SWEN 225
Software Design

UML Class Diagrams

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

Class Specification

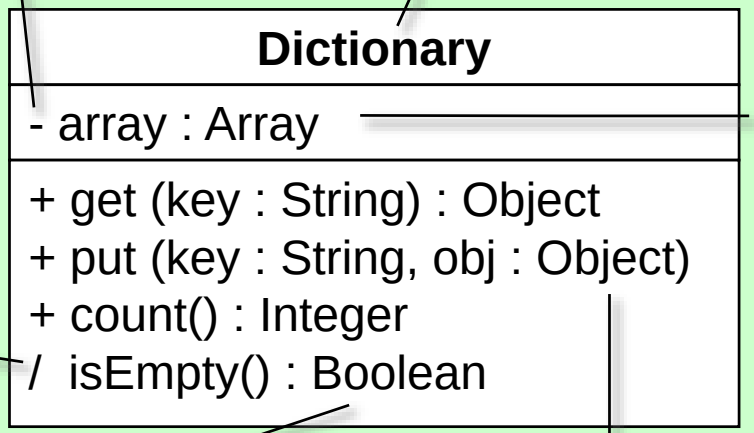
Visibilty

+ public
# protected
~ package
- private

(exact meaning is a semantic variation point)

Class Name

describes what instances are



Attribute

what data is the class responsible for?

Derived

redundant but handy information

Result Type

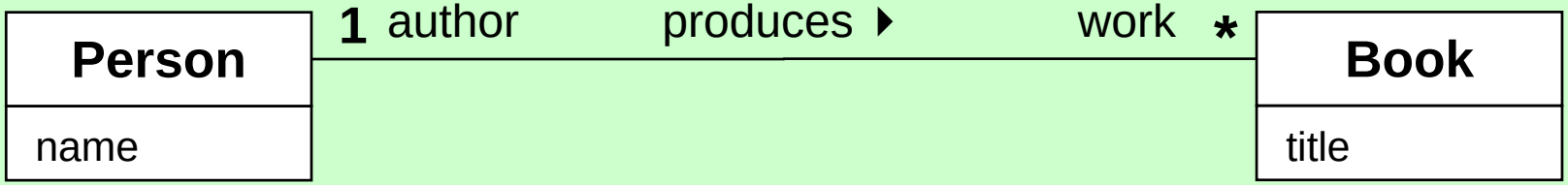
(notation is up to the user)

Argument Type

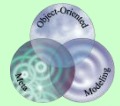
(notation is up to the user)

Associations

Enabling links between objects

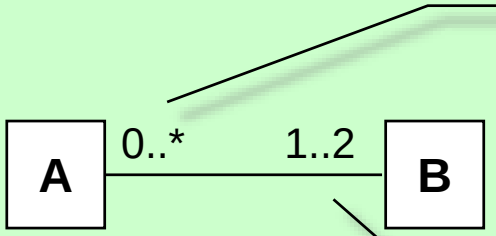


- Associations have names
  - » describe the relation
- Associations have role names
  - » describe the roles/functions of the participants



Multiplicities

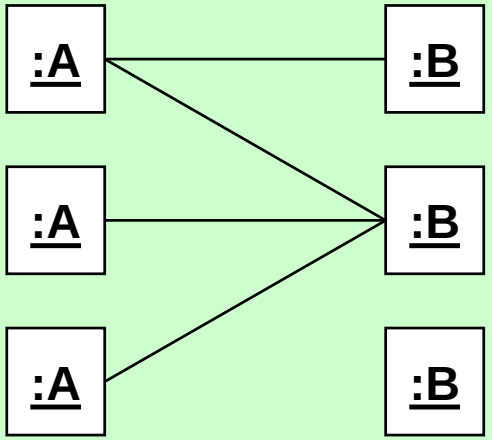
Class Level



determines the outdegree of B nodes

determines the outdegree of A nodes

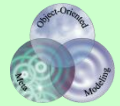
Object Level



Role Names

Named Association Ends

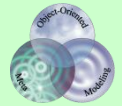
- Explain the role of a concept in a relationship
  - » one concept may have several roles in different contexts
- Disambiguate multiple relationships
  - » one object can have multiple roles



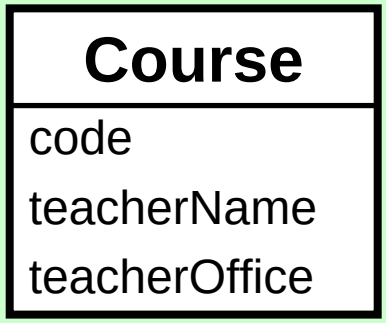
Attributes

What Information Should be Captured by Attributes?

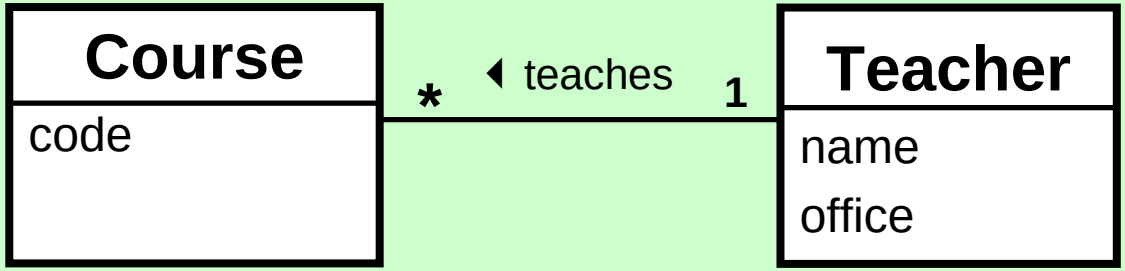
- Attribute types will typically be Datatypes
  - » values for which unique identity is not useful
  - » e.g. not usually meaningful to distinguish between
    - – instances of the number 5, or the string “cat” (all primitive Java types are datatypes)
    - – instances of PhoneNumber that contain the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

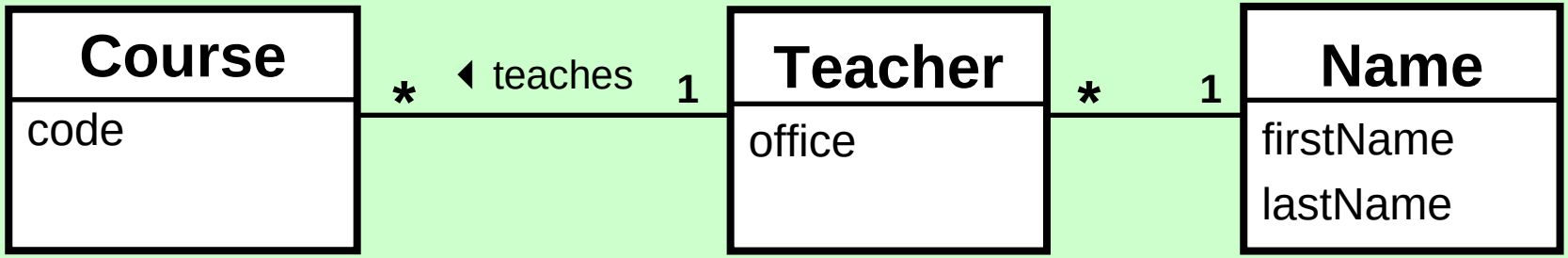


Attributes vs Associations



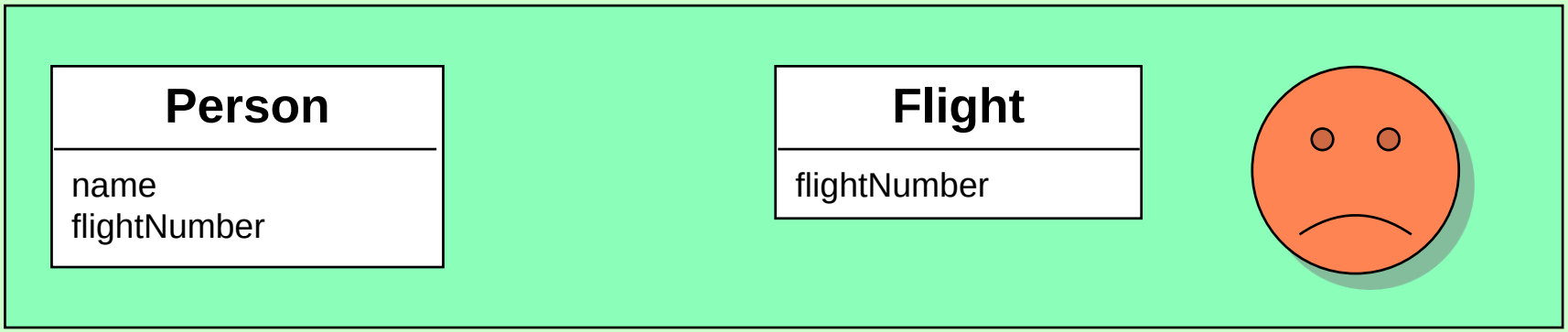
Which is Best?

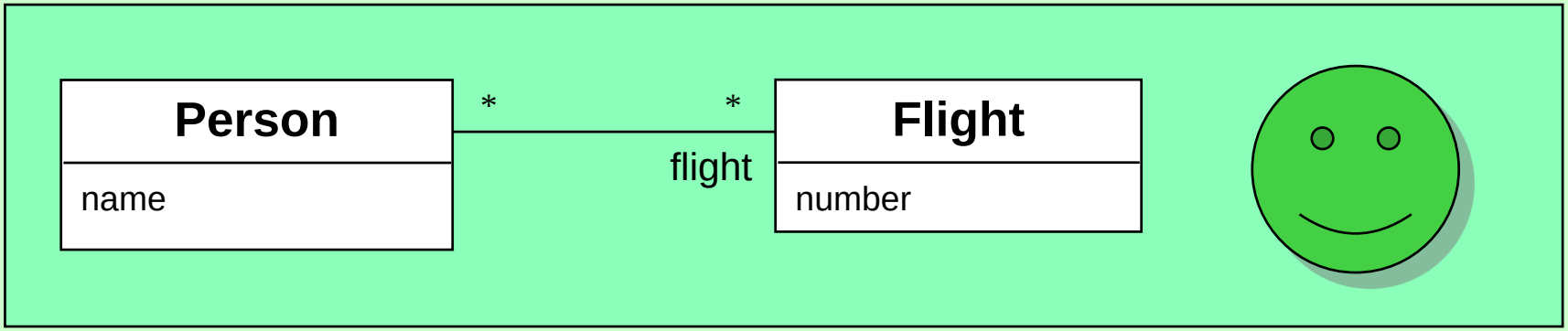




Attributes

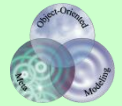
Do Not Use Attributes as Foreign Keys





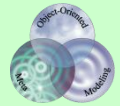
Association Examples

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



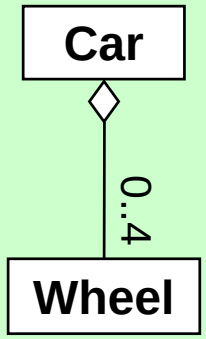
Association Examples

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

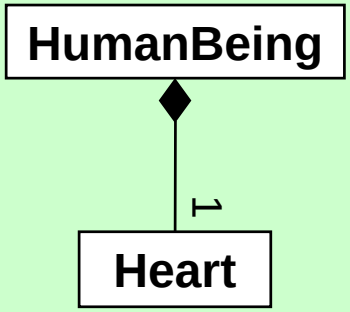


Aggregation vs Composition

- 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)



Associations

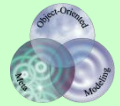
Design Phase

- 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

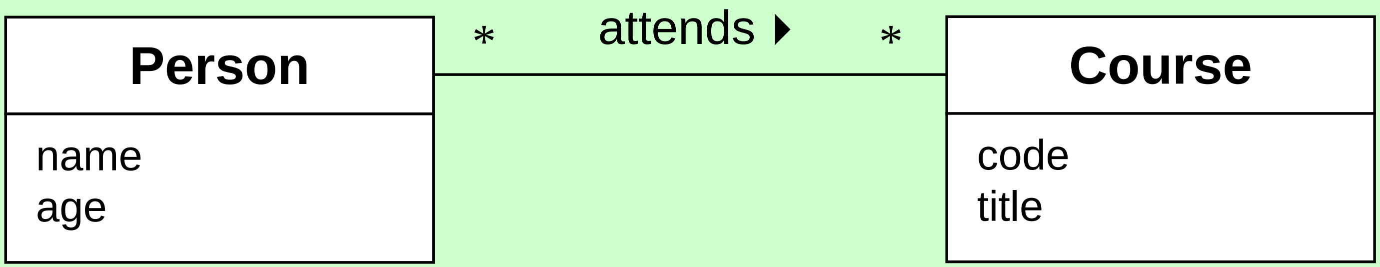
May result from

structural
- instance variables

temporal
- message arguments
- message results



Implementing UML Associations



- **Many** students attend **Many** courses
  - » How do we implement this in Java?

Implementation #1

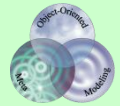
```
class Person {
    …
    List<Course> attending;

    void enroll(Course c) {
        attending.add(c);
    }
    …
}
```

```
class Course {
    …
    List<Person> attendees;

    void enroll(Person p) {
        attendees.add(p);
    }
    …
}
```

- Internal Collections
  - » supports traversal in both directions
  - » do you see a potential issue?

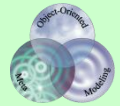


Implementation #2

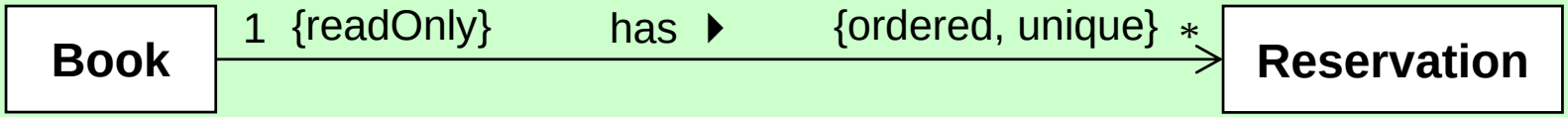
```
class Attends {
    Map<Person, List<Course>> attending;
    Map<Course, List<Person>> attendees;

    void enroll(Person p, Course c) {
        attendending.get(p).add(c);
        attendees.get(c).add(p);
    }
    …
}
```

- External Collection
  - » **Person** and **Course** are unaware of each other (decoupled)
  - » easy to support multiple associations per pair
  - » potential memory benefits



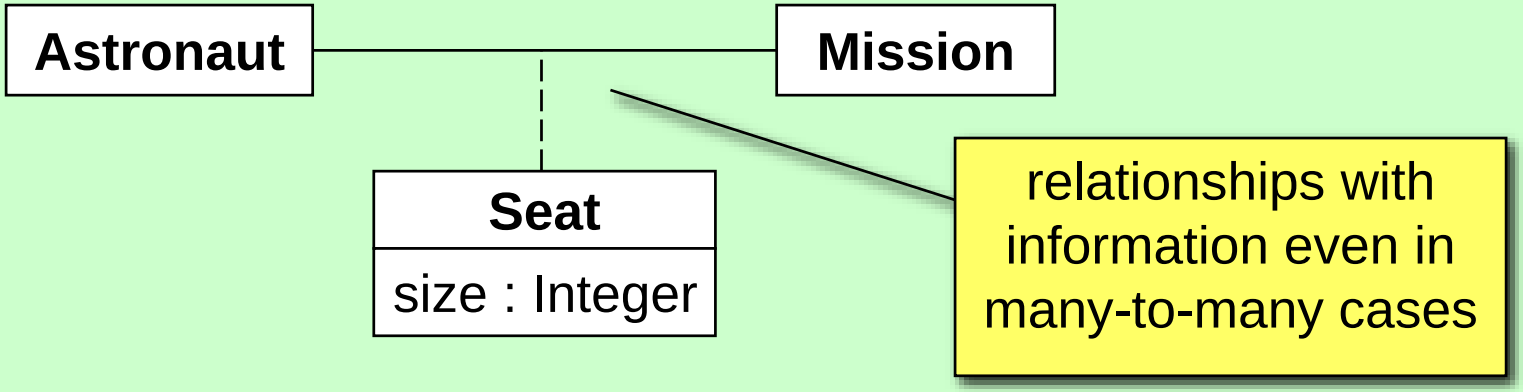
Collection Types



Ordering	Uniqueness	Collection Type
	unique	?
ordered	unique	?
		?
ordered		?

Association Classes

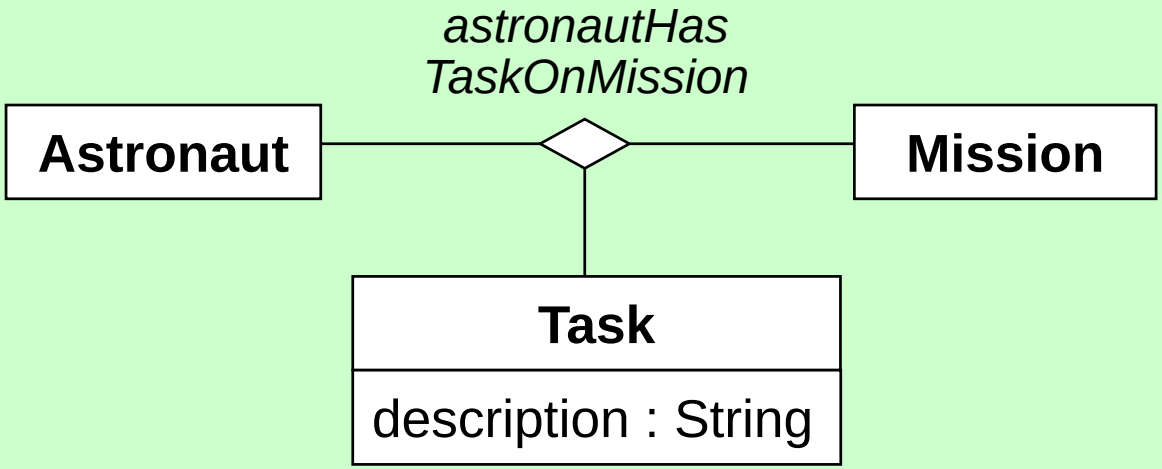
Avoiding Premature Assignment



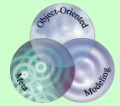
- Capture information associated with relationships
  - » does not enhance the relationship with identity; no multiple relationships between two objects

Beyond Binary Associations

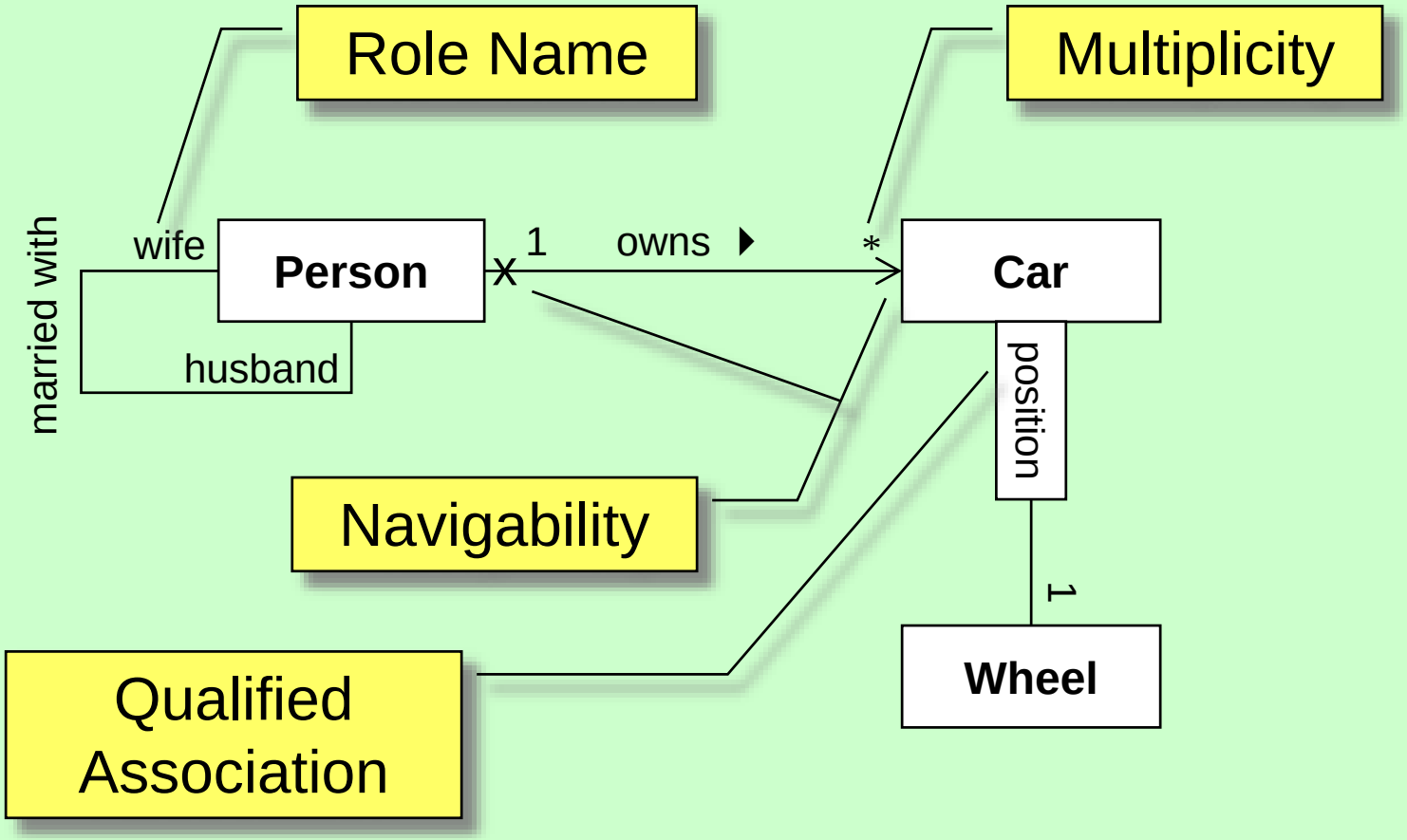
Higher Arity Associations



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



Advanced Relationships



Class Diagram