

The diagram illustrates the Ecore framework's class structure. Key components include:

- EModelElement**: Base class for model elements, with methods like `getEAnnotation(EString) : EAnnotation`.
- EAnnotation**: Represents annotations, with a `source : EString` attribute.
- ENamedElement**: Base class for named elements, with a `name : EString` attribute.
- EFactory**: Provides methods for creating and converting objects, such as `create(EClass) : EObject` and `convertToString(EDataType, EJavaObject) : EString`.
- ETypedElement**: Base class for typed elements, with attributes like `ordered : EBoolean`, `unique : EBoolean`, and `required : EBoolean`.
- EClassifier**: Base class for classifiers, with methods like `isInstance(EJavaObject) : EBoolean` and `getClassifierID() : EInt`.
- EPackage**: Represents packages, with attributes like `nsURI : EString` and `nsPrefix : EString`.
- EClass**: Base class for classes, with methods like `isSuperTypeOf(EClass) : EBoolean` and `getFeatureCount() : EInt`.
- EDataType**: Base class for data types, with a `serializable : EBoolean` attribute.
- EEnumLiteral**: Represents enumeration literals, with attributes like `value : EInt` and `literal : EString`.
- EEnum**: Represents enumerations, with methods like `getEEnumLiteral(EString) : EEnumLiteral`.
- EStructuralFeature**: Base class for structural features, with methods like `getFeatureID() : EInt` and `getContainerClass() : EJavaClass`.
- EReference**: Represents references, with attributes like `containment : EBoolean` and `resolveProxies : EBoolean`.
- EAttribute**: Represents attributes, with an `id : EBoolean` attribute.
- EObject**: Base class for objects, with methods like `getEAnnotation(EString) : EAnnotation`.
- EOperation**: Represents operations, with a `parameters : EList` attribute.
- EParameter**: Represents parameters, with a `name : EString` attribute.
- Cell**: A simple class representing a cell.
- Expression**: Base class for expressions, with a `subexprs : EList` attribute.
- BinExpression**: Represents binary expressions, with a `function : String` attribute.
- Value**: Represents a value.
- CellRef**: Represents a cell reference.

The diagram shows various associations and generalizations between these classes, such as `EModelElement` generalizing `ENamedElement` and `ETypedElement`, and `EClassifier` generalizing `EClass` and `EDataType`.

Q1 : Entrer le méta-modèle du tableur dans un nouveau fichier Ecore.

File -> New -> Other -> Ecore Model

L'éditeur d'Ecore permet de créer simplement des modèles conformes à notre méta-modèle.

Clic droit sur Cell dans le méta-modèle puis Create Dynamic Instance...

Ouvrir le fichier créé avec Sample Reflective Ecore Model Editor

Q2.1 : Créer un modèle d'un tableur comportant quelques cellules. Quel problème rencontre-t-on avec l'éditeur ?

Q2.2 : Proposer une solution pour remédier à ce problème.

Navigation dans un modèle

L'objectif est de visualiser les éléments du méta-modèle du tableur à l'aide de 3 opérations :

- **flat** : affiche la hiérarchie d'héritage des classes
- **short** : décrit les attributs et les opérations d'une classe
- **flatShort** : short en incluant les membres hérités des superclasses

L'affichage peut prendre la forme suivante, par exemple pour flatShort :

UneClasse :

attr nomAttr : NomType

op nomOp (nomArg : NomType)

SuperClasse :

ref nomRole : AutreClasse[0..]*

Q3.1 : Implémenter ces 3 opérations dans une classe xtend EcoreHelper.

Q3.2 : Implémenter ces 3 opérations dans un aspect EclassAspect en utilisant l'annotation @Aspect de Kermeta 3.

Q3.3 : Quels sont les avantages et les inconvénients d'utiliser @Aspect par rapport à du Xtend pur ?

On souhaite ajouter un identifiant unique à chaque classe. Cet identifiant sera calculé via la méthode **setIdentifiant** qui concaténera le hash code du nom de la classe avec le temps actuel.

id = maClasse.name.hashCode + "_" + System.currentTimeMillis

On appellera **setIdentifiant** sur toutes les classes du méta-modèle juste après le chargement de celui-ci afin de ne définir les identifiants qu'une seule fois.

Q4.1 : Implémenter **setIdentifiant** dans EcoreHelper et modifier les précédentes opérations pour ajouter l'affichage de cet identifiant.

Q4.2 : Idem dans EclassAspect.

Q4.3 : Quels sont les avantages et les inconvénients d'utiliser @Aspect par rapport à du Xtend pur ?