Model-Based Software Engineering

Lecture 04 – OCL and Concrete Syntax

Prof. Dr. Joel Greenyer

SOFTWARE ENGINEERING

April 26, 2016

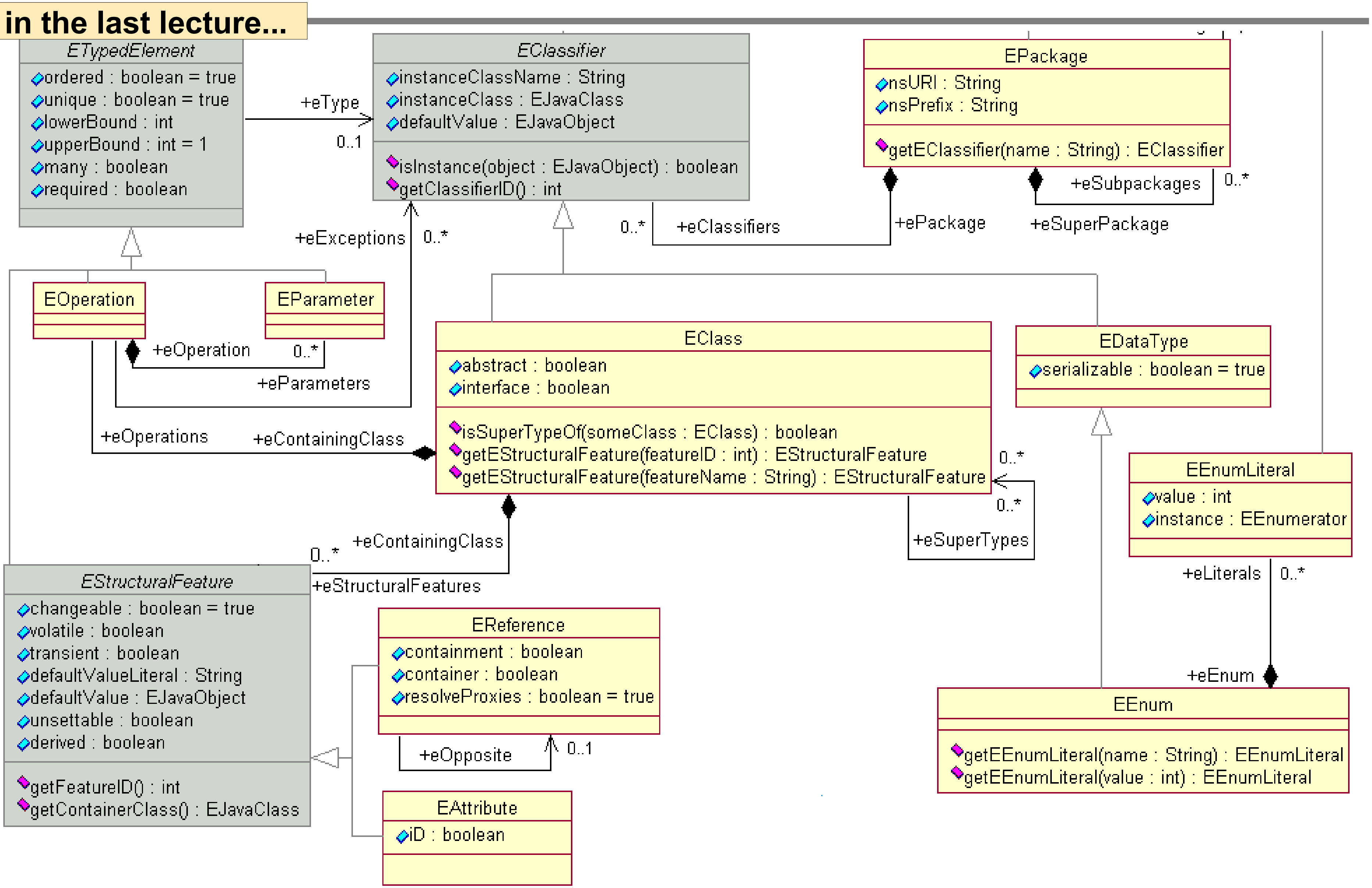
Leibniz Universität Hannover

Acknowledgment

- The slides of this lecture are inspired by lecture slides from
  - *Ekkart Kindler*: Course on Advanced Topics in Software Engineering, DTU Compute, 2015.
    - http://www2.imm.dtu.dk/courses/02265/f15/schedule.shtml
  - *Ina Schäfer, Christoph Seidl*: Modellbasierte Software-entwicklung, TU Braunschweig, 2015.
  - *Steffen Becker*: Model-Driven Software Development, Universität Paderborn, 2013
  - The Eclipse Open Model CourseWare (OMCW) Project:
    - https://eclipse.org/gmt/omcw/

A Close Look at the Ecore Meta-Metamodel

in the last lecture...



ETypedElement
ordered : boolean = true
unique : boolean = true
lowerBound : int
upperBound : int = 1
many : boolean
required : boolean

EClassifier
instanceClassName : String
instanceClass : EJavaClass
defaultValue : EJavaObject
isInstance(object : EJavaObject) : boolean
getClassifierID() : int

EPackage
nsURI : String
nsPrefix : String
getEClassifier(name : String) : EClassifier

EOperation

EParameter

EClass
abstract : boolean
interface : boolean
isSuperTypeOf(someClass : EClass) : boolean
getEStructuralFeature(featureID : int) : EStructuralFeature
getEStructuralFeature(featureName : String) : EStructuralFeature

EDataType
serializable : boolean = true

EEnumLiteral
value : int
instance : EEnumerator

EStructuralFeature
changeable : boolean = true
volatile : boolean
transient : boolean
defaultValueLiteral : String
defaultValue : EJavaObject
unsettable : boolean
derived : boolean
getFeatureID() : int
getContainerClass() : EJavaClass

EReference
containment : boolean
container : boolean
resolveProxies : boolean = true

EAttribute
iD : boolean

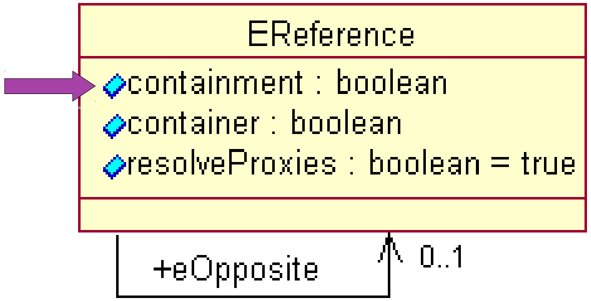
EEnum
getEEnumLiteral(name : String) : EEnumLiteral
getEEnumLiteral(value : int) : EEnumLiteral

see http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html

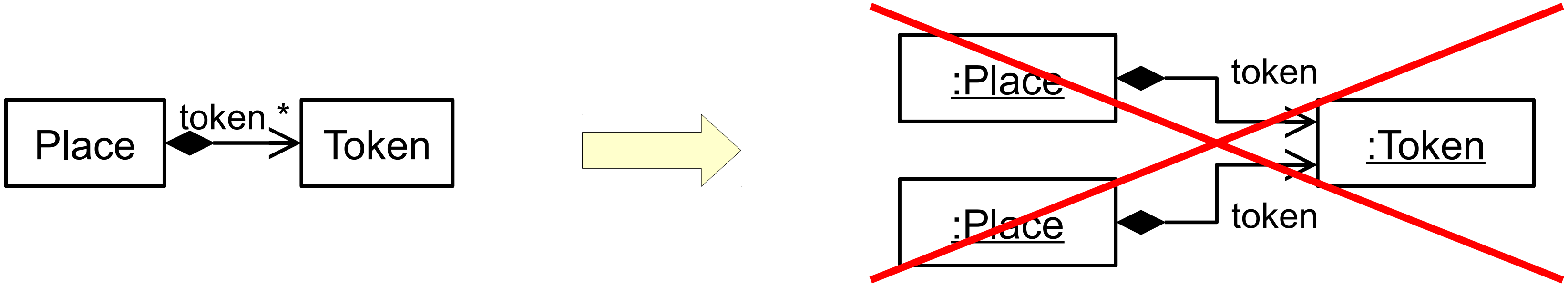
Specialties of EReferences

in the last lecture...

- **Containment**:
  - An object can only be be contained in *at most one* other object at a time
    - it can be target of at most one containment link at a time



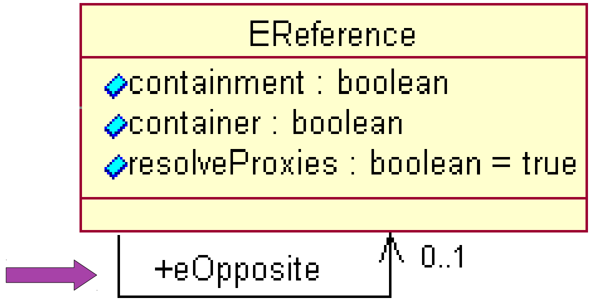
- Example:



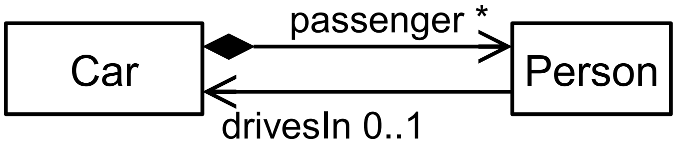
Specialties of EReferences

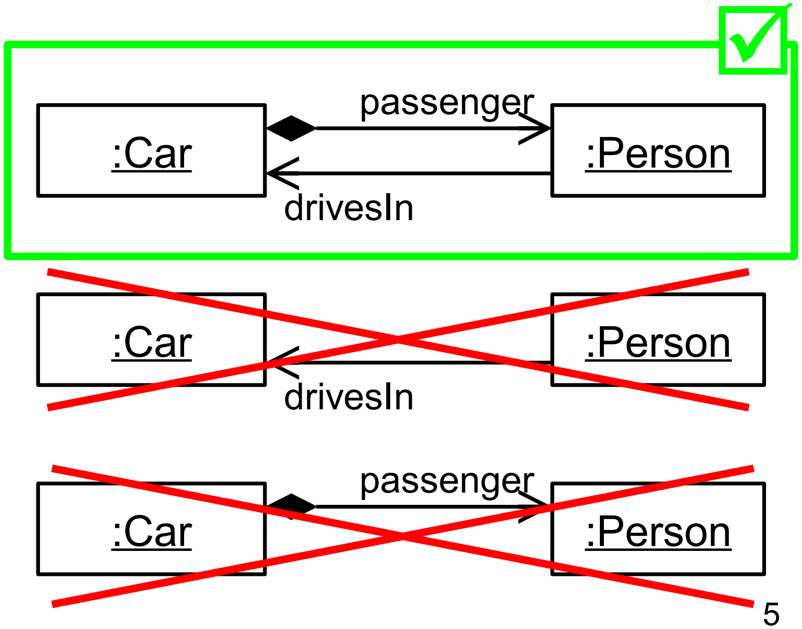
in the last lecture...

- **eOpposite**:
  - Two EReferences in opposite directions between two EClasses can be “opposites”
  - Thereby forming a bidirectional relationship
  - At the object level, there must be bidirectional links



- Example:





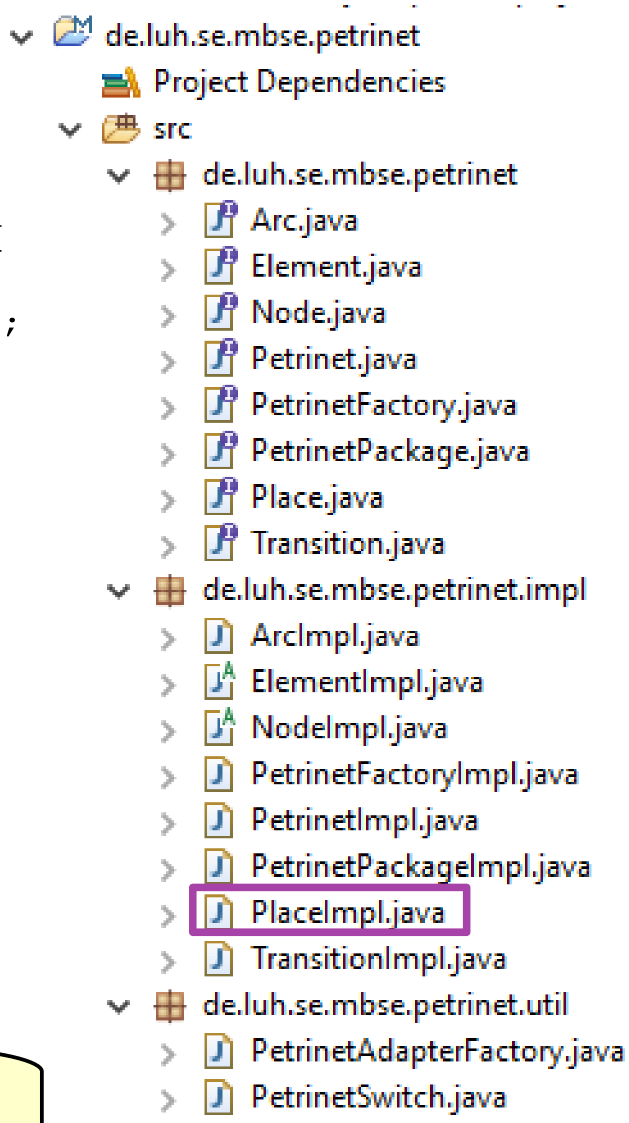
EMF Code Generation

in the last lecture...

- Separation of Interfaces and Implementation

```
public class PlaceImpl extends NodeImpl implements Place {

    protected static final int INTIAL_MARKING_EDEFAULT = 0;
    protected int intialMarking = INTIAL_MARKING_EDEFAULT;

    public int getIntialMarking() {
         return intialMarking;
    }

    public void setIntialMarking(int newIntialMarking) {
         int oldIntialMarking = intialMarking;
         intialMarking = newIntialMarking;
         if (eNotificationRequired())
                  eNotify(new ENotificationImpl(this,
                  Notification.SET,
                  PetrinetPackage.PLACE__INTIAL_MARKING,
                  oldIntialMarking,
                  intialMarking));
    }

    ...

} //PlaceImpl
```

Notification mechanism (observer pattern) built in



```
de.luh.se.mbse.petrinet
  Project Dependencies
  src
    de.luh.se.mbse.petrinet
      Arc.java
      Element.java
      Node.java
      Petrinet.java
      PetrinetFactory.java
      PetrinetPackage.java
      Place.java
      Transition.java
    de.luh.se.mbse.petrinet.impl
      ArcImpl.java
      ElementImpl.java
      NodeImpl.java
      PetrinetFactoryImpl.java
      PetrinetImpl.java
      PetrinetPackageImpl.java
      PlaceImpl.java
      TransitionImpl.java
    de.luh.se.mbse.petrinet.util
      PetrinetAdapterFactory.java
      PetrinetSwitch.java
```

Dynamic EMF

- The generated code allows us to create instances of our Ecore models
  - for example (factory method):

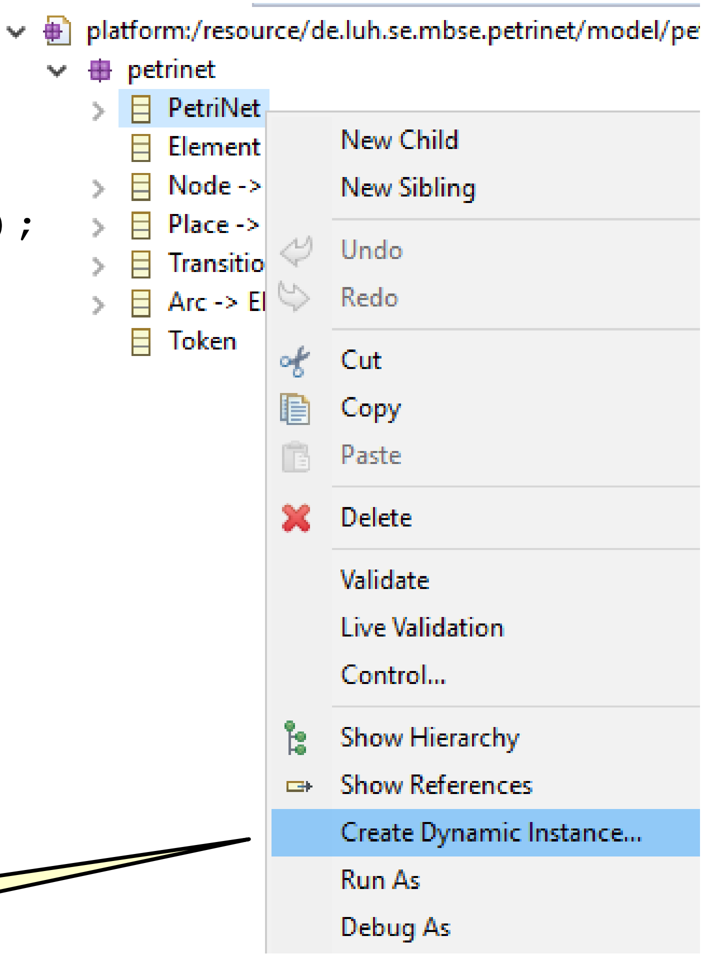
```
public Petrinet createPetrinet() {
   PetrinetImpl petrinet = new PetrinetImpl();
   return petrinet;
}
```

Dynamic EMF

- The generated code allows us to create instances of our Ecore models
  - for example (factory method):

```
public Petrinet createPetrinet() {
    PetrinetImpl petrinet = new PetrinetImpl();
    return petrinet;
}
```

- But EMF also supports working with **dynamic instances**
- EMF **interprets** the metamodels to allow us to work on instance models without code generation:



creating a dynamic object via the UI

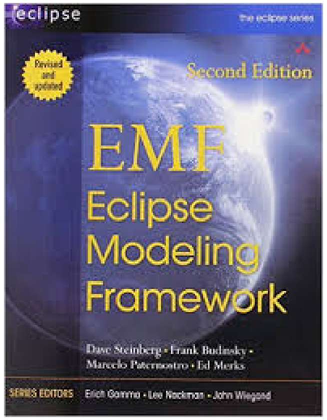
Dynamic EMF

- creating models and instances dynamically via the API:

```
// create package
EPackage petrinetPackage = EcoreFactory.eINSTANCE.createEPackage();

//create Place class
EClass placeClass = EcoreFactory.eINSTANCE.createEClass();
placeClass.setName("Place");
petrinetPackage.getEClassifiers().add(placeClass);

//create initialMarkings attribute and add it to the Place class
EAttribute initialMarkingsAttribute
               = EcoreFactory.eINSTANCE.createEAttribute();
initialMarkingsAttribute.setName("initialMarkings");
initialMarkingsAttribute.setEType(EcorePackage.eINSTANCE.getEInt());
placeClass.getEAttributes().add(initialMarkingsAttribute);

//create dynamic instance of Place class
EFactory petrinetFactory = petrinetPackage.getEFactoryInstance();
EObject place =  petrinetFactory.create(placeClass);
place.eSet(initialMarkingsAttribute, 2);
```

EMF Resources

- D. Steinberg, F. Budinski, M. Paternostro, E.Merks: EMF: Eclipse Modeling Framework, Addison Wesley, 2nd edition, 2008.


eclipse
the eclipse series
Revised and updated
Second Edition
EMF
Eclipse Modeling Framework
Dave Steinberg • Frank Budinsky • Marcelo Paternostro • Ed Merks
SERIES EDITORS Erich Gamma • Lee Nackman • John Wiegand

- Online resources
  - http://www.vogella.com/tutorials/EclipseEMF/article.html
  - http://eclipsesource.com/blogs/tutorials/emf-tutorial/
  - There are many more online resources...

Model-Based Software Engineering

Lecture 04 – OCL and Concrete Syntax

Prof. Dr. Joel Greenyer

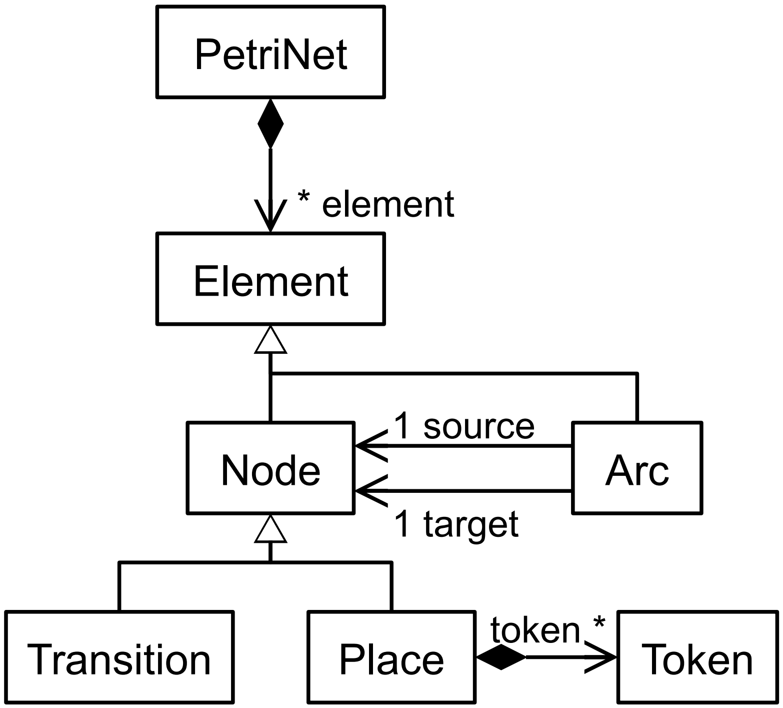
SOFTWARE ENGINEERING

April 26, 2016

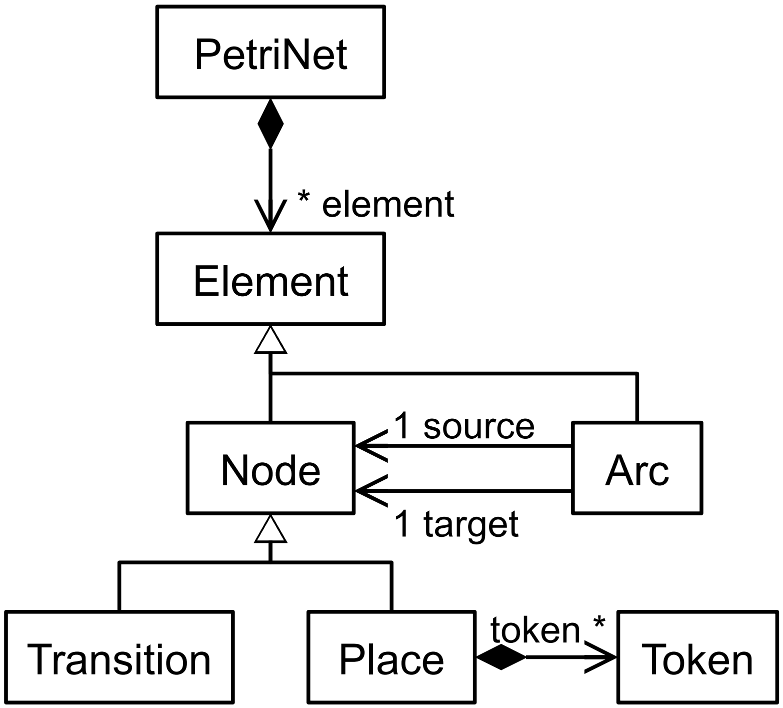
Leibniz Universität Hannover

3.1. Introduction to OCL

What's missing?



What's missing?



PetriNet

* element

Element

Node

1 source

1 target

Arc

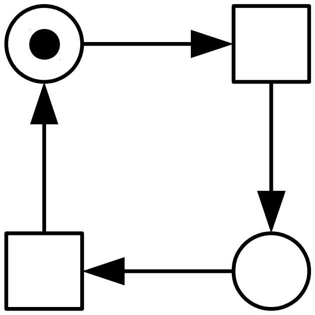
Transition

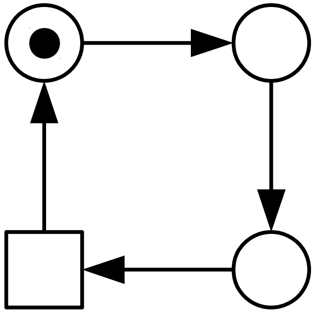
Place

token

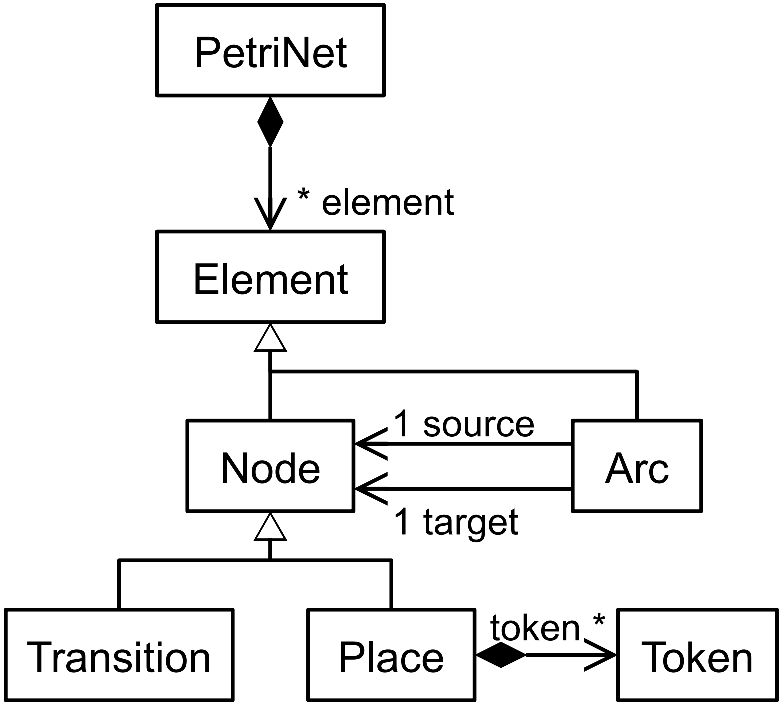
*

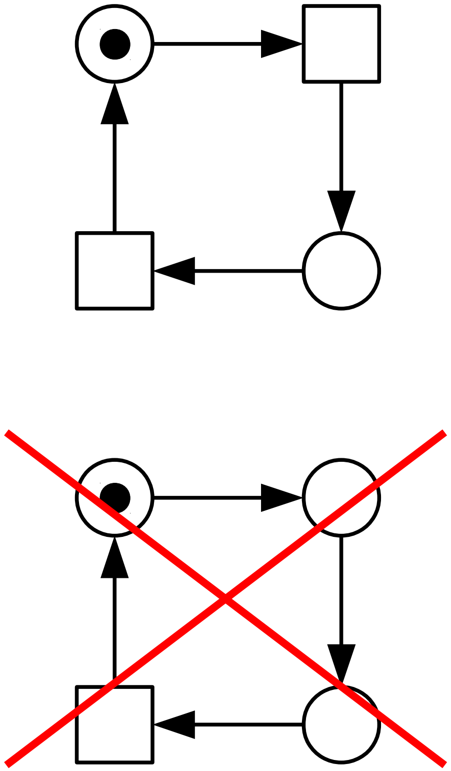
Token





What's missing?

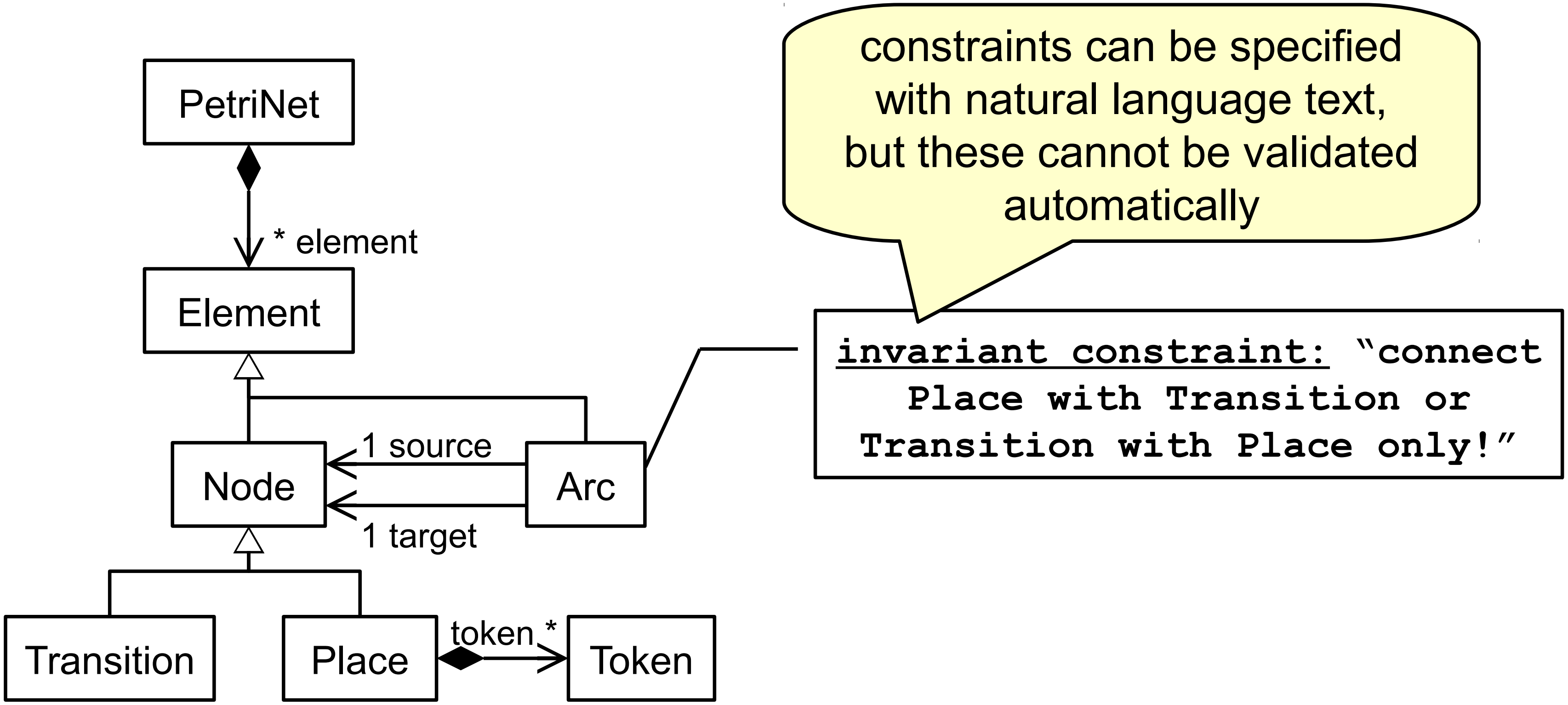




Obviously something is missing in the Petri net metamodel!

That is not a valid Petri net!: An Arc must only connect Places to Transitions or Transitions for Places

What's missing?



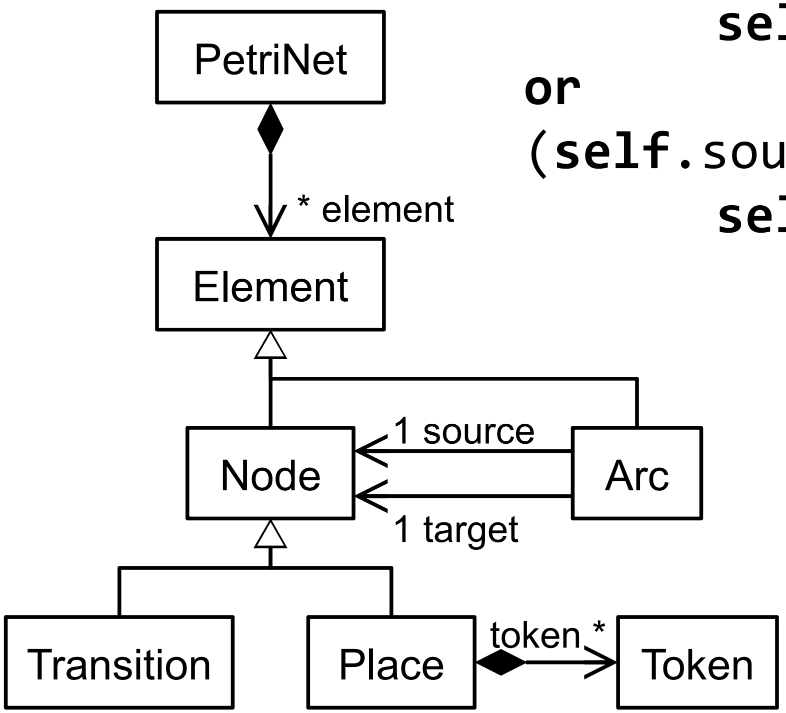
constraints can be specified with natural language text, but these cannot be validated automatically

invariant constraint: "connect Place with Transition or Transition with Place only!"

OCL – Example

- Invariant constraint written in the **Object Constraint Language (OCL):**

```
context Arc
inv “No Arcs Between Nodes Of The Same Kind”:
((self.source.oclIsKindOf(Place) and
       self.target.oclIsKindOf(Transition))
or
(self.source.oclIsKindOf(Transition) and
       self.target.oclIsKindOf(Place) ) );
```



OCL – Example

- The **Object Constraint Language (OCL)** is a formal textual language that allows us to specify **constraints** and **queries** on models with a MOF-style metamodel (UML, MOF, ...)
  - OMG standard: http://www.omg.org/spec/OCL/

- The OCL language and an interpreter are also implemented for EMF

- OCL is used in many other standards to express constraints: MOF, UML, QVT, …

OCL

- The **Object Constraint Language (OCL)** has been developed to achieve the following goals:
  - to be formal, precise, unambiguous
  - to be applicable for a large number of users (business or system modeler, programmers)
  - to be a constraint and query language, <u>not</u> a programming language
  - to be tool supported

OCL

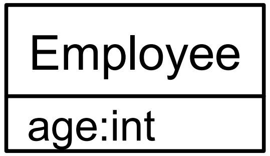
- OCL constraints and queries have **no side-effects**
- The **evaluation** of an OCL expression **returns a value**
  - multiple **types** are supported: we get to them shortly
  - When an **invariant constraint: Boolean**
- OCL is **not a programming language**
  - no program logic or flow control
  - no invocation of processes or activation of non-query operations
- OCL is a **typed language**
  - Each classifier in the model represents a distinct OCL type
    - we can define variables typed over classifiers in the model
  - Includes a set of predefined types

OCL

- OCL can be used
  - as a **query language**
  - to **specify invariants** on classes and types in a class model
  - to describe **pre- and post conditions** on operations
  - to describe **guards** (in UML behavior models)
  - to specify **derivation rules** for **derived features** (attributes or references/associations)

OCL

- Each OCL expression is related to an object, the instance of a class
  - A **context declaration** is used to determine the class
- **`self`** refers to the contextual instance
- Example:



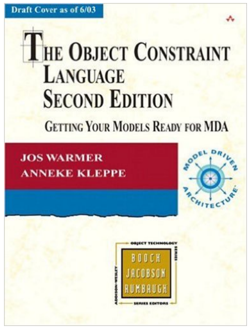
`context`: (an instance of) Employee

```
context Employee
inv: self.age >= 19
```

`inv`: an invariant constraint; must be true for all instances of the context class (here: for all Employee instances)

Resources

- Jordi Cabot, Martin Gogolla: Object Constraint Language (OCL): A Definitive Guide, in Formal Methods for Model-Driven Engineering, Volume 7320 of Lecture Notes in Computer Science, pp 58-90, 2012.
  - http://link.springer.com/chapter/10.1007%2F978-3-642-30982-3_3
  - http://modeling-languages.com/wp-content/uploads/2012/03/OCLChapter.pdf
- Jos Warmer, Anneke Kleppe: The Object Constraint Language: Getting Your Models Ready for MDA, Addison-Wesley Professional; 2nd edition, 2003.



Draft Cover as of 6/03

THE OBJECT CONSTRAINT LANGUAGE
SECOND EDITION
GETTING YOUR MODELS READY FOR MDA

JOS WARMER
ANNEKE KLEPPE

MODEL DRIVEN ARCHITECTURE

BOOCH
JACOBSON
RUMBAUGH

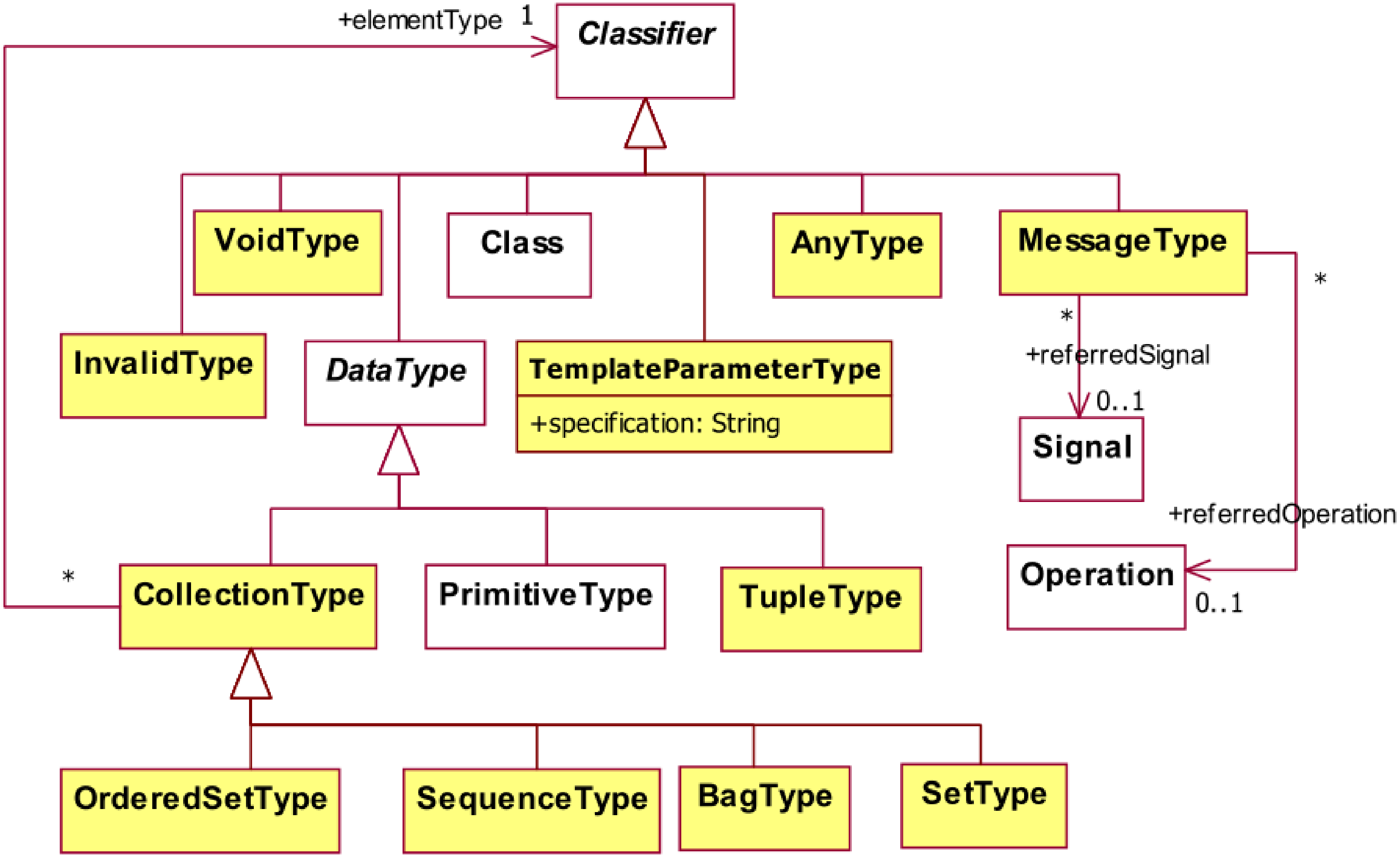
- Christian Hein, A presentation of OCL 2, Open Model CourseWare, 2006
  - https://eclipse.org/gmt/omcw/resources/chapter01/downloads/OCL2.Fraunhofer.ppt

3.2. OCL types

OCL Types

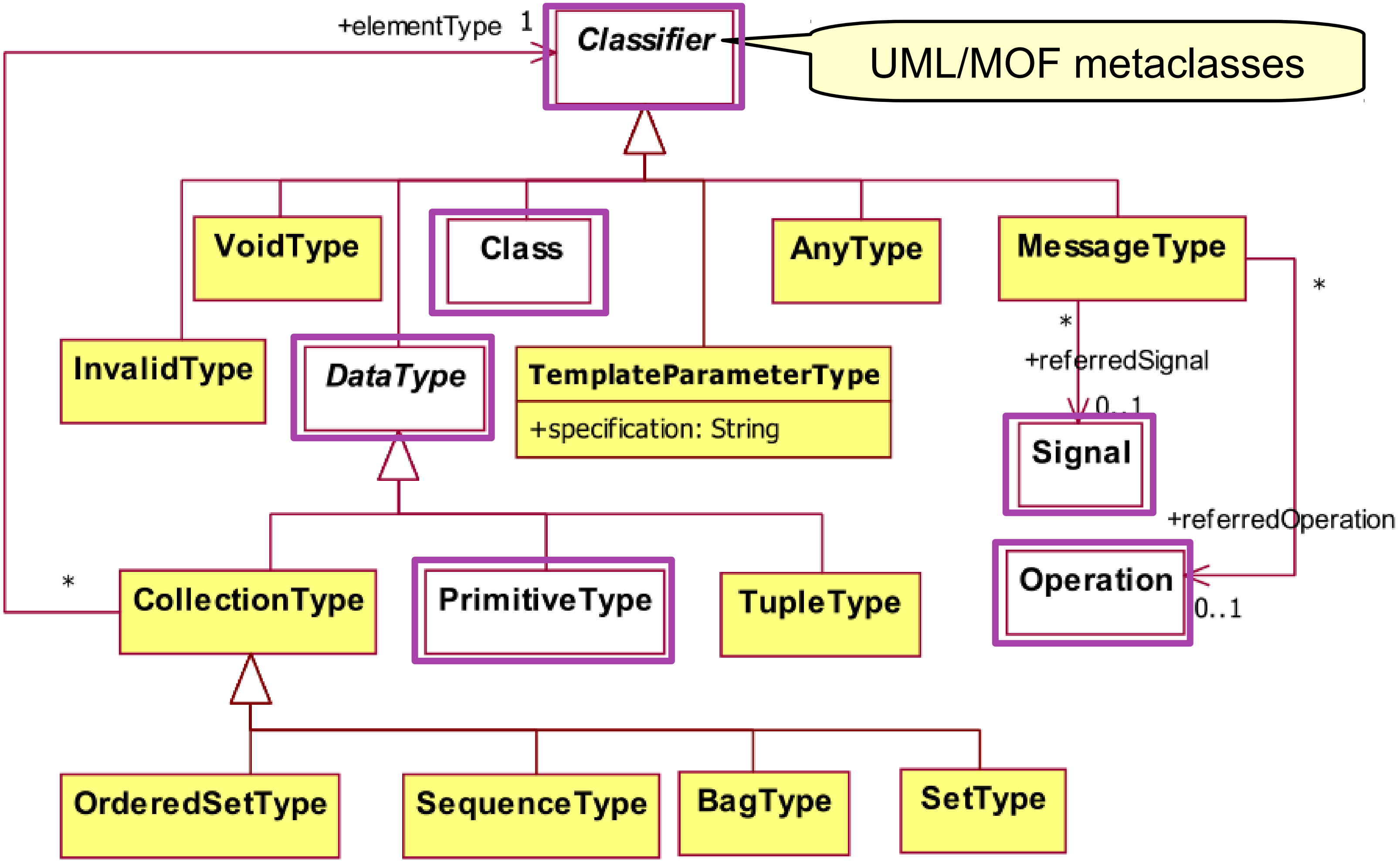
- OCL is a typed language
  - queries evaluate to values of certain types
  - we can work with variables of certain types
  - different types offer different functions
    - for example `collection->forAll(...)`

OCL Types Metamodel



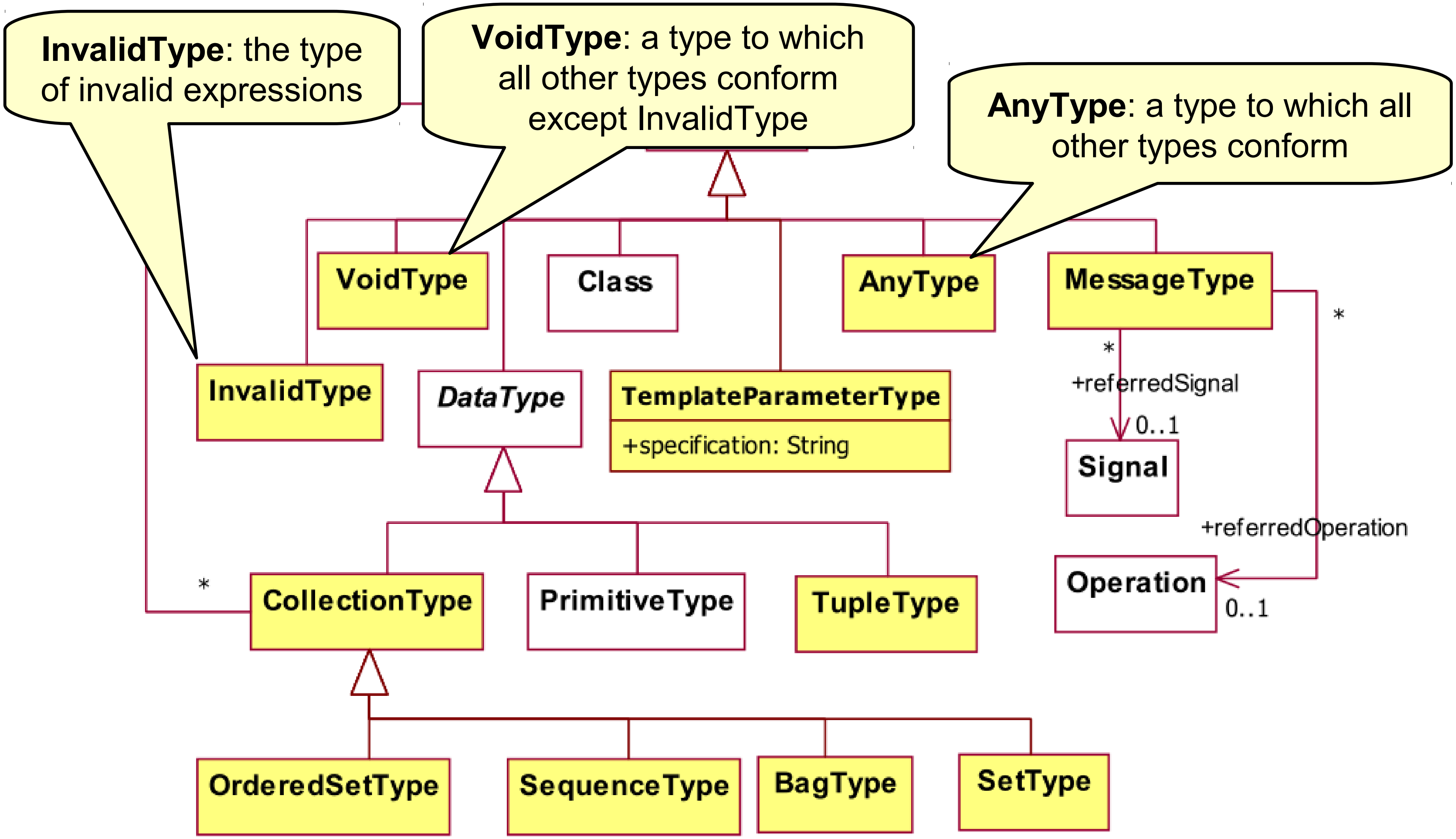
from http://www.omg.org/spec/OCL/2.4/PDF (Figure 8.1)

OCL Types Metamodel



from http://www.omg.org/spec/OCL/2.4/PDF (Figure 8.1)

OCL Types Metamodel



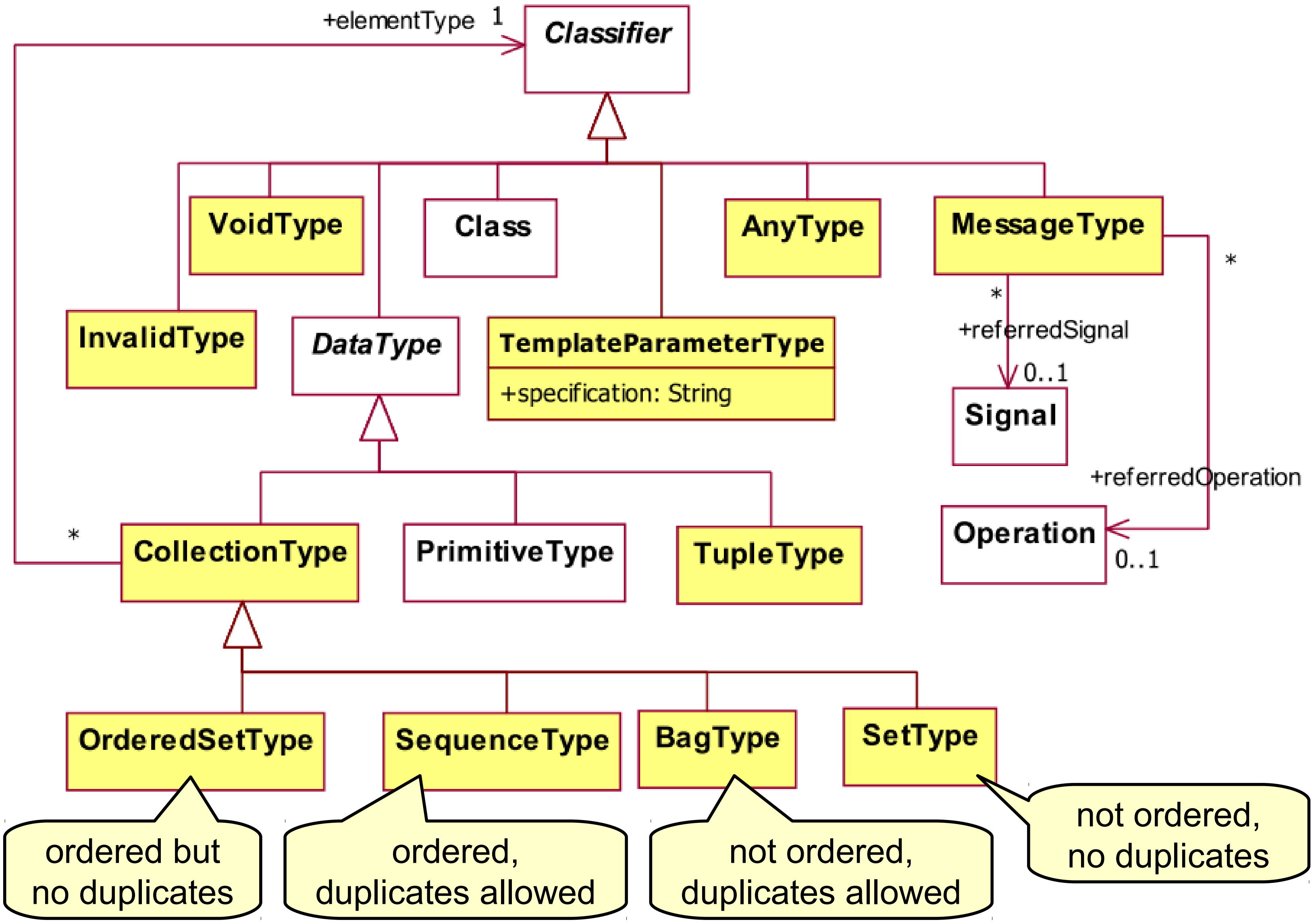
InvalidType: the type of invalid expressions

VoidType: a type to which all other types conform except InvalidType

AnyType: a type to which all other types conform

from http://www.omg.org/spec/OCL/2.4/PDF (Figure 8.1)

OCL Types Metamodel



ordered but no duplicates

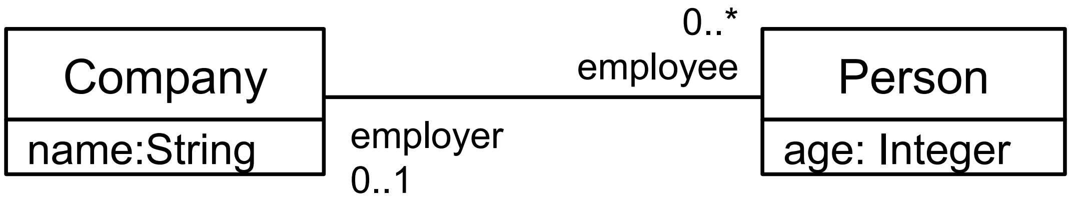
ordered, duplicates allowed

not ordered, duplicates allowed

not ordered, no duplicates

Operations on Collection Types

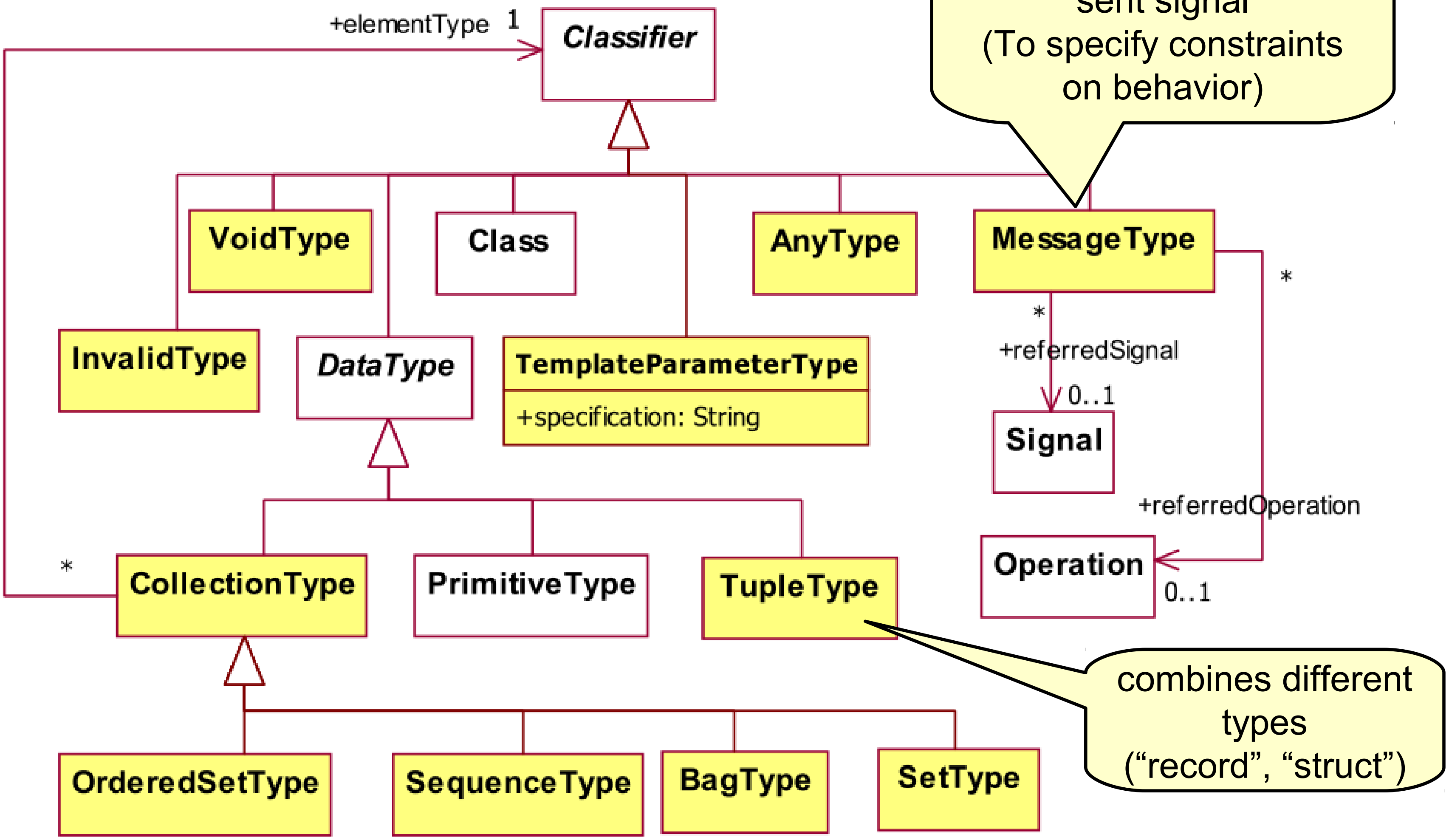
- For example:



```
context Company
inv:   self.employee->forAll( age <= 65 )
inv:   self.employee->forAll( p | p.age <= 65 )
inv:   self.employee->forAll( p : Person | p.age <= 65 )
```

from http://www.omg.org/spec/OCL/2.4/PDF

OCL Types Metamodel

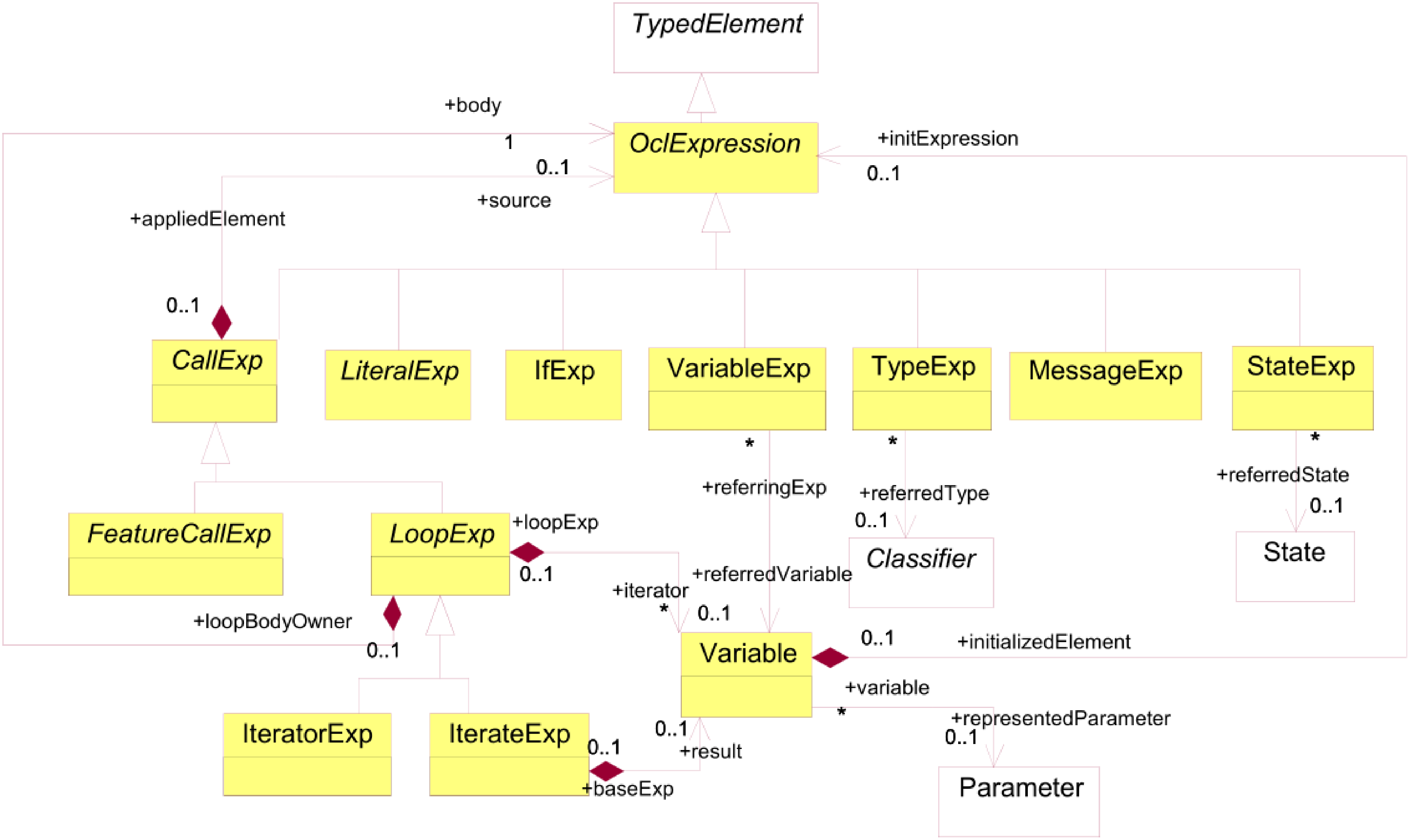


a called operation or a sent signal (To specify constraints on behavior)

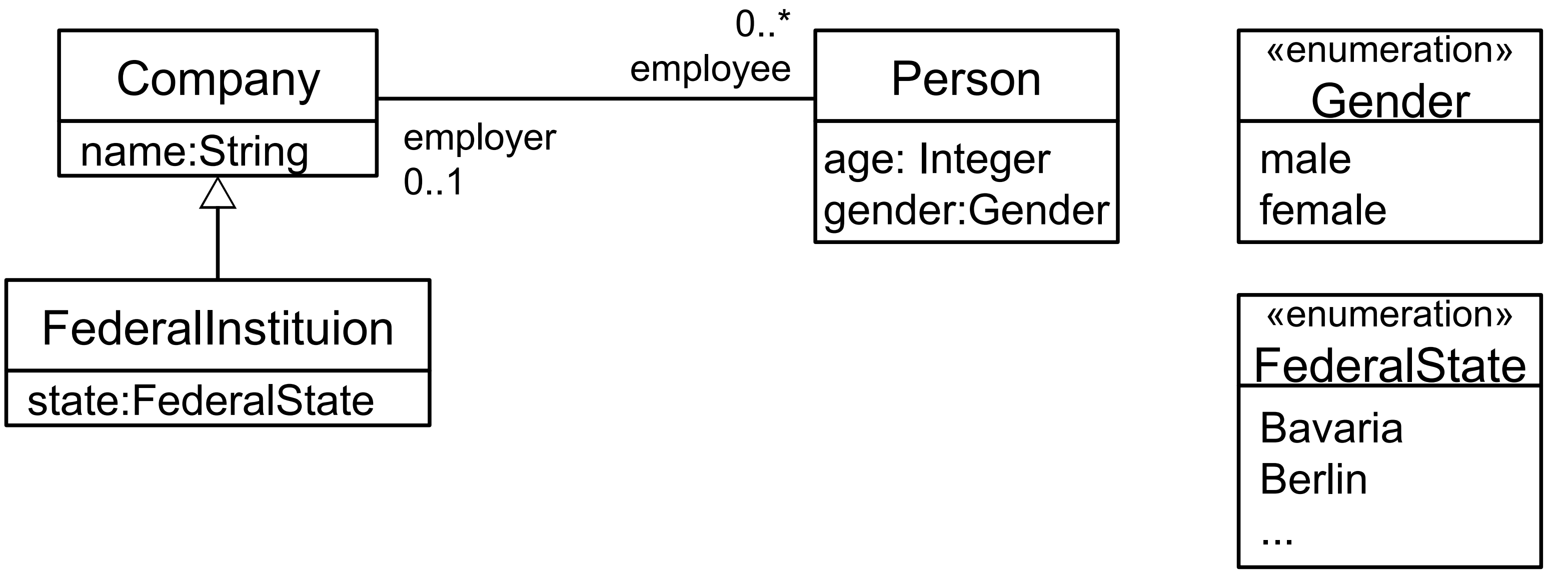
combines different types (“record”, “struct”)

3.2. OCL expressions

OCL Expressions



Accessing Objects and Properties

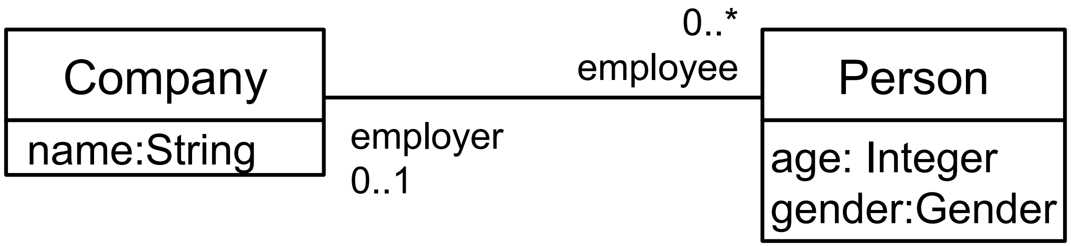


```
context Person
inv:    self.age > 18
inv:    self.gender <> Gender::male
inv:    self.employer.oclAsType(FederalInstitution).state =
        FederalState::LowerSaxony
```

cast

Collection Operations: Select and Reject

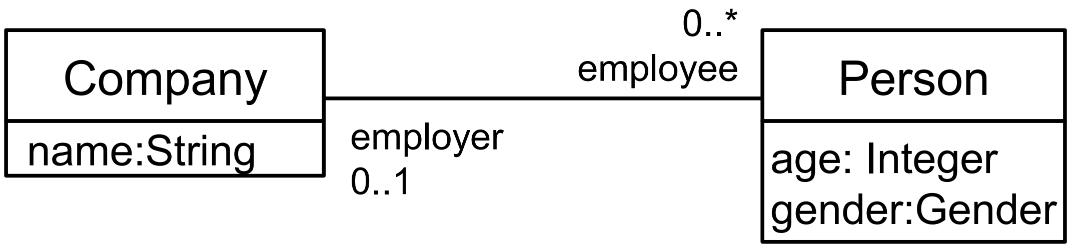
- **select** and **reject** are operations on collections to specify subsets
  - **select**: filters elements conforming to a condition
  - **reject**: excludes elements conforming to a condition
  - result type: same as original



```
context Company
inv:   self.employee->select( age > 65 )->isEmpty()
```

Collection Operations: Collect

- **collect** operations specify a collection derived from some other collection
  - result type: Bag

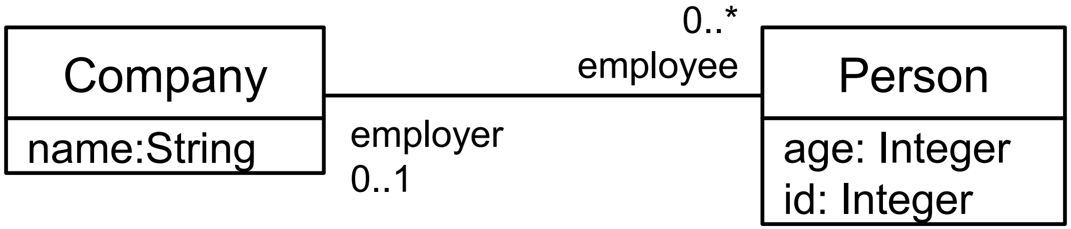


```
context Company
self.employee->collect(age)
```

returns a bag of integers, for example [32, 55, 43, 32, 27]

Collection Operations: ForAll

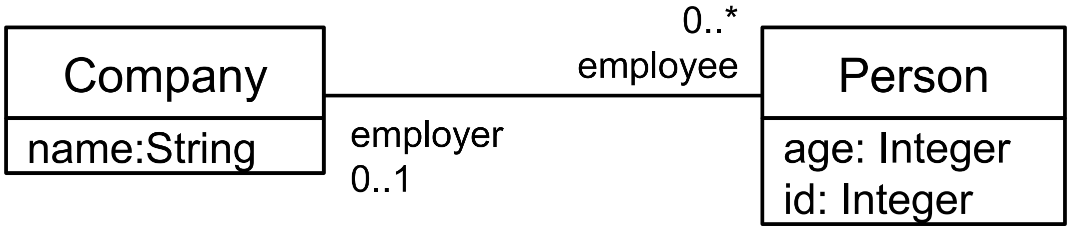
- A **forAll** operation specifies a condition that must hold for all objects in a collection
  - result type: Boolean



```
context Company
inv:   self.employee->forAll( age <= 65 )
inv:   self.employee->forAll( p | p.age <= 65 )
inv:   self.employee->forAll( p : Person | p.age <= 65 )
inv:   self.employee->forAll( p1 |
             self.employee->forAll( p2 |
                   p1 <> p2 implies p1.id <> p2.id ))
```

Collection Operations: Exists

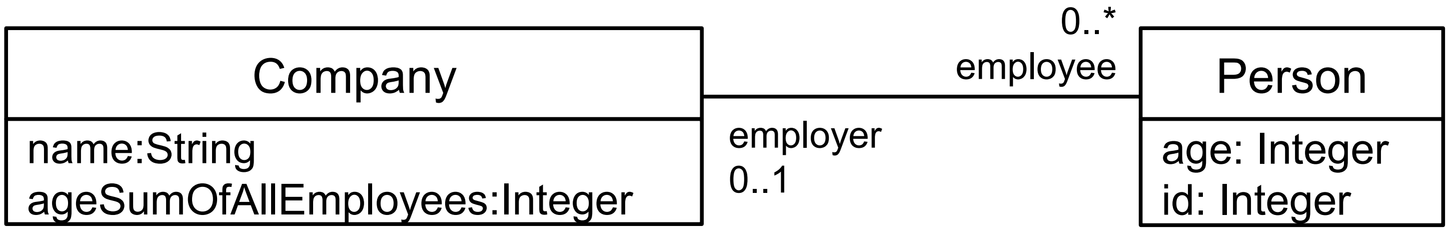
- An **exists** operation specifies a condition that must hold for at least one object in a collection
  - result type: Boolean



```
context Company
inv:   self.employee->exists( age <= 65 )
```

Collection Operations: Iterate

- An **iterate** operation iterates over objects in a collection and accumulates a value of in a certain return type



```
context Company.ageSumOfAllEmployees:Integer
body:  self.employee->iterate(    p:Person ;
                                  sum:Integer = 0 |
                                        sum + p.age)
```

defines the value of the derived attribute

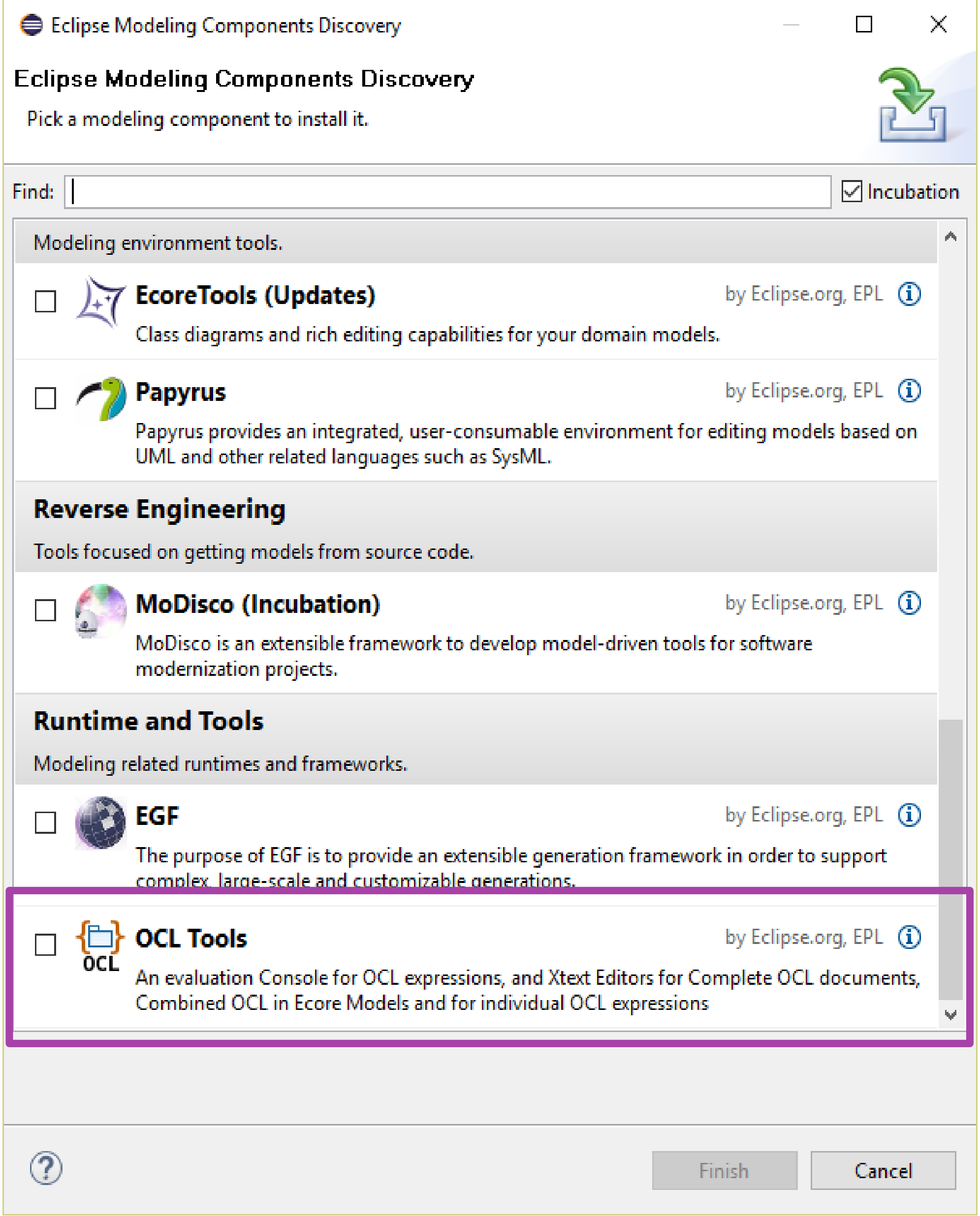
Further OCL Operations

- **self.oclIsTypeOf**`(t:OclType):Boolean`
  - returns true if the type of self and t are the same
- **self.oclIsKindOf**`(t:OclType):Boolean`
  - returns true if the type of self and t are the same or if t is a supertype of the type of self.
- **self.oclAsType**`(t:OclType):T`
  - "cast" operator, returns self as an object of type T.
- **allInstances()**
  - Operation on classes, interfaces, or enumerations
  - returns all instances of the type

3.3. OCL in Ecore

Eclipse OCL Tools

- Installation:



Eclipse Modeling Components Discovery

Eclipse Modeling Components Discovery

Pick a modeling component to install it.

Find: | ☑ Incubation

Modeling environment tools.

☐ **EcoreTools (Updates)** by Eclipse.org, EPL ⓘ
Class diagrams and rich editing capabilities for your domain models.

☐ **Papyrus** by Eclipse.org, EPL ⓘ
Papyrus provides an integrated, user-consumable environment for editing models based on UML and other related languages such as SysML.

Reverse Engineering

Tools focused on getting models from source code.

☐ **MoDisco (Incubation)** by Eclipse.org, EPL ⓘ
MoDisco is an extensible framework to develop model-driven tools for software modernization projects.

Runtime and Tools

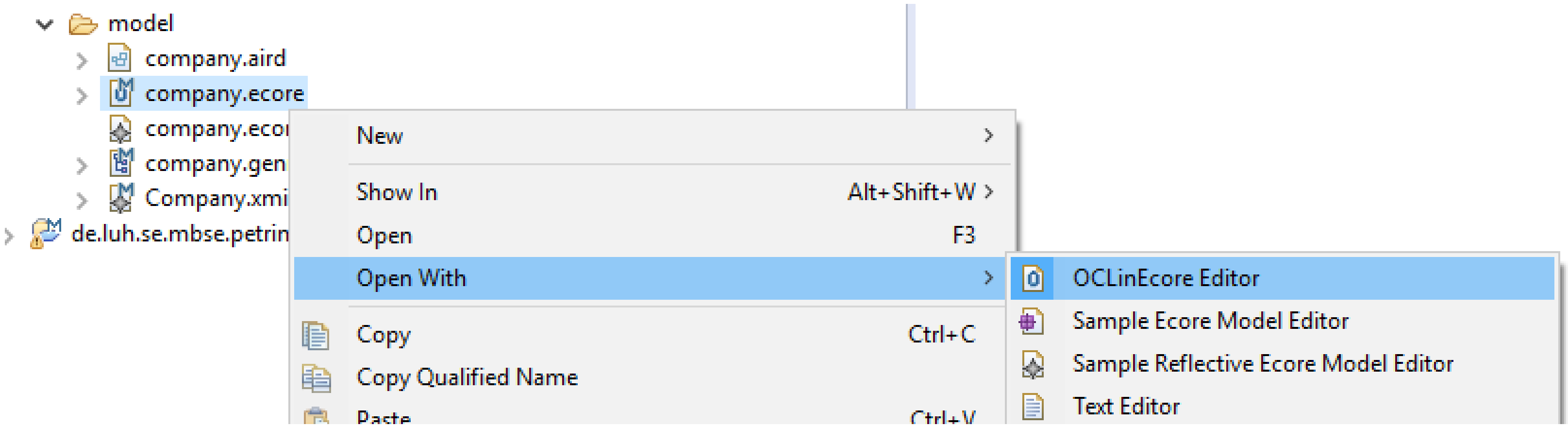
Modeling related runtimes and frameworks.

☐ **EGF** by Eclipse.org, EPL ⓘ
The purpose of EGF is to provide an extensible generation framework in order to support complex, large-scale and customizable generations.

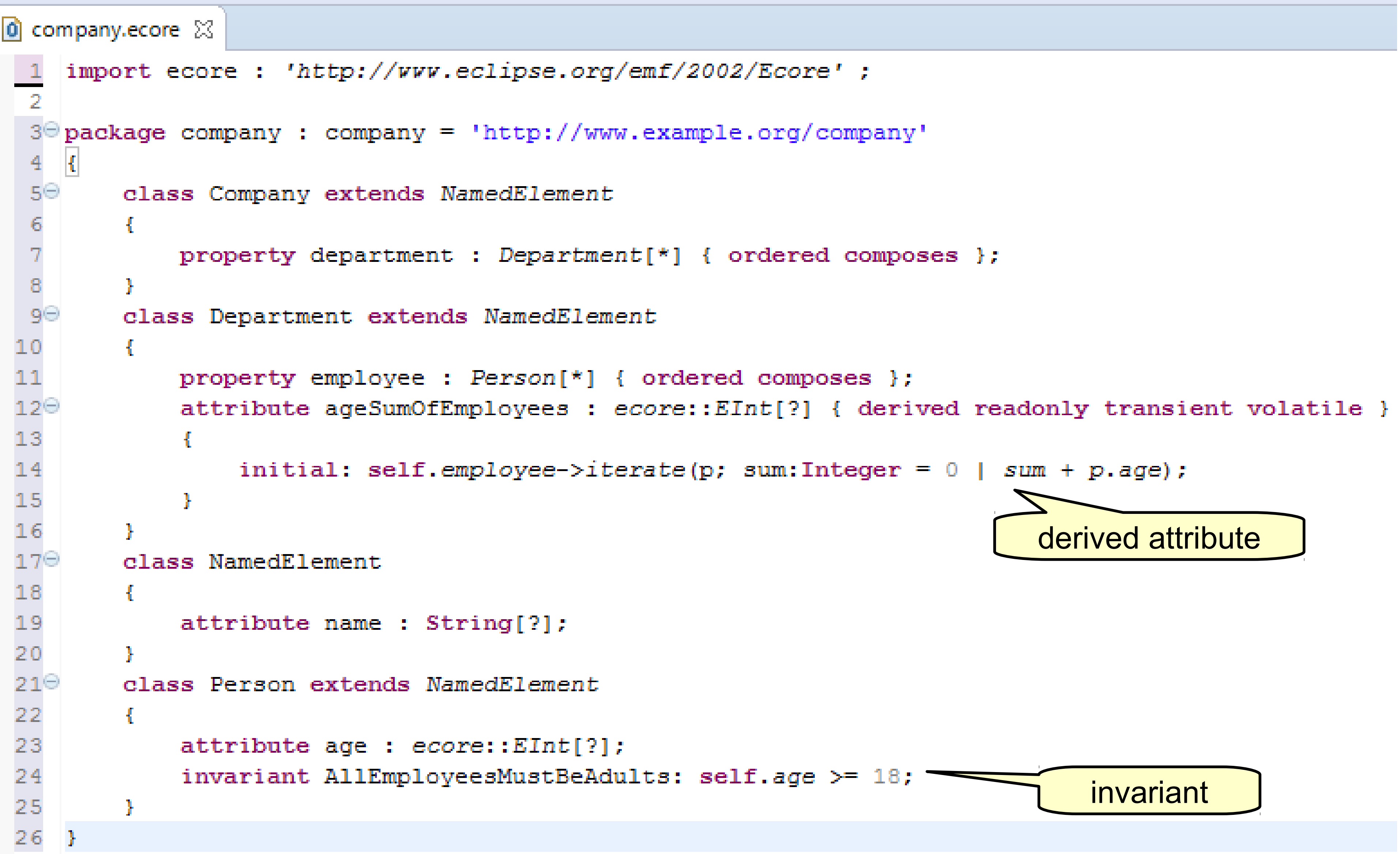
☐ **OCL Tools** by Eclipse.org, EPL ⓘ
An evaluation Console for OCL expressions, and Xtext Editors for Complete OCL documents, Combined OCL in Ecore Models and for individual OCL expressions

OCLinEcore Editor

- You can open .ecore files with the OCLinEcore editor



OCLinEcore Editor Example: Company



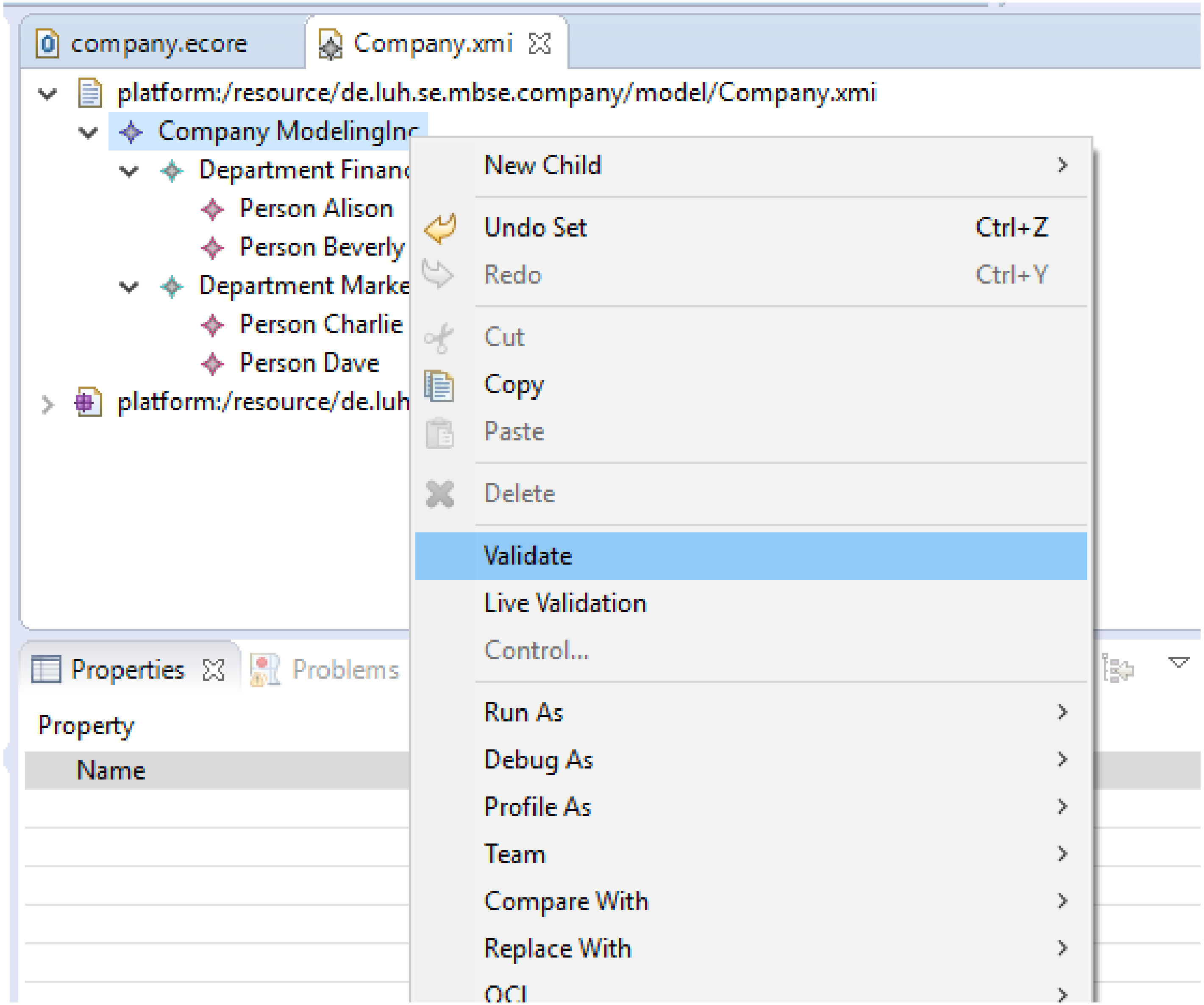
company.ecore

```
import ecore : 'http://www.eclipse.org/emf/2002/Ecore' ;

package company : company = 'http://www.example.org/company'
{
    class Company extends NamedElement
    {
        property department : Department[*] { ordered composes };
    }
    class Department extends NamedElement
    {
        property employee : Person[*] { ordered composes };
        attribute ageSumOfEmployees : ecore::EInt[?] { derived readonly transient volatile }
        {
            initial: self.employee->iterate(p; sum:Integer = 0 | sum + p.age);
        }
    }
    class NamedElement
    {
        attribute name : String[?];
    }
    class Person extends NamedElement
    {
        attribute age : ecore::EInt[?];
        invariant AllEmployeesMustBeAdults: self.age >= 18;
    }
}
```

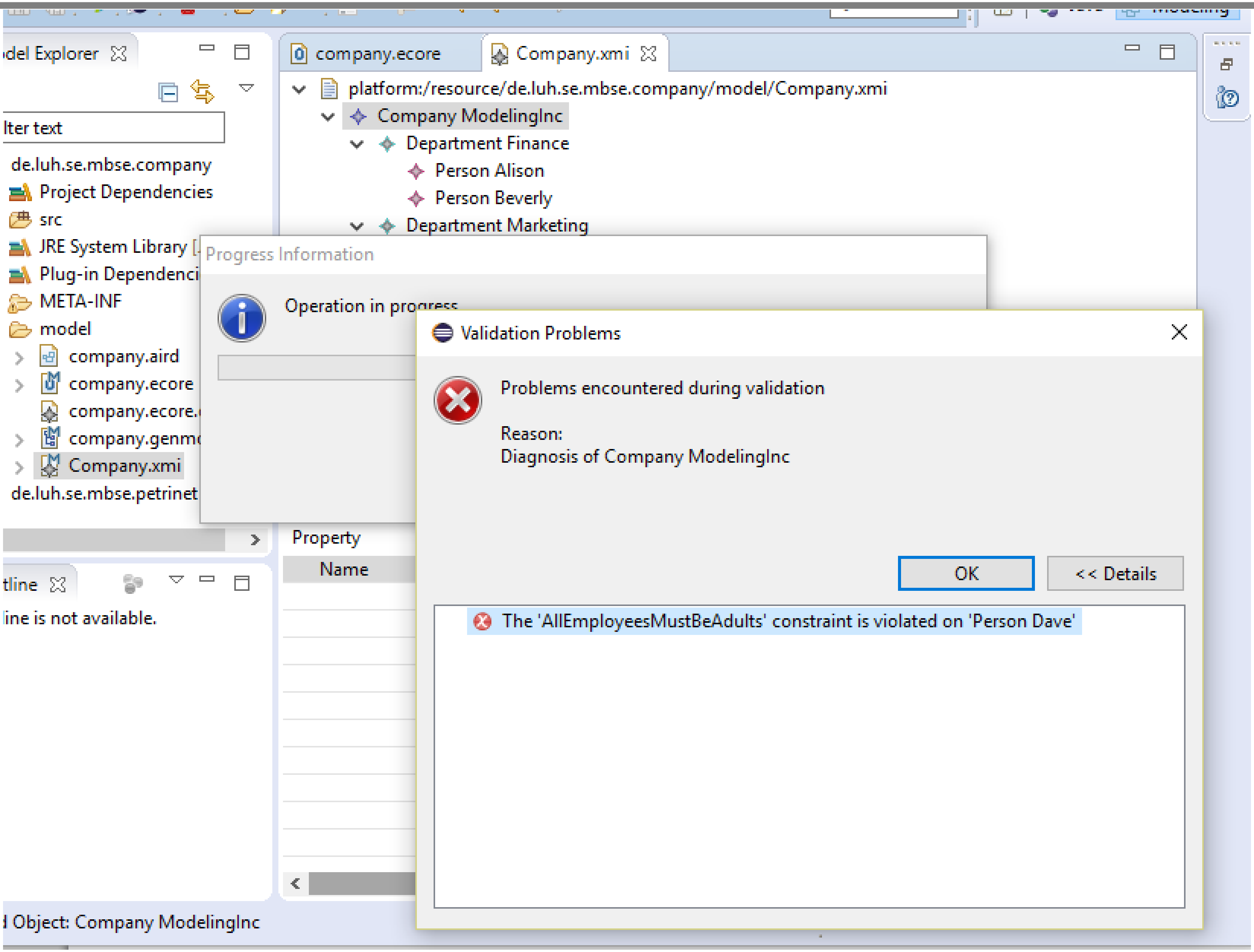
derived attribute

invariant

OCLinEcore Editor
Example: Company



OCLinEcore Editor
Example: Company

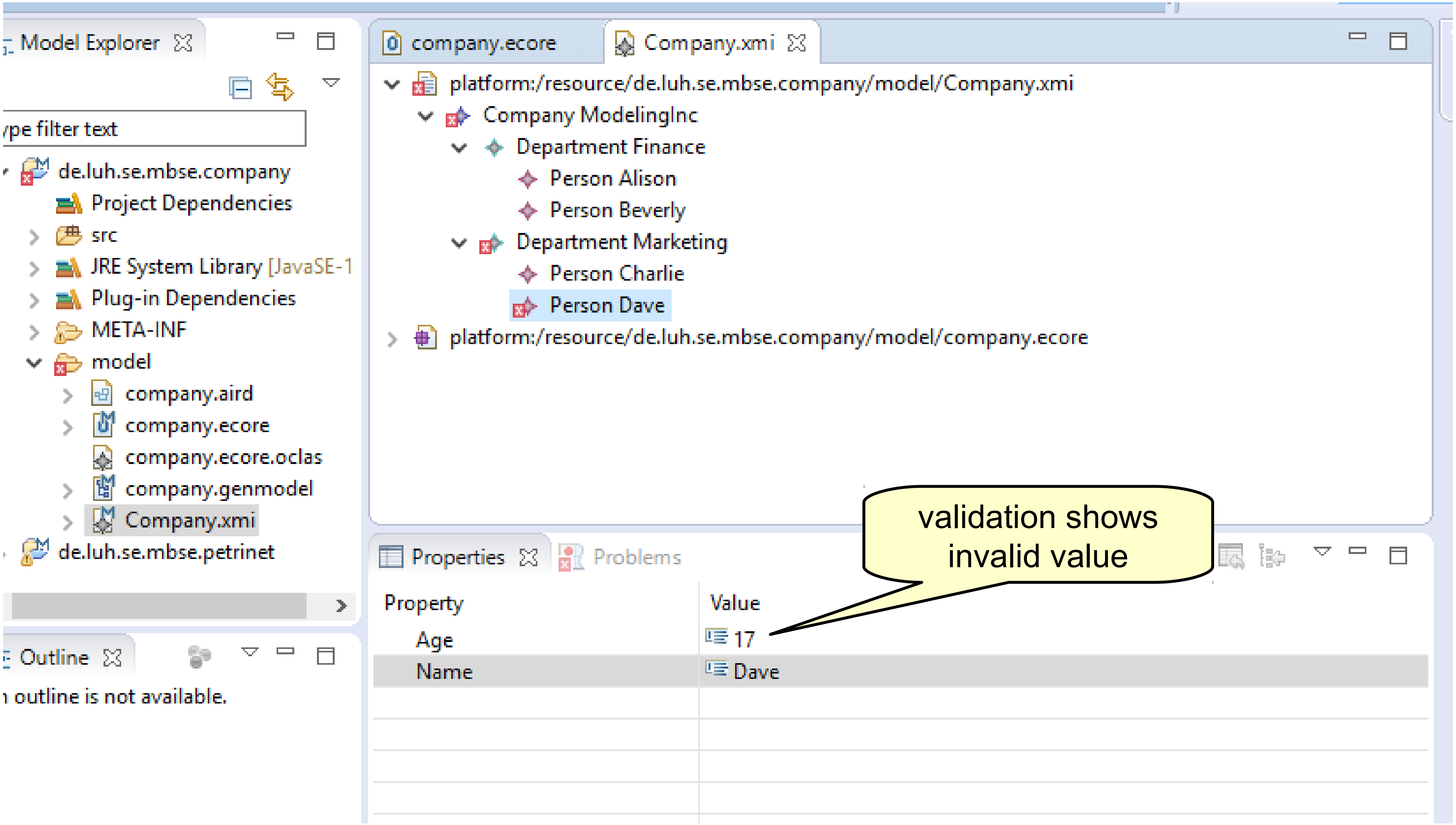


Problems encountered during validation

Reason:
Diagnosis of Company ModelingInc

The 'AllEmployeesMustBeAdults' constraint is violated on 'Person Dave'

OCLinEcore Editor
Example: Company

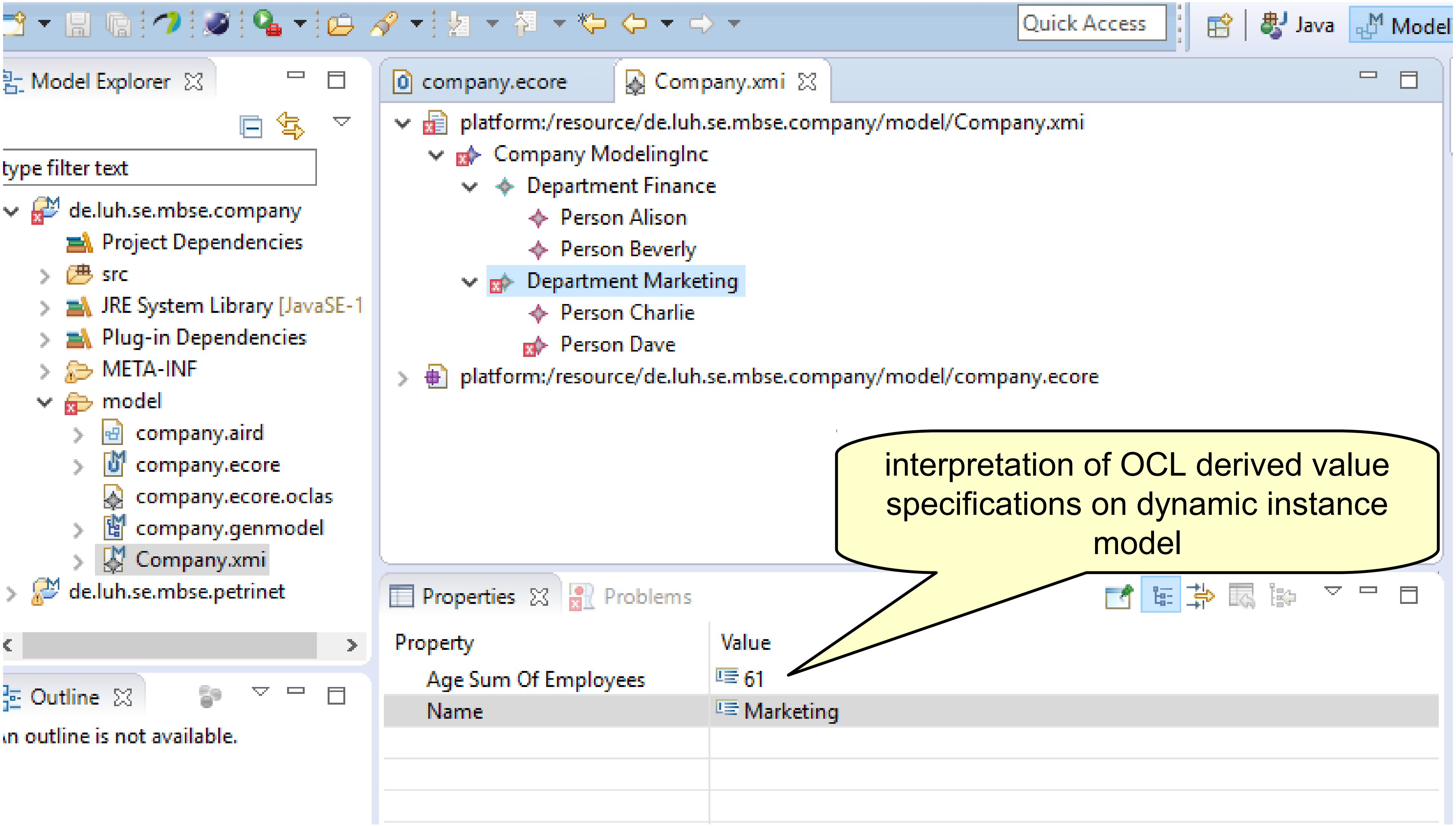


- platform:/resource/de.luh.se.mbse.company/model/Company.xmi
  - Company ModelingInc
    - Department Finance
      - Person Alison
      - Person Beverly
    - Department Marketing
      - Person Charlie
      - Person Dave
- platform:/resource/de.luh.se.mbse.company/model/company.ecore

Property	Value
Age	17
Name	Dave

validation shows invalid value

OCLinEcore Editor
Example: Company



- de.luh.se.mbse.company
  - Project Dependencies
  - src
  - JRE System Library [JavaSE-1
  - Plug-in Dependencies
  - META-INF
  - model
    - company.aird
    - company.ecore
    - company.ecore.oclas
    - company.genmodel
    - Company.xmi
- de.luh.se.mbse.petrinet

company.ecore | Company.xmi

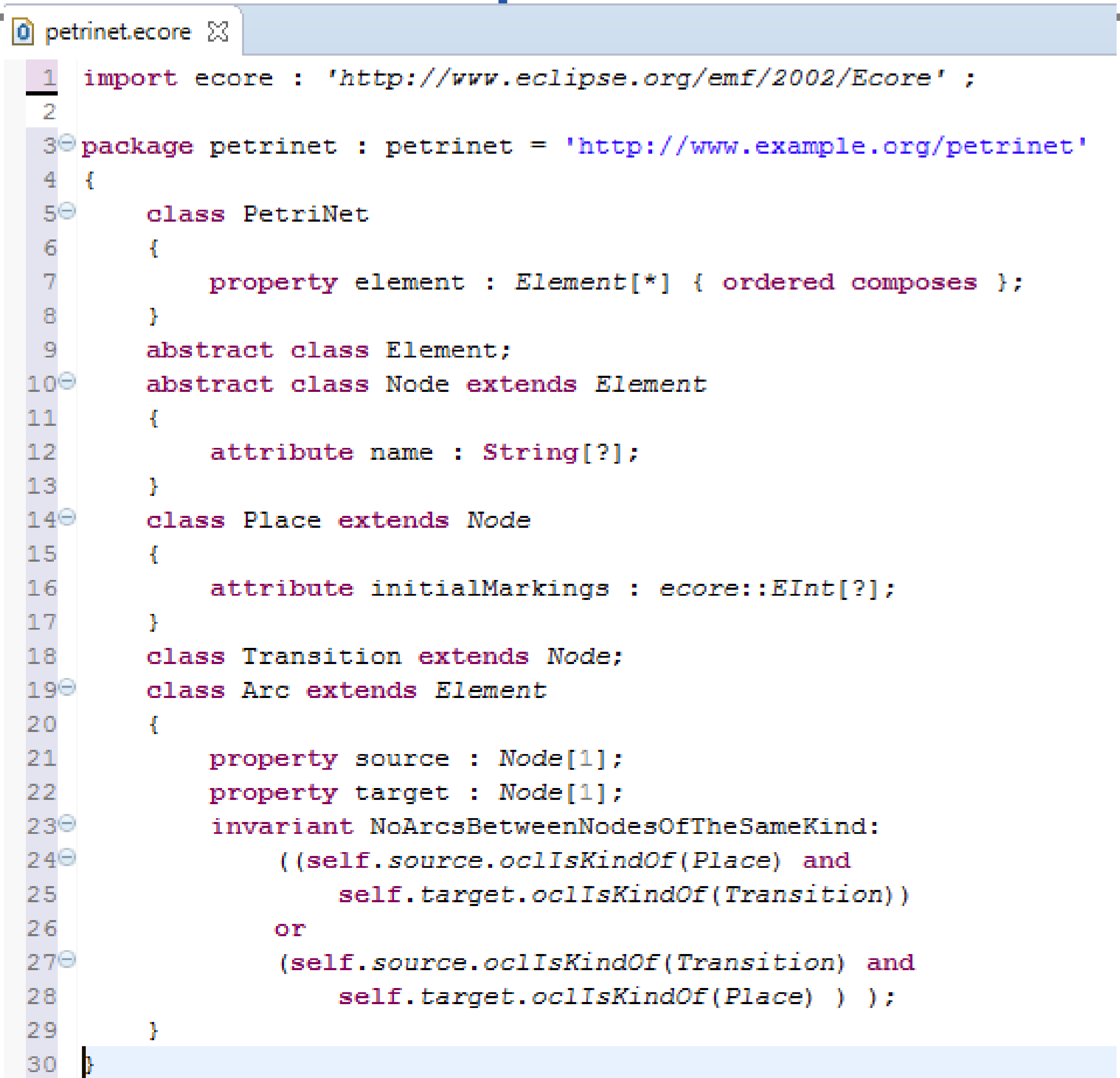
- platform:/resource/de.luh.se.mbse.company/model/Company.xmi
  - Company ModelingInc
    - Department Finance
      - Person Alison
      - Person Beverly
    - Department Marketing
      - Person Charlie
      - Person Dave
- platform:/resource/de.luh.se.mbse.company/model/company.ecore

Property	Value
Age Sum Of Employees	61
Name	Marketing

ın outline is not available.

interpretation of OCL derived value specifications on dynamic instance model

OCLinEcore Editor
Example: Petri net



```
import ecore : 'http://www.eclipse.org/emf/2002/Ecore' ;

package petrinet : petrinet = 'http://www.example.org/petrinet'
{
    class PetriNet
    {
        property element : Element[*] { ordered composes };
    }
    abstract class Element;
    abstract class Node extends Element
    {
        attribute name : String[?];
    }
    class Place extends Node
    {
        attribute initialMarkings : ecore::EInt[?];
    }
    class Transition extends Node;
    class Arc extends Element
    {
        property source : Node[1];
        property target : Node[1];
        invariant NoArcsBetweenNodesOfTheSameKind:
            ((self.source.oclIsKindOf(Place) and
                self.target.oclIsKindOf(Transition))
            or
            (self.source.oclIsKindOf(Transition) and
                self.target.oclIsKindOf(Place) ) );
    }
}
```

Summary OCL

- Formal, textual language for specifying queries and constraints on models with a MOF/UML metamodel
- Typed language
- No “programming”, no side-effects
- Tool support for EMF
- Used in other languages
  - we will see it again!