Jean-Loup Bergeron, 20134672
Antoine Gauthier-Drapeau, 20162765

Rapport TP4

1. Tests boite noire

En supposant que le code accepte seulement les devises USD, CAD, GBP, EUR, CHF, INR et AUD ainsi qu'uniquement les montants entre 0 et 10 000, nous en déduisons ces hypothèses.

Hypothèse 1 : La conversion d'un montant d'une devise x à une devise y suivit par la conversion du nouveau montant en devise y à la devise x devrait donner le montant initial.

Hypothèse 2 : La conversion d'un montant de devise x vers une devise non connu y (et vise-versa) devrait renvoyer une valeur Null.

Hypothèse 3 : Les paramètres « from » et « to » de la fonction « convert » devrait être limité à trois charactères. Dans le cas d'une entré plus grande ou plus petite que 3, la fonction devrait renvoyer un message d'erreur.

Hypothèse 4 : Un montant inférieur à 0 serait supposé de renvoyer un montant un message d'erreur.

Hypothèse 5 : Un montant supérieur à 10 000 devrait renvoyer un montant un message d'erreur.

Hypothèse 6 : Un montant égal à 0 ou égal à 10 000 devrait être accepté et convertit.

Hypothèse 7 : Un montant z > 10 000 convertit à partir d'un montant x <= 10 000 ne devrait pas être re-convertissable comme l'hypothèse 1 le décrit. Un message d'erreur devrait être retourné.

Hypothèse 8 : Un montant compris entre 0 et 10 000 devrait être accepté et convertit.

2. Tests boite blanche

La fonction « convert() » se défini comme suit :

$$convert(amout\ , from\ , to\ , conversion)$$

Et ce sont ces termes que nous utiliserons pour la suite.

Nous utiliserons aussi la fonction « containsKey() » utilisé comme suit:

$$conversion.getRates.containsKey(value)$$

A. <u>Critère de couverture des instructions :</u>
Ici on veut sélectionner un jeu de test qui nous permet d'exécuter chaque instruction au moins une fois. Pour constituer les jeux de test, on regroupe dans des classes Di les éléments du domaine d'entrée D qui activent les mêmes instructions.

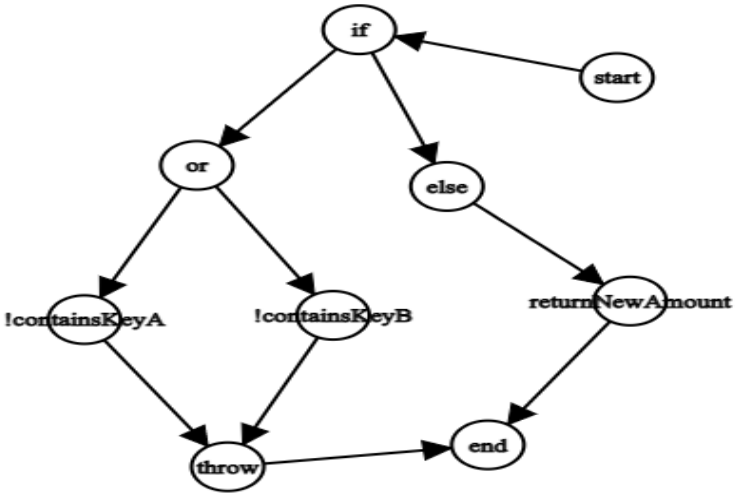
$$D1:\{(from,to)|\ \begin{cases} conversion.getRates.containsKey(from) \\ or \\ conversion.getRates.containsKey(to) \end{cases}\}$$

$$D2:\{(from,to)|\ \begin{cases} !\,conversion.getRates.containsKey(from) \\ or \\ !\,conversion.getRates.containsKey(to) \end{cases}\}$$

Jeux de test :

$$D1 = \{\ (from,to)|\ from = USD, to = CAD\}$$
$$D2 = \{\ (from,to)|\ from = CAD, to = USDD\}$$

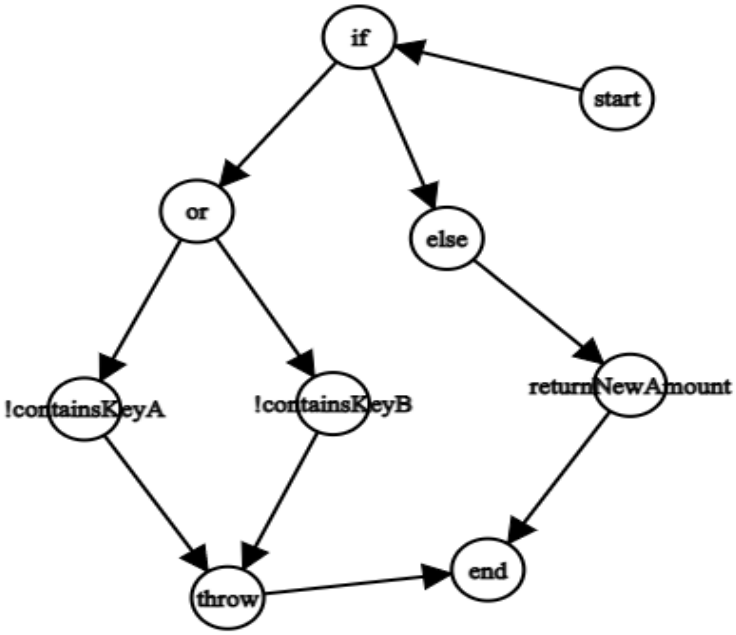
B. <u>Critère de couverture des arcs du graphe de flot de contrôle :</u>



Ici, les mêmes cas qu'en A nous permette de tester pour chaque instructions conditionnelles le cas où elle est vraie et le cas où elle est fausse.
Jeux de test :

$$D1 = \{\ (from,to)|\ from = USD, to = CAD\}$$
$$D2 = \{\ (from,to)|\ from = CAD, to = USDD\}$$

C. Critère de couverture des chemins indépendants du graphe de flot de control :



Complexité cyclomatique V(G) = 3

Les chemins indépendants sont :

Chemin 1: Start -> if -> or -> !containsKeyA -> throw -> end

Chemin 2: Start -> if -> or -> !containsKeyB -> throw -> end

Chemin 3: Start -> if -> else -> returnNewAmount -> end

Les jeux de test sont :

$$Chemin\ 1 = \{\ (from, to) |\ from = CA, to = USD\}$$
$$Chemin\ 2 = \{\ (from, to) |\ from = CAD, to = SD\}$$
$$Chemin\ 3 = \{\ (from, to) |\ from = USD, to = CAD\}$$

D. Critère de couverture des conditions :

Il ne fait pas de sens de tester ce critère car la fonction « convert() » ne contient pas de condition composée.

E. Critère de couverture des i-chemins :

Il ne fait pas de sens de tester ce critère car la fonction « convert() » ne contient pas de boucle.

3. Explication et résultats

Partie 1 : Pour la partie un nous nous somme uniquement basé sur les suppositions faites dans l'énoncé et sur ce que la fonction « convert() » était censé faire. À partir de là, nous avons essayé d'envisager tous les cas possibles d'entré et d'en définir la sortie la plus logique. Pour chaque hypothèse fait nous avons ces tests :

Hypothèse 1 : TestCurrencyConvertor.conversionOfConversio()
Hypothèse 2 : TestCurrencyConvertor.convertUnknownDevise1()
TestCurrencyConvertor.convertUnknownDevise2()
Hypothèse 3 : TestCurrencyConvertor.deviseSizeToBig()
TestCurrencyConvertor.deviseSizeToBig2()
TestCurrencyConvertor.deviseSizeToSmall()
TestCurrencyConvertor.deviseSizeToSmall2()
Hypothèse 4 : TestCurrencyConvertor.montantToSmall()
Hypothèse 5 : TestCurrencyConvertor.montantToBig()
Hypothèse 6 : TestCurrencyConvertor.montantLimit()
TestCurrencyConvertor.montantUpperLimit()
Hypothèse 7 : TestCurrencyConvertor.gotTobig()
Hypothèse 8 : TestCurrencyConvertor.inRange()

Nous obtenons les résultats suivants :

```
Failed tests:
  TestCurrencyConvertor.convertUnknownDevise1 Expected exception: java.text.ParseException
  TestCurrencyConvertor.convertUnknownDevise2 Expected exception: java.text.ParseException
  TestCurrencyConvertor.gotToBig Expected exception: java.text.ParseException
  TestCurrencyConvertor.montantToBig Expected exception: java.text.ParseException
  TestCurrencyConvertor.montantToSmall Expected exception: java.text.ParseException
```

En observant nos résultats il n'est pas clair de pourquoi les tests ont échoués. Est-ce parce que les valeurs testées étaient acceptables finalement ou est-ce parce que nous n'avons pas anticipé la bonne forme d'exception. Pour en être sur nous avons ajouté quelque test pour chaque test échoué :

Pour TestCurrencyConvertor.convertUnknownDevise1() nous avons ajouté :
TestCurrencyConvertor.convertUnknownDevise11()

Pour TestCurrencyConvertor.convertUnknownDevise2() nous avons ajouté :
TestCurrencyConvertor.convertUnknownDevise22()

Pour TestCurrencyConvertor.gotToBig() nous avons ajouté :
TestCurrencyConvertor. gotToBig 2()

Pour TestCurrencyConvertor.montantToBig() nous avons ajouté :
TestCurrencyConvertor.montantToBig2()

Pour TestCurrencyConvertor.montantToSmall () nous avons ajouté :
TestCurrencyConvertor.montantToSmall2()

Suite à l'ajout de ces tests qui ont tout pour but de vérifier si la fonction « convert() » revoit une valeur, nous obtenons exactement les mêmes résultats. Donc, chacun des nouveaux tests ont passé. Cela implique que nos hypothèses 2, 4, 5 et 7 ne sont pas bonne et que la fonction « convert() » réussi à calculer malgré les spécifications de l'énoncé.

Partie 2 : Pour cette section, nous nous sommes rendu compte que nos critères de couvertures sont testables avec les mêmes tests que la partie 1.

Pour le critère de couverture des instructions nous avons les jeux de test D1 et D2 respectivement testable avec « inRange() » et « deviseSizeTobig() ».

Pour le critère de couverture des arcs du graphe de flot de contrôle les jeux de test sont les mêmes que pour le critère de couverture des instructions. Donc, les tests de ceux-ci nous conviennent aussi.

Pour le critère de couverture des chemins indépendants du graphe de flot de control nous les jeux de test Chemin1, Chemin2 et Chemin3. Ou Chemin1 est testable avec « deviseSizeToSmall2() », Chemin2 avec « deviseSizeToSmall() » et Chemin trois avec « inRange() ».

Suite aux tests de la section 1, nous savons qu'aucun de ces tests ont échoués. Du fait même, il est correct de dire que nos tests de boite blanche passent et que la structure de la fonction « convert() » est bien définie.