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

E01_f

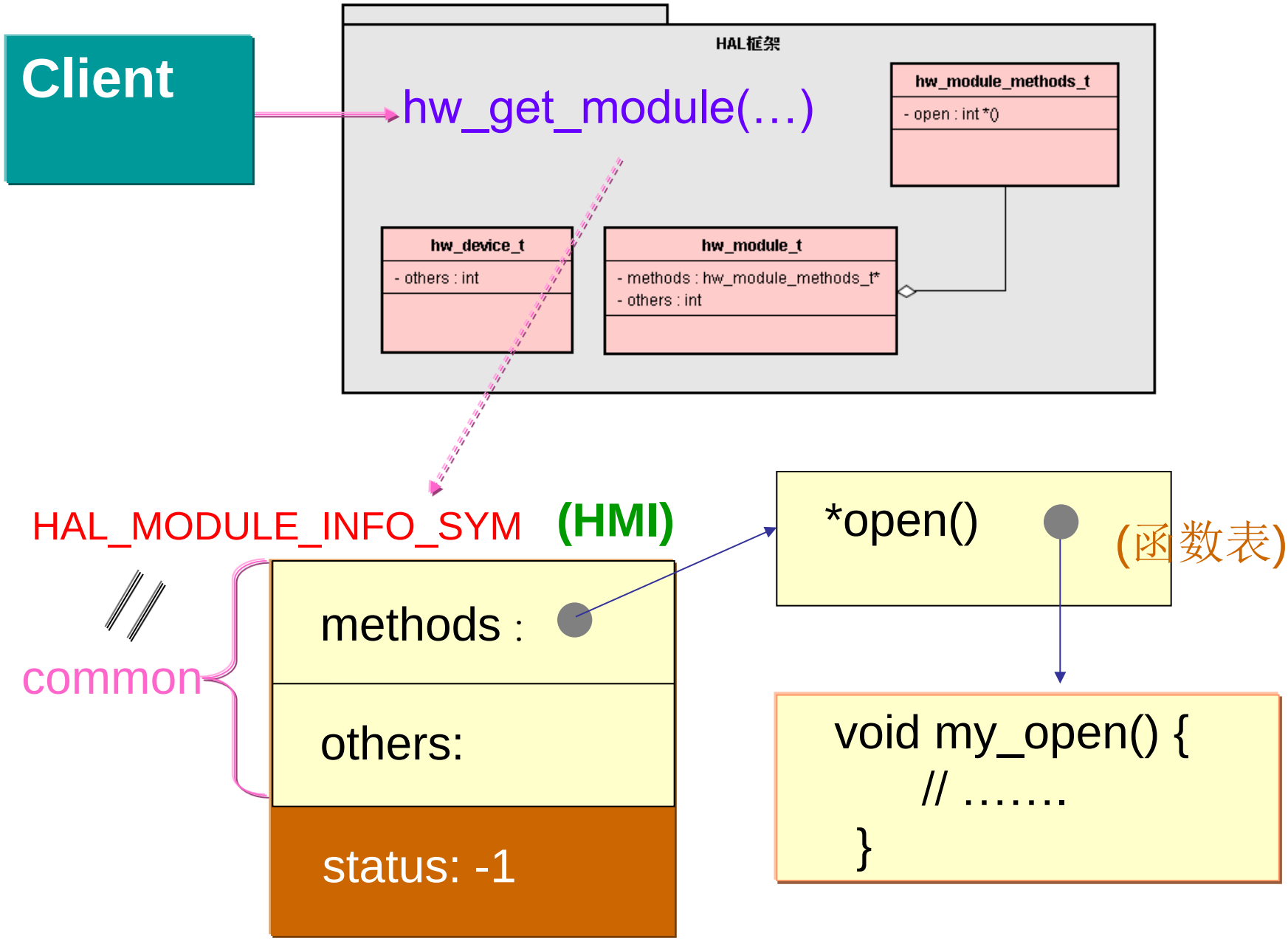
HAL框架与Stub开发(f)

By 高煥堂

4、HAL插件(Stub)的代码范例

前言

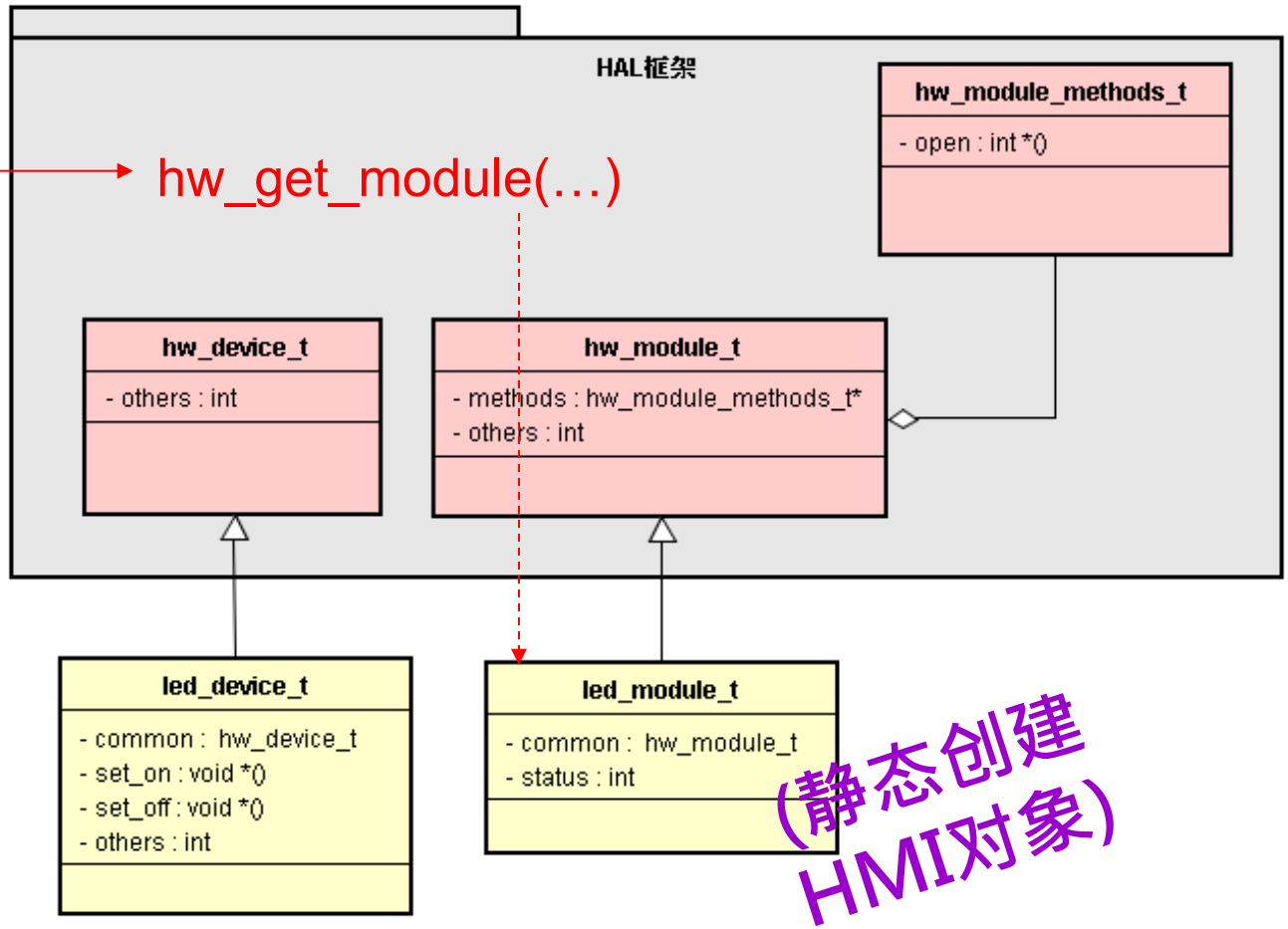
Client使用HAL的第1个步骤


Client
HAL框架
hw_get_module(…)
hw_module_methods_t
- open : int *()
hw_device_t
- others : int
hw_module_t
- methods : hw_module_methods_t*
- others : int
HAL_MODULE_INFO_SYM
(HMI)
common
methods :
others:
status: -1
*open()
(函数表)
void my_open() {
// …….
}

- Client调用hw_get_module()公用函数，找到取名为HMI(即led_module_t)对象的指针；并回传其指针。
- 此led_module_t对象内含一个hw_module_t对象。
- 这两个大、小对象的指针值是相同的。

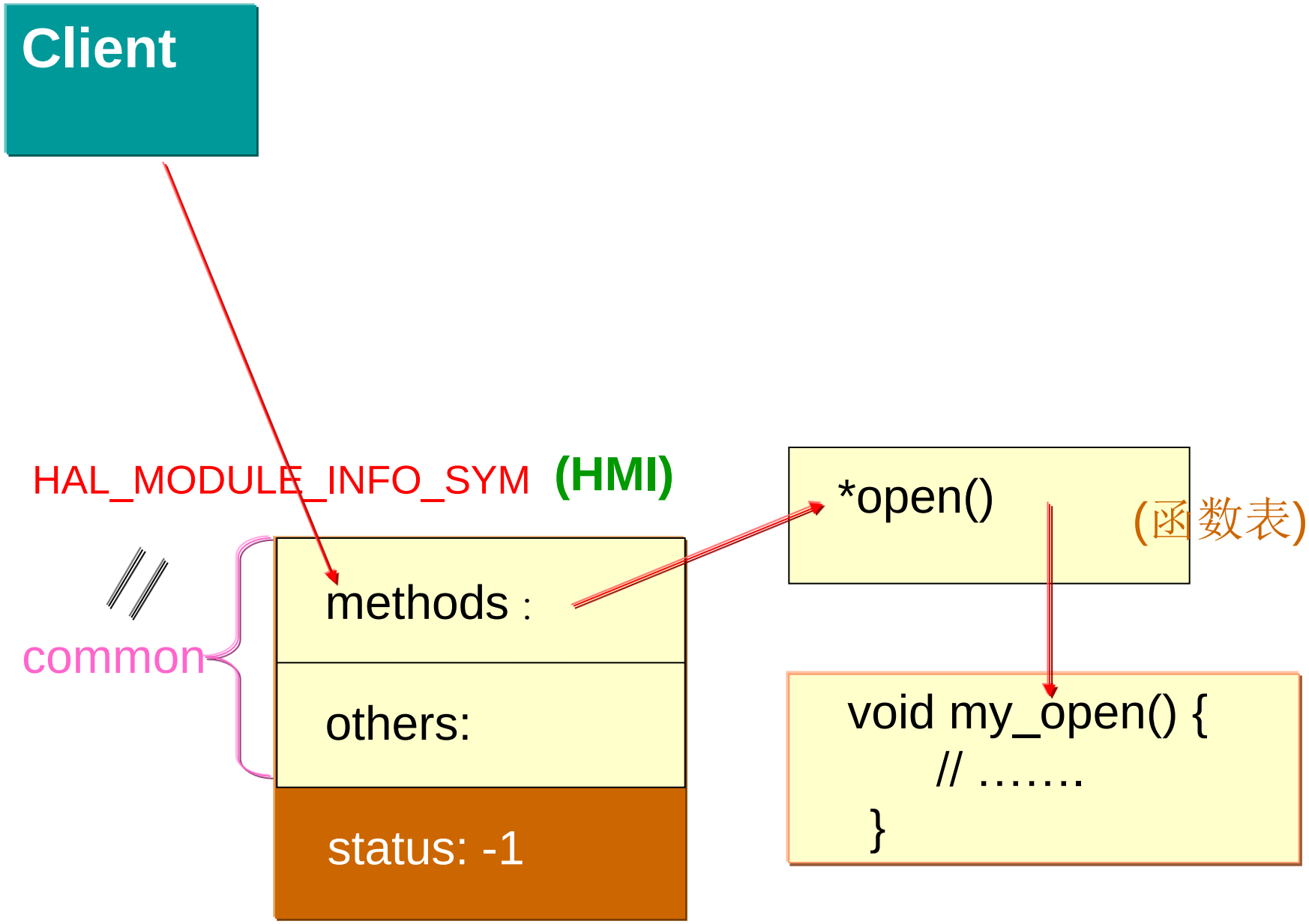
第1个步骤

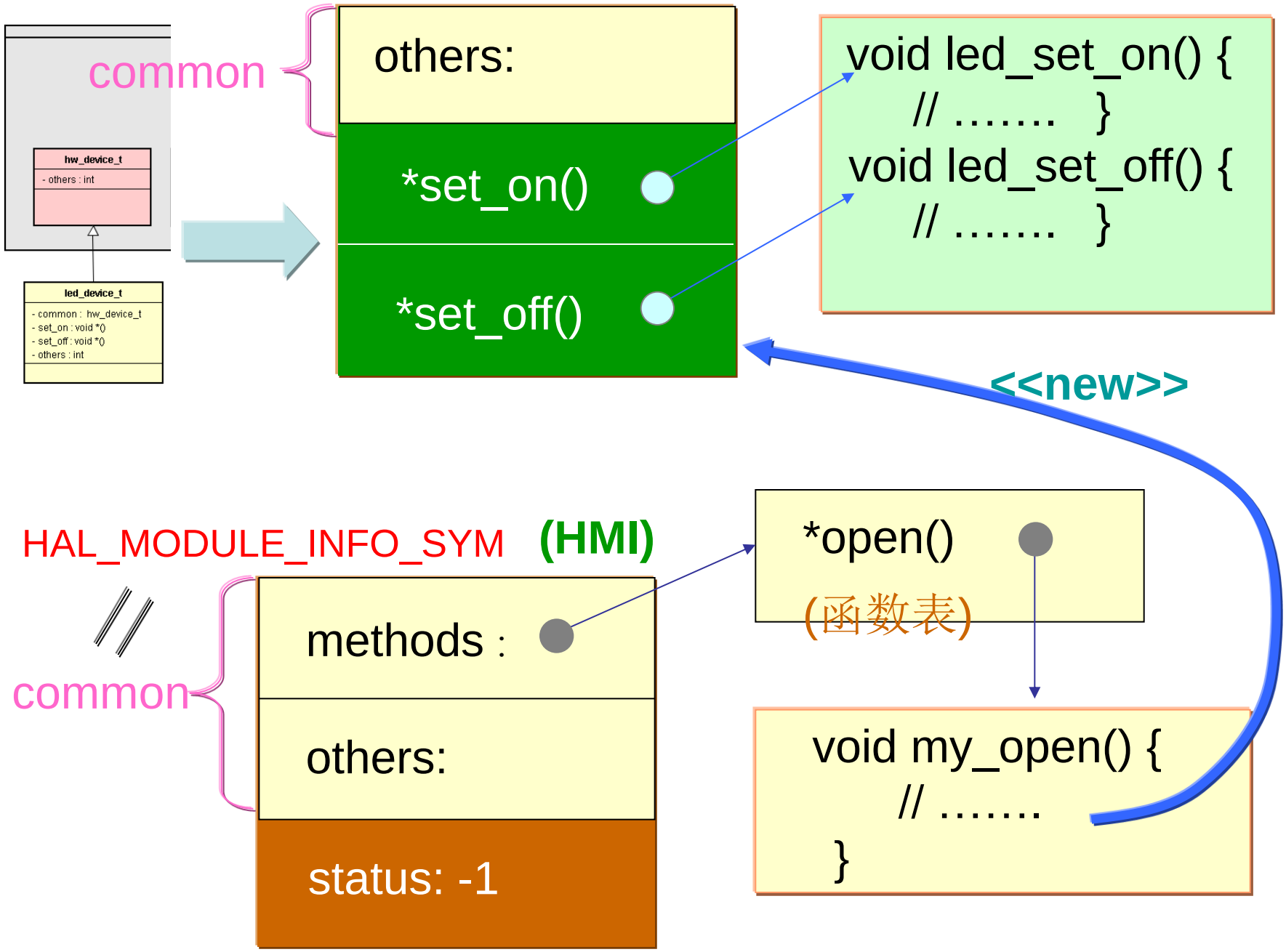
Client



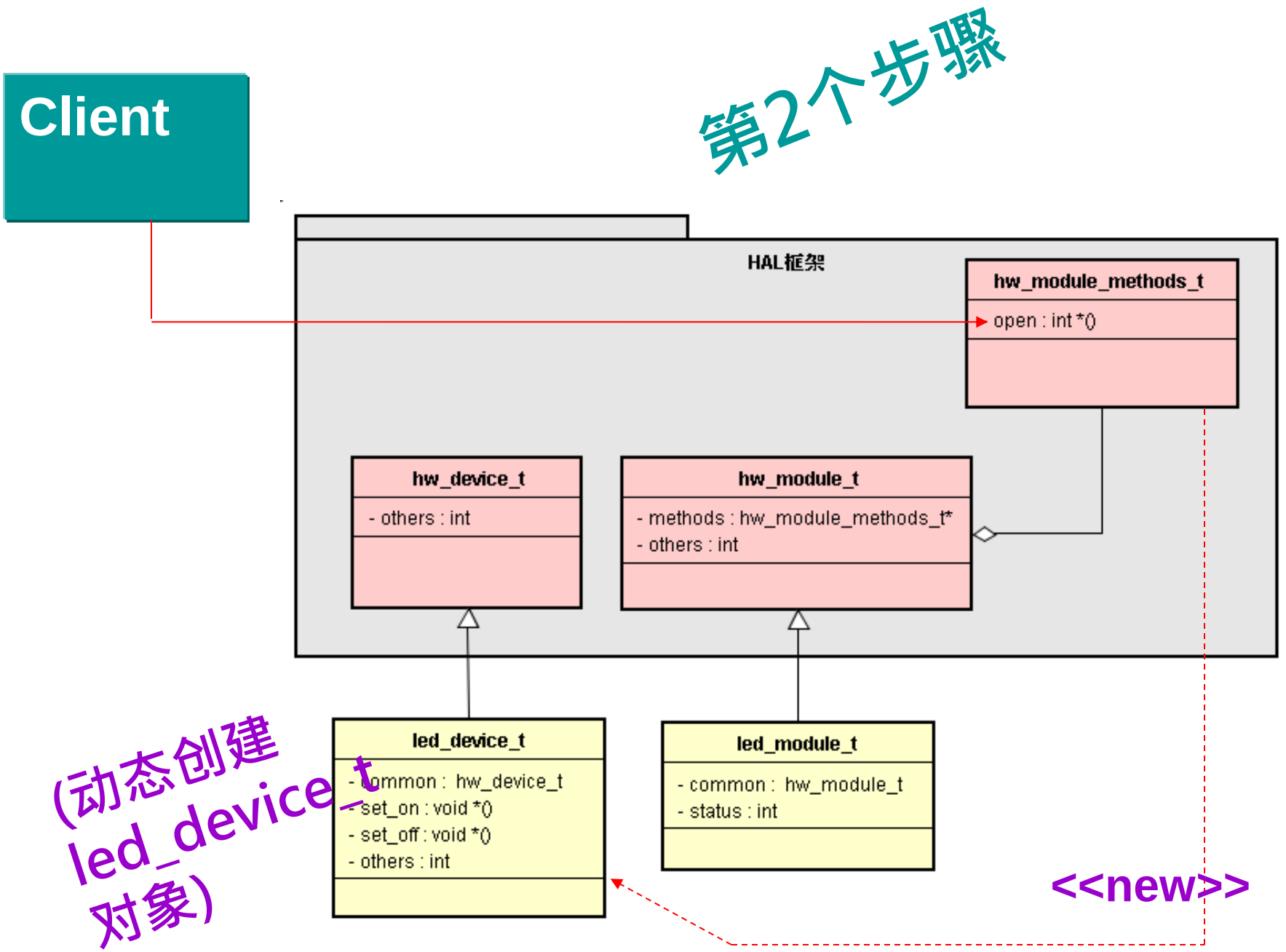
(静态创建HMI对象)

Client使用HAL的第2个步骤

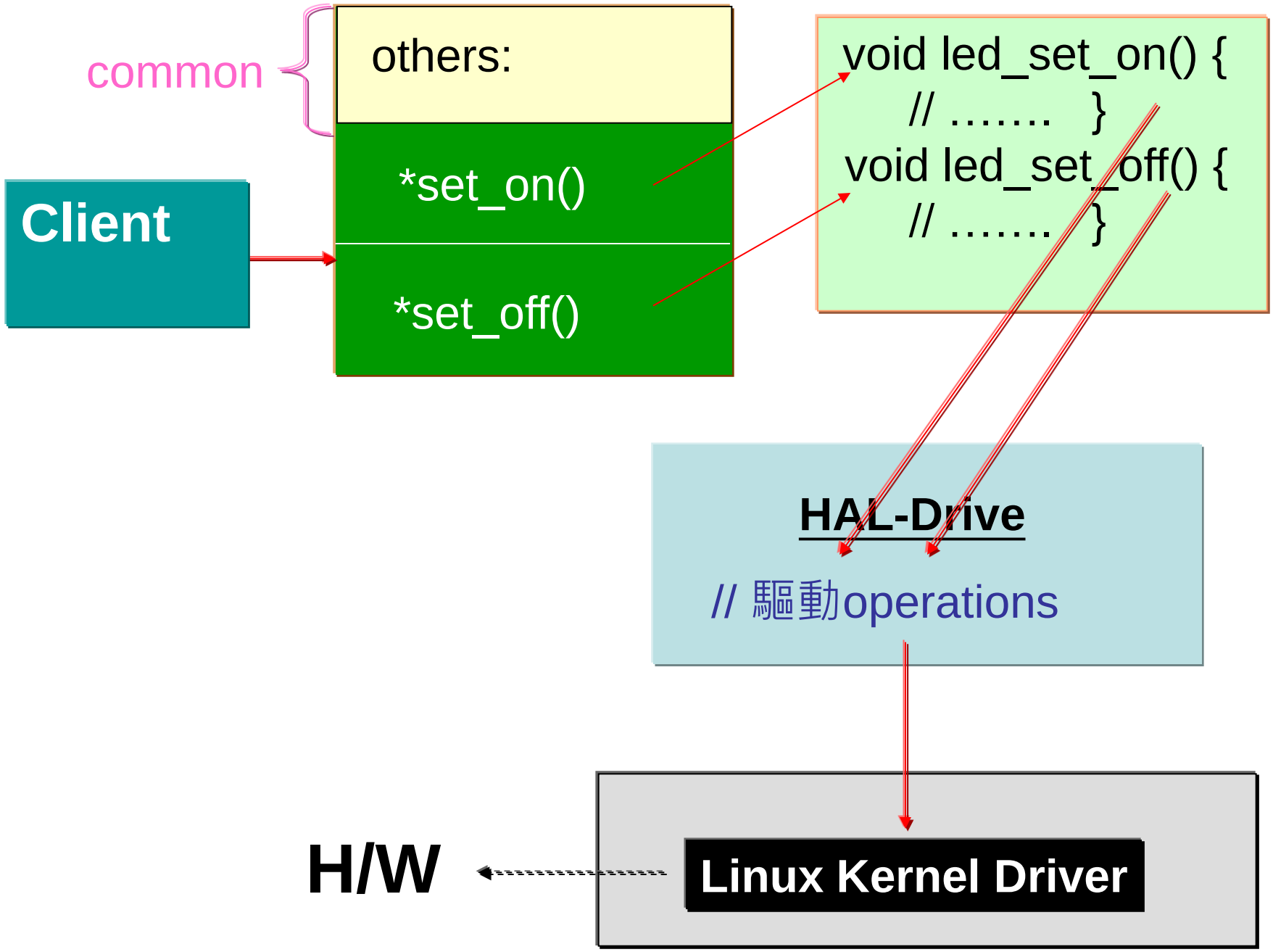

Client
HAL_MODULE_INFO_SYM (HMI)
common
methods :
others:
status: -1
*open()
(函数表)
void my_open() {
// .....
}


common
hw_device_t
- others : int
led_device_t
- common : hw_device_t
- set_on : void *()
- set_off : void *()
- others : int
others:
*set_on()
*set_off()
void led_set_on() {
// ……. }
void led_set_off() {
// ……. }
<<new>>
HAL_MODULE_INFO_SYM
(HMI)
common
methods :
others:
status: -1
*open()
(函数表)
void my_open() {
// …….
}

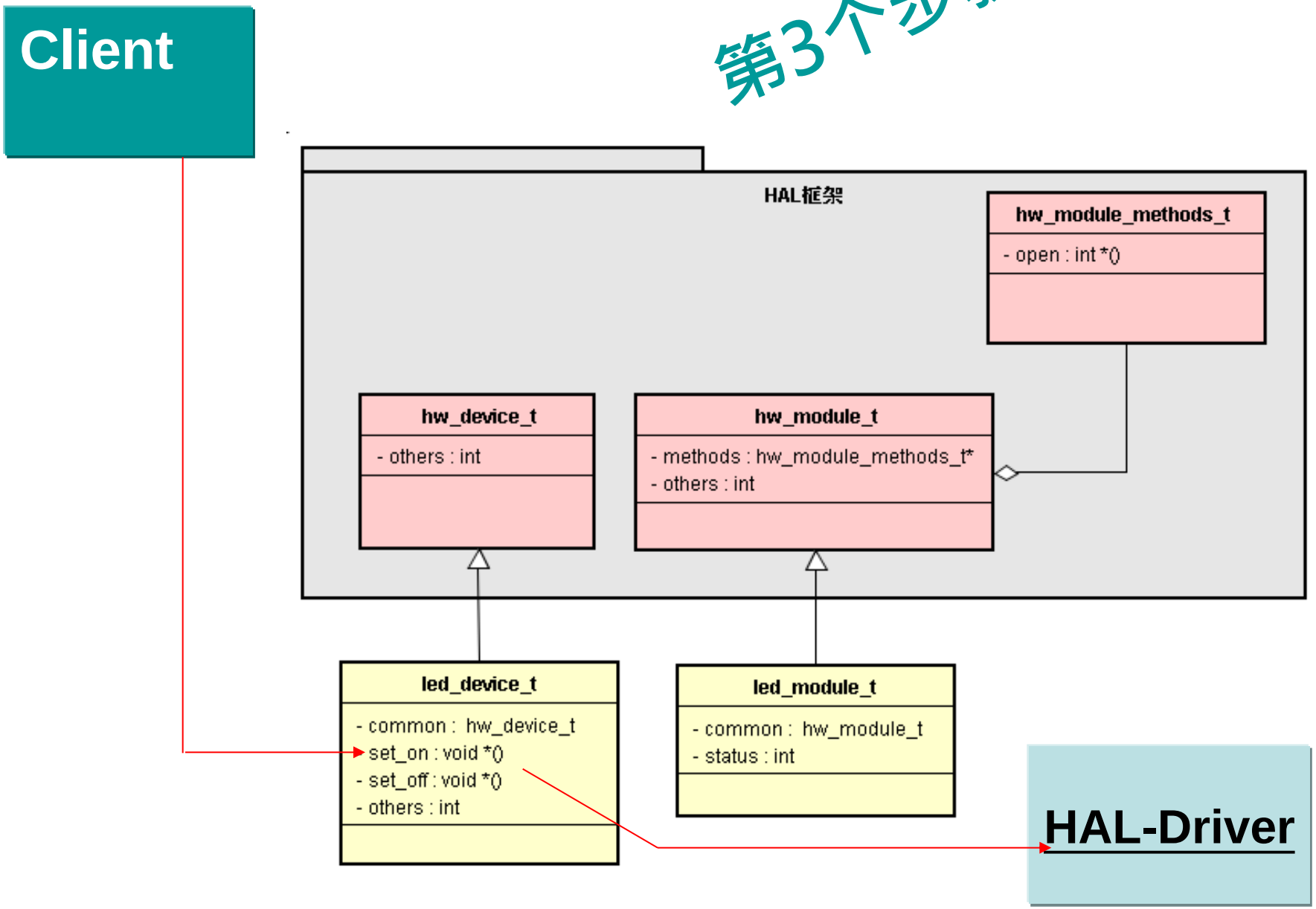
- Client调用hw_module_t的open()函数，也就是hw_module_methods_t里的open()函数。
- 此open()函数创建led_device_t对象，其内含一个hw_device_t对象；并回传其指针。
- 这两个大、小对象的指针值是相同的。


Client
第2个步骤
HAL框架
hw_module_methods_t
open : int *()
hw_device_t
- others : int
hw_module_t
- methods : hw_module_methods_t*
- others : int
led_device_t
- common : hw_device_t
- set_on : void *()
- set_off : void *()
- others : int
led_module_t
- common : hw_module_t
- status : int
(动态创建
led_device_t
对象)
<<new>>

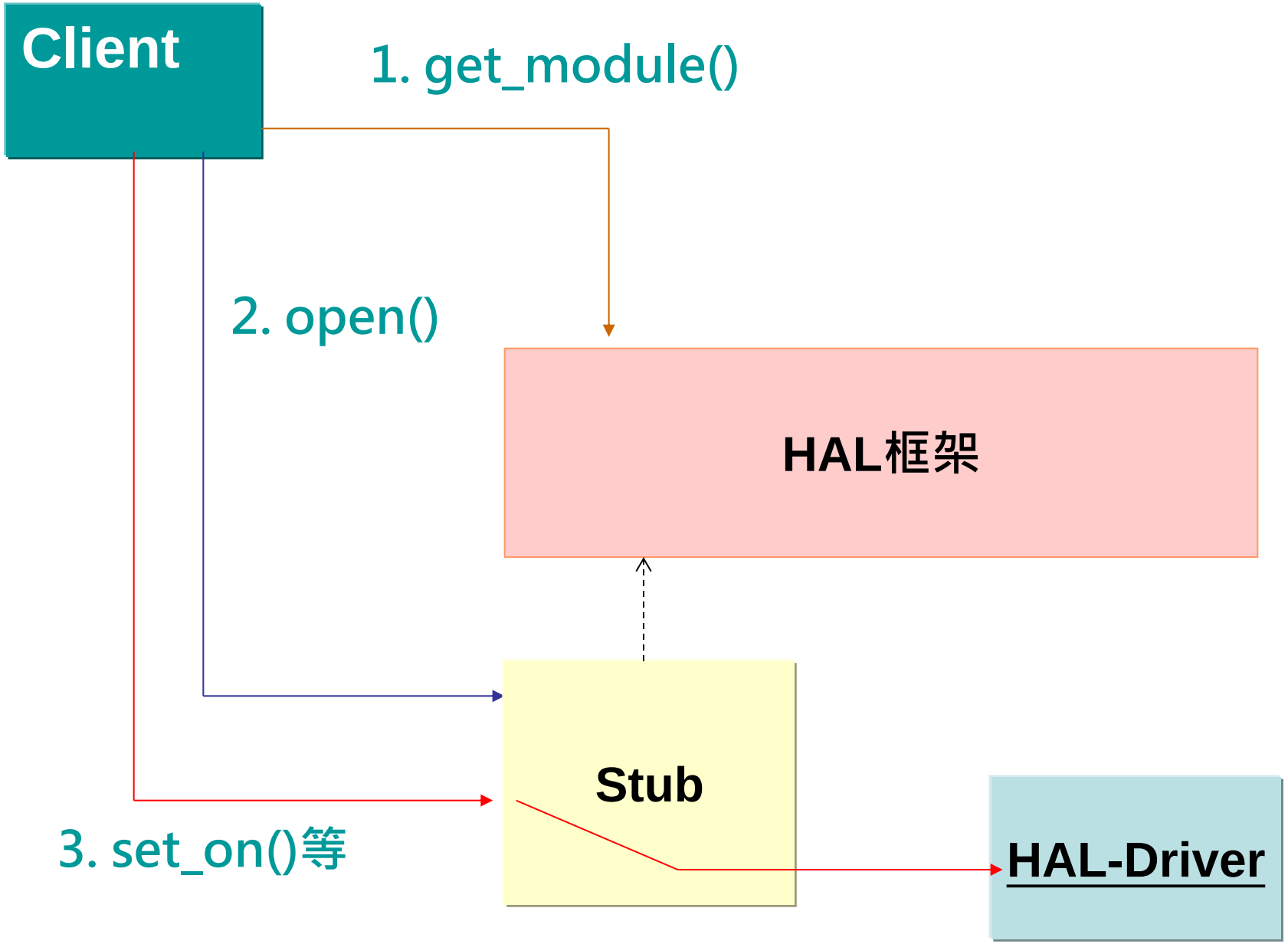
Client使用HAL的第3个步骤

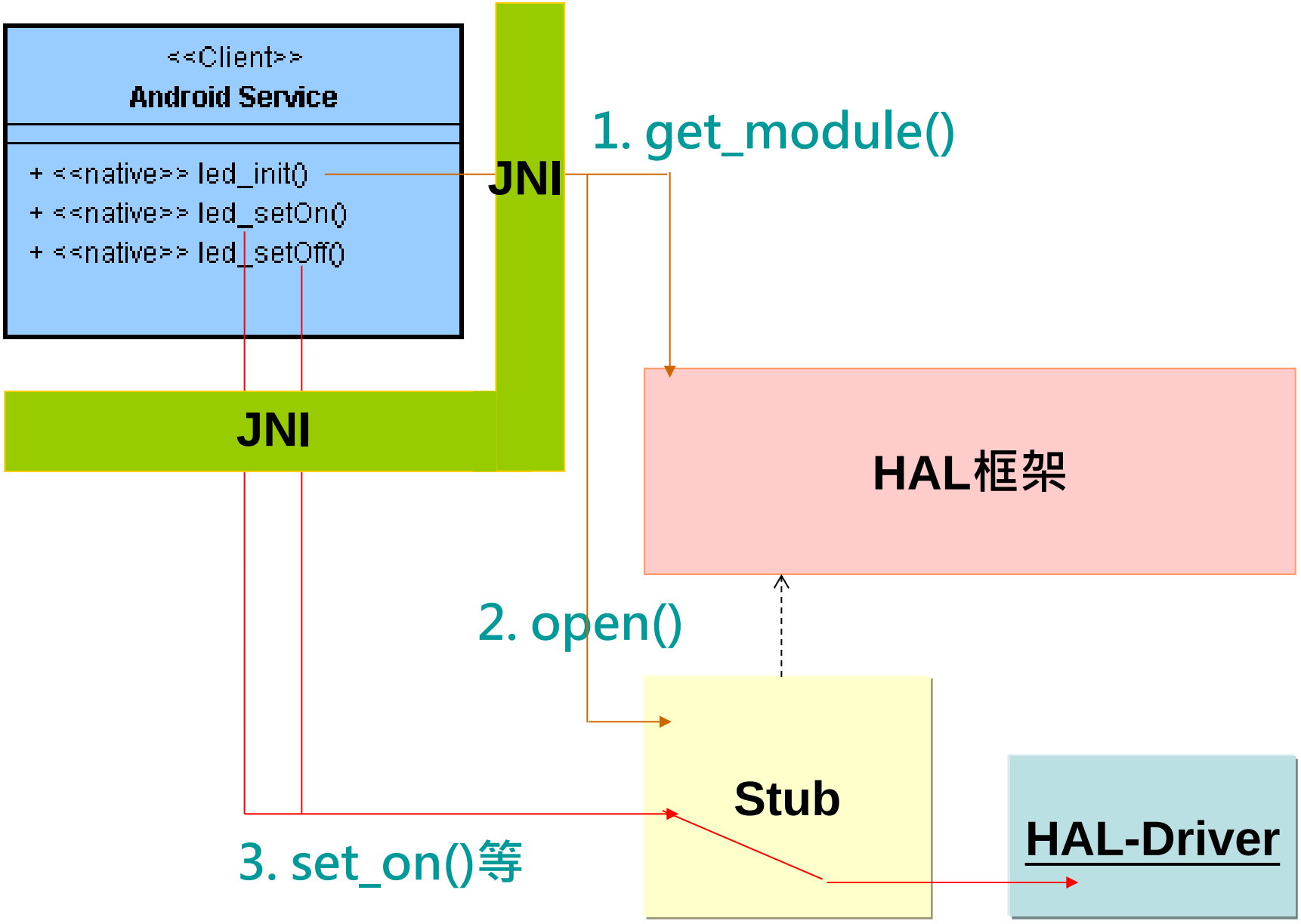

common
others:
*set_on()
*set_off()
Client
void led_set_on() {
// ……. }
void led_set_off() {
// ……. }
HAL-Drive
// 驅動operations
H/W
Linux Kernel Driver

第3个步骤

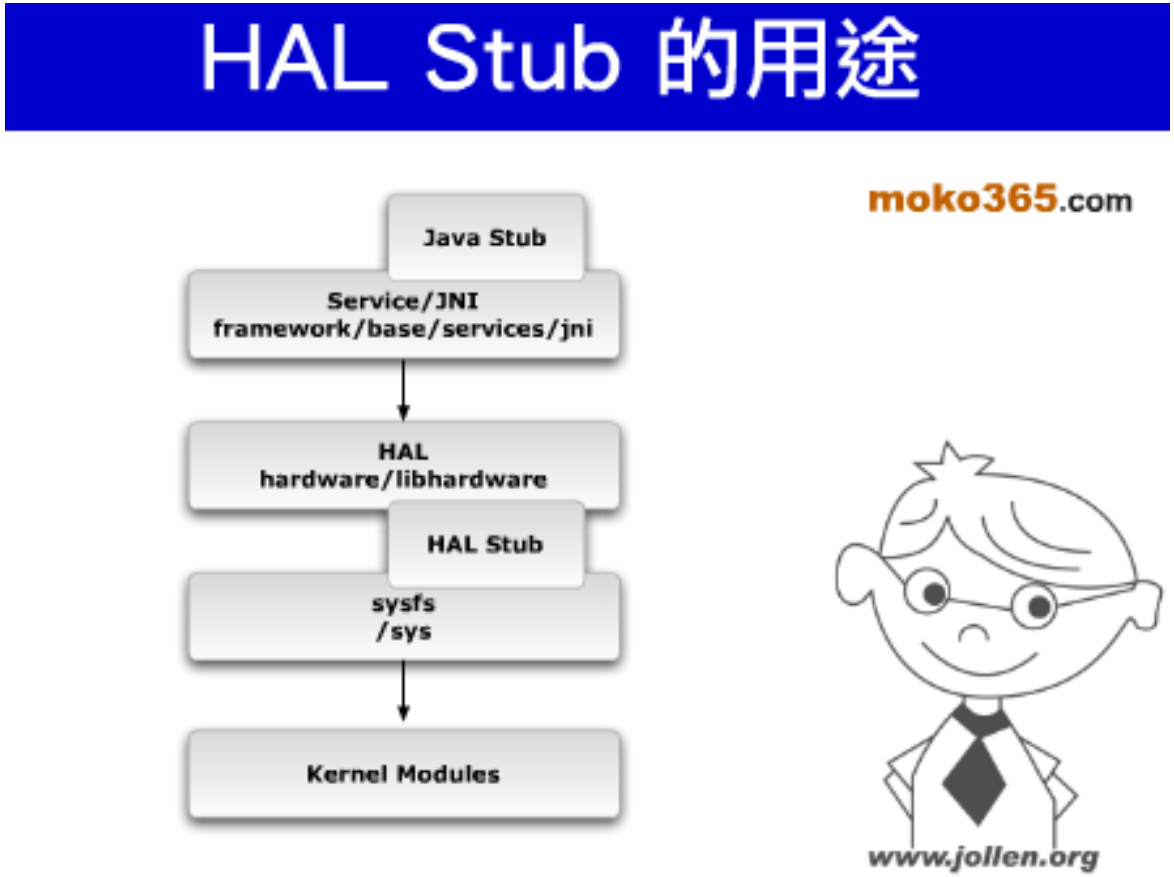

Client
HAL框架
hw_module_methods_t
- open : int *()
hw_device_t
- others : int
hw_module_t
- methods : hw_module_methods_t*
- others : int
led_device_t
- common : hw_device_t
- set_on : void *()
- set_off : void *()
- others : int
led_module_t
- common : hw_module_t
- status : int
HAL-Driver

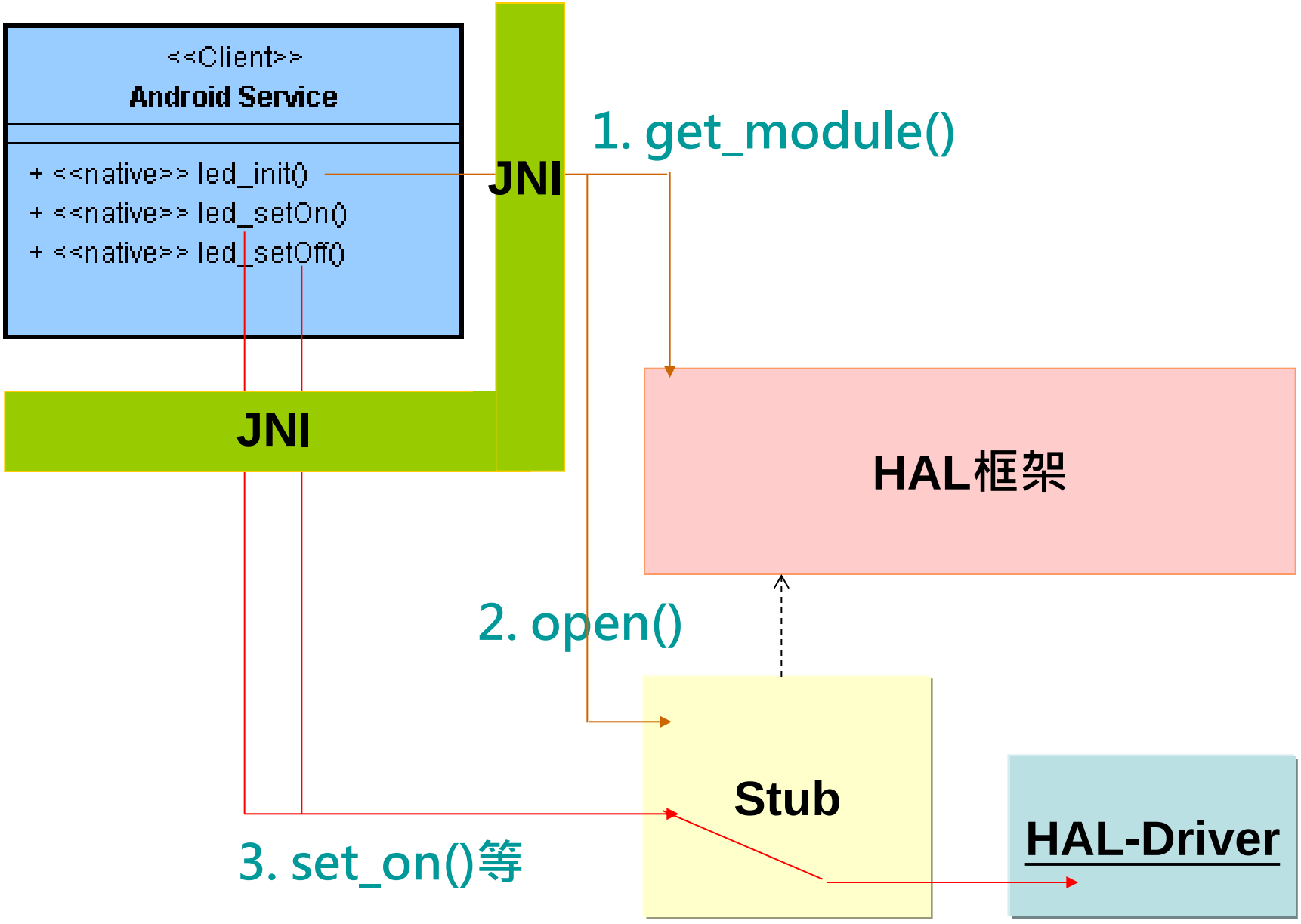
Summary：3步骤


Client
1. get_module()
2. open()
HAL框架
Stub
3. set_on()等
HAL-Driver


<<Client>>
Android Service
+ <<native>> led_init()
+ <<native>> led_setOn()
+ <<native>> led_setOff()
JNI
JNI
1. get_module()
HAL框架
2. open()
Stub
HAL-Driver
3. set_on()等

<< Jollen Chen 的LED范例 >>




<<Client>>
Android Service
+ <<native>> led_init()
+ <<native>> led_setOn()
+ <<native>> led_setOff()
JNI
JNI
1. get_module()
HAL框架
2. open()
Stub
HAL-Driver
3. set_on()等

```
/*  led.h  */
#include <hardware/hardware.h>
#include <fcntl.h>
#include <errno.h>
#include <cutils/log.h>
#include <cutils/atomic.h>
#define LED_HARDWARE_MODULE_ID "led"
struct led_module_t {
        struct hw_module_t common;
        /* support API for LEDServices constructor */
};
```

```
struct led_control_device_t {
   struct hw_device_t common;
   /* supporting control APIs go here */
   int (*getcount_led)(struct led_control_device_t *dev);
   int (*set_on)(struct led_control_device_t *dev);
   int (*set_off)(struct led_control_device_t *dev);
};
struct led_control_context_t {
        struct led_control_device_t device;
};
```

撰写函数实现代码

```
/*  led.c  */
#define LOG_TAG "LedStub"
#include <hardware/hardware.h>
#include <fcntl.h>
#include <errno.h>
#include <cutils/log.h>
#include <cutils/atomic.h>
#include <led.h>
static int led_device_close(struct hw_device_t* device){
        struct led_control_context_t* ctx =
                (struct led_control_context_t*)device;
        if (ctx)  free(ctx);
        return 0;
}
```

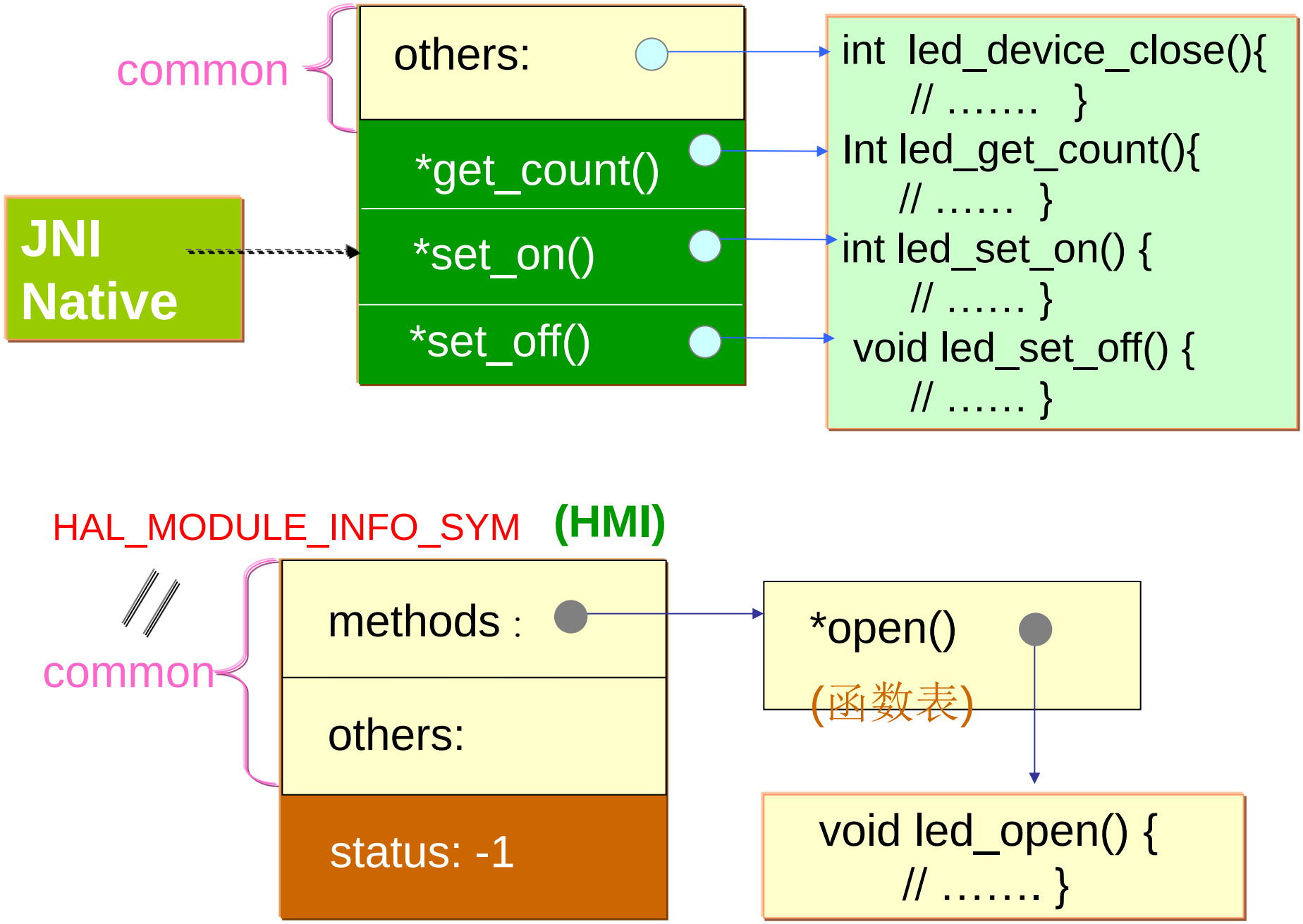
```
static int led_getcount(struct led_control_device_t *dev){
        LOGI("led_getcount");
        return 4;
}
static int led_set_on(struct led_control_device_t *dev){
        //FIXME: do system call to control gpio led
        LOGI("led_set_on");
        return 0;
}
static int led_set_off(struct led_control_device_t *dev){
        //FIXME: do system call to control gpio led
        LOGI("led_set_off");
        return 0;
}
```

```
static int led_open(const struct hw_module_t* module, const char* name,
        struct hw_device_t** device)
{
            struct led_control_context_t *context;
            LOGD("led_device_open");
            context = (struct led_control_context_t *)malloc(sizeof(*context));
            memset(context, 0, sizeof(*context));
            //HAL must init property
            context->device.common.tag= HARDWARE_DEVICE_TAG;
            context->device.common.version = 0;
            context->device.common.module= module;
            context->device.common.close = led_device_close;
            context->device.set_on= led_set_on;
            context->device.set_off= led_set_off;
            context->device.getcount_led = led_getcount;
            *device= (struct hw_device_t *)&(context->device);
            return 0;
}
```

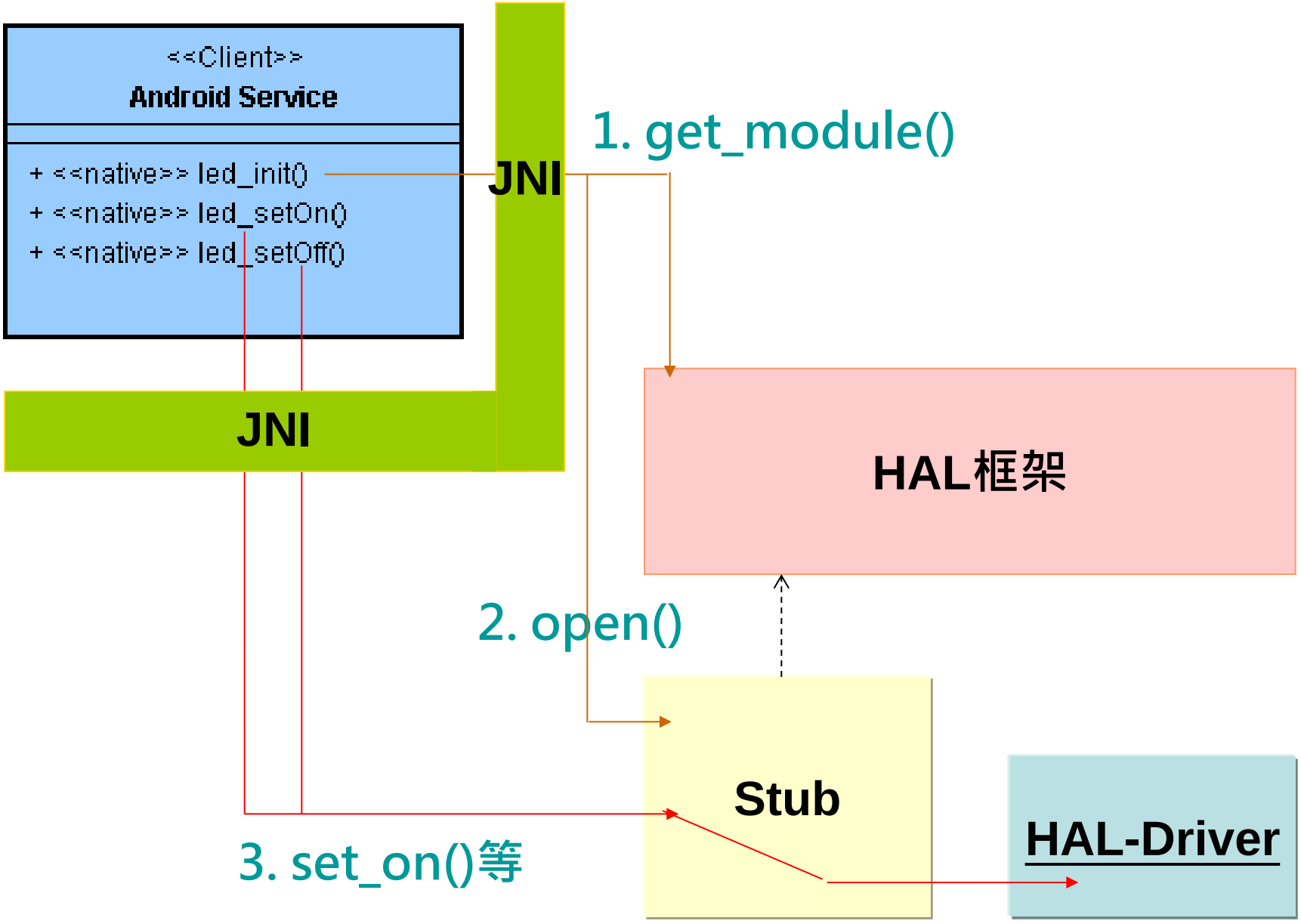
```
static struct hw_module_methods_t led_module_methods = {
    open: led_open
};
```

```
const struct led_module_t HAL_MODULE_INFO_SYM = {
    common: {
        tag: HARDWARE_MODULE_TAG,
        version_major: 1,
        version_minor: 0,
        id: LED_HARDWARE_MODULE_ID,
        name: “Test LED Stub",
        author: “Test Project Team",
        methods: &led_module_methods,
    }
     status: -1,
};
```

At run-time


common
others:
*get_count()
*set_on()
*set_off()
JNI
Native
int led_device_close(){
// ……. }
Int led_get_count(){
// …… }
int led_set_on() {
// …… }
void led_set_off() {
// …… }
HAL_MODULE_INFO_SYM
(HMI)
common
methods :
others:
status: -1
*open()
(函数表)
void led_open() {
// ……. }

5、JNI Native Client 的代码范例


<<Client>>
Android Service
+ <<native>> led_init()
+ <<native>> led_setOn()
+ <<native>> led_setOff()
JNI
JNI
1. get_module()
HAL框架
2. open()
Stub
HAL-Driver
3. set_on()等

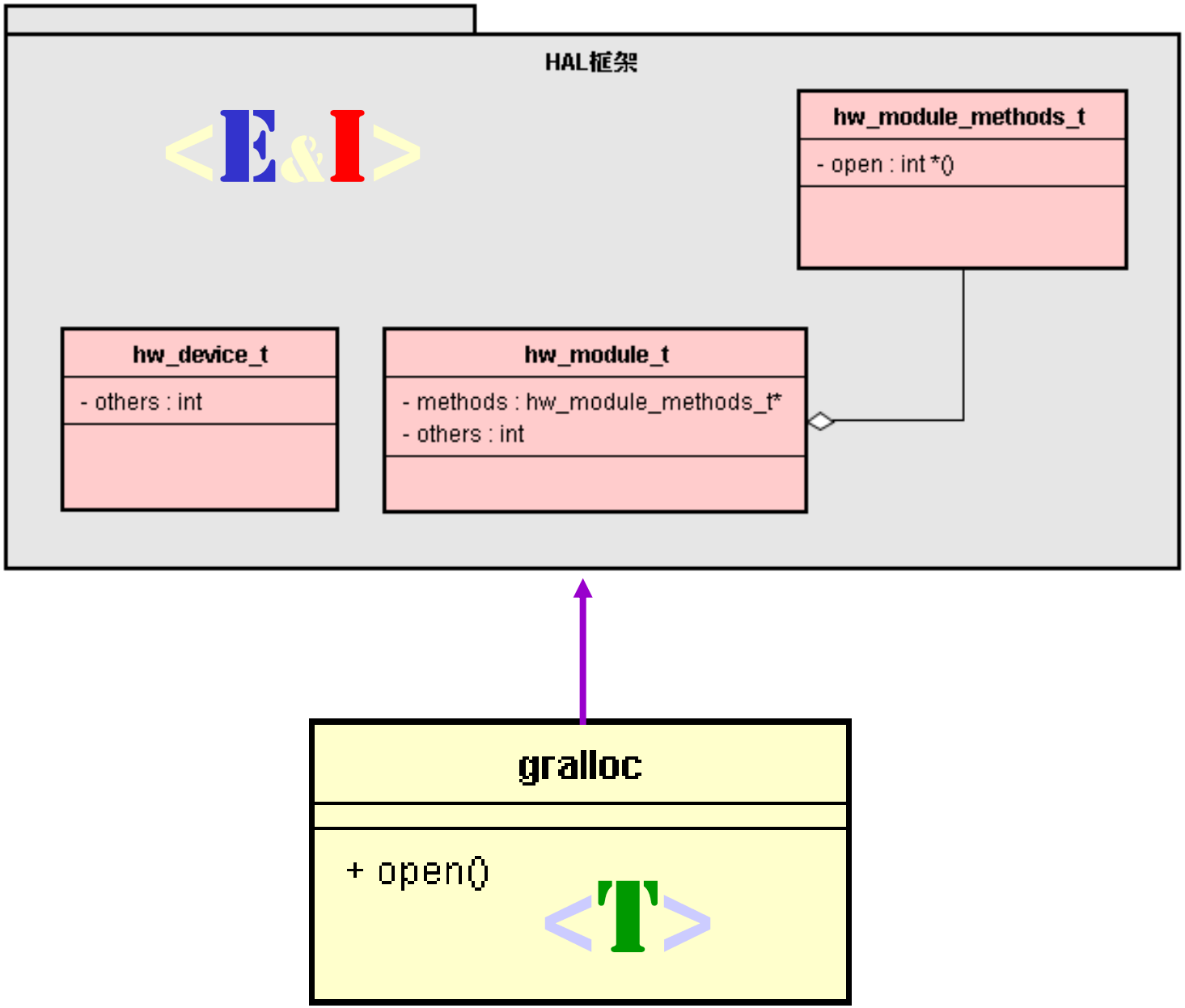
```
static jint led_init(JNIEnv *env, jclass clazz) {
    led_module_t const * module;
    LOGI("led_init");
    if (hw_get_module(LED_HARDWARE_MODULE_ID, (const
                    hw_module_t**)&module) == 0) {
        LOGI("get Module OK");
        sLedModule = (led_module_t *) module;
        if (led_control_open(&module->common, &sLedDevice) != 0) {
            LOGI("led_init error");
            return -1;
        }
     }
    LOGI("led_init success");
    return 0;
}
```

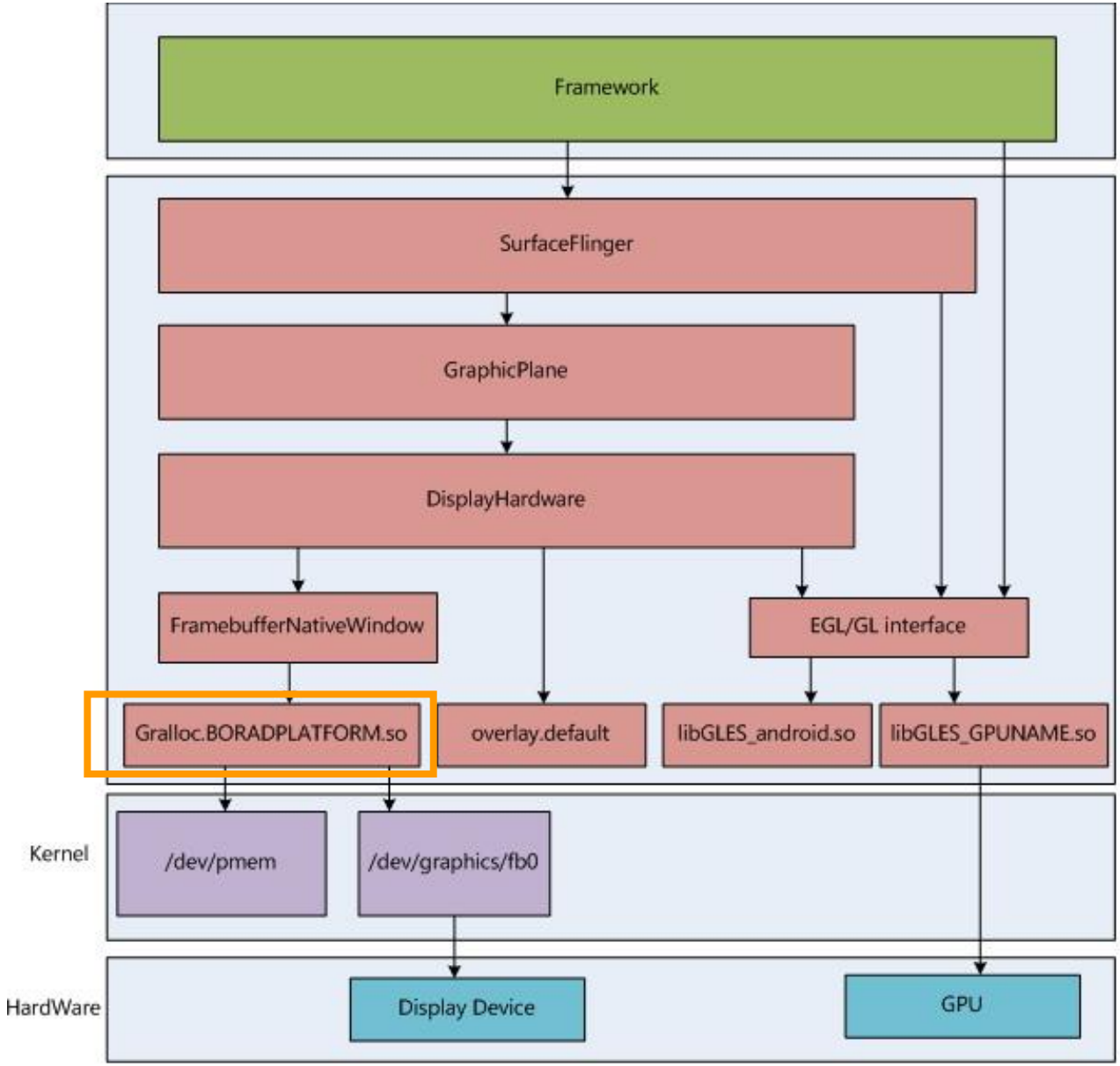
```
// JNI Native C函数
// .........
#include <led.h>
static led_control_device_t *sLedDevice = 0;
static led_module_t* sLedModule=0;

static int get_count(void) {          return 4;  }
static jint led_setOn(JNIEnv* env, jobject thiz) {
        LOGI("led_set_on");
        sLedDevice->set_on(sLedDevice);
        return 0;
}
```

```
static jint led_setOff(JNIEnv* env, jobject thiz) {
        LOGI("led_set_off");
       sLedDevice->set_off(sLedDevice);
       return 0;
}
/** helper APIs */
static inline int led_control_open(const struct hw_module_t* module,
         struct led_control_device_t** device)
        {
              LOGI("led_control_ope");
              return module->methods->open(module,
             LED_HARDWARE_MODULE_ID, (struct hw_device_t**)device);
         }
```

6、观摩Android的实际HAL-Stub范例

-- 用来初始化FrameBuffer的gralloc library


HAL框架
<E&I>
hw_module_methods_t
- open : int *()
hw_device_t
- others : int
hw_module_t
- methods : hw_module_methods_t*
- others : int
gralloc
+ open()
<T>


Framework
SurfaceFlinger
GraphicPlane
DisplayHardware
FramebufferNativeWindow
EGL/GL interface
Gralloc.BORADPLATFORM.so
overlay.default
libGLES_android.so
libGLES_GPUNAME.so
Kernel
/dev/pmem
/dev/graphics/fb0
HardWare
Display Device
GPU

- gralloc library模塊的範例是 HAL gralloc.msm7x30.so。
- 繪圖framebuffer的初始化需要通过 HAL gralloc.msm7x30.so 来完成与底层硬件驱动的适配。
- 不同的vendor可能会实现自己的gralloc library。

- Android通过hw_module_t框架来使用gralloc library，它为framebuffer的初始化提供了需要的gralloc.msm7x30.so业务。

- gralloc library的Stub结构体被命名为HAL_MODULE_INFO_SYM(HMI)。
- 例如，HAL_MODULE_INFO_SYM定义于hardware/msm7k/libgralloc-qsd8k/galloc.cpp。

Thanks...

高煥堂