

Model-Based Software Engineering

Lecture 03 – Metamodeling cont., OCL

Prof. Dr. Joel Greenyer

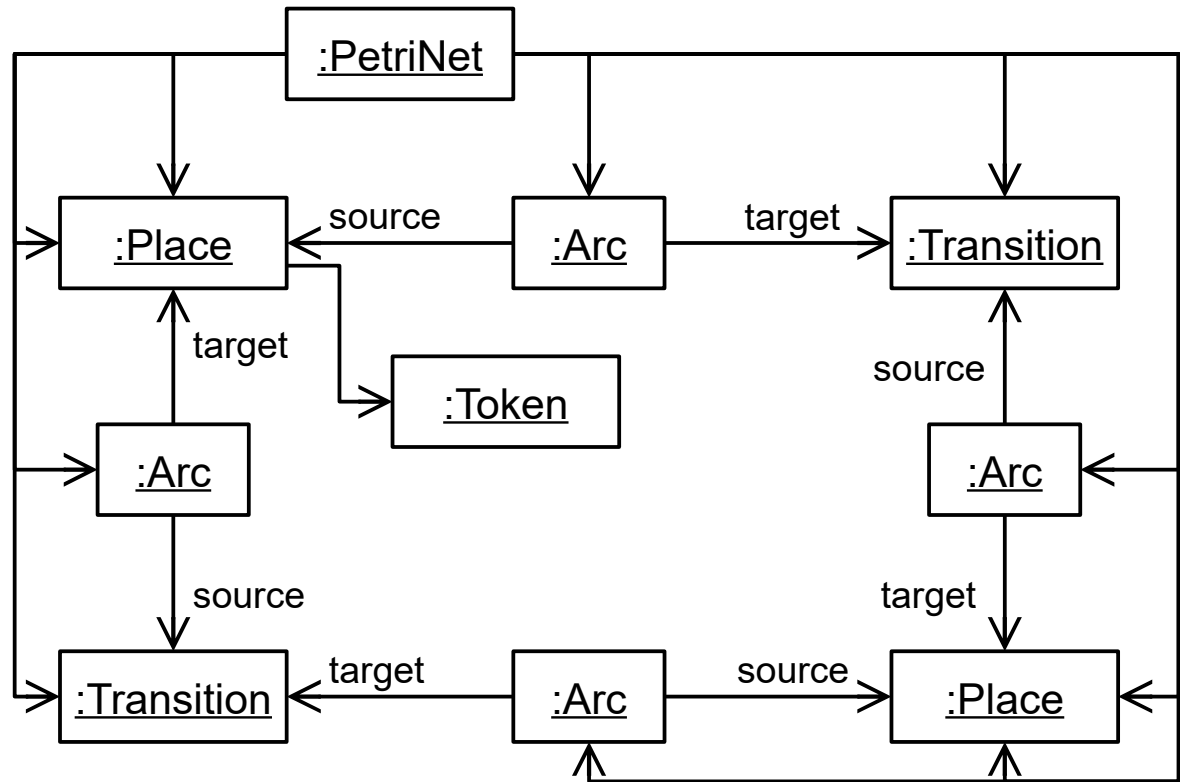
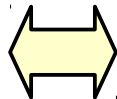
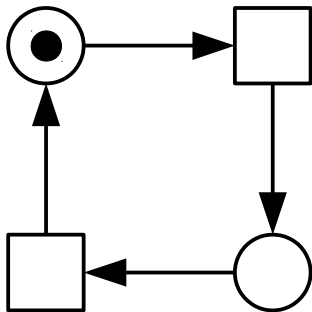


April 18, 2016



in the last lecture...

- **Step 1:** Understand a model as a **structure of objects**
- For the example:



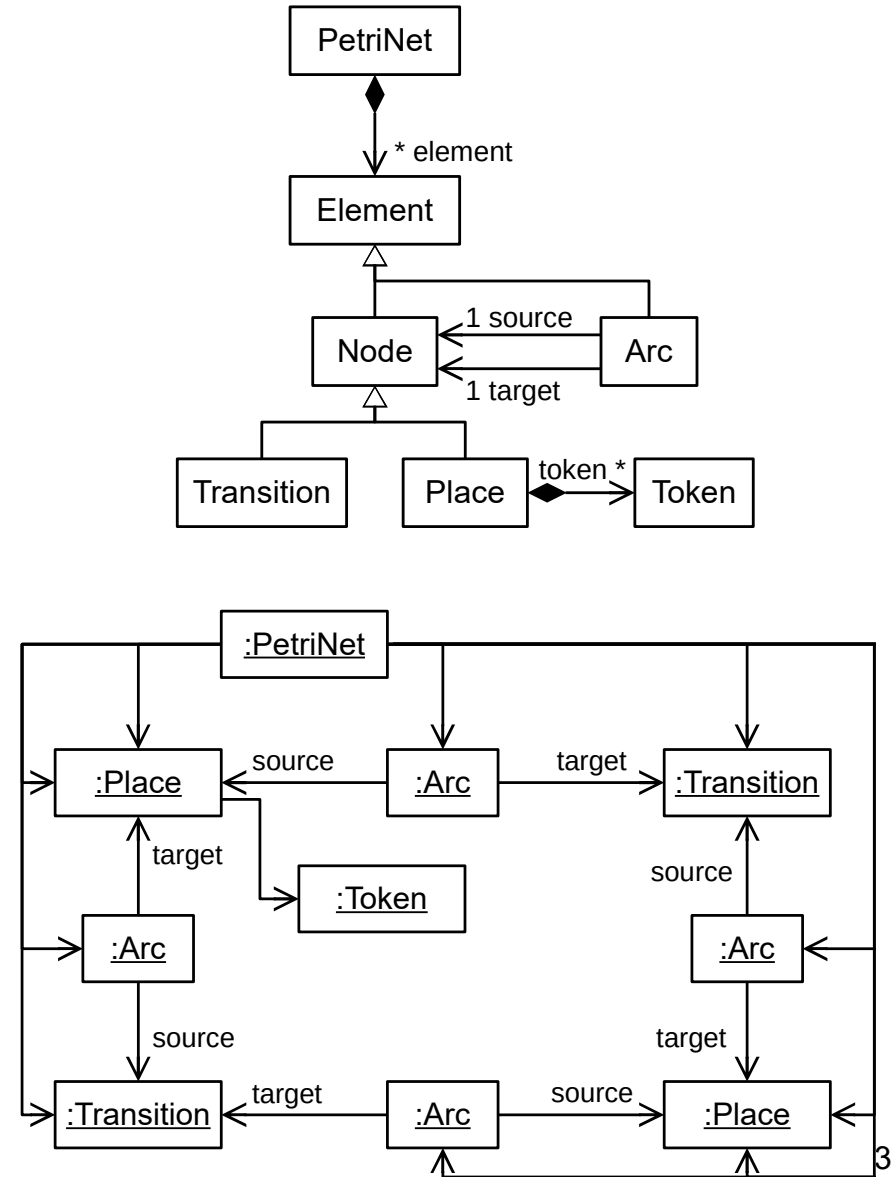
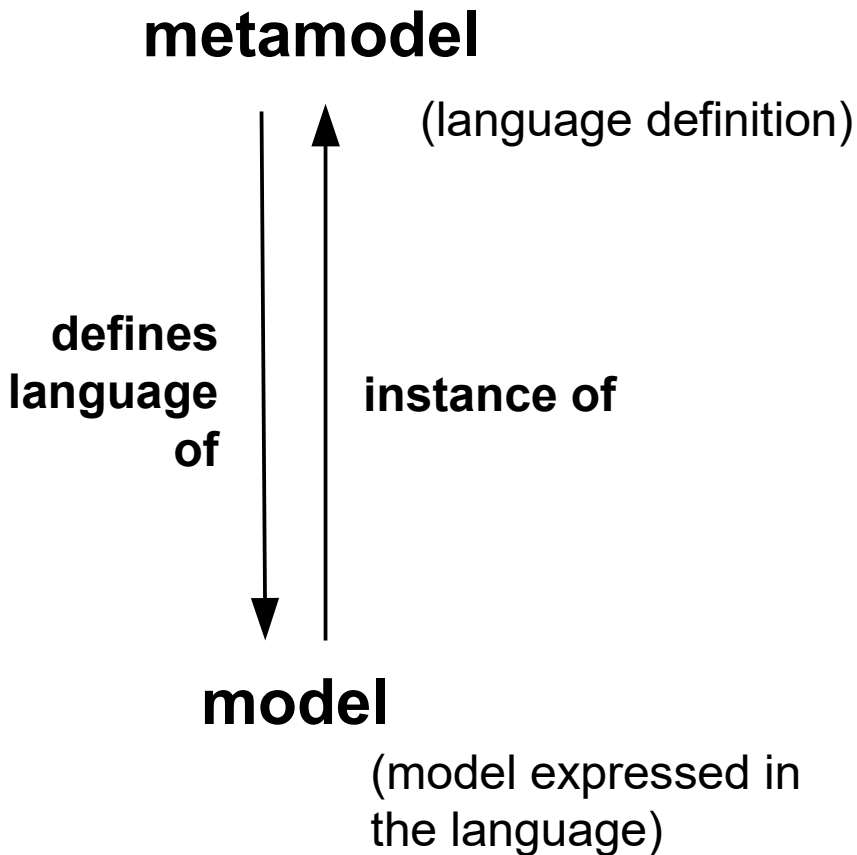
concrete syntax

(representation to the user)

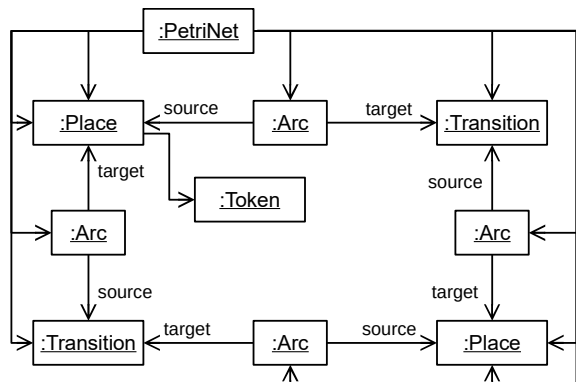
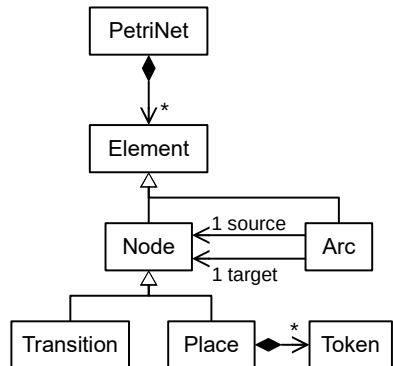
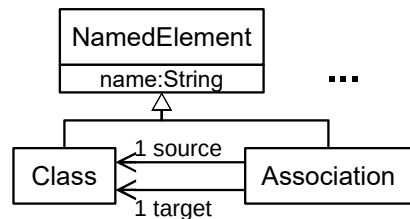
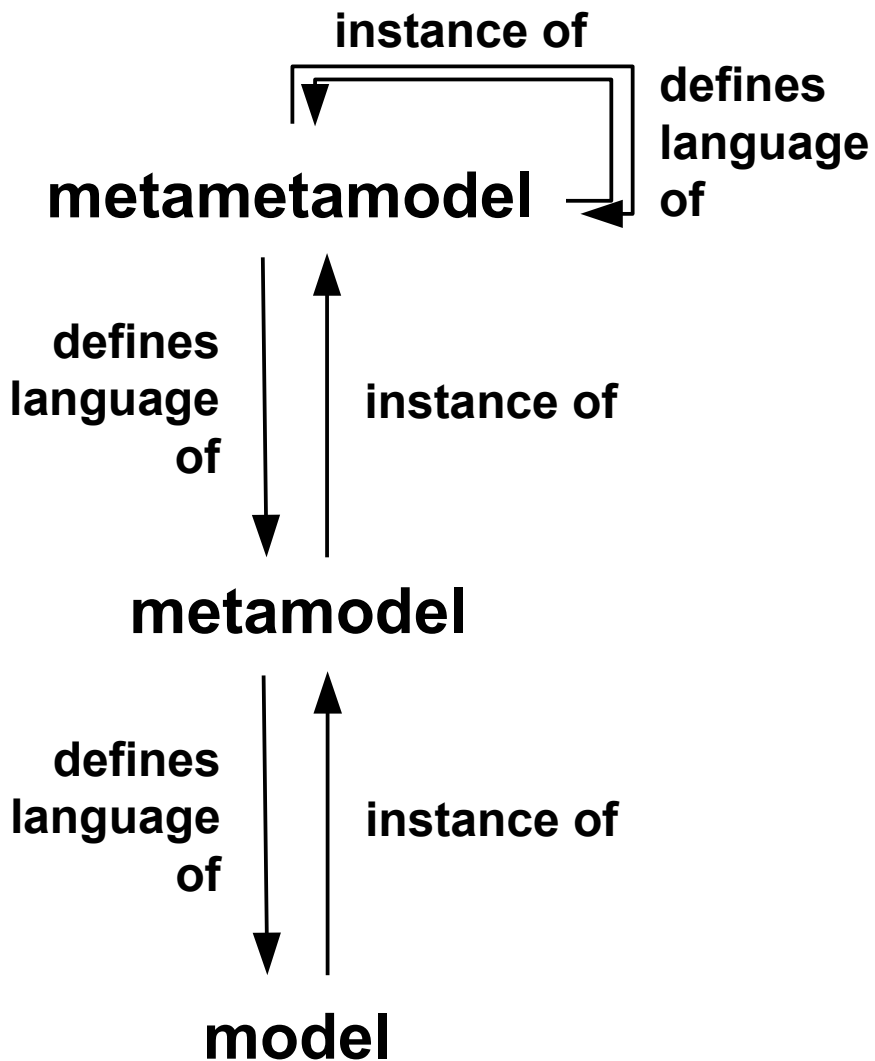
abstract syntax

(internal structure, occurrences of language constructs and their relationships)

in the last lecture...



in the last lecture...



Typical Meta-Level Descriptions

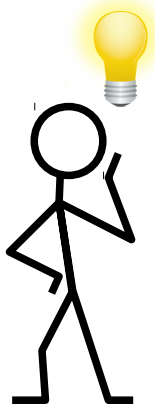
in the last lecture...

- Sometimes, we refer to the **four meta-levels** (M0-M3) originally defined by the MOF standard
 - MOF: Meta-Object Efacility, standard by the OMG (see <http://www.omg.org/mof/>)

M3	meta-metamodel to define metamodels on M2, also describes itself
M2	metamodels , for defining a modeling language on M1
M1	models of data or processes
M0	instance-model , concrete data

Vision: Build a Petri Net Modeling Tool

in the last lecture...



Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse

File Edit Diagram Navigate Search Project Run Window Help

Quick Access

Model Explorer

type filter text

- de.luh.se.mbse.myfirstpetrinetdiag
 - Project Dependencies
 - MyPetrinet.xml
 - Petrinet
 - new petrinet diagram
 - Place idle
 - Transition start
 - Place working
 - Transition stop
 - representations.aird
 - de.luh.se.mbse.myfirstpetrinetproj
 - de.luh.se.mbse.petrinet

new petrinet diagram

Palette

Node Creation...

- Place
- Transition
- Arc

Properties

Problems

Place idle

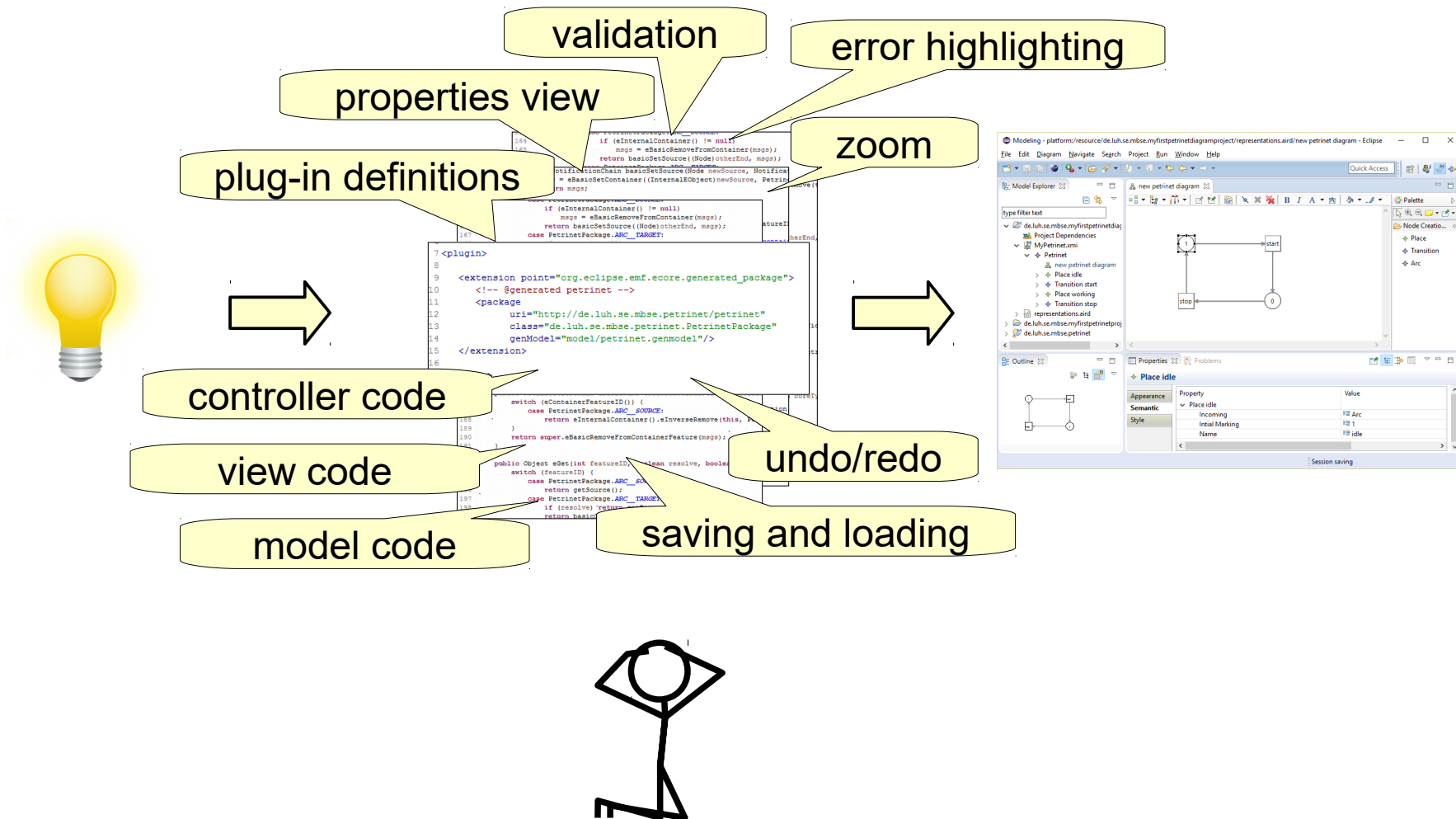
Property	Value
Place idle	
Incoming	Arc
Initial Marking	1
Name	idle

Session saving

Build a Petri Net Modeling Tool

in the last lecture...

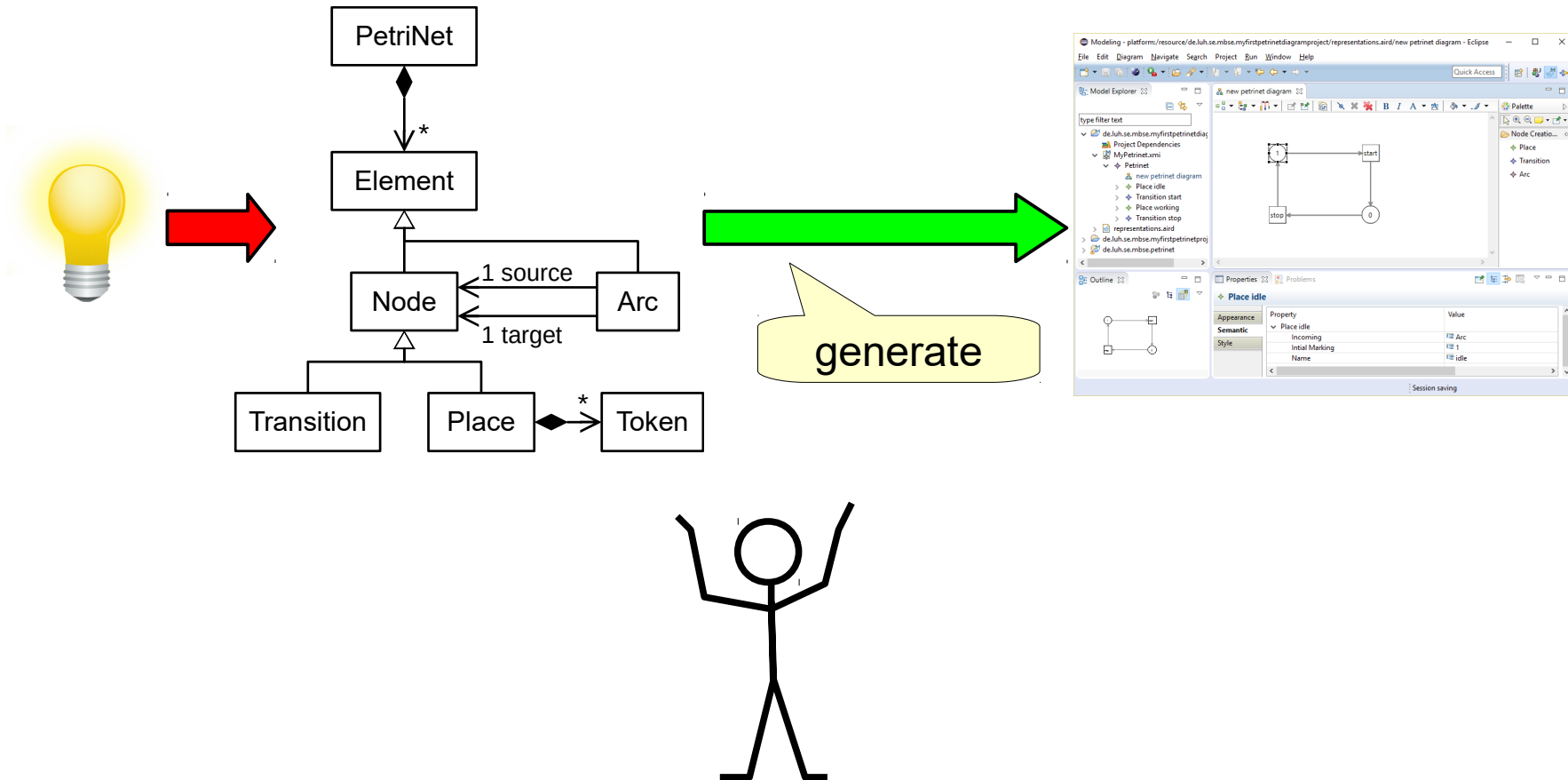
- Manual implementation: A lot of repetitive or generic code



Build a Petri Net Modeling Tool

in the last lecture...

- Model-based approach for building modeling tools: Provide only a few conceptual models and generate tool automatically



Eclipse Modeling Framework (Modeling a Petri Net Metamodel)

in the last tutorial...

Modeling - platform:/resource/de.luh.se.mbse.petrinet/model/petrinet.aird/petrinet - Eclipse

File Edit Diagram Navigate Search Project Run Window Help

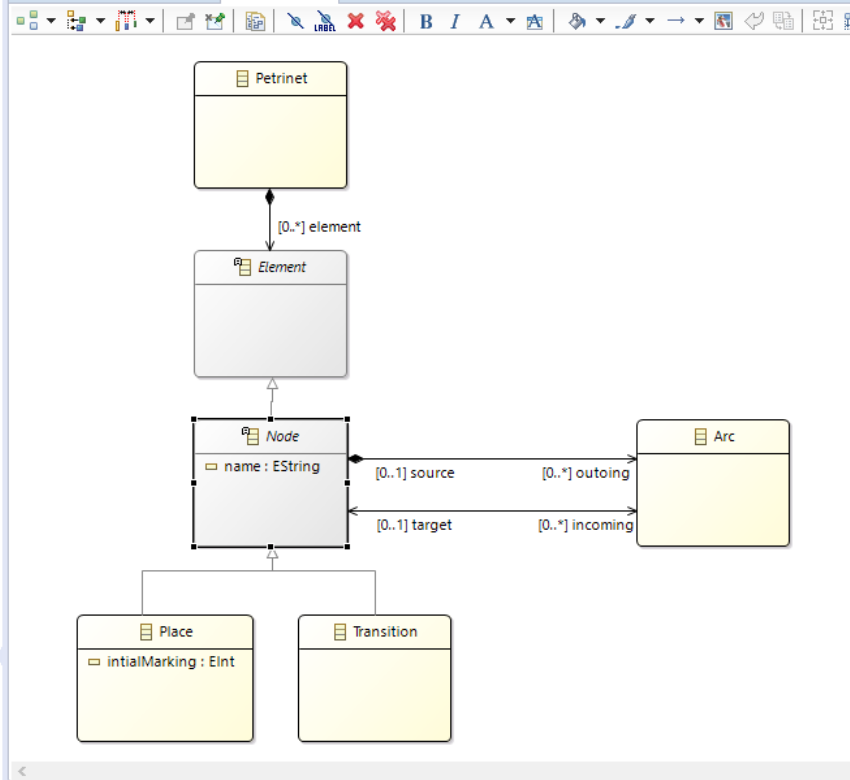
Model Explorer

type filter text

- de.luh.se.mbse.myfirstpetrinetdiagramproject
 - Project Dependencies
 - MyPetrinet.xmi
 - representations.aird
- de.luh.se.mbse.myfirstpetrinetproject
 - Project Dependencies
- de.luh.se.mbse.petrinet
 - Project Dependencies
 - src
 - JRE System Library [jre1.8.0_73]
 - Plug-in Dependencies
 - META-INF
 - model
 - petrinet.aird
 - Design
 - Entities
 - petrinet
 - petrinet.ecore
 - petrinet.genmodel
 - Petrinet
 - petrinet.odesign
 - build.properties
 - plugin.properties
 - plugin.xml

new petrinet diagram

petrinet



Palette

- Existing Elem...
- Add
- Remove
- Classifier
 - Class
 - Datatype
 - Enumeration
 - ETypeParameter
- Feature
 - Literal
 - Operation
 - Attribute
- Relation
 - SuperType
 - Reference
 - Bi-directional Reference
 - Composition
- Dynamic
 - Package
 - Package

Properties Problems

Node -> Element

Model

Instantiation

Annotation

Extended Metadata

GenModel Doc

Properties

Name: Node

☒ Abstract ☐ Interface

Inheritance

tutorial last week

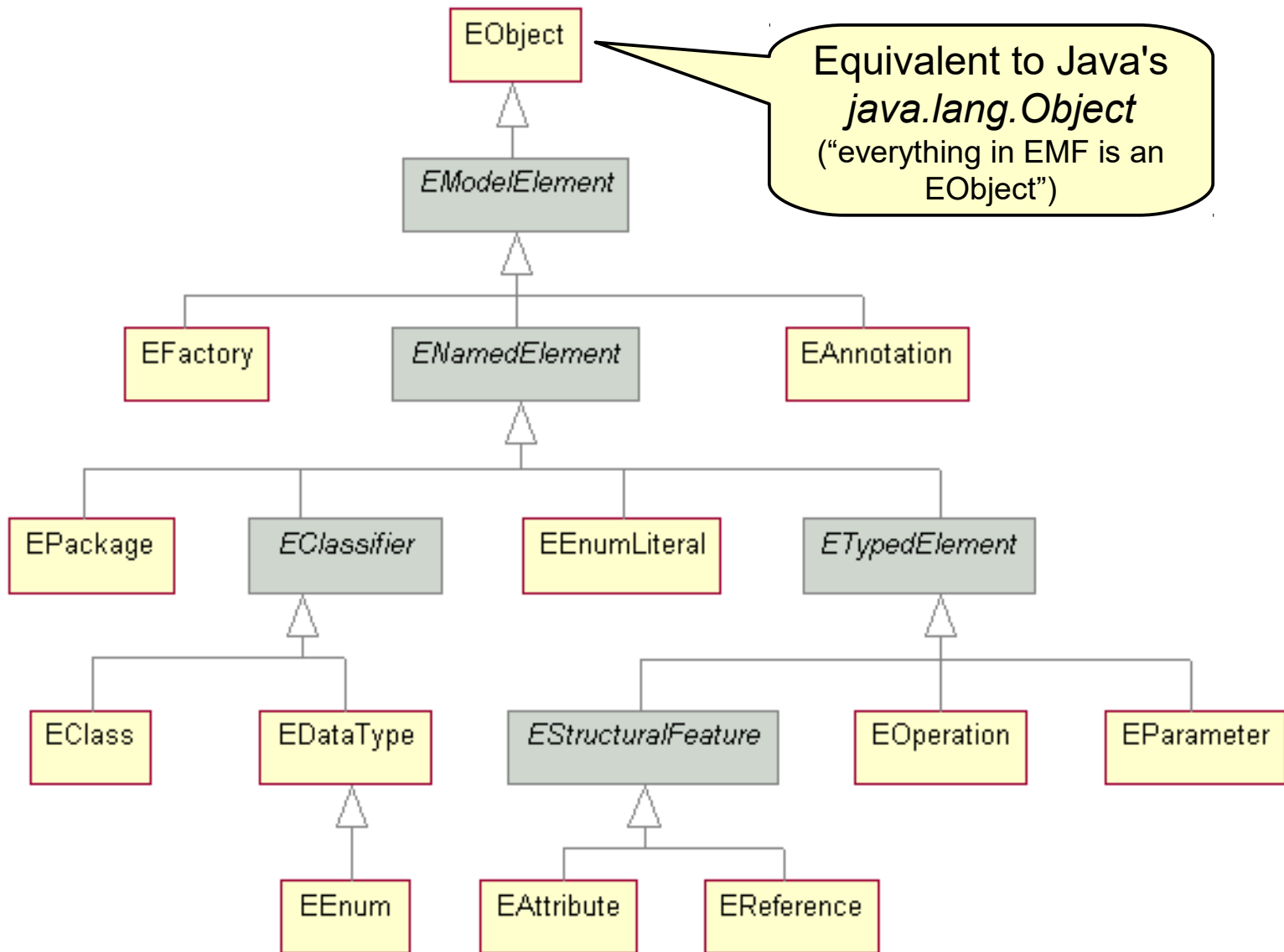
2.5. Eclipse Modeling Framework (EMF) and Ecore

Ecore vs EMOF

- MOF defines a meta-metamodel used to define metamodels in other OMG specifications (UML, IDL, CWM)
 - (see <http://www.omg.org/mof/>)
- MOF consists of two parts
 - “Essential MOF” (EMOF) is a core part of the MOF
 - the full MOF meta-metamodel is called “Complete MOF” (CMOF), which includes EMOF, i.e., is an extension of EMOF
 - EMOF and CMOF are *derived from the UML2 metamodel*
- Ecore is the meta-metamodel used by EMF
 - it is very similar to EMOF, some aspects are even simplified
 - it has some technology-specific specialties
 - we are going to look at Ecore mainly and later compare to MOF

A Close Look at the Ecore Meta-Model

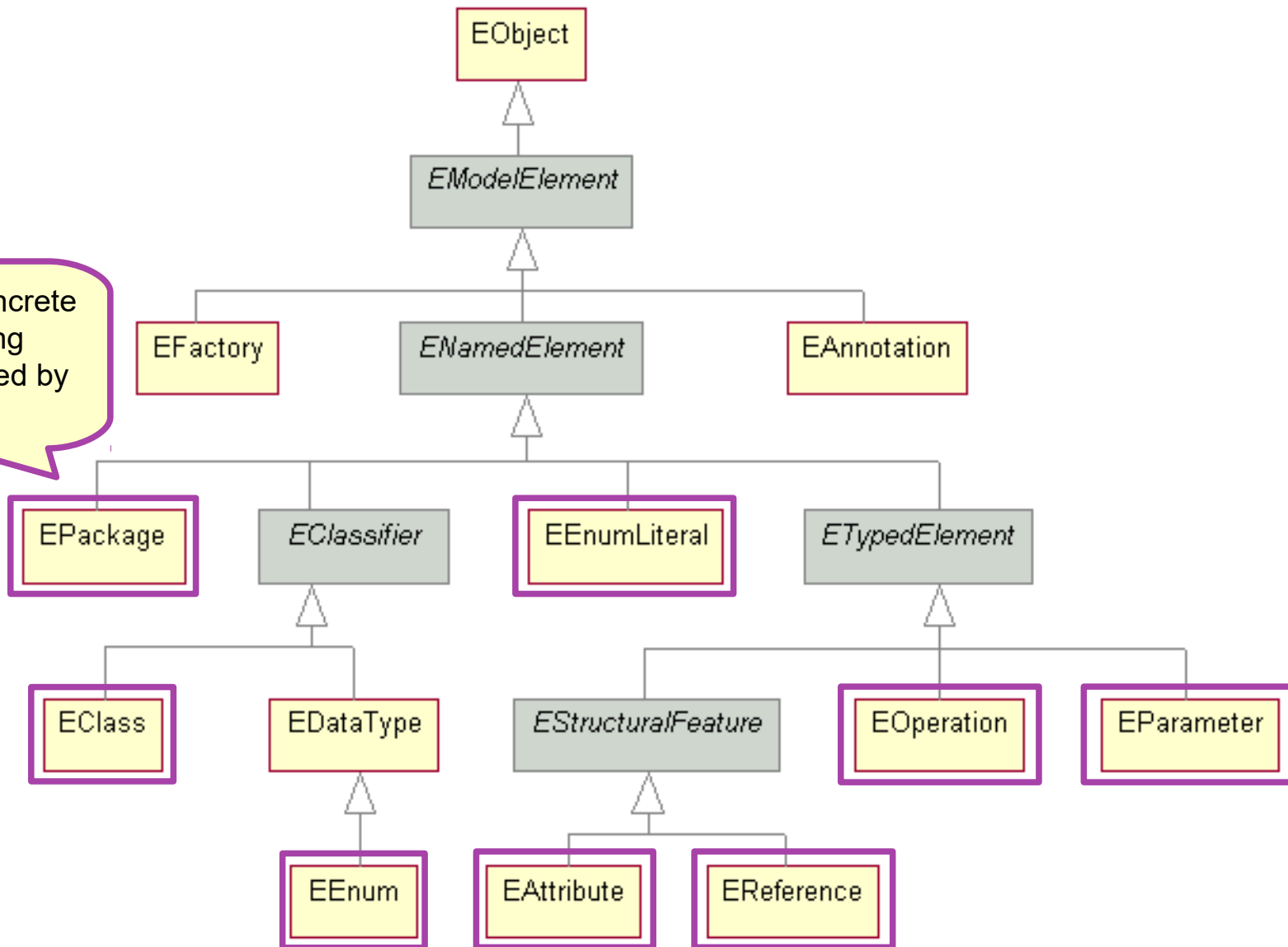
- ECore:



A Close Look at the Ecore Meta-Model

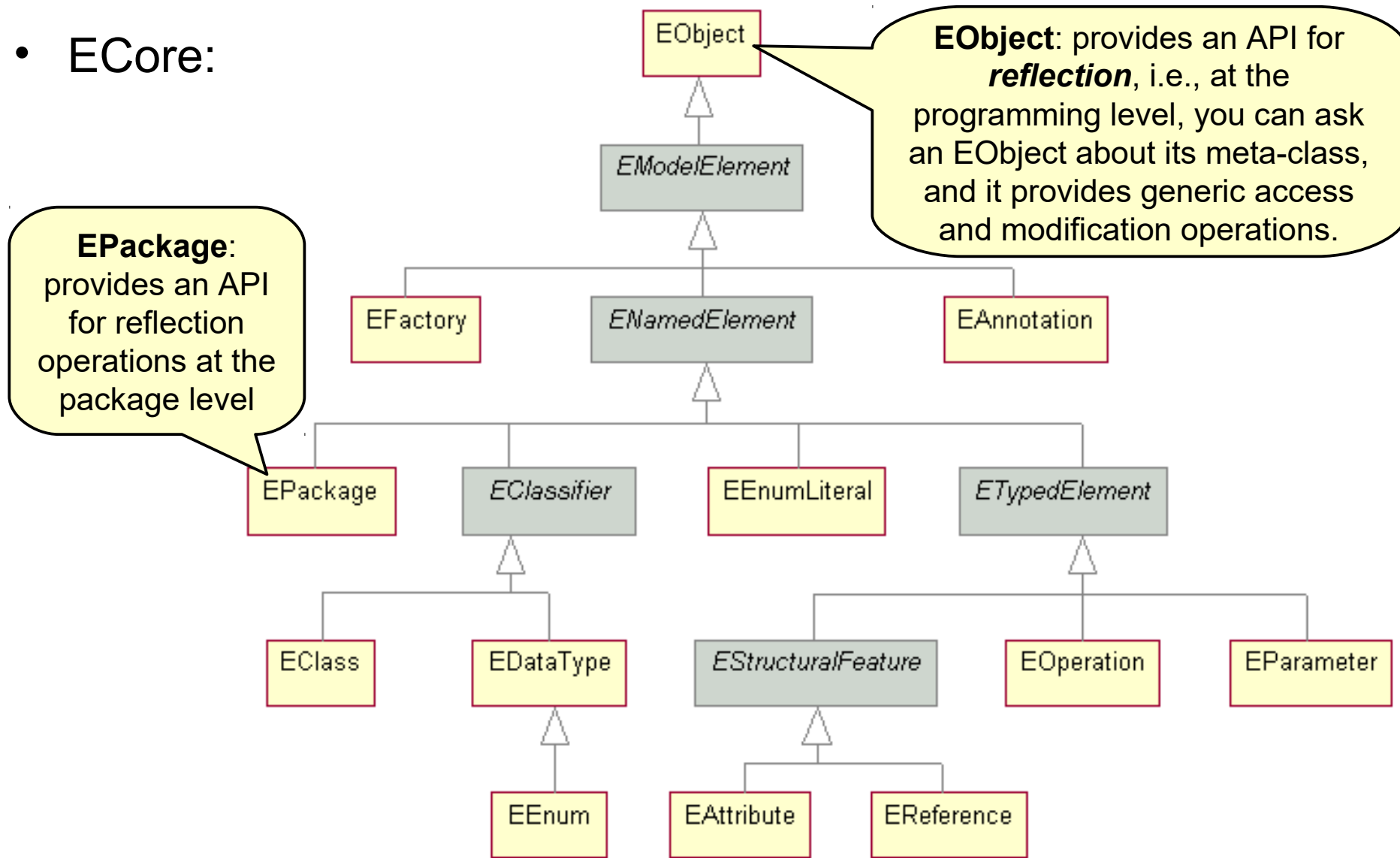
- ECore:

these are the concrete
metamodeling
concepts provided by
ECore



A Close Look at the Ecore Meta-Model

- ECore:



A Close Look at the Ecore Meta-Model

- The EObject API:

“what is my meta-class?”

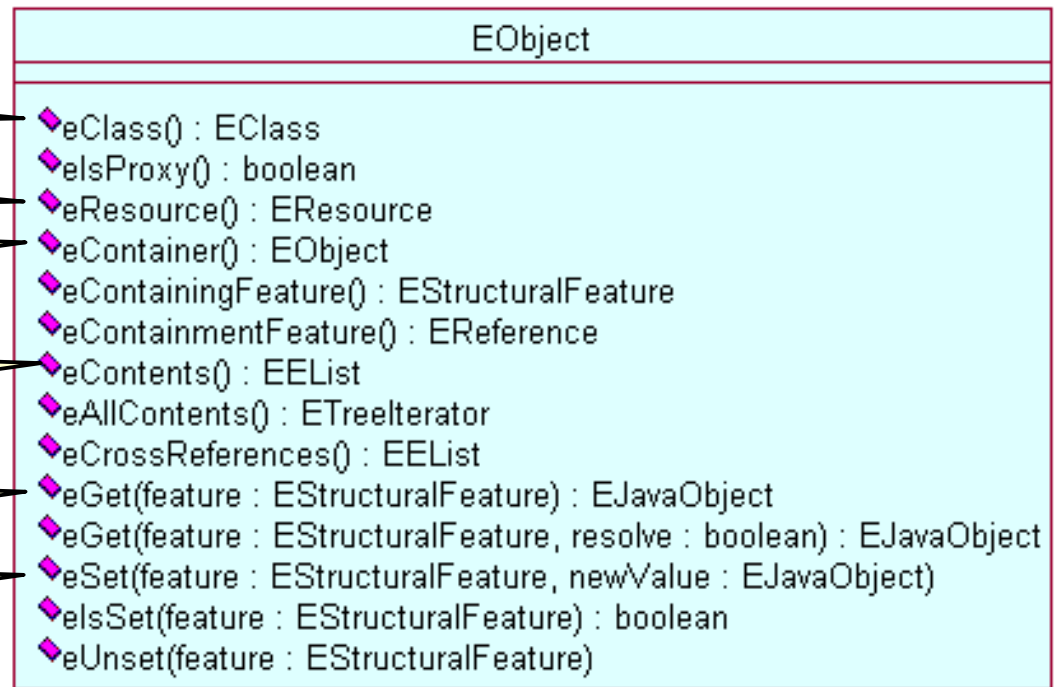
“where am I stored?”

“what is my containing EObject?”

“what are all the objects that I contain?”

generic get-method

generic set-method



A Close Look at the Ecore Meta-Model

- The EObject API:
- **Example:**
reflective getName() function:

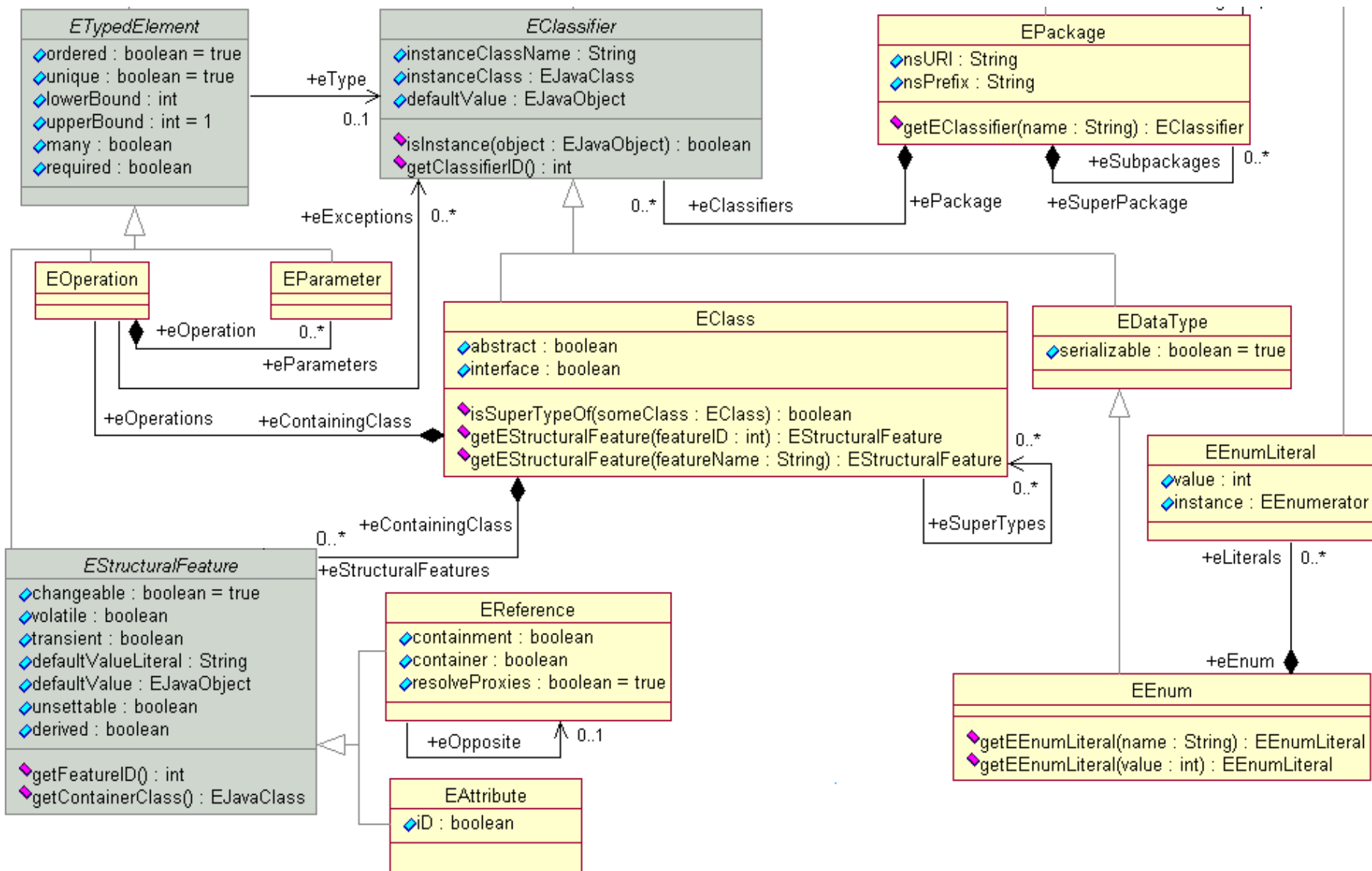
EObject
<ul style="list-style-type: none"> ◆ eClass() : EClass ◆ isProxy() : boolean ◆ eResource() : EResource ◆ eContainer() : EObject ◆ eContainingFeature() : EStructuralFeature ◆ eContainmentFeature() : EReference ◆ eContents() : EList ◆ eAllContents() : ETreeIterator ◆ eCrossReferences() : EList ◆ eGet(feature : EStructuralFeature) : EJavaObject ◆ eGet(feature : EStructuralFeature, resolve : boolean) : EJavaObject ◆ eSet(feature : EStructuralFeature, newValue : EJavaObject) ◆ isSet(feature : EStructuralFeature) : boolean ◆ eUnset(feature : EStructuralFeature)

```
public static String getName(EObject eObject) {
    EClass eClass = eObject.eClass();
    EStructuralFeature nameFeature
        = eClass.getEStructuralFeature("name");
    if (nameFeature != null
        && "EString".equals(nameFeature.getEType().getName())) {
        return (String) eObject.eGet(nameFeature);
    } else {
        return "";
    }
}
```





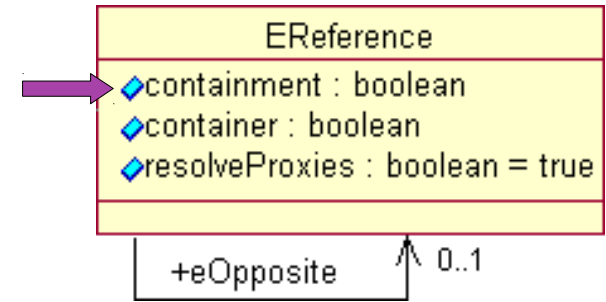

A Close Look at the Ecore Meta-Model



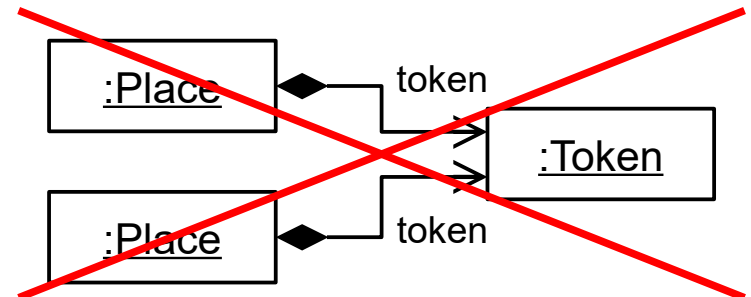
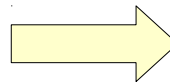
Specialties of EReferences

- **Containment:**

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



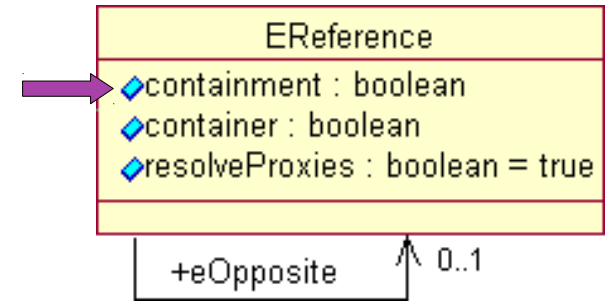
- **Example:**



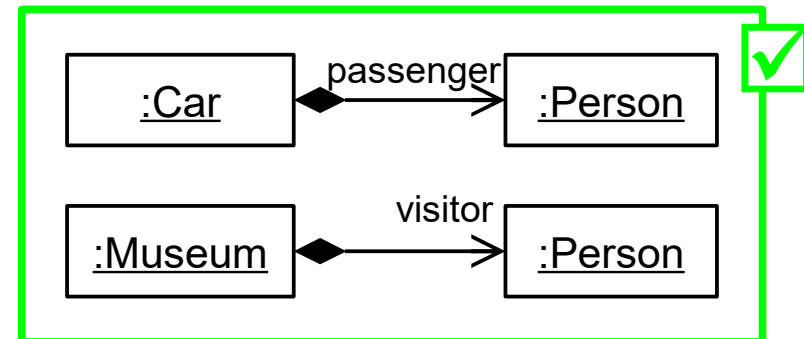
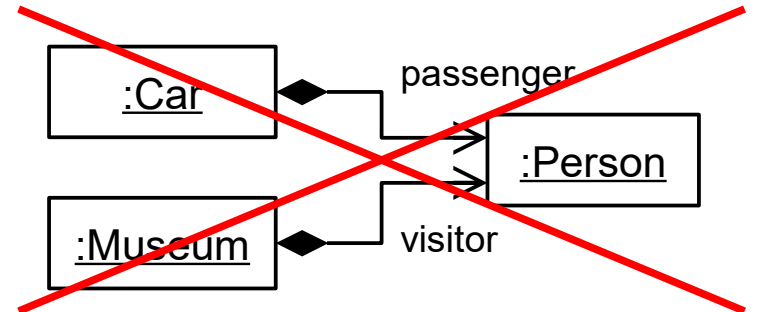
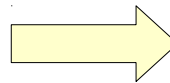
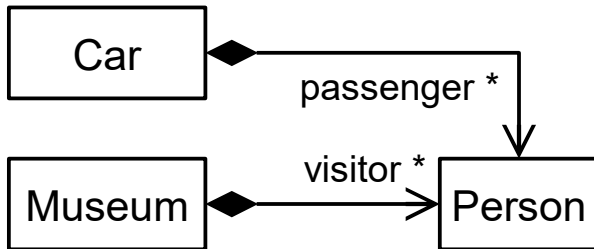
Specialties of EReferences

- **Containment:**

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

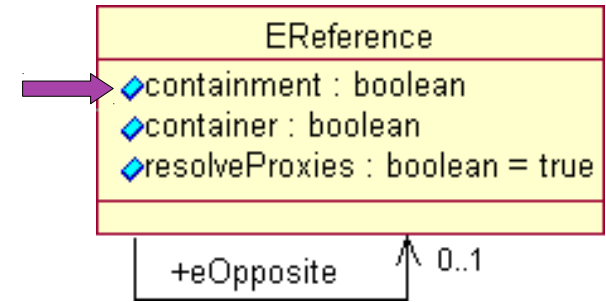


- **Example:**

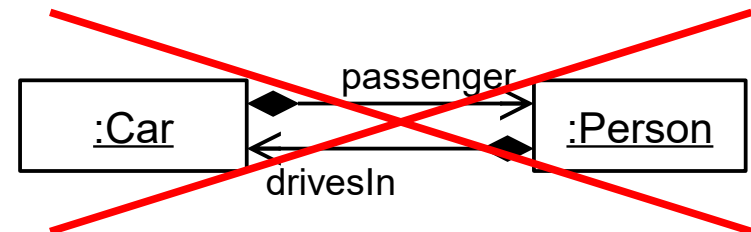
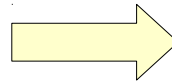
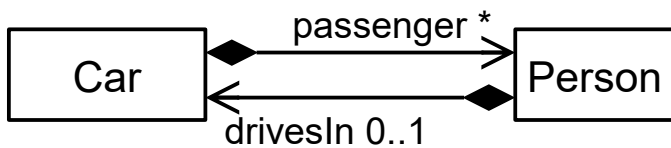


Specialties of EReferences

- **Containment:**
 - Containment links must not form a cycle
 - An object cannot contain any object that it is (transitively) contained in

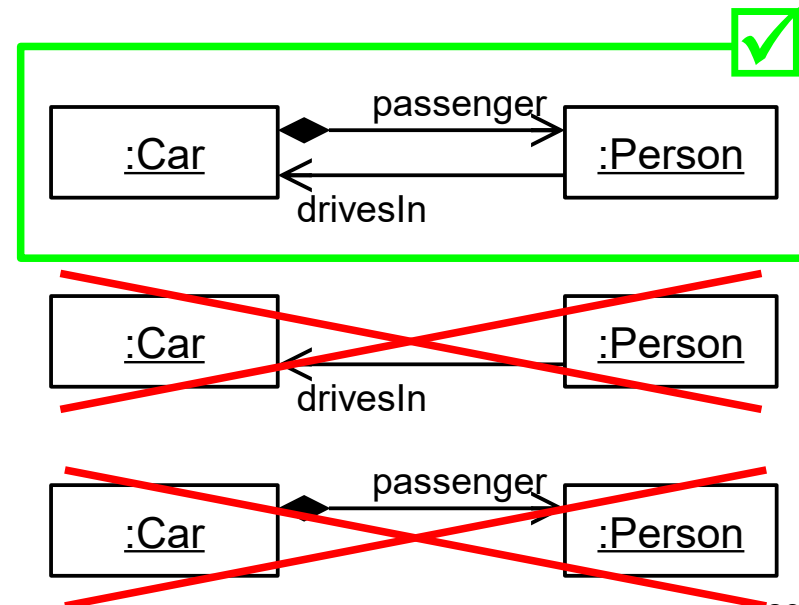
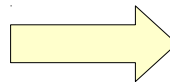
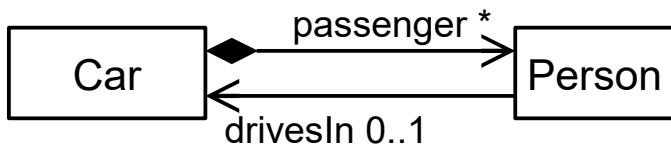


- **Example:**

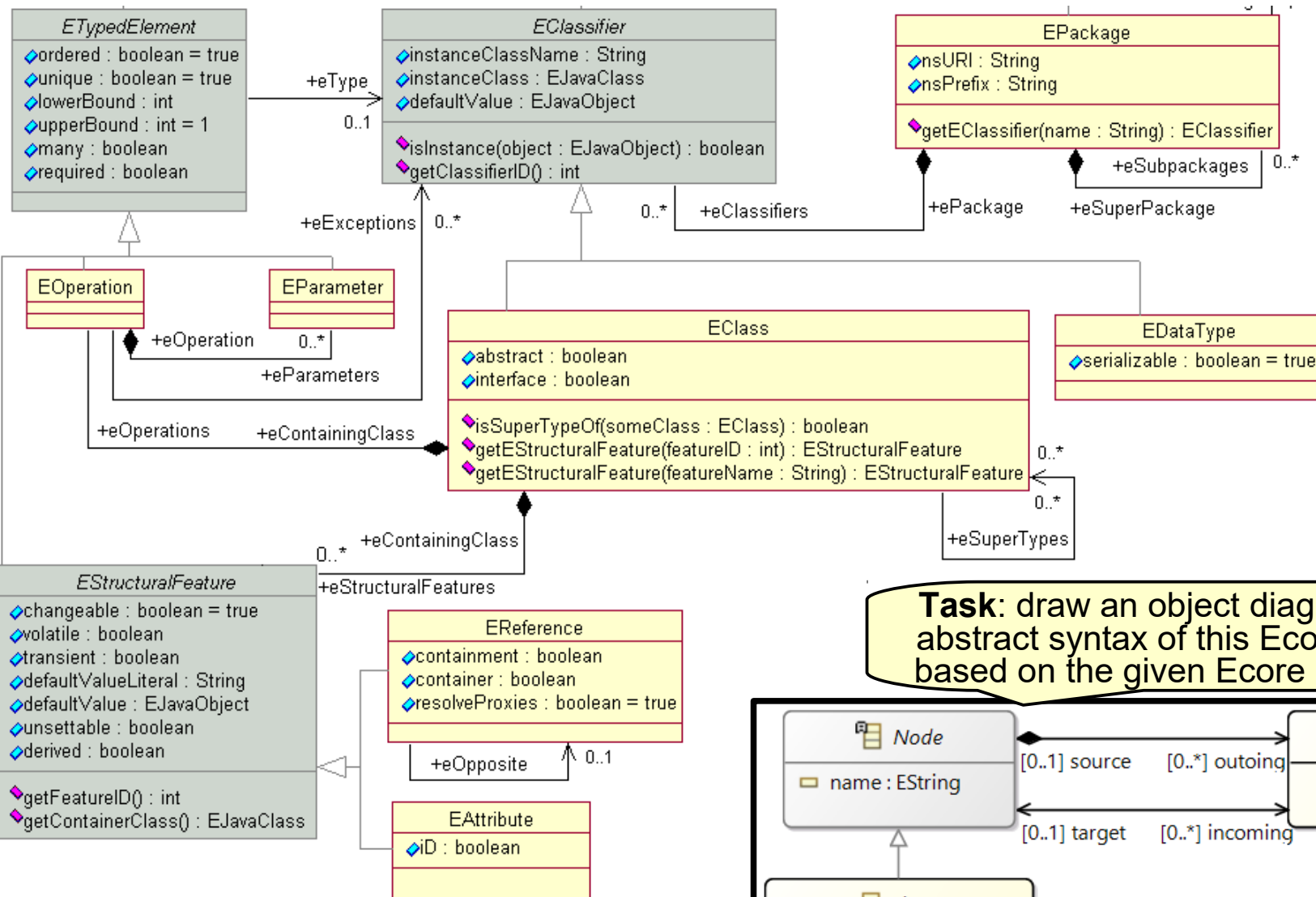


Specialties of EReferences

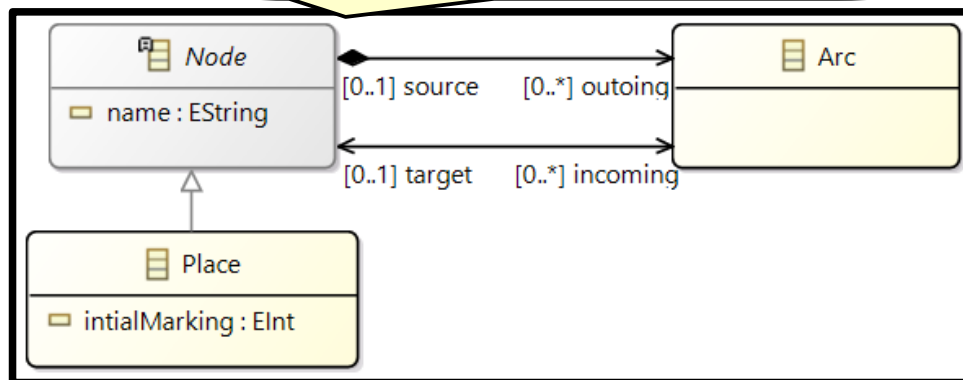
- **eOpposite:**
 - Two EReferences in opposite directions between two EClasses can be “opposites”
 - Thereby forming a bidirectional relationship
 - At the object level, there must be bidirectional links
- Example:



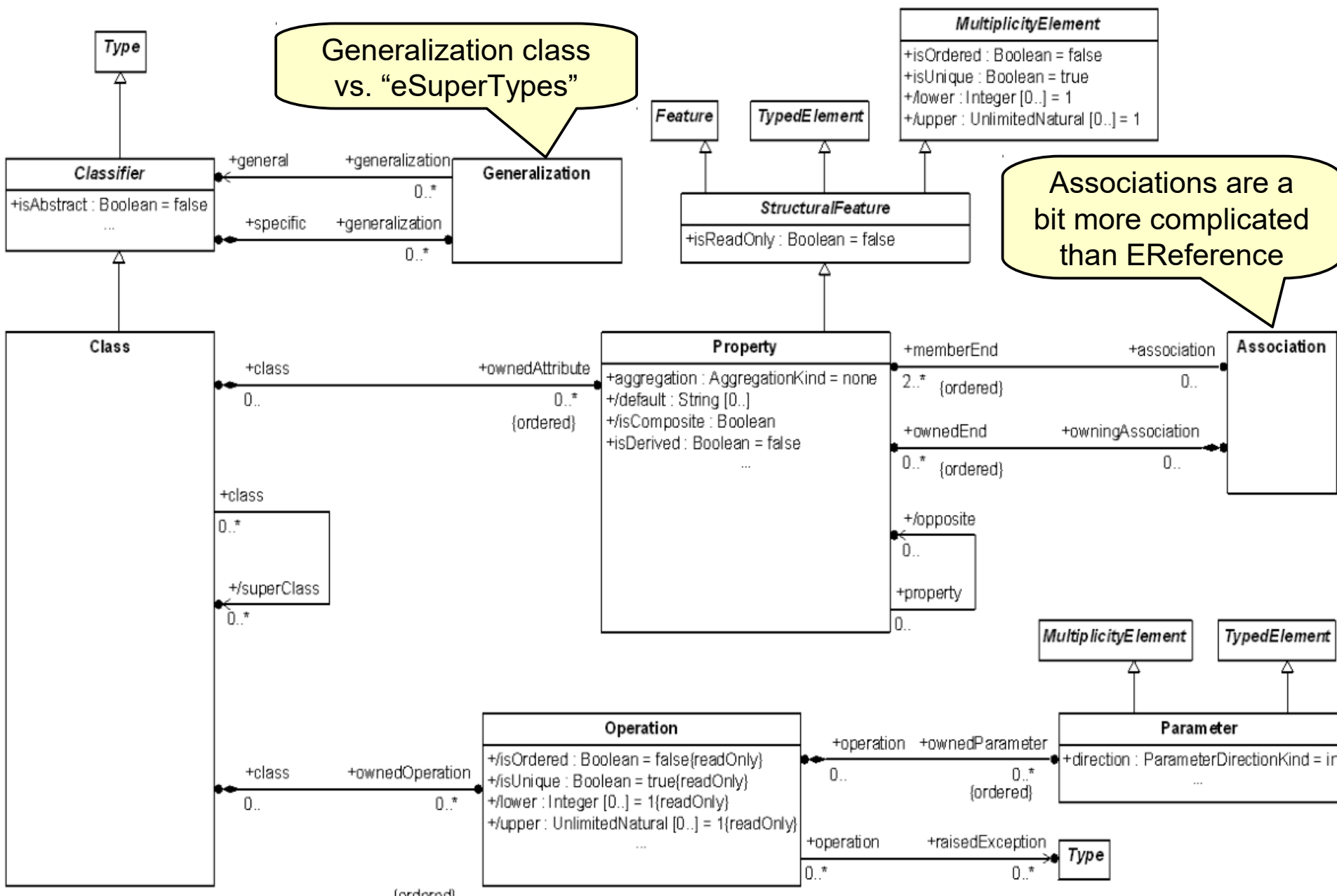
A Close Look at the Ecore Meta-Model



Task: draw an object diagram for the abstract syntax of this Ecore diagram based on the given Ecore metamodel



EMOF Classes – Difference to Ecore



Data Types in EMF

EMF uses some special **data types** that are mapped to standard Java data types

<code><<datatype>></code> EBoolean ◆ <code><<javaclass>></code> boolean	<code><<datatype>></code> EByte ◆ <code><<javaclass>></code> byte	<code><<datatype>></code> EChar ◆ <code><<javaclass>></code> char	<code><<datatype>></code> EDouble ◆ <code><<javaclass>></code> double
<code><<datatype>></code> EFloat ◆ <code><<javaclass>></code> float	<code><<datatype>></code> EInt ◆ <code><<javaclass>></code> int	<code><<datatype>></code> ELong ◆ <code><<javaclass>></code> long	<code><<datatype>></code> EShort ◆ <code><<javaclass>></code> short
<code><<datatype>></code> EString ◆ <code><<javaclass>></code> java.lang.String	<code><<datatype>></code> EJavaObject ◆ <code><<javaclass>></code> java.lang.Object	<code><<datatype>></code> EJavaClass ◆ <code><<javaclass>></code> java.lang.Class	
<code><<datatype>></code> EBooleanObject ◆ <code><<javaclass>></code> java.lang.Boolean	<code><<datatype>></code> EByteObject ◆ <code><<javaclass>></code> java.lang.Byte	<code><<datatype>></code> ECharacterObject ◆ <code><<javaclass>></code> java.lang.Character	
<code><<datatype>></code> EDoubleObject ◆ <code><<javaclass>></code> java.lang.Double	<code><<datatype>></code> EFloatObject ◆ <code><<javaclass>></code> java.lang.Float	<code><<datatype>></code> EIntegerObject ◆ <code><<javaclass>></code> java.lang.Integer	
<code><<datatype>></code> ELongObject ◆ <code><<javaclass>></code> java.lang.Long	<code><<datatype>></code> EShortObject ◆ <code><<javaclass>></code> java.lang.Short		
<code><<datatype>></code> EByteArray ◆ <code><<javaclass>></code> byte[]			

Data Types in EMF

EMF defines some additional data types

<<datatype>>
EDate

◆<<javaclass>> java.util.Date

<<datatype>>
EBigInteger

◆<<javaclass>> java.math.BigInteger

<<datatype>>
EBigDecimal

◆<<javaclass>> java.math.BigDecimal

<<datatype>>
EResource

◆<<javaclass>> org.eclipse.emf.ecore.resource.Resource

<<datatype>>
EResourceSet

◆<<javaclass>> org.eclipse.emf.ecore.resource.ResourceSet

<<datatype>>
EFeatureMapEntry

◆<<javaclass>> org.eclipse.emf.ecore.util.FeatureMap\$Entry

<<datatype>>
EFeatureMap

◆<<javaclass>> org.eclipse.emf.ecore.util.FeatureMap

<<datatype>>
EEnumerator

◆<<javaclass>> org.eclipse.emf.common.util.Enumerator

<<datatype>>
EEList

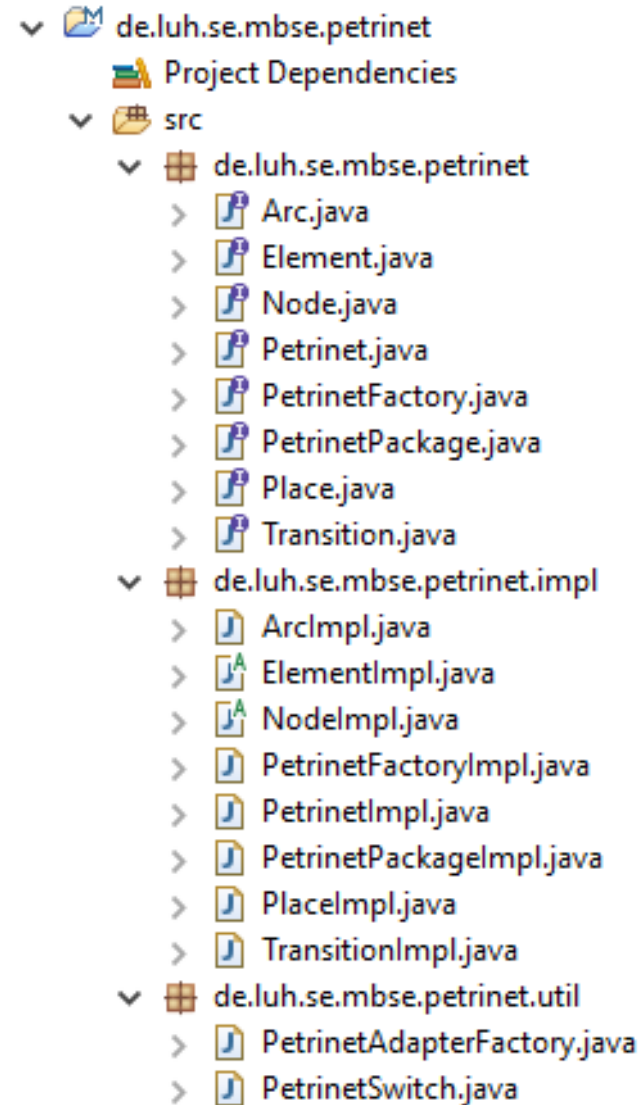
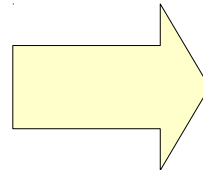
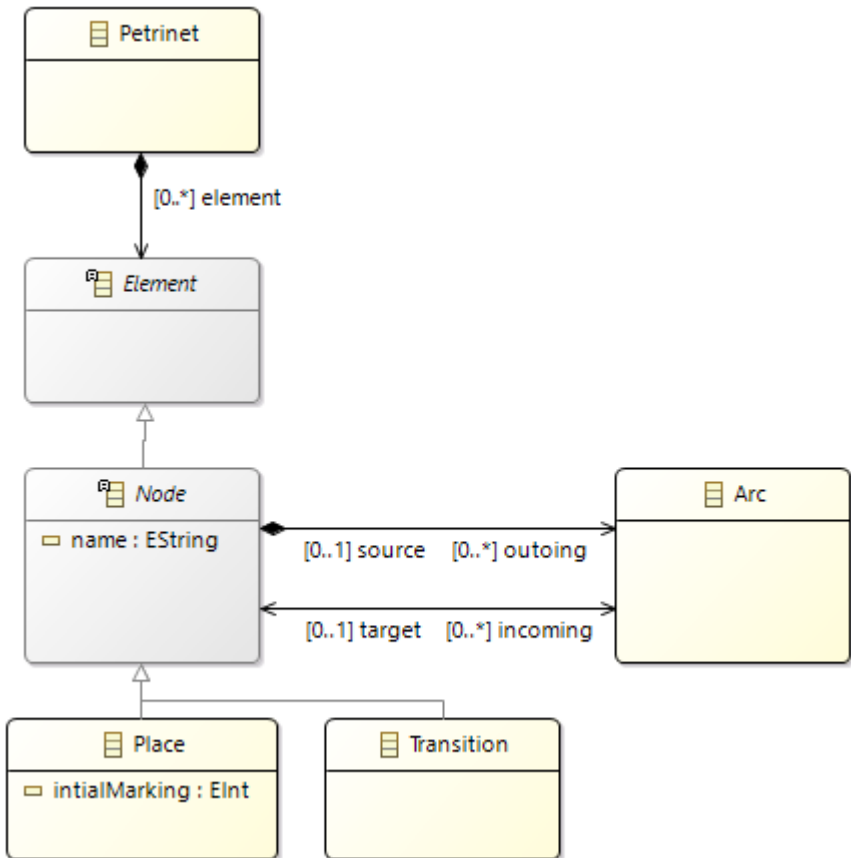
◆<<javaclass>> org.eclipse.emf.common.util.EList

<<datatype>>
ETreeIterator

◆<<javaclass>> org.eclipse.emf.common.util.TreeIterator

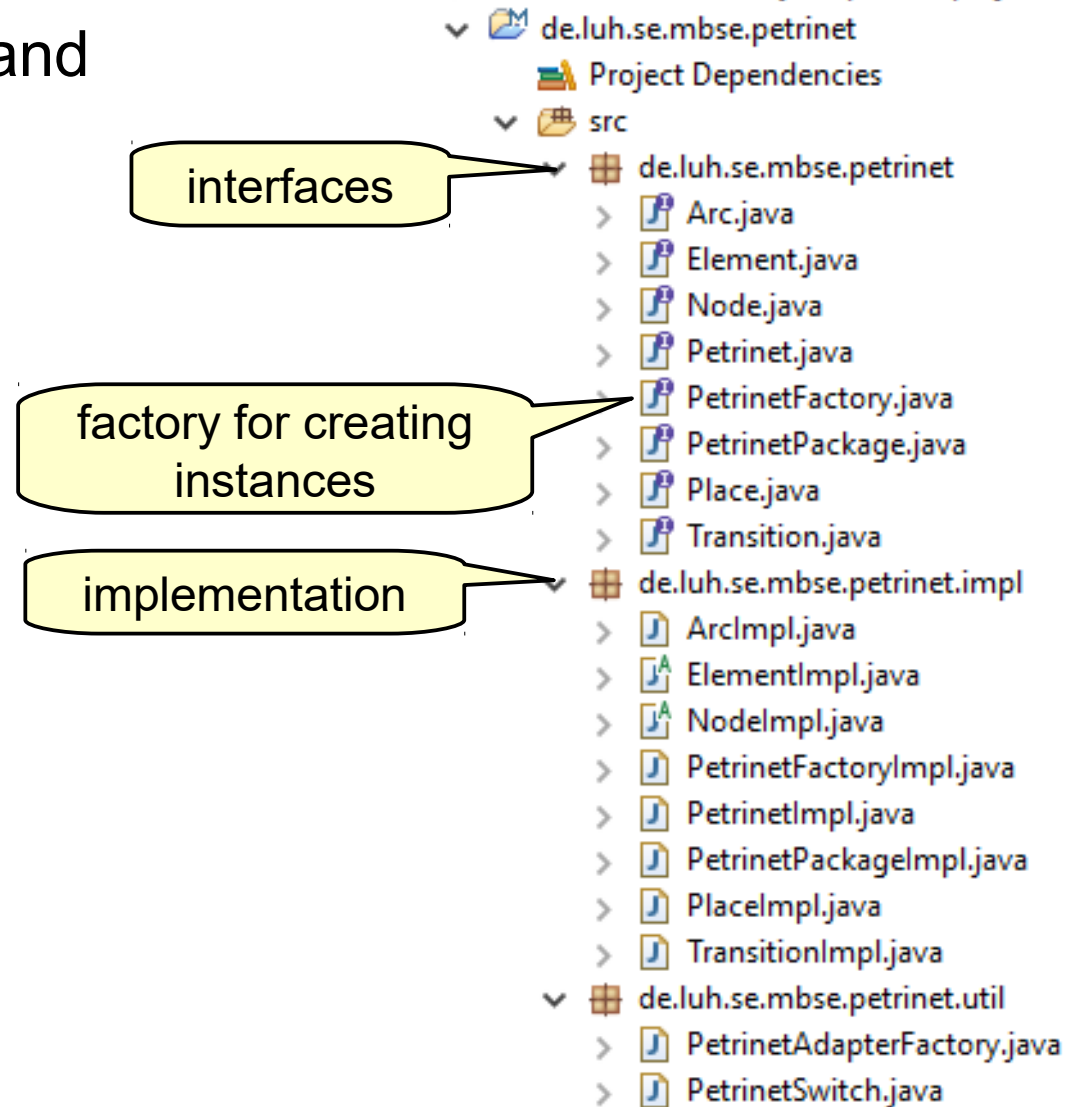
EMF Code Generation

- EMF supports Java code generation from Ecore models



EMF Code Generation

- Separation of Interfaces and Implementation

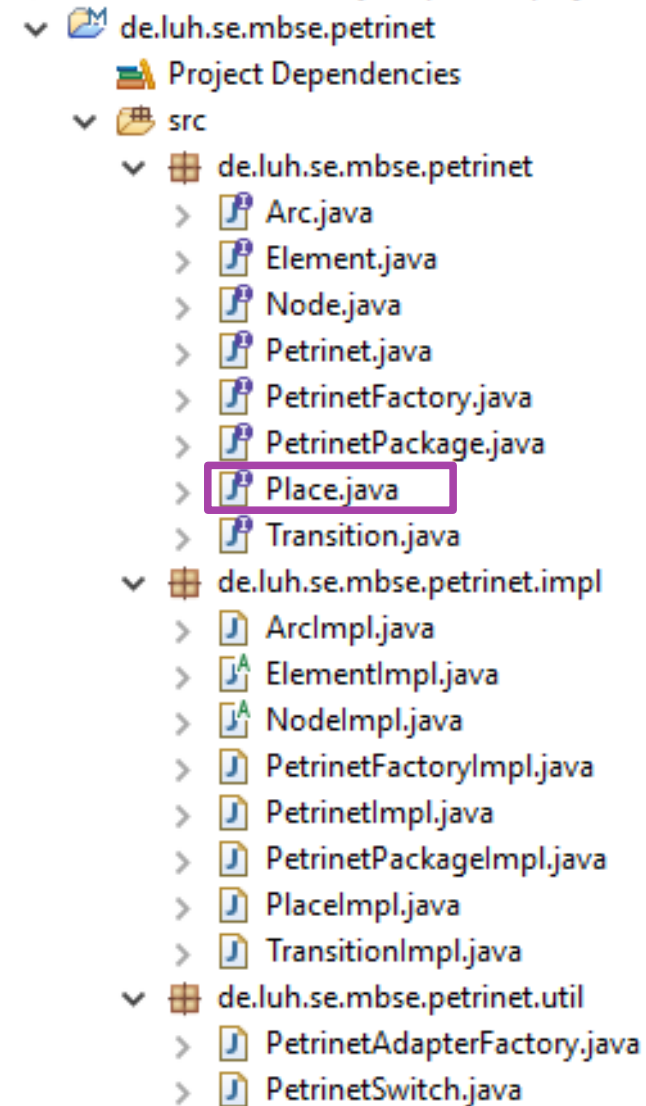


- Separation of Interfaces and Implementation

```
public interface Place extends Node {
    /**
     * Returns the value of the
     * '<em><b>Initial Marking</b></em>' attribute.
     * @generated
     */
    int getInitialMarking();

    /**
     * Sets the value of the
     * '<em>Initial Marking</em>' attribute.
     * @generated
     */
    void setInitialMarking(int value);

    ...
} // Place
```



- 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;

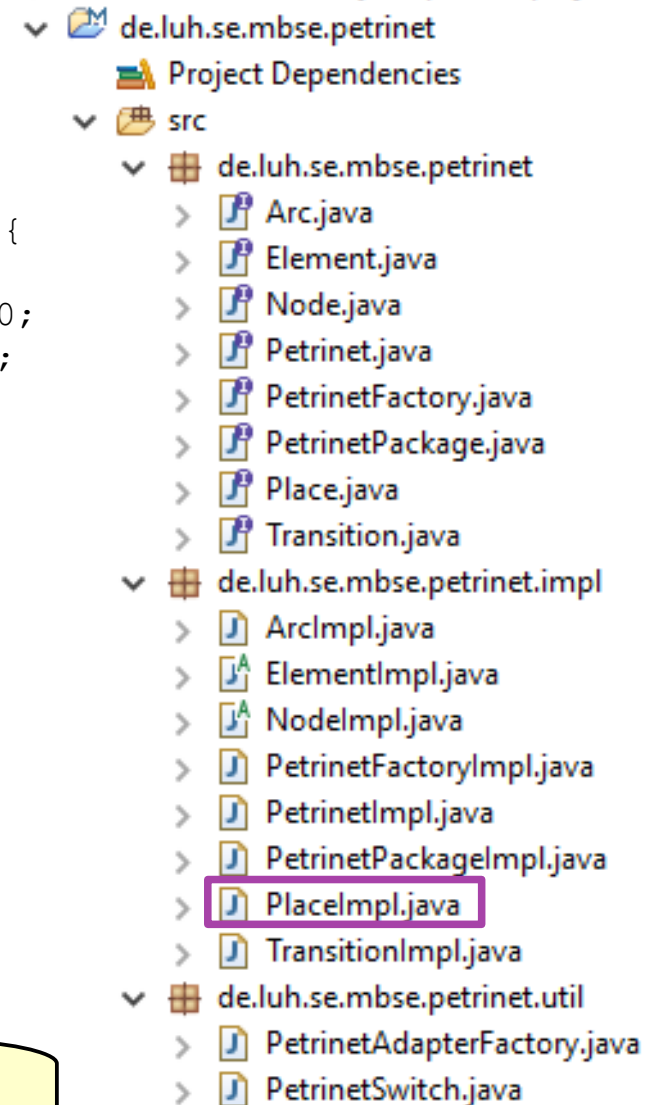
    public int getIntialMarking() {
        return intialMarking;
    }

    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



- Factory interface and implementation

```
public interface PetrinetFactory extends EFactory {

    PetrinetFactory eINSTANCE
    = de.luh.se.mbse.petrinet.impl.PetrinetFactoryImpl.init();

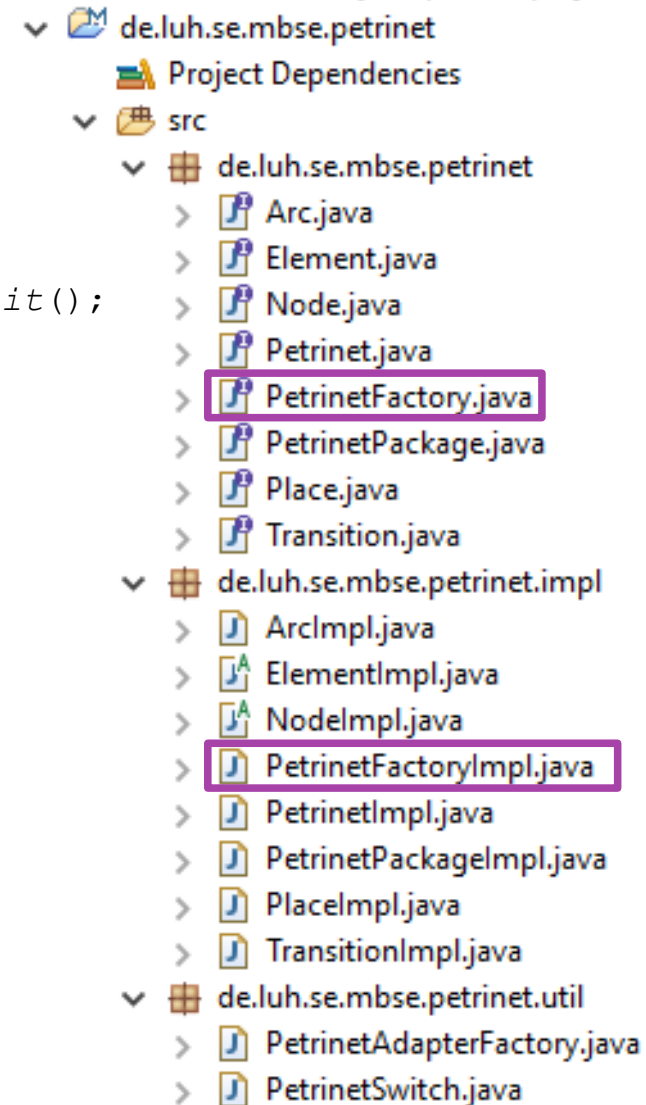
    Petrinet createPetrinet();
    ...
} //PetrinetFactory

public class PetrinetFactoryImpl extends EFactoryImpl
    implements PetrinetFactory {

    public static PetrinetFactory init() {
        return new PetrinetFactoryImpl();
    }

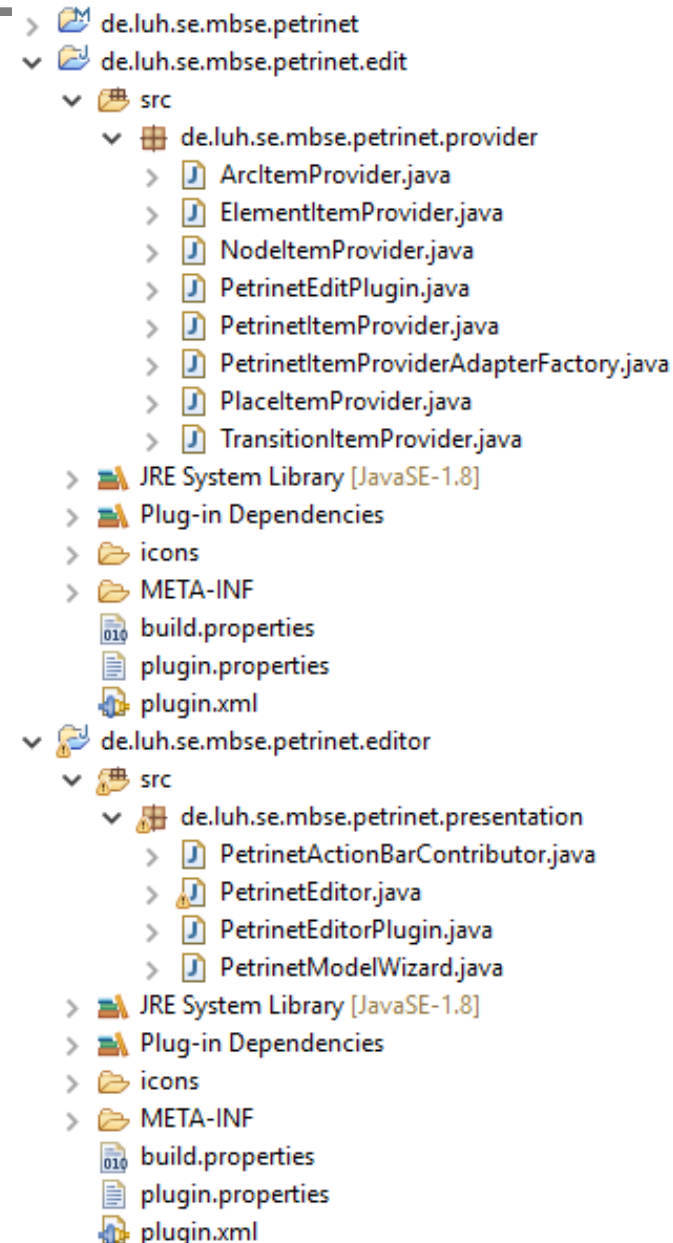
    public Petrinet createPetrinet() {
        PetrinetImpl petrinet = new PetrinetImpl();
        return petrinet;
    }
    ...

} //PetrinetFactoryImpl
```



Edit and Editor Code

- EMF also supports the generation of code for *building editors and views* within Eclipse:
 - *.edit* plug-in:
 - Content and label provider classes, for table and tree views
 - property source support for property pages
 - command framework for undo/redo
 - *.editor* plug-in:
 - tree editor
 - model creation wizards

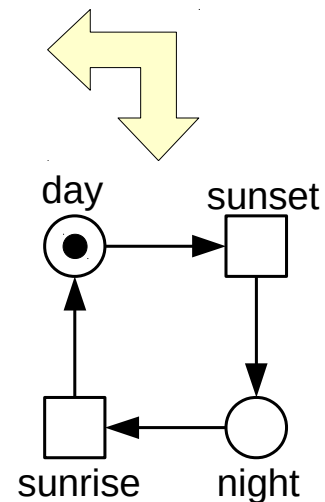


Model Persistence – XMI Serialization

- EMF models can be serialized in the **XML Metadata Interchange (XMI)** format
 - also an OMG standard: <http://www.omg.org/spec/XMI/>
 - XML format for exchanging models with metamodels conforming to MOF

Model Persistence – XML Serialization (Example)

```
<?xml version="1.0" encoding="UTF-8"?>
<petrinet:Petrinet xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:petrinet="http://www.example.org/petrinet"
  xsi:schemaLocation="http://www.example.org/petrinet
    ../de.luh.se.mbse.petrinet/model/petrinet.ecore">
  <element xsi:type="petrinet:Place" name="day"
    incoming="//@element.3/@outgoing.0" initialMarking="1">
    <outgoing target="//@element.1"/>
  </element>
  <element xsi:type="petrinet:Transition" name="sunset"
    incoming="//@element.0/@outgoing.0">
    <outgoing target="//@element.2"/>
  </element>
  <element xsi:type="petrinet:Place" name="night"
    incoming="//@element.1/@outgoing.0">
    <outgoing target="//@element.3"/>
  </element>
  <element xsi:type="petrinet:Transition" name="sunrise"
    incoming="//@element.2/@outgoing.0">
    <outgoing target="//@element.0"/>
  </element>
</petrinet:Petrinet>
```



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

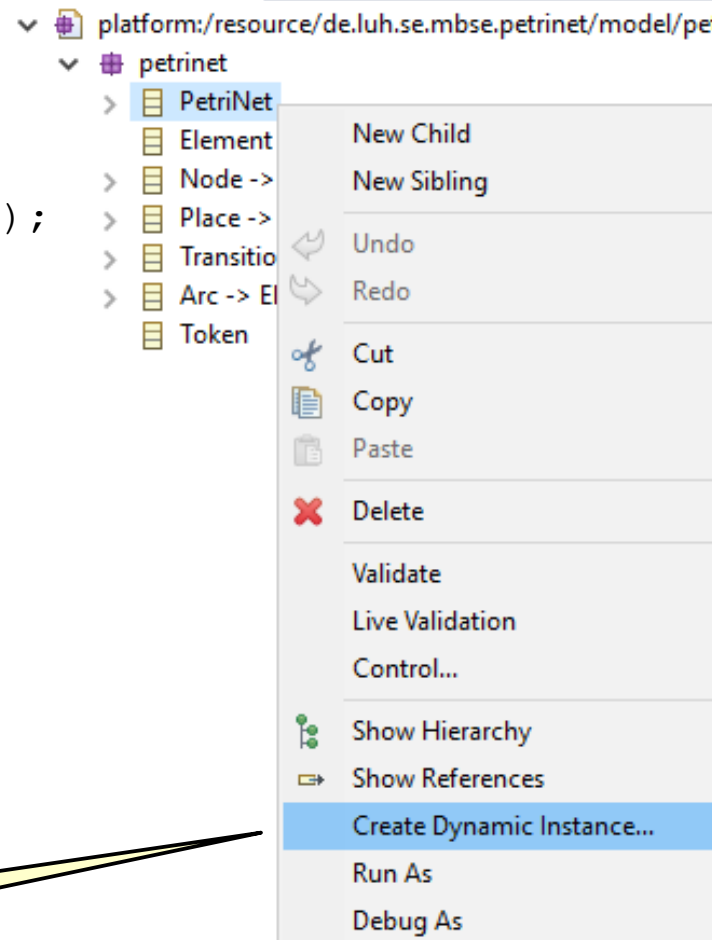
```
public Petrinet createPetrinet() {  
    PetrinetImpl petrinet = new PetrinetImpl();  
    return petrinet;  
}
```

- The generated code allows us to create instances of our Ecore models

– for example:

```
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

- creating models and instances dynamically via the API:

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

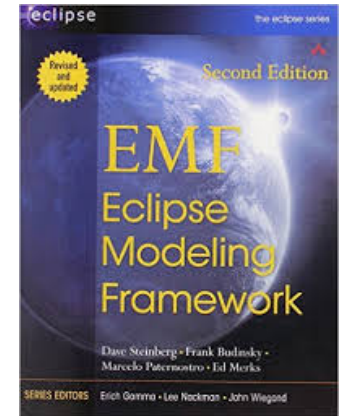
//create Place class
EClass placeClass = EcoreFactory.eINSTANCE.createEClass();
placeClass.setName("Place");
petrinetPackage.getEClassifiers().add(placeClass);

//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.



- 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 03 – Metamodeling cont., OCL

Prof. Dr. Joel Greenyer



April 18, 2016

