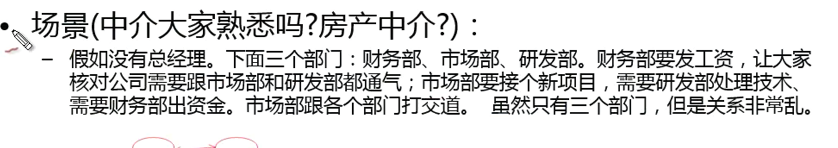
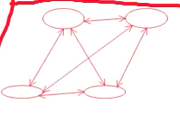
中介者模式

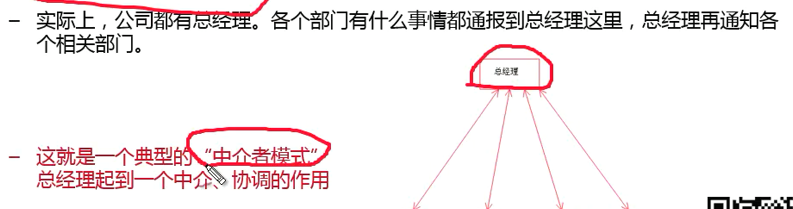




没有总经理协调 这三个部门直接沟通

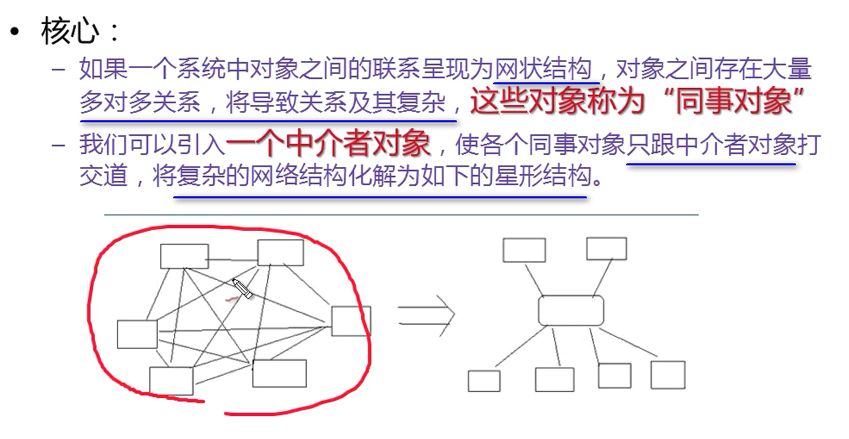
市场部有需求直接给研发部 需要发工资 就联系财务部 非常乱套





总经理是中介协调作用

就把复杂的网状多对多的管写 变成了多个一对多的关系【有点类似于数据库设计中的 多对多通过第三方表 双向一对多 这个第三方表 就是类似于中介者模式中的中介者】



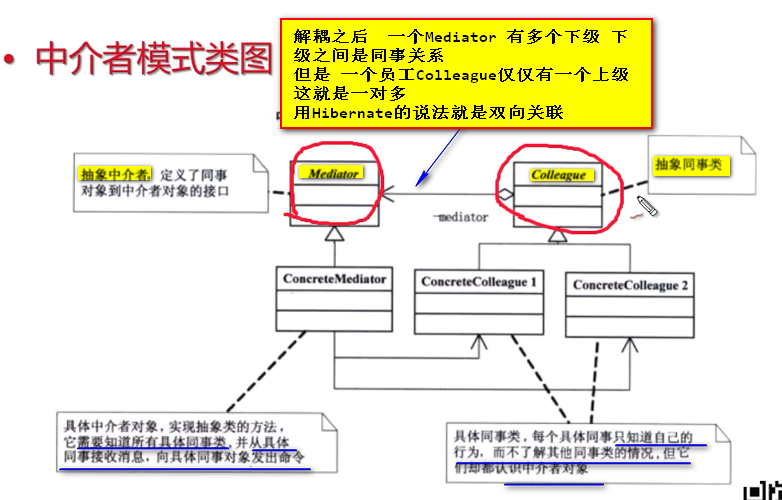
最后 这个模式有两个角色 一个是中介者Mediator 另一个是同事 Colleague

其他部门我不管 我**只对我的上级负责（总经理就是中介者）** 这就是中介者模式

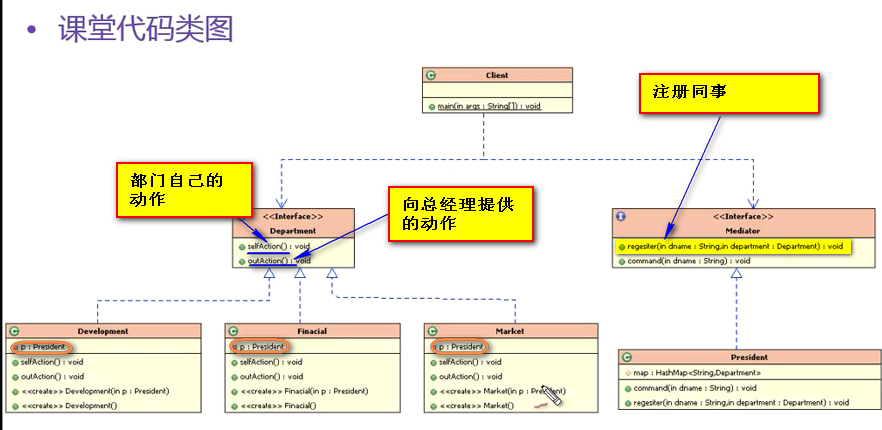
网状结构 化解为星形结构

如果对象关系简单 千万不要使用中介者模式 ---- 否则反而会变得复杂

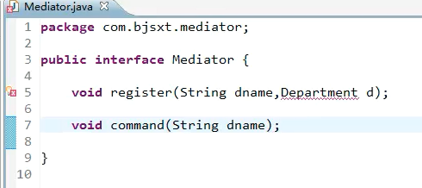
----- 比如 一个单位有三个人 设置两个总经理（两个中介者）



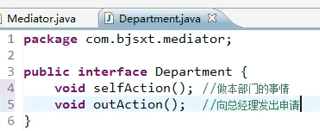
可以看到 colleague之间没有线路关联



首先实现Mediator接口



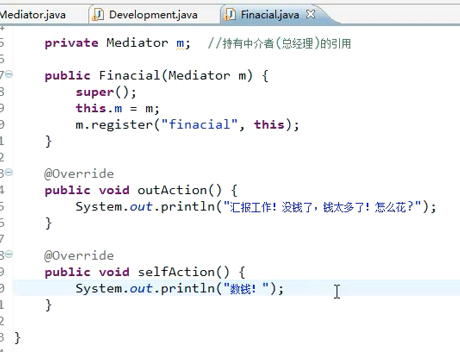
定义Department的接口



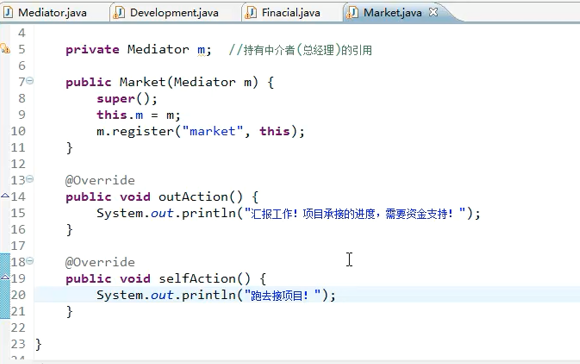


这个是研发部

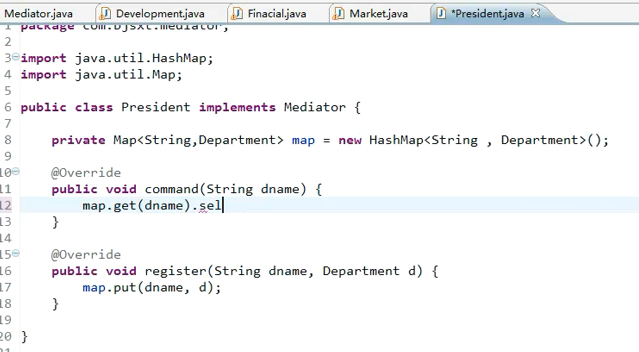
下面是财务部



最后做一个市场部



定义具体的总经理

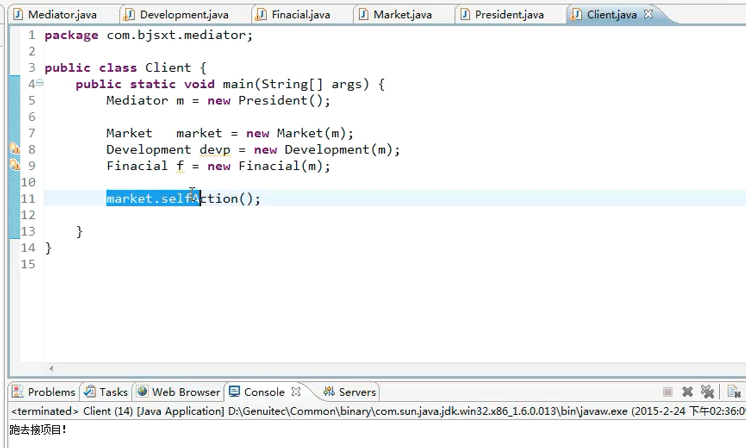


重点在OuterAction中 ----- 就是 我要给总经理汇报信息 ---- 总经理接到之后 给出Command 通过Command传入的参数 找到具体的其他同事 【有点类似于责任链】



这个command是具体的业务逻辑 这里面调用自己的动作

测试一下



发现市场部需要钱了 怎么交互？ 按照中介者模式 不能直接和财务部交互 --- 没办法 告诉总经理 ----- 总经理在哪？ 就在我们的同事类的成员中 ---- 在outerAction中直接调用即可 ---- 体现了中介者的职责





【缺少了一个完整的环节 财务部数钱完成之后 应该发回给市场部 那就应该在Finaicil中的outerAction中通过中介者调用Market】

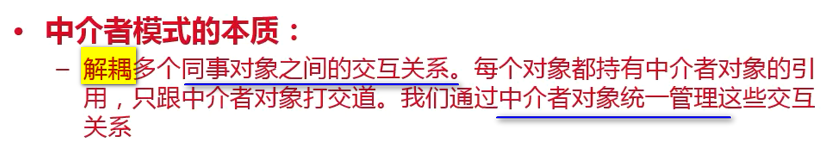
【中介者模式 和 代理模式 类似在 都是中介对象 替别人做事

中介者和观察者 类似在 观察者模式中的观察者和中介者模式中的中介者都持有所有的成员 也就是分别持有被观察者和同事列表 并且 持有人都可以向被持有人发号施令 ---- 但是反过来 同事可以持有中介者 但是 被观察者就是被动的 不能持有中介者

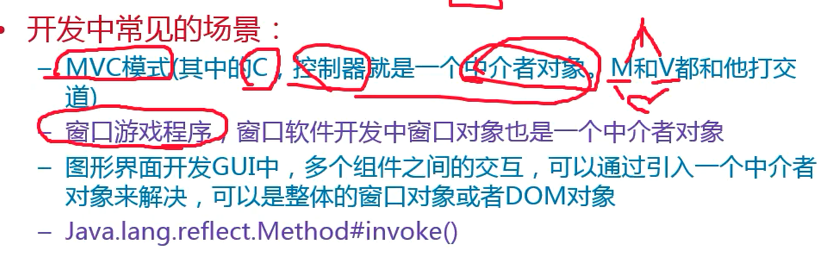
中介者模式 ----- 仅仅去代理委托人 原封不动做事情 不太侧重增强什么功能 但是 代理模式可以不是这样 非常侧重增强】

【因为同事类 也就是多的那一方关联了中介者类 所以 可以构造的时候 就直接注册 不用像观察者模式中 client统一注册】

【看使用的场景 代理模式 代理的就是一个代理模板 生成一个类的代理类 但是 中介者模式 要传递各种平行类之间的消息 并做协调】



MVC 模式 Model 和View之间都会和Controller打交道 直接不直接交互 直接和C交互



窗口对象往往也是中介者对象

反射方法invoke() 里面就会有中介者模式的影子 ---- 里面和Object对象交互的时候 就是中介者模式

【中介者模式 就是解耦复杂的网状通信系统的各个成员 ---- 代理模式 一对一的增强功能】