Institut Informatique
Claude Chappe

Master mention Informatique
Spécialité ISI

Génie Logiciel et modélisation
M1 / 177UD01

C5 – Langage de contraintes objet (*Object Constraint Language* : OCL)

Claudine Piau-Toffolon

Plan du cours

- Expression des contraintes en UML
  - Introduction
  - Ecriture des contraintes
  - Représentation des contraintes et contraintes prédéfinies
- OCL
  - Introduction
  - Illustration
- Typologie des contraintes OCL
  - Contexte
  - Invariants (*inv*)
  - Préconditions et postconditions (*pre, post*)
  - Résultat d'une méthode (*body*)
  - Définition d'attributs et de méthodes (*def* et *let …in*)
  - Initialisation (*init*) et évolution des attributs (*derive*)
- Types et opérations utilisables dans les expressions OCL

Expression de contraintes objet

- UML permet d'exprimer certaines formes de contraintes avec UML:
  - **Contraintes structurelles**
  - **Contraintes de type**
  - **Contraintes diverses**

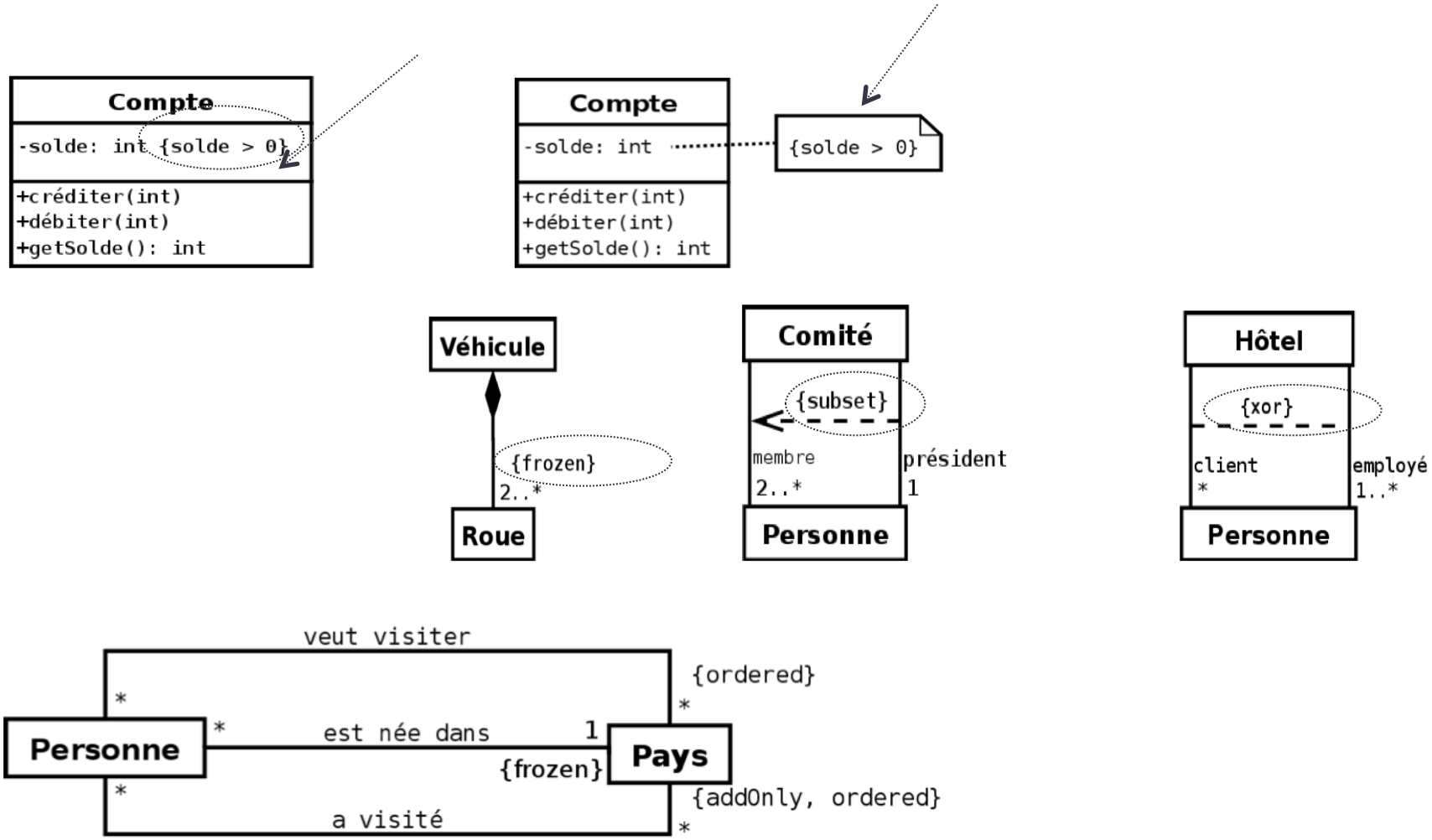
Une contrainte

- constitue une restriction sémantique exprimée sous forme d'instruction dans un langage textuel qui peut être naturel ou formel.
- peut être attachée à n'importe quel élément de modèle ou liste d'éléments de modèle.
- désigne une restriction qui doit être appliquée par une implémentation correcte du système.

- On représente une contrainte sous la forme d'une chaîne de texte placée entre accolades ({}).
- La chaîne constitue le corps écrit dans un langage de contrainte qui peut être :
  - naturel;
  - dédié, comme OCL;
  - ou encore directement issu d'un langage de programmation.

Représentation des contraintes

- UML pemet d'associer une contrainte à un, ou plusieurs, élément(s) de modèle de différentes façons



OCL - Introduction

- UML formalise l'expression des contraintes avec OCL
- OCL est un langage formel d'expression de contraintes bien adapté aux diagrammes d'UML, et en particulier au diagramme de classes
- OCL existe depuis la version 1.1 d'UML (contribution d'IBM)
- OCL fait partie intégrante de la norme UML depuis la version 1.3 d'UML
- Dans le cadre d'UML 2.0 les spécifications du langage OCL figurent dans un document indépendant de la norme UML
  - décrit en détail la syntaxe formelle et la façon d'utiliser ce langage.
- 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 :
  - des invariants sur les classes;
  - des préconditions et postconditions à l'exécution d'opérations;
  - des gardes sur des transitions de diagrammes d'états-transitions ou des messages de diagrammes d'interaction;
  - des ensemble d'objets destinataires pour un envoi de message;
  - des attributs dérivés, etc.

Exemple: Contexte application bancaire

Il faut gérér :

- Des clients et des banques

On aimerait intégrer les contraintes suivantes:

- Un compte doit avoir un solde toujours positif;
- Un client peut posséder plusieurs comptes;
- Une personne peut être cliente de plusieurs banques;
- Un client d'une banque possède au moins un compte dans cette banque;
- Un compte appartient forcément à un client;
- Une banque gère plusieurs comptes;
- Une banque possède plusieurs clients

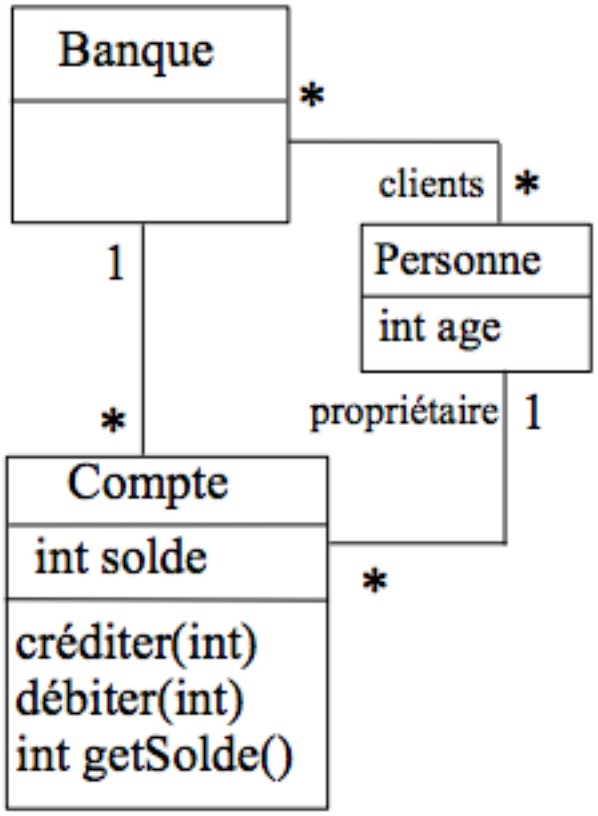


Diagramme d'instances-1/2

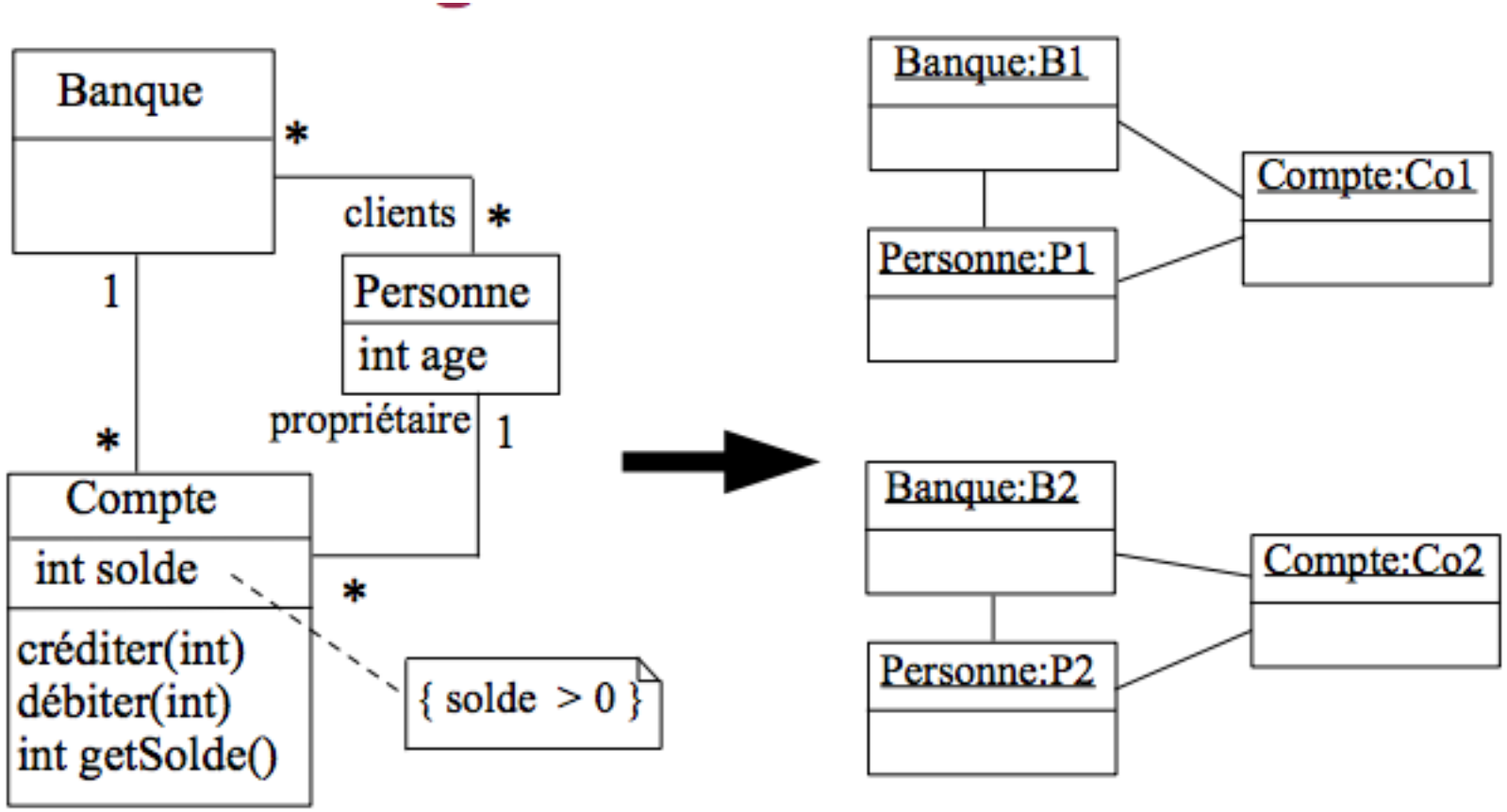
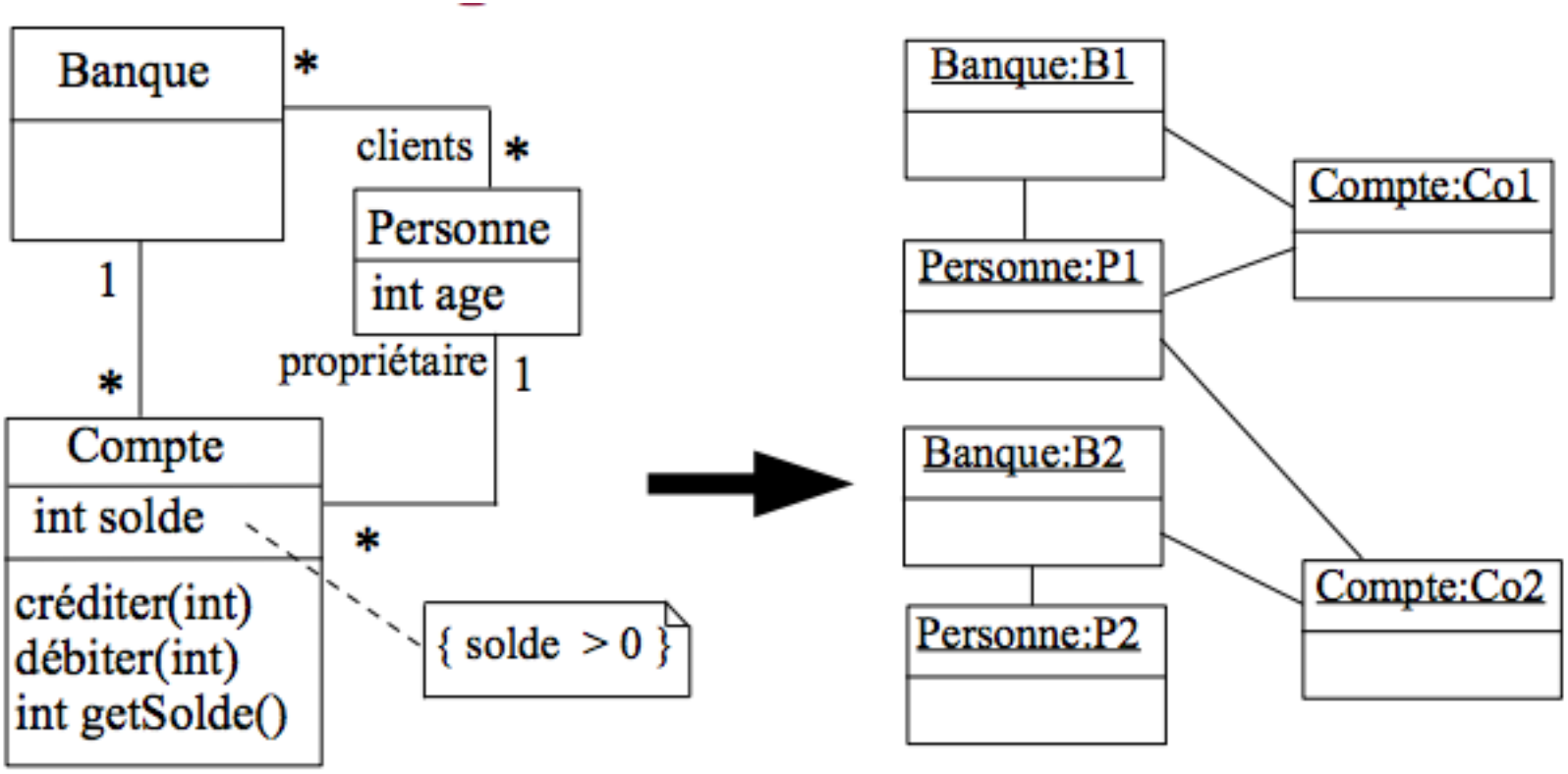
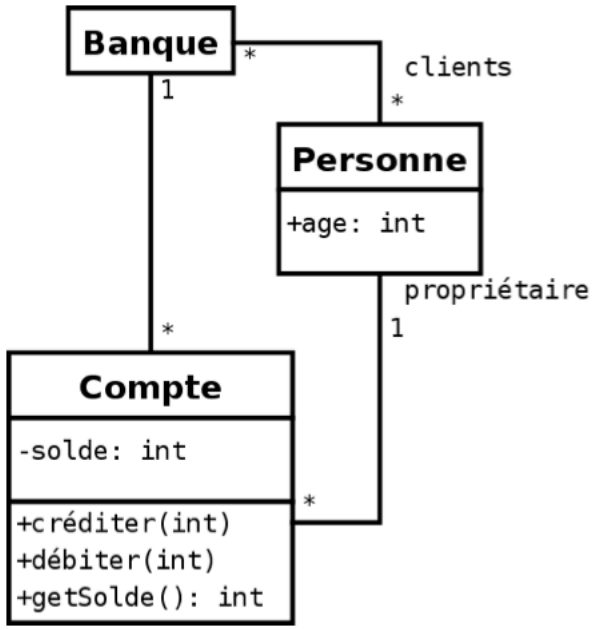


Diagramme d'instances 2/2



Exemple d'utilisation du langage de contrainte OCL

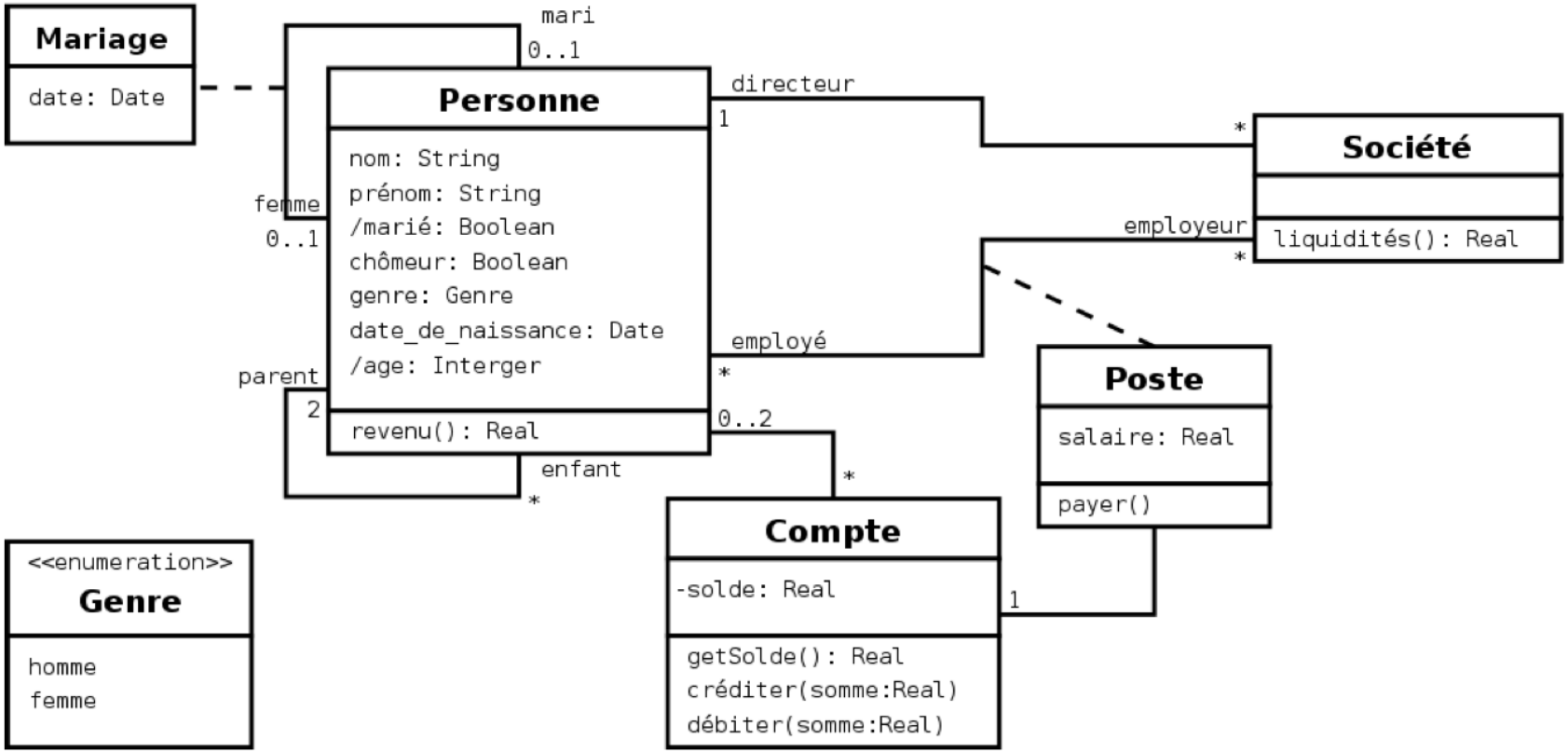


context Compte
inv : solde > 0

context Compte **: :** débiter(somme : int)
pre : somme > 0
post : solde = solde@pre - somme

context Compte
inv : banque.clients -> includes (propriétaire)

Exemple plus complet



Contexte

- Une contrainte est toujours associée à un élément de modèle.
- Cet élément constitue le contexte de la contrainte
- Syntaxe:

  `Context <élément>`

  `<élément>` peut être une classe, une opération, etc.

  Pour faire référence à un élément *op* (comme un opération) d'un classeur *C* (comme une classe), ou d'un paquetage, . . ., il faut utiliser les :: comme séparateur (comme *C*::*op*).

- Exemple

Le contexte est la classe *Compte* :

context Compte

Le contexte est l'opération *getSolde()* de la classe *Compte* :

context compte ::getSolde()

Invariants

- Un invariant exprime une contrainte prédicative sur un objet, ou un groupe d'objets, qui doit être respecté en permanence.
- Syntaxe

```
inv : <expression_logique>
```

`<expression_logique>` est une expression logique qui doit toujours être vraie

- Exemple

Le solde d'un compte doit toujours être positif.

context compte
inv : solde >0

Les femmes (au sens de l'association) des personnes doivent être des femmes (au sens du genre).

context Personne
inv : femme >forAll(genre=Genre:femme)