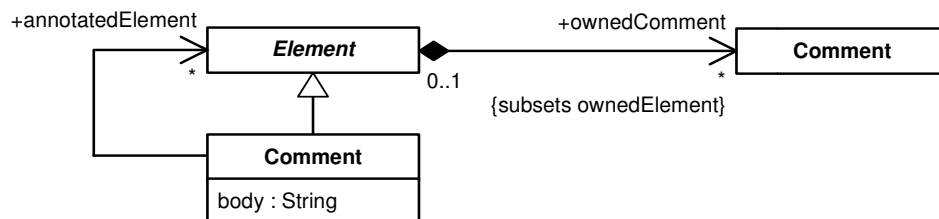


Die nachfolgenden Abschnitte zeigen die Metamodelle einiger Allgemeiner Elemente, die zwar nicht zu irgend einem spezifischen Diagramm gehören, Spielen aber eine wichtige Rolle für die Elemente der anderen Diagramme. Die Elemente, die Sie in diesem Dokument finden sind:

- Kommentar,
- Ausdruck,
- Randbedingung,
- Datentyp,
- Primitiver Typ,
- Aufzählungstyp,
- Literal.

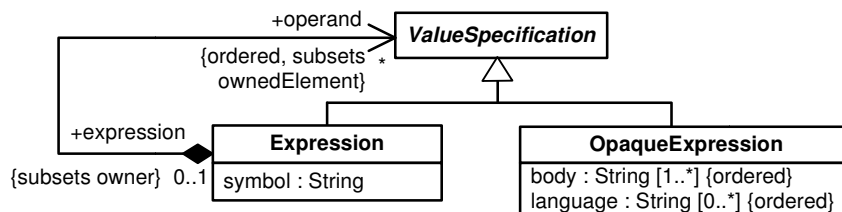
Der Kommentar im Metamodell



Die Klasse Comment im Metamodell

Abbildung 1 zeigt die Repräsentation des Kommentars im UML-Metamodell als Klasse Comment. Ein Kommentar *kann* als `ownedComment` von einem beliebigen Modellelement besessen oder als `annotatedElement` an ein oder mehrere Elemente geheftet werden. Das Attribut `body` vom Typ `String` repräsentiert den eigentlichen Kommentartext.

Der Ausdruck im Metamodell



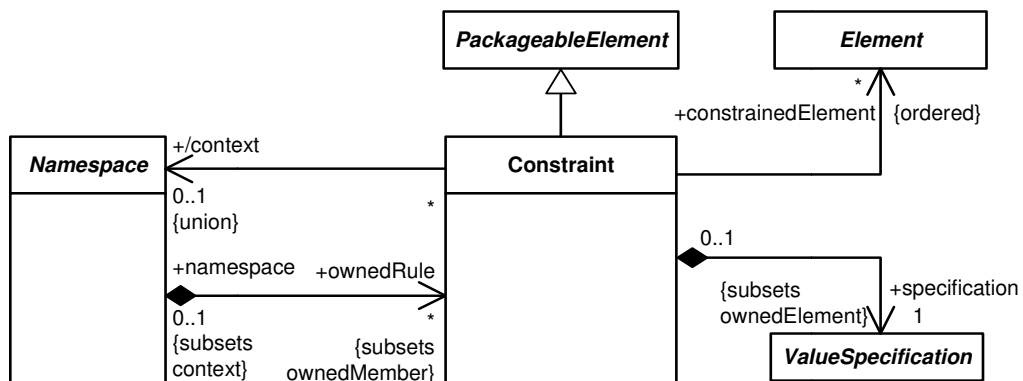
Die Klassen Expression und OpaqueExpression im Metamodell

Sowohl die Klasse `Expression` als auch `OpaqueExpression` ist eine Subklasse von `ValueSpecification`. Die Kompositionsbeziehung verdeutlicht, dass ein Ausdruck selbst wieder beliebig viele solcher Wertespezifikationen als *Operanden* enthalten kann und dass die Menge der Operanden eines Ausdrucks geordnet `{ordered}` ist. Das Symbol (z.B. `+` oder `*`) wird durch das `String`-Attribut `symbol` repräsentiert. Bei einer `OpaqueExpression` wird der oder die Texte durch `body` abgebildet. Falls optional noch die Sprache bzw. die Sprachen definiert sind, wird dies durch `Strings` von `language` realisiert. Beide Attribute sind geordnet, sodass die erste Sprache dem ersten Text, die zweite Sprache dem zweiten Text, usw. zugeordnet ist.

Die Randbedingung im Metamodell

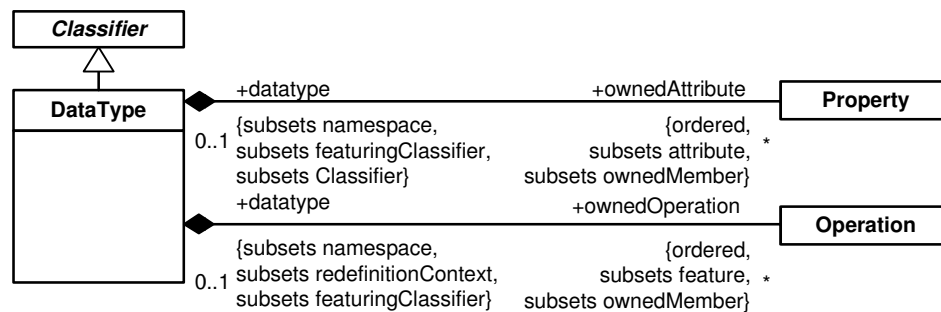
Eine Randbedingung wird durch die Klasse `Constraint` repräsentiert. Die Bedingung gilt für maximal einen Namensraum (`+ownedRule`), der den Kontext (`+context`) definiert. Randbedingungen können zusätzlich losgelöst von anderen Elementen in Paketen `PackageableElement` aufbewahrt werden. Daneben existiert eine gerichtete Assoziation (`+constrainedElement`) zu den Elementen, die durch die Randbedingung eingeschränkt/präzisiert werden.

Die eigentliche Bedingung wird durch eine Wertespezifikation (`ValueSpecification`) modelliert. Derartige Wertespezifikationen sind z.B. Ausdrücke (`Expressions`) oder sprachabhängige Ausdrücke (`OpaqueExpressions`).



Die Klasse `Constraint` im Metamodell

Der Datentyp im Metamodell

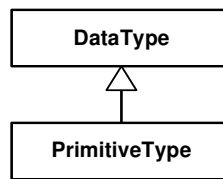


Die Datentyp-Klassen im UML-Metamodell

Ein Datentyp (Klasse `DataType`) ist ein spezieller `Classifier`. Wie die Kompositionsbeziehungen zwischen `DataType` und `Property` beziehungsweise `Operation` zeigen, kann ein Datentyp eine Struktur mit beliebig vielen Attributen und Operationen enthalten.

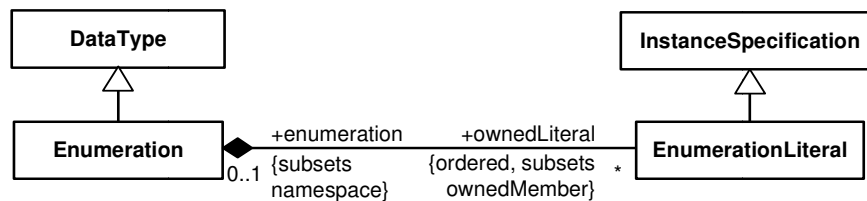
Primitiver Typ im Metamodell

Der Primitive Typ ist eine Spezialisierung des Datentyps. Die Klasse `PrimitiveType` besitzt keine weiteren Attribute oder Assoziationen und schränkt dadurch den Datentyp ein.



Die Klasse `PrIMITIVEType` im Metamodell

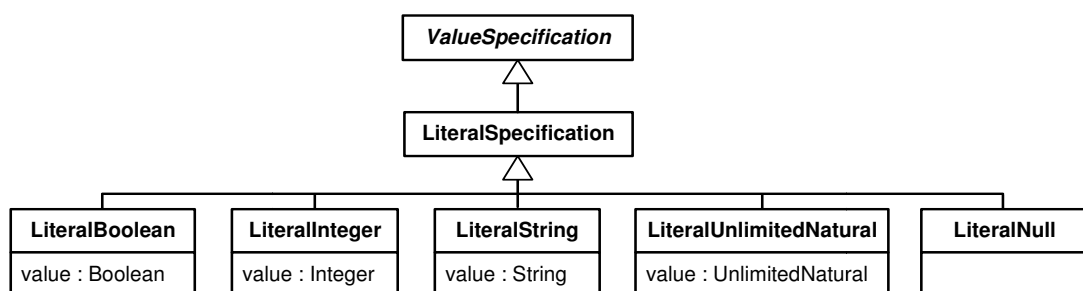
Der Aufzählungstyp im Metamodell



Die Klassen `Enumeration` und `EnumerationLiteral` im Metamodell

Wie die Generalisierungsbeziehung zeigt, ist ein Aufzählungstyp (`Enumeration`) ein spezieller Datentyp (`DataType`). Ein Aufzählungstyp kann beliebig viele Aufzählungswerte (`+ownedLiteral`) umfassen. Diese sind geordnet. Jeder Aufzählungswert `EnumerationLiteral` ist wiederum eine spezielle `InstanceSpecification`. Dies ist zwar nicht besonders sinnvoll, spiegelt aber den momentanen Stand der Spezifikation wider. Wir erwarten eine Umdefinition in ein `NamedElement` evtl. ergänzt um eine `ValueSpecification` damit jedem Aufzählungsliteral ein Wert zugewiesen werden kann.

Das Literal im Metamodell



Die Literal-Klassen im Metamodell

Alle Literal-Klassen im Metamodell sind Spezialisierungen der abstrakten Klasse `LiteralSpecification`, die nur zur Gruppierung aller Literale eingeführt wurde. Jedes Literal stellt auch eine `ValueSpecification` dar. Die verschiedenen Literal-Klassen besitzen ein Attribut `value` von dem Datentyp, den die jeweilige Literal-Klasse für die Modellierung definiert. Eine Ausnahme stellt die Klasse `LiteralNull` dar, da sie ja gerade das Nicht-Vorhandensein eines Wertes spezifiziert.