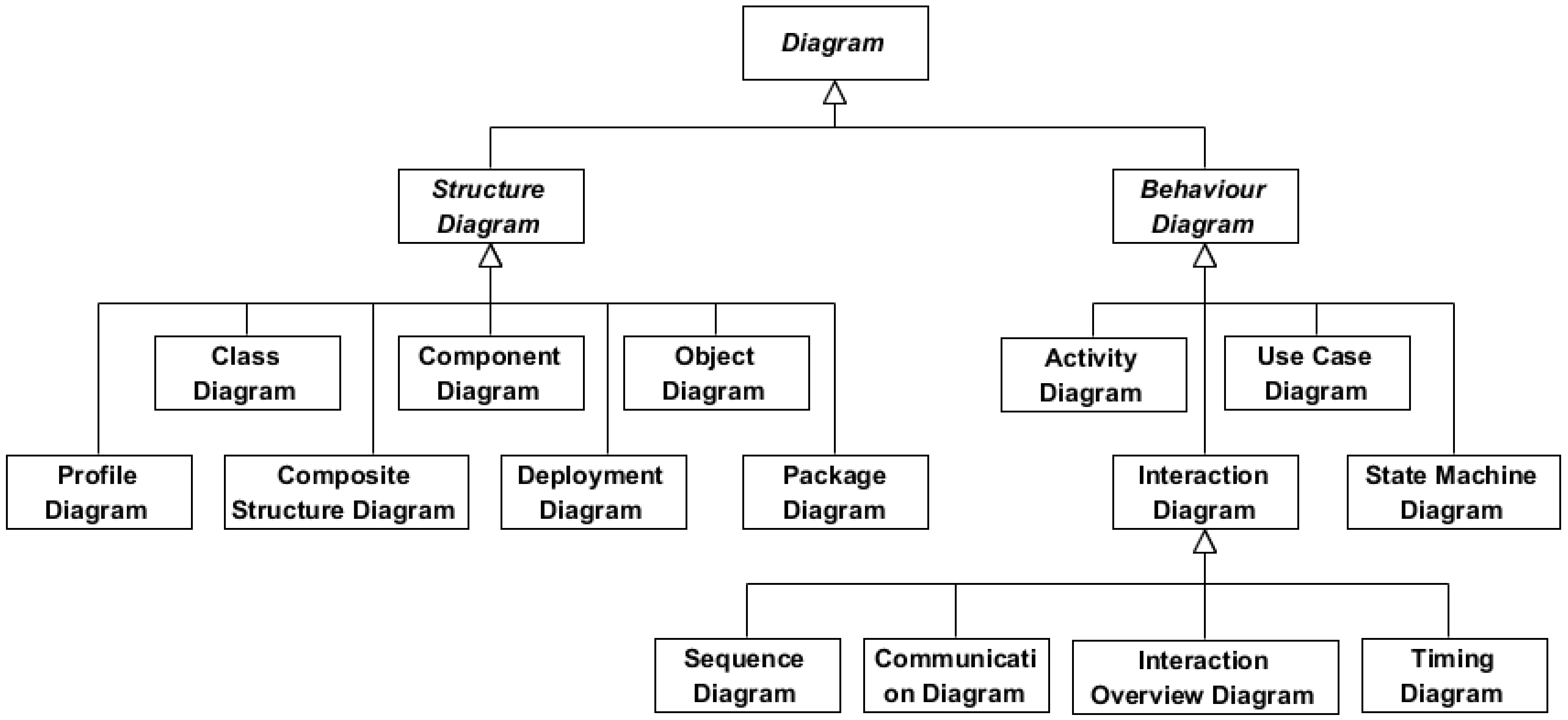
Modelling, introduction to UML and Class modeling

Anastasija Nikiforova

Institute of Computer Science

Based on materials developed by Marlon Dumas, Fabrizio Maria Maggi, Luciano García-Bañuelos



Source: https://www.tekportal.net/unified-modeling-language/

Should you use all of them within a project?

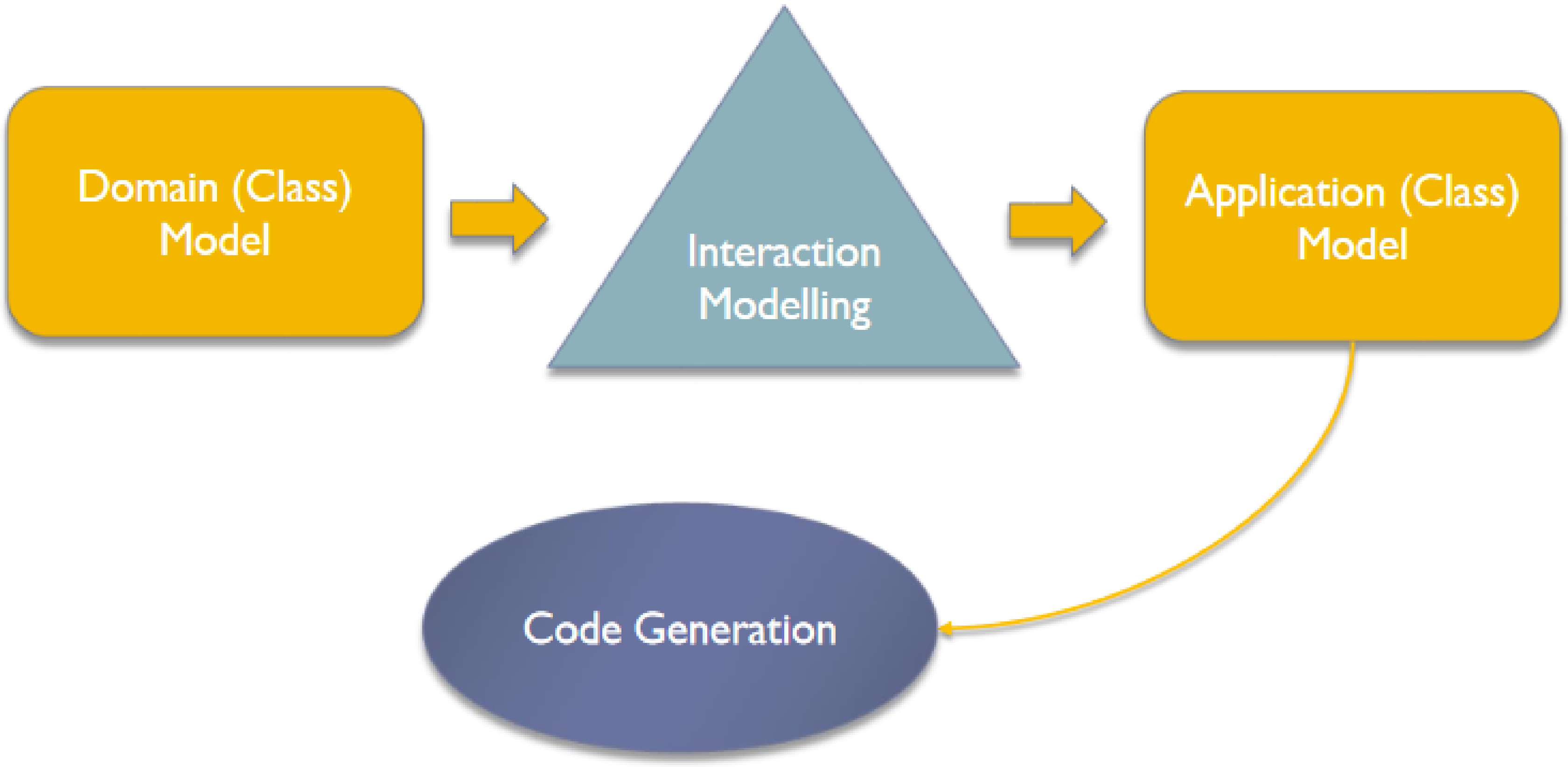
Domain (class) model

Domain (class) model

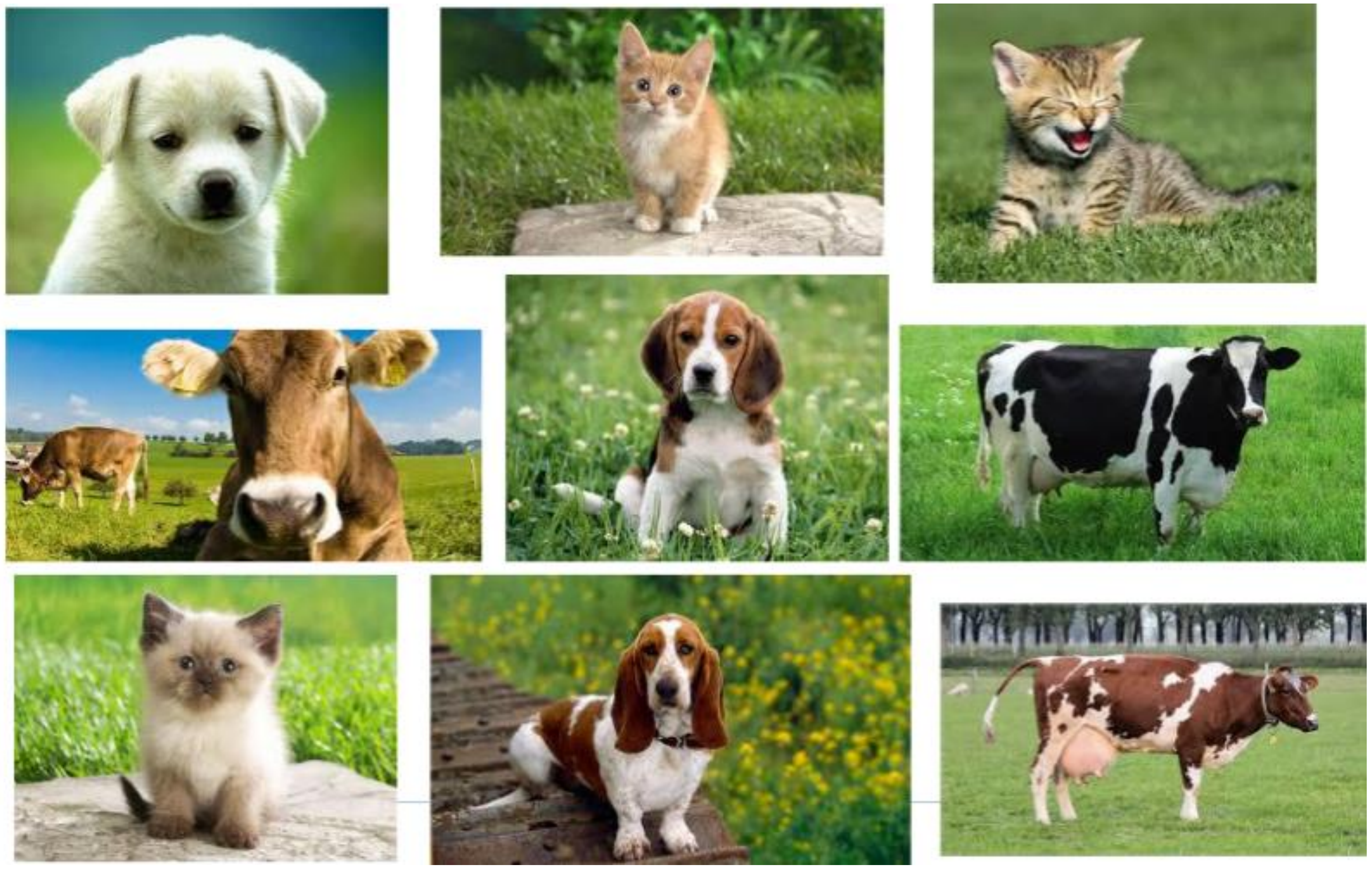
- To answer WHAT ? question, the domain model provides **classes with attributes and relations among them**

- **Operations are not specified**

Software Development Methodology



How many classes? And Instances?



Is there only one correct answer?

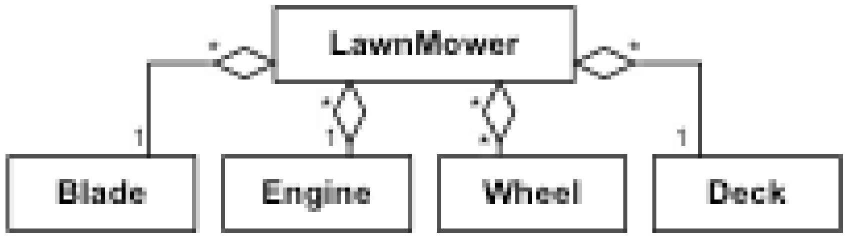
Agenda

- **Aggregation**
- **Composition**
- **Inheritance**
- **Enumerations**
- **Derived data**
- **Multiplicity of attributes**
- **Association classes**
- **Qualified associations**
- **Abstract class**
- **Abstract operations**
- **…**

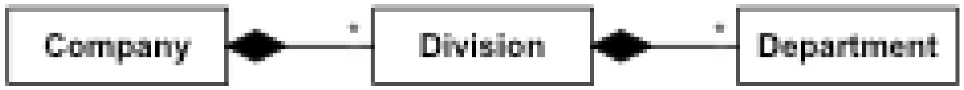
with most of them are familiar, so mostly "to sum up…" and examples!

Aggregation and composition

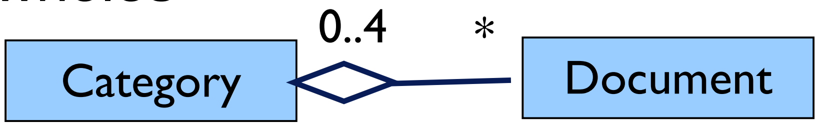
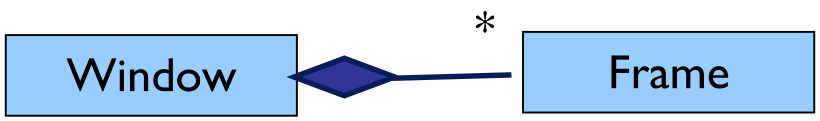
- Aggregation is a special form of association
  - Underlines the fact that an object is made of constituent parts



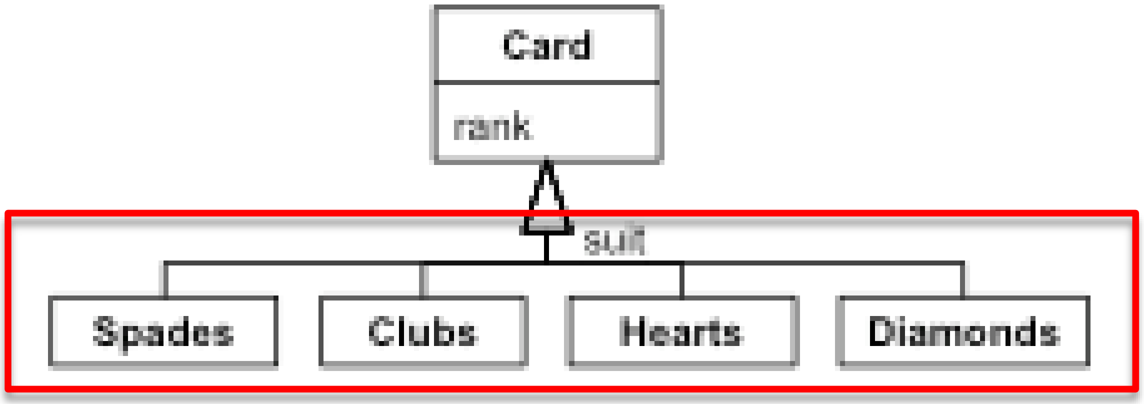
- Composition is a more restrictive form of aggregation
  - Two additional constraints
    - A constituent part can belong to at most one assembly
    - The part has a coincident lifetime as the assembly



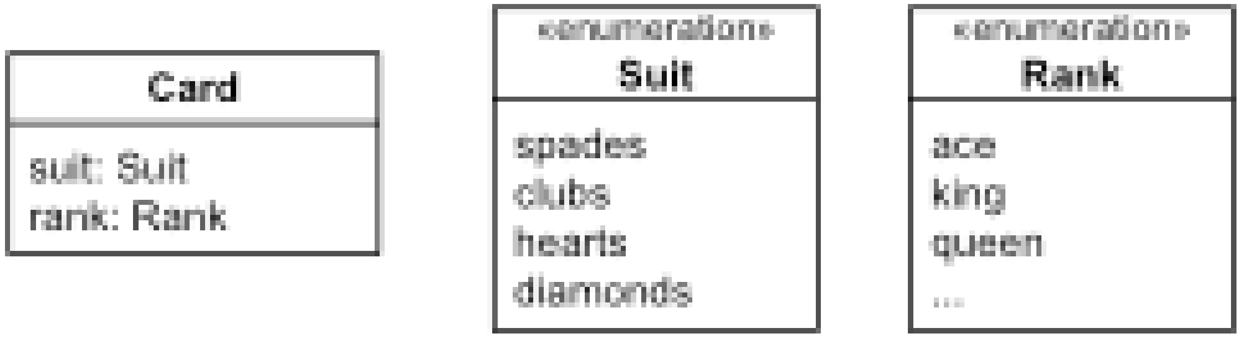
Association versus Composition

Aggregation	**Composition**
Part can be shared by several wholes 	Part is always a part of a single whole 
Parts can live independently (i.e., whole cardinality can be 0..*)	Parts exist only as part of the whole (e.g. when a Window is destroyed all other widgets are also destroyed)
Whole is not solely responsible for the object	Whole is responsible and should create/destroy the objects

Enumerations



An **enumeration** is a data type that has **a finite set of values**. You should avoid modeling enumerations as generalization hierarchies.



Enumerations

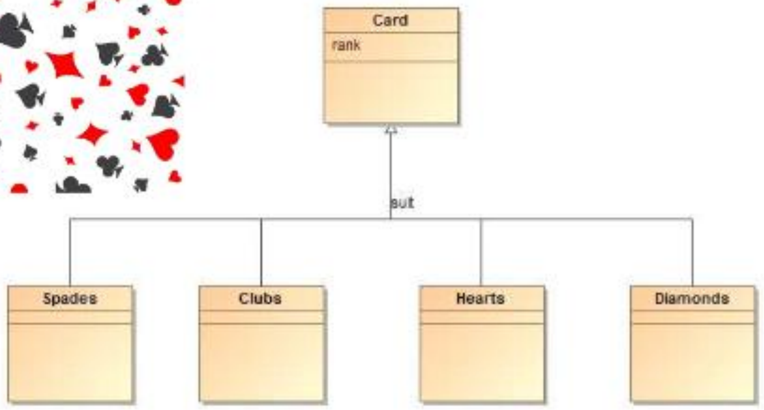
- An enumeration is a data type that has **a finite set of values**.
- **Enumeration is a data type**
  - you can declare an enumeration by listing the keyword enumeration in angle quotes (<< >>) above the enumeration name in the top section of a box.

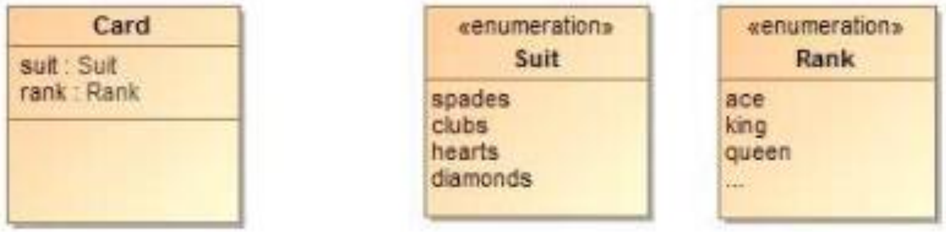
    The second section lists the enumeration values.

****in some cases generalization may seem to be appropriate as well, BUT!*

- **Do not use generalization to capture the values of an enumerated attribute**
  - **An enumeration is a list of values.**
  - **Introduce generalization <u>only</u> when at least one subclass has <u>significant attributes, operations, or associations that do not apply to the superclass</u>.**

Enumerations





preferable

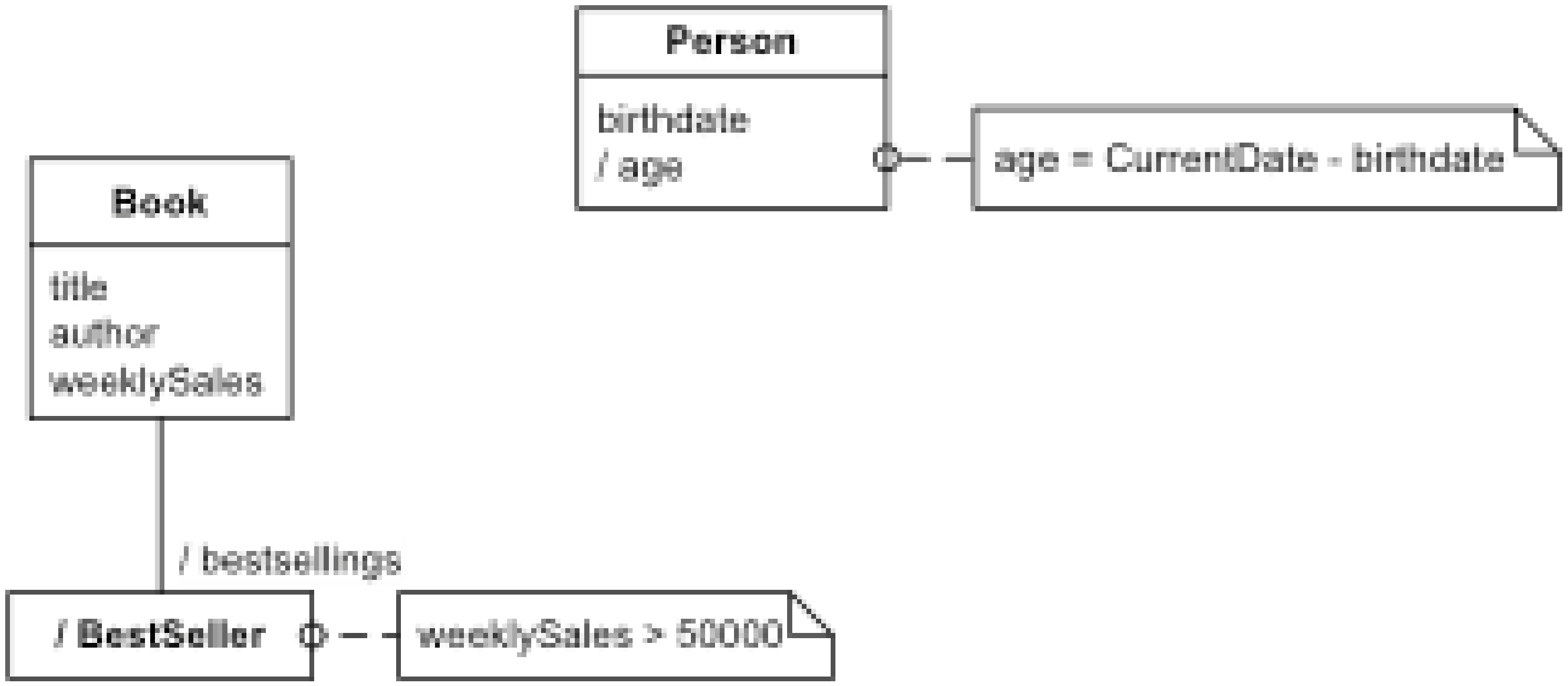
Think of other examples of enumeration! What are their main characteristics?

Enumerations

- ***Spring, Summer, Autumn, Winter***
- ***Monday, Tuesday, Wednesday,…***
- ***January, February, March,…***

Derived data

- A **derived element** is a function of one or more elements, which in turn can be derived



Example

A paper reviewing system has several conferences. Each conference has a title and a year and is managed by a chair and a list of committee members. Committee members and chairs must be assigned to one, but possibly more conferences. They have a name and an affiliation. A conference has several submitted papers, but a paper can be submitted to only one conference. A paper is assigned to 3 reviewers taken from the committee members. A paper can be accepted, rejected or under review. We also know the paper titles and list of authors with their names and affiliations.

Do we have enumerations here?

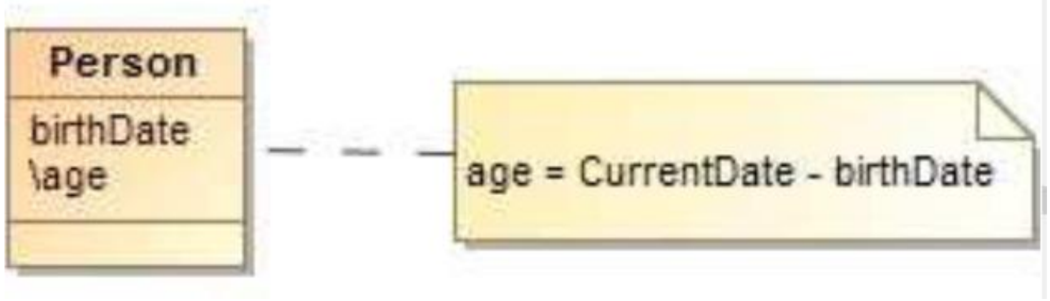
Example

In a system for handling shipments of products in an online bookshop, there can be three different types of items: book, greeting card, stationery item. A book has a title and a list of authors with names. A greeting card has a brand. A stationery item can be a pen, a pencil or a notebook. Each shipment has a priority that can be standard, high, or express.

Do we have enumerations here?

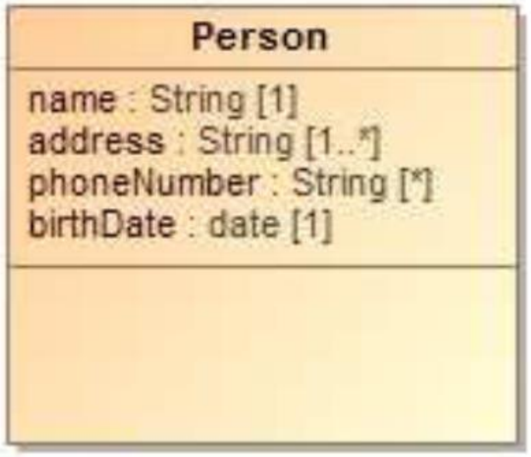
Derived data

- A derived element is **a function of one or more elements, which in turn can be derived**.
- The notation for a derived element is a slash in front of the element name.
- **The constraint that determines the derivation must be shown.**

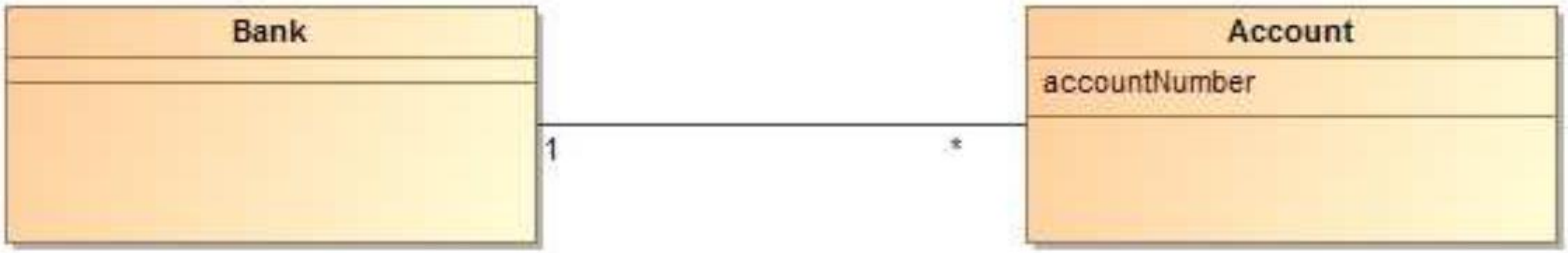


Multiplicity for attributes

You can specify if an attribute is single or multivalued, mandatory or optional



Qualified Associations



What is the meaning of this association?

How can we implement it?

Is this a realistic representation?

Qualified Associations

A **qualified association** is an association with a **qualifier**.

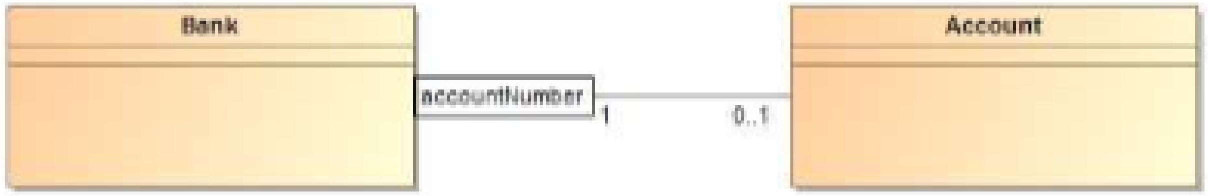
A qualifier may be used in an association; it **distinguishes the set of objects at the far end of the association <u>based on the qualifier value</u>**.

A **qualifier** that is used to select an object (or objects) from a larger set of related objects, based upon the qualifier key.

It is possible to define qualifiers for one-to-many and many-to-many associations

A qualifier selects among the target objects, reducing the multiplicity (usually) from many to one

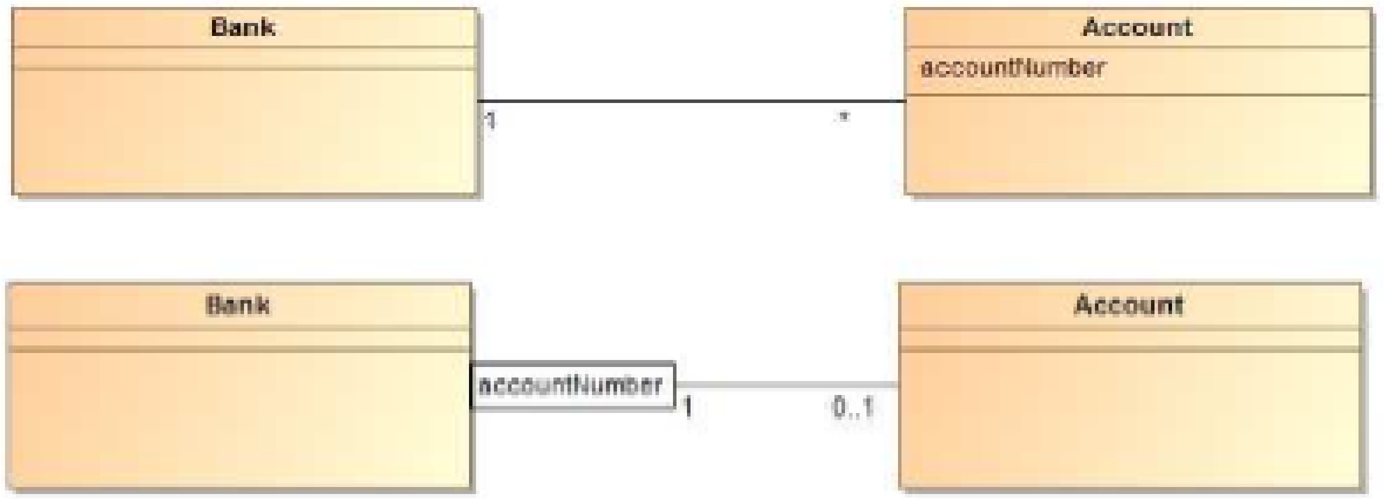
Informally, it suggests looking things up by a key, such as objects in a *HashMap*.



The qualifier says that in connection with a Bank, there may be one Account for each instance of accountNumber
== Given a Bank and a an account number, at most one account could be found
I.e. you cannot have two Accounts within a Bank for the same accountNumber.

https://www.oreilly.com/library/view/applying-uml-and/0130925691/0130925691_ch27lev1sec8.html

Qualified Associations

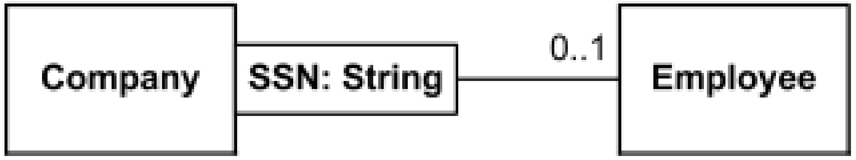


The qualifier says that in connection with a Bank, there may be one Account for each instance of accountNumber == Given a Bank and a an account number, at most one account could be found I.e. you cannot have two Accounts within a Bank for the same accountNumber.

https://www.oreilly.com/library/view/applying-uml-and/0130925691/0130925691_ch27lev1sec8.html

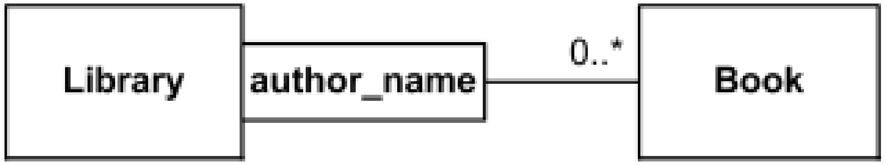
Qualified Associations

In the case in which the target multiplicity is 0..1, the qualifier value is **unique** with respect to the qualified object, and links to **at most one associated object.**

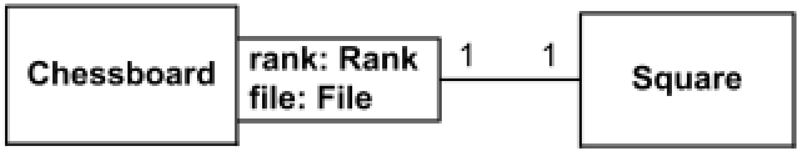


Given a company and a social security number (SSN) at most one employee could be found

In the case of target multiplicity 0..*, the set of associated instances is partitioned into **possibly empty subsets**, each selec



Given a library and author name none to many books could be found



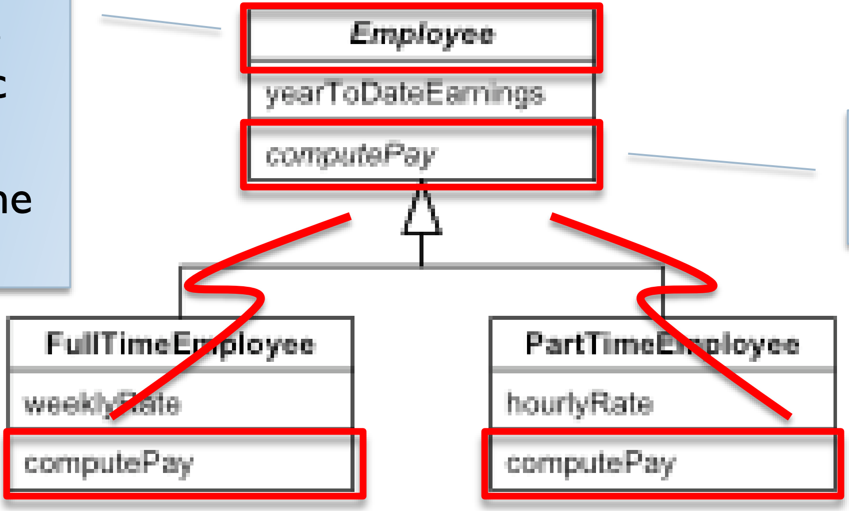
Given chessboard and specific rank and file we'll locate exactly 1 square.
UML specification provides no lucid explanation of what multiplicity 1 means for qualifier.

https://www.informit.com/articles/article.aspx?p=1398623&seqNum=16
, https://www.uml-diagrams.org/property.html#qualifier

Abstract classes

- An **abstract class** is a class that has no direct instances
  - It may define common properties
  - It may define some operation signatures called **abstract operations**

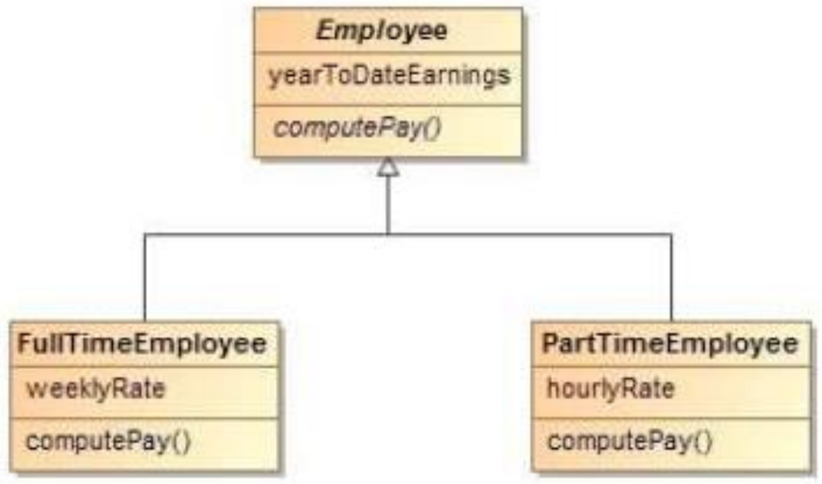
An abstract class is specified with a name in an italic font (or using {abstract} near the class name)



Abstract operation (use an italic font)

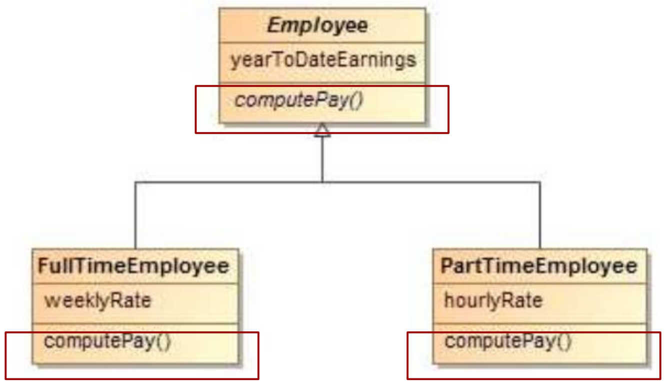
Abstract class

- An abstract class is **a class that has no direct instances but whose descendants classes have direct instances.**
- A concrete class is a class that is instantiable.
- A concrete class may have abstract subclasses, but they in turn must have concrete descendants: only concrete classes can be leaf classes in an inheritance tree.
- In the UML notation an abstract class name is listed in an italic font (or using {abstract} near the class name)



Abstract operations

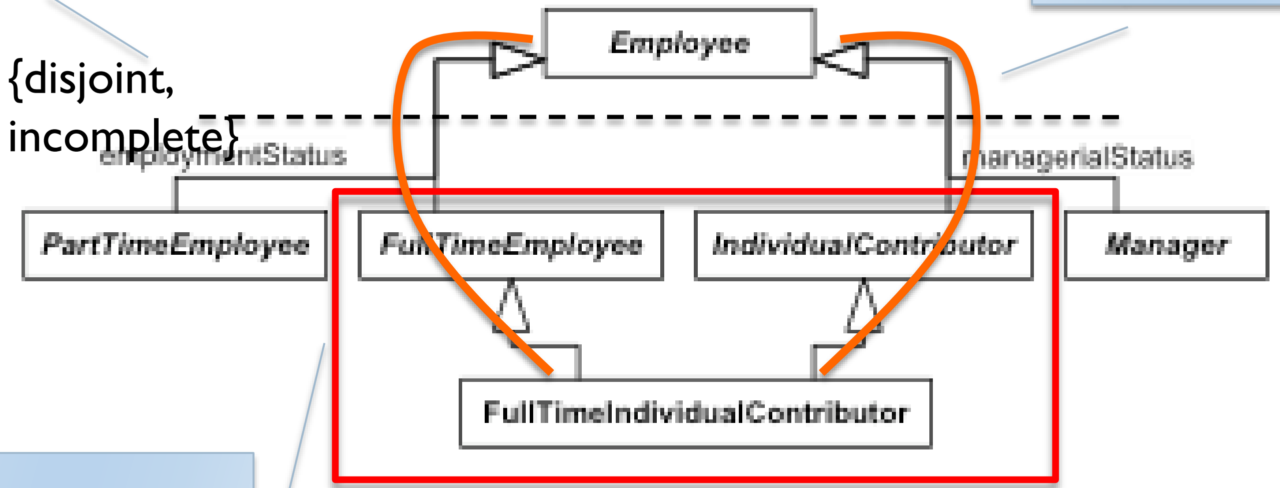
- Abstract classes can be used to define methods that can be inherited by subclasses.
- Abstract classes can define the signature of an operation without supplying a corresponding method.
  - **Abstract operations:**
    - An abstract operation defines **the signature of an operation <u>for which each concrete subclass must provide its own implementation</u>**.
    - An abstract operation is designated by italics or the keyword {abstract}.



Multiple inheritance

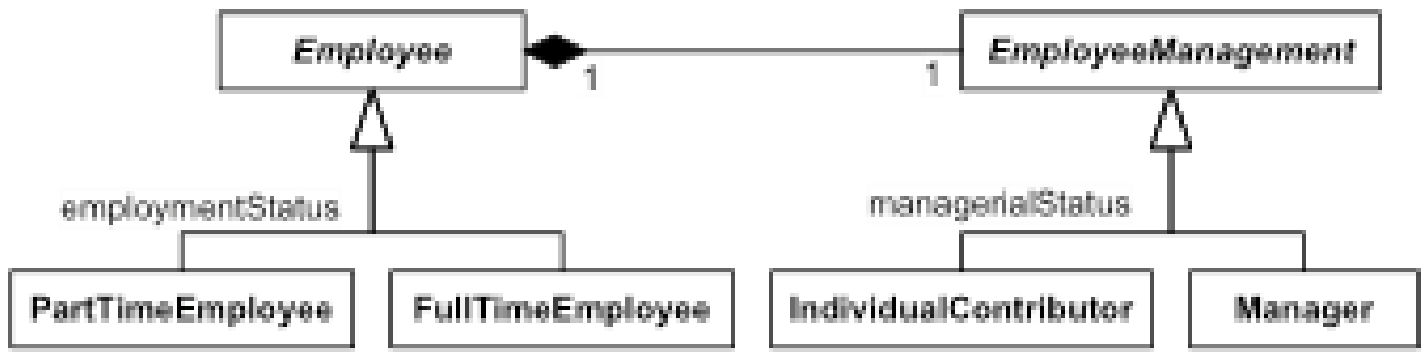
The "diamond problem" will not be present if the classes are disjoint

Be careful with the "diamond problem": some overlapping may cause conflicts



A class can specialize more than one superclasses

Delegation as an alternative to multiple inheritance



Note: Consider using interfaces…

Example

A shape is characterized by horizontal position, vertical position, fill type, fill color, line type and line color. A Rectangle is a shape with a length and a width. A triangle is a shape with a base and a height. Both have an operation to compute the area of the shape.

Do we have abstract classes OR abstract operations here?

Association classes

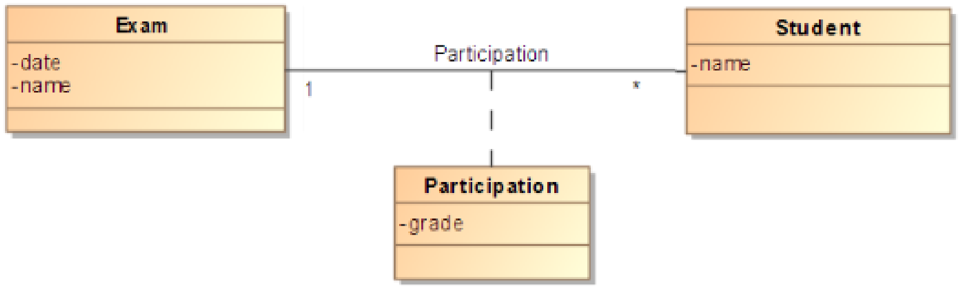
An **association class** is a **class that is part of an association relationship between two other classes OR an association class is an association that is also a class**

An association class can be attached to an association relationship to provide **additional information about the relationship.**

Like a class, an association class is **can** **contain operations, attributes, and other associations.**

Alternative representations in case of multiplicity one-to-many***

UML notation: class box attached to the association by a dashed / dotted line



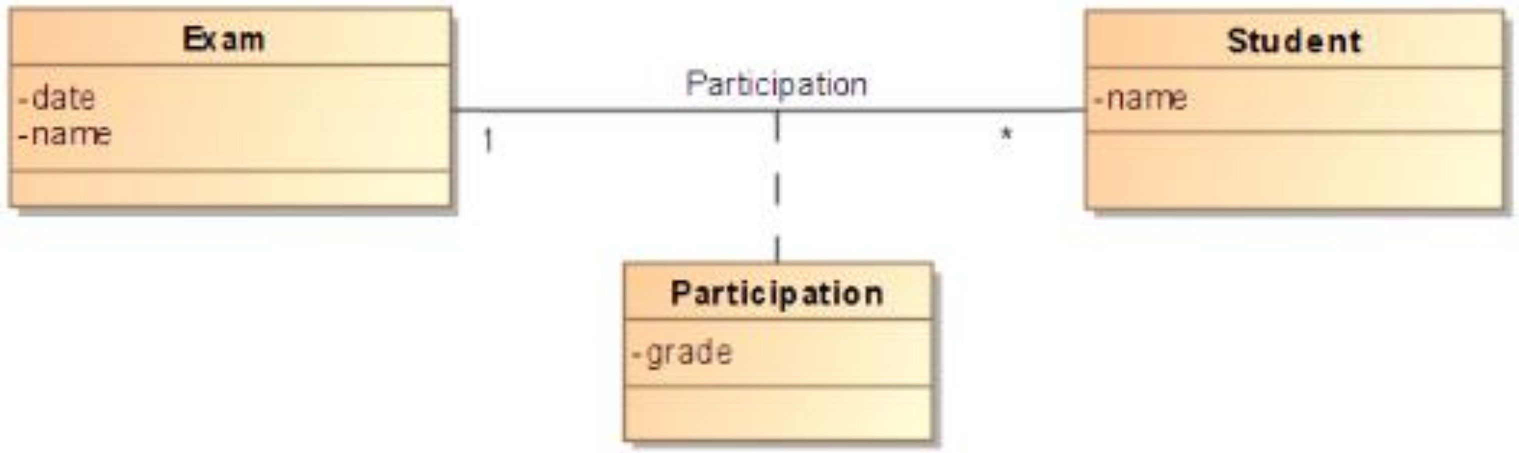
What makes it different from alternative representation?

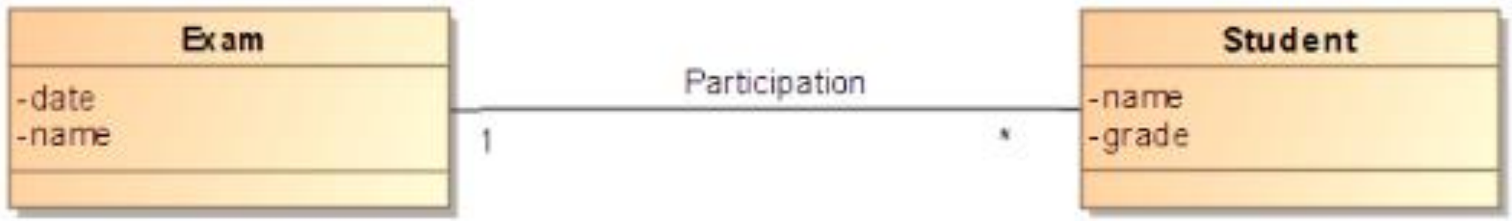
(lifetime control and uniqueness)

Source: https://www.ibm.com/docs/en/rsm/7.5.0?topic=diagrams-association-classes

Association classes ***

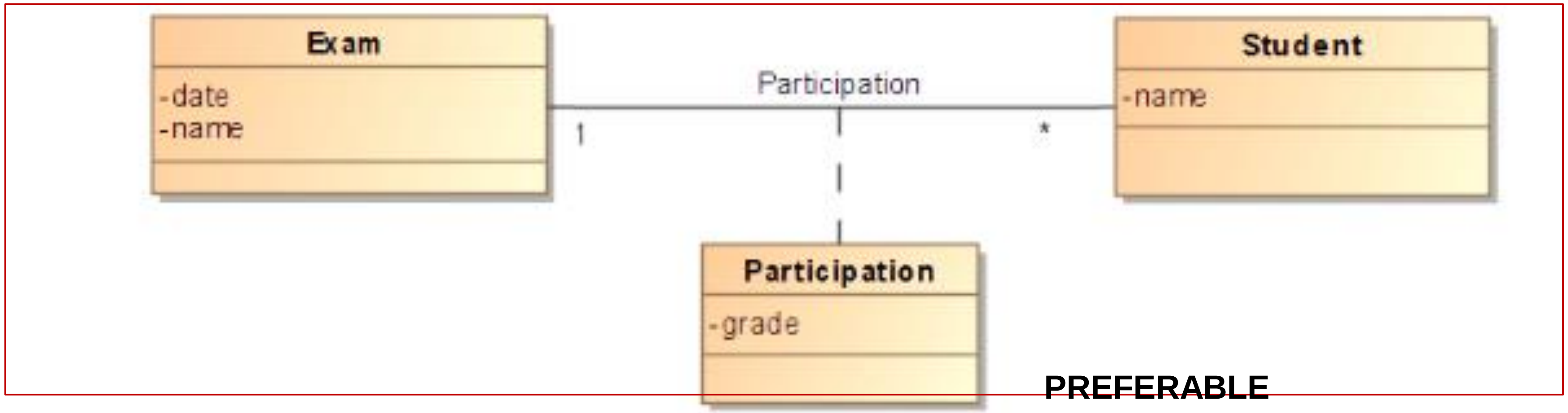
Alternative representations in case of multiplicity one-to-many



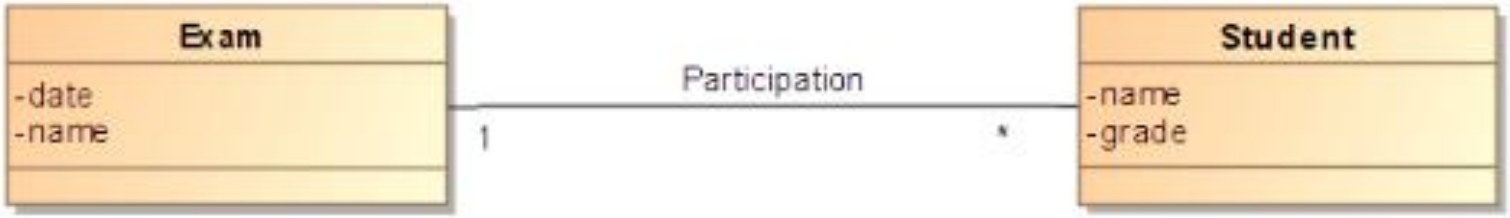


Association classes ***

Alternative representations in case of multiplicity one-to-many



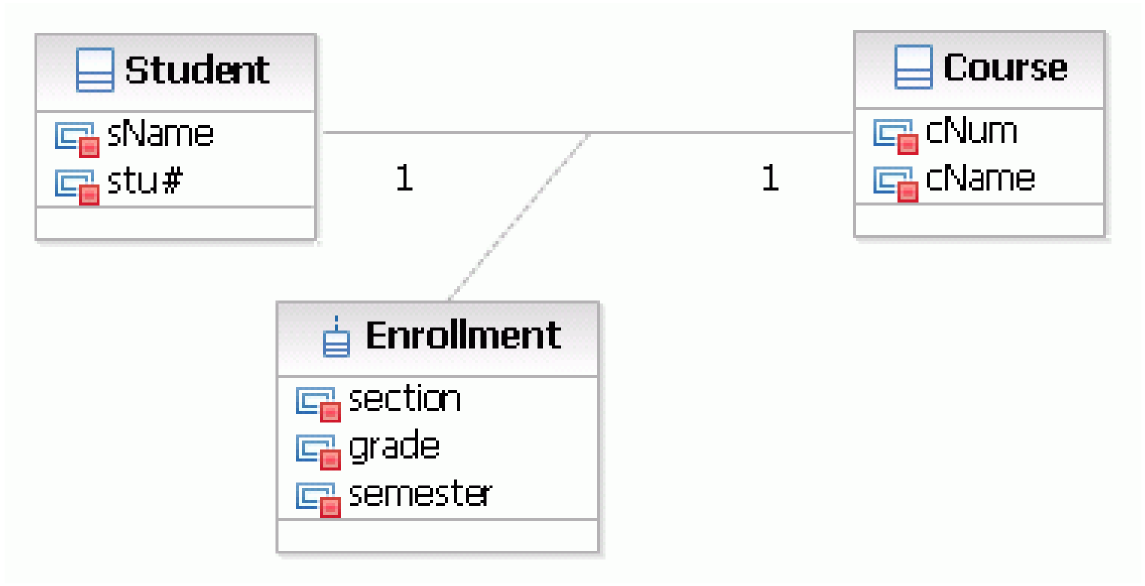
PREFERABLE



Association classes

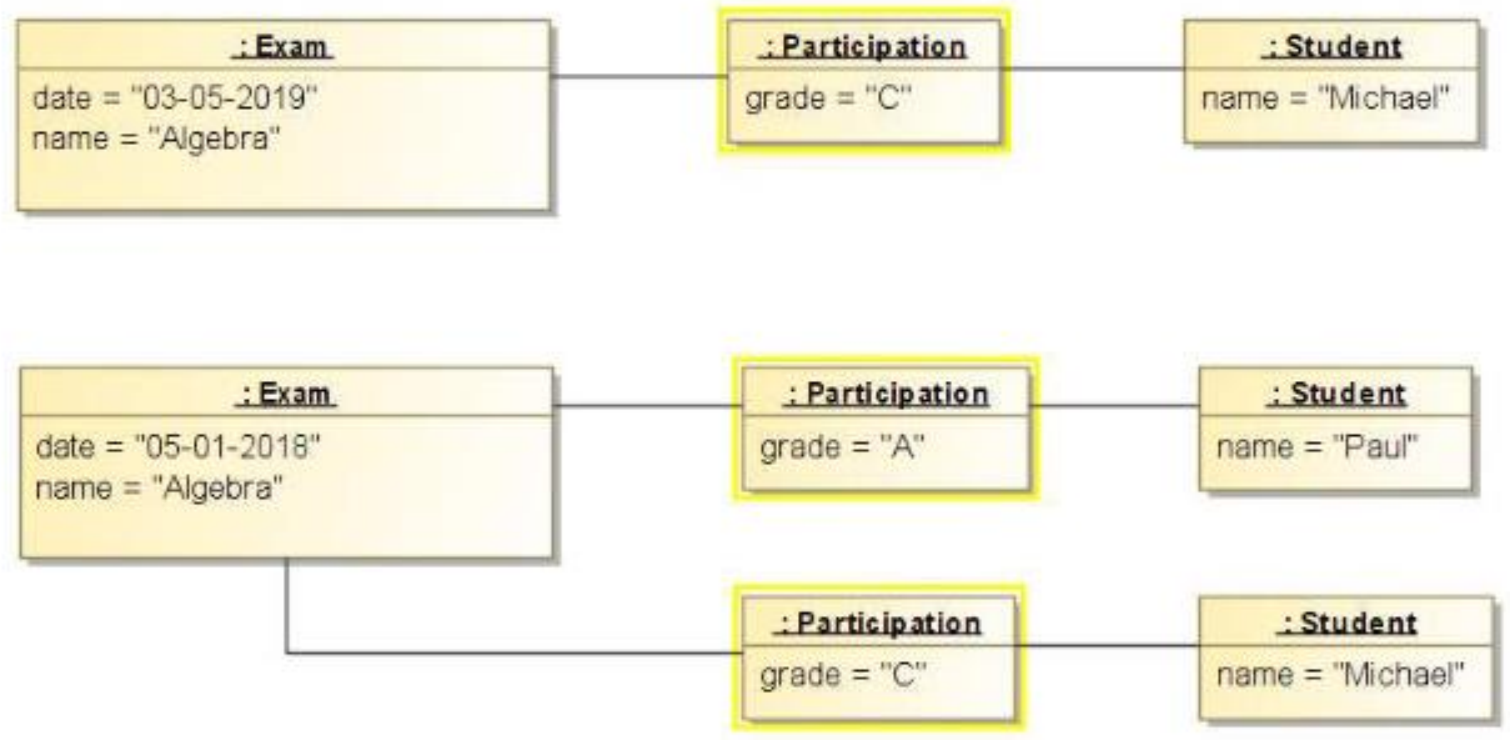
Example:

a class called Student represents a student and has an association with a class called Course, which represents an educational course. The Student class can enroll in a course. An <u>*association class called Enrollment further defines the relationship between the Student and Course classes*</u> *by providing section, grade, and semester information related to the association relationship*



Source: https://www.ibm.com/docs/en/rsm/7.5.0?topic=diagrams-association-classes

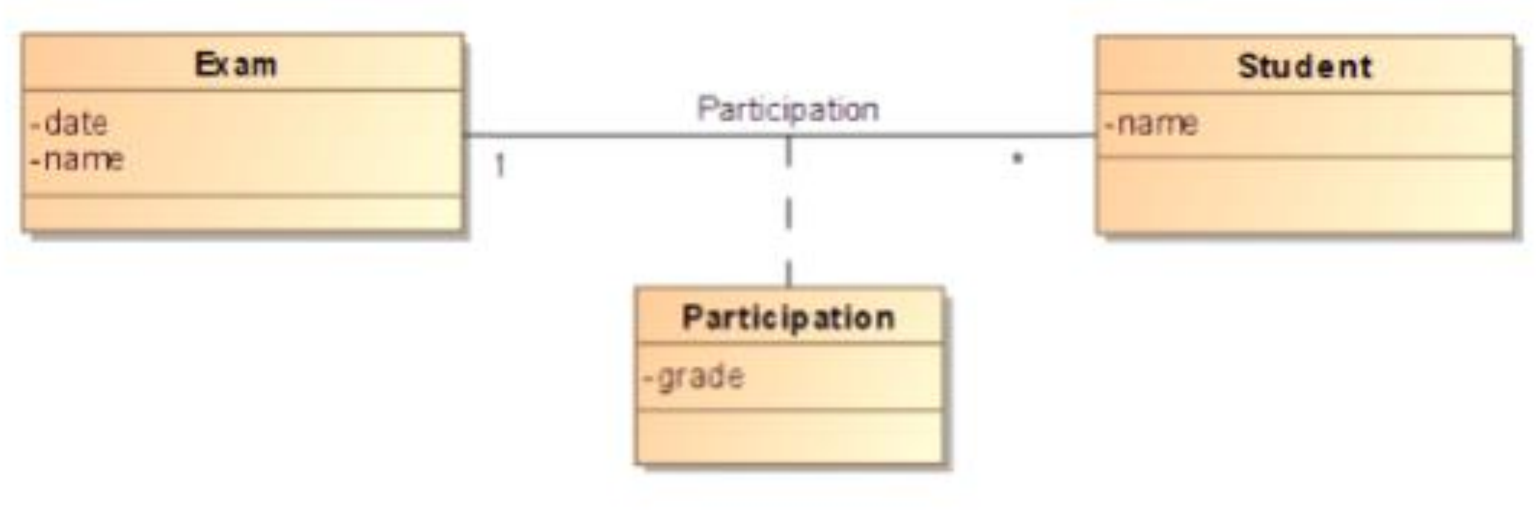
Association classes (instantiations)



An instantiation relationship *(general concept)* is a type of usage dependency between classifiers that indicates that the operations in one classifier create instances of the other classifier.

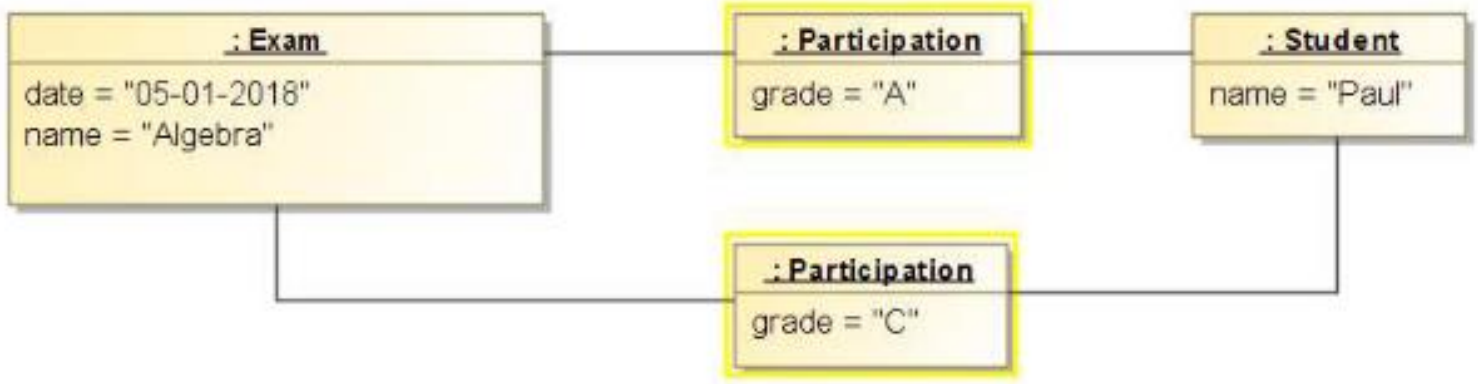
Association classes ***

Since it is not possible to have two links of the same association between the same two objects, there can be **only one instance of the association class between any two participating objects**



Association classes ***

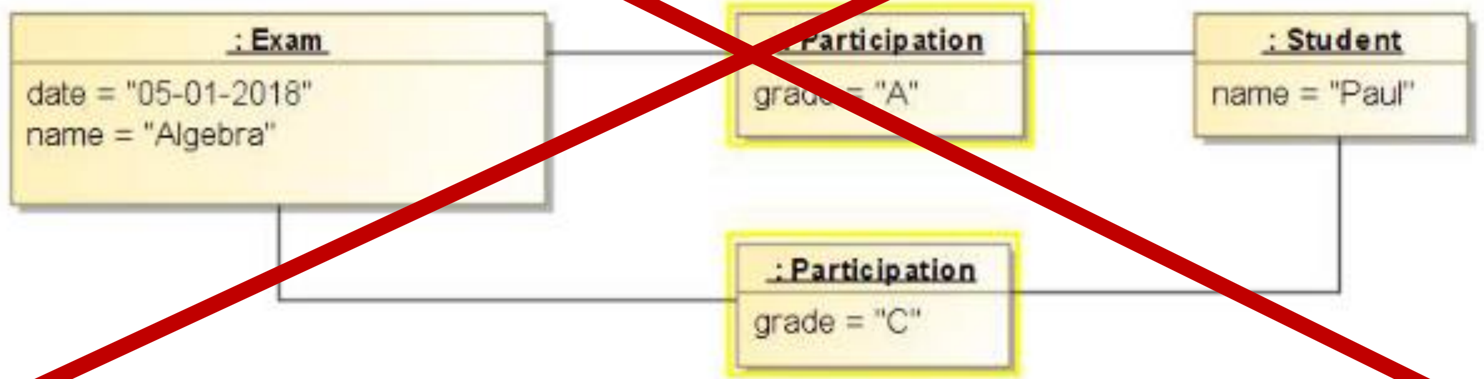
- NOT POSSIBLE (Participation is a link and it is not possible to have two links of the same association between the same two objects)



Since it is not possible to have two links of the same association between the same two objects, there can be only one instance of the association class between any two participating objects

Association classes ***

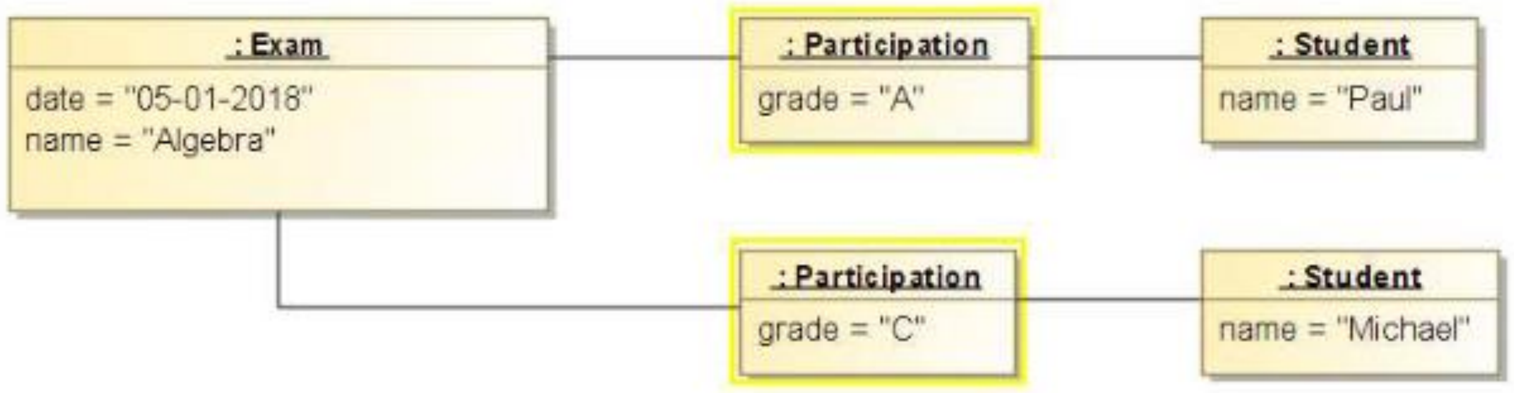
- NOT POSSIBLE (Participation is a link and it is not possible to have two links of the same association between the same two objects)



Since it is not possible to have two links of the same association between the same two objects, there can be only one instance of the association class between any two participating objects

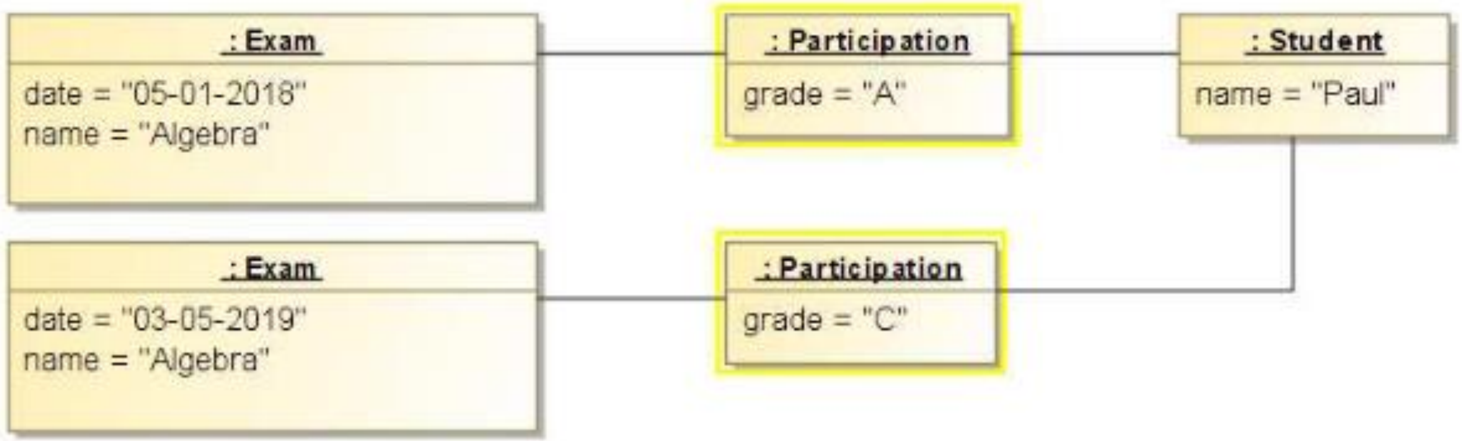
Association classes ***

- POSSIBLE



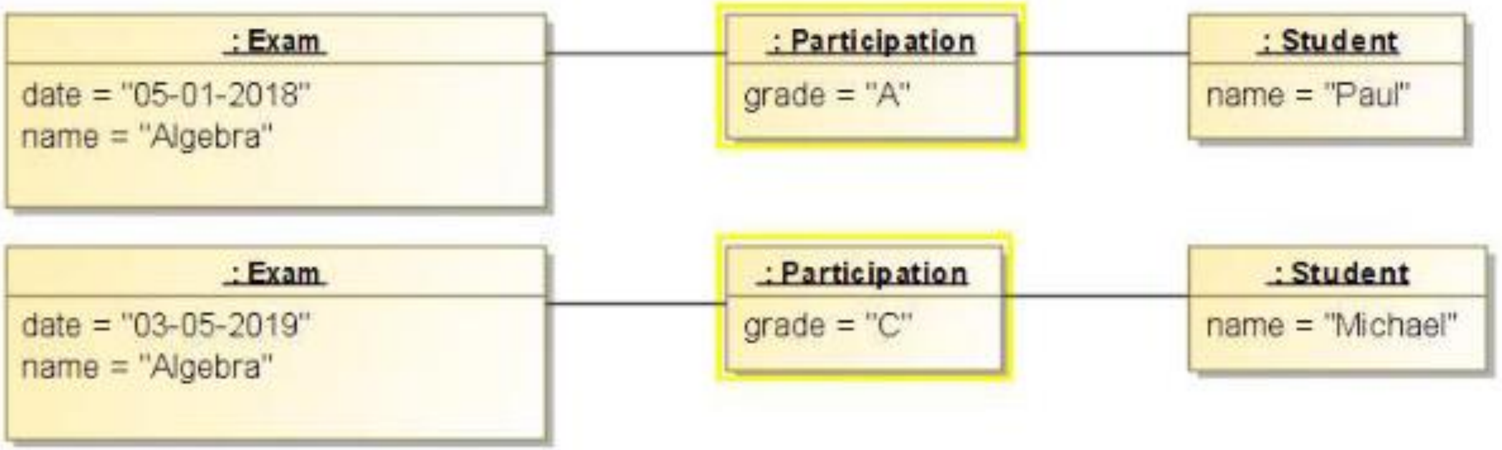
Association classes ***

- POSSIBLE



Association classes ***

- POSSIBLE



Class or Association Class?

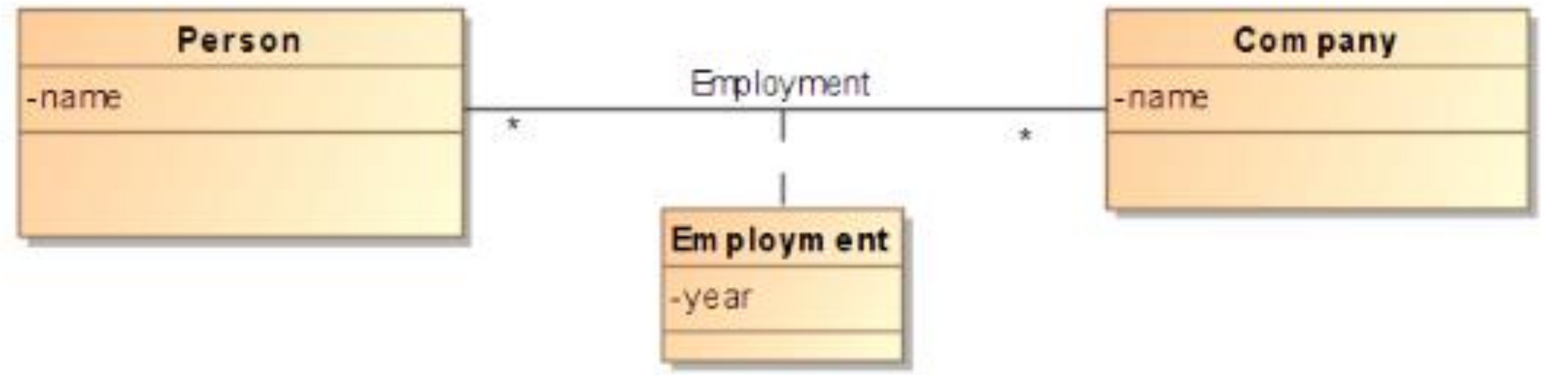
Let’s think:

The same person CANNOT be employed by the same company in different years (Employment is a link and it is not possible to have two links of the same association between the same two objects)

Class or Association Class?

Let's think:

The same person CANNOT be employed by the same company in different years (Employment is a link and it is not possible to have two links of the same association between the same two objects)



Class or Association Class?

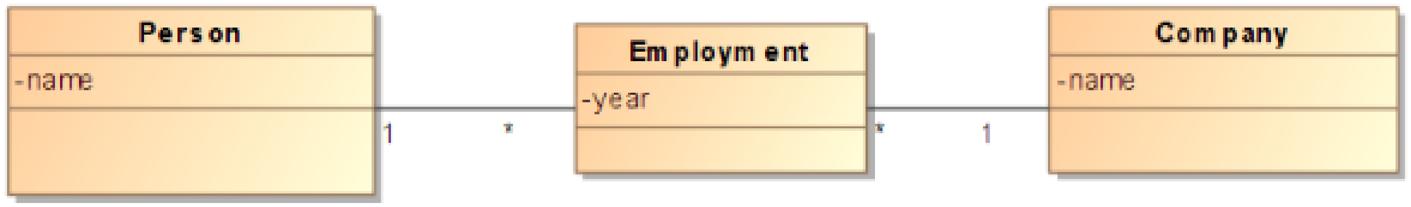
Let's think:

The same person CAN be employed by the same company in different years

Class or Association Class?

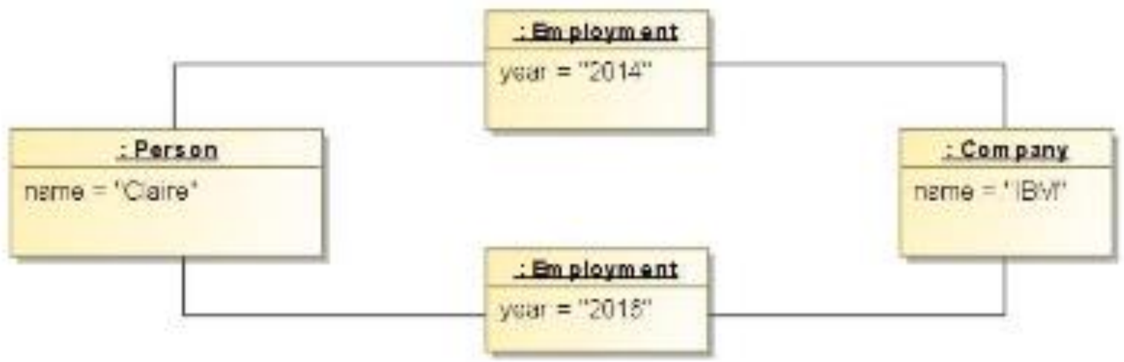
Let's think:

The same person CAN be employed by the same company in different years



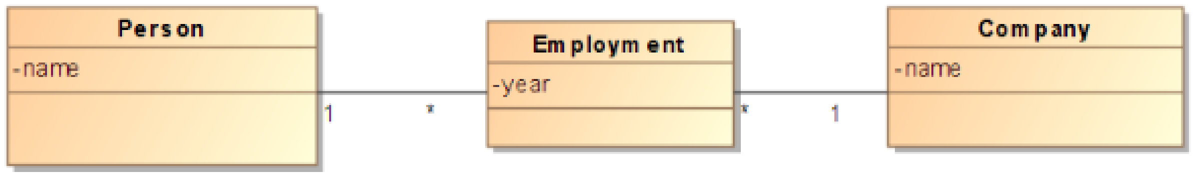
Class or Association Class?

- POSSIBLE (Employment is a class now and not a link)

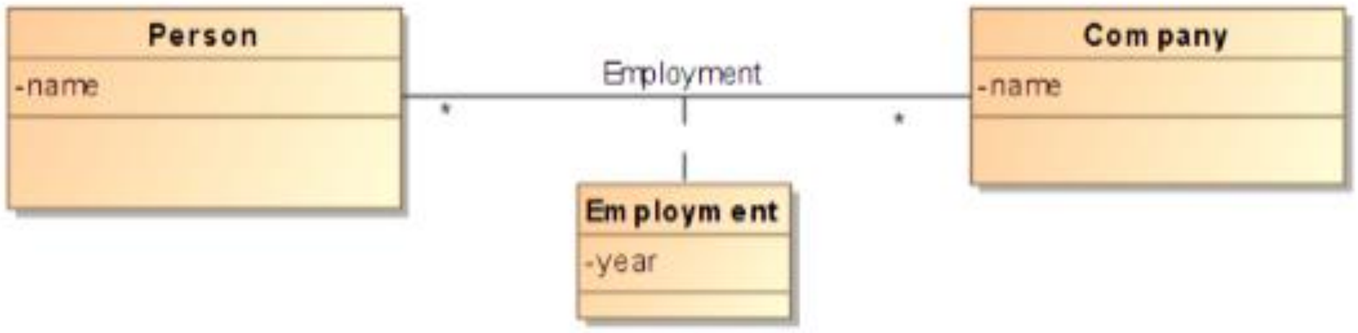


Class or Association Class?

The same person CAN be employed by the same company in different years

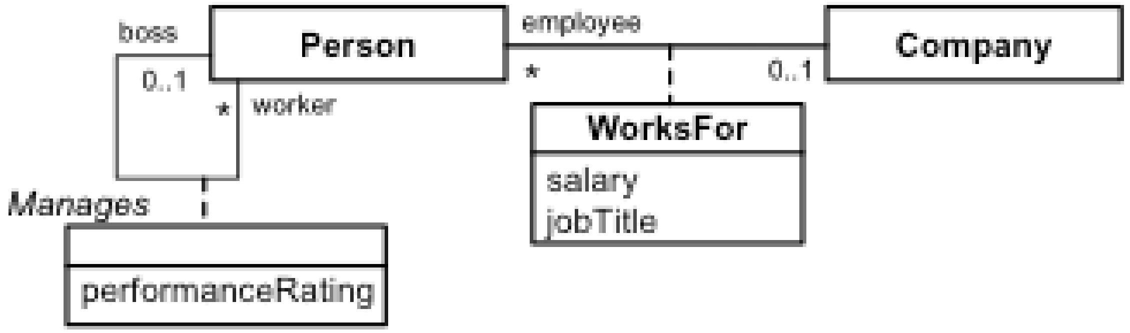


The same person CANNOT be employed by the same company in different years (Employment is a link and it is not possible to have 2 links of the same association between the same 2 objects)

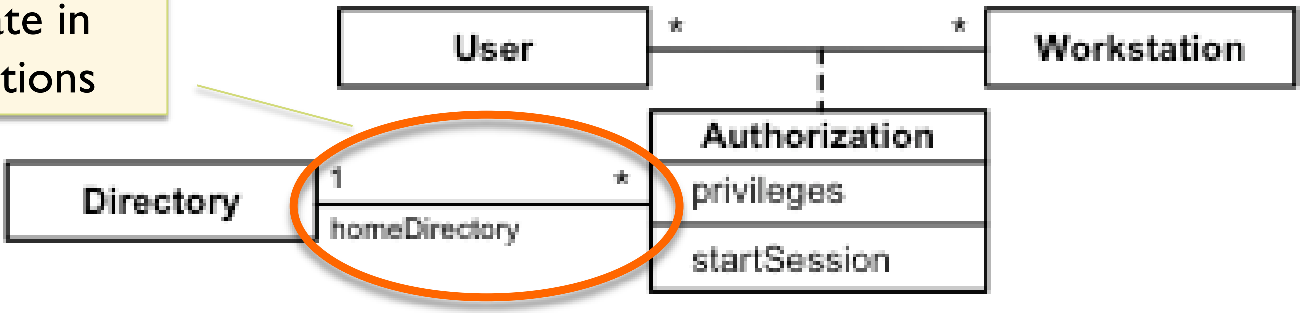


Association classes

Association may also have properties giving rise to association classes



Association classes may participate in other associations



Association classes. Yes / no?

The same two objects can be in an association with each other at most once.

I.e., the relationship instances form a set of pairs of objects; since the nature of a set prevents duplicates, then the same pair of objects is not allowed. This is limiting, as there are situations when two objects could relate to each other two or more times. In this case, more information is needed to differentiate between the many associations of the same two objects.

Since the relationship instances no longer form a set we cannot use binary associations in the model. Many-to-many association is not able to model these relationships.

***Example**: In a library, customers can borrow many books and each book can be borrowed by many customers.*

https://web.csulb.edu/colleges/coe/cecs/dbdesign/dbdesign.php?page=loan.php

Can we use association classes for this?

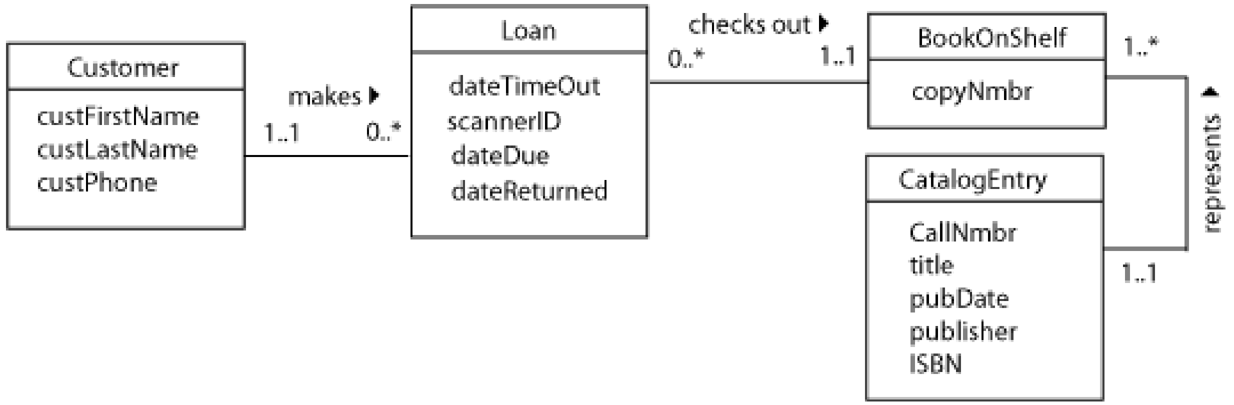
Association classes. Yes / no?

The same two objects can be in an association with each other at most once

***Example**: In a library, customers can borrow many books and each book can be borrowed by many customers.*

At first glance - a simple many-to-many association between customers and books.

BUT! any one customer may borrow a book, return it, and then borrow the same book again at a later time. The library records each book loan separately and needs to track all loans. This record keeping is crucial in libraries, where they need to demonstrate demand for their services to continue their funding. There is no invoice for each set of borrowed books and therefore no equivalent here of the Order in the order entry example.



And the last example…

The system must be able to keep track of which movie videos have been bought/rented and by whom.

For video bought, the system must record the quantity bought; for videos rented, the system must record which copy of the video has been rented and when it is due back.

The video shop will have a customer membership option for an annual fee, which will entitle the member to discounts on videos sales and rentals

Members should be able to make reservations for movie video rentals either in person at the store, by telephone or via the Web.

A member can reserve at most five movie videos at any one time, but there is no limit on how many movie videos a member or nonmember can rent at any one time.

The video shop allow customers to input reviews (up to 100 words) and a rating (from 1 to 5 stars) of movies they have rented. These reviews can anonymous, i.e. the customer should be able to specify whether or not he wants his name to be made known when other customers browse the reviews.

The video shop maintains the following information about all customers: name, address, phone number, fax number, age, sex, email address.

Members are assigned a membership number by the video shop when they become members and a password, which allows them to access the member's only area of the video shop's website, including accessing and changing their personal information.

An employee must be able to enter the basic information about a movie video (title, leading actor(s), director, producer, genre, synopsis, release year, running time, selling price, and rental price).

And the last example…

Classes & associations:

Customer *Buys* movie video

Customer *Rents* movie video

Movie video *Has* rental copy

Customer *Rents* rental copy

Member *Reserves* rental copy

Customer *Provides* review *IsFor* Movie Video,

i.e. Customer *Provides* Review & Movie Video *Has* Review

Generalization

Member *is a kind of* Customer

Constraints

max-card(rental copy, *Reserves*) = 5,

max-card(rental copy, *Rents*) = *

Attributes:

buys → quantity;

rentalCopy → copyNumber, dateDue

Review → review text, rating (attributes)

Review → anonymous

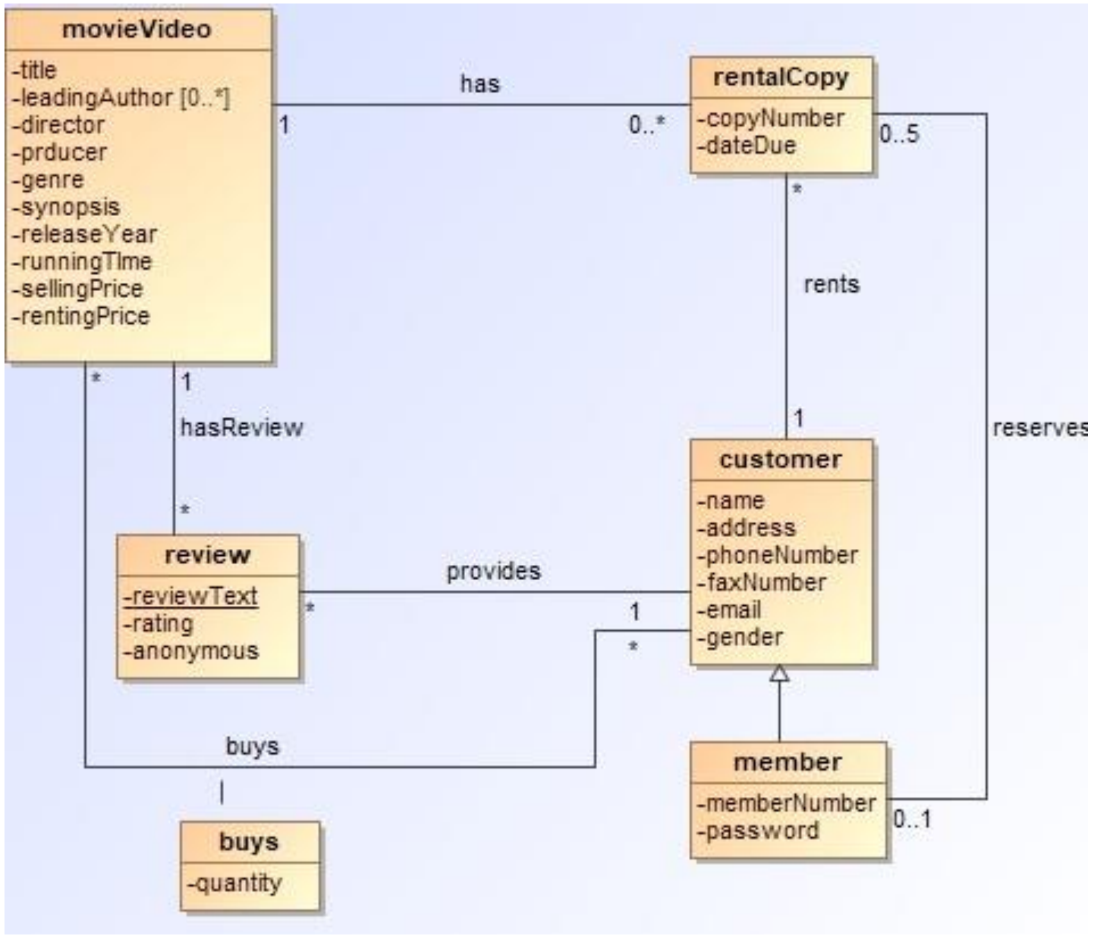
Customer → name, address, phone number, faxnumber, age, genderm email

Member → memberNumber, password

MovieVideo → title, leadingActor[0..*], director, producer, genre, synopsis, releaseYear, runningTime, sellingPrice, rentalPrice

Is it perfect? Is this the only possible version?

And the last example…



Is it perfect? Is this the only possible version?

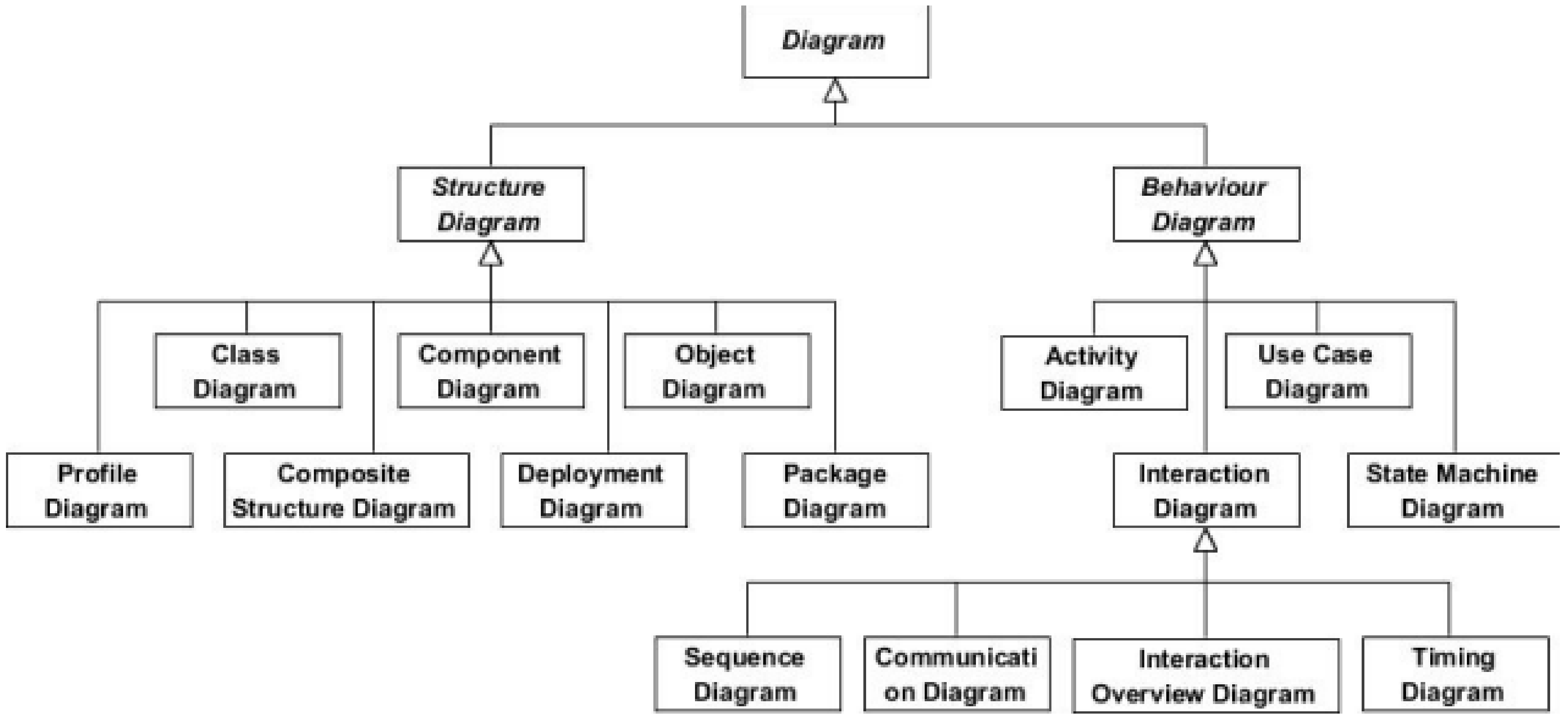
https://pt.slideshare.net/EagleEyes6/domain-model-29104290?related=2

Domain (class) model

? Domain model is a structural model of basic domain concepts and their relationships

? To answer **WHAT ?** question, the domain model provides **(conceptual) classes with attributes and relations among them**

? **Operations are not specified**



Source: https://www.tekportal.net/unified-modeling-language/

Key takeaways

- ? The class model is the graphical representation of the structure of the system and also the relation between the objects and classes in the system.
- ? Each object in the system has data structure and behaviour.
- ? Object sharing the same features are grouped to a class.
- ? Objects are the proper nouns and classes are common nouns identified in the problem statement provided for the development of the application.
- ? The relationship between the objects is the link and the group of links with the same structure is termed as an association.
- ? The class model focuses on the factors that are essential from the applications point of view.

https://binaryterms.com/class-model.html

Cheat sheet ;)

- Class Diagrams (IBM)
- https://www.uml-diagrams.org/
- Fully elaborated ATM example in UML by Russell Bjork

And → Michael Blaha and James Rumbaugh. Object-Oriented Modeling and Design with UML

Some reading…

- **Textbook**

- Michael Blaha and James Rumbaugh. Object-Oriented Modeling and Design with UML (2nd Edition), Prentice Hall, 2004
- Kurt Jensen and Lars M. Kristensen. Coloured Petri Nets. Springer 2009.

- **Links**

- Fully elaborated ATM example in UML by Russell Bjork
- UML 2.2 Stencil for Visio
- Story-driven Modeling by Albert Zündorf.
- Woped
- Workflow course by Wil van der Aalst
- CPN Tools home page