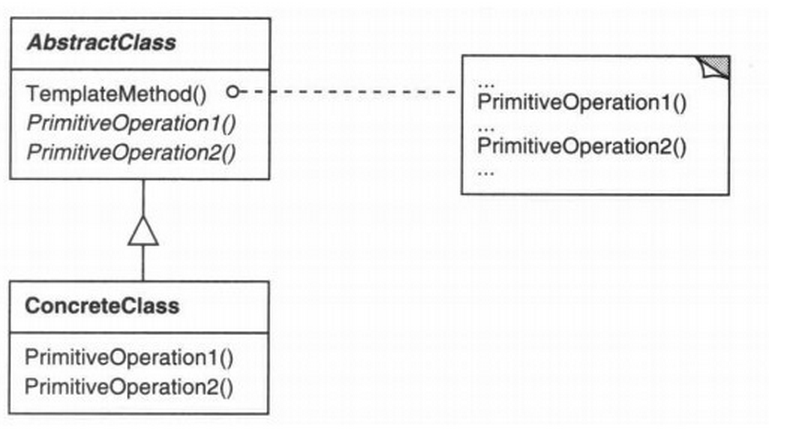
1. 概述

在面向对象开发过程中，通常我们会遇到这样的一个问题：我们知道一个算法所需的关键步骤，并确定了这些步骤的执行顺序。但是某些步骤的具体实现是未知的，或者说某些步骤的实现与具体的环境相关。

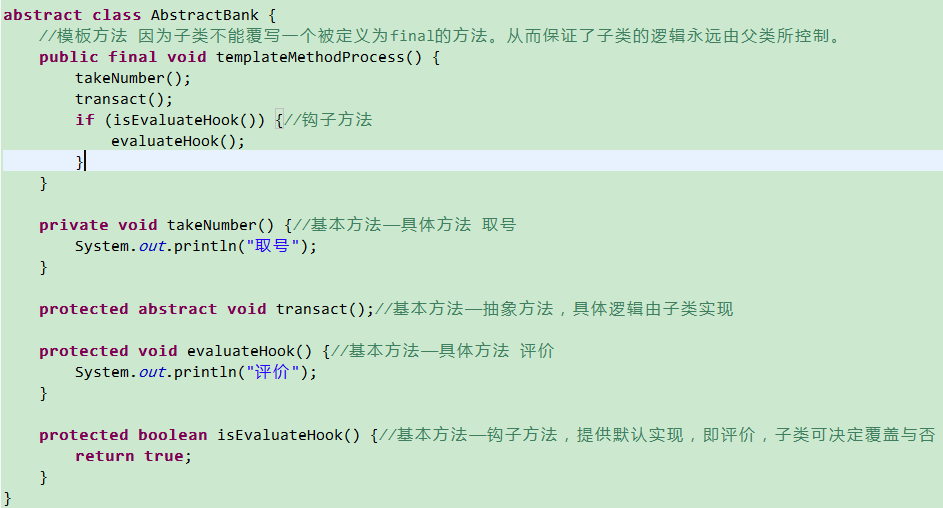
1. 模板方法：定义一个操作中的算法的骨架，而将一些步骤延迟到**子类**中。 T模板方法使得子类可以不改变一个算法的结构即可重定义该算法的某些特定步骤。
   1. 模板方法模式是**基于继承的**代码复用基本技术(策略模式基于对象组合)，在模板方法模式中，可以将相同的代码放在父类中，而将不同的方法实现放在不同的子类中。
   2. 在模板方法模式中，我们需要准备一个抽象类，将部分逻辑以具体方法以及具体构造函数的形式实现，然后声明一些抽象方法来让子类实现。不同的子类可以以不同的方式实现这些抽象方法，从而对剩余的逻辑有不同的实现，这就是模板方法模式的用意。
2. 结构

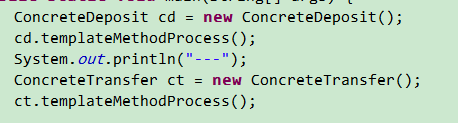
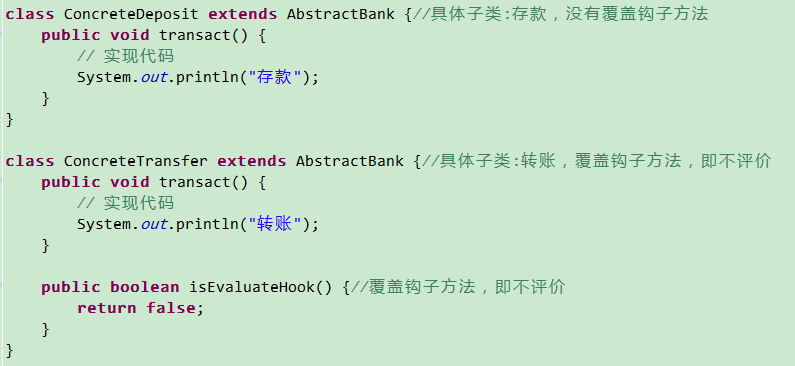


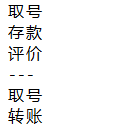
注：AbstractClass中也可以有具体的方法和钩子hook方法(钩子方法默认不做任何事，子类可以视情况覆盖，不一定是boolean方法)

1. 例：银行业务办理流程：在银行办理业务时，一般都包含几个基本固定步骤:取号排队->办理具体业务->对银行工作人员进行评分。

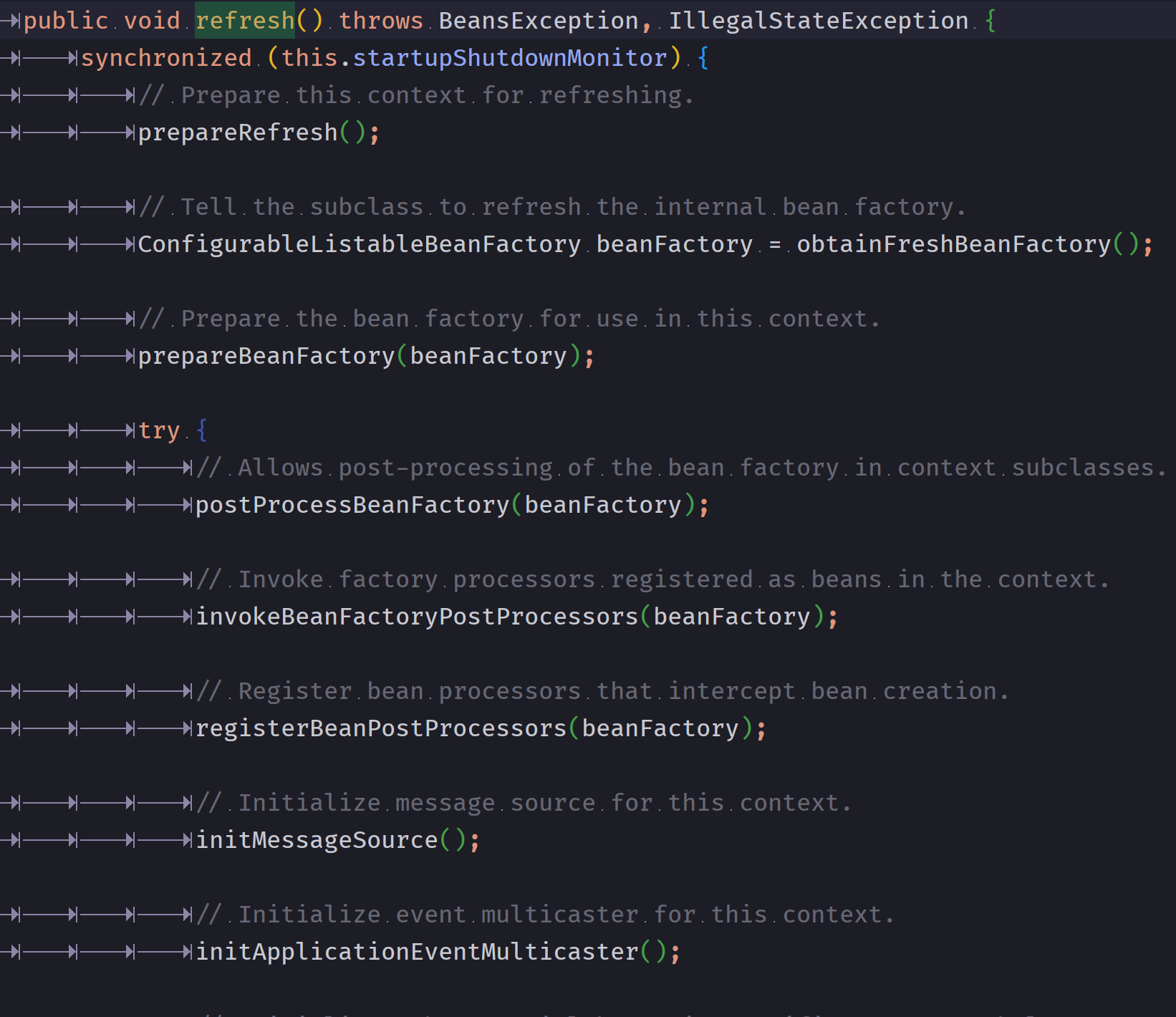
取号取号排队和对银行工作人员进行评分业务逻辑是一样的。但是办理具体业务是个不相同的，具体业务可能取款、存款或者转账。





输出：

1. 模板方法模式优缺点
   1. 模板方法模式的优点
      1. 模板方法模式在一个类中形式化地定义算法，而由它的子类实现细节的处理。其基本思想：算法只存在于一个地方，也就是在父类中，容易修改。需要修改算法时，只要修改父类的模板方法或者已经实现的某些步骤，子类就会继承这些修改
      2. 模板方法是一种代码复用的基本技术。它们在类库中尤为重要，它们提取了类库中的公共行为。既统一了算法，也提供了很大的灵活性。父类的模板方法确保了算法的结构保持不变，同时由子类提供部分步骤的实现
      3. 一般模板方法都加上**final**关键字，防止子类重写模板方法
      4. 模板方法模式导致一种反向的控制结构，这种结构有时被称为“好莱坞法则” ，即“别找我们，我们找你”通过一个父类调用其子类的操作(而不是相反的子类调用父类)，通过对子类的扩展增加新的行为，符合“开闭原则”
   2. 模板方法模式的缺点:
      1. 每个不同的实现都需要定义一个子类，这会导致类的个数增加，系统更加庞大，设计也更加抽象，但是更加符合“单一职责原则”，使得类的内聚性得以提高。
2. 详见：<http://blog.csdn.net/hguisu/article/details/7564039>
3. 开发中常见场景
   1. Junit单元测试
   2. Spring中JDBCTemplate、HibernateTemplate
   3. 数据库访问的封装
   4. Spring中org.springframework.context.support.AbstractApplicationContext#refresh方法使用了模板方法模式



postProcessBeanFactory方法默认什么也不做，是钩子方法

