

JUnit

**ITEC316 Software Engineering**

**Homework Assignment**

**S. Eser Ozvataf  
100178**

# JUnit

#### What Is JUnit?

JUnit is an unit-testing framework which is written for Java Platform. It’s completely an open source software and located in <http://www.junit.org/>. It has own user interface to test written source code in case of having bugs or problems.

#### Brief Information About Unit Testing

In Software Engineering, unit testing is a widely used method which verifies correctness of specific units by measuring method’s outputs.

Unit testing usually broken down into modules which enables the tester can observe each part’s functionality. It is extremely helpful to find out regressions and either dissatisfied or problematic outputs of program parts.

Unit Testing is a continuous software development process which is not limited just for testing phase. It can be applicable for both procedural (classic approach) and object oriented programming. Most of senior developers suggests writing test cases during early phases of development is better. It also has the role in maintenance, even planning and designing phases.

#### Test-Driven Development

Test-Driven development is a sample of extreme programming and shows us how unit testing can be done in designing phase. Steps of classic test-driven development are so:

* Defining the test cases during the design phase.
* Development of code which satisfies the expected outputs of test cases.
* Run the unit tests. If there are some bugs, fix them.
* Decide on new implementations (same in agile development) and define test cases for them.
* Repeat these steps until next version requirements has finished.

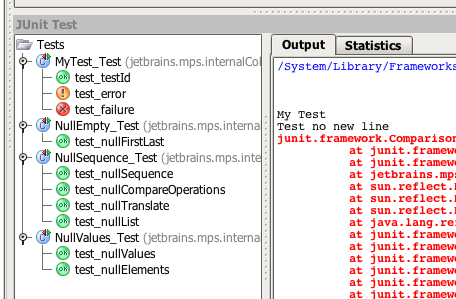
It can be applicable in different variations the key part of this approach is “testing comes first”.

#### Usage of JUnit: Graphical Runners

Some Java IDEs have built-on JUnit support called “graphical runner”. Independent or built-on editions don’t differ in functionality and usage for JUnit.

In graphical runners, JUnit displays the written test cases in a tree list and when tests are started, indicators of test cases turn into a colored icon which is depending on the related tests result.

Green icon means test passed OK.  
Yellow icon for errors and problems.  
And red one for major mistakes and failures.



intellij ide with junit graphical runner

#### Usage of JUnit: Console Runners

JUnit also has a Java package (org.junit.runner.JUnitCore) which can be referenced into a java application. By writing few lines of code test cases can be tested in object oriented environment.

#### Writing Tests

Tests cases must be a part of code produced. By referencing JUnit package to specific project, JUnit’s related classes can be usable.

Each test should be derived from TestCase class. It can be considered like “category” or “module”. In test cases each class method represents a test.

|  |
| --- |
| import java.io.File;  import junit.framework.\*;  import junit.extensions.\*;  public class ThingTester extends TestCase  {  public ThingTester(String name)  {  super(name);  }  public void testGetName() throws Exception  {  String fileSpec = new String("c:xxxyyyzzz.txt");  assertEquals("zzz.txt", getName(fileSpec));  }  }  public class Program {  public static void main(String[] args)  {  TestRunner.runAndWait(new TestSuite(ThingTester.class));  }  } |

a sample junit test case written in java

#### Running Tests

During the tests, a little dot will appear if all test cases is passed; otherwise, an assertion failure and a stack trace will let you know where the problem is.

#### Test Methods And Assertions

Assertions are the helper methods which reports expected conditions to the unit-testing framework. During the execution of test methods assertions checks some variables if they satisfies output expectation. Therefore unit testing framework indicates the specified unit is passed the test or not.

|  |  |
| --- | --- |
| Statement | Description |
| fail(String) | Let the method fail. Might be used to check that a certain part of the code is not reached. Or to have failing test before the test code is implemented. |
| assertTrue(Boolean) | Will always be true / false. Can be used to predefine a test result, if the test is not yet implemented. |
| assertTrue([message], Boolean) | Checks that the boolean condition is true. |
| assertsEquals([String message], expected, actual) | Tests that two values are the same. Note: for arrays the reference is checked not the content of the arrays. |
| assertsEquals([String message], expected, actual, tolerance) | Test that float or double values match. The tolerance is the number of decimals which must be the same. |
| assertNull([message], object) | Checks that the object is null. |
| assertNotNull([message], object) | Checks that the object is not null. |
| assertSame([String], expected, actual) | Checks that both variables refer to the same object. |
| assertNotSame([String], expected, actual) | Checks that both variables refer to different objects. |

a table of supported assert functions in junit

#### JUnit Alternatives

JUnit is the defacto standard in unit testing for Java platform. It also ported to other programming platforms such as FlexUnit for Actionscript, AUnit for AUnit, CUnit for C, CPPUnit for C++, fUnit for Fortran, DUnit for Delphi, FPCUnit for FreePascal, HUnit for Haskell, JSUnit for Javascript, NUnit for .NET, OCUnit for Objective-C, PHPUnit for PHP, PyUnit for Python and QTestLib for Qt.  
  
A complete list of JUnit alternatives can be found at various sources on the internet. But most famous ones for Java platform are so:

* Selenium: <http://seleniumhq.org/>
* Mockito: <http://mockito.org/>
* JTiger: <http://www.jtiger.org/>
* SureAssert: <http://www.sureassert.com/>
* TestNG: <http://testng.org/>

#### References

[1] JUnit Homepage  
<http://junit.org>

[2] Junit Alternatives: Testing Frameworks  
<http://geeknizer.com/junit-alternatives-testing-frameworks/>

[3] Unit Testing, Wikipedia  
<http://en.wikipedia.org/wiki/Unit_testing>