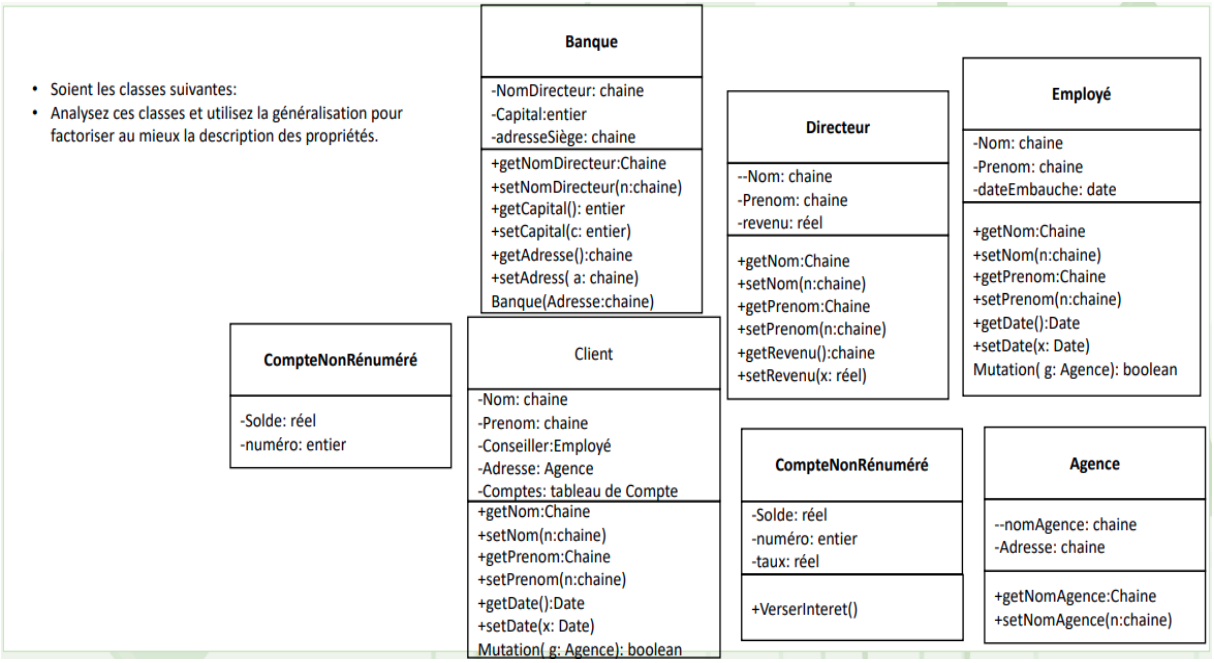
SERIE HERITAGE

EXERCICE 1 :



EXERCICE 2 :

1) Ecrivez une classe ***Bâtiment*** avec les attributs suivants: **adresse**.
La classe ***Bâtiment*** doit disposer des constructeurs suivants:
   - **Constructeur avec argument**

La classe ***Bâtiment*** doit contenir des accesseurs et mutateurs pour les différents attributs. La classe ***Bâtiment*** doit contenir une méthode **__str__()** donnant une représentation du ***Bâtiment***.

2) Ecrivez une classe ***Maison*** héritant de ***Bâtiment*** avec les attributs suivants:
   - **nbPieces: Le nombre de pièces de la maison.**

La classe ***Maison*** doit disposer des constructeurs suivants:
   - **Constructeur avec arguments**

La classe ***Maison*** doit contenir des accesseurs et mutateurs pour les différents attributs. La classe ***Maison*** doit contenir une méthode **__str__()** donnant une représentation de la ***Maison***.

3) Ecrivez une classe ***Immeuble*** héritant de Bâtiment avec les attributs suivants:
   - **nbAppart: Le nombre d'appartements de l'immeuble.**

La classe Immeuble doit disposer des constructeurs suivants :
   - **Constructeur avec arguments**

La classe ***Immeuble*** doit contenir des accesseurs et mutateurs pour les différents attributs. La classe ***Immeuble*** doit contenir une méthode **__str__()** donnant une représentation de l'***Immeuble***.

4) Ecrire un programme principal afin de tester les classes avec les différentes méthodes.

EXERCICE 3 :

Définir les classes suivantes:

1. Une classe **DateNaissance** avec trois attributs, **jour, mois, année**
   a. Ajouter un constructeur avec paramètres
   b. Ajouter la méthode **__str__()** qui convertit la date de naissance en chaine de caractères (jour/mois/annee)

2. Une classe **Personne** avec trois attributs, nom, prénom et date de naissance
    a. Ajouter un constructeur avec paramètres
    b. Ajouter la méthode **Afficher** pour afficher les données de chaque personne.
3. Une classe **Employé** qui *dérive* de la classe **Personne,** avec en plus un attribut ***Salaire***
    a. Ajouter un constructeur avec paramètres
    b. Redéfinir la méthode **Afficher** pour afficher les données de chaque employé.
4. Une classe **Chef** qui *dérive* de la classe **Employé,** avec en plus un attribut ***Service***
    a. Ajouter un constructeur avec paramètres
    b. Redéfinir la méthode **Afficher** pour afficher les données de chaque chef.

EXERCICE 4 :

On modélise une application devant servir à l'inventaire d'une bibliothèque. Elle devra traiter des documents de nature diverse : des livres, des dictionnaires, et autres types de documents qu'on ne connaît pas encore précisément mais qu'il faudra certainement ajouter un jour (articles, bandes dessinées...). Tous les documents possèdent un numéro d'enregistrement et un titre. A chaque livre est associé, en plus, un auteur et un nombre de pages, les dictionnaires ont, eux, pour attributs supplémentaires une langue et un nombre de tomes. On veut manipuler tous les articles de la bibliothèque au travers de la même représentation : celle d'un document.

A. La classe Document :
    1. Définissez la classe Document
    2. Ajouter un constructeur à la classe Document
    3. Ajouter les accesseurs et mutateur de la classe Document
    4. Définir une méthode qui retourne une chaine de caractère représentant les différents attributs de l'objet

B. La classe Livre :
    1. Définissez la classe Livre
    2. Ajouter un constructeur à la classe (vérifier si le nombre de page est inférieur ou égale à zéro)
    3. Ajouter les accesseurs et mutateur de la classe Livre
    4. Définir une méthode qui retourne une chaine de caractère représentant les différents attributs de l'objet

C. La classe Bibliothèque est caractérisée par un nom et une liste de document:
    1. Définissez une classe Bibliothèque
    2. Ajouter un constructeur pour initialiser le nom
    3. Définir la méthode ajouter(doc) qui permet d'ajouter le document doc à la liste
    4. Ajouter la méthode afficher() qui permet d'afficher tous les documents de la liste
    5. Ajouter la méthode rechercher(numero) qui permet de rechercher un document par numéro. s'il existe, la fonction retourne sa position, sinon elle retourne -1