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

C04_d

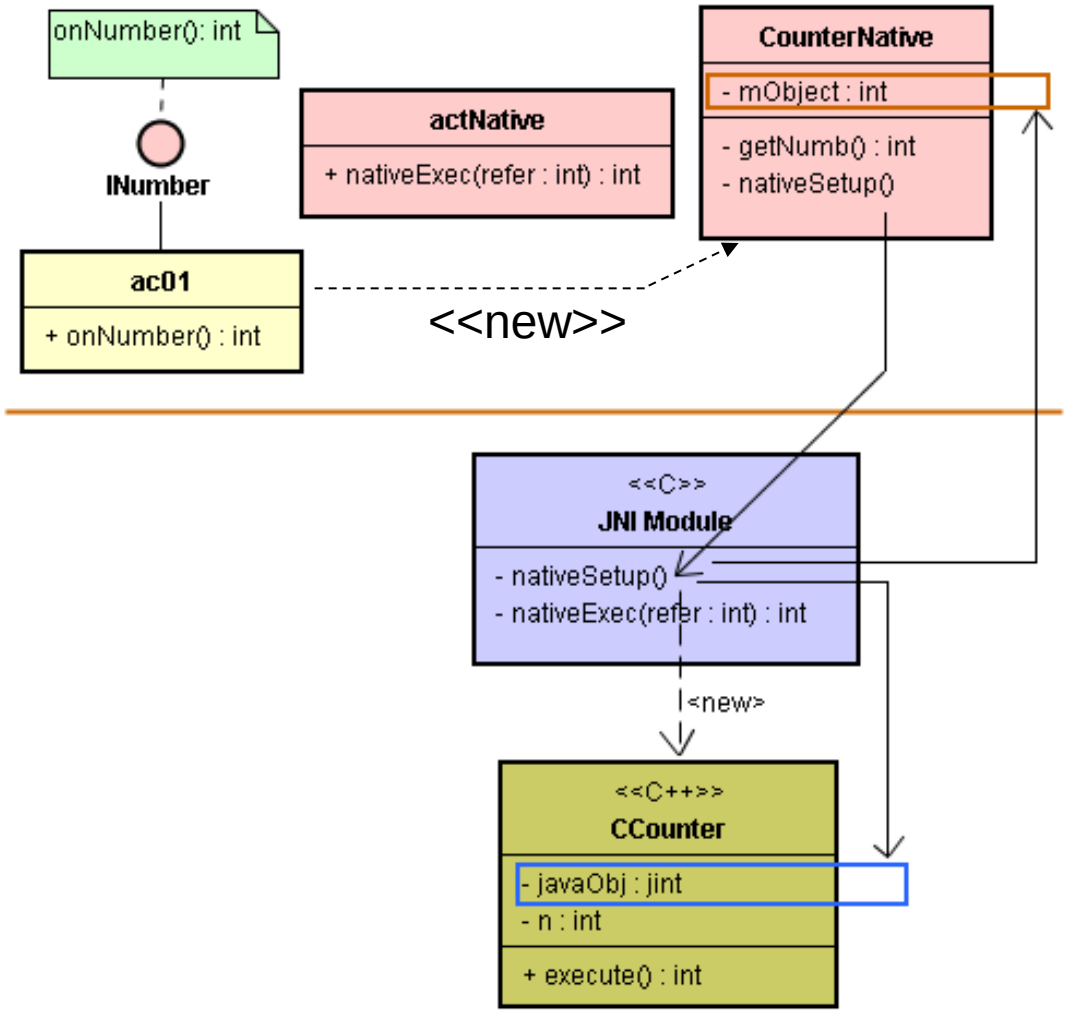
JNI：必要的优化设计(d)

By 高煥堂

5、Java与C++对象之间的<双向>对称关连

举例说明

- 上一节里，将C++对象指针储存于Java对象的属性里；成为<单向>的对称联结关系。
- 接下来，也可以将Java对象的参考储存于C++对象里。


onNumber(): int
INumber
actNative
+ nativeExec(refer : int) : int
CounterNative
- mObject : int
- getNumb() : int
- nativeSetup()
ac01
+ onNumber() : int
<<new>>
<<C>>
JNI Module
- nativeSetup()
- nativeExec(refer : int) : int
<new>
<<C++>>
CCounter
- javaObj : jint
- n : int
+ execute() : int

```
// INumber.java
package com.misoo.counter;
public interface INumber {
	int onNumber();
}
```

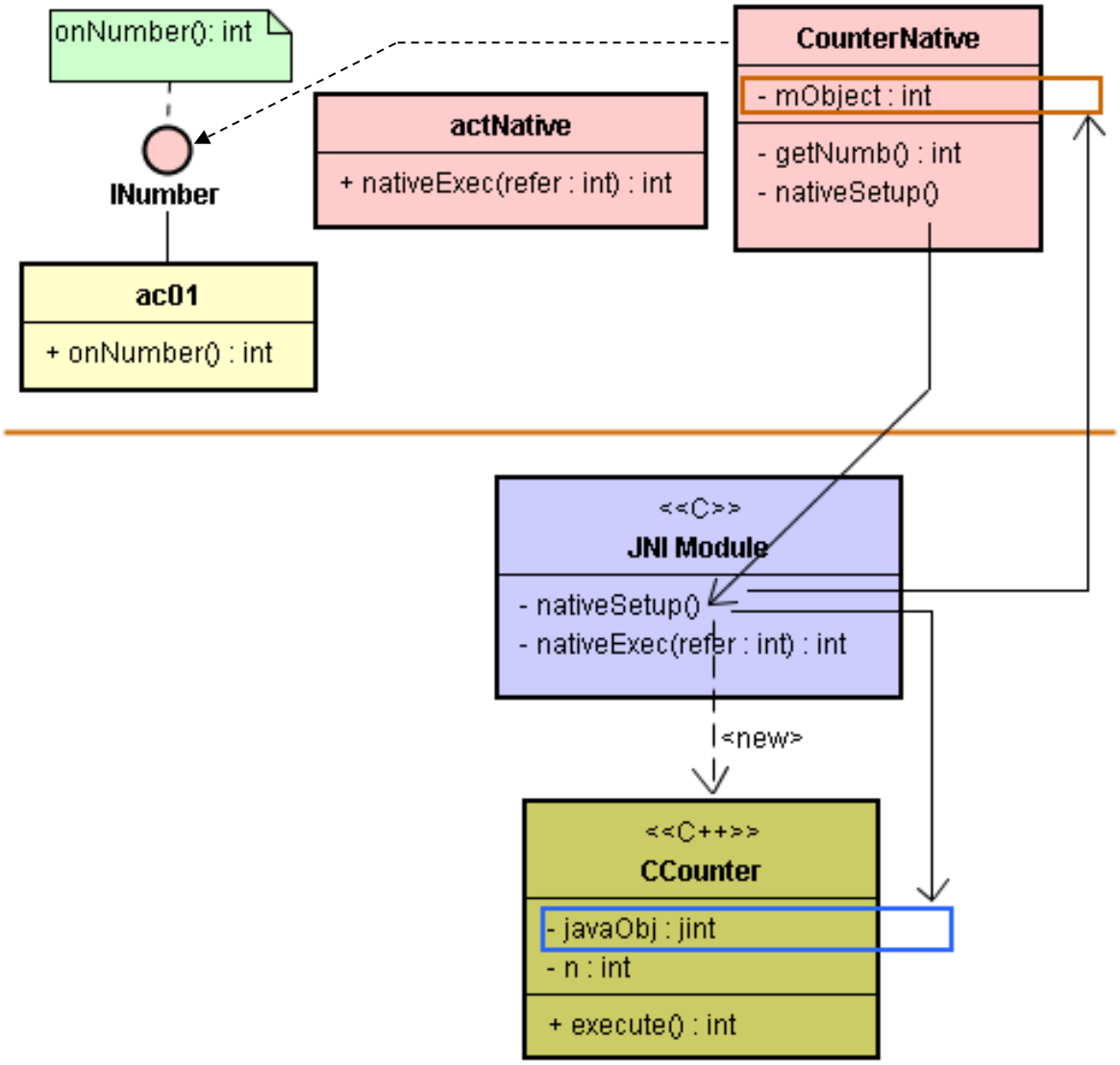
```
// ac01.java
// ……
public class ac01 extends Activity
                          implements OnClickListener, INumber {
  private CounterNative cn;

  @Override public void onCreate(Bundle savedInstanceState){
    //…….-
    cn = new CounterNative();
    cn.setOnNumber(this);
   }
```

```
@Override public void onClick(View v) {
        int sum;
        switch(v.getId()){
        case 101:
                sum = actNative.nativeExec(cn.mObject);
                setTitle("Sum = " + sum);
                break;
        case 103:
                finish();    break;
         }
 }
@Override public int onNumber()   {  return 17;  }
}
```

- 指令：

```
@Override public void onCreate(Bundle
    savedInstanceState){
    //…….-
    cn = new CounterNative();
    cn.setOnNumber(this);
  }
```


onNumber(): int
INumber
ac01
+ onNumber() : int
actNative
+ nativeExec(refer : int) : int
CounterNative
- mObject : int
- getNumb() : int
- nativeSetup()
<<C>>
JNI Module
- nativeSetup()
- nativeExec(refer : int) : int
<new>
<<C++>>
CCounter
- javaObj : jint
- n : int
+ execute() : int

```
// CounterNative.java
package com.misoo.counter;
public class CounterNative {
        public int mObject;
        private INumber listener;

       static {
              System.loadLibrary("MyCounter8");   }
        public CounterNative()
                        {  nativeSetup();   }
        public void setOnNumber(INumber plis)
                        {  listener = plis;   }
        private int getNumb()
                        {  return listener.onNumber();    }
        private native void nativeSetup();
}
```

```
/* com.misoo.counter.CounterNative.cpp */
#include "com_misoo_counter_actNative.h"
#include "com_misoo_counter_CounterNative.h“

class CCounter{
  public:
        int n;
        jint javaObj;
  public:
        CCounter() {}
        int execute() {
                    int i, sum = 0;
                    for(i=0; i<=n; i++)  sum+=i;
                    return sum;
          }
};
```

```
JNIEXPORT void JNICALL
Java_com_misoo_counter_CounterNative_nativeSetup
  (JNIEnv *env, jobject thiz) {
        CCounter *obj = new CCounter();
        jclass clazz = (jclass)env->GetObjectClass(thiz);
        jfieldID fid =
              (jfieldID)env->GetFieldID(clazz, "mObject", "I");
        env->SetIntField(thiz, fid, (jint)obj);
        jobject gThiz = (jobject)env->NewGlobalRef(thiz);
        obj->javaObj = (jint)gThiz;
}
```

```
JNIEXPORT jint JNICALL
Java_com_misoo_counter_actNative_nativeExec
  (JNIEnv *env, jclass clazz, jint refer) {
        CCounter *co = (CCounter*)refer;
        jobject jo = (jobject)co->javaObj;
        jclass joClazz = (jclass)env->GetObjectClass(jo);
        jmethodID mid = env->GetMethodID(joClazz,
                                  "getNumb", "()I");
        int numb = (int)env->CallIntMethod(jo, mid);
        co->n = numb;
        return (jint)co->execute();
}
```

关于nativeSetup()函数的动作

- 上述nativeSetup()函数里的指令：

```
CCounter *obj = new CCounter();
```

诞生一个CCounter对象。

- 指令：

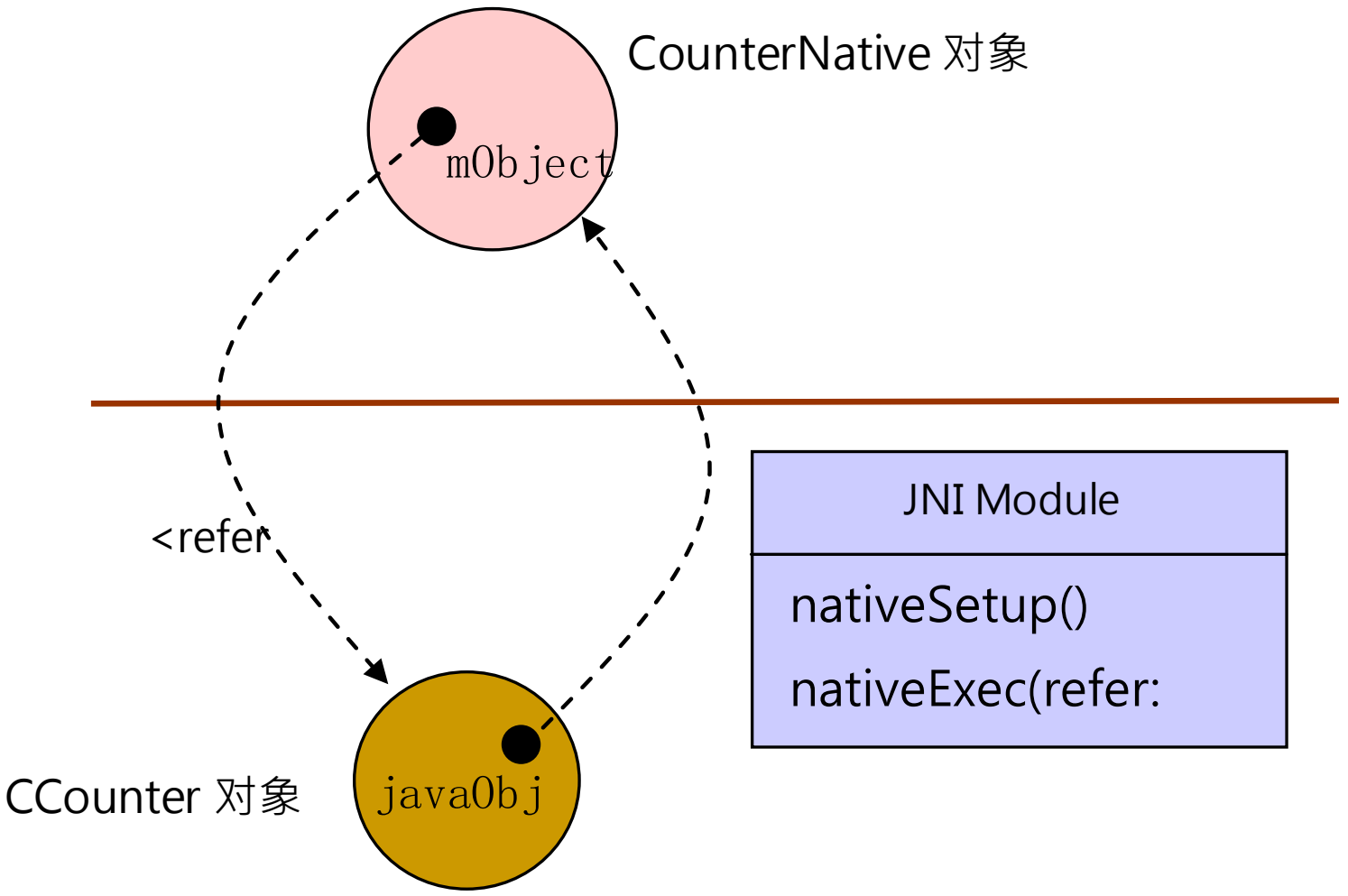
```
jfieldID fid = (jfieldID)env->GetFieldID(clazz,
                                         "mObject", "I");
env->SetIntField(thiz, fid, (jint)obj);
```

- 就将CCounter对象的指针值储存于CounterNative对象的mObject属性里。

- 指令：

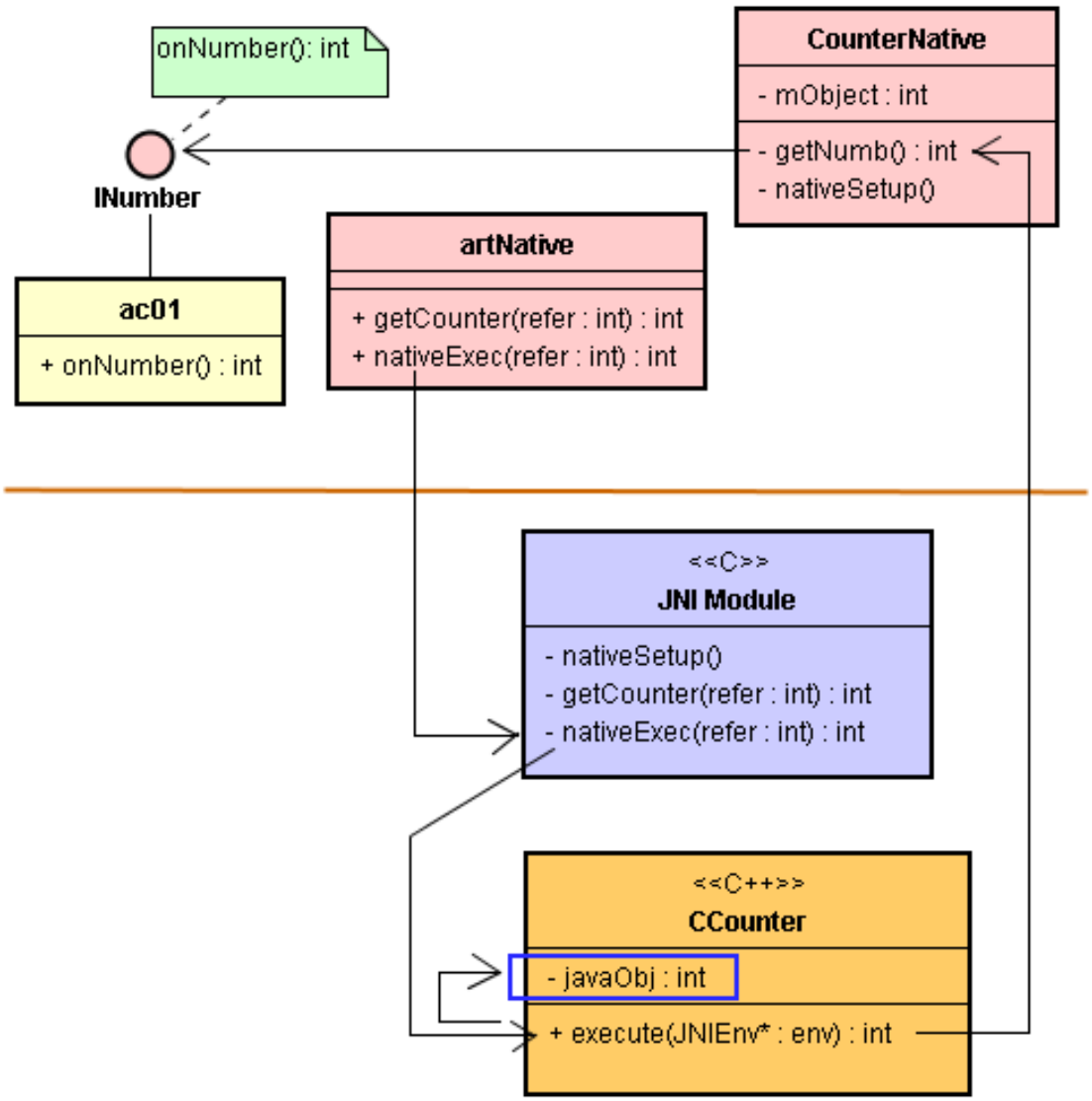
```
jobject gThiz = (jobject)env->NewGlobalRef(thiz);
obj->javaObj = (jint)gThiz;
```

- 就將CounterNative对象的指针值储存于CCounter对象的javaObj属性里，如此建立了CounterNative对象与CCounter对象之双向连结。如下图：


CounterNative 对象
mObject
<refer
JNI Module
nativeSetup()
nativeExec(refer:
CCounter 对象
javaObj

关于nativeExec()函数的动作

- 当ac01调用这个函数时，将CCounter对象的参考值传递给JNI层的nativeExec()本地函数。


onNumber(): int
INumber
ac01
+ onNumber() : int
CounterNative
- mObject : int
- getNumb() : int
- nativeSetup()
artNative
+ getCounter(refer : int) : int
+ nativeExec(refer : int) : int
<<C>>
JNI Module
- nativeSetup()
- getCounter(refer : int) : int
- nativeExec(refer : int) : int
<<C++>>
CCounter
- javaObj : int
+ execute(JNIEnv* : env) : int

// ac01.java
// .....

```
@Override public void onClick(View v) {
        int sum;
        switch(v.getId()){
        case 101:
                sum = actNative.nativeExec( cn.mObject );
                setTitle("Sum = " + sum);
                break;
        case 103:
                finish();    break;
         }
 }
@Override public int onNumber()   {  return 17;  }
}
```

```
// actNative.java
package com.misoo.counter;
public class actNative {
        public static native int getCounter(int refer);
        public static native int nativeExec(int refer);
}
```

- 指令：

  actNative.nativeExec(cn.mObject);

- 这nativeExec()函数的参数refer则参考到CCounter的对象。当其执行到指令：

  ```
  CCounter *co = (CCounter*)refer;
  jobject jo = (jobject)co->javaObj;
  ```

- 就从CCounter对象里取得javaObj属性值，并存入jo变量里。

- 此jo值正是Java层CounterNative对象的参考，所以透过jo可以调用CounterNative对象的getNumb()函数，进而调用ac01的onNumber()函数，顺利取得n值(即numb值)。

- 最后，指令：

```
co->n = numb;
return (jint)co->execute();
```

- 将取到的numb值存入CCounter对象里，并调用其execute()函数算出结果，回传给Java层。

~ Continued ~