

De la conception technique UML vers la génération Java

Marie-Pierre Gervais
L3 Miage
EC MPRO

La génération de code


- Fonction disponible dans la plupart des AGL UML
- Applicable essentiellement au diagramme de classes
- Dans certains AGL, applicable aussi au diagramme d'états
- Pas utilisable de la même façon selon les AGL
- Présentation pour Modelio

UML (Modelio) -> Java


- Ensemble de règles de traduction
- Pour 1 concept UML (un élément de modèle UML), la règle fait correspondre 1 concept Java (un élément de programme Java), quelquefois plusieurs...

Règle de traduction UML2Java pour une CLASSE

- 1 classe UML =
 - 1 classe java et 1 fichier .java

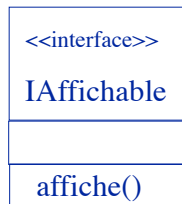
	Personne.java public class Personne { ... }
---------------------------------------------------------------------------------------	---------------------------------------------------

- 1 classe abstraite UML =
 - 1 classe abstraite java et 1 fichier .java

	Personne.java abstract public class Personne { ... }
---------------------------------------------------------------------------------------	------------------------------------------------------------

Règle de traduction UML2Java pour une INTERFACE

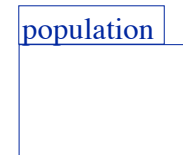
- 1 interface UML =
 - 1 interface java



```
public interface IAffichable {
    void affiche() ;
}
```

Règle de traduction UML2Java pour un PAQUETAGE

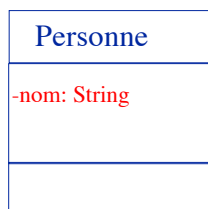
- 1 paquetage UML =
 - 1 paquetage java



```
package population ;
```

Règle de traduction UML2Java pour un ATTRIBUT

- 1 attribut UML = 1 attribut java
 - Type
 - Java ou classe Java correspondant à la trad' d'une classe UML
 - Visibilité
 - + public, # protected, - private (~ package en UML, non traduit)
 - Accès
 - Read, readwrite, write, no access => pour la génération des accesseurs
 - Initialisation de la valeur

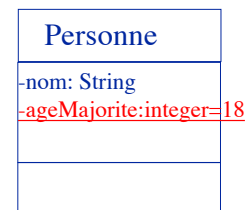


```
Personne.java
```

```
public class Personne {
    private String nom;
}
```

Règle de traduction UML2Java pour un ATTRIBUT DE CLASSE

- 1 attribut de classe (UML)
 - 1 attribut statique (java)

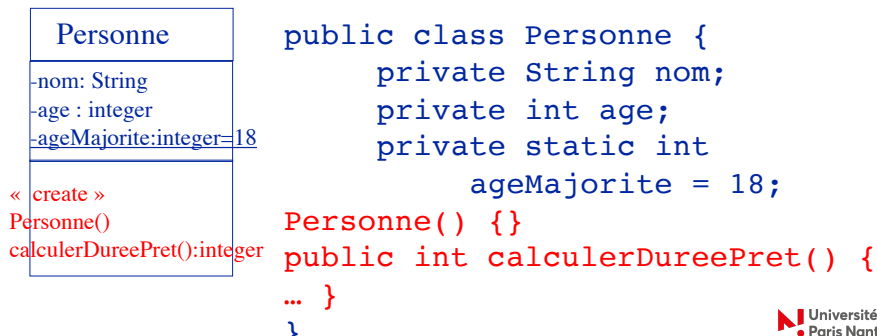


```
Personne.java
```

```
public class Personne {
    private String nom;
    private static int
        ageMajorite = 18;
}
```

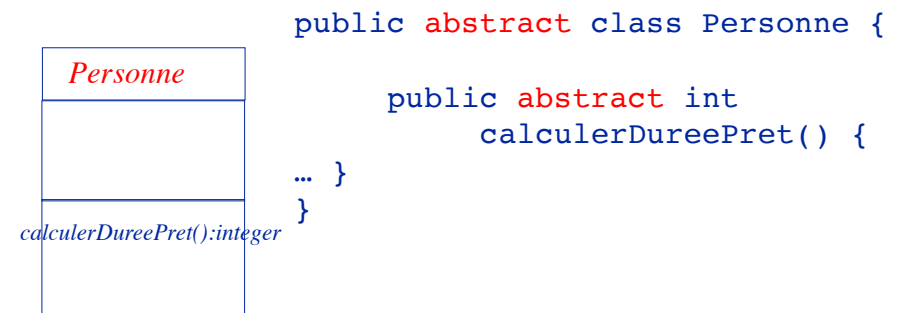
Règle de traduction UML2Java pour une OPÉRATION

- 1 opération UML = 1 signature de méthode java
 - Visibilité et type (des paramètres) **EN MODE IN UNIQUEMENT**
 - *Idem attribut*
 - Stéréotype « create » pour le constructeur



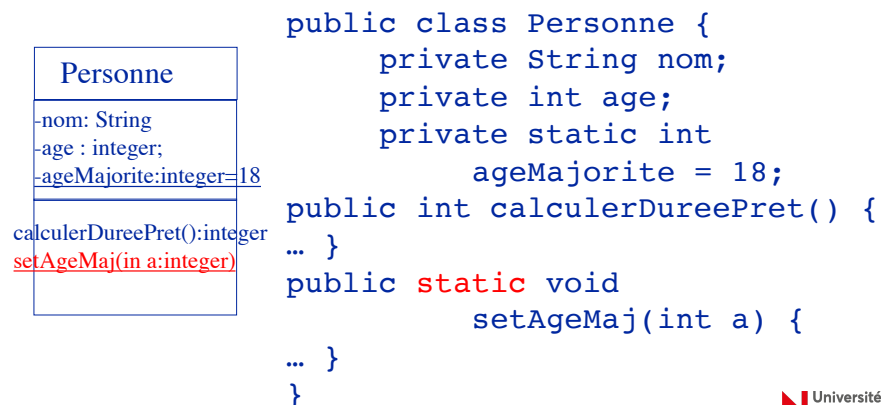
Règle de traduction UML2Java pour une OPÉRATION ABSTRAITE

- 1 opération qui est dans une classe abstraite (UML)
 - 1 méthode abstraite (java)



Règle de traduction UML2Java pour une OPÉRATION DE CLASSE

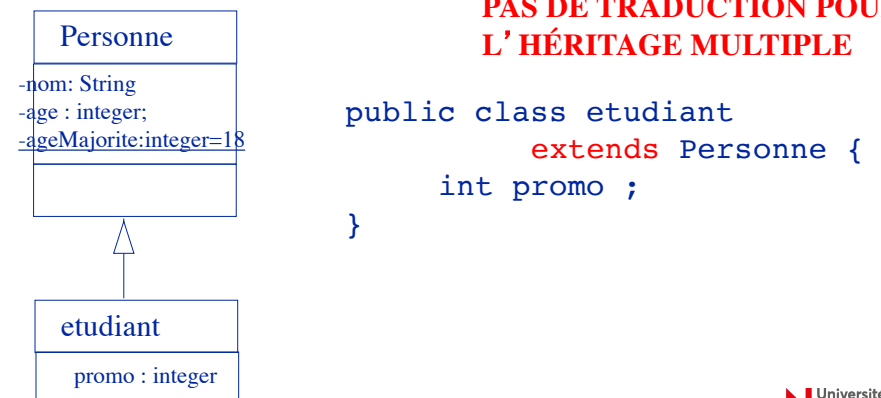
- 1 opération de classe (UML)
 - 1 méthode statique (java)



Règle de traduction UML2Java pour une Relation de généralisation

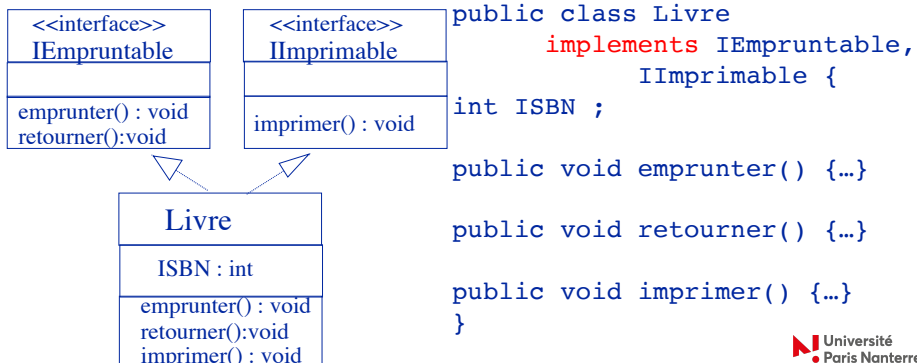
- Une généralisation/spécialisation UML =
 - Un héritage Java

**PAS DE TRADUCTION POUR
L'HÉRITAGE MULTIPLE**



Règle de traduction UML2Java pour une Relation de réalisation

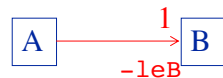
- Une classe UML réalise une ou +sieurs interface(s) UML =
 - Une classe java « implements » une ou +sieurs interface(s) java



Règle de traduction UML2Java pour une Association unidirectionnelle (1)

- À 1 association UML :
 - Navigable dans 1 sens
 - => **IL FAUT positionner l'indicateur de navigation**
 - Qui a une multiplicité de 1 sur la cible
 - => **IL FAUT positionner la multiplicité**
 - Qui a un nom de rôle leB sur l'extrémité d'association cible
 - => **IL FAUT positionner le nom de rôle**
- Correspond
 - 1 attribut d'instance privé (java) dans la classe source typé par la classe cible et ayant pour nom le nom de rôle de l'extrémité de l'association côté cible

Règle de traduction UML2Java pour une Association unidirectionnelle (2)



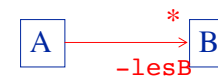
```

public class A {
    private B leB;
    ... }
    
```

- Pour déterminer la visibilité de l'attribut référence (ici, leB)
 - **IL FAUT positionner la visibilité de l'extrémité d'association**
- Pour déterminer les accesseurs à générer,
 - **IL FAUT positionner le mode d'accès**

Règle de traduction UML2Java pour une Association unidirectionnelle (3)

- Et si la multiplicité est * ?
- Alors :
 - = 1 attribut d'instance privé (java) dans la classe source de type **collection d'objets** par défaut, la collection est une liste, mais on peut choisir sa catégorie de collection

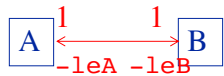


```

public class A {
    private List<B> lesB =
        new ArrayList<B> ();
    ... }
    
```

Règle de traduction UML2Java pour une Association bidirectionnelle

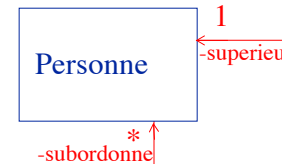
- 1 association UML navigable dans les 2 sens
 - Une paire de références (une dans chaque classe impliquée dans l'association)



```
public class A {  
    private B leB ;  
    ... }  
  
public class B {  
    private A leA ;  
    ... }
```

Règle de traduction UML2Java pour une Association réflexive

- 1 association réflexive UML =
 - Une référence sur un objet de la même classe



```
public class Personne {  
    private Personne superieur;  
    private Personne  
        subordonne[];  
}
```

*Cas où la collection
choisie est un tableau*

Règle de traduction UML2Java pour une Association agrégation

- 1 agrégation UML
 - Association non symétrique exprimant une relation de contenance
 - L'association ne peut contenir de marque d'agrégation qu'à une seule de ses extrémités
- = même règle de traduction en Java que pour une association simple

Correspondances UML2Java Association composition

- 1 composition UML
 - 1 agrégation pour laquelle
 - Une partie n'appartient qu'à un seul composite
 - La destruction du composite entraîne la destruction de toutes ses parties (le composite est responsable du cycle de vie des parties)
 - même règle de traduction en Java que pour une association simple

Règle de traduction UML2Java pour une Relation de dépendance

- 1 dépendance UML entre packages UML
= directives d'import en java

