Features

- … belong to a namespace (e.g., class or package)

TravelStage Handling

TravelStage Handling
+ delay : Minutes + boardingGate : String

Visibility kinds (no default)

		visible to elements …
+	public	that can access owning namespace (by membership, import, or access)
#	protected	with generalization to owning namespace
~	package	in the same package as the owning namespace
-	private	in owning namespace only

Connection
addOne(cp : ConnectionPart)

ConnectionPart
- addOne(c : Connection) {redefines addOne}
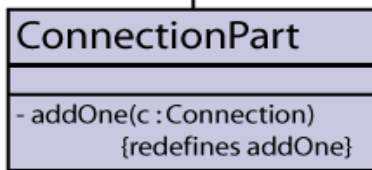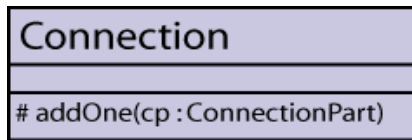
- … are redefinable (unless decorated by { leaf })
  - in classes that specialize the context class
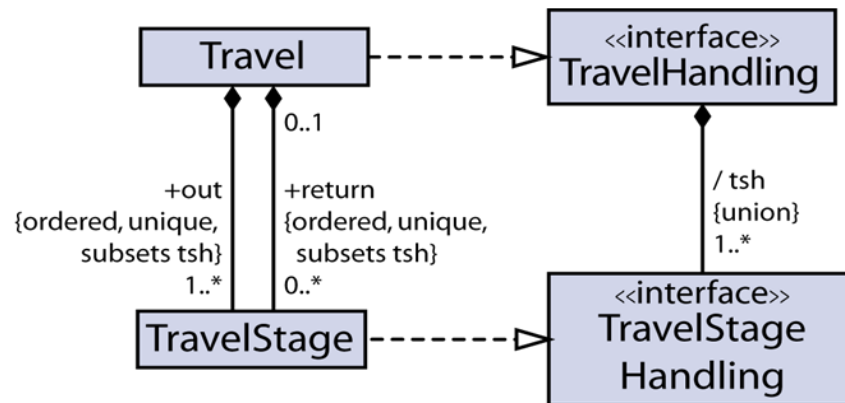
- … can be defined on instance or class level

TravelStage
maxDuration : int = 10

isStatic   default value

Properties

Aggregation kinds (default: *none*)

none	───────	reference
shared	◇───────	undefined (!)
composite	◆───────	value

{ ordered }	{ unique }	Collection type
√	√	OrderedSet
√	×	Sequence
×	√	Set (default)
×	×	Bag

/ ({ derived })   can be computed from other information (default: false)

{ readOnly }   can only be read, not written (default: false = unrestricted)

{ union }   union of subset properties (implies derived)

{ subsets … }   which property this property is a subset of

Behavioral features

- … are realized by behaviors (e.g., code, state machine).
  - { abstract } (virtual) behavioral features declare no behavior
    - behavior must be provided by specializations
  - Exceptions that may be thrown can be declared
  - Limited concurrency control
    - { active } classes define their own concurrency control

      | BoardingControl | – – – active class (with own behavior which starts on instance creation)
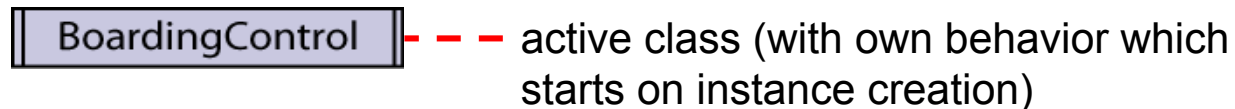
    - in passive classes:

      Call concurrency kinds (no default)

{ sequential }	no concurrency management
{ guarded }	only one execution, other invocations are blocked
{ concurrent }	all invocations may proceed concurrently

Operations (1)

- An **operation** specifies the name, return type, formal parameters, and constraints for invoking an associated behaviour.
  - «pre» / «post»
    - precondition constrains system state on operation invocation
    - postcondition constrains system state after operation is completed
  - { query }: invocation has no side effects
    - «body»: body condition describes return values
  - { ordered, unique } as for properties, but for return values
  - exceptions that may be thrown can be declared

Parameter direction kinds (default: in)

in	one way from caller
out	one way from callee
inout	both ways
return	return from callee (at most 1)

| Connection |
|---|
| + add(cps : ConnectionParts[0..*]) |

parameter name
parameter type
parameter multiplicity

Operations (2)

- Several *semantic variation points* for operations
  - What happens, if a precondition is not satisfied on invocation?
  - When inherited or redefined
    - invariant, covariant, or contravariant specialization?
    - How are preconditions combined?

- **No predefined resolution principle** for inherited or redefined operations
  - "The mechanism by which the behavior to be invoked is determined from an operation and the transmitted argument data is a semantic variation point."
  - a single-dispatch, object-oriented resolution principle is mentioned explicitly in the UML 2 specification

- Operation invocations may be **synchronous** or **asynchronous**.

Signals and receptions

- A **signal** is a specification of type of send request instances communicated between objects.
    - Signals are classifiers, and thus may carry arbitrary data.
    - A signal triggers a reaction in the receiver in an asynchronous way and without a reply (no blocking on sender).

- A **reception** is a declaration stating that a classifier is prepared to react to the receipt of a signal.
    - Receptions are behavioral features and thus are realized by behavior (e.g., a state machine).
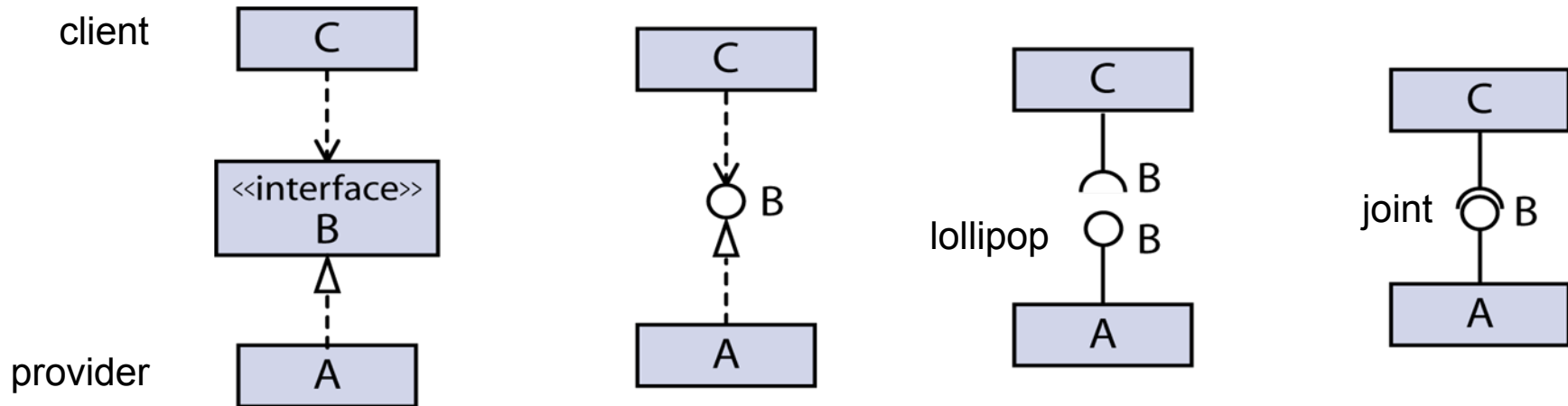
«signal» CardInserted

BoardingControl
«signal» CardInserted

‑ ‑ ‑**Reception**
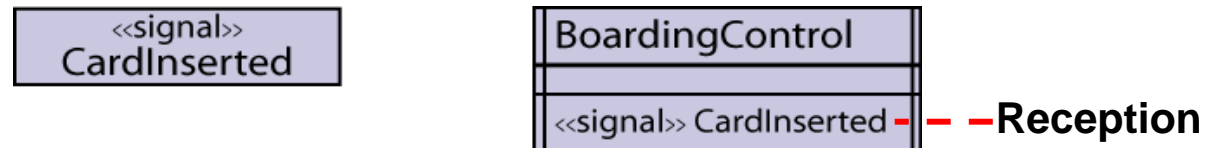
Interfaces

- **Interfaces** declare a set of coherent public features and obligations.
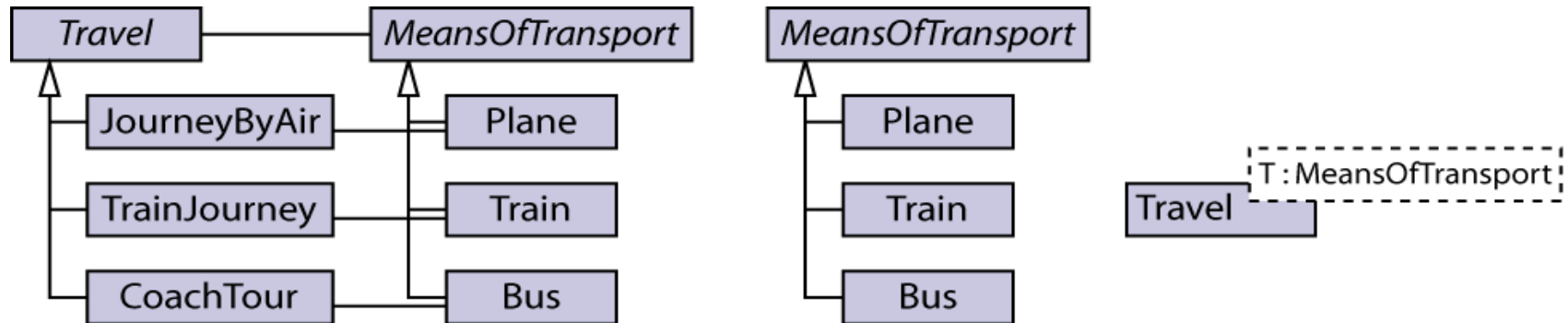 - i.e., specify a contract for implementers (realizers)

features to be offered
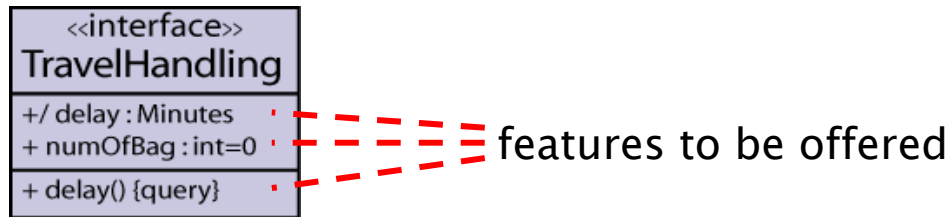
Several notations for client/provider relationship

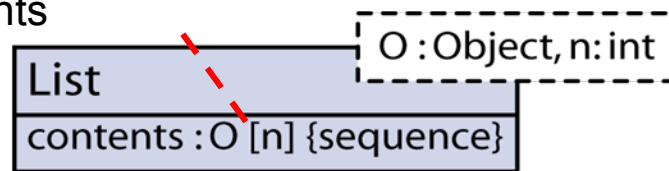client

provider

lollipop

joint

Templates

subtype polymorphism   vs.   **parametric** polymorphism

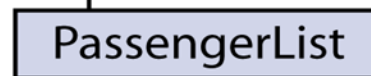exposed parameterable elements

Template class
(**ParameterableElement**)
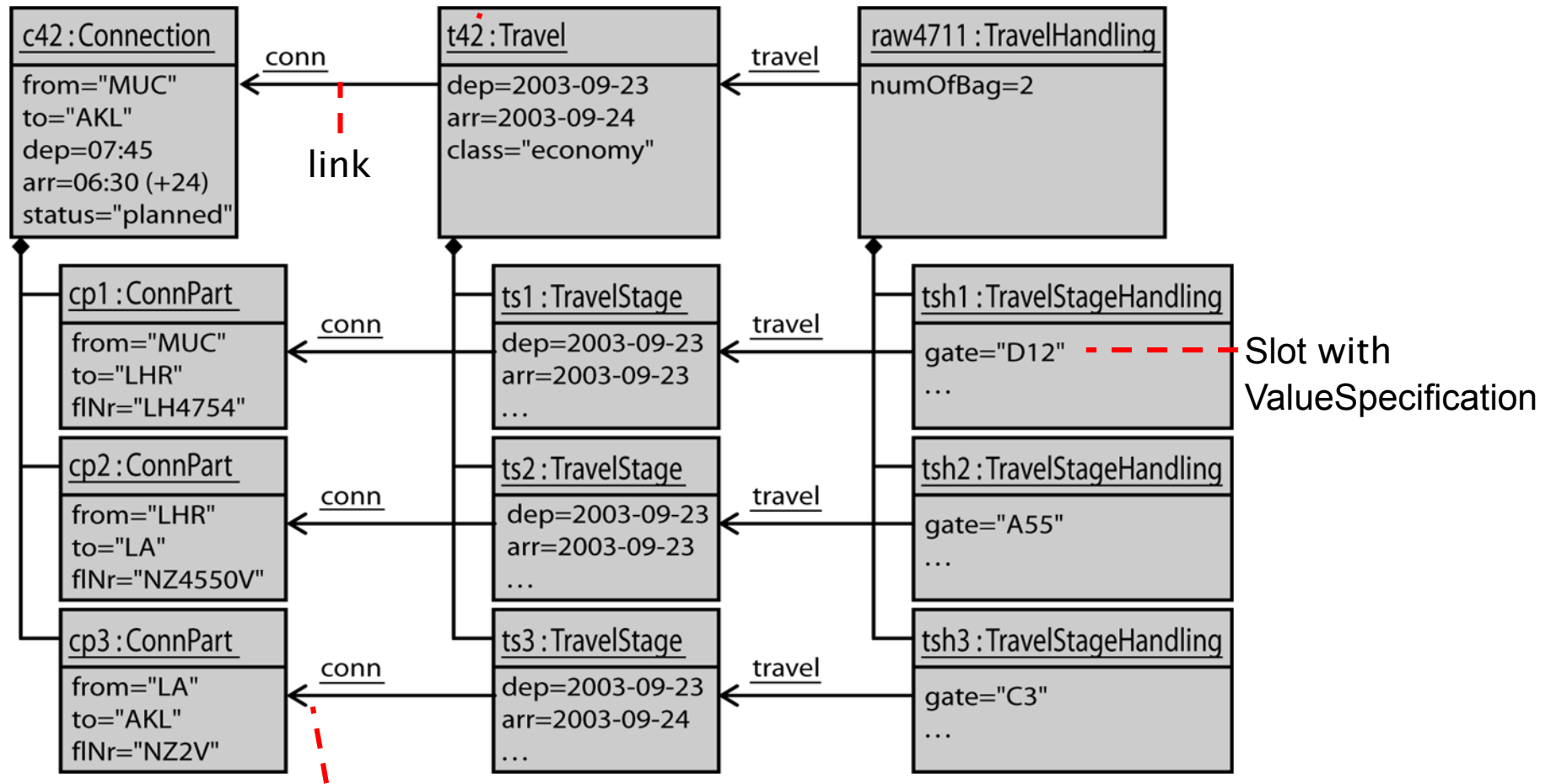
template parameters

template binding
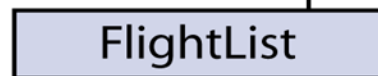
Bound class
(**TemplateableElement**)

Object diagram

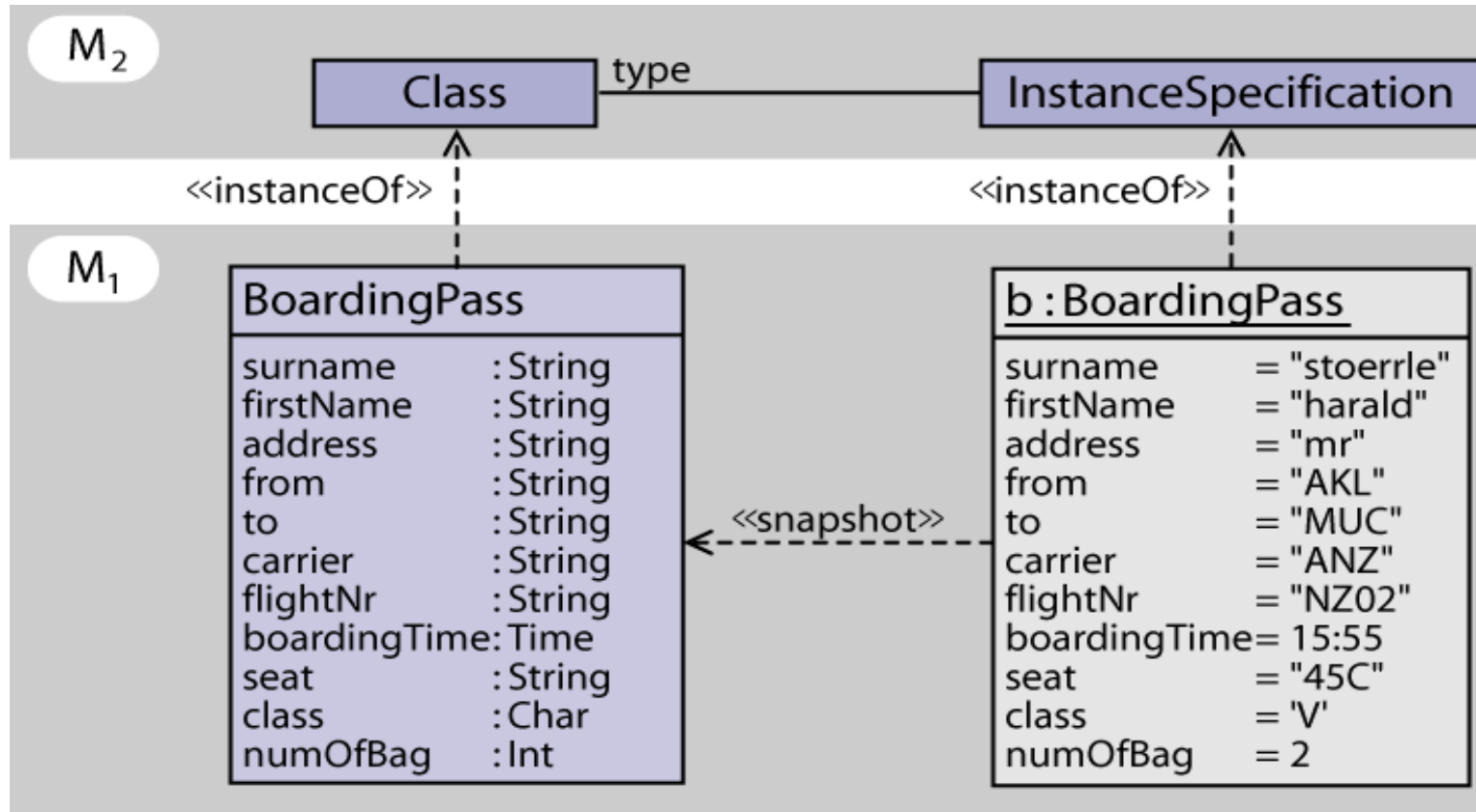InstanceSpecification   InstanceValue

Slot with ValueSpecification

link

underlining and association end adornments are optional

Instances specifications

UML metamodel

user model

Wrap up

- **Classifiers** and their **Relationships** describe the vocabulary of a system.

- Classifiers describe a set of instances with common **Features**.
  - StructuralFeatures (Property's)
  - BehavioralFeatures (Operations, Receptions)
- **Associations** describe structural relationships between classes.
  - Association ends are Property's.

- **Generalizations** relate specific Classifiers to more general Classifiers.

- **Packages** group elements
  - and provide a Namespace for grouped elements.

- **InstanceSpecifications** and links describe system snapshots.

Modelling with UML, with semantics