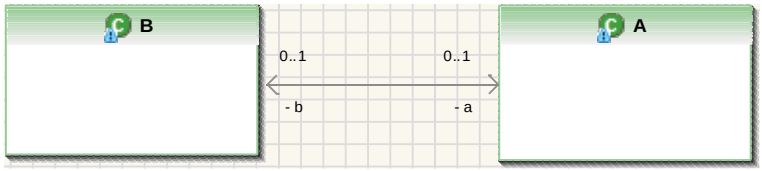
# TP 1 : du diagramme de classe UML à la mise en œuvre en Java

L'objectif de ce TP est de comprendre les principaux mécanismes pour la mise en œuvre en Java d'un diagramme de classe UML.

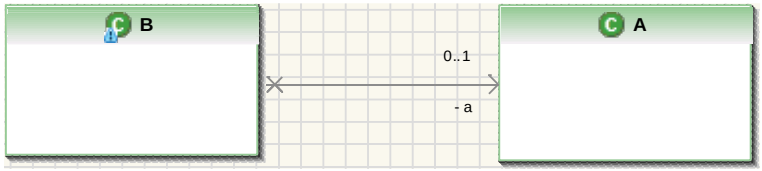
Consigne : Afin de bien voir les différences entre les différentes questions, créer un package par question.

## *Exercice 1 : Compréhension du mécanisme d'association*

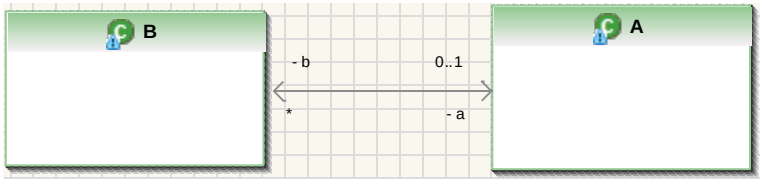
Q1: Mettre en œuvre en Java le diagramme de classe suivant :
Créer une class A et une classe B. Mettre en œuvre le fait que A joue le rôle a pour B et B joue le rôle de b pour A.




Q2: Mettre en œuvre en Java le diagramme de classe suivant :
Même exercice mais en empêchant la navigation de A vers B.




Q3: Mettre en œuvre cette liaison multiple à l'aide des types de données de la librairie standard et de la généricité. Justifier l'intérêt de la généricité dans ce cas.



Q4: Dans le cadre de la mise en œuvre, on souhaite offrir un mécanisme d'intégrité référentielle entre la classe A et la classe B. Proposer une mise en œuvre de ce mécanisme d'intégrité référentielle. Gérer le cas où l'attribut b de la classe A est mis à jour par une classe C.

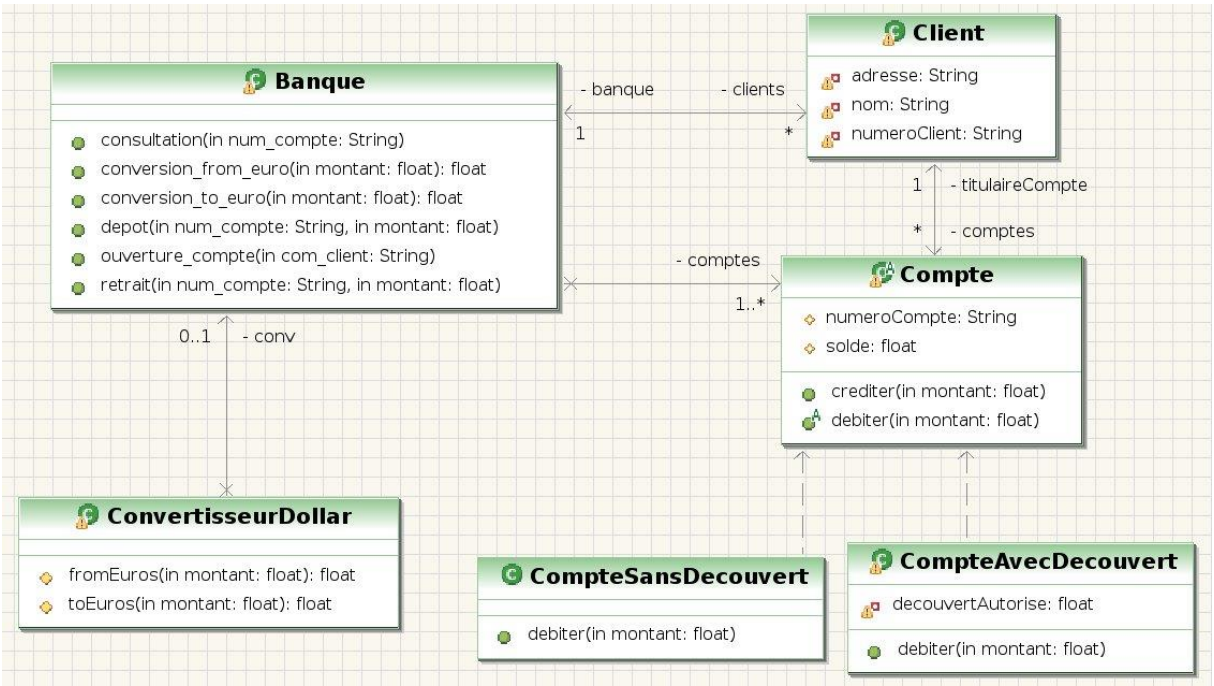
Q5: Proposer un mécanisme complet de gestion de relation n-n entre deux classes A et B.

## *Exercice 2 : Mise en œuvre du système d'information d'une banque*

Nous considérons dans cet exercice le cahier des charges suivant.

Chaque client peut posséder plusieurs comptes en banque. Un client est défini par un numéro, un nom et une adresse.
Un compte est défini par son numéro et le solde. Deux types de comptes existent, l'un autorisant un certain découvert au client, l'autre pas.
L'application contient aussi une classe *ConvertisseurDollar* qui permet de traduire les montants des transactions du Dollar vers l'euro et inversement.

**Q6:** Un de vos collègues vous fournit le diagramme de classe suivant. Après lui avoir notifié les trois erreurs dans ce diagramme, mettre en œuvre la structure de ce système d'information. Il n'est pas nécessaire de mettre en œuvre les méthodes de ces classes.



## *à méditer*

Pour prendre conscience de la difficulté du *mapping* entre UML et Java, vous pouvez réfléchir à vos heures perdues à la mise en œuvre de l'héritage multiple (autorisé en UML) en Java.