Features

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

Visibility kinds (no default)

<table>
<thead>
<tr>
<th></th>
<th>visible to elements …</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>public</td>
</tr>
<tr>
<td></td>
<td>that can access owning namespace</td>
</tr>
<tr>
<td></td>
<td>(by membership, import, or access)</td>
</tr>
<tr>
<td>#</td>
<td>protected</td>
</tr>
<tr>
<td></td>
<td>with generalization to owning namespace</td>
</tr>
<tr>
<td>~</td>
<td>package</td>
</tr>
<tr>
<td></td>
<td>in the same package as the owning namespace</td>
</tr>
<tr>
<td>-</td>
<td>private</td>
</tr>
<tr>
<td></td>
<td>in owning namespace only</td>
</tr>
</tbody>
</table>

• … are redefinable (unless decorated by \{ leaf \})
  • in classes that specialize the context class

• … can be defined on instance or class level

isStatic

TravelStage
maxDuration:int = 10 — default value
## Properties

### Aggregation kinds (default: *none*)

<table>
<thead>
<tr>
<th>Aggregation kind</th>
<th>Reference type</th>
<th>Collection type</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>none</em></td>
<td>reference</td>
<td><em>OrderedSet</em></td>
</tr>
<tr>
<td><em>shared</em></td>
<td>undefined (!)</td>
<td><em>Sequence</em></td>
</tr>
<tr>
<td><em>composite</em></td>
<td>value</td>
<td><em>Set</em> (default)</td>
</tr>
</tbody>
</table>

- / \{( 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:

<table>
<thead>
<tr>
<th>Call concurrency kinds (no default)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>{ sequential }</td>
<td>no concurrency management</td>
</tr>
<tr>
<td>{ guarded }</td>
<td>only one execution, other invocations are blocked</td>
</tr>
<tr>
<td>{ concurrent }</td>
<td>all invocations may proceed concurrently</td>
</tr>
</tbody>
</table>
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)

<table>
<thead>
<tr>
<th>Parameter direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>one way from caller</td>
</tr>
<tr>
<td>out</td>
<td>one way from callee</td>
</tr>
<tr>
<td>inout</td>
<td>both ways</td>
</tr>
<tr>
<td>return</td>
<td>return from callee (at most 1)</td>
</tr>
</tbody>
</table>

Connection

```
+ add(cps:ConnectionParts[0..*])
```

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Parameter type</th>
<th>Parameter multiplicity</th>
</tr>
</thead>
</table>

Modelling with UML, with semantics
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).
Interfaces

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

```
<<interface>>
TravelHandling
+ delay: Minutes
+ numOfBag: int = 0
+ delay() [query]
```

features to be offered

Several notations for client/provider relationship

- **Client**
  - `C`
  - `B`
  - `A`

- **Provider**
  - `B`

- **Lollipop**

- **Joint**
Templates

**subtype** polymorphism vs. **parametric** polymorphism

![Diagram showing subtype and parametric polymorphism]

Exposed parameterable elements

**Template class** `(ParameterableElement)`

- `List`
  - `contents : O[n] {sequence}`

  - `<<bind>> <O -> Flight, n=20>`
  - `FlightList`

  - `<<bind>> <O -> Passenger, n=300>`
  - `PassengerList`

**Bound class** `(TemplateableElement)`

`T : MeansOfTransport`
InstanceSpecification

InstanceValue

c42 : Connection
from = "MUC" to = "AKL"
dep = 07:45
arr = 06:30 (+24) status = "planned"

cp1 : ConnPart
from = "MUC" to = "LHR"
flNr = "LH4754"

cp2 : ConnPart
from = "LHR" to = "LA"
flNr = "NZ4550V"

cp3 : ConnPart
from = "LA" to = "AKL"
flNr = "NZ2V"

t42 : Travel
dep = 2003-09-23
arr = 2003-09-24
class = "economy"

ts1 : TravelStage
dep = 2003-09-23
arr = 2003-09-23

Slot with ValueSpecification

tsh1 : TravelStageHandling
gate = "D12"

tsh2 : TravelStageHandling
gate = "A55"

tsh3 : TravelStageHandling
gate = "C3"

underlining and association end adornments are optional
Instances specifications

UML metamodel

user model

Modelling with UML, with semantics
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.