Rapport TP4

Vyncent Larose (20189960)
Haunui Louis (21089376)

Boîte noire :

On suppose que la spécification du « Currency Converter » exige que:

• Il doit convertir des montants entre les devises suivantes : USD, CAD, GBP, EUR, CHF, AUD.
• Il doit seulement accepter des montants entre [0, 1 000 000].

Nous allons tout d'abord faire le partitionnement d'équivalence.
Pour les devises valides : D1= {USD, CAD, GBP, EUR, CHF, AUD}
Pour les devises invalides : D2 = {d | d n'appartient pas à D1}

Et pour les montants valides : D3 = [0,1 000 000],
ainsi que pour les montants invalides : D4 = {m | m < 0 ou m > 1 000 000}

Nous utilisons le partition du domaine des entrées en classes d'équivalence pour réduire le nombre de tests, la logique étant que si le logiciel fonctionne sur une entrée de la classe d'équivalence, il devrait fonctionner pour toutes les entrées de la même classe.

Pour les devises, nous testerons un sous-ensemble représentatif au lieu de tester toutes les conversions entre les paires possibles. Pour cela nous convertirons 100 USD en toutes les autres devises.

Nous testerons ensuite une devise non-incluse dans D1 (par exemple AUD)

En ce qui concerne les montants, nous testerons une valeur négative, une valeur dans l'intervalle et une valeur supérieure au maximum.

Nous ajoutons également les valeurs frontières aux tests, on obtient donc le jeu de test qui suit : {-1,0,1,500 000,999 999,1 000 000,1 000 001}.

Nous nous attendons à ce que le code retourne une valeur indiquant une erreur : null / exception / ou un zéro, pour les cas invalides (nous testons avec une valeur de zéro pour indiquer si la conversion a eu lieu).

Les résultats aux tests sont :

[ERROR] Failures:
[ERROR] CurrencyTest.testBoundaryValuesCurrency:21 Conversion should fail for invalid amount: -1.0 ==> expected: <0.0> but was: <-1.5>

[ERROR] CurrencyTest.testBoundaryValuesCurrency:21 Conversion should fail for invalid amount: 1000001.0 ==> expected: <0.0> but was: <1500001.5>
[ERROR] MainWindowTest.testBoundaryValues:73 Conversion should fail for invalid amount: -1.0 ==> expected: <0.0> but was: <-0.93>
[ERROR] MainWindowTest.testBoundaryValues:73 Conversion should fail for invalid amount: 1000001.0 ==> expected: <0.0> but was: <930000.93>
[ERROR] MainWindowTest.testConvertUSDtoAUD:45 Conversion failed from US Dollar to Australian Dollar ==> expected: <true> but was: <false>
[ERROR] MainWindowTest.testConvertUSDtoCAD:21 Conversion failed from US Dollar to Canadian Dollar ==> expected: <true> but was: <false>
[ERROR] MainWindowTest.testInvalidCurrency:55 Conversion should fail to Japanese Yen ==> expected: <0.0> but was: <12354.0>

Entre autres :

- Aucune des deux méthodes ne gèrent les valeurs en dehors des frontières (car elles fonctionnent et retourne un résultat au lieu de retourner zéro).
- La conversion en AUD et CAD ne fonctionne pas.
- La conversion en une valeur invalide (JPY) fonctionne.

À noter également que si on utilise @ParameterizedTest pour tenter d'être plus compacte, la méthode MainWindow.convert retourne toujours zéro pour des @ValueSource en strings, ce qui nous laisse croire qu'elle n'utilise pas une comparaison de String correcte (== au lieu d'utiliser String.equals).

Nos tests de boîte noire indiquent que les méthodes MainWindow.convert et Currency.convert ne sont pas conformes à la spécification supposée.

D'après ces tests, il semble que les méthodes suivent cette spécification :

• Il doit convertir des montants entre les devises suivantes : USD, GBP, EUR, CHF,JPY.
• Il accepte tous les montants.

Boîte blanche:

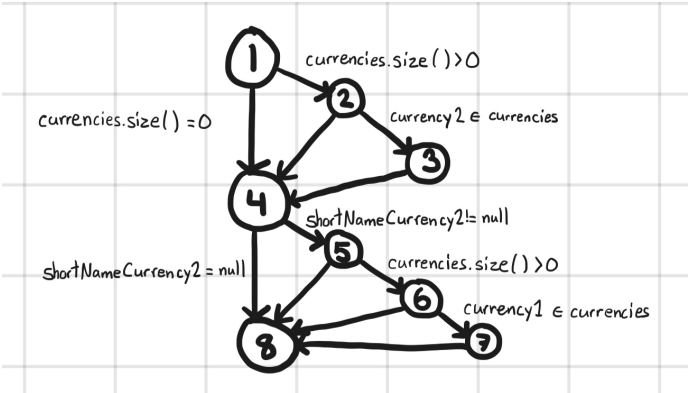
MainWindow.convert:
1. Couverture des instructions: Afin de couvrir toutes les lignes de code de cette méthode, en considérant les arguments suivants dans son appel: String currency1, String currency2, ArrayList<Currency> currencies, Double amount.
Nous devons faire des vérifications lorsque currencies.size() > 0; currencies.get(i).getName() == currency2; shortNameCurrency2 != null; currencies.get(i).getName() == currency1.

testConvertWithValidCurrencies(): On y utilise des monnaies existantes, “US Dollar” et “Euro”. L’arraylist currencies est donc non vide et on sait que shortNameCurrency2 n’est pas nul puisqu’on utilise une monnaie connue et
shortNameCurrency2 = currencies.get(i).getShortName()

2. Critère de couverture des arcs du graphe de flot de contrôle: tests pour les chemins non parcourus par le test de couverture des instructions, c’est à dire: si currency2 n’est pas dans currencies; si shortNameCurrency2 est nul; si currency1 n’est pas dans currencies. Ceci est fait avec testConvertWithInvalidCurrencies(). On teste donc avec deux monnaies inexistantes en même temps, ce qui fait d’ailleurs en sorte que shortNameCurrency2 reste nul pendant l’exécution. On vérifie aussi ce qu’il se produit lorsque chaque monnaie est invalide alors que l’autre est valide.

3-4-5. Vu la simplicité de la méthode étudiée, nous ne croyons pas nécessaire de faire un test pour la couverture des chemins indépendants. Le critère testé au paragraphe précédent suffit.
Les deux premiers tests pour cette méthode sont amplement suffisant, tester le critère de couverture des conditions (et des arcs) ainsi que le critère de couverture des i-chemins semble redondant.

Cependant, les méthodes testConvertNullShortNameCurrency2() et testConvertEmptyCurrencies() sont utilisées pour couvrir le plus de cas possibles. Nous considérons que cela suffit. Le graphe suivant nous a permis de s’assurer que c’était exhaustif:



À noter que nous sommes conscient que si currencies.size() = 0, il y aurait plus de chemins dans le graphe, car en aucun cas il serait possible d'avoir le chemin suivant: 1-4-5-6-7-8 (si currencies.size() = 0 au début, il ne peut pas changer de valeur pour passer de 5 à 6).

Pour ce qui est de tests boîte blanche pour Currency.convert, il ne suffit que d'utiliser des valeurs arbitraires afin de s'assurer que l'exécution est effectuée. Cependant, il n'y a aucune condition limitant l'accès à certaines lignes de code. Ceci étant dit, c'est plutôt les tests de boîte noire qui sont utiles ici, car seuls des erreurs d'utilisation peuvent causer un problème ici.

Aucun test de boîte blanche n'échoue.