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_c

复习基本OOP技术(c)

By 高煥堂

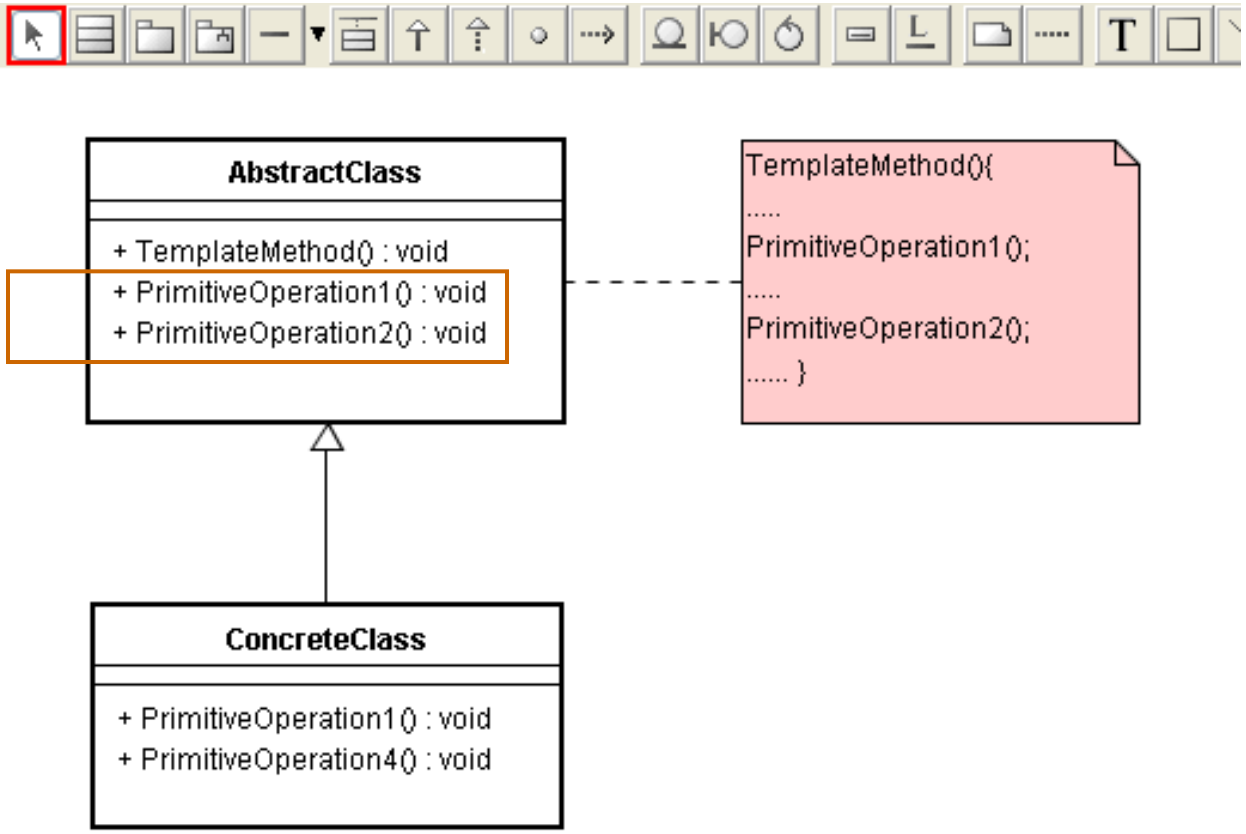
6、<基类/子类>结构的接口(卡榫函数)

卡榫函数

- 所谓「卡榫(Hook)」，就是用来接合两个东西的接口。如果两个东西于不同时间出现，则一方会预留虚空，给予另一边于未来时刻能以实体来填补该空间，两者虚实相依，就密合起来了。设计优良的卡榫，可以让实体易于新陈代谢、抽换自如(Plug and Play, 俗称PnP)。

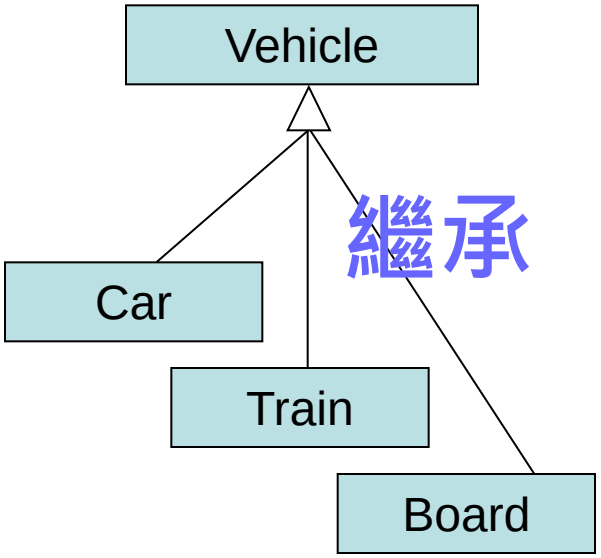
Template Method設計模式[GoF]

Hook函數



- 变与不变的分离(Separate code that changes from the code that doesn’t)是设计卡榫(Hook)函数及应用框架之基本原则和手艺。
- 分离出变(Variant)与不变(Invariant)部份之后，就可以将不变部份写在父类别(Super-class)里，而变的部份就写在子类别(Subclass)里。

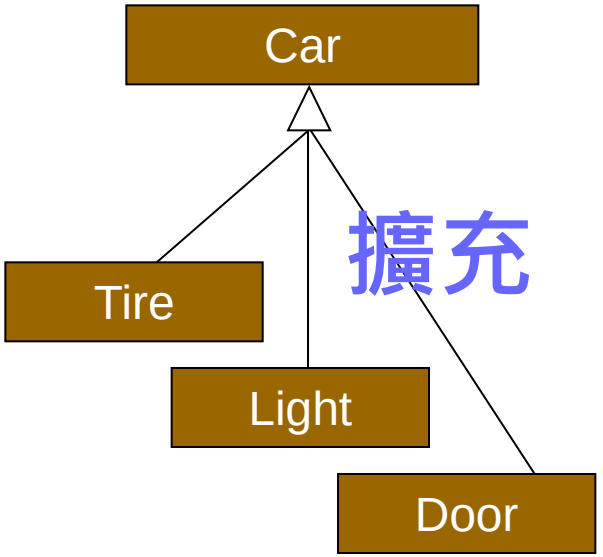
- 然后，藉由C++的类别继承(Inherit)机制组织起来 。

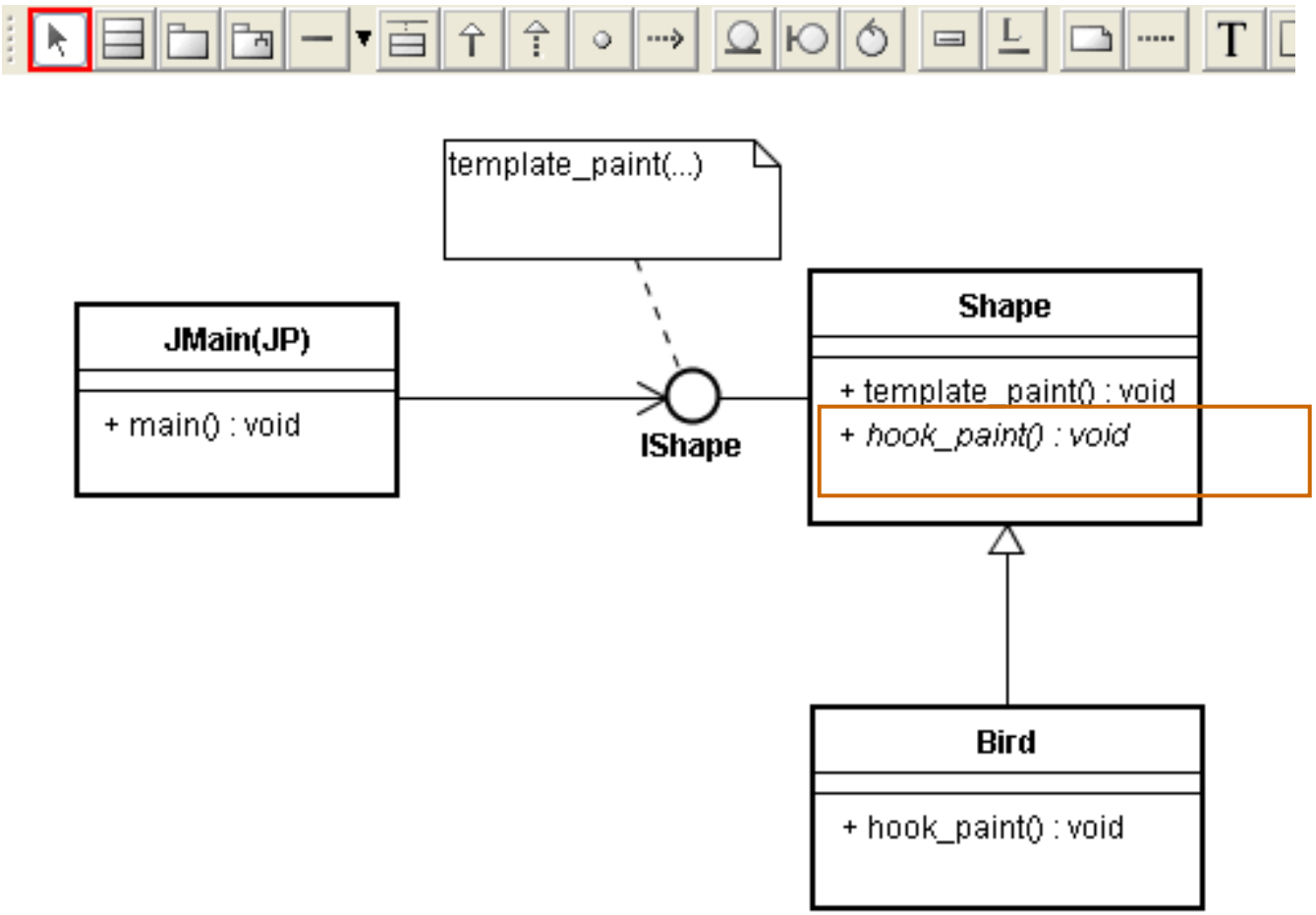



```
// C++
 class Car : Vehicle  {

//.....
}
```

- 或者，藉由Java的类别扩充(Extend)机制组织起来。

```
// Java
class Tire extends Car  {

//…..
}
```




template_paint(...)
JMain(JP)
+ main() : void
IShape
Shape
+ template_paint() : void
+ hook_paint() : void
Bird
+ hook_paint() : void

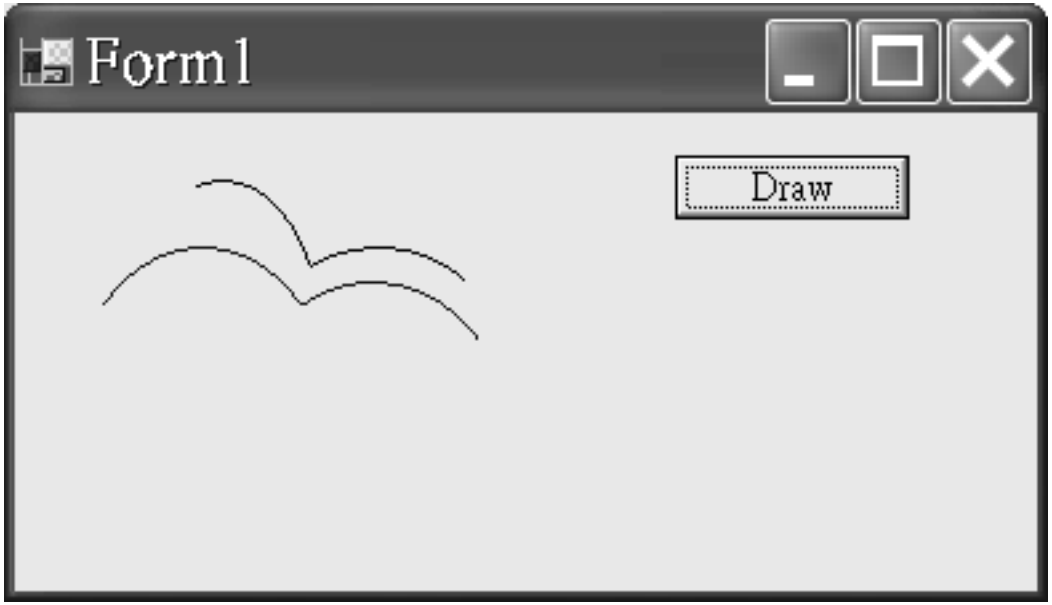
卡榫函数的Java实现

- 在Java里，使用抽象(abstract)函数或可覆写(overridable)函数来实现卡榫函数。

```
interface IShape {
    void template_paint(Graphics gr);
} // 一般接口
```

```
// Shape.java
import java.awt.*;
public abstract class Shape implements IShape {
    public void template_paint(Graphics gr){
        invariant_paint(gr);  // 畫背景
        hook_paint(gr);    // 畫前景
       }
    private void invariant_paint(Graphics gr){
         gr.setColor(Color.black);
         gr.fillRect(10,30, 200,100);
       }  // 默認行為
    abstract protected void hook_paint(Graphics gr);
}
```

```
// Bird.java
import java.awt.*;
public class Bird extends Shape {
    private void hook_paint(Graphics gr){
      // 畫圖(海鷗)指令
      gr.setColor(Color.cyan);
      gr.drawArc(30,80,90,110,40,100);
      gr.drawArc(88,93,90,100,40,80);
      gr.setColor(Color.white);
      gr.drawArc(30,55,90,150,35,75);
      gr.drawArc(90,80,90,90,40,80);
}
```


Form1
Draw

7、IoC机制与Default函数

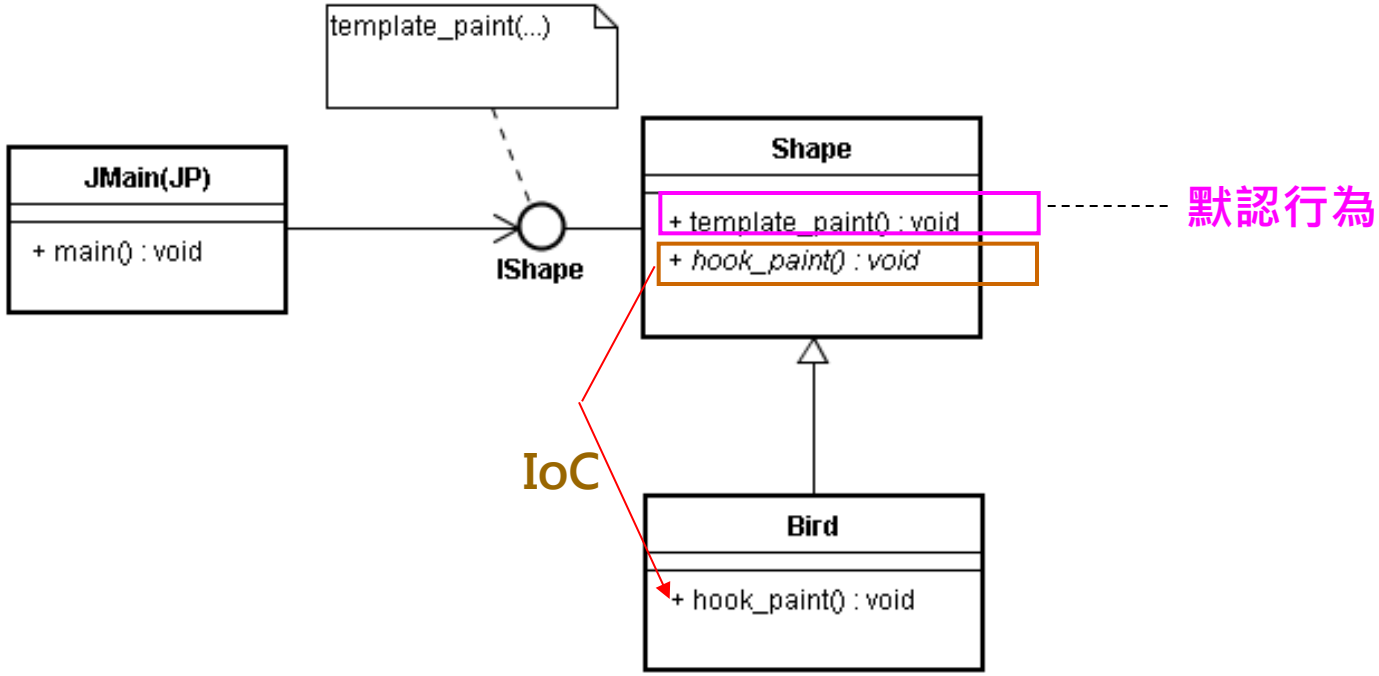
卡榫函数实现IoC机制

- 控制反转(IoC: Inversion of Control)
- IoC机制源自于OO语言(如C++等)的类别继承体系，例如C++语言中，基类的函数可以主动调用子类的函数，这就是典型的IoC机制。

- 基类与子类之间，主控权是在基类手上，透过Hook函数来调用子类
- 通常基类是撰写在先，而子类则撰写在后，这种前辈拥有主导权，进而「控制」后辈之情形，就通称为「控制反转」。

默认(Default)行为

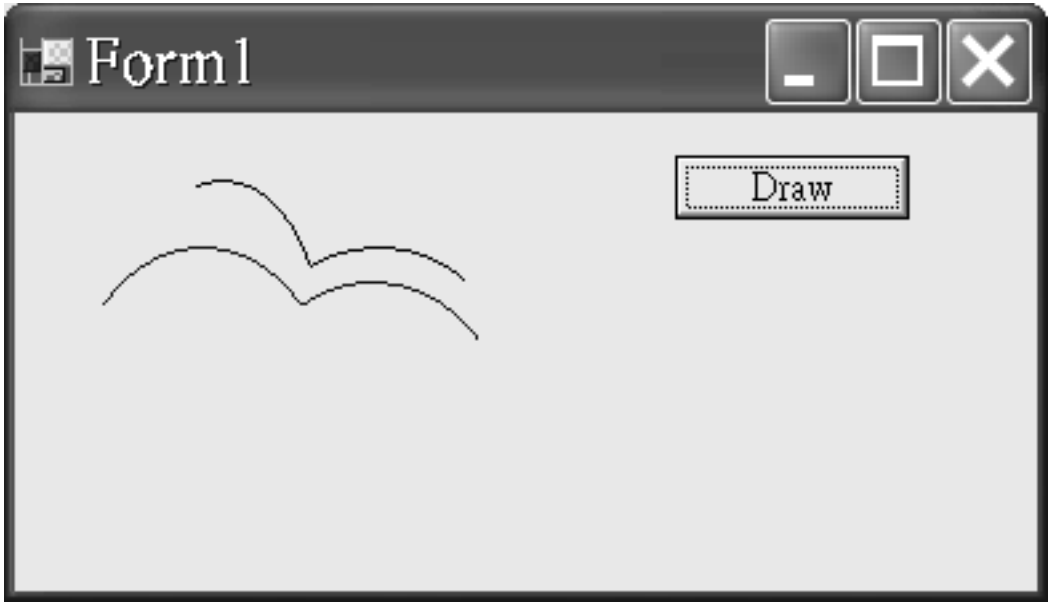
- 基类的重要功能：提供默认(预设)行为
- 基类可事先定义许多「默认」(Default)函数。这些默认函数可让子类来继承(或调用)之。


template_paint(...)
JMain(JP)
+ main() : void
IShape
Shape
+ template_paint() : void
+ hook_paint() : void
默認行為
IoC
Bird
+ hook_paint() : void

```
interface IShape {
    void template_paint(Graphics gr);
} // 一般接口
```

```
// Shape.java
import java.awt.*;
public abstract class Shape implements IShape {
    public void template_paint(Graphics gr){
        invariant_paint(gr);  // 畫背景
        hook_paint(gr);    // 畫前景
       }
    private void invariant_paint(Graphics gr){
         gr.setColor(Color.black);
         gr.fillRect(10,30, 200,100);
       }  // 默認行為
    abstract protected void hook_paint(Graphics gr);
}
```

```
// Bird.java
import java.awt.*;
public class Bird extends Shape {
    private void hook_paint(Graphics gr){
      // 畫圖(海鷗)指令
      gr.setColor(Color.cyan);
      gr.drawArc(30,80,90,110,40,100);
      gr.drawArc(88,93,90,100,40,80);
      gr.setColor(Color.white);
      gr.drawArc(30,55,90,150,35,75);
      gr.drawArc(90,80,90,90,40,80);
}
```


Form1
Draw

Thanks...

高煥堂