开闭原则（Open-Closed Principle, OCP）

定义：软件实体（类、模块、函数等）应当对扩展开放，对修改封闭。即在不修改现有代码的前提下，通过扩展（如继承、接口、组合）来增加新功能。

核心思想：

对扩展开放：新需求应通过新增代码（如子类、实现类）实现，而非修改旧代码。

对修改封闭：已有代码应保持稳定，避免因需求变化而被频繁修改。

实现方法：

抽象与多态：通过接口或抽象类定义稳定结构，具体功能由子类实现。例如，定义抽象类 Shape，子类 Circle 和 Square 分别实现 draw() 方法，新增图形类型时无需修改原有代码。

策略模式：将可变行为封装为独立类，通过组合动态替换。

与本项目遵循的开闭原则：

DatabaseManager类：提供了统一的数据库操作接口，新增功能可以通过添加新方法而不修改现有代码。

UI窗口类：如UsrWindow、Manager等，通过信号槽机制实现功能，可以扩展新功能而不修改现有代码。

迪米特法则（Law of Demeter, LoD）

定义：一个对象应仅与直接朋友交互（成员变量、方法参数、返回值等），避免与“陌生人”耦合。

核心思想：

降低耦合：减少类之间的直接依赖，提高模块独立性。

中介转发：通过中间类（如服务层）间接调用，而非直接访问陌生对象。

实现方法：

封装方法调用：将跨层操作封装在直接朋友类中。例如，Person 类不应直接调用 Car 的引擎方法，而是通过 Car 类封装 startEngine()。

本项目遵循的迪米特法则：

窗口类与DatabaseManager的交互：窗口类只与DatabaseManager交互，不直接操作底层数据库对象。

UI组件封装：如show\_user和show\_money封装了QSqlTableModel的使用细节。