



## 大话企业级 Android 开发 · 第六部分

### 本教程说明及版权声明

- 《大话企业级 Android 开发》是国土工作室为了方便中国 Android 开发者，推动 Android 企业级应用开发，特投入大量心血撰写的书籍，并在网络上免费发布，希望为移动互联网和智能手机时代贡献绵薄之力！所有相关文档版权均属国土工作室所有。
- 本教程是由国土工作室参考官方文档，综合市面相关书籍，经过充分的吸收消化，结合开发实践的一部原创作品，为了本教程及早与广大读者同仁见面、分享，特采用定稿一部分就发布一部分的连载方式发布。读者可以在本博客获取最新内容。
- 未经国土工作室授权，禁止将此文档及其衍生作品以标准（纸质）书籍形式发行。
- 本文档受有关法律的版权保护，对本文档内容的任何未经同意的复制和抄袭行为，将导致相应的法律责任。未经国土工作室同意，任何团体及个人不能用此教程牟利，违者必究。但是：在不收取其他人费用的前提下，您可以自由传播此文档，但必须保证版权信息、文档及其自带标示的完整性。
- 如果对该文档有任何疑问或者建议，请进入官方微博客  
<http://www.cnblogs.com/guoshiandroid/>留言或者直接与国土工作室联系（后附联系方式），我们会慎重参考您的建议并根据需要对本文档进行修改，以造福更多开发者！
- 《大话企业级 Android 开发》的最新及完整内容会在国土工作室官方微博客定期更新，请访问国土工作室博客  
<http://www.cnblogs.com/guoshiandroid/>获取更多更新内容。





## 关于国土工作室

我们(国土工作室)是一支专注于 Android 平台企业级应用开发的技术团队,对娱乐多媒体应用有着深刻的理解及研发能力,致力服务于企业用户。为音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络应用向移动互联网发展提供解决方案和技术支持,为企业提供 Android 培训服务等多种业务。

我们尤其擅长于提供从 Android 客户端到服务端的一站式解决方案和技术支持,服务端可以采用 Java EE,也可以采用轻量级流行的 LAMP 技术体系。目前,研发出了比 KU6、优酷更加强大和完善的 Android 视频网站娱乐多媒体客户端软件,并在持续升级中。

目前,我们正在务实而卓有成效的与音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络服务商合作,发展迅速,渴望有志之士的加入,和我们一起为成为世界最好的 Android 软件开发和咨询、培训公司而奋斗,为移动互联网和智能手机时代贡献力量!

## 联系我们

电话:15711060468

Email:[guoshiandroid@gmail.com](mailto:guoshiandroid@gmail.com)

博客: <http://www.cnblogs.com/guoshiandroid/>





小安: 记得以前做 Java 开发的时候利用 Junit 进行测试, 利用 `System.out.println()` 方法可以在控制台打印信息, 在 Android 平台下如何进行单元测试和信息输出呢? 我对单元测试的用途还不太了解, 博士您能给我讲讲吗?

大致: 那接下来我带你学习如何对 Android 应用进行单元测试及日志输出吧, 使用单元测试可以保证我们开发应用的质量, 一般我们开发完业务层代码之后应该对业务层进行单元测试, 确保业务层不出现 Bug。对业务层测试通过之后, 控制层就可以调用业务层完成需要的功能。另外, 单元测试会加快我们开发的速度。如果不进行单元测试, 我们需要将应用发布到模拟器上之后, 再对应用中的各项功能进行测试, 这样效率较低。(本书后面的许多例子都会进行单元测试, 希望能够掌握单元测试的方法)

## 1 单元测试与日志输出

### 1.1 单元测试步骤

进行单元测试需要以下几步:

第一步: 首先在 `AndroidManifest.xml` 中加入下面粗体代码:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.sharp.action" android:versionCode="1" android:versionName="1.0">
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <uses-library android:name="android.test.runner" />
        ....
    </application>
    <uses-sdk android:minSdkVersion="6" />
    <instrumentation android:name="android.test.InstrumentationTestRunner"
        android:targetPackage="cn.sharp.action" android:label="Tests for My App" />
</manifest>
```

说明:

- `<uses-library android:name="android.test.runner" />`  
该行代码必须位于 `<application>` 元素之内, 与 `<activity>` 元素同级。
- 上面 `targetPackage` 指定的包要和应用的 `package` 相同。如果不相同, 会出现找不到单元测试用例的错误。

第二步: 编写单元测试代码

测试类必须继承自 `AndroidTestCase` 类

第三步: 执行测试。

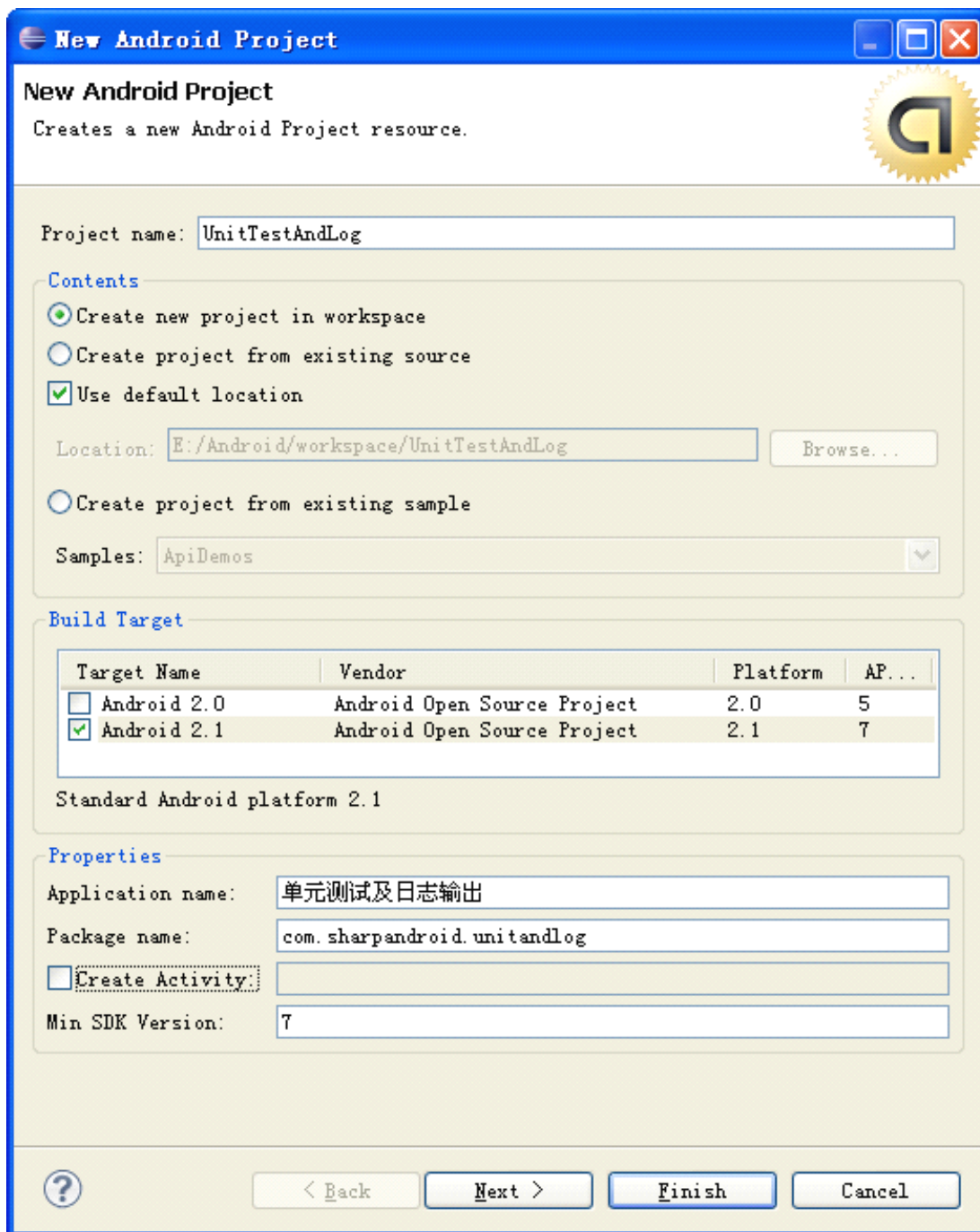




## 1.2 单元测试示例

### 1. 创建项目

创建名为“UnitTestAndLog”的项目，如下图。



需要注意的是，由于我们在本项目中并不需要使用 Activity，因此我们可以不创建 Activity，只需要将【Create Activity】前面的单选框去除选定就可以不用输入 Activity 名称。如果去除选定，在项目包下就不会创建 Activity。当然你也可以输入





一个名称, 虽然我们不会用到。

## 2. 在功能清单文件中配置单元测试的环境

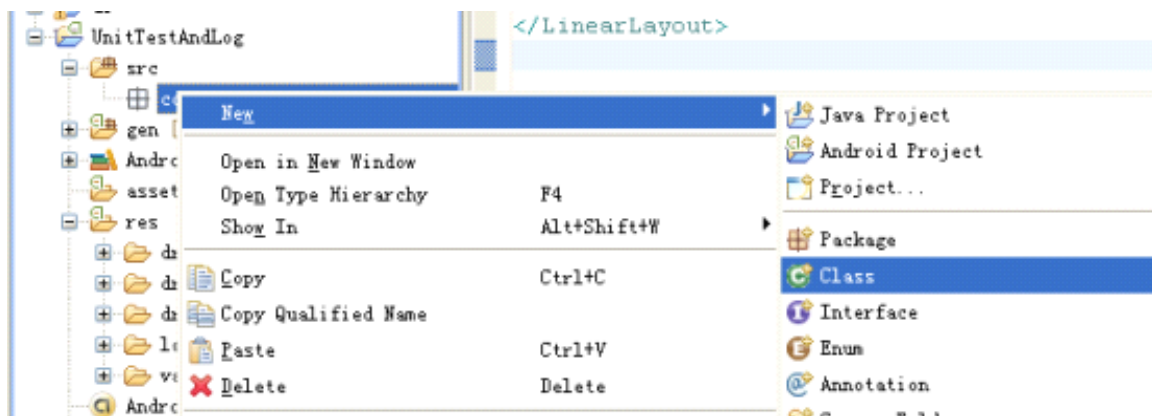
打开功能清单文件, 添加上面所述的代码, 最终代码如下。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sharpandroid.junit"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <uses-library android:name="android.test.runner" />
    </application>
    <uses-sdk android:minSdkVersion="7" />
    <instrumentation android:name="android.test.InstrumentationTestRunner"
        android:targetPackage="com.sharpandroid.junit"
        android:label="Tests for My App" />
</manifest>
```

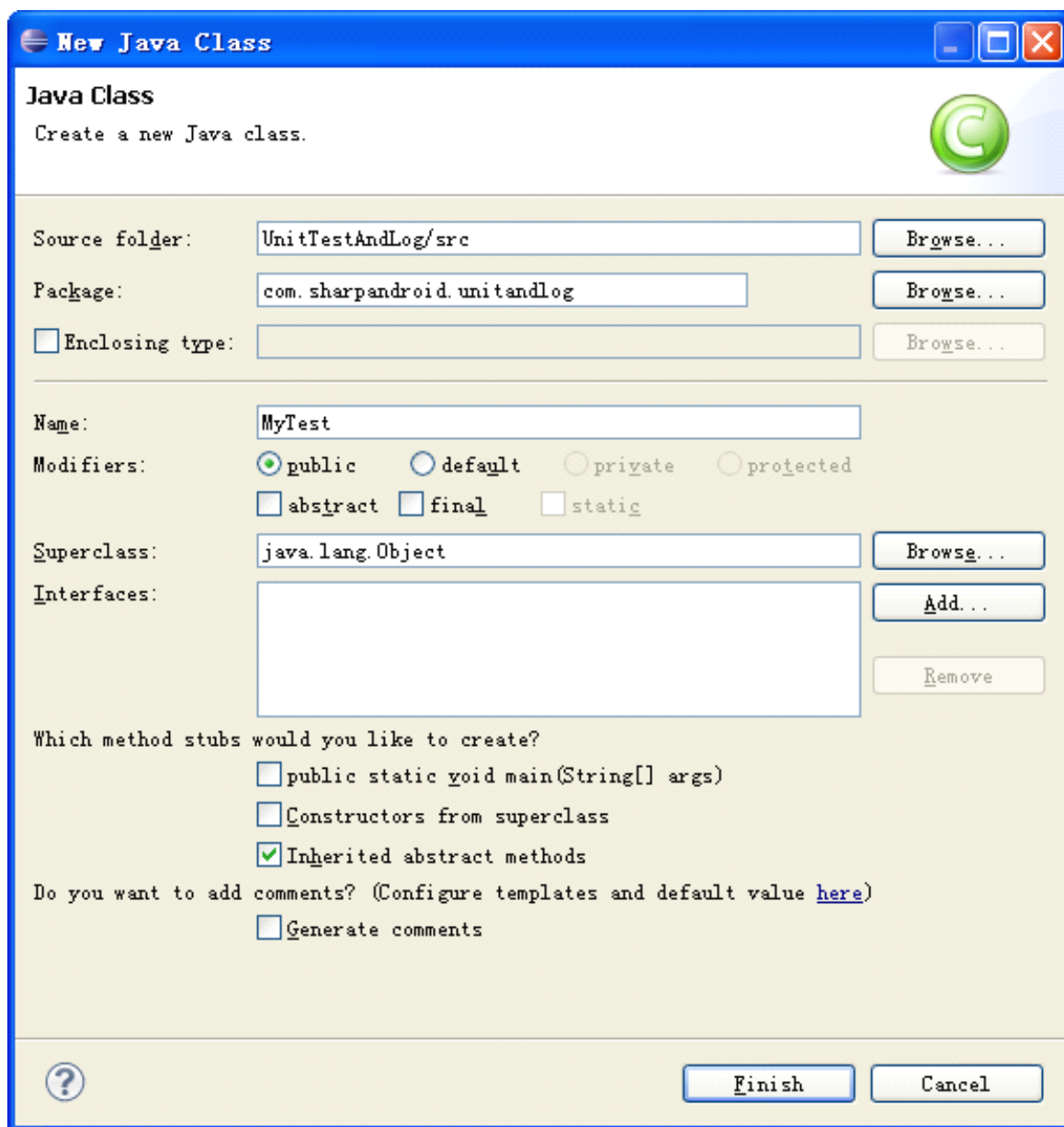
## 3. 新建测试用例

首先, 在项目包位置, 右键单击之后点击【New】→【Class】, 如下图所示。

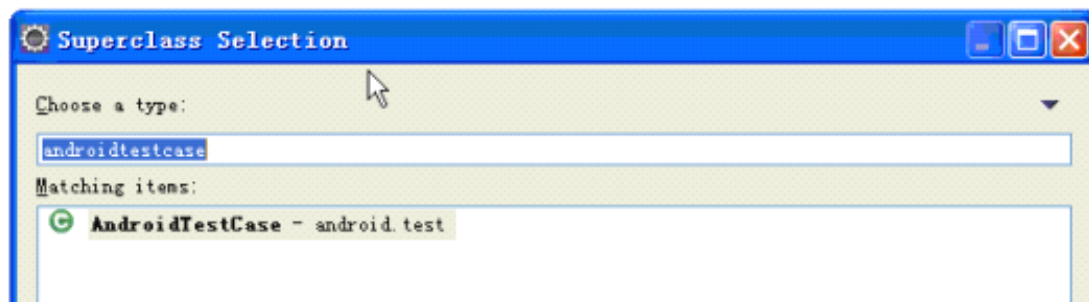


点击【Class】之后到如下界面。填入【Name】信息为 MyTest。





下面的 Superclass 填写的方式如下，点击其后面的【Browse】，界面如下。



在【Choose a type】下面的文本框中输入“androidtestcase”，点击【OK】。之后再点击【Finish】







完成测试用例的创建工作。在 AndroidTestCase 测试中, 我们采用的 Junit3, 因此测试方法的书写格式是:

```
public void testMethodName() { }
```

初学者常犯的错误是为测试方法添加参数。一定要注意测试方法是不需要传入参数的。如果测试方法内部遇到异常, 建议直接抛出, 而不要捕获异常。异常抛出后会被测试框架获取, 之后在控制台显示出来, 方便我们了解异常信息。

测试类内容如下:

MyTest.java

```
package com.sharpandroid.unitandlog;
```

```
import junit.framework.Assert;
```

```
import android.test.AndroidTestCase;
```

```
import android.util.Log;
```

```
public class MyTest extends AndroidTestCase {
```

```
    private static final String TAG = "MyTest";
```

```
    public void testSave() throws Throwable{
```

```
        int i = 4+8;
```

```
        Assert.assertEquals(5, i);
```

```
    }
```

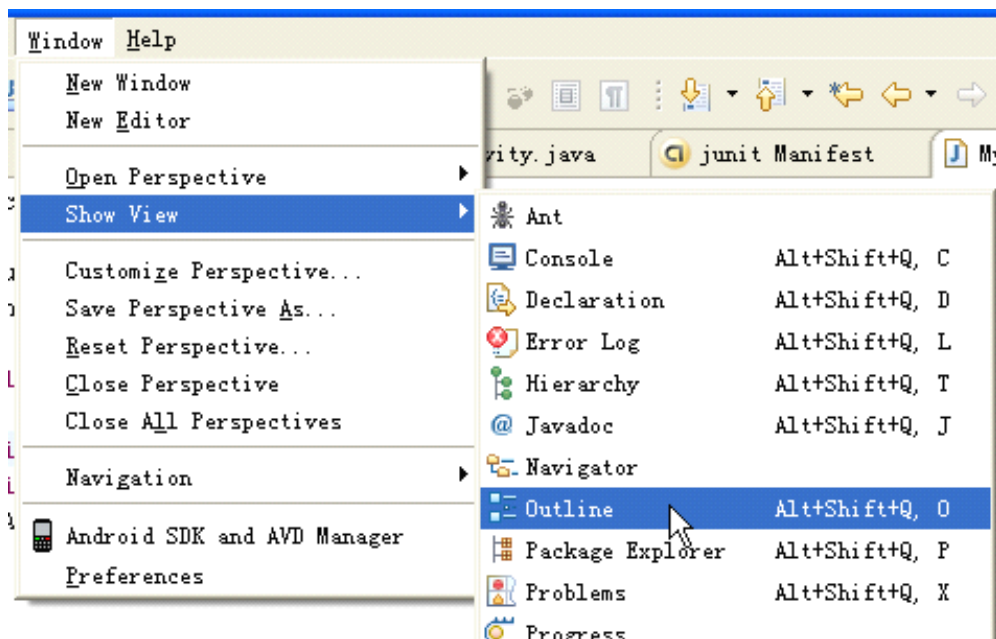
```
}
```

**Assert** 类在此的作用是判断所得到的结果与期望值的关系。这里判断 i 是否等于 5; 如果相等则通过, 不相等, 则抛出异常。

#### 4. 运行测试用例

首先, 进入“大纲视图”(Outline), 如果 Eclipse 中没有该视图, 可用如下图之方式打开。





打开之后，其位置可能不太合意，可以用鼠标拖动它的标题栏到指定的位置。其他的视图也可采用拖动的方式拖动到其他位置上去。

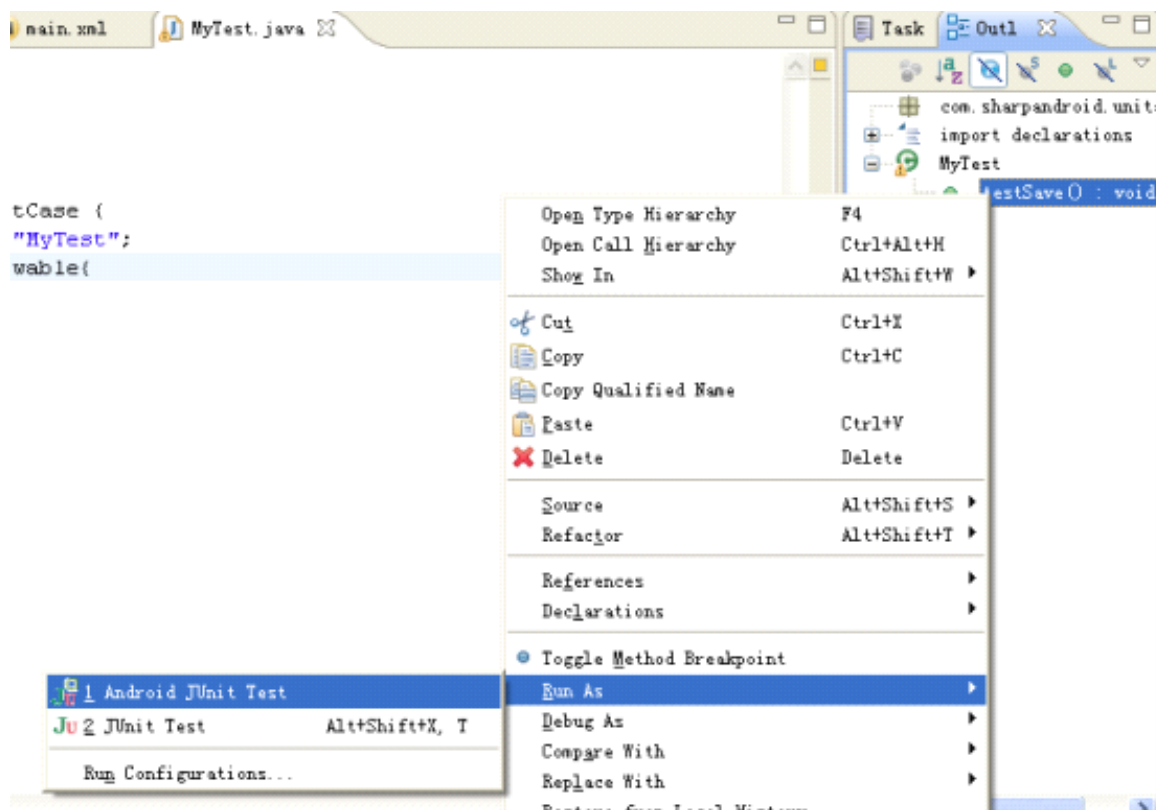




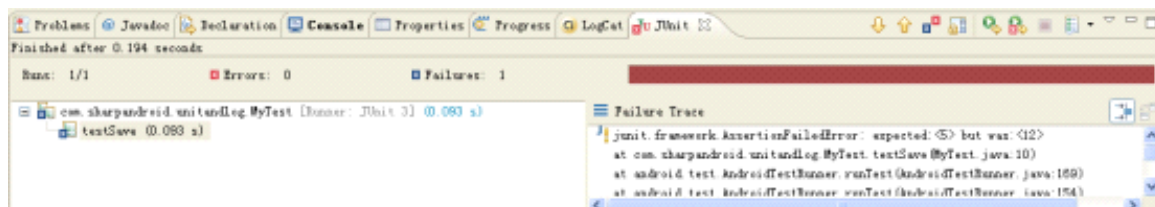


现在可以运行测试用例了。右键单击，之后【Run As】→【Android JUnit Test】，操作如下图所示。



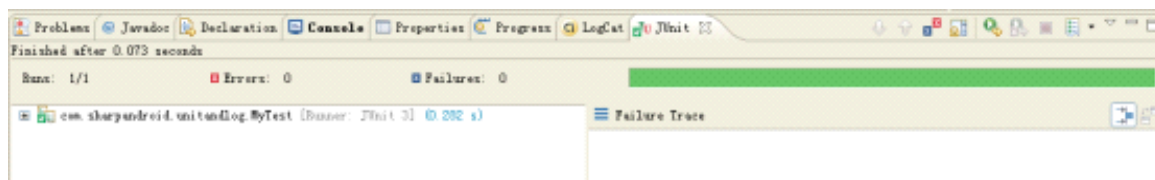


运行完成，我们观察 Junit 的控制台。结果如下图。



如上图，结果条显示为红色，说明测试没有通过。

下面将测试用例中的代码修改为 “Assert.assertEquals(12, i);” 之后再次运行。



如图，结果条显示为绿色，说明测试通过。

## 1.3 日志输出


前面已经提到 Console 控制台只能输出应用安装的信息。比如在程序中添加一行 `System.out.println(“sharpandroid”);`;


如果在 Eclipse 中进行 Java 程序开发，该语句会在 Console 中打印。但我们开发 Android

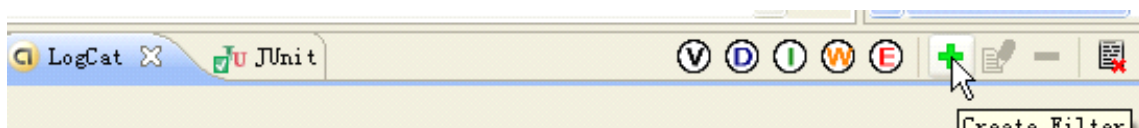




应用时, 该语句在 Console 控制台中并不会打印出字符串。该字符串会在 LogCat 中打印出来。

在 Android 程序中进行信息输出, 一般采用 `android.util.Log` 类的静态方法就可以实现。Log 类所输出的日志的内容从少到多分别是 ERROR、WARN、INFO、DEBUG、VERBOSE, 对应五种不同类型的首字母, 分别对应有 `Log.e()`、`Log.w()`、`Log.i()`、`Log.d()`、`Log.v()` 五种静态方法, 使用不同的方法输出的信息的颜色各不相同, 并且如下图, 在 Logcat 控制台右上侧有相应的按钮 , 点击每个按钮, 可以过滤出其自身类型及其右侧类型的

日志信息。如点击  按钮, 会显示 I、W、E 三种类型的信息, 而 V、D 类型的信息则不会显示。



### 1. 编写日志输出测试类

修改后的代码如下:

```
package com.sharpandroid.unitandlog;

import junit.framework.Assert;
import android.test.AndroidTestCase;
import android.util.Log;

public class MyTest extends AndroidTestCase {
    private static final String TAG = "MyTest";
    public void testSave() throws Throwable{
        int i = 4+8;
        Log.i(TAG, "i = " + i);
        Assert.assertEquals(12,i);
    }
}
```

说明:

- `Log.i(String tag, String msg);`  
tag : 为这条信息定义一个标签, 在开发时通常采用其所在类的类名, 这样方便我们追踪输出的信息。也方便我们在看到信息时知道其由哪个类输出。  
msg : 该参数为希望输出的信息的内容。

一般将第一个参数定义成一个静态常量, 按照 JAVA 语言的规范, 静态常量的命名一般全部采用大写。书写大写字母时不太习惯的, 可以先书写为小写字母, 再将其选中, 在 Eclipse 中使用 “Ctrl+Shift+X” 快捷键将字符串变成大写。使用 “Ctrl+Shift+Y” 恰好相反, 将英文字符串变成小写。

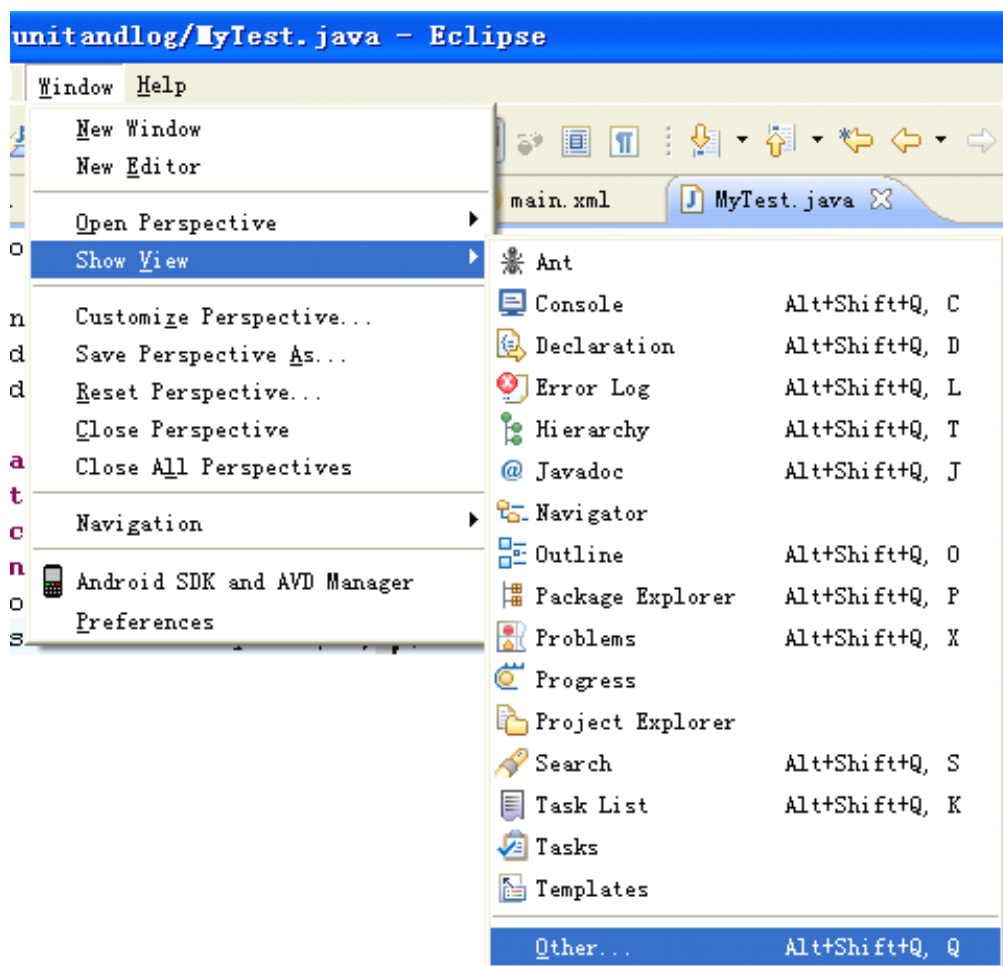


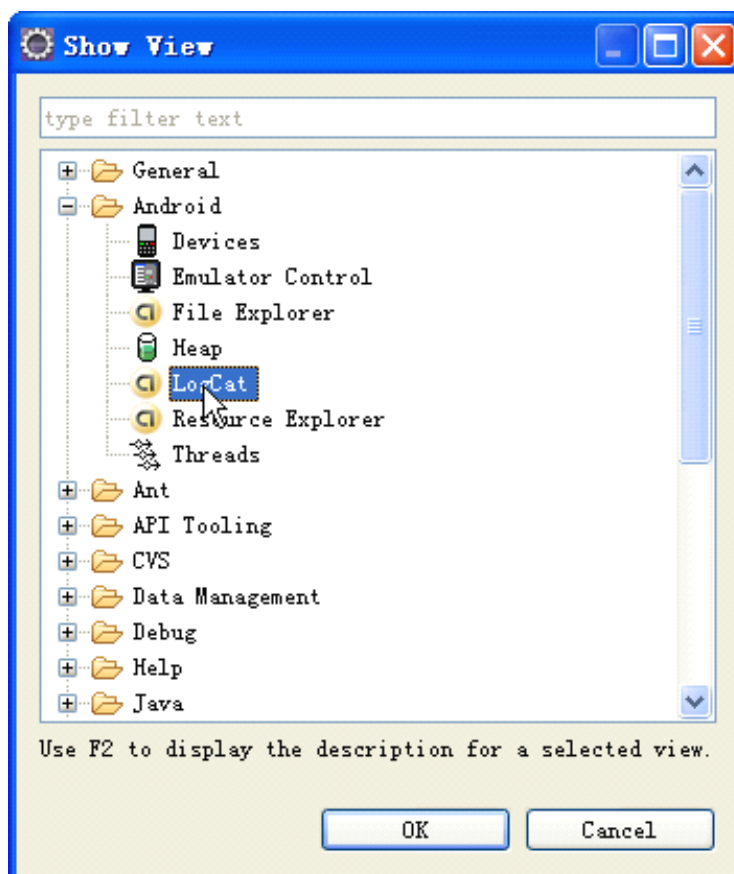


## 第二步 执行测试

执行的方法如同单元测试一节的执行测试方法。

打开日志信息查看器“LogCat”，查看是否输出该信息。打开步骤如下图：



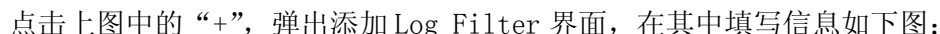
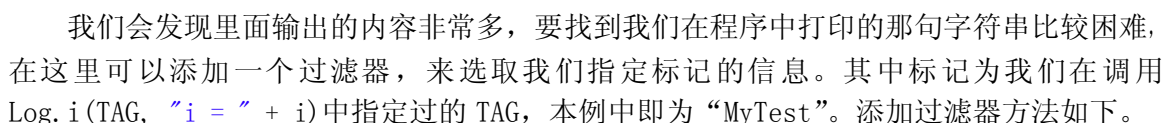


选择【LogCat】后，点击【OK】，此时观察 Eclipse 下方，出现 LogCat 控制台，如下图。





本教程官方讨论群: 65882321



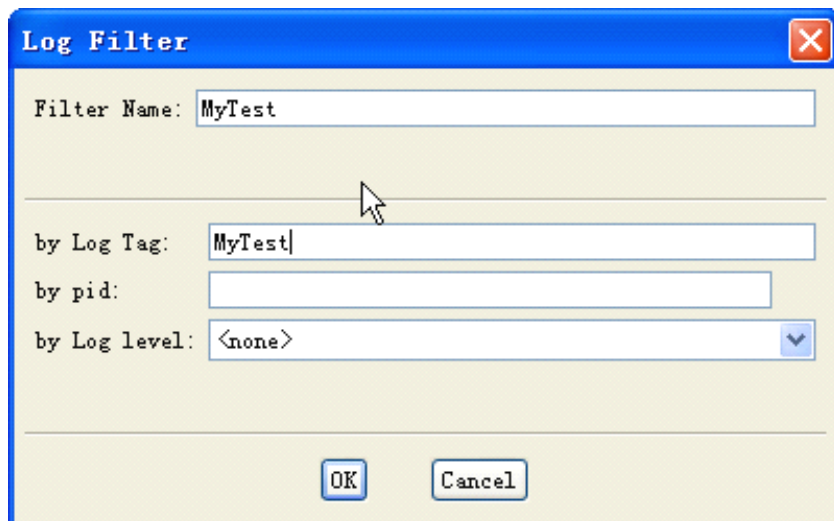
**电话:**15711060468

**Email:** [guoshiandroid@gmail.com](mailto:guoshiandroid@gmail.com)

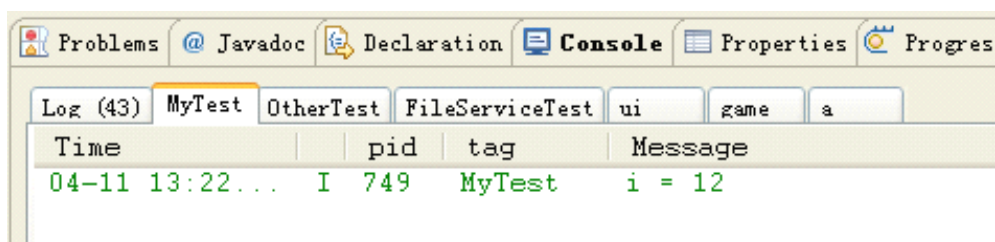
博客: <http://www.cnblogs.com/guoshiandroid/>

版权所有，请保留

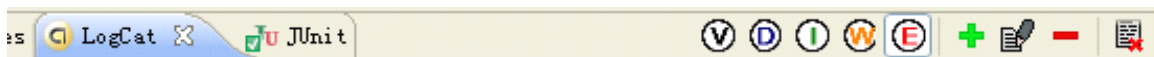




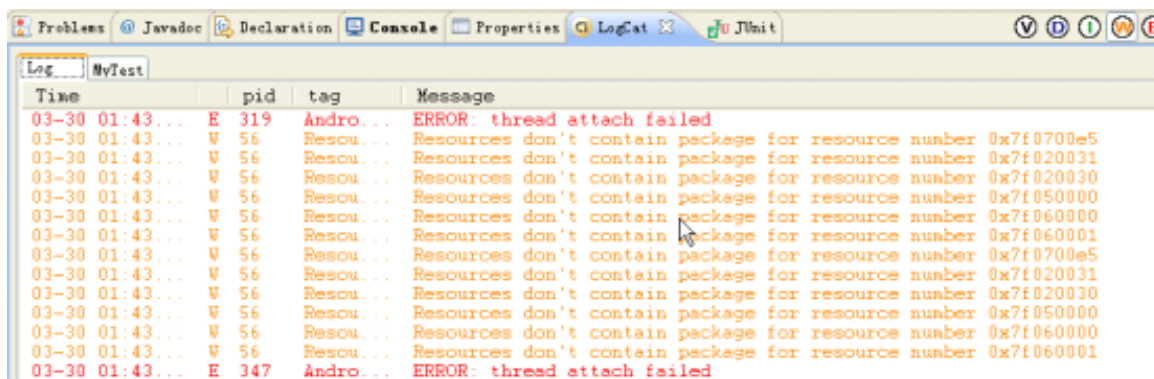
点击【OK】。在 LogCat 管理器中多出一个选项卡“MyTest”，在该选项卡中出现我们在代码中想要打印的信息。



下面介绍下 LogCat 管理器中几个按钮的功能。



点击上图中的“V”、“D”、“I”、“W”、“E”，可以过滤出当前选项卡中的对应级别的信息。如在 Log 选项卡下点击 W 键后结果如下图。







另外，LogCat 控制台右侧有另外三个图标，其对应含义如下：



编辑过滤器



删除过滤器



清除日志信息

## 2 揪出程序中的臭虫——程序调试示例

大致：小安，前面咱们已经做三个项目了，我现在给你出一道题目，做一个项目，项目的目标很简单，一个加法器，两个 EditText 用于输入两个数字，点击一个 Button 之后在另一个 EditText 中显示结果。怎么样？有没有信心搞定？

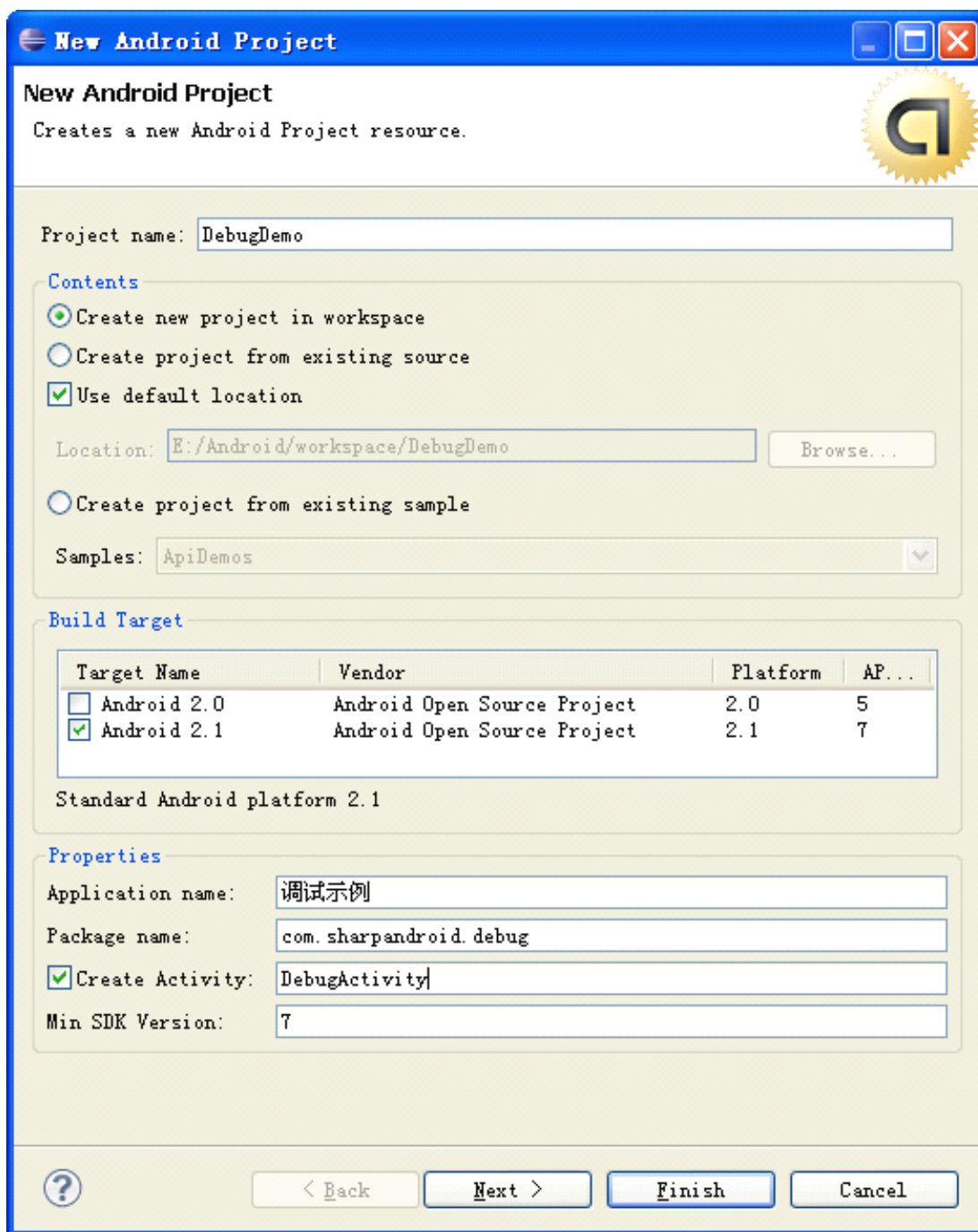
小安：小菜一碟，您等着吧，我立马搞定。嘿嘿！

下面是小安实现的全程实录。

### 1. 创建工程

创建一个名为“DebugDemo”的工程，如下图：





## 2. 编写 main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
```





```
<EditText
    android:layout_width="60px"
    android:layout_height="wrap_content"
    android:id="@+id/num1"
/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/add"
/>

<EditText
    android:layout_width="60px"
    android:layout_height="wrap_content"
    android:id="@+id/num2"
/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/equ"
/>

<EditText
    android:layout_width="80px"
    android:layout_height="wrap_content"
    android:id="@+id/result"
/>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/cal"
    android:id="@+id/calbutton"
/>

</LinearLayout>
```

### 3. 编写DebugActivity.java:

```
package com.sharpandroid.debug;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```





```
import android.widget.TextView;

public class DebugActivity extends Activity {
    private EditText num1Edit ;
    private EditText num2Edit ;
    private TextView resultText;
    private Button calButton ;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        num1Edit = (EditText) findViewById(R.id.num1);
        num2Edit = (EditText) findViewById(R.id.num2);

        calButton = (Button) findViewById(R.id.calbutton);
        calButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String num1Str = num1Edit.getText().toString();
                String num2Str = num2Edit.getText().toString();
                String result = getResult(num1Str, num2Str);
                resultText.setText(result);
            }
        });
    }

    public String getResult(String num1Str,String num2Str){
        float num1 = Float.parseFloat(num1Str);
        float num2 = Float.parseFloat(num2Str);
        float result = num1 + num2;
        return ""+result;
    }
}
```

做完这些，小安信心十足，非常高兴，认为大致博士怎么搞这么简单的题目，自己这么快就搞定了，心中不免暗自得意。于是执行程序，效果如下图：





这个情况让小安大跌眼镜了，怎么出现这么一个弹出框呢？以为是系统的Bug呢，点击【强行关闭】，之后又重复执行了几遍，谁知道Android一点面子都不给，还是坚持给了这么个提示。把小安给急的满头大汗，心想这么简单的程序怎么就出错了呢？这可如何是好？只好向大致博士求救。

小安：博士，这是怎么了？弹出这么个玩意儿？什么意思啊？

大致：我看看啊，哎呀，小安呐，你这肯定是程序中什么地方写错啊。别急别急，由于经验不足或者一些疏忽，在开发的道路上总会遇到这样或那样的问题，即便是一些具有多年开发经验的老手也会犯一些错误。初学者往往会被程序中出现的错误弄得焦头烂额、手足无措，你也经常在论坛或者QQ群里见有初学者发一些“十万火急、跪求大侠指点迷津”等求救的帖子吧？其实在本博士的法眼里多数错误是非常简单和低级的，是很容易发现的，只是这些发帖的朋友缺少一种能力，一种成为真正程序员所必备的可谓安身立命的在这个行业闯荡的基本本领，这就是——调错能力。犯了错误并不可怕，错误是难免的，问题是要及时发现错误修正错误，伟大的领袖曾经说过知错能改还是好同志。因而如何发现错误、如何改正错误就非常重要，接下来我传授你一套绝招，教你如何利用Eclipse工具跟踪整个程序的运行状况以及对错误的调试及纠正，让你如同福尔摩斯般的侦查出程序中的蛛丝马迹，找到程序的bug，成为一个高手！

下面是小安在大致指导下的操作流程。

1. 首先，检查了main.xml文件，仔细观察之后发现小安将EditText标签，误写成EditView





了, 于是全部修改为EditText。并且输入框没有添加内容限制, 这样不仅会在输入框中输入数字, 也有可能输入汉字或者其他非数字型字符, 因此在前两个输入框中添加了一个属性, 指定输入的字符必须为小数 (decimal) 或整数 (integer), 语句如下:

```
android:numeric="decimal"
android:numeric="integer"
```

另外, 显示结果的那个输入框不需要用户进行操作, 因此可以将其设置为不获得焦点, 语句如下: `android:focusable="false"`

修改后的 main.xml 文件如下:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <EditText
        android:layout_width="60px"
        android:layout_height="wrap_content"
        android:numeric="decimal"
        android:id="@+id/num1"
    />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/add"
    />
    <EditText
        android:layout_width="60px"
        android:layout_height="wrap_content"
        android:numeric="integer"
        android:id="@+id/num2"
    />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/equ"
    />
    <EditText
        android:layout_width="80px"
        android:layout_height="wrap_content"
        android:focusable="false"
        android:id="@+id/result"
```





```
    />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cal"
        android:id="@+id/calbutton"
    />
</LinearLayout>
```

2. 之后再次运行程序。结果如下图：



3. 在前两个输入框分别输入 10 和 15，点击【计算】按钮。效果如下图：

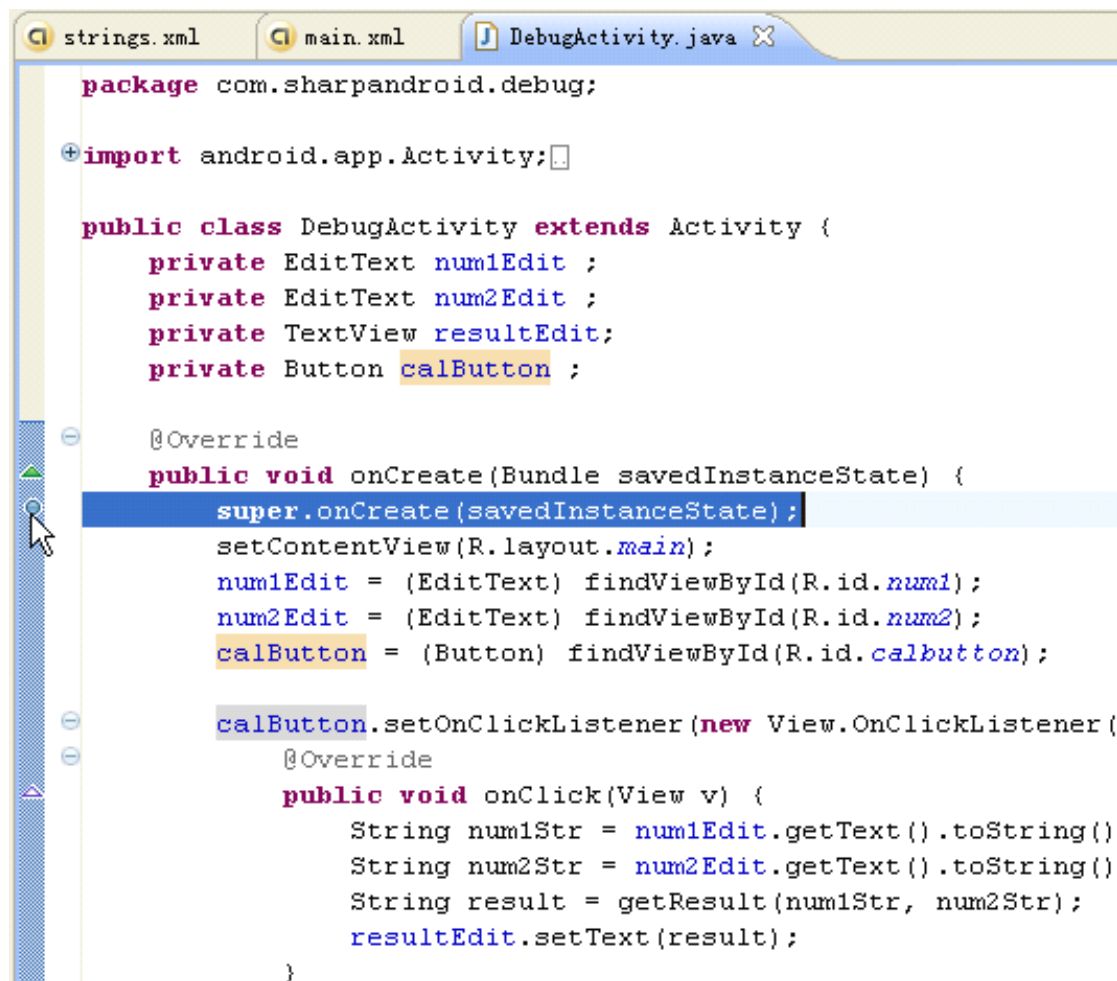






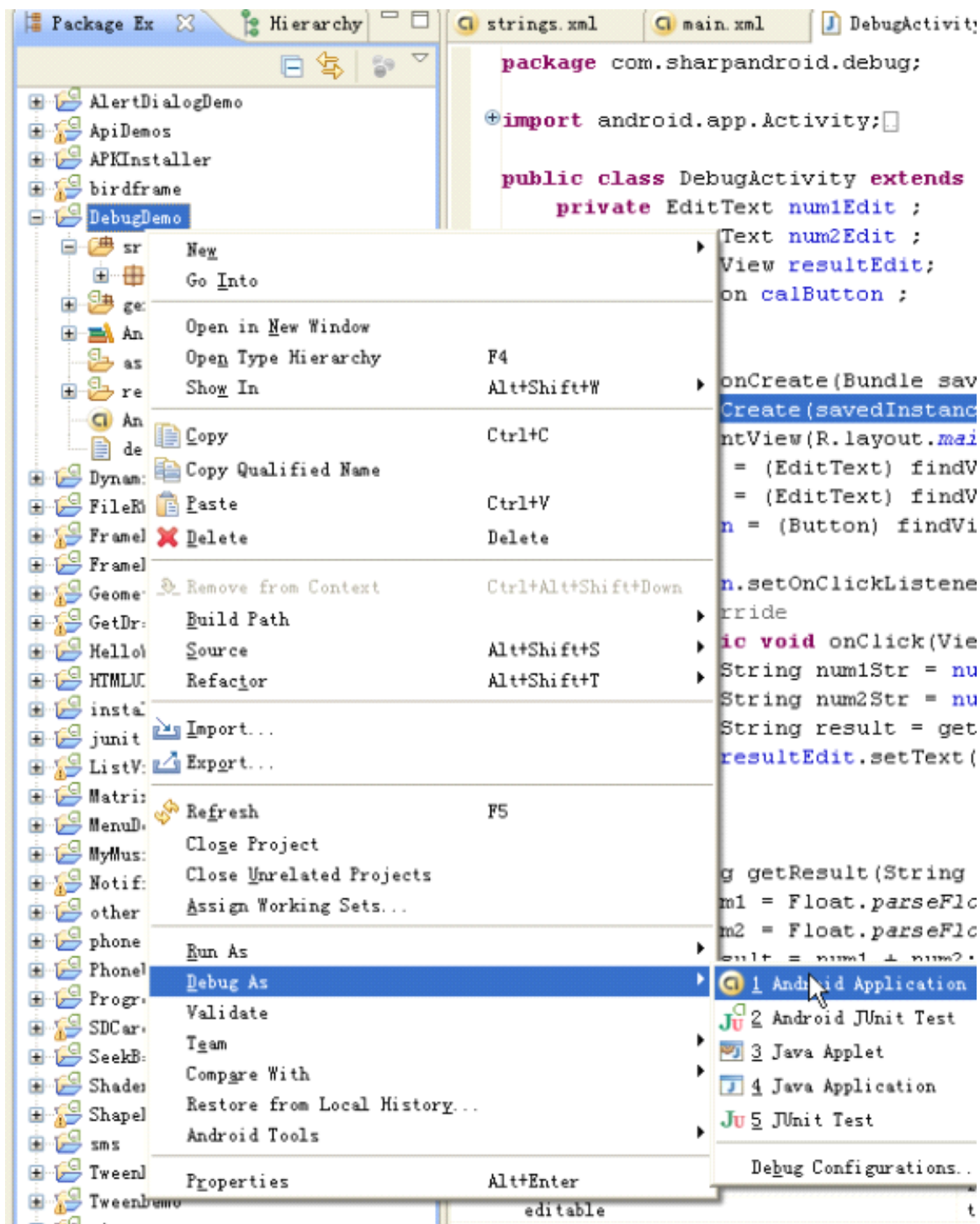
- 仍然有错误，但是此时界面已经没有错误，接着，开始对 Java 代码进行排错。
4. 首先为代码设置断点。断点的作用就是当调试程序的时候，程序执行到该行代码时会停止执行，等待开发人员的处理。设置断点的方法很简单，在 Eclipse 中的某行代码左侧（编辑器边框上）双击即可。将断点设置在 `onCreate()` 方法的第一行代码的位置，这样可以观察整个执行流程。如下图的位置：





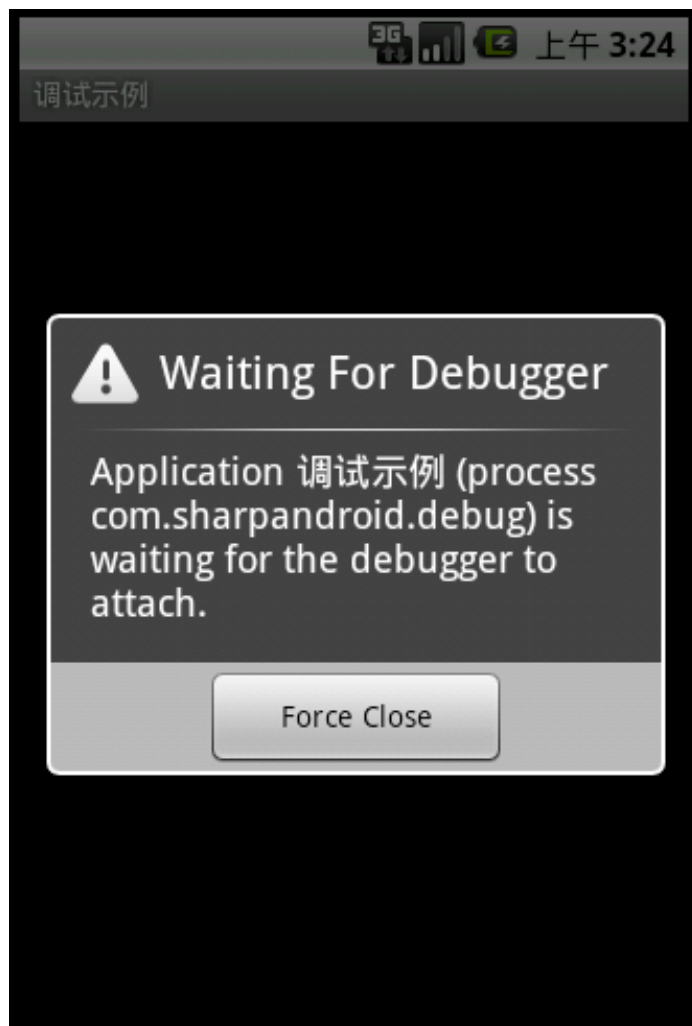
5. 设置完断点之后, 可以开始调试程序。需要注意的是调试程序和运行程序的方法有所区别。调试程序的启动方法是右击项目名, 选择【Debug As】→【Android Application】。如下图:





点击之后，观察模拟手机，显示如下界面，该界面询问用户是否进行调试，如果点击【Force Close】，则不进行调试，也不执行程序。因此我们不要点击【Force Close】按钮，片刻之后该提示就会自动消失。





此时模拟器界面如下:

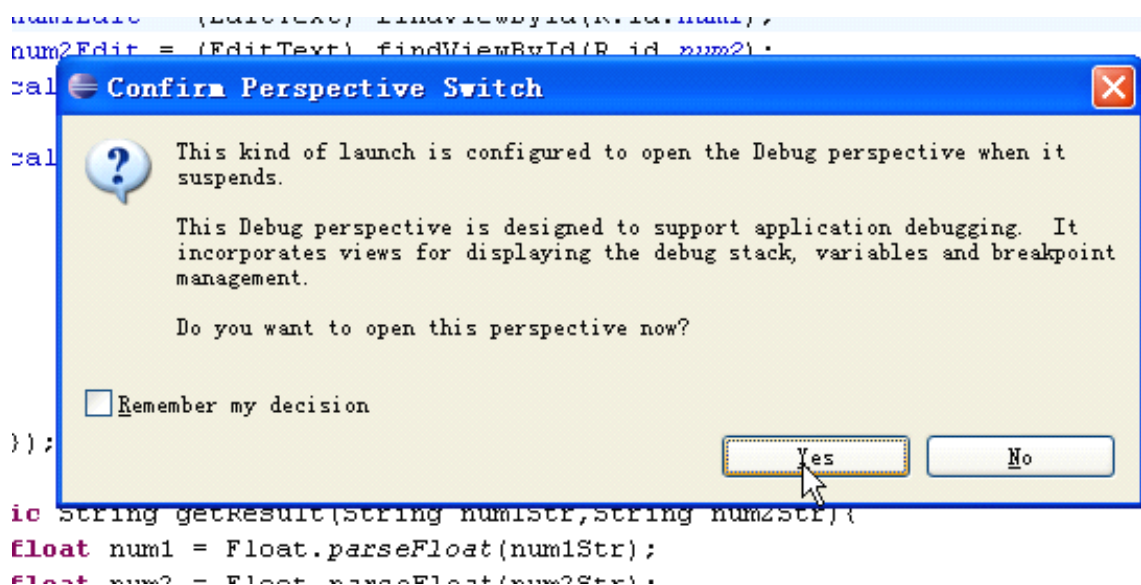


除了标题栏之外, 屏幕上没有任何内容显示。而此时在电脑下方的状态栏上 Eclipse 开发的状态条上显示为橙色, 如下图:

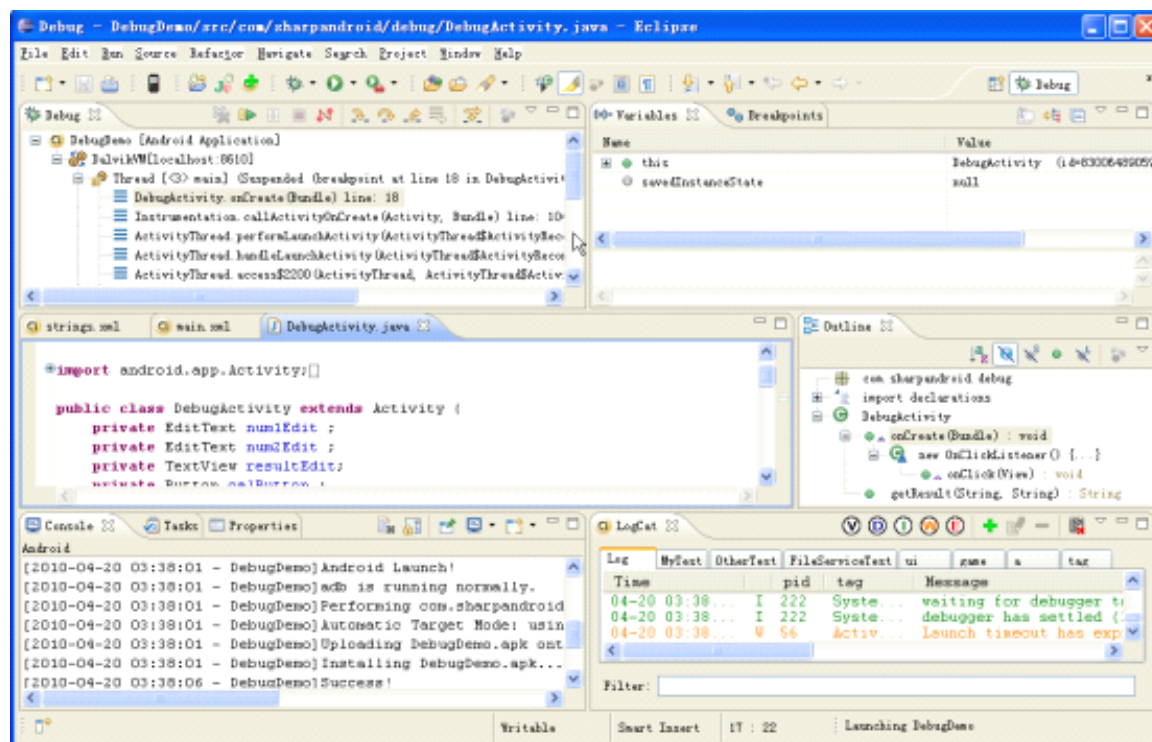




6. 点击查看出现什么状况, Eclipse 弹出如下提示, 询问是否打开 Debug 透视图。




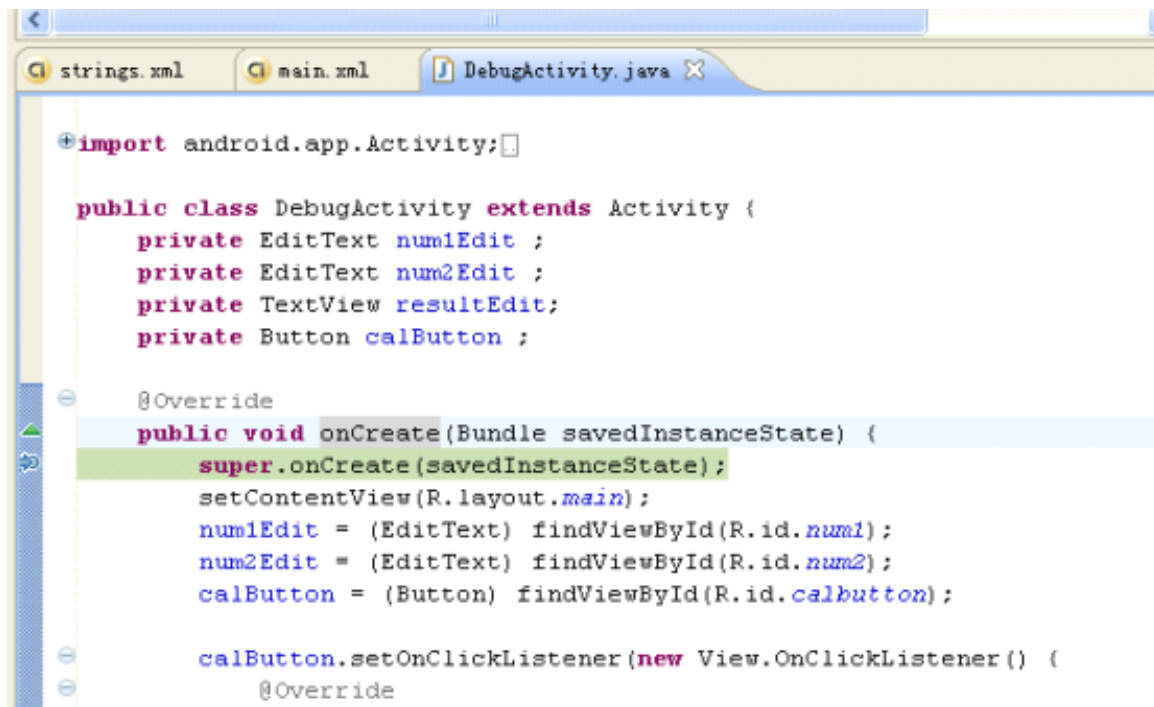
7. 点击【Yes】, 提示框消失, 弹出 Debug 视图。如下图:





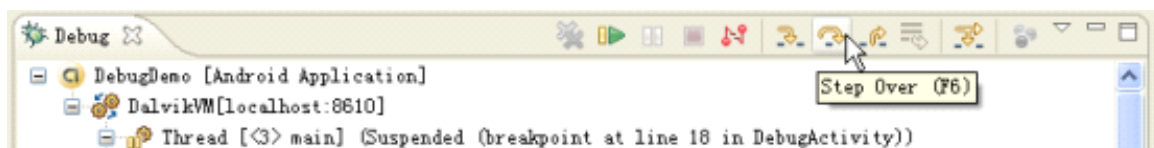





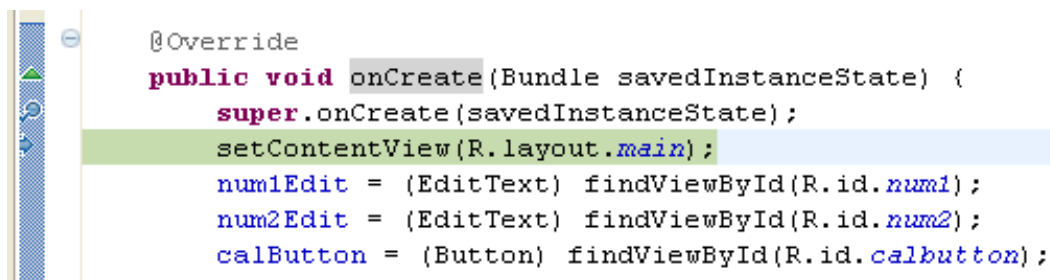
观察中间的代码窗口, 程序将要执行的行会以不同的背景色显示并且在其左侧会有一个的箭头指向该行, 如下图:



8. 想要程序继续执行, 可以点击标题为 Debug 的那一栏的按钮中的第二个按钮, 或者使用快捷键 F6, 其作用是单步跳过, 执行到下一行代码。如下图:



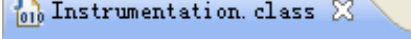

程序执行到下一行, 其下一行的背景色变化, 并且移至该行。




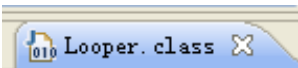
9. 继续点击按钮或者快捷键 F6。点击多次, 直到程序执行到一个陌生的类, 在 Eclipse



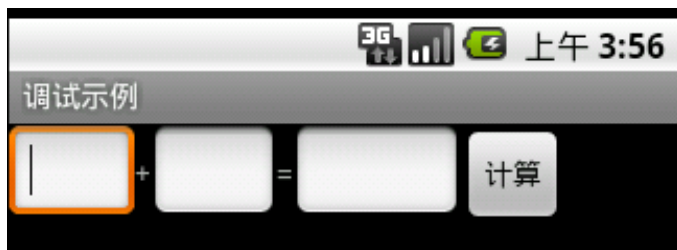


中间代码框上的标题为 ，此时代码已经执行到了 Android 框架为我们提供的处理代码，我们不必理会，此时应点击 

中的第三个按钮  或者快捷键 F7，作用是从该方法中返回。一直点击该按钮或者 F7

直到界面不发生变化为止，此时代码框上选项卡为 ，也即程序目前在 Looper 类内停止。

在此之前模拟器屏幕没有任何变化，为上面的空界面。此时再查看模拟器，发现我们的布局文件已经在屏幕上显示。如下图：



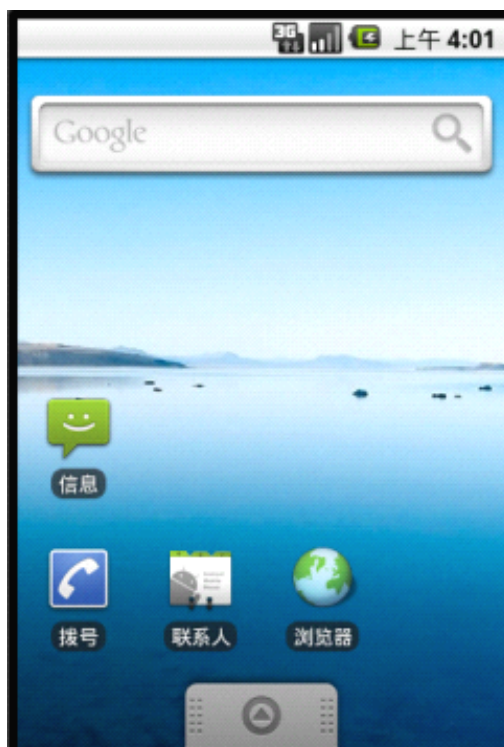
10. 如前，在输入框中输入数字，并点击【计算】按钮。界面无变化，进入 Eclipse 中查看。点击 F7，直到没有变化为止。查看界面，仍然提示错误，如下图：




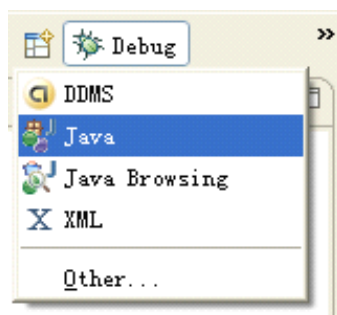




11. 点击【强行关闭】。模拟器退回主页，如下图：



12. 点击 Eclipse 右上方的  按钮，之后点击【Java】，如果没有【Java】选项，选择【Other】，之后再选择【Java】，进入 Java 视图，如下图：



13. 之前的程序我们已经测试没有错误，错误应该就在点击按钮所触发的事件的代码中，因此我们更改断点的位置，在 `onClick()` 方法内第一行代码处双击添加断点。在上一个 `super.onCreate(savedInstanceState);` 这一行前面的断点双击，取消该断点。如下图：





```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    num1Edit = (EditText) findViewById(R.id.num1);
    num2Edit = (EditText) findViewById(R.id.num2);
    calButton = (Button) findViewById(R.id.calbutton);

    calButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String num1Str = num1Edit.getText().toString();
            String num2Str = num2Edit.getText().toString();
            String result = getResult(num1Str, num2Str);
            resultEdit.setText(result);
        }
    });
}
```


14. 重复上面第 5 步的方式以【Debug As】的方式执行调试过程。

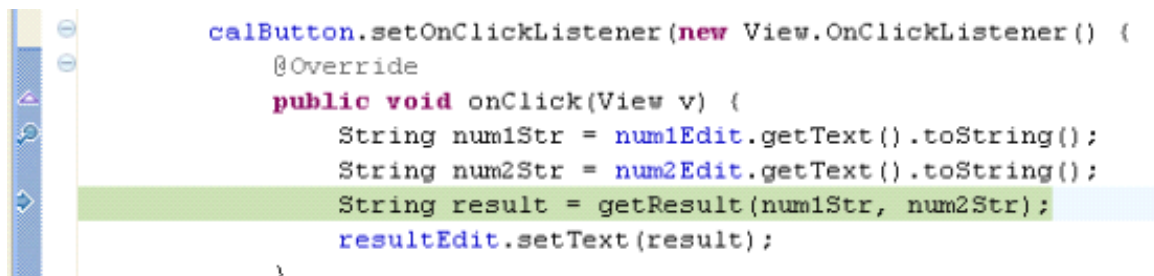
在屏幕中已经显示出界面，如前，在输入框中输入数字，并点击【计算】，进入 Eclipse 中查看，程序执行到了第 13 步所设的断点，如下：



```
calButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String num1Str = num1Edit.getText().toString();
        String num2Str = num2Edit.getText().toString();
        String result = getResult(num1Str, num2Str);
        resultEdit.setText(result);
    }
});
```

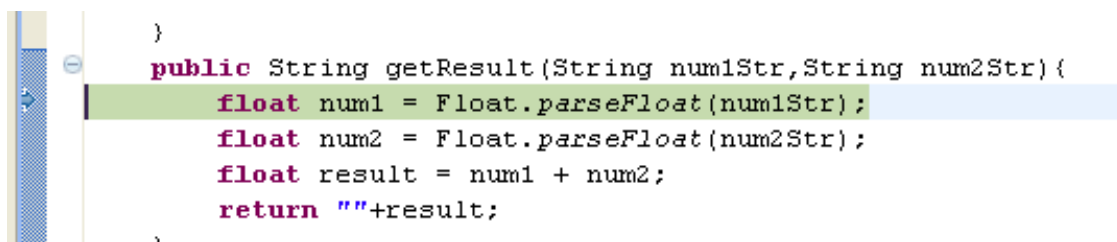





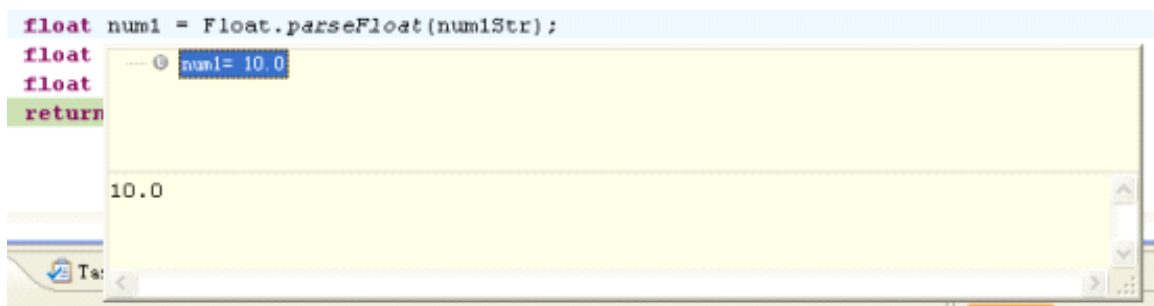
15. 点击 F5, 直到  指向 “String result = getResult(num1Str, num2Str);” 一行。如下图所示:



16. 此时点击  中的第一个按钮  或者快捷键 F5, 作用是单步跳入该行具有的方法, 即 getResult() 方法内部。点击 F5 后, 程序执行到 getResult() 方法内第一行, 效果如下图:

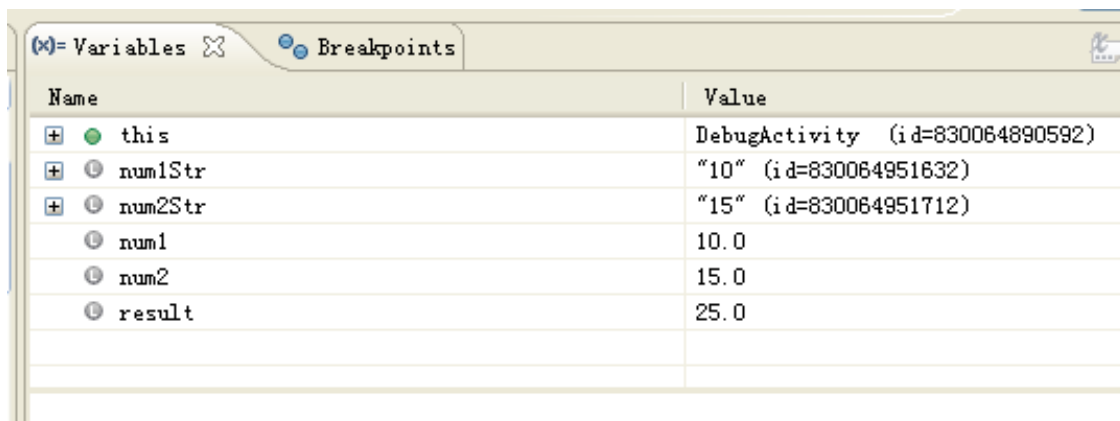


17. 点击 F6, 直到  指向该方法最后一行即 Return 语句时停止。
- 我们可以查看已经执行过的代码中的变量的值, 只需将鼠标在该变量上悬浮即可弹出提示框, 如下图。


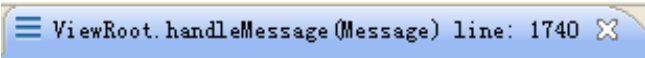


18. 另外在 Eclipse 右上角的 Variables 栏中也有已经执行过的代码行中的变量值。如下图。

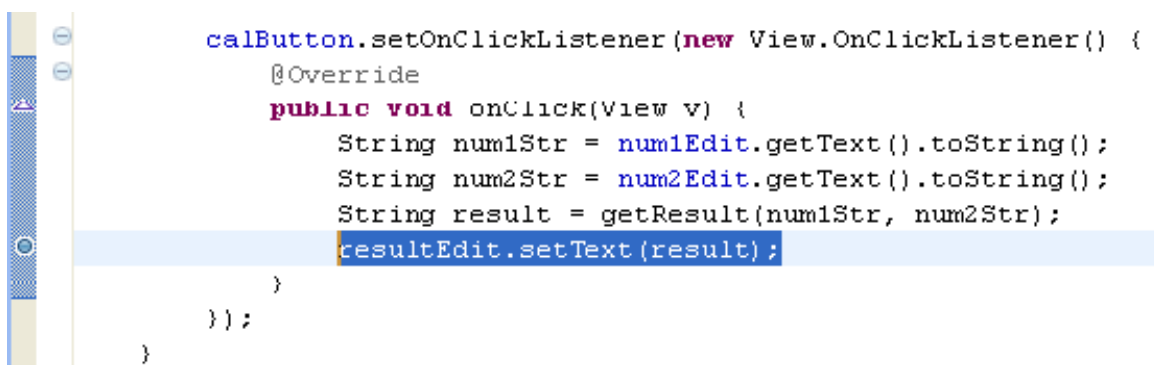




Name	Value
this	DebugActivity (id=830064890592)
num1Str	"10" (id=830064951632)
num2Str	"15" (id=830064951712)
num1	10.0
num2	15.0
result	25.0


19. 继续点击 F6, 或者 F7。  又指向 “String result = getResult(num1Str, num2Str);” 一行。点击 F6, 到下一行 “resultEdit.setText(result);”。再次点击 F6, 进入  类中。一直点击 F7 直到没有变化为止。此时查看模拟器, 又提示错误, 点击【强制关闭】。

由于是执行到 “resultEdit.setText(result);” 之后出错的, 现在可以定位错误就在该行。去除其他断点, 将断点设定在该行, 如下图:

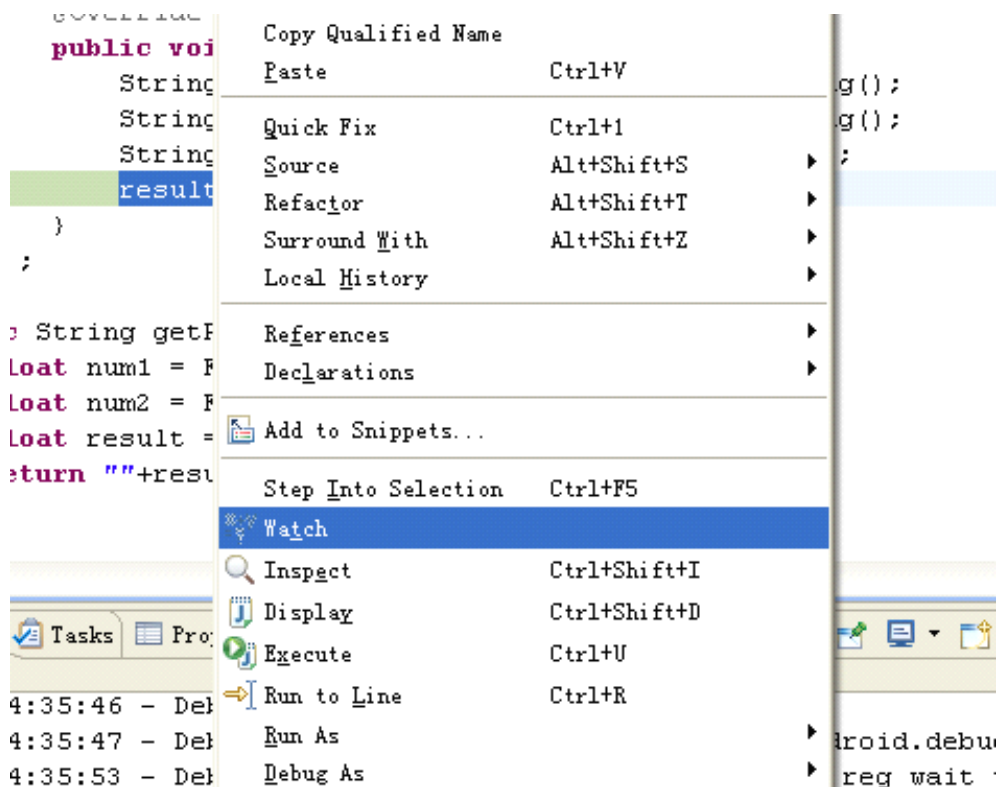



```
calButton.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        String num1Str = num1Edit.getText().toString();  
        String num2Str = num2Edit.getText().toString();  
        String result = getResult(num1Str, num2Str);  
        resultEdit.setText(result);  
    }  
});
```

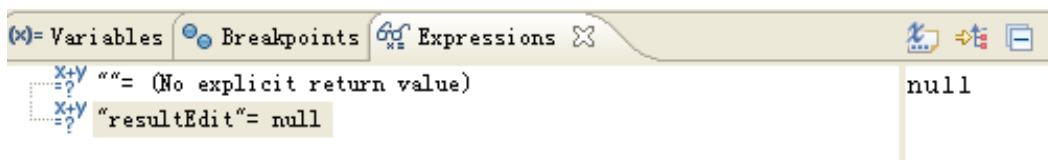
20. 按照第 5 步的方法, 以调试方式执行程序。在模拟器界面中填入数字, 点击【计算】。

进入 Eclipse 中, 目前  指向 “resultEdit.setText(result);” 一行。选中 resultEdit 这个变量, 右击鼠标, 选择【Watch】, 如下图:






查看右上角的  Expressions 栏，栏中信息如下：



发现 resultEdit=null。说明之前没有给 resultEdit 赋值，该对象为空，在 Java 中调用空对象的方法会发生一种常见的异常——空指针异常。这时错误已经找到，我们只需要在使用

resultEdit 对象之前为其添加赋值语句即可。点击 Debug 栏的 ，表示执行程序直到下一个断点为止，由于已没有其他断点，程序直接执行完毕。

21. 另外，大致博士对小安说，为代码添加注释是一个良好的习惯，于是为代码添加了详细的注释信息。

修改后的 DebugActivity.java 代码如下：

```
package com.sharpandroid.debug;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
```





```
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class DebugActivity extends Activity {
    private EditText num1Edit ;
    private EditText num2Edit ;
    private TextView resultEdit;
    private Button calButton ;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //得到相关对象
        num1Edit = (EditText) findViewById(R.id.num1);
        num2Edit = (EditText) findViewById(R.id.num2);
        // 少写的一行代码, 为 resultEdit 对象赋值。
        resultEdit = (TextView) findViewById(R.id.result);
        calButton = (Button) findViewById(R.id.calbutton);

        //为计算按钮添加单击监听器
        calButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //获取加数和被加数的字符串值
                String num1Str = num1Edit.getText().toString();
                String num2Str = num2Edit.getText().toString();
                //调用 getResult() 方法计算结果
                String result = getResult(num1Str, num2Str);
                //将结果在 resultEdit 上显示
                resultEdit.setText(result);
            }
        });
    }
    //将传入的两个字符串解析成 float 类型, 并求他们的和之后返回和。
    public String getResult(String num1Str, String num2Str) {
        //将第字符串解析成 float 类型
        float num1 = Float.parseFloat(num1Str);
        float num2 = Float.parseFloat(num2Str);
        //求它们的和
        float result = num1 + num2;
```





```
//在 float 型前面加上一个空字符串之后会以字符串形式返回结果
return ""+result;
}
}
```

22. 执行程序，输入数字，点击【计算】，效果如下：



此时已正确显示结果。程序调试通过。  
小安终于完成任务。

小安：博士，你太厉害了，要不是您我就不知道怎么办了，里面的错误还真不少，我以后一定小心。

大致：刚开始都这样，出错不可怕，只要能把错误调出来就 ok 了，调试错误的方法有很多种，我刚才教你的只是最基本和常用的，其他的你可要自己慢慢摸索了。加油了，小伙子！

小安：我一定继续努力，早日成为高手！呵呵。

