1 — Fondations de la POO

💼

Exercice Java – Compte ↔ Banque

PALIER 1 — Fondations de la POO

🎯

Objectifs pédagogiques

- Manipuler les attributs d'un objet à l'aide du mot-clé **this**.
- Appliquer l'**encapsulation** : rendre les attributs privés et y accéder via des **getters/setters**.
- Comparer deux objets correctement à l'aide de **equals()** et **hashCode()** (vs ==).
- Afficher proprement un objet avec **toString()**.
- Comprendre la **relation entre objets** (une banque contient plusieurs comptes, un compte "appartient" à une banque).

🧠

Contexte

Tu dois modéliser le fonctionnement basique d'une **banque** qui gère plusieurs **comptes bancaires**.

Chaque compte a :

- un **numéro unique** (ex: "BE123456"),
- un **solde** (float ou double).

La banque doit :

- pouvoir **ajouter** de nouveaux comptes,
- **afficher** tous les comptes,
- **rechercher** un compte via son numéro,

- **effectuer un transfert** d’argent entre deux comptes (⚡ variante).

🧩

Classes à créer

1.

Compte

Représente un compte bancaire individuel.

✳️ Attributs privés :

```
private String numero;
private double solde;
```

✳️ Constructeurs :

- Un constructeur avec paramètres (numero, solde).
- Un constructeur par défaut (optionnel, initialise avec des valeurs par défaut).

✳️ Méthodes :

- public String getNumero()
- public double getSolde()
- public void setSolde(double solde)
- public void crediter(double montant)
- public void debiter(double montant) (⚠️ vérifier que le solde reste positif)
- @Override public String toString() → retourne une chaîne du style : "Compte BE123456 – Solde : 2500.0 €"
- @Override public boolean equals(Object obj) → deux comptes sont égaux s’ils ont **le même numéro**
- @Override public int hashCode() → cohérent avec equals (basé sur numero)

💡 Utilise **this** dans ton code pour clarifier la référence à l’objet courant :

```
this.solde += montant;
```

2.

Banque

Représente une banque qui contient plusieurs comptes.

✳️ Attribut :

```
private List<Compte> comptes;
```

✳️ Constructeur :

- Initialise la liste des comptes (ex: this.comptes = new ArrayList<>();).

✳️ Méthodes :

- public void ajouterCompte(Compte compte)
- public void afficherTous()
- public Compte trouverCompte(String numero) (retourne le compte correspondant ou null s'il n'existe pas)

⚡ Variante :

Ajoute une méthode :

```
public void transferer(Compte source, Compte cible, double montant)
```

- Débite source, crédite cible.
- Affiche un message clair du transfert réussi ou de l'échec (ex: solde insuffisant).

3.

Main

Contient la méthode main() pour tester.

🧪 Scénario de test :

1. Crée une Banque vide.
2. Crée trois Compte :

- BE1001, solde 1000
- BE2002, solde 500
- BE3003, solde 2000

3. Ajoute-les à la banque.
4. Affiche tous les comptes.
5. Recherche un compte par numéro (BE2002).
6. Effectue un transfert de 200 € de BE1001 → BE2002.
7. Réaffiche tous les comptes après transfert.
8. Vérifie la comparaison equals entre deux comptes avec le même numéro.

🔍

Critères de réussite

✅ Respect de l'encapsulation

✅ Bonne utilisation du mot-clé this

✅ Redéfinition correcte de toString, equals, hashCode

✅ Logique claire dans les transferts

✅ Interaction **Banque** ↔ **Compte** bien modélisée

✅ Code lisible et structuré (Google Java Style)

⚠️

Pièges fréquents

- Comparer des chaînes avec == au lieu de .equals().
- Oublier d'utiliser this dans le constructeur.
- Modifier directement un attribut privé sans setter.
- Oublier la cohérence entre equals() et hashCode().
- Retourner null sans vérification dans trouverCompte().

💬

Variante bonus

Implémente une méthode dans Banque :

```
public double totalSolde()
```

→ renvoie la somme de tous les soldes.

Puis, appelle-la dans main pour afficher le total de la banque.