**SHOPPING BASKET ASSESSMENT**

**Useful Hints (2)**

**Unit Testing the Classes**

In order to ensure the correctness of the major classes, they should be unit tested.

You should develop a test plan, to show the unit tests that are to be carried out (see the form attached.

Then produce and run JUnit tests to prove that the methods in the classes are working .

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Plan** | | | | | |
| **Software Component Name:**  **Date:** | | **Version No:**  **Tester Name:** | | **Page No:** | |
| **Test Number** | **Purpose/Type of test** | **Steps** | **Expected Behaviour** | **Actual Behaviour** | **Test**  **Passed** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

JUnit – a review

# Introduction

The goal of this topic is to give an introduction to the **JUnit framework** that enables the reader to:

* Have a basic understanding of what JUnit is
* Implement test classes in JUnit 4
* Test exceptions using JUnit

**JUnit** is a framework that helps in the automation of unit tests for Java classes. It was created by Kent Beck and Erich Gamma.

Similar tools for languages other than Java are NUnit for the .NET framework,

CppUnit for C++, and PyUnit for Python.

The JUnit framework. Version 4 is the latest release of JUnit (as of writing). It is backward compatible with the previous version JUnit 3.8.

This new version was mainly released to take advantages of the annotations capabilities(JSR 175 [2]) introduced with Java 5.0.

JUnit plugins are available for several IDE like Eclipse, Netbeans, etc.

In this tutorial, we will not use any IDE for the demonstrations. We will run and collect test results from the command line.

# Steps involved in performing Unit Tests

1. **Prepare (or set up) the test environment to fulfill conditions that must be met, according to the test plan.** This means to define and set prefix values. e.g. initialize fields, turn on logging, etc.
2. **Execute the test case.** This means, executing (exercising) the part of the code to be tested. For that we use some test inputs (test case values), according to the test plan.
3. **Evaluate the results or side effects generated by the execution of the test case** to check if they match what has been defined in the test plan.
4. **Clean up (or tear down) the test environment if needed** so that further testing activities can be done, without being influenced by the previous test cases. We deal here with postfix values.

# Realization of the unit testing steps in JUnit4

Here we describe how the steps given in section 2 are to be implemented in the JUnit

framework.

The dfferent steps are implemented in a Java test class. The methods of

the test class receive specific annotations to make them known to the framework as

such:

1. Prepare (or set up) the test environment: the different actions to realize this are to be implemented in one or several methods, annotated with @Before. Those methods are then executed before each test case (test method).

@Before

public void setUp ( ) {

s = new Sample ( );

}

1. Execute the test case and evaluate the results (or side effects): This is done by exercising the code under test with test values within a (test) method. Each method annotated with @Test will be executed as a test case by JUnit. Evaluation of the expected results (or side effects) is done in the same method using assertions.

@Test

public void testAddition ( ) {

int a = 3 , b = 6 ;

int expectedOutput = ( a+b ) ;

int r e s = s . Addition ( a , b ) ;

assertEquals ( expectedOutput , r e s ) ;

}

1. Clean up (or tear down) the test environment is done in one or several methods, that are run after execution of each test method. To belong to this group, a method has to be annotated with @After.

@After

public void tearDown ( ) {

s = null ;

}

# Running Test Cases and Collecting the Results

The JUnit 4 framework provides the class org.junit.runner.JUnitCore (a so called

Test Runner) in order to run the tests and collect their results. The tests can be run from

Eclipse

The results are printed to the console:

JUnit version 4.8.1

....I

Time: 0.006

OK (4 tests)