Auteurs :Ithri Kendi (20222832) & Walid Boudehane (20206664)

IFT3913 – TP4 RAPPORT

1. TESTS BOITE NOIRE

Nous avons 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].

A partir de la documentation nous avons décidé que notre jeu de test valide serait comme suit :

- Pour la partition du domaine des entrées en classes d'équivalence : T = {-100, 100, 2000000}
- Pour l'analyse des valeurs frontières: T = {-1,0, 1000000, 1000001}
- Pour les devises T = {(USD,EUR) , (USD,CAD), (USD,GBP), (USD,CHF), (USD,AUD), (USD,XYZ), (XYZ,USD), (USD,EUR), (CAD,USDD), (CAD,)}

Pour la déclaration de la Arraylist<Currency> currencies qu'on doit utiliser pour la méthode MainWindow.convert, nous avons utilisé la methode Currency.init() qui est présente dans le code.

1.1 Currency.convert

Pour la méthode Currency.convert, comme les devises ne sont pas utilisées nous avons effectués uniquement sur le jeu de test sur les montants avec un exchange value fixe de 1.5 .
Pour les valeurs à l'intérieur de l'intervalle soit {0, 100, 1000000} les tests sont passés sans problèmes ce qui montre que la méthode fait les conversions correctement.
Mais pour les valeurs à l'extérieur de l'intervalle soit {-100, -1, 1000001} les tests ne passent pas car par rapport à la spécification la méthode la méthode devrait retourner 0 pour ces valeurs la mais la conversion se fait alors qu'elle ne devrait pas se faire. Notre hypothèse pour ce comportement serait sur le fait qu'il n'y a probablement pas de contraintes par rapport au domaine de la spécification.

1.2 MainWindow.convert

Pour la méthode MainWindow.convert, Nous avons utilisés tous les jeux de tests que nous avons montré au début en plusieurs combinaisons. Nous avons utilisé (USD, EUR) comme devises valides avec les différents montants (valides ou invalides) afin de tester en premier lieu les montants. De l'autre côté, nous avons utilisé le montant 100 comme montant valide avec les différentes devises (valides ou invalides) afin de tester les devises. Pour les tests avec les valeurs valides devrait faire la conversion sans problèmes et ceux avec les valeurs invalides devrait retourner une valeur de 0 lors de la conversions comme ces valeurs-là ne font pas partie de la spécification.
Lors de résultats des tests nous avons remarqué un comportement des tests non-attendu. Les tests avec les valeurs valides ne passaient pas sauf avec la valeur 0 mais les tests avec les valeurs invalides passaient sans problèmes. Après une investigation sur les résultats des tests, nous avons remarqué que tous les tests retournaient la valeur 0 peu importe la valeur de l'entrée qu'elle soit valide ou pas.
Après avoir investigué le code des méthodes nous faisons l'hypothèse que ce comportement est relié au format des devises, nous pensons que la méthode utilise les devises sous la forme longue (US Dollar) au lieu de la forme courte (USD) or que dans la spécification nous avons uniquement accès a la forme courte. Donc toutes les entrées de devises sont considérée invalides car elles ne sont pas dans le bon format.

2.TESTS BOITE BLANCHE :

Currency.convert :

Cette fonction ne contient aucune condition "if" ni de boucle, donc n'importe quel jeu de tests couvre toutes les instructions et les arcs. La fonction utilisée pour le test est : testBoiteBlanche0().

MainWindow.convert :

```
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;

                // Find shortname for the second currency
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              }

        // Find exchange value and call convert() to calcul the new price
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) Couverture des instructions :

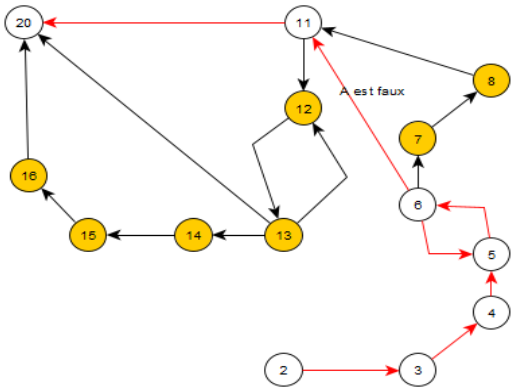
One peut couvrir les Instructions 2-20 :

D: {(currency1, currency2) | currecny1 ∈ currencies et currency2 ∈ currencies}

Et donc le jeu de test est : {(''Dollar Us'', ''Euro'')} et la fonction utilisé pour le test est : **testBoiteBlanche3()**

2) Couverture des arcs du graphe de flot de contrôle :

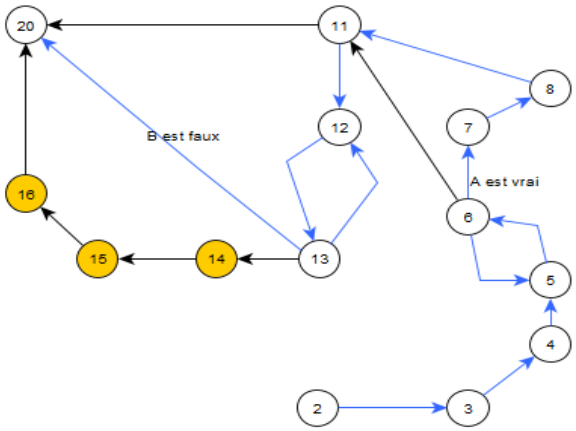
- Les somments blanches sont les sommets visitées .
- Les arcs colorés sont les arcs traversés .
- La condition A est : currency2 ∈ currencies.
- La condition B est : currecny1 ∈ currencies.
- L'ecriture juste est D={(currency1,currency2,currencies ,amount)} .



D1: {(currency1,currency2) | currency2 ∉ currencies}

Les arcs traversés: 2>3>4>5>6>11>20 , (6>5 aussi)

La fonction utilisée pour le test est :**testBoiteBlanche1()**

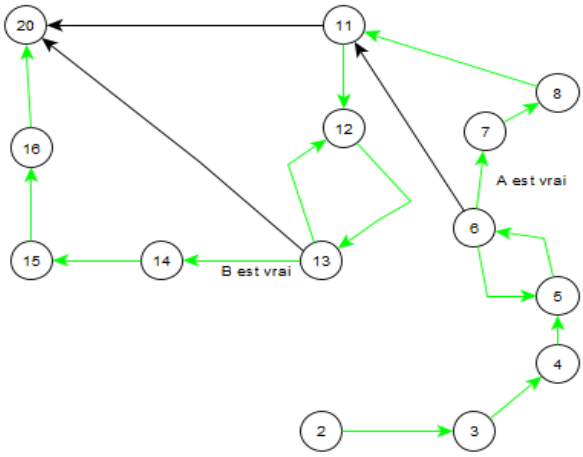


D2: {(currency1,currency2) | (currency2 ∈ currencies) ∧ (currency1 ∉ currencies)}

Les arcs traversés: 2>3>4>5>6>7>8>11>12>13>20 ,

(6>5 et 13>12 aussi)

La fonction utilisée pour le test est : **testBoiteBlanche2()**



D3: {(currency1,currency2) | (currency2 ∈ currencies) ∧ (currency1 ∈ currencies)}

Les arcs traversés:

2>3>4>5>6>7>8>11>12>13>14>15>16>20 , (6>5 et 13>12 aussi)

La fonction utilisée pour le test est : **testBoiteBlanche3()**

- Et donc notre jeu de Test est : {(''US Dollar'', ''euro''),(''us dollar'', ''Euro'') ,(''US Dollar'' , ''Euro'')}

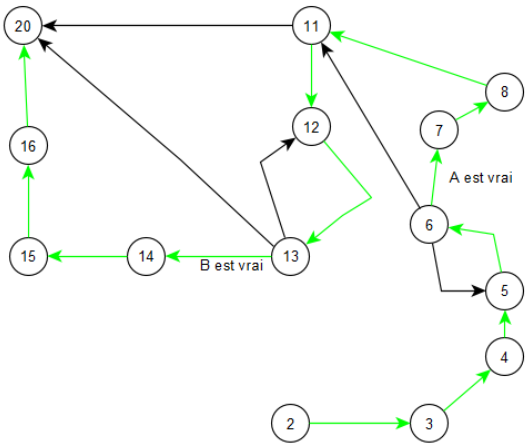
3) <u>Couverture des chemins indépendants du graphe de flot de contrôle :</u>

On remarque que le premier élément de currencies est toujours le ''US Dollar'' et donc pour exécuter les deux boucle une fois chacune, il faut que currency1=currency2= ''US Dollar'' .

```
public static ArrayList<Currency> init() {
        ArrayList<Currency> currencies = new ArrayList<Currency>();

        currencies.add( new Currency("US Dollar", "USD") );
        currencies.add( new Currency("Euro", "EUR") );
        currencies.add( new Currency("British Pound", "GBP") );
        currencies.add( new Currency("Swiss Franc", "CHF") );
        currencies.add( new Currency("Chinese Yuan Renminbi", "CNY") );
        currencies.add( new Currency("Japanese Yen", "JPY") );
        for (Integer i =0; i < currencies.size(); i++) {
                currencies.get(i).defaultValues();
        }

        return currencies;
    }
```



D: {(currency1,currency2) | (currency1 = currency2 = "US Dollar")}

Les arcs traversés: 2>3>4>5>6>7>8>11>12>13>20

La fonction utilisée pour le test est : **testBoiteBlanche4()**

- Le jeu de test est : (''US Dollar'', ''US Dollar'')

4) <u>Couverture des conditions :</u> ne s'applique pas.

5) <u>Couverture des i-chemins :</u>

<u>Saute de boucle :</u>

On peut sauter la deuxième si currency2 $\notin$ currencies.

D0={(currency1,currency2)|currency2 $\notin$ currencies)

Pour exécuter chaque boucle une seule fois :

D1={(currency1,currency2)|currency 1=currency2 = "US Dollar")

Pour exécuter chaque boucle 2 fois :

D2={(currency1,currency2)|currency 1=currency2 = "Euro")

Pour exécuter chaque boucle 3 fois :

D3={(currency1,currency2)|currency 1=currency2 = "British Pound")

Pour exécuter chaque boucle 4 fois :

D4={(currency1,currency2)|currency 1=currency2 = "Swiss Franc")

Pour exécuter chaque boucle 5 fois :

D5={(currency1,currency2)|currency 1=currency2 = "Chinese Yuan Renminbi")

Pour exécuter chaque boucle 5 fois :

D6={(currency1,currency2)|currency 1=currency2 = "Japanese Yen")

Le jeu de test: {(''Dinar'','' "US Dollar"), ("US Dollar", "US Dollar"),("Euro", "Euro"),("British Pound", "British Pound"),

("Swiss Franc"," Swiss Franc"), ("Chinese Yuan Renminbi","Chinese Yuan Renminbi''),("Japanese Yen"," Japanese Yen'') }

- La fonction utilisée pour le test est : **testBoiteBlanche5()**

Discussion des resultats :

Tous les tests de la boîte blanche ont passé avec succès, car, contrairement aux tests de la boîte noire, nous sommes au courant de l'implémentation des fonctions, et nous avons testé non seulement la couverture des instructions, mais surtout les conditions et les boucles bornées.