Exos recap bases Java

🧭 Objectif général

➡️ **Revoir les bases Java orientées objet** en te focalisant sur **l'interaction entre plusieurs objets** (pas juste une classe isolée).

➡️ **Intégrer progressivement** les mots-clés et principes :

this, super, héritage, polymorphisme, classes internes, abstraction, interfaces, classes anonymes, enums, et override d'objets (toString, equals, hashCode).

⚙️ MÉTHODE PÉDAGOGIQUE

🔁 Principe de progression

1. **Un concept à la fois**, mais **toujours dans une mini-application orientée objets** (ex: Compte ↔ Banque, Client ↔ Panier, Animal ↔ Zoo).
2. **Réutilisation** : à chaque nouveau concept, tu reprends un ancien projet et tu l'améliores (au lieu d'en créer un nouveau).
3. **Tests de comportements** (manuels ou JUnit) après chaque étape.
4. **Variante "entre objets" obligatoire** : toujours une interaction entre 2+ classes.
5. **Mini-fiche Obsidian** à la fin de chaque étape (concept + piège + mini-quiz).

📅 PLAN DE PROGRESSION PAR PALIERS

🧩

PALIER 1 — Fondations de la POO

> Concepts : this, encapsulation, toString, equals, hashCode

🎯 Objectif :

- Manipuler les attributs d'objet avec this.
- Comparer deux objets correctement (vs ==).

- Afficher proprement un objet.

💡 Exercice :

Compte

↔

Banque

- Classe Compte (numéro, solde)
- Classe Banque contenant plusieurs comptes
- Méthodes : ajouterCompte, afficherTous(), trouverCompte()
- Implémente toString, equals, hashCode
- ⚡ Variante : une méthode transferer(Compte source, Compte cible, double montant) → interaction directe entre objets

---

🧩

PALIER 2 — Héritage & super

> Concepts : héritage, super, surcharge de méthodes

🎯 Objectif :

- Comprendre la hiérarchie de classes.
- Appeler un constructeur parent avec super.
- Étendre des comportements (calculerInteret, afficherDetails, etc.).

💡 Exercice :

CompteCourant

&

CompteEpargne

- Compte devient une classe mère abstraite ou normale.
- CompteCourant et CompteEpargne héritent et redéfinissent calculInteret().
- Banque gère une List polymorphe.

- ⚡ Variante : ajoute une méthode afficherBilan() qui affiche un résumé global (utilise instanceof ou polymorphisme).

🧩

PALIER 3 — Polymorphisme & Interfaces

> Concepts : interface, implémentation, override, dynamique du type

🎯 Objectif :

- Comprendre comment une interface sert de contrat.
- Introduire des comportements communs mais personnalisés.

💡 Exercice :

Notifier

↔

User

- Interface Notifier avec méthode send(String message)
- Implémentations : EmailNotifier, SmsNotifier
- Classe User qui peut notifier via un Notifier
- ⚡ Variante : permet de changer de notifier à la volée (injection simulée)

🧩

PALIER 4 — Classes internes & anonymes

> Concepts : inner class, local class, anonymous class

🎯 Objectif :

- Voir comment une classe peut exister dans une autre (portée, encapsulation).
- Savoir créer un comportement ponctuel sans classe externe.

💡 Exercice : améliore

User

↔

Notifier

- Ajoute un anonymousNotifier temporaire dans le main.
- Ajoute une inner class pour un notifier spécial (ex: UrgentNotifier).
- ⚡ Variante : inner class qui accède à un attribut privé du parent.

🧩

PALIER 5 — Abstraction & enums

> Concepts : classes abstraites, enums, factorisation des comportements

🎯 Objectif :

- Créer une structure de base abstraite.
- Introduire un enum comme type logique fort.

💡 Exercice :

Employe

↔

Entreprise

- Classe abstraite Employe avec méthode abstraite calculerSalaire()
- Sous-classes : Commercial, Developpeur
- Enum Departement : IT, RH, COMPTA (avec nom + code + bonus)
- ⚡ Variante : Entreprise contient une liste d'employés et calcule la masse salariale totale par département.

🧩

PALIER 6 — Synthèse & refactor final

> Concepts : polymorphisme, abstraction, override, enums, interfaces combinés

🎯 Objectif :

Tout relier dans une mini-application complète.

💡 Exercice final :

GestionnaireDeTâches

- Tache abstraite → sous-classes TacheSimple, TacheRecurrente
- Enum Priorite { BASSE, MOYENNE, HAUTE }
- Interface Notifier pour rappeler les tâches proches
- Classe Utilisateur qui crée, supprime et reçoit des notifications
- ⚡ Bonus : mini-test JUnit sur Notifier ou Tache

📘 Format de chaque séance

Chaque notion → 1 séance structurée :

1. 🎯 Objectif clair
2. 🧠 Théorie condensée (2-3 phrases max)
3. 👨‍💻 Exercice principal
4. 🔁 Variante entre objets
5. 🧩 Mini-refactor
6. 📝 Fiche Obsidian : notions clés + quiz