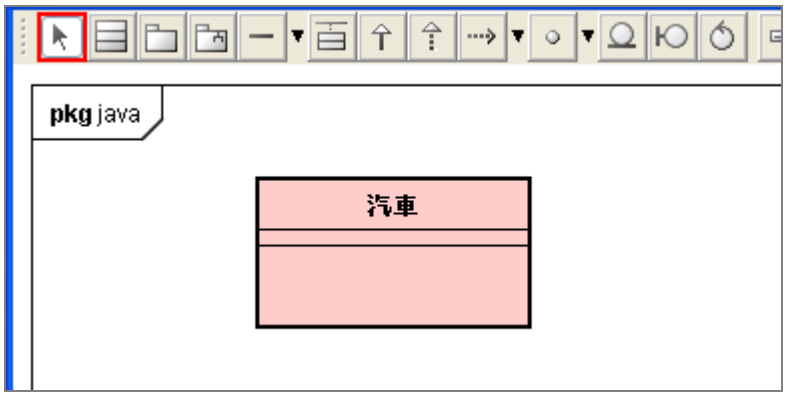
框架&API 開發演練(範例)

1. 框架&API 設計初步

1.1 使用 JUDE 工具

開啓 JUDE 建模工具，建立新的類圖(Class Diagram)，畫出一個類的圖像，並取名爲 ”汽車”：



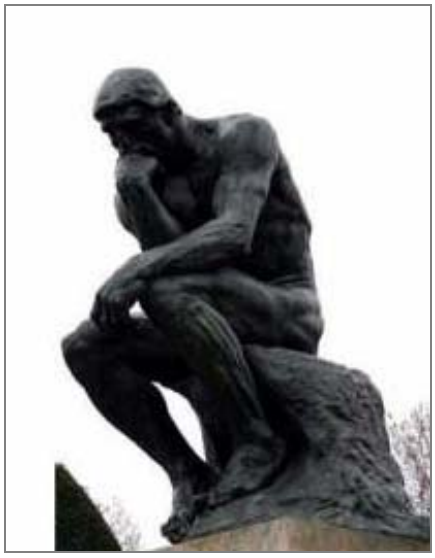
框架設計練習口訣：目前先”不”做輪胎

老子說：”無” 之以爲用(有之以爲利)

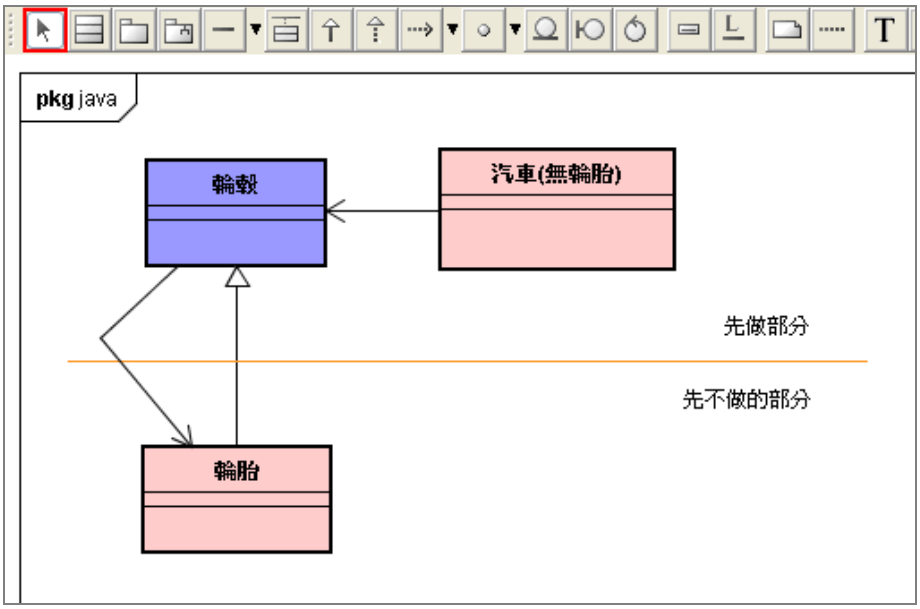
孔子說：知之爲知之，”不”知爲不知

1.2 羅丹的設計思維

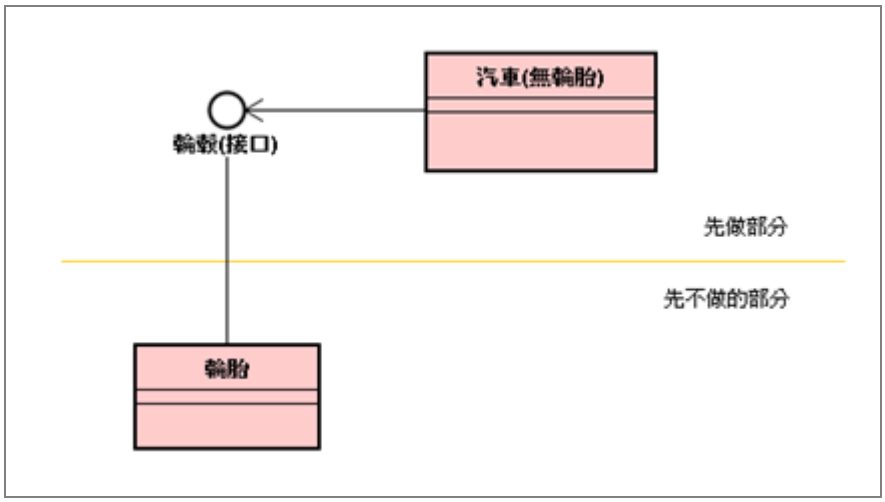
最偉大的雕刻師羅丹(Musée Rodin)說：把”不”必要的部分去掉。



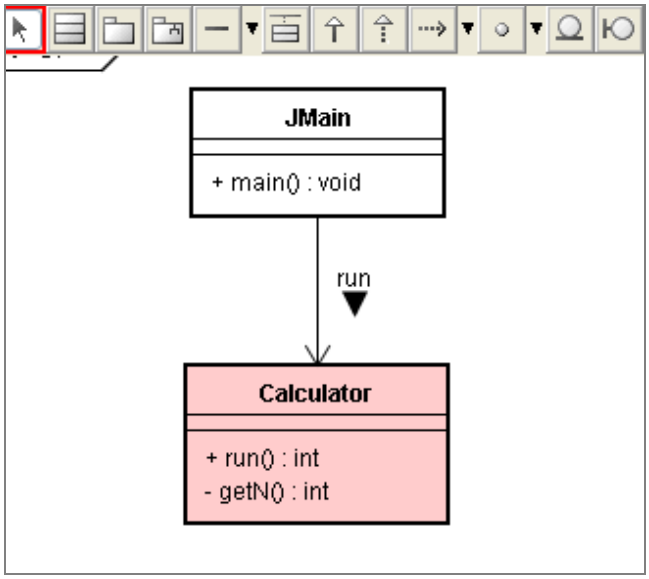
~~ 先不做輪胎 ~~

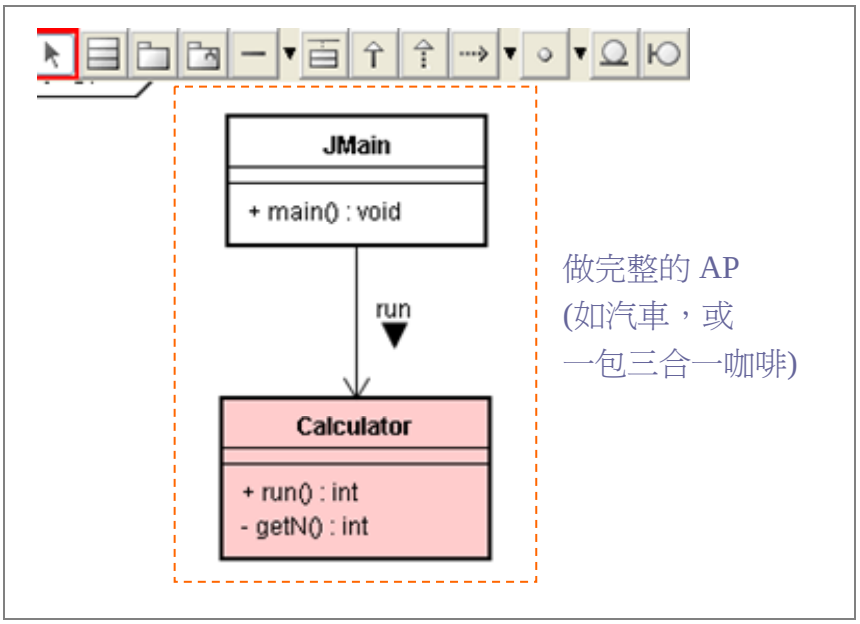


相當於：


汽車(無輪胎)
輪轂(接口)
先做部分
先不做的部分
輪胎

1.3 軟件實例示範

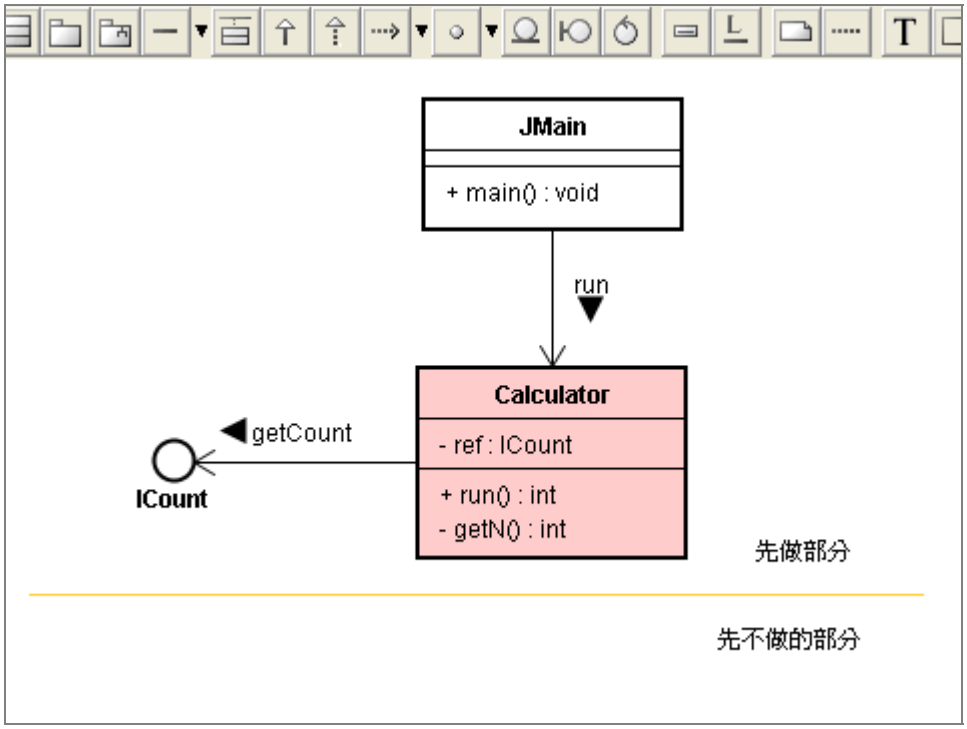

JMain
+ main() : void
run
Calculator
+ run() : int
- getN() : int




```
class Calculator {
        public int run() {
            int n = getN();
            int sum = 0;
            for(int i=0; i<=n; i++) {
                  sum += i;
              }
            return sum;
          }
          private int getN() {    return 10; }
}

//----------------------------------------
public class JMain {
       public static void main(String[] args) {
              Calculator cal = new Calculator();
              System.out.println(cal.run());
       }
}
```

~~ 先不做輪胎(先不加糖) ~~

~~ 先做輪轂(先準備砂糖紙包) ~~




```
// JMain.java
import Engine.Calculator;
public class JMain {
	public static void main(String[] args) {
		Calculator cal = new Calculator();
		System.out.println(cal.run());
	}
}
```

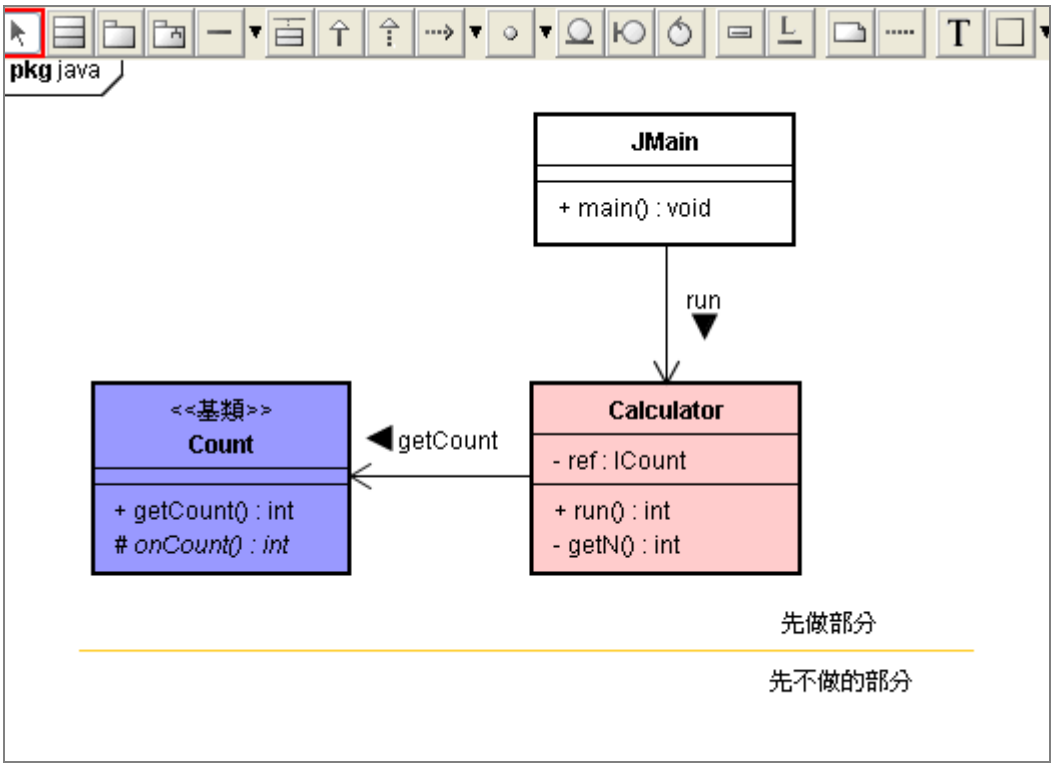
```
// 引擎部份
package Engine;
import Framework.ICount;
public class Calculator {
	 ICount ref;
	 public int run() {
```

```
                int n = getN();
                int sum = 0;
                for(int i=0; i<=n; i++) sum += i;
                return sum;
            }
        private int getN() {    return ref.getCount();    }
}
```

// 介面部分

```
package Framework;
public interface ICount {
        int getCount();
}
```

Java的Interface介面，就相當於純粹抽象基類(Pure abstract class)。所以，也能將上述 ICount 介面定義爲基類，如下：

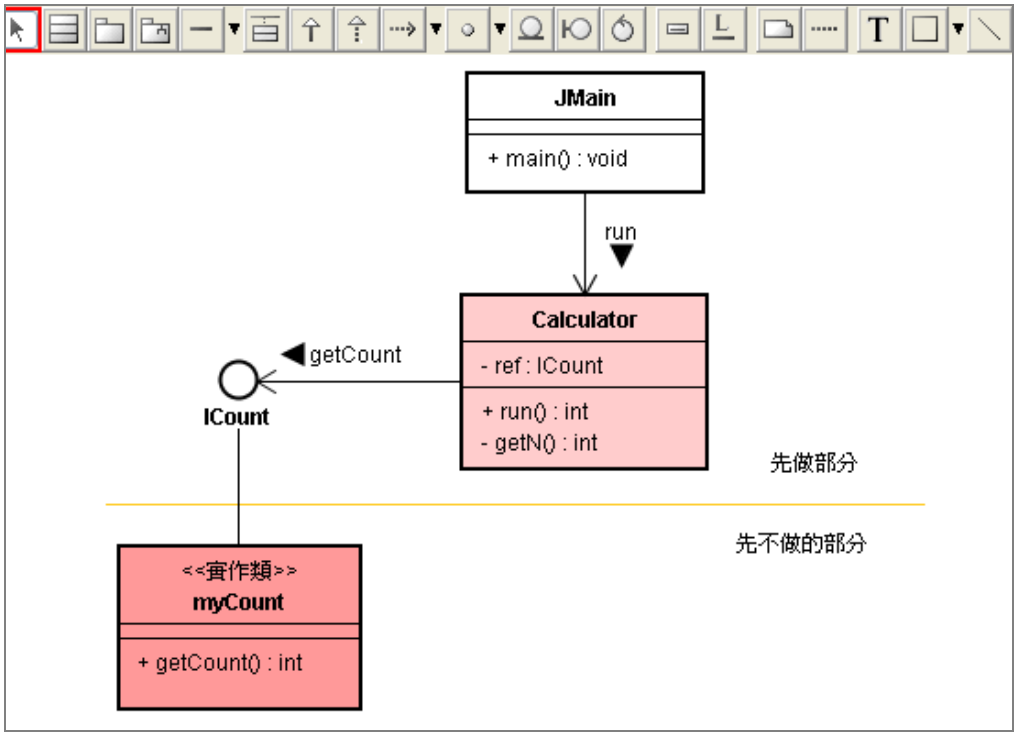


~~ 以上是做『分』的動作，將輪胎從汽車上”分” 離開來，或是將糖從三合一咖啡裡分離開來。

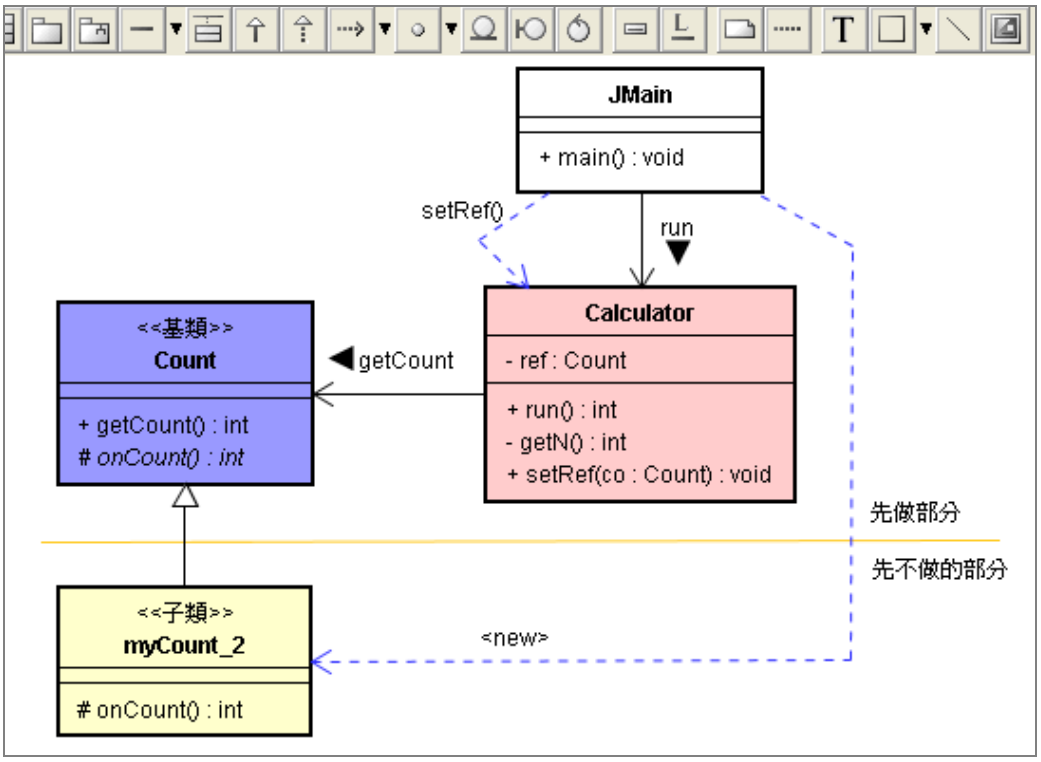
~~ 客人來了，主人詢問客人：需要冰糖或砂糖?

~~ 依據客人需求，主人開始做『合』的動作，將輪胎組”合”裝配到汽車上，或是將糖加入二合一咖啡，調”合”成可口的咖啡。

~~ 配上實作類(Implementation Class)：



~~ 配上子類(Sub Class)，並創建子類的對象~~



// JMain.java

```
import Engine.Calculator;
import Framework.Count;
public class JMain {
	public static void main(String[] args) {
		Calculator cal = new Calculator();
		Count co = new myCount();
		cal.setRef(co);
		System.out.println(cal.run());
	}
}
```

// 引擎部份

```
package Engine;
import Framework.Count;

public class Calculator {
```

```
        private Count ref;
        public int run() {
            int n = getN();
            int sum = 0;
            for(int i=0; i<=n; i++) sum += i;
            return sum;
          }
          private int getN() {      return ref.getCount();       }
          public void setRef(Count co){      ref = co;      }
}
```

//框架部份

```
package Framework;
public abstract class Count {
        public int getCount() {
                int cc = onCount();
                return cc;
       }
       protected abstract int onCount();
}
```

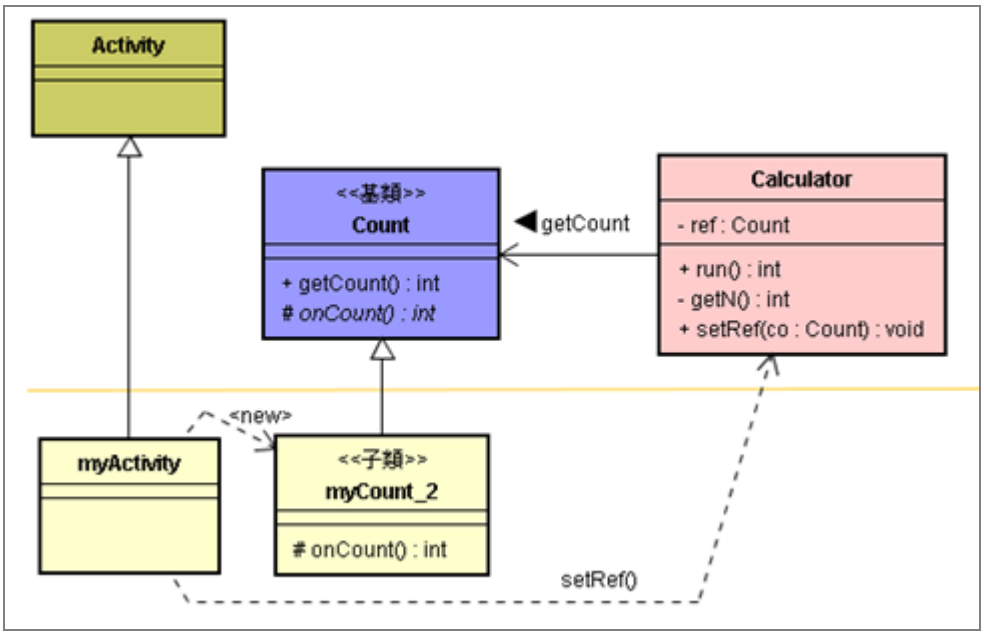
// 子類部份

```
import Framework.Count;
public class myCount extends Count{
       protected int onCount(){
             return 10;
       }
}
```

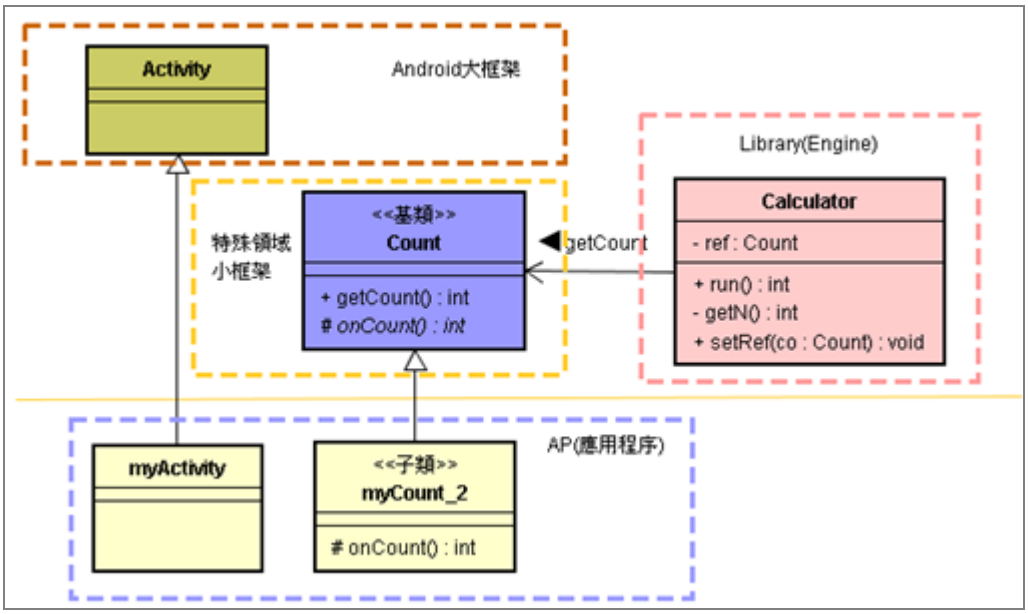
*** 這有一個矛盾的地方：

- 代碼 Count co = new myCount(); 意味著 myCount 類必須先於 JMain 類。
- 但是 JMain 又屬於框架層，顯然應該先於 myCount 類。

*** 如果使用 Android 的 myActivity 子類來替代 JMain 類，就解決上述的困境了。



*** 於是共含有兩層框架：Android 大框架和 DSF 小框架，如下圖：



總共含有 4 大模塊：

- Android 大框架
- DSF 小框架
- DSE 特殊領域引擎
- AP

*** 實踐代碼

// 引擎部份

```
package DS_Engine;
import DS_Framework.Count;

public class Calculator {
        private Count ref;
        public int run() {
            int n = getN();
            int sum = 0;
            for(int i=0; i<=n; i++) sum += i;
            return sum;
          }
          private int getN() {    return ref.getCount();    }
          public void setRef(Count co){    ref = co;    }
}
```

// DSF 小框架部份

```
package DS_Framework;
public abstract class Count {
        public int getCount() {
```

```
                int cc = onCount();
                return cc;
        }
        protected abstract int onCount();
}
```

// AP 部分
// myCount.java

```
package com.misoo.ppxx;
import DS_Framework.Count;

public class myCount extends Count{
        protected int onCount(){
                return 10;
        }
}
```

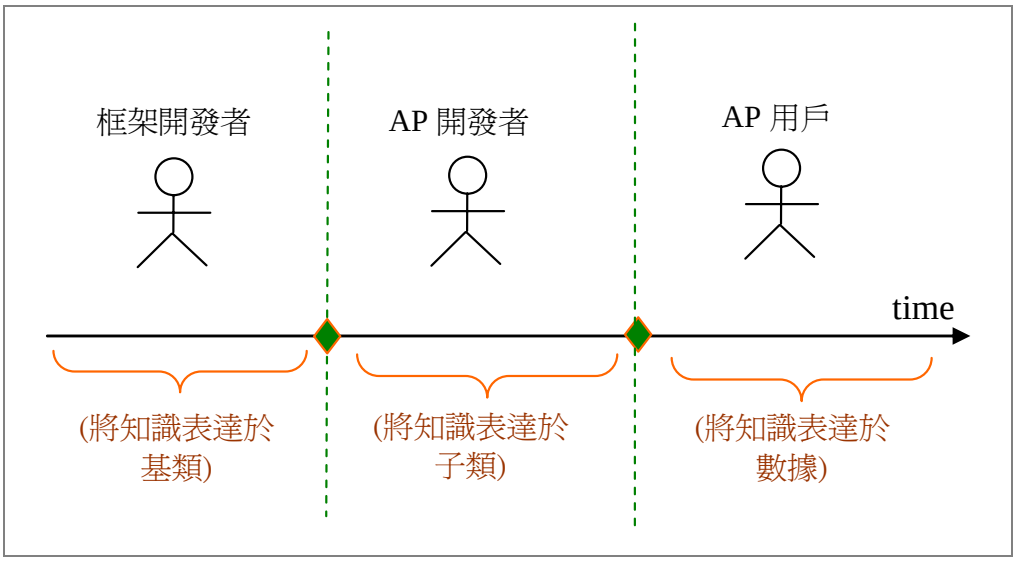
// myActivity.java

```
package com.misoo.ppxx;
import DS_Engine.Calculator;
import DS_Framework.Count;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.LinearLayout;

public class myActivity extends Activity implements OnClickListener {
        /** Called when the activity is first created. */
        private Button btn, btn2;

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                 LinearLayout layout = new LinearLayout(this);
```

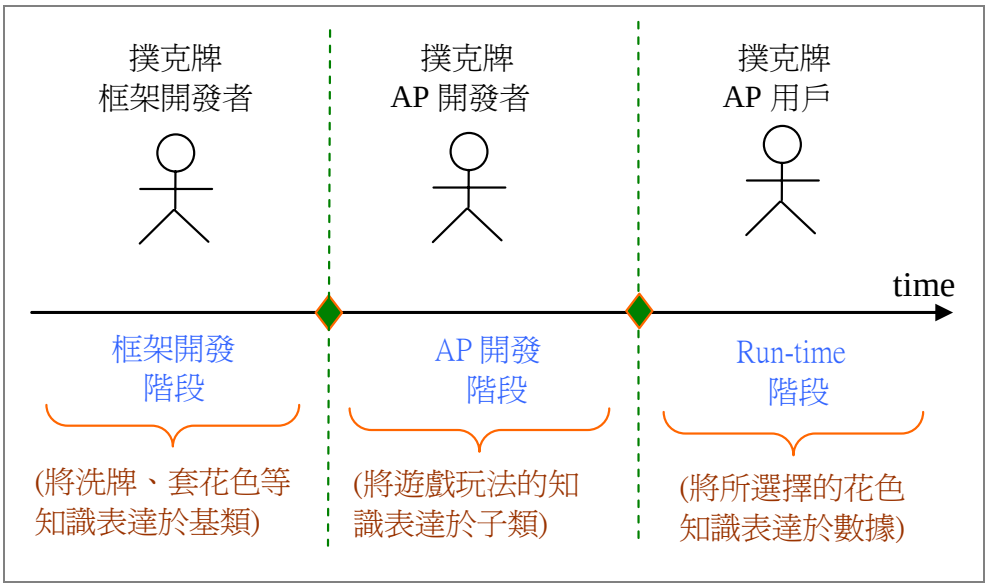
```
    layout.setOrientation(LinearLayout.VERTICAL);
        LinearLayout.LayoutParams param =
                new LinearLayout.LayoutParams(100, 55);
        param.leftMargin = 1;       param.topMargin = 3;

        btn = new Button(this);       btn.setId(101);
        btn.setText("Run");           btn.setOnClickListener(this);
        btn.setBackgroundResource(R.drawable.x_jude);
        layout.addView(btn, param);

        btn2 = new Button(this);     btn2.setId(102);
        btn2.setText("Exit");         btn2.setOnClickListener(this);
        btn2.setBackgroundResource(R.drawable.x_sky);
        layout.addView(btn2, param);
        setContentView(layout);
     }
   public void onClick(View v) {
         switch(v.getId()) {
          case 101:
                    Calculator cal = new Calculator();
                     Count co = new myCount();
                     cal.setRef(co);
                     int result = cal.run();
                     setTitle(String.valueOf(result));
                break;
          case 102:  finish();
                break;
  }}}
```

2. 認識「需求時間軸」

資料(Data)、子類(Subclass)與基類(Super class)是框架設計裏的三項重要機制。好的框架設計師都擅長應用它們來表達軟件發展三階段裏的不同知識。



以撲克牌遊戲爲例：



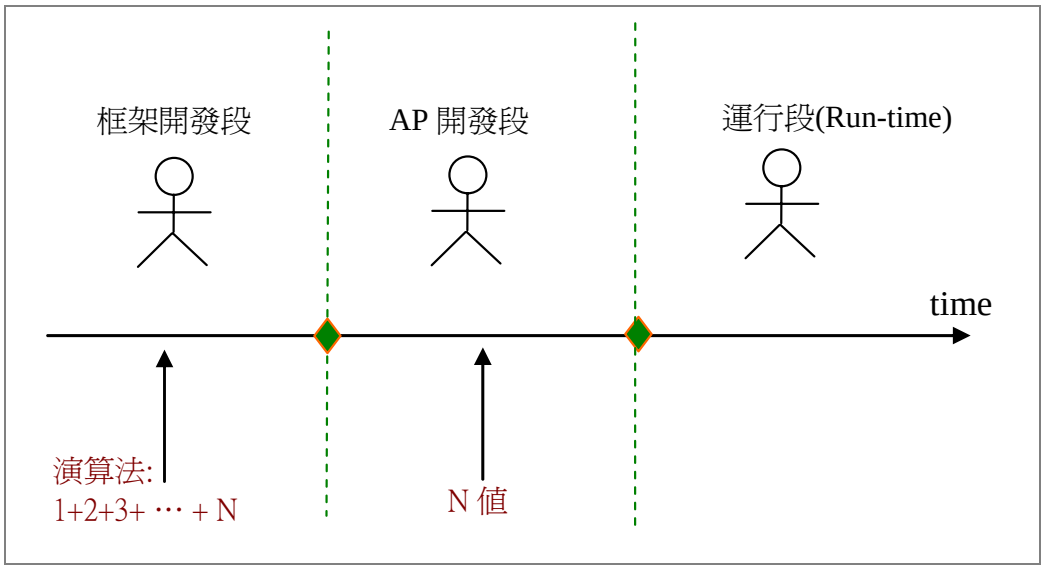
於是，我們就可以進行撲克牌軟件框架的開發工作了。完成框架設計之後，將其交給 AP 開發者，配上應用子類之後，編譯(Compile)之，就成爲可執

行的 AP 了。完成 AP 開發之後，將其交給用戶。於是用戶以資料來表達其心中屬意的花色，輸入電腦，就可以在電腦上玩撲克牌遊戲了。

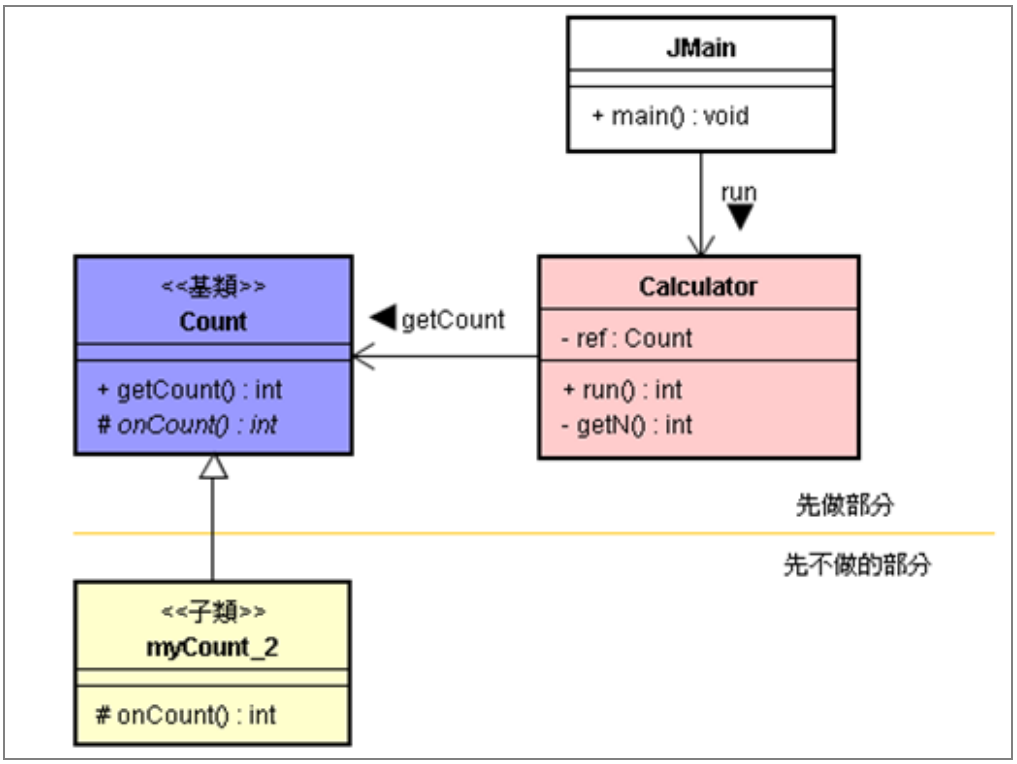
3. 框架&API 需求分析與設計

3.1 演練一

依據框架設計時間軸：



撰寫如下的源碼：



其中，框架部份的代碼(引擎是框架的一部分)：

// 引擎部份

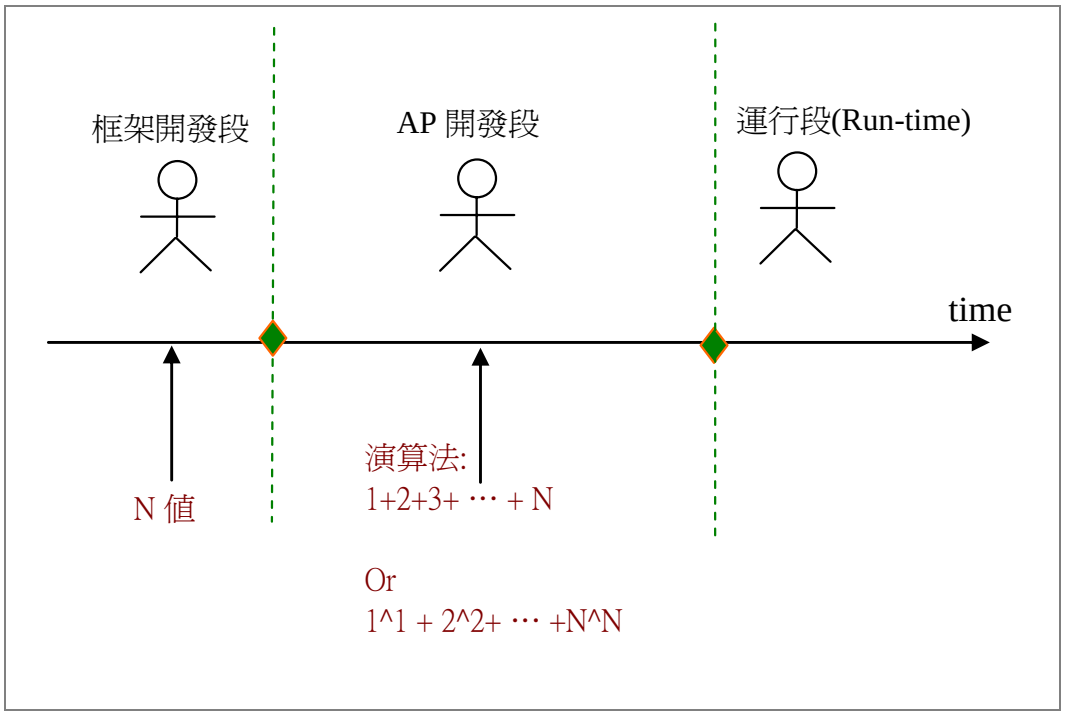
```
public class Calculator {
        private Count ref;
        public int run() {
            int n = getN();
            int sum = 0;
            for(int i=0; i<=n; i++)    sum += i;
            return sum;
         }
         private int getN() {       return ref.getCount();       }
         public void setRef(Count co){       ref = co;    }
}
```

//框架部份

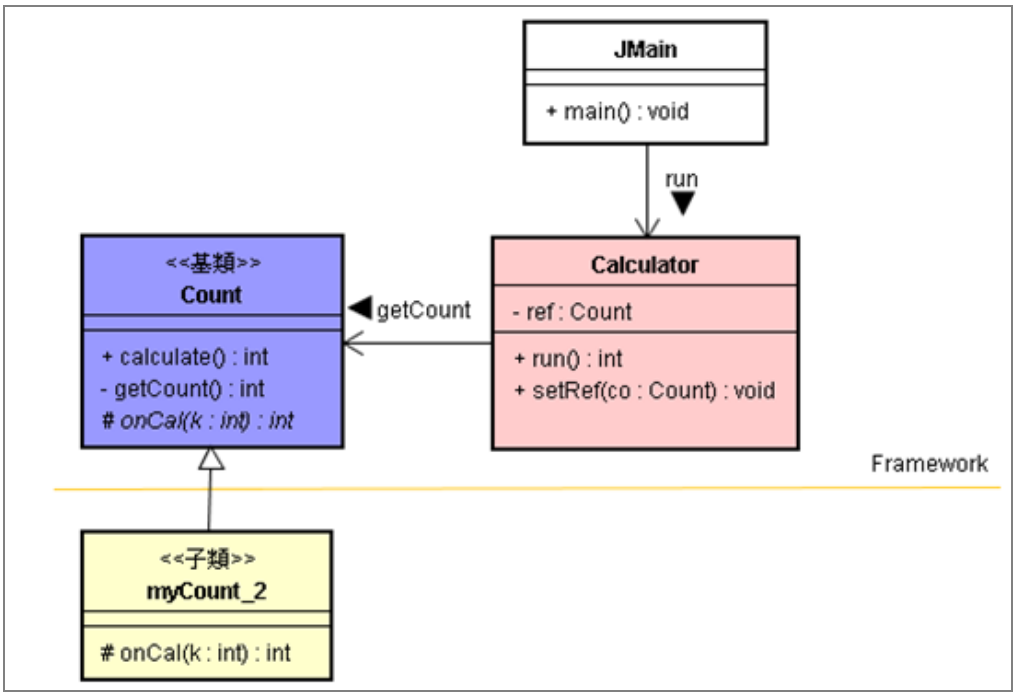
```
package Framework;
public abstract class Count {
        public int getCount() {
                int cc = onCount();
                return cc;
        }
        protected abstract int onCount();
}
```

3.2 演練二

如果時間軸改爲：



修改設計圖爲：



(<getCount 應改爲 calculate)

實現代碼如下：

//框架部份

```
package Engine;
import Framework.Count;
public class Calculator {
        private Count ref;
        public int run() {
              return ref.calculate();
           }
        public void setRef(Count co){
               ref = co;
       }
}
```

```
package Framework;
public abstract class Count {
        public int calculate(){
                int N = getCount();
                return onCal(N);
        }
        public int getCount() {
                return 10;
      }
      protected abstract int onCal(int k);
}
```

//應用部分

```
import Framework.Count;
public class myCount extends Count{
      protected int onCal(int N){
                int sum = 0;
                for(int i=0; i<=N; i++) {
                      sum += i;
                 }
                return sum;
      }
}
```

```
import Engine.Calculator;
import Framework.Count;
public class JMain {
      public static void main(String[] args) {
            Calculator cal = new Calculator();
            Count co = new myCount();
            cal.setRef(co);
            System.out.println(cal.run());
      }
}
```

當客戶需要計算：

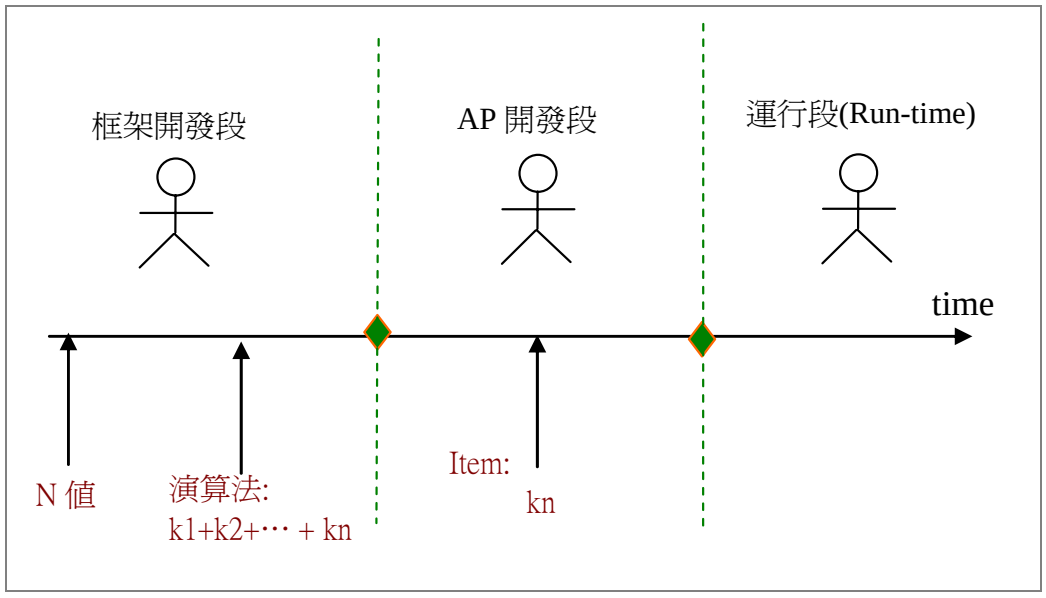
`1^1 + 2^2 + … + N^N` 時候，

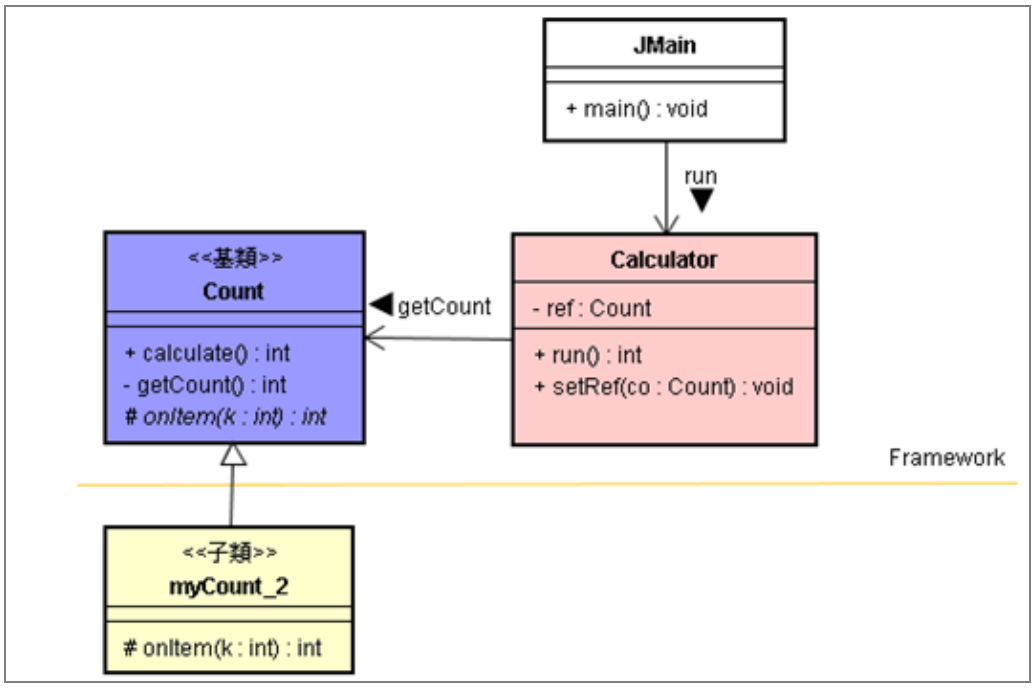
可撰寫應用子類如下：

```
import Framework.Count;
public class myCount _2 extends Count{
        protected int onCal(int N){
                int sum = 0;
                for(int i=0; i<=N; i++) {        sum += i*i;        }
                return sum;
        }}
```

3.3 演練三

如果時間軸改爲：





(<getCount 應改爲 calculate)

// 框架部份

```
package Framework;
public abstract class Count {
        public int calculate(){
              int N = getCount();
              int sum = 0;
                  for(int i=0; i<=N; i++)    sum += onItem(i);
                  return sum;
        }
        public int getCount() {      return 10;      }
       protected abstract int onItem(int k);
}
```

//應用子類部份

```
import Framework.Count;
public class myCount extends Count{
       protected int onItem(int k){
```

```
                return k;
        }
}
```

框架提供 Default 行爲：

// 基類部份

```
package Framework;
public abstract class Count {
        public int calculate(){
                int N = getCount();
                int sum = 0;
                    for(int i=0; i<=N; i++) {
                            sum += onItem(i);
                      }
                    return sum;
        }
        public int getCount() {
                  return 10;
        }
       protected int onItem(int k) {
              return k;
       }
}
```

// 應用子類

```
import Framework.Count;
public class myCount extends Count
{
}
```

~ END ~