IFT3913 - Qualité de Logiciel et Métriques
TP 4

Yalin Mo (20199655)
Vennila Sooben (20235256)

8 décembre 2023

Objectif

Dans le cadre de ce TP, l'objectif était de tester le logiciel "Currency Converter". Le code est accessible via le dépôt Git à `https://github.com/mfamelis/currency-converter-in-java`. Nous avons focalisé nos tests sur deux méthodes du "Currency Converter" :

— `currencyConverter.MainWindow.convert(String, String, ArrayList<Currency>, Double)`
— `currencyConverter.Currency.convert(Double, Double)`

1. Tests Boîte Noire

Choix des cas de tests Nous avons choisi les cas de tests en nous basant sur les classes d'équivalence et l'analyse des valeurs frontières, couvrant ainsi un large éventail de scénarios possibles. Nous avons également pris en compte les spécifications du "Currency Converter" lors de la création de nos tests.

Méthode `currencyConverter.MainWindow.convert`

Nous avons suivi une approche de partitionnement du domaine en classes d'équivalence et d'analyse des valeurs frontières pour les devises et les montants. En effet, les valeurs normales, extremes et invalides ont été utilisées.

— **Analyse des valeurs frontières pour les montants :**
  1. Montant normale : Valide
     — 10
  2. Montants extrêmes valides : Valide
     — 0
     — 1000000
  3. Montants proches des limites : Exception
     — -0.01
     — 1000000.01
  4. Montant négatif : Exception
     — -1

— **Analyse des valeurs frontières pour les monnaies :**
  1. Monnaie normale : Valide
     — USD, CAD, GBP, EUR, CHF, AUD.
  2. Monnaie non-inclue dans l'ArrayList : Exception
     — USD, CAD, GBP, EUR, CHF, AUD. dans l'ArrayList mais test avec XYZ

— **Hypothèses pour les entrées non-valides :**
  — Pour les devises invalides, nous nous attendons à un message d'erreur ou à l'ignorance de la conversion.

— Pour les montants invalides, nous anticipons une exception ou un message d'erreur.

— **Problématique Identifiée**
Au cours des tests, il a été observé que l'application présente un comportement inattendu dans certaines situations. Cependant, en tant que testeur boîte noire, nous ne sommes pas en mesure de fournir des détails sur l'implémentation interne, mais plutôt de signaler les comportements observés.

— **Recommandations**

— Il faut une vérification approfondie de la fonctionnalité de conversion de devises.
— Il faut poursuivre avec des tests boîtes blanche

— **Conclusion**
La spécification aurait dû fournir plus de détails notamment sur le ArrayList<Currency> et sur la classe Currency. En effet, un travail supplémentaire était nécessaire. Ceci ne respecte pas les tests boites noirs mais afin de mener à bien l'écriture de test, nous avons utilises les getters disponibles. Ceci était dans le but de rester dans l'esprit de ne pas regarder le code de MainWindow.convert. La seule impasse que nous nous sommes accordés était de vérifier quelle méthode utiliser pour initialiser l'ArrayList<Currency>.
De plus, les tests ne passent pas. En effet, non seulement CAD et AUD était indisponible mais en plus nous obtenons des 0 constamment.

Méthode `currencyConverter.Currency.convert`

Nous avons suivi une approche de partitionnement du domaine en classes d'équivalence et d'analyse des valeurs frontières pour les devises et les montants.

— **Classes d'équivalence pour les devises :**

1. Nous utilisons plusieurs devises pour représenter différentes classes d'équivalence. Chaque devise représente une classe d'équivalence distincte, comme le US Dollar (USD), Euro (EUR), British Pound(GBP), etc. Nous nous assurons que nos tests couvrent ces différentes devises pour vérifier si la méthode currency.convert fonctionne correctement lorsqu'elle traite des conversions entre différentes devises.

— **Classes d'équivalence pour les montants :**

1. Dans nos tests, nous prenons en compte les classes d'équivalence suivantes :
   — Montant positif —— 500000.00 : nous testons les montants positifs pour nous assurer que la méthode calcule correctement les conversions avec des montants positifs.
   — Montant plus que la limite ——1000000.01 : le programme va lancer une exception car le montant maximum disponible est 1000000
   — Montant négatif —— -1.00 : nous testons les montants négatifs pour nous assurer que la méthode génère une exception IllegalArgumentException dans ce cas

— **Analyse des valeurs frontières pour les montants :**

1. Montants proches de la limite inférieure (0.00) : Pour nous assurer que la méthode gère correctement le montant minimum valide.
2. Montants proches de la limite supérieure (1 000 000) : Pour nous assurer que la méthode gère correctement le montant maximum valide.
3. Montants hors la limite (1000000.01 et -0.01) : Pour nous assurer que la méthode génère une exception IllegalArgumentException
4. Montants entre la limite (2500.00) : Pour être sûre que le prix réel est le même que le prix attendu

— **Hypothèses pour les entrées non-valides :**

— Pour les devises invalides, nous nous attendons à un message d'erreur ou à l'ignorance de la conversion.
— Pour les montants invalides, nous anticipons une exception ou un message d'erreur.

— **Conclusion :**

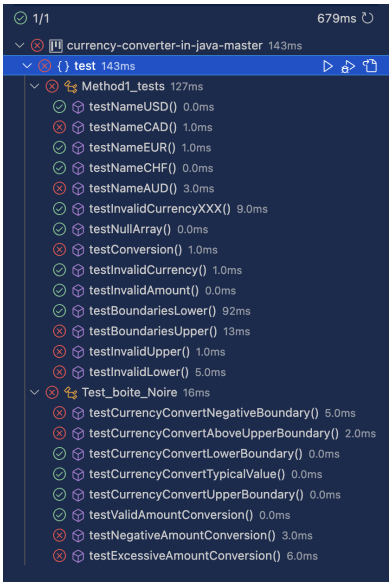


FIGURE 1 – Exécution des tests

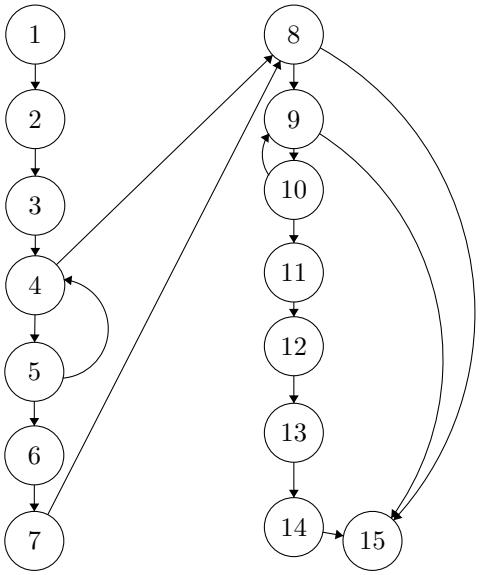
— Nos tests couvrent différentes devises et montants, y compris les limites inférieures esupérieures, garantissant ainsi que la méthode currency.convert fonctionne correctement dans diverses situations. Les hypothèses pour les entrées non-valides ont également été vérifiées aussi.En résumé, la méthode est correcte et gère les exceptions de manière appropriée

2. Tests Boîte Blanche

Méthode `currencyConverter.MainWindow.convert`

— **Critère de couverture des instructions :**
  — D1 : (currency1,currency2,currencies,amount) | currencies.get(i).getName() ==(currency2)
  — D2 : (currency1,currency2,currencies,amount) | currencies.get(i).getName() == currency1
  — **Jeu de test :**
  — T=("USD","EUR",liste de devises incluant USD et EUR,100.0)

— **Critère de couverture des arcs du graphe de flot de contrôle**



— **Critère de couverture des chemins indépendants du graphe de flot de contrôle**
  — 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
  — T=("USD", "EUR", liste de devises incluant USD et EUR, 100.0)
— **Critère de couverture des conditions**
Non-applicable.
On n'a pas de conditions composées.
— **Critère de couverture des i chemins**
  — **Jeu de test :**
  — On saute boucle : currencies est vide
  — 1 iteration - on veut un currencies de taille 1 : T=("USD","EUR",liste de devises avec uniquement USD,100.0)
  — 2 iterations - on veut un currencies de taille 1 : T=("USD","EUR",liste de devises avec uniquement USD et EUR,100.0)
  — m iterations - on veut un currencies de taille m : T=("USD","EUR",liste de devises avec m donnees,100.0)

Méthode currencyConverter.Currency.convert(Double, Double)

— **Critère de couverture des instructions :**
  — D1 : (price = amount * exchangeValue)
  — D2 : (price = Math.round(price * 100d) / 100d)
  — **Jeu de test :**
  — T=("USD","EUR","GBP","CHF","CNY","JPY",Montant au-delà et entre de l'intervalle [0,1000000])

— **Critère de couverture des arcs du graphe de flot de contrôle :**
  — Ce critère n'est pas applicable car il n'y a pas d'instructions conditionnelles (branches if-else) dans la méthode Currency.convert.
— **Critère de couverture des chemins indépendants du graphe de flot de contrôle :**
  — Semblable à la couverture des branches, ce critère n'est pas non plus applicable en raison de l'absence de chemins conditionnels. Il n'existe qu'un seul chemin d'exécution dans la méthode.
— **Critère de couverture des conditions :**
  — Étant donné qu'aucune condition booléenne n'est évaluée dans la méthode, ce critère ne s'applique pas à la méthode Currency.convert
— **Critère de couverture des i-chemins :**
  — Il n'y a pas de boucles dans la méthode Currency.convert, ce critère ne peut donc pas être appliqué.

Résultats des tests et observations

Suite aux tests de la méthode MainWindow.convert. Nous nous sommes aperçus de plusieurs soucis. En effet, nos tests ne passaient pas. Ceci était dû à une comparaison de Currency Name au lieu de Currency Short Name comme le voulait la spécification. De plus, certaines monnaies mentionnées dans la spécification ne figurait pas dans la méthode init(). Finalement, == ne devrait pas être utilisé avec des String. .equals() aurait été plus approprié.