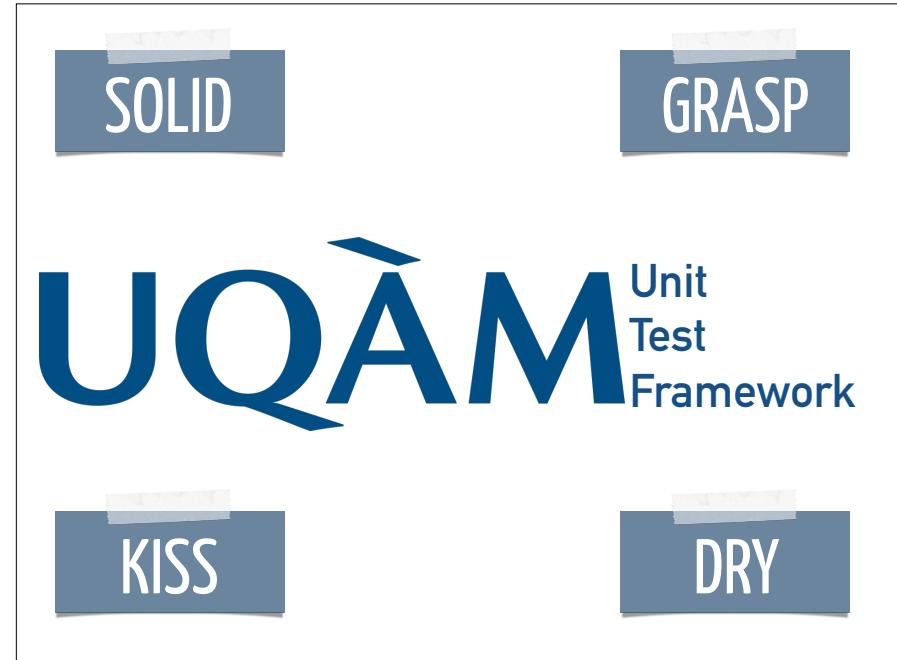


Génie Logiciel : Conception  
Étude de cas JUnit 3.x

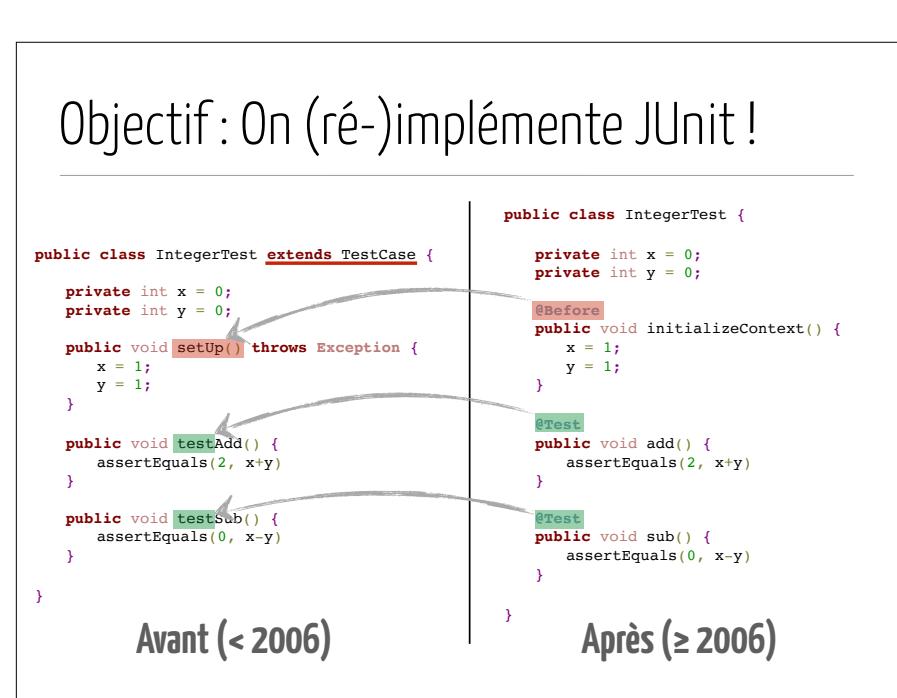
UQÀM | Département d'informatique

Crédit Images: Pixabay & Pexels

Sébastien Mosser  
INF 5153 - Cours #11- A19



		Objectif		
		Création	Structure	Comportement
Portée	classe	Factory	Adapter	Interpreter Template Method
		Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

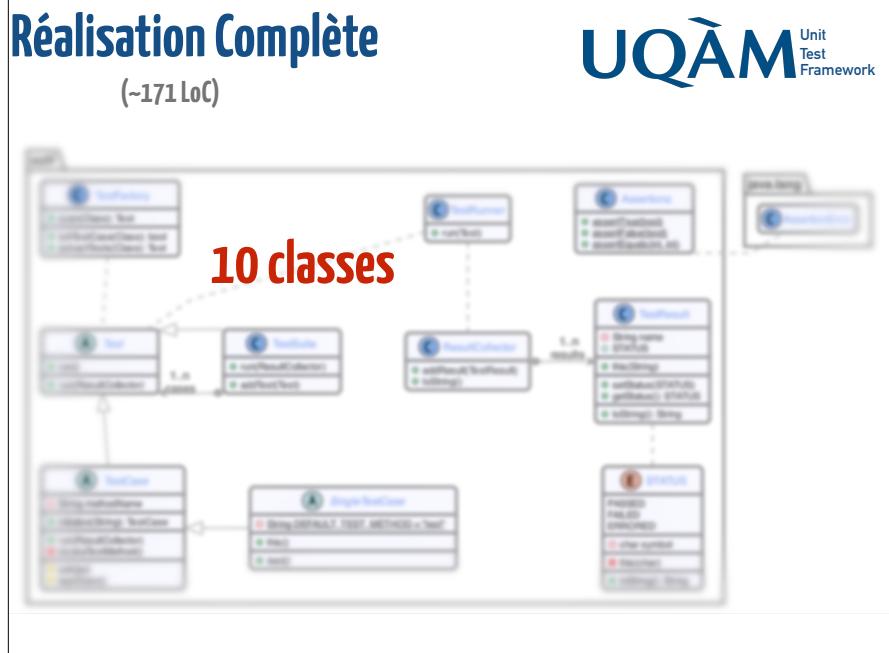


```
public class IntegerTest extends TestCase {  
  
    private int x, y = 0;  
  
    public void setUp() throws Exception {  
        x = 1; y = 1;  
    }  
  
    public void testAdd() {  
        assertEquals(2, x+y)  
    }  
}
```

# JUnit

**UQÀM** Unit Test Framework  
(~171 LoC)

```
public class IntegerTest extends TestCase {  
  
    private int x, y = 0;  
  
    public void setUp() {  
        x = 1; y = 1;  
    }  
  
    public void testAdd() {  
        assertEquals(2, x+y);  
    }  
}
```



# Agilité

(aka “conception en continu”)

# Problème :

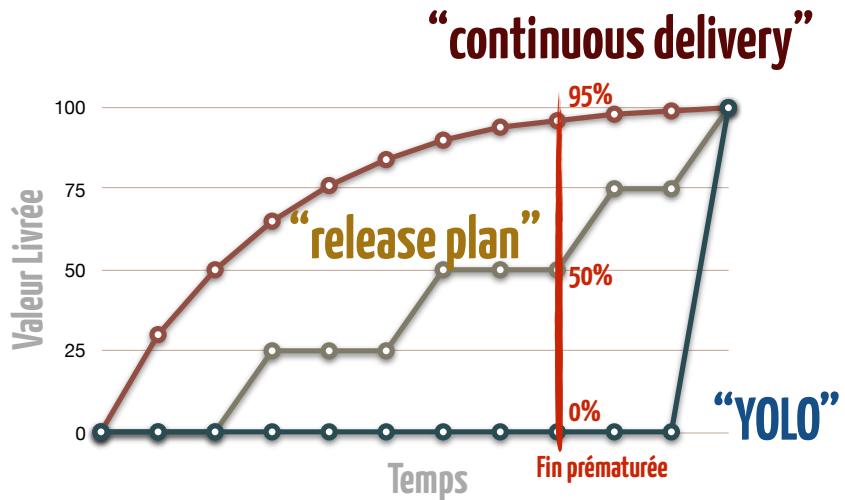
# Comment livrer en continu ?

 23 h 56 Bonsoir prof. Mosser, mon merge avec la branche master a ete effectue a 11:53, j'espere que ca n'affecte pas, ce fut un challenge! A ce mercredi!

23 h 14 Bonjour Mr , est ce qu'on doit ajouter le tag pour le tp2 ?

**Date limite : 23h50**

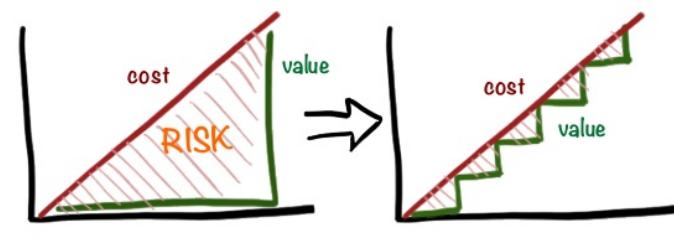
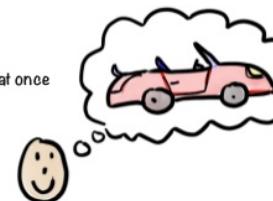
# Livraison de valeur au client



Agile = Iterative + Incremental

Don't try to get it all right  
from the beginning

Don't build it all at once



Le risque est l'intégrale du coût - celle de la valeur

## Étape 0 : Environnement Technique

```
lucifer:teaching$ mkdir inf5153-junit-demo
lucifer:teaching$ cd inf5153-junit-demo/
lucifer:inf5153-junit-demo$ touch pom.xml
lucifer:inf5153-junit-demo$ mkdir -p src/main/java
lucifer:inf5153-junit-demo$ touch src/main/java/Main.java
lucifer:inf5153-junit-demo$ touch .gitignore
lucifer:inf5153-junit-demo$ git init
lucifer:inf5153-junit-demo$ touch .travis.yml
```

Structure  
(Maven)

Gestion de version  
(Git)

Int. Continue  
(Travis)

Infos sur le projet

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">

<modelVersion>4.0.0</modelVersion>
<packaging>jar</packaging>

<groupId>io.github.ace-lectures.inf5153</groupId>
<artifactId>junit-demo</artifactId>
<version>1.0-SNAPSHOT</version>
<name>ACE :: INF5153 :: Junit Demo</name>

<properties>
  <maven.compiler.source>11</maven.compiler.source>
  <maven.compiler.target>11</maven.compiler.target>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```

Infos sur le code source

```
<build>
  <plugins>
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>exec-maven-plugin</artifactId>
      <version>1.6.0</version>
      <configuration>
        <mainClass>Main</mainClass>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Directives d'exécution

pom.xml

```

language: java

jdk:
  - openjdk11

notifications:
  email:
    on_success: never
    on_failure: always

```

~~.travis.yml~~

```

.idea/
target/
*~
*.iml

```

~~.gitignore~~

```

public class Main {

    public static void main(String[] args) {
        System.out.println("# JUnit Demonstration");
    }
}

Main.java

```

inf5153-junit-demo — bash — 80x5

[INFO] -----

lucifer:inf5153-junit-demo mosser\$ mvn -q clean package  
lucifer:inf5153-junit-demo mosser\$ mvn -q exec:java  
# JUnit Demonstration  
lucifer:inf5153-junit-demo mosser\$

lucifer:inf5153-junit-demo\$ touch add -A  
lucifer:inf5153-junit-demo\$ git commit -m "Initial arch."  
lucifer:inf5153-junit-demo\$ git push origin master

ace-lectures / inf5153-junit-demo build passing

Current Branches Build History Pull Requests > Build #1

master Initial architecture

- > Commit 7f346ae
- ↳ Compare 95fbfd..7f346ae
- Branch master
- Sébastien Mosser

#1 passed  
Ran for 39 sec  
less than a minute ago

Restart build Debug build

JDK: openjdk11 Java



lucifer:inf5153-junit-demo mosser\$ git tag step0  
lucifer:inf513-junit-demo mosser\$ git push --tags  
Enumerating objects: 7, done.  
Counting objects: 100% (7/7), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (4/4), 514 bytes | 514.00 KiB/s, done.  
Total 4 (delta 2), reused 0 (delta 0)  
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.  
To https://github.com/ace-lectures/inf5153-junit-demo.git  
 \* [new tag] step0 -> step0  
lucifer:inf5153-junit-demo mosser\$

2 commits 1 branch 1 release 1 contributor

Branch: master New pull request

Switch branches/tags

Latest commit 7f346ae an hour ago
Initial architecture
Initial architecture
initial commit
initial commit
initial commit

.travis.yml pom.xml

## Étape 0(bis) : Découpage fonctionnel

public class IntegerTest extends TestCase {

Intégration Java

```

private int x = 0;
private int y = 0;

public void setUp() throws Exception {
    x = 1;
    y = 1;
}

```

Mise en contexte

Suite  
De  
Tests

```

public void testAdd() {
    assertEquals(2, x+y)
}

public void testSub() {
    assertEquals(1, x-y)
}

```

Cas de tests

Assertions

Succès / Échec

## Étape 0(bis) : Découpage fonctionnel

1. Assertions
2. Cas de tests
3. Succès / Échec
4. Suite de tests
5. Mise en contexte
6. Intégration Java

```
public class IntegerTest extends TestCase {  
  
    private int x = 0;  
    private int y = 0;  
  
    public void setUp() throws Exception {  
        x = 1;  
        y = 1;  
    }  
  
    public void testAdd() {  
        assertEquals(2, x+y)  
    }  
  
    public void testSub() {  
        assertEquals(0, x-y)  
    }  
}
```



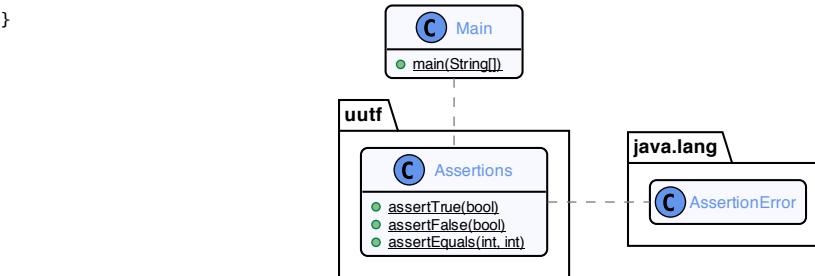
## Qu'est-ce qu'une assertion ?

- Méthode indépendante de l'objet qui la contient
- Doit vérifier une condition (vrai, faux, égalité, équivalence, ...)
- Il n'est pas de la responsabilité du développeur de traiter un échec d'assertion

Spécification

Choix de conception

```
package uutf;  
  
public class Assertions {  
  
    public static void assertTrue(boolean condition) {  
        if(!condition)  
            throw new AssertionError();  
    }  
  
    public static void assertFalse(boolean condition) {  
        assertTrue(!condition);  
    }  
  
    public static void assertEquals(int oracle, int value) {  
        assertTrue(oracle == value);  
    }  
}
```



```

import static uutf.Assertions.assertEquals;

public class Main {
    public static void main(String[] args) {
        System.out.println("# JUnit Demonstration");

        try {
            System.out.println("## Case 1: x+y == 2");
            int x = 1;
            int y = 1;
            assertEquals(2, x+y);
            System.out.println("PASSED");
        } catch(AssertionError ae) {
            System.out.println("FAILED");
        }

        try {
            System.out.println("## Case 2: x-y == 1");
            int x = 1;
            int y = 1;
            assertEquals(1, x-y);
            System.out.println("PASSED");
        } catch(AssertionError ae) {
            System.out.println("FAILED");
        }
    }
}

```



```

lucifer:inf5153-junit-demo mosser$ mvn -q clean package exec:java
# JUnit Demonstration
## Case 1: x+y == 2
PASSED
## Case 2: x-y == 1
FAILED
lucifer:inf5153-junit-demo mosser$ 

```

On avance pas à pas ...  
Et on ajoute de la valeur.



```

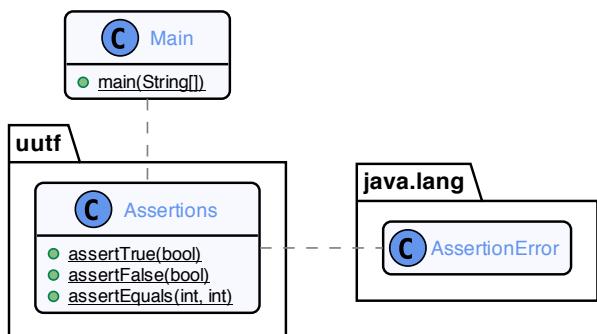
try {
    System.out.println("## Case 1: x+y == 2");
    int x = 1;
    int y = 1;
    assertEquals(2, x+y);
    System.out.println("PASSED");
} catch(AssertionError ae) {
    System.out.println("FAILED");
}

```

public void testAdd() {  
 assertEquals(2, x+y);  
}



**UQÀM** Unit Test Framework



Version 1. 13 lines de Java.

Cas de  
Test

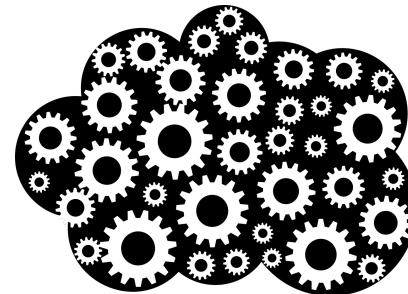
3

# Problème: Responsabilisation du test ?

```
try {
    System.out.println("## Case 1: x+y == 2");
    int x = 1;
    int y = 1;
    assertEquals(2, x+y);
    System.out.println("PASSED");
} catch(AssertionError ae) {
    System.out.println("FAILED");
}
```



```
public abstract class TestCase {
```



Canevas Logiciel

```
public class AddTwoNumbers extends TestCase {

    protected void test() {
        int x = 1;
        int y = 1;
        assertEquals(2, x+y);
    }
}
```



Utilisateur

```
public abstract class TestCase {

    public void run() {
        try {
            System.out.print("## Case " + this.getClass().getCanonicalName() + " ");
            test();
            System.out.println("[X]");
        } catch (AssertionError ae) {
            System.out.println("[  ]");
        } catch (Exception e) {
            System.out.println("[!]");
        }
    }

    protected abstract void test();
}

public class AddTwoNumbers extends TestCase {

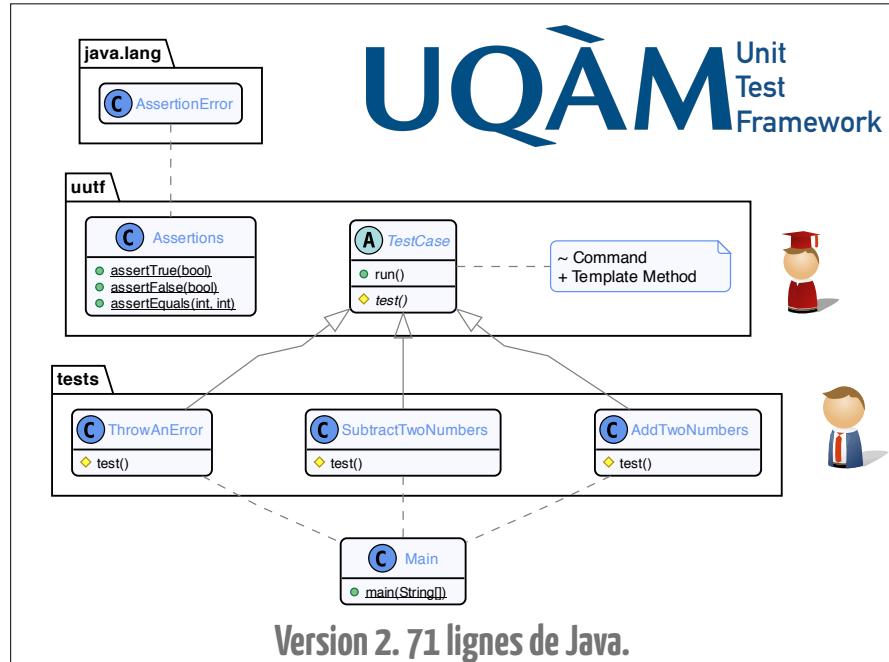
    @Override
    protected void test() {
        int x = 1;
        int y = 1;
        assertEquals(2, x+y);
    }
}
```

Template  
Method

Extension par  
Héritage

```
[lucifer:inf5153-junit-demo mosser]$ mvn -q clean package exec:java
# JUnit Demonstration
## Case tests.AddTwoNumbers [X]
## Case tests.SubtractTwoNumbers [ ]
## Case tests.ThrowAnException [!]
lucifer:inf5153-junit-demo mosser$
```

```
public static void main(String[] args) {
    System.out.println("# JUnit Demonstration");
    (new AddTwoNumbers()).run();
    (new SubtractTwoNumbers()).run();
    (new ThrowAnException()).run();
}
```

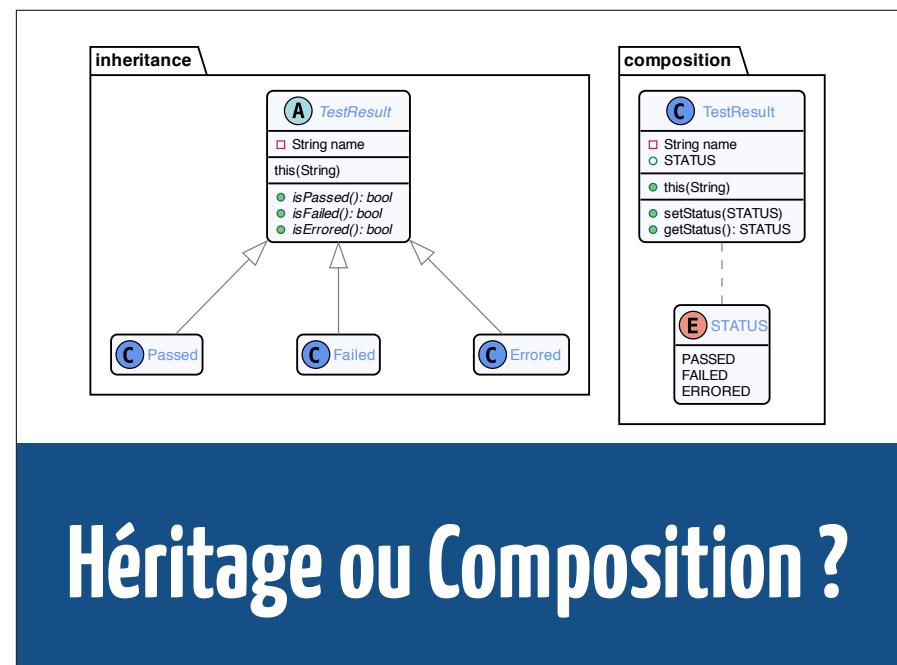


# Succès & échecs

4

## Comment traiter les résultats ?

```
# JUnit Demonstration
## [!] tests.ThrowAnException
## [X] tests.AddTwoNumbers
## [ ] tests.SubtractTwoNumbers
{[Passed]=1, [Failed]=1, [Errored]=1}
```



```

public class TestResult {

    public enum STATUS {
        PASSED('X'),
        FAILED(' '),
        ERRORED('!');
    }

    private char symbol;

    STATUS(char symbol) {
        this.symbol = symbol;
    }

    @Override
    public String toString() {
        return "[" + symbol + "]";
    }
}

public class TestResult {
    private String name;
    private STATUS status;

    public TestResult(String name) {
        this.name = name;
    }

    public void setStatus(STATUS status) {
        this.status = status;
    }

    public STATUS getStatus() {
        return status;
    }

    @Override
    public String toString() {
        return "## " + status + " " + name;
    }
}

```

```

public abstract class TestCase {

    public final TestResult run() {
        TestResult result = new TestResult(this.getClass().getCanonicalName());
        try {
            test();
            result.setStatus(STATUS.PASSED);
        } catch (AssertionError ae) {
            result.setStatus(STATUS.FAILED);
        } catch (Exception e) {
            result.setStatus(STATUS.ERRORED);
        }
        return result;
    }

    protected abstract void test();
}

```

```

public class TestRunner {

    private Set<TestCase> cases = new HashSet<>();

    public void addCase(TestCase tc) {
        this.cases.add(tc);
    }

    public void run() {
        Map<STATUS, Integer> counters = new HashMap<>();
        Set<TestResult> results = new HashSet<>();
        for(TestCase tc: cases) {
            TestResult r = tc.run();
            counters.put(r.getStatus(), counters.getOrDefault(r.getStatus(), 0)+1);
            results.add(r);
        }
        results.forEach(System.out::println);
        System.out.println(counters);
    }
}

```

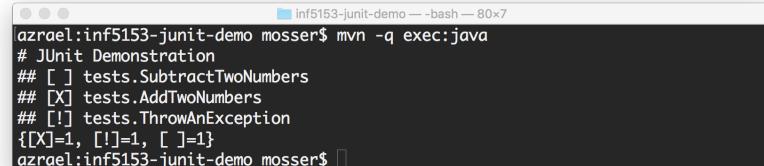
```

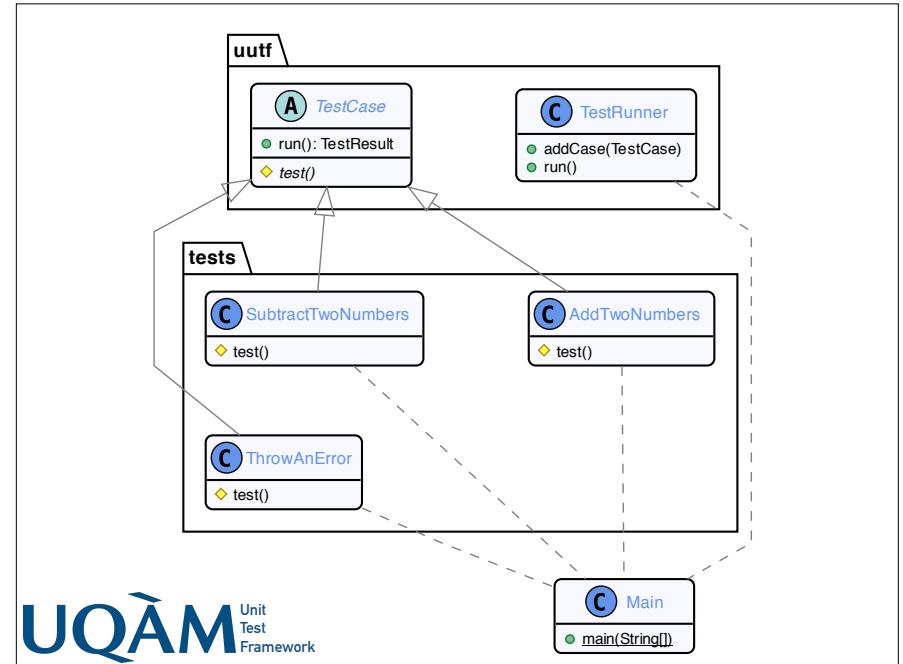
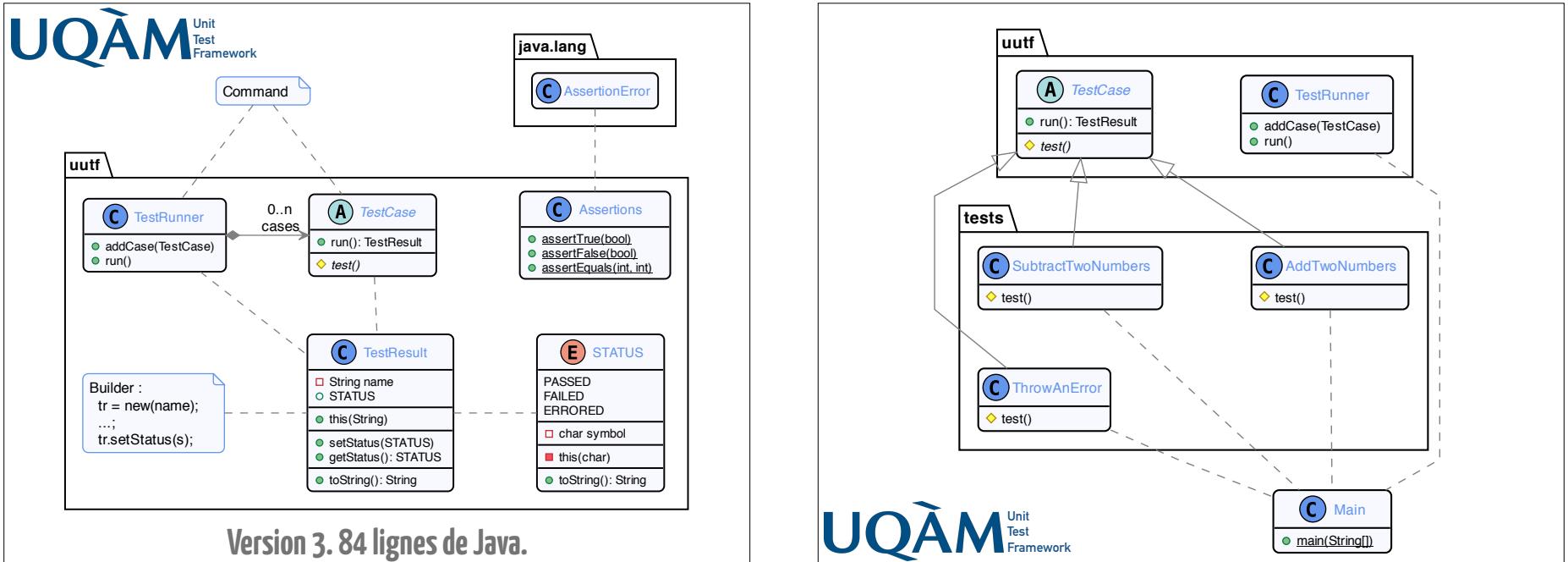
public class Main {

    public static void main(String[] args) {
        System.out.println("# JUnit Demonstration");

        TestRunner runner = new TestRunner();
        runner.addCase(new AddTwoNumbers());
        runner.addCase(new SubtractTwoNumbers());
        runner.addCase(new ThrowAnException());
        runner.run();
    }
}

```





**Problème :**  
**C'est le Runner qui compose !**

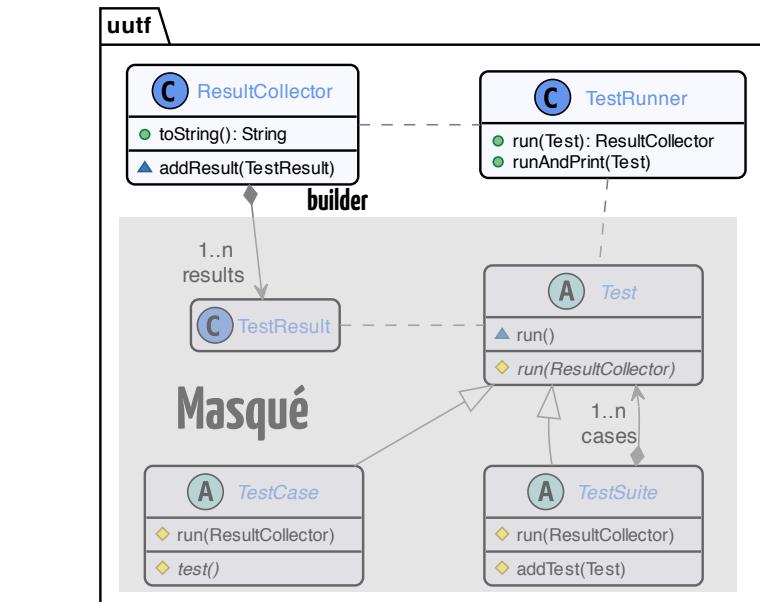
```

public class TestRunner {
    // ...
    public void run() {
        Map<STATUS, Integer> counters = new HashMap<>();
        Set<TestResult> results = new HashSet<>();
        for(TestCase tc: cases) {
            TestResult r = tc.run();
            counters.put(r.getStatus(), counters.getOrDefault(r.getStatus(), 0)+1);
            results.add(r);
        }
        results.forEach(System.out::println);
        System.out.println(counters);
    }
}
    
```

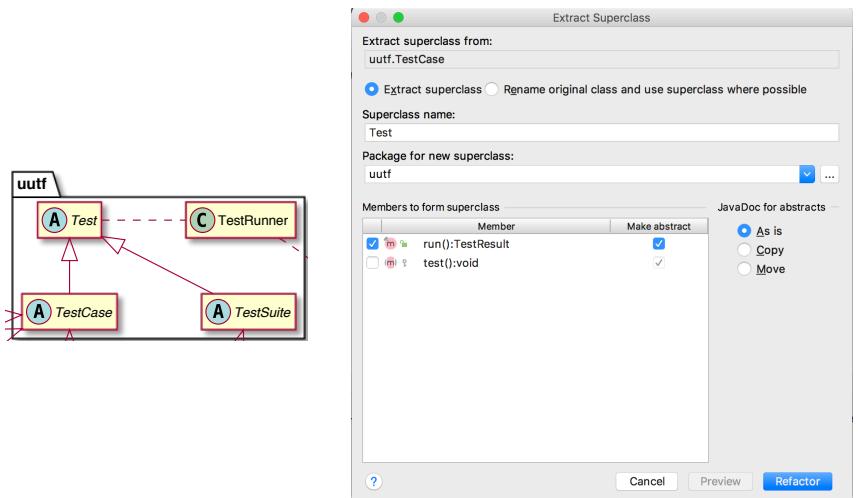
Responsabilité unique ...

# Proposition de conception

- Un **Test** peut être :
  - Un **TestCase** atomique (AddTwoNumbers, ...)
  - Une **TestSuite**, qui est composée de **Tests**(patron Composite)
- Un **Test** porte la responsabilité de s'exécuter
- Le **TestRunner** lance UN SEUL **Test** et affiche son résultat



## Extraction automatique de la super-classe Test



```
public abstract class Test {  
    final void run() {  
        ResultCollector rc = new ResultCollector();  
        this.run(rc);  
        System.out.println(rc);  
    }  
  
    protected abstract void run(ResultCollector rc);  
}  
  
public class ResultCollector {  
    private Set<TestResult> results = new HashSet<>();  
  
    public void addResult(TestResult tr) { this.results.add(tr); }  
  
    @Override  
    public String toString() {  
        StringBuilder builder = new StringBuilder();  
        Map<STATUS, Integer> counters = new HashMap<>();  
        for(TestResult tr: results) {  
            counters.put(tr.getStatus(), counters.getOrDefault(tr.getStatus(), 0)+1);  
            builder.append(tr).append("\n");  
        }  
        builder.append(counters);  
        return builder.toString();  
    }  
}
```

```

public abstract class TestCase extends Test {
    @Override
    protected final void run(ResultCollector rc) {
        TestResult result = new TestResult(this.getClass().getCanonicalName());
        try {
            test();
            result.setStatus(STATUS.PASSED);
        } catch (AssertionError ae) {
            result.setStatus(STATUS.FAILED);
        } catch (Exception e) {
            result.setStatus(STATUS.ERRORED);
        }
        rc.addResult(result);
    }

    protected abstract void test();
}

public class TestSuite extends Test {
    private Set<Test> cases = new HashSet<>();

    protected void addTest(Test t) { this.cases.add(t); }

    @Override
    protected void run(ResultCollector rc) {
        for(Test t: this.cases)
            t.run(rc);
    }
}

```

```

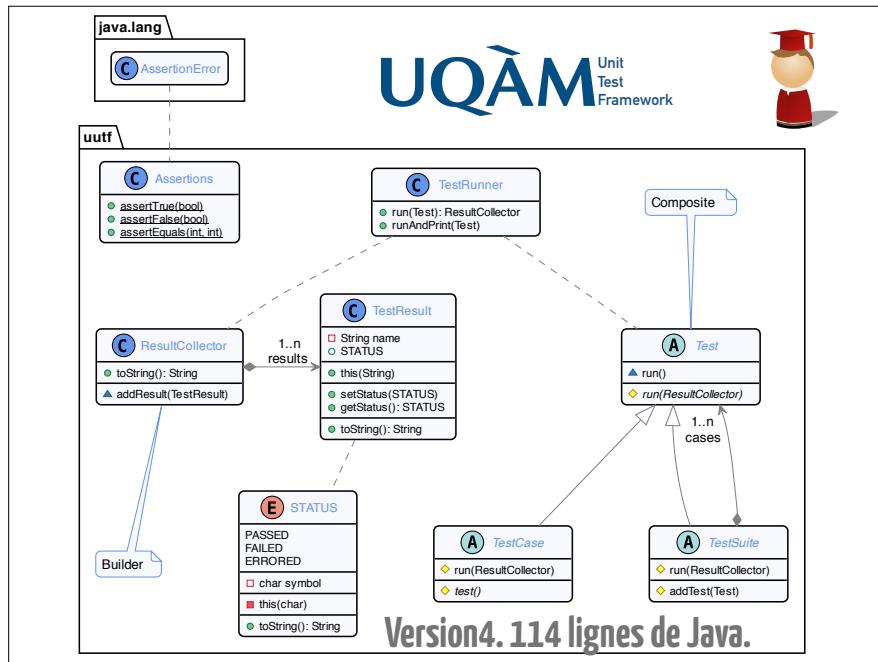
public class Main {
    public static void main(String[] args) {
        System.out.println("# JUnit Demonstration");
        TestRunner runner = new TestRunner();
        runner.runAndPrint(new IntegerTests());
    }
}

public class TestRunner {
    public void runAndPrint(Test t) {
        System.out.println(this.run(t));
    }

    public ResultCollector run(Test t) {
        ResultCollector rc = new ResultCollector();
        t.run(new ResultCollector());
        return rc;
    }
}

public class IntegerTests extends TestSuite {
    public IntegerTests() {
        addTest(new SubtractTwoNumbers());
        addTest(new ThrowAnException());
        addTest(new AddTwoNumbers());
    }
}

```



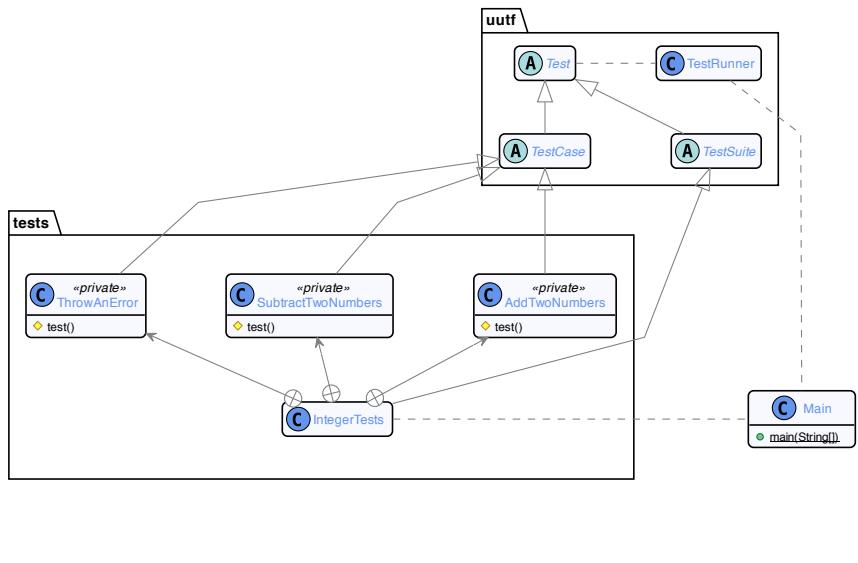
```

public class IntegerTests extends TestSuite {
    public IntegerTests() {
        addTest(new AddTwoNumbers());
    }

    private class AddTwoNumbers extends TestCase {
        private int x, y = 0;

        @Override
        protected void test() {
            x = 1; y = 1;
            assertEquals(2, x+y);
        }
    }
}

```



# Mise en contexte

6

## Problème : Contexte mélangé au test !

```

public class AddTwoNumbers extends TestCase {

    @Override
    protected void test() {
        int x = 1;
        int y = 1;
        assertEquals(2, x+y);
    }
}

```

Responsabilité unique ...

Junit 3.x utilise  
des méthodes  
prédéfinies

```

public class IntegerTest extends TestCase {

    private int x = 0;
    private int y = 0;

    public void setUp() throws Exception {
        x = 1;
        y = 1;
    }

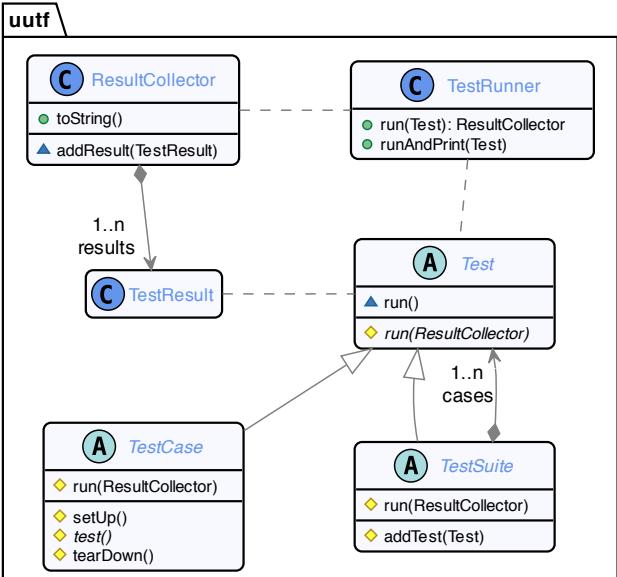
    public void testAdd() {
        assertEquals(2, x+y)
    }

    public void testSub() {
        assertEquals(0, x-y)
    }
}

```

**setUp()**

**tearDown()**

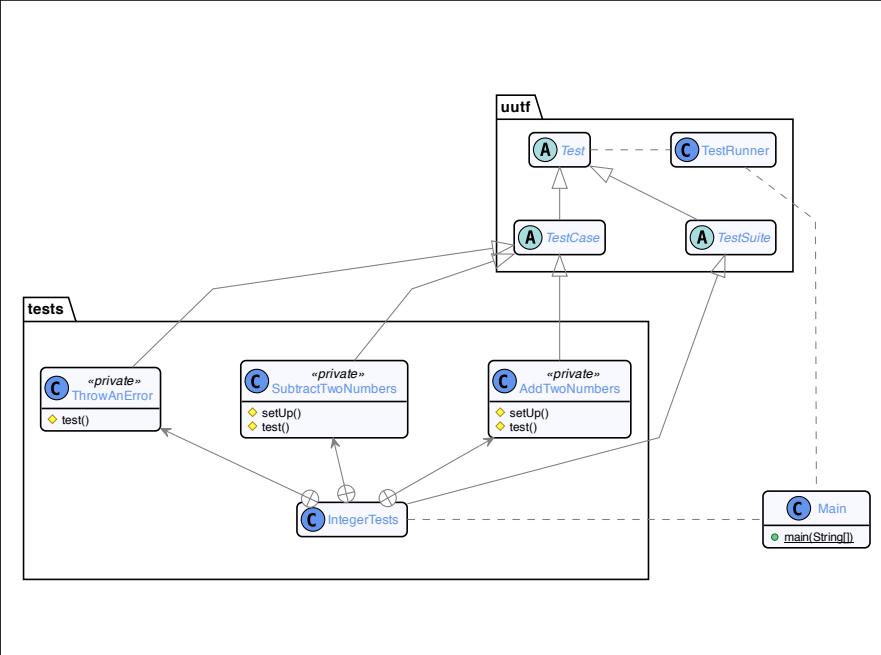
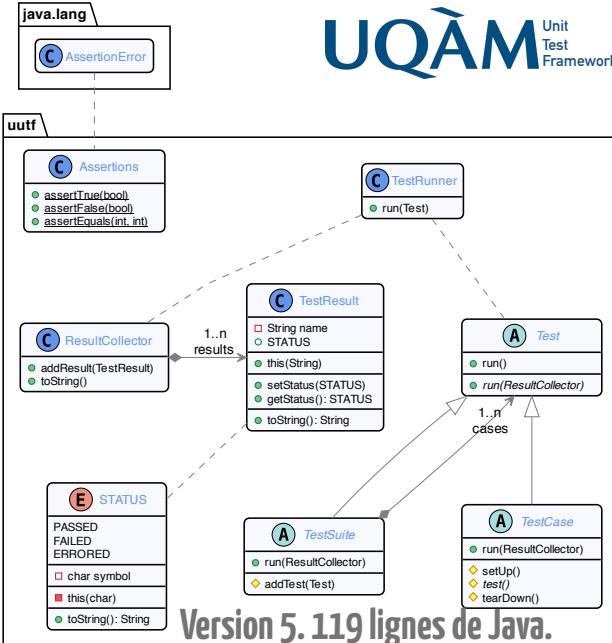


```

public abstract class TestCase extends Test {
    @Override
    protected final void run(ResultCollector rc) {
        TestResult result = new TestResult(this.getClass().getCanonicalName());
        try {
            setUp();
            test();
            result.setStatus(STATUS.PASSED);
        } catch (AssertionError ae) {
            result.setStatus(STATUS.FAILED);
        } catch (Exception e) {
            result.setStatus(STATUS.ERRORED);
        } finally {
            tearDown();
        }
        rc.addResult(result);
    }

    protected void setUp() { }
    protected abstract void test();
    protected void tearDown() { }
}
  
```

This is the Java code for the TestCase class, which extends the Test class. It overrides the run method to handle assertions and exceptions, setting the status of the TestResult accordingly. It also provides protected methods for setup, test, and teardown.



```

public class IntegerTests extends TestSuite {
    public IntegerTests() {
        addTest(new SubtractTwoNumbers());
    }

    private class AddTwoNumbers extends TestCase {
        private int x, y = 0;

        @Override protected void setUp() {
            x = 1; y = 1;
        }

        @Override
        protected void test() {
            assertEquals(2, x+y);
        }
    }
}

On y est  
presque ...

```

# Intégration

# Java

## Comment dégager les classes internes ?

```

private class AddTwoNumbers extends TestCase {
    private int x, y = 0;

    @Override protected void setUp() {
        x = 1; y = 1;
    }

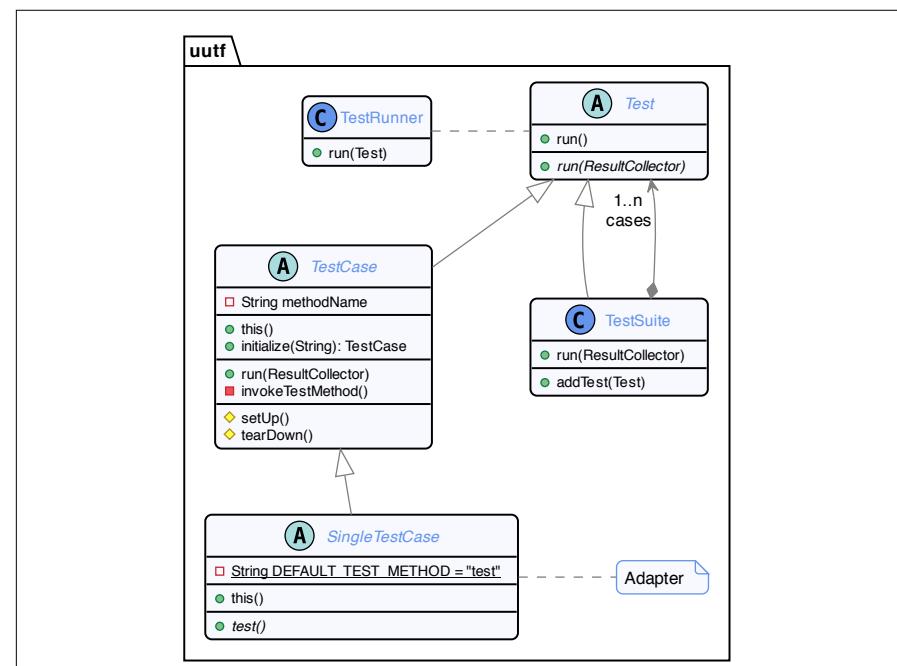
    @Override
    protected void test() {
        assertEquals(2, x+y);
    }
}

public class IntegerTest extends TestCase {
    private int x = 0;
    private int y = 0;

    public void setUp() throws Exception {
        x = 1;
        y = 1;
    }

    public void testAdd() {
        assertEquals(2, x+y)
    }
}

```



```

public abstract class TestCase extends Test {
    private String testMethodName;

    public TestCase initialize(String testMethodName) {
        this.testMethodName = testMethodName;
        return this;
    }

    @Override
    protected void run(ResultCollector rc) {
        TestResult result =
            new TestResult(this.getClass().getCanonicalName() + ":" + testMethodName);
        try {
            setUp();
            invokeTestMethod();
            result.setStatus(STATUS.PASSED);
        } catch (InvocationTargetException ite) {
            if (ite.getTargetException() instanceof AssertionException)
                result.setStatus(STATUS.FAILED);
            else
                result.setStatus(STATUS.ERRORED);
        } catch (Exception e) {
            result.setStatus(STATUS.ERRORED);
        } finally {
            tearDown();
        }
        rc.addResult(result);
    }

    private void invokeTestMethod() throws Exception {
        Method tm = this.getClass().getMethod(testMethodName);
        tm.invoke(this);
    }

    public void setUp() { }
    public void tearDown() { }
}

```

```

public abstract class SingleTestCase extends TestCase {
    private static final String DEFAULT_TEST_METHOD = "test";
    public SingleTestCase() { initialize(DEFAULT_TEST_METHOD); }

    public abstract void test();
}

public class TestSuite extends Test {
    private Set<Test> cases = new HashSet<>();

    public void addTest(Test t) { this.cases.add(t); }

    @Override
    protected void run(ResultCollector rc) {
        for (Test t: this.cases)
            t.run(rc);
    }
}

```

```

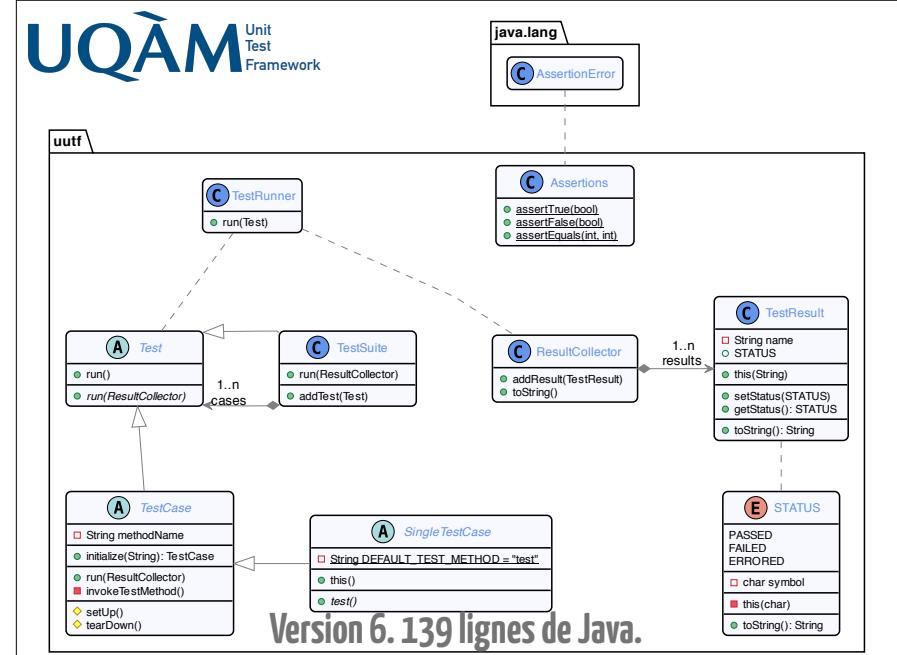
public class IntegerTests extends TestCase {

    public static Test build() {
        TestSuite suite = new TestSuite();
        suite.addTest(new IntegerTests().initialize("addTwoNumbers"));
        suite.addTest(new IntegerTests().initialize("subtractTwoNumbers"));
        suite.addTest(new IntegerTests().initialize("throwAnException"));
        return suite;
    }

    private int x, y = 0;

    @Override public void setUp() { x = 1; y = 1; }
    public void addTwoNumbers() { assertEquals(2, x+y); }
    public void subtractTwoNumbers() { assertEquals(1, x-y); }
    public void throwAnException() { throw new RuntimeException(); }
}

```



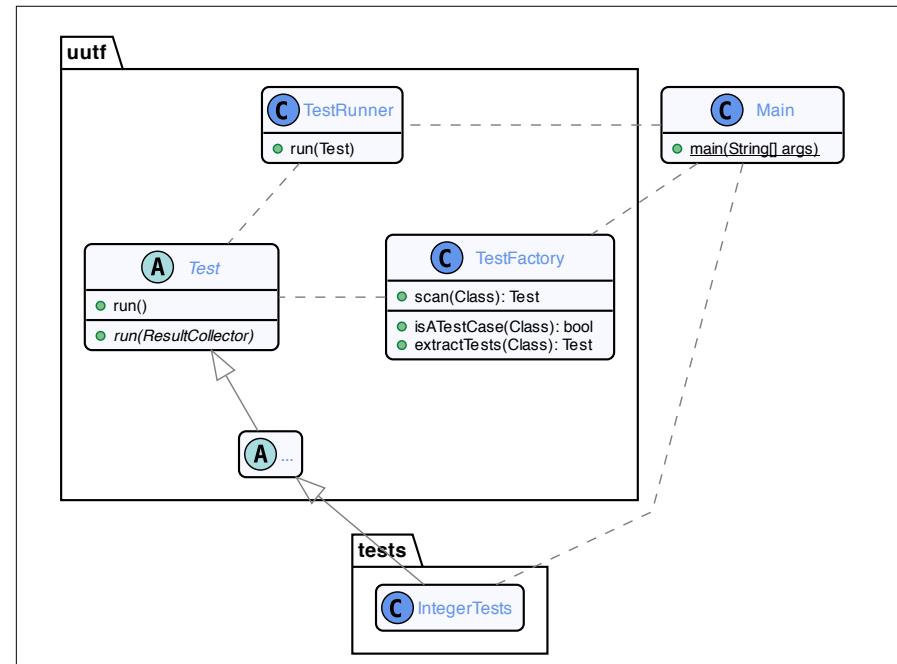
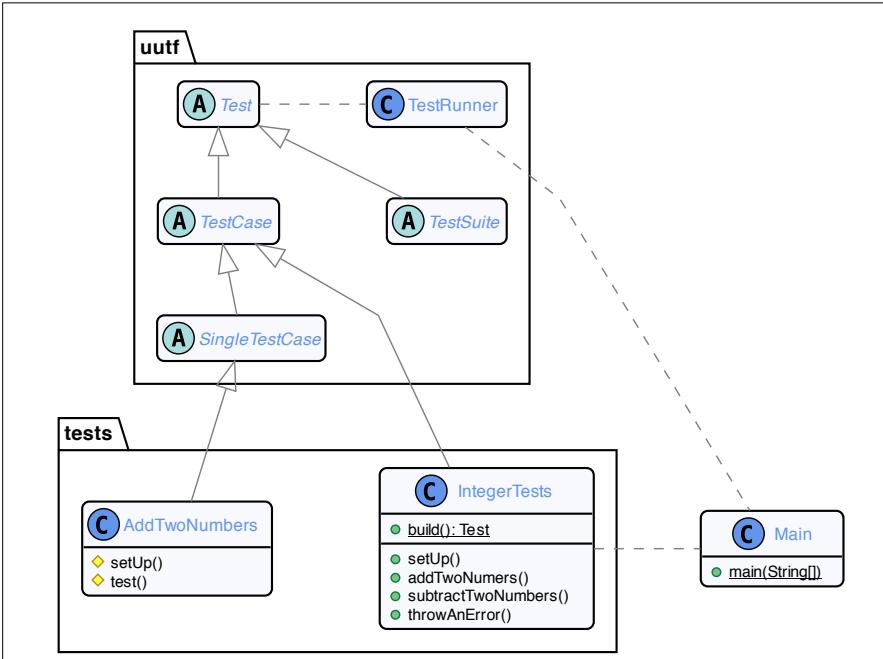
# Intégration Java

7  
Le retour

## Comment dégager la fonction de build ?

```
public static Test build() {
    TestSuite suite = new TestSuite();
    suite.addTest(new IntegerTests("addTwoNumbers"));
    suite.addTest(new IntegerTests("subtractTwoNumbers"));
    suite.addTest(new IntegerTests("throwAnException"));
    return suite;
}
```

Un peu comme si on voulait FABRIQUER des objets ...



```

public class TestFactory {
    private static final String TEST_METHOD_PREFIX = "test_";

    public Test scan(Class klass) {
        if (!isTestCase(klass))
            return new TestSuite();
        try {
            return extractTests(klass);
        } catch (Exception e) {
            return new TestSuite();
        }
    }

    private Test extractTests(Class klass) throws Exception {
    }

    private boolean isTestCase(Class klass) {
    }
}

```

## Méta-Programmation

```

private void buildTestCase(Class klass, TestSuite suite, Method m)
    throws Exception {
    TestCase tc =
        (TestCase) klass.getDeclaredConstructor().newInstance();
    suite.addTest(tc.initialize(m.getName()));
}

```

```

public static Test build() {
    TestSuite suite = new TestSuite();
    suite.addTest(new IntegerTests().initialize("addTwoNumbers"));
    suite.addTest(new IntegerTests().initialize("subtractTwoNumbers"));
    suite.addTest(new IntegerTests().initialize("throwAnException"));
    return suite;
}

```

## Programmation

```

public class Main {
    public static void main(String[] args) {
        System.out.println("# JUnit Demonstration");

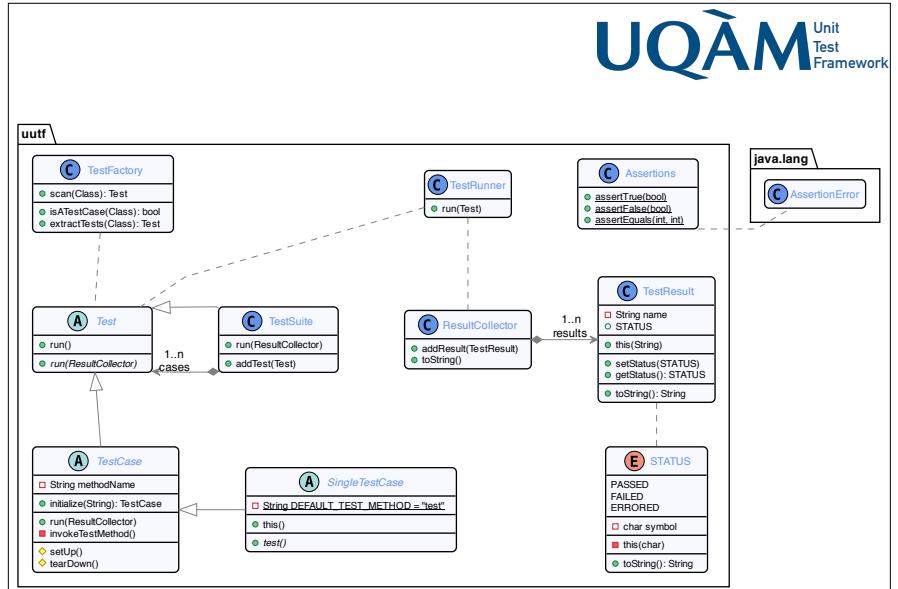
        TestFactory scanner = new TestFactory();
        TestRunner runner = new TestRunner();
        runner.runAndPrint(scanner.scan(IntegerTests.class));
    }
}

public class IntegerTests extends TestCase {
    private int x, y = 0;

    @Override public void setUp() { x = 1; y = 1; }
    public void test_addTwoNumbers() { assertEquals(2, x+y); }
    public void test_subtractTwoNumbers() { assertEquals(1, x-y); }
    public void test_throwAnException() { throw new RuntimeException(); }
}

```

**UQÀM** Unit Test Framework



Version 6bis. 171 lignes de Java.

# Conclusion

# 8

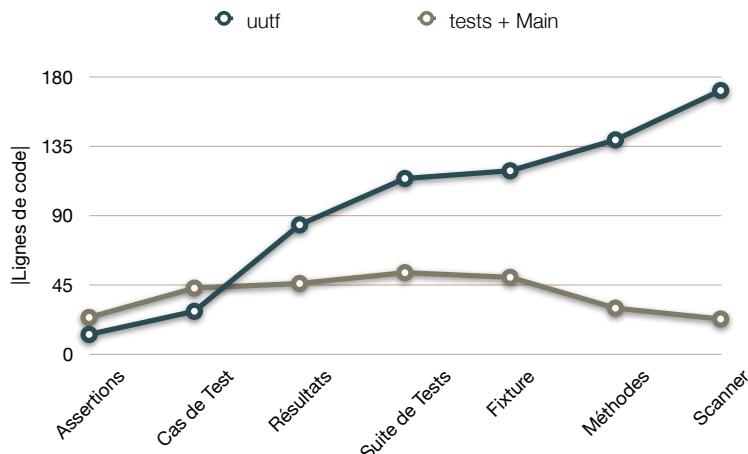
# JUnit

```
public class IntegerTest extends TestCase {  
  
    private int x, y = 0;  
  
    public void setUp() throws Exception {  
        x = 1; y = 1;  
    }  
  
    public void testAdd() {  
        assertEquals(2, x+y)  
    }  
}
```

```
public class IntegerTests extends TestCase {  
  
    private int x, y = 0;  
  
    public void setUp() {  
        x = 1; y = 1;  
    }  
  
    public void test_addTwoNumbers() {  
        assertEquals(2, x+y);  
    }  
}
```

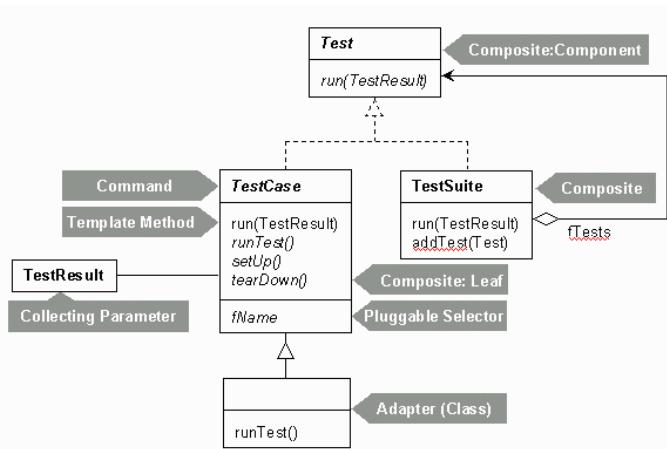
UQÀM Unit Test Framework

## Taille et Livraison de valeur



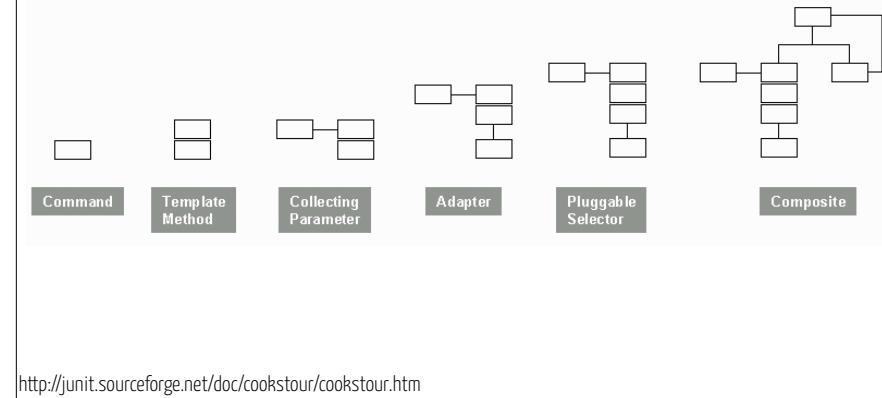
		Objectif		
		Création	Structure	Comportement
Portée	classe	Factory	Adapter	Interpreter Template Method
	objet	Abstract Factory Builder Prototype Singleton	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor	Adapter Bridge Composite Decorator Facade Flyweight Proxy

## On a un peu triché (mais pas tant que ça en fait)



<http://junit.sourceforge.net/doc/cookstour/cookstour.htm>

## Conception Itérative ("vrai histoire", pour JUnit 3.8.x)



<http://junit.sourceforge.net/doc/cookstour/cookstour.htm>