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

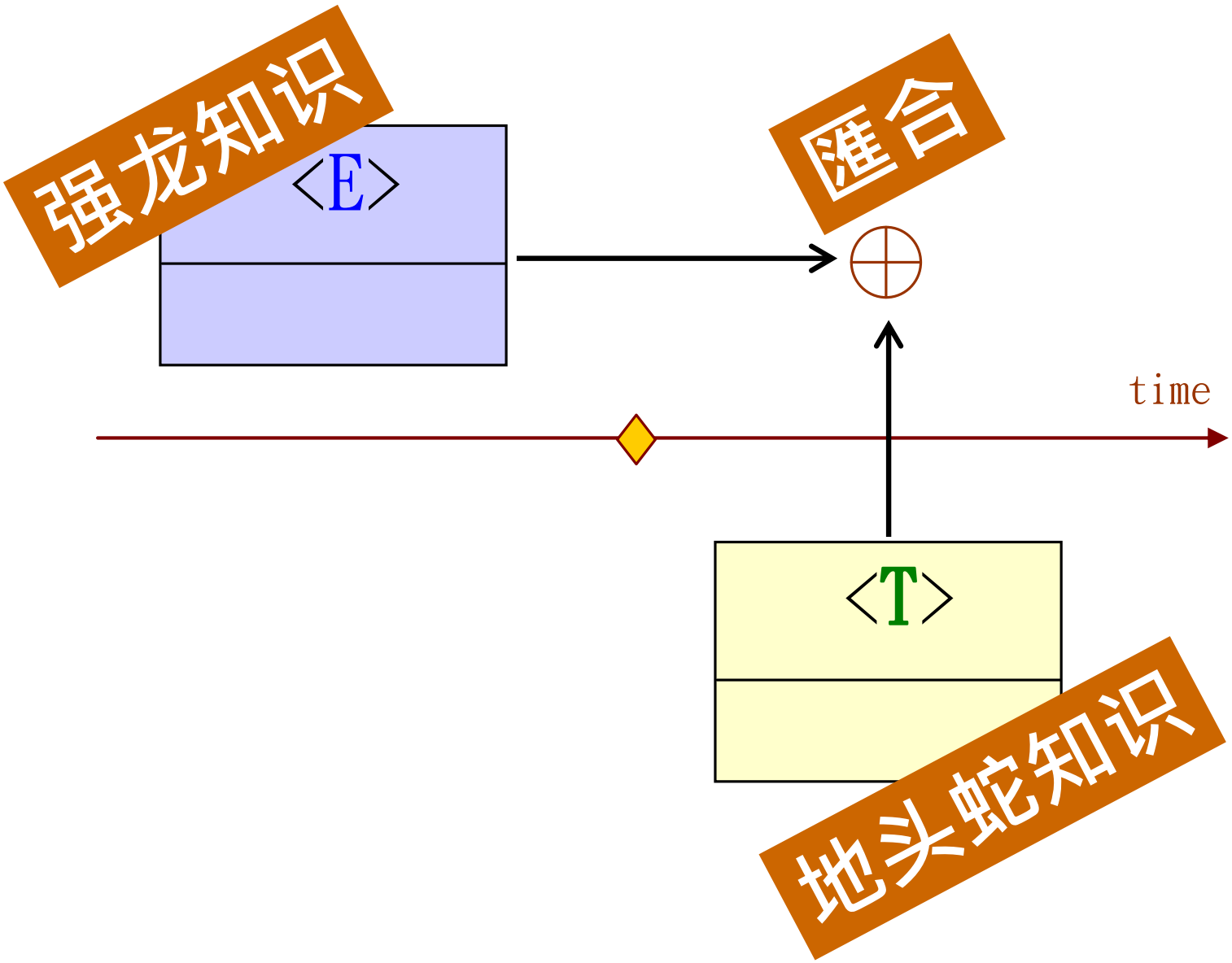
A07_b

从架构到代码的演练(b)

By 高焕堂

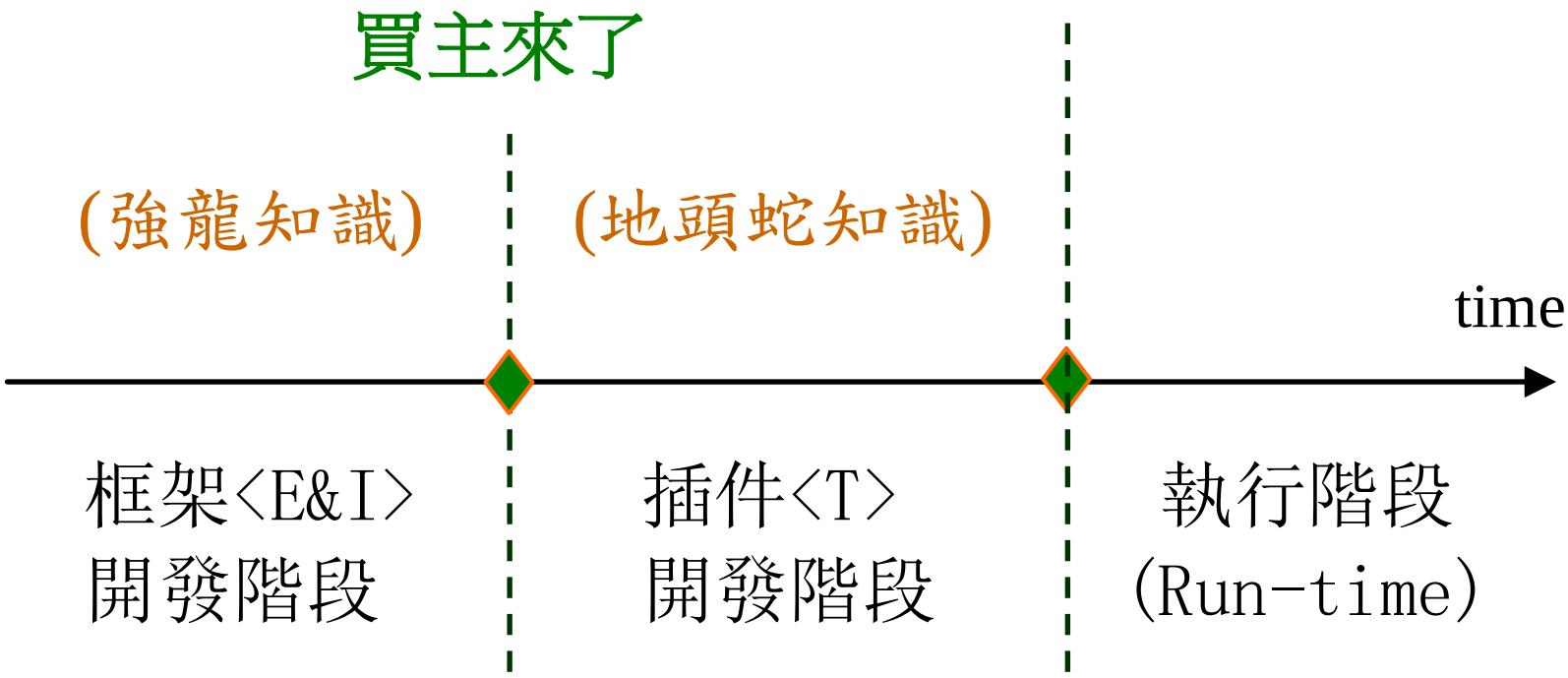
7、亲身体验：代码练习

- 于<买主来了>之前，架构师基于已具备的知识，来决定接口<I>；然后把其知识(又称**强龙**知识)写在基类<E>里。基于<I>，当买主出现了，才把买主的知识(又称**地头蛇**知识)写入子类<T>里。
- 然后，两者汇合，成为完整的知识了。如下图所示：

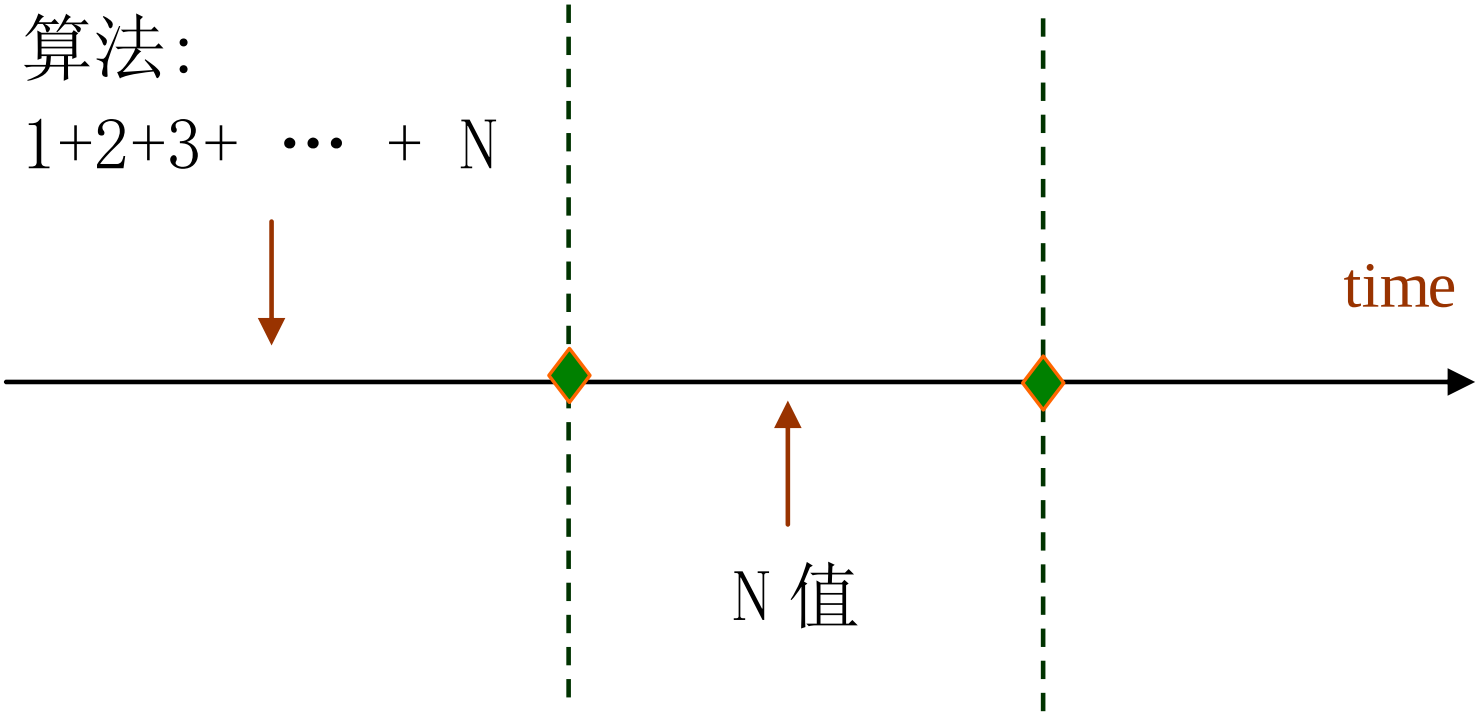

强龙知识
<E>
匯合
time
<T>
地头蛇知识

依循EIT，进行演练...

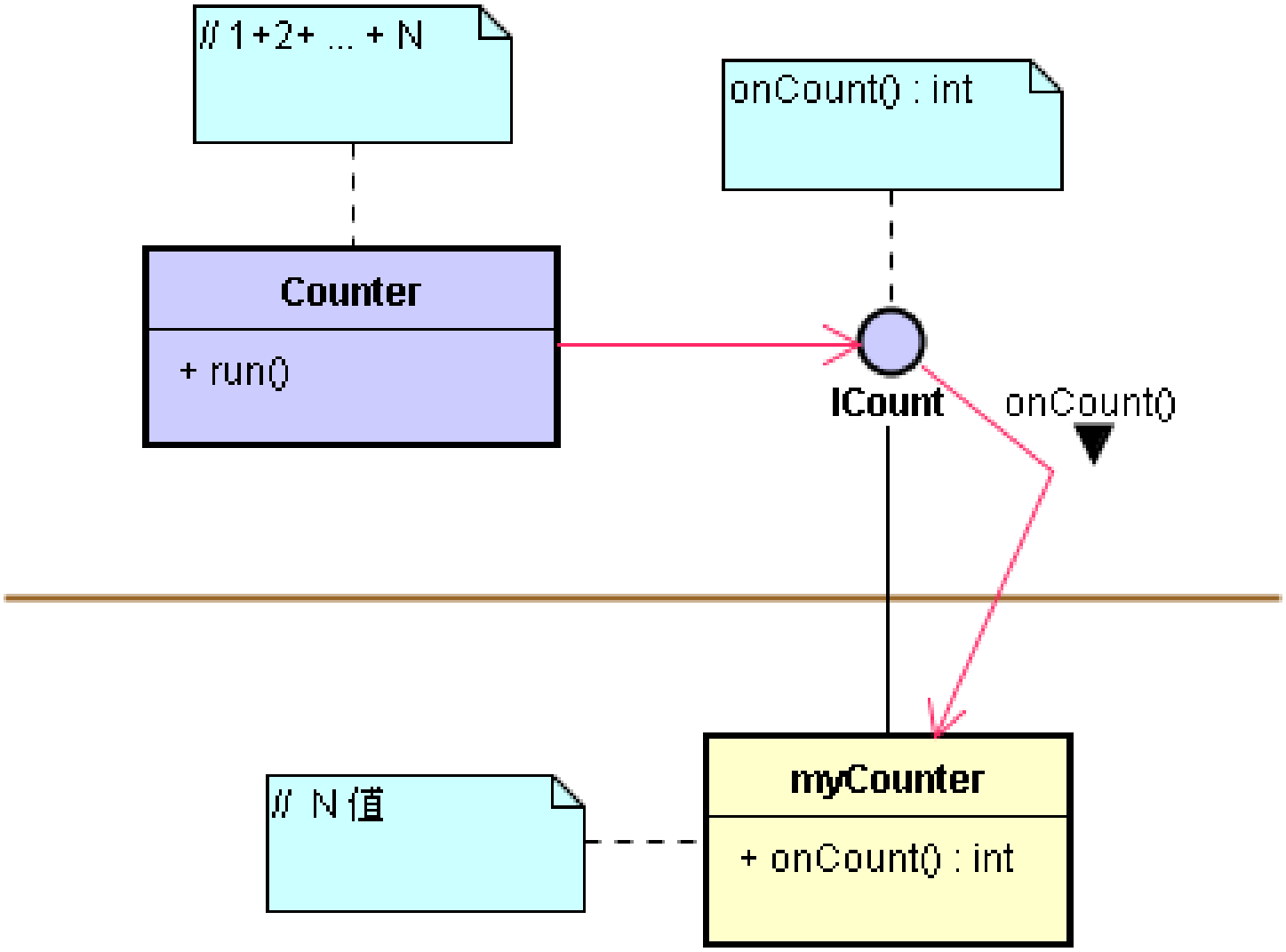
1. 架构师依循EIT造形，分出<E、I、T>三部分
2. 把强龙知识写入于<E>里
3. 把地头蛇(即买主)知识写入到<T>里
3. 详细定义接口<I>
4. <E>是控制点，透过<I>来调用<T>。


買主來了
(強龍知識)
(地頭蛇知識)
time
框架<E&I>
開發階段
插件<T>
開發階段
執行階段
(Run-time)

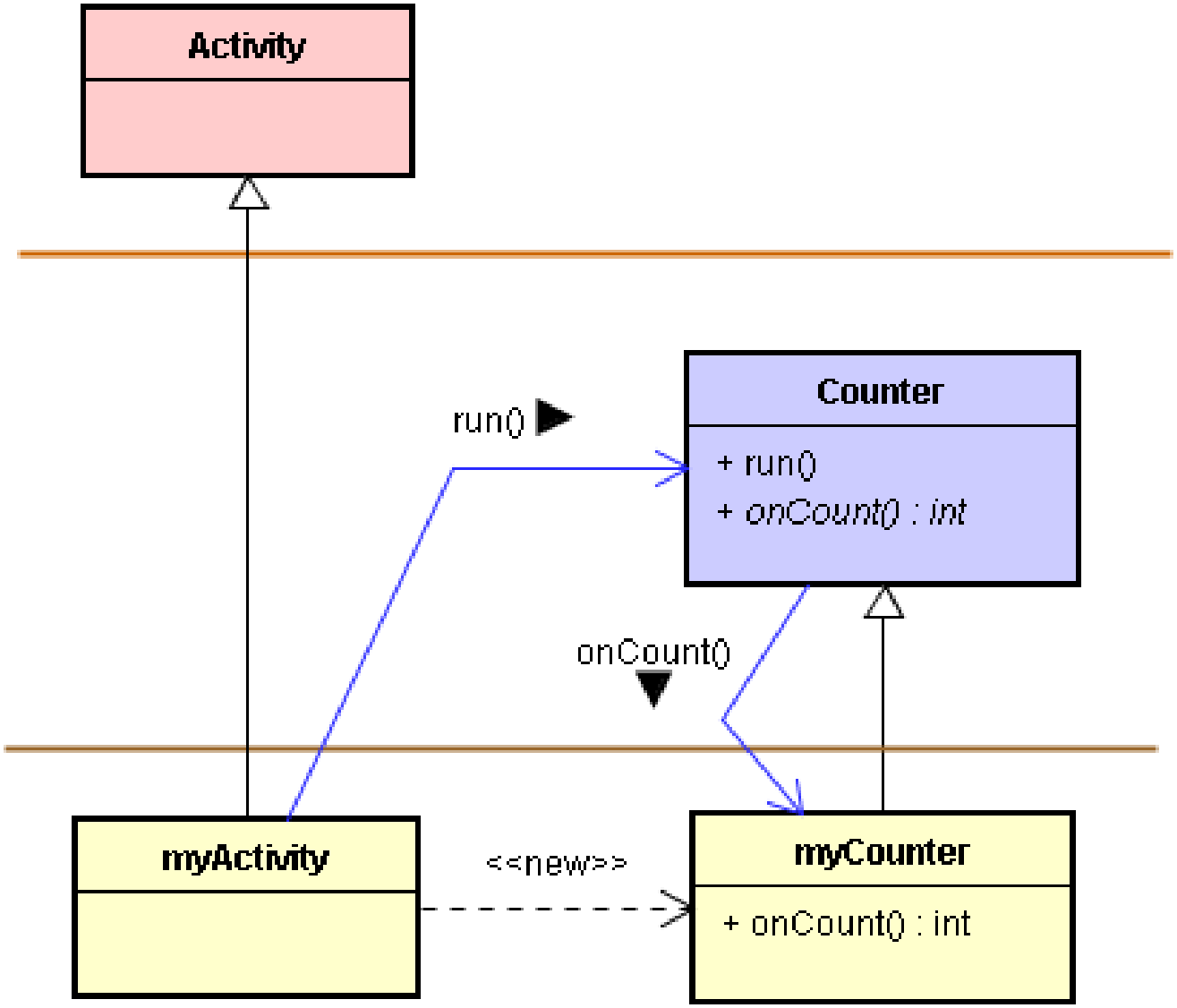
亲自演练：题目(一)



- 从上图可引导你决定三件事：
- 算法(1+2+3+... +N)必须撰写于<E>里。
- N值必须撰写于<T>里。
- 设计<I>里的函数，让<E>调用<T>去取得N值。


// 1+2+ ... + N
onCount() : int
Counter
+ run()
ICount
onCount()
// N 値
myCounter
+ onCount() : int

- 在架构設計上，可以将上图ICount接口裡的onCount()函數併入Counter類裡。
- 只要不改变onCount()函数，就不会影响到myCounter了。如下图：


Activity
Counter
+ run()
+ onCount() : int
run()
onCount()
myActivity
<<new>>
myCounter
+ onCount() : int

```
// Counter.java
package myFramework;

public abstract class Counter {
        public int run() {
            int N;
            N = this.onCount();
            int sum = 0;
            for(int i=1; i<=N; i++)  sum += i;
            return sum;
          }
        public abstract int onCount();
 }
```

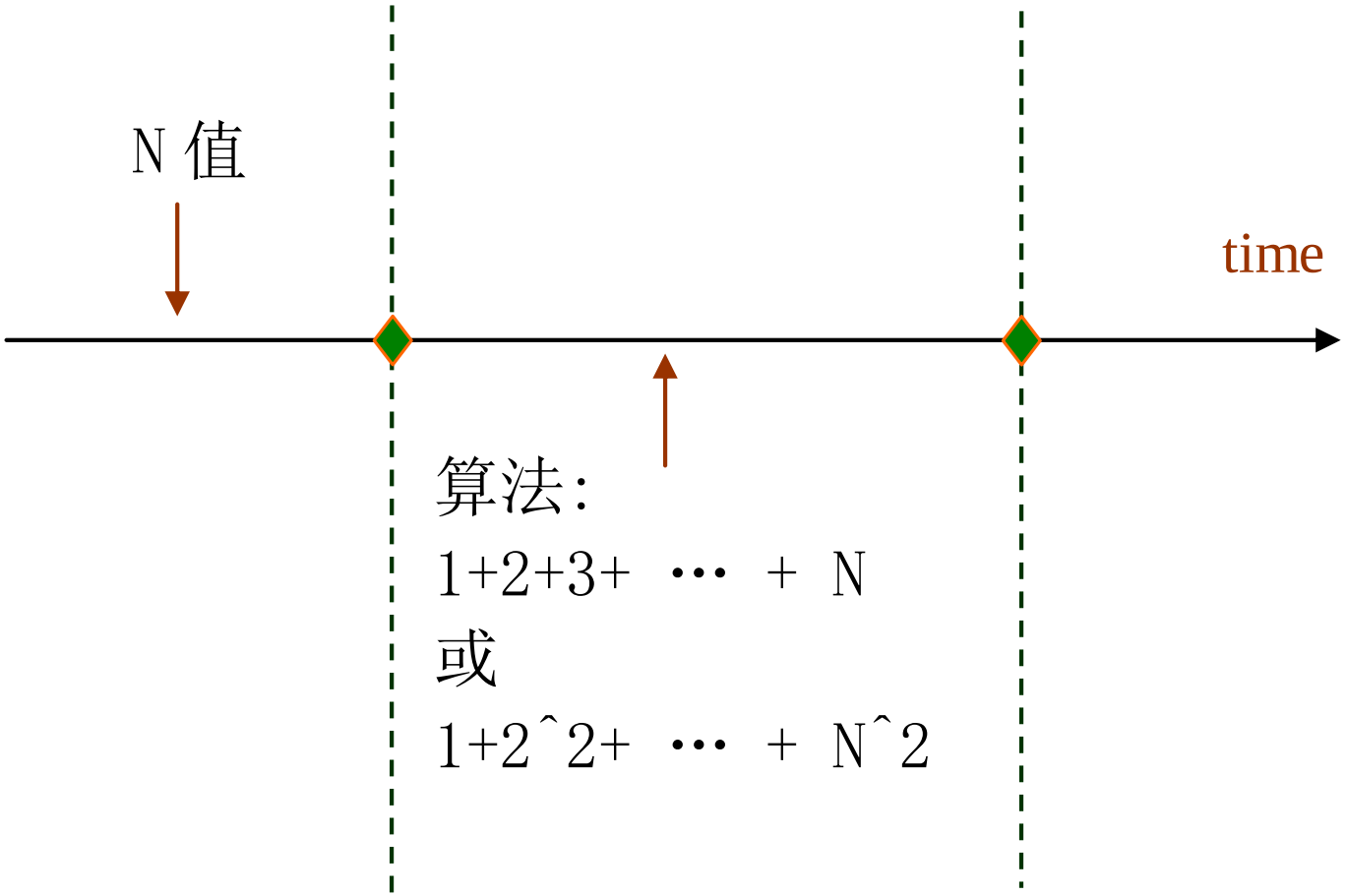
```java
// myCounter.java
package com.misoo.pk07;
import myFramework.Counter;

public class myCounter extends Counter{
   public int onCount() {
            return 10;
           }
}
```

```
// myActivity.java
// .......
public class myActivity extends Activity implements
    OnClickListener{
    private Button ibtn;
    private Counter counter;
    @Override
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        LinearLayout layout = new LinearLayout(this);
        layout.setOrientation(LinearLayout.VERTICAL);
        ibtn = new Button(this);
        ibtn.setOnClickListener(this);
        ibtn.setText("Exit");
```

```
        ibtn.setBackgroundResource(R.drawable.gray);
        LinearLayout.LayoutParams param1 =
            new LinearLayout.LayoutParams(150, 65);
        param1.topMargin = 10;
        param1.leftMargin = 5;
        layout.addView(ibtn, param1);
        setContentView(layout);
        counter = new myCounter();
        int sum = counter.run();
        setTitle("sum = " + String.valueOf(sum));
    }
    public void onClick(View arg0) {   finish();   }
}
```

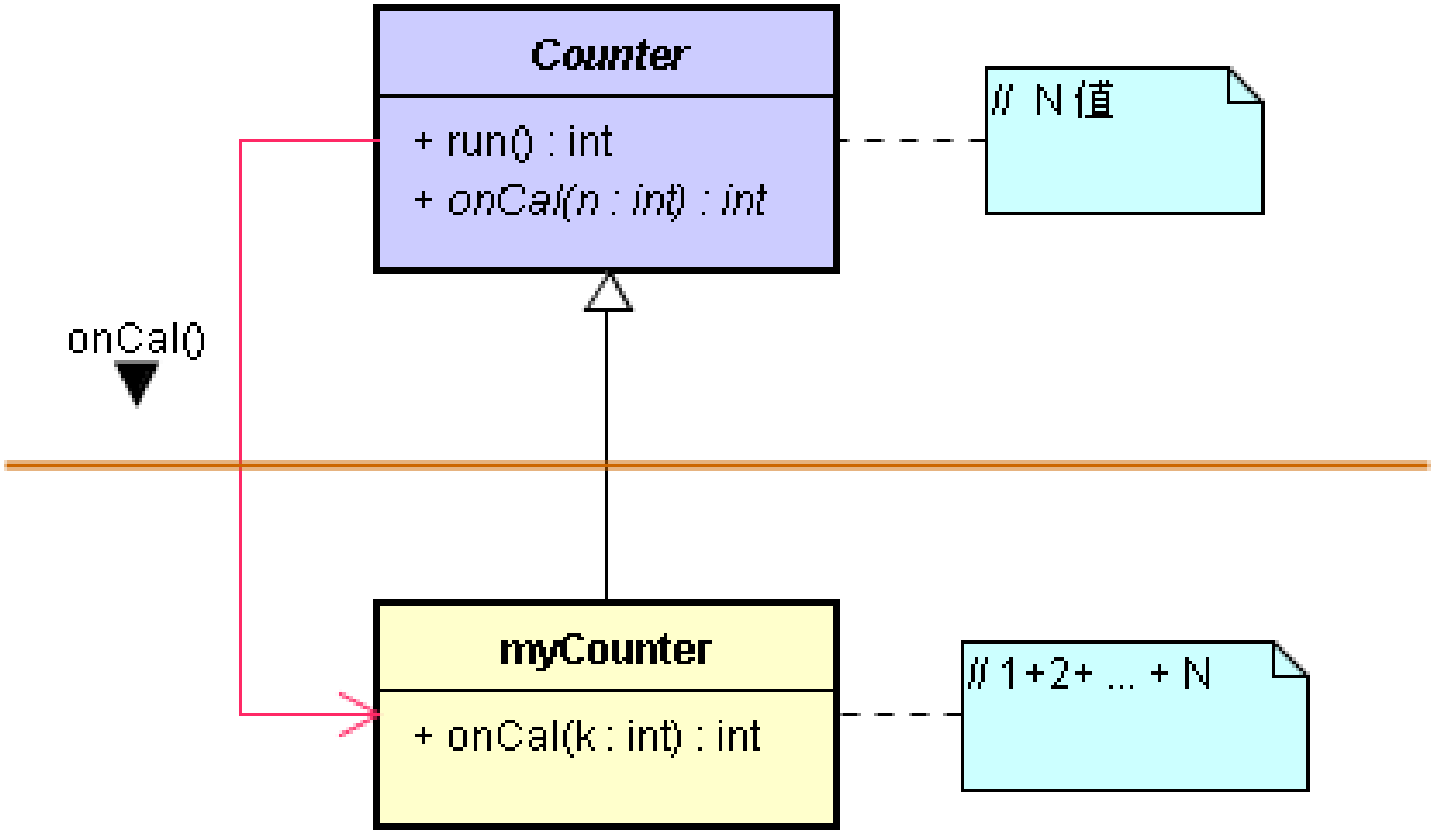
亲自演练：题目(二)



- 现在可以试试先想想接口<I>设计：<T>必须有个抽象函数，来反向调用(IoC)到<T>。在调用该函数时，顺便把<E>里的N值传递下去给<T>。
- 由<T>进行计算工作，然后将计算结果传回给<E>。

亦即，你可做下述三件事：

1) 将N值写在<T>里
2) 替各买主而设计一个子类别，将各自采取的算法(如1 + 2^2 + 3^2 +... +N^2)撰写于<T>里
3) 替<T>设计一个抽象函数onCal(n:int)，让<E>能调用<T>，取得<T>的计算结果


Counter
+ run() : int
+ onCal(n : int) : int
// N 值
onCal()
myCounter
+ onCal(k : int) : int
// 1+2+ ... + N

```
// Counter.java
public abstract class Counter {
        public int run(){
                        int N = getCount();
                        return onCal(N);
                }
        public int getCount() {  return 6;    }
        protected abstract int onCal(int n);
 }
```

```
// myCounter.java
public class myCounter extends Counter{
        @Override
        protected int onCal(int n) {
                int sum = 0;
                  for(int i=1; i<=n; i++) {
                     sum += i;
                   }
                  return sum;
        }
}
```

```
//……..
public class myActivity extends Activity{
        @Override
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        LinearLayout layout = new LinearLayout(this);
        //……..
        setContentView(layout);
        //----------------------------------------
        counter = new myCounter();
        int sum = counter.run();
        setTitle("sum = " + String.valueOf(sum));
    }
}
```

~ Continued ~