



CONCEPTION AVANCÉE

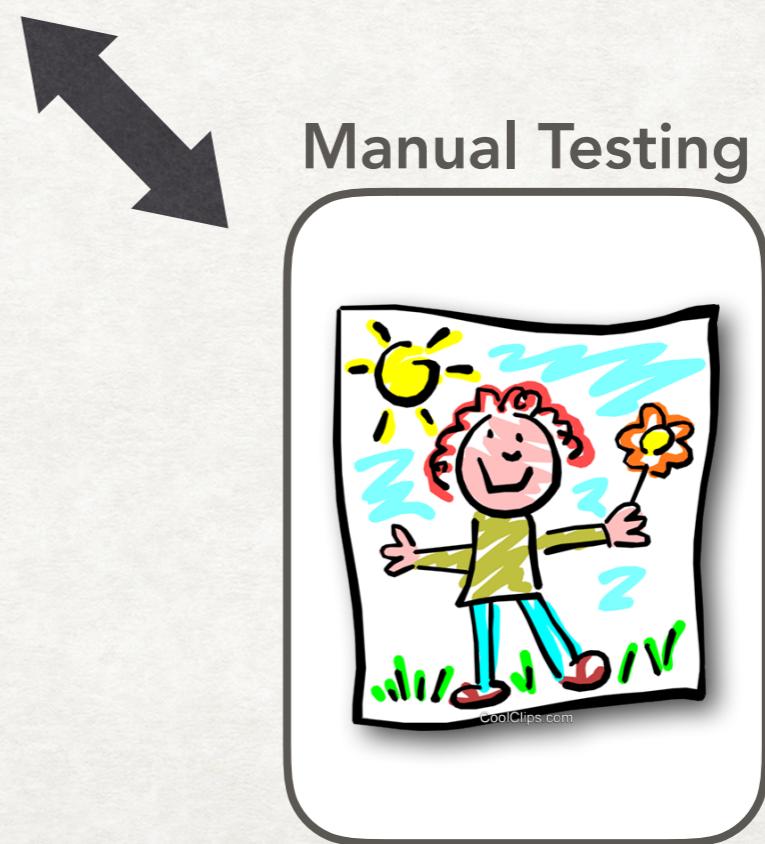
COURS 4: TEST UNITAIRE (JUNIT)

lazaar@lirmm.fr

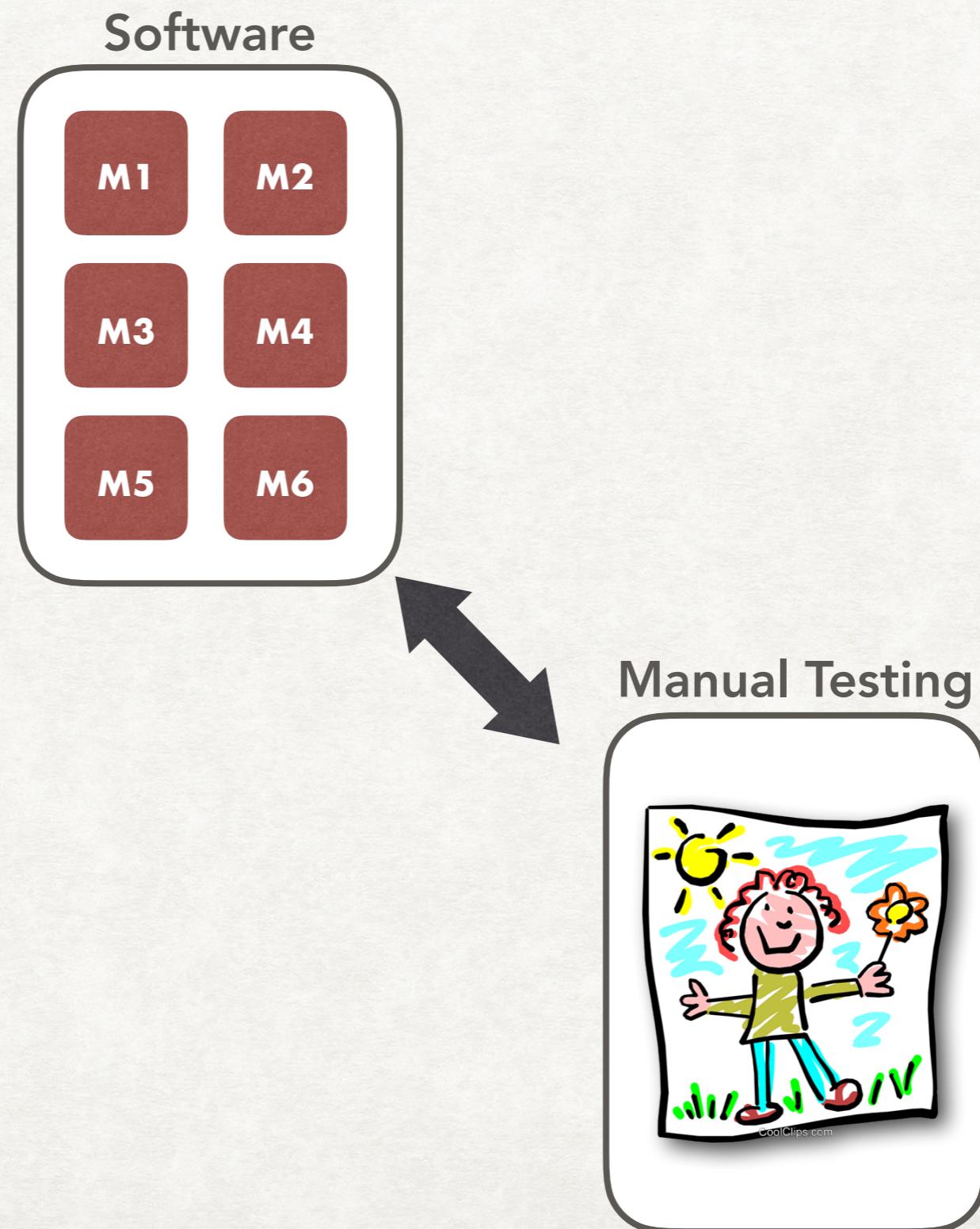
JAVA Unit Testing

+ JUnit 5

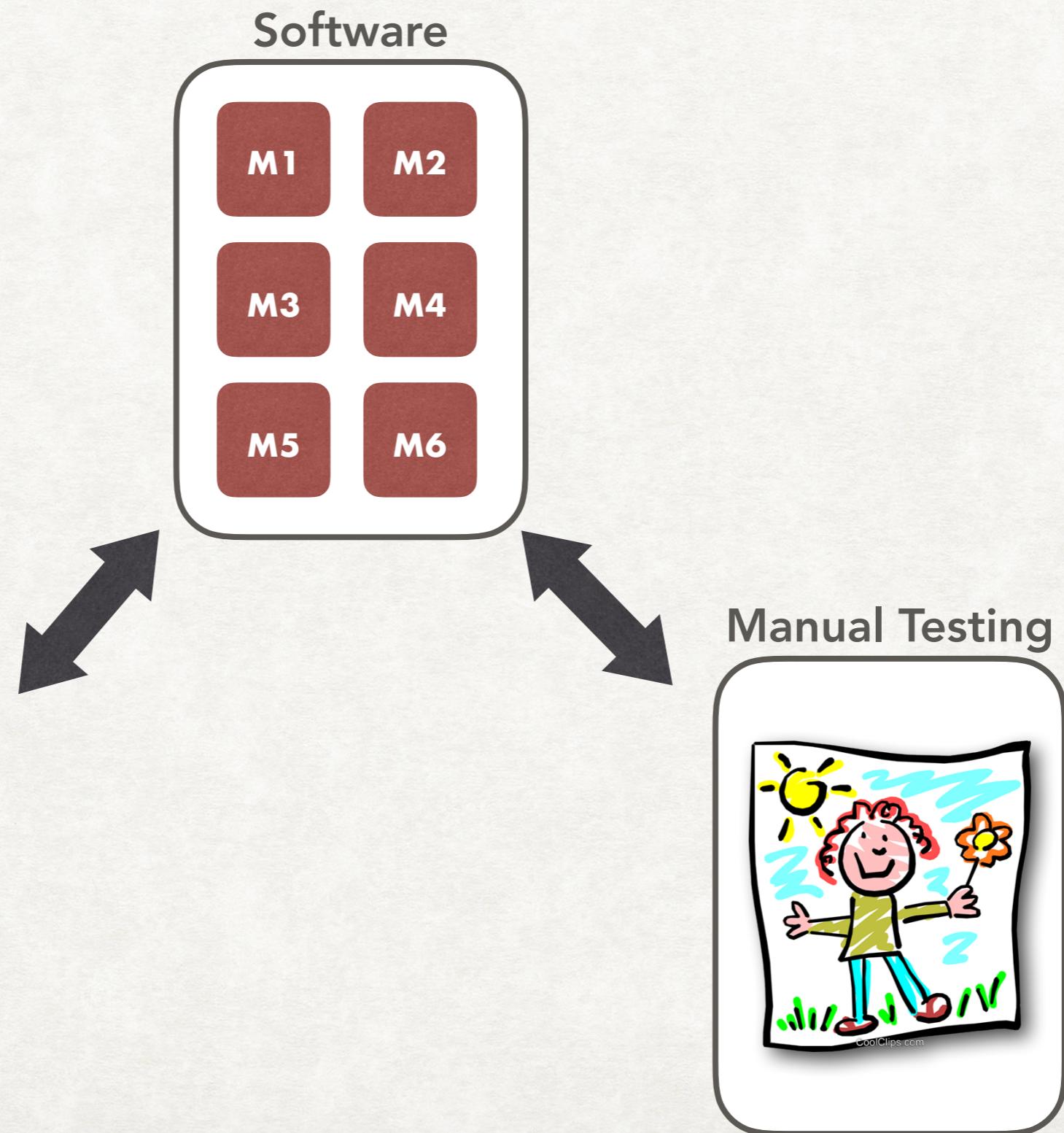
Tests



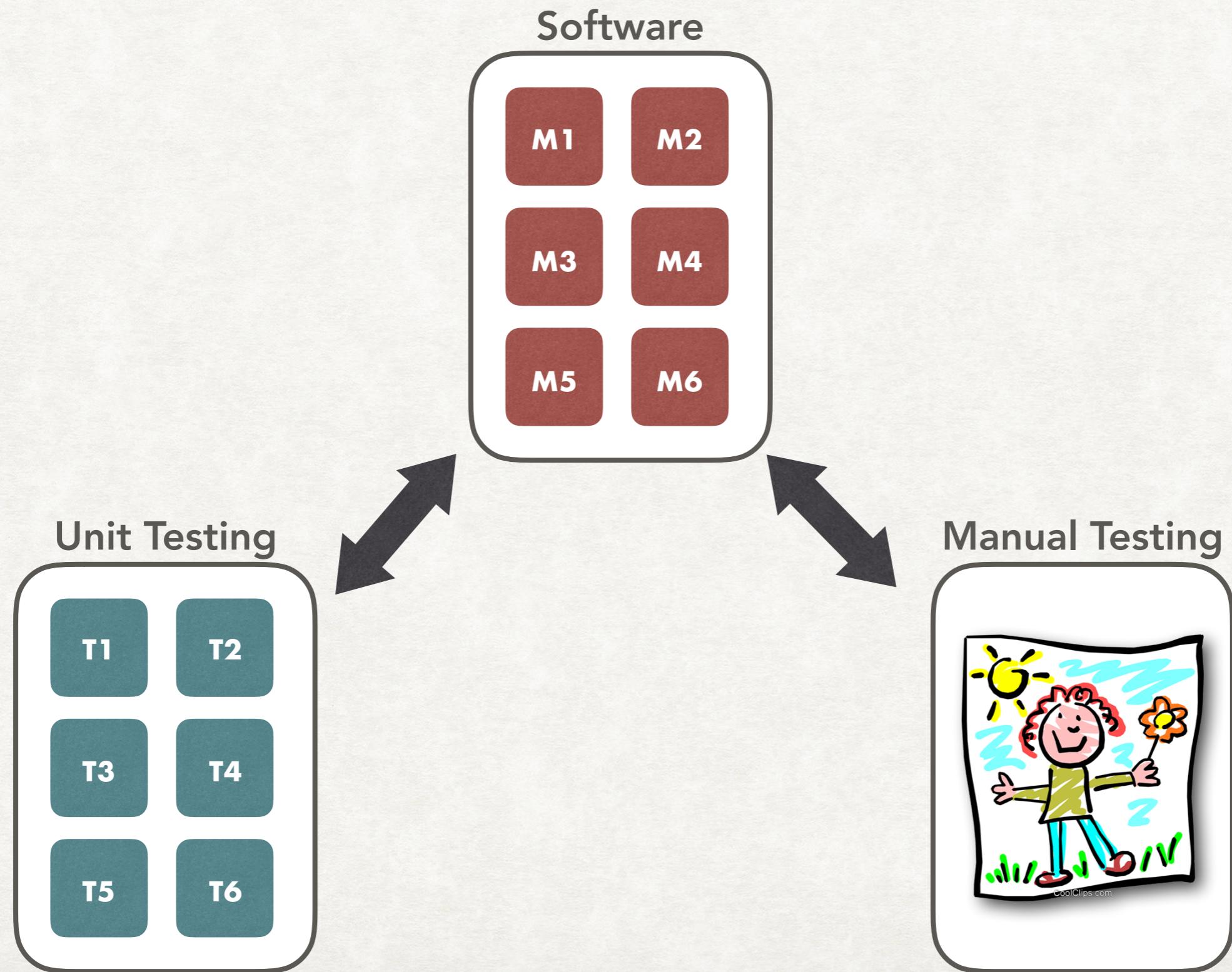
Tests



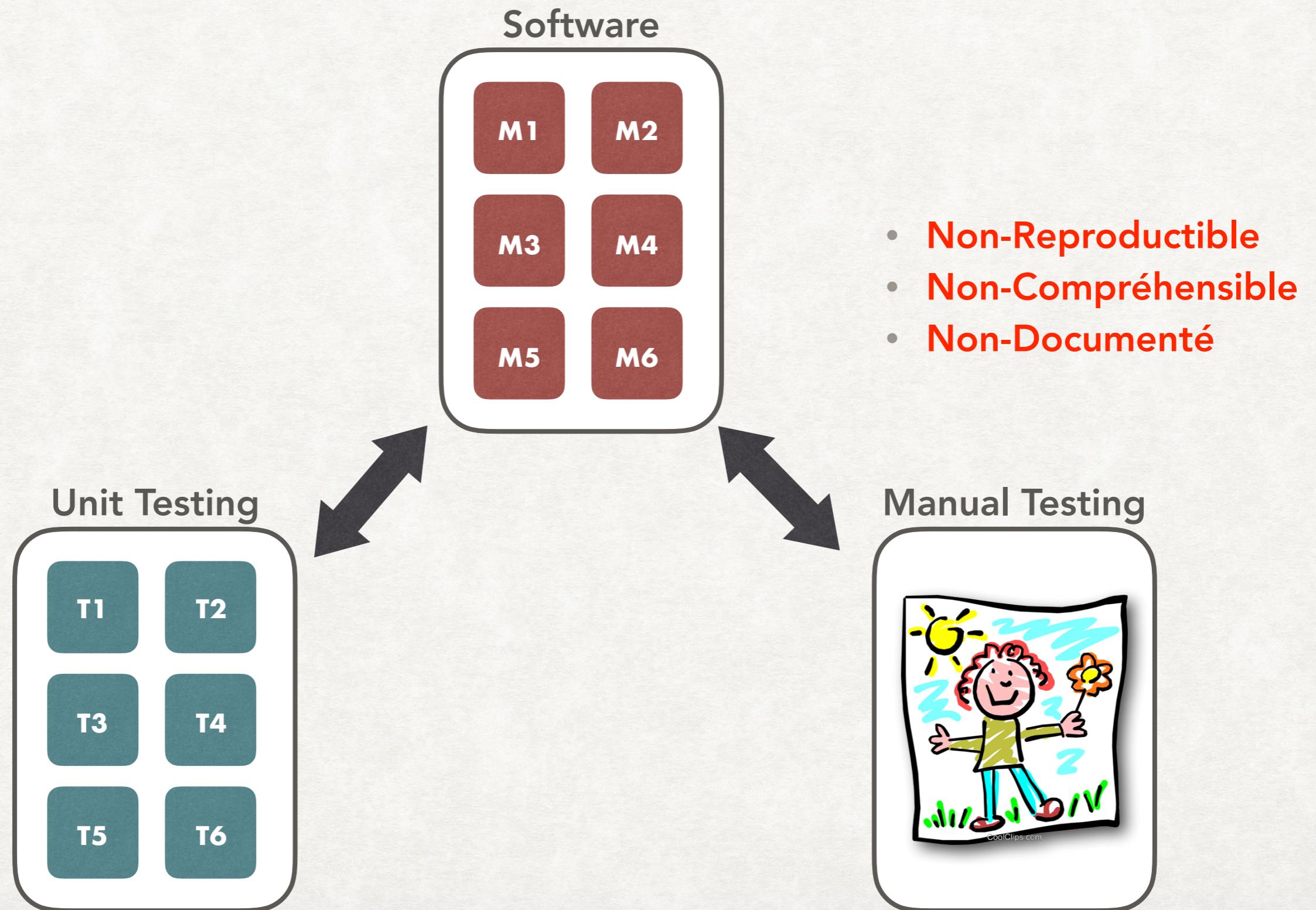
Tests



Tests



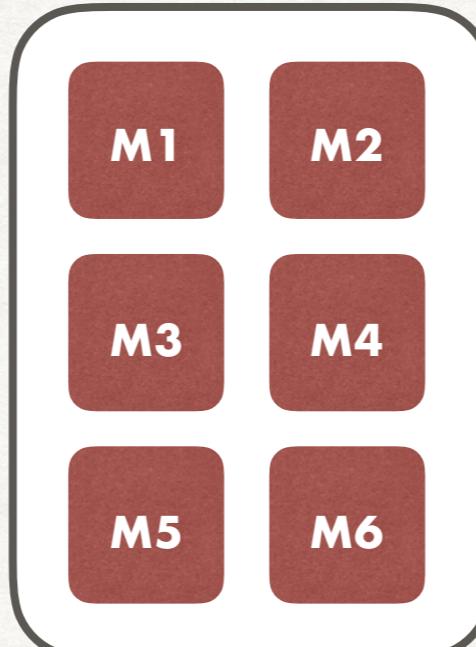
Tests



Tests

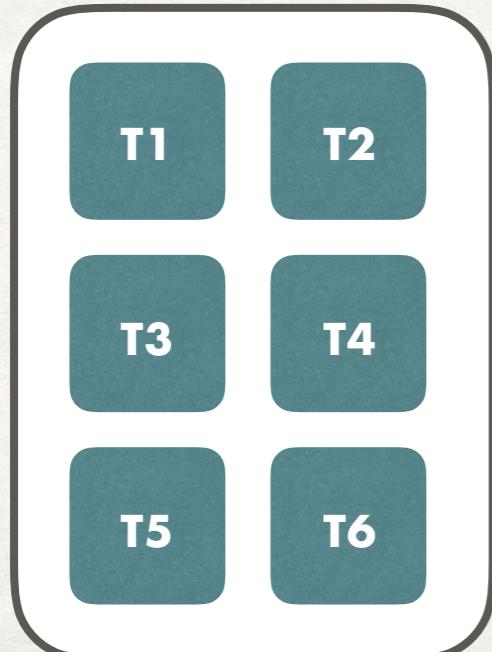
- Reproductible
- Compréhensible
- Documenté

Software



- Non-Reproducible
- Non-Compréhensible
- Non-Documenté

Unit Testing



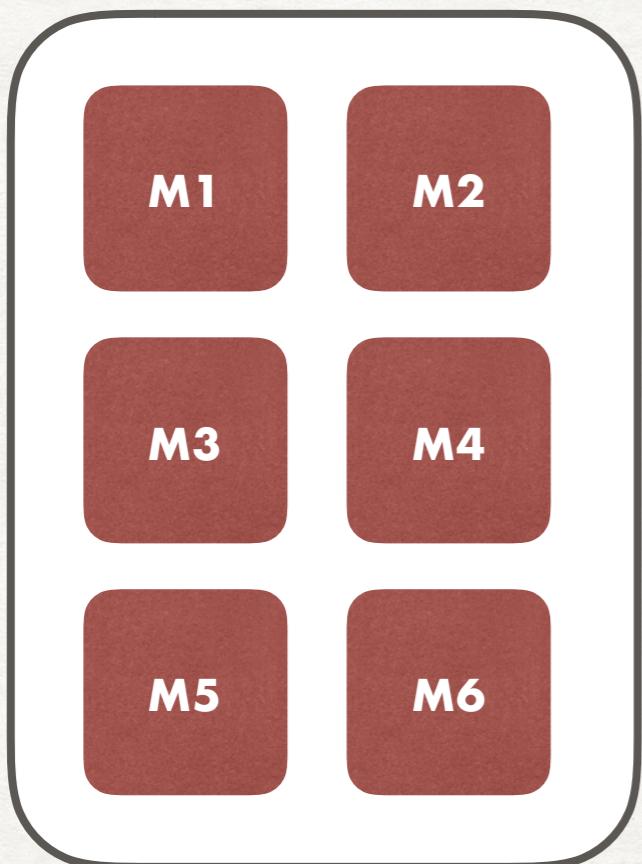
Manual Testing



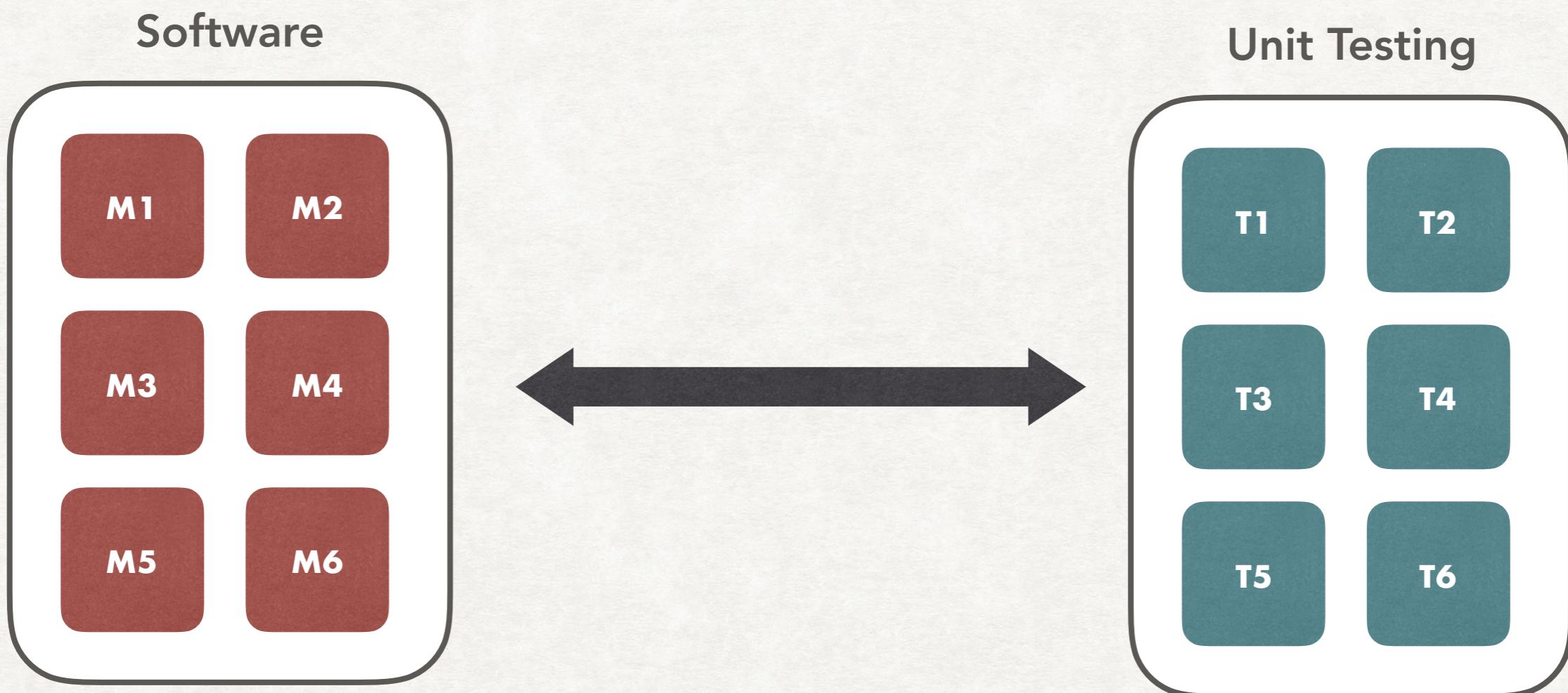
Tests

Tests

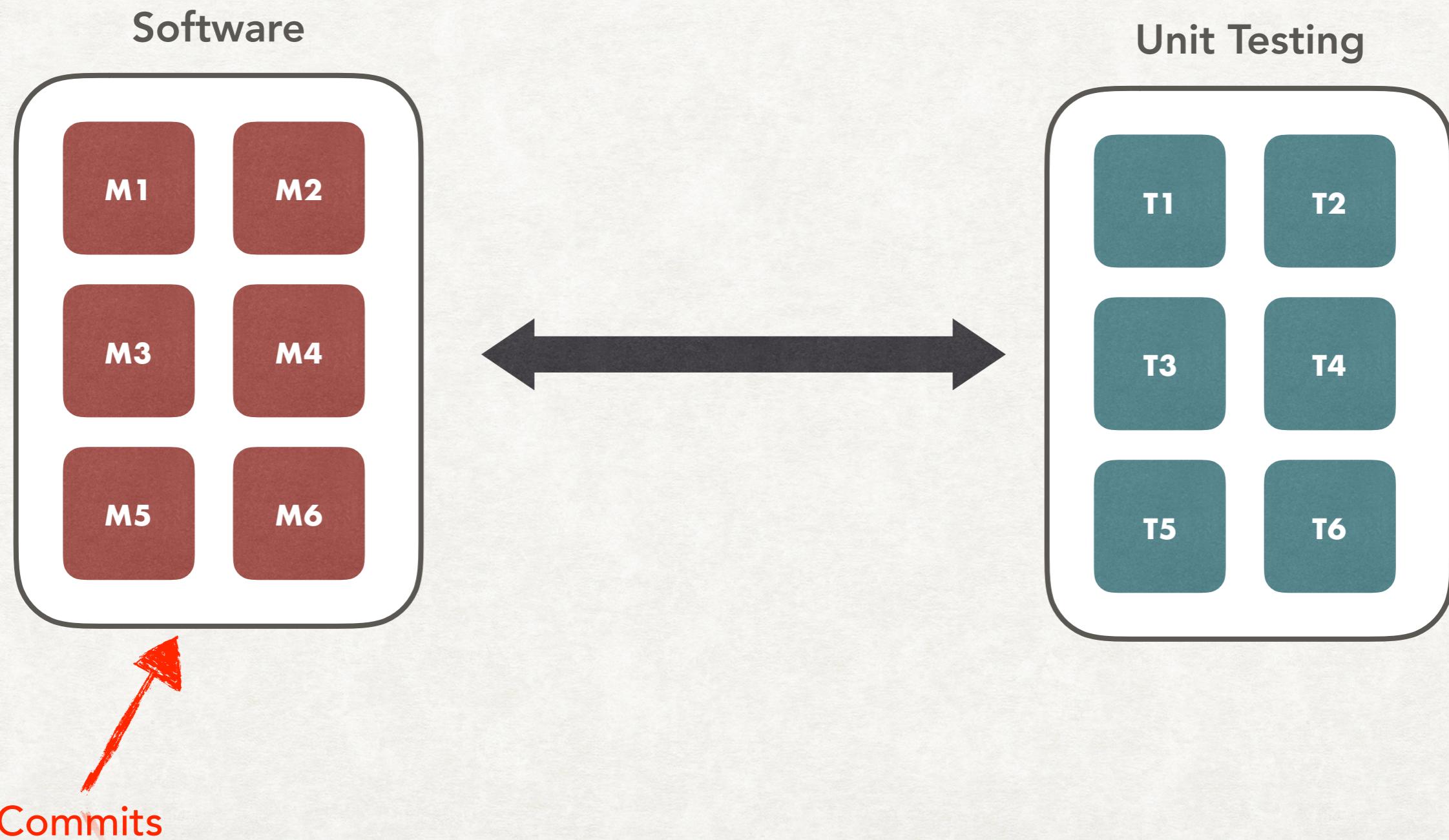
Software



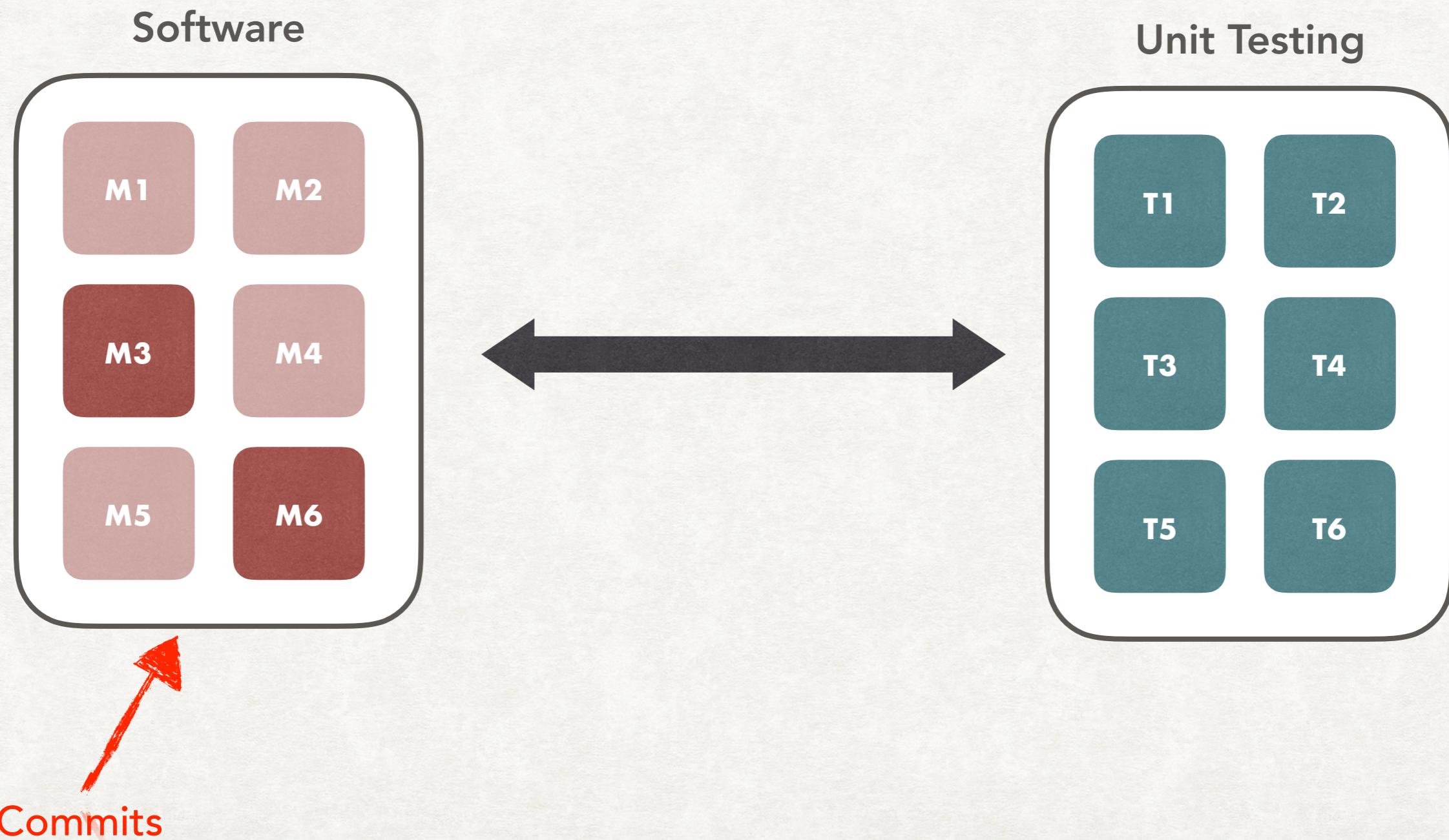
Tests



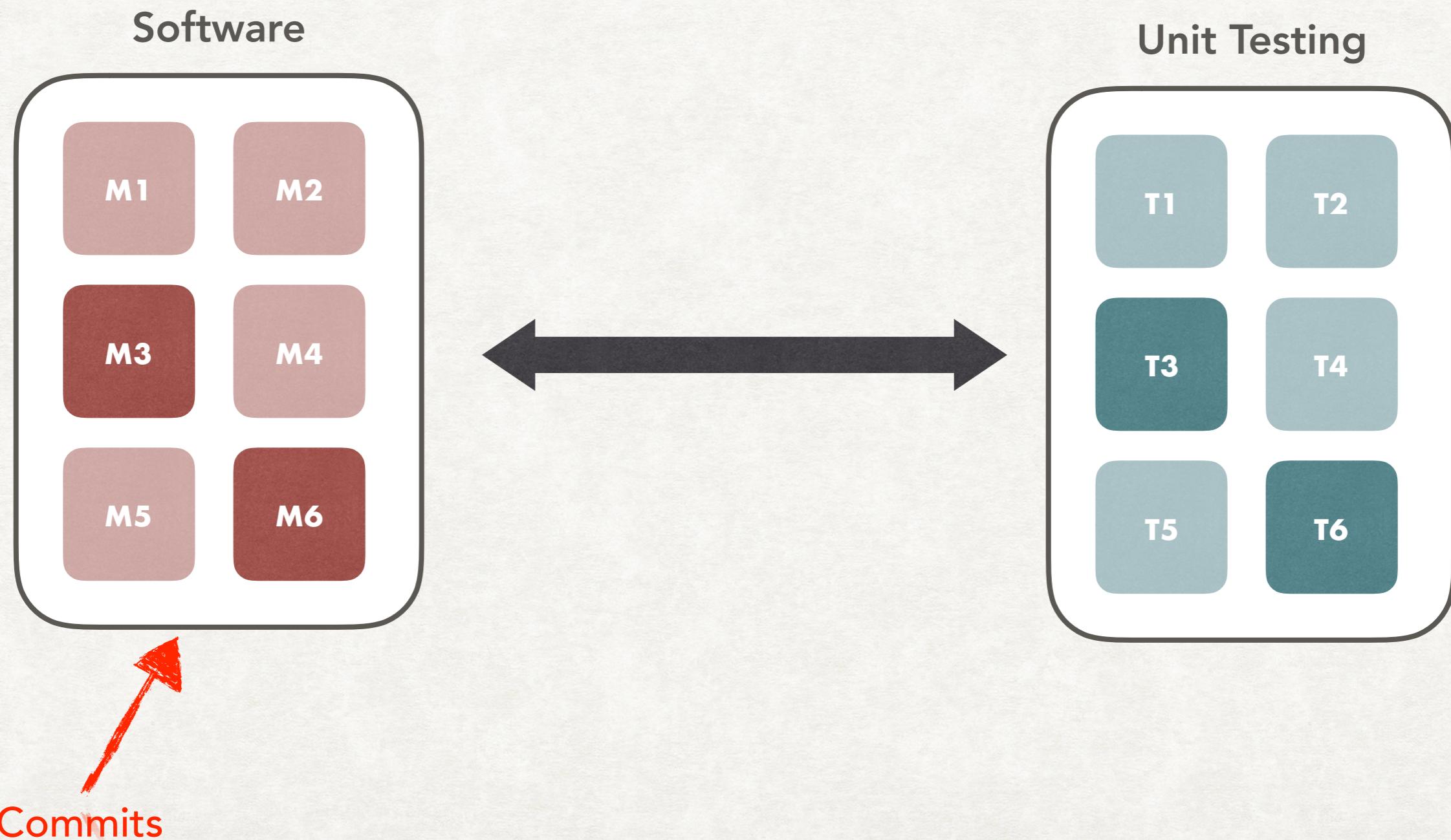
Tests



Tests

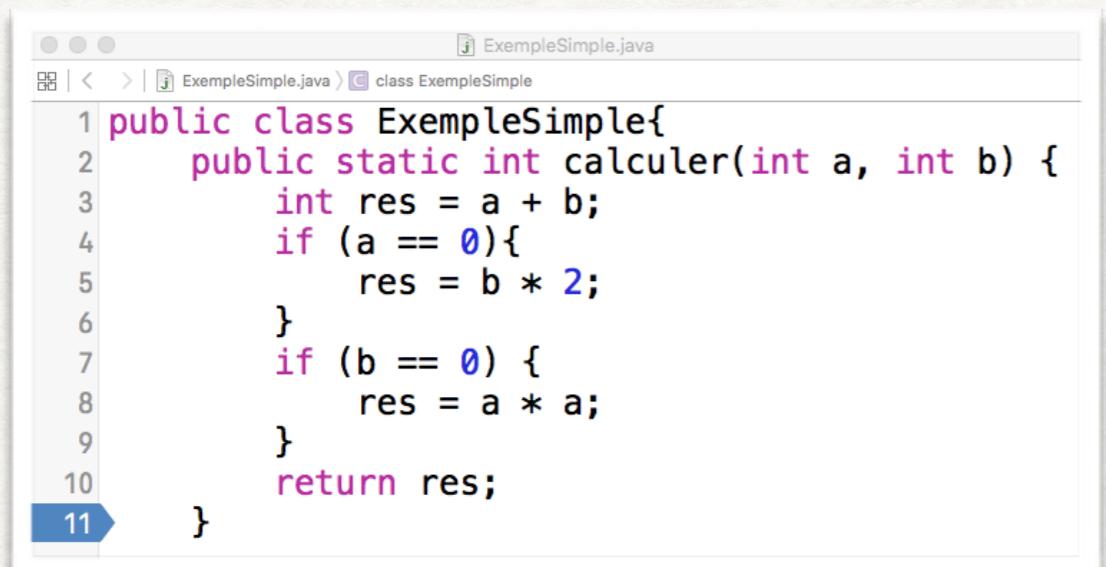


Tests



EXEMPLE TRÈS SIMPLE !

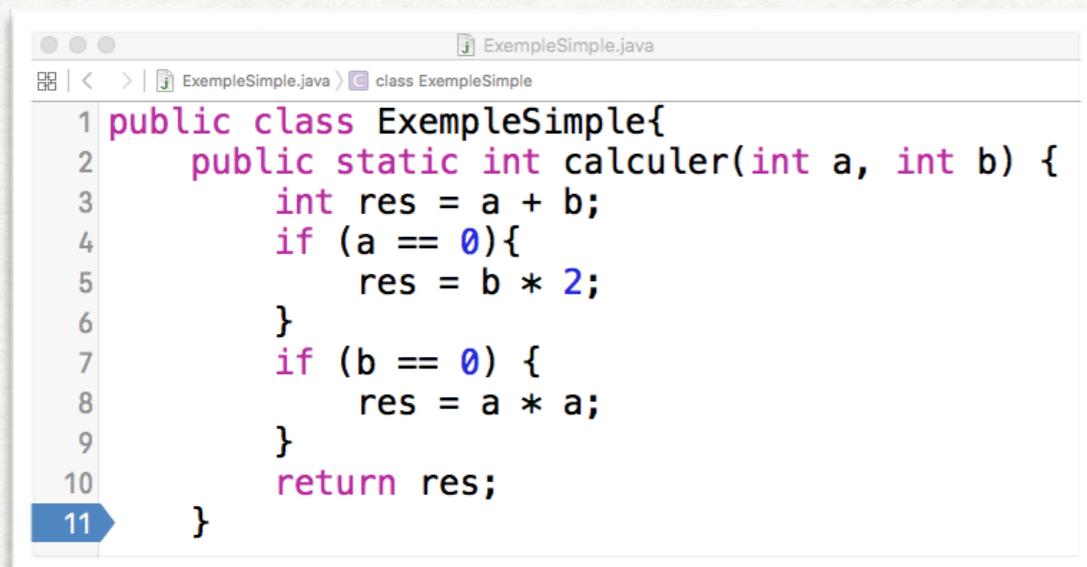
EXEMPLE TRÈS SIMPLE !



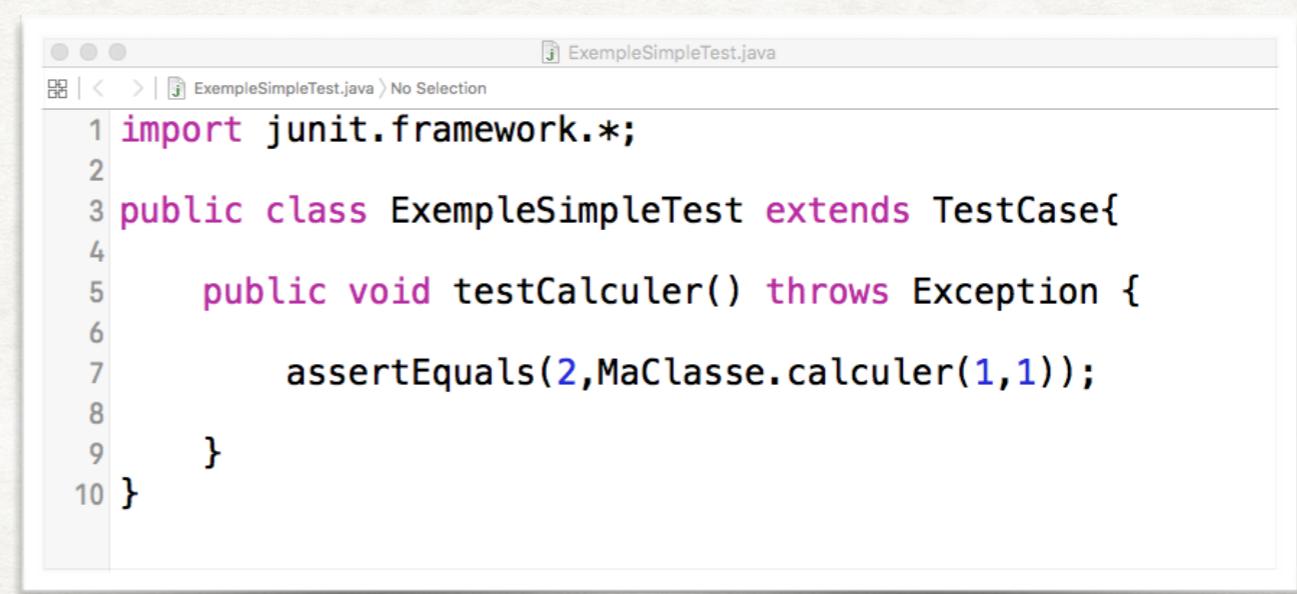
A screenshot of a Java code editor window titled "ExempleSimple.java". The code defines a public class "ExempleSimple" with a static method "calculer". The method takes two integers, "a" and "b", and returns their sum. If "a" is 0, it returns "b" doubled. If "b" is 0, it returns "a" squared. The code is numbered from 1 to 11.

```
1 public class ExempleSimple{
2     public static int calculer(int a, int b) {
3         int res = a + b;
4         if (a == 0){
5             res = b * 2;
6         }
7         if (b == 0) {
8             res = a * a;
9         }
10        return res;
11    }
```

EXEMPLE TRÈS SIMPLE !

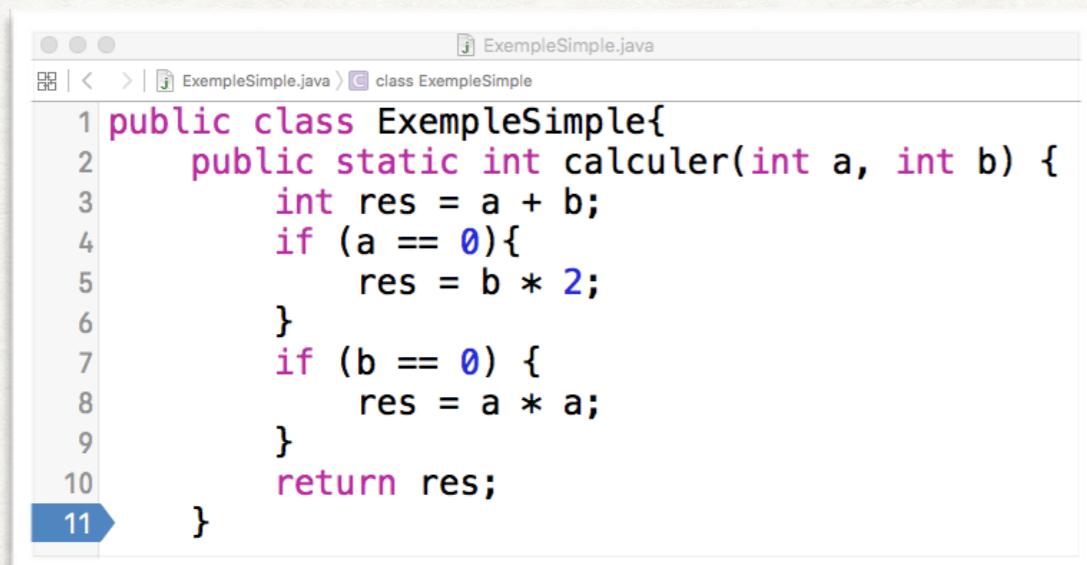


```
ExempleSimple.java
1 public class ExempleSimple{
2     public static int calculer(int a, int b) {
3         int res = a + b;
4         if (a == 0){
5             res = b * 2;
6         }
7         if (b == 0) {
8             res = a * a;
9         }
10        return res;
11    }
```

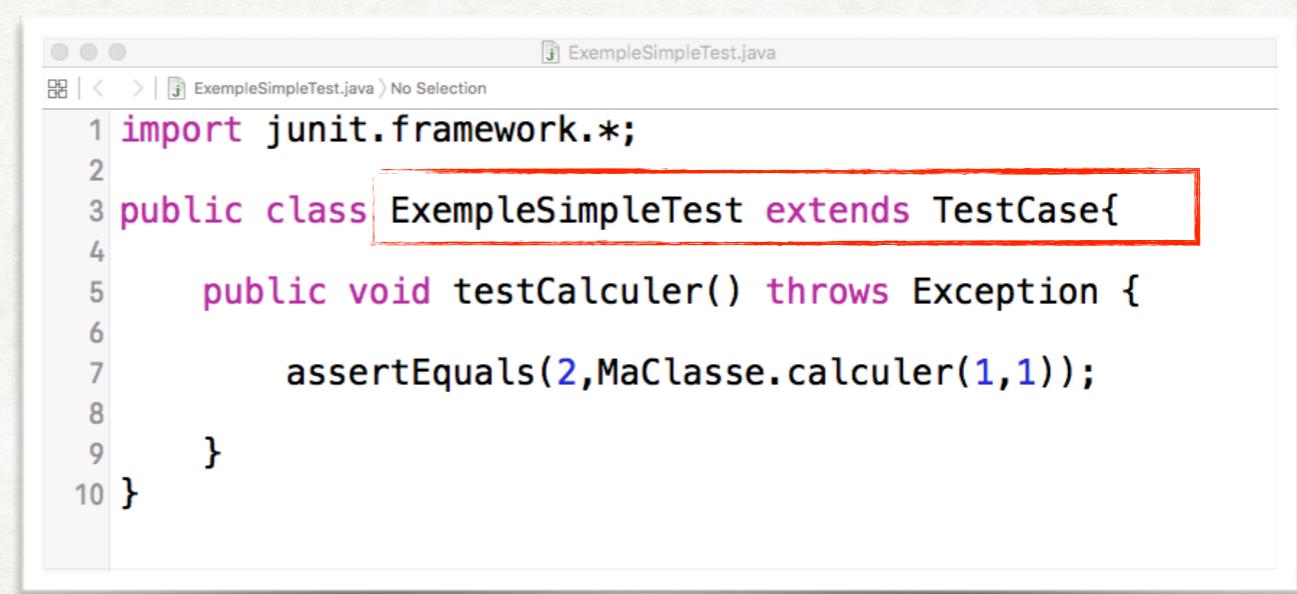


```
ExempleSimpleTest.java
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10 }
```

EXEMPLE TRÈS SIMPLE !

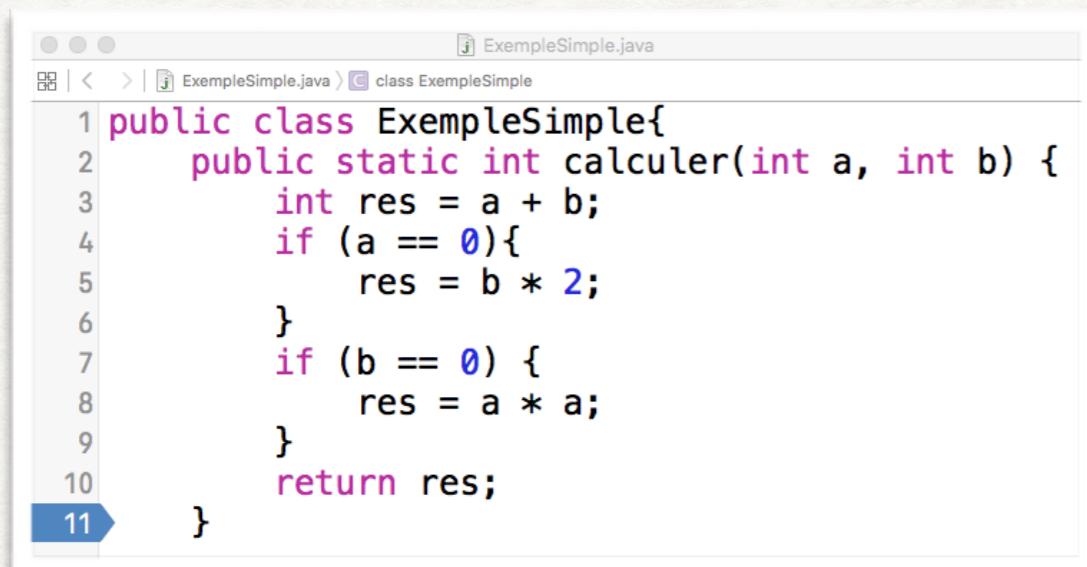


```
ExempleSimple.java
1 public class ExempleSimple{
2     public static int calculer(int a, int b) {
3         int res = a + b;
4         if (a == 0){
5             res = b * 2;
6         }
7         if (b == 0) {
8             res = a * a;
9         }
10        return res;
11    }
```

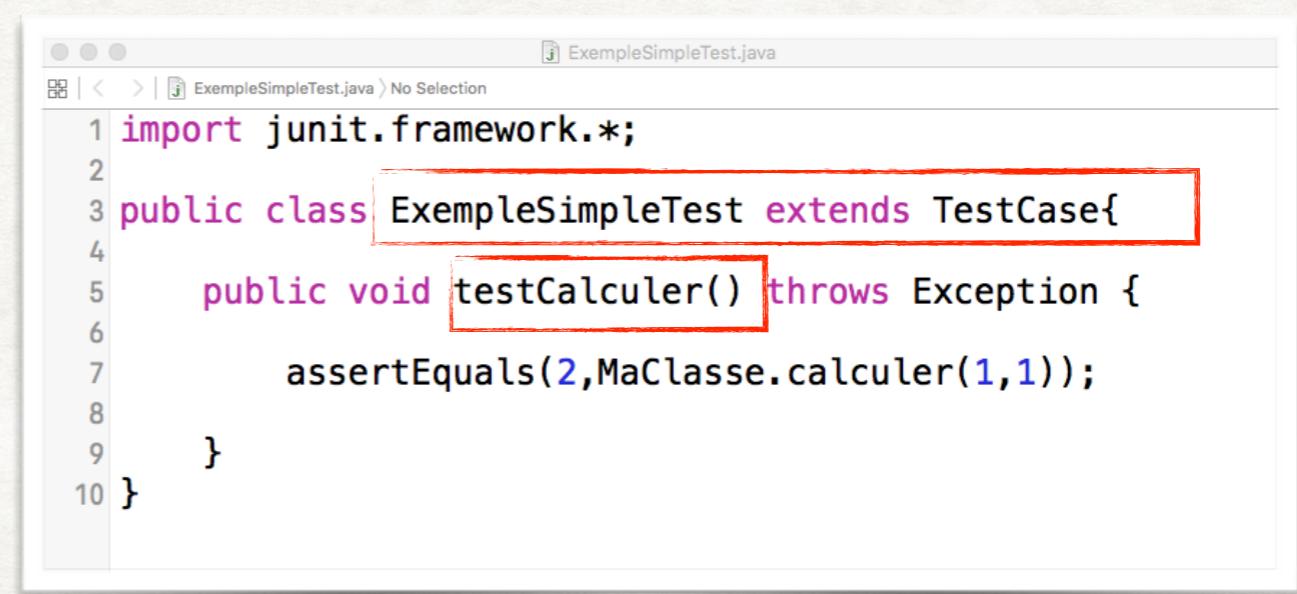


```
ExempleSimpleTest.java
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10 }
```

EXEMPLE TRÈS SIMPLE !

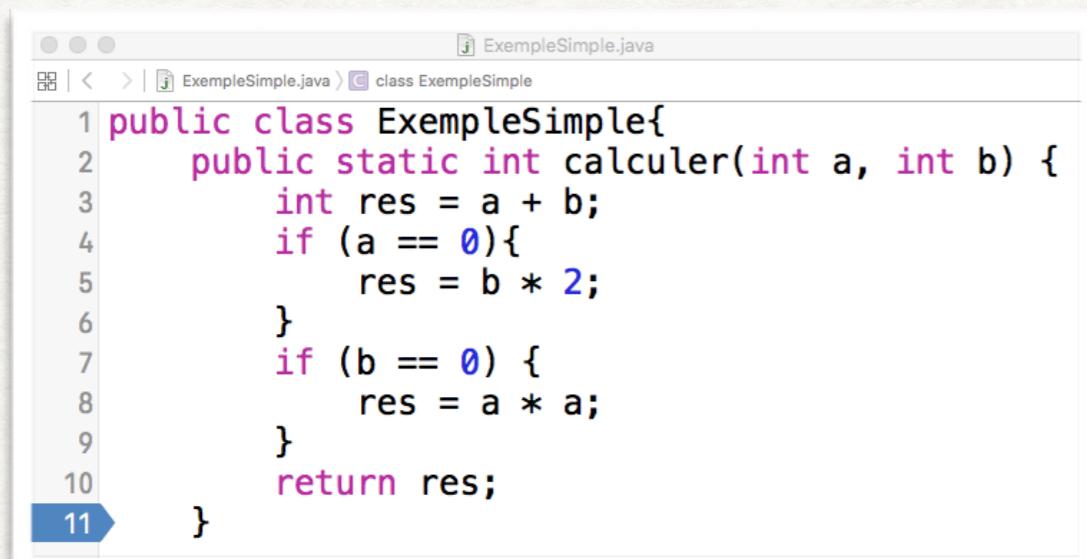


```
ExempleSimple.java
1 public class ExempleSimple{
2     public static int calculer(int a, int b) {
3         int res = a + b;
4         if (a == 0){
5             res = b * 2;
6         }
7         if (b == 0) {
8             res = a * a;
9         }
10        return res;
11    }
```

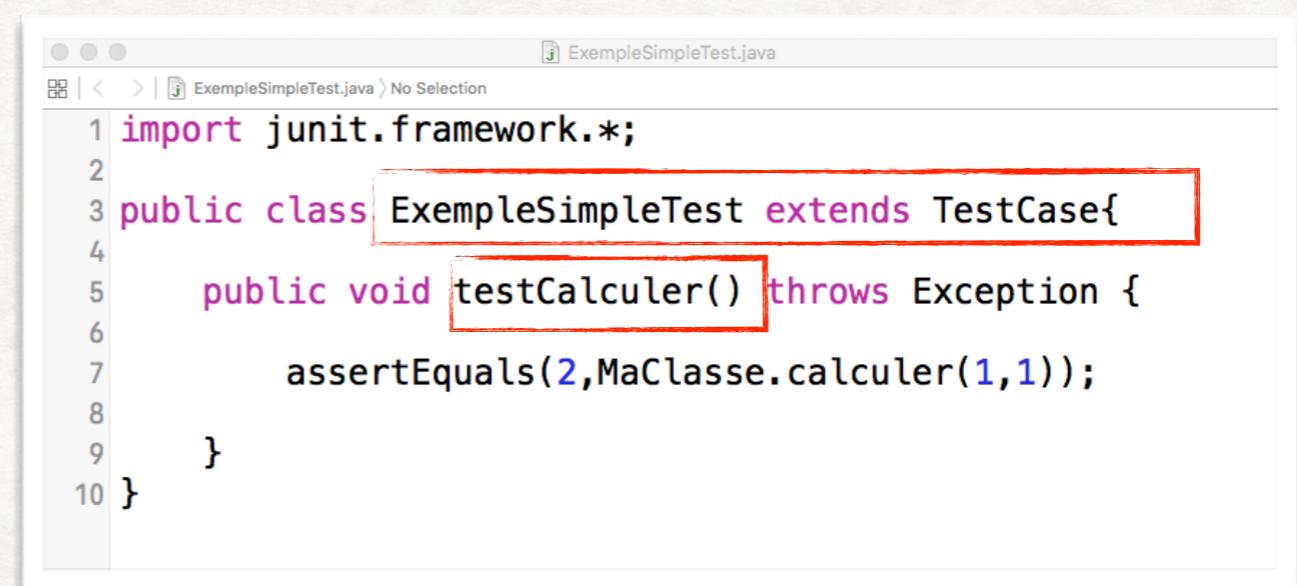


```
ExempleSimpleTest.java
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2, MaClasse.calculer(1,1));
8
9     }
10 }
```

EXEMPLE TRÈS SIMPLE !



```
ExempleSimple.java
1 public class ExempleSimple{
2     public static int calculer(int a, int b) {
3         int res = a + b;
4         if (a == 0){
5             res = b * 2;
6         }
7         if (b == 0) {
8             res = a * a;
9         }
10        return res;
11    }
}
```

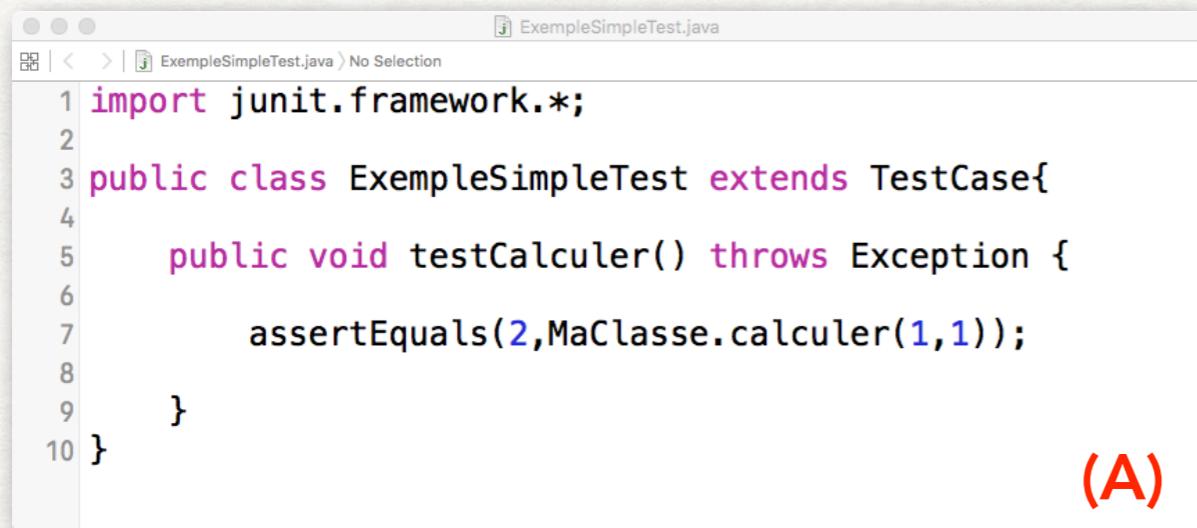


```
ExempleSimpleTest.java
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10 }
```

```
Java ExempleSimple.java
java -cp junit.jar;. junit.textui.TestRunner ExempleSimple
/java/testjunit>java -cp junit.jar;. junit.textui.TestRunner
ExempleSimpleTest
.
Time: 0,01
OK (1 test)
```

TROUVEZ L'INTRUS

TROUVEZ L'INTRUS

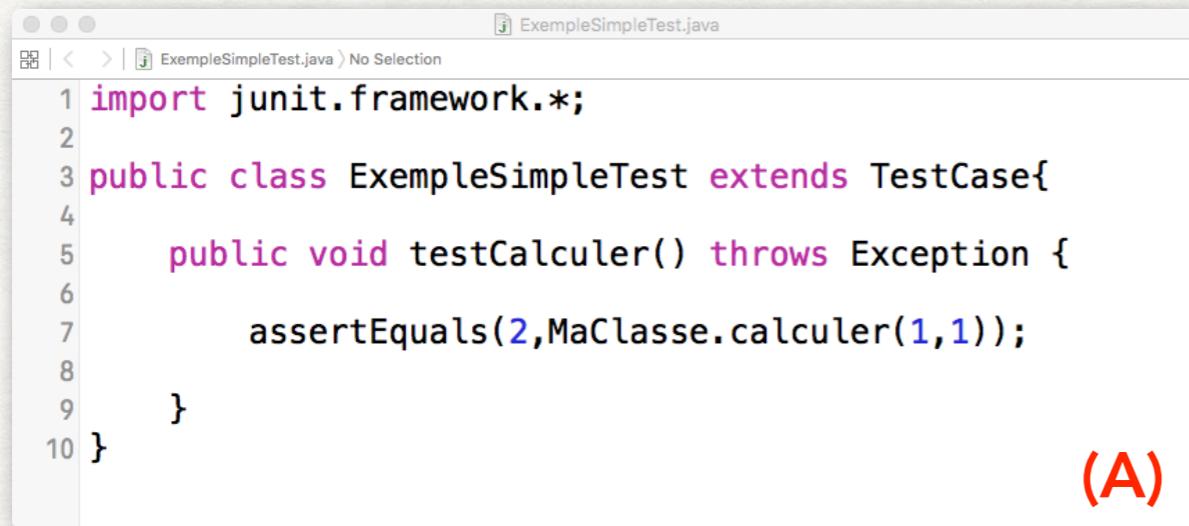


The screenshot shows a Java code editor window titled "ExempleSimpleTest.java". The code is a simple JUnit test case:

```
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10}
```

A red label "(A)" is positioned to the right of the code editor window.

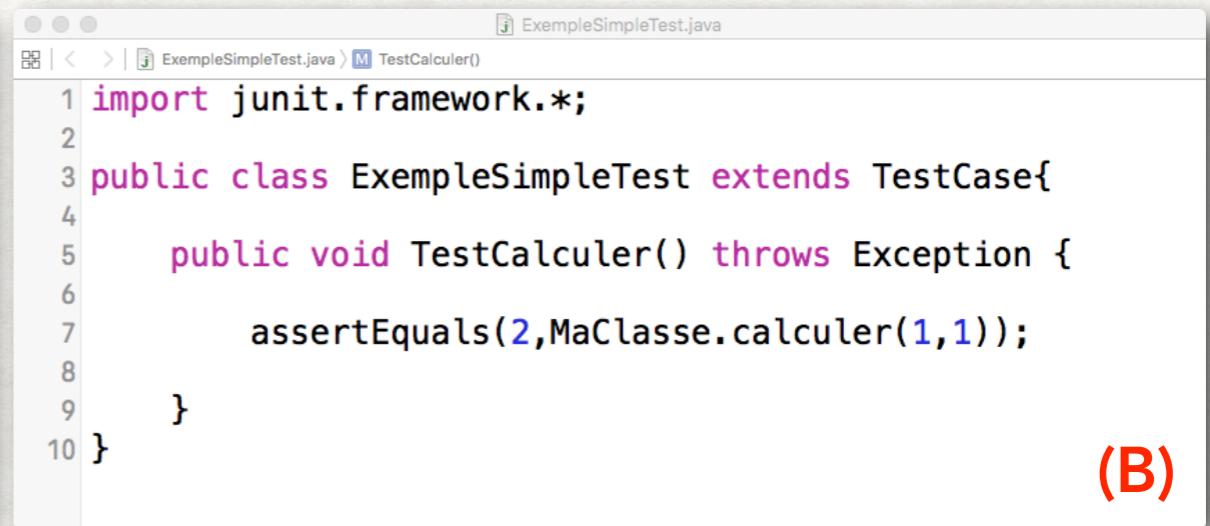
TROUVEZ L'INTRUS



The screenshot shows a Java code editor window titled "ExempleSimpleTest.java". The code is as follows:

```
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10}
```

(A)



The screenshot shows a Java code editor window titled "ExempleSimpleTest.java". The code is as follows:

```
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void TestCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10}
```

(B)

TROUVEZ L'INTRUS

The image shows two Java code snippets side-by-side. On the left, snippet (A) contains a single test method named `testCalculer`. On the right, snippet (B) contains a test method named `TestCalculer`, which is a misspelling of `testCalculer`.

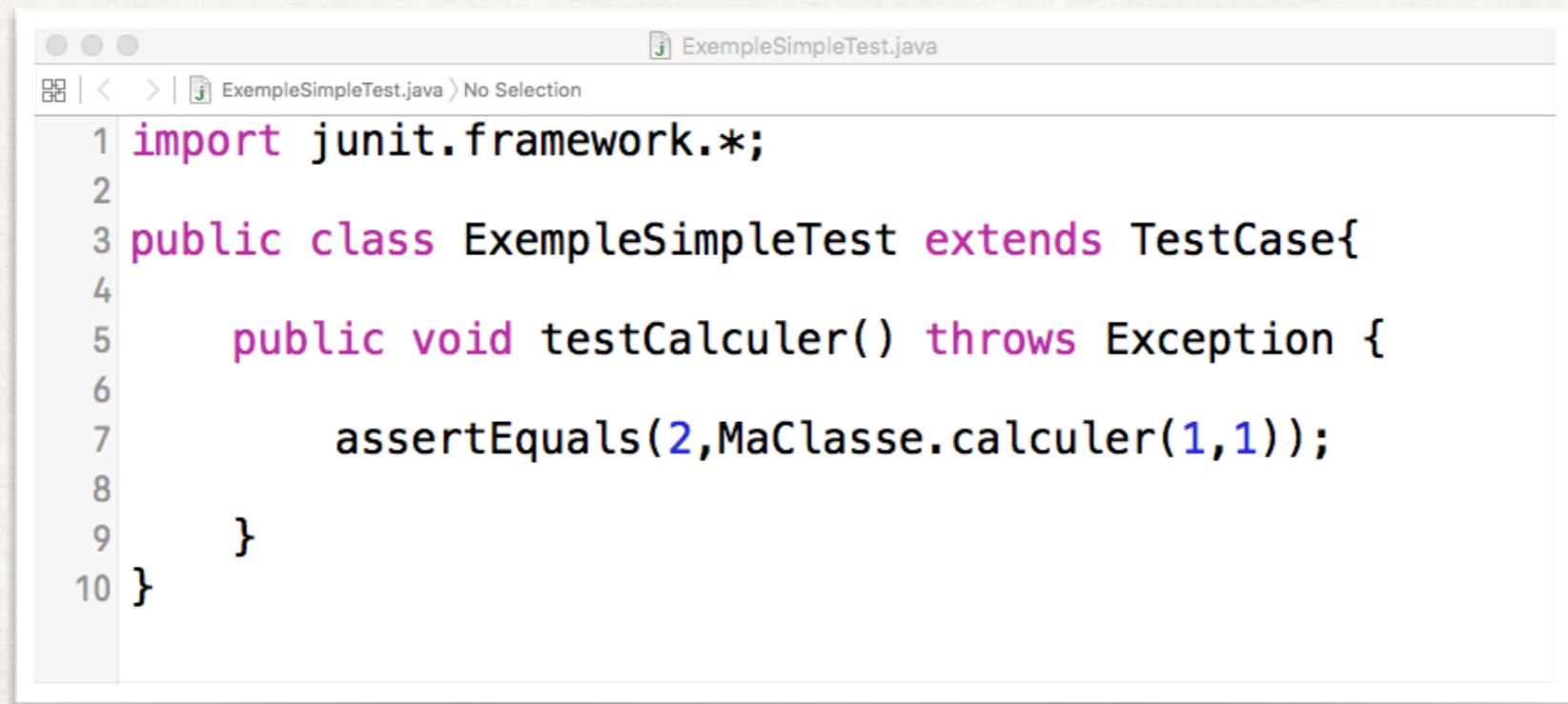
```
ExempleSimpleTest.java (A)
import junit.framework.*;
public class ExempleSimpleTest extends TestCase{
    public void testCalculer() throws Exception {
        assertEquals(2,MaClasse.calculer(1,1));
    }
}
```

```
ExempleSimpleTest.java (B)
import junit.framework.*;
public class ExempleSimpleTest extends TestCase{
    public void TestCalculer() throws Exception {
        assertEquals(2,MaClasse.calculer(1,1));
    }
}
```

A large red arrow points from the misspelling in snippet (B) down to a terminal window below.

```
/java/testjunit>java -cp junit.jar;. junit.textui.TestRunner  
ExempleSimpleTest  
.F  
Time: 0,01  
There was 1 failure:  
1) warning(junit.framework.TestSuite$1) junit.framework.AssertionFailedError: No  
tests found in MaClasseTest  
FAILURES!!!  
Tests run: 1, Failures: 1, Errors: 0
```

CLASSE DE TEST



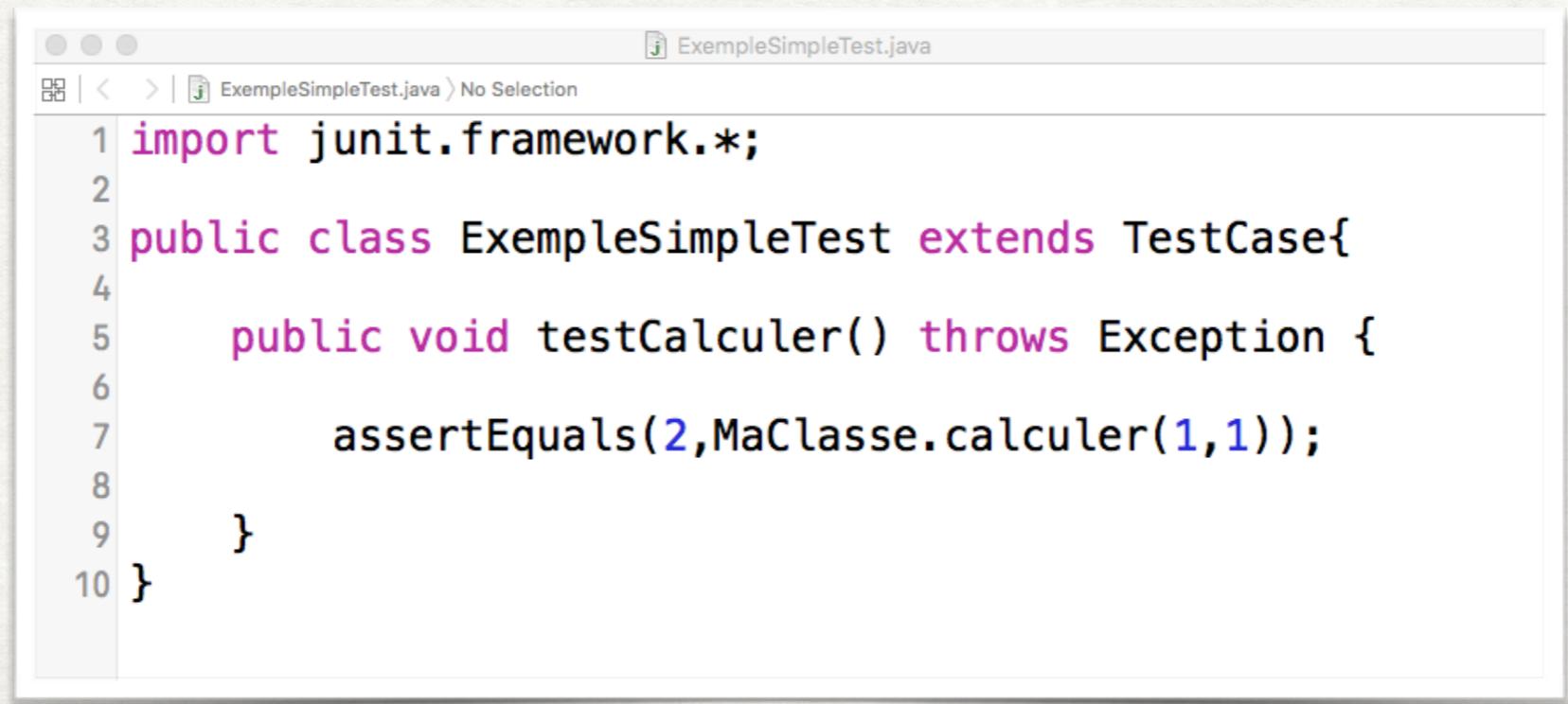
The screenshot shows a Java code editor window with the following details:

- Title Bar:** ExempleSimpleTest.java
- Toolbar:** Includes icons for file operations (New, Open, Save, etc.) and navigation (Back, Forward, Home).
- Code Area:** Displays the following Java code:

```
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10 }
```

CLASSE DE TEST

Dans une classe de test, il faut écrire une méthode dont le nom commence par "**test**" en minuscule suivi du nom du cas de test (généralement le nom de la méthode à tester).

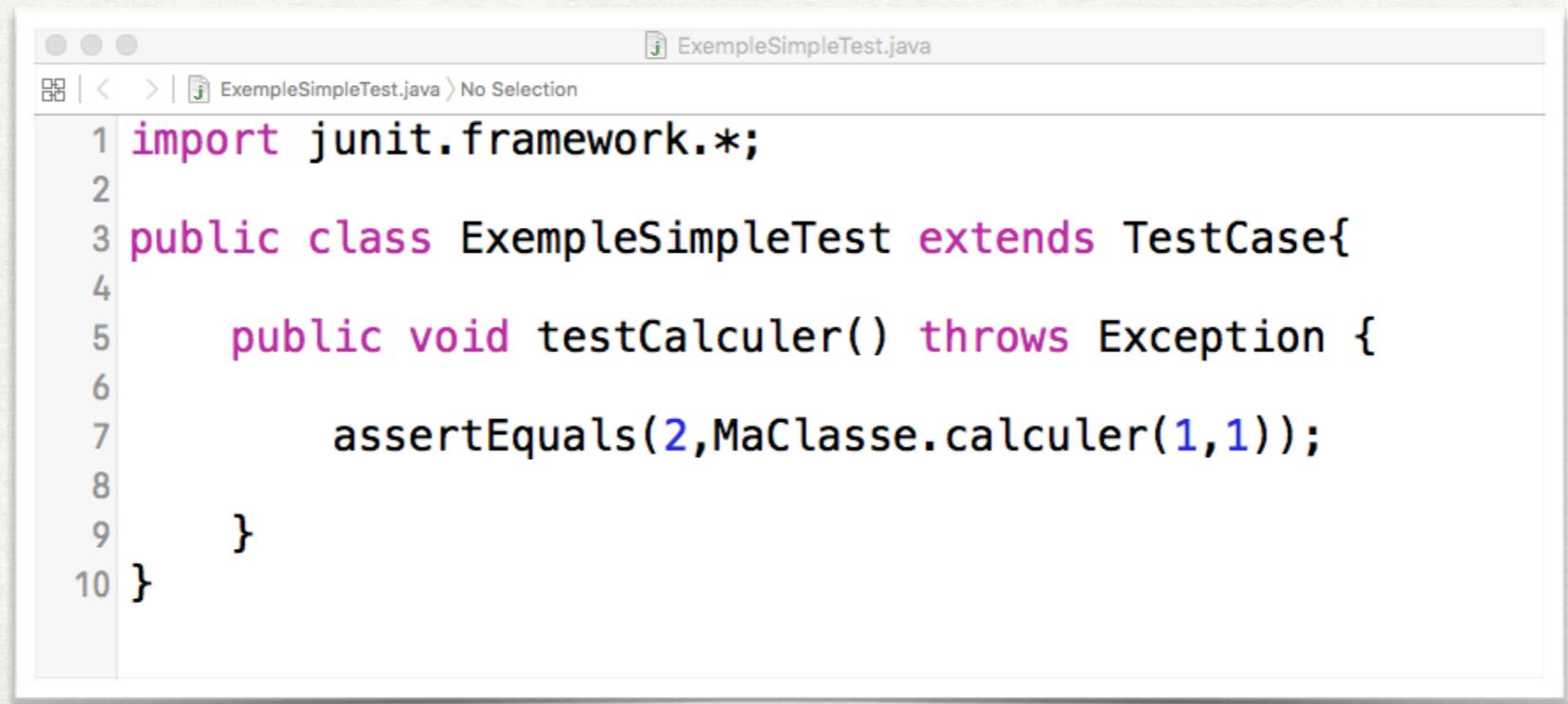


The screenshot shows a Java code editor window with the title bar "ExempleSimpleTest.java". The code editor displays the following Java code:

```
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10}
```

CLASSE DE TEST

Dans une classe de test, il faut écrire une méthode dont le nom commence par "**test**" en minuscule suivi du nom du cas de test (généralement le nom de la méthode à tester).



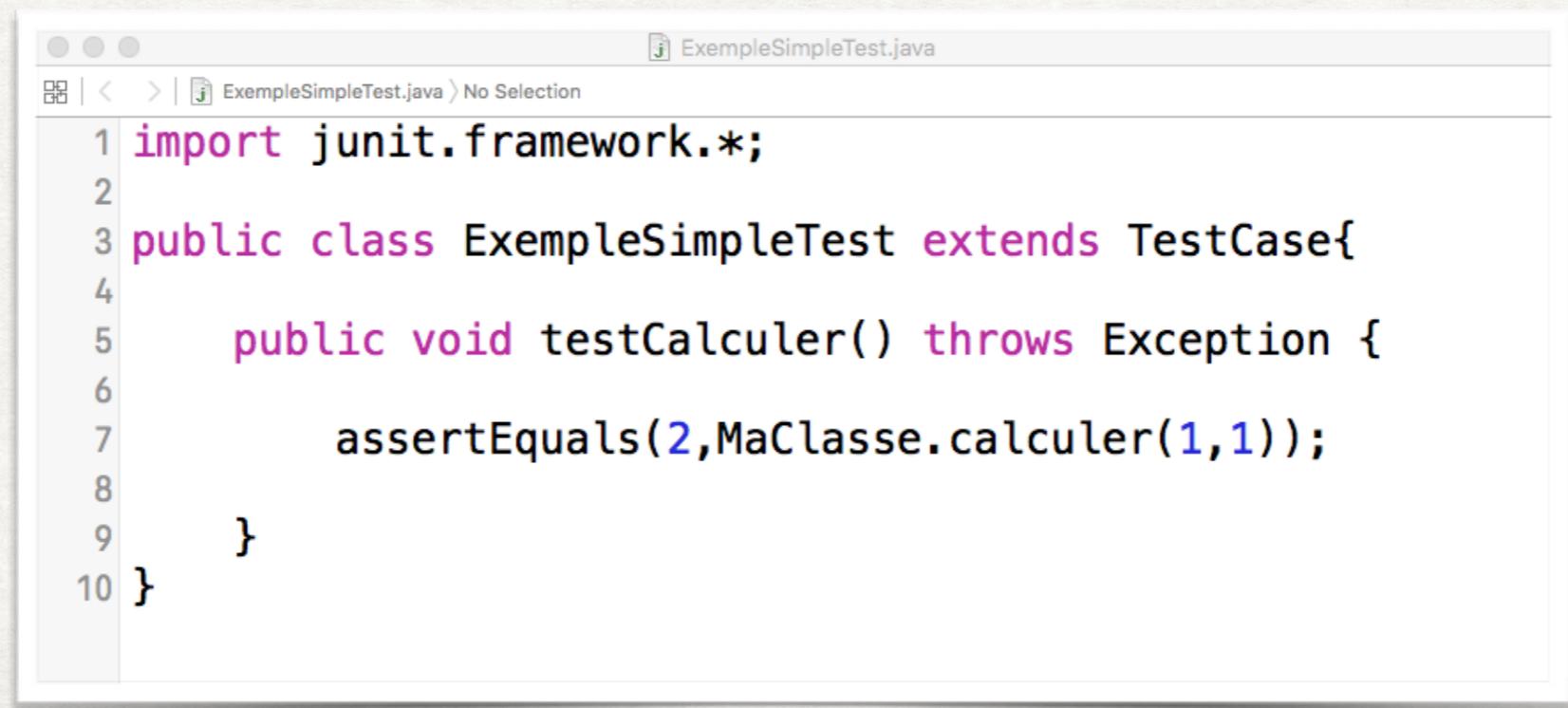
The screenshot shows a Java code editor window with the title bar "ExempleSimpleTest.java". The code editor displays the following Java code:

```
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10}
```

CLASSE DE TEST

Dans une classe de test, il faut écrire une méthode dont le nom commence par "**test**" en minuscule suivi du nom du cas de test (généralement le nom de la méthode à tester).

Chacune de ces méthodes doit avoir les caractéristiques suivantes :



The screenshot shows a Java code editor window with the title bar "ExempleSimpleTest.java". The code editor displays the following Java code:

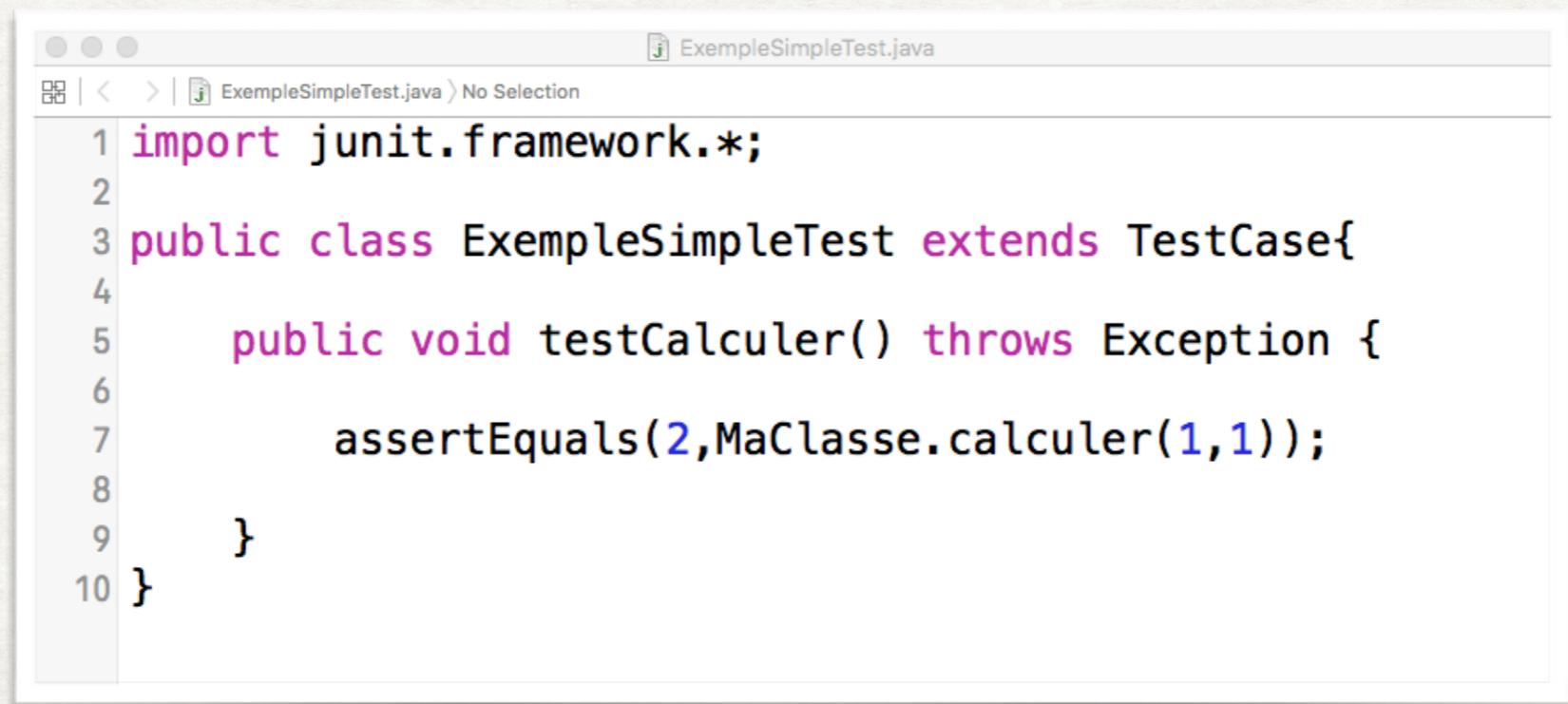
```
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10}
```

CLASSE DE TEST

Dans une classe de test, il faut écrire une méthode dont le nom commence par "**test**" en minuscule suivi du nom du cas de test (généralement le nom de la méthode à tester).

Chacune de ces méthodes doit avoir les caractéristiques suivantes :

- elle doit être déclarée public



The screenshot shows a Java code editor window with the title bar "ExempleSimpleTest.java". The code area contains the following Java code:

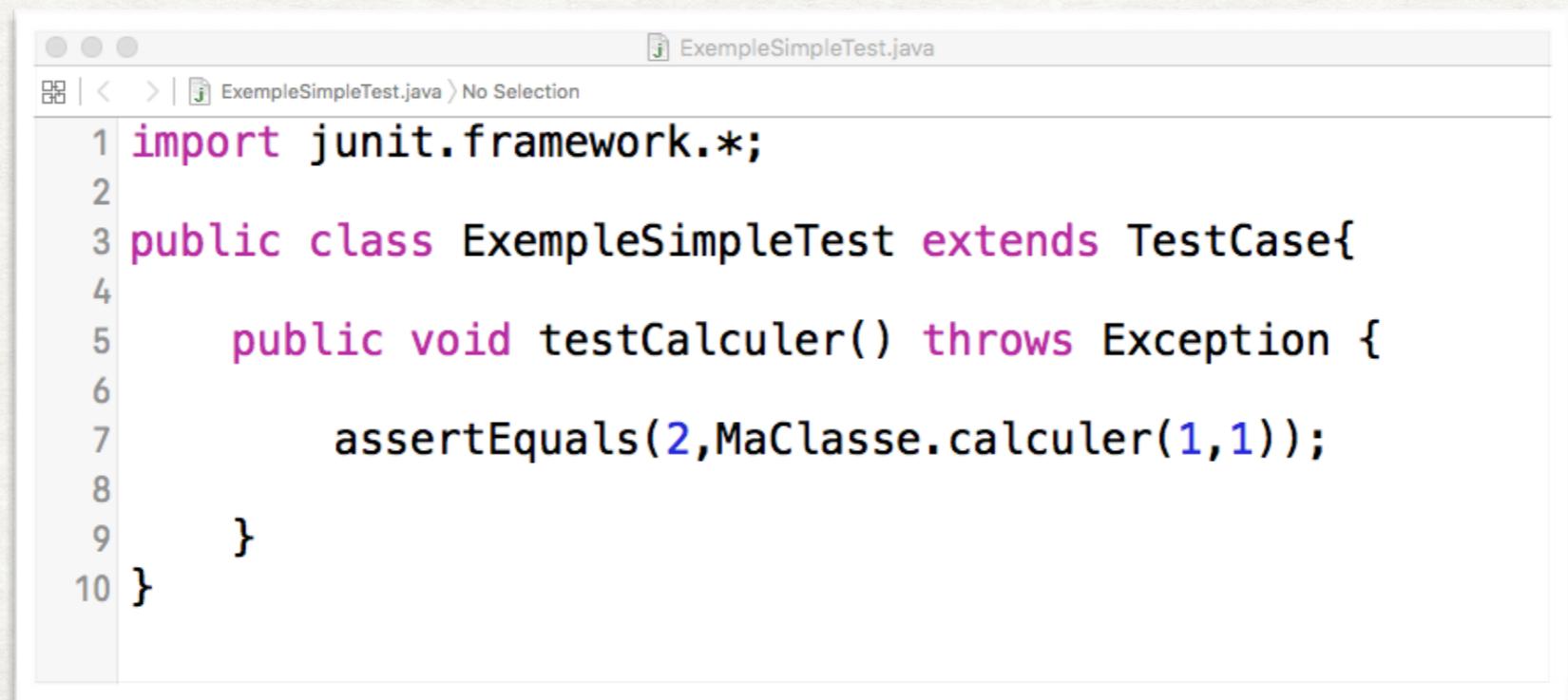
```
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10}
```

CLASSE DE TEST

Dans une classe de test, il faut écrire une méthode dont le nom commence par "**test**" en minuscule suivi du nom du cas de test (généralement le nom de la méthode à tester).

Chacune de ces méthodes doit avoir les caractéristiques suivantes :

- elle doit être déclarée public
- elle ne doit renvoyer aucune valeur



The screenshot shows a Java code editor window with the following details:

- Title Bar:** The title bar displays "ExempleSimpleTest.java".
- Toolbar:** A standard toolbar with icons for file operations like new, open, save, and cut/paste.
- Code Area:** The main area contains the following Java code:

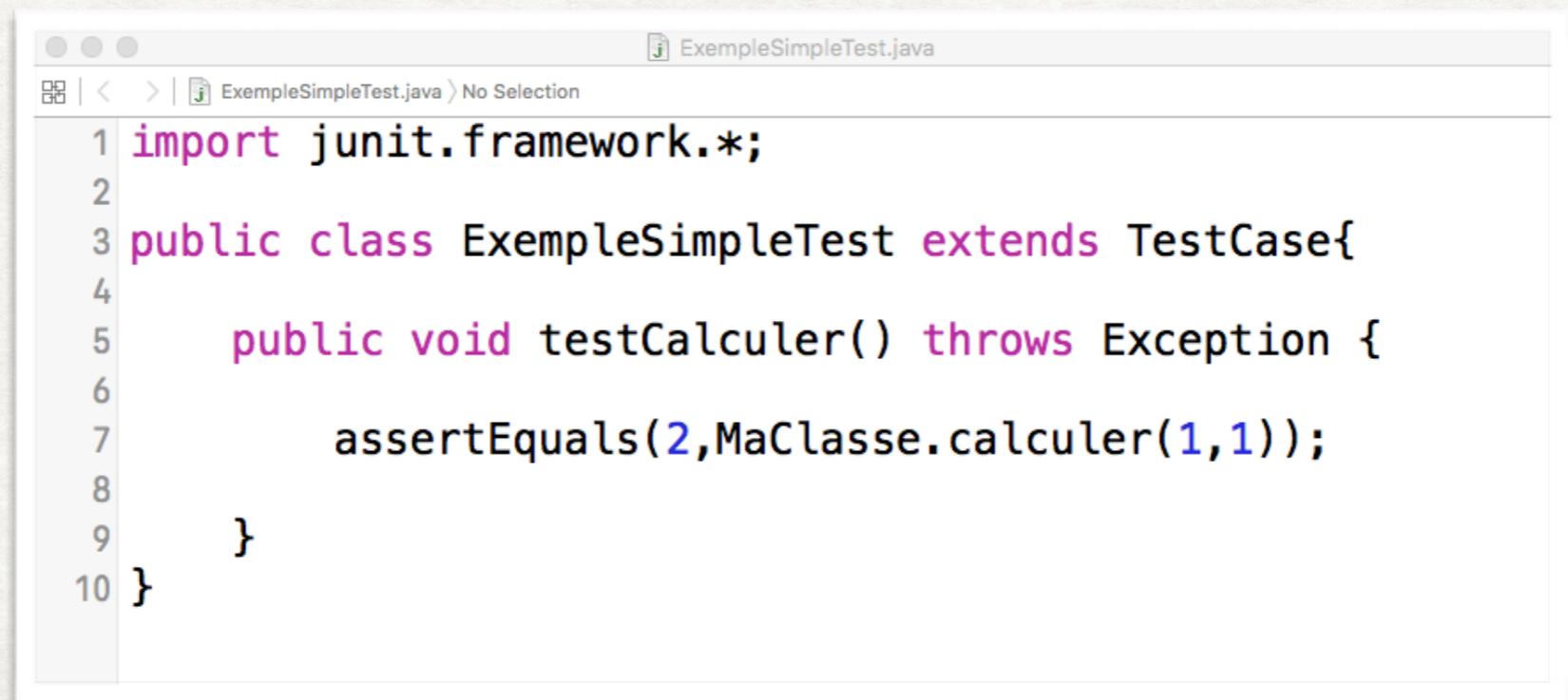
```
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10 }
```

CLASSE DE TEST

Dans une classe de test, il faut écrire une méthode dont le nom commence par "**test**" en minuscule suivi du nom du cas de test (généralement le nom de la méthode à tester).

Chacune de ces méthodes doit avoir les caractéristiques suivantes :

- elle doit être déclarée public
- elle ne doit renvoyer aucune valeur
- elle ne doit pas posséder de paramètres



The screenshot shows a Java code editor window with the following details:

- Title Bar:** The title bar displays "ExempleSimpleTest.java".
- Toolbar:** A standard toolbar with icons for file operations like new, open, save, and cut/paste.
- Code Area:** The main area contains the following Java code:

```
1 import junit.framework.*;
2
3 public class ExempleSimpleTest extends TestCase{
4
5     public void testCalculer() throws Exception {
6
7         assertEquals(2,MaClasse.calculer(1,1));
8
9     }
10 }
```

L'ÉCRITURE DES CAS DE TESTS

L'ÉCRITURE DES CAS DE TESTS

Chaque méthode de test contient généralement des traitements en trois étapes :

L'ÉCRITURE DES CAS DE TESTS

Chaque méthode de test contient généralement des traitements en trois étapes :

L'ÉCRITURE DES CAS DE TESTS

Chaque méthode de test contient généralement des traitements en trois étapes :

- Instanciation des objets requis

L'ÉCRITURE DES CAS DE TESTS

Chaque méthode de test contient généralement des traitements en trois étapes :

- Instanciation des objets requis
- Invocation des traitements sur les objets

L'ÉCRITURE DES CAS DE TESTS

Chaque méthode de test contient généralement des traitements en trois étapes :

- Instanciation des objets requis
- Invocation des traitements sur les objets
- Vérification des résultats des traitements

L'ÉCRITURE DES CAS DE TESTS

Chaque méthode de test contient généralement des traitements en trois étapes :

- Instanciation des objets requis
- Invocation des traitements sur les objets
- Vérification des résultats des traitements

JUnit ne garantit pas l'ordre d'exécution des cas de tests puisque ceux-ci sont obtenus par introspection

LES ASSERTIONS

LES ASSERTIONS

assertEquals()

Vérifier l'égalité de deux valeurs de type primitif ou objet (en utilisant la méthode equals())

Il existe de nombreuses surcharges de cette méthode :

- pour chaque type primitif,
- pour un objet de type Object
- pour un objet de type String

LES ASSERTIONS

LES ASSERTIONS

assertFalse() : Vérifier que la valeur fournie en paramètre est fausse

LES ASSERTIONS

assertFalse() : Vérifier que la valeur fournie en paramètre est fausse

assertTrue() : Vérifier que la valeur fournie en paramètre est vraie

LES ASSERTIONS

assertFalse() : Vérifier que la valeur fournie en paramètre est fausse

assertTrue() : Vérifier que la valeur fournie en paramètre est vraie

assertNull() : Vérifier que l'objet fourni en paramètre soit null

LES ASSERTIONS

assertFalse() : Vérifier que la valeur fournie en paramètre est fausse

assertTrue() : Vérifier que la valeur fournie en paramètre est vraie

assertNull() : Vérifier que l'objet fourni en paramètre soit null

assertNotNull() : Vérifier que l'objet fourni en paramètre ne soit pas null

LES ASSERTIONS

assertFalse() : Vérifier que la valeur fournie en paramètre est fausse

assertTrue() : Vérifier que la valeur fournie en paramètre est vraie

assertNull() : Vérifier que l'objet fourni en paramètre soit null

assertNotNull() : Vérifier que l'objet fourni en paramètre ne soit pas null

fail() : Force le cas de test à échouer

LES ASSERTIONS

LES ASSERTIONS

assertSame() :

LES ASSERTIONS

assertSame() :

Vérifier que les deux objets fournis en paramètre font référence à la même entité

LES ASSERTIONS

assertSame() :

Vérifier que les deux objets fournis en paramètre font référence à la même entité

LES ASSERTIONS

assertSame() :

Vérifier que les deux objets fournis en paramètre font référence à la même entité

Exemples identiques :

LES ASSERTIONS

assertSame() :

Vérifier que les deux objets fournis en paramètre font référence à la même entité

Exemples identiques :

```
assertSame("Les deux objets sont identiques", obj1, obj2);
```

LES ASSERTIONS

assertSame() :

Vérifier que les deux objets fournis en paramètre font référence à la même entité

Exemples identiques :

```
assertSame("Les deux objets sont identiques", obj1, obj2);  
assertTrue("Les deux objets sont identiques ", obj1 == obj2);
```

LES ASSERTIONS

assertSame() :

Vérifier que les deux objets fournis en paramètre font référence à la même entité

Exemples identiques :

```
assertSame("Les deux objets sont identiques", obj1, obj2);  
assertTrue("Les deux objets sont identiques ", obj1 == obj2);
```

LES ASSERTIONS

assertSame() :

Vérifier que les deux objets fournis en paramètre font référence à la même entité

Exemples identiques :

```
assertSame("Les deux objets sont identiques", obj1, obj2);  
assertTrue("Les deux objets sont identiques ", obj1 == obj2);
```

assertNotSame() :

LES ASSERTIONS

assertSame() :

Vérifier que les deux objets fournis en paramètre font référence à la même entité

Exemples identiques :

```
assertSame("Les deux objets sont identiques", obj1, obj2);  
assertTrue("Les deux objets sont identiques ", obj1 == obj2);
```

assertNotSame() :

Vérifier que les deux objets fournis en paramètre ne font pas référence à la même entité

LES ASSERTIONS

assertSame() :

Vérifier que les deux objets fournis en paramètre font référence à la même entité

Exemples identiques :

```
assertSame("Les deux objets sont identiques", obj1, obj2);  
assertTrue("Les deux objets sont identiques ", obj1 == obj2);
```

assertNotSame() :

Vérifier que les deux objets fournis en paramètre ne font pas référence à la même entité

TESTER UNE EXCEPTION

TESTER UNE EXCEPTION

```
public class MaClasse{  
  
    private int a;  
    private int b;  
  
    public MaClasse(int a, int b) {  
        this.a = a;  
        this.b = b;  
    }  
  
    public int sommer() throws IllegalStateException {  
        if ((a == 0) && (b==0)) {  
            throw new IllegalStateException("Les deux valeurs sont nulles");  
        }  
        return a+b;  
    }  
  
    public void setA(int unA) {  
        this.a = unA;  
    }  
    public void setB(int unB) {  
        this.b = unB;  
    }  
}
```

TESTER UNE EXCEPTION

```
public class MaClasse{  
  
    private int a;  
    private int b;  
  
    public MaClasse(int a, int b) {  
        this.a = a;  
        this.b = b;  
    }  
  
    public int sommer() throws IllegalStateException {  
        if ((a == 0) && (b==0)) {  
            throw new IllegalStateException("Les deux valeurs sont nulles");  
        }  
        return a+b;  
    }  
  
    public void setA(int unA) {  
        this.a = unA;  
    }  
    public void setB(int unB) {  
        this.b = unB;  
    }  
}
```

```
public class MaClasseTest extends TestCase{  
  
    public void testSommer() throws Exception {  
        MaClasse mc = new MaClasse(0,0);  
        mc.sommer();  
    }  
}
```

TESTER UNE EXCEPTION

```
public class MaClasse{
```

```
    private int a;  
    private int b;
```

```
    public MaClasse(int a, int b) {  
        this.a = a;  
        this.b = b;  
    }
```

```
    public int sommer() throws IllegalStateException {  
        if ((a == 0) && (b==0)) {  
            throw new IllegalStateException("Les deux valeurs sont nulles");  
        }  
        return a+b;  
    }
```

```
    public void setA(int unA) {  
        this.a = unA;  
    }
```

```
java -cp junit.jar;. junit.textui.TestRunner MaClasse2Test  
.E  
}  
Time: 0,01  
There was 1 error:  
1) testSommer(MaClasse2Test)java.lang.IllegalStateException: Les deux valeurs sont  
nulles.  
...  
FAILURES!!!  
Tests run: 2, Failures
```

```
public class MaClasseTest extends TestCase{
```

```
    public void testSommer() throws Exception {  
        MaClasse mc = new MaClasse(0,0);  
        mc.sommer();
```

TESTER UNE EXCEPTION

TESTER UNE EXCEPTION

```
public class MaClasse{  
  
    private int a;  
    private int b;  
  
    public MaClasse(int a, int b) {  
        this.a = a;  
        this.b = b;  
    }  
  
    public int sommer() throws IllegalStateException {  
        if ((a == 0) && (b==0)) {  
            throw new IllegalStateException("Les deux valeurs sont nulles");  
        }  
        return a+b;  
    }  
  
    public void setA(int unA) {  
        this.a = unA;  
    }  
    public void setB(int unB) {  
        this.b = unB;  
    }  
}
```

TESTER UNE EXCEPTION

```
public class MaClasse{  
  
    private int a;  
    private int b;  
  
    public MaClasse(int a, int b) {  
        this.a = a;  
        this.b = b;  
    }  
  
    public int sommer() throws IllegalStateException {  
        if ((a == 0) && (b==0)) {  
            throw new IllegalStateException("Les deux valeurs sont nulles");  
        }  
        return a+b;  
    }  
  
    public void setA(int unA) {  
        this.a = unA;  
    }  
    public void setB(int unB) {  
        this.b = unB;  
    }  
}  
  
public class MaClasseTest extends TestCase{  
    public void testSommer() throws Exception {  
        MaClasse mc = new MaClasse(1,1);  
  
        // cas de test 1  
        assertEquals(2,mc.sommer());  
  
        //cas de test 2  
        try {  
            mc.setA(0);  
            mc.setB(0);  
            mc.sommer();  
            fail("Une exception de type IllegalStateException  
                  aurait du etre levee");  
        } catch (IllegalStateException ise) {  
        }  
    }  
}
```