# TP2: L'Héritage

Nous souhaitons développer une application pour gérer des comptes dans une banque.
Nous allons commencer par créer un projet nommé **GestionComptes**..

**Exercice 1** : Créer la classe **Client** correspondant au schéma UML suivant :

| **Client** |
|---|
| nom : String<br>prenom : String<br>numeroTel : long |
| **Client(**nom : String, prenom: String, numero: long**)**<br>**getNom()**: String<br>**getPrenom()**: String<br>**getNumeroTel()**: String<br>**setNumeroTel(**numero :long**)**: void<br>**saisir()**: void<br>**afficher()**: void |

1. Déclarer des attributs nom, prenom et numeroTel comme dans le tableau ci-dessus. Ces attributs sont private pour assurer une bonne encapsulation.
2. Définir un constructeur prenant en paramètres nom, prenom et numeroTel.
3. Définir des accesseurs **getNom**(), **getPrenom()** et **getNumeroTel()** qui retournent respectivement les valeurs des attributs nom, prenom et numeroTel.
4. Définir un accesseurs **setNumeroTel**() qui permet de modifier la valeur d'attribut numeroTel.
5. Ajouter la méthode **saisir()** qui permet de saisir les informations d'un client.
6. Ajouter la méthode **afficher()** qui permet d'afficher les informations d'un client.

**Exercice 2** : Créer la classe **Compte** correspondant au schéma UML suivant :

| **Compte** |
|---|
| numero : long<br>solde : double<br>client : Client |
| **Compte(**numero: long, solde : double**)**<br>**Compte(**numero: long, solde : double, client: Client**)**<br>**getNumero()**: long<br>**getSolde()**: String<br>**getClient()**: Client<br>**deposer(**montant: double**)**: boolean<br>**retirer(**montant: double**)**: Boolean |

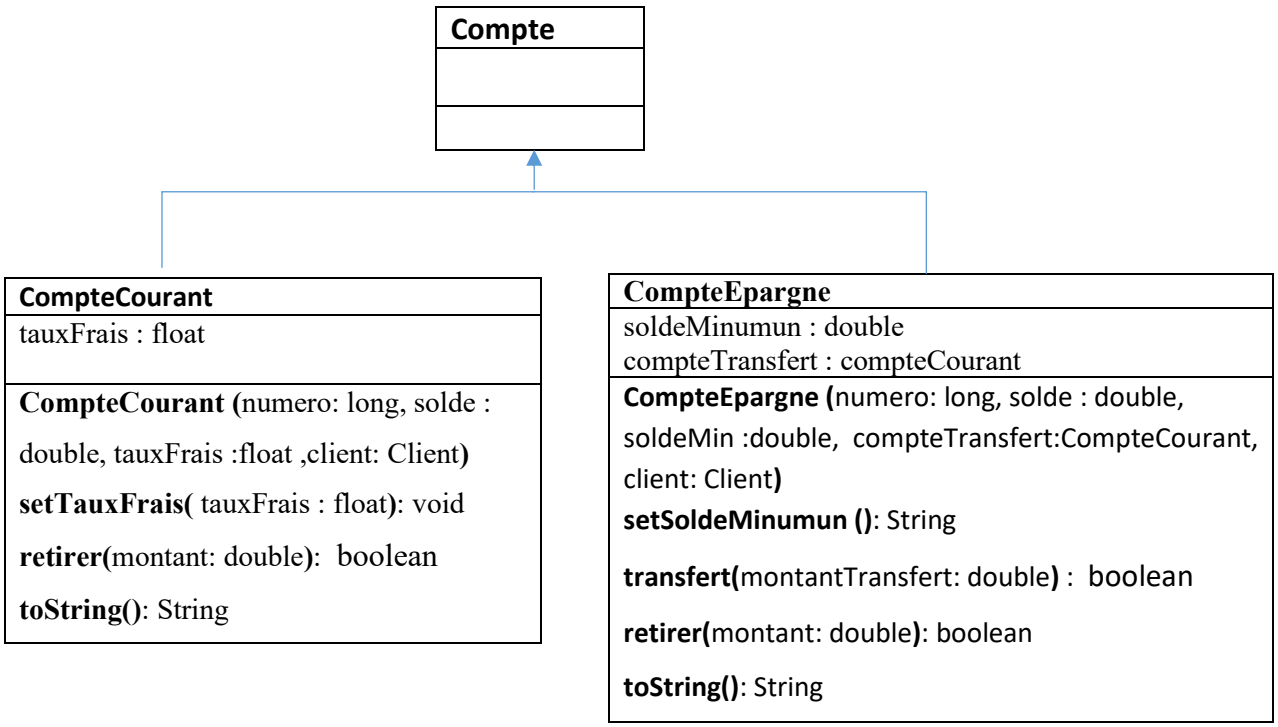
1. Déclarer des attributs numero, solde et client comme dans le tableau ci-dessus.
2. Définir un constructeur prenant en paramètres un numéro de compte et solde initial.
3. Définir un constructeur prenant en paramètres un numéro de compte, un solde initial et un client.
4. Définir des accesseurs **getNuméro**(), **getSolde()** et **getClient()** qui renvoient respectivement le numéro de compte, le solde et le client qui dispose du compte.
5. Définir une méthode **deposer** permettant de déposer un montant dans un compte.
6. Définir une méthode **retirer** permettant de retirer un montant d'un compte

**Exercice 3** : Dans la classe compte, nous redéfinissons la méthode **toString()** héritée de la classe Object (la classe Object la racine de l'arborescence d'héritage). La méthode toString() retourne « Compte numéro 12121 avec solde 120000 UM ».

En fait, nous pouvons distinguer 2 types de comptes spécialisés :

- Compte courant : un compte destiné aux transactions quotidiennes.
- Compte d'épargne : un compte de dépôt rémunérés qui permet des transactions limitées.

Créer les sous-classes **CompteCourant** et **CompteEpargne** héritées de la super-classe **Compte.** Utiliser le schéma UML ci-dessous pour créer les sous-classes :



## I. Compte courant

1. Créer une classe **CompteCourant** qui hérite de la classe **Compte**.
2. Ajouter un attribut **tauxFrais: int** qui représente le taux de frais qui doit être soustraits du solde à chaque opération de retrait.
3. Redéfinir la méthode **retirer** de telle sorte que les frais soient soustraits du solde.
4. Redéfinir la méthode toString() de telle sorte que l'affichage devienne : « Compte courant numéro … avec solde … ».

## II. Compte d'épargne

1. Créer une classe CompteEpargne qui hérite de la classe **Compte**.
2. Ajouter un attribut **soldeMinumun**: double qui représente le solde minimum requis pour maintenir un compte d'épargne.
3. Définir un attribut compteCourant permettant de faire référence au compte courant à partir duquel les transferts seront effectués.
4. Définir une méthode **transfert()** permettant de retirer du compte courant en utilisant le montant transféré.
5. Redéfinir la méthode retirer de telle sorte que le solde d'un compte d'épargne ne soit pas inférieur au **soldeMinimum**.
6. Redéfinir la méthode **toString**() de telle sorte que l'affichage devienne :« Compte d'épargne numéro … avec solde ... ».

**Exercice 4** : Chaque client peut posséder deux comptes ; un compte courant et un compte d'épargne**.**

1. Modifier la classe Client en ajoutant un tableau de deux comptes nommé **listComptes**. Ensuite, définir des accesseurs getter and setter pour l'attribut **listComptes.**
2. Dans la classe Client, définir une méthode **calculerSoldeTotal**() permettant de connaître le solde total d'un client, défini comme la somme des soldes de chacun de ses comptes.

**Exercice 5** : Pour lancer l'exécution de l'application, créez une classe TextExecution et ajoutez-y la méthode principale main.

Ensuite, testez les fonctionnalités de l'application avec les deux clients suivants :

1. Le client Mohamed Sidina (numéro de téléphone 36363334) propriétaire de 2 comptes :

   Un compte courant numéro 1001 avec un solde initial 100. Le taux de frais pour tous les comptes courants est 2.5%.

   Un compte d'épargne numéro 1002 avec un solde initial 20000, ce compte d'épargne est lié au compte courant numéro 1001. Le solde minimum pour tous les comptes d'épargnes est 150000.

2. Le client Oumar Jiddou (numéro de téléphone 46464434) propriétaire de 2 comptes :

   Un compte courant numéro 1003 avec un solde initial 300.

   Un compte d'épargne numéro 1004 avec un solde initial de 40 000, lié au compte courant numéro 1003.

3. Effectuer les opérations suivantes sur les comptes du client Oumar Jiddou :
   - Dépôt de 50000 sur le compte courant 1003.
   - Retrait de 30000 sur le compte courant 1003.
   - Transfer de 20000 du compte courant 1003 vers le compte d'épargne 1004.

4. Afficher les informations de chaque client, suivies des informations de ses comptes, en utilisant respectivement les méthodes **afficher**() et **toString**().

5. Afficher le solde total des comptes du client Oumar Jiddou.