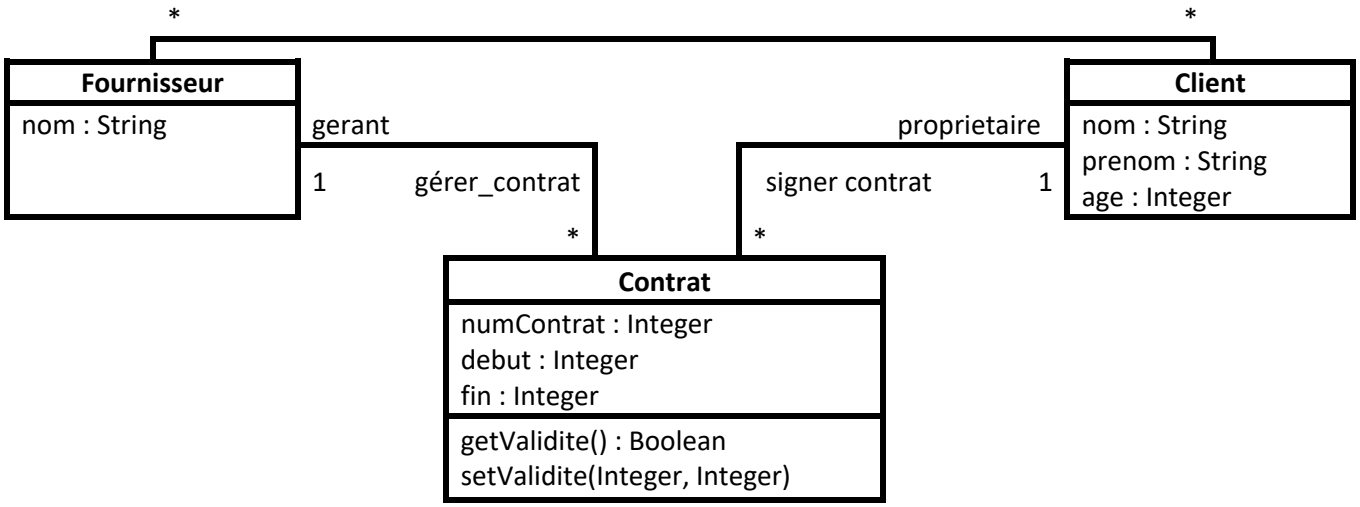
FICHE DE TD N° 2 – OCL : SOLUTION

Exercice 1 :

1. Soit le diagramme de classes suivant :



En partant du ***context*** *Client*, exprimer les contraintes suivantes on OCL :

2. La date de début de contrat doit être inférieur à sa fin pour que la méthode *getValidite* puisse être exécutée :

```
context Client
inv: signer_contrat. debut  < contrat. fin  implies  signer_contrat. getValidate()
```

3. Un contrat doit toujours avoir un *numContrat* positif.

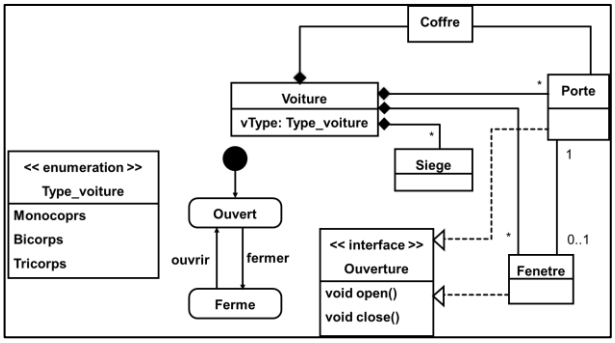
```
context Client
inv: signer_contrat. numContrat >=0
```

4. Un fournisseur ne peut gérer un contrat d'un client s'il n'est pas en association avec ce client.

```
context Client
inv: contrat.gérant->includeAll (fournisseur)
inv: fournisseur= contrat. fournisseur -- Alternative
```

Exercice 2 :

Soit le modèle suivant :



Exprimer les contraintes suivantes en OCL :

1. Un coffre contient soit une porte sans fenêtre, soit rien :

```
context Coffre
inv: porte.size () <=1 and  Fenêtre.size() =0
```

2. Une voiture à 5 portes doit avoir 5 sièges et 4 fenêtres :

```
context Voiture
porte.size() =5 implies siege.size() =5  and  fenêtre.size()
```

3. Une voiture monocorps doit avoir :
   a. Ou bien : 3 portes et 4 sièges
   b. Ou bien : 5 portes et 5 sièges

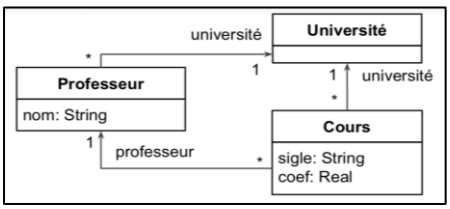
```
context Voiture
inv: Voiture->select(type= monocorps)->forAll(
        (Porte.size() =3 and Siege.size() =4) Or
        ( Porte.size() =3 and Siege.size() =5)
        )
```

4. Une voiture tricorps doit avoir :
   a. Ou bien : 2 portes et 4 sièges.
   b. Ou bien : 4 portes et 5 sièges.

```
context Voiture
inv: Voiture->select(type= tricorps)->forAll(
        (Porte.size() =2 and Siege.size() =4) Or
        ( Porte.size() =4 and Siege.size() =5)
        )
```

Exercice 3 :

Soit le diagramme de classe suivant. Exprimer en langage naturel les contraintes OCL suivantes :



1. **context** Professeur inv: nom <> null
   -- Le nom d'un prof ne doit pas être null
2. **context** Cours
   a. **inv:** sigle->size() = 3
   -- Le sigle d'un cours doit être composé de trois lettres
   b. **inv:** coef >= 0.0 and coef <= 100.0
   -- Le coefficient d'un cours doit être entre zero et 100
   c. **inv:** universite = professeur.universite
   -- Un prof doit travailler dans l'université qui l'embauche
   d. **inv:** coef=0 implies professeur.isEmpty()
   -- Un module à coefficient zero ne doit pas être affecté à un prof/ ou bien Un prof n'enseigne pas un module qui n'a pas de coefficient
3. **context** Universite
   a. **inv:** let nbrProf = professeur->size() in
      cours->size() <= nbrProf * 2 and
      cours->size() >= nbrProf
   -- Le nombre de cours doit être deux fois inférieur au nombre de professeur , et supérieur au nombre de professeur