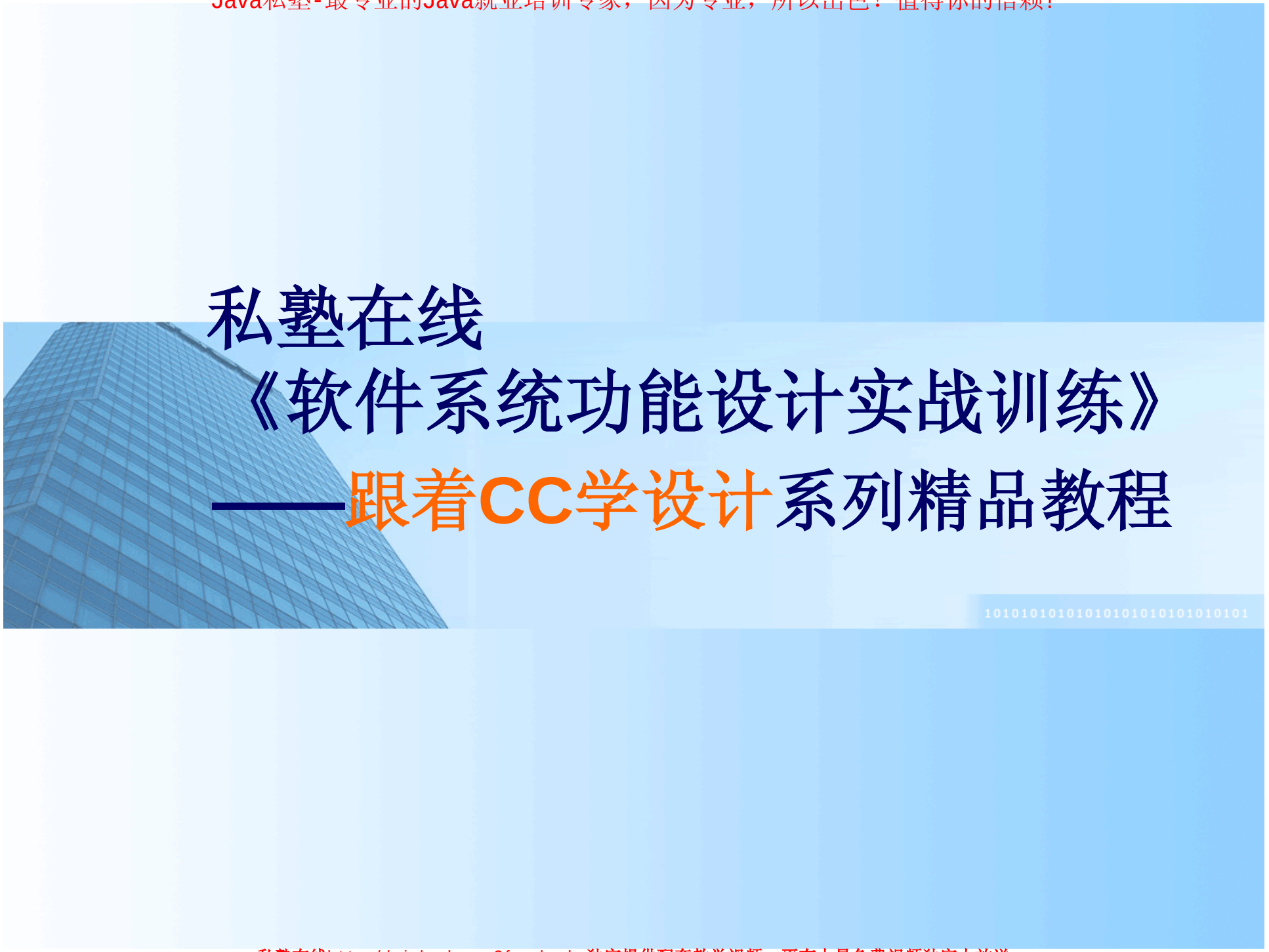


私塾在线
《软件系统功能设计实战训练》
——跟着CC学设计系列精品教程

本周设计作业

- 本周设计作业的项目背景：

  订单管理——来自韩国ST电子商务系统

- 学习目标：

  （1）学习多模块的业务状态跟踪处理的方式

  （2）学习处理多模块循环引用、交叉引用的问题

  （3）综合应用简单工厂模式、代理模式、中介者模式，包含着这些模式但不限于这些模式

该设计方式可应用于多种有类似功能的系统，比如：OA中的各种审批单流程.....

- 基本功能

  1：有效性核实

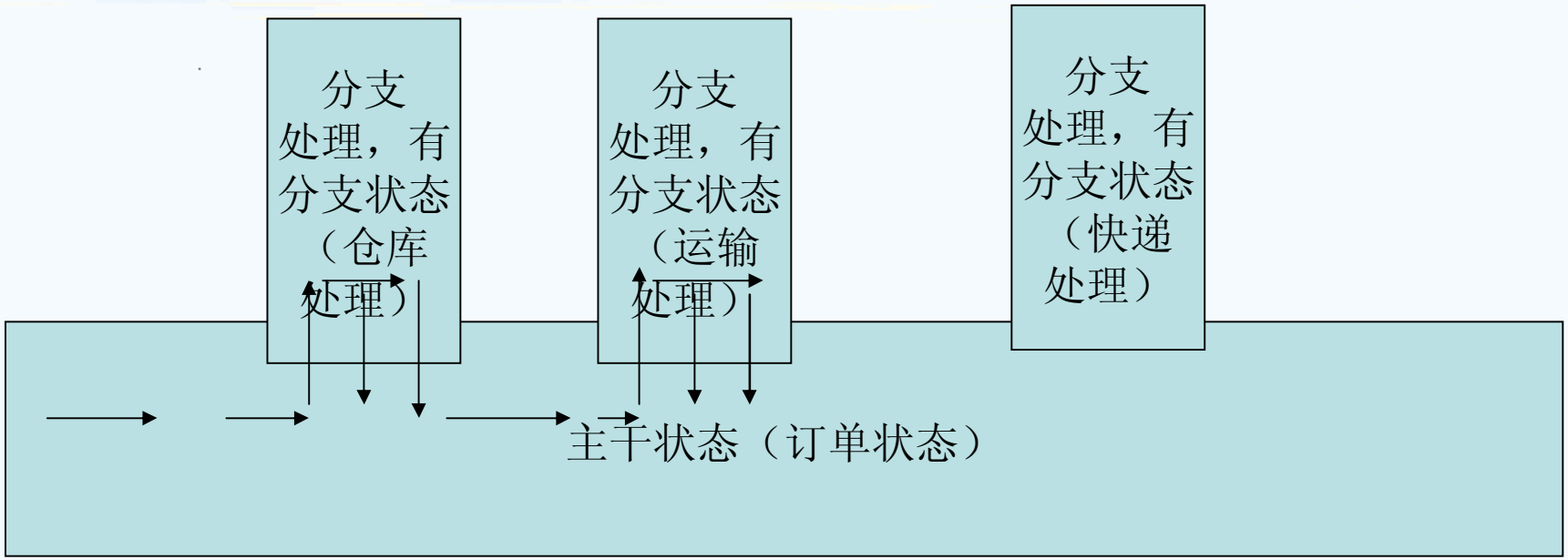
  2：分单，分给不同的业务部门或者仓库去处理

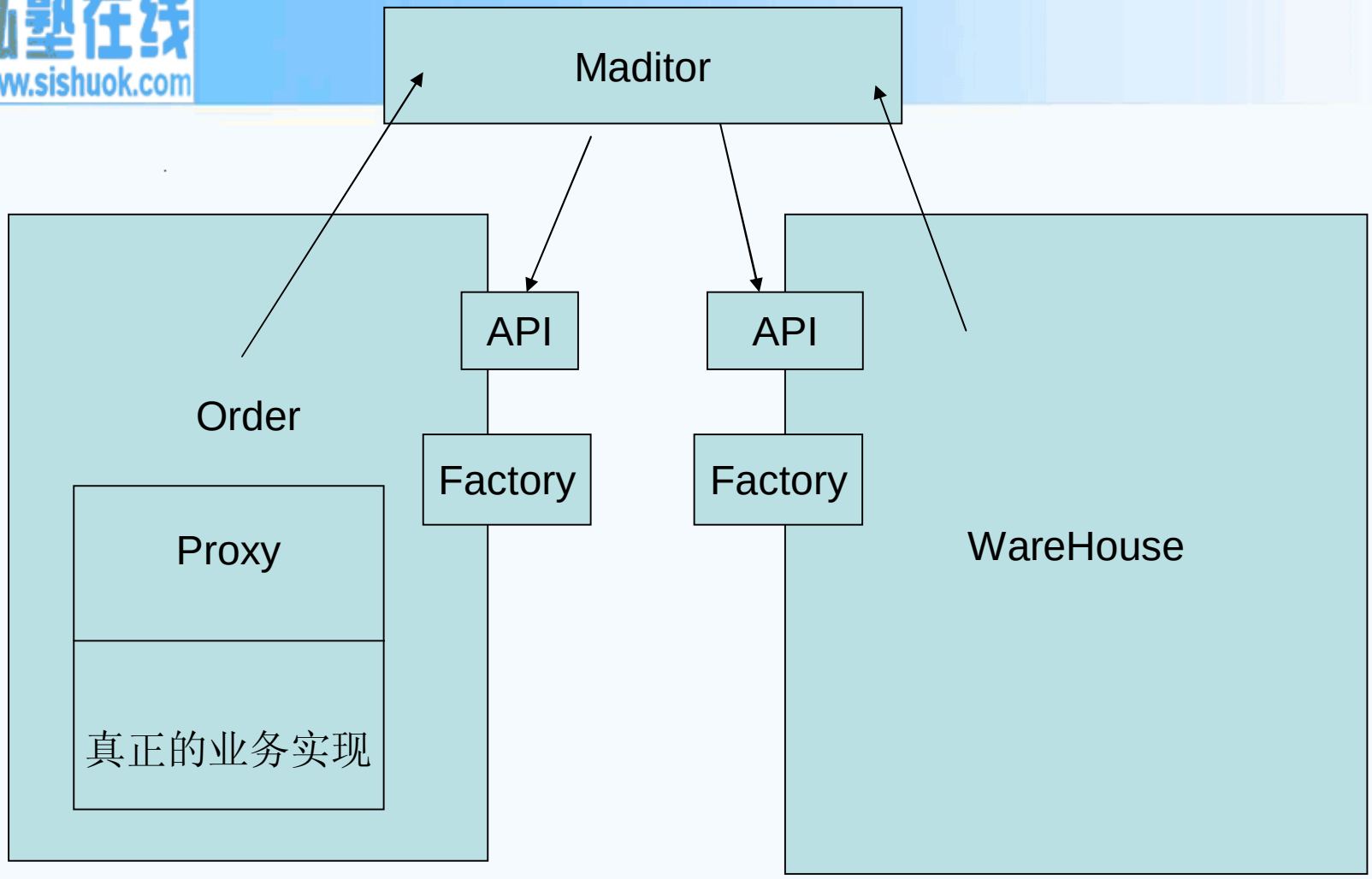
  3：仓库备货、出库

  4：运输部门负责运送

  5：快递部门负责送货，或者是自提

  6：收款结算





本周设计作业

- 基本要求
  1：订单处理模块需要能够察看到当前订单处理的阶段和状态
  2：仓储、运输、快递和财务模块都有自己的业务状态
  3：任何阶段都要支持退货的处理
  4：每个模块处理的时候，都需要检查订单状态适合本模块处理

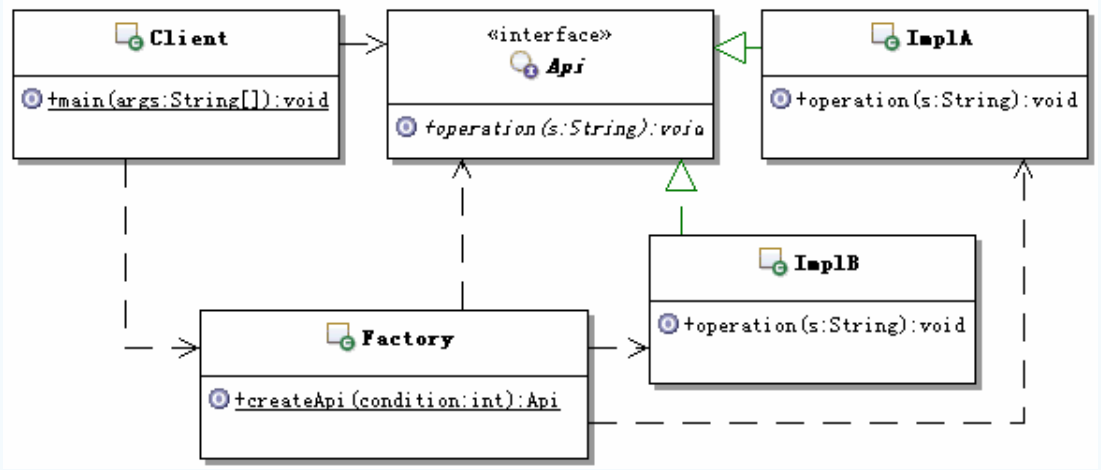
- 作业要求
  1：在老师给定的概要代码基础上，实现上述基本要求的功能设计
  2：重点在接口和模块交互设计上，无须关注具体实现
  3：对于每个api，可以适当写点样例代码，能够调用运行更佳
  4：考虑合理的结构，职责的划分，以及设计模式的合理使用

初识简单工厂

- 定义

  提供一个创建对象实例的功能，而无须关心其具体实现。被创建实例的类型可以是接口、抽象类，也可以是具体的类。

- 结构和说明



Api：定义客户所需要的功能接口
Impl：具体实现Api的实现类，可能会有多个
Factory：工厂，选择合适的实现类来创建Api接口对象
Client：客户端，通过Factory去获取Api接口对象，然后面向Api接口编程

简单工厂的知识要点

- 简单工厂的知识要点

1：简单工厂位于对外提供接口的模块内

2：简单工作的主要功能就是用来创建对象实例，被创建的对象可以是接口、抽象类或是普通的类

3：简单工厂可以实现成为单例，也可以实现成静态工厂

4：简单工厂的内部实现，主要是做“选择合适的实现”，实现是已经做好的，简单工厂只是来选择使用即可

5：简单工厂在进行选择的时候，需要的参数可以从客户端传入、配置文件、或者是运行期程序某个运行结果等

6：如果使用反射+配置文件的方式，可以写出通用的简单工厂

思考简单工厂

- 简单工厂的本质

  简单工厂的本质是：**选择实现**

- 何时选用简单工厂

1：如果想要完全封装隔离具体实现，让外部只能通过接口来操作封装体，那么可以选用简单工厂，让客户端通过工厂来获取相应的接口，而无需关心具体实现

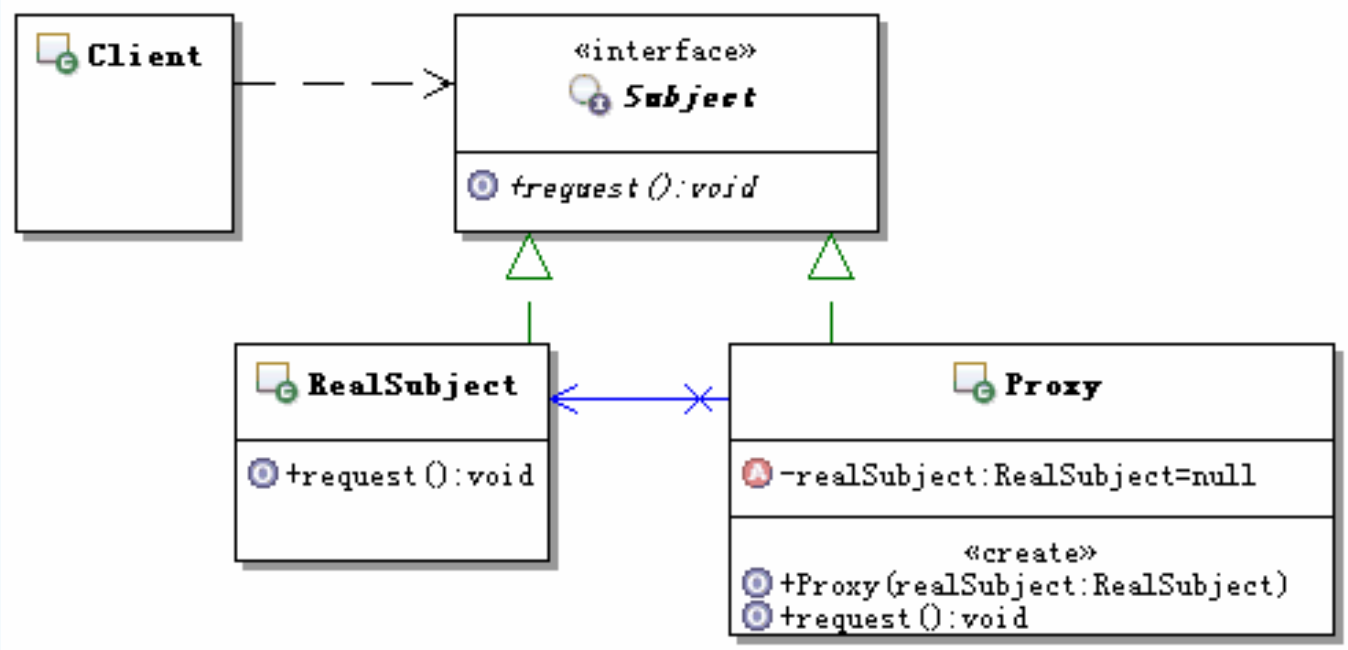
2：如果想要把对外创建对象的职责集中管理和控制，可以选用简单工厂，一个简单工厂可以创建很多的、不相关的对象，可以把对外创建对象的职责集中到一个简单工厂来，从而实现集中管理和控制

初识代理模式

- 定义

  为其它对象提供一种代理以控制对这个对象的访问。

- 结构和说明



代理模式的知识要点

- 代理模式的知识要点

1：代理模式是通过创建一个代理对象，用这个代理对象去代表真实的对象，客户端得到这个代理对象过后，直接当真实对象去操作

2：代理对象夹在客户端和被代理的真实对象中间，相当于一个中转，那么在中转的时候就可以做很多工作，代理模式的功能也主要通过在中转的时候进行实现，比如在中转前后附加很多操作等

3：代理分成很多种类，开发中最常用的是虚代理和保护代理

4：虚代理是：刚开始创建一个“虚”代理对象返回给客户端，直到客户端要真正使用这个对象的时候，代理才真正去创建这个对象，从而变相实现一个延迟装载，节省资源

5：保护代理是一种控制对原始对象访问的代理，保护代理会检查调用者是否具有请求所必需的访问权限，如果没有相应的权限，那么就不会调用目标对象，从而实现对目标对象的保护

6：Java中的静态和动态代理，不过Java的动态代理目前只能代理接口，基本的实现是依靠Java的反射机制和动态生成class的技术，来动态生成被代理的接口的实现对象

思考代理模式

- 代理模式的本质

  代理模式的本质是：**控制对象访问**

- 何时选用代理模式

1：需要为一个对象在不同的地址空间提供局部代表的时候，可以使用远程代理

2：需要按照需要创建开销很大的对象的时候，可以使用虚代理

3：需要控制对原始对象的访问的时候，可以使用保护代理

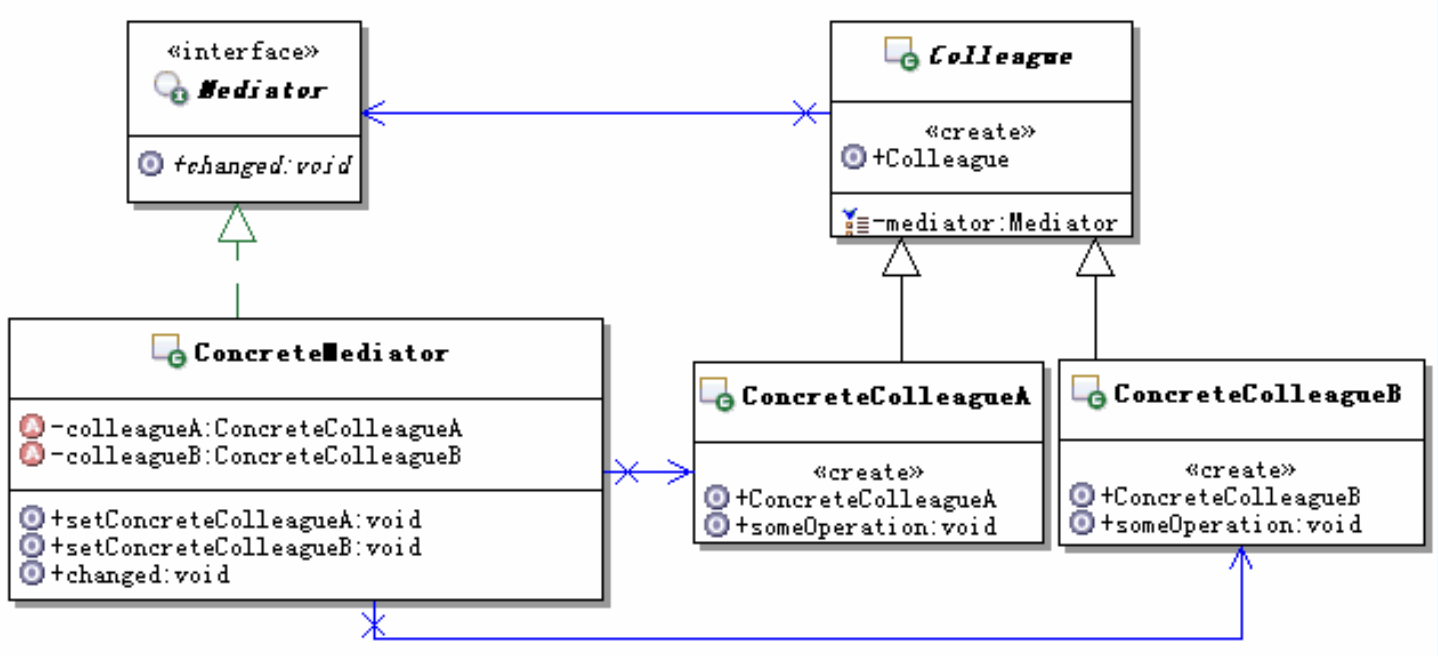
4：需要在访问对象的时候执行一些附加操作的时候，可以使用智能指引代理

初识中介者模式

- 定义

  用一个中介对象来封装一系列的对象交互。中介者使得各对象不需要显式地相互引用，从而使其耦合松散，而且可以独立的改变它们之间的交互。

- 结构和说明



中介者模式的知识要点

- 中介者模式的知识要点

1：中介者主要用来封装对象之间的交互，把对象之间的交互全部集中到中介者对象里面，所有的对象就只是跟中介者对象进行通信，相互之间不再有联系，从而松散对象间的耦合，并对交互关系进行统一的管理

2：在实现中介者的时候，如果中介者的实现只有一种，而且今后也没有扩展的需要，那么可以不要Mediator

3：同事和中介者必须有关系，首先是同事对象需要知道中介者对象是谁；反过来，中介者对象也需要知道相关的同事对象，这样它才能与同事对象进行交互。也就是说中介者对象和同事对象之间是相互依赖的

4：同事和中介者通信的方式，通常的实现方式，一种是在Mediator接口中定义一个特殊的通知接口，作为一个通用的方法，让各个同事类来调用这个方法；另一种实现方式是可以采用观察者模式，把Mediator实现成为观察者，而各个同事类实现成为Subject，这样同事类发生了改变，会通知Mediator

5：在实际应用开发中，经常会简化中介者模式，来使开发变得简单，我们称其为广义中介者模式

思考中介者模式

- 中介者模式的本质

  中介者模式的本质是：**封装交互**

- 何时选用中介者模式

1：如果一组对象之间的通信方式比较复杂，导致相互依赖、结构混乱，可以采用中介者模式，把这些对象相互的交互管理起来，各个对象都只需要和中介者交互，从而使得各个对象松散耦合，结构也更清晰易懂

2：如果一个对象引用很多的对象，并直接跟这些对象交互，导致难以复用该对象。可以采用中介者模式，把这个对象跟其它对象的交互封装到中介者对象里面，这个对象就只需要和中介者对象交互就可以了