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

C06_a

JNI：本地线程进入Java层(a)

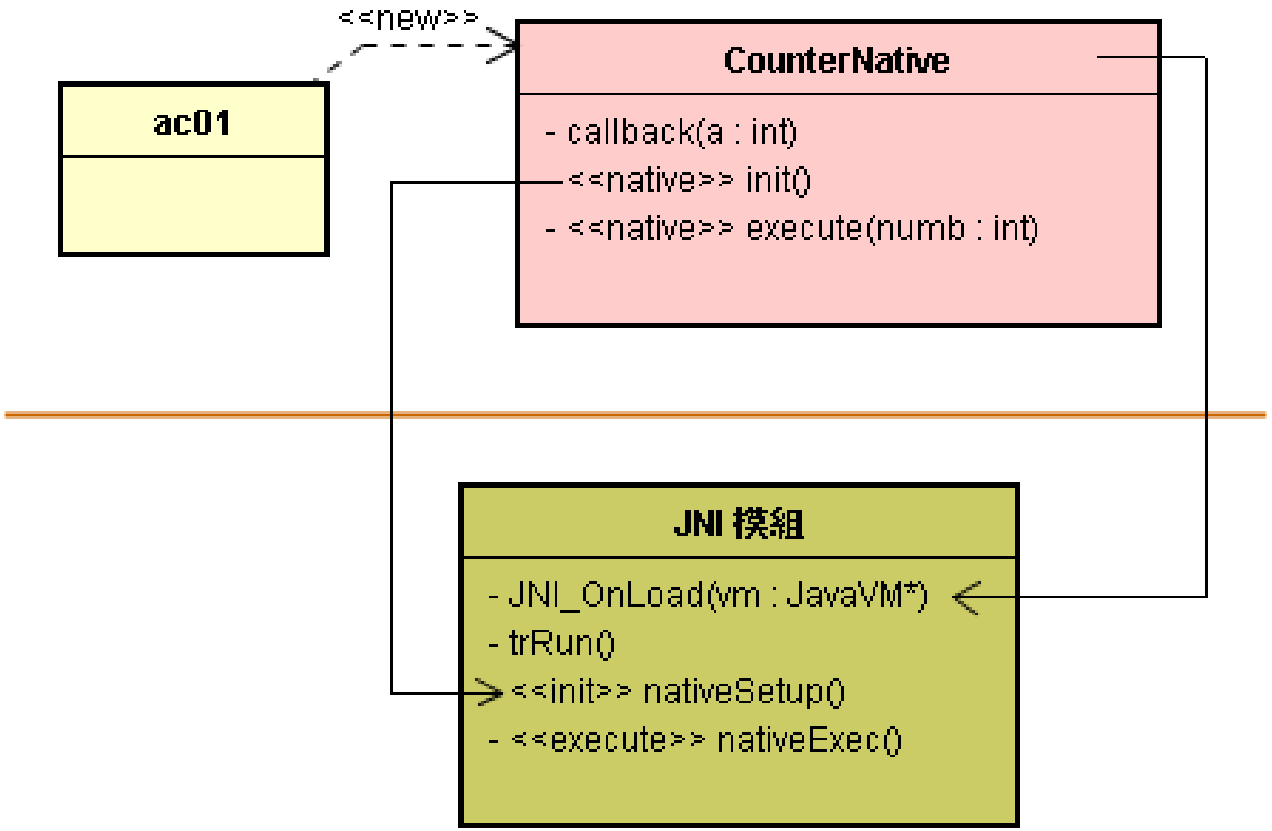
By 高煥堂

2、Native线程进入Java层

先取得JNIEnv对象

- C层新线程没有JNIEnv对象，无法调用到Java层函数。可以向VM登记而取得JNIEnv对象后，此线程就能进入Java层了。

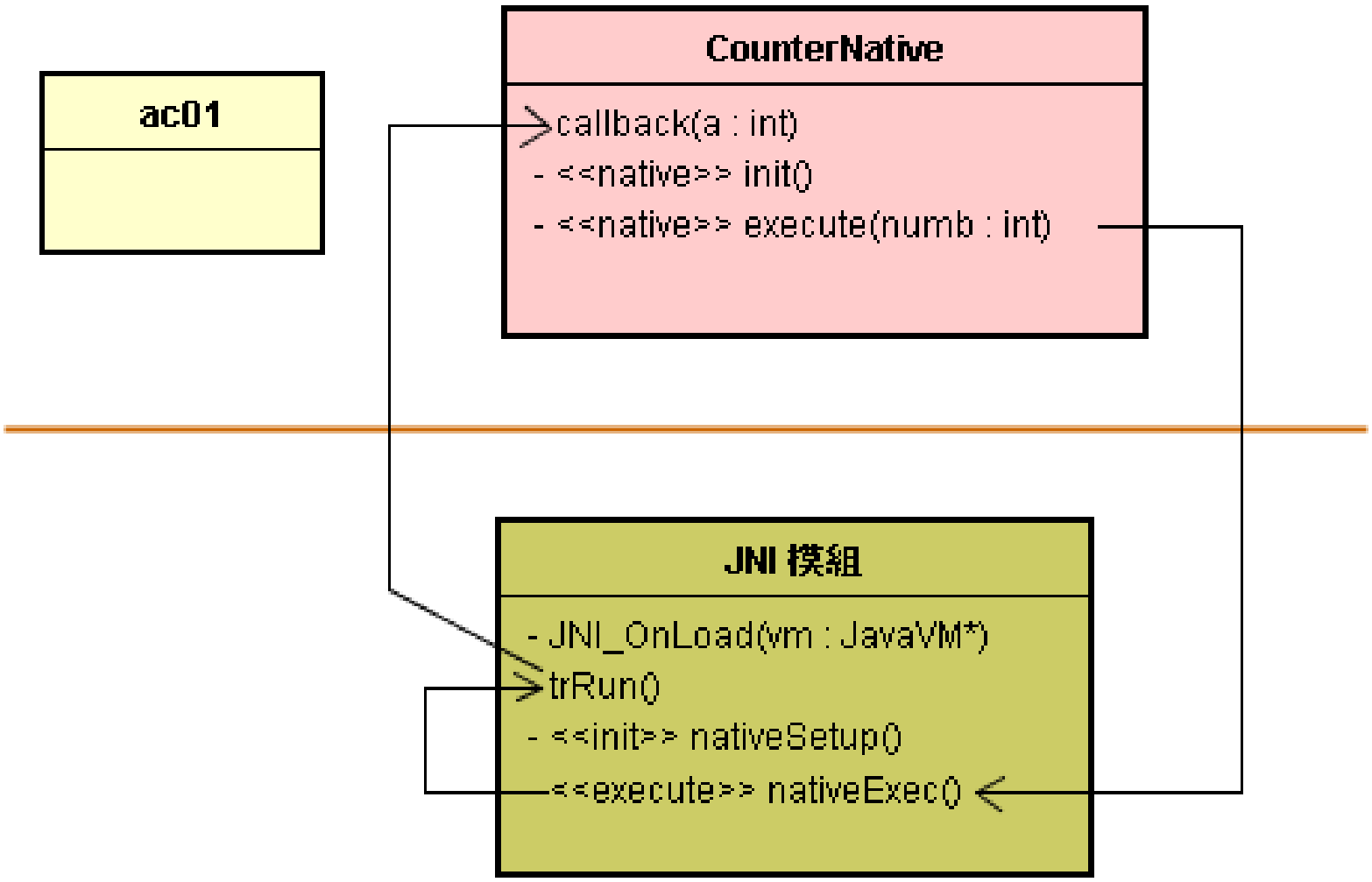
范例


<<new>>
ac01
CounterNative
- callback(a : int)
<<native>> init()
- <<native>> execute(numb : int)
JNI 模組
- JNI_OnLoad(vm : JavaVM*)
- trRun()
<<init>> nativeSetup()
- <<execute>> nativeExec()

- VM载入JNI模块时，调用JNI_OnLoad()函数，其定义init()成为nativeSetup()的别名。

别名	參數	本名
init()	()V	Java_com_misoo_counter_CounterNative_nativeSetup()
execute()	(I)V	Java_com_misoo_counter_CounterNative_nativeExec()

- 所以，Java调用init()时，会转而调用C层的nativeSetup()函数。


ac01
CounterNative
callback(a : int)
- <<native>> init()
- <<native>> execute(numb : int)
JNI 模組
- JNI_OnLoad(vm : JavaVM*)
trRun()
- <<init>> nativeSetup()
<<execute>> nativeExec()

- 也定义execute()成为nativeExec()的别名。
- 所以，Java调用execute()时，会转而调用C层的nativeExec()函数。
- 此时，nativeExec()诞生一个新线程去执行trRun()函数。
- 然后，新线程进入Java层去执行callback()函数。

- 在执行trRun()时，新线程向VM登记而取得JNIEnv对象，才能调用callback()函数，进入Java层执行了。例如，使用指令：

```
jvm->AttachCurrentThread(&env, NULL);
```

- 就向VM登记，要求VM诞生JNIEnv对象，并将其指针值存入env里。有了env值，就能调用Java层的函数了。

```
// CounterNative.java
// .........
public class CounterNative {
    private static Handler h;
    static {  System.loadLibrary("MyJT002");  }
    public CounterNative(){
            init();
            h = new Handler(){
            public void handleMessage(Message msg) {
                    ac01.ref.setTitle("Hello ...");
            }};
     }
```

```
    private static void callback(int a){
          Message m = h.obtainMessage(1, a, 3, null);
           h.sendMessage(m);
    }
   private native void init();
   public native void execute(int numb);
}
```

```
/* com.misoo.counter.CounterNative.cpp */
#include <stdio.h>
#include <pthread.h>
#include "com_misoo_counter_CounterNative.h"
jmethodID mid;
jclass mClass;
JavaVM *jvm;
pthread_t thread;
int n, sum;
void* trRun( void* );

void JNICALL
Java_com_misoo_counter_CounterNative_nativeSetup(
        JNIEnv *env, jobject thiz) {
    jclass clazz = env->GetObjectClass(thiz);
    mClass = (jclass)env->NewGlobalRef(clazz);
    mid = env->GetStaticMethodID(mClass, "callback", "(I)V");
 }
```

```
void JNICALL
Java_com_misoo_counter_CounterNative_nativeExec
(JNIEnv *env, jobject thiz, jint numb){
    n = numb;
    pthread_create( &thread, NULL, trRun, NULL);
 }
void* trRun( void* ){
     int status;
    JNIEnv *env;    bool isAttached = false;
    status = jvm->GetEnv((void **) &env, JNI_VERSION_1_4);
    if(status < 0) {
        status = jvm->AttachCurrentThread(&env, NULL);
        if(status < 0)  return NULL;
        isAttached = true;
      }
    sum = 0;
    for(int i = 0; i<=n; i++)  sum += i;
    env->CallStaticVoidMethod(mClass, mid, sum);
    if(isAttached)  jvm->DetachCurrentThread();
    return NULL;
}
```

```
static const char *classPathName =
                        "com/misoo/counter/CounterNative";
static JNINativeMethod methods[] = {
    {"init", "()V",
(void *)Java_com_misoo_counter_CounterNative_nativeSetup},
    {"execute", "(I)V",
 (void *)Java_com_misoo_counter_CounterNative_nativeExec}
};

static int registerNativeMethods(JNIEnv* env, const char*
          className, JNINativeMethod* gMethods,
          int numMethods){
       jclass clazz = env->FindClass(className);
       env->RegisterNatives(clazz, gMethods, numMethods);
       return JNI_TRUE;
}
```

```
static int registerNatives(JNIEnv* env){
    registerNativeMethods(env, classPathName,
          methods, sizeof(methods) /
          sizeof(methods[0]));
    return JNI_TRUE;
}
jint JNI_OnLoad(JavaVM* vm, void* reserved){
    JNIEnv *env;     jvm = vm;
    if (registerNatives(env) != JNI_TRUE)  return -1;
    return JNI_VERSION_1_4;
}
```

- 指令：

  pthread_create(&thread, NULL, trRun, NULL);

- 例如，当你创建一个本地C层的新线程时，可以使用指令：

  jvm->AttachCurrentThread(&env, NULL);

- 就向VM登记，要求VM诞生JNIEnv对象，并将其指针值存入env里。
- 有了env值，就能执行指令：

  env->CallStaticVoidMethod(mClass, mid, sum);
- 其调用Java层的函数了。

```
// ac01.java
// ……
public class ac01 extends Activity implements OnClickListener {
     // ……..
    @Override   protected void onCreate(Bundle icicle) {
      super.onCreate(icicle);
         ref = this;
         //……..
        obj = new CounterNative();
        }
     public void onClick(View v) {
           if(v == btn)
                    obj.execute(11);
           else if(v == btn3)
                   finish();
}}
```

- Java层主线程执行onClick()里的指令：

  obj.execute();

- 就进入C层的nativeExec()函数了。
- 此时，由这主线程诞生一个新的子线程，由子线程进入Java层的callback()里执行，将sum值带回到callback()函数里，透过Handler 而转交给主线程，然后显示出来。

~ Continued ~