（一）设计原则：

1.高内聚、低耦合

2.对扩展开放，对修改关闭

（二）初始设计：

Person()类中包含登录和修改密码方法，是否需要将其抽象成一个类设计？

（三）更新设计

登录类Register()

属性：账号、密码、身份、状态

登录方法：

选择身份登录；

输入账号和密码；

数据库验证，验证通过，设置状态为真，创建相对应的类parents、teacher、headteacher、manager，将账号和密码、身份等属性传到类；

验证失败，则设置状态为假，重新进入主函数。

Person类包含6个方法，是否是功能过于冗余？

主要的功能：查询、增加（修改），不存在删除数据的功能

addInfo可以单独写成一个类：

现在家长只可以写入联系方式，但是如果之后增加需求，家长可以写入更多的信息，那么就不仅只是一个方法可以解决。

addInfo()类：

但是这个类也无法满足所有的需求，如果单单是把方法写死，那么每次新增一个查询都会需要修改类，不满足对扩展开放，对修改关闭的原则。

需要把这些方法抽象成接口，由类实现接口，实现多态性。

面向抽象编程（参考设计模式和面向对象原则）

接下来的工作：

1. 先设计家长类、登陆类
2. 补充算法的具体流程
3. 学习设计模式、设计面向抽象的系统（学）

采用策略模式，把行为抽象成接口，

子系统只需要负责调用接口实现的类来完成操作，

接口实现的类可以由controller来动态的指定，

不需要修改类的方法，当新的功能或者新的需求需要修改的时候，只需要增加实现接口的类。