

Compte rendu TP n°2

Implémentation d'un projet ECORE

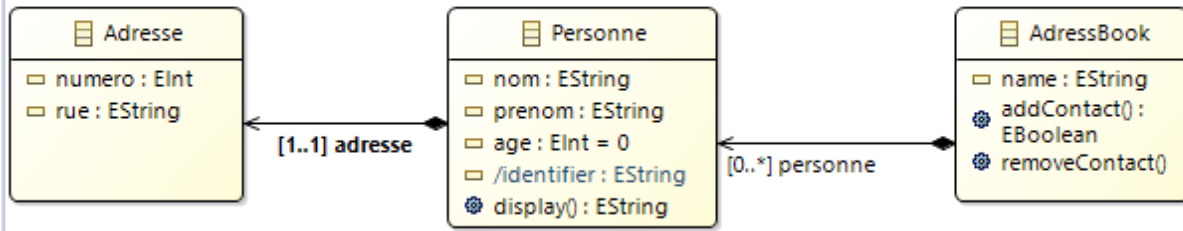
Contraintes OCL

Module UML

Réalisé par :

- MEDDAH Amine

## I – Diagramme UML



## II - Insertion de contrainte OCL

### Q II.1 Paramétrage du modèle

Modifications au projet :

Model Plug-in Variables	OCL_ECORE=org.eclipse.ocl.ecore...
-------------------------	------------------------------------

Dynamic Templates	<input checked="" type="checkbox"/> true
Facade Helper Class	<input type="checkbox"/> org.eclipse.emf.codegen.merge.java.facade.a
Force Overwrite	<input checked="" type="checkbox"/> false
Import Organizing	<input checked="" type="checkbox"/> true
Plugin Key	<input type="checkbox"/> addressbook
Redirection Pattern	<input type="checkbox"/>
Template Directory	<input type="checkbox"/> /{workspace}/{package-projet}/templates

- Logic Diagrams
- Maven
- Model Validation
- MoDisco
- Mwe2
- Mylyn
- OCL

### OCL

Options common to all bindings.

Executor targeted by the default OCL delegate	<input type="text" value="http://www.eclipse.org/emf/2002/Ecore/OCL/LPG"/>
Realisation of OCL embedded within Ecore models	<input type="text" value="Generate Java Code in *Impl classes"/>

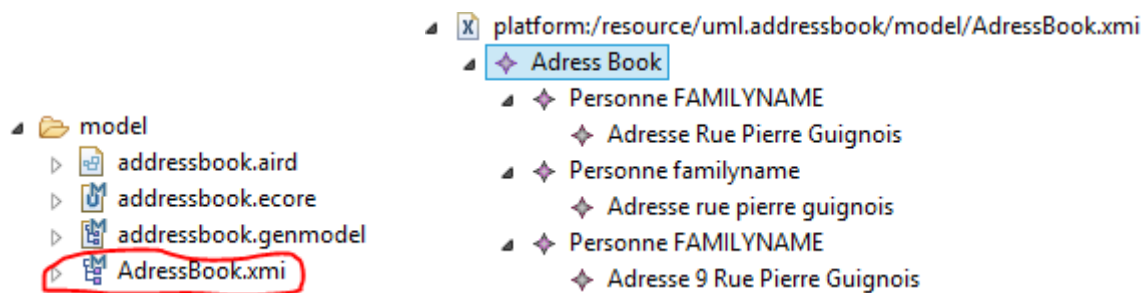
## Q II.2 Insertion des contraintes OCL

Ajout des contrainte à la Class Personne

```
class Personne
{
    { invariant minAge:
      self.age>16;
    invariant majNom:
      self.nom = self.nom.toUpperCase(); }
    operation display() : String[];
    attribute nom : String[];
    attribute prenom : String[];
    attribute age : ecore::EInt[] = '0';
    property adresse : Adresse[1] { composes };
    attribute identifier : String[] { derived readonly transient };
}
```

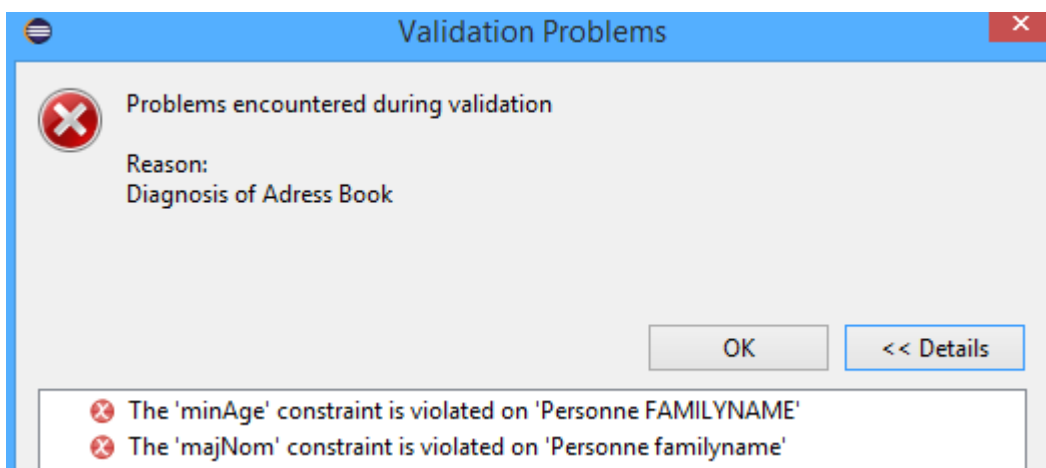
## Q II.3 Simulation des instances

Création de fichier xmi et compléter le carnet d'adresses

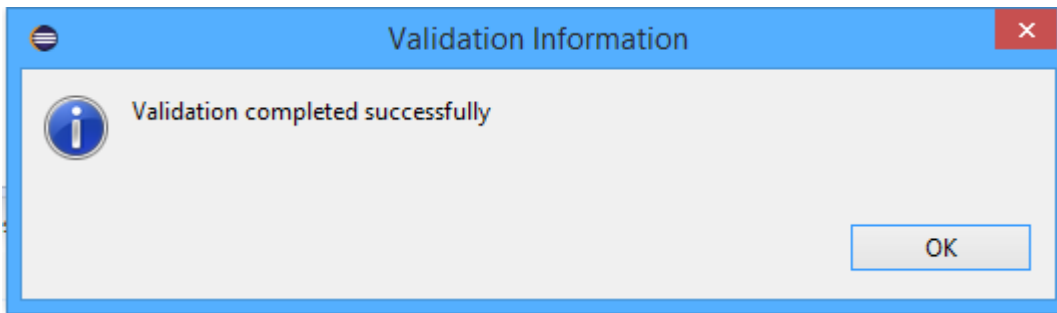


## Q II.4 Validation OCL

Validation de fichier xmi avec des erreurs dans la saisie des champs Age (<16) et Nom (en minuscule)



Après la correction des erreurs



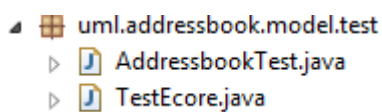
Affichage de contenu du fichier AdressBook.xmi

```
addressbook.genmodel  addressbook.ecore  AdressBook.xmi  AdressBook.xmi
1  <?xml version="1.0" encoding="UTF-8"?>
2  <addressbook:AdressBook xmi:version="2.0" xmlns:xmi="http://www.om
3  <personne nom="FAMILYNAMEA" prenom="FIRSTNAME" age="20">
4    <adresse numero="7" rue="Rue Pierre Guignois"/>
5  </personne>
6  <personne nom="FAMILYNAMEB" prenom="firsrtname" age="25">
7    <adresse numero="8" rue="rue pierre guignois"/>
8  </personne>
9  <personne nom="FAMILYNAMEC" prenom="firstname" age="22">
10    <adresse rue="9 Rue Pierre Guignois"/>
11  </personne>
12 </addressbook:AdressBook>
13
```

### III – Méta-Model Ecore

#### Q III.1 Analyse du méta-modèle Ecore

Création de la classe TestEcore



Après la validation et l'exécution du code

```
Adresse
    numero (EInt)
    rue (EString)
Personne
    nom (EString)
    prenom (EString)
    age (EInt)
    identifier (EString)
AdressBook
    name (EString)
```

## Q III.2 Affichage du contenu

Compléter le code de la classe créée précédemment et après la validation et l'exécution

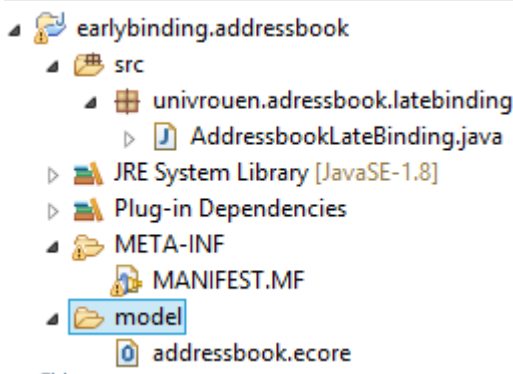
```
Adresse
    numero(EInt),  rue(EString),
Personne
    nom(EString),  prenom(EString),  age(EInt),  identifier(EString),
    Références : adresse(Adresse[1..1])
    Opérations : EString display,  EBoolean minAge,  EBoolean majNom,
AdressBook
    name(EString),
    Références : personne(Personne[0..-1])
    Opérations : EBoolean addContact,
```

## IV – Manipulation du modèle Ecore

### Q IV.1 Création d'instance via la méta-modèle Ecore

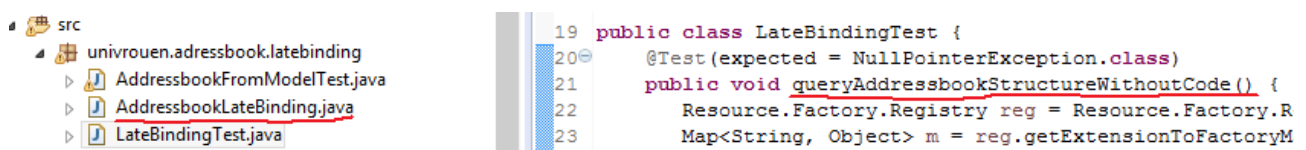
Création d'un projet EMF vide, ajout des dépendances dans le fichier MANIFEST.MF, création du package univrouen.adressbook.latebinding qui contient la creation de la classe AddressbookLateBinding .

Ainsi recopier le modèle Ecore « addressbook.ecore »



### Q IV.2 Analyse du méta-modèle Ecore

Création d'une nouvelle classe comportant la méthode de test nommée "queryAddressbookStructureWithoutCode()"



Compléter le code la classe et après la validation le résultat obtenu est la même que celui obtenu de la question (Q III.2)

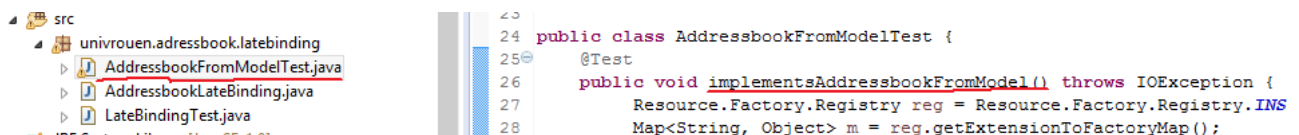
```

Adresse
    numero(EInt),  rue(EString),
Personne
    nom(EString),  prenom(EString),  age(EInt),  identifier(EString),
    Références : adresse(Adresse[1..1])
    Opérations : EString display,  EBoolean minAge,  EBoolean majNom,
AdressBook
    name(EString),
    Références : personne(Personne[0..-1])
    Opérations : EBoolean addContact,

```

## Q IV.3 Création d'une instance LateBinding

Création d'une nouvelle classe comportant la méthode de test nommée "implementsAdressbookFromModel()"



The screenshot shows the project structure on the left with 'src' expanded, showing 'univrouen.addressbook.latebinding' containing 'AddressbookFromModelTest.java', 'AddressbookLateBinding.java', and 'LateBindingTest.java'. The main window shows the code of 'AddressbookFromModelTest' with a test method 'implementsAddressbookFromModel()' that uses 'Resource.Factory.Registry' and 'Map'.

```

24 public class AddressbookFromModelTest {
25     @Test
26     public void implementsAddressbookFromModel() throws IOException {
27         Resource.Factory.Registry reg = Resource.Factory.Registry.INS
28         Map<String, Object> m = reg.getExtensionToFactoryMap();

```

L'ajout de deux Personnes

```

// recupere la liste des personnes
EcoreEList listPersonnes = (EcoreEList)addressBookImpl.eGet(ePersonneClass2);

// recuperer la premiere personne
DynamicEObjectImpl personneImpl1 = (DynamicEObjectImpl)listPersonnes.get(0);
EClass personne1 = personneImpl1.eClass();
EAttribute nom2 = (EAttribute)personne1.getEStructuralFeature("nom");
EAttribute prenom2 = (EAttribute)personne1.getEStructuralFeature("prenom");

// tester la premiere personne
Assert.assertEquals("NOM1", personneImpl1.eGet(nom2));
Assert.assertEquals("PRENOM1", personneImpl1.eGet(prenom2));

// recuperer la deuxieme personne
DynamicEObjectImpl personneImpl2 = (DynamicEObjectImpl)listPersonnes.get(1);
EClass personne = personneImpl2.eClass();
EAttribute nom3 = (EAttribute)personne.getEStructuralFeature("nom");
EAttribute prenom3 = (EAttribute)personne.getEStructuralFeature("prenom");

// tester la deuxieme personne
Assert.assertEquals("NOM2", personneImpl2.eGet(nom3));
Assert.assertEquals("PRENOM2", personneImpl2.eGet(prenom3));

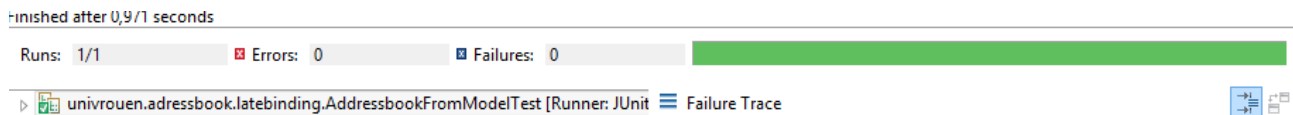
```

Ajout des Assert pour les tests

```
// tester la premiere personne
Assert.assertEquals("NOM1", personneImpl1.eGet(nom2));
Assert.assertEquals("PRENOM1", personneImpl1.eGet(prenom2));

// tester la deuxieme personne
Assert.assertEquals("NOM2", personneImpl2.eGet(nom3));
Assert.assertEquals("PRENOM2", personneImpl2.eGet(prenom3));
```

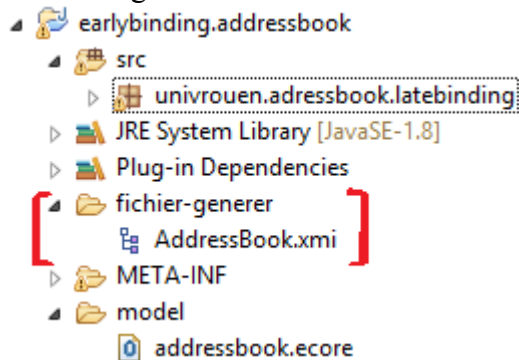
Après la validation des Assert



## Q IV.4 Persistance des données

Validation de format et sauvegarde des données

Le fichier généré nommé AddressBook.xmi dans la dossier fichier-generer



Le contenu du fichier AddressBook.xmi quand on l'ouvert avec Sample Reflective Ecore Model Editor

