Andrew Stallone

Table of Contents

Updated Use Case

Log In

1. The user enters login or registers account if new user.
2. The system authentication class authenticates the user.
3. The system will show different windows for seller or buyer.

Customer adds items to shopping cart

1. The Customer clicks the button add to cart
2. The Cart class adds a new item object to the cart list
3. The customer has the option to continue shopping or check out

Customer reviews product details

1. The customer clicks on an item
2. The system shows the variables in the item object such as product name, description, price and quantity available.
3. The customer may click the back button to see the product list.

Customer reviews /updates shopping cart

1. The customer clicks on the shopping cart button
2. The Cart class queue shows the list of item objects as well as the names, prices and subtotal.
3. The customer can change the quantity to update the list

Customer checks out

1. The customer clicks on the shopping cart
2. The customer clicks checkout
3. The System takes the item out of stock.

Seller reviews/updates inventory

1. The seller selects the items for sale queue
2. The seller enters the new quantity
3. The system updates the quantity

Seller adds new product

1. The seller clicks inventory

2. The seller selects add new item
3. The system creates a new item object and append it to the end of the queue
4. The seller enters the product name, description, price and quantity
5. The seller hits publish to show item for sale

Seller checks revenue

1. The seller clicks on financial reports
2. The system returns the revenue earned and list of items sold with quantities

CRC Cards

User	
Knows name Knows address Knows userid Knows password Knows if seller of customer	

Customer	
Knows User info Knows previous order info	User Order

Seller	
Knows User info Knows Customer order info Knows inventory	User Inventory Order

SellerInventory

Knows inventory size Knows current assets Allows updating of inventory	Item

Login	
Knows Seller or Customer Loads correct windows	

ShoppingCart	
Knows current item added Shows subtotal Add Remove update	Item

Order	
Knows orderId Knows details Knows price Knows product names Knows orderDate Submit order Update seller inventory	Shopping cart SellerInventory

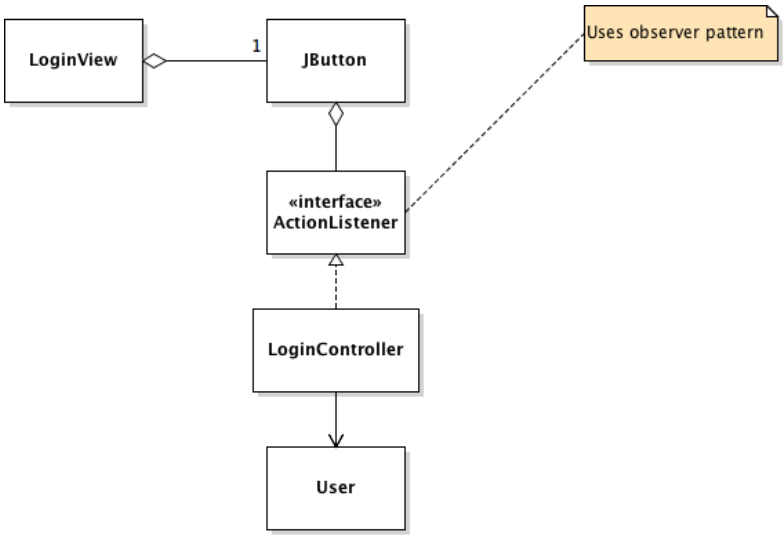
Item

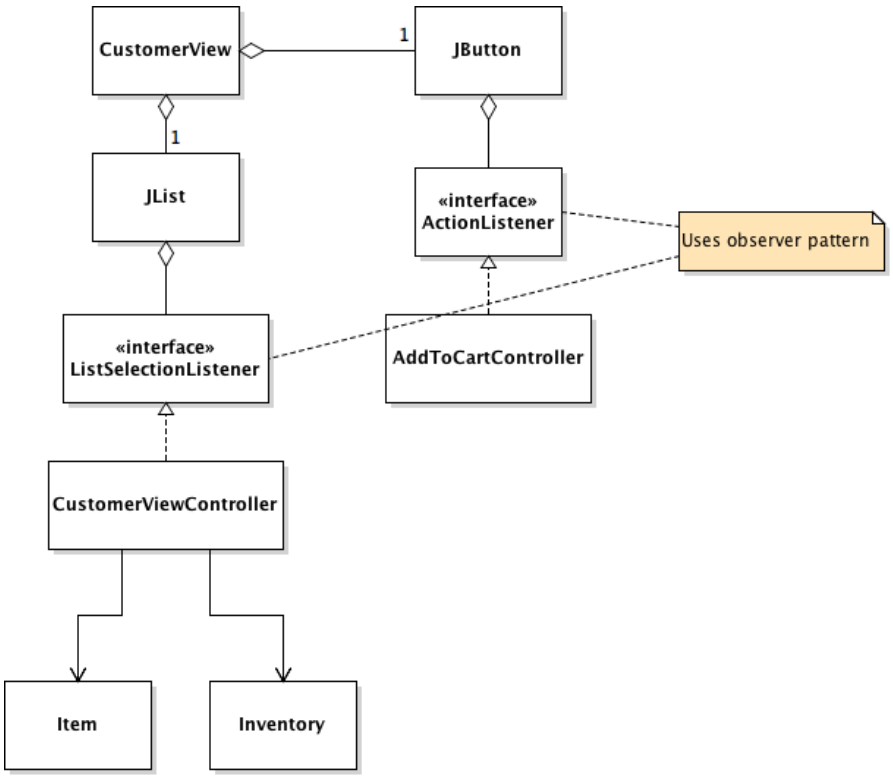
Knows name Knows description Knows productID Knows price	

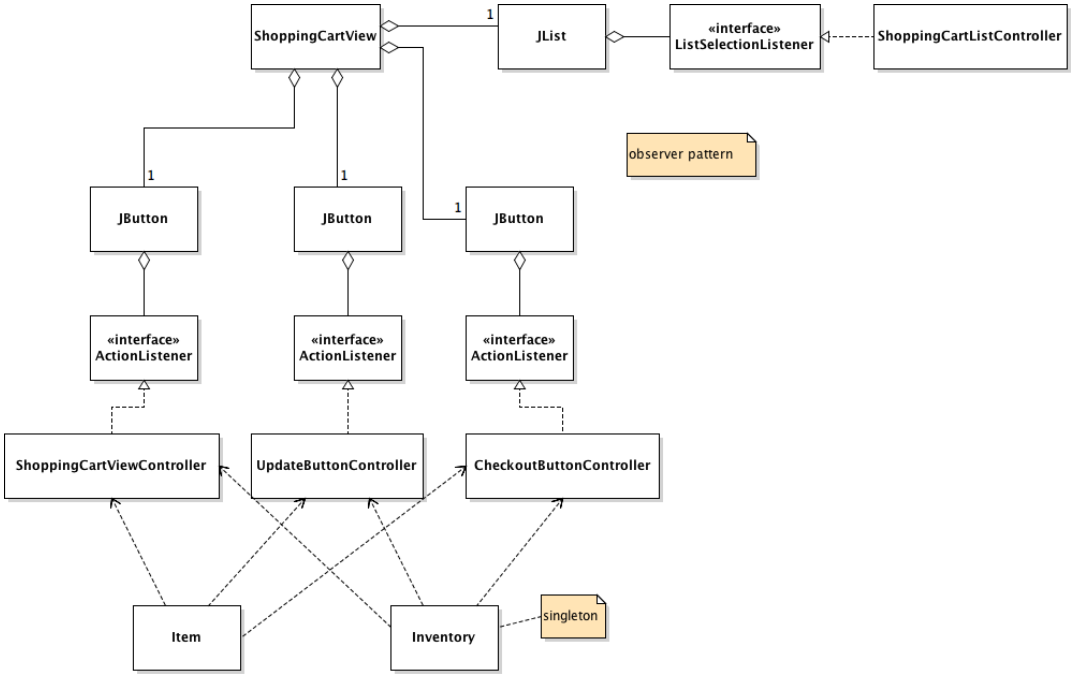
Payment	
Knows orderInfo ProcessPayment	Order

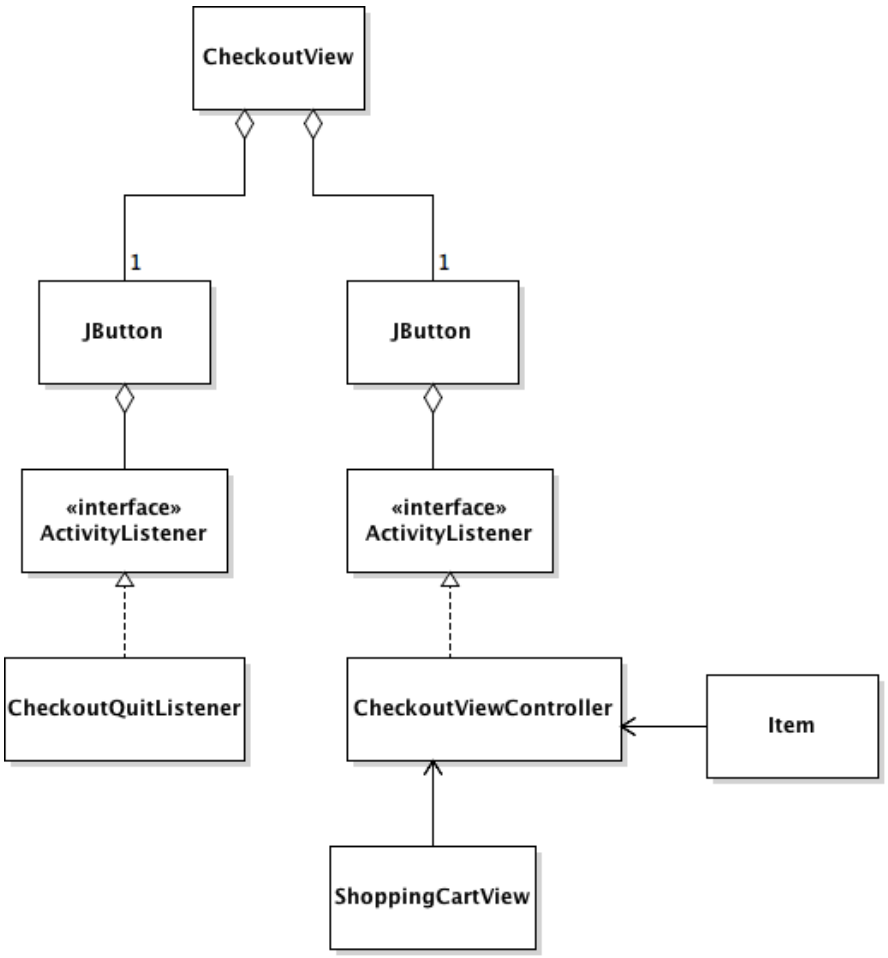
Reciept	
Knows payment type Knows orderInfo Sends receipt Update seller customer order list	Payment Seller Customer

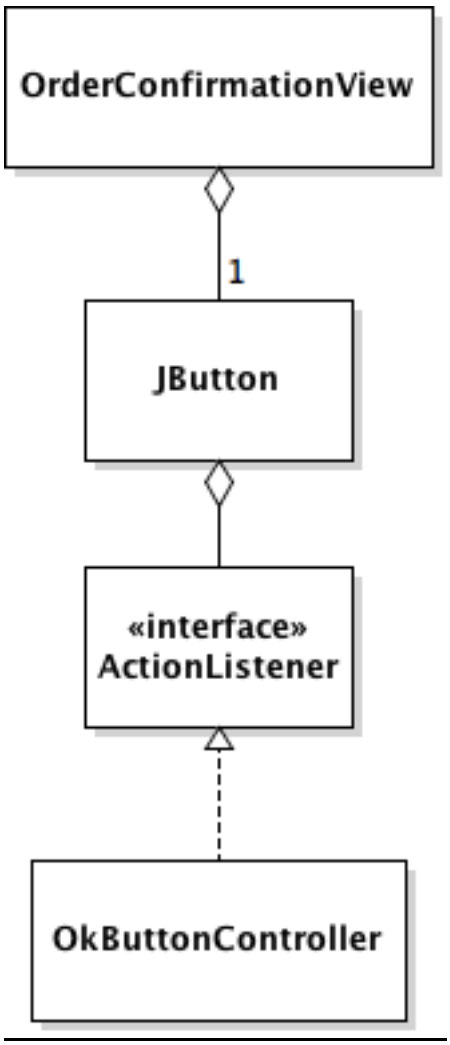
UML Diagrams

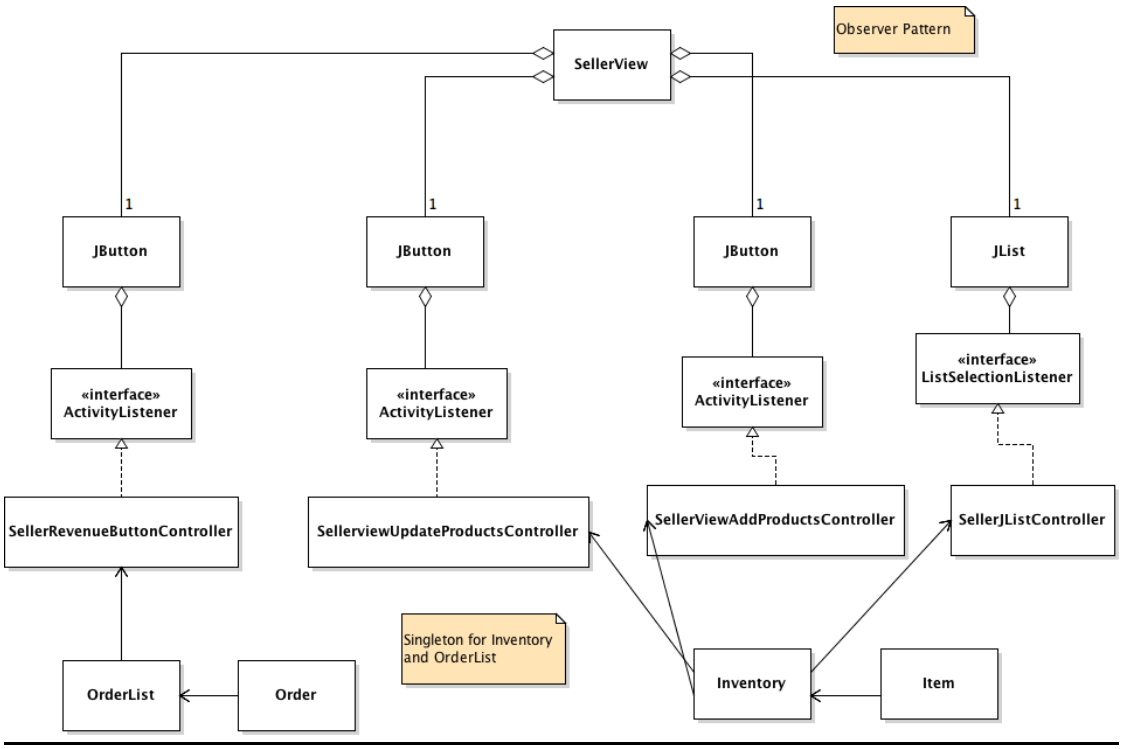


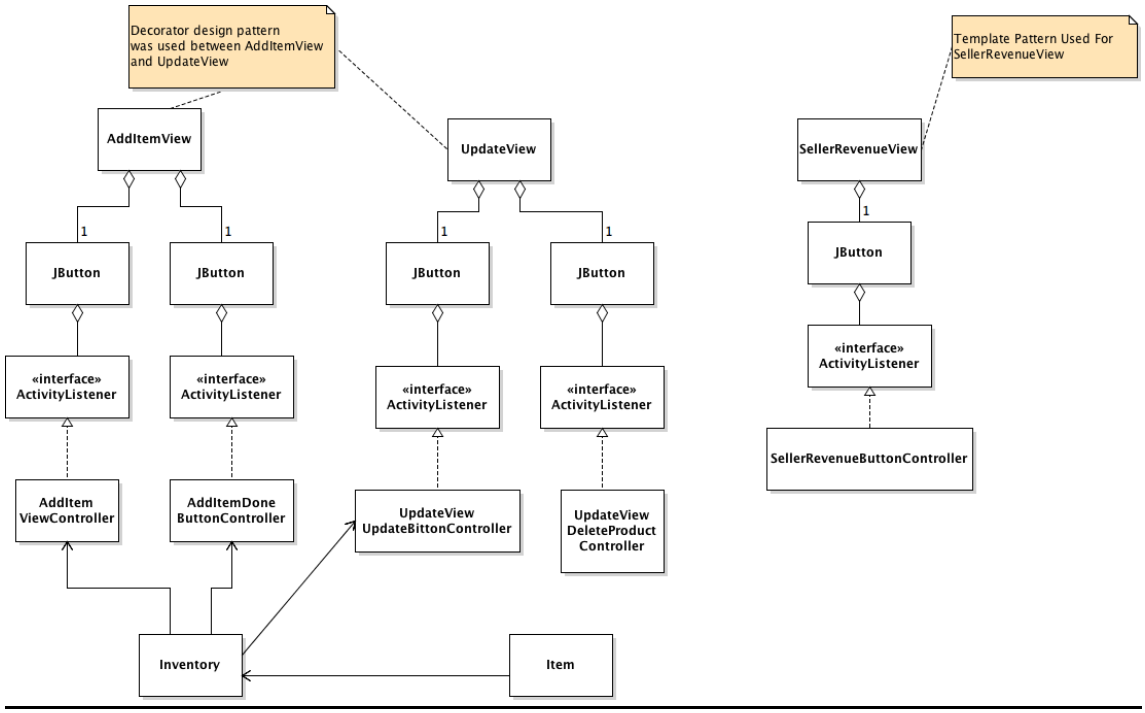

CustomerView
1
JButton
1
JList
«interface»
ActionListener
Uses observer pattern
«interface»
ListSelectionListener
AddToCartController
CustomerViewController
Item
Inventory


ShoppingCartView
1
JList
«interface»
ListSelectionListener
ShoppingCartListController
observer pattern
1
1
1
JButton
JButton
JButton
«interface»
ActionListener
«interface»
ActionListener
«interface»
ActionListener
ShoppingCartViewController
UpdateButtonController
CheckoutButtonController
Item
Inventory
singleton

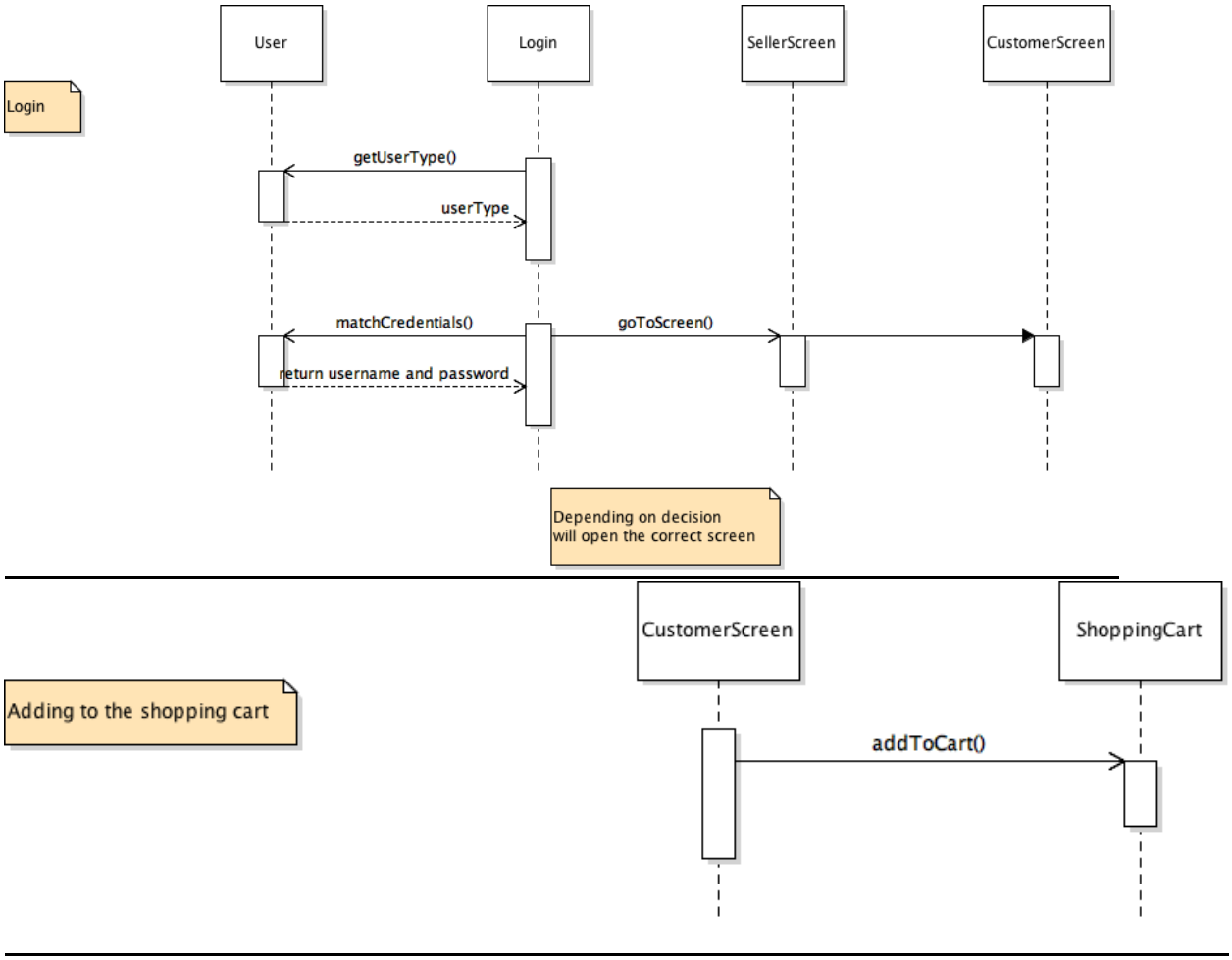

CheckoutView
1
1
JButton
JButton
«interface»
ActivityListener
«interface»
ActivityListener
CheckoutQuitListener
CheckoutViewController
Item
ShoppingCartView


OrderConfirmationView
1
JButton
«interface»
ActionListener
OkButtonController

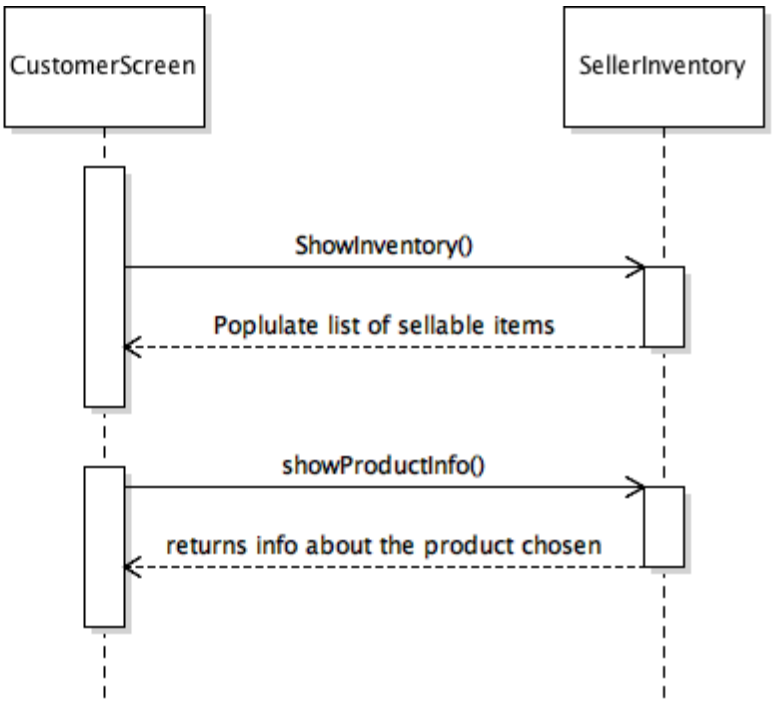

Observer Pattern
SellerView
1
1
1
1
JButton
JButton
JButton
JList
«interface»
ActivityListener
«interface»
ActivityListener
«interface»
ActivityListener
«interface»
ListSelectionListener
SellerRevenueButtonController
SellerviewUpdateProductsController
SellerViewAddProductsController
SellerJListController
Singleton for Inventory
and OrderList
OrderList
Order
Inventory
Item


Decorator design pattern
was used between AddItemView
and UpdateView
Template Pattern Used For
SellerRevenueView
AddItemView
UpdateView
SellerRevenueView
1
1
1
1
1
JButton
JButton
JButton
JButton
JButton
«interface»
ActivityListener
«interface»
ActivityListener
«interface»
ActivityListener
«interface»
ActivityListener
«interface»
ActivityListener
AddItem
ViewController
AddItemDone
ButtonController
UpdateView
UpdateBittonController
UpdateView
DeleteProduct
Controller
SellerRevenueButtonController
Inventory
Item

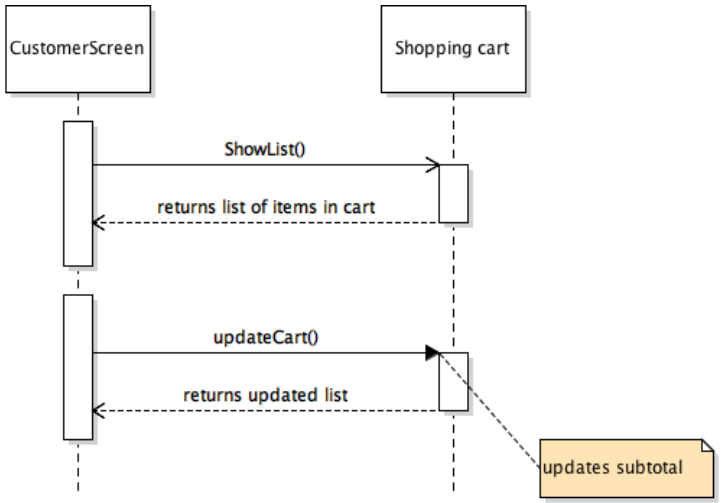
Sequence Diagrams for Use Cases

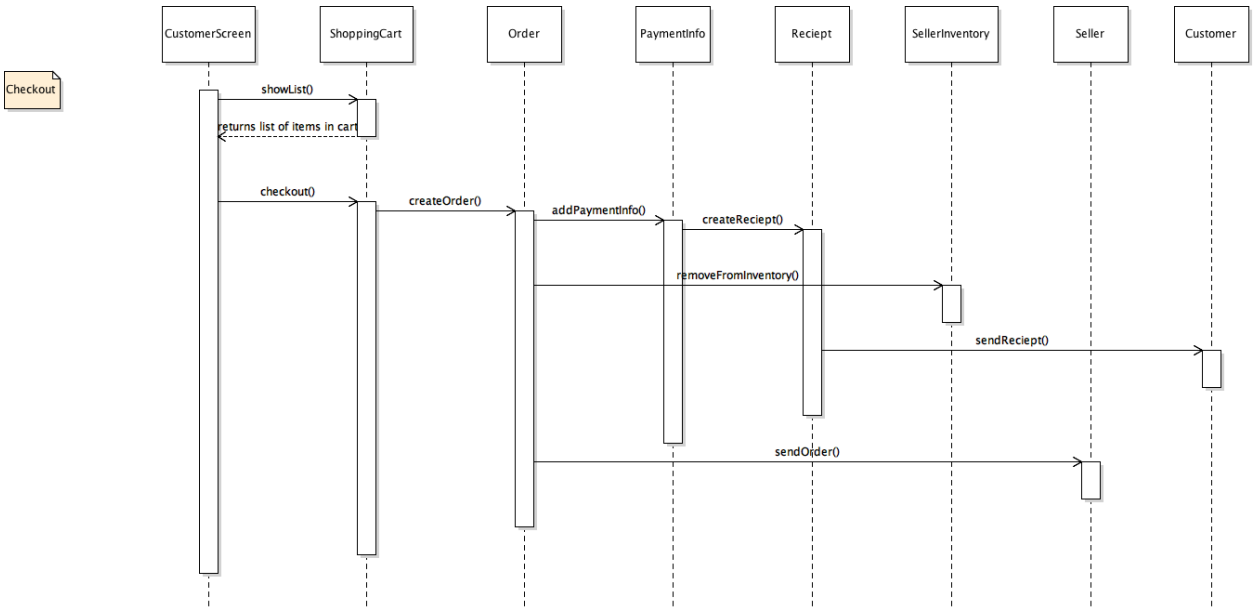


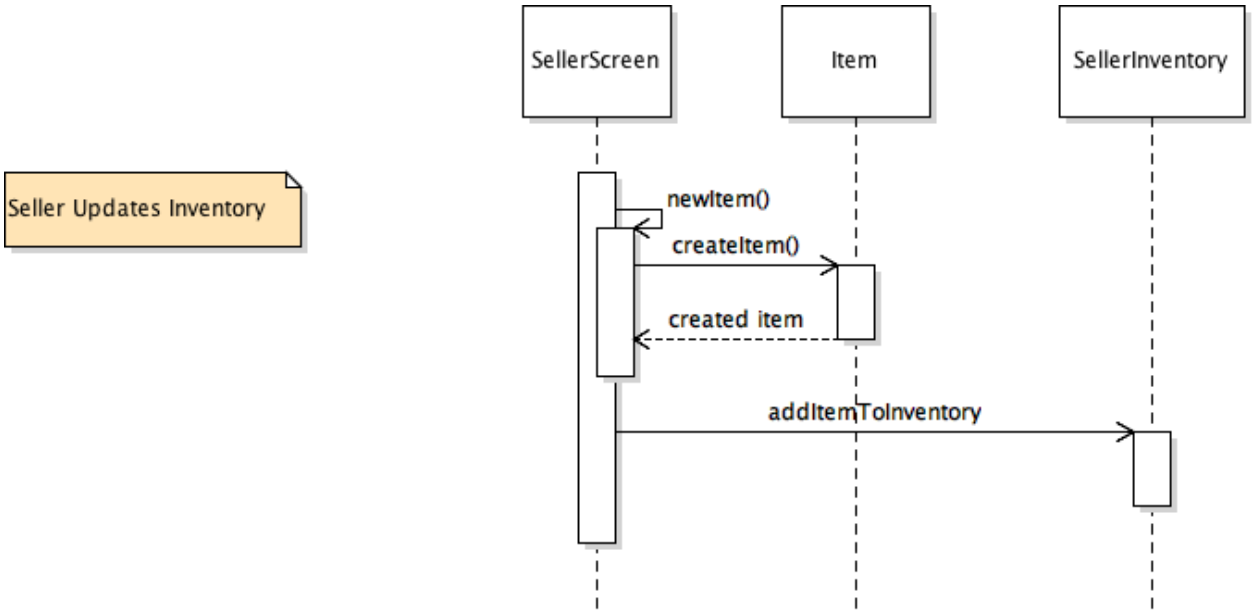
View Product Details

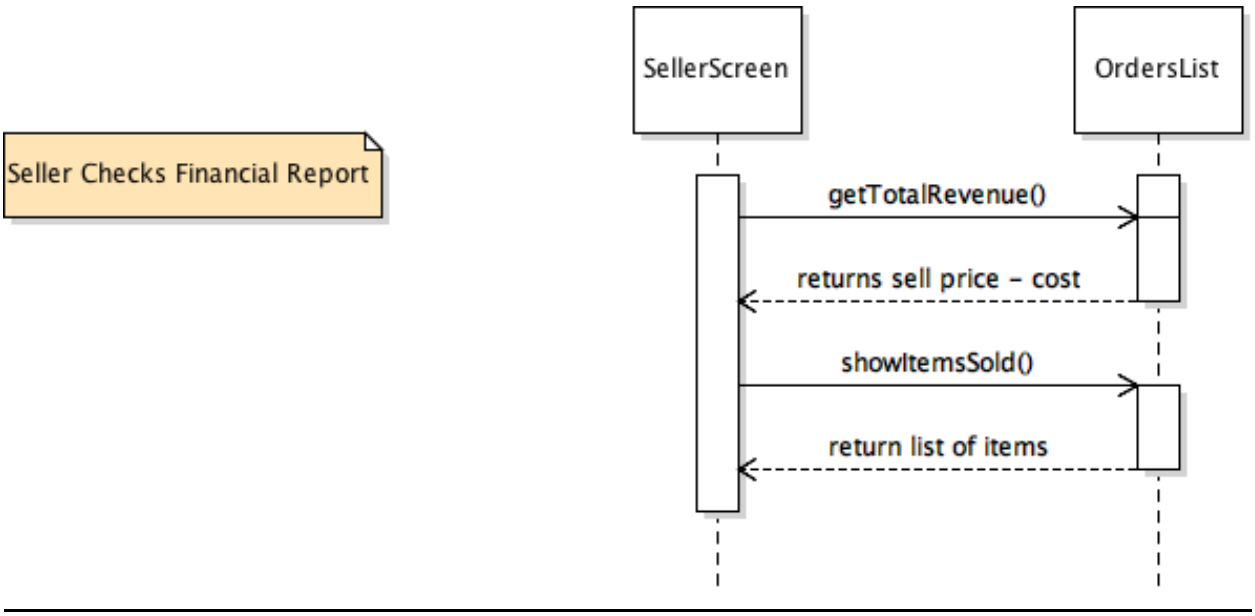


update/review shopping cart




Checkout
CustomerScreen
ShoppingCart
Order
PaymentInfo
Reciept
SellerInventory
Seller
Customer
showList()
returns list of items in cart
checkout()
createOrder()
addPaymentInfo()
createReciept()
removeFromInventory()
sendReciept()
sendOrder()


Seller Updates Inventory
SellerScreen
Item
SellerInventory
newItem()
createItem()
created item
addItemToInventory


Seller Checks Financial Report
SellerScreen
OrdersList
getTotalRevenue()
returns sell price - cost
showItemsSold()
return list of items

Inventory.java

```
package client;

import java.util.ArrayList;

/**
 *
 * @author Andrew Stallone Inventory Singleton class that holds items
 */
public class Inventory {
	private ArrayList<Item> itemList = new ArrayList<Item>();
	private static Inventory myObj;

	/**
	 * Constructor for the Inventory Object
	 *
	 * adds 3 items to the list for demo purposes
	 *
	 */
	private Inventory() {
		itemList.add(new Item("Macbook Pro", "256gb ssd, 16gb ram, 15\" screen", 5, 1999.00, 500.00));
		itemList.add(new Item("iPad Pro", "32gb ssd, 9.7\" screen", 25, 599.00, 200.00));
		itemList.add(new Item("iPad Pro", "32gb ssd, 12.9\" screen", 28, 799.00, 300.00));
		itemList.add(new Item("iPod touch", "16gb ssd holds 5,000 songs", 32, 149.00, 22.50));

	}

	/**
	 * getInstance
	 *
	 *
	 * @return a reference to the inventory object
	 */
	public static Inventory getInstance() {
		if (myObj == null) {
			myObj = new Inventory();
		}
		return myObj;
	}

	/**
	 * getList
	 *
	 * @return the list of items
	 */
	public ArrayList<Item> getList() {
		return itemList;
	}

}
```

Customer.java

```
package client;
/**
 *
 * @author Andrew Stallone
 * Customer class to create a Customer
 */
public class Customer extends User{

public Customer(String name, String username, String password) {
		super(name, username, password);
		// TODO Auto-generated constructor stub
	}

@Override
	public void setId(String id) {
		// set to customer
		super.setId(id);
	}


}
```

User.java

```
package client;
/**
 *
 * @author Andrew Stallone
 * @about User class to create a user
 */
public class User {
	private String name = "";
	private String identifier = "";  //seller or customer to load the correct screen
	private String username = "";
	private String password = "";

	/**
	 * Constructor for the User Object
	 *
	 * @param name of user
	 * @param username
	 * @param password
	 *
	 */

	public User (String name, String username, String password){
		this.name = name;
		this.username = username;
		this.password = password;
	}

	/**
	 * resetPassword
	 *
	 * @param new password
	 * @about resets password
```

```
 *
 */

public void resetPassword(String password){
        this.password = password;
}

/**
 *getPassword
 *
 * @about returns current password
 *
 */
public String getPassword(){return password;}

/**
 * getName
 *
 * @about returns current name
 *
 */
public String getName(){return name;};

/**
 * getUserName
 *
 * @about returns username
 *
 */
public String getUserName(){return username;};

/**
 * SetId
 *
 * @about sets ID
 * @return
 *
 */
public void setId(String id){
        this.identifier = id;

}

/**
 * getId
 *
 * @about gets id
 * @return id
 *
 */
public String getId(){
        return identifier;
}


}
```

Seller.java

```
package client;

/**
 *
 * @author Andrew Stallone
 * @about Seller class to create a Seller
 */
public class Seller extends User {

	public Seller(String name, String username, String password) {
		super(name, username, password);
		// TODO Auto-generated constructor stub
	}

	@Override
	public void setId(String id) {
		// set to seller
		super.setId(id);
	}

}
```

ShoppingCart.java

```
package client;

import java.util.LinkedList;

/**
 *
 * @author Andrew Stallone ShoppingCart class that holds items for checkout
 */

public class ShoppingCart {

	private LinkedList<Item> shoppingCart = new LinkedList<Item>();

	public ShoppingCart() {

	}

	/**
	 * getList
	 *
	 * @return returns the list of items in the shopping cart
	 *
	 */
	public LinkedList<Item> getList() {
		return shoppingCart;
	}

	/**
	 * addItem
	 *
	 * @param Item
	 *          to be added to the list if the shopping cart already contains
```

```
	 *		this item it increments the quantity counter.
	 */
	public void addItem(Item i) {

		if (shoppingCart.contains(i)) {
			i.setAmountInCart(i.getAmountInCart() + 1);
		} else {
			shoppingCart.add(i);
			i.setAmountInCart(i.getAmountInCart() + 1);
		}

	}

	/**
	 * removeItem
	 *
	 * @param Item
	 *		to be removed to the list if the shopping cart is empty
	 *		nothing happens otherwise the item is removed.
	 */
	public void removeItem(Item i) {
		if (!shoppingCart.isEmpty()) {
			shoppingCart.remove(i);
		}
	}

	/**
	 * getSubtotal
	 *
	 * for each item in the shopping cart the price is added to get the total.
	 *
	 * @return subtotal
	 */
	public double getSubtotal() {
		double subtotal = 0.0;
		for (Item i : shoppingCart) {
			subtotal += i.getPrice();
		}
		return subtotal;
	}

	/**
	 * getSubtotal
	 *
	 * gives the amount of items in the cart.
	 *
	 * @return size of cart
	 */
	public int getSize() {
		return shoppingCart.size();
	}
}
```

OrderList.java

```
package client;

import java.util.ArrayList;

/**
 *
 * @author Andrew Stallone Singleton OrderList class that holds orders
 */

public class OrderList {
	private ArrayList<Order> orderList = new ArrayList<Order>();
	private static OrderList myObj;

	/**
	 * Private Constructor for the OrderList Object
	 *
	 *
	 */

	private OrderList() {

	}

	/**
	 * getInstance
	 *
	 * @return the instance of the class
	 *
	 *
	 */

	public static OrderList getInstance() {
		if (myObj == null) {
			myObj = new OrderList();
		}
		return myObj;
	}

	/**
	 * add
	 *
	 * adds an order to the list
	 *
	 *
	 */
	public void add(Order o) {
		orderList.add(o);
	}

	/**
	 * getOrderList
	 *
	 * @return the list of orders
	 *
	 *
	 */
	public ArrayList<Order> getOrderList() {
```

```
		return orderList;
	}

	/**
	 * getTotalCost
	 *
	 * @return the total cost of all items in the list
	 *
	 *
	 */
	public double getTotalCost() {
		double cost = 0;
		for (Order o : orderList) {
			cost += o.totalCost();
		}
		return cost;
	}

	/**
	 * getTotalRevenue
	 *
	 * @return the total revenue of all items in the list
	 *
	 *
	 */
	public double getTotalRevenue() {
		double revenue = 0;
		for (Order o : orderList) {
			revenue += o.orderTotal();
		}
		return revenue;
	}

}
```

Order.java

```
package client;

import java.util.LinkedList;

public class Order {
	private LinkedList<Item> listOfPurchasedProducts = new LinkedList<Item>();
	public Order(){

	}

	public void addItem(Item i){
		listOfPurchasedProducts.add(i);
	}

	public double orderTotal(){
		double total = 0;
		for(Item i : listOfPurchasedProducts){
			total += i.getPrice() * i.getOrderAmount();
		}
		return total * 1.06;
```

```
	}

	public double totalCost(){
		double total = 0;
		for(Item i : listOfPurchasedProducts){
			total += i.getCost() * i.getOrderAmount();
		}
		return total;
	}
}
```

Item.java

```
package client;
/**
 *
 * @author Andrew Stallone
 * Item class
 */
public class Item {
	private String name = "";
	private String description = "";
	private int quantity = 0;
	private double price = 0.0;
	private int amountInCart = 0;
	private double cost = 0;
	private int orderAmount = 0;

	/**
	 * Constructor for the Student Object
	 *
	 * @param name of item
	 * @param description of item
	 * @param quantity of item
	 * @param price of item
	 * @param cost of item
	 *
	 */
	public Item(String name, String description, int quantity, double price, double cost){
		this.name = name;
		this.description = description;
		this.quantity = quantity;
		this.price = price;
		this.cost = cost;
	}

	/**
	 * getOrderAmount
	 * gets the total order amount
	 * @return the amount of the order
	 *
	 */
	public int getOrderAmount(){return orderAmount;}

	/**
	 * setOrderAmount
	 * sets the total order amount
	 *
```

```
 */
public void setOrderAmount(int orderAmount){
		this.orderAmount = orderAmount;
}

/**
 * getCost
 * gets the cost amount
 * @return the cost of the item
 *
 */
public double getCost(){return cost;}

/**
 * setCost
 * sets the cost of the item
 *
 */
public void setCost(double cost){
		this.cost = cost;
}

/**
 * getAmountInCart
 * gets the item amount in the cart
 * @return the amount items in the cart
 *
 */
public int getAmountInCart(){
		return amountInCart;
}

/**
 * setAmountInCart
 * sets the amount in the cart
 *
 *
 */
public void setAmountInCart(int amount){
		amountInCart = amount;
}

/**
 * setPrice
 * sets the price of the item
 *
 *
 */
public void setPrice(double price){
		this.price = price;
}

/**
 * getPrice
 * gets the pice of item
 * @return the price of the item
 *
 */
```

```
public double getPrice(){
		return price;
}

/**
 * setDecscription
 * sets the description of the item
 *
 */
public void setDecscription(String description){
		this.description = description;
}

/**
 * getDescription
 * gets the item description
 * @return description of the item
 *
 */
public String getDescription(){
		return description;
}

/**
 * setitemQuantity
 * sets the item quantity
 *
 *
 */
public void setQuantity(int quantity){
		this.quantity = quantity;
}

/**
 * getQuantity
 * gets the quantity of the item
 * @return the quantity of the item
 *
 */
public int getQuantity(){
		return quantity;
}

/**
 * getOrderAmount
 * gets the total order amount
 * @return the amount of the order
 *
 */
public void setName(String name){
		this.name = name;
}

/**
 * getName
 * gets the name of the item
 * @return the name of the item
 *
```

```
	 */
	public String getName(){
		return name;
	}
}
```

AddItemDoneButtonController.java

```
package gui;

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

/**
 *
 * @author Andrew Stallone
 * @about Done button controller on the add item screen
 */
public class AddItemDoneButtonController implements ActionListener{

	private AddItemView addItemView;

	/**
	 * Constructor for the AddItemDoneButtonController Object
	 * @param The AddItemView
	 */
	public AddItemDoneButtonController(AddItemView addItemView) {
		this.addItemView = addItemView;
	}
	@Override
	public void actionPerformed(ActionEvent e) {
		addItemView.setVisable(false);

	}

}
```

AddItemView.java

```
package gui;

import java.awt.BorderLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.event.ActionListener;
import java.util.ArrayList;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
```

```
import javax.swing.event.ListSelectionListener;

import client.Inventory;
import client.Item;
/**
 *
 * @author Andrew Stallone
 * @about View to add an item
 */
public class AddItemView {

	private JFrame frame = new JFrame();
	private JPanel mainPanel = new JPanel();
	private JPanel editPanel = new JPanel();
	private JPanel buttonPanel = new JPanel();
	private JButton doneButton = new JButton("Done");
	private JButton addProduct = new JButton("Add New Item");
	private JTextField nameTextField = new JTextField();
	private JTextField quantityTextField = new JTextField();
	private JTextField priceTextField = new JTextField();
	private JTextArea descriptionTextArea = new JTextArea();
	private JTextField costTextField = new JTextField();
	private BorderLayout borderLayout = new BorderLayout();
	private GridLayout gridLayout = new GridLayout(1,2);
	private GridBagLayout gridbag = new GridBagLayout();
	private GridBagConstraints c = new GridBagConstraints();
	private Inventory inventory = Inventory.getInstance();
	private ArrayList<Item> itemList = inventory.getList();
	private ActionListener done = new AddItemDoneButtonController(this);
	private SellerView sellerview;

	/**
	 * Constructor for the AddItemView Object
	 *
	 * @param SellerView
	 * @about setup the view
	 *
	 */
	public AddItemView(SellerView sellerView){
		this.sellerview = sellerView;
		ActionListener add = new AddItemViewController(this,sellerView);
		frame.add(mainPanel);
		mainPanel.setLayout(borderLayout);
		c.fill = GridBagConstraints.HORIZONTAL;
		mainPanel.add(buttonPanel, BorderLayout.SOUTH);
		buttonPanel.setLayout(gridLayout);
		buttonPanel.add(addProduct);
		buttonPanel.add(doneButton);
		addProduct.addActionListener(add);
		doneButton.addActionListener(done);
		initAddNewItem();
		frame.setSize(600, 200);
		frame.pack();
		frame.setVisible(true);
	}
	/**
	 * initAddNewItem
	 *
```

```
 *
 * @about sets up the editpanel with a gridbag layout creates all contraints and labels.
 *
 */
public void initAddNewItem(){

                        editPanel.setLayout(gridbag);

                        JLabel name = new JLabel("Name: ");
                        name.setHorizontalAlignment(SwingConstants.RIGHT);
                        c.ipady = 10;
                        c.weightx = 0.5;
                        c.gridwidth = 1;
                        c.gridx = 0;
                        c.gridy = 0;
                        gridbag.setConstraints(name, c);
                        editPanel.add(name);

                        c.ipady = 10;
                        c.weightx = 0.5;
                        c.gridx = 1;
                        c.gridy = 0;
                        gridbag.setConstraints(nameTextField, c);
                        editPanel.add(nameTextField);
                        nameTextField.setText("name");

                        JLabel price = new JLabel("Price: ");
                        price.setHorizontalAlignment(SwingConstants.RIGHT);
                        c.ipady = 10;
                        c.gridx = 2;
                        c.gridy = 0;
                        gridbag.setConstraints(price, c);
                        editPanel.add(price);

                        c.ipady = 10;
                        c.gridx = 3;
                        c.gridy = 0;
                        gridbag.setConstraints(priceTextField, c);
                        editPanel.add(priceTextField);
                        priceTextField.setText("price");

                        JLabel quantity = new JLabel("Quantity: ");
                        quantity.setHorizontalAlignment(SwingConstants.RIGHT);
                        c.ipady = 10;
                        c.gridx = 4;
                        c.gridy = 0;
                        gridbag.setConstraints(quantity, c);
                        editPanel.add(quantity);

                        c.ipady = 10;
                        c.gridx = 5;
                        c.gridy = 0;
                        gridbag.setConstraints(quantityTextField, c);
                        editPanel.add(quantityTextField);
                        quantityTextField.setText("quantity");

                        JLabel description = new JLabel("Description: ");
                        description.setHorizontalAlignment(SwingConstants.RIGHT);
```

```
                        description.setVerticalAlignment(SwingConstants.TOP);
                        c.ipady = 40;
                        c.weightx = 0.0;
                        c.gridwidth = 1;
                        c.gridx = 0;
                        c.gridy = 1;
                        gridbag.setConstraints(description, c);
                        editPanel.add(description);

                        c.ipady = 40;
                        c.weightx = 0.0;
                        c.gridwidth = 5;
                        c.gridx = 1;
                        c.gridy = 1;
                        gridbag.setConstraints(descriptionTextArea, c);
                        editPanel.add(descriptionTextArea);

                        JLabel costLabel = new JLabel("cost");
                        costLabel.setHorizontalAlignment(SwingConstants.RIGHT);
                        c.ipady = 10;
                        c.gridx = 0;
                        c.gridy = 5;
                        c.gridwidth = 1;
                        gridbag.setConstraints(costLabel, c);
                        editPanel.add(costLabel);


                        c.ipady = 10;
                        c.gridx = 1;
                        c.gridy = 5;
                        gridbag.setConstraints(costTextField, c);
                        editPanel.add(costTextField);
                        costTextField.setText("Cost");


                        mainPanel.add(editPanel, BorderLayout.CENTER);

                }
        /**
         * setVisable
         *
         * @param boolean true to see the view false to hide.
         * @about sets the frame visibilty
         *
         */
        public void setVisable(boolean visable){
                frame.setVisible(visable);
        }
        /**
         * addToInventory
         * @about adds an item to the singleton class Inventory and displays the labels
         *
         */
        public void addToInventory(){
                Item i = new Item(nameTextField.getText(), descriptionTextArea.getText(),
Integer.parseInt(quantityTextField.getText()),
                                Double.parseDouble(priceTextField.getText()),
Double.parseDouble(costTextField.getText()));
```

```
                itemList.add(i);
                nameTextField.setText("");
                descriptionTextArea.setText("");
                quantityTextField.setText("");
                priceTextField.setText("");

        }


}
```

AddItemViewController.java

```
package gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
*
* @author Andrew Stallone
* @about ActionListener for the add button in the addItemView
*/

public class AddItemViewController implements ActionListener{

        private AddItemView addItemView;
        private SellerView sellerView;
        /**
         * Constructor for the AddItemViewController Object
         *
         * @param AddItemView
         * @param SellerView
         *
         */
        public AddItemViewController(AddItemView addItemView, SellerView sellerView) {
                this.addItemView = addItemView;
                this.sellerView = sellerView;
        }
        @Override
        public void actionPerformed(ActionEvent e) {
                addItemView.addToInventory();
                sellerView.updateInventory();


        }
}
```

AddToCartController.java

```
package gui;

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

import client.ShoppingCart;
/**
```

```
 *
 * @author Andrew Stallone
 * @about ActionListener for the add to cart button in the CustomerView
 */
public class AddToCartController implements ActionListener{
private CustomerView customerView;
private ShoppingCart shoppingCart;
private ShoppingCartView shoppingCartView;

/**
 * Constructor for the AddToCartController Object
 *
 * @param CustomerView

 *
 */
	public AddToCartController(CustomerView c){
		customerView = c;
	}


		@Override
		public void actionPerformed(ActionEvent e) {
			shoppingCart = customerView.getShoppingCart();
			if(shoppingCart.getSize() == 0){
				customerView.addToCart();
				shoppingCartView = new ShoppingCartView(shoppingCart,customerView);


			}else{
				customerView.addToCart();
				shoppingCartView.repaint(shoppingCart);
				shoppingCartView.setVisable();
			}



	}
}
```

CheckoutButtonController

```
package gui;

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

import client.ShoppingCart;
/**
 *
 * @author Andrew Stallone
 * @about ActionListener for the checkout button in the shopping cart view
 */

public class checkoutButtonController implements ActionListener{
```

```
	private ShoppingCartView shoppingCartView;
	private CustomerView customerView;
	/**
	 * Constructor for the checkoutButtonController Object
	 *
	 * @param ShoppingCartView
	 * @param customerView
	 *
	 */

	public checkoutButtonController(ShoppingCartView shoppingCartView, CustomerView customerView){
		this.shoppingCartView = shoppingCartView;
		this.customerView = customerView;
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		CheckoutView checkoutView = new CheckoutView(shoppingCartView,customerView);
		shoppingCartView.hideCartView();

	}

}
```

CheckoutQuitListener.java

```
package gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
 *
 * @author Andrew Stallone
 * @about ActionListener for the quit button in the checkout view
 */
public class CheckoutQuitListener implements ActionListener{

	private CheckoutView checkoutView;
	private CustomerView customerView;
	private ShoppingCartView shoppingCartView;

	/**
	 * Constructor for the CheckoutQuitListener Object
	 *
	 * @param CheckoutView
	 * @param ShoppingCartView
	 * @param CustomerView
	 *
	 */
	public CheckoutQuitListener(CheckoutView checkoutView, ShoppingCartView shoppingCartView, CustomerView
customerView){
		this.checkoutView = checkoutView;
		this.customerView = customerView;
		this.shoppingCartView = shoppingCartView;
	}

	@Override
```

```
		public void actionPerformed(ActionEvent e) {
			customerView.showCustomerView();
			shoppingCartView.setVisable();
			checkoutView.hideView();

		}
}
```

CheckoutView.java

```
package gui;

import java.awt.BorderLayout;
import java.awt.Desktop.Action;
import java.awt.GridLayout;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.LinkedList;

import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;

import client.Inventory;
import client.Item;
import client.Order;
import client.OrderList;
import client.ShoppingCart;
/**
 *
 * @author Andrew Stallone
 * @about CheckoutView sets up the view for the checkout view
 */
public class CheckoutView {
		private Inventory productList = Inventory.getInstance();
		private ShoppingCart shoppingCart;
		private ShoppingCartView shoppingCartView;
		private LinkedList<Item> shoppingCartLinkedList = new LinkedList<Item>();
		private JFrame frame = new JFrame("CheckoutView");
		private JPanel checkout = new JPanel();
		private JScrollPane scrollPane = new JScrollPane();
		private BorderLayout borderLayout = new BorderLayout();
		private GridLayout gridLayout = new GridLayout(5,1);
		private JPanel totals = new JPanel();
		private JLabel subtotal = new JLabel("Subtotal");
		private JLabel tax = new JLabel("Tax");
		private JLabel total = new JLabel("Total");
		private JButton pay;
		private JButton Cancel;
		private JList<String> itemJList;
		private CustomerView customerView;
		private ActionListener payListener = new CheckoutViewController(this);
```

```
	/**
	 * Constructor for the CheckoutView Object
	 *
	 * @param ShoppingCartView
	 * @param CustomerView
	 * @about does initial setup for the view
	 *
	 */
	public CheckoutView(ShoppingCartView shoppingCarView, CustomerView customerView){
		this.shoppingCartView = shoppingCarView;
		this.customerView = customerView;
		ActionListener cancelListener = new CheckoutQuitListener(this, shoppingCartView, customerView);
		Cancel = new JButton("Cancel");
		Cancel.addActionListener(cancelListener);
		shoppingCartLinkedList = shoppingCartView.getShoppingCart();
		init();

	}
	/**
	 * init
	 *
	 * @about does initial setup for the view
	 *
	 */
	public void init(){
		pay = new JButton("Pay");
		pay.addActionListener(payListener);
		shoppingCartView.hideCustomerView();
		itemJList = populateProductList();
		scrollPane.setViewportView(itemJList);
		totals.setLayout(gridLayout);
		totals.add(subtotal);
		totals.add(tax);
		totals.add(total);
		totals.add(pay);
		totals.add(Cancel);
		checkout.setLayout(borderLayout);
		checkout.add(scrollPane, BorderLayout.CENTER);
		checkout.add(totals, BorderLayout.SOUTH);
		frame.add(checkout);
		showDetails();
		frame.pack();
		frame.setSize(200, 500);
		frame.setVisible(true);
	}
	/**
	 * populateProductList
	 *
	 * @about constructs the defaultListModel needed for the jlist
	 *
	 */
public JList<String> populateProductList(){

		DefaultListModel<String> model = new DefaultListModel<>();
		for(Item i: shoppingCartLinkedList){
			model.addElement(i.getName()+ " X " + Integer.toString(i.getAmountInCart()) + " = " +
		Double.toString(i.getPrice() * i.getAmountInCart()));
		}
```

```
                return new JList<>(model);
        }
/**
 * showDetails
 *
 * @about sets the text for the labels showing subtotal,tax and total
 *
 */
public void showDetails(){
        double subtotal = 0;
        double total;
        final Double TAX = 0.06;
        for(Item i: shoppingCartLinkedList){
                subtotal += i.getPrice() * i.getAmountInCart();
        }
        total = (subtotal * TAX) + subtotal;
        this.subtotal.setText("Subtotal $" + subtotal);
        this.tax.setText("Tax 6%");
        this.total.setText("Total $" + total);
        }

/**
 * hideView
 *
 * @about sets setVisable to false
 *
 */
public void hideView(){
        frame.setVisible(false);
        }

/**
 * updateInventory
 *
 * @about updates changes for the item in the inventory list and adds the items to an order
 *
 */
public void updateInventory(){
        Order order = new Order();
        ArrayList<Item> inventoryList = productList.getList();
        int index = 0;
        for(Item i:shoppingCartLinkedList){
                index = inventoryList.indexOf(i);
                i.setQuantity(i.getQuantity() - inventoryList.get(index).getAmountInCart());
                i.setOrderAmount(i.getAmountInCart());
                i.setAmountInCart(0);
                order.addItem(i);
        }
        OrderList orderList = OrderList.getInstance();
        orderList.add(order);
        }

}
```

CheckoutViewController.java

```
package gui;

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

import client.Inventory;
/**
 *
 * @author Andrew Stallone
 * @about Student class that sorts by date or name
 */
public class CheckoutViewController implements ActionListener{
private CheckoutView checkoutView;
/**
 * Constructor for the CheckoutViewController Object
 *
 * @param CheckoutView
 *
 */
	public CheckoutViewController(CheckoutView checkoutView){
		this.checkoutView = checkoutView;
	}
		@Override
	public void actionPerformed(ActionEvent e) {
				checkoutView.updateInventory();
				OrderConfirmationView orderConfirmationView = new OrderConfirmationView();
				checkoutView.hideView();

	}

}
```

CustomerView.java

```
package gui;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionListener;
import java.util.ArrayList;

import javax.lang.model.element.QualifiedNameable;
import javax.swing.BoxLayout;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.event.ListSelectionListener;

import client.Inventory;
import client.Item;
```

```
import client.ShoppingCart;
/**
 *
 * @author Andrew Stallone
 * @about CustomerView class that displays the customer screen
 */
public class CustomerView {

	private JFrame frame = new JFrame("CustomerScreen");
	private BorderLayout borderLayout = new BorderLayout();
	private JLabel customerName = new JLabel("Hello Andrew");
	private JButton addToCart;
	private JButton checkout;
	private JPanel buttonPanel = new JPanel();
	private GridLayout buttonGrid= new GridLayout(1, 2);
	private ListSelectionListener customerListSelectionListener = new CustomerViewController(this);
	private ShoppingCart shoppingCart = new ShoppingCart();
	private Inventory productList = Inventory.getInstance();
	private ArrayList<Item> inventoryList = productList.getList();
	private JPanel description = new JPanel();
	private GridLayout descriptionGrid = new GridLayout(2,2);
	private JLabel nameLabel = new JLabel();
	private JLabel descriptionLabel = new JLabel();
	private JLabel priceLabel = new JLabel();
	private JLabel quatityLabel = new JLabel();
	private JScrollPane scrollPane = new JScrollPane();
	private ActionListener addToCartListener = new AddToCartController(this);
	private Item i;

	/**
	 * Constructor for the CustomerView Object
	 *
	 * @about initializes the view
	 *
	 */

	public CustomerView(){
		checkout = new JButton("ShowCart");
		addToCart = new JButton("add to cart");
		frame.setLayout(borderLayout);
		customerName.setHorizontalAlignment(JLabel.CENTER);
		frame.add(customerName, BorderLayout.NORTH);
		JList<String> productListjList = populateProductList();
		scrollPane.setViewportView(productListjList);
		frame.add(scrollPane, BorderLayout.EAST);
		buttonPanel.setLayout(buttonGrid);
		buttonPanel.add(checkout);
		addToCart.addActionListener(addToCartListener);
		productListjList.addListSelectionListener(customerListSelectionListener);
		buttonPanel.add(addToCart);
		frame.add(buttonPanel, BorderLayout.SOUTH);
		description.setLayout(descriptionGrid);
		description.setSize(150,150);
		nameLabel.setHorizontalAlignment(JLabel.CENTER);
		descriptionLabel.setHorizontalAlignment(JLabel.CENTER);
		priceLabel.setHorizontalAlignment(JLabel.CENTER);
		quatityLabel.setHorizontalAlignment(JLabel.CENTER);
		description.add(nameLabel);
```

```
		description.add(priceLabel);
		description.add(descriptionLabel);
		description.add(quatityLabel);
		frame.add(description, BorderLayout.CENTER);
		frame.pack();
		frame.setSize(400, 150);
		frame.setLocationRelativeTo(null);
		frame.setVisible(true);

}

/**
 * populateProductList
 *
 * @about constructs the defaultListModel needed for the jlist
 *
 */
public JList<String> populateProductList(){

		DefaultListModel<String> model = new DefaultListModel<>();
		for(Item i: inventoryList){
			model.addElement(i.getName());
		}
		return new JList<>(model);
}
/**
 * showDetails
 *
 * @param index and integer value to correlates to an item in the JList
 * @about sets the labels for the product details
 *
 */
public void showDetails(int index){
		i = inventoryList.get(index);
		nameLabel.setText(i.getName());
		descriptionLabel.setText(i.getDescription());
		priceLabel.setText("$" +Double.toString(i.getPrice()));
		quatityLabel.setText(Integer.toString(i.getQuantity()) + " in Stock ");

}

/**
 * addToCart
 *
 * @about adds the selected item to the shoppingcart list
 *
 */
public void addToCart(){
		if(i.getAmountInCart() < i.getQuantity()){
			shoppingCart.addItem(i);
			System.out.println(Integer.toString(shoppingCart.getSize()));
		}else{

			System.out.println("too many");
		}
}

/**
```

```
	 * getShoppingCart
	 *
	 * @about getter for the shopping cart list
	 * @return returns the ShoppingCart object
	 */
	public ShoppingCart getShoppingCart(){
		return shoppingCart;
	}
	/**
	 * hideCustomerView
	 *
	 * @about hides frame
	 */
	public void hideCustomerView(){
		frame.setVisible(false);
	}

	/**
	 * showCustomerView
	 *
	 * @about shows frame
	 */
	public void showCustomerView(){
		frame.setVisible(true);
	}

}
```

CustomerViewController.java

```
package gui;


import javax.swing.JList;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
/**
 *
 * @author Andrew Stallone
 * @about ListSelectionListener for the CustomerViewController class notifies when the user clicks
 */


public class CustomerViewController implements  ListSelectionListener{
	private CustomerView customerView;
	/**
	 * Constructor for the CustomerViewController Object
	 *
	 * @param CustomerView
	 */
	public CustomerViewController(CustomerView c) {

		customerView = c;
	}


	@Override
```

```
	public void valueChanged(ListSelectionEvent e) {
		JList list = (JList) e.getSource();
		int selections[] = list.getSelectedIndices();
		customerView.showDetails((selections[0]));

	}


}
```

LoginController.java

```
package gui;

import java.awt.GridLayout;
import java.awt.Label;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
/**
 *
 * @author Andrew Stallone
 * @about ActionListener for the Login button
 *
 *  */

public class LoginController implements ActionListener{

private String usernameCustomer = "andrew";
private String passwordCustomer = "1234";
private String usernameSeller = "andrew";
private String passwordSeller = "5678";
private String username;
private String password;
private LoginView login;


/**
 * Constructor for the LoginController Object
 *
 * @param LoginView
 * @about handles the login
 *
 */

public LoginController(LoginView l){


	login = l;
}
	@Override
	public void actionPerformed(ActionEvent e) {
```

```
                username = login.getUsernameText();
                password = login.getPasswordText();
                if(usernameCustomer.equals(username) && passwordCustomer.equals(password)){
                        CustomerView customerView = new CustomerView();
                }else if(usernameSeller.equals(username) && passwordSeller.equals(password)){
                        SellerView sellerView = new SellerView();
                }else{
                        login.setLabel("try Again");
                }

        }



}
```

LoginView.java

```
package gui;

import java.awt.GridLayout;
import java.awt.Label;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
/**
 *
 * @author Andrew Stallone
 * @about LoginView class that displays the login frame
 */
public class LoginView {


        private JFrame frame = new JFrame("Login");
        private JTextField usernameTextField = new JTextField();
        private JTextField passwordTextField = new JTextField();
        private JLabel label = new JLabel();
        private JButton login = new JButton("Login");

        /**
         * Constructor for the LoginView Object
         *
         * @about sets up the view
         *
         */

        public LoginView(){
                frame.setLayout(new GridLayout(3, 2));
                frame.add(new Label("Username:"));
                frame.add(usernameTextField);
                frame.add(new Label("Password:"));
                frame.add(passwordTextField);
                frame.add(login);
```

```
                frame.add(label);
                frame.pack();
                frame.setSize(400, 100);
                frame.setLocationRelativeTo(null);
                frame.setVisible(true);

                ActionListener loginActionListener = new LoginController(this);
                login.addActionListener(loginActionListener);
        }

        /**
         * setLabel
         *
         * @param the text to be displayed
         * @about sets the label with the given text.
         *
         */
        public void setLabel(String label){
                this.label.setText(label);
        }

        /**
         * getUsernameText
         *
         * @return String
         * @about returns the text from the usernameTextField
         *
         */
        public String getUsernameText(){
                return usernameTextField.getText();
        }

        /**
         * getPasswordText
         *
         * @return String
         * @about returns the text from the passwordTextField
         *
         */
        public String getPasswordText(){
                return passwordTextField.getText();
        }


}
```

Main.java

```
package gui;



public class Main {

	public static void main(String[] args) {

		LoginView loginView = new LoginView();

	}

}
```

OrderConfirmationOKButtonController.java

```
package gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
*
* @author Andrew Stallone
* @about ActionListener for the ok button in the OrderConfirmationView
*/
public class OrderConfirmationOKButtonController implements ActionListener{
	private OrderConfirmationView orderView;

	public OrderConfirmationOKButtonController(OrderConfirmationView orderView){
		this.orderView = orderView;
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		orderView.hide();

	}

}
```

OrderConfirmationView.java

```
package gui;

import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

/**
 *
 * @author Andrew Stallone
```

```
 * @about OrderConfirmationView displays the order confirmation view
 */
public class OrderConfirmationView{

	private JFrame frame = new JFrame();
	private JPanel orderCompletePanel = new JPanel();
	private BorderLayout borderLayout = new BorderLayout();
	private JLabel orderCompleteLabel = new JLabel("Thank you for your order, it has been submitted!");
	private JButton okButton = new JButton("Continue");

	/**
	 * Constructor for the OrderConfirmationView Object
	 *
	 * @about sets up the view
	 *
	 */
	public OrderConfirmationView(){
	okButton.addActionListener(new OrderConfirmationOKButtonController(this));
	orderCompletePanel.setLayout(borderLayout);
	frame.add(orderCompletePanel);
	orderCompletePanel.add(orderCompleteLabel, BorderLayout.NORTH);
	orderCompletePanel.add(okButton, BorderLayout.SOUTH);
	frame.pack();
	frame.setSize(375, 150);
	frame.setVisible(true);
	}

	/**
	 * hide
	 *
	 * @about hides the view
	 *
	 */
	public void hide(){
		frame.setVisible(false);
	}


}
```

RevenueViewController.java

```
package gui;

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

/**
*
* @author Andrew Stallone
* @about Done button controller on the revenue screen
*/
public class RevenueViewController implements ActionListener{
	private SellerRevenueView sellerRevenueView;

	public RevenueViewController(SellerRevenueView sellerRevenueView) {
		this.sellerRevenueView = sellerRevenueView;
	}
```

```
	@Override
	public void actionPerformed(ActionEvent e) {
		sellerRevenueView.close();

	}

}
```

SellerJListController.java

```
package gui;

import javax.swing.JList;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
/**
*
* @author Andrew Stallone
* @about list selection controller on the sellerview
*/
public class SellerJListController implements ListSelectionListener{
	private SellerView sellerView;
	public SellerJListController(SellerView sellerView){
		this.sellerView = sellerView;
	}

	@Override
	public void valueChanged(ListSelectionEvent e) {
		JList list = (JList) e.getSource();
		int selections[] = list.getSelectedIndices();
		sellerView.updateText(selections[0]);

	}

}
```

SellerRevenueButtonController.java

```
package gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
 *
 * @author Andrew Stallone
 * @about displays the revenue view
 */
public class SellerRevenueButtonController implements ActionListener{

	@Override
	public void actionPerformed(ActionEvent e) {
		SellerRevenueView sellerRevenueView = new SellerRevenueView();
	}

}
```

SellerRevenueView.java

```
package gui;

import java.awt.BorderLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.event.ActionListener;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import client.Order;
import client.OrderList;
/**
 *
 * @author Andrew Stallone
 * @about SellerRevenueView class that displays the revenue
 */

public class SellerRevenueView {
	private ArrayList<Order> listOfOrders = new ArrayList<Order>();
	private JFrame frame = new JFrame();
	private JPanel mainPanel = new JPanel();
	private JPanel buttonPanel = new JPanel();
	private JPanel revenuePanel = new JPanel();
	private JLabel totalRevenue = new JLabel();
	private JLabel totalCost = new JLabel();
	private JLabel totalProfit = new JLabel();
	private JButton done = new JButton("done");
	private BorderLayout borderLayout = new BorderLayout();
	private GridBagLayout gridbag = new GridBagLayout();
	private GridBagConstraints c = new GridBagConstraints();
	private ActionListener doneListener = new RevenueViewController(this);

	/**
	 * Constructor for the SellerRevenueView Object
	 *
	 * @about sets up the view
	 *
	 */
	public SellerRevenueView(){
		OrderList orderList = OrderList.getInstance();
		listOfOrders = orderList.getOrderList();
		NumberFormat formatter = new DecimalFormat("00.00");
		String s = formatter.format(orderList.getTotalRevenue());
		totalRevenue.setText("Total Revenue $" + s);
		s = formatter.format(orderList.getTotalCost());
		totalCost.setText("Total Cost $" + s);
		s = formatter.format(orderList.getTotalRevenue() - orderList.getTotalCost());
		totalProfit.setText("Total Profit $" + s);
		frame.add(mainPanel);
		mainPanel.setLayout(borderLayout);
```

```
                revenuePanel.setLayout(gridbag);

                c.ipady = 10;
                c.weightx = 0.5;
                c.gridwidth = 1;
                c.gridx = 0;
                c.gridy = 0;
                gridbag.setConstraints(totalRevenue, c);
                revenuePanel.add(totalRevenue);

                c.gridy = 1;
                gridbag.setConstraints(totalCost, c);
                revenuePanel.add(totalCost);

                c.gridy = 2;
                gridbag.setConstraints(totalProfit, c);
                revenuePanel.add(totalProfit);

                mainPanel.add(revenuePanel, BorderLayout.CENTER);
                buttonPanel.add(done);
                done.addActionListener(doneListener);
                mainPanel.add(buttonPanel, BorderLayout.SOUTH);
                frame.setSize(500, 500);
                frame.pack();
                frame.setVisible(true);
        }

        /**
         * close
         *
         * @about disposes the view
         *
         */
        public void close(){
                frame.dispose();
        }
}
```

SellerView.java

```
package gui;

import java.awt.BorderLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.event.ActionListener;
import java.util.ArrayList;

import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
```

```
import javax.swing.event.ListSelectionListener;

import client.Inventory;
import client.Item;
/**
 *
 * @author Andrew Stallone
 * @about SellerView class to display the seller view
 */
public class SellerView {

	private JFrame frame = new JFrame();
	private JPanel mainPanel = new JPanel();
	private JPanel updatePanel = new JPanel();
	private JPanel editPanel = new JPanel();
	private JPanel productPanel = new JPanel();
	private JPanel itemPanel = new JPanel();
	private JPanel buttonPanel = new JPanel();
	private JScrollPane scrollPane = new JScrollPane();
	private JLabel sellersNameLabel = new JLabel("Hello Andrew");
	private JButton showReport = new JButton("Revenue Report");
	private JButton editProducts = new JButton("Edit");
	private JButton addProduct = new JButton("Add New Item");
	private JTextField nameTextField = new JTextField();
	private JTextField quantityTextField = new JTextField();
	private JTextField priceTextField = new JTextField();
	private JTextArea descriptionTextArea = new JTextArea();
	private JTextArea costTextField = new JTextArea();
	private JList<String> productList;
	private BorderLayout borderLayout = new BorderLayout();
	private GridLayout gridLayout = new GridLayout(1,3);
	private GridBagLayout gridbag = new GridBagLayout();
	private GridBagConstraints c = new GridBagConstraints();
	private ActionListener addItemsListener = new SellerViewAddProductsController(this);
	private ActionListener update = new SellerViewUpdateController(this);
	private ActionListener revenue = new SellerRevenueButtonController();
	private int index = 0;
	private ArrayList<Item> itemList = new ArrayList<Item>();
	private Inventory inventory = Inventory.getInstance();
	private ListSelectionListener listSelection;
	private JLabel name = new JLabel();
	private JLabel price = new JLabel();
	private JLabel quantity = new JLabel();
	private JLabel description = new JLabel();
	private JLabel cost = new JLabel();



	/**
	 * Constructor for the SellerView Object
	 *
	 * @about sets up the view
	 *
	 */
	public SellerView(){
		listSelection = new SellerJListController(this);
		frame.add(mainPanel);
```

```
		mainPanel.setLayout(borderLayout);
		c.fill = GridBagConstraints.HORIZONTAL;
		mainPanel.add(buttonPanel, BorderLayout.SOUTH);
		buttonPanel.setLayout(gridLayout);
		buttonPanel.add(addProduct);
		buttonPanel.add(editProducts);
		buttonPanel.add(showReport);
		editProducts.addActionListener(update);
		addProduct.addActionListener(addItemsListener);
		showReport.addActionListener(revenue);
		itemList = inventory.getList();
		productList = populateProductList();
		productList.addListSelectionListener(listSelection);
		scrollPane.setViewportView(productList);
		itemPanel.add(scrollPane);
		mainPanel.add(itemPanel, BorderLayout.EAST);
		mainPanel.add(sellersNameLabel, BorderLayout.NORTH);
		if(itemList.size() != 0){
		initlabelView(0);
		}else{
			initEmptyLabelView();
		}
		frame.setSize(300, 300);
		frame.pack();
		frame.setVisible(true);

}

public void updateInventory(){
		itemList = inventory.getList();
		productList = populateProductList();
		productList.addListSelectionListener(listSelection);
		scrollPane.setViewportView(productList);
		itemPanel.add(scrollPane);
		mainPanel.add(itemPanel, BorderLayout.EAST);
		frame.validate();
		frame.repaint();
}



public JList<String> populateProductList(){

		DefaultListModel<String> model = new DefaultListModel<>();
		model.clear();

		for(Item i: itemList){
			model.addElement(i.getName());
		}
		return new JList<>(model);
}

/**
 * initEmptyLabelView
 *
 * @about sets up the gridbag constraints for the view
 *
 */
```

```
public void initEmptyLabelView(){
		productPanel.setLayout(gridbag);

		c.ipady = 10;
		c.weightx = 0.5;
		c.gridwidth = 1;
		c.gridx = 0;
		c.gridy = 0;
		gridbag.setConstraints(name, c);
		productPanel.add(name);

		c.ipady = 10;
		c.gridx = 1;
		c.gridy = 0;
		c.gridwidth = 1;
		gridbag.setConstraints(price, c);
		productPanel.add(price);


		c.ipady = 10;
		c.gridx = 0;
		c.gridy = 1;
		c.gridwidth = 1;
		gridbag.setConstraints(quantity, c);
		productPanel.add(quantity);


		c.ipady = 40;
		c.weightx = 0.0;
		c.gridwidth = 1;
		c.gridx = 1;
		c.gridy = 1;
		gridbag.setConstraints(description, c);
		productPanel.add(description);

		c.ipady = 10;
		c.gridx = 2;
		c.gridy = 0;
		c.gridwidth = 1;
		gridbag.setConstraints(cost, c);
		productPanel.add(cost);


		mainPanel.add(productPanel, BorderLayout.CENTER);
		frame.validate();
		frame.repaint();

}

/**
 * initEmptyLabelView
 *
 * @param int index of item to be displayed
 * @about sets up the gridbag constraints for the view and sets labels
 *
 */
public void initlabelView(int index){
```

```
            Item i = itemList.get(index);
            name.setText(i.getName());
            price.setText(Double.toString(i.getPrice()));
            cost.setText("product cost $" + Double.toString(i.getCost()));
            quantity.setText(Integer.toString(i.getQuantity())+ " in stock");
            description.setText(i.getDescription());
            price.setHorizontalAlignment(SwingConstants.RIGHT);
            description.setHorizontalAlignment(SwingConstants.RIGHT);

            productPanel.setLayout(gridbag);

            c.ipady = 10;
            c.weightx = 0.5;
            c.gridwidth = 1;
            c.gridx = 0;
            c.gridy = 0;
            gridbag.setConstraints(name, c);
            productPanel.add(name);

            c.ipady = 10;
            c.gridx = 1;
            c.gridy = 0;
            c.gridwidth = 1;
            gridbag.setConstraints(price, c);
            productPanel.add(price);


            c.ipady = 10;
            c.gridx = 0;
            c.gridy = 1;
            c.gridwidth = 1;
            gridbag.setConstraints(quantity, c);
            productPanel.add(quantity);


            c.ipady = 40;
            c.weightx = 0.0;
            c.gridwidth = 1;
            c.gridx = 1;
            c.gridy = 1;
            gridbag.setConstraints(description, c);
            productPanel.add(description);

            c.ipady = 10;
            c.gridx = 0;
            c.gridy = 2;
            c.gridwidth = 1;
            gridbag.setConstraints(cost, c);
            productPanel.add(cost);


            mainPanel.add(productPanel, BorderLayout.CENTER);
            frame.validate();
            frame.repaint();

}

/**
```

```
	 * updateText
	 *
	 * @return int the index of the list
	 * @about updates the text in the view
	 *
	 */
	public void updateText(int index){
		this.index = index;
		Item i = itemList.get(index);
		name.setText(i.getName());
		price.setText(Double.toString(i.getPrice()));
		quantity.setText(Integer.toString(i.getQuantity())+ " in stock");
		description.setText(i.getDescription());
		cost.setText("Product cost $" + Double.toString(i.getCost()));
	}

	/**
	 * getIndex
	 *
	 * @return int the index of the list
	 * @about returns the index of the list
	 *
	 */
	public int getIndex(){
		return index;
	}



}
```

SellerViewAddProductsController.java

```
package gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
 *
 * @author Andrew Stallone
 * @about displays the additem view
 */
public class SellerViewAddProductsController implements ActionListener{
	private SellerView sellerView;
	public SellerViewAddProductsController(SellerView sellerView){
		this.sellerView = sellerView;
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		AddItemView itemView = new AddItemView(sellerView);

	}

}
```

SellerViewUpdateController.java

```
package gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
 *
 * @author Andrew Stallone
 * @about displays the update view
 */
public class SellerViewUpdateController implements ActionListener{
	private SellerView sellerView;
	public SellerViewUpdateController(SellerView sellerView){
		this.sellerView = sellerView;
	}
	@Override
	public void actionPerformed(ActionEvent e) {
		UpdateView updateView = new UpdateView(sellerView);

	}
}
```

ShoppingCartListController.java

```
package gui;

import javax.swing.JList;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
/**
 *
 * @author Andrew Stallone
 * @about listener for the jlist
 */
public class ShoppingCartListController implements ListSelectionListener{
	private ShoppingCartView shoppingCartView;
	public ShoppingCartListController(ShoppingCartView shoppingCartView){
		this.shoppingCartView = shoppingCartView;
	}

	@Override
	public void valueChanged(ListSelectionEvent e) {
		JList list = (JList) e.getSource();
		int selections[] = list.getSelectedIndices();
		shoppingCartView.displayDetails(selections[0]);
	}

}
```

ShoppingCartView.java

```
package gui;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
```

```
import java.awt.GridLayout;
import java.awt.event.ActionListener;
import java.util.LinkedList;
import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextField;
import javax.swing.event.ListSelectionListener;

import client.Item;
import client.ShoppingCart;
/**
 *
 * @author Andrew Stallone
 * @about displays the shoppingCart view
 */
public class ShoppingCartView {

	private JFrame frame = new JFrame("shoppingCartView");
	private ShoppingCart shoppingCart;
	private BorderLayout borderLayout = new BorderLayout();
	private JScrollPane scrollPane = new JScrollPane();
	private JPanel editInfo = new JPanel();
	private GridLayout grid = new GridLayout(6,2);
	private LinkedList<Item> shoppingCartLinkedList;
	private FlowLayout flowLayout =  new FlowLayout();
	private JPanel flowPanel = new JPanel();
	private JList<String> shoppingCartList;
	private JLabel productName = new JLabel();
	private JLabel productPrice = new JLabel();
	private JLabel numberOfProducts = new JLabel("quantity ");
	private JButton deleteFromShoppingCart;
	private JButton update;
	private JTextField updateQuantity = new JTextField();
	private JButton checkOutButton;
	private CustomerView customerView;
	private ListSelectionListener listSelector = new ShoppingCartListController(this);
	private ActionListener deleteButton = new ShoppingCartViewController(this);
	private ActionListener updateButtonListener = new UpdateButtonController(this);
	private JLabel subtotal;
	private int SelectedIndex = 0;


	/**
	 * Constructor for the ShoppingCartView Object
	 *
	 * @param ShoppingCart
	 * @param CustomerView
	 * @about sets up the view
	 *
	 */

	public ShoppingCartView(ShoppingCart shoppingCart, CustomerView customerView){
		this.customerView = customerView;
```

```
                ActionListener checkoutButtonListener = new checkoutButtonController(this, customerView);
                deleteFromShoppingCart = new JButton("Delete");
                update = new JButton("Update");
                checkOutButton = new JButton("Checkout");
                frame.setLayout(borderLayout);
                flowPanel.setLayout(flowLayout);
                flowPanel.add(numberOfProducts);
                flowPanel.add(updateQuantity);
                editInfo.setLayout(grid);
                editInfo.add(productName);
                editInfo.add(productPrice);
                editInfo.add(flowPanel);
                deleteFromShoppingCart.addActionListener(deleteButton);
                update.addActionListener(updateButtonListener);
                editInfo.add(update);
                editInfo.add(deleteFromShoppingCart);
                editInfo.add(checkOutButton);
                checkOutButton.addActionListener(checkoutButtonListener);
                productName.setHorizontalAlignment(JLabel.CENTER);
                productPrice.setHorizontalAlignment(JLabel.CENTER);
                this.shoppingCart = shoppingCart;
                shoppingCartLinkedList = shoppingCart.getList();
                shoppingCartList = populateProductList();
                shoppingCartList.addListSelectionListener(listSelector);
                scrollPane.setViewportView(shoppingCartList);
                frame.add(scrollPane, BorderLayout.EAST);
                frame.add(editInfo, BorderLayout.CENTER);
                productName.setText(shoppingCartLinkedList.get(0).getName());
                productPrice.setText(Double.toString(shoppingCartLinkedList.get(0).getPrice()));
                updateQuantity.setText(Integer.toString(shoppingCartLinkedList.get(0).getAmountInCart()));
                subtotal = new JLabel("Subtotal $" +Double.toString(getSubTotal()));
                frame.add(subtotal, BorderLayout.SOUTH);
                frame.pack();
                frame.setSize(400, 300);
                frame.setVisible(true);
        }


public JList<String> populateProductList(){

                DefaultListModel<String> model = new DefaultListModel<>();
                model.clear();

                for(Item i: shoppingCartLinkedList){
                        model.addElement(i.getName());
                }
                return new JList<>(model);
        }

/**
 * repaint
 *
 * @param ShoppingCart
 * @about refreshes the view
 *
 */

        public void repaint(ShoppingCart shoppingCart){
```

```
		frame.setLayout(borderLayout);
		shoppingCartLinkedList = shoppingCart.getList();
		shoppingCartList = populateProductList();
		shoppingCartList.addListSelectionListener(listSelector);
		scrollPane.setViewportView(shoppingCartList);
		frame.add(scrollPane, BorderLayout.EAST);
		editInfo.setLayout(grid);
		flowPanel.setLayout(flowLayout);
		flowPanel.add(numberOfProducts);
		flowPanel.add(updateQuantity);
		editInfo.add(productName);
		editInfo.add(productPrice);
		editInfo.add(flowPanel);
		editInfo.add(update);
		editInfo.add(deleteFromShoppingCart);
		editInfo.add(checkOutButton);
		frame.add(editInfo, BorderLayout.CENTER);
		frame.setLocation(frame.getLocation().x, frame.getLocation().y);
		frame.setSize(frame.getWidth(),frame.getHeight());
		subtotal.setText("Subtotal $" +Double.toString(shoppingCart.getSubtotal()));
		frame.add(subtotal, BorderLayout.SOUTH);
		if(shoppingCartLinkedList.size() == 0){
			frame.setVisible(false);
			frame.dispose();
		}else{
		frame.validate();
		frame.repaint();
		}

}

/**
 * displayDetails
 *
 * @param int the index of the list
 * @about displays the details
 *
 */
public void displayDetails(int index){
		SelectedIndex = index;
		Item i = shoppingCartLinkedList.get(index);
		productName.setHorizontalAlignment(JLabel.CENTER);
		productPrice.setHorizontalAlignment(JLabel.CENTER);
		productName.setText(i.getName());
		updateQuantity.setText(Integer.toString(i.getAmountInCart()));
		productPrice.setText("$" +Double.toString(i.getPrice() * i.getAmountInCart()));
		subtotal.setText("Subtotal $" +Double.toString(getSubTotal()));
		frame.add(subtotal, BorderLayout.SOUTH);
}

/**
 * deleteItem
 *
 * @about deletes item from the list
 *
 */
public void deleteItem(){
		if(shoppingCartLinkedList.size() > 0){
```

```
                        Item i = shoppingCartLinkedList.get(SelectedIndex);
                        i.setAmountInCart(0);
                        shoppingCartLinkedList.remove(SelectedIndex);
                        repaint(shoppingCart);
                        productName.setText("        ");
                        productPrice.setText("        ");

                }

}

/**
 * getSubTotal
 *
 * @about computes the subtotal
 * @return the subtotal
 */
public double getSubTotal(){
                double result = 0;
                for(Item i: shoppingCartLinkedList){
                        result += i.getPrice() * i.getAmountInCart();
                }
                return result;
}

/**
 * update
 *
 * @about updates item from the list
 *
 */
public void update(){
                Item i = shoppingCartLinkedList.get(SelectedIndex);
                if(i.getQuantity() >= Integer.parseInt(updateQuantity.getText())){
                i.setAmountInCart(Integer.parseInt(updateQuantity.getText()));
                productPrice.setText("$" +Double.toString(i.getPrice() * i.getAmountInCart()));
                subtotal.setText("Subtotal $" +Double.toString(getSubTotal()));
                }else{
                        updateQuantity.setText(Integer.toString(i.getQuantity()));
                }


}

/**
 * getShoppingCart
 *
 * @about getter for the shopping cart list
 * @return the shopping cart list
 */
public LinkedList<Item> getShoppingCart(){
                return shoppingCartLinkedList;
}

/**
 * hideCartView
 *
 * @about hides the shopping cart
```

```
	 *
	 */
	public void hideCartView(){
		frame.setVisible(false);
	}

	/**
	 * setVisable
	 *
	 * @about shows the shopping cart
	 *
	 */
	public void setVisable(){
		frame.setVisible(true);
	}

	/**
	 * hideCustomerView
	 *
	 * @about hides the customer view
	 *
	 */
	public void hideCustomerView(){
		customerView.hideCustomerView();
	}

	/**
	 * showCustomerView
	 *
	 * @about shows the customer view
	 *
	 */
	public void showCustomerView(){
		customerView.showCustomerView();
	}
}
```

ShoppingCartViewController.java

```
package gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
 *
 * @author Andrew Stallone
 * @about deletes item from the shopping cart
 */
public class ShoppingCartViewController implements ActionListener{
private ShoppingCartView shoppingCartView;
	public ShoppingCartViewController(ShoppingCartView shoppingCartView){
		this.shoppingCartView = shoppingCartView;
	}
	@Override
	public void actionPerformed(ActionEvent e) {
		shoppingCartView.deleteItem();
```

```
	}

}
```

UpdateButtonController.java

```
package gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
 *
 * @author Andrew Stallone
 * @about updates items in the shopping cart
 */
public class UpdateButtonController implements ActionListener{

private ShoppingCartView shoppingCartView;

	public UpdateButtonController(ShoppingCartView shoppingCartView){
		this.shoppingCartView = shoppingCartView;
	}
	@Override
	public void actionPerformed(ActionEvent e) {
		shoppingCartView.update();

	}

}
```

UpdateView.java

```
package gui;

import java.awt.BorderLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.event.ActionListener;
import java.util.ArrayList;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;

import client.Inventory;
import client.Item;

/**
 *
 * @author Andrew Stallone
```

```
 * @about UpdateView class that shows the update view
 */

public class UpdateView {

	private JFrame frame = new JFrame();
	private JPanel mainPanel = new JPanel();
	private JPanel updatePanel = new JPanel();
	private JPanel buttonPanel = new JPanel();
	private JButton addProduct = new JButton("Update");
	private JButton deleteProduct = new JButton("Delete");
	private JTextField nameTextField = new JTextField();
	private JTextField quantityTextField = new JTextField();
	private JTextField priceTextField = new JTextField();
	private JTextArea descriptionTextArea = new JTextArea();
	private JTextField costTextField = new JTextField();
	private BorderLayout borderLayout = new BorderLayout();
	private GridLayout gridLayout = new GridLayout(1,2);
	private GridBagLayout gridbag = new GridBagLayout();
	private GridBagConstraints c = new GridBagConstraints();
	private Inventory inventory = Inventory.getInstance();
	private ArrayList<Item> itemList = inventory.getList();
	private JLabel name = new JLabel();
	private JLabel price = new JLabel();
	private JLabel quantity = new JLabel();
	private JLabel description = new JLabel();
	private ActionListener add = new UpdateViewUpdateButtonController(this);
	private ActionListener delete = new UpdateViewDeleteProductController(this);
	private SellerView sellerView;

	/**
	 * Constructor for the UpdateView Object
	 *
	 * @param SellerView
	 * @about sets up the view
	 *
	 */
	public UpdateView(SellerView sellerView){
		this.sellerView = sellerView;
		frame.add(mainPanel);
		mainPanel.setLayout(borderLayout);
		c.fill = GridBagConstraints.HORIZONTAL;
		mainPanel.add(buttonPanel, BorderLayout.SOUTH);
		buttonPanel.setLayout(gridLayout);
		buttonPanel.add(deleteProduct);
		buttonPanel.add(addProduct);
		deleteProduct.addActionListener(delete);
		addProduct.addActionListener(add);
		initUpdateView();
		setTextFields();
		frame.setSize(600, 200);
		frame.pack();
		frame.setVisible(true);
	}

	/**
	 * Constructor for the UpdateView Object
	 *
```

```
 * @param SellerView
 * @about sets up the view
 *
 */
public void initUpdateView(){

		updatePanel.setLayout(gridbag);

		JLabel name = new JLabel("Name: ");
		name.setHorizontalAlignment(SwingConstants.RIGHT);
		c.ipady = 10;
		c.weightx = 0.5;
		c.gridwidth = 1;
		c.gridx = 0;
		c.gridy = 0;
		gridbag.setConstraints(name, c);
		updatePanel.add(name);

		c.ipady = 10;
		c.weightx = 0.5;
		c.gridx = 1;
		c.gridy = 0;
		gridbag.setConstraints(nameTextField, c);
		updatePanel.add(nameTextField);

		JLabel price = new JLabel("Price: ");
		price.setHorizontalAlignment(SwingConstants.RIGHT);
		c.ipady = 10;
		c.gridx = 2;
		c.gridy = 0;
		gridbag.setConstraints(price, c);
		updatePanel.add(price);

		c.ipady = 10;
		c.gridx = 3;
		c.gridy = 0;
		gridbag.setConstraints(priceTextField, c);
		updatePanel.add(priceTextField);


		JLabel quantity = new JLabel("Quantity: ");
		quantity.setHorizontalAlignment(SwingConstants.RIGHT);
		c.ipady = 10;
		c.gridx = 4;
		c.gridy = 0;
		gridbag.setConstraints(quantity, c);
		updatePanel.add(quantity);

		c.ipady = 10;
		c.gridx = 5;
		c.gridy = 0;
		gridbag.setConstraints(quantityTextField, c);
		updatePanel.add(quantityTextField);


		JLabel description = new JLabel("Description: ");
		description.setHorizontalAlignment(SwingConstants.RIGHT);
		description.setVerticalAlignment(SwingConstants.TOP);
```

```
		c.ipady = 40;
		c.weightx = 0.0;
		c.gridwidth = 1;
		c.gridx = 0;
		c.gridy = 1;
		gridbag.setConstraints(description, c);
		updatePanel.add(description);

		c.ipady = 40;
		c.weightx = 0.0;
		c.gridwidth = 5;
		c.gridx = 1;
		c.gridy = 1;
		gridbag.setConstraints(descriptionTextArea, c);
		updatePanel.add(descriptionTextArea);

		JLabel costLabel = new JLabel("cost");
		costLabel.setHorizontalAlignment(SwingConstants.RIGHT);
		c.ipady = 10;
		c.gridx = 0;
		c.gridy = 5;
		c.gridwidth = 1;
		gridbag.setConstraints(costLabel, c);
		updatePanel.add(costLabel);


		c.ipady = 10;
		c.gridx = 1;
		c.gridy = 5;
		gridbag.setConstraints(costTextField, c);
		updatePanel.add(costTextField);
		costTextField.setText("Cost");

		mainPanel.add(updatePanel, BorderLayout.CENTER);


}
	/**
	 * setVisable
	 *
	 * @param if true shows the frame if false hides the frame
	 * @about show/hide the frame
	 *
	 */
	public void setVisable(boolean visable){
		frame.setVisible(visable);
	}

	/**
	 * setTextFields
	 *
	 * @about sets the textfields with the proper information
	 *
	 */
	public void setTextFields(){
		int index = sellerView.getIndex();
		Item i = itemList.get(index);
		nameTextField.setText(i.getName());
```

```
                descriptionTextArea.setText(i.getDescription());
                quantityTextField.setText(Integer.toString(i.getQuantity()));
                priceTextField.setText(Double.toString(i.getPrice()));
                costTextField.setText(Double.toString(i.getCost()));
        }

        /**
         * updateInventory
         *
         * @about updates the inventory
         *
         */
        public void updateInventory(){
                int index = sellerView.getIndex();
                Item i = itemList.get(index);
                i.setName(nameTextField.getText());
                i.setDecscription(descriptionTextArea.getText());
                i.setQuantity(Integer.parseInt(quantityTextField.getText()));
                i.setPrice(Double.parseDouble(priceTextField.getText()));
                i.setCost(Double.parseDouble(costTextField.getText()));

                sellerView.updateText(index);


        }

        /**
         * deleteFromInventory
         *
         * @about deletes from the inventory
         *
         */
        public void deleteFromInventory(){
                int index = sellerView.getIndex();
                Item i = itemList.remove(index);
                sellerView.updateInventory();

        }


}
```

UpdateViewDeleteProductController.java

```
package gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
 *
 * @author Andrew Stallone
 * @about deletes a uitem from the list
 */
public class UpdateViewDeleteProductController implements ActionListener{
        private UpdateView updateView;
        public UpdateViewDeleteProductController(UpdateView updateView){
                this.updateView = updateView;
        }
```

```
	@Override
	public void actionPerformed(ActionEvent e) {
		updateView.deleteFromInventory();
		updateView.setVisable(false);

	}
}
```

UpdateViewButtonController.java

```
package gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
/**
 *
 * @author Andrew Stallone
 * @about updates inventory
 */
public class UpdateViewUpdateButtonController implements ActionListener{

	private UpdateView updateView;
	public UpdateViewUpdateButtonController(UpdateView updateView){
		this.updateView = updateView;
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		updateView.updateInventory();
		updateView.setVisable(false);

	}
}
```