Modellbasierte Softwareentwicklung (MODSOFT)

Part II

Domain Specific Languages

EMF Validation and EMF Tools (EMFT)

Prof. Joachim Fischer /
Dr. Markus Scheidgen / Dipl.-Inf. Andreas Blunk

{fischer,scheidge,blunk}@informatik.hu-berlin.de

LFE Systemanalyse, III.310

Agenda

prolog (1 VL)
Introduction: languages and their aspects, modeling vs. programming, meta-modeling and the 4 layer model

0. (2 VL)
Eclipse/Plug-ins: eclipse, plug-in model and plug-in description, features, *p2*-repositories, *RCP*s

➡ 1. (2 VL)
Structure: *Ecore*, *genmodel*, working with generated code, constraints with *Java* and *OCL*, *XML/XMI*

2. (3 VL)
Notation: Customizing the tree-editor, textural with *XText*, graphical with *GEF* and *GMF*

3. (4 VL)
Semantics: interpreters with Java, code-generation with *Java* and *XTend*, model-transformations with *Java* and *ATL*

epilog (2 VL)
Tools: persisting large models, model versioning and comparison, model evolution and co-adaption, modular languages with *XBase*, *Meta Programming System* (MPS)

EMF Validation

Validation – Overview

- Ecore defines “how models look like”, but does not define “how models do not look like” – meta-models define data, not rules, not behavior
- Not all Ecore features are covered by the Java mapping
  - e.g. multiplicity lower bounds, unique value sets, composition
  - some are caught at runtime
    - e.g. composition
  - others are not caught at all
    - e.g. lower bounds
- Not all *constraints* can be covered with Ecore
  - e.g. all elements in a container must have unique names
  - e.g. password must contain special character

Validation – Overview

- Types of constraints
  - check Ecore inherit constraints that are not covered by EMF Java mapping or runtime
    - already defined as part of the meta-model
  - named constraints
    - declared outside the meta-model
  - invariants
    - declared as part of the meta-model

Invariants in Ecore

- Declare via Annotation
  - source=http://www.eclipse.org/emf/2002/Ecore
    - key=constraints
    - value=ContraintNameA ConstraintNameB ...
- EMF generates a validator class
  - fido.util.FidoValidator
  - one method body for each constraint
    - fido.util.FidoValidator#validateOwner_ConstraintNameA

Invariants in Ecore

```java
/**
 * Validates the OwnerNameStartsWithCapital constraint of '<em>Owner</em>'.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
public boolean validateOwner_OwnerNameStartsWithCapital(Owner owner,
        DiagnosticChain diagnostics,
        Map<Object, Object> context) {
    // TODO implement the constraint
    // -> specify the condition that violates the constraint
    // -> verify the diagnostic details, including severity, code, and message
    // Ensure that you remove @generated or mark it @generated NOT
    if (false) {
        if (diagnostics != null) {
            diagnostics.add
                (createDiagnostic
                    (Diagnostic.ERROR,
                     DIAGNOSTIC_SOURCE,
                     0,
                     "_UI_GenericConstraint_diagnostic",
                     new Object[] { "OwnerNameStartsWithCapital",
                             getObjectLabel(owner, context) },
                     new Object[] { owner },
                     context));
        }
        return false;
    }
    return true;
}
```

ore

intNameA

Ir

```java
/**
 * Validates the OwnerNameStartsWithCa
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated
 */
public boolean validateOwner_OwnerName
        DiagnosticChain diagnostics,
        Map<Object, Object> context) {
    // TODO implement the constraint
    // -> specify the condition that v
    // -> verify the diagnostic detail
    // Ensure that you remove @generat
    if (false) {
        if (diagnostics != null) {
            diagnostics.add
                (createDiagnostic
                    (Diagnostic.ERROR
                     DIAGNOSTIC_SOURC
                     0,
                     "_UI_GenericCons
                     new Object[] {
                             getObjec
                     new Object[] { o
                     context));
        }
        return false;
    }
    return true;
}
```

```java
/**
 * Validates the OwnerNameStartsWithCapital constraint of '<em>Owner</em>'.
 * <!-- begin-user-doc -->
 * <!-- end-user-doc -->
 * @generated NOT
 */
public boolean validateOwner_OwnerNameStartsWithCapital(Owner owner,
        DiagnosticChain diagnostics,
        Map<Object, Object> context) {
    boolean success = true;
    String name = owner.getName();
    if (name != null) {
        String first = name.substring(0,1);
        success = first.toUpperCase().equals(first);
    }

    if (!success) {
        if (diagnostics != null) {
            diagnostics.add
                (createDiagnostic
                    (Diagnostic.ERROR,
                     DIAGNOSTIC_SOURCE,
                     0,
                     "_UI_GenericConstraint_diagnostic",
                     new Object[] { "OwnerNameStartsWithCapital",
                             getObjectLabel(owner, context) },
                     new Object[] { owner },
                     context));
        }
        return false;
    }
    return true;
}
```

Triggering Validation

```
Owner markus = FidoFactory.eINSTANCE.createOwner();
//markus.setName("markus");
Dog fido = FidoFactory.eINSTANCE.createDog();
fido.setName("Fido");
fido.setWeight(20);
markus.getPets().add(fido);

Diagnostic diagnostic = Diagnostician.INSTANCE.validate(markus);
if (diagnostic.getSeverity() != Diagnostic.OK) {
    System.out.println("ERROR in: " + diagnostic.getMessage());
    for (Diagnostic child: diagnostic.getChildren()) {
        System.out.println("    " + child.getMessage());
    }
}
```

xpensive

arch efforts

.g. from

- Validation can be triggered programatically

Triggering Validation

```
Owner markus = FidoFactory.eINSTANCE.createOwner();
//markus.setName("markus");
Dog fido = FidoFactory.eINSTANCE.createDog();
fido.setName("Fido");
fido.setWeight(20);
markus.getPets().add(fido);

Diagnostic diagnostic = Diagnostician.INSTANCE.validate(markus);
if (diagnostic.getSeverity() != Diagnostic.OK) {
    System.out.println("ERROR in: " + diagnostic.getMessage());
    for (Diagnostic child: diagnostic.getChildren()) {
        System.out.println("    " + child.getMessage());
    }
}
```

xpensive

rch efforts

.g. from

- Validation can be triggered programatically

Triggering Validation

```
Owner markus = FidoFactory.eINSTANCE.createOwner();
//markus.setName("markus");
Dog fido = FidoFactory.eINSTANCE.createDog();
fido.setName("Fido");
fido.setWeight(20);
markus.getPets().add(fido);

Diagnostic diagnostic = Diagnostician.INSTANCE.validate(markus);
if (diagnostic.getSeverity() != Diagnostic.OK) {
    System.out.println("ERROR in: " + diagnostic.getMessage());
    for (Diagnostic child: diagnostic.getChildren()) {
        System.out.println("    " + child.getMessage());
    }
}
```

```
ERROR in: Diagnosis of fido.impl.OwnerImpl@70e08379{#//}
   The 'OwnerNameStartsWithCapital' constraint is violated on 'fido.impl.OwnerImpl@70e08379{#//}'
```

expensive

arch efforts

.g. from

Triggering Validation

```
Owner markus = FidoFactory.eINSTANCE.createOwner();
//markus.setName("markus");
Dog fido = FidoFactory.eINSTANCE.createDog();
fido.setName("Fido");
fido.setWeight(20);
markus.getPets().add(fido);

Diagnostic diagnostic = Diagnostician.INSTANCE.validate(markus);
if (diagnostic.getSeverity() != Diagnostic.OK) {
    System.out.println("ERROR in: " + diagnostic.getMessage());
    for (Diagnostic child: diagnostic.getChildren()) {
        System.out.println("    " + child.getMessage());
    }
}
```

```
ERROR in: Diagnosis of fido.impl.OwnerImpl@60dd8b3d{#//}
   The required feature 'name' of 'fido.impl.OwnerImpl@60dd8b3d{#//}' must be set
```

expensive

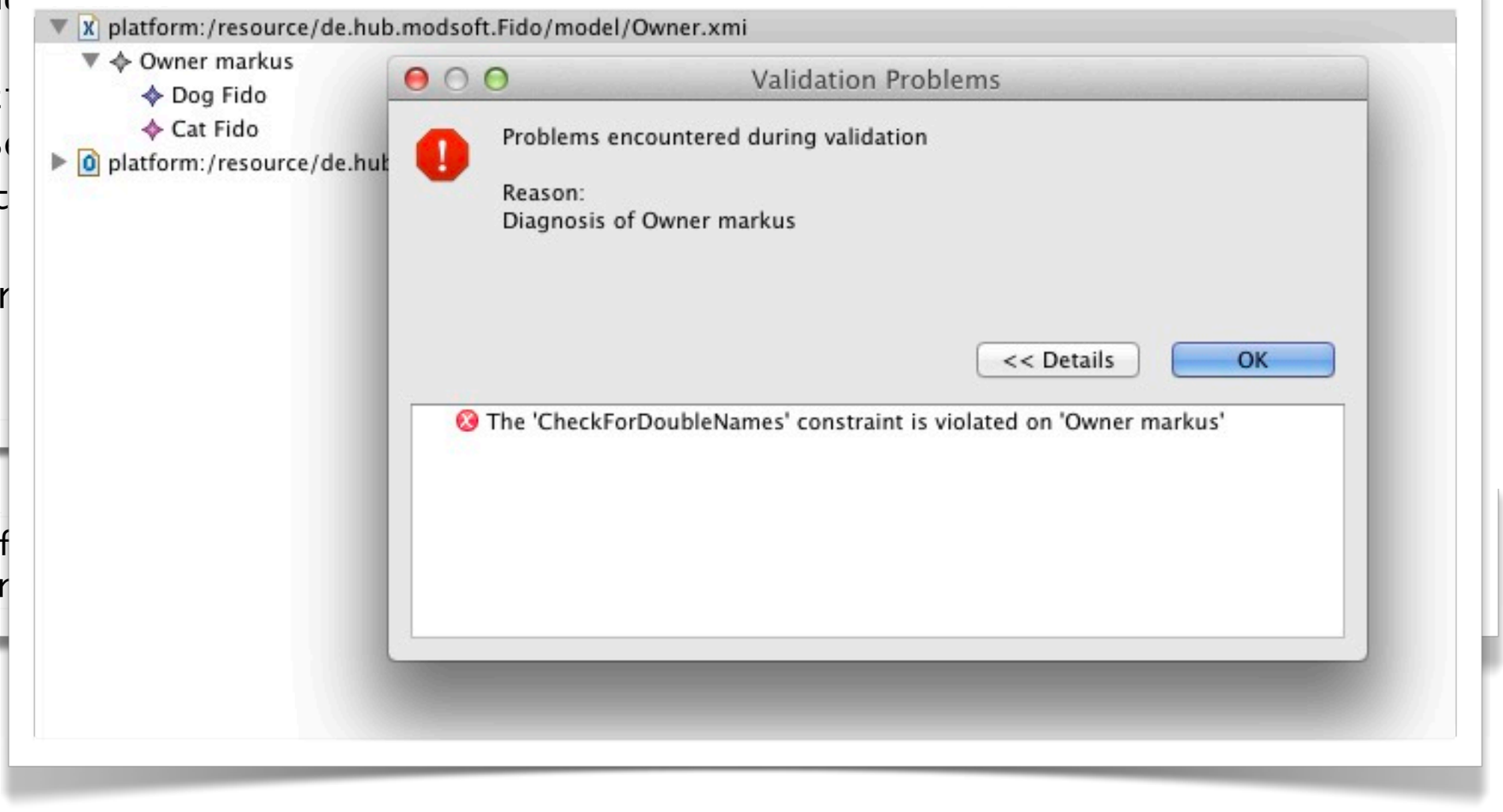
arch efforts

.g. from

Triggering Validation

```
Owner markus = FidoFactory.eINSTANCE.createOwner();
//markus.setName("markus");
Dog fido = FidoFactory.eINSTANCE.createDog();
fido.setName("Fido");
fido.setWeight(20);
markus.getPets().ad

Diagnostic diagnost
if (diagnostic.getS
    System.out.print
    for (Diagnostic
        System.out.pr
    }
}
```

```
ERROR in: Diagnosis of
   The required featur
```



Invariants with OCL

- Object Constraint Language (OCL)
  - OMG standard
  - rationale: predicate logic in not mathematical more engineering like notation
  - allows object navigation via structural features
  - expressions over value sets via higher order functions
  - local and arithmetical expressions
  - fully functional and stateless

```
self.pets->forAll(a : Pet, b : Pet | a.name = b.name implies a = b);
```

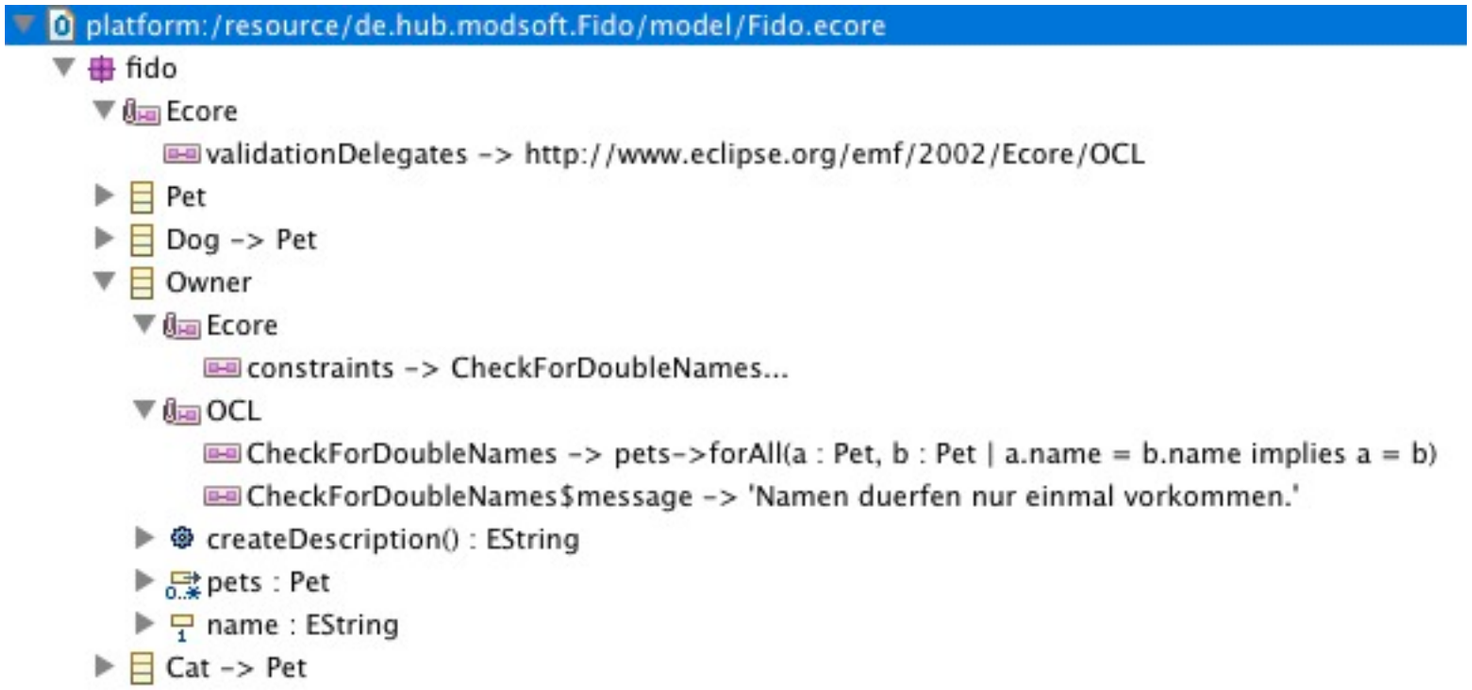
Invariants with OCL

- Object Constraint Language (OCL)
  - OMG standard
  - rationale: predicate logic in not mathematical more engineering
  -
  -
  -
  -

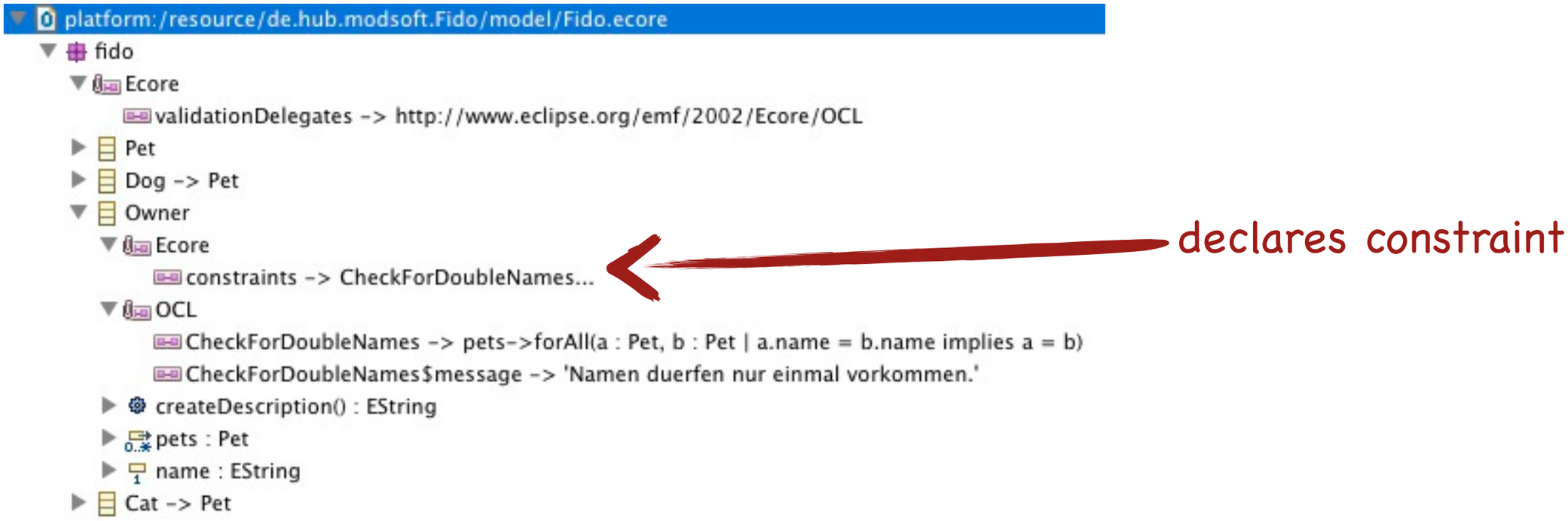
```
boolean success = true;
for(Pet a: owner.getPets()) {
    for(Pet b: owner.getPets()) {
        if (a.getName() != null && b.getName() != null && a.equals(b)) {
            success = a.equals(b);
        }
    }
}
return success;
```

```
self.pets->forAll(a : Pet, b : Pet | a.name = b.name implies a = b);
```

Invariant with OCL in EMF



Invariant with OCL in EMF

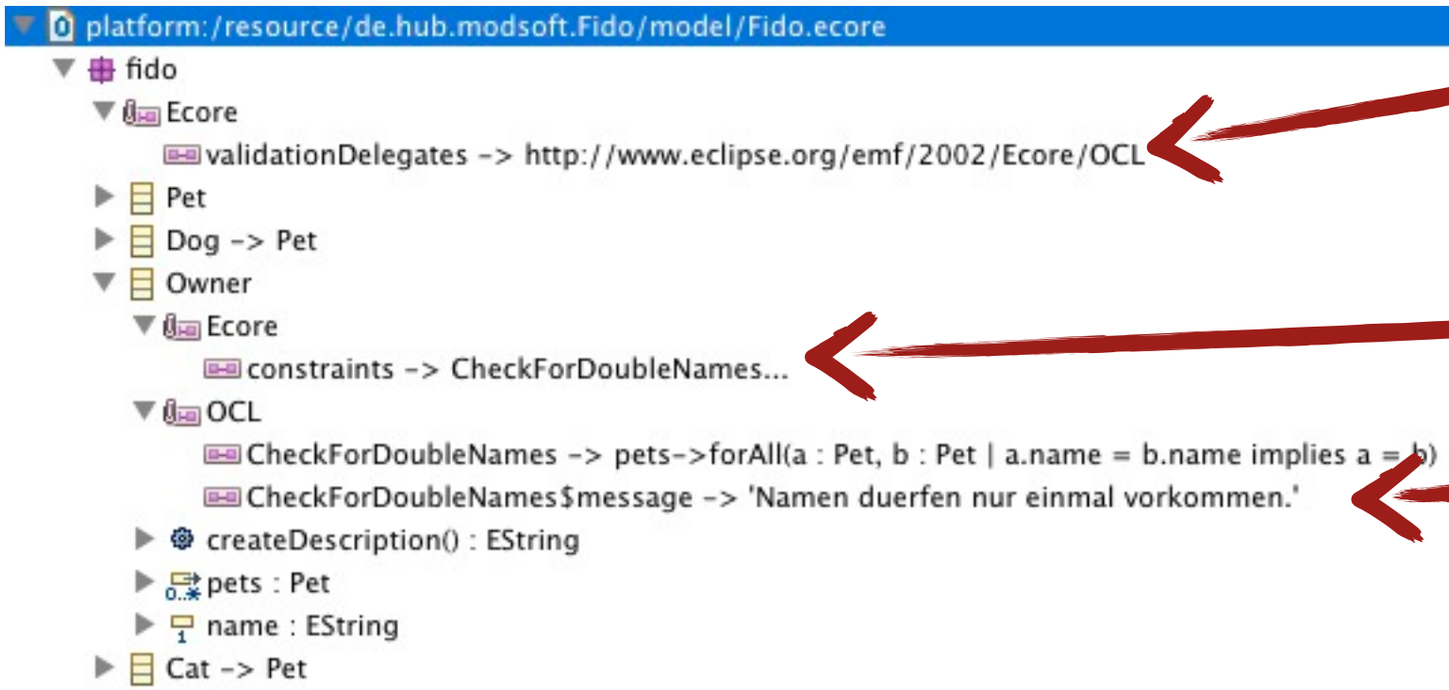


Invariant with OCL in EMF

declares constraint

defines constraint implementation in OCL

Invariant with OCL in EMF



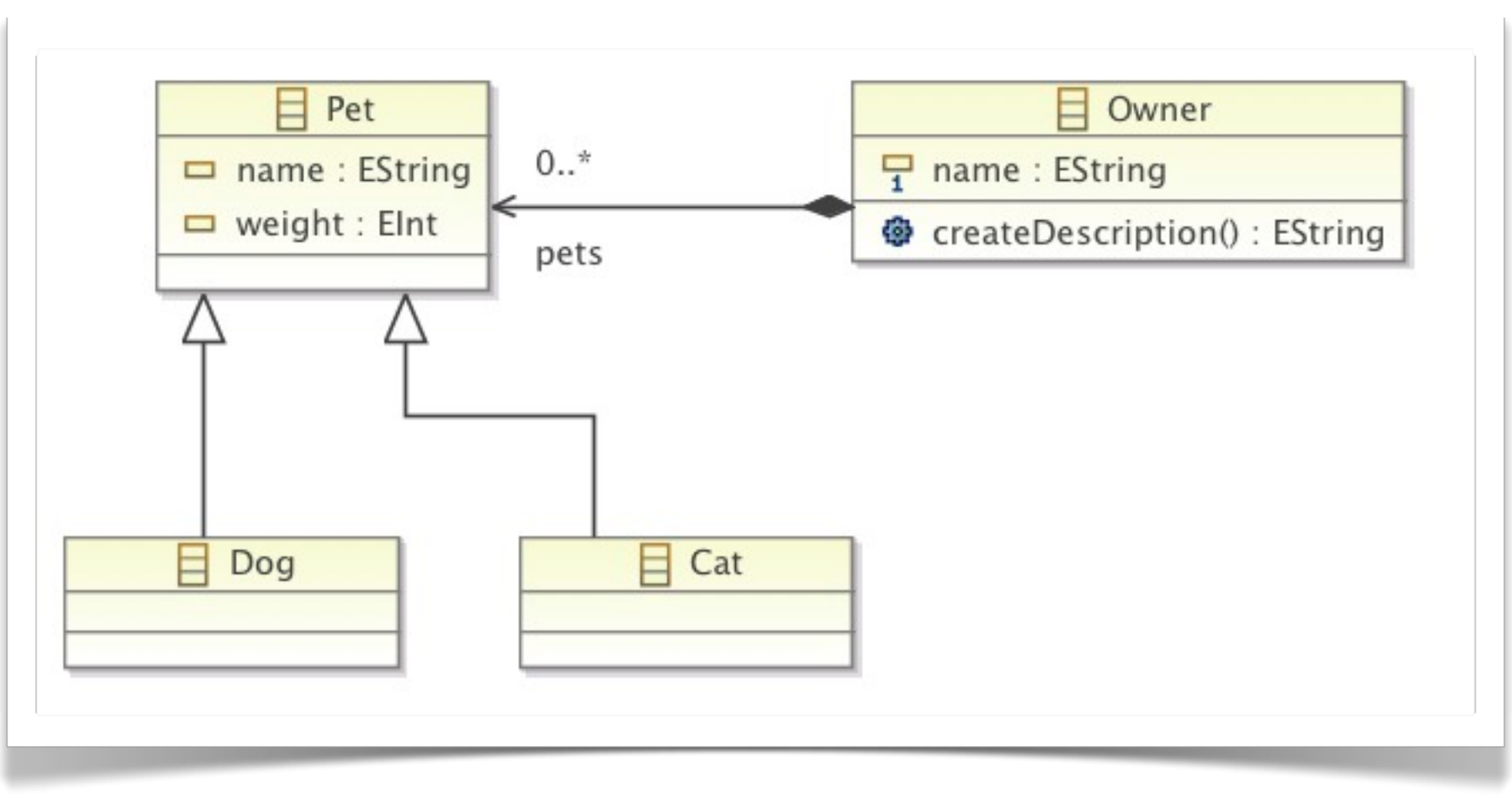
tells EMF to delegate validation to OCL implementation

declares constraint

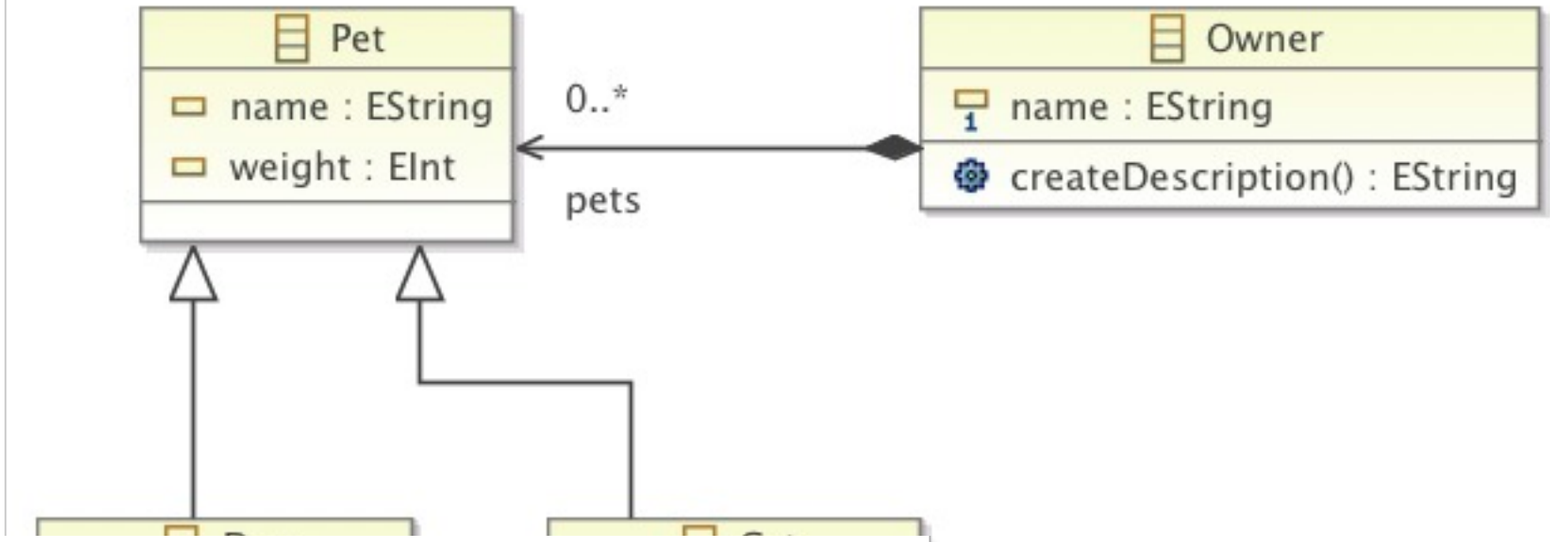
defines constraint implementation in OCL

Ecore as Text

- OclInEcore Editor
- EMFatic
- EMFText



- OclInEcore Editor
- EMFatic
- EMFText




```
module _'Fido.ecore'
import ecore : 'http://www.eclipse.org/emf/2002/Ecore#/';

package fido : fido = 'http://fido/1.0'
{
    abstract class Pet
    {
        attribute name : String[?] { ordered };
        attribute weight : ecore::EInt[?] { ordered };
    }
    class Dog extends Pet;
    class Owner
    {
        property pets : Pet[*] { ordered composes };
        attribute name : String[1] { ordered };
        operation createDescription() : String[?] { ordered };
    }
    class Cat extends Pet;
}
```

- OclInEcore Editor
- EMFatic
- EMFText

OCL as Part of Ecore as Text

OCL as Part of Ecore as Text

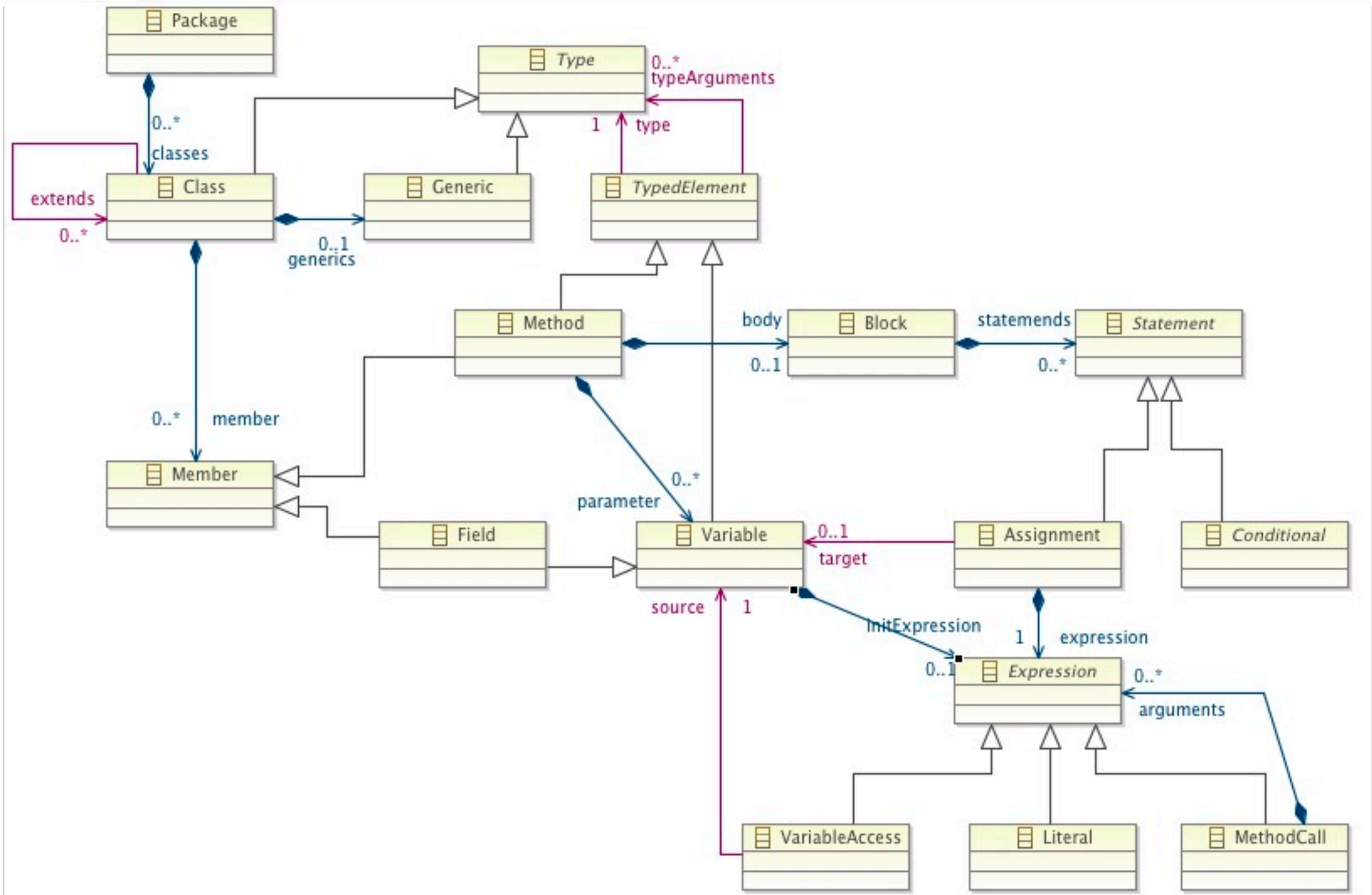
```
module _'Fido.ecore'
import ecore : 'http://www.eclipse.org/emf/2002/Ecore#/';

package fido : fido = 'http://fido/1.0'
{
    abstract class Pet
    {
        attribute name : String[?] { ordered };
        attribute weight : ecore::EInt[?] { ordered };
    }
    class Dog extends Pet;
    class Owner
    {
        invariant CheckForDoubleNames('Namen duerfen nur einmal vorkommen.'):
        pets->forAll(a : Pet, b : Pet | a.name = b.name implies a = b);

        property pets : Pet[*] { ordered composes };
        attribute name : String[1] { ordered };
        operation createDescription() : String[?] { ordered };
    }
    class Cat extends Pet;
}
```

OCL Validation in Standalone Applications

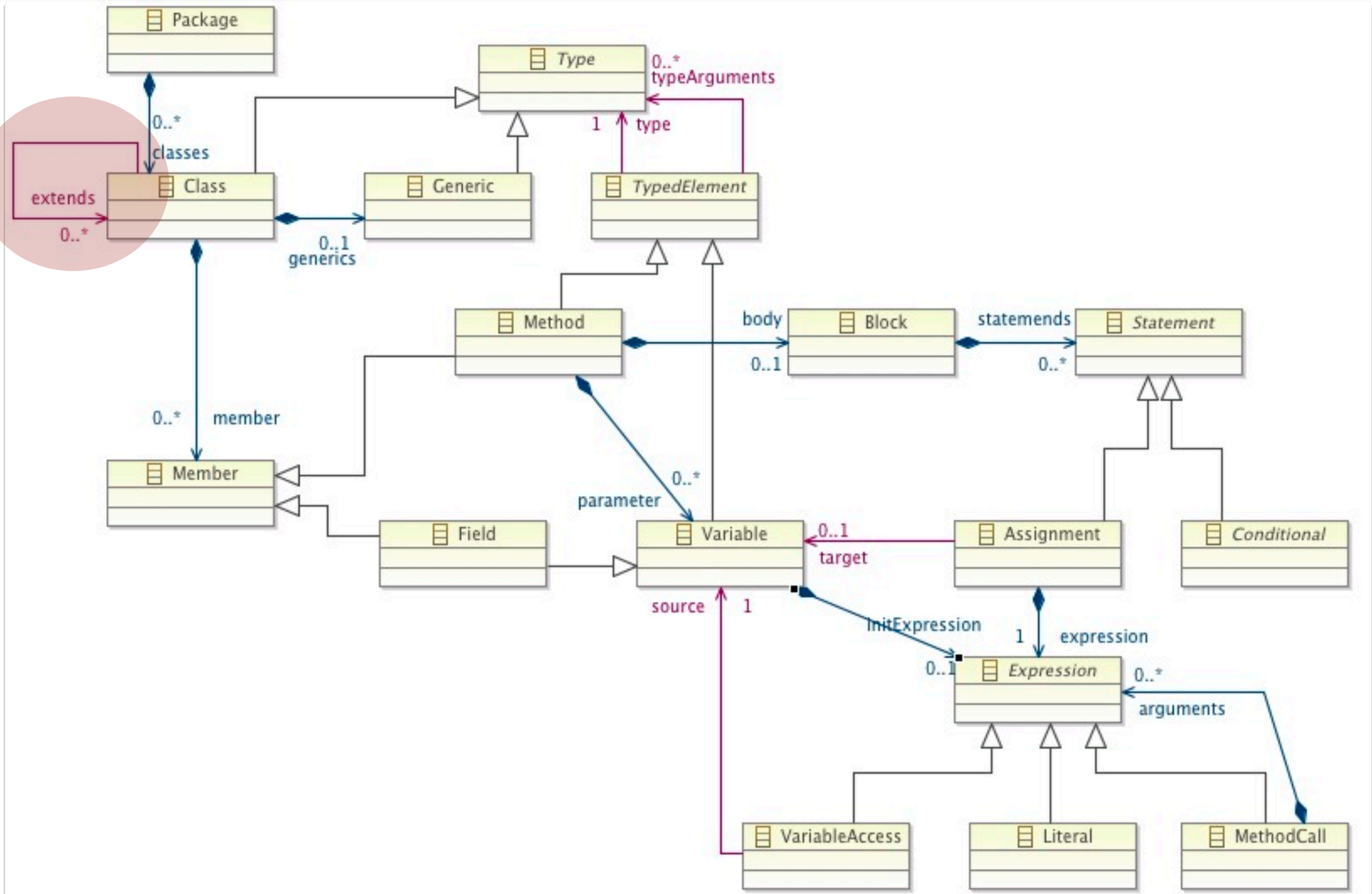
```
String oclDelegateURI = OCLDelegateDomain.OCL_DELEGATE_URI;
EValidator.ValidationDelegate.Registry.INSTANCE.put(
          oclDelegateURI, new OCLValidationDelegateFactory.Global());
```

- Register OCL implementation with EMF
- Not necessary within Eclipse runtime
- OCL Validation supports pure reflective EMF

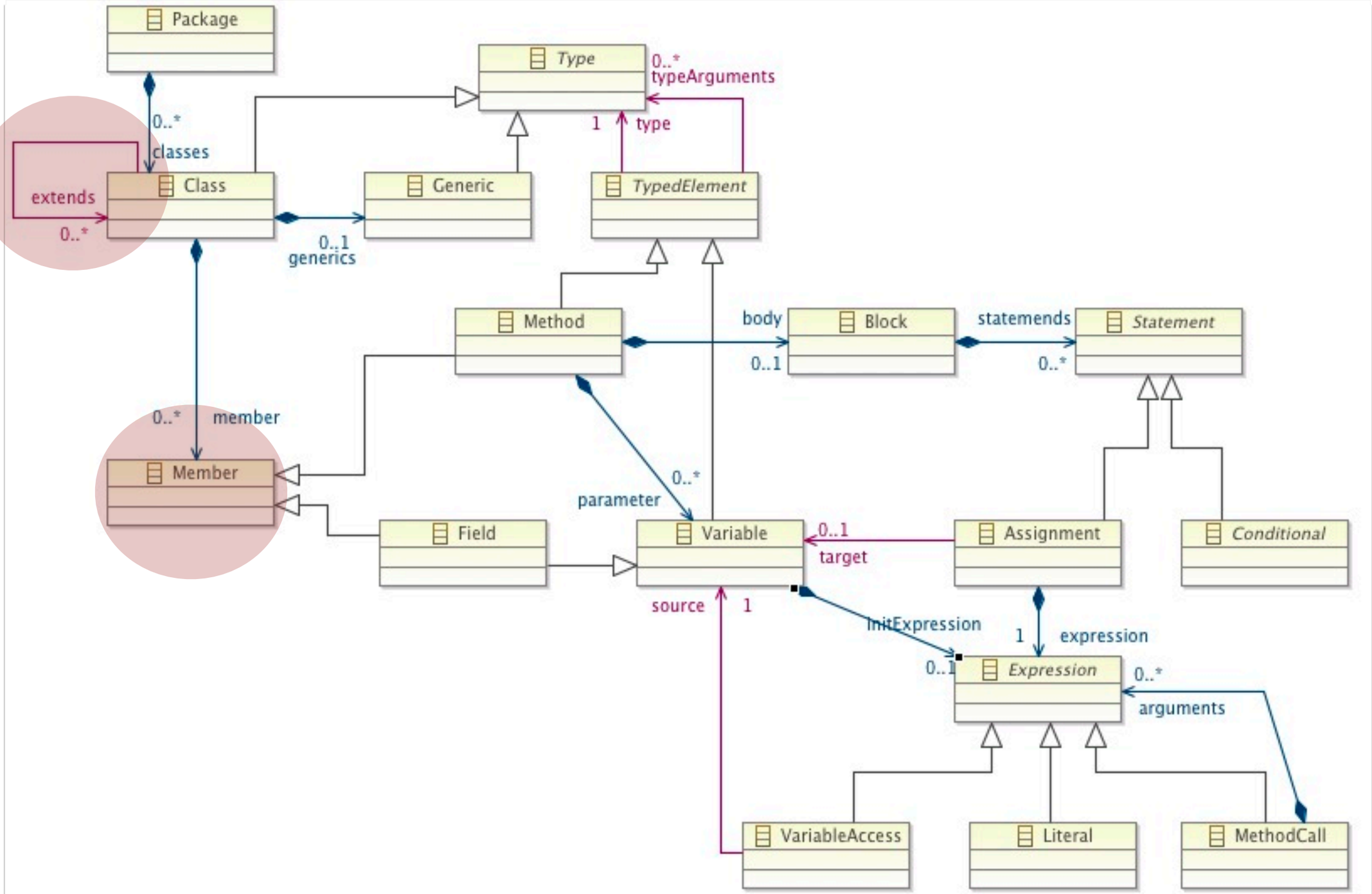
Further Examples for Constraints



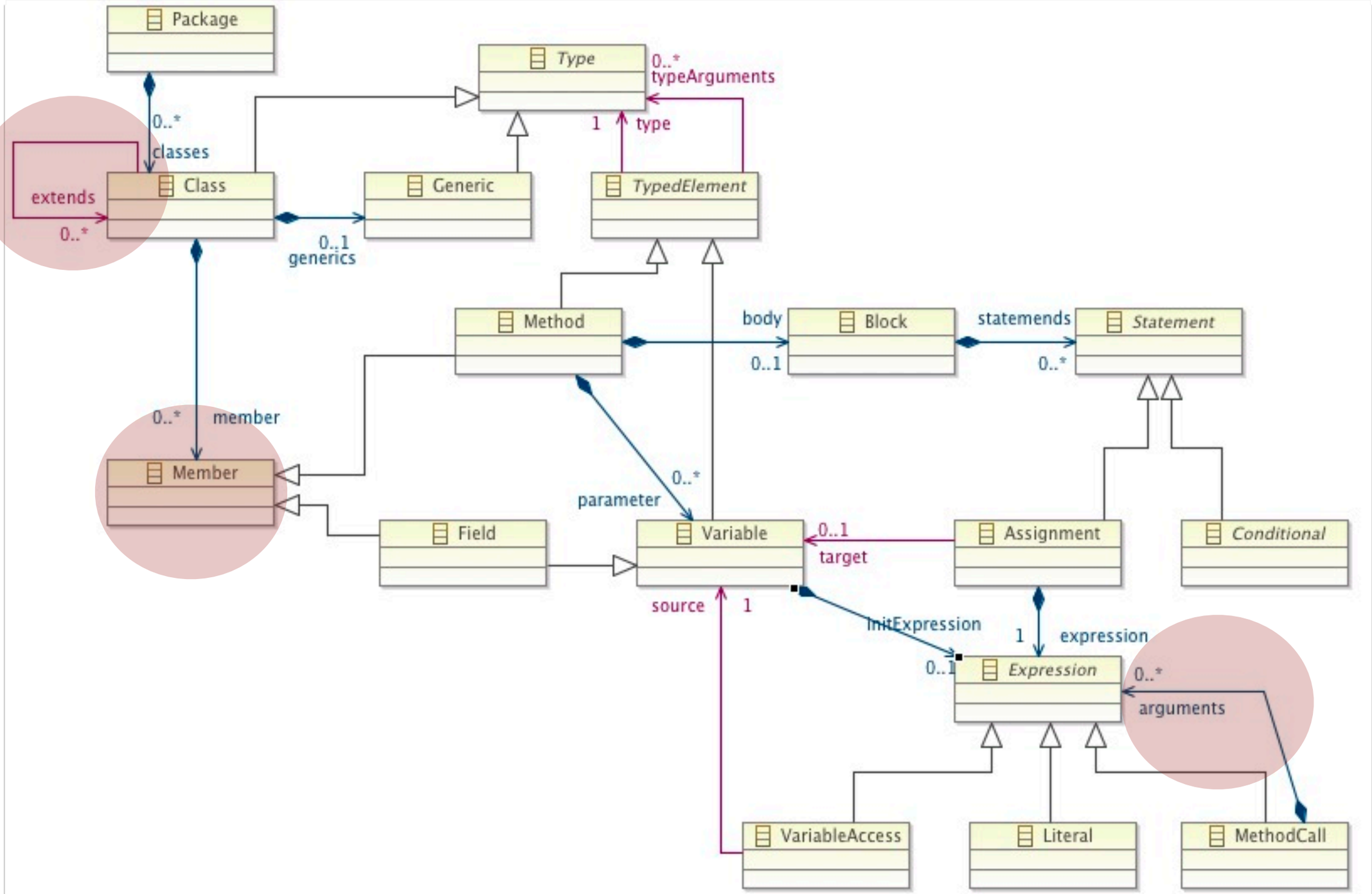
Further Examples for Constraints



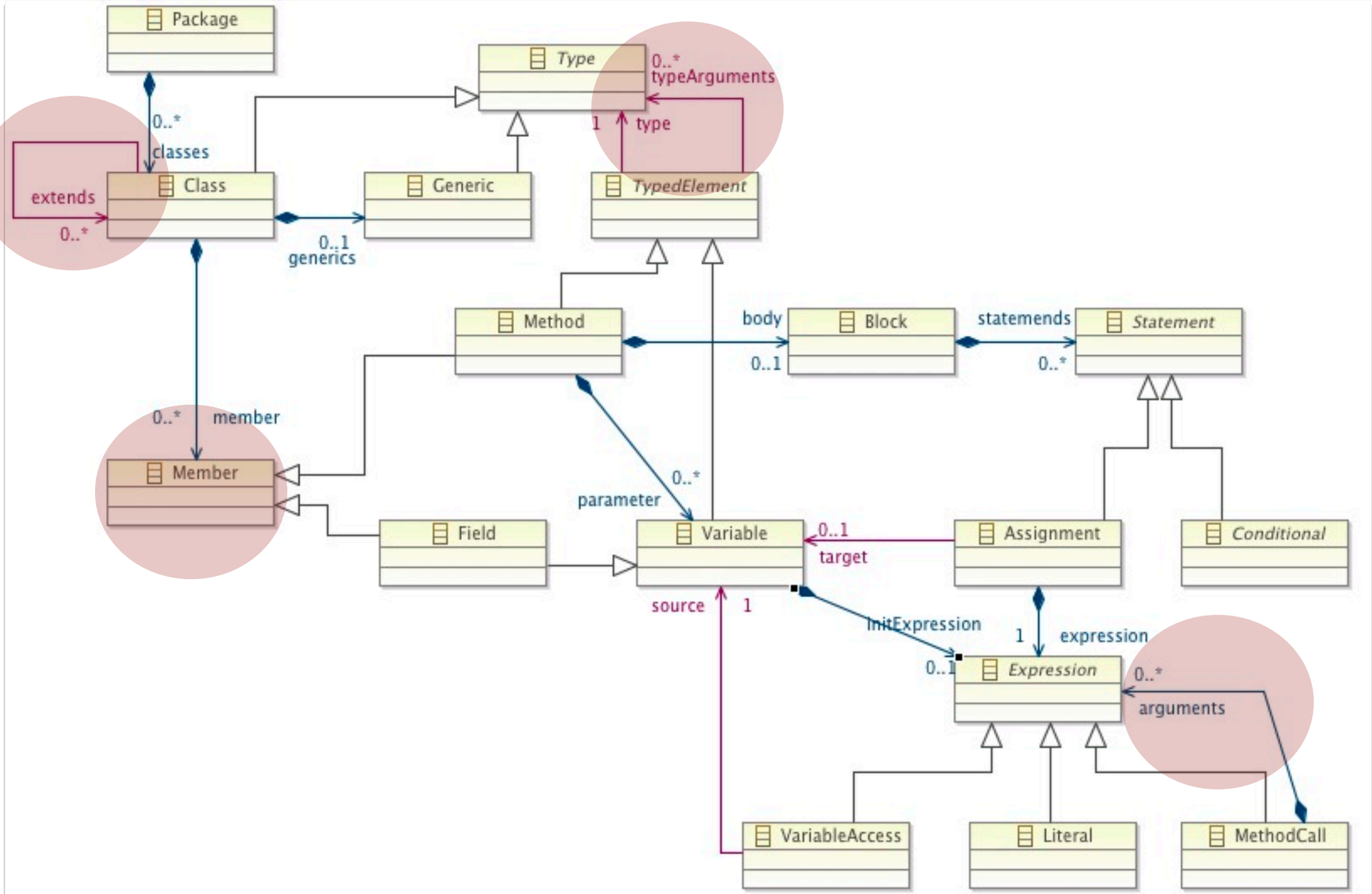
Further Examples for Constraints



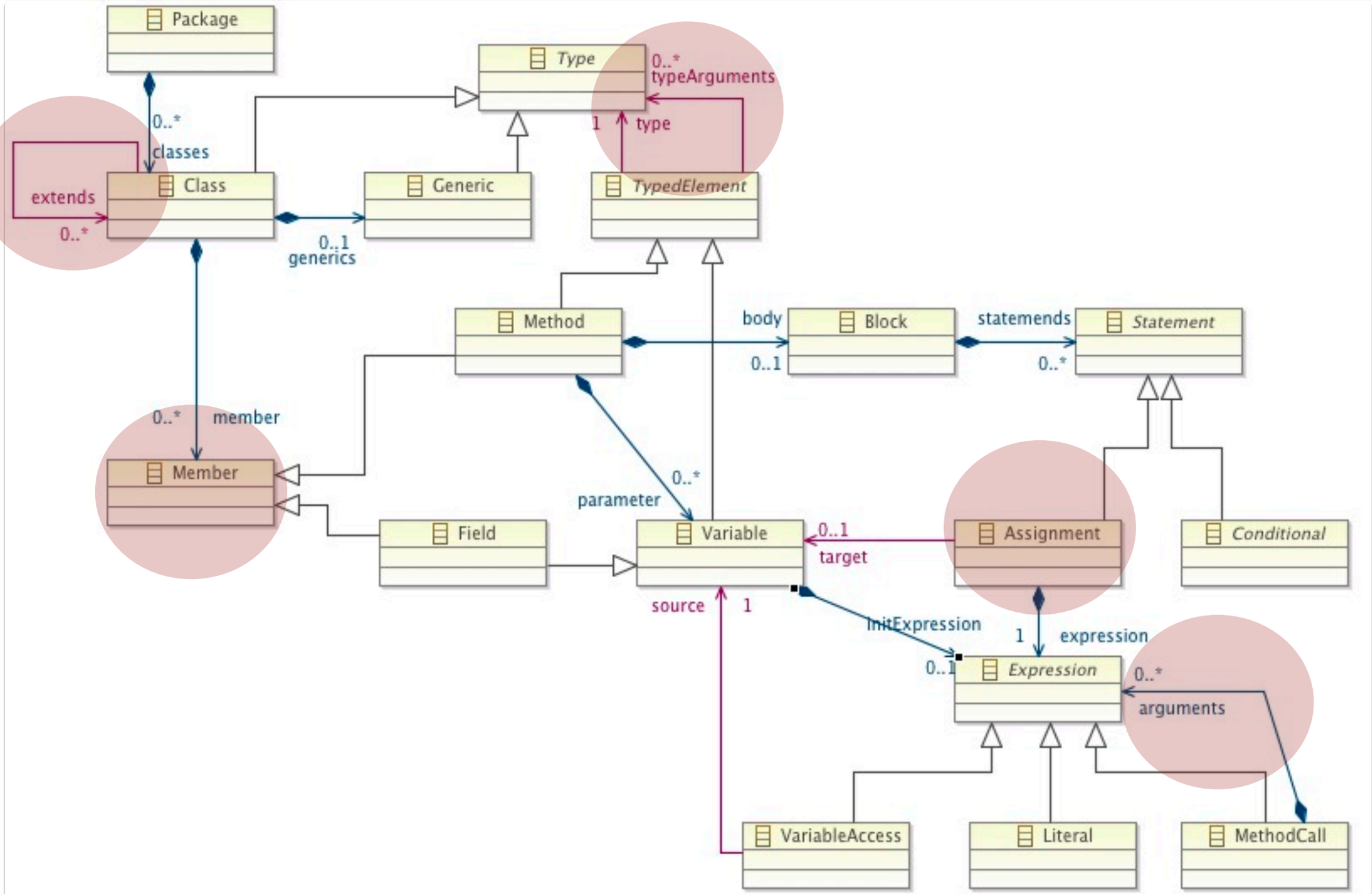
Further Examples for Constraints



Further Examples for Constraints



Further Examples for Constraints



Summary

- Validation for constraints (“rules”) that cannot be expressed in Ecore
- Validation has to be triggered “manually”
- Constraints can be declared in Ecore and implemented in Java or OCL

EMF Tools

- Ecore Tools Diagram Editor

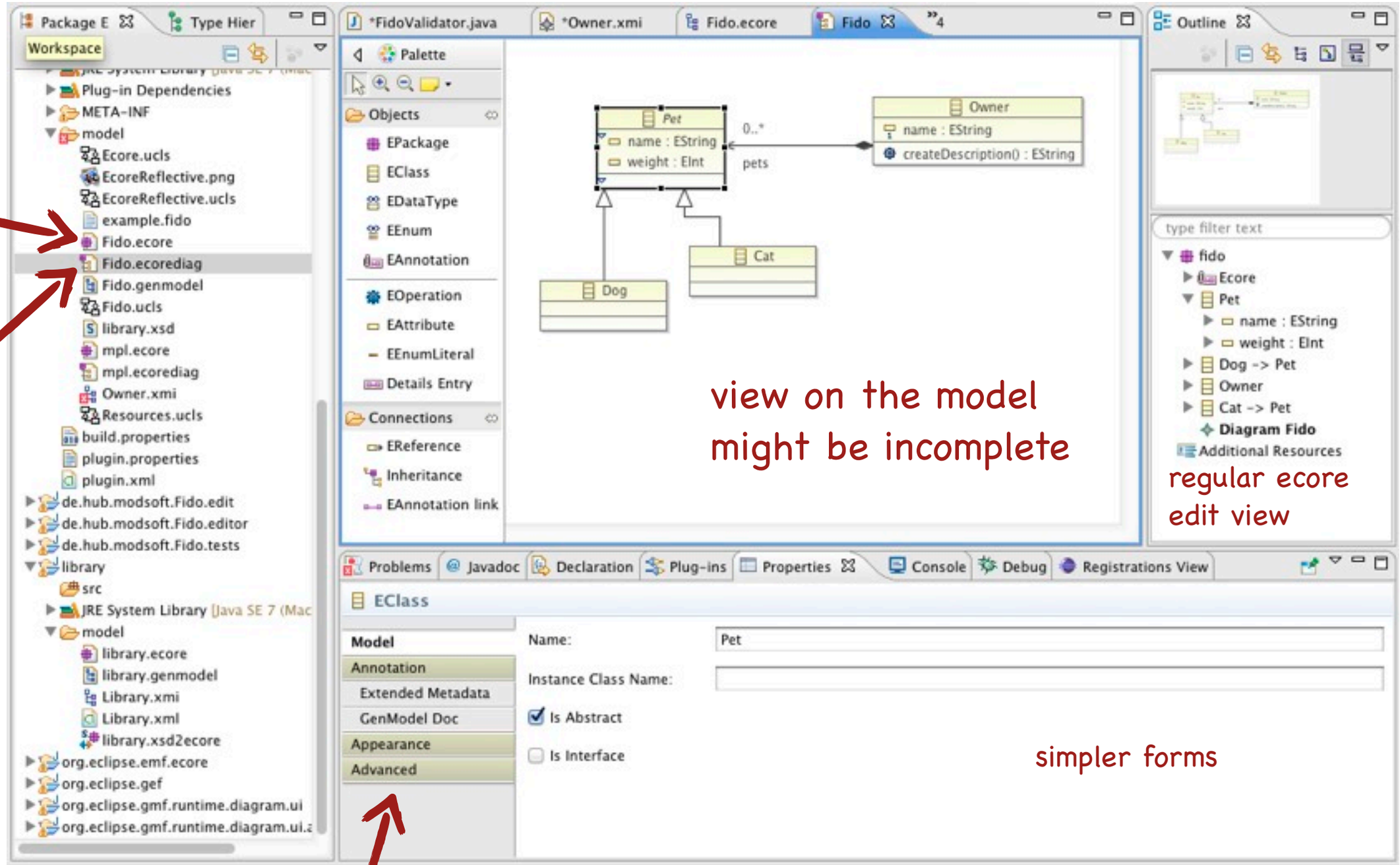
EMF Tools

- Eclipse
  - JDT
  - ...
  - Eclipse Modeling Framework
    - EMF (core)
    - ...
    - EMF Tools
      - Ecore Tools
      - EMF Compare
      - Search
      - EMF Generator Framework
      - Modeling Workflow Engine
      - Tedeo/CDO
      - Texo
      - ...
    - EMF Validation

Diagram Editor

model is saves as regular ecore XMI file

diagram specific data is stored in extra diagram XMI file



view on the model might be incomplete

regular ecore edit view

simpler forms

regular properties view from ecore edit

Summary

- Lots of useful frameworks
- Big mess

Agenda

prolog (1 VL)	**Introduction:** languages and their aspects, modeling vs. programming, meta-modeling and the 4 layer model
0. (2 VL)	**Eclipse/Plug-ins:** eclipse, plug-in model and plug-in description, features, *p2*-repositories, *RCP*s
 1. (2 VL)	**Structure:** *Ecore*, *genmodel*, working with generated code, constraints with *Java* and *OCL*, *XML/XMI*
2. (3 VL)	**Notation:** Customizing the tree-editor, textural with *XText*, graphical with *GEF* and *GMF*
3. (4 VL)	**Semantics:** interpreters with Java, code-generation with *Java* and *XTend*, model-transformations with *Java* and *ATL*
epilog (2 VL)	**Tools:** persisting large models, model versioning and comparison, model evolution and co-adaption, modular languages with *XBase*, *Meta Programming System* (MPS)

Learned so far

- What a meta-model and 4-layer modeling
- How to write Ecore models and Ecore model constraints
- Generate and use Ecore Java code
- Notifications, Serialization, and Edit-framework

- Everything we saw is based on these basics
  - diagram editor
  - text editor for Ecore and OCL
  - .ecore and .genmodel file format