Michael Del Campo, Thomas Sonderman, Christopher Mullen

Shopping Cart

The main purpose for the Shopping Cart System is to have buyers and sellers have an online marketplace that they can use to purchase and sell. It allows a GUI based interaction between the spring gui app and a buyer and seller.

Application Features

The Shopping Cart System allows sellers to list products for purchase and and buyers to browse products and to purchase products for sale. The Shopping System manages.

The application must provide the following features:

1. users can list their products for sale, providing:
   - price
   - product title
   - product full description

   system maintains a list of product for sale and current inventory.
2. users can browse the list of for-sale products.
3. users can select an product from the list and retrieve a full description.
4. users can purchase any products on the page.
5. users can checkout and have products shipped to them.
6. sellers can list an product.
7. sellers can update the information on an product.
8. sellers can remove an product from the online store.
9. sellers can add a new product to the online store.

The application is required to maintain a persistent product store between executions.

User Interface

The Shopping Cart System application has a gui-based user interface, centered on a menu of possible user options.

1. The "Default Menu" for seller displays:
   - a list of products for the seller.

    - an option to edit each product.
    - an option to remove an product.
2. The “Default Menu” for buyer displays
    - a list of products for sale.
    - an option to “add to cart”
    - a checkout button.
3. If the seller selects to list a new product, the Shopping Cart System will display the "List product" page. The user must enter:
    - product title
    - product full description
    - product price
4. If the user logs into system and picks an product from the main page to purchase. The actor then adds the product to their cart and pays for the product.
5. If the Actor(seller) logs into system. Actor adds a new product to their inventory.
6. If the Actor(seller) updates an product’s information to change the price. Actor(seller) logs in and goes to the product inventory page. On the product inventory page they select the product and then are directed to the product detail page. On the product detail page they are able to change the products information.
7. If the Actor(buyer) decides to cancel their products. Actor(buyer) selects the shopping cart icon and hits the clear cart. The shopping cart is cleared of all products.
8. If the Actor(buyer) changes the quantity of an product in their cart on the review/update screen. The change in the quantity then updates the price / total of their shopping cart.
9. If the Actor(buyer) cancels the order on the checkout screen. The shopping cart products will be cleared from their screen and they are returned to the main page.

Application Architecture

- The Auction System is implemented as a Java application with Swing Boot web based application and is executed from jar.

A customer logs in with a user-name and password and a window (frame) opens where he can browse through a list of available products that includes the product name, price, and available quantity. From this window the customer can select products and add them to the shopping cart or

they can click on a product and get the full product description, pricing and availability (quantity available) in a pop-up window. The
customer can add the product to the shopping cart (quantity), depending on availability. The shopping cart total amount is kept current on the main product browse window.

The customer can proceed to checkout at any time. On the checkout window, the shopping cart can be updated by changing the product count for each product in the cart. At checkout the customer verifies the shopping cart content and pays for the goods by supplying the credit card. The application does not arrange for shipping.

When the seller logs in, a window opens where the current state of the inventory is shown. The seller can update the inventory by adding products - specifying product name, invoice price, sell price and by updating the available quantity. The internal product representation includes ID, type, quantity, invoice price, and selling price.

The application must keep track of all costs, revenues and profits.
The seller can access this information from the application web interface.

Glossary Of Terms

- Seller Main Page - /seller/
- System Mailer - /mail for email
- System - swing gui app.
- Actor - either a buyer or seller.
- Buyer Main page - /store/
- Shopping Cart - The shopping cart is the temporary storage attached to each buyer that saves the state of their list of products.
- product - product that either the seller is selling or buyer is buying.
- Seller - Actor that logs in to sell/list/remove products from the online store.
- Buyer - Actor that logs in to purchase products from the online store.
- Inventory - The store's inventory of products that are for sale.

Use Cases

UC #1 User Logs In

1. The system requests that the actor enter his/her name and password.
2. The actor enters his/her name and password.
3. The system validates the entered name and password and logs the actor into the system.

UC #1 Variation 1 Bad/Username Password

1.1 In step 2, If in the *Basic Flow* the actor enters an invalid name and/or password, the system displays an error message.

1.2 The actor can choose to either return to the beginning of the *User Logs In* or cancel the login, at which point the use case ends.

UC #2 Customer Adds products to Shopping Cart

1. The actor selects Add to Cart button on product.
2. The system validates product is in inventory.
3. The system reserves product for user.
4. The system updates current shopping list with product.
5. Confirmation on screen to Actor that product has been added.

UC #2 Variation 1 Customer Signs off

1.1 Start at Step 3.
1.2 The system detects a sign off.
1.3 The system updates reserved products back to inventory.

UC #2 Variation 3 product not available in inventory

2.1 Start at Step 2.
2.1 The system detects insufficient inventory with product.
2.2 The system displays message to Actor that product isn't available and sets notification when available.

UC #3 Customer Reviews Product Details

1. Customer selects details button.
2. The system returns the information/details about product.

UC #4 Customer removes product from cart

1. The actor removes product from cart.
2. The system updates actor's cart removing product.
3. The system changes inventory and removes hold on product.

UC #5 Customer Reviews/Updates Shopping Cart

1. The actor selects Cart button.
2. The system returns all current products and quantities.
3. The system gives a total and shipping estimate.
4. The system provides a prompt for payment information.

UC #5 Variation 1 Customer Clears Cart

1.1 Start at Step 3.
1.2 The actor clears all products from cart.
1.3 The system returns reserved products to inventory.
The system returns actor to main page.

UC #6 Customer Checks Out

1. The actor reviews products, enters payment information and hits check out.

2. The system changes reserved products status to sold.
3. The system generates a packing slip and mailing label.
4. The actor receives an on screen notification order has been accepted.
5. The actor receives an email confirmation of order.

UC #7 Seller Reviews/Updates Inventory

1.The actor(seller) selects the *Review Inventory* button.
2. The system returns the current products listed under the seller.
3. The actor selects an product to update.
4.The system returns the edit product page with the product information.
5.The actor(seller) enters any changed information.
6. The actor selects the *Update product* button.
7. The system updates the product from the information entered.
8. The system returns actor(seller) to inventory page.

UC #7 Variation #1 Seller cancels update

1.1 Start from Step 3.
1.2 Systems returns actor(seller) to seller main page.

UC #8 Seller Adds New Product

1. The actor selects add new product button.
2. The system returns a form with fields for the product object.
3. The actor fills out the form fields.
4. The actor selects the submit button.
5. The system receives form fields and updates inventory.
6. The system returns actor to seller main page.

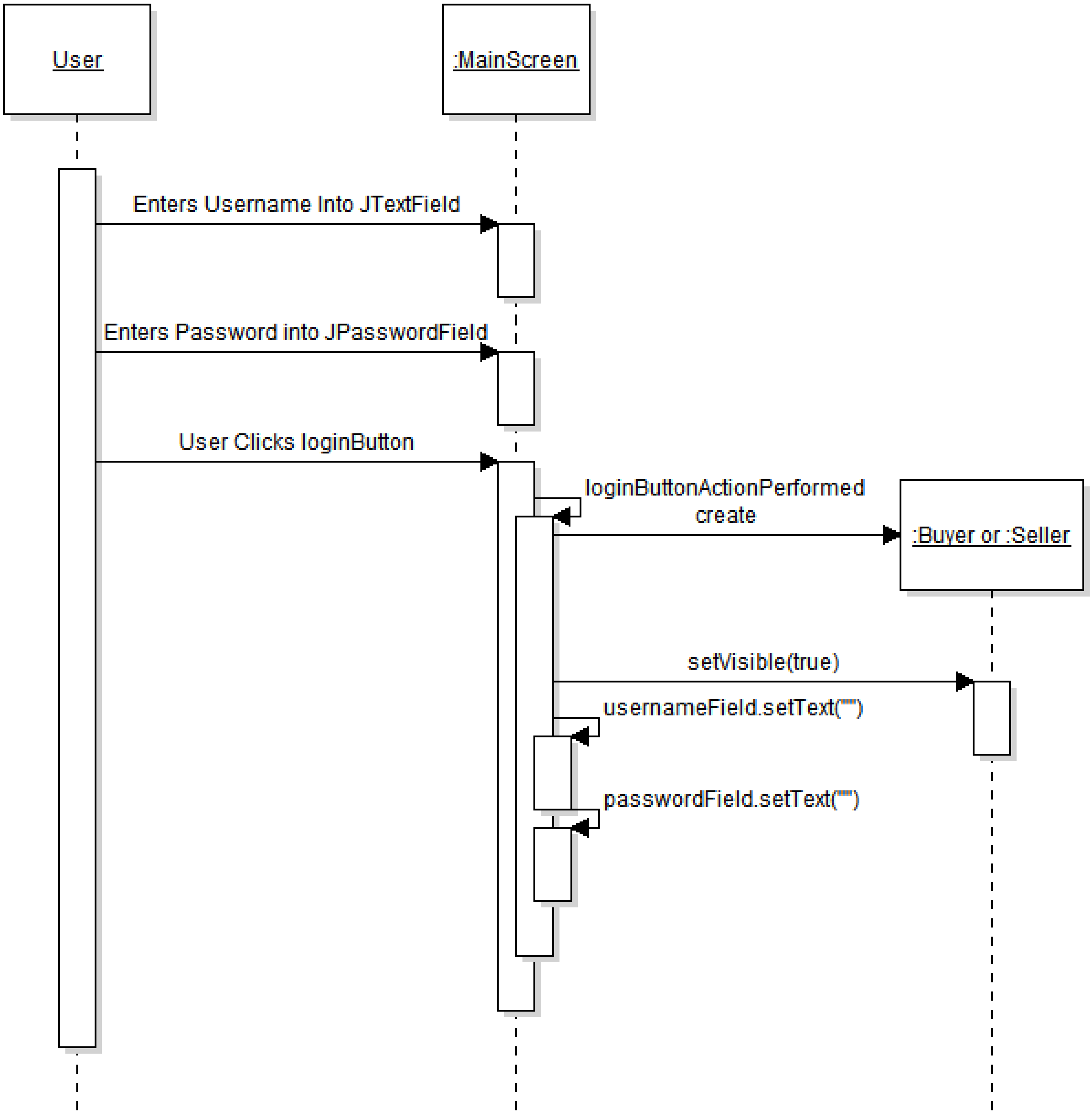
UC #8 Variation #1 Seller cancels Add New Product

1.1 Start from Step 3.
1.2 Systems returns actor(seller) to seller main page.

Detailed Use Cases

UC #1 Customer Logs In

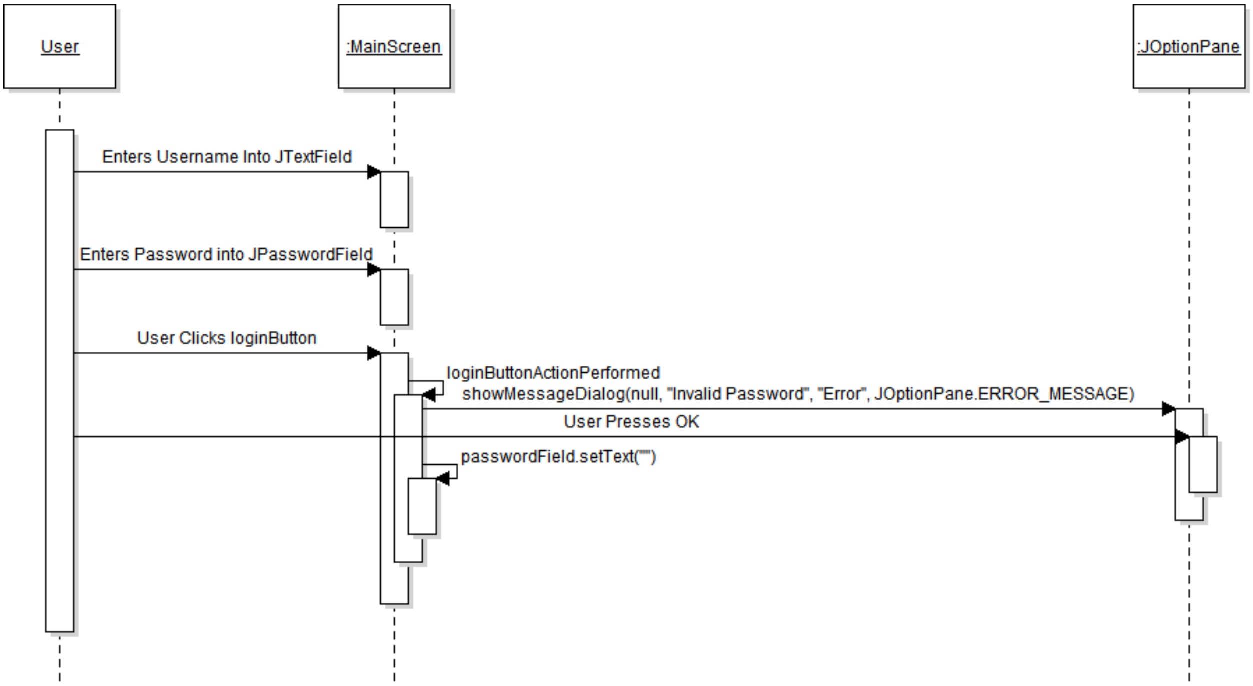
1. The system requests that the User enter his/her name and password.
2. The User enters his/her name and password.
3. The system validates the entered name and password and logs the User into the system.


User
:MainScreen
Enters Username Into JTextField
Enters Password into JPasswordField
User Clicks loginButton
loginButtonActionPerformed
create
:Buyer or :Seller
setVisible(true)
usernameField.setText("")
passwordField.setText("")

UC #1 Variation 1 Bad/Username Password

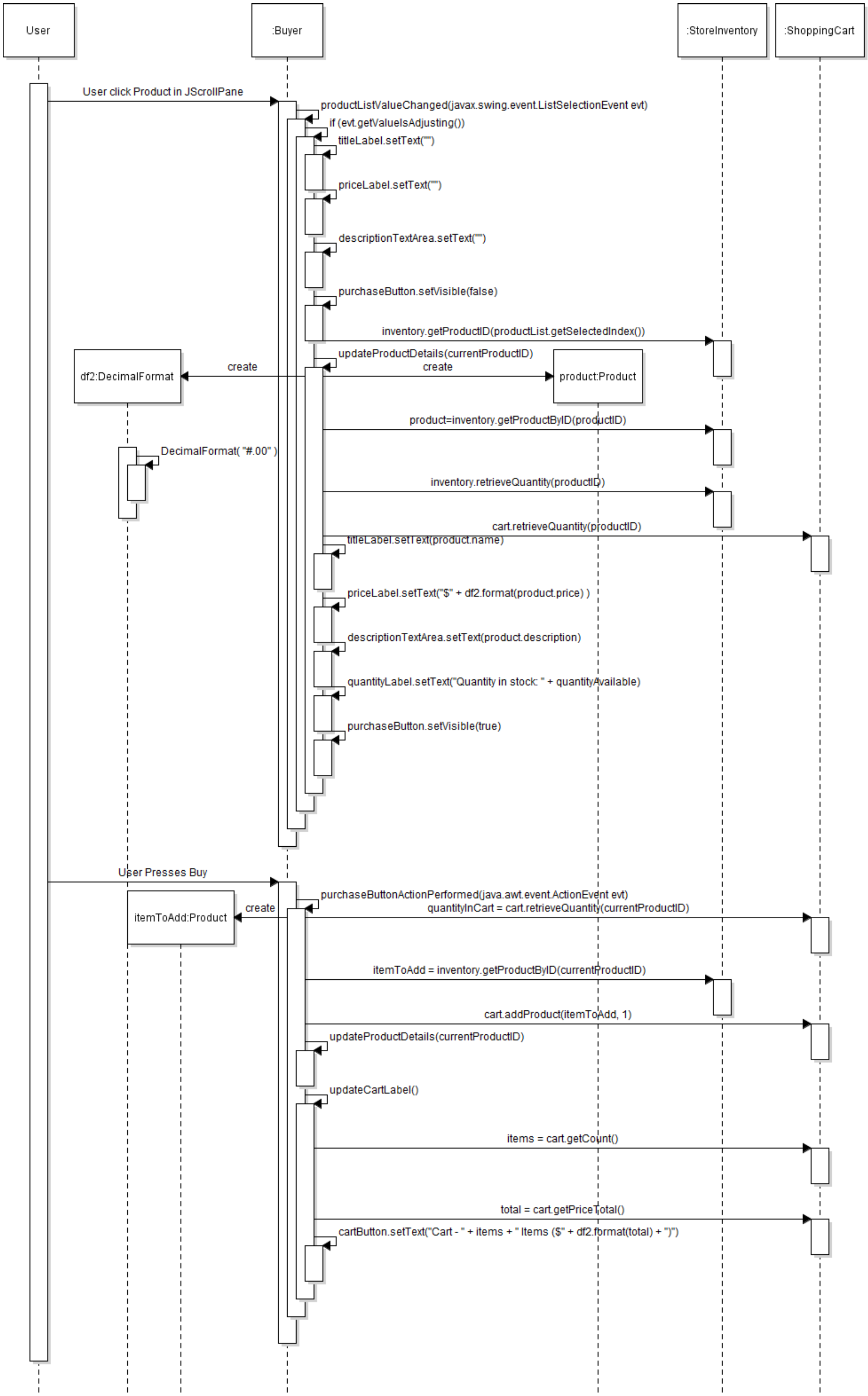
1.1 In step 2, If in the *Basic Flow* the User enters an invalid name and/or password, the system displays an error message.

1.2 The User can choose to either return to the beginning of the *Customer Logs In* or cancel the login, at which point the use case ends.


User
:MainScreen
:JOptionPane
Enters Username Into JTextField
Enters Password into JPasswordField
User Clicks loginButton
loginButtonActionPerformed
showMessageDialog(null, "Invalid Password", "Error", JOptionPane.ERROR_MESSAGE)
User Presses OK
passwordField.setText("")

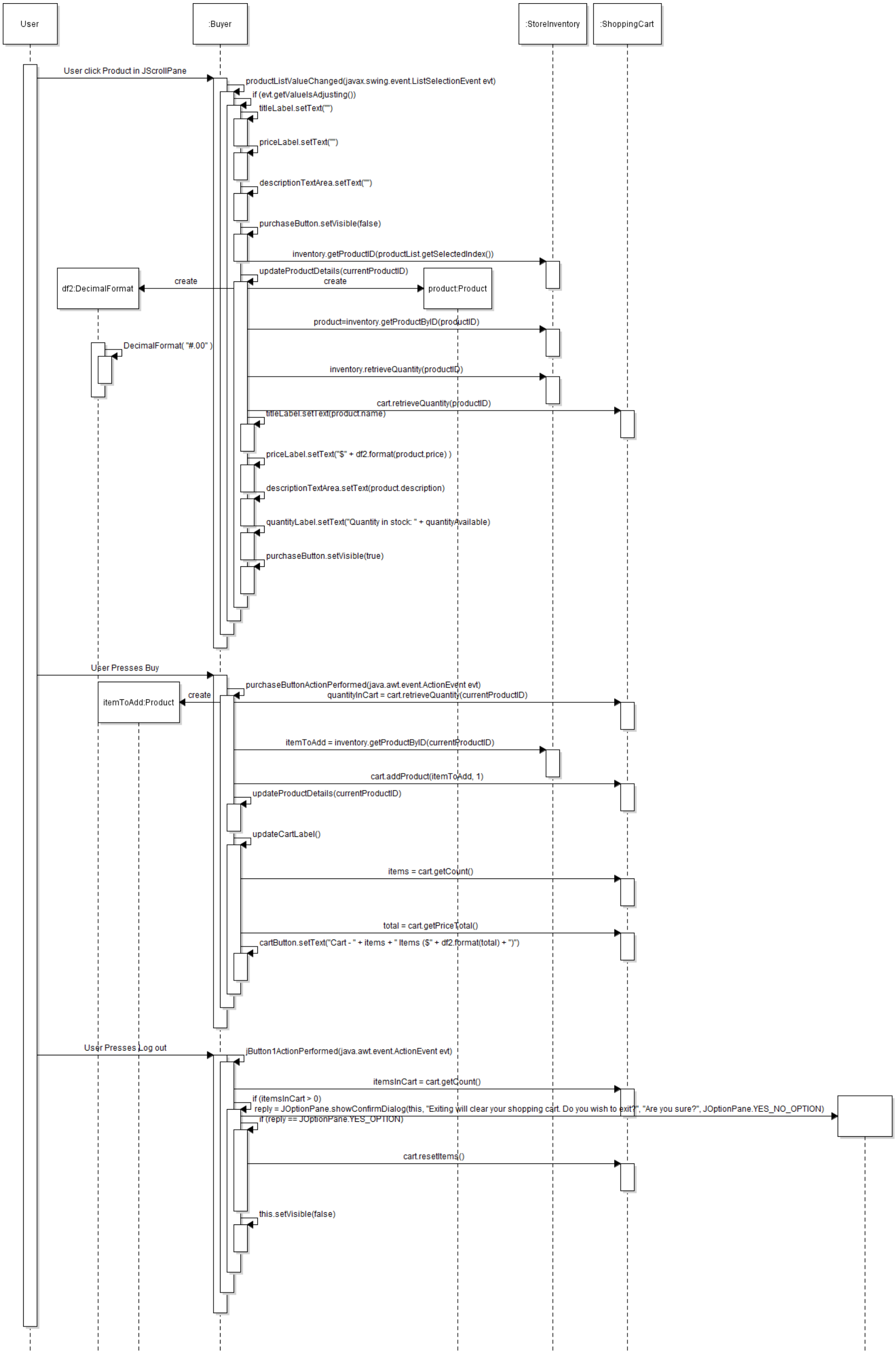
UC #2 Customer Adds Products to Shopping Cart

1. The User selects Add to Cart button on Product.
2. The system validates Product is in ProductList.
3. The system reserves Product for Customer.
4. The system updates current ShoppingCart with Product.
5. Confirmation on screen to User that Product has been added.(Updated Shopping Cart)


User
:Buyer
:StoreInventory
:ShoppingCart
User click Product in JScrollPane
productListValueChanged(javax.swing.event.ListSelectionEvent evt)
if (evt.getValueIsAdjusting())
titleLabel.setText("")
priceLabel.setText("")
descriptionTextArea.setText("")
purchaseButton.setVisible(false)
inventory.getProductID(productList.getSelectedIndex())
updateProductDetails(currentProductID)
create
df2:DecimalFormat
create
product:Product
product=inventory.getProductByID(productID)
DecimalFormat("#.00")
inventory.retrieveQuantity(productID)
cart.retrieveQuantity(productID)
titleLabel.setText(product.name)
priceLabel.setText("$" + df2.format(product.price))
descriptionTextArea.setText(product.description)
quantityLabel.setText("Quantity in stock: " + quantityAvailable)
purchaseButton.setVisible(true)
User Presses Buy
purchaseButtonActionPerformed(java.awt.event.ActionEvent evt)
create
itemToAdd:Product
quantityInCart = cart.retrieveQuantity(currentProductID)
itemToAdd = inventory.getProductByID(currentProductID)
cart.addProduct(itemToAdd, 1)
updateProductDetails(currentProductID)
updateCartLabel()
items = cart.getCount()
total = cart.getPriceTotal()
cartButton.setText("Cart - " + items + " Items ($" + df2.format(total) + ")")

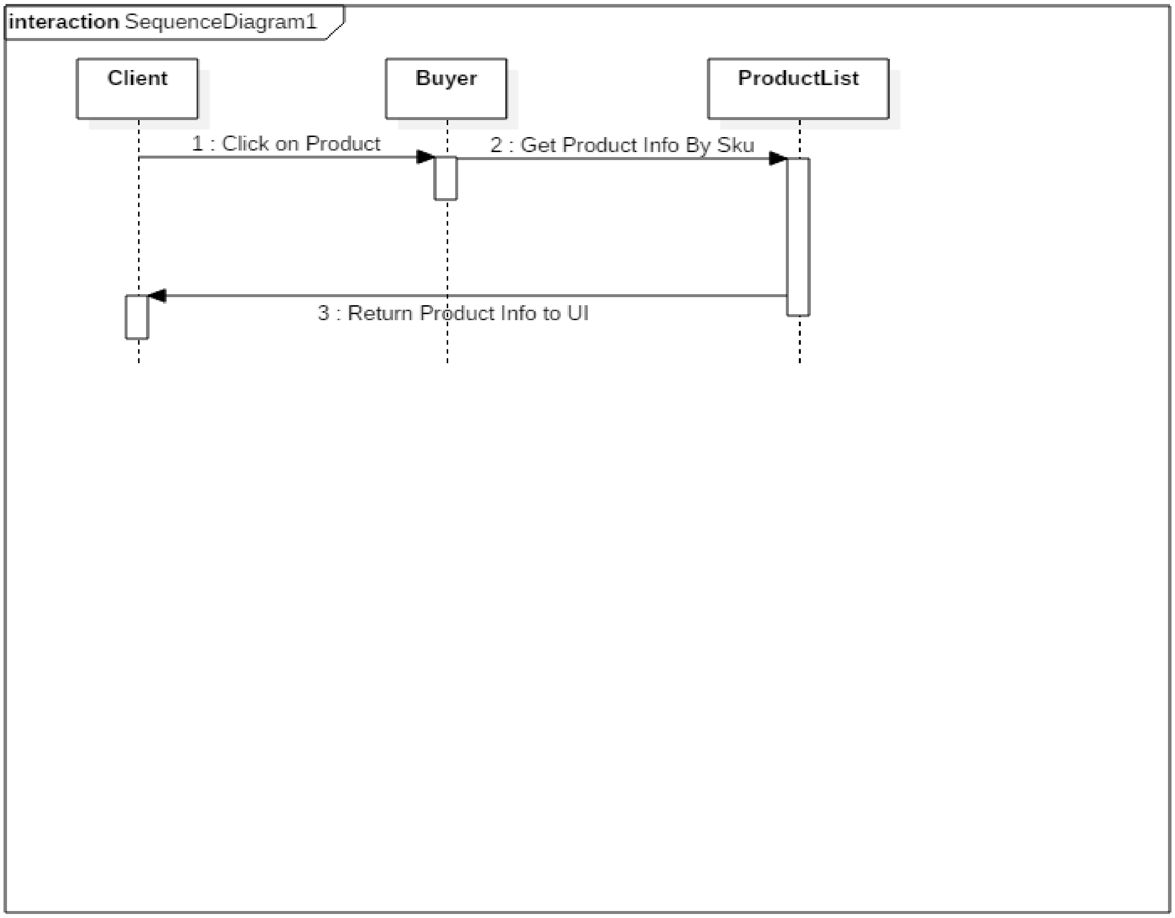
UC #2 Variation 1 Customer Signs off

1.1 Start at Step 3.
1.2 The system detects a sign off.
1.3 The system updates reserved Products back to ProductList.


User
:Buyer
:StoreInventory
:ShoppingCart
User click Product in JScrollPane
productListValueChanged(javax.swing.event.ListSelectionEvent evt)
if (evt.getValueIsAdjusting())
titleLabel.setText("")
priceLabel.setText("")
descriptionTextArea.setText("")
purchaseButton.setVisible(false)
inventory.getProductID(productList.getSelectedIndex())
updateProductDetails(currentProductID)
create
df2:DecimalFormat
create
product:Product
product=inventory.getProductByID(productID)
DecimalFormat("#.00")
inventory.retrieveQuantity(productID)
cart.retrieveQuantity(productID)
titleLabel.setText(product.name)
priceLabel.setText("$" + df2.format(product.price))
descriptionTextArea.setText(product.description)
quantityLabel.setText("Quantity in stock: " + quantityAvailable)
purchaseButton.setVisible(true)
User Presses Buy
purchaseButtonActionPerformed(java.awt.event.ActionEvent evt)
create
itemToAdd:Product
quantityInCart = cart.retrieveQuantity(currentProductID)
itemToAdd = inventory.getProductByID(currentProductID)
cart.addProduct(itemToAdd, 1)
updateProductDetails(currentProductID)
updateCartLabel()
items = cart.getCount()
total = cart.getPriceTotal()
cartButton.setText("Cart - " + items + " Items ($" + df2.format(total) + ")")
User Presses Log out
jButton1ActionPerformed(java.awt.event.ActionEvent evt)
itemsInCart = cart.getCount()
if (itemsInCart > 0)
reply = JOptionPane.showConfirmDialog(this, "Exiting will clear your shopping cart. Do you wish to exit?", "Are you sure?", JOptionPane.YES_NO_OPTION)
if (reply == JOptionPane.YES_OPTION)
cart.resetItems()
this.setVisible(false)

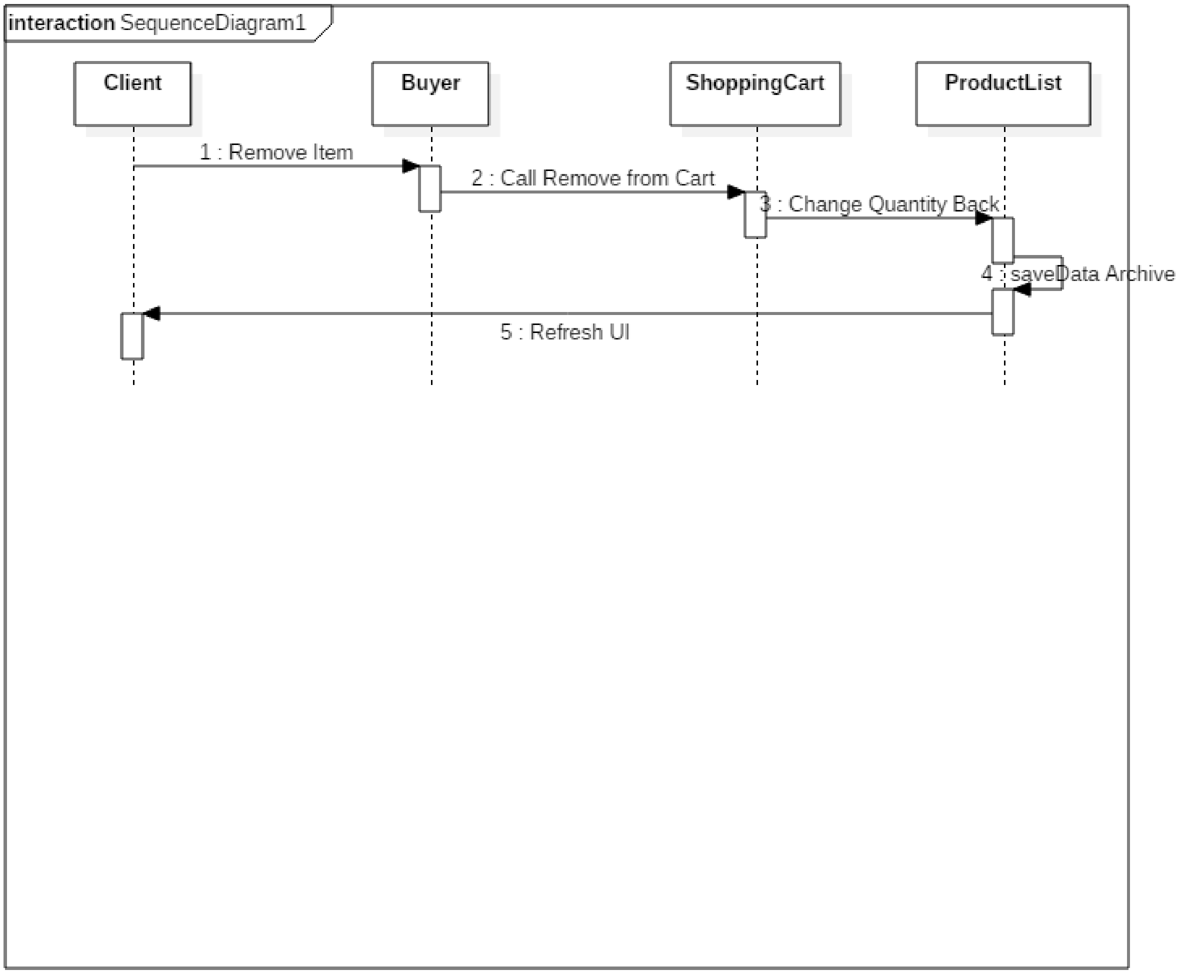
UC #3 Customer Reviews Product Details

1. Customer selects product.
2. The system returns the information/details about product in the right window.


interaction SequenceDiagram1
Client
Buyer
ProductList
1 : Click on Product
2 : Get Product Info By Sku
3 : Return Product Info to UI

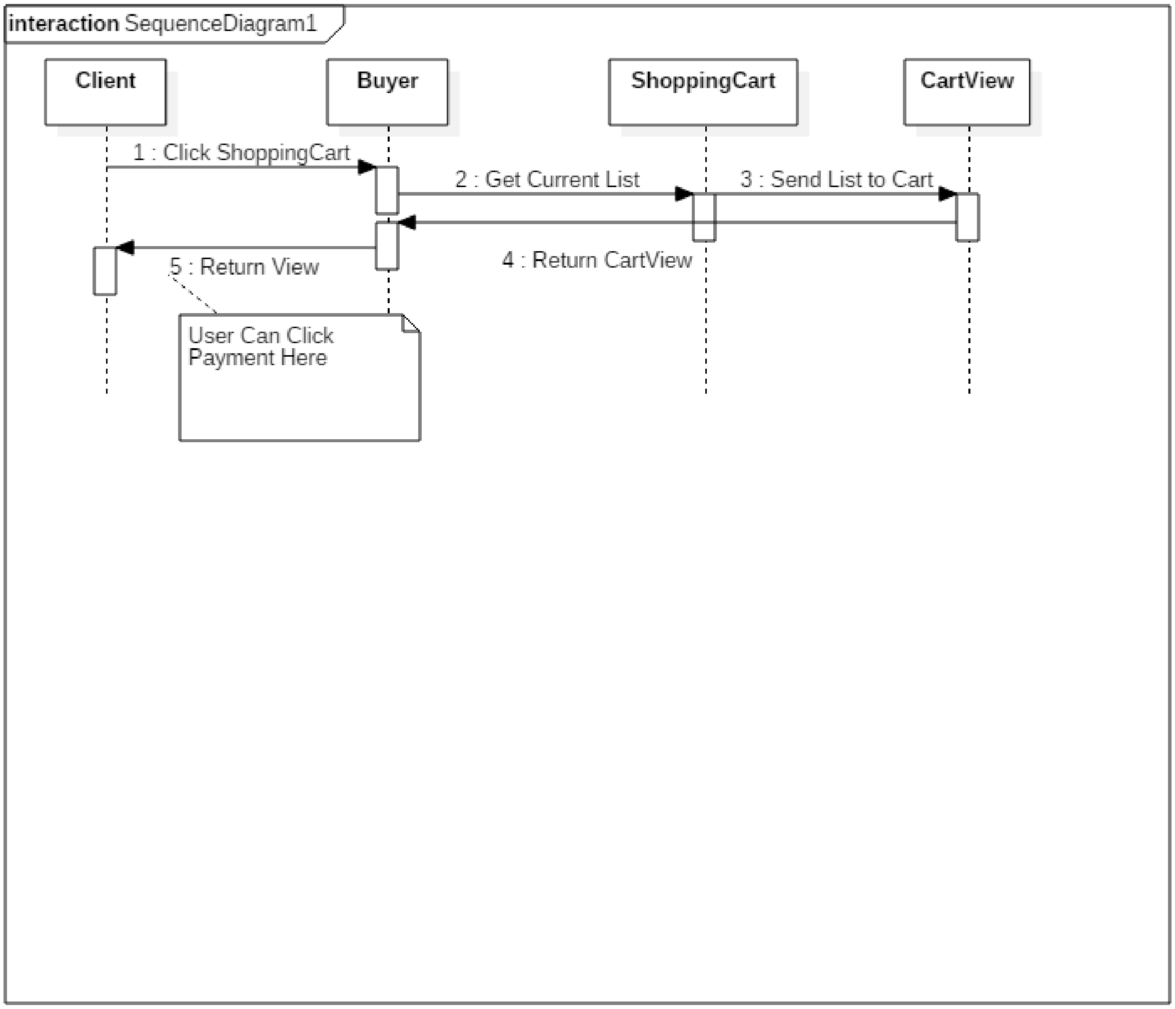
UC #4 Customer removes Product from cart

1. The User removes Product from ShoppingCart.
2. The system updates User’s ShoppingCart removing Product.
3. The system changes ProductList and removes hold on Product.



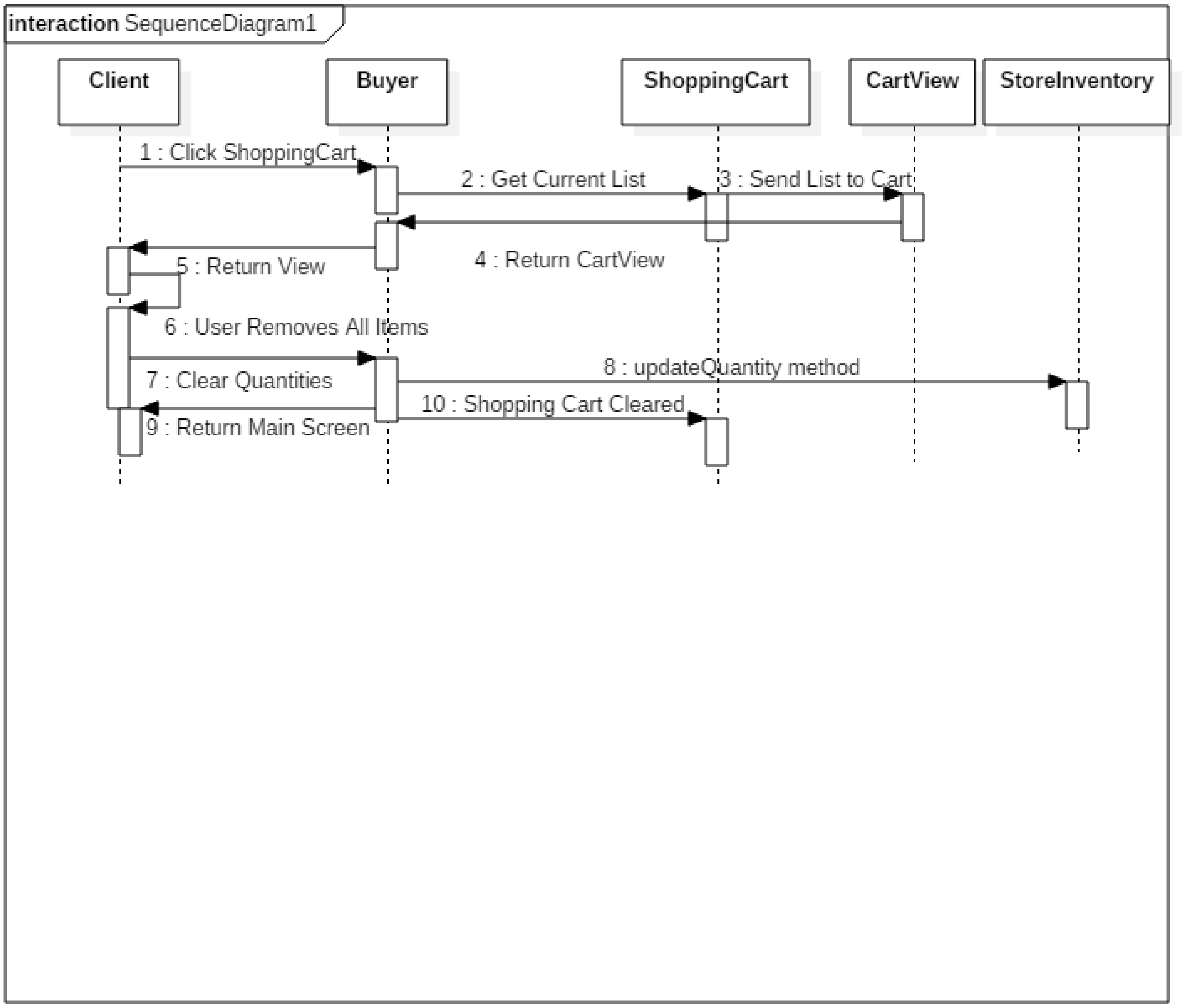
UC #5 Customer Reviews/Updates Shopping Cart

1. The User selects Cart button.
2. The system returns all current Products and quantities.
3. The system gives a total and shipping estimate.
4. The system provides a prompt for payment information.



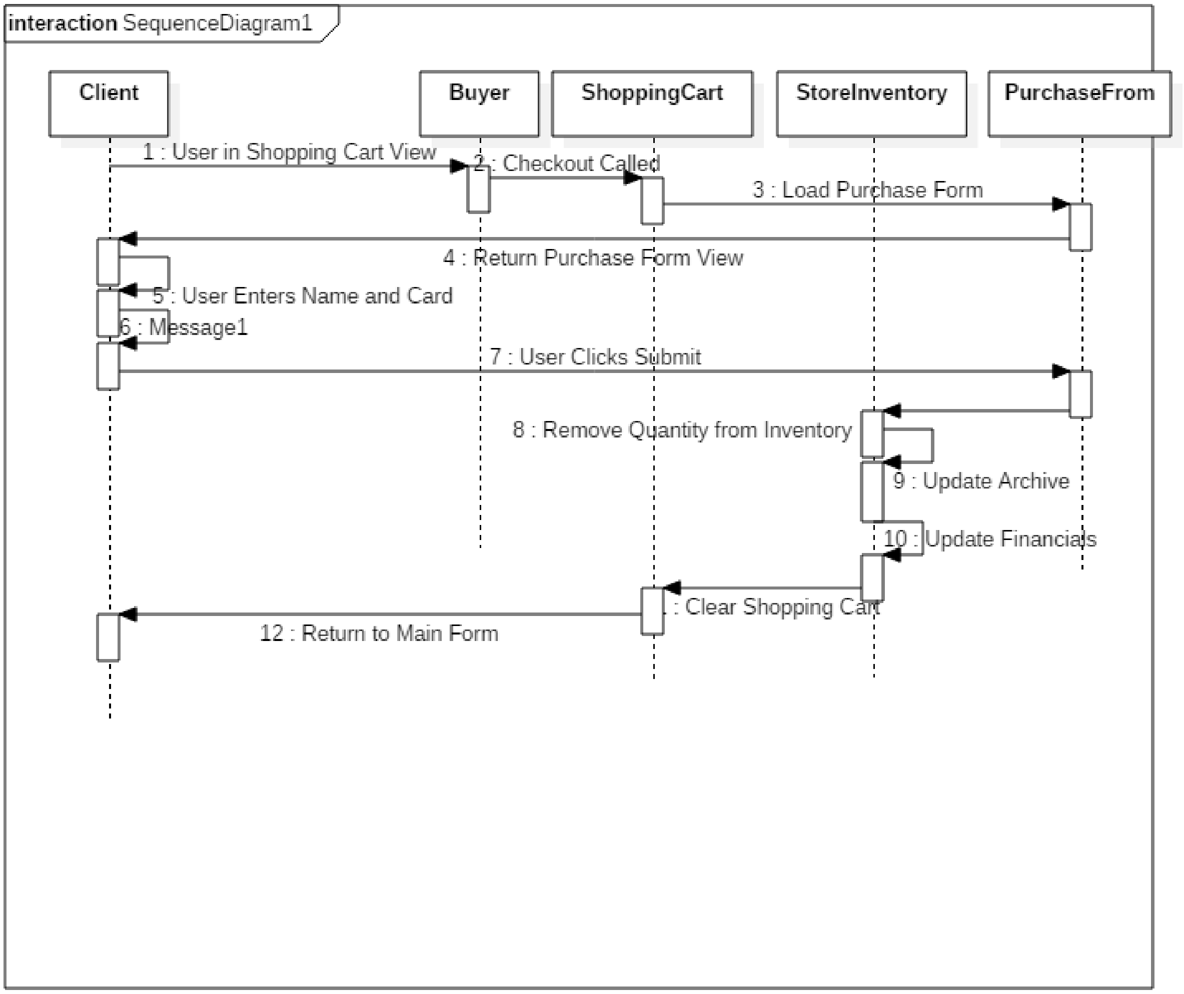
UC #5 Variation 1 Customer Clears Cart

1.1 Start at Step 3.
1.2 The User clears all Products from ShoppingCart.
1.3 The system returns reserved Products to ProductList.
The system returns User to main page.



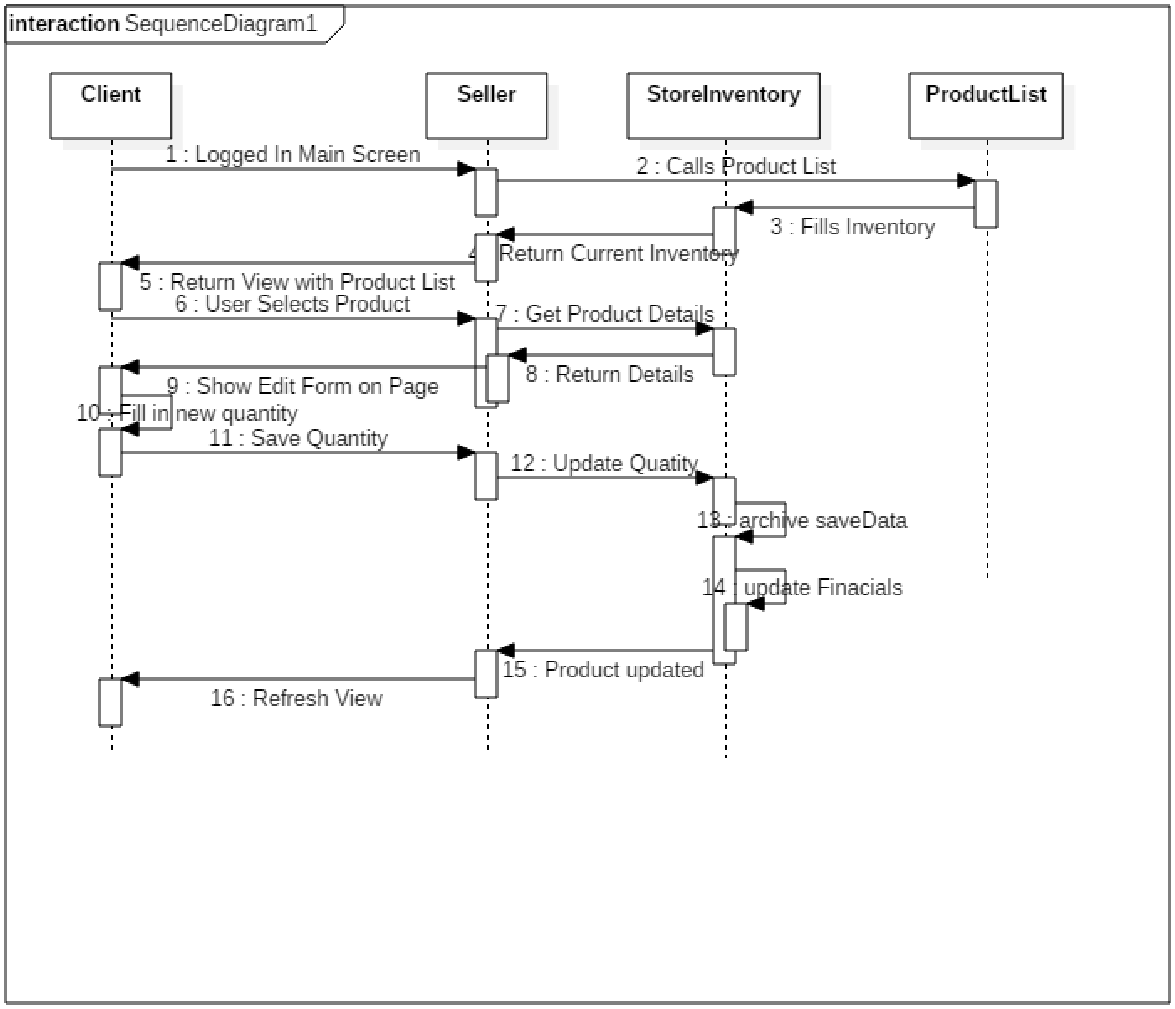
UC #6 Customer Checks Out

1. The User reviews Products
2. The system changes updates quantities.
3. Purchase Payment Screen loads and has a checkout.
4. The User receives an on screen notification order has been accepted.



UC #7 Seller Reviews/Updates ProductList

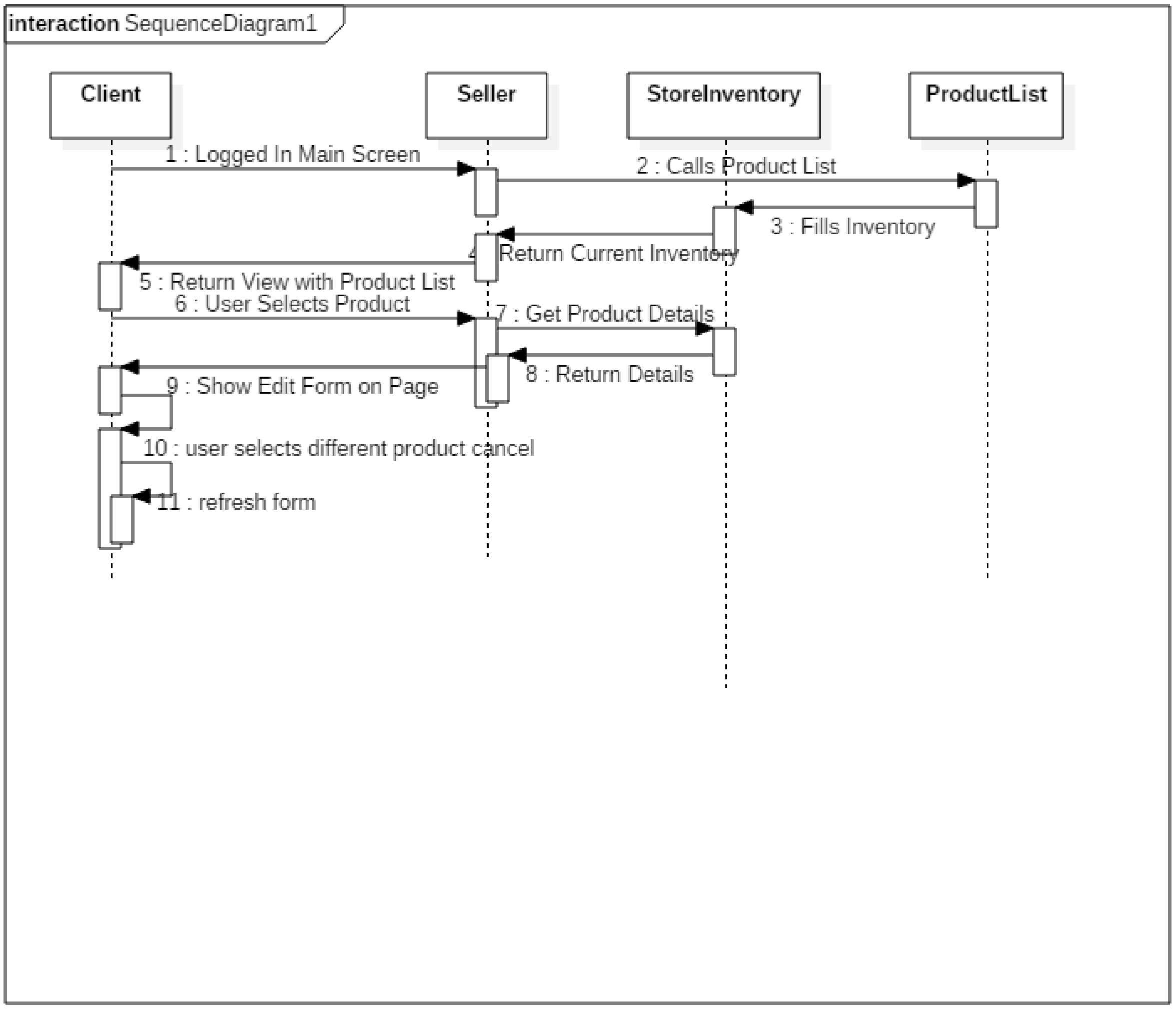
1.The User(seller) selects the *Review ProductList* button.
2. The system returns the current Products listed under the seller.
3. The User selects an Product to update.
4.The system returns the edit Product page with the product information.
5.The User(seller) enters any changed information.
6. The User selects the *Update Product* button.
7. The system updates the Product from the information entered.
8. The system returns User(seller) to ProductList page.



UC #7 Variation #1 Seller cancels update

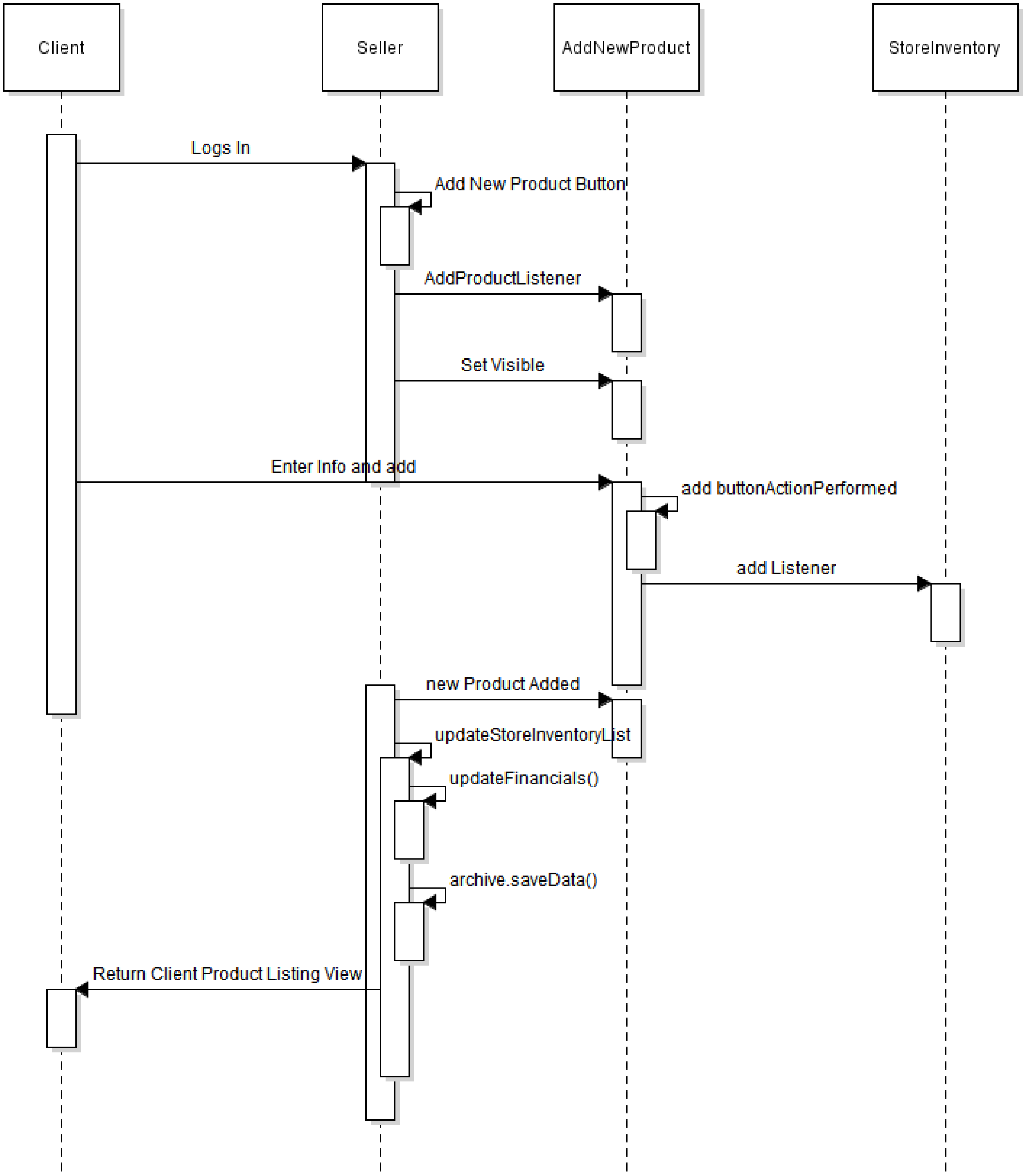
1.1 Start from Step 3.

1.2 Systems returns User(seller) to seller main page.


interaction SequenceDiagram1
Client
Seller
StoreInventory
ProductList
1 : Logged In Main Screen
2 : Calls Product List
3 : Fills Inventory
4 : Return Current Inventory
5 : Return View with Product List
6 : User Selects Product
7 : Get Product Details
8 : Return Details
9 : Show Edit Form on Page
10 : user selects different product cancel
11 : refresh form

UC #8 Seller Adds New Product

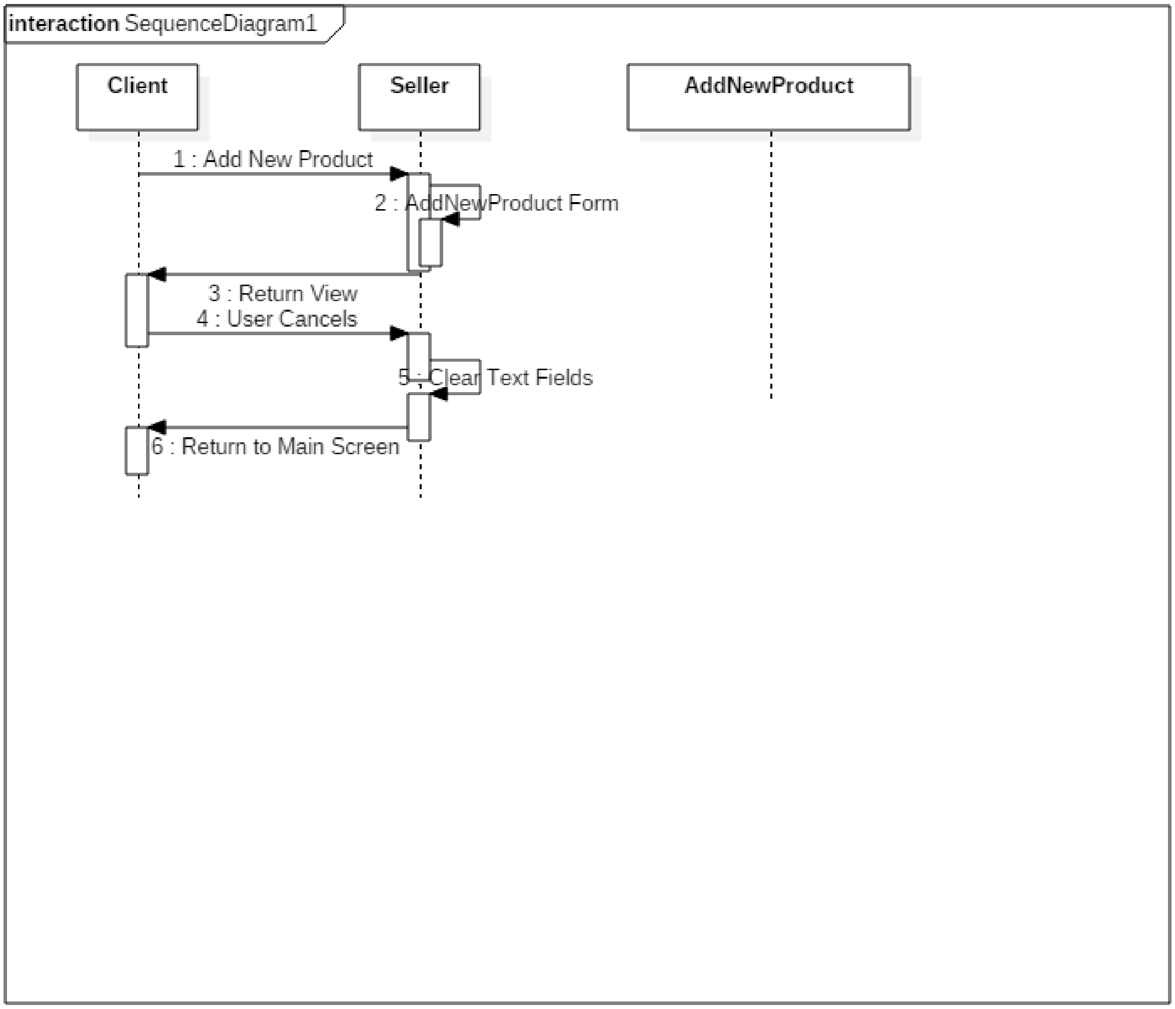
1. The User selects add new product button.
2. The system returns a form with fields for the product object.
3. The User fills out the form fields.
4. The User selects the submit button.
5. The system receives form fields and updates ProductList.
6. The system updates the archives saved files.
6. The system returns User to seller main page.



UC #8 Variation #1 Seller cancels Add New Product

1.1 Start from Step 3.

1.2 Systems returns User(seller) to seller main page.



CRC Cards

Customer	
Holds Customer Information Updates Customer Information May Be Seller of Product Has Previous Orders	User ShoppingCart Product Order

User	
Holds Login Information	Customer

ShoppingCart	
Current List of Products to Purchase Can Add Products Can Remove Products Updates Total Contains Discount	Customer

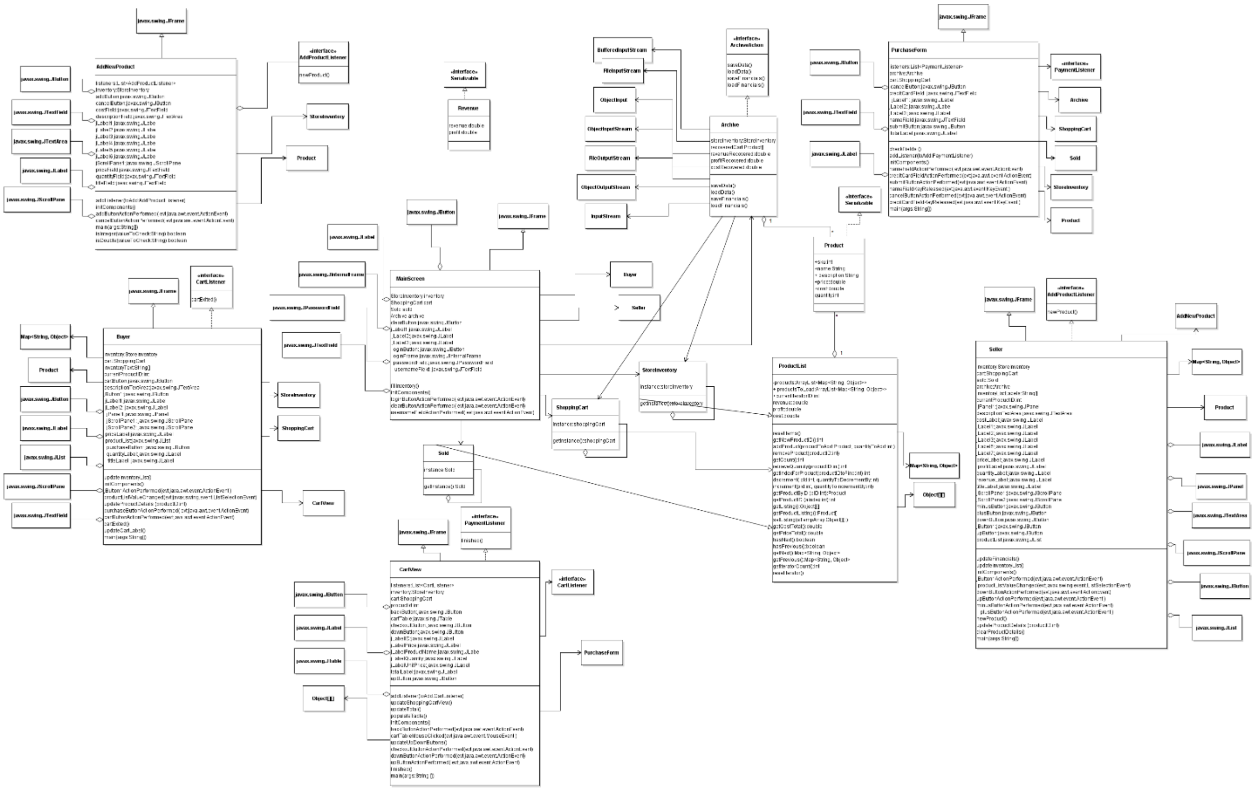
Product	
Product Information	Customer ProductList

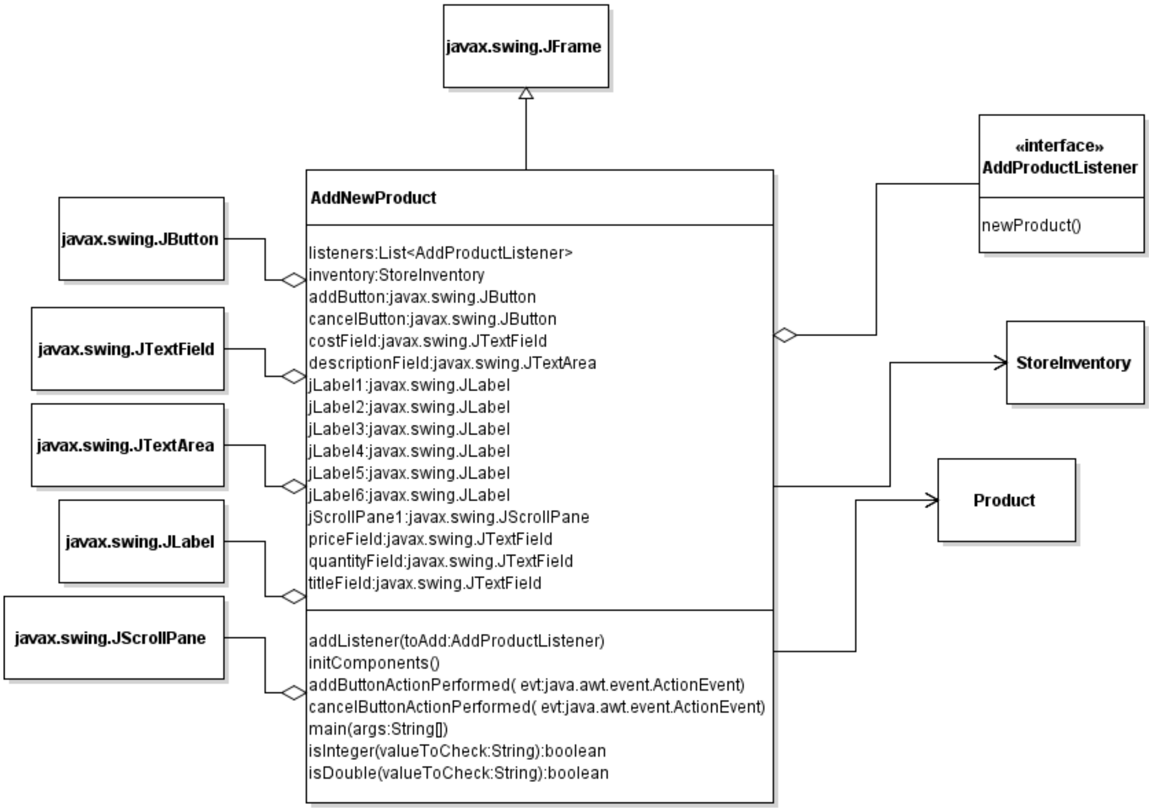
ProductList	
List of Products Add Products Delete Products Update Products	Product

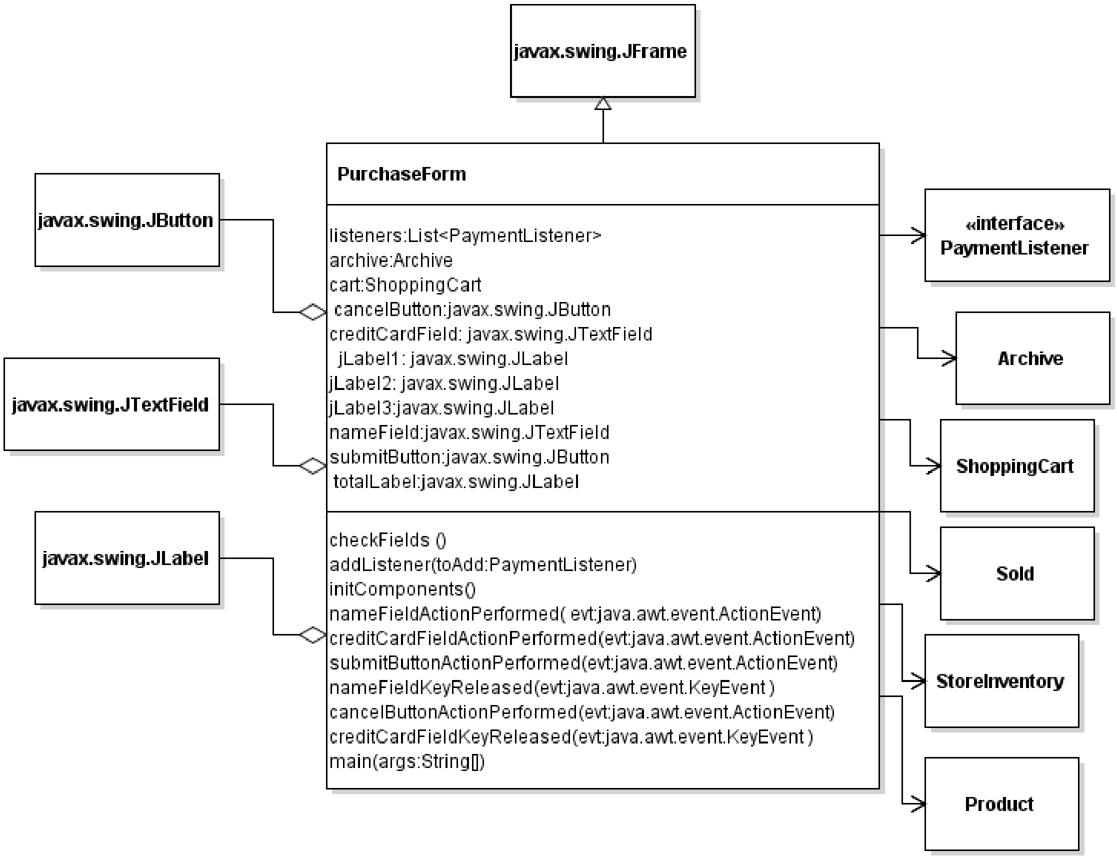
Order	
Previous Order Information	Customer

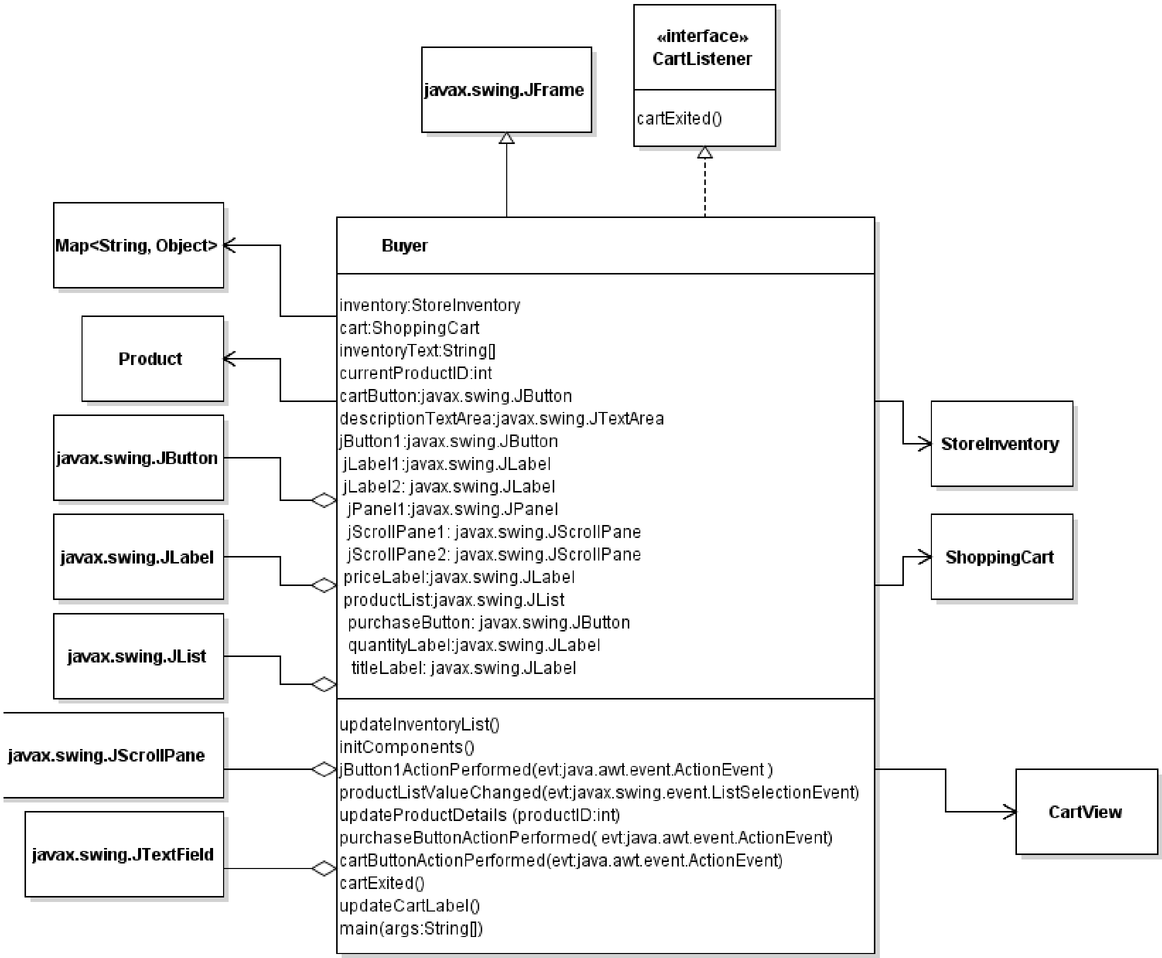
Archive	
Save Product Inventory Loads Product Inventory	StoreInventory Product

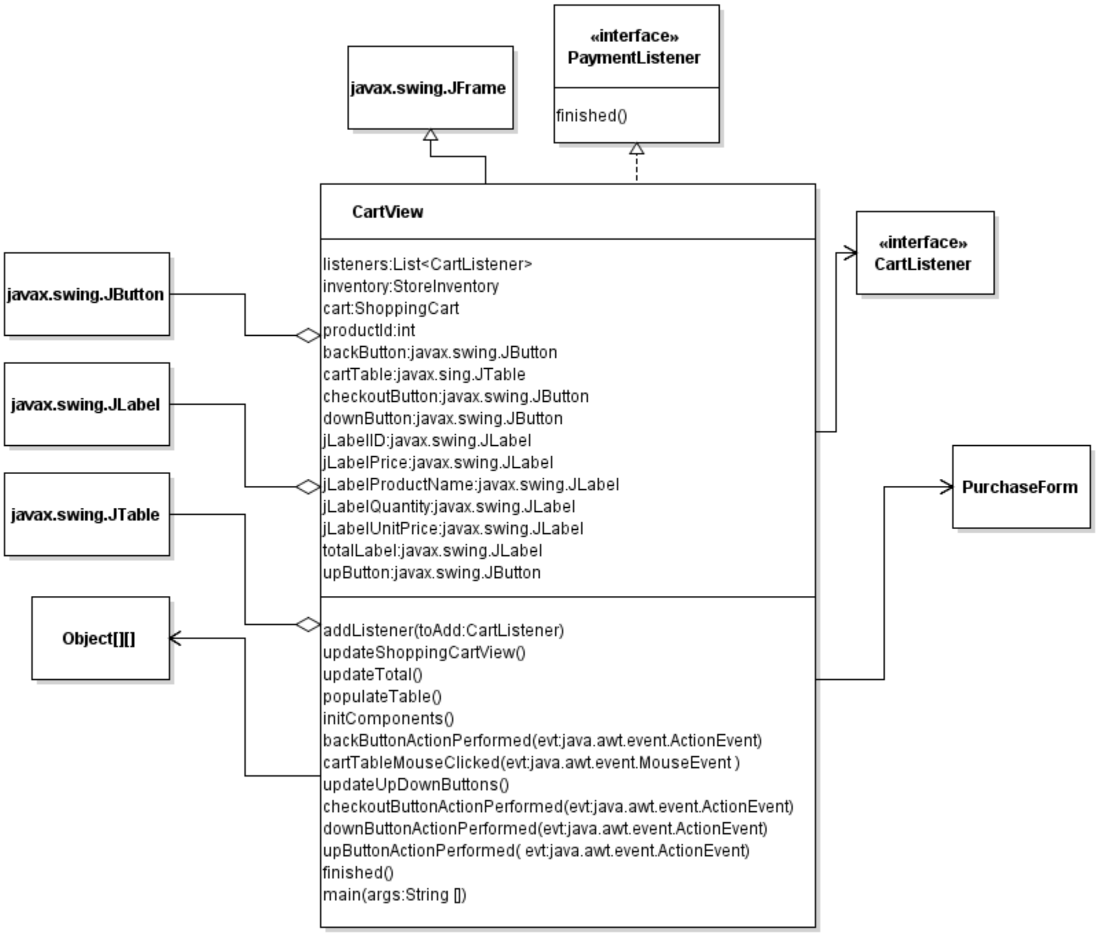
UML Diagram

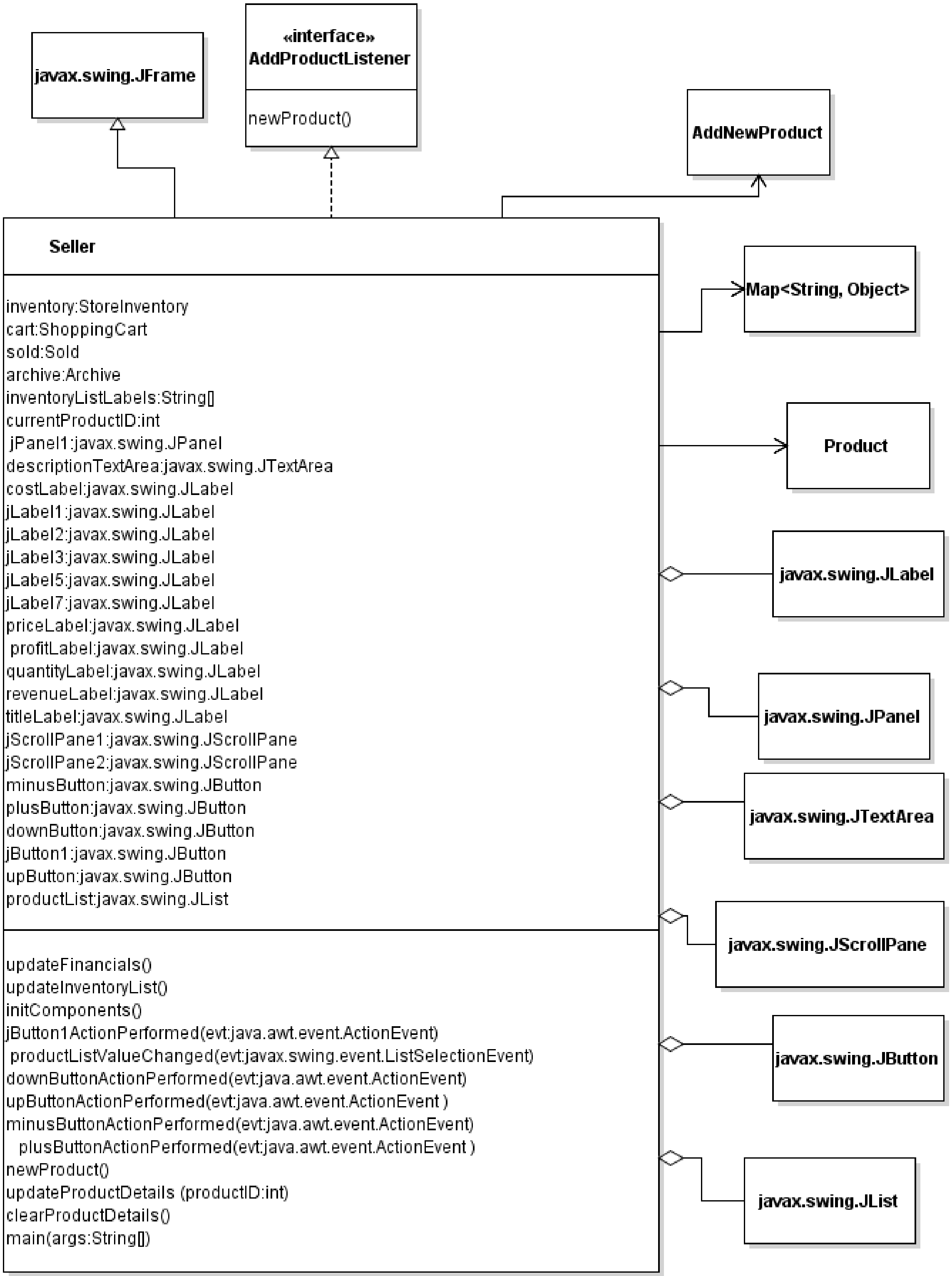

javax.swing.JFrame
AddNewProduct
«interface» AddProductListener
StoreInventory
Product
«interface» Serializable
Revenue
BufferedInputStream
FileInputStream
ObjectInput
ObjectInputStream
FileOutputStream
ObjectOutputStream
InputStream
«interface» ArchiveAction
Archive
PurchaseForm
«interface» PaymentListener
ShoppingCart
Sold
MainScreen
Buyer
Seller
«interface» CartListener
Map<String, Object>
StoreInventory
ShoppingCart
ProductList
CartView
Object[][]
javax.swing.JButton
javax.swing.JLabel
javax.swing.JTextField
javax.swing.JTextArea
javax.swing.JScrollPane
javax.swing.JList
javax.swing.JTable
javax.swing.JPanel
javax.swing.JInternalFrame
javax.swing.JPasswordField

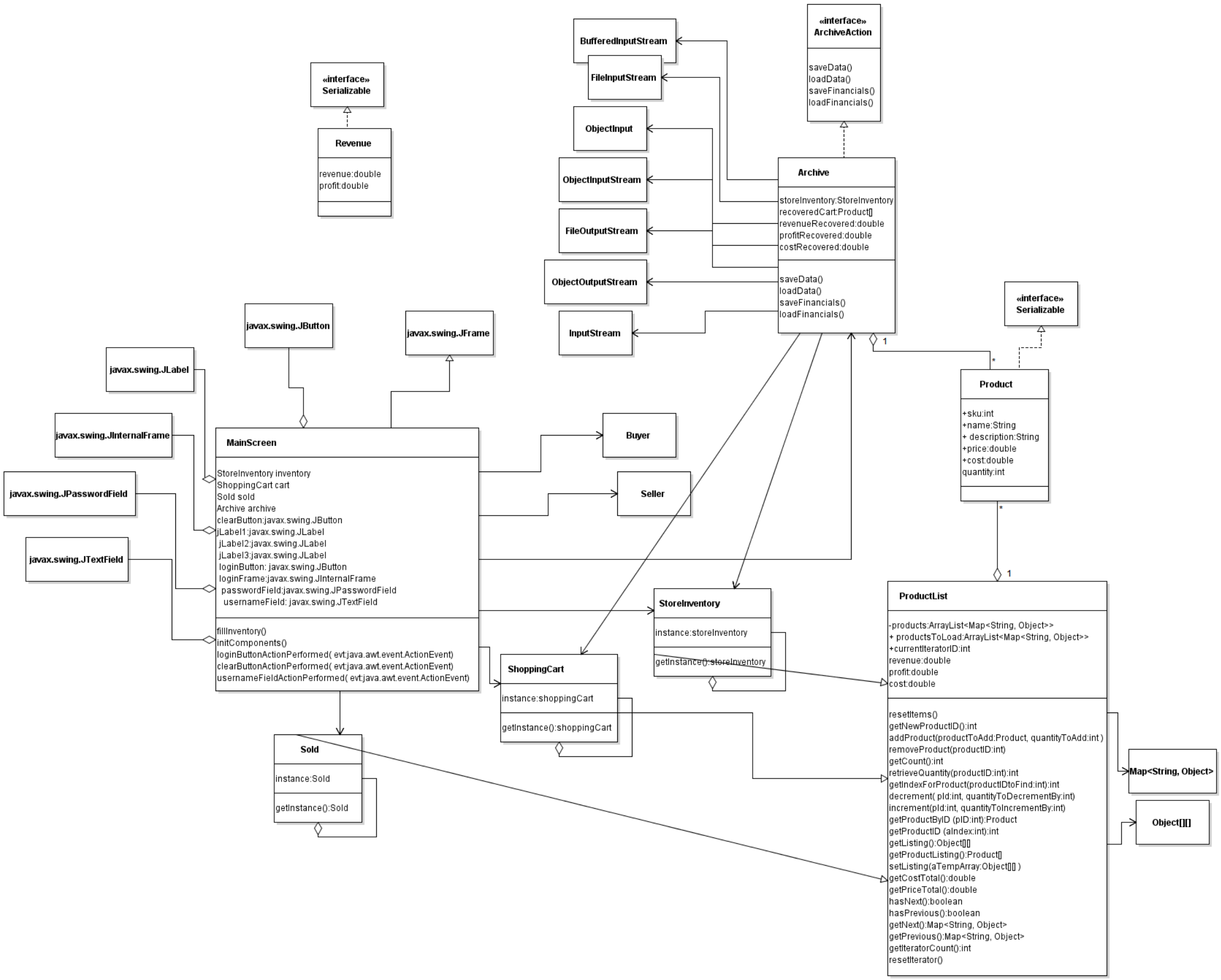

javax.swing.JFrame
AddNewProduct
listeners:List<AddProductListener>
inventory:StoreInventory
addButton:javax.swing.JButton
cancelButton:javax.swing.JButton
costField:javax.swing.JTextField
descriptionField:javax.swing.JTextArea
jLabel1:javax.swing.JLabel
jLabel2:javax.swing.JLabel
jLabel3:javax.swing.JLabel
jLabel4:javax.swing.JLabel
jLabel5:javax.swing.JLabel
jLabel6:javax.swing.JLabel
jScrollPane1:javax.swing.JScrollPane
priceField:javax.swing.JTextField
quantityField:javax.swing.JTextField
titleField:javax.swing.JTextField
addListener(toAdd:AddProductListener)
initComponents()
addButtonActionPerformed(evt:java.awt.event.ActionEvent)
cancelButtonActionPerformed(evt:java.awt.event.ActionEvent)
main(args:String[])
isInteger(valueToCheck:String):boolean
isDouble(valueToCheck:String):boolean
javax.swing.JButton
javax.swing.JTextField
javax.swing.JTextArea
javax.swing.JLabel
javax.swing.JScrollPane
«interface»
AddProductListener
newProduct()
StoreInventory
Product


javax.swing.JFrame
PurchaseForm
listeners:List<PaymentListener>
archive:Archive
cart:ShoppingCart
cancelButton:javax.swing.JButton
creditCardField: javax.swing.JTextField
jLabel1: javax.swing.JLabel
jLabel2: javax.swing.JLabel
jLabel3:javax.swing.JLabel
nameField:javax.swing.JTextField
submitButton:javax.swing.JButton
totalLabel:javax.swing.JLabel
checkFields ()
addListener(toAdd:PaymentListener)
initComponents()
nameFieldActionPerformed(evt:java.awt.event.ActionEvent)
creditCardFieldActionPerformed(evt:java.awt.event.ActionEvent)
submitButtonActionPerformed(evt:java.awt.event.ActionEvent)
nameFieldKeyReleased(evt:java.awt.event.KeyEvent)
cancelButtonActionPerformed(evt:java.awt.event.ActionEvent)
creditCardFieldKeyReleased(evt:java.awt.event.KeyEvent)
main(args:String[])
javax.swing.JButton
javax.swing.JTextField
javax.swing.JLabel
«interface»
PaymentListener
Archive
ShoppingCart
Sold
StoreInventory
Product

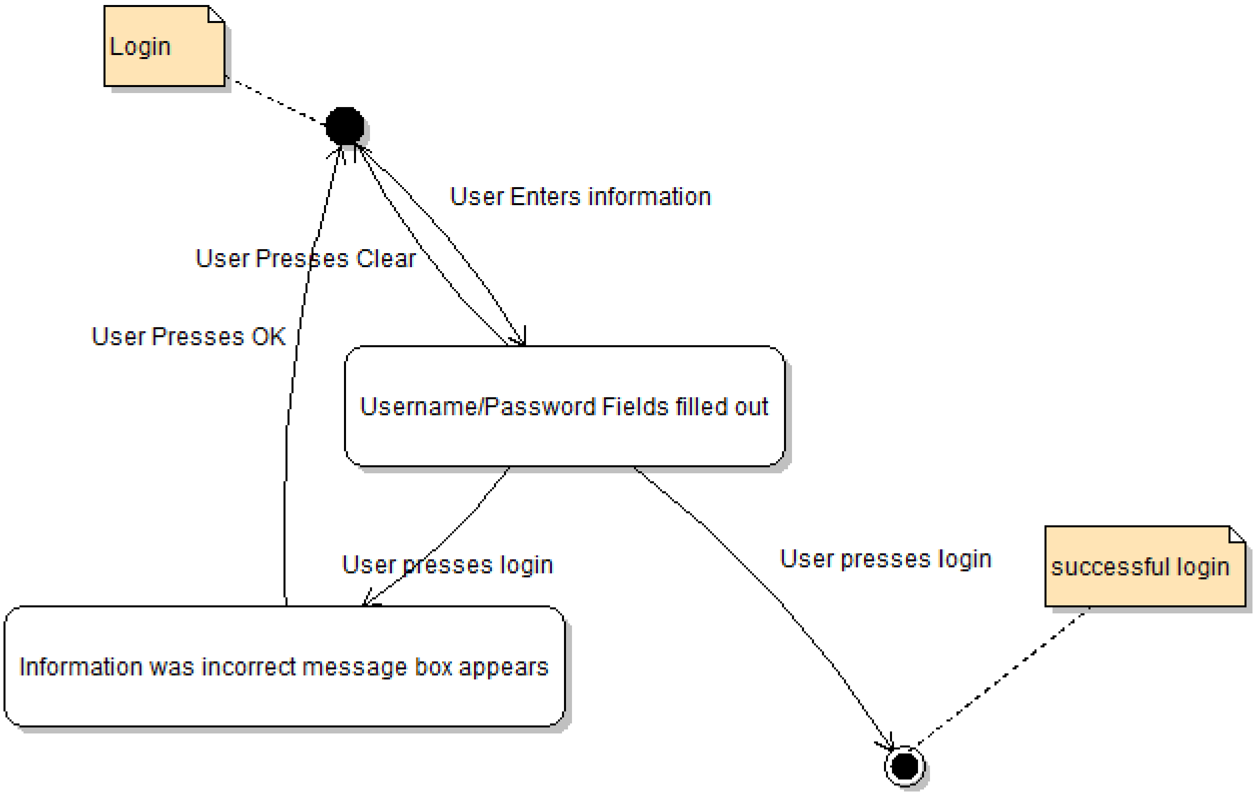

«interface»
CartListener
cartExited()
javax.swing.JFrame
Map<String, Object>
Product
javax.swing.JButton
javax.swing.JLabel
javax.swing.JList
javax.swing.JScrollPane
javax.swing.JTextField
Buyer
inventory:StoreInventory
cart:ShoppingCart
inventoryText:String[]
currentProductID:int
cartButton:javax.swing.JButton
descriptionTextArea:javax.swing.JTextArea
jButton1:javax.swing.JButton
jLabel1:javax.swing.JLabel
jLabel2: javax.swing.JLabel
jPanel1:javax.swing.JPanel
jScrollPane1: javax.swing.JScrollPane
jScrollPane2: javax.swing.JScrollPane
priceLabel:javax.swing.JLabel
productList:javax.swing.JList
purchaseButton: javax.swing.JButton
quantityLabel:javax.swing.JLabel
titleLabel: javax.swing.JLabel
updateInventoryList()
initComponents()
jButton1ActionPerformed(evt:java.awt.event.ActionEvent)
productListValueChanged(evt:javax.swing.event.ListSelectionEvent)
updateProductDetails (productID:int)
purchaseButtonActionPerformed(evt:java.awt.event.ActionEvent)
cartButtonActionPerformed(evt:java.awt.event.ActionEvent)
cartExited()
updateCartLabel()
main(args:String[])
StoreInventory
ShoppingCart
CartView

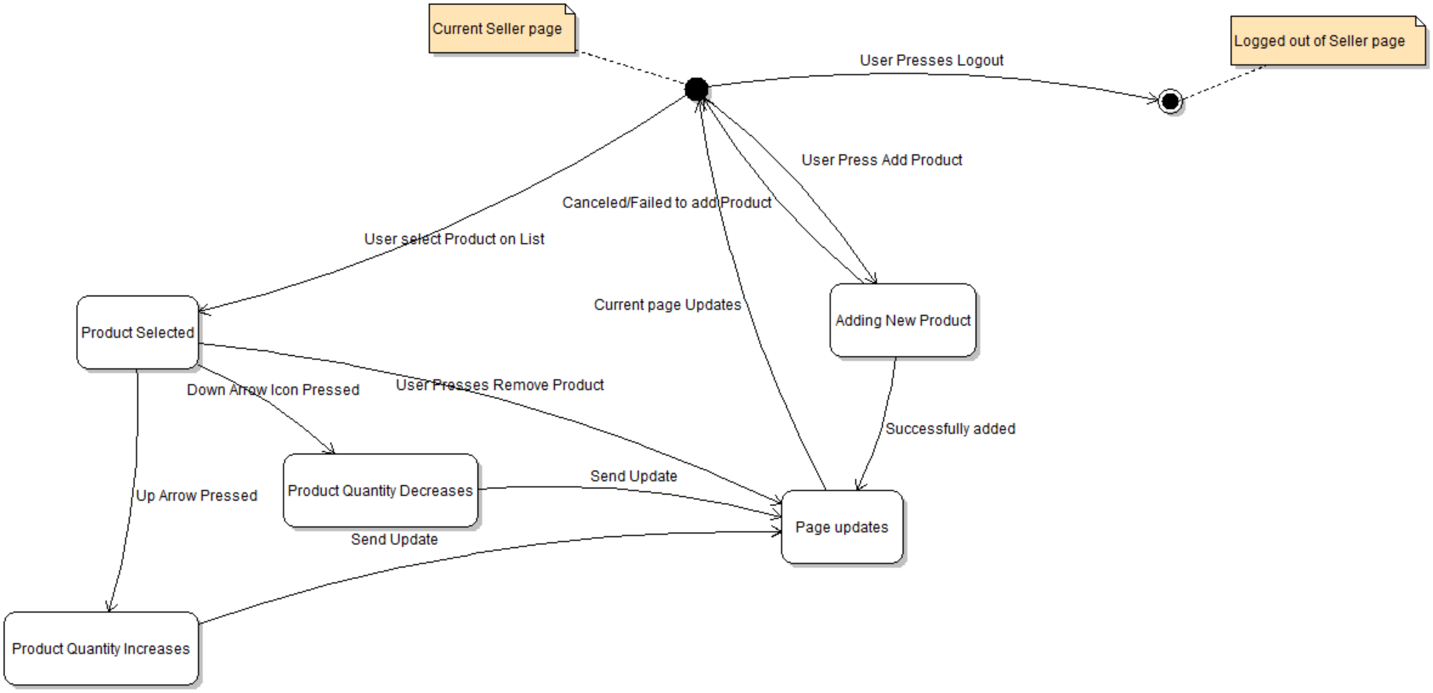

javax.swing.JFrame
«interface»
PaymentListener
finished()
CartView
listeners:List<CartListener>
inventory:StoreInventory
cart:ShoppingCart
productId:int
backButton:javax.swing.JButton
cartTable:javax.sing.JTable
checkoutButton:javax.swing.JButton
downButton:javax.swing.JButton
jLabelID:javax.swing.JLabel
jLabelPrice:javax.swing.JLabel
jLabelProductName:javax.swing.JLabel
jLabelQuantity:javax.swing.JLabel
jLabelUnitPrice:javax.swing.JLabel
totalLabel:javax.swing.JLabel
upButton:javax.swing.JButton
addListener(toAdd:CartListener)
updateShoppingCartView()
updateTotal()
populateTable()
initComponents()
backButtonActionPerformed(evt:java.awt.event.ActionEvent)
cartTableMouseClicked(evt:java.awt.event.MouseEvent)
updateUpDownButtons()
checkoutButtonActionPerformed(evt:java.awt.event.ActionEvent)
downButtonActionPerformed(evt:java.awt.event.ActionEvent)
upButtonActionPerformed(evt:java.awt.event.ActionEvent)
finished()
main(args:String [])
«interface»
CartListener
PurchaseForm
javax.swing.JButton
javax.swing.JLabel
javax.swing.JTable
Object[][]

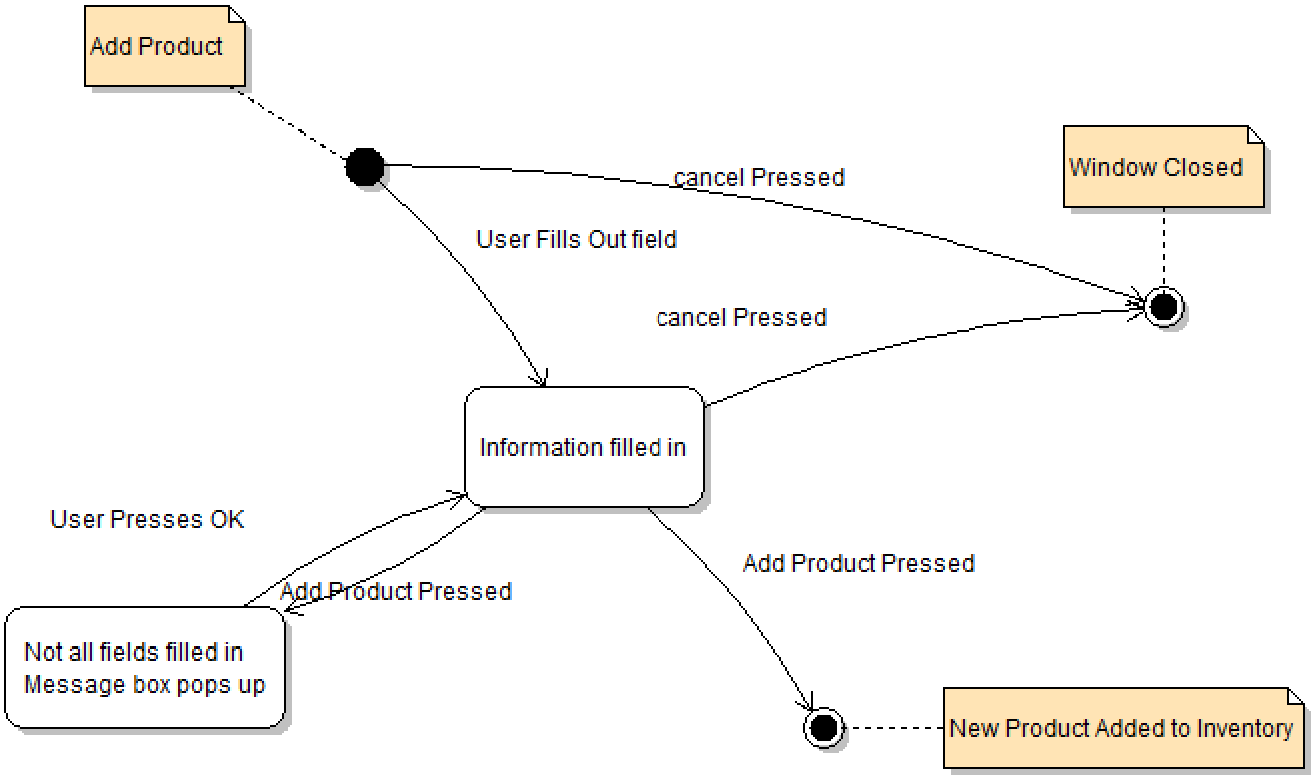

javax.swing.JFrame
«interface»
AddProductListener
newProduct()
AddNewProduct
Seller
inventory:StoreInventory
cart:ShoppingCart
sold:Sold
archive:Archive
inventoryListLabels:String[]
currentProductID:int
jPanel1:javax.swing.JPanel
descriptionTextArea:javax.swing.JTextArea
costLabel:javax.swing.JLabel
jLabel1:javax.swing.JLabel
jLabel2:javax.swing.JLabel
jLabel3:javax.swing.JLabel
jLabel5:javax.swing.JLabel
jLabel7:javax.swing.JLabel
priceLabel:javax.swing.JLabel
profitLabel:javax.swing.JLabel
quantityLabel:javax.swing.JLabel
revenueLabel:javax.swing.JLabel
titleLabel:javax.swing.JLabel
jScrollPane1:javax.swing.JScrollPane
jScrollPane2:javax.swing.JScrollPane
minusButton:javax.swing.JButton
plusButton:javax.swing.JButton
downButton:javax.swing.JButton
jButton1:javax.swing.JButton
upButton:javax.swing.JButton
productList:javax.swing.JList
updateFinancials()
updateInventoryList()
initComponents()
jButton1ActionPerformed(evt:java.awt.event.ActionEvent)
productListValueChanged(evt:javax.swing.event.ListSelectionEvent)
downButtonActionPerformed(evt:java.awt.event.ActionEvent)
upButtonActionPerformed(evt:java.awt.event.ActionEvent)
minusButtonActionPerformed(evt:java.awt.event.ActionEvent)
plusButtonActionPerformed(evt:java.awt.event.ActionEvent)
newProduct()
updateProductDetails (productID:int)
clearProductDetails()
main(args:String[])
Map<String, Object>
Product
javax.swing.JLabel
javax.swing.JPanel
javax.swing.JTextArea
javax.swing.JScrollPane
javax.swing.JButton
javax.swing.JList

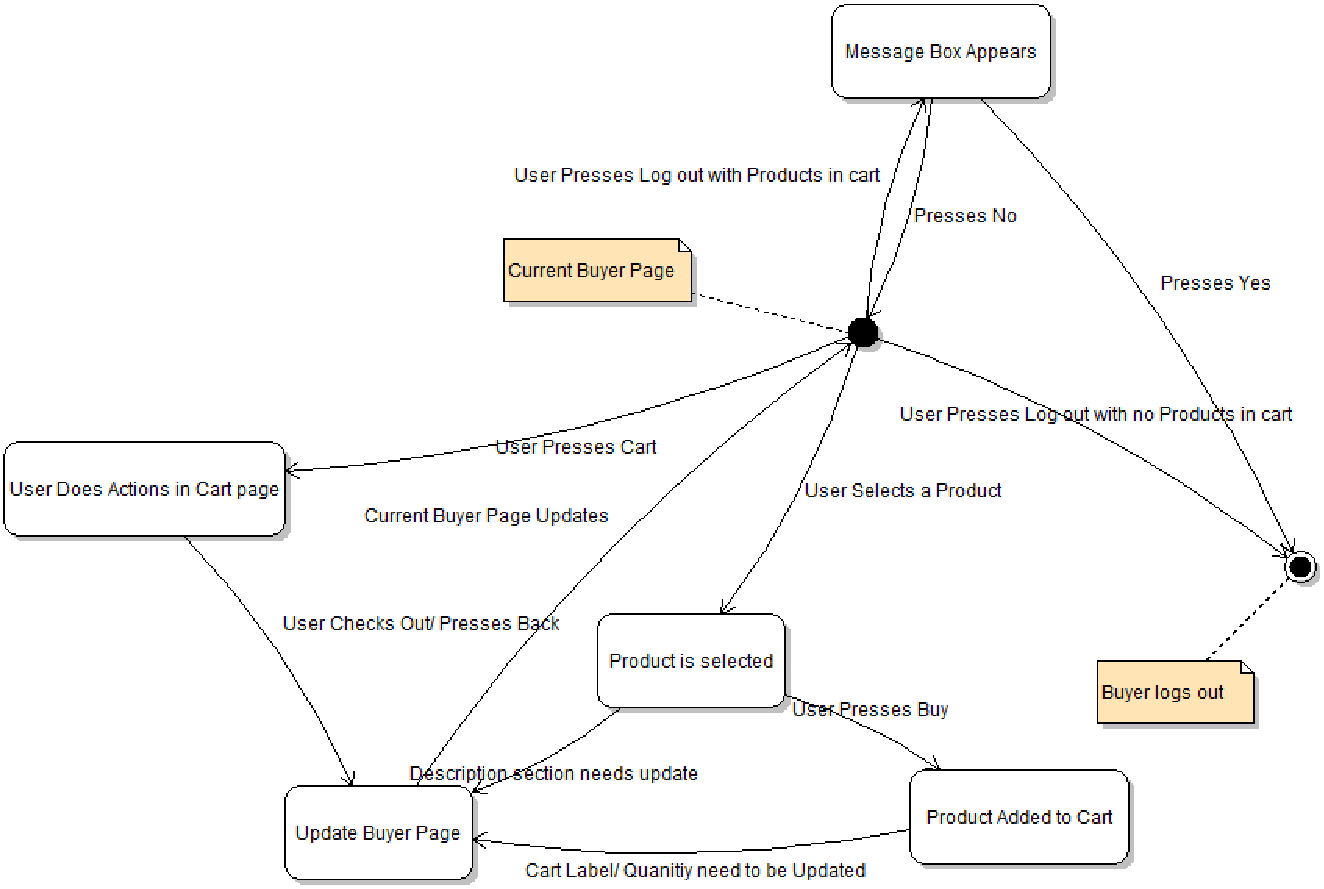

«interface»
Serializable
Revenue
revenue:double
profit:double
BufferedInputStream
FileInputStream
ObjectInput
ObjectInputStream
FileOutputStream
ObjectOutputStream
InputStream
«interface»
ArchiveAction
saveData()
loadData()
saveFinancials()
loadFinancials()
Archive
storeInventory:StoreInventory
recoveredCart:Product[]
revenueRecovered:double
profitRecovered:double
costRecovered:double
saveData()
loadData()
saveFinancials()
loadFinancials()
1
*
«interface»
Serializable
Product
+sku:int
+name:String
+ description:String
+price:double
+cost:double
quantity:int
*
1
javax.swing.JButton
javax.swing.JFrame
javax.swing.JLabel
javax.swing.JInternalFrame
javax.swing.JPasswordField
javax.swing.JTextField
MainScreen
StoreInventory inventory
ShoppingCart cart
Sold sold
Archive archive
clearButton:javax.swing.JButton
jLabel1:javax.swing.JLabel
jLabel2:javax.swing.JLabel
jLabel3:javax.swing.JLabel
loginButton: javax.swing.JButton
loginFrame:javax.swing.JInternalFrame
passwordField:javax.swing.JPasswordField
usernameField: javax.swing.JTextField
fillInventory()
initComponents()
loginButtonActionPerformed(evt:java.awt.event.ActionEvent)
clearButtonActionPerformed(evt:java.awt.event.ActionEvent)
usernameFieldActionPerformed(evt:java.awt.event.ActionEvent)
Buyer
Seller
StoreInventory
instance:storeInventory
getInstance():storeInventory
ShoppingCart
instance:shoppingCart
getInstance():shoppingCart
Sold
instance:Sold
getInstance():Sold
ProductList
-products:ArrayList<Map<String, Object>>
+ productsToLoad:ArrayList<Map<String, Object>>
+currentIteratorID:int
revenue:double
profit:double
cost:double
resetItems()
getNewProductID():int
addProduct(productToAdd:Product, quantityToAdd:int)
removeProduct(productID:int)
getCount():int
retrieveQuantity(productID:int):int
getIndexForProduct(productIDtoFind:int):int
decrement(pId:int, quantityToDecrementBy:int)
increment(pId:int, quantityToIncrementBy:int)
getProductByID (pID:int):Product
getProductID (aIndex:int):int
getListing():Object[][]
getProductListing():Product[]
setListing(aTempArray:Object[][])
getCostTotal():double
getPriceTotal():double
hasNext():boolean
hasPrevious():boolean
getNext():Map<String, Object>
getPrevious():Map<String, Object>
getIteratorCount():int
resetIterator()
Map<String, Object>
Object[][]

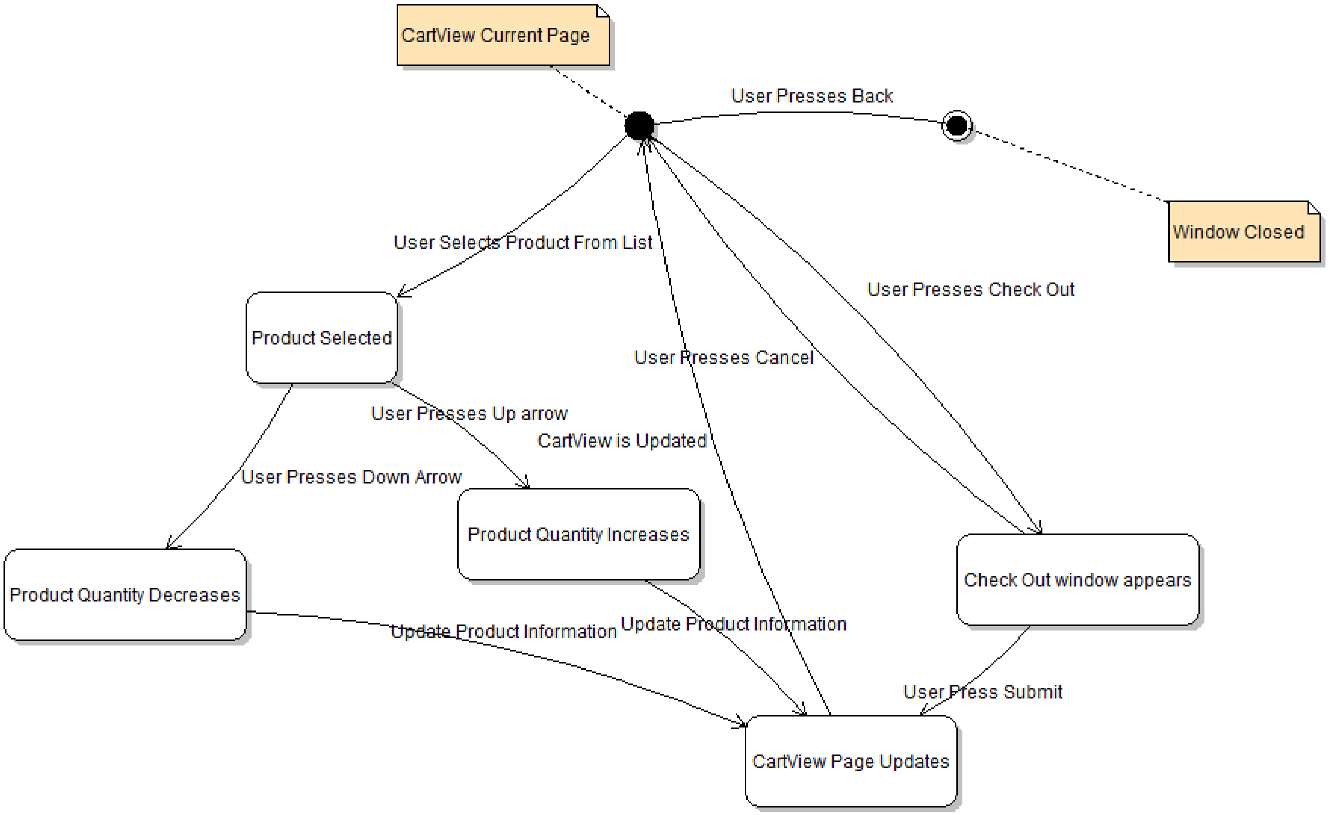
State Diagrams

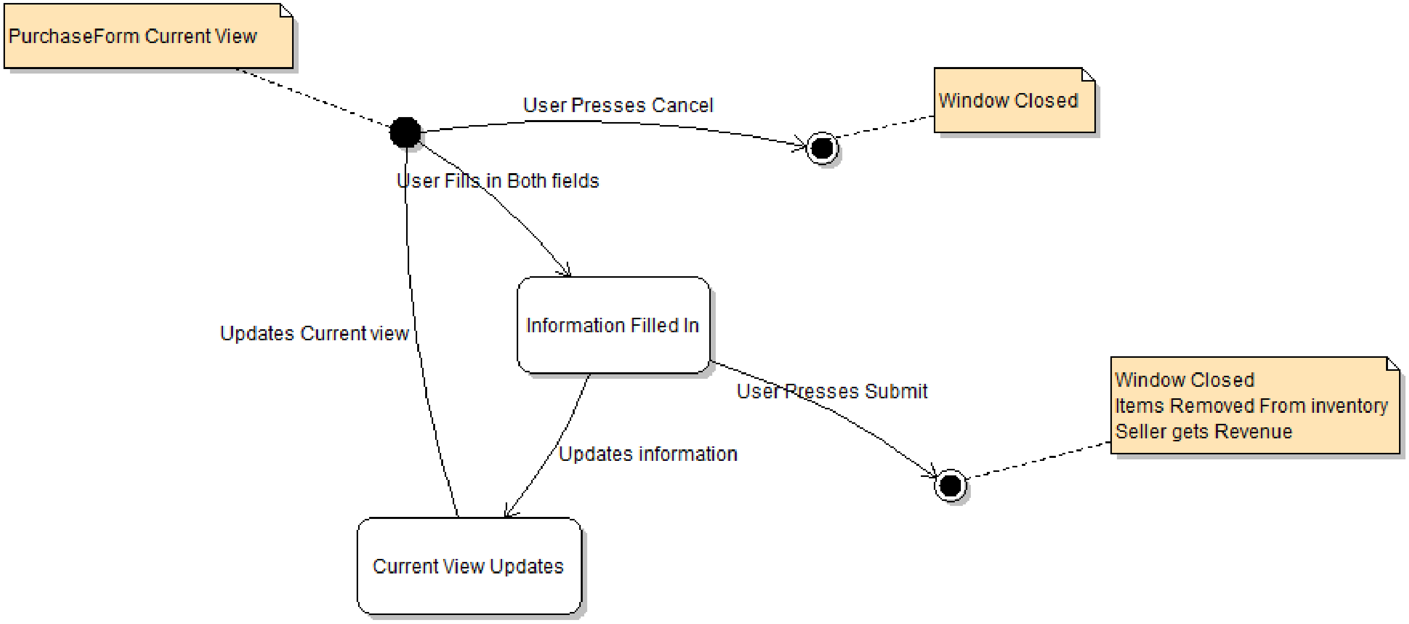

Login
User Enters information
User Presses Clear
User Presses OK
Username/Password Fields filled out
User presses login
User presses login
successful login
Information was incorrect message box appears


Current Seller page
User Presses Logout
Logged out of Seller page
User Press Add Product
Canceled/Failed to add Product
User select Product on List
Current page Updates
Product Selected
Adding New Product
Down Arrow Icon Pressed
User Presses Remove Product
Successfully added
Up Arrow Pressed
Send Update
Product Quantity Decreases
Page updates
Send Update
Product Quantity Increases

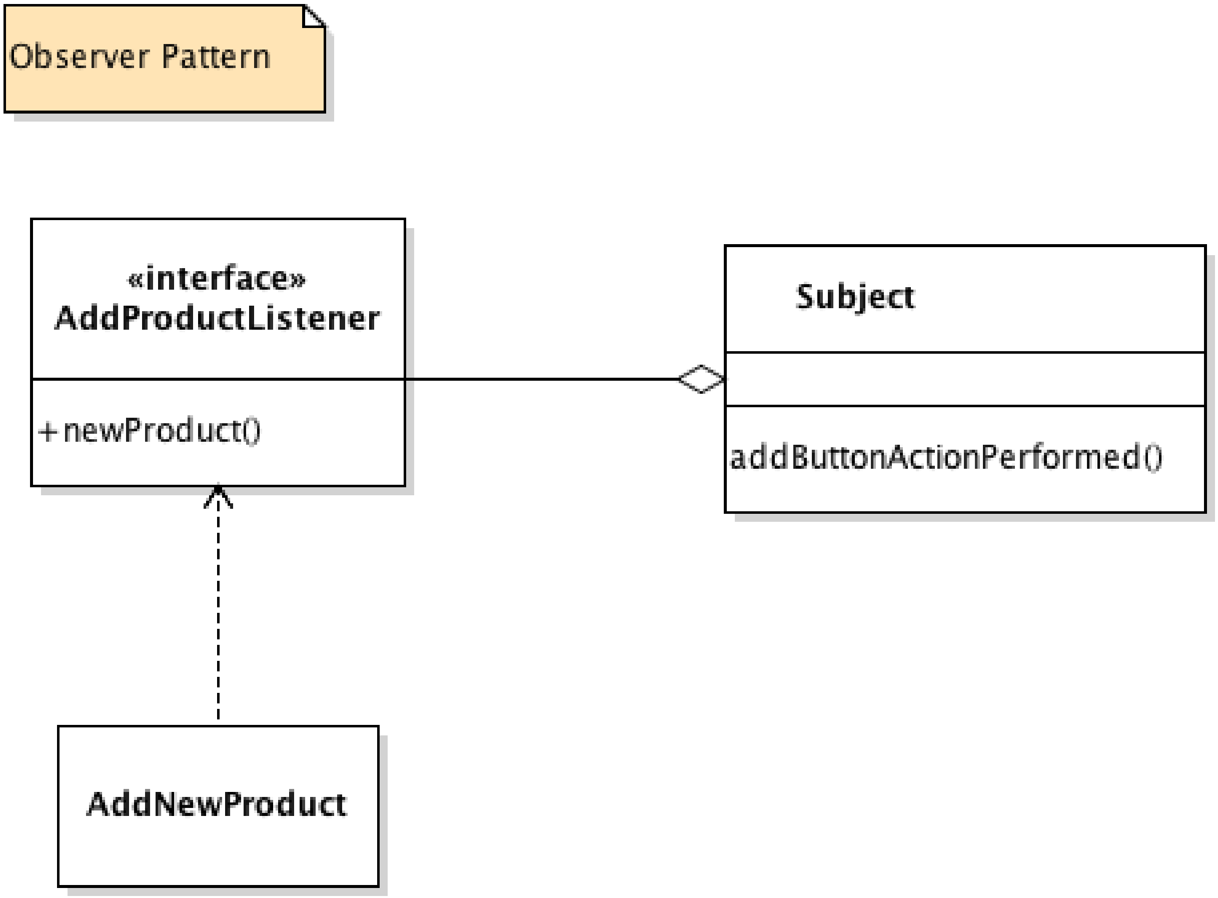

Add Product
cancel Pressed
Window Closed
User Fills Out field
cancel Pressed
Information filled in
User Presses OK
Add Product Pressed
Add Product Pressed
Not all fields filled in
Message box pops up
New Product Added to Inventory

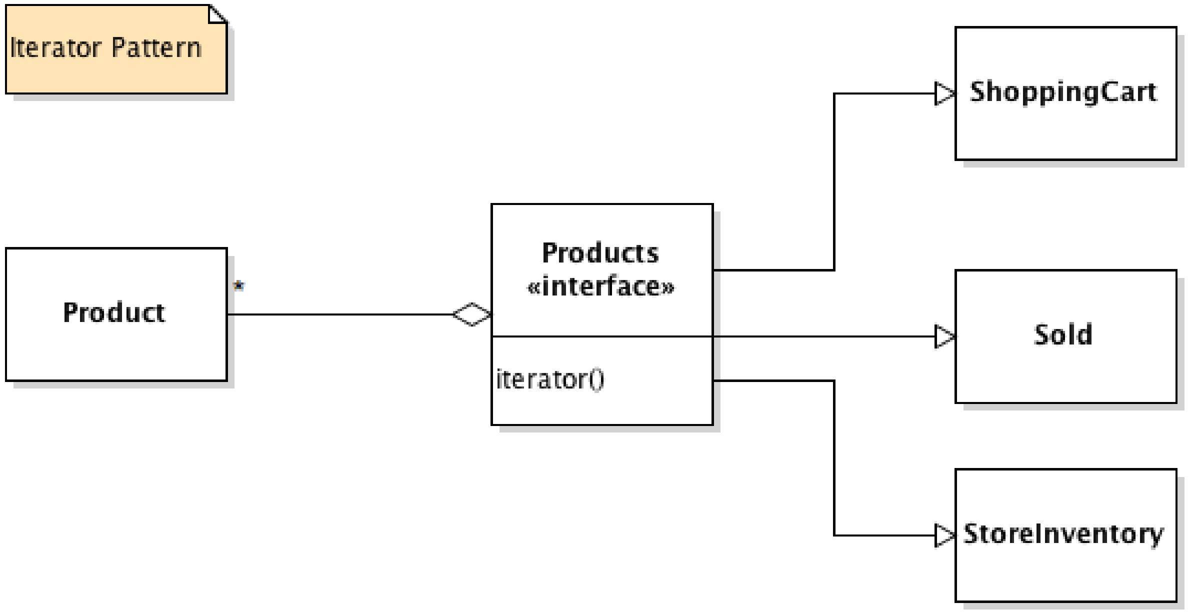

Message Box Appears
User Presses Log out with Products in cart
Presses No
Presses Yes
Current Buyer Page
User Presses Log out with no Products in cart
User Presses Cart
User Does Actions in Cart page
Current Buyer Page Updates
User Selects a Product
User Checks Out/ Presses Back
Product is selected
Buyer logs out
User Presses Buy
Description section needs update
Update Buyer Page
Product Added to Cart
Cart Label/ Quanitiy need to be Updated

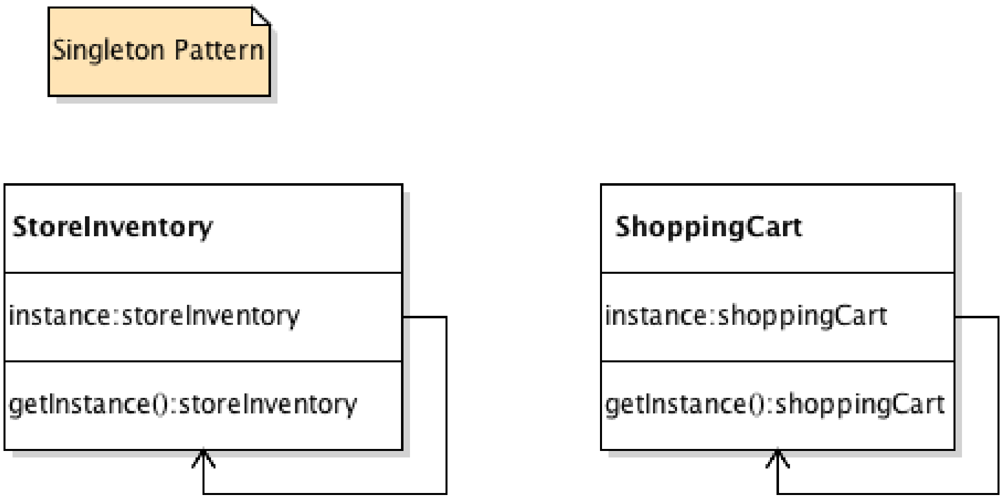


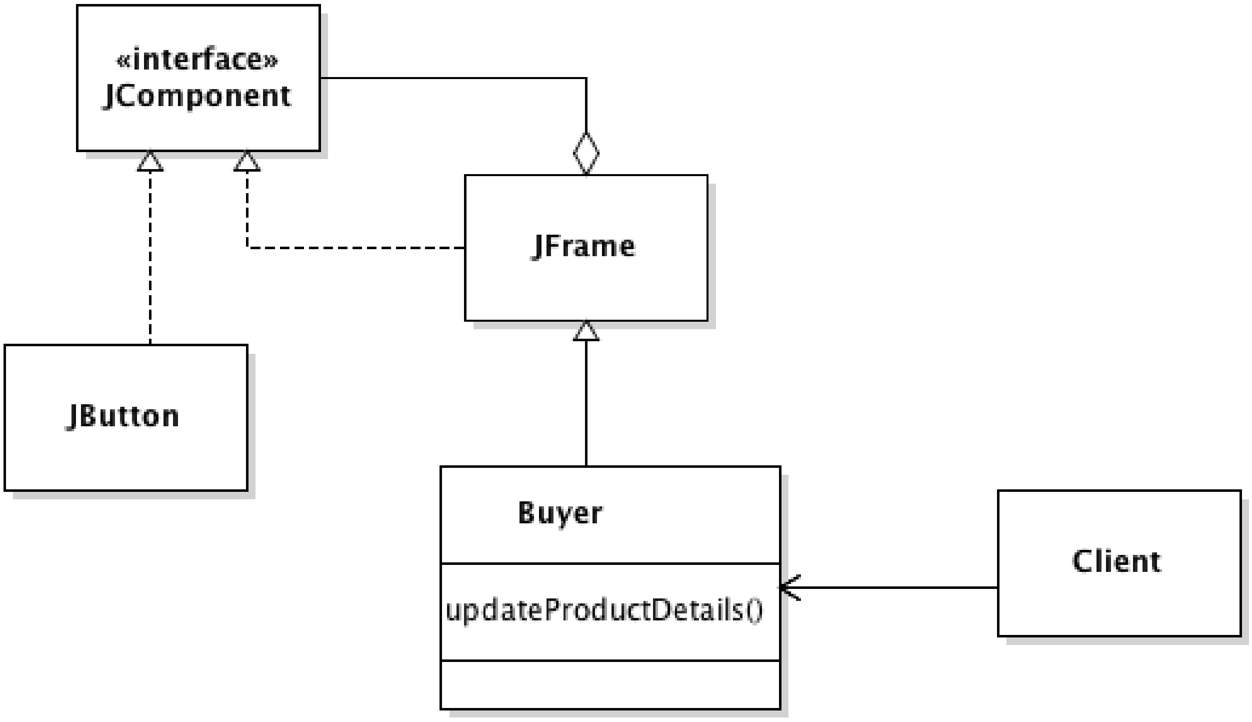


Design Patterns


Observer Pattern
«interface»
AddProductListener
+newProduct()
Subject
addButtonActionPerformed()
AddNewProduct


Iterator Pattern
Product
*
Products
«interface»
iterator()
ShoppingCart
Sold
StoreInventory


Singleton Pattern
StoreInventory
instance:storeInventory
getInstance():storeInventory
ShoppingCart
instance:shoppingCart
getInstance():shoppingCart



Source Files:

MainScreen.java

```
package com.store;


import javax.swing.*;
/**
 * Mainscreen login functionality
 * @author Michael Del Campo
 */
public class MainScreen extends javax.swing.JFrame {

    StoreInventory inventory;
    ShoppingCart cart;
    Sold sold;
    Archive archive;
    /**
     * Mainscreen Construtor grab archived items and fills
     */
    public MainScreen() {
        initComponents();
        inventory = StoreInventory.getInstance();
        cart = ShoppingCart.getCart();
        sold = Sold.getInstance();
        this.archive = new Archive();
        fillInventory();
    }
    /**
     * Fills inventory local fill is overwritten due to archived data.
     */
    private void fillInventory() {
        archive.loadData();
        /*Product test = new Product();
        test.sku = 0;
        test.cost = 5;
        test.name = "Thomas";
        test.price = 6;
        test.description = "blah";

        inventory.addItem(test,3);
        archive.saveData();*/
        /*inventory.revenue = 60.00;
        inventory.profit = 30.00;
        inventory.cost = 20.00;
        archive.saveFinancials();*/
        archive.loadFinancials();



    }
    /**
    * Build the components for the main screen
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        loginFrame = new javax.swing.JInternalFrame();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        usernameField = new javax.swing.JTextField();
        passwordField = new javax.swing.JPasswordField();
        loginButton = new javax.swing.JButton();
        clearButton = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Store");
        setName("homeFrame"); // NOI18N
        setResizable(false);

        loginFrame.setVisible(true);

        jLabel1.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
        jLabel1.setText("Account Login");

        jLabel2.setText("Username");

        jLabel3.setText("Password");

        usernameField.setToolTipText("");
        usernameField.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                usernameFieldActionPerformed(evt);
            }
        });

        loginButton.setText("Login");
        loginButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                loginButtonActionPerformed(evt);
            }
        });

        clearButton.setText("Clear");
        clearButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                clearButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout loginFrameLayout = new
javax.swing.GroupLayout(loginFrame.getContentPane());
        loginFrame.getContentPane().setLayout(loginFrameLayout);
        loginFrameLayout.setHorizontalGroup(

loginFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(loginFrameLayout.createSequentialGroup()
                .addGap(35, 35, 35)

.addGroup(loginFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING, false)
                    .addGroup(loginFrameLayout.createSequentialGroup()
                        .addComponent(loginButton)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 80,
Short.MAX_VALUE)
                        .addComponent(clearButton))
                    .addGroup(loginFrameLayout.createSequentialGroup()

.addGroup(loginFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                            .addComponent(jLabel2)
                            .addComponent(jLabel3))
                        .addGap(26, 26, 26)

.addGroup(loginFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING, false)
                            .addComponent(usernameField,
javax.swing.GroupLayout.DEFAULT_SIZE, 120, Short.MAX_VALUE)
                            .addComponent(passwordField))))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
loginFrameLayout.createSequentialGroup()
                .addContainerGap(74, Short.MAX_VALUE)
                .addComponent(jLabel1)
                .addGap(72, 72, 72))
        );
        loginFrameLayout.setVerticalGroup(

loginFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(loginFrameLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 25,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(loginFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(usernameField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(loginFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
```

```
                    .addComponent(jLabel3)
                    .addComponent(passwordField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
27, Short.MAX_VALUE)

.addGroup(loginFrameLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(loginButton)
                    .addComponent(clearButton))
                .addGap(25, 25, 25))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(30, 30, 30)
                .addComponent(loginFrame, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(42, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(loginFrame, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(26, 26, 26))
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>//GEN-END:initComponents

    private void usernameFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_usernameFieldActionPerformed
    }//GEN-LAST:event_usernameFieldActionPerformed
    private void clearButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_clearButtonActionPerformed
        usernameField.setText("");
        passwordField.setText("");
    }//GEN-LAST:event_clearButtonActionPerformed

    private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_loginButtonActionPerformed
        // TODO add your handling code here:
        String uname = usernameField.getText();

        String pwd = passwordField.getText();
        //if (username.equals("buyer")) {
        if (uname.equals("b")) {
            //if (password.equals("password")) {
            if (pwd.equals("")) {
                Buyer buyer = new Buyer();
                buyer.setVisible(true);
                usernameField.setText("");
                passwordField.setText("");
            }
            else {
                JOptionPane.showMessageDialog(null, "Invalid Password", "Error", JOptionPane.ERROR_MESSAGE);
                passwordField.setText("");
            }
        }
        //else if (username.equals("seller")) {
        else if (uname.equals("s")) {
            //if (password.equals("password")) {
            if (pwd.equals("")) {
                Seller seller = new Seller();

                seller.setVisible(true);

                usernameField.setText("");

                passwordField.setText("");
            }
            else {
                JOptionPane.showMessageDialog(null, "Invalid Password", "Error", JOptionPane.ERROR_MESSAGE);
                passwordField.setText("");
            }
        }
        else
        {
            JOptionPane.showMessageDialog(null, "Invalid Username", "Error",
JOptionPane.ERROR_MESSAGE);

        }
    }//GEN-LAST:event_loginButtonActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look
and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MainScreen.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MainScreen.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MainScreen.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MainScreen.class.getName()).log(java.util.logging.Le
vel.SEVERE, null, ex);
        }
        //</editor-fold>
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new MainScreen().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton clearButton;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JButton loginButton;
    private javax.swing.JInternalFrame loginFrame;
    private javax.swing.JPasswordField passwordField;
    private javax.swing.JTextField usernameField;
    // End of variables declaration//GEN-END:variables
}
```

AddNewProduct.java

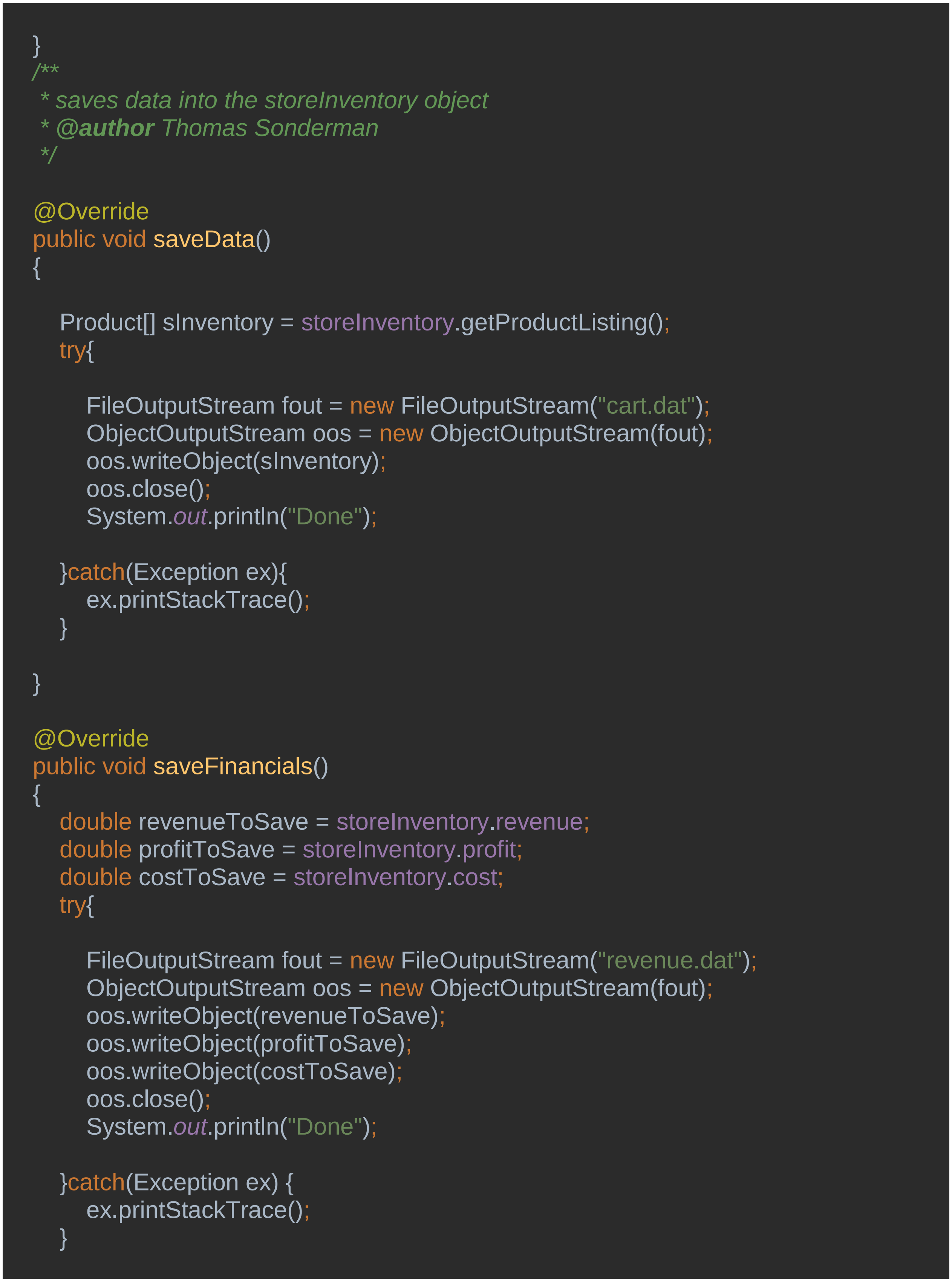
```
package com.store;

import javax.swing.*;
import java.util.ArrayList;
import java.util.List;

/**
 * Add Product Listener Interface
 * @author Thomas Sonderman
 */
interface AddProductListener {
    public void newProduct();
}

public class AddNewProduct extends javax.swing.JFrame {

    /**
     * Creates new form AddItem
     * @author Thomas Sonderman
     */
    public AddNewProduct() {
        initComponents();
        inventory = StoreInventory.getInstance();
    }
    /**
     * add Listener class for chagnes
     * @param toAdd
     */
    public void addListener(AddProductListener toAdd) {
        listeners.add(toAdd);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        titleField = new javax.swing.JTextField();
        costField = new javax.swing.JTextField();
        priceField = new javax.swing.JTextField();
        quantityField = new javax.swing.JTextField();
        jScrollPane1 = new javax.swing.JScrollPane();
        descriptionField = new javax.swing.JTextArea();
        addButton = new javax.swing.JButton();
        cancelButton = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Lucida Grande", 0, 17)); // NOI18N

        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

        jLabel1.setText("Add Product");

        jLabel2.setText("Product Name");

        jLabel3.setText("Description");
        jLabel4.setText("Cost");

        jLabel5.setText("Price");

        jLabel6.setText("Quantity");

        descriptionField.setColumns(20);
        descriptionField.setRows(5);
        jScrollPane1.setViewportView(descriptionField);

        cancelButton.setText("Cancel");
        addButton.setText("Add Product");

        addButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                addButtonActionPerformed(evt);
            }
        });
        cancelButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(layout.createSequentialGroup()
                                    .addContainerGap()
                                    .addComponent(jLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                            .addGroup(layout.createSequentialGroup()
                                    .addGap(22, 22, 22)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                                                    .addComponent(jLabel2)
                                                    .addGap(27, 27, 27))
```

```
                      .addGroup(layout.createSequentialGroup()
                          .addComponent(jLabel3)
                          .addGap(41, 41, 41)))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                      .addComponent(titleField,
javax.swing.GroupLayout.DEFAULT_SIZE, 270, Short.MAX_VALUE)
                      .addComponent(jScrollPane1))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                      .addComponent(jLabel5)
                      .addComponent(jLabel4)
                      .addComponent(jLabel6)
                      .addComponent(addButton,
javax.swing.GroupLayout.Alignment.TRAILING))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                      .addComponent(quantityField,
javax.swing.GroupLayout.Alignment.TRAILING)
                      .addComponent(costField)
                      .addComponent(priceField)
                      .addGroup(layout.createSequentialGroup()
                          .addComponent(cancelButton)
                          .addGap(1, 1, 1)))))
          .addContainerGap())
    );
    layout.setVerticalGroup(
      layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(layout.createSequentialGroup()
          .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
          .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
35, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
              .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(jLabel2)
                                        .addComponent(titleField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(jLabel3)
                                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(jLabel4)
                                        .addComponent(costField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(jLabel5)
                                        .addComponent(priceField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(jLabel6)
                                        .addComponent(quantityField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(cancelButton)
                                        .addComponent(addButton))))
                .addGap(29, 29, 29))
        );
```

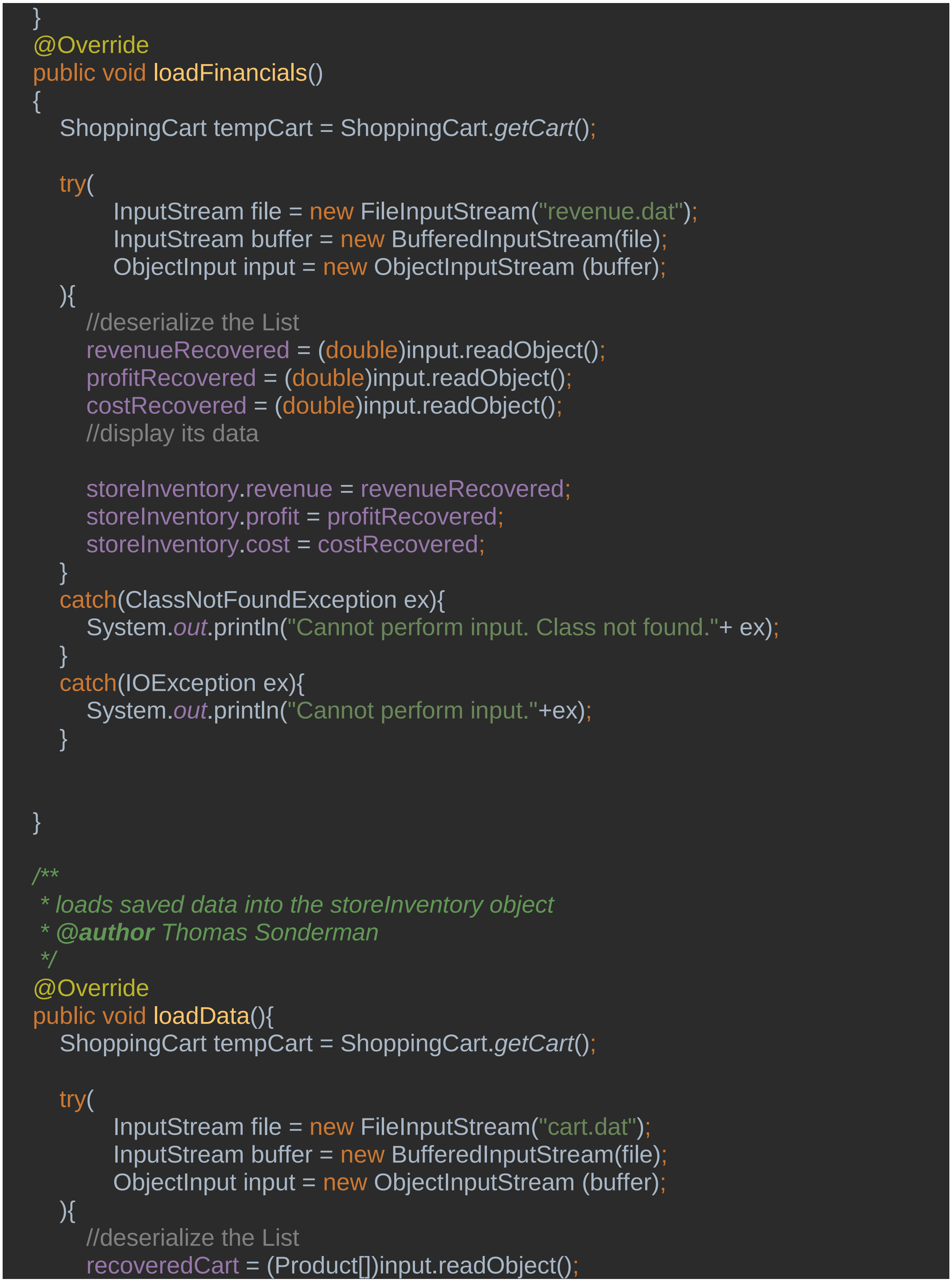
```java
        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>//GEN-END:initComponents

    private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_addButtonActionPerformed
        if (titleField.getText().length() > 0 && descriptionField.getText().length() > 0 && costField.getText().length() > 0 && priceField.getText().length() > 0 && quantityField.getText().length() > 0) {
            if (!isDouble(costField.getText())) {
                JOptionPane.showMessageDialog(null, "Cost invalid.", "Problem", JOptionPane.ERROR_MESSAGE);
            }
            else if (!isDouble(priceField.getText())) {
                JOptionPane.showMessageDialog(null, "Price invalid.", "Problem", JOptionPane.ERROR_MESSAGE);
            }
            else if (!isInteger(quantityField.getText())) {
                JOptionPane.showMessageDialog(null, "Quantity invalid.", "Problem", JOptionPane.ERROR_MESSAGE);
            }
            else {
                String title = titleField.getText();
                String description = descriptionField.getText();
                double cost = Double.parseDouble(costField.getText());
                double price = Double.parseDouble(priceField.getText());
                int quantity = Integer.parseInt(quantityField.getText());
                Product product = new Product();
                product.sku = -1;
                product.name = title;
                product.description = description;
                product.cost = cost;
                product.price = price;
                product.quantity = quantity;
                inventory.addProduct(product, quantity);
                for (AddProductListener hl : listeners)
                    hl.newProduct();
                this.setVisible(false);
            }
        }
        else {
            JOptionPane.showMessageDialog(null, "All fields are required.", "Problem", JOptionPane.ERROR_MESSAGE);
        }
    }//GEN-LAST:event_addButtonActionPerformed
    private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_cancelButtonActionPerformed
        this.setVisible(false);
    }//GEN-LAST:event_cancelButtonActionPerformed

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
                java.util.logging.Logger.getLogger(AddNewProduct.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
                java.util.logging.Logger.getLogger(AddNewProduct.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
                java.util.logging.Logger.getLogger(AddNewProduct.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
                java.util.logging.Logger.getLogger(AddNewProduct.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new AddNewProduct().setVisible(true);
            }
        });
    }

    /**
     * Check if string value formats to Double
     * @param valueToCheck string to convert to Double
     */
    private boolean isDouble(String valueToCheck)
    {
        try
        {
            Double.parseDouble(valueToCheck);
            return true;
        }
        catch (Exception ex)
        {
            return false;
        }
    }
    /**
     * private function to check if string converts to Integer
     * @param valueToCheck string to convert to Double
     */
    private boolean isInteger(String valueToCheck)
    {
        try
        {
            Integer.parseInt(valueToCheck);
            return true;
        }
        catch (Exception ex) {
            return false;
        }
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton addButton;
    private javax.swing.JButton cancelButton;
    private javax.swing.JTextField costField;
    private javax.swing.JTextArea descriptionField;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextField priceField;
    private javax.swing.JTextField quantityField;
    private javax.swing.JTextField titleField;
    // End of variables declaration//GEN-END:variables
    List<AddProductListener> listeners = new ArrayList<>();

    StoreInventory inventory;
}
```

Archive.java

```
package com.store;

import java.io.*;
import java.util.List;

/**
 * Archive Interface to be called on buyer seller pages
 * @author Thomas Sonderman
 */
interface ArchiveAction {
    public void saveData();
    public void loadData();
    public void saveFinancials();
    public void loadFinancials();
}
/**
 * Archive class that implements Archive Action
 * @author Thomas Sonderman
 */
public class Archive implements ArchiveAction{
    private StoreInventory storeInventory;
    public Product[] recoveredCart;
    public double revenueRecovered;
    public double profitRecovered;
    public double costRecovered;
    public Archive()
    {
        this.storeInventory = StoreInventory.getInstance();
```



```java
}
/**
 * saves data into the storeInventory object
 * @author Thomas Sonderman
 */

@Override
public void saveData()
{

    Product[] sInventory = storeInventory.getProductListing();
    try{

        FileOutputStream fout = new FileOutputStream("cart.dat");
        ObjectOutputStream oos = new ObjectOutputStream(fout);
        oos.writeObject(sInventory);
        oos.close();
        System.out.println("Done");

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

}

@Override
public void saveFinancials()
{
    double revenueToSave = storeInventory.revenue;
    double profitToSave = storeInventory.profit;
    double costToSave = storeInventory.cost;
    try{

        FileOutputStream fout = new FileOutputStream("revenue.dat");
        ObjectOutputStream oos = new ObjectOutputStream(fout);
        oos.writeObject(revenueToSave);
        oos.writeObject(profitToSave);
        oos.writeObject(costToSave);
        oos.close();
        System.out.println("Done");

    }catch(Exception ex) {
        ex.printStackTrace();
    }
```

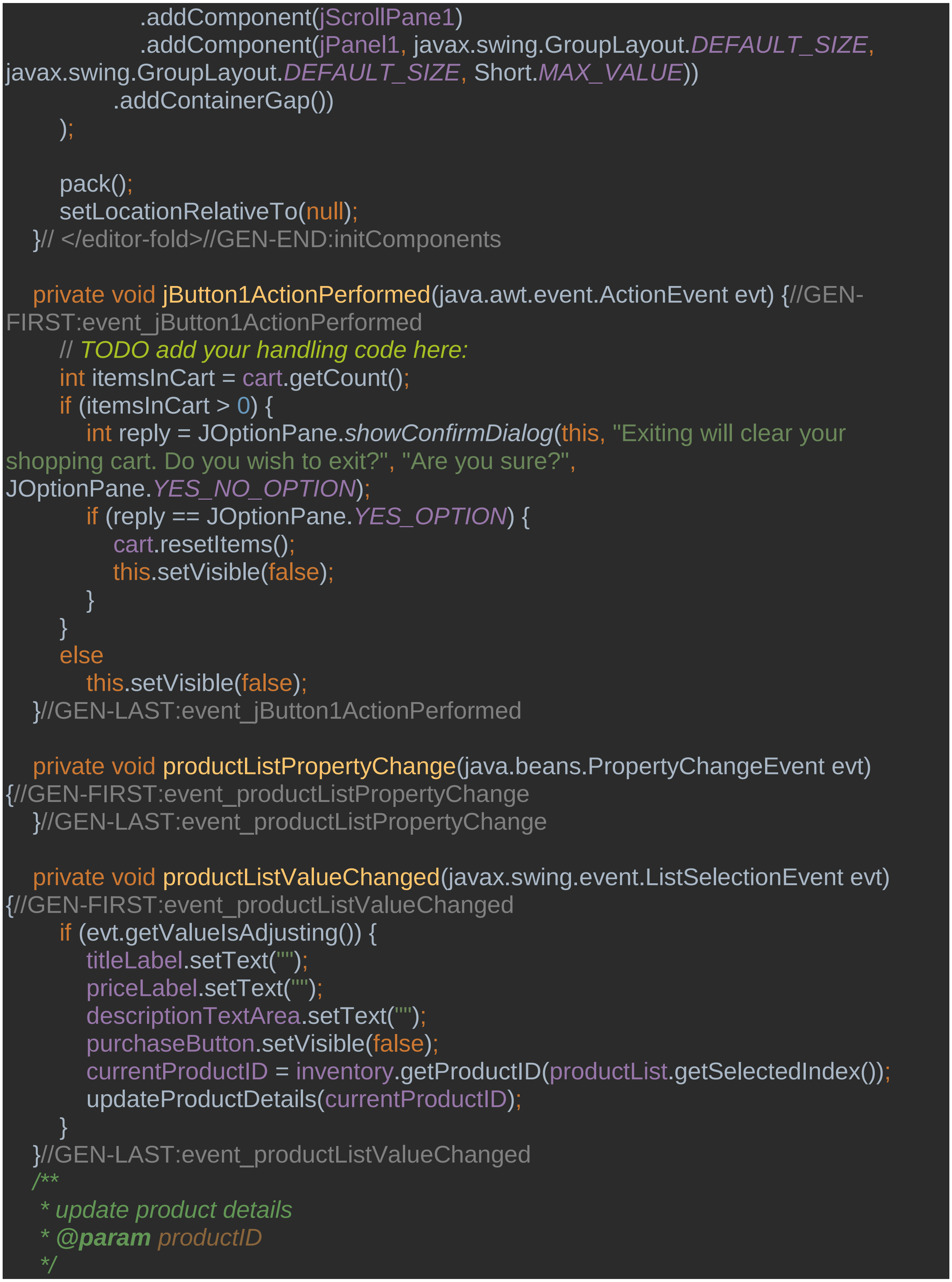


```
}
@Override
public void loadFinancials()
{
    ShoppingCart tempCart = ShoppingCart.getCart();

    try(
            InputStream file = new FileInputStream("revenue.dat");
            InputStream buffer = new BufferedInputStream(file);
            ObjectInput input = new ObjectInputStream (buffer);
    ){
        //deserialize the List
        revenueRecovered = (double)input.readObject();
        profitRecovered = (double)input.readObject();
        costRecovered = (double)input.readObject();
        //display its data

        storeInventory.revenue = revenueRecovered;
        storeInventory.profit = profitRecovered;
        storeInventory.cost = costRecovered;
    }
    catch(ClassNotFoundException ex){
        System.out.println("Cannot perform input. Class not found."+ ex);
    }
    catch(IOException ex){
        System.out.println("Cannot perform input."+ex);
    }


}

/**
 * loads saved data into the storeInventory object
 * @author Thomas Sonderman
 */
@Override
public void loadData(){
    ShoppingCart tempCart = ShoppingCart.getCart();

    try(
            InputStream file = new FileInputStream("cart.dat");
            InputStream buffer = new BufferedInputStream(file);
            ObjectInput input = new ObjectInputStream (buffer);
    ){
        //deserialize the List
        recoveredCart = (Product[])input.readObject();
```

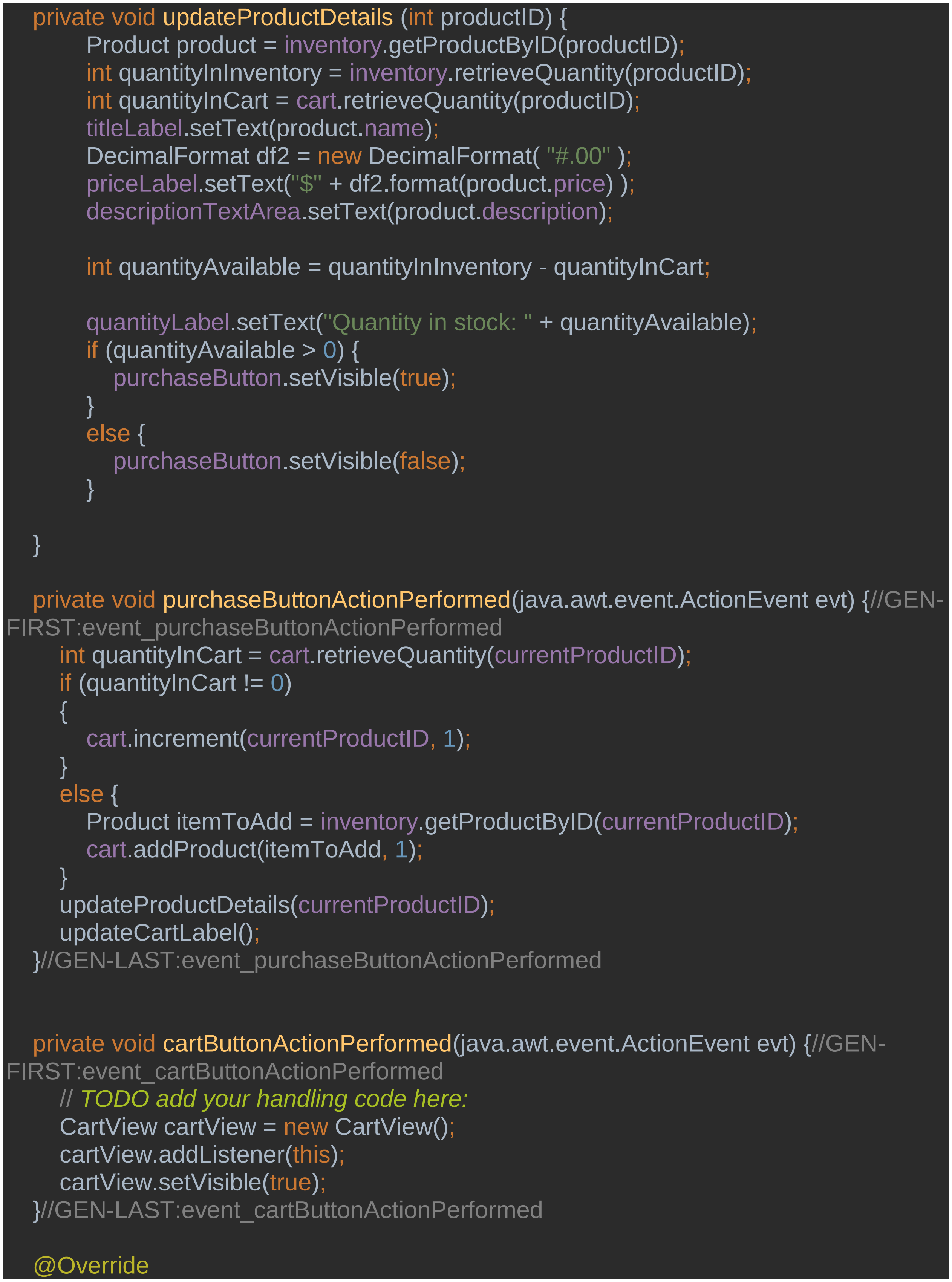
```
            //display its data
            int skuCounter = 0;
            for(Product o: recoveredCart) {

                o.sku = skuCounter;
                storeInventory.addProduct(o,o.quantity);
                skuCounter++;
            }
        }
        catch(ClassNotFoundException ex){
            System.out.println("Cannot perform input. Class not found."+ ex);
        }
        catch(IOException ex){
            System.out.println("Cannot perform input."+ex);
        }
    };

}
```

Buyer.java

```
package com.store;


import javax.swing.*;
import java.text.DecimalFormat;
import java.util.Map;
/**
 *Buyer Class that consumes seller items
 * @author Michael Del Campo
 */
public class Buyer extends javax.swing.JFrame implements CartListener {

    StoreInventory inventory;
    ShoppingCart cart;

    String[] inventoryText;

    int currentProductID;
    /**
     * Buyer Constructor
     */
    public Buyer() {
        initComponents();

        inventory = StoreInventory.getInstance();

        cart = ShoppingCart.getCart();

        updateInventoryList();

        currentProductID = 0;
    }
    /**
     * update inventory of store list
     */
    private void updateInventoryList() {
        String[] TextLabels = new String[inventory.getIteratorCount()];
        int iterator = 0;
        inventory.resetIterator();
        while (inventory.hasNext()) {
            Map<String, Object> inventoryItem = inventory.getNext();
            DecimalFormat df2 = new DecimalFormat( "#0.00" );
            Product product = (Product)inventoryItem.get("item");

            TextLabels[iterator] = product.name + " - $" + df2.format(product.price);
            iterator++;
        }
        inventory.resetIterator();

        productList.removeAll();
        productList.setModel(new javax.swing.AbstractListModel() {
            String[] strings = TextLabels;
            @Override
            public int getSize() { return strings.length; }
            @Override
            public Object getElementAt(int i) { return strings[i]; }
        });

        purchaseButton.setVisible(false);
    }
    /**
     * Build the componenents for the page
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        jButton1 = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        productList = new javax.swing.JList();
        jPanel1 = new javax.swing.JPanel();
        titleLabel = new javax.swing.JLabel();
        jScrollPane2 = new javax.swing.JScrollPane();
        descriptionTextArea = new javax.swing.JTextArea();
        priceLabel = new javax.swing.JLabel();
        purchaseButton = new javax.swing.JButton();
        quantityLabel = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        cartButton = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
        setPreferredSize(new java.awt.Dimension(672, 520));
        addFocusListener(new java.awt.event.FocusAdapter() {
            public void focusGained(java.awt.event.FocusEvent evt) {
                formFocusGained(evt);
            }
        });

        jButton1.setText("Log Out");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        productList.setToolTipText("");
        productList.setPreferredSize(new java.awt.Dimension(260, 411));
        productList.addPropertyChangeListener(new java.beans.PropertyChangeListener()
{
            public void propertyChange(java.beans.PropertyChangeEvent evt) {
                productListPropertyChange(evt);
            }
        });
        productList.addListSelectionListener(new javax.swing.event.ListSelectionListener()
{
            public void valueChanged(javax.swing.event.ListSelectionEvent evt) {
                productListValueChanged(evt);
            }
        });
        jScrollPane1.setViewportView(productList);

        jPanel1.setBackground(new java.awt.Color(255, 255, 255));
        titleLabel.setFont(new java.awt.Font("Lucida Grande", 0, 24)); // NOI18N

        descriptionTextArea.setColumns(20);
        descriptionTextArea.setRows(5);
        jScrollPane2.setViewportView(descriptionTextArea);

        priceLabel.setFont(new java.awt.Font("Lucida Grande", 0, 24)); // NOI18N
        priceLabel.setText(" ");

        purchaseButton.setText("Buy");
        purchaseButton.setToolTipText("");
        purchaseButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                purchaseButtonActionPerformed(evt);
            }
        });

        quantityLabel.setFont(new java.awt.Font("Lucida Grande", 0, 24)); // NOI18N
        quantityLabel.setText(" ");

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addComponent(jScrollPane2)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(titleLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 240,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(priceLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addGap(12, 12, 12))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                        .addComponent(quantityLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 251,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(purchaseButton)))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING, false)
                    .addComponent(priceLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(titleLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(18, 18, 18)
                .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE,
315, Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addComponent(purchaseButton)
                    .addComponent(quantityLabel))
                .addContainerGap())
        );

        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("Products");

        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel2.setText("Details");

        cartButton.setText("Cart - 0 Items ($0.00)");
        cartButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cartButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                            .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 275, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addGap(0, 0, Short.MAX_VALUE)
                        .addComponent(cartButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jButton1)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton1)
                    .addComponent(cartButton))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
12, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jLabel2))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
```



```
                    .addComponent(jScrollPane1)
                    .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap())
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>//GEN-END:initComponents

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton1ActionPerformed
        // TODO add your handling code here:
        int itemsInCart = cart.getCount();
        if (itemsInCart > 0) {
            int reply = JOptionPane.showConfirmDialog(this, "Exiting will clear your
shopping cart. Do you wish to exit?", "Are you sure?",
JOptionPane.YES_NO_OPTION);
            if (reply == JOptionPane.YES_OPTION) {
                cart.resetItems();
                this.setVisible(false);
            }
        }
        else
            this.setVisible(false);
    }//GEN-LAST:event_jButton1ActionPerformed

    private void productListPropertyChange(java.beans.PropertyChangeEvent evt)
{//GEN-FIRST:event_productListPropertyChange
    }//GEN-LAST:event_productListPropertyChange

    private void productListValueChanged(javax.swing.event.ListSelectionEvent evt)
{//GEN-FIRST:event_productListValueChanged
        if (evt.getValueIsAdjusting()) {
            titleLabel.setText("");
            priceLabel.setText("");
            descriptionTextArea.setText("");
            purchaseButton.setVisible(false);
            currentProductID = inventory.getProductID(productList.getSelectedIndex());
            updateProductDetails(currentProductID);
        }
    }//GEN-LAST:event_productListValueChanged
    /**
     * update product details
     * @param productID
     */
```



```
    private void updateProductDetails (int productID) {
            Product product = inventory.getProductByID(productID);
            int quantityInInventory = inventory.retrieveQuantity(productID);
            int quantityInCart = cart.retrieveQuantity(productID);
            titleLabel.setText(product.name);
            DecimalFormat df2 = new DecimalFormat( "#.00" );
            priceLabel.setText("$" + df2.format(product.price) );
            descriptionTextArea.setText(product.description);

            int quantityAvailable = quantityInInventory - quantityInCart;

            quantityLabel.setText("Quantity in stock: " + quantityAvailable);
            if (quantityAvailable > 0) {
                purchaseButton.setVisible(true);
            }
            else {
                purchaseButton.setVisible(false);
            }

    }

    private void purchaseButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_purchaseButtonActionPerformed
        int quantityInCart = cart.retrieveQuantity(currentProductID);
        if (quantityInCart != 0)
        {
            cart.increment(currentProductID, 1);
        }
        else {
            Product itemToAdd = inventory.getProductByID(currentProductID);
            cart.addProduct(itemToAdd, 1);
        }
        updateProductDetails(currentProductID);
        updateCartLabel();
    }//GEN-LAST:event_purchaseButtonActionPerformed


    private void cartButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_cartButtonActionPerformed
        // TODO add your handling code here:
        CartView cartView = new CartView();
        cartView.addListener(this);
        cartView.setVisible(true);
    }//GEN-LAST:event_cartButtonActionPerformed

    @Override
    public void cartExited() {
        updateInventoryList();
        updateCartLabel();
    }

    private void formFocusGained(java.awt.event.FocusEvent evt) {//GEN-
FIRST:event_formFocusGained
    }//GEN-LAST:event_formFocusGained

    private void updateCartLabel () {

        int items = cart.getCount();

        double total = cart.getPriceTotal();

        DecimalFormat df2 = new DecimalFormat("#.00");
        cartButton.setText("Cart - " + items + " Items ($" + df2.format(total) + ")");
    }


    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Buyer.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Buyer.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Buyer.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Buyer.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        }
        //</editor-fold>

        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Buyer().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton cartButton;
    private javax.swing.JTextArea descriptionTextArea;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JLabel priceLabel;
    private javax.swing.JList productList;
    private javax.swing.JButton purchaseButton;
    private javax.swing.JLabel quantityLabel;
    private javax.swing.JLabel titleLabel;
    // End of variables declaration//GEN-END:variables
}
```

CartView.java

```
package com.store;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.List;

interface CartListener {
    public void cartExited();
}

/**
 * CartView Class and Listener
 */
public class CartView extends javax.swing.JFrame implements PaymentListener {
```

```
List<CartListener> listeners = new ArrayList<CartListener>();

StoreInventory inventory;
ShoppingCart cart;
int productId;

public CartView() {
    initComponents();
    inventory = StoreInventory.getInstance();
    cart = ShoppingCart.getCart();
    productId = 0;
    updateShoppingCartView();
}

public void addListener(CartListener toAdd) {
    listeners.add(toAdd);
}

private void updateShoppingCartView() {
    populateTable();
    updateTotal();
    if (cartTable.getRowCount() > 0)
        checkoutButton.setEnabled(true);
    else
        checkoutButton.setEnabled(false);

}

private void updateTotal() {
    double total = cart.getPriceTotal();
    DecimalFormat df2 = new DecimalFormat("#0.00");
    String totalString = "Total: $" + df2.format(total);
    totalLabel.setText(totalString);
}

private void populateTable() {
    Object[][] cartListing = cart.getListing();
    cartTable.setModel(new javax.swing.table.DefaultTableModel(
        cartListing,
        new String [] {
            "ID", "Item", "Price", "Quantity", "SubTotal"
        }
    ) {
        boolean[] canEdit = new boolean [] {
            false, false, false, false, false
        };
```

```
            @Override
            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    private void initComponents() {

        cartTable = new javax.swing.JTable();
        backButton = new javax.swing.JButton();
        totalLabel = new javax.swing.JLabel();
        downButton = new javax.swing.JButton();
        upButton = new javax.swing.JButton();
        checkoutButton = new javax.swing.JButton();
        jLabelID = new javax.swing.JLabel();
        jLabelProductName = new javax.swing.JLabel();
        jLabelUnitPrice = new javax.swing.JLabel();
        jLabelQuantity = new javax.swing.JLabel();
        jLabelPrice = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
        setResizable(false);

        cartTable.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                cartTableMouseClicked(evt);
            }
        });
        cartTable.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
            public void propertyChange(java.beans.PropertyChangeEvent evt) {
                cartTablePropertyChange(evt);
            }
        });

        backButton.setText("Back");
        backButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                backButtonActionPerformed(evt);
            }
        });
        totalLabel.setFont(new java.awt.Font("Lucida Grande", 0, 18)); // NOI18N
        totalLabel.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
        totalLabel.setText("Total: $0.00");

        downButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/src/com/store/Alarm-Arrow-Down-
icon.png"))); // NOI18N
        downButton.setEnabled(false);
        downButton.setPreferredSize(new java.awt.Dimension(40, 40));
        downButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                downButtonActionPerformed(evt);
            }
        });

        upButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/src/com/store/Alarm-Arrow-Up-
icon.png"))); // NOI18N
        upButton.setEnabled(false);
        upButton.setPreferredSize(new java.awt.Dimension(40, 40));
        upButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                upButtonActionPerformed(evt);
            }
        });

        checkoutButton.setText("Check Out");
        checkoutButton.setEnabled(false);
        checkoutButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                checkoutButtonActionPerformed(evt);
            }
        });

        jLabelID.setText("ID");

        jLabelProductName.setText("Product Name");

        jLabelUnitPrice.setText("Unit Price");

        jLabelQuantity.setText("Quantity");

        jLabelPrice.setText("Price");

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(cartTable, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(downButton,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(upButton,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(totalLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 149,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(32, 32, 32)
                        .addComponent(checkoutButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(backButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 82,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(2, 2, 2)
                        .addComponent(jLabelID)
                        .addGap(74, 74, 74)
                        .addComponent(jLabelProductName)
                        .addGap(67, 67, 67)
                        .addComponent(jLabelUnitPrice)
                        .addGap(58, 58, 58)
                        .addComponent(jLabelQuantity)
                        .addGap(94, 94, 94)
                        .addComponent(jLabelPrice)
                        .addGap(0, 87, Short.MAX_VALUE)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabelID)
                    .addComponent(jLabelProductName)
                    .addComponent(jLabelUnitPrice)
                    .addComponent(jLabelQuantity)
                    .addComponent(jLabelPrice))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
8, Short.MAX_VALUE)
                .addComponent(cartTable, javax.swing.GroupLayout.PREFERRED_SIZE,
386, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(upButton, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 44, Short.MAX_VALUE)
                    .addComponent(downButton,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(backButton, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(checkoutButton,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(totalLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(22, 22, 22))
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>//GEN-END:initComponents

    private void backButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_backButtonActionPerformed
        for (CartListener hl : listeners)
            hl.cartExited();
        this.setVisible(false);
    }//GEN-LAST:event_backButtonActionPerformed

    private void cartTablePropertyChange(java.beans.PropertyChangeEvent evt) {//GEN-
FIRST:event_cartTablePropertyChange

    }//GEN-LAST:event_cartTablePropertyChange

    private void cartTableMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_cartTableMouseClicked
        int row = cartTable.getSelectedRow();
       // System.out.println("Row:" + row);
        if (row >= 0) {
            productId = Integer.parseInt(cartTable.getValueAt(row, 0).toString());
        //    System.out.println("currentItemID:" + productId);
            updateUpDownButtons();
        }
    }//GEN-LAST:event_cartTableMouseClicked

    private void updateUpDownButtons() {
        int quantityInInventory = inventory.retrieveQuantity(productId);
        int quantityInCart = cart.retrieveQuantity(productId);

        int quantityAvailable = quantityInInventory - quantityInCart;

        if (quantityInCart > 0)
            downButton.setEnabled(true);
        else
            downButton.setEnabled(false);

        if (quantityAvailable > 0)
            upButton.setEnabled(true);
        else
            upButton.setEnabled(false);
    }

    private void checkoutButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_checkoutButtonActionPerformed
        PurchaseForm pForm = new PurchaseForm();
        pForm.addListener(this);
        pForm.setVisible(true);
    }//GEN-LAST:event_checkoutButtonActionPerformed

    private void downButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_downButtonActionPerformed
        cart.decrement(productId, 1);
        updateUpDownButtons();
        updateShoppingCartView();
    }//GEN-LAST:event_downButtonActionPerformed

    private void upButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_upButtonActionPerformed
        cart.increment(productId, 1);
        updateUpDownButtons();
        updateShoppingCartView();
    }//GEN-LAST:event_upButtonActionPerformed

    @Override
    public void finished() {
        updateShoppingCartView();
    }

    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look
and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(CartView.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(CartView.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(CartView.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

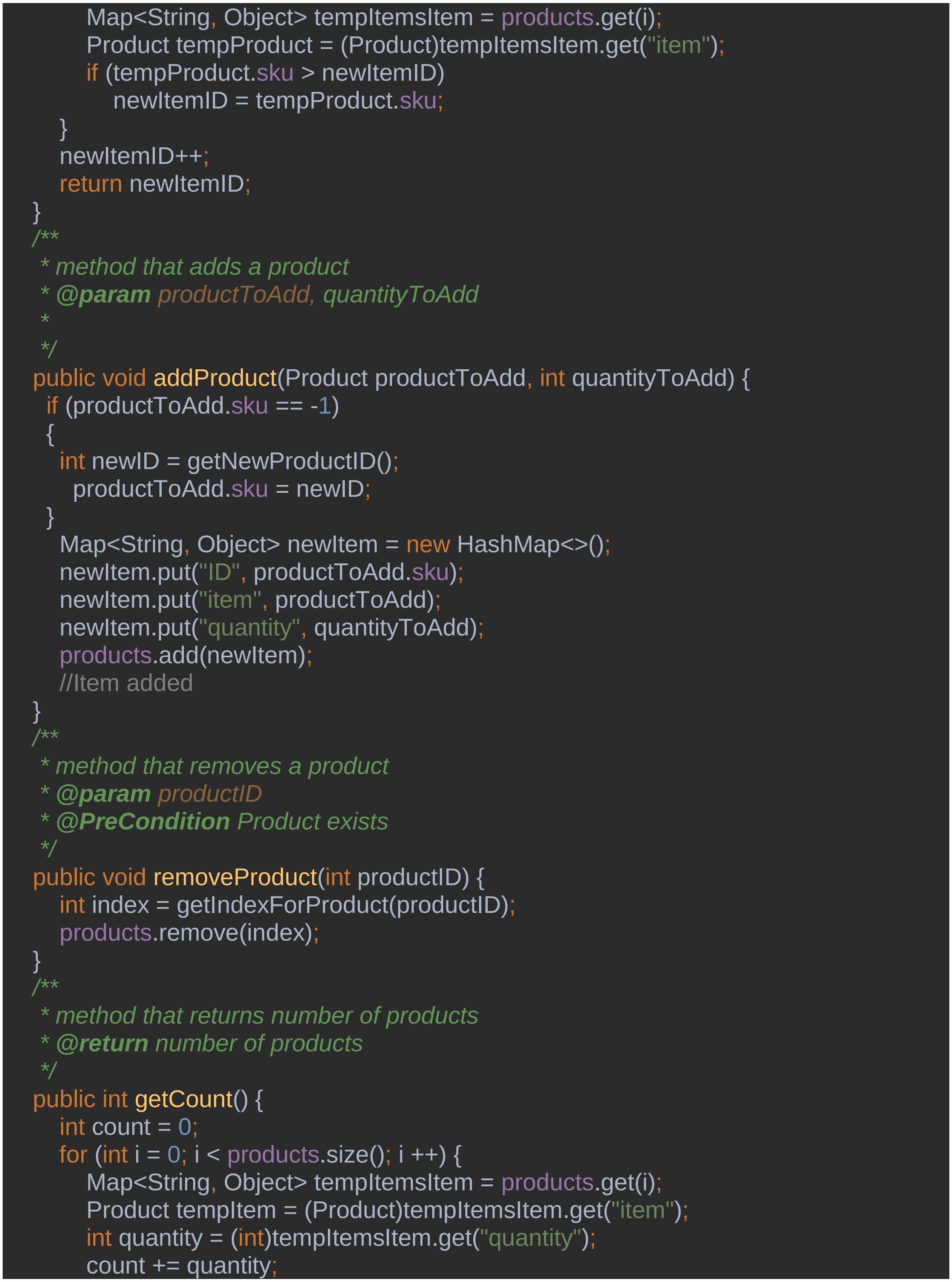
```
java.util.logging.Logger.getLogger(CartView.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new CartView().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton backButton;
    private javax.swing.JTable cartTable;
    private javax.swing.JButton checkoutButton;
    private javax.swing.JButton downButton;
    private javax.swing.JLabel jLabelID;
    private javax.swing.JLabel jLabelPrice;
    private javax.swing.JLabel jLabelProductName;
    private javax.swing.JLabel jLabelQuantity;
    private javax.swing.JLabel jLabelUnitPrice;
    private javax.swing.JLabel totalLabel;
    private javax.swing.JButton upButton;
    // End of variables declaration//GEN-END:variables
}
```

Product.java

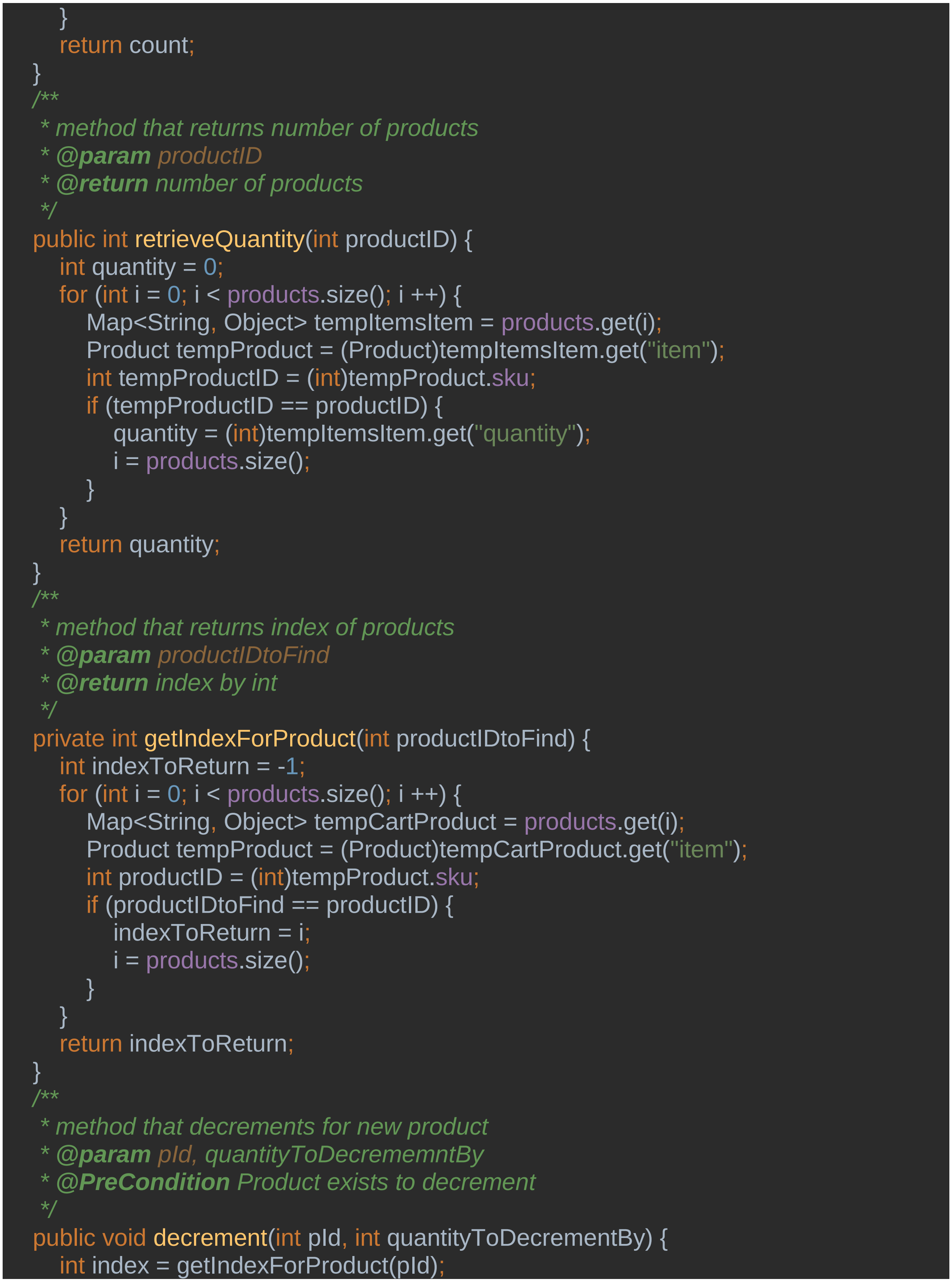
```
package com.store;

import java.io.Serializable;

/**
 * Product Class
 * @author Thomas Sonderman
 */
public class Product implements Serializable{
    public int sku;
    public String name;
    public String description;
    public double price;
    public double cost;
    public int quantity;
}
```

ProductList.java

```
package com.store;

/**
 * Created by WPTRS on 11/4/2015.
 */


import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

/**
 * ProductList Class controls adding removing
 * @author Thomas Sonderman
 */
public class ProductList {
    private ArrayList<Map<String, Object>> products;
    public ArrayList<Map<String, Object>> productsToLoad;
    double revenue;
    double profit;
    double cost;

    int currentIteratorID;

    protected ProductList() {

        products = new ArrayList<>();
        resetIterator();
    }

    public void resetItems() {
        products.removeAll(products);
    }
    /**
     * method that get a new ID for product
     * @return int
     */
    private int getNewProductID() {
        int newItemID = -1;
        for (int i = 0; i < products.size(); i++) {
```

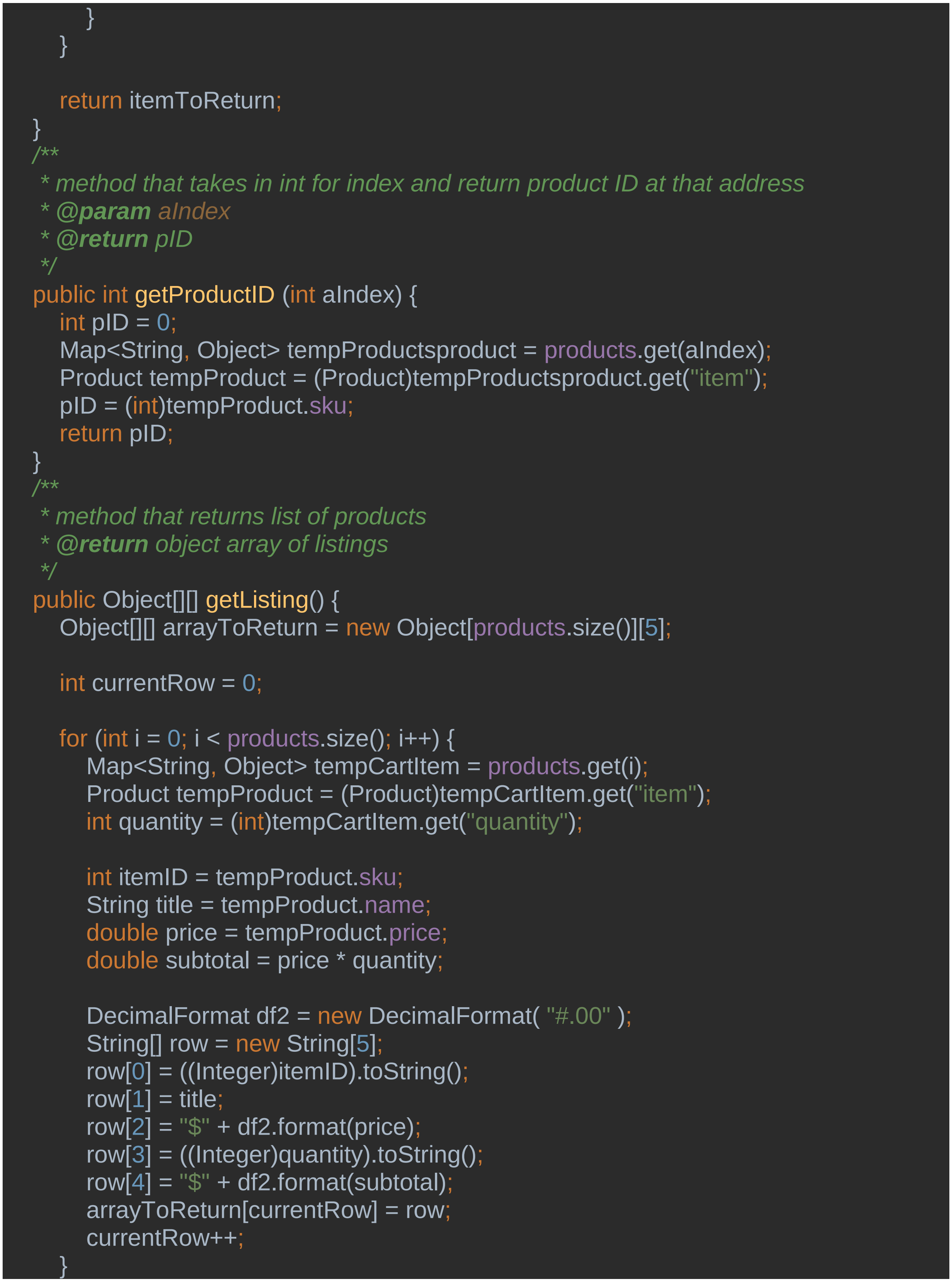


```
            Map<String, Object> tempItemsItem = products.get(i);
            Product tempProduct = (Product)tempItemsItem.get("item");
            if (tempProduct.sku > newItemID)
                newItemID = tempProduct.sku;
        }
        newItemID++;
        return newItemID;
    }
    /**
     * method that adds a product
     * @param productToAdd, quantityToAdd
     *
     */
    public void addProduct(Product productToAdd, int quantityToAdd) {
      if (productToAdd.sku == -1)
      {
        int newID = getNewProductID();
          productToAdd.sku = newID;
      }
        Map<String, Object> newItem = new HashMap<>();
        newItem.put("ID", productToAdd.sku);
        newItem.put("item", productToAdd);
        newItem.put("quantity", quantityToAdd);
        products.add(newItem);
        //Item added
    }
    /**
     * method that removes a product
     * @param productID
     * @PreCondition Product exists
     */
    public void removeProduct(int productID) {
        int index = getIndexForProduct(productID);
        products.remove(index);
    }
    /**
     * method that returns number of products
     * @return number of products
     */
    public int getCount() {
        int count = 0;
        for (int i = 0; i < products.size(); i ++) {
            Map<String, Object> tempItemsItem = products.get(i);
            Product tempItem = (Product)tempItemsItem.get("item");
            int quantity = (int)tempItemsItem.get("quantity");
            count += quantity;
```

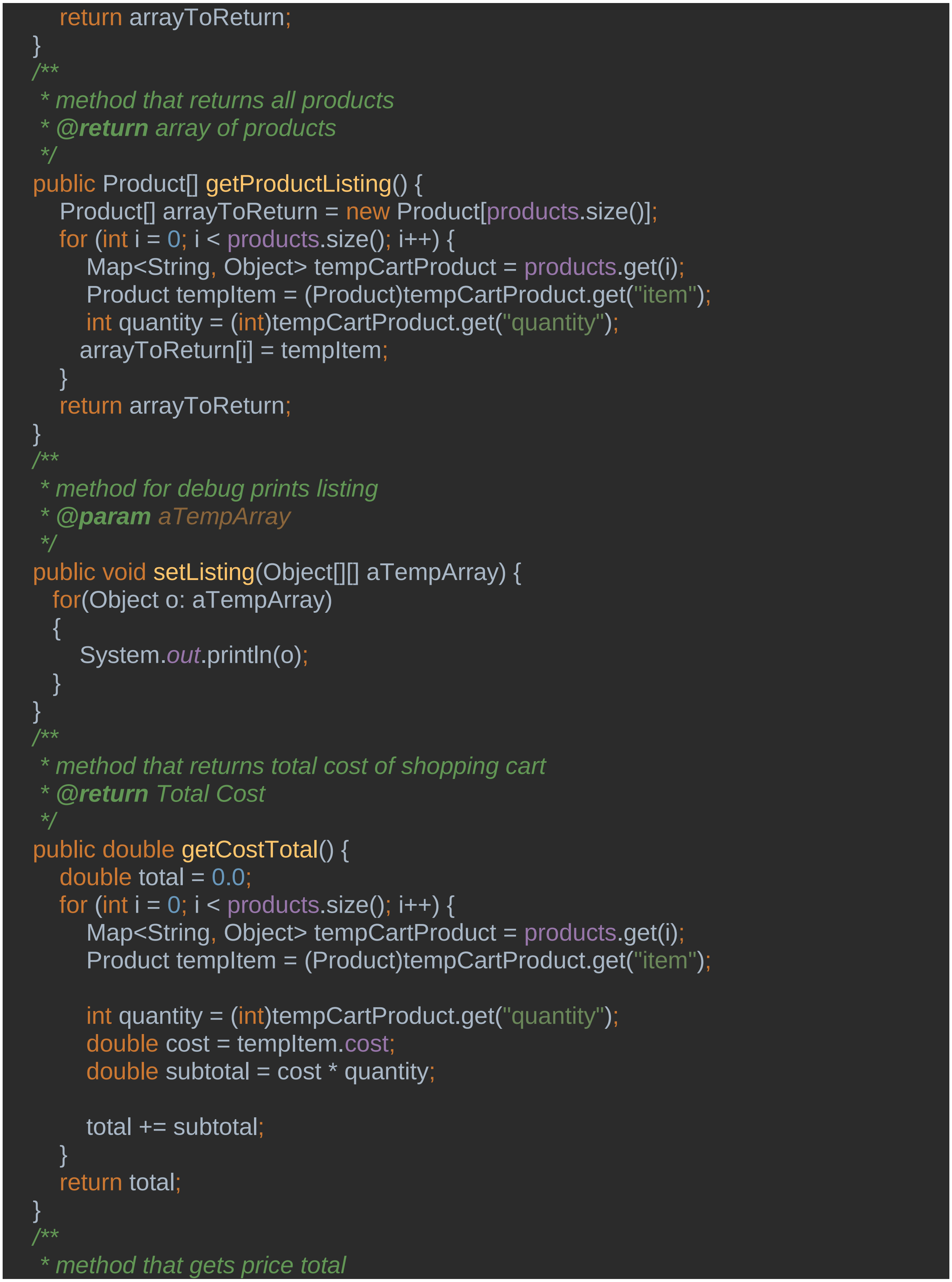


```
    }
    return count;
}
/**
 * method that returns number of products
 * @param productID
 * @return number of products
 */
public int retrieveQuantity(int productID) {
    int quantity = 0;
    for (int i = 0; i < products.size(); i ++) {
        Map<String, Object> tempItemsItem = products.get(i);
        Product tempProduct = (Product)tempItemsItem.get("item");
        int tempProductID = (int)tempProduct.sku;
        if (tempProductID == productID) {
            quantity = (int)tempItemsItem.get("quantity");
            i = products.size();
        }
    }
    return quantity;
}
/**
 * method that returns index of products
 * @param productIDtoFind
 * @return index by int
 */
private int getIndexForProduct(int productIDtoFind) {
    int indexToReturn = -1;
    for (int i = 0; i < products.size(); i ++) {
        Map<String, Object> tempCartProduct = products.get(i);
        Product tempProduct = (Product)tempCartProduct.get("item");
        int productID = (int)tempProduct.sku;
        if (productIDtoFind == productID) {
            indexToReturn = i;
            i = products.size();
        }
    }
    return indexToReturn;
}
/**
 * method that decrements for new product
 * @param pId, quantityToDecrememntBy
 * @PreCondition Product exists to decrement
 */
public void decrement(int pId, int quantityToDecrementBy) {
    int index = getIndexForProduct(pId);
```

```java
    if (index > -1) {
        Map<String, Object> tempCartProduct = products.get(index);
        Product tempItem = (Product)tempCartProduct.get("item");
        int quantity = (int)tempCartProduct.get("quantity");
        if (quantity - quantityToDecrementBy <= 0) {
            quantity = 0;
            tempCartProduct.put("quantity", quantity);
            products.set(index, tempCartProduct);
        }
        else {
            tempItem.quantity -= quantityToDecrementBy;
            quantity -= quantityToDecrementBy;
            tempCartProduct.put("quantity", quantity);
            products.set(index, tempCartProduct);
        }
    }
}
/**
 * method that increments for new product
 * @param pId, quantityToIncrememntBy
 * @PreCondition Product exists to increment
 */
public void increment(int pId, int quantityToIncrementBy) {
    int index = getIndexForProduct(pId);
    if (index > -1) {
        Map<String, Object> tempCartItem = products.get(index);
        int quantity = (int)tempCartItem.get("quantity");
        quantity += quantityToIncrementBy;
        tempCartItem.put("quantity", quantity);
        products.set(index, tempCartItem);
    }
}
/**
 * method that returns product by passed in ID
 * @param pID
 * @return product object
 */
public Product getProductByID (int pID) {
    Product itemToReturn = new Product();
    for (int i = 0; i < products.size(); i ++) {
        Map<String, Object> inventoryItem = products.get(i);
        Product tempProduct = (Product)inventoryItem.get("item");
        int tpID = (int)tempProduct.sku;
        if (tpID == pID) {
            itemToReturn = tempProduct;
            i = products.size();
```

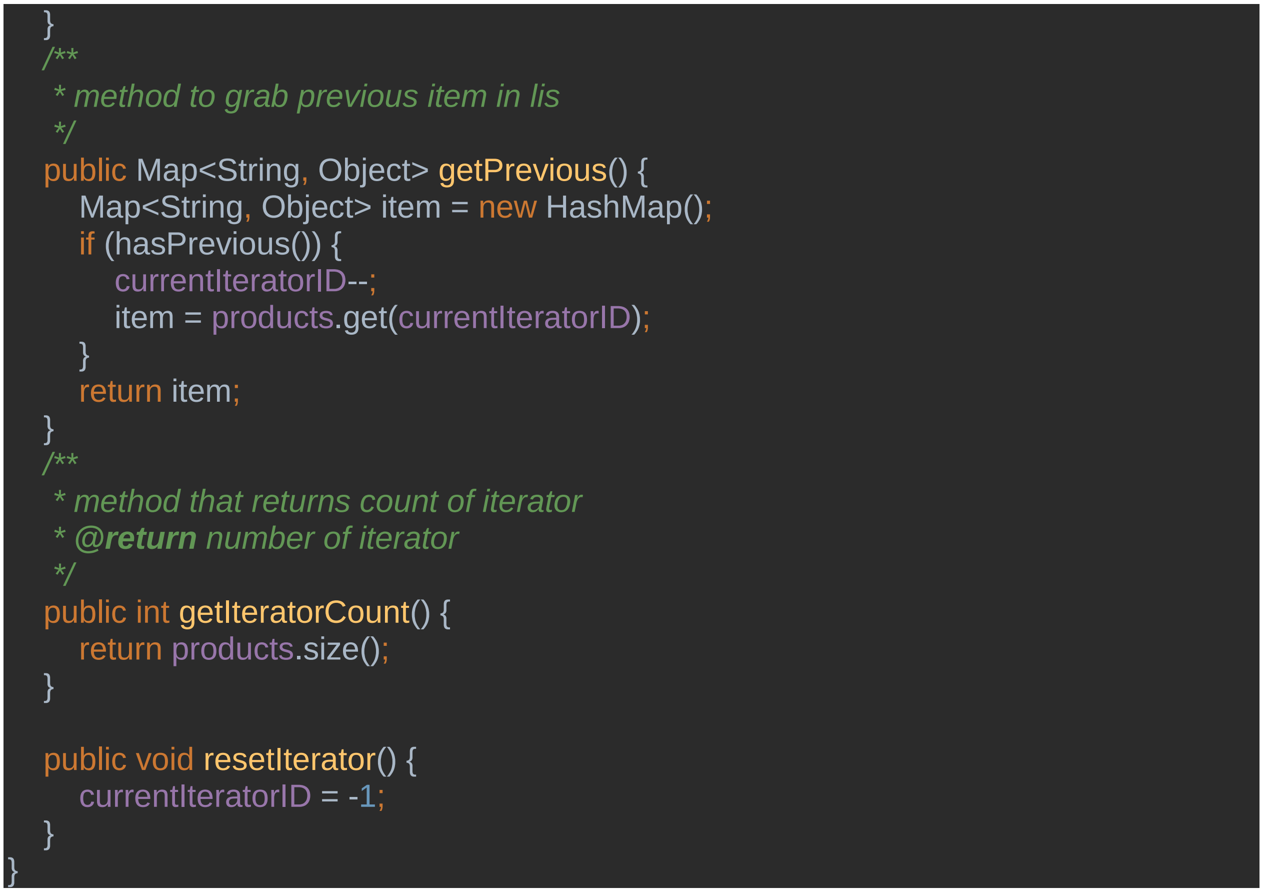


```java
            }
        }

        return itemToReturn;
    }
    /**
     * method that takes in int for index and return product ID at that address
     * @param aIndex
     * @return pID
     */
    public int getProductID (int aIndex) {
        int pID = 0;
        Map<String, Object> tempProductsproduct = products.get(aIndex);
        Product tempProduct = (Product)tempProductsproduct.get("item");
        pID = (int)tempProduct.sku;
        return pID;
    }
    /**
     * method that returns list of products
     * @return object array of listings
     */
    public Object[][] getListing() {
        Object[][] arrayToReturn = new Object[products.size()][5];

        int currentRow = 0;

        for (int i = 0; i < products.size(); i++) {
            Map<String, Object> tempCartItem = products.get(i);
            Product tempProduct = (Product)tempCartItem.get("item");
            int quantity = (int)tempCartItem.get("quantity");

            int itemID = tempProduct.sku;
            String title = tempProduct.name;
            double price = tempProduct.price;
            double subtotal = price * quantity;

            DecimalFormat df2 = new DecimalFormat( "#.00" );
            String[] row = new String[5];
            row[0] = ((Integer)itemID).toString();
            row[1] = title;
            row[2] = "$" + df2.format(price);
            row[3] = ((Integer)quantity).toString();
            row[4] = "$" + df2.format(subtotal);
            arrayToReturn[currentRow] = row;
            currentRow++;
        }
```



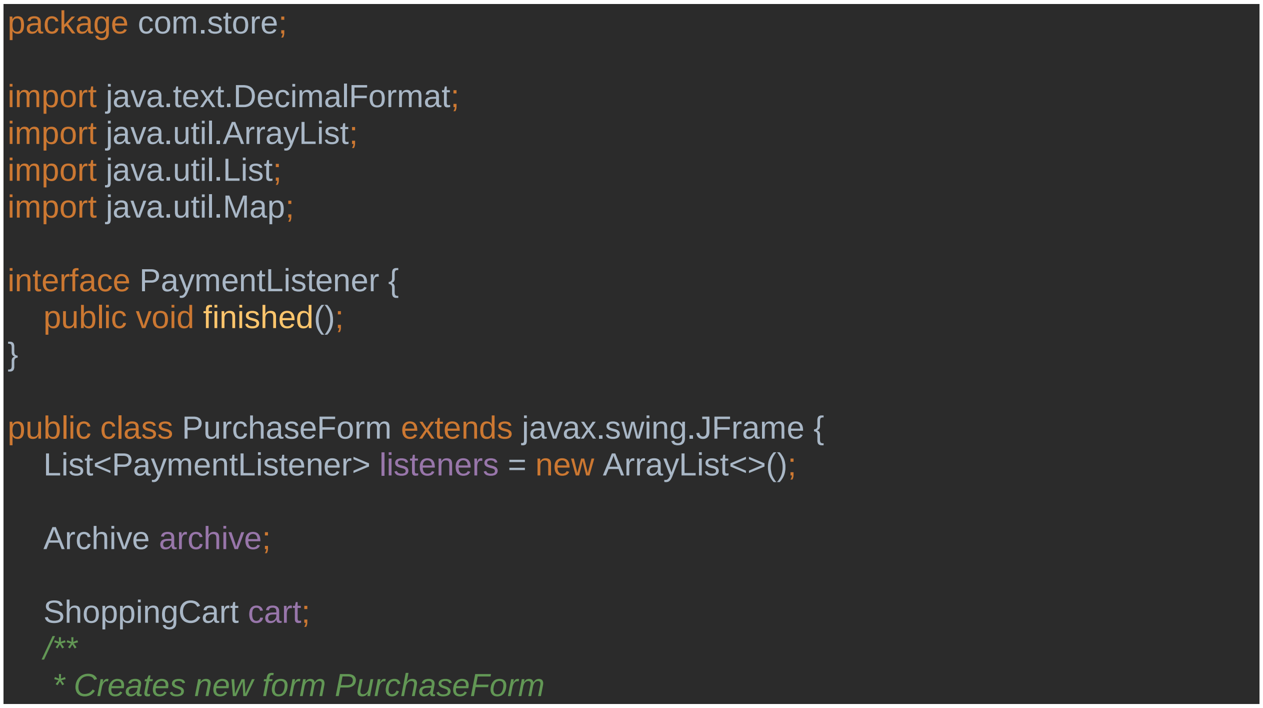
```
        return arrayToReturn;
    }
    /**
     * method that returns all products
     * @return array of products
     */
    public Product[] getProductListing() {
        Product[] arrayToReturn = new Product[products.size()];
        for (int i = 0; i < products.size(); i++) {
            Map<String, Object> tempCartProduct = products.get(i);
            Product tempItem = (Product)tempCartProduct.get("item");
            int quantity = (int)tempCartProduct.get("quantity");
           arrayToReturn[i] = tempItem;
        }
        return arrayToReturn;
    }
    /**
     * method for debug prints listing
     * @param aTempArray
     */
    public void setListing(Object[][] aTempArray) {
       for(Object o: aTempArray)
       {
           System.out.println(o);
       }
    }
    /**
     * method that returns total cost of shopping cart
     * @return Total Cost
     */
    public double getCostTotal() {
        double total = 0.0;
        for (int i = 0; i < products.size(); i++) {
            Map<String, Object> tempCartProduct = products.get(i);
            Product tempItem = (Product)tempCartProduct.get("item");

            int quantity = (int)tempCartProduct.get("quantity");
            double cost = tempItem.cost;
            double subtotal = cost * quantity;

            total += subtotal;
        }
        return total;
    }
    /**
     * method that gets price total
```

```
 * @return total price
 */
public double getPriceTotal() {
    double total = 0.0;
    for (int i = 0; i < products.size(); i++) {
        Map<String, Object> tempCartItem = products.get(i);
        Product tempProduct = (Product)tempCartItem.get("item");

        int quantity = (int)tempCartItem.get("quantity");
        double price = tempProduct.price;
        double subtotal = price * quantity;

        total += subtotal;
    }
    return total;
}
/**
 * method that returns whether next product exists
 * @return true or false
 */
public boolean hasNext() {
    boolean hasNext = false;
    if (currentIteratorID < products.size() - 1)
        hasNext = true;
    return hasNext;
}
/**
 * method that determines whether previous product exists
 * @return true or false
 */
public boolean hasPrevious() {
    boolean hasPrevious = false;
    if (currentIteratorID > 0)
        hasPrevious = true;
    return hasPrevious;
}
/**
 * method to grab next item with iterator
 */
public Map<String, Object> getNext() {
    Map<String, Object> item = new HashMap();
    if (hasNext()) {
        currentIteratorID++;
        item = products.get(currentIteratorID);
    }
    return item;
```



```
    }
    /**
     * method to grab previous item in lis
     */
    public Map<String, Object> getPrevious() {
        Map<String, Object> item = new HashMap();
        if (hasPrevious()) {
            currentIteratorID--;
            item = products.get(currentIteratorID);
        }
        return item;
    }
    /**
     * method that returns count of iterator
     * @return number of iterator
     */
    public int getIteratorCount() {
        return products.size();
    }

    public void resetIterator() {
        currentIteratorID = -1;
    }
}
```

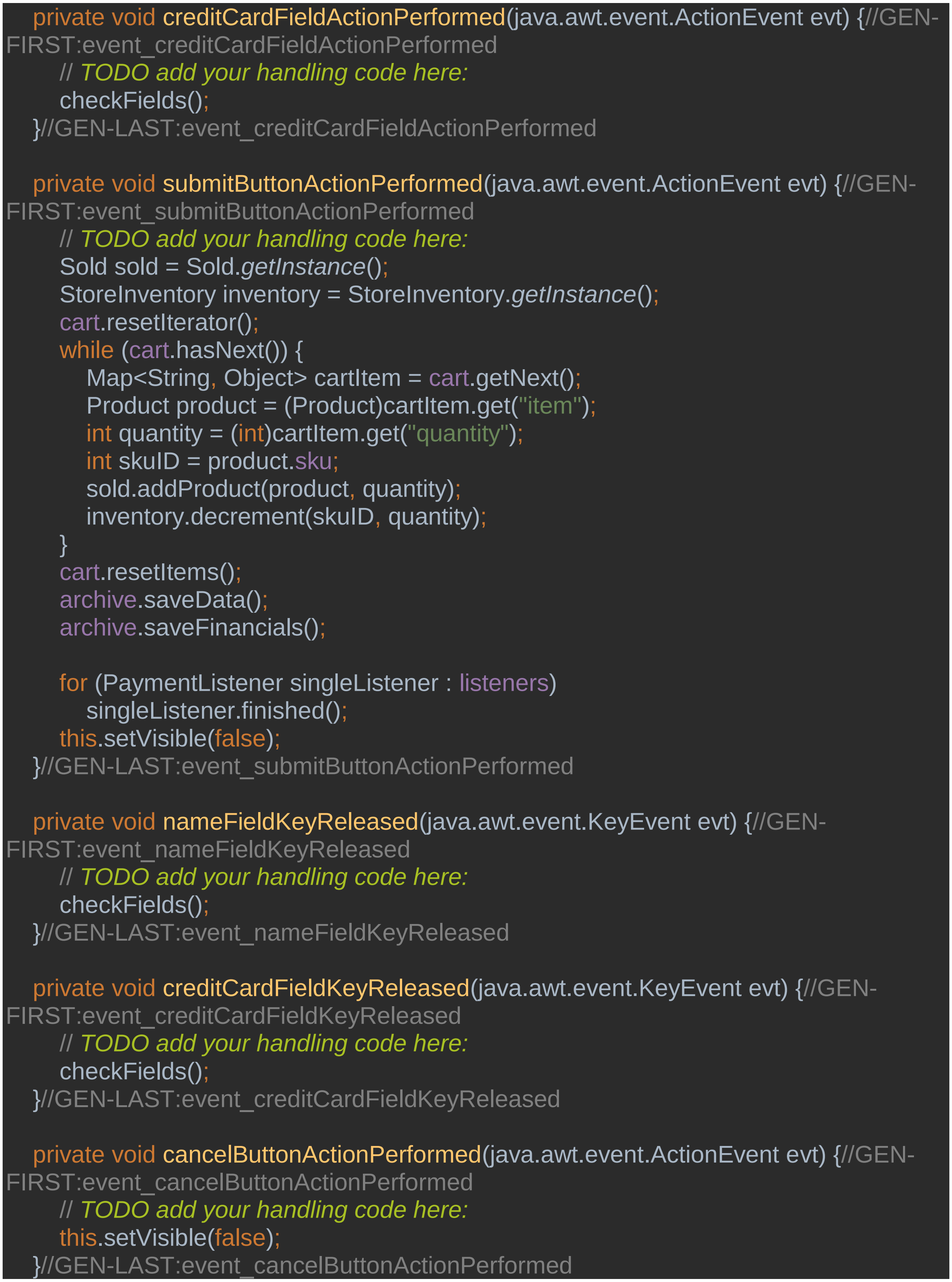
PurchaseForm.java



```
package com.store;

import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

interface PaymentListener {
    public void finished();
}

public class PurchaseForm extends javax.swing.JFrame {
    List<PaymentListener> listeners = new ArrayList<>();

    Archive archive;

    ShoppingCart cart;
    /**
     * Creates new form PurchaseForm
     */
    public PurchaseForm() {
        initComponents();
        cart = ShoppingCart.getCart();
        double total = cart.getPriceTotal();
        DecimalFormat df2 = new DecimalFormat("#0.00");
        String totalString = "$" + df2.format(total);
        totalLabel.setText(totalString);
        archive = new Archive();
    }

    public void addListener(PaymentListener toAdd) {
        listeners.add(toAdd);
    }

    public void checkFields () {
        String nameString = nameField.getText();
        String fakeCreditCard = creditCardField.getText();

        if (nameString.length() > 0 && fakeCreditCard.length() > 0) {
            submitButton.setEnabled(true);
        }
        else {
            submitButton.setEnabled(false);
        }
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        totalLabel = new javax.swing.JLabel();
        nameField = new javax.swing.JTextField();
        creditCardField = new javax.swing.JTextField();
        submitButton = new javax.swing.JButton();
        cancelButton = new javax.swing.JButton();
        jLabel3 = new javax.swing.JLabel();
```

```
setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);

        jLabel1.setText("Name");

        jLabel2.setText("Credit Card");

        totalLabel.setText("Total: $0.00");

        nameField.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                nameFieldActionPerformed(evt);
            }
        });
        nameField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                nameFieldKeyReleased(evt);
            }
        });

        creditCardField.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                creditCardFieldActionPerformed(evt);
            }
        });
        creditCardField.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyReleased(java.awt.event.KeyEvent evt) {
                creditCardFieldKeyReleased(evt);
            }
        });

        submitButton.setText("Submit");
        submitButton.setEnabled(false);
        submitButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                submitButtonActionPerformed(evt);
            }
        });

        cancelButton.setText("Cancel");
        cancelButton.setToolTipText("");
        cancelButton.setActionCommand("Cancel");
        cancelButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelButtonActionPerformed(evt);
            }
```

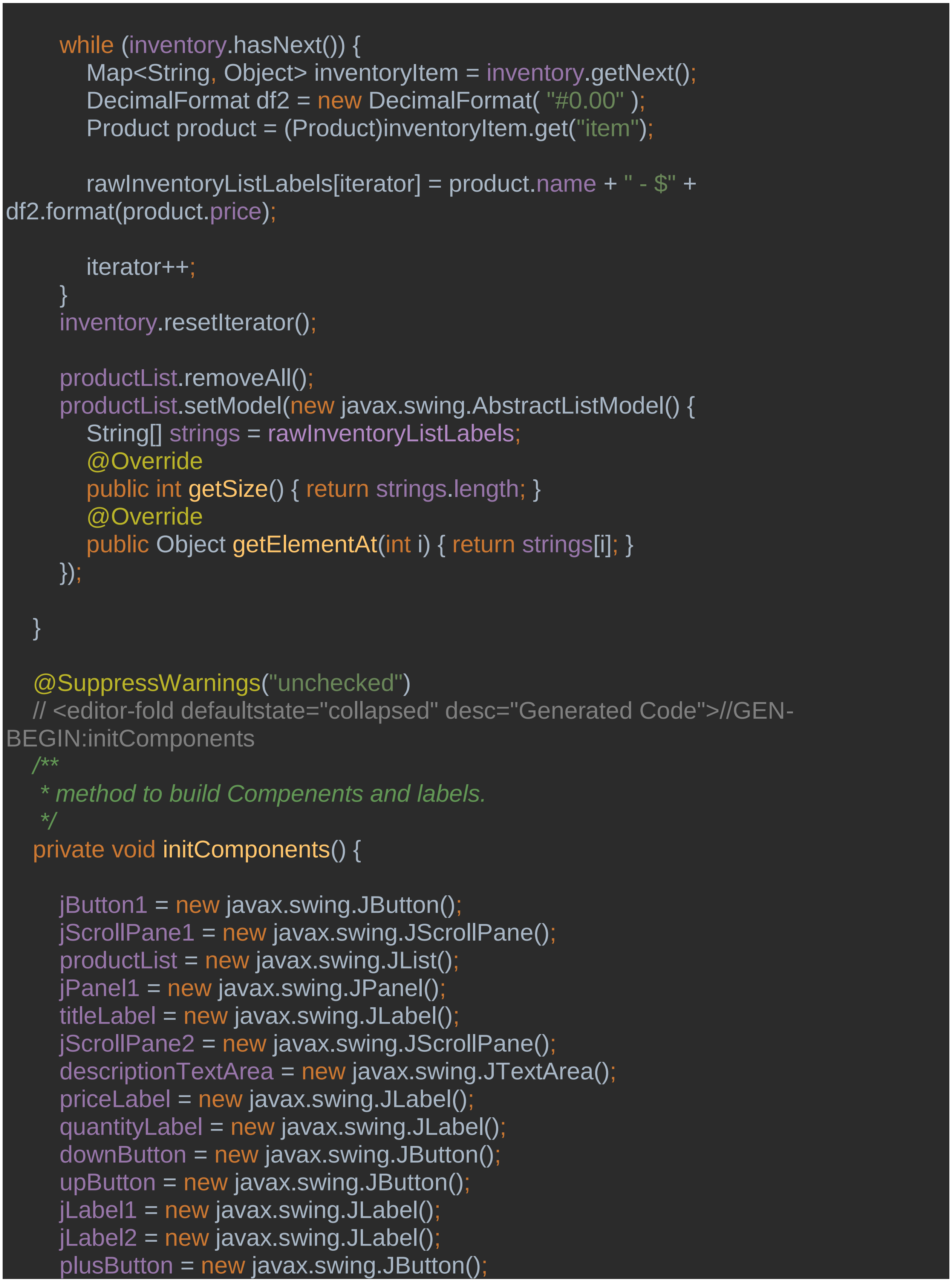
```
    });

    jLabel3.setFont(new java.awt.Font("Lucida Grande", 0, 18)); // NOI18N
    jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    jLabel3.setText("Checkout");

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(26, 26, 26)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel2)
                    .addComponent(jLabel1))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(6, 6, 6)
                        .addComponent(submitButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(cancelButton))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                        .addComponent(nameField)
                        .addComponent(creditCardField)
                        .addComponent(totalLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 268,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(29, Short.MAX_VALUE))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(25, 25, 25)
                    .addComponent(jLabel3,
javax.swing.GroupLayout.DEFAULT_SIZE, 361, Short.MAX_VALUE)
                    .addGap(26, 26, 26)))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(47, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(nameField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(creditCardField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addComponent(totalLabel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(submitButton)
                    .addComponent(cancelButton))
                .addGap(21, 21, 21))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap(179, Short.MAX_VALUE)))
    );

    pack();
    setLocationRelativeTo(null);
  }// </editor-fold>//GEN-END:initComponents

  private void nameFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_nameFieldActionPerformed
    // TODO add your handling code here:
    checkFields();
  }//GEN-LAST:event_nameFieldActionPerformed
```



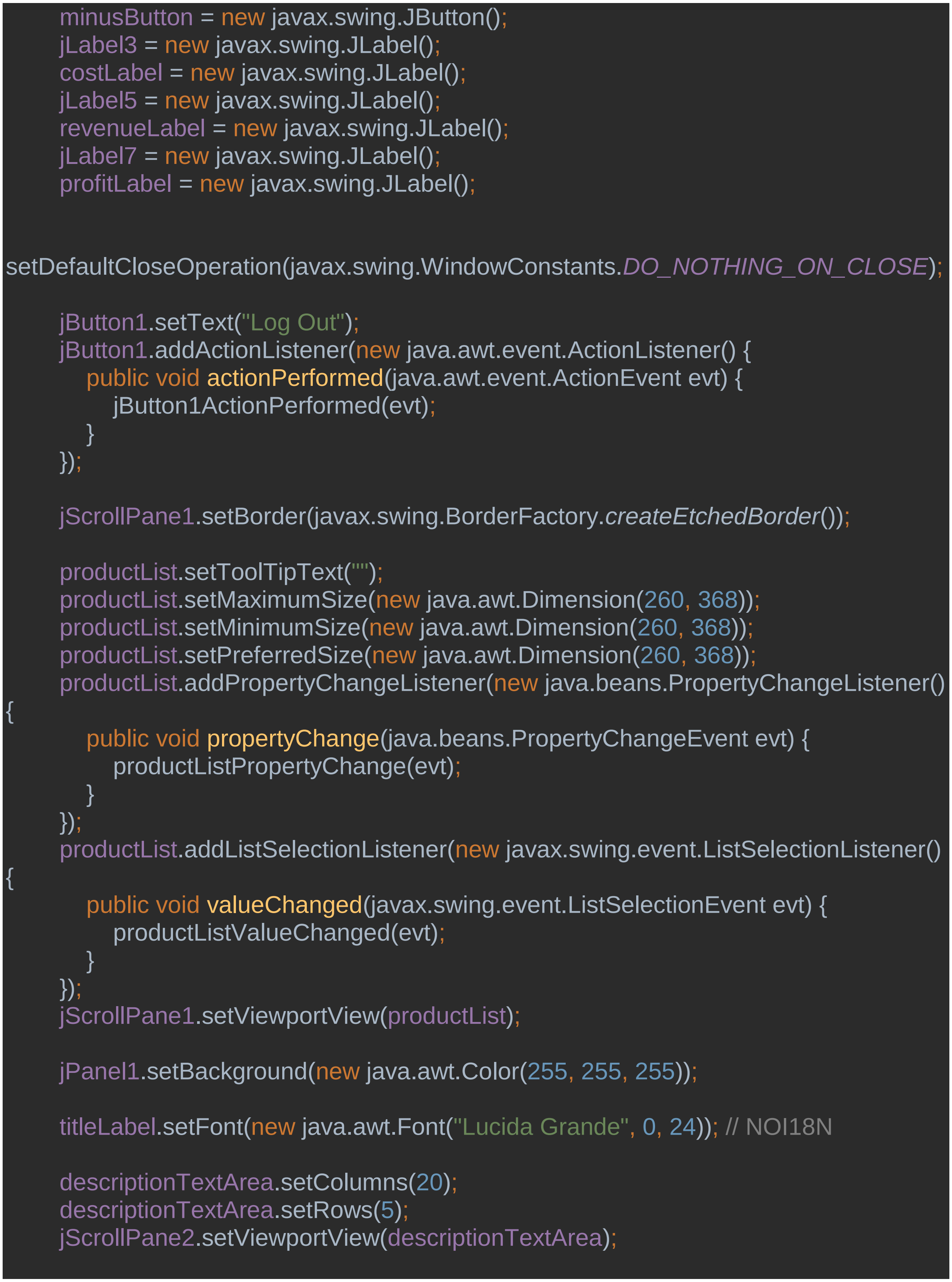
```
    private void creditCardFieldActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_creditCardFieldActionPerformed
        // TODO add your handling code here:
        checkFields();
    }//GEN-LAST:event_creditCardFieldActionPerformed

    private void submitButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_submitButtonActionPerformed
        // TODO add your handling code here:
        Sold sold = Sold.getInstance();
        StoreInventory inventory = StoreInventory.getInstance();
        cart.resetIterator();
        while (cart.hasNext()) {
            Map<String, Object> cartItem = cart.getNext();
            Product product = (Product)cartItem.get("item");
            int quantity = (int)cartItem.get("quantity");
            int skuID = product.sku;
            sold.addProduct(product, quantity);
            inventory.decrement(skuID, quantity);
        }
        cart.resetItems();
        archive.saveData();
        archive.saveFinancials();

        for (PaymentListener singleListener : listeners)
            singleListener.finished();
        this.setVisible(false);
    }//GEN-LAST:event_submitButtonActionPerformed

    private void nameFieldKeyReleased(java.awt.event.KeyEvent evt) {//GEN-
FIRST:event_nameFieldKeyReleased
        // TODO add your handling code here:
        checkFields();
    }//GEN-LAST:event_nameFieldKeyReleased

    private void creditCardFieldKeyReleased(java.awt.event.KeyEvent evt) {//GEN-
FIRST:event_creditCardFieldKeyReleased
        // TODO add your handling code here:
        checkFields();
    }//GEN-LAST:event_creditCardFieldKeyReleased

    private void cancelButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_cancelButtonActionPerformed
        // TODO add your handling code here:
        this.setVisible(false);
    }//GEN-LAST:event_cancelButtonActionPerformed
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look
and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(PurchaseForm.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(PurchaseForm.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(PurchaseForm.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(PurchaseForm.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new PurchaseForm().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton cancelButton;
    private javax.swing.JTextField creditCardField;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JTextField nameField;
    private javax.swing.JButton submitButton;
    private javax.swing.JLabel totalLabel;
    // End of variables declaration//GEN-END:variables
}
```

Seller.java

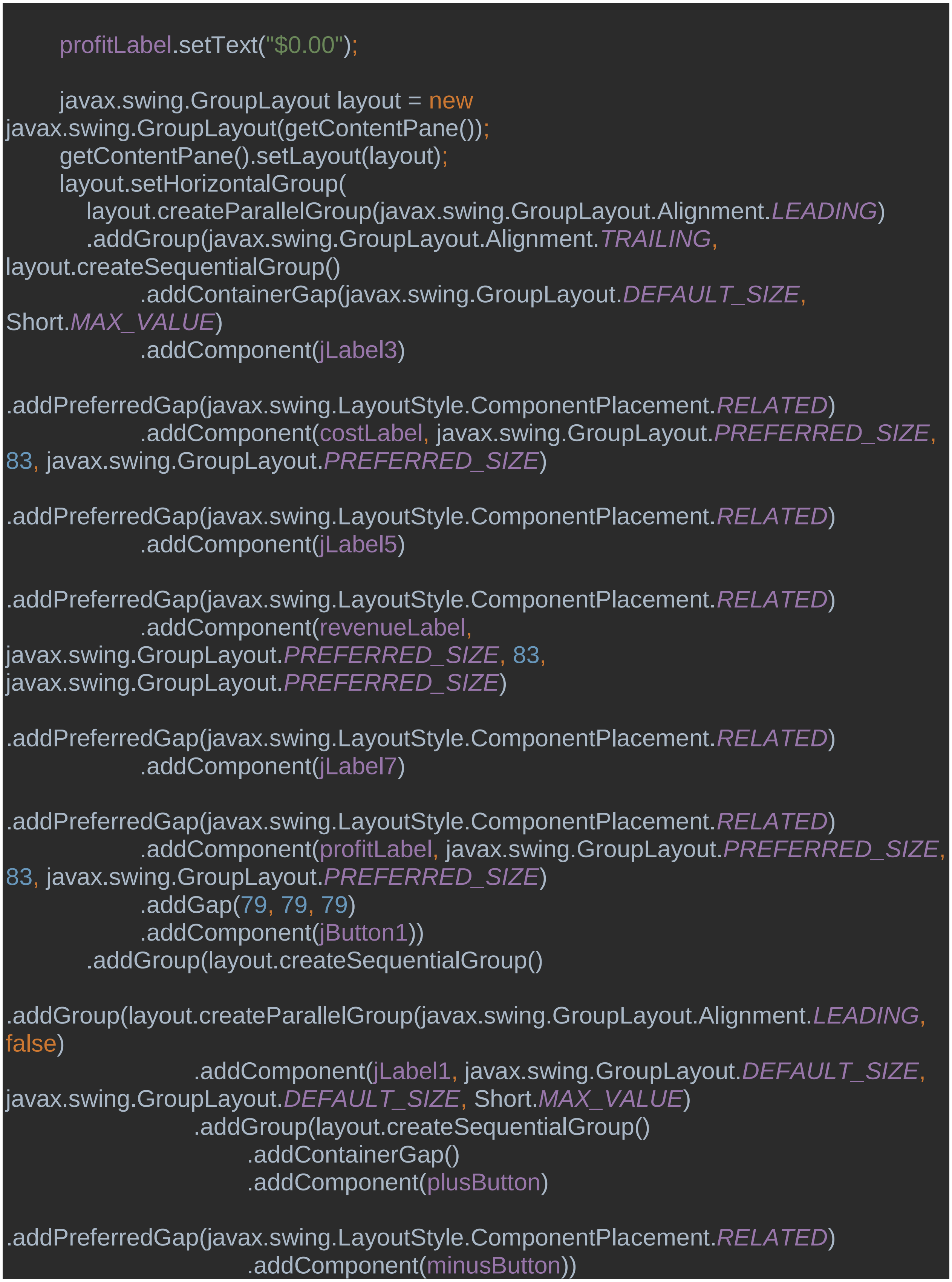
```
package com.store;

import java.awt.Color;
import java.text.DecimalFormat;
import java.util.Map;

/**
 * Seller Class for seller only function
 * Seller class extends the AddProductListener
 */
public class Seller extends javax.swing.JFrame implements AddProductListener {

    StoreInventory inventory;
    ShoppingCart cart;
    Sold sold;
    Archive archive;

    String[] inventoryListLabels;
    int currentProductID;
    /**
     * Seller Constructor
     */
    public Seller() {

        initComponents();

        inventory = StoreInventory.getInstance();

        cart = ShoppingCart.getCart();
        sold = Sold.getInstance();

        archive = new Archive();


        updateInventoryList();

        updateFinancials();


        currentProductID = 0;

    }
    /**
     * method to check and update financials
     */
    private void updateFinancials() {
        double cost = sold.getCostTotal() + inventory.getCostTotal() + inventory.cost;
        double revenue = sold.getPriceTotal() +  inventory.revenue;
        double profit = (revenue - cost) + inventory.profit;
        inventory.profit = inventory.profit + (sold.getPriceTotal()-sold.getCostTotal());
        inventory.revenue = inventory.revenue + sold.getPriceTotal();
        inventory.cost = sold.getCostTotal() + inventory.cost;
        archive.saveFinancials();

        DecimalFormat df2 = new DecimalFormat( "#0.00" );

        costLabel.setText("$" + df2.format(cost));
        revenueLabel.setText("$" + df2.format(revenue));

        if (profit < 0)
            profitLabel.setForeground(Color.RED);
        else
            profitLabel.setForeground(Color.GREEN);

        profitLabel.setText("$" + df2.format(profit));
    }
    /**
     * method to update current inventory list
     */
    private void updateInventoryList() {

        String[] rawInventoryListLabels = new String[inventory.getIteratorCount()];

        int iterator = 0;
```



```
        while (inventory.hasNext()) {
            Map<String, Object> inventoryItem = inventory.getNext();
            DecimalFormat df2 = new DecimalFormat( "#0.00" );
            Product product = (Product)inventoryItem.get("item");

            rawInventoryListLabels[iterator] = product.name + " - $" +
df2.format(product.price);

            iterator++;
        }
        inventory.resetIterator();

        productList.removeAll();
        productList.setModel(new javax.swing.AbstractListModel() {
            String[] strings = rawInventoryListLabels;
            @Override
            public int getSize() { return strings.length; }
            @Override
            public Object getElementAt(int i) { return strings[i]; }
        });

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents
    /**
     * method to build Compenents and labels.
     */
    private void initComponents() {

        jButton1 = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        productList = new javax.swing.JList();
        jPanel1 = new javax.swing.JPanel();
        titleLabel = new javax.swing.JLabel();
        jScrollPane2 = new javax.swing.JScrollPane();
        descriptionTextArea = new javax.swing.JTextArea();
        priceLabel = new javax.swing.JLabel();
        quantityLabel = new javax.swing.JLabel();
        downButton = new javax.swing.JButton();
        upButton = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        plusButton = new javax.swing.JButton();
```

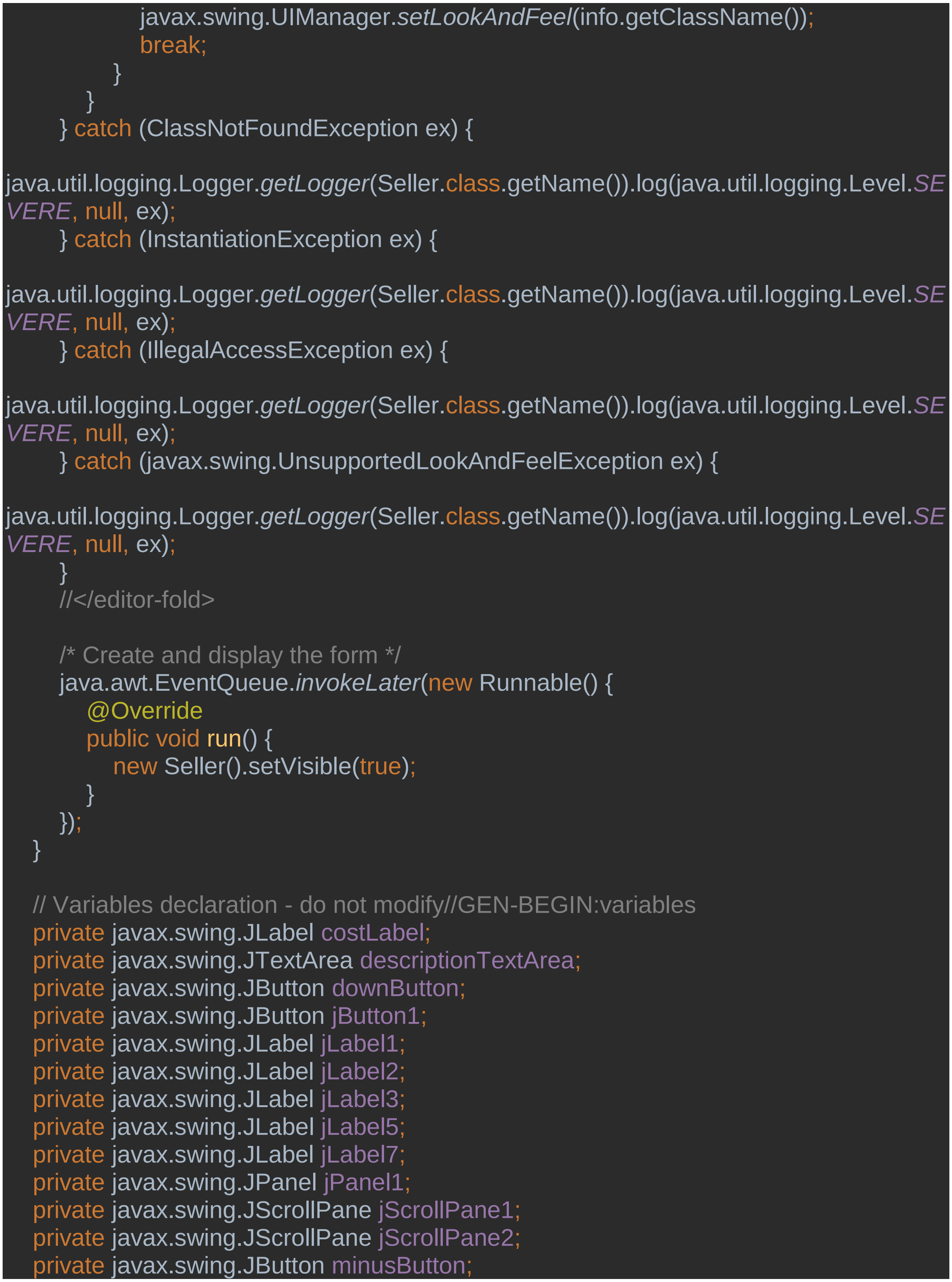


```
        minusButton = new javax.swing.JButton();
        jLabel3 = new javax.swing.JLabel();
        costLabel = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        revenueLabel = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        profitLabel = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);

        jButton1.setText("Log Out");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jScrollPane1.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        productList.setToolTipText("");
        productList.setMaximumSize(new java.awt.Dimension(260, 368));
        productList.setMinimumSize(new java.awt.Dimension(260, 368));
        productList.setPreferredSize(new java.awt.Dimension(260, 368));
        productList.addPropertyChangeListener(new java.beans.PropertyChangeListener()
{
            public void propertyChange(java.beans.PropertyChangeEvent evt) {
                productListPropertyChange(evt);
            }
        });
        productList.addListSelectionListener(new javax.swing.event.ListSelectionListener()
{
            public void valueChanged(javax.swing.event.ListSelectionEvent evt) {
                productListValueChanged(evt);
            }
        });
        jScrollPane1.setViewportView(productList);

        jPanel1.setBackground(new java.awt.Color(255, 255, 255));

        titleLabel.setFont(new java.awt.Font("Lucida Grande", 0, 24)); // NOI18N

        descriptionTextArea.setColumns(20);
        descriptionTextArea.setRows(5);
        jScrollPane2.setViewportView(descriptionTextArea);
        priceLabel.setFont(new java.awt.Font("Lucida Grande", 0, 24)); // NOI18N
        priceLabel.setText(" ");

        quantityLabel.setFont(new java.awt.Font("Lucida Grande", 0, 24)); // NOI18N
        quantityLabel.setText(" ");

        downButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/src/com/store/Alarm-Arrow-Down-
icon.png"))); // NOI18N
        downButton.setEnabled(false);
        downButton.setPreferredSize(new java.awt.Dimension(40, 40));
        downButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                downButtonActionPerformed(evt);
            }
        });

        upButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/src/com/store/Alarm-Arrow-Up-
icon.png"))); // NOI18N
        upButton.setEnabled(false);
        upButton.setPreferredSize(new java.awt.Dimension(40, 40));
        upButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                upButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                    .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                            .addComponent(jScrollPane2)
                            .addGroup(jPanel1Layout.createSequentialGroup()
                                    .addComponent(titleLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 240,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addGap(18, 18, 18)
                                    .addComponent(priceLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                                    .addComponent(quantityLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 251,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 22,
Short.MAX_VALUE)
                                    .addComponent(downButton,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                    .addComponent(upButton,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                    .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING, false)
                            .addComponent(priceLabel,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(titleLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .addGap(18, 18, 18)
                    .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE,
315, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                            .addGroup(jPanel1Layout.createSequentialGroup()
                                    .addComponent(quantityLabel)
                                    .addContainerGap())
                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                                    .addComponent(upButton,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                    .addComponent(downButton,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE))))
        );

        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("Products");

        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel2.setText("Details");

        plusButton.setFont(new java.awt.Font("Lucida Grande", 0, 14)); // NOI18N
        plusButton.setText("Add Product");
        plusButton.setToolTipText("");
        plusButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                plusButtonActionPerformed(evt);
            }
        });

        minusButton.setFont(new java.awt.Font("Lucida Grande", 0, 14)); // NOI18N
        minusButton.setText("Remove Product");
        minusButton.setToolTipText("");
        minusButton.setEnabled(false);
        minusButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                minusButtonActionPerformed(evt);
            }
        });

        jLabel3.setText("Cost: ");

        costLabel.setText("$0.00");

        jLabel5.setText("Revenue: ");

        revenueLabel.setText("$0.00");

        jLabel7.setText("Profit: ");
```



```
        profitLabel.setText("$0.00");

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(costLabel, javax.swing.GroupLayout.PREFERRED_SIZE,
83, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jLabel5)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(revenueLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 83,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jLabel7)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(profitLabel, javax.swing.GroupLayout.PREFERRED_SIZE,
83, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(79, 79, 79)
                    .addComponent(jButton1))
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                            .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addGroup(layout.createSequentialGroup()
                                    .addContainerGap()
                                    .addComponent(plusButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                                    .addComponent(minusButton))
```

```
                        .addComponent(jScrollPane1))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                 .addContainerGap())
       );
       layout.setVerticalGroup(
          layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
          .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(jButton1)
                        .addComponent(jLabel3)
                        .addComponent(costLabel)
                        .addComponent(jLabel5)
                        .addComponent(revenueLabel)
                        .addComponent(jLabel7)
                        .addComponent(profitLabel))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(jLabel1)
                        .addComponent(jLabel2))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGroup(layout.createSequentialGroup()
                               .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 368,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                                            .addComponent(plusButton,
javax.swing.GroupLayout.DEFAULT_SIZE, 39, Short.MAX_VALUE)
                                            .addComponent(minusButton,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))
                    .addContainerGap())
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>//GEN-END:initComponents

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton1ActionPerformed
        // TODO add your handling code here:
        this.setVisible(false);
    }//GEN-LAST:event_jButton1ActionPerformed

    private void productListPropertyChange(java.beans.PropertyChangeEvent evt)
{//GEN-FIRST:event_productListPropertyChange

    }//GEN-LAST:event_productListPropertyChange

    private void productListValueChanged(javax.swing.event.ListSelectionEvent evt)
{//GEN-FIRST:event_productListValueChanged
        if (evt.getValueIsAdjusting()) {
            titleLabel.setText("");
            priceLabel.setText("");
            descriptionTextArea.setText("");
            currentProductID = inventory.getProductID(productList.getSelectedIndex());
            minusButton.setEnabled(true);
            updateProductDetails(currentProductID);
            downButton.setEnabled(true);
            upButton.setEnabled(true);
        }
    }//GEN-LAST:event_productListValueChanged

    private void downButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_downButtonActionPerformed
        int quantity = inventory.retrieveQuantity(currentProductID);
        inventory.decrement(currentProductID, 1);
        quantity--;
        if (quantity < 1) {
            clearProductDetails();
            updateInventoryList();
        }
        else {
            updateProductDetails(currentProductID);
        }
        Product t = inventory.getProductByID(currentProductID);
        t.quantity = t.quantity-1;
        updateFinancials();
        archive.saveData();
    }//GEN-LAST:event_downButtonActionPerformed

    private void upButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_upButtonActionPerformed
        inventory.increment(currentProductID, 1);
        updateProductDetails(currentProductID);
        Product t = inventory.getProductByID(currentProductID);
        t.quantity = t.quantity+1;
        updateFinancials();
        archive.saveData();
    }//GEN-LAST:event_upButtonActionPerformed

    private void minusButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_minusButtonActionPerformed
        // TODO add your handling code here:
        clearProductDetails();
        inventory.removeProduct(currentProductID);
        updateInventoryList();
        updateFinancials();
        archive.saveData();
        archive.saveFinancials();
    }//GEN-LAST:event_minusButtonActionPerformed
    /**
     * method to add new product
     */
    private void plusButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_plusButtonActionPerformed
        // TODO add your handling code here:
        clearProductDetails();
        AddNewProduct newProduct = new AddNewProduct();
        newProduct.addListener(this);
        newProduct.setVisible(true);
    }//GEN-LAST:event_plusButtonActionPerformed

    @Override
    public void newProduct() {
        updateInventoryList();
        updateFinancials();
        archive.saveData();
    }
    /**
     * method to update product information
     * @param productID
     */
    private void updateProductDetails (int productID) {
        Product product = inventory.getProductByID(productID);
        int quantityInInventory = inventory.retrieveQuantity(productID);
        int quantityInCart = cart.retrieveQuantity(productID);
        titleLabel.setText(product.name);
        DecimalFormat df2 = new DecimalFormat( "#0.00" );
        priceLabel.setText("$" + df2.format(product.price) );
        descriptionTextArea.setText(product.description);

        int quantityAvailable = quantityInInventory - quantityInCart;

        quantityLabel.setText("Quantity in stock: " + quantityAvailable);
        archive.saveData();
    }

    private void clearProductDetails() {
        titleLabel.setText("");
        priceLabel.setText("");
        descriptionTextArea.setText("");
        quantityLabel.setText("");
        downButton.setEnabled(false);
        upButton.setEnabled(false);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional)
">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look
and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
```



```
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Seller.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Seller.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Seller.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Seller.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Seller().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel costLabel;
    private javax.swing.JTextArea descriptionTextArea;
    private javax.swing.JButton downButton;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JButton minusButton;
    private javax.swing.JButton plusButton;
    private javax.swing.JLabel priceLabel;
    private javax.swing.JList productList;
    private javax.swing.JLabel profitLabel;
    private javax.swing.JLabel quantityLabel;
    private javax.swing.JLabel revenueLabel;
    private javax.swing.JLabel titleLabel;
    private javax.swing.JButton upButton;
    // End of variables declaration//GEN-END:variables
}
```

ShoppingCart.java

```
package com.store;

/**
 * Shopping Cart
 *
 * Singleton Pattern
 */

public class ShoppingCart extends ProductList{
    private static ShoppingCart shoppingCart = null;
    //Singleton Pattern

    //load from interface for serialization here
    public static ShoppingCart getCart() {
        if(shoppingCart == null) {
            shoppingCart = new ShoppingCart();

        }
        return shoppingCart;
    }
}
```

Sold.java

```
package com.store;

/**
 * Sold Class of ProductList
 *
 * Singleton Pattern
 */
public class Sold extends ProductList{
    private static Sold instance = null;

    public static Sold getInstance() {
        if(instance == null) {
            instance = new Sold();
        }
        return instance;
    }
}
```

StoreInventory.java

```
package com.store;

/**
 * Store Inventory
 *
 * Singleton Pattern
 */
public class StoreInventory extends ProductList {
    private static StoreInventory instance = null;
    //Singleton Patter
    public static StoreInventory getInstance() {
        if(instance == null) {
            instance = new StoreInventory();
        }
        return instance;
    }

}
```

UnitTests.java

```
package Tests;
import com.store.Product;
import com.store.ShoppingCart;
import com.store.Sold;
import com.store.StoreInventory;
import junit.framework.*;
import org.junit.Test;
import org.junit.Ignore;
import static org.junit.Assert.assertEquals;
/**
 * Tests for the inventory portion
 */
public class UnitTests extends TestCase {
    private StoreInventory inventory;
    private ShoppingCart cart;
```

```
@Test
public void testAddProduct() {
    inventory = StoreInventory.getInstance();
    cart = ShoppingCart.getCart();
    Product test = new Product();
    test.sku = 0;
    test.cost = 5;
    test.name = "Chili";
    test.price = 6;
    test.description = "blah";
    inventory.addProduct(test, 3);
    assertEquals(inventory.getProductID(0), test.sku);
}
@Test
public void testGetNewProductId() {
    inventory = StoreInventory.getInstance();
    cart = ShoppingCart.getCart();
    Product test = new Product();
    test.sku = 1;
    test.cost = 5;
    test.name = "Beans";
    test.price = 6;
    test.description = "blah";
    inventory.addProduct(test, 3);
    assertEquals(inventory.getCount(), 6);
}
@Test
public void testGetQuantity() {
    inventory = StoreInventory.getInstance();
    cart = ShoppingCart.getCart();
    Product test = new Product();
    test.sku = 2;
    test.cost = 2;
    test.name = "Rice";
    test.price = 3;
    test.description = "blah";
    inventory.addProduct(test, 22);
    assertEquals(inventory.retrieveQuantity(2), 22);
}
@Test
public void testRemove() {
    inventory = StoreInventory.getInstance();
    cart = ShoppingCart.getCart();
    Product test = new Product();
```

```
        test.sku = 3;
        test.cost = 2;
        test.name = "Plantains";
        test.price = 3;
        test.description = "blah blah";
        Integer count = inventory.getCount();
        inventory.addProduct(test, 4);
        inventory.removeProduct(3);
        assertEquals(inventory.getCount(), (int)count);
    }


}
```