

RAPPORT DE TESTS LOGICIELS-TP1

Réalisé par :

- Ahmed Mahfoudhi
- Mazen Issaoui
- Fares Garrach
- Hamza Sdiri

Le premier test d'une classe

1-la classe SommeArgent



```
1 package tp_test1.junit.monprojet;
2
3 public class SommeArgent {
4     private int quantite;
5     private String unite;
6     public SommeArgent(int amount, String currency) {
7         quantite = amount;
8         unite = currency;
9     }
10    public int getQuantite() {
11        return quantite;
12    }
13    public String getUnite() {
14        return unite;
15    }
16    public SommeArgent add(SommeArgent m) {
17        return new SommeArgent(getQuantite()+m.getQuantite(), getUnite());
18    }
19 }
```

Outline:

- tp_test1.junit.monprojet
 - SommeArgent
 - quantite : int
 - unite : String
 - SommeArgent(int, String)
 - getQuantite() : int
 - getUnite() : String
 - add(SommeArgent) : SommeArgent
 - equals(Object) : boolean

2 - la méthode equals

```
20 @Override
21 public boolean equals(Object obj) {
22     if (this == obj)
23         return true;
24     if (obj == null)
25         return false;
26     if (getClass() != obj.getClass())
27         return false;
28     SommeArgent other = (SommeArgent) obj;
29     if (quantite != other.quantite)
30         return false;
31     if (unite == null) {
32         if (other.unite != null)
33             return false;
34     } else if (!unite.equals(other.unite))
35         return false;
36     return true;
37 }
```

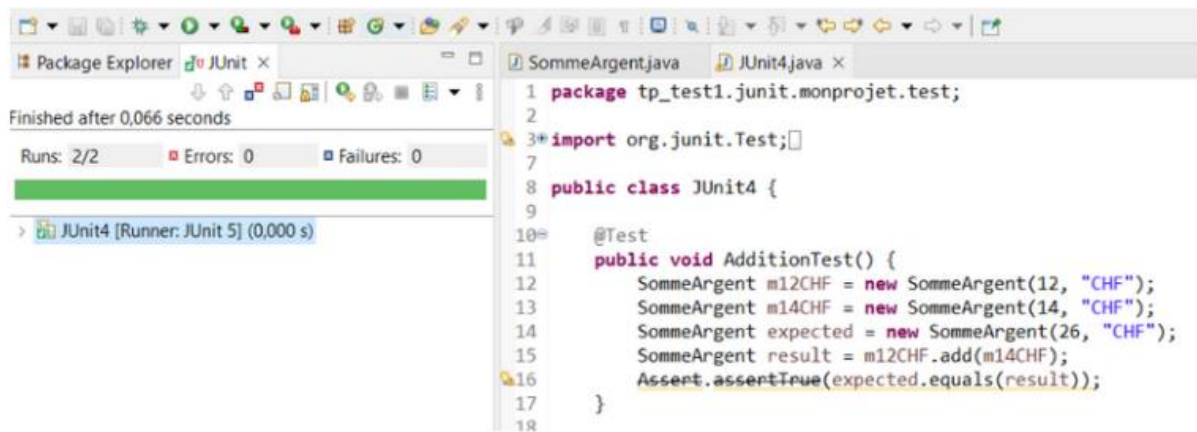
3 - classe de tests JUnit 4

```
1 package tp_test1.junit.monprojet.test;
2
3 import org.junit.Test;
4
5
6
7
8 public class JUnit4 {
9
10
11 }
12
```

4 –

```
1 package tp_test1.junit.monprojet.test;
2
3 import org.junit.Test;
4
5 public class JUnit4 {
6
7     @Test
8     public void AdditionTest() {
9         SommeArgent m12CHF = new SommeArgent(12, "CHF");
10        SommeArgent m14CHF = new SommeArgent(14, "CHF");
11        SommeArgent expected = new SommeArgent(26, "CHF");
12        SommeArgent result = m12CHF.add(m14CHF);
13        Assert.assertTrue(expected.equals(result));
14    }
15}
```

5 –



D'autres tests pour la classe SommeArgent

```
@Test
public void EqualsTest() {
    SommeArgent m12CHF = new SommeArgent(12, "CHF");
    SommeArgent m14CHF = new SommeArgent(14, "CHF");
    SommeArgent m14USD = new SommeArgent(14, "USD");

    // Test null comparison
    Assert.assertFalse(m12CHF.equals(null));

    // Test reflexive property of equals
    Assert.assertTrue(m12CHF.equals(m12CHF));

    // Test symmetric property of equals
    Assert.assertTrue(m12CHF.equals(new SommeArgent(12, "CHF")));
    Assert.assertTrue(new SommeArgent(12, "CHF").equals(m12CHF));

    // Test transitive property of equals
    SommeArgent m12CHF2 = new SommeArgent(12, "CHF");
    Assert.assertTrue(m12CHF.equals(m12CHF2));
    Assert.assertTrue(m12CHF2.equals(new SommeArgent(12, "CHF")));
    Assert.assertTrue(m12CHF.equals(new SommeArgent(12, "CHF")));

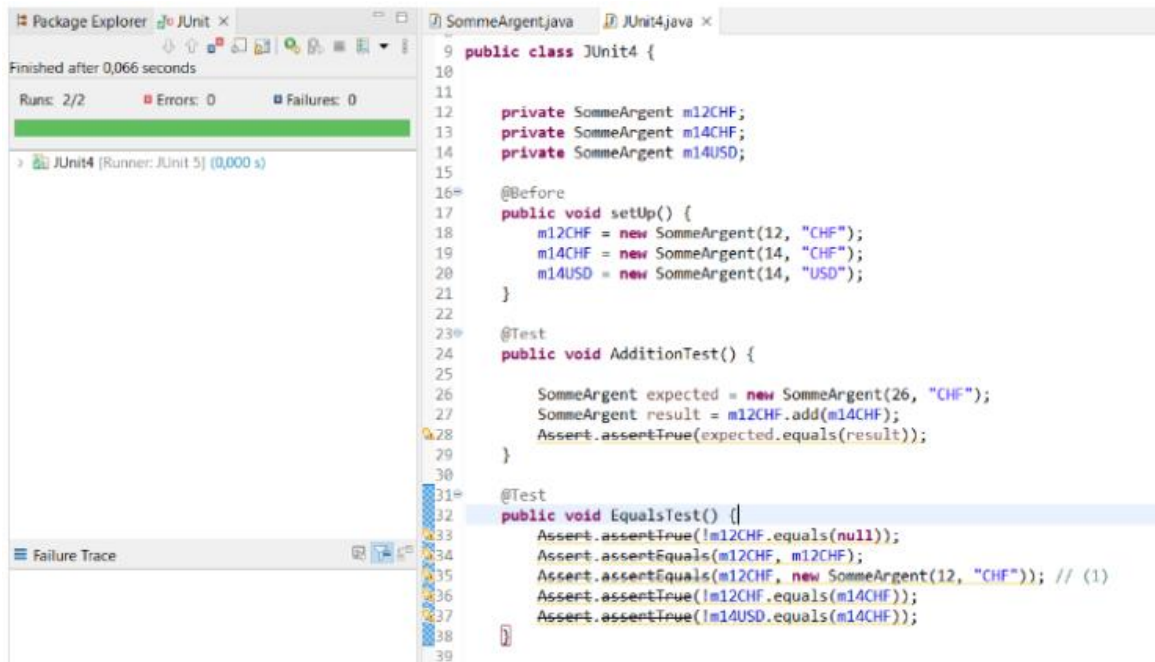
    // Test unequal amounts
    Assert.assertFalse(m12CHF.equals(m14CHF));

    // Test unequal currencies
    Assert.assertFalse(m14USD.equals(m14CHF));
}
```

6 –

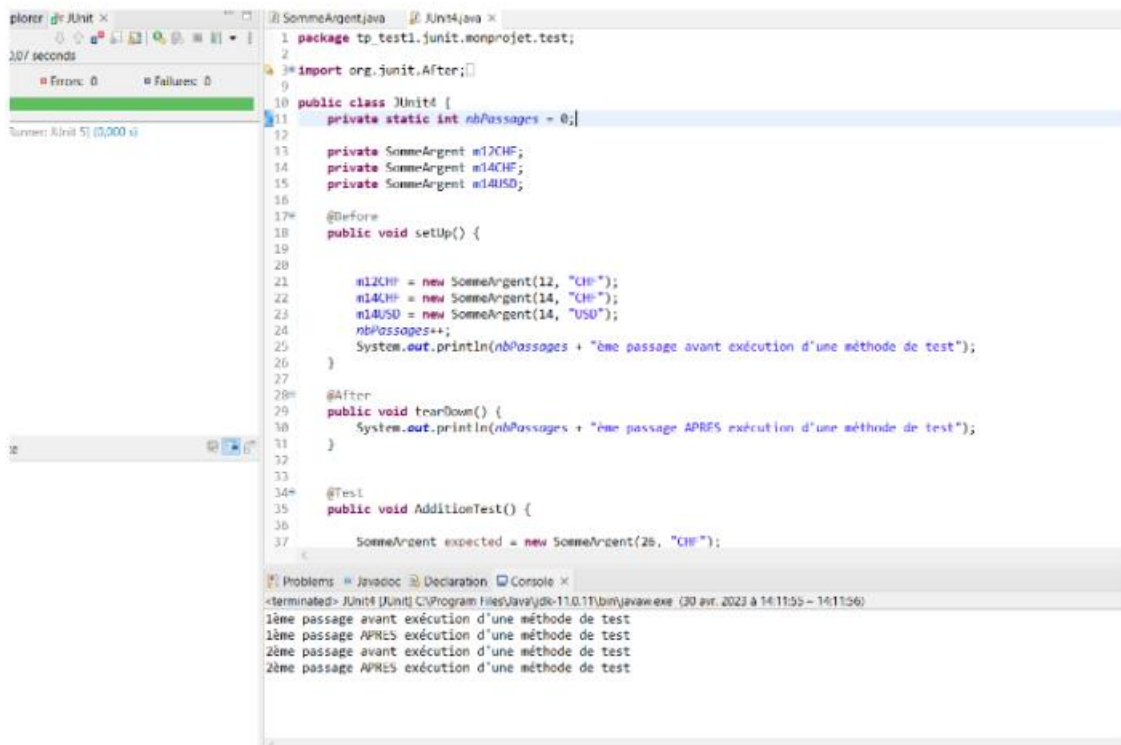
On teste si deux sommes d'argent avec des unités différentes ("CHF" et "USD") sont égales. Cette assertion doit retourner false car elles ont des unités différentes, même si leurs quantités sont les mêmes

7 –



8 – Il s'agit d'un ensemble d'objets qui sont créés et initialisés avant l'exécution d'une méthode de test, afin de s'assurer que l'environnement de test est stable et cohérent

9 –



```
package tp_test1.junit.monprojet;

public class UniteDistincteException extends Exception {
    private SommeArgent somme1, somme2;
    public UniteDistincteException(SommeArgent sa1, SommeArgent sa2) {
        somme1 = sa1;
        somme2 = sa2;
    }
    public String toString() {
        return "unité distincte : " + somme1.getUnite() + " != " +
            somme2.getUnite();
    }
}

public SommeArgent add(SommeArgent m) throws UniteDistincteException {
    if (!m.getUnite().equals(this.getUnite())) {
        throw new UniteDistincteException(this, m);
    }
    else return new SommeArgent(getQuantite()+m.getQuantite(), getUnite());
}

@Test
public void AdditionTest() throws UniteDistincteException {
    SommeArgent expected = new SommeArgent(26, "CHF");
    SommeArgent result = m12CHF.add(m14CHF);
    Assert.assertTrue(expected.equals(result));
}
```

The screenshot shows an IDE with three panels. The top panel displays the execution progress of a test suite, indicating that the test passed after 0.075 seconds. The middle panel shows the source code of the test class, which includes a test method that expects a `UnitéDistincteException` to be thrown. The bottom panel shows the failure trace, which indicates that the test failed because the expected exception was not thrown. The stack trace points to the `testAddUnitesDistinctes` method in the `UnitéDistincteException` class.

12 –

Finished after 0.067 seconds

Runs: 2/2 Errors: 0 Failures: 0

JUnit4 [Runner: JUnit 5] (0.000 s)

- EqualsTest (0.000 s)
- testAddUnitesDistinctes (0.000 s)

Failure Trace

```

23     m14CHF = new SommeArgent(14, "CHF");
24     m14USD = new SommeArgent(14, "USD");
25     nbPassages++;
26     System.out.println(nbPassages + "ème passage avant exécution d'une méthode
27 )
28
29 @After
30 public void tearDown() {
31     System.out.println(nbPassages + "ème passage APRES exécution d'une méthode
32 }
33
34
35 /**@Test
36 public void AdditionTest() throws UniteDistincteException {
37
38     SommeArgent expected = new SommeArgent(26, "CHF");
39     SommeArgent result = m12CHF.add(m14CHF);
40     Assert.assertTrue(expected.equals(result));
41 }*/
42
43 @Test(expected = UniteDistincteException.class)
44 public void testAddUnitesDistinctes() throws UniteDistincteException {
45     SommeArgent m12CHF = new SommeArgent(12, "CHF");
46     SommeArgent m14USD = new SommeArgent(14, "USD");
47     SommeArgent m = m12CHF.add(m14USD);
48 }
49
50
51
52 @Test
53 public void EqualsTest() {
54     Assert.assertTrue(!m12CHF.equals(null));
55 }

```

Problems Javadoc Declaration Console x

<terminated> JUnit4 [JUnit] C:\Program Files\Java\jdk-11.0.11\bin\javaw.exe (30 avr. 2023 à 14:41:58 - 14:41:59)

1ème passage avant exécution d'une méthode de test
1ème passage APRES exécution d'une méthode de test
2ème passage avant exécution d'une méthode de test
2ème passage APRES exécution d'une méthode de test

un porte-monnaie

13 –


```

PorteMonnaie.java × SommeArgent.java
2
3 import java.util.HashMap;
4 public class PorteMonnaie {
5     private HashMap<String, Integer> contenu;
6     public HashMap<String, Integer> getContenu() {
7         return contenu;
8     }
9     public PorteMonnaie() {
10        contenu = new HashMap<String, Integer>();
11    }
12
13    public void ajouteSomme(SommeArgent sa) {
14        String unite = sa.getUnite();
15        int quantite = sa.getQuantite();
16        if (contenu.containsKey(unite)) {
17            quantite += contenu.get(unite);
18        }
19        contenu.put(unite, quantite);
20    }

```

14 –

```

public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("porte-monnaie |");
    for (String unite : contenu.keySet()) {
        SommeArgent sommeObjet = new SommeArgent(contenu.get(unite), unite);
        sb.append(sommeObjet.toString());
        sb.append("\n");
    }
    return sb.toString();
}

```

15 –

```

public boolean equals(Object obj) {
    if (obj == null || getClass() != obj.getClass()) {
        return false;
    }
    PorteMonnaie other = (PorteMonnaie) obj;
    HashMap<String, Integer> otherContenu = other.getContenu();
    if (contenu.size() != otherContenu.size()) {
        return false;
    }
    for (String unite : contenu.keySet()) {
        if (!otherContenu.containsKey(unite) || !contenu.get(unite).equals(otherContenu.get(unite))) {
            return false;
        }
    }
    return true;
}

```


The screenshot shows an IDE with two windows. The top window displays the source code for `PortMonnaieTest.java`. The code is as follows:

```

1 package tp_test1.junit.monprojet.test;
2
3 import static org.junit.Assert.assertEquals;
4 import static org.junit.Assert.assertNotEquals;
5
6 import org.junit.Test;
7
8 import tp_test1.junit.monprojet.PortMonnaie;
9 import tp_test1.junit.monprojet.SommeArgent;
10
11 public class PortMonnaieTest {
12
13     @Test
14     public void testAjouteSommeEtEquals() {
15         PortMonnaie pm1 = new PortMonnaie();
16         pm1.ajouteSomme(new SommeArgent(5, "EUR"));
17         pm1.ajouteSomme(new SommeArgent(10, "USD"));
18
19         PortMonnaie pm2 = new PortMonnaie();
20         pm2.ajouteSomme(new SommeArgent(10, "USD"));
21         pm2.ajouteSomme(new SommeArgent(5, "EUR"));
22
23         PortMonnaie pm3 = new PortMonnaie();
24         pm3.ajouteSomme(new SommeArgent(5, "EUR"));
25
26         assertEquals(pm1, pm2);
27         assertNotEquals(pm1, pm3);
28     }
29
30 }

```

The bottom window shows the JUnit test runner results. It indicates that the test finished after 0,069 seconds, with 1/1 runs, 0 errors, and 0 failures. A green progress bar is shown, and the test name `PortMonnaieTest [Runner: JUnit 5] (0,000 s)` is listed.