e-Baazar

"Shopping Management System"

Object Oriented Programming Project

Session: 2015-2016(Sem III)

Batch: B13

Group Members:

Manasvi Duggal (14503003)

Shruti Sahu (14503011)

Aditya Mishra (14503027)

Ankit Goel (14503028)

ACKNOWLEDGEMENT

Beside every successful errand, there is not just one, but many people to thank…Realizing this from the bottom of my heart,we express our heartfelt gratitude to the one who has been instrumental in giving this project its shape. We specially thank our OOP teacher “Mr. Sandeep K. Singh” for her sincere co-operation and guidance to make this project a success in best possible way.

THANK YOU..!

PURPOSE OF THE PROJECT

We've descended upon an era where the needs and desires of the common man are endless. Whether his toothbrush broke or his favorite band released a new album, he requires an instant service to fulfill his requirements. The answer? E-commerce.
e-Baazar, A shopping management system (offline mode) is a part of e-commerce. E-commerce or business through net means distributing, buying, selling, marketing, and servicing of products or services over electronic systems such as the Internet and other computer networks. In this modern world most of us prefer to order products online be it traditional wear or electronic items…. So this project provides one the chance to exhibit the same!
The main aim of this project is to improve the services of Customers and vendors. It allows the customers to share a connection with, and view various vendors to shop from within the system. It maintains the details of customer payments, product receipts, addition of new customers, products and also updating, deletion for the same and many more features. It also stores the details of invoices generated by customer and payments made by them with all Payments details like credit card.
And so does this project attempts to fulfill the above aspects by highlighting some of the major concepts of Object Oriented programming in C++.

ADVANTAGES:-

- ⇒ An edge over the competition at an affordable price.
- ⇒ Broader customer reach across regions.
- ⇒ Builds a customer database.
- ⇒ Provides a channel for marketing and promotion lowering your advertising cost.
- ⇒ Helps in improved service.
- ⇒ Greater customer satisfaction.

PROJECT DESCRIPTION

The primary features of this project is high accuracy, design flexibility and easy navigation. This software has been divided into many pages and subprograms to allow the user to understand things more easily.

Some of the features included are:-

- EASY TO UNDERSTAND LANGUAGE
- SMOOTH INTERFACE
- FEATURE OF MENUS
- FEATURE OF ORDERING ITEMS
- USER FRIENDLY
- FEATURE OF MAINTAINING YOUR PROFILE
- EASY NAVIGATION THROUGH MENUS
- PAYMENTS THROUGH COD
- SHOW PRODUCTS ACCORDING TO PRICE, CATEGORY ETC.
- CART SYSTEM
- FEATURE OF PROMOCODES
- FEATURE OF COMBOS
- FEATURE OF FAQ

PROJECT REQUIREMENTS

Software Requirements:

⇒ **TEXT EDITOR:** Notepad++

⇒ **G++ COMPILER**

Hardware Requirements:

⇒ **PROCESSOR:** 450 MHz Pentium II-class processor or above

⇒ **OPERATING SYSTEM:** Microsoft Windows XP / Win7/Higher

⇒ **RAM:** 1GB or more

IDENTIFIED CLASSES

Guest - Showcases products to non-registered customers

Cart - Stores the list of products that the customer wishes to buy

Category - Categorises all the products into different categories

Product - Contains the details of a particular product

Customer - Contains the details of a particular customer

Order - Contains the details of all orders placed by a customer

Administrator - Updates catalog and manages the shopping system

FAQ - Contains Frequently Asked Questions to aid customer

Combo - Combines two products and offers discount

INCREMENTS

1. **Single Application, Single Class, Single Method**

A product is added by administrator. A product has a product id, name, category, cost etc. A product can only give its own complete information.

```
class product
{
          public:
          int productId;
          char prodName[20];
          char categ[20];
          int cost;
          int qtyAv;
          bool availability;
          char specs[50];

          void addProd() ;//SINGLE APPLICATION
};
```

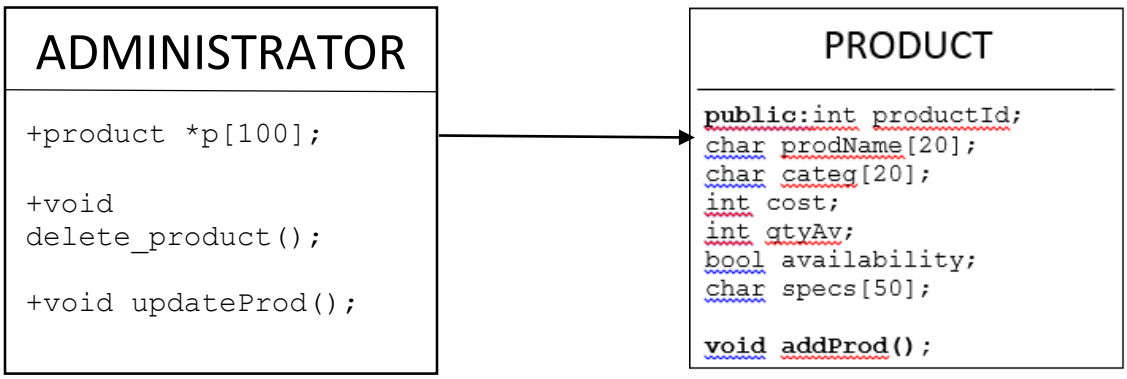
PRODUCT
public:int productId; char prodName[20]; char categ[20]; int cost; int qtyAv; bool availability; char specs[50]; **void addProd();**

2. **Single Application, Single Class, Multi Method**

A product's info can be set and edited by its administrator.

```
class administrator  //public product
{     friend class product;
      friend class customer;
      product *p[100];
      customer *c[100];
      int totProd;
      int totCus;

        string adminName;
        string email;
        //customer *c;
        public:
               friend void main_menu();
                      void admin_menu();
                      void delete_product();
                      void updateProd();
};
```

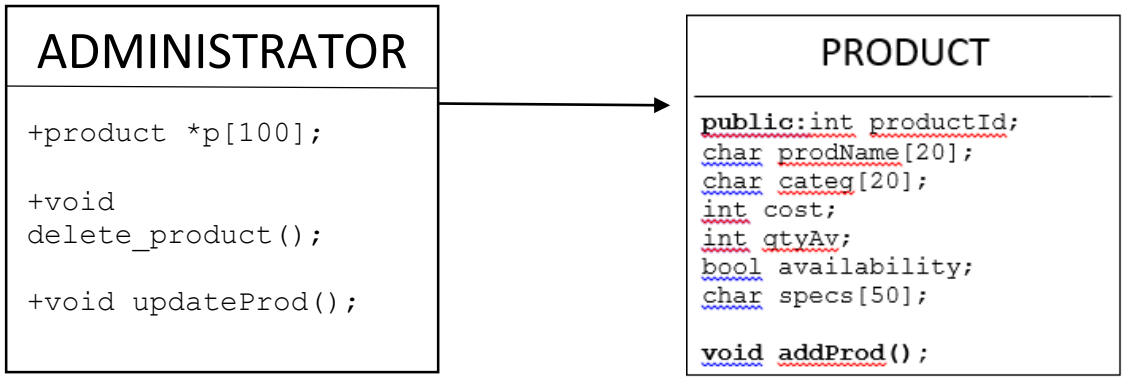
MULTI METHOD



3. Single Application, Multiple Class, Multi Method, Simple Relationship

A product can be ordered by many customers. An order is associated with a product and so is the customer associated with the order. Customer should be able to register on the system and place orders.

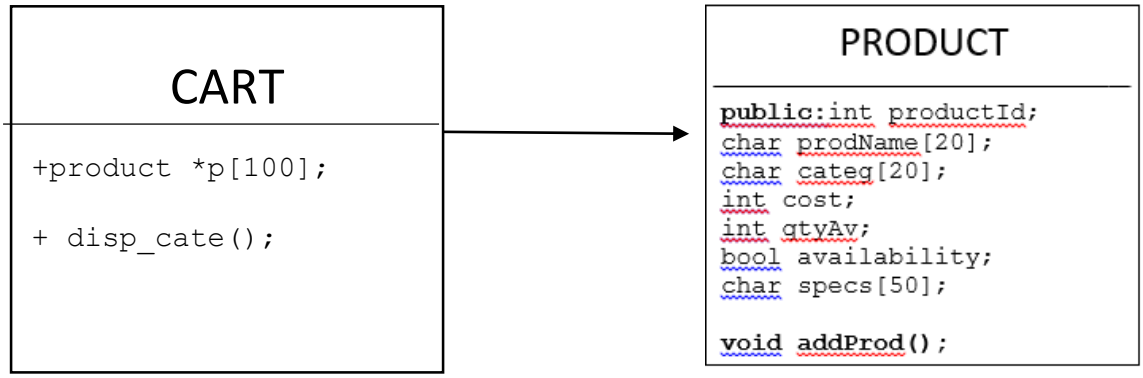
```
class order
{   static int totOrd;
     int orderId;
     product *p;       //SINGLE ASSOCIATION
     char c[5][20];
     char c1[5][20];
public: void disp_categ();
       int placeOrder();
};
```



4. Single Application, Multiple Class, Multi Method, Complex Association

Customers can order more than one products too (through cart). Cart is associated with many products.

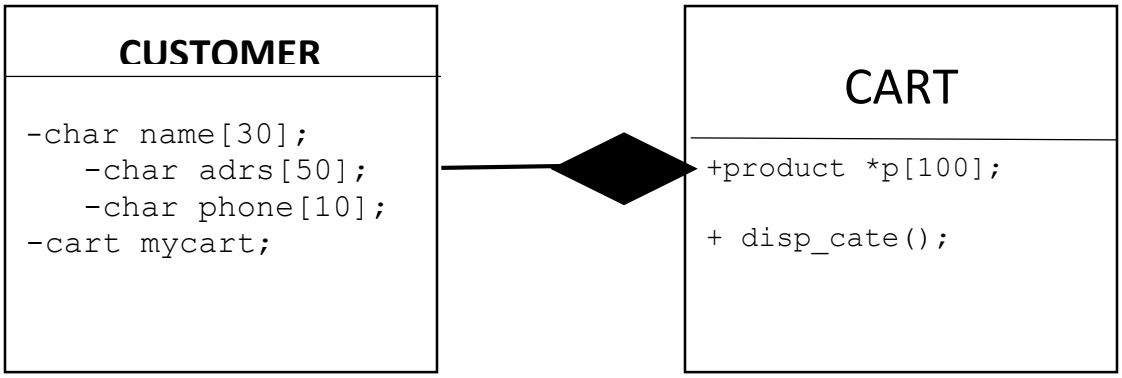
```
class cart
{
   static int totCart;
     int cartId;
     product *p[100]; // ONE TO MANY ASSOCIATION
     char c[5][20];
     char c1[5][20];

     public:
            cart()
              {

                  cartId=++totCart;
              }

              void disp_categ();
};
```



5. Single Application, Multiple Class, Multi Method, Composition

A customer has its own cart which consists of 1 or more products.

```
class customer:public guest
{
     friend class administrator;
   char name[30];
   char adrs[50];
   char phone[10];
   char email[20];
   char user_id[20];
   char passwd[20];
   int login_state;
   creditcard cc;
  cart mycart;//COMPOSITION
   order *ord;};
```



6. Single Application, Multiple Class, Multi Method, Inheritance

A customer inherits the properties of Guest who can only see the products list.

```
class customer:public guest //INHERITANCE
{
     friend class administrator;
   char name[30];
   char adrs[50];
   char phone[10];
   char email[20];
   char user_id[20];
   char passwd[20];
   int login_state;
   creditcard cc;
  cart mycart;
  order *ord;};
```

7. Single Application, Multiple Class, Multi Method, Polymorphism (Overriding&Overloading)

Guest has a virtual show function which is overridden in its base class customer.

```
class guest
{
      public: virtual void show_product();};
class customer:public guest
{
     friend class administrator;
   char name[30];
   char adrs[50];
   char phone[10];
   char email[20];
   char user_id[20];
   char passwd[20];
   int login_state;
   creditcard cc;
  cart mycart;
   order *ord;};
```

```
public:show_product();//OVERRIDING
};
combo operator+(product Px)
        {
            combo Cx;
            Cx.comboCost=cost+Px.cost;
            Cx.comboQtyAv=min(qtyAv, Px.qtyAv);
            strcpy(Cx.comboName, Px.prodName);
            strcat(Cx.comboName, " - ");
            strcat(Cx.comboName, prodName);
            //cout<<"\n operator overloading";
            return Cx;        }
```

8. Apply Templates in at any level of increment(s) above.

```
template<typename T>

            T d(T a)
            {
                return a;
            }
```

9. Apply Exception Handling

```
FILE *fp;
                    //EXCEPTION HANDLING
                     try{


                        fp=fopen("product.txt","a");
                        if(fp==NULL)
                       throw "Product.txt File cant be opened ";

                        }
                     catch(const char* msg)
                     {
                         cerr<<msg<<endl;
                         exit(0);
                     }
```

REFERENCE MATERIAL

- C++: The Complete Reference, 4th Edition by Herbert Schildt
- http://www.tutorialspoint.com/cplusplus/cpp_object_oriented.htm
- https://cppcodetips.wordpress.com/2013/12/23/uml-class-diagram-explained-with-c-samples/
- http://www.studytonight.com/cpp/cpp-and-oops-concepts.php