**The Benefits of using JUnit**

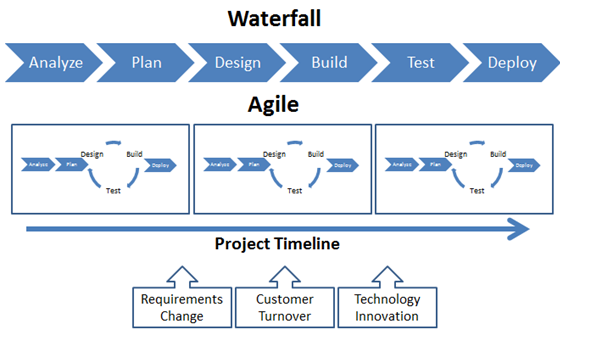
Software bugs are costly in terms of time and money. To minimise these problems, it would be advisable and makes sense that developers should regularly test their code. A developer can test their code in a number of ways: One of the most and commonly adopted practices is to test the newly written code once on the fly by running the program using the main method and manually matching the output with the desired results. This is only done once.

This is one time testing is not ideal because in the real world, requirements keep changing and most of the time developers continually modify the existing functions and classes. So the already written piece of code will not be tested every time.

By using the JUnit framework however, software developers can write a test cases with real simplicity. Developers can run it automatically all the time when their build gets completed or any time and in seconds it will provide you with a detailed report as to the status of the application.

Such a powerful tool for developers and has many benefits to developers and end users alike. The benefits of using JUnit include:

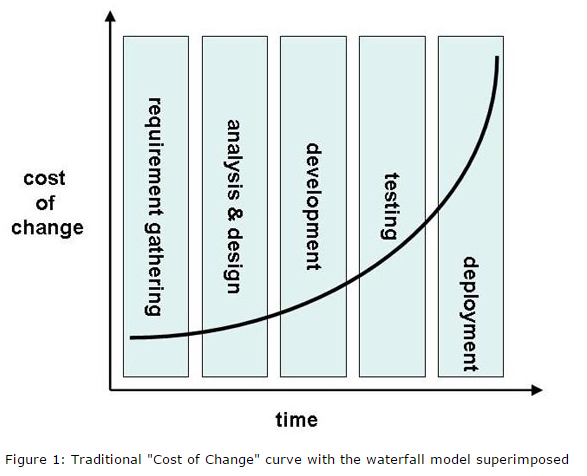
* Using JUnit, software developers ensure that every piece of the software being developed is tested before module and systems even before they commence module or system level testing.
* Junit allows the developer to go back and make modifications to the code in response to changing customer requirements without compromising any older functionality of the software. This ensures that every time the code is modified it will work as expected.
* New members of of the software development team will easily understand the test cases written in JUnit. This allows them the opportunity to contribute to writing more test cases with the resulting software more robust and stronger.
* JUnit is so popular among developers, it has become the standard testing framework for testing Java applications and is supported by almost all IDE's including Net Beans, Eclipse and JCreator. As it is supported, it is easy to use by simply selecting the relevant options in the IDE and viewing the test results immediately.
* JUnit allows the developers to easily create and manage a unit test case suite for the entire software in development.
* The process of continually developing your code is part of the Agile methodology in software development. This process, known as Test driven development (TDD), allows the development of quality software applications in less time and at significantly less cost than would otherwise be possible if the developers had adopted a more rigid approach such as Waterfall. So, JUnit can effectively help reduce costs associated with development and increase efficiency.
* By using JUnit, developers can ensure that all the units in an application have been tested for optimal functionality, both individually and in synergy with each other.

**http://diginimo.com/wp-content/uploads/2014/09/agile-and-waterfall.png**

The above diagram illustrates the differences between the Waterfall methodology and the Agile methodology, of which JUnit is an important tool. As outlined above, testing is a core element of the Waterfall approach in contrast to Waterfall, where it is only utilised prior to deployment.

**Reduced Costs**

The costs to developers of changing software whether due to inherent problems in the code or changing customer requirements can be considerable. Typically such costs can be much higher with a more rigid approach than with an approach such as Agile. So it makes sense to developers to test regularly. In this way Junit, if utilised correctly can reduce such costs to the software developers.



**Source: http://www.methodsandtools.com/archive/archive.php?id=20**

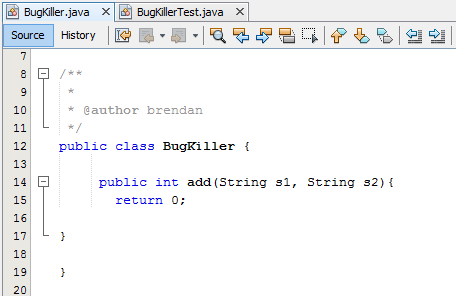
**Limitations**

* As good as JUnit is, and it is a powerful tool, developers need to remember that like all tools it needs to be used properly. The measure of how successful the developers will be in minimising bugs and potential problems will be down to how well the testing is implemented. It is up to the developers to correctly conceive and apply the tests. JUnit is not a magic wand and it is up to the developers to use it correctly to realise its full potential.
* Incorrectly conceived or applied tests may result in units that do not perform as well as expected in the real world. Even if all units perform as well as expected in isolation and in all scenarios conceived by the developers, the end users may ultimately use the software differently and different scenarios may arise that the developers hadn't conceived.
* The final results of using JUnit and the ultimate robustness and quality of the software are only as good as the tests that have been carried out and how thorough the developers were with that testing.

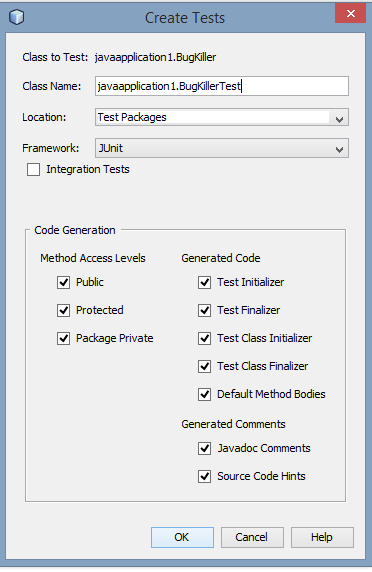
**A Simple Test in NetBeans using JUnit**

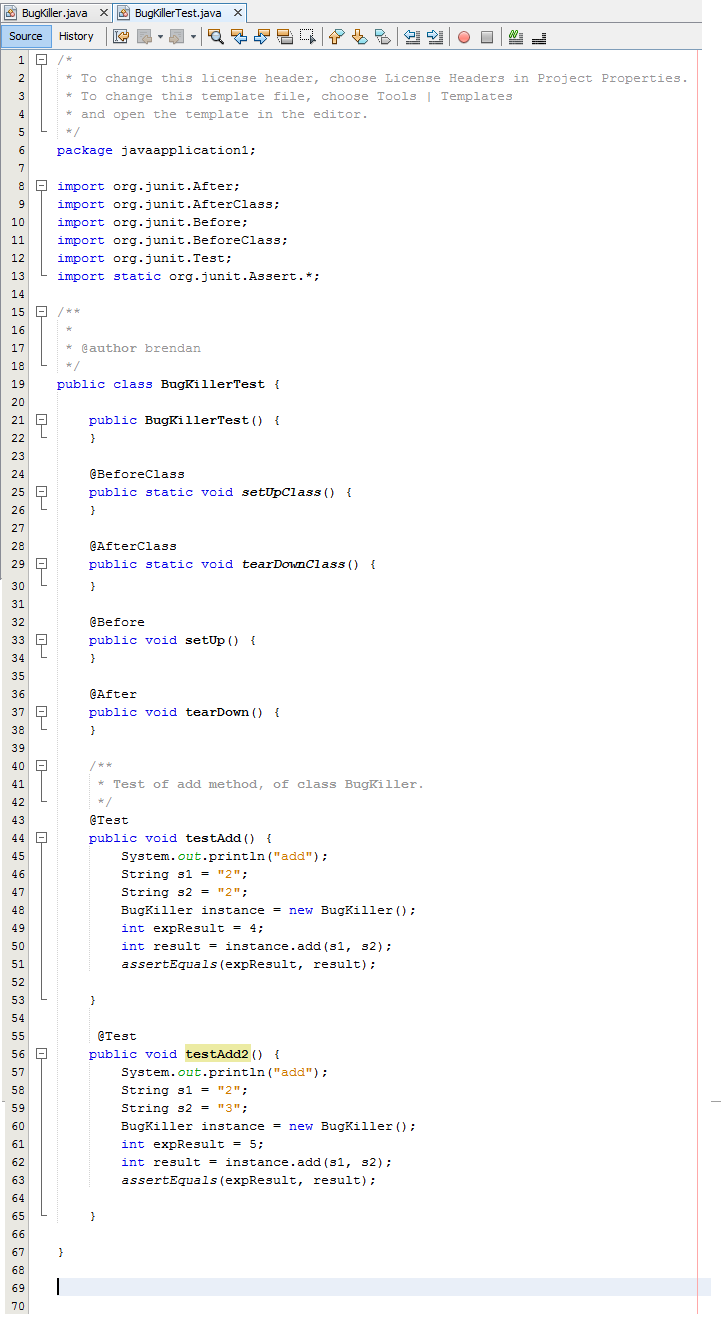
The below test will highlight the importance of testing your code with a testing framework such as JUnit. It will show the ease at which one can implement JUnit in the NetBeans IDE.

**Original Code (I want to test this):**



**Let's Run JUnit -This Creates Separate Class:**

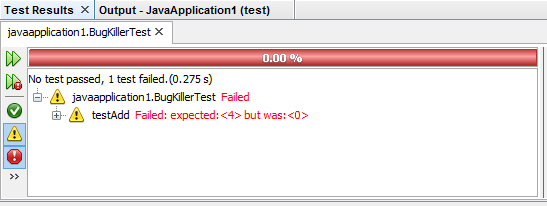




**Default is a call to Fail:**



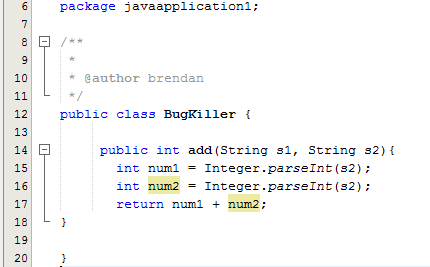
**Result: Fail!**



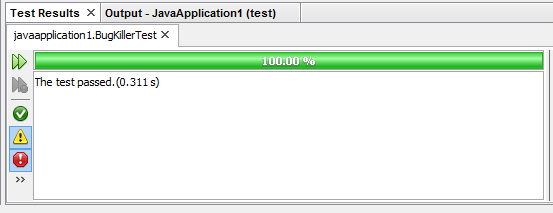
It expected a result of 4 but was 0 (in the method being tested)

**Now we need to modify the code in the test case so that it passes the test:**

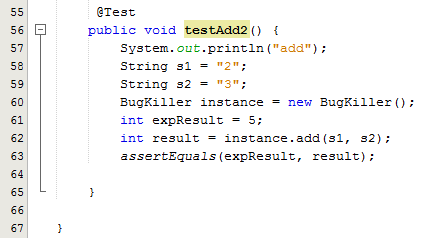
**Convert String Variables to Integer Variables, should return num1+num2:**



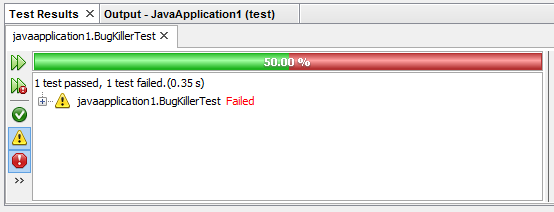
**Run The Test: Passed!**



**Time to run another test:**



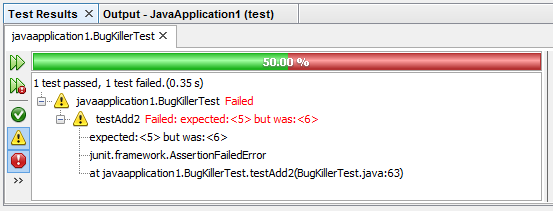
**Run The Test: Failed!**



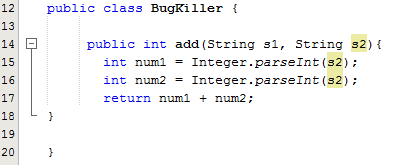
**Let's see why this test failed:**

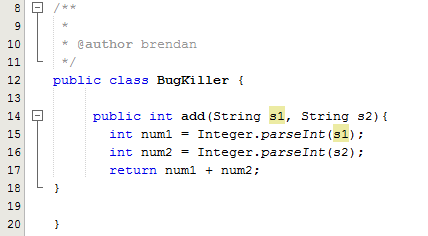
**Based on the first test, we were led to the false belief that our code worked, but the results of this additional test has shown us otherwise. This is why it is important to run as many tests as possible to ensure your code is bug free!**

**Let's go back to the original code, find the problem and fix it!**

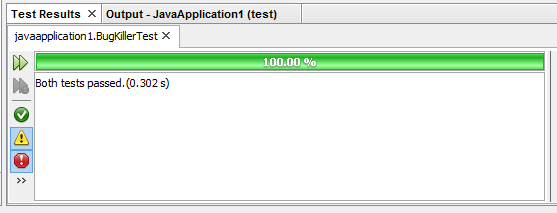


**The problem was I was doing some cutting and pasting and failed to notice I had put in the s2 String twice as my parameter, so of course it was going to fail!!**





**Ok, I've fixed it now! Let's run another test!**



**No Bugs!!**

**Conclusion:**

**Testing is an important part of software development, the above example highlights the need for testing. If I had not tested the code, I might not have noticed the error. By using this approach, I can be confident as to the quality of the code. The chance of a developer spotting errors in larger applications would be smaller again, so we believe by testing your code with JUnit, you will save much time and heartache by testing as you code!**