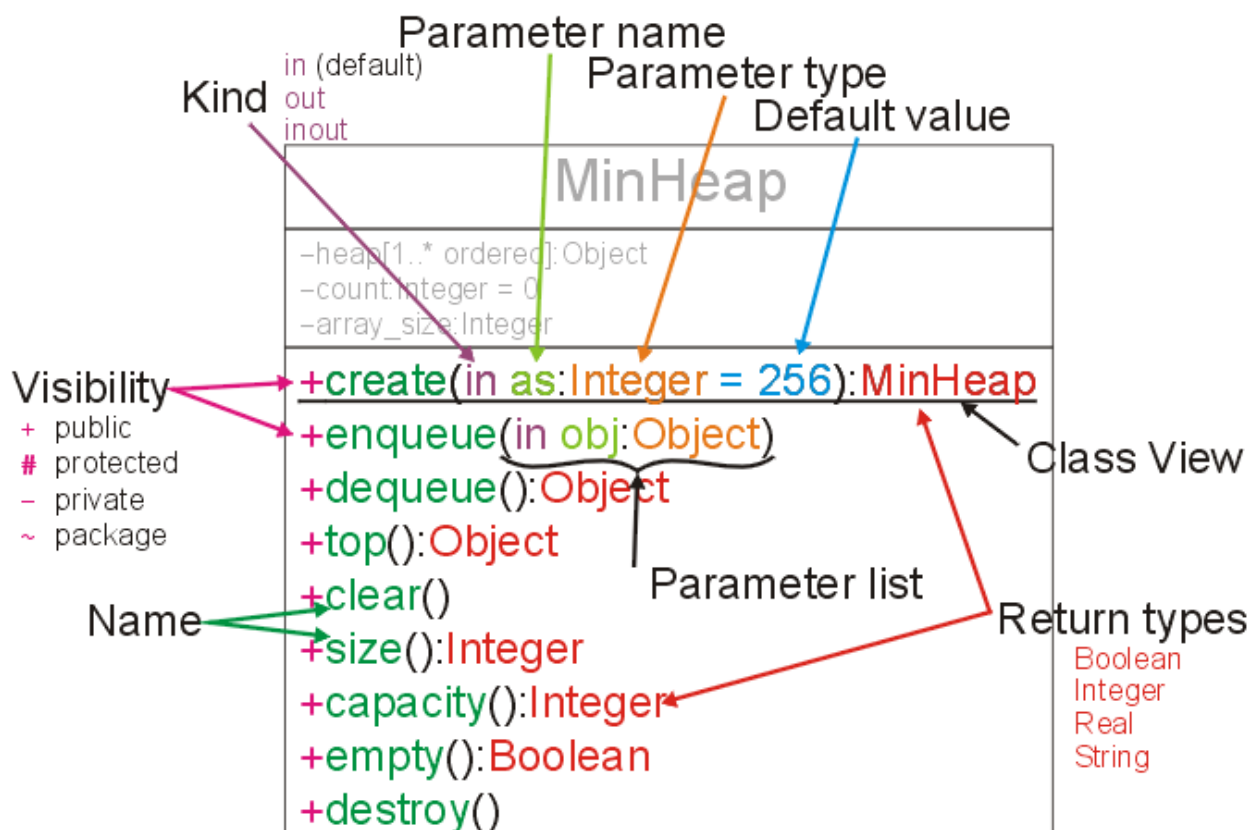
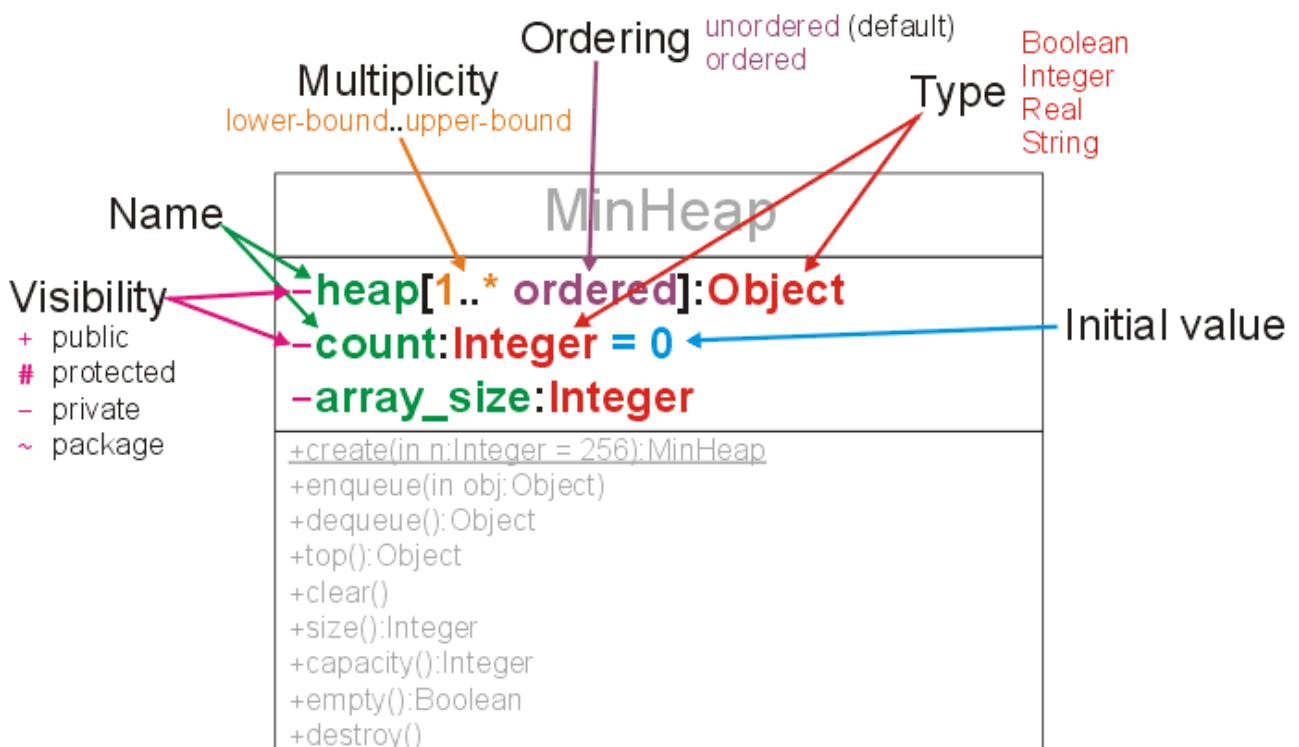


Concepts objet et éléments de notation UML

Diagramme de classes

Exemples de représentation d'une classe avec UML



Remarque : seul le nom de la classe est obligatoire et la représentation des types (attribut, méthode, paramètre, ...) est optionnelle.

1. Gestion des dépendances

La recherche de la qualité interne en programmation tient en quelques mots : *Gestion des dépendances* (dépendance entre classes, dépendances entre paquetages). L'objectif étant de limiter les dépendances entre parties implémentées, pour une meilleure réutilisation, maintenance et flexibilité. C'est un des objectifs majeurs en génie logiciel.

Le code source ne nous permet pas toujours de distinguer clairement les dépendances entre des classes. Les clauses import donne des indications, mais sont parfois abusives.

Comment représenter ces dépendances ? par un diagramme UML, le plus souvent un diagramme de classes et de composants (pour les vues statiques). UML devenant ainsi un outil d'aide à la décision, à l'analyse.

Un élément A est dépendant d'un élément B lorsque la définition de A inclut celle de B.

Il y a dépendance chaque fois qu'une relation existe entre des éléments.

2. Représentation des relations avec UML

UML définit quatre relations entre éléments : Association, Généralisation, Réalisation et Dépendance.

2.1 Association

Association : Relation structurelle qui décrit un ensemble de liens, un lien étant une connexion entre instances.

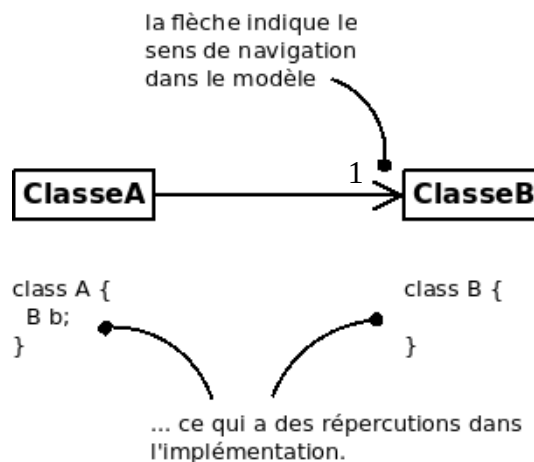


Exemple :

```
class A { B b; }
class B { A a; }
```

2.1. A Association et navigabilité

Il peut être utile parfois de préciser la **navigabilité** dans le modèle : *qui utilise qui ?*



Remarque : un lien bidirectionnel sera souvent représenté par un simple trait – et non pas par une flèche à deux sens)

2.1. B Association : rôle et multiplicité

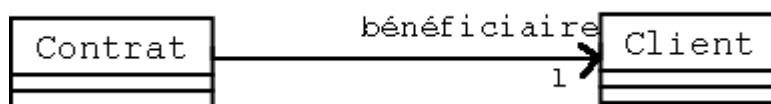
Exemple : Un contrat concerne un client.

```
class Contrat {  
    Client bénéficiaire;  
    ...  
}
```

Le bénéficiaire est une des caractéristiques d'un contrat ; la relation est structurelle (stable dans le temps)

Cette relation (l'association) est représentée par un **trait plein**, pouvant être **orienté**. La **multiplicité** est notée du côté du rôle cible de l'association. Elle spécifie le nombre d'instances pouvant être associées (liées) avec une seule instance source de la relation. Dans la phrase *un contrat concerne un client*, *contrat* est la source et *client* est la destination. Le **rôle** des instances de Client dans l'association est noté sur le lien (ici *bénéficiaire*), à côté de la multiplicité (à ne pas confondre avec le *nom* de l'association, qui s'inscrit au milieu du lien).

Exemple d'association orientée



Un contrat concerne un client.

```
class Contrat {  
    Client bénéficiaire;  
    ...  
}
```

Une multiplicité est exprimée soit par un couple de valeur N..M soit par une seule valeur lorsque N est M sont égaux.

Tableau Multiplicité

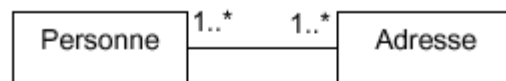
Exemple	Interprétation
1..1 ou 1	Un et un seul
0..1	zéro ou un seul
0..* ou *	zéro à plusieurs
3..4	trois à quatre
4	quatre et seulement quatre

Note : un lien relie une classe soit à elle-même (relation réflexive) soit à une autre classe. Au delà de 2 classes participant à la relation, les traits partent de chacun des rôles et convergent vers un même losange (référence à une classe association, concept non étudié ici).

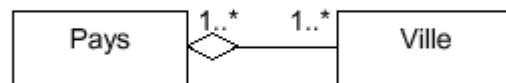
2.1. C Parmi les **associations**, on distingue les agrégations et compositions.

Association :

Relation structurelle qui décrit un ensemble de liens, un lien représentant une connexion entre objets. (Contrainte de stabilité)



Agrégation : Forme spéciale d'association qui définit une relation tout-partie entre l'agrégat (le tout) et un composant (la partie). Particularité : L'agrégation peut être partagée.



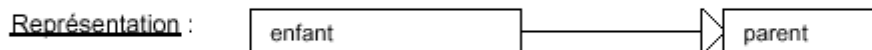
Composition : Forme d'agrégation qui véhicule des notions de forte priorité et de vie coïncidente des parties par rapport au tout. Deux contraintes : 1/ La durée de vie du composant est, au plus, égale à celle de son agrégat. 2/ Non partageabilité du composant avec d'autres objets. (une seule association de cardinalité 1 ou 0..1 du côté de l'agrégat).



Remarque : Agrégation et Composition ne sont pas toujours représentées dans un diagramme de classes.

2.2 Généralisation

Généralisation : Relation de spécialisation/généralisation, dans laquelle les objets de l'élément spécialisé (l'enfant) peuvent remplacer les objets de l'élément général (le parent).

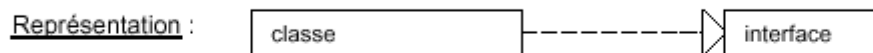


Exemple :

```
class Parent { }  
class Enfant extends Parent { }
```

2.3 Réalisation

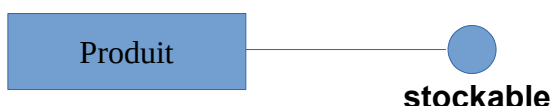
Réalisation : Relation sémantique entre classificateurs, dans laquelle un classificateur définit un contrat dont l'autre classificateur garantit l'exécution.



Exemple :

```
interface Stockable { public int getQte(); }  
class ProduitStockable implements Stockable {  
    int qte;  
    public int getQte(){ return this.qte; }  
}
```

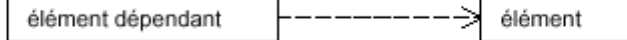
Notation simplifiée : Une interface peut être représentée par un **petit cercle** avec le nom de l'interface placé juste en dessous. Le cercle peut être attaché à une ou plusieurs classes d'implémentation. Exemple :



2.4 Dépendance

Dépendance : Relation sémantique entre deux éléments pour lesquels la modification de l'un peut influencer sur la sémantique de l'autre.

Représentation :



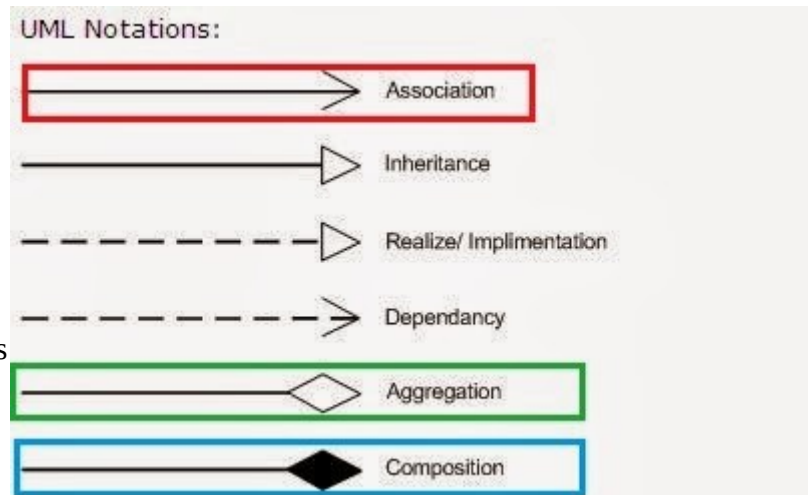
Exemple :

```
class Stock {  
    void sortir(ProduitStockable p) {  
        ...  
    }  
    ...  
}
```

En conclusion

De base UML propose un nombre restreint de relations entre classe que tout développeur doit maîtriser :

Doit-on pour autant représenter toutes les relations existantes entre les éléments présents dans un diagramme ?



Pour répondre à cette question, intéressons nous à la notion de **diagramme bien structuré**.

Un diagramme n'est pas forcément une représentation exhaustive d'un modèle. Un diagramme peut cacher des parties du modèle si cela sert la communication.

Nous reprenons ici la définition des "devoirs" d'un diagramme bien structuré, selon le "*guide de l'utilisateur UML*" [GUML] :

Un diagramme bien structuré doit :

- Être centré sur la communication d'un aspect d'une vue du système.
- Ne contenir que les éléments essentiels à la compréhension de cet aspect.
- Fournir les détails cohérents avec son niveau d'abstraction (ne montrer que les décorations essentielles à sa compréhension).
- Ne pas être trop minimaliste afin de ne pas désinformer le lecteur sur une sémantique importante.

Enfin, Martin Fowler [FOWLER] définit le **principe de modélisation** suivant :

Les modèles ne sont pas justes ou faux; ils sont seulement plus ou moins utiles.

Exercice de rétro-conception

Vous concevrez un diagramme de classes UML mettant en avant les relations significatives entre classes décrites par l'extrait de code java ci-dessous.

Extrait du code source :

```
public interface Observer {
    public void update(Observable o);
}

public class Observable {
    Collection<Observer> observateurs;

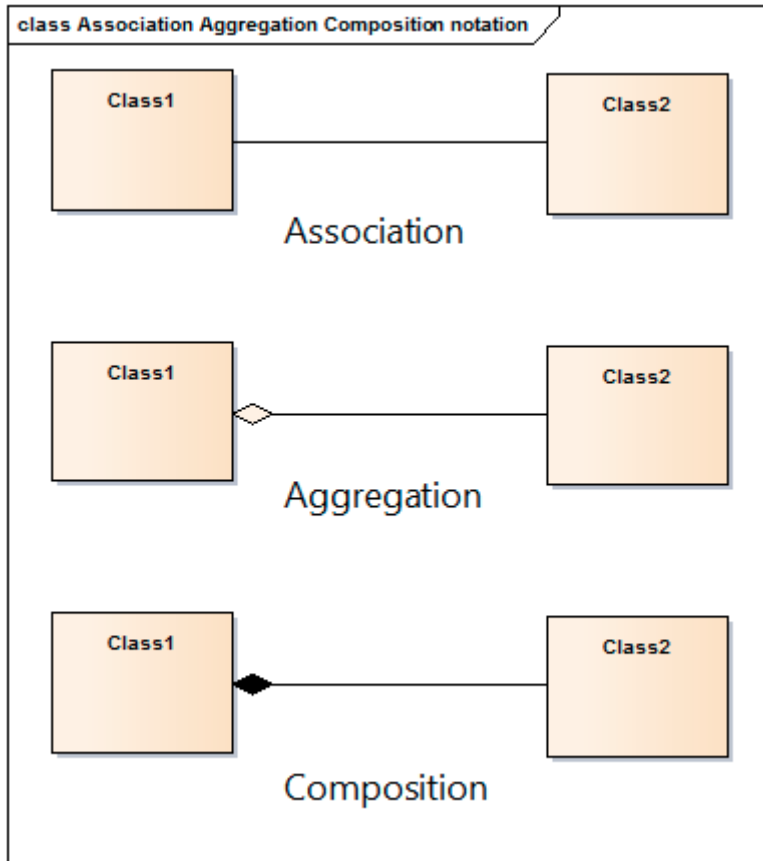
    public void notify() {
        for (Observer obsv : observateurs)
            obsv.update(this);
    }

    public void addObserver(Observer o) { observateurs.add(o); }
    ...
}

public class Bilan extends Observable {
    void setChange() { notify(); }
    ...
}

public class UIGraphe implements Observer {
    public void update(Observable o) {
        if (o instanceof Bilan) {
            Bilan unbilan = (Bilan) o;
            double compteResultat = unbilan.getCompteResultat();
            ...
        }
    }
    ...
}
```

UML : Exercice : classe et types d'association



Association : expression de liens entre instances (objets). Ici de type Class1 et Class2. Exemple : Personne et Adresse.

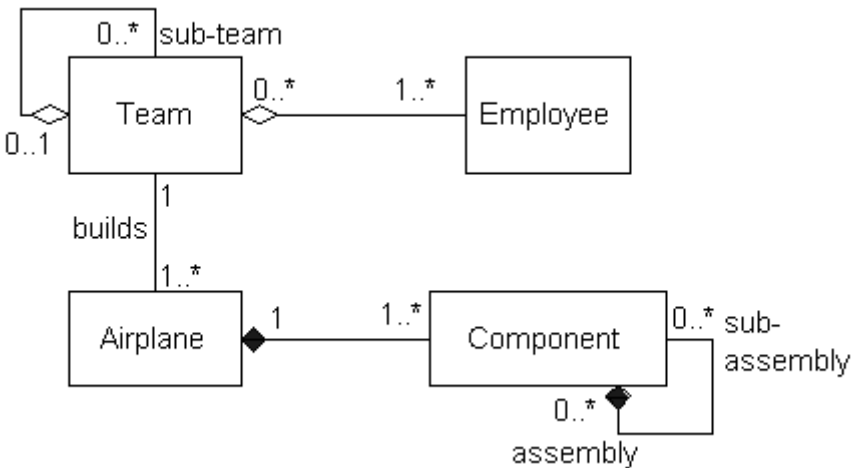
Agrégation : forme particulière d'association qui décrit une relation entre un tout et ses parties. Le losange vide côté du tout. Exemple : Voiture et ses roues.

Composition : forme particulièrement forte d'agrégation qui lit les composants à la chose composée : Le composant n'a pas de vie en dehors de sa relation à l'objet dont il est un constituant. Exemple : Fichier et ses octets.

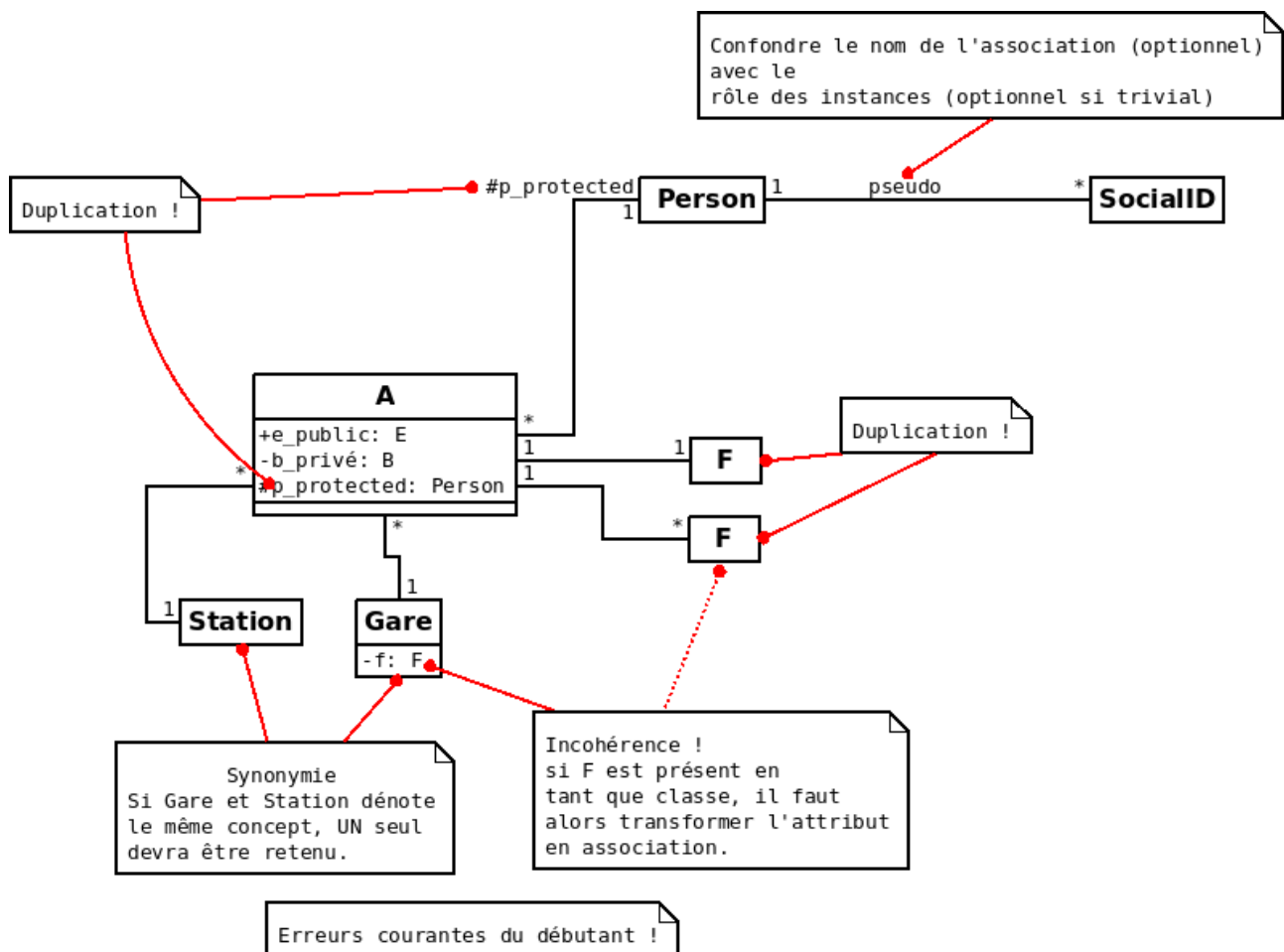
Exemples :

	Association	Agrégation	Composition
Voiture et son Moteur	[]	[]	[]
Immeuble et ses Salles	[]	[]	[]
Équipe et ses Employés	[]	[]	[]
Livre et ses Pages	[]	[]	[]
Vente et ses Produits	[]	[]	[]
Menu et ItemMenu	[]	[]	[]
Musicien et ses Instruments	[]	[]	[]

Exemple avec associations réflexives



Les erreurs courantes à ne pas commettre :



Version « corrigée »

