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_c

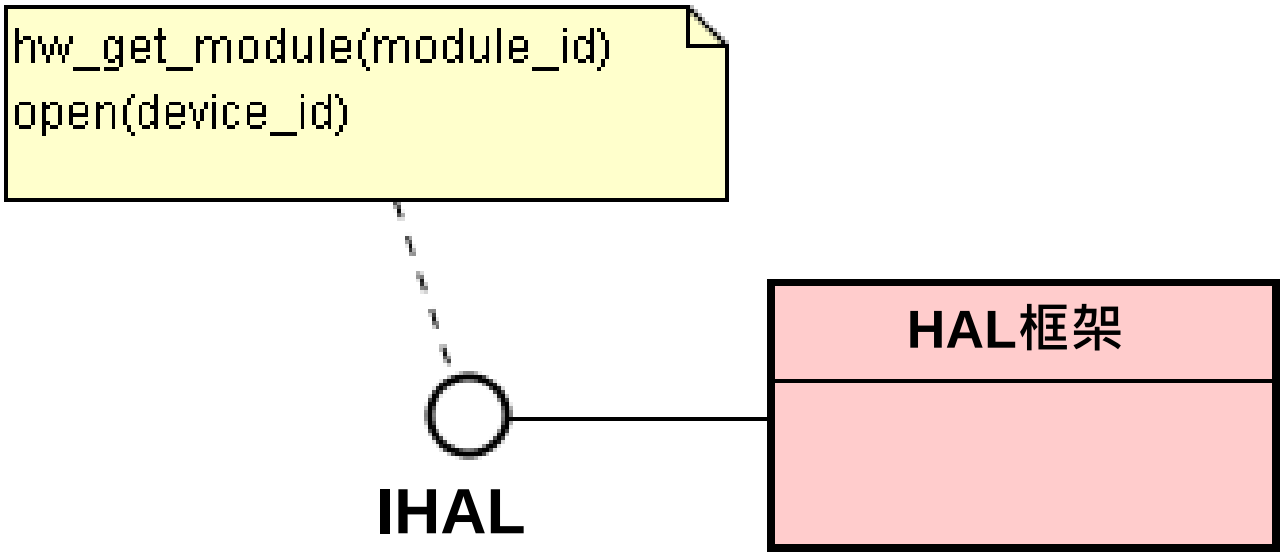
OOPC与HAL的美妙结合(c)

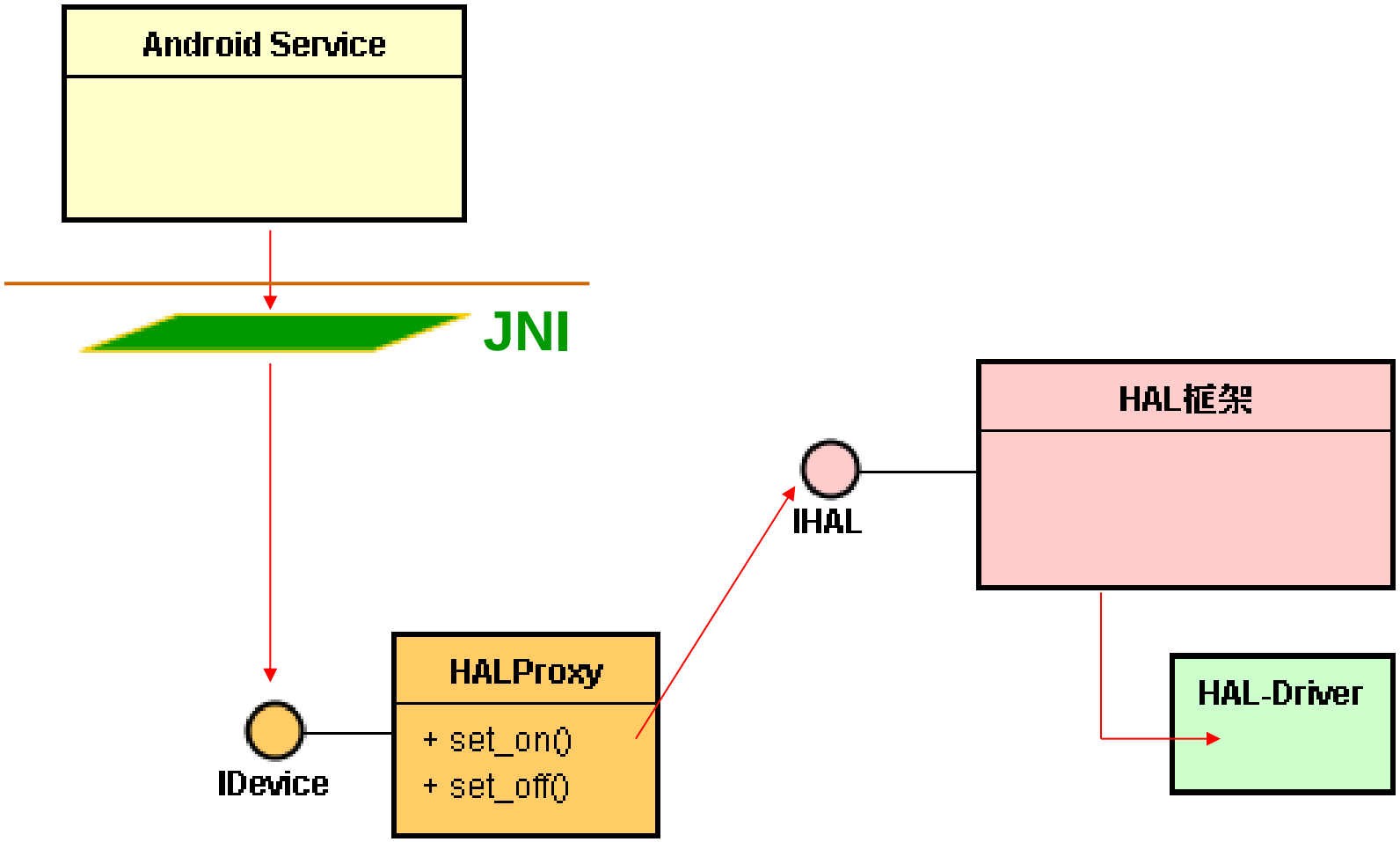
By 高煥堂

4、演练：以OOPC撰写HAL的Proxy类

设计HAL接口的Proxy类

- Android的原来HAL架构接口如下：

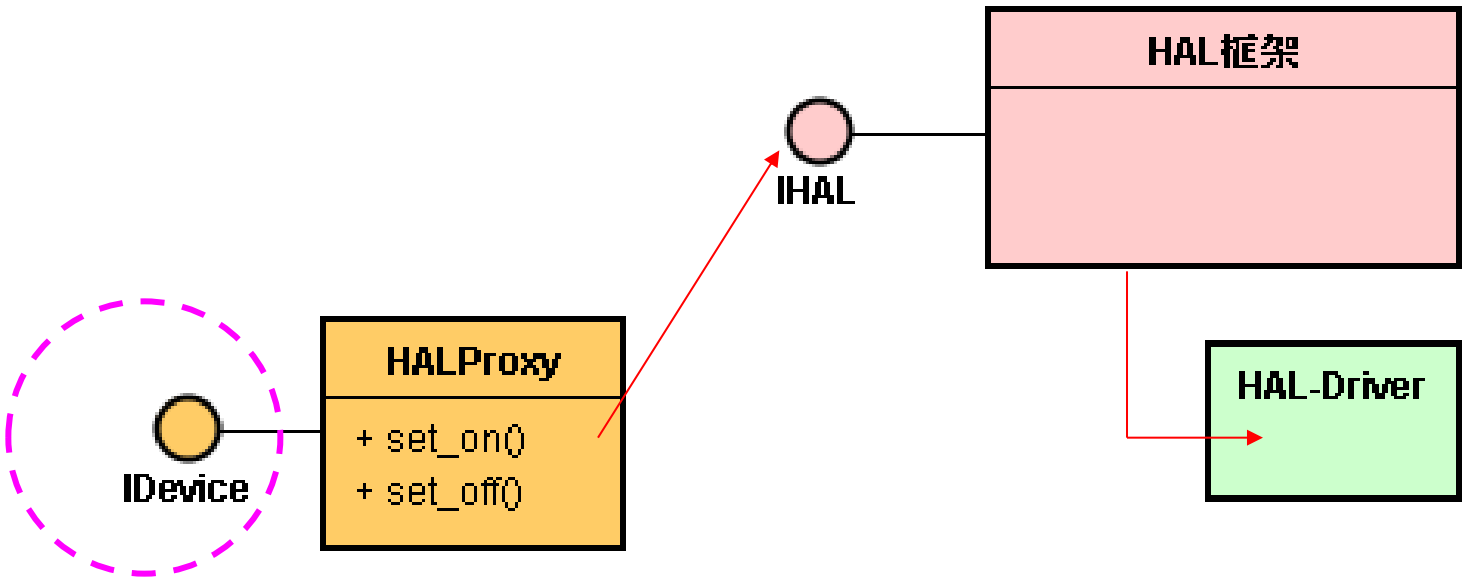



Android Service
JNI
HAL框架
IHAL
HALProxy
+ set_on()
+ set_off()
IDevice
HAL-Driver

撰写Proxy类的代码

- 这个HALProxy类，其实就是Proxy-Stub设计模式里的Proxy类。
- 这把IHAL接口包装起来，而呈现新的接口。此HALProxy类的定义如下：

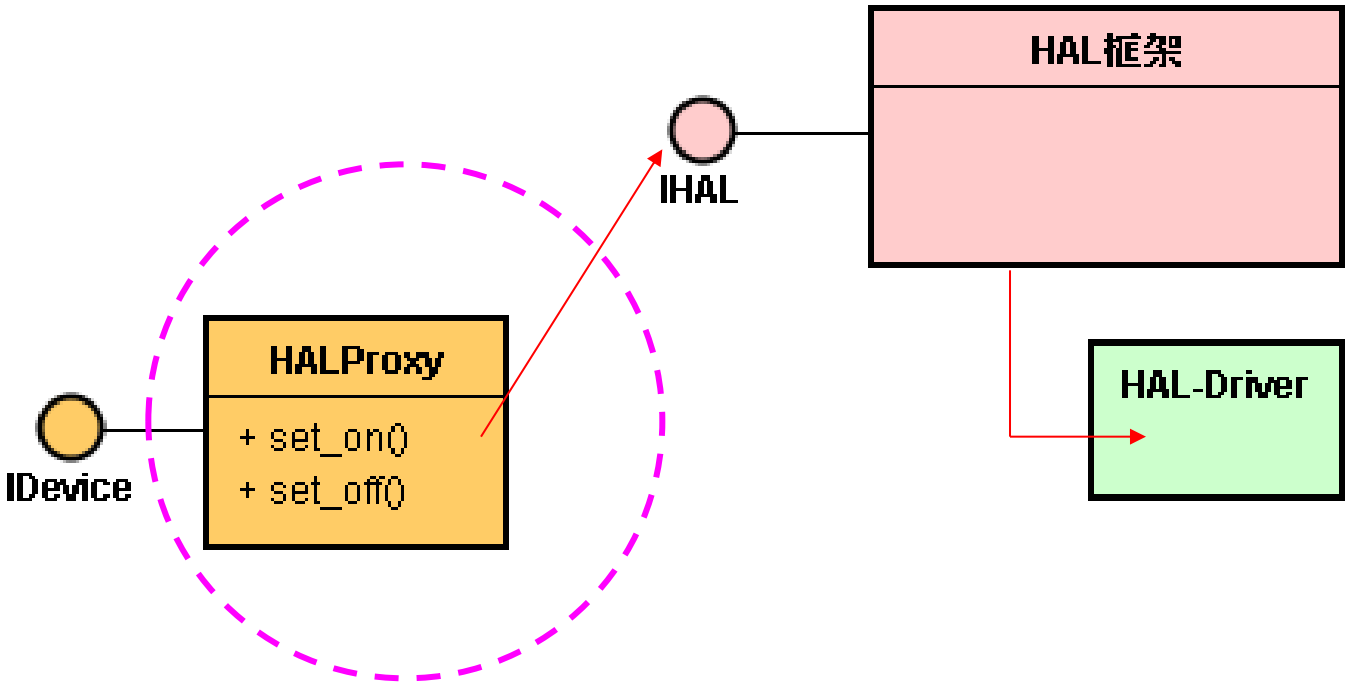
定义IDevice接口



定义IDevice接口

```
/*  ihal.h  */
#ifndef IDEVICE_H
#define IDEVICE_H
INTERFACE( IDevice )
  {
        void (*set_on)(void*);
        void (*set_off)(void*);
  }
#endif
```

定义HALProxy类(并实现IDevice接口)



```
/* HALProxy.h  */
#ifndef HALPROXY_H
#define HALPROXY_H
#include <misoo/lw_oopc.h>
#include <stdio.h>
#define LED_HARDWARE_MODULE_ID "led"
#define LED_HARDWARE_DEVICE_ID "led_dev   ″
```

```
 CLASS(HALProxy)
   {
     IMPLEMENTS( IDevice );
       struct hw_module_t* module;
       struct hw_device_t* device;
       char* module_id;
       char* device_id;

       void (*setModuleID)(HALProxy* thiz, const char* id);
       void (*setDeviceID)(HALProxy* thiz, const char* id);
       int (*getModule)(HALProxy* thiz);
       int (*getDevice)(HALProxy* thiz);
};
#endif
```

撰写HALProxy的实现代码

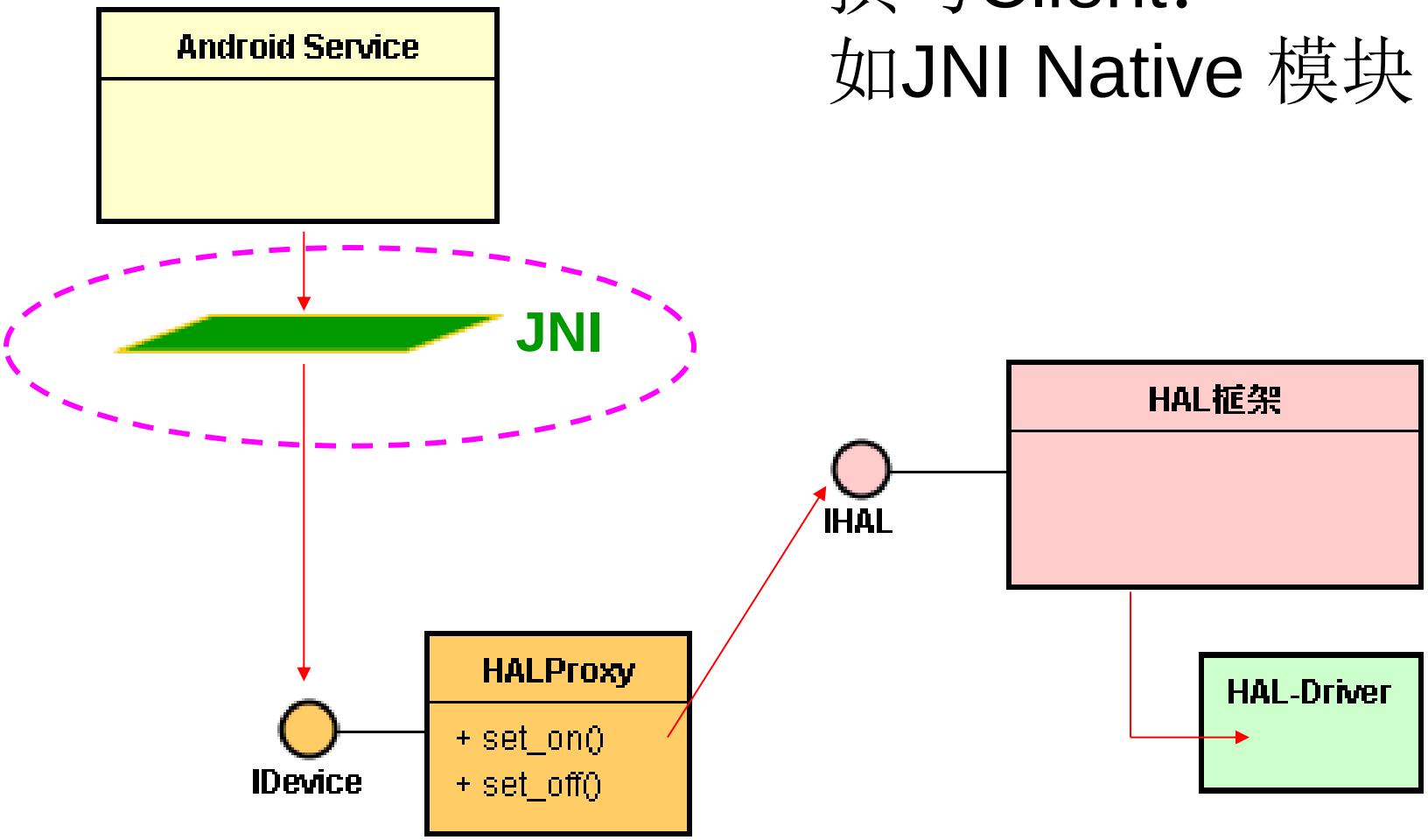
```
/*  HALProxy.c  */
#include "HALProxy.h"
#include <hardware/hardware.h>
#include <stdio.h>
static void setModuleID(HALProxy* thiz, const char* id)
    {
            thiz->module_id = id;
    }
static void setDeviceID(HALProxy* thiz, const char* id)
    {
            thiz->device_id = id;
    }
```

```
static int getModule(HALProxy* thiz)
{
        int t = hw_get_module( module_id,
                        (const struct hw_module_t**)(&module));
        if (t != 0 ) {
                printf("Error : hw_get_module = -1\n");   return 0;
        }
        return 1;
}
static int getDevice(HALProxy* thiz)
{
        thiz->module->methods->open(
            module, device_id, (struct hw_device_t**)(&device));
        return 1;
}
```

```
static void led_set_on(HALProxy* thiz)
{
        printf("set_on ...\n");
}
static void led_set_off(HALProxy* thiz)
{
        printf("set_off ...\n");
}
CTOR(HALProxy)
        FUNCTION_SETTING(setModuleID, setModuleID);
        FUNCTION_SETTING(setDeviceID, setDeviceID);
        FUNCTION_SETTING(getModule, getModule);
        FUNCTION_SETTING(getDevice, getDevice);
        FUNCTION_SETTING(IDevice.set_on, led_set_on);
        FUNCTION_SETTING(IDevice.set_off, led_set_off);
END_CTOR
```

- 撰写Client：如JNI Native 模块

- 撰写Client：
  如JNI Native 模块



```
HALProxy* proxy;
IDevice* idev;
static jint led_init(JNIEnv *env, jclass clazz)
{
    proxy = (HALProxy*)HALProxyNew();
    idev = (IDevice*)proxy;
    proxy->setModuleID(proxy,
                    LED_HARDWARE_MODULE_ID);
    proxy->getModule(proxy);
    proxy->setDeviceID(proxy,
                    LED_HARDWARE_DEVICE_ID) {
    proxy->getDevice(proxy);
    return 1;
}
```

```
static jint led_setOn(JNIEnv* env, jobject thiz) {
        idev->set_on(idev);
            return 1;
}

static jint led_setOff(JNIEnv* env, jobject thiz) {
        idev->set_off(idev);
            return 1;
}
```

- 有了 Proxy类，对于JNI Native模块的代码，有何影响呢?

~ Continued ~