RAPPORT - DEVOIR 4

Réalisé par Noémie Arbour et Nhung Dang Thi Hong

Test boite noire

Programme Currency Converter défini sur $\{USD,\ CAD,\ GBP,\ EUR,\ CHF,\ AUD\}$

On définit 2 classes d'équivalence pour les devises :

- Les valeurs d'entrées valides : $C_1 = \{USD,\ CAD,\ GBP,\ EUR,\ CHF,\ AUD\}$
- Les valeurs d'entrées invalides : $C_2 \notin C_1 = \{JPY,\ CNH,\ SEK,\ NZD\}$

Un jeu de test valide serait: $T_1 = \{USD,\ JPY\}$

Programme "Currency Converter" défini sur [0, 1000000]

Cela signifie que pour toutes devises, on définit 3 classes d'équivalence:

- $D_1 = \{0 \leq d \leq 1000000\}$
- $D_2 = \{d < 0\}$
- $D_3 = \{d > 1000000\}$

En ajoutant les valeurs aux frontières, on choisit le jeu de test :
$T_2 = \{-50,\ -0.01,\ 0,\ 5000,\ 1000000,\ 1000000.01,\ 1000500\}$

Hypothèse pour le comportement du code pour des entrées non-valides:

Une entrée non-valide lance ("throw") une exception.

Les tests pour la méthode 1 (currencyConverter.MainWindow.convert(String, String, ArrayList, Double))

1.	T1 = {"USD","USD", C_1, **-50**}	Fail	2.	T2 = {"**JPY**", "USD", C_1, **-50**}	Fail	3.	T3 = {"**JPY**", "**JPY**", C_1, **-50**}	Fail
4.	T4 = {"USD", "**JPY**", C_1, **-50**}	Fail	5.	T5 = {"USD","USD", C_1, **-0.01**}	Fail	6.	T6 = {"**JPY**", "USD", C_1, **-0.01**}	Fail
7.	T7 = {"**JPY**", "**JPY**", C_1, **-0.01**}	Fail	8.	T8 = {"USD", "**JPY**", C_1, **-0.01**}	Fail	9.	T9 = {"USD","USD", C_1, 0}	Pass
10.	T10 = {"**JPY**", "USD", C_1, 0}	Fail	11.	T11 = {"**JPY**", "**JPY**", C_1, 0}	Fail	12.	T12 = {"USD", "**JPY**", C_1, 0}	Fail
13.	T13 = {"USD","USD", C_1, 5000}	Pass	14.	T14 = {"**JPY**", "USD", C_1, 5000}	Fail	15.	T15 = {"**JPY**", "**JPY**", C_1, 5000}	Fail
16.	T16 = {"USD", "**JPY**", C_1, 5000}	Fail	17.	T17 = {"USD","USD", C_1, 1000000}	Pass	18.	T18 = {"**JPY**", "USD", C_1, 1000000}	Fail
19.	T19 = {"**JPY**", "**JPY**", C_1, 1000000}	Fail	20.	T20 = {"USD", "**JPY**", C_1, 1000000}	Fail	21.	T21 = {"USD","USD", C_1, **1000000.01**}	Fail
22.	T22 = {"**JPY**", "USD", C_1, **1000000.01**}	Fail	23.	T23 = {"**JPY**", "**JPY**", C_1, **1000000.01**}	Fail	24.	T24 = {"USD", "**JPY**", C_1, **1000000.01**}	Fail
25.	T25 = {"USD","USD", C_1, **1000500**}	Fail	26.	T26 = {"**JPY**", "USD", C_1, **1000500**}	Fail	27.	T27 = {"**JPY**", "**JPY**", C_1, **1000500**}	Fail
28.	T28 = {"USD", "**JPY**", C_1, **1000500**}	Fail						

Les tests pour la méthode 2 (currencyConverter.Currency.convert(Double, Double))

29.	T29={**0**, 1.23}	Pass	30.	T30={**-0.01** , 0.15}	Fail	31.	T33 = {**5000**, 1.23}	Pass
32.	T31={**1 000 000**, 0.55}	Pass	33.	T32 = {**1 000 000. 01**, 1.00}	Fail	34.	T34 = {-**50**, 1.2}	Fail

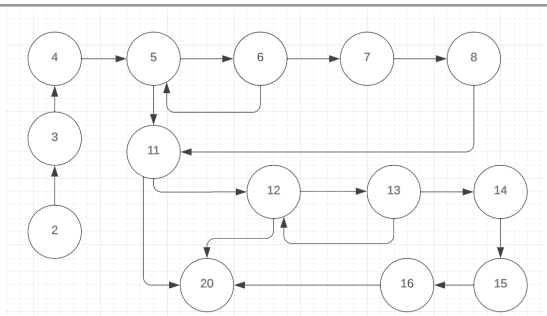
Test boit blanche

Assumons que la liste des currencies est fournie par le programme qui est toujours correct

currencies = {("US Dollar","USD"),("Euro", "EUR"),("British Pound", "GBP"), ("Swiss Franc", "CHF"), ("Chinese Yuan Renminbi", "CNY"), ("Japanese Yen", "JPY")}, la valeur d'exChangeValue est fournie par le programme et donc c'est toujours correct.

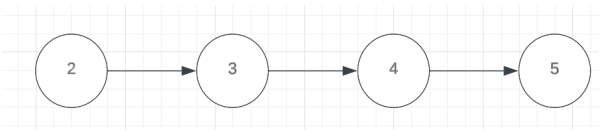
Critère de couverture des instructions:

```
1  public static Double convert(String currency1, String currency2,
   ArrayList<Currency> currencies, Double amount) {
2    String shortNameCurrency2 = null;
3    Double exchangeValue;
4    Double price = 0.0;
5    for (Integer i = 0; i < currencies.size(); i++) {
6      if (currencies.get(i).getName() == currency2) {
7        shortNameCurrency2 = currencies.get(i).getShortName();
8        break;
9      }
10   }
11   if (shortNameCurrency2 != null) {
12     for (Integer i = 0; i < currencies.size(); i++) {
13       if (currencies.get(i).getName() == currency1) {
14         exchangeValue = currencies.get(i).getExchangeValues().get
             (shortNameCurrency2);
15         price = Currency.convert(amount, exchangeValue);
16         break;
17       }
18     }
19   }
20   return price;
21 }
```




```
1 public static Double convert(Double amount, Double exchangeValue) {
2   Double price;
3   price = amount * exchangeValue;
4   price = Math.round(price * 100d) / 100d;

5   return price;
  }
```



Méthode 1:

- Instructions 5-12: D1={Avec l'ensemble défini currencies et amount,

  $\{currency \mid \exists c \in currencies, c.getName() = currency \ \wedge c.getShortName() \neq null\}$ }

- Instructions 13-20 : D2 = {Avec l'ensemble défini currencies et amount,

  $currency1 \in \{currency \mid \exists c \in currencies\ , c.getName() = currency \wedge c.getExchangeValues().contains(shortNameCurrency2)\}$}

- Les cas de test:

  T35 = ("**Euro**","**US Dollar**" currencies, 2000) => Valeur expectée: 2146.00
  T36 = ("**Canadian Dollar**", "Euro" currencies, 2000) => Valeur expectée: 0.0
  T37 = ("British Pound","**Canadian Dollar**", currencies, 2000) => Valeur expectée: 0.0

Méthode 2:

- Instructions 2-5: D3= {{(amount, exchangeValue) | amount $\in R$ et exchangeValue $\in R$ }}
- Les tests: On teste en même temps qu'en faisant les tests pour la méthode 1. Il y a seulement 1 chemin (2->3->4->5) dans la méthode 2. Cette méthode a une complexité V(G)=1 donc il n'y a pas d'exception ni de boucle. **Alors on n'a pas besoin de la tester individuellement (même chose pour les autres critères).**

Critère de couverture des arcs du graphe de flot de contrôle:

- Arc: 2 -> 3 -> 4-> 5-> 6-> 7 -> 8 -> 11 -> 12 -> 13 -> 14 -> 15 -> 16 -> 20

D4 = {(currency2,currency1) |($Avec\ l'ensemble\ défini\ currencies\ et\ amount,\ \exists c \in currencies,\ c.getName() = currency2 \wedge c.getShortName() \neq null$) ∧($\exists c1 \in currencies$, $c1.getName() = currency1$} => Cas de test: T35

- Arc: 2->3->4->(5->6)->11->20, dans(,) signifie qu'on exécute jusqu'à sortie la boucle

D5 = {(currency2)| ($\forall c \in currencies, c.getName() \neq currency2$)} => Cas de test : T37

- Arc: 2->3->4->5->11->(12->13)->20, dans(,) signifie qu'on execute la boucle jusqu'à la sortie

D6 = {(currency2, currency1)|($\exists c \in currencies, c.getName() = currency2 \wedge c.getShortName() \neq null$) ∧ ($\forall c1 \in currencies, c1.getName() \neq currency1$} => Cas de test : T36

Critère de couverture des chemins indépendants du graphe de flot de contrôle:

V(G) = 7 pour méthode 1

1. 2-3-4-5-11-20: même classe d'équivalence que D5 => Cas de test : T37
2. 2-3-4-5-6-7-8-11-20: impossible
3. 2-3-4-5-6-7-8-11-12-20: même classe d'équivalence que D6 => Cas de test : T36
4. 2-3-4-5-6-7-8-11-12-13-12-20:impossible
5. 2-3-4-5-6-7-8-11-12-13-14-15-16-20: même classe d'équivalence que D4 => Cas de test: T35
6. 2-3-4-5-11-12-20: impossible
7. 2-3-4-5-11-12-13-14-15-16-20: impossible

Critère de couverture des conditions:

Dans les deux méthodes, ce n'est pas pertinent pour évaluer ce critère, car il n'y a pas de condition composée.

Critère de couverture des i chemins:

La méthode 1 est le cas de boucle simple bornée par **n = 6** itérations équivalant au nombre d'éléments dans la liste **currencies** fournies par le programme. Vu qu'on a 2 boucles sur la liste currencies dépendamment à la valeur entrée de currencies 1 et 2, on va principalement tester sur la 1ère boucle et on suppose que l'autre est aussi correcte.

- On saute la boucle: liste de currencies est fixée et non vide, retourné par le programme donc le test pour une liste vide n'est pas pertinent.
- Une itération de la boucle: 1ère boucle exécutera 1 seule fois -> Cas de test : T35
- Deux itérations de la boucle: 1ère boucle exécutera 2 fois -> Cas de test: T38 = ("US Dollar","**Euro**", currencies, 2000) => Valeur expectée: 1860.00
- Quatre itérations de la boucle (m<n):1ère boucle exécutera 4 fois -> Cas de test: T39= ("US Dollar","**Swiss Franc**", currencies, 2000) => Valeur expectée: 2020.00
- Cinq itérations de la boucle (n-1):1ère boucle exécutera 5 fois -> Cas de test:T40 = ("US Dollar", "**Chinese Yuan Renminbi**", currencies, 2000) => Valeur expectée: 12720.00
- Six itérations de la boucle (n):1ère boucle exécutera 6 fois -> Cas de test:T41 = ("US Dollar", "**Japanese Yen**", currencies, 2000) => Valeur expectée: 247080.00

- Sept itérations de la boucle (n+1) : Vu que la liste currencies est fixée, donc dans ce cas on considère que le cas d'un currency inexistant dépasse la borne supérieure -> Cas de test: T36

Résultats

Décrivez les résultats de vos tests et donnez vos observations sur les différentes approches et critères de test

Boite noire

Tous les tests de la boîte noire ont terminés avec succès. Cela signifie que les tests qui devaient échoués n'ont pas échoués. En regardant le code, on voit que notre hypothèse sur les entrées non-valides n'était pas bonne, car une entrée non-valide retourne 0. De plus, notre programme accepte JPY et les nombres négatifs alors qu'il ne devrait pas selon les critères préétablis de la boîte noire. Finalement, cela nous fait remarquer que certains critère préétablis ne sont pas respectés. On sait comment la conversion de la deuxième méthode fonctionne et qu'elle accepte les nombres négatifs, mais pour la première, on ne teste pas la sortie, puisqu'on ne connaît pas le ratio de conversion qui est fait à l'intérieur du code. Tout ce qu'on sait, c'est qu'il n'y a pas eu d'exception.

Boite blanche

Tous les tests de la boîte blanche ont terminés avec succès. Dans certains cas, on doit fixer les données comme l'exchangeValue ou la liste de currencies, car c'est principalement retourné par le programme. On interprète ces conditions comme on a créé un environnement de test qui permet d'évaluer le fonctionnement du programme de manière contrôlée. Le succès de nos tests dans ces conditions indique que le programme est capable de gérer divers scénarios sans erreur. En utilisant divers critères de test boîte blanche, on a assuré que tous les besoins essentiels du programme ont été testés. Cela indique que le programme fonctionne correctement de manière fiable dans divers scénarios.

Observations supplémentaires

Bien que nos tests aient réussi dans un environnement contrôlé, il est nécessaire de considérer le comportement du programme avec des données qui sont plus dynamiques, et donc ce qui pourrait nécessiter des tests supplémentaires ou des méthodes mixtes en combinant les approches de boîte noire et de boîte blanche.