Test Driven Development
JUnit

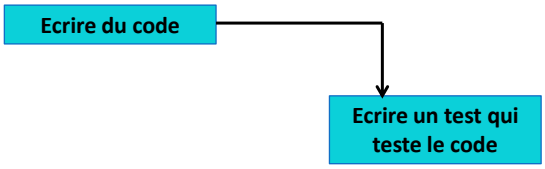
Mme Naoual Berbiche
Enseignant chercheur
Département Informatique- ESTS

Présentation

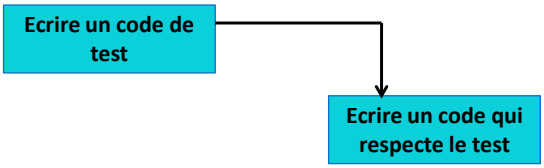
- **JUnit** est un framework open source pour le développement et l'exécution de tests unitaires automatisables pour java.
- Le principal intérêt est de s'assurer que le code répond toujours aux besoins même après d'éventuelles modifications.
- Ce type de tests est appelé **tests unitaires de non régression.**
- **JUnit** a été initialement développé par **Erich Gamma et Kent Beck.**

Méthode TDD

- La **méthode traditionnelle** de la rédaction des tests unitaires consiste à rédiger les tests d'une portion d'un programme afin de vérifier la validité du code implémenté.



La **méthode TDD** quand à elle consiste à rédiger les tests unitaires avant de procéder à la phase de codage.



Cycle de développement

Le cycle de développement préconisé par TDD comporte **cinq étapes** :

1. Ecriture d'un premier test
2. Exécuter le test et vérifier qu'il échoue (car le code qu'il teste n'a pas encore été implémenté)
3. Ecriture du code pour faire passer le test
4. Exécution des tests afin de contrôler que les tests passent et dans ce cas l'implémentation respectera les règles fonctionnelles des tests unitaires
5. Remaniement (Refractor) du code afin d'en améliorer la qualité mais en conservant les mêmes fonctionnalités

Principe de JUnit

JUnit propose :

- Un framework pour le développement des tests unitaires reposant sur **des assertions qui testent les résultats attendus**
- Des applications pour permettre l'exécution des tests et afficher les résultats

- Le but est d'automatiser les tests. Ceux-ci sont exprimés dans des classes sous la forme de cas de tests avec leurs résultats attendus. JUnit exécute ces tests et les comparent avec ces résultats.

Utilisation de JUnit

- Le code de la classe est séparé du code qui permet de la tester
  - Une classe de test par classe de programme
    - Classe Exemple
    - Classe de test ExempleTest
  - Une ou plusieurs méthodes de test par méthode de la classe
    - Méthode methode
    - Méthode de test testMethode
  - Une ou plusieurs assertions par méthode de test
- Fichiers sources des classes de test séparées de ceux des classes du programme

Utilisation de JUnit

- Les cas de tests sont regroupés dans des classes Java qui contiennent une ou plusieurs méthodes de tests.
- Les cas de tests peuvent être exécutés individuellement ou sous la forme de suites de tests.
- JUnit permet le développement incrémental d'une suite de tests.

Avantages

- La rédaction de cas de tests peut avoir un effet immédiat pour détecter des bugs
- Elle a surtout un effet à long terme qui facilite la détection d'effets de bords lors de modifications.

Junit

- eXtreme Programming préconise, entre autre, l'automatisation des tâches de tests unitaires définies avant l'écriture du code. JUnit est particulièrement adapté pour être utilisé avec cette méthode
- Les versions de Junit, utilisées dans ce cours sont:
  - La version 3.8.1
  - La version 4 .
- Le site officiel : **www.junit.org.**
- Pour pouvoir utiliser JUnit, il faut ajouter le fichier junit.jar au classpath.
- Il est intégré à Eclipse

Classe de test

JUnit 3.8

- Package junit.framework.* ;
- public class ExempleTest extends TestCase

JUnit 4

- Pas de classe à étendre

Méthode de test

JUnit 3.8

```
public void testMethode()
```

JUnit 4

- Package org.junit.* ;
- Utilisation de l'annotation @Test
-

```
@Test
public void testMethode()
```


Assertion

- Avec JUnit, la plus petite unité de tests est l'assertion dont le résultat de l'expression booléenne indique un succès ou une erreur.
- Les cas de tests utilisent des affirmations (assertion en anglais) sous la forme de méthodes nommées assertXXX() proposées par le framework.
- Il existe de nombreuses méthodes de ce type qui sont héritées de la classe **junit.framework.Assert**

AssertEquals

test d'égalité

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 et pour un objet de type String

- void assertEquals(Object expected, Object actual)
- void assertEquals(String message, Objectexpected, Object actual)
- void assertEquals(long expected, long actual)
- void assertEquals(String message, long expected,long actual)
- void assertEquals(double expected, double actual, double delta)
- void assertEquals(String message, double expected, double actual, double delta)

AssertTrue et AssertFalse

test de condition

Vérifier que la valeur fournie en paramètre est vraie. Vérifier que la valeur fournie en paramètre est fausse

- void assertTrue(boolean condition)
- void assertTrue(String message, boolean condition)
- void assertFalse(boolean condition)
- void assertFalse(String message, boolean condition)

AssertNull et AssertNotNull

Teste si un objet est null ou pas

Vérifier que l'objet fourni en paramètre soit null

Vérifier que l'objet fourni en paramètre ne soit pas null

- void assertNull(Object object)
- void assertNull(String message, Object object)
- void assertNotNull(Object object)
- void assertNotNull(String message, Object object)

AssertSame et AssertNotSame

Teste si deux objets sont les mêmes ou pas

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);

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

- void assertSame(Object expected, Object actual)
- void assertSame(String message, Object expected, Object actual)
- void assertNotSame(Object unexpected, Object actual)
- void assertNotSame(String message, Object unexpected, Object actual)

Autres

- assertArrayEquals : tests d'égalité entre deux tableaux (byte, char, int, long, short ou Object)
- assertThat : tests plus complexes
- Fail(message): échec du test (levée d'exception)

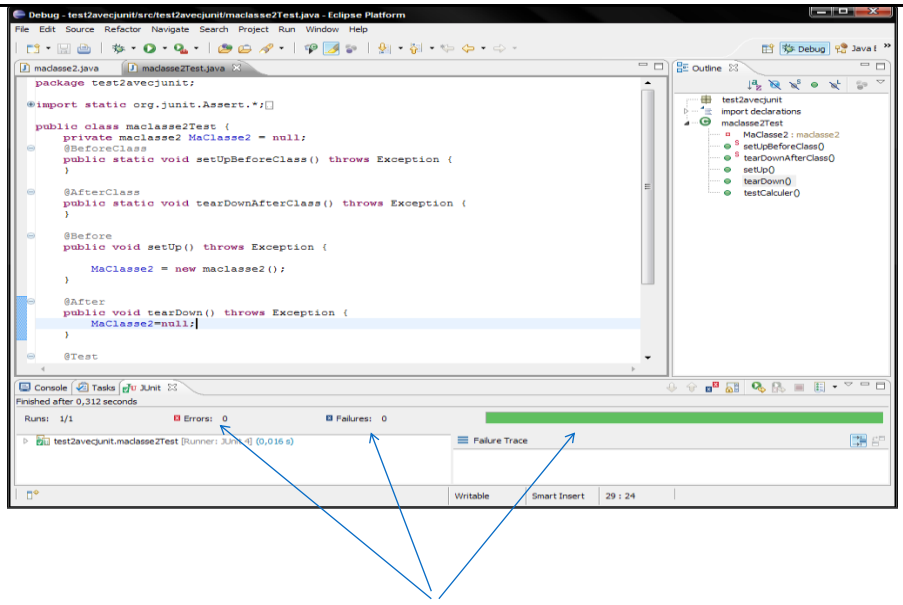
Exemple

```
package test2avecjunit;
import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class maclasse2Test {
        private maclasse2 MaClasse2 = null;
        @Before
        public void setUp() throws Exception {

                MaClasse2 = new maclasse2();
        }
        @After
        public void tearDown() throws Exception {
                MaClasse2=null;
        }
        @Test
        public void testCalculer() {
            assertTrue(MaClasse2.calculer(2,2)==4);
            assertTrue (MaClasse2.calculer(3,1)==4);
                }

}
```



Couleur verte signifie que le test s'est bien passé: 0 Errors et 0 failures

Contexte des tests

- Avec **JUnit**, l'unité de test est une classe dédiée qui regroupe des cas de tests. Ces cas de tests exécutent les tâches suivantes :
  - création d'une instance de la classe et de tout autre objet nécessaire aux tests
  - Méthodes exécutées automatiquement avant et après
  - appel de la méthode à tester avec les paramètres du cas de test
  - comparaison du résultat attendu avec le résultat obtenu : en cas d'échec, une exception est levée

Les Annotations

- Les annotations sont à placer avant les méthodes d'une classe de tests unitaires

JUnit 3.8

void setUp(): méthode exécutée avant chaque méthode testée
void tearDown(): méthode exécutée après l'exécution de chaque méthode testée
@Test
@Ignore

JUnit 4

@Test : méthode de test
@Before : méthode exécutée *avant chaque test*
@After : méthode exécutée *après chaque test*
@BeforeClass : méthode exécutée *avant le premier test*
@AfterClass : méthode exécutée *après le dernier test*
@Ignore : méthode qui ne sera pas lancée comme test

Exemple en Junit 3.8

```
import junit.framework.*;

public class FichierTest {
    File fichier ;
    public void setUp() {
      this.file = new File("toto");
    {
 public void tearDown() {
      this.file.close();
    {
...
{
```

Exemple en Junit 4

```
import org.junit.* ;
    public class FichierTest {
    File fichier ;
    @Before
          public void ouvreFichier() {
              this.file = new File("toto");
          {
    @After
          public void fermeFichier() {
              this.file.close();
          {
    ...
    {
```


Ensemble de tests

- Exécution d'un ensemble de classes de test
- Pour tester tout un programme
- Pour tester certains packages

Ensemble de tests

JUnit 3.8

- Création d'une classe qui regroupe tous les tests (AllTests)
- Implantation d'une méthod public static Test suite() :
- Instanciation d'un object de classe TestSuite
- Ajout des classes de test à exécuter avec la méthode : addTestSuite(Class testClass)

JUnit 4

- Création d'une classe qui regroupe tous les tests (AllTests)
- Annotations

Ensemble de tests

Junit 3.8

```
import junit.framework.*;
    public class AllTests {
        public static Test suite() {
          TestSuite suite = new TestSuite("Tests");
          suite.addTestSuite(Exemple1Test.class);
          suite.addTestSuite(Exemple2Test.class);
          return suite ;
          {
    {
```

Ensemble de tests

Junit 4

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;
   @RunWith(Suite.class)
   @SuiteClasses(value={
         Exemple1Test.class,
         Exemple2Test.class})
         public class AllTests {
   {
```


Appel des tests

en ligne de commande :

- java org.junit.runner.JUnitCore TestClass1 [...other test classes...]

depuis un code Java :

- org.junit.runner.JUnitCore.runClasses(TestClass1.class, ...) ;

depuis Eclipse : Run > Run As > JUnit test

Références

1. ***Développons en Java*** Jean-Michel DOUDOUX.
   http://www.jmdoudoux.fr/java/dej/chap-junit.htm

2. **http://igm.univ-mlv.fr/~dr/XPOSE2009/TDD/pagesHTML/PresentationTDD.html#principes_base**

3. Tests unitaires – Junit - Outils de génie logiciel
   dpt-info.u-strasbg.fr/~blansche/files/ogl_cours_1.pdf

4. **Tests unitaires - Développement dirigé par les tests, Utilisation de Junit -** Gauthier Picard
   École Nationale Supérieure des Mines de Saint-Étienne - gauthier.picard@emse.fr
   *www.emse.fr/~**picard**/cours/2A/**junit/junit**.pdf*