

## TD : Gestion de Comptes Bancaire

---

### Modèle de l'application :

---

Écrire une classe **Compte** définissant un compte bancaire :

Attributs : **idCompte**, **solde**, **date de création**, **tableau de journalisation** et un **propriétaire** (Client)  
Accès : getters, setters  
(**Solde initiale doit être positif**)  
(**id doit être auto-généré**)  
(**la journalisation de la création du compte est stocké dès l'initialisation du compte, ainsi que le dépôt du solde initiale si c'est pas 0dh**)

Méthodes : **public String** toString(),  
**public boolean** equals(**Object** autreCompte)

Écrire une classe **Client** définissant un Client de la banque :

Attributs : **idClient**, **nom**, **prénom**, **email**, **tableau de journalisation** et un **tableau de comptes**  
Accès : getters, setters  
(**email doit être conforme au format Email**)  
(**id doit être auto-généré**)

Méthodes : **public String** toString(),  
**public boolean** equals(**Object** autreClient)

Écrire une classe **Banque** lié par relation d'agrégation au deux classes Compte et Client :

Attributs : **idBanque**, **nomAgence**, **emailAgence**, **maxComptes**, **maxClients** et un **tableau de comptes**, **tableau de Clients**.  
Accès : getters, setters  
(**nombre max de Clients et Comptes doit être fixé dès la création de l'agence bancaire**)  
(**id de la banque doit être auto-généré**)

Méthodes : **public String** toString(),  
**public boolean** equals(**Object** autreBanque)

## Services de l'application :

---

Écrire une classe Service-Banque implémentant les différents services de l'application :

Attribut : **Banque**

Fonctions du Service :

- Service transactionnelle :: (**chaque service finit par une journalisation**)  
**public boolean** verser(**double** montant, **Compte** c)  
**public boolean** retirer(**double** montant, **Compte** c)  
**public boolean** virement(**double** montant, **Compte** src, **Compte** des)
  
- Service CRUD :: (création, consultation, modification et suppression)  
**public boolean** créerEtAjouterCompte(**Scanner** clavier)  
**public boolean** créerEtAjouterNouveauClient(**Scanner** clavier)  
**public boolean** lierCompteAuClient (**int** idClient, **int** idCompte)  
**public Compte** chercherCompte(**Scanner** clavier)  
**public Compte** chercherClient(**Scanner** clavier)  
**public void** consulterDetailCompte(**Scanner** clavier)  
**public void** consulterDetailClient(**Scanner** clavier)  
**public boolean** modifierCompte(**Scanner** clavier)  
**public boolean** modifierClient(**Scanner** clavier)  
**public boolean** supprimerCompte(**Scanner** clavier)  
**public boolean** supprimerClient(**Scanner** clavier)
  
- Service ++ ::  
**public void** consulterInformationsBanque(**Scanner** clavier)  
**public void** listerClientsDeLaBanque(**Scanner** clavier)
  - ➔ Ordonné par leur date d'ajout
  - ➔ Ordonné par leurs soldes de comptes (ordre croissant || décroissant)
  - ➔ Ordonné alphabétiquement (selon leur nom et prénom)**public Client[]** trierClientsParDate (**Scanner** clavier)  
**public Client[]** trierClientsParSolde (**Scanner** clavier)  
**public Client[]** trierClientsParNom (**Scanner** clavier)
  
- Service Utilitaire ::  
**public void** afficherMenuServiceBanque()  
**public Object[]** trierAsc(**Object[]** objets, **Object** orderByElement)  
**public Object[]** trierDec(**Object[]** objets, **Object** orderByElement)

**Test de l'application :**

---

Écrire une classe **Test-Service-Banque**, Créant une banque et la lier au service banque pour tester les différents opérations.