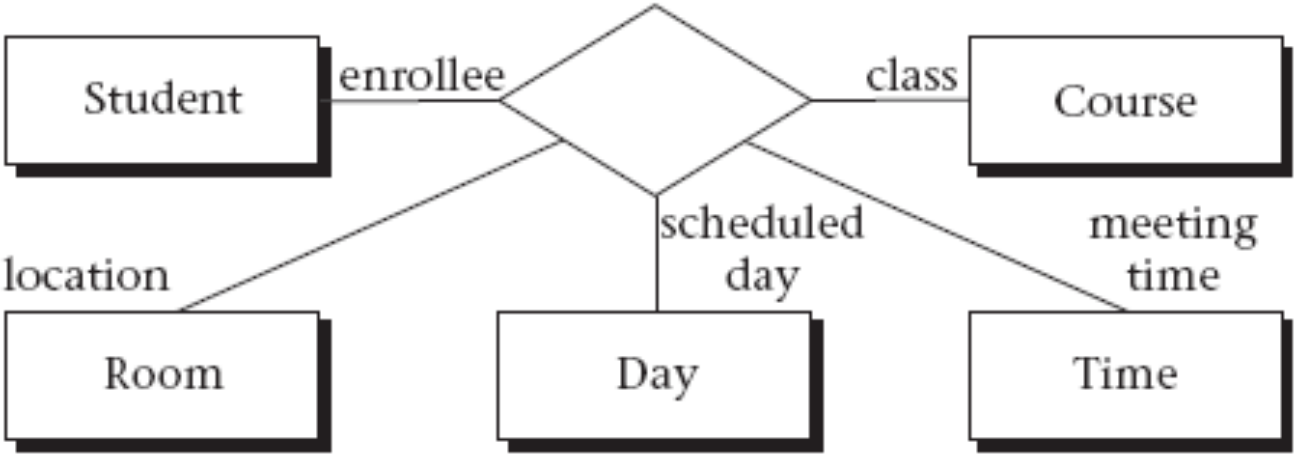
Generalization

- Generalization is another type of relationship; a superclass is a generalization of a subclass
- Specialization is the opposite of generalization; a subclass is a specialization of the superclass





UML *n*-ary relationship

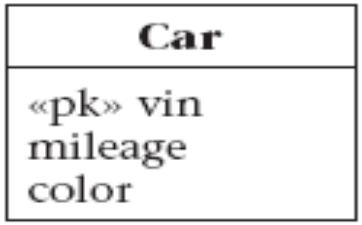


- The *n*ary relationship may be clarified by specifying roles next to the participating classes
  - A Student is an enrollee in a class, associated with a given Room location, scheduled Day, and meeting Time

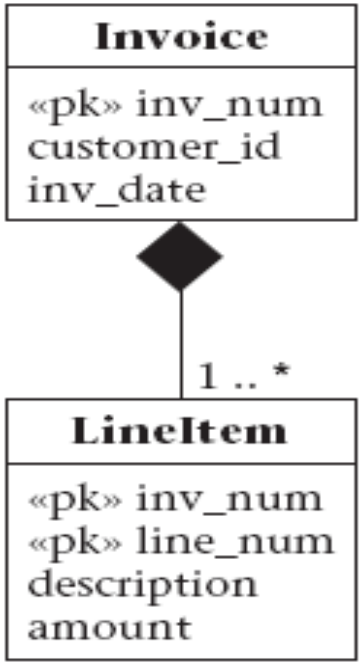
Primary Key

- UML does not have an icon representing a primary key
  - However, UML is extensible
- Stereotype
  - Depicted with a short natural language word or phrase
  - Enclosed in guillemets: « and »
- «pk»

Primary key as a stereotype



Composition example with primary keys



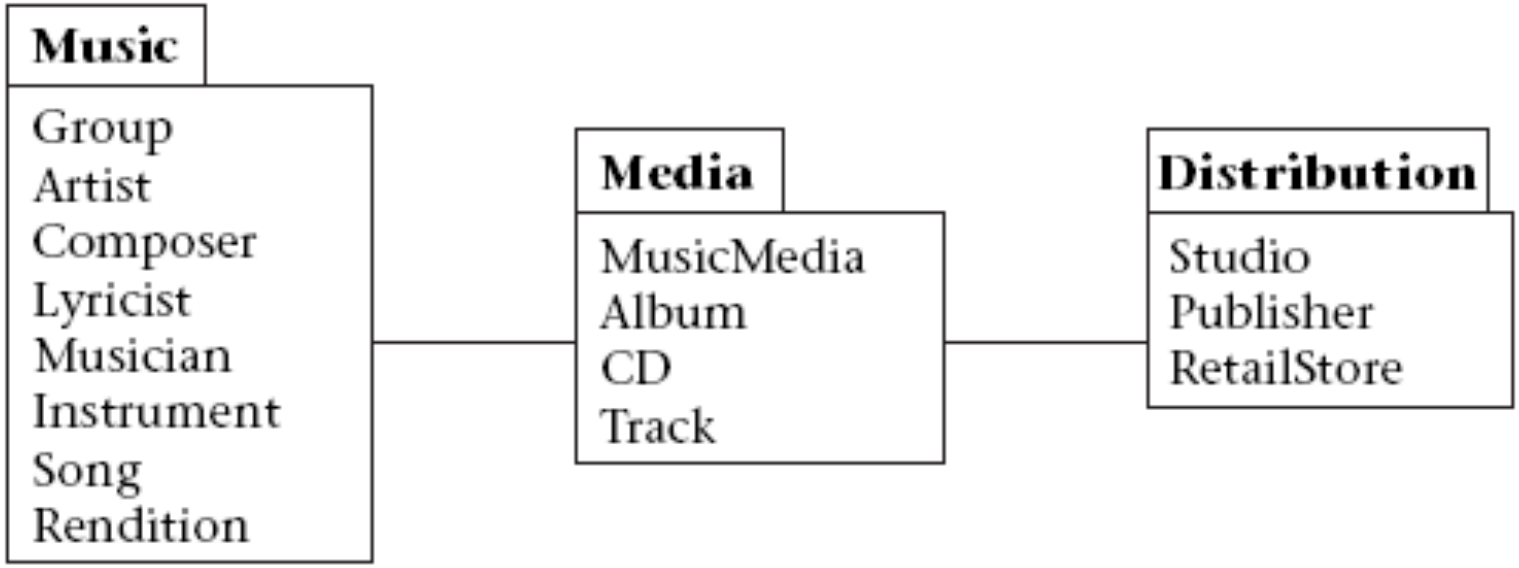
Packages

- Used to organize classes into groups
- Packages may themselves also be grouped into packages
- The goal of using packages
  - To make the overall design of a system more comprehensible
  - To group related classes together within a schema, and present the schema clearly
- Given a set of classes, different people may conceptualize different groupings
  - The division is a design decision, with no right or wrong answer
  - Whatever decisions are made, the result should enhance readability
- The notation for a package is a folder icon
  - The contents of a package can be optionally shown in the body of the folder

Packages

- Illustrates the notation with the music industry example at a very high level
  - Music is created and placed on Media
  - The Media is then Distributed

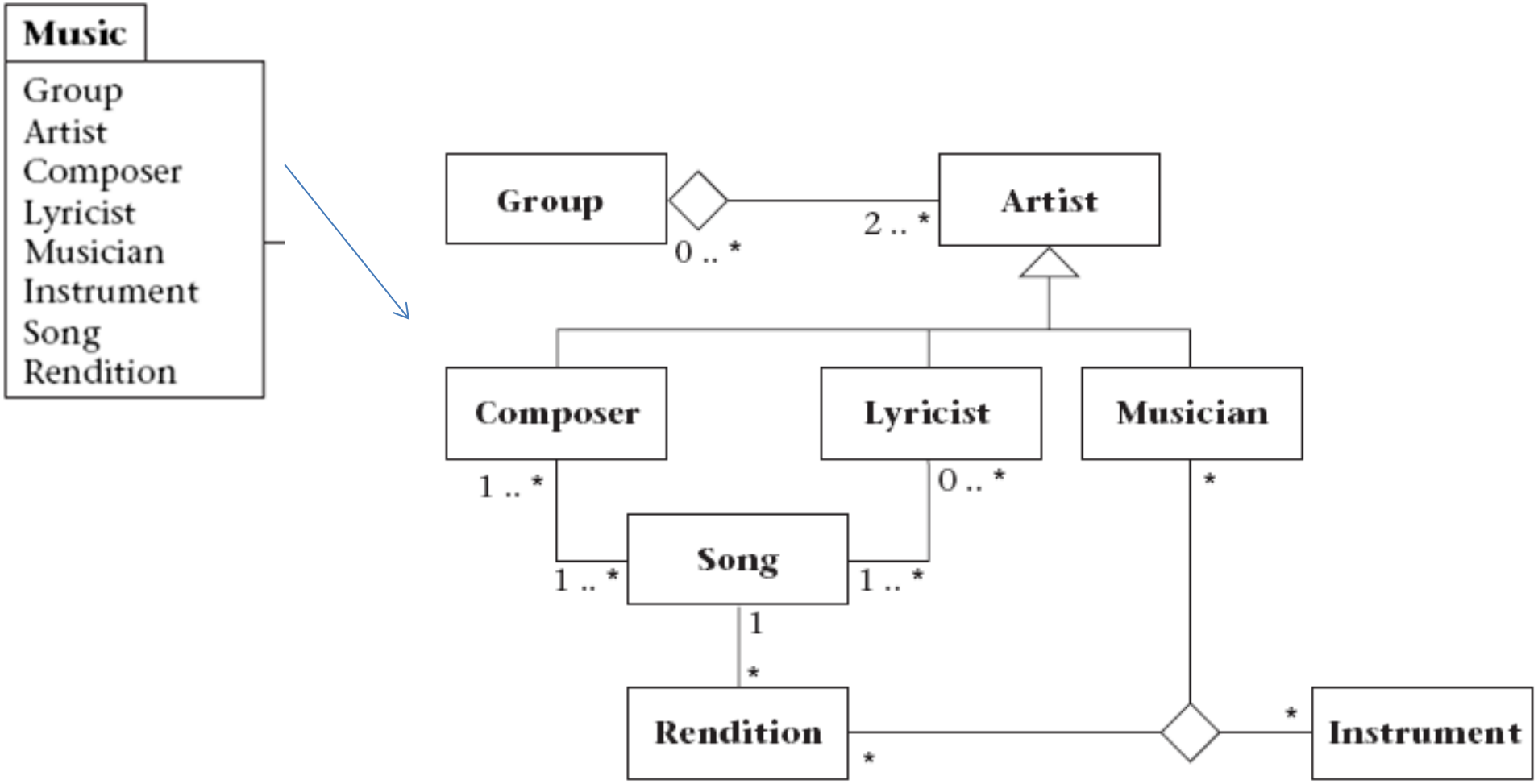
Packages



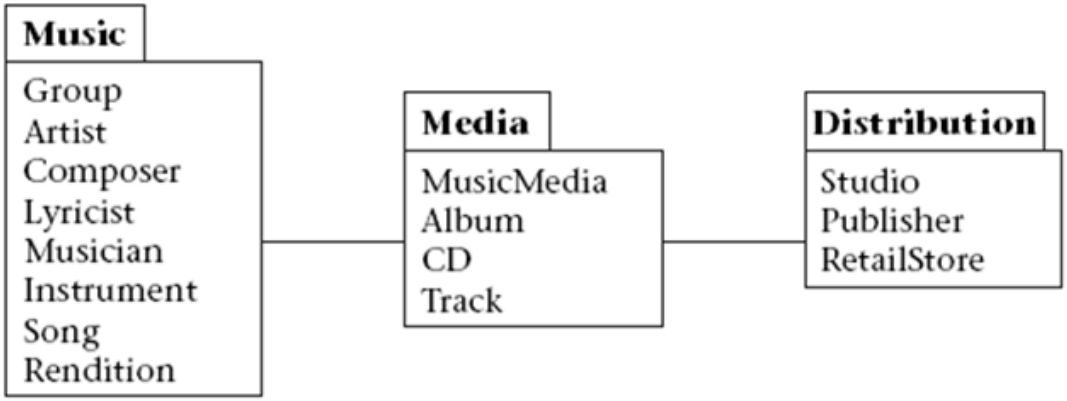
- The music industry is illustrated with the classes listed
  - The Music package contains classes that are responsible for creating the music
  - The Media package contains classes that physically hold the recordings of the music
  - The Distribution package contains classes that bring the media to you

Package

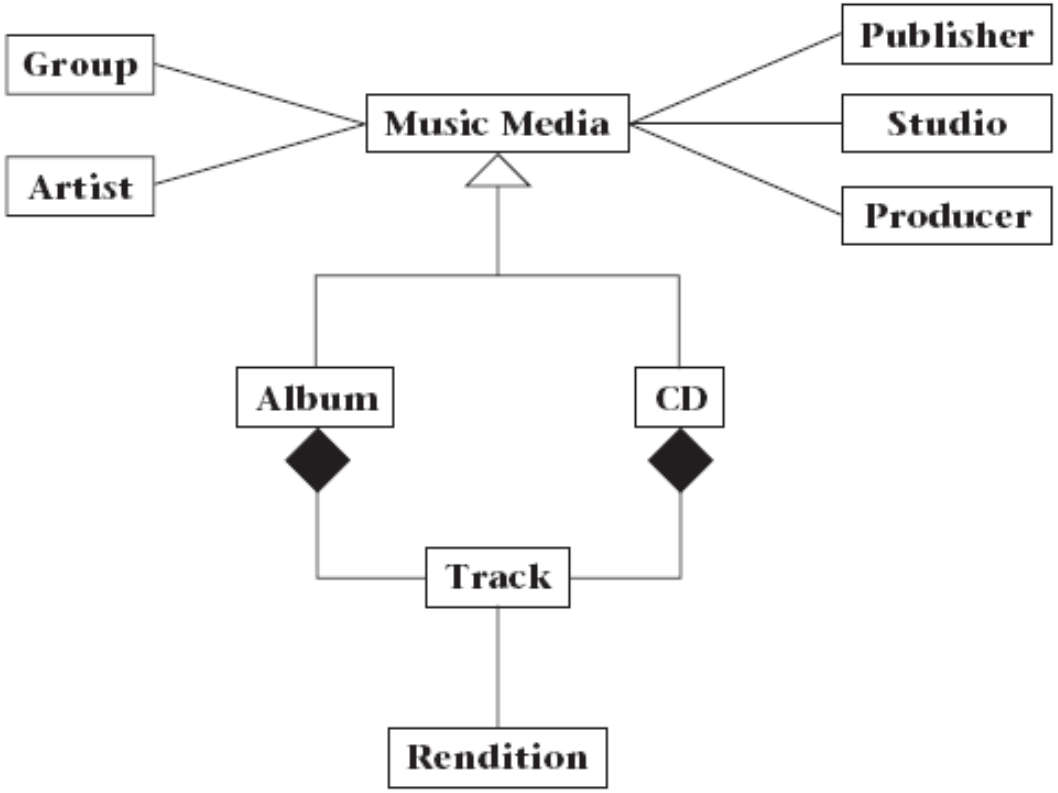
- The contents of a package can be expanded into greater detail
- The relationships of the classes within the Music package are illustrated as follow



Package



- A system can be understood more easily by shifting focus to each package in turn



- The associated classes from the Music and Distribution packages are also shown
  - Detailing how the Media package is related to the other two packages

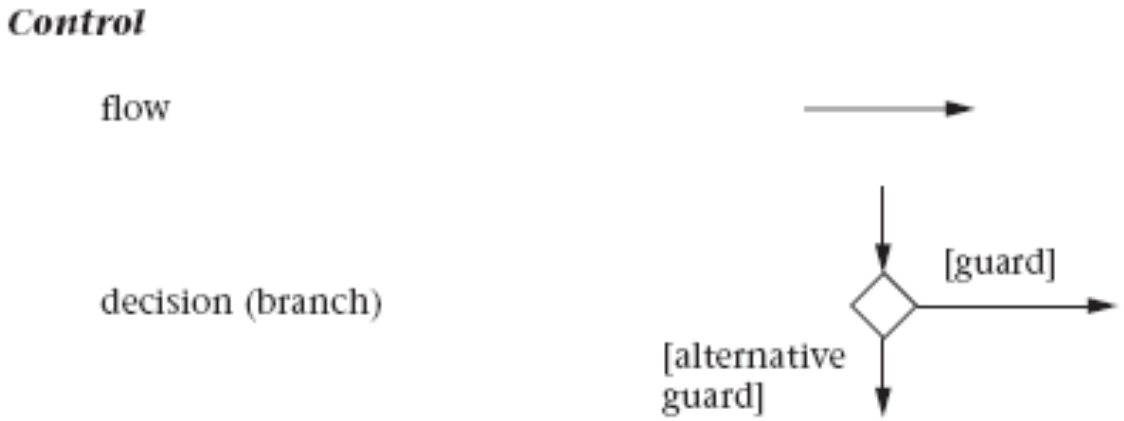
Activity Diagrams

- For specifying the activities and the flow of control in a process
- Process
  - May be a workflow followed by people, organizations, or other physical things
  - Alternatively, may be an algorithm implemented in software
- Process in the logical design of database: Workflow

Notations in Activity Diagram: Nodes

- Initial node, final node, and activity node
  - Any process begins with control residing in the initial node
- Final node
  - The process terminates when control reaches a final node
- Activity node
  - States where specified work is processed
    - ex: "Generate quote"
  - The name of an activity is typically a descriptive verb or short verb phrase

Notations in Activity Diagram: Control



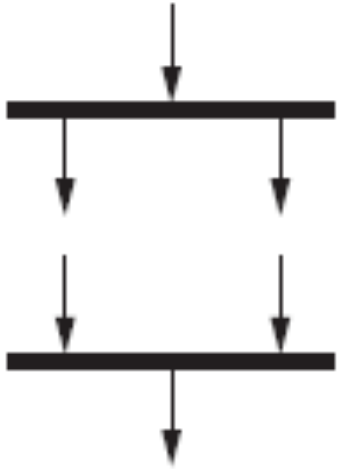
- Flow: Control flows in the direction of the arrow
- Decision
  - A hollow diamond with multiple outgoing flows
  - Each flow from a decision node must have a guard condition
    - Ex: [acceptable], [unacceptable]
    - Control flows in exactly one direction from a decision node and only follows a flow if the guard condition is true
    - The guard conditions associated with a decision node must be mutually exclusive
    - The guards must cover all possible test conditions, so that control is not blocked at the decision node
  - [else]: then control flows in that direction only if all the other guards fail

Notations in Activity Diagram: Control

Control

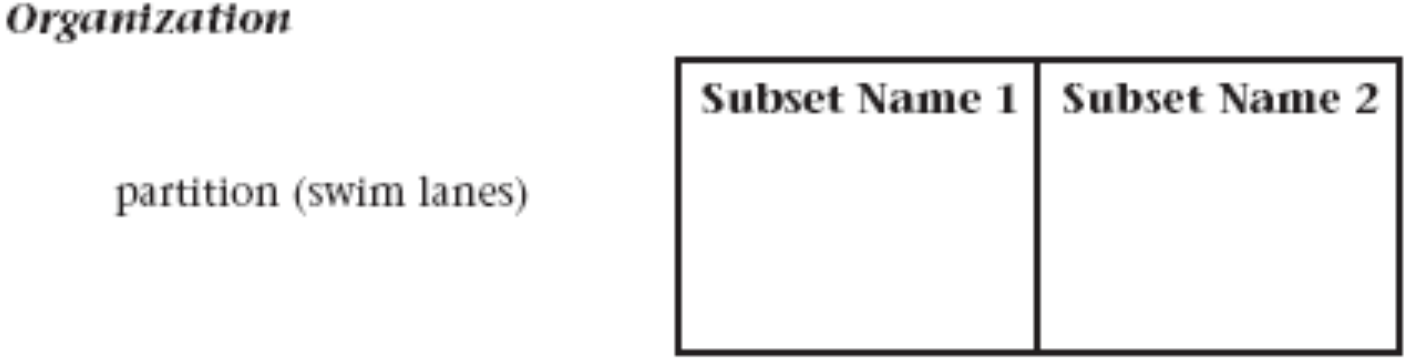
fork

join



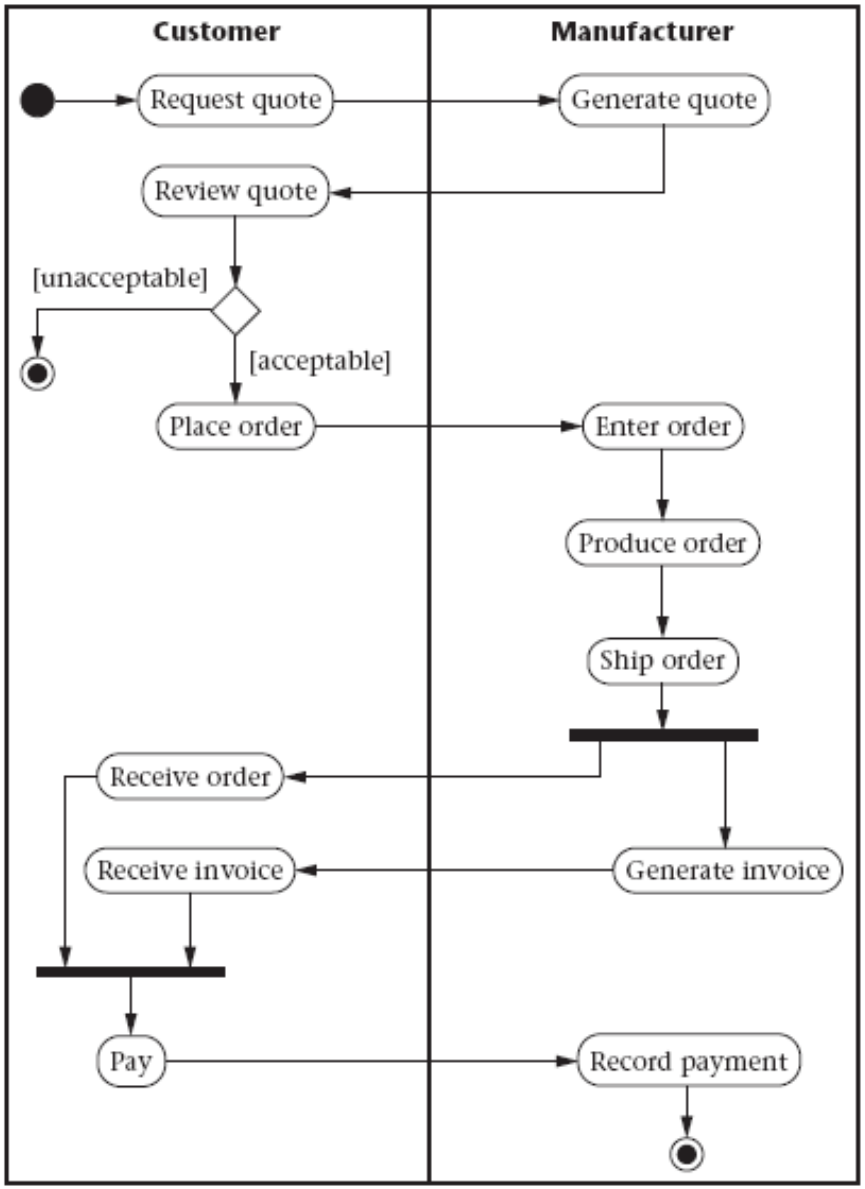
- Fork and Join
  - Both forms of synchronization
- Fork
  - When control flows to a fork, the control concurrently follows all the outgoing flows ← Referred to as concurrent threads
- Join
  - Has multiple incoming flows and one outgoing flow
  - Control flows from a join only when control has reached the join from each of the incoming flows

Notations in Activity Diagram: Organization



- Activity diagrams may be further organized using partitions known as swim lanes
- Partitions split activities into subsets, organized by responsible party
- Activity partitions may be arranged vertically, horizontally, or in a grid.

UML activity diagram, manufacturing example



Rules of Thumb for UML Usage

- Keep each UML diagram to one page
  - You should divide and organize your content into reasonable, understandable portions
  - Use packages to organize your presentation
- Use UML when it is useful
- Accompany your diagrams with textual descriptions
  - Combining natural language with UML is effective
- Take care to clearly organize each diagram
  - Avoid crossing associations