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

C02_a

认识JNI开发与NDK (a)

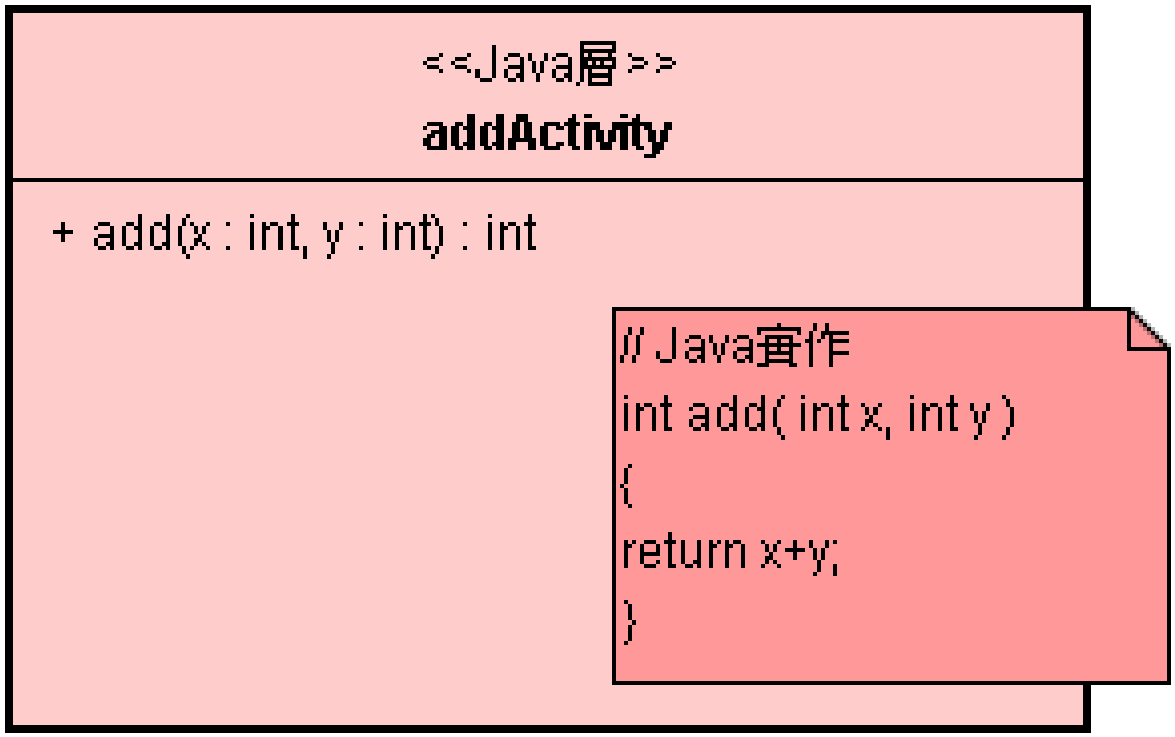
By 高煥堂

内容

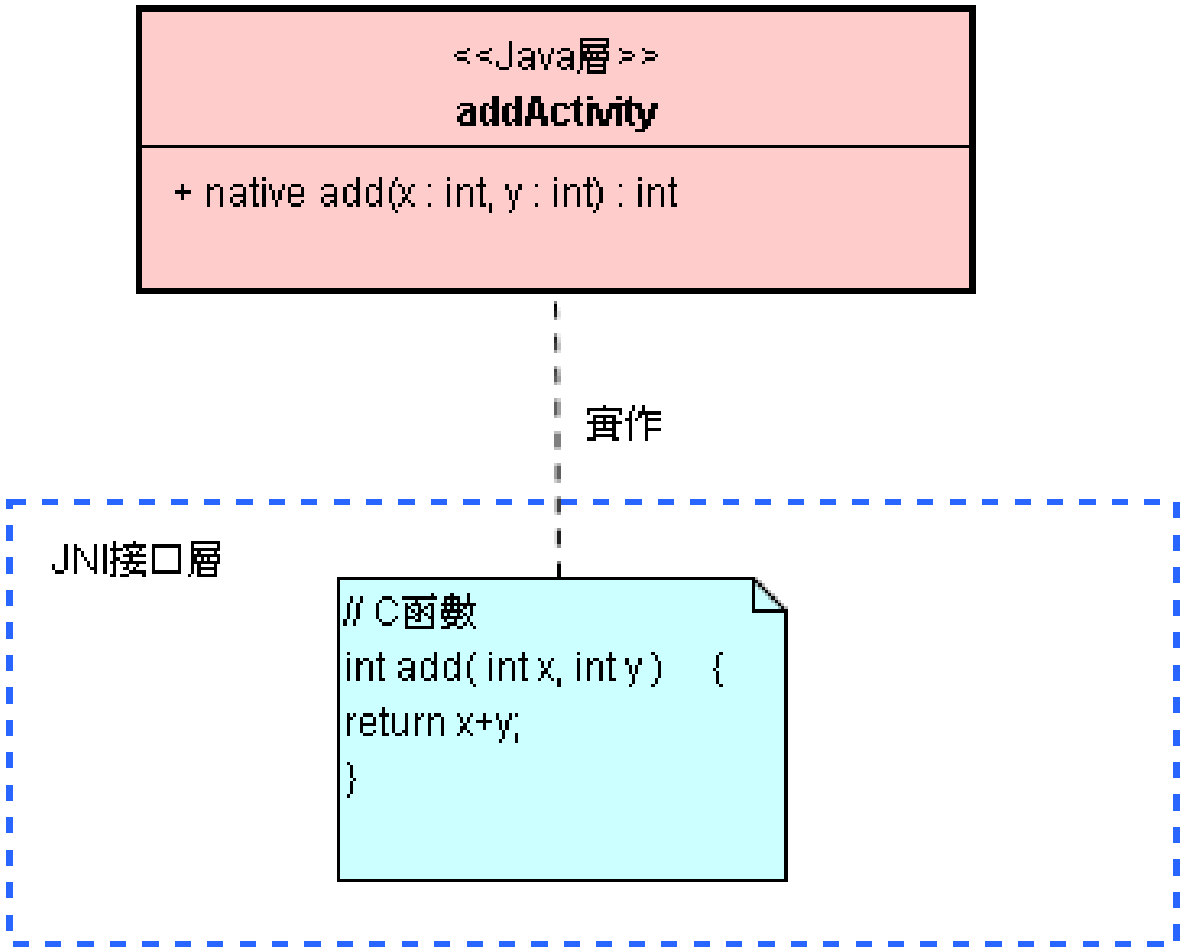
1. JNI基本概念
2. 使用Android NDK
3. 如何载入*.so档案
4. *.so的入口函数：JNI_OnLoad()

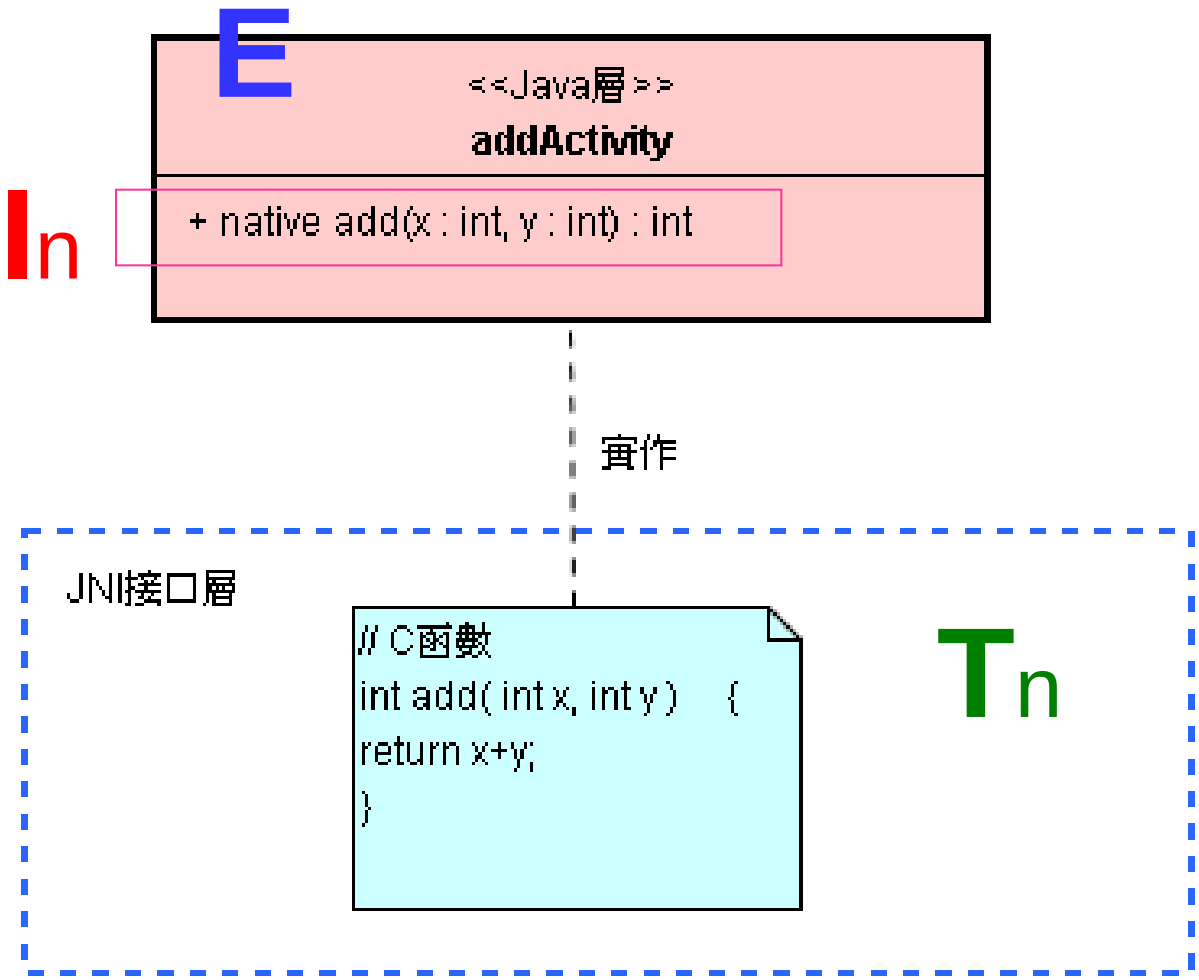
1、JNI基本概念

- 在Androd框架里，上层是Java框架，而下层是C/C++框架。这两层框架之间会有密切的沟通。此时JNI(Java Native Interface)就扮演双方沟通的接口了。
- 藉由JNI接口，可将Java层的基类或子类的函数实作部份挖空，而移到JNI层的C函数来实作之。例如，原来在Java层有个完整的Java类：

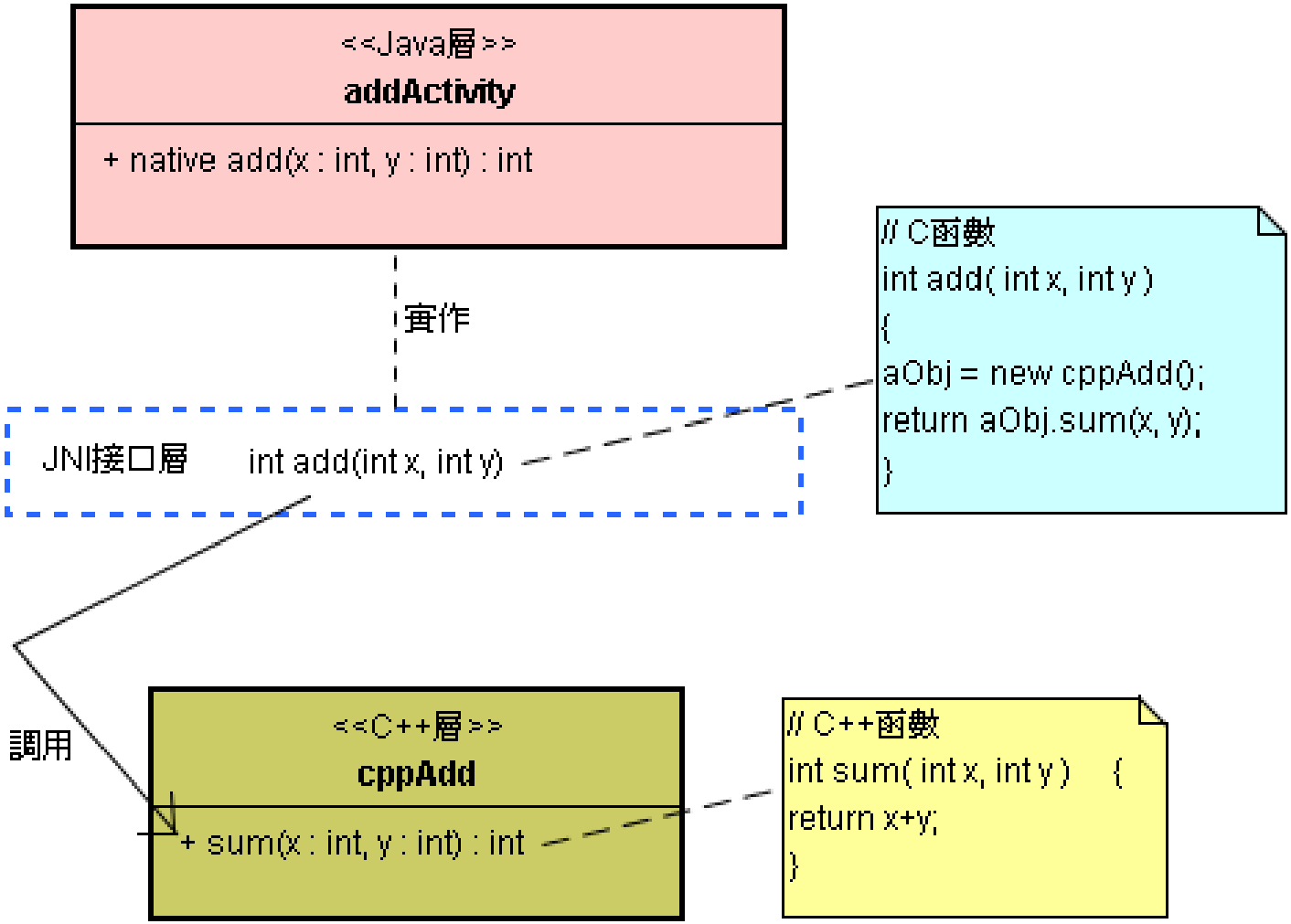

<<Java層>>
addActivity
+ add(x : int, y : int) : int
// Java實作
int add(int x, int y)
{
return x+y;
}

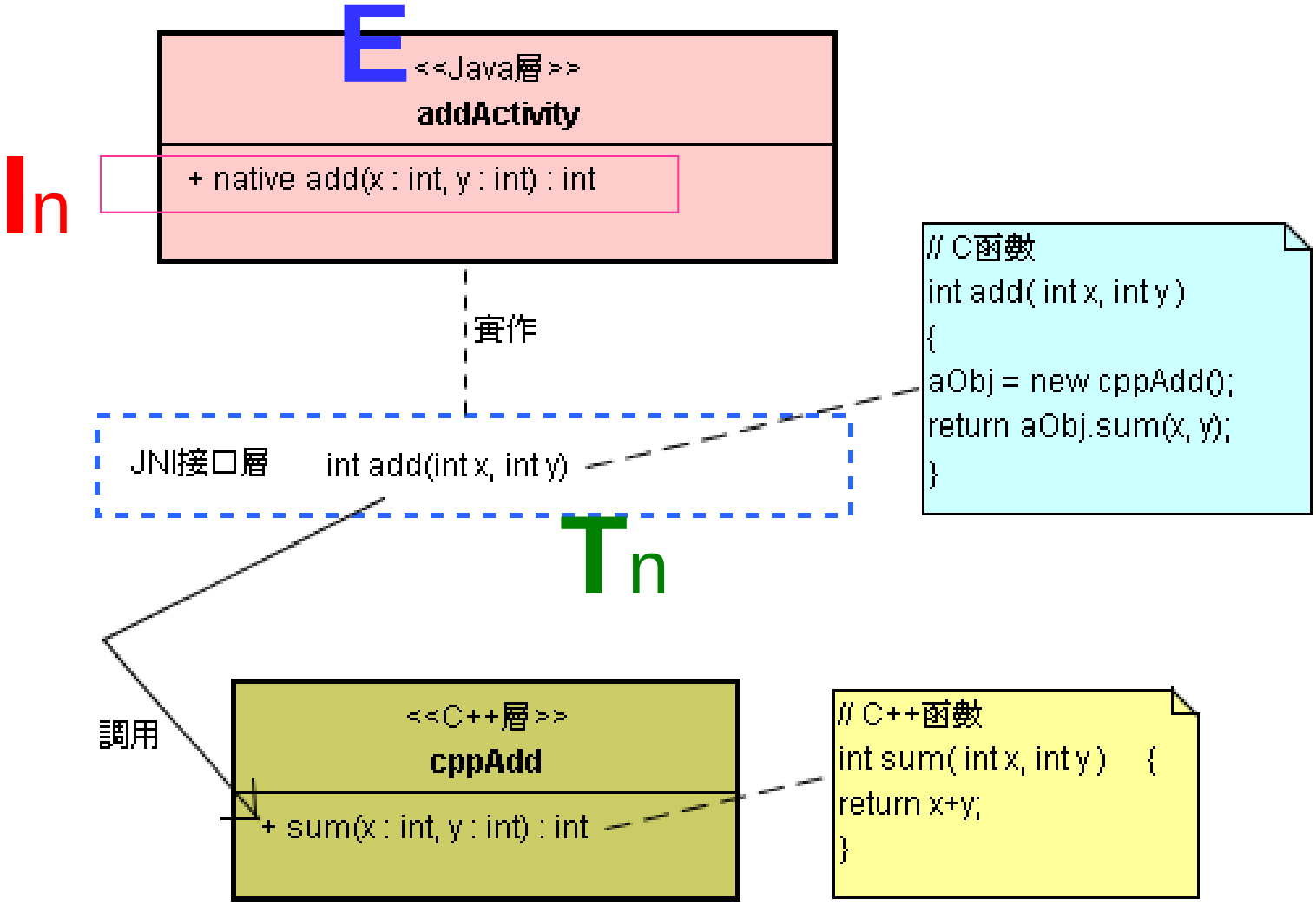
- 这是一个完整的Java类，其add()函数里有完整的实作(Implement)代码。如果从这Java类里移除掉add()函数里的实作代码(就如同抽象类里的抽象函数一般)，而成为本地(Native)函数；然后依循JNI接口协议而以C语言来实作之。如下图所示：

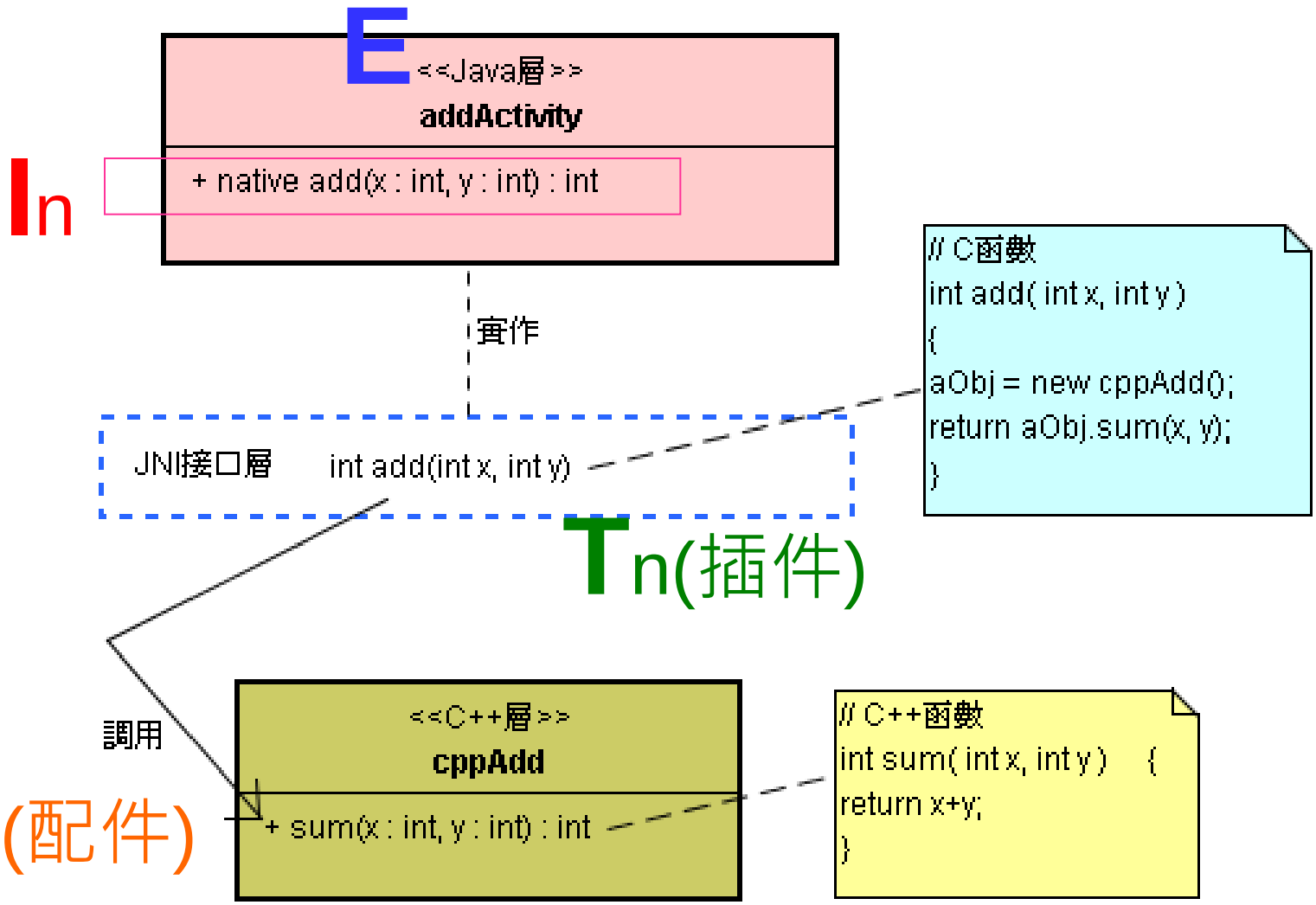

<<Java層>>
addActivity
+ native add(x : int, y : int) : int
實作
JNI接口層
// C函數
int add(int x, int y) {
return x+y;
}

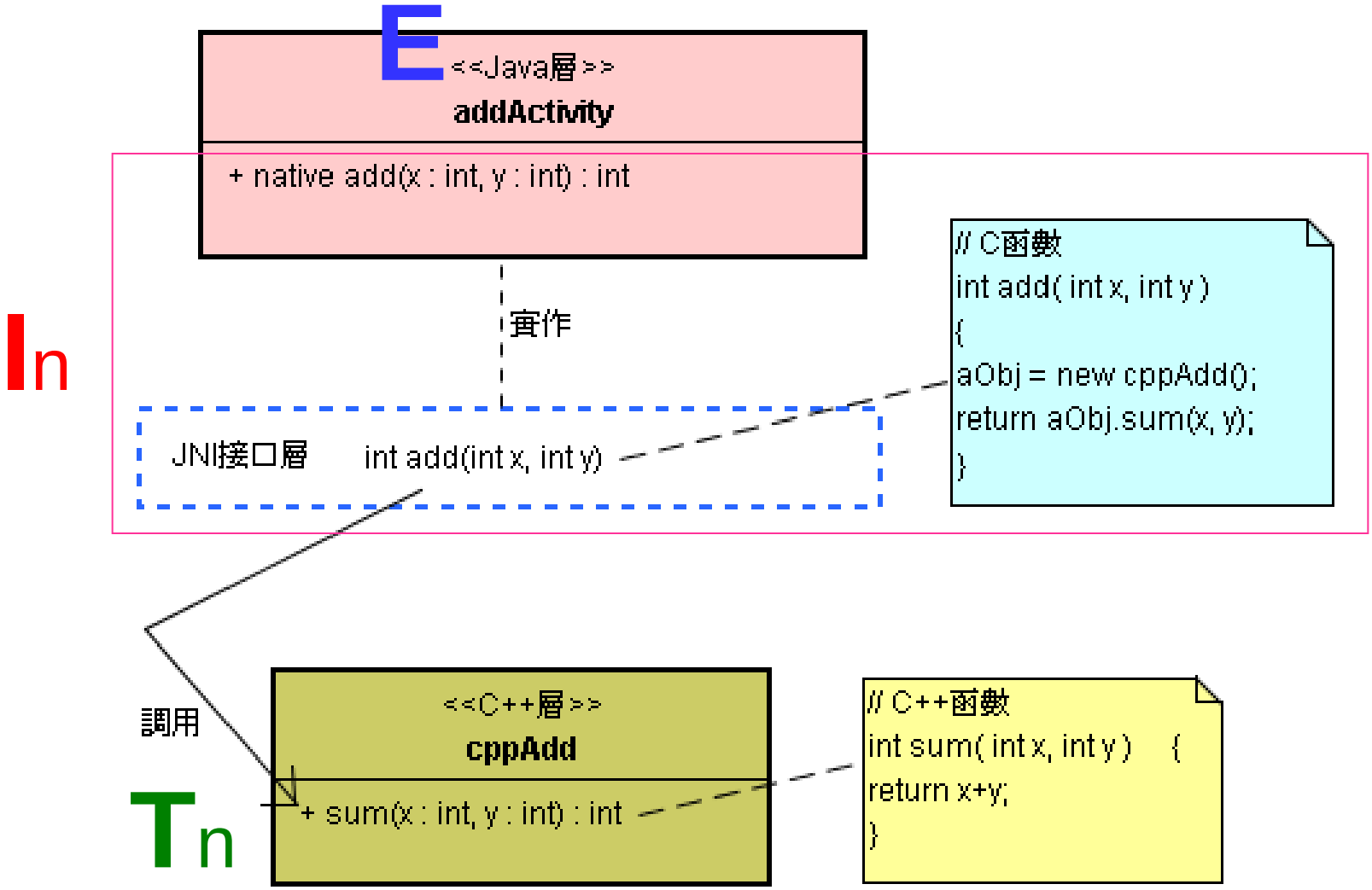

E
<<Java層>>
addActivity
In
+ native add(x : int, y : int) : int
實作
JNI接口層
// C函數
int add(int x, int y) {
return x+y;
}
Tn

- 这个add()函数仍然是Java类的一部分，只是它是用C语言来实作而已。为什么要将Java类的add()函数挖空呢? 其主要的理由是：Java代码执行速度较慢，而C代码执行速度快。然而Java代码可以跨平台，而C代码与本地平台设备息息相关，所以称之为本地(Native)代码。
- 在本地的C代码里，可以创建C++类的对象，并调用其函数。如下图：


<<Java層>>
addActivity
+ native add(x : int, y : int) : int
實作
// C函數
int add(int x, int y)
{
aObj = new cppAdd();
return aObj.sum(x, y);
}
JNI接口層
int add(int x, int y)
調用
<<C++層>>
cppAdd
+ sum(x : int, y : int) : int
// C++函數
int sum(int x, int y) {
return x+y;
}

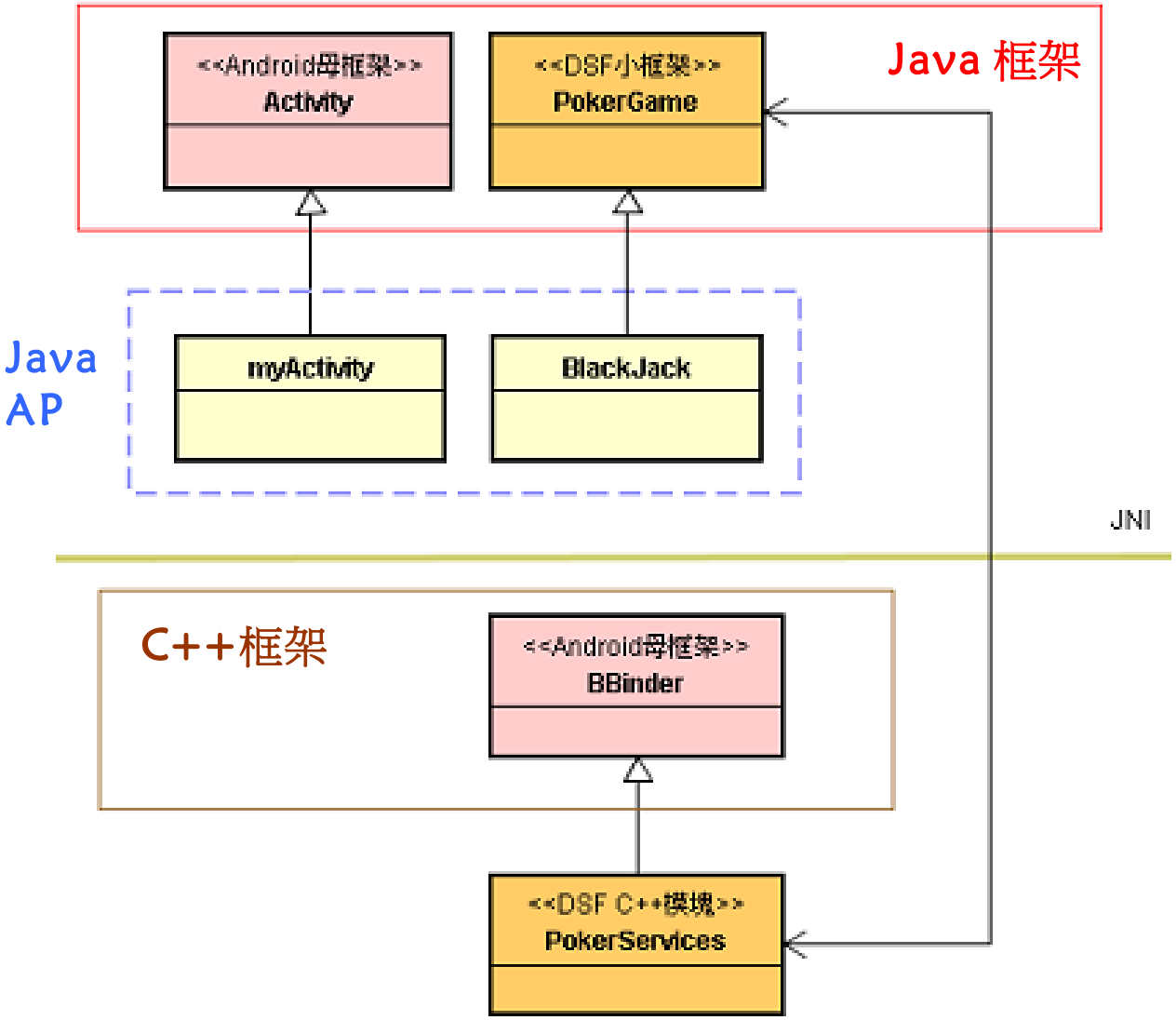

E
<<Java層>>
addActivity
In
+ native add(x : int, y : int) : int
實作
// C函數
int add(int x, int y)
{
aObj = new cppAdd();
return aObj.sum(x, y);
}
JNI接口層
int add(int x, int y)
Tn
調用
<<C++層>>
cppAdd
+ sum(x : int, y : int) : int
// C++函數
int sum(int x, int y) {
return x+y;
}


E
<<Java層>>
addActivity
In
+ native add(x : int, y : int) : int
實作
// C函數
int add(int x, int y)
{
aObj = new cppAdd();
return aObj.sum(x, y);
}
JNI接口層
int add(int x, int y)
Tn(插件)
調用
(配件)
<<C++層>>
cppAdd
+ sum(x : int, y : int) : int
// C++函數
int sum(int x, int y) {
return x+y;
}

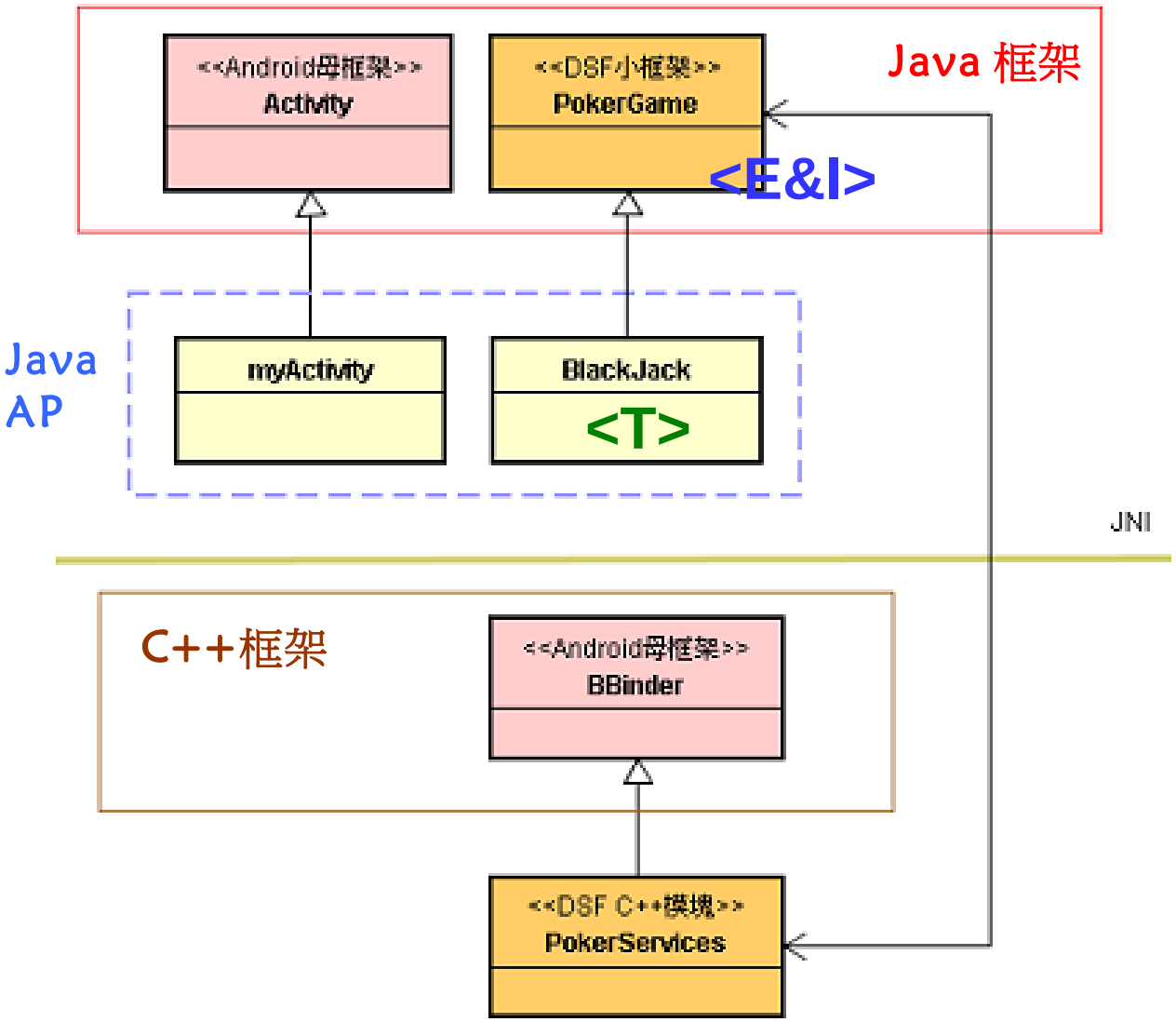

E
<<Java層>>
addActivity
+ native add(x : int, y : int) : int
實作
In
JNI接口層
int add(int x, int y)
// C函數
int add(int x, int y)
{
aObj = new cppAdd();
return aObj.sum(x, y);
}
調用
Tn
<<C++層>>
cppAdd
+ sum(x : int, y : int) : int
// C++函數
int sum(int x, int y) {
return x+y;
}

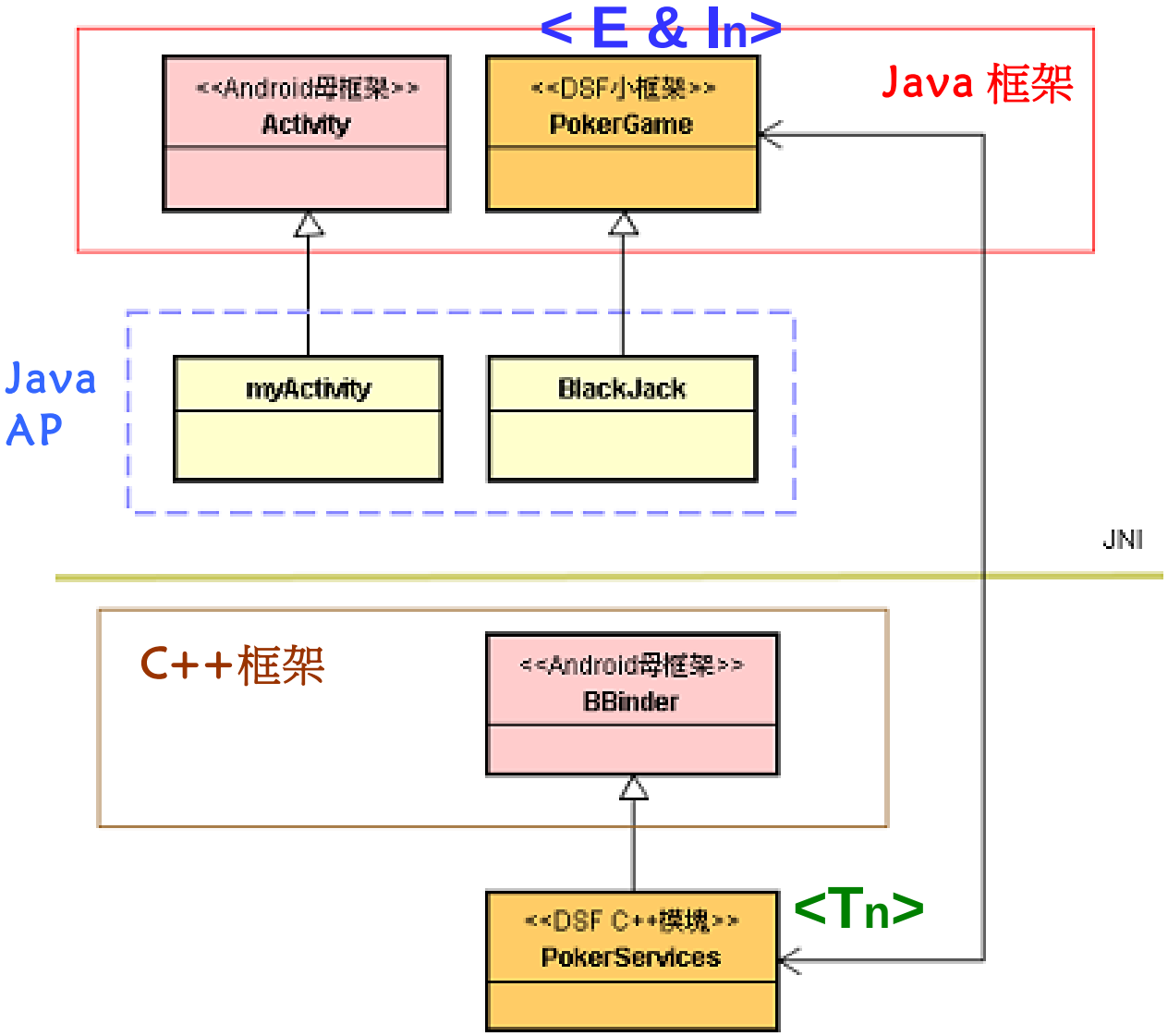
- 藉由JNI接口，就能让Java类与C++类互相沟通起来了。这也是Android双层框架的重要基础机制。如下图所示：

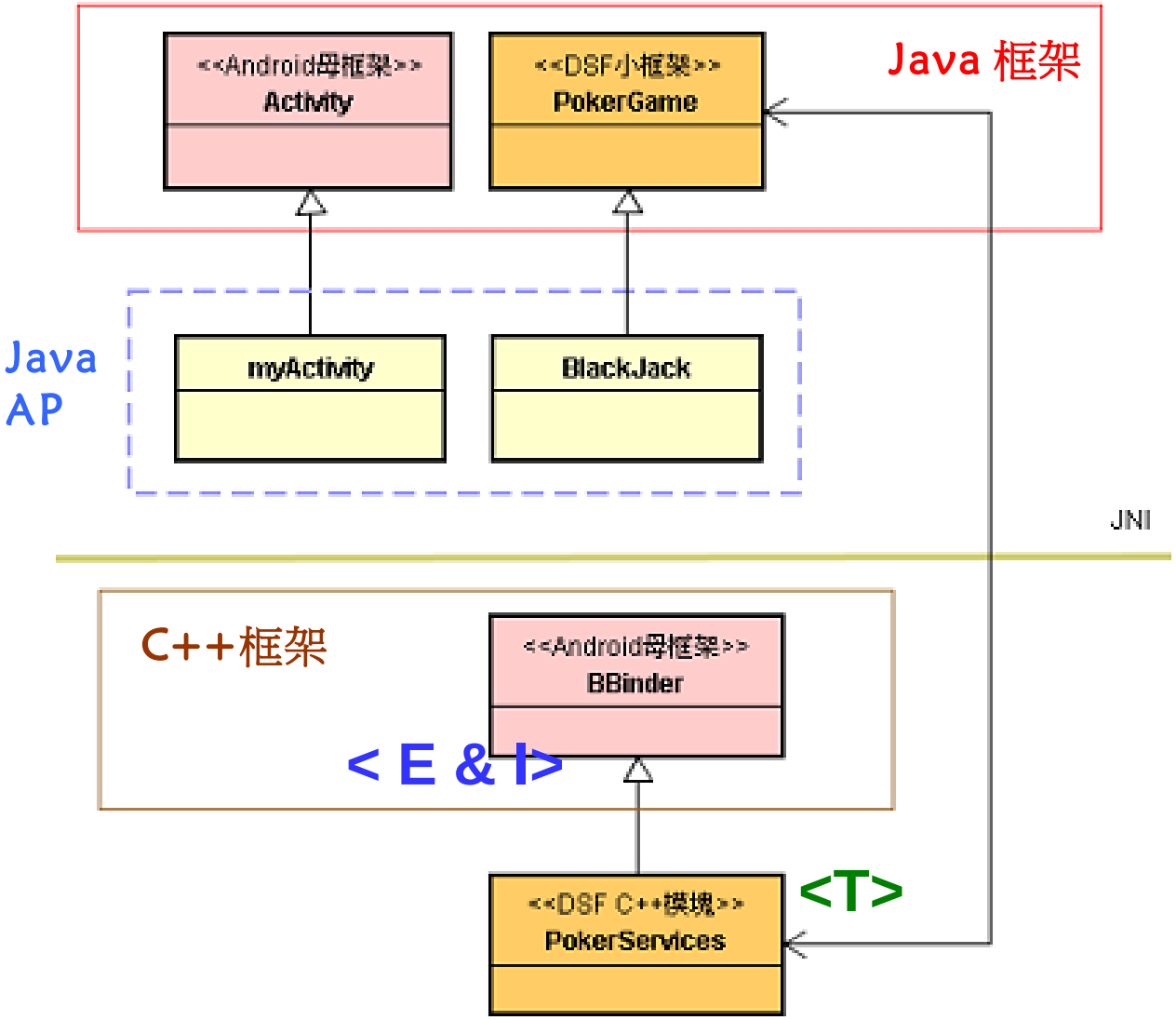
JNI接口是Android的
双层框架幕后的重要支柱


Java 框架
<<Android母框架>>
Activity
<<DSF小框架>>
PokerGame
Java
AP
myActivity
BlackJack
JNI
C++框架
<<Android母框架>>
BBinder
<<DSF C++模塊>>
PokerServices

- 从上述各图看来，只看到上层的Java函数调用中间JNI层的C函数，再往下调用C++层的函数。然而，在Android 环境里，从C/C++层函数反过来调用Java层函数，反而是更关键性的机制。
- 所以，我们更需要关注于从C/C++层调用Java层函数的方法和技术。


Java 框架
<<Android母框架>>
Activity
<<DSF小框架>>
PokerGame
<E&I>
Java
AP
myActivity
BlackJack
<T>
JNI
C++框架
<<Android母框架>>
BBinder
<<DSF C++模塊>>
PokerServices


< E & In>
Java 框架
<<Android母框架>>
Activity
<<DSF小框架>>
PokerGame
Java
AP
myActivity
BlackJack
JNI
C++框架
<<Android母框架>>
BBinder
<<DSF C++模塊>>
PokerServices
<Tn>


Java 框架
<<Android母框架>>
Activity
<<DSF小框架>>
PokerGame
Java
AP
myActivity
BlackJack
JNI
C++框架
<<Android母框架>>
BBinder
< E & I>
<<DSF C++模塊>>
PokerServices
<T>

~ Continued ~