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

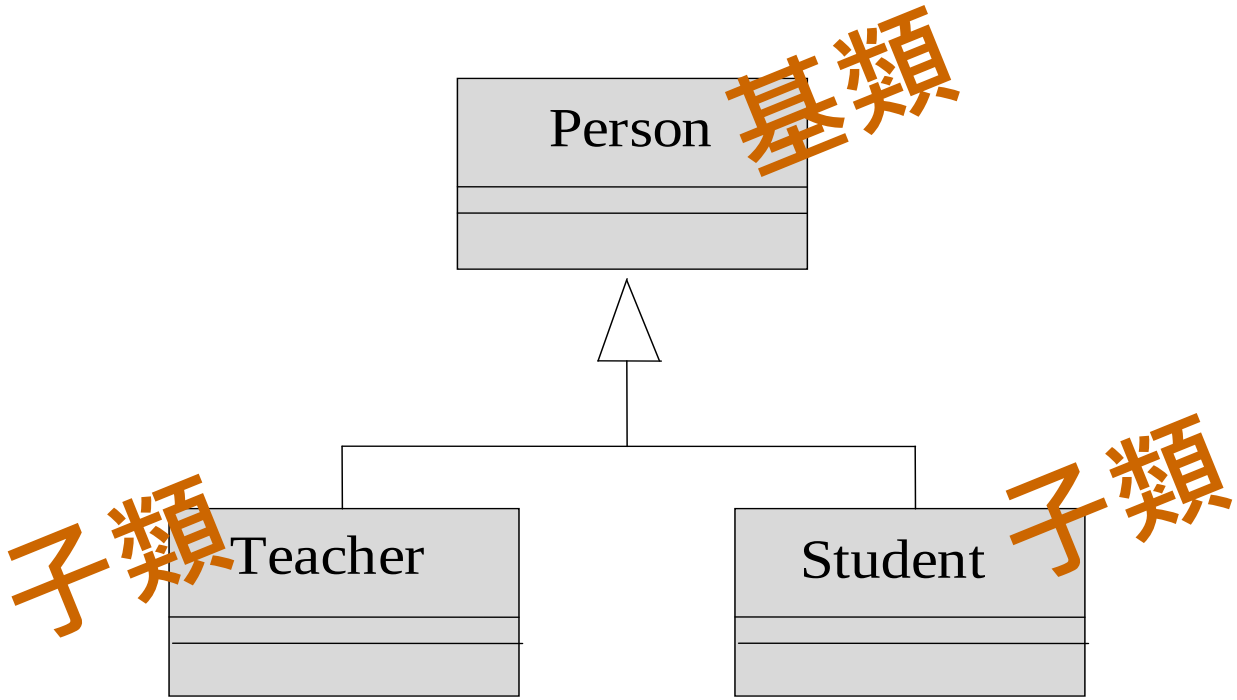
A01_b

复习基本OOP技术(b)

By 高煥堂

4、<基类/子类>结构用途(一)：表达继承

- 对众多对象加以分门别类，就可形成一个类继承体系。例如对学校人员加以分门别类，而得出类继承体系，如下图：



软件代码的表达是：

Step-1. 定义基类。如：

```
class Person {
        //.....
}
```

Step-2. 定义衍生类(即子类)。如：

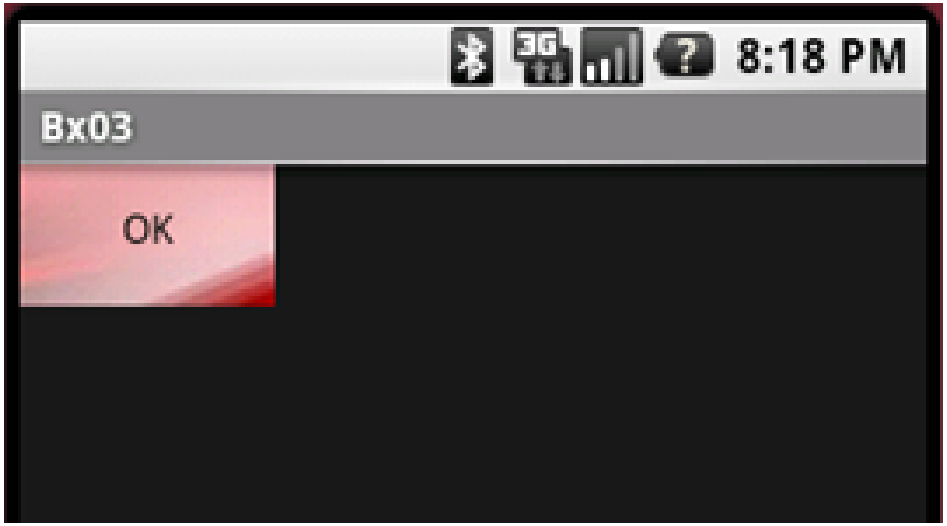
```
class Teacher extends Person {
        //.......
    }
class Student extends Person {
        //.......
    }
```

代码范例之一

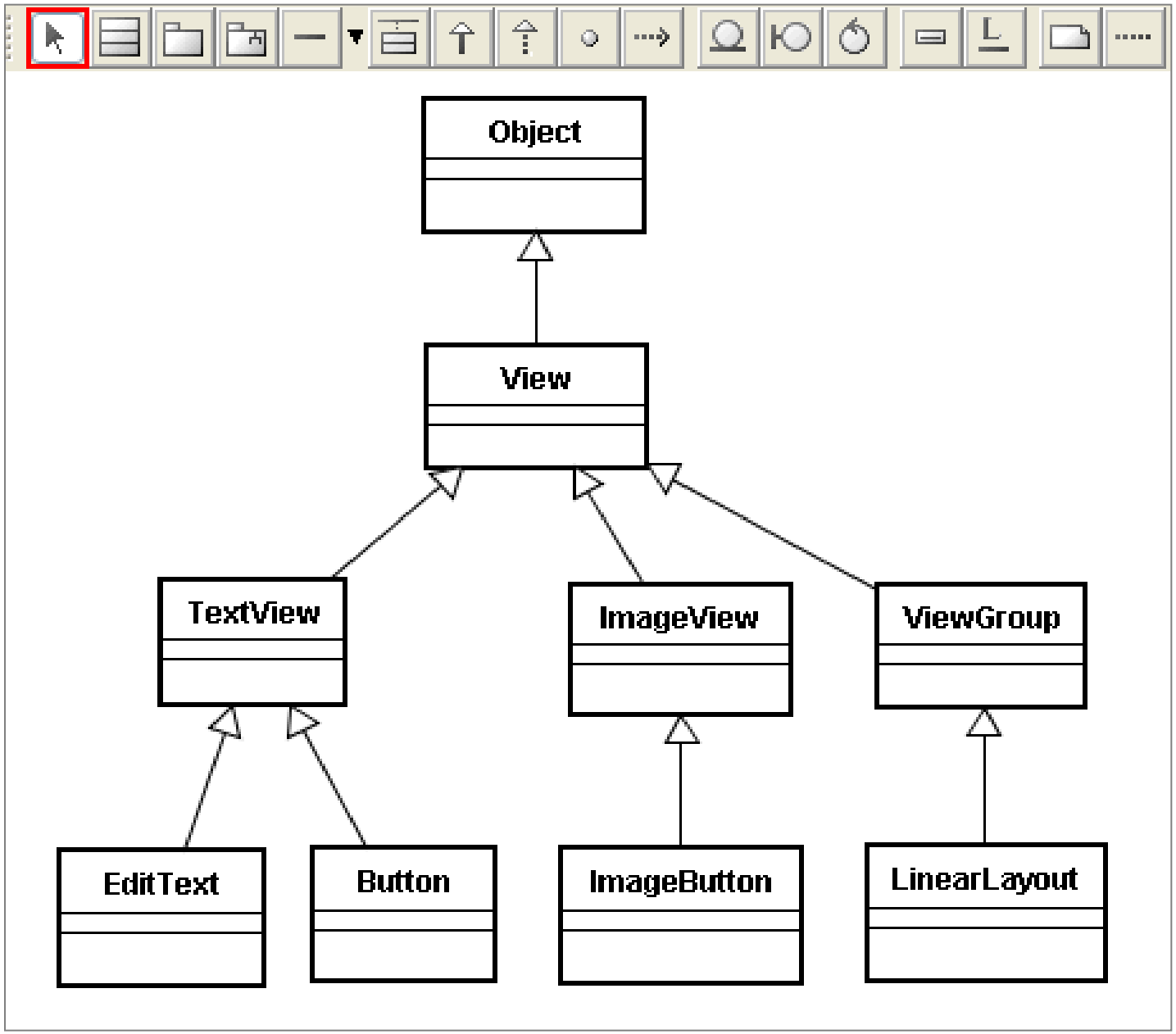
```
class Person {
    private String name;
    private int age;
    public void set_value(String na, int a)
          {   name = na;  age = a;        }
    public int birth_year()
          {  return 2009 - age;    }
    public void display()
         {  System.out.println( name + “, " + age);   }
}
```

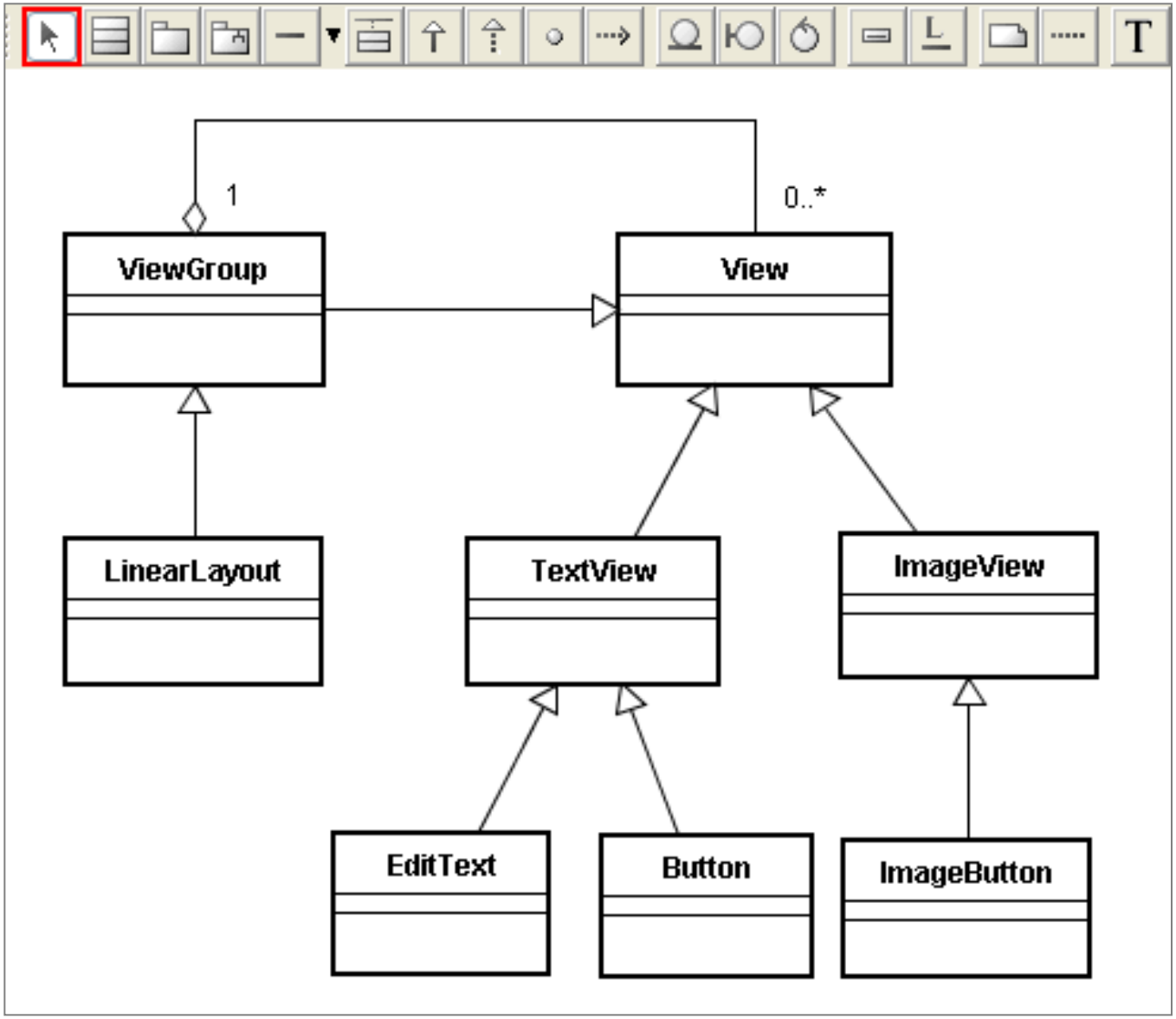
```
class Teacher extends Person {
    private double salary;
    public void teacher_set_value(String na, int a,
    double sa) {
        set_value(na, a); salary = sa;
    }
    public void print() {
        display();
        System.out.println(salary);
}}
```

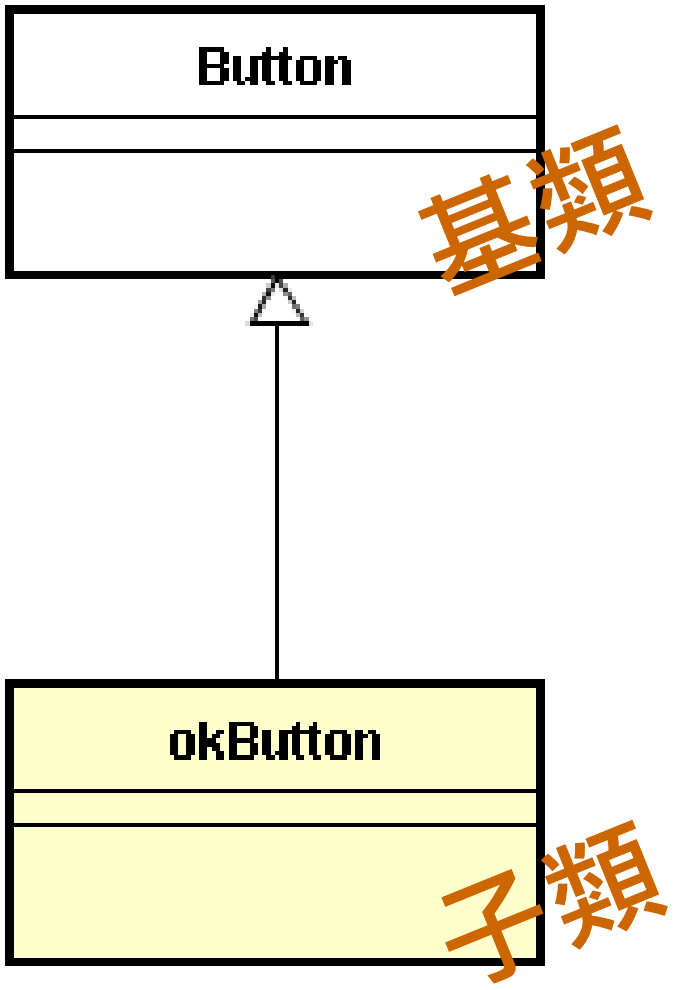
代码范例之二(Android)



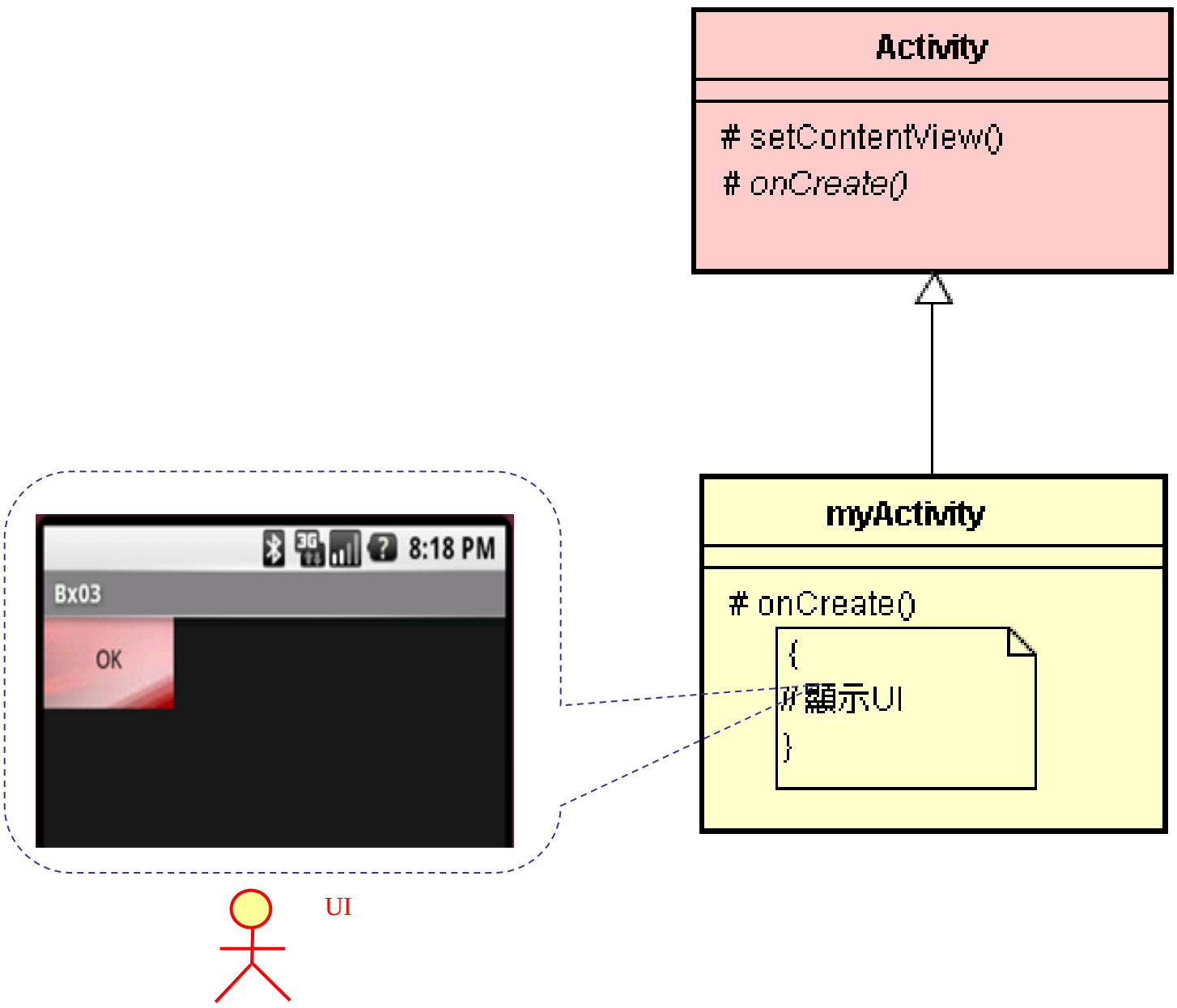
```
//android
public class okButton extends Button{
    int width, height;
    public okButton(Context ctx){
        super(ctx);
        super.setText("OK");
        super.setBackgroundResource(
                R.drawable.ok);
        width = 90;  height = 50;
    }
    public int get_width(){  return width;   }
    public int get_height(){  return height; }
}
```


Object
View
TextView
ImageView
ViewGroup
EditText
Button
ImageButton
LinearLayout


1
0..*
ViewGroup
View
LinearLayout
TextView
ImageView
EditText
Button
ImageButton


Button
基類
okButton
子類

```
// android
public class myActivity extends Activity {
    @Override public void onCreate(Bundle state) {
        super.onCreate(savedInstanceState);
        LinearLayout layout = new LinearLayout(this);
        layout.setOrientation(LinearLayout.VERTICAL);
        okButton ok_btn = new okButton(this);
        LinearLayout.LayoutParams param =
                    new LinearLayout.LayoutParams(
                        ok_btn.get_width(),
                        ok_btn.get_height());
         // ……
}}
```


Activity
setContentView()
onCreate()
myActivity
# onCreate()
{
//顯示UI
}
8:18 PM
Bx03
OK
UI

5、<基类/子类>结构用途(二)：表达组合

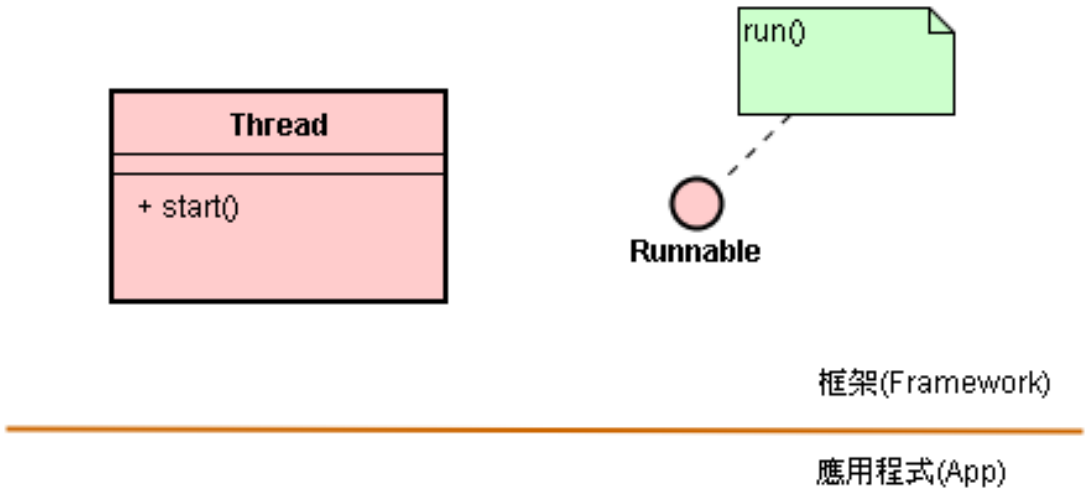
如何让Thread与Tasks组合起来

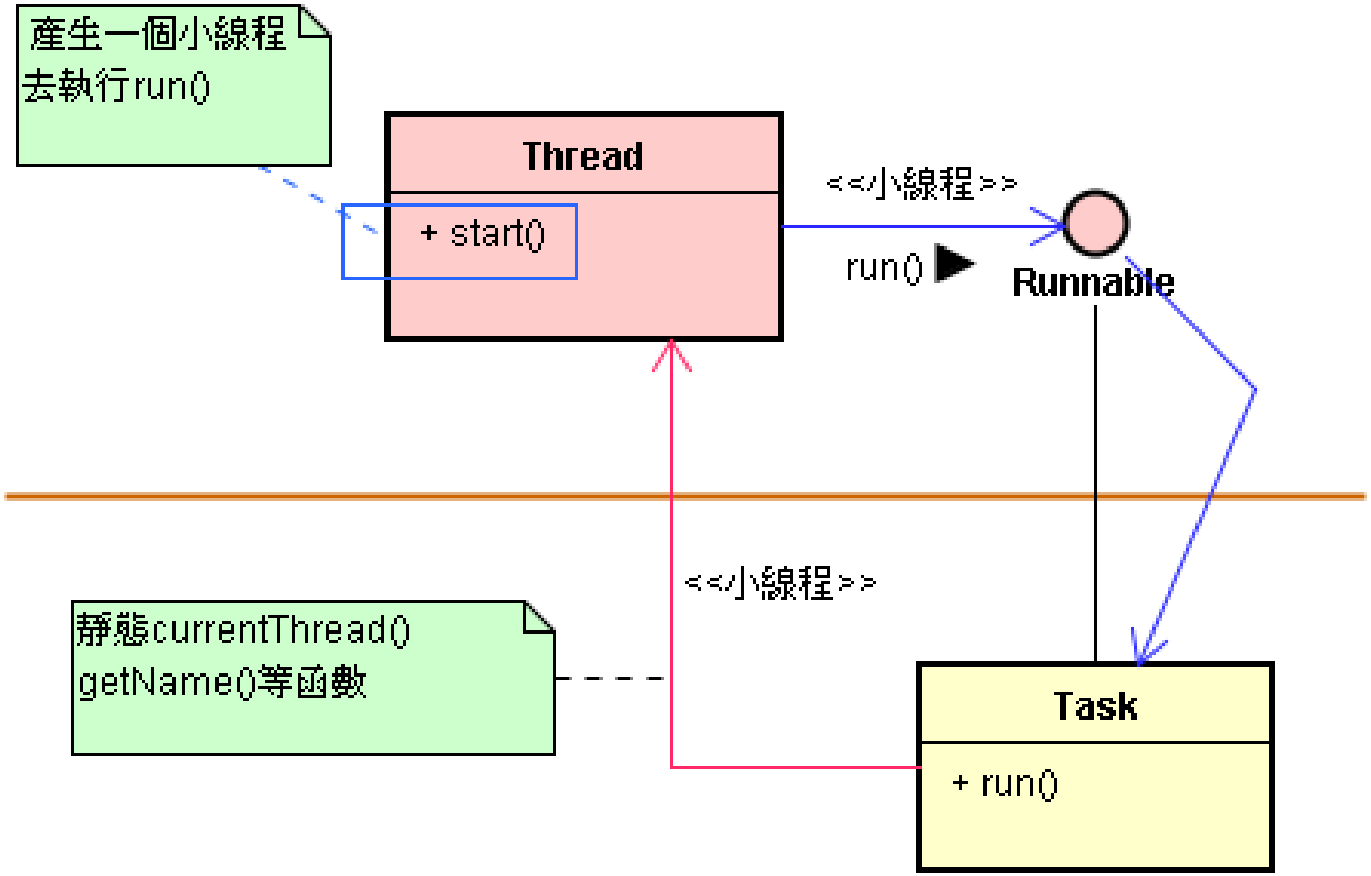
– 以创建小线程为例

Java代碼(一)

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

Java提供了一个Thread基类和一个Runnable接口；这两个元素就构成一个框架 。

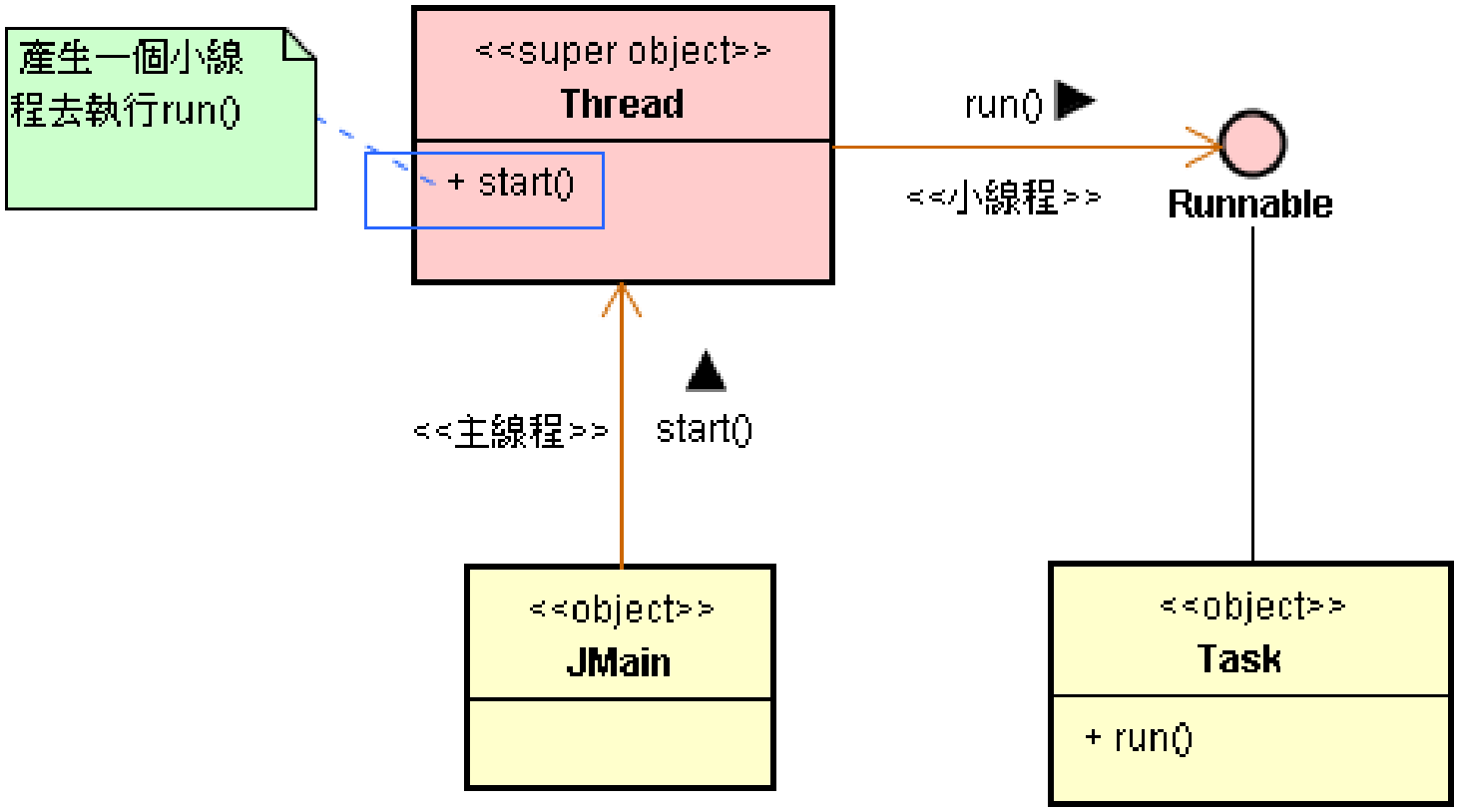



產生一個小線程
去執行run()
Thread
+ start()
<<小線程>>
run()
Runnable
<<小線程>>
靜態currentThread()
getName()等函數
Task
+ run()

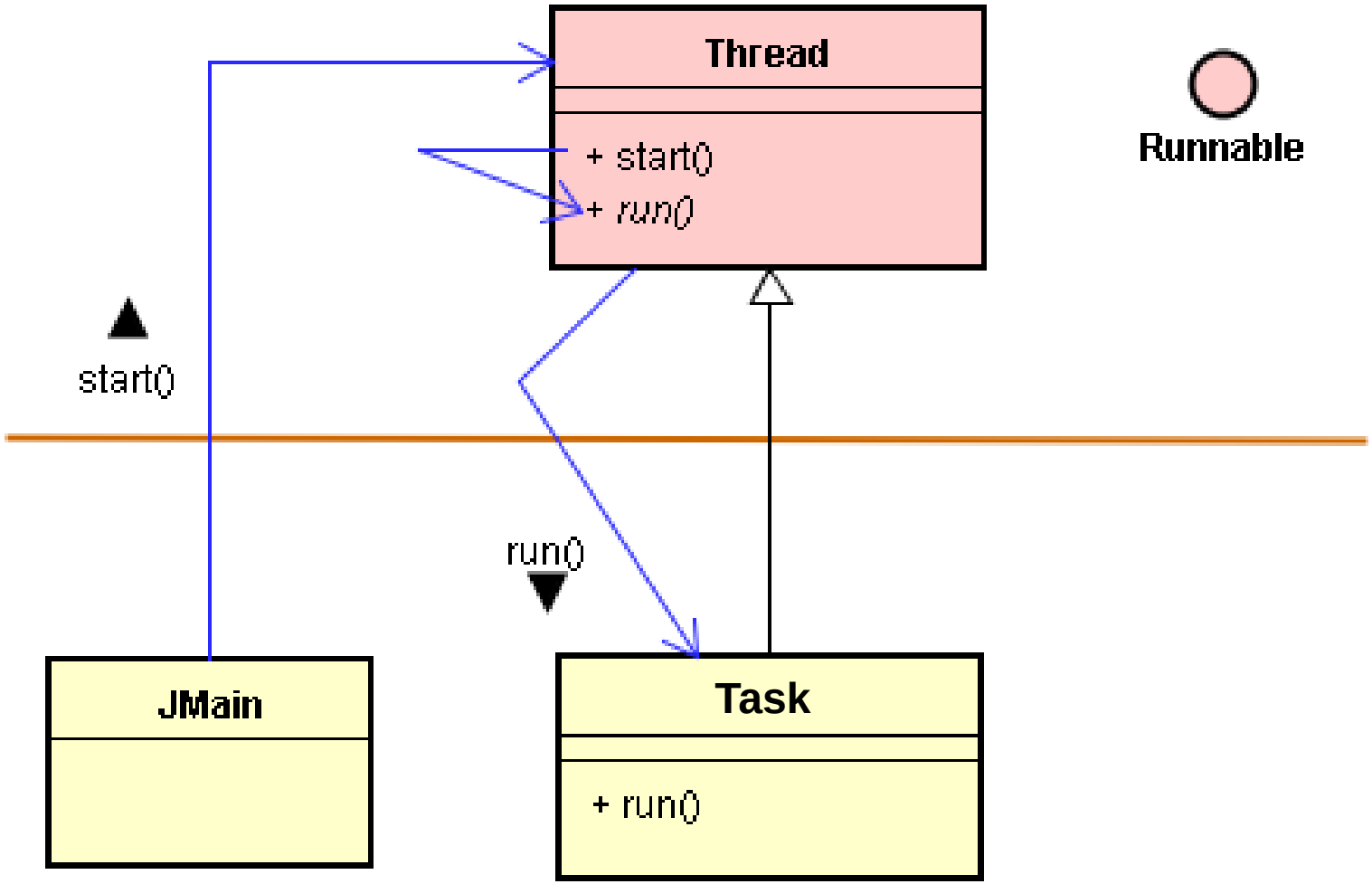
- 于此图里，框架的Thread基类会先诞生一个小线程，然后该小线程透过Runnable接口，呼叫(或执行)了Tasks类别的run()函数。

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

- 此时，main()先诞生一个Task类别的对象，并且诞生一个Thread基础的对象。接着，执行到下一个指令：t.start();
- 此时，main()就呼叫Thread的start()函数；这start()就产生一个小线程去执行run()函数。如下图：


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()

- 上图里的Runnable接口与Thread基类是可以合并起来的。也就是把run()函数写在Thread的子类里。如下图：

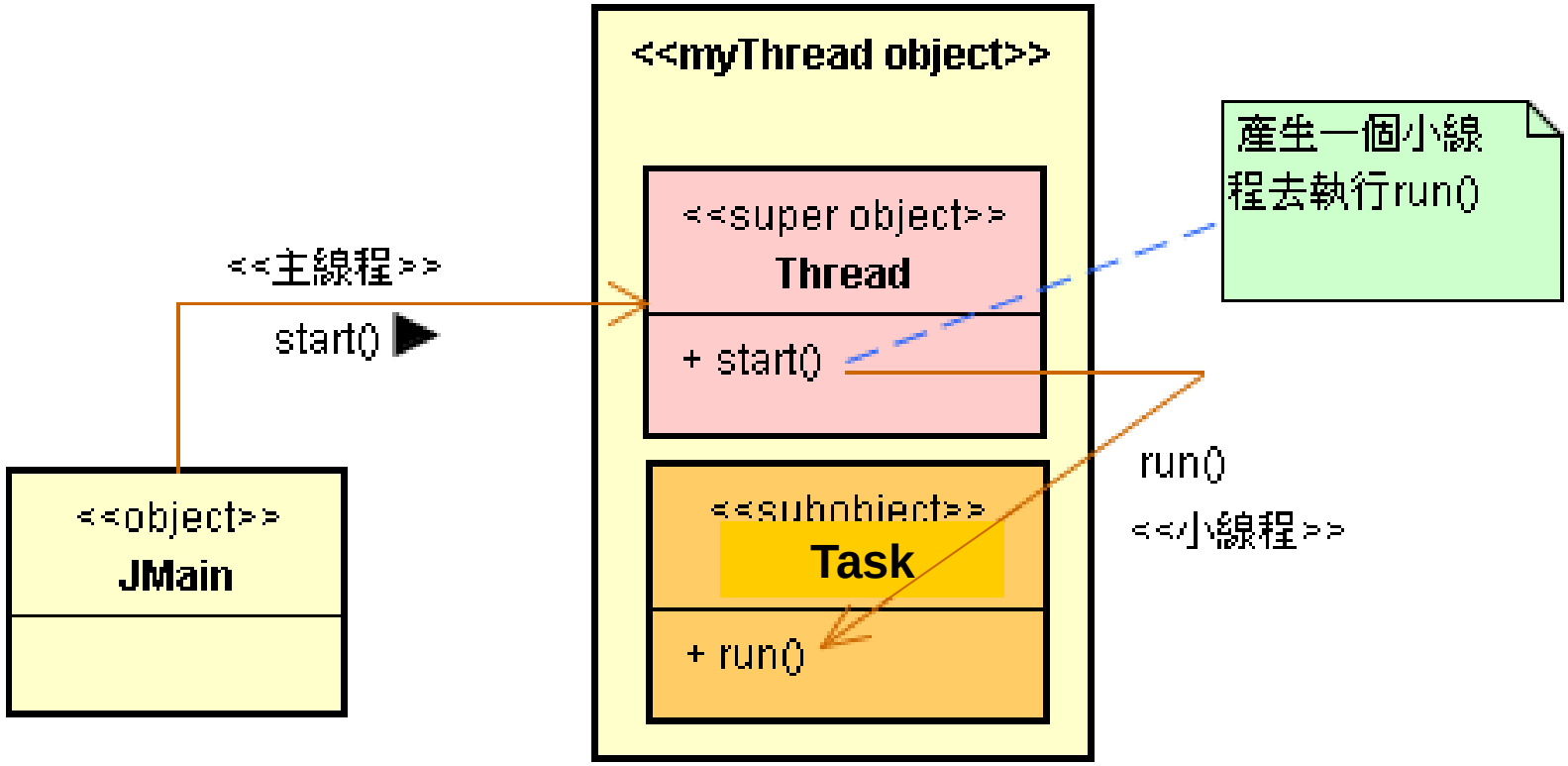

Thread
+ start()
+ run()
Runnable
start()
run()
JMain
Task
+ run()

Java代碼(二)

```
class Task extends Thread {
        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 Task();
         t.start();
         System.out.println("Waiting...");
    }
}
```

- 其诞生一个Task对象，并且由JMain呼叫Thread的start()函数。这start()就产生一个小线程去执行 myThread子类里的run()函数。上图是类别关系图，其对象关系图，可表示如下：


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

~ Continued ~