**Cahier des charges — Gestion simplifiée de comptes bancaires**

1. **Présentation du projet**

Ce projet vise à développer une application Java orientée objet pour gérer une banque simplifiée. Le système doit permettre la création et la gestion de comptes bancaires, simuler les opérations courantes (dépôts, retraits, virements), et offrir une consultation claire des soldes et historiques.

2. **Objectifs**

* Modéliser les différents types de comptes bancaires avec leurs caractéristiques propres.
* Mettre en œuvre les principes de la programmation orientée objet.
* Utiliser les collections Java pour gérer efficacement les comptes et leurs opérations.
* Gérer les erreurs métier via des exceptions personnalisées.
* Maintenir un historique des opérations réalisées sur chaque compte.
* Proposer une architecture claire et modulaire avec une séparation en packages.

3. **Fonctionnalités principales**

* **Création de comptes** : compte courant avec découvert autorisé, compte épargne avec taux d’intérêt.
* **Consultation de compte** : affichage des informations et de l’historique des opérations.
* **Opérations bancaires** : crédit, débit, virement entre comptes.
* **Gestion des erreurs** : exceptions pour solde insuffisant, compte introuvable, et interdiction de virements vers le même compte.
* **Gestion des collections** : utilisation de HashMap, HashSet, ArrayList pour stocker les comptes et leurs opérations.

4. **Architecture technique**

* **Packages** : séparation claire entre modèle, service, interfaces, exceptions, et application.
* **Classes principales** : Compte abstraite, CompteCourant, CompteEpargne, Operation, Banque.
* **Interfaces** : OperationsBancaires définissant les opérations bancaires.
* **Exceptions personnalisées** : SoldeInsuffisantException, CompteIntrouvableException.
* **Utilisation des enums** : TypeCompte, TypeOperation.
* **Application console** : interface utilisateur interactive avec menu textuel.

5. **Contraintes et exigences**

* Respect de l’encapsulation et des bonnes pratiques de la programmation orientée objet.
* Validation préalable avant modification des données (ex : solde suffisant).
* Ajout des opérations dans l’historique uniquement après modification réussie.
* Gestion robuste des exceptions métier.
* Compilation et exécution depuis la racine du projet pour respecter la structure des packages.
* Documentation et diagrammes UML (classes, cas d’utilisation, séquence) à jour.

8. **Conclusion**

Ce projet permet de mettre en pratique les concepts fondamentaux de la programmation orientée objet en Java, tout en développant une application bancaire fonctionnelle et modulaire. La gestion précise des opérations, l’historique détaillé et la robustesse apportée par les exceptions personnalisées garantissent une solution fiable et évolutive.