**Project 2**

**1. Rewrite two common use cases for each user story. Sketch the screens the system should display in each use case.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Home main screen:  Store Management  System    Manage New  Purchase  Manage New  Customer  Manage New  Product | Load/Save Product:  Save  Product  Load Product   |  |  | | --- | --- | | ProductID |  | | Name |  | | Price |  | | Quantity |  | |
| Load/Save Customer:  Save  Customer  Load Customer   |  |  | | --- | --- | | CustomerID |  | | Name |  | | Phone |  | | Address |  | | Load/Save Customer:  Save  Purchase  Load Purchase   |  |  | | --- | --- | | PurchaseID |  | | CustomerID |  | | ProductID |  | | Qauntity |  | |

**2. Redesign the Data Access layer at the client side that can load/save data to a remote server component.**

If we create an instance with the source code, StoreManager(“Networt”, DB\_FILE), then an instance of NetworkDataAdapter will be created. Client can load and save data for products, customers, and purchases remotely.

**3. Design the server component to perform load/save requests from the Data Access layer at the client side. Describe the protocol for two sides: client and server.**

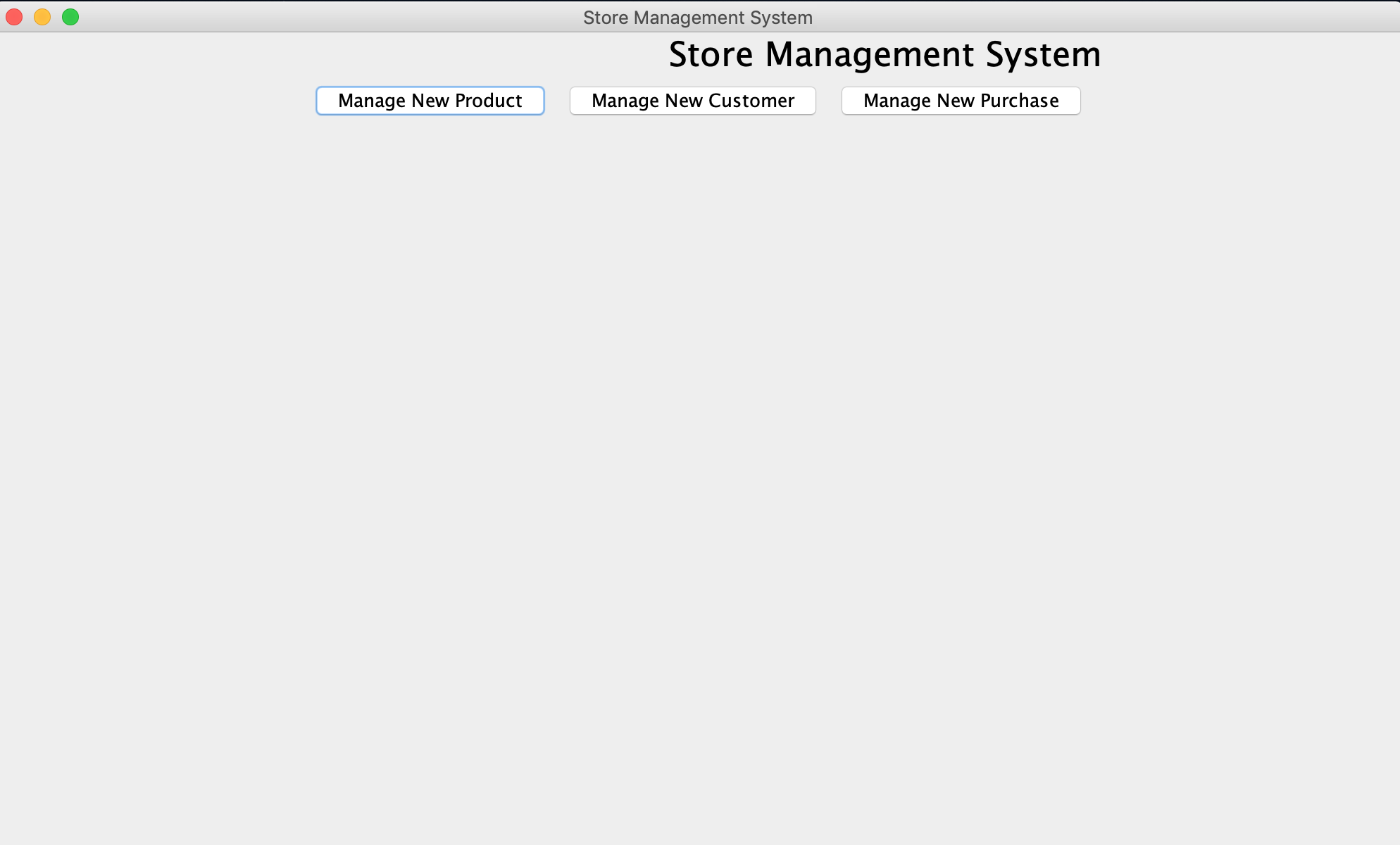
* public interface IDataAdapter {  
    
   public static final int *CONNECTION\_OPEN\_OK* = 100;  
   public static final int *CONNECTION\_OPEN\_FAILED* = 101;  
    
   public static final int *CONNECTION\_CLOSE\_OK* = 200;  
   public static final int *CONNECTION\_CLOSE\_FAILED* = 201;  
    
   public static final int *PRODUCT\_SAVE\_OK* = 0;  
   public static final int *PRODUCT\_DUPLICATE\_ERROR* = 1;  
   public static final int *PRODUCT\_SAVE\_FAILED* = 2;  
    
   public static final int *CUSTOMER\_SAVED\_OK* = 300;  
   public static final int *CUSTOMER\_DUPLICATE\_ERROR* = 301;  
    
    
   public static final int *PURCHASE\_SAVED\_OK* = 500;  
   public static final int *PURCHASE\_DUPLICATE\_ERROR* = 501;  
    
   public int connect(String dbfile);  
   public int disconnect();  
    
   public ProductModel loadProduct(int id);  
   public int saveProduct(ProductModel model);  
    
   public CustomerModel loadCustomer(int id);  
   public int saveCustomer(CustomerModel model);  
    
   public PurchaseModel loadPurchase(int id);  
   public int savePurchase(PurchaseModel model);  
  }

public interface INetworkAdapter {  
 public String send(String msg, String host, int port) throws Exception;  
  
 public MessageModel send(MessageModel msg, String host, int port) throws Exception;  
}

**4. Implement the system.**

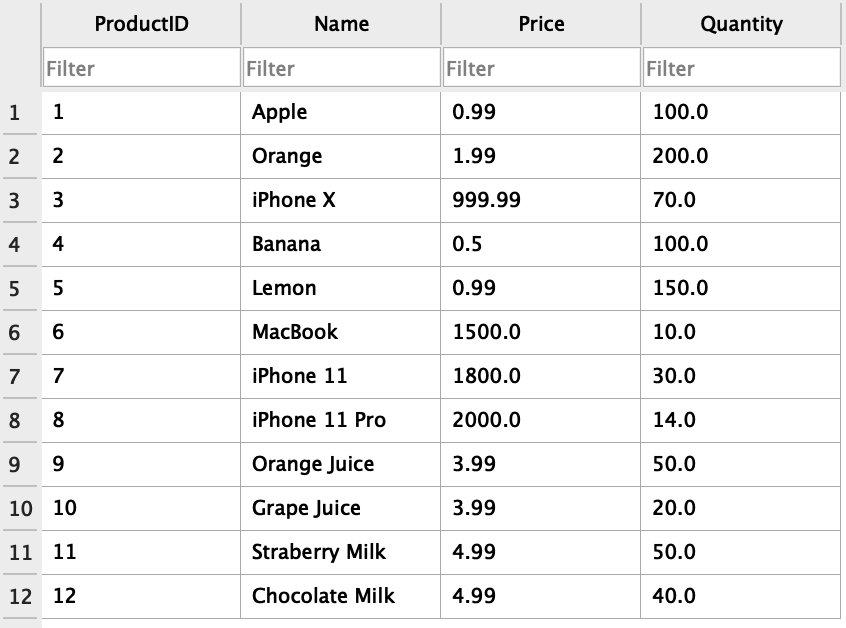
All java files are attached together.

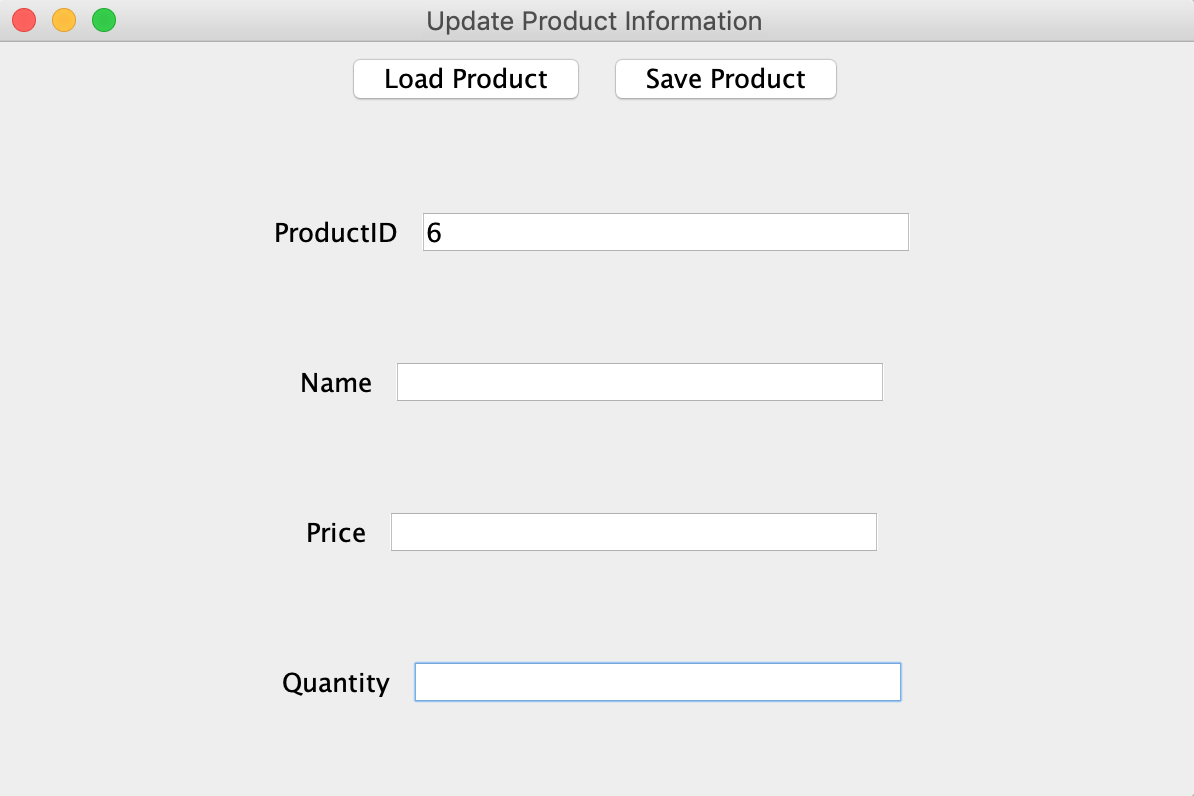
**5. Test the system with each use case.**

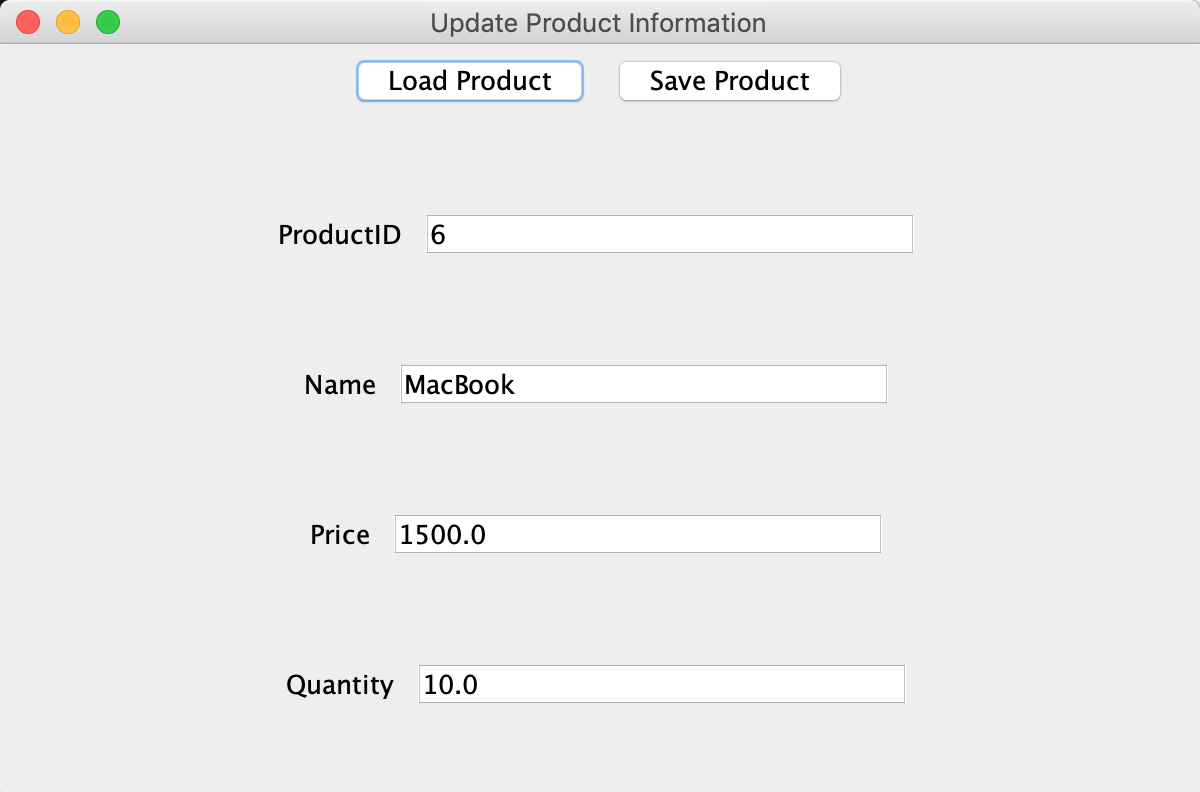
****

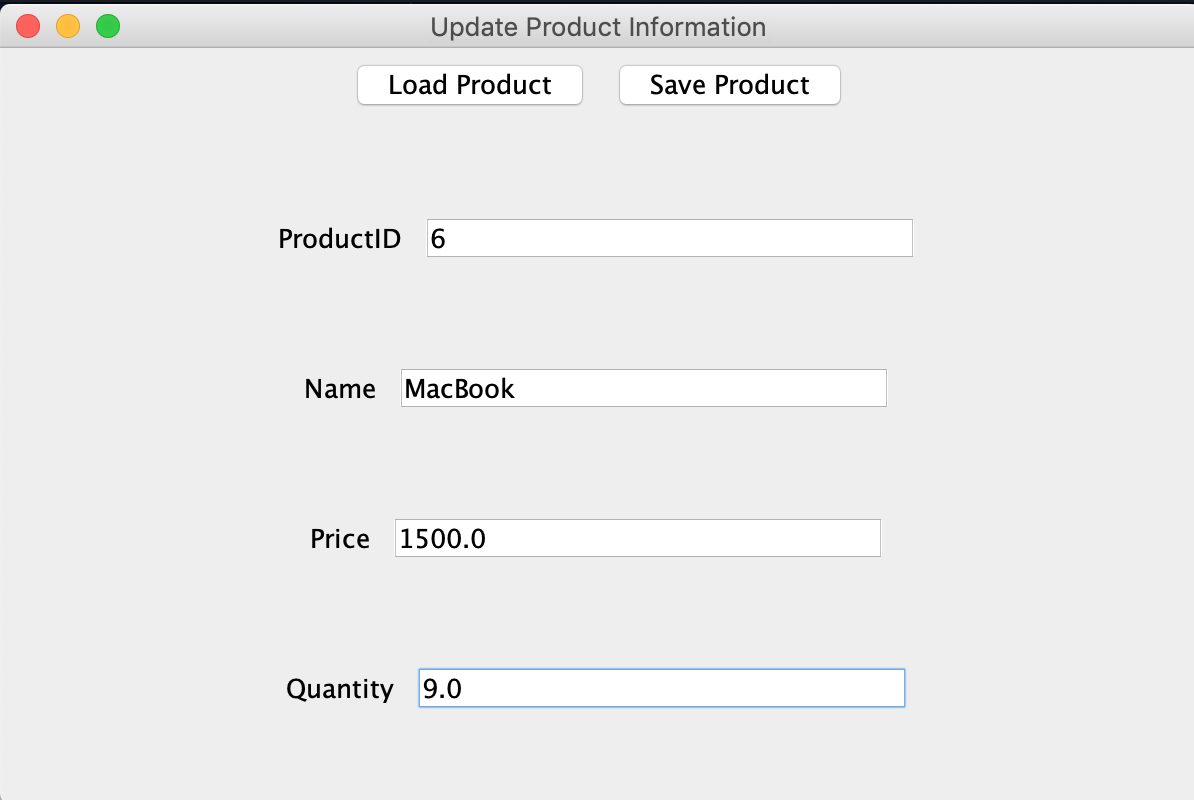
<Common cases>

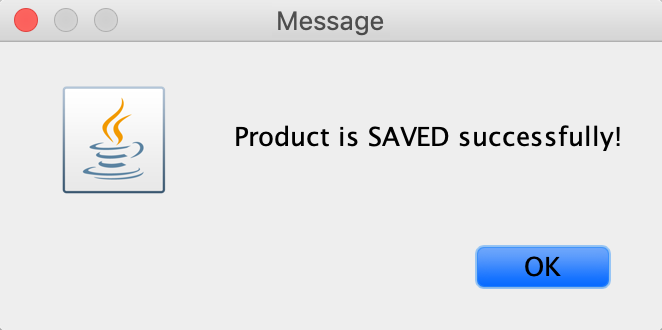
* Products





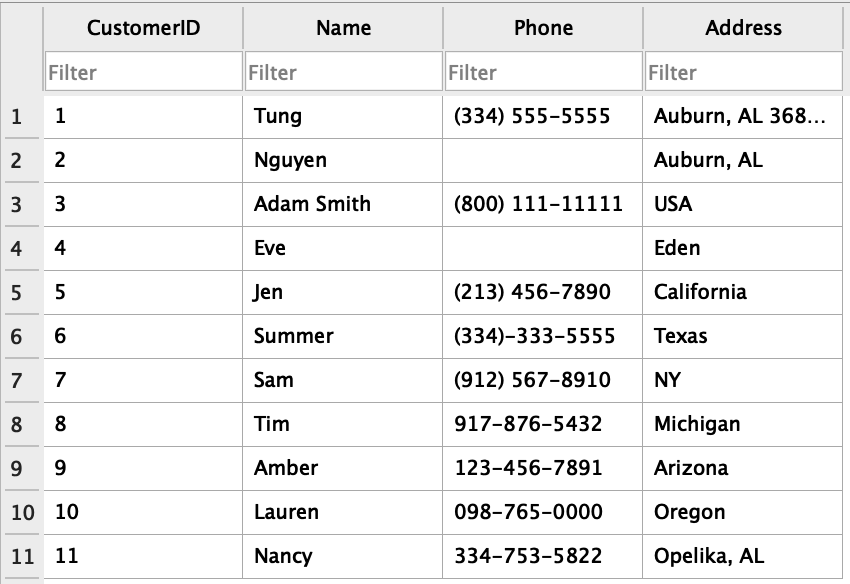


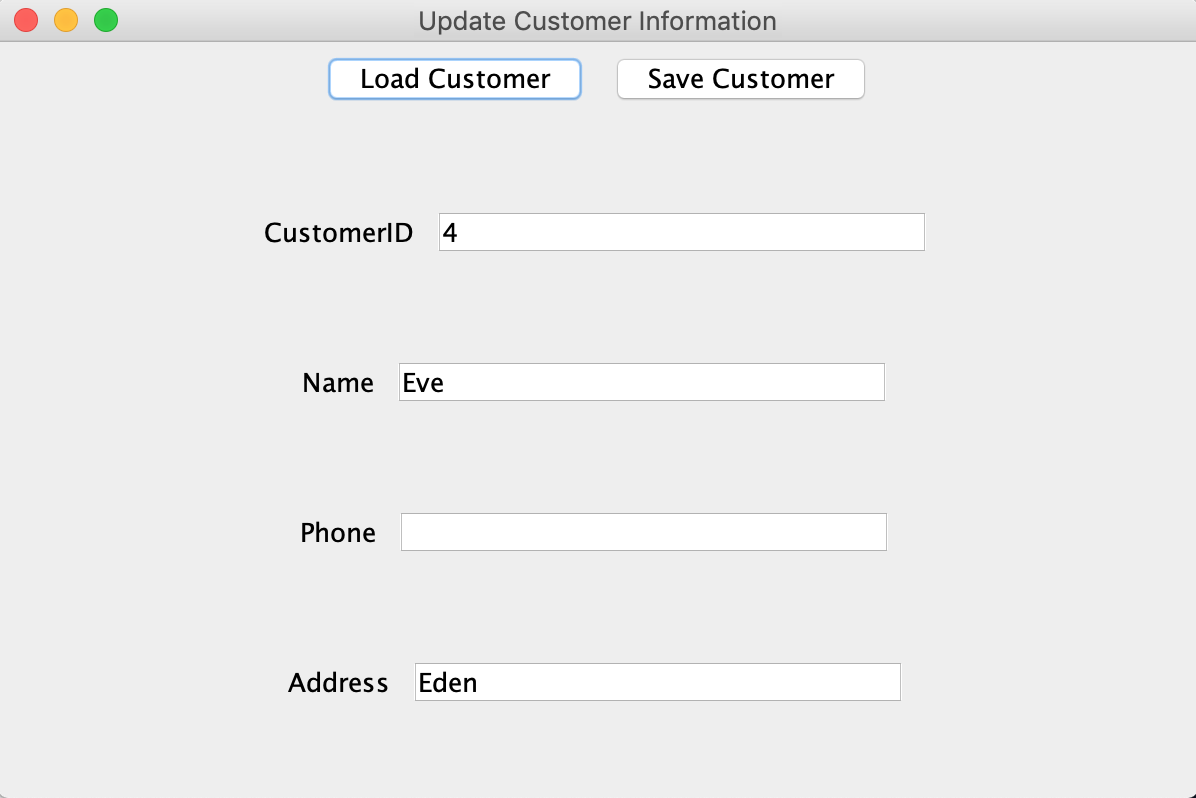


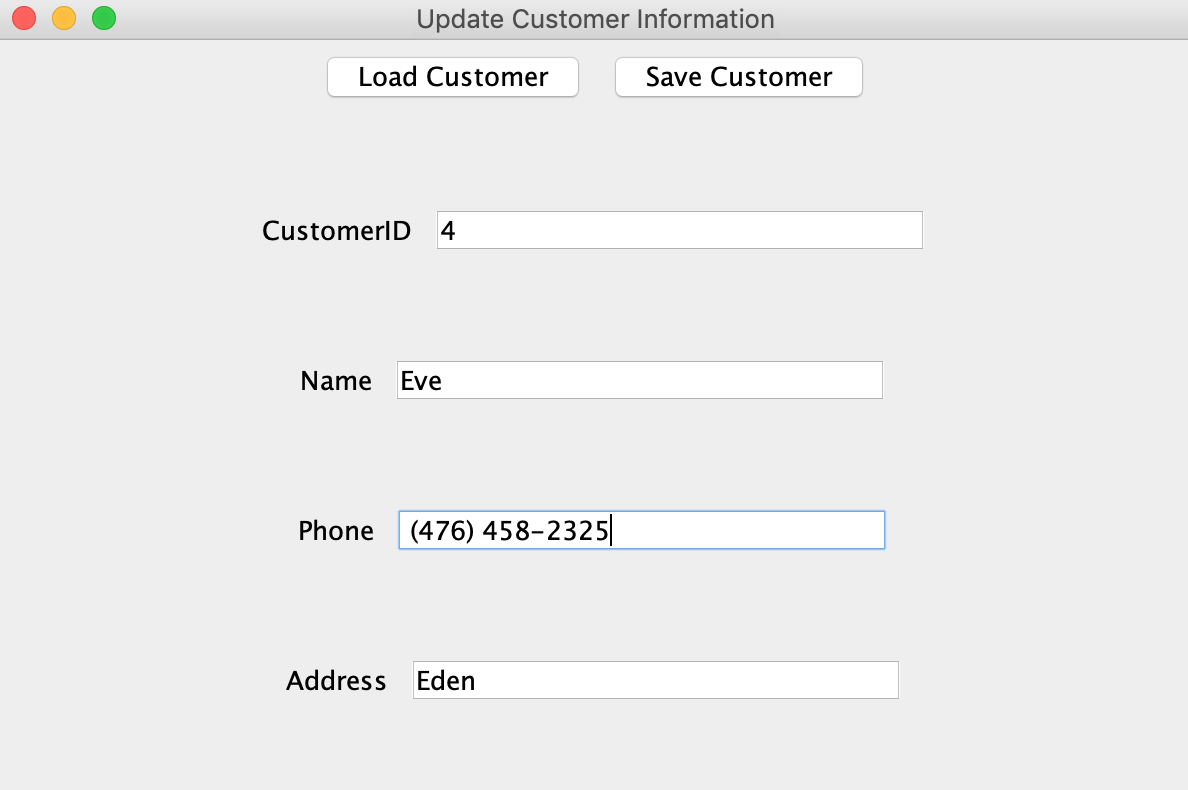


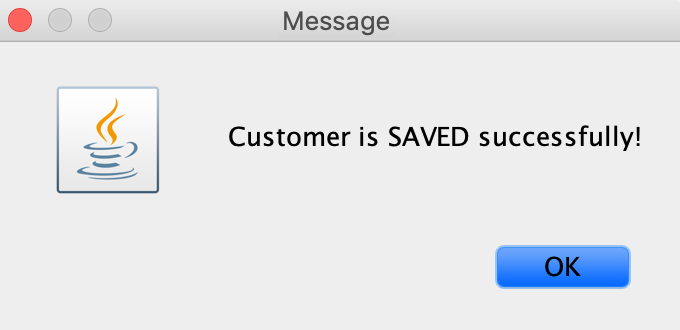
Screen%20Shot%202019-11-20%20at%203.53.24%20PM.png

* Customers



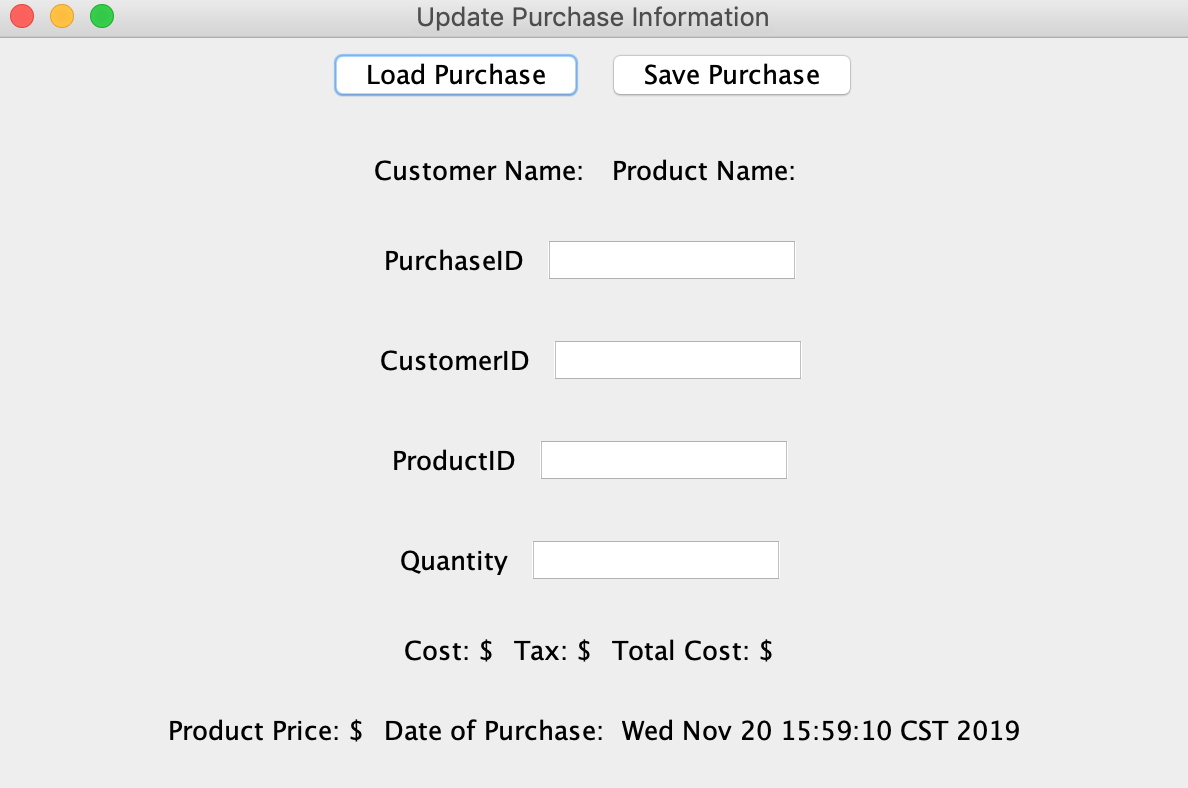


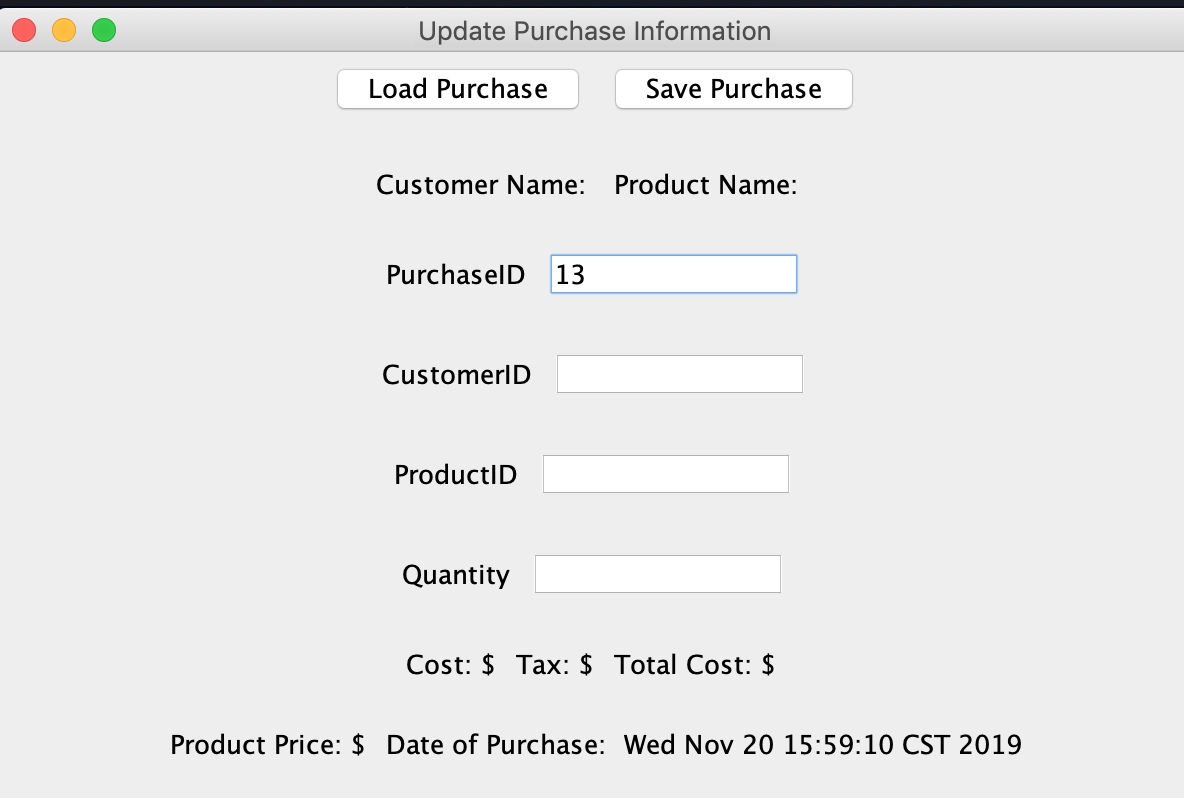


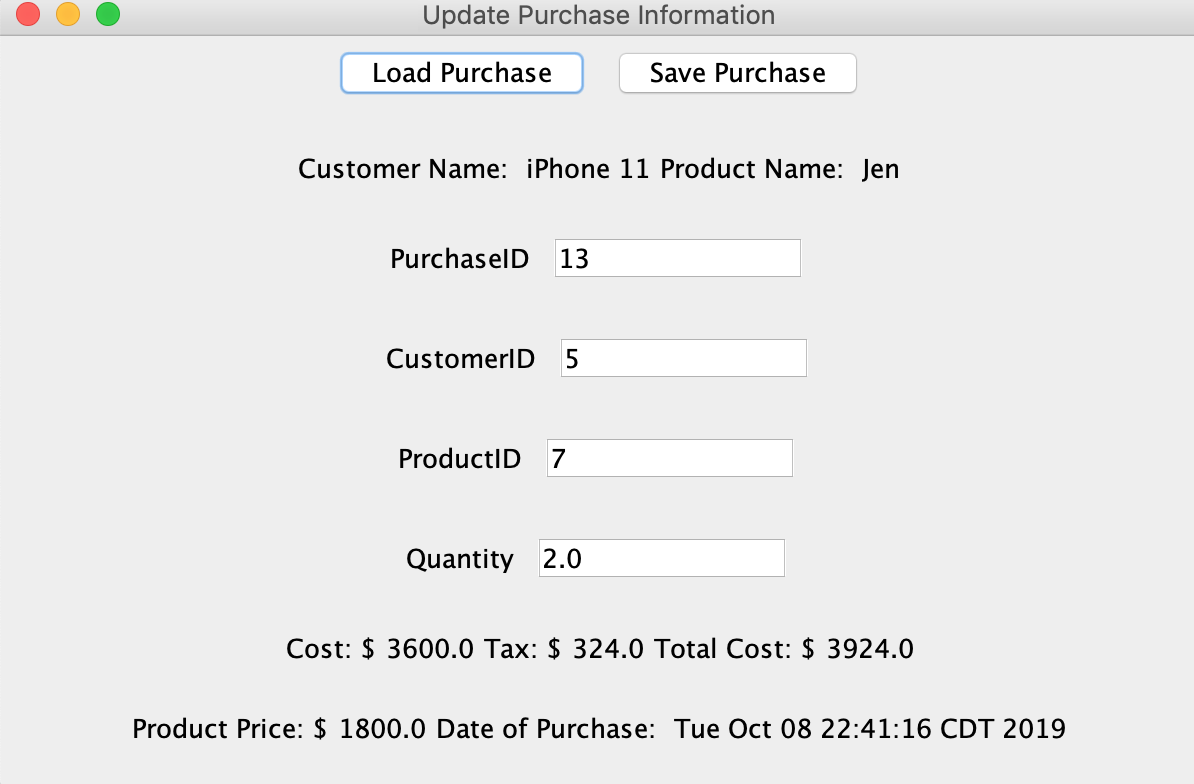


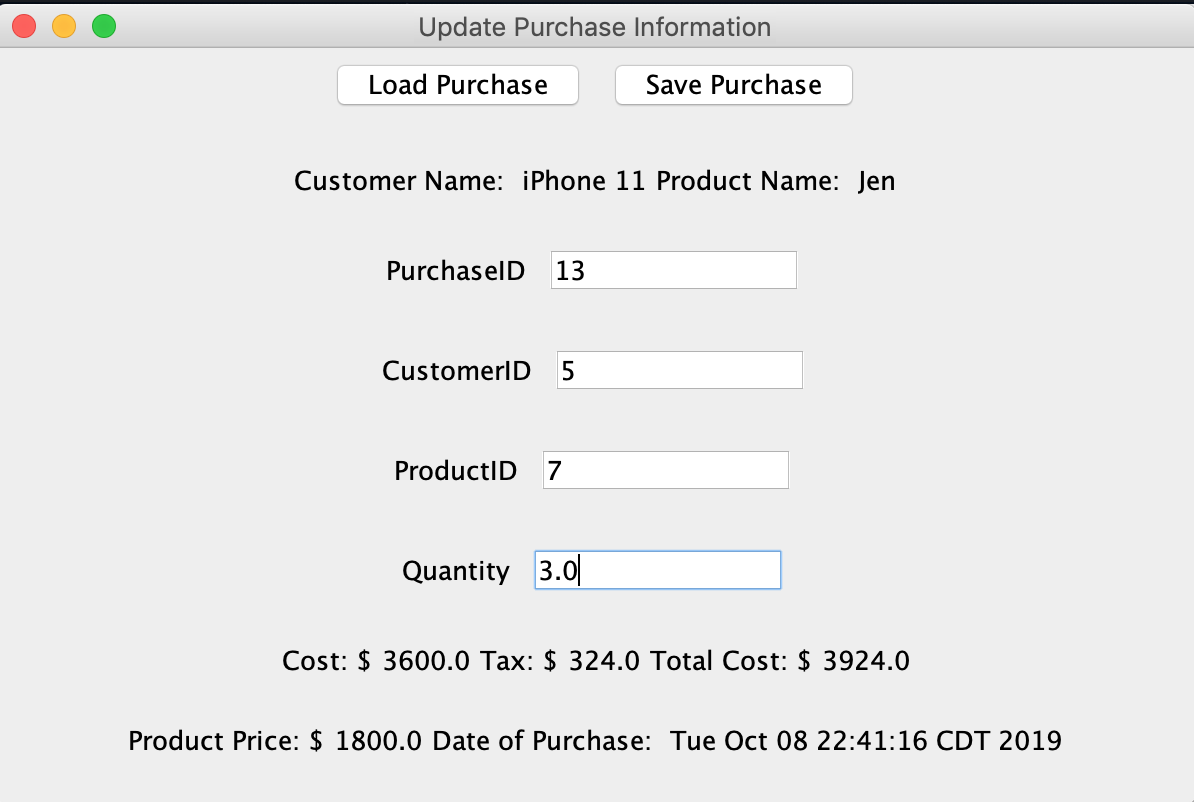
Screen%20Shot%202019-11-20%20at%203.57.47%20PM.png

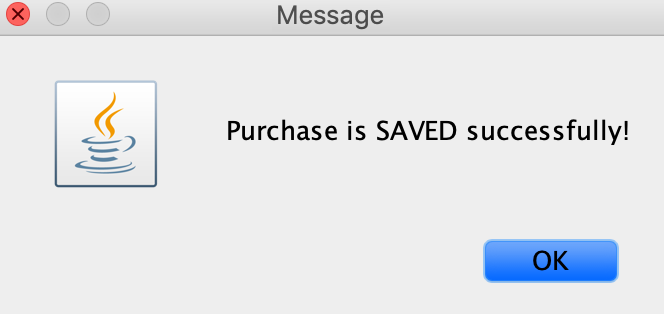
* Purchases







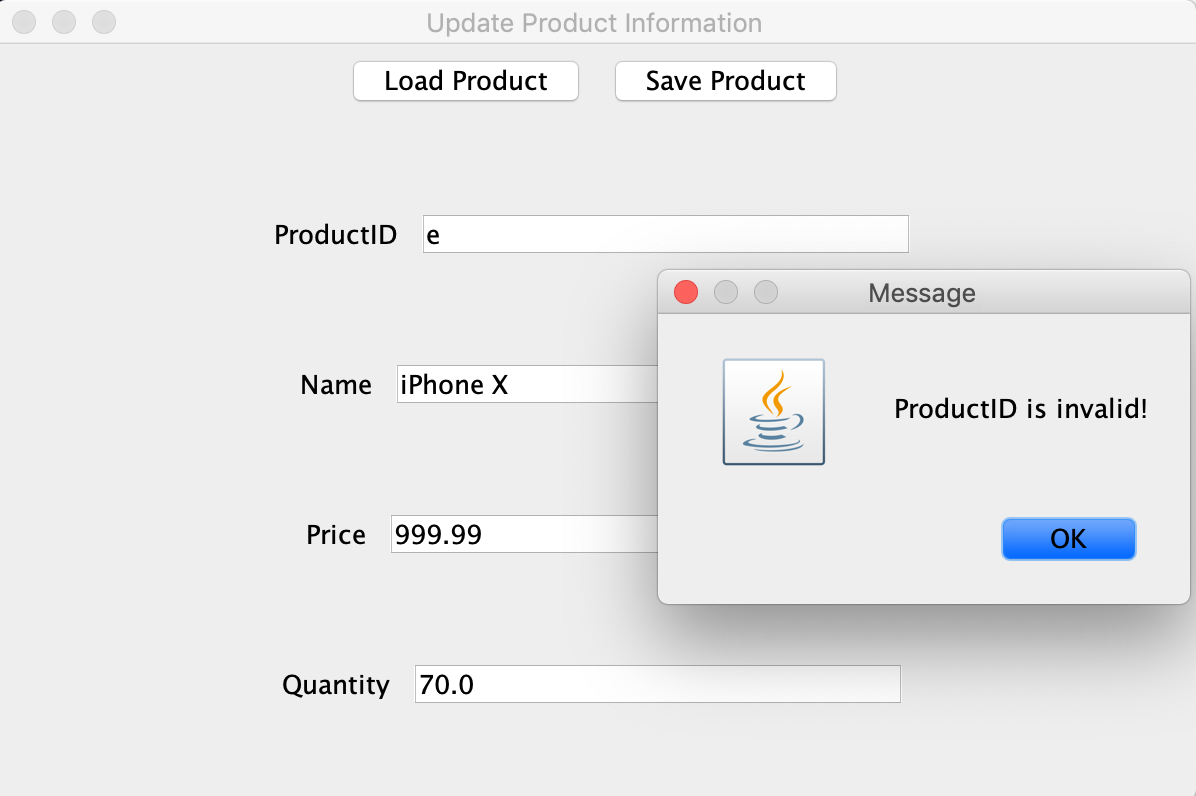


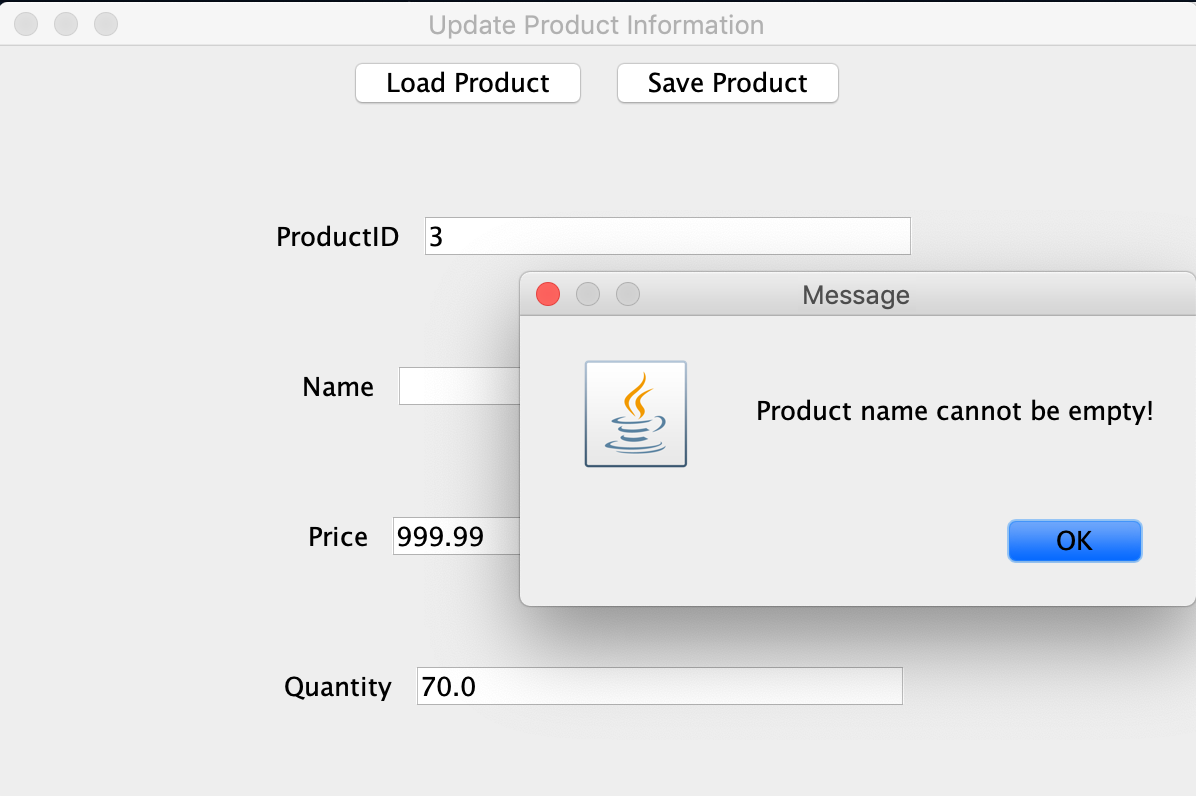


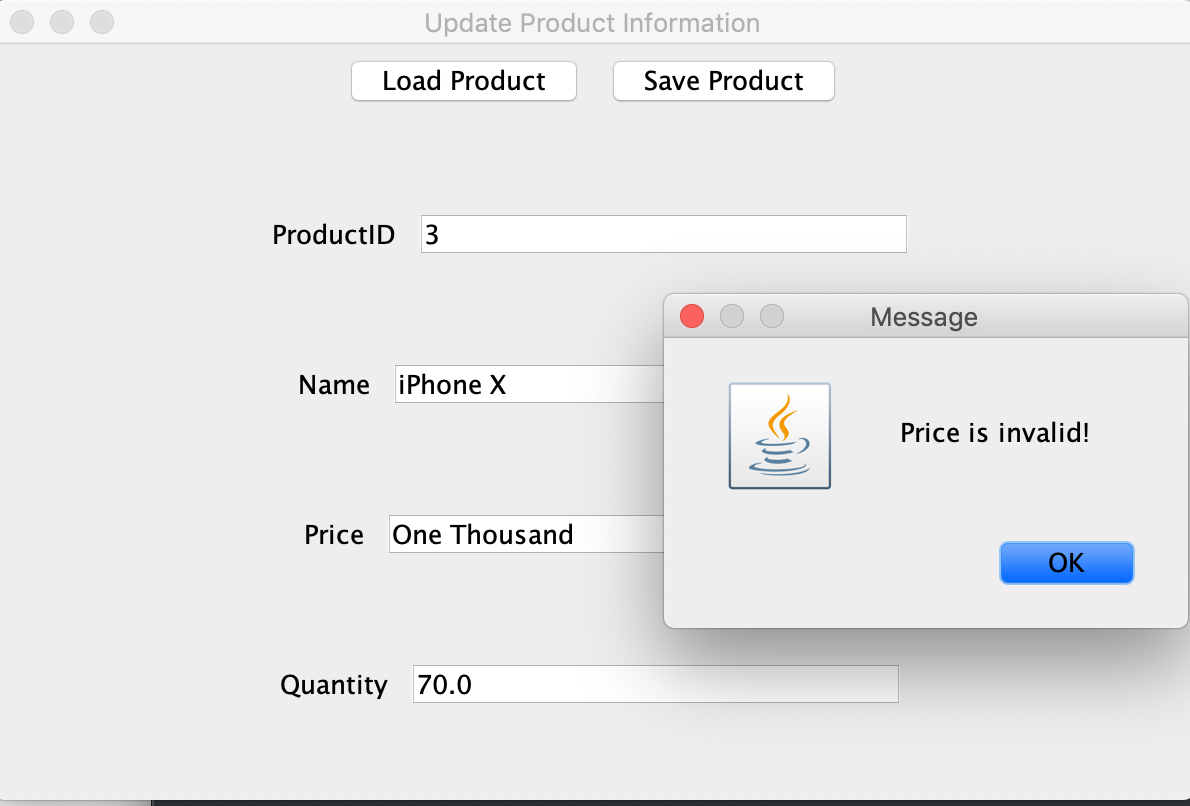
Screen%20Shot%202019-11-20%20at%204.00.15%20PM.pngScreen%20Shot%202019-11-20%20at%204.00.21%20PM.png

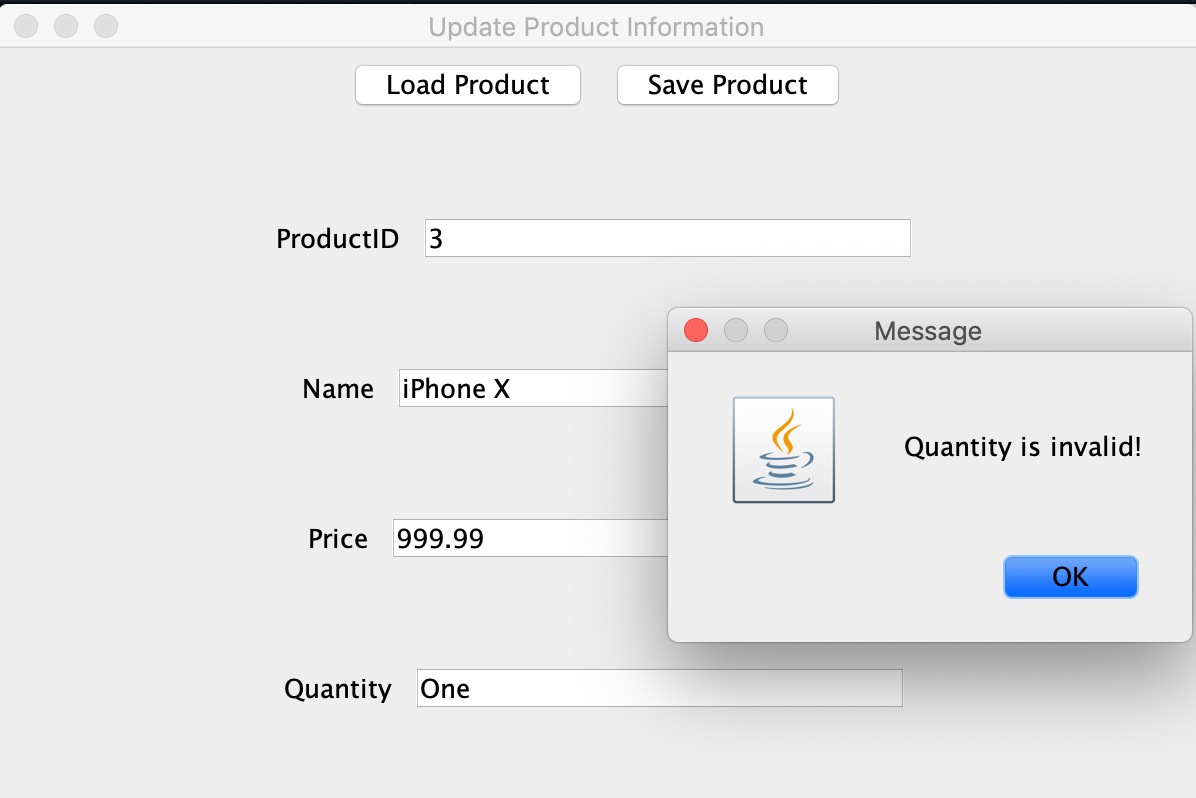
<Exceptions>

* Products

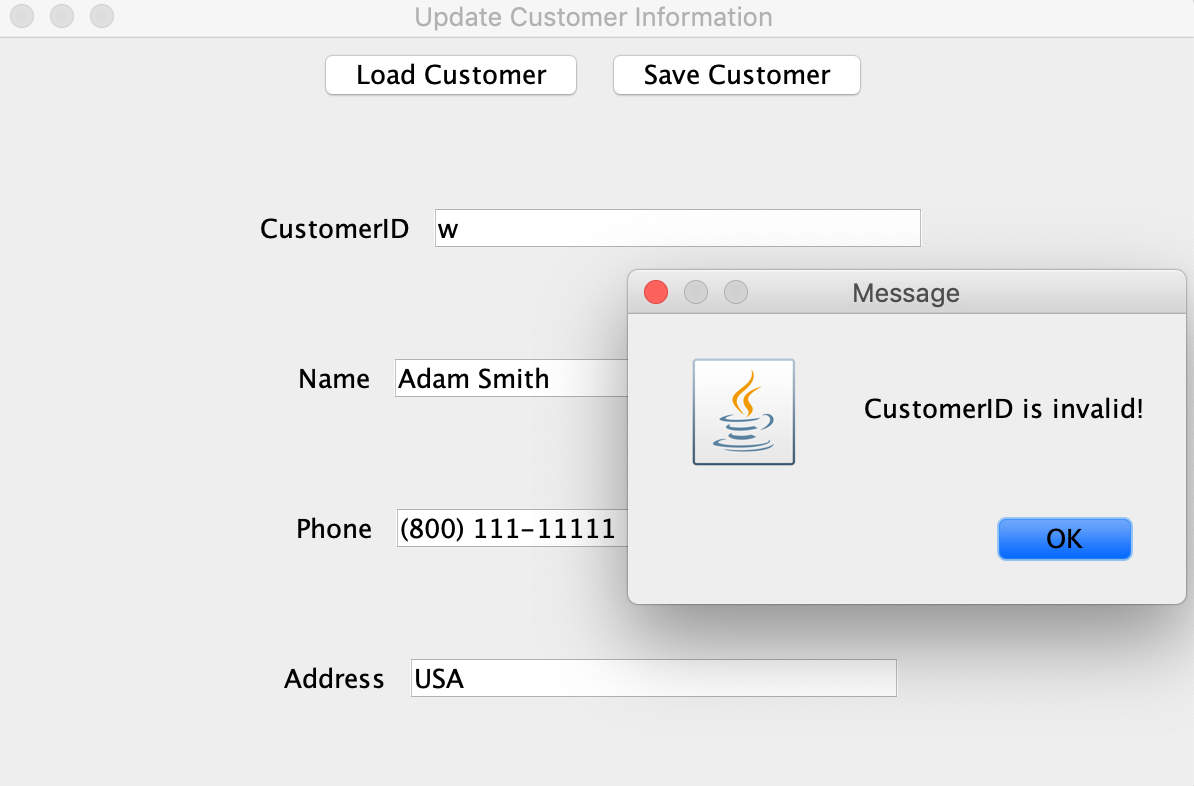


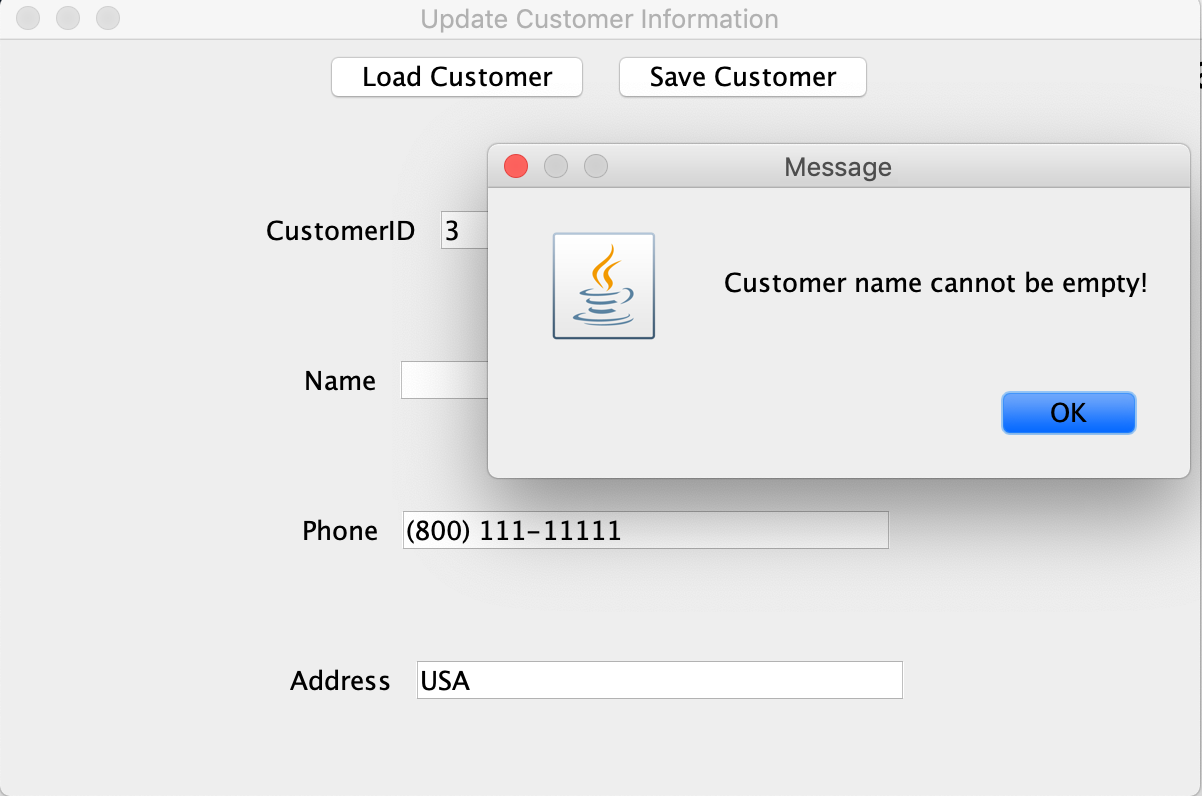


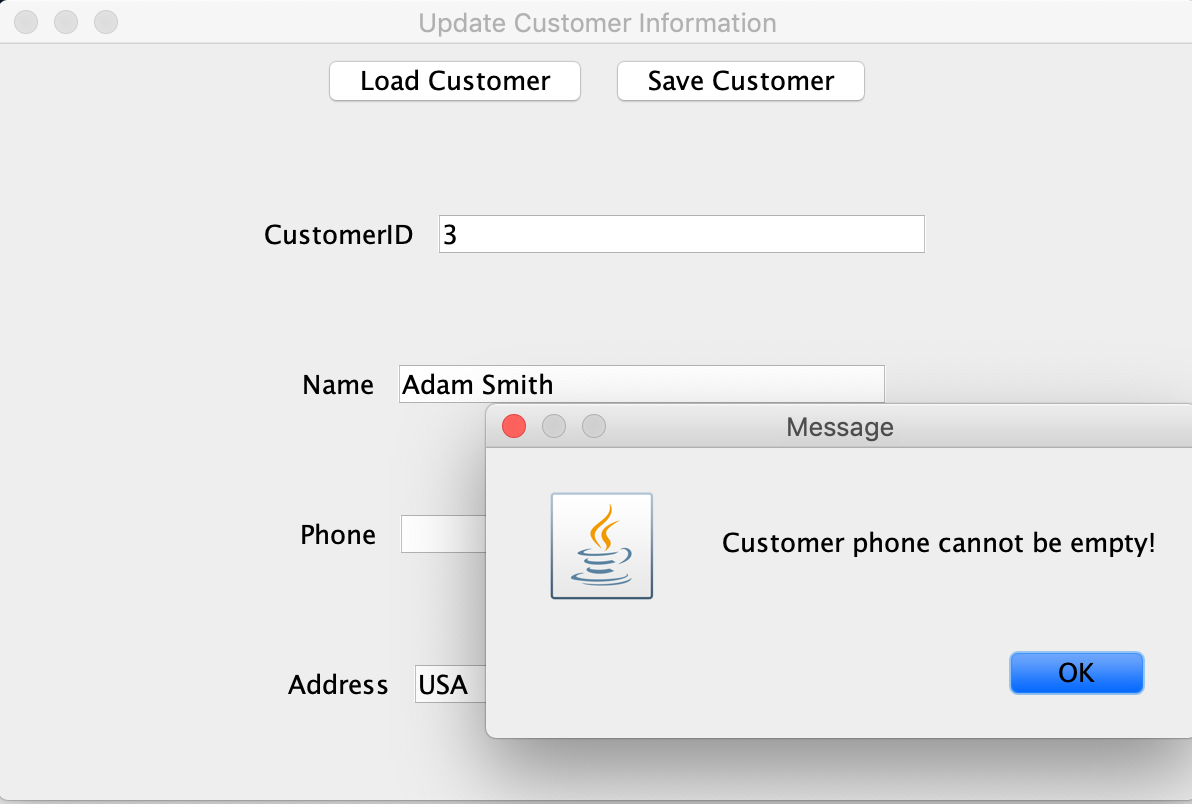




* Customers







* Purchases

