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）

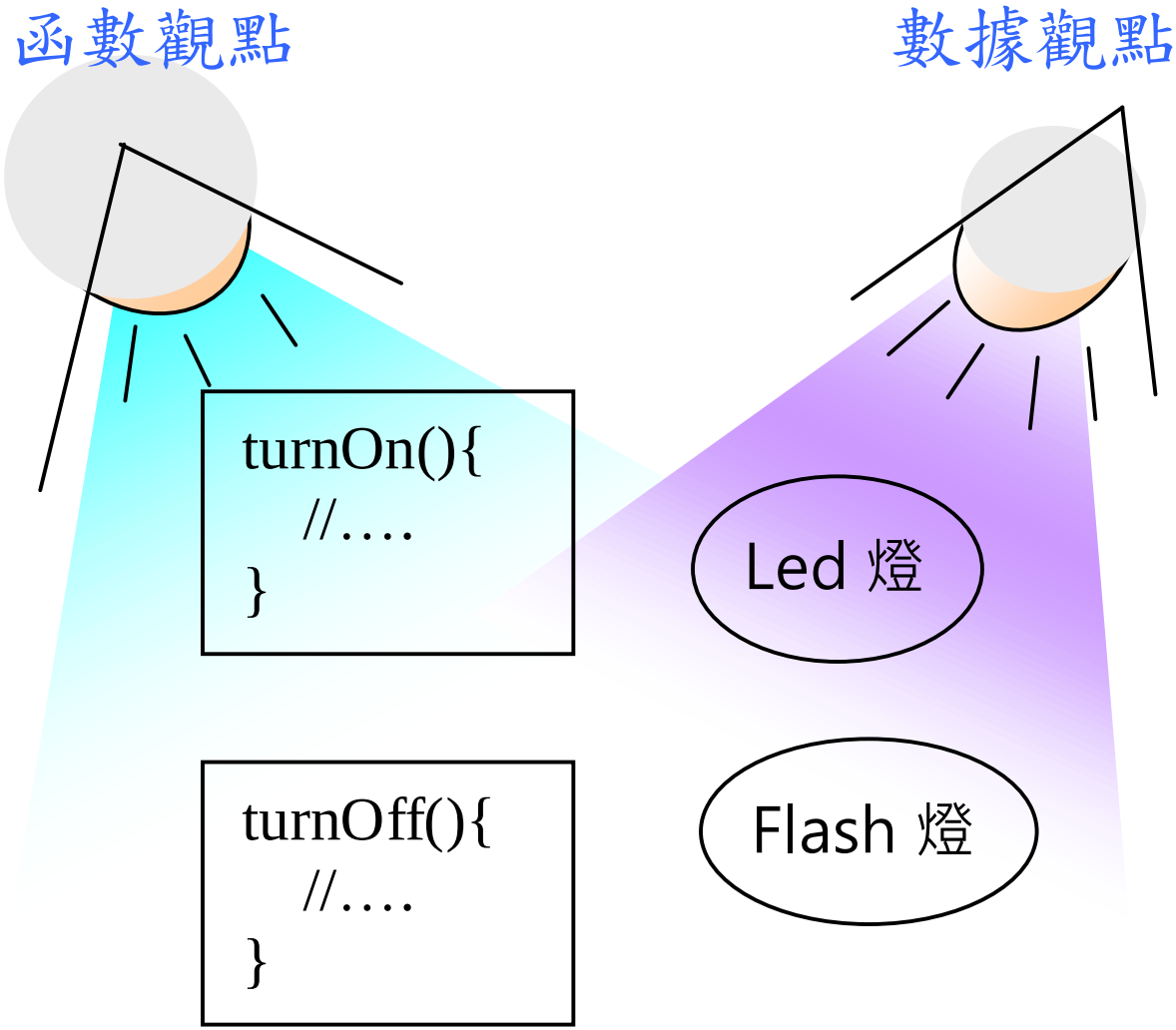
http://www.microoh.com

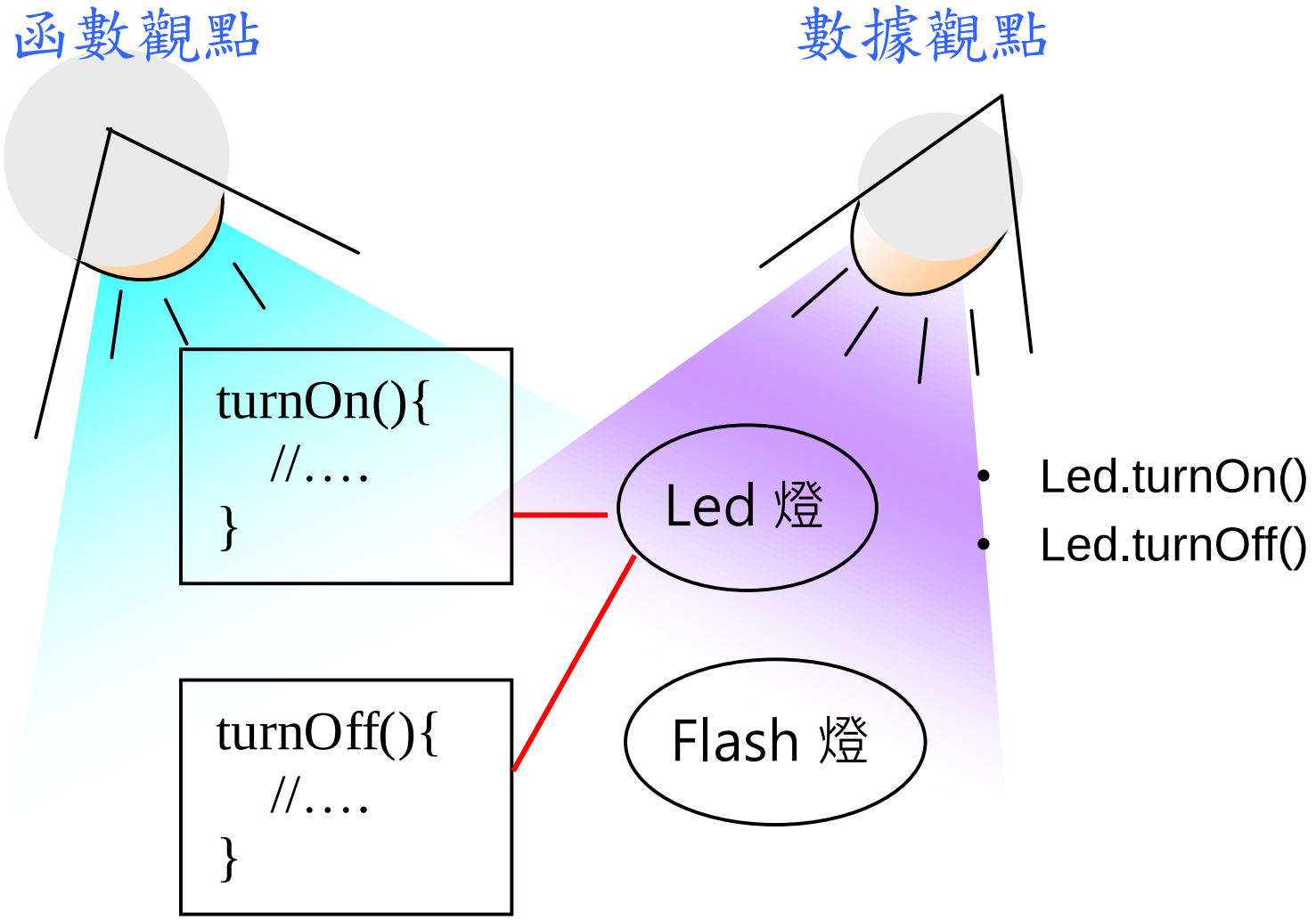
E02_b

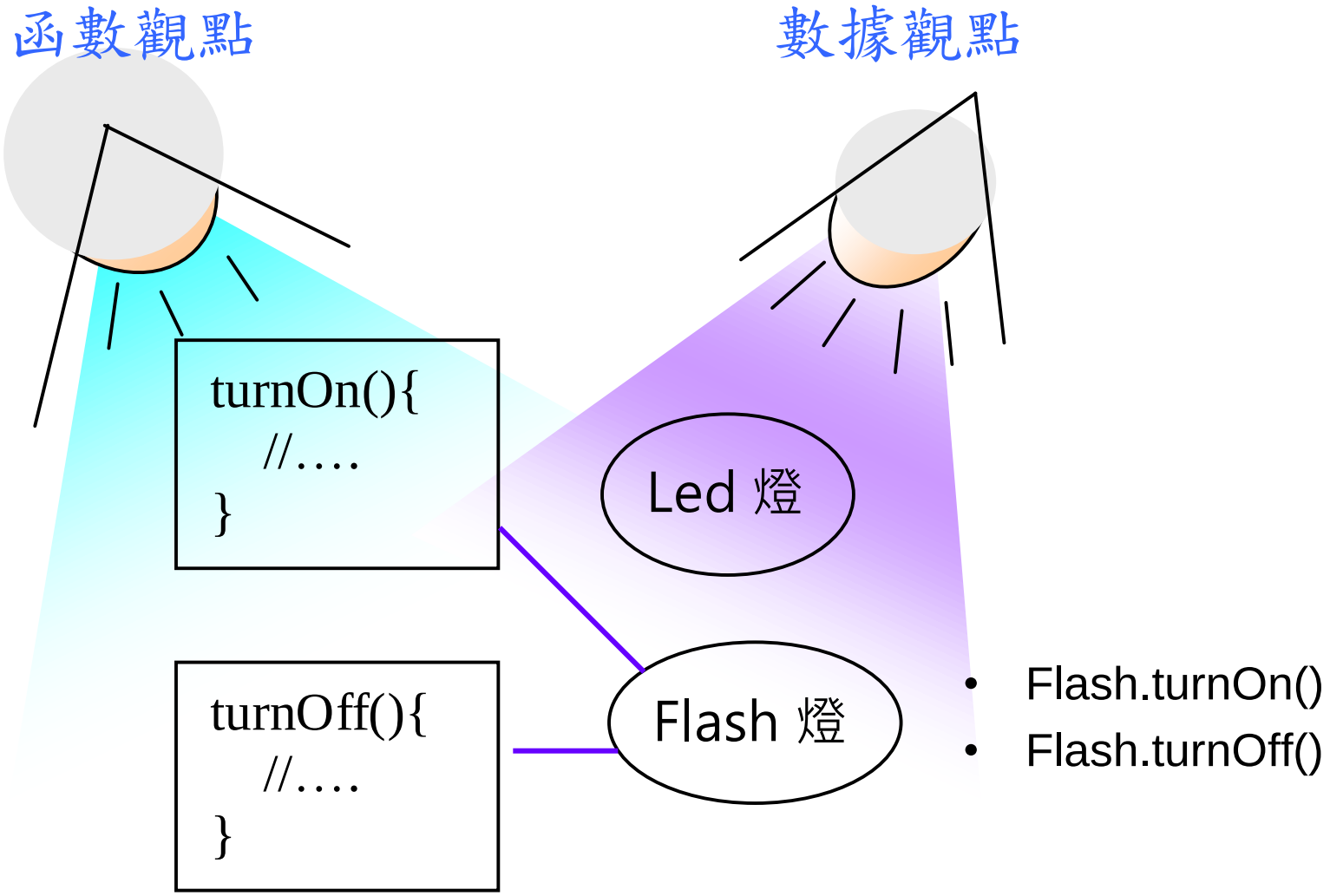
HAL框架与Stub开发 (b)

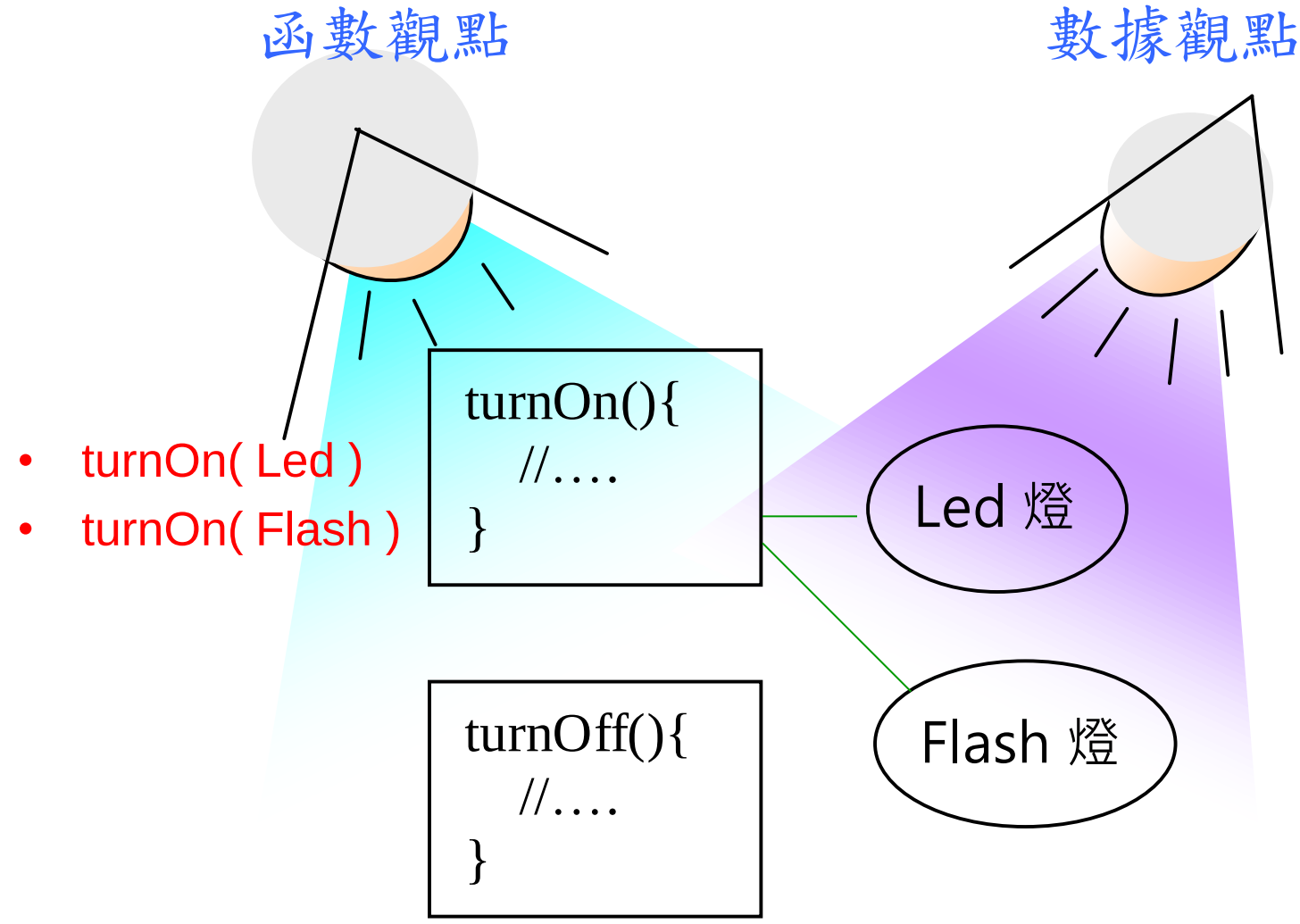
By 高煥堂

前言

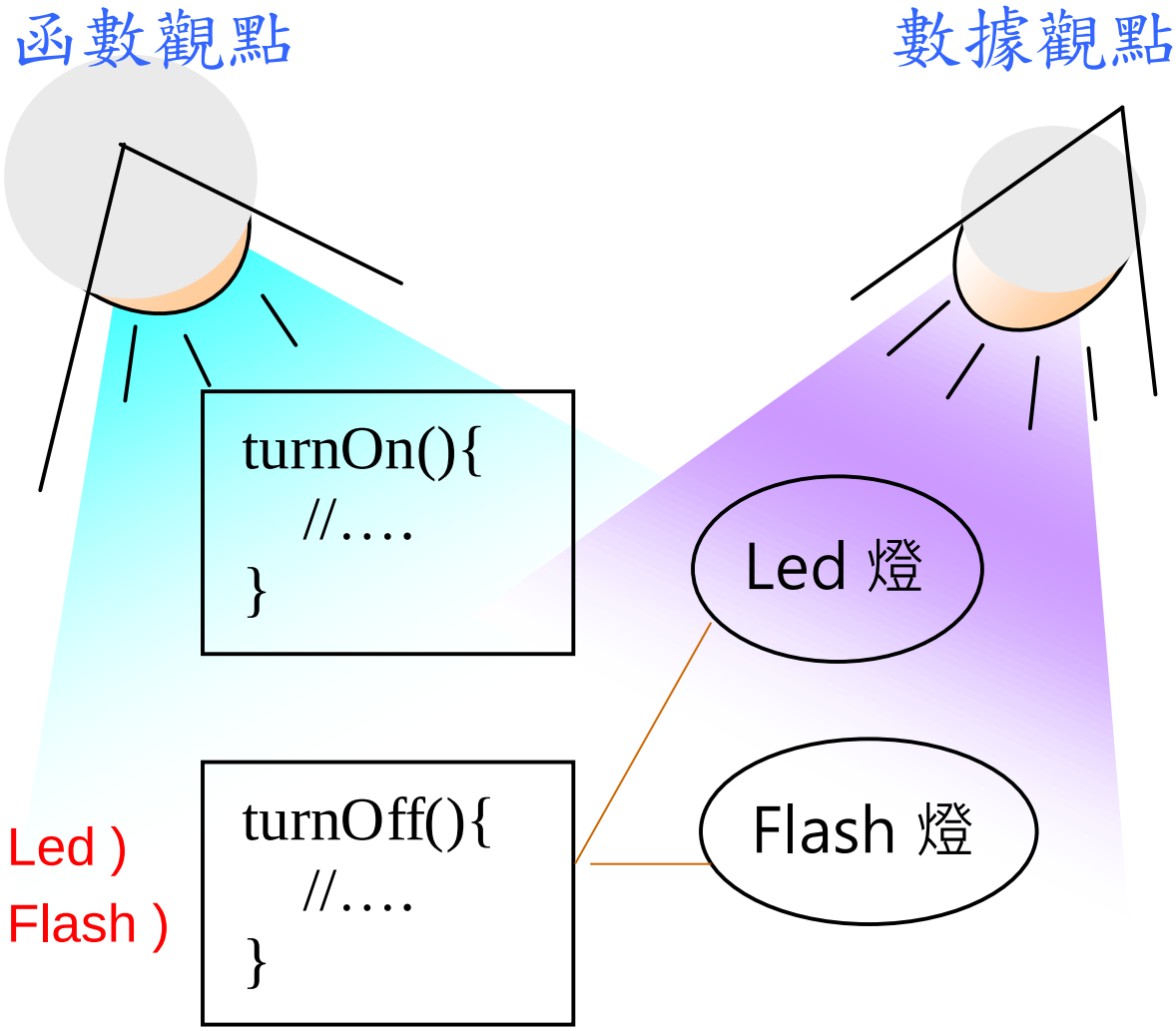

函數觀點
數據觀點
turnOn(){
//….
}
turnOff(){
//….
}
Led 燈
Flash 燈


函數觀點
數據觀點
turnOn(){
//….
}
turnOff(){
//….
}
Led 燈
Flash 燈
Led.turnOn()
Led.turnOff()


函數觀點
數據觀點
turnOn(){
//….
}
turnOff(){
//….
}
Led 燈
Flash 燈
Flash.turnOn()
Flash.turnOff()



- turnOn(Led)
- turnOn(Flash)



- turnOff(Led)
- turnOff(Flash)

1.4 以C结构表达类(class)，并创建对象(object)

- 目的：要了解Java对象如何与C函数对接?
- 途径：先了解C对象如何与C函数对接呢?

认识C函数指针

- struct里不能定义函数本身，但能定义函数指针(function pointer)属性。

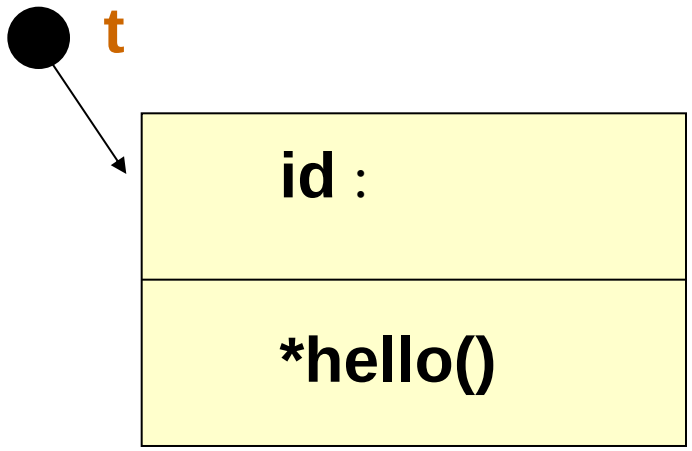
```
typedef struct cc {
    int id;
    void (*hello)();
} CC;
```

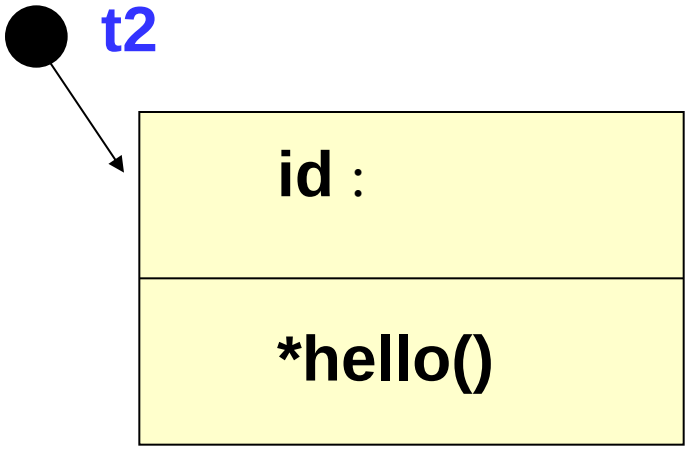
这个hello就是一个函数指针属性了。

```
static void my_hello() {
        printf("Hello");
 }
```

```
typedef struct cc {
            int id;
            void (*hello)();
          } CC;
```

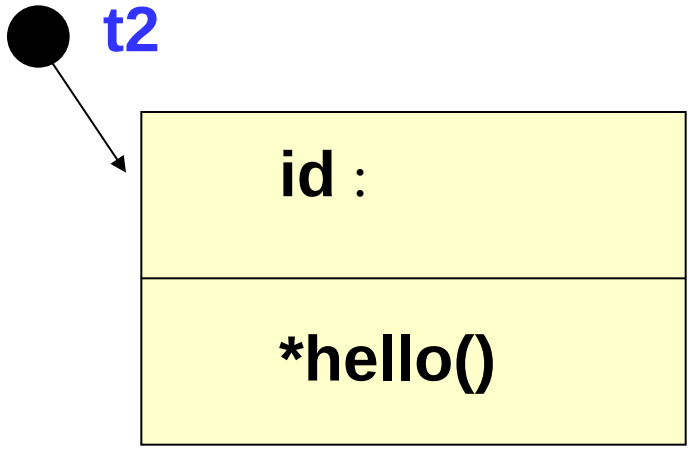
```
static void my_hello() {
        printf("Hello");
 }
```

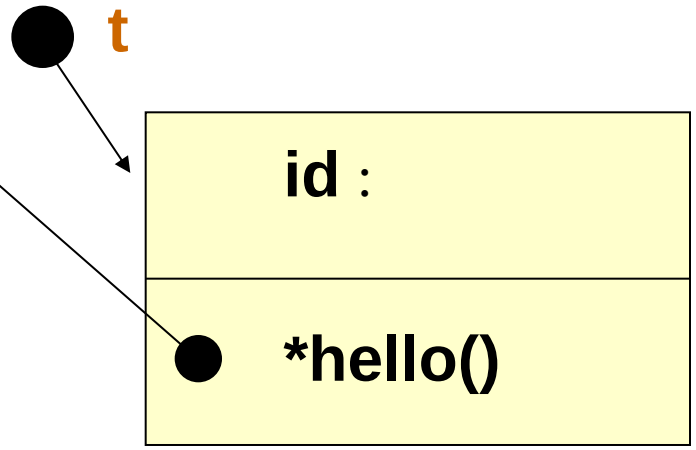




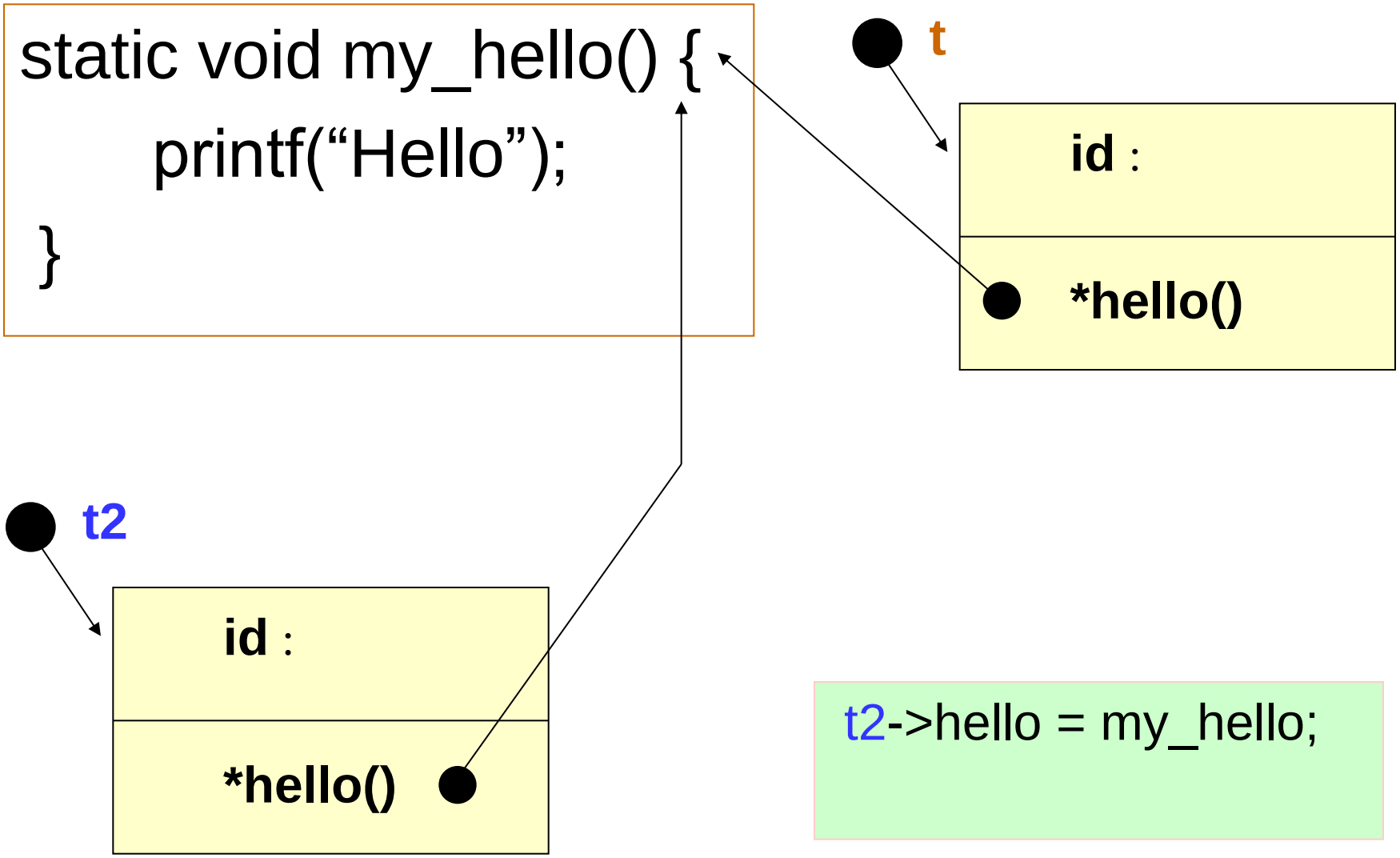

```
CC *t, *t2;
t = (CC *)malloc(sizeof(CC));
t2 = (CC *)malloc(sizeof(CC));
```

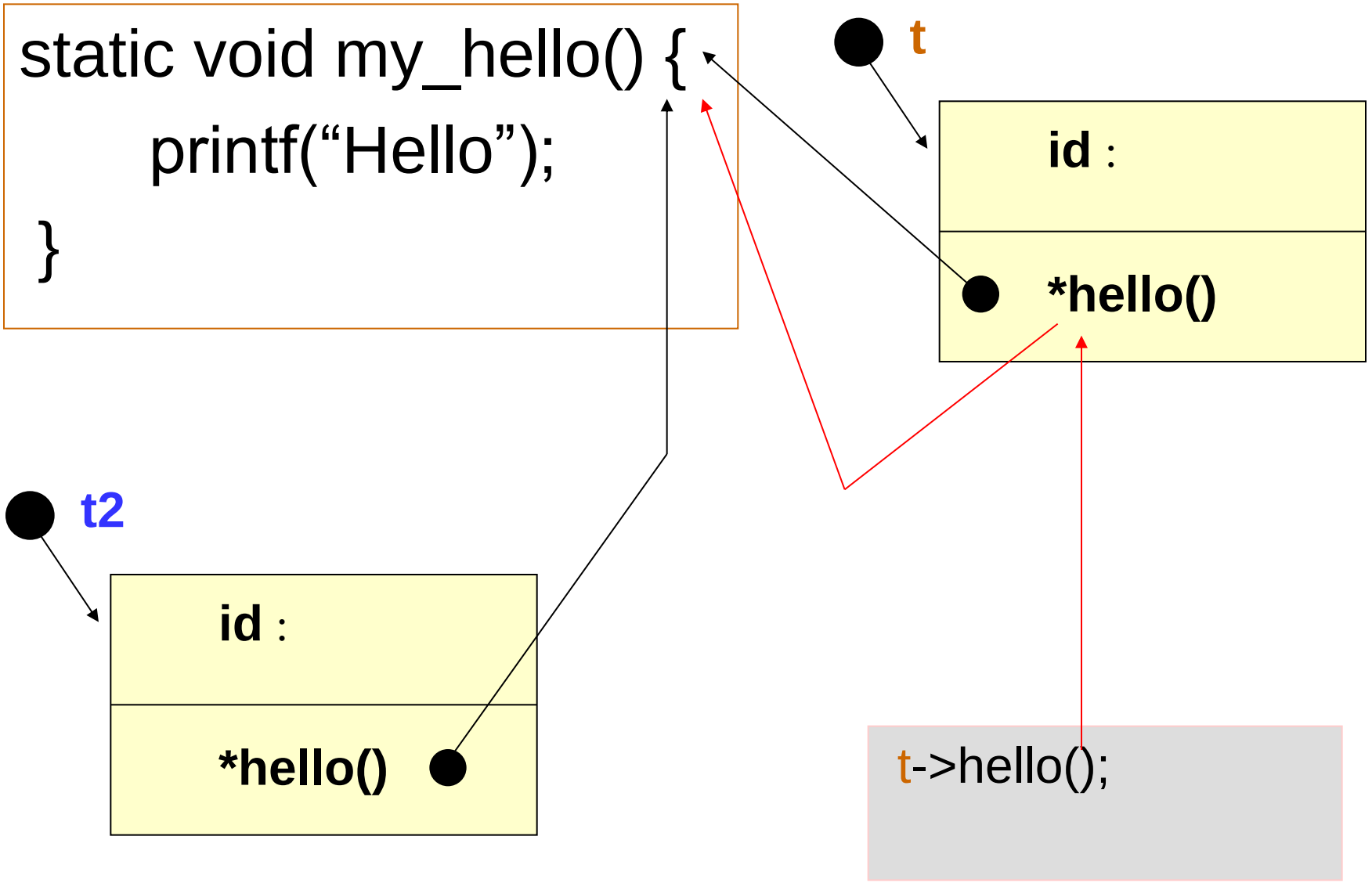
```
static void my_hello() {
        printf("Hello");
 }
```

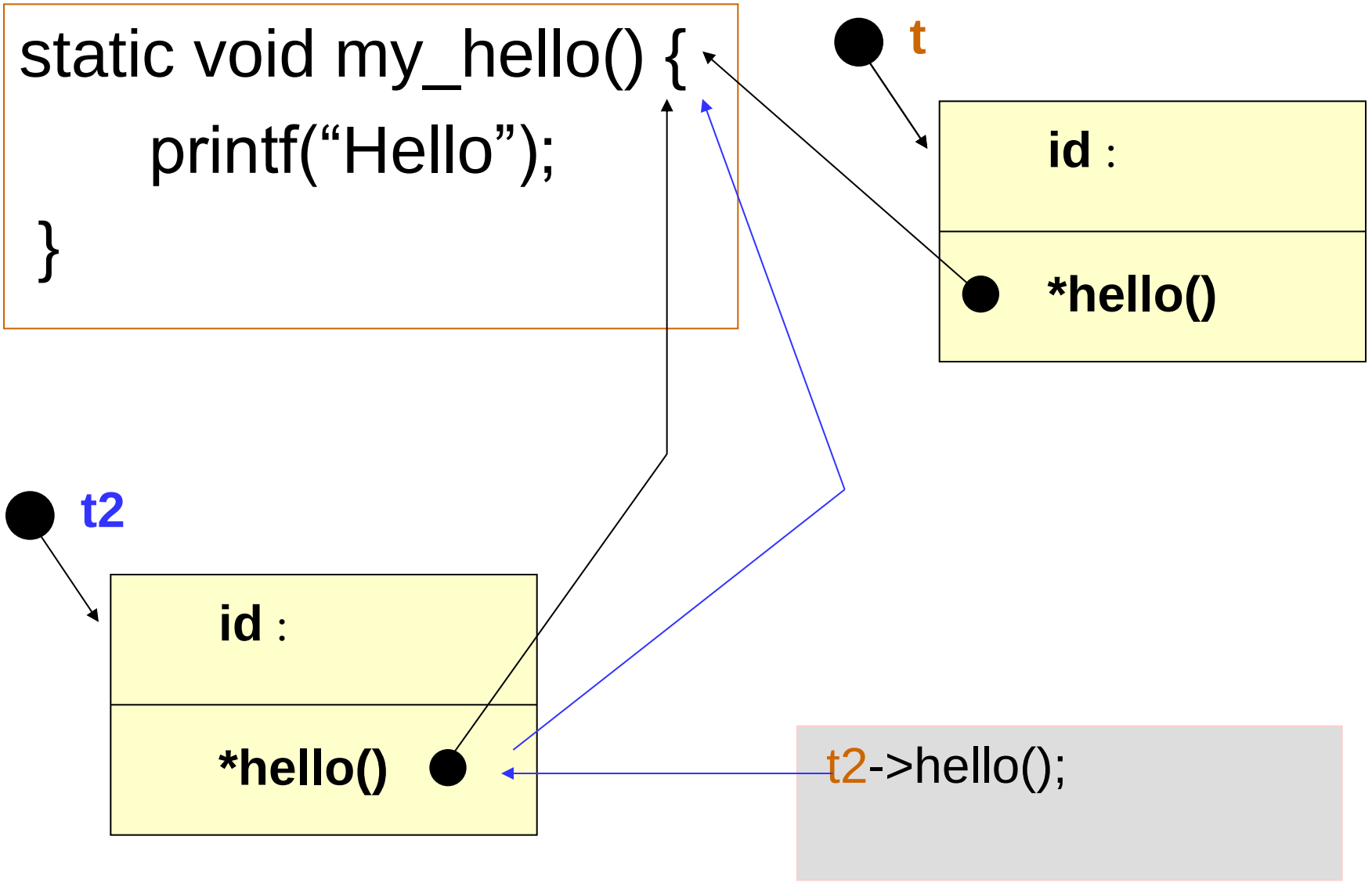





```
t->hello = my_hello;
```

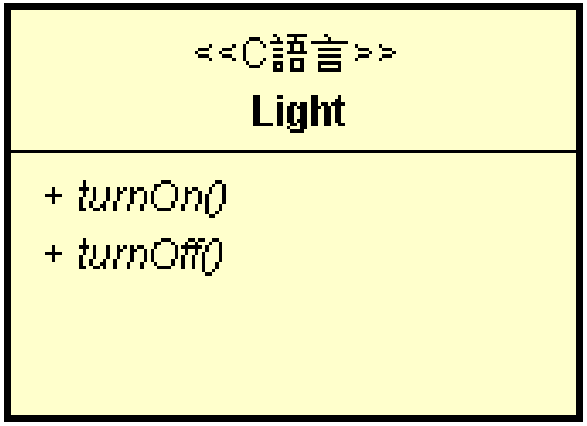

static void my_hello() {
printf("Hello");
}
t
id :
*hello()
t2
id :
*hello()
t2->hello = my_hello;


static void my_hello() {
printf("Hello");
}
t
id :
*hello()
t2
id :
*hello()
t->hello();


static void my_hello() {
printf("Hello");
}
t
id :
*hello()
t2
id :
*hello()
t2->hello();

範例

- 兹以C来定义一个Light类(class)，创建其对象(object)，并调用其函数。



定义Light类

```
struct Light {
    void (*turnOn)();
    void (*turnOff)();
};
typedef struct Light Light;
```

撰写函数：

```
static void turnOn(){
        printf( “ON” );
 }
static void turnOff() {
        printf( “OFF” ); }
```

创建对象，调用函数：

```
void main() {
    struct Light *led  =  (Light *)malloc(sizeof(Light));
    led->turnOn = turnOn;
    led->turnOff = turnOff;

    led->turnOn();
    led->turnOff();
  }
```

定義結構

```
typedef struct Light Light;
struct Light {
    void (*turnOn)();
    void (*turnOff)();
};
```

撰寫main()

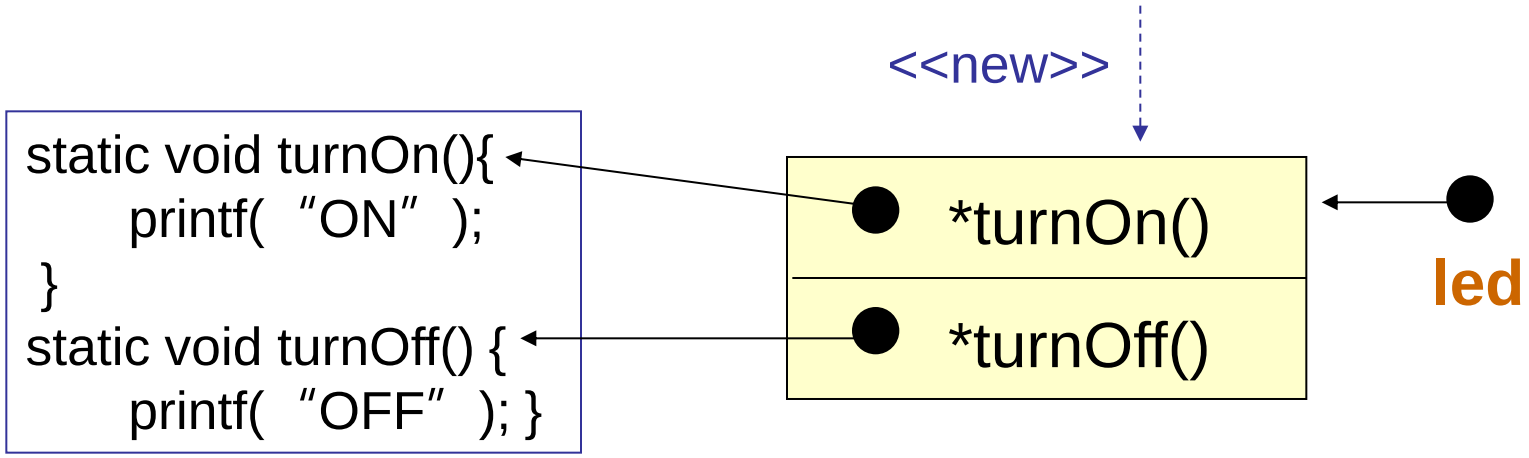
```
void main() {
    struct Light *led
        =  (Light *)malloc(sizeof(Light));
    led->turnOn = turnOn;
    led->turnOff = turnOff;
    led->turnOn();    led->turnOff();
     }
```

撰寫函數

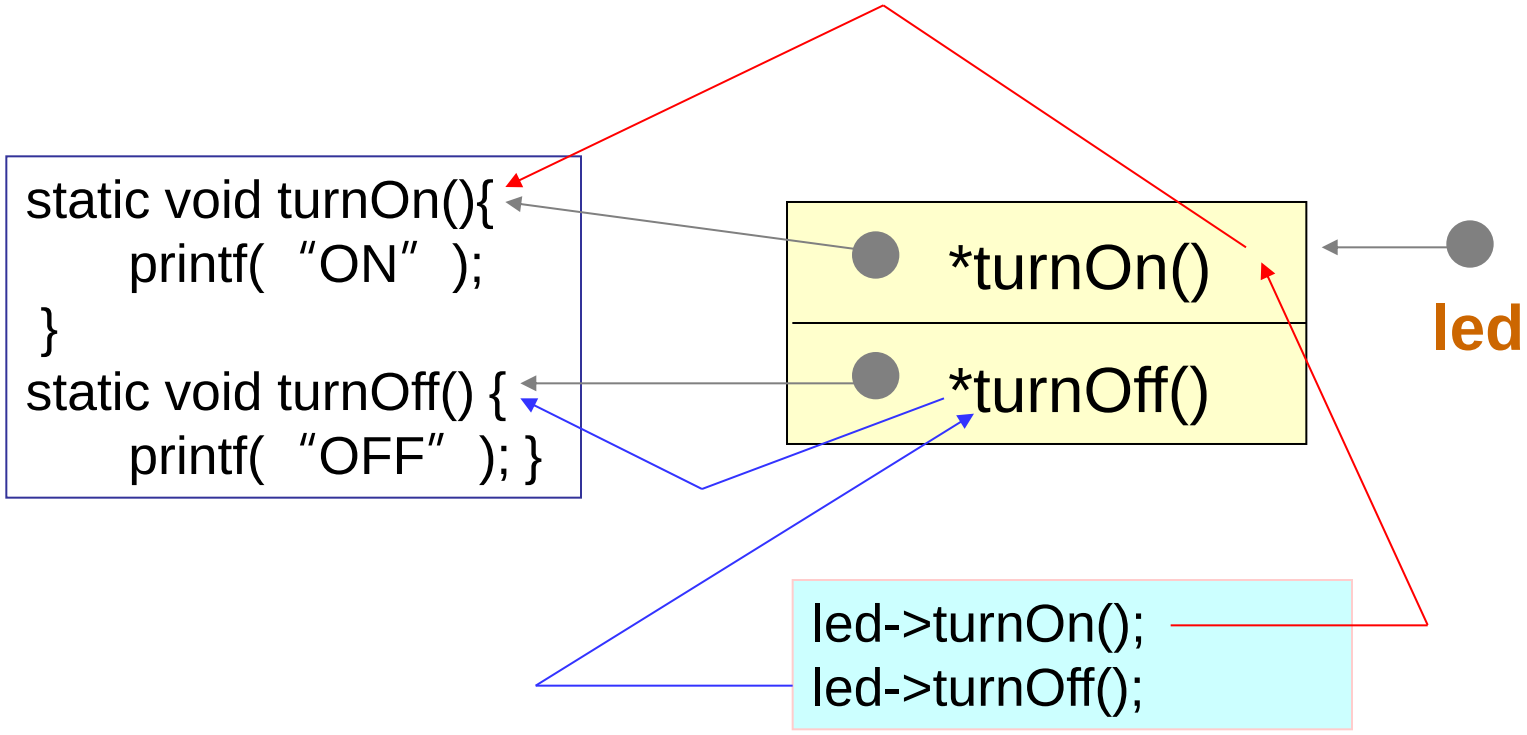
```
static void turnOn(){
        printf( “ON” );
 }
static void turnOff() {
        printf( “OFF” ); }
```

```
typedef struct Light Light;
struct Light {
    void (*turnOn)();
    void (*turnOff)();
};
```

```
void main() {
    struct Light *led
        =  (Light *)malloc(sizeof(Light));
    led->turnOn = turnOn;
    led->turnOff = turnOff;
    led->turnOn();
    led->turnOff();
     }
```

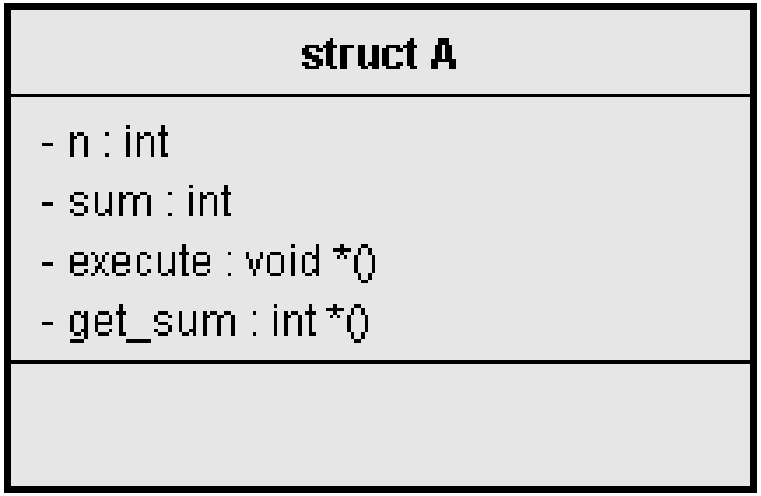


調用函數

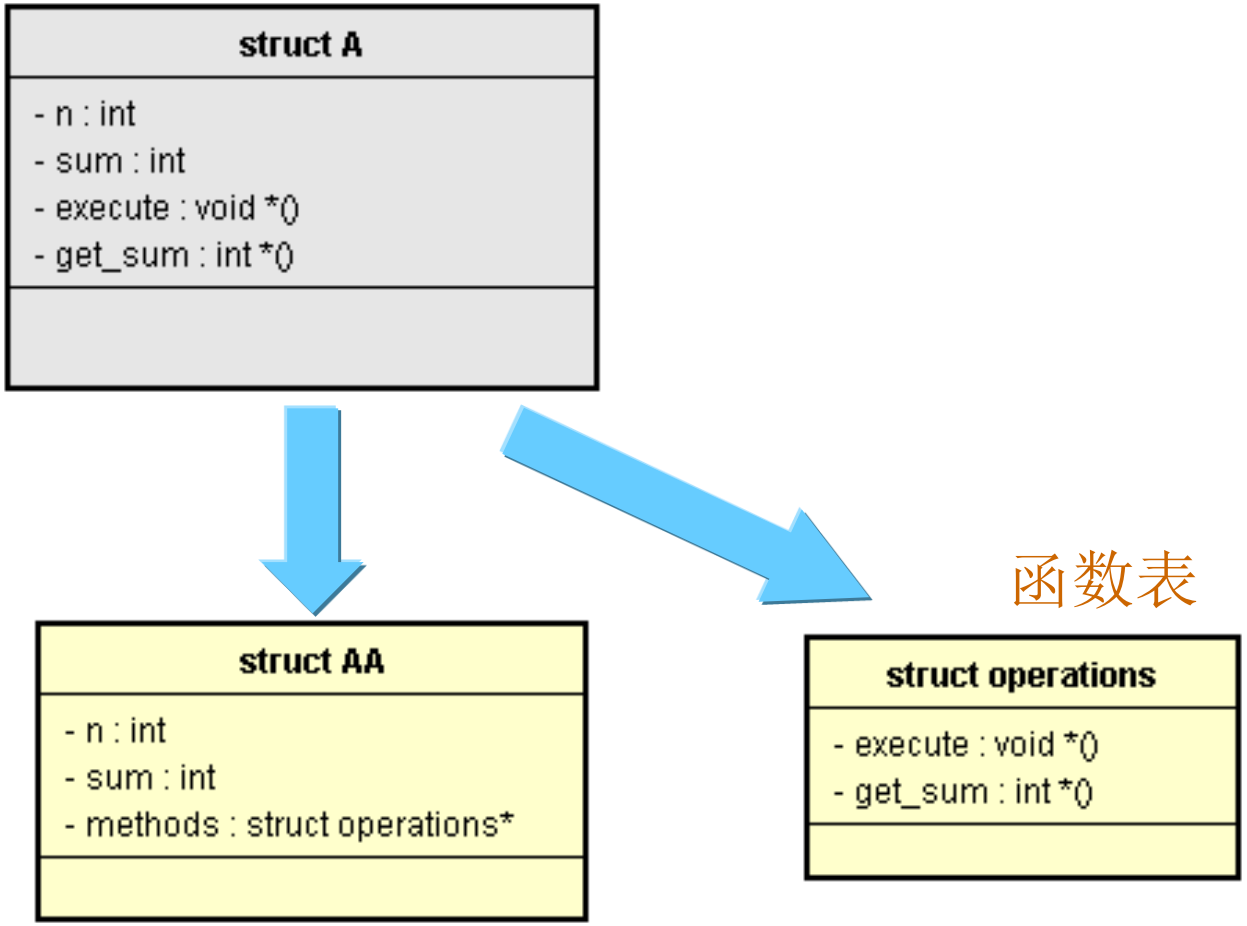

static void turnOn(){
printf(“ON”);
}
static void turnOff() {
printf(“OFF”); }
*turnOn()
*turnOff()
led
led->turnOn();
led->turnOff();

1.5 C的函数表(function)概念

- 从一个C的struct谈起

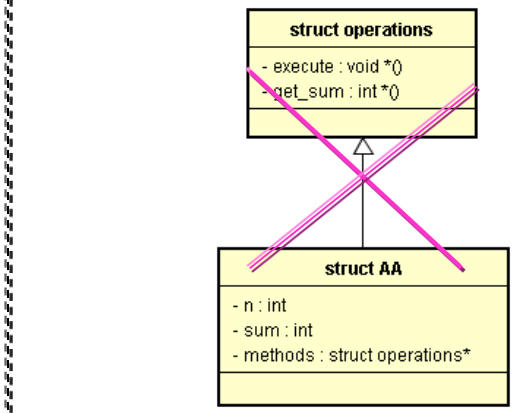


- 把函数部分独立出来，成为一个函数表(function table)。

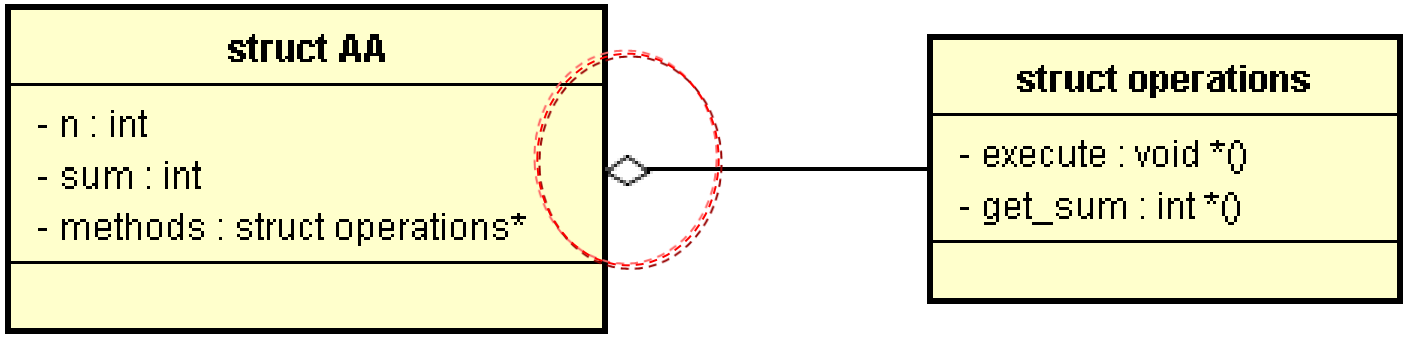


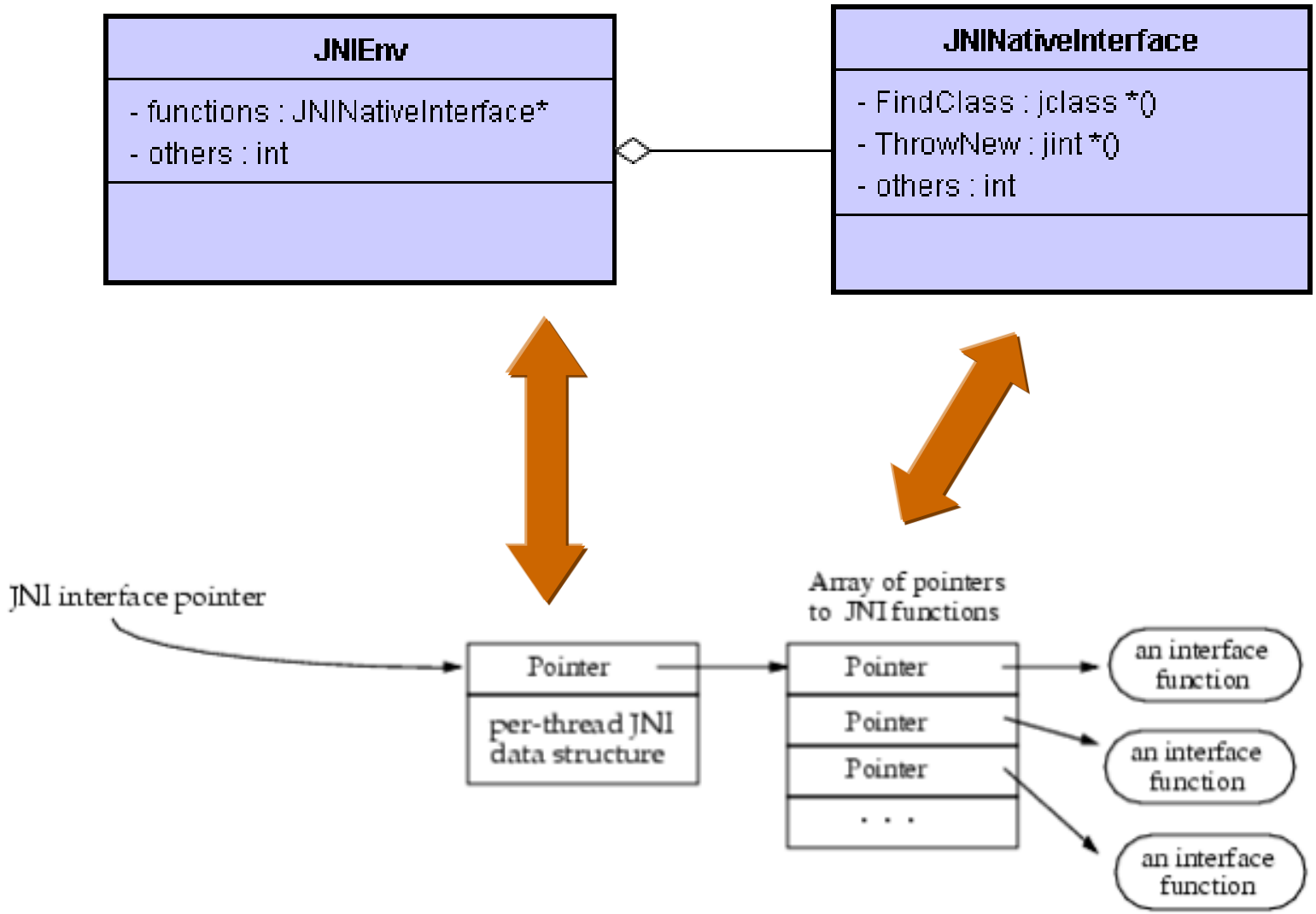
两者之间是什么关系呢?

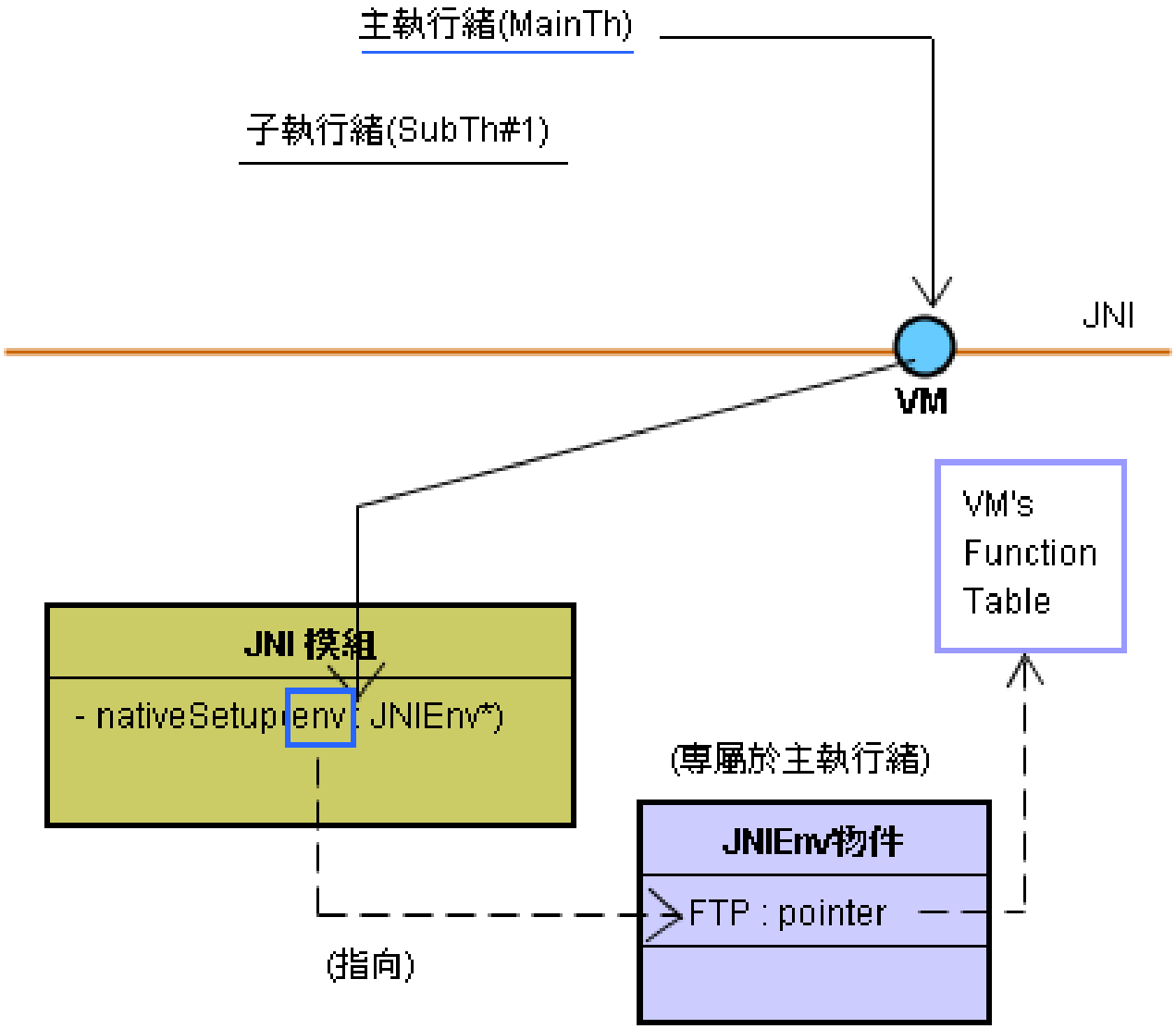
- 两者之间是一种Whole-Part组合(Aggregation)关系；而不是继承(Inheritance)关系。

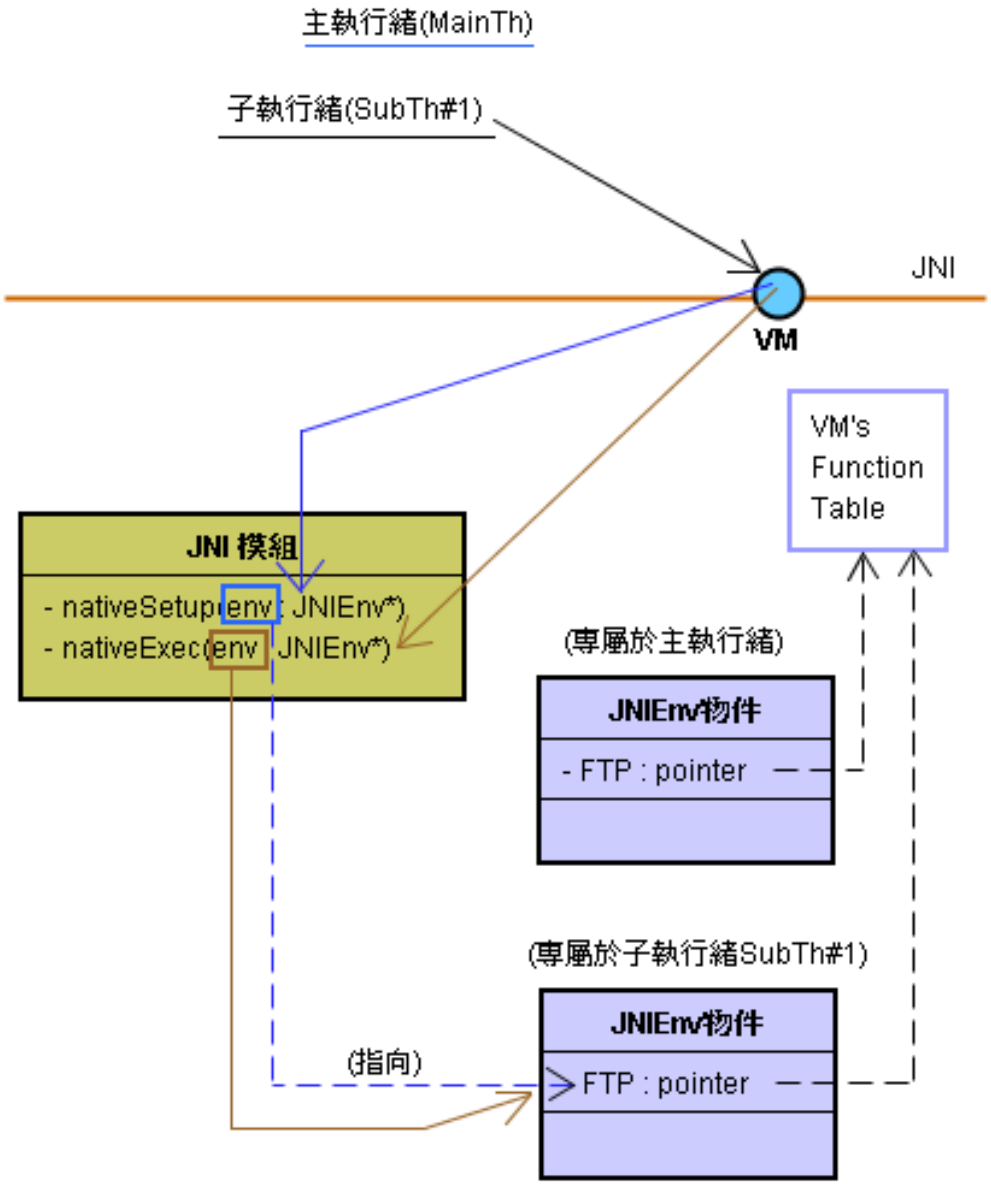


Whole-Part关系




JNIEnv
- functions : JNINativeInterface*
- others : int
JNINativeInterface
- FindClass : jclass *()
- ThrowNew : jint *()
- others : int
JNI interface pointer
Pointer
per-thread JNI data structure
Array of pointers to JNI functions
Pointer
Pointer
Pointer
. . .
an interface function
an interface function
an interface function


主執行緒(MainTh)
子執行緒(SubTh#1)
JNI
VM
VM's
Function
Table
JNI 模組
- nativeSetup(env : JNIEnv*)
(專屬於主執行緒)
JNIEnv物件
FTP : pointer
(指向)


主執行緒(MainTh)
子執行緒(SubTh#1)
JNI
VM
VM's
Function
Table
JNI 模組
- nativeSetup(env : JNIEnv*)
- nativeExec(env : JNIEnv*)
(專屬於主執行緒)
JNIEnv物件
- FTP : pointer
(專屬於子執行緒SubTh#1)
JNIEnv物件
FTP : pointer
(指向)

~ Continued ~