Karyana Management System Report

Course Project

OBJECT ORIENTED PROGRAMMING

Submitted by: Jawad Hassan

Roll No. : **2230-0035**

Submitted to:

Mr. Yasir Niazi

DATE:

28 April 2025

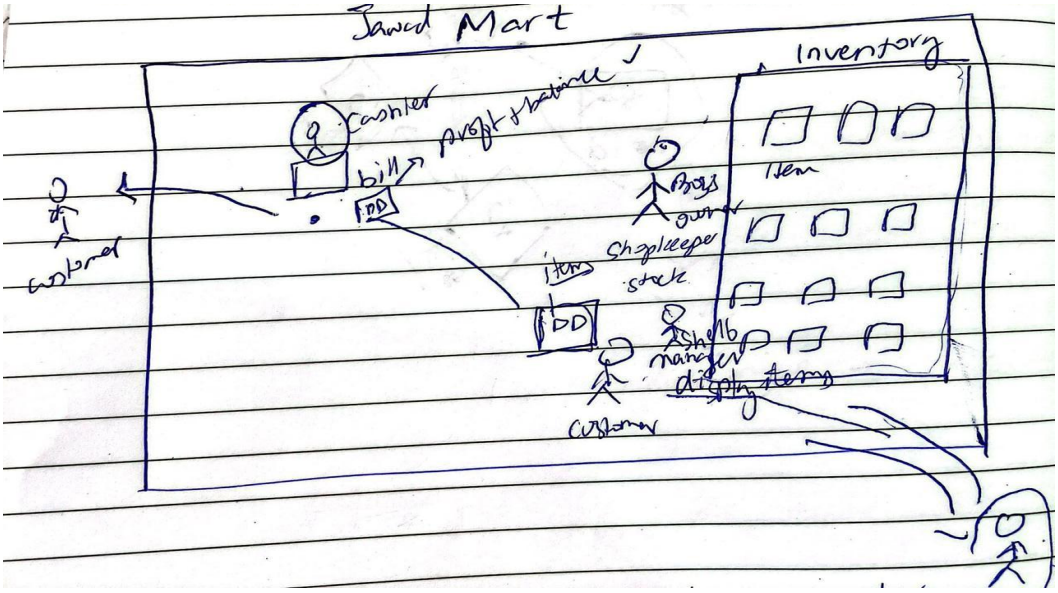
Main Idea of Project:

A karyana system starts with inventory being loaded by shop owner, when a customer decides to enter the store and shelfer shows and assist him by showing all items available in store, Customer selects items 1 by 1 and add them to cart, when he is done shopping he goes for checkout to the cashier, the cashier ask customer in line's info detail and generates bill. Customer pays and leaves Jawad Karyana Store with ease.

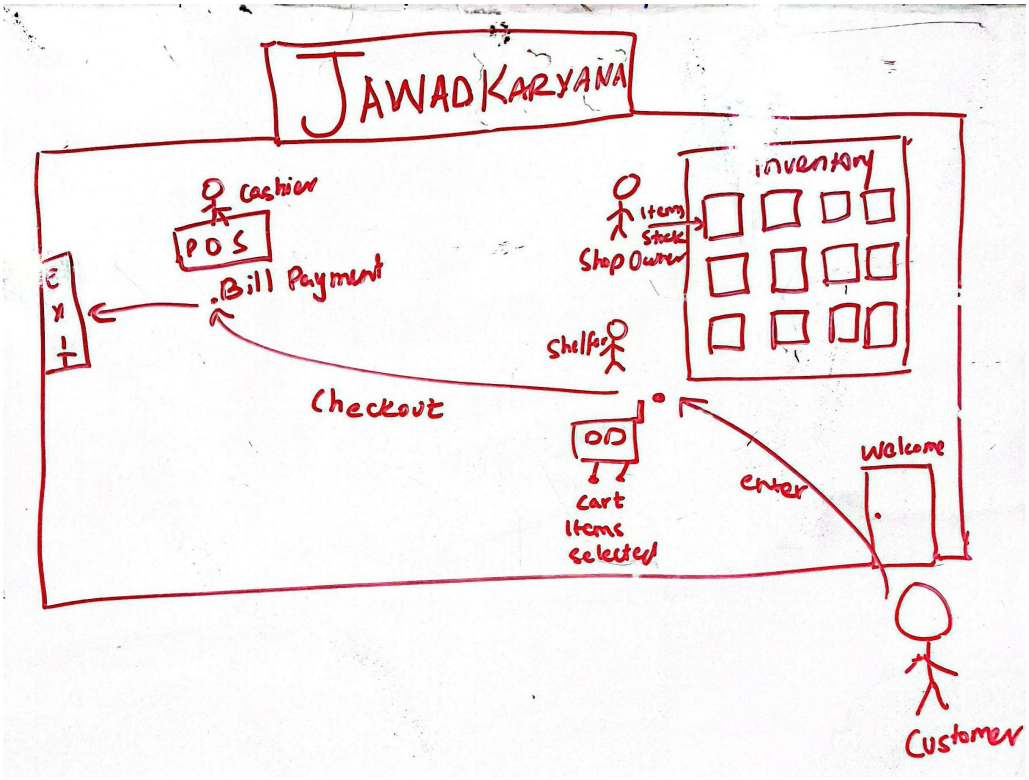
Purpose:

To make things easy and automated in small stores, it was a passion project of mine.

Idea started on a piece of paper:

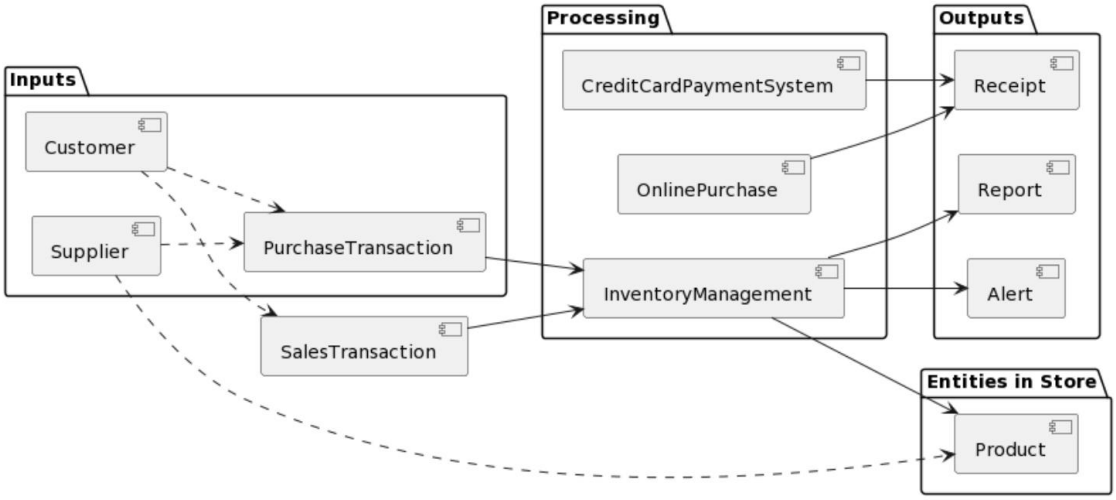


Idea moved to white board:

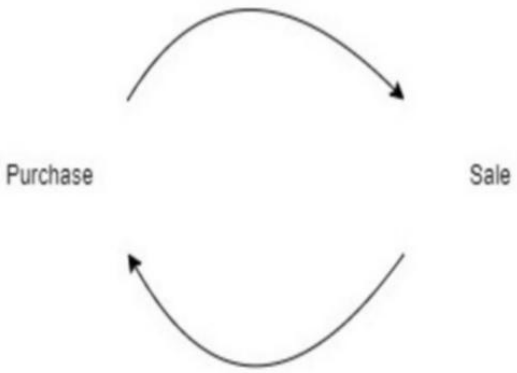


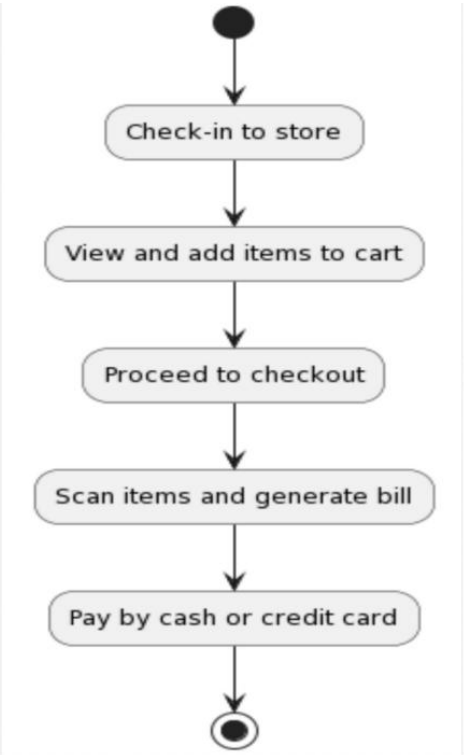
Diagrams:

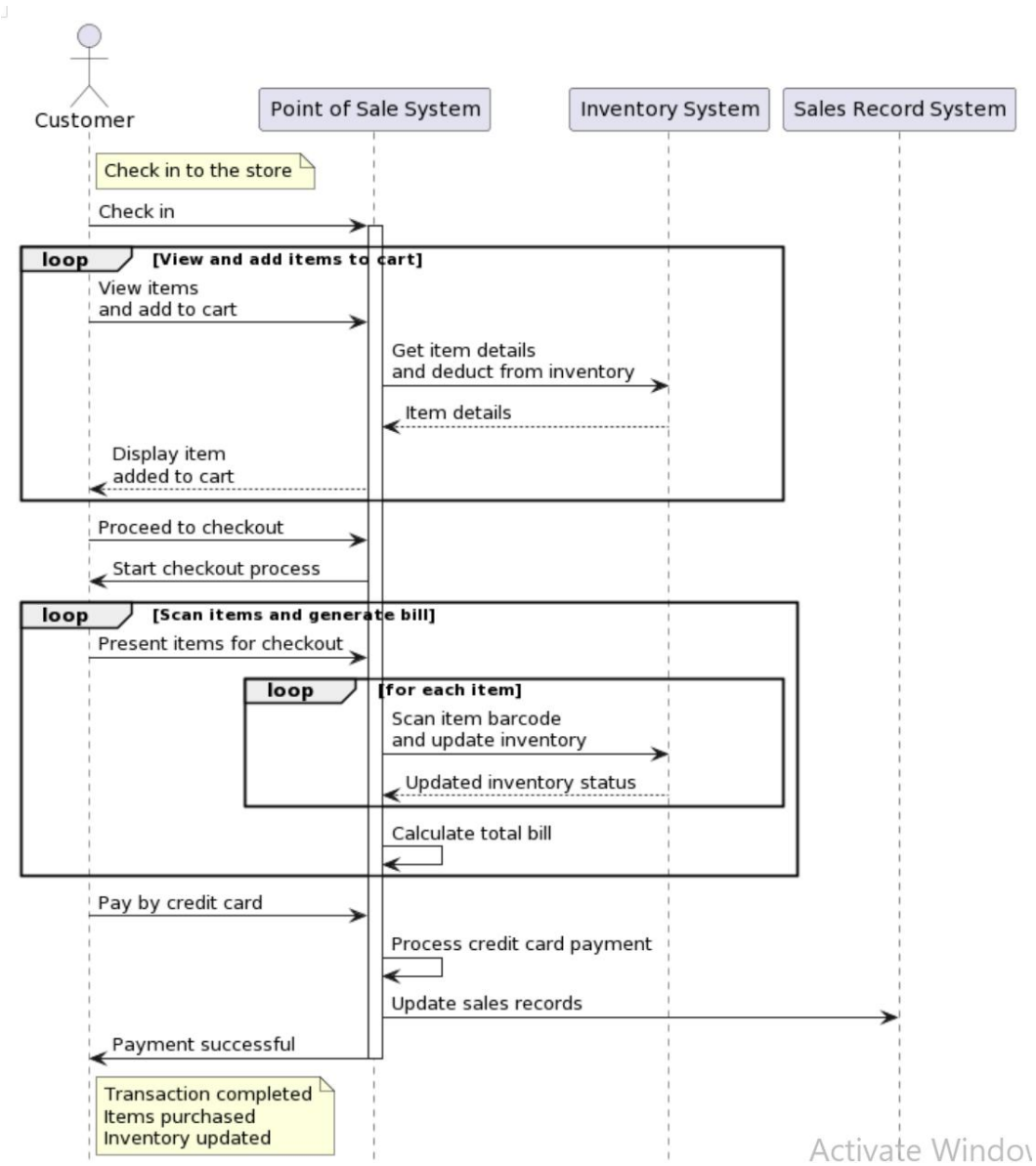
Flow of System:



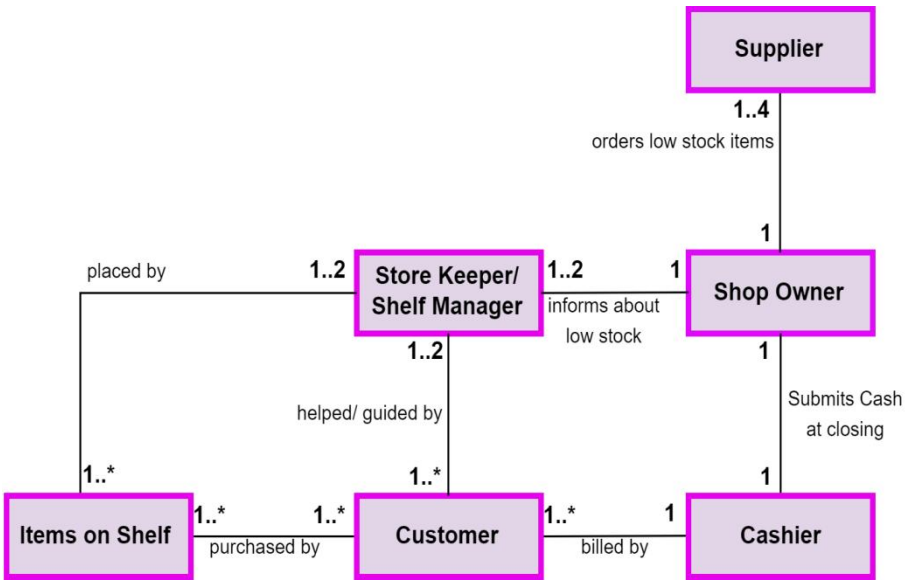
Purchase and sale in a karyana store are in a continuous loop:







Classes and Association in Karyana Management system:



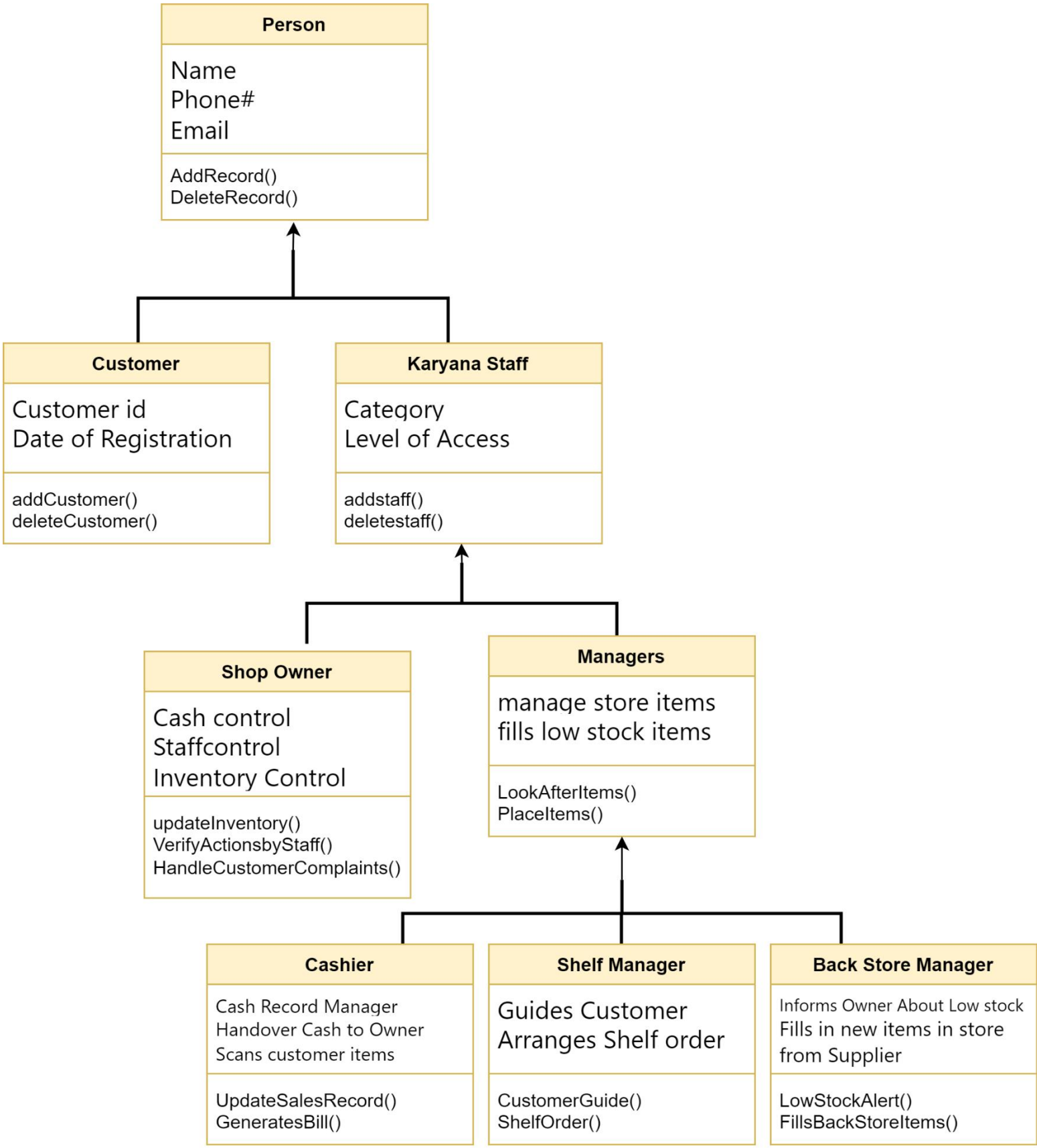
Class (Attributes and Operations):

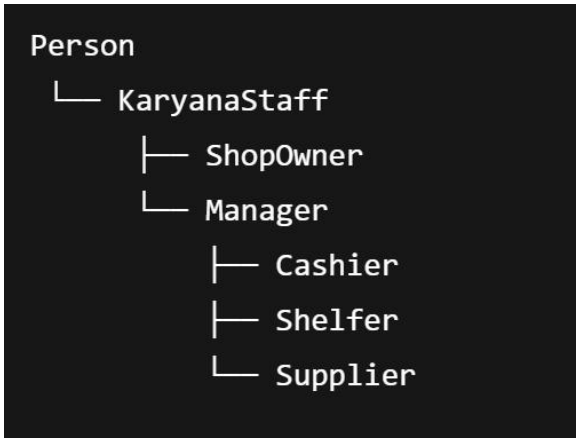
Point of Sales
Customer id Cashier Cart Date Time Items Barcode Scanner Bill Cash Cash Drawer
Scan() Add to Cart() Calculate Bill() Print Bill() Update Sales Record() Update transaction Record()

Inventory Management
Product code Barcode Price Manufacturer Supplier Stock level Shop owner Flavor Quantity Expiry Date
AddNewItem() DeleteItem() PriceUpdate() InventoryUpdate() Supplier Update() StockLevelAlert()

Sales and Report Analysis
Product ID Quantity Total price Timestamp Sales Data Mean Average Trendy Item
getSalesbyProduct() getSalesbyCategory() getSalesDaily() getSalesWeekly() getSalesMonthly() MostSoldStock() DeadStock()

A generalization hierarchy with added detail for 'Persons Related to





OOP concepts and Ideas I included in the code:

- Pointers and Dynamic memory
- Login password
- Constructor: default, parameter,deep copy
- Destructor
- Encapsulation
- Static
- Const function
- Mutable
- Operator overloading
- Inheritance
- Friend func
- Base class ptr virtual override
- Template
- st library vector
- deque
- Iterators
- Pure virtual
- abstraction
- File handling
- Polymorphism
- Regex
- Error Checking and Input Validation
- Use of _getch()
- Multilevel Inheritance ShopOwner → KaryanaStaff → Person
- Hierarchical Inheritance
- Multiple classes (Customer, KaryanaStaff) inherit from the same base class (Person).
- Multiple Inheritance
- Cashier inherits from both KaryanaStaff and SoldItem.
- Aggregation (Has-A Relationship)
- Cashier uses the CustomerLine to fetch Customer data for billing (not owns it).
- Friend Class
- Inventory<Item> is a friend of Item.
- Forward Declaration of Classes

Code:

```
#include <iostream>
#include <windows.h>//for set color and gotoxy
#include <conio.h>//for getcharacter
#include <string> //for getline
#include <cstring>// char string
#include<deque>//customers comes in line
#include<vector>//for inventory
#include<fstream>//file handling
#include<sstream>//provides access to a character array through the stream I/O interface
#include <regex>//only string no numbers
#include<cstdlib>//used for conversion atoi
using namespace std;
//global var
#define max 1000
#define filename "inventory.csv"
//global func:
bool isValidName(const string& name)
{
    // Only letters and spaces allowed
    return regex_match(name, regex("^[A-Za-z ]+$"));
}
void setcolor(int color)
{
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),color);
}
void gotoxy(int x,int y)
{
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);

}
    template<typename T>//forward class declaration
    class Inventory;
class Item {
    public:
    string name;
    double price;
    int quantity;
    protected://encapsulation
    int* id;
    mutable int MnfctrYr=2024;//mutable
    public:
    friend class Inventory<Item>; //friend class
    static int itemCount;//static
    Item():price(0),quantity(0){itemCount++;}//default constructor
    Item(string name,double p,int q,int i):price(p),quantity(q)//parameterized constructor
    {
        this->name=name;//this pointer
        id=new int;//dynamic memory and ptr
        *id=i;
        itemCount++;
    }
    Item(const Item &other)// deep copy constructor
    {
        name=other.name;
        price=other.price;
        quantity=other.quantity;
        id=new int;
        *id=*other.id;
        itemCount++;
    }
    static void getItemCount()//static func
    {
        cout<<"number of items created:"<<itemCount<<endl;
    }
    void ShowManufacture()const; //const func parameter
    bool operator==(Item const &obj)//operator overloading
    {
       return this->price==obj.price;
    }
    ~Item()//destructor
    {}
};
int Item::itemCount=0;//scope resolution
void Item::ShowManufacture()const //member (outside)
{
 MnfctrYr=2025;
 cout << "\nID\tName\tPrice\tQuantity  MnfctrYr\n";
 cout << *id<< "\t" <<name << "\t" <<price << "\t" <<quantity<< "\t  " <<MnfctrYr<< endl;
}
class SoldItem;//class forwarding
```

```
vector<SoldItem>solditem;
class SoldItem:public Item //inheritance
{
    static int solditemcount;
    public:
    SoldItem(){}
    SoldItem(const SoldItem& other) : Item(other)
     {
    this->quantity = other.quantity;
    this->name = other.name;
    // Copy other relevant fields if needed
     }
    SoldItem(string n,int q )
    {
    name=n;
    quantity=q;
    SoldItem::solditemcount++;
    }
    void selectItem()
    {
    string n;
    int q;
    cout<<"select item name and quantity:\n";
    cin>>n>>q;
    SoldItem soldobj(n,q);
    solditem.push_back(soldobj);
}
    void displayTotalSales()
{
    setcolor(6);
    cout << "Total Sales: Rs." << SoldItem::solditemcount<< endl;
}
};
int SoldItem::solditemcount=0;
template<typename T>//template class
class Inventory{
    vector<T*>arr;
    public:
    void addItem()
    {
    if (Item::itemCount >= max)
    {
        cout << "Inventory is full. Cannot add more items.\n";
        return;
    }
    string n;
    double p;
    int q,i;
    while (true)
    {
    cout << "Enter item name: ";
    getline(cin >> ws,n);//`ws` skips leading whitespaces
    if (isValidName(n)) break;
    cout << "Invalid name! Only letters and spaces allowed.\n";
    }
    cout << "Enter item price: ";
    cin >>p;
    cout << "Enter item quantity: ";
    cin >> q;
    cout << "Enter item id: ";
    cin >> i;
    Item* t=new Item(n,p,q,i);
    arr.push_back(t);
    cout << "Item added to inventory.\n";
    }
    void removeItem()
    {
    if (arr.empty())
    {
        cout << "Inventory is empty. Cannot remove item.\n";
        return;
    }
    typename vector<T*>::iterator it;// iterator created
    string n;
    while (true)
    {
    cout << "Enter item name to remove: ";
    getline(cin >> ws,n);//`ws` skips leading whitespaces
    if (isValidName(n)) break;
    cout << "Invalid name! Only letters and spaces allowed.\n";
    }
    for(it=arr.begin();it!=arr.end();it++)
    {
        if((*it)->name==n)
        {
```

```
            delete *it;         // Free dynamically allocated memory
            arr.erase(it);      // Remove pointer from vector
            Item::itemCount--;
            cout << "Item \"" << n << "\" removed from inventory.\n";
            return;
        }
    }

    }
    void searchItem()
    {
    typename vector<T*>::iterator it;// iterator created
    string n;
    while (true)
    {
    cout << "Enter item name to search: ";
    getline(cin >> ws,n);//`ws` skips leading whitespaces
    if (isValidName(n)) break;
    cout << "Invalid name! Only letters and spaces allowed.\n";
    }
    for(it=arr.begin();it!=arr.end();it++)
    {
        if((*it)->name==n)
        {
            cout << "Item found in inventory.\n";
            return;
        }
    }
   cout << "Item not found!\n";
    }
    void showItems()
   {
        cout << "Inventory:\n";
        for (auto& t : arr)
        {
          t->ShowManufacture();
        }
    }
    Item* findItemByName( const string& itemName)
     {
    for (auto& t : arr) {
        if (t->name == itemName) {
            return t;
        }
    }
        return nullptr;
    }
    void saveData()
    {
    ofstream outFile(filename);
         if (!outFile)
        {
        cout << "Error opening file for writing.\n";
        return;
        }
        for (auto& t : arr)
        {
        outFile<<*t->id<<","<<t->name<<","<<t->price<<","<<t->quantity<<","<<t->MnfctrYr<< "\n";
        }
        outFile.close();
         cout << "Inventory data saved to " << filename << "\n";
    }
    void loadData() // load data everytime program start
    {
    ifstream inFile(filename);
         if (!inFile)
        {
        cout << "Error opening file for reading.\n";
        return;
        }
        string line,idstr,namestr,pricestr,quantitystr,yearstr;
       while(getline(inFile,line))//takes from file enter in line
        {stringstream ss(line); //arrange line
            getline(ss,idstr,',');//from ss to each seperate data string
            getline(ss,namestr,',');
            getline(ss,pricestr,',');
            getline(ss,quantitystr,',');
            getline(ss,yearstr,',');
            int id = stoi(idstr);
            double price = stod(pricestr);
            int quantity = stoi(quantitystr);
            int year = stoi(yearstr);
            Item* t = new Item(namestr, price, quantity, id);
            t->MnfctrYr = year;
            arr.push_back(t);
```

```
    }

        inFile.close();
        cout << "Inventory data loaded from: " << filename << "\n";
    }
    ~Inventory()
    {
        for (auto& t : arr)
        {
           delete t;
        }
    }
};
// Hierarchical+ Multilevel Inheritance
class Person //base/super/parent class
{
protected:
    string name;
    string phone;
    string email;
public:
 virtual void Login()=0;          //absstraction, pure virtual func, interface
 virtual void Logout()=0;
 virtual void showDisplay()=0;
 virtual void generatebill(Inventory<Item>&inv)=0;
 virtual void supply()=0;
 virtual void searchItem()=0;
};
// derived/sub/child class
class Customer:public Person
{
    private:
    string customerID;
    string address;
    string paymentMethod;
    double totalSpent;
    int loyaltyPoints;
    bool isMember;
    public:
    friend void Setpersonalinfo(Customer&);//friend func to access private data
    friend void Getpersonalinfo(Customer&);
    virtual void Login()override{}
    virtual void Logout()override{}
    virtual void showDisplay()override{}
    virtual void generatebill(Inventory<Item>&inv)override{}
    virtual void supply()override{}
    virtual void searchItem()override{}
};
    deque<Customer>CustomerLine;
void Setpersonalinfo(Customer &c)
{
    while (true)
    {
    cout<<"enter customer name: ";
    getline(cin >> ws,c.name);//`ws` skips leading whitespaces
    if (isValidName(c.name)) break;
    cout << "Invalid name! Only letters and spaces allowed.\n";
    }
    cout<<"enter customer payment menthod: ";
    cin>>c.paymentMethod;
    cout<<"enter customer phone: ";
    cin>>c.phone;
    cout<<"enter customer address: ";
    cin>>c.address;
    CustomerLine.push_front(c);
}
void Getpersonalinfo(Customer &c)
{
    cout<<"First in queue customer name: ";
    cout<<c.name<<endl;
    cout<<"First in queue customer payment menthod: ";
    cout<<c.paymentMethod<<endl;
    cout<<"First in queue customer phone: ";
    cout<<c.phone<<endl;
    cout<<"First in queue customer address: ";
    cout<<c.address<<endl;
}
class KaryanaStaff:public Person
{
    protected:
    string category;
    string levelOfAccess;
    virtual void showDisplay()override{}
    virtual void generatebill(Inventory<Item>&inv)override{}
    virtual void supply()override{}
```

```
    virtual void searchItem()override{}
};
class ShopOwner:public KaryanaStaff
{
    public:
   //to overcome abstract class
    void generatebill(Inventory<Item>& inv) override {}
    void supply() override {}
    void searchItem() override {}
    ShopOwner()
    {
    showDisplay();
    }
    void Login()
    {
    string fullname;
    char savedpass[5] = {'1','2','3','4','\0'};
    char pass[5];
    while (true)
    {
    cout << "\nPlease Login To continue into the Karyana Software Admin Username:";
    getline(cin >> ws, fullname);//`ws` skips leading whitespaces
    if (isValidName(fullname)) break;
    cout << "Invalid name! Only letters and spaces allowed.\n";
    }
    do {

        cout << "Enter 4 digit Password: \n";
        for (int i = 0; i < 4; i++)
        {
            pass[i] = _getch();//to get char without showing on screen for privacy
            cout << "*";
        }
        pass[4] = '\0';//strcmp uses null to know where to end
        if (!strcmp(pass, savedpass))//compare two c-strings returns 0 if equal
        {
            cout << " \n Welcome to the Inventory and Billing Software " << fullname << " Boss" << endl;
            return;
        }
        else
            cout << "\n Invalid password "<<fullname <<". Try again,You are not the admin"<<endl;
    } while (1);
    }
    void Logout()
    {
    cout<<"Good bye Boss,saving data in Data Base :)\n";
    }
    void showDisplay()
    {
    setcolor(3);
    gotoxy(10,10);
    cout << "****************************************"<<endl;
    gotoxy(10, 11);
    cout << "* JAWAD MART INVENTORY AND BILLING SYSTEM *"<<endl;
    gotoxy(10, 12);
    cout << "****************************************";
    gotoxy(0, 13);
    setcolor(2);
    }
};
class Cashier:public KaryanaStaff,public SoldItem //multiple inheritance
{
    double total = 0.0;
    public:
    virtual void Login()override{}
    virtual void Logout()override{}
    virtual void showDisplay()override{}
    virtual void supply()override{}
    virtual void searchItem()override{}
    void generatebill(Inventory<Item>&inv)
    {
       if(solditem.empty())
       {
        cout<<"no items selected, can't generate bill!\n";
        return;
       }
        Customer Last= CustomerLine.back(); //Aggregation
       Getpersonalinfo(Last);
       CustomerLine.pop_back();
       cout << "\n--- Bill Receipt ---\n";
       double total = 0.0;
       for (const SoldItem& s : solditem)
      {
      Item* matched = inv.findItemByName(s.name);
        double itemTotal = matched->price * s.quantity;
```

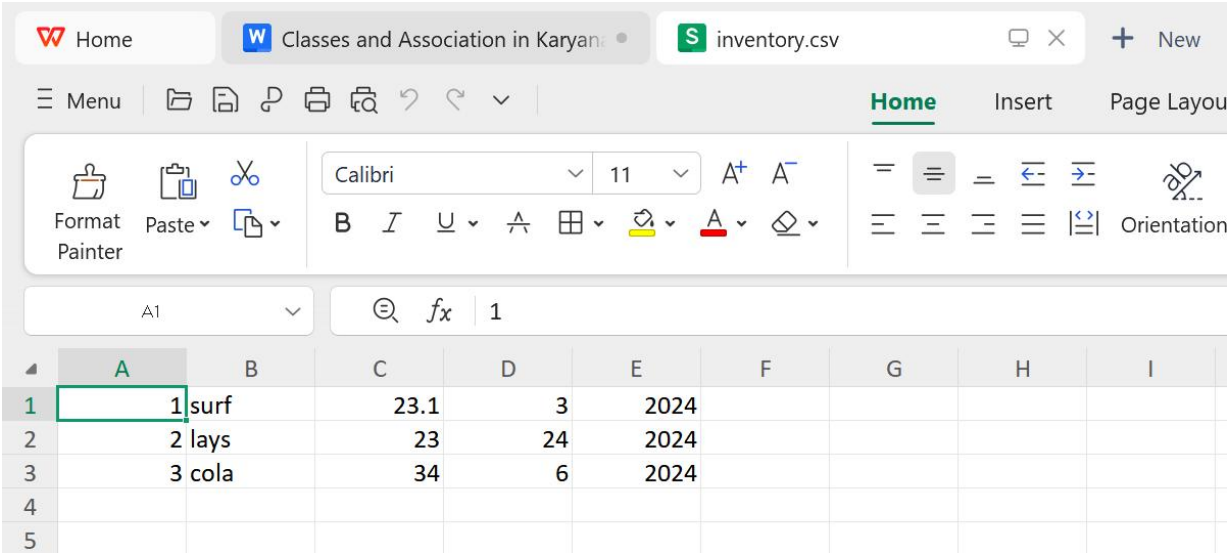
```
        cout << matched->name << " x" << s.quantity << " = " << itemTotal << "\n";
        total += itemTotal;
      }
      cout << "------------------------\n";
      cout << "Total Bill: " << total << "\n";
      // Clear sold items after billing
      solditem.clear();
    }
};
class Shelfer:public KaryanaStaff
{
    public:
    virtual void Login()override{}
    virtual void Logout()override{}
    virtual void showDisplay()override{}
    virtual void generatebill(Inventory<Item>&inv)override{}
    virtual void supply()override{}
    void searchItem()
    {
    cout<<" Searching Item . . .\n";
    }
};
class Supplier:public KaryanaStaff
{
    public:
    virtual void Login()override{}
    virtual void Logout()override{}
    virtual void showDisplay()override{}
    virtual void generatebill(Inventory<Item>&inv)override{}
    virtual void searchItem()override{}
    void supply()
    {
    cout<<" Supplies Items to Shop Owner. . .\n";
    }
};
int main()
{

     KaryanaStaff*  k;
     ShopOwner owner;
     k = &owner;
     Customer c;
     Cashier cashier;
     Inventory <Item>inv;//template class obj
     SoldItem sold;
     int choice1,choice2;
     inv.loadData();
    do {
        cout << "\n1. Owner--handle Inventory\n2. Customer--purchase Item\n3. Cashier--generate Bill\n4. Exit Karyana
System\n";
        while (true)
         {
         cout << "Enter your role in Karyana: ";
         cin >> choice1;
        if (!cin.fail() && choice1 > 0) break;
        cout << "Invalid input! Enter a valid number.\n";
         cin.clear(); // clear error state
         cin.ignore(1000, '\n'); // discard bad input
        }
        switch (choice1) {
        case 1:
        k->Login();
            do {
                cout << "\n1. Add Item to Inventory\n2. Delete Item from Inventory\n3. Search Item from Inventory\n4. LogOut
of Admin\n";
                cout << "Enter your choice: ";
                cin >> choice2;
                 switch (choice2)
                 {
                    case 1:
                    inv.addItem();
                    break;
                    case 2:
                    inv.removeItem();
                    break;
                    case 3:
                    inv.searchItem();
                    break;
                    case 4:
                    k->Logout();
                    inv.saveData();
                    break;
                    default:
                    cout << "Invalid choice. Please try again.\n";
                 }
```

```
            }while(choice2 != 4);
        break;
        case 2:
            do {
                cout << "\n1. Shelfer shows Items Available\n2. Select Items to Purchace\n3. Checkout\n";
                cout << "Enter your choice: ";
                cin >> choice2;
                 switch (choice2)
                 {
                    case 1:
                    inv.showItems();
                    break;
                    case 2:
                    sold.selectItem();
                    break;
                    case 3:
                    cout << "Going to Cashier for Bill Payment.\n";
                    break;
                    default:
                    cout << "Invalid choice. Please try again.\n";
                  }
            }while(choice2 != 3);
        break;
        case 3:
            do {
                cout << "\n1. Ask Customer Details\n2. Generate Bill\n";
                cout << "Enter your choice: ";
                cin >> choice2;
                 switch (choice2)
                 {

                    case 1:
                    Setpersonalinfo(c);
                    break;
                    case 2:
                    cashier.generatebill(inv);
                    break;
                    default:
                    cout << "Invalid choice. Please try again.\n";
                  }
               }while(choice2 != 2);
        break;
        case 4:
        cout << "Shutting Down Karyana System.\n";
        break;
        default:
        cout << "Invalid choice. Please try again.\n";
        }
    } while (choice1 != 4);
return 0;
}
```

CSV FILE FOR INVENTORY RECORD:



OUTPUT:

```
          ******************************************
          * JAWAD MART INVENTORY AND BILLING SYSTEM *
          ******************************************
Inventory data loaded from: inventory.csv

1. Owner--handle Inventory
2. Customer--purchase Item
3. Cashier--generate Bill
4. Exit Karyana System
Enter your role in Karyana: 1

Please Login To continue into the Karyana Software Admin Username:jawad12
Invalid name! Only letters and spaces allowed.

Please Login To continue into the Karyana Software Admin Username:Jawad Hassan
Enter 4 digit Password:
****
 Welcome to the Inventory and Billing Software Jawad Hassan Boss

1. Add Item to Inventory
2. Delete Item from Inventory
3. Search Item from Inventory
4. LogOut of Admin
Enter your choice: 1
Enter item name: cola
Enter item price: 34
Enter item quantity: 6
Enter item id: 3
Item added to inventory.

1. Add Item to Inventory
2. Delete Item from Inventory
```

```
3. Search Item from Inventory
4. LogOut of Admin
Enter your choice: 4
Good bye Boss,saving data in Data Base :)
Inventory data saved to inventory.csv

1. Owner--handle Inventory
2. Customer--purchase Item
3. Cashier--generate Bill
4. Exit Karyana System
Enter your role in Karyana: 2

1. Shelfer shows Items Available
2. Select Items to Purchace
3. Checkout
Enter your choice: 1
Inventory:

ID      Name    Price   Quantity  MnfctrYr
1       surf    23.1    3         2025

ID      Name    Price   Quantity  MnfctrYr
2       lays    23      24        2025

ID      Name    Price   Quantity  MnfctrYr
3       cola    34      6         2025

1. Shelfer shows Items Available
2. Select Items to Purchace
3. Checkout
Enter your choice: 2
select item name and quantity:
```

```
surf
1

1. Shelfer shows Items Available
2. Select Items to Purchace
3. Checkout
Enter your choice: 2
select item name and quantity:
lays
1

1. Shelfer shows Items Available
2. Select Items to Purchace
3. Checkout
Enter your choice: 2
select item name and quantity:
cola
1

1. Shelfer shows Items Available
2. Select Items to Purchace
3. Checkout
Enter your choice: 3
Going to Cashier for Bill Payment.

1. Owner--handle Inventory
2. Customer--purchase Item
3. Cashier--generate Bill
4. Exit Karyana System
Enter your role in Karyana: 3

1. Ask Customer Details
```

```
1. Ask Customer Details
2. Generate Bill
Enter your choice: 1
enter customer name: ali
enter customer payment menthod: cash
enter customer phone: 0334567742
enter customer address: islamabad

1. Ask Customer Details
2. Generate Bill
Enter your choice: 2
First in queue customer name: ali
First in queue customer payment menthod: cash
First in queue customer phone: 0334567742
First in queue customer address: islamabad

--- Bill Receipt ---
surf x1 = 23.1
lays x1 = 23
cola x1 = 34
-------------------------
Total Bill: 80.1
```