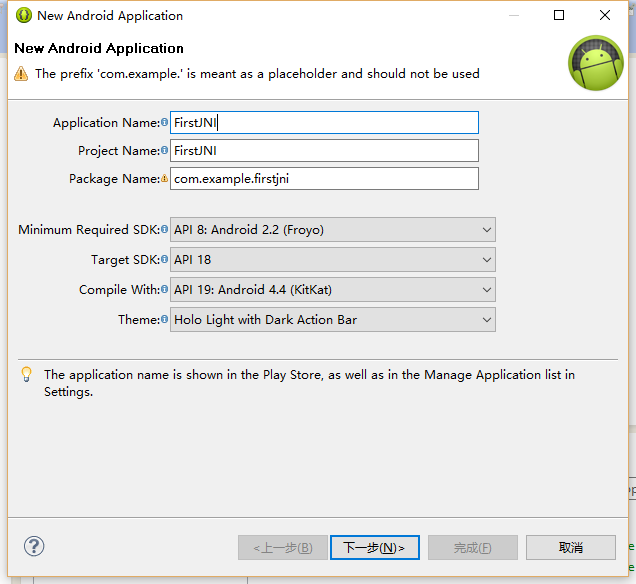
# Android开发中java与C++混合编程

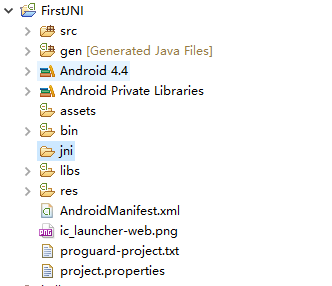
## 第1步：

打开eclipse，新建一个Android工程FirstJNI，在该工程中我们要使用C++开发部分函数。



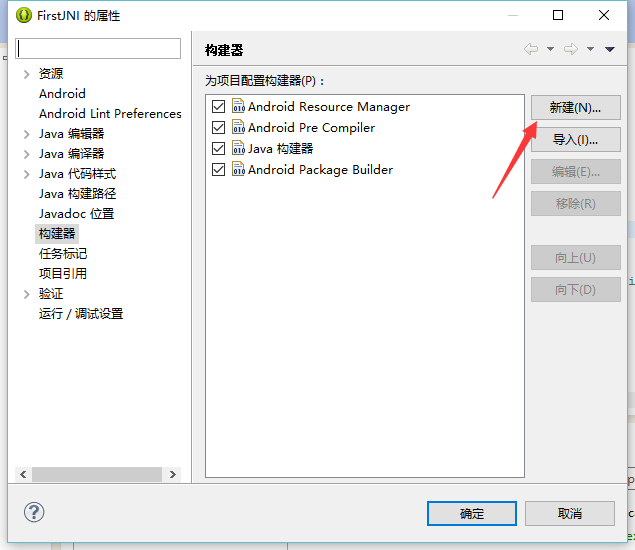
## 第2步：

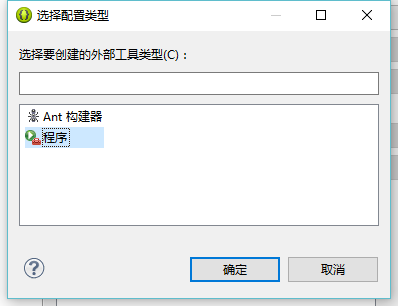
新建一个jni目录，在该目录下，我们要放置C++文件。



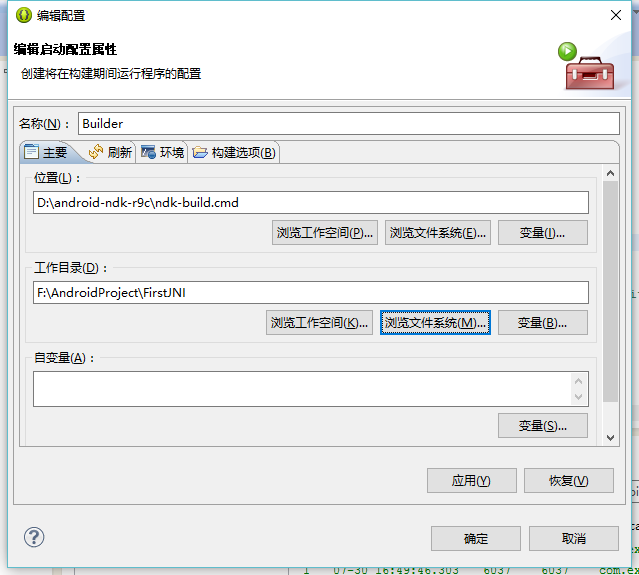
## 第3步：

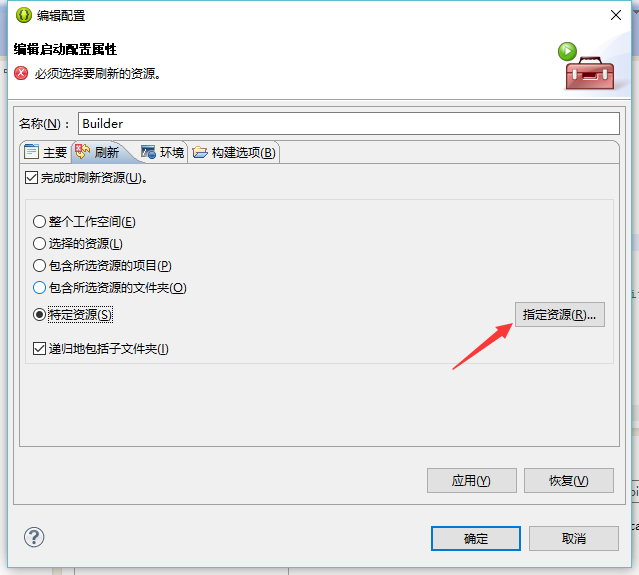
新建一个构建器，项目->属性

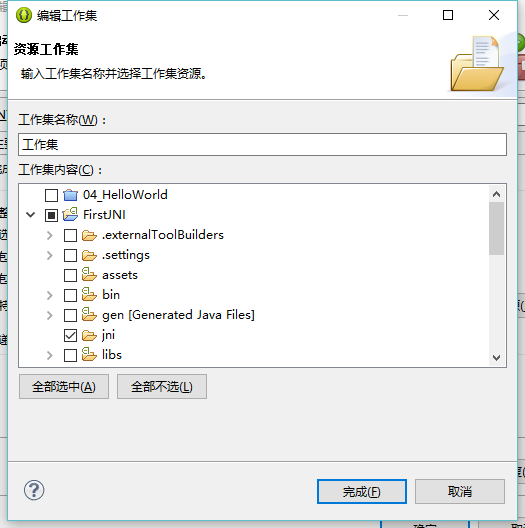




双击“程序”，出现编辑配置对话框，填入图中的值：

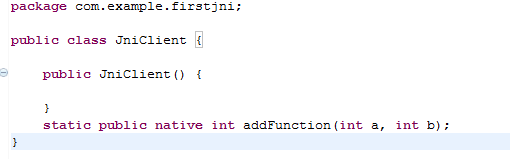






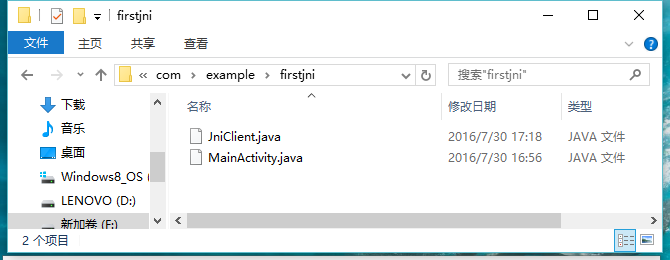
## 第4步：

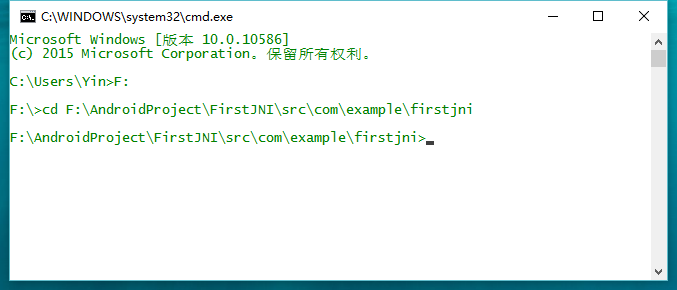
在com.example.firstJni包下新建一个JniClient类，在类中代码如下：



代码中的addFunction函数我们使用C++实现。

1. 打开cmd窗口，定位到F:\AndroidProject\FirstJNI\src\com\example\firstjni目录下

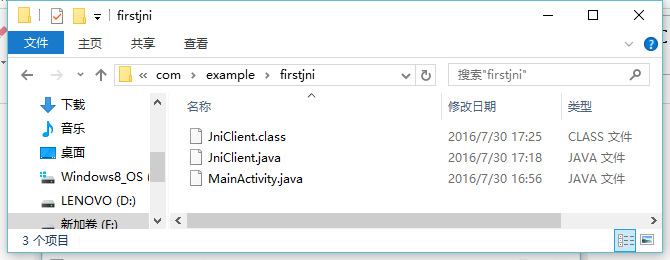




## 第5步：

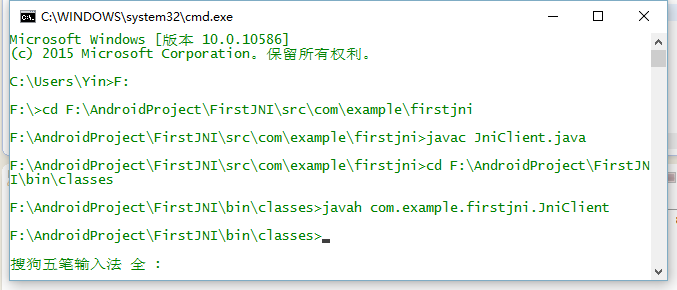
键入命令：javac JniClient.java生成JniClient.class文件：



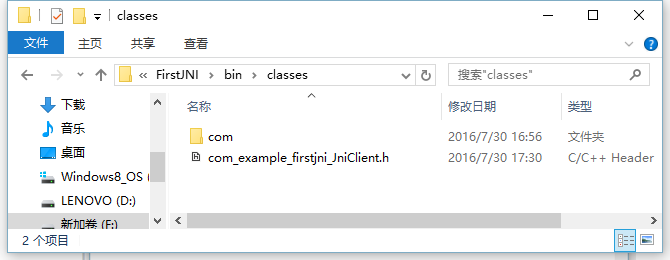


## 第6步：

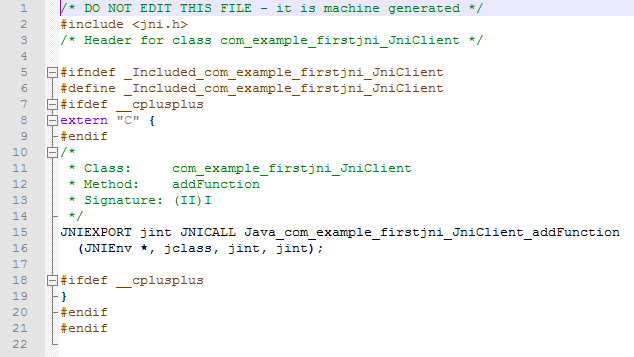
定位到F:\AndroidProject\FirstJNI\bin\classes目录，键入javah com.example.fristjni.JniClient命令。



在该目录下生成文件com\_example\_firstJni\_JniClient.h



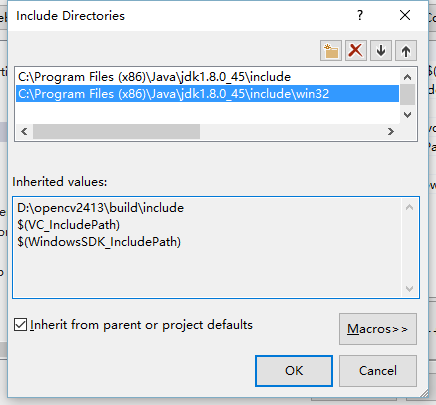
com\_example\_firstJni\_JniClient.h 内容如下：



## 第7步：

我们现在要实现Java\_com\_example\_firstjni\_JniClient\_addFunction函数。

打开visual studio，新建一个控制台工程，并添加上面生成的头文件，在工程的包含目录中添加以下目录：



在工程中添加一个com\_example\_firstjni\_JniClient.cpp文件，以实现addFunction函数，代码如下：

#include "com\_example\_firstjni\_JniClient.h"

#ifdef \_\_cplusplus

extern "C" {

#endif

/\*

\* Class: com\_example\_firstjni\_JniClient

\* Method: MakeMessage

\* Signature: (Ljava/lang/String;I)Ljava/lang/String;

\*/

int add(int a, int b)

{

return (a + b);

}

JNIEXPORT jint JNICALL Java\_com\_example\_firstjni\_JniClient\_addFunction(JNIEnv \*env, jclass obj, jint a , jint b)

{

jint res = add(a , b);

return res;

}

#ifdef \_\_cplusplus

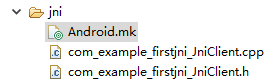
}

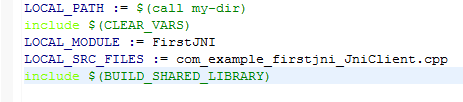
#endif

在以上代码中，定义了一个add函数，该函数完全是用C++编写的，该函数是内部函数，不会被导出，而addFuncion起到了一个接口的作用，它调用内部函数完成功能，是一个导出的外部函数，JAVA代码中通过调用该函数完成相应的功能。

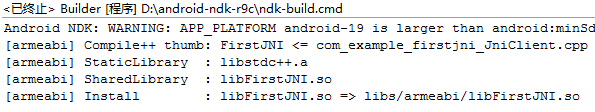
## 第8步：

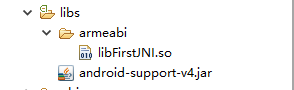
将com\_example\_firstjni\_JniClient.h和com\_example\_firstjni\_JniClient.cpp文件添加到eclipse中的FirstJNI的jni目录下，并创建一个Android.mk文件，内容如下：





保存工程，将自动编译生成libFirstJNI.so文件





该文件在libs目录下，注意，该文件名是模块名前加lib，但在后面我们引入该文件时，不需要lib。

## 第9步：

测试，依然在FisrtJNI工程中

布局文件：

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"*

tools:context=*".MainActivity"* >

<TextView

android:id=*"@+id/tv"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*/>

<Button

android:id=*"@+id/btn"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"确定"*/>

</LinearLayout>

MainActivity文件：

**package** com.example.firstjni;

**import** android.os.Bundle;

**import** android.app.Activity;

**import** android.view.View.OnClickListener;

**import** android.widget.\*;

**import** android.view.View;

**public** **class** MainActivity **extends** Activity {

Button btn;

TextView tv;

**static** {

System.*loadLibrary*("FirstJNI");

}

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

tv=(TextView)findViewById(R.id.*tv*);

btn=(Button)findViewById(R.id.*btn*);

btn.setOnClickListener(**new** OnClickListener(){

@Override

**public** **void** onClick(View v) {

**int** res=JniClient.*addFunction*(1, 2);

String str=String.*valueOf*(res);

tv.setText(str);

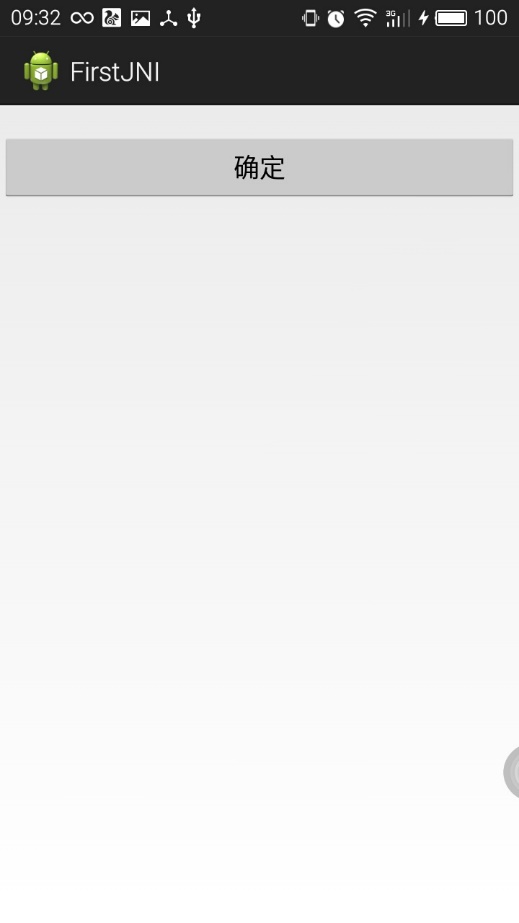
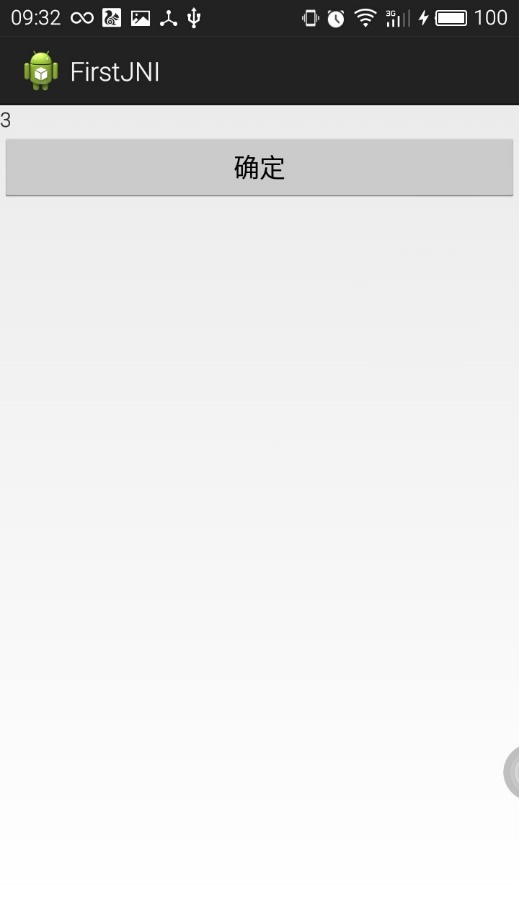
}

});

}

}

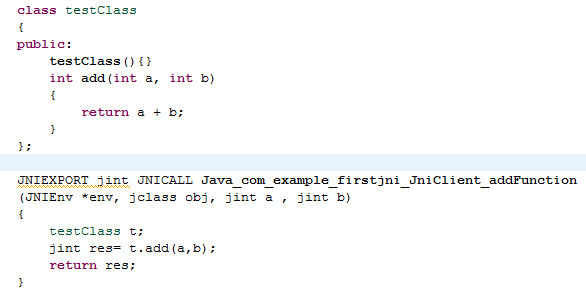
编译运行：

## 说明

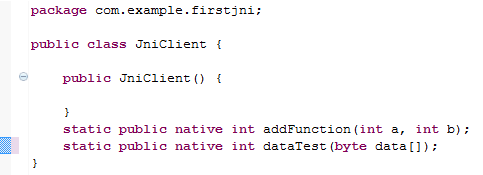
1、编译的so文件是与调用它的函数所在的类名及包名是一一对应的，每一个工程，如果要调用C++中的函数，都需要修改java文件，头文件，源文件，并重新编译so文件。

2、以上演示了android开发中调用C++函数的方法，在C++源文件中，接口函数可以调用该文件内的内部函数、类等，如同一般的C++开发一样。下面的代码在接口函数中使用了内部类，同样是可以的：



3、下面演示了数组的使用，在JAVA中有一byte数组，在C++中累加这些数组的值，返回累加结果：

JAVA文件：



头文件中

JNIEXPORT jint JNICALL Java\_com\_example\_firstjni\_JniClient\_dataTest

(JNIEnv \*, jclass, jbyteArray);

源文件中：

JNIEXPORT jint JNICALL Java\_com\_example\_firstjni\_JniClient\_dataTest

(JNIEnv \*env, jclass obj, jbyteArray data)

{

int len = env->GetArrayLength(data);

int sum = 0;

unsigned char\* pData = (unsigned char\*)env->GetByteArrayElements(data, 0);

for (int i = 0; i < len; i++)

{

sum = sum + pData[i];

}

return sum;

}