Rapport TP4

Rosine Rolande Simo Tegninko, 20183729
Yu Deng, 20151659

TESTS BOITE NOIRE

Dans ce test, il y a deux conditions de test requises :

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

Pour le type de devise

· **Classes d'équivalence**

Dans le code, il existe un moyen de construire des objets de type CurrencyConversion. Cela générera des objets contenant plusieurs devises, qui peuvent répondre à nos exigences de test. De plus, j'ai choisi de réutiliser le même code pour plus de commodité. La structure de base du test est la suivante. Déterminez si chaque cas de test réussit en jugeant le résultat de l'exécution de la couverture ou de l'exception levée.

```
// black box tests
@Test
void testConvert1(){
    OfflineJsonWorker  manager = new OfflineJsonWorker();
    CurrencyConversion conversion = manager.parser();
    try{
        double res = CurrencyConvertor.convert( amount: 1, from: "USD" , to: "CAD" ,conversion);
        assertTrue( condition: res>0);
    } catch (ParseException e){
        ParseException expected = new ParseException("Not correct format currency"
                + "", 0);
        fail("unsupported currency");
    }
}
```

Dans le cas de test 1, j'ai sélectionné USD et CAD pour la conversion. Les deux sont des types de devises qui peuvent être convertis. Cette affaire peut passer sans problème. Ensuite, j'autoriserai la conversion de sept devises pour passer le test par paires.

```
@Test
void testConvert2(){
    OfflineJsonWorker  manager = new OfflineJsonWorker();
    CurrencyConversion conversion = manager.parser();
    try{
        double res = CurrencyConvertor.convert( amount: 1, from: "GBP" , to: "EUR" ,conversion);
        assertTrue( condition: res>0);
    } catch (ParseException e){
        ParseException expected = new ParseException("Not correct format currency"
                + "", 0);
        fail("unsupported currency");
    }
}
```

Lorsque j'utilise AUD et FJD qui ne devraient pas être pris en charge pour la conversion, je trouve que l'assertion de résultat renvoie toujours le résultat normalement et j'utilise fail() pour générer des informations d'erreur, ce qui indique que convert() lui-même ne contraint pas le type de devise nous exigeons.

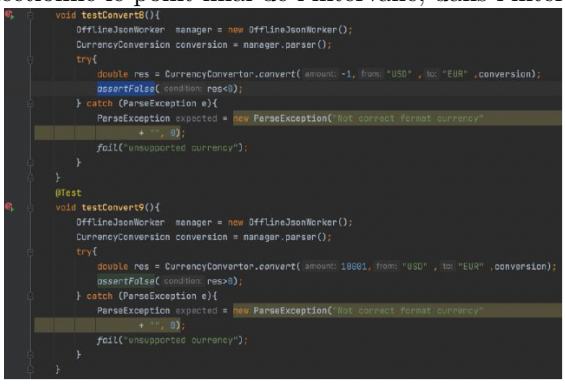
```
@Test
void testConvert4(){
    OfflineJsonWorker  manager = new OfflineJsonWorker();
    CurrencyConversion conversion = manager.parser();
    try{
        double res = CurrencyConvertor.convert( amount: 1, from: "FJD" , to: "AUD" ,conversion);
        fail("unexpected supported currency");
    } catch (ParseException e){
        ParseException expected = new ParseException("Not correct format currency"
                + "", 0);
        assertEquals(e.getMessage(),expected.getMessage());
    }
}
```

Pour plage numérique

· **D'analyse des valeurs frontières**

Pour le test d'intervalle numérique, j'ai sélectionné le point final de l'intervalle, dans l'intervalle et hors de l'intervalle.

```
void testConvert6(){
    OfflineJsonWorker  manager = new OfflineJsonWorker();
    CurrencyConversion conversion = manager.parser();
    try{
        double res = CurrencyConvertor.convert( amount: 0, from: "USD" , to: "EUR" ,conversion);
        assertTrue( condition: res==0);
    } catch (ParseException e){
        ParseException expected = new ParseException("Not correct format currency"
                + "", 0);
        fail("unsupported currency");
    }
}
@Test
void testConvert7(){
    OfflineJsonWorker  manager = new OfflineJsonWorker();
    CurrencyConversion conversion = manager.parser();
    try{
        double res = CurrencyConvertor.convert( amount: 10000, from: "USD" , to: "EUR" ,conversi
        assertTrue( condition: res>0);
    } catch (ParseException e){
        ParseException expected = new ParseException("Not correct format currency"
                + "", 0);
```



```
void testConvert8(){
    OfflineJsonWorker  manager = new OfflineJsonWorker();
    CurrencyConversion conversion = manager.parser();
    try{
        double res = CurrencyConvertor.convert( amount: -1, from: "USD" , to: "EUR" ,conversion);
        assertFalse( condition: res<0);
    } catch (ParseException e){
        ParseException expected = new ParseException("Not correct format currency"
                + "", 0);
        fail("unsupported currency");
    }
}
@Test
void testConvert9(){
    OfflineJsonWorker  manager = new OfflineJsonWorker();
    CurrencyConversion conversion = manager.parser();
    try{
        double res = CurrencyConvertor.convert( amount: 10001, from: "USD" , to: "EUR" ,conversion);
        assertFalse( condition: res>0);
    } catch (ParseException e){
        ParseException expected = new ParseException("Not correct format currency"
                + "", 0);
        fail("unsupported currency");
    }
}
```

Les valeurs dans l'intervalle et à la fin de l'intervalle ont passé avec succès la vérification des résultats, mais lorsque je suis allé à -1 et 10001, j'ai toujours renvoyé res normalement. En fait, ce n'est pas autorisé. Par conséquent, assertFalse échoue ici. De toute évidence, la fonction de conversion n'impose pas la limite numérique requise sur la valeur "amount".