

Jevon Harriott
Jamari Daniel
Group #16

Requirement Function

1.1. Function Specification for the Shopping Cart Application

The main purpose for the Shopping Cart Application is to use Object Oriented development process to create an application.

Application Features

The Shopping Cart allows customers to purchase items by selecting a product and purchasing it. The Shopping Cart allows seller to update the inventory by adding a product name, invoice price, sell price and item quantity. The seller can also see information about their inventory or view basic sale statistics.

The application requirement specification

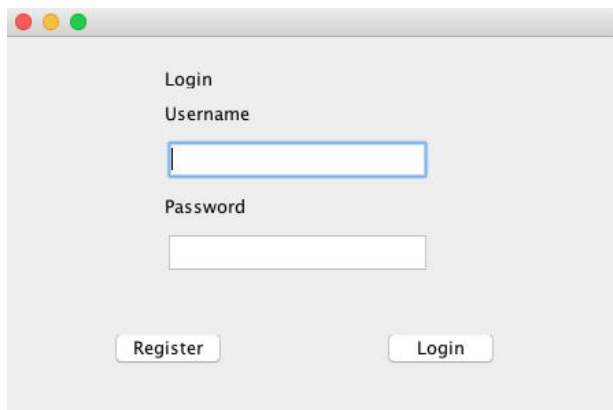
1. A customer or seller can login with username and password.
2. A user that is logged in as a customer:
 - a. View a list of available products displays on the screen.
 - b. If the customer selects a product, the description of the product displays on the right side of the screen.
 - c. The customer can add available products to their cart.
 - d. Once the customer has more than one product in their cart, they can checkout at anytime.
 - e. The total amount of the cart and the number of products displays on the screen.
 - f. The customer can remove or update quantity of products in their cart.
 - g. Once the customer clicks checkout ,the order confirmation page will displays on the screen.
3. When a user login as a seller:
 - a. The list of products in the inventory available.
 - b. The name, quantity , invoice, and sell price of each product in the inventory can be viewed.
 - c. The seller can add a product to the inventory by providing the following information:
 - Name
 - Price
 - Description

- Invoice Price
 - Sell Price
- d. The seller can remove product by selecting the product, then click the remove button.
 - e. The seller can modify a product quantity by clicking the modify button.
 - f. The seller can check the total cost, total revenue and total profit of their inventory.

User Interface

1. When app is launched user will be faced with the Login Screen displays:
 - a. Input box for username and password
 - b. User will be able to click register to register an account
 - c. User will be able to click login if he/she has previously registered

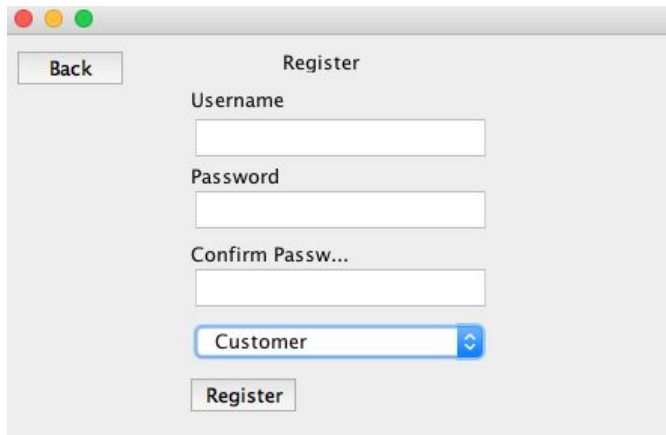
Login Page



The screenshot shows a login window with a title bar containing three colored buttons (red, yellow, green). The window has a light gray background. At the top, the word "Login" is displayed. Below it, the label "Username" is followed by a text input field with a blue border. Underneath, the label "Password" is followed by another text input field. At the bottom of the window, there are two buttons: "Register" on the left and "Login" on the right.

2. The user also has the option of register on the registration screen which displays:
 - a. Input box for username, password and confirm password
 - b. User will be able select if they are a seller or a customer
 - c. User will be able to click register when they have completed all fields

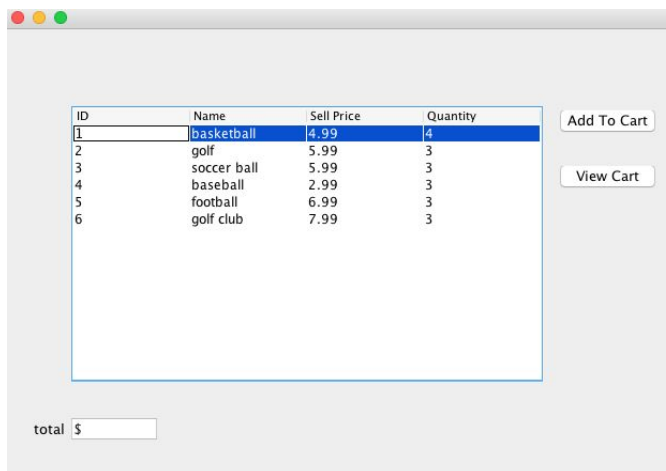
Register Page



A screenshot of a web application window titled "Register". It features a "Back" button in the top left corner. The form contains three text input fields labeled "Username", "Password", and "Confirm Passw...". Below these is a dropdown menu currently showing "Customer". At the bottom of the form is a "Register" button.

3. After a user has logged in as a customer they will see the shopping cart screen displays:
 - a. A list of available products.
 - b. The name, quantity , and price displays for each product.
 - c. A description of a product is displays when a user selects a product.
 - d. The add button is use to add products to the cart.
 - e. The total amount of the cart is display when the user add a product
 - f. The number of products in the cart is displayed on the cart button.

Shopping Cart Page



A screenshot of a web application window titled "Shopping Cart Page". It displays a table with the following data:

ID	Name	Sell Price	Quantity
1	basketball	4.99	4
2	golf	5.99	3
3	soccer ball	5.99	3
4	baseball	2.99	3
5	football	6.99	3
6	golf club	7.99	3

To the right of the table are two buttons: "Add To Cart" and "View Cart". At the bottom left, there is a "total \$" label followed by a text input field.

4. When the customer is ready to purchase they can click view cart, this will display the cart screen which displays:
 - a. The total amount for the product(s) in the cart.
 - b. The name, price and quantity for each product.
 - c. An option to modify the quantity or remove button to remove products from the cart.

- d. A checkout button.
5. Once the user clicks the checkout button, a order confirmation screen displays on the screen. This screen will display an itemized list of purchased products and the total of the transaction.
6. When the seller logs in they can see the inventory seller screen which displays:
 - a. A list of product in the inventory
 - b. The name, invoice price, sell price, and quantity for each product.
 - c. A option to add product to the inventory by entering the follow information:
name, invoice price,sell price, description and quantity.
 - d. A option to modify products quantity by selecting the modify button.
 - e. The total cost, total revenue and total profit of the products in the inventory.

Seller inventory page

The screenshot shows a window titled 'Seller inventory page'. At the top, there are input fields for 'ID', 'Invoice Pri...', 'Quantity', 'Name', 'Sell Price', and 'Description'. Below these fields is a table with the following data:

ID	Name	Invoice Price	Sell Price	Quantity
1	basketball	2.01	4.99	4
2	golf	2.0	5.99	3
3	soccer ball	2.0	5.99	3
4	baseball	2.0	2.99	3
5	football	2.0	6.99	3
6	golf club	2.0	7.99	3

To the right of the table are three buttons: 'Add', 'Update', and 'Remove'.

Application Architecture

The Shopping cart application is implemented using JAVA swing GUI and executed using command-prompt terminal.

Glossary

Profit: Revenue minus the cost.

Revenue: The sum of the sell price of all the items sold

Cost: The sum of the invoice price of all items bought in the inventory

Customer: A type of user that can use the shopping cart and make purchases

Seller: A type of user that can modify the inventory and view profit, revenue, and cost of total inventory

1.2. Essential Use Cases

Use Case Number 1	
Use Case Name	User creates account
Use Case Description	User is asked to log in or register. This will allow them to register so that they can access the store inventory
Primary Actor(s)	The user
Precondition	The user has launched the app
Trigger	The user clicks the Register button

Use Case Number 2	
Use Case Name	User logs in
Use Case Description	User is asked to login this will allow them access to the stores inventory
Primary Actor(s)	The user has created an account
Precondition	The user has created an account
Trigger	The user clicks the login button

Use Case Number 3	
Use Case Name	Customer adds item to Shopping Cart
Use Case Description	User has logged in and been identified as a customer. The customer selects an item from inventory and clicks add button to add item to the cart and update inventory(server file)
Primary Actor(s)	The User
Precondition	The user is logged in as a customer
Trigger	The Customer selects a product User clicks the add button to add item to the cart

Use Case Number 4	
Use Case Name	Customer Reviews/Update Shopping Cart
Use Case Description	Customer reviews shopping cart for correct items and quantities at the correct price. If anything is wrong Customer can update the cart.
Primary Actor(s)	The Customer
Precondition	Customer has clicked the checkout button
Trigger	Customer Clicks check button

Use Case Number 5	
----------------------	--

Use Case Name	Customer Checks Out
Use Case Description	Customer Clicks on the check out button when they are ready to make the purchase
Primary Actor(s)	The Customer
Precondition	Customer logged in
Trigger	Click on the check out button

Use Case Number 6	
Use Case Name	Seller Reviews/Updates Inventory
Use Case Description	This is when the seller wants to view the inventory and make possible changes.
Primary Actor(s)	The seller
Precondition	The seller has logged in
Trigger	Clicking on the edit inventory button

Use Case Number 7	
Use Case Name	Seller Adds New Product
Use Case Description	The seller logs in and clicks on the add button. This will open the add menu which will let the seller input the necessary information
Primary Actor(s)	The seller
Precondition	The seller has logged in

Trigger	Click add inventory
----------------	----------------------------

Use Case Number 8	
Use Case Name	Seller modifies product description, price or other information
Use Case Description	The seller logs in and clicks on the edit button. This will prompt the seller to edit the description.
Primary Actor(s)	The seller
Precondition	The seller has logged in
Trigger	Click edit button

1.3. Detailed Use Cases

User creates account

- User launches app
- System runs login frame
- User is presented with login screen
- User clicks register button
- System displays registration frame
- User inputs Username
- User inputs password
- User enters password to confirm previous password field matches
- User selects if they are seller or customer
- User clicks register button
- System checks to see if username already exist in database
- System sets account type

- System closes registration frame and opens login Frame

User logs in

- **User launches app**
- System runs login frame
- User is presented with login screen
- User inputs username
- **User inputs password**
- User clicks login button
- System check username and password to see if it's in the database
- System closes login frame and opens shopping Frame

Customer adds item to Shopping Cart

- **Customer logs in**
- **Customer selects available item from store inventory**
- **Customer clicks the add to cart button**
- **System checks availability and adds it to the cart if it's available**
- **If item is not available system will display message**
- **System updates cart total**

Customer Reviews/Update Shopping Cart

- **Customer logs in**
- **Customer adds item to cart**
- **Customer clicks cart button**
- **System displays cart frame**
- **System displays all items in cart and cart total**
- **Customer reviews list or make changes**
- **If change is made system updates the cart frame**

Customer Checks Out

- **Customer logs in**
- **Customer adds item to cart**
- **Customer clicks cart button**
- **Customer clicks the checkout button**
- **System displays Checkout frame**
- **System request a payment**
- **Customer enters payment information**
- **System processes(assuming all purchase are successful)**

- **System displays itemized list of purchased items along with a total and form of payment**

Seller Reviews/Updates Inventory

- **Seller logs in**
- **System check account type**
- **System closes login frame and opens inventory Frame**
- **System displays inventory frame**
- **Seller Makes changes**
- **Seller clicks update**
- **System saves changes**

Seller Adds New Product

- **Seller logs in**
- **System check account type**
- **System closes login frame and opens inventory Frame**
- **System displays inventory frame**
- **Seller inputs product information at top of frame**
- **Seller clicks add button**
- **System adds item and item information to inventory**
- **System updates inventory list**

Seller modifies product description, price or other information

- **Seller logs in**
- **System check account type**
- **System closes login frame and opens inventory Frame**
- **System displays inventory frame**
- **Seller selects an item**
- **Seller makes change**
- **Seller clicks update**
- **System updates information for selected product**

2. Design Specification

2.1. CRC cards

Customer	
-----------------	--

<ul style="list-style-type: none"> • Place order • Add items to the shopping cart • Pay for items 	Cart
---	-------------

Seller	
<ul style="list-style-type: none"> • Update items in the inventory • Add items to the inventory • Change the available quantity 	Inventory

Product	
<ul style="list-style-type: none"> • Product name • Product cost • Quantity • Sell Price • Invoice Price 	Inventory

Inventory	
<ul style="list-style-type: none"> • List of products 	Product

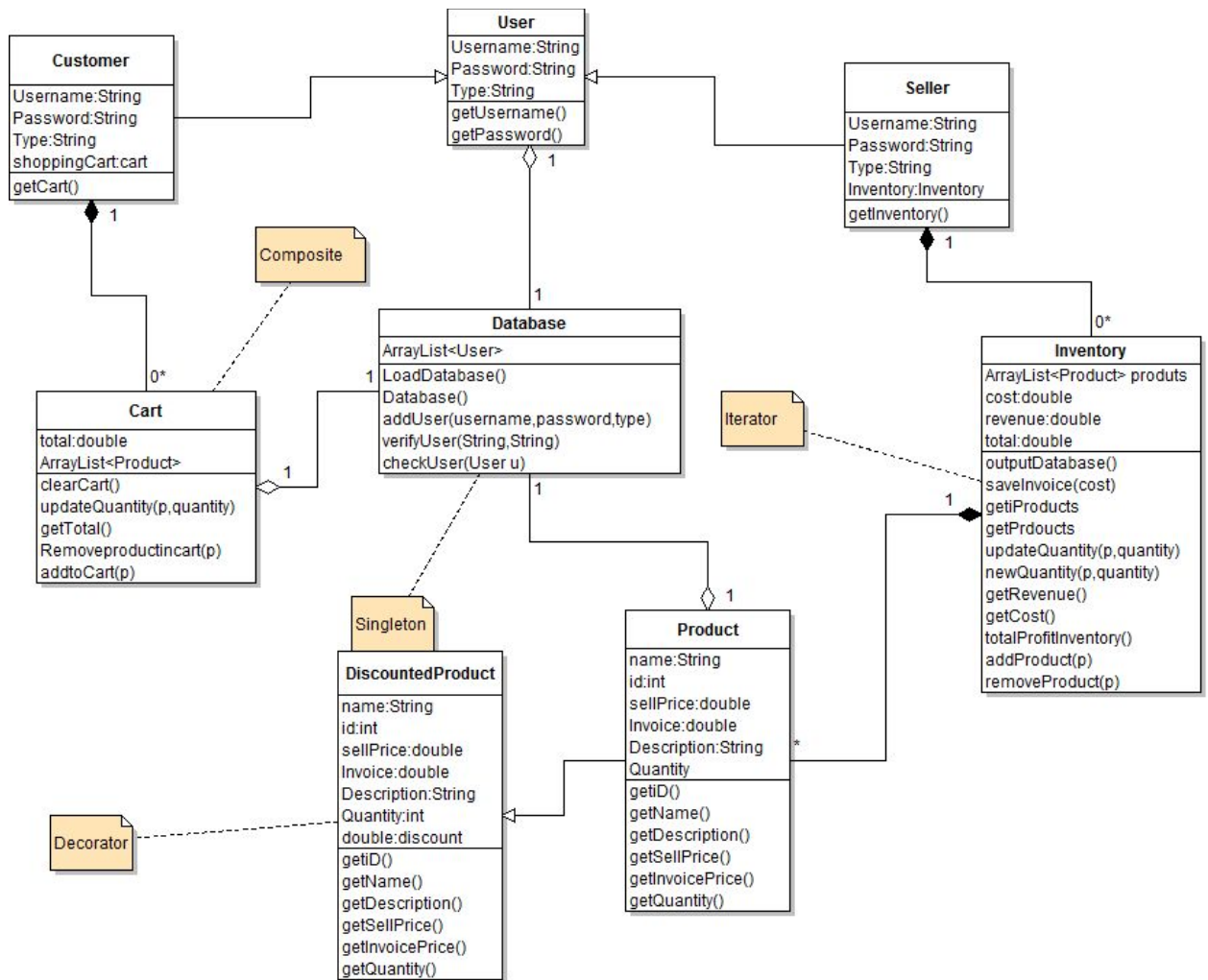
Cart	
<ul style="list-style-type: none"> • Update quantity of the products in the cart. • Total price in the cart • Remove products 	Customer

--	--

User	
<ul style="list-style-type: none"> • Username • Password 	Customer Seller

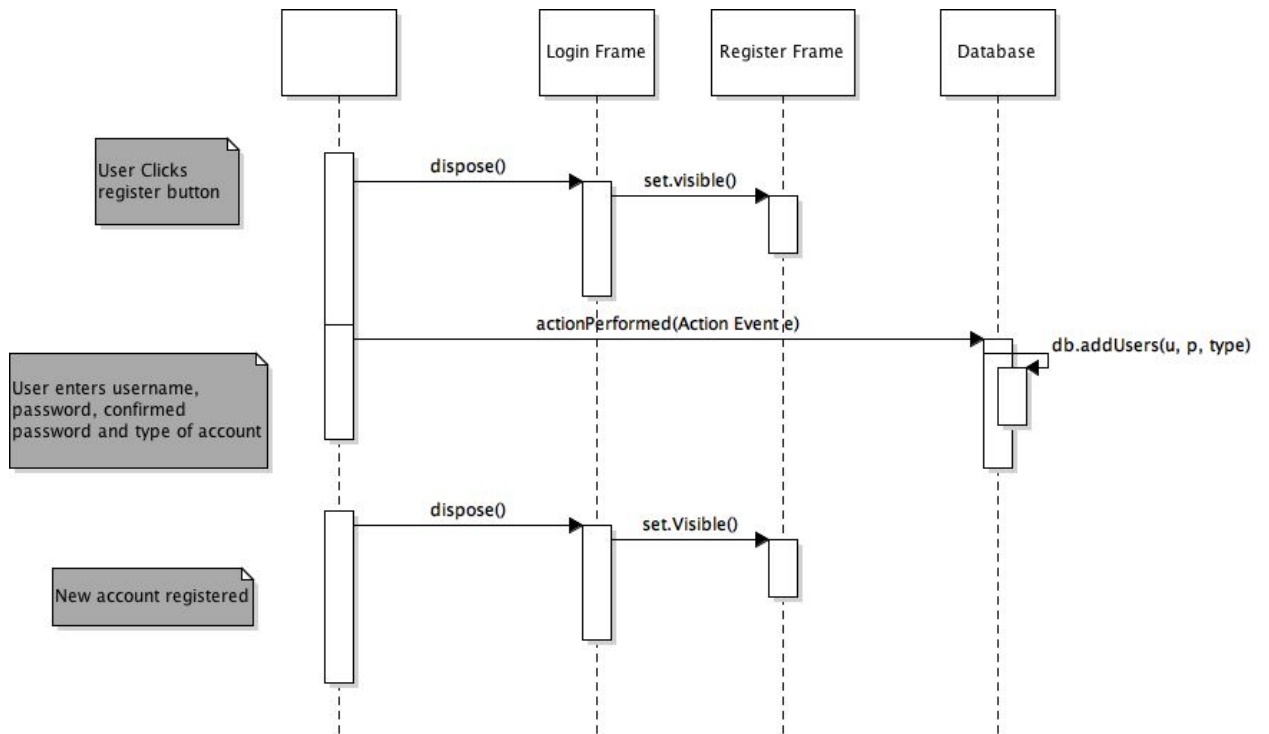
Database	
<ul style="list-style-type: none"> • Load Database • Output Database 	User

2.2. UML Class Diagram

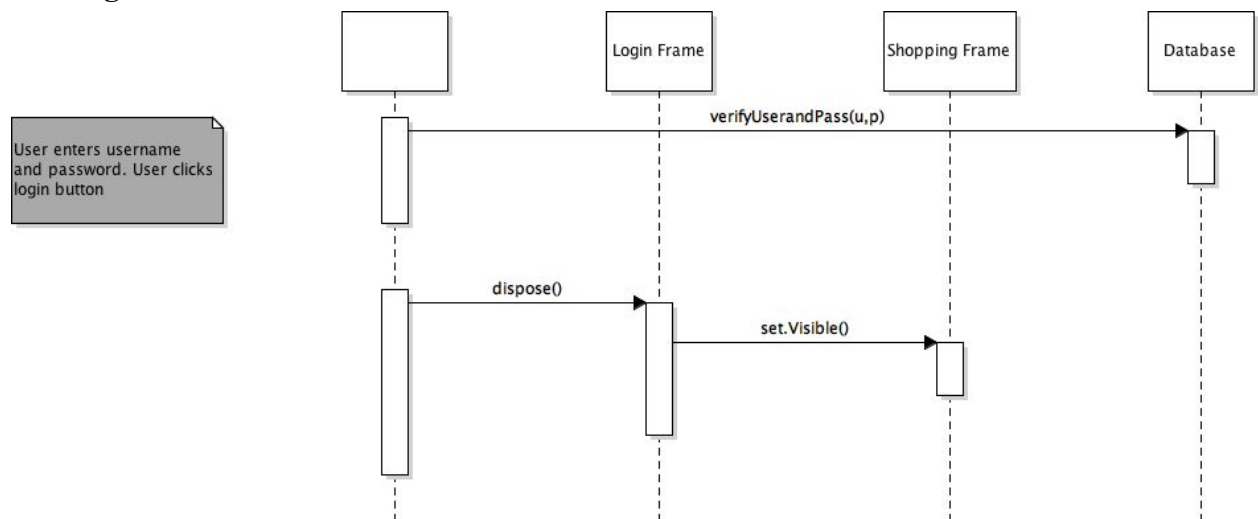


2.3. Sequence Diagram

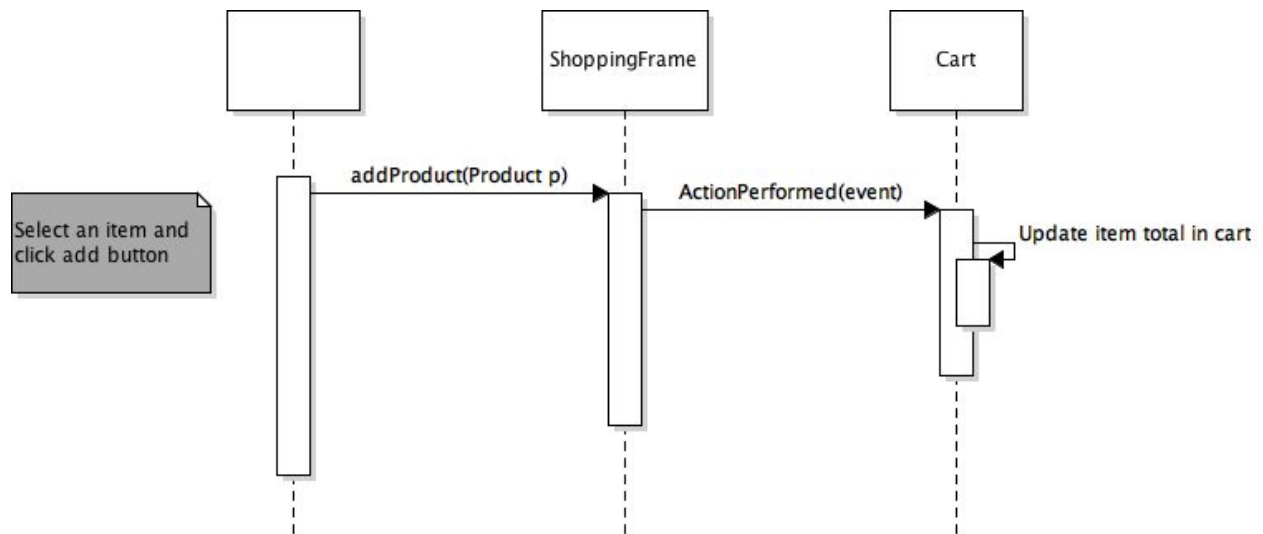
User creates account



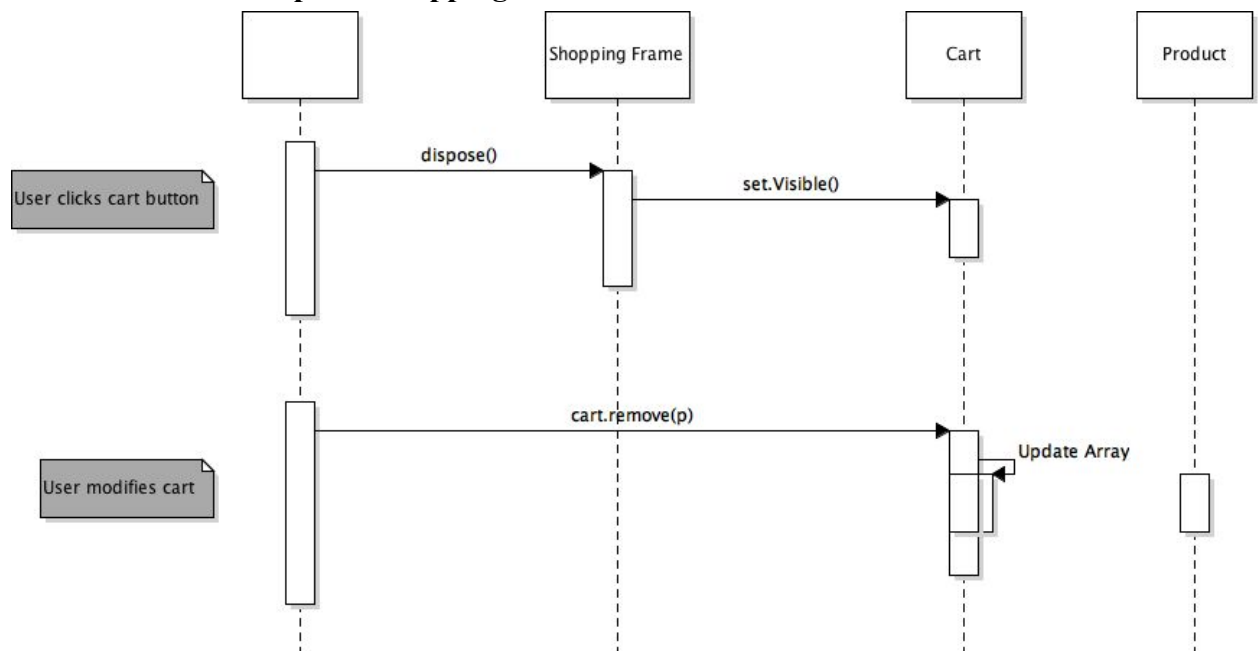
User logs in



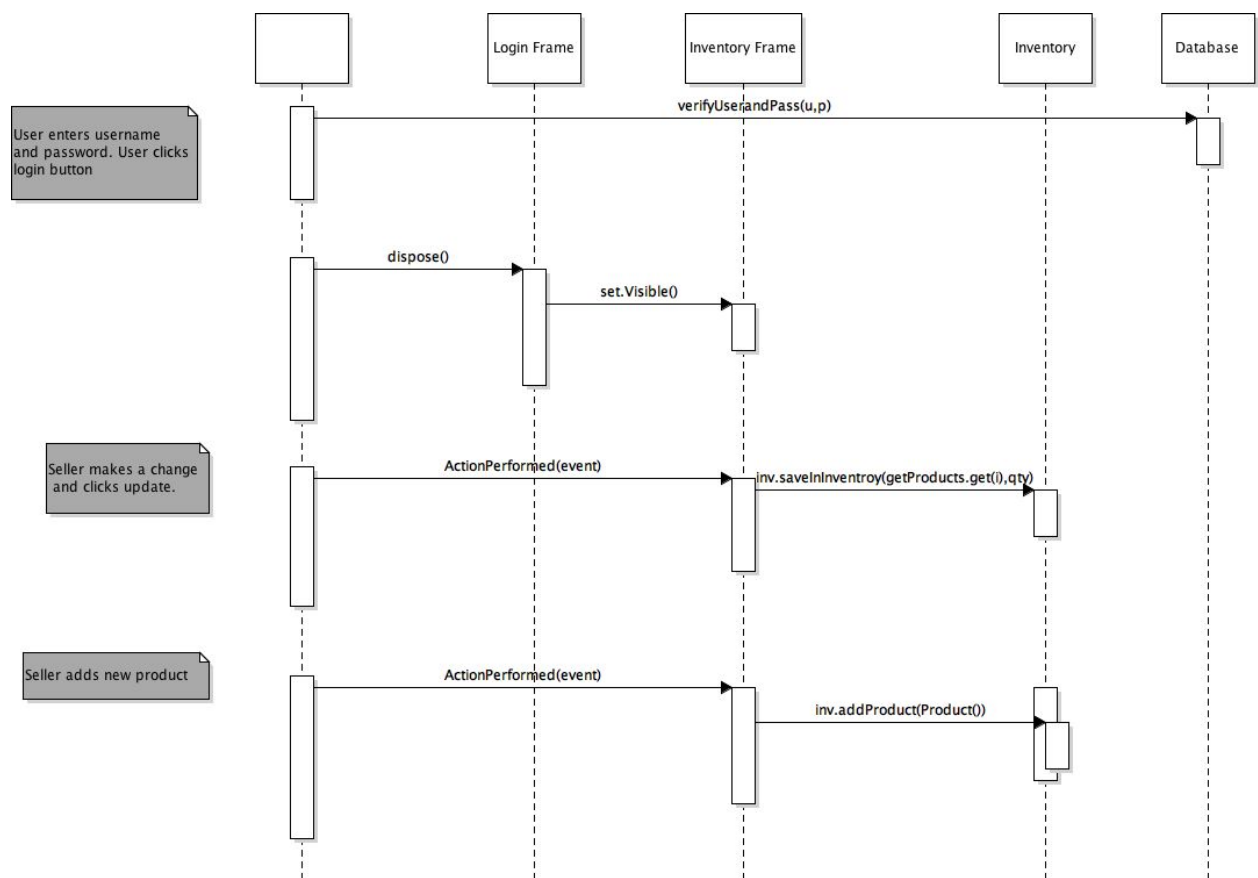
Customer adds item to Shopping Cart



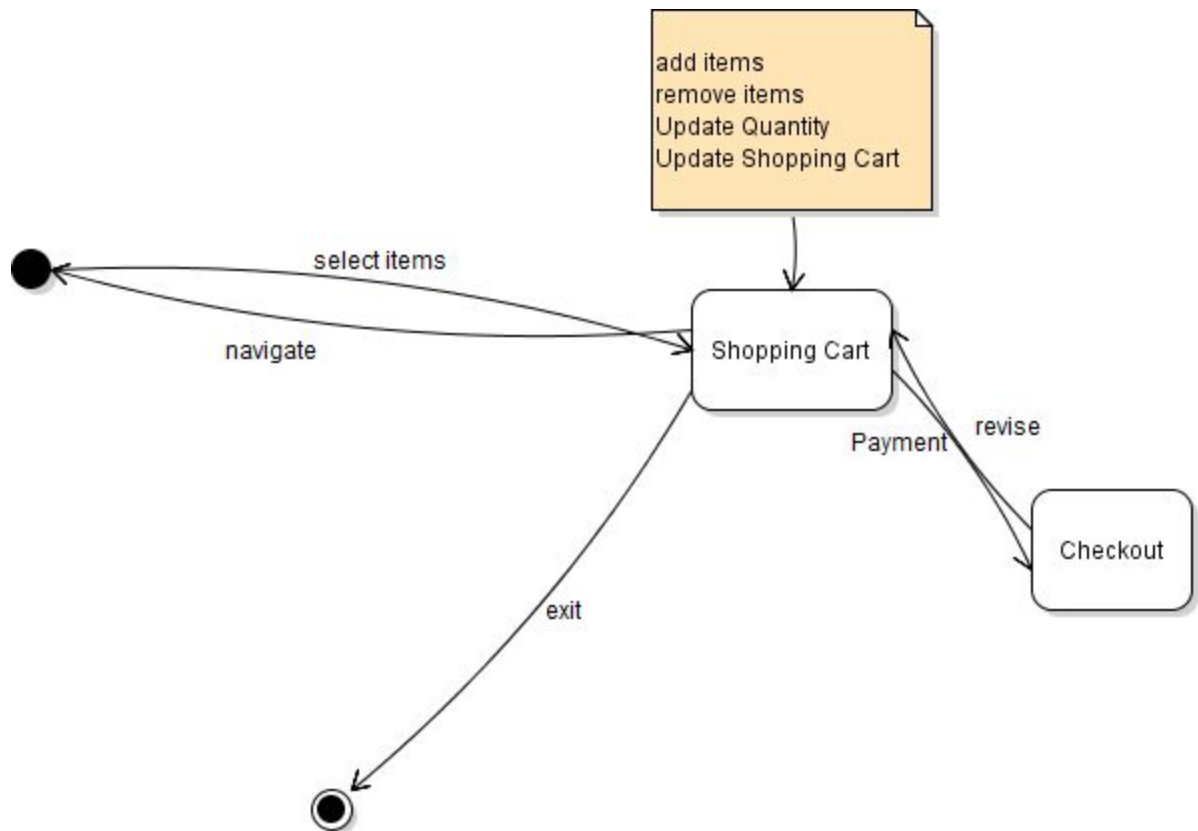
Customer Reviews/Update Shopping Cart



Seller use cases



2.4. State Diagram



Design Pattern

Compositie

Name in Design Pattern	Actual Name
Primitive	Product
Composite	Cart contain products
leaf	Product has no childrens
method()	getSellPrice()

Decorator

Name in Design Pattern	Actual Name
Component	DiscountedProduct
Concrete Component	Product

Decorator	DiscountedPrice()
method()	getSellPrice()

Iterator

Name in Design Pattern	Actual Name
Aggregate	List
ConcreteAggregate	ArrayList
Iterator	ListIterator
ConcreteIterator	Anonymous class implements ListIterator
CreateIterator	igetProduct()
next()	next()
isDone()	!hasNext()
currentItem()	hasNext()

Singleton

Name in the Design Pattern	Actual Name
Private Constructor	Database()
instance	instance
getInstance()	getInstance()

Cart.java
package project;

```
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Iterator;
```

```
import java.util.LinkedList;
```

```
public class Cart implements Serializable {
```

```
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    /**
     *
     */
```

```
    /*add product to the user cart
```

```
     * @param Product p is a object
```

```
     *
```

```
    */
```

```
        public void addToCart(Product p){
            cart.add(p);
```

```
        }
```

```
    /*
```

```
     * get the products in the user
```

```
     * @return the user products
```

```
    */
```

```
        public ArrayList<Product> getProducts(){
            return cart;
```

```
        }
```

```
    /*
```

```
     * remove the products in the user cart
```

```
     * @param Product p is the products in the user cart
```

```
    */
```

```
        public void RemoveProductinCart(Product p){
            cart.remove(p);
```

```
        }
```

```
    /*
```

```
     * this function get total amount of the cart by multiply quantity by the sell price
```

```
     * @ return the total cost of the cart
```

```
     *
```

```
    */
```

```
        public double getTotal(){
            total=0;
            for(Product p1: cart){
```

```

        total+=p1.getSellPrice()*p1.getQuantity();
    }
    return total;
}
/*
 * update the product quantity in the cart
 * @param Product p is a product in the cart
 * @param quantity is the quantity of the product in the cart
 */
public void updateQuantity(Product p,int quantity){
    p.setQuantity(quantity);

}
/*
 * clear the cart
 * @param cart is a ArrayList of Product
 */
public void clearCart(ArrayList<Product> cart){
    cart.clear();
}

```

```

    private double total;
    private ArrayList<Product> cart=new ArrayList<Product>();

}

```

Seller.java
package project;

```

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JTable;

```

```
import javax.swing.table.DefaultTableModel;
import javax.swing.JScrollPane;
```

```
import java.awt.Button;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.text.DecimalFormat;
import java.util.ArrayList;
```

```
import javax.swing.JTextField;
import javax.swing.JButton;
```

```
public class CartFrame extends JFrame {
    ;
    private JPanel contentPane;
    private JTable table;
    private JTextField textField;
    double total;
    DecimalFormat dc = new DecimalFormat("0.00");
    Inventory inv=new Inventory();
    int qty;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    CartFrame frame = new CartFrame();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
}
```

```

public CartFrame() {
    getContentPane().setLayout(null);
    setBounds(100, 100, 574, 397);
    JLabel lblEnterCreditCard_1 = new JLabel("Enter Credit Card");
    lblEnterCreditCard_1.setBounds(10, 284, 187, 26);
    getContentPane().add(lblEnterCreditCard_1);

    textField = new JTextField();
    textField.setBounds(10, 308, 227, 26);
    getContentPane().add(textField);
    textField.setColumns(10);

    JButton btnSubmit = new JButton("Submit");
    btnSubmit.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0) {
        }
    });
    btnSubmit.setBounds(418, 310, 89, 23);
    getContentPane().add(btnSubmit);

    JLabel lblNewLabel = new JLabel("Total");
    lblNewLabel.setBounds(10, 344, 153, 14);
    getContentPane().add(lblNewLabel);
}

```

```

}
public CartFrame(final Customer c){

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 574, 397);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);
    JScrollPane scrollPane = new JScrollPane();
    scrollPane.setBounds(59, 55, 410, 233);
    contentPane.add(scrollPane);
    table = new JTable();
    scrollPane.setViewportView(table);
    table.setModel(new DefaultTableModel(
        new Object[][] {
        },
    );
}

```

```

        new String[] {
            "ID", "Name", "Sell Price", "Quantity"
        }
    ));
    addtoRow(c.getCart());
    Button backButton = new Button("Back");
    backButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            new ShoppingFrame(c).setVisible(true);
            dispose();
        }
    });
    backButton.setBounds(478, 25, 70, 22);
    contentPane.add(backButton);
    for(int i=0;i<c.getCart().getProducts().size();i++){
        System.out.println(c.getCart().getProducts().get(i).getName());
    }
    JButton btnSubmit = new JButton("Submit");
    btnSubmit.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if(textField.getText().length()!=16){
                JOptionPane.showMessageDialog(rootPane, "Please
enter a 16 digits Credit Number");
            }
            else{
                JOptionPane.showMessageDialog(rootPane, "Your order
has been submitted!");
                for(int k=0;k<c.getCart().getProducts().size();k++){
                    inv.LoadInventory();
                    qty=c.getCart().getProducts().get(k).getQuantity();

System.out.println(c.getCart().getProducts().get(k).getQuantity());

                inv.newQuantity(c.getCart().getProducts().get(k),qty);
                System.out.println(c.getCart().getTotal());
                inv.saveInvoice(c.getCart().getTotal());
                c.getCart().clearCart(c.getCart().getProducts());
            }
        }
    }
}

```

```

        }
    });
    btnSubmit.setBounds(418, 310, 89, 23);
    getContentPane().add(btnSubmit);
    JLabel lblEnterCreditCard_1 = new JLabel("Enter Credit Card");
    lblEnterCreditCard_1.setBounds(10, 284, 187, 26);
    getContentPane().add(lblEnterCreditCard_1);
    textField = new JTextField();
    textField.setBounds(10, 308, 227, 26);
    getContentPane().add(textField);
    textField.setColumns(10);
    total=c.getCart().getTotal();
    String formattedText = dc.format(total);
    JLabel lblNewLabel = new JLabel("Total: $ "+ formattedText);

    lblNewLabel.setBounds(10, 344, 153, 14);
    getContentPane().add(lblNewLabel);
}

public void addtoRow(Cart c){
    DefaultTableModel model= (DefaultTableModel)table.getModel();
    Object rowData[]=new Object[5];
    for(int i=0;i<c.getProducts().size();i++){
        rowData[0]=c.getProducts().get(i).getId();
        rowData[1]=c.getProducts().get(i).getName();
        rowData[2]=c.getProducts().get(i).getSellPrice();
        rowData[3]=c.getProducts().get(i).getQuantity();
        model.addRow(rowData);
    }
}
}

```

```

}
CartTest.java
package project;

```

```

import static org.junit.Assert.*;

```

```

import org.junit.Assert;
import org.junit.Test;

```



```

public class CartTest {

    @Test
    public void test() {
        Customer c= new Customer("Test","Test","Customer");
        Cart cart=new Cart();
        Cart test=new Cart();

        cart.addToCart(new Product(1,"basketball","description",4.99,2.01,3));
        test.addToCart(new Product(1,"basketball","description",4.99,2.01,6));
        c.getCart().updateQuantity(cart.getProducts().get(0), 6);
        int w=cart.getProducts().get(0).getQuantity();
        int k=cart.getProducts().get(0).getQuantity();

        Assert.assertEquals(w,k);
    }

}

```

Customer.java
package project;

```
import java.io.Serializable;
```

```

public class Customer extends User implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 3766094891760049502L;
    /**
     *
     */
    /**
     * Customer constructor
     */
    public Customer(String username, String password, String type) {
        super(username, password, type);
    }
}

```

```

        this.shoppingcart=new Cart();

    }
    /*
    * get the customer cart
    * @return the customer shopping cart
    */
    public Cart getCart(){

        return shoppingcart;
    }

    private Cart shoppingcart;

}

```

Database.java
package project;

```

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

```

```

public class Database {

    private static Database project= new Database();

    private Database() {
        LoadDatabase();
        outputDatabase();
    }
}

```

```
}
```

```
public static Database getInstance(){  
    return project;  
}
```

```
@SuppressWarnings("unchecked")
```

```
/*
```

```
 * load the user database by reading the registry.ser file
```

```
*/
```

```
public void LoadDatabase(){
```

```
    try {
```

```
        //InputStream fileIn = Database.class.getResourceAsStream("/registry.ser");
```

```
        FileInputStream fileIn = new FileInputStream("registry.ser");
```

```
        ObjectInputStream out= new ObjectInputStream(fileIn);
```

```
        this.users = (ArrayList<User>)out.readObject();
```

```
        out.close();
```

```
        fileIn.close();
```

```
    }catch(IOException i) {
```

```
        i.printStackTrace();
```

```
    } catch (ClassNotFoundException e) {
```

```
        e.printStackTrace();
```

```
    }
```

```
}
```

```
/*
```

```
 * add the user to the registry
```

```
 * @param username is the username enter from the user
```

```
 * @param password is the password entered
```

```
 * @param type is the type of customer selected
```

```
*/
```

```
public void addUsers(String username,String password, String type){
```

```
    if(type.equals("Seller")){
```

```
        this.users.add(new Seller(username,password,"Seller"));
```

```
    }
```

```
    else if(type.equals("Customer")){
```

```
        this.users.add(new Customer(username,password,"Customer"));
```

```
    }
```

```
    try {
```

```

        new FileOutputStream("registry.ser").close();
    FileOutputStream fileOut =
    new FileOutputStream("registry.ser",true);
    ObjectOutputStream out = new ObjectOutputStream(fileOut);
    out.writeObject(users);
    out.close();
    fileOut.close();

    System.out.printf("Serialized data is saved in registry.ser");
    outputDatabase();
}catch(IOException i) {
    i.printStackTrace();
}

}
/*
 * verify that the username and password are equal
 * @param username is the username that is entered
 * @param password is the password that is entered
 * @return the username and password
 *
 */
public User verifyUserandPass(String username,String password){
    for(User u : users){
        if(u.getUsername().equals(username) && u.getPassword().equals(password)){
            return u;
        }
    }
    return null;
}

public void outputDatabase(){
    for(User u : users){
        System.out.println(u.getUsername() + ", " + u.getPassword());
    }
}
/*
 * check if the user is already taken
 * @param u is the username entered
 * @return the true or false
 */
public Boolean checkUser(String u){

```

```

        LoadDatabase();
        for(User u1:users){
            if(u1.getUsername().equals(u)){
                return false;
            }
        }
        return true;
    }

    private ArrayList<User> users=new ArrayList<User>();
}

```

DiscountedProduct.java

package project;

public class DiscountedProduct {

```

    private static final long serialVersionUID = 1L;
    public DiscountedProduct(double discount ,Product products){
        this.setProducts(products);
        this.discount=discount;
    }

```

```

    }

```

```

    public int getId() {
        return id;
    }

```

```

    public void setId(int id) {
        this.id = id;
    }

```

```

    public String getName() {
        return name;
    }

```

```

    public void setName(String name) {
        this.name = name + "discounted";
    }

```

```

    public String getDescription() {
        return description;
    }

```

```

    public void setDescription(String description) {

```

```

        this.description = description;
    }
    public double getSellPrice() {
        return SellPrice*discount;
    }
    public void setSellPrice(float sellPrice) {
        SellPrice = sellPrice;
    }
    public double getInvoicePrice() {
        return InvoicePrice;
    }
    public void setInvoicePrice(float invoicePrice) {
        InvoicePrice = invoicePrice;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    public double getDiscount() {
        return discount;
    }

    public void setDiscount(double discount) {
        this.discount = discount;
    }
    public Product getProducts() {
        return products;
    }

    public void setProducts(Product products) {
        this.products = products;
    }
    private int id;
    private String name;
    private String description;
    private double SellPrice;
    private double InvoicePrice;
    private int quantity;
    private double discount;
    private Product products;

```

```
}  
Inventory.java  
package project;
```

```
import java.io.EOFException;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.Iterator;
```

```
public class Inventory implements Serializable {
```

```
    /**  
     *  
     */
```

```
    private static final long serialVersionUID = 1L;
```

```
    /**  
     * add a product in the inventory  
     * @param Product p is a product  
     * @postcondition remove the product  
     *  
     */
```

```
    public void addProduct(Product p) {  
        products.add(p);  
    }
```

```
    /**  
     * remove a product in the inventory  
     * @precondition the product size is greater than 0  
     * @param Product p is a product  
     * @postcondition remove the product  
     *  
     */
```

```

public void removeProduct(Product p){

    products.remove(p);
}
/*
 * update a product Quantity in the inventory
 * @precondition the cart size is greater than 0
 * @param Product p is a product
 * @param quantity is the product quantity
 * @postcondition update the product quantity
 */

public void updateQuantity(Product p,int quantity){
    p.setQuantity(quantity);

}
/*
 * update a product Quantity in the inventory when a customer buy a product
 * @precondition the cart size is greater than 0
 * @param Product p is a product
 * @param quantity is the product quantity
 * @postcondition update the product quantity
 */

public void newQuantity(Product p,int quantity){
    for(Product p1: products){
        if(p1.getName().equals(p.getName())){

            System.out.println("p1 "+p1.getQuantity()+ " p"+ quantity );
            quantity=p1.getQuantity()-quantity;

            p1.setQuantity(quantity);
            saveInventory(products);

        }
    }
    for(Product p2: products){
        System.out.println(p2.getQuantity()+p2.getName());
    }
}

```



```

    }
    /*
    * get Products in the inventory
    * return the products in the Array list
    *
    */
    public ArrayList<Product> getProducts(){
        return products;
    }
    /*
    * get the total cost in the inventory
    * @precondition the cart size is greater than 0
    * return the cost of the inventory
    */

    public double getCost(){
        for(Product p1: products){
            cost+=p1.getInvoicePrice()*p1.getQuantity();
        }
        return cost;
    }
    /*
    * this function load the total revenue of product sold
    */

    public Iterator<Product> getProducts(){
        return new Iterator<Product>(){
            public boolean hasNext(){
                return current < products.size();
            }
            public Product next(){
                return products.get(current++);//Returns next item and increments current index
            }
            public void remove(){
                throw new UnsupportedOperationException();
            }
            private int current = 0;
        };
    }

    public void totalProfitInventory(){

```

```

        try {
            FileInputStream fileIn = new
FileInputStream("revenue.ser");
            //InputStream fileIn =
Inventory.class.getResourceAsStream("revenue.ser");

            ObjectInputStream in = new ObjectInputStream(fileIn);

            this.revenue= (Double) in.readObject();
            in.close();
            fileIn.close();
        }catch(IOException i) {
            i.printStackTrace();
            return;
        }catch(ClassNotFoundException c) {
            c.printStackTrace();
            return;
        }
    }

}

/*
 *
 * Load the Inventory
 */
@SuppressWarnings("unchecked")
public void LoadInventory(){
    try {
        FileInputStream fileIn = new FileInputStream("inventory.ser");
        //InputStream fileIn = Inventory.class.getResourceAsStream("inventory.ser");
        ObjectInputStream out= new ObjectInputStream(fileIn);
        this.products = (ArrayList<Product>)out.readObject();
        out.close();
        fileIn.close();

    }catch(IOException i) {
        i.printStackTrace();

    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
}
}

```

```

public void saveInventory(ArrayList<Product> products){
    try {

        new FileOutputStream("inventory.ser").close();
        FileOutputStream fileOut =
        new FileOutputStream("inventory.ser",true);
        ObjectOutputStream out = new ObjectOutputStream(fileOut);
        out.writeObject(products);
        out.close();
        fileOut.close();

    }catch(IOException i) {
        i.printStackTrace();
    }
    outputDatabase();
}
/*
 * save the customer purchases
 * @param cost is the cost of the products in the cart
 */
public void saveInvoice(double cost){
    totalProfitInventory();
    total=getRevenue();
    total=total+cost;
    try {

        new FileOutputStream("revenue.ser").close();
        FileOutputStream fileOut =
        new FileOutputStream("revenue.ser",true);
        ObjectOutputStream out = new ObjectOutputStream(fileOut);
        out.writeObject(total);
        out.close();
        fileOut.close();
        System.out.printf("cost.ser");

    }catch(IOException i) {
        i.printStackTrace();
    }
}

```

```

        public void outputDatabase(){
            for(int i=0;i<products.size();i++){
                System.out.println(products.get(i).getName());

            }

        }
        /*
        * get the total Revenue
        * @return the total revenue
        */

        public double getRevenue(){
            return revenue;
        }

        public ArrayList<Product> products=new ArrayList<Product>();
        private double cost;
        private double revenue;
        private double total;
    }
InventoryFrame.java
package project;

import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.net.URISyntaxException;
import java.text.DecimalFormat;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;

```

```

public class InventoryFrame extends JFrame {

    private JPanel contentPane;
    private JTable table;
    Inventory inv= new Inventory();
    Seller sell= new Seller("a","b","Seller");
    Boolean match;
    private JTextField textField;
    private JTextField textField_1;
    private JTextField textField_2;
    private JTextField textField_3;
    private JTextField textField_4;
    private JTextField textField_5;
    double cost;
    double profit;
    double revenue;
    DecimalFormat dc = new DecimalFormat("0.00");
    private JTextField textField_6;


    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    InventoryFrame frame = new InventoryFrame();

                    frame.setVisible(true);

                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public InventoryFrame() {

```

```

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(100, 100, 670, 455);
contentPane =
    new JPanel();
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(contentPane);
contentPane.setLayout(null);

JScrollPane scrollPane = new JScrollPane();
scrollPane.setBounds(85, 119, 423, 254);
contentPane.add(scrollPane);

table = new JTable();
table.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        DefaultTableModel model=(DefaultTableModel) table.getModel();
        textField.setText(model.getValueAt(table.getSelectedRow(),
0).toString());
        textField_1.setText(model.getValueAt(table.getSelectedRow(),
1).toString());
        textField_2.setText(model.getValueAt(table.getSelectedRow(),
2).toString());
        textField_3.setText(model.getValueAt(table.getSelectedRow(),
3).toString());
        textField_4.setText(model.getValueAt(table.getSelectedRow(),
4).toString());

    }
});
inv=sell.getInventory();

table.setModel(new DefaultTableModel(
    new Object[][] {
    },
    new String[] {
        "ID", "Name", "Invoice Price", "Sell Price", "Quantity"
    }
));
addtoRow();

scrollPane.setViewportViewView(table);

```

```

JButton btnAdd = new JButton("Add");
btnAdd.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        DefaultTableModel model=(DefaultTableModel) table.getModel();

        if(!textField_1.getText().trim().equals("")||!textField_1.getText().trim().equals("")||!textField_2.getText().trim().equals("")||!textField_3.getText().trim().equals("")||!textField_4.getText().trim().equals("")||!textField_5.getText().trim().equals("")){

            String text=textField.getText();
            int id=Integer.parseInt(text);
            String name=textField_1.getText();
            double invoice =
Double.parseDouble(textField_2.getText());
            double sell = Double.parseDouble(textField_3.getText());
            String text1=textField.getText();
            int qty=Integer.parseInt(text1);
            String desc=textField_5.getText();
            match=false;
            for(int i=0;i<inv.getProducts().size();i++){
                if(id == inv.getProducts().get(i).getId()){

JOptionPane.showMessageDialog(rootPane, "This id already exist");

                    match=true;

                }

            }
            if(match!=true){
                model.addRow(new
Object[]{textField.getText(),textField_1.getText(),textField_2.getText(),textField_3.getText(),text
Field_4.getText()});

                inv.addProduct(new
Product(id,name,desc,sell,invoice,qty));

                inv.saveInventory(inv.getProducts());

            }
            for(int i=0;i<inv.getProducts().size();i++){
                System.out.println(inv.getProducts().get(i).getId());
                System.out.println(inv.getProducts().get(i).getName());

            }

        }else{

```

```
        JOptionPane.showMessageDialog(rootPane, "Please fill  
out every field");  
    }  
  
    }  
});  
btnAdd.setBounds(531, 122, 89, 23);  
contentPane.add(btnAdd);  
  
JLabel lblNewLabel = new JLabel("ID");  
lblNewLabel.setBounds(23, 28, 84, 14);  
contentPane.add(lblNewLabel);  
  
textField = new JTextField();  
textField.setBounds(51, 25, 130, 20);  
contentPane.add(textField);  
textField.setColumns(10);  
  
JLabel lblNewLabel_1 = new JLabel("Name");  
lblNewLabel_1.setBounds(10, 53, 56, 14);  
contentPane.add(lblNewLabel_1);  
  
textField_1 = new JTextField();  
textField_1.setBounds(49, 53, 132, 20);  
contentPane.add(textField_1);  
textField_1.setColumns(10);  
  
JLabel lblInvoicePrice = new JLabel("Invoice Price");  
lblInvoicePrice.setBounds(185, 28, 77, 14);  
contentPane.add(lblInvoicePrice);  
  
textField_2 = new JTextField();  
textField_2.setBounds(262, 25, 71, 20);  
contentPane.add(textField_2);  
textField_2.setColumns(10);  
  
JLabel lblSellPrice = new JLabel("Sell Price");  
lblSellPrice.setBounds(191, 53, 71, 14);  
contentPane.add(lblSellPrice);  
  
textField_3 = new JTextField();  
textField_3.setBounds(262, 50, 71, 20);
```



```

contentPane.add(textField_3);
textField_3.setColumns(10);

JLabel lblQuantity = new JLabel("Quantity");
lblQuantity.setBounds(368, 28, 67, 14);
contentPane.add(lblQuantity);

textField_4 = new JTextField();
textField_4.setBounds(416, 25, 32, 20);
contentPane.add(textField_4);
textField_4.setColumns(10);

JButton btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        DefaultTableModel model= (DefaultTableModel)table.getModel();
        if(table.getSelectedRow()!=-1){
            if(table.getRowCount()==0){
                JOptionPane.showMessageDialog(rootPane,
"Table is empty");
            }
            else{
                JOptionPane.showMessageDialog(rootPane, "You
must select a product");
            }
        }
        else{

model.setValueAt(textField_4.getText(),table.getSelectedRow(), 4);
        String text1=textField_4.getText();
        int qty=Integer.parseInt(text1);
        String text=textField.getText();
        int id=Integer.parseInt(text);
        for(int i=0;i<inv.getProducts().size();i++){
            if(id==inv.getProducts().get(i).getId()){
                inv.updateQuantity(inv.getProducts().get(i),
qty);
                inv.saveInventory(inv.getProducts());
            }
        }
    }
}

```

```

        //for(int i=0;i<inv.getProducts().size();i++){

//System.out.println(inv.getProducts().get(i).getName());

//System.out.println(inv.getProducts().get(i).getQuantity());

        //}

    }

});
btnUpdate.setBounds(531, 156, 89, 23);
contentPane.add(btnUpdate);

JLabel lblDescription = new JLabel("Description");
lblDescription.setBounds(343, 53, 92, 14);
contentPane.add(lblDescription);

textField_5 = new JTextField();
textField_5.setBounds(416, 50, 190, 38);
contentPane.add(textField_5);
textField_5.setColumns(10);

JButton btnRemove = new JButton("Remove");
btnRemove.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        DefaultTableModel model= (DefaultTableModel)table.getModel();
        if(table.getSelectedRow()!=-1){
            if(table.getRowCount()==0){
                JOptionPane.showMessageDialog(rootPane,
"Table is empty");
            }
            else{
                JOptionPane.showMessageDialog(rootPane, "You
must select a product");
            }
        }
    }
});
model.removeRow(table.getSelectedRow());
String text=textField.getText();
int id=Integer.parseInt(text);

```

```

        System.out.println(id);
        for(int i=0;i<inv.getProducts().size();i++){
            if(id==inv.getProducts().get(i).getId()){
                inv.removeProduct(inv.getProducts().get(i));
                inv.saveInventory(inv.getProducts());
            }
        }
        for(int i=0;i<inv.getProducts().size();i++){
            System.out.println(inv.getProducts().get(i).getId());

System.out.println(inv.getProducts().get(i).getName());

        }

    }
});
btnRemove.setBounds(531, 190, 89, 23);
contentPane.add(btnRemove);

JButton btnLogout = new JButton("Logout");
btnLogout.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        new LoginFrame().setVisible(true);
        dispose();
    }
});
btnLogout.setBounds(555, 11, 89, 23);
contentPane.add(btnLogout);

cost=inv.getCost();
String formattedText = dc.format(cost);

JLabel lblTotalCost = new JLabel("Total Cost " + "$" + formattedText);
lblTotalCost.setBounds(20, 391, 114, 14);
contentPane.add(lblTotalCost);
inv.totalProfitInventory();
revenue=inv.getRevenue();

```

```

        String formattedText1 = dc.format(revenue);
        JLabel lblTotalProfit = new JLabel("Total Revenue " + "$" +formattedText1);
        lblTotalProfit.setBounds(193, 391, 140, 14);
        contentPane.add(lblTotalProfit);
        profit=revenue-cost;
        String formattedText2 = dc.format(profit);

        JLabel lblTotalRevenue = new JLabel("Total Profit "+"$"+formattedText2);
        lblTotalRevenue.setBounds(368, 391, 162, 14);
        contentPane.add(lblTotalRevenue);

    }

    public void addtoRow(){

        inv.LoadInventory();

        DefaultTableModel model= (DefaultTableModel)table.getModel();
        Object rowData[]=new Object[5];
        for(int i=0;i<inv.getProducts().size();i++){

            rowData[0]=inv.getProducts().get(i).getId();
            rowData[1]=inv.getProducts().get(i).getName();
            rowData[2]=inv.getProducts().get(i).getInvoicePrice();
            rowData[3]=inv.getProducts().get(i).getSellPrice();
            rowData[4]=inv.getProducts().get(i).getQuantity();
            model.addRow(rowData);

        }

    }

}

```

InventoryTest.java

package project;

import static org.junit.Assert.*;

import org.junit.Assert;

import org.junit.Test;

```

public class InventoryTest {

    @Test
    public void testUpdateQuantity() {
        Seller s= new Seller("Test","Test","Seller");
        Inventory inventory=new Inventory();
        Inventory inventoryTest=new Inventory();

        inventory.addProduct(new Product(1,"basketball","description",4.99,2.01,3));
        inventoryTest.addProduct(new Product(1,"basketball","description",4.99,2.01,6));
        s.getInventory().updateQuantity(inventory.getProducts().get(0), 6);
        int w=inventory.getProducts().get(0).getQuantity();
        int k=inventoryTest.getProducts().get(0).getQuantity();

        Assert.assertEquals(w,k);

    }

    @Test
    public void testRemoveQuantity(){

        Inventory inventoryTest=new Inventory();

        inventoryTest.addProduct(new Product(1,"basketball","description",4.99,2.01,6));

        inventoryTest.removeProduct(inventoryTest.getProducts().get(0));

        int k=inventoryTest.getProducts().size();

        System.out.println(k);
        Assert.assertEquals(0,k);

    }

}

```

```
LoginFrame.java
package project;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.plaf.ComboBoxUI;
import javax.swing.JLabel;
import javax.swing.SwingConstants;

import java.awt.FlowLayout;

import javax.swing.JComboBox;
import javax.swing.BoxLayout;
import javax.swing.GroupLayout;
import javax.swing.JOptionPane;
import javax.swing.GroupLayout.Alignment;

import java.awt.CardLayout;

import javax.swing.JTextField;
import javax.swing.SpringLayout;

import java.awt.GridLayout;

import javax.swing.JPasswordField;
import javax.swing.JButton;
import javax.swing.DefaultComboBoxModel;

import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class LoginFrame extends JFrame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
```

```

/**
 *
 */

private JPanel contentPane;
private JTextField userTextfield;
private JPasswordField passwordTextfield;
private RegisterFrame h=new RegisterFrame();
public static User login=null;
/**
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                LoginFrame frame = new LoginFrame();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

/**
 * Create the frame.
 */
public LoginFrame() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    userTextfield = new JTextField();
    userTextfield.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            userTextfield.setText("");
        }
    });
}

```

```
userTextfield.setBounds(117, 73, 194, 31);
contentPane.add(userTextfield);
userTextfield.setColumns(10);
```

```
JLabel lblLogin = new JLabel("Login");
lblLogin.setBounds(117, 23, 146, 14);
contentPane.add(lblLogin);
```

```
passwordTextfield = new JPasswordField();
passwordTextfield.setColumns(10);
passwordTextfield.setBounds(117, 140, 194, 31);
contentPane.add(passwordTextfield);
```

```
JButton loginButton = new JButton("Login");
loginButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String u= userTextfield.getText();
        String p = new String(passwordTextfield.getPassword());
```

```
        Database db= Database.getInstance();
        login=db.verifyUserandPass(u,p);
```

```
        if(login!=null && login instanceof Customer){
            JOptionPane.showMessageDialog(rootPane, "Customer");
            System.out.println(u);
            System.out.println(p);
            System.out.println(login.getTypeOfUser());
            Customer c= new Customer(u,p,"Customer");
            dispose();
            new ShoppingFrame(c).setVisible(true);
        }
```

```
        else if(login!=null && login instanceof Seller){
            JOptionPane.showMessageDialog(rootPane, "Seller");
            System.out.println(u);
            System.out.println(p);
            System.out.println(login.getTypeOfUser());
            Seller s= new Seller(u,p,"Seller");
            dispose();
            new InventoryFrame().setVisible(true);
```

```
        }
        else{
```



```

JOptionPane.showMessageDialog(rootPane, "Username
or Password doesn't match");
    }
}

});
loginButton.setBounds(274, 214, 89, 23);
contentPane.add(loginButton);

JButton registerButton = new JButton("Register");
registerButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        dispose();
        h.setVisible(true);
    }
});
registerButton.setBounds(75, 214, 89, 23);
contentPane.add(registerButton);

JLabel lblNewLabel = new JLabel("Username");
lblNewLabel.setBounds(117, 48, 194, 14);
contentPane.add(lblNewLabel);

JLabel lblNewLabel_1 = new JLabel("Password");
lblNewLabel_1.setBounds(117, 115, 194, 14);
contentPane.add(lblNewLabel_1);

}
}

```

Product.java
package project;

import java.io.Serializable;

public class Product implements Serializable {

/**

*

*/

private static final long serialVersionUID = 1L;

```
    public Product(int id,String name,String description,double SellPrice,double
InvoicePrice,int quantity ){
        this.id=id;
        this.name=name;
        this.description=description;
        this.SellPrice=SellPrice;
        this.InvoicePrice=InvoicePrice;
        this.quantity=quantity;

    }

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public double getSellPrice() {
        return SellPrice;
    }
    public void setSellPrice(float sellPrice) {
        SellPrice = sellPrice;
    }
    public double getInvoicePrice() {
        return InvoicePrice;
    }
    public void setInvoicePrice(float invoicePrice) {
        InvoicePrice = invoicePrice;
    }
    public int getQuantity() {
        return quantity;
    }
}
```

```

    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    private int id;
    private String name;
    private String description;
    private double SellPrice;
    private double InvoicePrice;
    private int quantity;
}

```

RegisterFrame

package project;

import java.awt.BorderLayout;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import javax.swing.JPasswordField;

import javax.swing.JComboBox;

import javax.swing.DefaultComboBoxModel;

import java.awt.Button;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

public class RegisterFrame extends JFrame {

private JPanel contentPane;

private JTextField rTextfield;

private JPasswordField jpasswordField;

private JPasswordField jcpasswordField;

private JComboBox comboBox;

private Button backButton;

```

/**
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                RegisterFrame frame = new RegisterFrame();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

```

```

/**
 * Create the frame.
 */
public RegisterFrame() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JLabel registerLabel = new JLabel("Register");
    registerLabel.setBounds(186, 11, 138, 14);
    contentPane.add(registerLabel);

    rTextfield = new JTextField();

    rTextfield.setBounds(125, 53, 199, 31);
    contentPane.add(rTextfield);
    rTextfield.setColumns(10);

    JLabel lblUsername = new JLabel("Username");
    lblUsername.setBounds(125, 36, 75, 14);
    contentPane.add(lblUsername);
}

```

```

jpasswordField = new JPasswordField();
jpasswordField.setBounds(125, 101, 199, 31);
contentPane.add(jpasswordField);

jcpasswordField = new JPasswordField();
jcpasswordField.setBounds(125, 153, 199, 31);
contentPane.add(jcpasswordField);

JLabel lblNewLabel = new JLabel("Password");
lblNewLabel.setBounds(125, 87, 199, 14);
contentPane.add(lblNewLabel);

JLabel lblConfirmPassword = new JLabel("Confirm Password");
lblConfirmPassword.setBounds(125, 139, 199, 14);
contentPane.add(lblConfirmPassword);

comboBox = new JComboBox();
comboBox.setModel(new DefaultComboBoxModel(new String[] {"Customer",
"Seller"}));

comboBox.setBounds(124, 190, 200, 31);
contentPane.add(comboBox);

Button rButton = new Button("Register");
rButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String u= rTextField.getText();
        String p = new String(jpasswordField.getPassword());
        String type= comboBox.getSelectedItem().toString();
        String cp = new String(jcpasswordField.getPassword());

        if (p.equals(cp))
        {
            Database db= Database.getInstance();
            if(db.checkUser(u)==true ){
                db.addUsers(u, p, type);
                new LoginFrame().setVisible(true);
                dispose();
            }
            else{
                JOptionPane.showMessageDialog(rootPane, "Username
exists");
            }
        }
    }
});

```

```

        }
        else {
            JOptionPane.showMessageDialog(rootPane, "Password doesn't
match");
        }
    }

});
rButton.setBounds(125, 229, 70, 22);
contentPane.add(rButton);

backButton = new Button("Back");
backButton.setActionCommand("backButton");
backButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        dispose();
        new LoginFrame().setVisible(true);
    }
});
backButton.setBounds(10, 11, 70, 22);
contentPane.add(backButton);
}
}

```

Seller.java
package project;

```

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.Serializable;

public class Seller extends User implements Serializable {

    /**
     *

```

```

*/
private static final long serialVersionUID = 1L;
/*
 * create a Seller constructor
 *
 */
public Seller(String username, String password, String type) {
    super(username, password, type);
    this.inventory=new Inventory();
    this.inventory.addProduct(new Product(1,"basketball","description",4.99,2.01,3));
    this.inventory.addProduct(new Product(2,"golf","description",5.99,2.00,3));
    this.inventory.addProduct(new Product(3,"soccer ball","description",5.99,2.00,3));
    this.inventory.addProduct(new Product(4,"baseball","description",2.99,2.00,3));
    this.inventory.addProduct(new Product(5,"football","description",6.99,2.00,3));
    this.inventory.addProduct(new Product(6,"golf club","description",7.99,2.00,3));

}
/*
 * get the Inventory
 * @return the inventory
 */
public Inventory getInventory(){
    return this.inventory;

}
/*
 * save the inventory
 * @param inventory is the Inventory
 */
public void saveInventory(Inventory inventory){
    try {
        new FileOutputStream("inventory.ser").close();
        FileOutputStream fileOut =
        new FileOutputStream("inventory.ser",true);
        ObjectOutputStream out = new ObjectOutputStream(fileOut);
        out.writeObject(inventory);
        out.close();
        fileOut.close();
        System.out.printf("inventory.ser");

    }catch(IOException i) {
        i.printStackTrace();
    }
}

```

```

    }
    public void outputDatabase(){
        for(int i=0;i<inventory.getProducts().size();i++){
            System.out.println(inventory.getProducts().get(i).getName());

        }

    }

    private Inventory inventory;

}

```

ShoppingCart.java

package project;

```

import java.awt.EventQueue;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;

```

```

public class ShoppingCart {
    void check() {

```

```

    }

```

```

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {

```

```

                    LoginFrame frame = new LoginFrame();
                    frame.setVisible(true);

```

```

                } catch (Exception e) {
                    e.printStackTrace();
                }

```

```

            }
        });

```

```

}

```



```

}
ShoppingFrame.java
package project;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import javax.swing.JScrollPane;
import javax.swing.JButton;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Collections;

import javax.swing.JLabel;
import javax.swing.JTextField;

public class ShoppingFrame extends JFrame {

    private JPanel contentPane;

    Inventory inv =new Inventory();
    private double total;

    private JTable table;
    private JTextField textField;
    DecimalFormat dc = new DecimalFormat("0.00");
    boolean match;
    int i;

```

```

/**
 * Launch the application.
 */
public static void main(String[] args) {

    EventQueue.invokeLater(new Runnable() {

        public void run() {
            try {
                ShoppingFrame frame = new ShoppingFrame();

                frame.setVisible(true);

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

public ShoppingFrame(){
    getContentPane().setLayout(null);

    JButton btnNewButton_2 = new JButton("New button");
    btnNewButton_2.setBounds(488, 27, 105, 28);
    getContentPane().add(btnNewButton_2);

}

/**
 * Create the frame.
 */
public ShoppingFrame(final Customer c) {

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 640, 455);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JScrollPane scrollPane = new JScrollPane();
    scrollPane.addMouseListener(new MouseAdapter() {
        @Override

```

```

        public void mouseClicked(MouseEvent e) {
            DefaultTableModel model=(DefaultTableModel) table.getModel();
            model.getValueAt(table.getSelectedRow(), 0);
            model.getValueAt(table.getSelectedRow(), 1);
            model.getValueAt(table.getSelectedRow(), 2);
            model.getValueAt(table.getSelectedRow(), 3);
            model.getValueAt(table.getSelectedRow(), 4);
        }
    });
    scrollPane.setBounds(64, 72, 440, 258);
    contentPane.add(scrollPane);

    table = new JTable();
    scrollPane.setViewportViewView(table);
    table.setModel(new DefaultTableModel(
        new Object[][] {
        },
        new String[] {
            "ID", "Name", "Sell Price", "In Stock","Quantity"
        }
    ));

    JButton btnNewButton = new JButton("Add To Cart");
    btnNewButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

            DefaultTableModel model=(DefaultTableModel) table.getModel();
            if (table.isEditing())
                table.getCellEditor().stopCellEditing();
            if(table.getSelectedRow()!=-1){
                if(table.getRowCount()==0){
                    JOptionPane.showMessageDialog(rootPane,
"Table is empty");
                }
                else{
                    JOptionPane.showMessageDialog(rootPane, "You
must select a product");
                }
            }
        }
    });
    int qty;
    int stock;

```

```

qty=Integer.parseInt(model.getValueAt(table.getSelectedRow(), 4).toString());

stock=Integer.parseInt(model.getValueAt(table.getSelectedRow(), 3).toString());
    for(i=0;i<inv.getProducts().size();++i){

if(inv.getProducts().get(i).getId()==(Integer)(model.getValueAt(table.getSelectedRow(), 0))){

                                if(c.getCart().getProducts().size()==0 &&
qty!=0 && qty<=stock){

                                c.getCart().addToCart(new
Product(inv.getProducts().get(i).getId(),inv.getProducts().get(i).getName(),inv.getProducts().get(i)
).getDescription(),inv.getProducts().get(i).getSellPrice(),inv.getProducts().get(i).getInvoicePrice()
.qty));

                                }
                                else if(qty>stock){

JOptionPane.showMessageDialog(rootPane, "You have exceed the quantity limit");

                                }
                                else{

for(i=0;i<c.getCart().getProducts().size();i++){

                                match=false;

if(c.getCart().getProducts().get(i).getId()==(Integer)(model.getValueAt(table.getSelectedRow(),
0))&& qty<=stock){

                                match=true;

if(qty==c.getCart().getProducts().get(i).getQuantity()){

JOptionPane.showMessageDialog(rootPane, "Product quantity already exists");
                                }else{

JOptionPane.showMessageDialog(rootPane, "Quantity have been updated");

c.getCart().updateQuantity(c.getCart().getProducts().get(i),qty);

model.setValueAt(qty,table.getSelectedRow(), 4 );

                                }
                                }
}
}

```

```

else if(match==false){

for(i=0;i<inv.getProducts().size();++i){

if(inv.getProducts().get(i).getId()==(Integer)(model.getValueAt(table.getSelectedRow(), 0))){

c.getCart().addToCart(new
Product(inv.getProducts().get(i).getId(),inv.getProducts().get(i).getName(),inv.getProducts().get(i)
).getDescription(),inv.getProducts().get(i).getSellPrice(),inv.getProducts().get(i).getInvoicePrice()
,qty));

}
}
else{

JOptionPane.showMessageDialog(rootPane, "Product already exist in the cart");

}

}

}

}

total=c.getCart().getTotal();
String formattedText = dc.format(total);
System.out.println(total);
textField.setText("$" + formattedText);

}

```

```

    }
});
JButton btnNewButton_2 = new JButton("Logout");
btnNewButton_2.setBounds(514, 27, 101, 28);
getContentPane().add(btnNewButton_2);
btnNewButton.setBounds(514, 75, 101, 23);
contentPane.add(btnNewButton);
btnNewButton_2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        new LoginFrame().setVisible(true);
        dispose();
    }
});

JButton btnNewButton_1 = new JButton("View Cart");
btnNewButton_1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {

        new CartFrame(c).setVisible(true);

        dispose();

    }
});
btnNewButton_1.setBounds(514, 127, 101, 23);
contentPane.add(btnNewButton_1);

JLabel lblTitle = new JLabel("total");
lblTitle.setBounds(29, 367, 46, 14);
contentPane.add(lblTitle);

textField = new JTextField();
total=c.getCart().getTotal();
String formattedText = dc.format(total);

textField.setText("$"+formattedText);
textField.setBounds(61, 364, 86, 20);
contentPane.add(textField);
textField.setColumns(10);

addtoRow();

```

```

    }

    public void addtoRow(){

        inv.LoadInventory();
        DefaultTableModel model= (DefaultTableModel)table.getModel();
        Object rowData[]=new Object[5];
        for(int i=0;i<inv.getProducts().size();i++){

            rowData[0]=inv.getProducts().get(i).getId();
            rowData[1]=inv.getProducts().get(i).getName();
            rowData[2]=inv.getProducts().get(i).getSellPrice();
            rowData[3]=inv.getProducts().get(i).getQuantity();

            model.addRow(rowData);
        }
    }
}

```

```

User.java
package project;

import java.io.Serializable;

public class User implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    /**
     * create a User Constructor
     */
    public User(String username,String password,String type){
        this.username=username;
        this.password=password;
    }
}

```

```

        this.typeOfUser=type;
    }
    /*
    * get the username
    * @return the username
    */
    public String getUsername() {
        return username;
    }
    /*
    * get the user password
    * @return password
    */
    public String getPassword() {
        return password;
    }
    /*
    * get the user type
    * @return the type
    */
    public String getTypeOfUser() {
        return typeOfUser;
    }

    private String username;
    private String password;
    private String typeOfUser;
}

```