

RAPPORT DE TESTS LOGICIELS-TP 1

RÉALISÉ PAR:

Hadil Helali

Soulaima Khala

Raoua Trimech

Mouhamed Zied Brahmi



PREMIÈRE PARTIE - Le premier test d'une classe

1-la classe SommeArgent

```
SommeArgent.java x
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 x

- 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

```
19
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 }
38 }
```

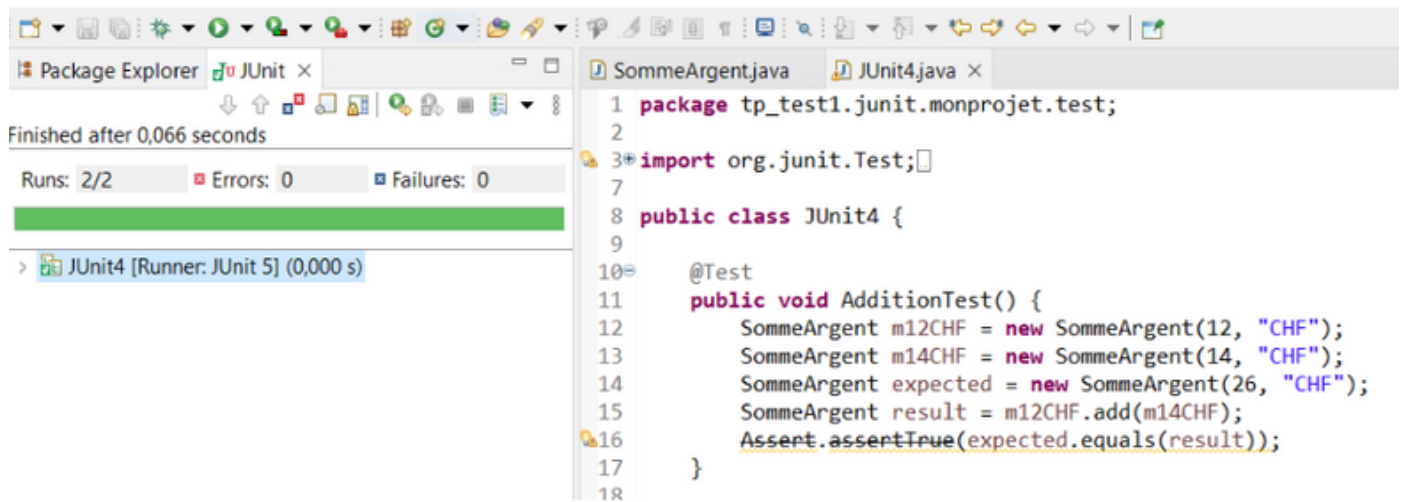
3-classe de tests JUnit 4

```
SommeArgent.java xJUnit4.java x
1 package tp_test1.junit.monprojet.test;
2
3 import org.junit.Test;
4
5 public class JUnit4 {
6
7 }
8
9
10
11
12
```

5-

```
SommeArgent.java xJUnit4.java x
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 }
```

6-



The screenshot shows an IDE with two tabs: 'SommeArgent.java' and 'JUnit4.java'. The 'JUnit4.java' tab is active, displaying the following code:

```
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 }
```

On the left, the 'Package Explorer' shows the project structure. Below it, a status bar indicates 'Finished after 0,066 seconds', 'Runs: 2/2', 'Errors: 0', and 'Failures: 0'. A console window at the bottom shows the command: '> JUnit4 [Runner: JUnit 5] (0,000 s)'.

SECONDE PARTIE - D'autres tests pour la classe SommeArgent

7-

```
@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));
}
```

Dans la dernière ligne de ce code, 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.

8-

```

9 public class JUnit4 {
10
11
12     private SommeArgent m12CHF;
13     private SommeArgent m14CHF;
14     private SommeArgent m14USD;
15
16     @Before
17     public void setUp() {
18         m12CHF = new SommeArgent(12, "CHF");
19         m14CHF = new SommeArgent(14, "CHF");
20         m14USD = new SommeArgent(14, "USD");
21     }
22
23     @Test
24     public void AdditionTest() {
25
26         SommeArgent expected = new SommeArgent(26, "CHF");
27         SommeArgent result = m12CHF.add(m14CHF);
28         Assert.assertTrue(expected.equals(result));
29     }
30
31     @Test
32     public void EqualsTest() {
33         Assert.assertTrue(!m12CHF.equals(null));
34         Assert.assertEquals(m12CHF, m12CHF);
35         Assert.assertEquals(m12CHF, new SommeArgent(12, "CHF")); // (1)
36         Assert.assertTrue(!m12CHF.equals(m14CHF));
37         Assert.assertTrue(!m14USD.equals(m14CHF));
38     }
39 }

```

9- Ces ensembles d'objets sont appelés "**fixtures**" en anglais. 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.

10-

```

1 package tp_test1.junit.monprojet.test;
2
3 import org.junit.After;
4
5 public class JUnit4 {
6     private static int nbPassages = 0;
7
8     private SommeArgent m12CHF;
9     private SommeArgent m14CHF;
10    private SommeArgent m14USD;
11
12    @Before
13    public void setUp() {
14
15        m12CHF = new SommeArgent(12, "CHF");
16        m14CHF = new SommeArgent(14, "CHF");
17        m14USD = new SommeArgent(14, "USD");
18        nbPassages++;
19        System.out.println(nbPassages + "ème passage avant exécution d'une méthode de test");
20    }
21
22    @After
23    public void tearDown() {
24        System.out.println(nbPassages + "ème passage APRES exécution d'une méthode de test");
25    }
26
27    @Test
28    public void AdditionTest() {
29
30        SommeArgent expected = new SommeArgent(26, "CHF");

```

```

<terminated> JUnit4 [JUnit] C:\Program Files\Java\jdk-11.0.11\bin\javaw.exe (30 avr. 2023 à 14:11:55 - 14:11:56)
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

```

11-

```
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 "unite 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));
}
```

11.2-

Finished after 0,075 seconds

Runs: 3/3 Errors: 0 Failures: 1

JUnit4 [Runner: JUnit 5] (0,004 s)

- EqualsTest (0,000 s)
- testAddUnitesEquals (0,003 s)
- testAddUnitesDistinctes (0,001 s)

Failure Trace

```
java.lang.AssertionError: Expected exception: tp_test1.junit.monprojet.UniteDistincteException
    at java.base/java.util.stream.ForEachOps$ForEachOp$OfRef.accept(ForEachOps.java:183)
    at java.base/java.util.stream.ReferencePipeline$3$1.accept(ReferencePipeline.java:195)
    at java.base/java.util.Iterator.forEachRemaining(Iterator.java:133)
    at java.base/java.util.Spliterators$IteratorSpliterator.forEachRemaining(Spliterators.java:180)
    at java.base/java.util.stream.AbstractPipeline.copyInto(AbstractPipeline.java:484)
    at java.base/java.util.stream.AbstractPipeline.wrapAndCopyInto(AbstractPipeline.java:474)
    at java.base/java.util.stream.ForEachOps$ForEachOp.evaluateSequential(ForEachOps.java:151)
    at java.base/java.util.stream.ForEachOps$ForEachOp$OfRef.evaluateSequential(ForEachOps.java:15)
    at java.base/java.util.stream.AbstractPipeline.evaluate(AbstractPipeline.java:234)
    at java.base/java.util.stream.ReferencePipeline.forEach(ReferencePipeline.java:497)
```

```

25     nbPassages++;
26     System.out.println(nbPassages + "ème passage avant exécution d'une méthode de
27 }
28
29 @After
30 public void tearDown() {
31     System.out.println(nbPassages + "ème passage APRES exécution d'une méthode de
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 @Test(expected = UniteDistincteException.class)
51 public void testAddUnitesEquals() throws UniteDistincteException {
52     SommeArgent m12CHF = new SommeArgent(12, "CHF");
53     SommeArgent m14USD = new SommeArgent(14, "CHF");
54     SommeArgent m = m12CHF.add(m14USD);
55 }
56

```

< Problems Javadoc Declaration Console x

<terminated> JUnit4 [JUnit] C:\Program Files\Java\jdk-11.0.11\bin\javaw.exe (30 avr. 2023 à 14:40:53 – 14:40:54)

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
3ème passage avant exécution d'une méthode de test
3ème passage APRES exécution d'une méthode de test

11.3-

The screenshot shows an IDE with a JUnit test runner on the left and a Java code editor on the right. The test runner indicates that the tests passed successfully. The code editor shows a Java class with several test methods, including a test for adding distinct units and an equals test.

```
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
```

TROISIEME PARTIE - un porte-monnaie

12-

The screenshot shows a code editor with two files open: PorteMonnaie.java and SommeArgent.java. The code in PorteMonnaie.java implements a wallet class that uses a HashMap to store the contents of a wallet.

```
PorteMonnaie.java x 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     }
}
```

13-

```

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();
}

```

14-

```

1 public boolean equals(Object obj) {
2     if (obj == null || getClass() != obj.getClass()) {
3         return false;
4     }
5     PorteMonnaie other = (PorteMonnaie) obj;
6     HashMap<String, Integer> otherContenu = other.getContenu();
7     if (contenu.size() != otherContenu.size()) {
8         return false;
9     }
10    for (String unite : contenu.keySet()) {
11        if (!otherContenu.containsKey(unite) || !contenu.get(unite).equals(otherContenu.get(unite)))
12            return false;
13    }
14    return true;
15 }

```

```

PorteMonnaie.java  SommeArgent.java  PortMonnaieTest.java X
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 }

```

```

Package Explorer  JUnit x
Finished after 0,069 seconds

Runs: 1/1  Errors: 0  Failures: 0

> PortMonnaieTest [Runner: JUnit 5] (0,000 s)

```