Domain Modelling

Jörg Kienzle & Alfred Strohmeier

Domain Model Overview

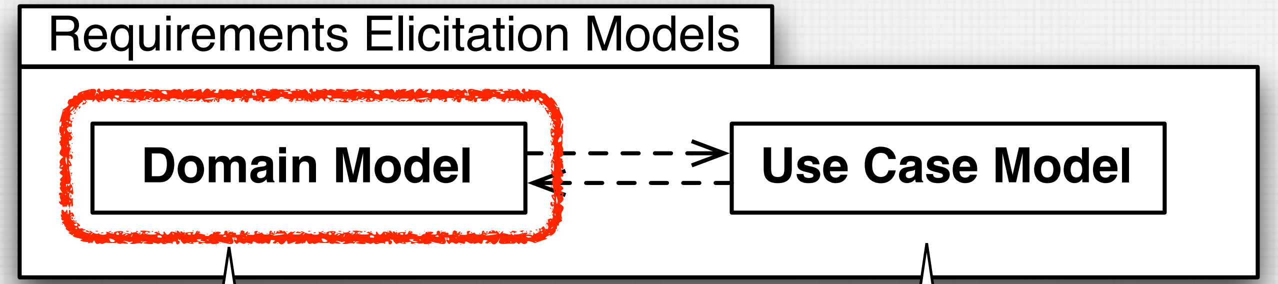
- Analysis Objects and Classes
- Association
- Multiplicity
- Aggregation
- Composition
- Building a Domain Model
- Structuring a Domain Model

Requirements Elicitation Activity

- Discover the requirements of the system to develop
  - Functional requirements
  - User expectations
  - Non-functional requirements / qualities
    - Distribution
    - Security
    - Safety
    - Reliability
    - Fault Tolerance
    - Availability

Requirements Elicitation Models

Start with either one, or establish them simultaneously

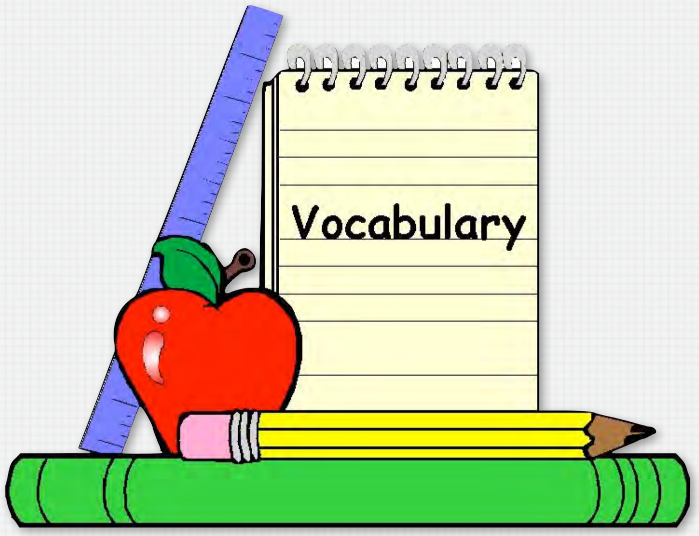


UML Class Diagram, describing the concepts of the problem domain and their relationships

UML Use Case Diagram + textual template, describing the different ways users / stakeholders interact with the system

Domain Model Definition

- The Domain Model captures the concepts in the domain of the problem, and the relationships between them.
- It establishes the vocabulary of the problem domain.



Objects during Requirements Phases

- An object is a thing or concept that can be distinctly identified, e.g. a specific person, organization, machine, or message
- An object can have static properties, called attributes. The values of the attributes can change, but not their number and names
- Attributes of an object during requirements elicitation or analysis are not allowed to be objects
- An object during requirements elicitation or analysis does *not* have operations/methods

Finding Attributes Question

- Find some typical attributes for the following classes:
  - A person who might receive a letter
  - A person suspected in a criminal case
  - The person the phone bill is sent to
  - An employee of a company using a security access card

Associations (1)

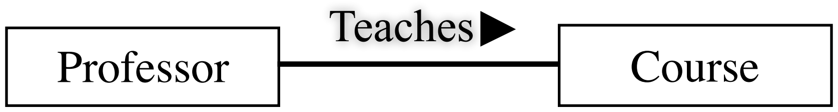
- Associations are the "glue" that holds together an object-oriented system
  - Without associations, there is only a set of unconnected classes.
- An association models a relationship between objects
- The existence of an association is conditional: all related classes must exist
- Similarly, an occurrence of an association can only exist if the connected objects all exist

Association: Mathematical Definition

- An association between objects corresponds to the mathematical concept of a relationship R between the classes (sets) X_i to which the objects x_i belong.
- The relationship R is a subset of the cartesian product of the sets X_1, X_2, ... X_n.
- The elements of R are called the occurrences of the relationship or the links of the association.
- The relationship is said to be true for the objects x_1, x_2, ... x_n if and only if (x_1, x_2, ... x_n) belongs to R.
- The sets X_1, X_2, ... X_n are called the domain of the relationship and *n* is its arity.
- The sets X_i are not necessarily different.

Association in UML

- An association is represented in UML as a line connecting two (or more) classes
- The name of the association is placed “on” or “beneath” the line
- Optionally, a black triangle indicates how to read the association relationship



Association Occurrences

- The links of an association can be represented by a table of tuples
- Each link is uniquely identified by the objects it connects together. In the table, there cannot be any identical tuples

Teaches	
Professor	Course
Bruce	Percolation Theory
Bruce	Polyhedral Computation
Jörg	Object-Oriented Software Development
Jörg	Software Fault Tolerance
Luc	Probabilistic Algorithms

Association Question

- The printer Neo, of type Phaser 4400N, made by Xerox, located in room McConnell 322...
- Mr. Rich, business man, 42 years old, living in Zug, Switzerland, married with Mrs. Dufour, ...
- Mr. Rich owns an account with the Swiss Union Bank.
- A flight corresponds to an aircraft flying a route on some weekday.

Association Multiplicity (1)

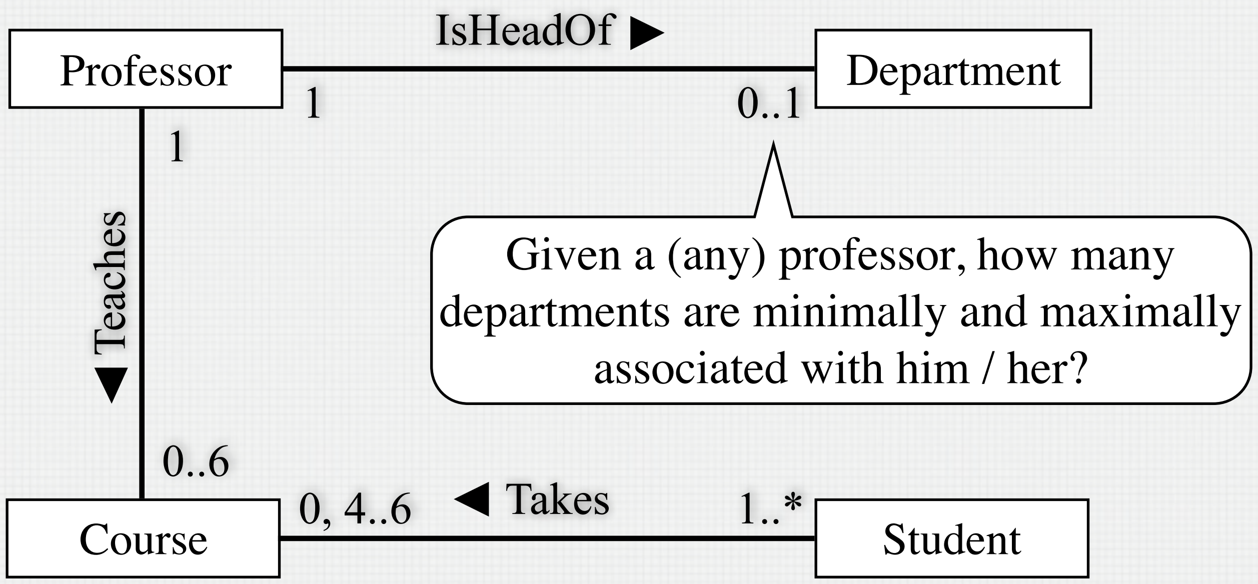
- Consider a tuple $(x_2, \ldots x_n)$. We may ask if there are any objects x_1 in X_1 such that $(x_1, x_2, \ldots x_n)$ belongs to R, and how many they are
- Their minimal and maximum numbers are called the multiplicity (cardinality) of X_1 in the association R

- card(X_1 in R) = (min, max)
  card($\{(x_2, \ldots x_n) \mid (x_1, x_2, \ldots x_n) \in R\}$)

Association Multiplicity (2)

- The multiplicity (or cardinality) defines the number of objects which are, at any given time, allowed to be associated with each other in an association
  - Minimally and Maximally
- Multiplicity is shown by annotating the ends of the association line, called the association end
- The full form is a range (or even a set of ranges), e.g. 0..1, 1..1, 1..4, 0..*, 1..*, an asterisk * meaning that the upper limit is unlimited
  - Sometimes short forms are used: 1 for 1..1, 2 for 2..2, etc., and * for 0..*.

Association Multiplicity in UML



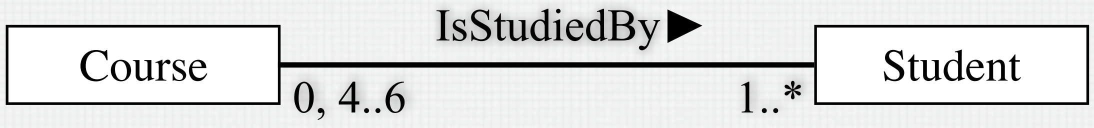
IsHeadOf ▶
Professor
Department
1
0..1
1
Teaches ▶
0..6
Course
0, 4..6
◀ Takes
1..*
Student
Given a (any) professor, how many departments are minimally and maximally associated with him / her?

Works For Question

- Develop a Domain Model, including multiplicities, for a system where a person works for at most one company

Association Occurrences Question (1)

- Look carefully at the domain model below and at the object and association instances shown in the tables on the next slide. Do they implement the domain model correctly? Find 3 errors.



Association Occurrences Question (2)

Course
Percolation Theory
Probabilistic Algorithms
Software Fault Tolerance
Computational Geometry
Object-Oriented Software Development
Polyhedral Computation

Student
Erin
Sadaf
Wisam
Onur

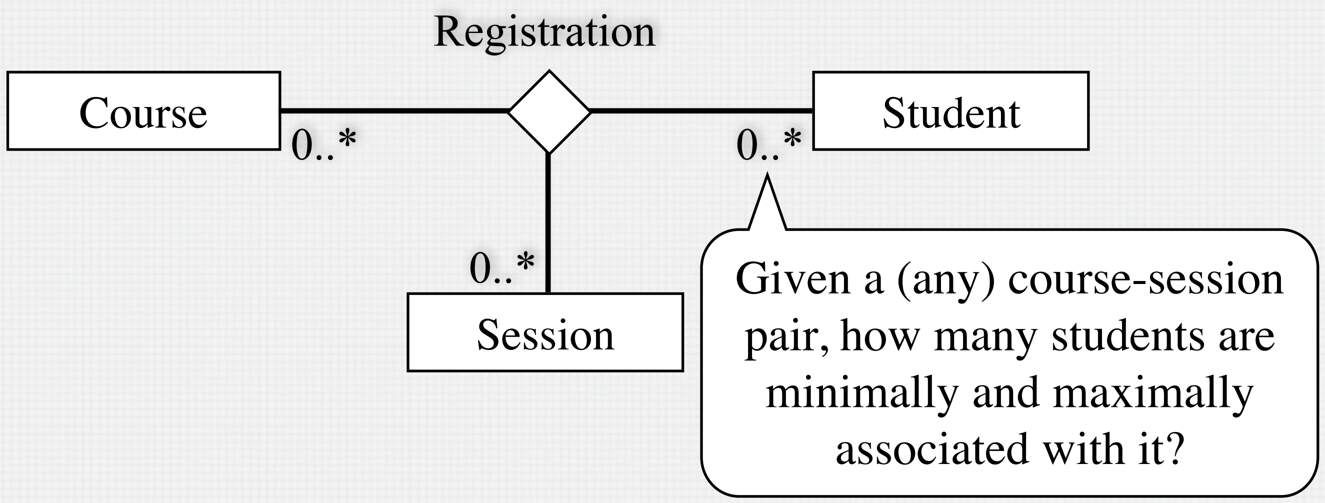
IsStudiedBy	
Percolation Theory	Erin
Software Fault Tolerance	Onur
Probabilistic Algorithms	Erin
Polyhedral Computation	Erin
Software Evolution	Onur
Probabilistic Algorithms	Onur
Computational Geometry	Erin
Software Fault Tolerance	Sadaf
Computational Geometry	Onur

Higher Order Associations

- The arity of an association is the number of objects that participate in each occurrence of the association.
- Most associations are binary.
- The semantics of higher-order associations is often unclear.
- Higher-order associations can often be decomposed into binary associations.

Higher Order Association Example

- A test is taken by a student during a session, where he finally gets a mark. Note that the same test may be taken by the same student at several sessions.
- For a student and a session there are zero or several tests. For a test and a session there are many students.

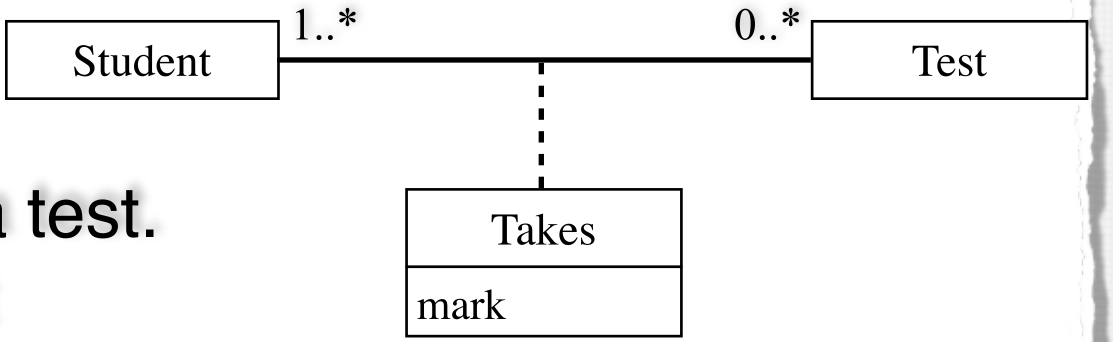

Registration
Course
0..*
Student
0..*
0..*
Session
Given a (any) course-session pair, how many students are minimally and maximally associated with it?

Association Class

- An association class is an association that is also a class.
- An association class has both association and class properties: it connects two or more classes, and it also has attributes and sometimes operations
- Like for all associations, the identity of an occurrence stems from the connected objects
- In UML, an association class can participate in an association.
  - We do not recommend this practice.

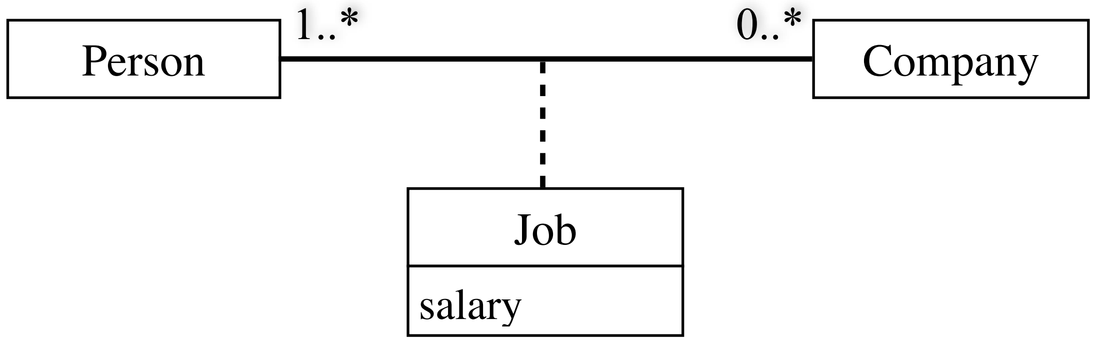
Association Class Example (1)

- An association class is used when an attribute goes with the association rather than with any of the connected classes.
- We want to keep track of the mark a student has got for a test.
- The attribute does not belong to the student, because a student may take many tests.
- Neither does it belong to a test, because many students take the same test.



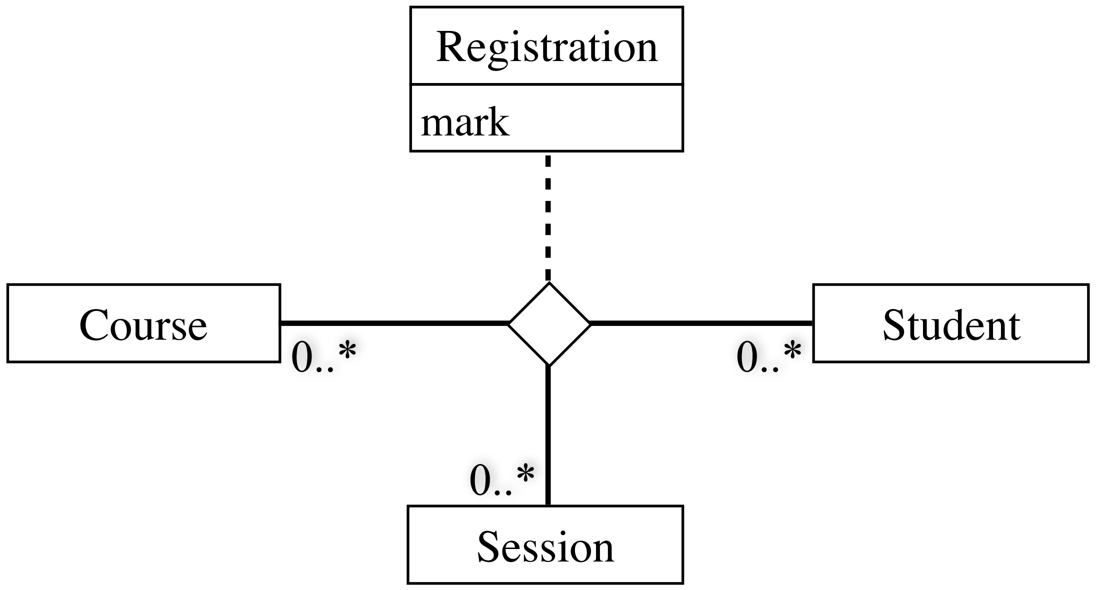
Association Class Example (2)

- A person works for a company, and gets a salary for this job.
- The name of an association class is often a noun, rather than a verb.



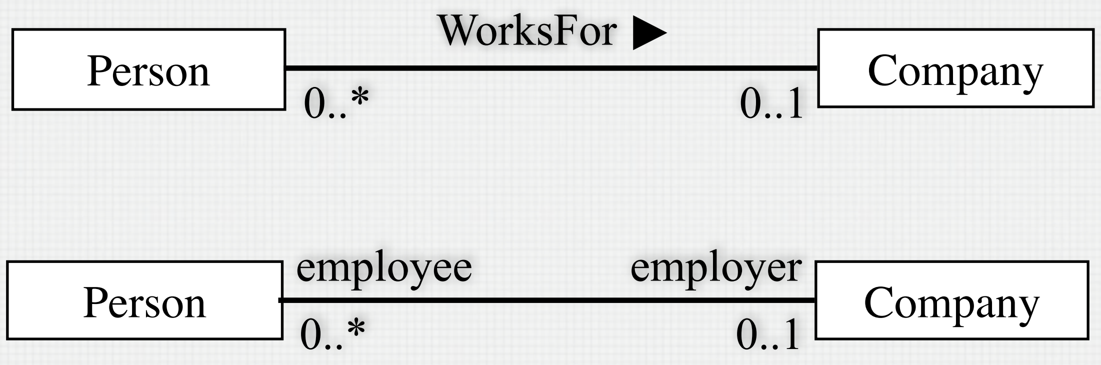
Higher Order Association Class

- A test is taken by a student during a session, where he finally gets a mark. Note that the same test may be taken by the same student at several sessions.



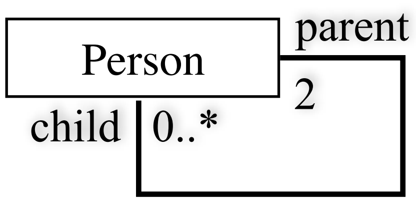
Association Roles (1)

- An association need not have a name. Usually, role names are more convenient because they avoid the problem of which way to read the name and they provide names for navigation and code generation



Association Roles (2)

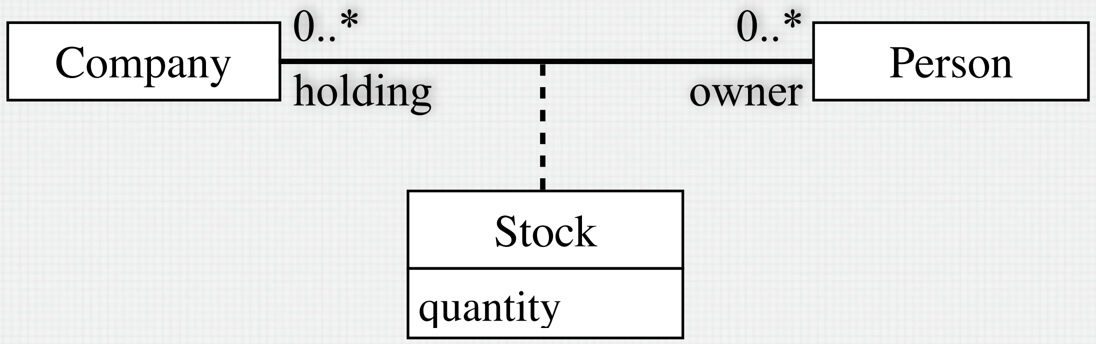
- Role names clarify the semantics of an association, especially when reflexive.



- As a child, a person has exactly two parents.
- As a parent, a person may have several children.

Association Roles Examples

- People hold quantities of shares of companies.



Aggregation (1)

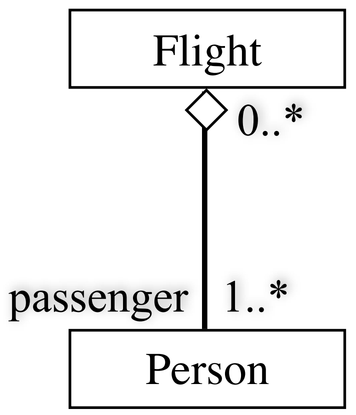
- Aggregation is a form of binary association that specifies a whole-part relationship between an aggregate (a whole) and a constituent part.
- The aggregation relationship is transitive and antisymmetric across all aggregation links, even across those from different aggregation associations.
  - If a is part of b and b is part of c, then a is part of c (transitivity).
  - If a is part of b (directly or indirectly), then b cannot be part of a (antisymmetry).

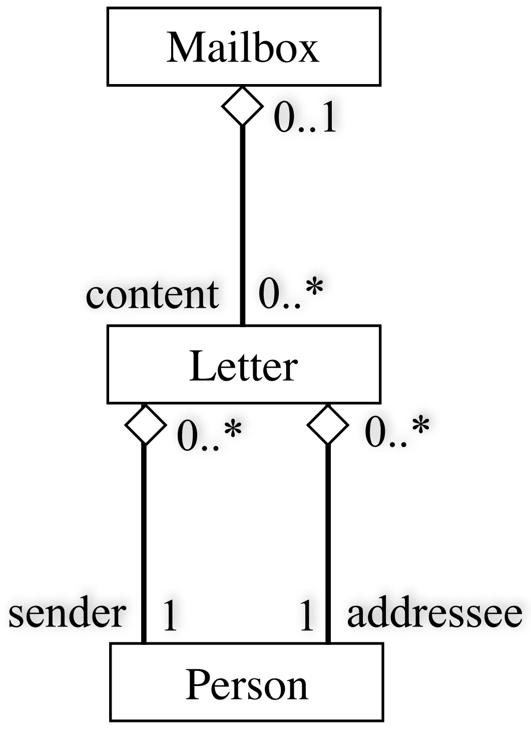
Aggregation (2)

- A part can belong to more than one aggregate, and it may exist independently of the aggregate.
- Often the aggregate needs the parts, i.e. it is a collection of the parts, but the parts can exist by themselves without being regarded only as parts.
- Example
    - A path is an ordered set of segments. A segment can belong to several paths. The path "needs" its segments.
- The distinction between a plain association and an aggregation is often a matter of taste.

Aggregation Examples

- Aggregation is graphically represented with an empty diamond on the association end of the aggregate



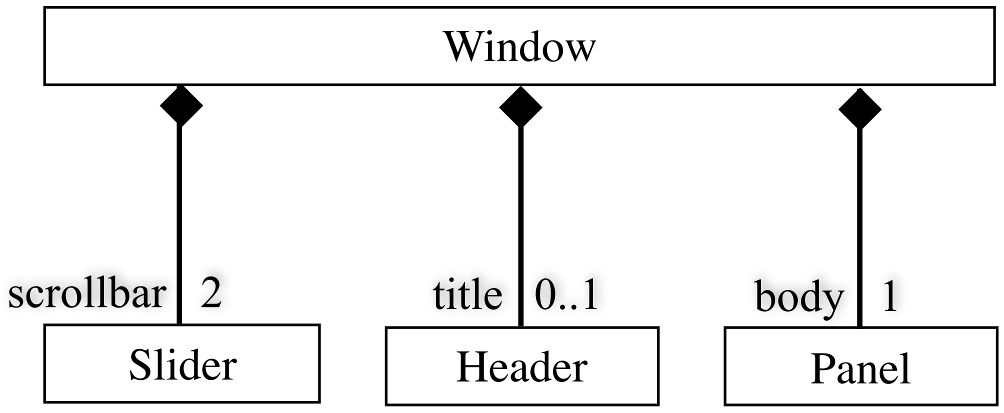


Composition

- Composition is a strong form of aggregation. It has more specific semantics that correspond to physical containment and various notions of ownership.
- A part may belong (directly) to only one composite (at a time). The multiplicity of the composite element must therefore be 1 or 0..1.
- The composite object has responsibility for the disposition of all its parts, i.e. for their creation and destruction.

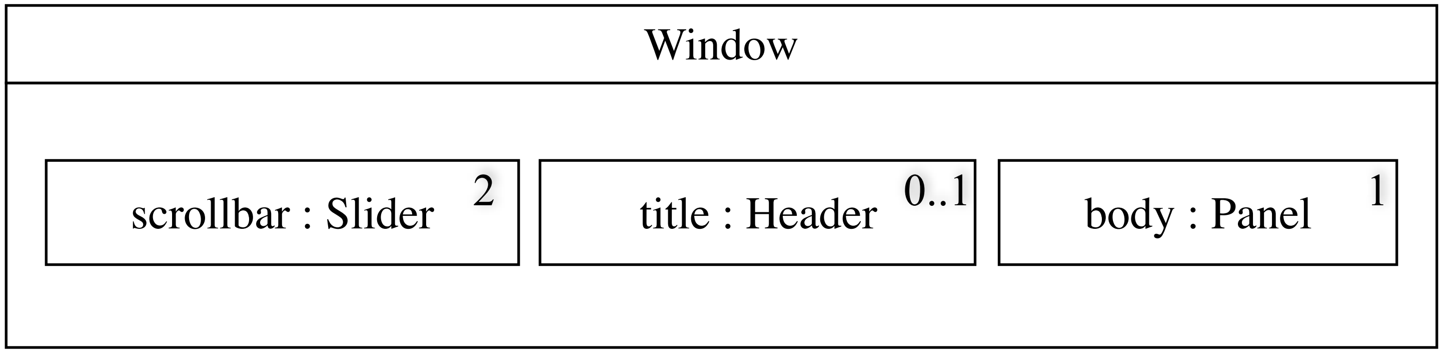
Composition Example (1)

- Composition is graphically represented with a filled diamond on the association end of the composite

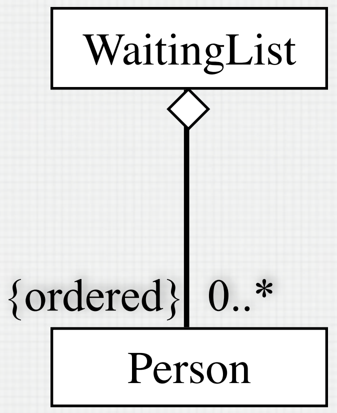


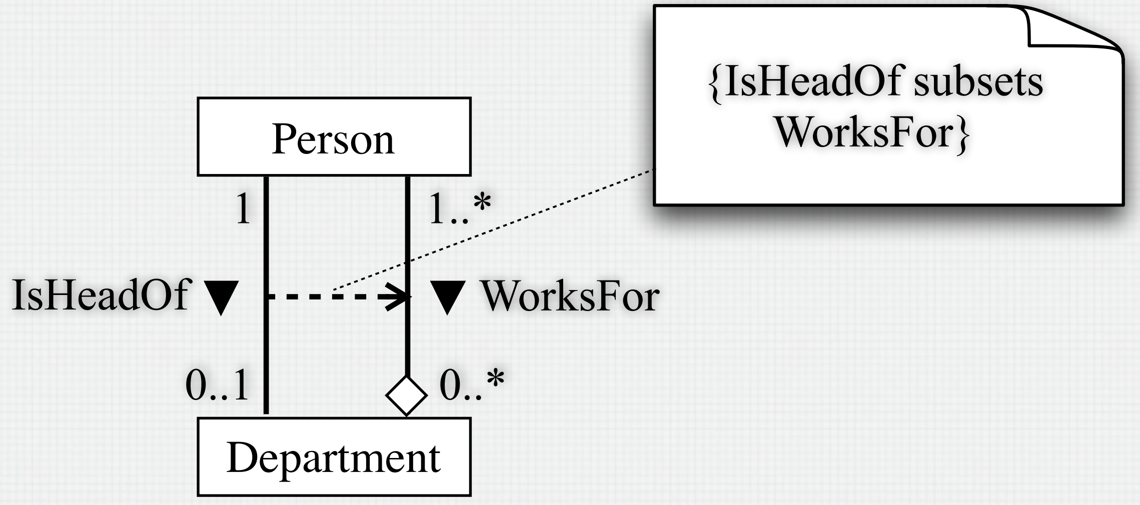
Composition Example (2)

- Composition may also be shown by graphical nesting. A nested element's role name is written in front of its class name

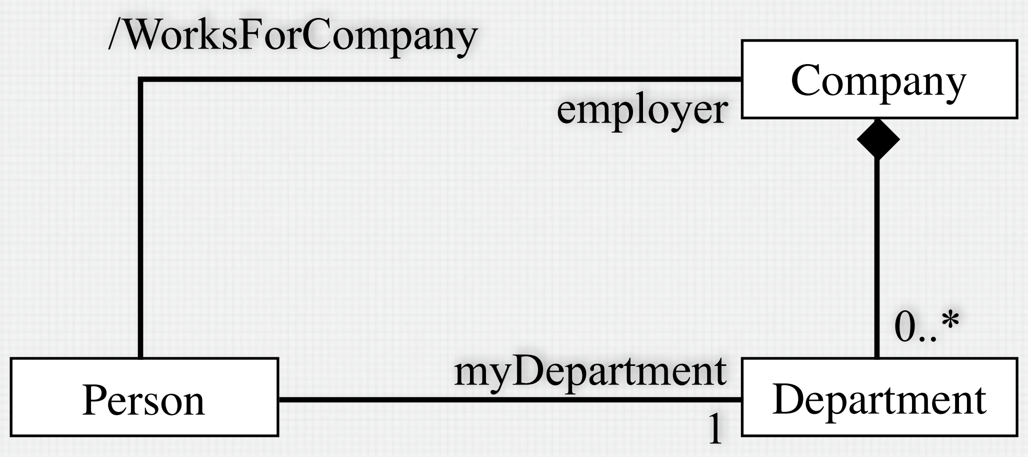


Associations with Constraints



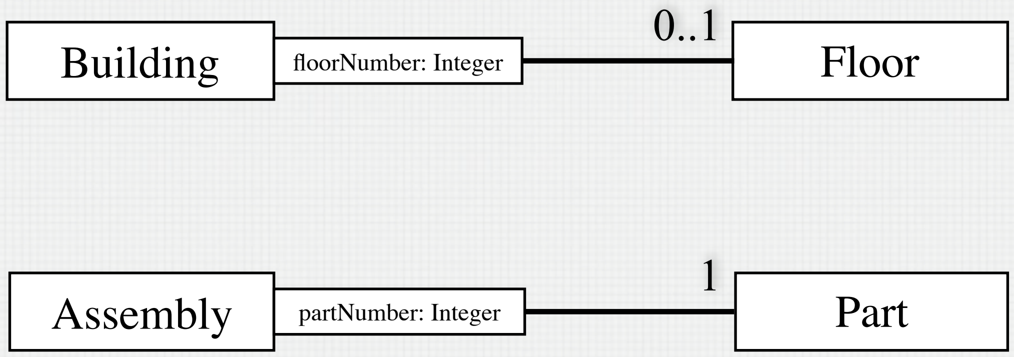


Derived Association



{context: Person
self.employer = self.myDepartment.company}

Qualified Association



Employee Question

- Using a Class Diagram, show the generalization and aggregation relationships between: personnel, woman, employee, man, personnel manager, usher, president, person, company.

Relationships Question

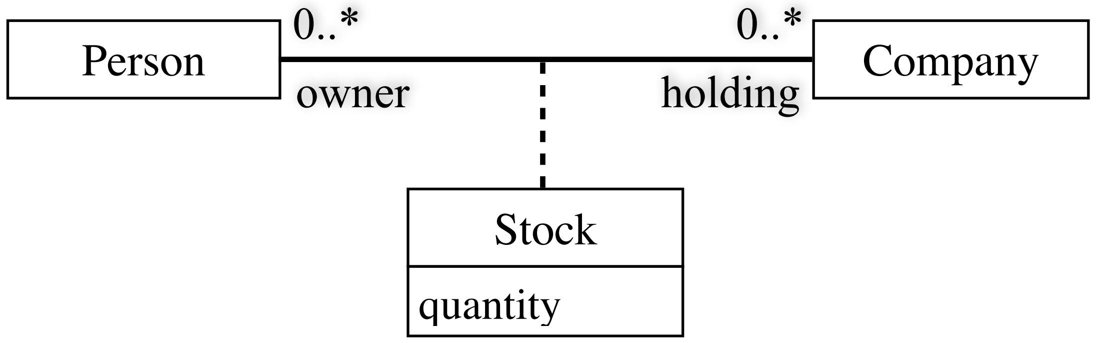
- Decide whether the statement pertains rather to an association, an aggregation, a composition or a generalization-specialization relationship?
  - A country has a capital.
  - A file is a regular file or a directory.
  - A file belongs to a directory.
  - Files contain records.
  - A polygon is composed of segments.
  - A person uses a programming language.

Discovering the Domain Model

- Brainstorm a list of candidate objects, classes and associations:
  - Physical objects: car, pen, etc.
  - People, organizations: clerk, employee, company, bank, etc.
  - Places: building, room, etc.
  - Concepts: trajectory, ledger, flight, etc.
- Distinguish objects from classes, and roles from classes.
- Suppress the redundant, irrelevant, vague classes.
- Look at the use cases!

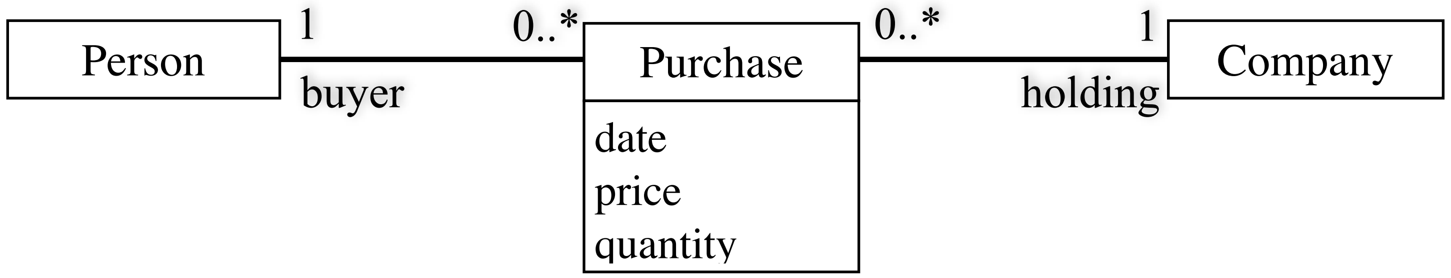
Association versus Class (1)

- People hold shares of companies; there is at most one occurrence of a pair (company, person).



Association versus Class (2)

- People purchase shares of companies. Each purchase has its own date and price in addition to quantity.



Graph and Multigraph

1. Develop a domain model, with multiplicities, for a graph. Recall that a graph is composed of a set of vertices (nodes) and a set of arcs (edges), each arc connecting two vertices. Two vertices are connected by at most one arc!
   - You might also want to show how to model arc lengths or a directed graph.
2. Show how multigraphs can be modelled. Recall that in a multigraph, several arcs might connect the same two vertices.

Association versus Attribute

- An attribute is an association between a class and a set of values.
- An attribute usually does not have multiple occurrences
  - Example: since some professors teach several courses, course is better made a class, and `IsTeacherOf` an association between professors and courses.
- An attribute cannot have attributes; the association class `IsTeacherOf` between professors and courses could have a term attribute showing when the course is given by the professor.
- An association is therefore more flexible, but don’t overshoot…, e.g. no need to model the date of birth by an association between a person and a date.

Part Versus Attribute

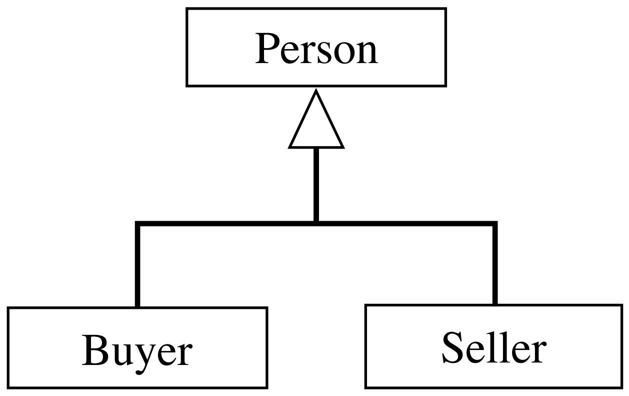
- A pile of plates is made up of plates.
- Each plate keeps its identity, but cannot be used as long as it is in the pile (the parts cannot be shared).
- The destruction of the pile leads to the destruction of the plates.
- The number of plates in a pile varies between zero and many.
- The size of a pile is an attribute, and not a component of the pile.

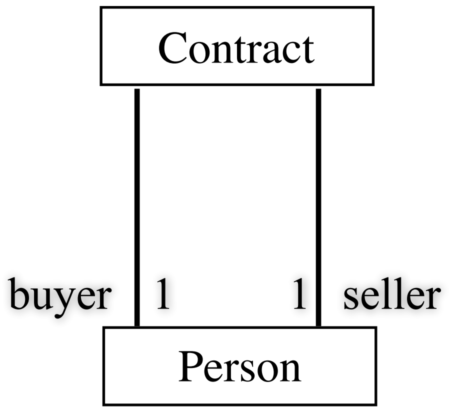
Association / Aggregation / Composition

- A family is an aggregate object made up of the parents and their children.
- The relationships father-child, mother-child, brother-sister are plain associations.
- The contents of a suitcase changes as the user likes; the relationship between a container and its contents is often an aggregation, rather than a composition association.
- Is the liquid content of a bottle really an object?

Class versus Role

- It is not always easy to distinguish between Class / Generalization and Role.

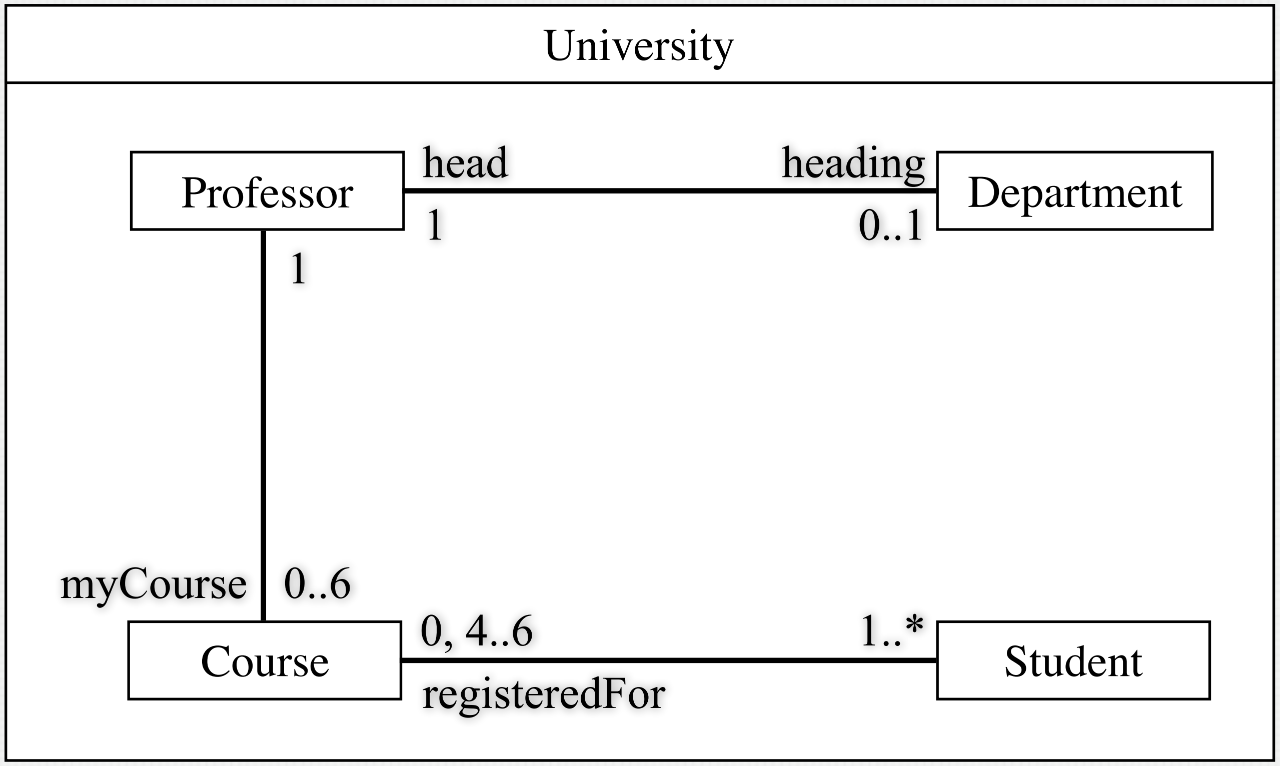




Domain Model Constraints

- Constraints are used to model requirements, assertions, and invariants that cannot be expressed by other means, e.g. by multiplicity in associations
- Constraints are expressed by textual annotations, or by using OCL, the Object Constraint Language

Domain Model Constraints Example (1)



Domain Model Constraints Example (2)

- OCL:

context: u: University
inv: u.professor→**forall**(h, non_h: Professor |
  (h.heading→**notEmpty**() **and**
   non_h.heading→**isEmpty**()) **implies**
     h.myCourse→**size**() < non_h.myCourse→**size**())

What is the English equivalent of this Constraint?

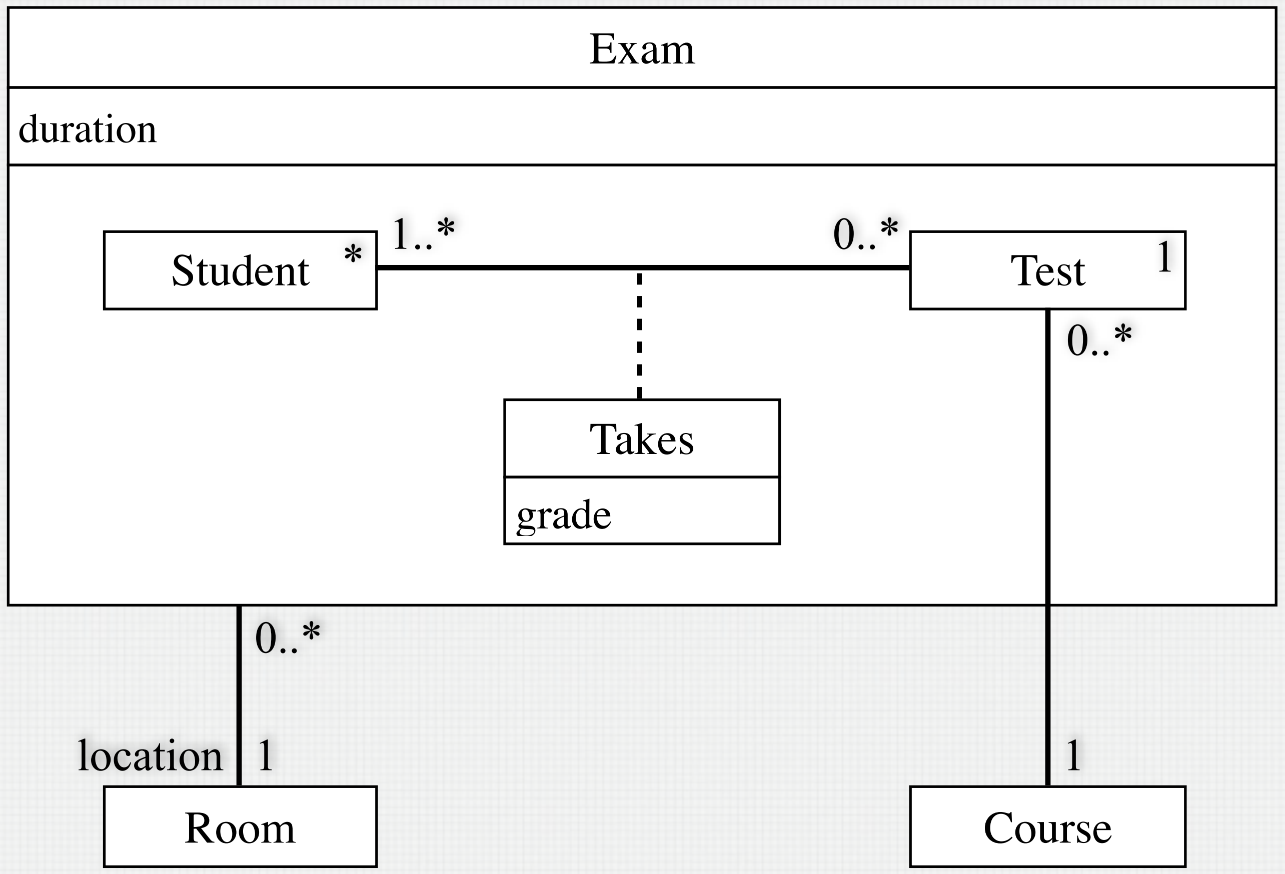
Reification

- The process of making out of a concept an object or class is called reification, meaning “making it an object”.
- An important consequence of reification is that some entity suddenly gets an identity.
- Reification can be applied to associations and connected classes.
- A reified association has all the properties of a class. Especially, it can participate in relationships.
- Reification can be used to structure a class model.

Reification Example (1)

- A flight is a reification of “an aircraft scheduled on some route on some weekday”.
- A person owns a car. A title is a reification of this relationship. A title has an identity, it has attributes on its own, it can be transferred.
- The result of physical composition is naturally an object: e.g. a car is composed of a motor, wheels, etc.
- Reification is often applied to non physical composition: e.g. a department with its personnel, buildings, budget, etc.

Reification Example (2)



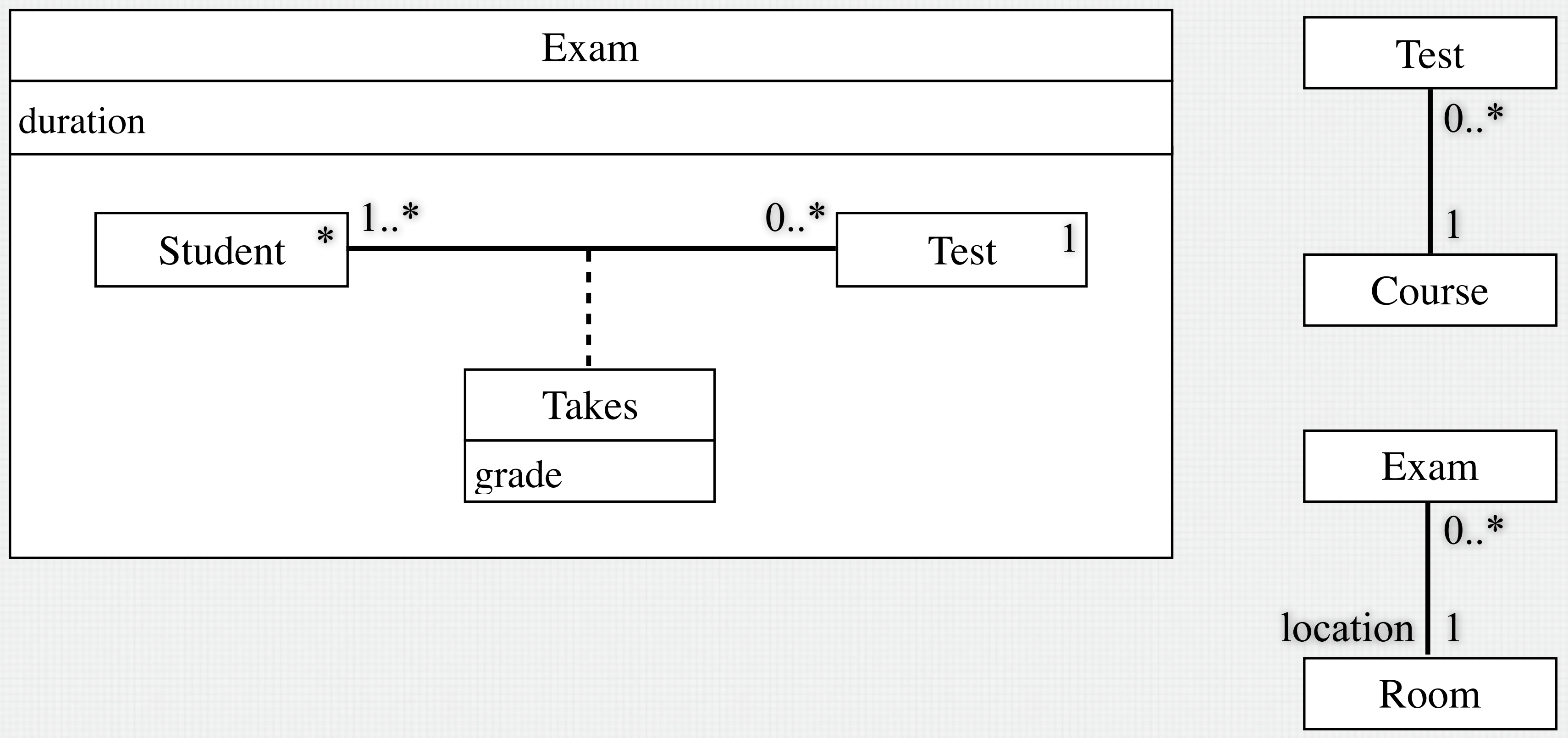
Reification Example (3)

- Exam contains pairs of sets of students and tests belonging to the association `Takes`
- Exam has the attribute `duration`
- An exam is held in a room, which shows that a composition may participate in an association.
- A test examines some course, which shows that a component class can participate in an association.
- Notice how the reification decomposes the ternary association Student-Test-Room into two binary associations.

Structuring a Domain Model

- Often, domain models are too big to fit into one diagram
    - The complete domain model is the union of all the diagrams, together with the related constraints.
    - Two class boxes with the same name denote the same class.
- Reification and composition provide means for levelling the diagrams.
- Packages can be used for defining name spaces, and therefore controlling name scopes.

Structuring Example



Domain Model Development Summary

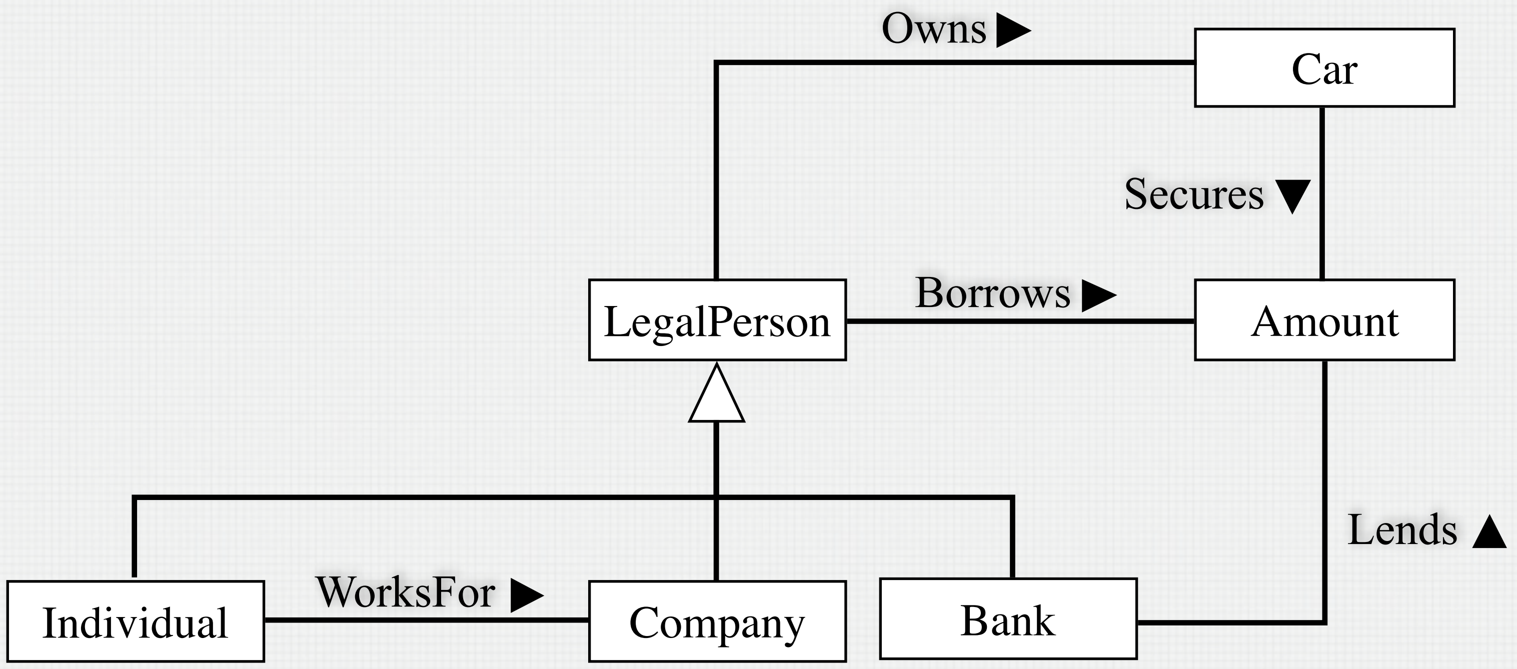
- Develop the Domain Model for the problem domain
  - Brainstorm a list of candidate classes and relationships
  - Incrementally produce the Domain Model, looking for:
    - Generalizations, modelling "kind of" and "is a" relationships,
    - Aggregations, modelling "part of" and "has a" relationships,
    - Associations,
    - Derived associations,
    - Attributes of classes,
    - Attributes of associations,
    - Multiplicities of associations,
    - Constraints, to be recorded in OCL (and/or in text).
  - Look at the use cases, if available

Questions?

Automobiles Question

- Sketch the Domain Model for the following situation:
  - A person can work for one or several companies.
  - A car is owned by a person, a bank, or a company.
  - Banks give loans for buying cars.
  - A loan can be secured against a car.
- Add multiplicities to the associations, and state possible other constraints (invariants).

Automobiles for Discussion



Library Question (1)

The system to build is an automated library book borrowing system that is to be used in the departmental libraries of a university. The goal is to relieve the librarians from processing book loans.

The system does not require users to identify themselves to search for books according to certain criteria and to check the availability of a particular book. However, to check-out books, to check their respective book loan status, and to place holds on books that are already on loan, users must first identify themselves to the system.

Library Question (2)

A single receipt is printed for each user check-out session; it details for each book: the title of the book, the unique identifier of the book, the date the book was borrowed, and the date the book is to be returned by. At the start of each week, the system sends warning emails to all borrowers that have overdue books. All information about what category of member a user belongs to (see borrowing rules below), his/her email address, etc. is available from the central university server.

Books have physically attached barcodes, which are used for the identification of books that are checked out (a barcode scanner is to be used). If the book does not scan, it should be also possible to enter the barcode manually. Book renewals (i.e., extending the due date) are not supported in the current version of the system, but this feature is planned for a later release. It is perceived as an important feature to be supported via the web.

Library Question (3)

These are the borrowing rules:

- Each user has a maximum number of books that are allowed to be on loan at any one time. The limit is dependent on the category of the member, e.g. department heads can take out as many as 100 books, where research assistants are limited to 15 books.
- Often, a certain category of book a different loan period for each category of member, e.g. professors are allowed to borrow a standard book for 6 months, whereas graduate students are allowed only 3 months.
- Different categories of books have different loan periods (e.g., lecturers may borrow a standard book for 3 months, but can only loan a periodical for 1 month). It might even not be possible, for a certain category of member, to borrow certain books (e.g. undergraduate students cannot borrow reference books).
- Books that are on reserve are not available for loan.