Concordia University
Engineering and Computer Science

COEN 6312
Model Driven Software Engineering
Winter 2016

Deliverable 3
(ONLINE BOOK STORE)

(Team-Singh)

Guneet Singh
Rattandeep Singh
Satinder pal Singh

Table of Contents

List of Figures

Class Diagram of Online Book Store

Class Diagram

The **class diagram** is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and booking different aspects of a system but also for constructing executable code of the software application.

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object, oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram.

In the diagram, classes are represented with boxes that contain three parts:

- The top part contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle part contains the attributes of the class. They are left aligned and the first letter is lowercase.
- The bottom part contains the methods the class can execute. They are also left aligned and the first letter is lowercase.

UML Class diagram of Online Bookstore System is:

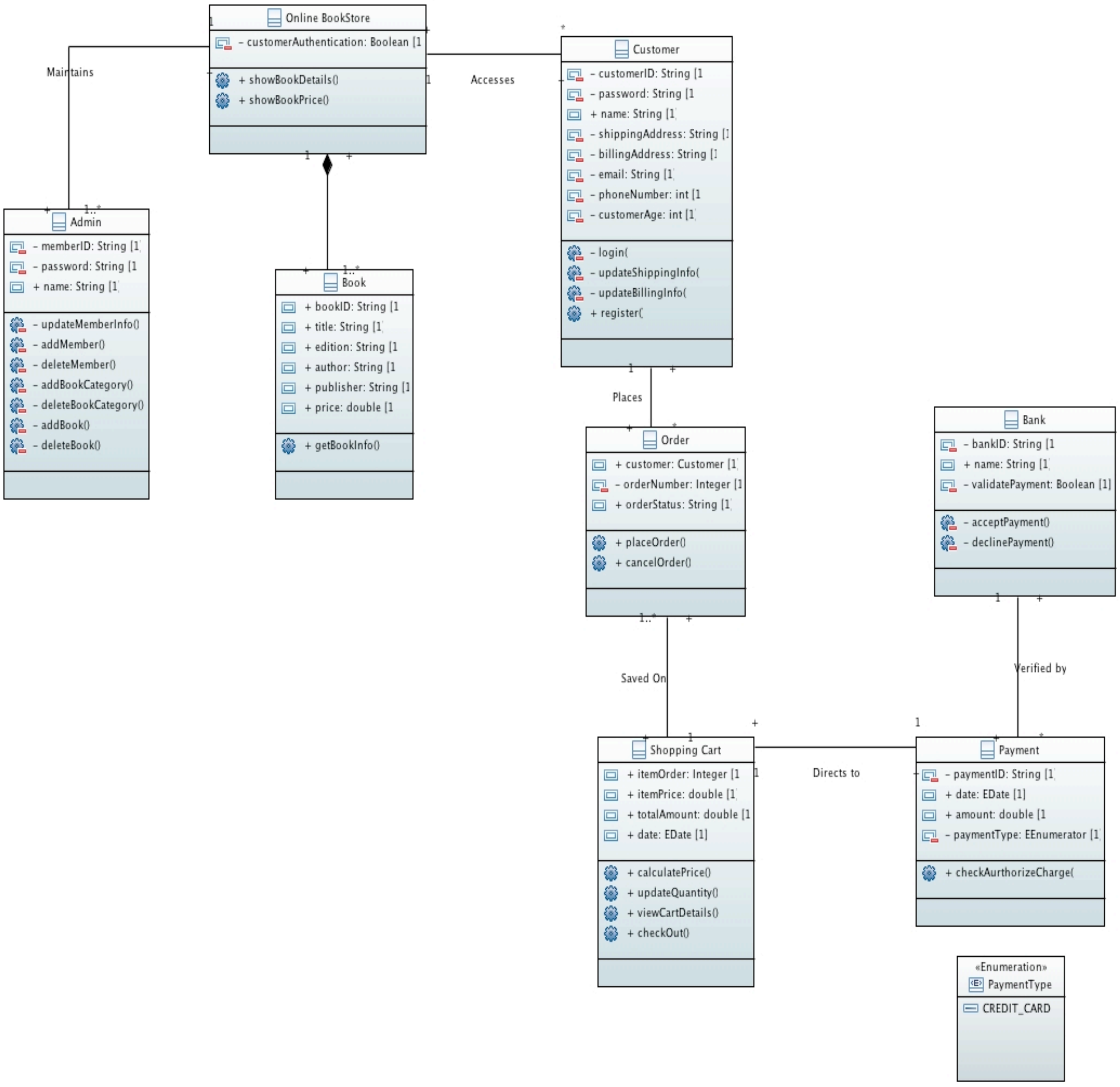

Fig: Class diagram for “Online BookStore System” made in Papyrus.

Customer

The customer searches for the book and if finds, registers with the system and makes the payment. Once the Order is placed, the customer receives the order. The Customer receives notifications via email.

The attributes and methods involved in this class are:

Attributes:

- **customerID:** The CustomerID is the unique id that a customer will have to be able to be differentiated from the other customers.
- **customerAge:** The Customer age should be greater than 15 to be able to create an account with the system.

Methods:

- **login:** The customer will be able to access the system after logging in.
- **updateShippingInfo:** The customer can update the shipping information.
- **register:** The customer creates an account and gets access to buy the books.
- **updateBillingInfo:** The customer can update the billing information.

Book

The Book contains various books that are available for purchase to the customer. These are added to the system by the admin. It keeps track of the status of the books. An unregistered customer can search for the books but cannot buy it unless it gets registered with the system.

The attributes and methods involved in this class are:

Attributes:

- **bookID:** The Book id is the unique id using which one book is separated from the other book. The customer can also search a book using the BookID.
- **author**: The Author is the one who writes or authors the book.
- **title:** The Title is the name using which a customer can search for the book.
- **edition:** The edition gives the various editions of a book that are available.
- **publisher:** The Publisher shows the publisher of a particular book. The customer can search for a particular book by its publisher or can search different books that are available by the mentioned publisher.

- **price:** The price attribute gives the amount of the book.

Methods:
- **getBookInfo:** The book info consists of book related information like author, publisher, information about the latest additions from the previous version (if any) etc.

Payment

The customer needs to pay the amount in order to buy the desired book. The customer can pay the amount using a Credit.

The attributes involved in this class are:

Attributes:

- **paymentID:** The payment id is required to identify the total payment associated with the purchase.
- **amount:** The amount is associated with the book selected.
- **paymentType:** The payment can only be made using a credit card.

Methods:

- **checkAurthorizeCharge:** To check if the amount charged is valid in order to proceed.

Admin

The administrator takes care of all the operations involved in the BookStore System. The administrator allows the customer to create an account. The Administrator manages the books and customers. The Administrator is responsible for registering and deregistering Customers. The administrator also takes care of adding and removing the Books.

The attributes and methods involved in this class are:

Attributes:

- **memberID:** The administrator uses the member id to register, deregister and manage

the account of that particular customer.
- **password:** The administrator uses the password to log into the system.

Methods:

- **updateMemberInfo:** The Administrator is responsible for updating the member information.
- **addMember:** The administrator is responsible for registering the customer.
- **deleteMember:** The administrator has full access to deregister the customer.
- **addBookCategory:** The administrator is responsible for adding new book categories to the system.
- **deleteBookCategory:** The administrator can remove/delete a book category if needed.
- **addBook**: the administrator adds the books authored by the author, as author does not have access to add the books to the system directly. The author sends the books to the admin and the admin add's it to the system.
- **deleteBook:** The administrator can remove the book, if the book needs to be removed from the system.

Order

The customer searches for a book and if found, places an order to get it delivered. One order can contain many books. The customer can update and cancel the order.

The attributes and methods involved in this class are:

Attributes:

- **customer:** It is required to be able to track who has placed the order.
- **orderNumber:** The order number is required for the customer and the administrator to track the status of the order.
- **orderStatus:** The order status shows if the order is placed or cancelled.

Methods:

- **placeOrder:** The customer has the flexibility to update the order before placing it.
- **cancelOrder:** The customer can discard the order or cancel the order, if not placed.

Shopping Cart

The shopping cart contains the books that have been selected by the customer. The customer can look at the shopping cart to make any changes before buying the desired books. The Customer can view the final price of all the selected books and can update i.e. add or delete quantities of the book, from the cart.

The attributes and methods involved in this class are:

Attributes:

- **itemOrder:** This is used to view the books and the quantity of each book that the customer has selected.
- **itemPrice:** ItemPrice is used to view the price of each book present in the cart.
- **totalAmount:** This shows the total amount for all the books present in the shopping cart.
- **date:** It is used to identify date associated with each order.

Methods:

- **calculatePrice:** It calculates the total price for all the books.
- **updateQuantity:** Books can be added or deleted from the cart before final checkout.
- **viewCartDetails:** Cart details can be viewed before checkout. It shows the books the customer has selected and their quantities.
- **checkout:** The customer can proceed to checkout for the payment.

Bank

The bank validates the credit card information provided by the customer and verifies the funds. If sufficient funds are available, it accepts the payment and the order is placed and if funds are insufficient, the order is cancelled.

The attributes and methods involved in this class are:

Attributes:

- **bankID:** The BankId is used to identify the bank associated with the credit card provided by the customer.
- **name:** It provides the name of the bank.
- **validatePayment:** It validates the credit card information provided by the customer and verifies the funds. If sufficient funds are available, it accepts the payment and the order is placed and if funds are insufficient, the order is cancelled.

Methods:

- **acceptPayment:** If sufficient funds are available, then the payment is accepted and the order is placed.
- **declinePayment:** If sufficient funds are not available, then the payment is not accepted and the order is cancelled.

Constraints

1. Customer already has a customerID

Pre: Customer.allInstances.customerID->includes (customerID)

Post: Customer.allInstances.customerID = customer.allInstances.customerID@pre

Explanation:

According to this constraint, the customer already has a CustomerID associated to it.

2. Each customer should have different customerID

Context Customer

Inv distinctcustomerID: Customer.allInstances -> forAll (customer1, customer2 | customer1 <> customer2 implies customer1.customerID<> customer2.customerID)

Explanation:

According to this constraint, No two customers can have same customerID. Each customer has a different customerID.

3. Each Book should belong to exactly one Category

Context Category

Inv BookHasoneCategory: Category.allInstances -> forAll (C1, C2 |C1<>C2 && C1.getCategoryBooks -> includes (book) implies C2. getCategoryBooks -> excludes (book))

Explanation:

According to this constraint, each book available in the system should belong to only one of the categories (i.e. either it can belong to Fiction, Biography, History, Crime etc.).

4. Each Book should have a different bookID

Context Book

Inv DistinctBookID: Book.allInstances -> forAll (B1, B2 |B1 <> B2 implies B1.bookID<>B2.bookID)

Explanation:

According to this constraint, no two books can have a same bookID. Each book has a different bookID associated to it.

5. The OrderNumber for each Order must be different

Context Order

Inv DistinctOrderNumber: Order.allInstances -> forAll (O1, O2 |O1 <> O2 implies O1.OrderNumber<> O2.OrderNumber)

Explanation:

According to this constraint, each order number must be different from one another. It is not possible for two order numbers to be same.

6. Each Order should have some books

Context Order

Inv OrderHasbooks self.contains -> notEmpty ()

Explanation:

According to this constraint, No bookOrder can be empty or without a book. A book order must contain a minimum of one book.

7. Each BookOrder belongs to exactly one customer

Context Order

Inv OrdertoOneCustomer Order.allInstances -> forAll (O1, O2 | O1.OrderNumber <> O2.OrderNumber implies O1.customerID <> O2.customerID)

Explanation:

According to this constraint, a book order can only belong to one particular customer.

8. Quantity should always be a positive value

Context Order

Inv Order Positive self.quantity > 0

Explanation:

According to this constraint, the quantity can never have a negative value.

9. Each Shopping cart belongs to only one customer

Context ShoppingCart

Inv CarthasOneCustomer ShoppingCart.allInstances -> forAll (SC1, SC2 | SC1.OrderNumber <>SC2.OrderNumber implies SC1.customerID <> SC2.customerID)

Explanation:

According to this constraint, a shopping cart belongs to only one customer. It cannot belong to multiple customers.

10. In search Low price should be less than High Price

Context Search

Inv PriceCompare self. bookLowCost < self. bookHighCost

Explanation:

According to this constraint, while searching for a book, the low price should always be lesser than the high price.

11. Price should always be a positive value

Context Book

Inv BookPricePositive: self.price > 0

Explanation:

According to this constraint, the price associated with the book can never be a negative value.

12. Customer age for registration should not be less than 15 years and should not be more than 85 years

Context Customer

Inv: self.customer_age>=15 and self.customer_age <= 85

Explanation

According to this constraint, customer can only be registered if he is the age group of 15 – 85 else he should not be registered.

References

[1] http://www0.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-03-04/OCL.pdf
[2] http://modeling-languages.com/ocl-tutorial/
[3] https://st.inf.tu-dresden.de/files/general/OCLByExampleLecture.pdf
[4] http://www.sciencedirect.com/science/article/pii/S131915781200002X
[5] http://creately.com/diagram/hfqgfy63/Online%20Book%20Store%20Class%20Diagra