2022/2023

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

2 - la méthode equals

```java
@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    SommeArgent other = (SommeArgent) obj;
    if (quantite != other.quantite)
        return false;
    if (unite == null) {
        if (other.unite != null)
            return false;
    } else if (!unite.equals(other.unite))
        return false;
    return true;
}
```

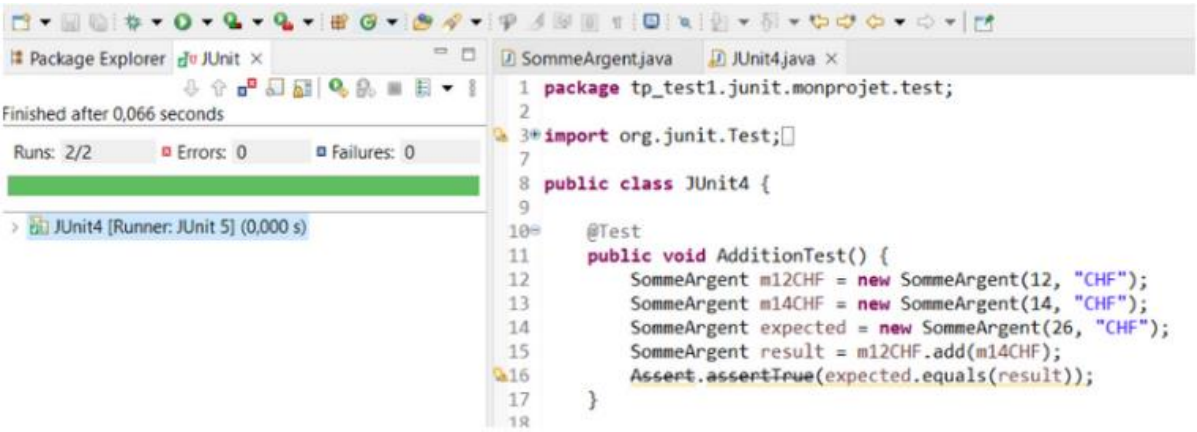
3 - classe de tests JUnit 4

```java
package tp_test1.junit.monprojet.test;

import org.junit.Test;

public class JUnit4 {

    

}
```

4 –

```java
package tp_test1.junit.monprojet.test;

import org.junit.Test;

public class JUnit4 {

    @Test
    public void AdditionTest() {
        SommeArgent m12CHF = new SommeArgent(12, "CHF");
        SommeArgent m14CHF = new SommeArgent(14, "CHF");
        SommeArgent expected = new SommeArgent(26, "CHF");
        SommeArgent result = m12CHF.add(m14CHF);
        Assert.assertTrue(expected.equals(result));
    }

```

5 –



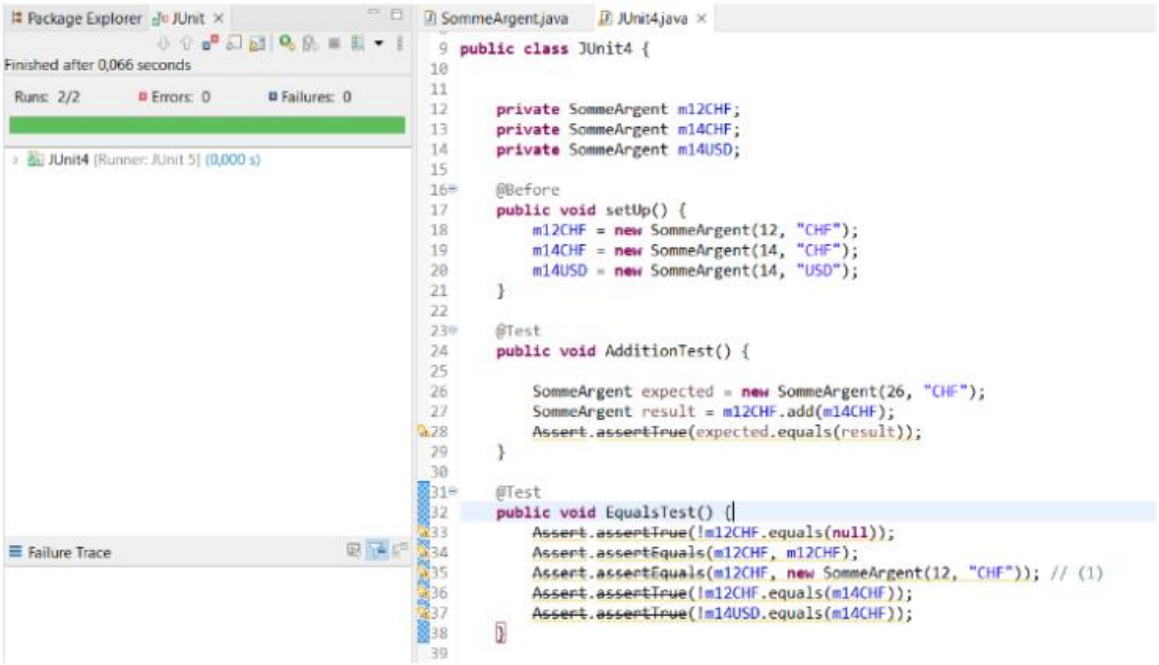
D'autres tests pour la classe SommeArgent

```java
@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 –

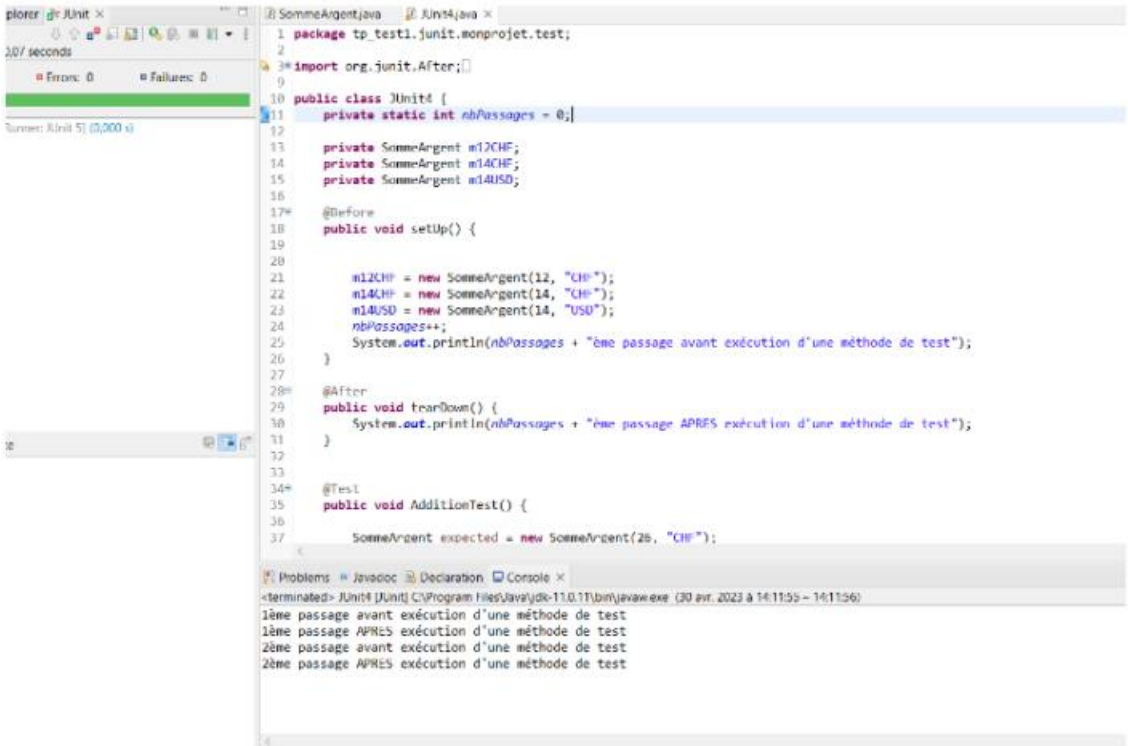
On teste si deux sommes d'argent avec des unités différentes ("CHF" et "USD") sont égales. Cette assertion doit retourner false car elles ont des unités différentes, même si leurs quantités sont les mêmes

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 –

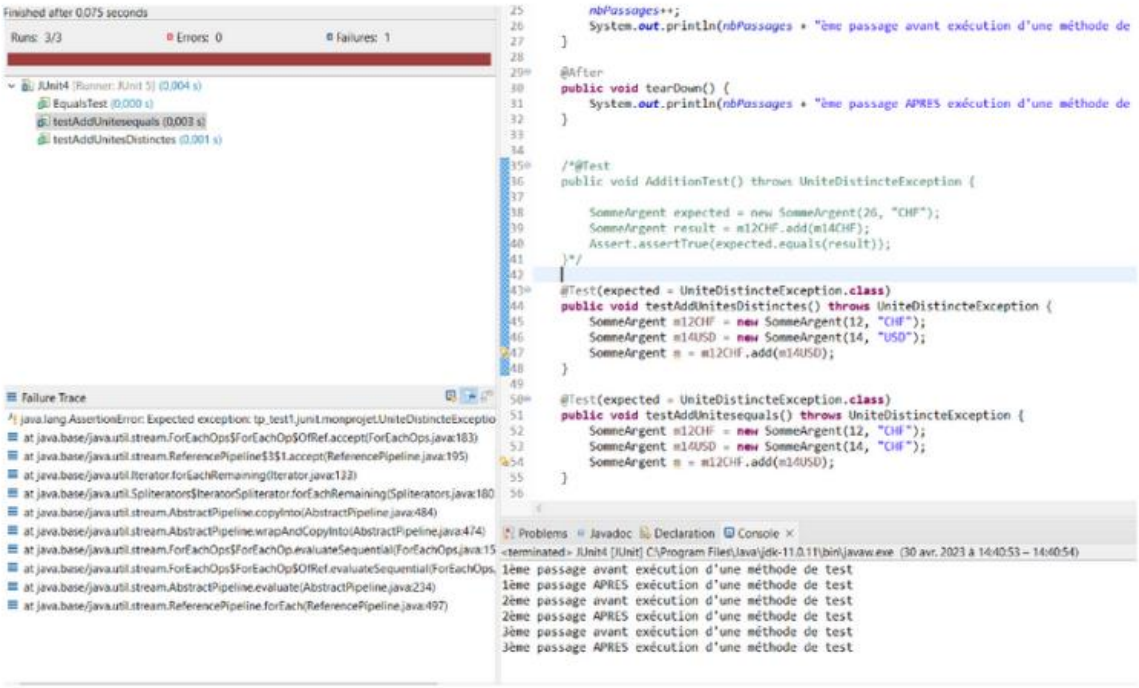


10 –

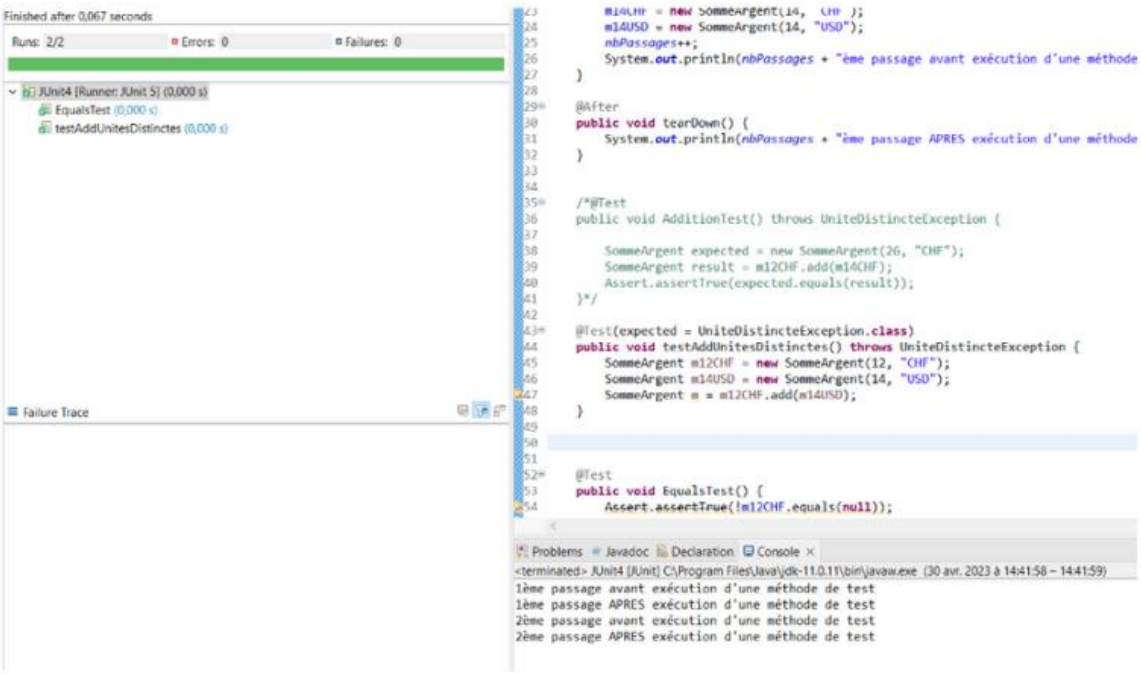
```
package tp_test1.junit.monprojet;

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));
}
```

11 –



12 –



un porte-monnaie

13 –

PorteMonnaie.java × SommeArgent.java

```java

import java.util.HashMap;
public class PorteMonnaie {
private HashMap<String, Integer> contenu;
public HashMap<String, Integer> getContenu() {
return contenu;
}
public PorteMonnaie() {
contenu = new HashMap<String, Integer>();
}

public void ajouteSomme(SommeArgent sa) {
    String unite = sa.getUnite();
    int quantite = sa.getQuantite();
    if (contenu.containsKey(unite)) {
        quantite += contenu.get(unite);
    }
    contenu.put(unite, quantite);
}
```

14 –

```java
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 –

```java
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;
}
```

16 –

PorteMonnaie.java | SommeArgent.java | PortMonnaieTest.java ×

```
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.PorteMonnaie;
9  import tp_test1.junit.monprojet.SommeArgent;
10
11 public class PortMonnaieTest {
12
13     @Test
14     public void testAjouteSommeEtEquals() {
15         PorteMonnaie pm1 = new PorteMonnaie();
16         pm1.ajouteSomme(new SommeArgent(5, "EUR"));
17         pm1.ajouteSomme(new SommeArgent(10, "USD"));
18
19         PorteMonnaie pm2 = new PorteMonnaie();
20         pm2.ajouteSomme(new SommeArgent(10, "USD"));
21         pm2.ajouteSomme(new SommeArgent(5, "EUR"));
22
23         PorteMonnaie pm3 = new PorteMonnaie();
24         pm3.ajouteSomme(new SommeArgent(5, "EUR"));
25
26         assertEquals(pm1, pm2);
27         assertNotEquals(pm1, pm3);
28     }
29
30
31 }
```