**外观模式（Facade Pattern）**

**简单介绍：**

外观模式（Facade Pattern）：

外观模式（Facade）的定义:**为子系统中的一组接口提供一个一致的界面，用来访问子系统中的一群接口。**

比如调用子系统1和子系统2的Print方法，如果没有外观模式那么你需要一个个的调用，子系统如果是少还好，多了就要疯了，这里就可以使用外观模式，调用外观模式对外提供的方法一次性完成对子系统的层层调用

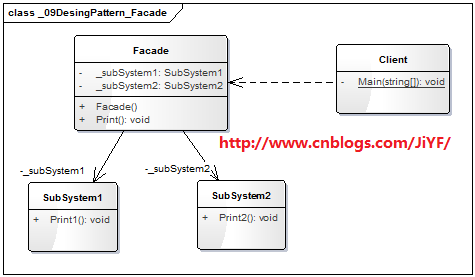
生活实例：比如笔记本是一个整体，看似是一件东西，实际上只是将主机和显示器组装在一起而已，对外提供一个电源按钮，当按下电源按钮的时候，启动了主机和显示器，这里的这个电源按钮就差不多是外观模式。对外只提供了一个电源按钮。

外观模式组成：

Facade：负责子系统的的封装调用

Subsystem Classes：具体的子系统，实现由外观模式Facade对象来调用的具体任务

**外观模式类图：**



**外观模式C#代码举例：**

**SubSystem1：子系统1**

[复制代码](javascript:void(0);)

1 /// <summary>

2 /// 子系统1

3 /// </summary>

4 public class SubSystem1

5 {

6 public void Print1()

7 {

8 Console.WriteLine("SubSystem1");

9 }

10 }

[复制代码](javascript:void(0);)

**SubSystem2：子系统2类**

[复制代码](javascript:void(0);)

1 /// <summary>

2 /// 子系统2

3 /// </summary>

4 public class SubSystem2

5 {

6 /// <summary>

7 /// 子系统2

8 /// </summary>

9 public void Print2()

10 {

11 Console.WriteLine("SybSystem2");

12 }

13 }

[复制代码](javascript:void(0);)

**Facade：外观模式对外提供接口类**

[复制代码](javascript:void(0);)

1 /// <summary>

2 /// 外观模式对外接口类

3 /// </summary>

4 public class Facade

5 {

6 private SubSystem1 \_subSystem1;

7 private SubSystem2 \_subSystem2;

8

9 public Facade()

10 {

11 \_subSystem1 = new SubSystem1();

12 \_subSystem2 = new SubSystem2();

13 }

14

15 public void Print()

16 {

17 \_subSystem1.Print1();

18 \_subSystem2.Print2();

19 }

20 }

[复制代码](javascript:void(0);)

**用户测试类：**

[复制代码](javascript:void(0);)

1 /// <summary>

2 /// 客户端测试类

3 /// </summary>

4 class Client

5 {

6 static void Main(string[] args)

7 {

8 Facade facade = new Facade();

9 facade.Print();

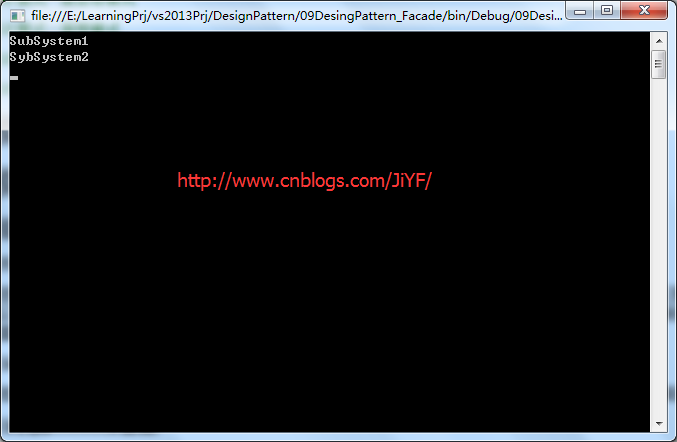
10 Console.Read();

11 }

12 }

[复制代码](javascript:void(0);)

**运行结果：**



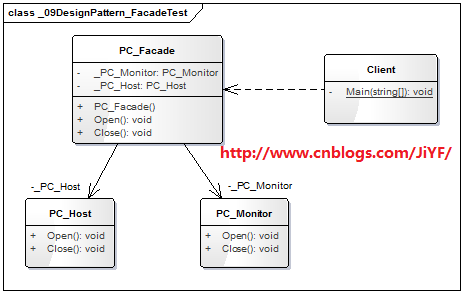
**外观模式生活实例笔记本例子：**

生活实例：比如笔记本是一个整体，看似是一件东西，实际上只是将主机和显示器组装在一起而已，对外提供一个电源按钮，当按下电源按钮的时候，启动了主机和显示器，这里的这个电源按钮就差不多是外观模式。对外只提供了一个电源按钮。

假设笔记本是由来个子系统：显示器和主机组成

当按下电源按钮，开启主机和显示器

**外观模式生活实例笔记本例子类图：**



**外观模式生活实例笔记本例子C#代码：**

**PC\_Monitor：笔记本显示器类**

[复制代码](javascript:void(0);)

1 /// <summary>

2 /// 笔记本显示器

3 /// </summary>

4 public class PC\_Monitor

5 {

6 public void Open()

7 {

8 Console.WriteLine("开启显示器");

9 }

10 public void Close()

11 {

12 Console.WriteLine("关闭显示器");

13 }

14 }

[复制代码](javascript:void(0);)

**PC\_Host：笔记本主机类**

[复制代码](javascript:void(0);)

1 /// <summary>

2 /// 笔记本主机

3 /// </summary>

4 public class PC\_Host

5 {

6 public void Open()

7 {

8 Console.WriteLine("开启主机");

9 }

10 public void Close()

11 {

12 Console.WriteLine("关闭主机");

13 }

14 }

[复制代码](javascript:void(0);)

**PC\_Facade：笔记本外观模式对外接口类**

[复制代码](javascript:void(0);)

1 /// <summary>

2 /// 笔记本对外接口类

3 /// </summary>

4 public class PC\_Facade

5 {

6 private PC\_Monitor \_PC\_Monitor;

7 private PC\_Host \_PC\_Host;

8

9 public PC\_Facade()

10 {

11 \_PC\_Monitor = new PC\_Monitor();

12 \_PC\_Host = new PC\_Host();

13 }

14

15 public void Open()

16 {

17 Console.WriteLine("开机……");

18 \_PC\_Monitor.Open();

19 \_PC\_Host.Open();

20 }

21

22 public void Close()

23 {

24 Console.WriteLine("关机……");

25 \_PC\_Host.Close();

26 \_PC\_Monitor.Close();

27 }

28 }

[复制代码](javascript:void(0);)

**用户测试代码：**

[复制代码](javascript:void(0);)

1 class Client

2 {

3 /// <summary>

4 /// 用户测试类

5 /// </summary>

6 /// <param name="args"></param>

7 static void Main(string[] args)

8 {

9 PC\_Facade pc\_facade = new PC\_Facade();

10 //开机

11 pc\_facade.Open();

12 //关机

13 pc\_facade.Close();

14 Console.Read();

15 }

16

[复制代码](javascript:void(0);)

**运行结果：**

