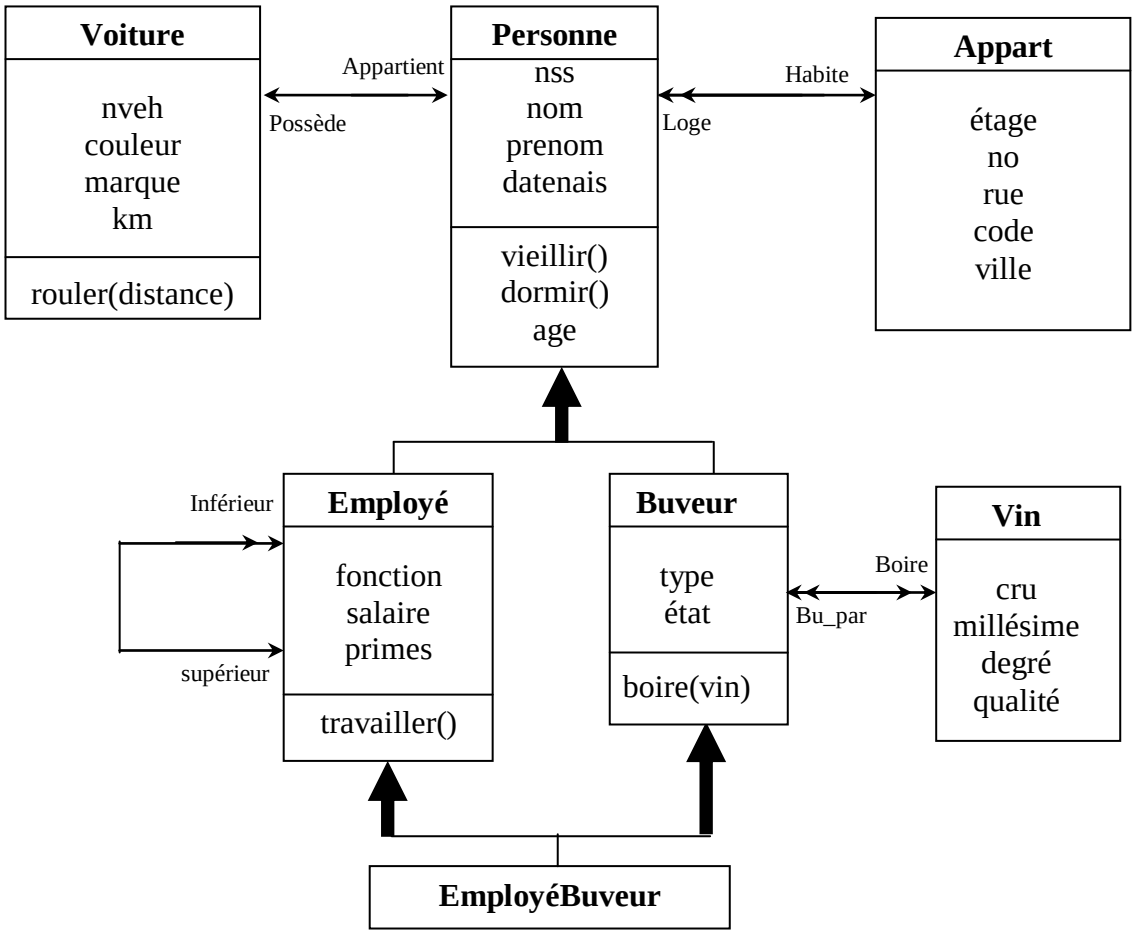
Matière : Bases de Données Avancées — Niveau : **3ème Année Ingénieur (GLSI)**

Enseignants : **R. ZAAFRANI – R. KAABI** — Octobre 2014

TD n°3 : Base de données Orientée Objets

On considère le schéma d'une base de données objet avec la notation graphique des schémas en ODL.



La base décrit simplement des situations du monde réel : des personnes possèdent des voitures, et habitent des appartements. Parmi elles, certaines sont employées, d'autres sont buveurs. Parmi les employés il y a des buveurs. Les buveurs boivent des vins.

1) Définir un Schéma **ODL** possible pour la base de données. L'attribut **fonction** d'employé est un attribué énuméré pouvant avoir comme valeurs : ingénieur, secrétaire, analyste ou programmeur. L'attribut **primes** d'employé est un attribute multivalué. L'attribut **type** de buveur peut prendre l'une de ces valeurs : petit, moyen ou gros et l'attribut **état** de buveur peut prendre l'une des deux valeurs : normal, ivre. Les attributs de la classe appart sont encapsulés en un seul attribut structuré : **adresse**.
2) Définir les requêtes **OQL** suivantes :
   a. Nous attribuons le nom persistant MAVOITURE à l'objet désignant ma voiture. Donnez sa couleur ?
   b. Donnez le nom de son propriétaire ?
   c. Retrouvez les noms et les prénoms des grands buveurs ?
   d. Retrouvez les noms et les prénoms des employés grands buveurs ?
   e. Retrouvez les noms et les prénoms des buveurs qui ont bu du Volnay?
   f. Retrouvez les doublets (nom, ville) pour chaque gros buveur?
   g. Donner la liste des identifiants d'objets des voitures de la marque « Renault » sous la forme d'un tableau?
   h. Retrouvez le nom, la ville d'habitation et l'âge des employés dont le salaire est supérieur à 10000 et l'âge inférieur à 30 ans.
   i. Donnez le nom et la liste des inférieurs de chaque employé qui sont mieux payés que lui ?
   j. Donnez pour chaque ville le salaire moyen des employés s'il est supérieur à 5000 ?

Correction du TD n°3

1) Schéma ODL possible pour la base de données

Une décision à prendre lors de la définition est de choisir entre interfaces et classes. Certaines classes peuvent avoir des extensions. Nous avons choisi d'implémenter les extensions de Voiture, Vin, Appartement, Buveur et Employé et Personne.

```
class Voiture (extent voitures key nveh) {
  attribute string nveh ;
  attribute string couleur;
  attribute string marque;
  attribute short km;
  relationship Personne Appartient inverse Personne :: Possède;
  short rouler(in short distance);
} ;

class Personne (extent personnes key nss) {
  attribute string nss ;
  attribute string nom;
  attribute string prenom;
  attribute date datenaissance;
  relationship Appart habite inverse Appart :: loge;
  relationship Voiture Possède inverse Voiture :: Appartient;
  short vieillir();
  void dormir();
  short age();
} ;

class Employé extends Personne (extent employés key nss) {
  attribute enum fonct{ingénieur, secrétaire, analyste, programmeur} fonction ;
  attribute float salaire;
  attribute list<float> primes; //attribut multivalué
  relationship Employé inférieur inverse :: supérieur;
  relationship Set<Employé> supérieur inverse :: inférieur;
  void travailler();
} ;

class Buveur : Personne (extent buveurs key nss) {
  attribute typebuv{petit, moyen, gros} type ;
  attribute etabuv{normal, ivre} etat ;
  relationship list<Vin> boire inverse vin :: bu_par;
  void boire(in Vin v);
} ;

class Appart (extent Apparts) {
  attribute struct adresse(short etage, unsigned short numero, string rue, unsigned short code, string ville) ;
  relationship Set<personne> loge inverse Personne::habite;} ;

class Vin (extent Vins) {
  attribute string cru;
  attribute string millesime;
  attribute string degre;
```

```
attribute       string qualite;
relationship    list<Buveur> bu_par    inverse    Vin ::boire;} ;
```

L'héritage multiple n'étant pas possible au niveau des classes, les employés buveurs sont des employés avec les propriétés de buveurs répétées.

```
class EmployéBuveur extends Employé    (extent employés_buveurs) {
  attribute       typebuv{petit, moyen, gros} type ;
  attribute       etabuv{normal, ivre} etat ;
  relationship    list<Vin>    boire  inverse    vin ::bu_par;
  void            boire(in Vin v);} ;
```

2) **OQL** :

a. Nous attribuons le nom persistant MAVOITURE à l'objet désignant ma voiture. Donnez sa couleur ?

```
(Q1):    mavoiture.couleur;
```

b. Donnez le nom de son propriétaire ?

```
(Q2):    mavoiture.appartient.nom;
```

c. Retrouvez les noms et les prénoms des grands buveurs ?

```
(Q3):    SELECT    b.nom, b.prénom
         FROM      b in buveurs
         WHERE     b.type = 'gros';
```

d. Retrouvez les noms et les prénoms des employés grands buveurs ?

```
(Q4):    SELECT    eb.nom, eb.prénom
         FROM      eb in employés_buveurs
         WHERE     eb.type = 'gros';
```

e. Retrouvez les noms et les prénoms des buveurs qui ont bu du Volnay?

```
(Q5):    SELECT    b.nom, b.prénom
         FROM      b in buveurs, v in b.boire
         WHERE     v.cru= 'Volnay';
```

Les collections dépendantes se parcourent par des variables imbriquées derrière le FROM.

f. Retrouvez les doublets (nom, ville) pour chaque gros buveur ?

```
(Q6):    SELECT    STRUCT (nom : b.nom, ville:b.habite.adresse.ville)
         FROM      b in buveurs
         WHERE     b.type = 'gros';
```

g. Donner la liste des identifiants d'objets des voitures de la marque « Renault » ?

```
(Q7):    ARRAY( SELECT V
                FROM    v in voiture
                WHERE   v.marque = 'renault')
```

h. Retrouvez le nom, la ville d'habitation et l'âge des employés dont le salaire est supérieur à 10000 et l'âge inférieur à 30 ans ?

```
(Q8):    SELECT DISTINCT e.nom, e.habite.adresse.ville, e.age()
         FROM      e in employés
         WHERE     e.salaire > 10000 AND e.age()<30;
```

i. Donnez le nom et la liste des inférieurs de chaque employé qui sont mieux payés que lui ?

```
(Q9):    SELECT DISTINCT (nom:e.nom, inf_mieux_payés :
         LIST( SELECT i
               FROM     i in e.inférieur
               WHERE    i.salaire > e.salaire))
         FROM e in employés;
```

j. Donnez pour chaque ville le salaire moyen des employés s'il est supérieur à 5000 ?

```
(Q10):  SELECT STRUCT(ville:e.habite.adresse.ville, moy:AVG(salaire))
         FROM      e in employés
         GROUP BY  e.habite.adresse.ville
         HAVING AVG (SELECT e.salaire FROM e in PARTITION)> 5000;
```