12. Tests unitaires

Test unitaire : Test d'une unité ou d’un module

Ex: Test du bon fonctionnement d’une méthode d’une classe

⇨ JUnit en java

Plusieurs tests unitaires possibles pour une même méthode

Classe à tester

```
public class Calculette {
    public int add (int a, int b)
            { return a+b; }
    public int subtract (int a, int b)
            { return a*b;  }
}
```

Classe de test

```
public class CalculetteTest {
        private Calculette aTester;
        @Before
        public void setUp( ) throws Exception {
                aTester= new Calculette(); }
        @Test
        public void testAdd( ) {
                Assert.assertEquals(20, aTester.add(15,5));}
        @Test
        public void testSubtract( ) {
                Assert.assertEquals(10, aTester.subtract(30,20));}
}
```

Méthodes (et annotations) possibles dans la JUnit

@BeforeClass

public static void ***setUpBeforeClass***() throws Exception { … }

↳ *méthode exécutée une seule fois avant le début de tous les tests*

@AfterClass

public static void ***tearDownAfterClass***() throws Exception { … }

↳ *méthode exécutée une seule fois après la fin de tous les tests*

@Before

public void ***setUp***() throws Exception { … }

↳ *méthode exécutée avant chaque test (prépare l'environnement de tests)*

@After

public void ***tearDown***() throws Exception { … }

↳ *méthode exécutée après chaque test*

Méthodes (et annotations) possibles dans la Junit

@Test

public void test…() { … }

↳ *méthode de test*

N.B: **@Test (expected = ExceptionAttendue.class)**

↳ *le test échoue si l'exception attendue n'est pas retournée*

@Test (timeout = 100)

↳ *le test échoue si l'exécution de la méthode test prend plus de 100 millisecondes*

@Ignore

public void test…() { … }

↳ *méthode de test est ignorée (ex: si elle est obsolète parce que le code testé a été modifié et que la méthode de test n'a pas encore été adaptée)*

Assert

Assert.**assertEquals** (**int** expected, **int** actual)

Assert.**assertEquals** (**double** expected, **double** actual, double delta)
N.B: Attention avec les réels, jamais vraiment valeurs parfaitement égales, mais on peut tester qu'elles sont égales à une valeur delta près

Assert.**assertEquals** (**String** expected, **String** actual)

Assert.**assertSame** (**Object** expected, **Object** actual)

Assert.**assertNotSame** (**Object** expected, **Object** actual)

Assert.**assertNull** (**Object**)

Assert.**assertNotNull** (**Object**)

Test Suite

Pour lancer une suite de tests (contenus de plusieurs classes de test).
*Exemple : lancer les tests contenus dans la classe **CalculetteTest** et ceux contenus dans la classe **CalculetteBisTest***

```
@RunWith(Suite.class)
@SuiteClasses (value = {CalculetteTest.class,CalculetteBisTest.class})
public class AllTests {
                                    // vide!
}
```

Attention, aucun code à placer dans la classe AllTest :
↯ Tout se fait par annotations !!!