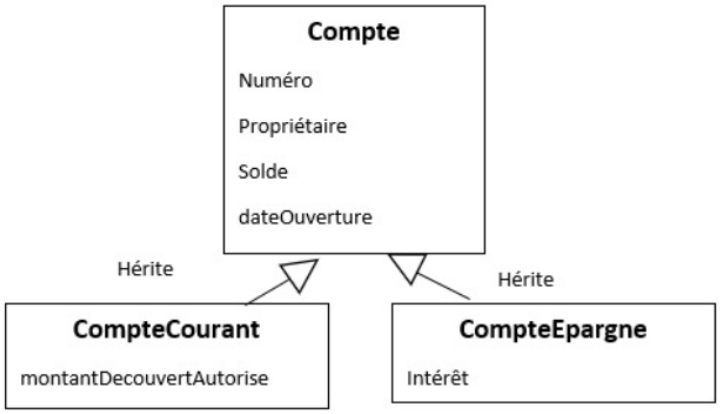
Travaux Pratiques : Gestion des Comptes Bancaires

1. Objectif

L'objectif de ce TP est de manipuler la **programmation orientée objet** en Python et de structurer le code en **modules**. Vous allez concevoir un **système de gestion de comptes bancaires** en respectant des règles de modélisation et de bonnes pratiques de programmation.



2. Modélisation des classes

1. Créez trois modules : **Compte.py**, **CompteCourant.py** et **CompteEpargne.py**, qui définissent respectivement les classes **Compte**, **CompteCourant** et **CompteEpargne**.
2. Implémentez un **constructeur** pour chaque classe en respectant les attributs du diagramme UML.
3. Écrivez des **getters** pour chaque attribut.
4. Définissez la méthode **___str___()** pour chaque classe afin d'afficher les informations du compte sous forme de chaîne de caractères.

3. Gestion des transactions et exceptions

1. Ajoutez une méthode **deposer(montant)** dans la classe **Compte** qui permet d'ajouter un montant au solde.

2. Ajoutez une méthode **retirer(montant)** dans **CompteCourant** qui permet de retirer de l'argent en respectant la limite du découvert autorisé.

3. Gérez les exceptions suivantes :
   - Interdire un **dépôt négatif**.
   - Interdire un **retrait négatif**.
   - Lever une exception **SoldeInsuffisantError** si un retrait dépasse le solde disponible (en tenant compte du découvert pour **CompteCourant**).
   - Lever une exception **ValeurInvalideError** si un montant saisi n'est pas un nombre positif.

4. Questions logiques et manipulation de chaînes

1. Ajoutez une méthode **calculer_interet()** dans **CompteEpargne** qui applique un **taux d'intérêt** au solde.
   - Le taux doit être **positif** et **inférieur à 1** (ex : 0.05 pour 5- Si le taux est invalide, lever une exception.

2. Implémentez **verifier_numero_compte()** dans **Compte**, qui valide le **format du numéro de compte** :
   - Doit commencer par **"CB-"** suivi de **6 chiffres** (ex : "CB-123456").
   - Utilisez une **expression régulière** pour vérifier ce format.

5. Test et validation

1. Créez un fichier **main.py** qui :
   - Crée un **CompteCourant** et un **CompteEpargne**.
   - Effectue des **dépôts** et **retraits** en capturant les erreurs possibles.
   - Vérifie la **validité des numéros de compte** avec **verifier_numero_compte()**.
   - Affiche les **détails des comptes**.