面向对象程序设计与实践(C++)2

购物系统程序说明书

- 班级:2015211301
- 姓名:魏晓
- 学号:2015211301
- 班内序号: 18
- 日期:2017.12.23

- 第一部分:银行系统的设计
    - A.设计要求
    - B.类设计
    - C.实现方法
    - D.设计效果
- 第二部分:购物系统的设计
    - A.设计要求
    - B.实现思路
    - C.设计原则
    - D.设计效果
- 第三部分:Socket通信设计
    - A.设计要求
    - B.实现思路
    - C.代码重构
    - D.设计效果

设计背景

- 我们经常会在淘宝、京东等电商平台购买产品，这次编程的任务是做一个类似的电商交易平台。此外，为了实现交易功能，我们还要设计一个银行系统，用于产品的支付。
    - 注：请将所有的类中的数据置为私有，请将不需要被该类外部访问的函数设置为私有；如有良好的界面设计有额外加分

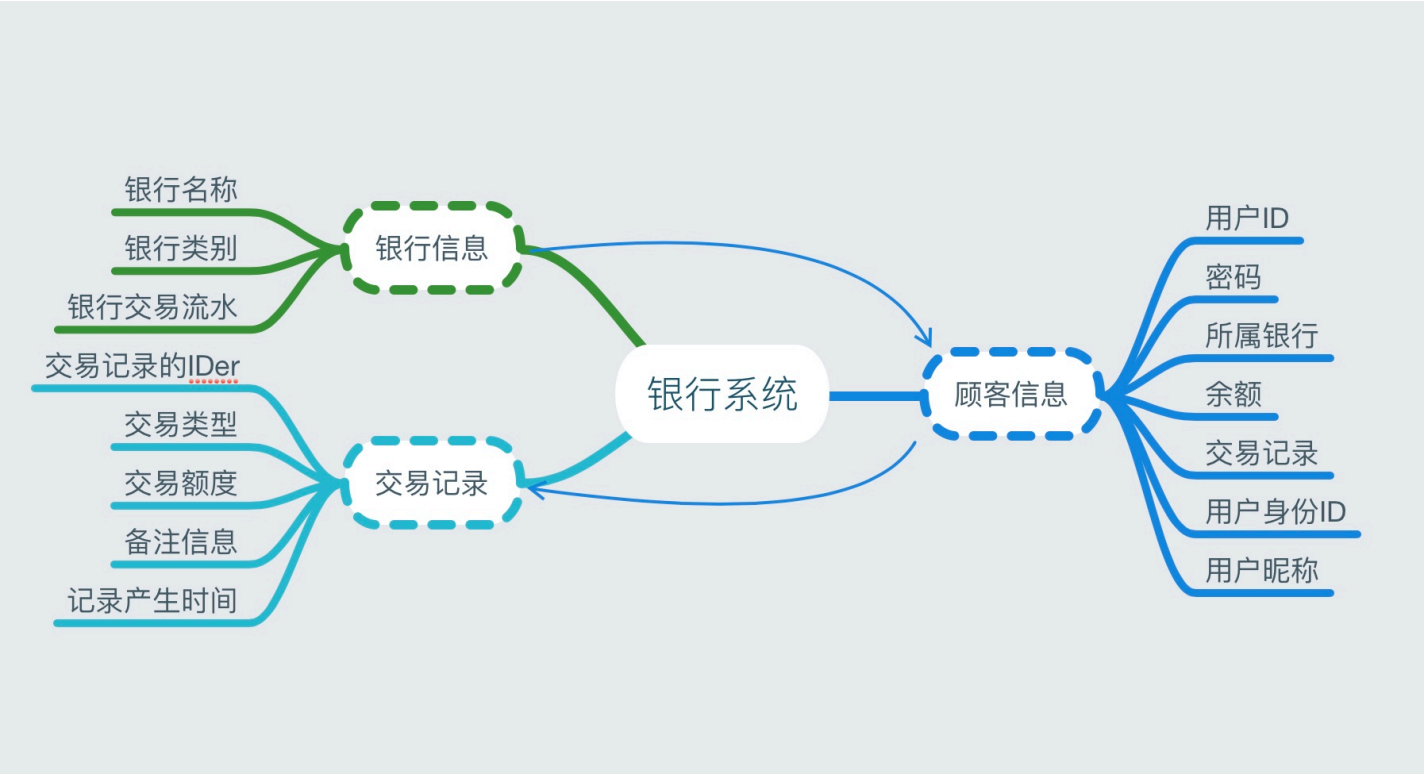
第一部分:银行系统的设计

设计要求

- 银行:该门户是一个单独的程序，程序要求至少支持以下功能：
    - 注册银行卡，银行可选；
    - 修改银行卡密码；
    - 存款取款；
    - 银行卡至少需要有卡号，密码，所属银行名称，持卡人身份证号，卡内金额等内容
    - 请把所有的银行名写入文件（或数据库），注册银行卡的时候，要求只能选择已经存在的银行进行注册。
    - 请做好错误场景的处理，例如读银行文件错误，输入数据不合法等等。

类设计

- 根据设计要求,银行至少要包含以下信息
    - 顾客信息
    - 交易记录
    - 银行信息
- 由此画出银行人员结构



实现方法

- 代码结构

```
typedef struct sbankuser{//用户结构
    string bankusername;//必须是符合id,且第一位不能是数字,等于10位
    string bankuserid;//8位数字
    string bankuseridcard;//十八位数字字符串
    string bankuserpassword;//密码,要实现隐形输入
    string banktype;//可选名称中的一种,注册后不可改变
    double userbalance;//余额
    }
}banker;

typedef struct Transactionrecord{//记录结构
    time_t bankrecarditime;//产生记录的时间
    string bankrecordid;//记录的id
    string bankrecordtype;//记录的 类型
    string bankaddition;//记录的备注信息
    double bankrecordnum;//记录的顺序
    }

}bankrecord;
```

```
class sBankinterface{//银行类
private:
    vector<string> bankname;//设置可以选择的银行名单

    map<string,banker> bankerlist;//id/人员记录

    vector<bankrecord> BTrecord;//银行记录

    string nowlocal;//当前登陆的顾客

public:

    sBankinterface();
    //构造函数
    ~sBankinterface();
    //析构函数
    bool banklogin(string a,string b);
    //登陆,a和b分别是用户名和密码
    void bankregister();
    //zhuceyonghu
    void banksignout();
    //退出登录
    void banktransfer(string distination,double b);
    //转账
    int  quicktransfer(string owner,string pass,string distination,double b);
    //服务器状态快速下转账
    void bankchangeinformation();
    //修改信息
    void getinformation();
    //得到信息
    void Moneydeal();
    //自己存钱取钱
    string getlocal();
    //得到档期按登陆
    int sBankrun();
    //运行银行
    int jianting();
    //运行socket
};
```

- 在此基础上构造函数和析构函数的作用就是把运行期间内存中的信息毫无保留的保存,并在程序运行开始的时候进行还原
- 设置单独的run函数方便代码重构,在未来添加服务器端的时候就可以进行选择,并且能对特定的

客户进行服务
- 查看自己的信息修改信息,为了保障银行安全,标示id由系统自动生成
- 完善的try-catch机制可以在运行的过程中持续关注代码运行的质量,并且在发生错误的时候有完善的提示信息

设计效果

- 实现了所有设计要求功能
- 完善了银行的功能
- 增加了交易记录分析功能,用户可以查看自己的记录,管理员可以查看所有的记录
- 新功能:
- 记录/存取款/额外属性

第二部分:电商平台的设计

设计要求

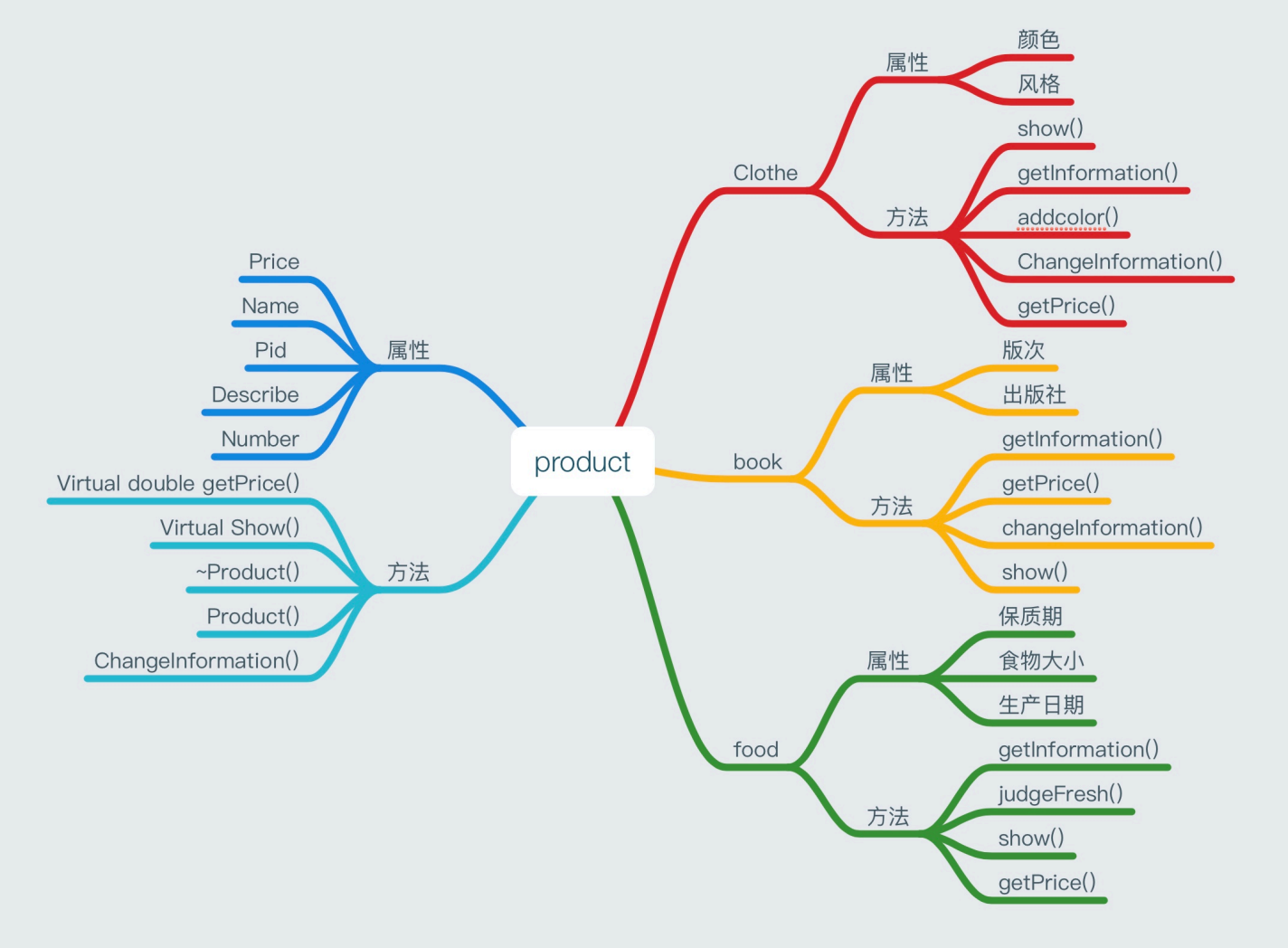
- 注册&登录：支持新用户注册平台账号，已注册用户用平台账号登录平台。（要求已注册用户的信息长久保留。）
    - 浏览平台产品信息。
    - 优惠活动：支持对同一品类下所有产品打折的活动，支持单笔订单满X减Y的活动。
    - 购买产品：支持用户添加产品到购物车，查看实际应付的产品价格，提交订单。
    - 在题目二我们暂时不考虑提交订单后支付等后续需求。
        - 电商平台上至少有三类产品：如食物、服装、图书等，每类产品中至少有三个具体的产品（如图书中可以有《C++ Primer》、《Effertive C++》等），每个具体的产品请至少包含产品描述，产品原价，产品剩余量等数据。所有的产品信息需要存储在数据库或文件中，不能写在代码中，平台管理员通过直接修改数据库或文件，管理本平台上的产品，包括产品的增加和删除，修改数量以及具体产品的属性信息等。
        - 请至少设计一层继承体系（产品基类-产品子类），设计一个产品基类，然后让图书类、电子产品类和服装类等产品子类类继承它，具体的产品是产品子类的实例对象（<> 是图书类的实例对象） 。产品基类请至少具有一个虚函数getPrice()用于计算具体产品的价格。
        - 请通过为每个产品子类定义“品类折扣系数”来支持对同产品子类下所有产品打折扣的活动（如图书全场5折，则图书类这一产品子类的折扣系数为0.5）。

类设计

- 电商的基础就是商品,所以商品类的设计是这个题目的基石,商品主要包含三个种类,食物/书籍/衣服,他们之间共同的元素是
  - 商品名称
  - 商品价格
  - 商品ID
  - 商品描述
  - 商品数量
  - 有如下方法
    - 获得每一个属性的值和改变每一个属性的值的方法
    - 虚函数getPrice()
    - 构造函数和析构函数
- 食物产品设计:
  - 继承商品父类
  - 有如下特殊属性
    - 生产日期
    - 保质期
    - 食物大小
  - 有如下特殊方法
    - 获得每一个属性的值和改变每一个属性的值的方法
    - 判断食物是不是超过保质期
    - show()函数,展示食物信息
- 衣服产品设计:
  - 继承商品父类
  - 有如下特殊属性
    - 衣服风格
    - 衣服颜色
  - 有如下特殊方法
    - 获得每一个属性的值和改变每一个属性的值的方法
    - 添加和显示颜色
    - show()函数,展示衣服信息
- 书籍产品设计:
  - 继承商品父类
  - 有如下特殊属性
    - 出版社

        - 版本号
    - 有如下特殊方法
        - 获得每一个属性的值和改变每一个属性的值的方法
        - show()函数,展示书籍信息

- 由以上设计构造产品类:



- 代码结构

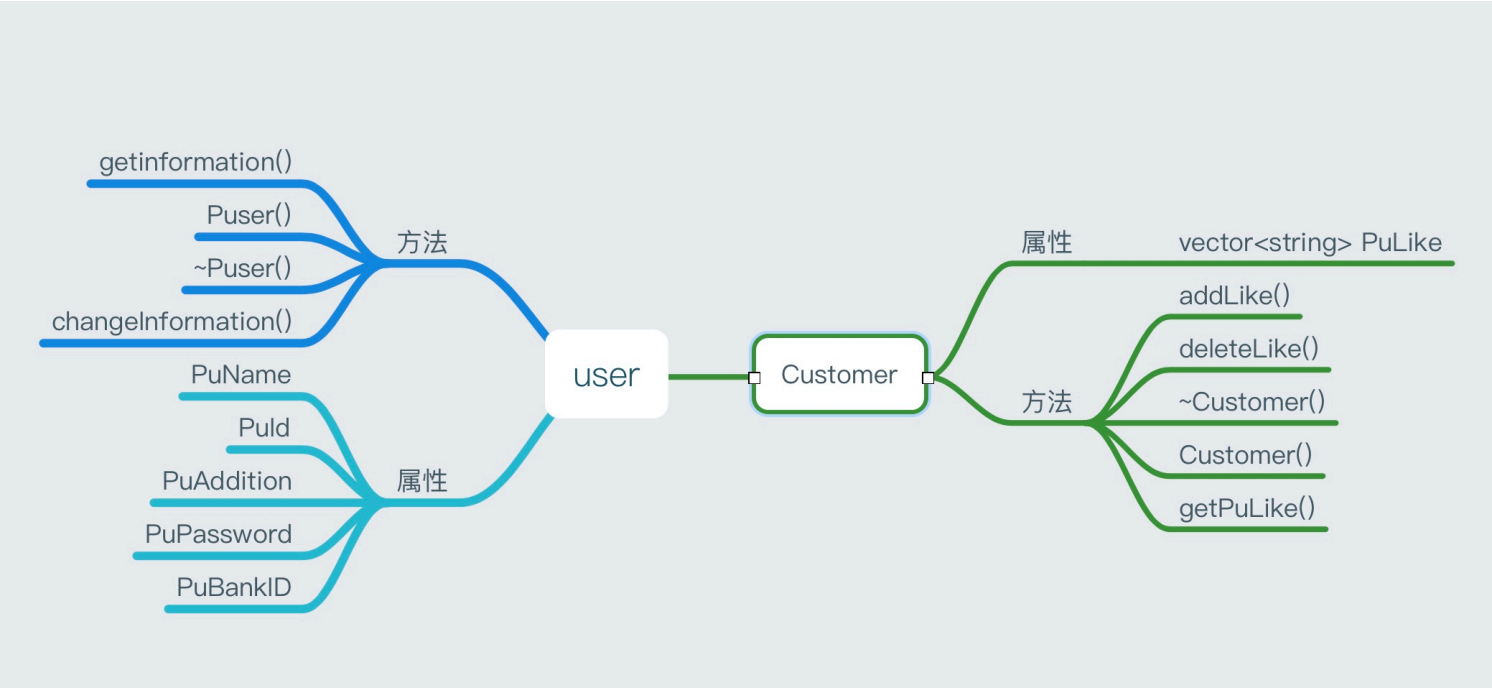
```
class Pproduct
{
    public:
        Pproduct();
        Pproduct(double p,string n,int nu,string d,string id);
        ~Pproduct();

        void changePrice(double p);
        virtual double getPrice();
```

```
        virtual void show()=0;
        double gp();
        void changename(string n);
        string getname();

        void changenumber(int n);
        int getnumber();

        void changeid(string i);
        string getid();
        void changedescribe(string d);
        string getdescribe();

    private:
        double Price;
        string name;
        int number;
        string describe;
        string pid;
};

class Pfood:public Pproduct
{
private:
    int producetime;
    int term;
    double Foodsize;
public:

    Pfood();
    Pfood(int p,int t,double fs,double pp,string n,int nu,string d,string id);
    ~Pfood();
    void show();
    void fchangeproducetime(int pr);
    int getproducetime();
    void changefoodsize(double fs);
    double getfoodsize();
    virtual double getPrice();
    void fchangeterm(int te);
    int getterm();
    bool judgefresh();
};

class Pclother:public Pproduct
{
```

```
private:
    string color;
    string style;
public:
    Pclother(string c,string st,double p,string n,int nu,string d,string id);
    ~Pclother();
    string getstyle();
    string getcolor();
    void show();
    void changestyle(string s);
    void addcolor(string s);
};

class Pbook:public Pproduct
{
private:
    int edition;
    string press;
public:
    Pbook(int e,string p,double pp,string n,int nu,string d,string id);
    ~Pbook();
    void show();
    void changeedition(int n);
    void changepress(string p);
    string getpress();
    int getedition();
};
```

- 这是基础的物品的结构,在设计结构的时候遵循的原则就是尽量的数据隐藏,在具体实现的过程中,除了设置基础的增删改查外还额外增加了一些特殊的固有方法,比如food属性的方法
    - void Pfood::judgeFresh()

员工类



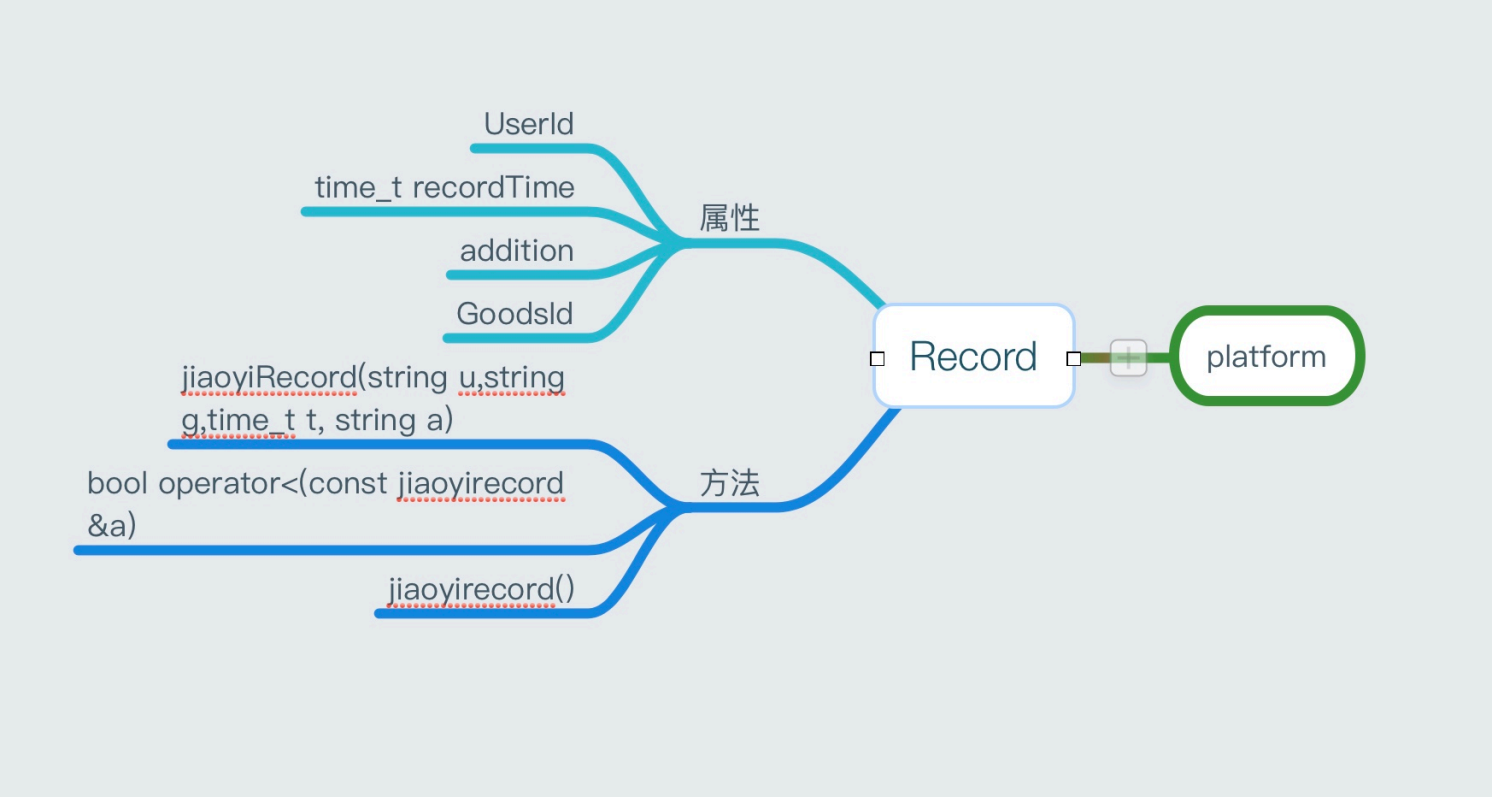
- 代码结构

```
    class Puser
{
    public:
        Puser();
        Puser(string na,string id,string ba,string ad,string pa);
        virtual ~Puser();
        void changename(string newname);
        void changeid(string newid);
        void changebankid(string newbankid);
        void changeaddition(string newaddition);
        void changepassword(string newpassword);
        string getPuname();
        string getPuid();
        string getPubankid();
        string getPuaddition();
        string getpassword();
    private:
        string Puname;
        string Puid;
        string PuBankid;
        string Puaddition;
        string Pupassword;
};
class customer:public Puser
{
```

```
private:
    vector <string> Pulike;
public:
    customer(vector <string> &li,string na,string id,string ba,string ad,string
 pa);
    ~customer();
    void addlike(string productid);
    void deletelike(string productid);
    void getPulike(vector <string> &ans);
};
```

- 记录检索
- 名称模糊搜索
- 员工子类
- 管理员
-

记录结构

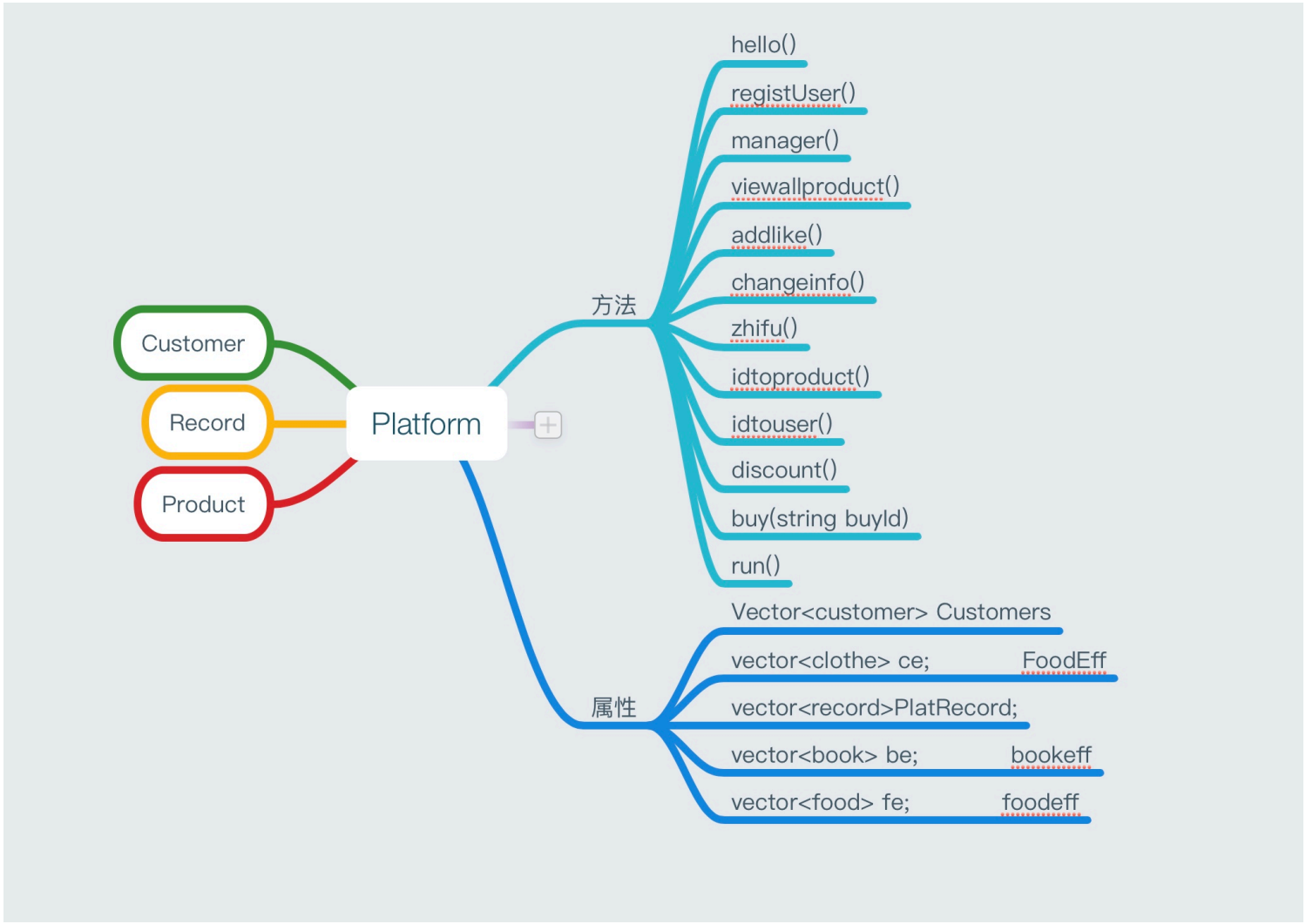


- 代码结构

```
typedef struct jiaoyirecord{
```

```
    string Userid;
    string goodsid;
    time_t recordtime;
    string addi;//∏Ωº”Àµ√˜£¨±»»Áπ∫¬Ú∑Ω Ω³π”–Δ‰À˚µƒ
    jiaoyirecord(string u,string g,string a)
    {
        Userid=u;
        goodsid=g;
        recordtime=time(NULL);
        addi=a;
    }
    jiaoyirecord(string u,string g,time_t t,string a)
    {
        Userid=u;
        goodsid=g;
        recordtime=t;
        addi=a;
    }
    bool operator <(const jiaoyirecord &a)
    {
        if(a.recordtime<recordtime)
            return false;
        else
            return true;
    }
}record;
```

平台类



- 代码结构

```
class Pplatform
{
    public:
        Pplatform();
        ~Pplatform();
        void registuser();
        void hello();
        void run();
        void buy(string buyid);
        void manager();//ZHIYOU GUANLI YUAN YOU NENG LI
        void discount();
        void ViewAllProduct();
        void addlike();
        int idtouser(string id);
        int idtoproduct(string id);
        void changeinfo();
        bool zhifu(double localjine);
```

```
    private:
        vector<customer> customers;

        vector<Pfood> fe;// ≥Œ
        double Pfoodeff;//'€ø€œμ ˝
        vector<Pclother> ce;//"¬∑˛
        double Pclothereff;
        vector<Pbook> be;// È
        double Pbookeff;

        vector<record> platrecord;
        int local;
};
```

- 设计思路

设计原则

- 在写代码之前首先类设计,确定想要选用的数据结构
- 根据属于隐藏要求,设定私有属性,确定私有方法和公有方法
- 最后把数据类作为整个平台的一个元素
- 在设计数据元素的时候留下一个额外属性,方便以后代码重构
    - product.addition
- 使用map代替vector,使得数据结构在存储大量数据的时候复杂度降到log N

设计效果

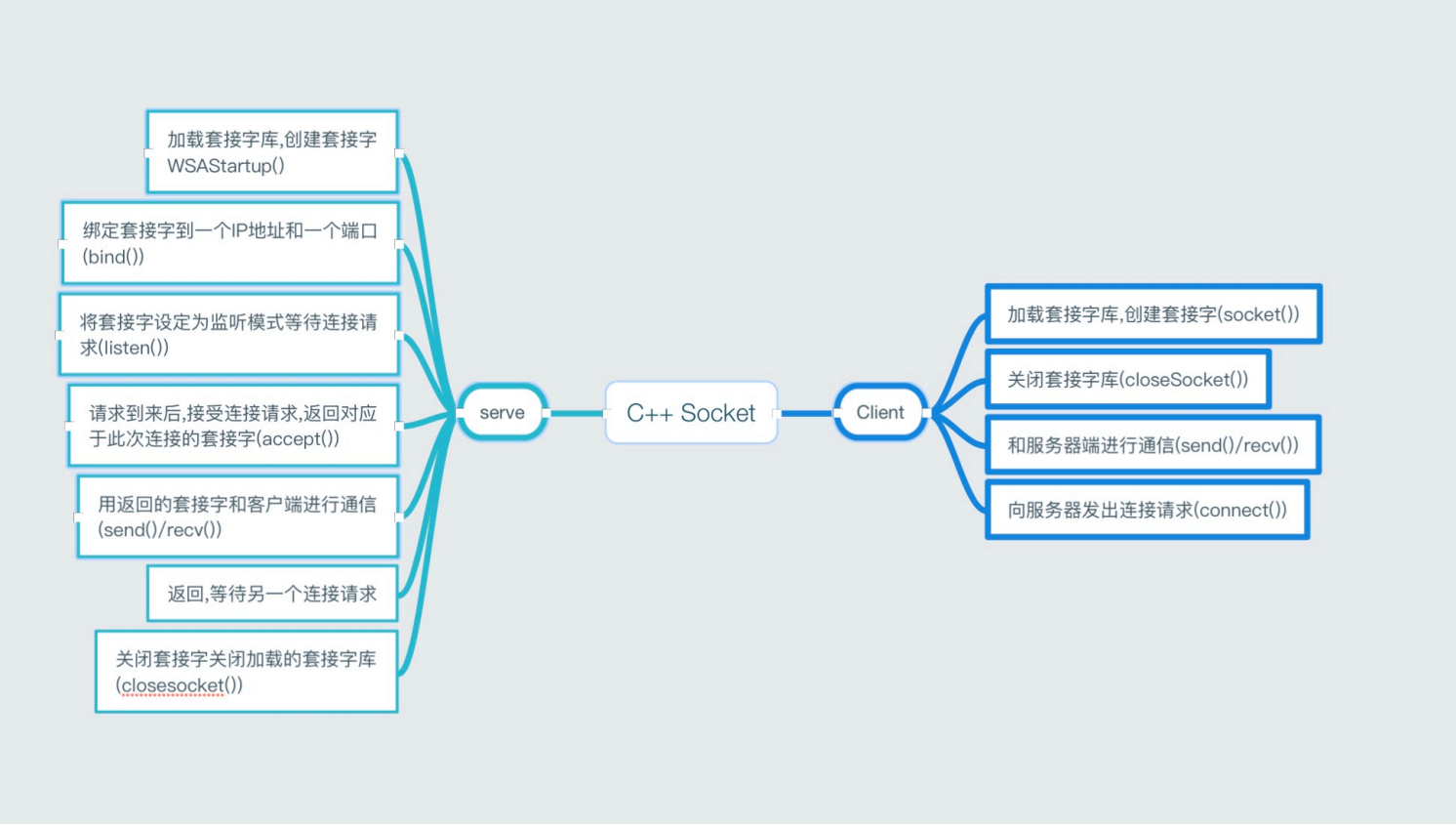
第三部分:网络支付的设计

设计要求

- 实现题目三会让你设计的平台像个真正的运行在网络上的平台。要求在题目一、二的基础上支持通过网上支付在电商平台上购物，请至少实现以下功能：
    - 绑定银行卡：支持电商平台账号绑定银行卡。同一账号可以绑定多张银行卡，而且银行卡可以属于不同银行。
    - 网上支付：选择绑定的任一银行卡支付电商平台上的订单。
        - 绑定银行卡和选择银行卡进行支付时都要求提供对应的银行卡密码。

- 当在电商平台上绑定银行卡或者进行网上支付的时候，请不要直接打开某个文件查找银行卡的信息，因为银行卡信息文件是银行门户系统私有的，电商平台系统无法直接访问，而应该由电商平台系统去向银行门户系统对接数据。请用socket通信来传送数据。
- 请做好错误场景的处理，如绑定银行卡失败，支付失败等。

实验思路



- 在程序代码中,为了实现服务器端对客户的监督,酸则使用select,这样就能检测当前是谁发送了请求,如果在这个过程中产生危险动作,就可以记录
- 对与确认信息,回答的信息是可以理解的,计算机通过判定返回字符串的种类来确定支付是不是成功
-

代码重构

- 代码重构的过程中, ### 设计效果
- 设计效果如下:
- 实现了所有的基础功能:

- 实现了这些附加功能 ## 实验感想
- 这次实验于我而言是特别曲折的,最开始在mac平台上写的代码,后来遇到无法解决的问题,又把代码迁移到windows,使得我的节奏一下子放缓了好多
- 在迁移的过程中使我意识到了代码通用型的重要性