Tests boîte noire

Partition du domaine des entrées en classes d'équivalence
C1 = [USD, CAD, GBP, EUR, CHF, AUD]
C2 = [JPY, CNY]

D1 = { 0 <= d <= 1 000 000 }
D2 = {d < 0}
D3 = {d > 1 000 000}

Analyse des valeurs frontières
On veut inclure dans le de test pour les montants les valeurs de -1, 0, 1 000 000 et 1 000 001.

Hypothèse du comportement du code pour des entrées non-valides
On suppose que si on entre une devise invalide ou un montant invalide, le code renvoie -1.

Jeu de test pour les montants pour les deux méthodes
On va tester les montants suivants
{-500 000, -1, 0, 500 000, 1 000 000, 1 000 001, 1 500 000}

Jeu de test pour les devises pour la méthode de *MainWindow*
On va tester les couples de devises suivants
{{USD,EUR}, {USD,CNY}, {CNY,JPY}}

Tests boîte blanche

currencyConverter.MainWindow.convert(String, String, ArrayList, Double)
Critères de sélection de jeux de test

1. Critère de couverture des instructions
Le code contient des conditions *if... then... else...*, alors il faut s'assurer que notre jeu de test contient les cas où les conditions sont respectées pour exécuter les instructions qu'elles contiennent.

2. Critère de couverture des arcs du graphe de flot de contrôle
Le code contient des conditions *if... then... else...*, alors il faut s'assurer que notre jeu de test contient les cas où les conditions sont respectées pour traverser les arcs de ces parties du graphe de flot de contrôle.

3. Critère de couverture des chemins indépendants
Le code contient des conditions *if... then... else...*, alors il faut s'assurer que notre jeu de test

contient les cas où les conditions sont respectées pour traverser les arcs de ces parties du graphe de flot de contrôle et d'autres cas où on ne traverse pas ces arcs.

4. Critère de couverture des conditions
On ne prend pas en compte ce critère, car le code de cette méthode ne contient pas de conditions composées.

5. Critère de couverture des i-chemins
Le code contient des boucles simples, alors notre jeu de test doit couvrir chacun des cas suivants:
• on saute la boucle
• une itération de la boucle
• deux itérations
• m itérations (m<n)
• n-1, n et n+1 itérations…

On ne peut pas sauter la première boucle *for*..., mais on peut sauter la deuxième boucle. Pour ce faire, il faut que le deuxième paramètre soit une devise qui n'est pas supportée par la méthode comme "US Dollars", puisque ce n'est pas écrit "US Dollar".

Pour faire une seule itération de la première boucle, il faut que le deuxième paramètre *currency2* soit "US Dollar". Pour faire une seule itération de la deuxième boucle, il faut que le premier paramètre *currency1* soit "US Dollar" et que le deuxième paramètre soit une devise supportée par la méthode comme "British Pound".

Pour faire deux itérations de la première boucle, il faut que le deuxième paramètre *currency2* soit "Euro". Pour faire deux itérations de la deuxième boucle, il faut que le premier paramètre *currency1* soit "Euro" et que le deuxième paramètre soit une devise supportée par la méthode comme "British Pound".

Pour faire m=4<n itérations de la première boucle, il faut que le deuxième paramètre *currency2* soit "Swiss Franc". Pour faire m=4<n itérations de la deuxième boucle, il faut que le premier paramètre *currency1* soit "Swiss Franc" et que le deuxième paramètre soit une devise supportée par la méthode comme "British Pound".

Pour faire n-1 itérations de la première boucle, il faut que le deuxième paramètre *currency2* soit "Chinese Yuan Renminbi". Pour faire n-1 itérations de la deuxième boucle, il faut que le premier paramètre *currency1* soit "Chinese Yuan Renminbi" et que le deuxième paramètre soit une devise supportée par la méthode comme "British Pound".

Pour faire n itérations de la première boucle, il faut que le deuxième paramètre *currency2* soit "Japanese Yen". Pour faire n itérations de la deuxième boucle, il faut que le premier paramètre *currency1* soit "Japanese Yen" et que le deuxième paramètre soit une devise supportée par la méthode comme "British Pound".

Pour faire n+1 itérations de la première boucle, il faut que le deuxième paramètre *currency2* soit une devise non supportée par la méthode comme “US Dollars”. Pour faire n+1 itérations de la deuxième boucle, il faut que le premier paramètre *currency1* soit une devise non supportée par la méthode comme “US Dollars” et que le deuxième paramètre soit une devise supportée par la méthode comme “British Pound”.

Donc, notre jeu de test est le suivant :
Le jeu de test ne prend en compte que les deux premiers paramètres de la méthode. Le troisième paramètre *currencies*, est toujours le même et est défini par la méthode *init()* de la classe *Currency*. Le quatrième paramètre n’a pas d’importance tant qu’il est entre 0 et 1 000 000 inclusivement.

{ {*currency1,currency2}, ... }*

{ {“British Pound”,“US Dollars”}, {“British Pound”,“US Dollar”}, {“US Dollar”, “Bristish Pound”}, {“British Pound”,“Euro”}, {“Euro”, “Bristish Pound”}, {“British Pound”,“Swiss Franc”}, {“Swiss Franc”, “Bristish Pound”}, {“British Pound”,“Japanese Yen”}, {“Japanese Yen”, “Bristish Pound”}, {“British Pound”,“US Dollars”}, {“US Dollars”, “Bristish Pound”} }

currencyConverter.Currency.convert(Double, Double)
Critères de sélection de jeux de test
1. Critère de couverture des instructions
Peu importe comment on exécute cette méthode, toutes les instructions seront exécutées une fois.

2. Critère de couverture des arcs du graphe de flot de contrôle
Peu importe comment on exécute cette méthode, tous les arc du graphe de flot de contrôle est traversé une fois. Le graphe de flot de contrôle est linéaire.

3. Critère de couverture des chemins indépendants
On ne prend pas en compte ce critère, car le graphe de flot de contrôle est linéaire et donc la complexité cyclomatique est de 1. Ce critère est trop complexe pour cette méthode simple.

4. Critère de couverture des conditions
On ne prend pas en compte ce critère, car le code de cette méthode ne contient pas de condition *if… then… else…* .

5. Critère de couverture des i-chemins
On ne prend pas en compte ce critère, car le code de cette méthode ne contient pas de boucle.

Le deuxième critère est plus couvrant que le premier, on utilise que le deuxième critère, le critère de couverture des arcs du graphe de flot de contrôle.

Tant que les valeurs passées en paramètre sont entre 0 et 1000 000 (inclusivement) pour le premier et entre 0 et 1 (inclusivement) pour le second, tous les arcs seront traversés une fois. Puisque le graphe de flot de contrôle est linéaire, notre jeu de test peut être un seul couple de valeur comme celui-ci {5000, 0.74} où la première valeur est le premier paramètre de la méthode et la deuxième valeur est le deuxième paramètre.

Résultats de nos tests

Pour les méthodes *convert* de *MainWindow* et *Currency*, les tests de la boîte noire que nous avons définis ne sont pas tous réussis, car on a supposé que la spécification des *convert* accepte seulement des montants entre [0, 1 000 000] et on a aussi supposé que les méthodes retournent -1 lorsqu'un montant hors de cet intervalle est passé en paramètre.

Tous les tests de la boîte blanche que nous avons écrits réussissent.

Nous observons que les tests de la boîte noire sont moins concluants, puisque nous les avons écrit à l'aveugle, c'est-à-dire sans regarder le code.

Selon les tests écrits, les tests de la boîte blanche semblent être plus efficaces pour des méthodes simples. En effet, on a eu besoin d'un seul test pour couvrir les critères pertinents à *convert* de la classe *Currency*, qui a 4 lignes de code, et on a eu besoin de 7 tests de la boîte noire. Pour *convert* de la classe *MainWindow*, qui est plus complexe que celle de *Currency*, on a eu besoin de 11 tests de la boîte blanche pour couvrir les critères pertinents et 9 tests de la boîte noire. Ceci est dû au fait que les critères de couverture des chemins indépendants du graphe de flot de contrôle, de couverture des conditions et de couverture des i-chemins ne sont pas pris en compte lors de l'écriture des tests pour une méthode aussi simple de *convert* de la classe *Currency*, et les critères de couverture des instructions et de couverture des arcs du graphe de flot de contrôle peut être respectés avec un seul test. Toutefois, pour une méthode qui contient des boucles et des conditions comme *convert* de la classe *MainWindow*, nous avons besoin de davantage de tests pour respecter toutes les conditions. La quantité de tests de la boîte noire écrits ne varie pas beaucoup d'une méthode à une autre puisqu'on prend seulement en compte la spécification des méthodes et non leur complexité.