Doc fonctionnelle et technique – TD Banque Cyril KRZYZOWSKI

Utilisation de l'application :

1/ Démarrer l'application via le fichier « Main »

2/ Une fois démarré, le programme vous listera tous les comptes de la banque et vous demandera d'en sélectionner un.

3/ Pour en sélectionner un, entrez l'ID du compte et appuyez sur entrée.

4/ Un nouveau menu s'affiche alors vous proposant 7 actions :

- Créer un nouveau compte bancaire, pour cela il vous suffira d'entrer votre nom et la création se fera automatiquement après ça. Un nouveau compte client sera également généré à ce moment.
- Effectuer un retrait, vous n'aurez qu'à entrer le montant que vous souhaitez retirer du compte que vous avez préalablement séléctionné et l'argent sera débité automatiquement, et le nouveau solde s'affichera.
- Effectuer un dépôt, idem, entrez le montant et le compte sera crédité de l'argent immédiatement, et le nouveau solde s'affichera.
- Convertir des euros en dollars, entrez une valeur et le programme fera la conversion automatique d'euros vers les dollars.
- Convertir des dollars en euros, idem qu'avant, entrez une valeur et la conversion inverse s'effectuera alors.
- Consulter tous les comptes de la banque vous affichera, comme au début, tous les comptes disponibles au sein de la banque.
- Quitter l'application, vous fera quitter l'application.

Ajouter la base de données :

- Il vous suffira de récupérer le fichier .sql et d'exécuter le script pur une base de données MySQL (j'ai fait la mienne sous un Wamp peut-être un peu vieux, il peut y avoir des incompatibilités)

Détails de l'application et doc technique :

Premièrement, par rapport au diagramme UML, j'ai rajouté une méthode dans la classe banque, « allComptes » qui me permet de retourner tous les comptes de la base de données pour pouvoir l'afficher dans le programme.
J'ai également rajouté dans la classe Compte l'id client qui est associé au compte pour respecter la liaison avec la base de données. Pour ce qui est du compte avec découvert ou sans ,j'ai réalisé deux classes filles à la classe Compte pour gérer ça.

En ce qui concerne les DAO, je me suis focalisé sur la DAO des comptes, car les fonctionnalités étaient plus riches que des clients et le temps étant assez court, j'ai bien fait de m'y concentrer pour que tout fonctionne. Les DAO ont toutes les deux les fonctions pour créer soit un compte soit un client. En revanche, la DAO compte a l'update ainsi que le delete ce que n'a pas la DAO client.

J'ai également rajouté des fonctionnalités supplémentaire à la DAOCompte que celles de base, comme par exemple pour récupérer tous les comptes, un compte spécifique ou le dernier compte enregistré.

Pour ce qui est de ce qui manque, il aurait fallu faire la connexion à la BDD en singleton pour éviter d'eventuels connexions multiples à celle-ci, j'ai préférer me focaliser sur les requêtes plutôt que de créer l'instanciation de celle-ci.
Au niveau du programme il manque donc une majeure partie de la DAOClient, elle ne nous permet que de créer un client et de récupérer le dernier client créé (pour pouvoir en créer un nouveau lorsqu'on créé un compte).
Idem pour les tests, il en manque beaucoup et j'ai passé bien trop de temps à régler les soucis des simples tests qu'il y a pour qu'ils fonctionnent que j'ai pas réussi à en réaliser d'autres.
Pour ce qui est de l'interface en front-end, elle mériterait des améliorations en terme de cohérence car on demande un ID de compte avant de demander ce que l'utilisateur veut faire, sachant qu'il n'est pas nécessaire d'avoir un nom de compte pour faire certains actions(conversion, affichage des comptes…).

J'aurais du également utiliser Mockito pour les tests qu'il fallait faire (et rien que pour le peu que j'ai pu réaliser), mais je n'arrivais pas à le mettre en place correctement alors j'ai préféré y aller « à la dure » et avoir des tests qui fonctionnent.