

## **UML TP 2 :**

### **Implémentation d'un projet ECORE Contraintes** **OCL**

GUETTOUCHE Islam

FEGHOUL Ghiles

Master 1 Génie de l'informatique logicielle (G.I.L)

## Table des matières

Introduction :	3
Objectifs de ce TP :	3
Présentation des technologies utilisées:	3
Le schéma UML du modèle de l'application créée lors du TP précédent:	4
Insertion de contrainte OCL :	4
Paramétrage du modèle:	4
Insertion des contraintes OCL :	4
Simulation des instances:	5
Validation OCL:	6
Méta-Model Ecore:	8
Analyse du méta-modèle Ecore:	8
Affichage du contenu :	8
Manipulation du modèle Ecore :	10
Analyse du méta-modèle Ecore :	10
Référence :	11

## Introduction :

### Objectifs de ce TP :

- ✓ Créer des instances du modèle défini par son schéma UML (instanciation des classes générées, réflexivité).
- ✓ Manipuler les données (objets) produits par le modèle (persistance des données).
- ✓ Manipuler la structure du modèle Ecore.
- ✓ Insertion de contrainte OCL.

### Présentation des technologies utilisées:






- ✓ Eclipse Modeling Framework (EMF) :

Est un framework de modélisation, Le cœur d'EMF contient le framework de modélisation ainsi que l'infrastructure de génération de code et de manipulation des modèles EMF. Tout modèle EMF est une instance d'un modèle EMF avec pour racine commune le modèle Ecore fourni par EMF. EMF permet non seulement de créer un méta-modèle représentant les concepts désirés par l'utilisateur mais il permet ensuite à l'utilisateur de créer des modèles issus de ce méta-modèle et de les manipuler avec un outillage adapté.

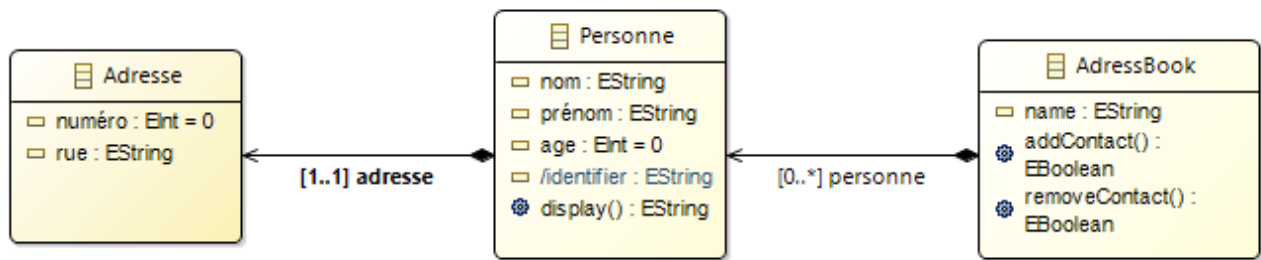
- ✓ OCL :

Il s'agit d'un langage formel d'expression de contraintes bien adapté aux diagrammes d'UML, et en particulier au diagramme de classes.

OCL peut s'appliquer sur la plupart des diagrammes d'UML et permet de spécifier des contraintes sur l'état d'un objet ou d'un ensemble d'objets comme :

-  Des invariants sur des classes.
-  Des préconditions et des post conditions à l'exécution d'opérations :
  - Les préconditions doivent être vérifiées avant l'exécution.
  - Les post conditions doivent être vérifiées après l'exécution.
-  Des gardes sur des transitions de diagrammes d'états-transitions ou des messages de diagrammes d'interaction.
-  Des ensembles d'objets destinataires pour un envoi de message.
-  Des attributs dérivés, etc.

Le schéma UML du modèle de l'application créée lors du TP précédent:



## Insertion de contrainte OCL :

Paramétrage du modèle:

Apportez les modifications suivantes au projet :

- Accédez aux propriétés de "addressbook.genmodel" (fenêtre centrale) et positionnez les paramètres suivants :

Model	
Model Plug-in Variables	OCL_ECORE=org.eclipse.oc1.ecore
Template & Merge	
Dynamic Templates	true
Template Directory	{workspace}/{package-projet}/templates

- Modifiez les propriétés du projet : menu window → preferences → ocl  
realisation of OCL embedded within ECORE models  
generate Java Code in \*Impl classes

## Insertion des contraintes OCL :

Ajoutez les contraintes suivantes à la classe Personne :

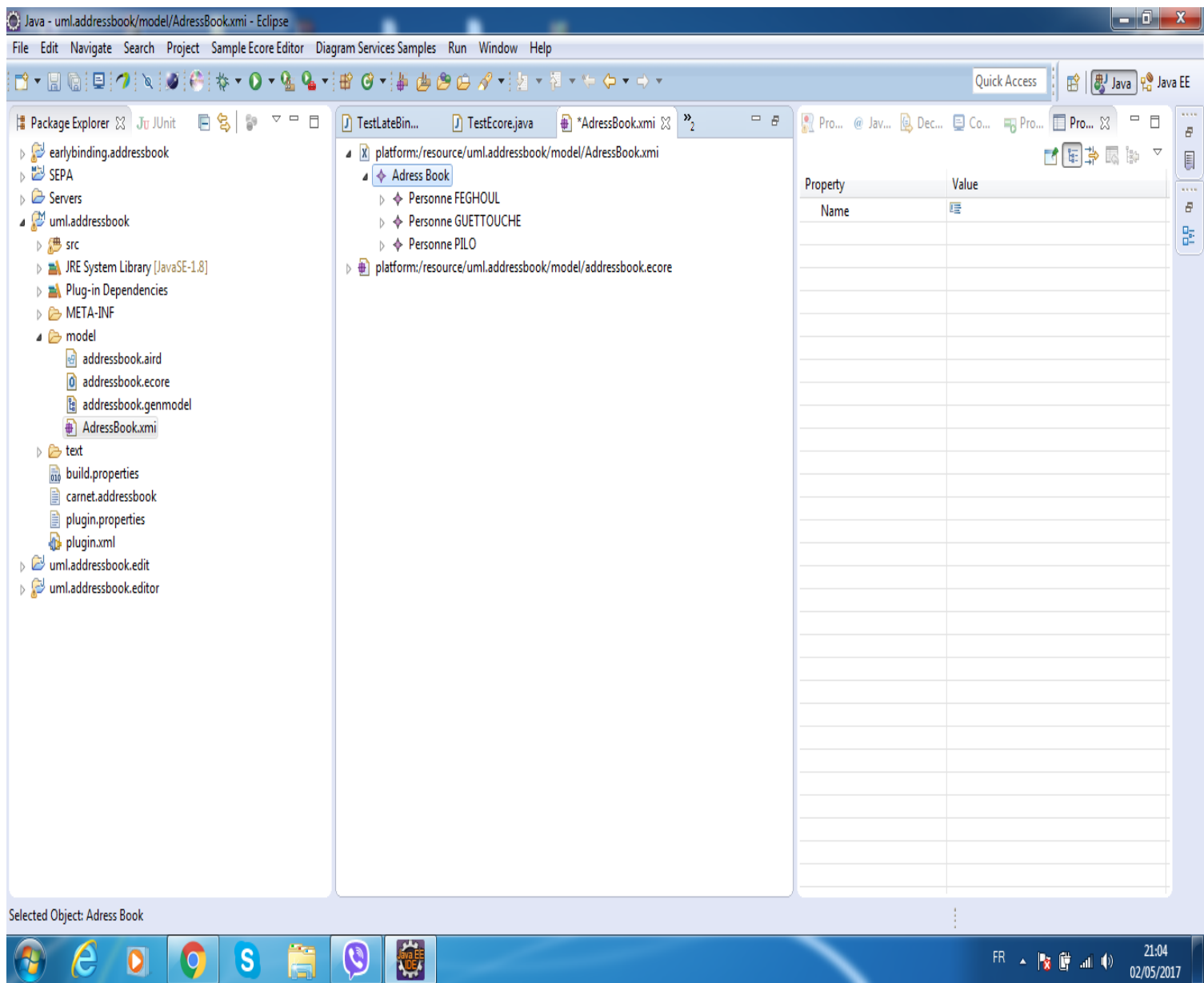
- ✓ La propriété âge doit être supérieure à 16.
- ✓ La propriété nom doit être écrite en majuscules.

```
class Personne
{
    invariant minAge:
        self.age > 16;

    invariant uppName:
        self.nom = self.nom.toUpperCase();

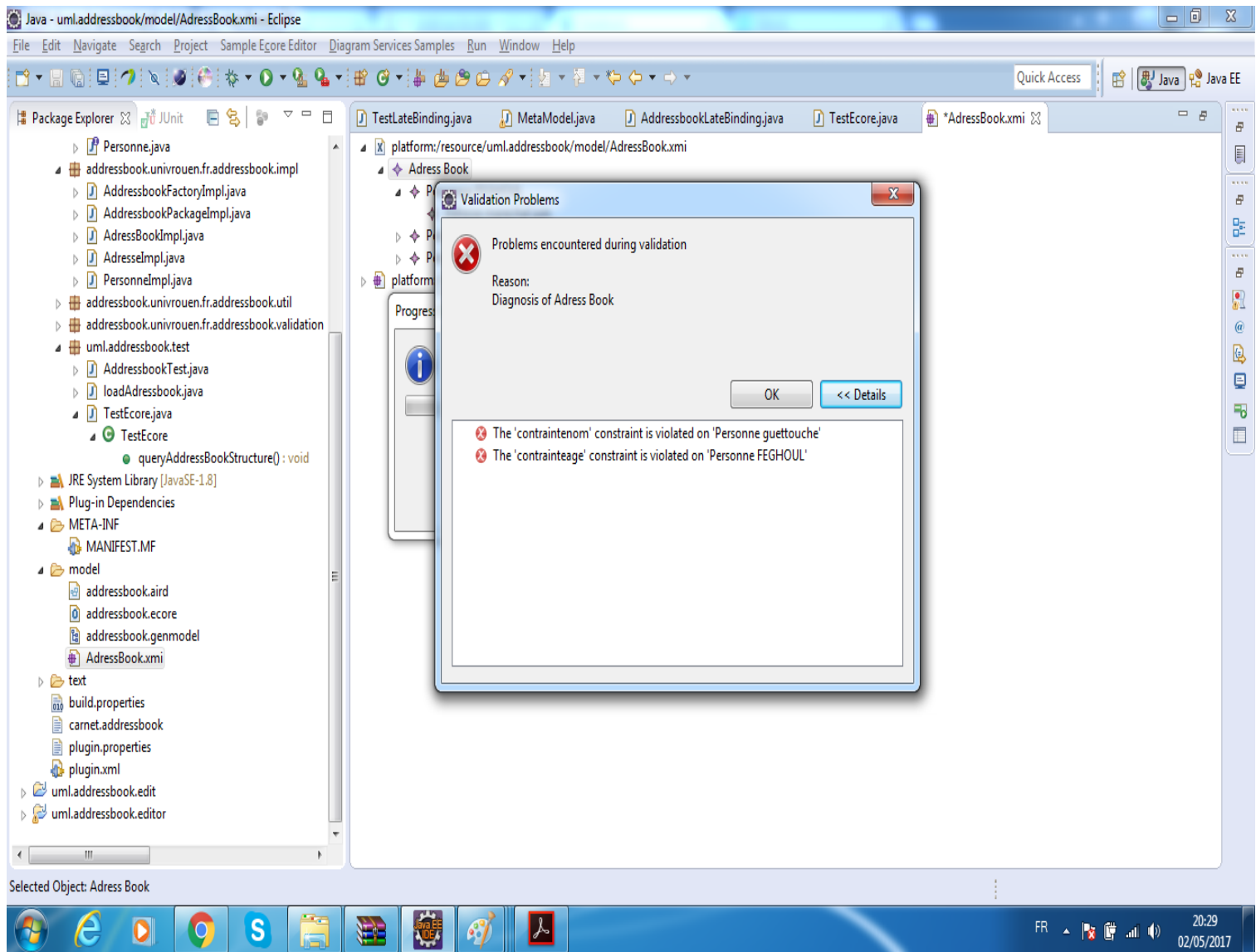
    operation display() : String[?];
    attribute nom : String[?];
    attribute prenom : String[?];
    attribute age : ecore::EInt[?];
    property adresse : Adresse[1] { composes };
    attribute identifiant : String[?] { derived readonly transient };
}
```

### Simulation des instances:

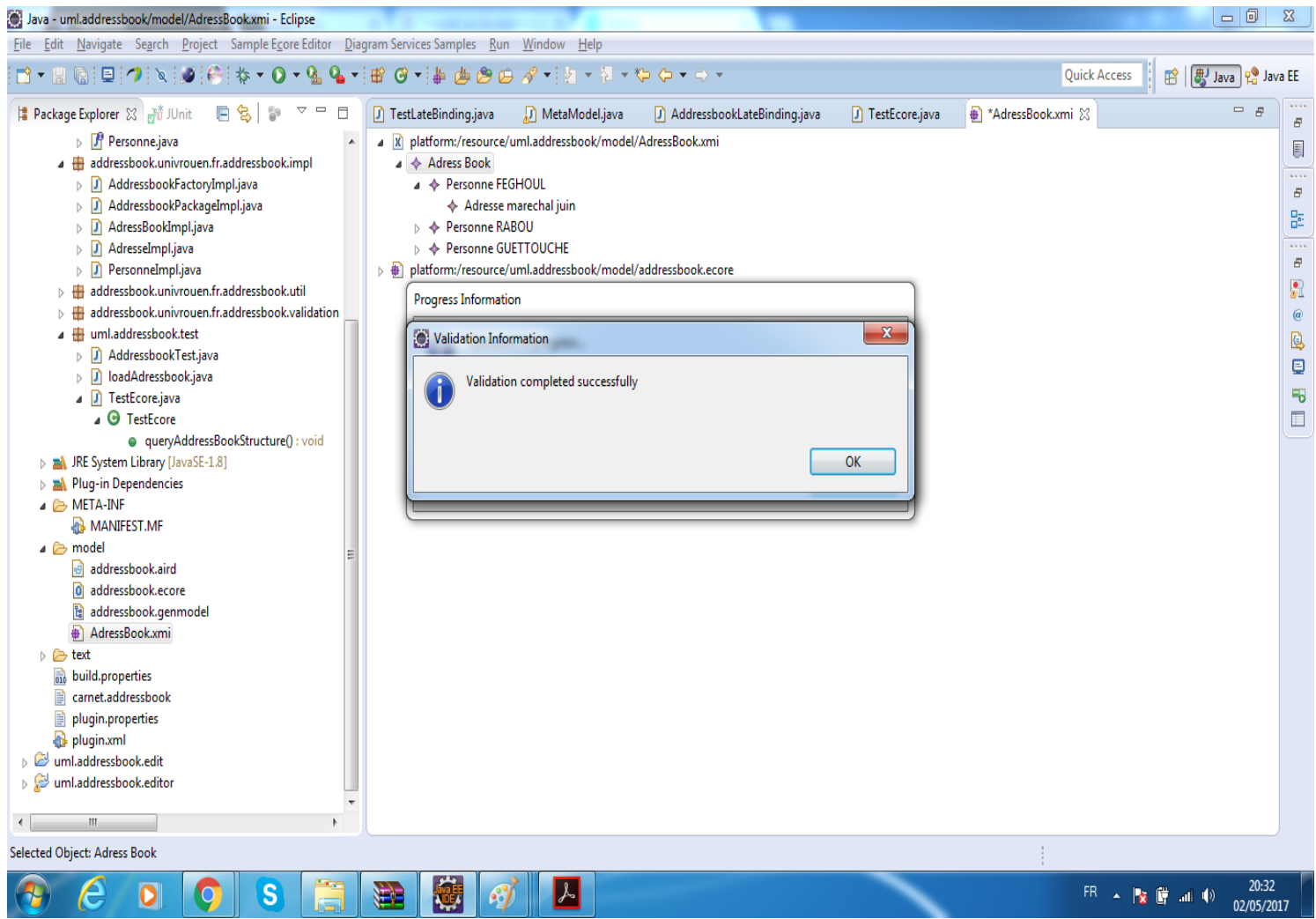


## Validation OCL:

L'image suivantes représente les différentes erreurs qui on était retournées après la premier validation :

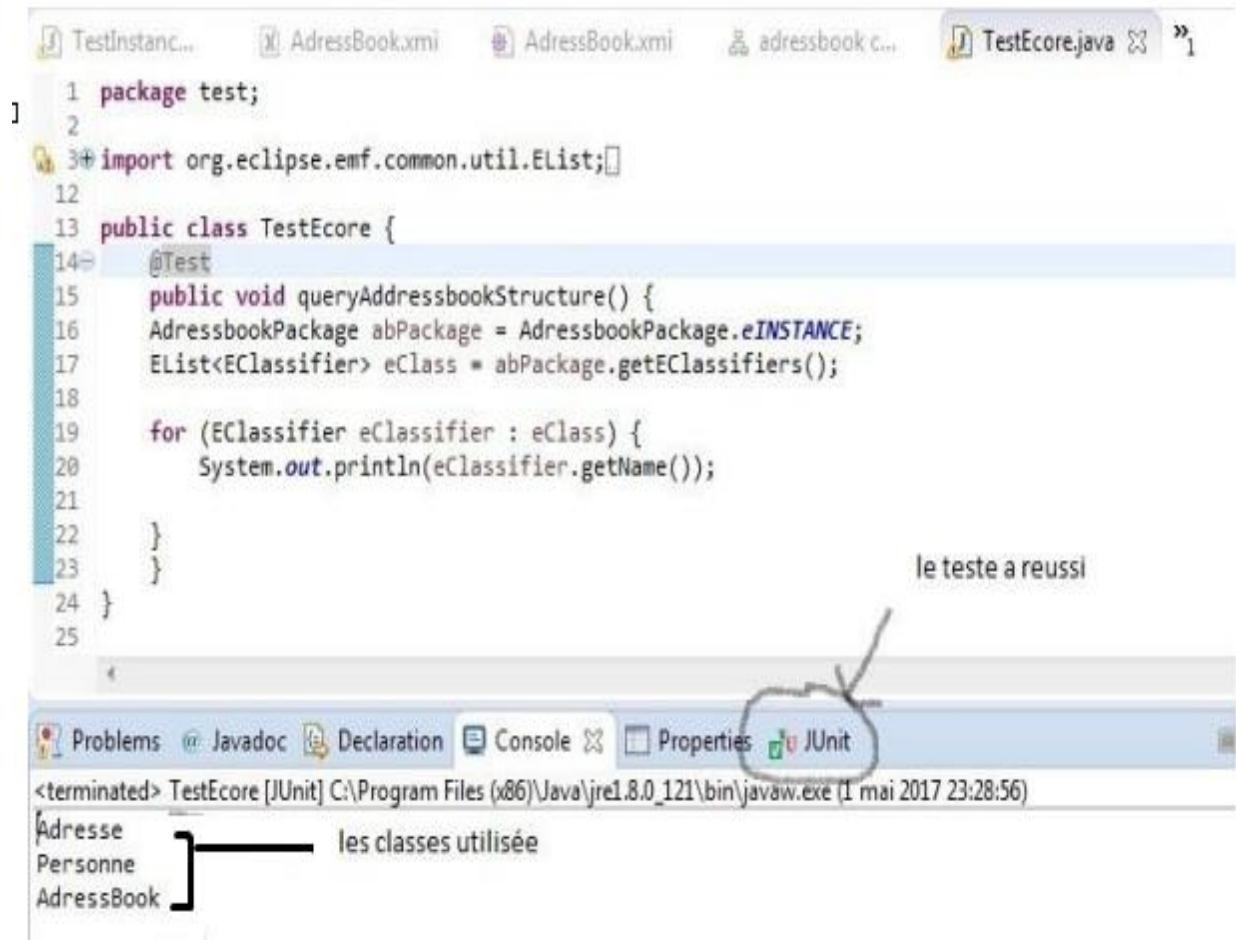


Après intervention (la gestion des erreurs) les contraintes on était valider avec succès:



## Méta-Model Ecore:

### Analyse du méta-modèle Ecore:



### Affichage du contenu :

L'image suivante représente les détails de chaque élément (format, attributs, opérations, ...) Après avoir compléter le code de la fonction fourni dans la classe `TestEcore.java` :



### Adresse

numero(EInt) rue(EString)

□

### Personne

nom(EString) prenom(EString) age(EInt) identifier(EString)

Références : adresse(Adresse[1..1])

Operation : EString display EBoolean minAge EBoolean uppName

### AdressBook

name(EString)

Références : personne(Personne[0..-1])

Operation : EBoolean addContact

The screenshot shows the Eclipse IDE interface. The main editor displays the source code of `TestEcore.java` in the `uml.addressbook.test` package. The code imports `org.eclipse.emf.common.util.EList` and defines a `TestEcore` class with a `@Test` annotated `queryAddressBookStructure()` method. This method iterates over `EClassifier` objects, printing their names and details. A tooltip for `org.eclipse.emf.ecore.EClass` is visible, listing supported features like `Abstract`, `Interface`, `ESuper Types`, `EOperations`, `EAll Attributes`, and `EAll References`.

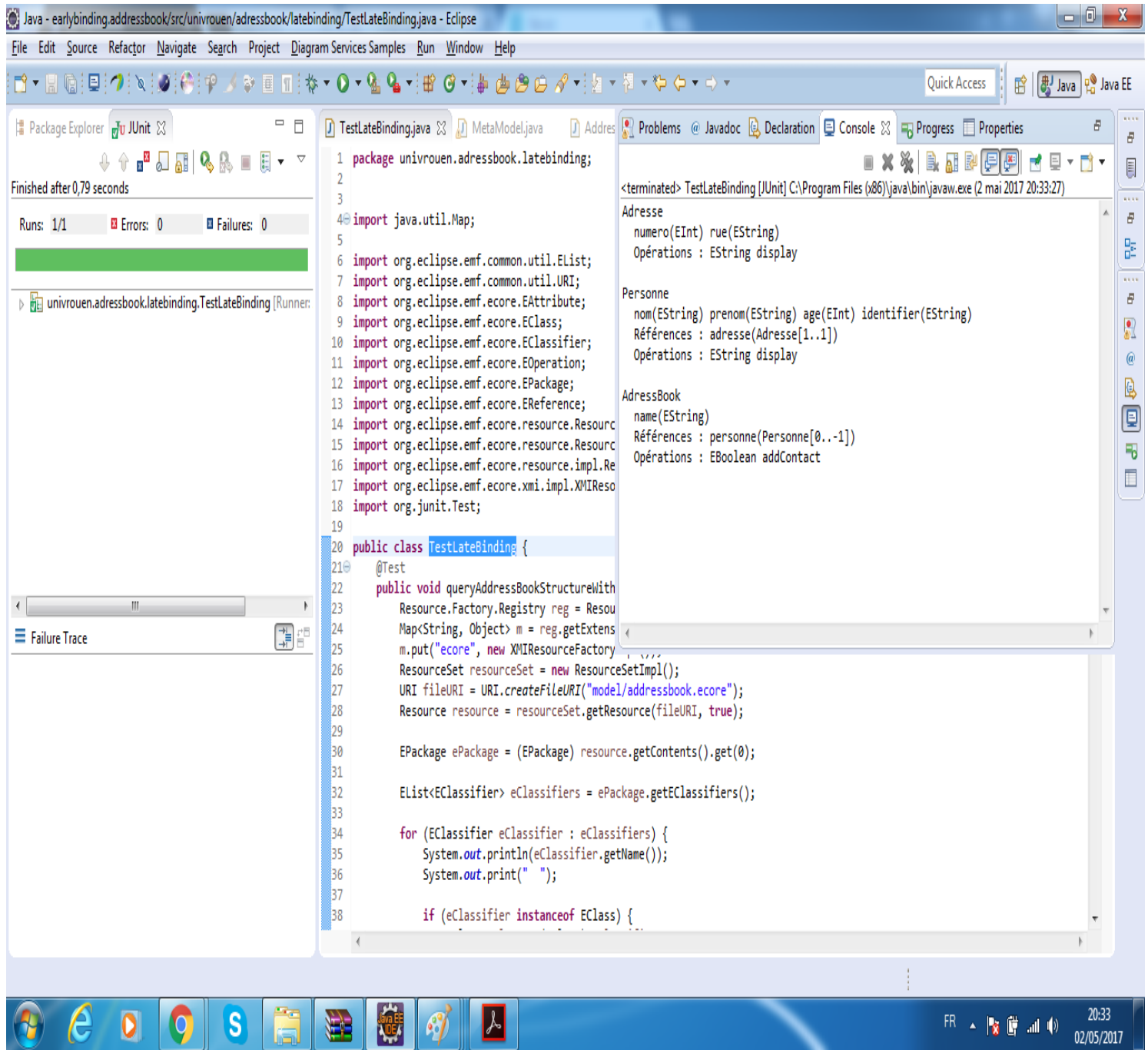
On the right, the UML diagram shows the `Adresse`, `Personne`, and `AdressBook` classes. `Adresse` has attributes `numero(EInt)` and `rue(EString)` and an operation `display`. `Personne` has attributes `nom(EString)`, `prenom(EString)`, `age(EInt)`, and `identifier(EString)`, and an operation `display`. It has a reference to `Adresse` with multiplicity `1..1`. `AdressBook` has an attribute `name(EString)` and a reference to `Personne` with multiplicity `0..-1`. It has an operation `addContact`.

The Package Explorer on the left shows the project structure, and the Run Console at the bottom indicates the test execution completed successfully after 0.937 seconds.

## Manipulation du modèle Ecore :

### Analyse du méta-modèle Ecore :

L'image suivante représente les résultats après avoir compléter la fonction fournie dans la classe TestLateBinding.java du projet earlybinding.addressbook :



## Référence :

On s'est inspiré dans ce mini projet sur :

<http://mbaron.developpez.com/tutoriels/eclipse/emf/creation-instanciation-modeles/>