Modellbasierte Softwareentwicklung (MODSOFT)

Part II

Domain Specific Languages

EMF

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*

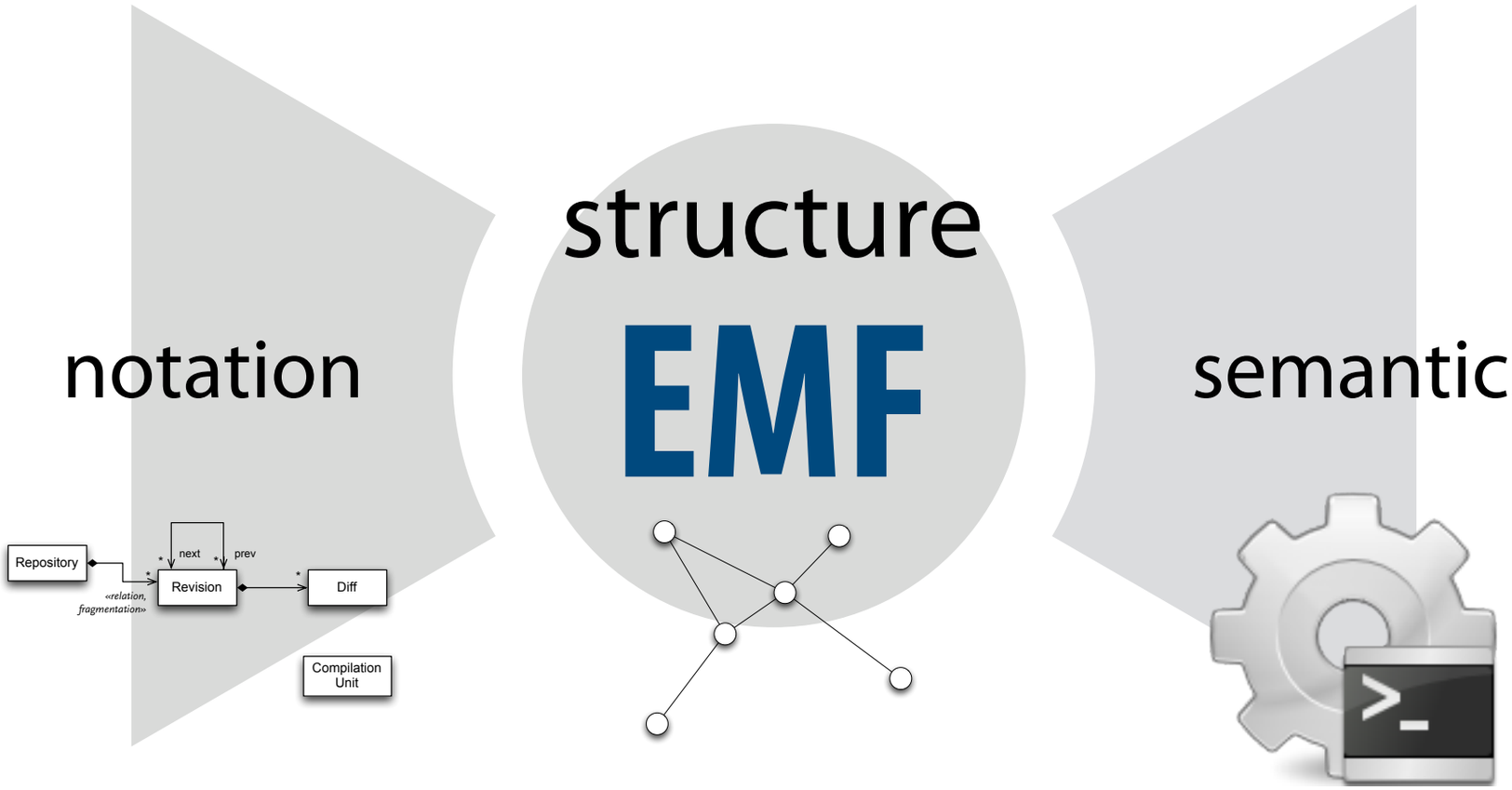
2. (3 VL)
Notation: Customizing the tree-editor, textural with *XText*, graphical with *GEF* and *GMF*

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)

Previously on MODSOFT

Eclipse Modeling Framework

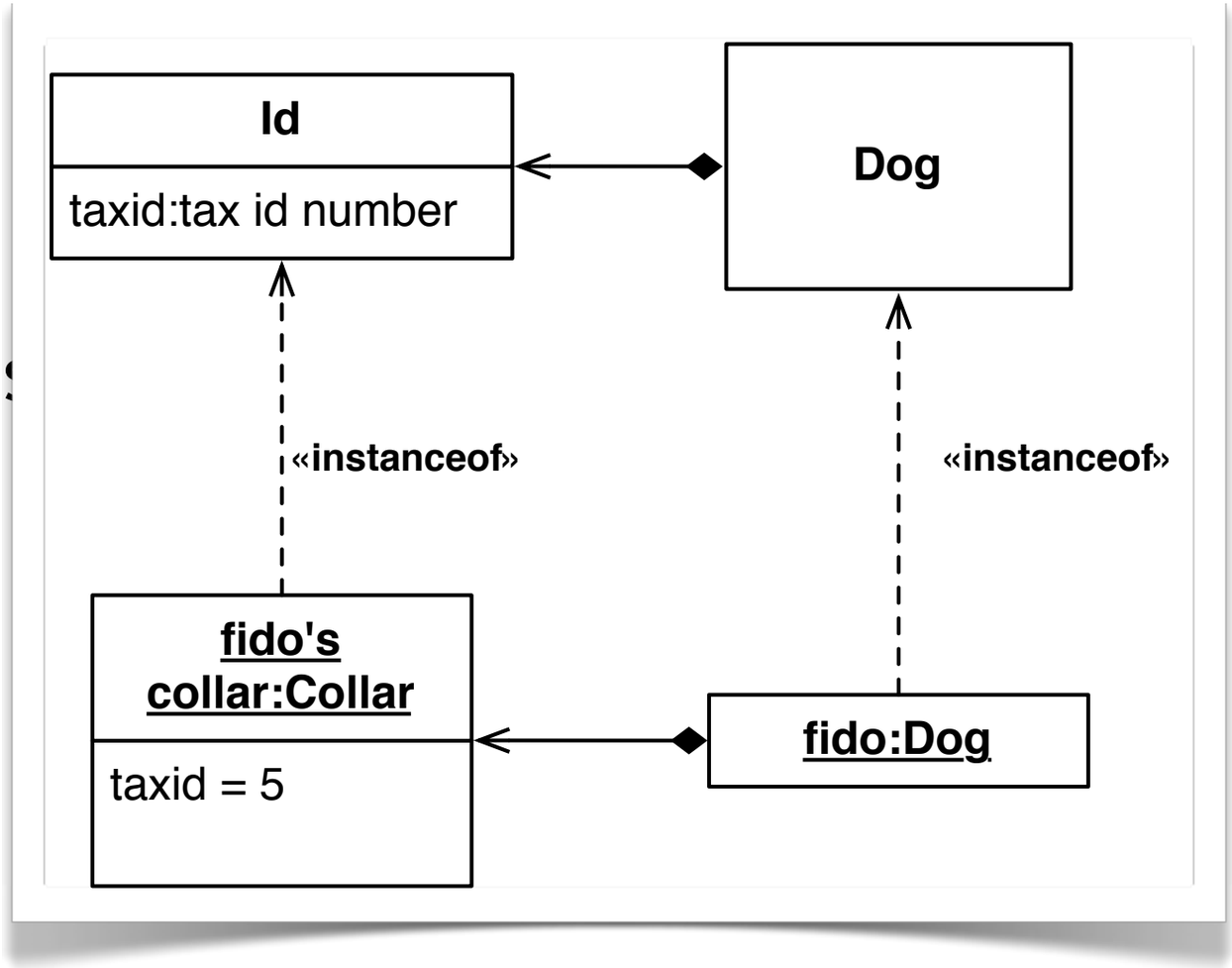


Instantiation: Attribute, Reference/ Property, Link

- Instantiation extends to class features.
- A structural feature defines
  - name
  - type (usually a classifier)
  - multiplicity, visibility
- A structural feature is instantiated through sets (multiplicity!) of properties and links (i.e. value sets)

Instantiation: Attribute, Reference/ Property, Link

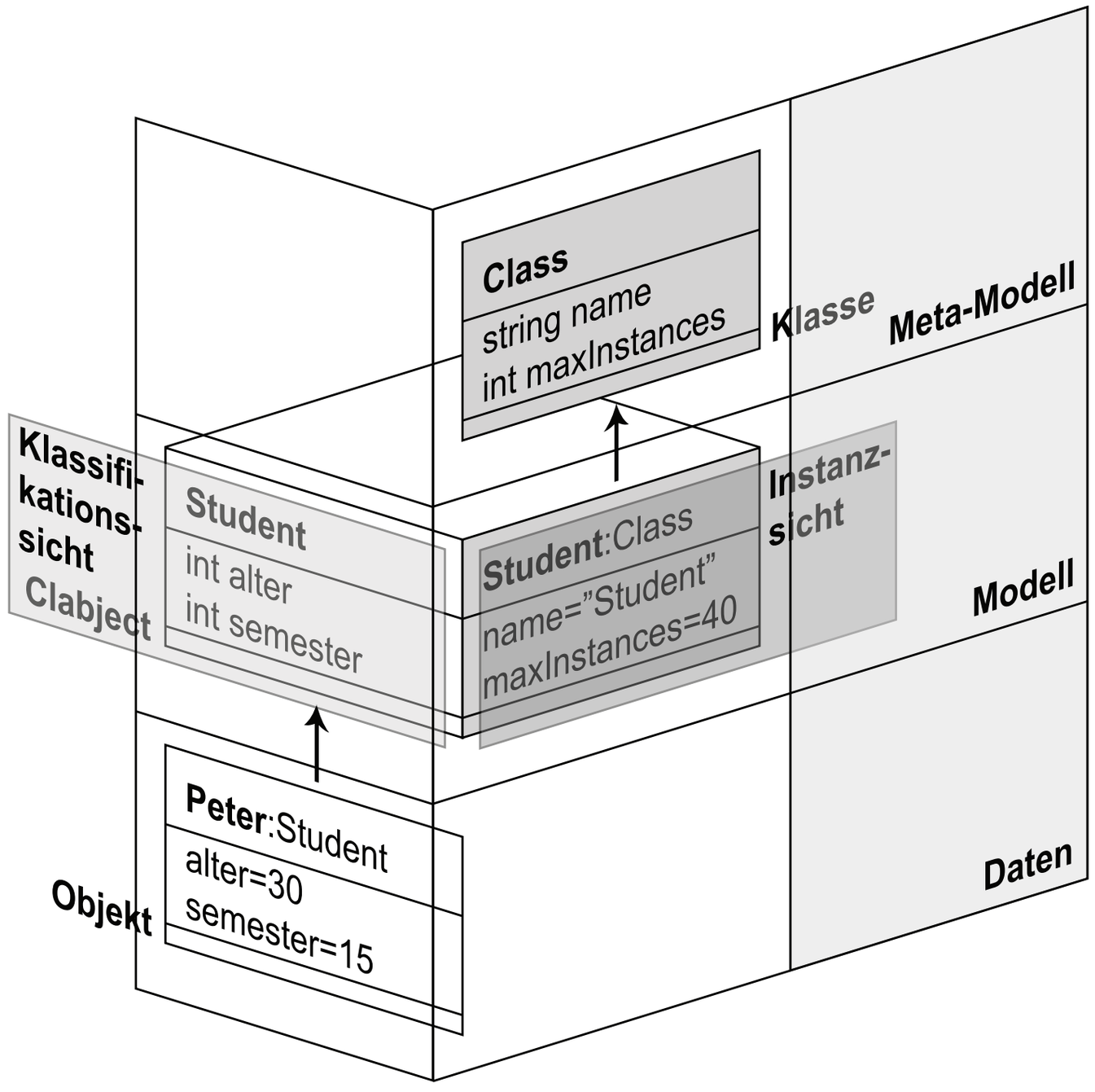
- Instantiation extends to class
- A structural feature defines
  - name
  - type (usually a classifier)
  - multiplicity, visibility



- A structural feature is instantiated through sets (multiplicity!) of properties and links (i.e. value sets)

Clabject

- Model elements are both Objects and Types (Classes) [1]
- Clabject is an illustration of this duality [2]



1. Jim Odell: *Power Types*, Journal of Object-Oriented Programming, 1994
2. Colin Atkinson, Thomas Kühne: *Meta-Modeling Distributed Environments*, Enterprise Distributed Computing, 1997

Examples for Meta-Modeling Formalisms

	OO-Meta-Modeling	**XML**	**Rel. DBs.**	**Grammars**	**Automata**
M3	MOF	Schema-Schema	Rel. Alg.	Math	Math
M2	UML	Schema	E.-R.-M.	Grammar	Automata
M1	Model	XML	DB	Program	Word
M0	Things	Data	Data	Thing	Things

Eclipse Modeling Framework (EMF)

Examples for Meta-Modeling Formalisms

	OO-Meta-Modeling	**EMF**
M3	MOF	ECore
M2	UML	Ecore model – Java classes
M1	Model	Model – Java objects
M0	Things	Data

EMF Concepts

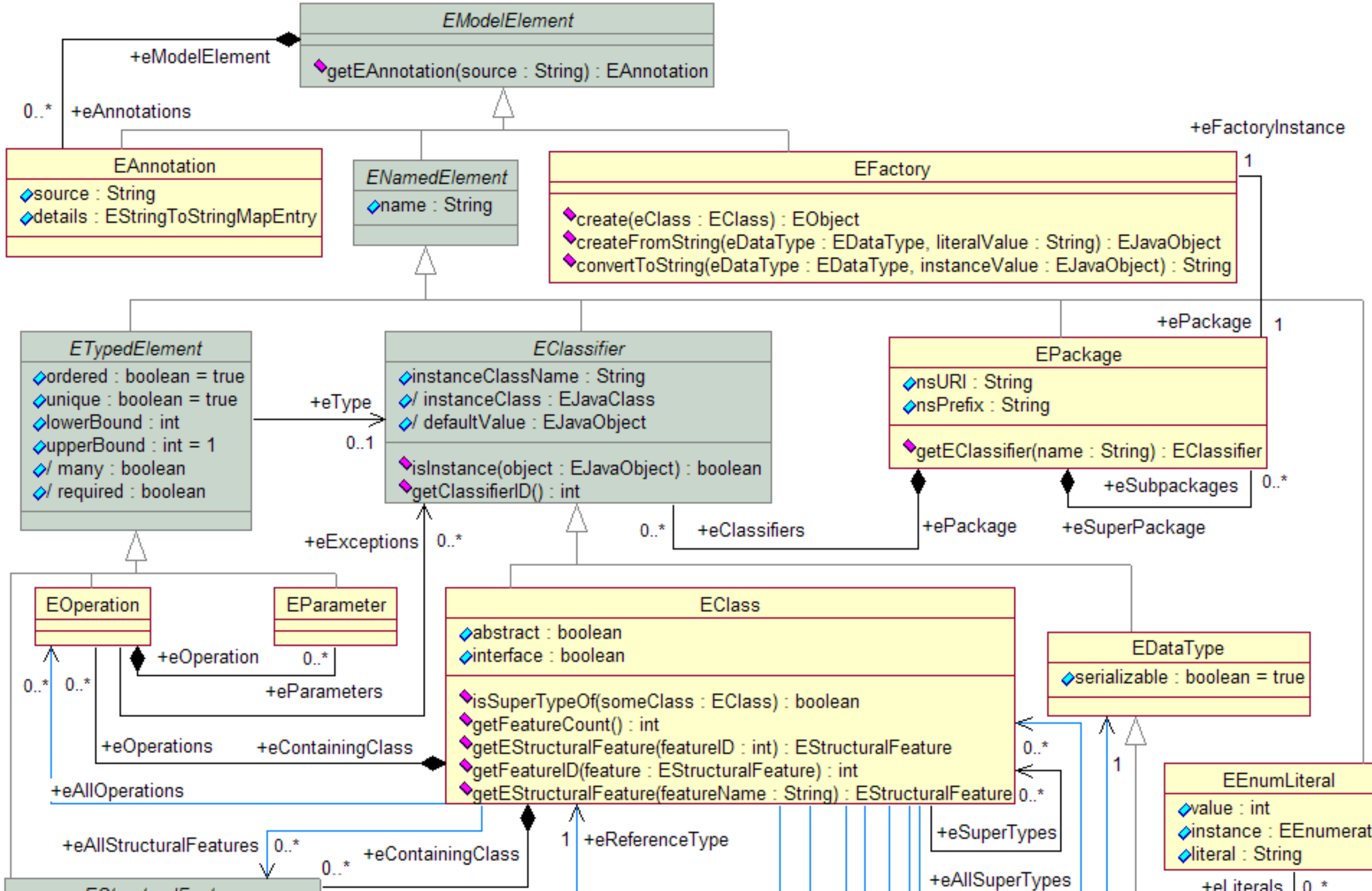
- Ecore model
- Java mapping
- Reflection
- XMI, URIs, resources, and resource sets
- Notifications and Adapter
- Edit and editor
- Generator models

The Ecore Model of (EMF)

Ecore Meta-Meta-Model

- Basically EMOF from MOF 2.0
  - all classifier names start with “E” (to deal with *replication of concepts*)
  - no associations, only association ends (`EReference`)
  - aligned with Java
    - Primitive types resemble Java primitive types (`EInt`, `EString`, `EDouble`, etc.)
    - Java classes as `EDataType`
  - objects are not directly part of the Ecore model, but each element on the lower layers will inherit from Java class `EObject` (*Power Type*-like approach to deal with *ambiguous instantiation*)

Ecore Meta-Meta-Model



source : String
details : EStringToStringMapEntry

ENamedElement
name : String

create(eClass : EClass) : EObject
createFromString(eDataType : EDataType, literalValue : String) : EJavaObject
convertToString(eDataType : EDataType, instanceValue : EJavaObject) : String

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

+eType
0..1

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

+ePackage
1

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

+eSubpackages
0..*
+eSuperPackage

0..*
+eClassifiers
+ePackage

+eExceptions
0..*

EOperation
EParameter
0..*
+eOperation
0..*
+eParameters
0..*
0..*
+eOperations
+eContainingClass
+eAllOperations

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

EDataType
serializable : boolean = true

0..*
0..*
+eSuperTypes
+eAllSuperTypes

1

EEnumLiteral
value : int
instance : EEnumerator
literal : String

+eLiterals
0..*
+eEnum

+eAttributeType

+eAllStructuralFeatures
0..*
0..*
+eContainingClass
+eStructuralFeatures

1
+eReferenceType

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

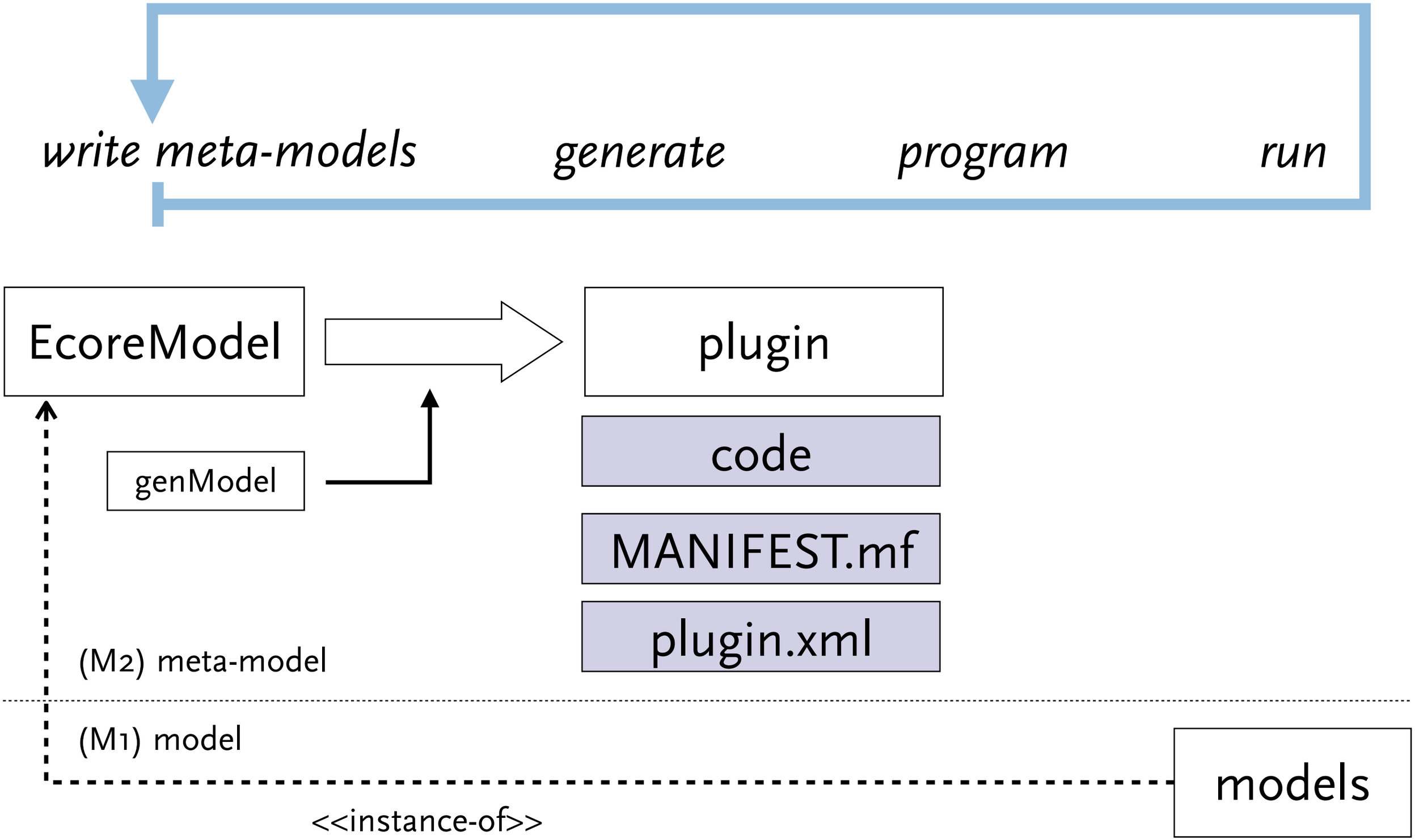
+eAllContainments
0..*
EReference
containment : boolean
/ container : boolean
resolveProxies : boolean = true
0..*
+eAllReferences
0..*
+eReferences
+eOpposite
0..1

0..*
+eAllAttributes
0..*
+eAttributes
0..1
+eIDAttribute
EAttribute
iD : boolean

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

The Ecore-to-Java Mapping(s) of EMF

EMF – Workflow



Overview

- Strict separation of interface and implementation
- Interface
- Implementation, varies in
  - Generated classes
  - Abstract classes
  - Meta-class depended behavior

Names

- Requires well-formed Java names
  - class name “$Dog()” -> error in .genmodel
- Names are sometimes modified to match conventions
  - reference name “pets” -> access method name “getPets”
- Mapping of names is often configurable
  - e.g. package prefixes “fido”->”de.hub.modsoft.fido”

Packages

- For each EPackage EMF generates three (more or less) Java packages
  - `<genmodel-prefix>.<name>`
  - `<genmodel-prefix>.<name>.impl`
  - `<genmodel-prefix>.<name>.util`

Interfaces

- EPackage ← <Name>Package

  EClass get<ClassName>()

  EStructuralFeature get<ClassName>_<FeatureName>()

- EFactory ← <Name>Package

  <Class> create<ClassName>()

- EObject ← <Name>
  - getter and setter for single valued features

    <Type> get<Name>()

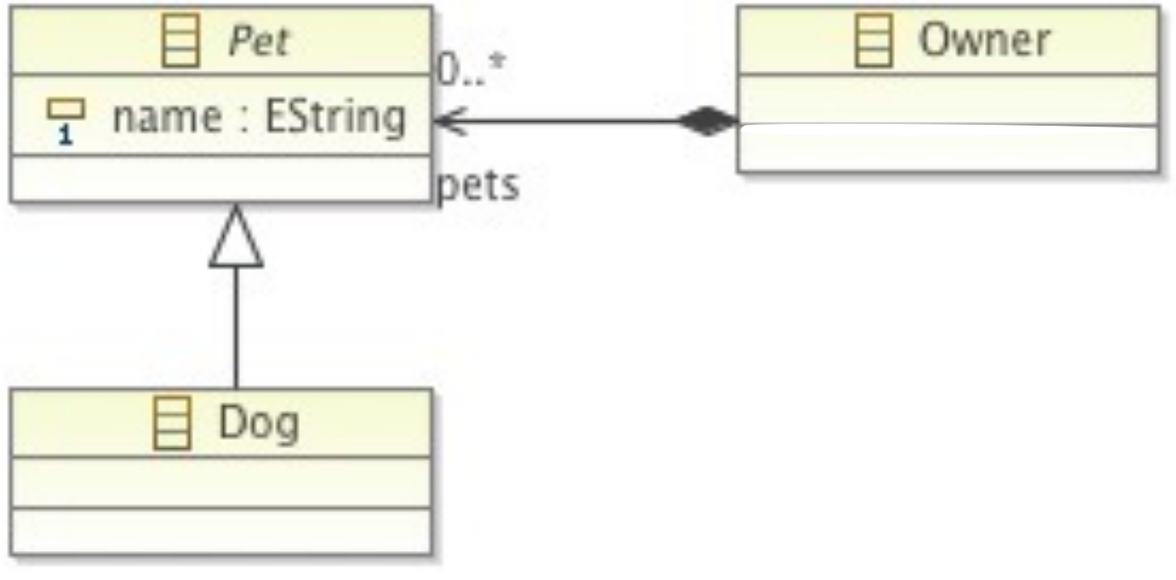
    void set<Name>(<Type>)
  - getter for multi valued features

    EList<Type> get<Name>()

- EList
  - interface for value sets

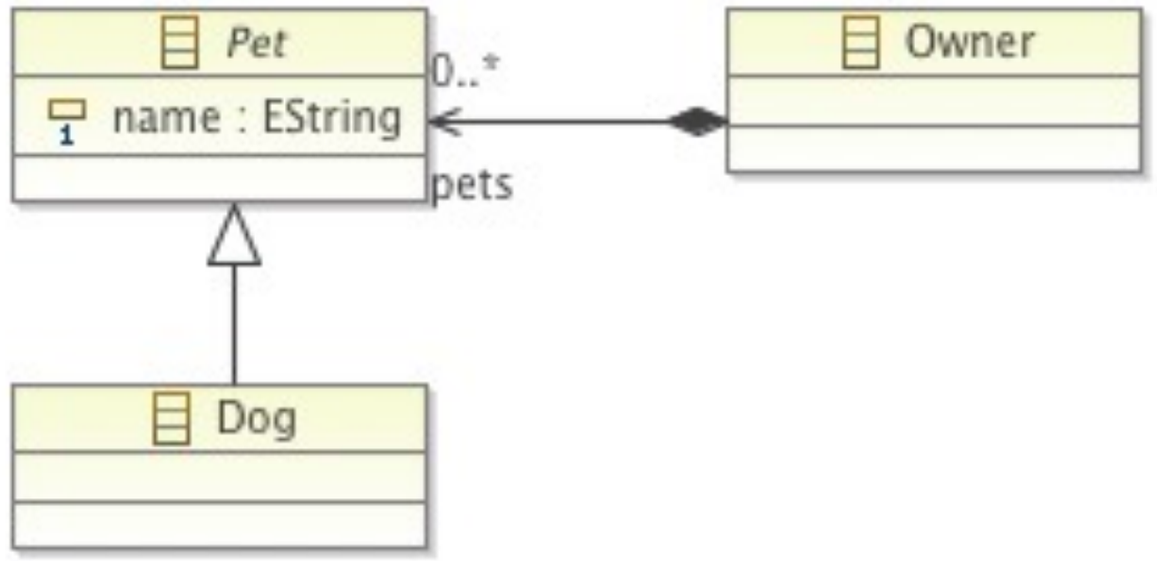
Interfaces – EPackage, EFactory, EClass/EObject

```
package fido;

public interface FidoPackage extends EPackage {
    static FidoPackage eINSTANCE;

    EClass getPet();
    EClass getDog();
    EClass getOwner();
    EReference getOwner_pets();
    EAttribute getPet_name();
}

public interface FidoFactory extends EFactory {
    static FidoFactory eINSTANCE;

    Dog createDog();
    Owner createOwner();
}

public interface Pet extends EObject { ... }

public interface Dog extends Pet { ... }

public interface Owner extends EObject { ... }
```



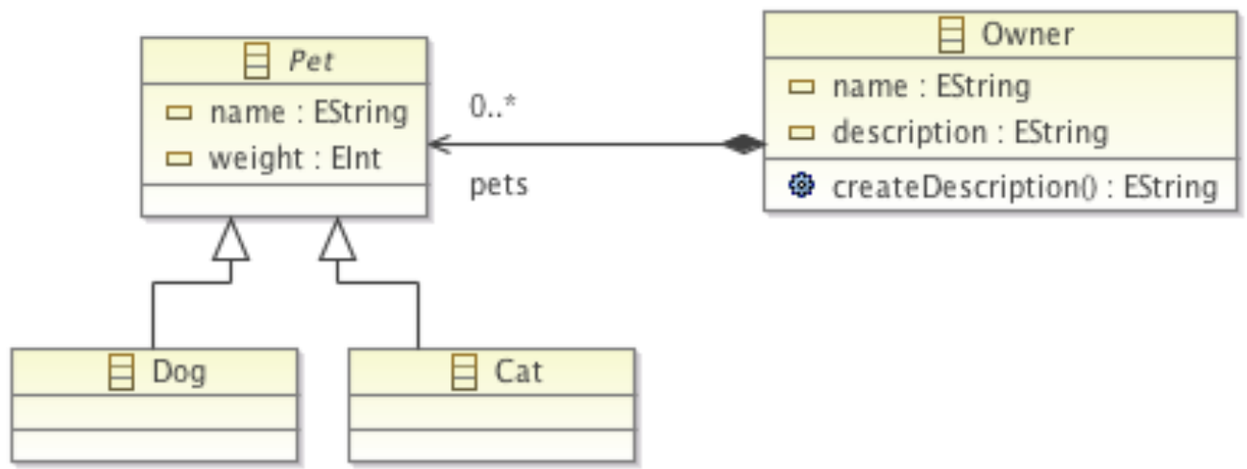
Interfaces – EClass/EObject, EAttribute, EReference

```
package fido;

public interface Pet extends EObject {
    String getName();
    void setName(String name);
}

public interface Dog extends Pet {
}

public interface Owner extends EObject {
    EList<Pet> getPets();
}

public interface EList<E> extends List<E> {
    ...
}
```



Interfaces – Operations, derived Features

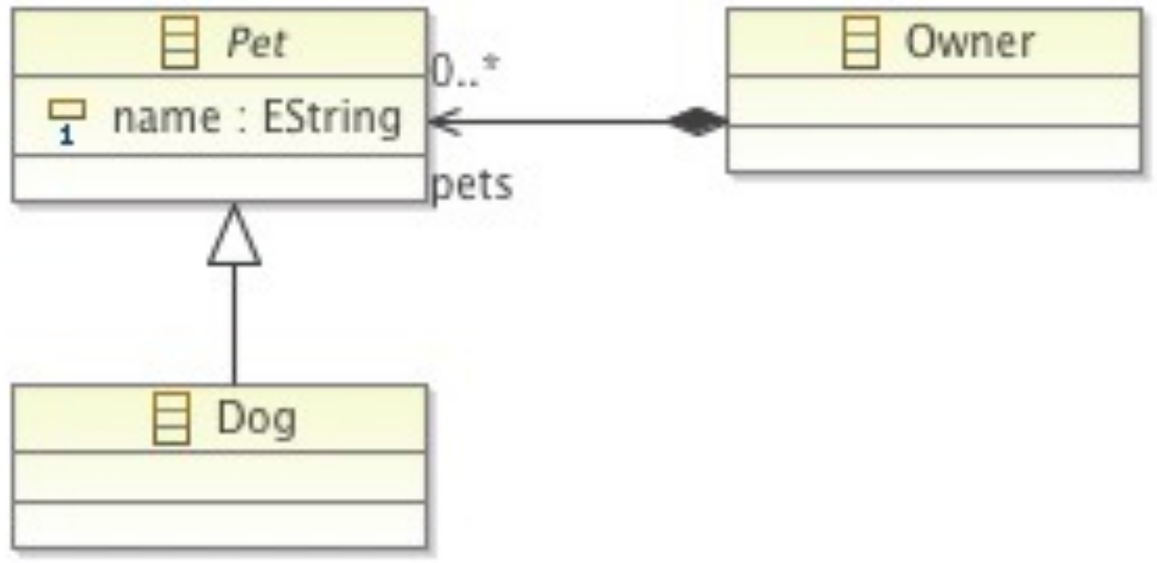
`description` is **volatile**



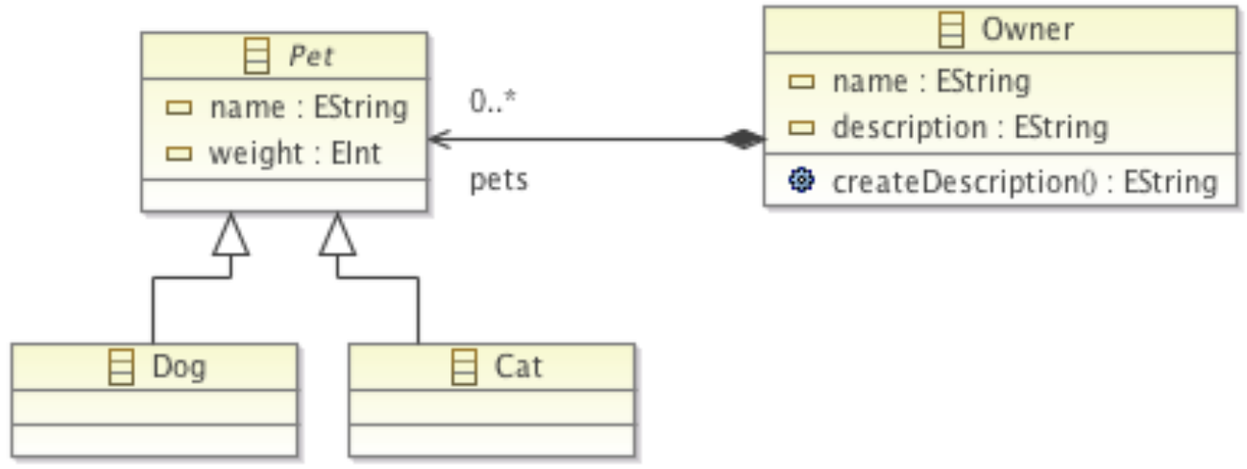

```
package fido;

public interface Owner extends EObject {

    EList<Pet> getPets();

    String getDescription();

    boolean isSetDescription();

    String createDescription();
}
```

Implementation – Objects and Features

```
package fido.impl;

public abstract class PetImpl extends EObjectImpl
      implements Pet {
    protected String name;
    ...
}

public class DogImpl extends PetImpl implements Dog {
    ...
}

public class OwnerImpl extends EObjectImpl
      implements Owner {
    protected EList<Pet> pets;
    ...
}
```

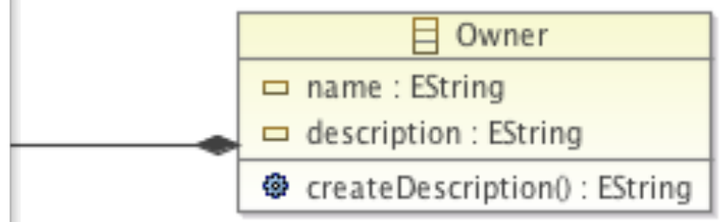


Implementation – Operations, derived Features



Implementation – Operations, derived Features

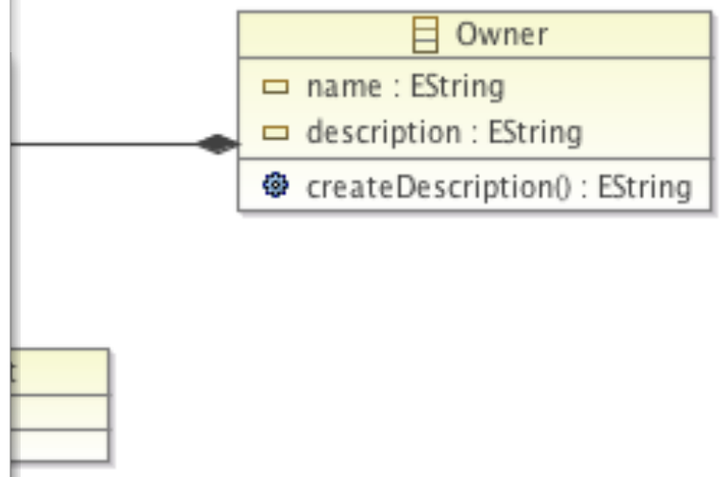
```java
/**
 * @generated
 */
public String getDescription() {
    // TODO: implement this method to return the 'Description' attribute
    // Ensure that you remove @generated or mark it @generated NOT
    throw new UnsupportedOperationException();
}

/**
 * @generated
 */
public boolean isSetDescription() {
    // TODO: implement this method to return whether the 'Description'
    // attribute is set
    // Ensure that you remove @generated or mark it @generated NOT
    throw new UnsupportedOperationException();
}

/**
 * @generated
 */
public String createDescription() {
    // TODO: implement this method
    // Ensure that you remove @generated or mark it @generated NOT
    throw new UnsupportedOperationException();
}
```



Implementation – Operations, derived Features

```
/**
 * @g
 */
publi
     /
     /
     t
}

/**
 * @g
 */
publi
     /
     /
     t
}

/**
 * @g
 */
publi
     /
     /
     t
}
```

```
     /**
      * @generated NOT
      */
     public String getDescription() {
          return createDescription();
     }

     /**
      * @generated NOT
      */
     public boolean isSetDescription() {
          return true;
     }

     /**
      * @generated NOT
      */
     public String createDescription() {
          StringBuffer result = new StringBuffer();
          result.append(getName());
          boolean first = true;
          for(Pet pet: getPets()) {
               if (first) {
                    result.append(" has");
                    first = false;
               } else {
                    result.append(" and");
               }
               result.append(" a " + pet.eClass().getName());
               result.append(" " + pet.getName() + "(" + pet.getWeight() + "kg)");
          }
          return result.toString();
     }
```



Additional, Yet Important Artifacts

- fido.util
- MANIFEST.mf

```
Export-Package: fido,
 fido.impl,
 fido.util
Require-Bundle: org.eclipse.core.runtime,
 org.eclipse.emf.ecore;
```

- plugin.xml

```
<extension point="org.eclipse.emf.ecore.generated_package">
    <package
          uri="http://fido/1.0"
          class="fido.FidoPackage"
          genModel="model/Fido.genmodel"/>
</extension>
```

Example Usage of the Generated Code

Example Usage of the Generated Code

```
package de.hub.modsoft.fido;

import fido.Dog;
import fido.FidoFactory;
import fido.Owner;

public class TestFido {

    public static void main(String[] args) {
        Owner markus = FidoFactory.eINSTANCE.createOwner();
        markus.setName("Markus");
        Dog fido = FidoFactory.eINSTANCE.createDog();
        fido.setName("Fido");
        fido.setWeight(20);
        markus.getPets().add(fido);

        System.out.println(markus.getDescription());
    }

}
```

Example Usage of the Generated Code

```
/**
 * @generated NOT
 */
public String createDescription() {
    StringBuffer result = new StringBuffer();
    result.append(getName());
    boolean first = true;
    for(Pet pet: getPets()) {
        if (first) {
            result.append(" has");
            first = false;
        } else {
            result.append(" and");
        }
        result.append(" a " + pet.eClass().getName());
        result.append(" " + pet.getName() + "(" + pet.getWeight() + "kg)");
    }
    return result.toString();
}
```

Example Usage of the Generated Code

```
/**
 * @generated NOT
 */
public String createDescription() {
    StringBuffer result = new StringBuffer();
    result.append(getName());
    boolean first = true;
    for(Pet pet: getPets()) {
        if (first) {
            result.append(" has");
            first = false;
        } else {
            result.append(" and");
        }
        result.append(" a " + pet.eClass().getName());
        result.append(" " + pet.getName() + "(" + pet.getWeight() + "kg)");
    }
    return result.toString();
}
```

```
Markus has a Dog Fido(20kg)
```

Thats all? I could have done this with just regular Java classes. Why bother with EMF?

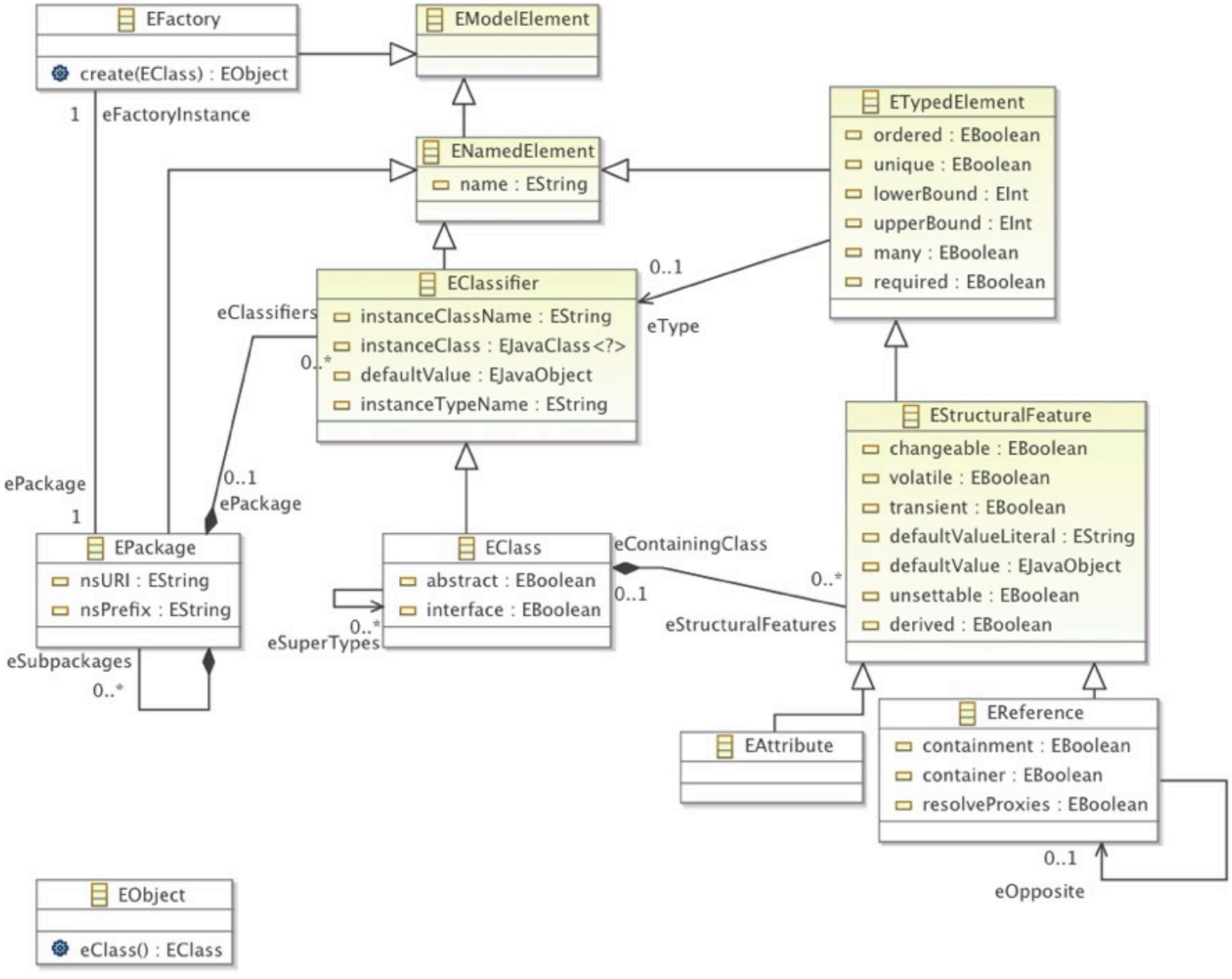
```
/**
 * @ger
 */
public
    Str
    resu
    boo
    for(Pet pet: getPets()) {
        if (first) {
            result.append(" has");
            first = false;
        } else {
            result.append(" and");
        }
        result.append(" a " + pet.eClass().getName());
        result.append(" " + pet.getName() + "(" + pet.getWeight() + "kg)");
    }
    return result.toString();
}
```

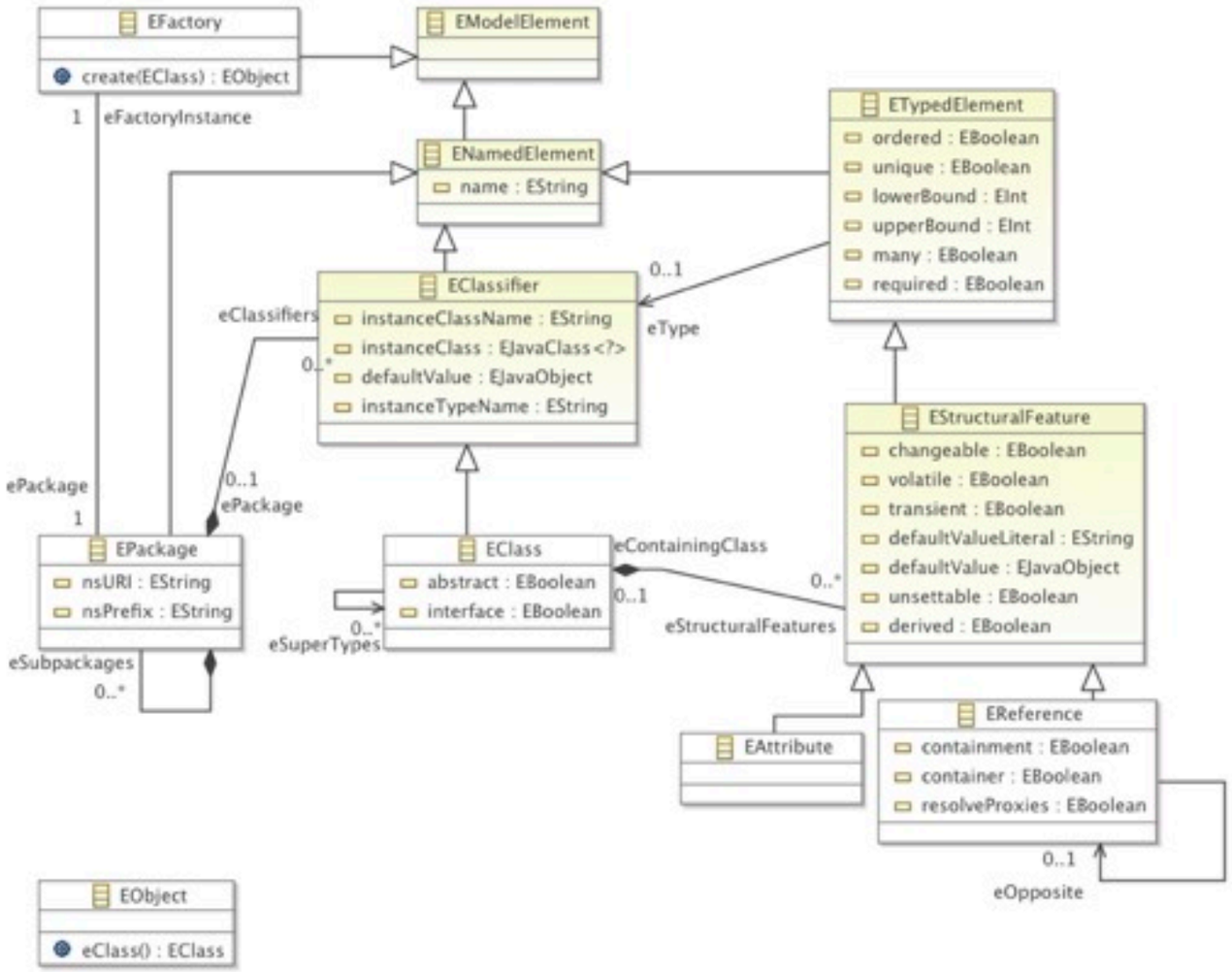
```
Markus has a Dog Fido(20kg)
```

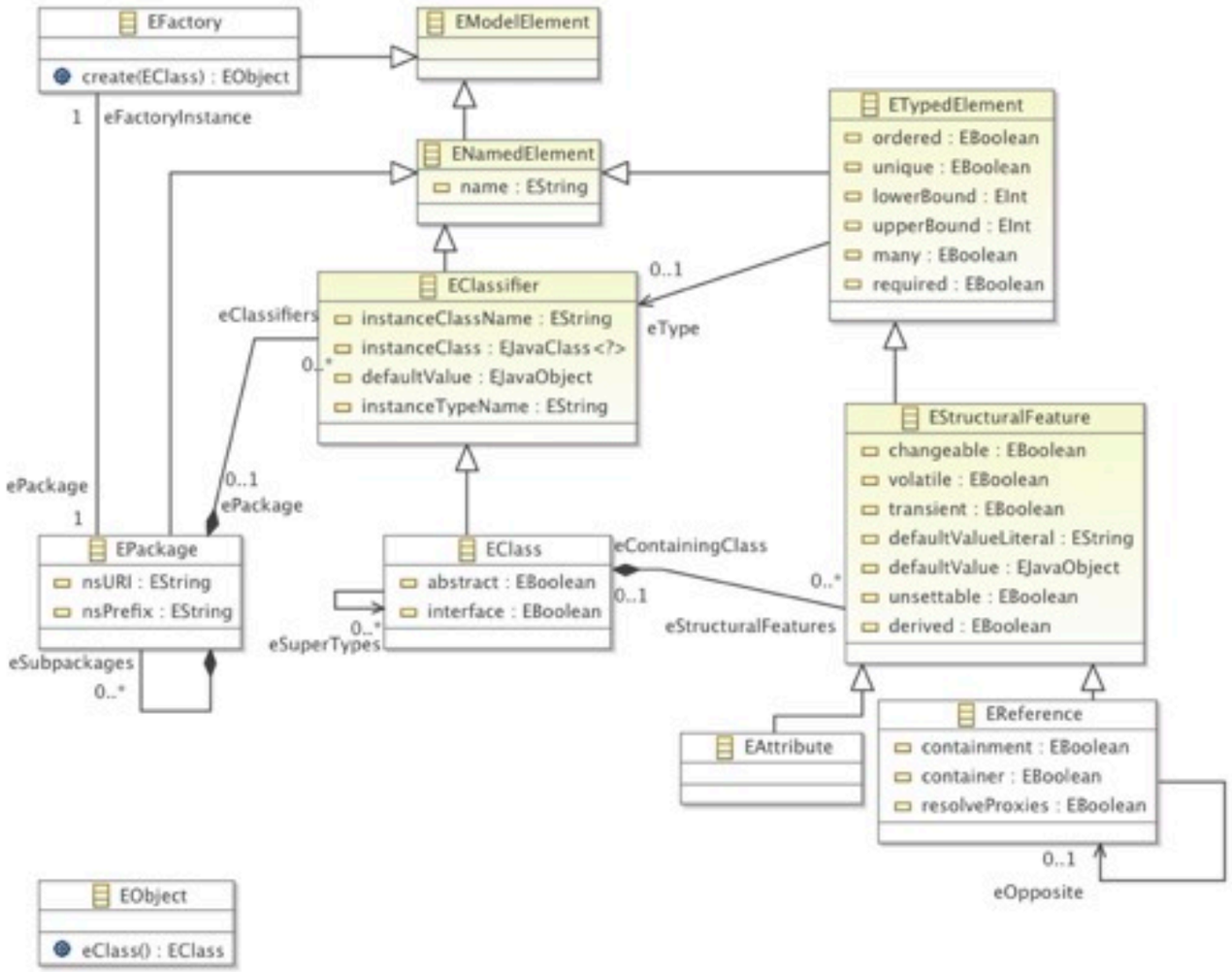
This plain Java Mapping is Only the Basis for ...

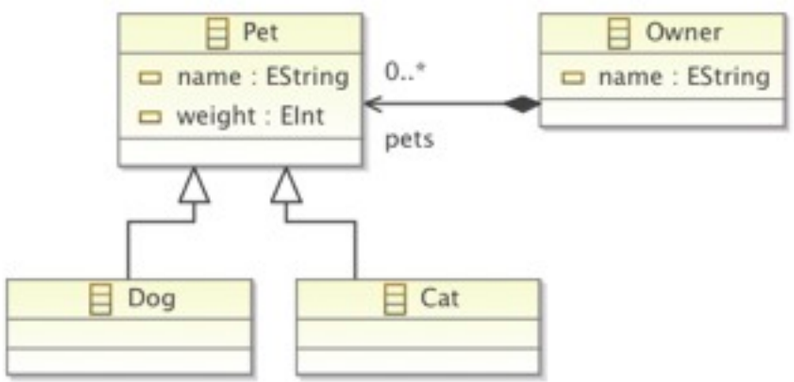
- EMF
  - Serialization
  - Notifications
  - Edit and editor
  - Validation
- other frameworks for ...
  - textual and graphical notations
  - code generation
  - model transformations
  - ...

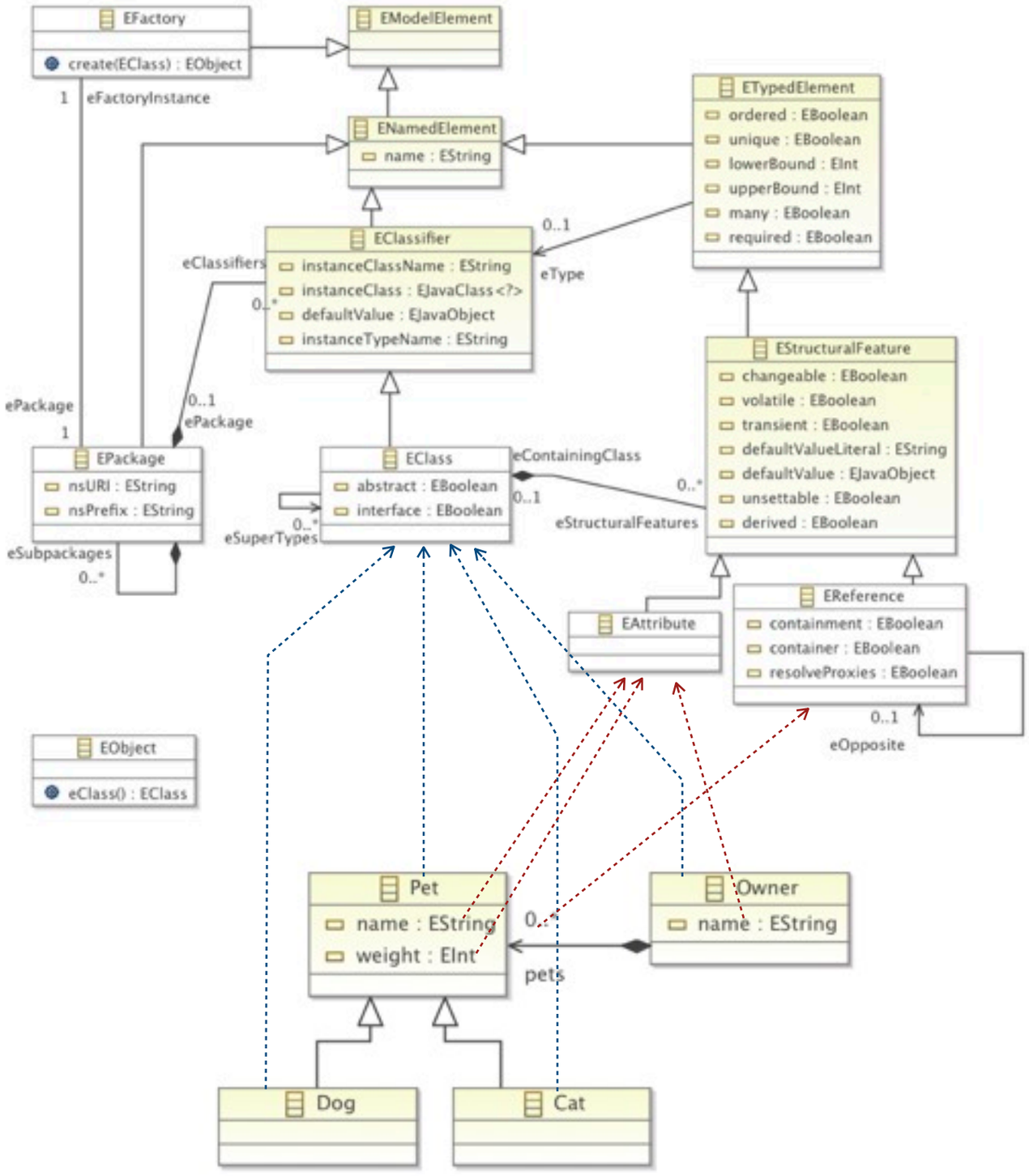
The Ecore-to-Java Mapping(s) of EMF in the Context of Multi-Layer Meta-Modeling

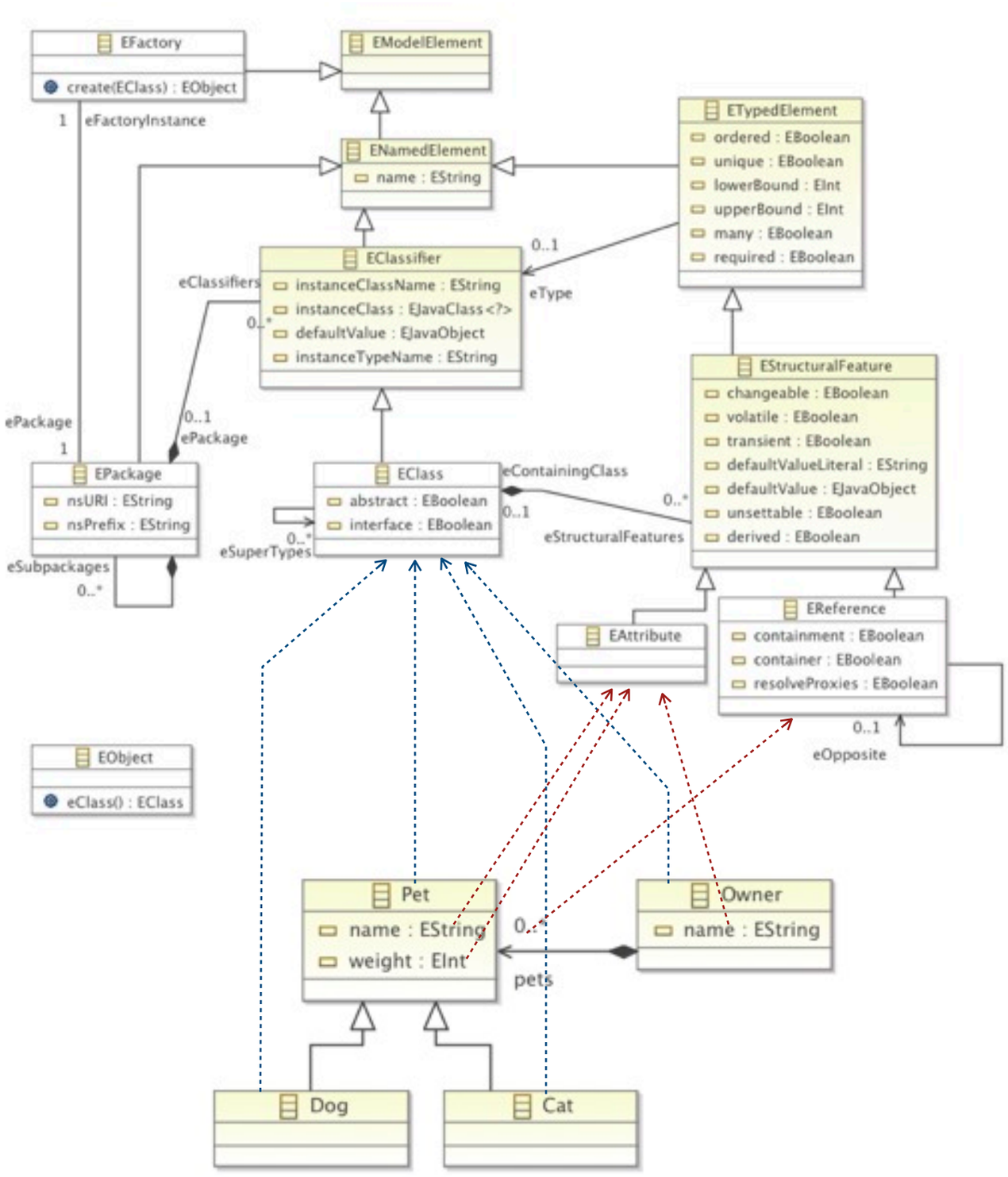

EFactory
create(EClass) : EObject
EModelElement
1
eFactoryInstance
ENamedElement
name : EString
ETypedElement
ordered : EBoolean
unique : EBoolean
lowerBound : EInt
upperBound : EInt
many : EBoolean
required : EBoolean
0..1
eClassifiers
EClassifier
instanceClassName : EString
instanceClass : EJavaClass<?>
defaultValue : EJavaObject
instanceTypeName : EString
0..*
eType
EStructuralFeature
changeable : EBoolean
volatile : EBoolean
transient : EBoolean
defaultValueLiteral : EString
defaultValue : EJavaObject
unsettable : EBoolean
derived : EBoolean
ePackage
1
0..1
ePackage
EPackage
nsURI : EString
nsPrefix : EString
eSubpackages
0..*
EClass
abstract : EBoolean
interface : EBoolean
0..*
eSuperTypes
eContainingClass
0..1
0..*
eStructuralFeatures
EAttribute
EReference
containment : EBoolean
container : EBoolean
resolveProxies : EBoolean
0..1
eOpposite
EObject
eClass() : EClass


EFactory
create(EClass) : EObject
1
eFactoryInstance
EModelElement
ENamedElement
name : EString
ETypedElement
ordered : EBoolean
unique : EBoolean
lowerBound : EInt
upperBound : EInt
many : EBoolean
required : EBoolean
0..1
eType
EClassifier
eClassifiers
instanceClassName : EString
instanceClass : EJavaClass<?>
defaultValue : EJavaObject
instanceTypeName : EString
0..*
EStructuralFeature
changeable : EBoolean
volatile : EBoolean
transient : EBoolean
defaultValueLiteral : EString
defaultValue : EJavaObject
unsettable : EBoolean
derived : EBoolean
ePackage
1
0..1
ePackage
EPackage
nsURI : EString
nsPrefix : EString
eSubpackages
0..*
EClass
abstract : EBoolean
interface : EBoolean
0..*
eSuperTypes
eContainingClass
0..1
0..*
eStructuralFeatures
EAttribute
EReference
containment : EBoolean
container : EBoolean
resolveProxies : EBoolean
0..1
eOpposite
EObject
eClass() : EClass

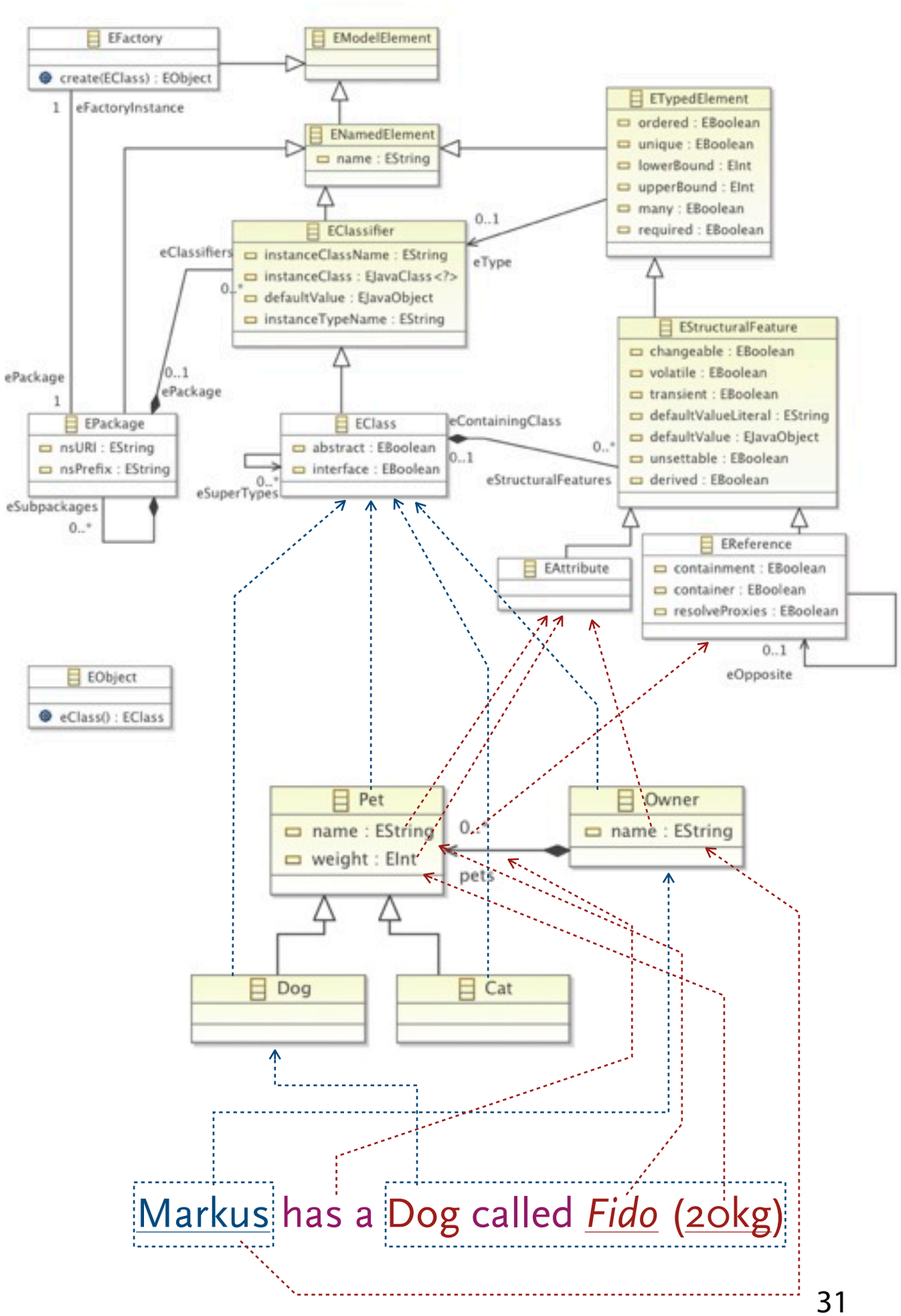

EFactory
create(EClass) : EObject
1
eFactoryInstance
EModelElement
ENamedElement
name : EString
ETypedElement
ordered : EBoolean
unique : EBoolean
lowerBound : EInt
upperBound : EInt
many : EBoolean
required : EBoolean
0..1
eType
EClassifier
eClassifiers
instanceClassName : EString
instanceClass : EJavaClass<?>
defaultValue : EJavaObject
instanceTypeName : EString
0..*
EStructuralFeature
changeable : EBoolean
volatile : EBoolean
transient : EBoolean
defaultValueLiteral : EString
defaultValue : EJavaObject
unsettable : EBoolean
derived : EBoolean
ePackage
1
0..1
ePackage
EPackage
nsURI : EString
nsPrefix : EString
eSubpackages
0..*
EClass
abstract : EBoolean
interface : EBoolean
0..*
eSuperTypes
eContainingClass
0..1
0..*
eStructuralFeatures
EAttribute
EReference
containment : EBoolean
container : EBoolean
resolveProxies : EBoolean
0..1
eOpposite
EObject
eClass() : EClass

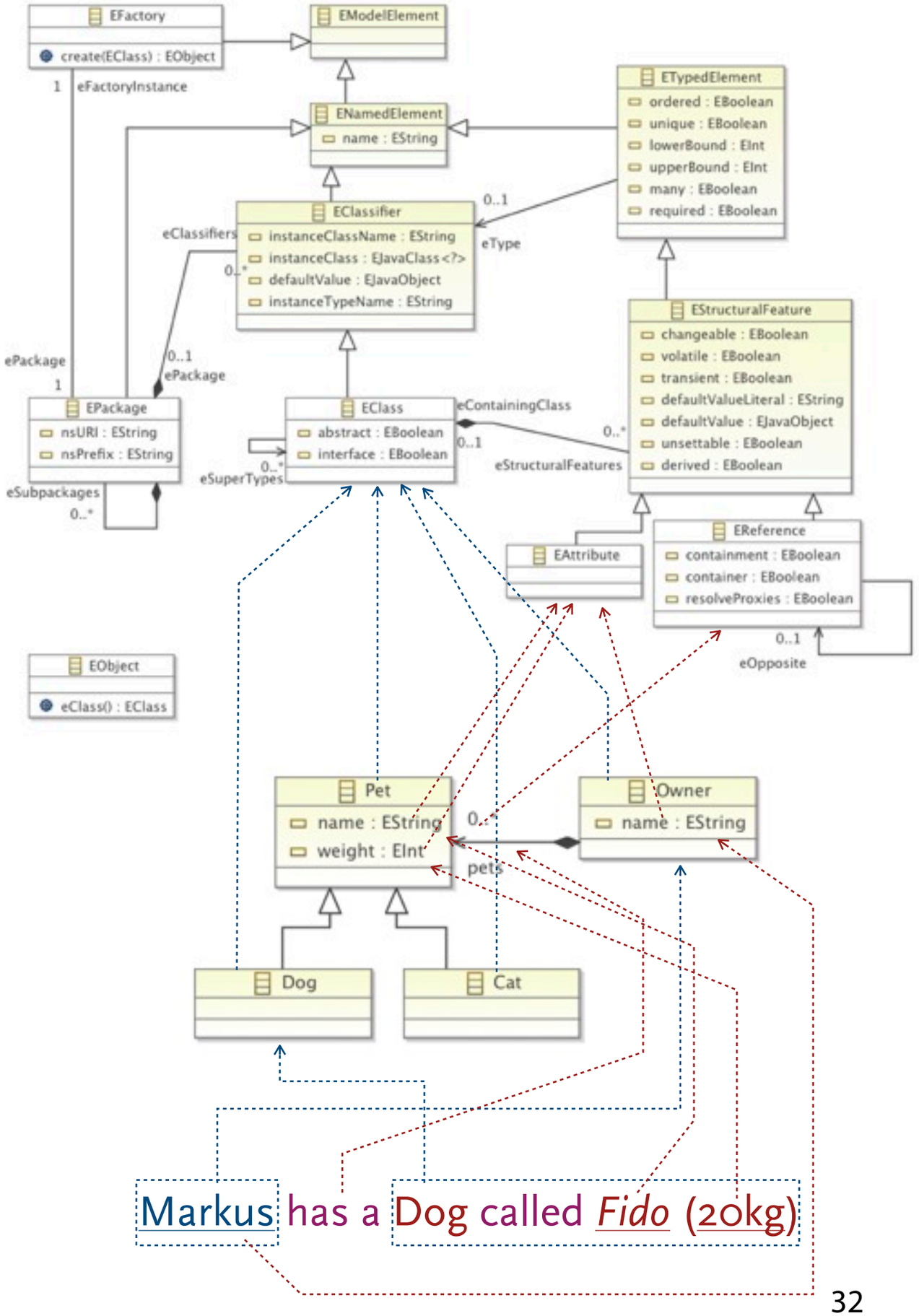

Pet
name : EString
weight : EInt
0..*
pets
Owner
name : EString
Dog
Cat

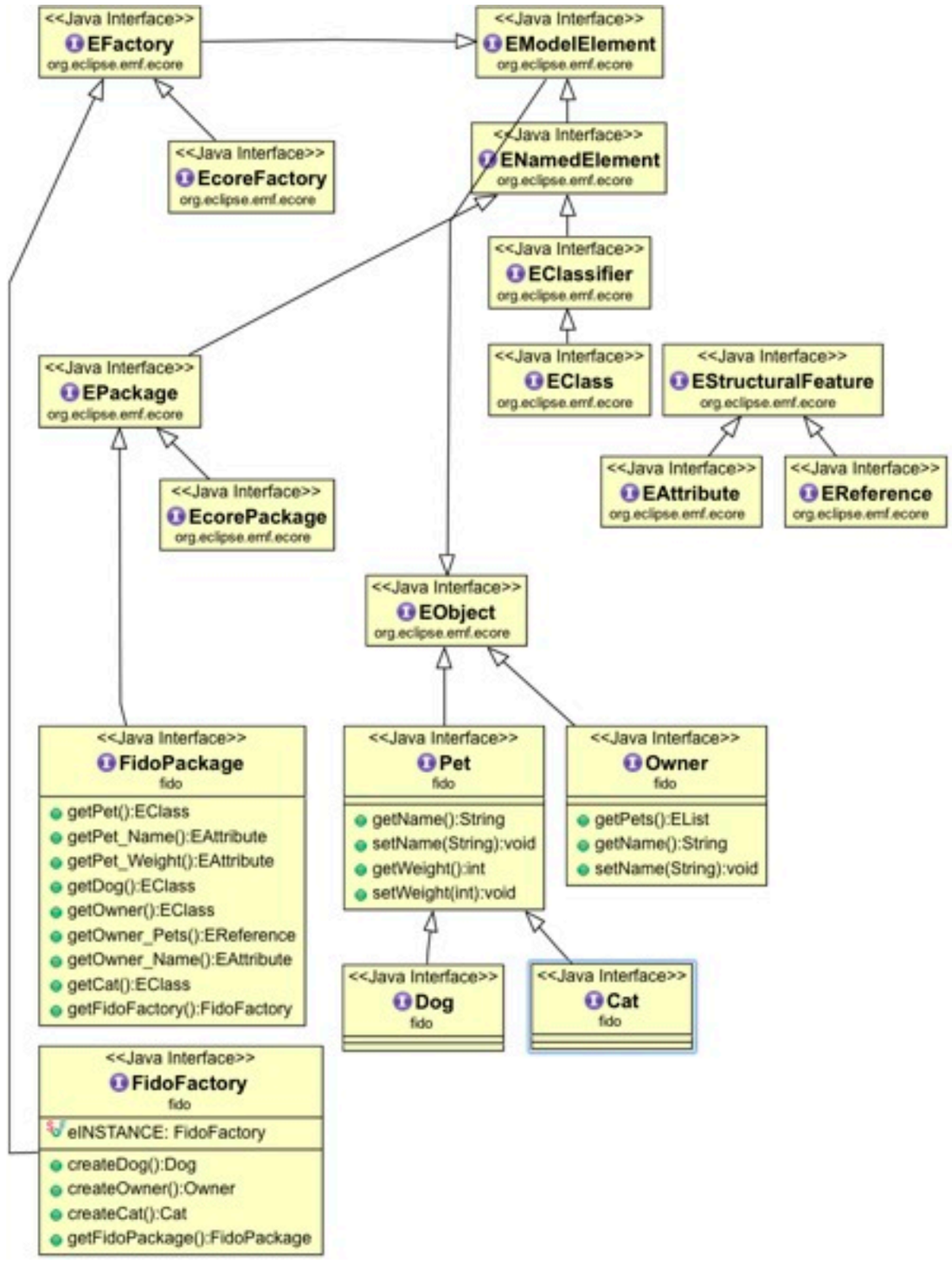

EFactory
create(EClass) : EObject
1
eFactoryInstance
EModelElement
ENamedElement
name : EString
ETypedElement
ordered : EBoolean
unique : EBoolean
lowerBound : EInt
upperBound : EInt
many : EBoolean
required : EBoolean
0..1
eType
EClassifier
eClassifiers
instanceClassName : EString
instanceClass : EJavaClass<?>
defaultValue : EJavaObject
instanceTypeName : EString
0..*
EStructuralFeature
changeable : EBoolean
volatile : EBoolean
transient : EBoolean
defaultValueLiteral : EString
defaultValue : EJavaObject
unsettable : EBoolean
derived : EBoolean
ePackage
1
0..1
ePackage
EPackage
nsURI : EString
nsPrefix : EString
eSubpackages
0..*
EClass
abstract : EBoolean
interface : EBoolean
0..*
eSuperTypes
eContainingClass
0..1
0..*
eStructuralFeatures
EAttribute
EReference
containment : EBoolean
container : EBoolean
resolveProxies : EBoolean
0..1
eOpposite
EObject
eClass() : EClass
Pet
name : EString
weight : EInt
0..*
pets
Owner
name : EString
Dog
Cat

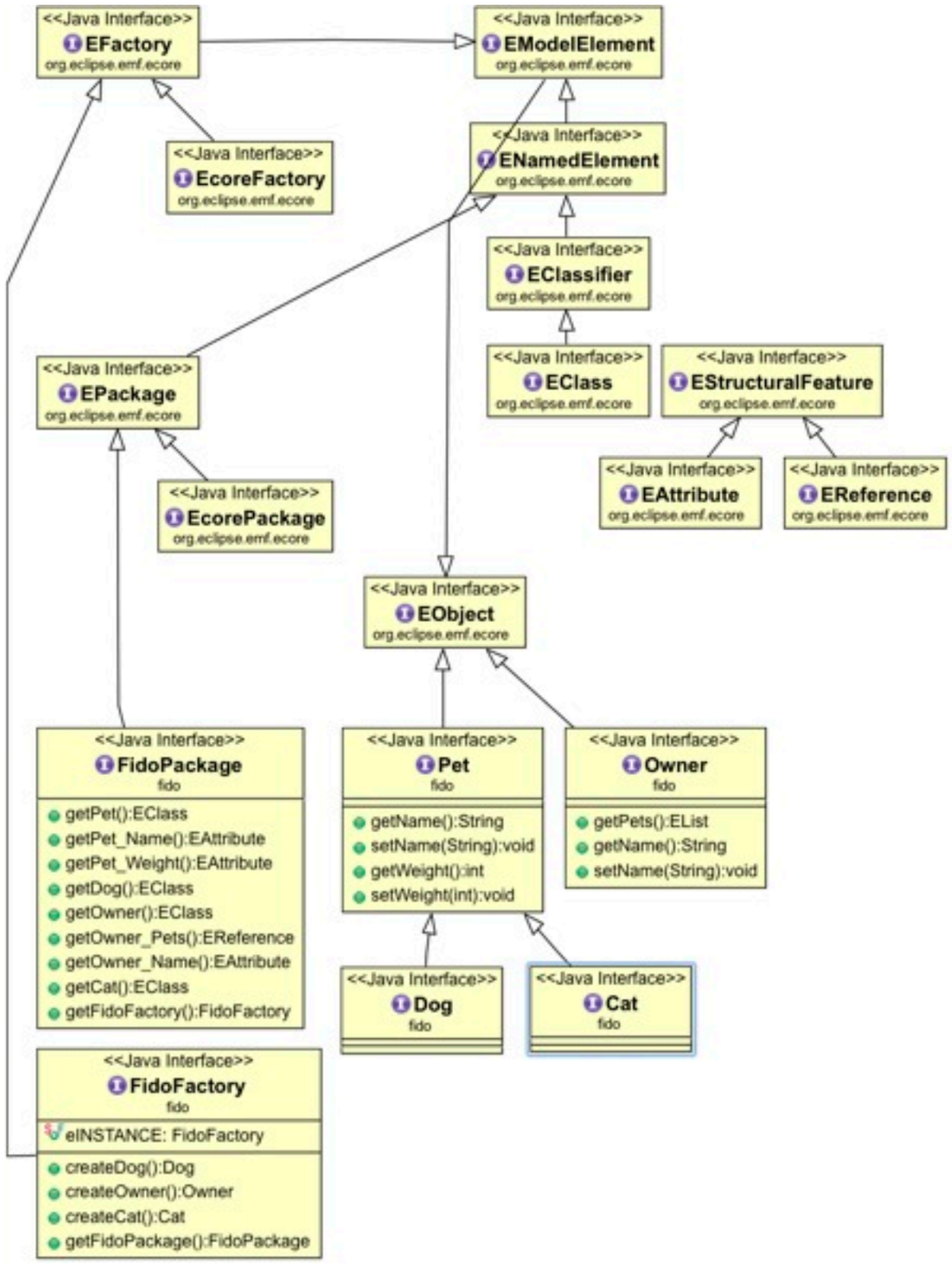


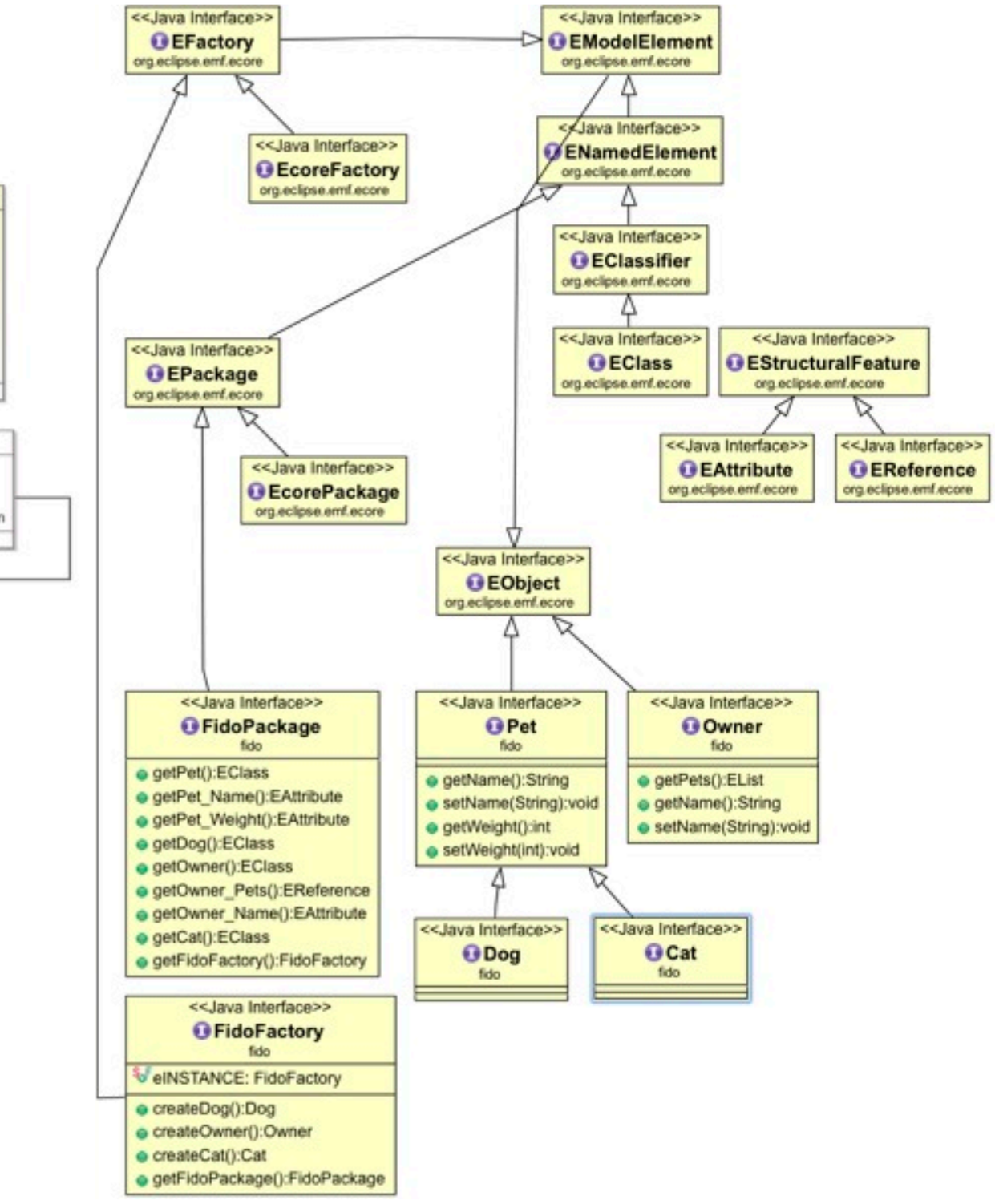
Markus has a Dog called *Fido* (20kg)

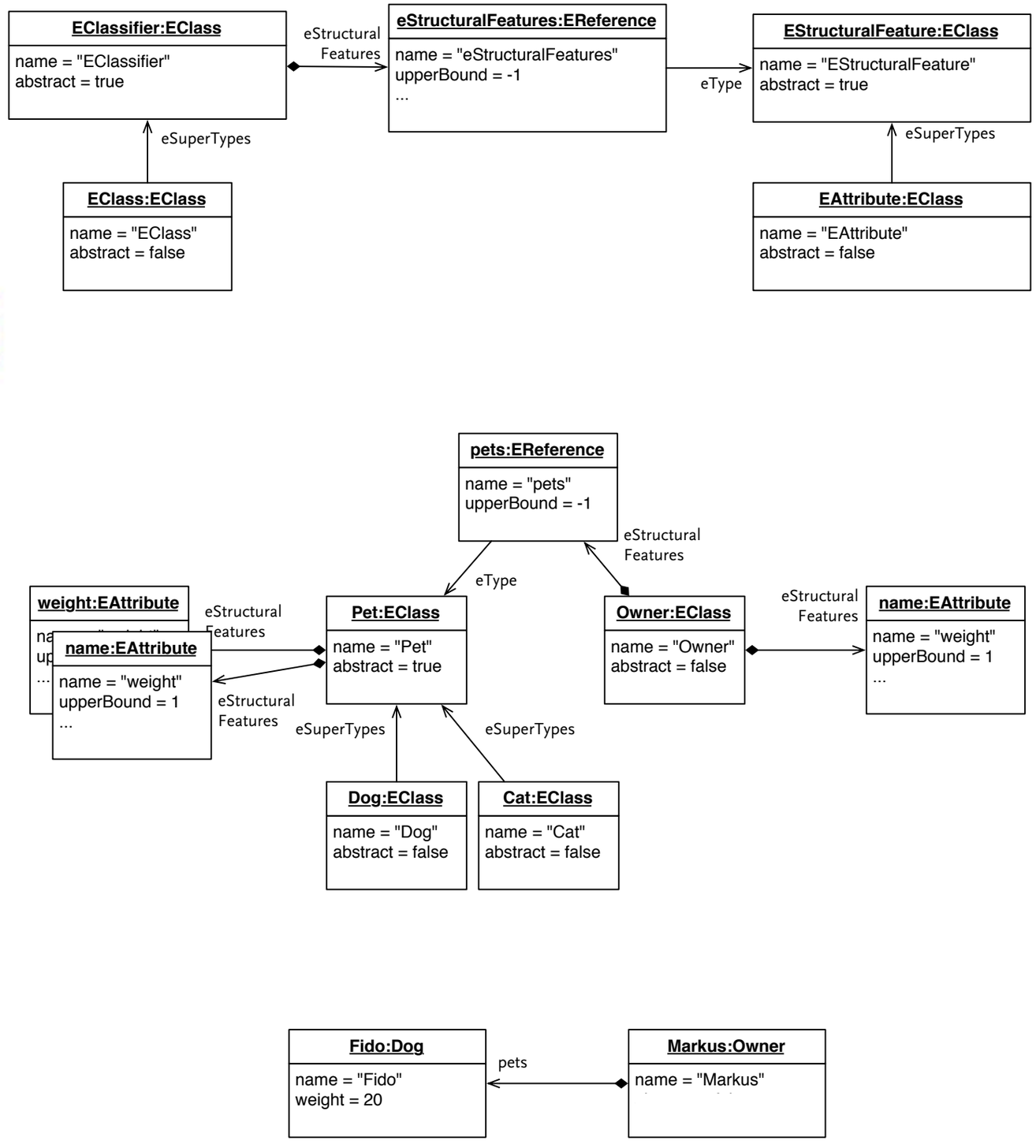

EFactory
create(EClass) : EObject
1
eFactoryInstance
EModelElement
ENamedElement
name : EString
ETypedElement
ordered : EBoolean
unique : EBoolean
lowerBound : EInt
upperBound : EInt
many : EBoolean
required : EBoolean
0..1
eType
EClassifier
eClassifiers
instanceClassName : EString
instanceClass : EJavaClass<?>
defaultValue : EJavaObject
instanceTypeName : EString
0..*
EStructuralFeature
changeable : EBoolean
volatile : EBoolean
transient : EBoolean
defaultValueLiteral : EString
defaultValue : EJavaObject
unsettable : EBoolean
derived : EBoolean
ePackage
1
0..1
ePackage
EPackage
nsURI : EString
nsPrefix : EString
eSubpackages
0..*
EClass
abstract : EBoolean
interface : EBoolean
0..*
eSuperTypes
eContainingClass
0..1
0..*
eStructuralFeatures
EAttribute
EReference
containment : EBoolean
container : EBoolean
resolveProxies : EBoolean
0..1
eOpposite
EObject
eClass() : EClass
Pet
name : EString
weight : EInt
0..*
pets
Owner
name : EString
Dog
Cat
Markus has a Dog called Fido (20kg)

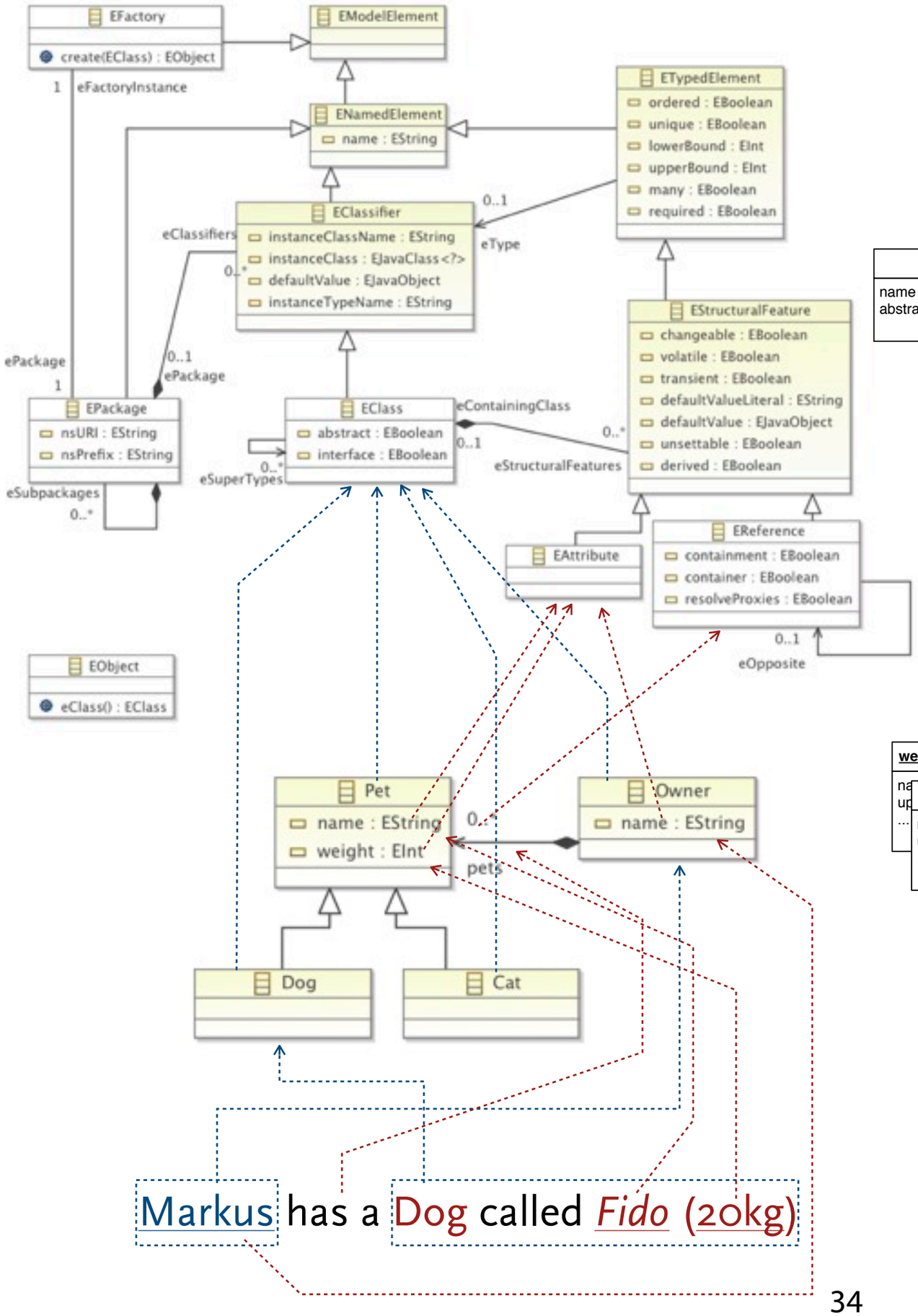

EFactory
create(EClass) : EObject
1
eFactoryInstance
EModelElement
ENamedElement
name : EString
ETypedElement
ordered : EBoolean
unique : EBoolean
lowerBound : EInt
upperBound : EInt
many : EBoolean
required : EBoolean
0..1
eType
EClassifier
eClassifiers
instanceClassName : EString
instanceClass : EJavaClass<?>
defaultValue : EJavaObject
instanceTypeName : EString
0..*
ePackage
1
0..1
ePackage
EStructuralFeature
changeable : EBoolean
volatile : EBoolean
transient : EBoolean
defaultValueLiteral : EString
defaultValue : EJavaObject
unsettable : EBoolean
derived : EBoolean
EPackage
nsURI : EString
nsPrefix : EString
eSubpackages
0..*
EClass
abstract : EBoolean
interface : EBoolean
0..*
eSuperTypes
eContainingClass
0..1
0..*
eStructuralFeatures
EAttribute
EReference
containment : EBoolean
container : EBoolean
resolveProxies : EBoolean
0..1
eOpposite
EObject
eClass() : EClass
Pet
name : EString
weight : EInt
0..*
pets
Owner
name : EString
Dog
Cat
Markus has a Dog called Fido (20kg)


<<Java Interface>>
EFactory
org.eclipse.emf.ecore
<<Java Interface>>
EModelElement
org.eclipse.emf.ecore
<<Java Interface>>
EcoreFactory
org.eclipse.emf.ecore
<<Java Interface>>
ENamedElement
org.eclipse.emf.ecore
<<Java Interface>>
EClassifier
org.eclipse.emf.ecore
<<Java Interface>>
EPackage
org.eclipse.emf.ecore
<<Java Interface>>
EClass
org.eclipse.emf.ecore
<<Java Interface>>
EStructuralFeature
org.eclipse.emf.ecore
<<Java Interface>>
EAttribute
org.eclipse.emf.ecore
<<Java Interface>>
EReference
org.eclipse.emf.ecore
<<Java Interface>>
EcorePackage
org.eclipse.emf.ecore
<<Java Interface>>
EObject
org.eclipse.emf.ecore
<<Java Interface>>
FidoPackage
fido
getPet():EClass
getPet_Name():EAttribute
getPet_Weight():EAttribute
getDog():EClass
getOwner():EClass
getOwner_Pets():EReference
getOwner_Name():EAttribute
getCat():EClass
getFidoFactory():FidoFactory
<<Java Interface>>
Pet
fido
getName():String
setName(String):void
getWeight():int
setWeight(int):void
<<Java Interface>>
Owner
fido
getPets():EList
getName():String
setName(String):void
<<Java Interface>>
Dog
fido
<<Java Interface>>
Cat
fido
<<Java Interface>>
FidoFactory
fido
eINSTANCE: FidoFactory
createDog():Dog
createOwner():Owner
createCat():Cat
getFidoPackage():FidoPackage


<<Java Interface>>
EFactory
org.eclipse.emf.ecore
<<Java Interface>>
EModelElement
org.eclipse.emf.ecore
<<Java Interface>>
EcoreFactory
org.eclipse.emf.ecore
<<Java Interface>>
ENamedElement
org.eclipse.emf.ecore
<<Java Interface>>
EClassifier
org.eclipse.emf.ecore
<<Java Interface>>
EPackage
org.eclipse.emf.ecore
<<Java Interface>>
EClass
org.eclipse.emf.ecore
<<Java Interface>>
EStructuralFeature
org.eclipse.emf.ecore
<<Java Interface>>
EAttribute
org.eclipse.emf.ecore
<<Java Interface>>
EReference
org.eclipse.emf.ecore
<<Java Interface>>
EcorePackage
org.eclipse.emf.ecore
<<Java Interface>>
EObject
org.eclipse.emf.ecore
<<Java Interface>>
FidoPackage
fido
getPet():EClass
getPet_Name():EAttribute
getPet_Weight():EAttribute
getDog():EClass
getOwner():EClass
getOwner_Pets():EReference
getOwner_Name():EAttribute
getCat():EClass
getFidoFactory():FidoFactory
<<Java Interface>>
Pet
fido
getName():String
setName(String):void
getWeight():int
setWeight(int):void
<<Java Interface>>
Owner
fido
getPets():EList
getName():String
setName(String):void
<<Java Interface>>
Dog
fido
<<Java Interface>>
Cat
fido
<<Java Interface>>
FidoFactory
fido
eINSTANCE: FidoFactory
createDog():Dog
createOwner():Owner
createCat():Cat
getFidoPackage():FidoPackage

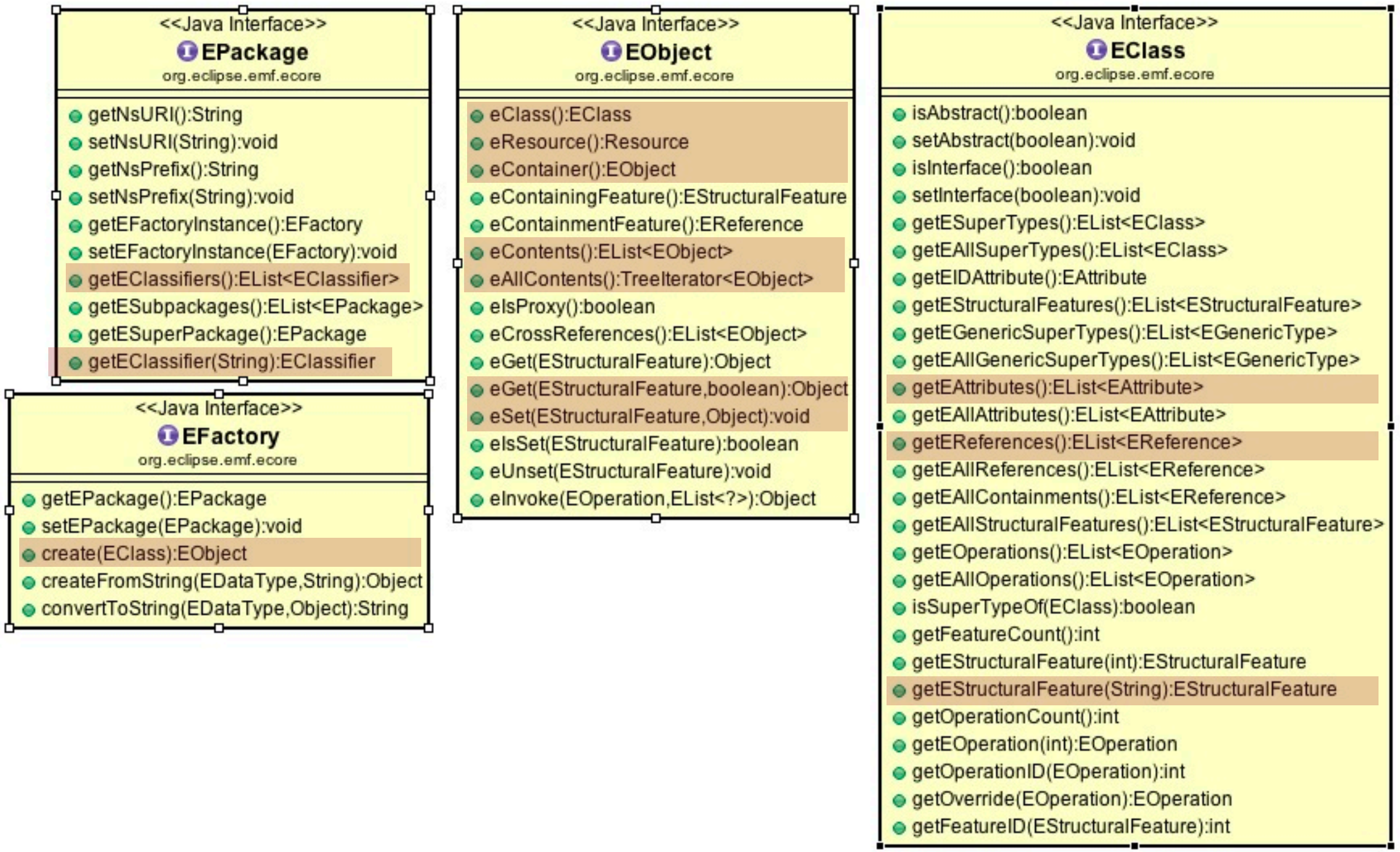

<<Java Interface>>
EFactory
org.eclipse.emf.ecore
<<Java Interface>>
EModelElement
org.eclipse.emf.ecore
<<Java Interface>>
EcoreFactory
org.eclipse.emf.ecore
<<Java Interface>>
ENamedElement
org.eclipse.emf.ecore
<<Java Interface>>
EClassifier
org.eclipse.emf.ecore
<<Java Interface>>
EClass
org.eclipse.emf.ecore
<<Java Interface>>
EStructuralFeature
org.eclipse.emf.ecore
<<Java Interface>>
EPackage
org.eclipse.emf.ecore
<<Java Interface>>
EAttribute
org.eclipse.emf.ecore
<<Java Interface>>
EReference
org.eclipse.emf.ecore
<<Java Interface>>
EcorePackage
org.eclipse.emf.ecore
<<Java Interface>>
EObject
org.eclipse.emf.ecore
<<Java Interface>>
FidoPackage
fido
getPet():EClass
getPet_Name():EAttribute
getPet_Weight():EAttribute
getDog():EClass
getOwner():EClass
getOwner_Pets():EReference
getOwner_Name():EAttribute
getCat():EClass
getFidoFactory():FidoFactory
<<Java Interface>>
Pet
fido
getName():String
setName(String):void
getWeight():int
setWeight(int):void
<<Java Interface>>
Owner
fido
getPets():EList
getName():String
setName(String):void
<<Java Interface>>
Dog
fido
<<Java Interface>>
Cat
fido
<<Java Interface>>
FidoFactory
fido
eINSTANCE: FidoFactory
createDog():Dog
createOwner():Owner
createCat():Cat
getFidoPackage():FidoPackage


EClassifier:EClass
name = "EClassifier"
abstract = true
eStructural Features
eStructuralFeatures:EReference
name = "eStructuralFeatures"
upperBound = -1
...
eType
EStructuralFeature:EClass
name = "EStructuralFeature"
abstract = true
eSuperTypes
eSuperTypes
EClass:EClass
name = "EClass"
abstract = false
EAttribute:EClass
name = "EAttribute"
abstract = false
pets:EReference
name = "pets"
upperBound = -1
eStructural Features
eType
weight:EAttribute
name:EAttribute
name = "weight"
upperBound = 1
...
eStructural Features
eStructural Features
Pet:EClass
name = "Pet"
abstract = true
Owner:EClass
name = "Owner"
abstract = false
eStructural Features
name:EAttribute
name = "weight"
upperBound = 1
...
eSuperTypes
eSuperTypes
Dog:EClass
name = "Dog"
abstract = false
Cat:EClass
name = "Cat"
abstract = false
Fido:Dog
name = "Fido"
weight = 20
pets
Markus:Owner
name = "Markus"


EFactory
create(EClass) : EObject
1
eFactoryInstance
EModelElement
ENamedElement
name : EString
ETypedElement
ordered : EBoolean
unique : EBoolean
lowerBound : EInt
upperBound : EInt
many : EBoolean
required : EBoolean
0..1
eType
EClassifier
instanceClassName : EString
instanceClass : EJavaClass<?>
defaultValue : EJavaObject
instanceTypeName : EString
eClassifiers
0..*
ePackage
1
0..1
ePackage
EStructuralFeature
changeable : EBoolean
volatile : EBoolean
transient : EBoolean
defaultValueLiteral : EString
defaultValue : EJavaObject
unsettable : EBoolean
derived : EBoolean
EPackage
nsURI : EString
nsPrefix : EString
eSubpackages
0..*
EClass
abstract : EBoolean
interface : EBoolean
0..*
eSuperTypes
eContainingClass
0..1
0..*
eStructuralFeatures
EAttribute
EReference
containment : EBoolean
container : EBoolean
resolveProxies : EBoolean
0..1
eOpposite
EObject
eClass() : EClass
Pet
name : EString
weight : EInt
0..*
pets
Owner
name : EString
Dog
Cat
Markus has a Dog called Fido (20kg)
EClassifier:EClass
name = "EClassifier"
abstract = true
eStructural Features
eSuperTypes
EClass:EClass
name = "EClass"
abstract = false
eStructuralFeatures:EReference
name = "eStructuralFeatures"
upperBound = -1
...
eType
EStructuralFeature:EClass
name = "EStructuralFeature"
abstract = true
eSuperTypes
EAttribute:EClass
name = "EAttribute"
abstract = false
pets:EReference
name = "pets"
upperBound = -1
eStructural Features
eType
eStructural Features
weight:EAttribute
name:EAttribute
name = "weight"
upperBound = 1
...
eStructural Features
eStructural Features
Pet:EClass
name = "Pet"
abstract = true
Owner:EClass
name = "Owner"
abstract = false
eStructural Features
name:EAttribute
name = "weight"
upperBound = 1
...
eSuperTypes
eSuperTypes
Dog:EClass
name = "Dog"
abstract = false
Cat:EClass
name = "Cat"
abstract = false
Fido:Dog
name = "Fido"
weight = 20
pets
Markus:Owner
name = "Markus"

Reflective Java Interface to EMF

Reflective Interface



Example Usage of the Reflective Interface

Example Usage of the Reflective Interface

```
public class TestFidoRefective {

    public static void main(String[] args) {

        EPackage fidoPackage = FidoPackage.eINSTANCE;
        EFactory factory = fidoPackage.getEFactoryInstance();

        EClass ownerClass = (EClass)fidoPackage.getEClassifier("Owner");
        EObject markus = factory.create(ownerClass);
        markus.eSet(ownerClass.getEStructuralFeature("name"), "markus");

        EClass dogClass = (EClass)fidoPackage.getEClassifier("Dog");
        EObject fido = factory.create(dogClass);
        fido.eSet(dogClass.getEStructuralFeature("name"), "fido");
        fido.eSet(dogClass.getEStructuralFeature("weight"), 20);

        @SuppressWarnings("unchecked")
        EList<EObject> pets = (EList<EObject>)markus.eGet(
            ownerClass.getEStructuralFeature("pets"));
        pets.add(fido);

        System.out.println(markus.eGet(
            ownerClass.getEStructuralFeature("description")));
    }
}
```

Example Usage of the Reflective Interface

```
public class TestFidoRefective {

    public static void main(String[] args) {

        EPackage fidoPackage = FidoPackage.eINSTANCE;
        EFactory factory = fidoPackage.getEFactoryInstance();

        EClass ownerClass = (EClass)fidoPackage.getEClassifier("Owner");
        EObject markus = factory.create(ownerClass);
        markus.eSet(ownerClass.getEStructuralFeature("name"), "markus");

        EClass dogClass = (EClass)fidoPackage.getEClassifier("Dog");
        EObject fido = factory.create(dogClass);
        fido.eSet(dogClass.getEStructuralFeature("name"), "fido");
        fido.eSet(dogClass.getEStructuralFeature("weight"), 20);

        @SuppressWarnings("unchecked")
        EList<EObject> pets = (EList<EObject>)markus.eGet(
            ownerClass.getEStructuralFeature("pets"));
        pets.add(fido);

        System.out.println(markus.eGet(
            ownerClass.getEStructuralFeature("description")));
    }
}
```

```
Markus has a Dog Fido(20kg)
```

Example Usage of the Reflective Interface

```
public static String reflectiveToString(EObject obj, String indent, boolean includeFeatures) {
    StringBuffer result = new StringBuffer();
    EClass eClass = obj.eClass();
    result.append(indent);
    EStructuralFeature nameFeature = eClass.getEStructuralFeature("name");
    if (nameFeature != null && nameFeature.getEType() == EcorePackage.eINSTANCE.getEString()) {
        result.append(obj.eGet(nameFeature));
    }
    result.append(":" + eClass.getName());
    if (includeFeatures) {
        result.append(" {\n");
        for (EStructuralFeature feature: eClass.getEAllStructuralFeatures()) {
            result.append(indent + "  " + feature.getName() + "=");
            if (feature instanceof EAttribute) {
                result.append(obj.eGet(feature) + "\n");
            } else {
                EReference eReference = (EReference)feature;
                if (eReference.isMany()) {
                    result.append("[\n");
                    for(EObject child: (EList<EObject>)obj.eGet(feature)) {
                        result.append(reflectiveToString(
                                child, indent + "    ", eReference.isContainment()));
                    }
                    result.append(indent + "  ]\n");
                } else {
                    result.append(reflectiveToString((EObject)obj.eGet(feature),
                            indent + "  ", eReference.isContainment()));
                }
            }
        }
        result.append(indent + "}\n");
    }
    return result.toString();
}
```

```
public static String reflectiveToString(EObject obj, String indent, boolean includeFeatures) {
    StringBuffer result = new StringBuffer();
    EClass eClass = obj.eClass();
    result.append(indent);
    EStructuralFeature nameFeature = eClass.getEStructuralFeature("name");
    if (nameFeature != null && nameFeature.getEType() == EcorePackage.eINSTANCE.getEString()) {
        result.append(obj.eGet(nameFeature));
    }
    result.append(":" + eClass.getName());
    if (includeFeatures) {
        result.append(" {\n");
        for (EStructuralFeature feature: eClass.getEAllStructuralFeatures()) {
            result.append(indent + " " + feature.getName() + "=");
            if (feature instanceof EAttribute) {
                result.append(obj.eGet(feature) + "\n");
            } else {
                EReference eReference = (EReference)feature;
                if (eReference.isMany()
                    result.append("[\n
                    for(EObject child:
                        result.append(
                                child
                    }
                    result.append(inde
                } else {
                    result.append(refl
                            indent + "
                }
            }
        }
        result.append(indent + "}\n");
    }
    return result.toString();
}
```

```
markus:Owner {
 pets=[
   fido:Dog {
    name=fido
    weight=20
   }
 ]
 name=markus
 description=markus has a Dog fido(20kg)
}
```

Pure Reflection

Pure Reflection

```
public static void main(String[] args) {
    Resource.Factory.Registry.INSTANCE.getExtensionToFactoryMap()
            .put("fido", new XMIResourceFactoryImpl());
    Resource.Factory.Registry.INSTANCE.getExtensionToFactoryMap()
            .put("ecore", new XMIResourceFactoryImpl());

    ResourceSet rs = new ResourceSetImpl();
    Resource metaModel = rs.getResource(
        URI.createFileURI("model/fido.ecore"), true);
    EPackage metaPackage = (EPackage)metaModel.getContents().get(0);
    EPackage.Registry.INSTANCE.put(metaPackage.getNsURI(), metaPackage);

    Resource resource = rs.getResource(
        URI.createFileURI("model/example.fido"), true);
    EObject contents = resource.getContents().get(0);
    System.out.println(reflectiveToString(contents, "", true));
}
```

Pure Reflection

```
public static void main(String[] args) {
    Resource.Factory.Registry.INSTANCE.getExtensionToFactoryMap()
            .put("fido", new XMIResourceFactoryImpl());
    Resource.Factory.Registry.INSTANCE.getExtensionToFactoryMap()
            .put("ecore", new XMIResourceFactoryImpl());

    ResourceSet rs = new ResourceSetImpl();
    Resource metaModel = rs.getResource(
        URI.createFileURI("model/fido.ecore"), true);
    EPackage metaPackage = (EPa
    EPackage.Registry.INSTANCE.
    Resource resource = rs.getR
        URI.createFileURI("model
    EObject contents = resource
    System.out.println(reflecti
}
```

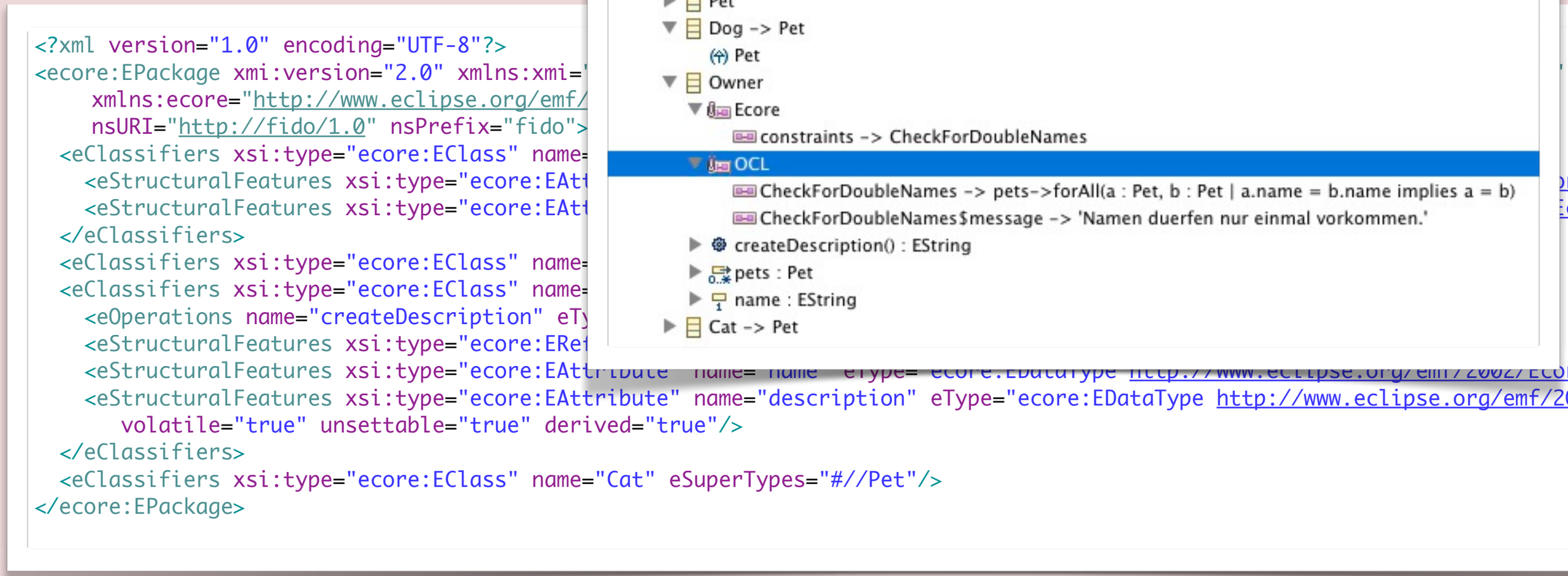
```
markus:Owner {
 pets=[
   fido:Dog {
    name=fido
    weight=20
   }
 ]
 name=markus
 description=markus has a Dog fido(20kg)
}
```

Basic File Persistence:

Resource, ResourceSet, URI, XMI, XML, ...

Composition

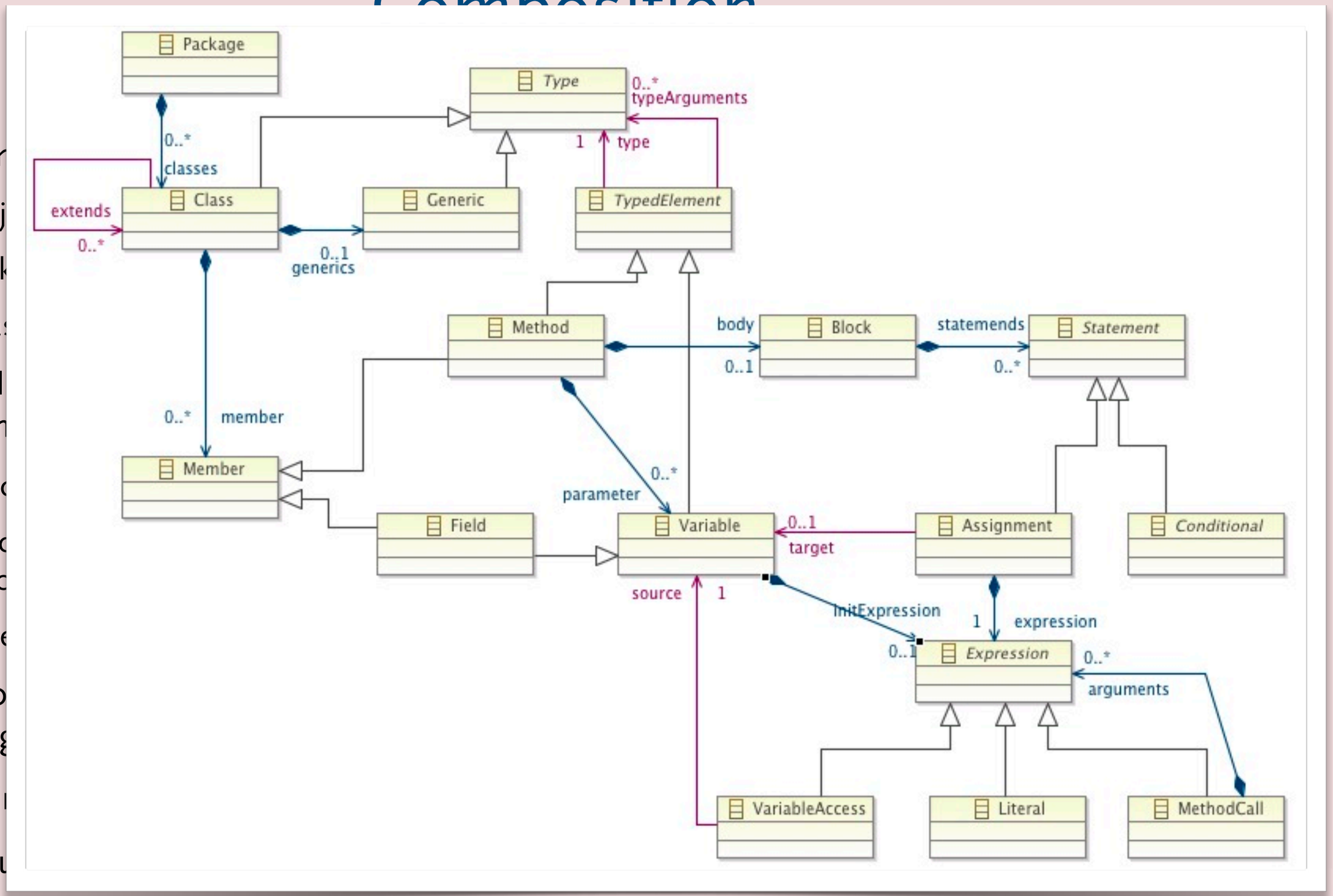
- EMF models are directed labeled graphs
  - objects are nodes
  - links (instances of references) are edges
  - classifier for objects and links are labels for nodes and edges
- An EMF model graph contains a subgraph that is a tree (i.e. an acyclic directed graph) that is span by links of composite references (EReference with containment=true).
  - Such trees are called *containment-trees* or *containment-hierarchy*
  - Each model element as a container (eContainer:EObject) and contents (eContents:EObject[]), accessible via EMF's reflection interface
  - The root of such a tree is a root element (eContainer==null)
- In a good EMF model, the containment-tree is a spanning tree (a tree that covers all nodes in the graph/model). But EMF allows multiple trees and root elements anyway.
- Containment-trees
  - ... used in XML/XMI serialization, tree-editors and outline views.
  - ... drive code-generation and model-transformations. They make that models have a start, and a natural iteration order.



- In a good EMF model, the containment-tree is a spanning tree (a tree that covers all nodes in the graph/model). But EMF allows multiple trees and root elements anyway.
- Containment-trees
  - ... used in XML/XMI serialization, tree-editors and outline views.
  - ... drive code-generation and model-transformations. They make that models have a start, and a natural iteration order.

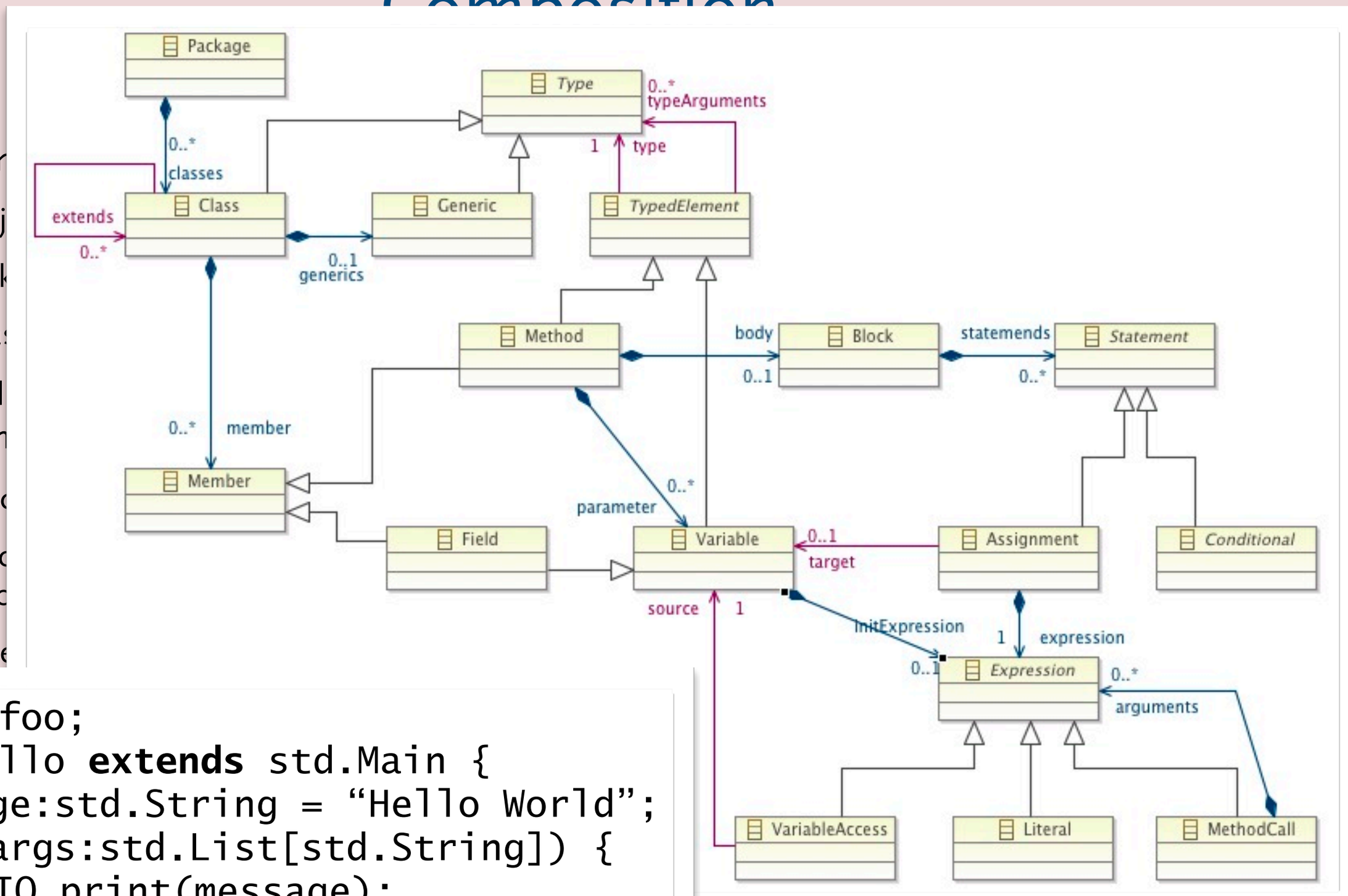
Composition

- EMF m
  - obj
  - link
  - cla
- An EM
  is span
  - Suc
  - Eac
    acc
  - The
- In a go
  in the g
- Contai
  - ... u
  - ... drive code-generation and model-transformations. They make that models have a start, and a natural iteration order.



Composition

- EMF m
  - obj
  - link
  - cla
- An EM
  is span
  - Suc
  - Eac
    acc
  - The

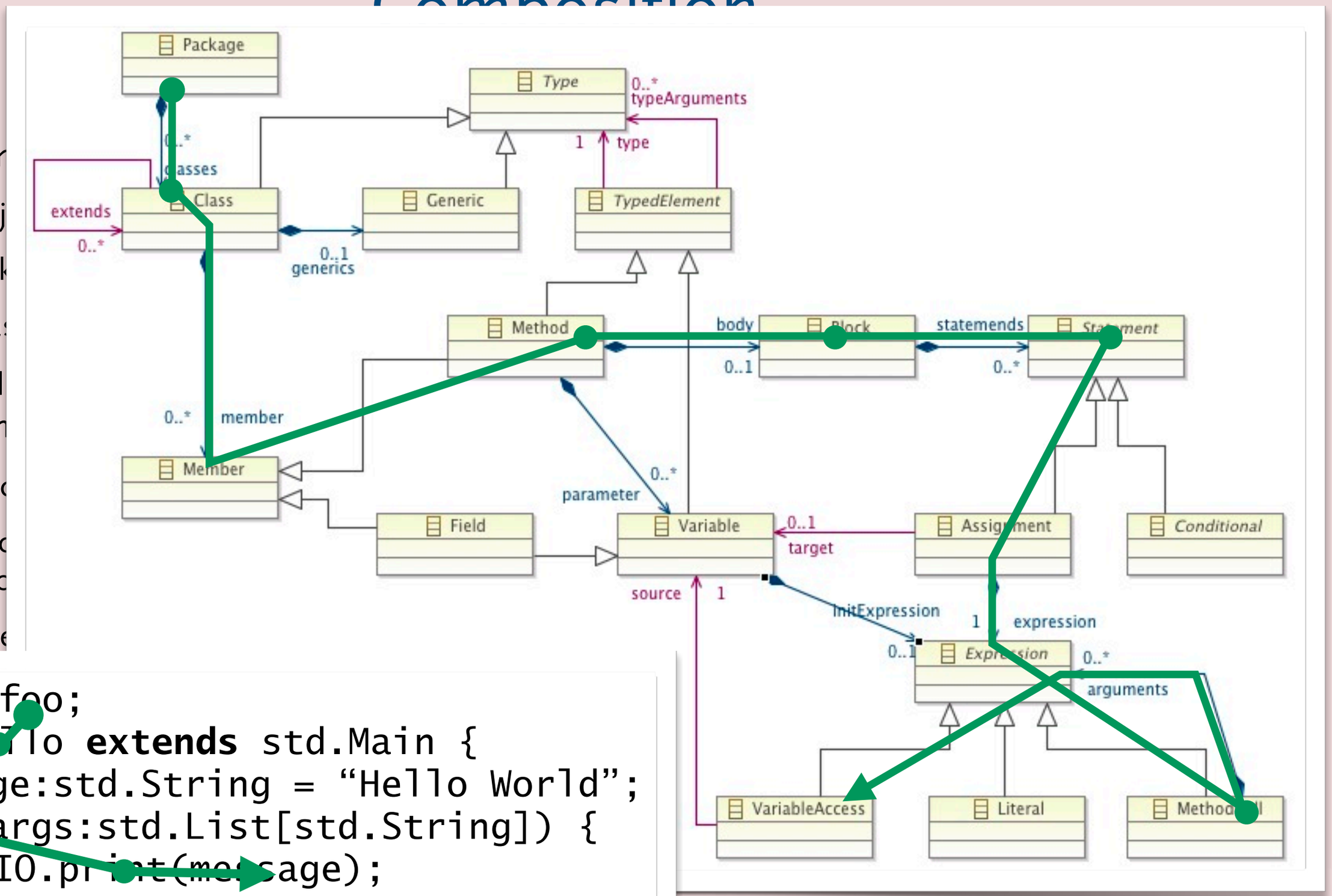



```
package foo;
class Hello extends std.Main {
   message:std.String = “Hello World”;
   main(args:std.List[std.String]) {
     std.IO.print(message);
   }
}
```

They make that models have a start, and a

Composition

- EMF m
  - obj
  - link
  - cla
- An EM
  is span
  - Suc
  - Eac
    acc
  - The




```
package foo;
class Hello extends std.Main {
   message:std.String = “Hello World”;
   main(args:std.List[std.String]) {
     std.IO.print(message);
   }
}
```

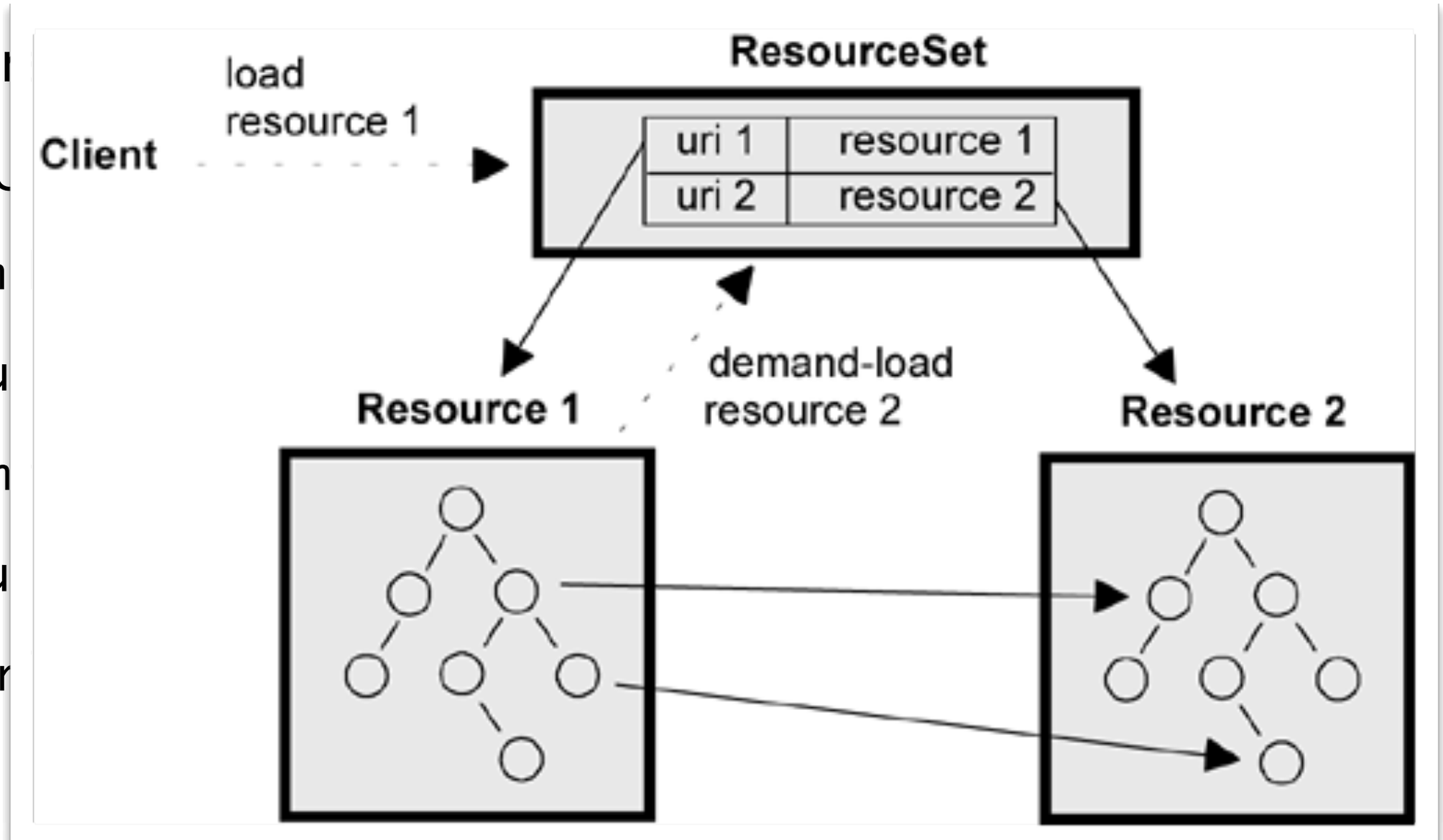
They make that models have a start, and a

Model Persistence

- EMF Persistence Interface
  - URI (Uniform Resource Identifier)
    - Standard for resource identification/location
  - Resource
    - Common interface for different storage types
  - ResourceSet
    - Interface for Resource collections

Model Persistence

- EMF Per
  - URI (U
    - Stan
  - Resou
    - Com
  - Resou
    - Inter



URI Interface

- URI = three parts string
  - **Scheme**: “file”, “jar”, “platform”, etc.
  - **Scheme-specific part**, is scheme-dependent
  - **Fragment**, identifies a part of the contents of the resource specified by the scheme
- Examples
  - The “library.xml” resource

    ```
    platform:/resource/project/fido.xml
    ```

  - First book in the “library.xml” resource

    ```
    platform:/resource/project/fido.xml#//@pets.0
    ```

Resource Interface (1/2)

- EMF Resource interface
  - Common API for different storage types
  - Persistent container of EObjects
    - EObject = base object of the EMF framework
- Resource location identified by an URI
  - Resource types identified by URI extensions
- EMF implementations
  - XMLResource
  - XMIResource (default)

Resource Interface (2/2)

- Resource interface (excerpt)
  - Interactions with persistent storage

```
void load(Map /*options*/);
void save(Map /*options*/);
void unload();
```

  - Updating contents

```
void getContents().add(Object);
void getContents().remove(Object);
```

  - Accessing contents

```
EObject getEObject(String /*URIFragment*/);
String  getURIFromEObject(EObject);
```

ResourceSet Interface

- ResourceSet = collection of Resources
  - Resources created or loaded together
  - Resource allocation
  - Automatic loading of cross-referenced Resources

- ResourceSet interface (excerpt)
  - Empty Resource creation

```
Resource createResource(URI);
```

  - Accessing contents

```
Resource getResource(URI, boolean /*loadOnDemand*/);
EObject   getEObject(URI, boolean /*loadOnDemand*/);
```

Registries

- File extensions, lets EMF know how to interpret file contents (e.g. as XMI, XML, Binary, custom format)
  - `Resource.Factory.Registry.INSTANCE.getExtensionToFactoryMap()`
  - `Map<String, ResourceFactory>`
  - e.g., `XMIResourceFactoryImpl`
  - there are local registries to each resource set
- Packages, lets EMF know what XML namespace (in XMI or XML) corresponds to what package
  - `EPackage.Registry.INSTANCE`
  - `Map<String, EPackage>`
  - e.g., `FidoPackage.eINSTANCE.getNsURI()`
- Usually not necessary when run within Eclipse. Registry is filled based on (generated) extensions.

XMI example

```
public static void main(String[] args) throws IOException {
    Owner markus = FidoFactory.eINSTANCE.createOwner();
    markus.setName("Markus");
    Dog fido = FidoFactory.eINSTANCE.createDog();
    fido.setName("Fido");
    fido.setWeight(20);
    markus.getPets().add(fido);

    Resource.Factory.Registry.INSTANCE.getExtensionToFactoryMap()
        .put("fido", new XMIResourceFactoryImpl());
    ResourceSet rs = new ResourceSetImpl();
    Resource resource = rs.createResource(URI.createFileURI("model/example.fido"));
    resource.getContents().add(markus);
    resource.save(null);
}
```

```
<?xml version="1.0" encoding="ASCII"?>
<fido:Owner xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:fido="http://fido/1.0" name="Markus">
  <pets xsi:type="fido:Dog" name="Fido" weight="20"/>
</fido:Owner>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="fido"
    nsURI="http://fido/1.0" nsPrefix="fido">
  <eClassifiers xsi:type="ecore:EClass" name="Pet" abstract="true">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
       eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="weight"
       eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Dog" eSuperTypes="#//Pet"/>
  <eClassifiers xsi:type="ecore:EClass" name="Owner">
    <eOperations name="createDescription"
       eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="pets" upperBound="-1"
       eType="#//Pet" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
       eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="description"
       eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
       volatile="true" unsettable="true" derived="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Cat" eSuperTypes="#//Pet"/>
</ecore:EPackage>
```

XMI (1)

- XMI is short for XML Metadata Interchange
- XMI is a standard (and a trademark) from the Object Management Group (OMG)
- XMI is a framework for
  - Defining, interchanging, manipulating and integrating XML data and objects
- Used for integration
  - Tools, applications, repositories, data warehouses
  - Typically used as interchange format for UML tools
- XMI defines rules for schema definition
  - Definition of schema from any valid Meta Object Facility (MOF) model
  - Schema production
- XMI defines rules for metadata generation
  - Metadata according to a MOF metamodel is generated into XML according to the generated XML schema

XMI (2)

- XMI 1.1 corresponds to MOF 1.3
- XMI 1.2 corresponds to MOF 1.4
- XMI 1.3 (added Schema support) corresponds to MOF 1.4
- XMI 2.0 (adds Schema support and changes document format) corresponds to MOF 1.4
- XMI 2.1 corresponds to MOF 2.0

XMI vs. XML

- XMI defines one mapping from ECore to XSD
- XML allows you to determine the mapping from ECore to XSD
  - e.g. via annotations
- There is also a mapping (and generator) from XSD to Ecore

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.example.com/Library"
        xmlns:lib="http://www.example.com/Library"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:complexType name="Book">
        <xsd:sequence>
            <xsd:element name="title" type="xsd:string"/>
            <xsd:element name="pages" type="xsd:int"/>
            <xsd:element name="author" type="xsd:anyURI"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Writer">
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element maxOccurs="unbounded" minOccurs="0"
                name="books" type="xsd:anyURI"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Library">
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element maxOccurs="unbounded" minOccurs="0"
                name="books" type="lib:Book"/>
            <xsd:element maxOccurs="unbounded" minOccurs="0"
                name="writers" type="lib:Writer"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www
        xmlns:lib="http://www.example.c
        xmlns:xsd="http://www.w3.org/20
    <xsd:complexType name="Book">
        <xsd:sequence>
            <xsd:element name="title" t
            <xsd:element name="pages" t
            <xsd:element name="author"
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Writer">
        <xsd:sequence>
            <xsd:element name="name" ty
            <xsd:element maxOccurs="unb
                name="books" type="xsd:
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Library">
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element maxOccurs="unbounded" minOccurs="0"
                name="books" type="lib:Book"/>
            <xsd:element maxOccurs="unbounded" minOccurs="0"
                name="writers" type="lib:Writer"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
```

```xml
<?xml version="1.0" encoding="ASCII"?>
<library:Library
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:library="http://www.example.com/Library"
    xsi:schemaLocation="http://www.example.com/Library library.ecore"
        name="Pope's Library">
  <books title="Bible" pages="1024" author="Moses et al"/>
</library:Library>
```

XMI

```
<xsd:sequence>
        <xsd:element name="name" ty
        <xsd:element maxOccurs="unb
            name="books" type=
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Library
    <xsd:sequence>
        <xsd:element name="nar
        <xsd:element maxOccurs
            name="books" type=
        <xsd:element maxOccurs
            name="writers" typ
    </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```


```xml
<?xml version="1.0" encoding="ASCII"?>
<library:Library
    xmlns:library="http://www.example.com/Library"
    name="Pope's Library">
  <Book>
     <title>Bible</title>
    <pages>1024</pages>
     <author>Moses et al</author>
  </Book>
</library:Library>
```

XML

Annotations

- Each Ecore element can contain annotations
- Similar rationale as Java annotations
- Annotation consists of
  - source URI or reference that identifies who defined/uses this kind of annotation
  - key/value pairs to provide data to the annotation source
- Example usages
  - XML mapping
  - Validation
  - Delegation (e.g. Operations)
  - Java Doc generation

XMI vs. Binary

- `BinaryResourceImpl` provides a binary persistence format that is smaller, faster, but not human readable and proprietary

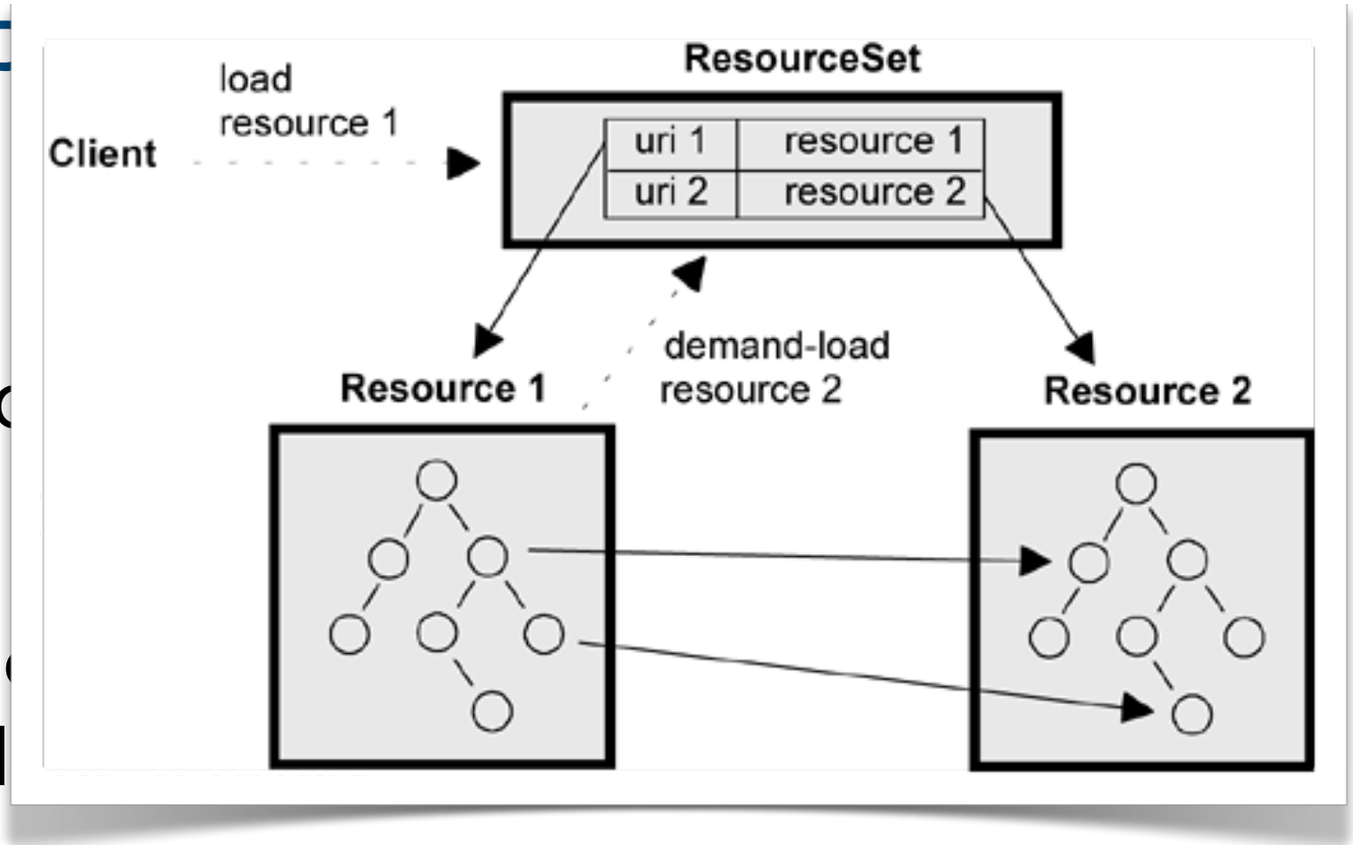
Object “ID”s

- Fragment
  - the default
  - slash-delimited path computes based on containment in the resource
- Intrinsic
  - used if an `EObject`s `EClass` defines an “ID” attribute and that attribute has a value (`isSet`)
- Extrinsic
  - used if an ID was assigned to an object
  - some resource implementations do this automatically

Proxies

- Not all resources in a resource set have to be loaded all the time. What happens with cross-references?
  - If you programmatically access an object from a resource that is not loaded, this object will be a *proxy*
  - looks like a regular `EObject`, but features cannot be accessed
  - check via `eIsProxy()`
  - resolve with `ResourceSet#resource(eObject, context)`
    - `ResourceSet`
    - `Resource`
    - containing `EObject`

P



- Not all resources in a reso
time. What happens with

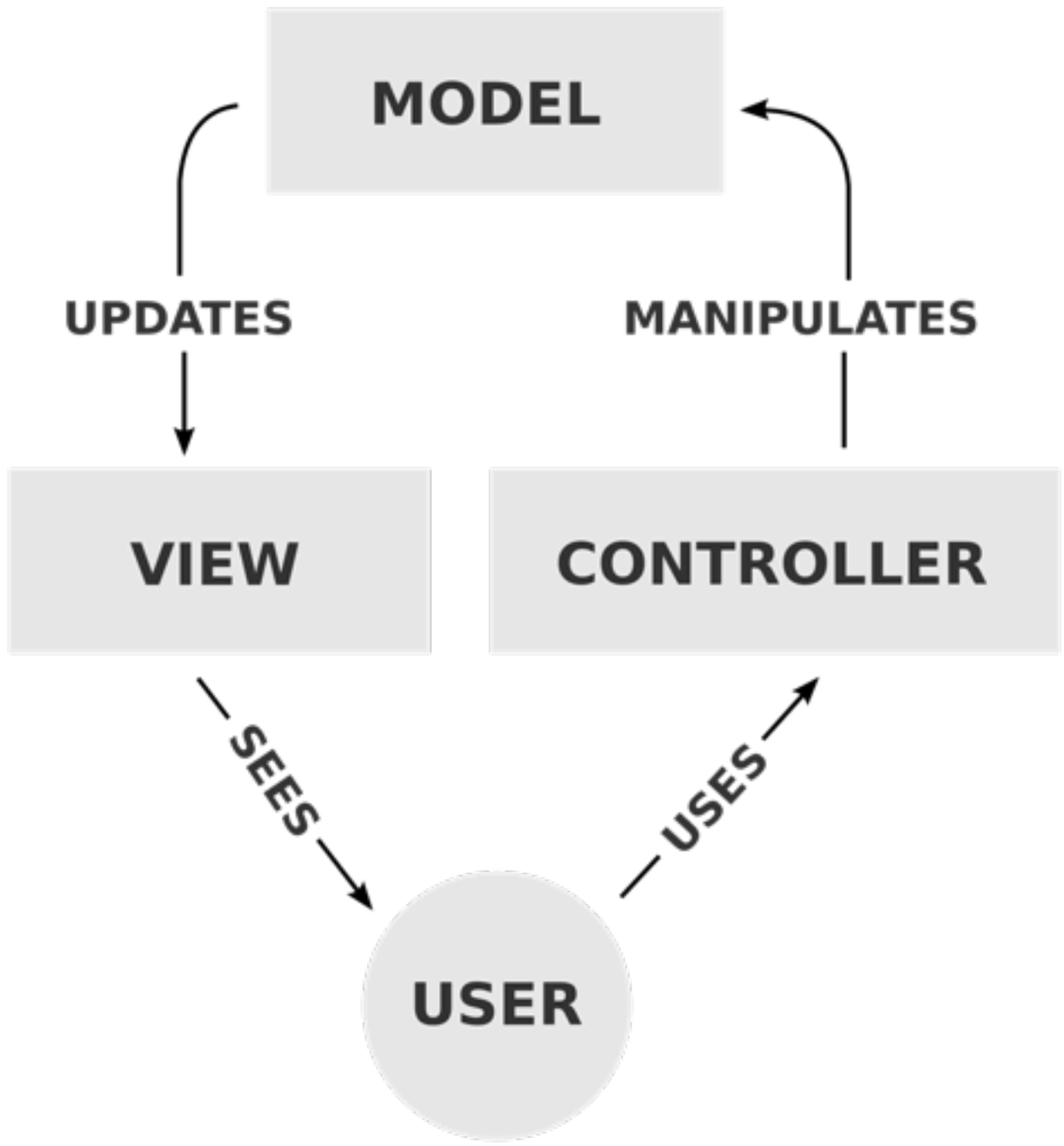
  - If you programmatically a
not loaded, this object wil
  - looks like a regular `EObject`, but features cannot be accessed
  - check via `eIsProxy()`
  - resolve with `ResourceSet#resource(eObject, context)`
    - `ResourceSet`
    - `Resource`
    - containing `EObject`

Common Errors

- `Cannot create a resource for 'model/fido.ecore'; a registered resource factory is needed`
  - no ResourceFactory for “.ecore” registered
- `org.eclipse.emf.ecore.xmi.PackageNotFoundException: Package with uri 'http://fido/1.0' not found.`
  - no EPackage for a namespace uri registered
- `org.eclipse.emf.ecore.xmi.DanglingHREFException: The object 'com.example.Foo@2f5dda ()' is not contained in a resource.`
  - a referenced object does not belong to the same resource set

Notifications and Adapter

Model-View-Controller based Applications



Notification and Adapter Interfaces

- Independent from ECore
- Adapters
  - are implemented by you
  - can be installed on
    - Resources
    - ResourceSets
    - EObjects
- Notifications are implemented by EMF

```
Owner markus = FidoFactory.eINSTANCE.createOwner();
markus.eAdapters().add(new AdapterImpl() {
    @Override
    public void notifyChanged(Notification notification) {
        if (notification.getEventType() == Notification.ADD) {
            System.out.println("ADD a "
                    + notification.getNewValue().getClass().getSimpleName()
                    + " to feature "
                    + ((EStructuralFeature)notification.getFeature()).getName());
        } else if (notification.getEventType() == Notification.SET) {
            System.out.println("SET a "
                    + notification.getNewValue().getClass().getSimpleName()
                    + " to feature "
                    + ((EStructuralFeature)notification.getFeature()).getName());
        } else {
            System.out.println(notification);
        }
        super.notifyChanged(notification);
    }
});

markus.setName("Markus");
Dog fido = FidoFactory.eINSTANCE.createDog();
markus.getPets().add(fido);

fido.setName("Fido");
fido.setWeight(20);
```

```
Owner markus = FidoFactory.eINSTANCE.createOwner();
markus.eAdapters().add(new AdapterImpl() {
    @Override
    public void notifyChanged(Notification notification) {
        if (notification.getEventType() == Notification.ADD) {
            System.out.println("ADD a "
                    + notification.getNewValue().getClass().getSimpleName()
                    + " to feature "
                    + ((EStructuralFeature)notification.getFeature()).getName());
        } else if (notification.getEventType() == Notification.SET) {
            System.out.println("SET a "
                    + notification.getNewValue().getClass().getSimpleName()
                    + " to feature "
                    + ((EStructuralFeature)notification.getFeature()).getName());
        } else {
            System.out.println(notification);
        }
        super.notifyChanged(notification);
    }
});

markus.setName("Markus");
Dog fido = FidoFactory.eINSTANCE.createDog
markus.getPets().add(fido);

fido.setName("Fido");
fido.setWeight(20);
```

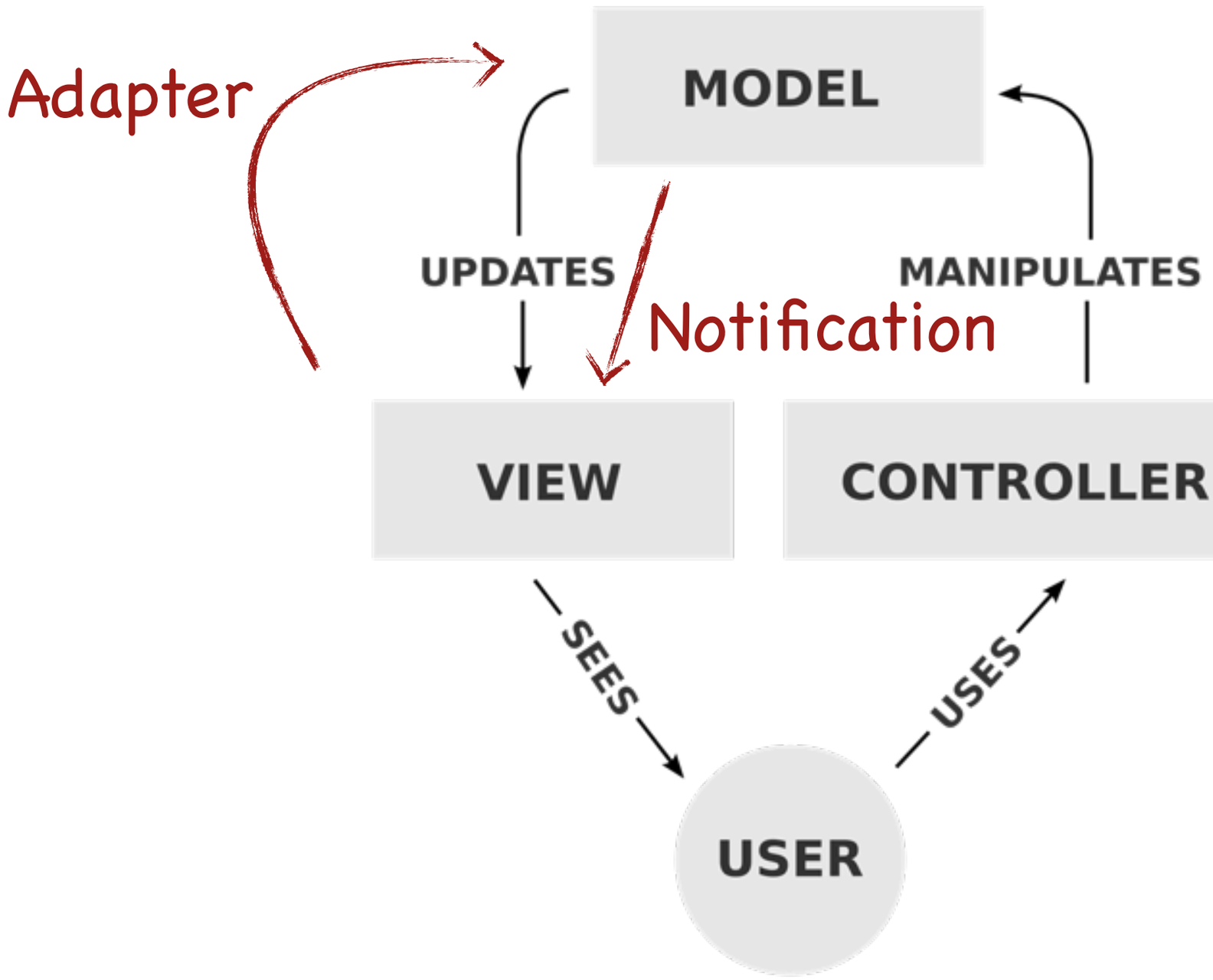
```
SET a String to feature name
ADD a DogImpl to feature pets
```

```
Owner markus = FidoFactory.eINSTANCE.createOwner();
markus.eAdapters().add(new EContentAdapter() {
    @Override
    public void notifyChanged(Notification notification) {
        if (notification.getEventType() == Notification.ADD) {
            System.out.println("ADD a "
                    + notification.getNewValue().getClass().getSimpleName()
                    + " to feature "
                    + ((EStructuralFeature)notification.getFeature()).getName());
        } else if (notification.getEventType() == Notification.SET) {
            System.out.println("SET a "
                    + notification.getNewValue().getClass().getSimpleName()
                    + " to feature "
                    + ((EStructuralFeature)notification.getFeature()).getName());
        } else {
            System.out.println(notification);
        }
        super.notifyChanged(notification);
    }
});

markus.setName("Markus");
Dog fido = FidoFactory.eINSTANCE.createDog();
markus.getPets().add(fido);

fido.setName("Fido");
fido.setWeight(20);
```

```
Owner markus = FidoFactory.eINSTANCE.createOwner();
markus.eAdapters().add(new EContentAdapter() {
    @Override
    public void notifyChanged(Notification notification) {
        if (notification.getEventType() == Notification.ADD) {
            System.out.println("ADD a "
                    + notification.getNewValue().getClass().getSimpleName()
                    + " to feature "
                    + ((EStructuralFeature)notification.getFeature()).getName());
        } else if (notification.getEventType() == Notification.SET) {
            System.out.println("SET a "
                    + notification.getNewValue().getClass().getSimpleName()
                    + " to feature "
                    + ((EStructuralFeature)notification.getFeature()).getName());
        } else {
            System.out.println(notification);
        }
        super.notifyChanged(notification);
    }
});

markus.setName("Markus");
Dog fido = FidoFactory.eINSTANCE.createDog(
markus.getPets().add(fido);

fido.setName("Fido");
fido.setWeight(20);
```

```
SET a String to feature name
ADD a DogImpl to feature pets
SET a String to feature name
SET a Integer to feature weight
```

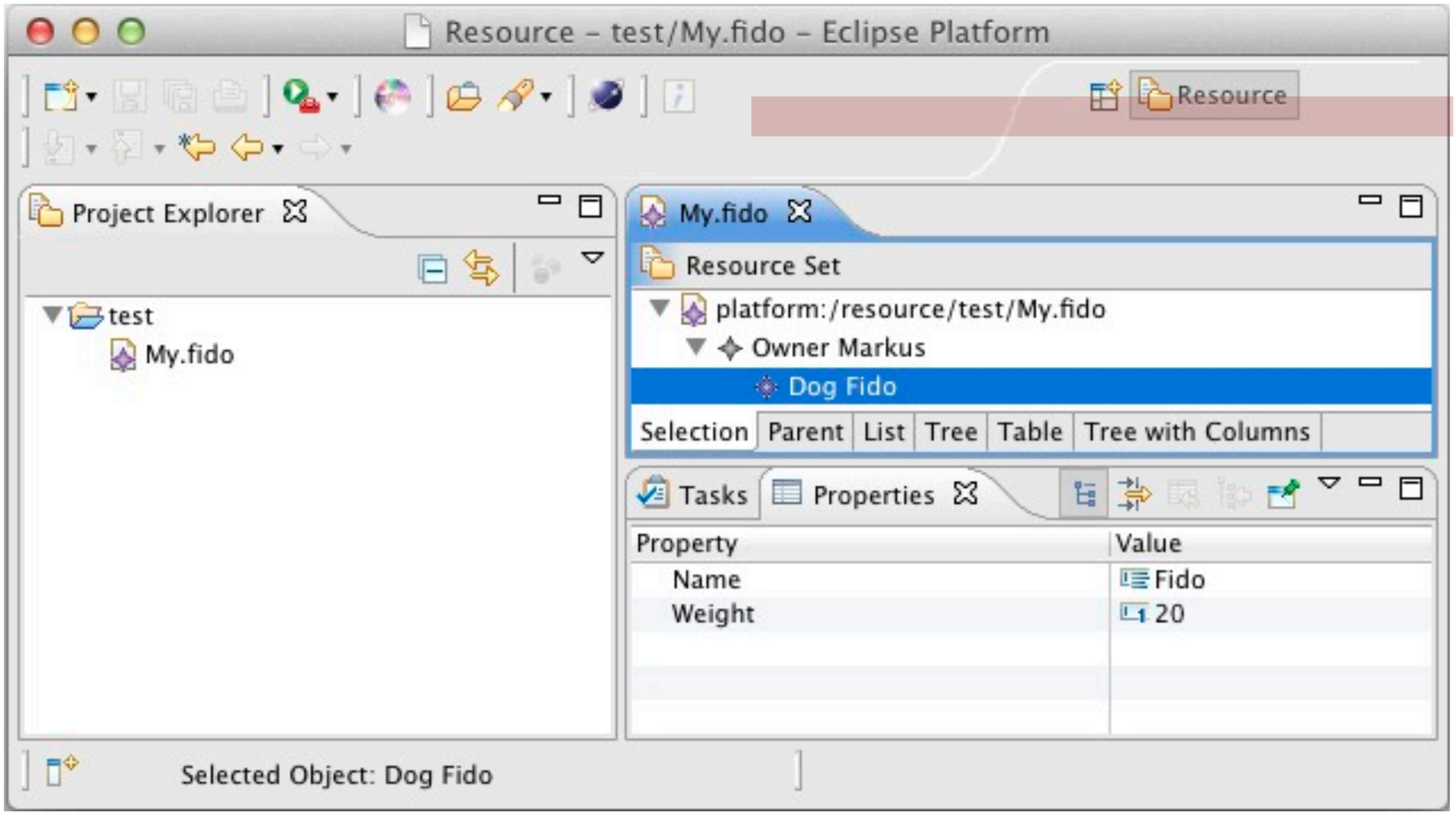
Model-View-Controller based Applications



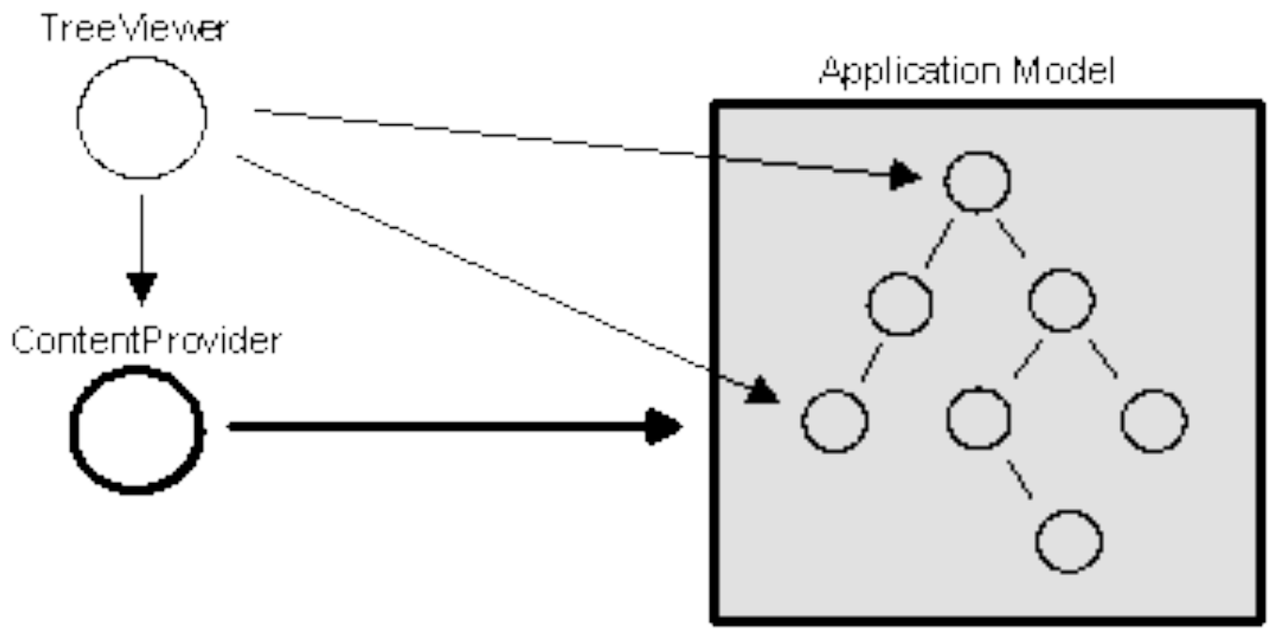
Edit and Editor Plugins

- fido.edit
  - ItemProviderAdapterFactory for each EPackage
  - ItemProvider for each EClass
    - icon
    - text
    - children
    - ...
- fido.editor
  - Tree editor
  - Action bar contributor

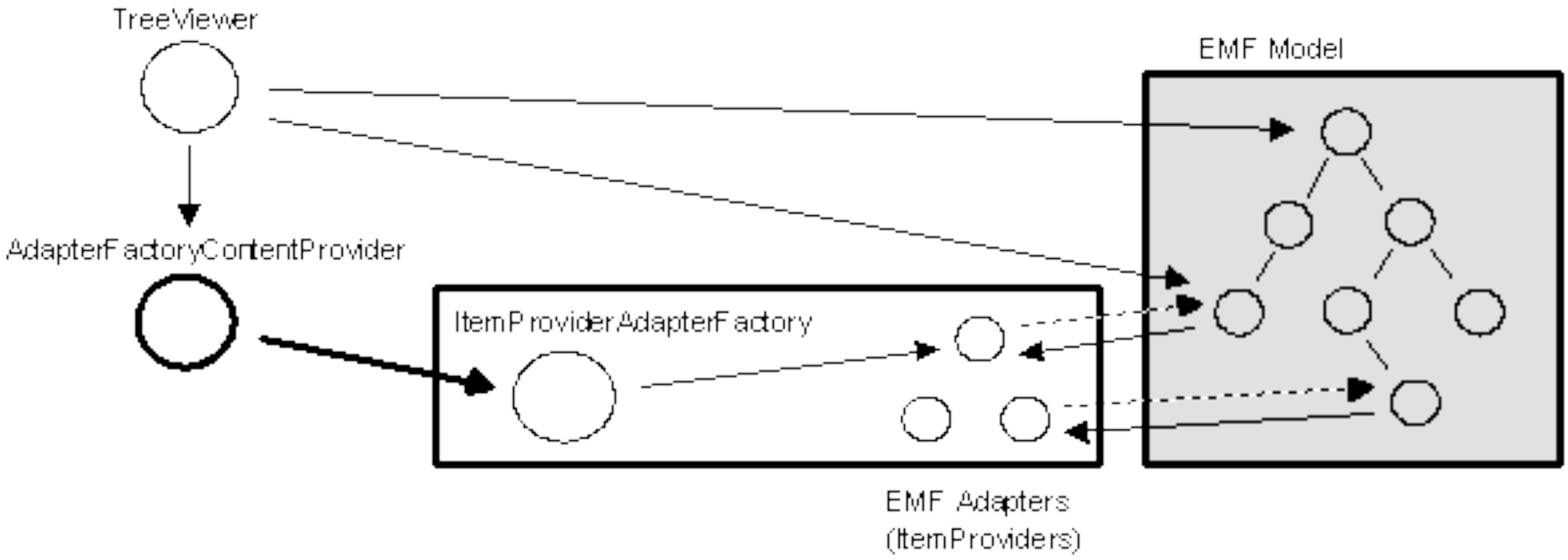
Tree Editor for Fido Example



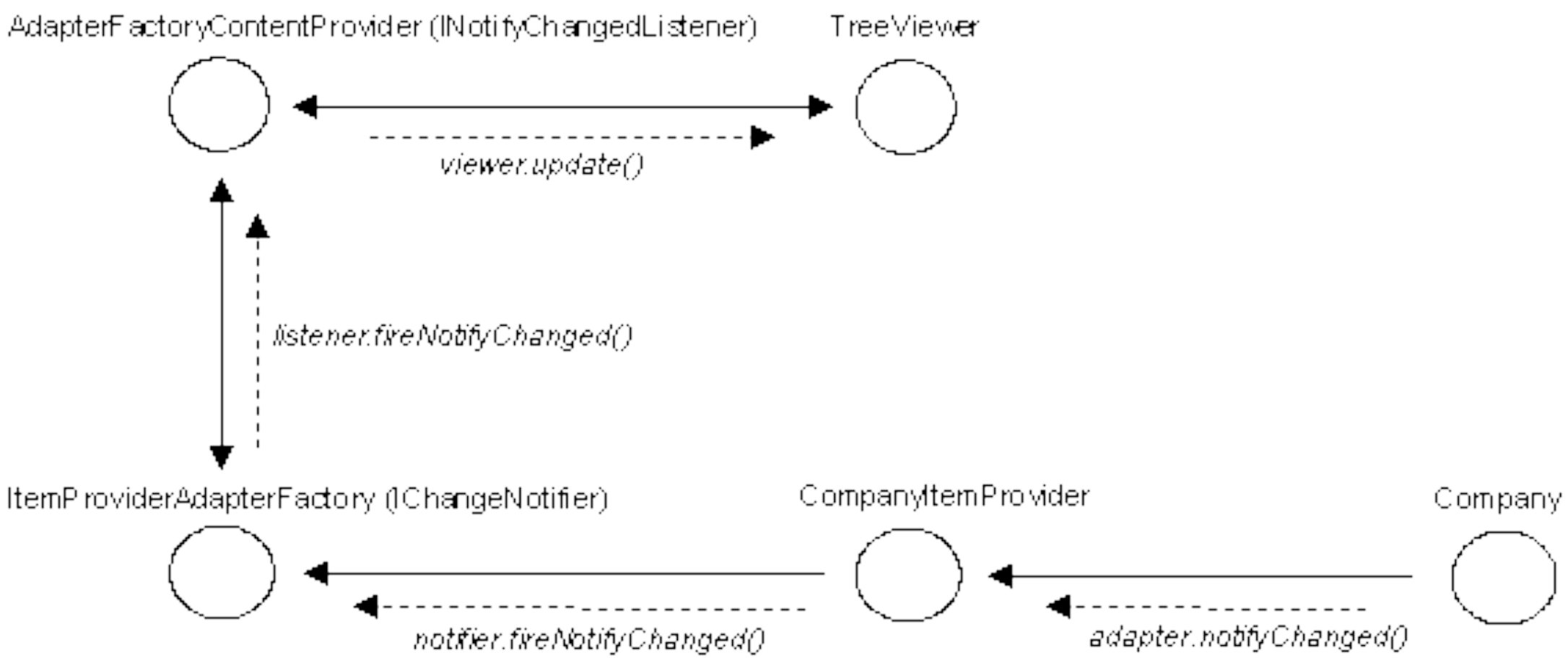
MVC-based Tree Viewer in General



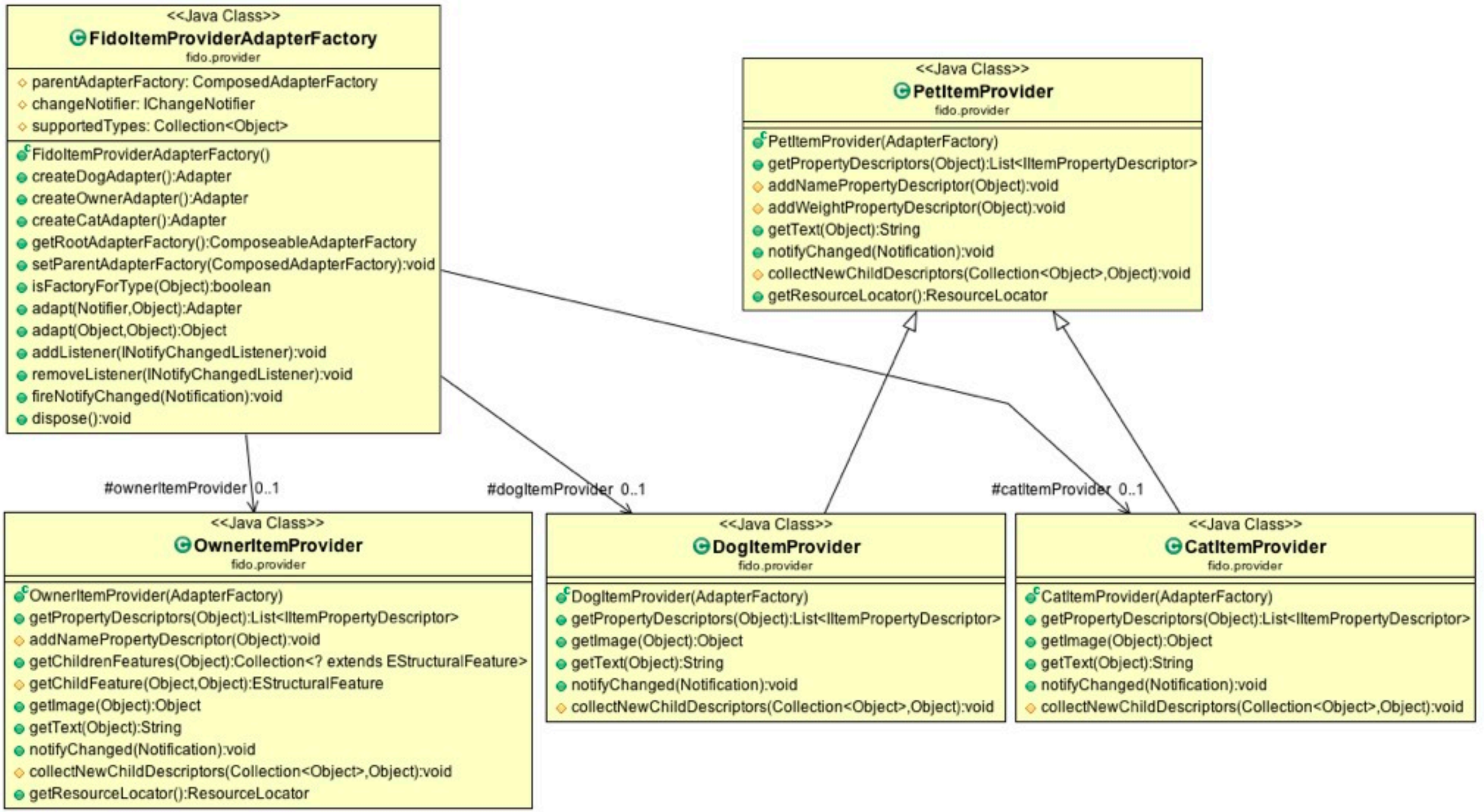
EMF's MVC-based Tree Viewer



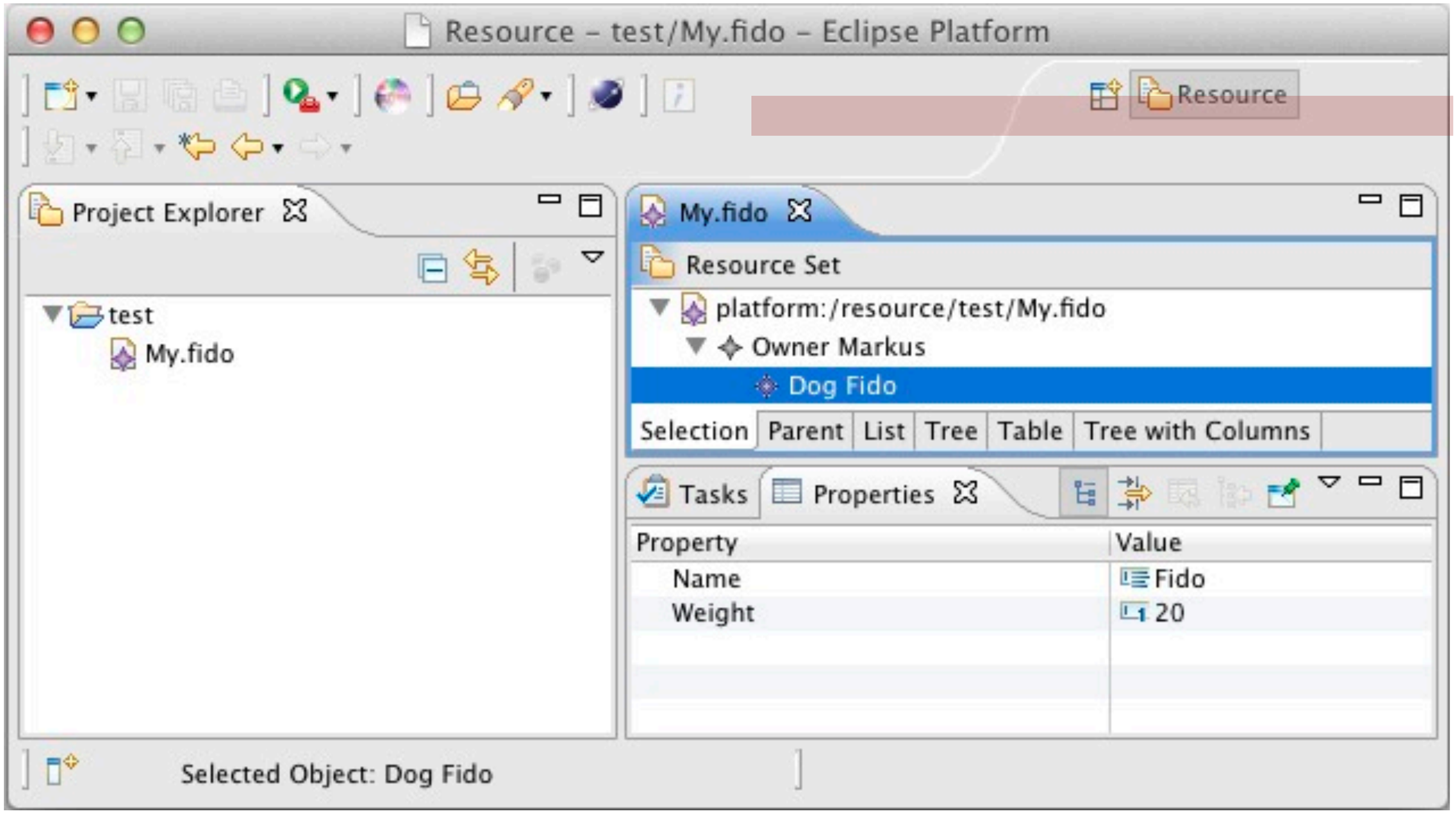
Notifications in EMF Tree Viewer



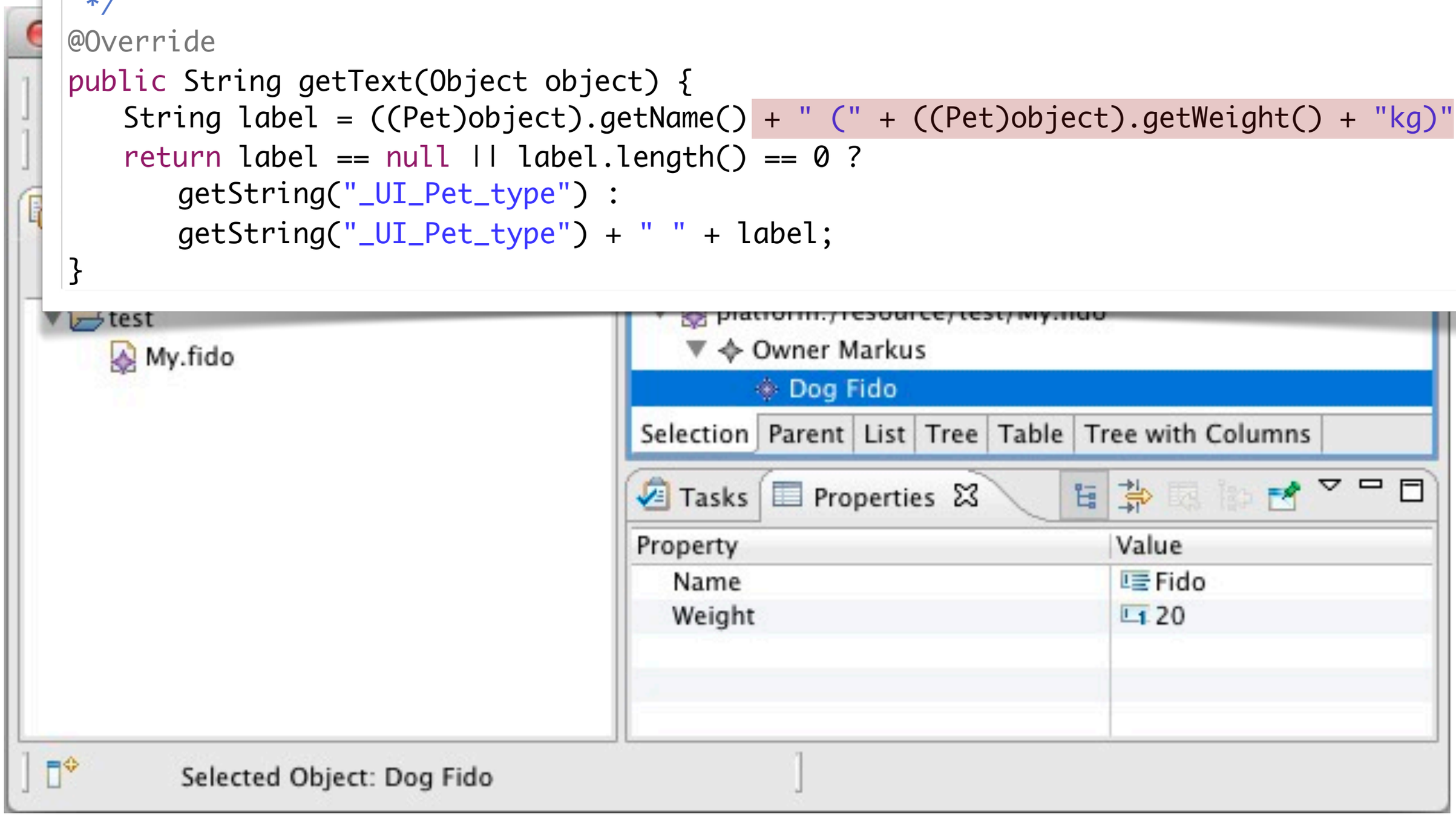
ItemProviderAdapterFactory and ItemProvider Examples



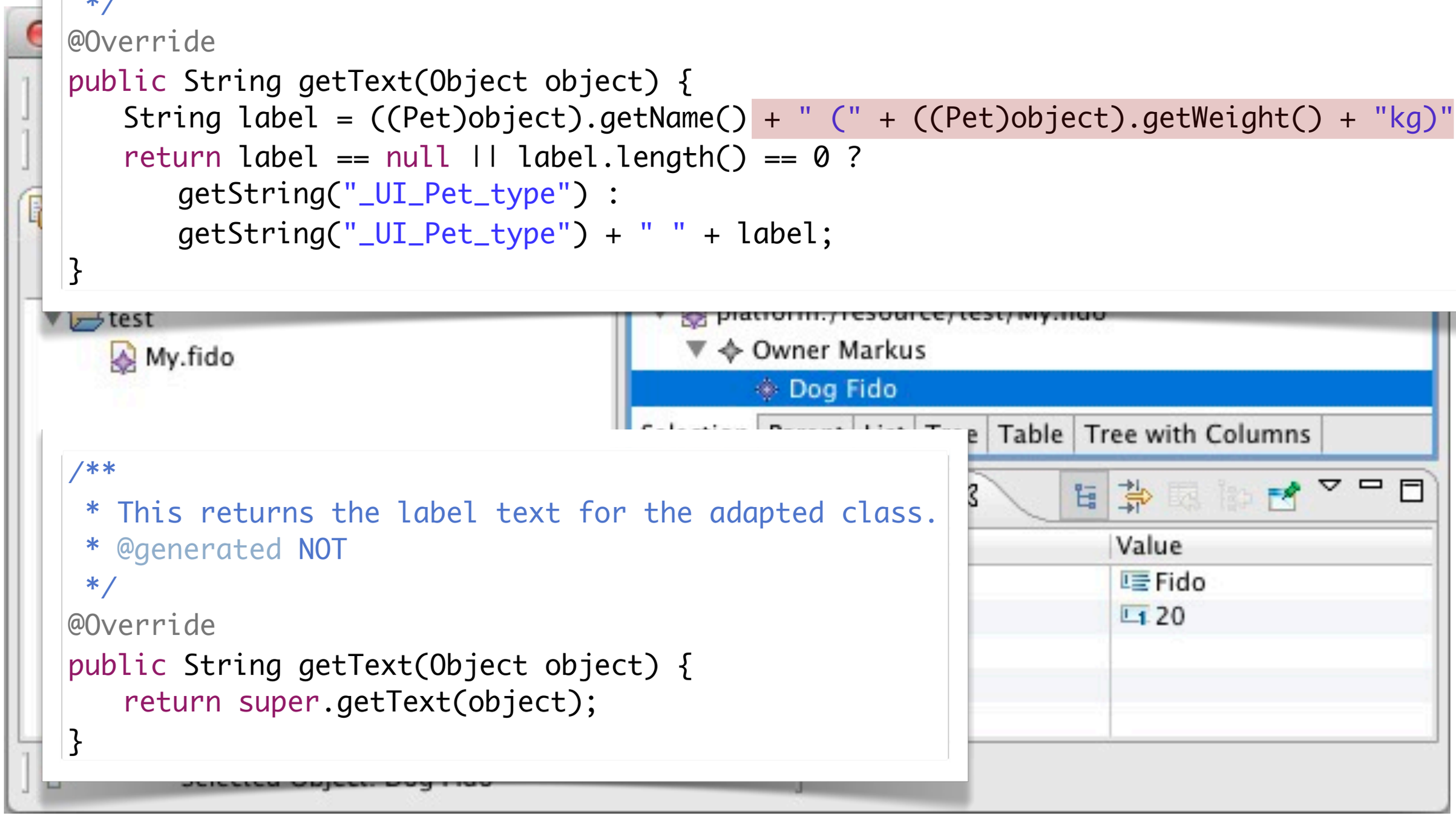
Tree Editor for Fido Example



```java
/**
 * This returns the label text for the adapted class.
 * @generated NOT
 */
@Override
public String getText(Object object) {
    String label = ((Pet)object).getName() + " (" + ((Pet)object).getWeight() + "kg)";
    return label == null || label.length() == 0 ?
        getString("_UI_Pet_type") :
        getString("_UI_Pet_type") + " " + label;
}
```

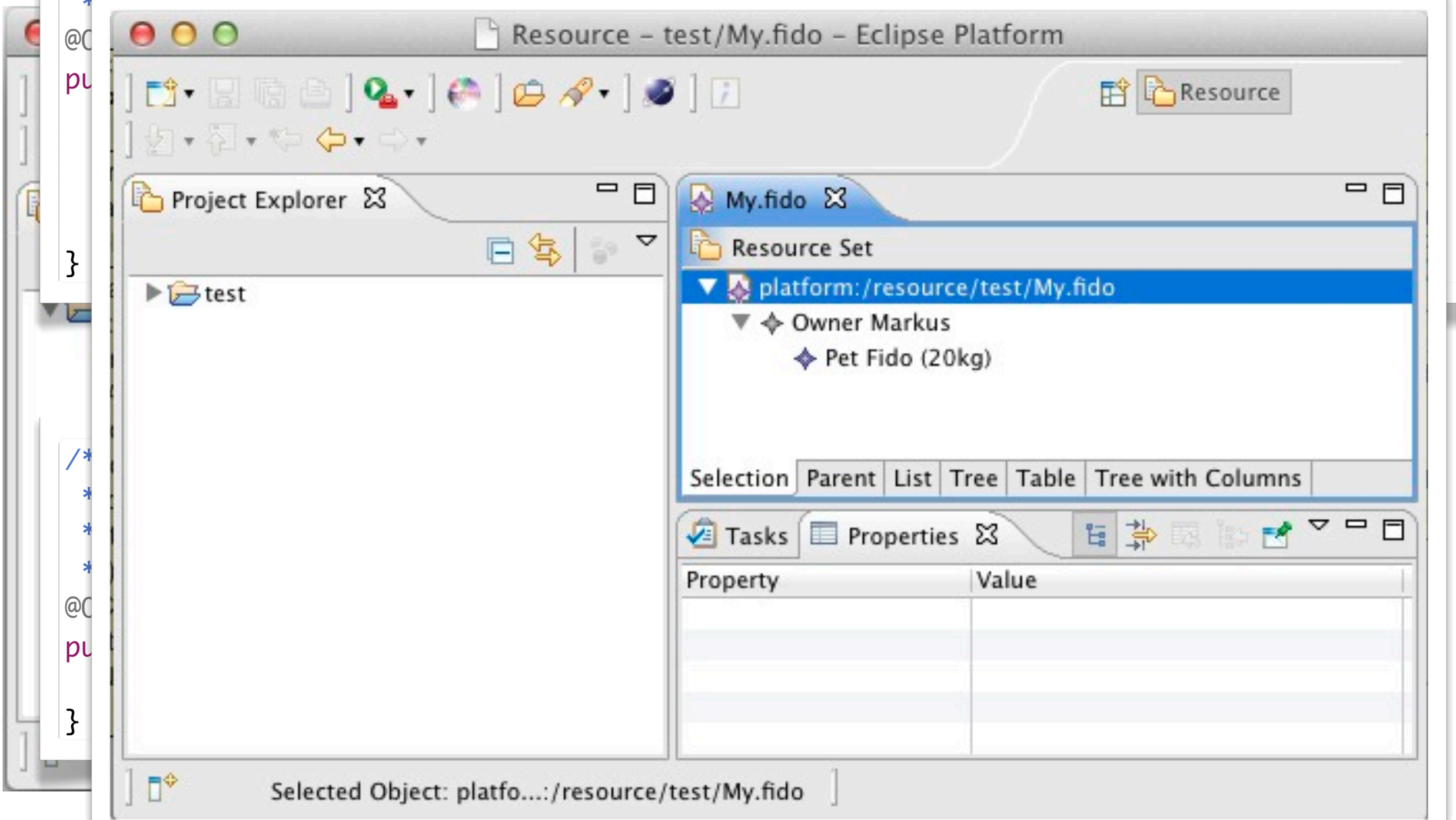


```java
/**
 * This returns the label text for the adapted class.
 * @generated NOT
 */
@Override
public String getText(Object object) {
    String label = ((Pet)object).getName() + " (" + ((Pet)object).getWeight() + "kg)";
    return label == null || label.length() == 0 ?
        getString("_UI_Pet_type") :
        getString("_UI_Pet_type") + " " + label;
}
```




```java
/**
 * This returns the label text for the adapted class.
 * @generated NOT
 */
@Override
public String getText(Object object) {
    return super.getText(object);
}
```

```
/**
 * This returns the label text for the adapted class.
 * @generated NOT
 *
@C
pu


}
```



.genmodel

.genmodel (1)

- One .genmodel per Ecore Model
- Stears code generation
- Allows to customize generation for
  - model code
  - each package, class, attribute
  - edit code
  - editor code

.genmodel (2)

- Package names/structure
- Containment proxies
- Suppressing various features
- Feature delegation
- Operation reflection
- minimal reflective methods
- root class, interface, e.g. MinimalEObject (virtual feature delegation)
- Extending/replacing generator templates

Summary

- ECore allows to model EMOF-like meta-models
- EMF provides
  - meta-model dependent interface
  - reflective interface
  - XMI/XML serialization
  - notifications
  - simple tree editor, extendable