Rapport TP4

IFT-3919
Qualité du logiciel et métriques
A2022

Jenny Diep (20036864)
Charbel Machaalani (20204556)

Université de Montréal

Tâche 1: TESTS BOITE NOIRE

La méthode convert ne satisfait pas la spécification suivante:

- Convertir des montants uniquement entre les devises suivantes : USD, CAD, GBP, EUR, CHF, INR, AUD.
- Seulement accepter des montants entre [0, 10000].

En effet on peut observer cela dans le code.

```
public final class CurrencyConvertor {

    public static double convert(double amount, String from, String to, CurrencyConversion conversion
        if(!conversion.getRates().containsKey(to)||!conversion.getRates().containsKey(from))
            throw new ParseException("Not correct format currency"
                    + "", errorOffset: 0);
        double curencyTo = conversion.getRates().get(to);
        double curencyFrom = conversion.getRates().get(from);
        return amount*(curencyTo/curencyFrom);

    }
}
```

On observe que la seule exception posé dans cette méthode est celle du format de la devise - il y a lors une faute est une défaillance.. Nous allons créer 2 tests:

Test #1 - testAmount():

Partition du domaine des entrées en classes d'équivalence

$D_1 = \{0 \leq d \leq 10000\}$

$D_2 = \{d < 0\}$

$D_3 = \{d > 10000\}$

Jeu de test valide: T = {1000, -1, 20000}

Test #2 - testCurrency():

Partition du domaine des entrées en classes d'équivalence

Nous allons utiliser que des strings.

D = { toute les devises existantes }

D_1 = { "USD", "CAD", "GBP", "EUR", "CHF", "INR", "AUD" }

D_2 = { "JPY", "RUB", "ARS", "BBD", "BRL", "KHR", "KYD" }

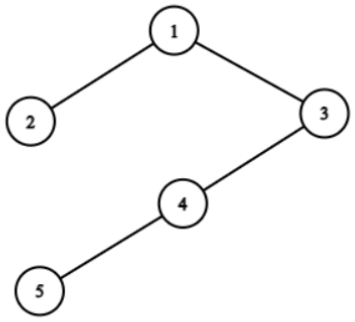
Jeu de test valide T = { "CAD", "KYD" }

En observant les résultats des tests, on observe des erreurs lorsque qu'on test pour -1, 20000 et "KYD" puisque ce sont des entrées invalides que nous avons choisi dans les domaine des entrées en classe d'équivalence. Donc, nous avons ajouter des try-catch pour ses cas.

Tâche 2: TESTS BOITE BLANC

Selon le critère de couverture des instructions, nous voulons un jeu de test qui permet de couvrir l'entièreté du code. C'est pour cela que nous prenons un jeu de test avec qui prend en considération l'entrée dans le if pour l'exécution du code if ainsi qu'un autre test qui évite d'entrer dans le if pour exécuter le reste du code.

Avec les mêmes tests, nous remplissons le critère de couverture des arcs du graph de contrôle. En effet nous voyons ici le graph de flux de contrôle:



Nous couvrons tous les arcs possibles en entrant vers la droite après la racine lorsque la mauvaise currency est entrée et lorsqu'on va vers la gauche quand la bonne currency est entrée. Nous constatons que ce jeu de tests permet aussi la couverture des chemins indépendants du graph de contrôle car nous allons vérifier l'exécution de chaque instruction pour un chemin sans tenir compte des autres chemins possibles. Nous allons d'un côté de la condition if, puis dans l'autre cas nous ignorons le if.

Nous avons pu tester le critère de couverture des conditions en faisant des tests qui prennent en considération les deux tests de la condition if. Tout d'abord, nous essayons d'entrer dans le if en ayant la variable from comme étant le currency erroné. Ensuite, nous entrons dans le if en ayant la variable to comme étant le currency erroné.

Conclusion

Enfin, nous avons déterminé qu'il y a une défaillance quand le montant est plus petit que 0. En effet, le résultat attendu est un nombre positif puisqu'on parle de devise dans notre cas. On peut alors dire que la faute est le manque de condition pour les nombres attendus dans la spécification.

Lors de ce TP, nous avons réaliser que les tests boîtes blanches sont simples puisqu'il n'y a pas de while loop. Ces tests ne sont pas trop conclusifs.