外卖软件设计

褚有刚*

2020 年 6 月 23 日

1 引言

在实现该软件的过程中，选用了 Java 编程语言，图形化界面采用 JavaFX 框架，数据库选用 MySQL 和 MyBatis 框架，其他基础功能通过网络，集合，反射，序列化等 JavaSE 技术实现。

目录结构

- code：源代码
- jars：可执行 jar 包 (server.jar 为服务器，client.jar 为客户端)
- db.sql：创建数据库及相应数据表的 SQL 语句
- report.pdf：报告文件

运行环境说明

- 运行平台：Windows
- MySQL 版本：8.0.16(数据库名为 oosc，用户名为 root，密码为 123456)
- JDK 版本：1.8

2 目标系统的分析与设计

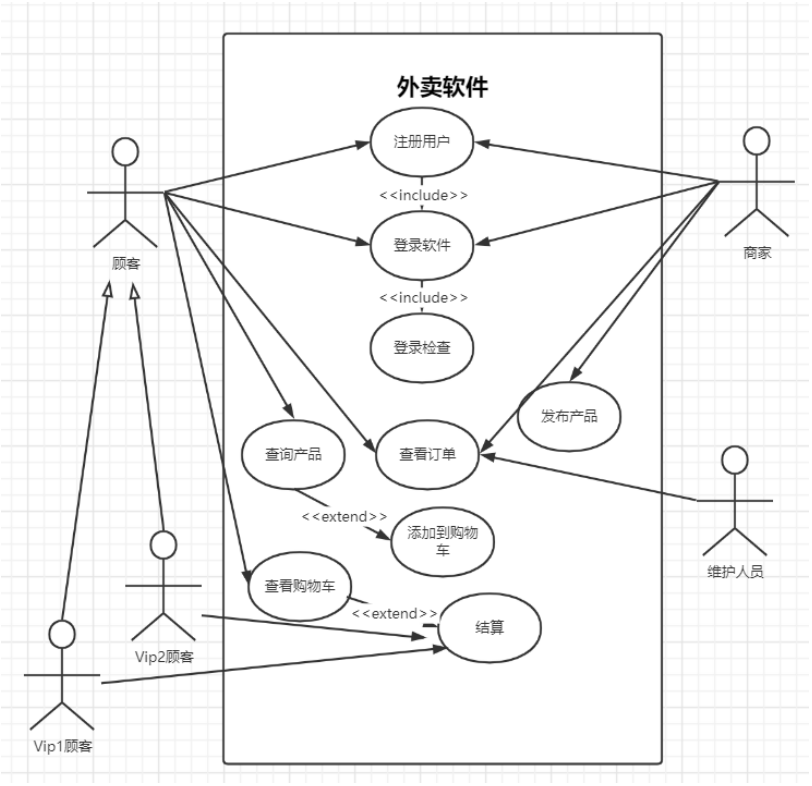
通过分析该软件的功能需求，得到了如图1所示的用例图，通过该用例图，可以直观的看到系统中出现的角色，以及系统需要提供的功能，接下来可以着手进行系统的设计。

如图2为该外卖软件的类图，为了简化起见，没有标注每个类的属性及其方法，只是给出了类名。

通过用例图分析，该系统显在的实体有顾客（包括 vip 顾客），商家以及维护人员，这部分实体使用 User 继承体系来建模，除此之外，由于该系统需要支持多个用户同时在线并且互相进行消息传递，因此将该软件设计为 C/S 架构，采用服务器 + 客户端的方式，在服务器端进行数据库相关的增删改查操作，在客户端负责和用户交互的相关逻辑。

*学号：171860526，邮箱：1143375493@qq.com

图 1: 系统用例图



服务器端的设计相对比较简单并且扩展性要求较低，需要针对应用设计好表结构，然后提供相应的增删改查操作即可，如图3为各个数据库表结构及他们之间的关系，这里采用的是 MySQL 数据库并且采用了 MyBatis 框架创建相应的 DAO(Database Access Object) 对象进行增删改查操作，在此处为了屏蔽服务器端的复杂性，采用了门面模式进行封装，增强了服务器软件的易用性。

客户端的设计相对比较复杂并且变化点较多，典型的变化点有用户类型的变动，客户端界面的变动等。此处在设计的时候考虑到尽可能多的变化点并且使用设计模式进行封装以提高软件的可复用性。

3 实现方案

下面是在设计该外卖软件中使用到的一些设计模式

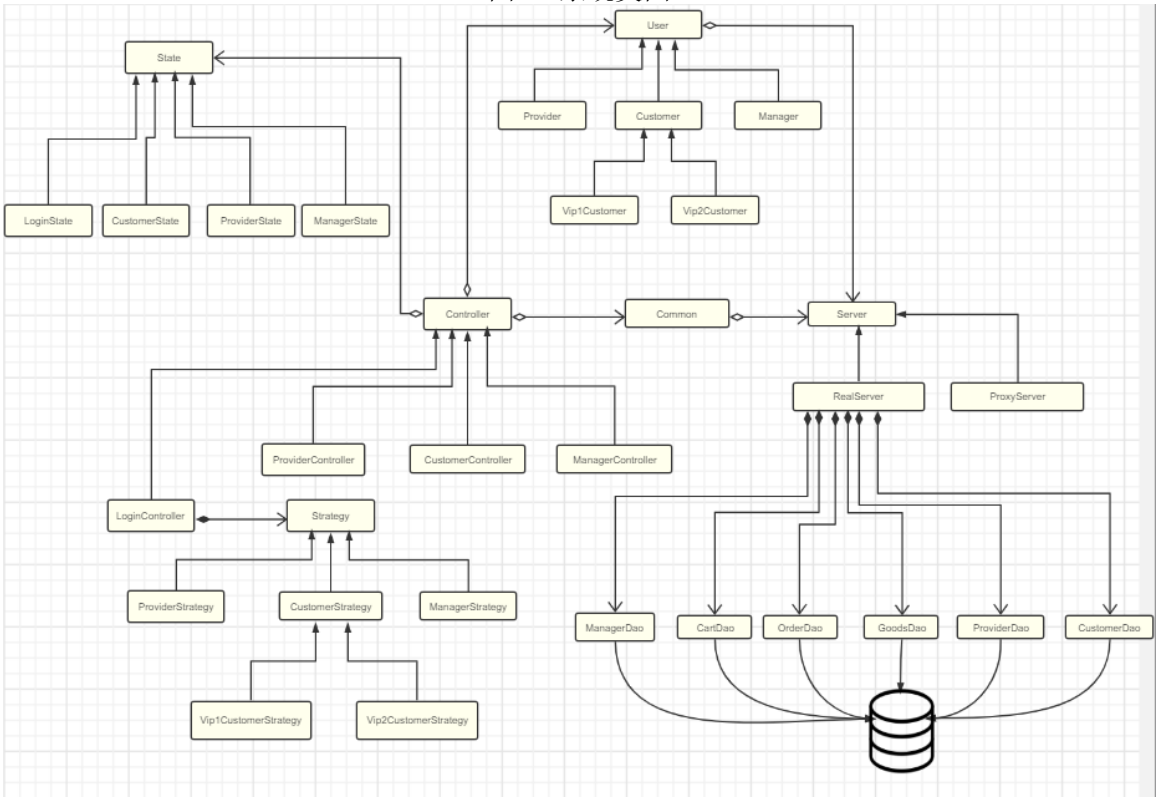
(1) 代理模式

在 C/S 架构中，客户端需要通过网络和远程的服务器进行交互，直接访问会带来很多麻烦，因此采用了远程代理的方式，在代理类 ProxyServer 中向服务器发送请求并且处理服务器的回复；

(2) 适配器模式

在控制器类中，需要负责一部分和服务器的交互，如果直接使用 Server 对象，则会使得控制器被迫依赖于其不需要使用的方法，违反了单一职责原则，通过引入适配器类 Common

图 2: 系统类图



适配 Server 解决了这一问题；

(3) 状态模式

客户端通常需要在不同状态中进行切换，比如用户点击了登录要跳转到用户界面，点击了注销又要跳转到登录界面，并且在不同的状态下有不同的行为，因此可以将状态转换和状态下的一些公共行为封装在状态类 State 中；

(4) 单例模式

在状态类中，每个状态类在整个系统中只要存在一个即可，客户端经常需要在不同状态中切换，这会导致状态类对象频繁的创建和销毁，因此可以将状态类设计为单例类，使用静态内部类的方式实现，从而提高整个系统的性能；

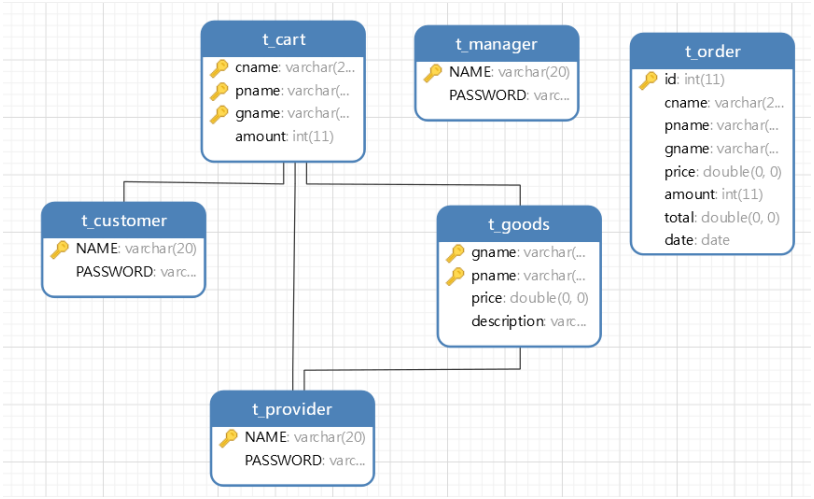
(5) 策略模式

在本系统中，存在多种类型的用户（顾客，商家，平台维护人员等），并且后续还可能增加新的用户类型，由于不同的用户存储在不同的数据表中，因此在登录的时候针对不同类型的用户需要查询不同的表，为了避免多重条件判断，引入策略类 Strategy 进行灵活配置；

(6) 简单工厂

策略类 Strategy 是针对登录界面时不同类型用户的登录方式设计的，而在登录界面用户可以灵活选择作为不同类型的用户登录，因此实例化策略类对象的时候需要根据用户输入的内容判断，此处采用了简单工厂，将判断的工作封装在工厂类中，同时采用工厂和抽象基类合并的方式，减少了系统中的类数量；

图 3: 数据库表结构



(7) 门面模式

服务器端主要是针对数据表的增删改查操作，但是由于数据表比较多并且后续还可能引入新的数据表，这会使得客户端和服务器的交互难度进一步增加，因此引入一个门面类 Server，将服务器的所有工作封装在这一个类中，降低了客户端和服务器之间的依赖性。

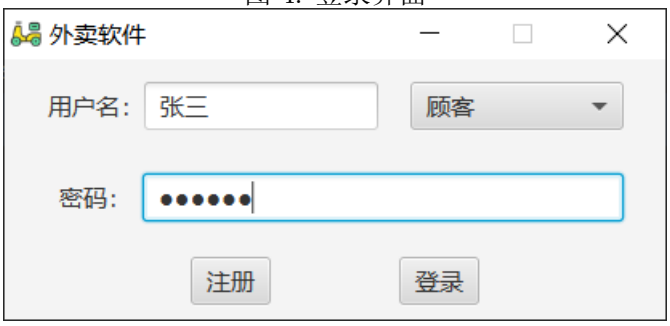
4 功能介绍和扩展性分析

4.1 功能介绍

在启动客户端程序之前需要首先启动服务器程序，因为客户端需要主动和服务器通过网络进行交互，启动顺序为 server.jar 然后是 client.jar。

4.1.1 登录界面

图 4: 登录界面



如图4为客户端软件的登录界面，在组合框中可以选择登录身份 (有顾客，vip1-顾客，vip2-顾客，商家，管理员可选)，数据用户名和密码之后点击注册创建新用户，或者点击登录进入用户界面。

4.1.2 顾客界面

图 5: 顾客界面

如图5是顾客登录之后的状态，在商品浏览页面可以查看不同商家上架的产品，勾选之后可以添加到购物车，由于商家可能在线添加新的产品，点击刷新可以查看到商家的最新状态。

点击购物车可以查看该用户当前购物车中的内容，每个商品条目中的商品数量可以通过双击的方式修改，勾选之后可以进行删除或者结算，结算之后的项目会从购物车中删除并且出现在历史订单中，同时订单会发送给相应的商家，在历史订单页面可以查看结算之后的订单信息，顾客看到的订单界面和商家看到的订单界面基本一致，因此在商家界面的地方一并说明。

4.1.3 商家界面

图 6: 商家界面

如图6为商家登录之后的状态，在商品界面商家可以在文本框中输入商品信息然后点击添加商品按钮来上架新的产品，在历史订单界面商家能够看到和自己店铺相关的订单信息，如果是顾客视角则可以看到和自己相关的订单信息。

图 7: 维护人员界面

4.1.4 维护人员界面

如图7，为维护人员登录之后的界面，在组合框中可以选择查看最近一周还是最近一月的信息，在表格中会显示所有用户在对应的时间段内的订单信息。

4.2 扩展性分析

对于现有类型的用户功能的扩展和修改相对比较简单，只需要继承自原来的用户类然后重写或者增加新的方法即可，必要时还需要改变用户的界面以及控制器类相应的逻辑，不能体现设计模式带来的可重用性，因此下面通过介绍如何通过扩展的方式向系统中添加新的用户类型支持来展示软件对开闭原则的支持程度。

当需要增加一个新的类型的用户的时候，要改变的地方如下

- 数据库增加该用户类型对应的数据表来存储用户信息
- 服务器增加该数据表对应的 DAO，Server 接口需要添加对该数据表的支持
- 客户端添加该类型用户对应的界面文件，以及相应的控制器类，状态类，和登录中的策略类

综上所述，添加一个新的类型的用户，需要修改的只要 Server 接口以及 Strategy 静态工厂中的创建方法，其他地方都是通过增加新的类的方式解决，几乎是在最大程度上符合了开闭原则，对扩展性的支持十分友好。

5 小结

在设计这个外卖软件的过程中，虽然最终版本的代码结构看起来很简单，但是这个版本的代码和最初的构想已经发生了很大的不同，很多构想在实践过程中才发现不合理之处，在《重

构与模式》一书中说到，模式不是设计出来的，而是重构出来的。除此之外，在完成基础功能再实现扩展功能的过程中，才会发现采用设计模式使得扩展不需要推倒重来，几乎就是水到渠成的工作，这也正是设计模式的强大之处。