

# Analyse 1

ANA2

Geneviève Cuvelier  
(CUV)

Christine Leignel (CLG)

Thibaut Nicodème (TNI)

Pantélis Matsos (PMA)

# Où en sommes-nous ?

1. Qu'est-ce que l'analyse?
2. Diagramme d'activités
3. Les classes et objets
4. Les associations 1-1 et 1-N
5. Les associations N-N
6. Les compositions et énumérations
7. Les classes associations
8. L'héritage
9. Les interfaces

# Associations

## Définition

L'association exprime la connexion sémantique durable entre des classes.

# Associations – Exercice 1

Traduisez le code Java en diagramme UML.

```
public class Client {  
    private final String nom;  
    private final String prenom;  
  
    public Client(String nom, String prenom) {...}  
    public String getNom() {...}  
    public String getPrenom() {...}  
}
```

```
public class Compte {  
    private final String numero;  
    private int solde;  
    private final Client titulaire;  
  
    public Compte(String numero, int solde, Client titulaire) {...}  
    public String getNumero() {...}  
    public int getSolde() {...}  
    public Client getTitulaire() {...}  
}
```

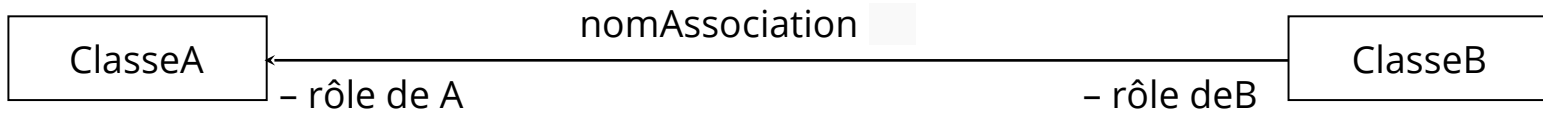
# Associations – Solution 1 – Partie 1

Client
<ul style="list-style-type: none"><li>– nom: String</li><li>– prenom: String</li></ul>
<ul style="list-style-type: none"><li>+ Client(nom: String, prenom: String)</li><li>+ getNom(): String</li><li>+ getPrenom(): String</li></ul>

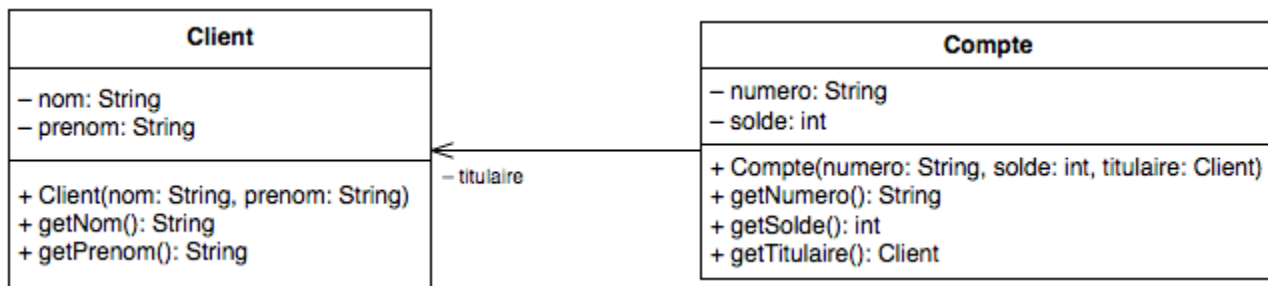
Compte
<ul style="list-style-type: none"><li>– numero: String</li><li>– solde: int</li><li>– titulaire: Client</li></ul>
<ul style="list-style-type: none"><li>+ Compte(numero: String, solde: int, titulaire: Client)</li><li>+ getNumero(): String</li><li>+ getSolde(): int</li><li>+ getTitulaire(): Client</li></ul>

Comment représenter l'association entre compte et client ?

# Associations



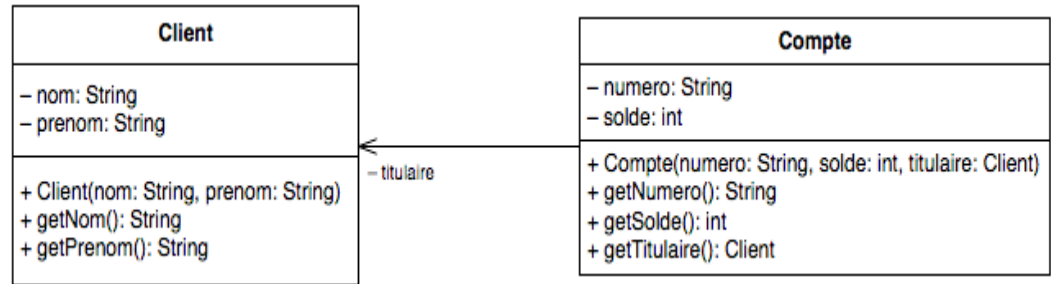
# Associations – Solution 1 – Partie 2



# Diagramme de classes en UML - Associations

La signification d'une association fléchée (unidirectionnelle) indique que l'accès est unidirectionnel.

Par exemple ici, on sais accéder au titulaire à partir d'un compte mais on ne peut accéder directement au compte à partir d'un client.





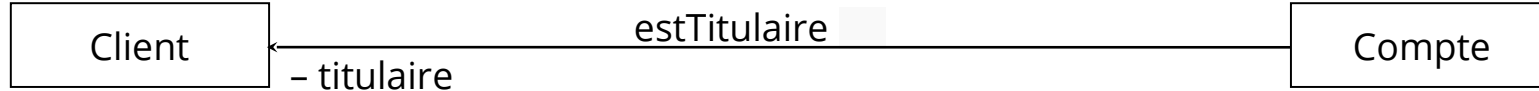
# Associations – Diagramme de classes en UML





Une association peut être définie par un **verbe** accompagné d'une flèche qui indiquant le sens, ou par un **rôle** d'une classe dans l'association.

L'association fléchée indique que l'accès est unidirectionnel. Par exemple ici, on peut d'un compte accéder au client, mais à partir d'un client on ne peut pas connaître son compte.

# Associations – Diagramme de classes en UML



Attention à ne pas confondre

- la flèche montrant le sens de lecture (  ) : cette flèche n'a aucune signification fonctionnelle. Elle ne sert qu'à faciliter la lecture du diagramme.
- la flèche de l'association dirigée (  ) : indique la navigabilité entre les objets des deux classes. Elle implique et remplace généralement un attribut, ici titulaire, de type de la classe cible dans la classe origine.

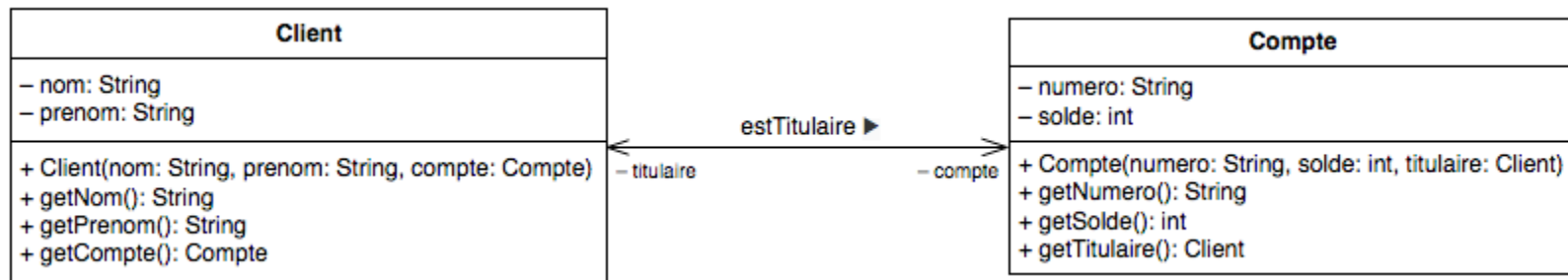
# Associations – Exercice 2

Traduisez le code Java en diagramme UML.

```
public class Client {  
    private final String nom;  
    private final String prenom;  
    private final Compte compte;  
  
    public Client(  
        String nom, String prenom, Compte compte  
    ) {...}  
    public String getNom() {...}  
    public String getPrenom() {...}  
    public Compte getCompte() {...}  
}
```

```
public class Compte {  
    private final String numero;  
    private int solde;  
    private final Client titulaire;  
  
    public Compte(String numero, int solde, Client titulaire) {...}  
    public String getNumero() {...}  
    public int getSolde() {...}  
    public Client getTitulaire() {...}  
}
```

# Associations – Solution 2



# Associations - Exercice 3

Écrivez le code des méthodes de l'exercice 2.

Quelles sont les avantages et les inconvénients d'une association bidirectionnelle ?

# Associations - Solution 3

On peut retenir un principe important:

La redondance de donnée est à éviter.

Elle entraîne des possibilités:

- d'incohérences de données
- de complexifier le code
- d'interblocage (Paradox de l'oeuf ou la poule)
- ...

# Diagramme d'objets en UML – Liens entre objets



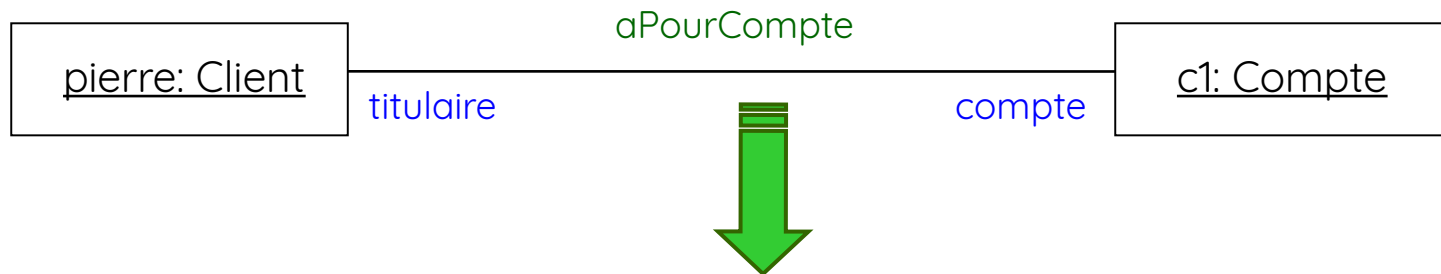
Les objets sont liés entre eux par des lignes pleines dans un diagramme.

Les noms des liens sont des **formes verbales ou nominales** et commencent par une minuscule.

indiquent le sens de la lecture (ex: « pierre est titulaire de c1 »)

# Rôles sur les liens

Chacun des deux objets joue un **rôle** différent dans le lien



- (1) pierre a pour compte c1
  - (2) c1 joue le rôle de compte pour pierre
  - (3) pierre joue le rôle de titulaire pour c1
- 
- (2.b) le [un des] compte[s] de pierre est c1
  - (3.b) le [un des] titulaire[s] de c1 est pierre

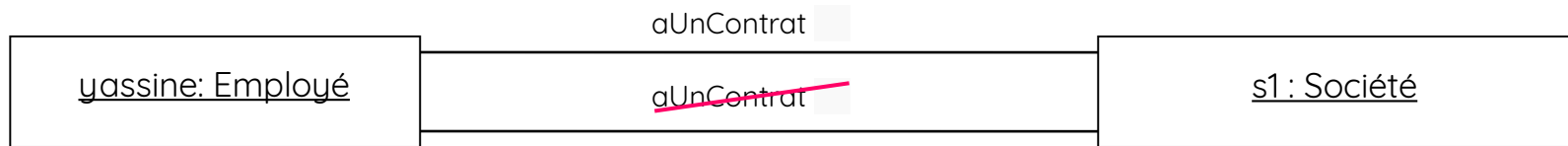
Note de style :

- choisir un groupe nominal pour désigner un rôle
- si un nom de rôle est omis, le nom de la classe fait office de nom de rôle



# Contrainte sur les liens

Au maximum un lien d'un type donné entre deux objets donnés



yassine a un contrat avec la société s1

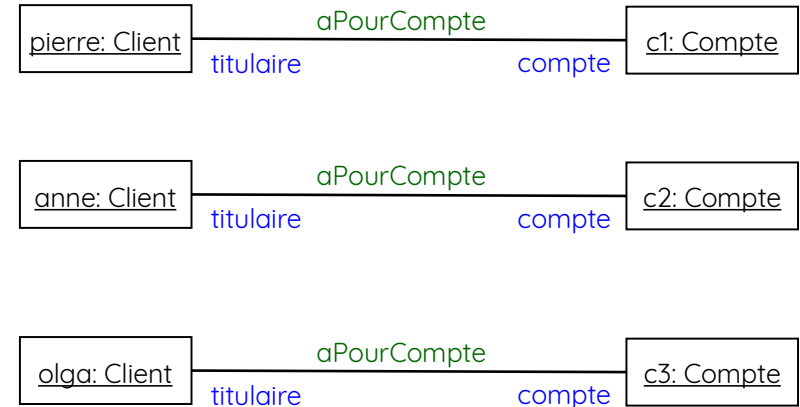
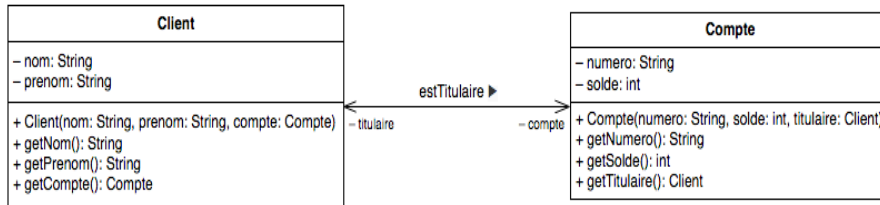
~~yassine a un contrat avec la société s1~~

# Différences entre les diagrammes de classes et d'objets

Diagramme d'objets	Diagramme de classes
<i>Instances</i> de classes	<i>Classes</i>
<i>Liens</i> entre les instances	<i>Associations</i> entre les classes

Un objet est une instance d'une classe  
et un lien est une instance d'une association.

# Relations entre les diagrammes de classes et d'objets



# Associations - Exercice 4 – Le lièvre et la tortue MVC

Traduisez le code Java en diagramme UML.

```
public class Tortue {  
    private int avancée;  
    public Tortue() {... }  
    public void avancer() {... }  
    public boolean estArrivée() {... }  
    public int getAvancée() {... }  
}
```

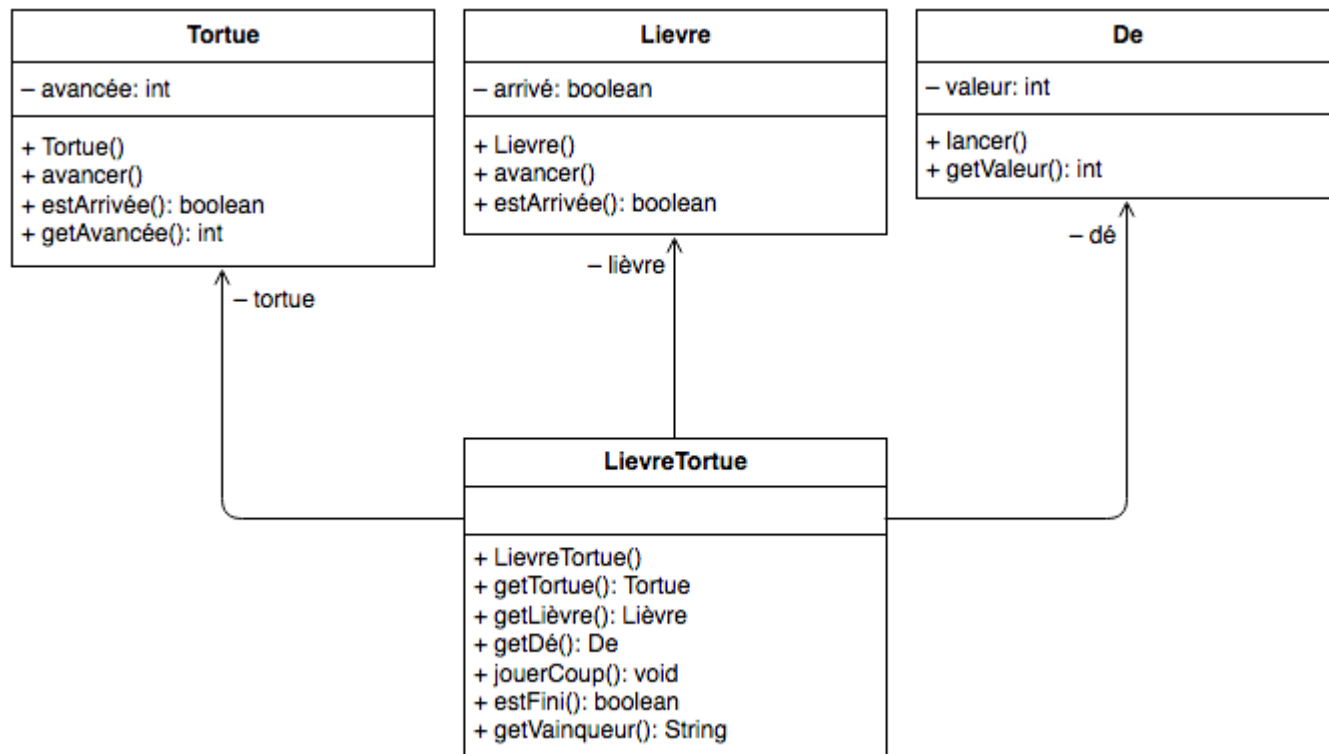
```
public class Lievre {  
    private boolean arrivé;  
    public Lievre() {... }  
    public void avancer() {... }  
    public boolean estArrivé() {... }  
}
```

```
public class LievreTortue {  
    private final Tortue tortue;  
    private final Lievre lièvre;  
    private final De dé;  
    public LievreTortue(){... }  
    public Tortue getTortue() {... }  
    public Lievre getLièvre(){... }  
    public De getDé(){... }  
    public void jouerCoup(){... }  
    public boolean estFini(){... }  
    public String getVainqueur(){... }  
}
```

```
import mcd.util.Hasard;  
public class De {  
    private int valeur;  
    public void lancer() {... }  
    public int getValeur() {... }  
}
```

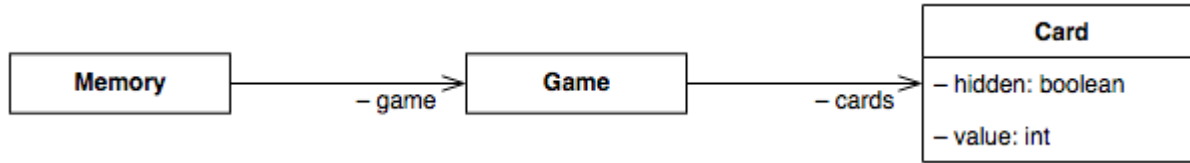
// ... voir [cours algo p. 27](#)

# Associations - Solution 4 – Le lièvre et la tortue MVC



# Associations - Exercice 5 – Retour sur le *memory*

Traduisez le diagramme de classes en code Java.



# Associations - Solution 5 – Retour sur le *memory*

```
public class Memory {  
    private final Game game;  
}  
  
public class Card {  
    private boolean hidden;  
    private int value;  
}  
  
public class Game {  
    private Card cards; /* une seule carte!!!*/  
}
```

Note : pour modéliser que cards est un tableau, on a besoin des multiplicités...