

## Génie Logiciel : Conception

Étude de cas JUnit (3.x)

Sébastien Mosser  
INF-5153, Hiver 2019, Cours #9

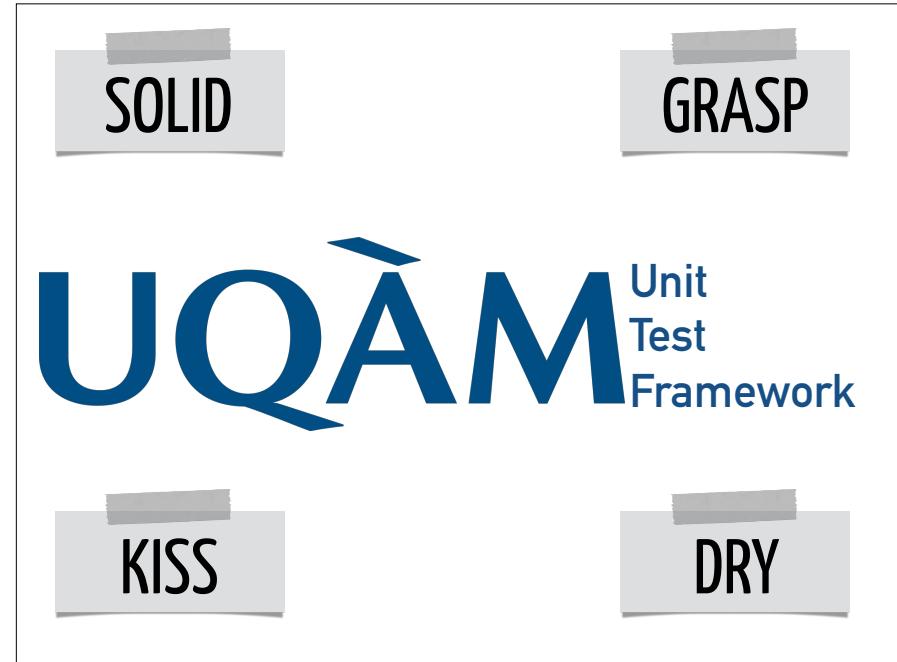
UQÀM

Images: Pixabay

1

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

3



2

## Objectif: On (ré-)implémente JUnit !

```
public class IntegerTest {
    private int x = 0;
    private int y = 0;

    @Before
    public void initializeContext() {
        x = 1;
        y = 1;
    }

    @Test
    public void add2numbers() {
        assertEquals(2, x+y)
    }

    @Test
    public void subtract2numbers() {
        assertEquals(0, x-y)
    }
}
```

4-1

# Objectif: On (ré-)implémente JUnit !

```
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);
    }
}
```

Avant (< 2006)

```
public class IntegerTest {
    private int x = 0;
    private int y = 0;

    @Before
    public void initializeContext() {
        x = 1;
        y = 1;
    }

    @Test
    public void add2numbers() {
        assertEquals(2, x+y);
    }

    @Test
    public void subtract2numbers() {
        assertEquals(0, x-y);
    }
}
```

Après (≥ 2006)

4-2

```
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);
    }
}
```

(-171 LoC)

JUnit

UQÀM Unit Test Framework

5-2

```
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

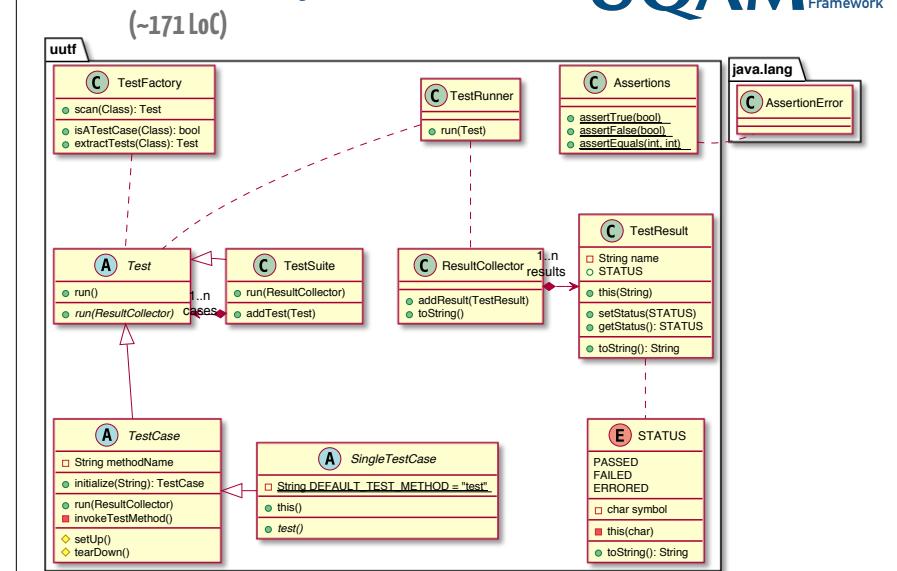
```
public class IntegerTests extends TestCase {
    private int x, y = 0;

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

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

5-1

## Réalisation Complète



6

# Agilité

(aka "conception en continu")

7

## Livraison de valeur au client

"continuous delivery"



8-1

## Livraison de valeur au client

"continuous delivery"

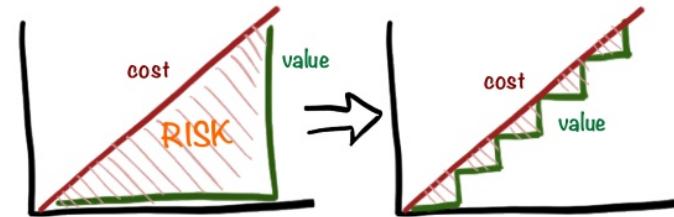
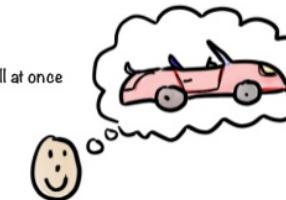


8-2

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

9

## 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    Gestion de version    Int. Continue  
(Maven)        (Git)                  (Travis)

10

```
language: java

jdk:
  - openjdk11

notifications:
  email:
    on_success: never
    on_failure: always

.travis.yml ✖
.gitignore ✖

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

```
<?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>

  <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>
</project>
```

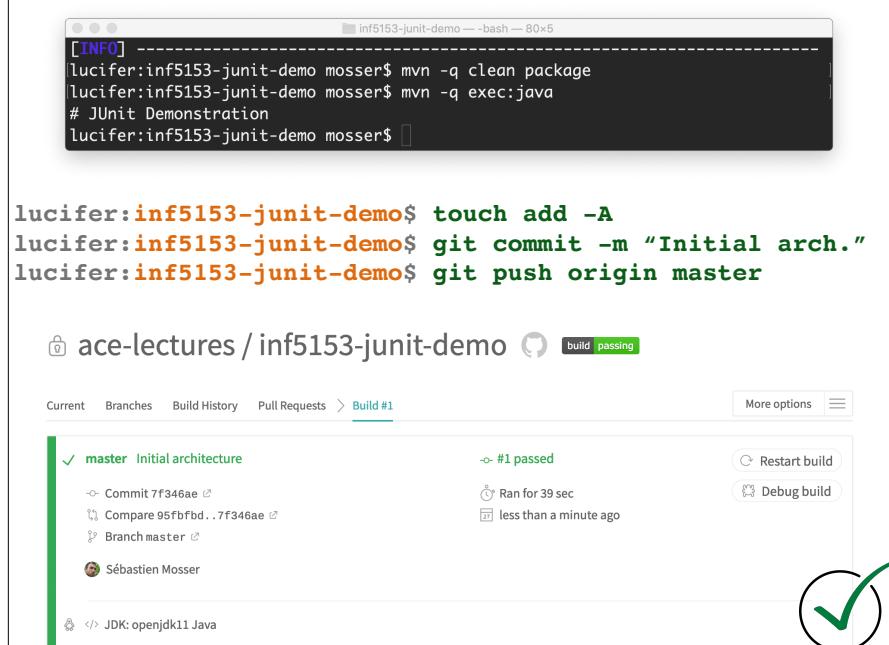
pom.xml ✖

Infos sur le projet

Infos sur le code source

Directives d'exécution

11



12

13

```

lucifer:inf5153-junit-demo mosser$ git tag step0
lucifer:inf5153-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

Create new file Upload files Find File Clone or download

Switch branches/tags

Find a tag... Latest commit 7f346ae an hour ago

Branches	Tags	
<a href="#">step0</a>		
<a href="#">.travis.yml</a>		Initial architecture an hour ago
<a href="#">pom.xml</a>		Initial architecture an hour ago
		initial commit 2 hours ago
		initial commit 2 hours ago
		initial commit 2 hours ago

14

## Étape 0(bis) : Découpage fonctionnel

```

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(1, x-y)
    }
}

```

15-1

## Étape 0(bis) : Découpage fonctionnel

Suite  
De  
Tests

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(1, x-y)
    }
}

```

Mise en contexte

Cas de tests

Succès / Échec

Assertions

15-2

## 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)
    }
}

```

16

# Assertions



17

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

- Méthode indépendante de l'objet qui la contient
  - Elle sera donc statique
- Doit vérifier une condition (vrai, faux, égalité, équivalence, ...)
  - On peut tout ramener à la vérification de “vrai” (réutilisation)
- Il n'est pas de la responsabilité du développeur de traiter un échec d'assertion
  - On va donc utiliser des exceptions plutôt que des valeurs de retours

Spécification

Choix de conception

18-2

## 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

18-1

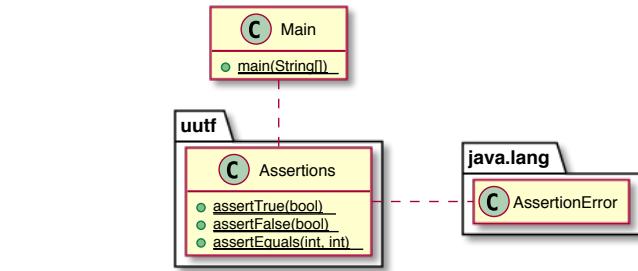
```
package uutf;

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);
    }
}
```



19

```

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");
        }
    }
}

```



20

```

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.



```

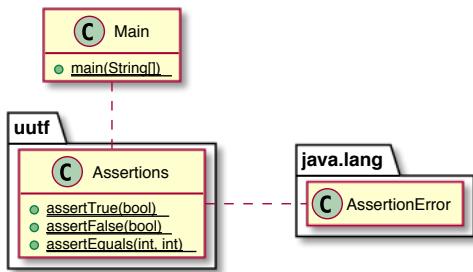
try {
    System.out.println("## Case 1: x+y == 2");
    int x = 1;
    int y = 1;
    assertEquals(2, x+y);
    System.out.println("PASSED");
} catch(AssertionError ae) {
    System.out.println("FAILED");
}

```

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

21

# UQÀM Unit Test Framework



Version 1.13 lines de Java.

22

# Cas de Test

# 3

23

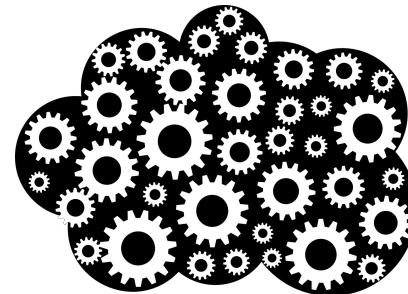
# Problème: Responsabilisation du test ?

```
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");
}
```



24

```
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();
        } catch (AssertionError ae) {
            System.out.println("[X]");
        } catch (Exception e) {
            System.out.println("[!]");
        }
    }

    protected abstract void test();
}
```



Canevas Logiciel

```
public class AddTwoNumbers extends TestCase {

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



Utilisateur

25-2

```
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();
```

Template  
Method

```
public class AddTwoNumbers extends TestCase {

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

Extension par  
Héritage

26

```
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$
```

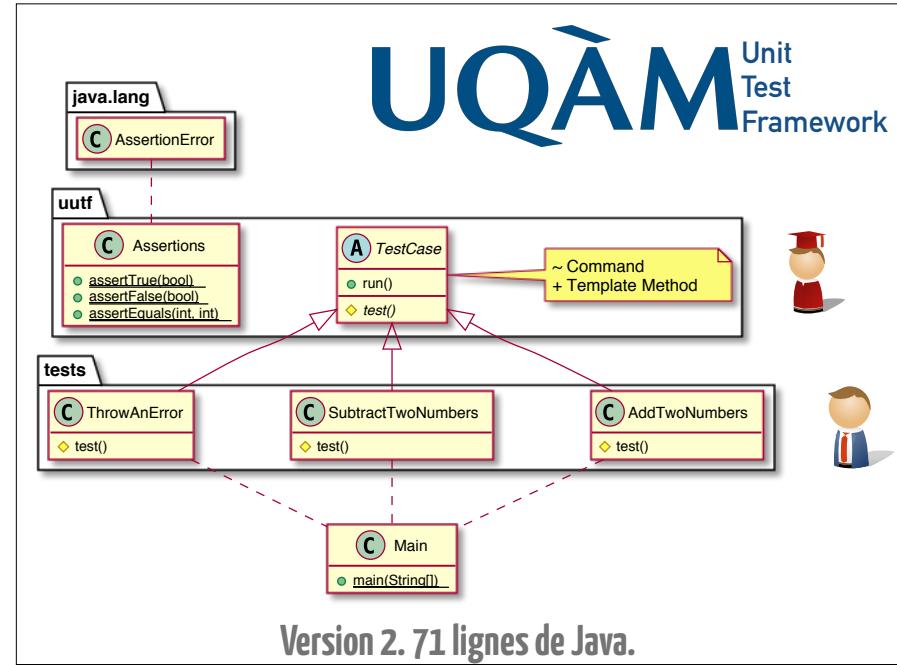
```
public static void main(String[] args) {
    System.out.println("# JUnit Demonstration");
    (new AddTwoNumbers()).run();
    (new SubtractTwoNumbers()).run();
    (new ThrowAnException()).run();
}
```

27

# Succès & échecs

4

29



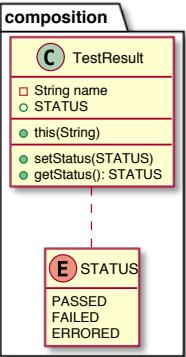
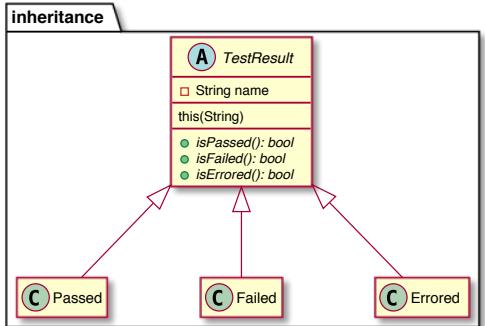
Version 2. 71 lignes de Java.

28

## Comment traiter les résultats ?

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

30



# Héritage ou Composition ?

31

```

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 TestResult {

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

    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 "["+symbol+"]";
    }
}

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

```

32

```

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);
    }
}

```

33

34

```

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();
    }
}

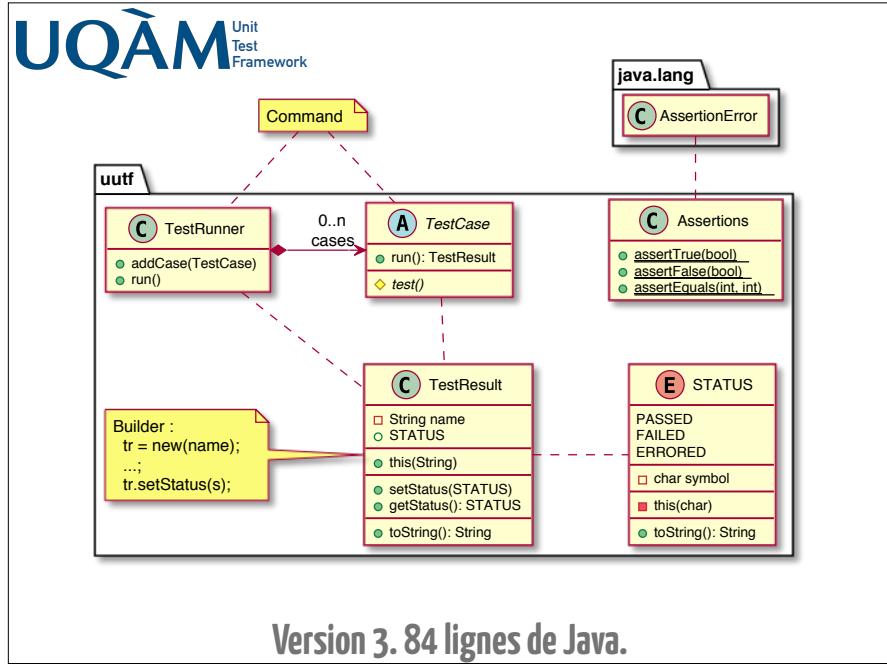
```

```

azrael:inf5153-junit-demo mosser$ mvn -q exec:java
# JUnit Demonstration
## [ ] tests.SubtractTwoNumbers
## [X] tests.AddTwoNumbers
## [!] tests.ThrowAnException
{[X]=1, [!]=1, [ ]=1}
azrael:inf5153-junit-demo mosser$

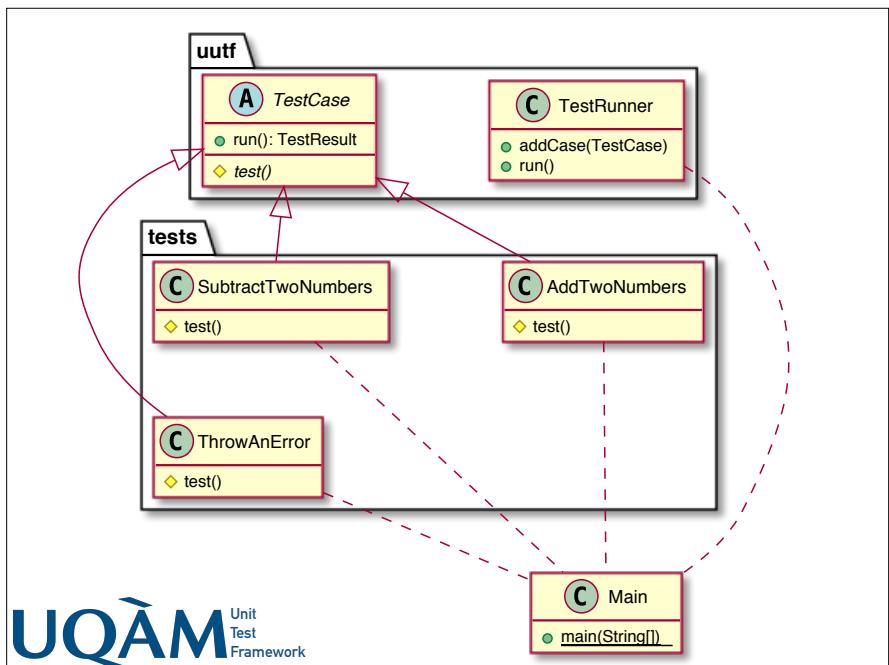
```

35



Version 3. 84 lignes de Java.

36



37

38

# 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 ...

39

## Proposition de conception

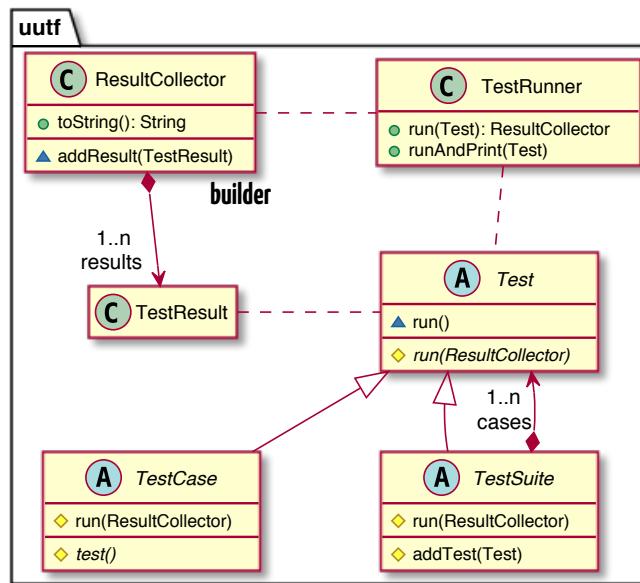
- Un **Test** peut être :
  - Un **TestCase** atomique (AddTwoNumbers, ...)
  - Une **TestSuite**, qui est composée de **Tests**(patron Composite)
- Un **Test** porte la responsabilité de s'exécuter
- Le **TestRunner** lance UN SEUL **Test** et affiche son résultat
- Problèmes :
  - Comment composer les résultats des tests ?
  - Comment ré-usiner le code pour en faire le moins possible ?

40-2

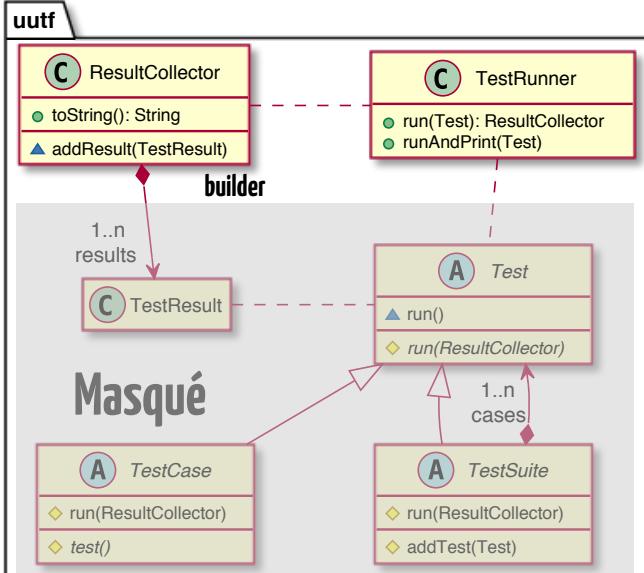
## Proposition de conception

- 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

40-1

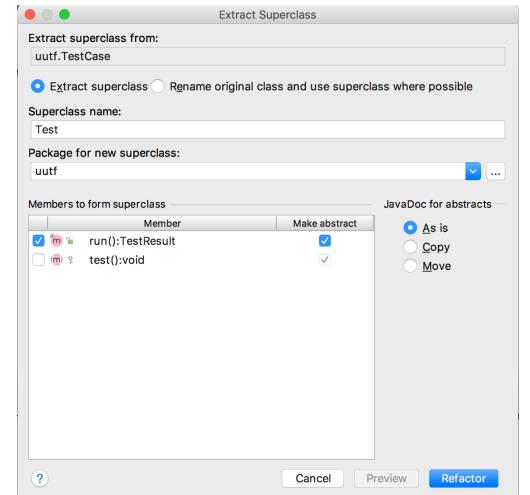


41-1



41-2

## Extraction automatique de la super-classe Test



42

```

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);
    }
}
  
```

43

44

```

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

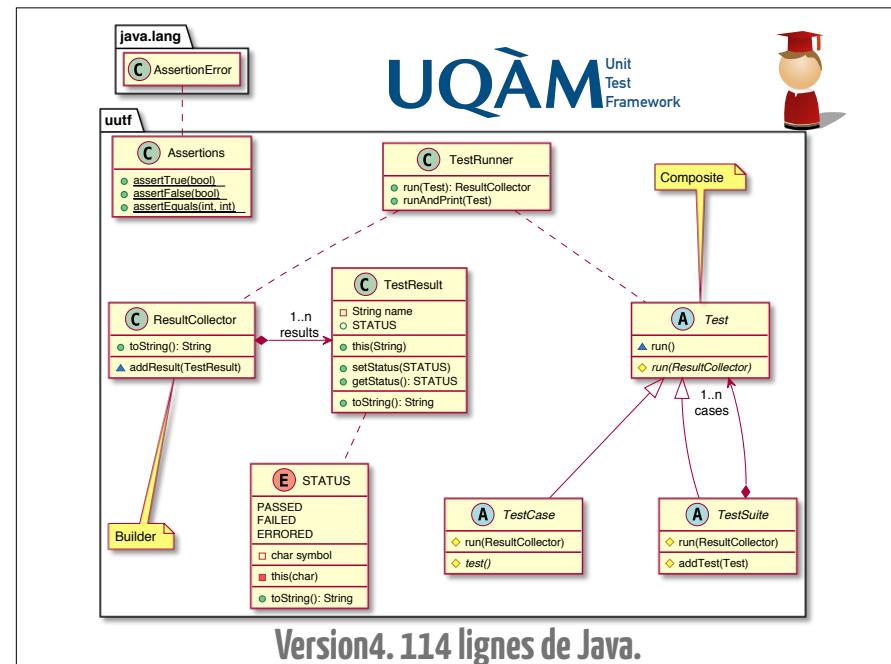
public class TestRunner {
    public void runAndPrint(Test t) {
        System.out.println(this.run(t));
    }

    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());
    }
}

```

45



46

```

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

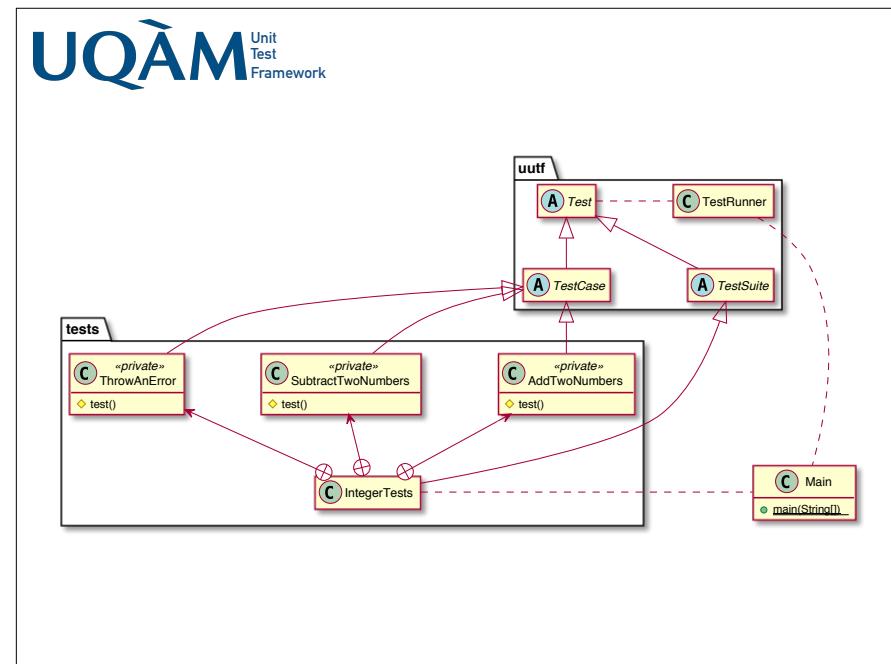
    private class AddTwoNumbers extends TestCase {
        private int x, y = 0;

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

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

```

47



48

# Mise en contexte



49

```
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)  
    }  
}
```

Junit 3.x utilise des méthodes prédéfinies

**setUp()**

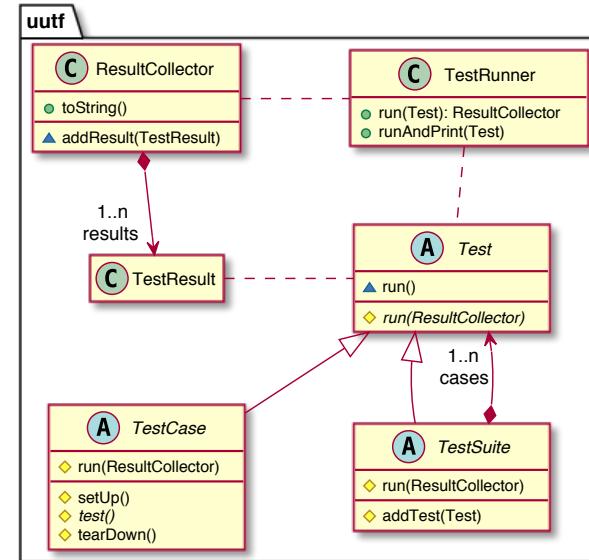
**tearDown()**

## 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 ...

50



51

52

```

public abstract class TestCase extends Test {

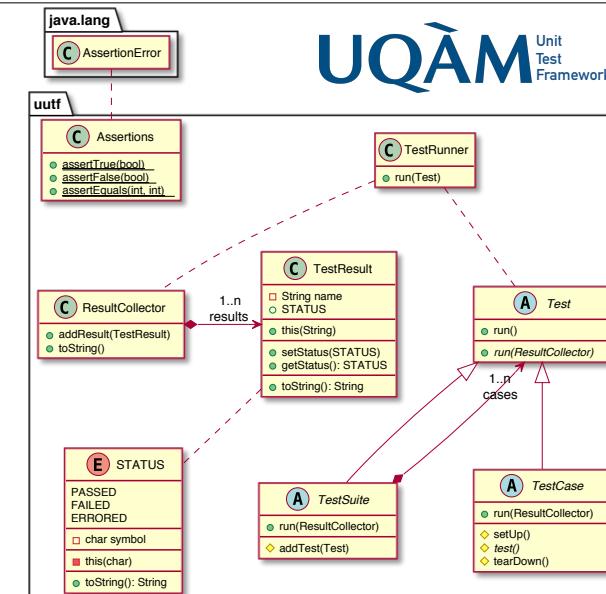
    @Override
    protected final void run(ResultCollector rc) {
        TestResult result = new TestResult(this.getClass().getCanonicalName());
        try {
            setUp();
            test();
            result.setStatus(STATUS.PASSED);
        } catch (AssertionError ae) {
            result.setStatus(STATUS.FAILED);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

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

}

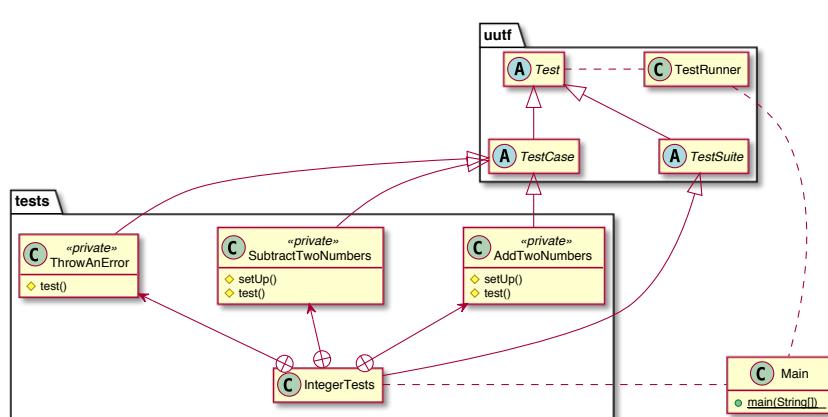
```

53



Version 5.119 lignes de Java.

54



55

```

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);
        }
    }

    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)
        }
    }
}

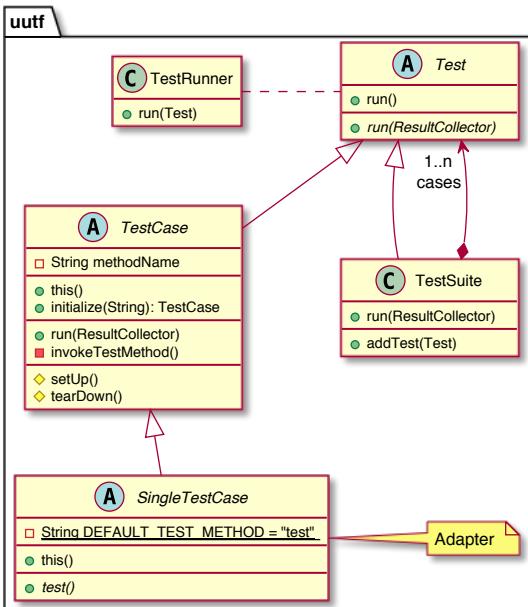
```

On y est  
presque ...

56

# Intégration Java

57



59

## 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);
    }
}
  
```

58

```

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() { }
}
  
```

60

```

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);
    }

}

```

61

```

public class IntegerTests extends TestCase {

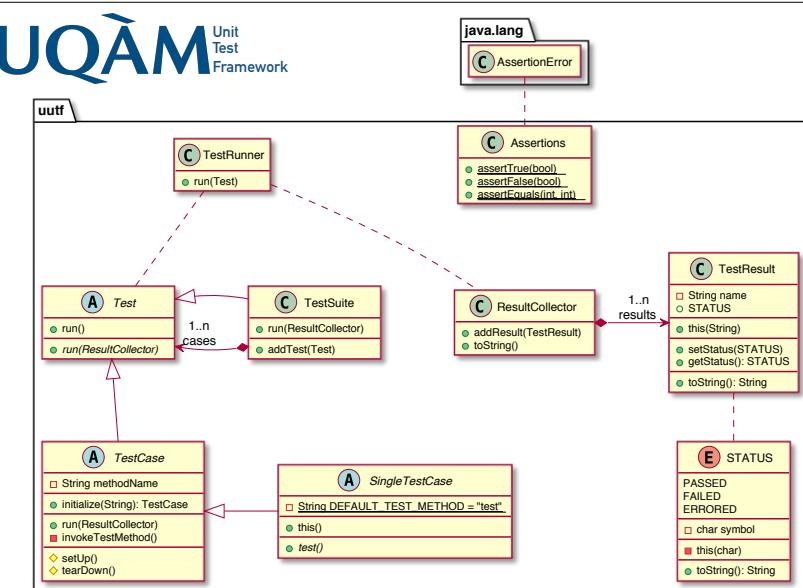
    public static Test build() {
        TestSuite suite = new TestSuite();
        suite.addTest(new IntegerTests().initialize("addTwoNumbers"));
        suite.addTest(new IntegerTests().initialize("subtractTwoNumbers"));
        suite.addTest(new IntegerTests().initialize("throwAnException"));
       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

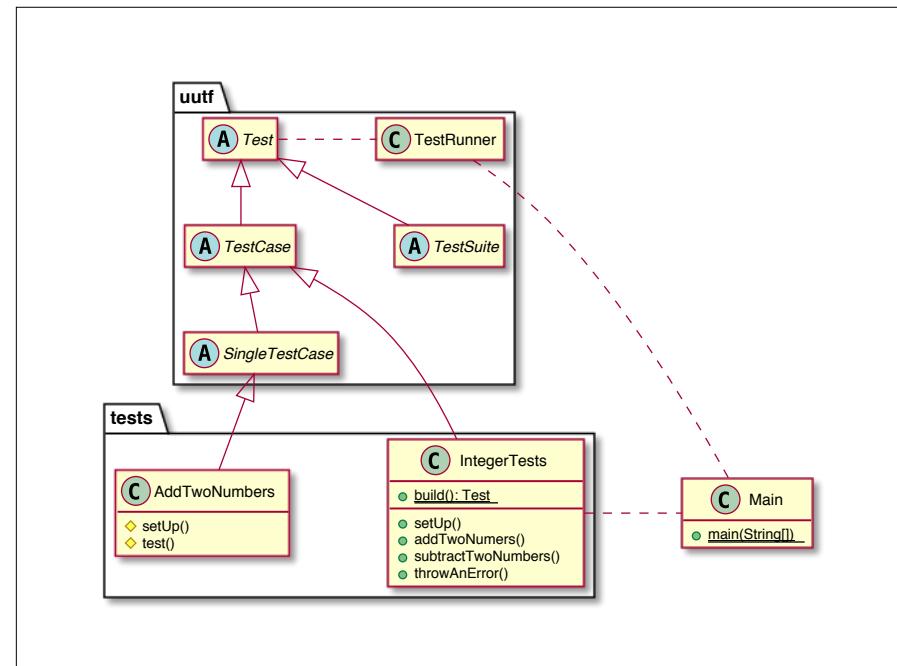
```

62



Version 6.139 lignes de Java.

63

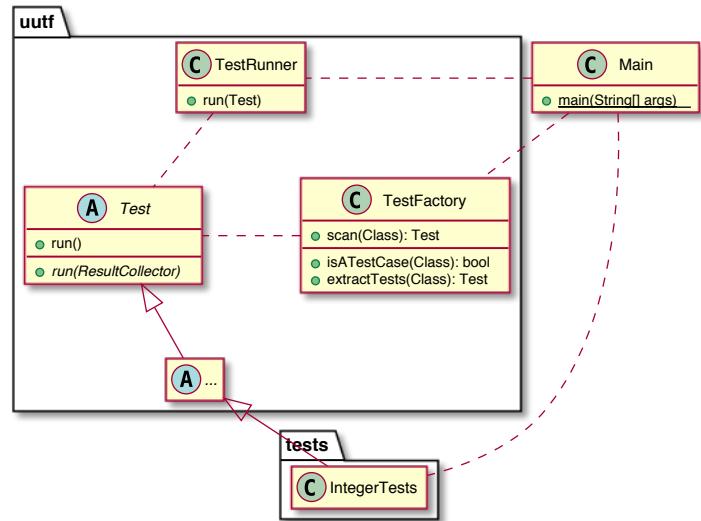


64

# Intégration Java

Le retour

65



67

## 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 ...

66

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

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

    private Test extractTests(Class klass) throws Exception {
    }

    private boolean isATestCase(Class klass) {
    }
}
```

68-1

```

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 {
        // Implementation
    }

    private boolean isTestCase(Class klass) {
        return TestCase.class.isAssignableFrom(klass);
    }
}

```

68-2

```

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 {
        TestSuite suite = new TestSuite();
        for(Method m: klass.getMethods()) {
            if (m.getName().startsWith(TEST_METHOD_PREFIX)) {
                buildTestCase(klass, suite, m);
            }
        }
        return suite;
    }

    private void buildTestCase(Class klass, TestSuite suite, Method m)
        throws Exception {
        // Implementation
    }

    private boolean isTestCase(Class klass) {
        return TestCase.class.isAssignableFrom(klass);
    }
}

```

68-3

## 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

69

```

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 {
        TestSuite suite = new TestSuite();
        for(Method m: klass.getMethods()) {
            if (m.getName().startsWith(TEST_METHOD_PREFIX)) {
                buildTestCase(klass, suite, m);
            }
        }
        return suite;
    }

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

    private boolean isTestCase(Class klass) {
        return TestCase.class.isAssignableFrom(klass);
    }
}

```

68-4

```

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;

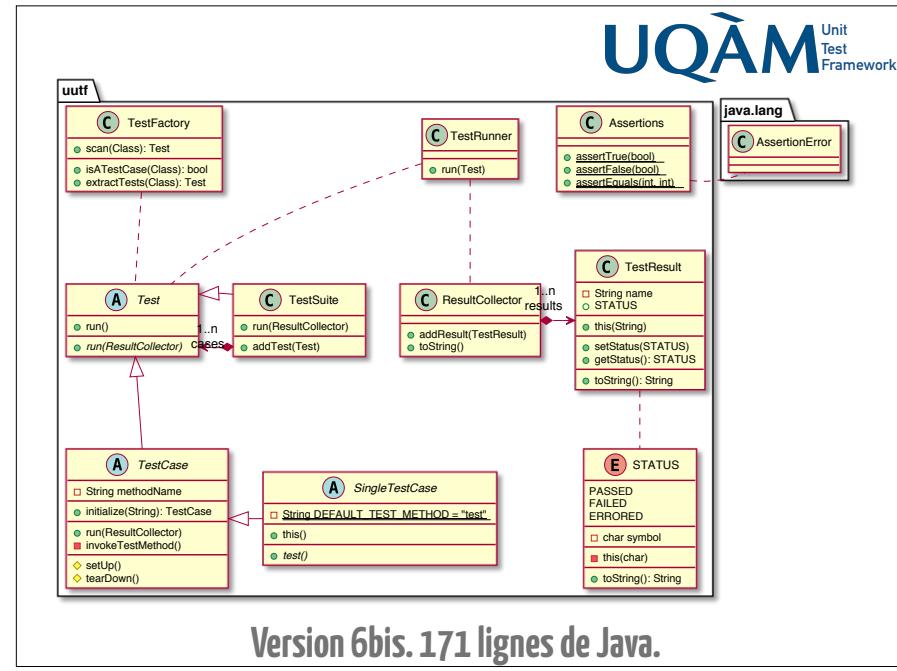
    @Override public void setUp() { x = 1; y = 1; }
    public void test_addTwoNumbers() { assertEquals(2, x+y); }
    public void test_subtractTwoNumbers() { assertEquals(1, x-y); }
    public void test_throwAnException() { throw new RuntimeException(); }
}

```

70

# Conclusion

# 8



71

```

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

```

public class IntegerTests extends TestCase {

    private int x, y = 0;

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

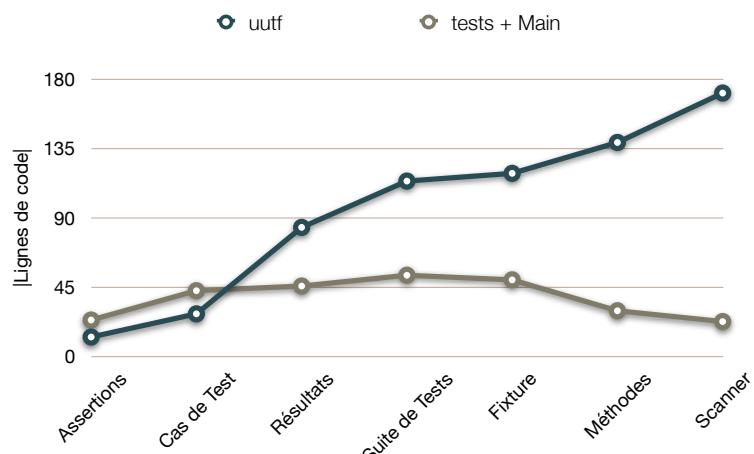
    public void test_addTwoNumbers() {
        assertEquals(2, x+y);
    }
}

```

72

73

## Taille et Livraison de valeur

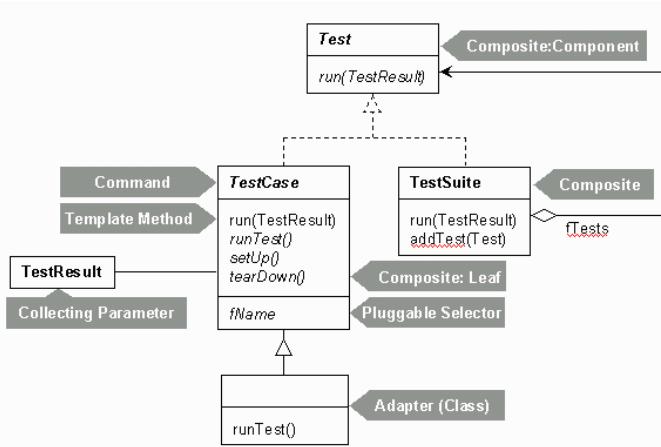


74

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

75

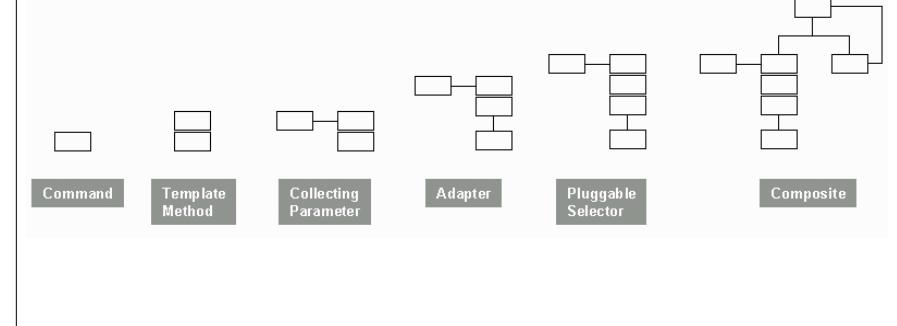
## On a un peu triché (mais pas tant que ça en fait)



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

76

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



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

77