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

C07_e

问题集：

进程、线程和JNI架构(e)

By 高煥堂

3、JNI基础

問題

A3.1-JNI

- Java代码在VM上执行；如下图所示。
- 在执行Java代码的过程中，如果Java需要与本地代码(如以C写成的*so动态库)沟通时，VM就会把*.so視为插件<T>而加载到VM里，然后让Java函数顺利地呼叫到这插件<T>里的C函数。

- 请问：VM在那一个时间点，会去加载所需要的插件<T>呢?

提示

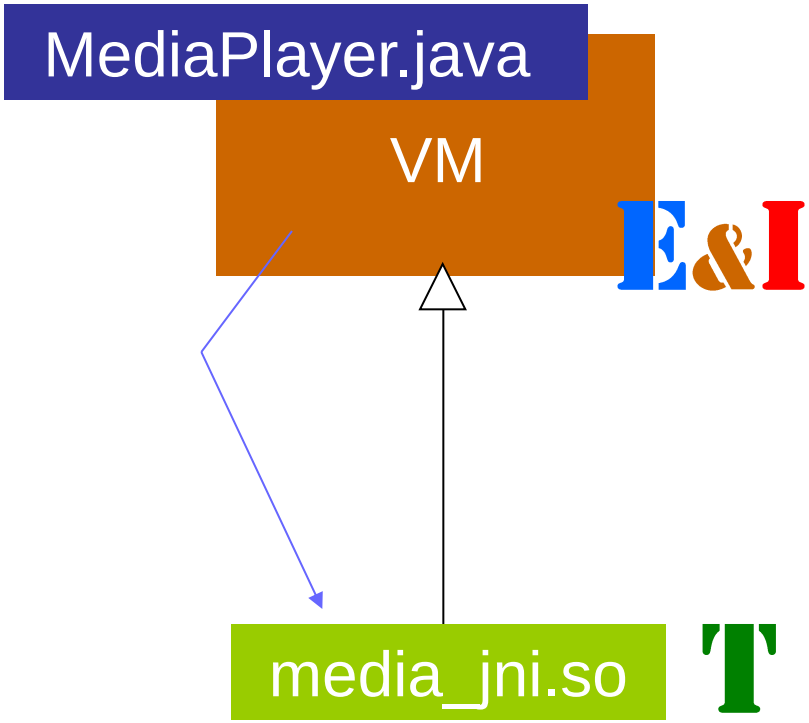
- 请参考下述的代码范例：

例如，MediaPlayer.java类：

```
public class MediaPlayer{
  static {
     System.loadLibrary("media_jni");
  }
 ……
}
```

相关问题

- 此时，VM扮演着<E&I>的角色；Java扮演Client角色；而C函数则扮演<T>角色。载入插件(*.so)之后，如下图所示。
- 请问：载入后，VM会先调用<T>里的那一个函数呢? 其目的是什么?


MediaPlayer.java
VM
E&I
media_jni.so
T

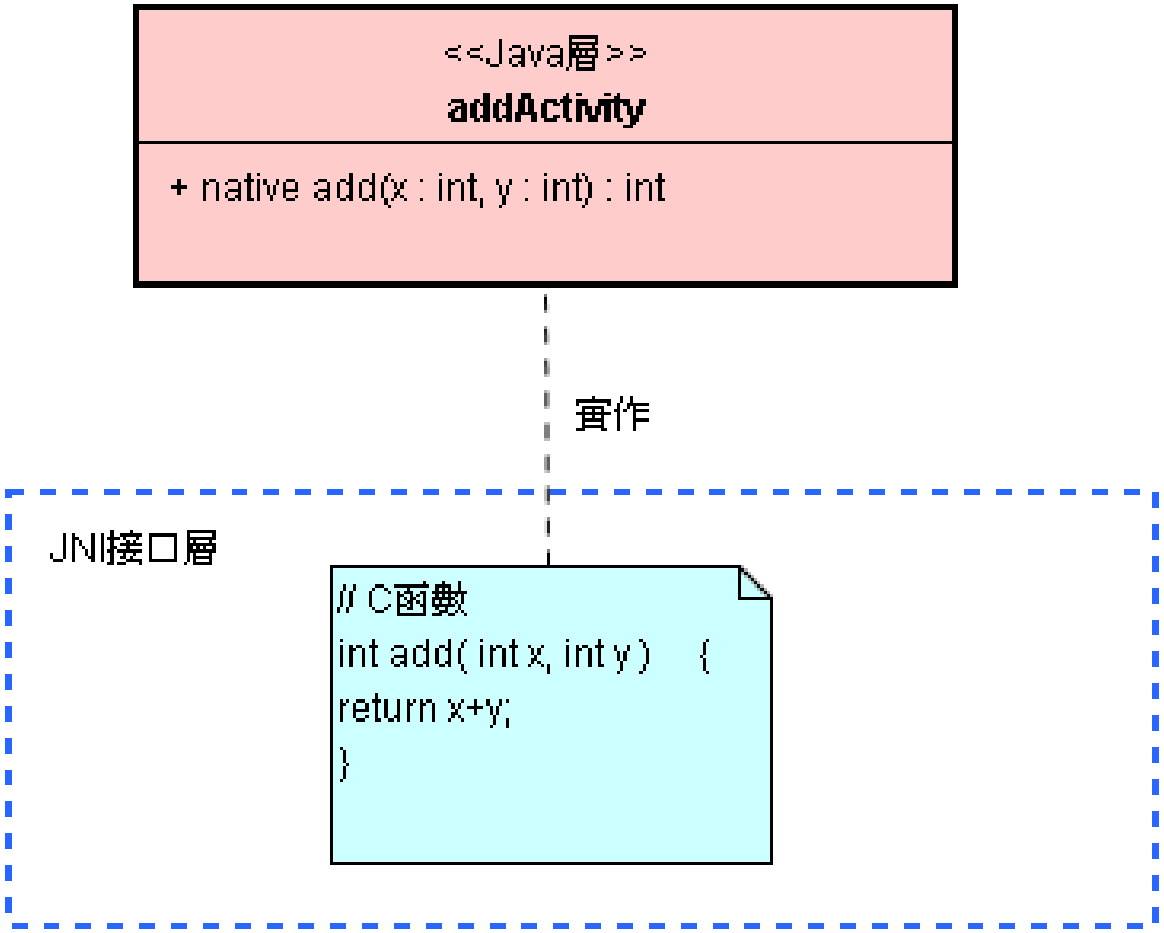
- VM载入JNI模块时，调用JNI_OnLoad()函数。

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

問題

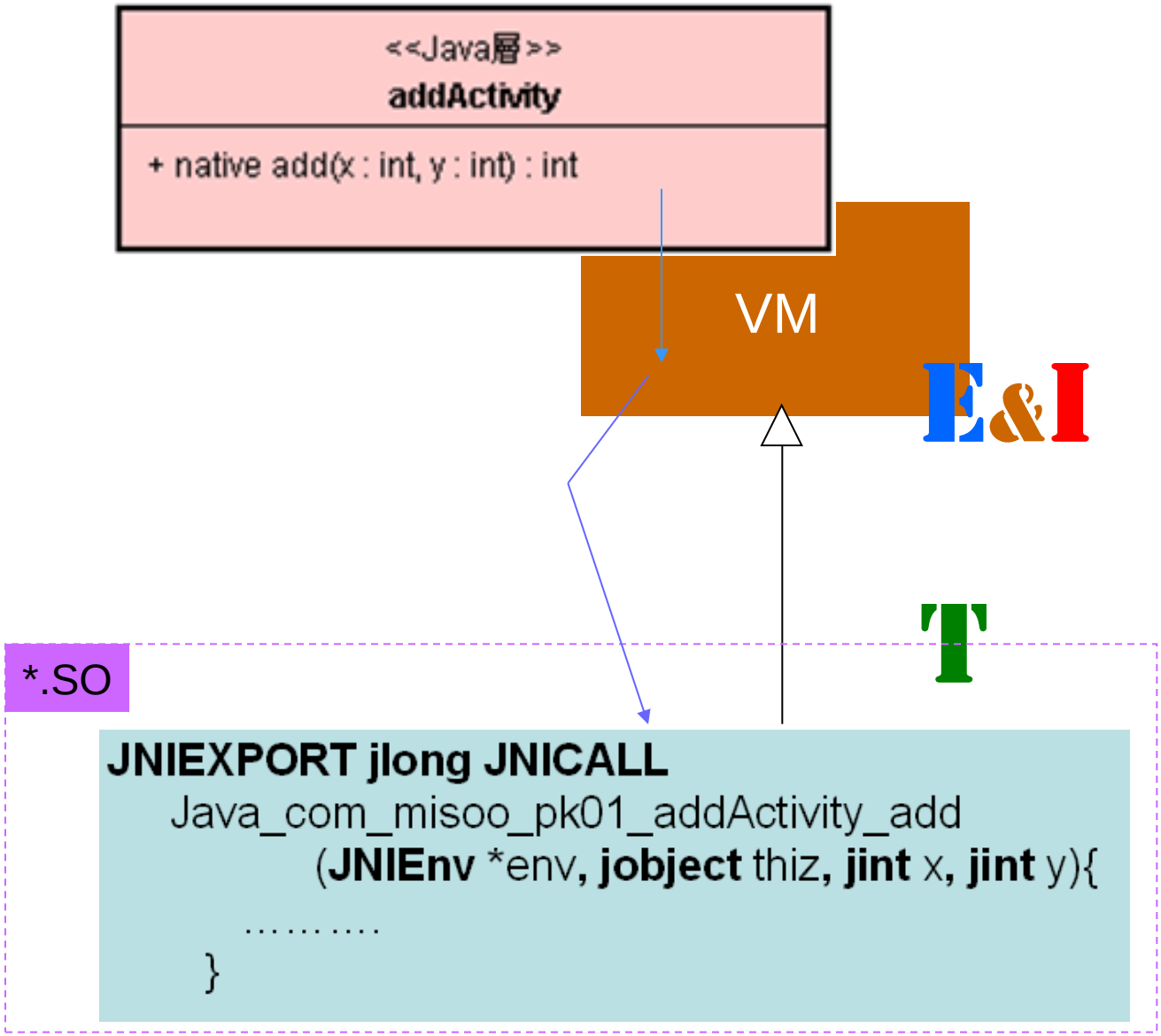
A3.2-JNI

- addActivity是一个完整的Java类，其add()函数里有完整的实作(Implement)代码。如果从这Java类里移除掉add()函数里的实作代码，而以C语言来实作之。如下图所示。
- 请问：为什么Java与C函数不能直接互相调用呢?


<<Java層>>
addActivity
+ native add(x : int, y : int) : int
實作
JNI接口層
// C函數
int add(int x, int y) {
return x+y;
}

提示

- 这add()函数所构成的*.so成为VM(即<E&I>的插件<T>；如下图所示。
- Java函数透过<E&I>来调用<T>，并不直接调用<T>(即*.so)里的本地add()函数。
- 请你说说其幕后的理由吧。


<<Java層>>
addActivity
+ native add(x : int, y : int) : int
VM
E&I
T
*.SO
JNIEXPORT jlong JNICALL
Java_com_misoo_pk01_addActivity_add
(JNIEnv *env, jobject thiz, jint x, jint y){
.....
}

相关问题

- 在执行Java程序的过程中，什么时间点才会要求VM去调用本地的 add()函数呢?

```
class addActivity extends Activity {
    void onCreate( … ){
            // ……
            this.add( 2, 4 );
     }
     int native add(int x, int y);
}
```

- 如果在Java里将add()函数定义为static函数，则上述的答案会有何不同呢?

```
class addActivity extends Activity {
    // ……
    int static native add(int x, int y);
}
```

```
class myActivity extends Activity {
    // ……
    addActivity.add(200, 300);
    // ……
}
```

問題

A3.3-JNI

- 接续上一个题目，本地函数add()的第1个参数是：JNIEnv *env；如下页的代码所示。
- 请问，这个JNIEnv类的内涵是什么? 这个env指针(Pointer)有什么用途呢?

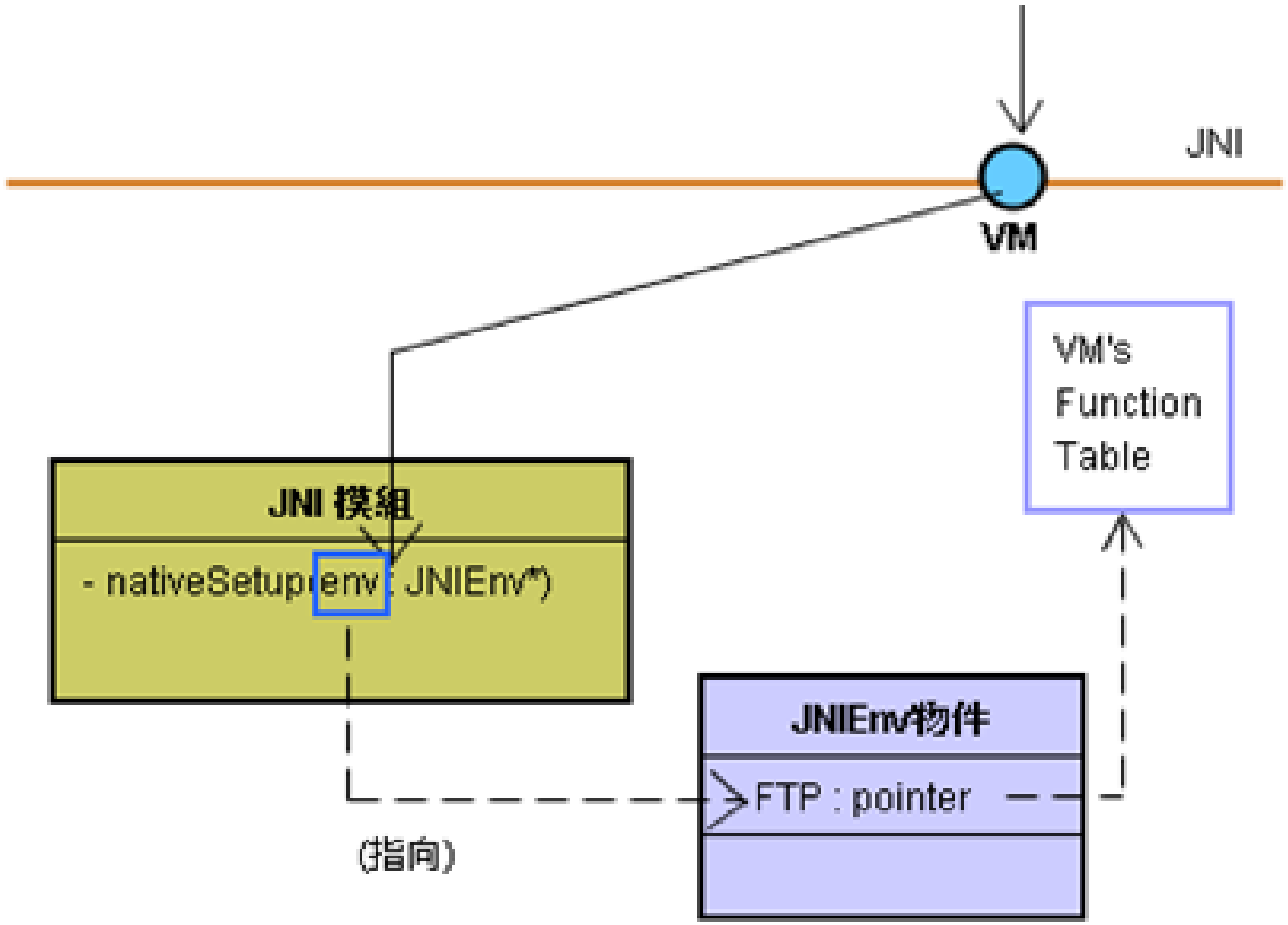
本地的add()函数的参数型态为：

```
// com_misoo_pk01_addActivity.cpp
……
JNIEXPORT jlong JNICALL
    Java_com_misoo_pk01_addActivity_add
            (JNIEnv *env, jobject thiz, jint x, jint y){
        // …….
    }
// ……
```

提示

- 在Android环境里，每一个线程(Thread)第一次进入VM(即<E&I>)去调用本地函数时，VM会替它诞生一个相对映的JNIEnv对象。如下图所示。
- 所以一个线程每次调用本地函数时，都会将其对映的JNIEnv对象指针值传递给本地函数。

線程thread


JNI
VM
VM's
Function
Table
JNI 模組
- nativeSetup(env : JNIEnv*)
JNIEnv物件
FTP : pointer
(指向)

~ Continued ~