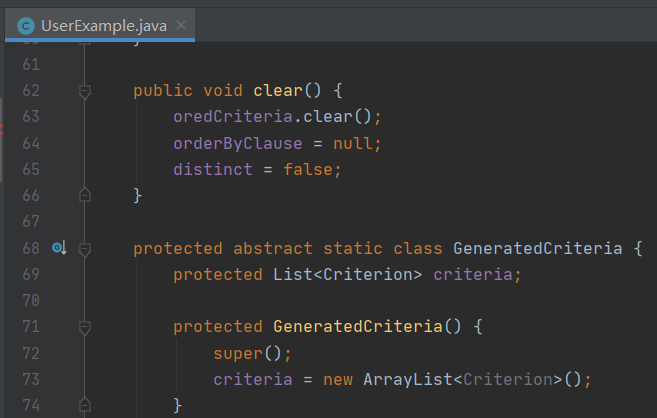
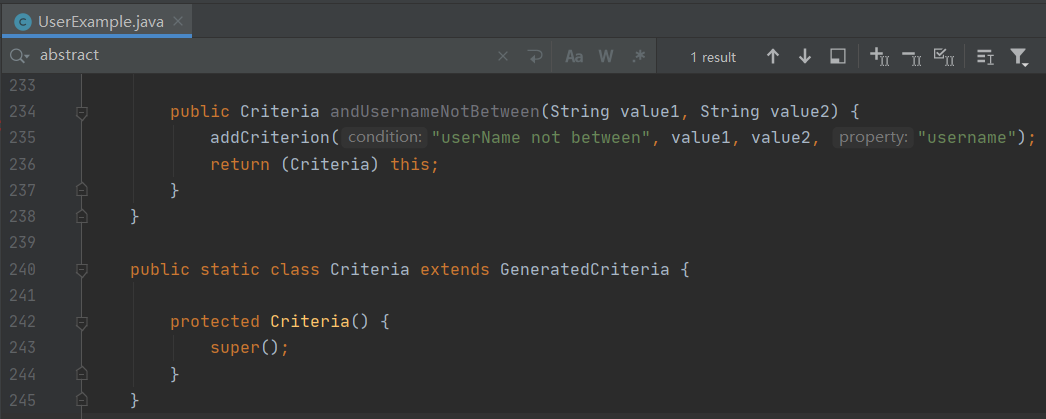
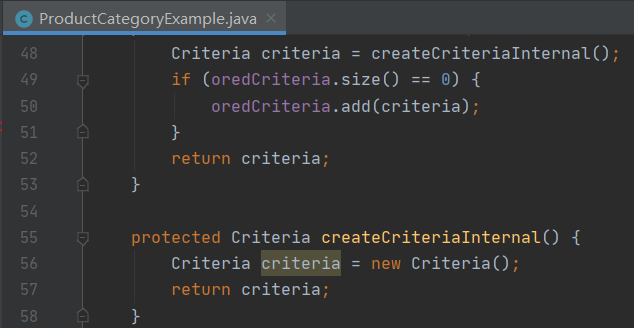
设计模式实验报告

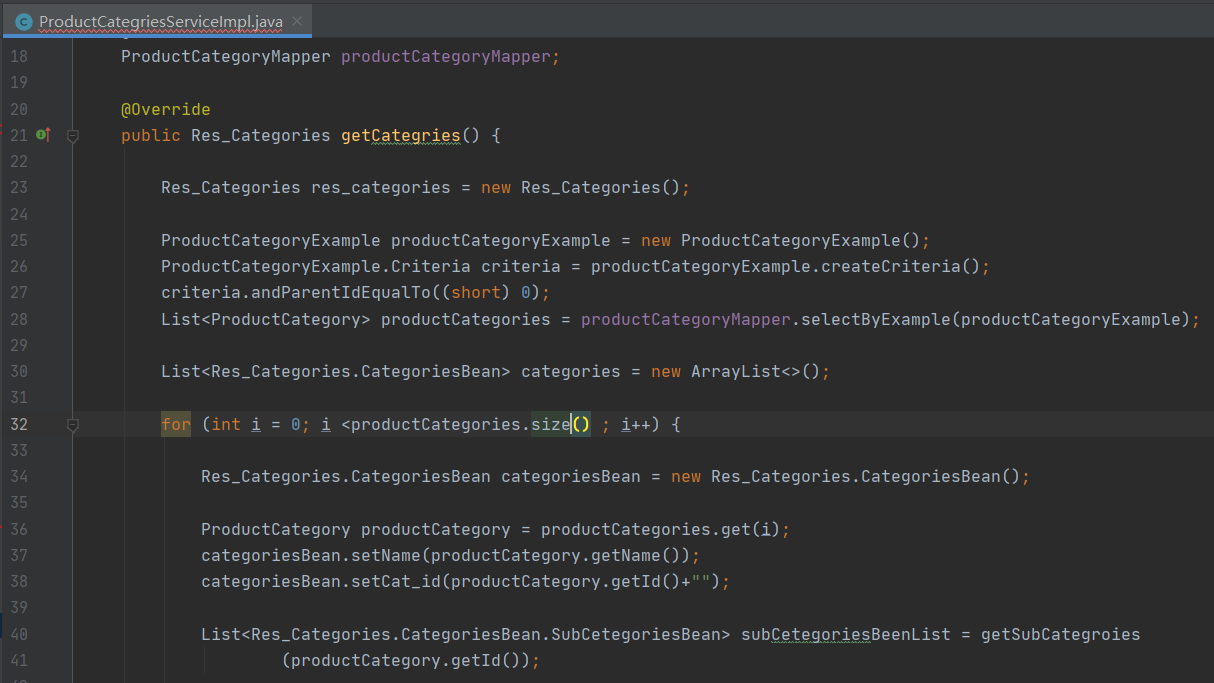
小组成员：刘自强 曾围江 吴畅

1. 抽象工厂模式。



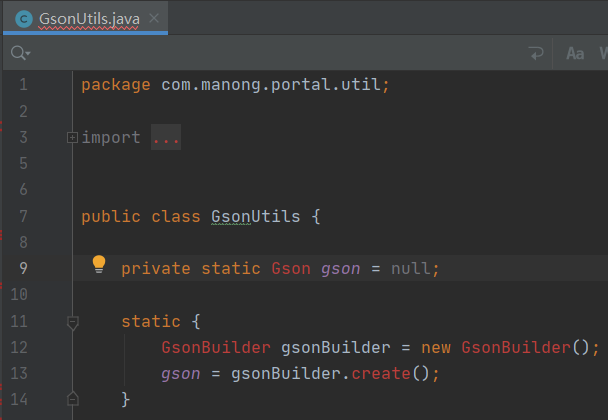






抽象工厂模式隔离了具体类的生成，使得客户端并不需要知道什么被创建；当一个产品族中的多个对象被设计成一起工作时，它能够保证客户端始终只使用同一个产品族中的对象；增加新的产品族很方便，无须修改已有系统，符合开闭原则。

1. 单例模式。



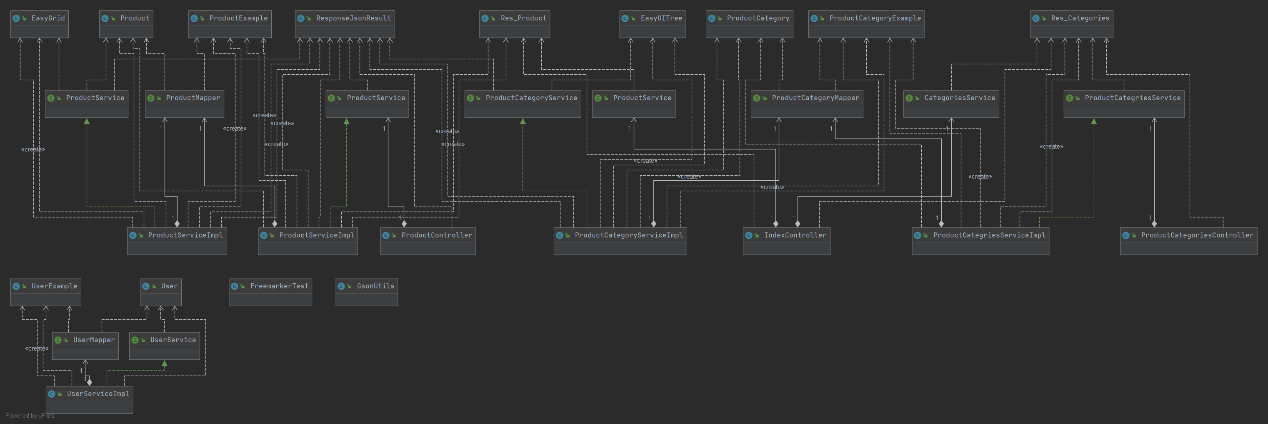
确保一个类只有一个实例，并提供一个全局访问点来访问这个唯一实例。

涉及到一个单一的类，该类负责创建自己的对象，同时确保只有单个对象被创建。这个类提供了一种访问其唯一的对象的方式，可以直接访问，不需要实例化该类的对象。

在内存里只有一个实例，减少了内存的开销，尤其是频繁的创建和销毁实例

避免对资源的多重占用。

1. MVC模式。



本项目采用SSM架构进行开发，使用了spring进行业务层开发，mybatis进行数据层封装，SpringMVC进行表现层开发。

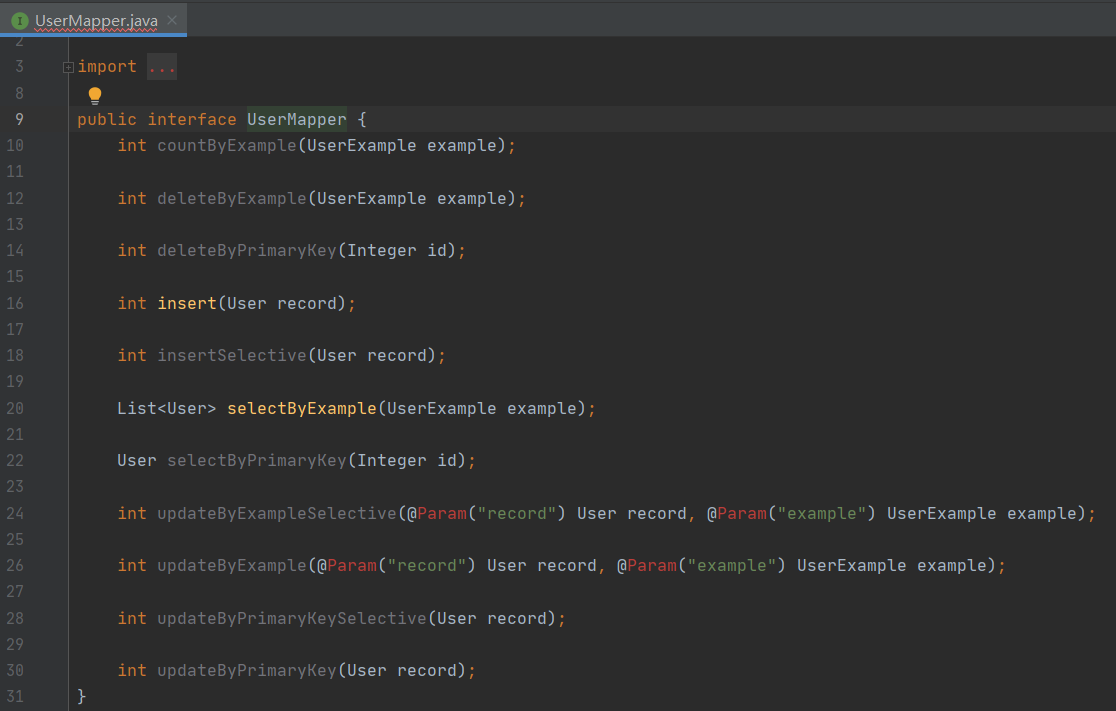
MVC 模式代表 Model-View-Controller（模型-视图-控制器） 模式。这种模式用于应用程序的分层开发。

Model（模型） - 模型代表一个存取数据的对象或 JAVA POJO。它也可以带有逻辑，在数据变化时更新控制器。

View（视图） - 视图代表模型包含的数据的可视化。

Controller（控制器） - 控制器作用于模型和视图上。它控制数据流向模型对象，并在数据变化时更新视图。它使视图与模型分离开。

1. 外观模式。



一个子系统的外部与其内部的通信通过一个统一的外观类进行，外观类将客户类与子系统的内部复杂性分隔开，使得客户类只需要与外观角色打交道，而不需要与子系统内部的很多对象打交道。它对客户端屏蔽了子系统组件，减少了客户端所需处理的对象数目，并使得子系统使用起来更加容易它实现了子系统与客户端之间的松耦合关系，这使得子系统的变化不会影响到调用它的客户端，只需要调整外观类即可一个子系统的修改对其他子系统没有任何影响，而且子系统的内部变化也不会影响到外观对象。