

# Modélisation UML

Identifiant relatif

Choix d'une méthode

Erreurs communes à éviter

Dérivation vers programme

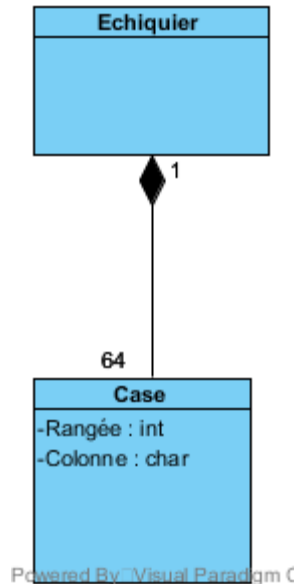
# Plan du cours

- Identifiants relatifs
- Choix entre alternatives de modélisation
- Erreurs communes à éviter
- Dérivation vers programme Java

# Association qualifiée

- Dans certains cas, l'association entre 2 entités n'est pas assez précise
- Certains attributs de la classe cible permettent de *qualifier* l'association
- Cela permet de restreindre une multiplicité de  
\* côté cible vers 1

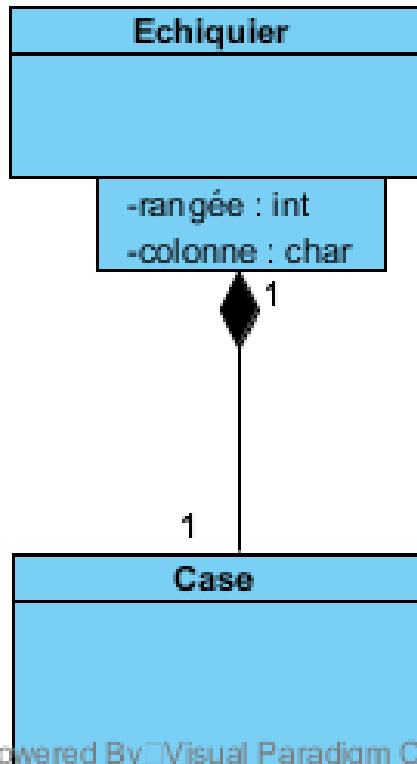
# Exemple - échiquier



On parle également *d'identifiant relatif*

Modèle d'origine: la classe Echiquier est composée de 64 Cases

# Modélisation Echiquier avec association qualifiée



Echiquier: classe qualifiée

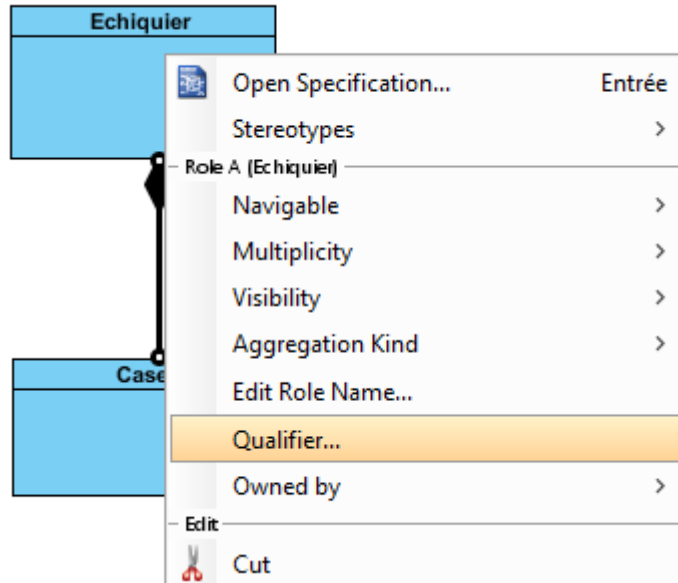
Qualificatifs

1: multiplicité après qualification

Case: classe cible

Une instance du **triplet** {Echiquier, rangée, colonne} est en association avec une instance *unique* de la classe Case

# Utilisation dans Visual Paradigm

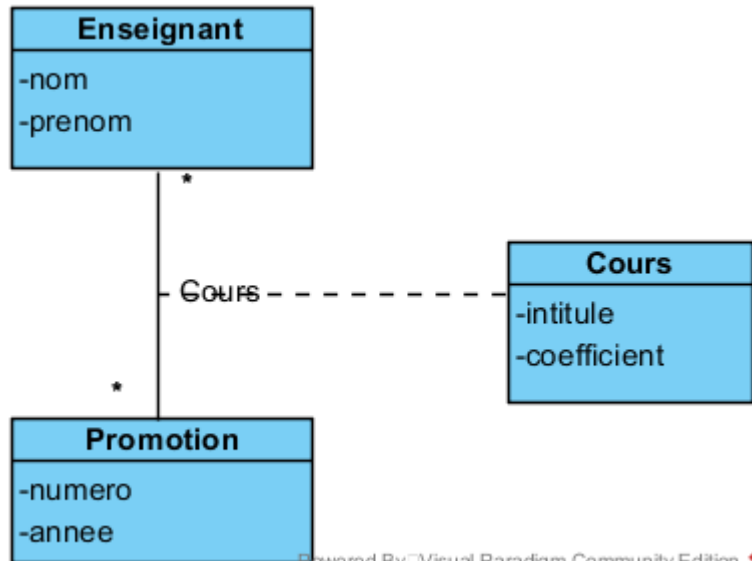


Clic droit sur l'association (Composition) côté Echiquier puis →  
Qualifier  
Ensuite Entrez les Attributs qui qualifient la composition

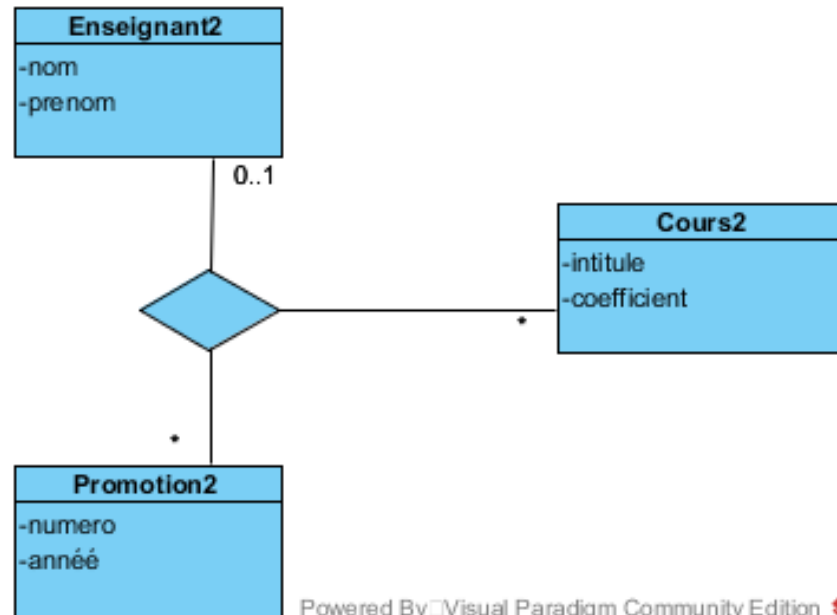
# Plan du cours

- Identifiants relatifs
- Choix entre alternatives de modélisation
- Erreurs communes à éviter
- Dérivation vers programme Java

# Choix entre alternatives de modélisation



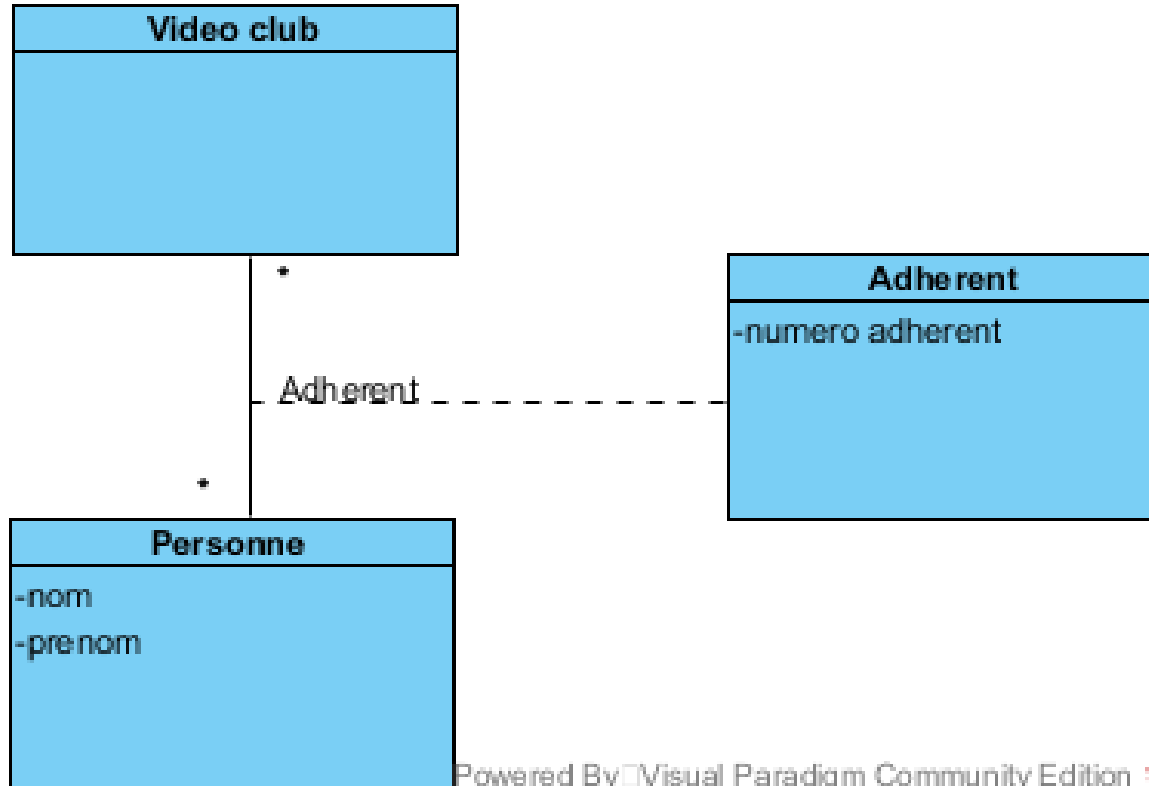
Powered By Visual Paradigm Community Edition



Powered By Visual Paradigm Community Edition



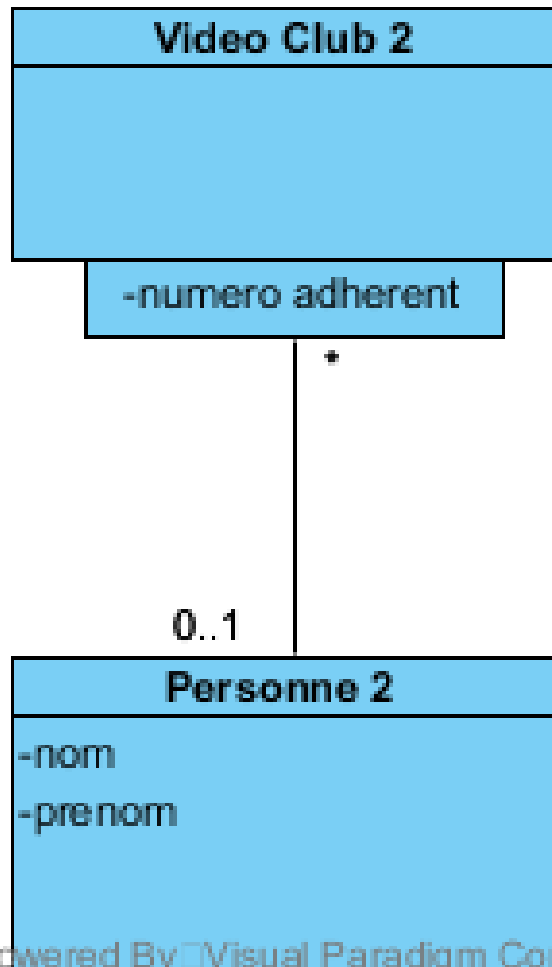
# Choix entre alternatives de modélisation



Powered By Visual Paradigm Community Edition

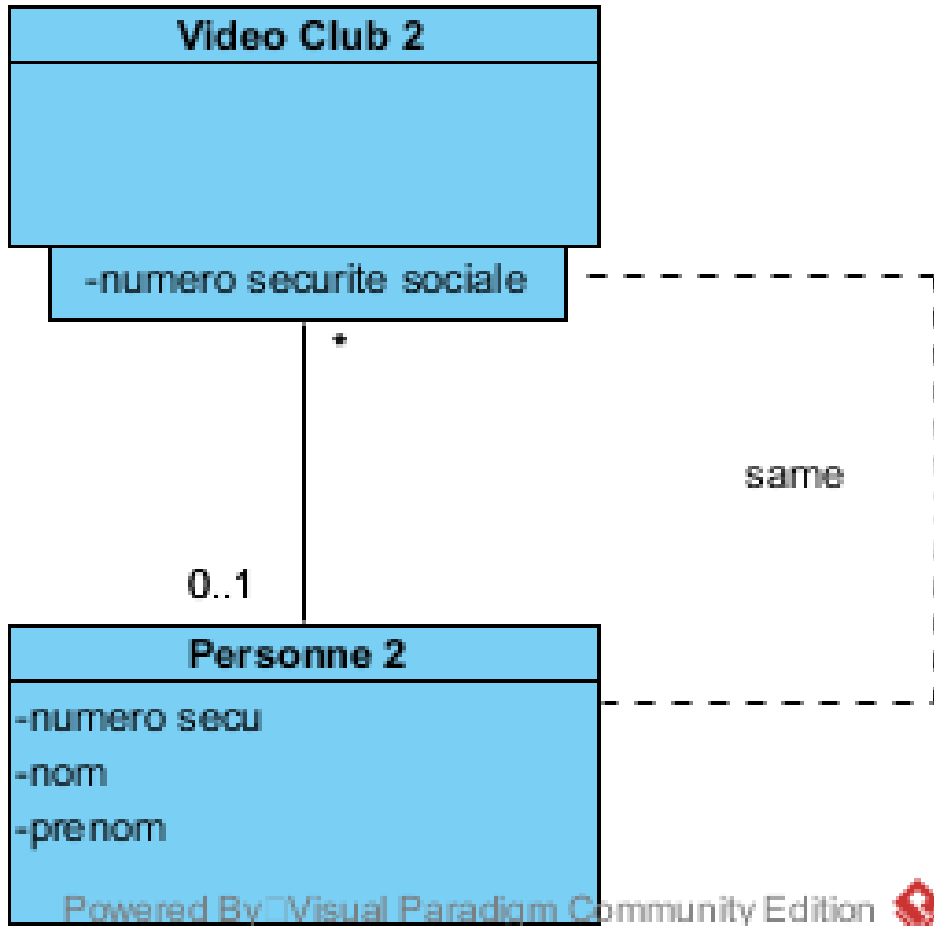
Ici on place une classe association pour indiquer que l'association entre une personne et un club vidéo se fait au moyen d'un numéro d'adhérent

# Choix entre alternatives de modélisation



On utilise cette fois ci une association qualifiée: les adhérents sont indexés via le numéro d'adhérent

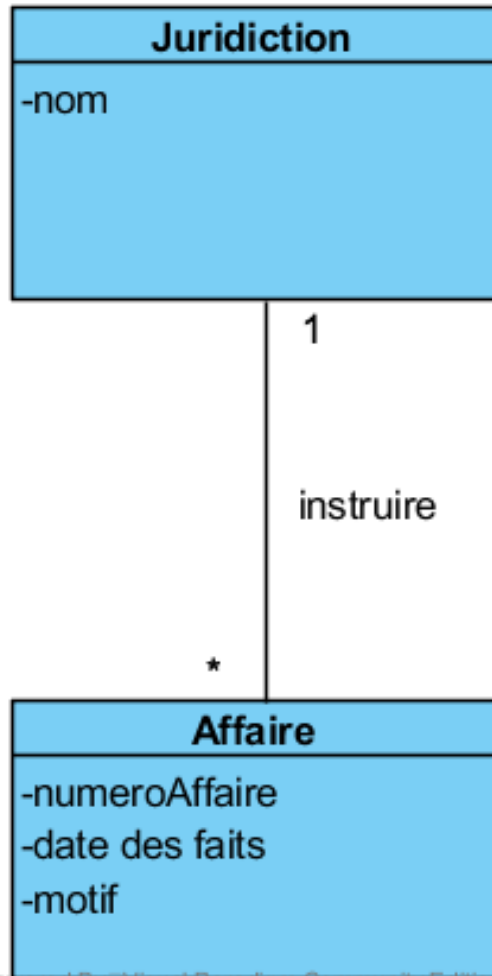
# Choix entre alternatives de modélisation



Pour cette association qualifiée, on utilise un identifiant déjà présent pour chaque personne, en précisant une contrainte de type « same »

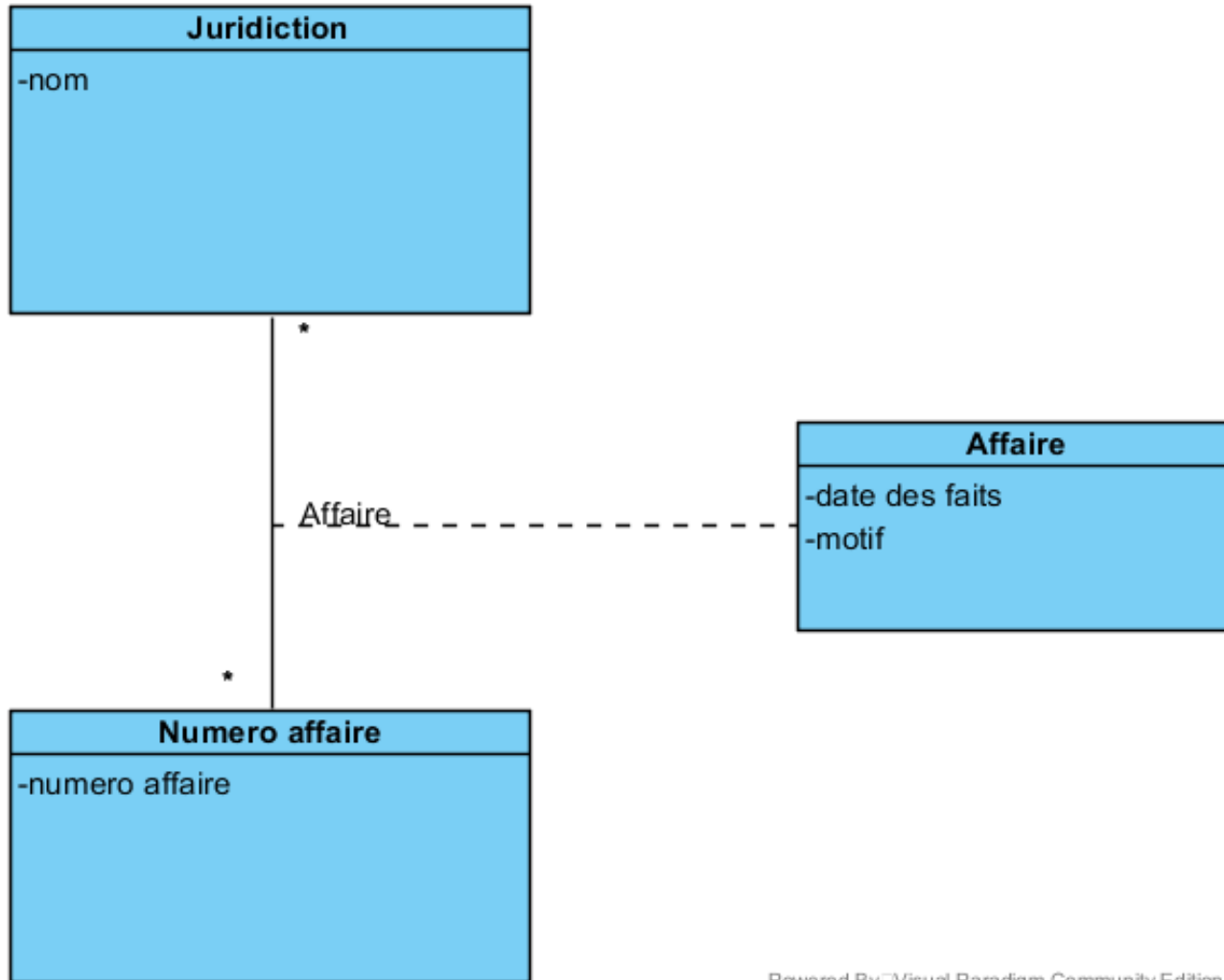


# Alternatives - prison Nantes Barbier (UML2)

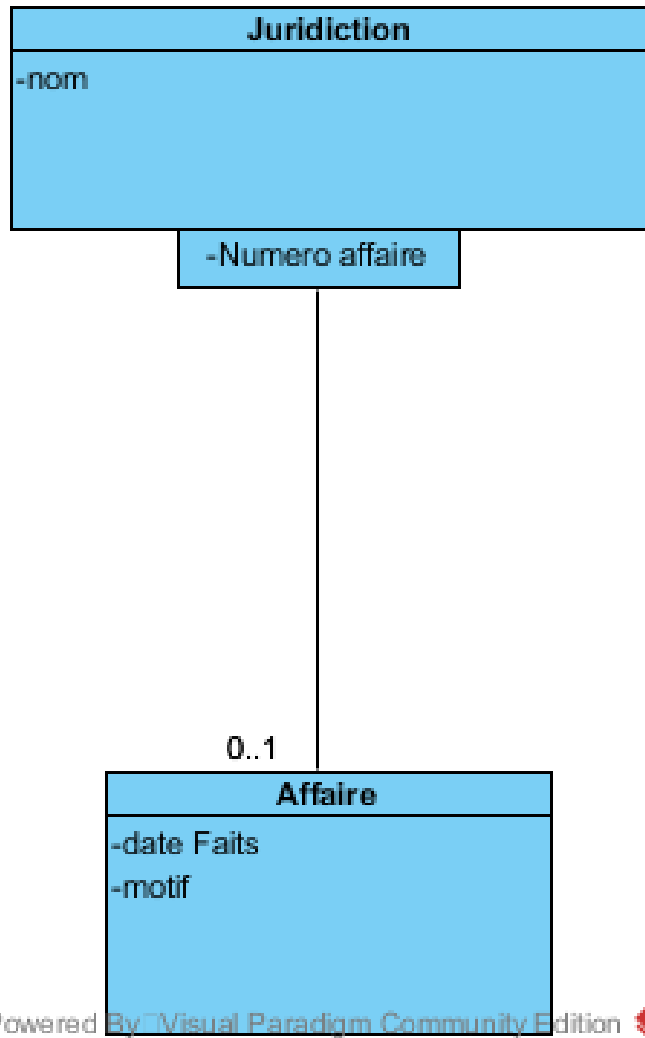


Une juridiction  
instruit plusieurs  
affaires

# Alternatives - prison Nantes Barbier (UML2)

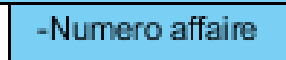
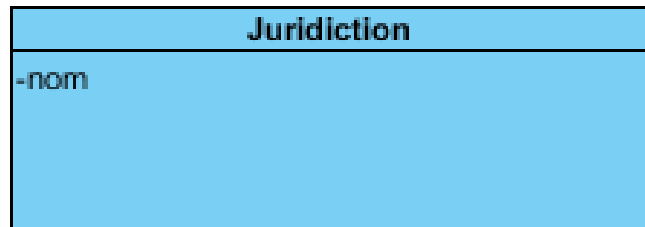


# Alternatives - prison Nantes Barbier (UML2)

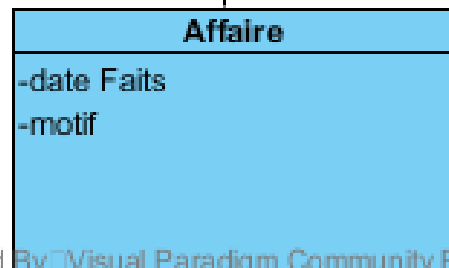


Association qualifiée  
où une affaire est  
accessible via le  
*numéro d'affaire* pour  
une juridiction  
donnée

# Alternatives - prison Nantes Barbier (UML2)



0..1



## Question de validation:

1. Le numéro d'affaire est il unique ?
2. Pour une juridiction , le numéro d'affaire est il unique ?

# Plan du cours

- Identifiants relatifs
- Choix entre alternatives de modélisation
- Erreurs communes à éviter
- Dérivation vers programme Java



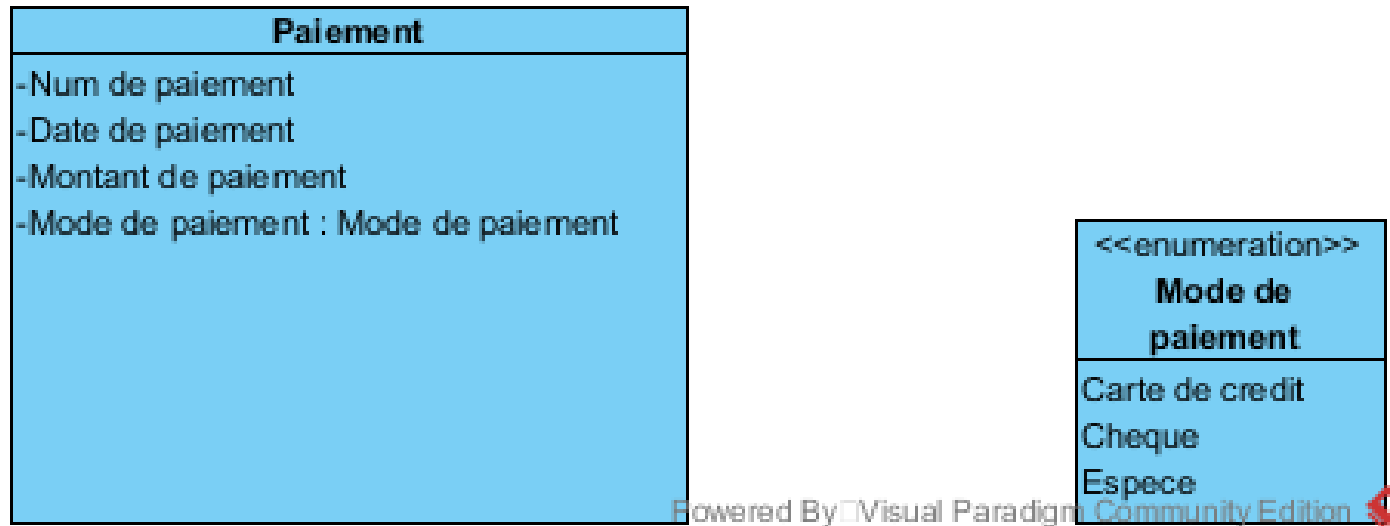
# Confusion entre attribut et valeur d'attribut

Paiement
-Num paiement
-Date de paiement
-Montant de paiement
-Carte de crédit : boolean
-Espèces : boolean
-Cheque : boolean

Powered By Visual Paradigm Community Edition

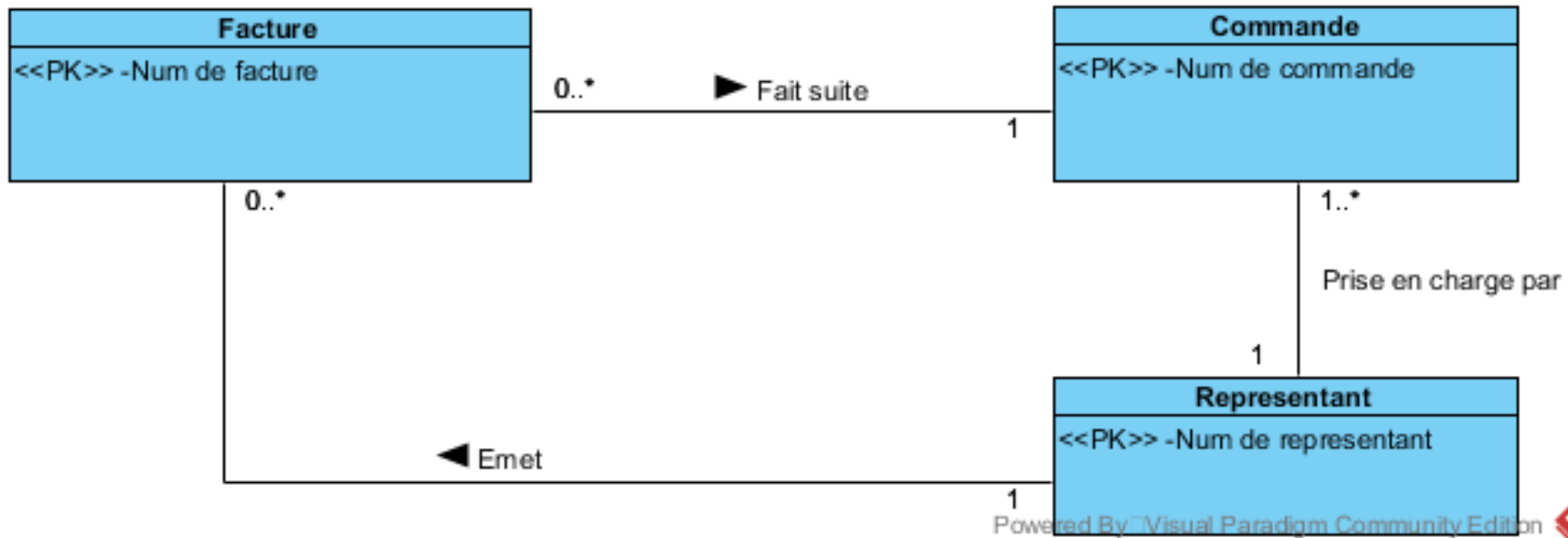
Ici Date et montant de paiement sont OK  
mais Carte de crédit, Espèces et Cheque  
ne sont pas de vrais attributs

# Confusion entre attribut et valeur d'attribut

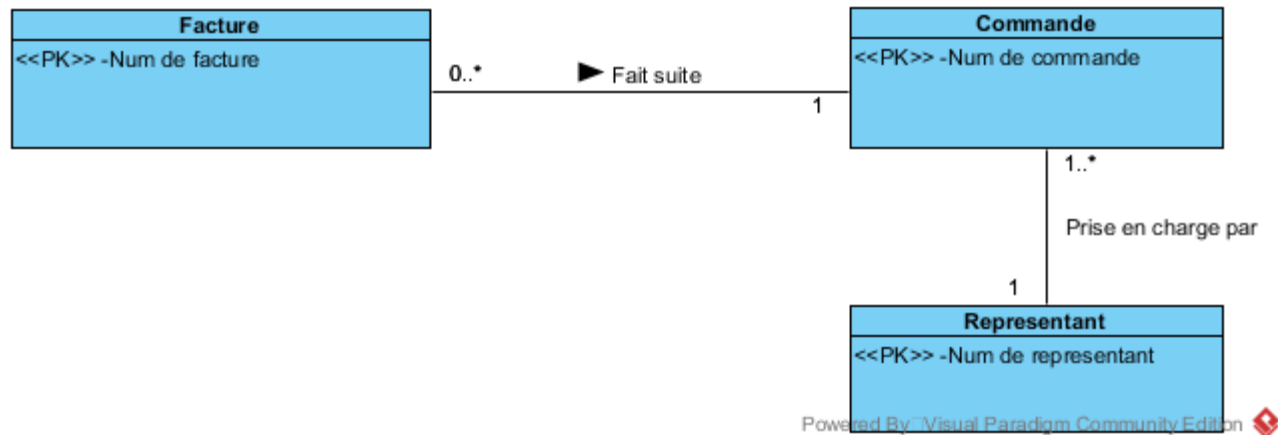


Un seul attribut est nécessaire: Mode de paiement, qui peut prendre la valeur Carte de Crédit, Cheque ou Espece

# Eviter les associations redondantes



# Eviter les associations redondantes



L 'association directe entre Facture et Représentant était redondante.

A partir du moment où la multiplicité est 1..1 entre Facture et Commande puis entre Commande et Representant, le lien est redondant

# Plan du cours


- Identifiants relatifs
- Choix entre alternatives de modélisation
- Erreurs communes à éviter
- Dérivation vers programme Java

# Implémentation en Java

- UML étant plus abstrait et plus large qu'un langage objet comme Java, il s'agit de traduire un diagramme de classe en un code Java
- Les classes UML donnent des classes Java
- Une interface UML donne une interface Java
- Un héritage simple entre 2 classes Parent et Enfant s'implémente avec
  - 1 classe parente Parent
  - 1 classe Enfant qui « étend » (extends) Enfant

# Exercice

- Reproduire sous Visual Paradigm la classe suivante
- Traduire sous Android Studio, en Java, la classe suivante
- Préciser la visibilité de chaque attribut et opération

Salarie
+nomSalarie : string ~serviceSalarie : string #dateEmbauche : long -salaire : double <u>-entreprise : string</u>
+travaille(int) : void <u>+seReunir() : string</u>
Powered By Visual Paradigm Community Edition 

# Association unidirectionnelle



**En Java, cela donne:**

```
public class ClasseA {  
  
    private ClasseB maClasseB;  
    ...  
}
```

**Pour la classe B**

```
public class ClasseB {  
  
    // la classe B ne connaît pas  
    // la classe A  
}
```



# Association bidirectionnelle 1 - 1



**En Java, cela donne:**

```
public class ClasseA {  
  
    private ClasseB roleB;  
    ...  
}
```

**Pour la classe B**

```
public class ClasseB {  
  
    private ClasseA roleA ;  
    ...  
}
```

# Association unidirectionnelle 1 - \*



En Java, cela donne:

```
public class ClasseA {  
  
    private Set<ClasseB> roleB  
        = new HashSet<ClasseB>();  
    ...  
}
```

Pour la classe B

```
public class ClasseB {  
  
    // la classe B ne connaît pas  
    // la classe A  
    ...  
}
```

# Association bidirectionnelle 1 - \*



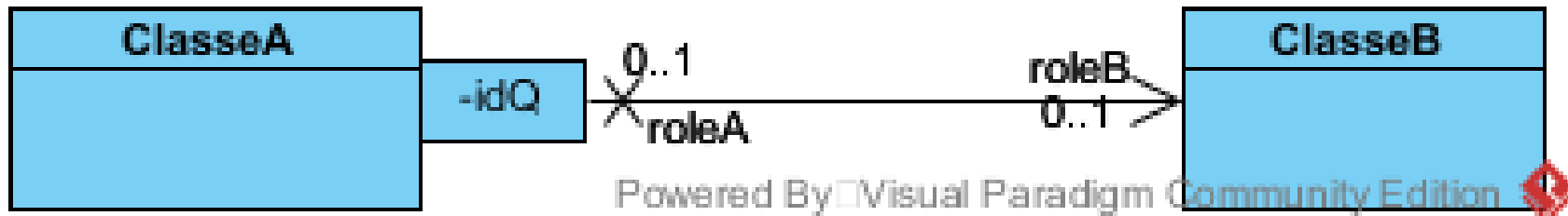
En Java, cela donne:

```
public class ClasseA {  
  
    private Set<ClasseB> roleB  
        = new HashSet<ClasseB>();  
    ...  
}
```

Pour la classe B

```
public class ClasseB {  
  
    private ClasseA roleA ;  
    ...  
}
```

# Association qualifiée



En Java, cela donne:

```
public class ClasseA {  
  
    private Map<idQ,ClasseB> roleB  
        = new HashMap<idQ,ClasseB>();  
    ...  
}
```

Pour la classe B

```
public class ClasseB {  
  
    ...  
}
```

# Agrégation et Composition

- Les **agrégations** sont implémentées comme des associations
- Une composition s'implémente comme une association unidirectionnelle