SWEN 223
Software Engineering Analysis

UML Class Diagrams

Thomas Kühne
Victoria University of Wellington
Thomas.Kuehne@ecs.vuw.ac.nz, Ext. 5443, Room Cotton 233

Class Specification

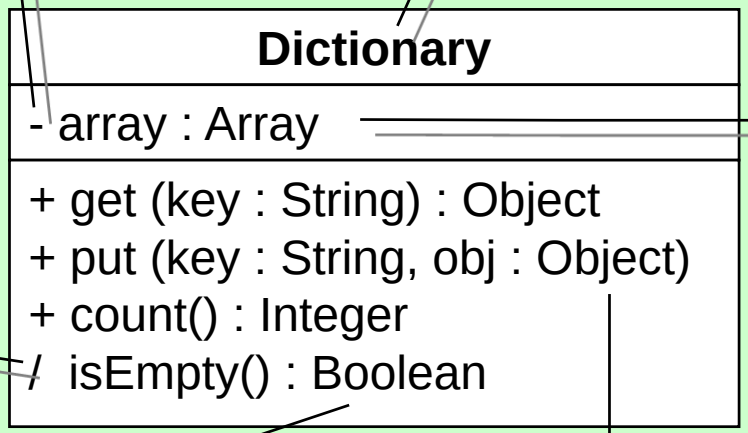
Visibilty

+ public
# protected
~ package
- private

(exact meaning is a semantic variation point)

Class Name

describes what instances are



Attribute

less important than operations, in particular in the analysis phase

Derived

redundant but handy information

Result Type

(notation is up to the user)

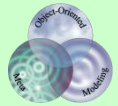
Argument Type

(notation is up to the user)

Attributes

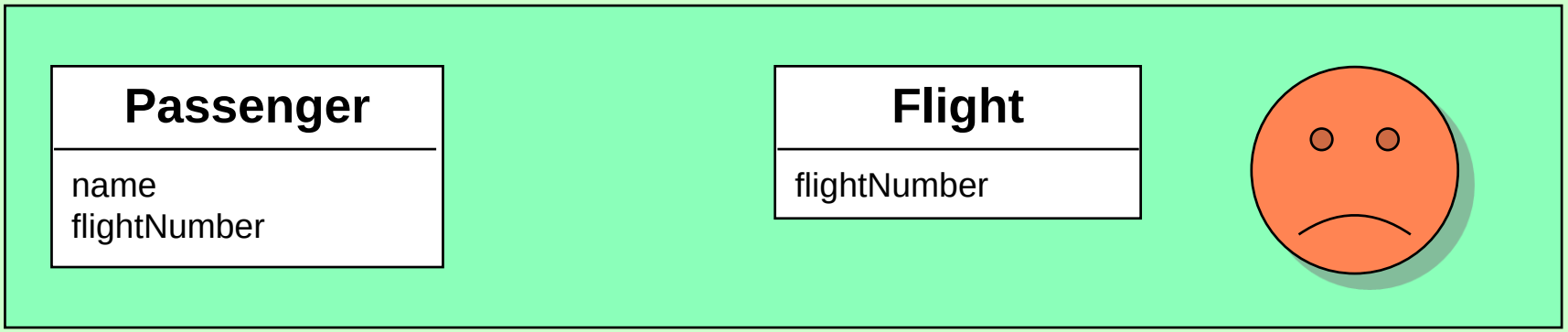
What Information Should be Captured by Attributes?

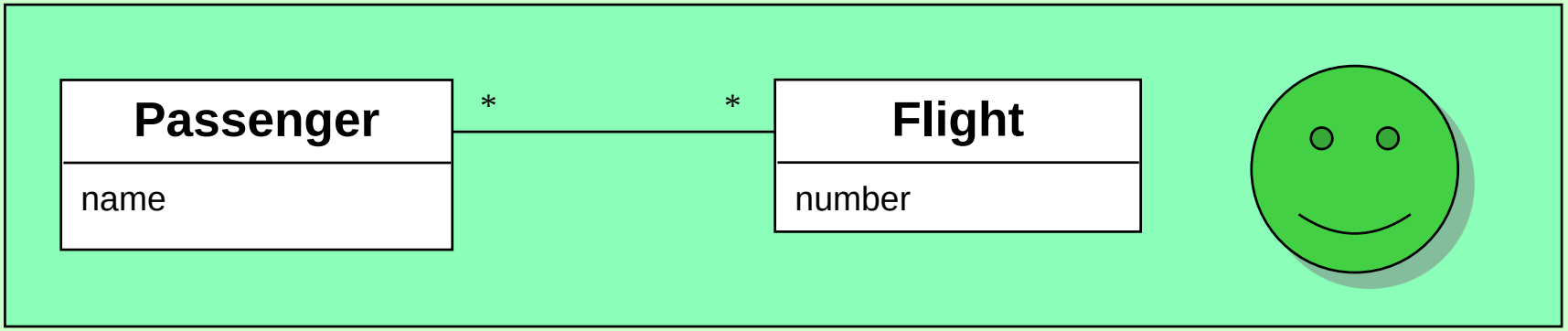
- Attribute types will typically be Datatypes
  - » values for which unique identity is not useful
  - » e.g. not usually meaningful to distinguish between
    - instances of the number 5, or the string “cat” (all primitive types are datatypes)
    - instances of PhoneNumber that contain the same number
    - etc.



Attributes

Do Not Use Attributes as Foreign Keys





Associations

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

Association Examples

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

Association Examples

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

Associations

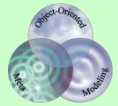
Design Phase

- 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

May result from

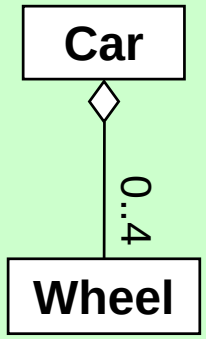
structural
- instance variables

temporal
- message arguments
- message results

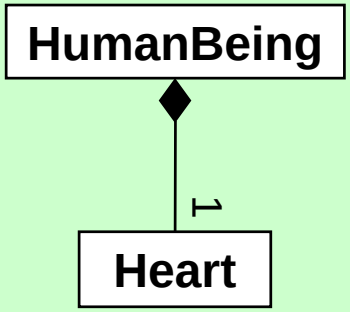


Aggregation vs Composition

- Aggregation (white diamond)
  - » regular association with whole-part connotation (anti-symmetric, transitive)
  - » no additional semantics attached
  - » if in doubt, use a regular association

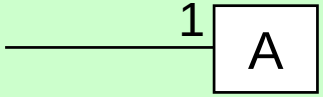


- Composition (black diamond)
  - » parts cannot exist without the whole
  - » synchronises lifetimes (transitively)

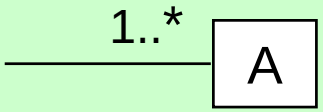


Multiplicities

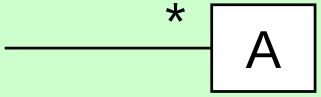
Some Standard Cases



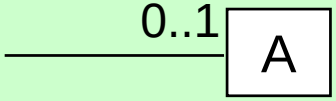
exactly one



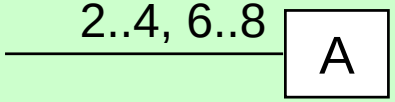
mandatory (one or more)



many (zero or more)



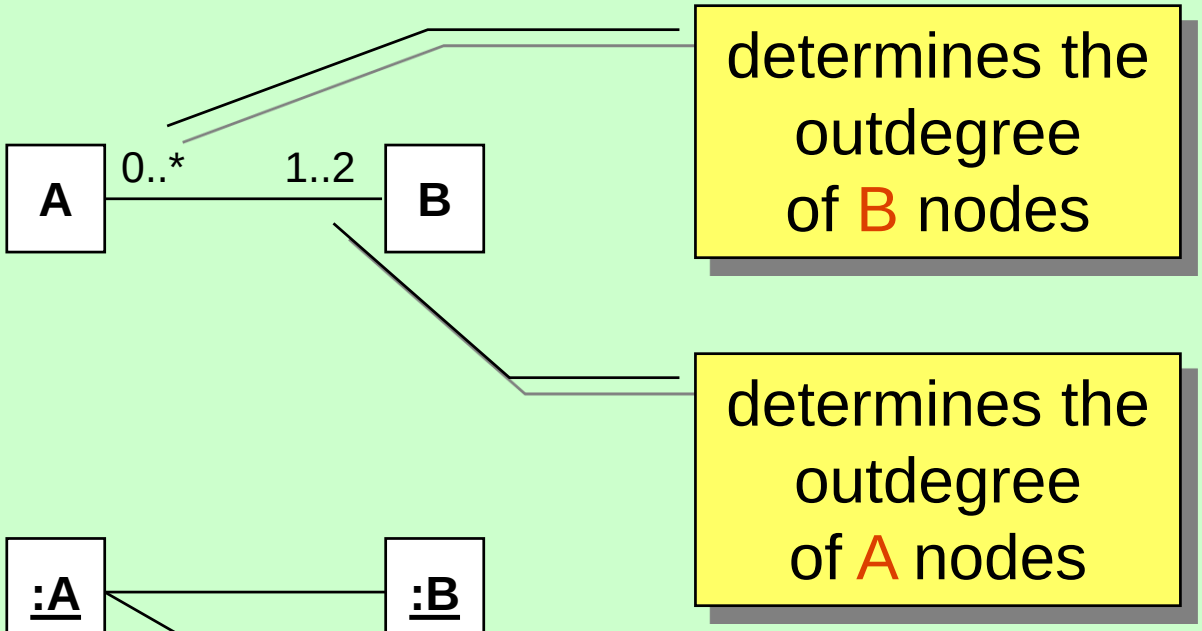
optional (zero or one)



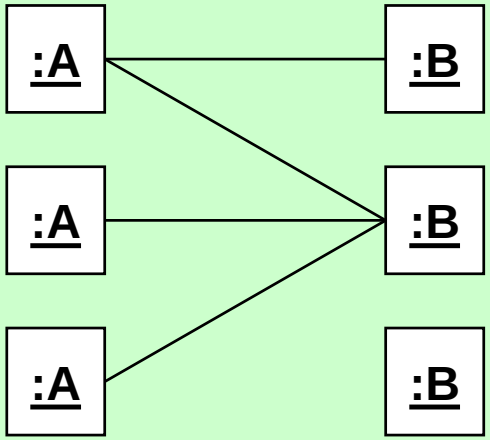
numerically specified

Multiplicities

Class Level

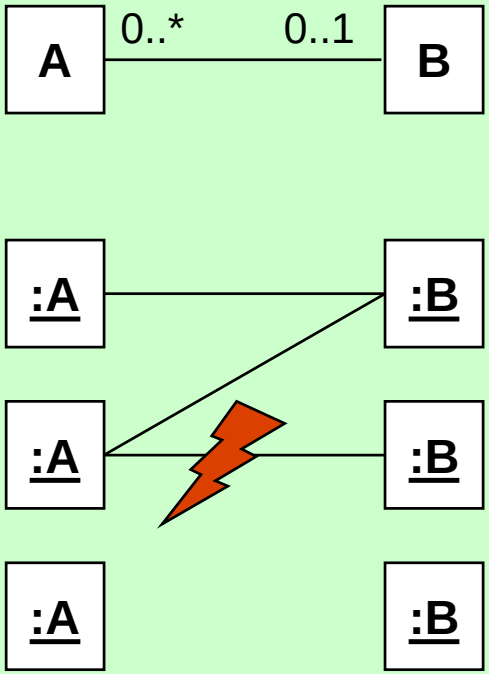


Object Level

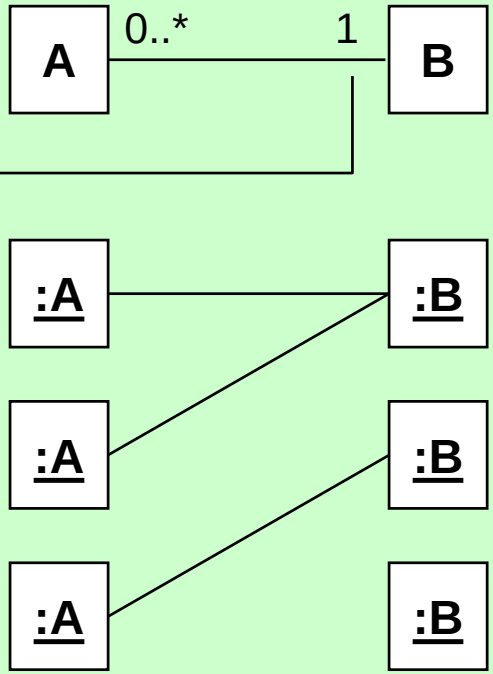


Multiplicities

Functional



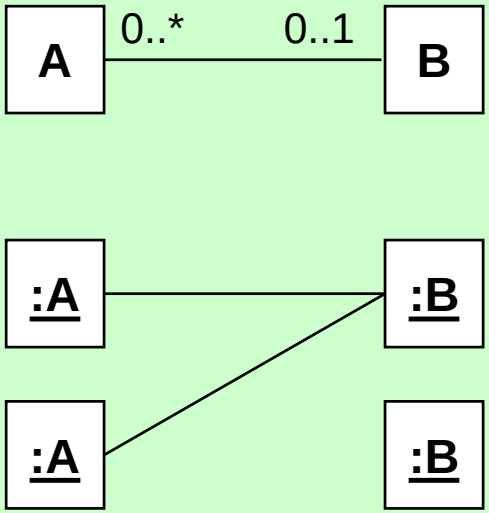
Functional & Total



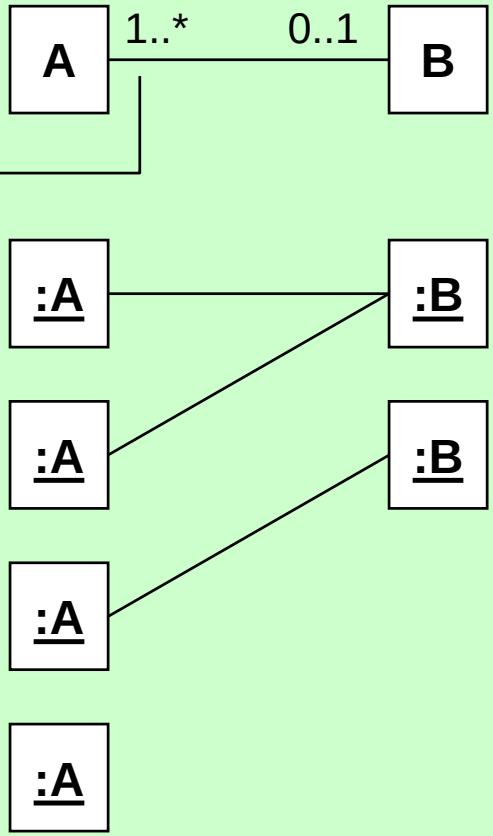
there cannot be an A without a B

Multiplicities

Functional



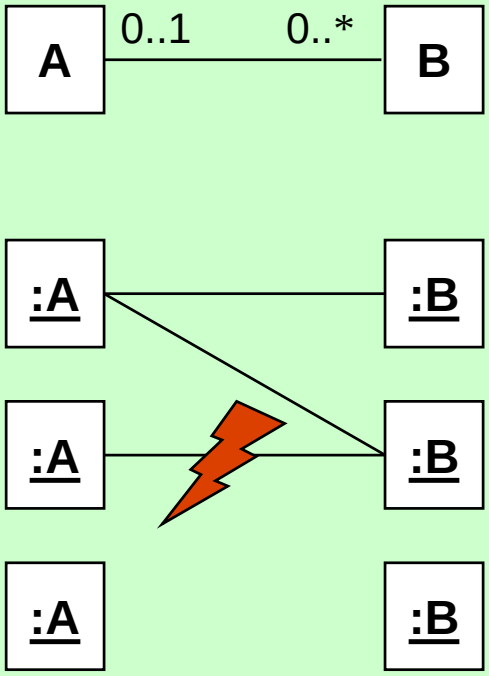
Functional & Surjective



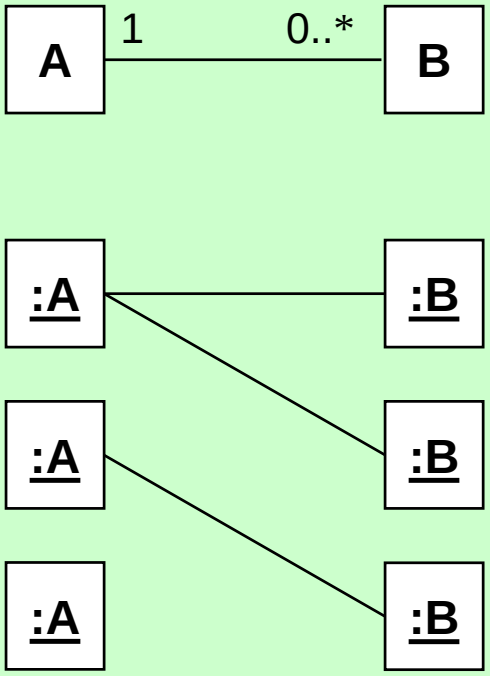
there cannot be a B without an A

Multiplicities

Injective

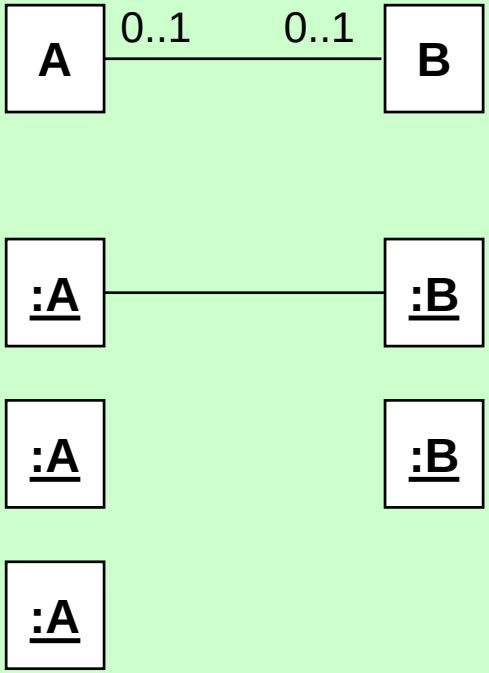


Injective & Surjective

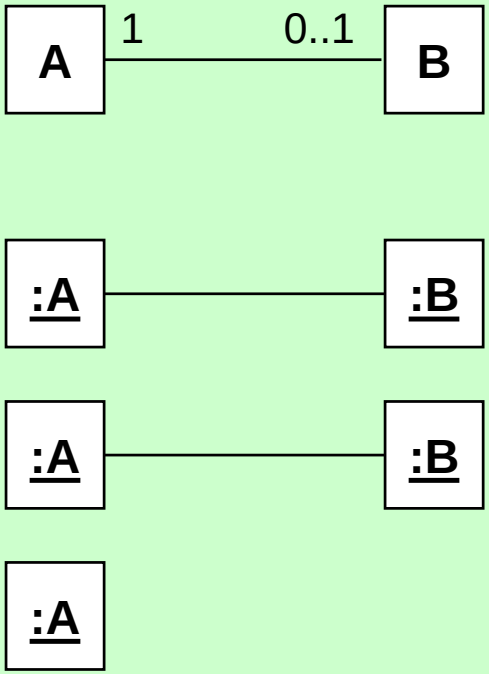


Multiplicities

Functional & Injective

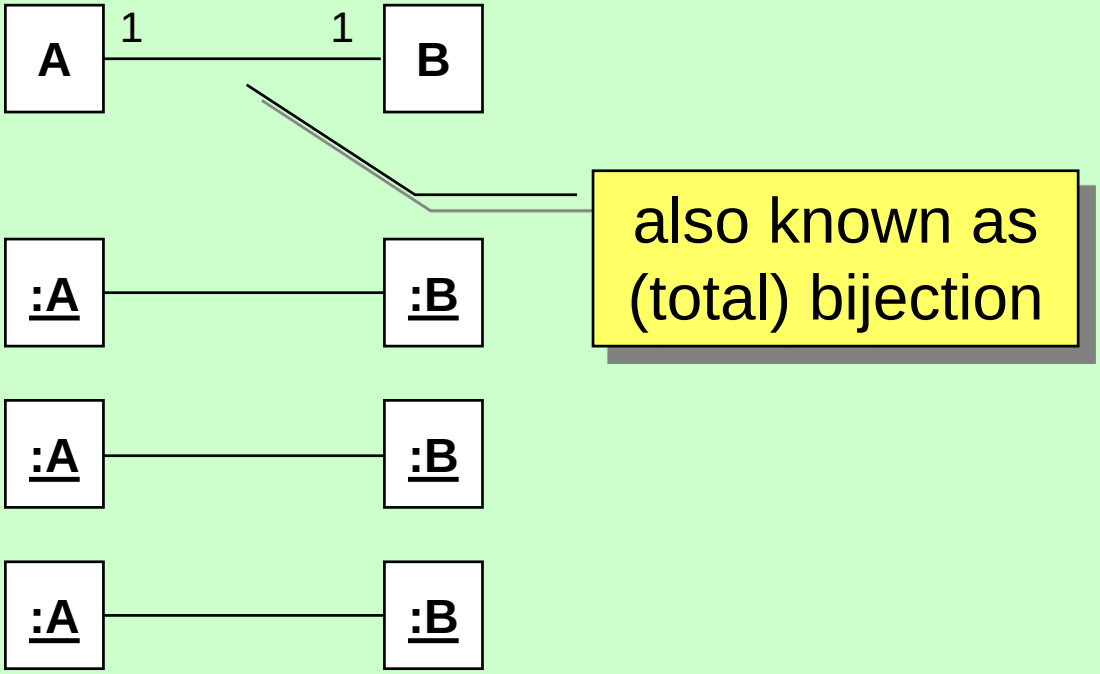


Functional & Injective & Surjective

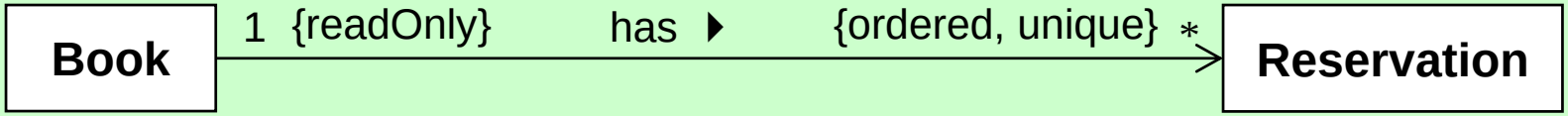


Multiplicities

Functional & Injective & Surjective & Total



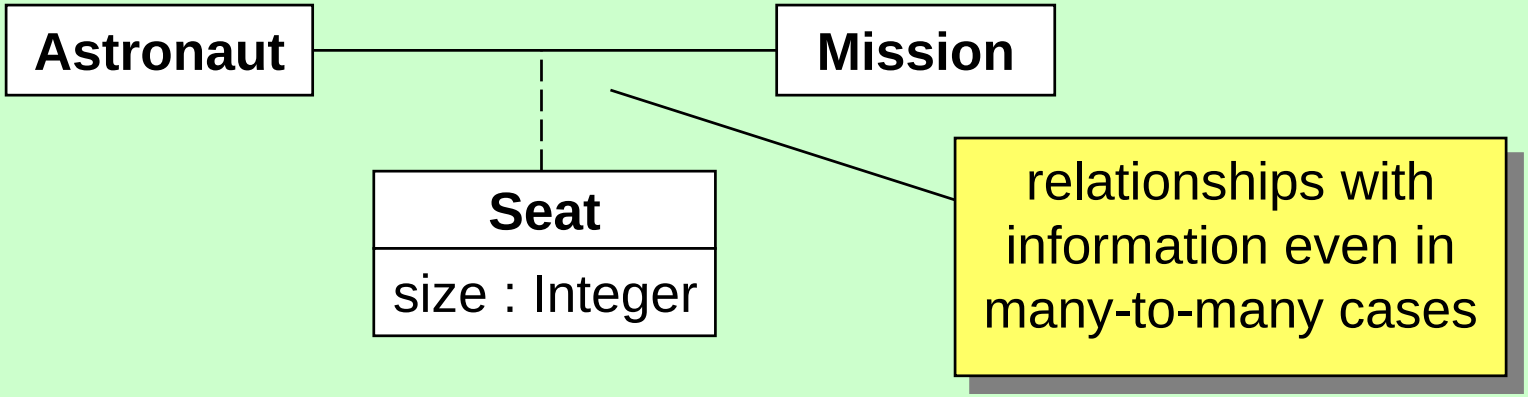
Collection Types



Ordering	Uniqueness	Collection Type
	unique	Set
ordered	unique	OrderedSet
		Bag (aka MultiSet)
ordered		Sequence (aka List)

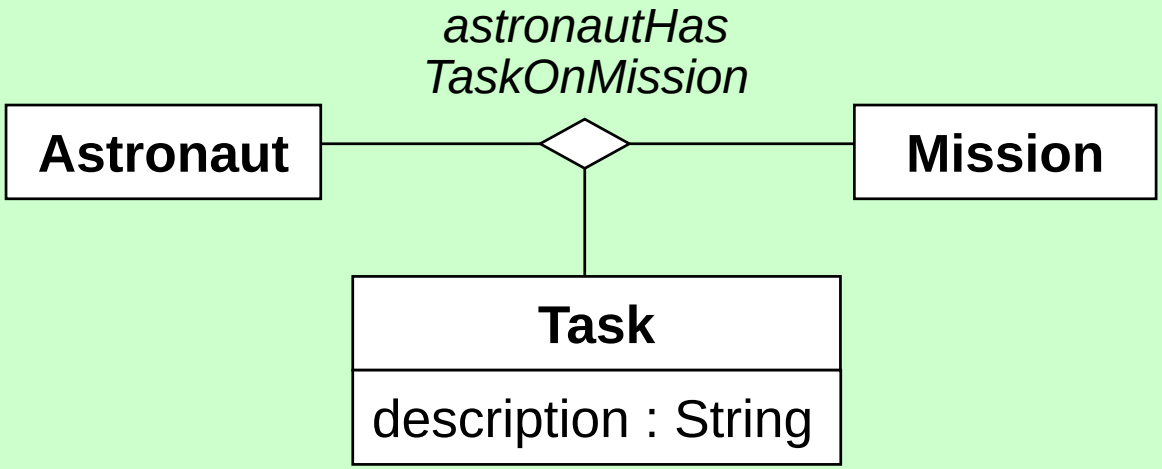
Association Classes

Avoiding Premature Assignment



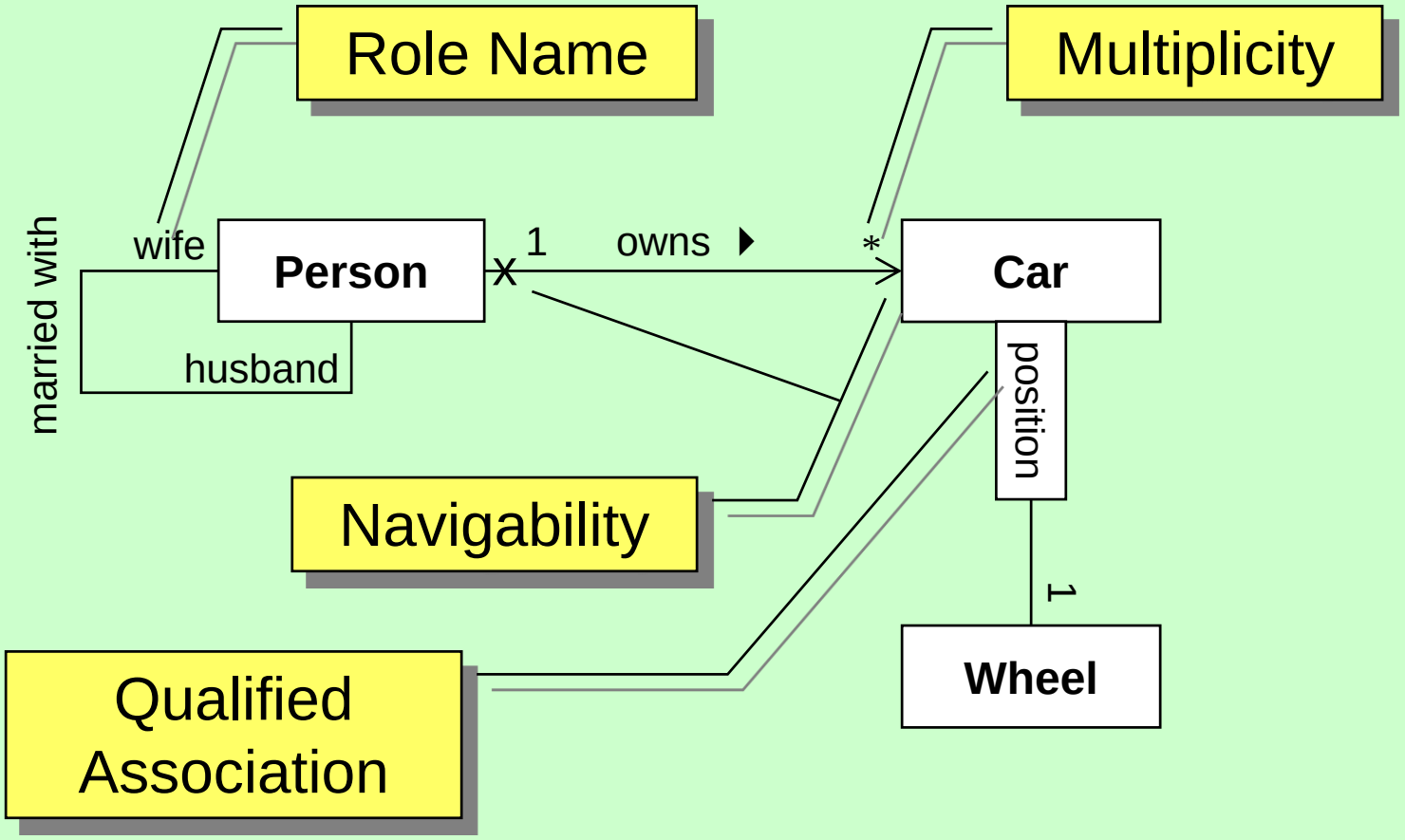
- Capture information associated with relationships
  - » does not enhance the relationship with identity
  - » no multiple relationships between two objects

Higher Arity Associations



- Symmetric contribution to relationship
  - » useful, but rare
  - » often replaced by classes, storing additional information

Advanced Relationships

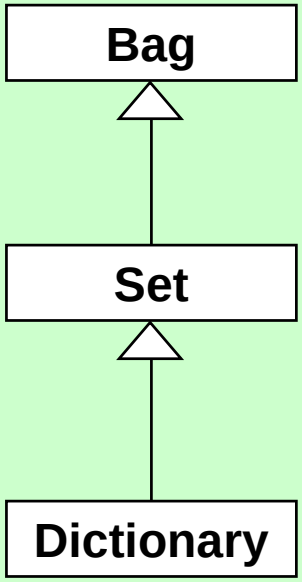


Named Association Ends

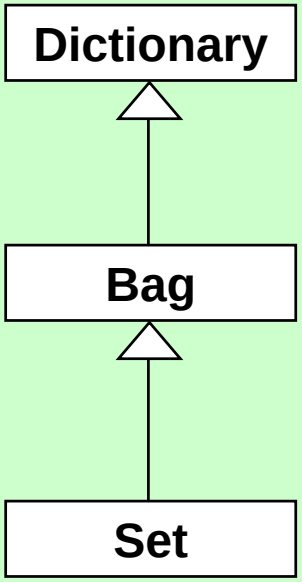
- Explain the role of a concept in a relationship
  - » one concept may have several roles in different contexts
- Disambiguate multiple relationships
  - » one object can have multiple roles

Competing Views

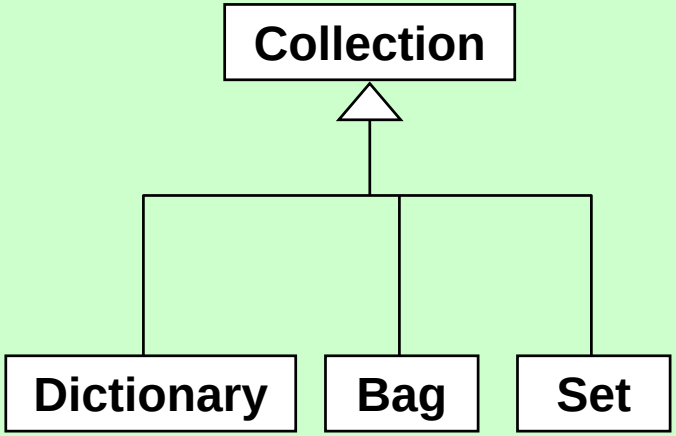
Is-a



Reuse

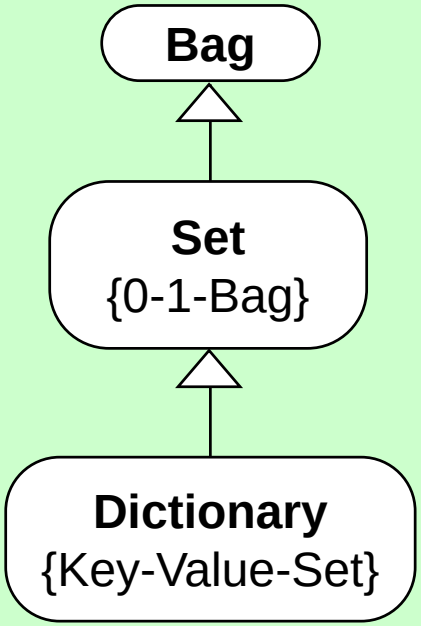


Subtyping

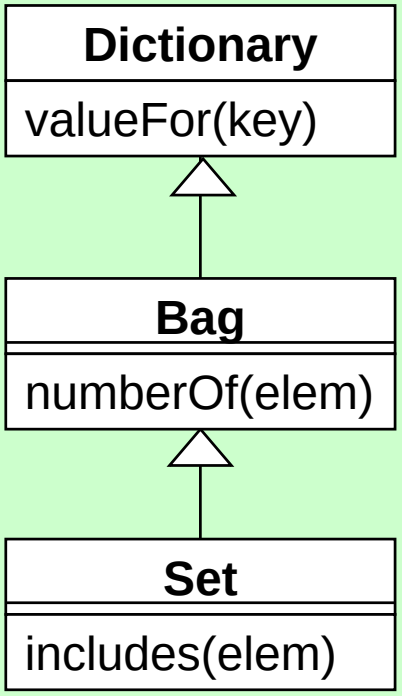


Competing Views

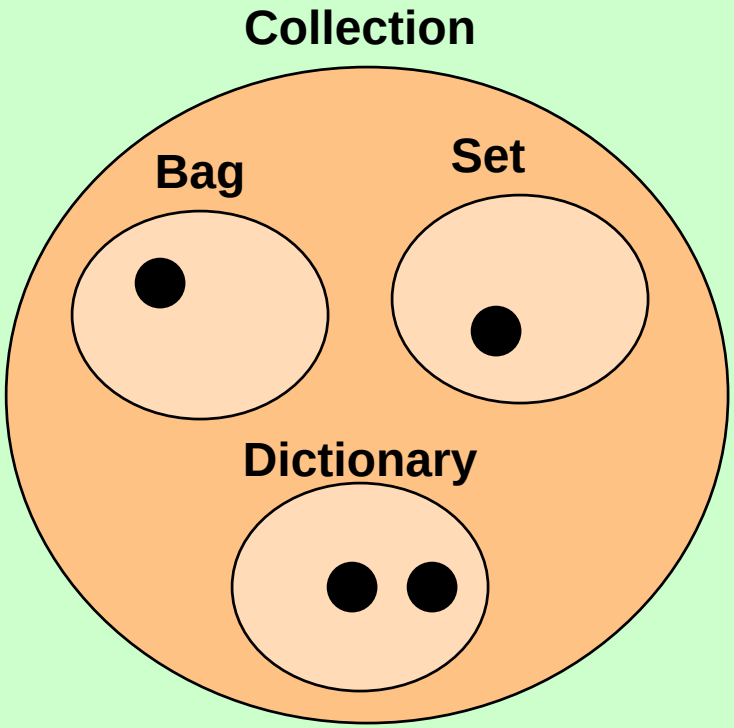
Is-A
w.r.t. concepts



Reuse
w.r.t. defintions



Subtyping
w.r.t. objects



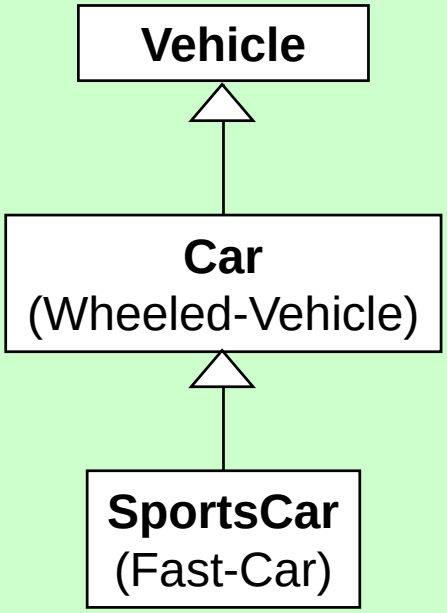
Generalisation

Used for

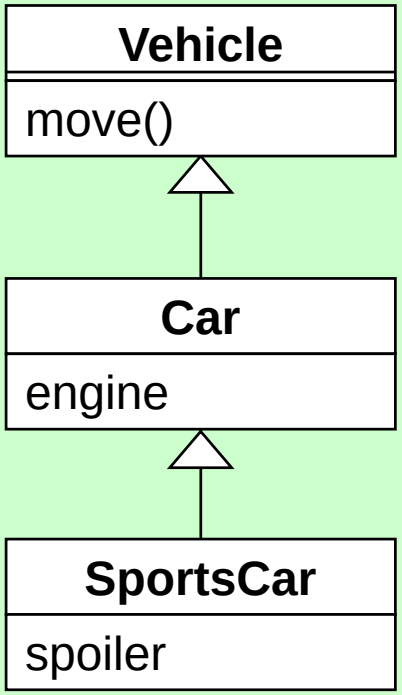
- Classification (*is-a*)
  - » System understanding
- Code Reuse (*subclassing*)
  - » division between common and specialized code
  - » easy library creation
- Substitution principle (*subtyping*)
  - » behavioural equality with extensions
  - » easy library usage

No Competition

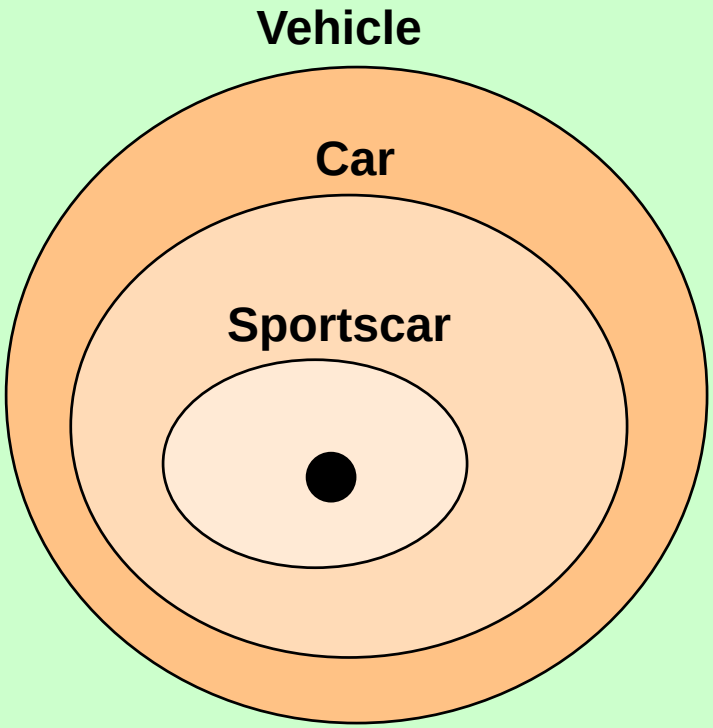
Is-A



Reuse



Subtyping



Class Diagram