Rapport TP4

Tache 1 : TEST BOITE NOIRE

1. Classe MainWindow.convert(String, String, ArrayList<currencies>, Double)

a) Partition du domaine des entrées en classes d'équivalence

USD, CAD, GBP, EUR, CHF, AUD ensemble de devises convertissables.

Montant accepter [0 ; 1000000]

On définit 3 classes d'équivalences :

Soit D = entier et P défini sur [0 ; 1000000]

On définit 3 classes d'équivalences :

D1 : {0<=d<=1000000} valeur valide appartenant à l'intervalle de d

D2 : {d<0} valeur invalide inferieur à l'intervalle de d

D3 : {d > 1000000} valeur invalide supérieures à l'intervalle de d

Un jeu de tests valide on a : T = {100, -5, 3000000}

On doit avoir nos devises valides sur nos classes d'équivalence :

D = Devises = {USD, CAD, GBP, EUR, CHF, etc}

D1 : {USD, CAD, GBP, EUR, CHF, AUD} devises valides

D2 : {CNY, RUB, CFA} devise invalides

Un jeu de tests valide on a : T = {AUD, CNY}

b) Analyse des valeurs frontières

D1 : {0<=d<=1000000}

D2 : {d<0}

D3 : {d > 1000000}

On aura les valeurs T= {-10000, -1, 0, 500000, 1000000, 1000001, 1010000} : typique, : frontière.

À partir de nos jeux de tests et des classes d'équivalence on a les tests suivants :

2. Classe Currency.convert (Double, Double)

Cette classe prend un montant et un taux de change et fait la conversion.

c) Partition du domaine des entrées en classes d'équivalence

On définit 3 classes d'équivalences :

t = taux d'échange positif

D1 :{(d,t) | 0<=d<=1000000 et t = taux d'échange} valeur valide appartenant à l'intervalle de d

D2 : {(d,t) | d<0 et t = taux d'échange} valeur invalide inferieur à l'intervalle de d

D3 : {(d,t) | d > 1000000 et t = taux d'échange} valeur invalide supérieures à l'intervalle de d Un jeu de test valide serait :

T = {(6000,1.5), (-50, 0.5), (1000501, 2.2)}

d) Analyse des valeurs frontières

D1 : {0<=d<=1000000}

D2 : {d<0}

D3 : {d > 1000000}

On aura les valeurs T= {(-500,1.25), (-1, 1.5), (0, 1.7), (500000,1.5), (1000000,0.75), (1000001, 0.6), (1000500, 0.25)}

: typique, : frontière.

À partir de nos jeux de tests et des classes d'équivalence on a les tests suivants :

Tache 2 : TEST BOITE BLANCHE

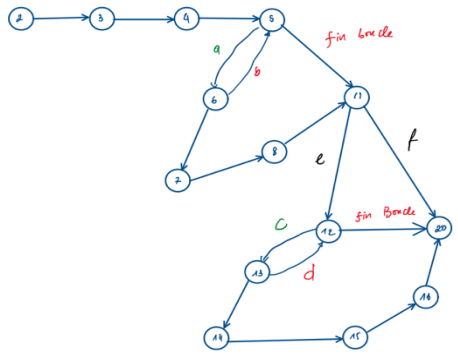
1. Classe MainWindow.convert

A) Couverture des instructions

La méthode Convert se situe de la ligne de code 173 à 197. Boucle for avec un if da la ligne 179 à la ligne 184. Pour pouvoir passe à travers toute les lignes de code, notre String 1 et String2 dans (String1, String2,'','', d) doivent être des devise valide. Donc un jeu test valide serait : T = {"EUR", "CHF", currencies,500}.

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

```
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  }
```



a : Currencies.get(i).getname == currency2 c : Currencies.get(i).getname == currency1 e : short name currencies != null

b = Currencies.get(i).getname != currency2 d = Currencies.get(i).getname != currency1 f : short name Currencies == null

Les Arc sont : 1) 2 à 5 – 6 - 7 - 8 -11 – 12 – 13 – 14 – 15 – 16 – 20 ; 2) 2 à 5 – 6 – 5 – 11 – 20 ; 3) 2 à 5 – 6 - 7 - 8 -11 – 12 – 13 – 12 – 20.

Les tests découlant sont : T = {("USD", "CHF", currencies, 500), ("USD", "QAR", currencies, 500), ("CFA", "GBP", currencies, 500)}

C) Critère de couverture des chemins indépendants.

V(G) = 6

Les Chemins sont : 1) 2 à 5 – 6 – 7 – 8 – 11 – 12 – 13 – 14 – 15 – 16 – 20 cas test : ("USD", "CHF", currencies, 1000).

2) 2 à 5 – 6 – 7 – 8 – 11 – 20 (impossible)

3) 2 à 5 – 6 – 5 – 11 – 20 ; cas de test : ("CHF", "RUB", currencies, 2000)

4) 2 à 5 – 6 – 7 – 8 – 11 – 12 – 13 – 12 – 20. Cas de test : ("QAR", "USD", currencies, 3000)

5) 2 à 5 – 6 – 11 – 12 – 13 – 12 – 20 (Impossible) 6) 2 à 5 – 6 – 5 – 11 – 12 – 13 – 12 – 20 (impossible)

D) Critère de couverture des conditions (et des arc) (pas applicable à notre méthode)

E) Critère de couverture des i-chemins

Ici on a deux boucles indépendantes concaténer de ce fait chaque boucle sera traité comme une boucle simple

Boucle 1 : On saute la boucle => pas de test

Une itération de boucle => (“GBP”, “USD”, currencies, 600)

Deux itérations de boucle => (“GBP”, “EUR”, currencies, 200)

m (m=3) itération (m<n) => (“EUR”,“GBP”, currencies, 4000)

1-n itération => (“GBP”, “CNY”, currencies, 3500)

n itération => (“USD”, “JPY”, currencies, 6300)

Boucle 2 : On saute la boucle => (“CHF”, “CFA”, currencies, 6000)

Une itération de boucle => (“USD”, “GBP”, currencies, 200)

Deux itérations de boucle => (“EUR”, “EUR”, currencies, 400)

m (m=4) itération (m<n) => (“CHF”,“EUR”, currencies, 400)

1-n itération => (“CNY”, “EUR”, currencies, 3000)

n itération => (“JPY”, “USD”, currencies, 6300)

2. Classe Currency.Convert

A) Couverture des instructions

Cette méthode de la classe n’ayant aucune boucle conditionnelle ou de condition aura toutes ses instructions exécutées qu’il arrive. D1:{(d,t) | 0<=d<=1000000 et t = taux d’échange} avec valeur de d valide un jeu de test serait T = {5000, 1.25}.

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

Vu qu’Il n’y a pas de manque de boucle et de condition dans la méthode on passera par chaque instruction à l’appel de la méthode.

```
1  public static Double convert(Double amount, Double exchangeValue) {
2      Double price;
3      price = amount * exchangeValue;
4      price = Math.round(price * 100d) / 100d;
5
6      return price;
7  }
```

Le seul arc : 2 – 3 – 4 – 6. T = (300, 1.50)

C) Critère de couverture des chemins indépendants.

V(g) = 1 et le chemin sera 2-3-4-6 et un jeu de test T = (200, 0.3)

D) Critère de couverture des conditions (et des arc) (pas applicable à notre méthode)

F) Critère de couverture des i-chemins (pas de boucle donc pas applicable)

TEST

1.Classe MainWindow.convert

Pour cette classe, nous avons décidé d’utiliser le nom des devises au lieu des short-names des devises afin que nos tests puissent être vérifiable. En effet, le code à tester est écrit de sort à ce qu’il compare le nom des devises au lieu de leur short-names. Sinon, avec l’utilisant les short-names, on aura juste des tests qui ne passent pas.

On a fait : l’hypothèse que tout résultat de valeur négatif ou qui ne respecte pas la spécification du code doit retourner 0. Donc on a comparé les cas qui ne doivent pas marche à 0 (ou 0.0). Pour le Critère de couverture des i-chemins, on a pris écrit un code modifié de MainWindow.convert pour chaque boucle afin de pouvoir compter les itérations. Pour la méthode testConvertValeurFrontier(), on va utiliser des devises fixes comme EUR et USD pour s’assurer que juste le montant est varié et de ce fait mieux tester. Enfin, on s’attend à ce que tous nos tests passent.

2.Classe Currency.convert

Pour cette classe, Vu que cela n'était pas spécifié, on a décidé:

Hypothèse 1: Une valeur d'échange ne peut pas être négative. Hypothèse 2 : tout résultat de valeur négatif ou qui ne respecte pas la spécification du code doit retourner 0.

Enfin, on s'attend à ce que tous nos tests passent.

ANALYSE DES TESTS

1.Classe MainWindow.convert

Teste de la boite noire

Pour la classe testConvertPourClasseDEquivalence() : les tests avec notre cas de $d < 0$ et $d > 1000000$ ne passent pas car le code fourni accepte et traite les cas out of Bound et de ce fait le code lui-même ne respecte pas les spécifications données et donc les tests ont échoué et nos hypothèses sont pas vérifiées. Aussi, on s'est rendu compte que les devises CAD et AUD qui ne sont pas pris en compte par le code et les devises JPY et CNY sont pris en compte par le code contrairement à ce que dit la spécification donc les tests avec ces devises-là n'ont pas passé.

Pour la classe testConvertValeurFrontier() : les tests sur les valeurs frontières hors intervalle n'ont pas passé car notre hypothèse de retour sur le résultat n'est pas vérifié vu que la spécification théorique du code n'est pas conforme à l'implémentation réel du code. Finalement, tous les autres tests passent.

Tests de la boite blanche

Tous nos tests passent.

2.Classe Currency.convert

Test Boite Blanche

Notre Hypothèse 1 n'est pas vérifié car dans la méthode testConvertPourClasseDEquivalence() le test sur le cas des valeurs d'échange négative ne passe pas. En effet, nous avons émis cette hypothèse car un taux d'échange ne peux pas conduit à un montant négatif donc nous avons pensé que le code était code pour refuser les valeurs d'échange négative.

Notre Hypothèse 2 n'est pas vérifiée car les tests, les valeurs hors limite dans les tests de cette même méthode et la méthode testConvertValeurFrontier() ont été traité par le code et n'ont pas retourné une valeur nulle (0). Cela signifie que le code ne vérifie pas les entrées pour voir s'ils sont conformes à la spécification avant de faire le calcul.

Test Boite Blanche

Tous nos tests passent.