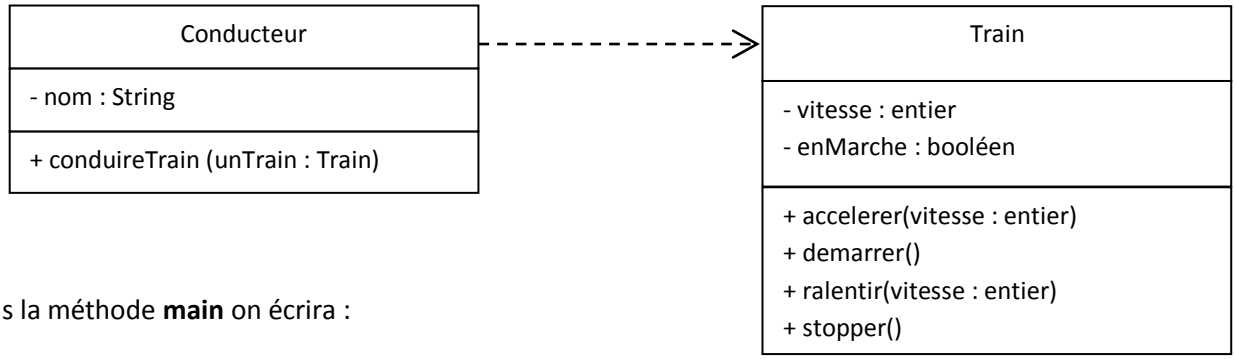
BLOC 2 - Ch. 10 – Relations entre classes

1 Les relations entre classes avec UML

1.1 Relation de dépendance

La **dépendance** est la relation la moins contraignante. Elle est unidirectionnelle et définit une relation d'utilisation.
On utilise le langage de modélisation **UML** (**Unified modeling language**) :



Dans la méthode **main** on écrira :

```
public static void main(String[] args)
{
      Conducteur unConducteur = new Conducteur("Joe") ;

      Train unTrain = new Train(0, false) ;

      // envoi d'un message paramétré au conducteur
      unConducteur.conduireTrain(unTrain) ;
}
```

Dans la classe **Conducteur** on écrira la méthode :

```
public void conduireTrain(Train unTrain)
{
      // le conducteur « utilise » le train
      unTrain.demarrer();
      unTrain.accelerer(100);
      unTrain.ralentir(50);
      unTrain.stopper() ;
}
```

Exercice 1 : Dans un TP, nous avons utilisé une classe **Compte** et une classe **Ecran.**
La classe Ecran a une méthode **afficher** qui permet d'afficher le contenu d'un compte :

```
      public static void afficher(Compte unCompte)
```

Travail à faire : Dessiner le diagramme de classes partiel avec les classes **Compte** et **Ecran** et la méthode **afficher(Compte unCompte)**.

1.2 Relation d'association

1.2.1 L'association entre deux classes

Au niveau des classes, nous pouvons construire une **association**. Une association relie des classes dont les instances sont sémantiquement liées. Une association ne sous-entend aucune hiérarchie dans la relation. Une association porte un nom.

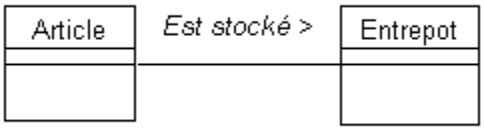


Diagramme de classe

L'association a un nom : "Est stocké", le symbole ">" indique seulement un **sens de lecture**.

L'association suivante est équivalente car elle concerne les mêmes couples d'objets :

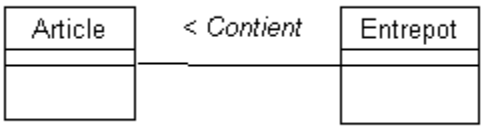


Diagramme de classe

Remarque : nous utilisons une forme verbale pour qualifier le nom de l'association.

1.2.2 Les classes jouent un rôle dans l'association

Le nom de l'association traduit la nature du lien entre les objets de classes : "Les lits sont stockés dans l'entrepôt de Montreuil". On peut être amené à préciser le rôle que joue chaque classe dans l'association, ici il s'agit des rôles de "gardien" côté entrepôt et "gardé" côté article.

Mais la notion de rôle prend tout son sens lorsqu'un couple de classes est relié par deux associations sémantiquement distinctes :

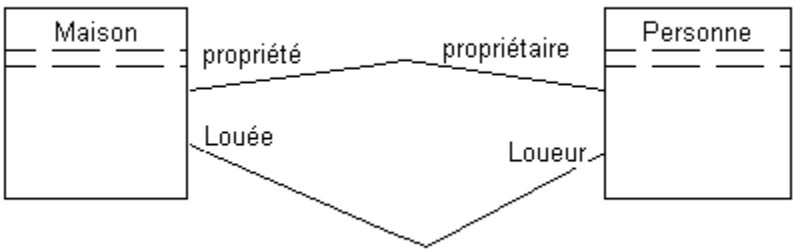
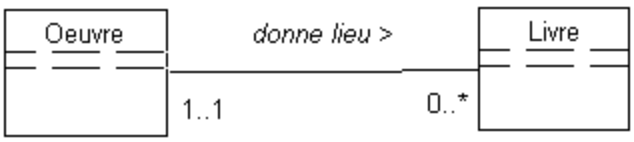


Diagramme de classe

1.2.3 Les associations comportent des cardinalités

Cette notion est identique à celle contenue dans Merise. Dans UML, on parle de **valeurs de multiplicité**.
Exemple 1 :

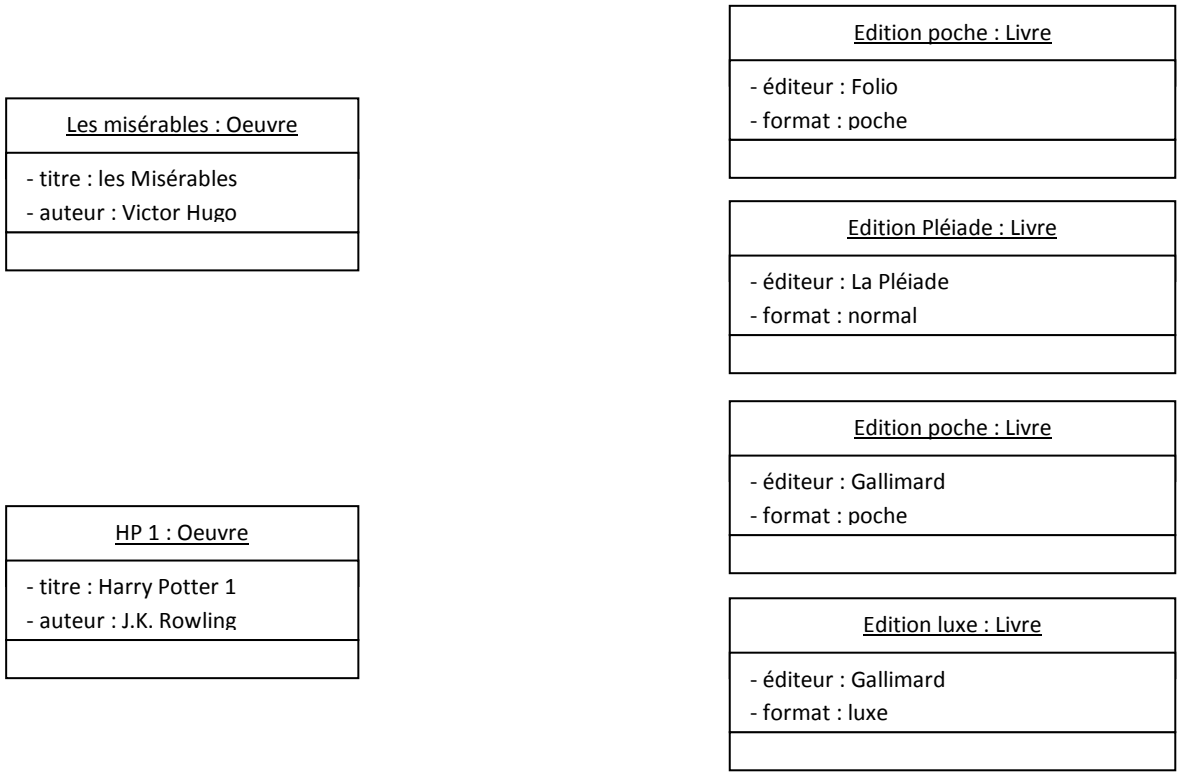


Interprétation :
Une œuvre donne lieu à 0 ou n livres. Un livre est issu d'une et une seule œuvre.

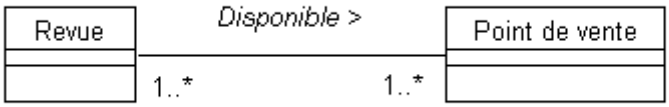
Commentaires :

- Notez le positionnement des multiplicités, **inverse de** la modélisation **Merise**
- La multiplicité **0..*** est parfois traduite par simplement le symbole ***

Activité : dans le diagramme d'objets suivant, reliez les livres aux œuvres correspondantes.



Exemple 2 :

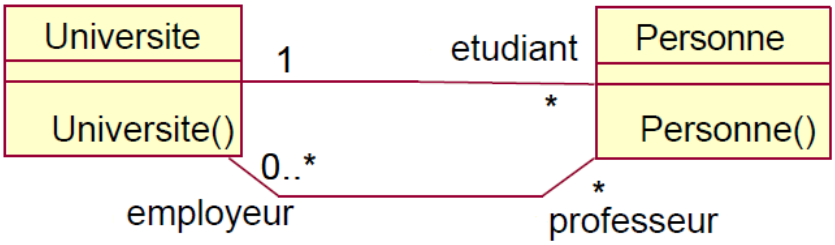


Interprétation :
Une revue est disponible dans un ou plusieurs points de vente.
Un point de vente distribue de une à plusieurs revues.
. . .

Valeurs de multiplicité :

1..1	Un et un seul
1	Un et un seul
0..1	Zéro ou un
m..n	De m à n
*****	De zéro à plusieurs
0..*	De zéro à plusieurs
1..*	De un à plusieurs

Exercice 2 : Interprétez le diagramme suivant, concernant une université :



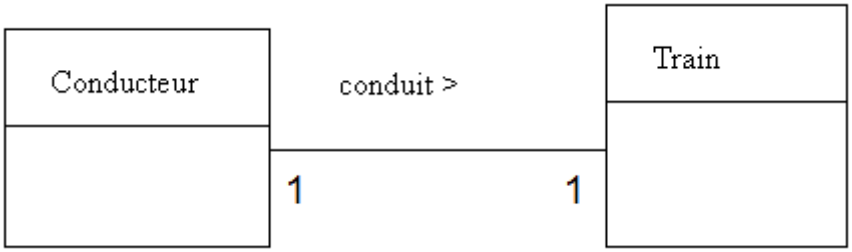
Exercice 3 : Dessiner le diagramme de classes partiel correspondant au cas suivant : une personne habite dans une seule maison ; elle peut travailler dans une entreprise, ou pas (si elle est chômeuse ou bien travaille à son compte). Une maison abrite plusieurs personnes. Une entreprise emploie plusieurs personnes (et au moins une).

2 Passage au code Java

Nous allons montrer les grandes lignes du passage entre un modèle UML et Java dans les cas les plus courants.

2.1 Cas d'association un à un

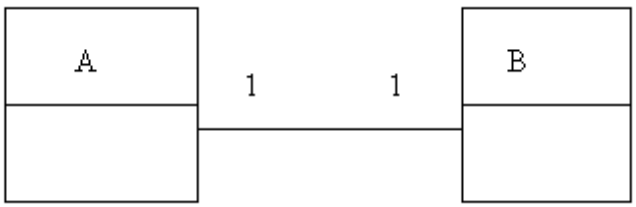
Considérons le diagramme ci-dessous :



Le code Java correspondant à ce diagramme sera :

```
public class Conducteur
{
    // le train associé au conduteur

    private . . .
}

public class Train
{
    // le conducteur associé au train

    private . . .
}
```

Cas général :




```
public class A
{
    private B leB;
}

public class B
{
    private A leA;
}
```

2.2 *Association de un à plusieurs*

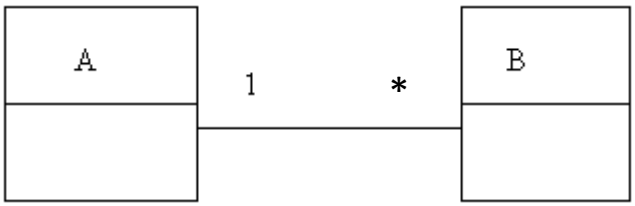
Activité : **Banque et comptes**

Une banque permet de gérer un ensemble de comptes bancaires. Voici le diagramme de classes partiel avec les deux classes **Banque** et **Compte :**

Travail à faire : complétez le code Java correspondant (uniquement les déclarations des classes et leurs attributs).

```
public class Banque
{
    // plusieurs comptes sont associés à la banque

    private . . .
}

public class Compte
{
    // une banque est associée au compte

    private . . .
}
```

Cas général



Avec une collection (ArrayList) :

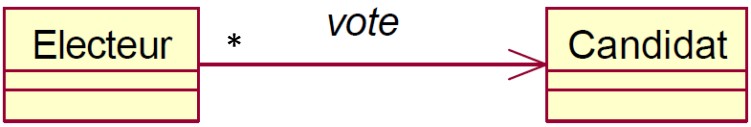
```
public class A
{
    private ArrayList<B> listeB = new ArrayList<B>();
}

public class B
{
    private A leA;
}
```

3 Navigabilité des associations

3.1 Notion de navigabilité des associations

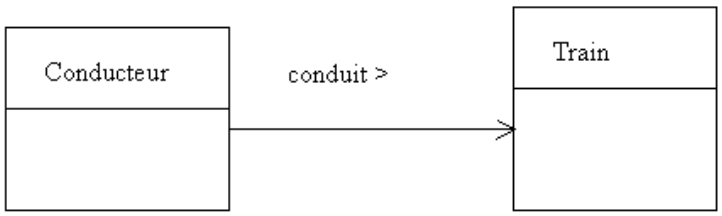
On peut assimiler une association à une relation permettant un parcours de l'information. On "passe" d'une classe à l'autre en parcourant les associations ; ainsi par défaut les associations sont "navigables" dans les deux sens.

Il peut être pertinent au moment de l'analyse de réduire la navigabilité à un seul sens : on parle de **navigabilité restreinte** :



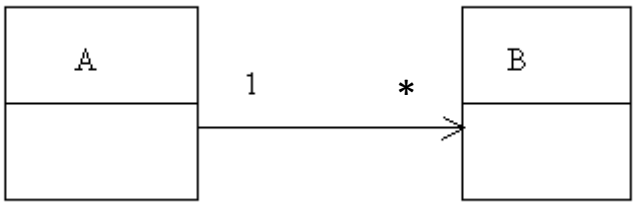
Commentaires :

- Un électeur connaît un candidat
- Un candidat ne connaît pas un électeur : on ne peut pas naviguer du candidat vers l'électeur.



Commentaire : on peut considérer que le train n'a pas besoin de "connaître" son conducteur.

3.2 Passage au code en cas de navigabilité restreinte




```
public class A
{
	private ArrayList<B> listeB = new ArrayList<B>();
}

public class B
{
	// pas de référence à un objet de la classe A
}
```

Exercice 4 : Jockey et Cheval.
Dans les haras de Lamballe, un jockey possède un cheval, et ce cheval n'appartient qu'à ce jockey :

Travail à faire : écrivez le code Java correspondant.

Exercice 5 : Bibliothèque.
Une bibliothèque propose plusieurs livres. Un livre fait partie uniquement de cette bibliothèque. Dessinez le diagramme de classes partiel. Ecrivez le code Java correspondant, utilisez des collections de type **ArrayList**.