JUnit

A unit testing framework for Java classes

JUnit – useful links

- http://www.junit.org/
  JUnit homepage
- http://junit.sourceforge.net/doc/cookbook/cookbook.htm
  A cookbook for implementing tests with JUnit.
- http://www.oreilly.com/catalog/jextprockbk/chapter/ch04.pdf
  A simple tutorial
- http://junit.sourceforge.net/doc/testinfected/testing.htm
  The JUnit chapter from *Java Extreme Programming Cookbook* (O'Reilly, 2003)
- http://junit.sourceforge.net/javadoc_40/index.html
  API documentation generated with javadoc.
- http://junit.sourceforge.net/doc/faq/faq.htm
  Some frequently asked questions about using JUnit.

What is JUnit?

- JUnit is a simple, open source framework to write and run repeatable tests.
- It is an instance of the xUnit architecture for unit testing frameworks.
- JUnit features include:
  - **Assertions** for testing expected results
  - **Test fixtures** for sharing common test data
  - **Test runners** for running tests

http://junit.sourceforge.net/doc/faq/faq.htm

JUnit

- Widely used for unit testing of Java classes because:
- Test classes are easy to write and modify.
- JUnit can be used from the command line and also from an IDE.
- It is integrated in NetBeans, Eclipse, JDev, etc.
- The JUnit test classes can be run automatically in a suite.
- It has graphical and textual test runners.

How to write a test case

In JUnit 3.X

- Import junit.framework.*
- Extend TestCase
- Name the test methods with a prefix of 'test'
- Validate conditions using one of the several **assert** methods

In JUnit 4.X

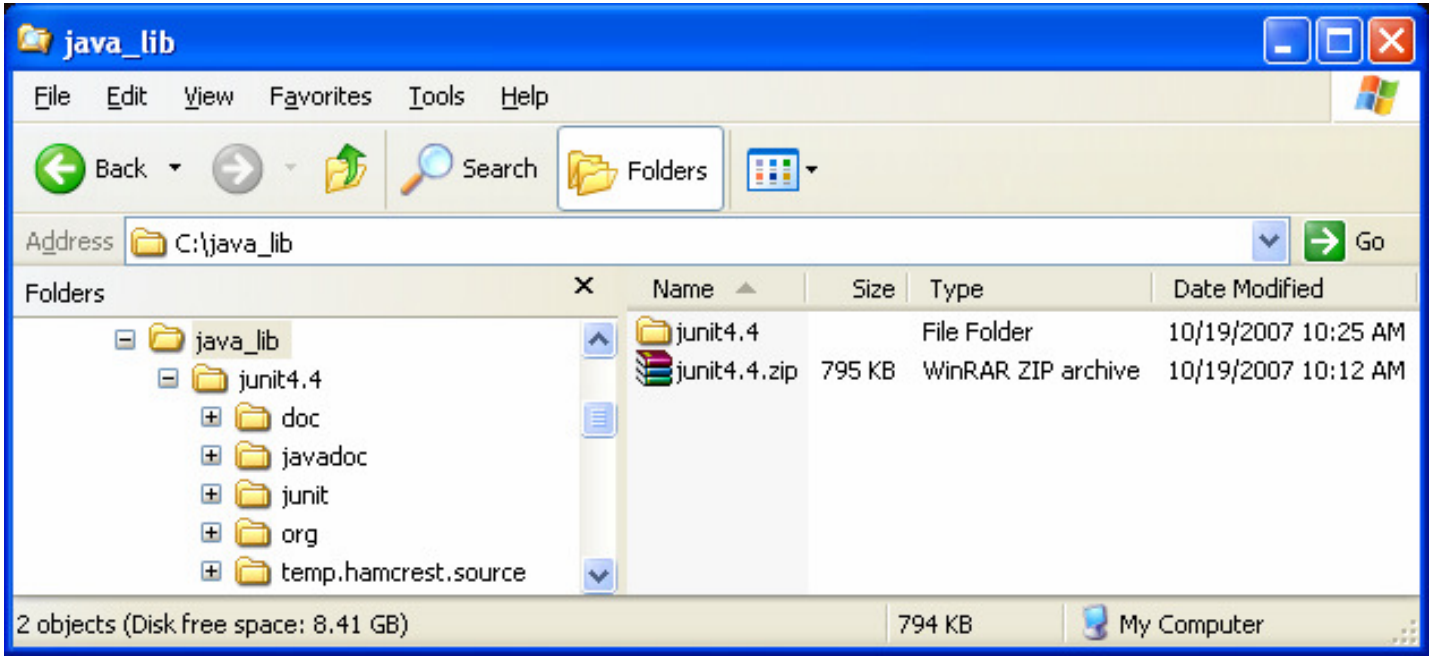
- Do not extend from Junit.framework.TestCase
- Use one of the **assert** methods
- Run the test using JUnit4TestAdapter
- **@NAME** syntax introduced

Sample assertions

- **assertEquals** (expected, actual)
- **assertEquals** (message, expected, actual)
- **assertEquals** (expected, actual, delta)
- **assertEquals** (message, expected, actual, delta)
- **assertFalse** (condition)
- **assertFalse** (message, condition)
- **assert(Not)Null** (object)
- **assert(Not)Null** (message, object)
- **assert(Not)Same** (expected, actual)
- **assert(Not)Same** (message, expected, actual)
- **assertTrue** (condition)
- **assertTrue** (message, condition)

Using JUnit from the command line

- Download JUnit from http://sourceforge.net/projects/junit/
- Unzip the junitX.Y.zip file (junit4.5.zip)



Running the tests

- My Computer → Properties → Advanced → Environment Variables → System Variables → **Path** → Edit → add the path to javac and java (for example "C:\Program Files\Java\jdk1.6.0_07\bin;")
- Modify the **Classpath** to include the path to the junit .jar archive

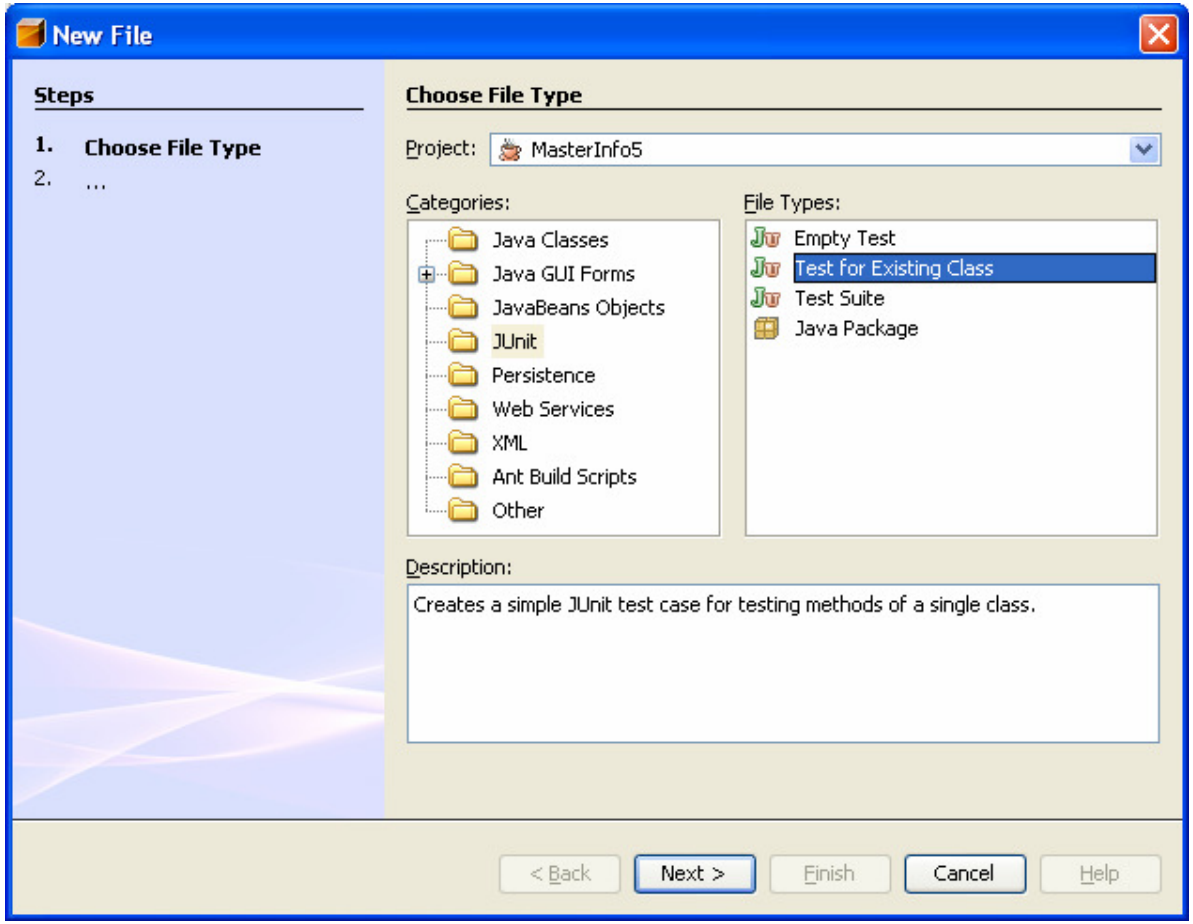
C:\WINDOWS\system32\cmd.exe

```
C:\>cd teste

C:\teste>set CLASSPATH=%CLASSPATH%;C:\java_lib\junit4.4\junit-4.4.jar;

C:\teste>javac *.java

C:\teste>java junit.textui.TestRunner TestBubbleSort
.
Time: 0

OK (1 test)


C:\teste>
```

Using JUnit from NetBeans

File → New File → JUnit →

- Empty Test
- Test for Existing class
- Test Suite



Test method automatically generated → JUnit 3.8

```
/**
 * Test of modul method, of class masterinfo5.Complex.
 */
public void testModul() {
    System.out.println("modul");

    Complex instance = null;

    double expResult = 0.0;
    double result = instance.modul();
    assertEquals(expResult, result);

    // TODO review the generated test code and remove the default
call to fail.
    fail("The test case is a prototype.");
}
```

Test method automatically generated → JUnit 4.1

```
/**
 * Test of modul method, of class Complex.
 */
@Test
public void testModul() {
    System.out.println("modul");

    Complex instance = null;

    double expResult = 0.0;
    double result = instance.modul();
    assertEquals(expResult, result);

    // TODO review the generated test code and remove the default
call to fail.
    fail("The test case is a prototype.");
}
```

Differences

In JUnit 3.X	In JUnit 4.X
import junit.framework.*;	import org.junit.After; import org.junit.AfterClass; import org.junit.Before; import org.junit.BeforeClass; import org.junit.Test; import static org.junit.Assert.*;
public class ComplexTest extends TestCase { …	public class ComplexTest { …
public void testModul() { …	@Test public void testModul() { …
	@BeforeClass, @AfterClass, @Before, @After, @Test

TestCase (JUnit 3.X)

- **protected void setUp()**
  Used to initialize the objects used for testing.
  It is called before each test method.

- **protected void tearDown()**
  Used to deallocate or reset the objects used for testing.
  It is called after each test method.

- **public static Test suite()**
  Prepares and returns a suite of tests.

Test fixtures (JUnit 4.X)

- A test fixture is the **state** of the test
  - Objects and variables that are used by more than one test
  - Initializations (*prefix* values)
  - Reset values (*postfix* values)
- Different tests can use the objects without sharing the state
- Objects used in test fixtures should be declared as instance variables
- They should be initialized in a **@Before** method
- Can be deallocated or reset in an **@After** method

Example (JUnit 3.8)

```
public static int cmmdc(int a, int b){

int c;

if (b == 0) return a;
while (b != 0) {
    c = a%b;
    a = b;
    b = c;
    }
return a;
}
```

```
public void testCmmdc() {

System.out.println("cmmdc");

int a = 12;
int b = 18;

int expResult = 6;
int result = MyClass.cmmdc(a,
    b);
assertEquals(expResult,
    result);
}
```

Test results (NetBeans)

JUnit Test Results

Statistics | Output

1 test passed, 2 tests failed.

- masterinfo5.MyMainClass5Test FAILED
  - testCmmdc1 passed (0.015 s)
  - testCmmdc2 FAILED (0.0 s)
  - testMain FAILED (0.0 s)

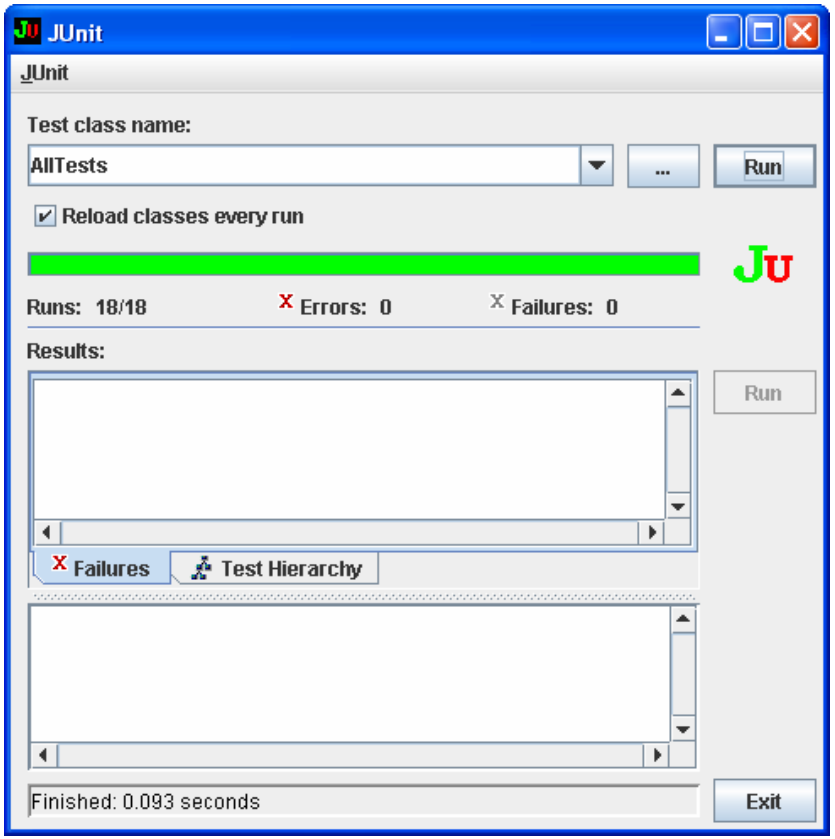
JUnit Test Results

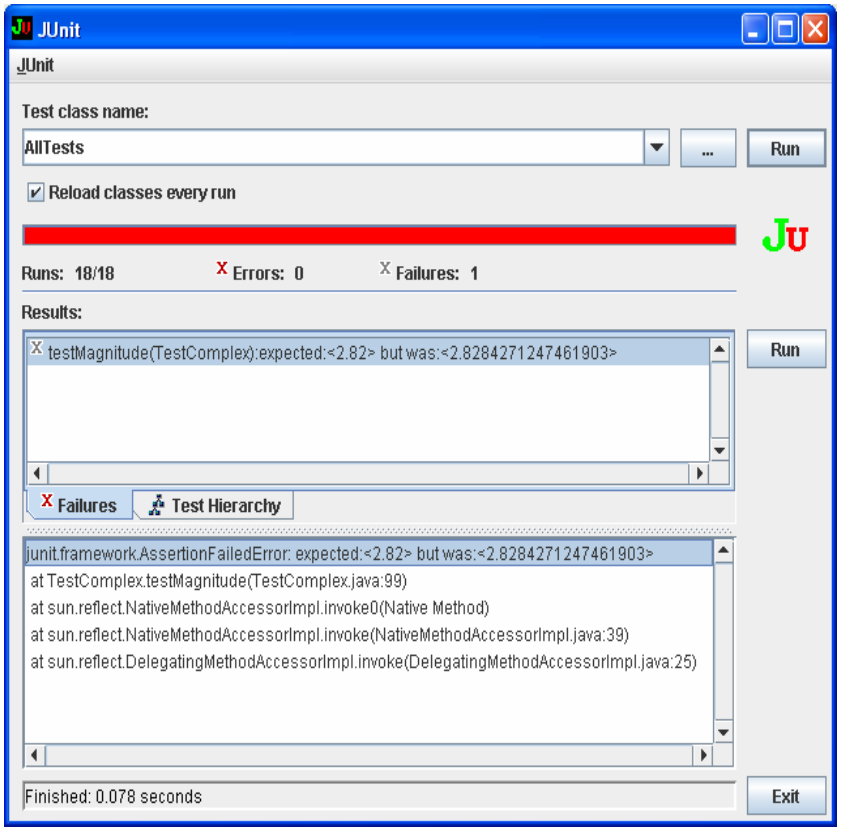
Statistics | Output

1 test passed, 2 tests failed.

- masterinfo5.MyMainClass5Test FAILED
  - testCmmdc1 passed (0.0 s)
  - testCmmdc2 FAILED (0.015 s)
    - expected:<8> but was:<6>
    - junit.framework.AssertionFailedError
    - at masterinfo5.MyMainClass5Test.testCmmdc2(MyMainClass5Test.java:53)
  - testMain FAILED (0.0 s)

Test results, using the Swing test runner for JUnit 3.8





Unit Tests Code Coverage Viewer

- http://codecoverage.netbeans.org/
- The Code Coverage Plugin is a Netbeans plugin that enhances the existing Netbeans functionality with new code coverage features.
- The plugin works as a transparent additional service that colors all java files according to the unit tests coverage information.
- With code Coverage plugin enabled user continues to work with his/her project in the usual way but can easily view the test coverage of the project classes.
- The code coverage plugin will update the code coverage data and refresh editors markup every time a unit test is executed for the project.

Unit Tests Code Coverage Viewer

Complex.java | ComplexTest.java | MyMainClass5.java | MyClass.java | Welcome | MyMainClass5Test.java

```
    public int getImag() {
        return imag;
    }

    public double modul() {
        return Math.sqrt(real * real + imag * imag);
    }

    public static Complex suma(Complex c1, Complex c2) {
        return new Complex(c1.real + c2.real, c1.imag + c2.imag);
    }

    public static Complex produs(Complex c1, Complex c2) {
        return new Complex(c1.real * c2.real - c1.imag * c2.imag, c1.real * c2.imag + c1.imag * c2.real);
    }

    public boolean equals(Object o) {
        if (!(o instanceof Complex)) {
            return false;
        }
        Complex other = (Complex) o;
        return real == other.real && imag == other.imag;
    }

}
```

Partial coverage

55:28 INS

Coverlipse

- http://coverlipse.sourceforge.net/index.php
- Coverlipse is an Eclipse plugin that visualizes the code coverage of JUnit tests.
- Features:
  - All Uses Coverage
  - Block Coverage (Statement Coverage)
  - *Branch Coverage (Not so soon to come)*
- Just one test run is needed for evaluation of all coverage criteria
- Easy way to include/exclude packages from the test
- Direct feedback in the Eclipse Java Editor
- Explanation of the results in specialized views

Installing Coverlipse

- In Eclipse, click Help → Software Updates → Find and Install.
- Select “Search for new features to install” → Next.
- Add a “New Remote Site”, with a name, e.g. "Coverlipse update site“ and the URL http://coverlipse.sf.net/update/
- Finish.
- Eclipse installs the Coverlipse plugin.
- Restart Eclipse

TestHello.java | BubbleSort.java | TestBubbleSort.java | TestComplex.java | Complex.java | Computation.java | ComputationTest.java

```
1  public class BubbleSort
2  {
3
4      public static int[] sort(int[] a)
5      {
6          int i, j, t;
7
8          for (i = a.length - 1; i > 0; i--)
9              for (j = 0; j < i; j++)
10             {
11                 if (a[j] > a[j + 1])
12                 {
13                     t = a[j];
14                     a[j] = a[j + 1];
15                     a[j + 1] = t;
16                 }
17             }
18         return a;
19     }
20
```

JUnit | Coverlipse Class View | Coverlipse Markers View | Console | Problems | Search | Progress | Call Hierarchy | Show definitions

Content Description of CoverageMarkerView

Message	Line	covered uses	uncovered uses	Resource
Definition of the variable j	9	9 9	11 11 13 14 14 15	BubbleSort.java
This use was covered	9			BubbleSort.java
Definition of the variable name	11	12		TestBubbleSort.java
This use was not covered	11			BubbleSort.java
Definition of the variable t	13		15	BubbleSort.java
This use was not covered	13			BubbleSort.java
This use was not covered	14			BubbleSort.java
This use was not covered	15			BubbleSort.java
Definition of the variable a	38	40		TestBubbleSort.java
Definition of the variable b	40	41 42 43 44		TestBubbleSort.java

Java - Computation.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help

JUnit | Coverlipse Class View | Coverlipse Markers View | Console | Problems | Search | Progress | Call Hierarchy | Show definitions

Content Description of CoverageMarkerView

Message	Line	covered uses	uncovered uses	Resource
Definition of the variable arg2	5	6		Computation.java
Definition of the variable arg1	5	6 9		Computation.java
Definition of the variable result	6	8 14	11	Computation.java
Definition of the variable myInt	7	14		Computation.java
Definition of the variable result2	8	13		Computation.java
Definition of the variable result3	13			Computation.java
Definition of the variable m	18	20 20		Computation.java
Definition of the variable n	18	22		Computation.java
Definition of the variable result	19	22 24		Computation.java
Definition of the variable j	20	20 20		Computation.java
Definition of the variable j	20	20 20		Computation.java
Definition of the variable result	22	22 24		Computation.java
Definition of the variable arg1	28		29	Computation.java
Definition of the variable arg2	28		29	Computation.java
Definition of the variable result	29		30	Computation.java
Definition of the variable i	35	38		Computation.java
Definition of the variable i	42			Computation.java
Definition of the variable e	44			Computation.java
Definition of the variable i	48			Computation.java
Definition of the variable divident	53	54 60		Computation.java
Definition of the variable divisor	53	60		Computation.java
Definition of the variable result	56		62 63	Computation.java
Definition of the variable result	60	62 63		Computation.java
This use was not covered	62			Computation.java
Definition of the variable result2	62			Computation.java
This use was not covered	63			Computation.java
Definition of the variable a	68	71 73		Computation.java
Definition of the variable i	69	69 71		Computation.java
Definition of the variable i	71	71 69		Computation.java
Definition of the variable <unknown>	6			ComputationTest.java
Definition of the variable addResult	10	11		ComputationTest.java
Definition of the variable divideResult	21	22		ComputationTest.java

Definition of the variable n

JCoverage

- http://cms.jcoverage.com/
- Eclipse plugin (not free)
- Features:
  - Line coverage
  - Branch coverage
  - LOC, number of methods
  - Cyclomatic complexity (1-10 low, 11-20 moderate, 21-50 high, 51+ very high)
  - Viewer integrated with Eclipse;
  - Can export statistics in HTML

jcoverage report - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address C:\Documents and Settings\Raluca\Desktop\jcoverage report\master.BranchLine.html

Coverage Report

jcoverage | summary | package | file

Overall	files	lines	%line	indicator	%branch	indicator
jcoverage summary for master.BranchLine	1	13	38%		50%	

Source

Line	Hits	remainder
1		`package master;`
2		
3		`public class BranchLine {`
4		
5		`public static int fct(int a)`
6		`{`
7	1	`int s=0;`
8	1	`if (a>0)`
9		`{`
10	0	`s = s+1;`
11	0	`s = s+2;`
12	0	`s = s+1;`
13	0	`s = s+2;`
14	0	`s = s+1;`
15	0	`s = s+2;`
16	0	`s = s+1;`
17	0	`s = s+2;`
18		`}`
19		`else`
20		`{`
21	1	`s = -35;`
22		`}`
23	1	`return s;`
24		`}`

Done My Computer

Thank you!