R5.A.08 Qualité de dev.

Cycle de vie de logiciel

Tests, Définitions,Types

plan

- Tests
  - Tests (unitaires, de régression, fonctionnel, de conformité, de sécurité, de robustesse, etc.)
  - Tests en isolation
  - Patron Given When Then
- Tests et gestion de projet
  - CMMI, TMMI,
  - Cycle de vie, agilité
  - spécifications
- Test et environnements
- Tester avec de l'IA ?

Les types de test

- **Phase de Validation de l'implantation :**
  - aussi appelée la phase de Test,
  - permet de vérifier que l'implantation satisfait un certain nombre de propriétés de la spécification. Elle permet donc la détection de défectuosités du fonctionnement de l'implantation du système.
- Différents types de test : conformité, robustesse, performance, interopérabilité, …

Définition du test

- According to the classic definition of Myers "*Software Testing is the process of executing a program or system with the intent of finding errors.*"
-
- According to the definition given by Hetzel, "*Testing involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results*"

Les boites

- Système ou logiciel vu sous forme d'une ou plusieurs boites:
- Boite blanche, noire, grise
- Test en boite blanche: ceux-ci, aussi appelés tests structurels, sont effectués sur des systèmes dont la structure interne est connue et observable.
  - exemple le test de boucle qui vise à valider toutes les boucles d'un programme

Les boites

- Test en boite noire :

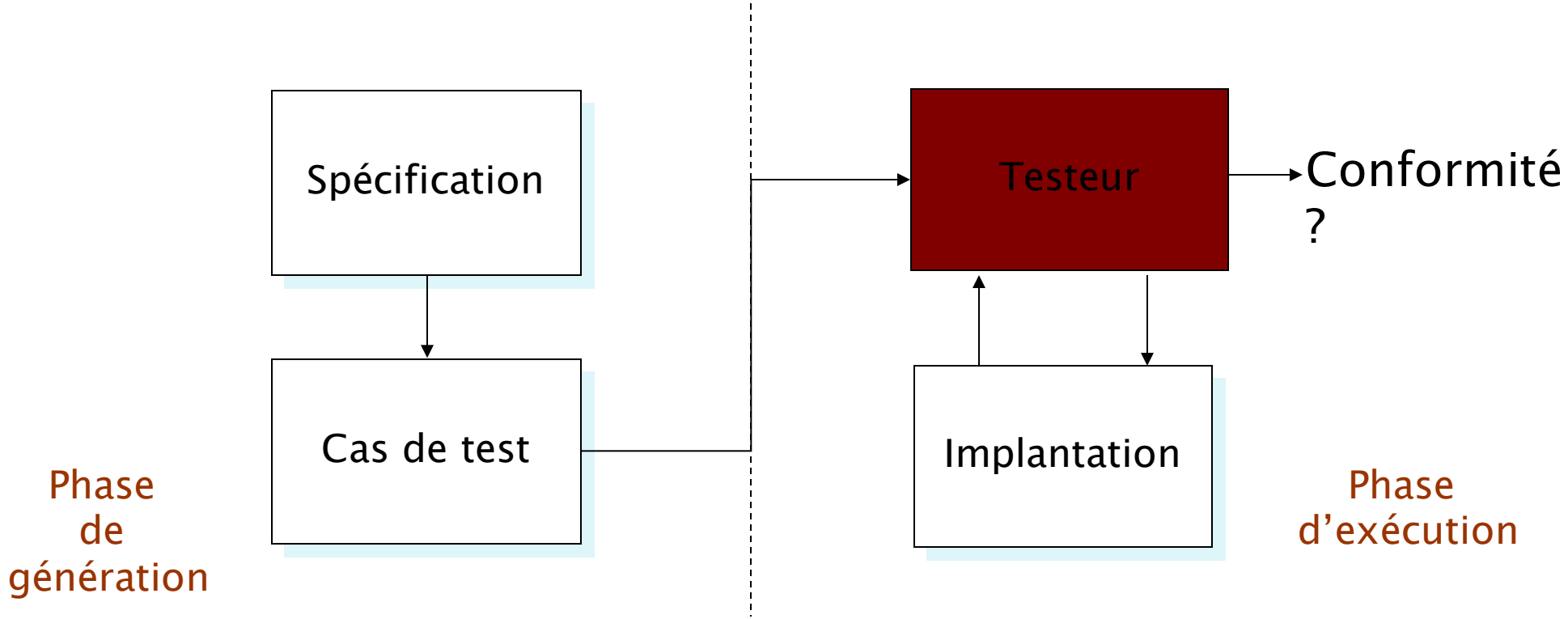
- cette catégorie rassemble les tests, aussi appelés tests fonctionnels, qui sont appliqués sur des systèmes où la structure interne est inconnue.

- Seules les interfaces reliant le système avec l'environnement extérieur sont connues. (PCO)

Les boites

- Test en boite grise
- Connaissance d'une partie du système, de l'environnement,…
  - Ex: accès au serveur web ou est ébergé une appli web
- Connaissance d'hypothèses sur l'application
  - Ex: l'appli web ne prends jamais de mails invalides

Phases de test classique (≠TDD)



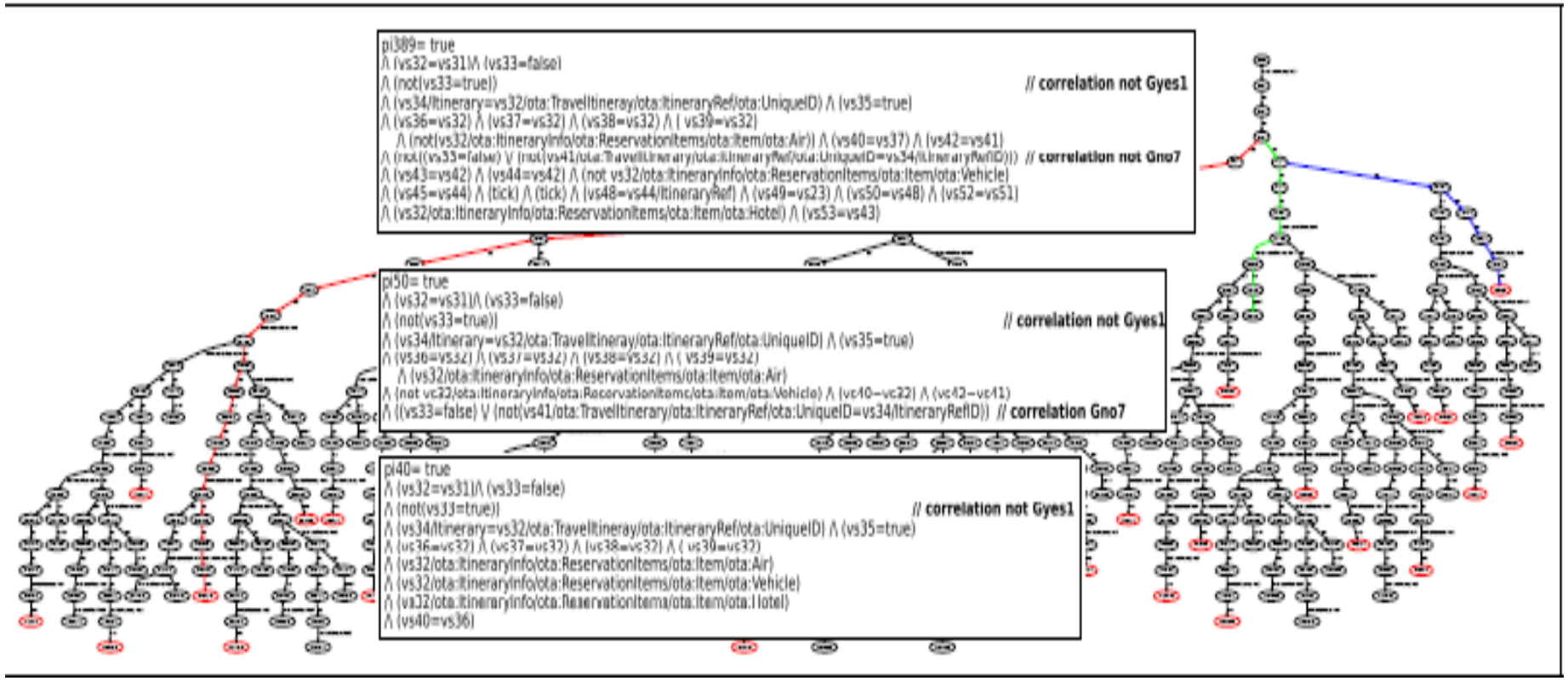
Exemples de Cas de test

- Junit:

```
import static org.junit.Assert.assertEquals;
import org.junit.Test; public class

MyFirstJUnitTest { @Test public void simpleAdd(){ int result = 1; int expected = 1;
   assertEquals(result, expected); } }
```

Exemples de Cas de test



Cas de test

- Verdict:
- PASS:
- FAIL:
- INCONCLUSIVE:

Types de test

- Tests de caractéristiques: conformité, robustesse, interopérabilité, etc.
- Tests de granularité: nous groupons les tests que l'on trouve souvent en entreprise et qui expriment la granularité utilisée par un test
- Test d'accessiblité: rassemble les approches en boite blanche, grise et noire.

Exemples de types de test

- Tests de l'usager
  - Ces tests sont effectués au niveau de l'usager qui manipule le système pour vérifier si ce dernier répond bien à ses besoins. Ces tests, communément appelés beta-tests, permettent de vérifier les services les plus demandés.

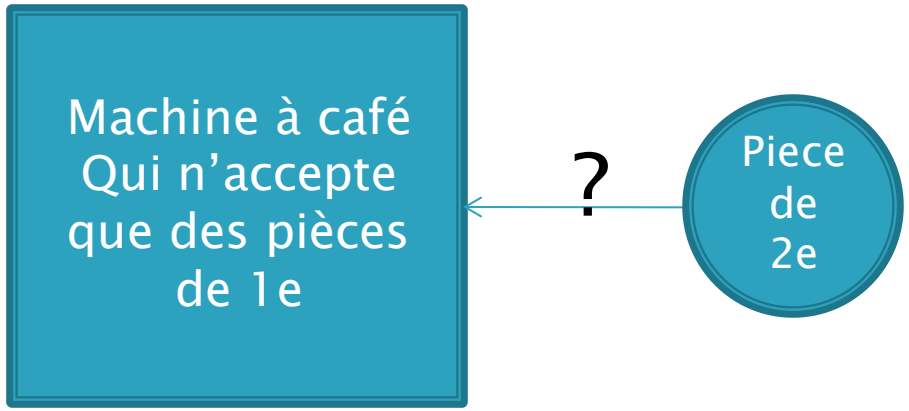
- Tests d'Interopérabilité
  - Ces tests vérifient si le système développé interagit d'une façon correcte avec d'autres systèmes extérieurs en observant les fonctionnements des différents systèmes et des communications engendrées. Ces tests permettent de vérifier un service plus global fourni aux utilisateurs.

Exemples de types de test

- Tests de Robustesse
  - Ces tests consistent à vérifier la réaction d'un système dans des conditions d'utilisations extrêmes ou bien son exécution dans un environnement dit hostile.
  - Ces conditions ne sont généralement pas prévues dans la spécification, cette dernière référençant des conditions de fonctionnement normales.
  - Ces tests permettent ainsi de vérifier si d'autres erreurs existent telles que des fraudes ou des erreurs d'utilisation du système.
  - En java: Jcrasher

Exemples de types de test

- Tests de Robustesse
- Exemple:



Exemples de types de test

- Test de conformité
  - Cas de test générés à partir de la spécification pour vérifier si le comportement de l'implantation est conforme à celui de la spécification
  - Test en boite noire ou grise
  - Pas d'équivalence: "on peut détecter la présence de fautes pas leur absences"

Test de conformité

- Beaucoup, beaucoup de méthodes
  - Sur divers modèles
  - Par rapport à diverses architectures (syst. , syst. Distribué, temps réel, stockastique, services,…)
  - Test unitaire, de non regression, fonctionnel (voir plus loin)

Test de conformité

- Test basé modèle:
  - spécification décrite par langage formel (mathématique),
  - automatisation de la gen. des tests)
  - Relations d'implantations: on donne une équivalence entre une specif. et une boite noire (qu'on ne connait pas)
  - Plusieurs relations plus ou moins fortes (isomorphisme, équivalence de trace, ioco, etc.)

Le test

En entreprise (unitaire, regression, fonctionel, outils, etc.)

Test de conformité d'applications (en entreprise)

- Tests dits de granularité

- Test unitaire
- Test d'intégration
- Test du système
- Test d'acceptation du client
- Test de non régression

Test de conformité d'applications (en entreprise)

- Test unitaire
  - Méthode permettant de s'assurer que la plus petite partie d'une application fonctionne correctement "en isolation".
  - Plus petite partie = classe, une librairie, souvent à définir…
  - Exemples: JUNIT, XXXUnit, mais aussi citrus (ws), etc.

Test de conformité d'applications (en entreprise)

- Test d'integration et test fonctionnel
  - Méthode permettant de vérifier l'aspect fonctionnel, la validation, la performance la sécurité d'un système (ou partie)
  - Fait suite au test unitaire
  - Méthodes top-down ou bottom-up
  - Exemple d'outils:
    - Selenium, robotium, SOAPUI (ws), Zaproxy (sécurité), etc.

Test de conformité d'applications (en entreprise)

- Test de non régression:
  - On teste si la modification de code n'a pas eu d'inpact sur les fonctionnalités existantes
  - On utilise pour cela les tests unitaires existants et on vérifie qu'aucune nouvelle erreur n'est détectée

Test de conformité d'applications (en entreprise)

- Test de non régression:
  - Utilisation d'outils d'intégration continue:

- Ex: Jenkins, gitlab CI/CD

Test de conformité d'applications (en entreprise)

- Test de sécurité

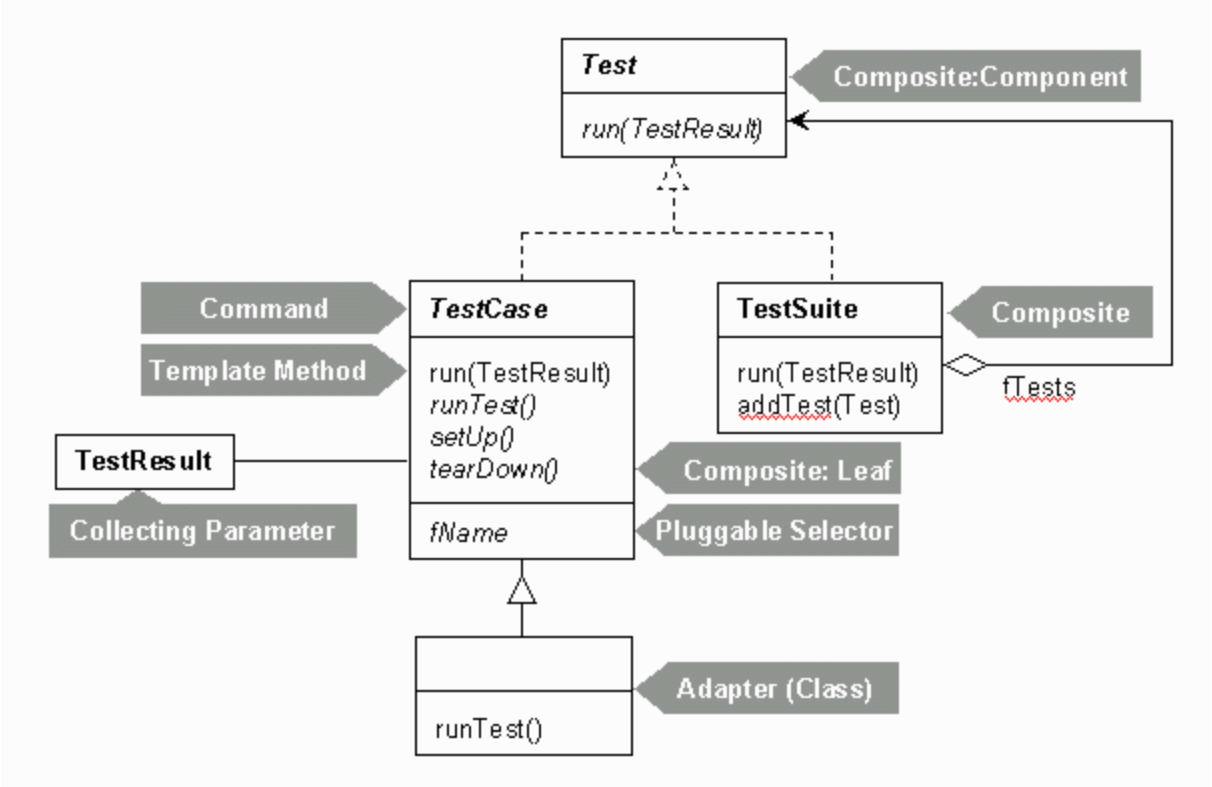
  - Contrôle qu'il n'existe pas de vulnérabilités (de failles)
  - Sécurité réseau, d'accès aux BD, de paramètres de champs (web)
  - Test des log
  - Test de pénétration (buffer overflow, liens symboliques, race condition (pb de concurrence), troyens, …
  - Crack des mots de passe

JUNIT (rappels?)

- (Extrait d'un doc de dvp.com)
- Framework JAVA de rédaction et d'exécution de cas de test (test de conformité unitaire)
- Représentation des tests par une classe à part
- 1 cas de test (test unitaire) = 1 méthode permettant de valider les méthodes de la classe cible

JUNIT

- Construction manuelle des cas de test par rapport à une spécification
- architecture



JUNIT

- Programmeur doit manipuler les classes Testcase et Testsuite

- **TestCase** implémente l'interface **Test** et dérive en outre de la classe **Assert** dont les nombreuses méthodes vous permettront de valider votre code.

Méthode	Rôle
assertEquals	Vérifie que deux objets sont égaux
assertFalse	Vérifie que l'expression est fausse
assertNotNull	Vérifie que l'objet n'est pas nul
assertNotSame	Vérifie que deux références ne sont pas les mêmes
assertNull	Vérifie qu'un objet est nul
assertSame	Vérifie que deux références sont les mêmes
assertTrue	Vérifie que l'expression est vrai
fail	Provoque l'échec du test

JUNIT

Assertion :expression booléenne supposée vraie.

Vérifier au fil du code que tout se passe comme attendu;

Ex:

AssertTrue(maClasse.isVisible());

JUNIT

```
class StandardTests {
@BeforeAll
static void initAll() { }
@BeforeEach
 void init() { }

@Test
void succeedingTest() { }
@Test
void failingTest() { fail("a failing test"); }
@Test @Disabled("for demonstration purposes")
void skippedTest() {  //  not executed }

@AfterEach
void tearDown() { }
@AfterAll
static void tearDownAll() { } }
```

Initialisation 1 fois

Initialisation avant chaque test

libération

JUNIT

Assertions particulières

- Test d'exception :

```
assertThrows(NullPointerException.class,  () -> {
        test.length();
    });
```

- Test timeout

```
assertTimeout(ofMillis(10), () -> { // Simulate task that takes more than 10 ms. Thread.sleep(100);
```

```
assertAll("properties", () -> {
String firstName = person.getFirstName();
assertNotNull(firstName);
// Executed only if the previous assertion is valid.
assertAll("first name",
() -> assertTrue(firstName.startsWith("J")),
() -> assertTrue(firstName.endsWith("e")) ); }
```

JUNIT

Assertions particulières

Ordonancement des tests:

```
class OrderedTestsDemo {
@Test @Order(1)
void nullValues() { // perform assertions against null values }
@Test @Order(2)
void emptyValues() { // perform assertions against empty values }
```

Répéter des tests:

```
@RepeatedTest(10) void repeatedTest() { // ... }
```

JUNIT

- Tests paramétrisés

```
@ParameterizedTest
@ValueSource(ints = { 1, 2, 3 })
void testWithValueSource(int argument) { assertTrue(argument > 0 && argument < 4);
   }
```

JUNIT assertions par contrats (hamcrest)

- Assertions classiques:
  - Pas de messsage explicite de l'echec
  - Pas de réelle combinaison d'assertion possible

- Assertions par contrats => assertThat
  - Facilite la lisibilité:

- Assertions de base:
  - Assert.assertThat("texte", IsSame.sameInstance("texte"));
  - Assert.assertThat("texte", IsEqual.equalTo("texte"));
  - Assert.assertThat("texte", IsInstanceOf.instanceOf(String.class));
  - Assert.assertThat(lTexte, IsNull.nullValue());

  - Assert.assertThat("texte", AnyOf.anyOf(
  - IsInstanceOf.instanceOf(Integer.class), IsEqual.equalTo("Texte")
  -));

JUNIT assertions par contrats

- Messages explicites

```
@Test
public void
testExplicite() {
int
x = 50;
assertThat(x, OrderingComparisons.lessThan(10));
}
```

- Message:

```
java.lang.AssertionError:
Expected: a value less than <10>
got: <50>
at org.junit.Assert.assertThat(Assert.java:750)
at org.junit.Assert.assertThat(Assert.java:709)
at TestNonExplicite.testExplicite(TestNonExplicite.java:29
```

JUNIT

Les suppositions:

Conditions qui doivent être remplies avant test.
Si non, pas d'erreur mais le test s'arrête.

```
final File lFile = new File("fichier.txt");
        Assume.assumeTrue(lFile.exists());
```

Les suppositions:

- `assumeNoException`
- vérifie qu'une opération s'est déroulée sans lever de
- `Throwable`
- `assumeNotNull`
- vérifie qu'aucun paramètre n'est nul
- `assumeThat`
- vérifie qu'une condition par contrat est respectée
- `assumeTrue`
- vérifie que le paramètre est vrai

JUNIT assertions par contrats

- Combinaison de contrats
  - AllOf.allOf(les contrats)
  - AnyOf.anyOf(les contrats)

- Exemple:

```
Assert.assertThat("texte", AnyOf.anyOf(
        IsInstanceOf.instanceOf(Integer.class),     IsEqual.equalTo("Texte")
    ));
```

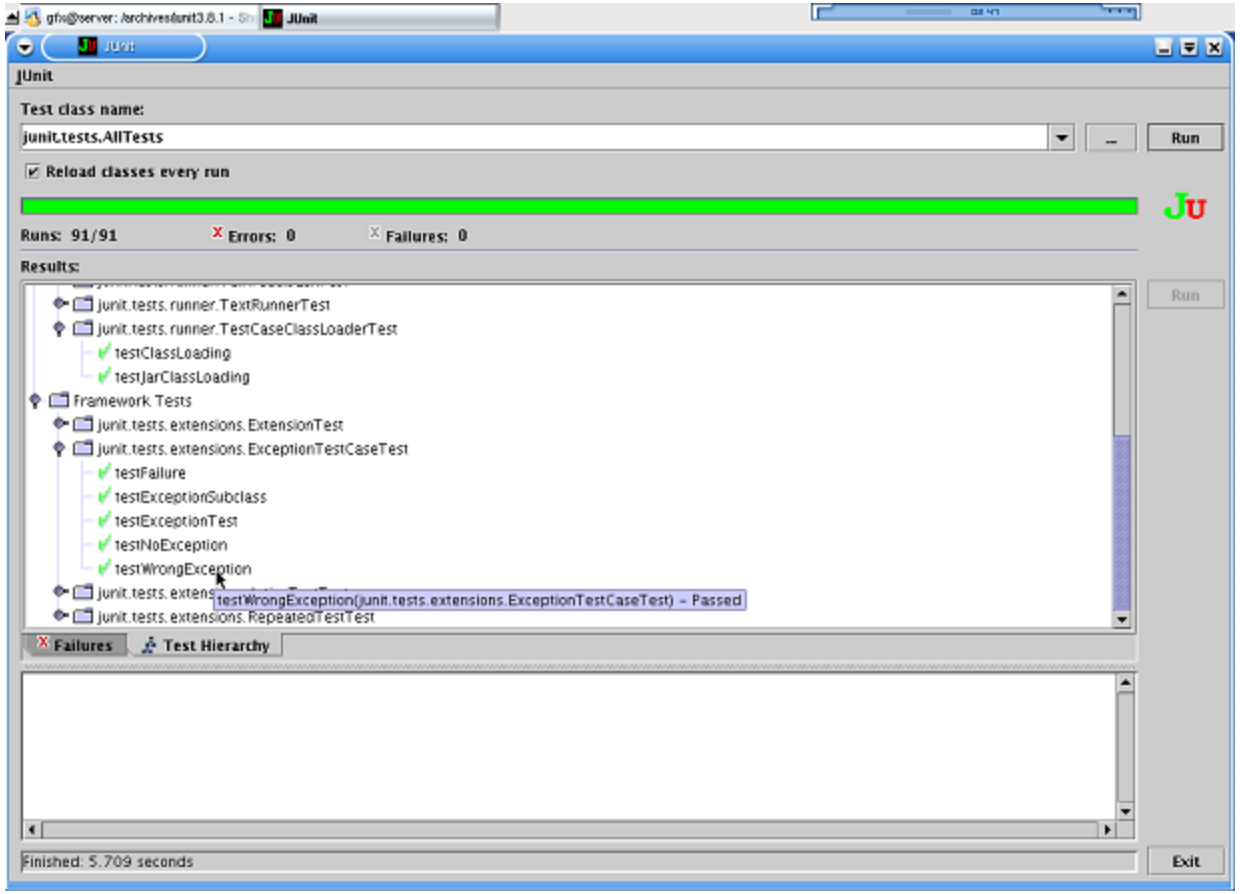
JUNIT assertions par contrats

- Bibliothèque des contrats:

org.hamcrest.beans
org.hamcrest.collection
org.hamcrest.core
org.hamcrest.number
org.hamcrest.object
org.hamcrest.text
org.hamcrest.xml

JUNIT

- Interface graphique



TestNG (testng.org)

- Extension de Junit
  - Permet de créer des groupes de tests, de méthodes
  - Peut // les tests, permettre de tester sans concurrence (Threadsafe)
  - définir un ordre d'exécution des tests ou pas, etc.
  - Configurations de tests,
  - Data driven testing

- avec
  - des annotations (extension de @before et @after)
  - Des nouvelles assertions

TestNG (testng.org)

- Exemple: dépendance de méthodes

```
import org.testng.annotations.Test;

public class App {

        @Test
        public void method1() {
                System.out.println("This  is method 1");
                throw new RuntimeException();
        }

        @Test(dependsOnMethods = { "method1" })
        public void method2() {
                System.out.println("This  is method 2");
        }}
```

Méthode 2 est dépendante de Méthode 1, si le test de méthode 1 est FAIL -> le test de méthode 2 ne se fait pas

PHPUnit

- https://phpunit.de/
- Très proche de Junit dans le fonctionnement.
- Exemple:

```
<?php
use PHPUnit|Framework|TestCase;
class StackTest extends TestCase {
public function test_pass(){
 $boolean = false;
 $this->assertFalse($boolean);
 }
}
```

PHPUnit

- Notion de dependence:

```
/**
    * @depends Previousmethod
    */
```

- Test des sorties

```
$this->expectOutputString('foo');
        print 'foo';
```

- Exceptions:

```
public function testException() {
$this->expectException(InvalidArgumentException::class); $this->
expectExceptionMessage('Mon message');
}
```

PHPUnit

- Fixture

Utiliser les méthodes setUp(), tearDown(), setUpBeforeClass(), tearDownAfterClass() ou les annotations

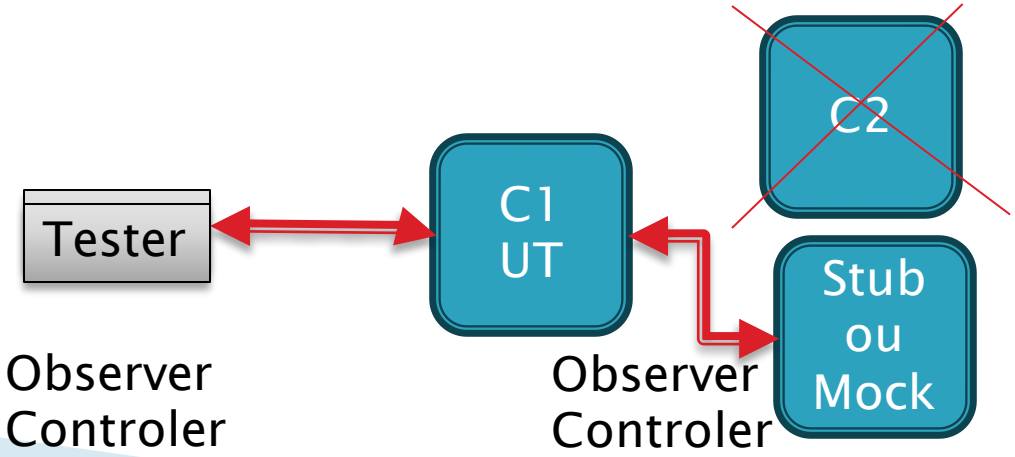
```
/** * @beforeClass
*/
/** * @before
*/
Public function fixture()…
```

- Suites de test:

```
<phpunit>
  <testsuites>
    <testsuite name="money">
      <directory>tests</directory>
// ou <file>tests/test1.php</file>
    </testsuite>
  </testsuites>
</phpunit>
```

Tester en isolation

- Création d'objets simulés lorsque d'autres objets, en plus de ceux testés, sont nécessaires
  - Framework Mockito (google) en Java
  - Intégré dans PHPUnit
  - MockServer (pour simuler des services Web)



Objects simulés

Stub: on crée un objet qui remplace un objet cible et on crée son comportement (appels de méthodes)

Mock : on vérifie en plus comment est utilisé l'objet simulé

Mockito en Java

Création d'instances d'objet « simulées » pour aider au test de classes en isolation

init

```
class Test1 {
@Mock List mockedList;
//List mockedList = mock(List.class);

@BeforeEach
void setUp() throws Exception {
//mock creation
MockitoAnnotations.initMocks(this);
```

implementation

```
//using mock object
mockedList.add("one");
mockedList.clear();
when(mockedList.get(1)).thenReturn("element");
when(mockedList.get(2))
  .thenThrow(new RuntimeException());
}
```

utilisation

```
@Test
void test() {
assertEquals( mockedList.get(1),
"element");
}
@Test
    void exceptionTesting() {
        Exception exception =
assertThrows(RuntimeException.class, ()
->
        mockedList.get(2));

    }
}
```

Mockito

- Injection d'un mock

```
@Mock
private Convertion conv;
//injecte le Mock dans money (par constructeur, par champ etc. selon le nom, voir doc)
@InjectMocks
private Money money;
```

Stub,Mock avec PHPUnit

- Stub:

```
$stub = $this->createStub(SomeClass::class);  // Configure the stub.
$stub->method('doSomething')
        ->willReturn('foo');
$stub->method('doSomething')
         ->will($this->throwException(new Exception));
```

- Mock (simple):

```
$observer = $this->createMock(Observer::class);
$observer->expects($this->once())
            ->method('doSomething')
            ->willReturn('foo');
```

On vérifie que l'appel se fait une seule fois.

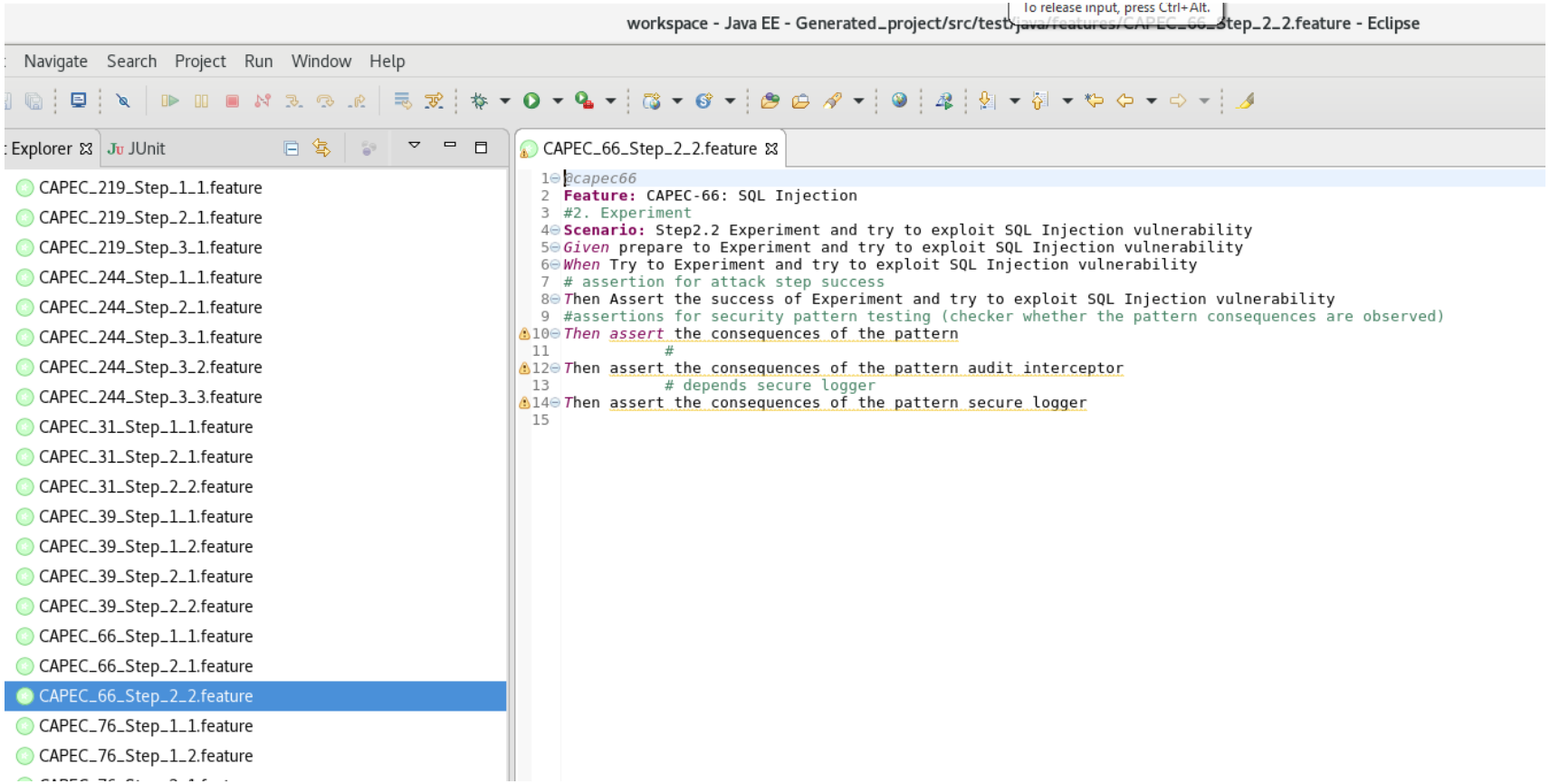
Cucumber

- Tests basés sur des user stories
- Utilise le pattern

Given (précondition)
When (action)
Then (test)

Cucumber

- Exemple (test de sécurité)



Cucumber

- Exemple (test de sécurité)

```
@When("Try to Determine user-controllable input susceptible to injection")
public void trydetermineusercontrollableinputsusceptibletoinjection (){
//  Try one of the following techniques :
//1.  Use web browser to inject input through text fields or through HTTP GET
      parameters.
//2.  Use a web application debugging tool such as Tamper Data, TamperIE,
      WebScarab,etc. to modify HTTP POST parameters, hidden fields, non-freeform
      fields, etc.
//3.  Use XML files to inject input.
//4.  Use network-level packet injection tools such as netcat to inject input
//5.  Use modified client (modified by reverse engineering) to inject input.
List<HarEntry> URLlist = j.getHistory();
for (int i=0 ; i<URLlist.size();i++ ) {
url = j.getHistory().get(i).getRequest().getUrl();
j.scan(url);
int complete = 0;
int scanId = j.getLastScannerScanId();
while (complete < 100) {
complete = j.getScanProgress(scanId);
try {Thread.sleep(1000);} catch (InterruptedException e) { e.printStackTrace();} }}}
```

Cucumber

- Exemple (test de sécurité)

```
@Then("^Input Guard security pattern is present$")
public void Input_Guard_security_pattern_is_present() {
try{
//check that an erroneous input is not propagated
//Recept of empty outputs or outputs showing unauthorised accesses
//Recept of HTTP status showing incorrect, unauthorised accesses
assertThat("Pass_sp",app.getdriver().getPageSource(), anyOf(equals(""),
      containsString( "error"), containsString("forbidden"),
      containsString("unauthorized")));
//HTTP status(503, 408 for quiescent state, the others for Unauthorized accesses)
assertThat("Pass_sp",con.getResponseCode(),anyOf(is(200),is(503),is(408), is(400),
      is(401), is(403), is(405), is(409), is(500));
}catch(Exception e){fail("Inconclusive_sp")}}
```

Test fonctionnel Selenium

- Test fonctionnel
  - Test de l'application complète vue par l'utilisateur, en utilisant les interfaces

- En Web, tests remplissent pages Web et vérification de résultats (Sélenium-> création de suites de test, enregistrement, etc.)

Test fonctionnel Selenium

Exemple :

```
WebDriver driver;
System.setProperty("webdriver.gecko.driver",  "***path***");
driver =new FirefoxDriver();
driver.get("https://duckduckgo.com/");

// Recherche d'un élément ici par son nom,
WebElement element = driver.findElement(By.name( "search_form_input_homepage"));
// On rentre le texte suivant dans l'élément Web
element.sendKeys("Sébastien  Salva");
// On submit le formulaire, Webriver le recherche pour vous
element.submit();
// on affiche le titre de la page
System.out.println("Titre  : " + driver.getTitle());

//Le test ici en junit
Assert.AssertEquals("sebastien  salva at DuckDuckGo",  driver.getTitle())

//Fermeture de Firefox
driver.quit();
```

A voir également

- Arquillian (http://arquillian.org/, redhat): test unitaire, d'intégration, d'acceptation

- Plateforme de test (Java, J2EE) permettant:
  - Lancer des tests dans le vrai conteneur (serveur, BD etc.) et d'y injecter ce que l'on veut (BD, servlet, etc.)
  - De gérer le conteneur, de le lancer, de deployer,etc.
  - De construire un jar de test contenant uniquement les classes à tester via ShrinkWrap

-)

RESTful APIs

Integration, functional testing

- Selenium
- https://rest-assured.io/
- https://github.com/intuit/karate
- https://citrusframework.org/ (integration testing, BD)

Manual testing

- Postman

Performance

- http://jmeter.apache.org/index.html
- https://gettaurus.org/

Mock testing

- https://github.com/mock-server/mockserver
- https://hoverfly.io/