

Institut Supérieur d'Informatique et du Multimédia de Sfax

Ingénierie des modèles

Auditoire: 2 éme Ingénieur

INFO **2024/2025**

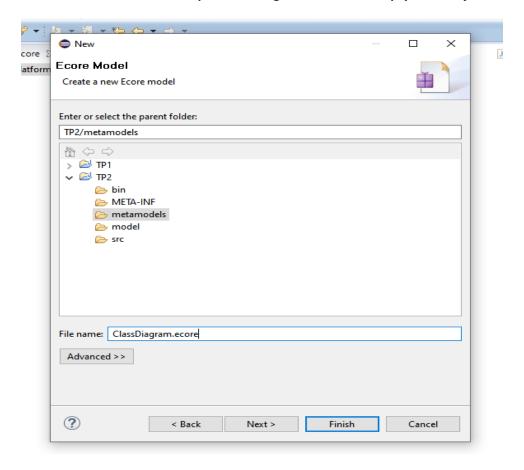
Responsable de cours : Mohamed Hadj Kacem Enseignante TP : ZAINEB

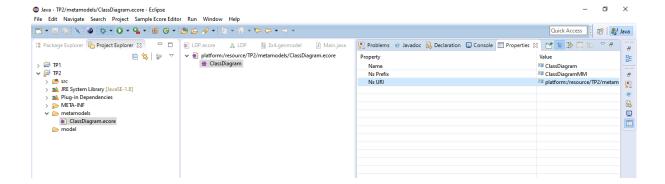
SAKHRAOUI

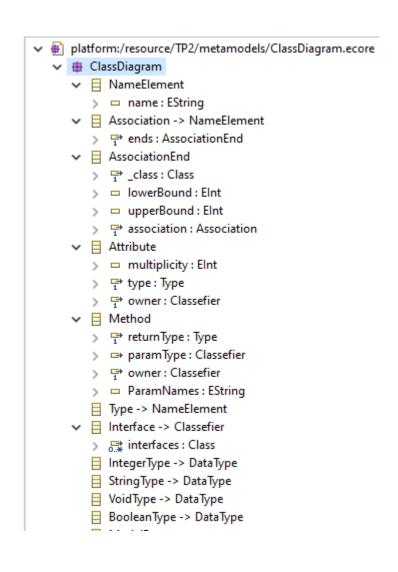
TP 2 : Diagramme de classes UML

Exercice 1 : Création d'un méta-modèle d'un diagramme de classes

File -> New -> Other -> Eclipse Modeling Framework -> Empty EMF Project







✓ ■ ModelBase
 → allAssociations: Association
 → allInterfaces: Interface
 → integerT: IntegerType
 → StringT: StringType
 → VoidT: VoidType
 → boolT: BooleanType
 → allClasses: Class
 ✓ ■ Class -> Classefier
 → EmplementBy: Interface
 → associations: AssociationEnd
 ✓ ■ Classefier -> Type
 → methods: Method
 → attributes: Attribute
 ■ DataType -> Type

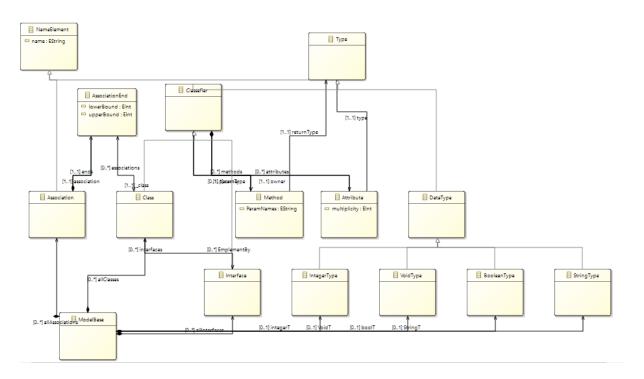
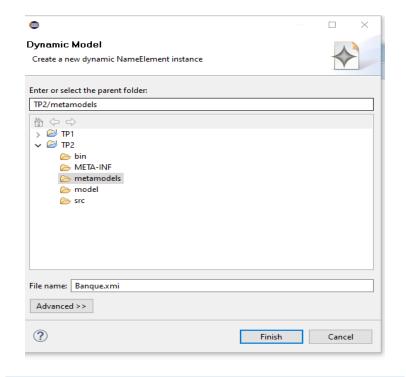


Figure 1: Un Méta-modèle d'un diagramme de classes.

Exercice 2 : Création et édition d'un modèle (La banque)



- Aplatform:/resource/TP2/metamodels/Banque.xmi
 - ✓ ♦ Model Base
 - → Association accounts
 - Association End 0
 - Association owner
 - Association End 0
 - Interface loperation
 - Interface Association
 - ♦ Integer Type
 - String Type
 - ♦ Void Type
 - ✓ ♦ Class Client
 - Method getNom
 - ♦ Method changerAdr
 - Attribute Nom
 - ♦ Attribute Adr
 - ✓ ♦ Class Account
 - ♦ Method getOverdraft
 - ♦ Method getId
 - Attribute id
 - ♦ Attribute solde
 - Attribute overdraft
- > # platform:/resource/TP2/metamodels/ClassDiagram.ecore

Figure 2 Création d'une instance Banque.xmi.

Exercice 3 : Validation de modèle et de méta-modèle

1. Ajouter le contenu du code suivant qui présente un invariant vérifiant qu'un client possède un nom unique.

```
class client extends Class {
    attribute Nom:String[1];
    attribute Adr:String;
    operation getAdr():String;
    invariant NameUnique: client.allInstances()-> forAll(cl,c2|c1<>c2 implies c1.Nom<>c2.Nom);
}
```

Figure 3 class client

2. Ajouter un invariant pour garantir que toutes les associations doivent avoir un nom non vide.

```
class Association extends NameElement
{
    property ends : AssociationEnd[1] { composes };
    invariant NotEmpty: self.name<> null and self.name.size() >0;
}
```

Figure 4 associations non vide

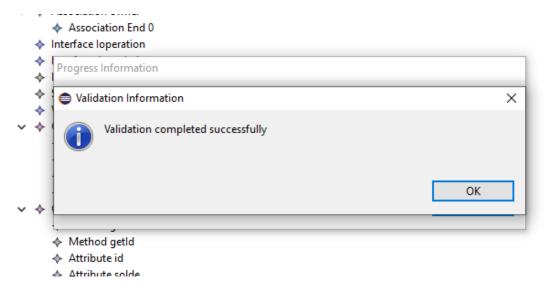


Figure 5 validation