UML Model Report

(In support of assignment 1 for CSC407f)

UML Model

Derived from Iconixsw.com example at

http://www.iconixsw.com/UMLworkbook.html

Thanks to Doug Rosenberg for permission to use this material.

Adapted by Mathew Zaleski

Note: After BACKGROUND section comments added by Zaleski are in Italics.

Table of Contents

PREFACE

This preface and the following introduction added by Mathew Zaleski in support of the first assignment of CSC407 at the University of Toronto for the fall semester of the 2003 academic year.

This document illustrates one way in which the design of an object oriented software system can be captured.

The assumption behind this assignment is that a good design is much easier to follow than to create. No one I consulted has tried an assignment like this before so only time & effort will tell if this approach is pedagogically sound.

GOAL OF ASSIGNMENT 1

Our goal is to introduce, you, a seasoned student programmer, to the productivity of implementing from a good design. Hopefully, in the future, the experience will lead you to create a good design before you code whenever possible. In any case programming from a model is probably a more interesting way of learning use cases, robustness diagrams and sequence diagrams than cramming from a book.

BACKGROUND

The "internet bookstore" has been a popular example program for several years. Presumably it was for this reason Doug Rosenberg chose it as the main example in [2]. The book steps the reader through Rosenberg's design method culminating in the design presented below. Rosenberg has kindly allowed us to use this example in assignment 1. He has expressed interest in your implementations and perhaps, with your permission, will use them as the basis for another book about implementation experience implementing sound designs. Conceivably one or several of your solutions will find their way into an upcoming Addison-Wesley book.

A few editorial comments are in order:

- This may be a one shot opportunity. Creating a design as good as this one is difficult and time consuming.

- In terms of its functionality this is only a small slice of a useful internet bookstore and a somewhat contrived one to boot. We handle the front end of the system (logging in and searching for a book by author and adding it to a virtual shopping car) but omit any back-end functions like checking out, shipping and so on. Rosenberg's example includes more functionality but we felt it would require too much effort to implement.

- A real internet bookstore needs a user interface. However, your implementations would be unnecessarily complicated by a graphical or web interface. A very significant aspect of Rosenberg's design is that it identifies "boundary" objects that can conceptually support different user interface technologies. In our case they will implement a textual interface. In this way we will gain two benefits. First, we will simplify the implementation considerably. Second, we will learn one approach to building a test scaffold to drive interactive applications.

TASK OF ASSIGNMENT 1

Your task is to build a program that faithfully implements the design presented below. Presumably you will choose to build the program in Java. We will expect you to name your classes the same as in the design below.[1] Do not redesign this system for this assignment!

The classes identified as "boundary" classes (see the Robustness diagrams following) interact directly with the user. Your program must be tested with every use case and alternate course. Thus, you must find a way to feed input into and collect output out of the boundary objects such that such testing is possible. The input and output produced by each test must be presented in a document that your TA can examine to determine whether your program works. Note we are NOT specifying precisely what this input and output should be so there will be no attempt made to automatically verify that your programs work on tests other than the ones you provide.

HOW WILL YOU LEARN UML IN TIME?

We will do a crash review of UML in the first few lectures. Though this will be insufficient to teach how to create a UML design it should suffice to build code from an existing design. There are many good books available on UML.

You should start by reestablishing reasonable familiarity with UML, perhaps by quickly re-reading a small UML book like [1]. Then, you should carefully study the robustness and sequence diagrams below. They provide an important way of moving from domain model concepts to design.

Robustness diagrams are fully described in [3].

[1] Several of the classes below include blanks in their names, for instance, "Shopping Cart". In Java you should drop the blank, thus naming the class "`ShoppingCart`"

USE CASE MODEL REPORT

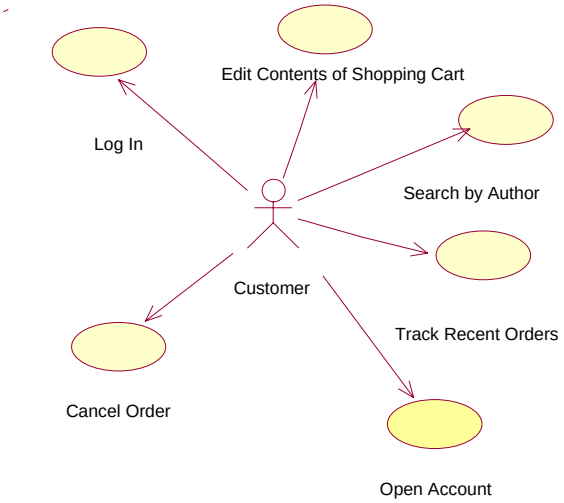
Use cases are used to describe how the system should behave under stimuli. The intent is to anchor the process of domain analysis in actual work that needs to be done by the system. As you will see, the uses cases drive Rosenberg's design method at every stage.

Note that use cases are part of the "domain model" NOT the design model. This is to say they record how an actor might interact with the objects that make up the problem domain. Naturally we will attempt to structure our design so that some of the classes of our design, the so called "entity" classes, correspond to the problem domain.

Use Case Model

The following diagram illustrates how actors relate to use cases. In our case, with one actor, it is not particularly useful

Use Case Diagram - Main



Actor - Customer

Documentation:

The one actor of the system is the user of the bookstore, the person seeking and eventually buying books. (Rosenberg's design had actors representing shipping personnel as well.)

Use Case - Add Item to Shopping Cart

Documentation:

This use case was not documented presumably because Iconix decided it was too simple. In any case, it adds little knowledge once the "edit shopping cart" use case is analyzed.. This use case is "invoked" by other use cases when a selected book is to be added to the shopping cart. In those cases the shopping cart is not visible so the action is behind the scenes. See Edit shopping cart for the interesting version.

List of Associations

Search Results Page Communicates with Add Item to Shopping Cart

Use Case - Cancel Order

Documentation:

Basic Course

The system ensures that the Order is cancelable (in other words, that its status isn't "shipping" or "shipped"). Then the system displays the relevant information for the Order on the Cancel Order Page, including its contents and the shipping address. The Customer presses the Confirm Cancel button. The system marks the Order status as "deleted" and then invokes the Return Items to Inventory use case.

Alternate Course

If the status of the Order is "shipping" or "shipped," the system displays a message indicating that it's too late for the Customer to cancel the order.

(Note, since we have dropped all the actors other than customers from Rosenberg's example the alternative courses can never occur, as there is no one to ship the book in our system.)

List of Associations

Customer Communicates with Cancel Order

Use Case - Edit Contents of Shopping Cart

Documentation:

Basic Course

On the Shopping Cart Page, the Customer modifies the quantity of an Item in the Shopping Cart and then presses the Update button. The system stores the new quantity and then computes and displays the new cost for that Item. The Customer presses the Continue Shopping button. The system returns control to the use case from which it received control.

Alternate Courses

If the Customer changes the quantity of the Item to 0, the system deletes that Item from the Shopping Cart.

If the Customer presses the Delete button instead of the Update button, the system deletes that Item from the Shopping Cart.

If the Customer presses the Check Out button instead of the Continue Shopping button, the system passes control to the Check Out use case.

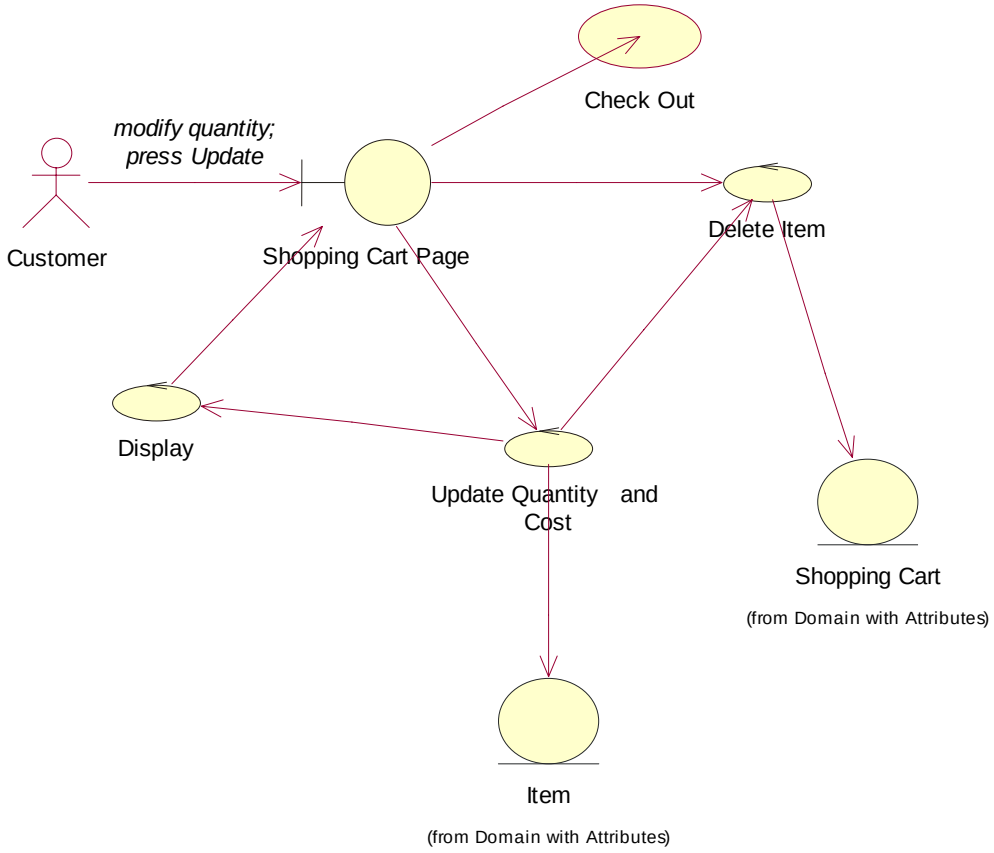
List of Associations

Customer Communicates with Edit Contents of Shopping Cart

Class Diagram - Edit Contents of Shopping Cart Robustness

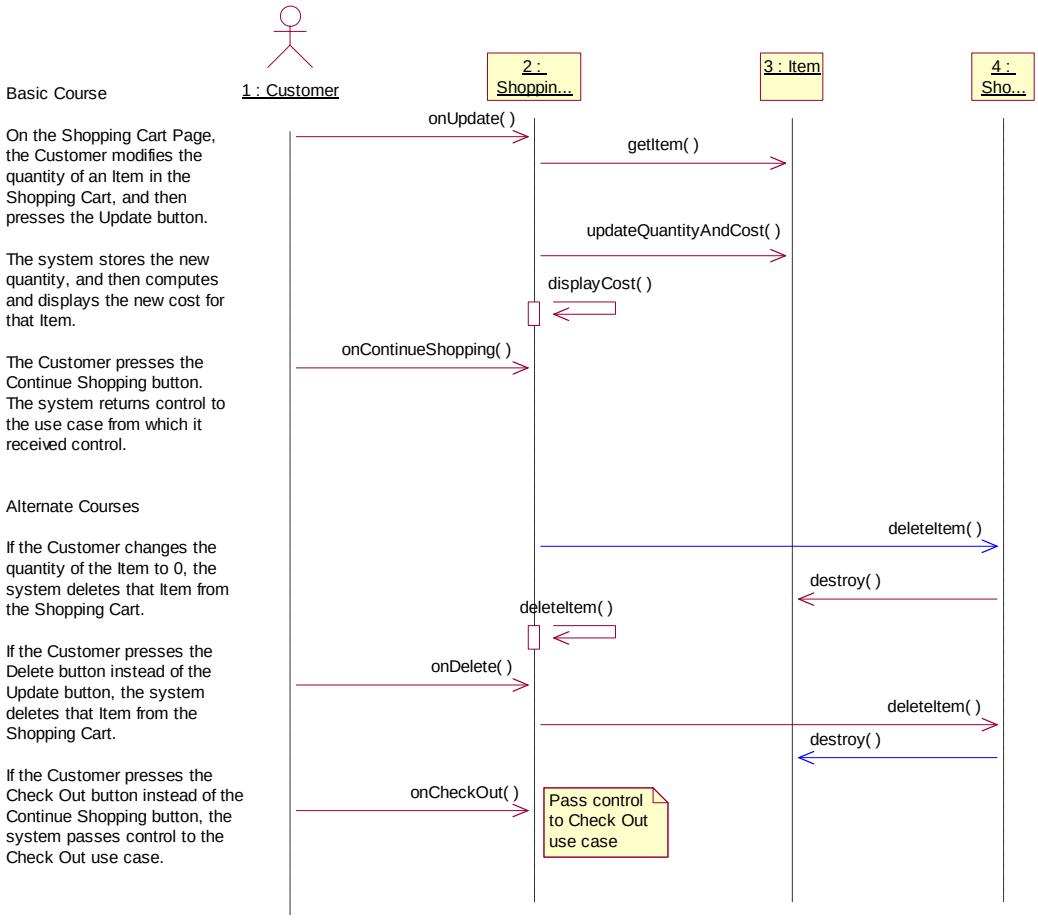
This is the first example of a "robustness" diagram. Note that in addition to actors, boundary (shopping cart page), control (delete item, update quantity and cost, display) entity (shopping cart, item) and use case (check out) domain classes are denoted by specific symbols.

Robustness diagrams are a key intermediate step between the use cases, the domain model and the eventual static class model (or design).



Interaction Diagram - Edit Contents of Shopping Cart Sequence

This is the first sequence diagram or the design. Here we begin the process of identifying design classes by placing methods corresponding to the behavior observed in the use cases and robustness diagram. Note that many of the "controller" classes robustness diagrams wind up as methods.



Use Case - Log In

Documentation:

Basic Course

The Customer clicks the Log In button on the Home Page. The system displays the Login Page. The Customer enters his or her user ID and password and then clicks the Log In button. The system validates the login information against the persistent Account data and then returns the Customer to the Home Page.

Alternate Courses

If the Customer clicks the New Account button on the Login Page, the system invokes the Open Account use case.

If the Customer clicks the Reminder Word button on the Login Page, the system displays the reminder word stored for that Customer, in a separate dialog box. When the Customer clicks the OK button, the system returns the Customer to the Login Page.

If the Customer enters a user ID that the system does not recognize, the system displays a message to that effect and prompts the Customer to either enter a different ID or click the New Account button.

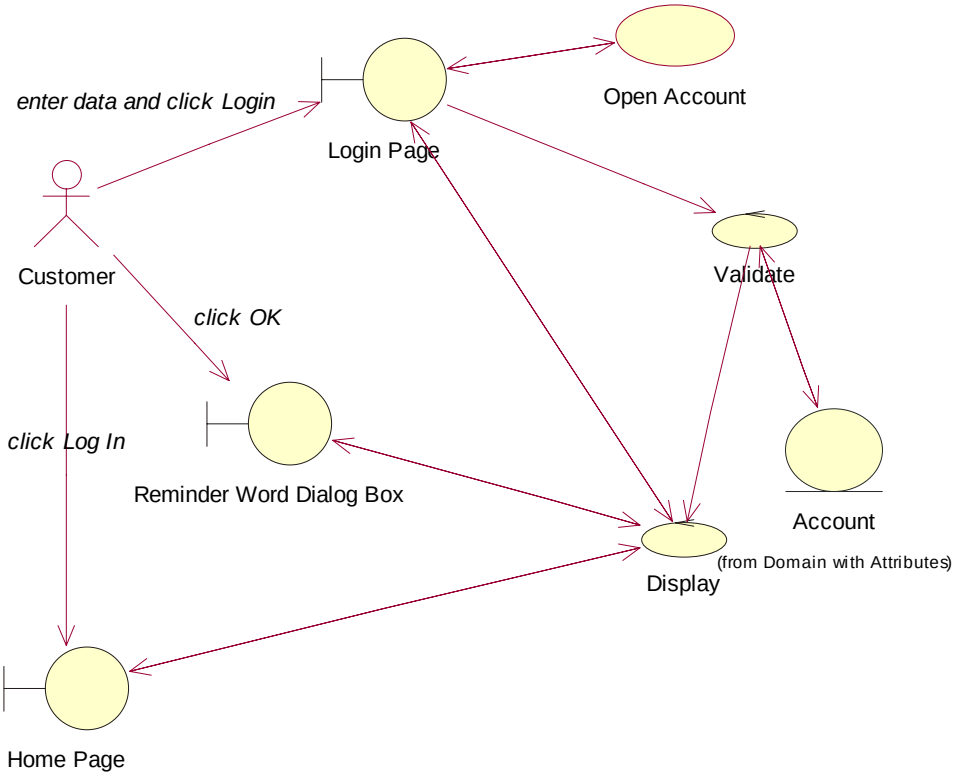
If the Customer enters an incorrect password, the system displays a message to that effect and prompts the Customer to reenter his or her password.

If the Customer enters an incorrect password three times, the system displays a page telling the Customer that he or she should contact customer service, and also freezes the Login Page.

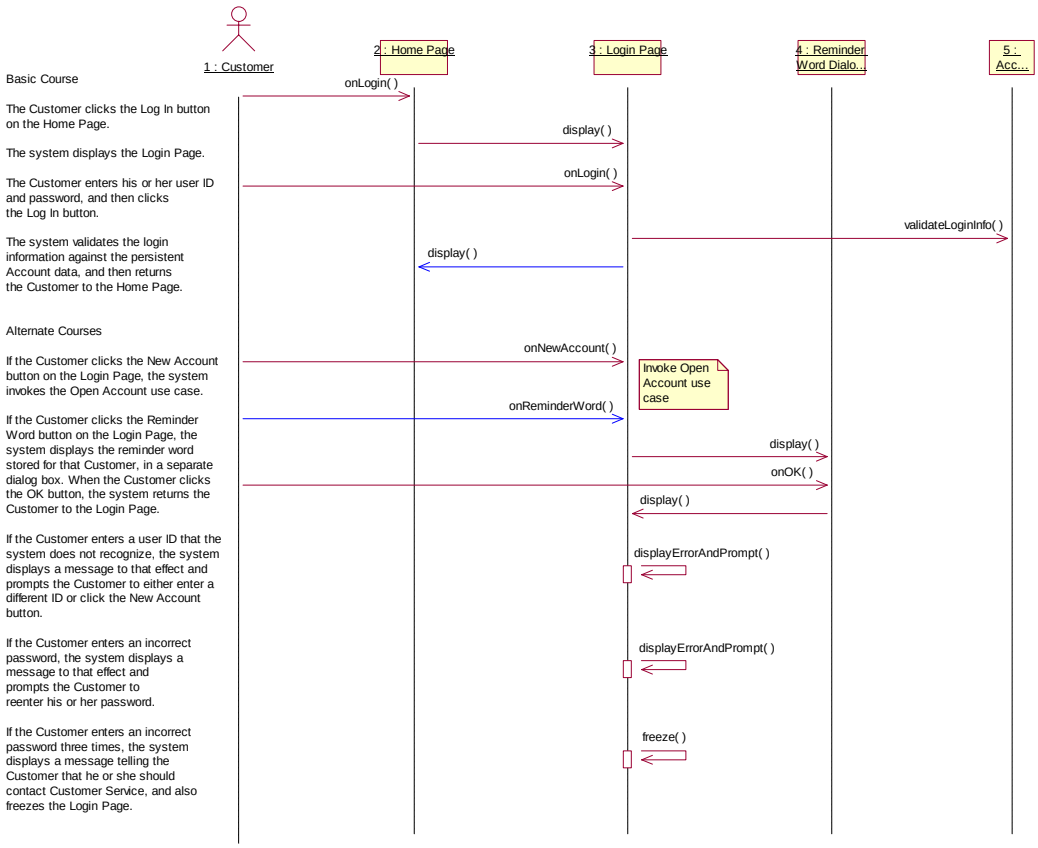
List of Associations

Customer Communicates with Log In

Class Diagram - Log In Robustness



Interaction Diagram - Log In Sequence



Use Case - Open Account

Documentation:

Basic Course

The system displays the New Account Page. The Customer types his or her name, an e-mail address, and a password (twice), and then presses the Create Account button. The system ensures that the Customer has provided valid data and then adds an Account to the Master Account Table using that data. Then the system returns the Customer to the Home Page.

Alternate Courses

If the Customer did not provide a name, the system displays an error message to that effect and prompts the Customer to type a name.

If the Customer provided an email address that's not in the correct form, the system displays an error message to that effect and prompts the Customer to type a different address.

If the Customer provided a password that is too short, the system displays an error message to that effect and prompts the Customer to type a longer password.

If the Customer did not type the same password twice, the system displays an error message to that effect and prompts the Customer to type the password correctly the second time.

If the account is already in the master account table, notify the user.

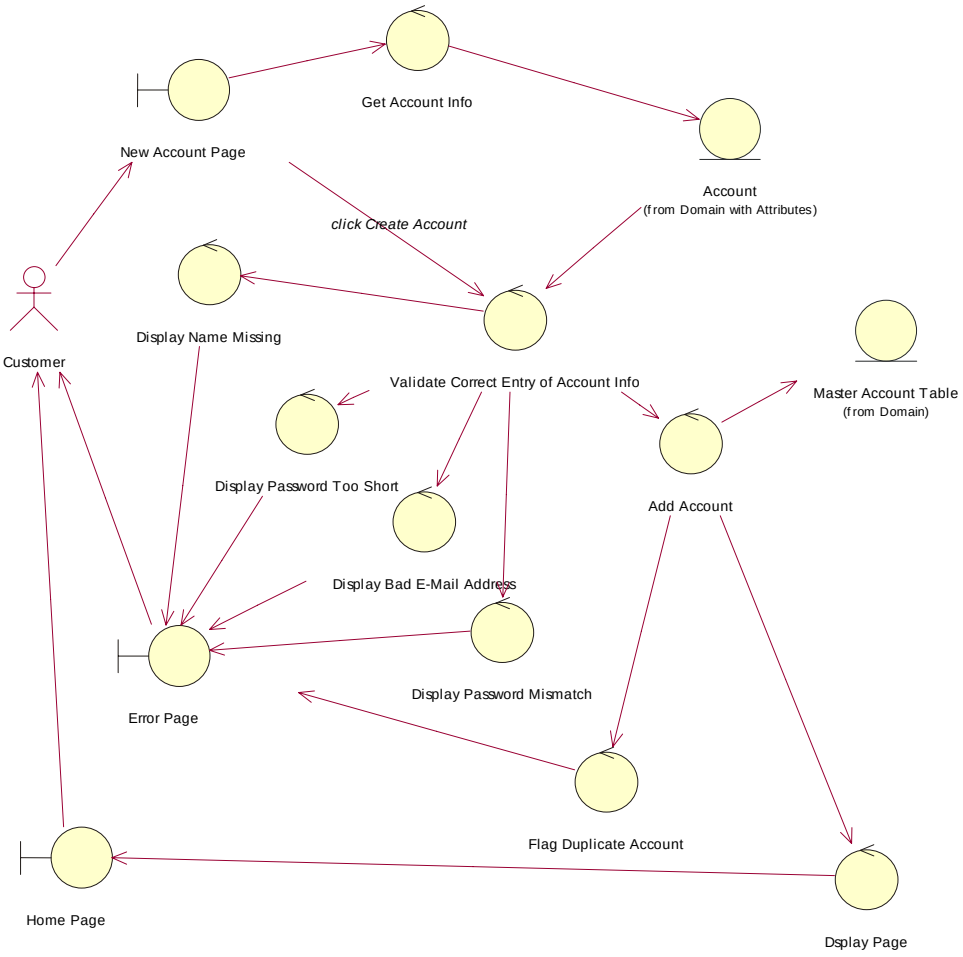
List of Associations

Customer Communicates with Open Account

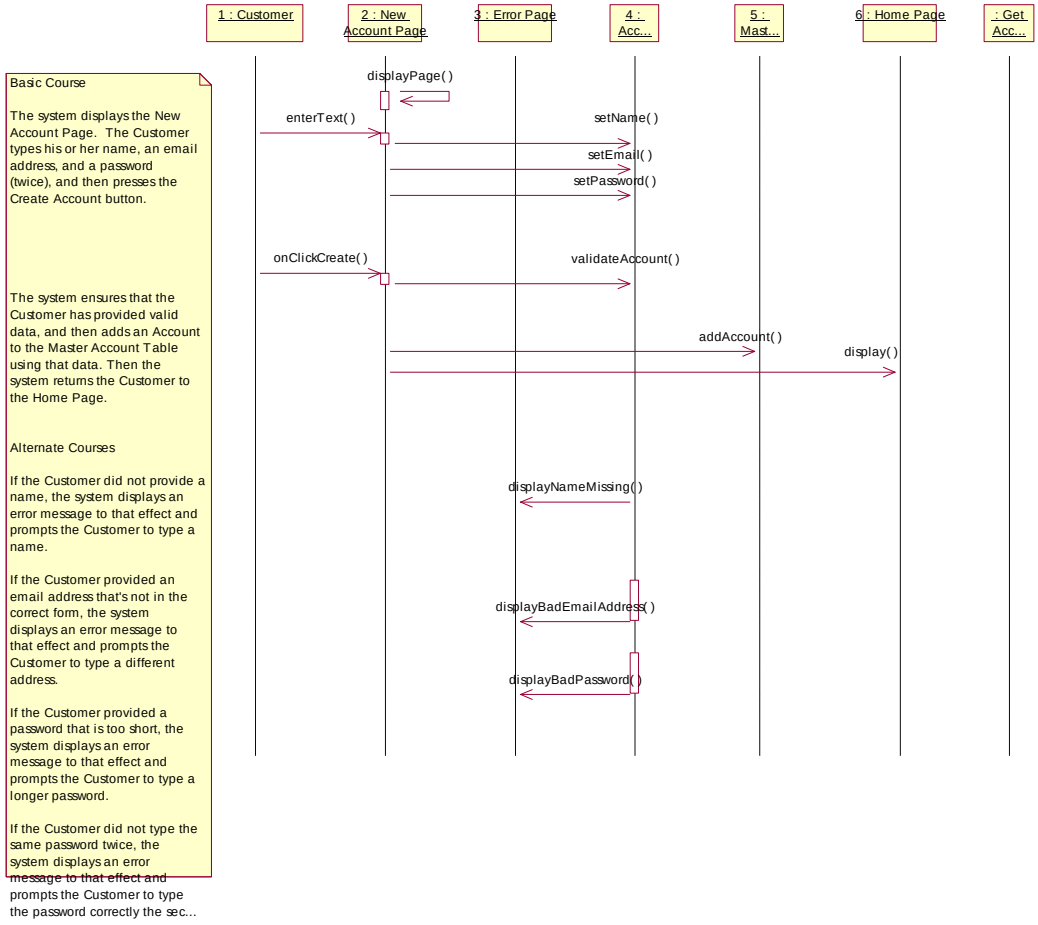
Login Page Communicates with Open Account

Open Account Communicates with Login Page

Class Diagram - Open Account Robustness



Interaction Diagram - Open Account Sequence Diagram



Use Case - Search by Author

Documentation:

Basic Course

The Customer types the name of an Author on the Search Page and then presses the Search button. The system ensures that the Customer typed a valid search phrase, and then searches the Catalog and retrieves all of the Books with which that Author is associated. The the system retrieves the important details about each Book, and creates a Search Results object with that information.

Then the system displays the list of Books on the Search Results Page, with the Books listed in reverse chronological order by publication date. Each entry has a thumbnail of the Book's cover, the Book's title and authors, the average Rating, and an Add to Shopping Cart button. The Customer presses the Add to Shopping Cart button for a particular Book. The system passes control to the Add Item to Shopping Cart use case.

Alternate Courses

If the Customer did not type a search phrase before pressing the Search button, the system displays an error message to that effect and prompts the Customer to type a search phrase.

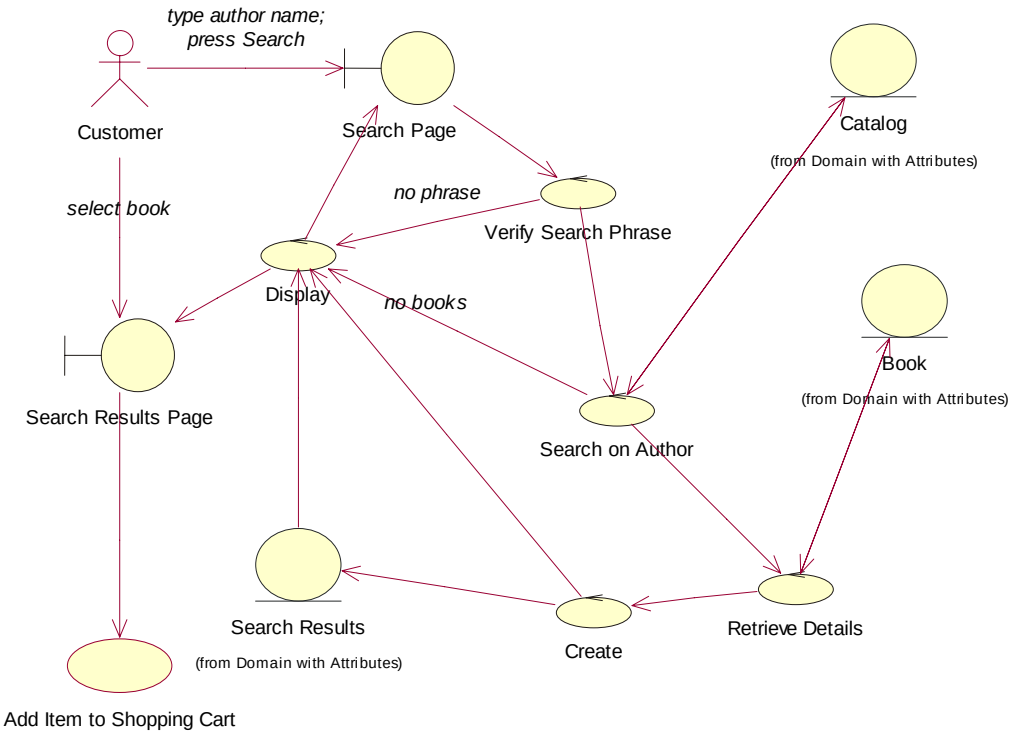
If the system was unable to find any Books associated with the Author that the Customer specified, the system displays a message to that effect and prompts the Customer to perform a different search.

If the Customer leaves the page in a way other than by pressing an Add to Shopping Cart button, the system returns control to the use case from which this use case received control.

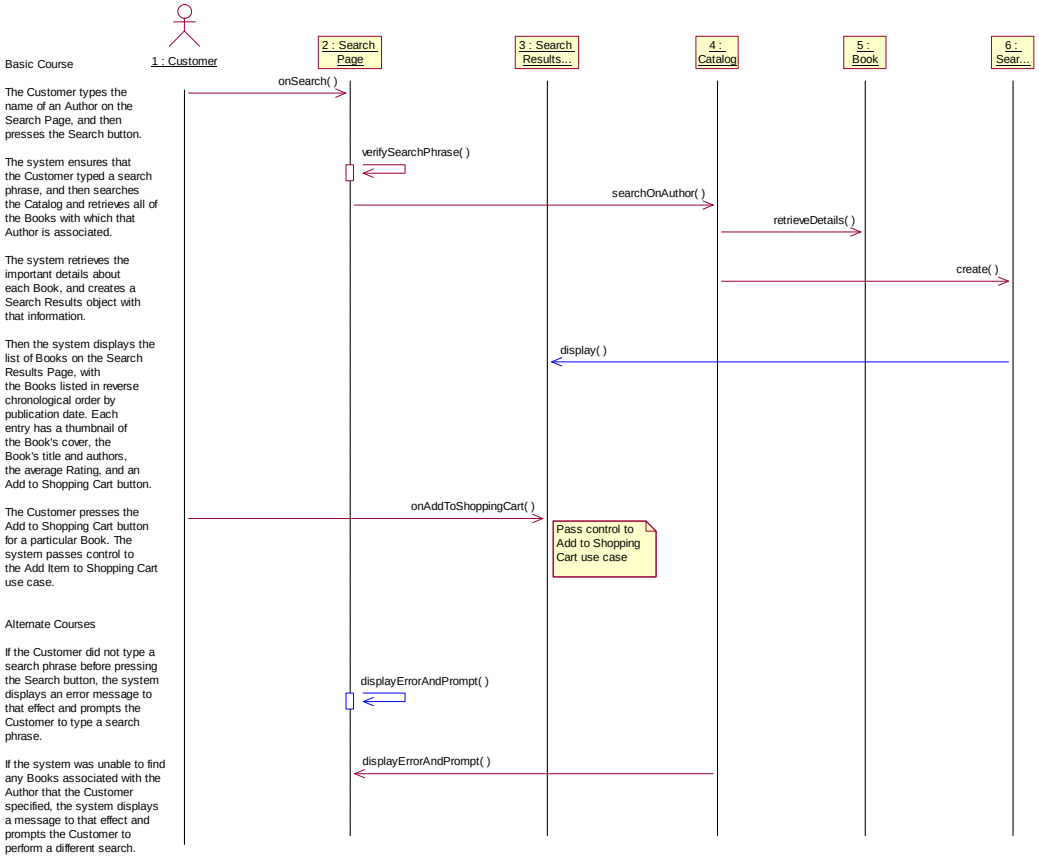
List of Associations

Customer Communicates with Search by Author

Class Diagram - Search by Author Robustness



Interaction Diagram - Search by Author Sequence



Use Case – Add item to shopping cart

To simply matters assume that it can only be invoked from author search. Thus we learn little from this use case over what "edit shopping cart" has to offer.

Use Case - Track Recent Orders

Documentation:

Basic Course

The system retrieves the Orders that the Customer has placed within the last 30 days and displays these Orders on the Order Tracking Page. Each entry has the Order ID (in the form of a link), the Order date, the Order status, the Order recipient, and the Shipping Method by which the Order was shipped.

The Customer clicks on a link.The system retrieves the relevant contents of the Order, and then displays this information, in view-only mode, on the Order Details Page. The Customer presses OK to return to the Order Tracking Page.

Once the Customer has finished viewing Orders, he or she clicks the Account Maintenance link on the Order Tracking Page. The system returns control to the invoking use case.

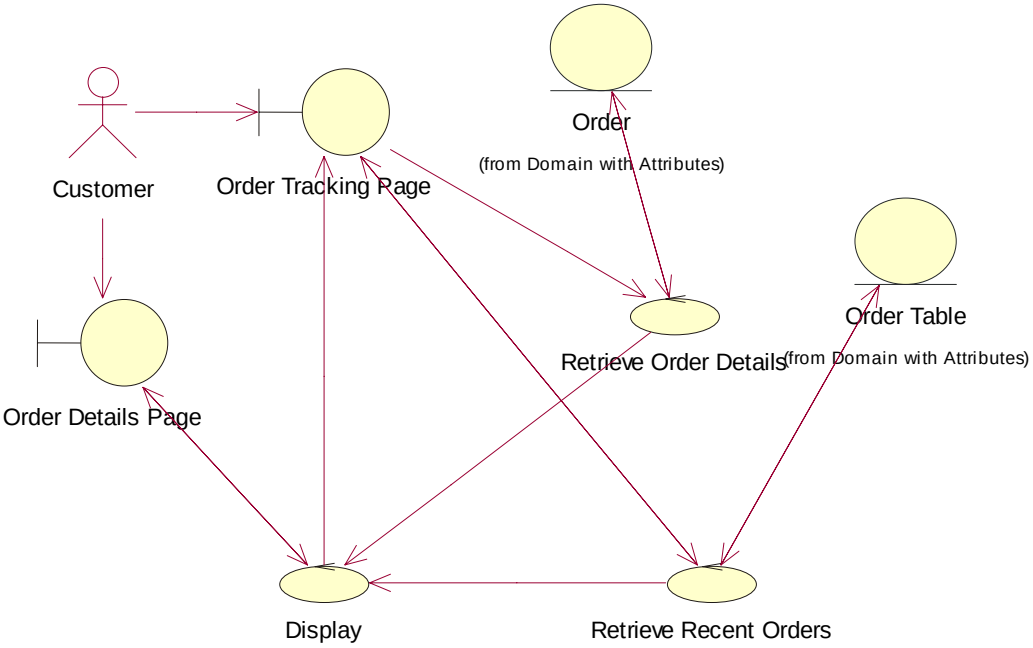
Alternate Course

If the Customer has not placed any Orders within the last 30 days, the system displays a message to that effect on the Order Tracking Page.

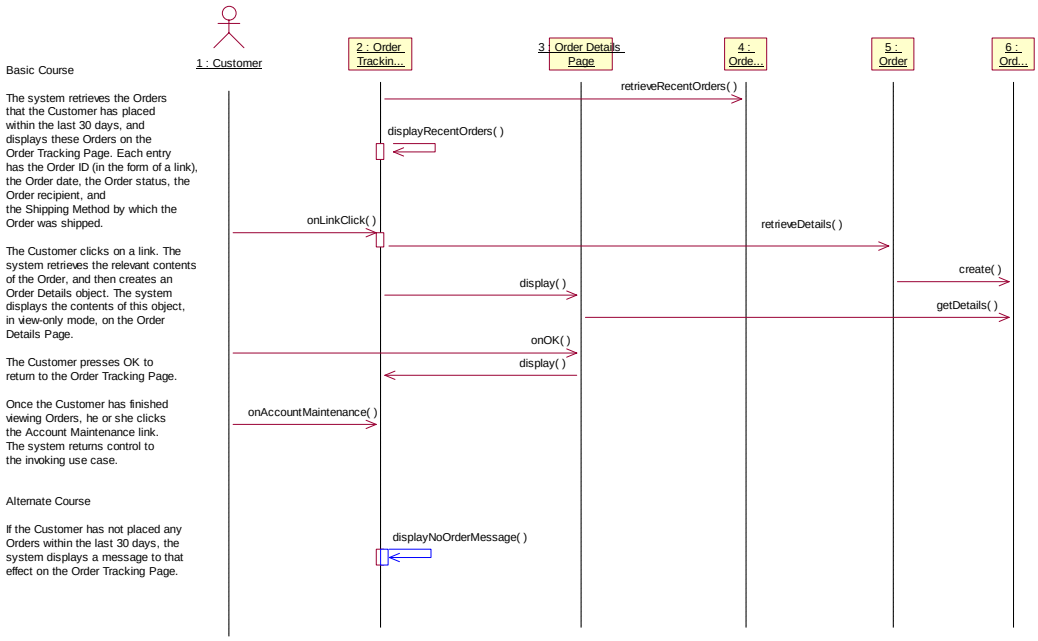
List of Associations

Customer Communicates with Track Recent Orders

Class Diagram - Track Recent Orders Robustness



Interaction Diagram - Track Recent Orders Sequence



Basic Course

The system retrieves the Orders that the Customer has placed within the last 30 days, and displays these Orders on the Order Tracking Page. Each entry has the Order ID (in the form of a link), the Order date, the Order status, the Order recipient, and the Shipping Method by which the Order was shipped.

The Customer clicks on a link. The system retrieves the relevant contents of the Order, and then creates an Order Details object. The system displays the contents of this object, in view-only mode, on the Order Details Page.

The Customer presses OK to return to the Order Tracking Page.

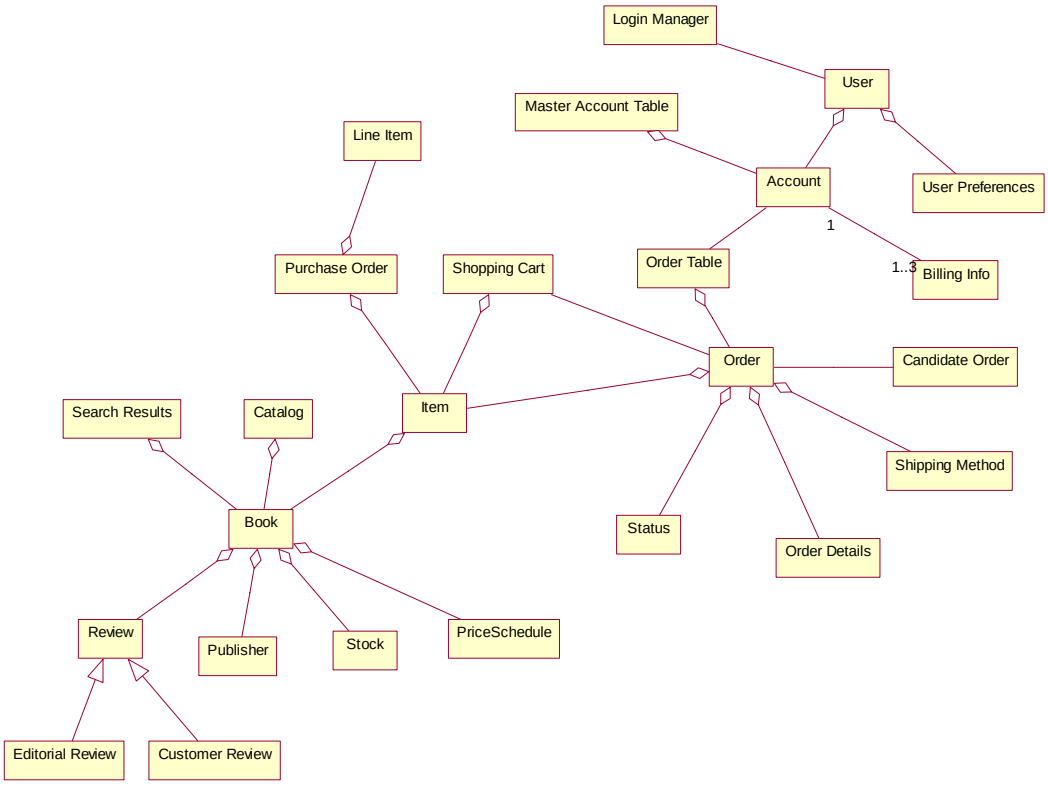
Once the Customer has finished viewing Orders, he or she clicks the Account Maintenance link. The system returns control to the invoking use case.

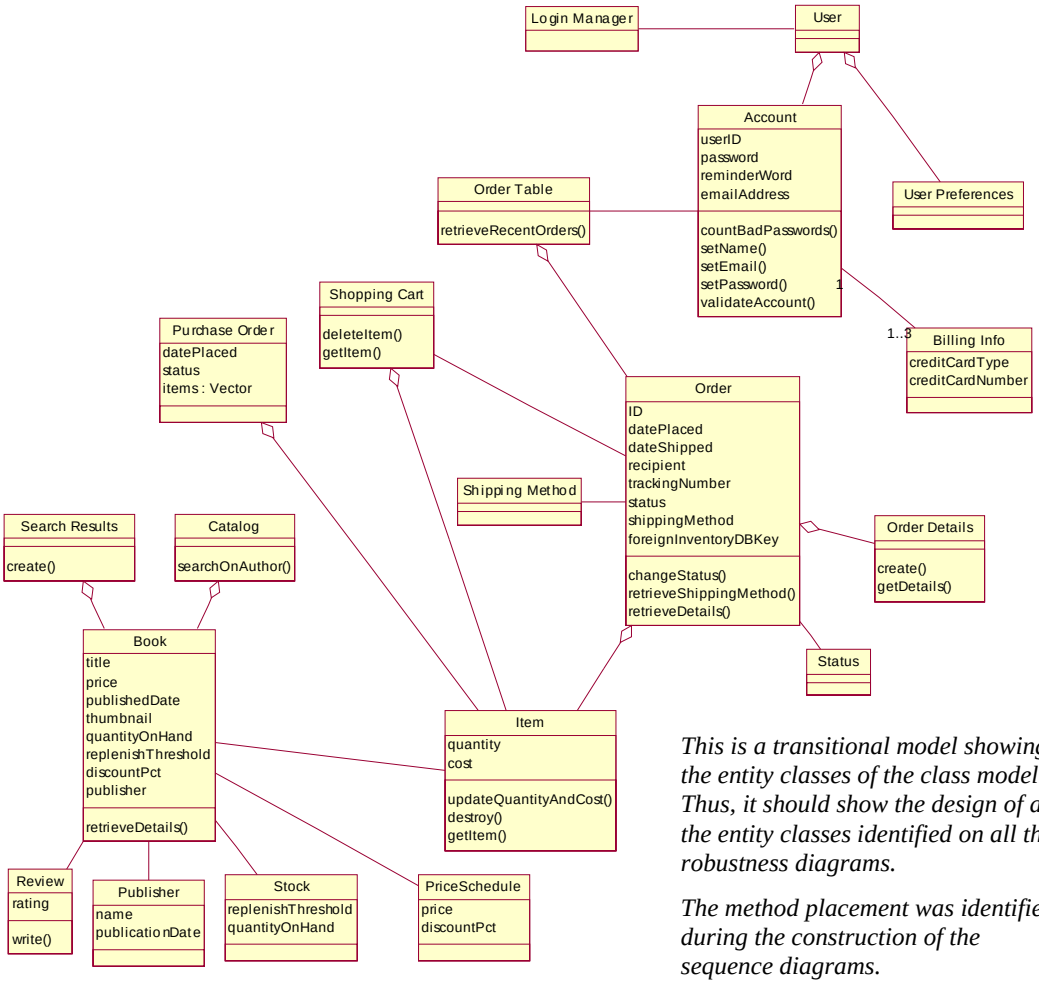
Alternate Course

If the Customer has not placed any Orders within the last 30 days, the system displays a message to that effect on the Order Tracking Page.

Domain Model

This is a model of the problem domain – not a model of proposed software. The boxes on this diagram represent concepts in our model of the problem domain. One of the advantages of object oriented techniques, of course, is that we can build software that mimics the structure of the domain model reasonably naturally. This domain model includes all the entity classes that we needed to build the robustness diagrams.

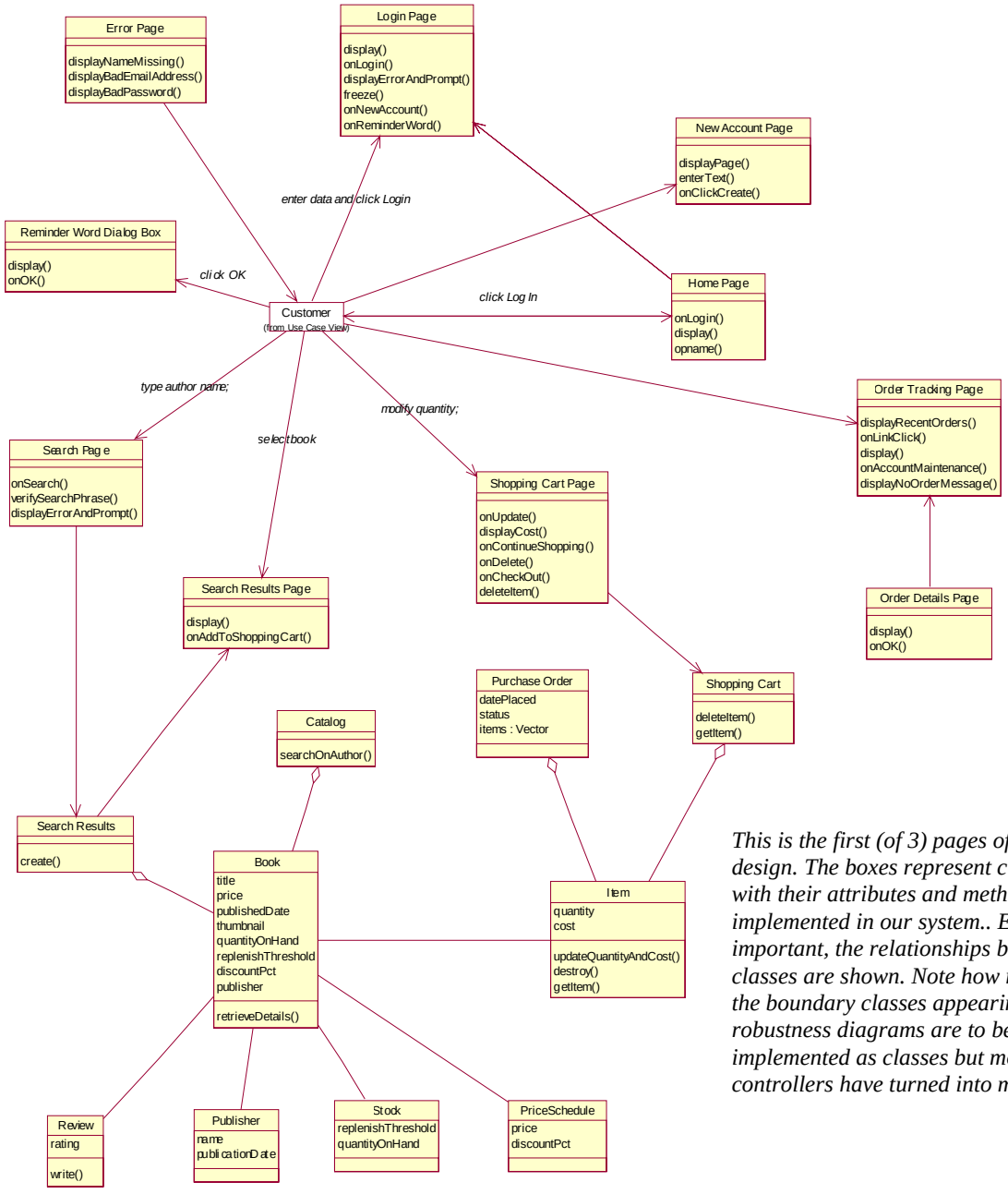




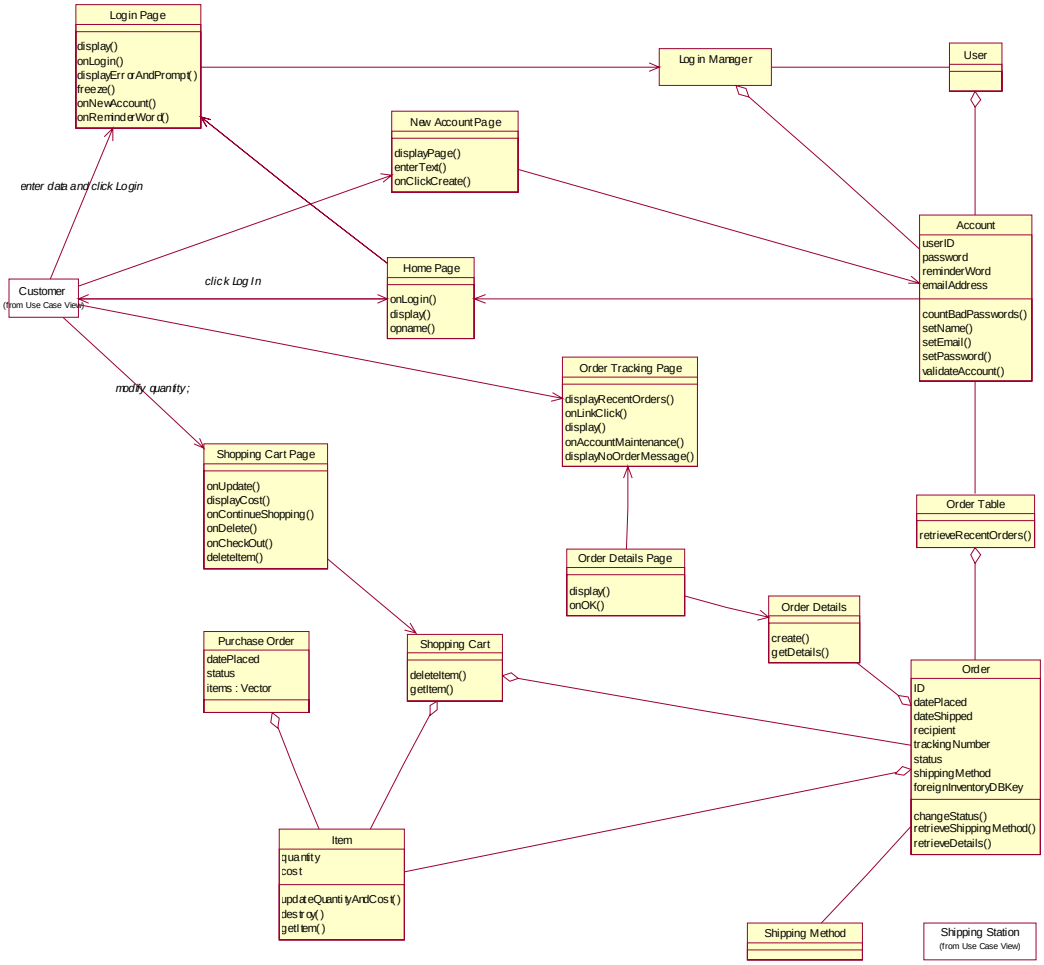
This is a transitional model showing the entity classes of the class model. Thus, it should show the design of all the entity classes identified on all the robustness diagrams.

The method placement was identified during the construction of the sequence diagrams.

Static Object Model

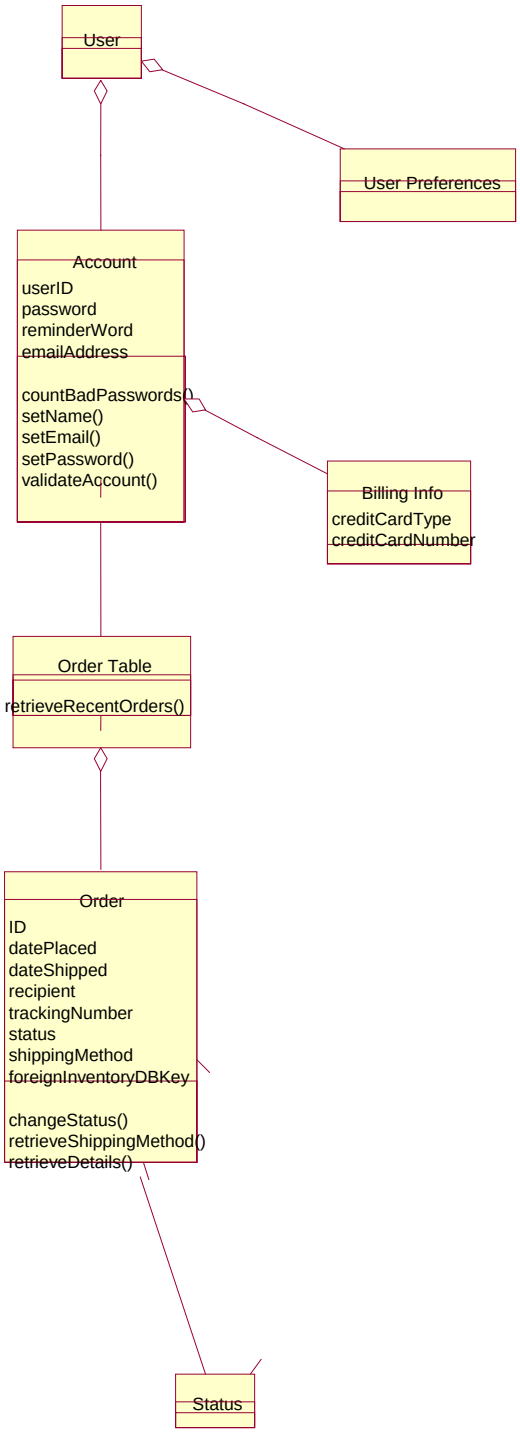


This is the first (of 3) pages of our design. The boxes represent classes with their attributes and methods to be implemented in our system.. Equally important, the relationships between classes are shown. Note how most of the boundary classes appearing on the robustness diagrams are to be implemented as classes but most controllers have turned into methods.


Login Page
display()
onLogin()
displayErrorAndPrompt()
freeze()
onNewAccount()
onReminderWord()
Login Manager
User
New Account Page
displayPage()
enterText()
onClickCreate()
enter data and click Login
Account
userID
password
reminderWord
emailAddress
countBadPasswords()
setName()
setEmail()
setPassword()
validateAccount()
Home Page
onLogin()
display()
opname()
click Log In
Customer
(from Use Case View)
Order Tracking Page
displayRecentOrders()
onLinkClick()
display()
onAccountMaintenance()
displayNoOrderMessage()
modify quantity;
Shopping Cart Page
onUpdate()
displayCost()
onContinueShopping()
onDelete()
onCheckOut()
deleteItem()
Order Table
retrieveRecentOrders()
Order Details Page
display()
onOK()
Order Details
create()
getDetails()
Purchase Order
datePlaced
status
items : Vector
Shopping Cart
deleteItem()
getItem()
Order
ID
datePlaced
dateShipped
recipient
trackingNumber
status
shippingMethod
foreignInventoryDBKey
changeStatus()
retrieveShippingMethod()
retrieveDetails()
Item
quantity
cost
updateQuantityAndCost()
destroy()
getItem()
Shipping Method
Shipping Station
(from Use Case View)

This page intentionally left blank

(to ensure that three pages of the static design appear on separate leaves when printed double sided.)


User
User Preferences
Account
userID
password
reminderWord
emailAddress
countBadPasswords()
setName()
setEmail()
setPassword()
validateAccount()
Billing Info
creditCardType
creditCardNumber
Order Table
retrieveRecentOrders()
Order
ID
datePlaced
dateShipped
recipient
trackingNumber
status
shippingMethod
foreignInventoryDBKey
changeStatus()
retrieveShippingMethod()
retrieveDetails()
Status

REFERENCES

1. Martin Fowler and Kendall Scott, UML Distilled Second Edition, A Brief Guide to the Standard Object Modeling Language, Addison-Wesley, 1999.

2. Doug Rosenberg and Kendall, Scott, Applying Use Case Driven Object Modeling with UML, An Annotated E-commerce example", Addison-Wesley, 2001.

3. Doug Rosenberg and Kendall, Scott, Use Case Driven Object Modeling with UML, A Practical Approach", Addison-Wesley, 1999.