Guide Utilisateur – Système de Gestion Bancaire

Laraichi Bedoui Kenza

1-Création d'un client

Objectif : Ajouter un client dans le système.
Code : `Client client = new Client("Laraichi", "Kenza", "AB123", "Fes");`
Résultat attendu : Client créé et prêt à posséder des comptes.

```
C:\Users\kenza\.jdks\ms-17.0.17\bin\java.exe "-javaagent:C:\Program Files\JetBra
Système de Gestion Bancaire - Démonstration

Client créé : Laraichi Kenza
```

2-Création de comptes bancaires

Types de comptes :

-Compte Courant (découvert autorisé)

-Compte Épargne (taux d'intérêt, pas de découvert)

-Compte Entreprise (plafond de retrait élevé)

Exemple :

```
//Création des comptes
CompteCourant compteCourant = new CompteCourant( numero: "C001",  soldeInitial: 1000);
CompteEpargne compteEpargne = new CompteEpargne( numero: "E001",  soldeInitial: 2000);
CompteEntreprise compteEntreprise = new CompteEntreprise( numero: "ENT001",  soldeInitial: 50000);
```

Explication :

Chaque compte a un **numéro unique** et un **solde initial**.
Les comptes sont associés au client via `ajouterCompte()`.

3-Effectuer des opérations bancaires

Les opérations bancaires supportées par le système sont : - Dépôt d'argent - Retrait d'argent avec validation du solde et des limites - Virement entre comptes - Application des intérêts pour les comptes épargne

Chaque opération génère automatiquement une transaction enregistrée dans l'historique du compte.

1-Dépôt

```
//  Dépôts
compteCourant.deposer( montant: 500);
compteEpargne.deposer( montant: 1000);
compteEntreprise.deposer( montant: 20000);
```

2-Retrait

```
//Retraits
compteCourant.retirer( montant: 300);
compteEpargne.retirer( montant: 500);
compteEntreprise.retirer( montant: 10000);
```

3-Virement entre comptes

```
//  Virement entre comptes du même client
compteCourant.retirer( montant: 200);
compteEpargne.deposer( montant: 200);
```

4-Consulter le solde et l'historique

a) Afficher le solde

Code:

```
// Affichage des soldes


System.out.println("Soldes après opérations :");
System.out.println("Compte Courant (C001) : " + compteCourant.getSolde());
System.out.println("Compte Epargne (E001) : " + compteEpargne.getSolde());
System.out.println("Compte Entreprise (ENT001) : " + compteEntreprise.getSolde() + "\n");
```

Résultat:

```
Soldes après opérations :
Compte Courant (C001) : 1000.0
Compte Epargne (E001) : 2781.0
Compte Entreprise (ENT001) : 60000.0
```

b) Afficher l'historique des transactions

Chaque transaction est représentée par la classe Transaction, contenant :

- la date et l'heure de l'opération (LocalDateTime)

- le type de transaction (dépôt, retrait, virement)

- le montant

Le système permet :

- l'affichage de l'historique des transactions

- la génération d'un relevé mensuel

- le filtrage des transactions par type ou par période

Un service supplémentaire permet de générer des relevés au format HTML.

Code:

```
//  Affichage de l'historique des transactions pour le compte courant
System.out.println("Transactions Compte Courant :");
for (Transaction t : compteCourant.getTransactions()) {
    System.out.println(t);
}
```

Résultat:

```
Transactions Compte Courant :
2026-01-11T02:05:31.568069900 | DEPOT | 500.0
2026-01-11T02:05:31.568069900 | RETRAIT | 300.0
2026-01-11T02:05:31.568069900 | RETRAIT | 200.0
```

5-Notification du solde faible

Code :

```
//Notif du solde faible
EmailNotification notif = new EmailNotification(compteCourant.getSolde());
notif.envoyer("Attention ! Votre solde est faible.");
```

Résultat:

```
Email non envoyé, solde suffisant : 1000.0 MAD
```

6-Cas d'usage concrets

Cas d'usage	Etapes	Resultat attendu
Déposer 500 MAD sur le compte courant	compteCourant.deposer(500)	Le solde du compte augmente de 500
Retirer 300 MAD du compte épargne	compteEpargne.retirer(300)	Le solde du compte augmente de 500
Virement interne	Retirer 200 du courant et déposer sur épargne	Les deux comptes sont mis à jour correctement
Historique des transactions	getTransactions()	Liste complète des opérations avec date et type

7-Cas exceptionnels

Les exceptions métier suivantes ont été implémentées : - SoldeInsuffisantException - MontantInvalideException - LimiteDepasseeException

Elles permettent de sécuriser le système et de garantir le respect des règles bancaires.

Montant Invalide	`public void deposer(double montant) throws MontantInvalideException {`
	`  if (montant <= 0)`
	`  throw new MontantInvalideException("Montant invalide");`
		On ne peut pas deposer un motant nul ou négatif

Limite Depasser	`public void retirer(double montant) throws Exception {`
	`  if (montant > plafondJournalier)`
	`  throw new LimiteDepasseeException("Plafond journalier dépassé");`
		On ne peut pas dépasser un plafond fixe par jour

Solde Insuffisant	`if (montant > solde)`
	`  throw new SoldeInsuffisantException("Solde insuffisant");`
	`}`
		Si le montant demandé est plus grand que le solde du compte