S8384

Test Driven Development Using JUnit

1. The Project

The aim of this lab is to demonstrate Test Driven Development by developing a simple program to calculate Fibonacci numbers.

From wikipedia:

> In mathematics, the Fibonacci numbers are a sequence of numbers named after Leonardo of Pisa, known as Fibonacci, whose *Liber Abaci* published in 1202 introduced the sequence to Western European mathematics.
>
> The sequence is defined by the following recurrence relation:
>
> $$F(n) := \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F(n-1) + F(n-2) & \text{if } n > 1. \end{cases}$$
>
> That is, after two starting values, each number is the sum of the two preceding numbers. The first Fibonacci numbers, also denoted as F_n, for n = 0, 1, … , are:
>
> > 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657, 46368, 75025, 121393, ...

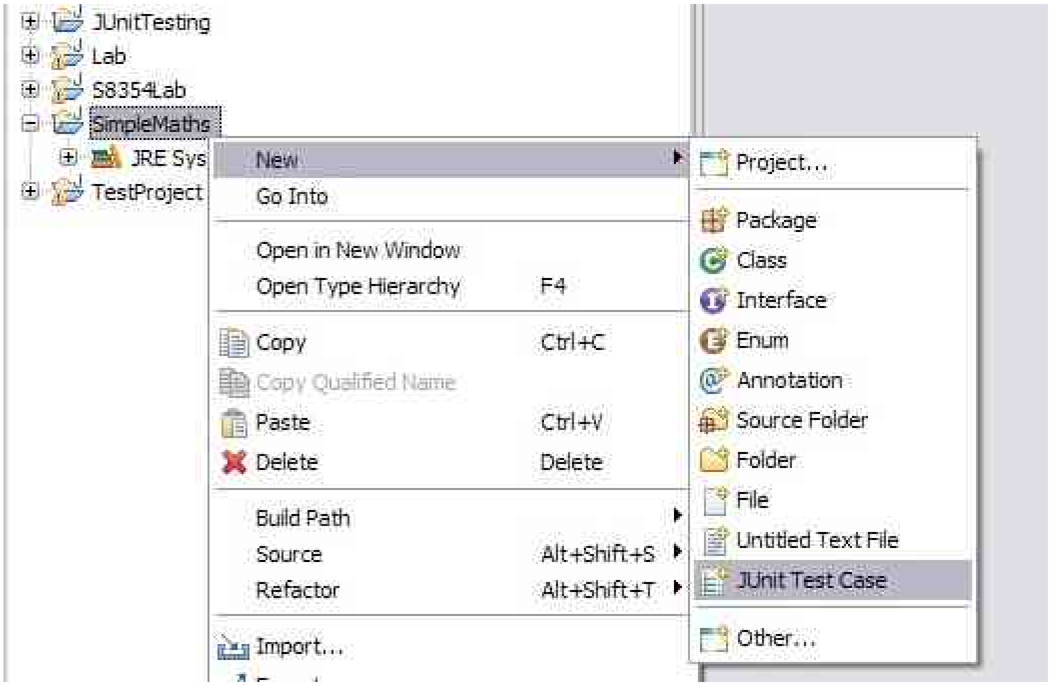
2. Writing the Tests

First we need to create a project and write the tests for the Fibonacci program. After writing some tests we can develop the program and prove that it works.

Create a Project, **File**>**New**>**Project**

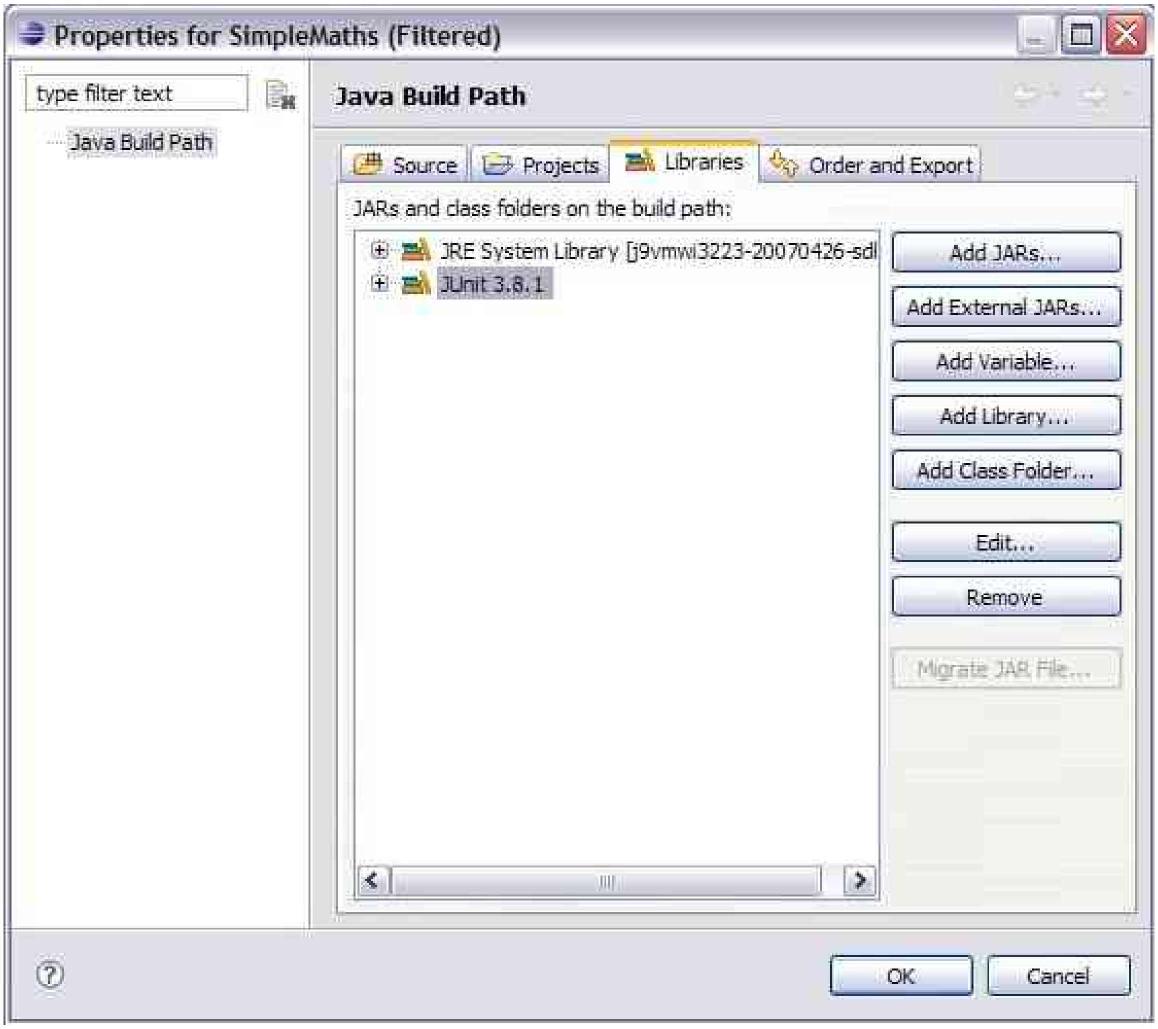
1. Select Java Project
2. Call the project SimpleMaths
3. Select **Finish**.

Right-Click on the SimpleMaths project, select **New>JUnit Test Case**



1. Call the test FibonacciTest.
   Before clicking Finish, note the warning:
   `JUnit 3.8.1 is not on the build path of project 'SimpleMaths'.` `Click here` `to add JUnit 3.8.1 to the build path and open the build path dialog.`

   `Click here` to add the JUnit dependency. The Java Build Path should now look like this:



2. Click OK and Finish.

   The framework for the test has been created for you:

```
import junit.framework.TestCase;

public class FibonacciTest extends TestCase {

      protected void setUp() throws Exception {
            super.setUp();
      }
}
```

   We now need to create a new test for our fibonacci application.

3. Add the following code to the FibonacciTest class, don’t worry about the errors for now.

```
public void testFibonacci() {
            assertEquals(0, Fibonacci.Fibonacci(0));
            assertEquals(1, Fibonacci.Fibonacci(1));
```

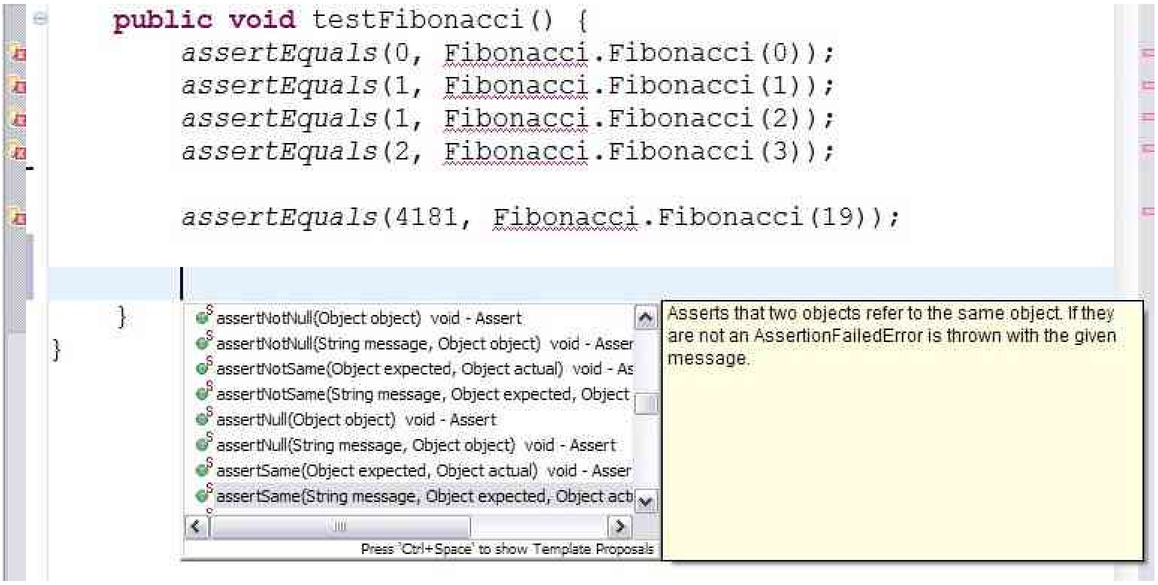
```
                assertEquals(1, Fibonacci.Fibonacci(2));
                assertEquals(2, Fibonacci.Fibonacci(3));

                assertEquals(4181, Fibonacci.Fibonacci(19));
    }
```

4. Awesome, we now have a test case that asserts several (randomly) selected Fibonacci values. Add a few more if you like.

   JUnit has several assertions methods that can be used to test Objects. These include:

   ```
   assertEquals()
   assertNotEquals()
   assertSame()
   assertNotSame()
   assertTrue()
   assertFalse()
   assertNull()
   assertNotNull()
   ```

   To get further details on them, hit the Code-Assist key (ctrl-space) in the testFibonacci method:



```
public void testFibonacci() {
    assertEquals(0, Fibonacci.Fibonacci(0));
    assertEquals(1, Fibonacci.Fibonacci(1));
    assertEquals(1, Fibonacci.Fibonacci(2));
    assertEquals(2, Fibonacci.Fibonacci(3));

    assertEquals(4181, Fibonacci.Fibonacci(19));

    
  }
}
```

```
assertNotNull(Object object)  void - Assert
assertNotNull(String message, Object object)  void - Asser
assertNotSame(Object expected, Object actual)  void - As
assertNotSame(String message, Object expected, Object
assertNull(Object object)  void - Assert
assertNull(String message, Object object)  void - Assert
assertSame(Object expected, Object actual)  void - Asser
assertSame(String message, Object expected, Object act
```

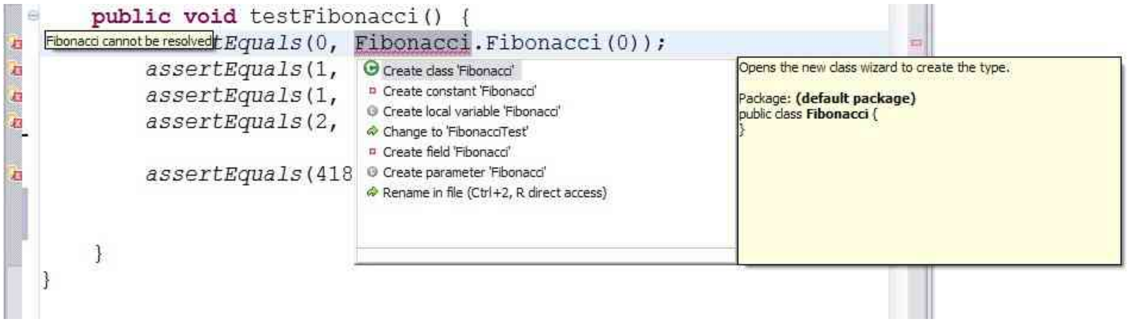
Asserts that two objects refer to the same object. If they are not an AssertionFailedError is thrown with the given message.

   Ok, we now have a test case for Fibonacci numbers. There are errors in the test however. Looking at these, you will see the message `Fibonacci cannot be resolved`.

5. Writing the Fibonacci Application

Using features of Eclipse we can create the necessary files and classes easily using the Quick-Fix feature.

1. Use Quick-Fix to Create class ‘Fibonacci’ in the default package.



```
    public void testFibonacci() {
Fibonacci cannot be resolvedtEquals(0, Fibonacci.Fibonacci(0));
        assertEquals(1,
        assertEquals(1,
        assertEquals(2,

        assertEquals(418


    }
}
```

Opens the new class wizard to create the type.

Package: **(default package)**
public class **Fibonacci** {
}

Create class 'Fibonacci'
Create constant 'Fibonacci'
Create local variable 'Fibonacci'
Change to 'FibonacciTest'
Create field 'Fibonacci'
Create parameter 'Fibonacci'
Rename in file (Ctrl+2, R direct access)

2. There are still errors in the testFibonacci method as the Fibonacci class does not have a Fibonacci method yet. Use Quick-Fix to create one. Ensure it gets defined as `public static int Fibonacci(int i)`. Note, you may need to modify the return type as Eclipse cannot always work this out for you. Fibonacci.java should now look like:

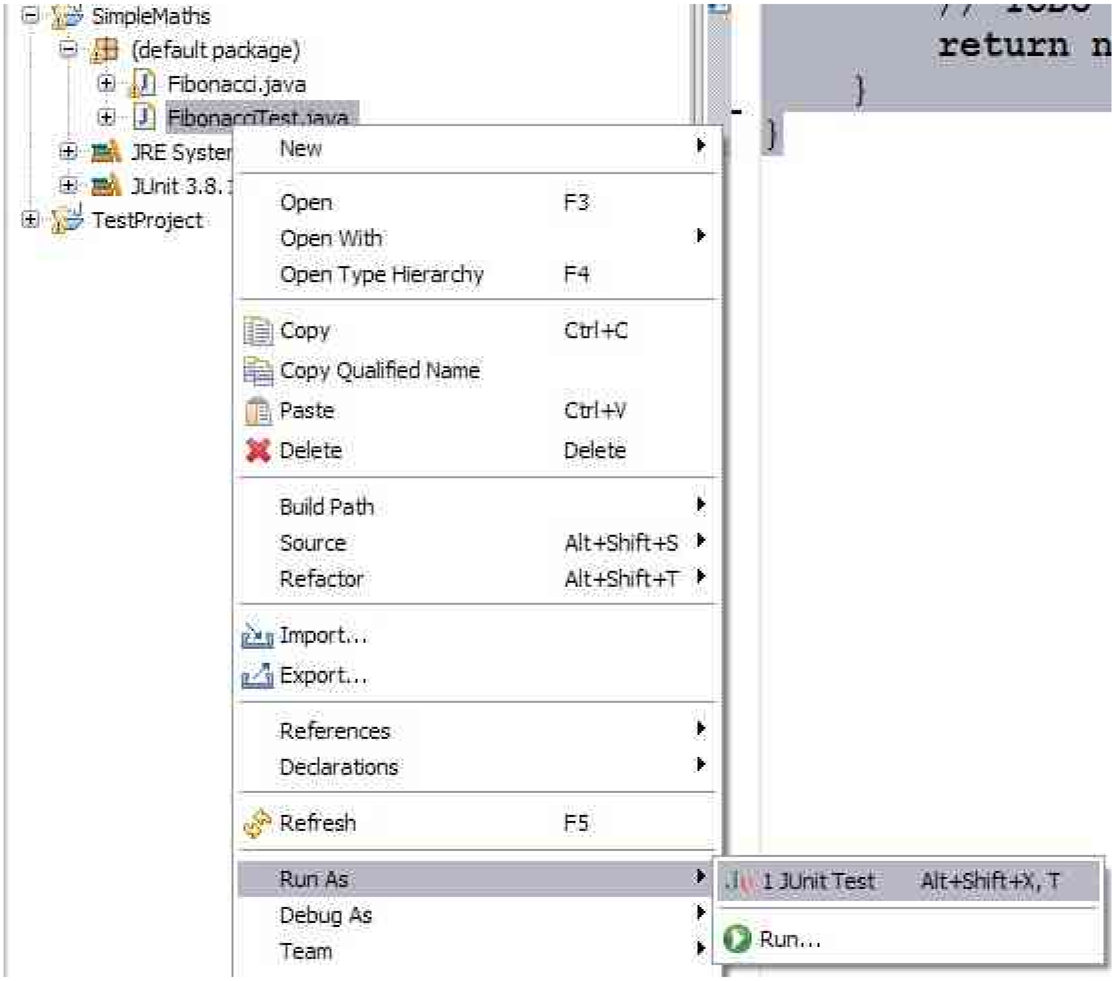
```
public class Fibonacci {
      public static int Fibonacci(int i) {
            // TODO Auto-generated method stub
            return null;
      }
}
```

3. All errors in the FibonacciTest class should now be resolved. The next step is to run the test and demonstrate that it fails at this stage.

4. Running the Test

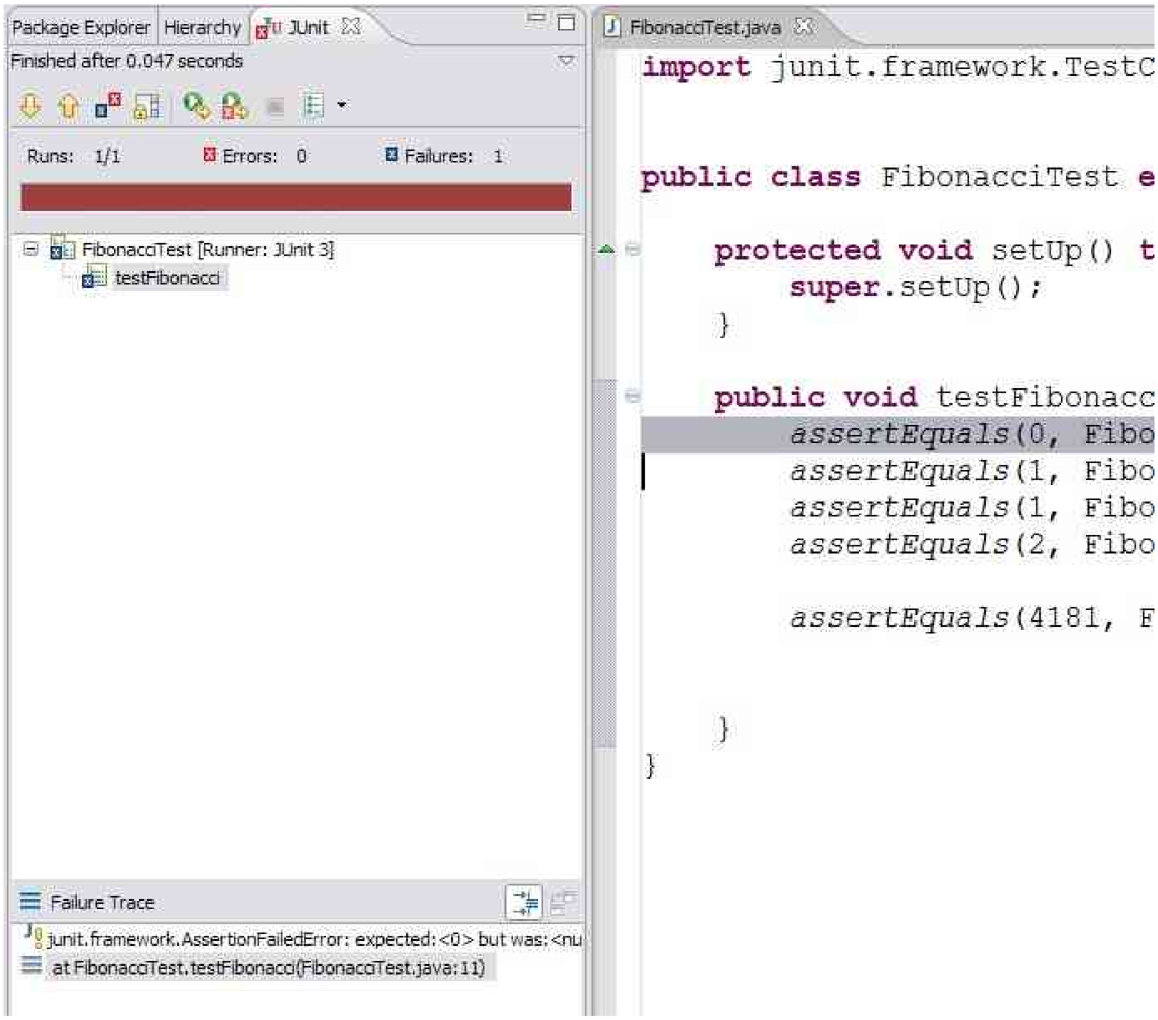
To run the test do the following:

1. Right-Click FibonacciTest.java in the Package Explorer and select **Run As>JUnit Test**.



2. A JUnit tab should appear in the left set of panels and show Runs: 1/1 Errors: 0 Failures: 1.

   The Failure Trace panel shows where the failure occurred. Try double-clicking on the failure to jump to the test that failed.



Finished after 0.047 seconds

Runs: 1/1 Errors: 0 Failures: 1

FibonacciTest [Runner: JUnit 3]
testFibonacci

```
junit.framework.AssertionFailedError: expected:<0> but was:<nu
at FibonacciTest.testFibonacci(FibonacciTest.java:11)
```

```
import junit.framework.TestC


public class FibonacciTest e

    protected void setUp() t
        super.setUp();
    }

    public void testFibonacc
        assertEquals(0, Fibo
        assertEquals(1, Fibo
        assertEquals(1, Fibo
        assertEquals(2, Fibo

        assertEquals(4181, F


    }
}
```

The failure occurred as we haven't written any code yet. Let us add the Fibonacci code to the Fibonacci method (in the Fibonacci class). Add the following:

```
public static int Fibonacci(int n) {
      if (n == 0 || n == 1)
            return 0;
      else
            return Fibonacci(n-1) + Fibonacci(n-2);
}
```

Try running the test again. What happens now?

Can you fix the code so the tests work? (hint: what should the base case return value be).

When the test passes successfully you should see a green bar in the JUnit view and Errors: 0, Failures: 0.

5. Further Work

The test we have written should work for all positive integers. What happens if we pass in a negative integer?

Write a new test for the Fibonacci method which works with negative numbers and modify the Fibonacci method so it passes.