河北师范大学软件学院
Software College of Hebei Normal University

第九章 外观模式

任课教师：武永亮
wuyongliang@edu2act.org

上节回顾

- 合成/聚合复用原则
- 桥接模式是“将抽象部分和其实现部分分离，使它们都可以独立的变化”

课程内容

- 环境及问题
- 外观模式详解
- 外观模式实现
- 扩展练习

课程内容

- 环境及问题
- 外观模式详解
- 外观模式实现
- 扩展练习

环境

- 电脑的组成你了解么？
- 你了解电脑的启动过程吗？
- 请设计一个电脑类。其中包含电脑的开启和关闭方法。

请利用15分钟时间对该系统进行设计编码

环境

```
class Cpu
{
    public void start()
    {
        System.Console.WriteLine("Cpu start");
    }
    public void stop()
    {
        System.Console.WriteLine("Cpu stop");
    }
}
```

```
class Disk
{
    public void start()
    {
        System.Console.WriteLine("Disk start");
    }
    public void stop()
    {
        System.Console.WriteLine("Disk stop");
    }
}
```

```
class Mem
{
    public void start()
    {
        System.Console.WriteLine("Mem start");
    }
    public void stop()
    {
        System.Console.WriteLine("Mem stop");
    }
}
```

```
class Computer
{
    protected Cpu c = new Cpu();
    protected Mem m = new Mem();
    protected Disk d = new Disk();

    public void startCpu()
    {
        c.start();
    }

    public void startMem()
    {
        m.start();
    }
    public void startDisk()
    {
        d.start();
    }
    public void stopCpu()
    {
        c.stop();
    }
    public void stopMem()
    {
        m.stop();
    }
    public void stopDisk()
    {
        d.stop();
    }
}
```

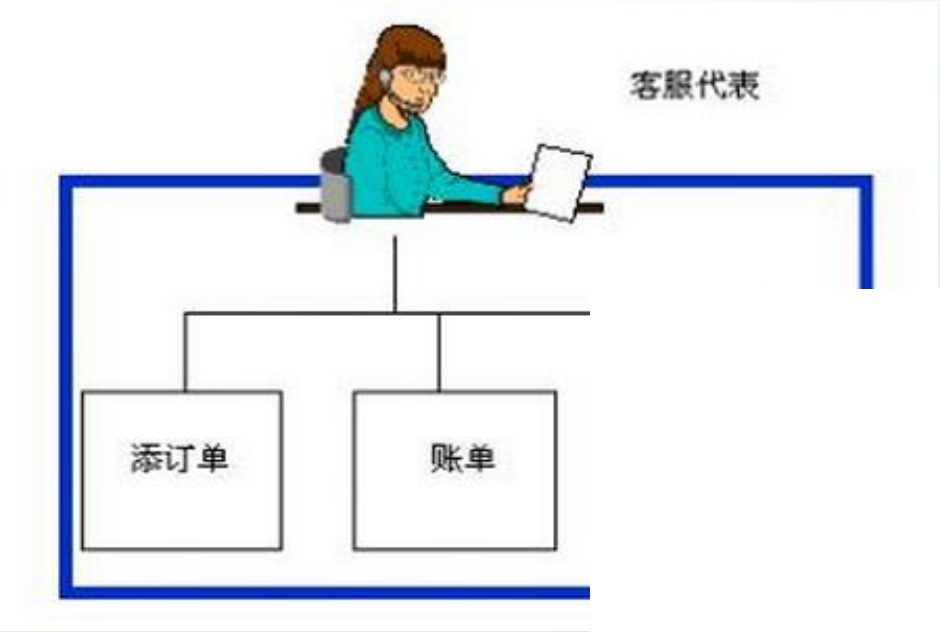
```
static void Main(string[] args)
{
    Computer com = new Computer();

    System.Console.WriteLine("====Computer Start=====");
    com.startCpu();
    com.startDisk();
    com.startMem();

    System.Console.WriteLine("====Computer Stop=====");
    com.startCpu();
    com.stopDisk();
    com.stopMem();


    System.Console.Read();
}
```

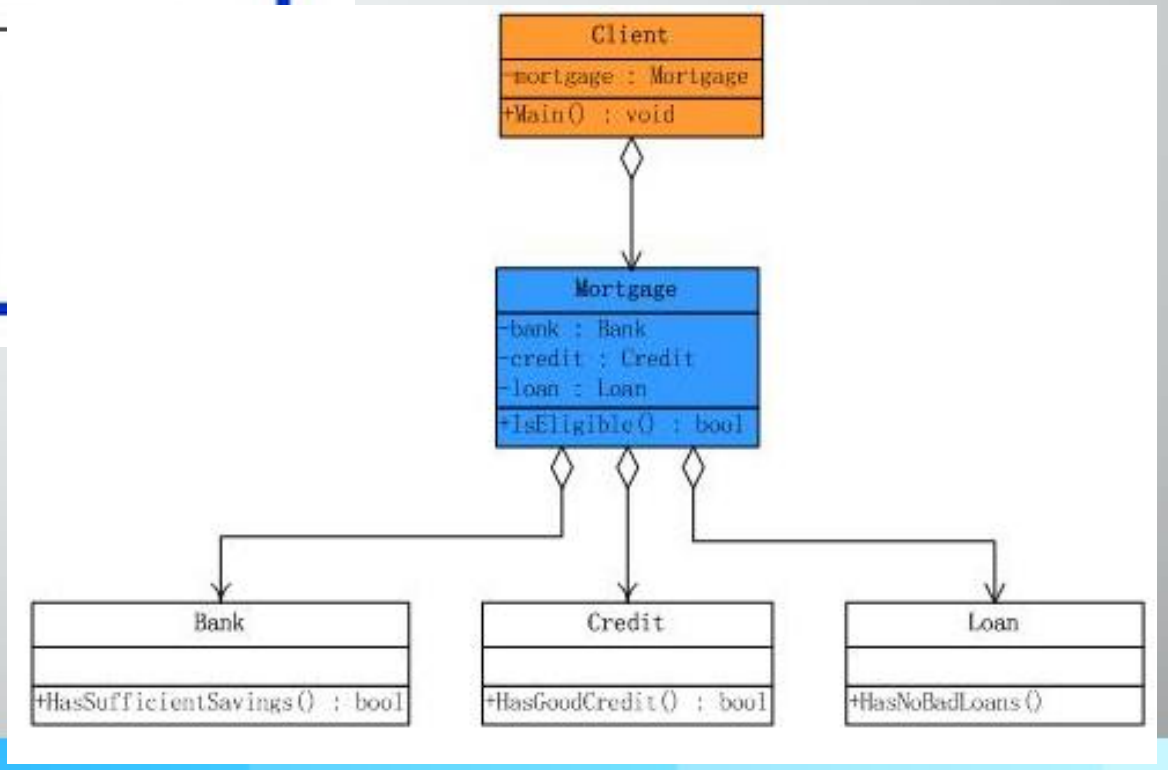
问题

- **背景**：用户希望使用一个比较复杂的子系统。但是用户不希望跟子系统的复杂的模块交互，也不想了解复杂的子系统内部的结构，且子系统结构变化后，不需要改变用户的使用方式。

外观模式（ Facade ）

生活中的例子


客服代表
添订单
账单


Client
-mortgage : Mortgage
+Main() : void
Mortgage
-bank : Bank
-credit : Credit
-loan : Loan
+IsEligible() : bool
Bank
+HasSufficientSavings() : bool
Credit
+HasGoodCredit() : bool
Loan
+HasNoBadLoans()

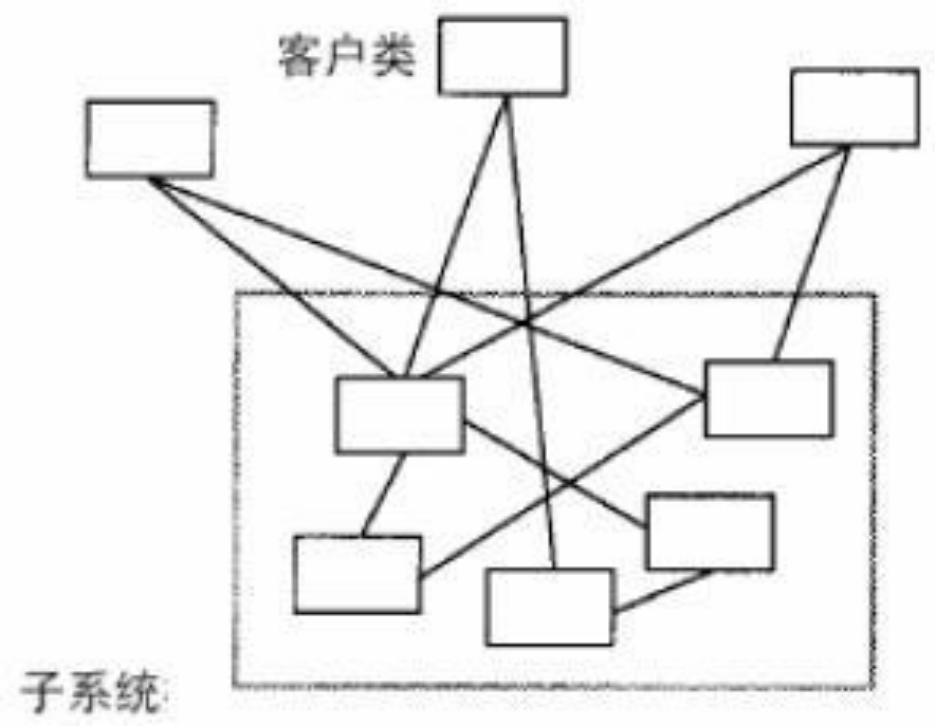
课程内容

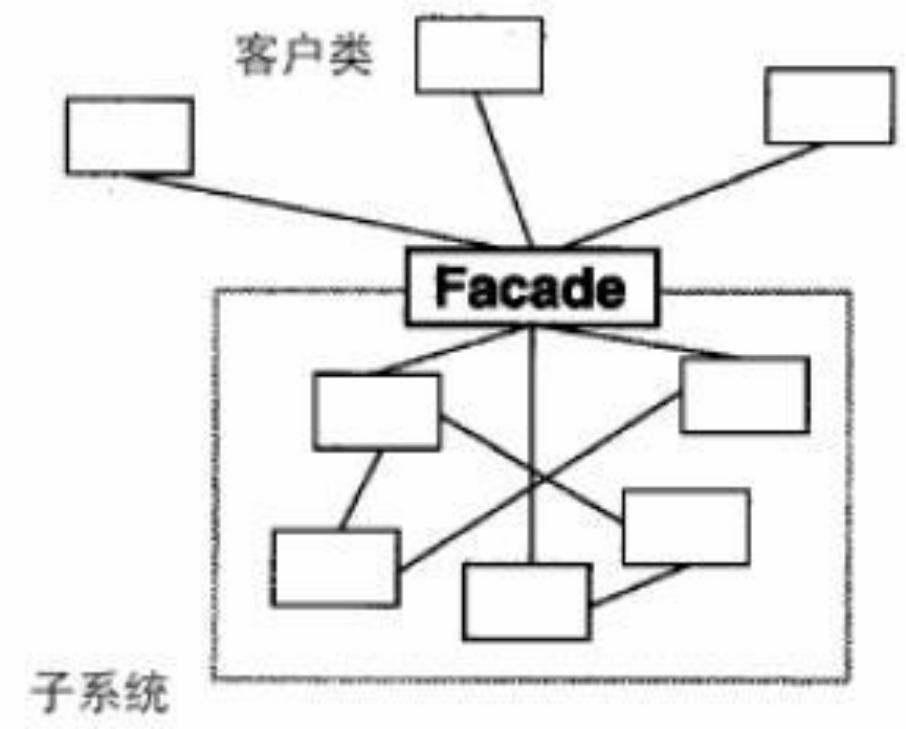
- 环境及问题
- 外观模式详解
- 外观模式实现
- 扩展练习

外观模式（Facade）

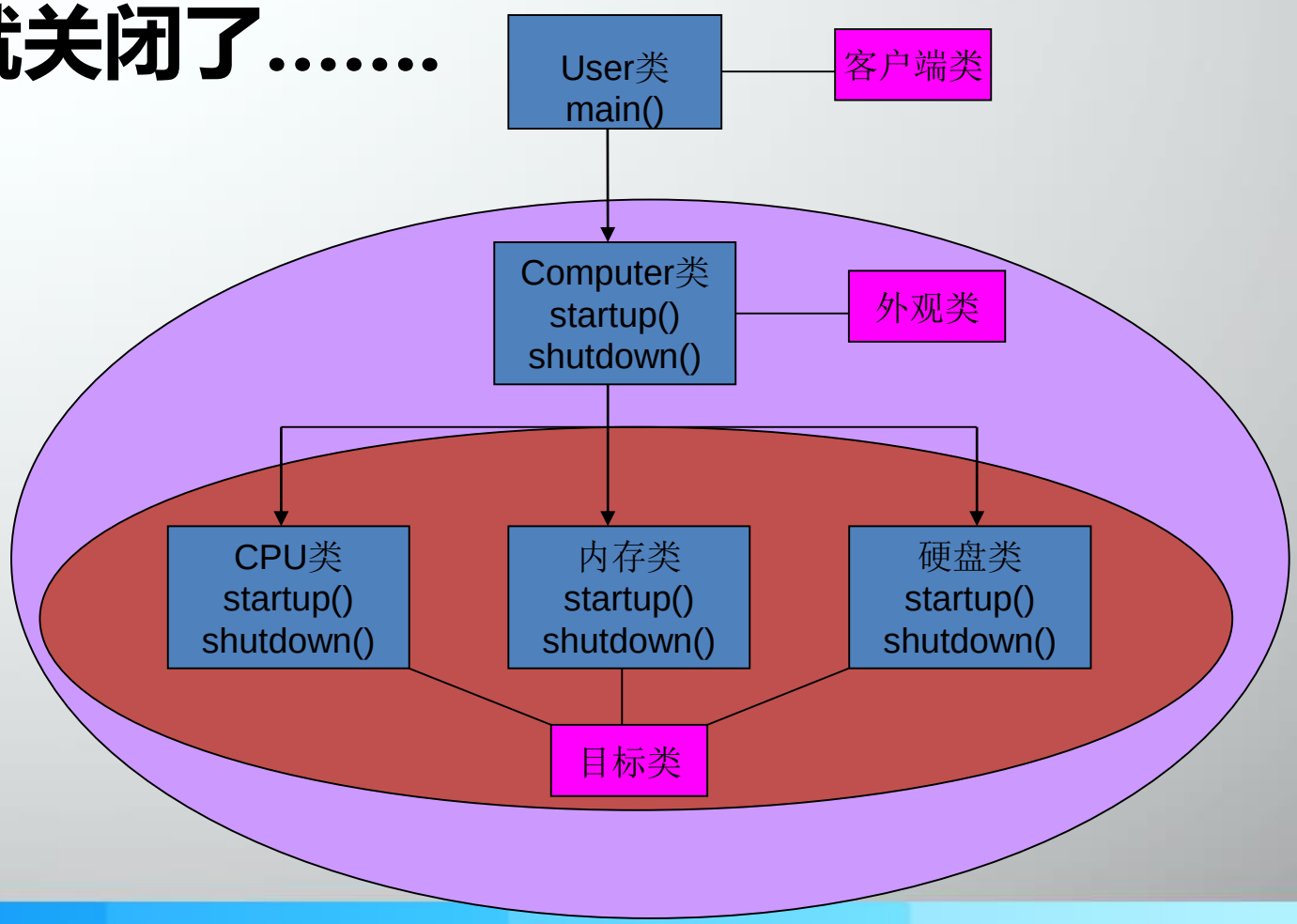
- 为子系统中的一组接口提供一个一致的界面，外观模式定义了一个高层接口，这个接口使得这一子系统更加容易使用。
- 使客户尽量少的与子系统内部的组件打交道，尽量维护子系统的统一的接口。
- 使系统的用户和系统通过façade类解耦。当子系统改变时，只要保证façade类的接口不变，用户的使用方式就无需改变。
- 外观模式的实现一般采用：在子系统外部封装façade类的方式实现。

- 角色：
  - 目标类：子系统类的合集。
  - 外观类：一个相对复杂的子系统类的外观类。
  - 客户端类：要使用子系统类中各个方法的用户类。





- 我们仅仅了解，当我按下电脑开关的时候电脑就打开了，当我关闭电脑开关的时候电脑就关闭了.....



课程内容

- 环境及问题
- 外观模式详解
- 外观模式实现
- 扩展练习

外观模式实现代码

```
class Cpu
{
    public void start()
    {
        System.Console.WriteLine("Cpu start");
    }
    public void stop()
    {
        System.Console.WriteLine("Cpu stop");
    }
}
```

```
class Disk
{
    public void start()
    {
        System.Console.WriteLine("Disk start");
    }
    public void stop()
    {
        System.Console.WriteLine("Disk stop");
    }
}
```

```
class Mem
{
    public void start()
    {
        System.Console.WriteLine("Mem start");
    }
    public void stop()
    {
        System.Console.WriteLine("Mem stop");
    }
}
```

```
class Computer
{
    protected Cpu c = new Cpu();
    protected Mem m = new Mem();
    protected Disk d = new Disk();

    public void start()
    {
        startCpu();
        startDisk();
        startMem();
    }

    public void stop()
    {
        stopCpu();
        stopDisk();
        stopMem();
    }

    public void startCpu()...

    public void startMem()...
    public void startDisk()...
    public void stopCpu()...
    public void stopMem()...
    public void stopDisk()...
}
```

```
static void Main(string[] args)
{
    Computer com = new Computer();

    System.Console.WriteLine("====Computer Start=====");
    com.start();


    System.Console.WriteLine("====Computer Stop=====");
    com.stop() ;



    System.Console.Read();
}
```

课程内容

- 环境及问题
- 外观模式详解
- 外观模式实现
- 扩展练习

扩展说明

- 外观模式为复杂子系统提供了一个简单接口，并不为子系统添加新的功能和行为。
- 外观模式实现子系统与客户之间的松耦合关系。
- 外观模式没有封装子系统的类，只是提供了简单的接口。如果应用需要，它并不限制客户使用子系统类。因此可以在系统易用性与通用性之间选择。
- 外观模式注重的是简化接口，它更多的时候是从架构层次去看整个系统，而并非单个类的层次。

扩展练习

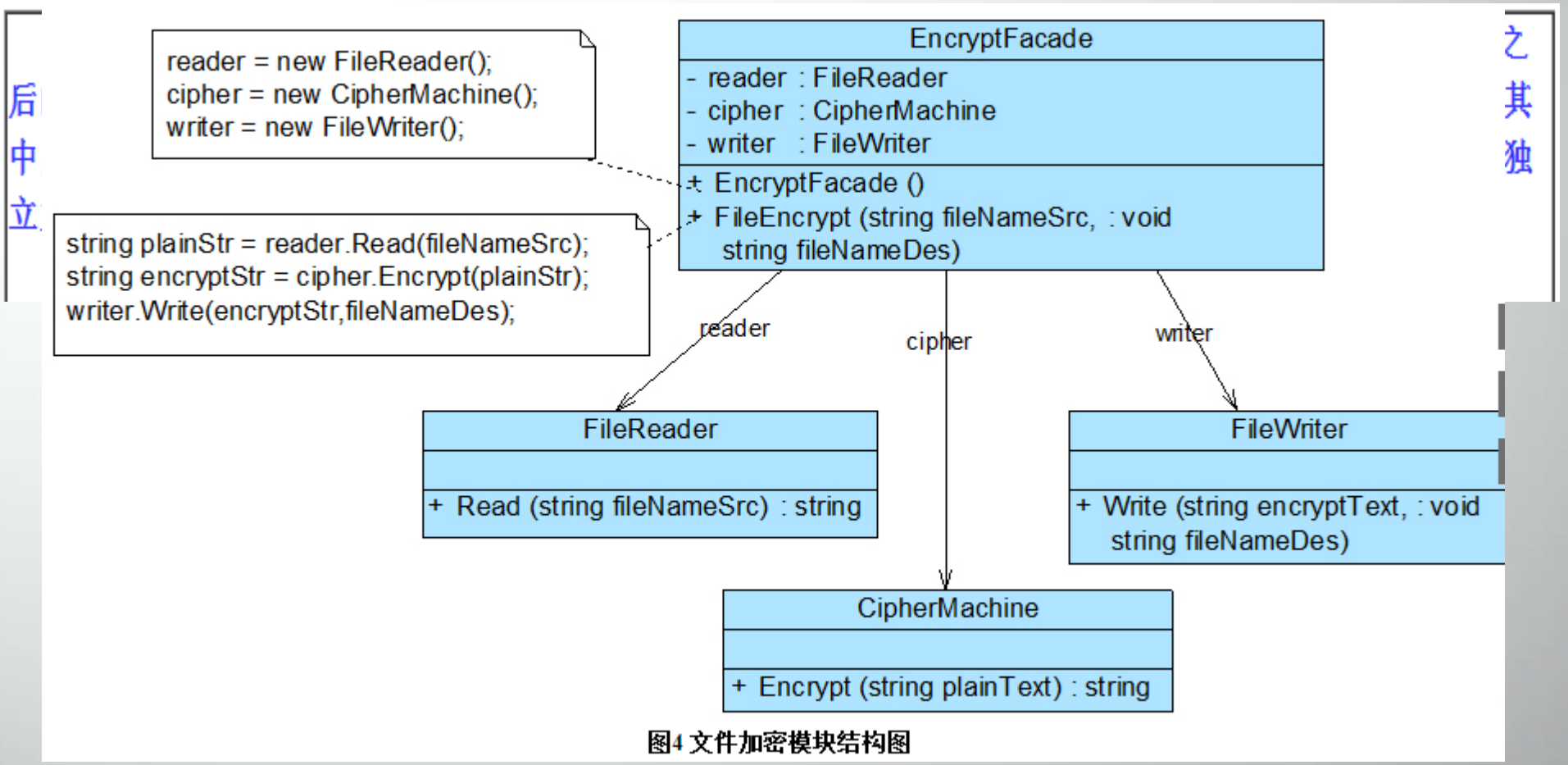


图4 文件加密模块结构图

小结

- 外观模式解决的问题是“如何简单的使用一套子系统”
- 外观模式的解决方案是利用对处理逻辑的封装产生外观类

Thank You，谢谢！