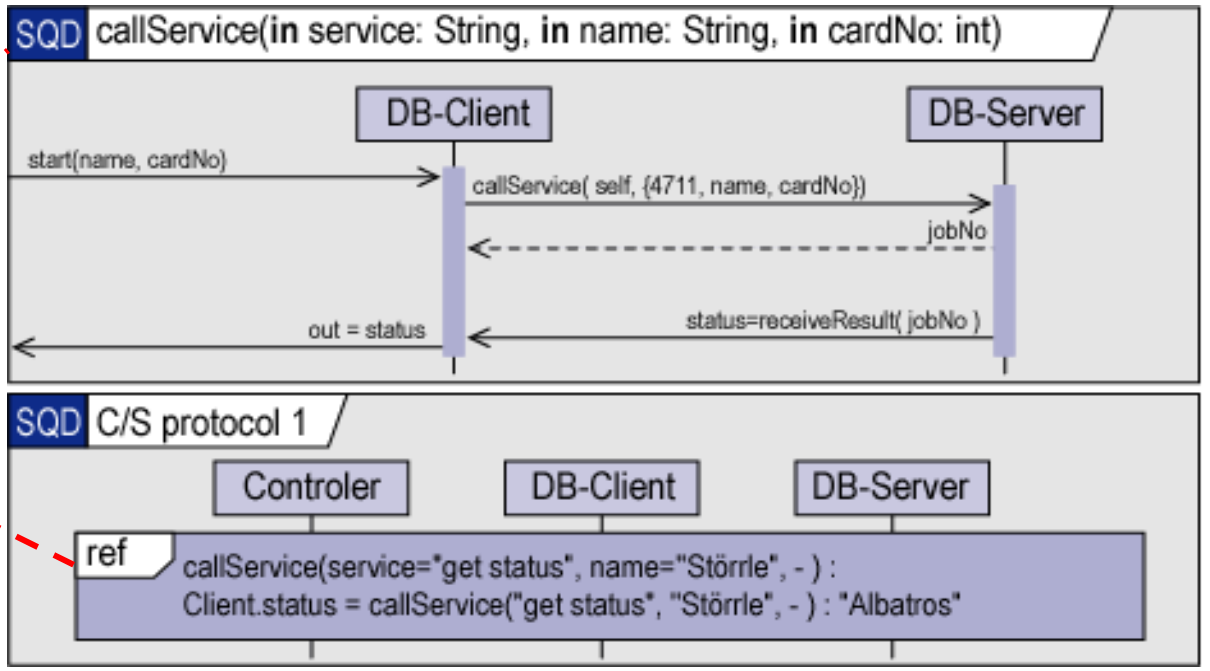
Interaction operator ref & parameters

- **ref**
  - refers to a fragment defined elsewhere (macro-expansion)
  - Formal and actual parameters (bindings) are declared in the diagram head.

declaration

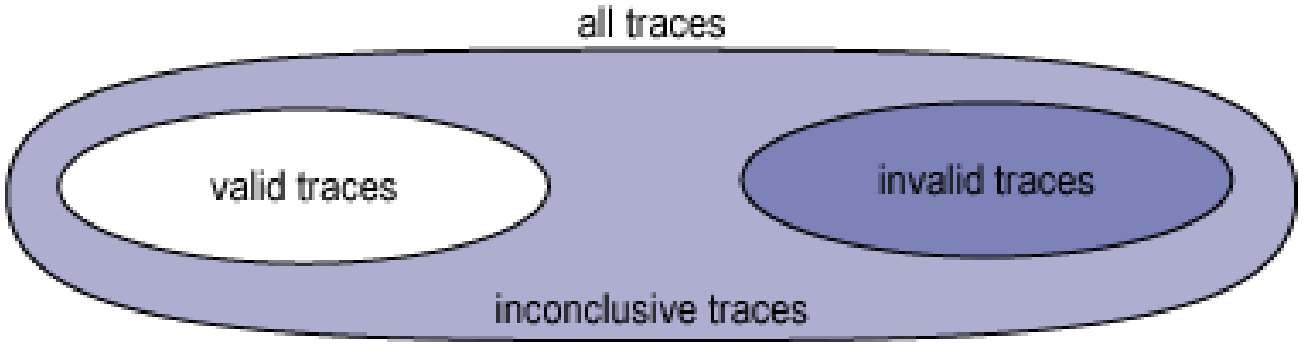
call



- Signals to the containing classifier appear as arrows form the diagram border.

Interaction operators: negation

- The semantics of neg and assert is unclear.
- In contrast to that the other operators, they refer not just to the positive traces, but to invalid and inconclusive traces as well.



- **neg**
  - declare all valid traces as invalid
  - inconclusive traces: unknown

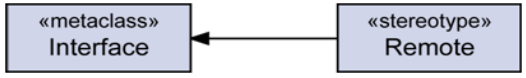
- **assert**
  - remove uncertainty by declaring all inconclusive traces as invalid

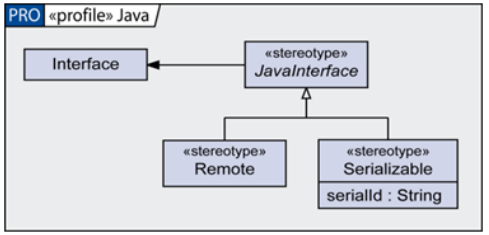
Wrap up

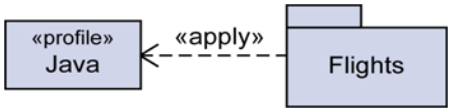
- Complex interactions **like high-level MSCs** added.
- New diagram types:
  - timing diagrams (like oscilloscope), and
  - interaction overview (similar to restricted activity diagram)
  - renamed collaboration diagram to communication diagram
- Completely **new metamodel**.
- Almost formal three-valued semantics of valid, invalid and inconclusive interleaving traces of events.
- Some semantical problems are yet to be solved.

Unified Modeling Language 2

Profiles



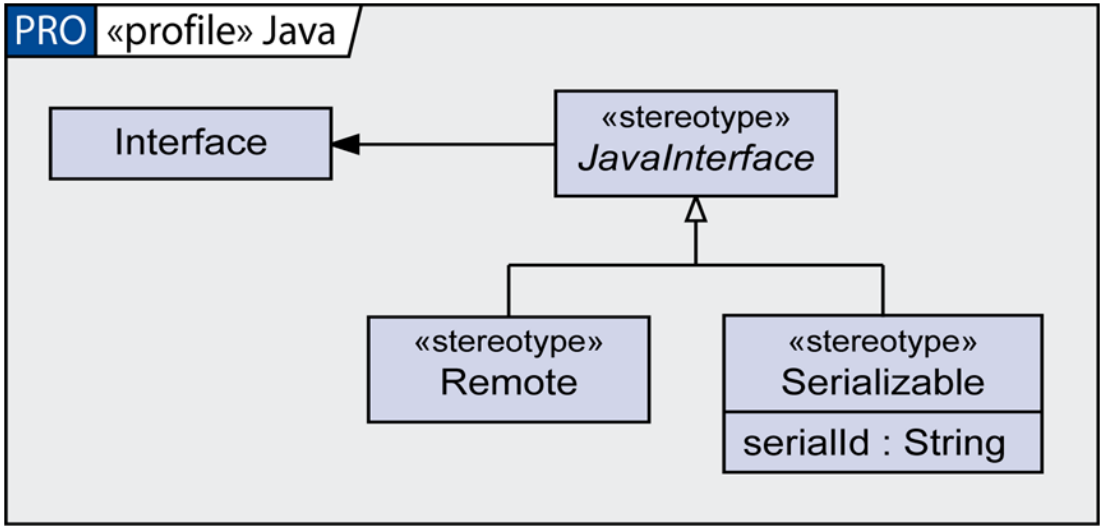




Usage scenarios

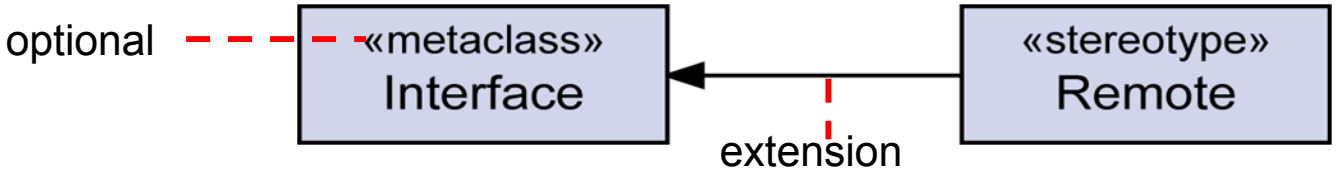
- **Metamodel customization** for
  - adapting terminology to a specific platform or domain
  - adding (visual) notation
  - adding and specializing semantics
  - adding constraints
  - transformation information

- **Profiling**
  - packaging domain-specific extensions
  - “domain-specific language” engineering

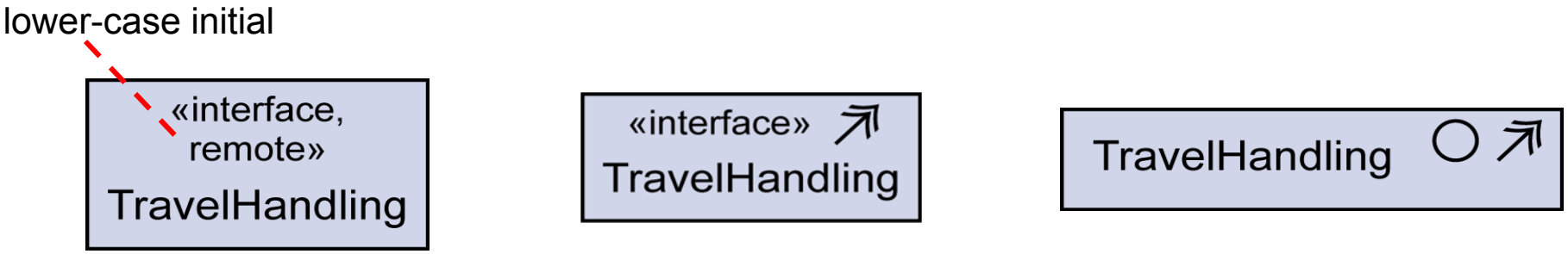


Stereotypes (1)

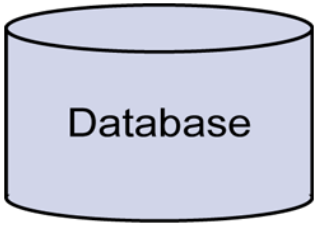
- Stereotypes define how an existing (UML) metaclass may be extended.



- Stereotypes may be applied **textually** or **graphically**.

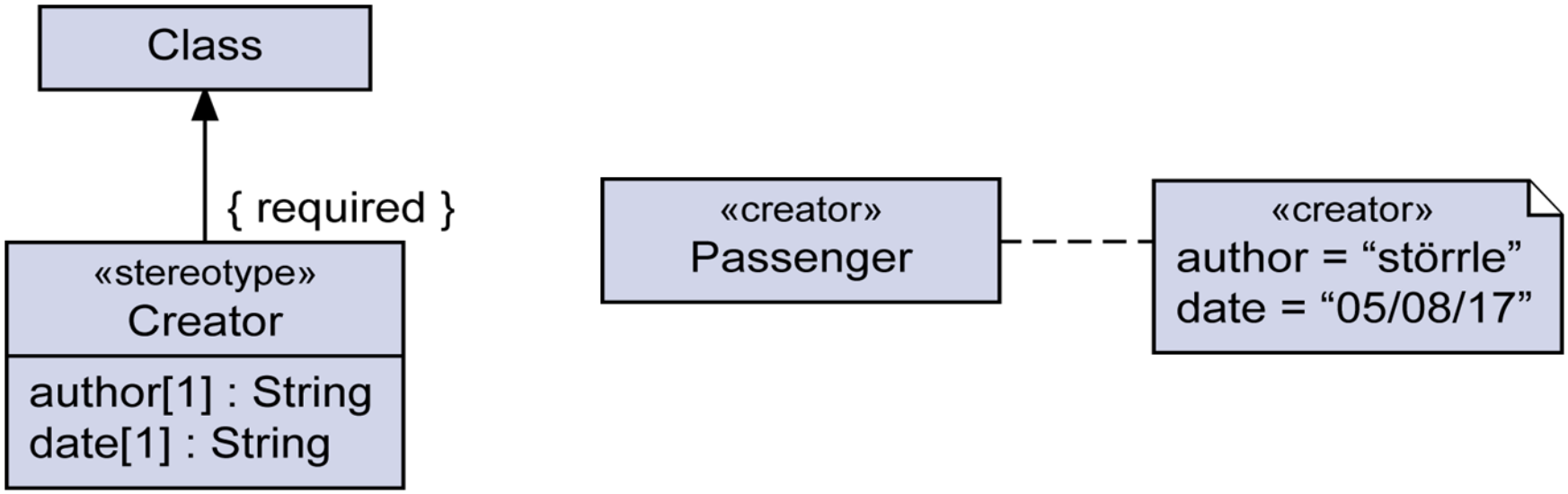


- Visual stereotypes may replace original notation.
  - But the element name should appear below the icon…



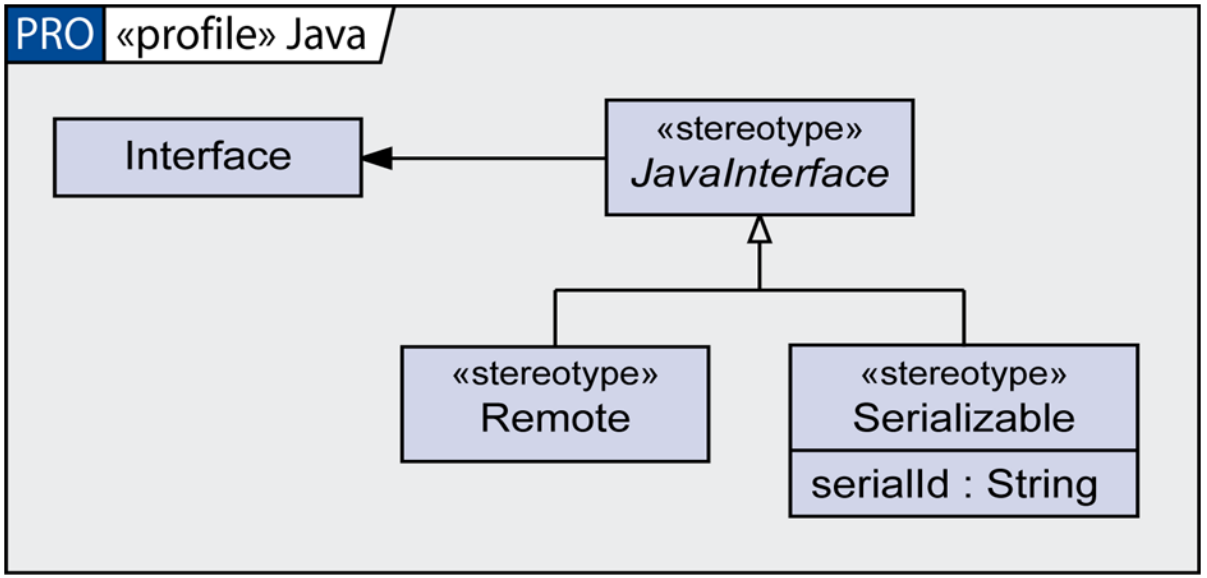
Stereotypes (2)

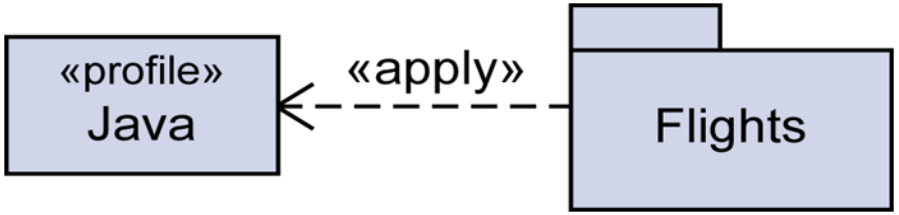
- Stereotypes may define **meta-properties**.
  - commonly known as “tagged values”
- Stereotypes can be defined to be **required**.
  - Every instance of the extended metaclass has to be extended.
  - If a required extension is clear from the context it need not be visualized.



Profiling

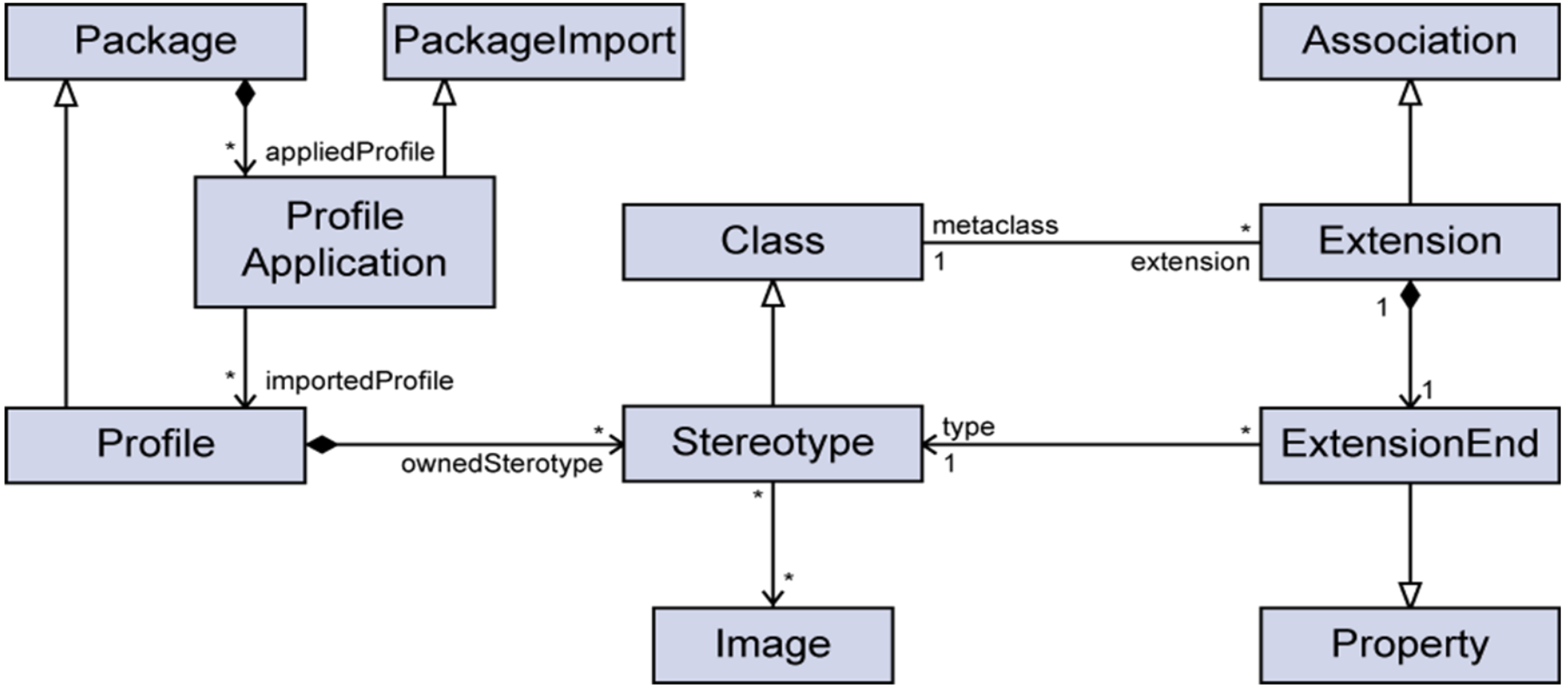
- Profiles **package** extensions.





Metamodel

- Based on **infrastructure library** constructs
  - Class, Association, Property, Package, PackageImport



Metamodeling with Profiles

- Profile extension mechanism imposes **restrictions** on how the UML metamodel can be modified.
  - UML metamodel considered as “read only”.
  - No intermediate metaclasses
- Stereotypes metaclasses below UML metaclasses.

Wrap up

- Metamodel extensions
  - with stereotypes and meta-properties
  - for restricting metamodel semantics
  - for extending notation
- Packaging of extensions into profiles
  - for declaring applicable extensions
  - “domain-specific language” engineering