F[20] IPOD

Tests

J.-M. Bruel -- jbruel@gmail.com -- v1.0 2020-11-29

Materials in live…📱💻

http://bit.ly/innopolis-map

https://jmbruel.github.io/InnopolisModernApplicationProduction/

Tests

Whatever development method you apply, tests are **the only way to ensure** that the delivered product **conforms to the client's requirements**.

http://douche.name/blog/nomenclature-des-tests-logiciels/

Unit Tests

They are the simplest. But hence it is needed to ...

- forget 'manual' approaches
- explore their 'limitations'
- treate both the 'qualitative' and 'quantitative' aspects

Unit Tests

Let’s consider the implementation of an abstract type:

`MatrixInt`

Unit Tests (operations)

- Operations
  - `createMatrix : Int * Int → MatrixInt`
  - `getNbLines: MatrixInt → Int`
  - `getNbColumns: MatrixInt → Int`
  - ...

Unit Tests (preconditions)

- Preconditions
  - `createMatrix(l,c) valid IF and ONLY IF ( l > 0 ) AND ( c > 0 )`
  - `getElement(m,i,j) valid IF and ONLY IF ( 0 < = i < getNbLines(m) ) AND ( 0 < = j < getNbColumns(m))`
  - …

Unit Tests (axioms)

- Axioms
  - `getNbLines(createMatrix(l,c)) = l`
  - `getNbColumns(createMatrix(l,c)) = c`
  - `getElement(createMatrix(l,c),i,j) = 0`
  - `isSquared(createMatrix(l,c)) IF and ONLY IF l = c`
  - …

Unit Tests (testing operations)

JAVA

```java
import junit.textui.TestRunner;
import junit.framework.TestSuite;
import junit.framework.TestCase;

public class MatriceEntierOperationsTest extends TestCase {
  static int totalAssertions = 0;
  static int bilanAssertions = 0;

  /*
   Types des operations du type MatriceEntier
  */
  public void test_type_new_MatriceEntier() throws Exception {
    MatriceEntier m = new MatriceEntier(3,3) ;

    totalAssertions++ ;
    assertEquals("new MatriceEntier(3,3) retourne une MatriceEntier", "MatriceEntier",
m.getClass().getName());
    bilanAssertions++ ;
  }

  public void test_type_get() throws Exception {
    MatriceEntier m = new MatriceEntier(3,4) ;

    totalAssertions++ ;
    assertTrue("getNbLignes() > 0", m.getNbLignes() > 0);
    bilanAssertions++ ;

    totalAssertions++ ;
    assertTrue("getNbColonnes() > 0", m.getNbColonnes() > 0);
    bilanAssertions++ ;

    for (int i=0; i<m.getNbLignes(); i++) {
     for (int j=0; j<m.getNbColonnes(); j++) {
```

Unit Tests (testing preconditions)

JAVA

```java
import junit.textui.TestRunner;
import junit.framework.TestSuite;
import junit.framework.TestCase;

public class MatriceEntierPreconditionsTest extends TestCase {
  static int totalAssertions = 0;
  static int bilanAssertions = 0;

  /*
   Préconditions du type Pile
  */
  public void test_precondition1() {
    MatriceEntier m ;
    boolean exception = false ;
    try { m = new MatriceEntier(0,1) ; }
    catch (Exception e) { exception = true ; };

    totalAssertions++ ;
    assertTrue("new MatriceEntier(0,1) leve une exception", exception);
    bilanAssertions++ ;

    exception = false ;
    try { m = new MatriceEntier(1,0) ; }
    catch (Exception e) { exception = true ; };

    totalAssertions++ ;
    assertTrue("new MatriceEntier(1,0) leve une exception", exception);
    bilanAssertions++ ;

    exception = false ;
    try { m = new MatriceEntier(0,0) ; }
    catch (Exception e) { exception = true ; };
```

Unit Tests (testing axioms)

JAVA

```
import junit.textui.TestRunner;
import junit.framework.TestSuite;
import junit.framework.TestCase;

public class MatriceEntierAxiomesTest extends TestCase {
  static int totalAssertions = 0;
  static int bilanAssertions = 0;

  /*
   Axiomes du type MatriceEntier
  */
  public void test_get() throws Exception {
    MatriceEntier m = new MatriceEntier(3,4) ;

    totalAssertions++ ;
    assertEquals("getNbLignes() == 3", 3, m.getNbLignes());
    bilanAssertions++ ;

    totalAssertions++ ;
    assertEquals("getNbColonnes() == 4", 4, m.getNbColonnes());
    bilanAssertions++ ;

    for (int i=0; i<m.getNbLignes(); i++) {
     for (int j=0; j<m.getNbColonnes(); j++) {
      totalAssertions++ ;
      assertEquals("getElement("+i+","+j+") == 0", 0, m.getElement(i,j));
      bilanAssertions++ ;
     }
    }

    m = new MatriceEntier(3,3) ;
    totalAssertions++ ;
    assertEquals("setPremiereDiagonale(99).getNbLignes() == getNbLignes()",
```

Unit Tests (testing additional operations)

Testing program for Additional Operations

JAVA

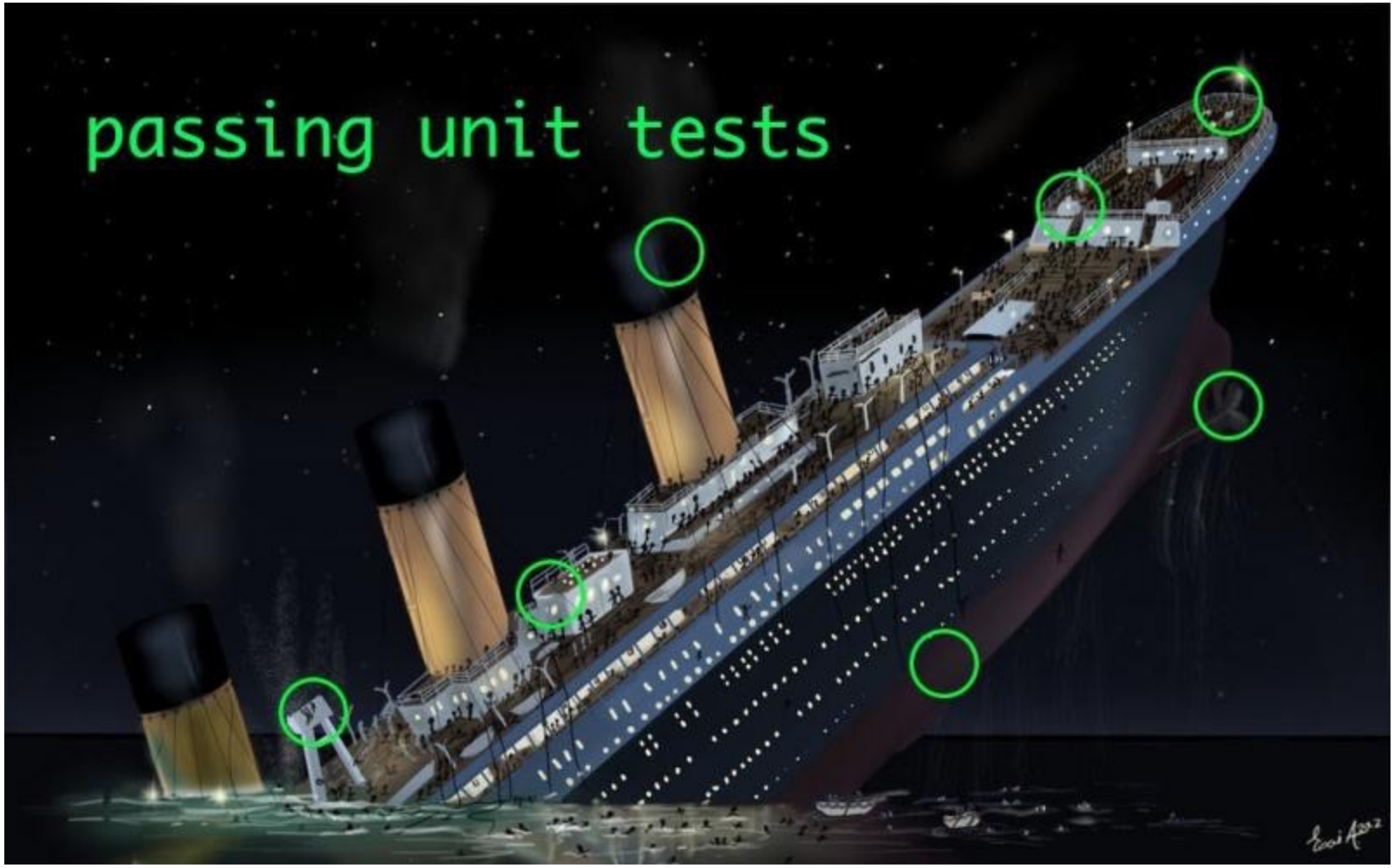
```
import junit.textui.TestRunner;
import junit.framework.TestSuite;
import junit.framework.TestCase;

public class MatriceEntierOpSupTest extends TestCase {
  static int totalAssertions = 0;
  static int bilanAssertions = 0;

  /*
   Opérations supplémentaires du type MatriceEntier
  */
  public void test_toString() throws Exception {
    MatriceEntier m = new MatriceEntier(3,3) ;
    m.setPremiereDiagonale(1).setSecondeDiagonale(2) ;

    String ln = System.getProperty("line.separator") ;
    String attendu = "1 0 2 " + ln + "0 2 0 " + ln + "2 0 1 " + ln ;
    totalAssertions++ ;
    assertEquals("toString() == ", attendu, m.toString());
    bilanAssertions++ ;
  }

  public void test_toHTML() throws Exception {
    MatriceEntier m = new MatriceEntier(3,3) ;
    m.setPremiereDiagonale(1).setSecondeDiagonale(2) ;

    String ln = System.getProperty("line.separator") ;
    String attendu = "<table border=\"1\">" + ln ;
```

Unit Tests

The testing program is :

- a **regression** detection tool
  - that should be run on each modification of the `MatrixInt` class
- a **specification** *documentation*
  - precise AND concise
- a programmers' *documentation*
  - operational

Integration Tests



Integration Tests

Integration Tests

It a more complicated activity. It is required to:

- test the client’s expectations/requirements
- test systems' interactions
- no redo the unit tests

Integration Tests

Testing program of `JourSuivantAvecLibDate.class`

JAVA

```
import junit.textui.TestRunner;
import junit.framework.TestSuite;
import junit.framework.TestCase;
import java.io.*;

public class JourSuivantAvecLibDateTest extends TestCase {
  static String programmeATester = "JourSuivantAvecLibDate" ;   1
  Process executionProgrammeATester ;                           2
  BufferedReader ecranProgrammeATester ;                        3
  BufferedWriter clavierProgrammeATester ;                      4

  String finDeLigne = System.getProperty("line.separator") ;    5

  public static void main(String[] args) {
    if ( args.length > 0 ) { programmeATester = args[0] ; }
    System.out.println("Tests du programme : " + programmeATester);
    junit.textui.TestRunner.run(new TestSuite(JourSuivantAvecLibDateTest.class)); 6
  }

  protected void setUp() throws IOException {  7
    executionProgrammeATester = Runtime.getRuntime().exec("java -cp . "+programmeATester); 8
    ecranProgrammeATester = new BufferedReader(new  InputStreamReader(
executionProgrammeATester.getInputStream() )); 9
    clavierProgrammeATester  = new BufferedWriter(new OutputStreamWriter(
executionProgrammeATester.getOutputStream() )); 10
  }

  // Saisies valides
  public void test_31_1_2013() throws IOException {
    assertEquals("Affiche : 'Saisir une date : jour mois annee ? '","Saisir une date : jour mois annee ?
" ecranProgrammeATester.readLine()); 11
```

Tests implement simple algorithms

JAVA

```java
public void test_dates_invalids() {
  int[][] tabJeuDEssaiDatesInvalids = {  (1)
      {1,1,1581},{0,1,2013},{99,99,2099},
      {32,1,2013},{29,2,2013},{32,3,2013},
      {31,4,2013},{32,5,2013},{31,6,2013},
      {32,7,2013},{32,8,2013},{31,9,2013},
      {32,10,2013},{31,11,2013},{32,12,2013},
      {29,2,1900},{30,2,2000}
  } ;
  for ( int indice = 0, taille = tabJeuDEssaiDatesInvalids.length;
        indice < taille ;
        indice = indice + 1){
    int[] date = tabJeuDEssaiDatesInvalids[indice] ;
    assertFalse(date[0]+"/"+date[1]+"/"+date[2]+" est invalid"
    , LibDate.datevalid(date[0],date[1],date[2])); (2) (3)
  }
  bilanAssertions = bilanAssertions + tabJeuDEssaiDatesInvalids.length ;
}
```

1. **given**: in the following situations
2. **when**: when we check the validity of a date
3. **then**: we should get `false`

Can everything be tested ?

- librairies
- system interactions (concurrency, etc.)
- network services
- graphical interfaces (html, java, flash, etc.)
- ...
- MAY BE NOT, but try anyway!

All the strategies are exploitable

- Write a program that run the tests
- Write the tests of a given program
- Code a feature AND add the corresponding test
- Add a test AND integrate the corresponding functionnality (TDD)
- Find back the balance between Program / Tesing Program

Most common mistakes in programming

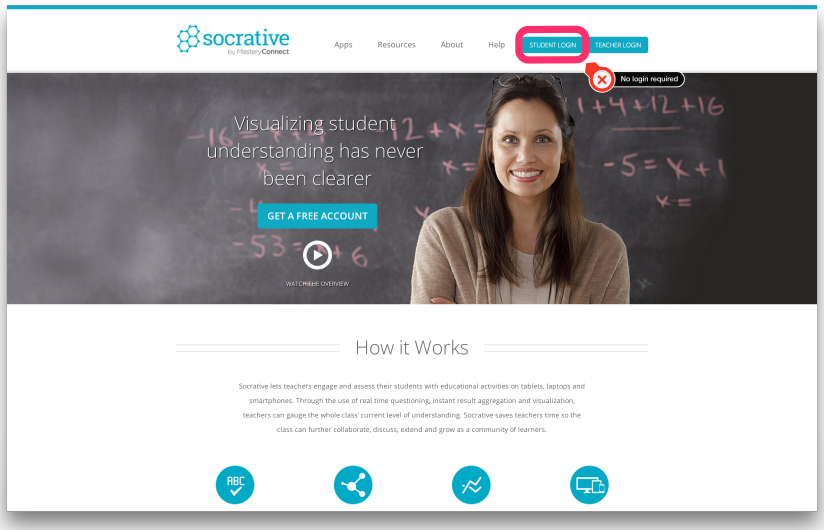
http://blog.takipi.com/the-top-10-exceptions-types-in-production-java-applications-based-on-1b-events/?utm_content=buffer0c58b&utm_medium=social&utm_source=twitter

Ready for a quizz?

Ready for a quizz?

QUESTION

- Connect on: http://www.socrative.com/ (**student login**)
- Or download the student app STUDENT
- Choose room # **JMB42**



<Thank You!>

@jmbruel

https://jmbruel.netlify.com

github.com/jmbruel