实验报告

|  |  |
| --- | --- |
| 一、实验目的及要求  综合运用合适的设计模式，按照题目要求完成类图设计和Java程序代码。 | |
| 二、实验内容  某网红小店（AStore）以地方特色小吃担担面、豆花鱼、赖汤圆等小吃为主要商品，由于纯手工的制作工艺，每天只有 100 份进行售卖。现在为该小店设计一套点餐系统，顾客在扫码点餐后（可以通过微信、支付宝、银联等不同客户端扫一扫进入）可以在程序界面看到每种小吃的单价和当前剩余的份数。当有一个顾客点了一份小吃后，相应的其他顾客的界面能看到对应小吃的数量变化。同时，点餐结束后支付时，系统能根据当前扫码的客户端选择对应的支付方式（包括微信、支付宝、银联等多种支付）。为了方便小店扩充小吃菜单，以及未来接入不同的支付手段，在设计时需要充分考虑扩展性。请选择合适的设计模式实现该系统，给出类图设计和相应的示例代码。  注意：  1. 该题目中设计的微信、支付宝、银联等支付方式只是展示用，不用实际去对接真实的支付接口。  2. 相应的应用程序也不需要实现 UI 界面，只需要命令行展示功能流程即可。  3. 用户点餐是通过不同支付子系统的 app 进入，可以以不同的对象来代替，不用实际实现对应的子系统。  4. 类图设计要附上对题目的分析和选择特定设计模式的原因。 | |
| 三、实验过程记录 | |
| 类图 | 根据题目描述，首先在‌扫码进入系统并选择支付方式‌这一部分，使用‌策略模式比较合适。因为用户可以通过不同的方式（微信、支付宝、银联）进入系统，但后续的操作流程（如点餐、付款）是一致的。虽然状态模式和策略模式的结构很相似，但是这里可以体现出两者的区别，并不是系统自动转换状态，也不存在不同状态下进行的操作不同，所以选择策略模式。  其次，在‌菜单显示与选择菜品‌的流程中，当用户选择菜品时，菜单上的数量需要相应更新。这适合使用‌观察者模式‌，因为多个用户（观察者）能够实时收到菜品数量（被观察者）的变化信息。而为了表示菜品的不同状态（如剩余数量），可以使用‌状态模式‌。因为状态模式通常涉及状态的自动转移和基于状态的行为变化，而这里的状态（即菜品数量）在减少到0时会触发状态转移（可售状态转换为售罄状态），所以我一开始打算使用状态模式设计菜品状态部分。但是后来因为时间和水平有限，并没有很好地完善状态转移部分的代码。  为了方便未来的‌菜单与支付方式扩展，应该对菜单和支付方式引入抽象层。对于菜单，可以使用简单‌工厂模式‌来创建不同的菜品对象，根据名称返回不同的实例。但其实简单工厂模式对开闭原则的支持并不是最好的，在增加新的具体产品类时，需要在工厂类当中修改根据参数返回实例的代码，而且可能会增加过多的分支语句结构，使工厂类的职责过重，维护不便。但简单工厂模式还是保留了工厂模式的最大好处，就是用户只需要关心菜品的名称和价格，并不需要关注实例的创建过程。对于支付方式，同样也引入抽象策略类，在扩展时只需要让新的具体策略类实现抽象策略类接口‌，因为所有支付方式的目标都是完成支付，只是实现方式不同。 |
| 实验过程记 录 | 我原先的设计思路是，用策略模式实现用户选择不同的客户端进入系统并采用相应的付款方式，用观察者模式实现菜单的实时更新通知，用状态模式实现菜品数量减为零时自动转为不可售状态；又因为简单工厂扩展困难，抽象工厂模式在增加新的产品等级结构时会违背开闭原则，所以设想采用工厂方法模式，再将多例类融合进工厂方法，实现菜品实例的创建数量限制，当菜品数量到达100时则不再创建新实例，因为创建了实例而不使用会浪费系统资源，占用内存空间。然而因为时间和水平有限，这些设计思路并没有完全实现，在后面的附录部分将会展现按照初始设计编写的程序代码，可能并不完善，但也是过程的记录。所以根据实际情况，类的设计简化如下：  ‌抽象策略类（PaymentStrategy）‌：定义支付方法的接口，包含pay()方法。  ‌具体策略类（WeChatPay、Alipay、UnionPay）‌：实现PaymentStrategy接口，分别代表微信、支付宝、银联支付。  ‌上下文类（OrderContext）‌：维护了一个PaymentStrategy对象，用于执行支付操作。在构造时通过策略模式选择支付方式。  ‌被观察者类（MenuItem）‌：表示菜单上的菜品，包含名称、单价、剩余数量等属性，以及通知观察者的方法。  ‌观察者类（OrderObserver）‌：表示用户界面，观察MenuItem的变化，并更新显示。  ‌工厂类（MenuItemFactory）‌：用于创建MenuItem对象，可以根据菜品名称生成对应的菜品实例。  ‌主程序类（Main）‌：模拟用户操作，包括扫码进入系统、选择菜品、支付等。 |
| 总结 | 通过这个大作业的设计过程，我体会到了各种设计模式的综合应用，在动手实践中明白了软件设计需要考虑的问题是非常多的。首先经过逐步分析需求，我学会了比较不同设计模式的特点，选择我认为最适用的模式。从创建型的工厂模式，到对象行为型的状态模式、策略模式、观察者模式，我更加理解了它们的优缺点和适用环境。希望以后能接触更多的实际问题，提高自己的软件设计和编程能力。 |
| 附录 | |  | | --- | | 屏幕截图 2024-12-10 175247 | | 屏幕截图 2024-12-10 175316 | | 屏幕截图 2024-12-10 175346 | | 屏幕截图 2024-12-10 175425 | | 屏幕截图 2024-12-10 175504 | | 屏幕截图 2024-12-10 175533 | | 屏幕截图 2024-12-10 175552 | | 屏幕截图 2024-12-10 175619 | |
| 课程截图 | |  | | --- | | 屏幕截图 2024-12-10 160731 | | 屏幕截图 2024-12-10 160909 | | 屏幕截图 2024-12-10 160842 | | 屏幕截图 2024-12-10 160814 | | 屏幕截图 2024-12-10 161039 | | 屏幕截图 2024-12-10 161112 | | 屏幕截图 2024-12-10 161421屏幕截图 2024-12-10 161348 | | 屏幕截图 2024-12-10 161512 | | 屏幕截图 2024-12-10 161627 | | 屏幕截图 2024-12-10 161740 | |