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

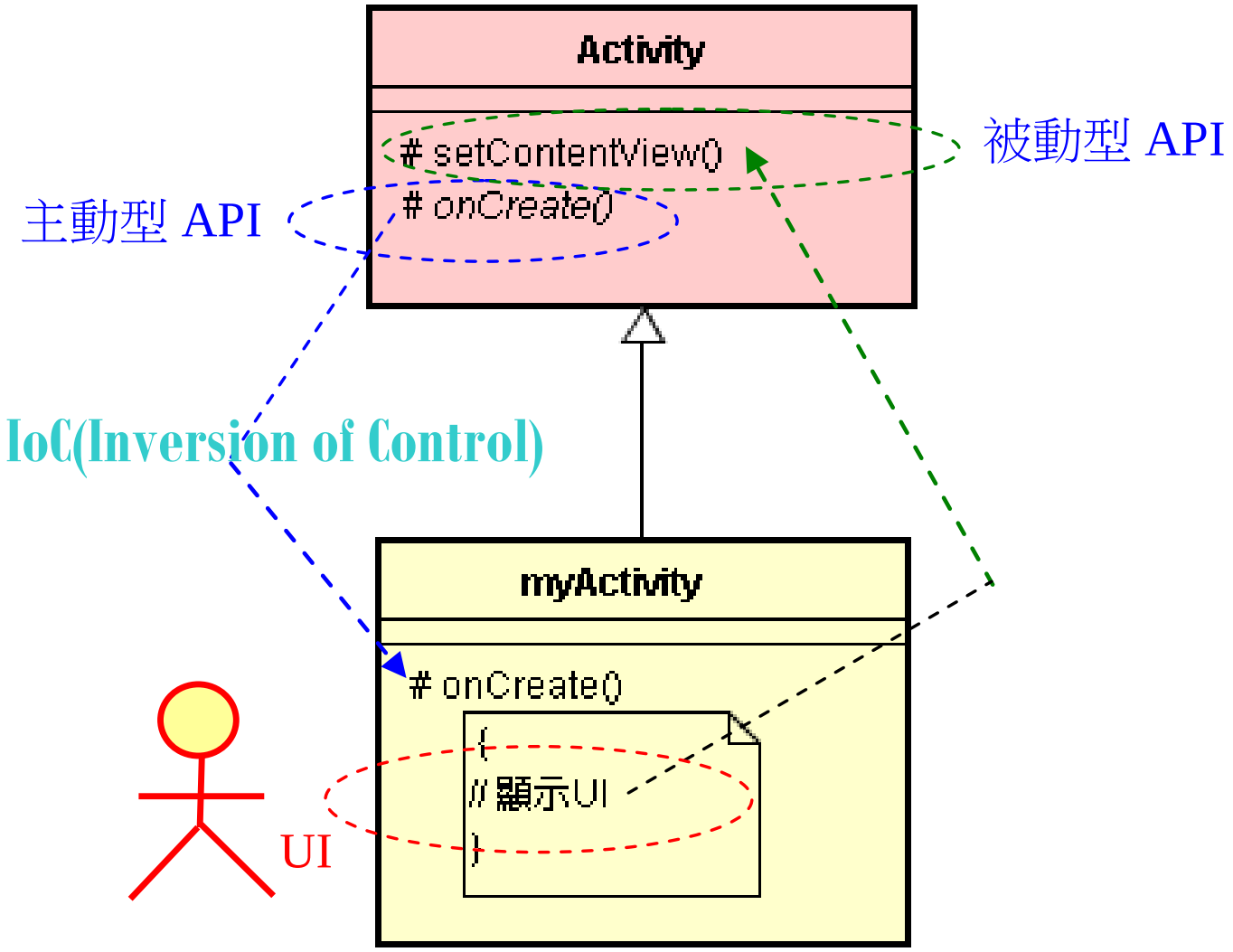
A01_d

复习基本OOP技术(d)

By 高煥堂

8、主动型 vs. 被动型API

卡榫函数实现API


Activity
setContentView()
onCreate()
被動型 API
主動型 API
IoC(Inversion of Control)
myActivity
# onCreate()
{
// 顯示UI
}
UI

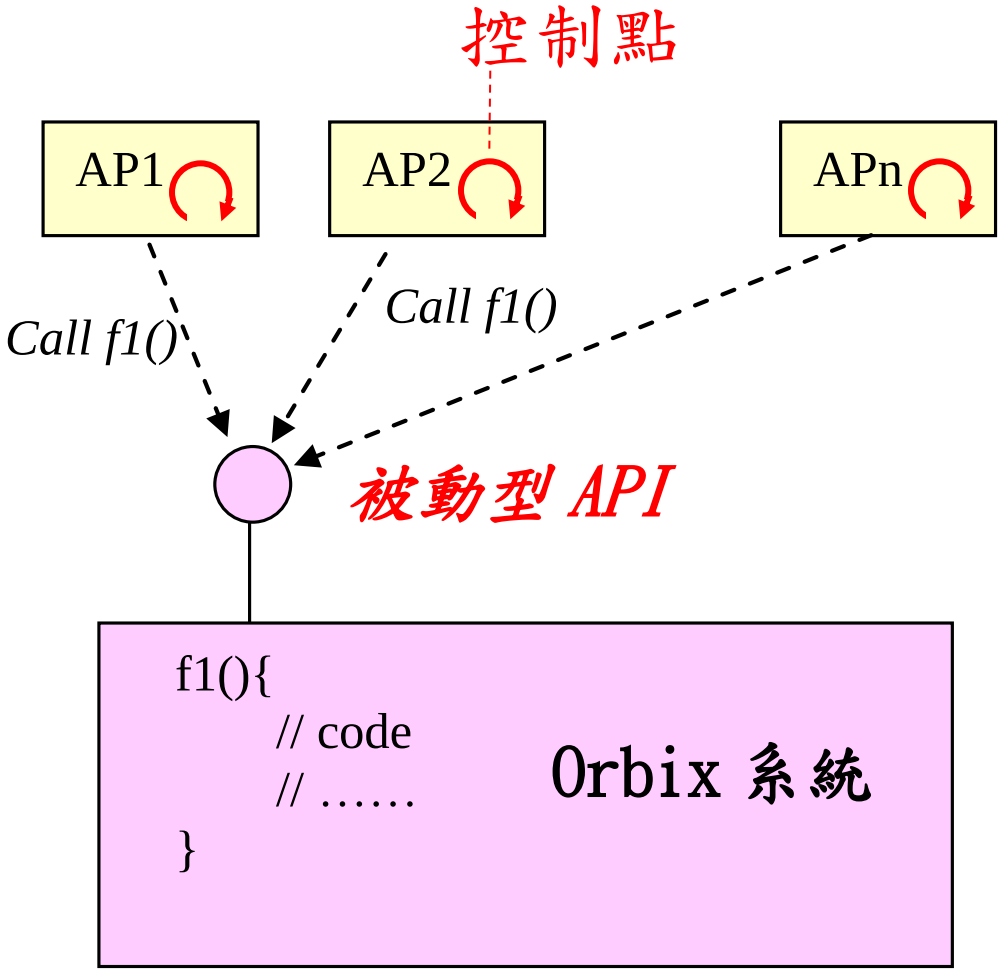
API的分类

- API这个名词，有3个密切关联的动词：
  定义(Define)
  实作(Implement)
  呼叫(Invoke or Call)

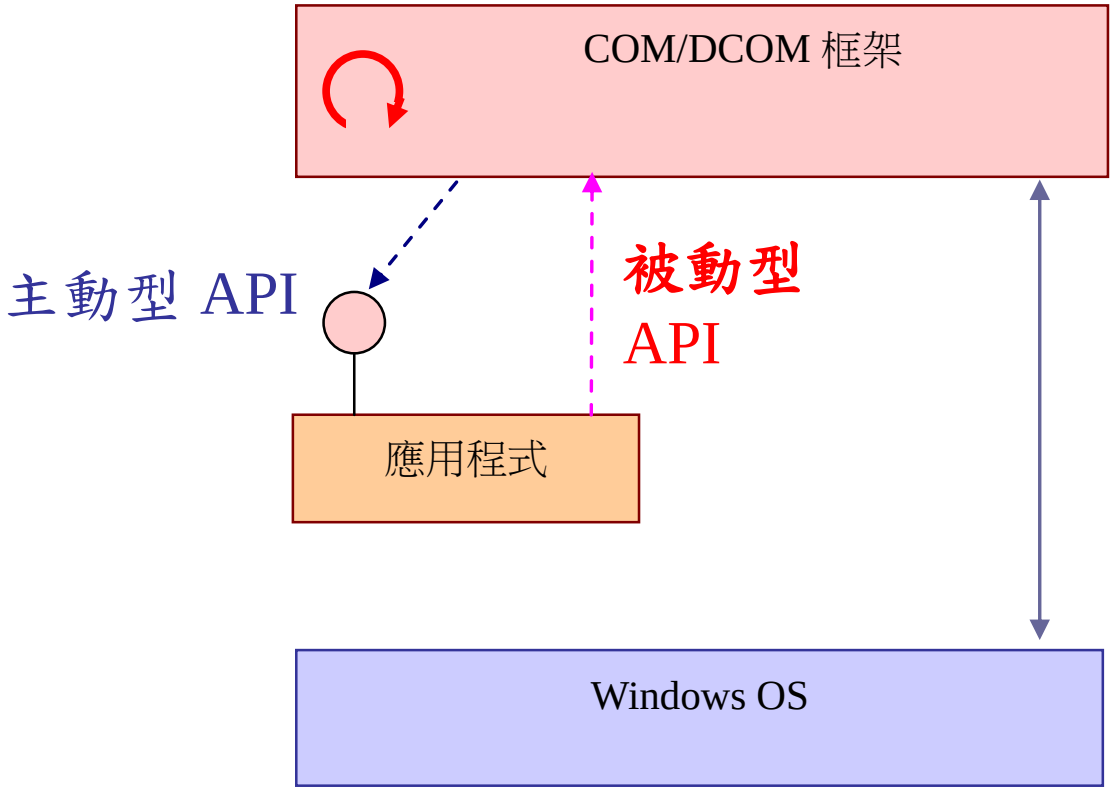
- 根据这3个角度，可将API区分为「主动型」与「被动型」两种。

API类型	
被动型API	•我(即强龙)定义API •且我来实作API •而让对方(即地头蛇)来呼叫我
主动型API	•我定义API •由地头蛇来遵循、实作API •让我来呼叫对方

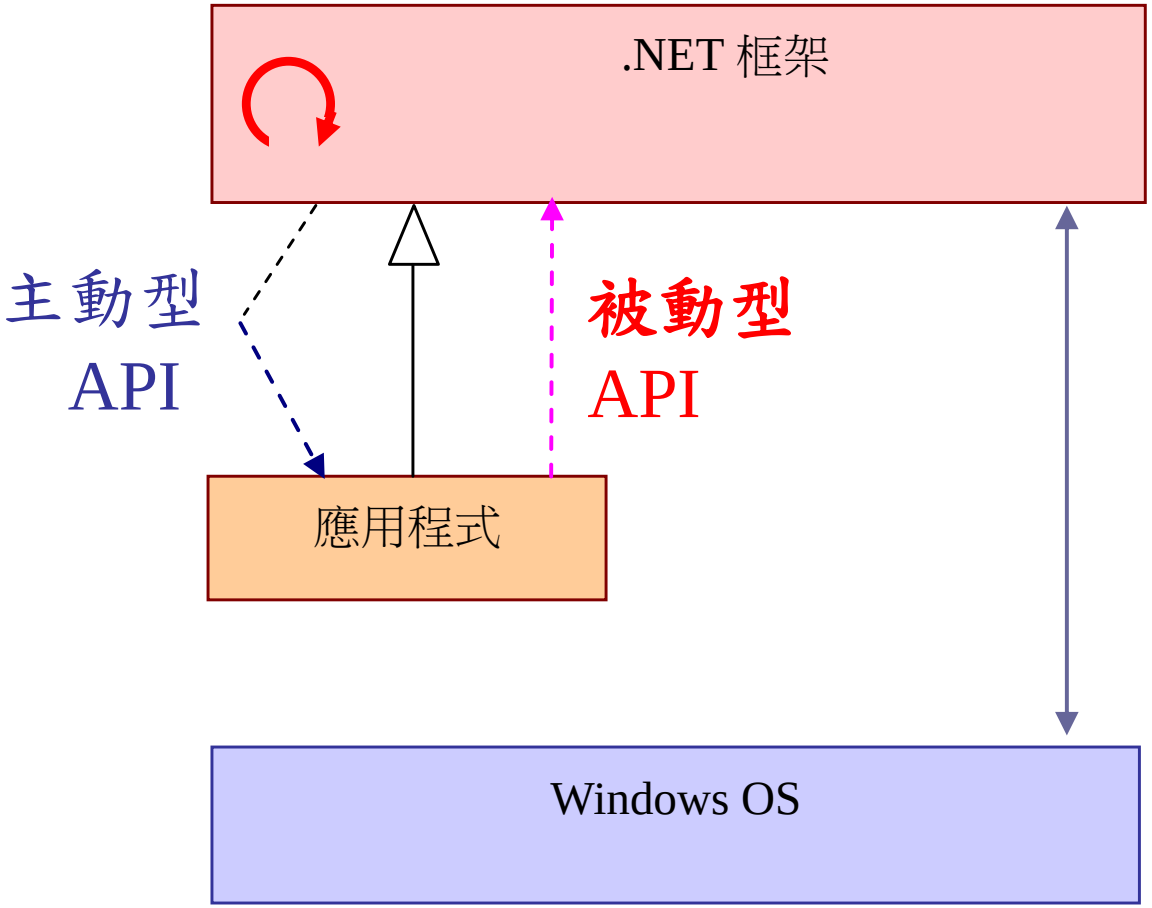
回顧：1990年代初的CORBA和Orbix



1995年的COM/DCOM



2001年的.NET



API =>控制力

◎大家都知道，接口(Interface)是双方接触的地方，也是双方势力或地盘的界线。

◎谁拥用接口的制定权，谁就掌握控制点，就能获得较大的主动权(或称为主导权)，而位居强龙地位；而另一方则处于被动地位，成为弱势的一方，扮演地头蛇角色。

以清朝时期的马关条约为例

<<马关条约>>

第二条：中国把辽东半岛、台湾、澎湖诸岛之权及该地城垒、兵器制造所及国有物永远割给日本。

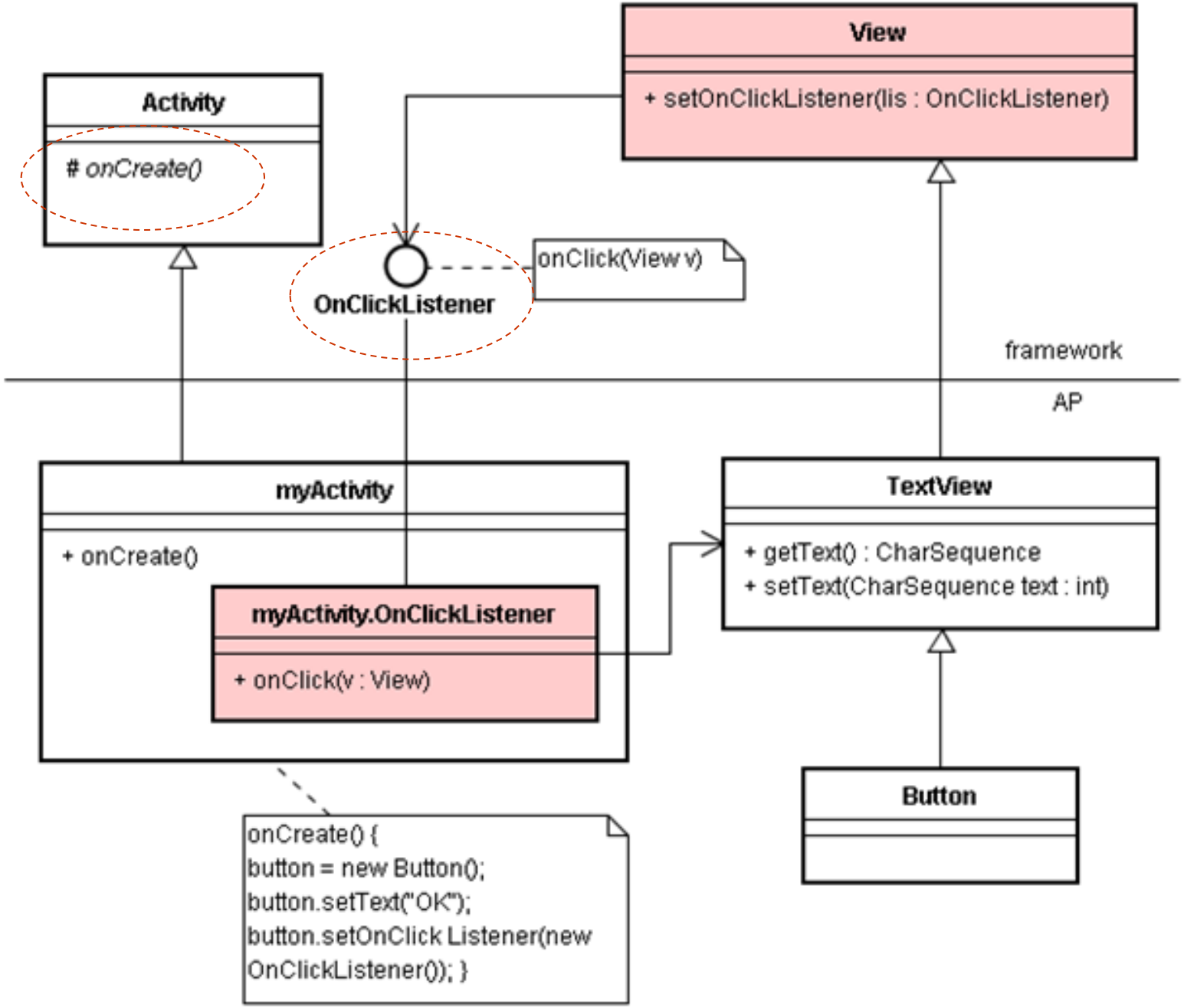
第四条：中国支付日本赔款2亿两白银。

API的范例 (Android)

```
// myActivity.java
public class myActivity extends Activity {
    @Override public void onCreate(Bundle icicle) {
      super.onCreate(icicle);
        setContentView(R.layout.main);
        Button button = new Button(this);
        button.setText("OK");
        button.setBackgroundResource(R.drawable.heart);
        button.setOnClickListener(this);
        setContentView(button);
    }
```

```
OnClickListener clickListener = new OnClickListener() {
        @Override public void onClick(View v) {
            String name = ((Button)v).getText().toString();
            setTitle( name + " button clicked");
    }};
}
```

API


View
+ setOnClickListener(lis : OnClickListener)
Activity
onCreate()
OnClickListener
onClick(View v)
framework
AP
myActivity
+ onCreate()
myActivity.OnClickListener
+ onClick(v : View)
TextView
+ getText() : CharSequence
+ setText(CharSequence text : int)
Button
onCreate() {
button = new Button();
button.setText("OK");
button.setOnClick Listener(new
OnClickListener()); }

9、结语&复习：接口与类(别)

接口的表示

- 在OOP里，将接口定义为一种特殊的类别(Class)。

- 在C++裡，類別包括3種：

1. 一般(具象)類別
   -- 所有函數都是具象(內有指令)

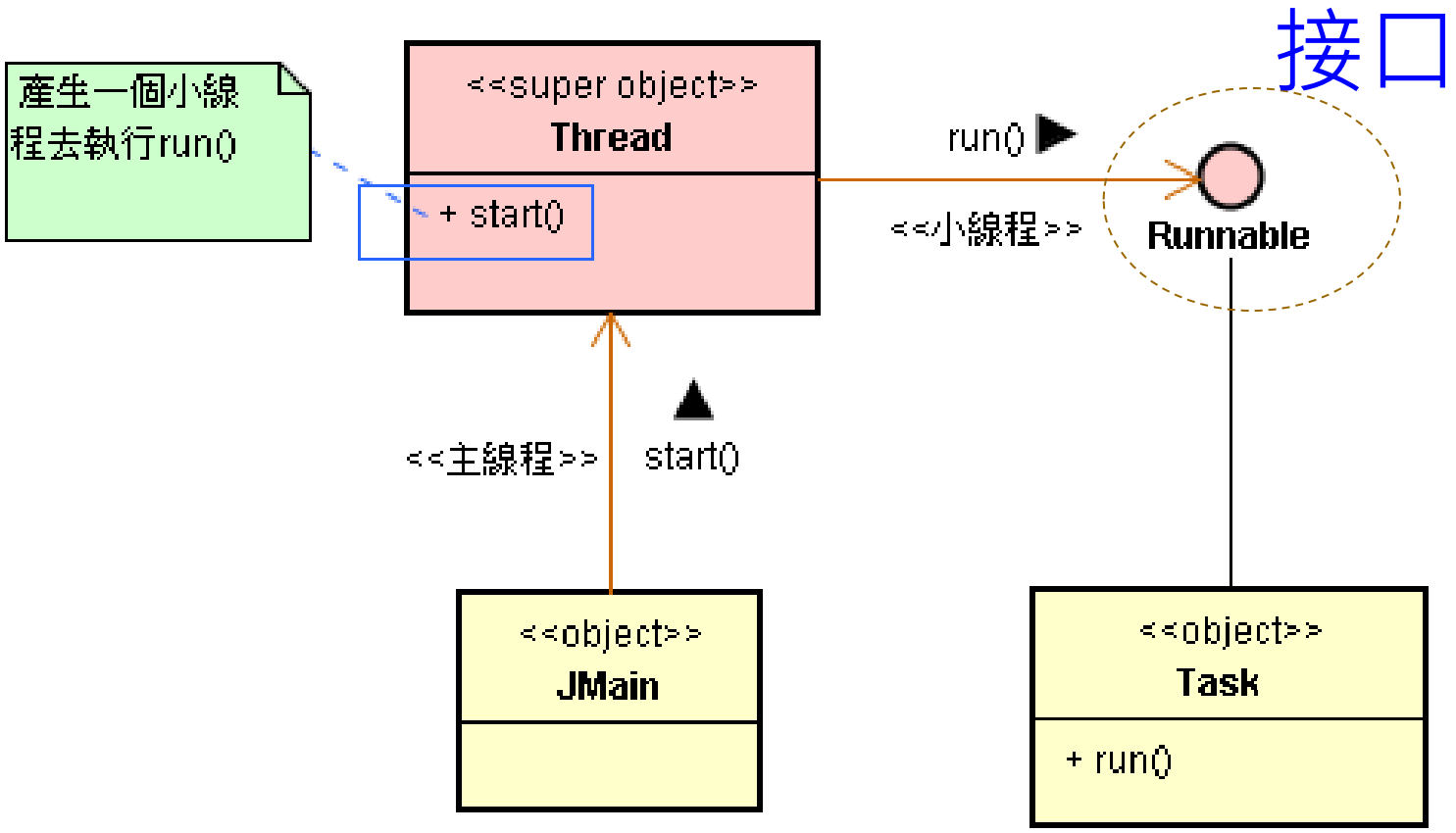
2. 抽象(abstract)類別
   -- 有一個或多個函數是抽象的(內無指令)

3. 純粹抽象(pure abstract)類別
   -- 所有函數都是抽象的

- 在Java里，将上述的「 纯粹抽象类别」称为接口(Interface)

- 在UML里，以圆圈()来表示接口。

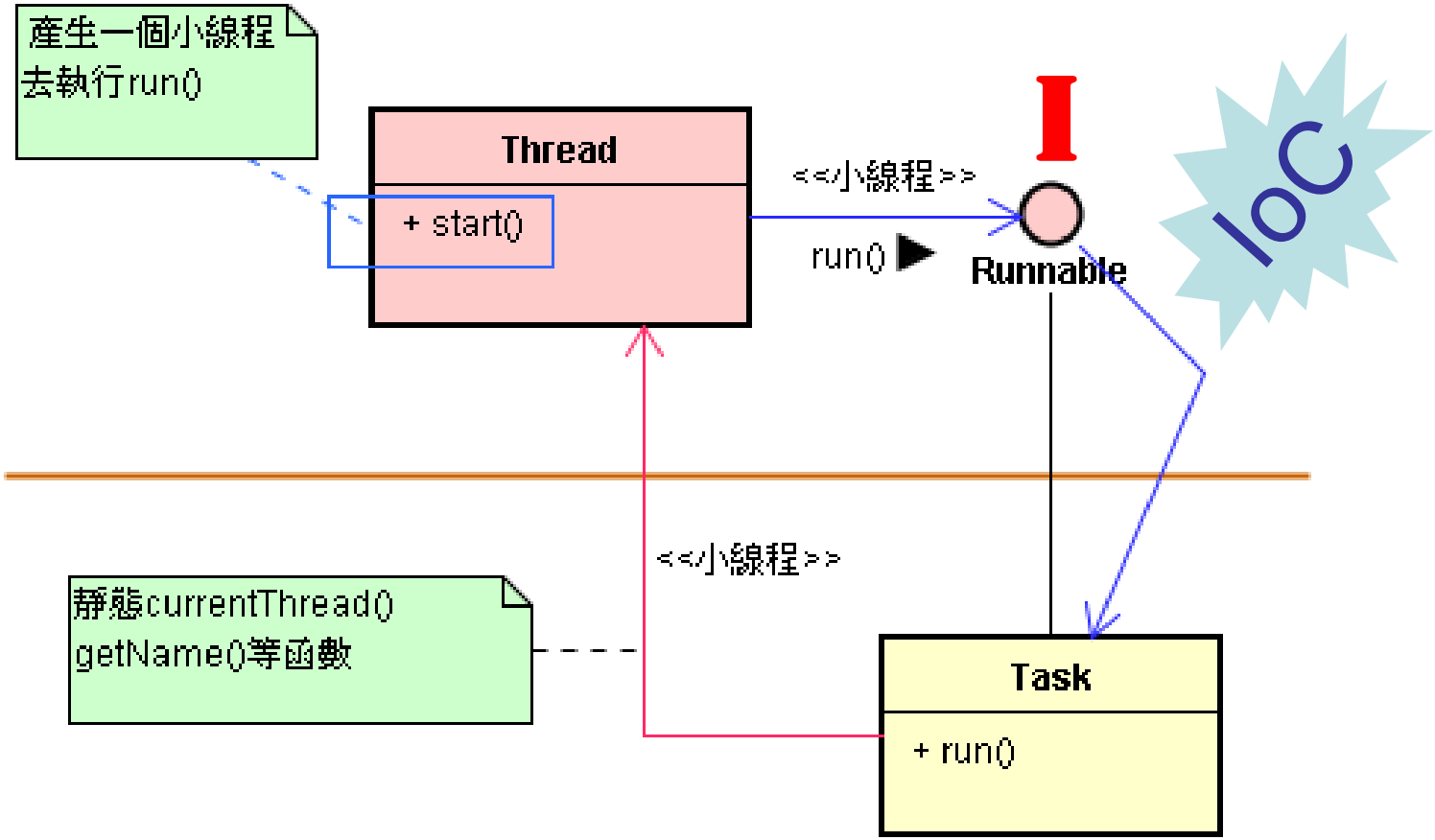
Java的接口表示

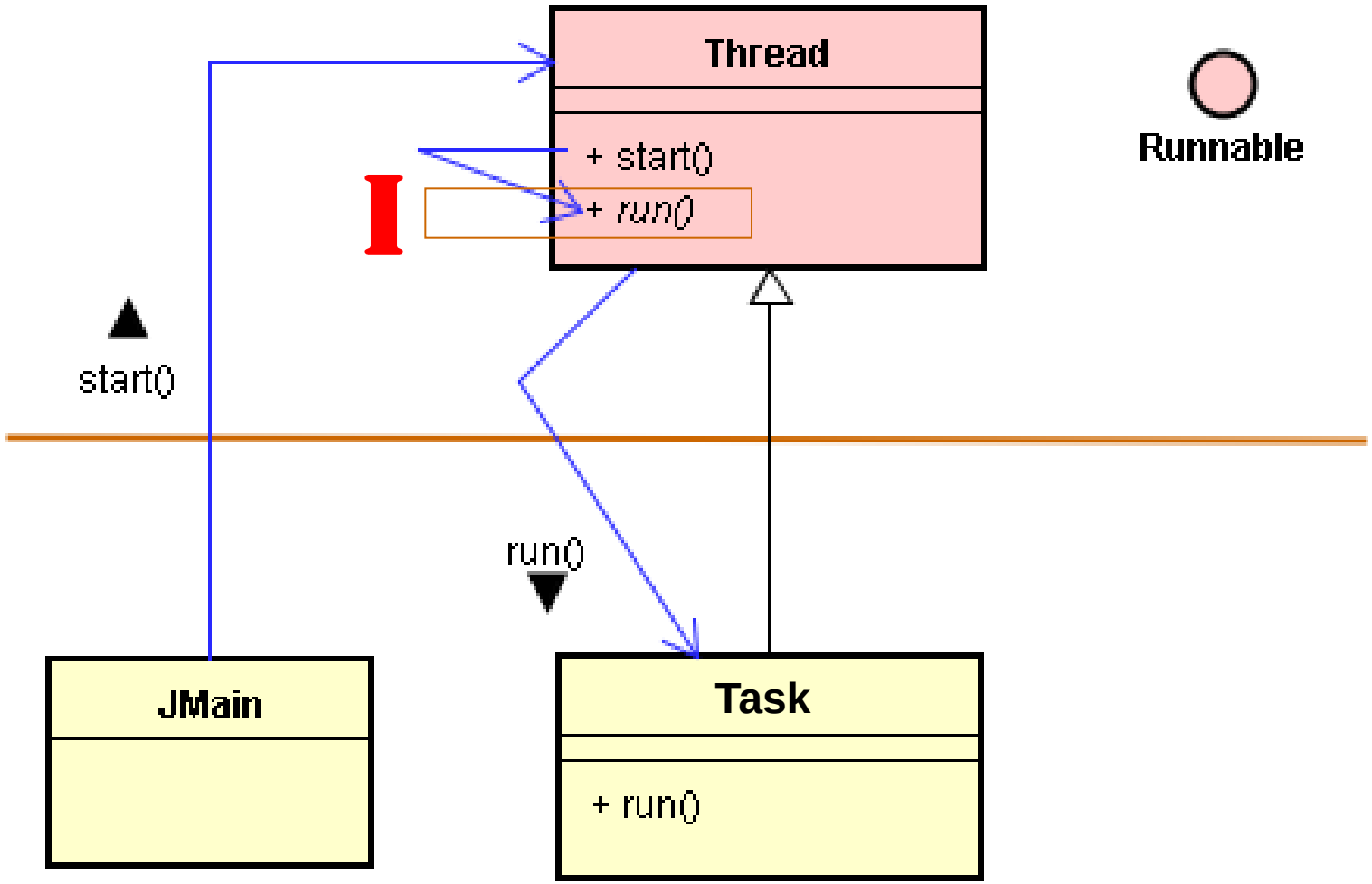

接口
產生一個小線程去執行run()
<<super object>>
Thread
+ start()
run()
<<小線程>>
Runnable
<<主線程>>
start()
<<object>>
JMain
<<object>>
Task
+ run()

落实为代码：

```
class Task implements Nunnable{
        public void run() {
                int sum = 0;
                for (int i = 0; i <= 100; i++)        sum += i;
                System.out.println("Result: " + sum);
        }}

public class JMain {
        public static void main(String[] args) {
          Thread t = new Thread( new Tasks());
          t.start();
          System.out.println("Waiting...");
        }}
```


產生一個小線程
去執行run()
Thread
+ start()
<<小線程>>
run()
Runnable
I
IoC
<<小線程>>
靜態currentThread()
getName()等函數
Task
+ run()


Thread
+ start()
+ run()
Runnable
start()
run()
JMain
Task
+ run()

Thanks...

高煥堂