

Writing a Test Case


Writing a Test Case

Test cases are crucial for debugging your code. Test cases can also verify that changes to the codebase do not break existing/debugged/working functionality.

This tutorial shows how to write a JUnit test case using the Netbeans IDE.

Test Case Skeleton

Automatic Test Case generation

 You can automatically generate a test class with complete test cases in Netbeans. With the class open in the editor, go to the "Tools" menu and select "Create JUnit Tests" or use the shortcut Ctrl+Shift+U. Alternatively, you can right-click on a class in the Files browser and follow the same path, "Tools -> Create JUnit Tests".

Below is minimal code that can be used as a template for writing org.junit test cases. It is not meant as an example, working test case but as an illustration of the essential parts of a JUnit test case. Each section will be covered in detail.

```
import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;

public class TestCaseExample extends TestCase
{
    public void TestCaseExample() throws Exception
    {}

    public static Test suite()
    {
        return new TestSuite(TestCaseExample.class);
    }

    protected void setUp() throws Exception
    {
        // DO SETUP HERE
    }

    public void testIt() throws Exception
    {
        // DO TEST HERE
    }
}
```

First, three JUnit classes need to be imported.

```
import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;
```

TestCase is the base class for writing tests using the JUnit package.

Test and *TestSuite* provide classes for the framework to create testable objects from your test case.

All test cases need to extend the *TestCase* class so that JUnit knows that the class provides a test case.

```
public class TestCaseExample extends TestCase
```

The constructor for this class should be public, or JUnit cannot access it.

```
public void TestCaseExample() throws Exception
```

The *suite()* function is necessary boilerplate for JUnit. Replace "TestCaseExample" with the actual name of your class.

Test cases in `org.lcsim` need to implement this function as follows.

```
public static Test suite()
{
    return new TestSuite(TestCaseExample.class);
}
```

If your test case requires setup for all its test functions, implement this function, which will be called before any of the tests are executed.

```
protected void setUp() throws Exception
```

Finally, implement individual test cases by writing functions that start with the word "test". JUnit will automatically execute any public function starting with this string.

```
public void testIt() throws Exception
```

Writing Test Statements

The easiest way to make a test is throwing an error when a result does not match an expected quantity.

In Java 1.5, the `assert` statement can be used.

```
assert( results == expected );
```

In this case, an exception will be thrown if the two variables are not equal. Both results and expected will be printed if this error occurs.

Alternately, an exception can be thrown directly.

```
if ( results != expected ) {
    throw new RuntimeException("test failed");
}
```

If any method or constructor called by your test case throws an exception, the test will fail unless the exception is caught and ignored. In general, test methods should propagate exceptions upward so that failure occurs, unless the exception does not indicate an error. This means that test methods should be declared to throw exceptions.

File and Directory Structure

Maven establishes an organization for tests. These go into the directory `tests` in the project root area. The test should have the same package as the class it is testing in order that classes in the package can be accessed easily without a lot of import statements. All classes that extend `junit.framework.TestSuite` will be executed when Maven builds the project.

Accessing Test Data

Class Resources

Test data can be embedded into the test classes package directory and then accessed as a stream. (This is built-in Java functionality.)

For example, here is an example of accessing an XML file as a stream, based on `org.lcsim.geometry.GeometryReaderTest` in the `GeomConverter` package.

```
InputStream in = GeometryReaderTest.class.getResourceAsStream( "GeometryReaderTest.xml" );
GeometryReader reader = new GeometryReader();
detector = reader.read(in);
```

This technique should be used for data that is small (< 1 MB). Generally, we only embed text files such as XML into the jar files.

File Cache

For data that is too large to embed into the `org.lcsim` jar file, the file cache can be used to download and access files.

```
import org.lcsim.util.cache.FileCache;
```

This shows how to download an LCIO file from an example URL.

```
URL url = new URL("http://www.example.com/mytestdata/someTestData.slcio");  
FileCache cache = new FileCache();  
File file = cache.getCachedFile(url);
```

Now use the Java File object as usual, as it should point to a local file on the user's system.

Actual test data is kept at <http://www.lcsim.org/datasamples>, which can be browsed. Your test data can also be added here. Contact jeremy@slac.stanford.edu if you want to upload your own test data.

Using the Event Loop

Many test cases need to use the org.lcsim event loop to execute a driver over some events.

Assuming that the *file* variable points to a cached Lcio file, an event loop can be setup and executed as follows.

```
LCSimLoop loop = new LCSimLoop();  
loop.setLCIORRecordSource(file);  
loop.add(new MyTestDriver());  
loop.loop(-1);  
loop.dispose();
```

Replace "MyTestDriver" with an actual driver you are trying to test.

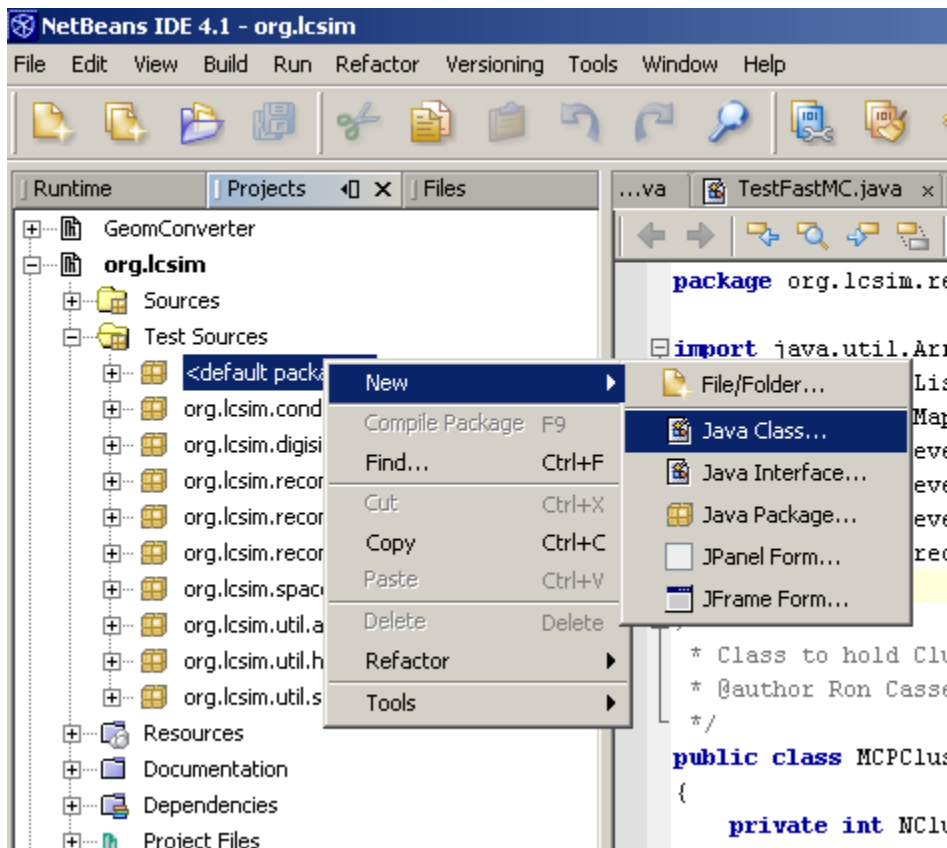
Passing -1 to the loop command will process all the events in the Lcio file.

A Real Example

For an example test case, look at *HitPositionTest* in the org.lcsim package. It illustrates a number of the concepts covered in this document.

Adding from Netbeans

To add a test case from Netbeans, expand the project navigation window to the "Test Sources" folder. Right-click on the package where the test case should be added. In this case, the base package area is used, but you should use the actual package of the class being tested.



Now type in the name of the class and click "Finish". Follow the template given previously to write the actual test case.