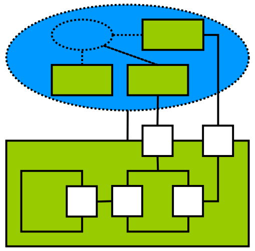
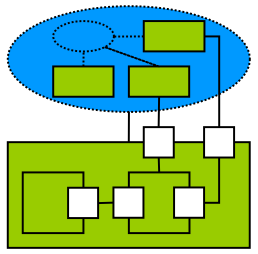
Composite Structure Diagram

Abstract

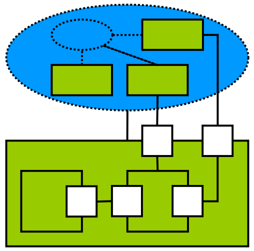


- UML 2.0 composite structure diagram
- Basic concepts
  - Structure, structured entity, internal structure
- Elements
  - Property
  - Connector
  - Nested notation
    - Description power
  - Classes & Structured Classes
    - Instance specification
    - Namespace behaviour
  - Collaboration
    - Purpose
  - Roles
  - Role binder
  - Collaboration Occurrence
    - Occurrence binder
    - <<occurrence>> & <<represent>>
  - Port
    - Visibility
    - Interfaces
- Examples

UML 2.0 diagram



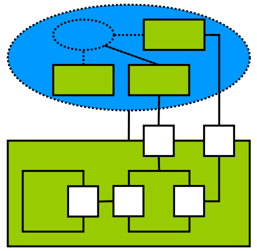
- Classified in UML 2.0 structural diagrams
- New:
  this diagram was not available in UML 1.*



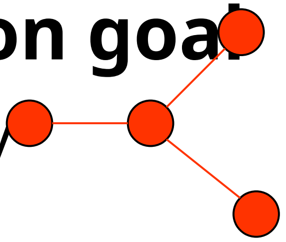
Purpose

- Composite structure diagrams can be used to describe:
  - structures of interconnected parts
  - run-time structures of interconnected instances
  - Example:
    Description of the parts of an engine that are interconnected to perform the engine functioning

Structure

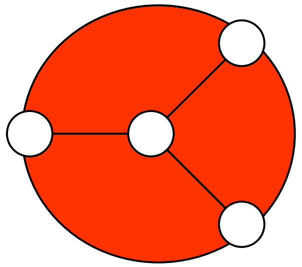


- A set of instances that communicate and collaborate at run-time to realize a **common goal**
  - Ex.: net routers that realize a particular journey



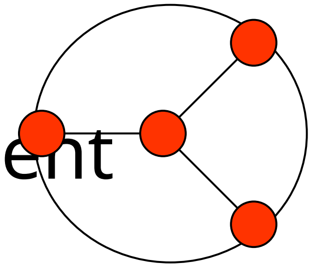
Structured element:

- An **element realized** by a structure
  - Ex.: a net realized by routers



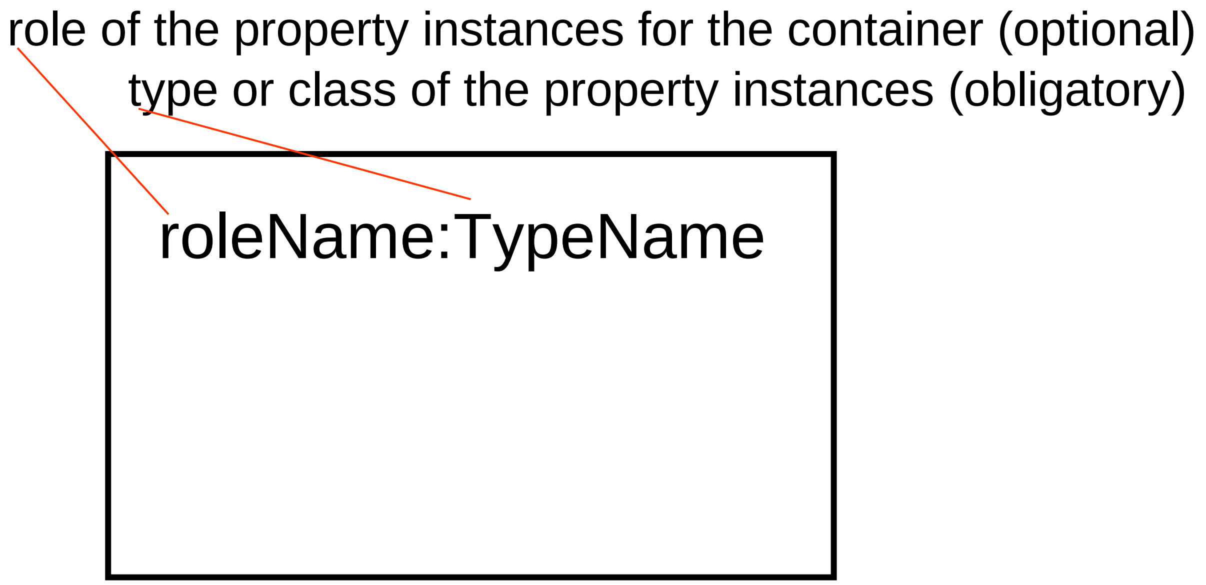
Internal structure:

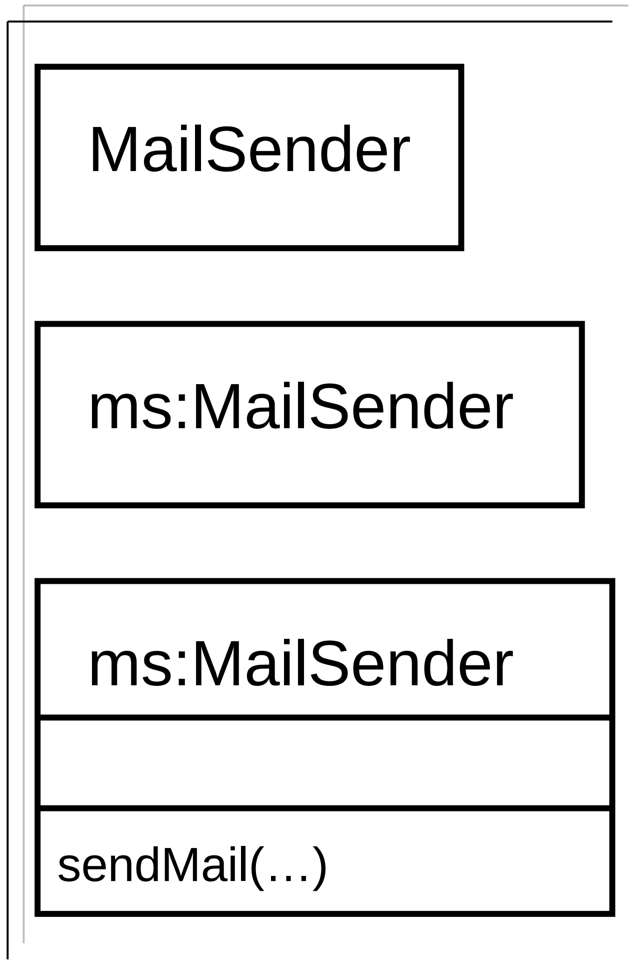
- A structure that **realize** a structured element
  - Ex.: all the routers in a net



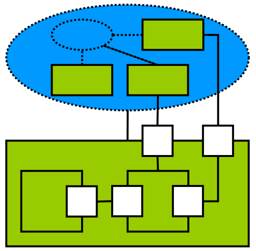
Property

- A set of **instances contained** in a structured instance

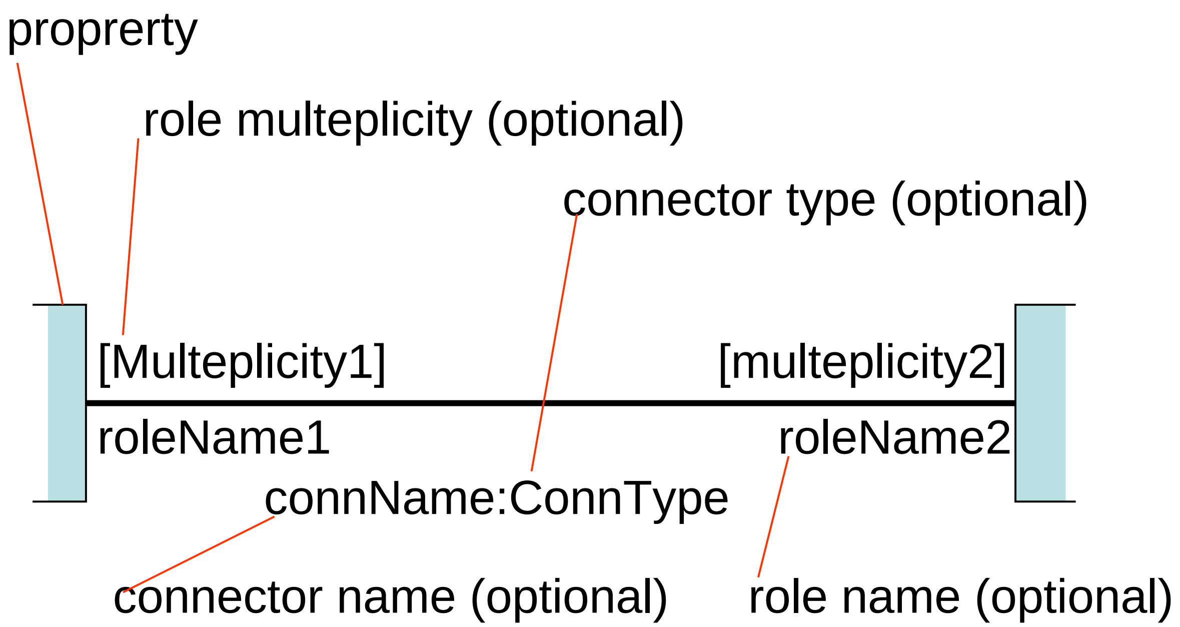

role of the property instances for the container (optional)
type or class of the property instances (obligatory)
roleName:TypeName

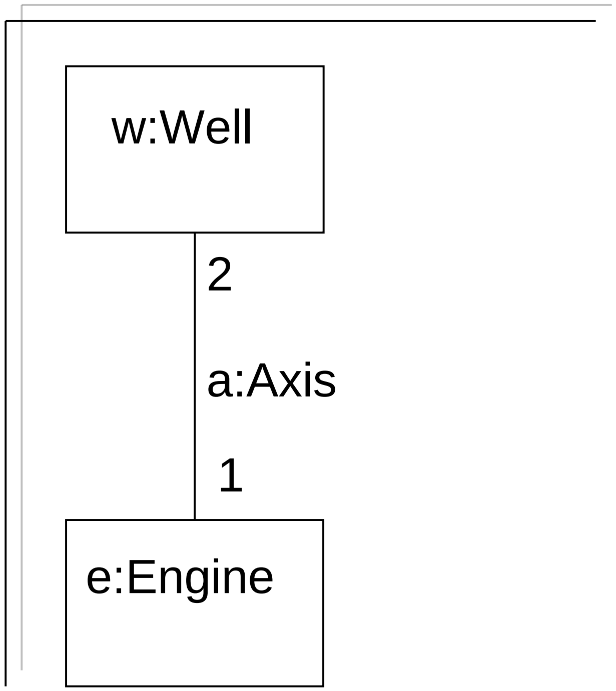


Connector



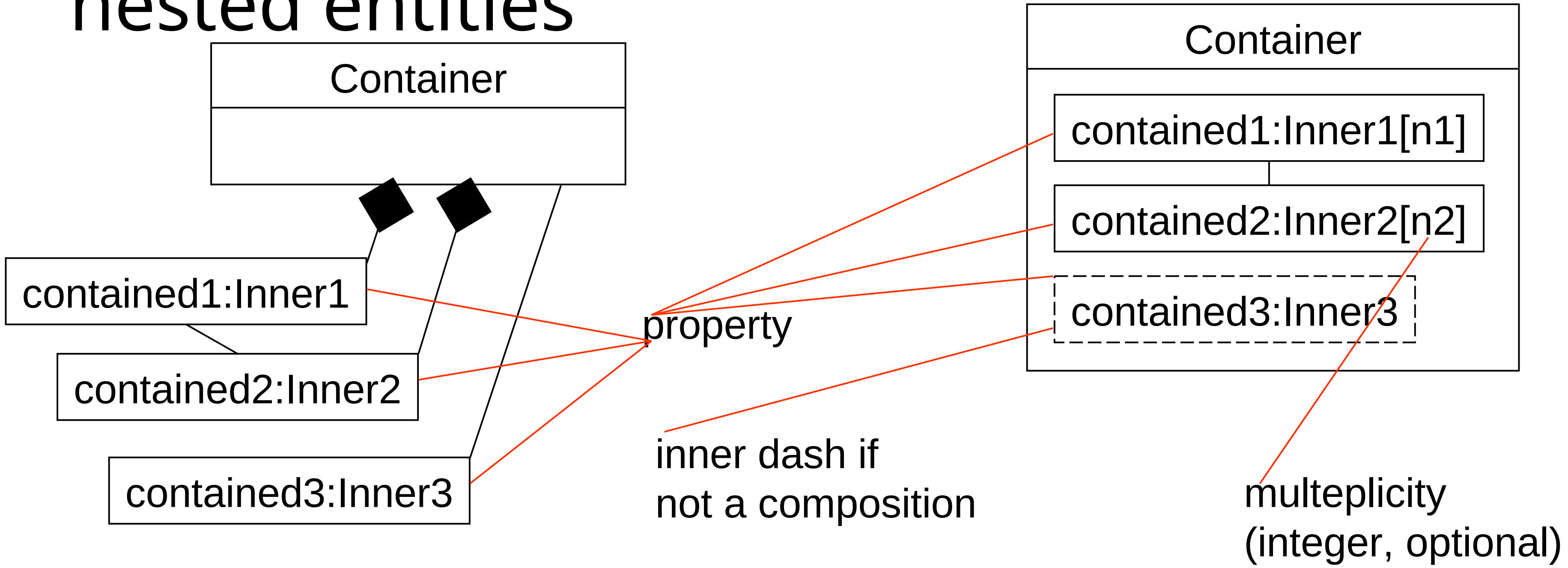
- Represents
  - the **visibility** between two property
  - a **communication way**



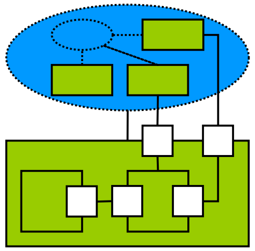


Nested notation

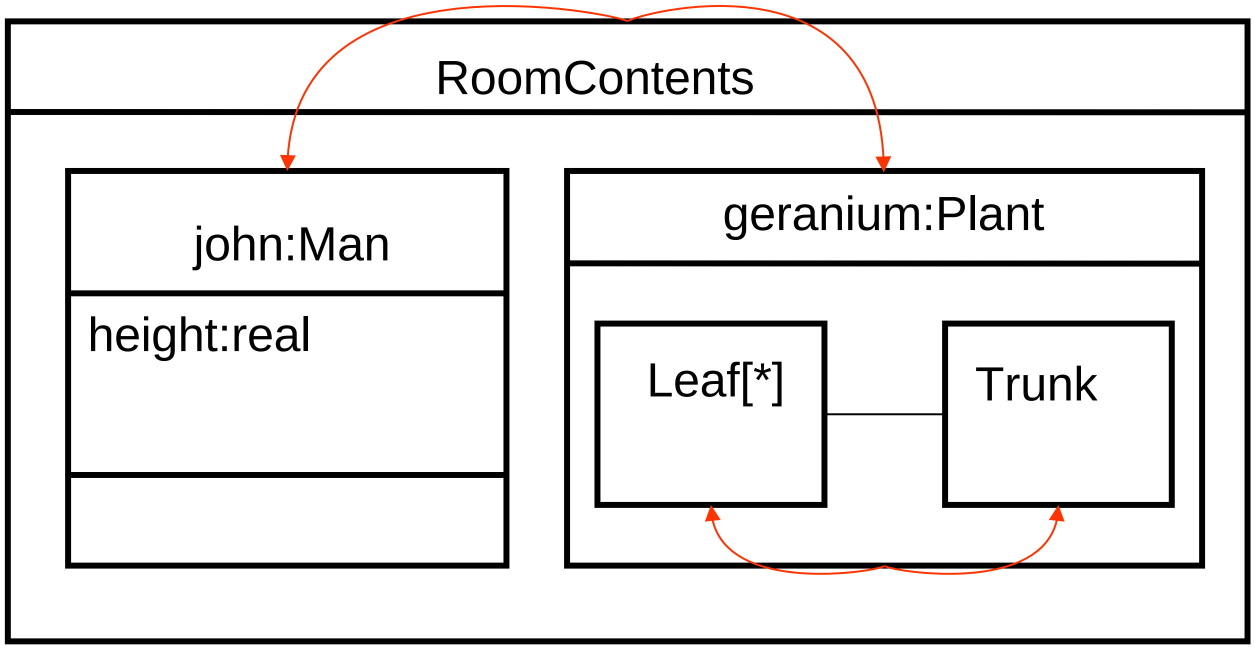
- Composite structure diagrams allows to use class diagram-like or **nested notation**
- It is permitted to **recursively nest** already nested entities



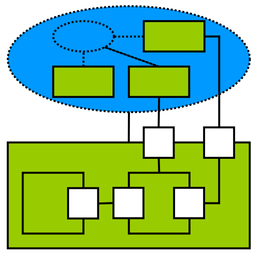
Structured Class: example



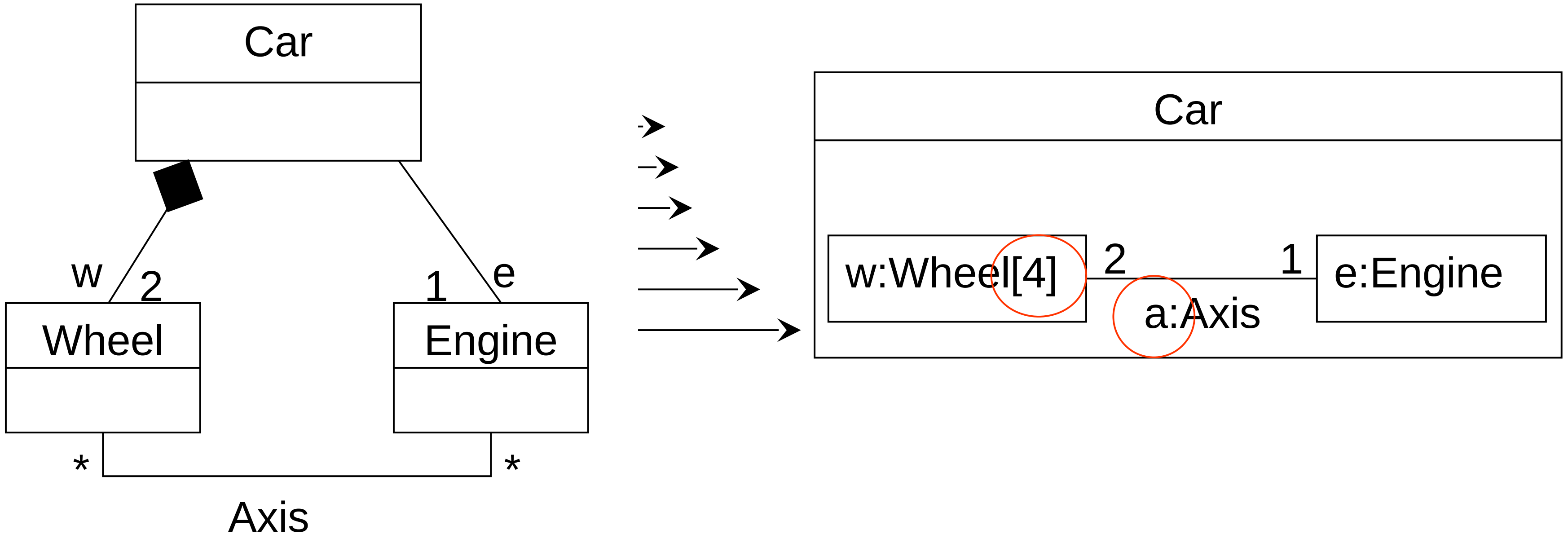
- **Recursive** application of nested notation can be done inside a single diagram



Description power



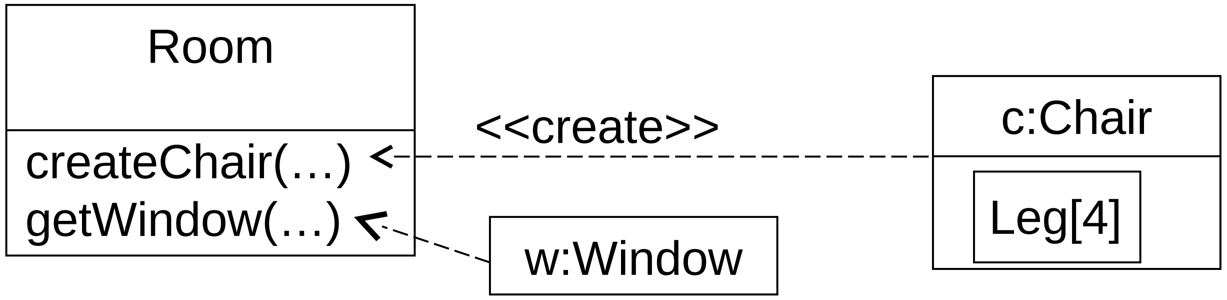
Nested notation can describe all things describable in 1.* class diagrams notation, and a little more



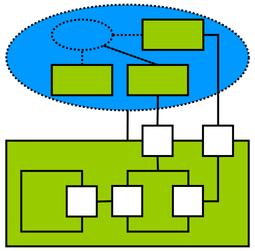
Instance specification

- describes the property **which is returned** by an operation call, the operation is pointed by the arrow at the end of a dashed line that starts from the returned type description

<<create>> is an optional label and specify that label exists only after the operation call

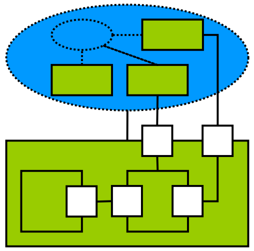


Namespace behaviour

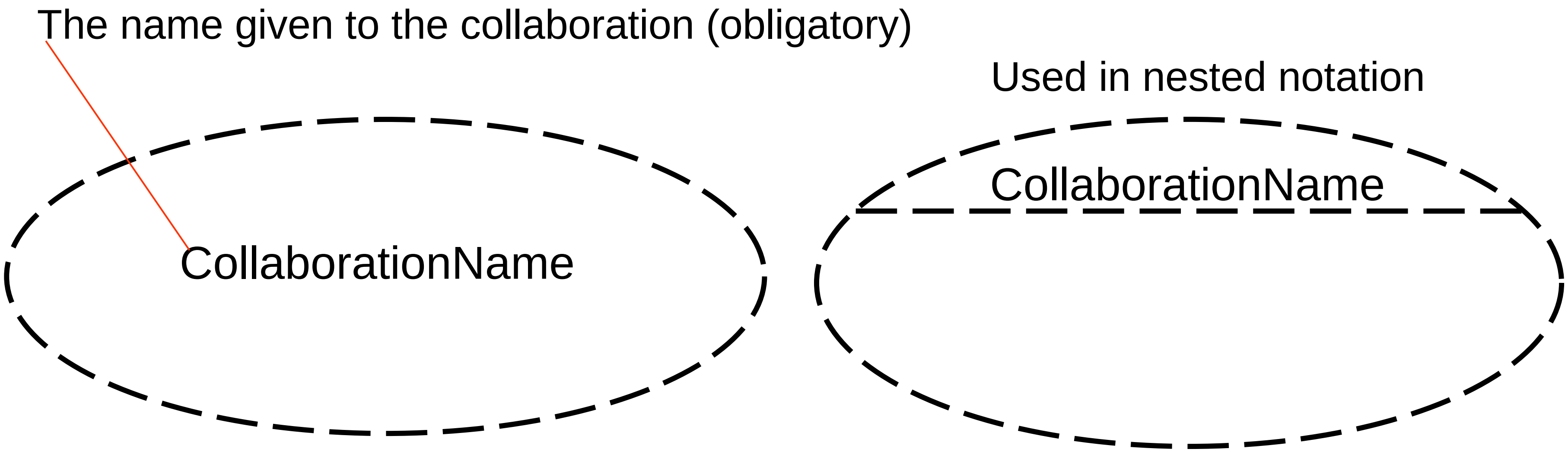


- A structured class acts as a **namespace** for its internal descriptions, so descriptions are not implicitly exported

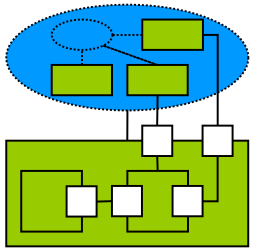
Collaboration



- A joining of structure elements that collaborate to **collectively perform** a task



Collaboration purpose

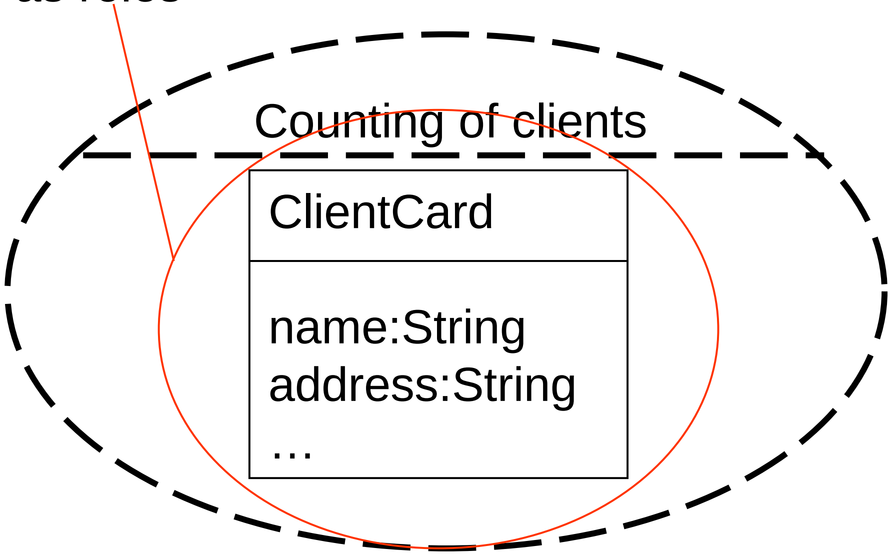


- A collaboration wants to **describe a structure behaviour** made by structure property
- Must be connected only with property which are **required** to perform its described behaviour

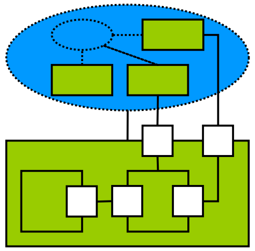
Collaboration role

- **property which collaborate** to perform the collaboration goal, interpreting roles
- Each collaboration role perform a **specific task**

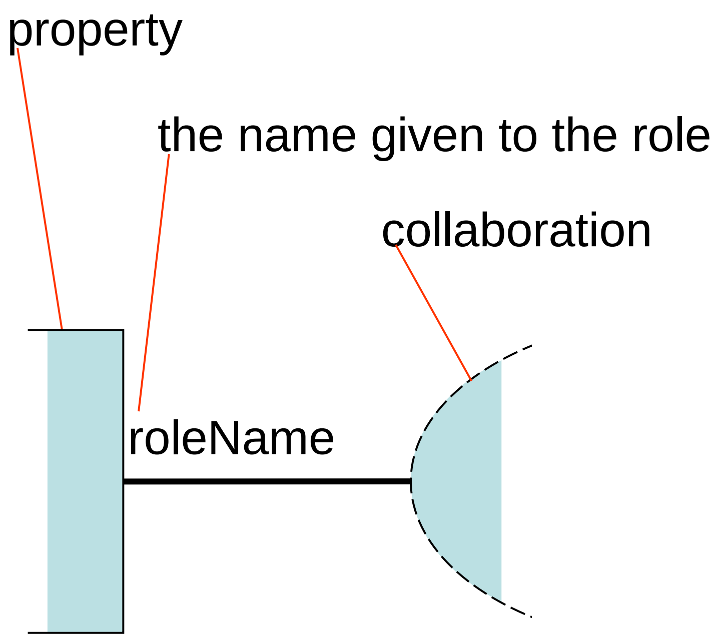
“Counting of clients” have some ClientCard as roles

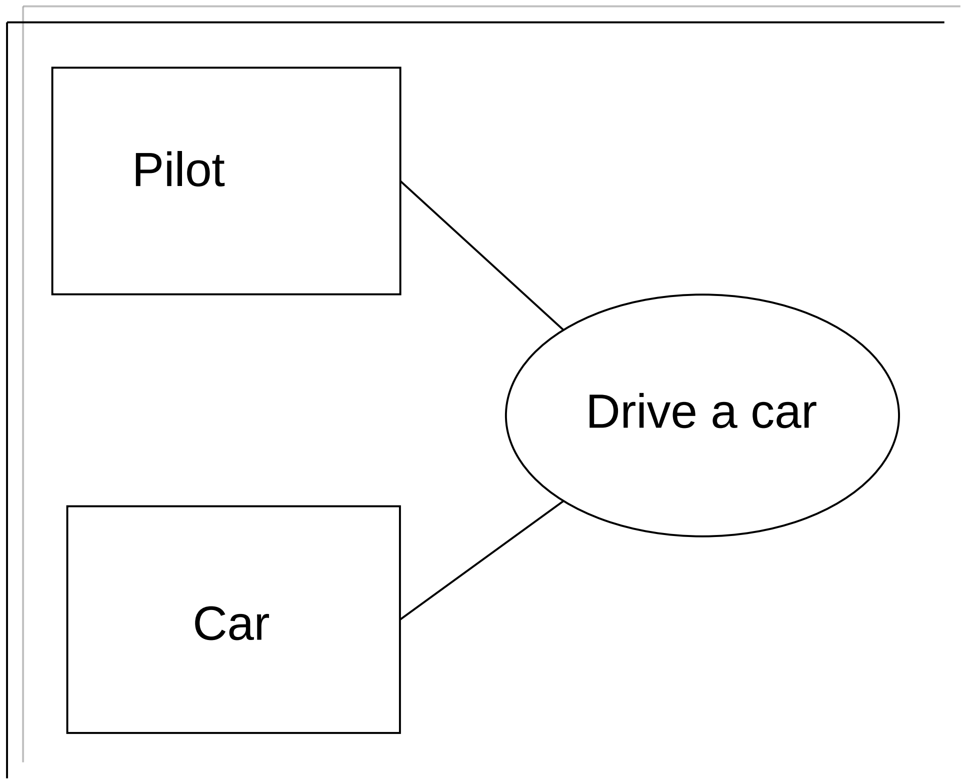


Collaboration Role Binder

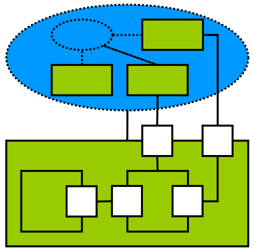


- represent a **participation** of the role to the collaboration

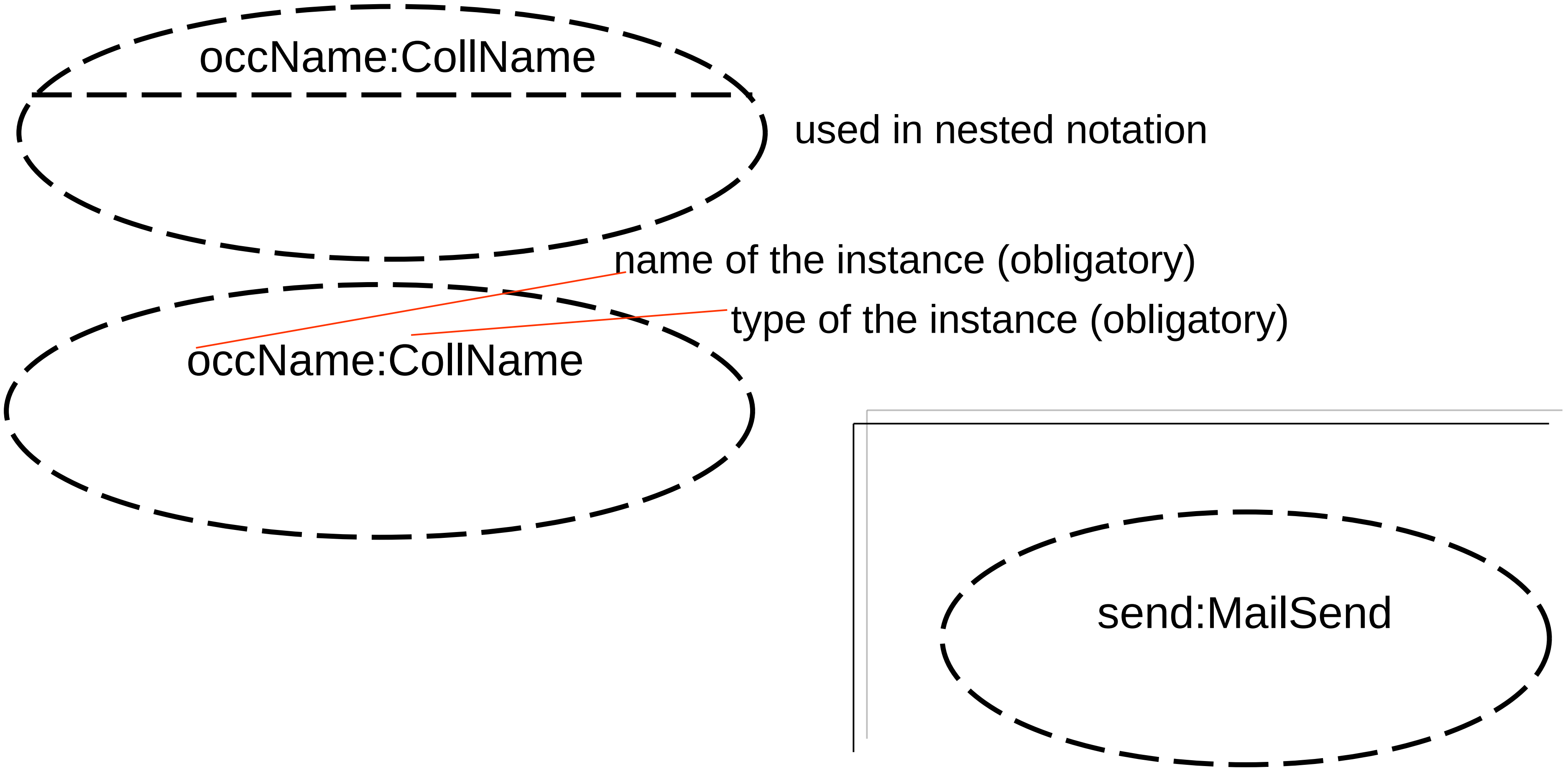




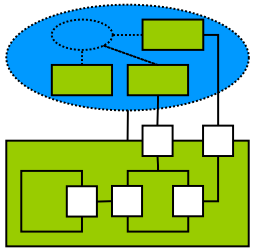
Collaboration Occurrence



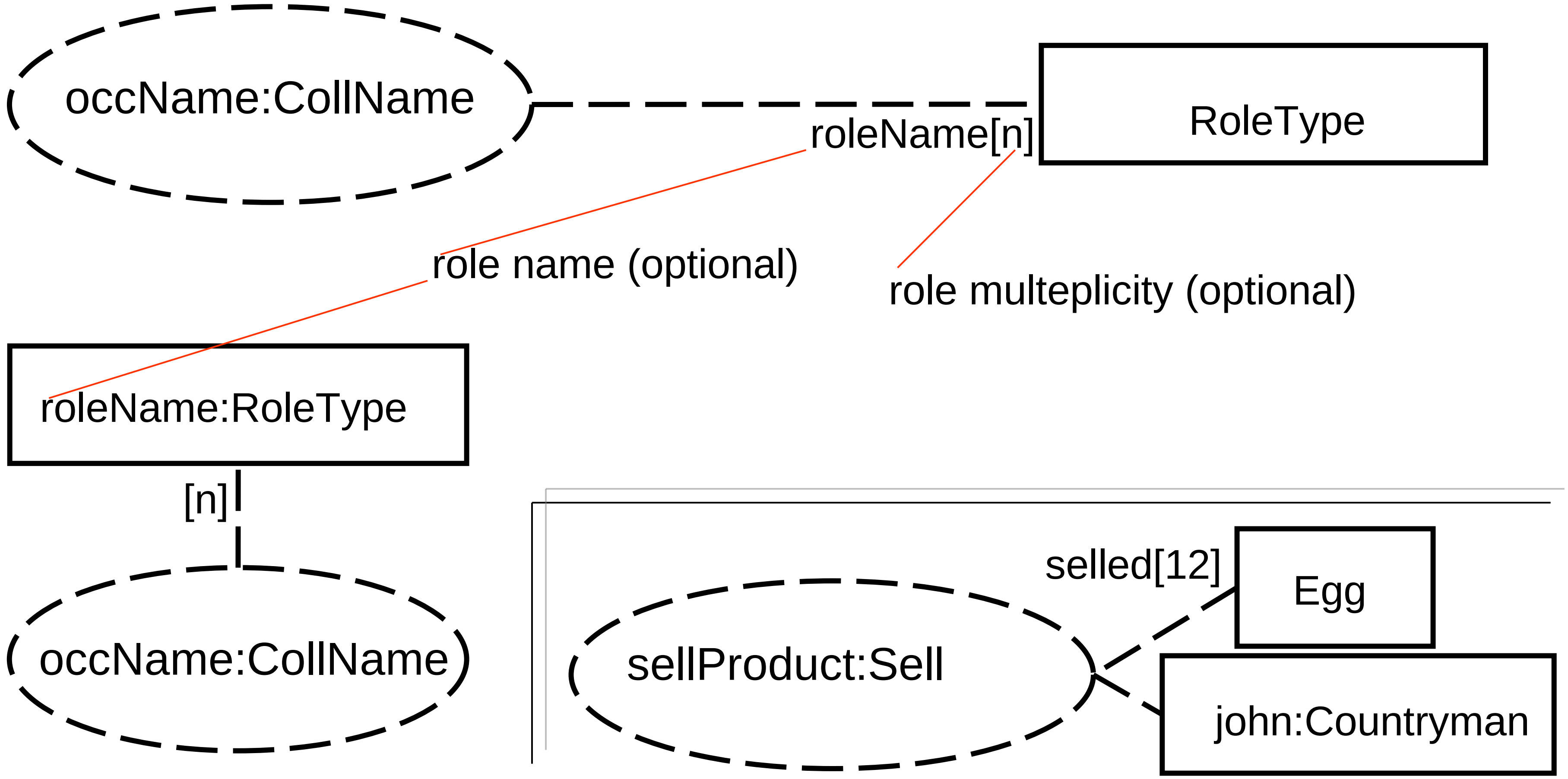
- A specific collaboration **instance**



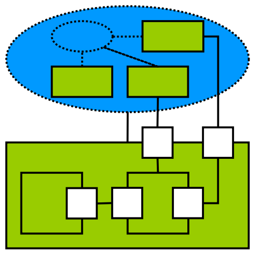
Occurrence binder

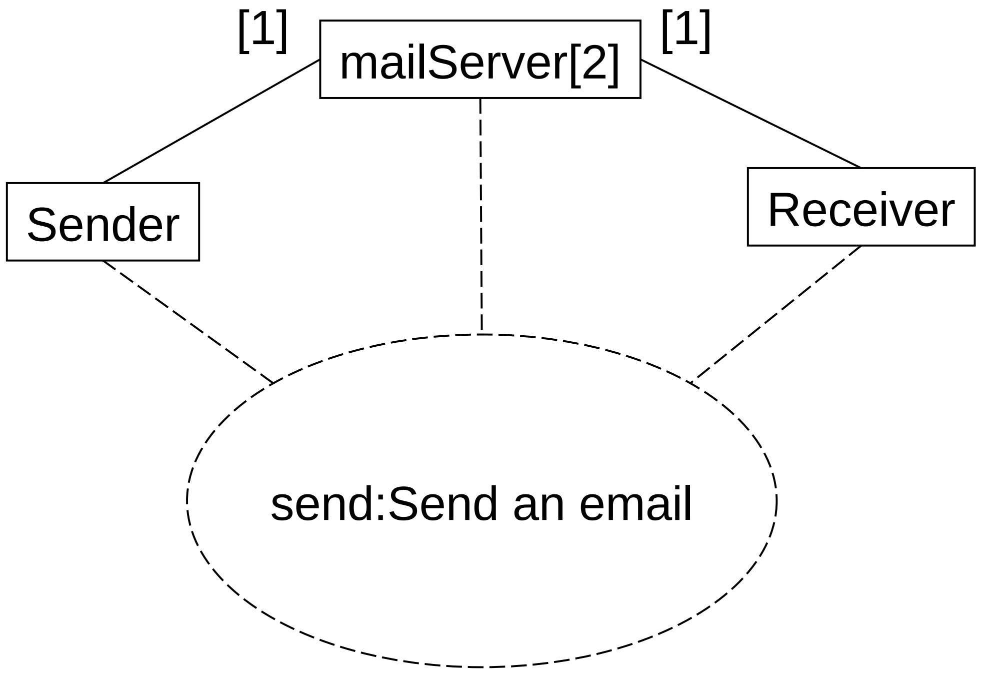


- Binds an occurrence to a role, maybe specifing how many occurrence **repetitions** are present



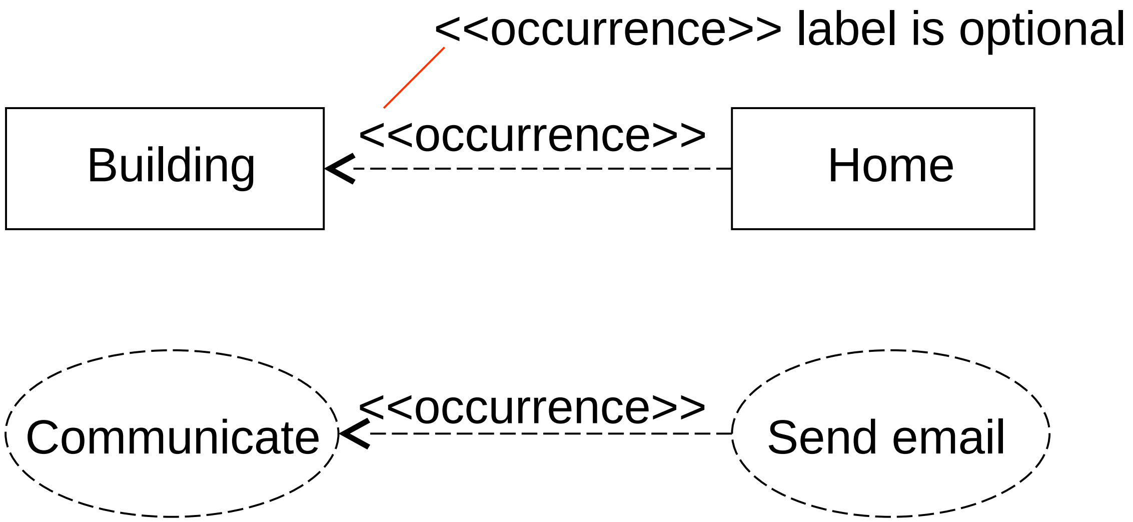
Example: occurrence - Send mail





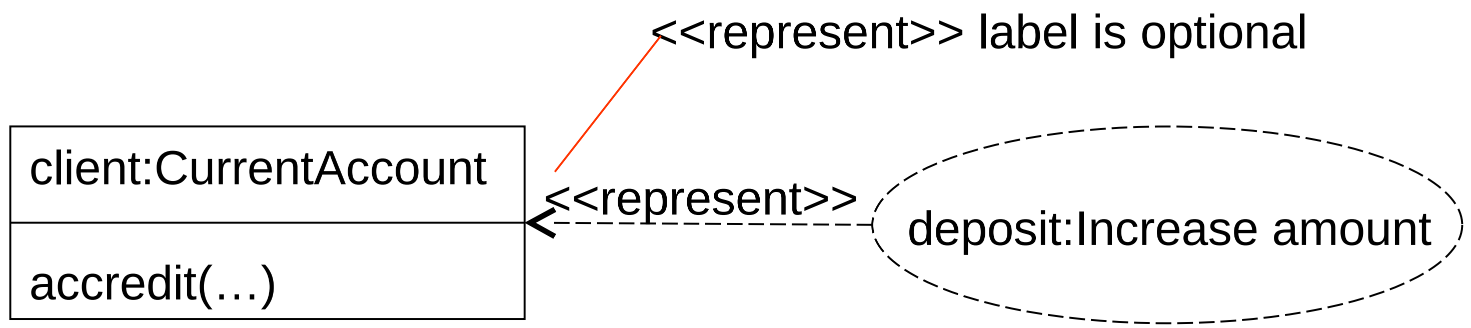
<<occurrence>> label

- A dashed arrow between two symbol of same type means that pointing symbol represent pointed symbol, like in a specialization

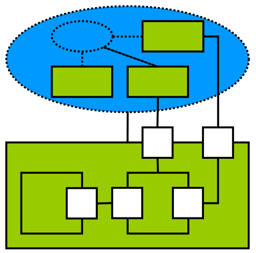


<<represent>> label

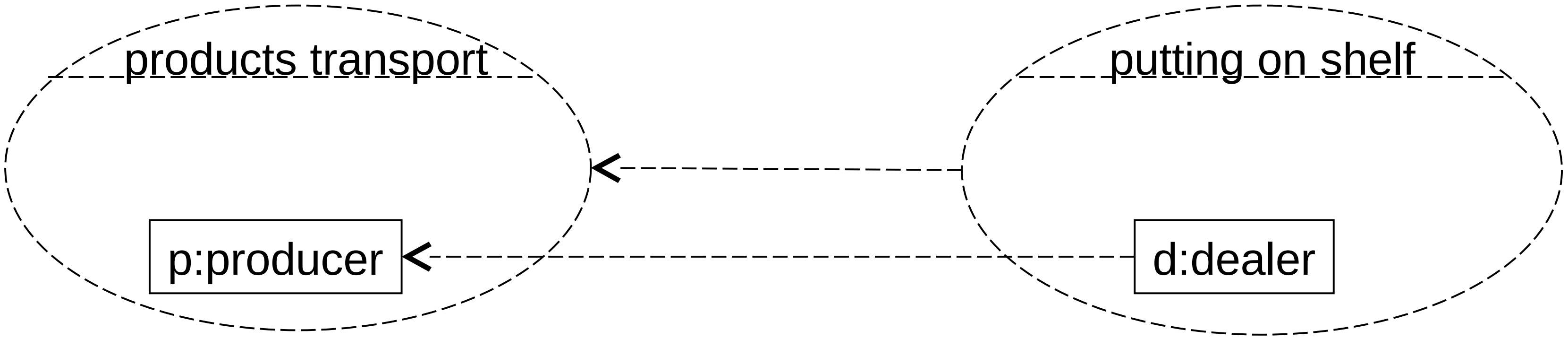
- A dashed arrow from a collaboration or an occurrence to a property means that property use the other instance, like a client or a use case primary actor



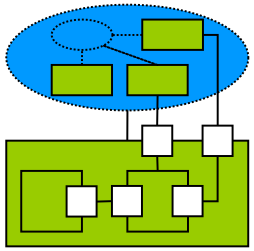
Occurrence binding nest crossing



- Occurrence binding is admitted between **separately nested** elements with also have occurrence bindings

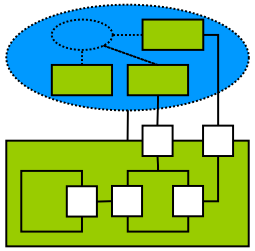


Port

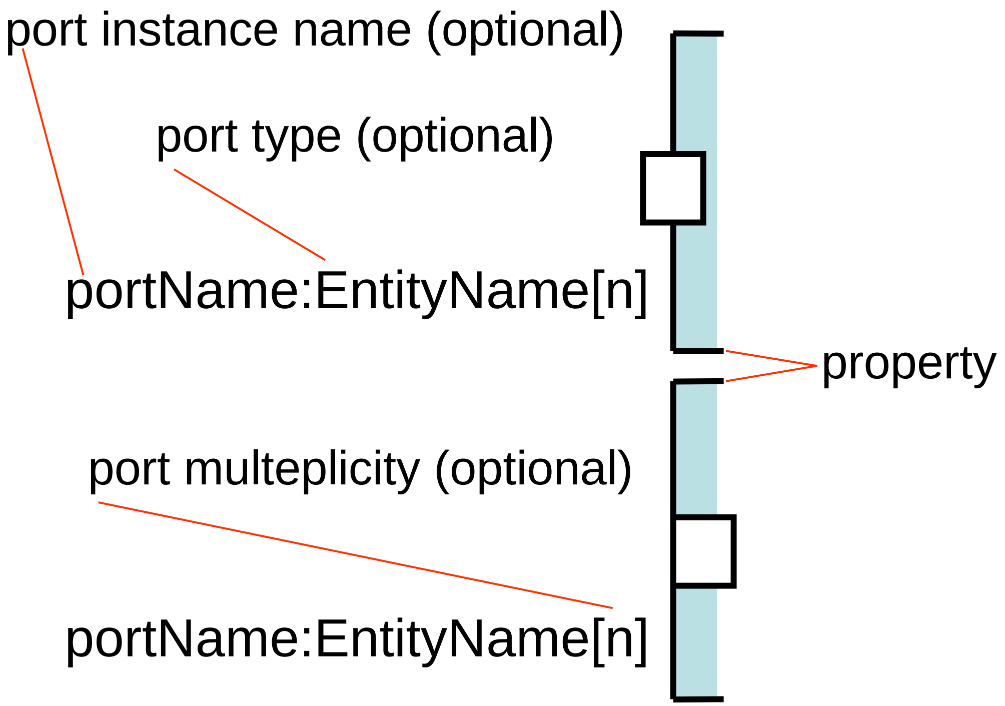


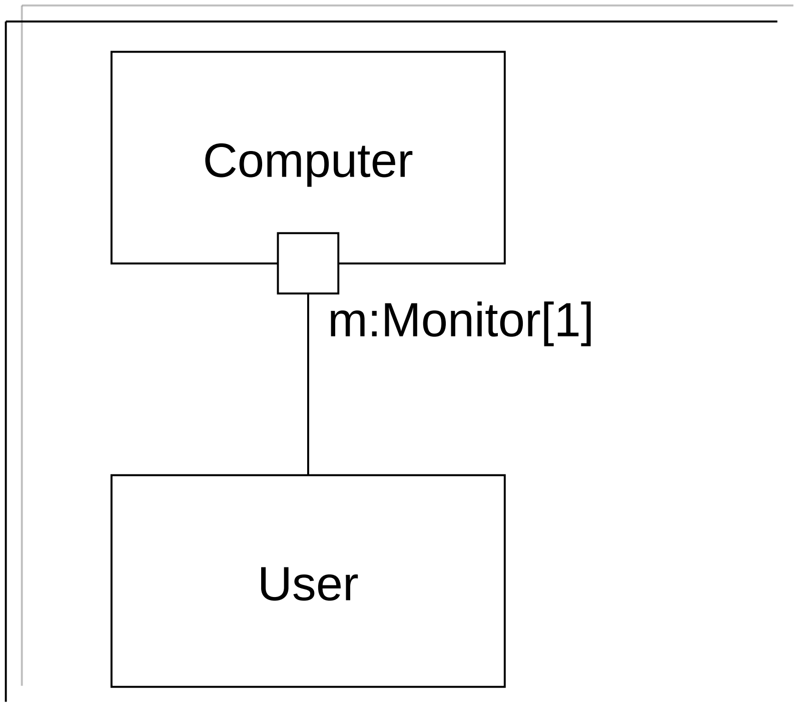
- represent a property **communication point**, and is always placed where the property joins whit its connector
- two types of communications:
  - Between a property and its **external environment**
  - Between a property and its **internal structure**

Ports: visibility marking

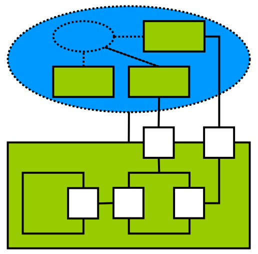


- If port symbol cover a rectangle boundary his visibility is **public**
- If port symbol is placed inside a rectangle, adjacent to his boundary, his visibility is **protected**



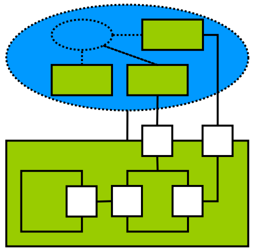


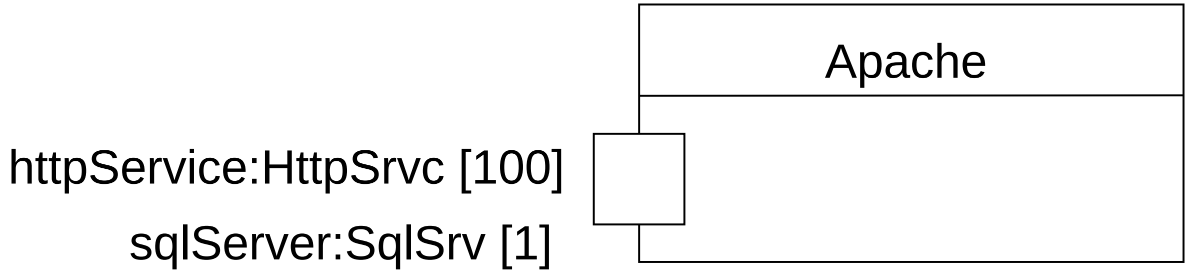
Ports: interfaces

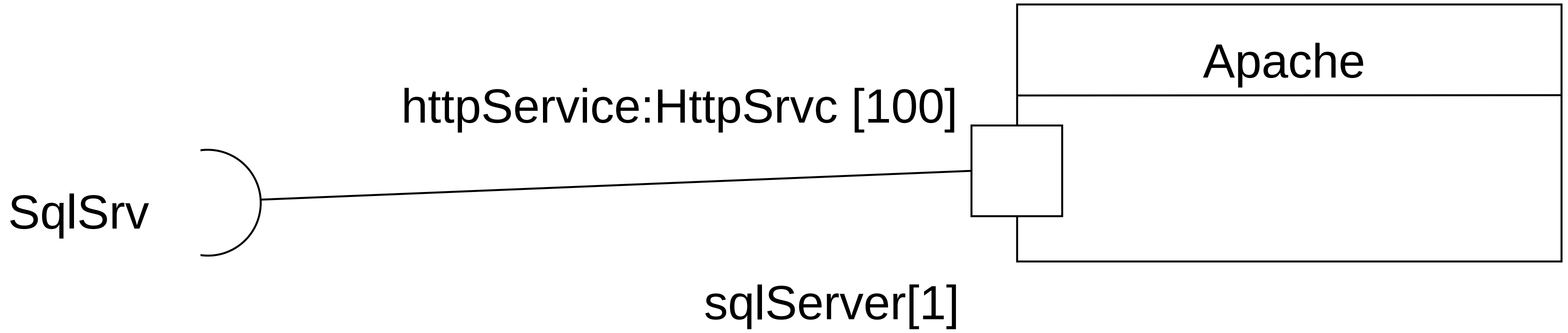


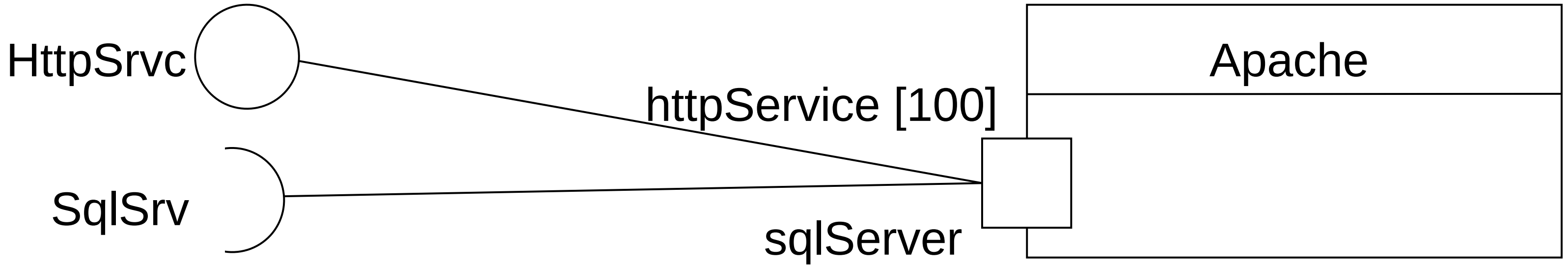
- An interface **exported** by a port is a little circle (interface symbol) connected with the port symbol by a line
- An interface **needed** by a port is a little semicircle (socket symbol) connected with the port symbol by a line
- If interface is present, interface **type** is signed near interface symbol

Ports: interfaces examples

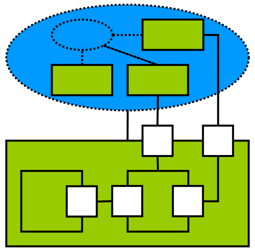




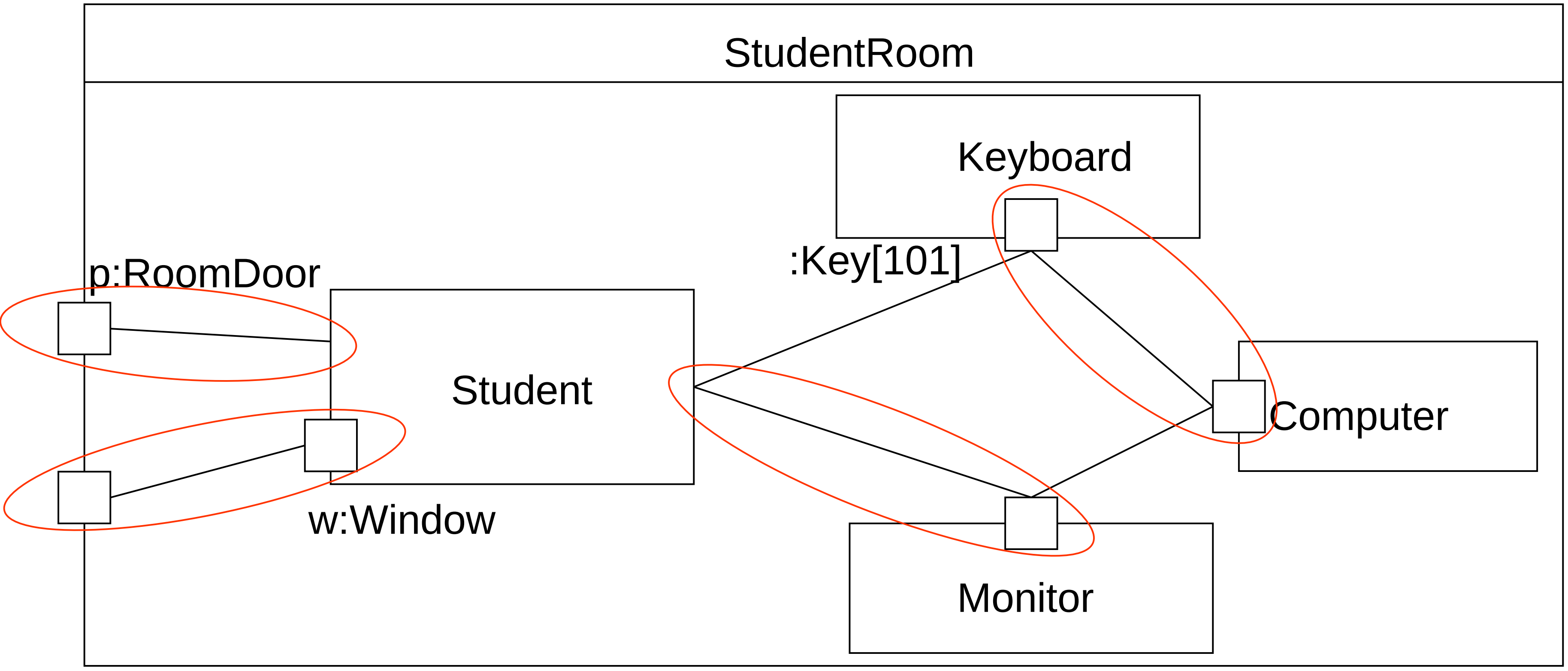




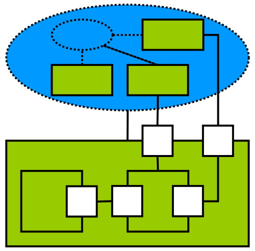
Example: Port

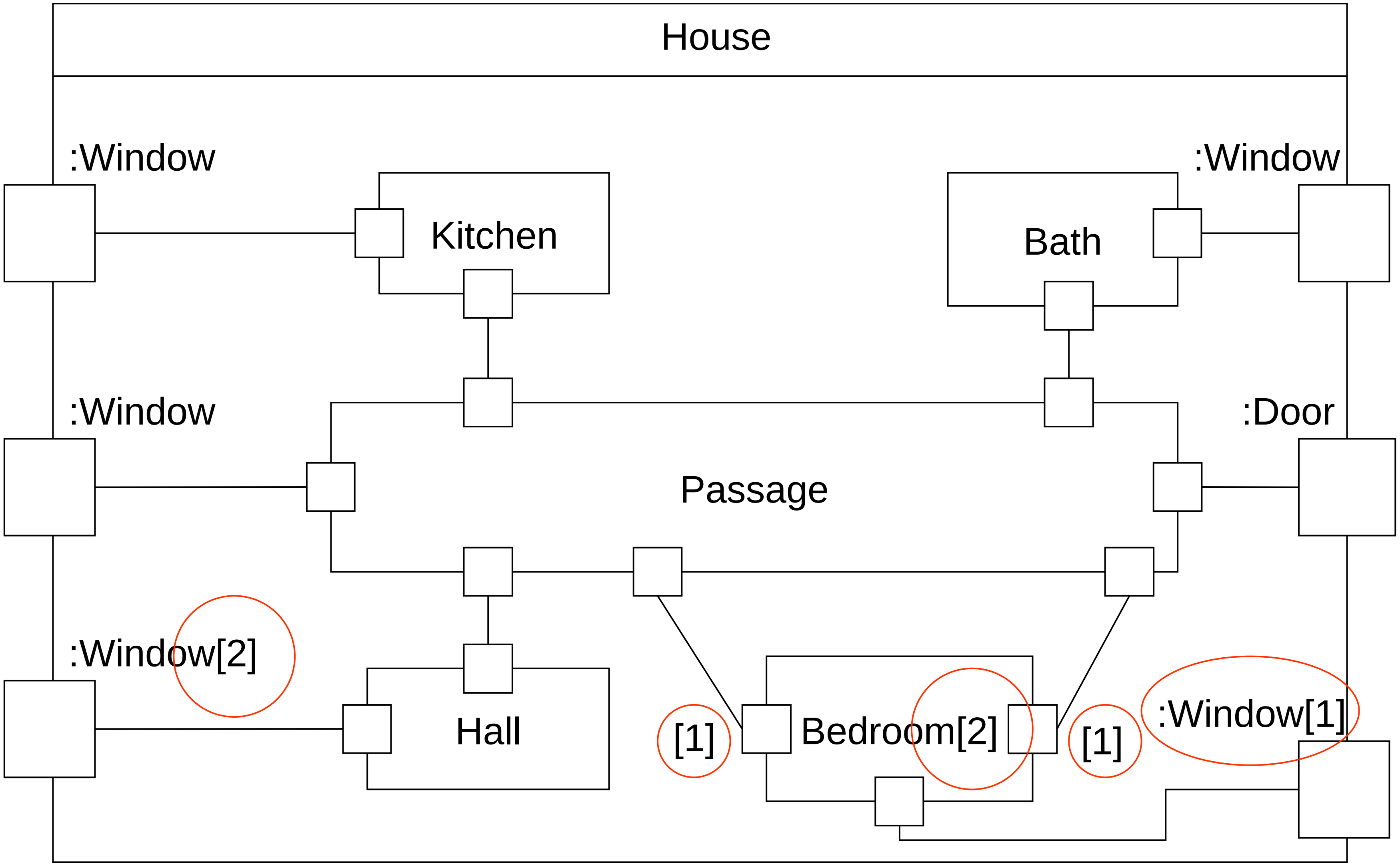


Structure with many kinds of connectors and ports:

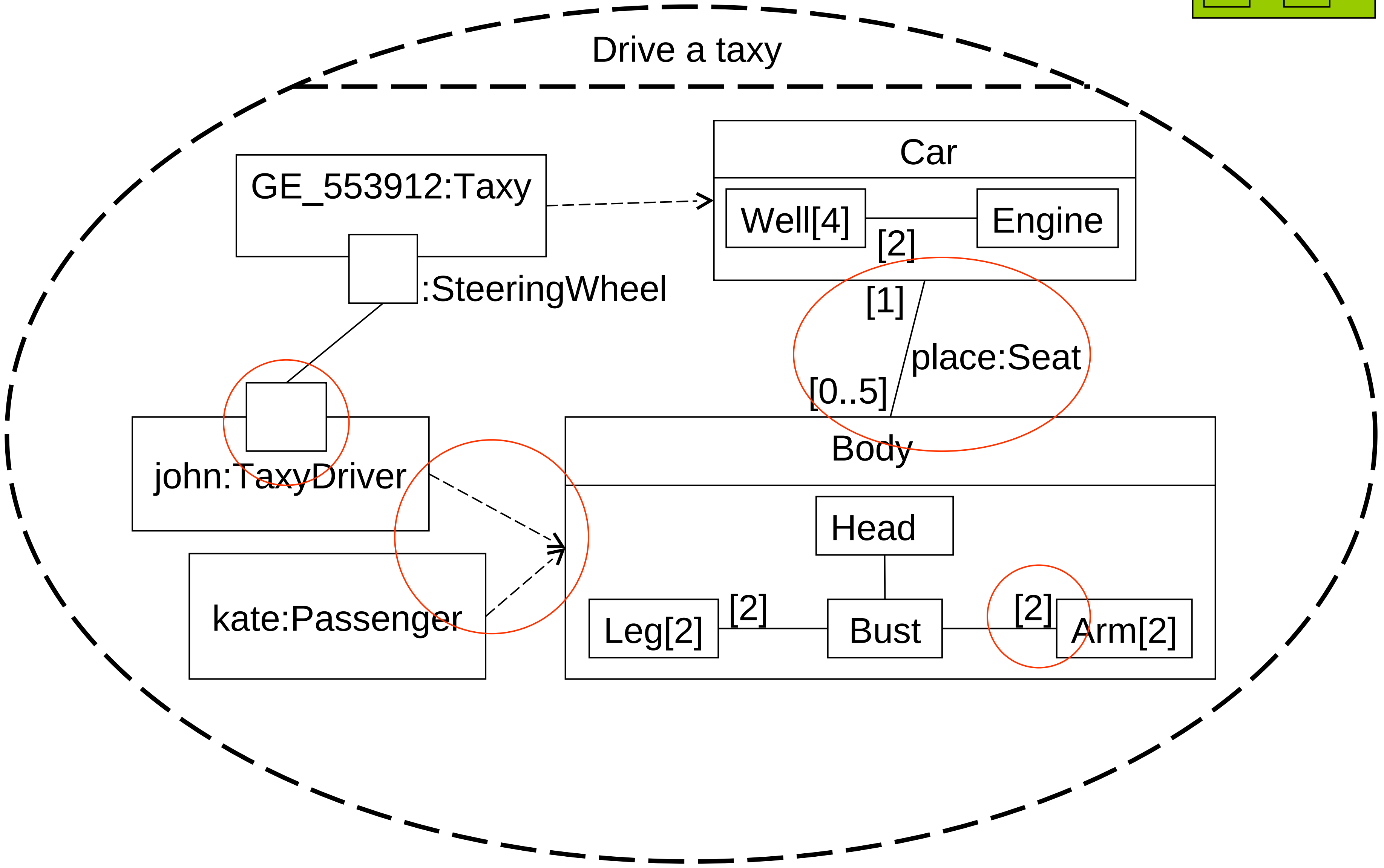


Example: structured class - House

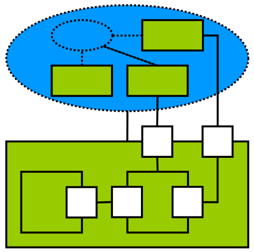




Example: collaboration - Drive a taxi



References



- OMG official site for UML:
  http://www.uml.org
- Agile software association
  Composite structure diagrams:
  http://www.agilemodeling.com/artifacts/compositeStructureDiagram.htm