Rapport de TP4 IFT3395

Wen Yin (20179082) Jimmy Yassin Hassanaly (20190749)

Automne 2023

Tâche 3 :

Une hypothèse a été posée pour ces tests : les entrées non valides devraient conduire à un montant converti de -1.

Tests boîtes noire:

Les tests ont été structurés en utilisant JUnit pour réaliser des tests boîte noire, en se basant sur les spécifications fonctionnelles du logiciel. Deux fichiers de test ont été créés pour chaque méthode.

- Fichier **« CurrencyTest1.java»**

Ce fichier teste la méthode currencyConverter.Currency.convert. Les cas de test incluent :
Conversion de montants valides.
Gestion de montants nuls.
Traitement des montants aux limites supérieures spécifiées.
Gestion des montants négatifs et hors limite.
Traitement de taux de change invalides.

- Fichier « **MainWindowTest1.java »**

Ce fichier teste la méthode currencyConverter.MainWindow.convert. Les tests couvrent :
Conversion entre différentes devises supportées pour des montants valides.
Gestion de montants nuls et montants aux limites supérieures.
Gestion de montants négatifs, hors limites, et avec des devises non supportées.

Tests Réussis :

Les tests pour les montants valides, nuls et aux limites supérieures ont réussi, confirmant que le logiciel gère correctement ces scénarios.

Ils démontrent une adéquation du logiciel avec les exigences fonctionnelles de base en termes de conversion de devises et de traitement des montants dans les limites spécifiées.

Tests Échoués :

Les tests pour les montants négatifs, hors limite, et avec des taux de change invalides ou des devises non supportées ont échoué. Cela indique que le logiciel ne gère pas correctement les entrées non valides selon les hypothèses pré-établies.

L'échec des tests pour des cas d'entrée non valides souligne un manque de robustesse dans le traitement des erreurs et des entrées inattendues. Ceci est crucial pour la fiabilité et la sécurité du logiciel.

Tests boîtes blancs :

MainWindow.convert() :

testNormalConversion (Couverture des Instructions) :
Ce test est fait pour exécuter le scénario le plus normal de conversion de devise. Il s'assure que le chemin principal de la méthode convert est exécuté, ce qui inclut le calcul de la conversion de devise. C'est essentiel pour s'assurer que les instructions de base du code fonctionnent comme prévu.
Resultat : Succès.

testUnsupportedCurrencyConversion (Couverture des Arcs) :
En utilisant des devises inexistantes ("XXX" et "YYY"), ce test vise à parcourir une branche spécifique dans le code si elle existe qui gère les devises non reconnues. Ce test est essentiel pour vérifier que le code se comporte correctement quand il rencontre des devises qu'il ne peut pas traiter, couvrant ainsi les transitions de contrôle dans le code pour les entrées inattendues.
Résultat : Echec car retourne 0 mais devrait retourner -1 ou un message d'erreur.

testNegativeAmountConversion (Couverture des Chemins) :
Ce test vérifie le comportement du code lorsque le montant de conversion est négatif, ce qui devrait normalement être traité comme une entrée invalide. Il s'assure que la méthode « convert» gère correctement ce cas particulier, couvrant un chemin d'exécution distinct qui pourrait impliquer une logique de validation ou de gestion des erreurs.
Résultat : Echec car retourne une valeur négative calculée au lieu de retourner -1 ou message d'erreur.

testZeroAmountConversion (Couverture des Conditions) :
Ce test évalue comment le code réagit à une conversion avec un montant de zéro. Cela permet de tester une condition importante (le montant étant exactement zéro), ce qui est important pour s'assurer que les cas limites sont correctement gérés dans le code.
Résultat : Succès car retourne 0.

testAmountAboveLimitConversion (Couverture des i-Chemins) :
Ce test explore un scénario où le montant est juste au-dessus de la limite autorisée. Il teste les interactions entre les conditions de limite de montant et de validité de la devise, couvrant des cas où différentes branches du code interagissent.
Résultat : Echec car retourne le montant calculé alors que c'est au dessus de la limite autorisé au lieu de retourner -1 ou message d'erreur.

testUnsupportedCurrenciesAndInvalidAmount (Couverture des i-Chemins) :
En fournissant à la fois des devises non supportées et un montant invalide, ce test évalue comment le code gère la combinaison de plusieurs conditions d'erreur. Cela

aide à couvrir des chemins dans le code où différentes branches de logique se croisent, et ainsi pour mieux identifier l'erreur.
Résultat : Echec car retourne 0 au lieu de retourner -1 ou message d'erreur.

testAmountAtUpperLimit (Couverture des Conditions) :
Ce test vérifie le comportement du code pour un montant qui est exactement à la limite supérieure acceptée. Cela permet de tester une condition limite spécifique, s'assurant que la méthode « convert » traite correctement les montants qui sont à la frontière de la validité.
Résultat : Succès car retourne la valeur attendu.

Convert.convert() :
testNormalConversion (Couverture des Instructions) :
Teste une conversion de devise normal avec des entrées valides. Cela garantit que les instructions de base pour calculer la conversion sont correctement exécutées, couvrant ainsi le chemin d'exécution principal de la méthode.
Résultat : Succès car retourne la valeur attendue.

testNegativeAmount (Couverture des Arcs) :
En testant avec un montant négatif, ce test cible une section spécifique du code qui doit gérer les montants négatifs comme étant invalides. Il s'assure que le code réagit correctement à ces entrées, en testant les transitions de contrôle pour les cas d'erreur.
Résultat : Echec car retourne un résultat négatif calculé au lieu de retourner -1 ou message d'erreur.
testZeroExchangeRate (Couverture des Chemins) :
Ce test examine un scénario spécifique où le taux de change est zéro. Cela permet de voir la réaction du code à un taux de change nul.
Résultat : Succès.

testNegativeExchangeRate (Couverture des Conditions) :
Teste la réaction du code à un taux de change négatif. Cela teste une condition spécifique dans le code, s'assurant que les valeurs de taux de change invalides sont correctement gérées.
Résultat : Echec car retourne un résultat négatif calculé au lieu de retourner -1 ou message d'erreur.

testZeroAmountNegativeExchangeRate (Couverture des i-Chemins) :
Teste la conversion avec un montant nul et un taux de change négatif. Ce test couvre les interactions entre deux conditions indépendantes, ce qui est important pour vérifier le comportement du code dans des scénarios où plusieurs conditions sont impliquées.
Résultat : Echec car retourne 0 calculé au lieu de retourner -1 ou message d'erreur.

Conclusion :

En général, les méthodes *convert* des 2 classes sont conçues pour gérer les cas des conversions normales uniquement, donc des devises valides, des montants positifs et des taux de change positifs. Par contre, elles ne gèrent en aucun cas les cas d'erreurs en particulier les devises inexistantes, des montants négatives et des taux de changes négatives alors que ces cas là devraient générer des cas d'erreurs. On peut donc dire que les méthodes contiennent une seule branche qui calcule la conversion peut importe ce qu'on donne en entrée. Chaque test réalisé permet d'entrer dans une branche spécifique du code si elle existe et tester les valeurs frontières du code. Cependant, la plupart des tests spécifiques ont échouées. La structure du code des méthodes et le non traitement de ces cas spécifiques fait en sorte que les tests boîtes noirs et boîtes blanches se ressemblent énormément. Le code a une logique linéaire et directe, sans de multiples chemins ou conditions complexes, les tests nécessaires pour couvrir le code (boîte blanche) ressemblent donc à ceux qui vérifient simplement que la fonctionnalité fonctionne comme prévu (boîte noire).