TD5 : OCL
UML

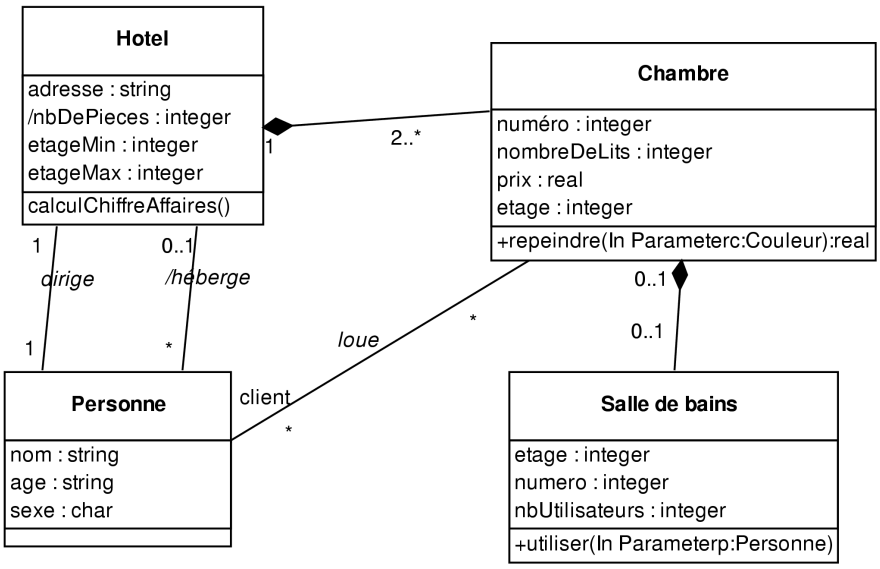
Pierre Gérard
pierre.gerard@univ-paris13.fr

DUT Informatique S2D
Université de Paris 13

1 Version

Le directeur d'une chaîne d'hôtels vous demande de concevoir une application de gestion de ses hôtels. Un hôtel est constitué d'un certain nombre de chambres. Un responsable de l'hôtel gère la location des chambres. Chaque chambre se loue à un prix donné. L'accès aux salles de bains est compris dans le prix de la location d'une chambre. certaines chambres comportent une salle de bains, mais pas toutes. Les hôtes de chambres sans salle de bain peuvent utiliser une salle de bains sur le palier. Ces dernières peuvent être utilisées par plusieurs hôtes. Les pièces de l'hôtel qui ne sont ni des chambres ni des salles de bain (hall d'accueil, cuisine...) ne font pas partie de l'étude (hors sujet). Des personnes peuvent louer une ou plusieurs chambres d'hôtel afin d'y résider. En d'autres termes : l'hôtel héberge un certain nombre de personnes, ses hôtes (il s'agit des personnes qui louent au moins une chambre de l'hôtel).

Le diagramme de classes suivant modélise ce problème :



Question : Donnez une formulation en langage naturel pour chacune des contraintes OCL suivantes :

```
context Chambre inv :
  self.etage <>13
context SalleDeBains inv :
  self.etage <>13
```

Un hôtel ne contient jamais d'étage numéro 13

```
context Chambre inv :
```

```
  client->size <= nombreDeLits or
    (client->size = nombreDeLits +1 and
      client->exists(p:Personne | p.age < 4))
```

Le nombre de personnes par chambre doit être inférieur ou égal au nombre de lits dans la chambre louée. Les enfants (accompagnés) de moins de 4 ans ne comptent pas dans cette règle de calcul (à hauteur d'un enfant de moins de 4 ans maximum par chambre)

```
context Hotel inv :
  self.chambre->forAll (c : Chambre |
    c.etage <= self.etageMax and c.etage >= self.etageMin)
```

L'étage de chaque chambre est compris entre le premier et le dernier étage de l'hôtel

```
context Hotel inv :
  Sequence{etageMin..etageMax}->forAll(i : Integer |
    if i<>13 then
      self.chambre->select(c : Chambre | c.etage = i)->notEmpty
    endif)
```

Chaque étage possède au moins une chambre (sauf le 13 qui n'existe pas, bien entendu...)

```
context Chambre::repeindre(c:Couleur)
  pre :  client->isEmpty
  post : prix = prix@pre * 1.1
```

On ne peut repeindre une chambre que si elle n'est pas louée. Une fois repeinte, une chambre coûte 10% de plus.

```
context SalleDeBains::utiliser(p:Personne)
  pre : if chambre->notEmpty then
          chambre.client->includes(p)
        else
          p.chambre.etage = self.etage
        endif
  post : nbUtilisateurs = nbUtilisateurs@pre + 1
```

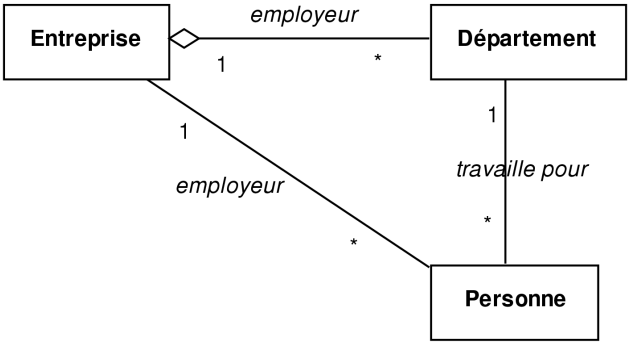
Une salle de bain privative ne peut être utilisée que par des personnes qui louent la chambre contenant la salle de bains et une salle de bains sur le palier ne peut être utilisée que par les clients qui logent sur le même palier

```
context Hotel::calculerLoyer() : integer
  post : result = self.chambre->select(client->notEmpty).prix->sum
```

Le loyer de l'hôtel est égal à la somme du prix de toutes les chambres louées

2 Thème

Considérez le diagramme de classes suivant :



Question : Donnez une expression OCL qui permette d'indiquer que la personne qui travaille dans le département est la même que celle qui est employée par l'entreprise.

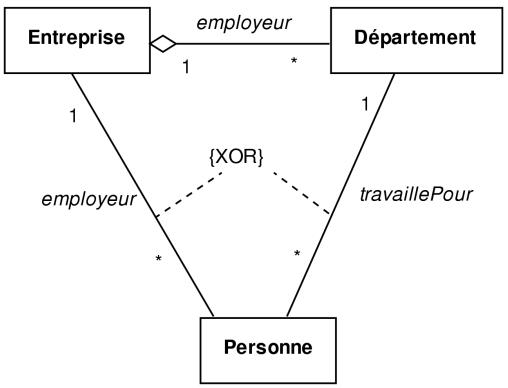
context Personne inv :
self.employeur = self.departement.employeur

Question : Donner une expression OCL qui permette d'indiquer qu'une personne travaillant pour une entreprise doit être âgée de 18 ans et plus. On suppose que la classe Personne a un attribut âge.

context Personne inv :
self.age > 18

Question : Modifier, graphiquement, le diagramme des classes précédent pour prendre en compte la contrainte suivante :

```
context personne inv :
  (self.departement -> isEmpty)
  xor
  (self.Entreprise -> isEmpty)
```



Question : Ajouter la contrainte indiquant que deux personnes ne doivent pas avoir le même nom.

context Personne inv :
Personne.allinstances -> forAll(p1,p2 | p1<>p2 implies p1.nom <> p2.nom)

Question : Aucune personne n'est âgée de plus de 130 ans. Le jour d'anniversaire de la personne, son âge est augmenté de 1 an. Ajouter l'expression OCL permettant de représenter cette précondition et cette post-condition associées à l'opération anniversaire de Personne.

context Personne : :anniversaire()
pre : age >= 1 and age <130
post : age = age@pre +1

Question : Les personnes qui travaillent dans l'entreprise sont âgées de 18 à 65 ans. Donner l'expression OCL correspondante.

context Entreprise inv :
self.employeur.forAll(Personne p | p.age >=18 and p.age <=65)