MICROOH 麦可网

Android-从程序员到架构师之路

出品人：Sundy
讲师：高焕堂（台湾）

http://www.microoh.com

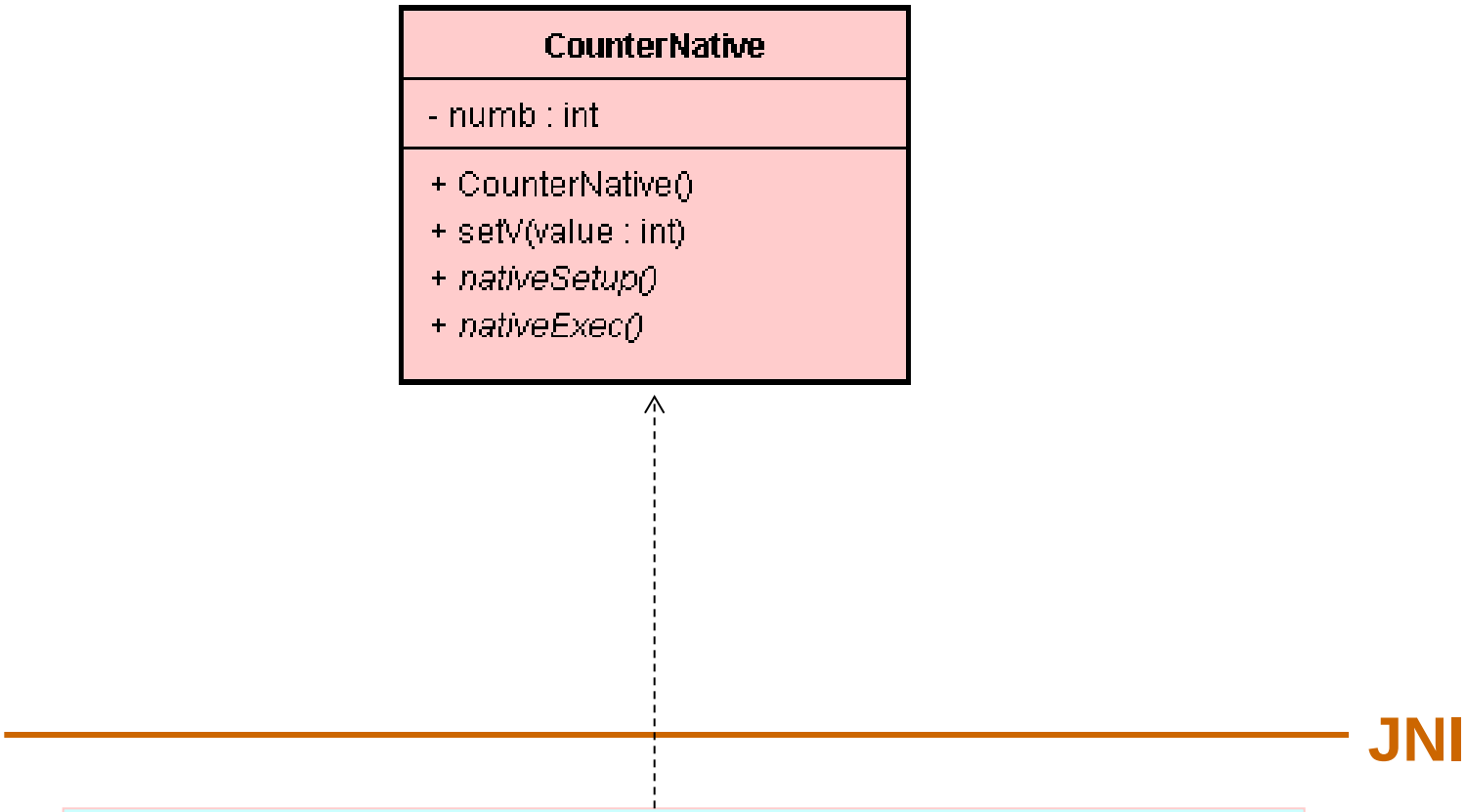
C03_d

JNI：从C调用Java函数 (d)

By 高煥堂

4、C存取Java对象的值

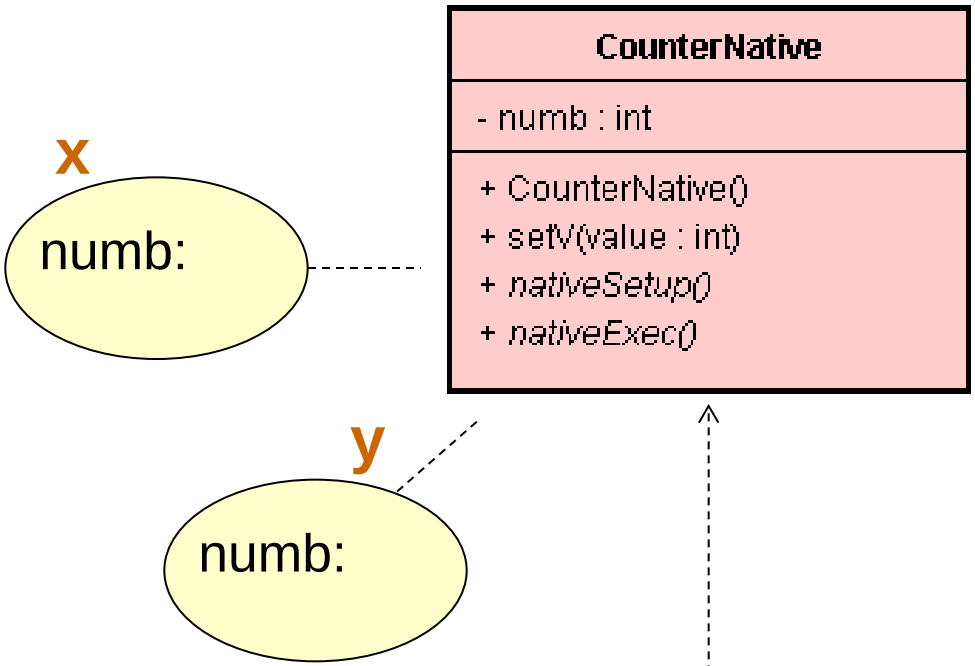
前言




```
/*  com.misoo.counter.CounterNative.c  */
// .....
JNIEXPORT void JNICALL
Java_com_misoo_counter_CounterNative_nativeExec
      (JNIEnv *env, jobject thiz) {
            //.....
}
```

在Java层创建2个对象

```
CounterNative x, y;
x = new CounterNative();
y = new CounterNative();
```

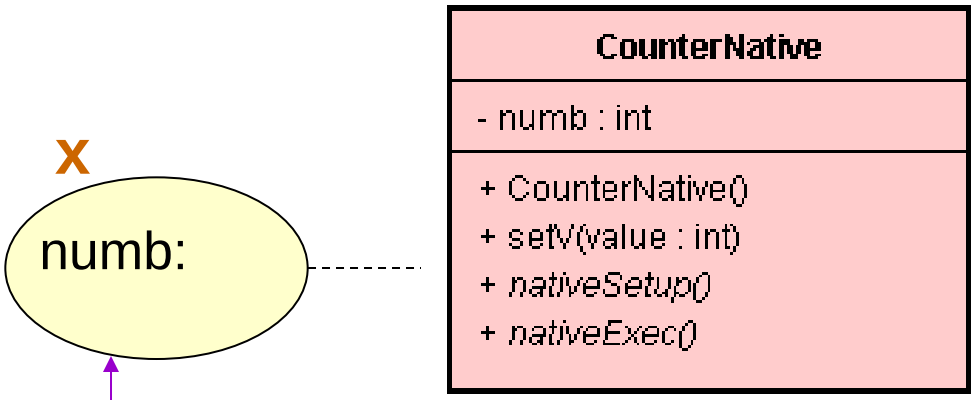


JNI

```
/*  com.misoo.counter.CounterNative.c  */
// .....
JNIEXPORT void JNICALL
Java_com_misoo_counter_CounterNative_nativeExec
      (JNIEnv *env, jobject thiz) {
            //.....
}
```

从Java函数调用C函数

```
x.nativeExec();
```

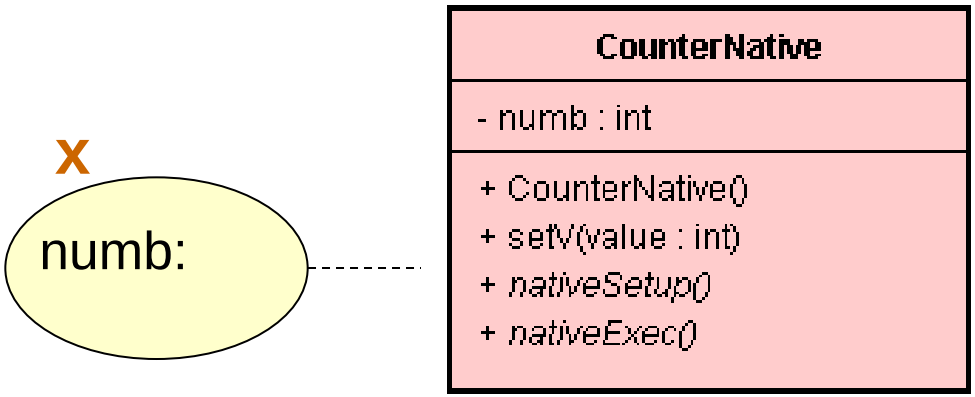


JNI

```
/*  com.misoo.counter.CounterNative.c  */
// .....
JNIEXPORT void JNICALL
Java_com_misoo_counter_CounterNative_nativeExec
      (JNIEnv *env, jobject thiz) {
            //.....
}
```

从Java函数调用C函数

```
y.nativeExec();
```



JNI

```
/*  com.misoo.counter.CounterNative.c  */
// .....
JNIEXPORT void JNICALL
Java_com_misoo_counter_CounterNative_nativeExec
      (JNIEnv *env, jobject thiz) {
            //.....
}
```

基本步骤

0. 有了Java层对象(thiz)
1. 问这个对象thiz的类，得到clazz

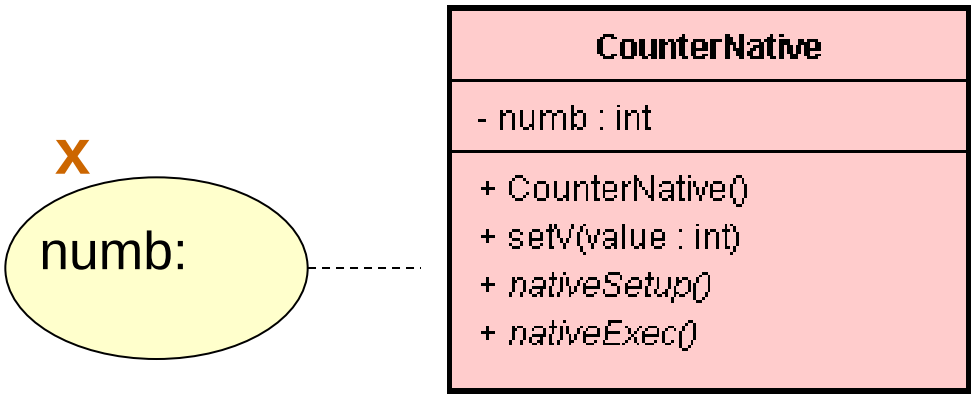
```
jclass clazz = (*env)->GetObjectClass(env, thiz);
```

2. 问这个类里的setV()函数，得到methodID

```
m_mid = (*env)->GetMethodID(env, clazz, "setV", "(I)V");
```

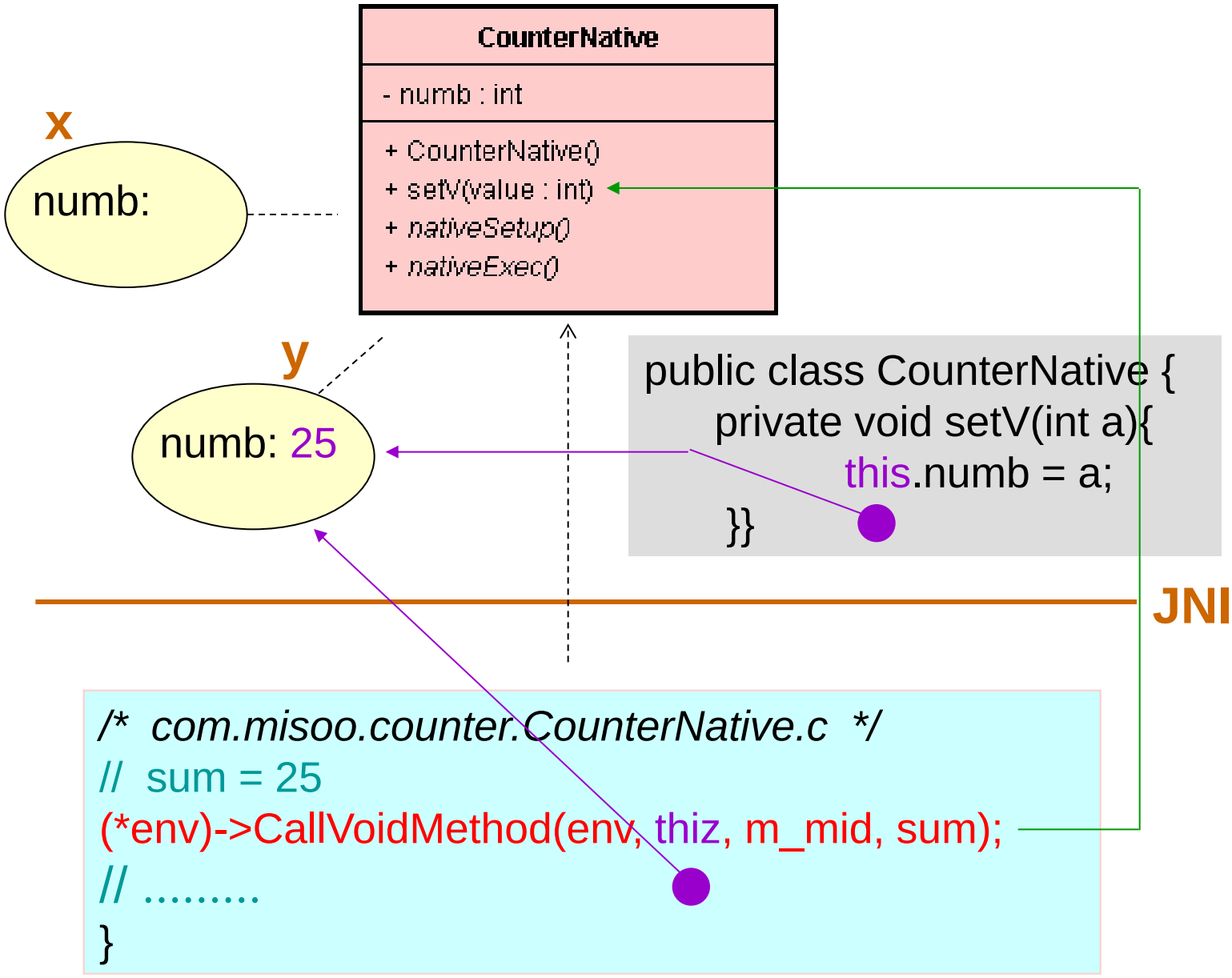
3. 基于methodID和thiz，调用setV()函数

```
int sum = 25;
(*env)->CallVoidMethod(env, thiz, m_mid, sum);
```



JNI

```
/*  com.misoo.counter.CounterNative.c  */
//  sum = 25
(*env)->CallVoidMethod(env, thiz, m_mid, sum);
// .........
}
```


CounterNative
- numb : int
+ CounterNative()
+ setV(value : int)
+ nativeSetup()
+ nativeExec()
x
numb:
y
numb: 25
public class CounterNative {
private void setV(int a){
this.numb = a;
}}
JNI
/* com.misoo.counter.CounterNative.c */
// sum = 25
(*env)->CallVoidMethod(env, thiz, m_mid, sum);
// .....
}

C函数直接存取属性值

- 刚才是透过函数调用(function call)来存取Java对象的属性值。
- C函数也能直接存取属性值。

基本步骤

0. 有了Java层对象(thiz)
1. 问这个对象thiz的类，得到clazz

```
jclass clazz = (*env)->GetObjectClass(env, thiz);
```

2. 问这个类里的numb属性，得到fieldID

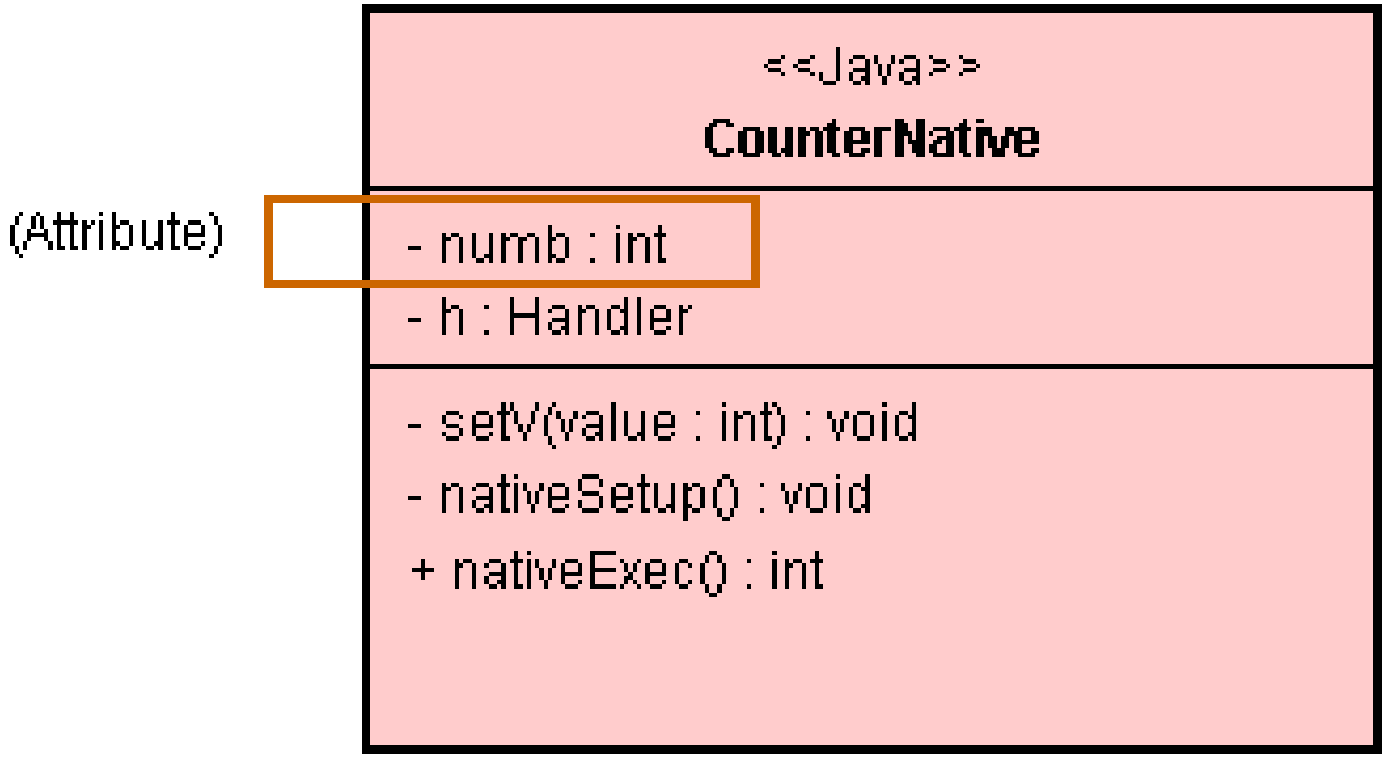
```
m_fid = (*env)->GetFieldID(env, clazz, "numb", "I");
```

3. 基于fieldID和thiz，直接存取numb值

```
n = (int)(*env)->GetObjectField(env, m_object, m_fid);
```

范例

- 例如，在CounterNative里可定义numb等属性，如下：


(Attribute)
<<Java>>
CounterNative
- numb : int
- h : Handler
- setV(value : int) : void
- nativeSetup() : void
+ nativeExec() : int

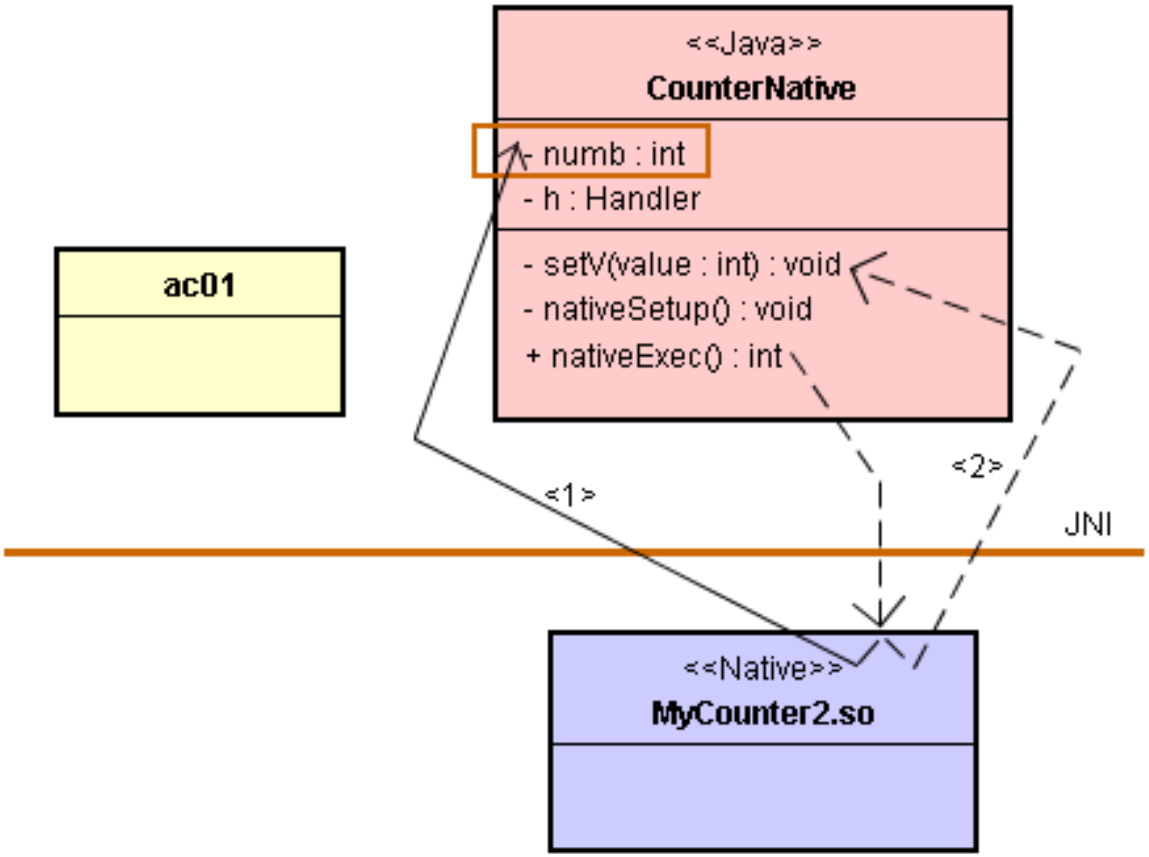
拿目前对象换取它的类

拿此类换取某属性ID

依据对象和属性ID，取得属性值

调用setV()函数，将sum回传到Java层

范例代码


<<Java>>
CounterNative
- numb : int
- h : Handler
- setV(value : int) : void
- nativeSetup() : void
+ nativeExec() : int
ac01
<1>
<2>
JNI
<<Native>>
MyCounter2.so

```
// ac01.java
// .........
public class ac01 extends Activity implements OnClickListener {
    static public ac01 ref;

    @Override
    public void onCreate(Bundle savedInstanceState){
            ref = this;
            // ...........
     }
    @Override public void onClick(View v) {
          switch(v.getId()){
          case 101:
              CounterNative cn = new CounterNative();
              cn.nativeExec();    break;
          case 103:
              finish();  break;
  }}}
```

- 指令：cn.nativeExec()。由于nativeExec()是个本地函数，就转而调用到com_misoo_counter_CounterNative_nativeExec()函数。
- 其先取得Java层的numb值，计算出sum值，再调用Java层的setV()函数，显示出来。

```
// CounterNative.java
// .....
public class CounterNative {
        private static Handler h;
        private int numb;
        static {  System.loadLibrary("MyCounter2");  }

        public CounterNative(){
            h = new Handler(){
                public void handleMessage(Message msg) {
                        ac01.ref.setTitle(msg.obj.toString());
             }};
            numb = 25;
            nativeSetup();
         }
```

```
    private void setV(int value){
        String str = "Value = " + String.valueOf(value);
        Message m = h.obtainMessage(1, 1, 1, str);
        h.sendMessage(m);
    }
    private native void nativeSetup();
    public native void nativeExec();
}
```

- 由于本地的C函数仍属于CounterNative类的一部分，所以C函数仍可以调用到CounterNative类的private函数(如setV()函数)。
- 此外，本地函数nativeSetup()只提供给建构函数来调用，而不给其它类别使用，所以可以将nativeSetup()宣告为private函数。

```
/* com.misoo.counter.CounterNative.c */
#include "com_misoo_counter_CounterNative.h"
jobject   m_object;
jmethodID m_mid;
jfieldID  m_fid;

JNIEXPORT void JNICALL
Java_com_misoo_counter_CounterNative_nativeSetup
    (JNIEnv *env, jobject thiz) {
            jclass clazz = (*env)->GetObjectClass(env, thiz);
            m_object = (jobject)(*env)->NewGlobalRef(env, thiz);
            m_mid = (*env)->GetMethodID(env, clazz, "setV", "(I)V");
            m_fid = (*env)->GetFieldID(env, clazz, "numb", "I");
            return;
 }
```

```
JNIEXPORT void JNICALL
Java_com_misoo_counter_CounterNative_nativeExec
  (JNIEnv *env, jobject thiz) {
        int n, i, sum = 0;
        n = (int)(*env)->GetObjectField(env, m_object, m_fid);
        for(i=0; i<=n; i++)
              sum+=i;
        (*env)->CallVoidMethod(env, m_object, m_mid, sum);
        return;
   }
```

- 其中的thiz就是从Java层传递过来的对象指针。首先将thiz传给VM的GetObjectClass()函数，取得该对象的类指针(即clazz)。
- 接着，将clazz传给VM的GetFieldID()函数来取得numb属性的ID。

- 当ac01调用CounterNative类的nativeExec()本地函数时，就转而调用C语言的nativeExec()函数。
- 这个C函数调用VM的GetObjectField()函数，使用刚才取得的m_fid值，而取得CounterNative类的对象里的numb属性值。

~ Continued ~