**71115112 余泽晨**

网上银行:

1. 抽象原理:抽象出每个用户的共有属性和共同行为,并以此描述软件需求的解.在完成软件的过程中,采用过程抽象和数据抽象,简化开发难度.

原因:抽象是封装的基础,便于处理系统的复杂性,有助于减少部件耦合和接口与实现的分离等.

1. 封装原理:将服务器,客户分别封装成类,每个类中包含数据和相应的行为,便于对对象的管理和信息隐藏,同时方便对接口的调用.

原因:封装保护事物内部信息不受破坏,使不同抽象之间有明确的界限,便于非功能特性的实现(如重用性:对不同页面的UI进行重用).

1. 信息隐藏:对用户隐藏部件的实现细节,用户不直接访问数据和进行具体操作,客户端只负责发送信息给服务器,由服务器完成相应操作后,将相应信息发送给客户端.

原因:更好地处理系统的复杂性,减少模块间的耦合有助于接口与实现的分离.

1. 模块化原理:将软件分成客户端,服务器,数据库三个模块,各个模块又分成更小的模块.如将客户端分成通讯模块和交互模块,交互模块又分成各个不同的页面等.

原因:增强了系统的资源管理,维护和应用的逻辑和条理性,使应用设计更加灵活,有利于系统的维护和升级减少变更设计造成的危害和付出的代价.

1. 耦合和内聚原则:客户端的通讯模块与服务器的通讯模块构成数据耦合,达到低耦合的要求.通讯模块只完成接受消息,传递消息,发送消息功能,达到了高聚合的要求.

原因:弱耦合部件的设计可以降低系统的复杂性,高内聚可以使同一模块负责的功能明确单一,便与开发和维护,使重用价值更高.

1. 接口和实现分离原则:由于一个用户可能有不同类型的账户,如活期,半年期,一年期等,通过对账户接口的设计,可以规范不同类型账户的使用方法,再通过不同类型账户类的实现完成接口与实现分离.

原因:防止部件的客户接触到实现的细节而造成意外的影响,便于部件在系统中的更改,避免用户直接受到部件变化的影响.

1. 分而治之原理:将问题横向分割,将一个部分的功能分割成更小的功能点进行实现.

原因:便于实现和维护.

1. 层次化原理:将系统分成M(模型),V(视图),C(控制器)三层.

原因:支持给予抽象层次递增的设计,不同层次的改变互不影响,易于实现和维护,支持功能增强和灵活的使用和重用.