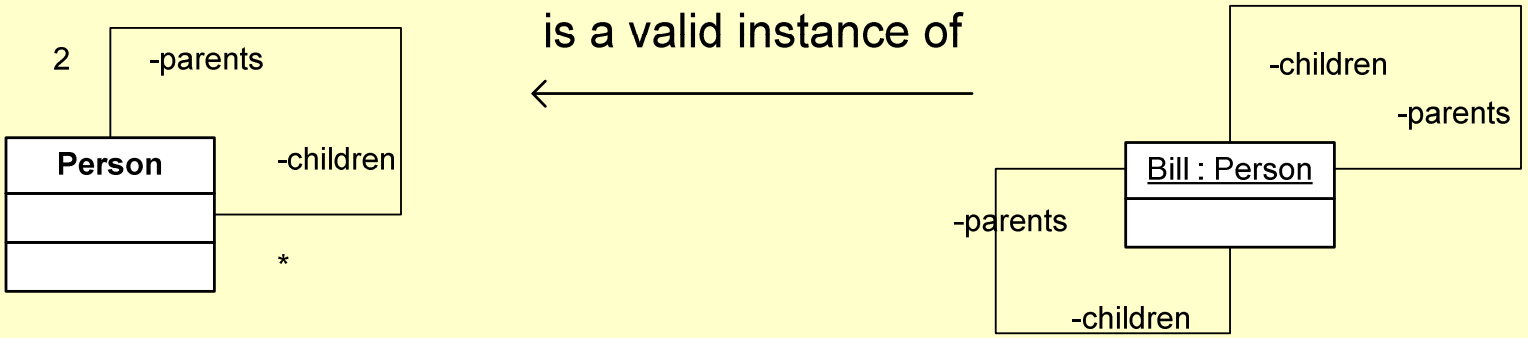
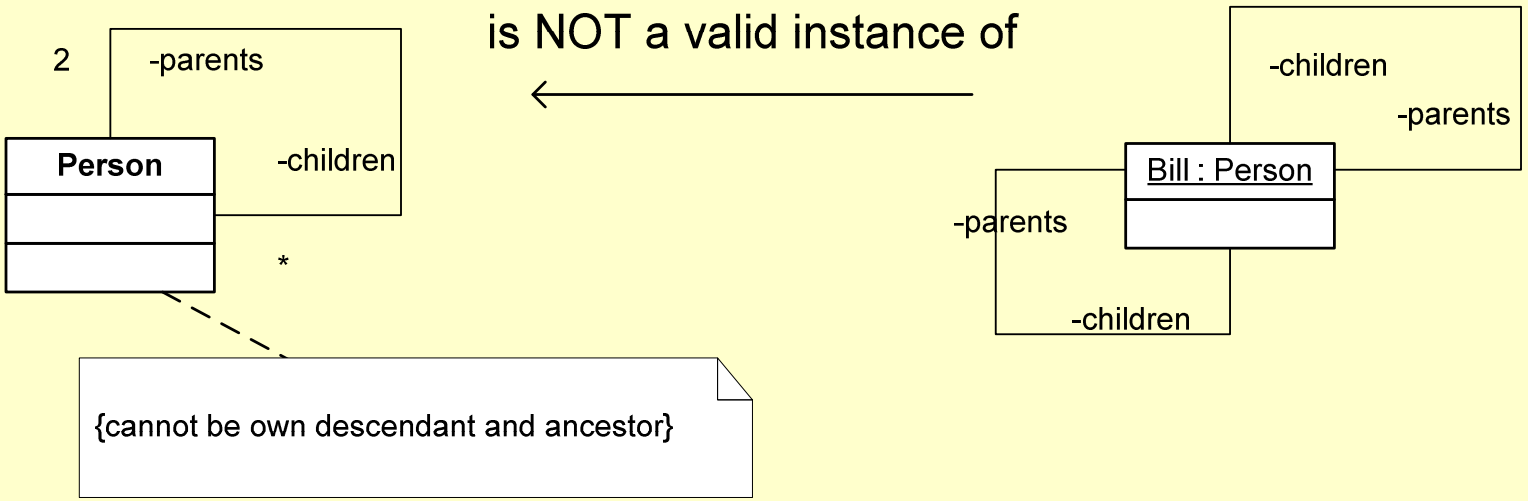
Object Constraint Language

OCL 2.0

UML does not tell us everything!



Enter constraints



What is OCL?

- A language to express constraints in our UML models
- A precise and unambiguous language that can be read and understood by developers and customers
- A purely declarative language: it describes *what* and not *how*

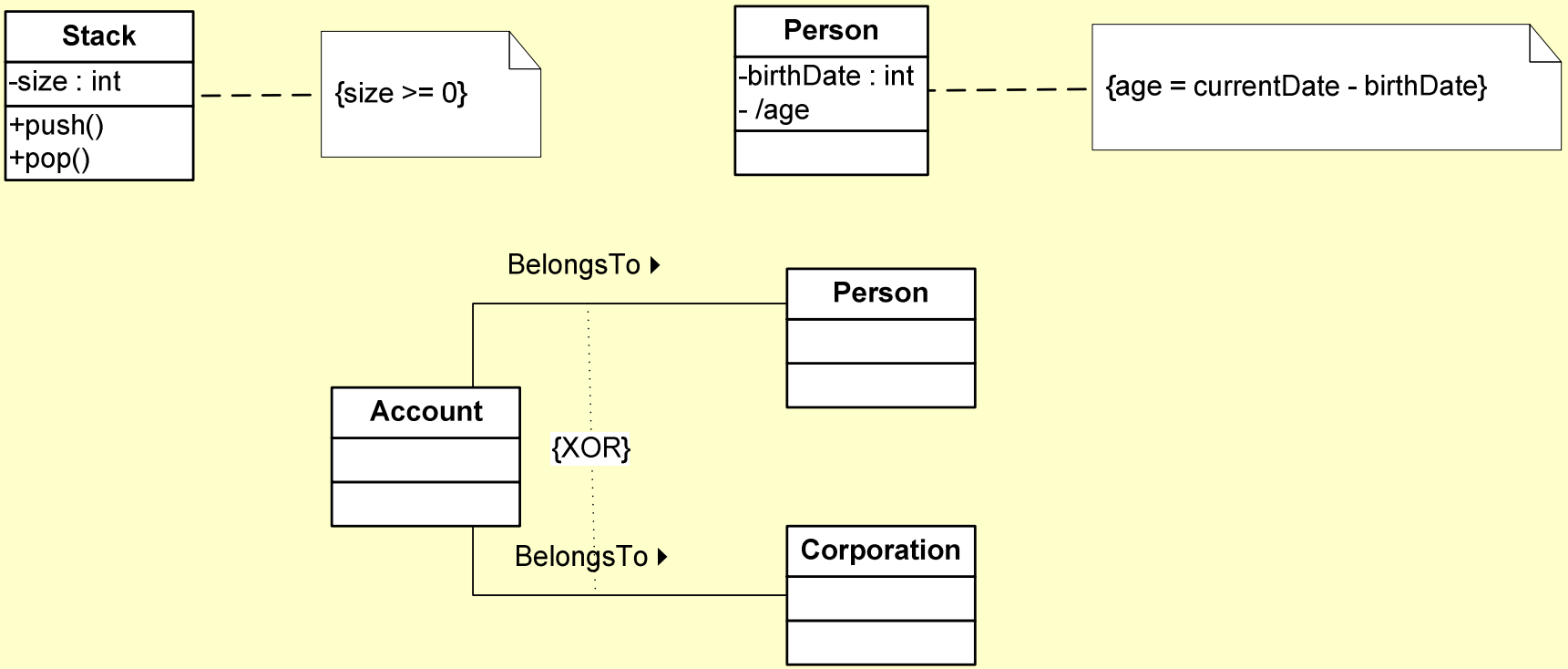
What is an OCL constraint?

- An OCL constraint is an OCL expression that evaluates to true or false (i.e. a Boolean OCL expression)
- Constraints are expressed as : {constraint}
- They are often put after text elements in a UML diagram, or in a note
- Users can define their own (named) constraints:

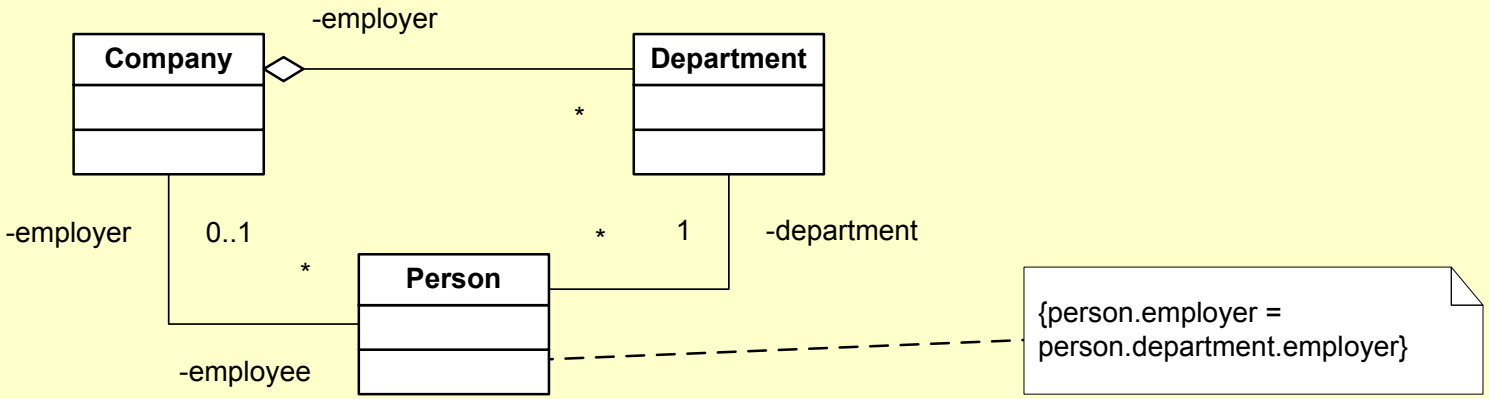
  <constraint> ::= {[<name>:] <Boolean expression>}
- Constraints can be of three kinds:
  - Invariants
  - Pre-conditions
  - Post-conditions

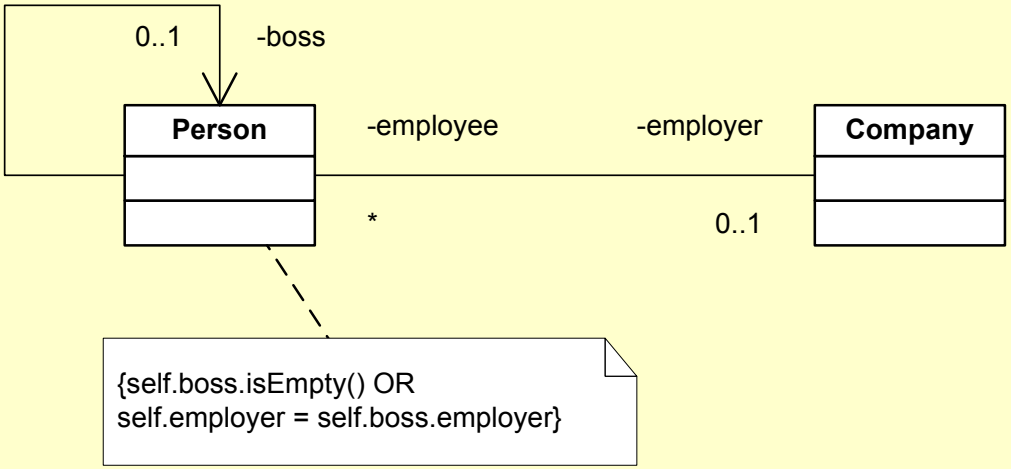
  More on this later…

OCL constraints: examples



OCL constraints: more examples

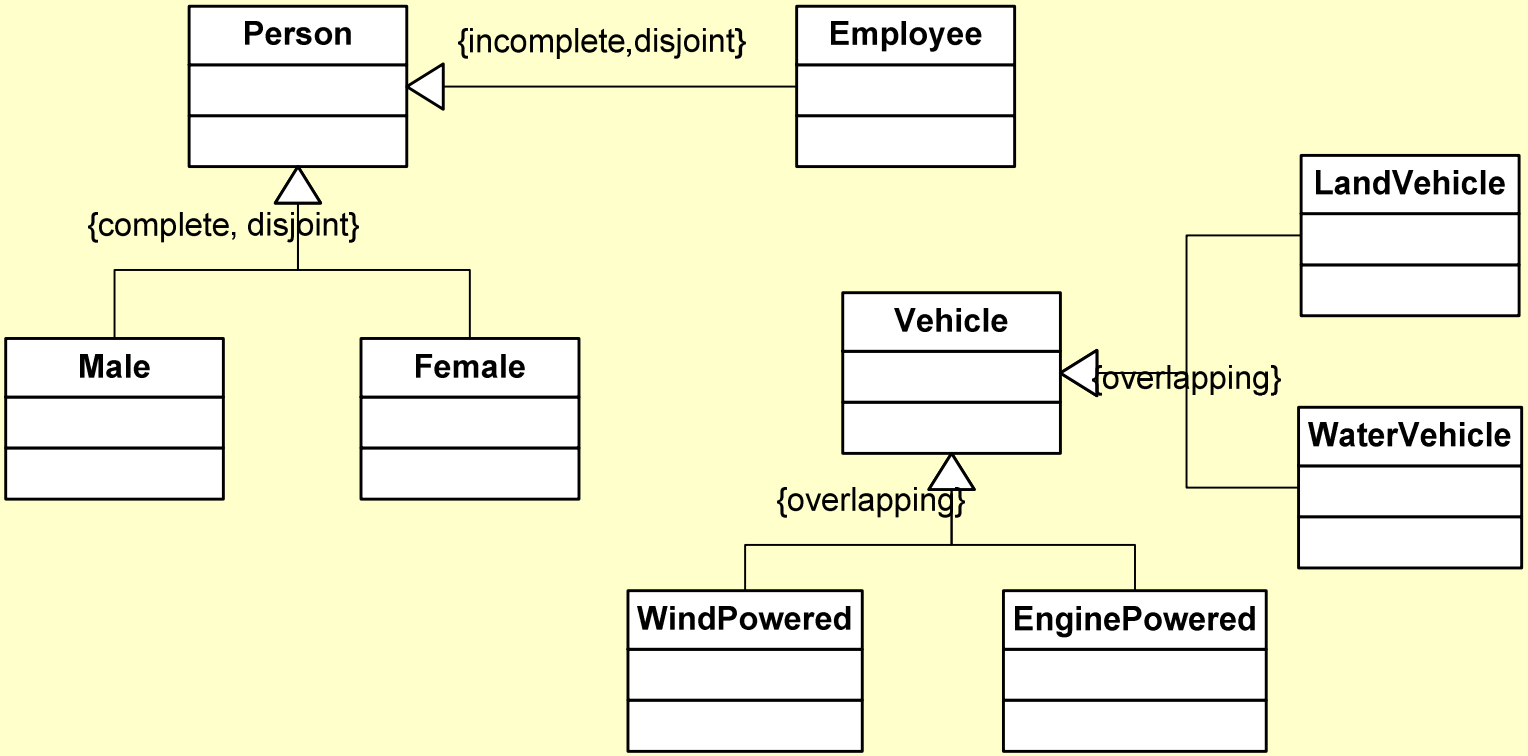




Generalization constraints

- {complete, disjoint}
  - Not extensible, with no common instances
- {incomplete, disjoint}
  - Extensible, with no common instances
- {complete, overlapping}
  - Not extensible, with common instances
- {incomplete, overlapping}
  - Extensible, with common instances
- By default : {incomplete, disjoint}

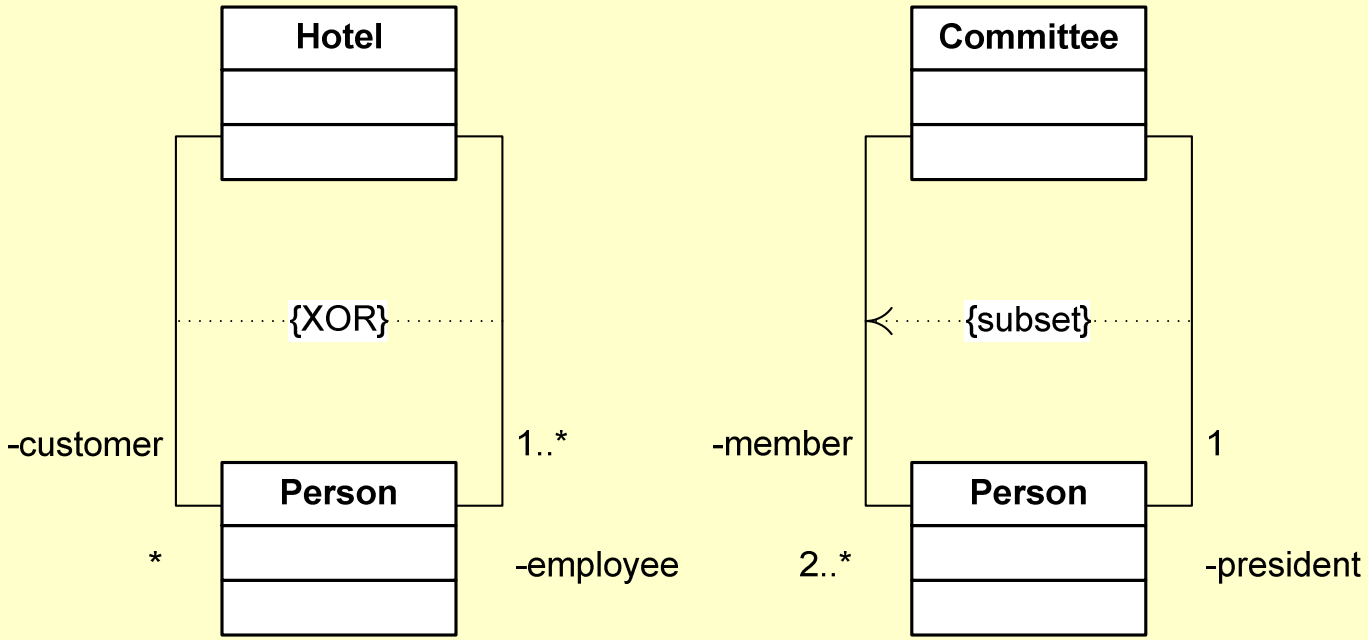
Generalization constraints: examples



Association constraints

- {subsets <property_name>}
- {redefines <property_name>}
- {union}
- {ordered}
- {bag}
- {sequence} or {seq}

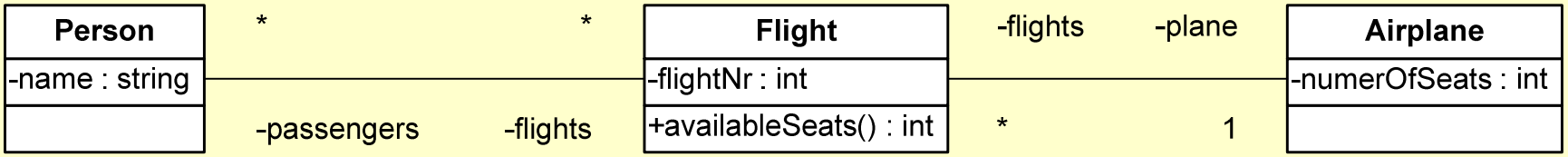
Association constraints examples



OCL: Context

- The context of an OCL expression is the UML element (class, attribute, relation, …) to which it is attached
- An OCL expression is always evaluated for a particular instance (the contextual instance)
  - Default naming: *self* keyword
    - `context Person inv : self.age >= 18`
  - Explicit naming:
    - `context p : Person inv p.age >= 18`
  - Omitted
    - `context Person : inv age >= 18`

Accessing Class Properties



- Dot "." notation is used
- Example: if Flight is the context, to access:
  - an attribute: self.flightNr
  - an operation: self.availableSeats()
  - the opposite association end: self.plane
- Note the importance of roles!

Property Specification

- Double-colon notation “::”
- Example: if Flight is the context,
  - For an attribute:

```
context Flight::flightNb : int
```

  - For an operation:

```
context Flight::availableSeats() : int
```

  - For an association end

```
context Flight::plane : Airplane
```

Constraints: Invariants

- Invariant: a constraint on a (group of) object(s) which must be always verified

```
context Account
inv: self.balance >= self.min AND self.min >= 0
```

- Invariants can be combined:

```
context Account
inv: self.balance >= self.min
inv: self.min >= 0
```

Constraints: Pre/Post conditions

- In OCL we can specify pre/post conditions for operations
  - Pre-conditions: must be verified before operation call
  - Post-conditions: must be verified after operation call

- In post-conditions, two specific elements can be accessed
  - result: refers to the value returned by the operation
  - @pre: refers to the value of an attribute before the call

Constraints: an Example

```
context Compte::debiter(montant: int)
pre: montant > 0 AND
      montant < self.solde – self.plancher
post: self.solde = self.solde@pre – montant

context Compte::getSolde(): int
post: result = self.solde

Context Compte::crediter(montant: int)
Pre: montant > 0
Post: self.solde = self.solde@pre + montant
```

Naming Constraints

- Syntax:

```
context class
inv ConstraintName : constraintExpression
```

- Examples

```
context Compte
inv soldePositif : self.solde > 0

context Compte::debiter(montant: int)
Pre montantPositif: montant > 0
Pre montantDebite: self.solde = self.solde@pre - montant
```

Comments

- Syntax:

```
-- comment
```

- Examples

```
context Compte
inv self.solde > 0  -- solde positif

context Compte::debiter(montant: int)
Pre montant > 0  -- montant positif
Pre montantDebite: self.solde = self.solde@pre - montant
```

Operation Body Expression

- An OCL expression can be used to indicate the result of a query operation

```
context TypeName::operationName(param1 : Type1, …): retType
body: -- some expression
```

- Example

```
context Person::getCurrentSpouse(): Person
pre: self.isMarried = true
body: self.marriages->select( m | m.ended = false).spouse
```

Initial or Derived Values

- An OCL expression can be used to indicate the initial or derived value of an attribute or association end
  - **context** TypeName::AttributeName: Type
    init: -- some expression representing the initial value
  - **context** TypeName::AttributeName: Type
    derive: -- some expression representing the derivation rule
- Examples

```
context Person::income : int
init: 0

context: Person::age: int
derive: currentDate – self.birthdate
```