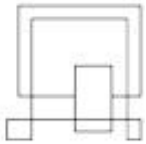
VIA University College
Software Engineering

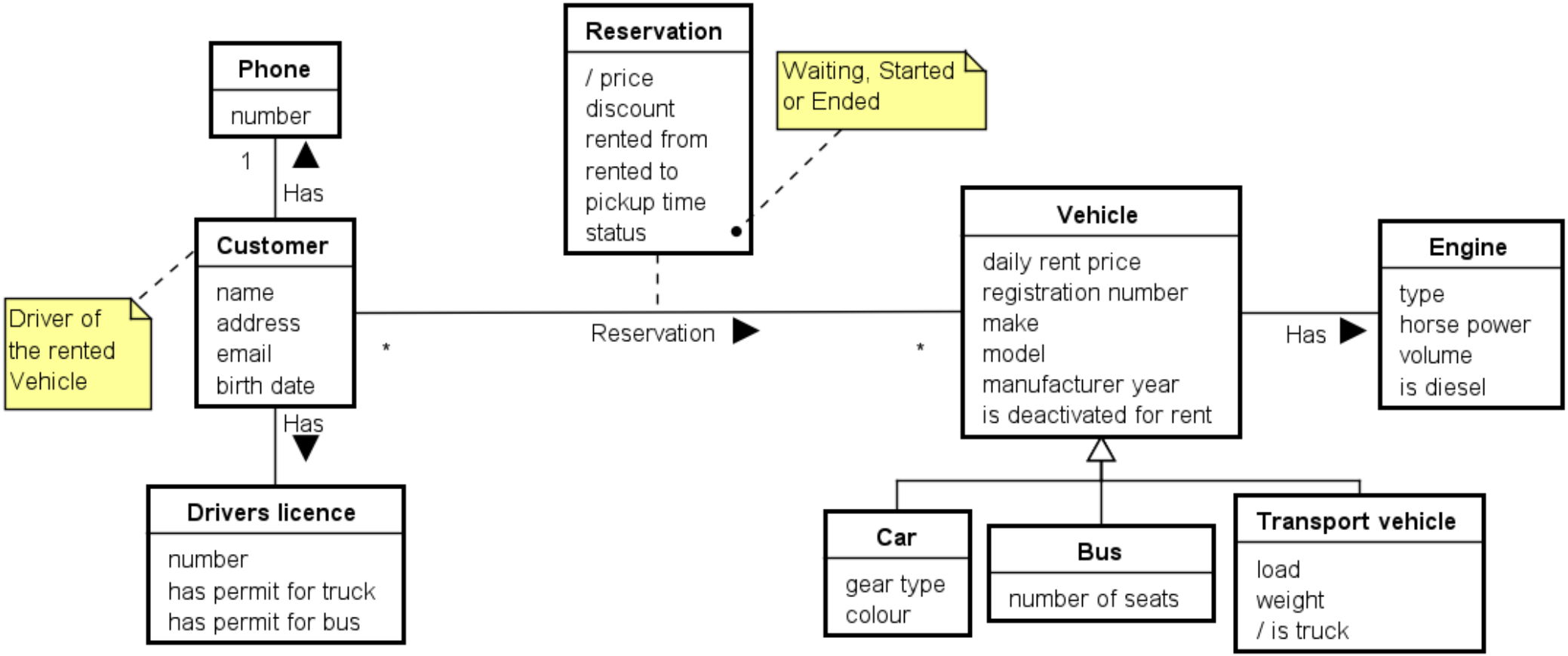


Domain model

[SWE1]

Domain model

- An "end product" of Analysis – a capture of domain entities from all requirements, including what to store



From Requirements to Domain Model

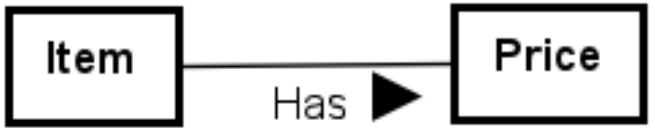
- Definition: Domain model is a conceptual model of the domain that incorporates both behavior and data.
  - NOT software model / classes
- Use words/vocabulary already known in the domain
- Same thing has often different names – ask the domain expert which to use, and be consistent
  - Example: User, Employee, Administrator, Librarian
- Only include things from the problem domain
  - NOT software things (ModelManager, GUI, Database, …)
  - NOT design related classes and structures
- Not too many details – only what is needed to understand the problem (an End product of analysis)
- Discuss your Domain Model with domain experts and your customer

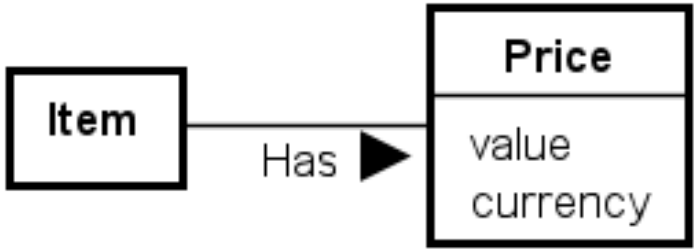
Classes vs. Attributes

Example: An Item has a price

A price attribute?

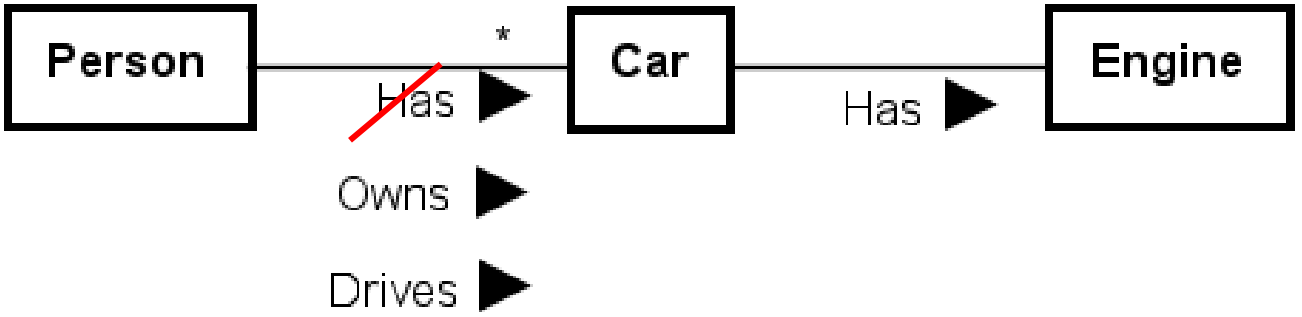
Or two classes?

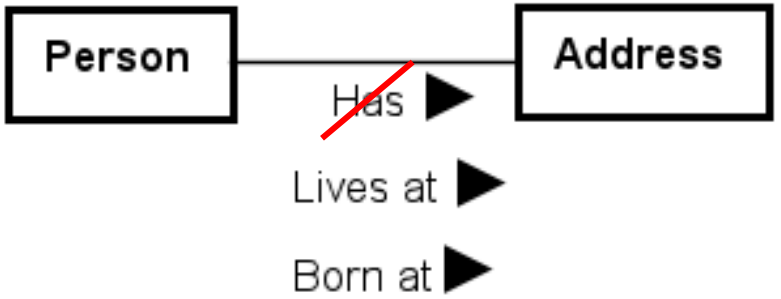




Ambiguous association names

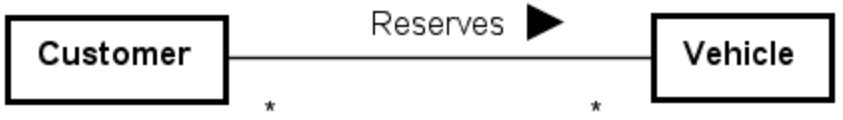
- Be careful not to use names that could be misunderstood





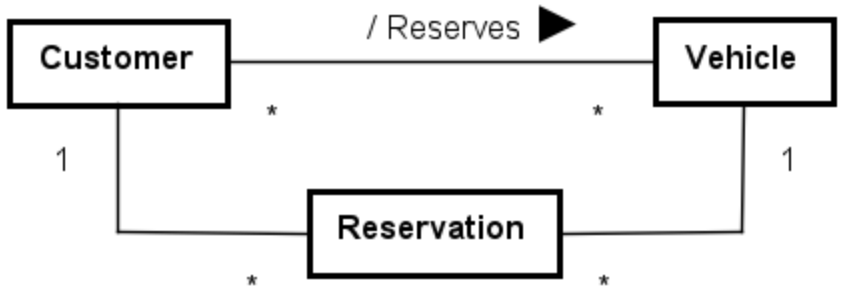
Derived associations and Association classes

- Association



Derived associations and Association classes

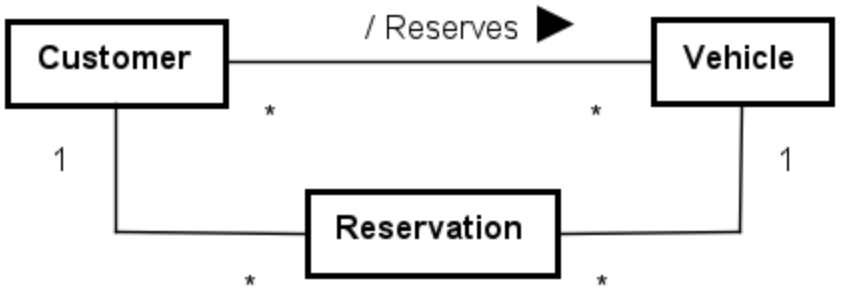
- Derived Association



- A "/" (slash) in front of the association name
- An indirect association

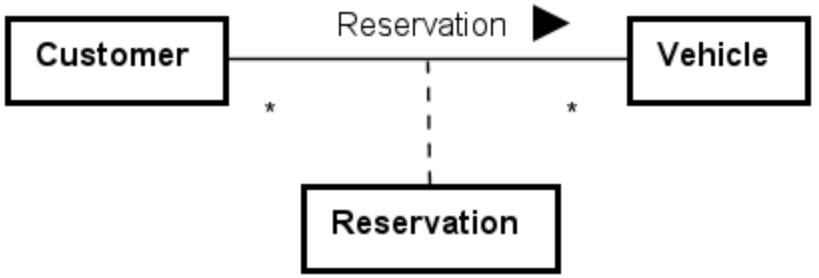
Derived associations and Association classes

- Derived Association



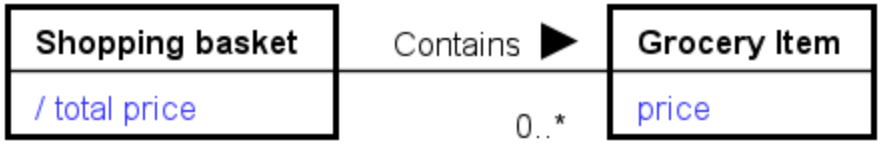
  - A "/" (slash) in front of the association name
  - An indirect association

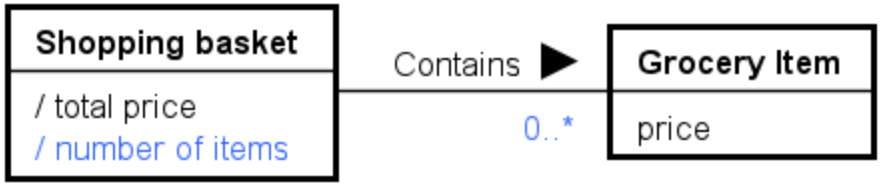
- Association class



Derived attributes

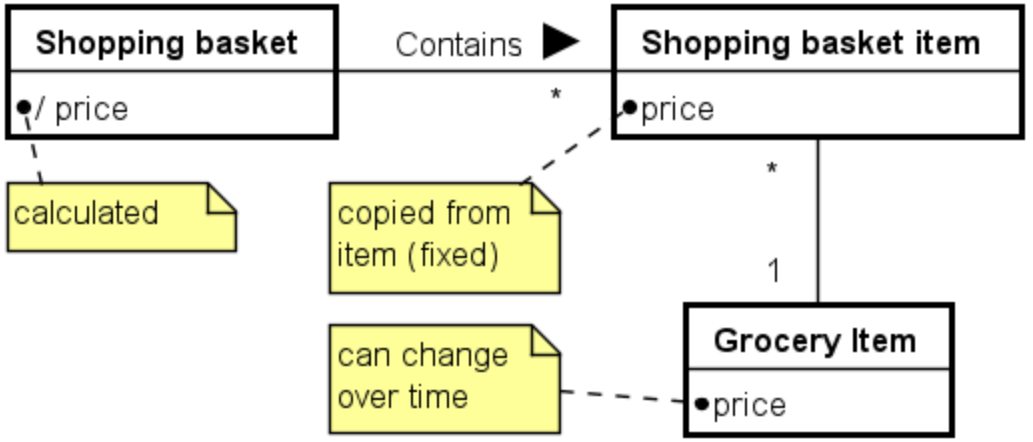
- Shown with a "/" (slash) in front of the variable
- Can be found or calculated (not really an attribute)
- Only shown to clarify or give a better readability



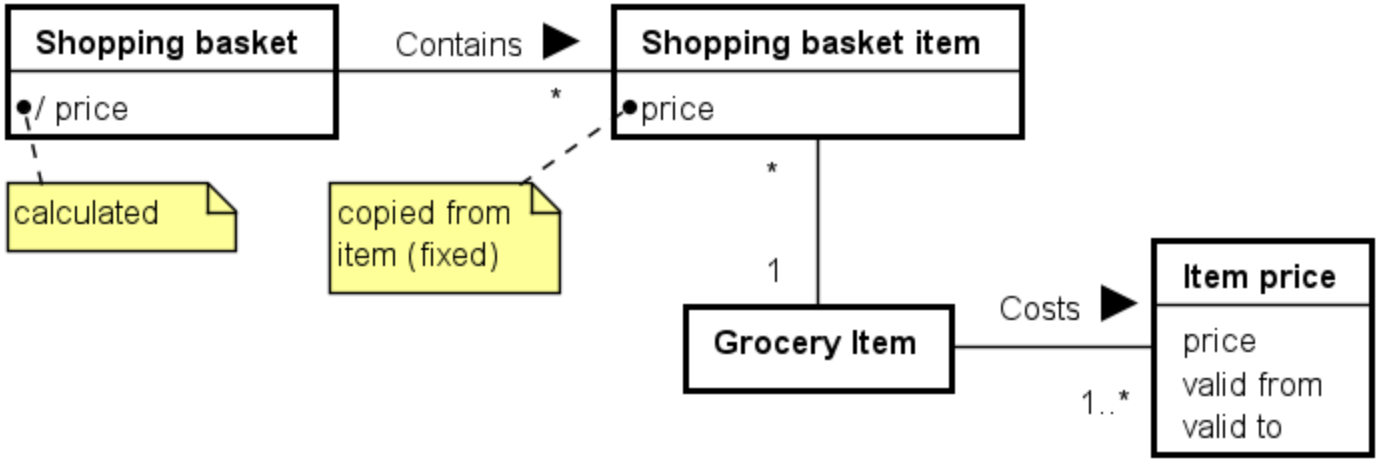


Historical data

- If a price changes after added to a shopping basket

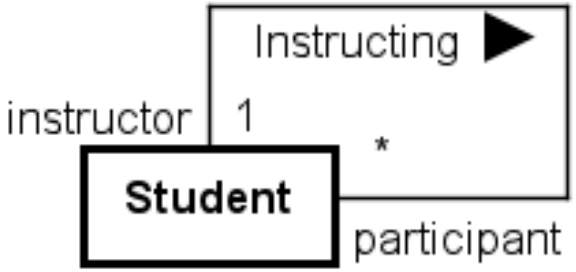


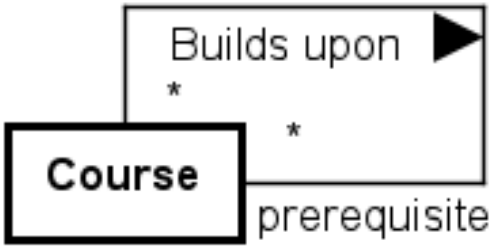
- Historical data



Reflexive associations

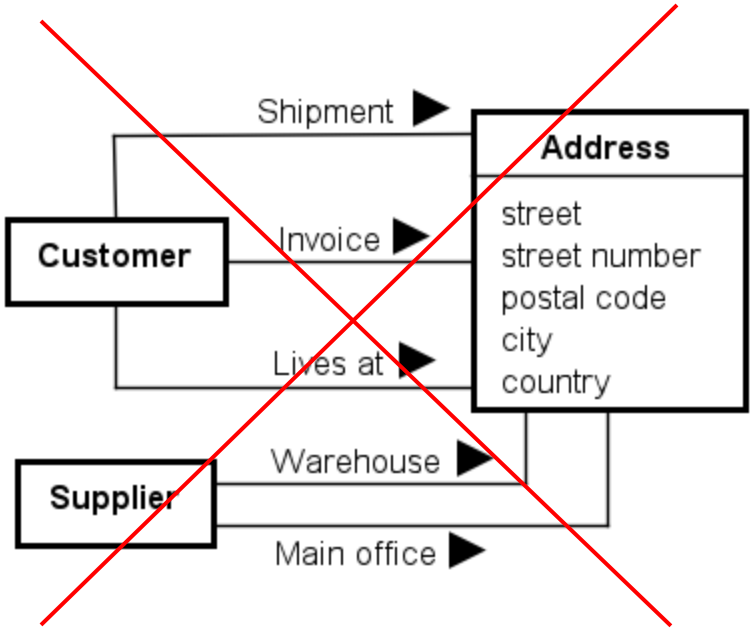
- Association to the class itself
- Self references
- Self-referential class

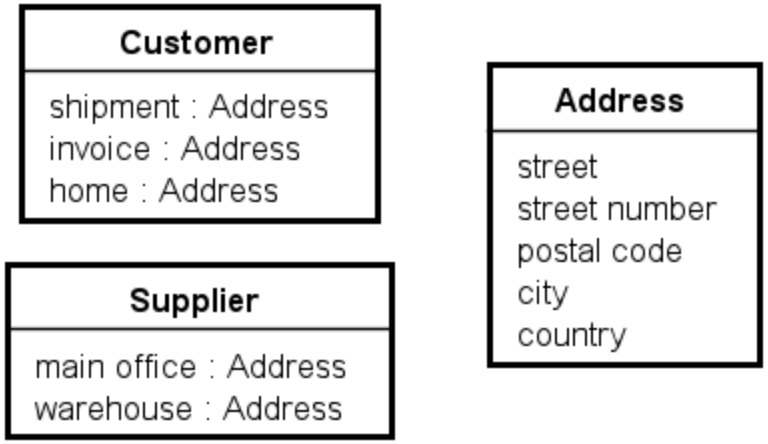




Data type classes

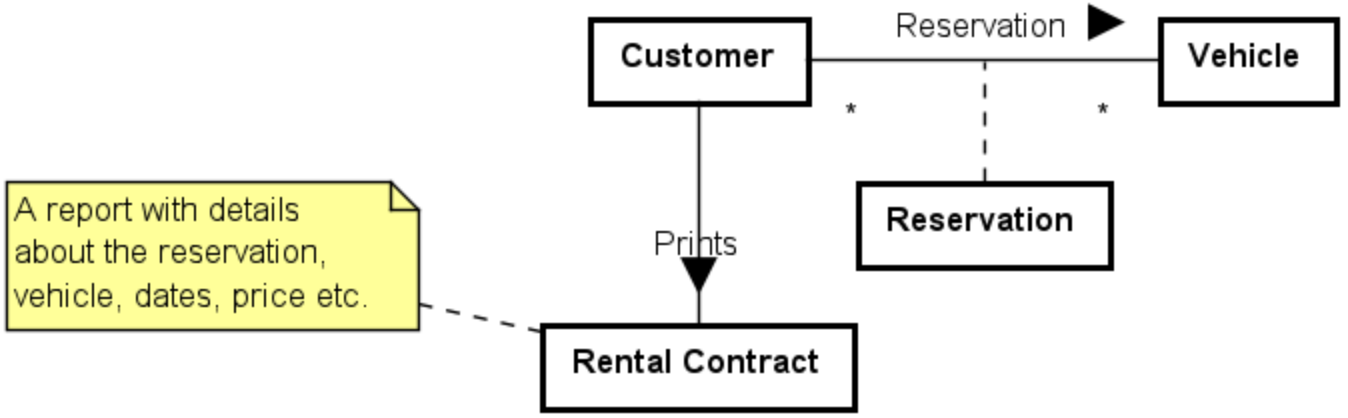
- Readability?





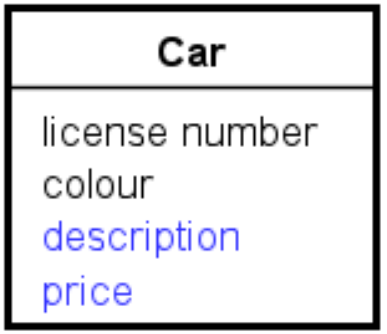
Report classes

- May give an overview
- Are normally not in the implementation, but can be assembled via other relationships

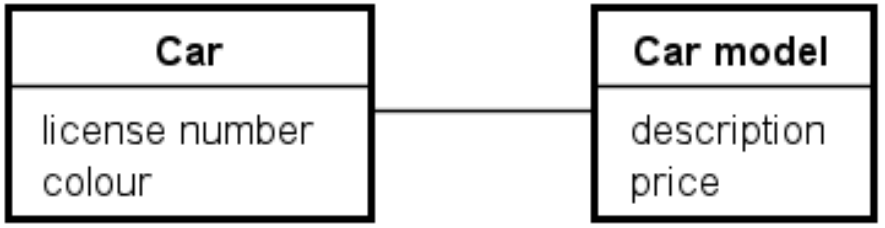


Description classes

- Example: Each car of the same model has the same description and price

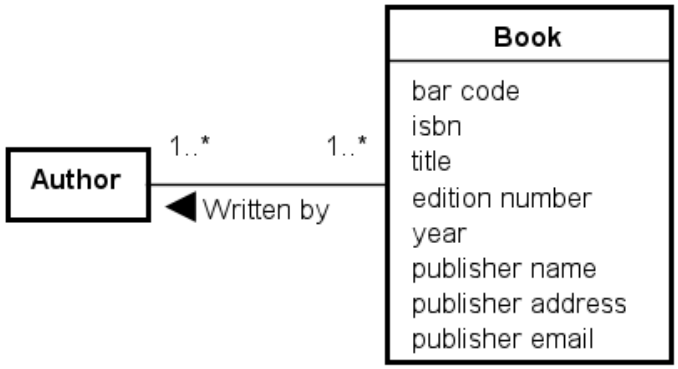


- Description class to avoid duplication

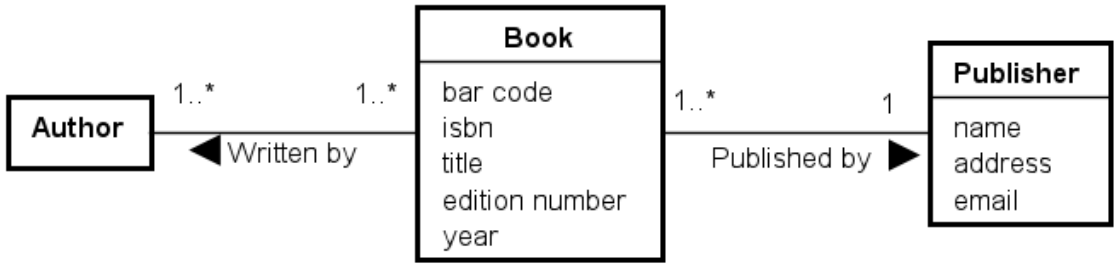


Redundant or duplicated information

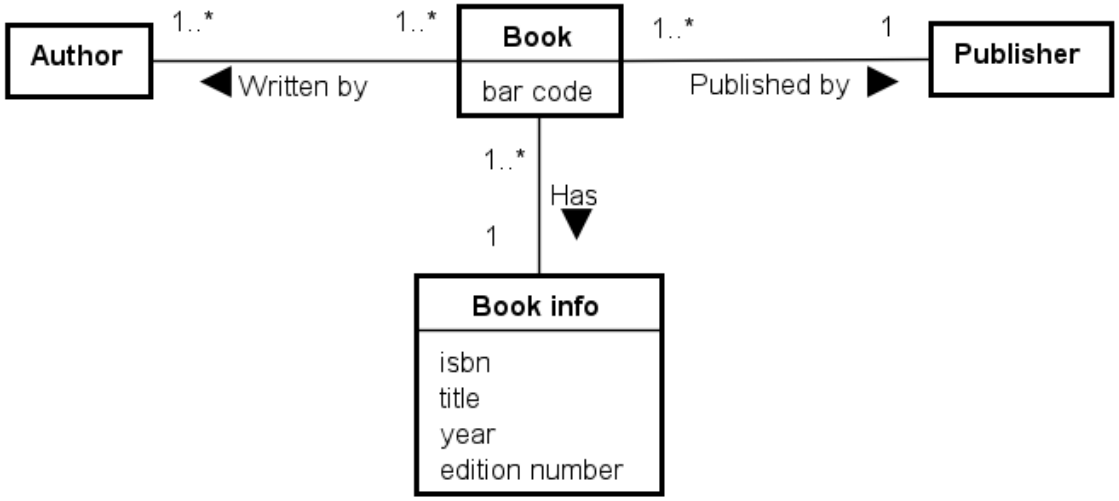
- Same author for several books



- Same publisher for several books

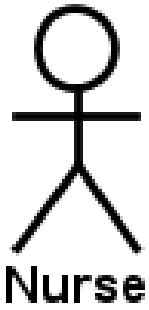


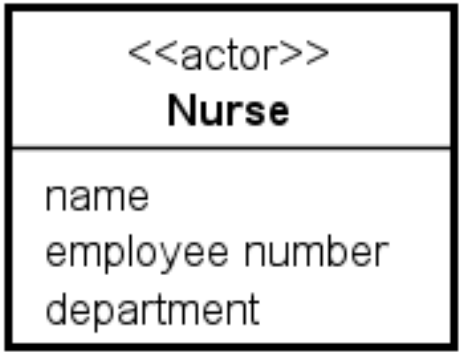
- Same book info for several books (more copies of the same book)



Actor classes

- Actor classes (only) makes sense if you are storing their information in your system





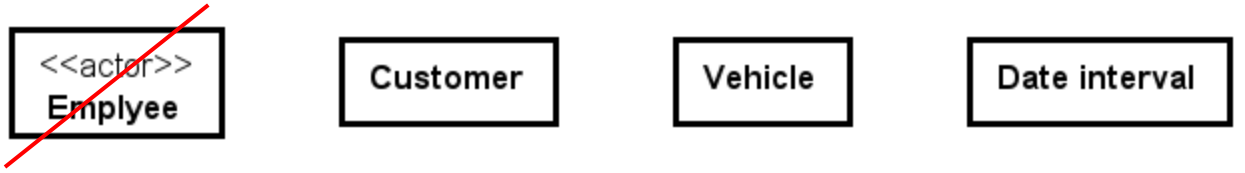
Domain model guidelines

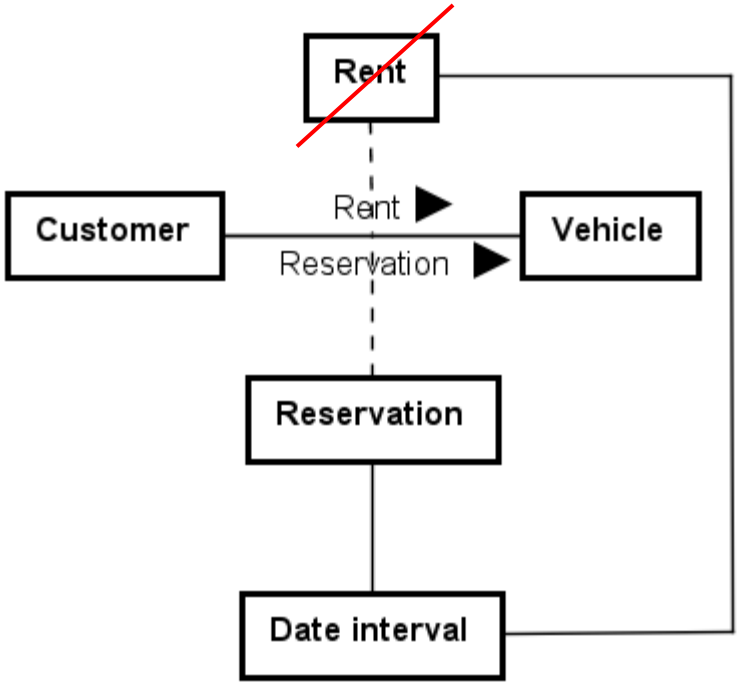
- Class names and attributes may contain spaces
- Multiplicity only shown if it not clear (if not given then it is defined to be 1)
- Navigability not shown
- Association names and directions (if not already clear)
- No types for attributes (unless for data classes)
- No operations (methods)
- May have names for association ends
- No visibility for attributes or association end names

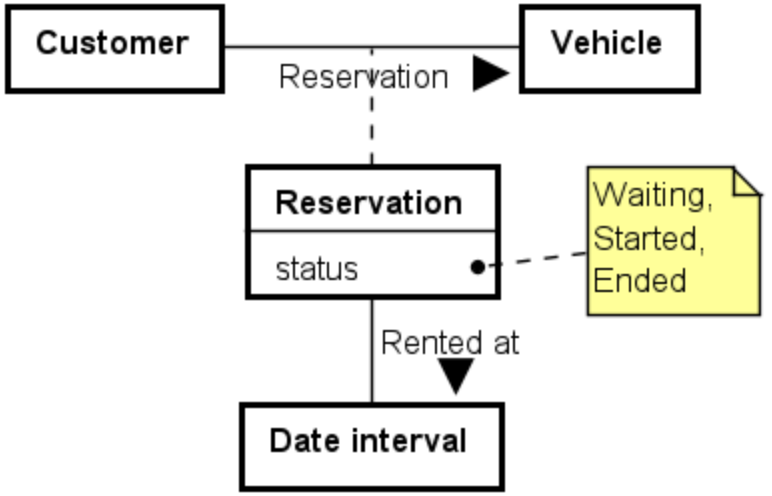
Simplicity is beautiful - only show relevant parts

Requirements and Use cases: Nouns and Verbs

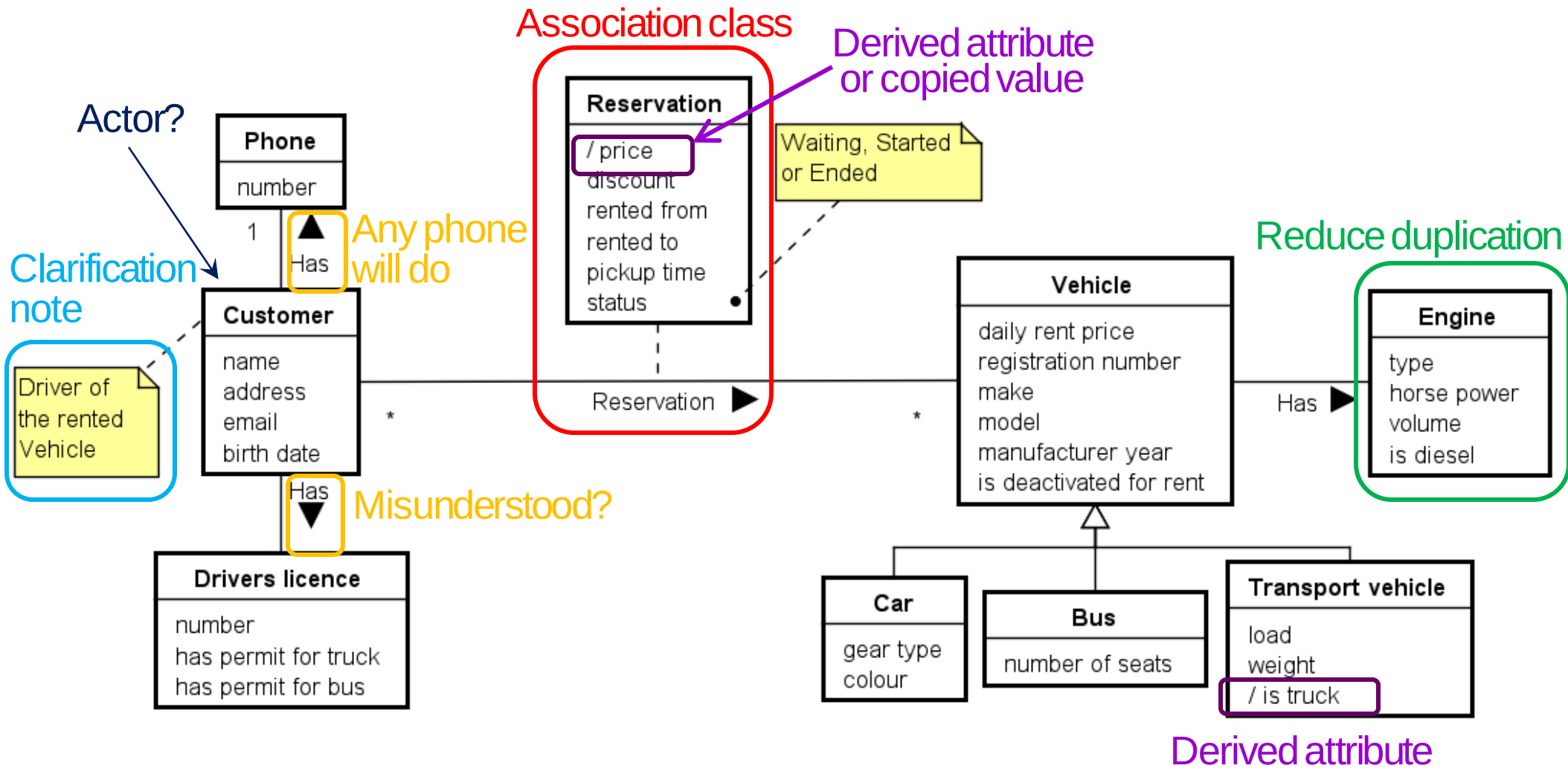
1. As an employee, I want to reserve a vehicle for a customer in a given date interval such that the customer can rent the vehicle.





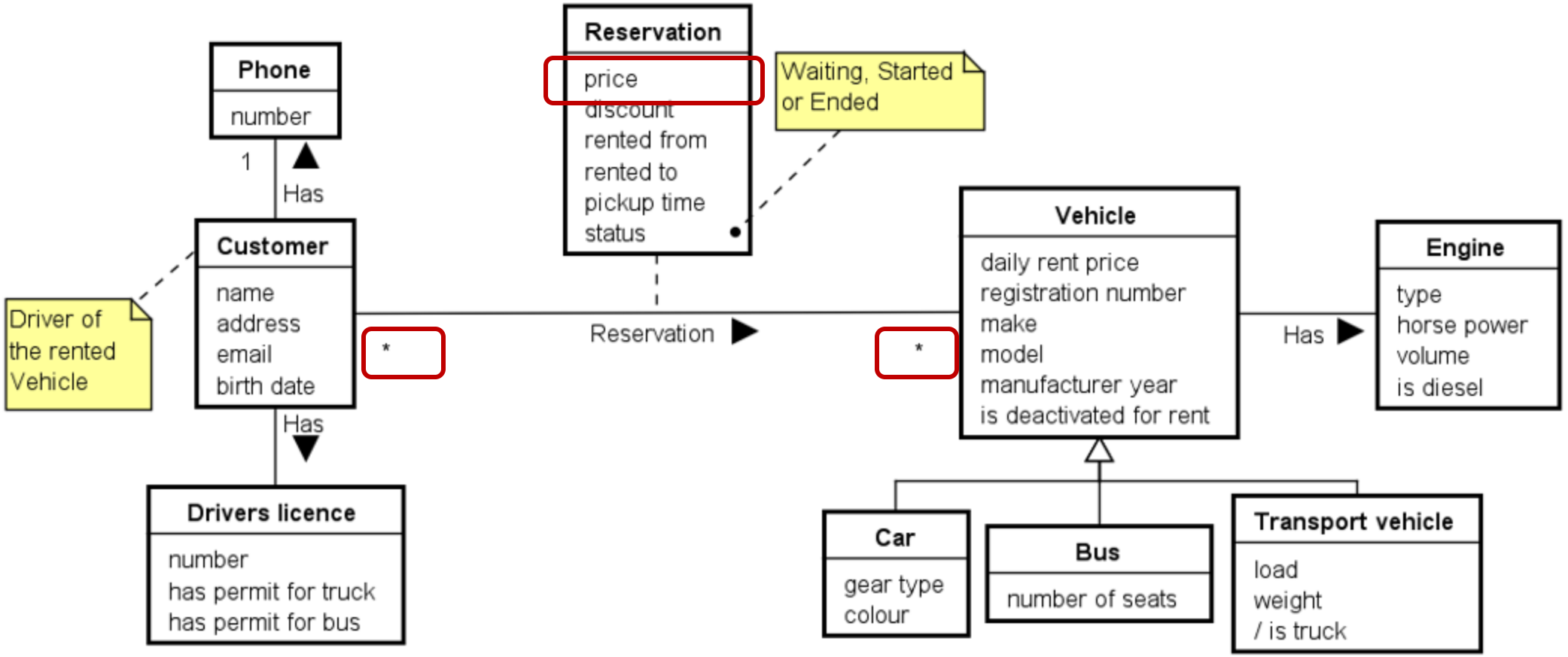


Domain model (Vehicle rental company)



Go through all Requirements (or use cases)

- Can a Customer reserve a Vehicle in a given time interval?
- Can a Customer get a cheaper price for renting a vehicle?
- Can one Customer reserve more than one Vehicle?
- Can one Vehicle be reserved by more than one Customer?



Domain model – in Astah

Diagram → Class diagram

[Left menu] → Initial Visibility
→ [deselect] Operation Compartment
→ [deselect] Attribute and Operation Visibility Kind
→ [deselect] Attributes Type
→ [click] Apply to existing elements

[Use] Name for Associations – starting with Uppercase
[Use] Regular names with spaces (not CamelCase)