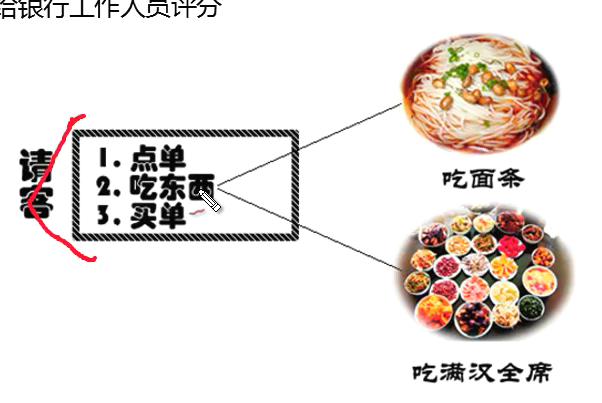
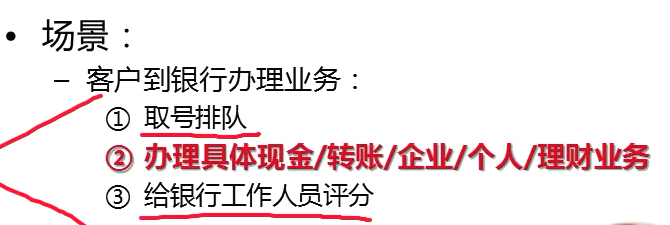
模板方法模式 template method

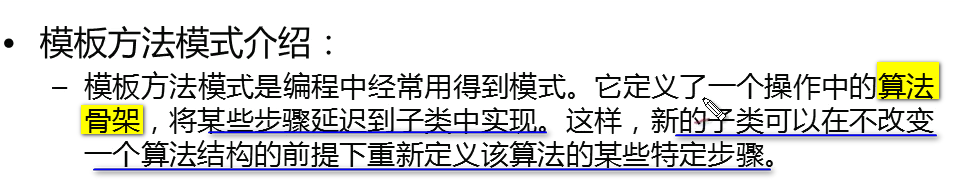


点单 和买单 可以想象出来 但是 具体吃什么不知道

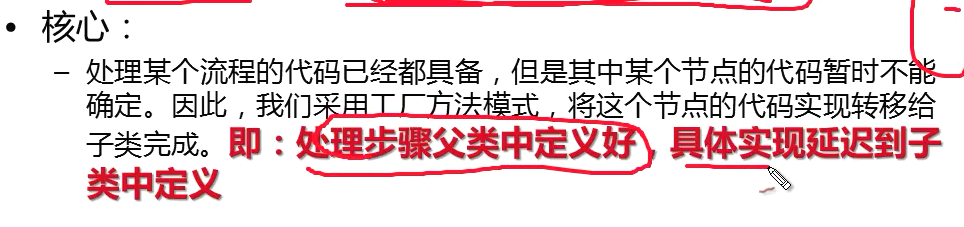


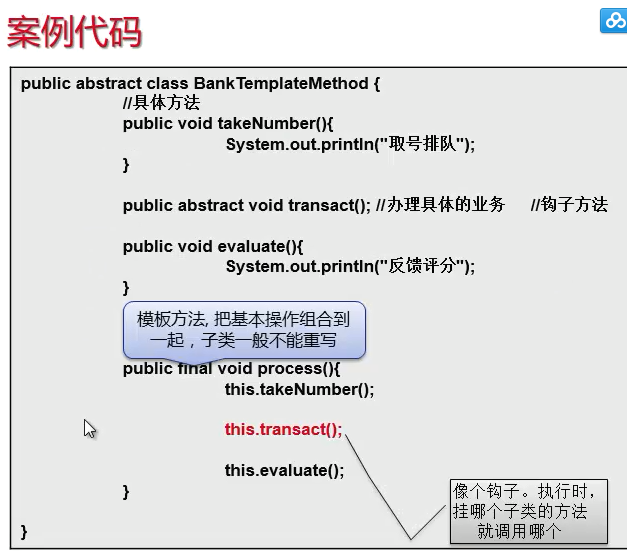
流程骨架是知道的 具体办什么业务未知

这时候 使用模板方法模式



点菜之后 才知道吃什么 延迟到子类实现



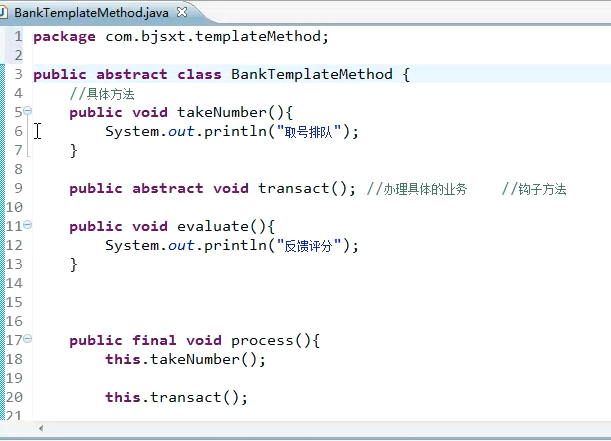


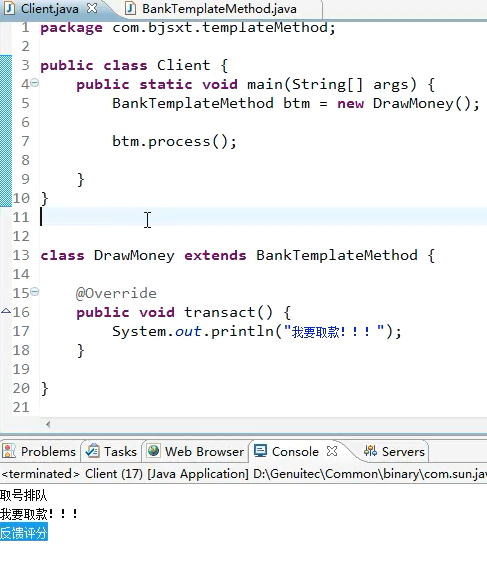
就是模拟银行取号的流程

transact 是抽象方法

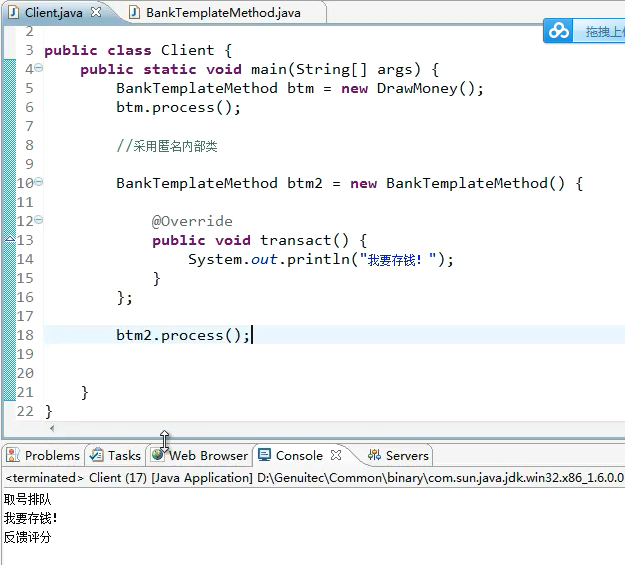
---- 像钩子一样 挂到哪个子类 就执行哪个子类的方法

骨架是不能变的 所以 使用final





要是存钱 如下 匿名内部类



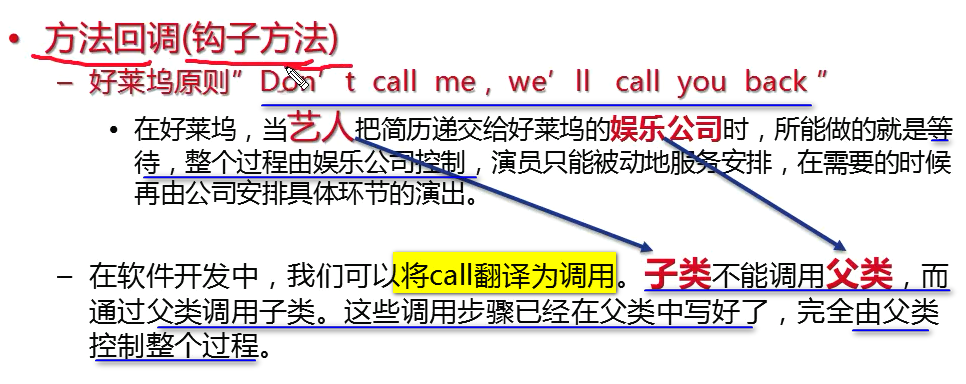
骨架都是一样的



实际上就是一个多态的应用

上面的调用关系 本质上就是多态

钩子方法 也成为回调



演员只能被动安排

艺人相当于子类 娱乐公司相当于父类 ----- 主动权归谁的问题

【



这个就是 匿名内部类实现的这个方法 就会被父类的骨架回调 ----- 这个就是钩子方法的实现 理解回调 理解实现

对于我们root代码

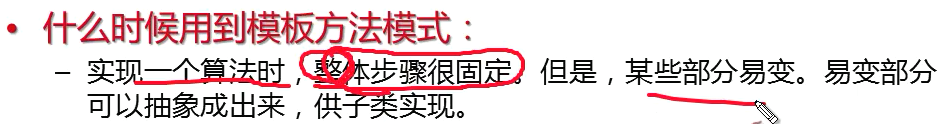


实际上 这个是使用CC的研磨设计模式中的方式 实现的回调 ---- 直接传递子类接口的匿名子类

实际上 这个匿名子类没有骨架算法 骨架算法都在被传递的方法中完成的

 这个就是模板方法模式的变种 实际上 就是JDK中的回调 也就是 你的子类没有权利决定谁来调用我 只能由传递进去的一方来决定 ------ 也就是具体的业务实现不是你来调用 是传递进去之后的方法来调用 ---- uml上面讲 就是 依赖你这个匿名对象的类来决定 ---- 这个依赖匿名对象的类的方法 就反过来来调用这些实现 ---- 所以叫回调 JDK回调

这样 这里面的】





这个模式很重要