



Valentin LABAT
ING3 FINTECH

Rapport du projet Java Infrastructure bancaire

Effectué de septembre à novembre 2021

A l'attention de Mme BAAZAOUI Hajer

I - Description et contexte du projet :

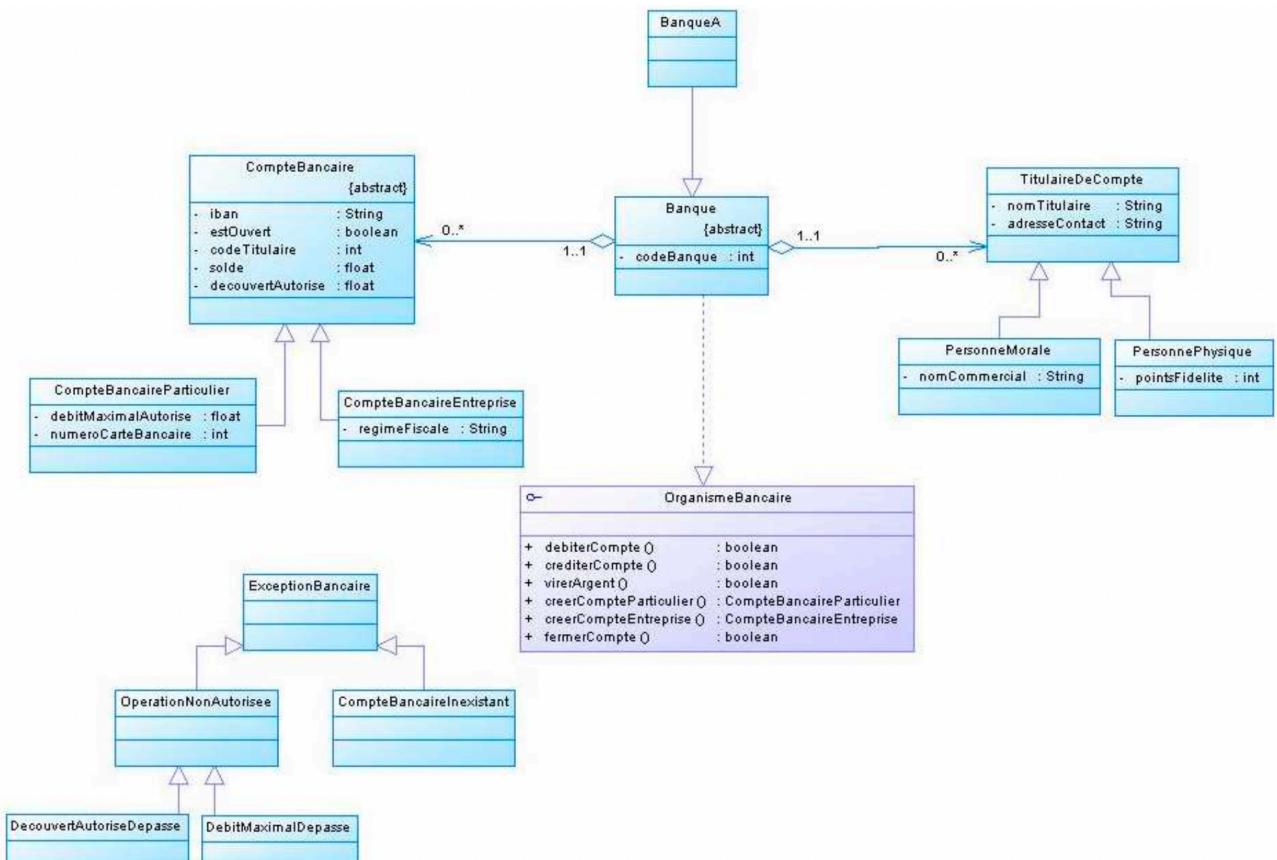
Dans le cadre du cours de Programmation Pour la Finance, nous devions effectuer un projet en Java. Ce projet, que j'ai effectué seul et qui porte sur l'infrastructure bancaire basique, a pour but de montrer les connaissances que nous avons acquises tout au long de ce cours, et de les implémenter dans un domaine en rapport avec notre domaine d'études. Ainsi, à partir d'un schéma de base, nous devions prendre des décisions et effectuer à minima les fonctions demandées, et toute fonction supplémentaire était bénéfique. Vous trouverez donc, dans le rapport ci dessous, une description de mon rendu, qui comprendra également mes prises de décisions expliquées, et les pistes d'amélioration.

L'exécution principale du code se fait dans le fichier main, et sera explicité dans le chapitre IV.

Ce projet a donc été effectué sur éclipse, avec la version datant de septembre 2021.

II - Schéma de base et changements majeurs effectués :

Pour commencer notre projet, le schéma suivant nous était fournis.



On retrouve ici les éléments suivants :

- une classe « principale » **BanqueA**,

- une classe abstraite **Banque**,
- une interface de Banque, **OrganismeBancaire**,
- une classe abstraite **CompteBancaire**, et ses classes filles **CompteBancaireParticulier** et **CompteBancaireEntreprise**,
- une classe **TitulaireDeCompte** et ses classes filles **PersonneMorale** et **PersonnePhysique**,
- Une classe d'exception **ExceptionBancaire** et ses classes filles **OperationNonAutorisee** et **CompteBancaireInexistant**,
- Les deux classes filles de **OperationNonAutorisee**, **DcouvertAutoriseDepasse** et **DebitMaximalDepasse**

J'ai donc globalement suivi le même schéma, avec globalement les mêmes fonctions. Cependant, je n'ai pas implémenté la classe **BanqueA**, car elle ne m'était pas utile, et j'ai choisi de supprimer le caractère abstrait de la classe **Banque**. Ainsi, toutes les fonctions associées, ainsi que mon main, se trouvent directement dans la classe Banque, ce qui facilite grandement le traitement des données.

III - Explication du projet et des classes :

Pour toutes les classes ci-dessous, bien que ce ne soit pas explicité, les constructeurs, guetters et setters ont bien évidemment été définis. Il en va de même pour les fonctions **toString()**, qui ont elles aussi été implémentés au besoin. De plus, les noms de fonctions et d'arguments ont été choisis pour être transparents. Ainsi, je n'expliciterait pas longuement certains noms qui ne semblent pas nécessiter de plus d'explications.

a) Classe Banque :

La classe **Banque** est ma classe « principale » du programme. C'est notamment dans celle-ci qu'il y a la fonction main. A noter, certaines fonctions concernent des comptes bancaires, je les ai simplement mises dans cette classe là pour un soucis de simplicité et pour laisser le caractère abstrait de la classe **CompteBancaire**.

Dans cette classe se trouvent 3 arguments :

- un *String* nomBanque
- un *int* codeBanque
- un *ArrayList<CompteBancaire>* listeComptes

Ainsi, l'objectif est pouvoir créer plusieurs comptes dans chaque banque, chacun étant désigné par un nom et un code.

La fonction *ajouterCompte()* permet d'ajouter un compte bancaire pris en paramètre à la liste des comptes existants.

La fonction *ouvrirCompteBancaire()* permet d'ouvrir un compte bancaire de type pris en paramètre dans la banque. Ici, les types implémentés sont « entreprise » ou « particulier ». Dans cette fonction, on crée de façon aléatoire un Iban selon le format official trouvé sur internet ('FR' + 25 chiffres). De plus, on crée les arguments classiques de chaque compte en fonction de si le compte est particulier ou d'entreprise.

La fonction *fermerCompteBancaire()* permet de fermer le compte donc l'iban est renseigné en paramètre de la fonction. [Attention, ne pas confondre un compte fermé et un compte supprimé]

La fonction *supprimerCompteBancaire()* permet de supprimer le compte donc l'iban est renseigné en paramètre de la fonction de la liste des comptes de la banque.

La fonction *compteExistant()* permet de vérifier si le compte d'iban entré en paramètre existe ou non. Elle renvoie une exception définie dans la classe CompteBancaireInexistant (voir ci-après).

La fonction *rechercherCompteBancaire()* utilise la fonction précédente pour déterminer si un compte existe dans la liste de comptes de la banque.

La fonction *banqueNomAleatoire()* permet de créer une banque en créant un nom aléatoire puis en utilisant les constructeurs de la classe.

La fonction *afficheEnTete()* permet d'afficher l'en-tête lors de l'exécution du code.

La fonction *menu1()* permet d'afficher le premier menu.

La fonction *main()* sera décrite dans le chapitre suivant.

b) Classe **CompteBancaire** :

La classe **CompteBancaire** est la classe définissant tous les comptes bancaires.

Dans cette classe se trouvent 9 arguments :

- un *String iban*
- un *boolean estOuvert*
- un *int codeTitulaire*
- un *double soldé*
- un *double découvertAutorisé*
- une *Date dateOuverture*
- un *ArrayList<String> tracesCompte*
- un *double tauxInteret*
- un *double debitMaximalAutorisé*

Ainsi, l'objectif de cette classe est pouvoir créer des comptes bancaires.

La fonction *ajouterTraceCompte()* permet d'ajouter la ligne entrée en paramètre en trace du compte.

La fonction *checkDecouvertAutorise()* permet de vérifier si l'opération ne dépasse pas le découvert autorisé. Elle renvoie une exception définie dans la classe *DecouvertAutoriseDepasse* (voir ci-après).

La fonction *checkDebitMaximal()* permet de vérifier si l'opération ne dépasse pas le débit maximal autorisé. Elle renvoie une exception définie dans la classe *DebitMaximalAutorise* (voir ci-après).

La fonction *debiterCompte()* permet de débiter le compte de la valeur, si le débit maximal et le découvert ne sont pas dépassés (en utilisant les deux fonctions précédentes).

La fonction *virerArgent()* permet de virer de l'argent d'un compte vers un autre, en utilisant les deux mêmes fonctions que dans la fonction précédente.

c) Classe CompteBancaireEntreprise :

La classe **CompteBancaireEntreprise** est la classe définissant tous les comptes bancaires d'entreprise.

Dans cette classe se trouvent un seul argument particulier :

- un *String regimeFiscal*.

A cet attribut s'ajoutent les 9 de la classe mère **CompteBancaire**.

Ainsi, l'objectif de cette classe est pouvoir créer des comptes bancaires à destination des entreprises.

Rien de particulier à noter sur cette classe. Seuls les getters, setters, constructeurs et *toString()* sont présents dans cette classe.

d) Classe CompteBancaireInexistant :

La classe **CompteBancaireInexistant** est la classe définissant l'exception concernant la recherche de comptes bancaires inexistantes notamment.

e) Classe CompteBancaireParticulier :

La classe **CompteBancaireParticulier** est la classe définissant tous les comptes bancaires des particuliers.

Dans cette classe se trouvent un seul argument particulier :

- un *double numeroCarteBancaire*.

A cet attribut s'ajoutent les 9 de la classe mère **CompteBancaire**.

Ainsi, l'objectif de cette classe est pouvoir créer des comptes bancaires à destination des particuliers.

Comme pour la classe **CompteBancaireEntreprise** rien de particulier à noter sur cette classe. Seuls les getters, setters, constructeurs et `toString()` sont présents dans cette classe.

f) **Classe DebitMaximalAutorise** :

La classe **DebitMaximalAutorise** est la classe définissant l'exception concernant le débit maximal autorisé.

g) **Classe DecouvertAutoriseDepasse** :

La classe **DecouvertAutoriseDepasse** est la classe définissant l'exception concernant le dépassement du découvert autorisé.

h) **Classe ExceptionBancaire** :

La classe **ExceptionBancaire** est la classe mère des exceptions définissant toutes les exceptions bancaires citées ci-dessus.

i) **Classe OperationNonAutorisee** :

La classe **OperationNonAutorisee** est également une classe d'exceptions. C'est une classe fille de **ExceptionBancaire** et la classe mère de **DecouvertAutoriseDepasse** et **DebitMaximalAutorise**.

j) **Interface OrganismeBancaire** :

L'interface **OrganismeBancaire** est l'interface principale du projet. A noter que, dans mon cas, l'interface n'a pas beaucoup d'intérêt, mais les interfaces sont très utiles, notamment dans les projets de grande envergure, car elles permettent de définir une structure globale.

k) Classe **TitulaireDeCompte** :

La classe **TitulaireDeCompte** est la classe définissant l'ensemble des titulaires de comptes. Elle possède deux classes filles définies ci-après.

Dans cette classe se trouvent 2 arguments :

- un *String nomTitulaire*
- un *String adresseContact*

Ainsi, l'objectif de cette classe est pouvoir créer des titulaires de comptes.

Rien de particulier à noter sur cette classe. Seuls les getters, setters et constructeurs sont présents dans cette classe.

l) Classe **PersonneMorale** :

La classe **PersonneMorale** est une classe fille de la classe **TitulaireDeCompte** définissant l'ensemble des personnes morales titulaires de comptes.

Dans cette classe se trouvent 1 arguments :

- un *String nomCommercial*

A cet argument sont ajoutés les deux arguments de la classe mère.

Ainsi, l'objectif de cette classe est pouvoir créer des titulaires de comptes d'entreprise.

m) Classe **PersonnePhysique** :

La classe **PersonnePhysique** est l'autre classe fille de la classe **TitulaireDeCompte** définissant l'ensemble des personnes physiques titulaires de comptes.

Dans cette classe se trouvent 1 arguments :

- un *int pointsFidelite*

A cet argument sont ajoutés les deux arguments de la classe mère.

Ainsi, l'objectif de cette classe est pouvoir créer des titulaires de comptes particuliers.

IV - Description du programme principal :

Lors de l'exécution de mon programme, plusieurs choses se déroulent. Elles seront expliquées dans ce chapitre, et démontrées par des captures d'écran. A noter que si vous exécutez le fichier en état, les résultats seront les mêmes. J'ai essayé d'être le plus exhaustif possible, en essayant d'afficher la majorité des cas dans ma fonction main, afin de montrer ce dont est capable mon programme. Si vous voulez changer les arguments des fonctions par exemple, vous pourrez tester les différentes fonctionnalités implémentés. Notons donc que les étapes après l'étape 2 se font automatiquement. De plus, un suivant des comptes disponibles dans la banque sera affiché à chaque étape.

Etape 0 : En-tête et menu

Lors de l'exécution, vous tombez sur un menu, avec une entrée clavier afin de choisir si vous voulez exécuter le programme ou non. De plus, vous trouvez également une en-tête qui permet de personnaliser légèrement l'exécution. Ainsi, le choix 1 nous permet d'exécuter le programme, alors que tout autre choix vous fait quitter celui-ci.

```
---- Bienvenue dans votre service de gestion bancaire ValLabank---
---- Vous allez pouvoir exécuter le programme implémenté ---
-- Que voulez-vous faire ? --
Tapez 1 : Dérouler le programme
Autre touche : Quitter
```

Lorsque l'on tape 1, une banque avec un nom aléatoire est créée :

```
'La banque suivante a bien été créée :
nomBanque : pccuetadbz ;
codeBanque : 252 ;
listeComptes : [] ;
```

Etape 1 : Ouverture d'un compte entreprise

```
Etape 1 :  
Ouverture d'un compte entreprise dans la banque  
La banque est composée de ces comptes :  
[Iban : FR3477322454372512888287305  
Ouvert  
CodeTitulaire : 1  
Solde : 0.0  
debitMaximalAutorise : 2000.0  
DecouvertAutorise : 200.0  
RegimeFiscal : impot sur le revenu  
dateOuverture : Fri Nov 19 15:13:09 CET 2021  
tracesCompte : [Fri Nov 19 15:13:09 CET 2021 , ouverture , +0€ , 0€]  
tauxInteret : 0.5  
]
```

On voit que la création a bien été effectuée.

Etape 2 : Utilisation de la fonction recherche de compte

Pour cette étape, on récupère l'iban du compte qui vient d'être créé, et on utilise la fonction recherche.

```
Recherche du compte entreprise dans la banque  
Le compte existe :  
Iban : FR3477322454372512888287305  
Ouvert  
CodeTitulaire : 1  
Solde : 0.0  
debitMaximalAutorise : 2000.0  
DecouvertAutorise : 200.0  
RegimeFiscal : impot sur le revenu  
dateOuverture : Fri Nov 19 15:13:09 CET 2021  
tracesCompte : [Fri Nov 19 15:13:09 CET 2021 , ouverture , +0€ , 0€]  
tauxInteret : 0.5
```

Logiquement, le compte a été trouvé. La fonction recherche a bien fonctionné.

Etape 3 : Ouverture d'un compte particulier

```
Ouverture d'un compte particulier dans la banque
La banque est composée de ces comptes :
[IBAN : FR3477322454372512888287305
Ouvert
CodeTitulaire : 1
Solde : 0.0
debitMaximalAutorise : 2000.0
DecouvertAutorise : 200.0
RegimeFiscal : impôt sur le revenu
dateOuverture : Fri Nov 19 15:13:09 CET 2021
tracesCompte : [Fri Nov 19 15:13:09 CET 2021 , ouverture , +0€ , 0€]
tauxInteret : 0.5
, iban : FR7064220224474142115415531
ouvert
codeTitulaire : 1
solde : 0.0
decouvertAutorise : 200.0
debitMaximalAutorise : 2000.0
numeroCarteBancaire : 5835304163712740
dateOuverture : Fri Nov 19 15:13:10 CET 2021
tracesCompte : [Fri Nov 19 15:13:10 CET 2021 , ouverture , +0€ , 0€]
tauxInteret : 0.5
]
```

Le compte particulier a bien été créé, et ajouté à la liste des comptes de la banque.

Etape 4 : Fermeture d'un compte entreprise

```
Fermeture du compte entreprise dans la banque
La banque est composée de ces comptes :
[IBAN : FR3477322454372512888287305
Fermé
CodeTitulaire : 1
Solde : 0.0
debitMaximalAutorise : 2000.0
DecouvertAutorise : 200.0
RegimeFiscal : impôt sur le revenu
dateOuverture : Fri Nov 19 15:13:09 CET 2021
tracesCompte : [Fri Nov 19 15:13:09 CET 2021 , ouverture , +0€ , 0€]
tauxInteret : 0.5
, iban : FR7064220224474142115415531
ouvert
codeTitulaire : 1
solde : 0.0
decouvertAutorise : 200.0
debitMaximalAutorise : 2000.0
numeroCarteBancaire : 5835304163712740
dateOuverture : Fri Nov 19 15:13:10 CET 2021
tracesCompte : [Fri Nov 19 15:13:10 CET 2021 , ouverture , +0€ , 0€]
tauxInteret : 0.5
]
```

On remarque dans l'attribut du premier compte que celui-ci est bien fermé.

Etape 5 : Suppression d'un compte entreprise

```
Suppression du compte entreprise dans la banque
La banque est composée de ces comptes :
[iban : FR7064220224474142115415531
ouvert
codeTitulaire : 1
solde : 0.0
decouvertAutorise : 200.0
debitMaximalAutorise : 2000.0
numeroCarteBancaire : 5835304163712740
dateOuverture : Fri Nov 19 15:13:10 CET 2021
tracesCompte : [Fri Nov 19 15:13:10 CET 2021 , ouverture , +0€ , 0€]
tauxInteret : 0.5
]
```

On voit que le compte a bien été supprimé, il n'apparaît plus dans la liste des comptes.

Etape 6 : Recherche du compte entreprise

Pour confirmer que le compte soit supprimé, on cherche si le compte possédant l'iban du compte qui vient d'être supprimé existe.

```
Recherche du compte entreprise dans la banque
Compte inexistant.
```

Logiquement, le compte est introuvable.

Etape 7 : On crédite le compte particulier de 1000€

```
Vous avez bien été crédité  
[iban : FR7064220224474142115415531  
ouvert  
codeTitulaire : 1  
solde : 1000.0  
decouvertAutorise : 200.0  
debitMaximalAutorise : 2000.0  
numeroCarteBancaire : 5835304163712740  
dateOuverture : Fri Nov 19 15:13:10 CET 2021  
tracesCompte : [Fri Nov 19 15:13:10 CET 2021 , ouverture , +0€ , 0€, Fri Nov 19 15:13:10 CET 2021 , crédit , +1000.0€ , 1000.0€]  
tauxInteret : 0.5  
]
```

On remarque via le solde et la trace que l'opération a bien été effectuée.

Etape 8 : On débite le compte particulier de 1300€

```
Opération refusée. Votre découvert autorisé est dépassé.  
[iban : FR7064220224474142115415531  
ouvert  
codeTitulaire : 1  
solde : 1000.0  
decouvertAutorise : 200.0  
debitMaximalAutorise : 2000.0  
numeroCarteBancaire : 5835304163712740  
dateOuverture : Fri Nov 19 15:13:10 CET 2021  
tracesCompte : [Fri Nov 19 15:13:10 CET 2021 , ouverture , +0€ , 0€, Fri Nov 19 15:13:10 CET 2021 , crédit , +1000.0€ , 1000.0€]  
tauxInteret : 0.5  
]
```

Via le message d'erreur (levé par l'exception) et le solde, on voit que l'opération a été refusée. En effet, si on effectue cette opération, le solde du compte sera de $1000 - 1300 = 300$ €, et le découvert autorisé est de 200€. C'est donc logiquement que l'opération est refusée.

Etape 9 : On crédite le compte particulier de 4000€ puis on le débité de 3000€

```
Vous avez bien été crédité  
Opération refusée. Vous ne pouvez pas débiter autant.  
[iban : FR7064220224474142115415531  
ouvert  
codeTitulaire : 1  
solde : 5000.0  
decouvertAutorise : 200.0  
debitMaximalAutorise : 2000.0  
numeroCarteBancaire : 5835304163712740  
dateOuverture : Fri Nov 19 15:13:10 CET 2021  
tracesCompte : [Fri Nov 19 15:13:10 CET 2021 , ouverture , +0€ , 0€, Fri Nov 19 15:13:10 CET 2021 , crédit , +1000.0€ , 1000.0€,  
tauxInteret : 0.5  
]
```

Le crédit a bien été effectué, mais pas le débit. En effet, $3000 > 2000$, qui est le débit maximal autorisé. C'est donc une deuxième exception qui est levée ici.

Etape 10 : On crédite le compte particulier de 800€

Vous avez bien été débité

```
[iban : FR7064220224474142115415531
ouvert
codeTitulaire : 1
solde : 4200.0
decouvertAutorise : 200.0
debitMaximalAutorise : 2000.0
numeroCarteBancaire : 5835304163712740
dateOuverture : Fri Nov 19 15:13:10 CET 2021
tracesCompte : [Fri Nov 19 15:13:10 CET 2021 , ouverture , +0€ , 0€, Fri Nov 19 15:13:10 CET 2021 , crédit , +1000.0€ , 1000.0€, Fri Nov 19 15:13:
tauxInteret : 0.5
]
```

Fri Nov 19 15:13:10 CET 2021 , débit , -800.0€ , 4200.0€]

On voit dans les traces et au niveau du solde que le débit a bien été effectué. En effet, toutes les conditions sont remplies pour qu'il ait lieu.

Etape 11 : On crée un nouveau compte entreprise et on le crédite de 1000€

Vous avez bien été crédité

```
[iban : FR7064220224474142115415531
ouvert
codeTitulaire : 1
solde : 4200.0
decouvertAutorise : 200.0
debitMaximalAutorise : 2000.0
numeroCarteBancaire : 5835304163712740
dateOuverture : Fri Nov 19 15:13:10 CET 2021
tracesCompte : [Fri Nov 19 15:13:10 CET 2021 , ouverture , +0€ , 0€, Fri Nov 19 15:13:10 CET 2021 , crédit , +1000.0€ , 1000.0€, Fri Nov 19 15:13:
tauxInteret : 0.5
, Iban : FR0655234582006346647718337
Ouvert
CodeTitulaire : 1
Solde : 1000.0
debitMaximalAutorise : 2000.0
DecouvertAutorise : 200.0
RegimeFiscal : impôt sur le revenu
dateOuverture : Fri Nov 19 15:13:10 CET 2021
tracesCompte : [Fri Nov 19 15:13:10 CET 2021 , ouverture , +0€ , 0€, Fri Nov 19 15:13:10 CET 2021 , crédit , +1000.0€ , 1000.0€]
tauxInteret : 0.5
]
```

Etape 12 : Virement de 100€ du compte entreprise vers le compte personnel

Le virement a bien été effectué

```
[iban : FR7064220224474142115415531
ouvert
codeTitulaire : 1
solde : 4300.0
decouvertAutorise : 200.0
debitMaximalAutorise : 2000.0
numeroCarteBancaire : 5835304163712740
dateOuverture : Fri Nov 19 15:13:10 CET 2021
tracesCompte : [Fri Nov 19 15:13:10 CET 2021 , ouverture , +0€ , 0€, Fri Nov 19 15:13:10 CET 2021 , crédit , +1000.0€ , 1000.0€, Fri Nov 19 15:13:
tauxInteret : 0.5
, Iban : FR0655234582006346647718337
Ouvert
CodeTitulaire : 1
Solde : 900.0
debitMaximalAutorise : 2000.0
DecouvertAutorise : 200.0
RegimeFiscal : impôt sur le revenu
dateOuverture : Fri N Capture d'écran 2021
tracesCompte : [Fri Nov 19 15:13:10 CET 2021 , ouverture , +0€ , 0€, Fri Nov 19 15:13:10 CET 2021 , crédit , +1000.0€ , 1000.0€, Fri Nov 19 15:13:
tauxInteret : 0.5
]
```

On remarque que le virement a bien été effectué.

Etape 13 : Crédation d'une personne physique

```
-- Etape 13 :  
Création d'une personne physique :  
nomTitulaire : Labat Valentin  
adresseContact : 12 rue de CYTech  
pointsFidelite : 1
```

On peut voir que la personne a bien été créée.

Etape 14 : Crédation d'une personne morale

```
-- Etape 14 :  
Création d'une personne morale :  
nomTitulaire : Labat & Co  
adresseContact : 12 rue de CYTech  
nomCommercial : SARL
```

V - Améliorations possibles :

Tout au long de mon projet, j'ai pu noter des pistes d'amélioration et d'implémentation possibles. Ainsi, j'ai pu implémenter quelques unes des ces idées au fur et à mesure en fonction du temps dont je disposais. Je vais donc lister ci-dessous celles auxquelles j'ai pensé mais que je n'ai pas eu le temps d'implémenter.

a) Classe Banque :

- Changer la formation de l'iban pour qu'il y ait une clé identique (au début ou à la fin) pour les comptes d'une même banque.
- Vérifier le caractère unique de l'iban. Même si cela paraît fondamental, je ne l'ai pas codé en priorité car notre application est petite, et il est très peu probable qu'il y ait un doublon sur un nombre de 25 chiffres implantés aléatoirement. Ce n'était donc pas un priorité.
- Donner la possibilité lors de la création du compte d'entrer certains paramètres comme le découvert maximal autorisé, etc afin de rendre l'application encore plus profonde.

- Donner la possibilité de créer une banque à la main.
- Vérifier que le code banque est unique dans la liste de banques.

b) Améliorations globales :

- Relier un utilisateur à un compte en banque. Utile notamment si implémentation d'une base de données liée au programme et essentiel pour la fiabilité du programme et l'unicité des entités.
- Implémentation plus poussée du menu, afin que les utilisateurs puissent faire les actions qu'ils souhaitent.
- Amélioration du modèle en implémentant des classes filles aux comptes bancaires comme de Comptes Courantes, Livrets A, etc.
- Possibilité d'améliorer la recherche de compte en ne faisant pas que par l'ban, mais par le nom du propriétaire, etc.

VI - Conclusion :

Ce projet m'a donc permis de rappeler les bases de Java et de les implémenter sur un projet de moyen-terme. Il m'a permis de mettre en avant les points que je maîtrise, et de me rendre compte du grand nombre de possibilités de codage lié à un projet qui peut paraître de prime abord assez simple. En effet, tout au long de ma programmation, j'ai eu plusieurs idées qui me sont venues et qu'il aurait été bien d'implémenter. Finalement, j'ai donc plutôt bien progressé en Java, et ces bases là seront sans doutes très utiles pour mes emplois futurs, Java étant un langage répandu dans le milieu.

$T_{ab} = 1, 3, 8, 10, 12, 20$

$N = 3$