INFO-H-200 : Programmation orientée objet
TP 4 - UML

Professeur : Hugues Bersini
http://cs.ulb.ac.be/public/teaching/infoh200

Année académique 2014-2015

Exercice 4.1 : Diagramme de classes

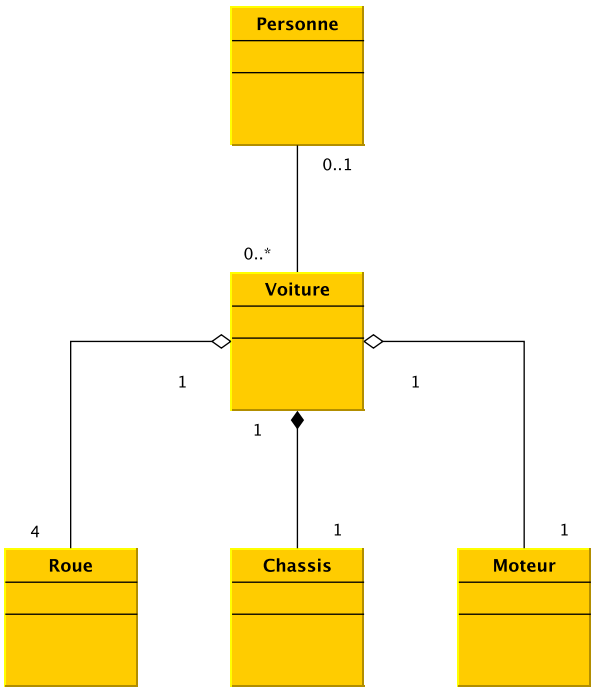
Dessiner pour les différentes situations suivantes le diagramme de classes simplifié correspondant :

1. Un pays possède une capitale.
2. Une personne dîne avec une fourchette.
3. Un fichier contient des enregistrements
4. Des personnes utilisent un langage pour un projet
5. Une équipe est composée de plusieurs personnes
6. Une route connecte deux villes.
7. Un dessin est soit du texte, soit une forme géométrique, soit un groupe de dessins.
8. Un ordinateur est composé d'un ou plusieurs moniteurs, d'un boîtier, d'une souris optionnelle et d'un clavier. Un boîtier possède un châssis métallique, une carte mère, plusieurs barrettes de mémoire (RAM, ROM et cache), un ventilateur optionnel, des supports de stockage (disquette, disque-dur, CD-ROM, DVD-ROM...), et des cartes périphériques (son, réseau, graphique...). Un ordinateur possède toujours au moins un lecteur de disquette ou un disque-dur.
9. L'Université comporte de nombreux inscrits dans ses registres ; du personnel administratifs et techniques, mais aussi des enseignants, des étudiants et des chercheurs (qui sont tous des personnes). Certains étudiants peuvent être des chercheurs (les doctorants) ou des enseignants (les assistants enseignants). Certaines personnes (étudiants ou non) peuvent être à la fois chercheurs et enseignants.

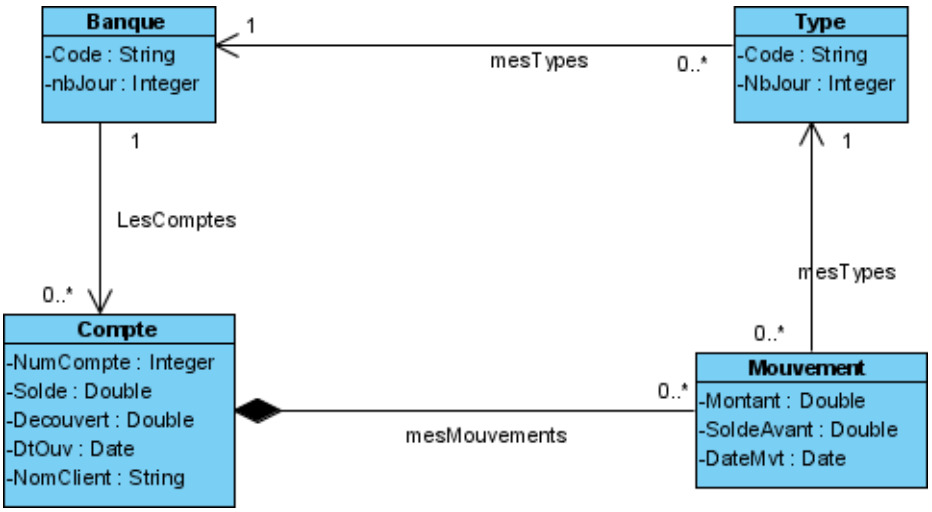
Exercice 4.2 : Interprétation des diagrammes de classes

Expliquer et interpréter le contenu des diagrammes de classes partiels suivants :

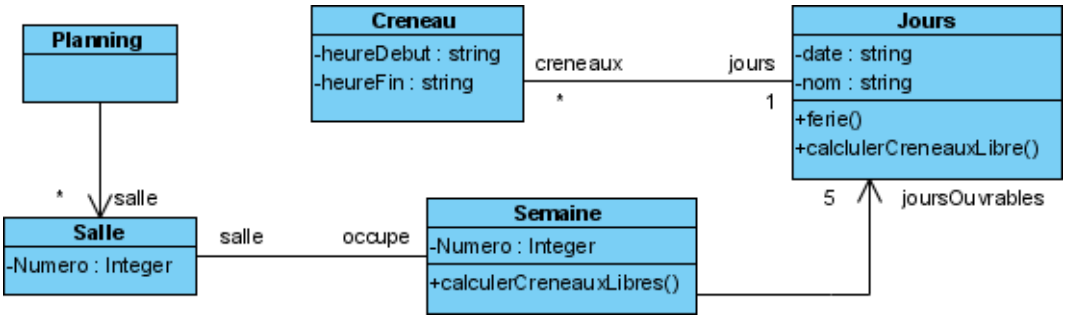
1.


Personne
0..1
0..*
Voiture
1
1
1
4
1
1
Roue
Chassis
Moteur

2.

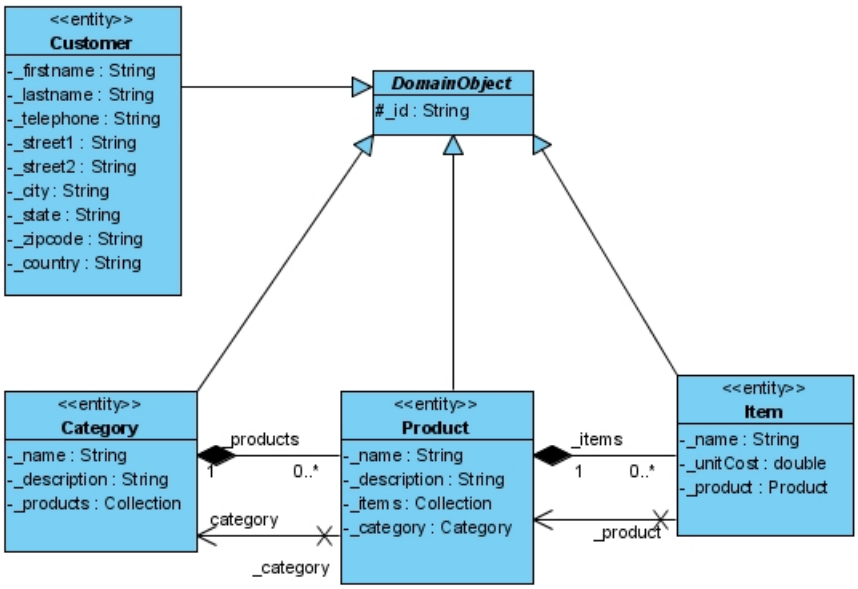

Banque
-Code : String
-nbJour : Integer
1
mesTypes
0..*
Type
-Code : String
-NbJour : Integer
1
LesComptes
0..*
1
mesTypes
0..*
Compte
-NumCompte : Integer
-Solde : Double
-Decouvert : Double
-DtOuv : Date
-NomClient : String
0..*
mesMouvements
Mouvement
-Montant : Double
-SoldeAvant : Double
-DateMvt : Date

3.


Planning
Creneau
-heureDebut : string
-heureFin : string
creneaux
*
jours
1
Jours
-date : string
-nom : string
+ferie()
+calclulerCreneauxLibre()
*
salle
5
joursOuvrables
Salle
-Numero : Integer
salle
occupe
Semaine
-Numero : Integer
+calculerCreneauxLibres()

Exercice 4.3 : Implémentation de diagrammes de classes

Ecrivez le squelette de code Java qui pourrait être automatique généré à partir de ces diagrammes de classes UML suivant. Collection fait référence à une ArrayList.



Exercice 4.4 : Diagramme de séquence

Un restaurant décide de se munir d'un système informatique pour pouvoir gérer les commandes et faire un suivi de la clientèle. Lorsqu'un client passe commande, il fourni un identifiant (ex : son numéro de téléphone), ainsi que sa commande composée d'une série de plats et de leurs quantités respectives. En réponse à sa commande, le client va recevoir le prix de sa commande. S'il effectue le paiement demandé, le restaurant va lancer les préparations pour ensuite les livrer au client.

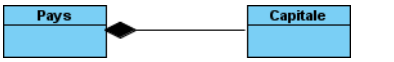
Pour suivre la clientèle, le restaurant sauvegarde dans son système séparément chaque commande pour ses différents clients. Il peut ainsi retrouver toutes les informations de ces derniers à partir de son identifiant et y ajouter toutes les informations relatives à sa nouvelle commande. Une commande se composera à son tour d'une liste de pizzas, identifiées à partir de leur nom, et de leur nombre.

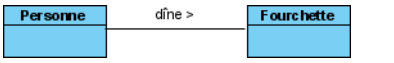
Dessiner un diagramme de séquence répondant au problème, en supposant les interactions d'un acteur vis-à-vis d'objets de type `Restaurant`, `Client`, `Commande` et `Pizza`.

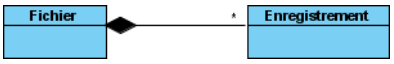
INFO-H-200 - Programmation orientée objet
TP 4

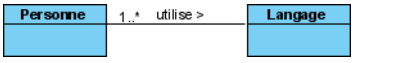
Corrections

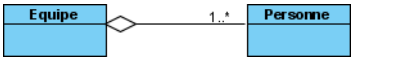
Exercice 4.1

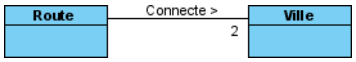
1.


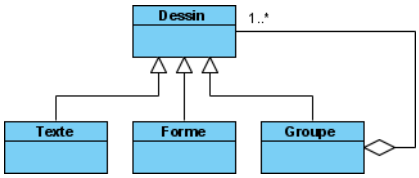
2.


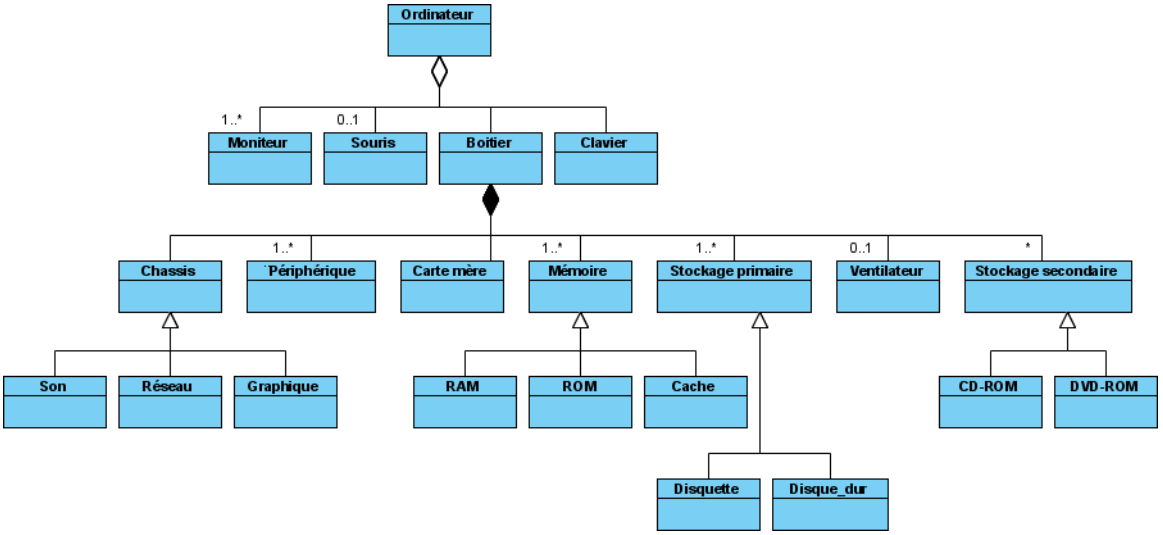
3.


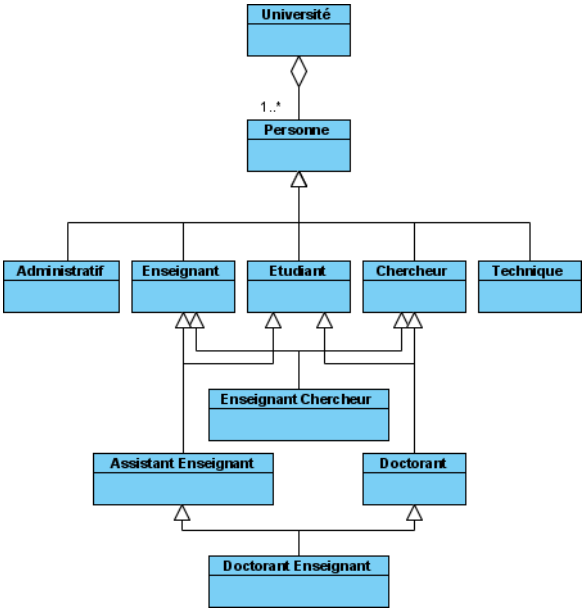
4.


5.


6.


7.


8.


9.


Exercice 4.2

1. L'ensemble du diagramme décrit un voiture. Une voiture appartient : soit à aucun propriétaire, soit à une seule personne. Par contre, une même personne peut avoir de 0 jusqu'à une infinité de voitures. Le châssis est un élément indissociable d'une voiture, d'où la composition (agrégation forte). Par contre, le moteur et les roues peuvent être utilisés dans d'autres voitures. Une voiture a également obligatoirement quatre roues. Les cardinalité sont assimilables '1' sont assimilables à '1..1', de plus, les indiquer n'est pas obligatoire puisqu'elle sont la cardinalité par défaut (voir slides). Donc, un moteur ne peut servir que pour une seule voiture à la fois, et une voiture ne peut avoir qu'un seul moteur.
2. Un mouvement bancaire est une opération concernant un compte. Un mouvement possède une date de réalisation, un montant (en crédit ou en débit) et est d'un certain type. Il est dépendant d'un compte (si on supprime un compte, tous les mouvements disparaissent également) ; à l'inverse, les comptes sont indépendants d'une banque. Il existe plusieurs types de mouvements bancaires, mais un mouvement est d'un seul type ! Par contre, la banque en accepte de différentes sortes également.
3. La classe `Créneau` contient les informations d'un créneau horaire déterminé par une heure de début et une heure de fin. Un créneau est associé à un jour. Il permet d'identifier un créneau d'occupation d'une salle un jour donné.
   La classe `Jour` contient les informations sur l'ensemble des créneaux occupés ce jour. Un jour est déterminé par une date et un nom (jour de la semaine). Il est possible de savoir si un jour correspond à un jour férié et quels sont ses créneaux libres. Cette dernière opération est possible grâce à l'association navigable entre les classes Jours et Creneau, qui permet d'avoir la liste des créneaux occupés pour un jour donné. Cette classe permet donc d'identifier les créneaux d'occupation d'une salle pour un jour donné.
   La classe `Semaine` contient les informations sur les créneaux composant l'ensemble des jours de la semaine. Une semaine est représentée par son numéro. L'association navigable de la classe Semaine vers la classe Jour permet d'avoir accès aux informations sur les créneaux occupés de l'ensemble des jours de la semaine. Cette association est nécessaire pour la réalisation de l'opération calculerCreneauxLibres(), qui permet d'identifier les créneaux libres pour une semaine donnée. De plus, nous avons l'information suivante : une Semaine est composée de cinq jours ouvrables exactement. Cette classe permet donc d'identifier les créneaux d'occupation d'une salle pour une semaine donnée.
   La classe `Salle` contient les informations sur l'occupation d'une salle. Une salle est représentée par son numéro. L'association navigable entre la classe Salle et la classe Semaine permet de récupérer l'ensemble des informations sur l'occupation de la salle pour un nombre quelconque de semaines. Une semaine est associée à une seule salle. Une même semaine (en terme de numéro) existera en autant d'exemplaires que de salles occupées cette même semaine. Chacune de ces semaines représente l'occupation d'une salle en particulier. Cette classe permet donc d'identifier les créneaux d'occupation d'une salle pour un ensemble de semaines.
   La classe `Planning` contient les informations sur l'occupation d'un ensemble de salles (éventuellement vides). Elle ne contient aucune propriété. Grâce à une association vers la classe Salle, il est possible d'accéder à l'ensemble des salles et de récupérer les informations sur leur occupation.

Exercice 4.3

Fichier Customer.java

```
public class Customer extends DomainObject {
    private String _firstname;
    private String _lastname;
    private String _telephone;
    private String _street1;
    private String _street2;
    private String _city;
    private String _state;
    private String _zipcode;
    private String _country;

    public Customer() {
    }
}
```

Fichier Category.java

```
public class Category extends DomainObject {
    private String _name;
    private String _description;
```

```
    private ArrayList _products;

    public Category() {
    }
}
```

Fichier Product.java

```
public class Product extends DomainObject {
    private String _name;
    private String _description;
    private ArrayList _items;
    private Category _category;

    public Product() {
    }
}
```

Fichier Item.java

```
public class Item extends DomainObject{
    private String _name;
    private double _unitCost;
    private Product _product;

    public Item() {
    }
}
```

Fichier DomainObject.java

```
public class DomainObject {
    protected String _id;

    public DomainObject() {
    }
}
```

Exercice 4.4