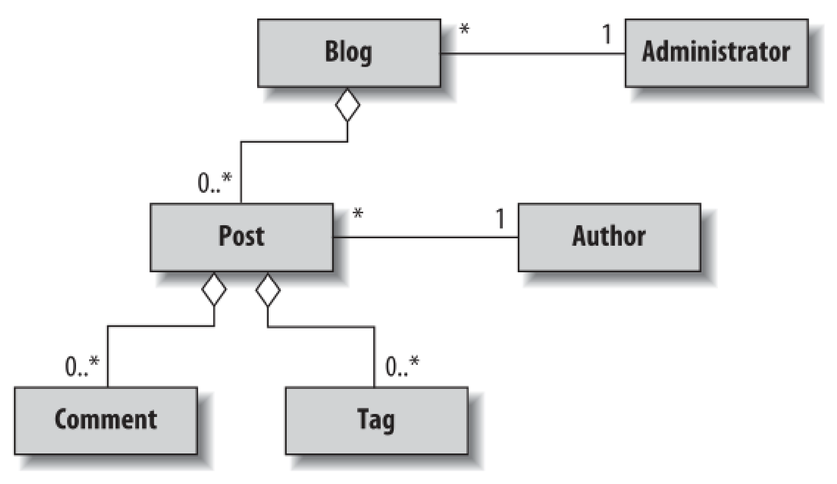
DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

Object-Oriented Modelling

Object-Oriented Fundamentals

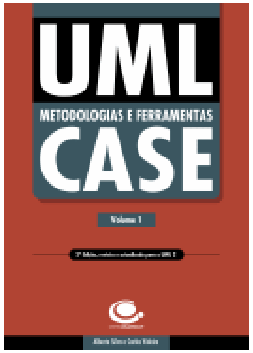
Domain Modeling



Bibliography

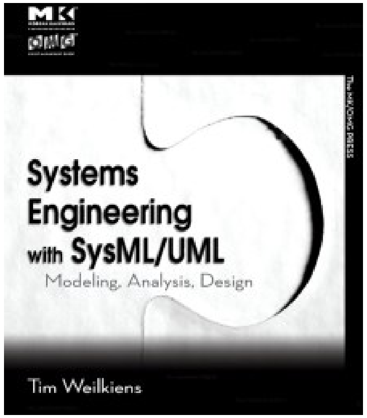
- Silva&Videira

  (UML Use Case Diagrams, Chapter 6)



- Weilkiens

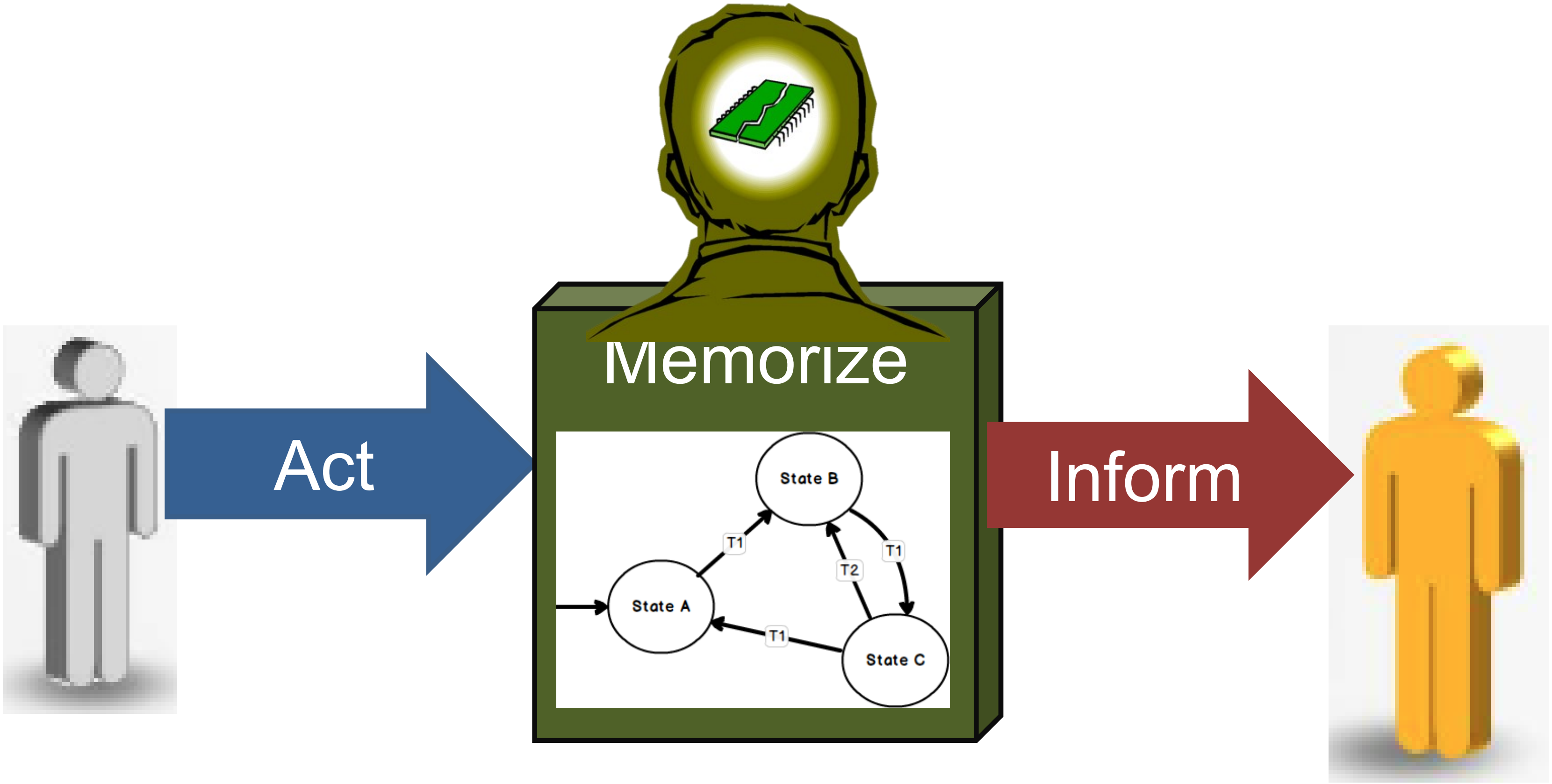
  (Chapters 3.1, 3.2)



- …

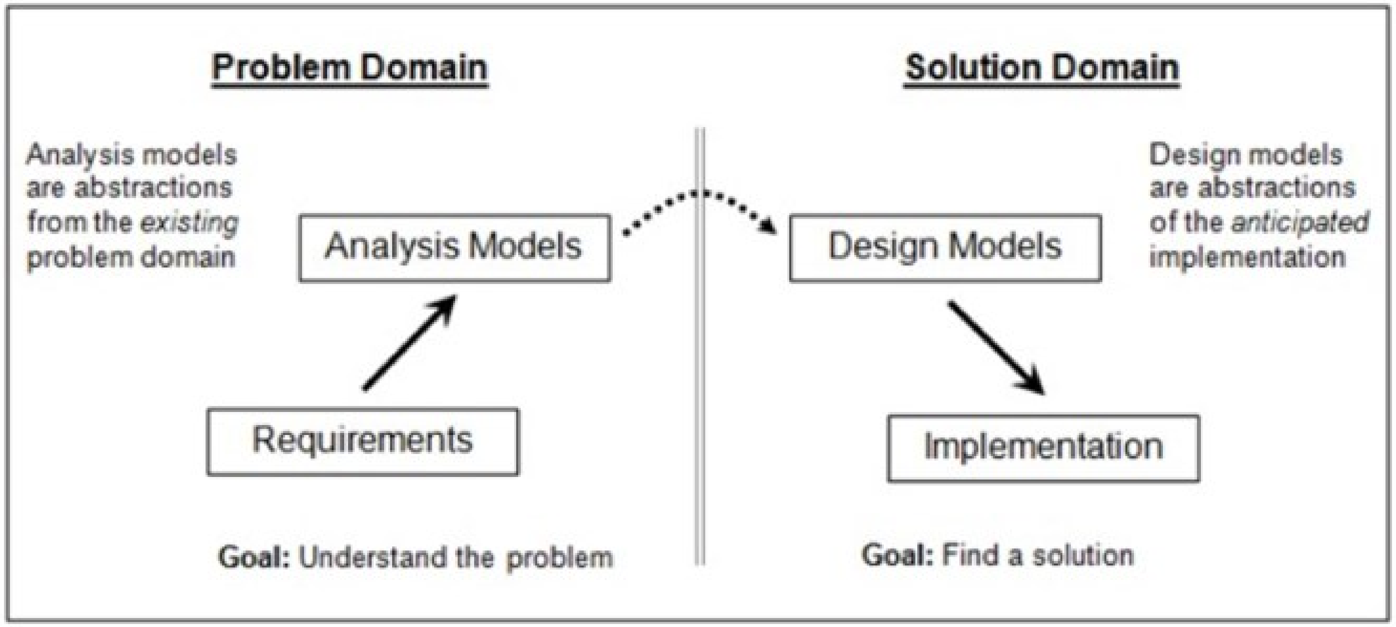
What a logical system does…:

- **Memorize**: to maintain the **state of the system domain**.
- **Inform**: to inform about the **state of the system domain**.
- **Act**: to act to change the **state of the system domain**.



On the design of systems…

- Design is difficult because design is an abstraction of the solution which has yet to be created

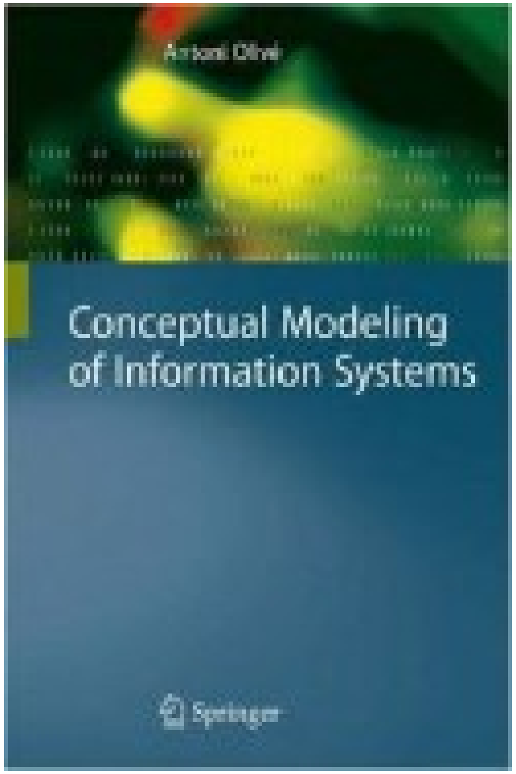

Problem Domain
Solution Domain
Analysis models are abstractions from the *existing* problem domain
Analysis Models
Requirements
Design Models
Implementation
Design models are abstractions of the *anticipated* implementation
Goal: Understand the problem
Goal: Find a solution

http://slideplayer.com/slide/5265721/

The System Domain

“In the field of information systems, we make the fundamental assumption that **a domain consists of a set of objects and the relationships between them, which are classified into concepts.**
The state of a particular domain, at a given time, therefore consists of a set of objects, a set of relationships, and a set of concepts into which these objects and relationships are classified.

For example, in the domain of a company, we may have the concepts of a customer, a product and a sale. At a given moment, we have objects classified as customers, objects classified as products, and relationships between customers and products classified as sales.”


Antoni Olivé

Conceptual Modeling of Information Systems

Springer

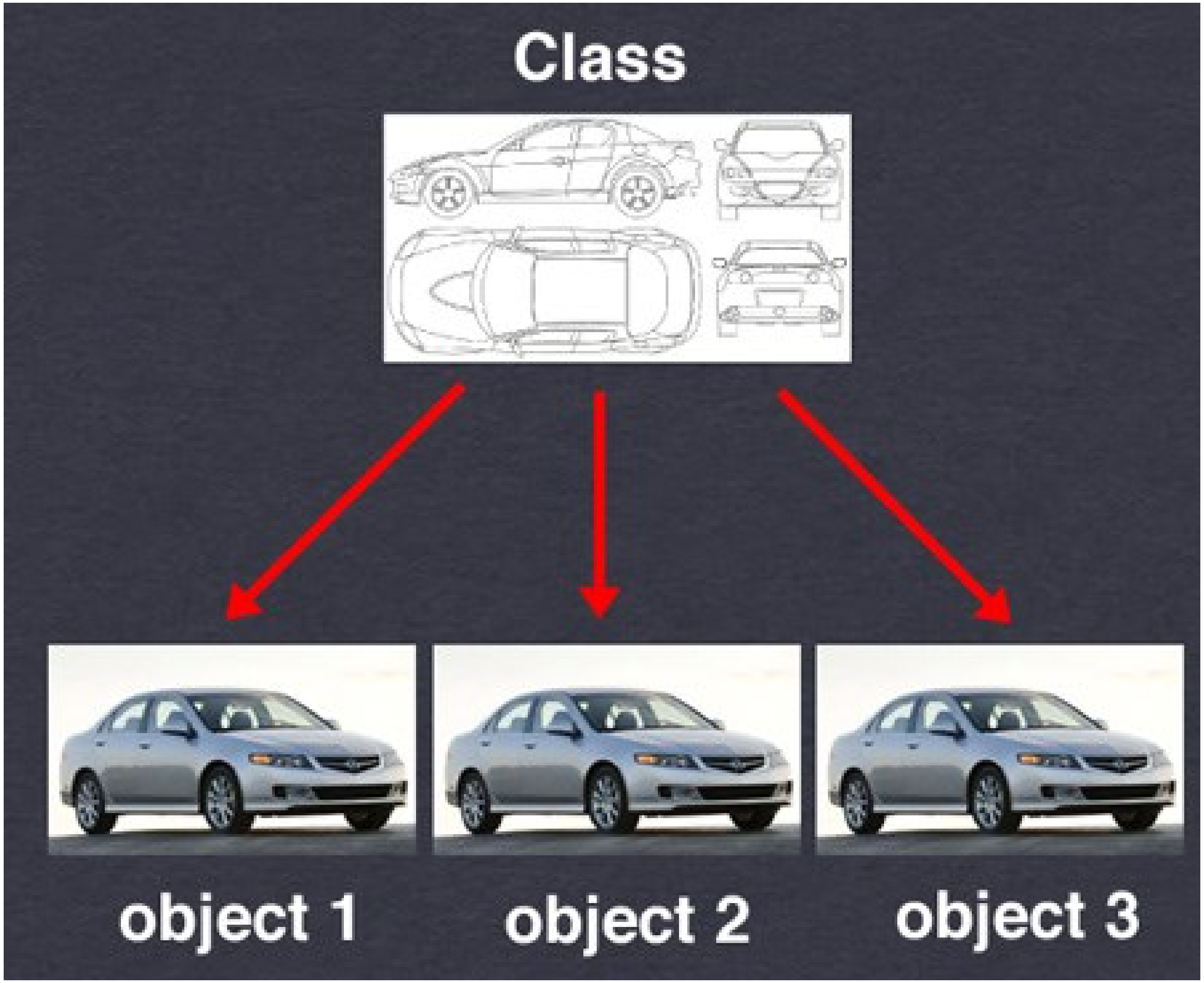
Object-Orientation and Abstraction

- A way of thinking about the world and solving problems.
- We can conceive the world as objects which we can understand and describe.
- We conceptualize systems as representations of objects interacting with each other.
- Entails the *classification* of objects.
  - Classification is an abstraction process where objects are abstracted as being part of classes.
  - Classes must always be defined according to the interest of the viewer.

Domain and Domain Model

Domain Model

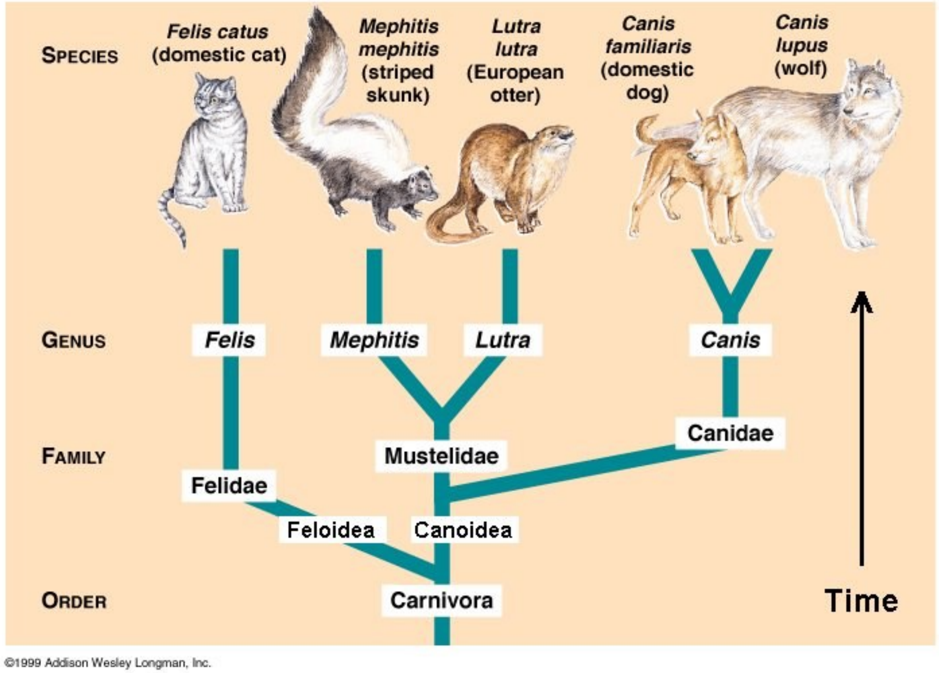
Domain



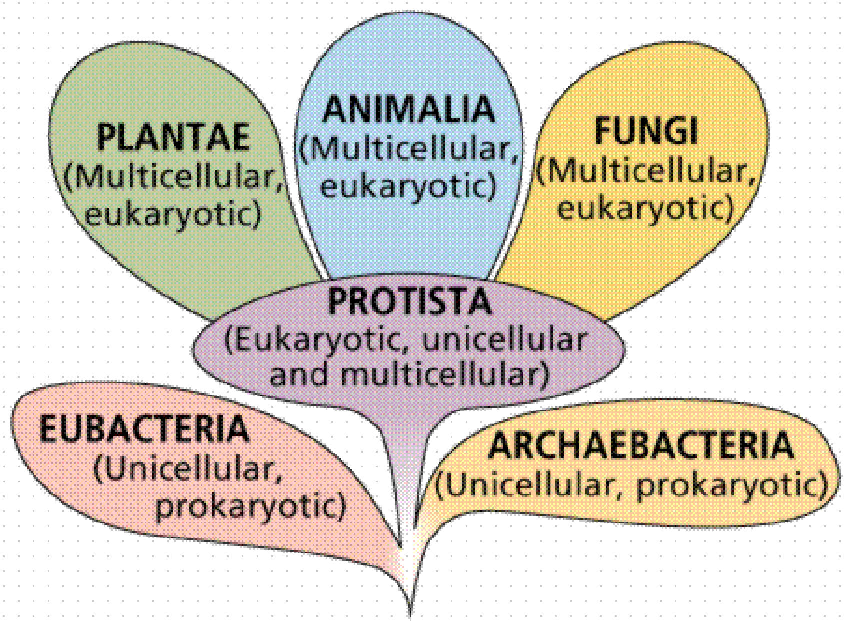
http://multimedia.journalism.berkeley.edu/media/upload/tutorials/flash/actionscript/class-object.jpg

Domain and Domain Model

A domain diagram of a system is a visual representation of part the domain of that system with the representation of the concepts and the relationships between them.


SPECIES
Felis catus (domestic cat)
Mephitis mephitis (striped skunk)
Lutra lutra (European otter)
Canis familiaris (domestic dog)
Canis lupus (wolf)
GENUS
Felis
Mephitis
Lutra
Canis
FAMILY
Felidae
Mustelidae
Canidae
Feloidea
Canoidea
ORDER
Carnivora
Time
©1999 Addison Wesley Longman, Inc.

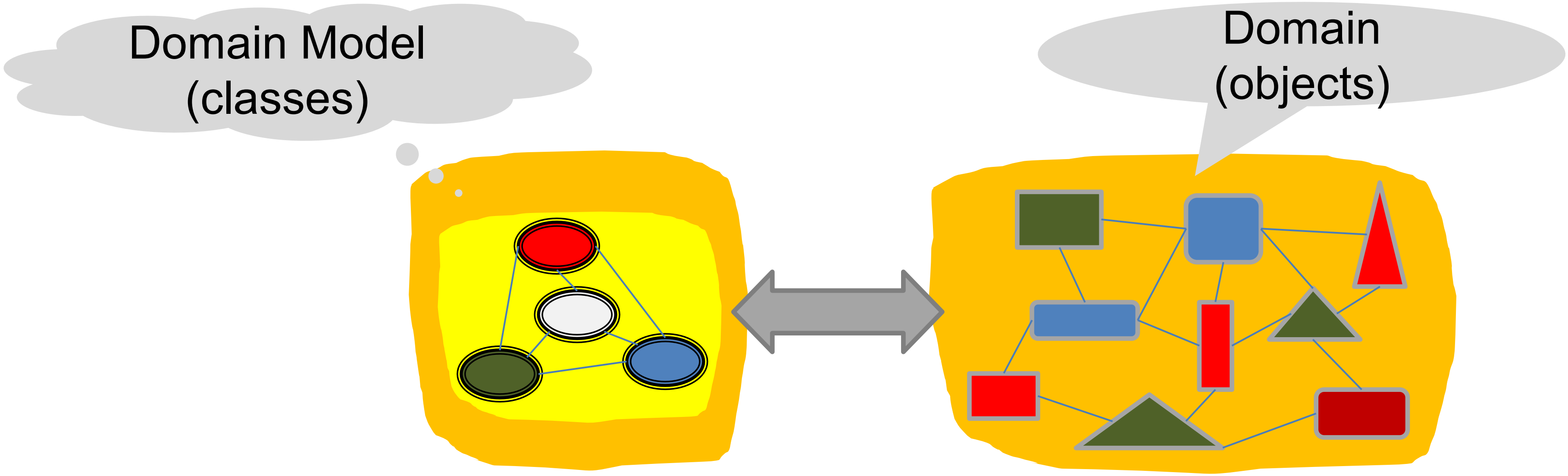
http://www.mun.ca/biology/scarr/139416_Natural_classification.jpg


PLANTAE (Multicellular, eukaryotic)
ANIMALIA (Multicellular, eukaryotic)
FUNGI (Multicellular, eukaryotic)
PROTISTA (Eukaryotic, unicellular and multicellular)
EUBACTERIA (Unicellular, prokaryotic)
ARCHAEBACTERIA (Unicellular, prokaryotic)

http://www.emc.maricopa.edu/faculty/farabee/biobk/kingdoms.gif

Domain and Domain Model

- The **domain** of a system is the set of objects and associations between these objects that in a specific moment fully define the system.
- A **domain model** of a system is a conceptualization [*classification*] of the **domain** of that system.
- In UML
  - A **domain model** is represented by a UML **class diagram**.
  - A **domain (diagram)** is a UML **object diagram**.

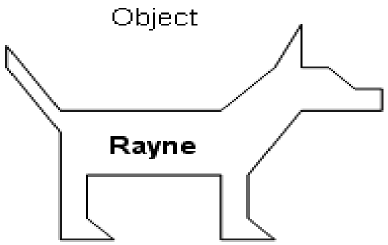


Domain Model
(classes)

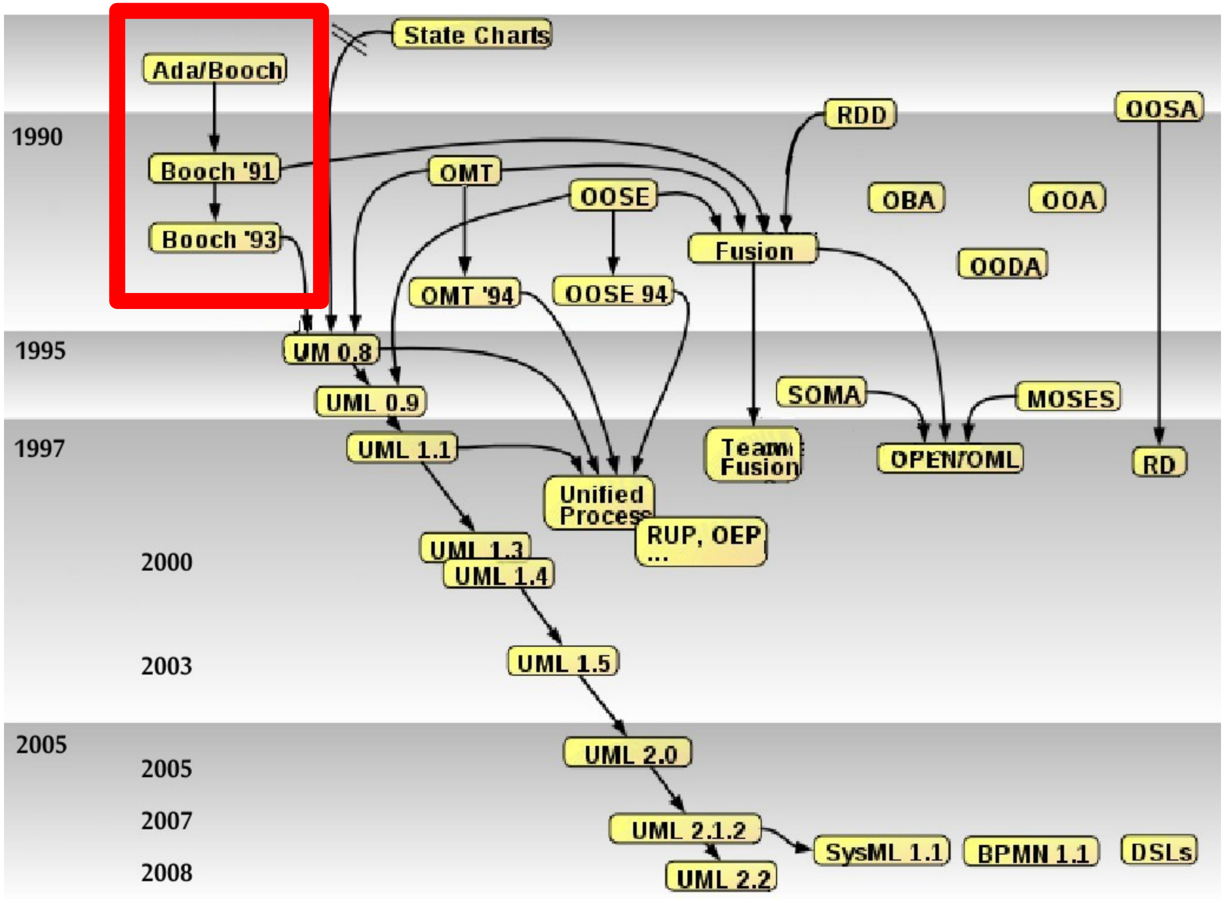
Domain
(objects)

The four principles of object-orientation according to Booch

1. Abstraction
2. Encapsulation
3. Modularity
4. Hierarchy



Property values	**Methods**
Color: Gray, White, and Black	Sit
Eye Color: Blue and Brown	Lay Down
Height: 18 Inches	Shake
Length: 36 Inches	Come
Weight: 30 Pounds	

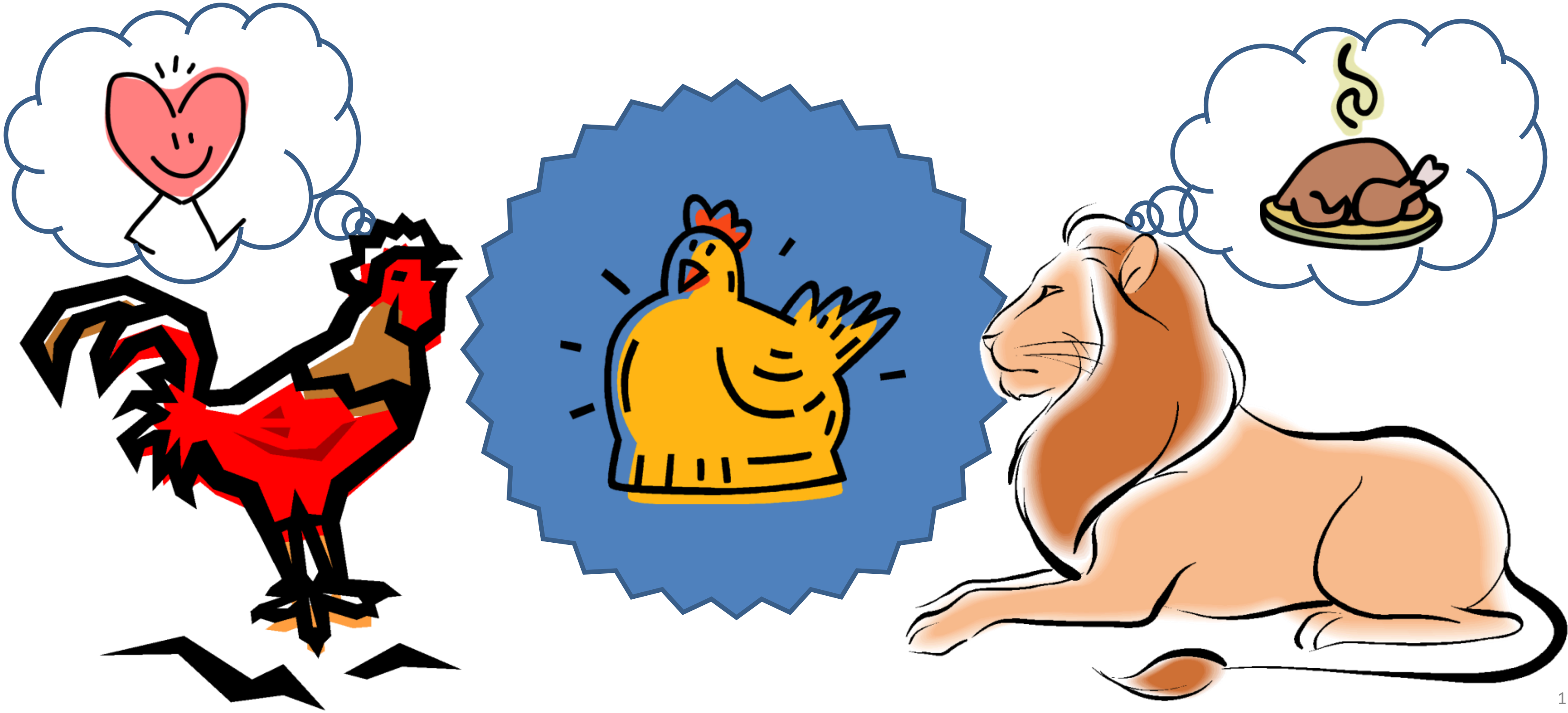


http://sitepointstatic.com/graphics/03fig04.png

Abstraction

(focus in what matters)

An **abstraction** denotes **the essential characteristics of an object that distinguish it from all other kinds of objects** and thus provide well-defined conceptual boundaries **relative to the perspective of the viewer**.



Encapsulation

(self contextualize)

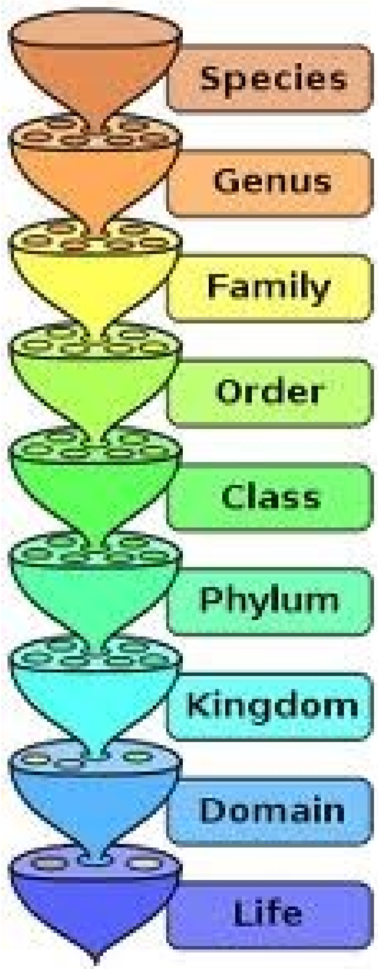
Encapsulation is the process of **compartmentalizing the elements of an abstraction that constitute its structure and behavior**; encapsulation serves to separate the contractual interface of an abstraction and its implementation.

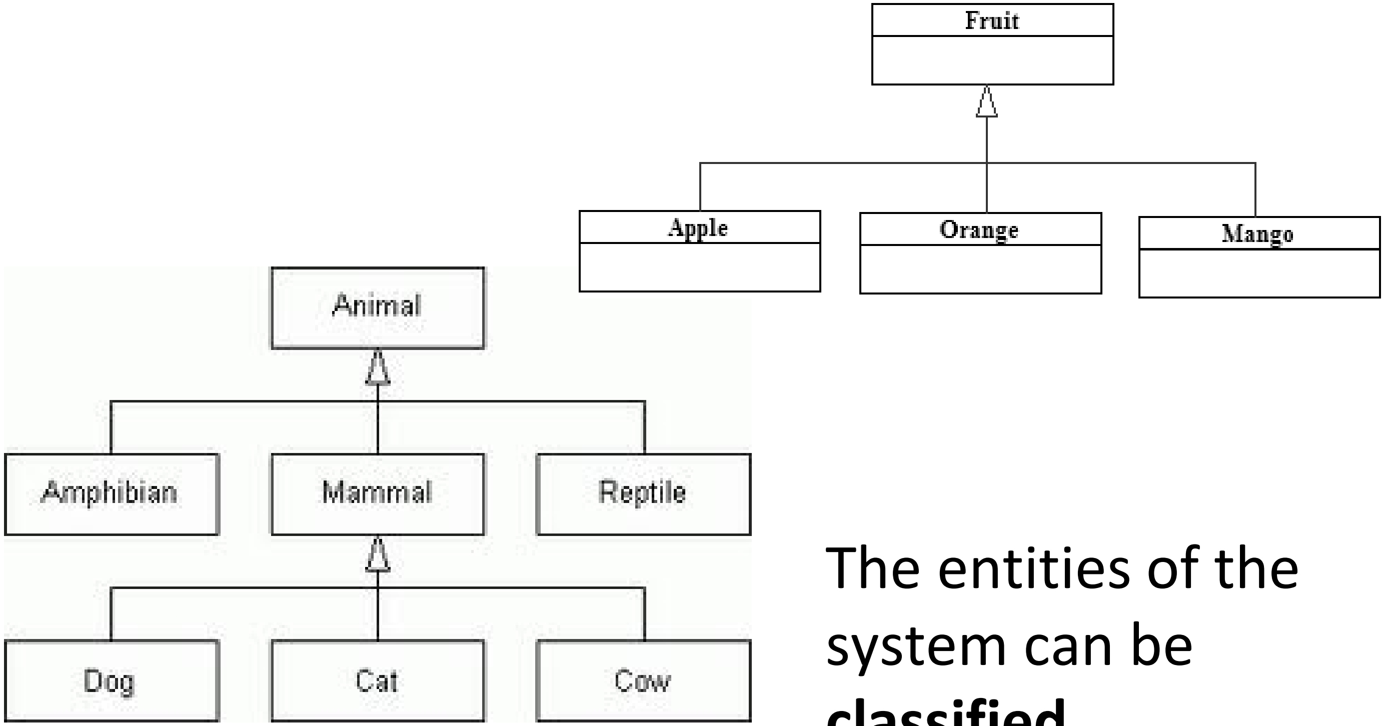
A conceptual model of a system consist of:

- A **structural model:** the description of the **entities**, relationships and concepts of a system.
- A **behavioural model:** the **actions** that the system can perform and how the domain may change.

Hierarchy (*specialization, aka inheritance*)

"**Hierarchy** is a **ranking** or **ordering** of **abstractions**"





The entities of the system can be **classified** according to **hierarchies**.

http://simple.wikipedia.org/wiki/File:Biological_classification_L_Pengo.svg

Modularity

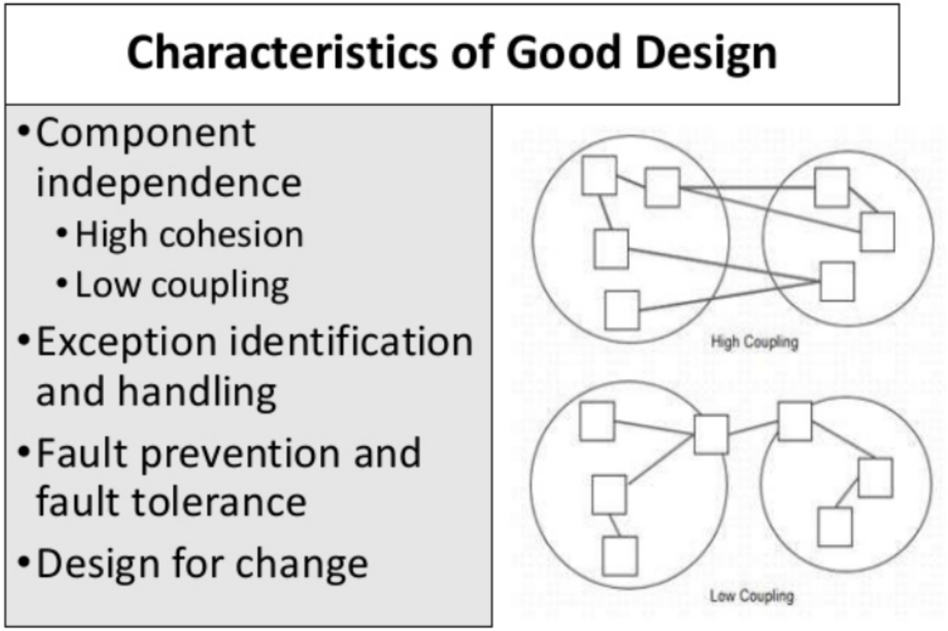
(divide and conquer)

Modularity is the property of a system that has been **decomposed** into a set of **highly cohesive** and **loosely coupled** modules.

The **domain** of a system can be decomposed as a structure of interrelated entities (objects)

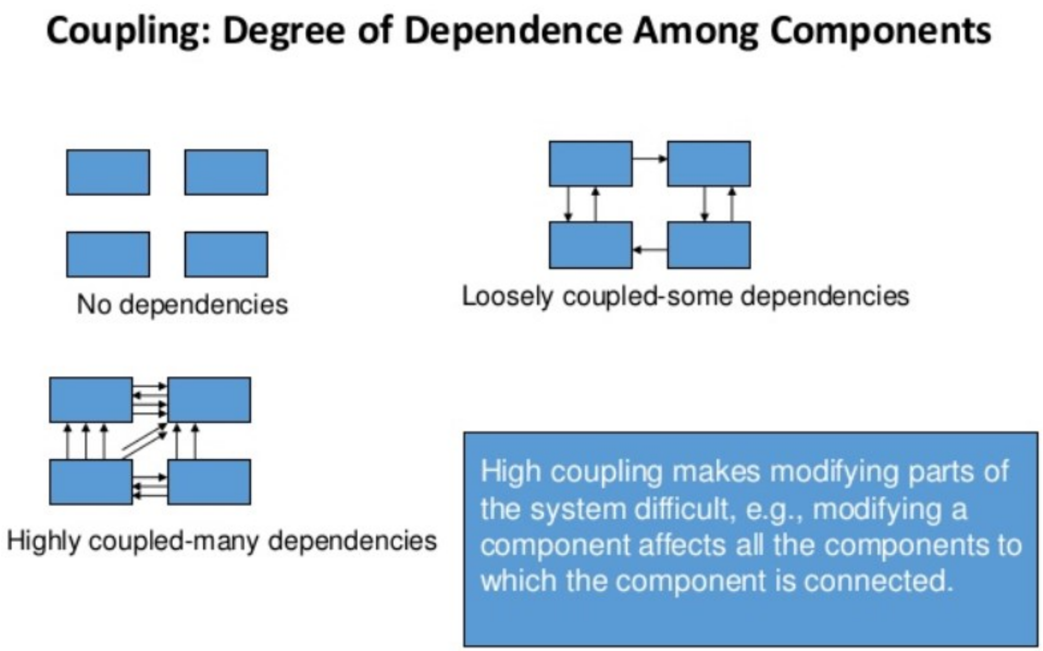
Modularity & Coupling

Coupling between modules refers to their degree of mutual interdependence. In a system's structure, a **lower coupling between the system's entities is always better**.



Characteristics of Good Design

- Component independence
  - High cohesion
  - Low coupling
- Exception identification and handling
- Fault prevention and fault tolerance
- Design for change



Coupling: Degree of Dependence Among Components

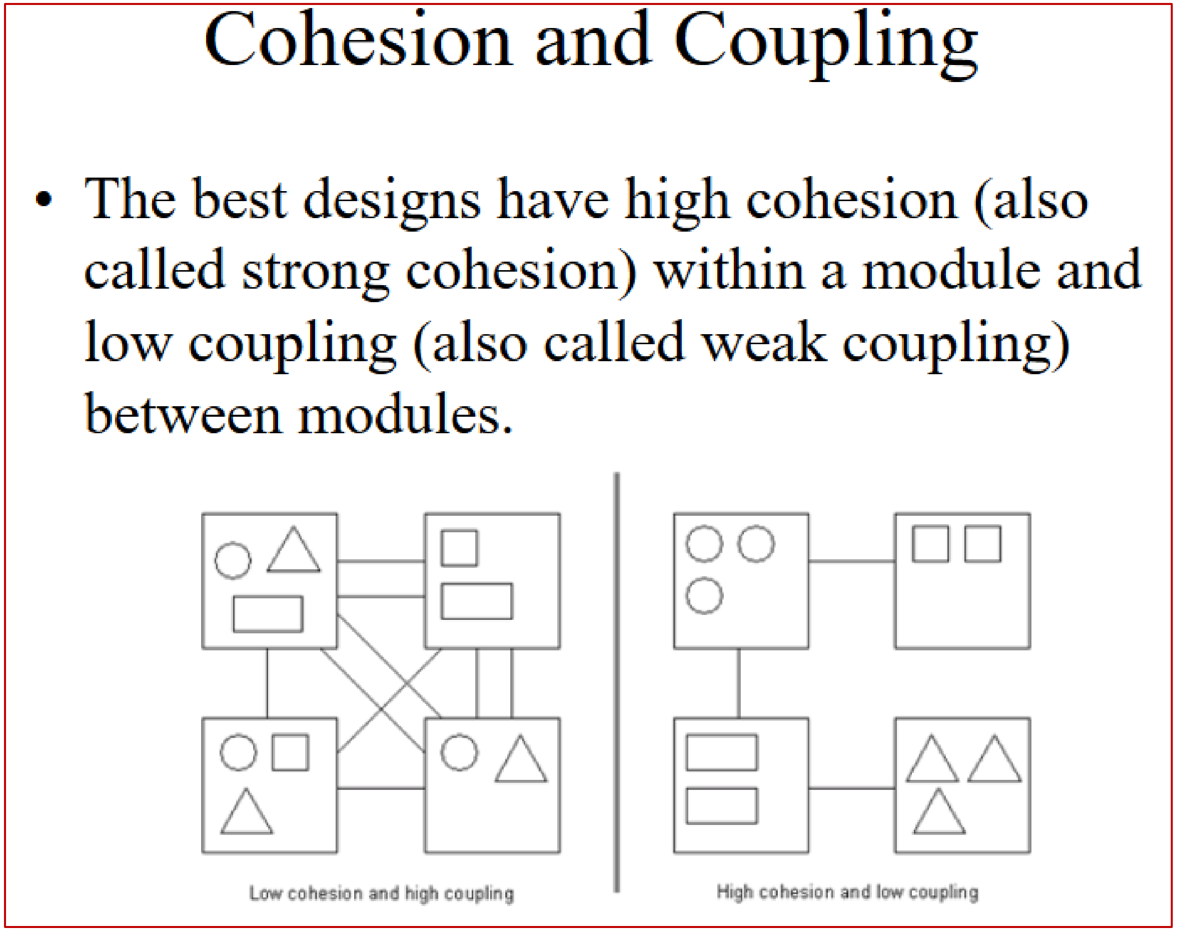
No dependencies
Loosely coupled-some dependencies
Highly coupled-many dependencies

High coupling makes modifying parts of the system difficult, e.g., modifying a component affects all the components to which the component is connected.

https://www.slideshare.net/AdilAslam4/coupling-and-cohesion-in-software-engineering

Modularity & Cohesion

- **Cohesion** is a measure of how strongly-related the pieces of functionality of an object are **Higher is better**.
- A and B are **coupled** when B must change behaviour only because A changed.
  - Coincidental cohesion (worst)
  - Logical cohesion
  - Temporal cohesion
  - Procedural cohesion
  - Communicational cohesion
  - Sequential cohesion
  - Functional cohesion (best)



Cohesion and Coupling

- The best designs have high cohesion (also called strong cohesion) within a module and low coupling (also called weak coupling) between modules.

Low cohesion and high coupling

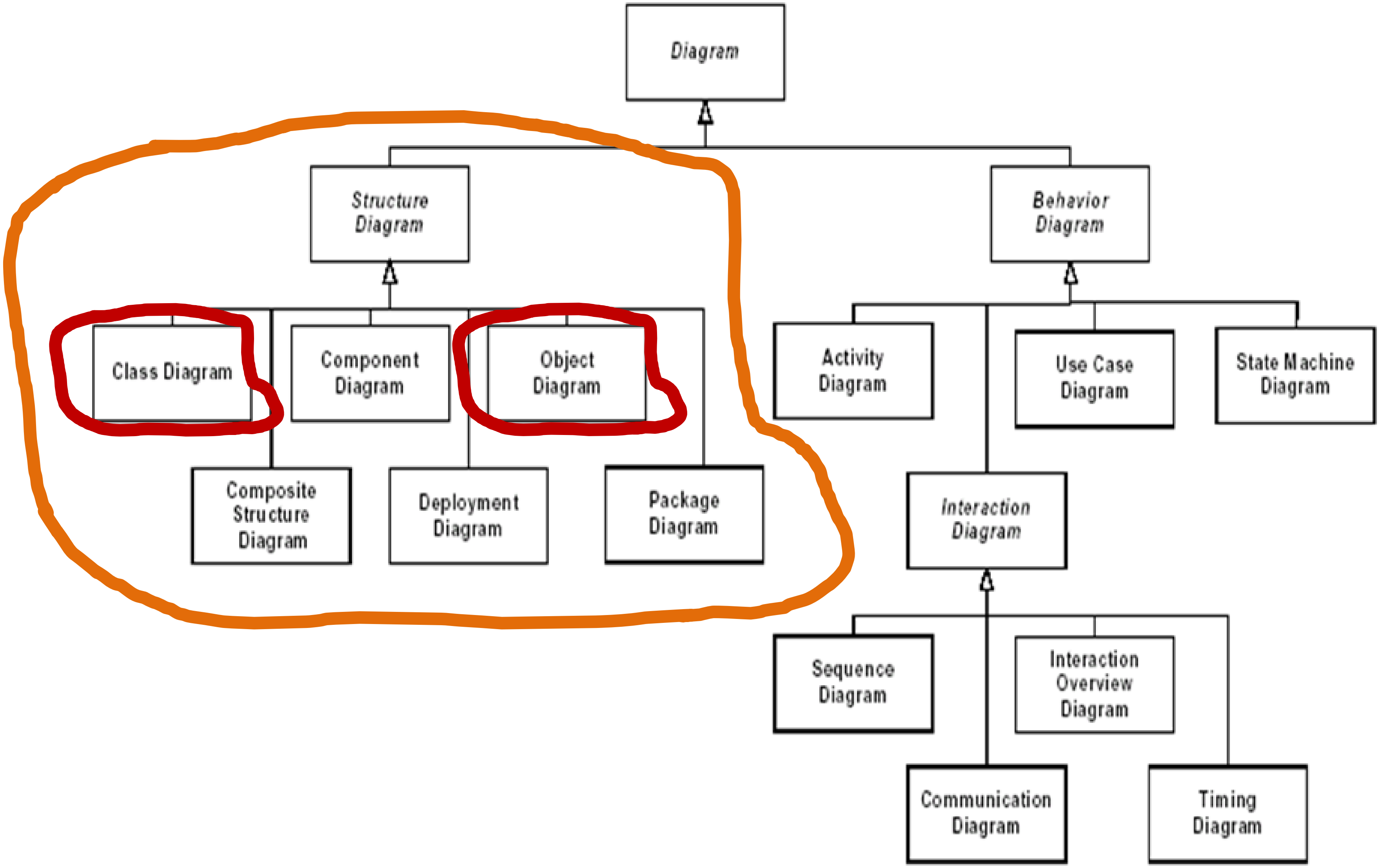
High cohesion and low coupling

http://slideplayer.com/slide/5265721/

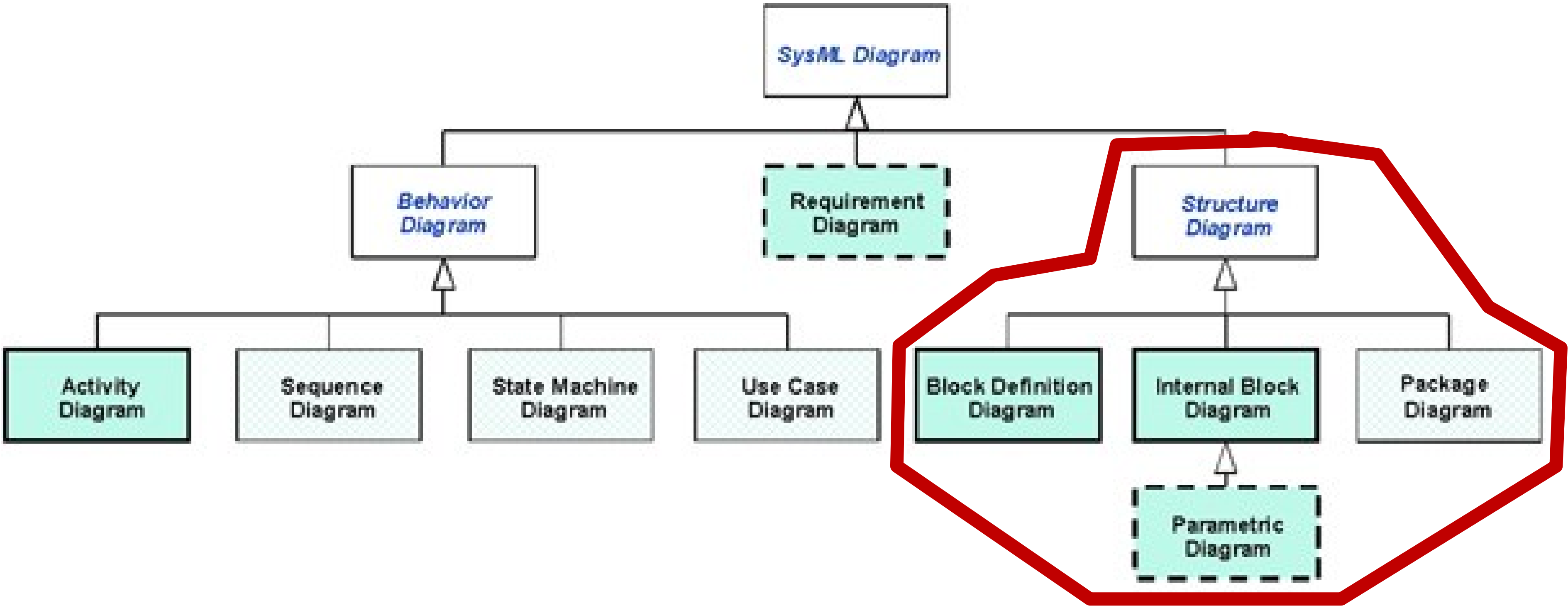
Structural Modelling with UML and SysML

…

Structure Modelling in UML

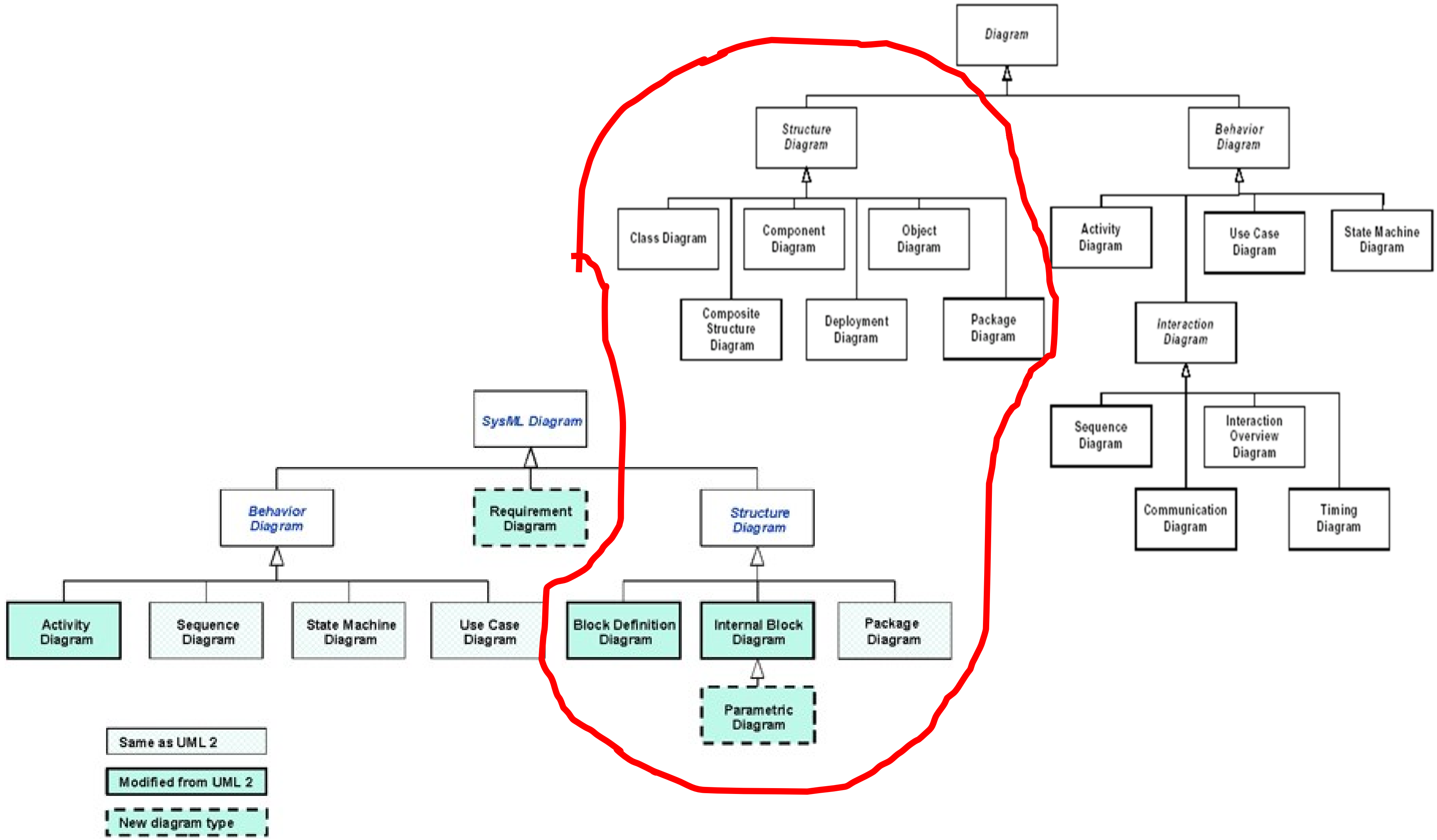


Structural Diagrams in SysML



- The **«block»** is the basic unit of structure in SysML.
- It can be used to represent any kind of hardware (including facilities, persons, etc.), software, or any other system element.
- The system structure is represented by **block definition diagrams** and **internal block diagrams**.

Structural diagrams in UML and SysML



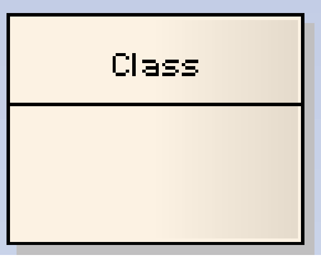
OO Vocabulary

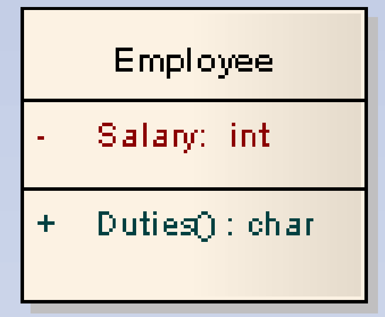
Abstract class	A class that does not have objects instantiated from it
Abstraction	The identification of the essential characteristics of an item
Aggregation	Represents “is part of” or “contains” relationships between two classes or components
Aggregation hierarchy	A set of classes that are related through aggregation
Association	Objects are related (associated) to other objects
Attribute	Something that a class knows (data/information)
Class	A software abstraction of similar objects, a template from which objects are created
Cohesion	The degree of relatedness of an encapsulated unit (such as a component or a class)
Collaboration	Classes work together (collaborate) to fulfill their responsibilities
Composition	A strong form of aggregation in which the “whole” is completely responsible for its parts and each “part” object is only associated to the one “whole” object
Concrete class	A class that has objects instantiated from it
Coupling	The degree of dependence between two items
Encapsulation	The grouping of related concepts into one item, such as a class or component
Information hiding	The restriction of external access to attributes
Inheritance	Represents “is a”, “is like”, and “is kind of” relationships. When class “B” inherits from class “A” it automatically has all of the attributes and operations that “A” implements (or inherits from other classes)
Inheritance hierarchy	A set of classes that are related through inheritance
Instance	An object is an instance of a class
Instantiate	We instantiate (create) objects from classes

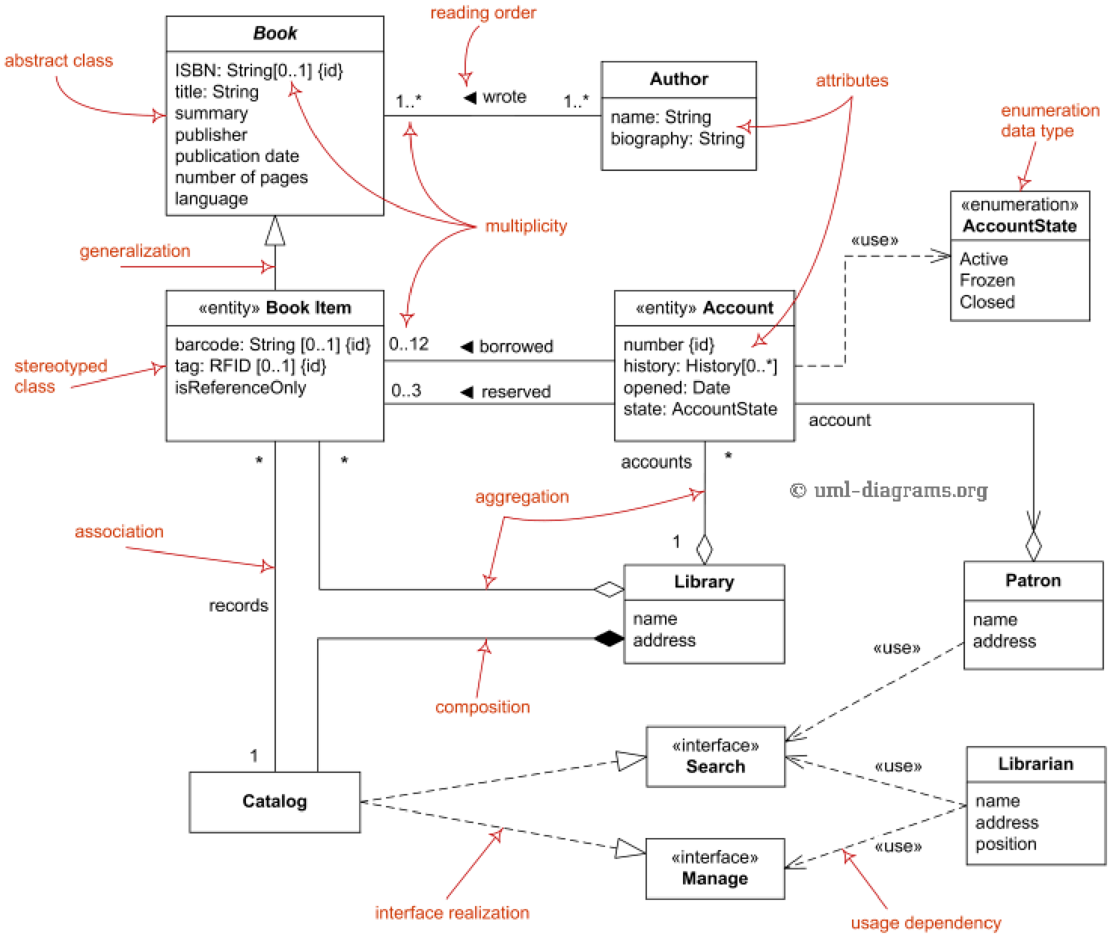
Interface	The definition of a collection of one or more operation signatures that defines a cohesive set of behaviors
Message	A message is either a request for information or a request to perform an action
Messaging	In order to collaborate, classes send messages to each other
Multiple inheritance	When a class directly inherits from more than one class
Multiplicity	A UML concept combining the data modeling concepts of cardinality (how many) and optionality.
Object	A person, place, thing, event, concept, screen, or report
Object space	Main memory + all available storage space on the network, including persistent storage such as a relational database
Operation	Something a class does (similar to a function in structured programming)
Override	Sometimes you need to override (redefine) attributes and/or methods in subclasses
Pattern	A reusable solution to a common problem taking relevant forces into account
Persistence	The issue of how objects are permanently stored
Persistent object	An object that is saved to permanent storage
Polymorphism	Different objects can respond to the same message in different ways, enable objects to interact with one another without knowing their exact type
Single inheritance	When a class directly inherits from only one class
Stereotype	Denotes a common usage of a modeling element
Subclass	If class “B” inherits from class “A,” we say that “B” is a subclass of “A”
Superclass	If class “B” inherits from class “A,” we say that “A” is a superclass of “B”
Transient object	An object that is not saved to permanent storage

Classes

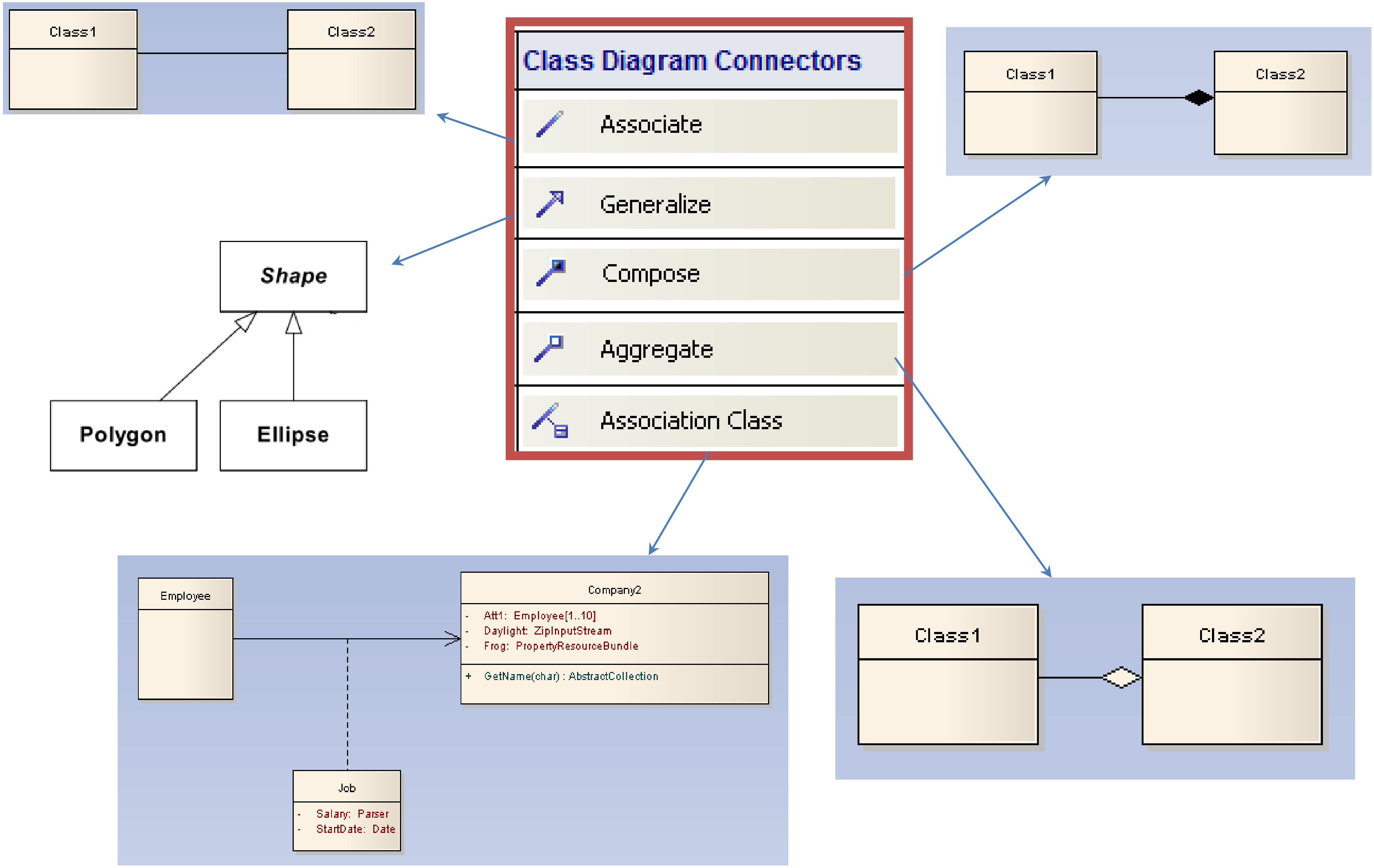
A **class** is an abstraction of a **set of objects** that share the same attributes, operations, relationships and semantics.



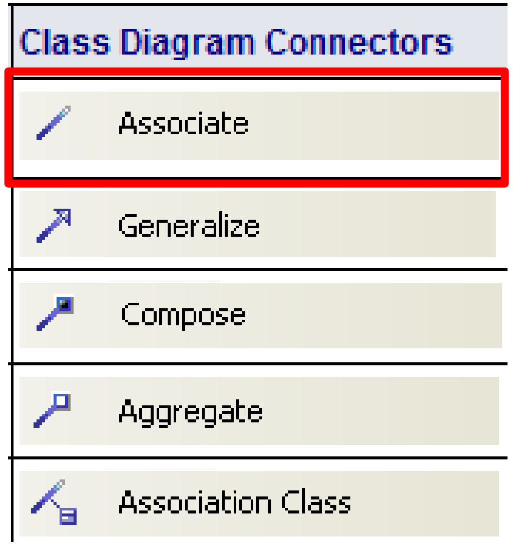




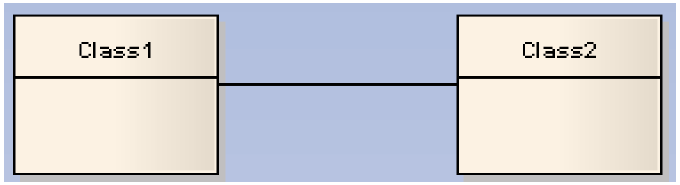
Class Relationships

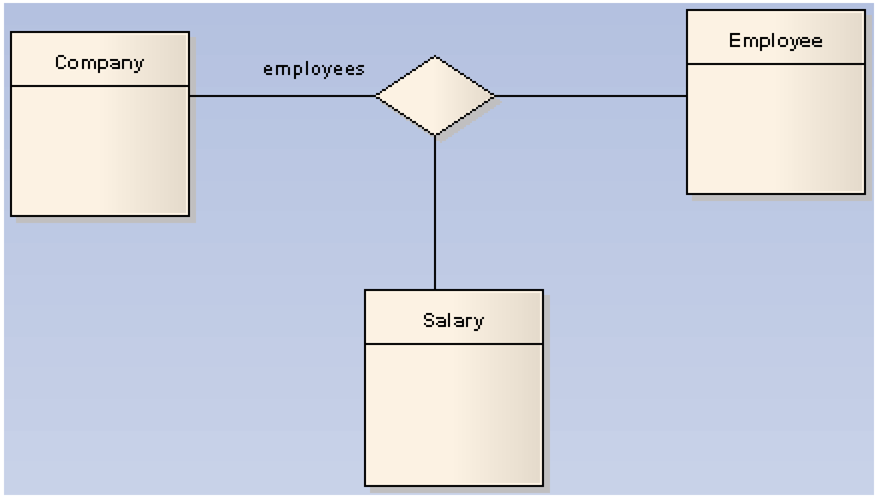


Associations

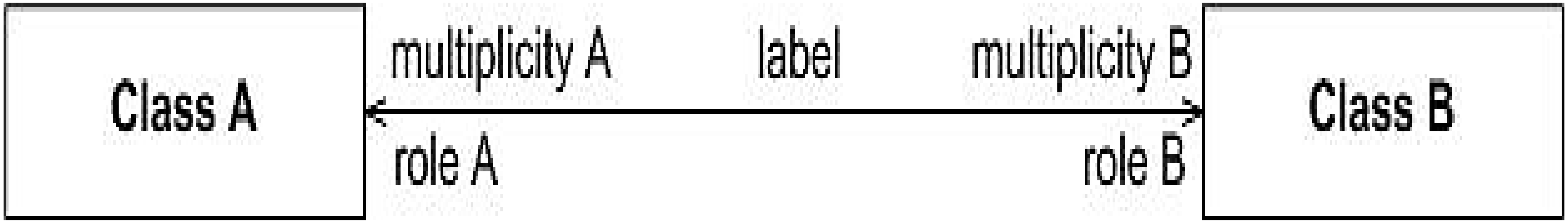


- Association is a generic relationship between elements with weak semantics.

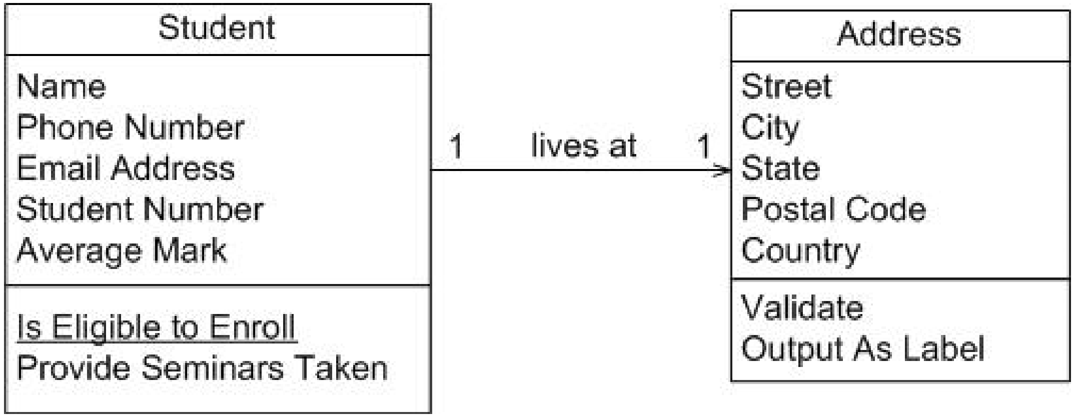




Associations

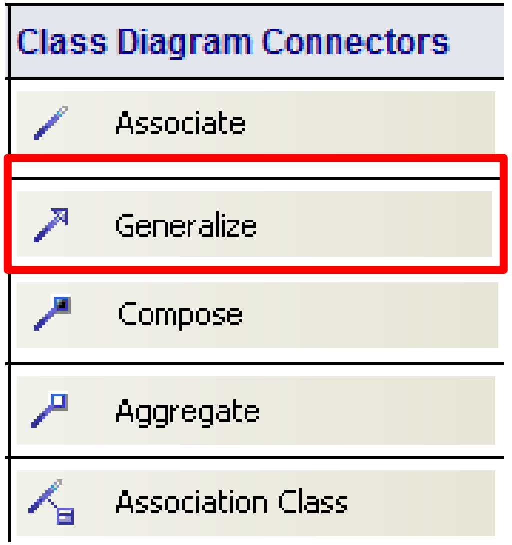


Multiplicity	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	At least one
n	*n* only
0..n	Zero to *n* (where *n* >= 1)
1..n	One to *n* (where *n* > 1)

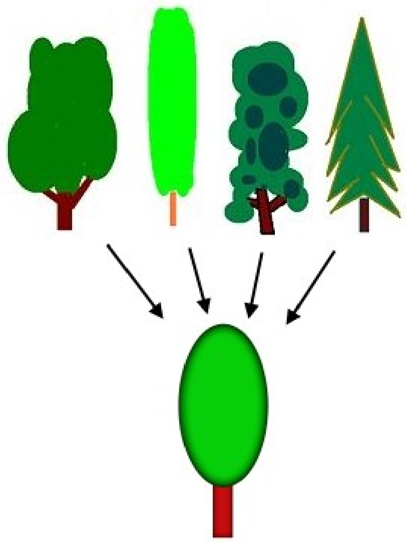


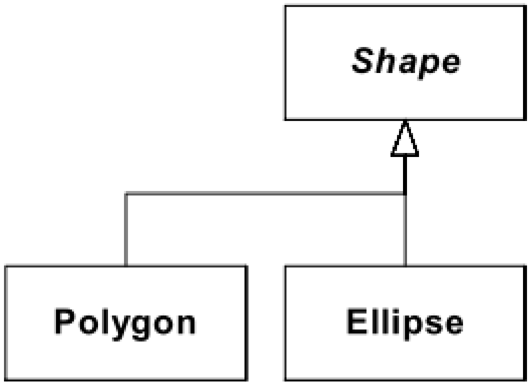
Examples from http://www.agilemodeling.com/artifacts/classDiagram.htm

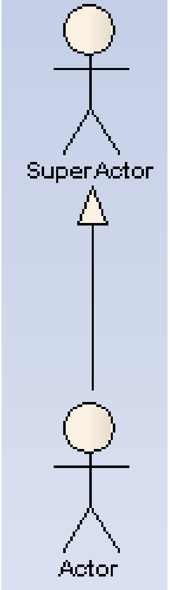
Generalization



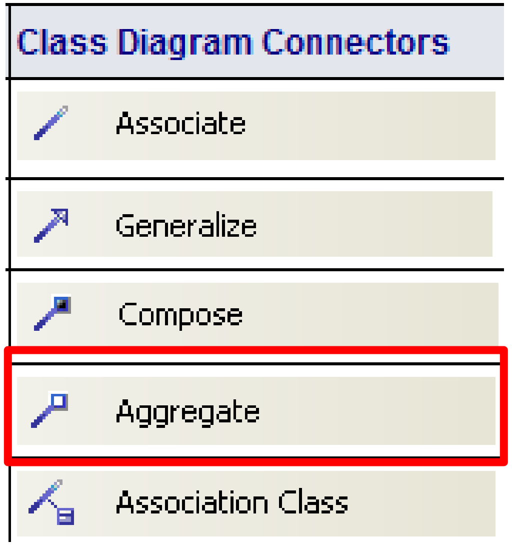
- A is a *generalization* of *B* iff
  1. Every instance of concept *B* is also an instance of concept *A*
  2. There are instances of concept *A* which are not instances of concept *B*
- *Animal* is a generalization of *Bird* because (1) every *Bird* is an *Animal* and (2) some *Animals* are not *Birds*.
- A *generalizes* B => B ***is kind of*** A







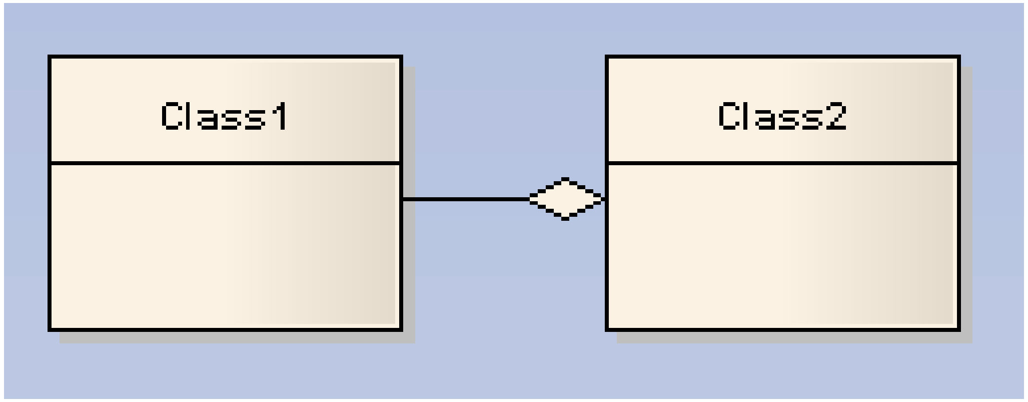
Aggregation



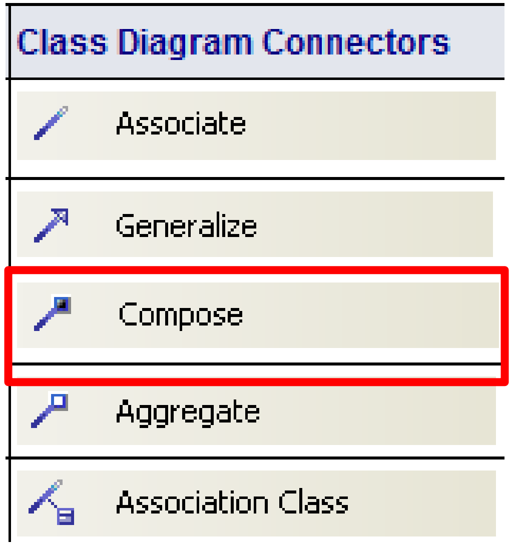
Class Diagram Connectors

- A (weak) **aggregation** specifies that an element contains other elements.

- Class2 ***contains*** Class1

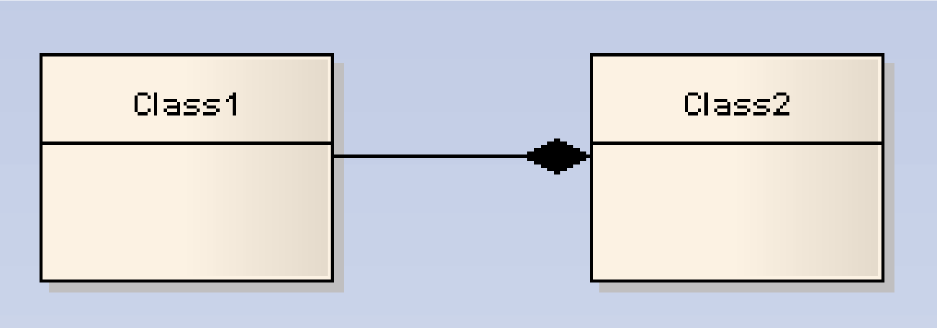


Composite Aggregation



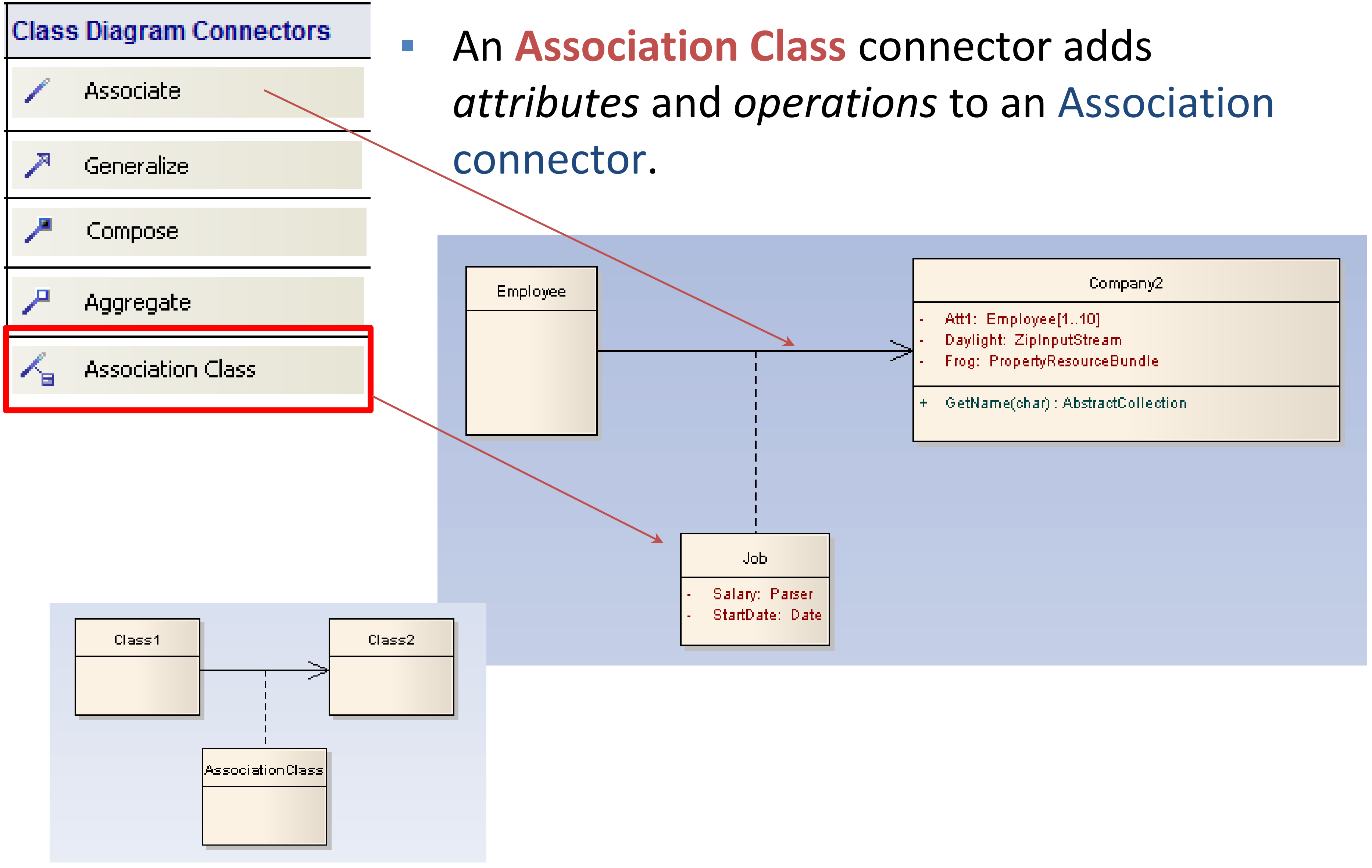
Class Diagram Connectors

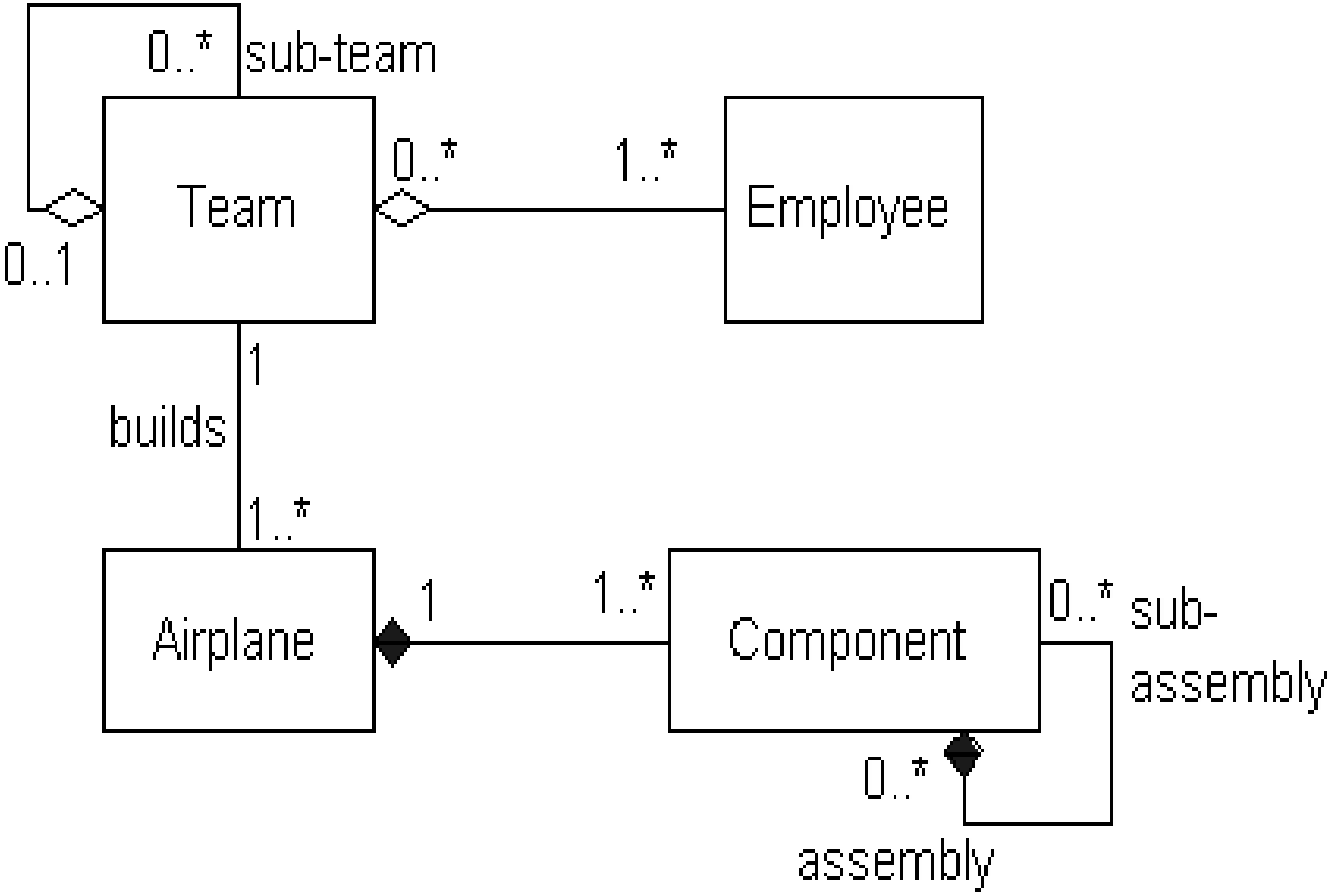
- A **composite aggregation** specifies that an element is composed by other elements.
- An instance can only be included in one composition at a time (strong form of aggregation).
- Class1 ***is part of*** Class2



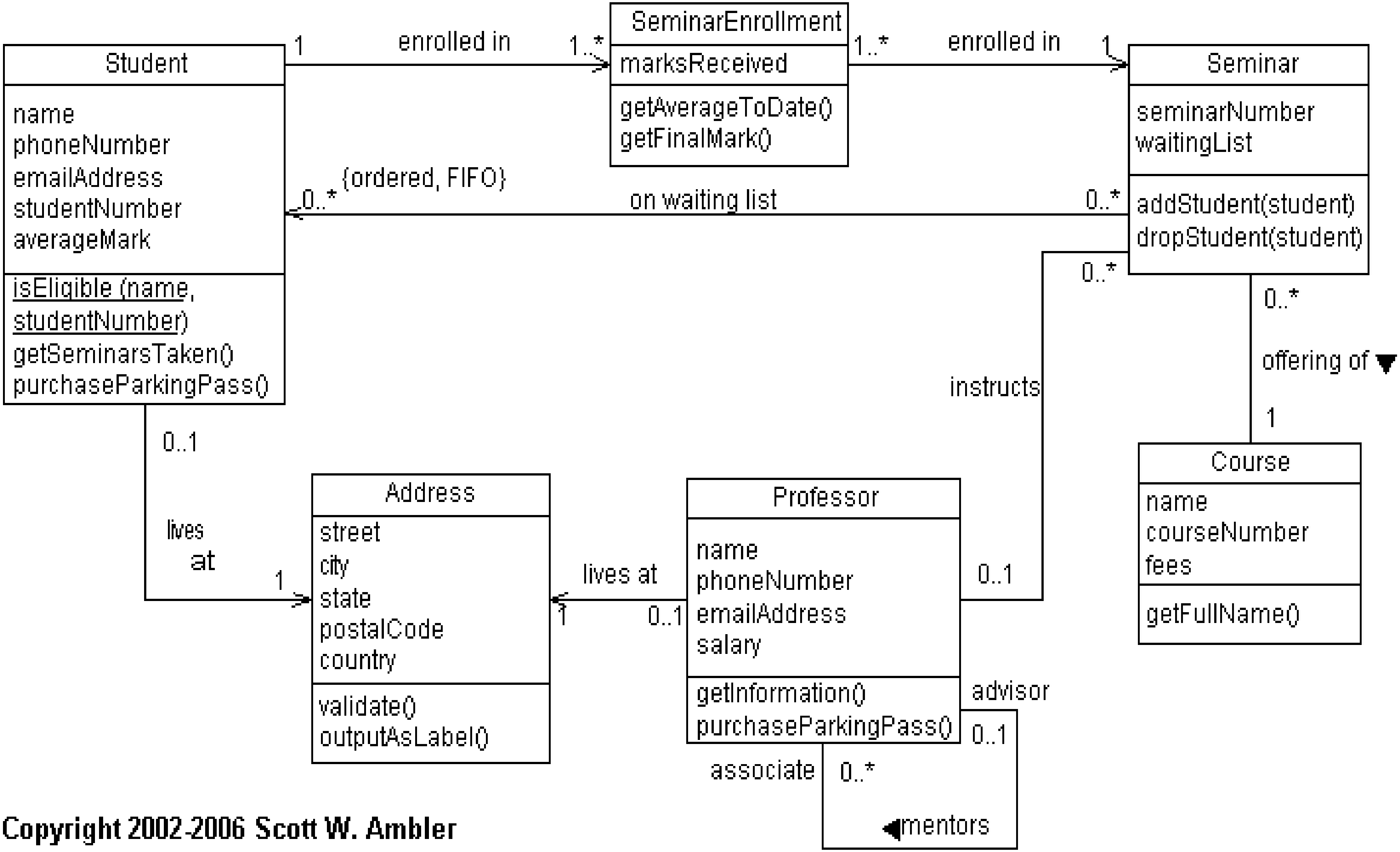
Association Class

- An **Association Class** connector adds *attributes* and *operations* to an Association connector.





http://www.agilemodeling.com/style/classDiagram.htm



Student

name
phoneNumber
emailAddress
studentNumber
averageMark

isEligible (name, studentNumber)
getSeminarsTaken()
purchaseParkingPass()

SeminarEnrollment

marksReceived

getAverageToDate()
getFinalMark()

Seminar

seminarNumber
waitingList

addStudent(student)
dropStudent(student)

Course

name
courseNumber
fees

getFullName()

Address

street
city
state
postalCode
country

validate()
outputAsLabel()

Professor

name
phoneNumber
emailAddress
salary

getInformation()
purchaseParkingPass()

1 enrolled in 1..*

1..* enrolled in 1

{ordered, FIFO}

0..* on waiting list 0..*

0..* instructs 0..1

0..* offering of ▼ 1

0..1 lives at 1

lives at 1 0..1

advisor 0..1

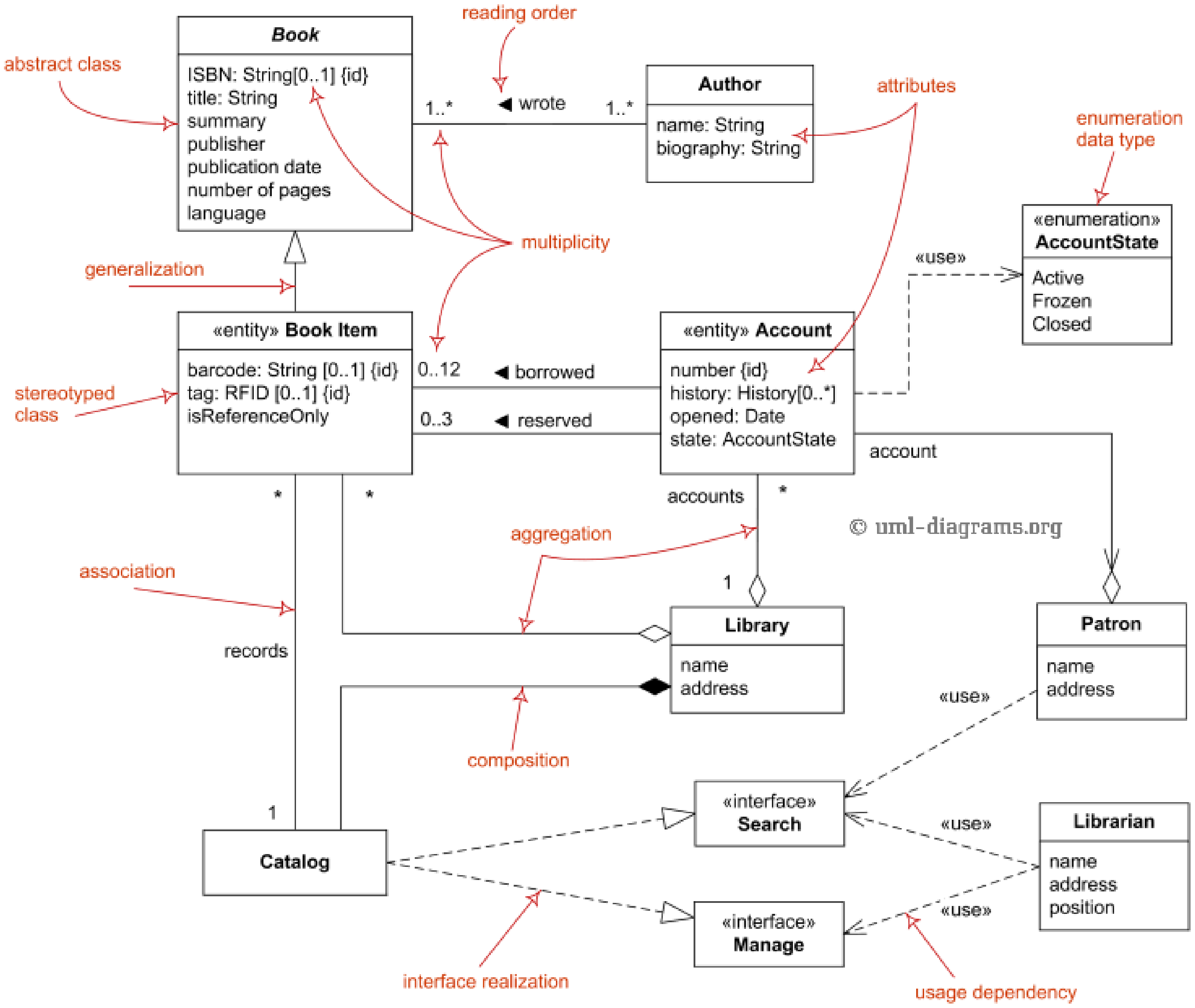
associate 0..*

◀mentors

Copyright 2002-2006 Scott W. Ambler

(source: S. Ambler, The Object Primer, 3rd edition, Cambridge, 2004)

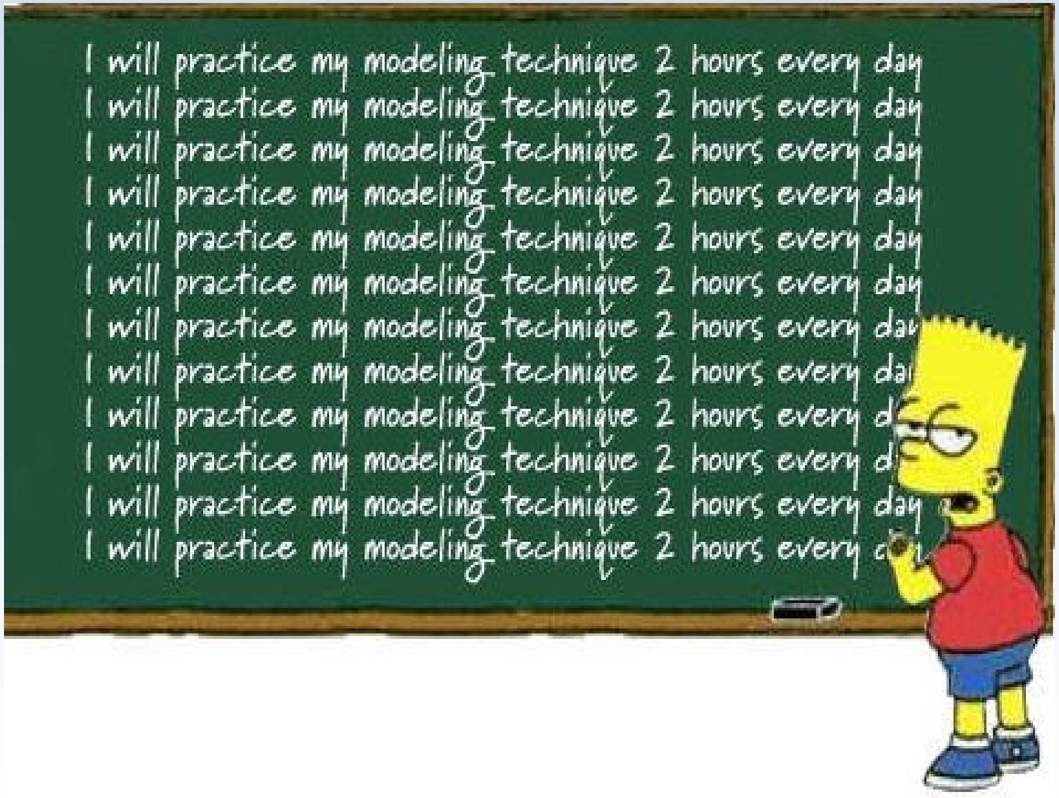
(Domain) Class Diagrams - Summary



https://www.uml-diagrams.org/class-diagrams-overview.html

Structural Modelling with UML

exercise…

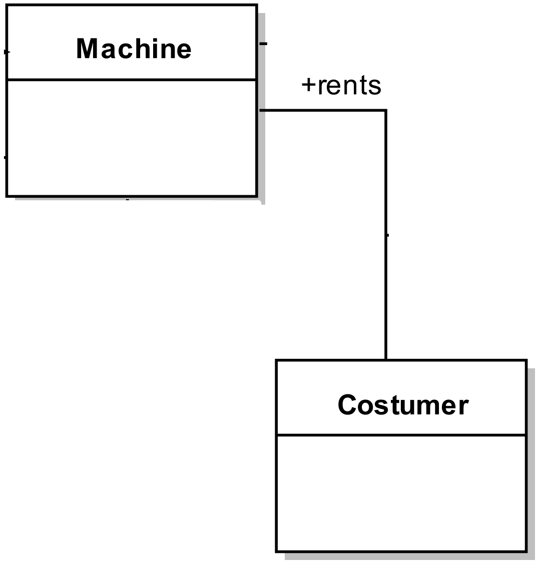

I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every da
I will practice my modeling technique 2 hours every da
I will practice my modeling technique 2 hours every d
I will practice my modeling technique 2 hours every d
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every d

Example “Renting Machines”

“RM: Renting Machines” is a company that rents machines to customers. Customers are know by RM only when they rent the first machine; customers can rent more than one machine. Rents are made according a contract, defined for each case. Each machine is supplied by a specific supplier, who can supplies more then one machine. RM also has inspectors, and each machine must be assigned to two inspectors for regular inspection. The machines also are maintained by engineers; for that role each engineer must have a license. Some of the machines are electrical, with one or more electrical engines. Each electrical engine can be moved from one machine to other, and can be repaired by electricians.

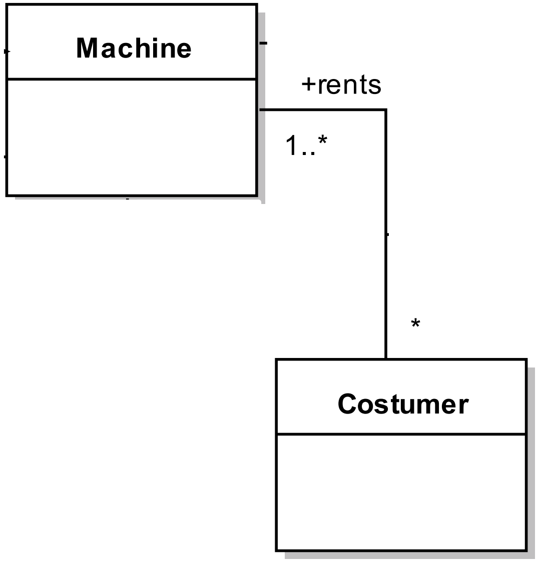
“Renting Machines”

“RM - Renting Machines” is a company that rents machines to customers. Customers are know by RM only when they rent the first machine; customers can rent more than one machine. Rents are made according a contract, defined for each case. Each machine is supplied by a specific supplier, who can supplies more then one machine. RM also has inspectors, and each machine must be assigned to two inspectors for regular inspection. The machines also are maintained by engineers; for that role each engineer must have a license. Some of the machines are electrical, with one or more electrical engines. Each electrical engine can be moved from one machine to other, and can be repaired by electricians.



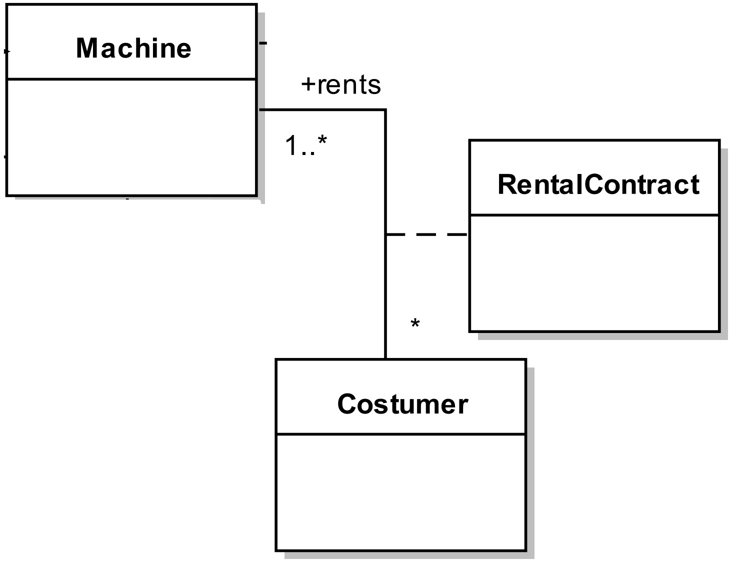
“Renting Machines”

“RM - Renting Machines” is a company that rents machines to customers. **Customers are know by RM only when they rent the first machine; customers can rent more than one machine.** Rents are made according a contract, defined for each case. Each machine is supplied by a specific supplier, who can supplies more then one machine. RM also has inspectors, and each machine must be assigned to two inspectors for regular inspection. The machines also are maintained by engineers; for that role each engineer must have a license. Some of the machines are electrical, with one or more electrical engines. Each electrical engine can be moved from one machine to other, and can be repaired by electricians.



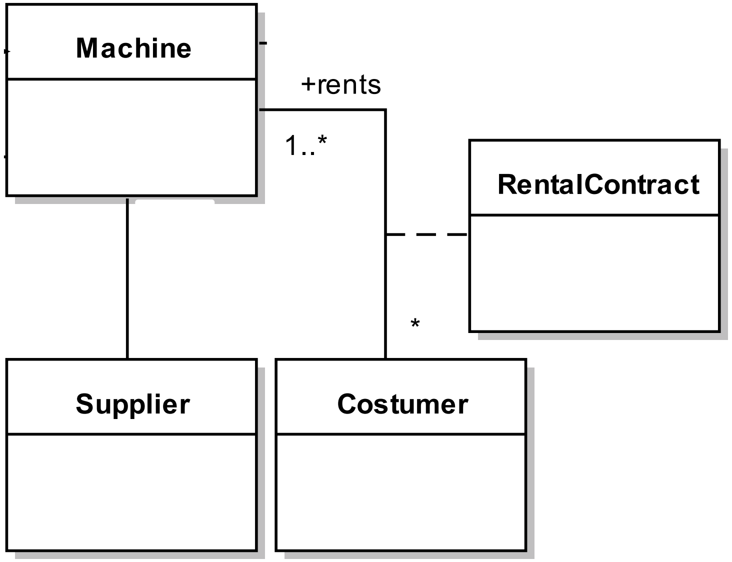
“Renting Machines”

“RM - Renting Machines” is a company that rents machines to costumers. Costumers are know by RM only when they rent the first machine; costumers can rent more than one machine. **Rents are made according a contract, defined for each case.** Each machine is supplied by a specific supplier, who can supplies more then one machine. RM also has inspectors, and each machine must be assigned to two inspectors for regular inspection. The machines also are maintained by engineers; for that role each engineer must have a license. Some of the machines are electrical, with one or more electrical engines. Each electrical engine can be moved from one machine to other, and can be repaired by electricians.



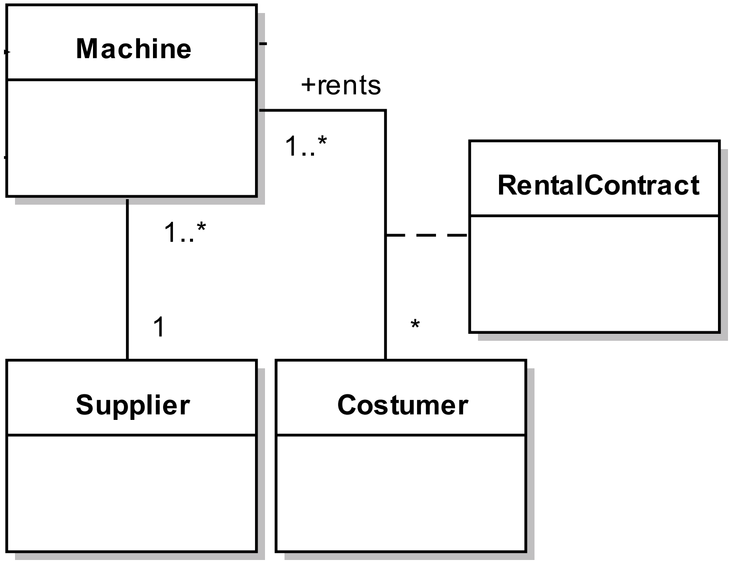
“Renting Machines”

“RM - Renting Machines” is a company that rents machines to costumers. Costumers are know by RM only when they rent the first machine; costumers can rent more than one machine. Rents are made according a contract, defined for each case. **Each machine is supplied by a specific supplier** who can supplies more then one machine. RM also has inspectors, and each machine must be assigned to two inspectors for regular inspection. The machines also are maintained by engineers; for that role each engineer must have a license. Some of the machines are electrical, with one or more electrical engines. Each electrical engine can be moved from one machine to other, and can be repaired by electricians.



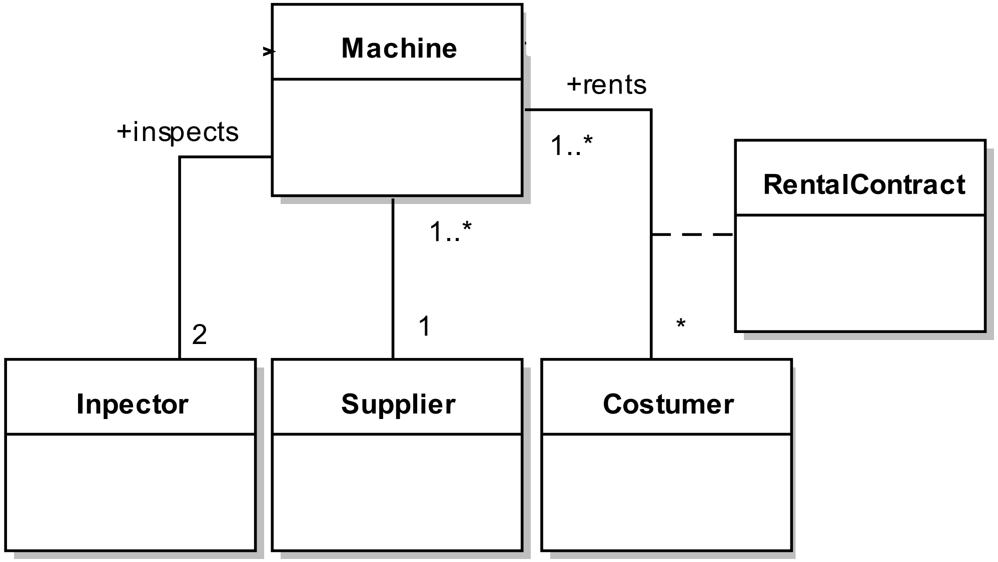
“Renting Machines”

“RM - Renting Machines” is a company that rents machines to costumers. Costumers are know by RM only when they rent the first machine; costumers can rent more than one machine. Rents are made according a contract, defined for each case. Each machine is supplied by a specific supplier, **who can supply more then one machine**. RM also has inspectors, and each machine must be assigned to two inspectors for regular inspection. The machines also are maintained by engineers; for that role each engineer must have a license. Some of the machines are electrical, with one or more electrical engines. Each electrical engine can be moved from one machine to other, and can be repaired by electricians.



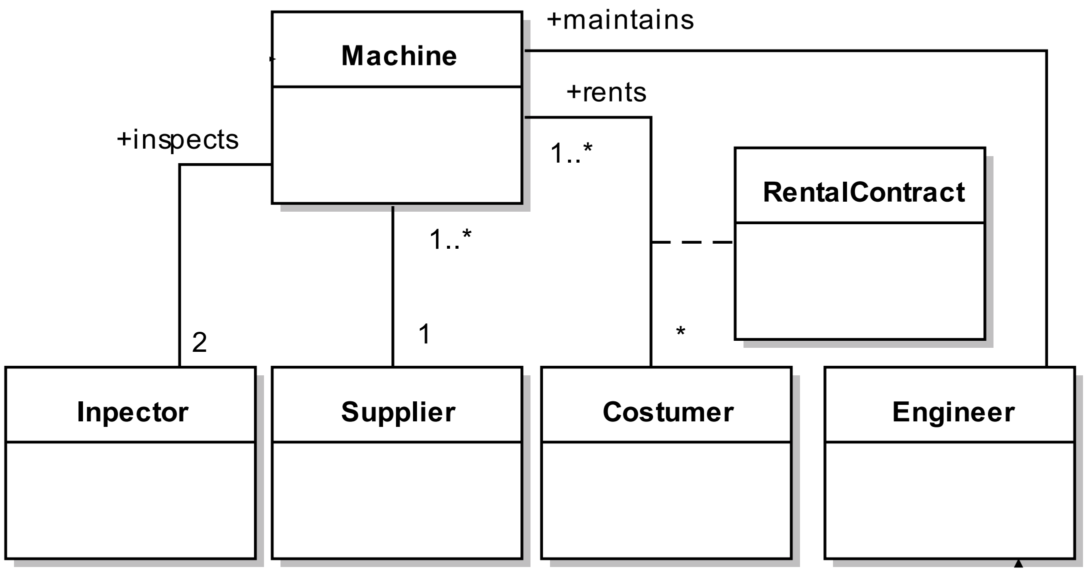
“Renting Machines”

“RM - Renting Machines” is a company that rents machines to costumers. Costumers are know by RM only when they rent the first machine; costumers can rent more than one machine. Rents are made according a contract, defined for each case. Each machine is supplied by a specific supplier, who can supplies more then one machine. **RM also has inspectors, and each machine must be assigned to two inspectors for regular inspection.** The machines also are maintained by engineers; for that role each engineer must have a license. Some of the machines are electrical, with one or more electrical engines. Each electrical engine can be moved from one machine to other, and can be repaired by electricians.



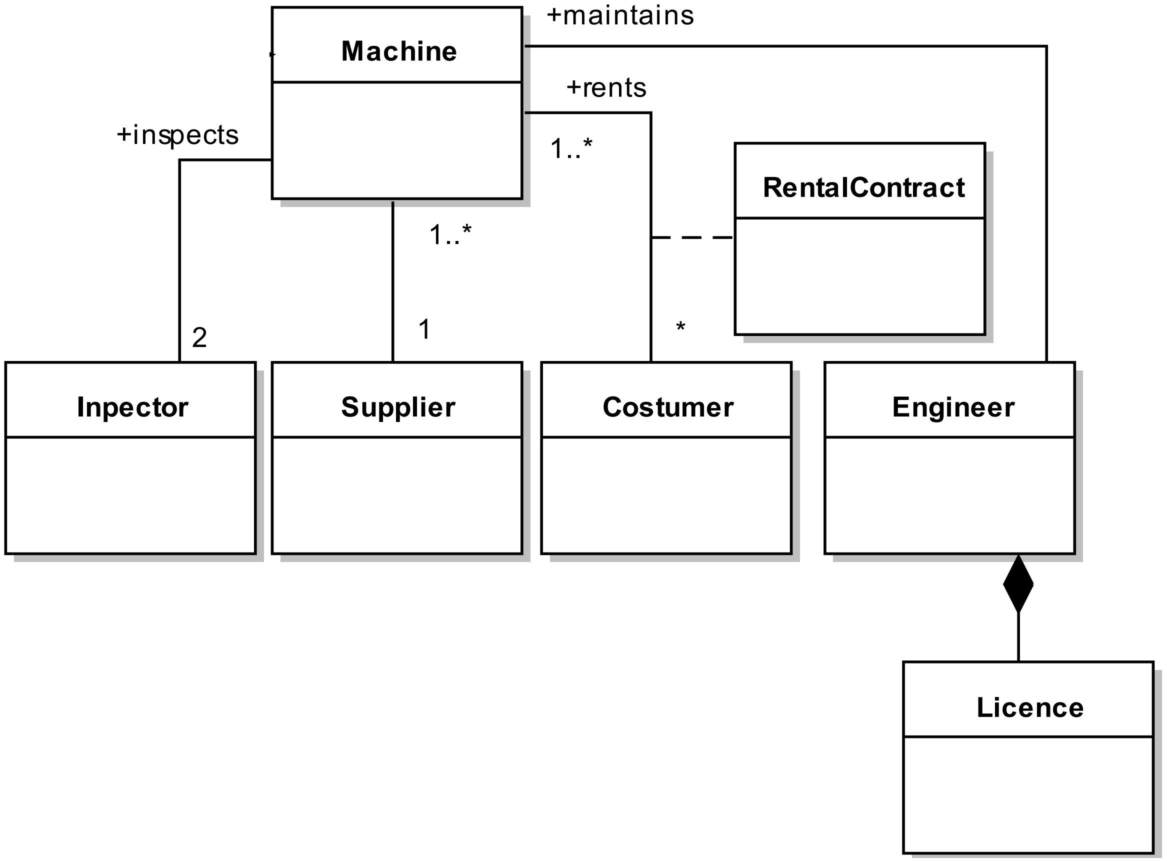
“Renting Machines”

“RM - Renting Machines” is a company that rents machines to costumers. Costumers are know by RM only when they rent the first machine; costumers can rent more than one machine. Rents are made according a contract, defined for each case. Each machine is supplied by a specific supplier, who can supplies more then one machine. RM also has inspectors, and each machine must be assigned to two inspectors for regular inspection. **The machines are maintained by engineers** for that role each engineer must have a license. Some of the machines are electrical, with one or more electrical engines. Each electrical engine can be moved from one machine to other, and can be repaired by electricians.



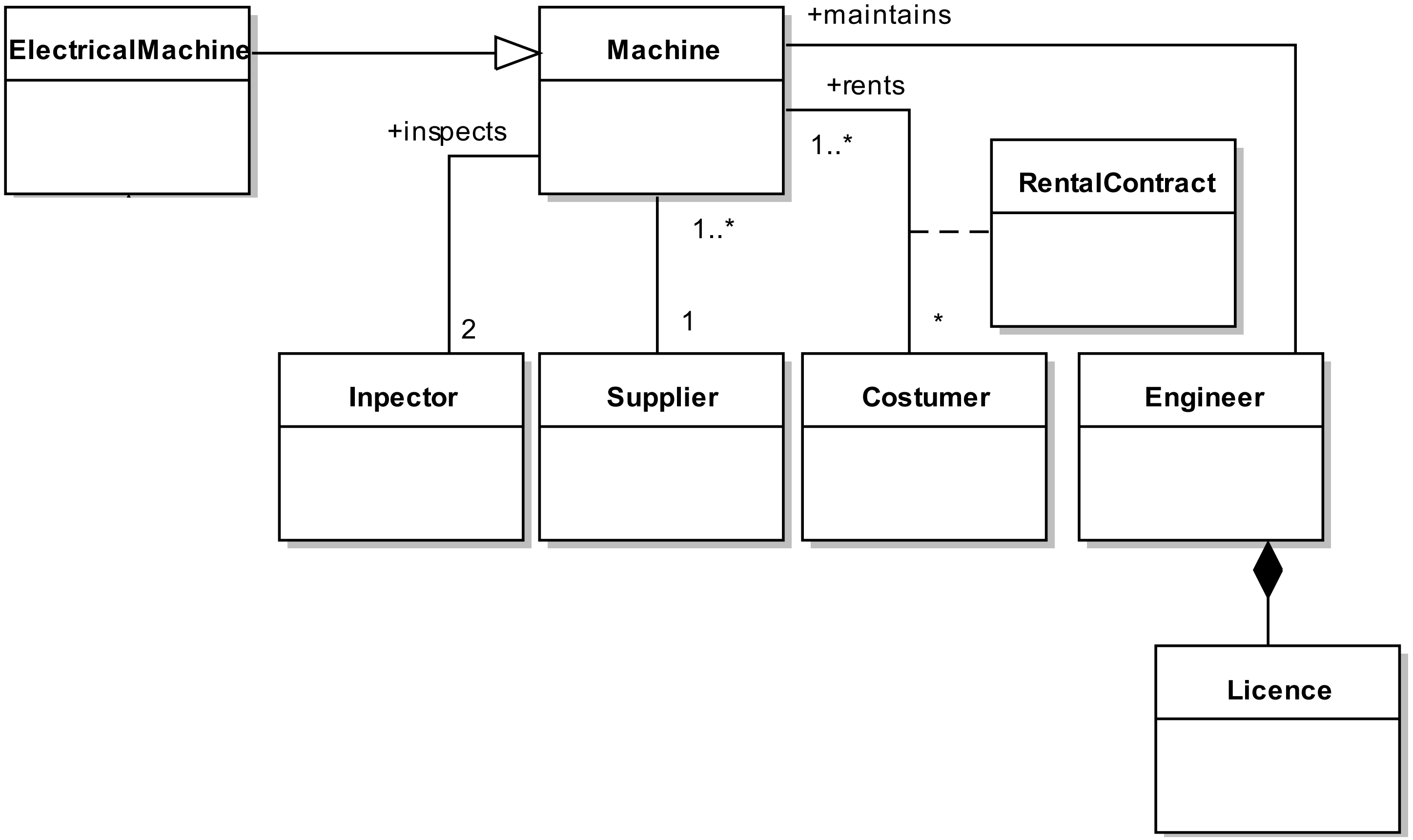
“Renting Machines”

“RM - Renting Machines” is a company that rents machines to costumers. Costumers are know by RM only when they rent the first machine; costumers can rent more than one machine. Rents are made according a contract, defined for each case. Each machine is supplied by a specific supplier, who can supplies more then one machine. RM also has inspectors, and each machine must be assigned to two inspectors for regular inspection. The machines also are maintained by engineers; **for that role each engineer must have a license**. Some of the machines are electrical, with one or more electrical engines. Each electrical engine can be moved from one machine to other, and can be repaired by electricians.



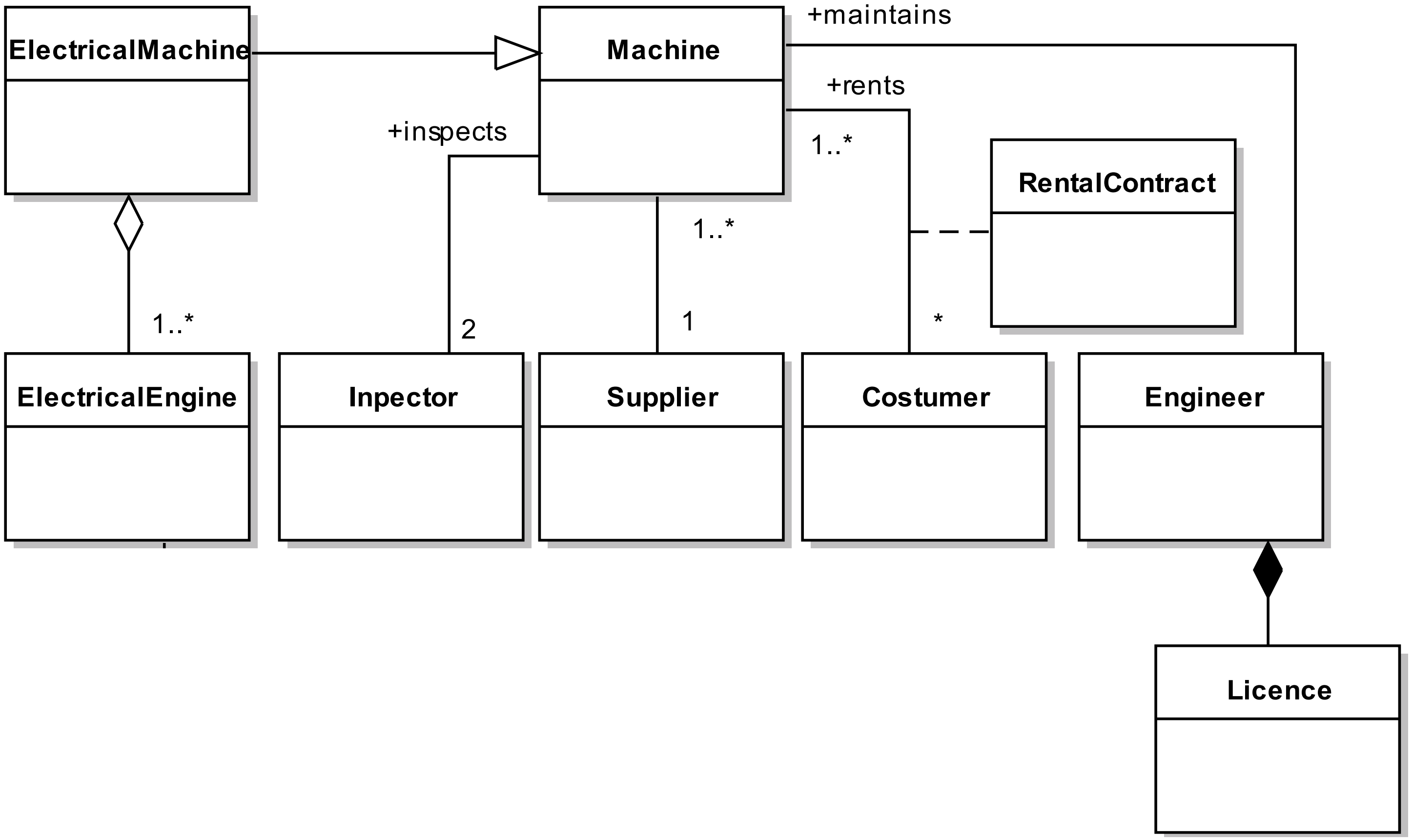
“Renting Machines”

“RM - Renting Machines” is a company that rents machines to costumers. Costumers are know by RM only when they rent the first machine; costumers can rent more than one machine. Rents are made according a contract, defined for each case. Each machine is supplied by a specific supplier, who can supplies more then one machine. RM also has inspectors, and each machine must be assigned to two inspectors for regular inspection. The machines also are maintained by engineers; for that role each engineer must have a license. **Some of the machines are electrically operated** with one or more electrical engines. Each electrical engine can be moved from one machine to other, and can be repaired by electricians.



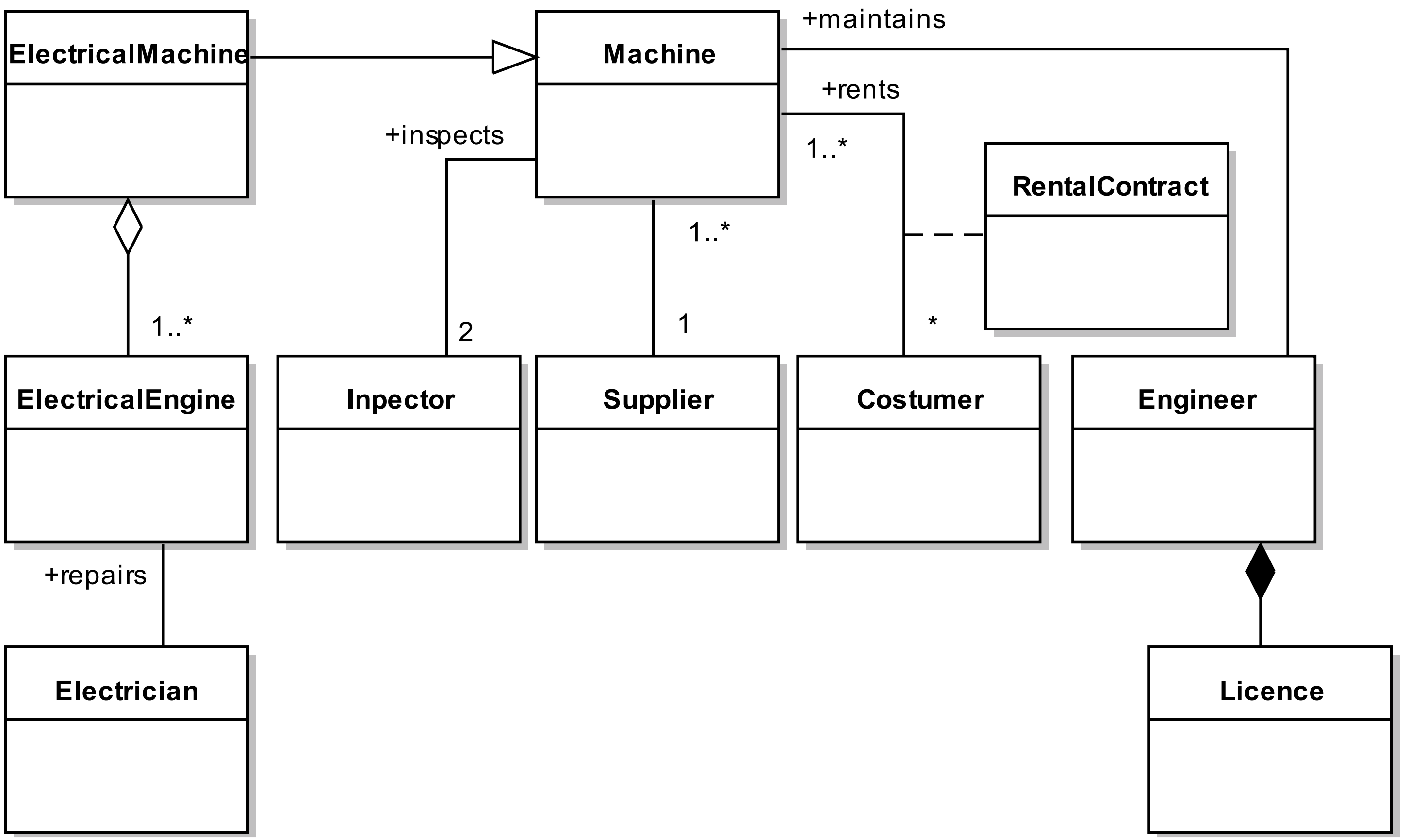
“Renting Machines”

“RM - Renting Machines” is a company that rents machines to costumers. Costumers are know by RM only when they rent the first machine; costumers can rent more than one machine. Rents are made according a contract, defined for each case. Each machine is supplied by a specific supplier, who can supplies more then one machine. RM also has inspectors, and each machine must be assigned to two inspectors for regular inspection. The machines also are maintained by engineers; for that role each engineer must have a license. Some of the machines are electrical, **by one or more electrical engines. An electrical engine can be moved from one machine to other**, and can be repaired by electricians.

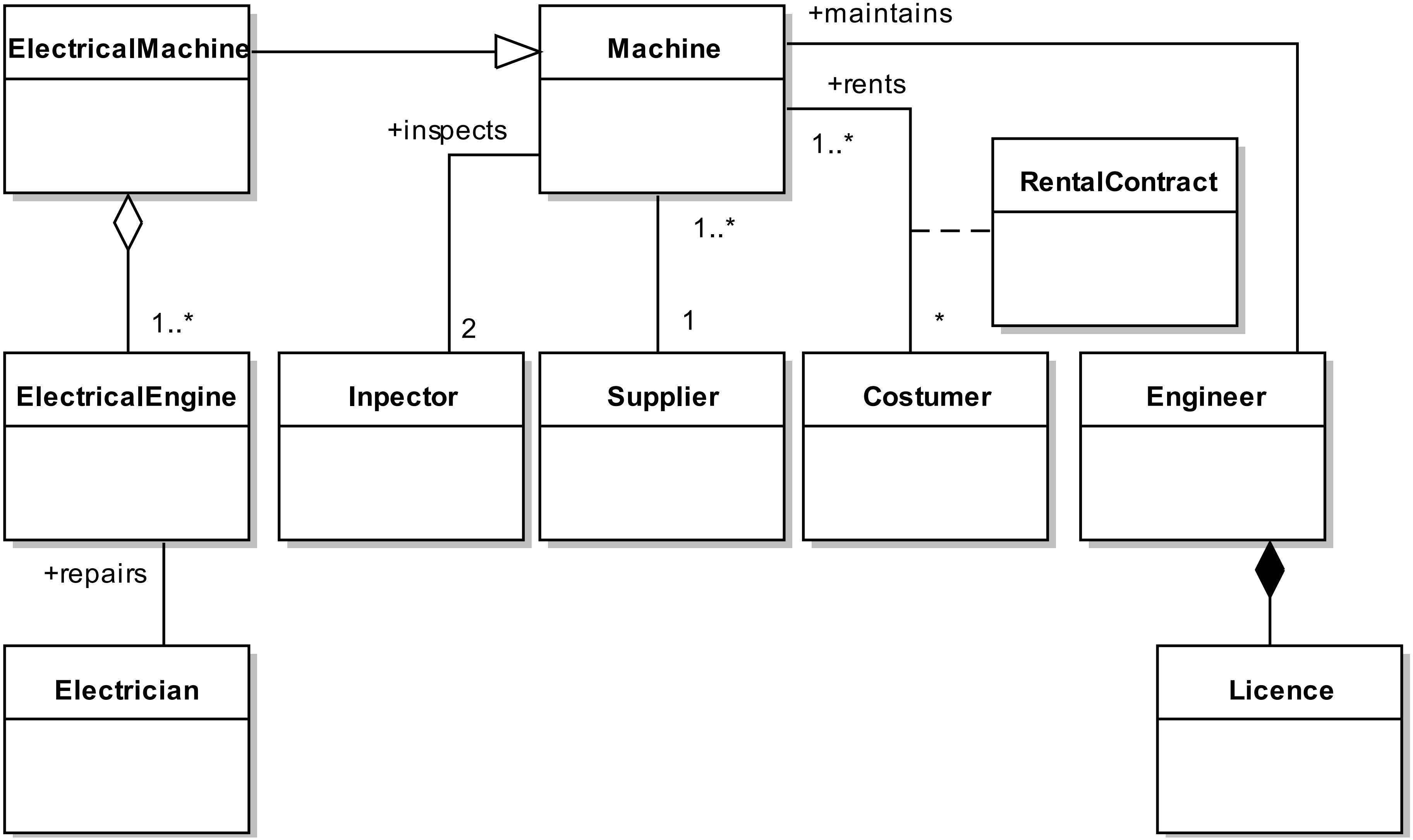


“Renting Machines”

“RM - Renting Machines” is a company that rents machines to costumers. Costumers are know by RM only when they rent the first machine; costumers can rent more than one machine. Rents are made according a contract, defined for each case. Each machine is supplied by a specific supplier, who can supplies more then one machine. RM also has inspectors, and each machine must be assigned to two inspectors for regular inspection. The machines also are maintained by engineers; for that role each engineer must have a license. Some of the machines are electrical, with one or more electrical engines. Each electrical engine can be moved from one machine to other, **and can be repaired by electricians.**



“Renting Machines”



UML
-
Instances / Objects

http://smellyfunny.blogspot.com/2008/08/funny-objects-with-faces.html

Instances

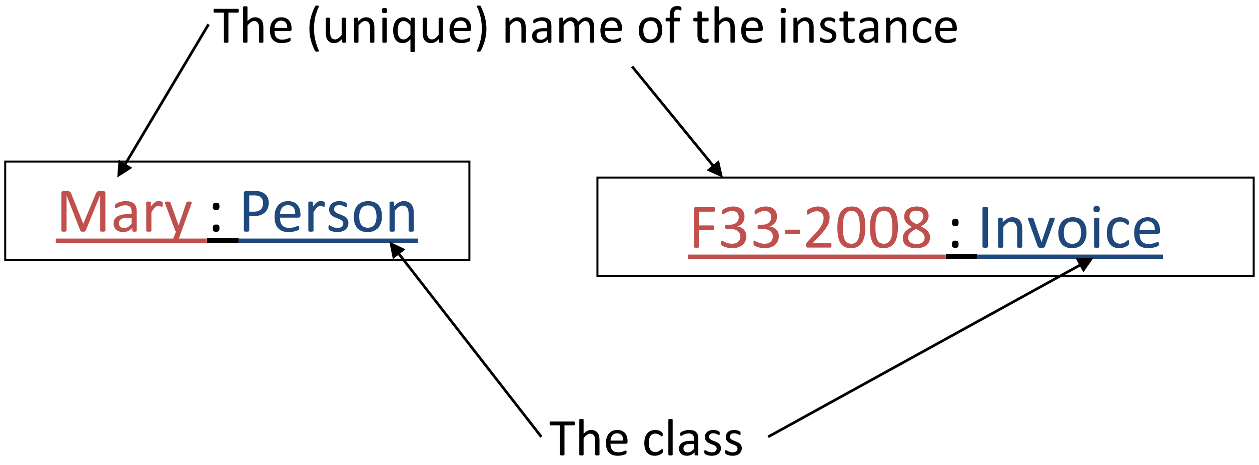
- An **instance** is a concrete manifestation of a concept.
- Instances have a **state**, which can be changed by operations.

- Some examples:
  - An instance of a **class** is an **object** (UML)
  - An instance of an **association** is a **link** (UML/SysML)
  - An instance of a **use case** is a **scenario** (UML/SysML)

Objects (UML)

- An object is an instance of a class
- Instances are unique

name-of-the-object : name-of-the-class



The (unique) name of the instance

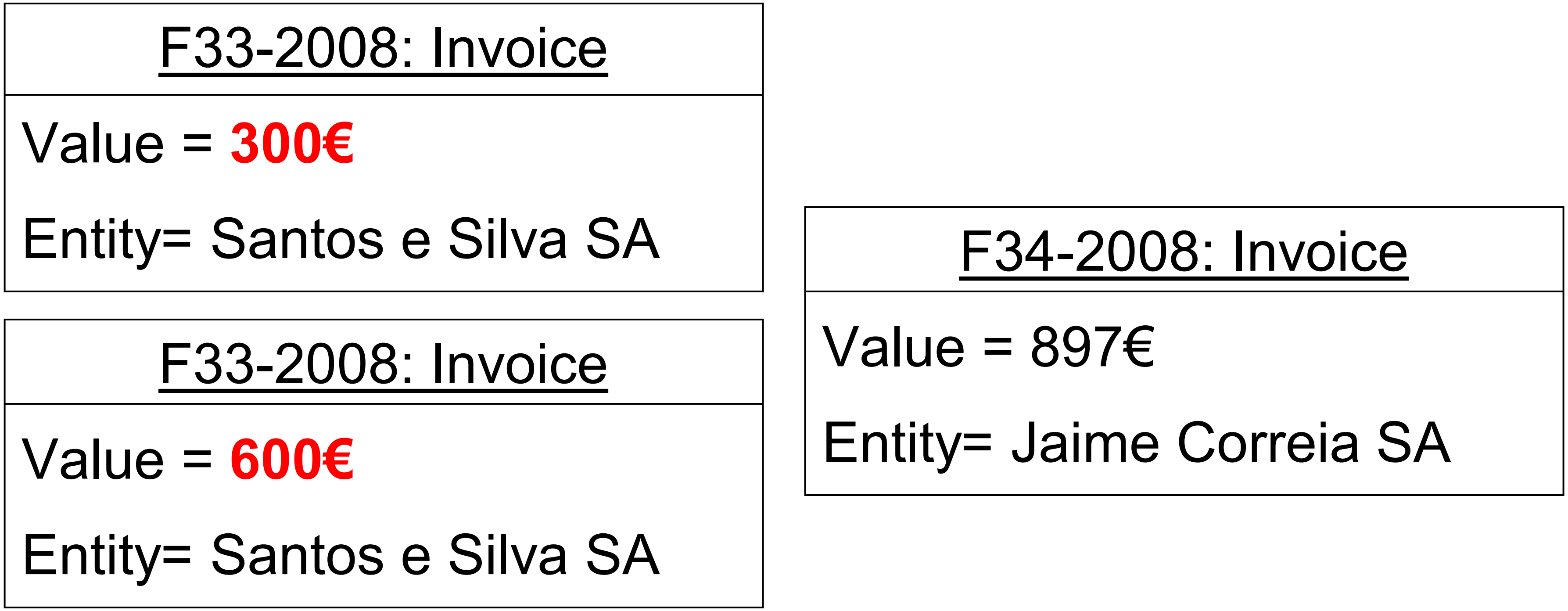
Mary : Person

F33-2008 : Invoice

The class

Objects (UML)

- The attributes of an object and their values in a specific moment **define the state of the object**.
- The **state** of an object can change along time while the object interacts with other objects.



F33-2008: Invoice

Value = **300€**

Entity= Santos e Silva SA

F33-2008: Invoice

Value = **600€**

Entity= Santos e Silva SA

F34-2008: Invoice

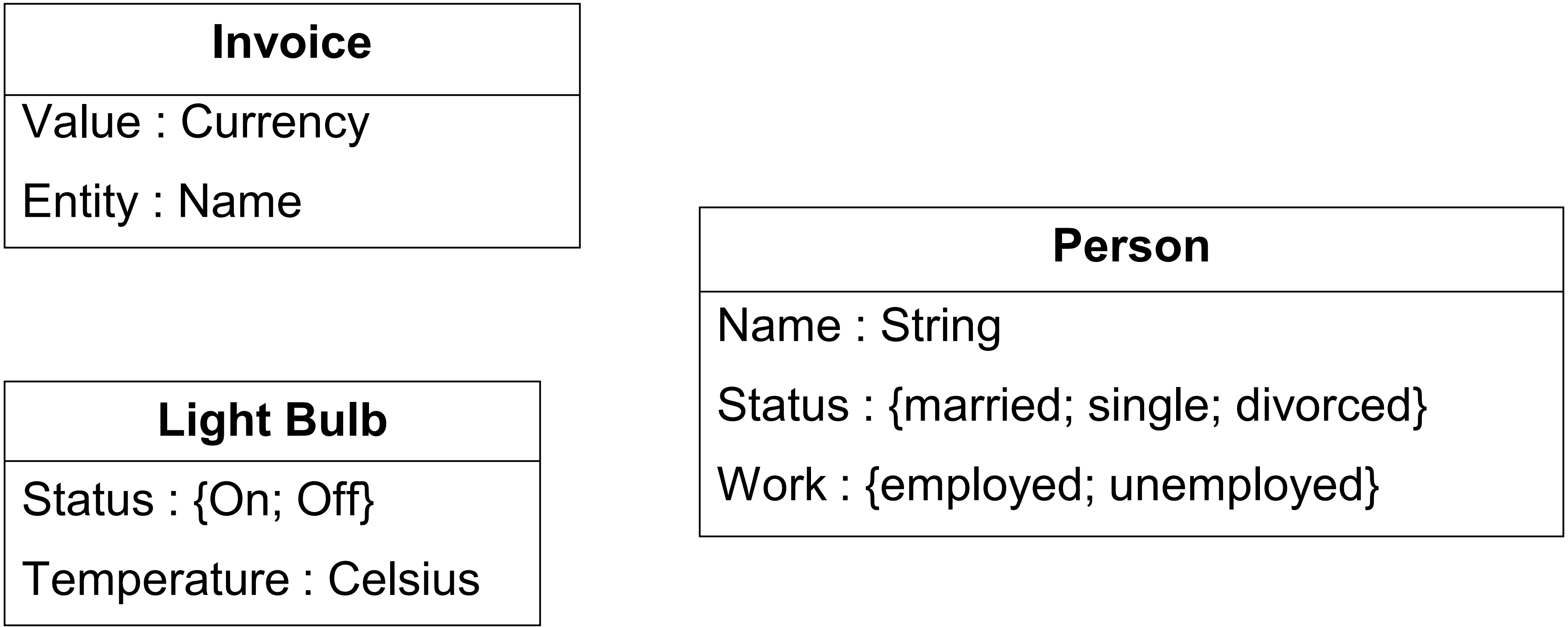
Value = 897€

Entity= Jaime Correia SA

Objects and states (UML)

Discussion

- What are the possible states of objects of these classes?
- How to define a state?



Invoice

Value : Currency

Entity : Name

Person

Name : String

Status : {married; single; divorced}

Work : {employed; unemployed}

Light Bulb

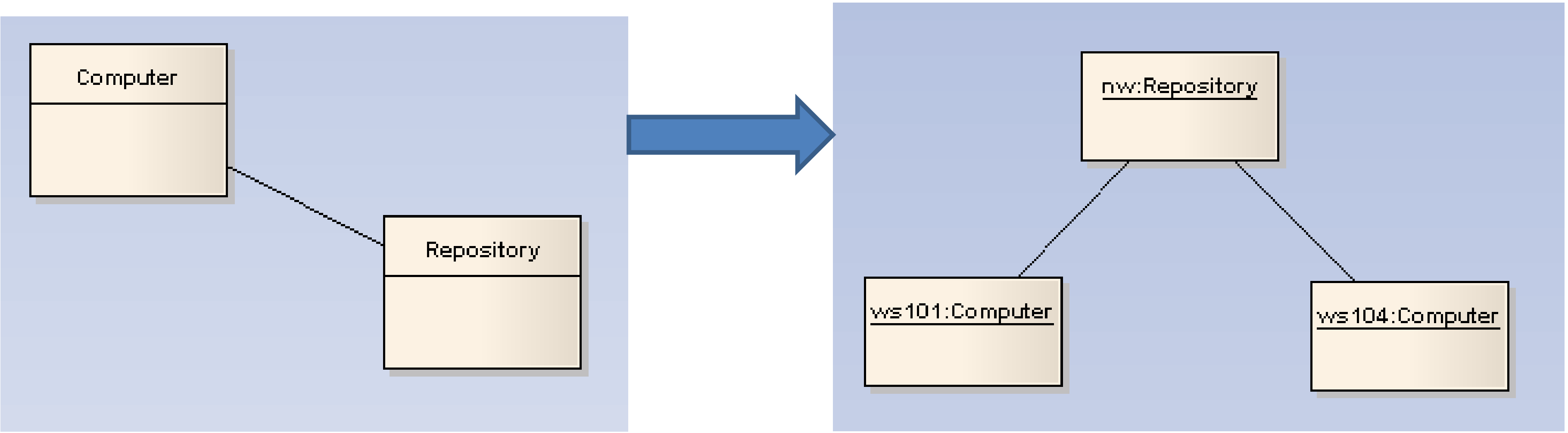
Status : {On; Off}

Temperature : Celsius

Object Diagram (UML)

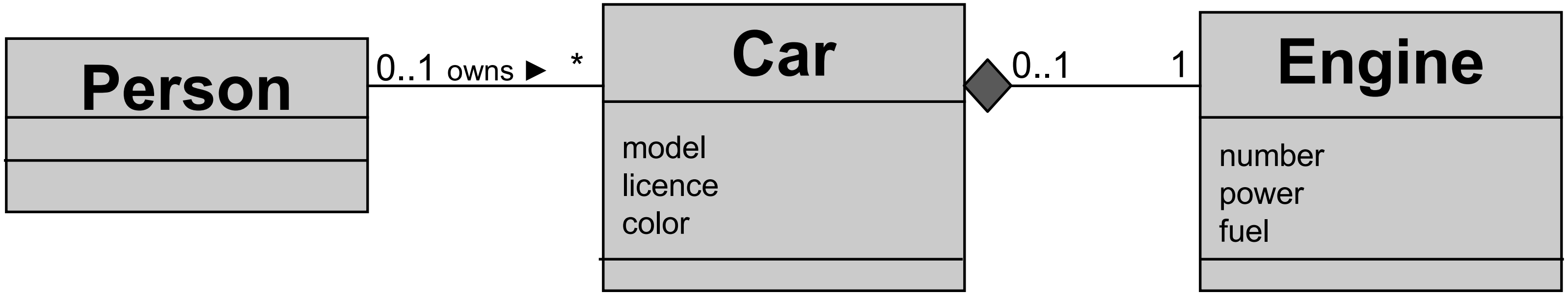
An **Object Diagram** shows **instances of Classes** and their **relationships** at a given point in time.

Usually they represent only parts of the system and are useful in understanding a complex Class diagram by describing different scenarios in which the relationships apply to.



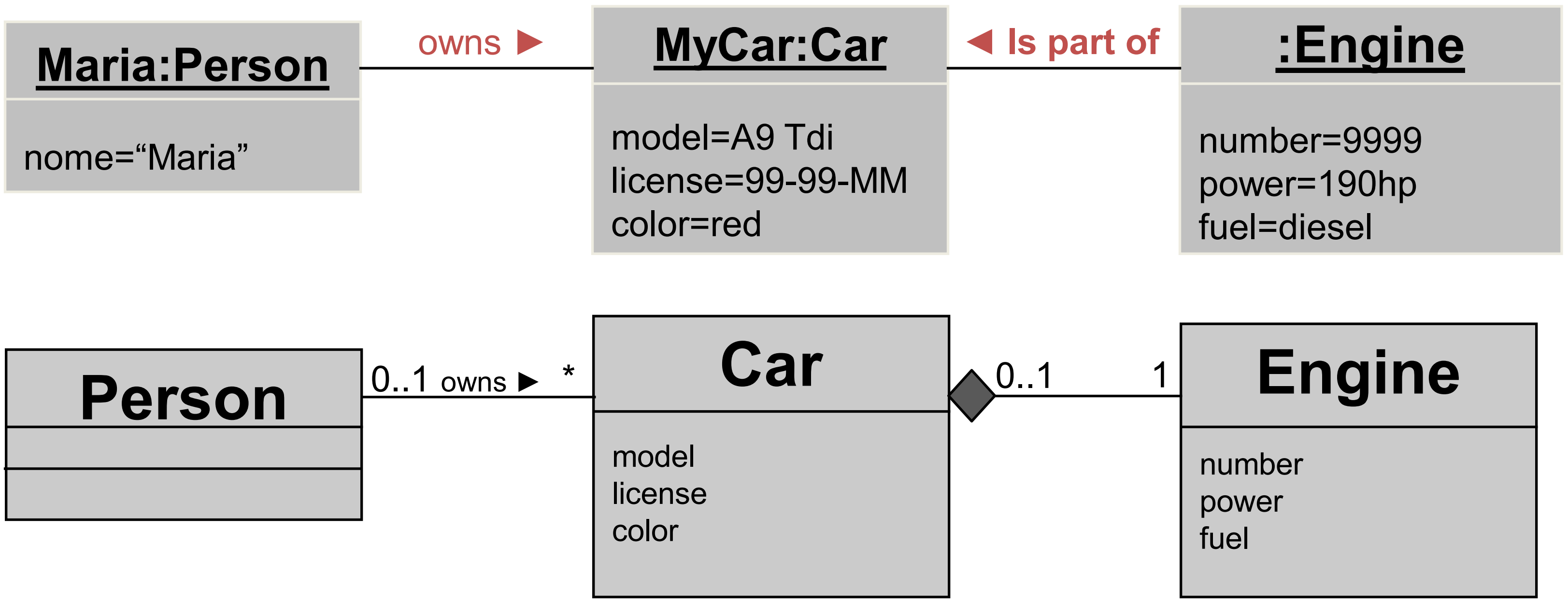
UML: Object Diagram

- One Person may *own any number of* Cars.
- A Car *is owned by at most one* Person.
- A Car *has one* Engine.
- An Engine may be *part of at most one* Car.
- An Engine cannot be shared by Cars (composition).



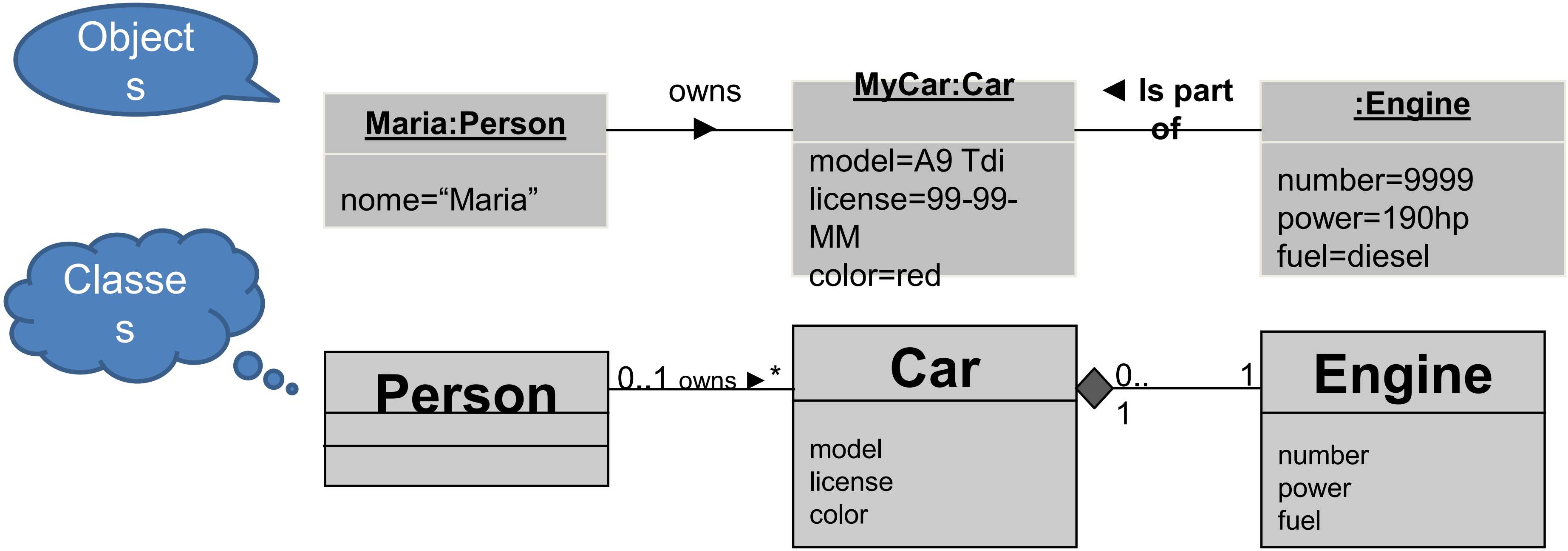
UML: Object Diagram

Maria has a red car named MyCar, model A9 Tdi, with a 190hp diesel engine with serial number 9999. The licence plate of the car is 99-99-MM.



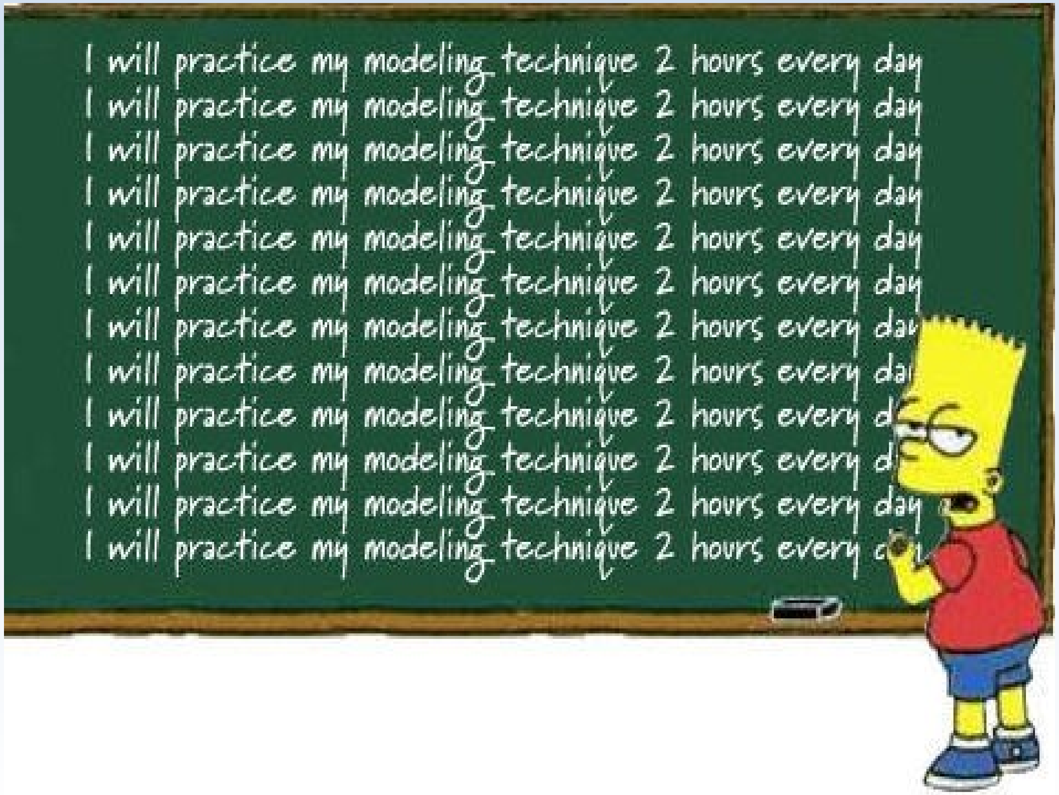
Instances in UML: Objects

- In UML an **object** is always an instance of a **class**
- name-of-the-object : name-of-the-class

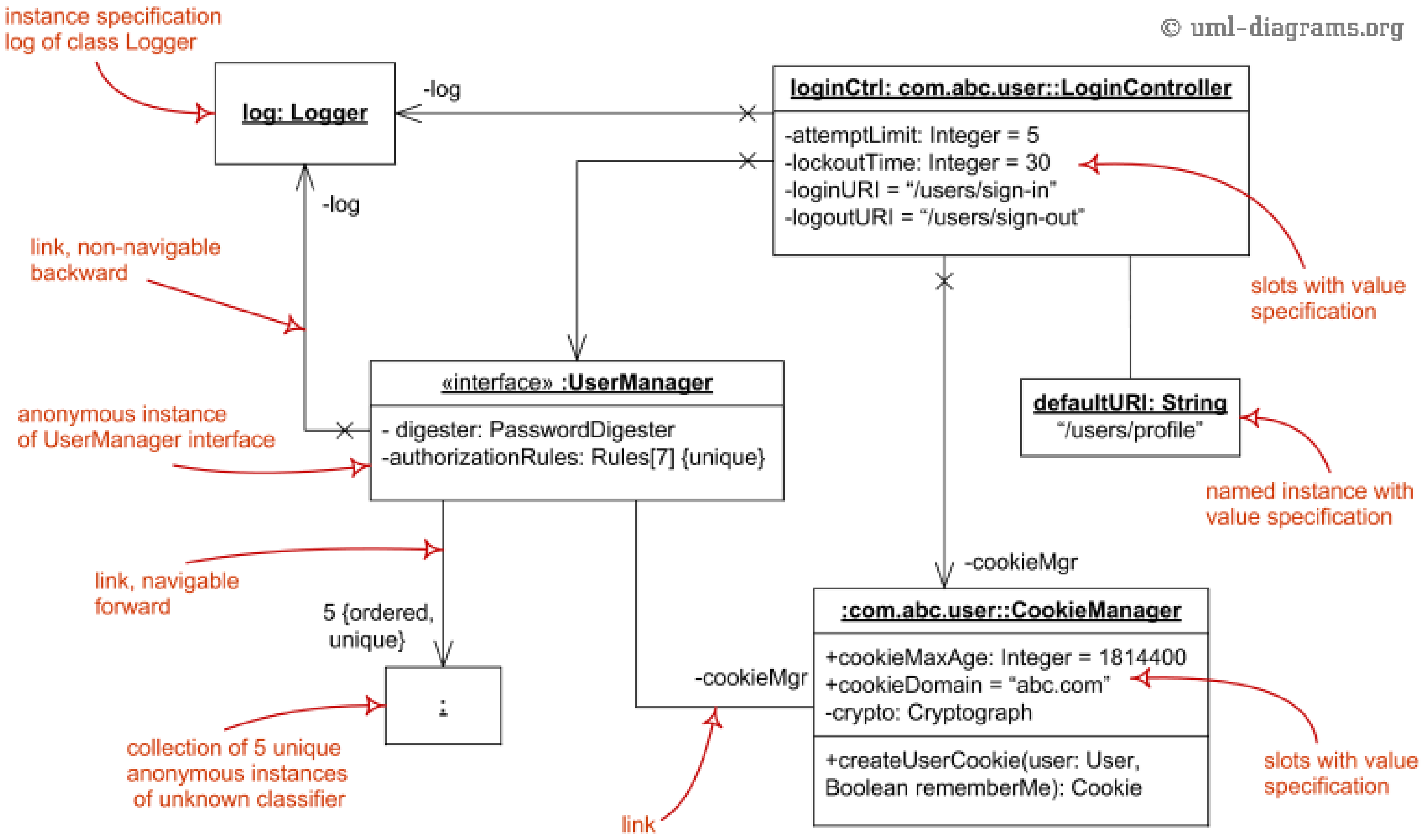


Structural Modelling with UML

exercise…


I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day

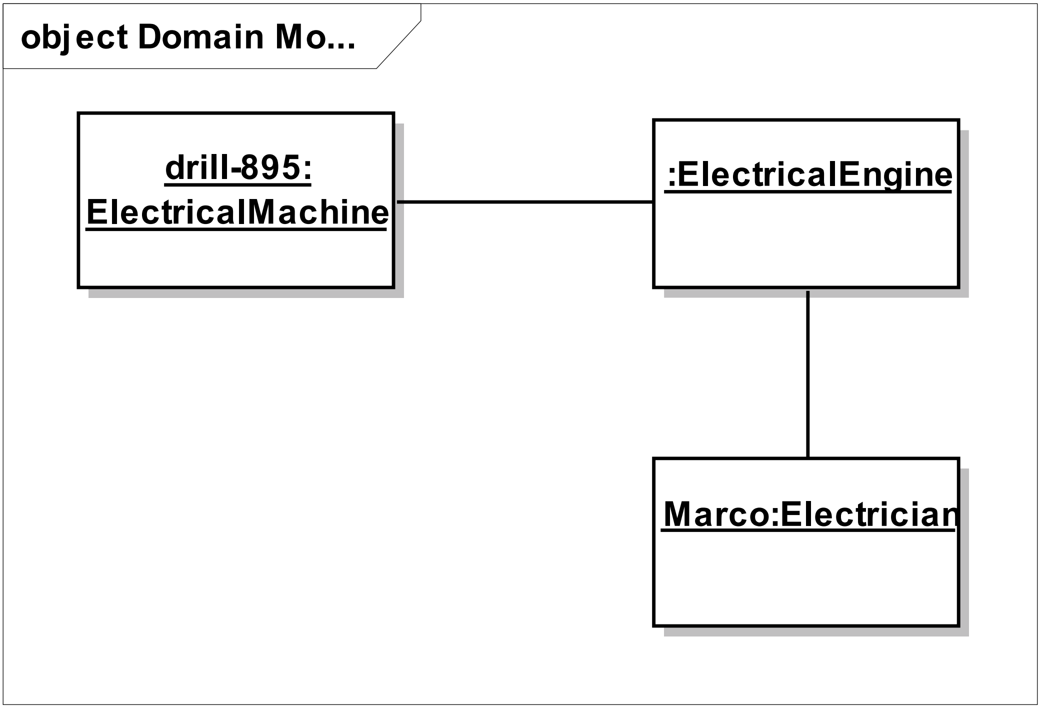
Object Diagrams - Summary

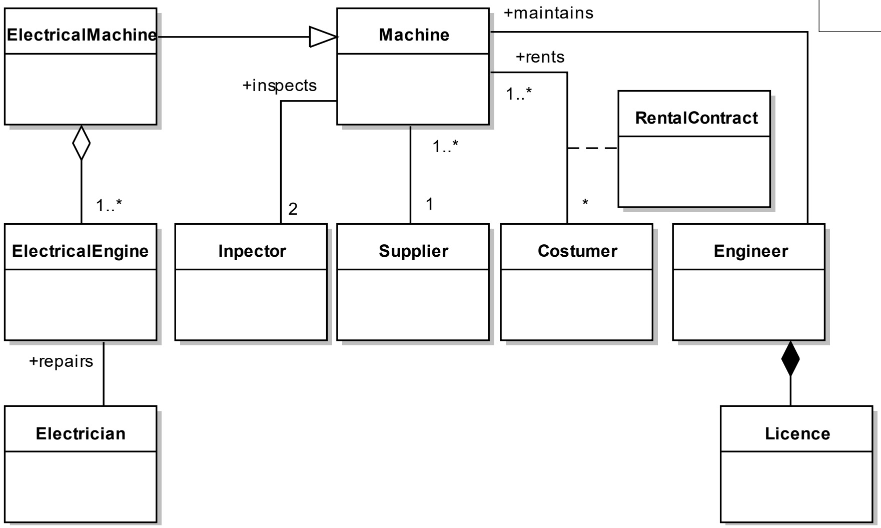


https://www.uml-diagrams.org/class-diagrams-overview.html

“Renting Machines” – Domain examples

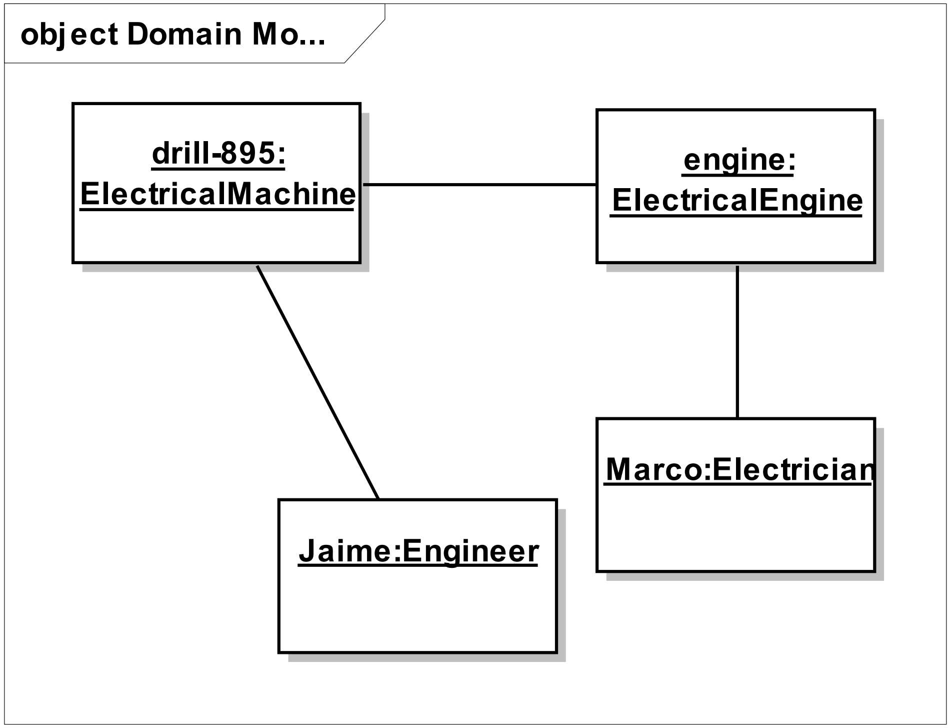
- Marco is the electrician that repairs the electrical engine of the drill 895.

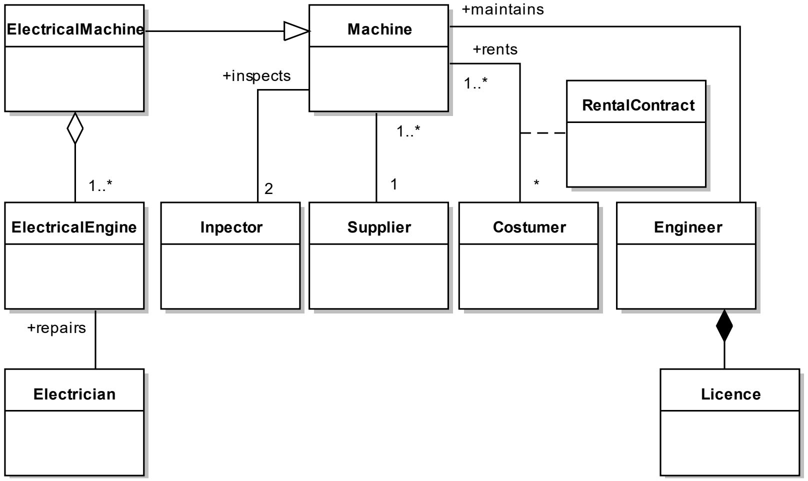




“Renting Machines” – Domain examples

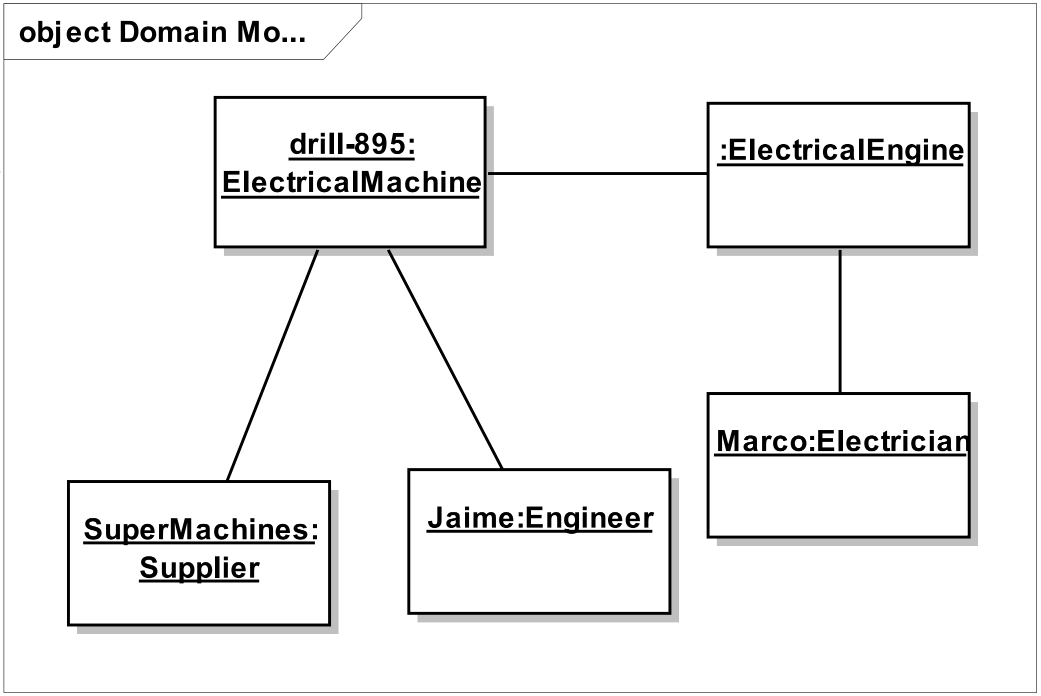
- Marco is the electrician that repairs the electrical engine of the drill 895
- **Jaime is the engineer that maintains the machine drill 895**

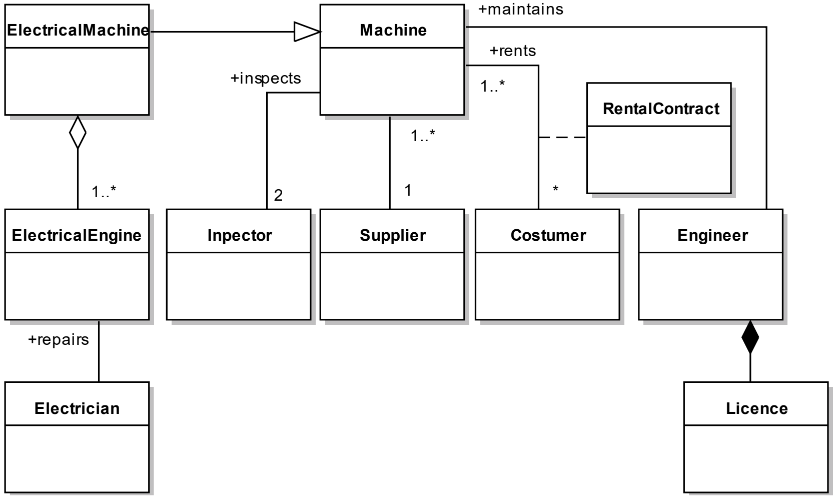




“Renting Machines” – Domain examples

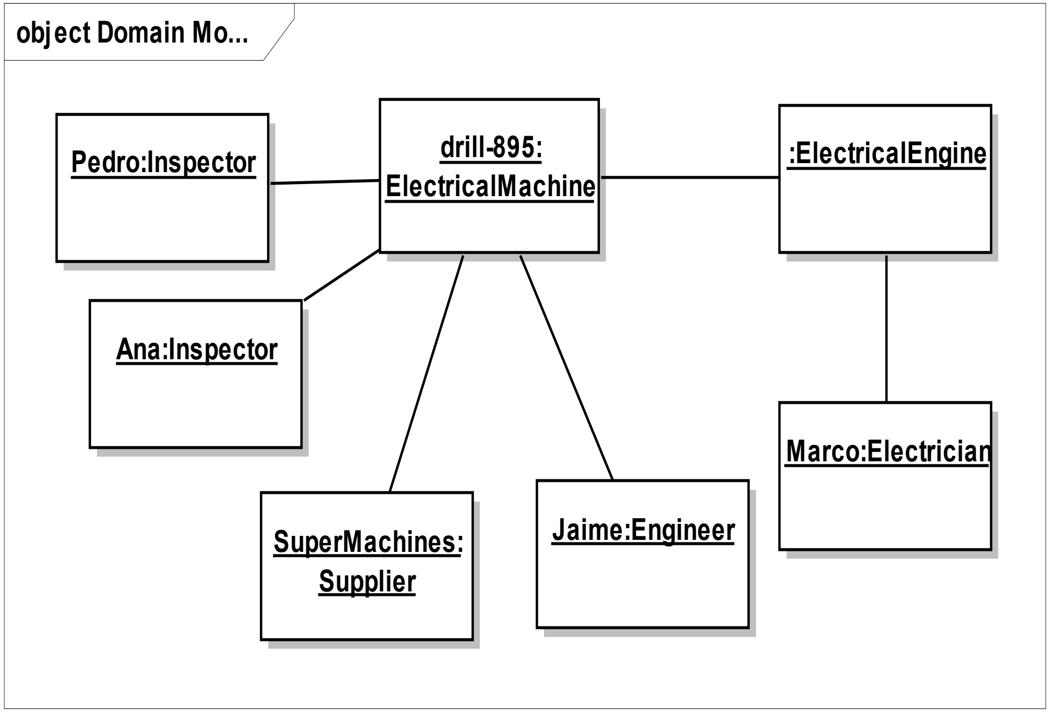
- Marco is the electrician that repairs the electrical engine of the drill 895
- Jaime is the engineer that maintains the machine drill 895
- **The machine drill 895 was supplied by Super Machines**

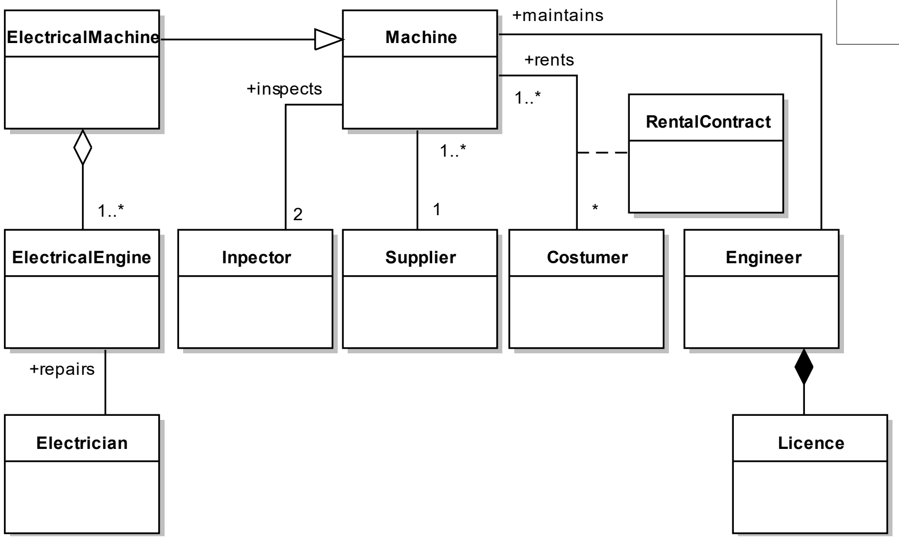




“Renting Machines” – Domain examples

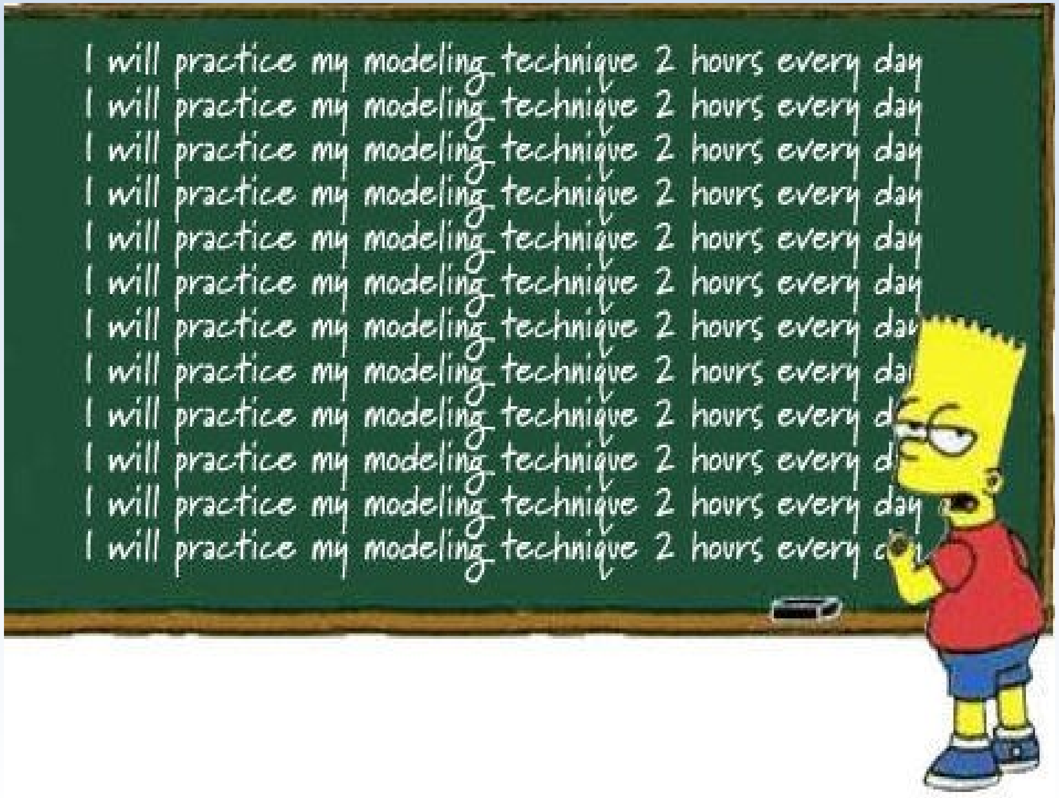
- Marco is the electrician that repairs the electrical engine of the drill 895
- Jaime is the engineer that maintains the machine drill 895
- The machine drill 895 was supplied by Super Machines
- **Pedro and Ana are the inspectors of the machile drill 895**





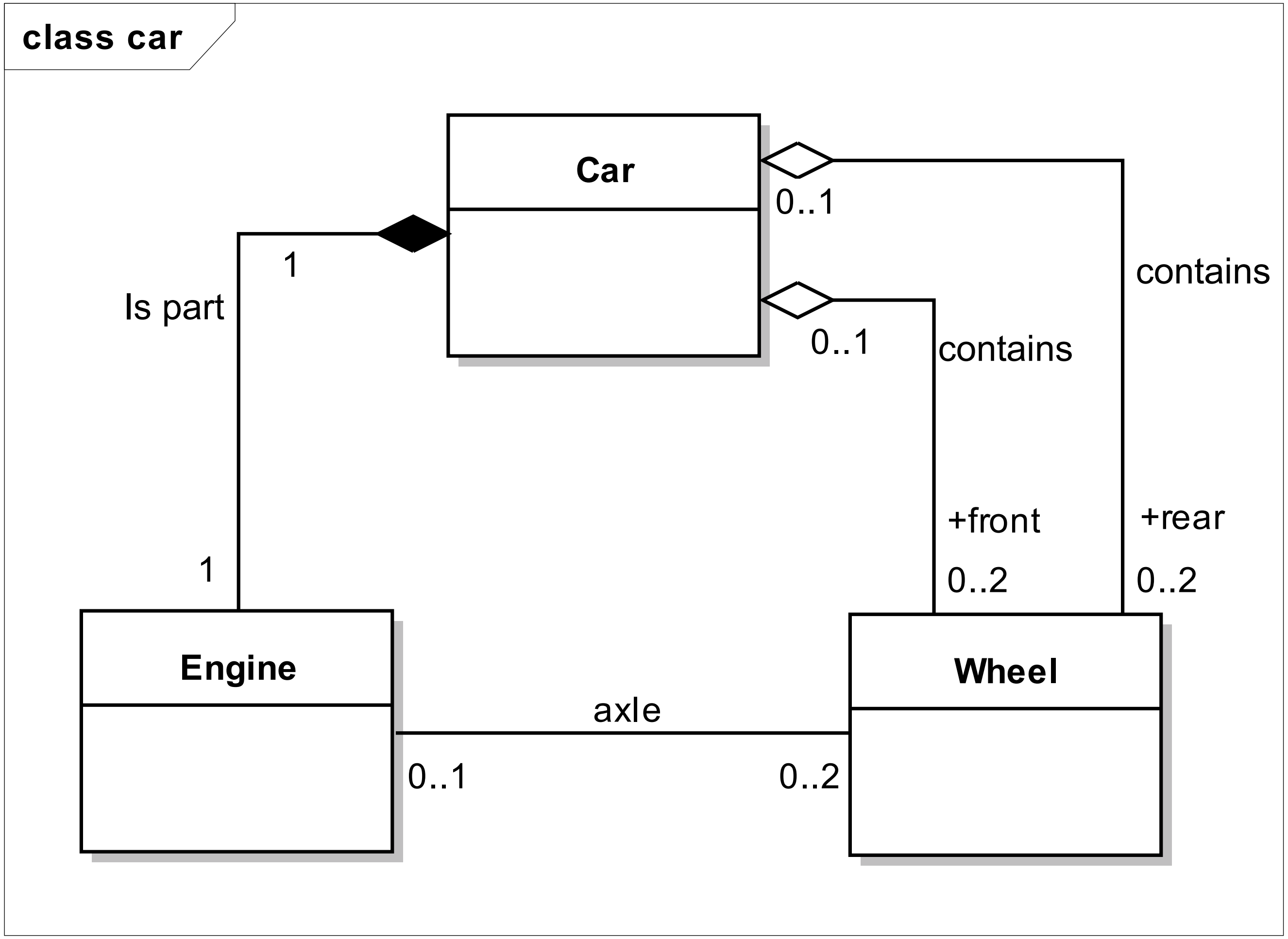
Structural Modelling with UML

A simple exercise (but with some tricks…)


I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every da
I will practice my modeling technique 2 hours every da
I will practice my modeling technique 2 hours every d
I will practice my modeling technique 2 hours every d
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every d

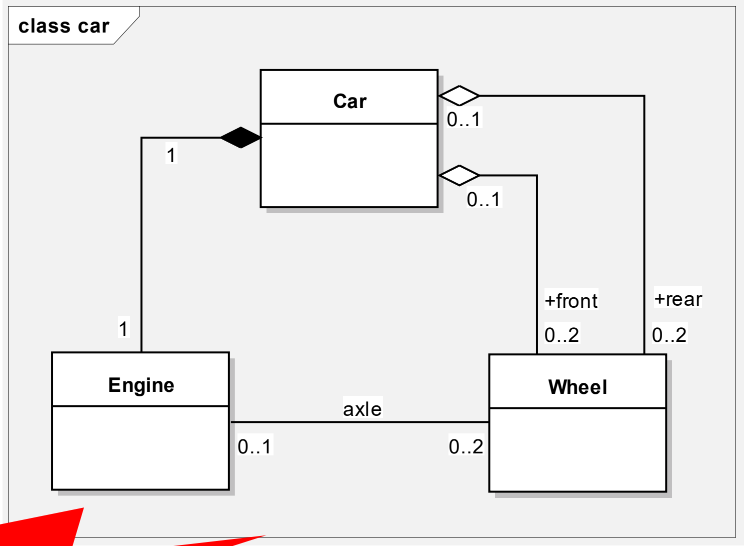
What is wrong with this domain?

A car must have an engine and can have up to two wheels in front and up to two wheels at rear. The engine is connected to up to two wheels by an axle.

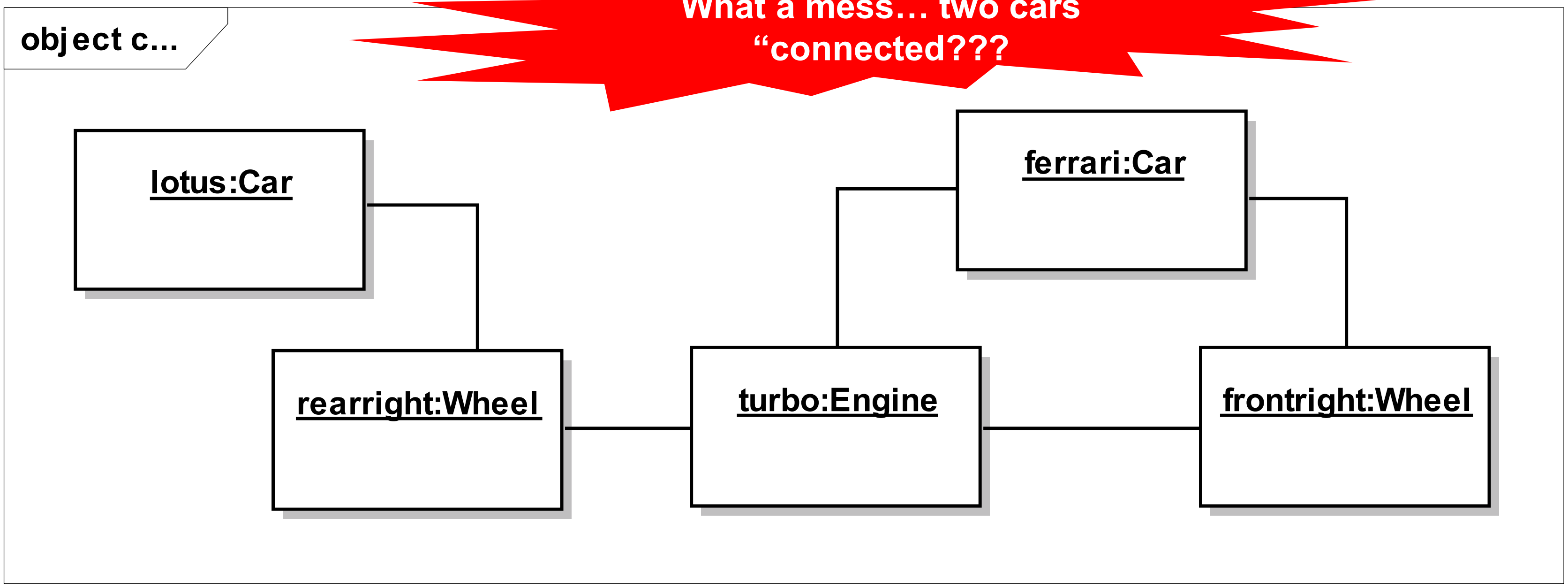


What is wrong with this domain?

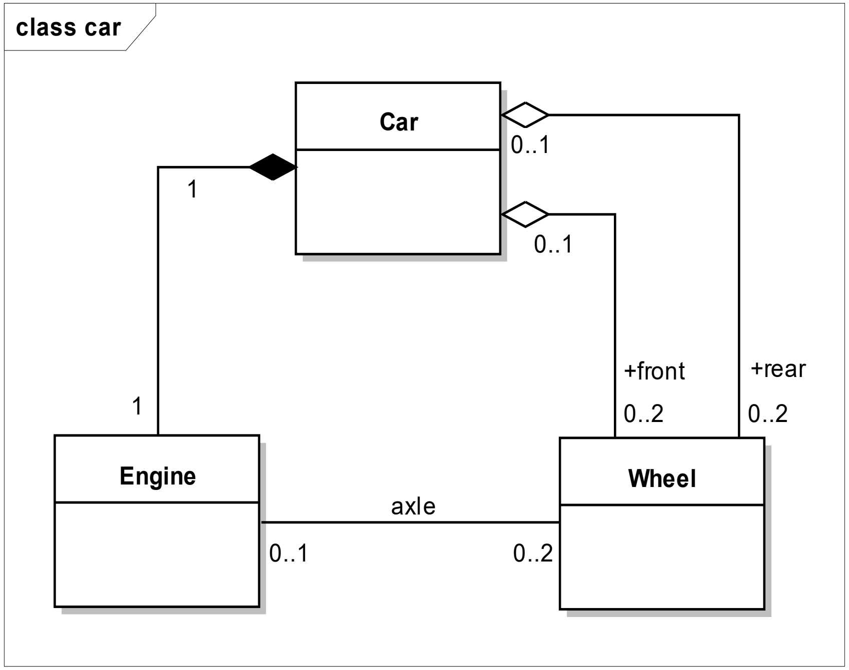
A car must have an engine and can have up to two wheels in front and up to two wheels at rear. The engine is connected to up to two wheels by an axle.



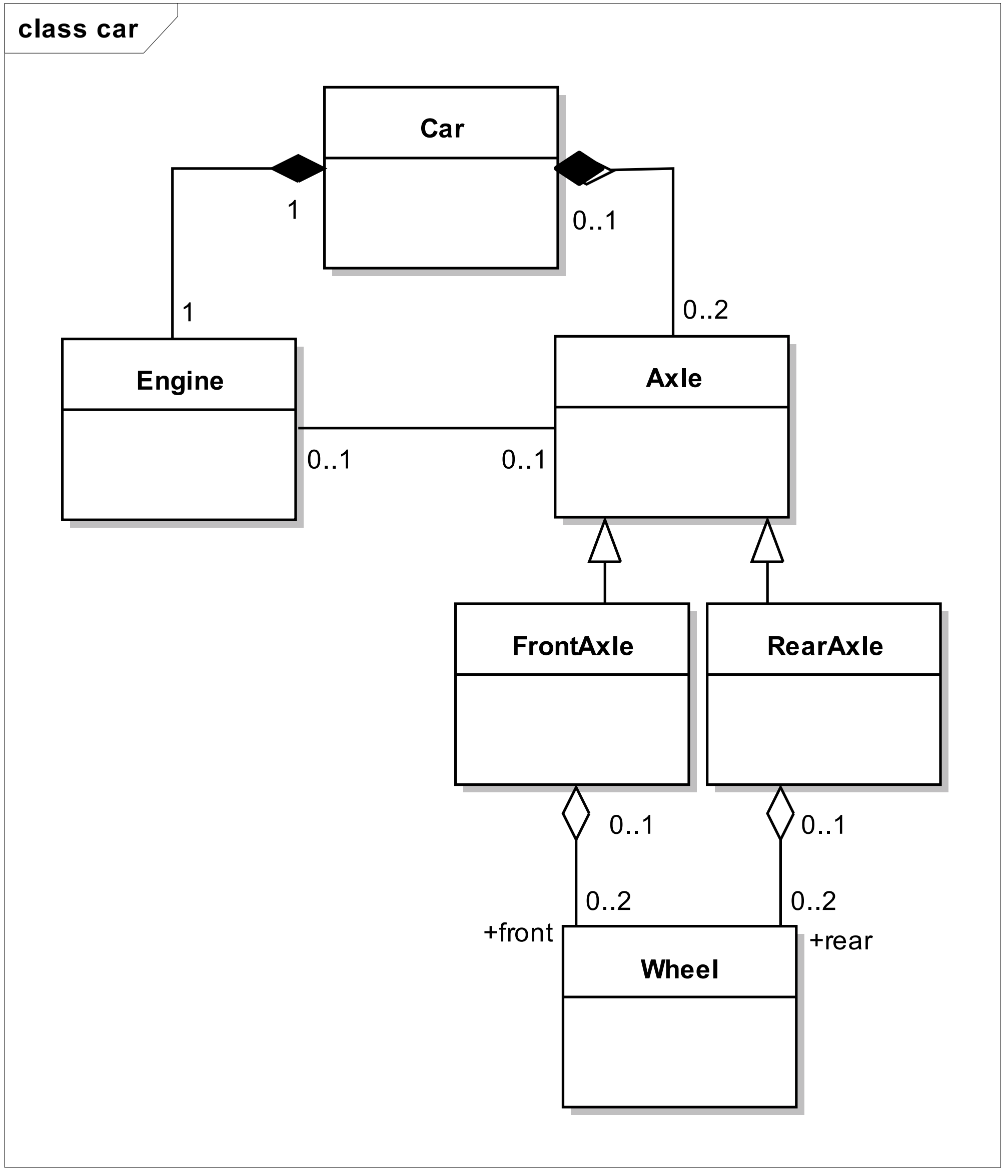
What a mess… two cars “connected???



"Axle" seems too relevant to be only an association, so it must be considered as a domain entity (OK now? Not yet…! A good step, but something stills wrong)

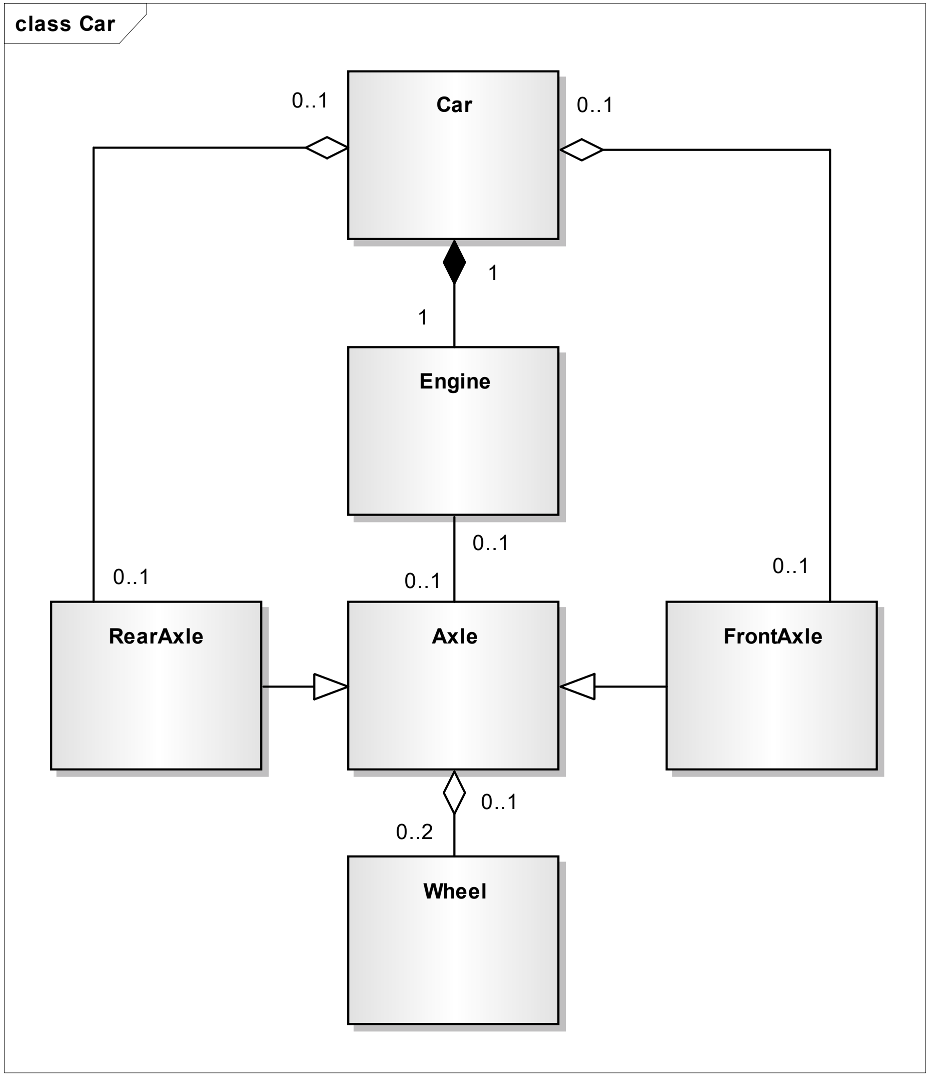


A car must have an engine and can have up to two wheels in front and up to two wheels at rear. The engine is connected to up to two wheels by an axle.



Much better now? Looks OK! Or not? ... discuss ...

A car must have an engine and can have up to two wheels in front and up to two wheels at rear. The engine is connected to up to two wheels by an axle.



IMPORTANT: Many more examples of this case could be developed… the CORRECT one is not possible to be designed without the expression of more precise requirements to enforce decisions

UML

Object Constraint Language (OCL)

http://www.omg.org/spec/OCL/

Object Constraint Language (OCL)

- OCL is a formal language that describes expressions on UML models.

- OCL expressions can be used to specify operations/actions.

- However, OCL expressions only tell “what” the system does, and not “how” it is done (that is supposed to be declared by the UML/SysML diagrams)

Object Constraint Language (OCL)

- OCL is not a programming language: It is not expected to write program logic or flow control in OCL.

- OCL is a pure specification language. When an OCL expression is evaluated, it simply returns a value. It cannot change anything in the model. This means that the state of the system will never change because of the evaluation of an OCL expression, even though an OCL expression can be used to specify a state change (e.g., in a post-condition).

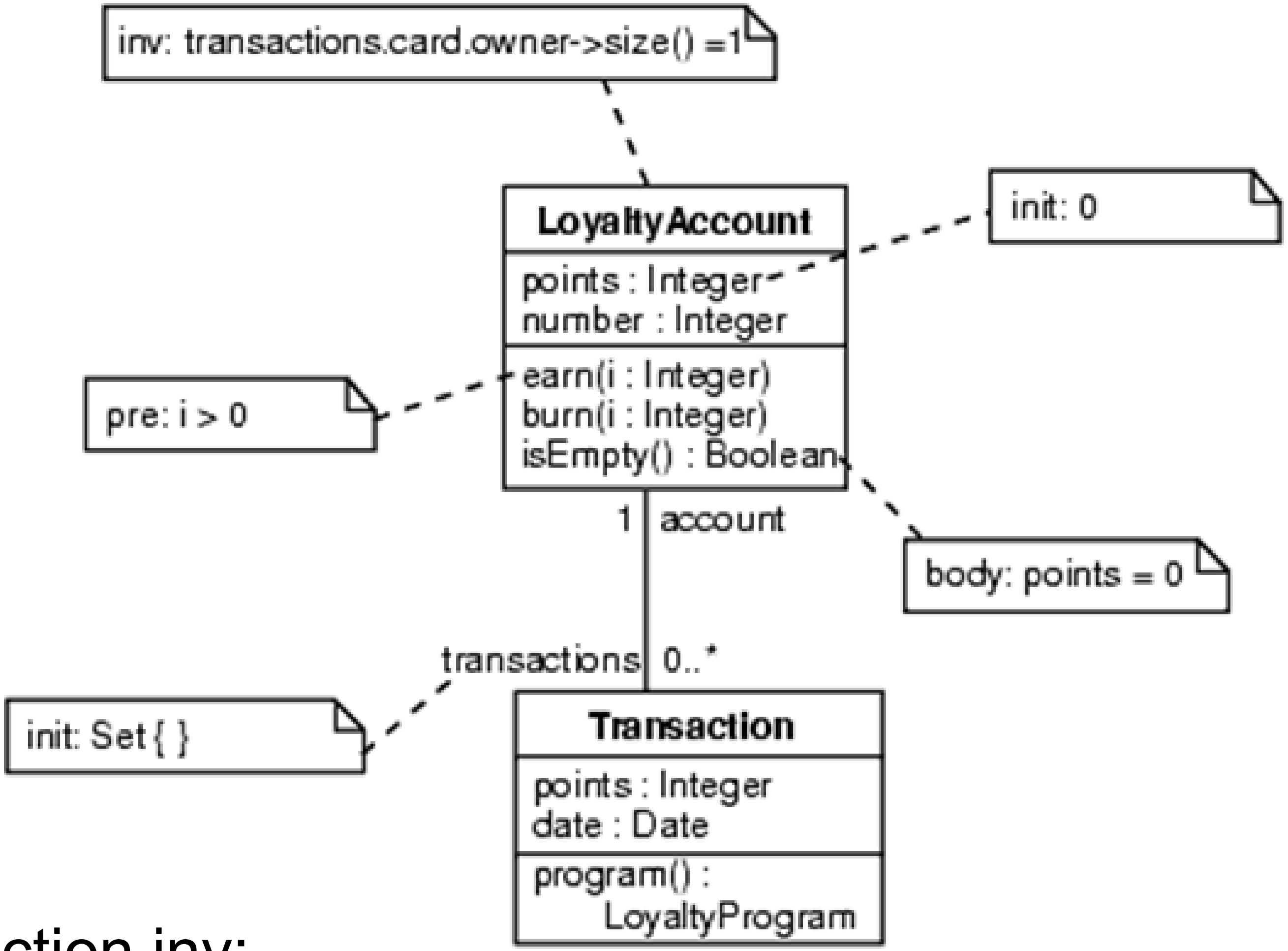
- OCL expressions can be used in any diagram

OCL

- Types of OCL expressions:
  - Declare the initial value of an attribute of expression
  - A rule for an attribute or expression
  - An instance, condition or value for a parameter in a behaviour diagram…
  - etc…

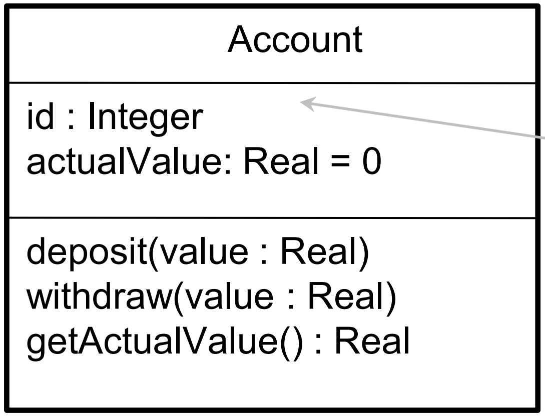
- Types of OCL constraints:
  - **Invariant** a permanent restriction concerning the system
  - **Precondition** a restriction that must be true in order to execute an action
  - **Post condition** a restriction that must be true at the end of the execution of an action
  - **Guard** a restriction that must be true to perform a state transition

Example of OCL with classes



context Transaction inv:
points >= 0

Examples of expressions and constraints



Account

id : Integer
actualValue: Real = 0

deposit(value : Real)
withdraw(value : Real)
getActualValue() : Real

Usual types:
Integer, Real, String, Boolean

context Account::withdraw (value : Real)
pre: value <= actualValue
post: actualValue = actualValue@pre – value

context Account::getActualValue() : Real
post: result = actualValue

Usual operators:
= < > <> <= >= + - * /
mod() div() max()
min() round() abs()
and or xor not implies
if_then_else_endif …

OCL expressions in State Machine Diagrams

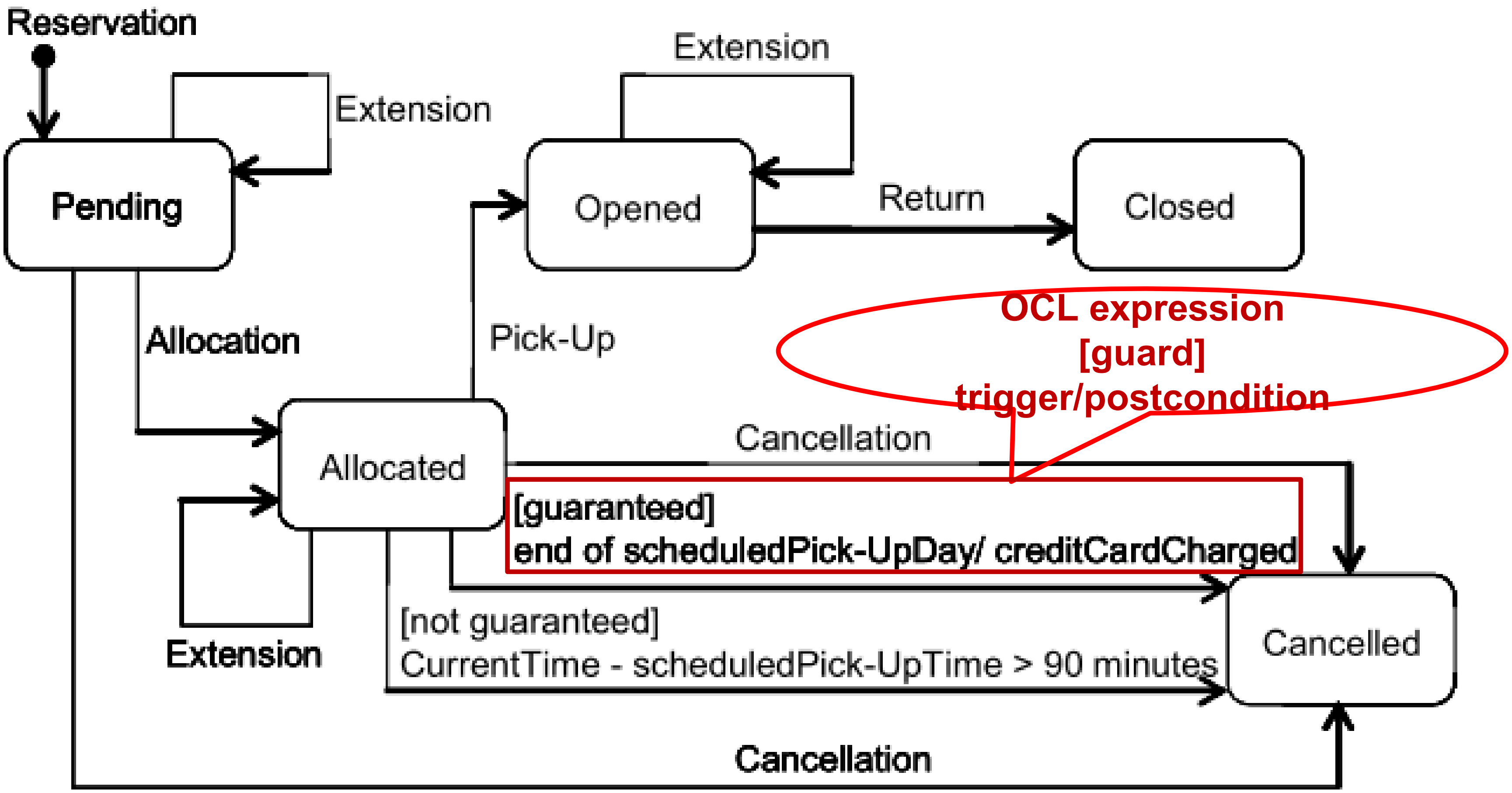
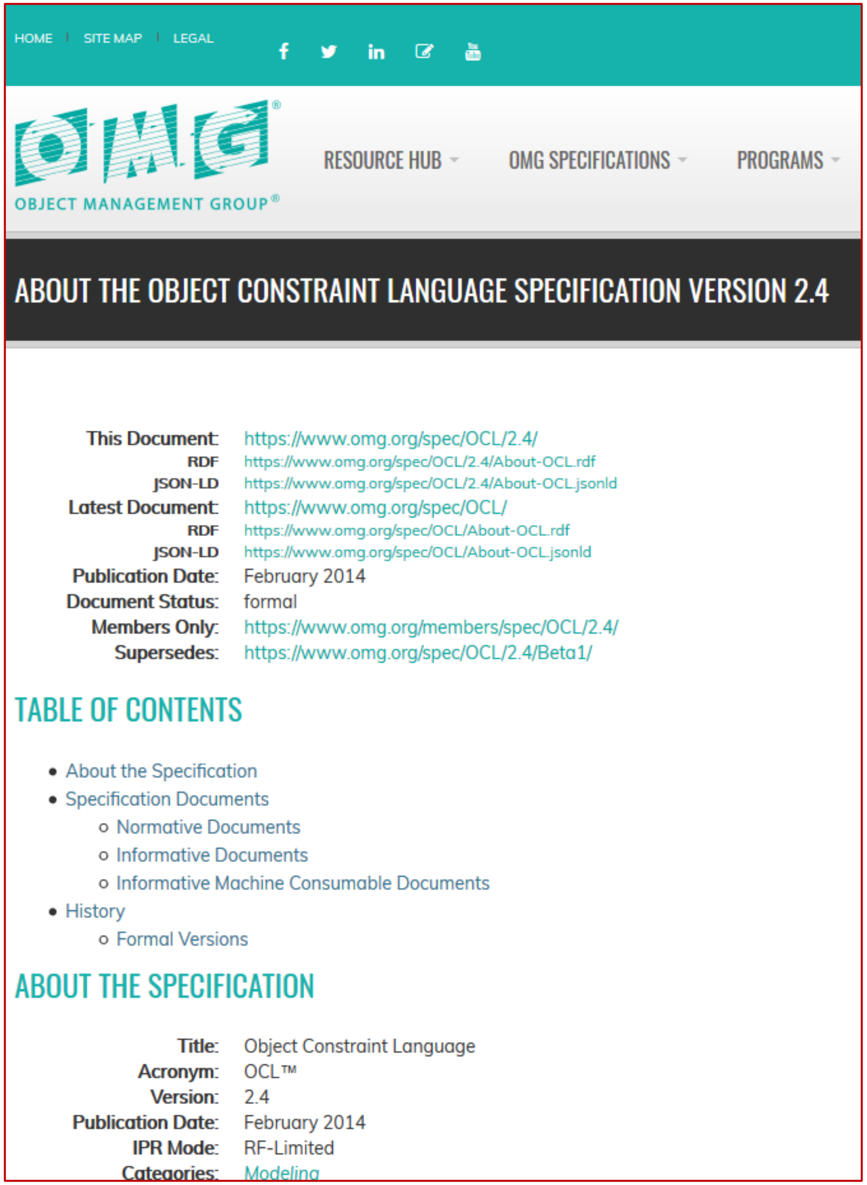


Fig. 13.3. State transition diagram of *Rental*

Conceptual Modeling of Information Systems (chapter 13, page 307)

More (if you have curiosity…):
https://www.omg.org/spec/OCL/About-OCL/
https://modeling-languages.com/ocl-tutorial/



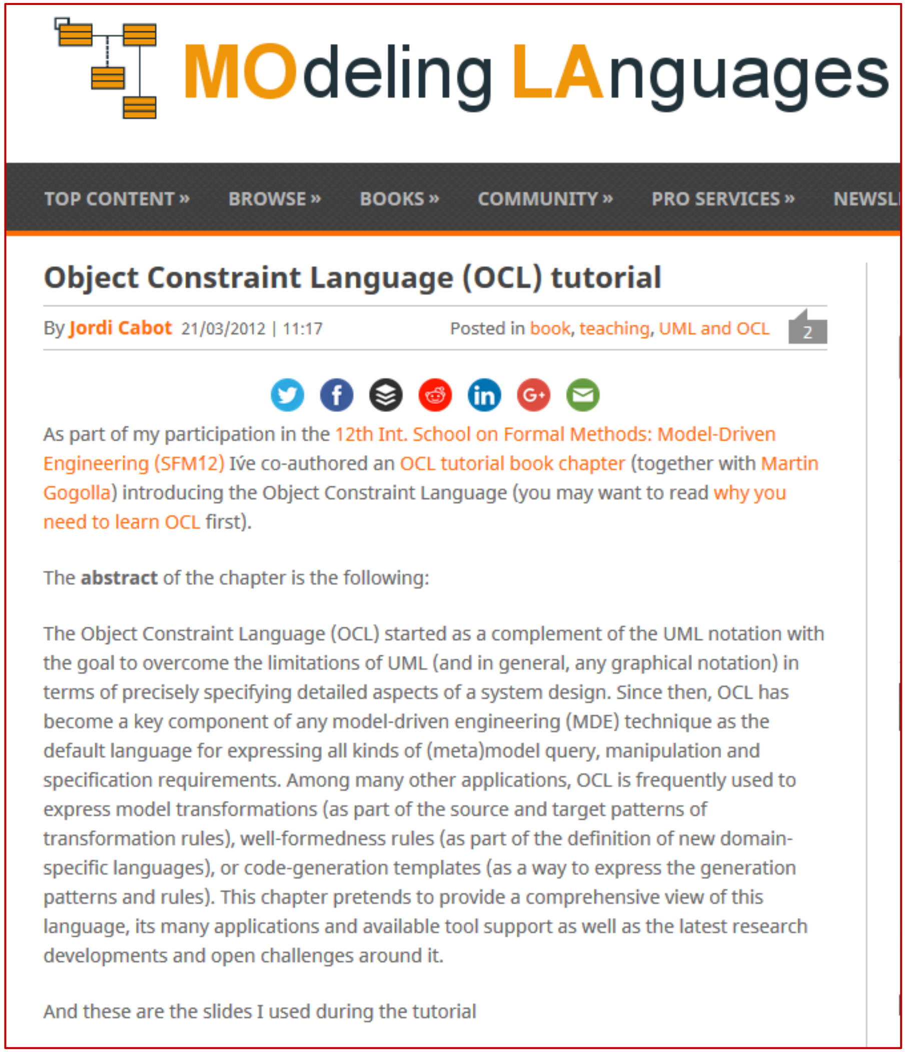
ABOUT THE OBJECT CONSTRAINT LANGUAGE SPECIFICATION VERSION 2.4

This Document:	https://www.omg.org/spec/OCL/2.4/
RDF	https://www.omg.org/spec/OCL/2.4/About-OCL.rdf
JSON-LD	https://www.omg.org/spec/OCL/2.4/About-OCL.jsonld
Latest Document:	https://www.omg.org/spec/OCL/
RDF	https://www.omg.org/spec/OCL/About-OCL.rdf
JSON-LD	https://www.omg.org/spec/OCL/About-OCL.jsonld
Publication Date:	February 2014
Document Status:	formal
Members Only:	https://www.omg.org/members/spec/OCL/2.4/
Supersedes:	https://www.omg.org/spec/OCL/2.4/Beta1/

TABLE OF CONTENTS

ABOUT THE SPECIFICATION

Title:	Object Constraint Language
Acronym:	OCL™
Version:	2.4
Publication Date:	February 2014
IPR Mode:	RF-Limited
Categories:	Modeling



Object Constraint Language (OCL) tutorial

By **Jordi Cabot** 21/03/2012 | 11:17 Posted in book, teaching, UML and OCL 2

As part of my participation in the 12th Int. School on Formal Methods: Model-Driven Engineering (SFM12) Iv́e co-authored an OCL tutorial book chapter (together with Martin Gogolla) introducing the Object Constraint Language (you may want to read why you need to learn OCL first).

The **abstract** of the chapter is the following:

The Object Constraint Language (OCL) started as a complement of the UML notation with the goal to overcome the limitations of UML (and in general, any graphical notation) in terms of precisely specifying detailed aspects of a system design. Since then, OCL has become a key component of any model-driven engineering (MDE) technique as the default language for expressing all kinds of (meta)model query, manipulation and specification requirements. Among many other applications, OCL is frequently used to express model transformations (as part of the source and target patterns of transformation rules), well-formedness rules (as part of the definition of new domain-specific languages), or code-generation templates (as a way to express the generation patterns and rules). This chapter pretends to provide a comprehensive view of this language, its many applications and available tool support as well as the latest research developments and open challenges around it.

And these are the slides I used during the tutorial