Group 12

SER 216-20179

Iteration #3

April 14, 2017

Functional Testing:

JFunc 1.1

Executive Summary

The software engineering team requires the ability to test further than unit testing of the Plotter program. By default all tests are fatal in jUnit, but the team needs the ability to continue the test execution even when one (or more) assertions fail. This where we can use functional testing, which involves black box testing and is not concerned with the source code of the application. Each and every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results. JFunc 1.1 is provided as an extension of JUnit of which the team is already familiar with. This will allow for easy integration into the team's current testing.

Origins of JFunc

The framework for JFunc was developed in 2002 by Shane Celis and supported by Terrespring, Inc. JFunc was used extensively in the development and testing of Terraspring’s products, in late 2002 Terraspring was acquired by Sun Microsystems, Inc. JFunc is open source and available for free from sourceforge.net.

Features of JFunc

JFunc is primarily intended for developing functional tests. One of its main features (as mentioned previously) is the ability to handle multiple failures. Other features include using one test subject for a series of tests, rather than one test instance per test. The ability for test methods to accept arguments. Concise, typesafe, suite construction using proxies. JFunc also provides an enhanced test runner allowing verbose assertions and the ability to pass arguments to the suite constructor.

Using JFunc

As JFunc framework is provided as an extension to JUnit, it can easily be used by using “JFunc” jar. To start using JFunc, all you need to do is extend your test class to junit.extensions.jFunc.JFuncTestCase base class instead of junit.framework.TestCase.

Summary

JFunc provides an easy transition for the team into functional (black box) testing. As the team is already familiar with JUnit, extending its use can be done efficiently (and free).