TP4 – IFT 3913 Qualité du logiciel et métriques – Automne 2022

Auteurs : Anthony Grange (20160453) & Luchino Allix-Lastrego (20222844)

TESTS BOÎTE NOIRE :

Nous devons étudier à l'aide de tests boîte noire la méthode *convert()*. Pour cela nous devons définir une partition du domaine des entrées. On suppose que la spécification exige que les devises soient : {USD, CAD, GBP, EUR, CHF, INR, AUD}. Et que le montant soit dans l'intervalle : [0,10000].

Vu que *convert()* requiert deux devises, nous avons opté pour deux devises de la spécification et deux qui n'en font pas partie, respectivement {CAD, EUR} et {NOK, BOB}.

Pour les valeurs des montant, nous avons pris {-1000,3000,15000} pour représenter les classes d'équivalences {0 ≤ d ≤ 10000}, {d ≤ 0} et {10000 ≤ d}. Pour les valeurs frontières nous avons pris {0-i,0} et {10000,10000+i} où i = 2^{-1074}, la plus petite valeur qu'un double puisse représenter en java. Ce qui nous donne comme jeu de valeurs {-1000, 0-i,0, 3000,10000,10000+i,15000}.

Pour chaque valeur nous avons tester la fonction avec deux fois la même devise valide, deux devises valides différentes, une devise valide et une non valide, deux devises invalides différentes et deux fois la même devise invalide. Ce qui nous fais 5 cas différents pour 7 montant différents soit 35 tests à effectuer. Les tests boite noire sont tous défini sous la forme :

```
@Test
Public void TesBNX() throws ParseException{…}
```

Où X prend une valeur entre 1 et 35. Dans chaque test nous comparons l'appel de la fonction avec un calcul manuel de la conversion, à savoir :

$$conversion = montant * \frac{taux\ devise\ d'arriver}{taux\ devises\ de\ départ}$$

Résultats : Peu importe les devises et les montants, le calcul manuel de la conversion donne le même résultat que *convert()*.

Conclusion : Les valeurs qui doivent fonctionner fonctionnent bien. Les valeurs qui ne sont pas dans la spécification, où l'on pourrait s'attendre à une erreur, fonctionnent aussi. Cela veut donc dire que la méthode *convert()*. n'est pas limitée à sa spécification en termes d'entrées de montant ou de devises. Nous n'avons donc pas détecté d'erreur ou d'omission, mais cela n'indique pas qu'il ne peut pas y en avoir.

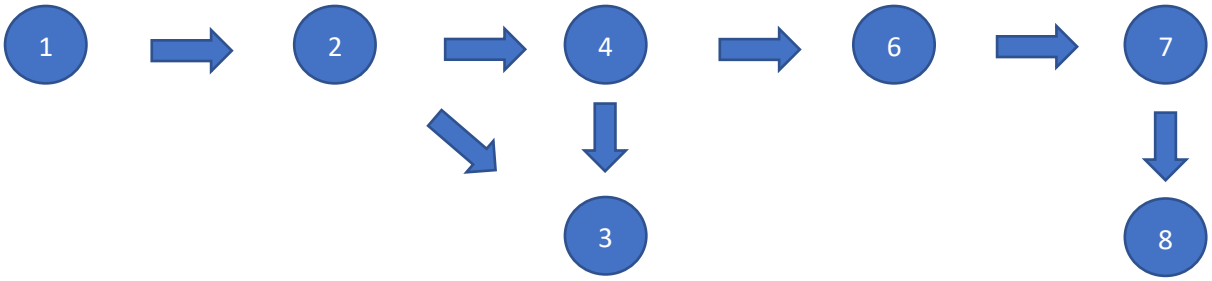
TESTS BOÎTE BLANCHE :

Nous devons maintenant étudier à l'aide de tests boîte blanche la méthode *convert()*. Pour cela voici une copie du code avec des annotation pour faciliter la compréhension :

```
public static double convert(double amount, String from, String to,
CurrencyConversion conversion) throws ParseException{ (1)
	if(!conversion.getRates().containsKey(to)) (2)
		throw new ParseException("Not correct format currency" + "", 0); (3)
	else if(!conversion.getRates().containsKey(from)) (4)
		throw new ParseException("Not correct format currency" + "", 0); (5)
	double curencyTo = conversion.getRates().get(to); (6)
	double curencyFrom = conversion.getRates().get(from); (7)
	return amount*(curencyTo/curencyFrom); (8)
}
```

* Nous avons diviser le branchement conditionnel « `if` » en deux car il contient un « `||` ». Et l'on peut noter que `(3) = (5)`.

Regardons le graph de flot de contrôle :



Regardons les différents critères :

A) Critère de couverture des instructions : Il faut couvrir toutes les instructions, vu qu'il y a un if qui contient un or, il faut tester les deux conditions séparément. Pour le reste, il suffit de mettre des arguments valides, ainsi on arrivera au return et nous aurons essayé toutes les instructions.

B) Critère de couverture des arcs du graphe de flot de contrôle : Nous devons nous intéresser aux branchements de contrôle, vu qu'il n'y en a qu'un, cela revient à regarder les mêmes paramètres que le test A.

C) Critère de couverture des chemins indépendants du graphe de flot de contrôle : Revient à regarder les critères de A et B car il faut regarder tous les 1-chemins, or vu qu'il n'y a pas de boucle dans le code, tous les chemins sont des 1-chemins.

D) Critère de couverture des conditions : Encore une fois, vu qu'il n'y a pas de boucle, emprunter tous les arcs au moins une fois revient à regarder les critères des couvertures des tests précédant.

E) Critère de couverture des i-chemins : De même, couvrir les i-chemins revient à regarder les critères précédant vu qu'il n'y a que des 1-chemins.

Donc pour parcourir tous les chemins il faut un set avec des valeurs valides, un set où la devise de départ est invalide et un ou la devise d'arriver est invalide.

Nous avons donc lancé le programme avec comme paramètres (3000, "CAD", "EUR"), (3000, "", "EUR") et (3000, "EUR", ""). Les tests boite noire sont tous défini sous la forme :

```
@Test

Public void TesBBX(){…}
```

Où X prend une valeur entre 1 et 3. Le test avec de bonnes entrées se déroule comme les tests boite noire, pour les autres nous vérifions qu'une erreur soit bien appelée.

Touts les tests sont réussis, donc on peut déterminer qu'il n'y a pas d'erreur mais on ne peut pas déterminer qu'il n'y a pas d'omissions.