UNIVERSITÉ Concordia UNIVERSITY

MODEL DRIVEN SOFTWARE ENGINEERING

COEN 6312, WINTER 2016

DELIVERABLE: 3

CLASS DIAGRAM

Submitted By:

Desai Parth Mukeshbhai 27397364

Jarin Manuvel Mathew 27168470

Manjot Singh Gill 27299699

Vivek Khatri 27292848

Dollar Kumar Bansal 27599188

Contents

Objective

In this document we explain the class diagram, which further explains and helps in the development of code for our inventory management system, along with the constraints in the model. The class diagram helps in lowering the representational gap between the system design and coding. OCL further enhances the class diagram by adding a set of predefined constraints and structured rules which help in governing the system.

Class Diagram

A UML class diagram is a diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships between objects. A Class is abstraction of objects. All of the objects have the same characteristics (attributes) and behave the same way, and all objects are subject to and conform to the same rules and policies. There are several ways to represent a class depending on whether the attributes and operations are displayed or not, we have used the below representation to represent our classes and have used papyrus in eclipse to create our class diagram. [1][2]

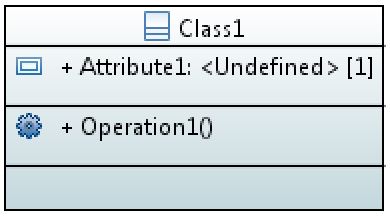

Figure 1: Sample Class in Papyrus

In this representation the class name is mentioned on the top followed by attribute its type and multiplicity and then the methods of the class.

We have also used enumeration classes which helps to choose a set of enumeration literal which we have created for various attributes of other classes. Below structure is used to represent an enumeration class.

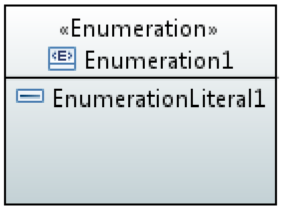

Figure 2: Sample enumeration class in Papyrus

UML Class Diagram for Inventory Management System

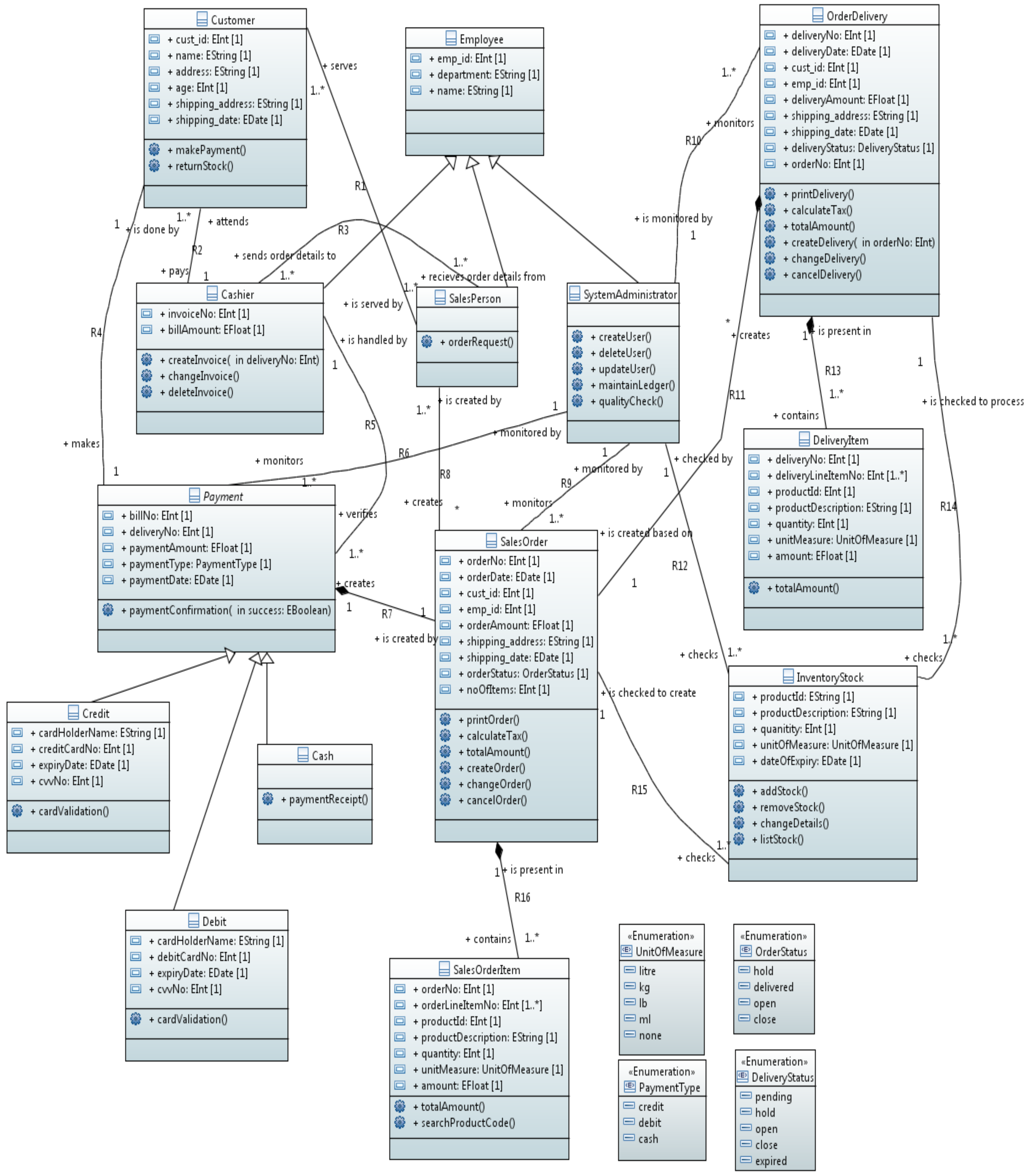

Figure 3: Class Diagram for Inventory Management System

Class Customer

Customer class is the class which is used to represent the customer details of the inventory management system. It can request salesperson to create sales order and also performs payment for the item purchased.

Attributes

- cust_id: Unique id of the customer to identify the customer during the entire process
- name: Name of the customer
- address: Demographic details of the customer
- age: age of the customer
- shipping_address: shipping address of the customer to which the item is to delivered, it may or may not be same as address.
- shipping_date: date on which customer wants the item to be shipped from the warehouse

Methods

- makePayment: This method is called when customer decides to make payment and when an invoice is generated by cashier
- returnStock: This method is called when customer wants to return or replace the item if it has any defects

Class Payment

Once the items are ordered then the salesperson sends the information to cashier about the payment. Payment class records the payment completed by the customer. It is also super class for credit, debit and cash.

Attributes

- billNo: unique number used to represent the payment performed by customer and it is same as invoice number in class cashier
- deliveryNo: it is a unique number, and it is created based the order request created by the salesperson
- paymentAmount: total amount which is payable by the customer in order to purchase the item which is inclusive of taxes
- paymentType: customer can choose payment type between credit, debit and cash from paymentType enumeration class
- paymentDate: Date on which the customer makes the payment.

Methods

- paymentConfirmation: Method used to determine

Class Credit

Credit class is one of the inherited class from payment which is one the mode selected by the customer to make payment for the item purchased.

Attributes

- cardHolderName: This attribute is used to input the name of the card holder
- creditCardNo: 16 digit credit card number of the card holder
- expiryDate: expiry date of the credit card
- cvvNo: 3 digit security code of the card

Methods

- cardValidation: method called to validate the card details

Class Debit

Debit class is another inherited class from payment which is one the mode selected by the customer to make payment for the item purchased.

Attributes

- cardHolderName: This attribute is used to input the name of the card holder
- creditCardNo: 16 digit debit card number of the card holder
- expiryDate: expiry date of the debit card
- cvvNo: 3 digit security code of the card

Methods

- cardValidation: method called to validate the card details

Class Cash

Cash class is the third class inherited from payment which is one the mode selected by the customer to make payment for the item purchased. It does not have any attributes as it uses only the inherited attributes from payment class.

Methods

- paymentReciept: method called once the customer chooses to pay the amount by cash

Class Employee

Employee class is the super class for cashier, salesperson, systemadministrator. This class is used to represent the type of employee of the company.

Attributes

- emp_id: unique identification number used for each employee
- department: department details under which the employee is working
- name: name of the employee

Class Cashier

Cashier class is an inherited class from employee which is used to represent the employee who handles the payment made by the customer. Cashier receives the payment information from salesperson. I

Attributes

- invoiceNo: unique number created when the user decides to make the payment
- billAmount: total amount of the items including taxes

Methods

- createInvoice: method called to create the invoice details of the purchase
- changeInvoice: method called when cashier decides to change the details of the invoice which is already created
- deleteInvoice: method called to delete already existing invoice

Class SalesPerson

SalesPerson class is used to represent the employee who handles the sale of the item. Salesorder is created by salesperson and salesperson serves all the customers who wants to make the purchase of items. It is one of the inherited class from employee. This class does not have any attributes of its own and hence it inherits all the attributes from its super class.

Methods

- orderRequest: method called to request to create the sales order

Class SystemAdministrator

SystemAdministrator class is used to represent the employee who handles the responsibilities of creating various users, deleting users, update user details. System administrator also monitors various processes like Payment, SalesOrder, OrderDelivery and also checks the InventoryStock. It is also one of the inherited class from employee.

Methods

- createUser: method called to create a new user in the system
- deleteUser: method called to delete any user from the system
- updateUser: method called to update the user details in the system
- maintainLedger: this method is used to generate various reports
- qualityCheck: this method is called when the administrator needs to check the quality of an item in the inventory

Class InventoryStock

This class is used to record all the details related to the item in the inventory. It has the product ID, description, quantity present in the warehouse and its expiry information. Inventory details are checked while creating an order and also after making the delivery.

Attributes

- productID: unique identification number for each product in the inventory
- productDescription: information about the product in the inventory
- quantity: total number of a particular item in the inventory
- unitOfMeasure: attribute used to identify the type of measurement for the product. It can litre, kg, lb, ml and none.
- dateOfExpiry: it is the expiry date for a given product

Methods

- addStock: method called when a new item is added to the stock or when the existing stock of an item is updated
- removeStock: method called an item is removed from the inventory

- changeDetails: method called when administrator chooses to update the details of an item
- listStock: method to get the list of all the items in the inventory

Class SalesOrder

This class is used to create, change and update sales order. Sales order class keeps the information about customer shipping date, shipping address which is same as in delivery and customer class. It also contains the sales order status during the processing and updates its status as further processed. The information in sales order cannot be changed after sales order is completed.

Attributes

- orderNo: unique number of sales order
- orderDate: sales order date represents the date in which sales order is created.
- cust_id: Unique id of the customer to identify the customer during the entire process
- emp_id: unique identification number used for each employee
- orderAmount: total amount of the sales order which includes all taxes
- shipping_address: shipping address of the customer to which the item is to delivered, it may or may not be same as address.
- shipping_date: date on which customer wants the item to be shipped from the warehouse
- orderStatus: it gives the status of the order at that time. It contains the enumeration values like, hold, delivered, close, open
- noOfItems: it gives the total number of item available in sales order.

Methods

- printOrder(): This method print the sales order number with all important details.
- calculateTax(): This method calculate all taxes .
- totalAmount(): This method counts the total amount of sales order which includes all taxes
- createOrder(): This method is called when sales person invokes an operation to create sales order.
- changeOrder(): This method is called when sales person wants to change the details in sales order.
- cancelOrder(): this method is called when sales person want to cancel the sales order and don't want to process further.

Class SalesOrderitem

This class contains the sales order line item. In sales order each line item contains unique line item number, productId, productDescription, quantity, unitMeasure and amount. Product id can be search while creating the sales order and total amount is calculated.

Attributes

- orderNo: unique number of sales order
- orderLineItemNo: unique number of each line item which is in sequence.
- productID: unique identification number for each product
- productDescription: information about the product in the inventory
- quantity: total number of a particular item ordered by the customer
- unitMeasure: attribute used to identify the type of measurement for the product. It can be litre, kg, lb, ml and none.
- amount: it is the total value of quantity.

Methods

- totalAmount(): This method gives the total value of salesOrderItem which contains the multiple line item.
- searchProductCode(): This method is invokes when sales person searching a particular product.

Class OrderDelivery

This class is used to create, change and update order delivery. To create delivery order it checks the sales order first. The quantity and amount in delivery order is not always same as sales order. It might be possibility of creating the multiple deliveries against the sales order.

Attributes

- deliveryNo: unique number of develiry
- deliveryDate: delivery order date represents the date on which the item should be delivered.
- cust_id: Unique id of the customer to identify the customer during the entire process
- emp_id: unique identification number used for each employee
- deliveryAmount: total amount of the delivery which includes all taxes
- shipping_address: shipping address of the customer to which the item is to delivered, it may or may not be same as address.
- shipping_date: date on which customer wants the item to be shipped from the warehouse
- deliveryStatus: it gives the status of the delivery at that time. It contains the enumeration values like, pending, hold, expired, close, open

Methods

- printOrder(): This method print the delivery order number with all important details.
- calculateTax(): This method calculate all taxes .
- totalAmount(): This method counts the total amount of order delivery which includes all taxes
- createDelivery (int OrderNo): This method is called when sales person invokes an operation to create order delivery. Delivery is created based on order number.
- changeDelivery (): This method is called when sales person wants to change the details in order delivery. This method cannot change the product id and product description.
- cancelDelivery(): this method is called when sales person want to cancel the order delivery and don’t want to process further.

Class DeliveryItem

This class contains the delivery line item. In delivery each line item contains unique line item number, productId, productDescription, quantity, unitMeasure and amount. Delivered quantity can be same as the ordered quantity but not more than the ordered quantity.

Attributes

- deliveryNo: unique number of Delivery
- deliveryLineItemNo: unique number of each line item which is in sequence.
- productID: unique identification number for each product
- productDescription: information about the product in the inventory

- quantity: total number of a particular item delivered to the customer
- unitMeasure: attribute used to identify the type of measurement for the product. It can be litre, kg, lb, ml and none.
- amount: it is the total value of quantity.

Methods

- totalAmount(): This method gives the total value of DeliveryItem which contains the multiple line item.

Enumeration Classes

UnitOfMeasure: UnitOfMeasure classifiers that consist of literal values which are: litre, kg, lb, ml and none.

OrderStatus: These classifiers gives the value of OrderStatus and their values are: hold, delivered, open, close.

PaymentType: PaymentType can be any value of the given literal values which are: credit, debit and cash.

DeliveryStatus: These classifiers gives the value of DeliveryStatus and their values are: hold, pending, open, close, and expired.

OCL Constraints

1) An employee should have unique employee id.

    Context Employee
    inv: self.allInstances->forAll(e1,e2->Employee | e1<>e2 implies e1.emp_id <>e2.emp_id)

2) A new unique customer ID should be created for every new customer

    Context Customer
    inv: self.cust_id->notEmpty() implies self.allInstances -> forAll(c1,c2 ->Customer | c1<>c2 implies c1.cust_id <> c2.cust_id)

3) We cannot exceed the number of items in a sales order more than the no. of those items(quantity) in inventory

    Context SalesOrder
    inv: self.R15.quantity -> size() > self.noOfItems->size()

4) Shipping address should not be empty for OrderDelivery, SalesOrder and Customer

    Context OrderDelivery
    inv: self.shipping_address -> notEmpty()

    Context SalesOrder
    inv: self.shipping_address -> notEmpty()

    Context Customer

inv: self.shipping_address -> notEmpty()

5) Invoice number of a sale generated by cashier should be unique

    Context Cashier

    inv: self.allInstances->forAll(i1,i2->Cashier |i1<>i2 implies i1.invoiceNo <>e2.invoiceNo)

6) Before making the payment, invoice should be generated

    Context Payment

    inv: self.billNo -> notEmpty() implies self.R5.invoiceNo->notEmpty()

7) The current date should be less than the expiry date of the items to be delivered

    Context InventoryStock

    inv: self.dateOfExpiry->isBefore(today)

8) Order Number should be unique.

    Context SalesOrder

    inv: self.allInstances->forAll(o1,o2->SalesOrder |o1<>o2 implies o1.orderNo <>o2.orderNo)

9) Bill number should be unique.

    Context Payment

    inv: self.allInstances->forAll(b1,b2->Payment |b1<>b2 implies b1.billNo <>b2.billNo)

10) Delivery Number should be unique

    Context OrderDelivery

    inv: self.allInstances->forAll(d1,d2->OrderDelivery |d1<>d2 implies d1.deliveryNo <>d2.deliveryNo)

11) Product ID should be unique

    Context InventoryStock

    inv: self.allInstances->forAll(p1,p2->InventoryStock |p1<>p2 implies p1.productId <>p2.productId)

12) Bill amount paid by the customer should be equal to the payment amount in payment

    Context Customer

    inv: self.R2 ->forAll(self.R2 ->notEmpty() implies self.R2.billAmount = self.R2.R5.paymentAmount)

13) Orderstatus should be "hold" until the payment is successful when the sales order has been generated

    Context Payment :: paymentConfirmation(success: Boolean)

    post: if(success = false)

    then self.R7.orderStatus = 'hold'

    endif

14) The shipping date in SalesOrder and Order delivery should not be less than the current date

Context SalesOrder
inv: self.shipping_date ->isAfter(today)

Context OrderDelivery
inv: self.shipping_date ->isAfter(today)

15) A delivery cannot be cancelled or changed after the delivery status = "close"

Context OrderDelivery::changeDelivery()
pre: self.deliveryStatus <> 'close'

context OrderDelivery::cancelDelivery()
pre: self.deliveryStatus <> 'close'

16) Shipping Address of the customer should be match with shipping address of the sales order.

Context Customer
inv: self.R1.R8 -> size() = 1 implies self.cust_id = self.R1.R8.cust_id and self.shipping_address = self.R1.R8.shipping_address

References

[1] Class Notes Dr. Abdel Wahab on OCL which is available on Concordia moodle

[2] https://en.wikipedia.org/wiki/Class_diagram

[3] http://wiki.eclipse.org/Papyrus_User_Guide