PADROSA DANIEL **30/12/05**
Master 2 TOIL

Test de l'outil OCL Toolkit de Dresden.

Sommaire:

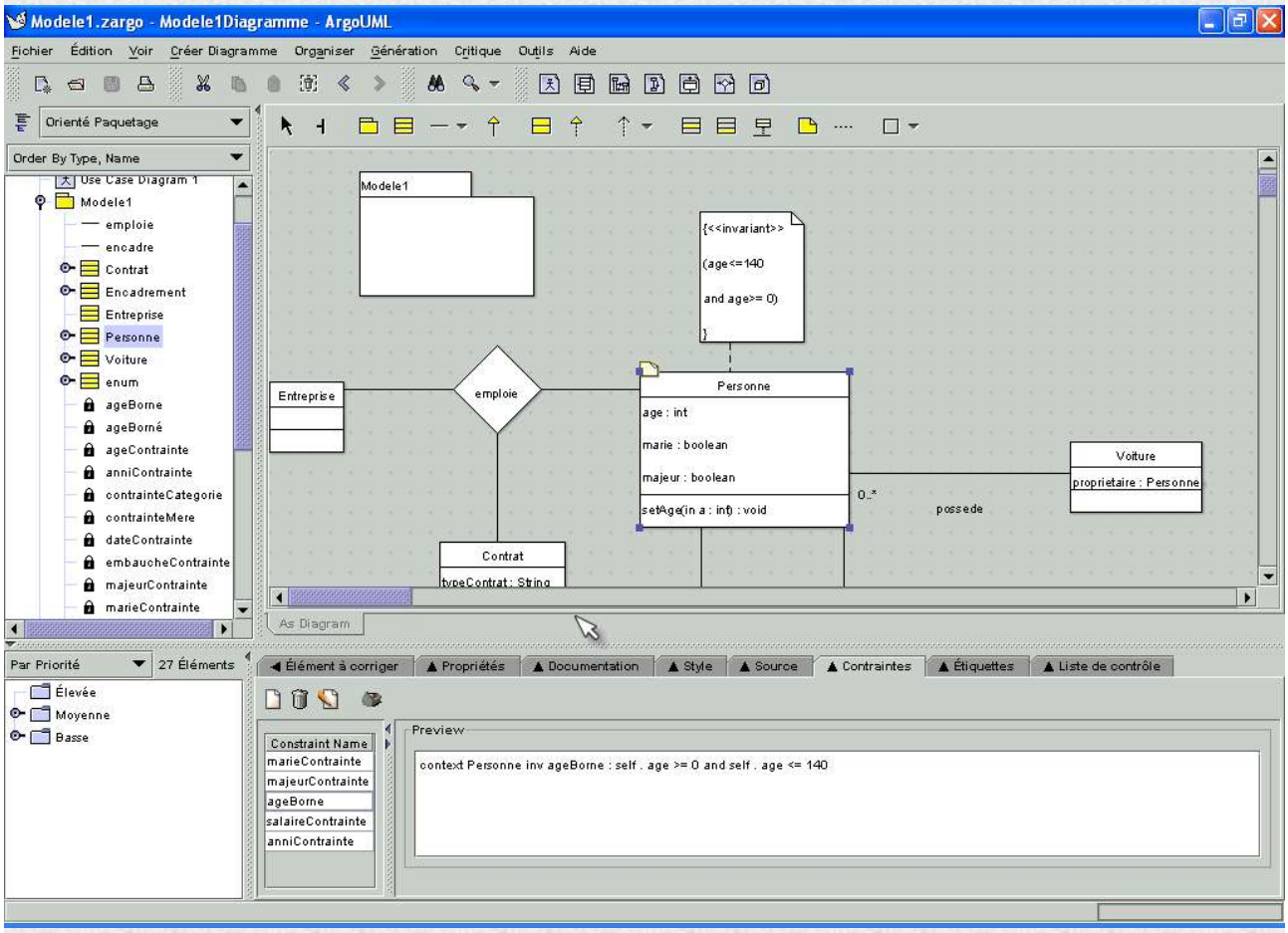
I-Intégration avec ArgoUML.

I-1 Introduction:

L'api java ArgoUML permet de créer des diagrammes de classes et spécificier des contraintes sur les modéles.Une fois les diagrammes de classes créés et les contraintes spécifiées ont peut générer du code java.
A partir du code généré , on peut avec l'api Toolkit de Dresden généré une implémentation des contraintes définies avec la syntaxe OCL.

Environnement ArgoUML.

Télèchargement du projet java argouml.jar sur le site http://argouml.tigris.org.



I-2 Création du Diagramme de classe :

-on créé le package Modelepackage comprenant:(voir figure 2)
- la classe Personne.
- La classe Etudiant.
- La classe Voyageur.
- la classe Entreprise.
- la classe Voiture.
- la classe Contrat.
- la classe Encadrement.

Les attributs de relations sont par défaults representés par des vecteurs qu'il faudra initialisés une fois le code généré.

I-3 Spécifications des contraintes OCL:

On peut spécifier des contraintes sur les classes ,sur les attributs et les methodes.

Exemple de spécifications sur une méthode:
Ici feteAnniversaire() a pour contraintes que l'age soit inférieur à 140 ans et que age soit augmenté de un an.

```
 /**
  * @postcondition contrainteAnniversaire: age < 140 and age = age @ pre + 1
  */
 public void feteAnniversaire() {
 }
```

Autre exemple avec la methode setAge()

```
/**
  * @precondition agecontrainteSet2: ageCorrect ( a ) and a >= age
  * @postcondition agecontrainteSet2: age = a
  */
 public void setAge(int a) {
 }
```

Ici l'attribut age a pour contrainte d'être positif et inférieur à 140.

```
  /**
  * @invariant newConstraint_0: self . age > 0 and self . age < 140
  */
 public  int age = 0;
```

I-3-1 Défauts et points obscurs sur l'outil ArgoUML et du Toolkit OCL de Dresden:

Le principal défaut de cette api est qu'on ne peut pas définir des fonctions en OCL.
On doit préalablement définir des méthodes en Java puis y appliquer des contraintes OCL
qui pourront être parsés par l'outil de Dresden pour généré une implémentation de ces contraintes.

Ce défaut est principalement lié à ArgoUML qui ne génére pas les methodes décrit en OCL.
Ce défauts vient il précisement d'ArgoUML? Au quel cas ,on peut se poser la question si l'outil de Dresden permet de créer des fonctions? Je n'ai rien trouvé pour l'instant.
De même , le type Enum n'existe pas avec ArgoUML mais le parseur de Dresden reconnaît les erreurs sur le type Enum, Question Ouverte.
Malgré ces points obscurs la documentation de Dresden recommande ArgoUML pour l'intégration des contraintes.

I-3-2 Avantages de l'outil ArgoUML et du Toolkit OCL de Dresden:

Le principal avantage de cette api est dans la génération de code en Java qui permet une implémentation efficace des contraintes OCL sur les modéles.

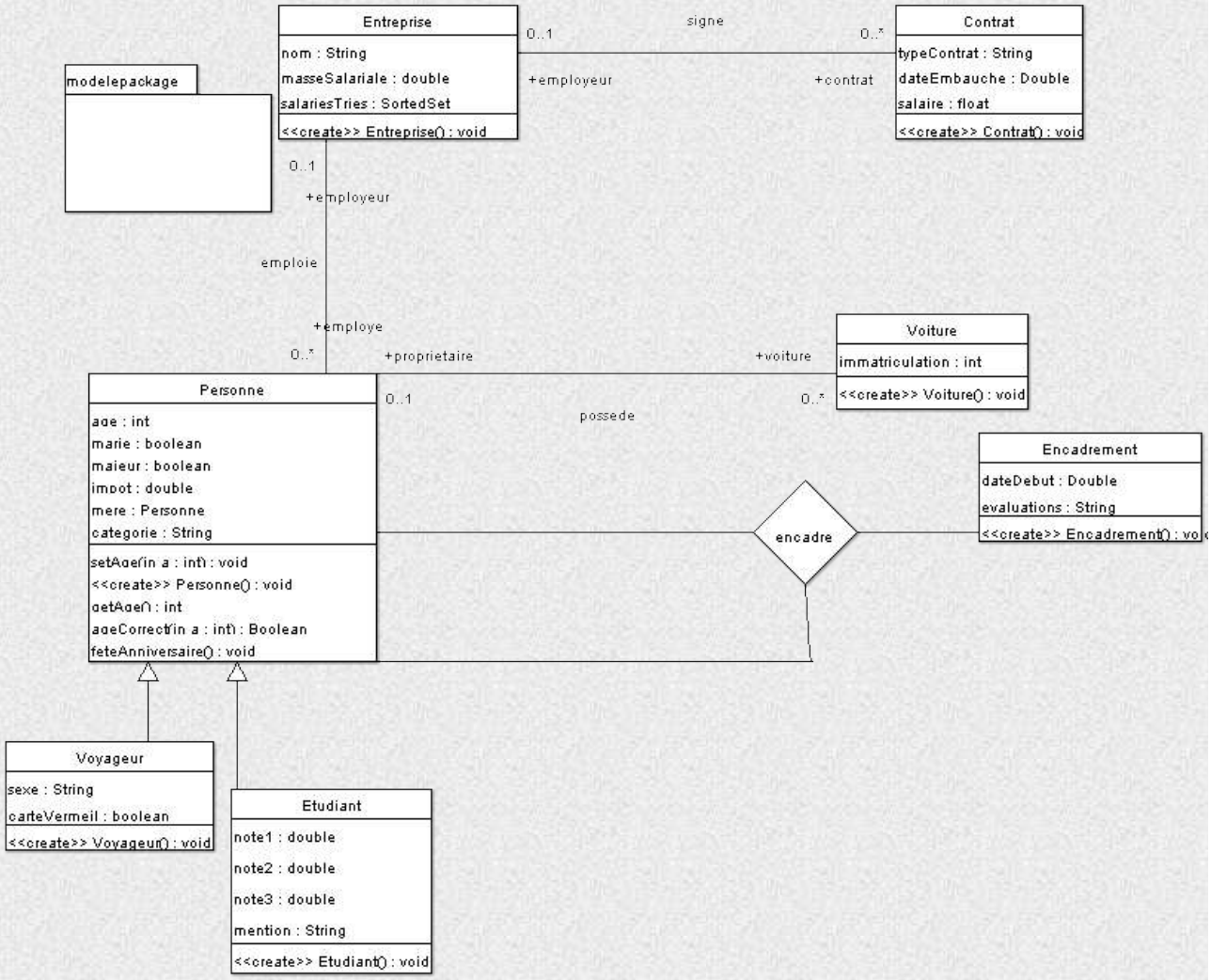


Figure 2-Diagramme de classes de modelepackage généré avec ArgoUML.

I-4 Génération de code avec ArgoUML.

I-4-1 Principe:

ArgoUML permet de faire de la génération de code java en spécifiant les contraintes en commentaires.Ces commentaires pourront être parsés par le parseur OCL de Dresden afin de générer des fichiers java.injected.Ces fichiers une fois recompilés par le compilateur javac donne des classes permettant une implémentation des contraintes OCL.

Exemple avec le fichier Personne.java généré avec ArgoUML.

```
package modelepackage;

import java.util.Vector;


  /**
   * @invariant ageContrainte_: marie implies majeur
   * @invariant majeurContrainte: if age >= 18 then majeur = true else majeur = false endif
   * @precondition anniContrainte: age < 140
   * @postcondition anniContrainte_1: age = age @ pre + 1
   * @invariant ageContrainte2: self . age > 0 and self . age < 140
   * @invariant mariecontrainte: marie implies majeur
   * @invariant majeurboolean: majeur = ( age >= 18 )
   * @invariant salaireContrainte: self . contrat . salaire >= 0
   * @precondition anniversairecontrainte: age < 140
   * @postcondition anniversairecontrainte_1: age = age @ pre + 1
   * @invariant contrainteLet: let montantImposable : Real = contrat . salaire * 0.8 in if
( montantImposable >= 100000 ) then impot = montantImposable * 45 / 100 else if
( montantImposable >= 50000 ) then impot = montantImposable * 30 / 100 else impot =
montantImposable * 10 / 100 endif endif
   * @invariant ageContrainte3: ageCorrect ( age )
   * @invariant merecontrainte: self . mere <> self and self . age < self . mere . age
   * @postcondition getAgeContrainte: result = age
   */
public class Personne {
  /* {src_lang=Java}*/


    /**
   *
   * @invariant newConstraint_0: self . age > 0 and self . age < 140
   */
  public final int age = 0;
  /* {transient=false, volatile=false}*/

  public boolean marie;
  /* {transient=false, volatile=false}*/

  public boolean majeur;
```

```
 /* {transient=false, volatile=false}*/

 public double impot;
 /* {transient=false, volatile=false}*/

 public Personne mere;
 /* {transient=false, volatile=false}*/

  /**
  * @invariant newConstraint_0: if age <= 12 then categorie = 'enfant' else if age <= 18 then
categorie = 'ado' else categorie = 'adulte' endif endif
  */
 public String categorie;
 /* {transient=false, volatile=false}*/

  public Personne agentSecretariat;
  public Encadrement encadrement;
  public Personne responsable;

  /**
  *
  * @element-type Voiture
  */
 public Vector  voiture;
  public Entreprise employeur;
  public Contrat contrat;

  /**
  * @precondition agecontrainteSet2: ageCorrect ( a ) and a >= age
  * @postcondition agecontrainteSet2: age = a
  */
 public void setAge(int a) {
 }

 public Personne() {
 }

  /**
  * @postcondition getAgeContrainte: result = age
  */
 public int getAge() {
 return 0;
 }

  /**
  * @postcondition ageCorrectContrainte: a >= 0 and a <= 140
  */
 public Boolean ageCorrect(int a) {
 return null;
 }

  /**
  * @postcondition contrainteAnniversaire: age < 140 and age = age @ pre + 1
```

```
  */
  public void feteAnniversaire() {
  }
}
```

I-4-2 Modification du code:
Aprés création du code, on modifie les divers attributs par exemple initialisation des vecteurs.

II-Génération de code avec le parseur OCL de Dresden:
II-1 Principe :
On fabrique les fichiers avec l'intégration des contraintes OCL dans le code java.
Pour cela on génère d'abord les classes des fichiers parsés par ArgoUML.
Puis on fabrique les fichiers injectés avec la commande:
java tudresden/ocl/injection/ocl/Main -m -r nomPackage nomPackage/*.java.
L'option -m permet d'ecraser les fichiers java générés par argoUML par les fichiers parsés en Java par l'outil de Dresden.
Sinon les fichiers sont sauvegardés dans des fichiers java.injected.
Une fois les fichiers parsés , il faut les recompilés avec javac.

Exemple de contrainte OCL implémentée en Java avec l'outil de Dresden:

Contrainte Ocl définie avec ArgoUML:

```
 /**
  * @invariant majeurboolean: majeur = ( age >= 18 )
*/
*
```

Contrainte implémentée en Java avec le parseur OCL Toolkit de Dresden:

```
/
**
An object representing ocl invariant majeurboolean on this object. Generated automatically, DO NOT CHANGE!
@author ocl_injector
*/private final tudresden.ocl.injection.ocl.lib.Invariant zzzCheckOclInvariantObject812374_majeurboolean=new tudresden.ocl.injection.ocl.lib.Invariant ("majeurboolean", this);/**
Checks ocl invariant majeurboolean on this object. Generated automatically, DO NOT CHANGE!
@author ocl_injector
*/public final void zzzCheckOclInvariantMethod812374_majeurboolean(){
tudresden.ocl.lib.OclAnyImpl.setFeatureListener (zzzCheckOclInvariantObject812374_majeurboolean);
final tudresden.ocl.lib.OclAnyImpl tudOclNode52=tudresden.ocl.lib.Ocl.toOclAnyImpl ( tudresden.ocl.lib.Ocl.getFor(this) );
final tudresden.ocl.lib.OclBoolean tudOclNode53=tudresden.ocl.lib.Ocl.toOclBoolean (tudOclNode52.getFeature("majeur"));
final tudresden.ocl.lib.OclInteger tudOclNode54=tudresden.ocl.lib.Ocl.toOclInteger (tudOclNode52.getFeature("age"));
final tudresden.ocl.lib.OclInteger tudOclNode55=new tudresden.ocl.lib.OclInteger(18);
final tudresden.ocl.lib.OclBoolean tudOclNode56=tudOclNode54.isGreaterEqual(tudOclNode55);
final tudresden.ocl.lib.OclBoolean tudOclNode57=tudOclNode53.isEqualTo(tudOclNode56);
tudresden.ocl.lib.OclAnyImpl.clearFeatureListener();
if(!tudOclNode57.isTrue())System.out.println("violated ocl invariant 'majeurboolean' on object '"+this+"'.");}
```

II-2 Phases de Tests:
Exemple avec la contrainte sur l'age:

```
/**
   * @invariant ageContrainte: self . age >= 0 and self . age < 140
   */
```

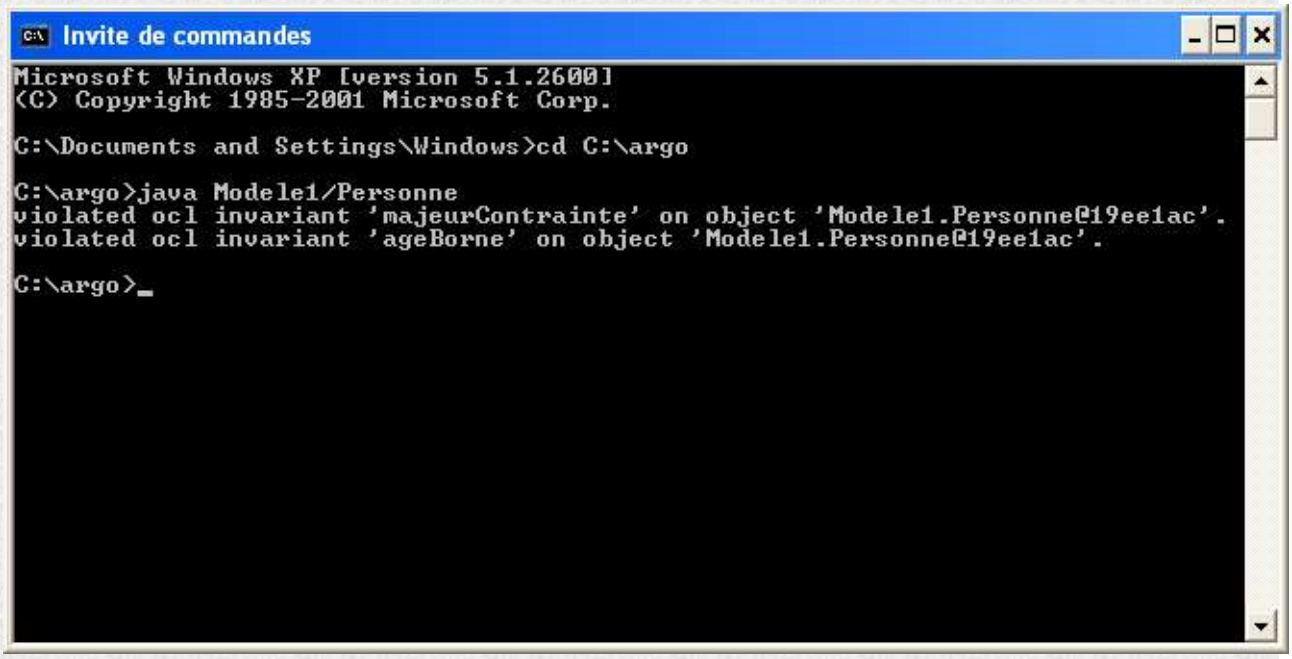
Contrainte générée avec Dresden:

```
/**
An object representing ocl invariant ageContrainte on this object. Generated automatically, DO
NOT CHANGE!
@author ocl_injector
*/private final tudresden.ocl.injection.ocl.lib.Invariant
zzzCheckOclInvariantObject812374_ageContrainte=new tudresden.ocl.injection.ocl.lib.Invariant
("ageContrainte", this);/**
Checks ocl invariant ageContarinte on this object. Generated automatically, DO NOT CHANGE!
@author ocl_injector
*/public final void zzzCheckOclInvariantMethod812374_ageContarinte(){
tudresden.ocl.lib.OclAnyImpl.setFeatureListener
(zzzCheckOclInvariantObject812374_ageContarinte);
final tudresden.ocl.lib.OclAnyImpl tudOclNode33=tudresden.ocl.lib.Ocl.toOclAnyImpl
( tudresden.ocl.lib.Ocl.getFor(this) );
final tudresden.ocl.lib.OclInteger tudOclNode34=tudresden.ocl.lib.Ocl.toOclInteger
(tudOclNode33.getFeature("age"));
final tudresden.ocl.lib.OclInteger tudOclNode35=new tudresden.ocl.lib.OclInteger(0);
final tudresden.ocl.lib.OclBoolean tudOclNode36=tudOclNode34.isGreaterEqual(tudOclNode35);
final tudresden.ocl.lib.OclInteger tudOclNode37=tudresden.ocl.lib.Ocl.toOclInteger
(tudOclNode33.getFeature("age"));
final tudresden.ocl.lib.OclInteger tudOclNode38=new tudresden.ocl.lib.OclInteger(140);
final tudresden.ocl.lib.OclBoolean tudOclNode39=tudOclNode37.isLessThan(tudOclNode38);
final tudresden.ocl.lib.OclBoolean tudOclNode40=tudOclNode36.and(tudOclNode39);
tudresden.ocl.lib.OclAnyImpl.clearFeatureListener();
if(!tudOclNode40.isTrue())System.out.println("violated ocl invariant 'ageContrainte' on object
'"+this+"'.");
}
```

on rajoute la fonction main:

```
public static void main(String[]args){
Personne p=new Personne();
p.setAge(152);
}
```

On voit que cela léve une contrainte OCL, l'âge d'une personne doit être infèrieure à 140 ans.



II-2-1 Exemple de Mirabelle Nebut avec la gestion de comptes en banque.

Petit DAB créé avec Dresden.

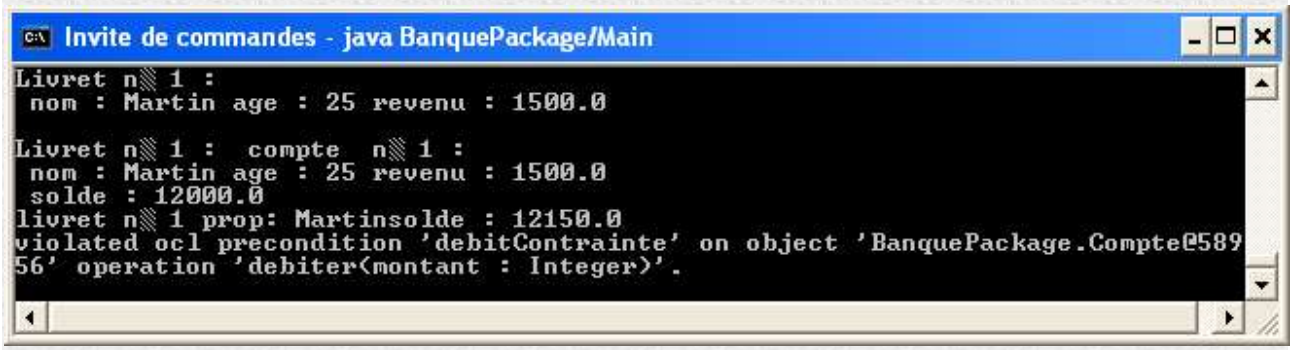
-Création d'un livret puis création d'un compte.
-Possibilité de consulter un livret.
-Possibilité de débiter ou créditer le compte d'un livret.

Création d'un livret:
Il faut spécifier le nom ,l'age ,un montant et un revenu valide.

Création d'un ou plusieurs compte:
Une fois un livret créé , on peut créé dessus un ou plusieurs comptes.
Il suffit de rajouter le numero de livret.

Opération sur les comptes de livret:
Une fois le numero de livret spécifié ainsi que le numero de compte et le nom du proprietaire.
On peut consulter le compte du livret ou bien debiter ou créditer le compte.

Exemple de débit sur le compte n°1 du livret n°1 du proprietaire Martin.



Génération d'une erreur OCL sur l'operation de débit.

III-Conclusion.

L'outil de Dresden est très efficace pour la génèration de code en Java et l'impléméntation des contraintes OCL en Java.
Mais il présente des défauts sur la possibilité de spécifier des contraintes OCL,il limite l'utilisation de la syntaxe OCL, mais on peut en contre-partie spécifier des contraintes sur les attributs ,les méthodes définies préalablement dans le modèle que l'on veut implémenter.
Vous trouverez ci-joint les problêmes rencontrés avec les exemples du cours mais aussi une implémentation efficace en Java de l'exemple de Mirabelle Nebut sur la gestion d'un compte en banque.