Centre Universitaire de Mila
Institut des sciences et de la technologie
Informatique- Semestre 5

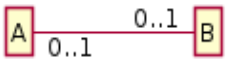
Génie logiciel
Série de TP N°03

TP N°03 : Concepts UML vers Déclaration JAVA (Suite)

I. Associations entre classe

III.1. Association bidirectionnelle

III.1.1. Association bidirectionnelle 1 vers 1




```
public class A {
    private B b; (getter et setter publics)

}

public class B {
    private A a; (getter et setter publics)

}
```

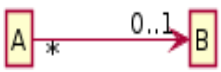
III.1.2. Association bidirectionnelle 1 vers *


```
public class A {
    private List <B> b; (getter, adder et remover publics)
}

public class B {
    private A a; (getter et setter publics)
}
```

III.2. Association unidirectionnelle

III.2.1. Association unidirectionnelle * vers 1




```
class A {
    private B b;
    (getter et setter publics)
}
```

III.2.2. Association unidirectionnelle * vers *


```
class A {
    private List<B> b;
    (getter, adder et remover publics)
}
```

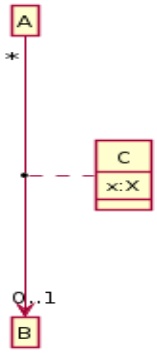
Ou avec tableaux :

```
class A {
    private B[] b;
    (getter, adder et remover publics)
}
```

- NB1: Les agrégations s'implémentent comme les associations.
- NB2 : Les compositions peut s'implémenter comme une association unidirectionnelle.

III.3. Classe association

III.3.1. Classe association unidirectionnelle * vers 1

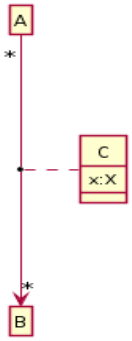



```
class A {

    private B b;
    private C c;
    (getB, setB et éventuellement getC)
}

class C {
    private X x;
    (getter et setter publics)
}
```

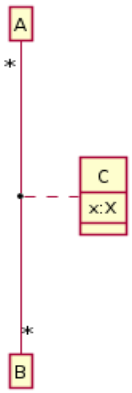
III.3.2. Classe association unidirectionnelle * vers *



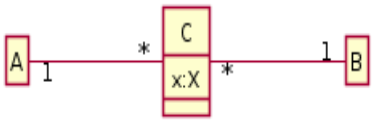

```
class A {
    private List<C> c;
    (getter et setter publics)
}


class C {
    private B b;
    private X x;
    (getters et setters publics)
}
```

III.3.3. Classe association bidirectionnelle * vers *



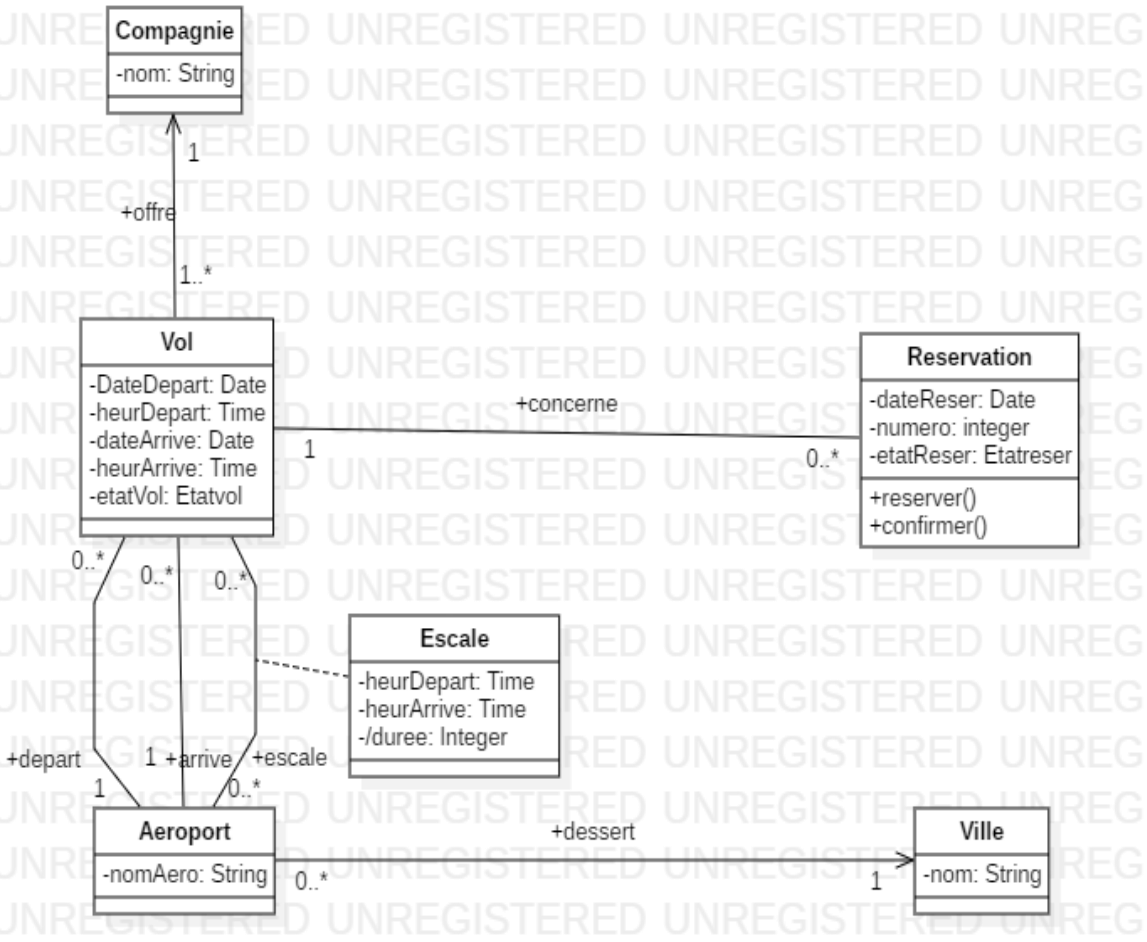
En pratique, on se ramène à un cas comme :



II. Exercice1 : Diagramme de classe UML vers code Java

Une compagnie aérienne offre plusieurs vols. Le vol part d'un aéroport ville vers un autre aéroport d'une ville et peut faire escale dans un ou plusieurs aéroports pour une certaine durée. Plusieurs réservations (confirmées ou pas) peuvent se faire sur un vol. Le diagramme de classes ci-dessous montre les relations entre les classes.

- Codez en java ces classes et testez en une classe principale des exemples d'instances.



III. Exercice2 : Code Java vers diagramme de classe

```
public interface Dessinable {
public void dessiner ( );
public void effacer ( );
}
abstract public class Figure implements Dessinable {
protected String couleur;
protected String getCouleur ( ) { return couleur; }
protected void setCouleur ( String c ) { couleur = c; }
}
public class Point {
private float x; private float y;
public float getX ( ) { return x; }
public float getY ( ) { return y; }
public void Point ( float x, float y) { ... }
}
```

```
public class Cercle extends Figure {
private float rayon;
private Point centre;
public Cercle ( Point centre, float rayon) { ... }
public void dessiner ( ) { ... }
public void effacer ( ) { ... }
}
public class Rectangle extends Figure {
protected Point sommets[] = new Point[2];
public Rectangle ( Point p1, Point p2) { ... }
public void dessiner ( ) { ... }
public void effacer ( ) { ... }}
public class Losange extends Figure {
protected Point sommets[] = new Point[2];
public Losange ( Point p1, Point p2) { ... }
public void dessiner ( ) { ... }
public void effacer ( ) { ... }
}
```

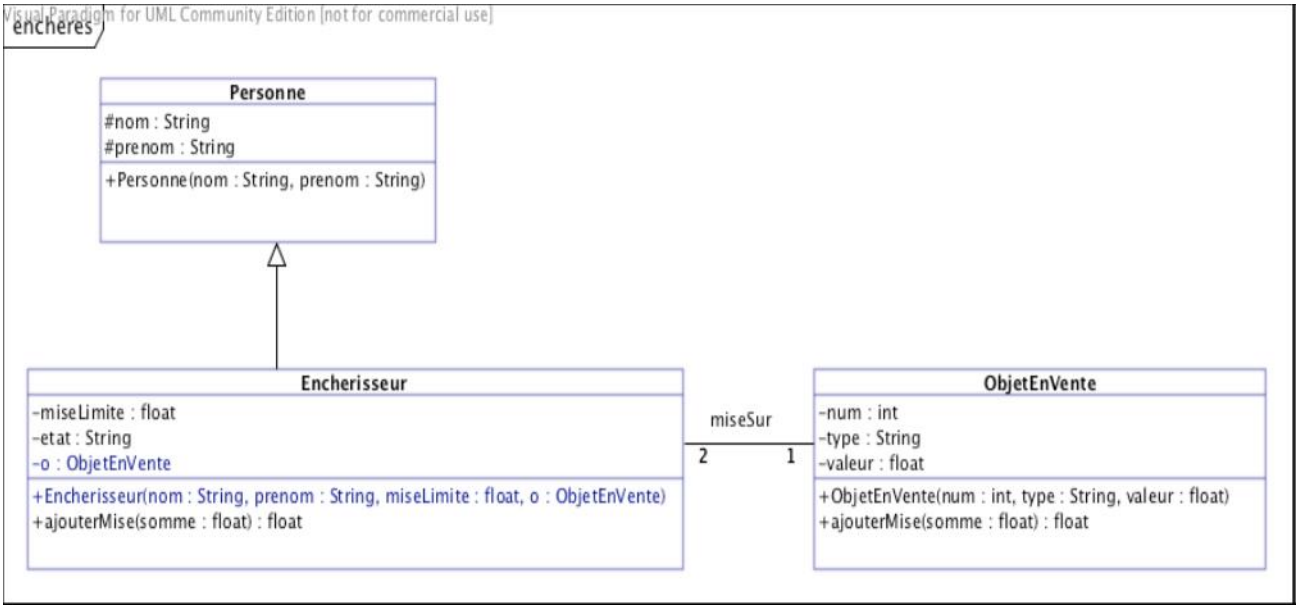
1. Donnez un diagramme de classes correspondant au code source suivant
2. Modéliser le diagramme de classe avec « Visuel Paradigm Entreprise » ou « StarUML » et produire un diagramme de classe au format png.
3. Générer le code java correspondant.

IV. Exercice2 : Vente aux enchères

Nous allons dans cette partie vous initier aux concepts de base en Java en réalisant une petite application de vente aux enchères, qui permet à deux personnes (appelés enchérisseurs) de miser sur un objet en vente. Cet objet est défini par un numéro (par exemple 1273), un type (par exemple tableau) et une valeur (en dinars) représentant son prix à l'instant présent (par exemple 500). L'objet en vente est initialisé à la valeur que précise le responsable de la vente.

Les deux enchérisseurs ont chacun une limite de prix à ne pas dépasser : si le prix de l'objet dépasse cette limite, ils ne pourront pas acheter l'objet. Ils misent une somme à tour de rôle. Le premier dont la limite est atteinte quitte l'enchère, et c'est son adversaire qui acquiert l'objet.

1- Implémenter le modèle dans la figure ci-dessous



2- Implémenter toutes les méthodes dans les classe Encherisseur et ObjetEnVente de manière à ce que : Quand un enchérisseur veut ajouter une somme à sa mise, il teste d'abord si la valeur de l'objet en vente a atteint sa mise limite. Si c'est le cas, il abandonne la partie en affichant : « l'enchérisseur nom_encherisseur a abandonné la partie » et met son état à « out ». Si ce n'est pas le cas, il appelle la méthode ajouterMise de l'objet en vente, et met la somme qu'il propose comme paramètre. Quand la méthode ajouterMise de la classe ObjetEnVente est appelée, la somme passée en paramètre sera ajoutée à l'attribut valeur. Vous pouvez ajouter d'autres méthodes si besoin est.

3- Implémenter une classe Main et :

• Créer un nouvel objet obj de type ObjetAVendre, qui porte le numéro 123, de type « tableau » et de valeur initiale 1000.

• Créer deux enchérisseurs : le premier s'appelle « Ali Sadouki » et le second « Samir Hadji». Ali a mis comme mise limite 2000da et Samir 2500da. Tous les deux veulent acheter l'objet obj.

• Tant que les deux enchérisseurs sont dans la course (in), Ali va ajouter la somme de 100 da à la mise, et Samir ajoute 200 da.

• Le programme doit afficher à la fin le nom de l'enchérisseur qui a remporté l'objet, ainsi que la somme finale de l'objet.