Shopping cart

Group # 20
Joshua Sohan
Mohammed Ahmed

Glossary

1. Requirement specification

1.1 Functional Requirements (Josh)

When the application if first started, the user sees a simple boxed application where they are prompted to log in or create a customer/ seller account. From there the user will be signed in as either a customer or seller.

After signing in, if you are a customer, the System will take you to the product list where you can add can add items to your cart. You can choose how much of that item you want and then click the “add to cart button” next to the item. You can go to your shopping cart by clicking the “Shopping cart” button. From there you can change how much of the item you want and update it with the “Update” Button. Once you are satisfied you can go to the checkout page that displays your total and prompts you for your information. Once you confirm you shopping cart is cleared and all items are “bought”.

If you are a seller then the system takes you to your product inventory where you can change each item by clicking the “edit” button next to it and save it once you are done with the “save” button which replaces the edit button once pressed. You can add a discounted product by selecting the check mark next to an item and entering a new price in the text box near the “Add discounted product” button. You can also select multiple items to either delete or put into a bundle. The Bundle button replaces the “Add discounted product” once more than one item is selected. You can choose to enter a custom price, or it will automatically add up all product prices. The discounted and bundled items cannot but edited but only removed.

All data is saved once the user closes the application.

1.2 Essential Use Cases (Josh/Mohammed)

1. User Logs In:

- Login screen is displayed.
    1) User enters username and password.
    2) System logs the User in as seller.
    3) System takes User to Inventory Editor
- Login screen is displayed.
    1) User enters username and password.
    2) System logs the User in as Customer.
    3) System takes User to main product page.

1. User creates Account:
   - Login screen is displayed.
     4) User desired enters username and password in the create user field.
     5) User selects Customer
     6) System creates a Customer account for user
     7) System takes User to the product list
   - Login screen is displayed.
     4) Users desired enters username and password in the create user field.
     5) User selects Seller.
     6) System creates Seller account for User
     7) System takes User to their new Seller main page.

3. User Adds Items to Shopping Cart:
   - User selects item.
     1) User selects quantity of item from drop down box.
     2) User selects the “Add items to cart” button.
     3) System adds items to shopping cart.

4. User Reviews Product Details:
   - User selects item to review
     1) User goes to main page
     2) System displays all products and product details.

5. User Reviews/Updates Shopping Cart:
   - User views shopping cart
     1. User navigates to the shopping cart
     2. System displays all items in the shopping cart with number of items selected
   - User updates shopping cart
     1. User changes the number of desired items from a comobox
     2. User clicks “update” button
     3. System updates shopping cart with added changes

6. User Checks Out:
   - Checkout
     1. User navigates to checks out after viewing items in shopping cart.
     2. The System brings prompts the User to enter information and displays a total.
     3. The system validates payment
     4. The System clears shopping cart.

7. User Reviews/Updates Inventory:
    - User Reviews inventory:
        1. User Logs in as Seller
        2. System displays all items along with options to manipulate items add new items to inventory.

8. User Adds New Product:
    - Adds New Product
        1. User fills out new product form
        2. User clicks “add”
        3. System updates inventory and refreshes the display.

1.2 Detailed Use Cases (Josh)

1. User Logs In:
    - Variation #2:
        1. User enters incorrect username and/or password.
        2. System denies User.
        3. System shows message that username/password don’t match.

4. User Reviews/Updates Shopping Cart:
    - Variation #2:
        1. User navigates to the shopping cart
        2. The User has selected no items and therefore they system displays no items

5. User Checks Out:
    - Variation #2:
        1. User navigates to checks out after viewing items in shopping cart.
        2. The System alerts the user with a message saying “No items in shopping cart”.
    - Variation #3:
        1. User navigates to checks out after viewing items in shopping cart.
        2. The System brings up all items along with the total and the user is prompted to pay.
        3. System declines payment and the User is prompted to pay another way.

6. User Reviews/Updates Inventory:
    - Variation #2:
        1. The User has no items and therefore the system displays none.

7. User Adds New Product:
    - Adds New Discount Product
        1. User selects item to add as discount
        2. User enters new price for item and clicks add
        3. System adds discounted item and updates display
    - Bundle Product
        1. User selects multiple products
        2. System changes “add discount product” to “add Bundle Product”
        3. User enters desired price and presses the add button
        4. System adds product and updates display

2. Design specification

2.1 CRC Cards (Josh)

Customer	
Creates shopping cart objects **Manages Shopping cart of that specific buyer**	**ShoppingCart** **Name** **Username** **Password**

Seller	
Creates inventory **Manage inventory of that specific seller**	**Inventory** **ShoppingCart** **Name** **Username** **Password**

Product	
Stores product info	**Double price** **String description** **int inventory**

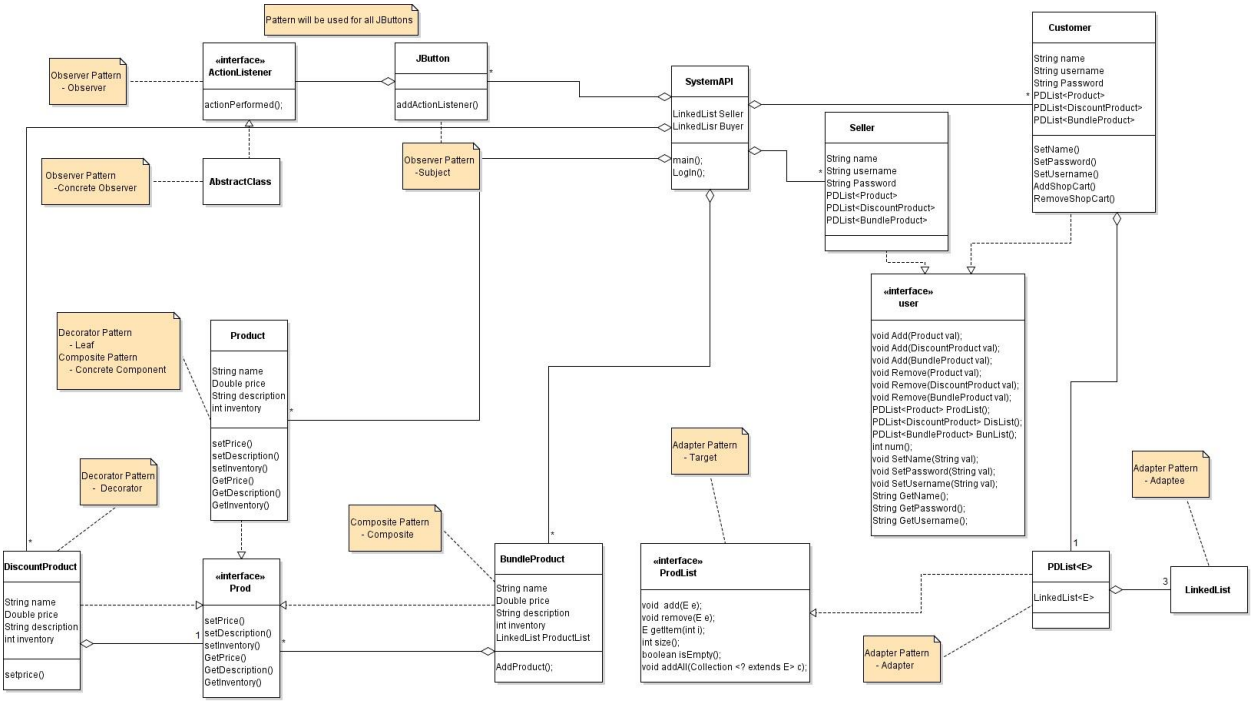
bundleProduct	
A collection of Products **Manages items in list**	**LinkedList**

DiscountProduct	
Creates a new product with the price changed	**Product**

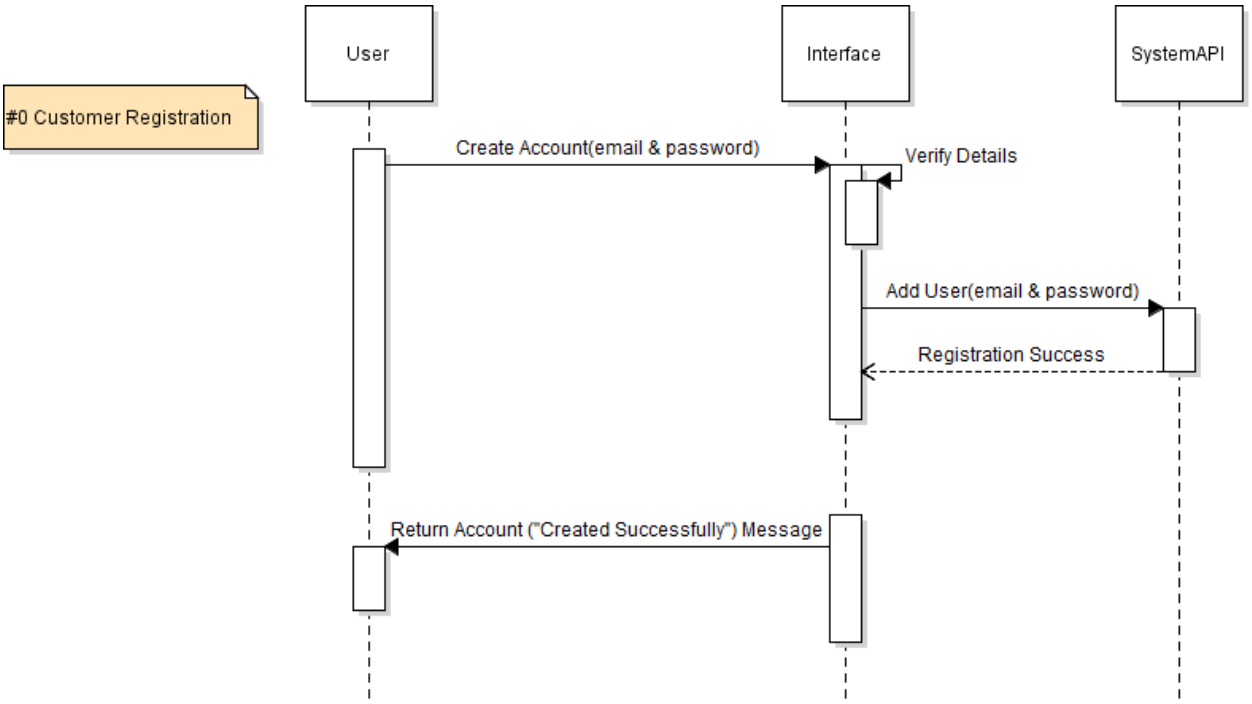
ProdList<E>	
Manages a list of objects (different product) **Adapts Linked list to add collections of object E**	**linkedList<E>**

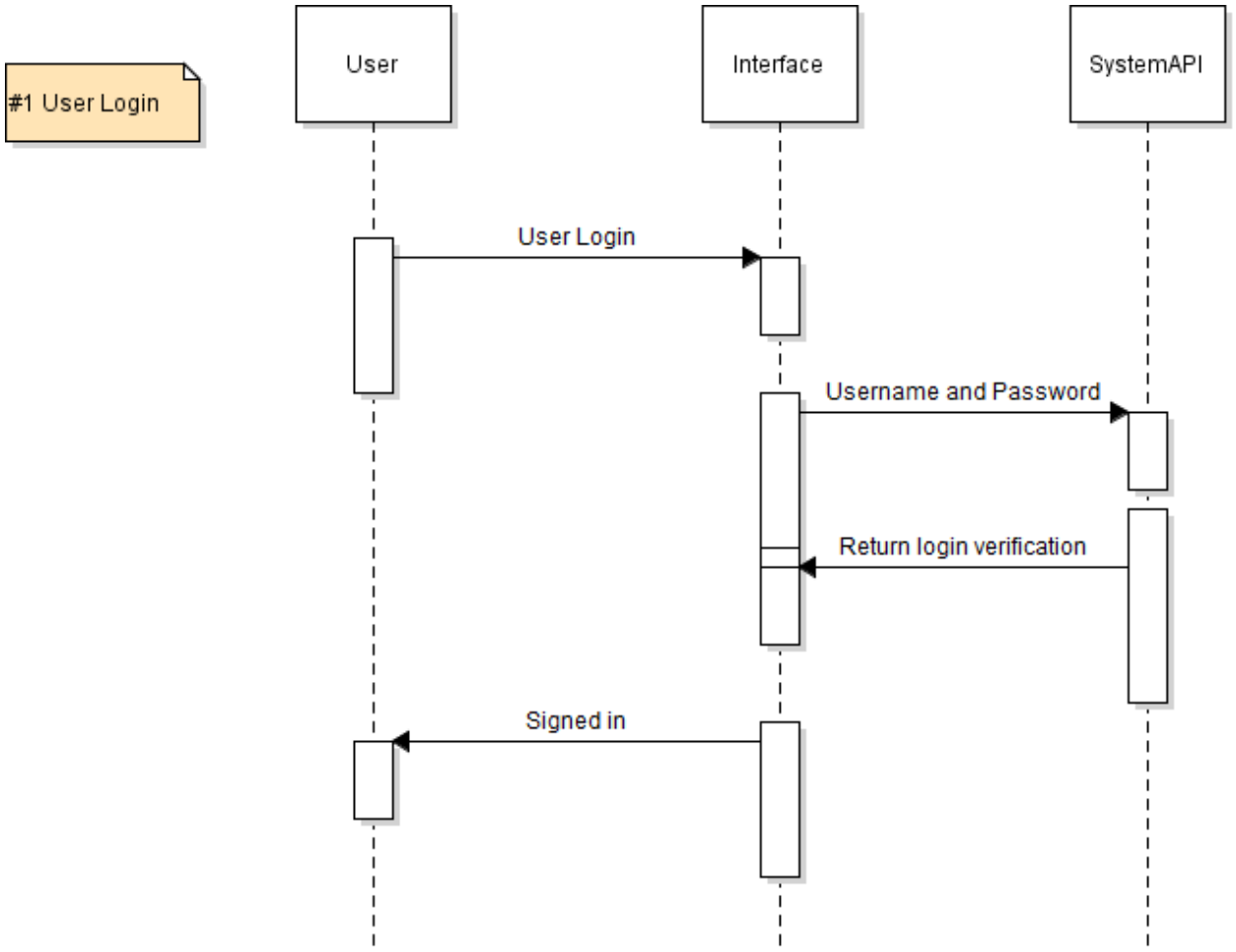
SystemAPI	
Manages the UI and API of the application **Logs in user** **Changes and modifies the customers and sellers** **Modifies users inventory or shopping cart**	**LinkedList<Customers>** **LinkedList<Sellers>**

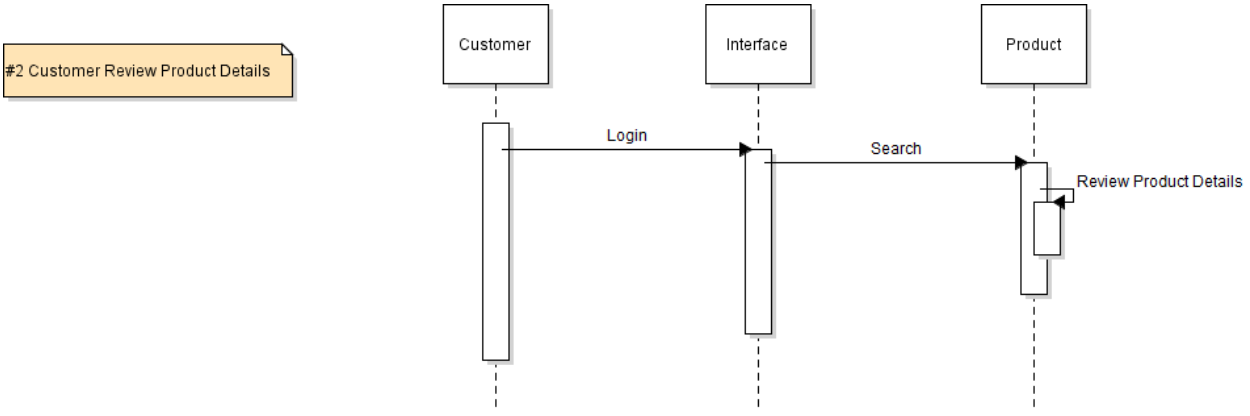
2.2 Class Diagram: (Josh)

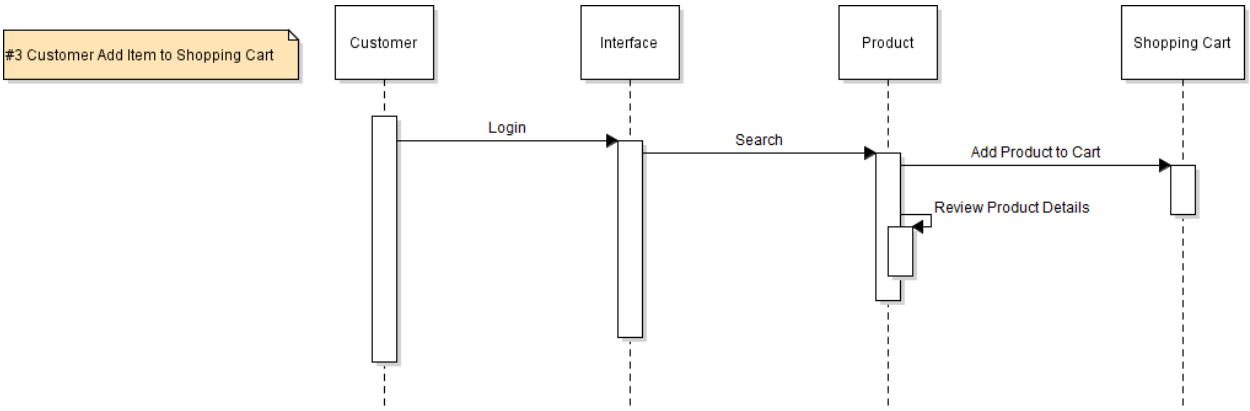


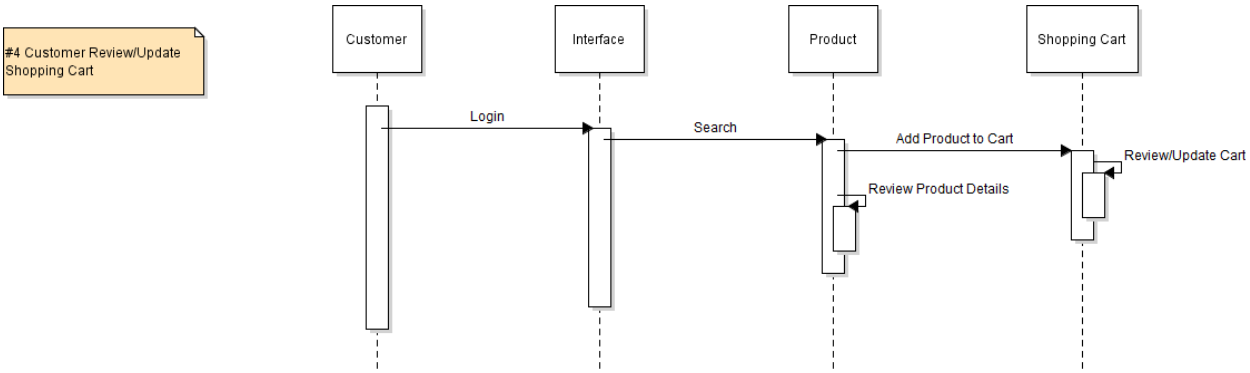
2.2 Sequence diagrams: (Mohammed)

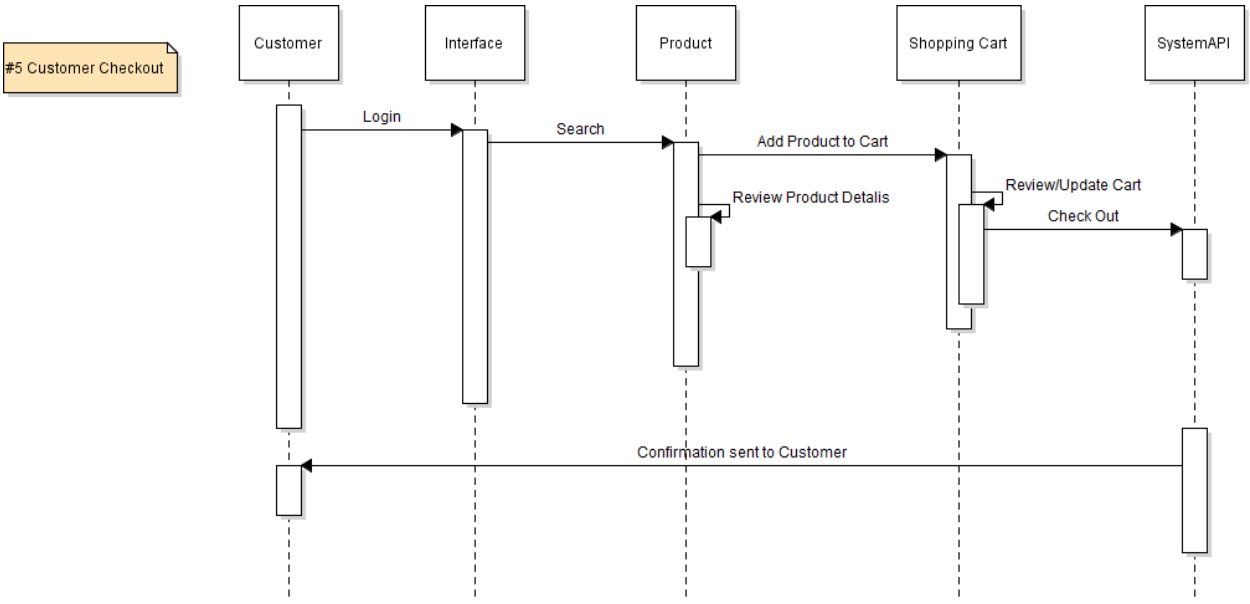


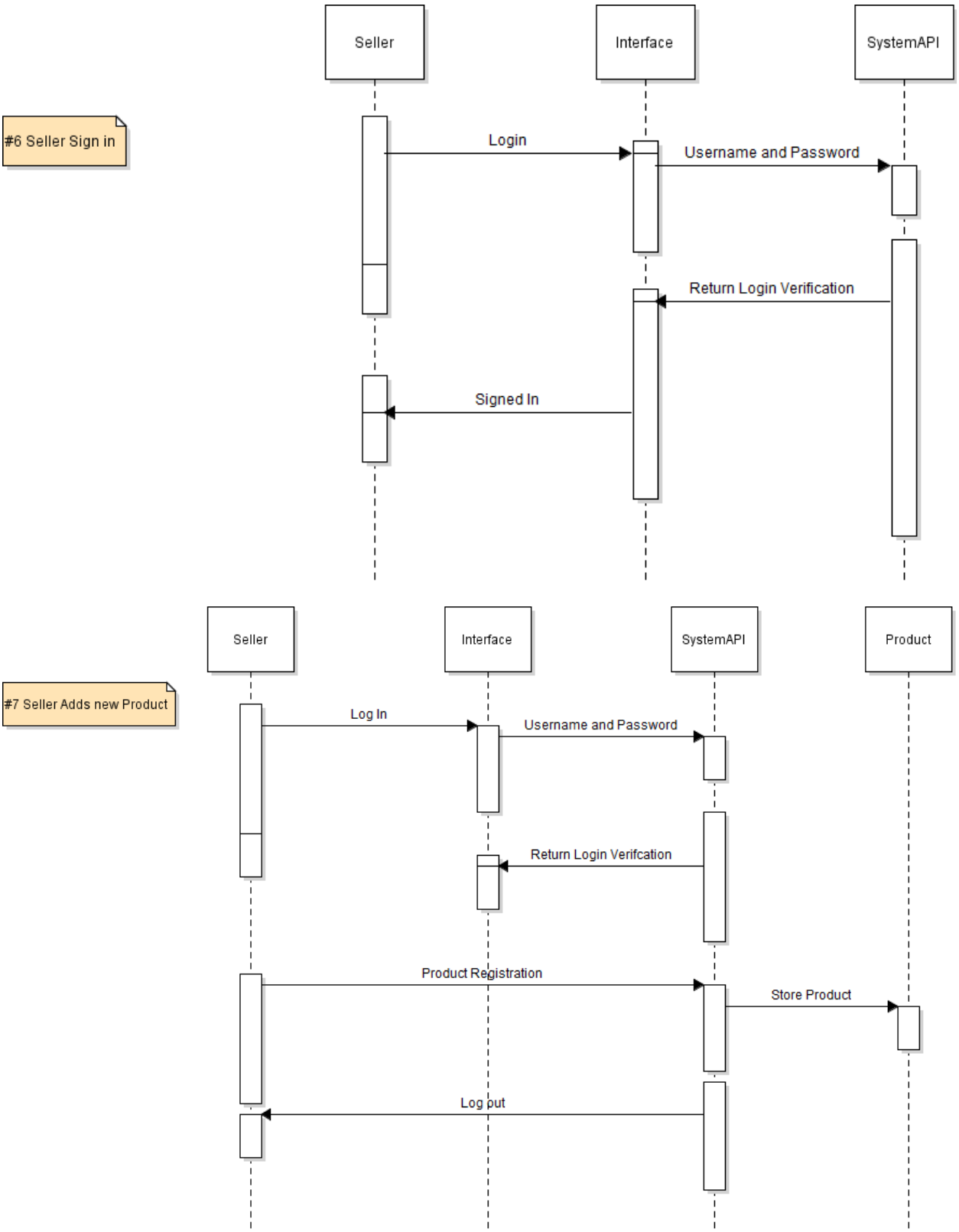

#1 User Login
User
Interface
SystemAPI
User Login
Username and Password
Return login verification
Signed in

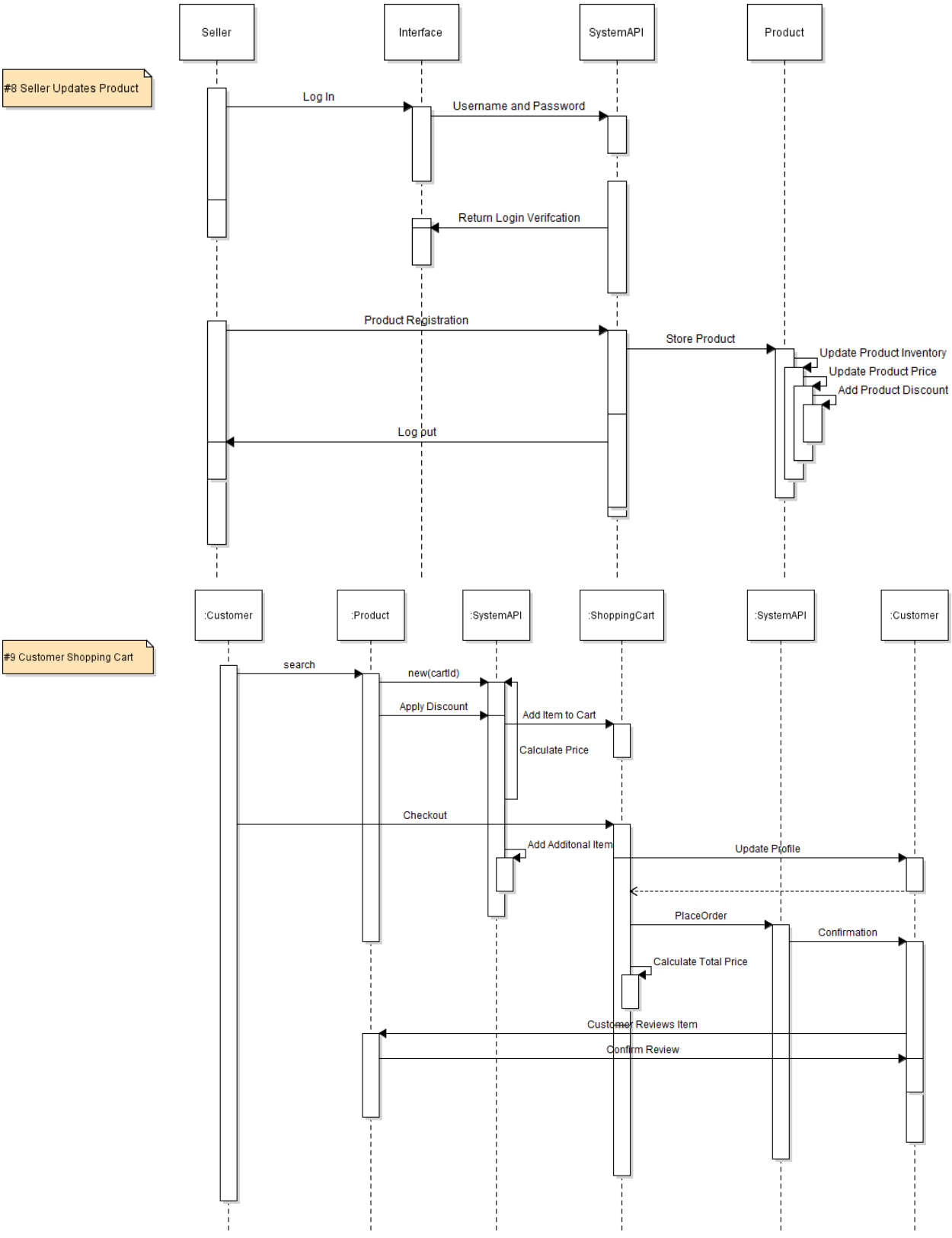

#2 Customer Review Product Details
Customer
Interface
Product
Login
Search
Review Product Details


#3 Customer Add Item to Shopping Cart
Customer
Interface
Product
Shopping Cart
Login
Search
Add Product to Cart
Review Product Details

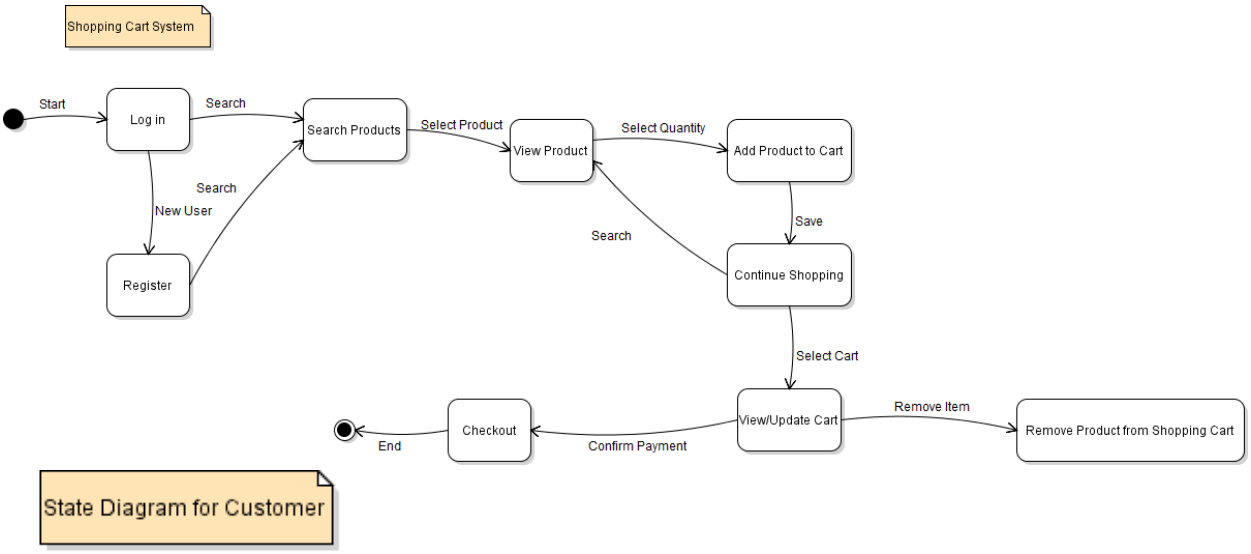

#4 Customer Review/Update Shopping Cart
Customer
Interface
Product
Shopping Cart
Login
Search
Add Product to Cart
Review/Update Cart
Review Product Details


#5 Customer Checkout
Customer
Interface
Product
Shopping Cart
SystemAPI
Login
Search
Add Product to Cart
Review/Update Cart
Review Product Details
Check Out
Confirmation sent to Customer

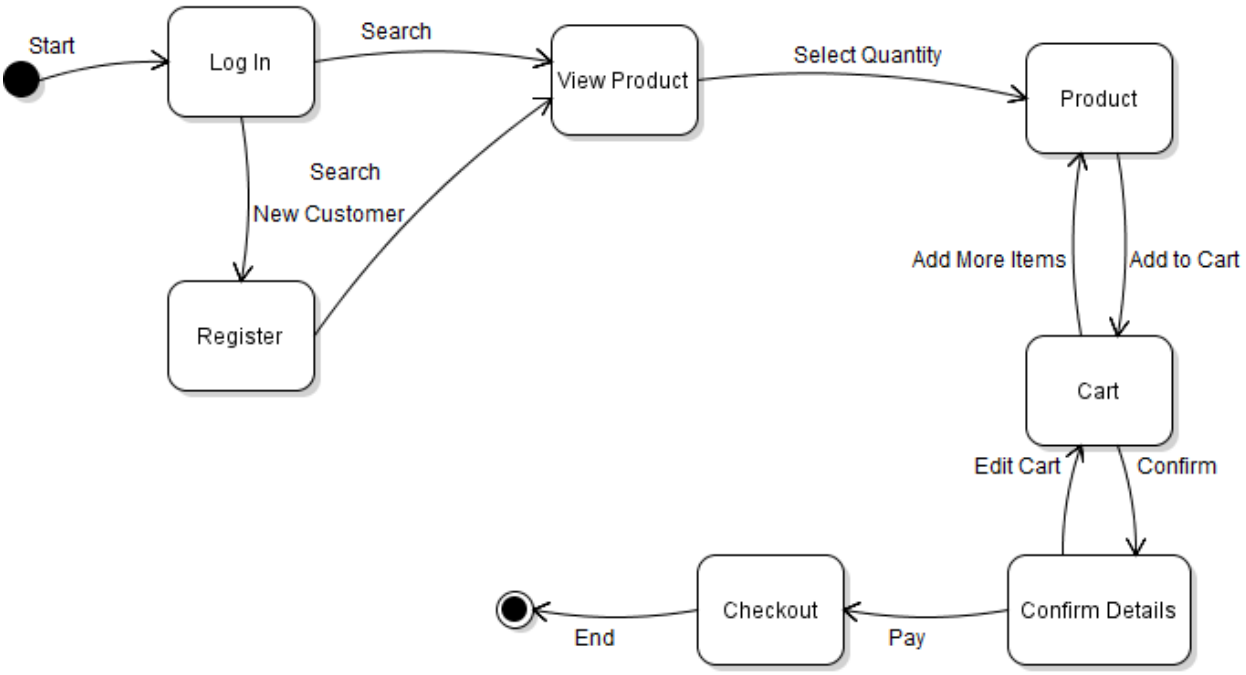

#6 Seller Sign in
Seller
Interface
SystemAPI
Login
Username and Password
Return Login Verification
Signed In
#7 Seller Adds new Product
Seller
Interface
SystemAPI
Product
Log In
Username and Password
Return Login Verifcation
Product Registration
Store Product
Log out

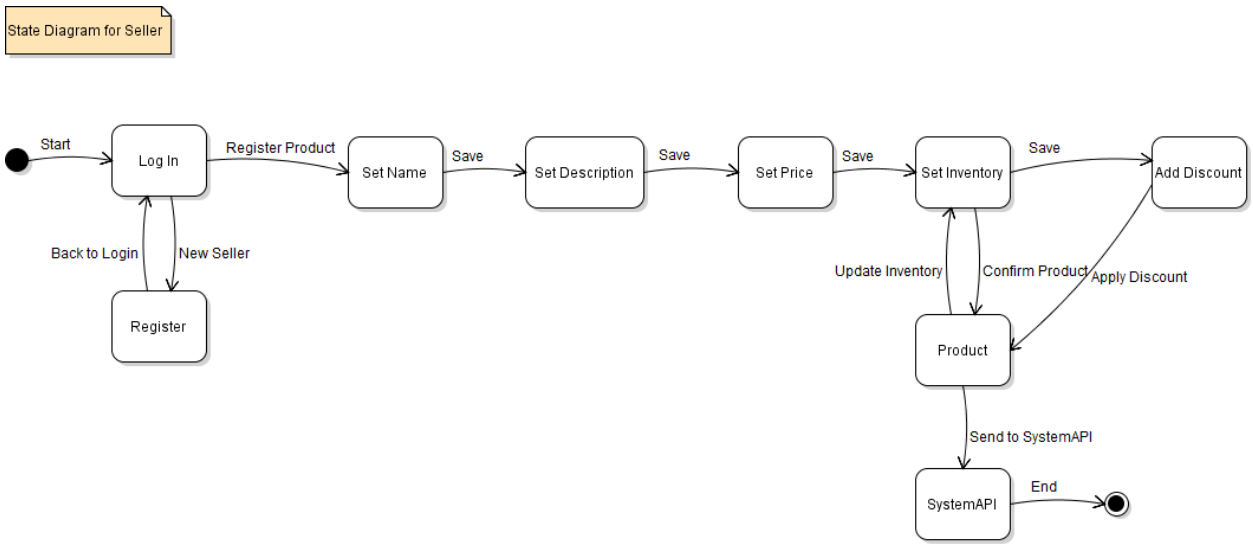

#8 Seller Updates Product
Seller
Interface
SystemAPI
Product
Log In
Username and Password
Return Login Verifcation
Product Registration
Store Product
Update Product Inventory
Update Product Price
Add Product Discount
Log out
#9 Customer Shopping Cart
:Customer
:Product
:SystemAPI
:ShoppingCart
:SystemAPI
:Customer
search
new(cartId)
Apply Discount
Add Item to Cart
Calculate Price
Checkout
Add Additonal Item
Update Profile
PlaceOrder
Confirmation
Calculate Total Price
Customer Reviews Item
Confirm Review

2.2 State diagrams: (Mohammed)



State Diagram for Customer




State Diagram for Seller
Start
Log In
Register Product
Set Name
Save
Set Description
Save
Set Price
Save
Set Inventory
Save
Add Discount
Back to Login
New Seller
Register
Update Inventory
Confirm Product
Apply Discount
Product
Send to SystemAPI
SystemAPI
End

3. code (Josh)

Prod.java

```
package shoppingcart;

/**
 *
 * @author Joshua Sohan
 */
public interface Prod {

   public void SetName(String val);
   public void SetPrice(double val);
   public void SetDescription(String val);
   public void SetInventory(int val);
   public double GetPrice();
   public String GetDescription();
   public String Getname();
   public int GetInventory();
}
```

User.java

```
package shoppingcart;

import java.util.LinkedList;

/**
 * @author Joshua Sohan
 */
public interface User {

   void Add(Product val);
   void Add(DiscountProduct val);
   void Add(BundleProduct val);
   void Remove(Product val);
   void Remove(DiscountProduct val);
   void Remove(BundleProduct val);
   PDList<Product> ProdList();
   PDList<DiscountProduct> DisList();
   PDList<BundleProduct> BunList();
   int num();
   void SetName(String val);
   void SetPassword(String val);
   void SetUsername(String val);
```

```
    String GetName();
    String GetPassword();
    String GetUsername();

}
```

ProdList.java

```
package shoppingcart;

import java.util.Collection;
import javax.swing.JPanel;

/**
 *
 * @author Joshua sohan
 */
public interface ProdList<E> {

    void  add(E e);
    void remove(E e);
    E getItem(int i);
    int size();
    boolean isEmpty();
    void removeAll();
    void addAll(Collection <? extends E> c);
}
```

Product.java

```
package shoppingcart;

import java.io.Serializable;

/**
 *
 * @author Joshua Sohan
 */
public class Product implements Prod, Serializable{


    private String name;
    private double price;
    private String description;
    private int inventory;

    public Product(){
        name =null;
        price = 0;
        description = null;
```

```
    inventory  = 0;
}

/**
 * Sets the name
 * @param val String that becomes the name
 */
@Override
public void SetName(String val) {
    name = val;
}


/**
 * Sets the price
 * @param val double that becomes the price
 */
@Override
public void SetPrice(double val) {
    price = val;

}


/**
 * Sets the Description
 * @param val String the becomes the description
 */
@Override
public void SetDescription(String val) {
    description = val;
}


/**
 * Sets the inventory
 * @param val Int that becomes the inventory
 */
@Override
public void SetInventory(int val) {
    inventory = val;
}


/**
 * Gets the price
 * @return double of the items price price
 */
```

```java
@Override
public double GetPrice() {
   return price;

}


/**
 * Gets the description
 * @return String of the items description
 */
@Override
public String GetDescription() {
   return description;
}


/**
 * Gets the name
 * @return String of items name
 */
@Override
public String Getname(){
   return name;
}


/**
 * Gets inventory
 * @return int of the items inventory
 */
@Override
public int GetInventory() {
   return inventory;
}


/**
 * returns copy of the product
 * @return Prodcut type that is a copy of itself
 */
public Product Getcopy() {
   Product n = new Product();
   n.SetPrice(price);
   n.SetDescription(description);
   n.SetName(name);
   n.SetInventory(inventory);
```

```
        return n;
    }

}
```

BundleProduct.java

```
package shoppingcart;

import java.io.Serializable;
import java.util.LinkedList;
import java.lang.String;

/**
 *
 * @author Joshua sohan
 */
public class BundleProduct implements Prod, Serializable{

    private String name;
    private double price;
    private String description;
    private int inventory;

    private LinkedList<Product> prod = new LinkedList<>();
    private LinkedList<DiscountProduct> discountProd = new LinkedList<>();

    /**
     *Class constructor initializes all variables to null or 0
     */
    public void BundleProduct(){
        name = null;
        price = 0;
        description = "";
        inventory = 0;
    }

    /**
     * Adds product type object to the Bundle
     * @param val Product type object you want to add to the bundle
     */

    public void AddProd(Product val){
        Product temp = new Product();
        temp.SetDescription(val.GetDescription());
        temp.SetInventory(val.GetInventory());
        temp.SetPrice(price + val.GetPrice());
        temp.SetName(val.Getname());
```

```
    prod.add(temp);
    SetDescription("");
}


/**
 * Adds DiscountProduct object type to the Bundle
 * @param val The DiscountProduct you want to add to the bundle
 */
public void AddProd(DiscountProduct val){
    Product temp = new Product();
    temp.SetDescription(val.GetDescription());
    temp.SetInventory(val.GetInventory());
    temp.SetName(val.Getname());
    temp.SetPrice(price + val.GetPrice());
    discountProd.add(val);
    SetDescription("");
}


/**
 * Sets the Bundle's name
 * @param val String you want to set as the Bundles name
 */
@Override
public void SetName(String val) {
    name = val;
}

/**
 * Sets the Bundle's price
 * @param val Double type of price
 */
@Override
public void SetPrice(double val) {
    price = val;
}

/**
 * Sets the Bundle's description
 * @param val The string you want to set as the description
 */
@Override
public void SetDescription(String val) {
    String t = "Bundle Of: " + this.GetString();
    description = t;
}
```

```java
/**
 * Sets the Bundle's inventory (How many in stocks)
 * @param val  int to set as the inventory
 */
@Override
public void SetInventory(int val) {
   inventory = val;
}

/**
 * If Price is never specified then it adds up the total prices for each product added
 * @return Returns he Price of Bundle
 */

@Override
public double GetPrice() {
   double temp = 0;
   if (price == 0 ){
      for (Product i : prod){
         temp = temp + i.GetPrice();
      }
      for (DiscountProduct i: discountProd){
         temp = temp + i.GetPrice();
      }
   }
   else{
      temp = price;
   }
   return temp;
}

/**
 * Returns the Bundle's description
 * @return String of the Bundle's description
 */
@Override
public String GetDescription() {
  return description;
}

/**
 * If custom name is not given then it creates a string of all names in bundle and returns it
 * @return A string of the name of the Bundle
 */
@Override
public String Getname() {
   String temp = "";
```

```
    if (name == null){
        temp = this.GetString();
    }
    else{
        temp = name;
    }
    return temp;
}


/**
 * Returns the Bundle's inventory
 * @return int of the items inventory
 */
@Override
public int GetInventory() {
    int temp = prod.get(0).GetInventory();
    if (inventory == 0){
        for (Product i: prod){
            if(temp > i.GetInventory()){
                temp = i.GetInventory();
            }
        }
        for (DiscountProduct i: discountProd){
            if(temp > i.GetInventory()){
                temp = i.GetInventory();
            }
        }
    }
    else
    {
        temp = inventory;
    }


    return temp;
}


/**
 * Returns a string of all the items inside the bundle
 * @return a string of all the items inside the bundle
 */
public String GetString(){
    String temp ="";

    for (int i = 0; i < prod.size(); i ++){
            if (i == 0){
```

```
            temp = temp + prod.get(i).Getname();
            }
            else{
               temp = temp + ", " + prod.get(i).Getname();
            }
         }
         for (DiscountProduct i: discountProd){
            temp = temp + ", " + i.Getname();
         }


      return temp;
   }


  /**
   * returns a copy of the BundleProduct
   * @return Prodcut objct type that is an exact copy
   */
   public BundleProduct Getcopy() {

      BundleProduct t = new BundleProduct();
      for (Product j: prod){
         t.AddProd(j);
      }

      for (DiscountProduct j: discountProd){
         t.AddProd(j);
      }


      t.SetDescription(this.GetDescription());
      t.SetInventory(this.GetInventory());
      t.SetName(this.Getname());
      t.SetPrice(this.GetPrice());

      return t;
   }

}
```

DiscountProduct.java

```
package shoppingcart;

import java.io.Serializable;
```

```java
/**
 *
 * @author Joshua Sohan
 */
public class DiscountProduct implements Prod, Serializable {

   private Product prod;


   /**
    * Class constructor, gets product and the price your wish to change
    * @param val Product
    * @param price Price
    */
   public DiscountProduct(Product val, double price){
      prod = new Product();
      prod.SetDescription(val.GetDescription());
      prod.SetInventory(val.GetInventory());
      prod.SetPrice(price);
      prod.SetName(val.Getname() + "[Discounted]");
   }

   public DiscountProduct(Product val){
      prod = new Product();
      prod.SetDescription(val.GetDescription());
      prod.SetInventory(val.GetInventory());
      prod.SetPrice(val.GetPrice());
      prod.SetName(val.Getname());
   }

   /**
    * Sets the name
    * @param val String that becomes the name
    */
   @Override
   public void SetName(String val) {
      prod.SetName(val);
   }

   /**
    * Sets the price
    * @param val double that becomes the price
    */
   @Override
   public void SetPrice(double val) {
      prod.SetPrice(val);
   }
```

```java
/**
 * Sets the Description
 * @param val String the becomes the description
 */
@Override
public void SetDescription(String val) {
   prod.SetDescription(val);
}

/**
 * Sets the inventory
 * @param val Int that becomes the inventory
 */
@Override
public void SetInventory(int val) {
   prod.SetInventory(val);
}

/**
 * Gets the price
 * @return double of the items price price
 */
@Override
public double GetPrice() {
   return prod.GetPrice();

}

/**
 * Gets the description
 * @return String of the items description
 */
@Override
public String GetDescription() {
   return prod.GetDescription();
}

/**
 * Gets the name
 * @return String of items name
 */
@Override
public String Getname(){
   return prod.Getname();
}

/**
```

```
    * Gets inventory
    * @return int of the items inventory
    */
    @Override
    public int GetInventory() {
        return prod.GetInventory();
    }


    /**
    * Makes a copy of itself
    * @return DiscountProduct that is a copy of itself
    */
    public DiscountProduct Getcopy() {
        DiscountProduct n = new DiscountProduct(prod);
        return n;
    }

}
```

Seller.java

```
package shoppingcart;

import java.io.Serializable;

/**
 *
 * @author Joshua Sohan
 */
public class Seller implements User, Serializable{
    private final PDList<Product> prod;
    private final PDList<DiscountProduct> discount;
    private final PDList<BundleProduct> bundle;
    private int arrayNum;
    private String username;
    private String name;
    private String password;

    public Seller(){
        arrayNum = 0;
        prod = new PDList();
        discount = new PDList();
        bundle = new PDList();
    }

    /**
    * Add prodcut to product
```

```
 * @param val Prodcut you want to add
 */
@Override
public void Add(Product val) {
   prod.add(val);
   arrayNum++;
}

/**
 * Add Discount product
 * @param val DiscountProdcut you wish to add
 */
@Override
public void Add(DiscountProduct val) {
   discount.add(val);
   arrayNum++;
}

/**
 * Add Bundle Product
 * @param val BundleProduct you wish to add
 */
@Override
public void Add(BundleProduct val) {
   bundle.add(val);
   arrayNum++;
}

/**
 * Remove Product
 * @param val Product you wish to remove
 */
@Override
public void Remove(Product val) {
   prod.remove(val);
   arrayNum--;
}

/**
 * removes Discount product
 * @param val DiscountProduct you wish to remove
 */
@Override
public void Remove(DiscountProduct val) {
   discount.remove(val);
   arrayNum--;
}
```

```java
/**
 * Removes Bundle Product
 * @param val BundleProduct you wish to remove
 */
@Override
public void Remove(BundleProduct val) {
   bundle.remove(val);
   arrayNum--;
}


/**
 * returns the number of total products
 * @return int of total products
 */
@Override
public int num() {
   return arrayNum;
}


/**
 * returns the list of of products
 * @return PDList of products
 */
@Override
public PDList<Product> ProdList() {
   return (PDList<Product>) prod;
}

/**
 * returns the list of DiscountProducts
 * @return PDList of discount products
 */
@Override
public PDList<DiscountProduct> DisList() {
   return (PDList<DiscountProduct>) discount;
}

/**
 * returns the list of BundleProducts
 * @return PDList type object
 */
@Override
public PDList<BundleProduct> BunList() {
   return (PDList<BundleProduct>) bundle;
}
```

```
/**
 * Sets name
 * @param val String to set name too
 */
@Override
public void SetName(String val) {
   name = val;
}

/**
 * Sets password
 * @param val String to set password to
 */
@Override
public void SetPassword(String val) {
   password = val;

}

/**
 * Sets Username
 * @param val String to set username too
 */
@Override
public void SetUsername(String val) {
   username = val;
}

/**
 * Gets name
 * @return String of name
 */
@Override
public String GetName() {

   return name;
}

/**
 * Gets password
 * @return String of password
 */
@Override
public String GetPassword() {
   return password;
}

/**
```

```
     * Gets Usename
     * @return String of username
     */
    @Override
    public String GetUsername() {
        return username;
    }
}
```

Customer.java

```
package shoppingcart;

import java.io.Serializable;

/**
 *
 * @author Joshua Sohan
 */
public class Customer implements User, Serializable{
    private final PDList<Product> prod;
    private final PDList<DiscountProduct> discount;
    private final PDList<BundleProduct> bundle;
    private int arrayNum;
    private String username;
    private String name;
    private String password;


    /**
     * Constructor class, intializes all vaulues to zero
     */
    public Customer(){
        this.arrayNum = 0;
        this.prod = new PDList();
        this.discount = new PDList();
        this.bundle = new PDList();
    }

    /**
     * Add prodcut to product
     * @param val Prodcut you want to add
     */
    @Override
    public void Add(Product val) {
        this.prod.add(val);
```

```
    this.arrayNum++;
}

/**
 * Add Discount product
 * @param val DiscountProdcut you wish to add
 */
@Override
public void Add(DiscountProduct val) {
    this.discount.add(val);
    this.arrayNum++;
}


/**
 * Add Bundle Product
 * @param val BundleProduct you wish to add
 */
@Override
public void Add(BundleProduct val) {
    this.bundle.add(val);
    this.arrayNum++;
}

/**
 * Remove Product
 * @param val Product you wish to remove
 */
@Override
public void Remove(Product val) {
    this.prod.remove(val);
    this.arrayNum--;
}

/**
 * removes Discount product
 * @param val DiscountProduct you wish to remove
 */
@Override
public void Remove(DiscountProduct val) {
    this.discount.remove(val);
    this.arrayNum--;
}

/**
 * Removes Bundle Product
 * @param val BundleProduct you wish to remove
 */
```

```java
@Override
public void Remove(BundleProduct val) {
   this.bundle.remove(val);
   this.arrayNum--;
}

/**
 * returns the number of total products
 * @return int of total products
 */
@Override
public int num() {
   return arrayNum;
}

/**
 * returns the list of of products
 * @return PDList of products
 */
@Override
public PDList<Product> ProdList() {
   return (PDList<Product>) this.prod;
}

/**
 * returns the list of DiscountProducts
 * @return PDLis of discount products
 */
@Override
public PDList<DiscountProduct> DisList() {
    return (PDList<DiscountProduct>) this.discount;
}

/**
 * returns the list of BundleProducts
 * @return PDList type object
 */
@Override
public PDList<BundleProduct> BunList() {
   return (PDList<BundleProduct>) this.bundle;
}

/**
 * Sets name
 * @param val String to set name too
 */
```

```java
@Override
public void SetName(String val) {
   this.name = val;
}

/**
 * Sets password
 * @param val String to set password to
 */
@Override
public void SetPassword(String val) {
   this.password = val;
}

/**
 * Sets Username
 * @param val String to set username too
 */
@Override
public void SetUsername(String val) {
   this.username = val;
}

/**
 * Gets name
 * @return String of name
 */
@Override
public String GetName() {

   return this.name;
}


/**
 * Gets password
 * @return String of password
 */
@Override
public String GetPassword() {
   return this.password;
}
```

```
    /**
     * Gets Usename
     * @return String of username
     */
    @Override
    public String GetUsername() {
        return this.username;
    }
}
```

PDList.java

```
package shoppingcart;

import java.io.Serializable;
import java.util.Collection;
import java.util.Iterator;
import java.util.LinkedList;
import javax.swing.JPanel;

/**
 *
 * @author Joshua Sohan
 */
public class PDList<E> implements ProdList<E>, Iterable<E>, Serializable {

    private LinkedList<E> lst;

    /**
     * Class constructor
     */
    public PDList(){
        lst = new LinkedList();
    }

    /**
     * Removes first occurence of object
     * @param e object to be removed
     */
    @Override
```

```java
public void remove(E e) {
   lst.remove(e);
}

/**
 * gets item at specific index
 * @param i index to get item
 * @return object at that index
 */
@Override
public E getItem( int i) {
  return  lst.get(i);
}

/**
 * Gets the size
 * @return int of how many items are in PDList
 */
@Override
public int size() {
   return lst.size();
 }

/**
 * tells you if it is empty
 * @return boolean on whether its true or not
 */
@Override
public boolean isEmpty() {
   return lst.isEmpty();
}


/**
 * Adds element to PDList
 * @param e element you wish to add
 */
@Override
public void add(E e) {
   lst.add(e);
}

/**
```

```
     * allows you to iterate
     * @return Iterator
     */
    @Override
    public Iterator<E> iterator() {
        return lst.iterator();
    }

    /**
     * adds a collection of objects to PDList
     * @param c any Collection containing the same object types
     */
    @Override
    public void addAll(Collection<? extends E> c) {
        for (E temp : c){
            lst.add(temp);
        }
    }

    @Override
    public void removeAll() {
        lst.clear();
    }
}
```

SystemAPI.java

```
package shoppingcart;

import java.awt.BorderLayout;
import java.awt.CardLayout;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.FileInputStream;
```

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.LinkedList;
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.math.BigDecimal;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.text.ParseException;
import java.util.Locale;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JPasswordField;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.ScrollPaneConstants;

/**
 *
 * @author Joshua Sohan
 */
public class SystemAPI {
    private LinkedList<Customer> cList;
    private LinkedList<Seller> sList;
    private Customer ccurrent;
    private Seller scurrent;
    int flag = 0;
    int flag2 = 0;
    private JPanel cards;
    private JFrame frame;
    JPanel card1;
    JPanel card2;
    JPanel card3;
    JPanel card4;
    JPanel card5;

    public SystemAPI() throws ClassNotFoundException{
```

```
// creates card layout
cards = new JPanel(new CardLayout());
frame = new JFrame("Shopping Cart");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

card1 = new JPanel();
card1.setLayout(new WrapLayout());
cards.add(new JScrollPane(card1), "Seller");

card3 = new JPanel();
card3.setLayout(new WrapLayout());
cards.add(new JScrollPane(card3), "Cart");

card4 = new JPanel();
card4.setLayout(new WrapLayout());

card2 = new JPanel();
card2.setLayout(new WrapLayout());
cards.add(  new JScrollPane(card2), "Customer");

card5 = new JPanel();
card5.setLayout(new WrapLayout());
cards.add( new JScrollPane(card5), "CheckOut");
cList = new LinkedList();
sList = new LinkedList();


//listener to sterilaize when you exit
frame.addWindowListener( new WindowListener(){
   @Override
   public void windowOpened(WindowEvent e) {}
   @Override
   public void windowClosed(WindowEvent e) {}
   @Override
   public void windowIconified(WindowEvent e) {}
   @Override
   public void windowDeiconified(WindowEvent e) {}
   @Override
   public void windowActivated(WindowEvent e) {}
   @Override
   public void windowDeactivated(WindowEvent e) {}
   @Override
```

```
    public void windowClosing(WindowEvent e) {
      try{
      FileOutputStream file = new FileOutputStream("List.dat");
      ObjectOutputStream out = new ObjectOutputStream(file);

      // Method for serialization of object
      out.writeObject(cList);
      out.writeObject(sList);

      out.close();
      file.close();
      System.out.println("Sucessfull Sterylize");
      }
      catch(IOException ex){
        System.out.println("Did not Sterylize");
      }
    }
  });


  try {
    //desteralizes and adds the objects into the linked lists
    FileInputStream file = new FileInputStream ("List.dat");
    ObjectInputStream in = new ObjectInputStream (file);

    cList.addAll((LinkedList<Customer>) in.readObject());

    sList.addAll((LinkedList<Seller>) in.readObject());

   in.close();
   file.close();
  }
  catch(IOException ex){
  }

  //shows the login API by default
  this.LoginAPI();
  CardLayout cl = (CardLayout)(cards.getLayout());
  cl.show(cards, "Login");

}

//helps convert money string to num
```

```
  public static BigDecimal parse(final String amount, final Locale locale) throws ParseException
{
    final NumberFormat format = NumberFormat.getNumberInstance(locale);
    if (format instanceof DecimalFormat) {
      ((DecimalFormat) format).setParseBigDecimal(true);
    }
    return (BigDecimal) format.parse(amount.replaceAll("[^\\d.,]",""));
  }

  //login API
  public void LoginAPI(){


    //welcome greating
    JLabel ProductLabel = new JLabel("Welcome, please enter Username/Password");
    Font fontprod = ProductLabel.getFont();
    float sizeprod = fontprod.getSize() + 20.0f;
    ProductLabel.setFont(fontprod.deriveFont(sizeprod));
    JPanel prodLabelPan = new JPanel();
    prodLabelPan.setLayout(new WrapLayout());
    prodLabelPan.setPreferredSize(new Dimension(1022,55));
    prodLabelPan.add(ProductLabel);

    //shows if uesrname/password dont match
    JLabel wrong = new JLabel("Sorry your Username and Password don't match");
    wrong.setFont(fontprod.deriveFont(sizeprod-10));
    JPanel wrongPan = new JPanel();
    wrongPan.setLayout(new WrapLayout());
    wrongPan.setPreferredSize(new Dimension(1022,55));
    wrongPan.setVisible(false);
    wrongPan.add(wrong);


    //password and username field to log in
    final JPanel loginPan1 = new JPanel();
    loginPan1.setLayout(new BoxLayout(loginPan1, BoxLayout.X_AXIS));

    JTextArea username = new JTextArea("Username:",1, 6);
    Font font = username.getFont();
    float size = font.getSize() + 10.0f;
    username.setFont(font.deriveFont(size));
    username.setMargin(new Insets(10,5,5,0));
    username.setEditable(false);
```

```
JTextArea userText = new JTextArea("",1, 10);
userText.setEditable(true);
userText.setFont(font.deriveFont(size));
userText.setMargin(new Insets(10,5,5,0));

loginPan1.add(username);
loginPan1.add(userText);

final JPanel loginPan = new JPanel();
loginPan.setLayout(new BoxLayout(loginPan, BoxLayout.X_AXIS));

JTextArea password = new JTextArea("Password:",1, 6);
password.setFont(font.deriveFont(size));
password.setMargin(new Insets(10,5,5,0));
password.setEditable(false);

JPasswordField passText = new JPasswordField(12);
passText.setEditable(true);
passText.setFont(font.deriveFont(size));
passText.setMargin(new Insets(10,5,5,0));

loginPan.add(password);
loginPan.add(passText);

card4.add(wrongPan);
card4.add(prodLabelPan);
card4.add(loginPan1);
card4.add(loginPan);

//button and action listner to check if real
JButton sign = new JButton("Login");
sign.setFont(font.deriveFont(size));

//action listener to log in cutomer
sign.addActionListener(new ActionListener(){
  private int i = 0;
  @Override
  public void actionPerformed(ActionEvent e) {
    //1 = costumer
    //2 = Seller
    //0 = no usernames match
    String pass = new String(passText.getPassword());
```

```
            String username = userText.getText().toLowerCase();

            for(int j = 0; j < cList.size(); j++){
               if (cList.get(j).GetUsername().equals(username) &&
cList.get(j).GetPassword().equals(pass)){
                  i = 1;
                  ccurrent = cList.get(j);
               }
            }
            for(int j = 0; j < sList.size(); j++){
               if (sList.get(j).GetUsername().equals(username) &&
sList.get(j).GetPassword().equals(pass)){
                  i = 2;
                  scurrent = sList.get(j);
               }
            }
            if (i == 1){
               CustomerAPI();
               CardLayout cl = (CardLayout)(cards.getLayout());
               cl.show(cards, "Customer");
            }
            if (i == 2){
               SellerAPI();
               CardLayout cl = (CardLayout)(cards.getLayout());
               cl.show(cards, "Seller");
            }
            if (i == 0){
               wrongPan.setVisible(true);
            }
         }

      });

      card4.add(sign);

      //create a new user textAreaas
      JLabel newUserLabel = new JLabel("Welcome, Create new Customer/Seller account here");
      newUserLabel.setFont(fontprod.deriveFont(sizeprod));
      JPanel newUserPan = new JPanel();
      newUserPan.setLayout(new BoxLayout(newUserPan, BoxLayout.X_AXIS));
      newUserPan.setPreferredSize(new Dimension(1022,100));
      newUserPan.setMinimumSize(new Dimension(1022,100));
      newUserPan.add(newUserLabel);
```

```
final JPanel createUser1 = new JPanel();
createUser1.setLayout(new BoxLayout(createUser1, BoxLayout.X_AXIS));

JTextArea username2 = new JTextArea("Username:",1, 6);
username2.setFont(font.deriveFont(size));
username2.setMargin(new Insets(10,5,5,0));
username2.setEditable(false);

JTextArea userText2 = new JTextArea("",1, 10);
userText2.setEditable(true);
userText2.setFont(font.deriveFont(size));
userText2.setMargin(new Insets(10,5,5,0));

createUser1.add(username2);
createUser1.add(userText2);

final JPanel createUser = new JPanel();
createUser.setLayout(new BoxLayout(createUser, BoxLayout.X_AXIS));

JTextArea password2 = new JTextArea("Password:",1, 6);
password2.setFont(font.deriveFont(size));
password2.setMargin(new Insets(10,5,5,0));
password2.setEditable(false);

JPasswordField passText2 = new JPasswordField(12);
passText2 .setEditable(true);
passText2 .setFont(font.deriveFont(size));
passText2 .setMargin(new Insets(10,5,5,0));

createUser.add(password2);
createUser.add(passText2);

//check what kind of user to create
String[] select1 = {"Customer"," Seller"};

JComboBox select = new JComboBox(select1);


//button and action istner to create new user and log them in
JButton create = new JButton("Create");
create.setFont(font.deriveFont(size));
create.addActionListener(new ActionListener (){
```

```
    @Override
    public void actionPerformed(ActionEvent e) {
      //check if boxes are empty
      if (userText2.getText() != "" && passText2.getPassword().length != 0){

        String name = userText2.getText().toLowerCase();
        String pass = new String(passText2.getPassword());

        //see if customer or seller selected
        if (select.getSelectedItem().equals(select1[0])){
          Customer temp = new Customer();
          temp.SetUsername(name);
          temp.SetPassword(pass);
          cList.add(temp);
          ccurrent = temp;

          CustomerAPI();
          CardLayout cl = (CardLayout)(cards.getLayout());
          cl.show(cards, "Customer");
        }
        else {

          Seller temp = new Seller();
          temp.SetUsername(name);
          temp.SetPassword(pass);
          sList.add(temp);
          scurrent = temp;
          SellerAPI();
          CardLayout cl = (CardLayout)(cards.getLayout());
          cl.show(cards, "Seller");
        }

      }

    }

});
//add to cards
card4.add(newUserLabel);
card4.add(createUser1);
card4.add(createUser);
card4.add(select);
card4.add(create);
```

```
    cards.add(card4, "Login");
    frame.add(cards);
    frame.pack();
            frame.setVisible(true);

}


//handles the API for the Cusomer Users
public void CustomerAPI(){

    //panel where items are laoded into
    final JPanel prodPan = new JPanel();
    prodPan.setLayout(new WrapLayout());

    //greating + name of cutomer
    JLabel ProductLabel = new JLabel("Welcome, " + ccurrent.GetUsername());
    Font fontprod = ProductLabel.getFont();
    float sizeprod = fontprod.getSize() + 20.0f;
    ProductLabel.setFont(fontprod.deriveFont(sizeprod));
    JPanel prodLabelPan = new JPanel();
    prodLabelPan.setLayout(new BorderLayout());
    prodLabelPan.setPreferredSize(new Dimension(1022,55));
    prodLabelPan.add(ProductLabel, BorderLayout.WEST);


    //button that takes you to schopping cart
    JButton shopButton = new JButton("Shopping Cart");
    shopButton.addActionListener(new ActionListener(){
      @Override
      public void actionPerformed(ActionEvent e) {
        card3.removeAll();
        ShopCart();
        CardLayout cl = (CardLayout)(cards.getLayout());
        cl.show(cards, "Cart");
      }

    });

    prodLabelPan.add(shopButton, BorderLayout.EAST);
    card2.add( prodLabelPan);
```

```
//for loop that goes through all seller accounts
for (int j = 0; j < sList.size(); j++)
{
   Seller current = sList.get(j);

   /// gooes through all current sellers items and lists them
   for (int i = 0; i < current.ProdList().size(); i ++){

      //current product
      final Product prod = current.ProdList().getItem(i);

      //temp panel to laod all items onto
      final JPanel temp = new JPanel();
      temp.setLayout(new BoxLayout(temp, BoxLayout.X_AXIS));

      //name text area
      JTextArea name = new JTextArea(prod.Getname(),4, 15);
      Font font = name.getFont();
      float size = font.getSize() + 3.0f;
      name.setFont(font.deriveFont(size + 6.0f));
      name.setMargin(new Insets(2,10,0,10));
      name.setLineWrap(true);
      name.setWrapStyleWord(true);

      //descreiption
      JTextArea description = new JTextArea(prod.GetDescription(),4, 20);
      description.setFont(font.deriveFont(size));
      description.setLineWrap(true);
      description.setWrapStyleWord(true);
      description.setMargin(new Insets(0,10,0,10));
      JScrollPane d = new JScrollPane(description);
      d.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

      //inventory
      JTextArea inventory = new JTextArea(Integer.toString(prod.GetInventory()),4, 5);
      inventory.setFont(font.deriveFont(size));
      inventory.setMargin(new Insets(20,30,0,0));

      //price
      NumberFormat formatter = NumberFormat.getCurrencyInstance();
      JTextArea price = new JTextArea(formatter.format(prod.GetPrice()),4, 10);
      price.setFont(font.deriveFont(size));
      price.setMargin(new Insets(20,30,0,0));
```

```
//inventory combobox
String[] comboxs = new String[prod.GetInventory()+1];

for(int k = 0; k < prod.GetInventory()+1; k++){
   comboxs[k] = Integer.toString(k);
}

JComboBox inbox = new JComboBox(comboxs);
inbox.setSelectedIndex(0);
inbox.setPreferredSize(new Dimension(50,10));

//sets fields to false
name.setEditable(false);
description.setEditable(false);
inventory.setEditable(false);
price.setEditable(false);

//buton to add item to shopping cart
JButton addCart = new JButton("Add To Cart");
addCart.setVisible(true);

//action listener to add items to shoppping cart
ActionListener add = new ActionListener(){
   @Override
   public void actionPerformed(ActionEvent e) {
      int num = Integer.parseInt((String)inbox.getSelectedItem());

      if (num > 0){
         //makes copy of prodcut and changes the inventory
         Product newp = prod.Getcopy();
         newp.SetInventory(num);
         ccurrent.Add(newp);

         //sets the items inventory acordingly
         prod.SetInventory(prod.GetInventory() - num);

         //refersh customer
         card2.removeAll();
         CustomerAPI();
         card2.repaint();
         card2.revalidate();
```

```
                //refesh shopping cart
                card3.removeAll();
                ShopCart();
                card3.repaint();
                card3.revalidate();
            }
          }

        };///////////////==========================================================
        //add Listeners
        addCart.addActionListener(add);

        //add componenets to the row
        temp.add(name);
        temp.add(d);
        temp.add(inventory);
        temp.add(price);
        temp.add(inbox);
        temp.add(addCart);

        //ad row to the product panel
        prodPan.add(temp);
    }



    /////does the same thing as the Product loop but for Discountedproduct type
items++++++++++++++++++++++++++++
    for (int i = 0; i < current.DisList().size(); i ++){

        //current product
        final DiscountProduct prod = current.DisList().getItem(i);

        final JPanel temp = new JPanel();
        temp.setLayout(new BoxLayout(temp, BoxLayout.X_AXIS));

        //name text area
        JTextArea name = new JTextArea(prod.Getname(),4, 15);
        Font font = name.getFont();
        float size = font.getSize() + 3.0f;
        name.setFont(font.deriveFont(size + 6.0f));
        name.setMargin(new Insets(2,10,0,10));
```

```
name.setLineWrap(true);
name.setWrapStyleWord(true);

//descreiption
JTextArea description = new JTextArea(prod.GetDescription(),4, 20);
description.setFont(font.deriveFont(size));
description.setLineWrap(true);
description.setWrapStyleWord(true);
description.setMargin(new Insets(0,10,0,10));
JScrollPane d = new JScrollPane(description);
d.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

//inventory
JTextArea inventory = new JTextArea(Integer.toString(prod.GetInventory()),4, 5);
inventory.setFont(font.deriveFont(size));
inventory.setMargin(new Insets(20,30,0,0));

//price
NumberFormat formatter = NumberFormat.getCurrencyInstance();
JTextArea price = new JTextArea(formatter.format(prod.GetPrice()),4, 10);
price.setFont(font.deriveFont(size));
price.setMargin(new Insets(20,30,0,0));

String[] comboxs = new String[prod.GetInventory()+1];

for(int k = 0; k < prod.GetInventory()+1; k++){
   comboxs[k] = Integer.toString(k);
}

JComboBox inbox = new JComboBox(comboxs);
inbox.setSelectedIndex(0);
inbox.setPreferredSize(new Dimension(50,10));


name.setEditable(false);
description.setEditable(false);
inventory.setEditable(false);
price.setEditable(false);

JButton addCart = new JButton("Add To Cart");
addCart.setVisible(true);
```

```
    ActionListener add = new ActionListener(){
      @Override
      public void actionPerformed(ActionEvent e) {
        int num = Integer.parseInt((String)inbox.getSelectedItem());

        if (num > 0){

          DiscountProduct newp = prod.Getcopy();
          newp.SetInventory(num);
          ccurrent.Add(newp);

          prod.SetInventory(prod.GetInventory() - num);

          card2.removeAll();
          CustomerAPI();
          card2.repaint();
          card2.revalidate();


          card3.removeAll();
          ShopCart();
          card3.repaint();
          card3.revalidate();

        }
      }
    };
    //add Listeners
    addCart.addActionListener(add);

    //add componenets to the row
    temp.add(name);
    temp.add(d);
    temp.add(inventory);
    temp.add(price);
    temp.add(inbox);
    temp.add(addCart);

    //ad row to the product panel
    prodPan.add(temp);
  }///////////////++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
        ///////Does the same thing but fo Bundle Type
products**********************************
        for (int i = 0; i < current.BunList().size(); i ++){

            //current product
            final BundleProduct prod = current.BunList().getItem(i);

            final JPanel temp = new JPanel();
            temp.setLayout(new BoxLayout(temp, BoxLayout.X_AXIS));

            //name text area
            JTextArea name = new JTextArea(prod.Getname(),4, 15);
            Font font = name.getFont();
            float size = font.getSize() + 3.0f;
            name.setFont(font.deriveFont(size + 6.0f));
            name.setMargin(new Insets(2,10,0,10));
            name.setLineWrap(true);
            name.setWrapStyleWord(true);

            //descreiption
            JTextArea description = new JTextArea(prod.GetDescription(),4, 20);
            description.setFont(font.deriveFont(size));
            description.setLineWrap(true);
            description.setWrapStyleWord(true);
            description.setMargin(new Insets(0,10,0,10));
            JScrollPane d = new JScrollPane(description);
            d.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

            //inventory
            JTextArea inventory = new JTextArea(Integer.toString(prod.GetInventory()),4, 5);
            inventory.setFont(font.deriveFont(size));
            inventory.setMargin(new Insets(20,30,0,0));

            //price
            NumberFormat formatter = NumberFormat.getCurrencyInstance();
            JTextArea price = new JTextArea(formatter.format(prod.GetPrice()),4, 10);
            price.setFont(font.deriveFont(size));
            price.setMargin(new Insets(20,30,0,0));

            String[] comboxs = new String[prod.GetInventory()+1];

            for(int k = 0; k < prod.GetInventory()+1; k++){
```

```
    comboxs[k] = Integer.toString(k);
}

JComboBox inbox = new JComboBox(comboxs);
inbox.setSelectedIndex(0);
inbox.setPreferredSize(new Dimension(50,10));


name.setEditable(false);
description.setEditable(false);
inventory.setEditable(false);
price.setEditable(false);

JButton addCart = new JButton("Add To Cart");
addCart.setVisible(true);


ActionListener add = new ActionListener(){
   @Override
   public void actionPerformed(ActionEvent e) {
      int num = Integer.parseInt((String)inbox.getSelectedItem());

      if (num > 0){

         BundleProduct newp = prod.Getcopy();
         newp.SetInventory(num);
         ccurrent.Add(newp);

         prod.SetInventory(prod.GetInventory() - num);

         card2.removeAll();
         CustomerAPI();
         card2.repaint();
         card2.revalidate();


         card3.removeAll();
         ShopCart();
         card3.repaint();
         card3.revalidate();
      }
   }
```

```
        };
        //add Listeners
        addCart.addActionListener(add);

        //add componenets to the row
        temp.add(name);
        temp.add(d);
        temp.add(inventory);
        temp.add(price);
        temp.add(inbox);
        temp.add(addCart);

        //ad row to the product panel
        prodPan.add(temp);
      }///////////////**********************************************************

    }


    //makes it prodpan scrolable and adds it to card
    JScrollPane prodScroll = new JScrollPane(prodPan);
    prodScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    prodScroll.setPreferredSize(new Dimension(1022,700));
    card2.add(prodScroll);
    frame.pack();
}


//shopping cart API
public void ShopCart(){


    //flag to hide Succefull Purchase label
    flag2 = 0;
    //panel that holds all products in cart
    JPanel shop = new JPanel();
    shop.setLayout(new WrapLayout());

    //greeting
    JLabel ProductLabel = new JLabel("Shopping Cart:" );
    Font fontprod = ProductLabel.getFont();
    float sizeprod = fontprod.getSize() + 20.0f;
    ProductLabel.setFont(fontprod.deriveFont(sizeprod));
```

```
JPanel prodLabelPan = new JPanel();
prodLabelPan.setLayout(new BorderLayout());
prodLabelPan.setPreferredSize(new Dimension(1022,55));
prodLabelPan.add(ProductLabel, BorderLayout.WEST);


//go back to shopping cart and refresh customerAPI
JButton main = new JButton("Go Back Shopping");
main.addActionListener(new ActionListener(){
  @Override
  public void actionPerformed(ActionEvent e) {
     card2.removeAll();
     CustomerAPI();
     card2.repaint();
     card2.revalidate();
     CardLayout cl = (CardLayout)(cards.getLayout());
     cl.show(cards, "Customer");
  }

});
prodLabelPan.add(main, BorderLayout.EAST);


//takes cutomer to checkout
JButton next = new JButton("Check out");
next.addActionListener(new ActionListener(){
  @Override
  public void actionPerformed(ActionEvent e) {
     card5.removeAll();
     Checkout();
     card5.repaint();
     card5.revalidate();
     CardLayout cl = (CardLayout)(cards.getLayout());
     cl.show(cards, "CheckOut");
  }

});
prodLabelPan.add(next, BorderLayout.CENTER);


card3.add(prodLabelPan);
```

```
//shopping cart pordcuts in a list
for(int i = 0; i < ccurrent.ProdList().size(); i++){
   Product prod = ccurrent.ProdList().getItem(i);

   final JPanel temp = new JPanel();
   temp.setLayout(new BoxLayout(temp, BoxLayout.X_AXIS));

   //Name
   JTextArea name = new JTextArea(prod.Getname(),4, 15);
   Font font = name.getFont();
   float size = font.getSize() + 3.0f;
   name.setFont(font.deriveFont(size + 6.0f));
   name.setMargin(new Insets(2,10,0,10));
   name.setLineWrap(true);
   name.setWrapStyleWord(true);

   //description
   JTextArea description = new JTextArea(prod.GetDescription(),4, 20);
   description.setFont(font.deriveFont(size));
   description.setLineWrap(true);
   description.setWrapStyleWord(true);
   description.setMargin(new Insets(0,10,0,10));
   JScrollPane d = new JScrollPane(description);
   d.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

   //inventory
   JTextArea inventory = new JTextArea(Integer.toString(prod.GetInventory()),4, 5);
   inventory.setFont(font.deriveFont(size));
   inventory.setMargin(new Insets(20,30,0,0));

   //price
   NumberFormat formatter = NumberFormat.getCurrencyInstance();
   JTextArea price = new JTextArea(formatter.format(prod.GetPrice()),4, 10);
   price.setFont(font.deriveFont(size));
   price.setMargin(new Insets(20,30,0,0));

   //combbox to display how many items you want
   String[] comboxs = new String[prod.GetInventory()+1];

   for(int k = 0; k < prod.GetInventory()+1; k++){
      comboxs[k] = Integer.toString(k);
   }
```

```
        JComboBox inbox = new JComboBox(comboxs);
        inbox.setSelectedIndex(prod.GetInventory());
        inbox.setPreferredSize(new Dimension(50,10));


        name.setEditable(false);
        description.setEditable(false);
        inventory.setEditable(false);
        price.setEditable(false);

        //updates item button
        JButton update = new JButton("Update Item");
        update.setVisible(true);

        //update shpoing carts and adjusts inventory
        ActionListener updateCart = new ActionListener(){
          @Override
          public void actionPerformed(ActionEvent e) {
            int num = Integer.parseInt((String)inbox.getSelectedItem());

            if (num > 0){
              //fix item inventory
              int add = prod.GetInventory() - num;
              prod.SetInventory(num);

              //fixes inventory for the Seller
              for(int i = 0 ; i< sList.size(); i++){
                Seller current = sList.get(i);
                for (int j = 0; j < current.ProdList().size(); j++){
                  Product t = current.ProdList().getItem(j);

                  if (prod.Getname().equals(t.Getname()) &&
prod.GetDescription().equals(t.GetDescription()) && prod.GetPrice() == t.GetPrice()){
                    t.SetInventory(t.GetInventory() + add);
                    break;
                  }

                }
              }


              card3.removeAll();
              ShopCart();
```

```
            card3.repaint();
            card3.revalidate();

        }
        else{
            ccurrent.Remove(prod);
            shop.remove(temp);
            shop.revalidate();
            shop.repaint();
        }
      }
    };
    //add Listeners
    update.addActionListener(updateCart);

    //add componenets to the row
    temp.add(name);
    temp.add(d);
    temp.add(inventory);
    temp.add(price);
    temp.add(inbox);
    temp.add(update);
    shop.add(temp);
}


//same thig as other for loop bu thtis adds discount items
for(int i = 0; i < ccurrent.DisList().size(); i++){
    DiscountProduct prod = ccurrent.DisList().getItem(i);

    final JPanel temp = new JPanel();
    temp.setLayout(new BoxLayout(temp, BoxLayout.X_AXIS));

    //Name
    JTextArea name = new JTextArea(prod.Getname(),4, 15);
    Font font = name.getFont();
    float size = font.getSize() + 3.0f;
    name.setFont(font.deriveFont(size + 6.0f));
    name.setMargin(new Insets(2,10,0,10));
    name.setLineWrap(true);
    name.setWrapStyleWord(true);

    //description
```

```
JTextArea description = new JTextArea(prod.GetDescription(),4, 20);
description.setFont(font.deriveFont(size));
description.setLineWrap(true);
description.setWrapStyleWord(true);
description.setMargin(new Insets(0,10,0,10));
JScrollPane d = new JScrollPane(description);
d.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

//inventory
JTextArea inventory = new JTextArea(Integer.toString(prod.GetInventory()),4, 5);
inventory.setFont(font.deriveFont(size));
inventory.setMargin(new Insets(20,30,0,0));

//price
NumberFormat formatter = NumberFormat.getCurrencyInstance();
JTextArea price = new JTextArea(formatter.format(prod.GetPrice()),4, 10);
price.setFont(font.deriveFont(size));
price.setMargin(new Insets(20,30,0,0));

String[] comboxs = new String[prod.GetInventory()+1];

for(int k = 0; k < prod.GetInventory()+1; k++){
   comboxs[k] = Integer.toString(k);
}

//
JComboBox inbox = new JComboBox(comboxs);
inbox.setSelectedIndex(prod.GetInventory());
inbox.setPreferredSize(new Dimension(50,10));


name.setEditable(false);
description.setEditable(false);
inventory.setEditable(false);
price.setEditable(false);

JButton update = new JButton("Update Item");
update.setVisible(true);


ActionListener updateCart = new ActionListener(){
   @Override
   public void actionPerformed(ActionEvent e) {
```

```
            int num = Integer.parseInt((String)inbox.getSelectedItem());

            if (num > 0){
               int add = prod.GetInventory() - num;
               prod.SetInventory(num);

               for(int i = 0 ; i< sList.size(); i++){
                  Seller current = sList.get(i);
                  for (int j = 0; j < current.DisList().size(); j++){
                     DiscountProduct t = current.DisList().getItem(j);

                     if (prod.Getname().equals(t.Getname()) &&
prod.GetDescription().equals(t.GetDescription()) && prod.GetPrice() == t.GetPrice()){
                        t.SetInventory(t.GetInventory() + add);

                        break;
                     }

                  }
               }

               card3.removeAll();
               ShopCart();
               card3.repaint();
               card3.revalidate();
            }
            else{
               ccurrent.Remove(prod);
               shop.remove(temp);
               shop.revalidate();
               shop.repaint();
            }
         }
      };
      //add Listeners
      update.addActionListener(updateCart);

      //add componenets to the row
      temp.add(name);
      temp.add(d);
      temp.add(inventory);
      temp.add(price);
      temp.add(inbox);
```

```
    temp.add(update);
    shop.add(temp);
}

///Smae  thing as other loops but this adds Bundled Items
for(int i = 0; i < ccurrent.BunList().size(); i++){
    BundleProduct prod = ccurrent.BunList().getItem(i);

    final JPanel temp = new JPanel();
    temp.setLayout(new BoxLayout(temp, BoxLayout.X_AXIS));

    //Name
    JTextArea name = new JTextArea(prod.Getname(),4, 15);
    Font font = name.getFont();
    float size = font.getSize() + 3.0f;
    name.setFont(font.deriveFont(size + 6.0f));
    name.setMargin(new Insets(2,10,0,10));
    name.setLineWrap(true);
    name.setWrapStyleWord(true);

    //description
    JTextArea description = new JTextArea(prod.GetDescription(),4, 20);
    description.setFont(font.deriveFont(size));
    description.setLineWrap(true);
    description.setWrapStyleWord(true);
    description.setMargin(new Insets(0,10,0,10));
    JScrollPane d = new JScrollPane(description);
    d.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

    //inventory
    JTextArea inventory = new JTextArea(Integer.toString(prod.GetInventory()),4, 5);
    inventory.setFont(font.deriveFont(size));
    inventory.setMargin(new Insets(20,30,0,0));

    //price
    NumberFormat formatter = NumberFormat.getCurrencyInstance();
    JTextArea price = new JTextArea(formatter.format(prod.GetPrice()),4, 10);
    price.setFont(font.deriveFont(size));
    price.setMargin(new Insets(20,30,0,0));

    String[] comboxs = new String[prod.GetInventory()+1];

    for(int k = 0; k < prod.GetInventory()+1; k++){
```

```
            comboxs[k] = Integer.toString(k);
        }

        JComboBox inbox = new JComboBox(comboxs);
        inbox.setSelectedIndex(prod.GetInventory());
        inbox.setPreferredSize(new Dimension(50,10));


        name.setEditable(false);
        description.setEditable(false);
        inventory.setEditable(false);
        price.setEditable(false);

        JButton update = new JButton("Update Item");
        update.setVisible(true);

        ActionListener updateCart = new ActionListener(){
          @Override
          public void actionPerformed(ActionEvent e) {
            int num = Integer.parseInt((String)inbox.getSelectedItem());

            if (num > 0){
              int add = prod.GetInventory() - num;
              prod.SetInventory(num);

              for(int i = 0 ; i< sList.size(); i++){
                Seller current = sList.get(i);
                for (int j = 0; j < current.BunList().size(); j++){
                  BundleProduct t = current.BunList().getItem(j);

                  if (prod.Getname().equals(t.Getname()) &&
prod.GetDescription().equals(t.GetDescription()) && prod.GetPrice() == t.GetPrice()){
                    t.SetInventory(t.GetInventory() + add);
                    break;
                  }

                }
              }

              card3.removeAll();
              ShopCart();
              card3.repaint();
              card3.revalidate();
```

```
            }
            else{
               ccurrent.Remove(prod);
               shop.remove(temp);
               shop.revalidate();
               shop.repaint();
            }
          }
       };
       //add Listeners
       update.addActionListener(updateCart);

       //add componenets to the row
       temp.add(name);
       temp.add(d);
       temp.add(inventory);
       temp.add(price);
       temp.add(inbox);
       temp.add(update);
       shop.add(temp);
    }


    JScrollPane prodScroll = new JScrollPane(shop);
    prodScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    prodScroll.setPreferredSize(new Dimension(1022,700));

    card3.add(prodScroll);
    frame.pack();

}




//Seller API
public void SellerAPI(){

    //main panel for products
    final JPanel prodPan = new JPanel();
    prodPan.setLayout(new WrapLayout());
```

```
//main panel for discount and Bundle prodcuts
final JPanel disPan = new JPanel();
disPan.setLayout( new WrapLayout());

//greeting
JLabel ProductLabel = new JLabel("View and Edit Products");
Font fontprod = ProductLabel.getFont();
float sizeprod = fontprod.getSize() + 20.0f;
ProductLabel.setFont(fontprod.deriveFont(sizeprod));

JPanel prodLabelPan = new JPanel();
prodLabelPan.setLayout(new BorderLayout());
prodLabelPan.setPreferredSize(new Dimension(1022,55));


prodLabelPan.add(ProductLabel, BorderLayout.SOUTH);
card1.add(prodLabelPan);


//linked list to hold items adn jpanles they are in
final LinkedList<Product> prodtemp = new LinkedList();
final LinkedList<JPanel> prodPanelLst = new LinkedList();

//buttons to add discount and bundle, gets added to the card later on
JButton addDis = new JButton("Discount Selected item");
JButton addBun = new JButton("Bundle Selected Items");
addBun.setVisible(false);

//header panel to show what the Product items are organizes ass
JPanel head = new JPanel();
head.setLayout(new BoxLayout(head, BoxLayout.X_AXIS));
head.setPreferredSize(new Dimension(1022, 50));

//ads textAreas to Panel
for (int i = 0; i < 1; i++){
   JTextArea name = new JTextArea("Name",2, 10);
   Font font = name.getFont();
   float size = font.getSize() + 10.0f;
   name.setFont(font.deriveFont(size));
   name.setMargin(new Insets(10,10,0,0));
   name.setLineWrap(true);
   name.setWrapStyleWord(true);
```

```
    //descreiption
    JTextArea description = new JTextArea("Description",2, 5);
    description.setFont(font.deriveFont(size));
    description.setLineWrap(true);
    description.setWrapStyleWord(true);
    description.setMargin(new Insets(10,10,0,0));
    JScrollPane d = new JScrollPane(description);
    d.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

    //inventory
    JTextArea inventory = new JTextArea("Inventory",2, 10);
    inventory.setFont(font.deriveFont(size));
    inventory.setMargin(new Insets(10,0,0,0));

    JTextArea price = new JTextArea("Price",2, 10);
    price.setFont(font.deriveFont(size));
    price.setMargin(new Insets(10,0,0,0));

    //save and edit buttons
    name.setEditable(false);
    description.setEditable(false);
    inventory.setEditable(false);
    price.setEditable(false);

    head.add(name);
    head.add(description);
    head.add(inventory);
    head.add(price);
}

card1.add(head);


//get all items in current Sellers product list
for (int i = 0 ; i < scurrent.ProdList().size(); i++){

    final Product prod = scurrent.ProdList().getItem(i);

    final JPanel temp = new JPanel();
    temp.setLayout(new BoxLayout(temp, BoxLayout.X_AXIS));


    //name text area
```

```
JTextArea name = new JTextArea(prod.Getname(),4, 15);
Font font = name.getFont();
float size = font.getSize() + 3.0f;
name.setFont(font.deriveFont(size + 6.0f));
name.setMargin(new Insets(2,10,0,10));
name.setLineWrap(true);
name.setWrapStyleWord(true);

//descreiption
JTextArea description = new JTextArea(prod.GetDescription(),4, 20);
description.setFont(font.deriveFont(size));
description.setLineWrap(true);
description.setWrapStyleWord(true);
description.setMargin(new Insets(0,10,0,10));
JScrollPane d = new JScrollPane(description);
d.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

//inventory
JTextArea inventory = new JTextArea(Integer.toString(prod.GetInventory()),4, 10);
inventory.setFont(font.deriveFont(size));
inventory.setMargin(new Insets(20,30,0,0));

//price
NumberFormat formatter = NumberFormat.getCurrencyInstance();
JTextArea price = new JTextArea(formatter.format(prod.GetPrice()),4, 10);
price.setFont(font.deriveFont(size));
price.setMargin(new Insets(20,30,0,0));

//save and edit buttons
name.setEditable(false);
description.setEditable(false);
inventory.setEditable(false);
price.setEditable(false);

// edit/save
JButton edit = new JButton(" Edit ");
JButton save = new JButton("Save");
save.setVisible(false);
JCheckBox box = new JCheckBox();

//set text boxes to editbale
ActionListener editable = new ActionListener(){
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            name.setEditable(true);
            description.setEditable(true);
            inventory.setEditable(true);
            price.setEditable(true);
            edit.setVisible(false);
            save.setVisible(true);
        }
    };

    //sets textareas to uneditable and updates the UI
    ActionListener savable;
    savable = new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
            int flag = 0;
            int inv = 0;
            double value = 0;
            try{
                value = parse(price.getText(), Locale.US).doubleValue();
                inv =parse(inventory.getText(), Locale.US).intValue();
            }
            catch(ParseException j){
                flag = 1;
            }
            //if exception is not triggered then updat product
            if (flag == 0){
                //update inventory
                prod.SetPrice(value);
                price.replaceRange(formatter.format(prod.GetPrice()), 0, price.getText().length());

                //update Price
                prod.SetInventory(inv);
                inventory.replaceRange(Integer.toString(prod.GetInventory()), 0,
inventory.getText().length());

                //update name
                prod.SetName(name.getText());

                //update description
                prod.SetDescription(description.getText());
```

```
            //update
            //refresh screem
            card1.revalidate();
            name.setEditable(false);
            description.setEditable(false);
            inventory.setEditable(false);
            price.setEditable(false);
            edit.setVisible(true);
            save.setVisible(false);
            card1.removeAll();
            SellerAPI();
            card1.repaint();

        }
    }
};


//ceck box action listener
ItemListener morf = new ItemListener(){
    int flag =0;
    @Override
    public void itemStateChanged(ItemEvent e) {
        //if stat is checked add item to temprorary linked list
        if (e.getStateChange() == ItemEvent.SELECTED) {
            prodtemp.add(prod);
            prodPanelLst.add(temp);
            flag = 1;
        }
        //if unchecked remove items from temp linked list
        else if (flag == 1){

            prodtemp.remove(prod);
            prodPanelLst.remove(temp);
            flag = 0;
        }

        //of more than one itme is checked it replaces the Discount
        //button with the bundle button
        if (prodtemp.size()>1){
            addBun.setVisible(true);
            addDis.setVisible(false);
        }
```

```
            //does the opposite
            else{
              addBun.setVisible(false);
              addDis.setVisible(true);
            }
          }
        };

        //add Listeners
        edit.addActionListener(editable);
        save.addActionListener(savable);
        box.addItemListener(morf);

        //add componenets to the row
        temp.add(name);
        temp.add(d);
        temp.add(inventory);
        temp.add(price);
        temp.add(box);
        temp.add(edit);
        temp.add(save);

        //ad row to the product panel
        prodPan.add(temp);
      }


      //set up too add new product to the sellers products and the
API//////////////////////////////////////////
      final JPanel newtemp = new JPanel();
      newtemp.setLayout(new BoxLayout(newtemp, BoxLayout.X_AXIS));

      for (int i = 0; i< 1;i++){
        //name text area
        JTextArea name = new JTextArea("",2, 16);
        Font font = name.getFont();
        float size = font.getSize() + 3.0f;
        name.setFont(font.deriveFont(size + 6.0f));
        name.setMargin(new Insets(2,5,0,10));
        name.setLineWrap(true);
        name.setWrapStyleWord(true);

        //descreiption
```

```
JTextArea description = new JTextArea("",2, 21);
description.setFont(font.deriveFont(size));
description.setLineWrap(true);
description.setWrapStyleWord(true);
description.setMargin(new Insets(2,10,0,10));
JScrollPane d = new JScrollPane(description);

//inventory
JTextArea inventory = new JTextArea("",2, 10);
inventory.setFont(font.deriveFont(size));
inventory.setMargin(new Insets(2,30,0,0));

//price
NumberFormat formatter = NumberFormat.getCurrencyInstance();
JTextArea price = new JTextArea("",2, 10);
price.setFont(font.deriveFont(size));
price.setMargin(new Insets(2,30,0,0));

//save and edit buttons
name.setEditable(true);
description.setEditable(true);
inventory.setEditable(true);
price.setEditable(true);

// button to add new item to products list
JButton save = new JButton("Add Product");

//saves product listener adds it to the API and the PDList****
ActionListener savable;
savable = new ActionListener(){
  @Override
  public void actionPerformed(ActionEvent e) {
    int flag = 0;
    int inv = 0;
    double value = 0;
    try{
      value = parse(price.getText(), Locale.US).doubleValue();
      inv =parse(inventory.getText(), Locale.US).intValue();
    }
    catch(ParseException j){
      flag = 1;
    }
    //if exception is not triggered then updat product
```

```
        if (flag == 0){
          //update inventory
          Product newP = new Product();

          newP.SetPrice(value);
          price.setText(null);

          //update Price
          newP.SetInventory(inv);
          inventory.setText(null);

          //update name
          newP.SetName(name.getText());
          name.setText(null);

          //update description
          newP.SetDescription(description.getText());
          description.setText(null);
          scurrent.ProdList().add(newP);

          //refresh screem
          card1.removeAll();
          SellerAPI();
          card1.repaint();
          card1.revalidate();
        }
      }
    };
    save.addActionListener(savable);
    newtemp.add(name);
    newtemp.add(d);
    newtemp.add(inventory);
    newtemp.add(price);
    newtemp.add(save);
}//End save new API////////////////////////////////////////////////


//set up to add new Discount/bundle item
JTextArea setprice = new JTextArea("Set Price:",1, 5);
Font fontt = setprice.getFont();
float sizet = fontt.getSize() + 5.0f;
setprice.setEditable(false);
setprice.setFont(fontt.deriveFont(sizet));
```

```
        setprice.setMargin(new Insets(2,5,2,0));

        JTextArea numprice = new JTextArea("",1, 55);
        numprice.setFont(fontt.deriveFont(sizet));
        numprice.setMargin(new Insets(2,5,2,5));

        //deletes products
selected////////////////////////////////////////////////////////////
        JPanel prodButtons = new JPanel();
        prodButtons.setPreferredSize(new Dimension(1000,27));
        prodButtons.setLayout(new BoxLayout(prodButtons, BoxLayout.X_AXIS));
        JButton delProd = new JButton("Delete Selected Item(s)");
        delProd.addActionListener(new ActionListener(){
          @Override
          public void actionPerformed(ActionEvent e) {

            for(int i = 0; i < prodtemp.size(); i++){
              scurrent.ProdList().remove(prodtemp.get(i));
            }

            prodtemp.clear();
            //refresh screem
            card1.removeAll();
            SellerAPI();
            card1.repaint();
            card1.revalidate();
          }
        });////////////////////////////////////////////////////////////////////

        ///add action listeners to discount and bundle
buttons>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
        addDis.addActionListener(new ActionListener(){
          @Override
          public void actionPerformed(ActionEvent e) {
            if (prodtemp.size() == 1 && numprice.getText() != ""){
              int f = 0;
              double value = 0;
              try{
                value = parse(numprice.getText(), Locale.US).doubleValue();
              }
              catch(ParseException j){
                f = 1;
              }
```

```
            if (f != 1 && value > 0){
                DiscountProduct dis = new DiscountProduct(prodtemp.get(0), value);

                scurrent.Add(dis);

                card1.removeAll();
                SellerAPI();
                card1.repaint();
                card1.revalidate();
            }

        }
    }

});

//////>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>

    //action listner to add Bundled product
    addBun.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
            if (prodtemp.size() > 1){
                int f = 0;
                double value = 0;
                try{
                    value = parse(numprice.getText(), Locale.US).doubleValue();
                }
                catch(ParseException j){

                }
                if (f != 1){
                    BundleProduct bun = new BundleProduct();
                    for (int i = 0; i < prodtemp.size(); i++){
                        bun.AddProd(prodtemp.get(i));
                        System.out.println(prodtemp.size());
                        System.out.println(i);
                        System.out.println(prodtemp.get(i).Getname());
                    }
                    if (value >0){
                        bun.SetPrice(value);
                    }
```

```
            scurrent.Add(bun);

            //refresh screen
            card1.removeAll();
            SellerAPI();
            card1.repaint();
            card1.revalidate();
          }

        }
      }

  });


  //add panels to card*********************************************************
  JScrollPane prodScroll = new JScrollPane(prodPan);
  prodScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
  prodScroll.setPreferredSize(new Dimension(1022,300));
  card1.add(prodScroll);
  card1.add(newtemp);
  prodButtons.add(delProd);
  prodButtons.add(addDis);
  prodButtons.add(addBun);
  prodButtons.add(setprice);
  prodButtons.add(numprice);
  card1.add(prodButtons);
  //////////*************************************************************************

  //label for the budle product
  JLabel distLabel = new JLabel("View and Edit Discount/Bundle Products");
  distLabel.setFont(fontprod.deriveFont(sizeprod));
  JPanel disLabelPan = new JPanel();
  disLabelPan.setPreferredSize(new Dimension(1022,80));
  disLabelPan.setLayout(new BorderLayout());
  disLabelPan.add(distLabel, BorderLayout.SOUTH);
  card1.add(disLabelPan);

//API that shows Bundle/discount products_____________________________________
  for (int i = 0; i < scurrent.DisList().size(); i ++){

    //current product
```

```
final DiscountProduct prod = scurrent.DisList().getItem(i);

final JPanel temp = new JPanel();
temp.setLayout(new BoxLayout(temp, BoxLayout.X_AXIS));

//name text area
JTextArea name = new JTextArea(prod.Getname(),4, 15);
Font font = name.getFont();
float size = font.getSize() + 3.0f;
name.setFont(font.deriveFont(size + 6.0f));
name.setMargin(new Insets(2,10,0,10));
name.setLineWrap(true);
name.setWrapStyleWord(true);

//descreiption
JTextArea description = new JTextArea(prod.GetDescription(),4, 20);
description.setFont(font.deriveFont(size));
description.setLineWrap(true);
description.setWrapStyleWord(true);
description.setMargin(new Insets(0,10,0,10));
JScrollPane d = new JScrollPane(description);
d.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

//inventory
JTextArea inventory = new JTextArea(Integer.toString(prod.GetInventory()),4, 10);
inventory.setFont(font.deriveFont(size));
inventory.setMargin(new Insets(20,30,0,0));

//price
NumberFormat formatter = NumberFormat.getCurrencyInstance();
JTextArea price = new JTextArea(formatter.format(prod.GetPrice()),4, 10);
price.setFont(font.deriveFont(size));
price.setMargin(new Insets(20,30,0,0));

//save and edit buttons
name.setEditable(false);
description.setEditable(false);
inventory.setEditable(false);
price.setEditable(false);

// deletes procuts
JButton delete = new JButton("Delete Product");
```

```
    //acction listener to delete button
    ActionListener dele;
    dele = new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
            scurrent.DisList().remove(prod);
            //refresh screem
            card1.removeAll();
            SellerAPI();
            card1.repaint();
            card1.revalidate();
        }
    };
    delete.addActionListener(dele);

    //add componenets to the row
    temp.add(name);
    temp.add(d);
    temp.add(inventory);
    temp.add(price);
    temp.add(delete);


    //ad row to the product panel
    disPan.add(temp);
}


//API that shows Bundle products_____________________________________________
for (int i = 0; i < scurrent.BunList().size(); i ++){

    //current product
    final BundleProduct prod = scurrent.BunList().getItem(i);

    final JPanel temp = new JPanel();
    temp.setLayout(new BoxLayout(temp, BoxLayout.X_AXIS));

    //name text area
    JTextArea name = new JTextArea(prod.Getname(),4, 15);
    Font font = name.getFont();
    float size = font.getSize() + 3.0f;
    name.setFont(font.deriveFont(size + 6.0f));
    name.setMargin(new Insets(2,10,0,10));
```

```
name.setLineWrap(true);
name.setWrapStyleWord(true);

//descreiption
JTextArea description = new JTextArea(prod.GetDescription(),4, 20);
description.setFont(font.deriveFont(size));
description.setLineWrap(true);
description.setWrapStyleWord(true);
description.setMargin(new Insets(0,10,0,10));
JScrollPane d = new JScrollPane(description);
d.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

//inventory
JTextArea inventory = new JTextArea(Integer.toString(prod.GetInventory()),4, 10);
inventory.setFont(font.deriveFont(size));
inventory.setMargin(new Insets(20,30,0,0));

//price
NumberFormat formatter = NumberFormat.getCurrencyInstance();
JTextArea price = new JTextArea(formatter.format(prod.GetPrice()),4, 10);
price.setFont(font.deriveFont(size));
price.setMargin(new Insets(20,30,0,0));

//save and edit buttons
name.setEditable(false);
description.setEditable(false);
inventory.setEditable(false);
price.setEditable(false);

// delets product
JButton delete = new JButton("Delete Product");

//delete action listener to delet BundleProdcut
ActionListener savable;
savable = new ActionListener(){
  @Override
  public void actionPerformed(ActionEvent e) {
    scurrent.BunList().remove(prod);
    //refresh screem
    card1.removeAll();
    SellerAPI();
    card1.repaint();
    card1.revalidate();
```

```
        }
    };
    delete.addActionListener(savable);

    //add componenets to the row
    temp.add(name);
    temp.add(d);
    temp.add(inventory);
    temp.add(price);
    temp.add(delete);
    //ad row to the product panel
    disPan.add(temp);
}

JScrollPane disScroll = new JScrollPane(disPan);
disScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
disScroll.setPreferredSize(new Dimension(1025,300));
card1.add(disScroll);


///gets total stats
int prod = 0;
int inv = 0;
double total = 0;
for(int i = 0; i < scurrent.ProdList().size(); i++){

    Product t = scurrent.ProdList().getItem(i);
    total = total + t.GetPrice() * t.GetInventory();
    inv = inv + t.GetInventory();
    prod++;
}
for(int i = 0; i < scurrent.DisList().size(); i++){
    DiscountProduct t = scurrent.DisList().getItem(i);
    total = total + t.GetPrice() * t.GetInventory();
    inv = inv + t.GetInventory();
    prod++;
}
for(int i = 0; i < scurrent.BunList().size(); i++){
    BundleProduct t = scurrent.BunList().getItem(i);
    total = total + t.GetPrice() * t.GetInventory();
    inv = inv + t.GetInventory();
    prod++;
}
```

```
NumberFormat formatter = NumberFormat.getCurrencyInstance();

String price = formatter.format(total);

//display total stats
JPanel totalpan = new JPanel();
totalpan.setLayout(new BoxLayout(totalpan, BoxLayout.X_AXIS));

JTextArea tot= new JTextArea("Total Price:",1, 5);
Font font = tot.getFont();
float size = font.getSize() + 4.0f;
tot.setFont(font.deriveFont(size + 5.0f));
tot.setMargin(new Insets(10,5,5,0));
tot.setEditable(false);

JTextArea totaltext = new JTextArea(price,1,10);
totaltext.setEditable(false);
totaltext.setFont(font.deriveFont(size));
totaltext.setMargin(new Insets(14,5,5,0));



JTextArea totinv= new JTextArea("Total Inventory:",1, 6);
totinv.setFont(font.deriveFont(size + 5.0f));
totinv.setMargin(new Insets(10,5,5,0));
totinv.setEditable(false);

JTextArea totinvtext = new JTextArea(Integer.toString(inv),1,10);
totinvtext.setEditable(false);
totinvtext.setFont(font.deriveFont(size));
totinvtext.setMargin(new Insets(14,5,5,0));



JTextArea totprod= new JTextArea("Total Products:",1, 6);
totprod.setFont(font.deriveFont(size + 5.0f));
totprod.setMargin(new Insets(10,5,5,0));
totprod.setEditable(false);

JTextArea totprodtext = new JTextArea(Integer.toString(prod),1,10);
totprodtext.setEditable(false);
totprodtext.setFont(font.deriveFont(size));
totprodtext.setMargin(new Insets(14,5,5,0));
```

```
    totalpan.add(totprod);
    totalpan.add(totprodtext);
    totalpan.add(totinv);
    totalpan.add(totinvtext);
    totalpan.add(tot);
    totalpan.add(totaltext);

    card1.add(totalpan);
    frame.pack();
}


//Checkout API
public void Checkout(){
    //main panel to hold everything
    final JPanel main = new JPanel();
    main.setLayout(new BoxLayout(main, BoxLayout.Y_AXIS));


    //shows up when payment goes throguh
    JLabel success = new JLabel("Purchase Successfull");
    Font font1 = success.getFont();
    float size1 = font1.getSize() + 10.0f;
    success.setFont(font1.deriveFont(size1));
    if (flag2 == 0){
        success.setVisible(false);
    }
    else{
        success.setVisible(true);
    }
    main.add(success);

    //panel to hold button to go back to shopping cart
    JPanel prodLabelPan = new JPanel();
    prodLabelPan.setLayout(new BorderLayout());

    //allows user to go back to shopppng cart
    //refreshes shopping cart
    JButton back = new JButton("Go Back to Shopping Cart");
    back.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e) {
```

```
        card3.removeAll();
        ShopCart();
        card3.repaint();
        card3.revalidate();
        CardLayout cl = (CardLayout)(cards.getLayout());
        cl.show(cards, "Cart");
    }

});
prodLabelPan.add(back, BorderLayout.WEST);
main.add(prodLabelPan);


///Name Jpanel
JPanel name = new JPanel();
name.setLayout(new BoxLayout(name, BoxLayout.X_AXIS));

JTextArea username = new JTextArea("            Name:", 1, 6);
username.setEditable(false);
Font font = username.getFont();
float size = font.getSize() + 10.0f;
username.setFont(font.deriveFont(size));
username.setMargin(new Insets(10,5,5,0));

JTextArea userText = new JTextArea("",1, 20);
userText.setEditable(true);
userText.setFont(font.deriveFont(size));
userText.setMargin(new Insets(10,0,5,0));

name.add(username);
name.add(userText);


main.add(name);


//card jpanel
JPanel cardnum = new JPanel();
cardnum.setLayout(new BoxLayout(cardnum, BoxLayout.X_AXIS));

JTextArea card= new JTextArea("Card Number:",1, 6);
card.setFont(font.deriveFont(size));
card.setMargin(new Insets(10,5,5,0));
```

```
card.setEditable(false);

JTextArea cardtext = new JTextArea("",1, 20);
cardtext.setEditable(true);
cardtext.setFont(font.deriveFont(size));
cardtext.setMargin(new Insets(10,5,5,0));

cardnum.add(card);
cardnum.add(cardtext);

main.add(cardnum);


///Address panel
JPanel address = new JPanel();
address.setLayout(new BoxLayout(address, BoxLayout.X_AXIS));

JTextArea addr= new JTextArea("       Address:",1, 6);
addr.setFont(font.deriveFont(size));
addr.setMargin(new Insets(10,5,5,0));
addr.setEditable(false);

JTextArea addrtext = new JTextArea("",1, 20);
addrtext.setEditable(true);
addrtext.setFont(font.deriveFont(size));
addrtext.setMargin(new Insets(10,5,5,0));

address.add(addr);
address.add(addrtext);

main.add(address);

 //gets total amount in shopping cart
double total = 0;
for(int i = 0; i < ccurrent.ProdList().size(); i++){

   Product t = ccurrent.ProdList().getItem(i);
   total = total + t.GetPrice() * t.GetInventory();
}
for(int i = 0; i < ccurrent.DisList().size(); i++){
   DiscountProduct t = ccurrent.DisList().getItem(i);
   total = total + t.GetPrice() * t.GetInventory();
}
```

```
    for(int i = 0; i < ccurrent.BunList().size(); i++){
       BundleProduct t = ccurrent.BunList().getItem(i);
       total = total + t.GetPrice() * t.GetInventory();
    }
    NumberFormat formatter = NumberFormat.getCurrencyInstance();

    String price = formatter.format(total);


    JPanel totalpan = new JPanel();
    totalpan.setLayout(new BoxLayout(totalpan, BoxLayout.X_AXIS));

    JTextArea tot= new JTextArea("            Total:",1, 6);
    tot.setFont(font.deriveFont(size));
    tot.setMargin(new Insets(10,5,5,0));
    tot.setEditable(false);

    JTextArea totaltext = new JTextArea(price,1, 20);
    totaltext.setEditable(false);
    totaltext.setFont(font.deriveFont(size));
    totaltext.setMargin(new Insets(10,5,5,0));

    totalpan.add(tot);
    totalpan.add(totaltext);
    main.add(totalpan);

    //button to confirm puchase
    JButton confirm = new JButton("Confirm Purchase");
    confirm.addActionListener(new ActionListener(){
       @Override
       public void actionPerformed(ActionEvent e) {
          // if all fields have text then allow button press
          if(!addrtext.getText().equals("") && !cardtext.getText().equals("") &&
!userText.getText().equals("")){
             //clears shoping cart info
             ccurrent.ProdList().removeAll();
             ccurrent.BunList().removeAll();
             ccurrent.DisList().removeAll();
             System.out.println("Cleared");

             //refresh view
             card5.removeAll();
             //allows the payment successfull panel to show
```

```
                flag2 = 1;
                Checkout();
                card5.repaint();
                card5.revalidate();
            }

        }

    });
    main.add(confirm);


    card5.add(main);
    frame.pack();
  }

  public static void main (String args[]) throws ClassNotFoundException{
    //calls system and starts API
    SystemAPI n = new SystemAPI();
  }

}
```

Unit tests in zipfile