[**在Eclipse中使用JUnit4进行单元测试**](http://blog.csdn.net/andycpp/article/details/1327147)

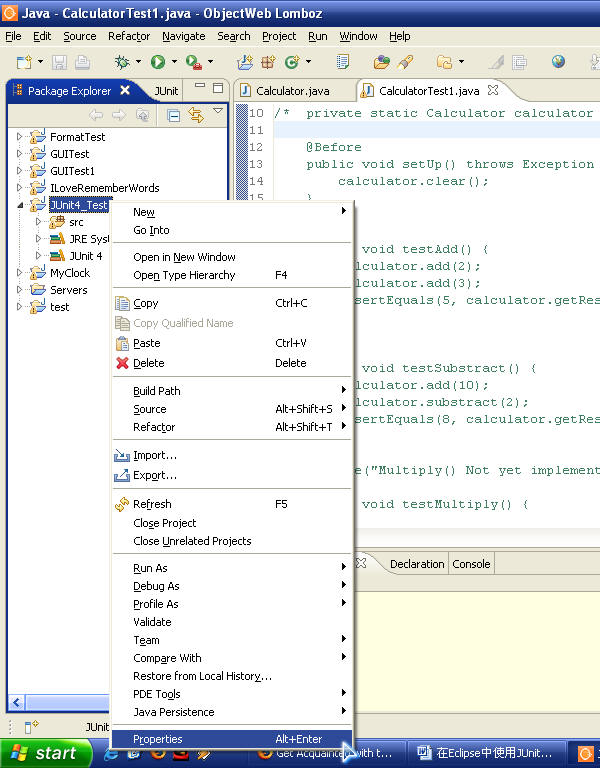
我们在编写大型程序的时候，需要写成千上万个方法或函数，这些函数的功能可能很强大，但我们在程序中只用到该函数的一小部分功能，并且经过调试可以确定，这一小部分功能是正确的。但是，我们同时应该确保每一个函数都完全正确，因为如果我们今后如果对程序进行扩展，用到了某个函数的其他功能，而这个功能有bug的话，那绝对是一件非常郁闷的事情。所以说，每编写完一个函数之后，都应该对这个函数的方方面面进行测试，这样的测试我们称之为单元测试。传统的编程方式，进行单元测试是一件很麻烦的事情，你要重新写另外一个程序，在该程序中调用你需要测试的方法，并且仔细观察运行结果，看看是否有错。正因为如此麻烦，所以程序员们编写单元测试的热情不是很高。于是有一个牛人推出了单元测试包，大大简化了进行单元测试所要做的工作，这就是JUnit4。本文简要介绍一下在Eclipse3.2中使用JUnit4进行单元测试的方法。

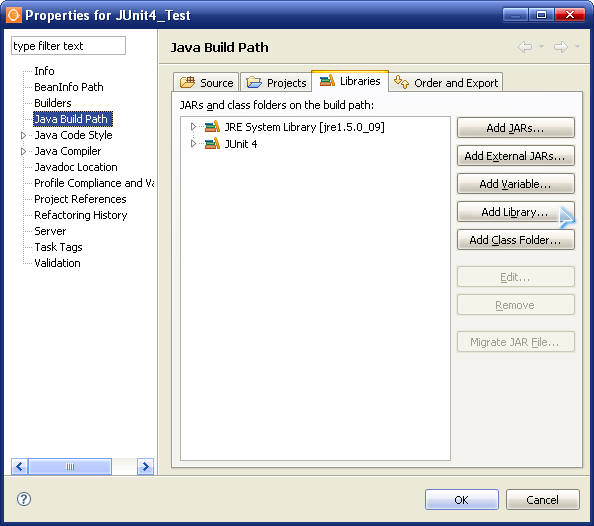
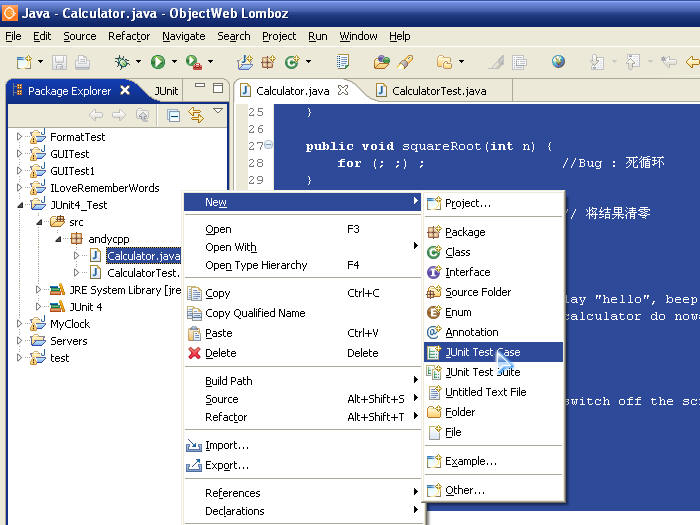
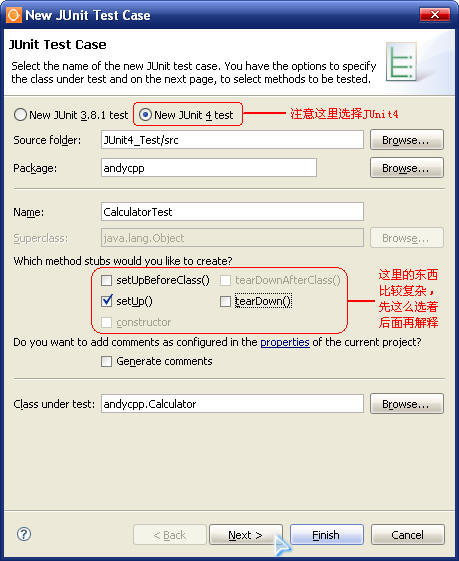
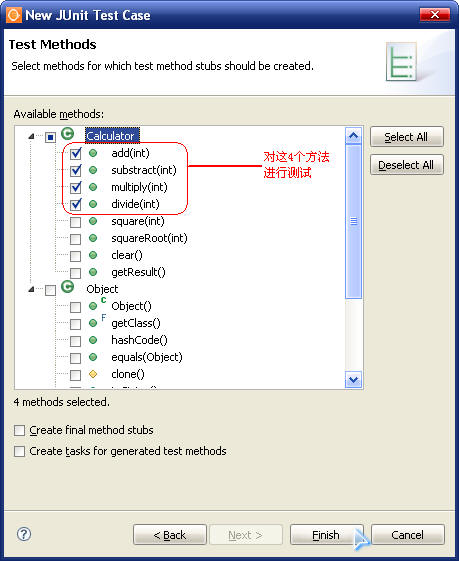
首先，我们来一个傻瓜式速成教程，不要问为什么，Follow Me，先来体验一下单元测试的快感！

首先新建一个项目叫JUnit\_Test，我们编写一个Calculator类，这是一个能够简单实现加减乘除、平方、开方的计算器类，然后对这些功能进行单元测试。这个类并不是很完美，我们故意保留了一些Bug用于演示，这些Bug在注释中都有说明。该类代码如下：

http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gifpackage andycpp;  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gif  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedBlock.gifpublic class Calculator ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif    private static int result; // 静态变量，用于存储运行结果  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public void add(int n) ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        result = result + n;  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public void substract(int n) ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        result = result - 1;  //Bug: 正确的应该是 result =result-n  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public void multiply(int n) ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }         // 此方法尚未写好  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public void divide(int n) ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        result = result / n;  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public void square(int n) ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        result = n \* n;  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public void squareRoot(int n) ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        for (; ;) ;            //Bug : 死循环  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public void clear() ...{     // 将结果清零  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        result = 0;  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public int getResult() ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        return result;  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedBlockEnd.gif}  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gif

第二步，将JUnit4单元测试包引入这个项目：在该项目上点右键，点“属性”，如图：

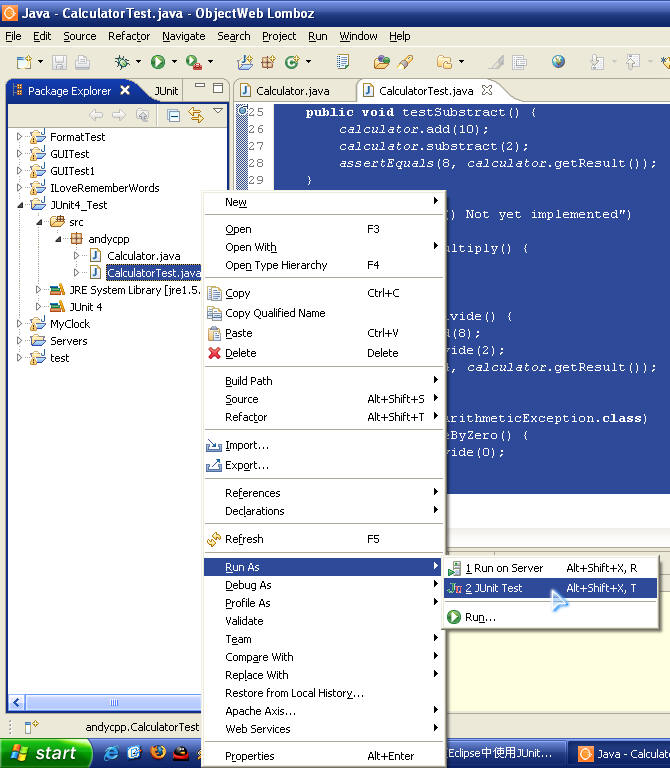


在弹出的属性窗口中，首先在左边选择“Java Build Path”，然后到右上选择“Libraries”标签，之后在最右边点击“Add Library…”按钮，如下图所示：  
  
  
然后在新弹出的对话框中选择JUnit4并点击确定，如上图所示，JUnit4软件包就被包含进我们这个项目了。  
  
    第三步，生成JUnit测试框架：在Eclipse的Package Explorer中用右键点击该类弹出菜单，选择“New  JUnit Test Case”。如下图所示：  
  
  
在弹出的对话框中，进行相应的选择，如下图所示：  
  
  
    点击“下一步”后，系统会自动列出你这个类中包含的方法，选择你要进行测试的方法。此例中，我们仅对“加、减、乘、除”四个方法进行测试。如下图所示：  


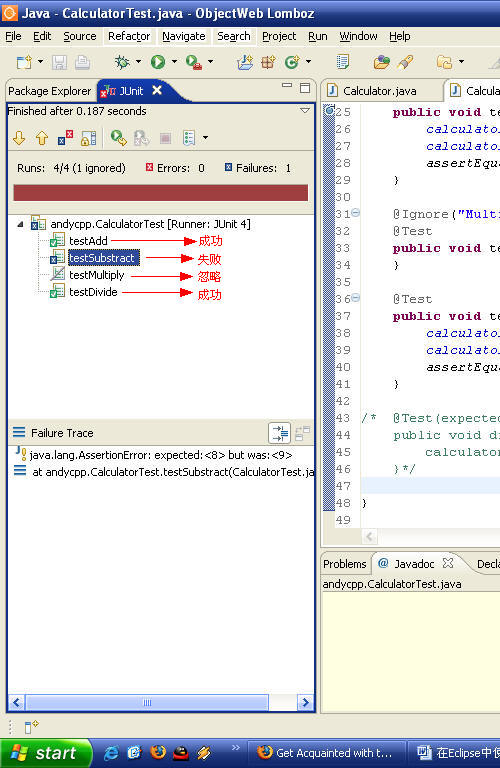
之后系统会自动生成一个新类CalculatorTest，里面包含一些空的测试用例。你只需要将这些测试用例稍作修改即可使用。完整的CalculatorTest代码如下：

http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gifpackage andycpp;  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gif  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gifimport static org.junit.Assert.\*;  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gifimport org.junit.Before;  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gifimport org.junit.Ignore;  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gifimport org.junit.Test;  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/None.gif  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedBlock.gifpublic class CalculatorTest ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif    private static Calculator calculator = new Calculator();  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif      
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif    @Before  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public void setUp() throws Exception ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        calculator.clear();  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif    @Test  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public void testAdd() ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        calculator.add(2);  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        calculator.add(3);  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        assertEquals(5, calculator.getResult());  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif    @Test  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public void testSubstract() ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        calculator.add(10);  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        calculator.substract(2);  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        assertEquals(8, calculator.getResult());  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif    @Ignore("Multiply() Not yet implemented")  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif    @Test  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public void testMultiply() ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif    @Test  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockStart.gifhttp://images.csdn.net/syntaxhighlighting/OutliningIndicators/ContractedSubBlock.gif    public void testDivide() ...{  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        calculator.add(8);  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        calculator.divide(2);  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/InBlock.gif        assertEquals(4, calculator.getResult());  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://images.csdn.net/syntaxhighlighting/OutliningIndicators/ExpandedBlockEnd.gif}

第四步，运行测试代码：按照上述代码修改完毕后，我们在CalculatorTest类上点右键，选择“Run As  JUnit Test”来运行我们的测试，如下图所示：



运行结果如下：



进度条是红颜色表示发现错误，具体的测试结果在进度条上面有表示“共进行了4个测试，其中1个测试被忽略，一个测试失败”

            至此，我们已经完整体验了在Eclipse中使用JUnit的方法。在接下来的文章中，我会详细解释测试代码中的每一个细节！