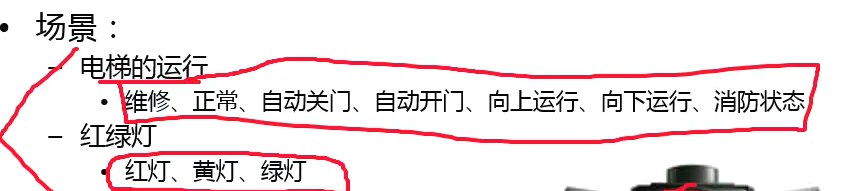
状态模式 State

状态模式遇到这种场景 **不同的状态对应不同的行为** ---- 把不同的状态和行为封装到一起

红灯是一个状态 停 绿灯走 黄灯注意



一个对象有多种状态 不同的状态对应不同的行为 可以考虑使用状态模式

还有就是公文审批 ---- 公文审批的过程是责任链模式

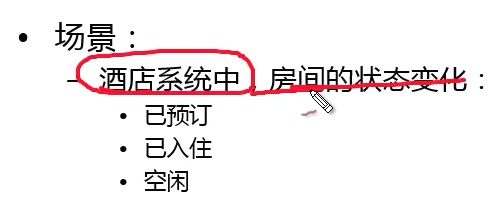
但是 公文本身是有状态的 ---- 上次是审批的过程 ---- 假条本身有状态 -----

未提交 提交 未批准 批准 重写 这些状态 有不同的状态 ---进行不同的动作

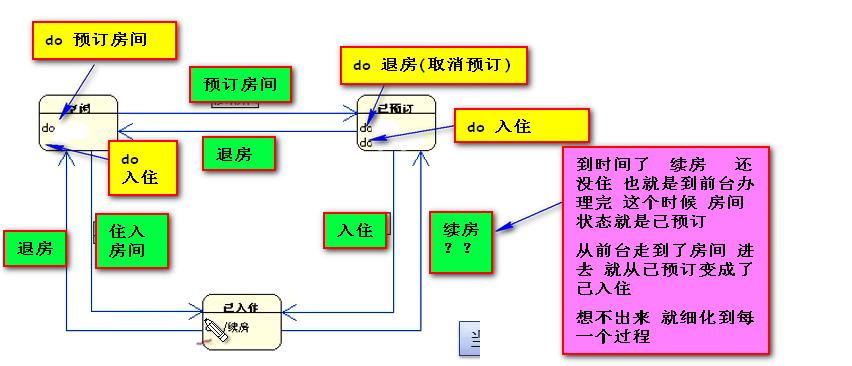
一个模块中用到了不同的模式 需要多种设计模式



所以 在我们实际开发中 总结了状态模式



【实际上 应该是有一个状态转换图才行 分析状态转换】



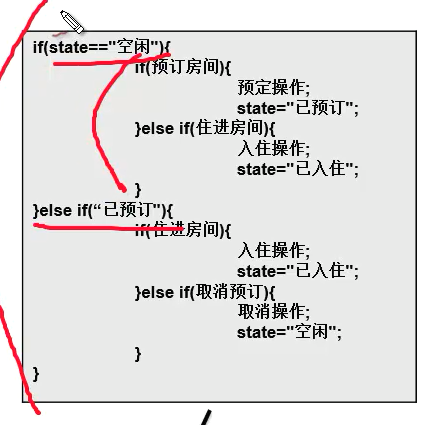
【常理来讲 已入住怎么编程已预订呢？想一个场景】

UML中的状态图

【从哪一个状态出发的动作 就算做这个装下的动做】

【空闲的时候 可以入住 ----- 所以 已入住的时候 可以】

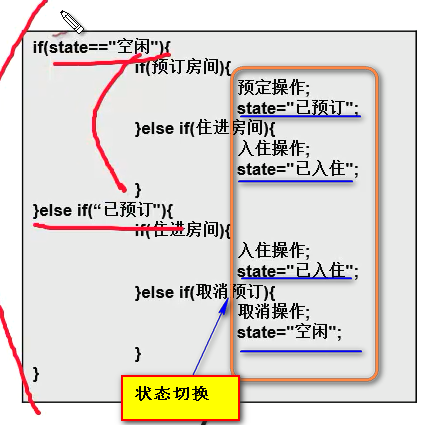
使用if else切换

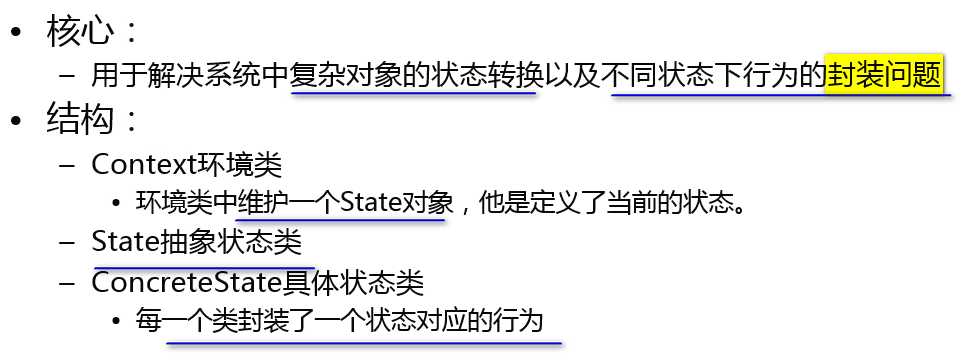
 ---- 第二层操作的if是动作 第一层是状态

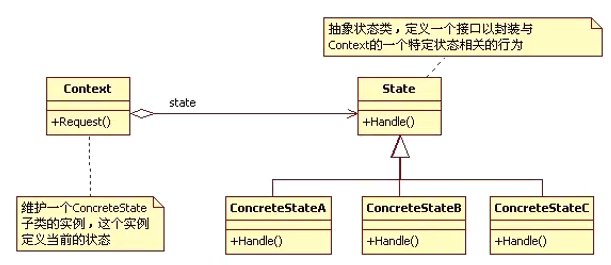
遇到if else语句 并且是不同状态的切换 应该使用状态模式

【之前的策略模式 也是切换算法 策略 ----- 但是 这个切换更多是人为的切换

但是 状态模式 切换的是状态 ---- 不是人为的切换 而是 状态有状态转换图 依据客观事物的变化过滤来切换状态 状态机】







核心 解决状态转换 以及 不同状态下行为的封装

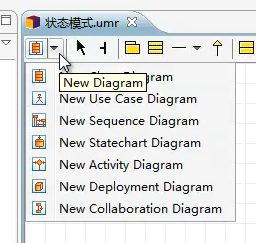
策略模式 就是 context给了哪个策略 就使用的是哪个策略 和这个很像

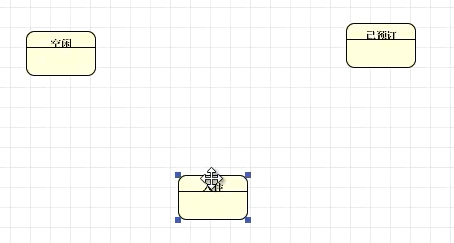
Context中维护了**当前状态State**

State是一个抽象类

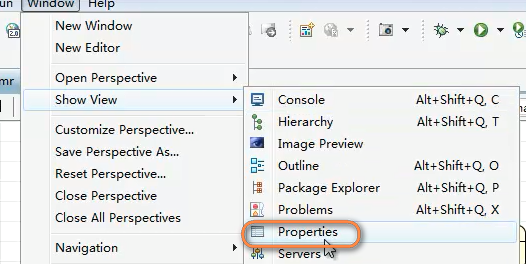
通过context管理不同状态之间的切换 ---- 每一个State中封装了该State下的行为 所以是handle

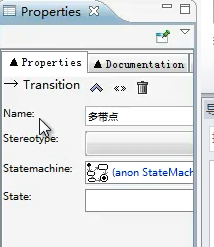
现在使用eclipse绘制状态转换图

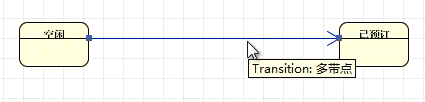
 StateChart Diagram

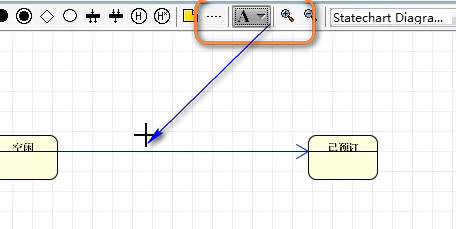


打开property

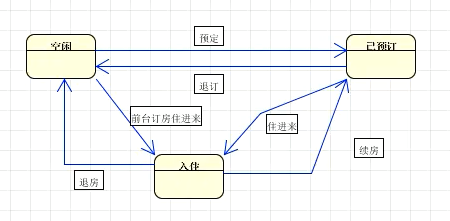


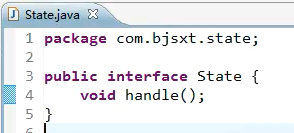
 这个地方就会可以填写





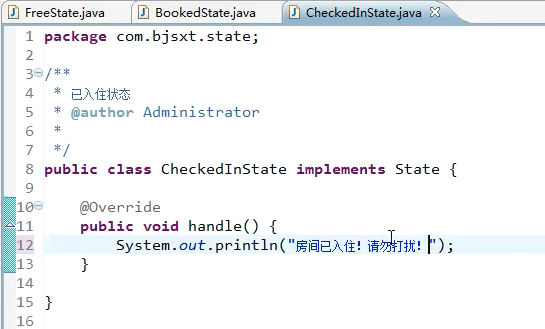
这块从入住到已预订 老师说是续房【前面已经分析】





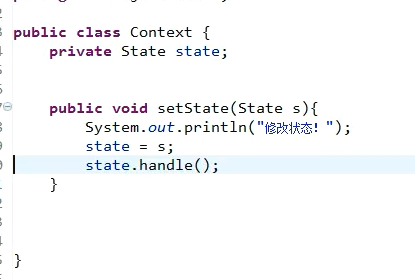




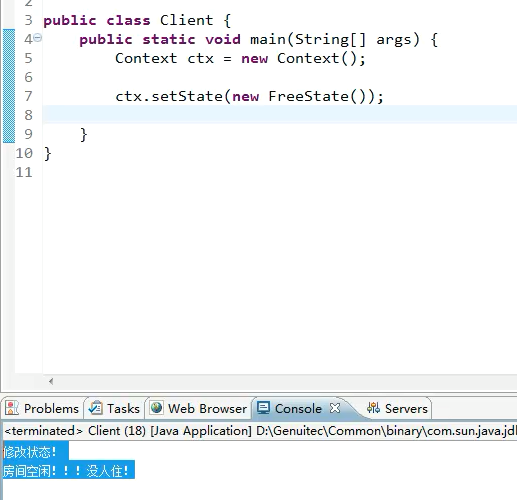


这是我们子状态的类

还有一个context类

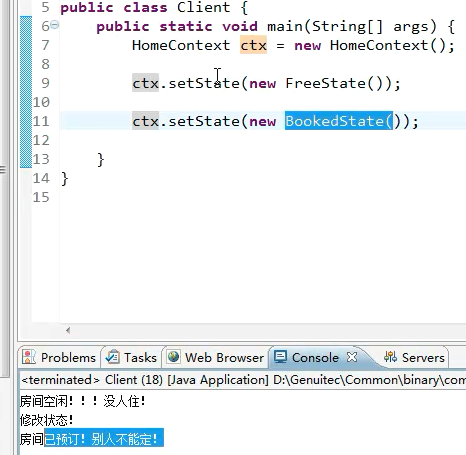
 ---- Context维持不同状态的切换

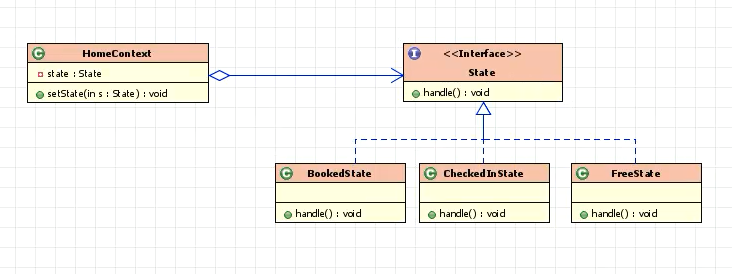
银行系统 这个Context就相当于银行账号 根据金额的不同进行切换状态 【按照后面的状态机 状态是在内部确定的 state熟悉实际上是不应该提供setter 来给client进行调用的！ ----- 现在高淇这种做法 完全是策略模式 写的是错误的】





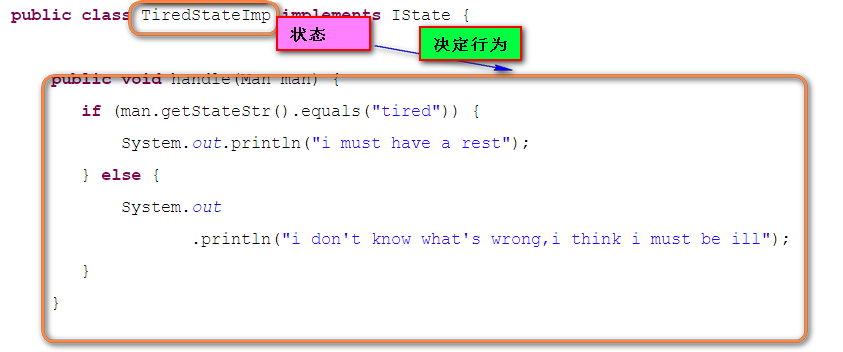
修改了一下类名

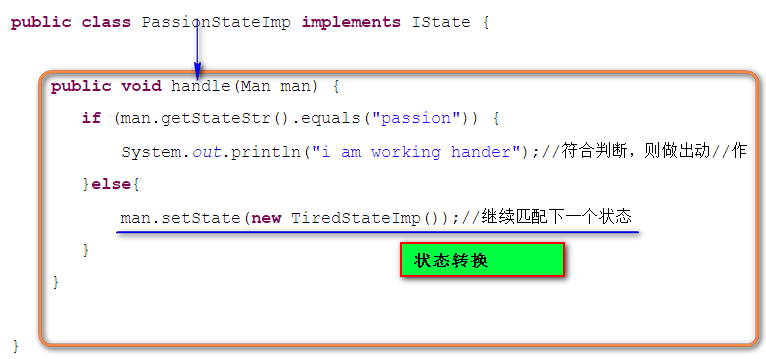
【代码上看 没有状态转换 也是 状态转换 要人来控制 ---- 这种做法是错误的 下面的链接做法对】



【这里面的状态转换是复杂的 双向的 比CC的那个状态要复杂】

<http://blog.csdn.net/turkeyzhou/article/details/2792840>





这样定义好下一个状态是什么

