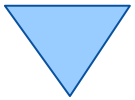
"ooRexxUnit: A JUnit Compliant Testing Framework for ooRexx Programs"

2006 International Rexx Symposium

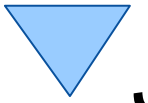
Austin, Texas, U.S.A. (April 2006)

Rony G. Flatscher (Rony.Flatscher@wu-wien.ac.at)
Wirtschaftsuniversität Wien, Austria (http://www.wu-wien.ac.at)



Agenda

- JUnit
- Architecture
- Examples
- Testing Open Object Rexx (ooRexx)
  - Needs many test cases
  - Rexx community's help needed !
- Roundup and Outlook



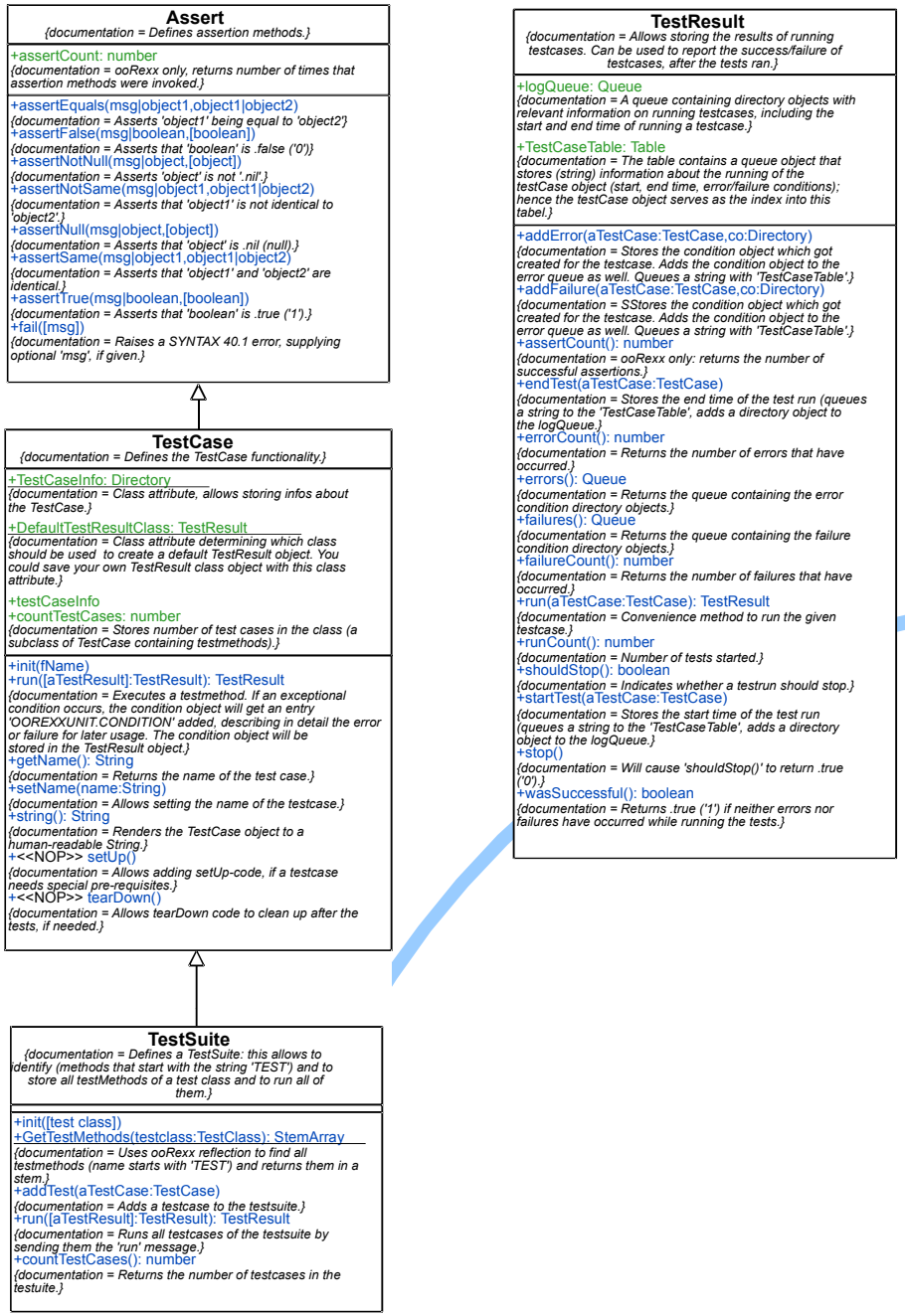
JUnit

- JUnit
  - www.junit.org
- An application framework
  - Creating, running tests and log the results
  - Defines classes
    - "Assert"
    - "TestCase"
    - "TestSuite"
    - "TestResult"

ooRexxUnit.cls
Public Classes & Routines

- Public classes
  - Assert
  - TestCase
  - TestSuite
  - TestResult
- Public routines
  - makeDirTestInfo(aTestCaseClass, arrLines)
  - simpleDumpTestResults(aTestResult, title)
  - makeTestSuiteFromFileList(testCaseFileList [, ts])

Architecture - Overview
ooRexx Classes



Architecture – Assert Class ooRexx

Assert

{documentation = Defines assertion methods.}

+assertCount: number
{documentation = ooRexx only, returns number of times that assertion methods were invoked.}

+assertEquals(msg|object1,object1|object2)
{documentation = Asserts 'object1' being equal to 'object2'}
+assertFalse(msg|boolean,[boolean])
{documentation = Asserts that 'boolean' is .false ('0')}
+assertNotNull(msg|object,[object])
{documentation = Asserts 'object' is not '.nil'.}
+assertNotSame(msg|object1,object1|object2)
{documentation = Asserts that 'object1' is not identical to 'object2'.}
+assertNull(msg|object,[object])
{documentation = Asserts that 'object' is .nil (null).}
+assertSame(msg|object1,object1|object2)
{documentation = Asserts that 'object1' and 'object2' are identical.}
+assertTrue(msg|boolean,[boolean])
{documentation = Asserts that 'boolean' is .true ('1').}
+fail([msg])
{documentation = Raises a SYNTAX 40.1 error, supplying optional 'msg', if given.}

Architecture - TestCase
ooRexxUnit.cls

TestCase

{documentation = Defines the TestCase functionality.}

+TestCaseInfo: Directory

{documentation = Class attribute, allows storing infos about the TestCase.}

+DefaultTestResultClass: TestResult

{documentation = Class attribute determining which class should be used to create a default TestResult object. You could save your own TestResult class object with this class attribute.}

+testCaseInfo
+countTestCases: number

{documentation = Stores number of test cases in the class (a subclass of TestCase containing testmethods).}

+init(fName)
+run([aTestResult]:TestResult): TestResult

{documentation = Executes a testmethod. If an exceptional condition occurs, the condition object will get an entry 'OOREXXUNIT.CONDITION' added, describing in detail the error or failure for later usage. The condition object will be stored in the TestResult object.}

+getName(): String

{documentation = Returns the name of the test case.}

+setName(name:String)

{documentation = Allows setting the name of the testcase.}

+string(): String

{documentation = Renders the TestCase object to a human-readable String.}

+<<NOP>> setUp()

{documentation = Allows adding setUp-code, if a testcase needs special pre-requisites.}

+<<NOP>> tearDown()

{documentation = Allows tearDown code to clean up after the tests, if needed.}

Architecture - TestSuite
ooRexxUnit.cls

TestSuite

{documentation = Defines a TestSuite: this allows to identify (methods that start with the string 'TEST') and to store all testMethods of a test class and to run all of them.}

+init([test class])

+GetTestMethods(testclass:TestClass): StemArray

{documentation = Uses ooRexx reflection to find all testmethods (name starts with 'TEST') and returns them in a stem.}

+addTest(aTestCase:TestCase)

{documentation = Adds a testcase to the testsuite.}

+run([aTestResult]:TestResult): TestResult

{documentation = Runs all testcases of the testsuite by sending them the 'run' message.}

+countTestCases(): number

{documentation = Returns the number of testcases in the testuite.}

Architecture - TestResult

ooRexxUnit.cls

TestResult

{documentation = Allows storing the results of running testcases. Can be used to report the success/failure of testcases, after the tests ran.}

+logQueue: Queue

{documentation = A queue containing directory objects with relevant information on running testcases, including the start and end time of running a testcase.}

+TestCaseTable: Table

{documentation = The table contains a queue object that stores (string) information about the running of the testCase object (start, end time, error/failure conditions); hence the testCase object serves as the index into this tabel.}

+addError(aTestCase:TestCase,co:Directory)

{documentation = Stores the condition object which got created for the testcase. Adds the condition object to the error queue as well. Queues a string with 'TestCaseTable'.}

+addFailure(aTestCase:TestCase,co:Directory)

{documentation = SStores the condition object which got created for the testcase. Adds the condition object to the error queue as well. Queues a string with 'TestCaseTable'.}

+assertCount(): number

{documentation = ooRexx only: returns the number of successful assertions.}

+endTest(aTestCase:TestCase)

{documentation = Stores the end time of the test run (queues a string to the 'TestCaseTable', adds a directory object to the logQueue.}

+errorCount(): number

{documentation = Returns the number of errors that have occurred.}

+errors(): Queue

{documentation = Returns the queue containing the error condition directory objects.}

+failures(): Queue

{documentation = Returns the queue containing the failure condition directory objects.}

+failureCount(): number

{documentation = Returns the number of failures that have occurred.}

+run(aTestCase:TestCase): TestResult

{documentation = Convenience method to run the given testcase.}

+runCount(): number

{documentation = Number of tests started.}

+shouldStop(): boolean

{documentation = Indicates whether a testrun should stop.}

+startTest(aTestCase:TestCase)

{documentation = Stores the start time of the test run (queues a string to the 'TestCaseTable', adds a directory object to the logQueue.}

+stop()

{documentation = Will cause 'shouldStop()' to return .true ('0').}

+wasSuccessful(): boolean

{documentation = Returns .true ('1') if neither errors nor failures have occurred while running the tests.}

A Nutshell Example
ooRexxUnit @ Work

```
aTR=.TestResult~new                              /* create a TestResult     */
.NutshellTestUnit~new('someTestCaseMethod')~run(aTR)
.NutshellTestUnit~new('this does not exist!!')~run(aTR)
.NutshellTestUnit~new('test.ABS')~run(aTR)       /* this should execute o.k.*/
call simpleDumpTestResults aTR                   /* dump results            */

::requires ooRexxUnit.cls     -- load the ooRexxUnit classes

::class "NutshellTestUnit" subclass TestCase public

::method "test.ABS"           -- a TestCase method
   self~assertEquals("subTest1",  ABS('12.3'),    12.3)
   self~assertEquals("subTest2",  ABS(' -0.307'), 0.307)

::method "someTestCaseMethod"
   a=" RexxLA   "
   b="RexxLA"
   self~assertEquals("testing for equality using 'assertEquals()'...", a, b)
   self~assertSame("testing for identity using 'assertSame()'...", a, b)
```

```
Output:
nr of test runs:             3
nr of successful assertions: 3
nr of failures:              1
   [20060410 18:09:37.565000]: [failure]   testCase: [someTestCaseMethod]
     (a NutshellTestUnit@1A8CF21F), msg=[testing for identity using 'assertSame()'...
@assertFailure assertSame: expected=[ RexxLA  ], received=[RexxLA]]
nr of errors:                1
   [20060410 18:09:37.565000]: [error]     testCase: [this does not exist!!]
     (a NutshellTestUnit@C69AF21F) ---> [NOMETHOD ERROR The NIL object]
```

ooRexxUnit.cls Additional Resources, 1

- *.testUnit
  - ooRexxUnit.cls reference implementations
  - ooRexx programs having a structure
    - Comments can be used in reports
      - name":" value
    - Determines optionally mandatory tests to be carried out
    - Tests can be run on their own
  - Other programs can take advantage of it
    - runAllTests.rex [-r] [directory]

ooRexxUnit.cls Samples

- ooRexx.Base.BIF.testUnit
  - oorexxtest\oorexxunit\ooRexx\base\BIF.testUnit
- ooRexx.Base.Class.String.testUnit
  - oorexxtest\oorexxunit\ooRexx\base\class\String.testUnit
- ...

ooRexxUnit.cls
Additional Resources, 2

- *.runTests
  - Example of denoting and running testUnits of a directory
  - e.g. ooRexx.Base.runTests
  - oorexxtest\oorexxunit\ooRexx\base\BIF.testUnit

TestCases for Open Object Rexx

- Needs many test cases
  - Testing every function
  - Testing every class
- Rexx community's help needed !
  - Too many areas need testcases
  - No one person could create all needed testcases
  - Continued need for creating new testcases
- Acquired knowledge helps yourself
  - Creating testcases while creating applications

Creating a ooRexxUnit Standardized TestUnit Program, 1

- Advised parts
  - Comment block at the top of the program, giving explanations on the program in the form of "name: explanation" sequences
  - Initialization part
    - Define the classes in the program that have testcase methods defined
      - Start with the string "test"
    - Define a list of the names of the mandatory testmethods in the file
      - Empty list means all testmethods are regarded mandatory

Creating a ooRexxUnit Standardized TestUnit Program, 2

- Example: ooRexx.Base.String.testUnit

```
/*
   name:             ooRexx.Base.String.testUnit
   author:           Rony G. Flatscher
   date:             2005-08-07
   version:          0.0.2

-- line commented lines are ignored, when building the directory of infos from this header
   changed:          2005-08-21, ---rgf, renamed the testUnit-class to match exactly the
                                 file name, i.e. a dot (.) between class name and "TestUnit";
                                 added convention to return a list of arrays containing the
                                 test case class object and an optional list of mandatory test
                                 case methods from it

   languageLevel:    6.0
   purpose:          Test the String functions.
   remark:           Initial test unit for demonstration purposes, needs to be completed.

   license:          CPL 1.0 (Common Public License v1.0, see below)
   link:

   category1:        ooRexx
   category2:        Base
   category3:        String
*/
```

Creating a ooRexxUnit Standardized TestUnit Program, 3

- Lines to adapt:

```
-------------------------------------------------------------------------------
-- ===> adapt the "testUnitList" to your testCase classes; each element in the list is <===
-- ===> an array object, the first element containing the testCase class object, the   <===
-- ===> second element is a list of test method names which are regarded to be         <===
-- ===> mandatory (if the list remains empty all test methods are mandatory)           <===

   /* list of array objects, each containing the testUnit class object and an
      optional list of mandatory test case methods name                        */
mandatoryTestMethods=.list~new   -- no mandatory tests for this testCase class
testUnitList=.list~of( .array~of(.ooRexx.Base.String.testUnit,  mandatoryTestMethods) )


-------------------------------------------------------------------------------
-- ===> the following code needs not to be individualized                          <===
```

Creating a ooRexxUnit Standardized TestUnit Program, 4

- Walkthrough of initialisation code

```
   -- read top comment, containing infos about this program
arrLines=.array~new
do i=1 to 150 until arrLines[i]="*/"
   arrLines[i]=sourceline(i)
end
      -- supply information for the testClass(es) in this file; the class attribute
      -- "TestCaseInfo" (a directory object, index points to a queue) will store
      -- the parsed infos
aTestUnitClass=testUnitList~at(testUnitList~first)[1] -- get first testClass

   -- will parse the array lines and store result in class object
call makeDirTestInfo aTestUnitClass, arrLines
tmpDir=aTestUnitClass~TestCaseInfo
parse source s   -- op_sys invocationType fullPathToThisFile
tmpDir~setentry("testCase-source", s)

   -- now add this directory to other testCase classes, if any left
do arr over testUnitList
   if arr[1]=aTestUnitClass then iterate  -- already handled
   arr[1]~TestCaseInfo=tmpDir             -- save info in class object
end

-- if this file is CALLed or REQUIRED then define an entry "bRunTestLocally" in .local
-- and set it to .false; this way the independent local invocation of the tests is inhibited
if .local~hasentry("bRunTestsLocally")=.false then
   .local~bRunTestsLocally=.true -- if this file is executed directly, then run tests
```

Creating a ooRexxUnit Standardized TestUnit Program, 5

- Walkthrough of initialisation code (continued)

```
if .bRunTestsLocally=.true then  -- run ALL tests in this test unit
do
   ts=.testSuite~new             -- create a testSuite
   do arr over testUnitList
      -- create a testSuite for the given test case class, use all its testmethods
      ts~addTest( .testSuite~new(arr[1]))
   end
   -- testResult=.testSuite~new(testUnitClass)~run
   testResult=ts~run       -- now run all the tests

   call simpleDumpTestResults testResult
end

   /* return list of array objects containing test case classes and
      optionally list of mandatory test methods                      */
return testUnitList

::requires ooRexxUnit.cls     -- load the ooRexxUnit classes

::class "ooRexx.Base.String.testUnit" subclass TestCase public

   -- test the ABBREV BIF, using examples from the documentation
::method "testABBREV"

   word="Print"
   self~assertEquals("subTest1", ABBREV(word, "Pri"),     .true)
...
```

Creating a ooRexxUnit Standardized TestUnit Program, 6

- Standardized TestUnit programs
  - Allow executing all tests by default, if invoking the .testUnit Rexx program on its own
  - Allows executing all tests under the control from other programs, e.g.
    - ooRexx.Base.runTests
    - runAllTests.rex
- One ("central") TestResult object could be used to log the running of all selected testUnit programs

Roundup and Outlook

- ooRexxUnit
  - Modelled after JUnit
  - Every programmer who is accustomed to JUnit can immediately apply the knowledge
  - Allows creating, invoking and managing arbitrary testCases (= testmethods) in testUnit programs
  - Creating and collecting ooRexx testUnit programs allows testing ooRexx itself
  - With the help of the community we might establish a comprehensive set of tests
  - **A start: the OoRexxUnit-Hackathon !!!**