TD/TP 3

L'objectif de ce TD est de mettre en place les concepts de l'héritage et du polymorphisme.

Exercice 1 (Gestion des comptes bancaires) :

Une agence bancaire a une adresse et un numéro d'agence incrémenté automatiquement.

Un client ne peut avoir des comptes que dans une seule agence de la banque. Il a un code incrémenté automatiquement, un nom, un prénom et une adresse. A sa création, le client doit préciser son agence bancaire.

Un compte bancaire possède à tout moment une donnée : son solde. Ce dernier peut être positif (compte créditeur) ou négatif (compte débiteur).

Chaque compte est caractérisé par un code et un solde. A sa création :

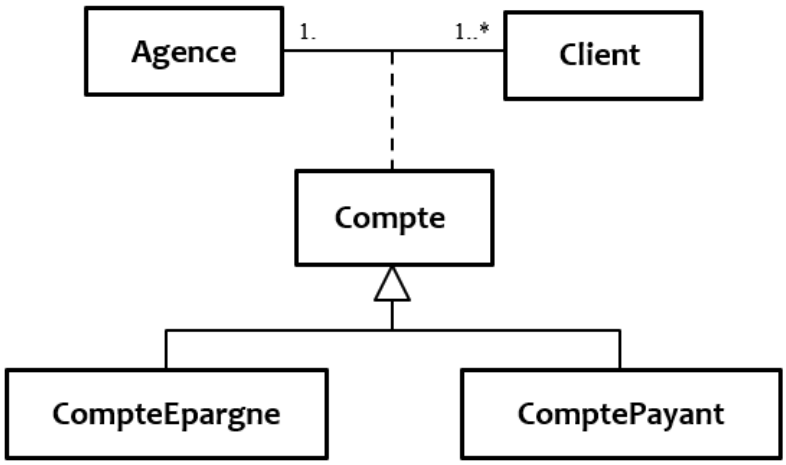
- Son code est incrémenté automatiquement, tandis que le solde peut être nul ou initialisé par une somme.
- Doit être associé à son propriétaire (un client).
- Préciser l'agence bancaire auprès de laquelle le compte a été ouvert.

Utiliser son compte consiste à pouvoir y faire des dépôts et des retraits. Pour ces deux opérations, il faut aussi consulter le solde de son compte par la méthode `getSolde()`.

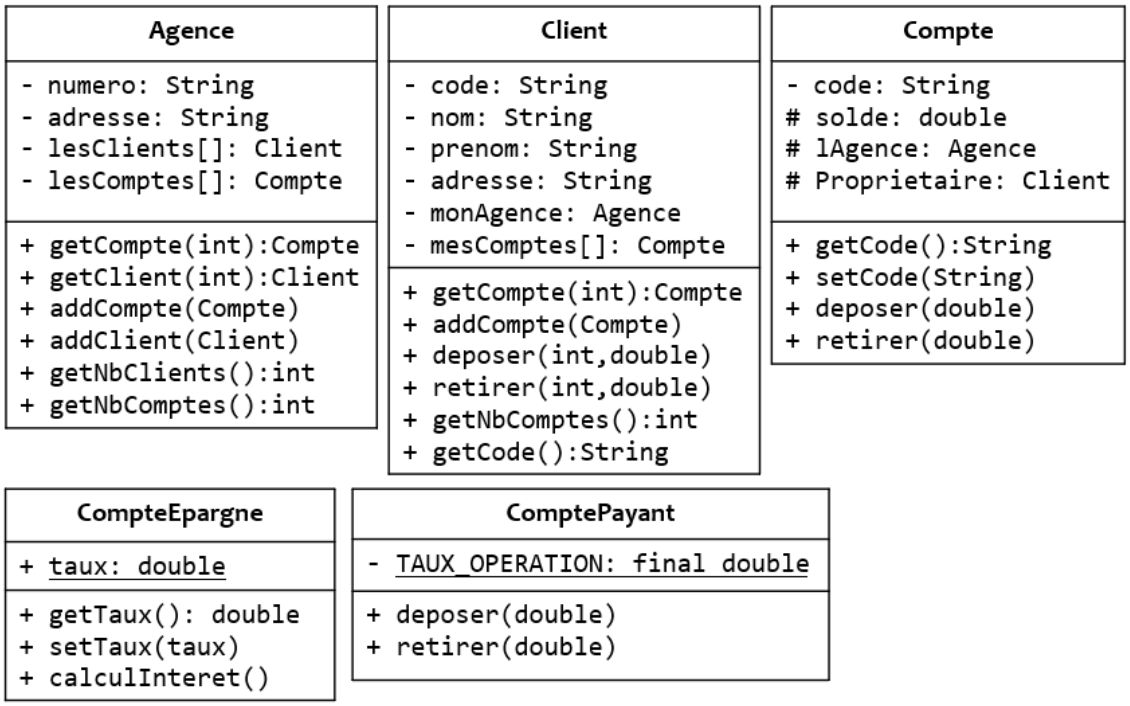
Les comptes sont de nature différente selon qu'ils soient payants ou d'épargne :

- Un compte Epargne est un compte bancaire qui possède en plus un champ "`TauxIntérêt=6`" et une méthode `calculIntérêt()` qui permet de mettre à jour le solde en tenant compte des intérêts.
- Un compte payant est un compte bancaire pour lequel chaque opération de retrait et de versement est payante et vaut 5dh.

Une première lecture de l'énoncé permettrait de modéliser ce système selon le diagramme de classes suivant :



Les classes à implémenter avec leurs membres :



Travail à faire :

1) Développer les différentes classes.

2) Chaque classe doit contenir au moins un constructeur d'initialisation. A la création d'un objet donné, l'identifiant (code/numéro) doit être initialisé et incrémenté automatiquement selon le format suivant :

- Identifiant = NomClasse:NuméroObjet
- Exemples d'identifiants des deux premières instances de type Client : **`Client:1`** et **`Client:2`**
- Utiliser les méthodes `getClasse()` et `getName()` pour une initialisation automatique.

3) Chaque classe doit redéfinir la méthode `toString()`.

4) Développer une classe `ApplicationBancaire`, dans cette classe on demande de :

a) Créer une seule agence bancaire.

b) Créer un tableau composé de 4 clients :

- Un client ayant un compte d'épargne (solde = 1000dh).
- Un client ayant un compte payant (solde = 2500dh).
- Un client ayant 2 comptes payants (soldes 0dh et 3000dh).
- Un client avec un compte d'épargne et un compte payant (soldes 2300dh et 0dh).

c) Permettre à un client de déposer une somme quelconque dans son/ses compte(s) en appelant la méthode `deposer(...)`.

d) Permettre à un client de retirer une somme quelconque dans son/ses compte(s) en appelant la méthode `retirer(...)`.

e) Définir la méthode `addClient(...)` permettant d'ajouter un client crée ainsi que ses comptes à l'agence. Utiliser cette méthode pour ajouter tous les clients crées.

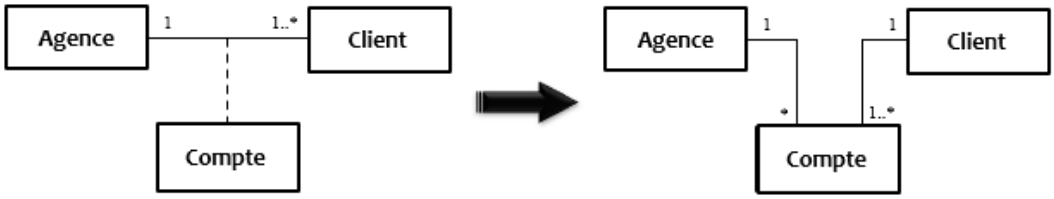
f) Appliquer la méthode `CalculIntérêt(...)` sur tous les comptes d'épargne de l'agence.

g) Afficher les informations suivantes :

- Liste des différents clients avec leurs différents comptes.
- Liste des comptes d'épargne de l'agence.
- Liste des comptes payants de l'agence.
- Le solde total des comptes d'un client.
- Classement des différents clients selon le solde total de chacun.

Remarques :

- Ajouter les éventuelles méthodes et/ou attributs nécessaires pour répondre aux questions ci-dessous.
- En ce qui concerne la classe association « `Compte` » et pour la coder correctement, penser à la remplacer par deux autres associations, comme illustré ci-dessous :



Exercice 2 (Gestion des salaires pour une PME) :

On souhaite développer un programme de calcul de salaire pour une entreprise PME, l'analyse du service « salaire » de cette entreprise a permis d'aboutir aux informations suivantes :

Information	Description
Matricule	Unique pour chaque employé et ne peut être modifié
Nom	Chaîne de caractères
Adresse	Chaîne de caractères
Salaire	Réel

1) Donner la déclaration de la classe Employé Comportant au moins un constructeur et les méthodes d'accès.

2) Cette entreprise comporte plusieurs types d'employés :

i. Des employés qui sont payés suivant le nombre d'heures qu'ils ont travaillé dans la semaine. Ils sont payés à un certain tarif horaire.

ii. D'autres employés, payés au forfait par jour.

iii. Les commerciaux sont payés avec une somme fixe pour la semaine.

Donnez les classes qui héritent de la classe `Employe` et redéfiniront la méthode `setSalaire()` pour chaque type des employés. Les classes comporteront deux constructeurs : un qui ne prend en paramètre que le matricule, le nom et l'adresse de l'employé et l'autre qui prend en paramètres en plus toutes les informations pour le calcul du salaire.

3) sUne classe `Paie` comportera une unique méthode `main()` qui entrera les informations sur des employés des différents types (3 commerciaux, 2 de type i.et 2 de type ii.).

- Les employées seront enregistrées dans un tableau Employé.
- Au moins un des employés sera créé avec le constructeur ne possédant comme paramètres que le Matricule, le Nom et l'Adresse et vous entrerez ensuite les informations pour son salaire avec la méthode `setSalaire()`.
- Pour un autre employé, vous utilisez le constructeur pour entrer les informations sur le salaire.

Ecrire une méthode qui affichera le salaire hebdomadaire de chacun des employés et la masse salariale Hebdomadaire des employés enregistrés dans le tableau.