**IFT3913 QUALITÉ DE LOGICIEL ET MÉTRIQUES – AUTOMNE 2019 – TRAVAIL PRATIQUE 3**

**0.introduction**

Le bût du travail est de créer des tests pour deux méthodes du « Currency Converter » :

• currencyConverter.MainWindow.convert(String, String, ArrayList<Currency>, Double)

• currencyConverter.Currency.convert(Double, Double)

**1.TESTS BOITE NOIRE**

On teste les deux méthodes en utilisant les deux de teste boite noir vue.

On connait deux spécifications du « Currency Converter » qui sont :

Il doit convertir des montants entre les devises suivantes : USD, CAD, GBP, EUR, CHF, CNY.

Il doit seulement accepter des montants positives.

Pour la méthode **currencyConverter.MainWindow.convert(String, String, ArrayList<Currency>, Double),** on suppose que on ne veut pas teste la ArrayList<Currency>.

* 1. **Approches de partition du domaine des entrées en classes d’équivalence**

Le domaine des valeurs **String** est D = P définie sur {USD, CAD, GBP, EUR, CHF, CNY} selon la spécification mais les teste nous fait remarque que **CAD** n’est pas dans le domaine.

On définit 2 classes d’équivalence pour les valeurs **String**

-Une classe d’équivalence d’entre valide D1 = {USD, GBP, EUR}

-Une classe d’équivalence d’entre invalide D2 = {FCT, GHC,CAD}

**Le jeu de test valide serait T** = -Valide (USD,USD,Currency,5), (EUR,USD,Currency,5)

-Invalid (CAD,USD,Currency,5), (EUR,CAD,Currency,5)

Le domaine des valeurs **Double** est D = P définie sur [0, ∞]

On définit 2 classes d’équivalence pour les valeurs **Double**

-Une classe d’équivalence pour P valide D1 = {0 ≤ P ≤ ∞}

-Une classe d’équivalence pour P invalide D2 = {P < 0}

**Le jeu de test valide serait T** = -Valide (80,452)

-Invalide (-55,-4240), (,USD,USD,Currency,-5)

On utilise les jeux de test obtenu pour les 2 méthode en fonction des valeurs d’entre de la méthode.

* 1. **Analyse des valeurs frontières.**

On définit 2 classes d’équivalence pour les valeurs **Double**

-Une classe d’équivalence pour P valide D1 = {0 ≤ P ≤ ∞}

-Une classe d’équivalence pour P invalide D2 = {P < 0}

-On ne peut pas étudier la frontière supérieure.

**Le jeu de test valide serait T** = (0,5), (-1,1000)

Pour les valeurs **String** on le domaine n’étant pas un intervalle, il n’y a donc pas de frontière a analysé.

* 1. **Conclusion**

Les teste effectue dans cette partie en boite noir nous a permis de remarquer que contrairement aux spécifications, les méthodes acceptent des montants positives et que la devise CAD n’est pas incluse dans le code.

**2.TESTS BOITE BLANC**

**2.1. Critère de couverture des instructions**

On cherche des jeux de test qui permettent de couvrir toutes les instructions du programme pour se faire on va les regroupe en classe.

**La methods MainWindow.convert(String, String, ArrayList<Currency>, Double)**

D1= {(x,y) |currencies.get(i).getName() = x et currencies.get(i).getName() = y}

D2= {(x,y) |currencies.get(i).getName() ≠ x | currencies.get(i).getName() = Y }

D3= {(x,y) | currencies.get(i).getName() = x | currencies.get(i).getName() ≠y }

D4= {(x,y) | currencies.get(i).getName() ≠ x | currencies.get(i).getName() ≠ y }

**Le jeu de test T** = {(EUR,USD, currencies,5), (USD,KL, currencies,5), (KL,USD, currencies,5), (GHB,KL, currencies,5)}

**La méthode Currency.convert(Double, Double)**

Pour cette méthode toutes les instructions sont exécutées à chaque exécution de la méthode donc elle n’a pas besoin de teste de couverture

**2.2. Critère de couverture des arcs du graphe de flot de contrôle**

On recherche ici les branchements de contrôle conditionnels dans un programme.

**La méthode MainWindow.convert(String, String, ArrayList<Currency>, Double)**

1

2

3

4

5

6

7

8

9

10

11

12

13

14

On obtient les chaines :

**-1,2,3,4,8,14**

(CAD,KL,Currency,5)

**-1,2,3,4,5,6,7,8,9,14**

(USD,CAD, Currency,5)

**-1,2,3,4,8,9,10,11,12,13,14**

(EUR,USD, Currency,5)

**La méthode Currency.convert(Double, Double)**

1

2

On obtient les chaines :

**-1,2** = (451,5000)

**2.3. Critère couverture des chemins indépendants du graphe de flot de contrôle**

On va utiliser de nouveau les graphes obtenus au point 2.2

**La méthode MainWindow.convert(String, String, ArrayList<Currency>, Double)**

Pour cette fonction la complexité est : P + 1 = 5 + 1 = 6 donc on s’attend à avoir 6 vecteurs

Les vecteurs sont positionnés comme suit :

1) 1-2

2)2-3

3)3-4

4)4-8

5)4-5

6)5-4

7)5-6

8)6-7

9)7-8

10)8-14

11)8-9

12)9-14

13)9-10

14)10-9

15)10-11

16)11-12

17)12-13

18)13-14

Les Vecteurs

(111100000100000000) = impossible

(111111000100000000) = (RT,DF, Currency,5)

(111010111100000000) = impossible

(111010111011000000) =impossible

(111010111011100000) = (RT,CAD, Currency,5)

(111010111010111111) = (USD,CAD, Currency,5)

**La méthode Currency.convert(Double, Double)**

Pour cette fonction la complexité est : P + 1 = 0+ 1 = 1 donc on s’attend à avoir 6 vecteurs

Les vecteurs sont positionnés comme suit :

1)1-2

Les Vecteurs

(11) = (1220,963)

**2.4. Critère couverture des conditions**

On voit que la couverture des conditions couvre la même partie que la couverture des arcs du graphe et on utilise les mêmes cas de test pour arriver à couvrir les deux

**2.5. Critère de couverture des i-chemins**

On sait que l’ordre de la ArrayList<Currency> est USD, CAD, GBP, EUR, CHF, CNY.

La méthode MainWindow.convert(String, String, ArrayList<Currency>, Double), on va utiliser la position de nom de devise dans le tableau pour avoir le nombre d’itération voulu.

**On a dans la méthode 2 boucle for qui parcoure la ArrayList<Currency>**

Trouver un jeu de test permettant de couvrir chacun des cas suivants :

-une itération des boucles : (USD,USD,Currency,5)

-deux itérations : (EUR, EUR,Currency,5)

-m itérations  : (CNY, CNY,Currency,5)

**La méthode Currency.convert(Double, Double)**

Cette fonction ne possède pas de boucle donc le teste de couverture des i-chemins ne pas possible pour cette méthode