**Step 1: Case Study Scenario**

Miss Client wants to develop a software system for her departmental store. She wants this

system to have the following functionalities.

As an Admin, she can

* Add Products.
* View All Products.
* Find Product with Highest Unit Price.
* View Sales Tax of All Products.
* Products to be Ordered. (less than threshold)

Following is the information that is required to save for the product.

Name of Product.

Product Category.

Product Price.

Available Stock Quantity.

Minimum

Stock threshold

Quantity after which the owner wants to order the product.

On All Grocery type of products, the sales tax is 10%, on all fruit types the tax is 5% and

if there is any other type the tax is 15%

**She also wants that**

1. The Customers to view all the products

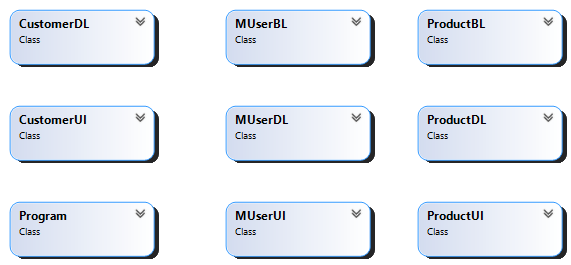
2. Customers can buy the products (When a customer buy a product then its

quantity should decrease from the stock)

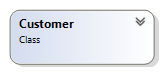
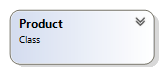
3. Generate invoice (While calculating the price of the products that the

customer has bought, sales tax should be applied.)

**Step 2: Domain Model with Only the Class Names**



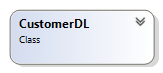
**Step 3: Domain Model with Relations and Constraints**



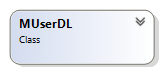
Purchases

Contains

Contains



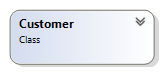
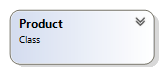




Contain

possess

**Step 4: Domain Model with Multiplicity**



Purchases

1

∞

1

∞

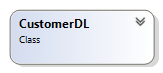
∞

Contains

Contains

1

1

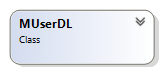




1

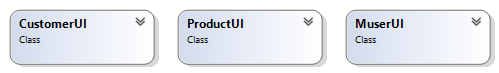
∞

Contains

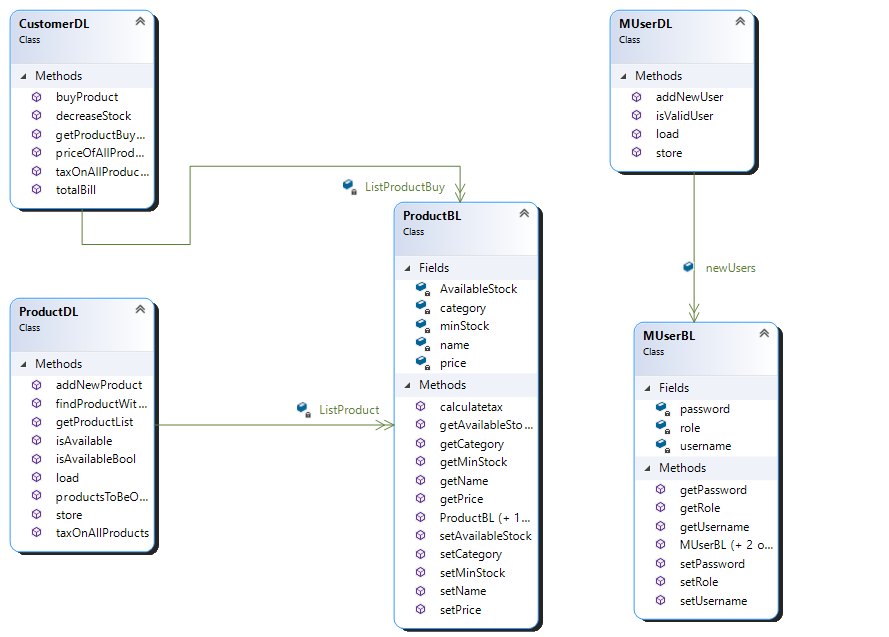


1

posses



**Step 5: Class Diagram Attributes and Functions**

****

∞

1

1

∞

**DL Code: CustomerDL.cs**

class CustomerDL

{

public static void decreaseStock(ProductBL x)

{

x.setAvailableStock(x.getAvailableStock() -1);

}

public static void buyProduct(ProductBL name)

{

ListProductBuy.Add(name);

decreaseStock(name);

}

public static float taxOnAllProductsCust()

{

float taxOnAllProducts = 0;

for (int i = 0; i < ListProductBuy.Count; i++)

{

if (ProductDL.isAvailableBool(ListProductBuy[i].getName()))

{

taxOnAllProducts = taxOnAllProducts + ListProductBuy[i].calculatetax();

}

}

return taxOnAllProducts;

}

public static float priceOfAllProductsCust()

{

float price = 0;

for (int i = 0; i < ListProductBuy.Count; i++)

{

price = price + ListProductBuy[i].getPrice();

}

return price;

}

public static float totalBill()

{

float total = priceOfAllProductsCust() - taxOnAllProductsCust();

return total;

}

public static List<ProductBL> getProductBuyList()

{

return ListProductBuy;

}

}

**BL Code: ship.cs**

**DL Code: shipDL.cs**

**UI Code: shipUI.cs**

**BL Code: Product.cs**

class ProductBL

{

private string name;

private string category;

private int price;

private int AvailableStock;

private int minStock;

public ProductBL() { }

public ProductBL(string name,string category,int price,int AvailableStock,int minStock)

{

this.name = name;

this.category = category;

this.price = price;

this.AvailableStock = AvailableStock;

this.minStock = minStock;

}

public void setName(string name)

{

this.name = name;

}

public void setCategory(string category)

{

this.category = category;

}

public void setPrice(int price)

{

this.price = price;

}

public void setAvailableStock(int AvailableStock)

{

this.AvailableStock = AvailableStock;

}

public void setMinStock(int minStock)

{

this.minStock = minStock;

}

public string getName()

{

return name;

}

public string getCategory()

{

return category;

}

public int getPrice()

{

return price;

}

public int getAvailableStock()

{

return AvailableStock;

}

public int getMinStock()

{

return minStock;

}

public float calculatetax()

{

if (category == "grocery") return (float)(0.10 \* price);

else if (category == "fruit") return (float)(0.5 \* price);

else return (float)(0.15 \* price);

}

}

**U**

**BL Code: MUser.cs**

class MUserBL

{

private string username;

private string password;

private string role;

public MUserBL(string username,string password,string role)

{

this.username = username;

this.password = password;

this.role = role;

}

public MUserBL(string username, string password)

{

this.username = username;

this.password = password;

}

public MUserBL()

{

}

public string getUsername()

{

return username;

}

public string getPassword()

{

return password;

}

public void setUsername(string username)

{

this.username = username;

}

public void setPassword(string password)

{

this.password = password;

}

public string getRole()

{

return role;

}

public void setRole(string role)

{

this.role = role;

}

}

**Driver Program: Program.cs**

class Program

{

static void Main(string[] args)

{

MUserDL.load();

ProductDL.load();

int option = 0;

do

{

option = MUserUI.MainMenu();

if (option == 1)

{

string role = MUserDL.isValidUser(MUserUI.takeSignInInput());

if(role == "admin")

{

int admin\_option = 0;

do

{

admin\_option = MUserUI.adminMenu();

if (admin\_option == 1)

{

ProductDL.addNewProduct((ProductUI.takeUserInput()));

}

else if (admin\_option == 2)

{

ProductUI.viewProducts(ProductDL.getProductList());

}

else if (admin\_option == 3)

{

ProductUI.printString(ProductDL.findProductWithHighestPrice());

}

else if (admin\_option == 4)

{

ProductUI.printFloat(ProductDL.taxOnAllProducts());

}

else if (admin\_option == 5)

{

ProductUI.viewProducts(ProductDL.productsToBeOrdered());

}

}

while (admin\_option != 6);

}

else if(role == "customer")

{

int customer\_option = 0;

do {

customer\_option = MUserUI.customerMenu();

if(customer\_option == 1)

{

ProductUI.viewProducts(ProductDL.getProductList());

}

else if (customer\_option == 2)

{

CustomerDL.buyProduct(ProductDL.isAvailable(CustomerUI.takeCustomerInput()));

}

else if (customer\_option == 3)

{

CustomerUI.printInvoice(CustomerDL.priceOfAllProductsCust(),CustomerDL.taxOnAllProductsCust(),CustomerDL.totalBill());

}

} while (customer\_option != 4);

}

}

else if(option == 2)

{

MUserDL.addNewUser(MUserUI.takeSignUpInput());

}

}

while (option != 3);

MUserDL.store();

ProductDL.store();

}

}