SWEN 223

Software Engineering Analysis  
UML Class Diagrams

Thomas Kuhne

Victoria University of Wellington  
[Thomas.Kuehne@ecs.vuw.ac.nz](mailto:Thomas.Kuehne@ecs.vuw.ac.nz), Ext. 5443, Room Cotton 233



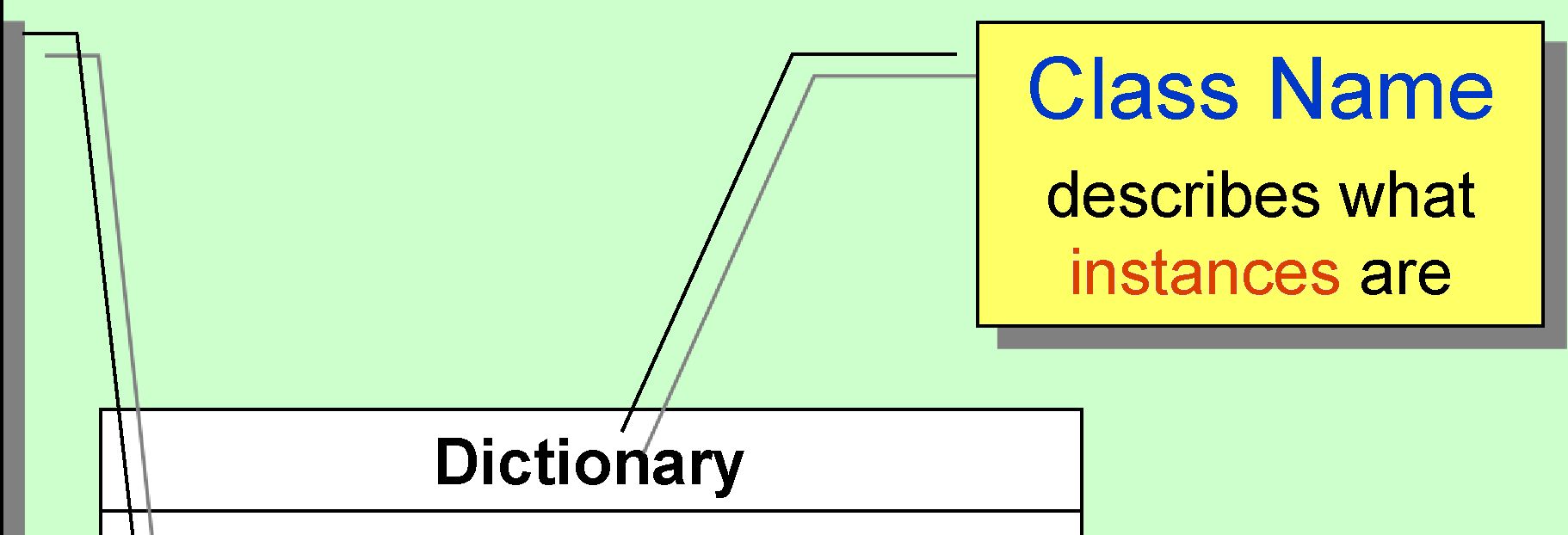
Class Specification

+ get (key : String) : Object + put (key : String, obj: Object) + count() : Integer '7 isEmptyO : Boolean

Result Type

(notation is up to the user)

- array : Array



Attribute

less important than  
operations, in  
particular in the  
analysis phase

Argument Type

(notation is up to the user)

Visibilty

+ public # protected ~ package - private

(exact meaning is a semantic variation point)

Derived

redundant but  
handy information



Attributes

What Information Should be Captured by Attributes?

• Attribute types will typically be Datatypes

» values for which unique identity is not useful » e.g. not usually meaningful to distinguish between

* instances of the number 5, or the string “cat”

(all primitive types are datatypes)

* instances of PhoneNumber that contain the same number
* etc.

—i

Attributes

Do Not Use Attributes as Foreign Keys

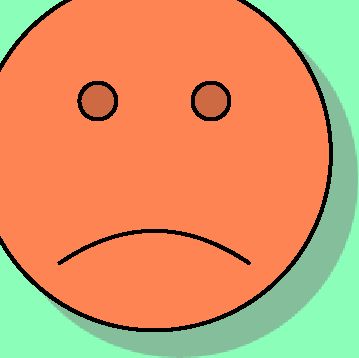
Passenger

name

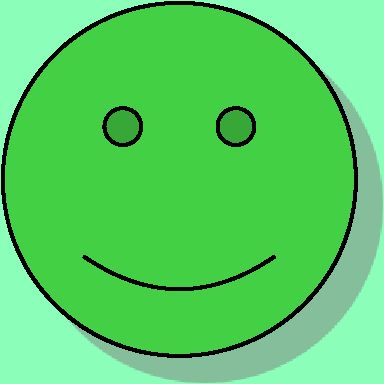
flightNumber

Flight

flightNumber



|  |  |  |
| --- | --- | --- |
| Passenger | \* \* | Flight |
|  |
| name | number |



-I ,ru\.

Analysis Phase

* Two (or more) concepts in the problem domain are in a relationship

» denotes some connection of some sort » usually with unspecified navigability

* Relational style is the norm

» association ends are owned by the association

» concepts are agnostic of the relationships they participate in

—i

|  |  |
| --- | --- |
| Category | Example |
| A is a member of B | Pilot—Airline |
| A uses or manages B | CEO—Airline |
| A communicates with B | ReservationAgent—Passenger |
| A is related to B | Reservation—Cancellation |
| A is next to B | City—City |
| A is owned by B | Plane—Airline |
| A is an organizational subunit of B | MaintenanceDepartment—Airline |



|  |  |
| --- | --- |
| Category | Example |
| A is logged / recorded in B | Reservation—FlightManifest |
| A is a physical part of B | Wing—Airplane |
| A is a logical part of B | FlightLeg—FlightRoute |
| A is physically contained in/on B | Passenger—Airplane |
| A is logically contained in B | Flight—FlightSchedule |
| A is a description for B | FlightDescription—Flight |



Design Phase

May result from

structural instance variables

temporal

message arguments message results

• Respective objects maintain links with each other

» required for access to objects, in particular for message sending

• Reference style is the norm

» association ends are owned by classes

» historically motivated implementation view which may be challenged in the future



Aggregation vs Composition

* Aggregation (white diamond)

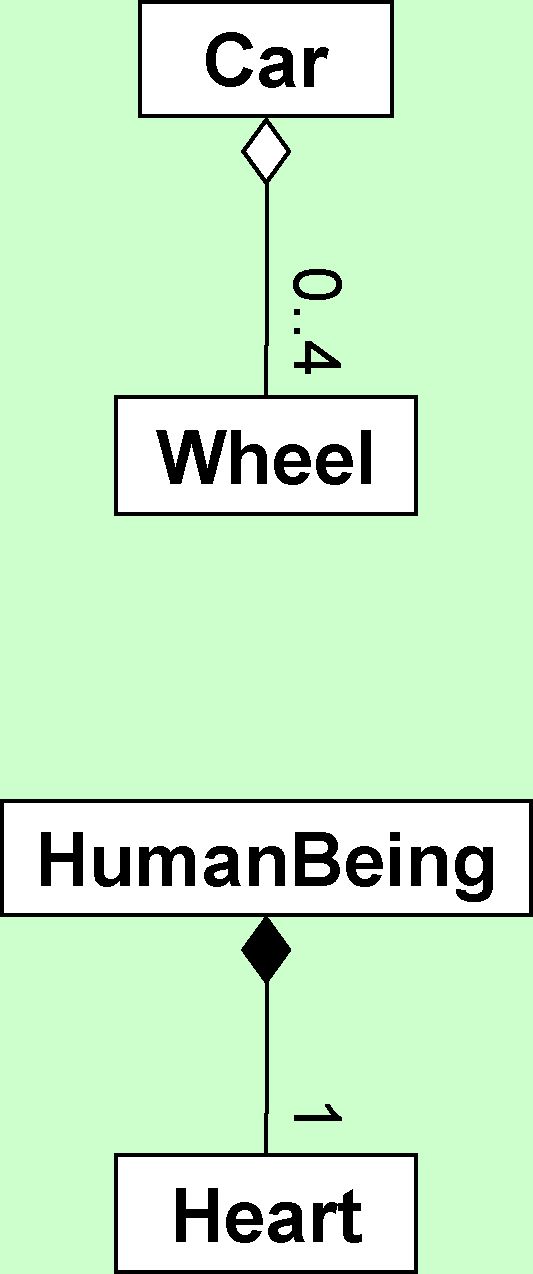
» regular association with whole-part connotation (anti-symmetric, transitive)

» no additional semantics attached

» if in doubt, use a regular association

* Composition (black diamond)

» parts cannot exist without the whole » synchronises lifetimes (transitively)



Some Standard Cases

1

A

1..\*

A

mandatory (one or more)

exactly one

|  |  |
| --- | --- |
| \* | A |
|  | A |

many (zero or more)

0..1

A

2..4, 6..8

A

optional (zero or one)numerically specified

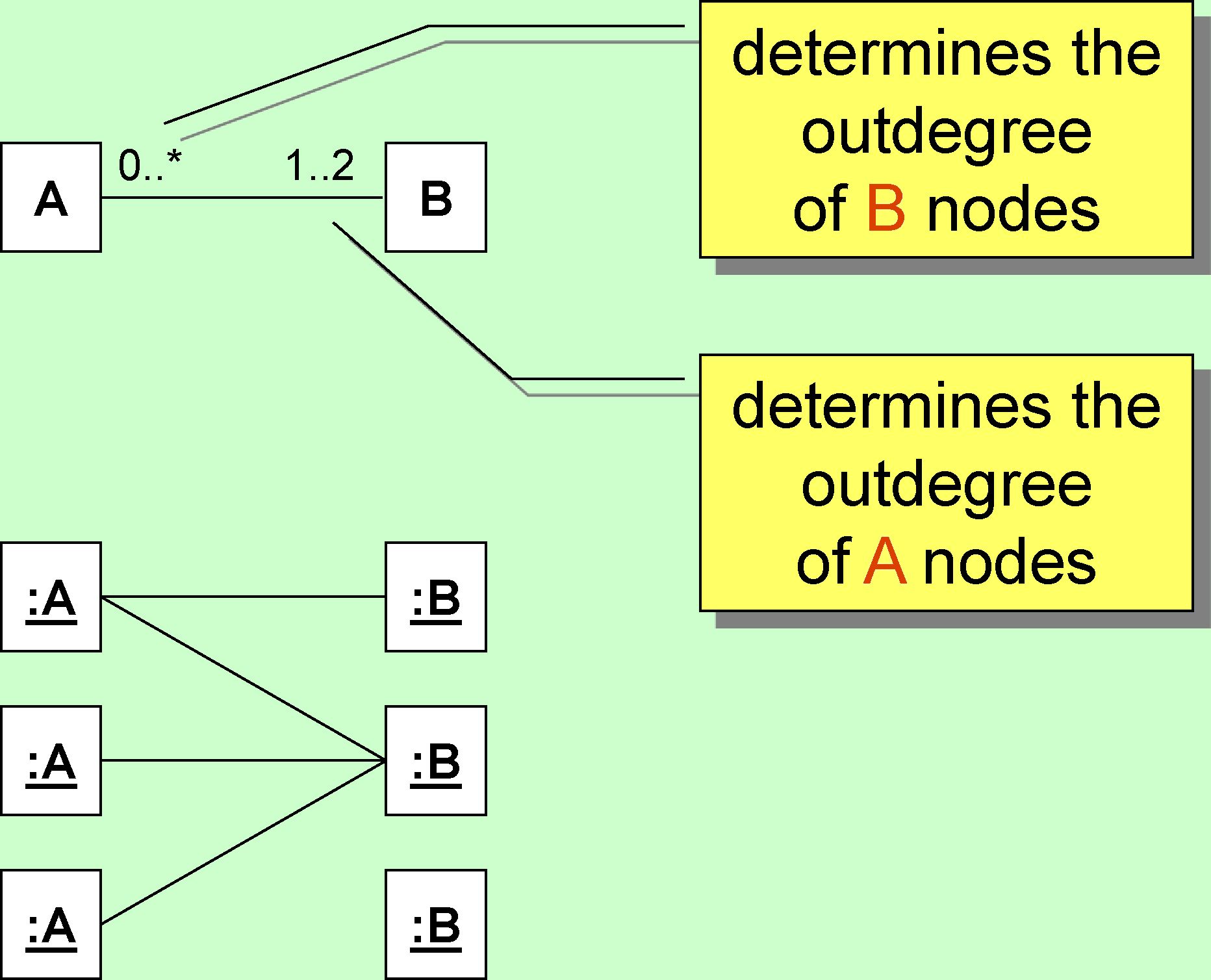


\*



Class Level

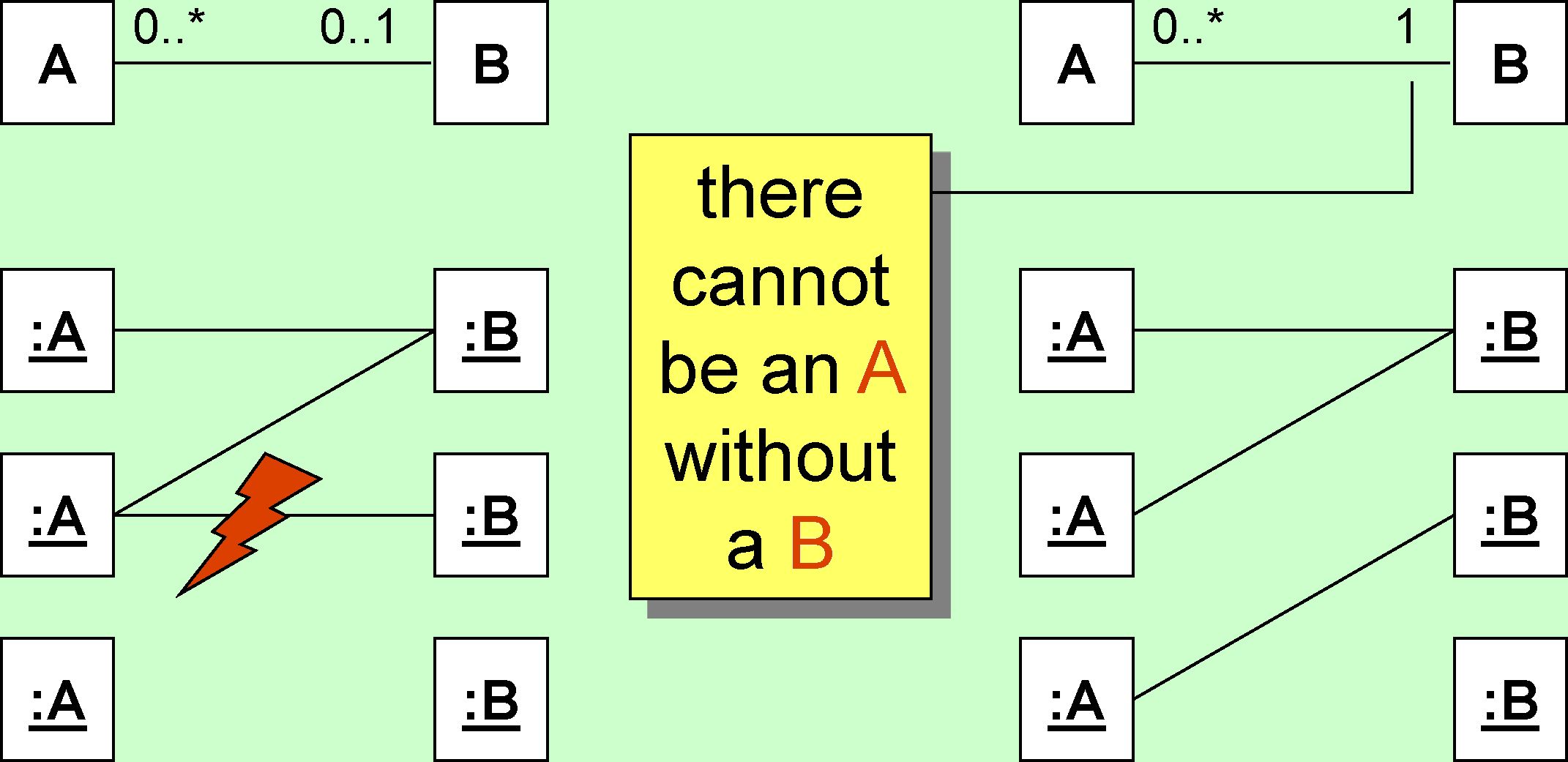
Object Level



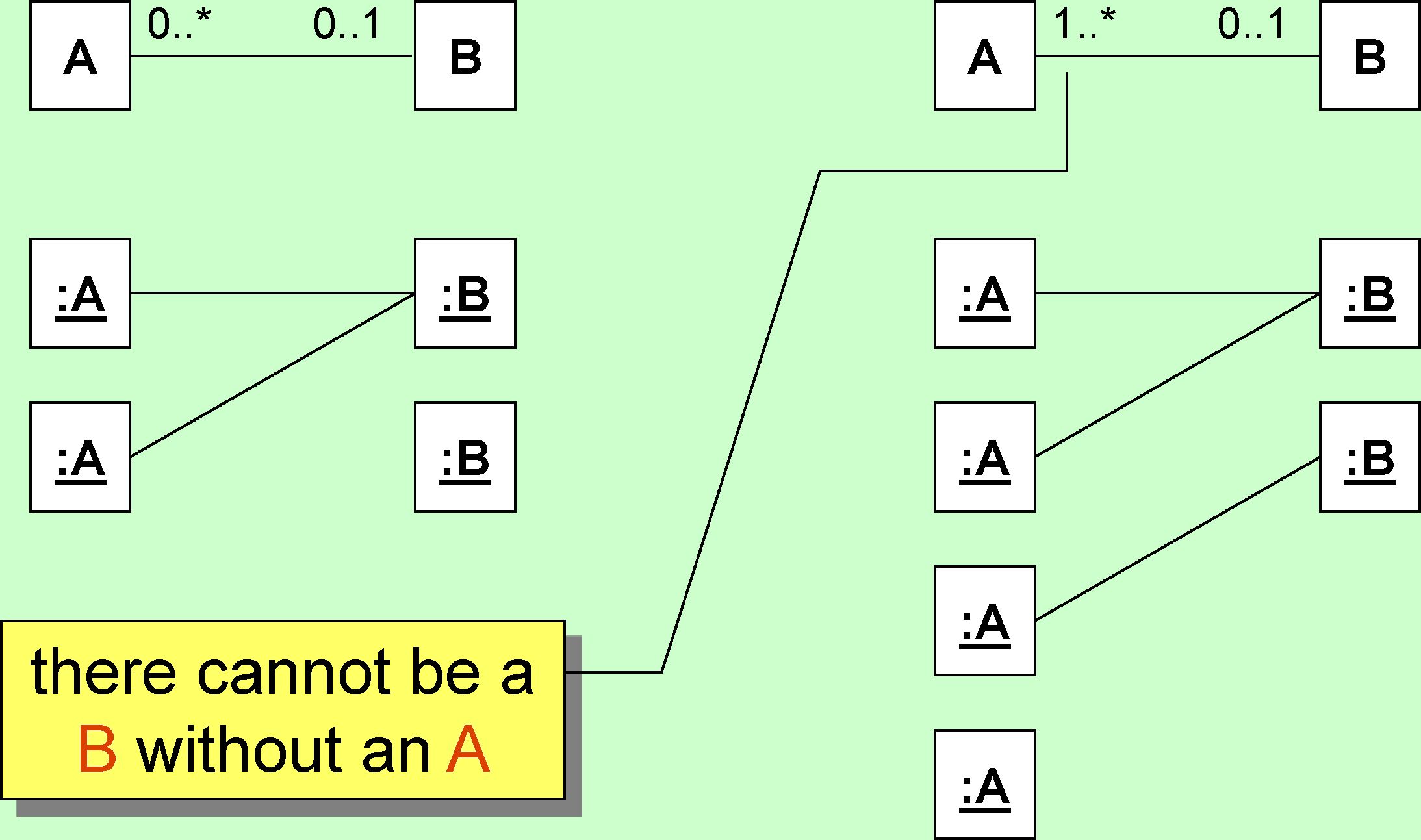
-I ,ru\.

Functional & Total

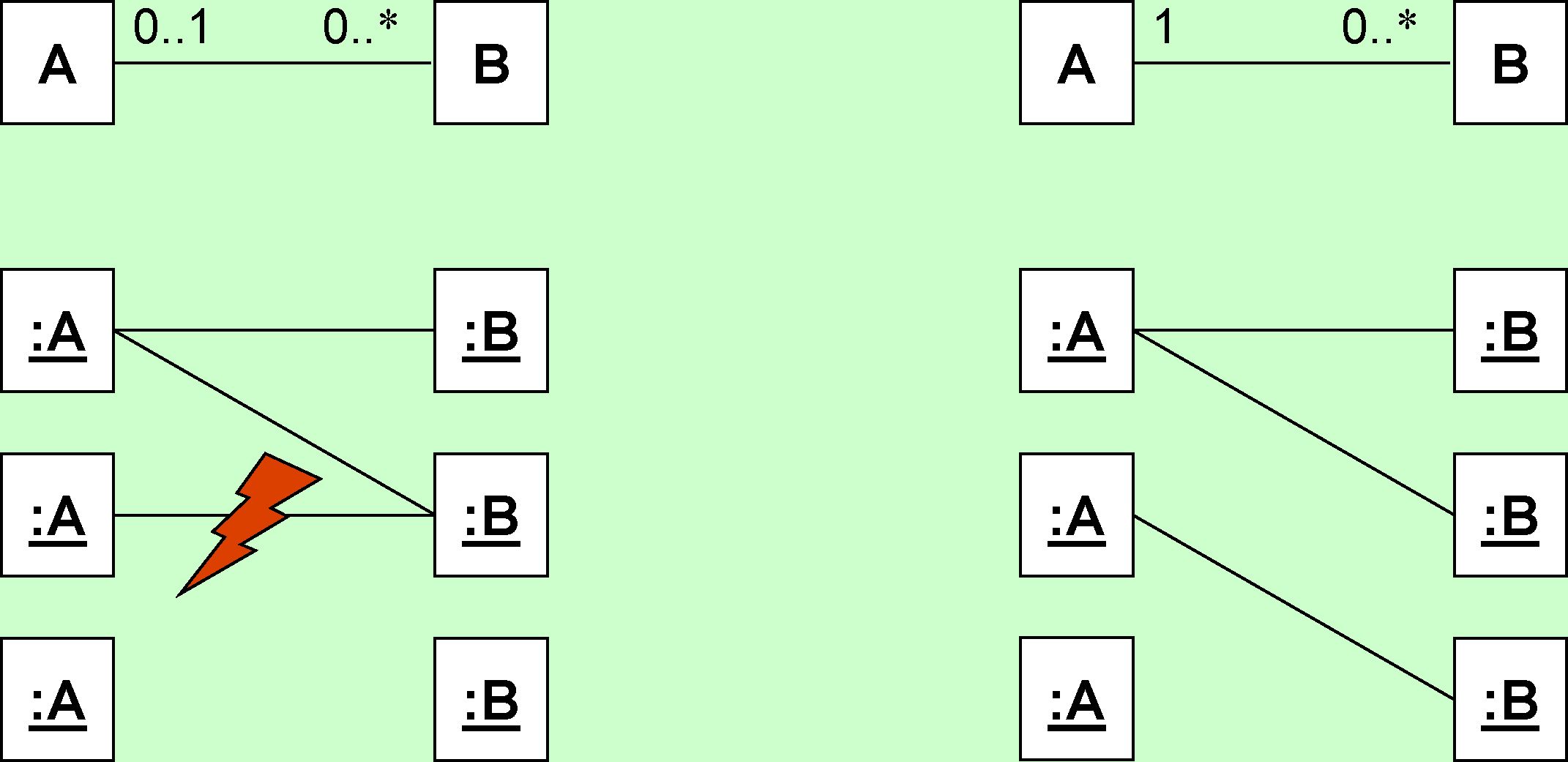
Functional



Functional Functional & Surjective



Injective Injective & Surjective

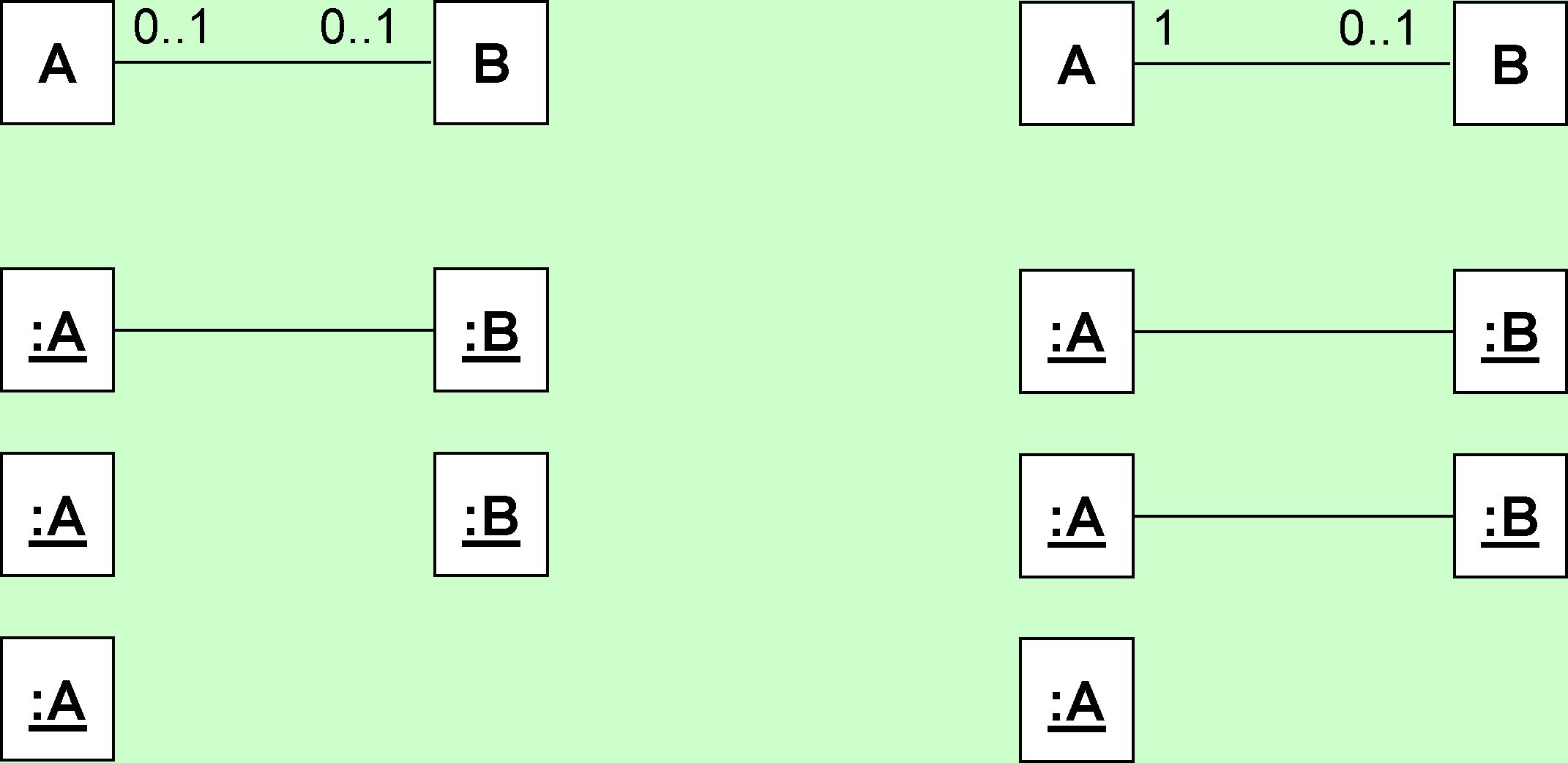


\*



Functional & Injective  
& Surjective

Functional &  
Injective

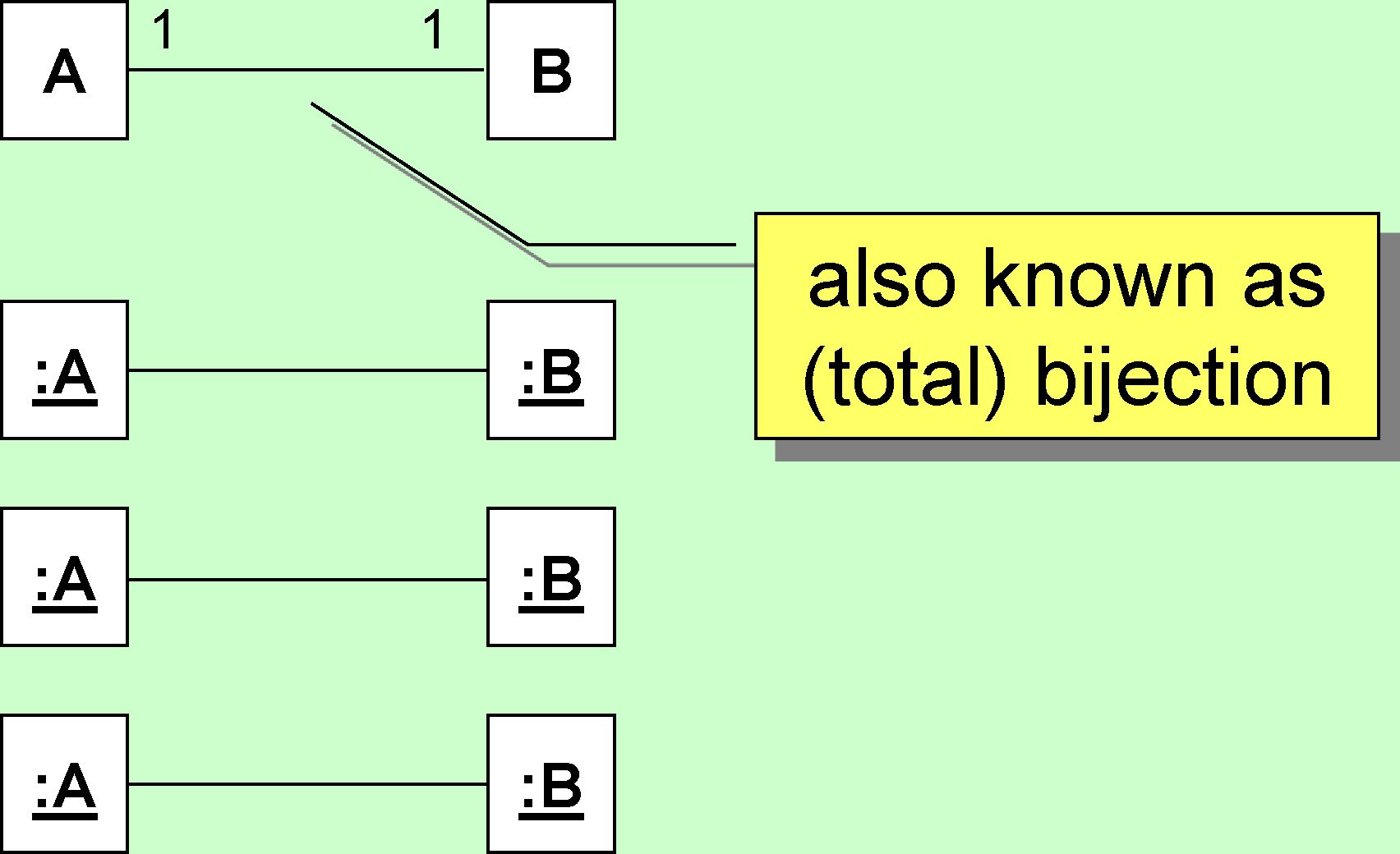


\*



Functional &

Injective & Surjective & Total



\*



Collection Types

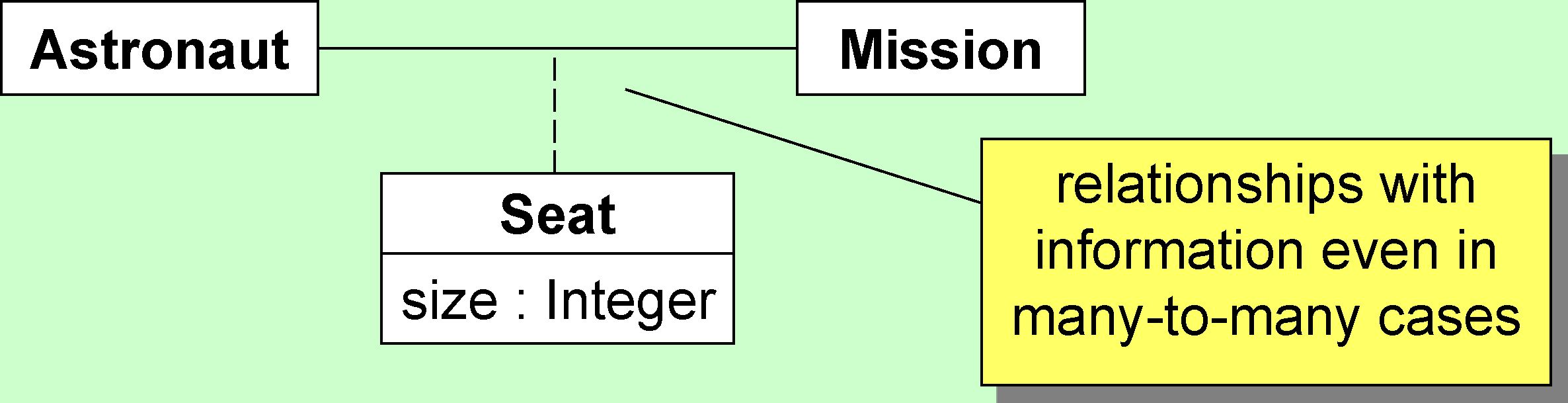


1 {readonly} has ► {ordered, unique} \*

|  |  |  |
| --- | --- | --- |
| Ordering | Uniqueness | Collection Type |
|  | unique | ?et |
| ordered | unique | OrderedSet |
|  |  | ?ag (aka Multiset) |
| ordered |  | ?equence (aka List) |

Association Classes

Avoiding Premature Assignment



• Capture information associated with relationships

» does not enhance the relationship with identity » no multiple relationships between two objects

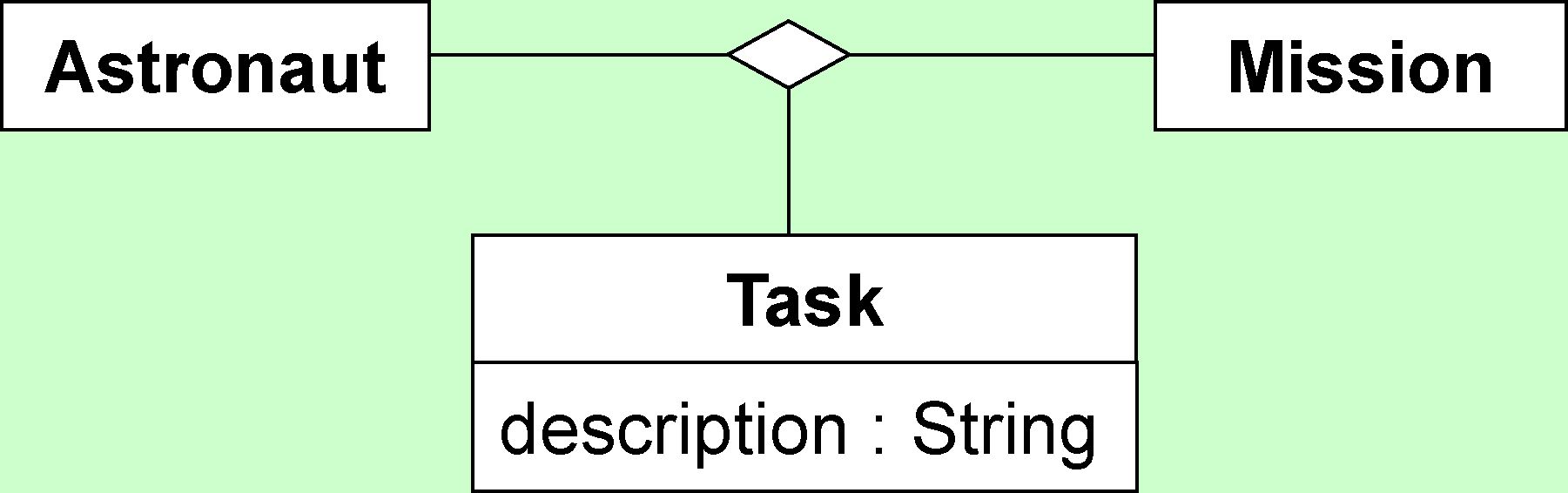
—i r\

Beyond Binary Associations

Higher Arity Associations

astronautHas

TaskOnMission



• Symmetric contribution to relationship

» useful, but rare

» often replaced by classes, storing additional information

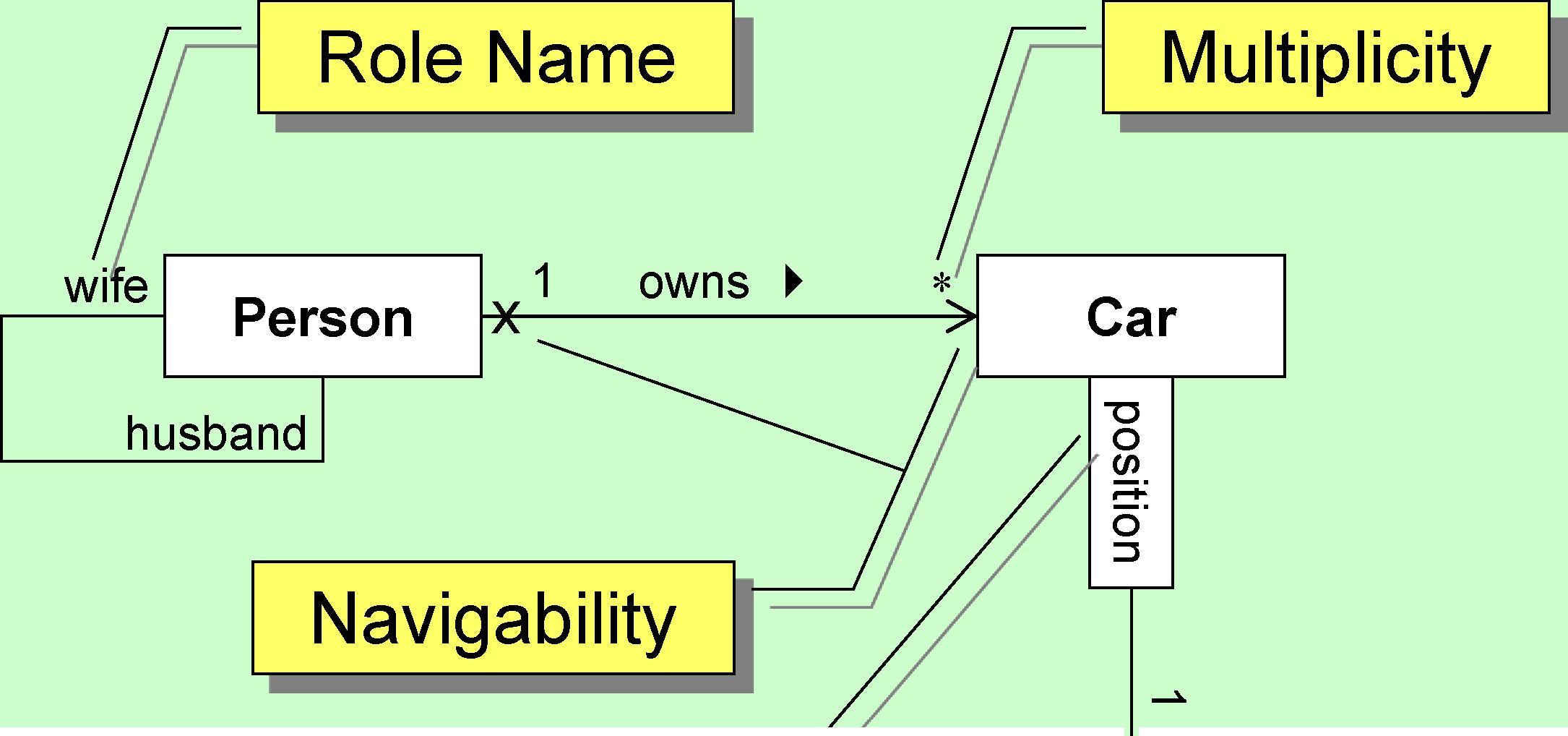
—i r\

"O

0

CD

E



|  |  |  |
| --- | --- | --- |
|  |  | Wheel |
| Qualified  Association |  |
|  | |



Roles

Named Association Ends

|  |  |  |
| --- | --- | --- |
|  | \* crewMember |  |
| Astronaut |  | Mission |
| 1 commander |

* Explain the role of a concept in a relationship

» one concept may have several roles in different contexts

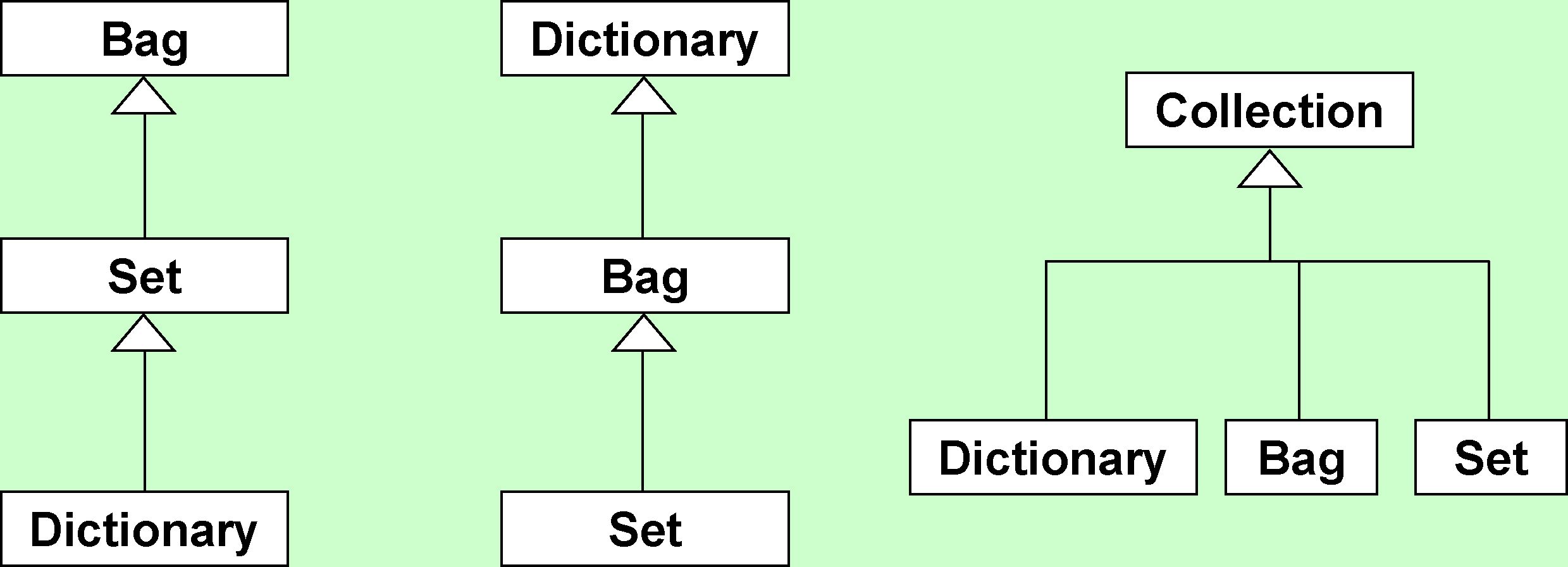
* Disambiguate multiple relationships

» one object can have multiple roles

—i r\

Is-a

Reuse Subtyping



\*



Is-A Reuse

w.r.t. concepts w.r.t. defintions

Subtyping

w.r.t. objects

Bag

Dictionary

|  |  |
| --- | --- |
| Set "1 |  |
| ^ {0-1-Bag} J | Bag |
| z\ | numberOf(elem) |
|  |  |
| Dictionary |  |

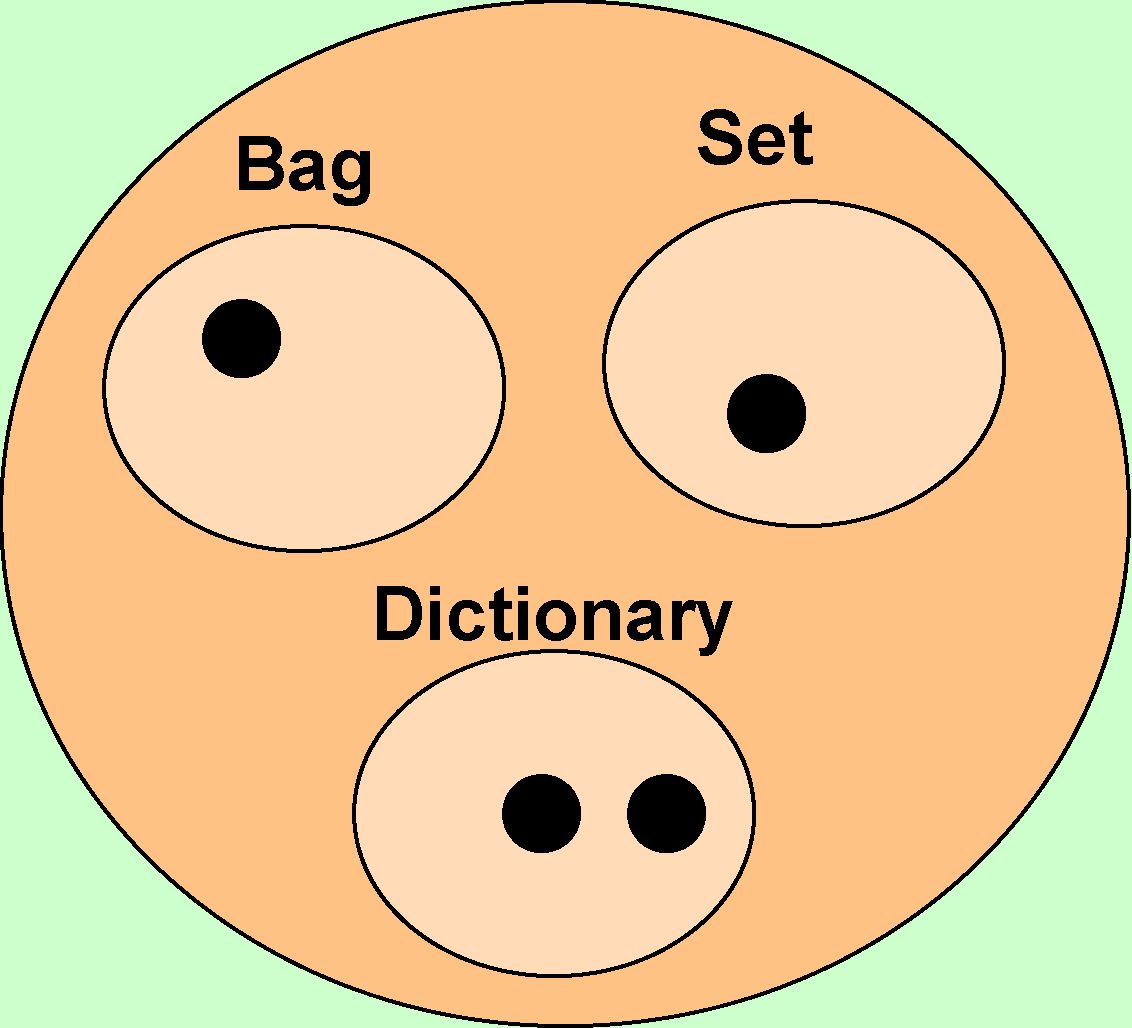
{Key-Value-Set}

valueFor(key)

Set

includes(elem)

Collection



Generalisation

Used for

* Classification (is-a)

» System understanding

* Code Reuse *(subclassing)*

» division between common and specialized code » easy library creation

* Substitution principle (subtyping)

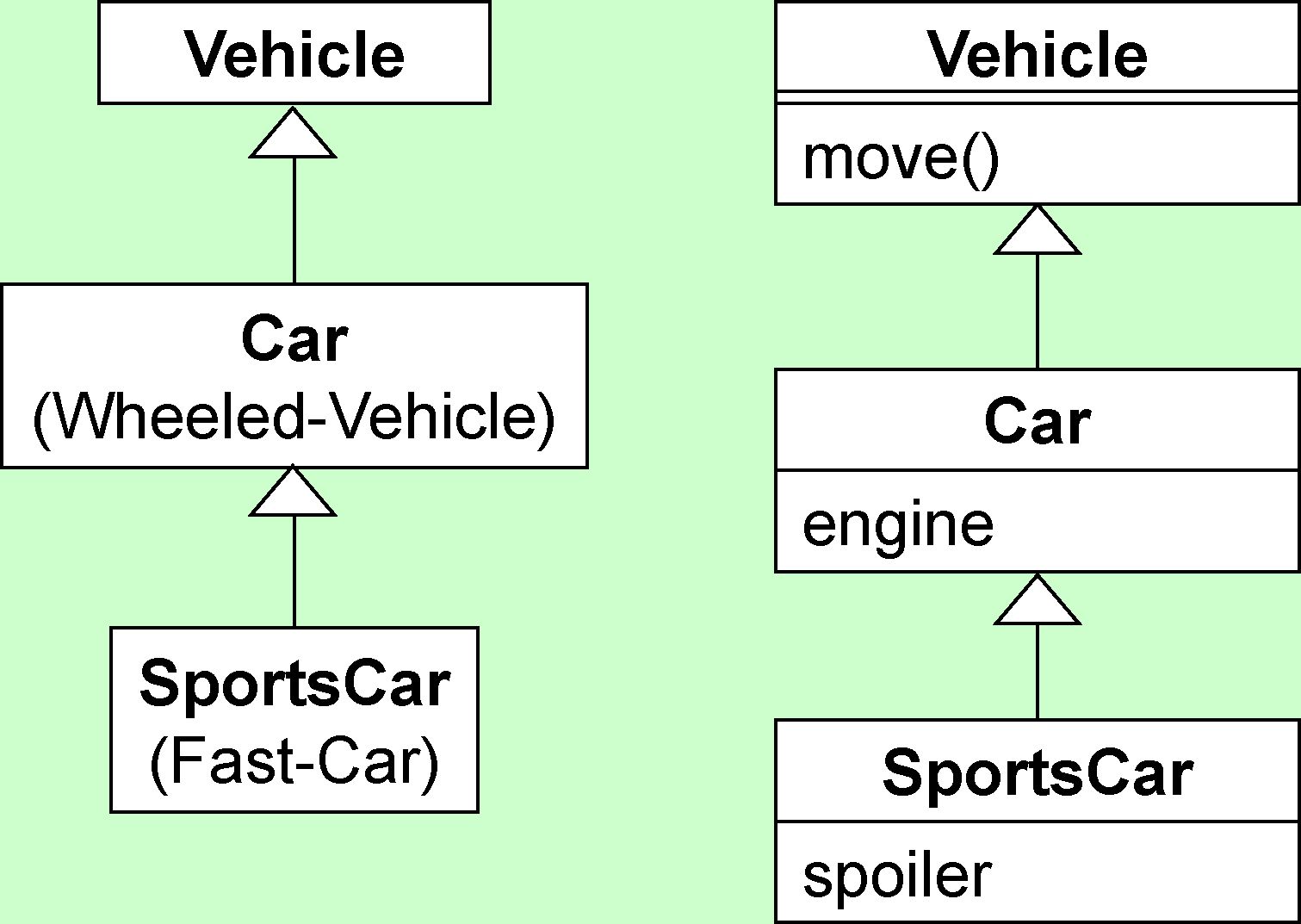
» behavioural equality with extensions » easy library usage

—i

[

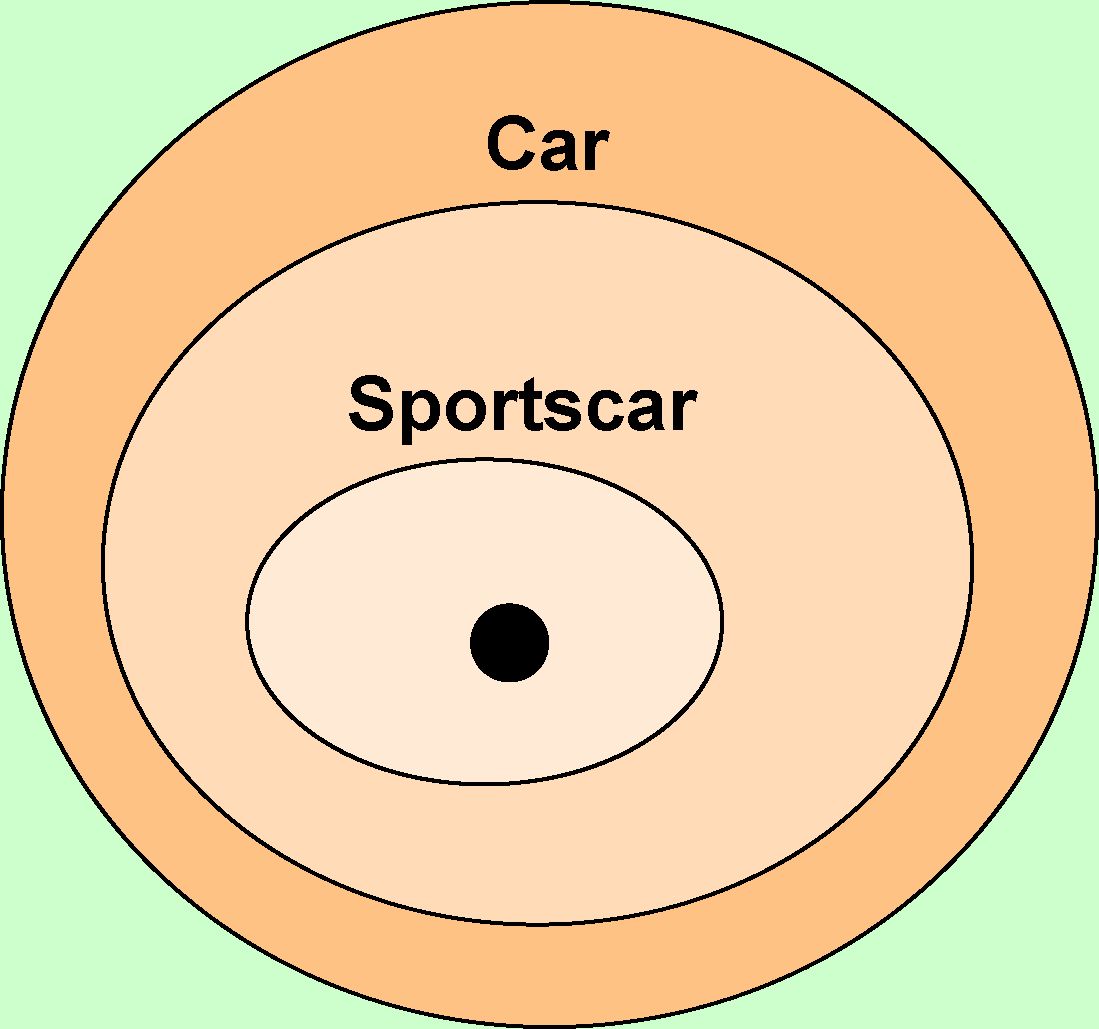
Is-A

Reuse



Subtyping

Vehicle



\*



Class Diagram

Room

Dining room

Kitchen

Person

ordered by

cooked by ►

1

Order

1 {ordered} \*

\*

creation

Dish O

\*

Item

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| \* |  |  |  | empty |  |  | |  |  | |  |
| Table | | \* |  | waiting | Waiter | |  | Client | | Cook | |
|  | |  |  | served |  | - | |  |  | o |  |

person

E

o

o

served by

CO

CD

O

style

