Bases de la POO / Java

La relation de composition

Construction par assemblage (1)

➔ Nature des attributs d'une classe

- Attributs de types primitifs
- Scalaires ou vectoriels

Extension à des attributs de type OBJET

Construction par assemblage (2)

➔ Exemple

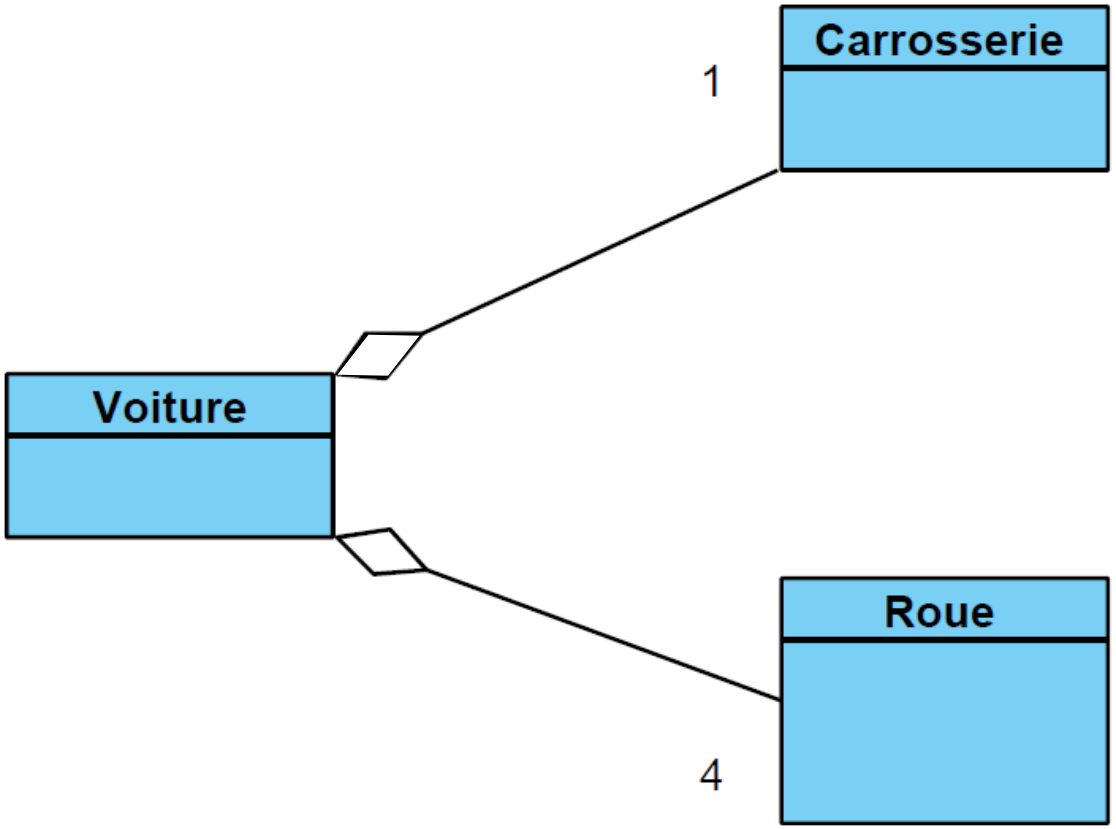
- Description d'une voiture :
  - Une carrosserie — **Classe Carrosserie**
  - 4 roues — **Classe Roue**
  - **...**
- Description d'une roue :
  - diamètre
  - largeur
  - ...

Construction par assemblage (3)

➔ Objectifs de la relation "is_part_of"

- Favoriser la réutilisation des codes sources
- Réutilisation des données
- Respecter les règles d'encapsulation
- Favoriser la conception par composition formelle

Diagramme de classes UML



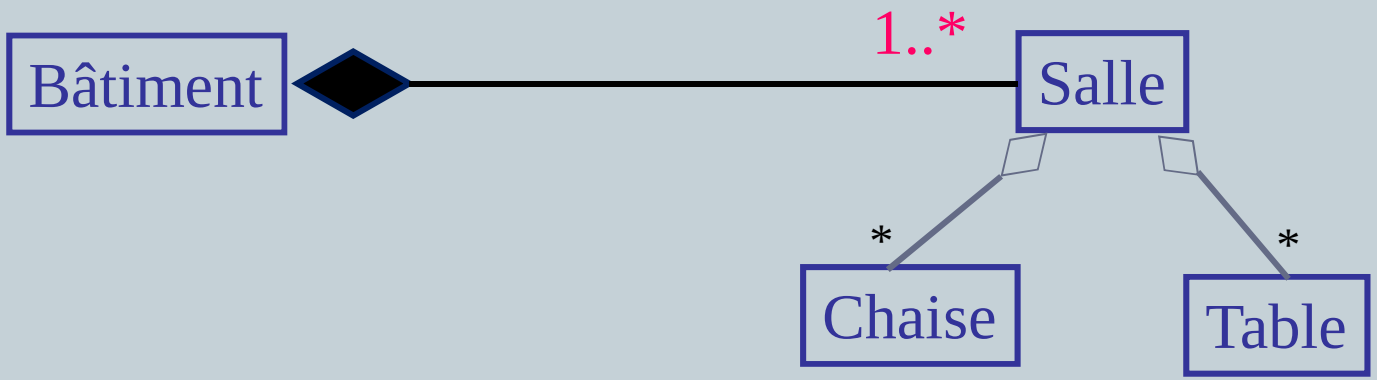
Construction par assemblage (4)

➔ Portée de la relation

- Relation exclusivement entre classes
- Classe composée
- Composants objets
- Structure composée éventuellement de plusieurs niveaux

Exemple

- Un bâtiment de différents étages comporte des salles, qui contiennent des chaises et des tables



Définition d'une classe composée


```
public class Voiture  {

private Carrosserie m_carrosserie;
private Roue m_RAvG, m_RAvD,
m_RArG, m_RArD;

--- Autres attributs éventuels
String m_immat;
}
```

Constructeur d'instances


```
public Voiture ( Carrosserie obj1, Roue obj2,
Roue obj3, Roue obj4, Roue obj5, String im)
{
    m_carrosserie= obj1;
    m_RAvG=obj2;
    m_RAvD=obj3;
    m_RArG=obj4
    m_RArD=obj5;
    // Affectation des autres attributs éventuels
    m_immat = im;
}
```

Création des instances


```
public static void main (String[] args)
{
// création des composants
Carrosserie laCarrosserie= new Carrosserie (list params);
Roue laRoue1= new Roue (listparams);
Roue laRoue2= new Roue (listparams);
Roue laRoue3= new Roue (listparams);
Roue laRoue4= new Roue (listparams);
// création du composé
Voiture  maVoiture = new Voiture (laCarrosserie, laRoue1, laRoue2, laRoue3, laRoue4, "11AB06");


}
```

Classe composée Segment

un segment est composé de 2 points distincts

```
public class Segment
{
    private Point origine;
    private Point extremite;

}
```

Constructeur par défaut


```
public Segment()   {
        origine= new Point();
        extremite= new Point(1., 0.);
}
```

Le choix du segment [(0, 0), (1,0)] est arbitraire.

Constructeur normal


```
public Segment (Point a, Point b) throws Throwable
{
        if (a.confondus(b)) throw new Throwable("-1");
        origine= a;
        extremite= b;
}
```

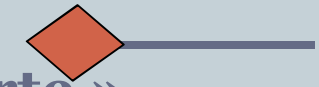
Méthode toString


```
public String toString ( )
{
        return "[" +
                origine.toString() +
                "," "  +
                extremite.toString() + "]";
}
```

Le choix de l'ordre des deux points est arbitraire.

UML

- **Agrégation**
  
  - Le composant existe en dehors du composé
  - Exemple : les 2 points du segment

- **Composition**
  
  - **Agrégation « forte »**
  - Tous les composants sont détruits quand on détruit le composé
  - Exemple : la tête d'un humain

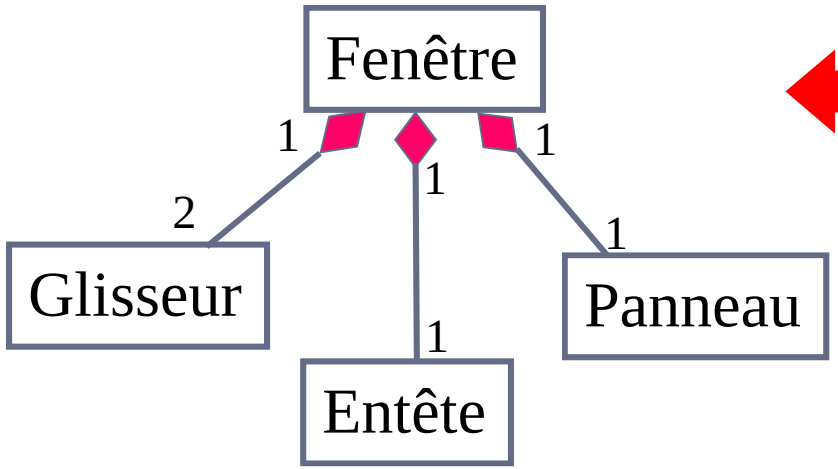
- **Cardinalité**
  - **Nombre d'objets composants dans le composé**

Exercices

- Un segment est constitué de 2 points distincts
- Une fenêtre graphique comporte 2 ascenseurs, 1 barre de titre, 1 zone client
- Un match de double au tennis : 2 joueurs, 1 date, 1 score
- Un bâtiment de différents étages comporte des salles, qui contiennent des chaises et des tables
- Une boîte peut contenir un seul objet et d'autres boîtes

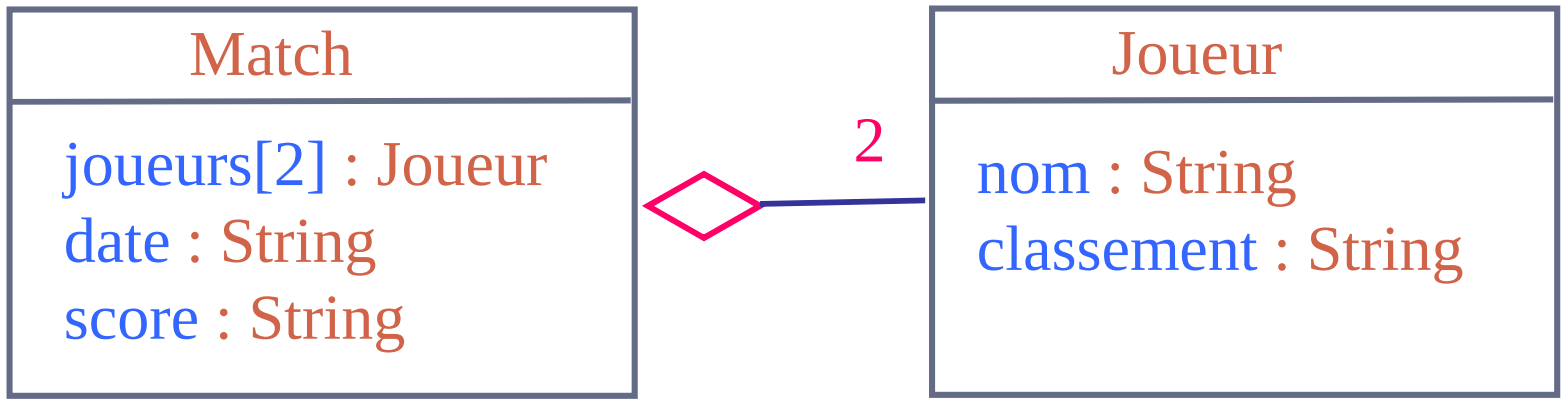
Un segment est constitué de 2 points distincts

Une fenêtre graphique comporte 2 ascenseurs, 1 barre de titre, 1 zone client



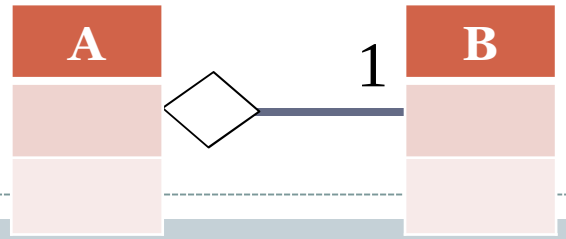
Fenêtre
ascenseur[2] : Glisseur titre : Entête corps : Panneau

Un match de double au tennis : 2 joueurs, 1 date, 1 score



Bâtiment

Agrégation




```
class A
{
   private B b;

   public A(B x)  { b=x; }

   public void montrer()
   { System.out.print("mon agrégé:");
     b.montrer();
   }

}
```

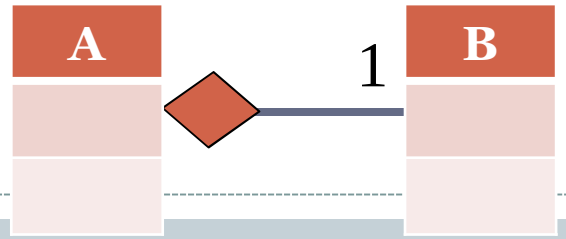
```
class B
{
   private String nom;

   public B(String x){nom=x;}

   public void montrer()
      { System.out.println(nom); }
}
```

```
public static void main(String[] args)
{
        B b = new B(“unB");
        b.montrer();
        A a = new A(b);
        a.montrer();
}
```

Composition




```
class A
{
    private B b;

    public A() { b=new B("mon B"); }
    public A(String nomB) { b=new B(nomB); }

    public void montrer()
     { System.out.print("mon composé:");
       b.montrer();
     }
}
```

```
class B
{
   private String nom;

   public B(String x) {nom=x;}

   public void montrer()
     { System.out.println(nom); }
}
```

```
public static void main(String[] args)
   {
            A a = new A();
            a.montrer();
            A a2 = new A("toto");
            a2.montrer();
   }
```