Sara Gair (20216922) Geneviève Ouellette (20141307) IFT3913
8 décembre 2023

Tests boîtes noires

Les tests sont basés sur ces deux spécifications:

- Convertir des devises: USD, CAD, GBP, EUR, CHF, AUD.
- Accepter des montants entre 0 et 1 000 000.

MainWindowConvertTest qui teste currencyConverter.MainWindow.convert(String, String, ArrayList, Double) :

Pour tester la première spécification, MainWindowConvertTest prend un montant de valeur 5 et le convertit selon les devises. Donc, par exemple, 5 USD à 5 * devise CAD, 5 USD à 5 * devise GBP, etc.

Le jeu de test est donc: {USD, CAD, GBP, EUR, CHF, AUD}.
Résultat: les tests donnent les bonnes valeurs pour convertir USD, GBP, EUR et CHF. Ils réussissent aussi pour CNY et JPY. Ils ne marchent pas pour CAD et AUD.

Hypothèse si les devises CAD et AUD seraient implantées:
En prenant la devise 1 USD = 1.36 CAD (date: 7 décembre 2023), le résultat attendu est 5$ USD = 6.8$ CAD et l'inverse serait 5$ CAD = 3.68 $ USD. Il y aurait aussi les autres combinaisons qui marcheraient (exemple GBP→CAD/ CAD→GBP). Pour AUD, c'est le même processus avec 1 USD = 1.52 AUD. Il faudrait ajouter son taux de change et tester les autres combinaisons comme CAD.

Pour la deuxième spécification, on assume ces résultats:

- Montant < 0 : test retourne 0 ou échoue
- 0 < Montant < 1 000 000: test réussi et donne le bon montant selon le taux de change
- 1 000 000 < le test échoue (le montant est refusé)

Pour la première et la troisième condition, les tests ne donnent pas les valeurs attendues. En effet, avec un montant USD de -2$, le programme retourne -247.08 JPY, ce qui devrait être 0 ou donner une valeur nulle puisque le programme ne devrait pas accepter de valeur négative. Pour la limite de droite, le test échoue si, par exemple, on donne un montant de 1 500 000 USD. Il retourne 1 395 000 EURO alors qu'il devrait retourner une valeur nulle pour indiquer qu'il ne peut pas accepter de valeur plus grande que zéro.

TestCurrency qui teste currencyConverter.Currency.convert(Double, Double)

En ce qui concerne la première spécification, cette méthode réussit avec les devises CAD et AUD en donnant les bons taux de change (et les autres devises aussi). Deux combinaisons de devises sont testées: USD et CAD. Les tests sont répétitifs donc seulement ces deux devises sont testées mais elles sont représentatives de l'ensemble du jeu de données.
On teste donc USD→CAD, USD→EUR, USD→GBP… en mettant les taux de change. Le même processus est fait pour CAD avec les tests CAD→USD, CAD→EUR…
Les devises pour AUD et CAD sont prises sur Google.com en date du 7 décembre tandis que les autres taux de change proviennent du fichier Currency.java.
La première spécification est respectée pour ce jeu de donnée: {USD, CAD, GBP, EUR, CHF, AUD}.
Les tests des limites ont donné les mêmes résultats que MainWindowConvertTest. Cela indique que c'est un cas n'ayant pas été couvert par le programme.

Tests boîte blanche

MainWindow.convert(nameCurrency1, nameCurrency2, currencies, amount)

(Les jeux de test ont été exclus pour sauver de l'espace. Ils sont disponibles en commentaire dans le code.)

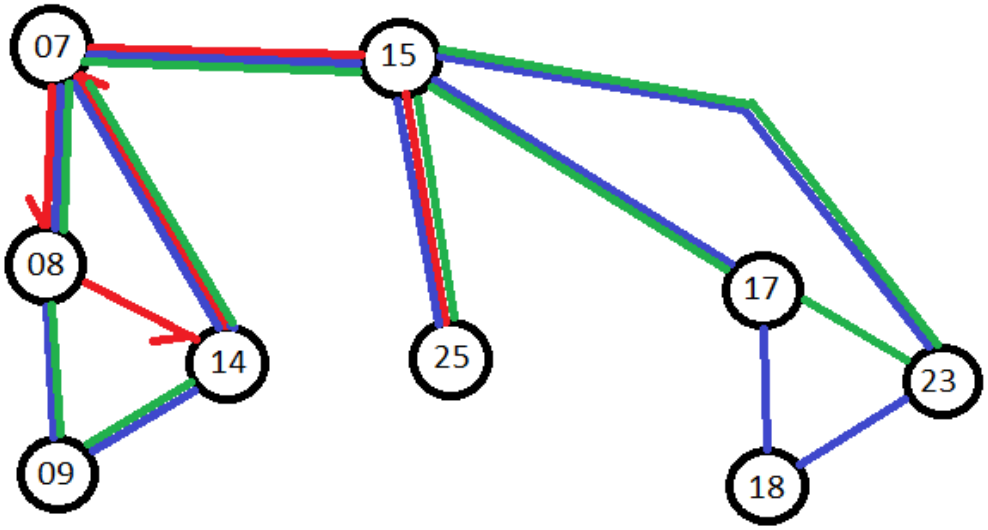
A. Critère de couverture des instructions
D1 = {(nameCurrency1, nameCurrency2, currencies, amount)| nameCurrency1 est dans currency.names (sans être le premier), nameCurrency2 est dans currency.names (sans être le premier)}
Ainsi, toutes les lignes sont parcourues.

B. Critère de couverture des arcs du graphe de flot de contrôle

D1 = {(nameCurrency1, nameCurrency2, currencies, amount)| nameCurrency2 is not in currency.names}

D2 = {(nameCurrency1, nameCurrency2, currencies, amount)| nameCurrency1 is in currency.names, nameCurrency2 is in currency.names}

D3 = {(nameCurrency1, nameCurrency2, currencies, amount)| nameCurrency1 is not in currency.names, nameCurrency2 is in currency.names}



C. Critère de couverture des chemins indépendants
A: Boucle for de 07-12, 3 choix: {skip A, run A (avec ifA == false), run A (avec ifA == true)}
C: If de 15-22, 2 choix: {else C (passer B), if C (continuer vers B)}
B: Boucle for de 16-22, 3 choix: {skip B, run B (avec ifB == false), run B (avec ifB == true)}
Chemin 1 : (skip A, else C)
Chemin 2 (X): (skip A, if C, run B)(impossible d'avoir skip A et if C)
Chemin 3-4 (X): (run A, skip B) (impossible, car si A run, B run aussi, car ils sont tous les deux des boucles sur currencies)
Chemin 5 : (run A, ifA true, if C, run B, ifB true)
Chemin 6 : (run A, ifA true, if C, run B, ifB false)
Chemin 7 (X): (run A, ifA false, if C) (impossible, car la ligne 9 de A n'est pas exécutée et elle est nécessaire pour avoir if C)
Chemin 8 : (run A , ifA false, skip C)

D. Critère de couverture des conditions (et des arcs)
Pas de conditions composées

E. Couverture des i-chemins
A: Boucle for de 07-12, 7 choix: {skip A, run A une fois (ifA true), run A deux fois (ifA true), run A six fois (ifA true), run A une fois (ifA toujours false), run A deux fois (ifA toujours false), run A six fois (ifA toujours false)}
C: If de 15-22, 2 choix: {else C (passer B), if C (continuer vers B)}

B: Boucle for de 16-22, 7 choix: {skip B, run B une fois (ifB true), run B deux fois (ifB true), run B six fois (ifB true), run B une fois (ifB toujours false), run B deux fois (ifB toujours false), run B six fois (ifB toujours false)}
Sur la base des chemins valides trouvés en C.
Chemin 1 : (skip A, else C)
Chemin 5A:(run A (1 fois), ifA true, if C, run B (1 fois), ifB true)
Chemin 5B:(run A (1 fois), ifA true, if C, run B (2 fois), ifB true)
Chemin 5C:(run A (1 fois), ifA true, if C, run B (6 fois), ifB true)
Chemin 5D:(run A (2 fois), ifA true, if C, run B (1 fois), ifB true)
Chemin 5E:(run A (2 fois), ifA true, if C, run B (2 fois), ifB true)
Chemin 5F:(run A (2 fois), ifA true, if C, run B (6 fois), ifB true)
Chemin 5G:(run A (6 fois), ifA true, if C, run B (1 fois), ifB true)
Chemin 5H:(run A (6 fois), ifA true, if C, run B (2 fois), ifB true)
Chemin 5I:(run A (6 fois), ifA true, if C, run B (6 fois), ifB true)
Chemin 6A:(run A (1 fois), ifA true, if C, run B (1 fois), ifB false)
Chemin 6B:(run A (1 fois), ifA true, if C, run B (2 fois), ifB false)
Chemin 6C:(run A (1 fois), ifA true, if C, run B (6 fois), ifB false)
Chemin 6D(X):(run A (2 fois), ifA true, run C, run B (1 fois), ifB false)(impossible : B tourne jusqu'à currency.size = 2)
Chemin 6E:(run A (2 fois), ifA true, if C, run B (2 fois), ifB false)
Chemin 6F:(run A (2 fois), ifA true, if C, run B (6 fois), ifB false)
Chemin 6G(X):(run A (6 fois), ifA true, run C, run B (1 fois), ifB false)(impossible : B tourne jusqu'à currency.size = 6)
Chemin 6H(X):(run A (6 fois), ifA true, run C, run B (2 fois), ifB false)(impossible : B tourne jusqu'à currency.size = 6)
Chemin 6I:(run A (6 fois), ifA true, run C, run B (6 fois), ifB false)
Chemin 8A:(run A (1 fois), ifA false, skip C)
Chemin 8B:(run A (2 fois), ifA false, skip C)
Chemin 8C:(run A (6 fois), ifA false, skip C)

Currency.convert(amount, exchangeValue)

A. Critère de couverture des instructions
D1 = {(x, y)|x est dans R (les réels) et y est dans R}
Il n'y a ni boucle ni if-else, toutes les lignes sont visitées.

B. Critère de couverture des arcs du graphe de flot de contrôle
C. Critère de couverture des chemins indépendants
D. Critère de couverture des conditions (et des arcs)
E. Couverture des i-chemins
Pour B, C, D, E : Aucun if-else, aucune boucle, pas de changements