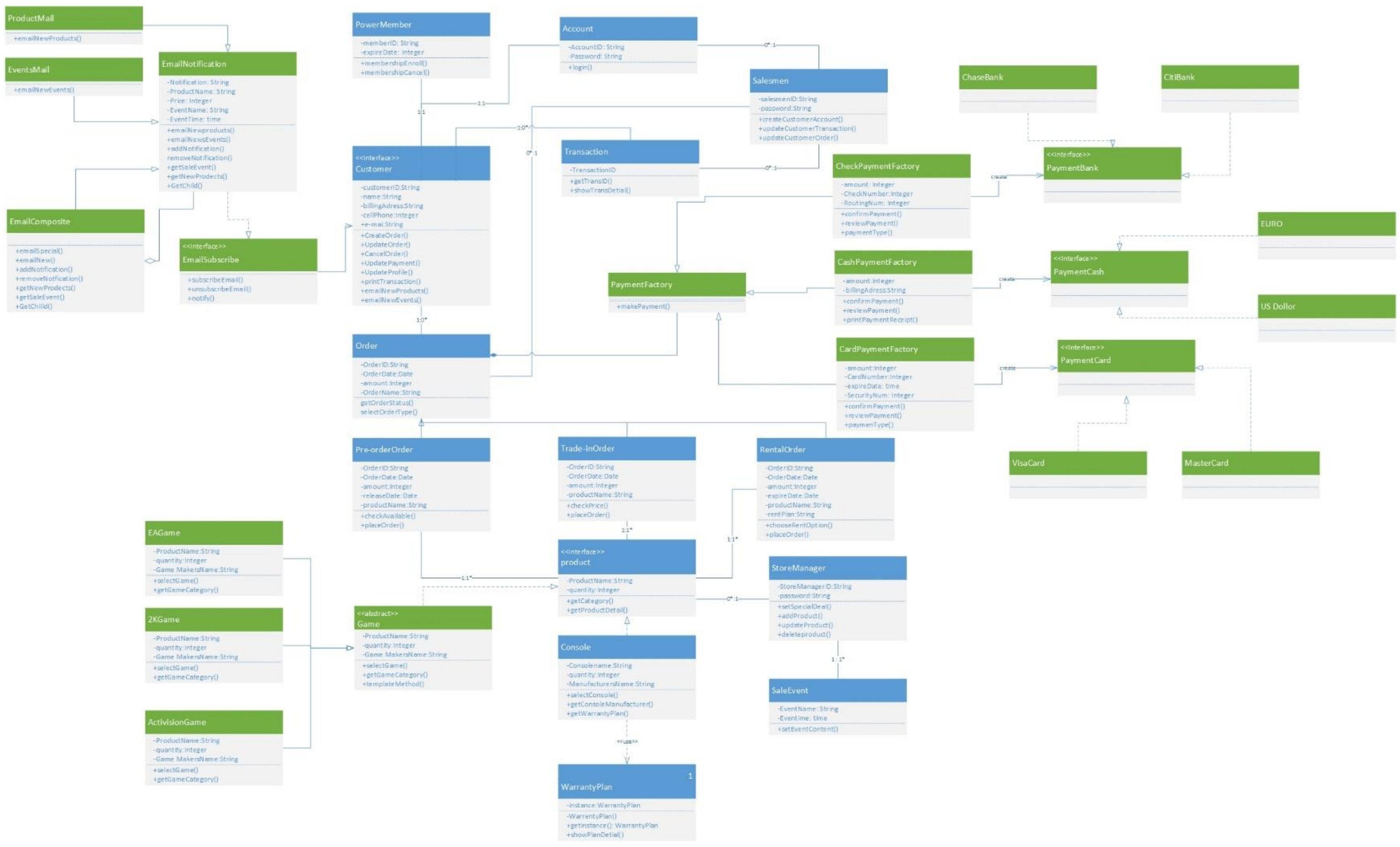
Software Modeling Dev with UML
CSP- 586
Assignment #4

Chen Xu A20377739

1. Design Model class diagram



2. List of the Design pattern(s)

A. Assignment 3:

i. Singleton Design Pattern

ii. Factory Method Design Pattern

iii. Observer Design Pattern

B. Assignment 4:

i. Abstract Factory Design Pattern

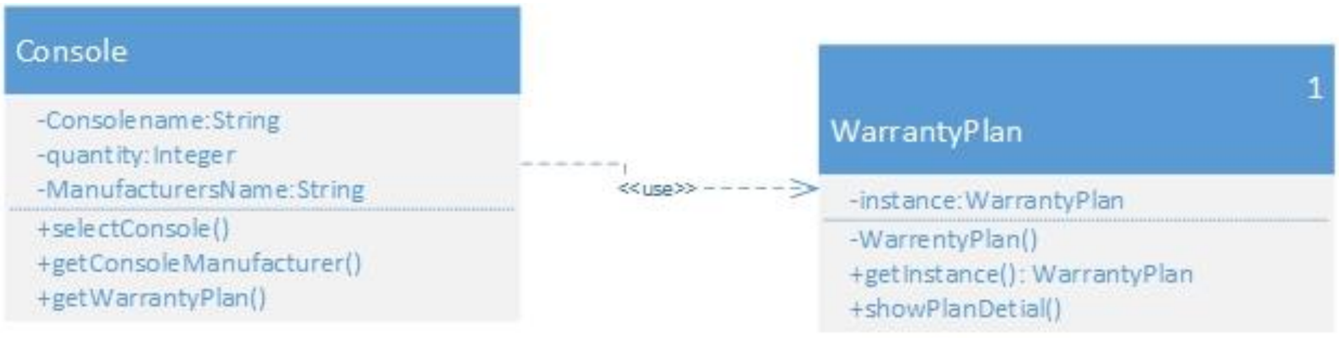
ii. Composite Design Pattern

iii. Template Method Design Pattern

3. Documentation of used design patterns

A. Assignment 3:

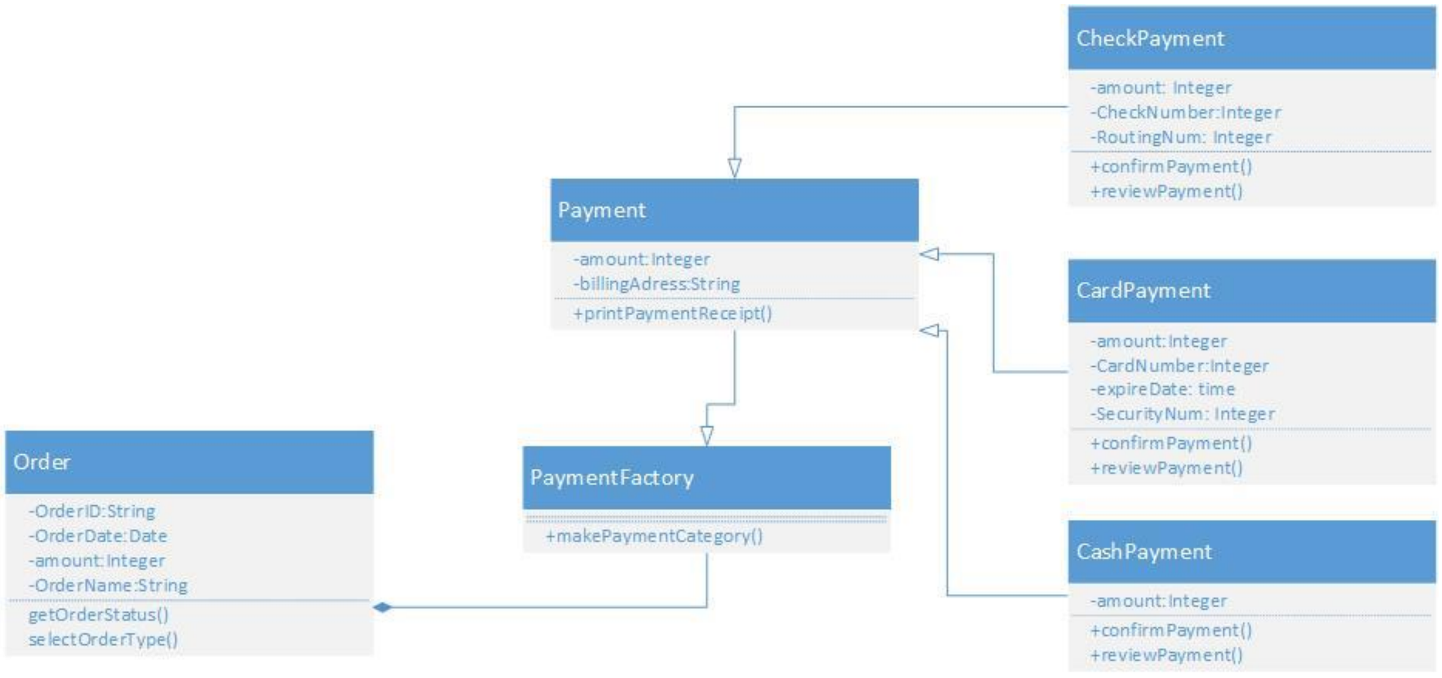
1) Singleton Design Pattern: Implement select console Warranty plan.



I. Use singleton Pattern to make every console use the instance that instantiated by WarrantyPlan Class.

II. WarrantyPlan has Instance in the Class and Console can get it by using getInstance() method.

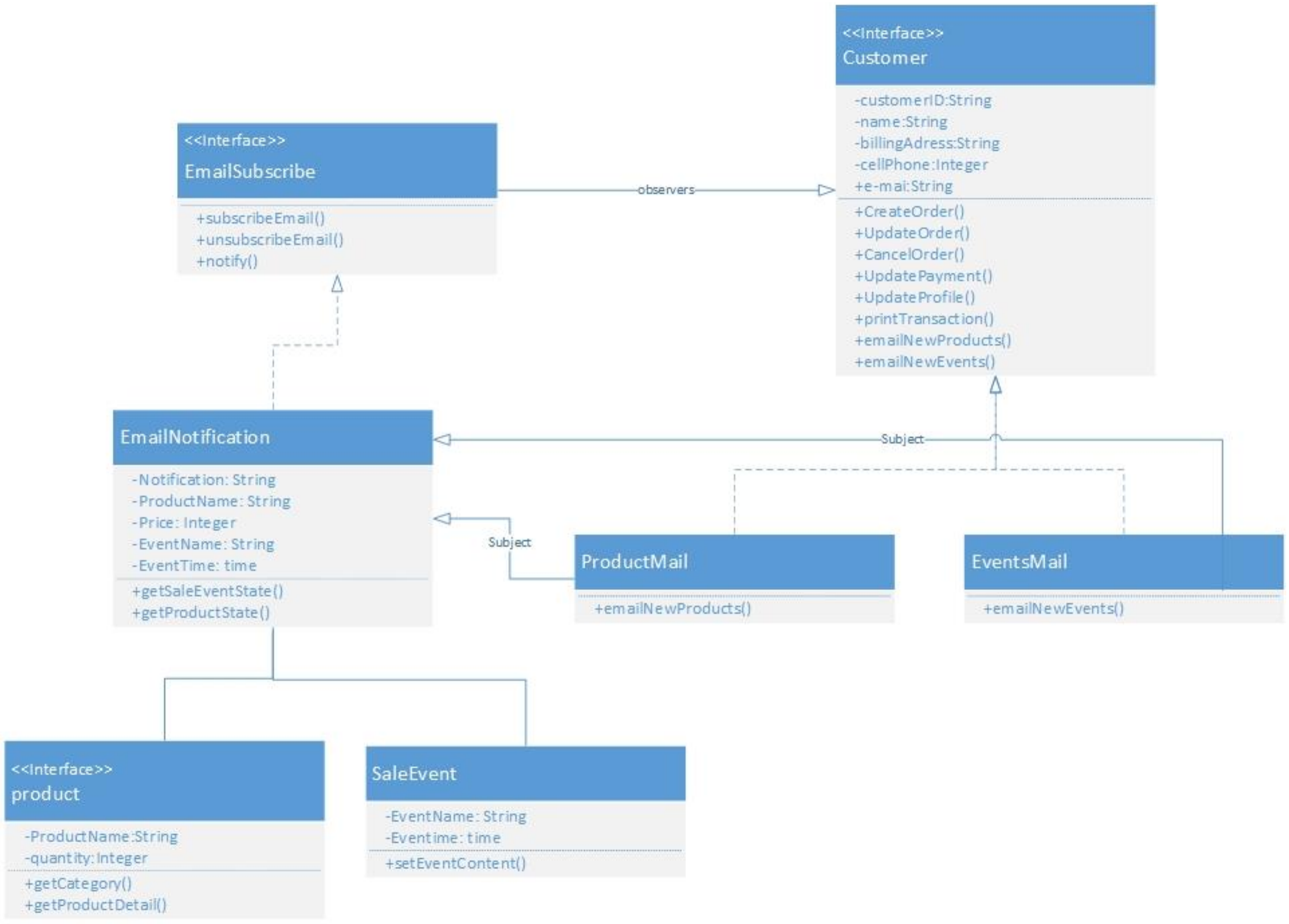
2) Factory Method Design Pattern: Implement Order make payment.



I. Order get payment instance through PaymentFactory Class and use the payment category of the Order to instance the payment in right way.

II. CheckPayment , CardPayment and CashPayment Class are the different category of payment that inherit from the Payment Class.

3) Observer Design Pattern: Implement email subscriptions.



I. the customer will subscribe or unsubscribe to get the notification of new products and events .

II. The Customer is the observer interface, it can use ProductMail and EventsMail Class to get new product and event notification.

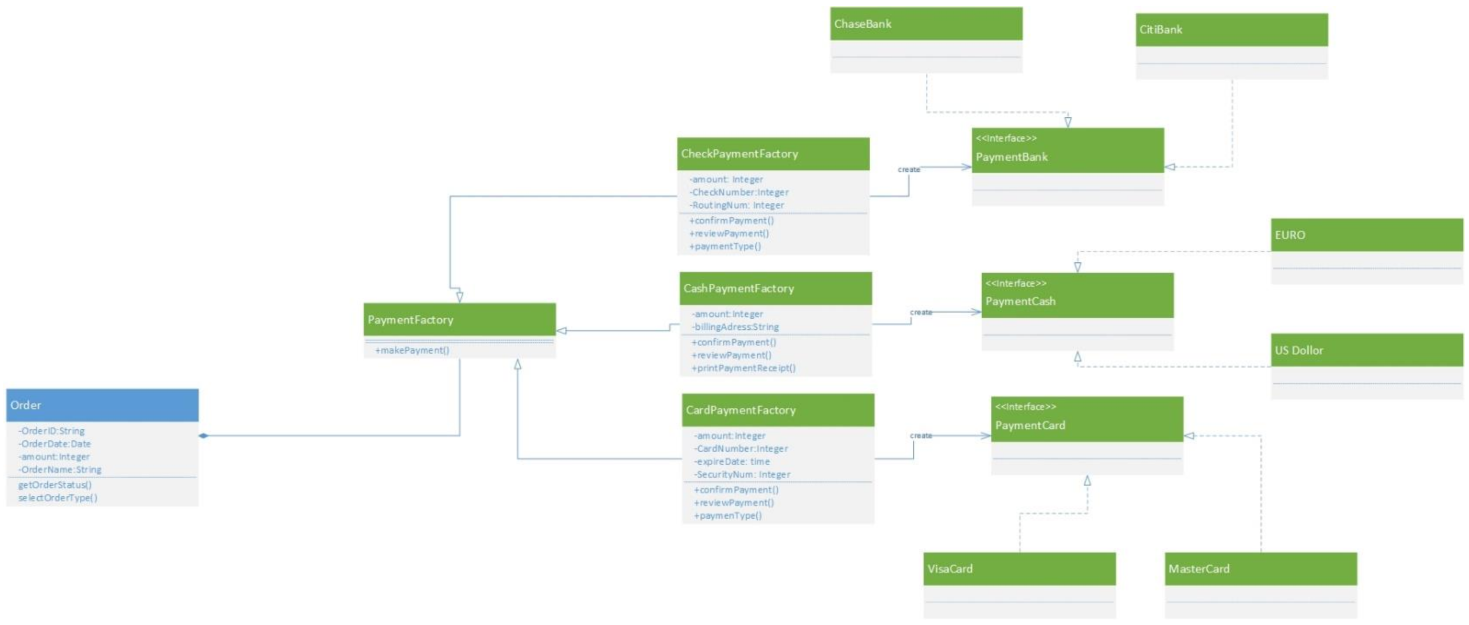
III. The EmailNotification Class implement the subject interface EmailSubscribe.

IV. ProductMail and EventMail Class are subclass of EmailNotification Class.

V. EmailNotification Class can get state of information from the product and SaleEvent Class.

B. Assignment 4:

1) Abstract Factory Design Pattern: Implement make payment.

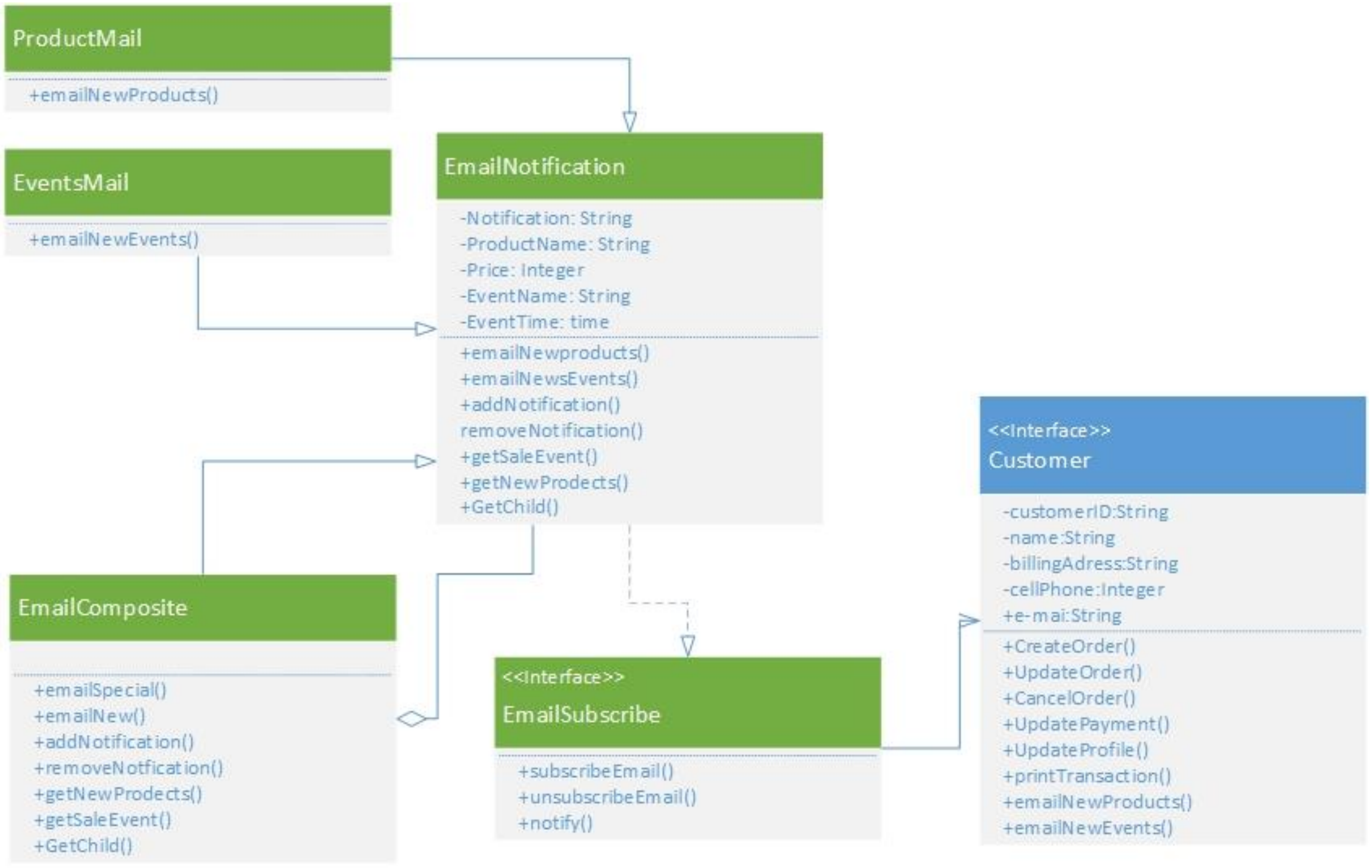


I. Use an abstract PaymentFactory Class to choose and create payment from individual factories.

II. CardPaymentFactory, CashPaymentFactory and CardPaymentFactory these concrete subClasses create a family of payments for each type.

III. PaymenBank, PaymentCash and PaymentCard are the interface of each payment types create parallel sets of their payment families.

2) Composite Design Pattern: Implement Email subscibptions.



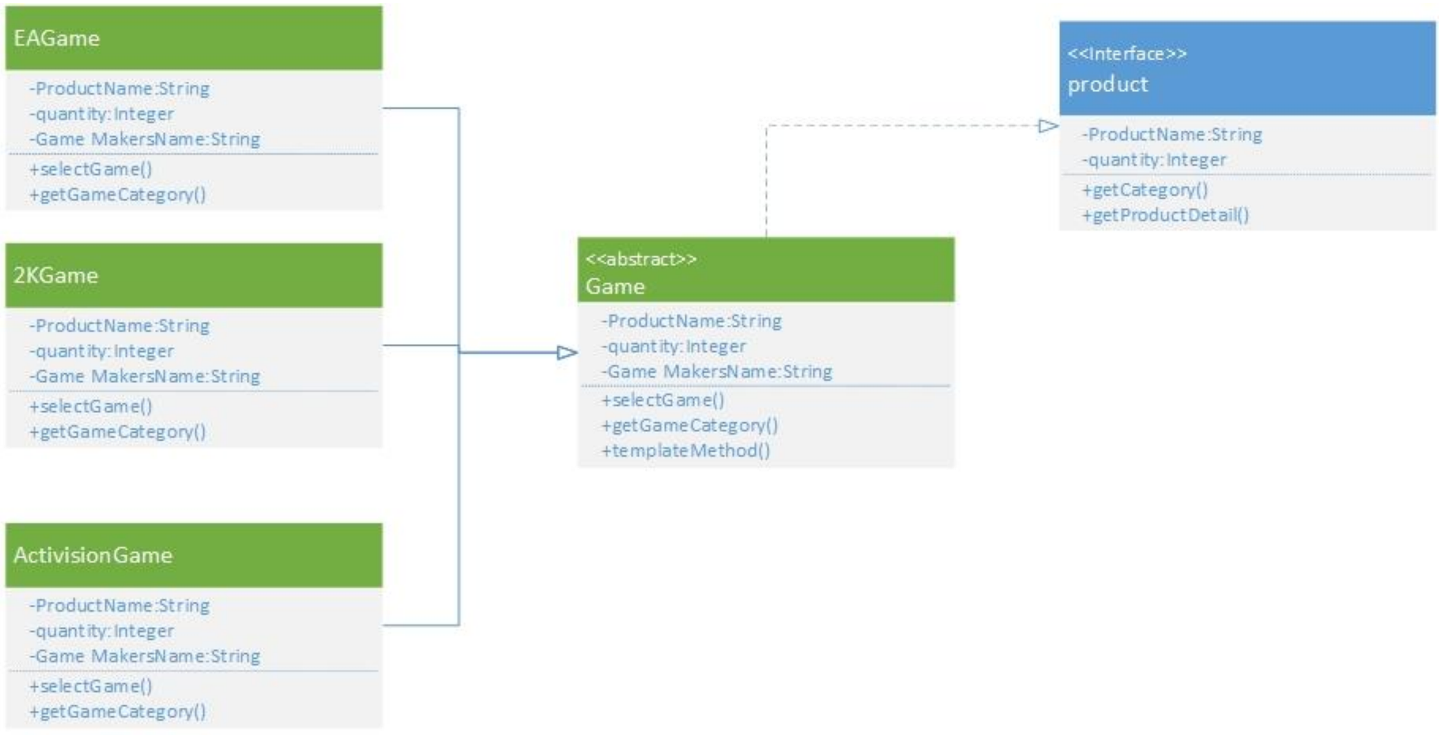
I. Use the EmailSubscibe Class as interface to manipulate the objects in the composition

II. Use EmailComposite Class to define the behavior of the components having children and to store child components. It implements the child related operations.

III. EmailNotification Class is the is the abstraction for all components, including EmailComposite Class. It declares the interface for objects in the composition.

IV. ProductMail and EventMail are the leaf Classes, they are the elemnts to help implement the composition.

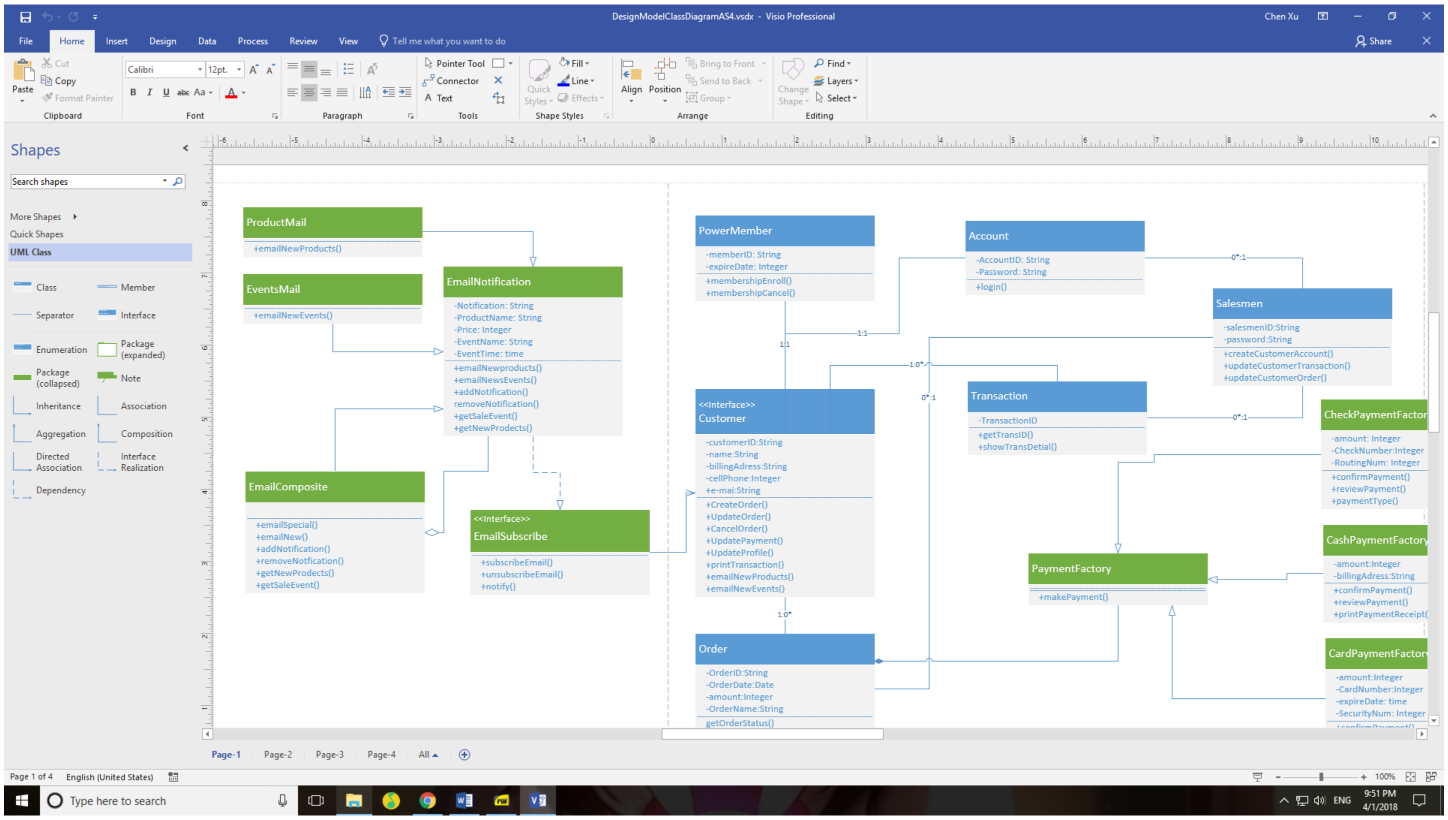
3) Template Method Design Pattern

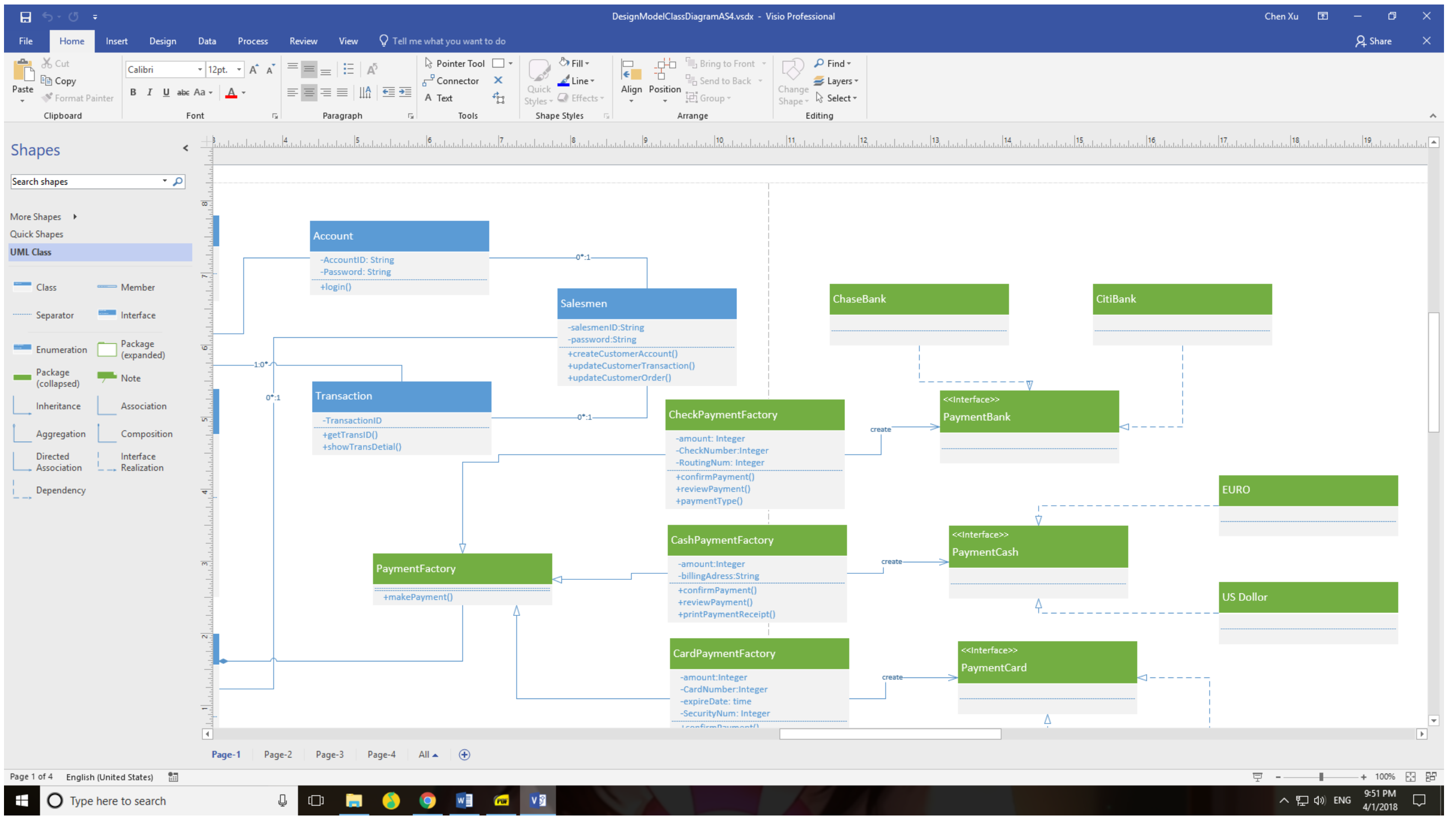


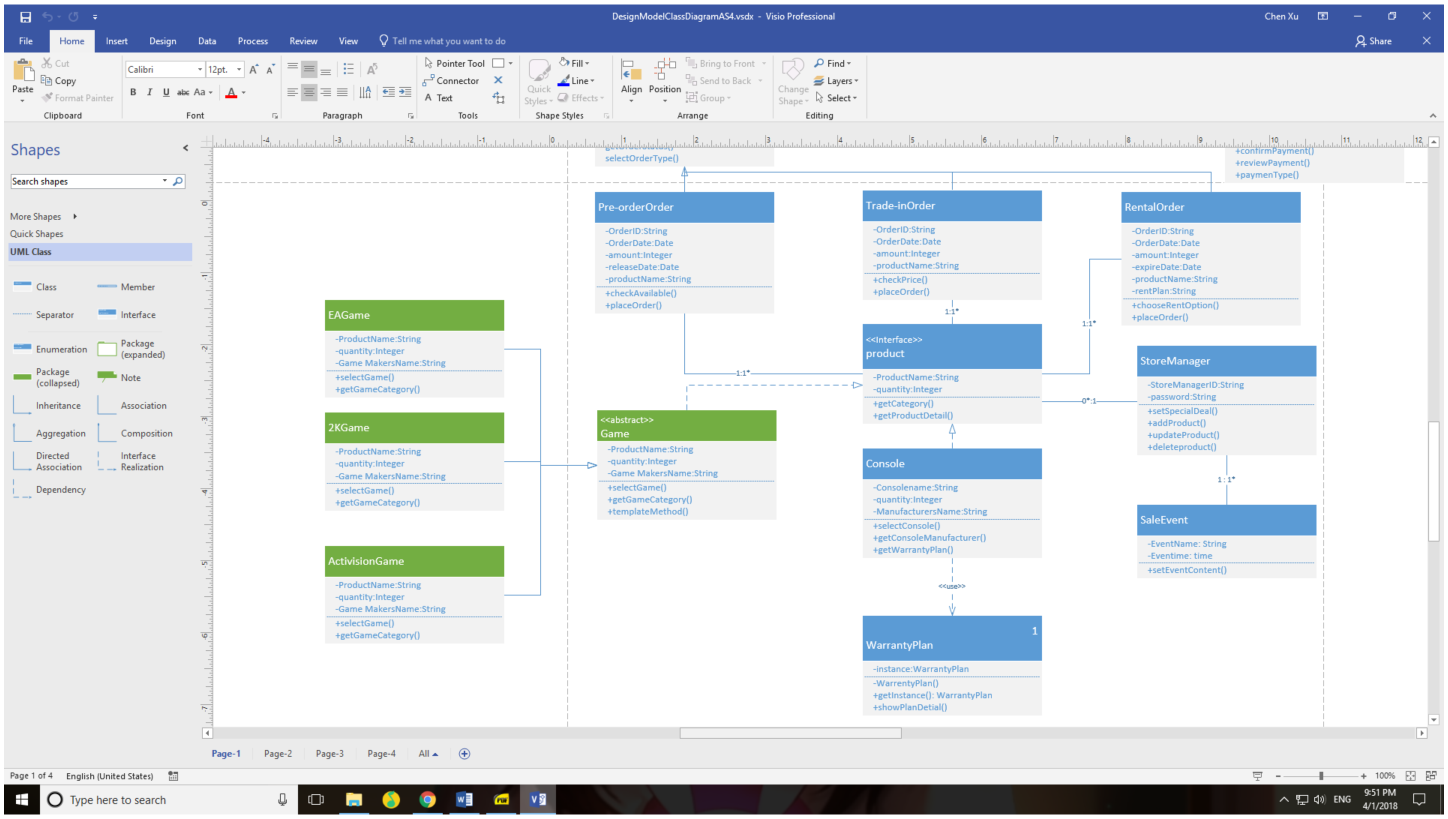
I. the Game Class defines a templateMethod() operation that defines the template of a behavior by implementing the invariant parts to each subClasses.

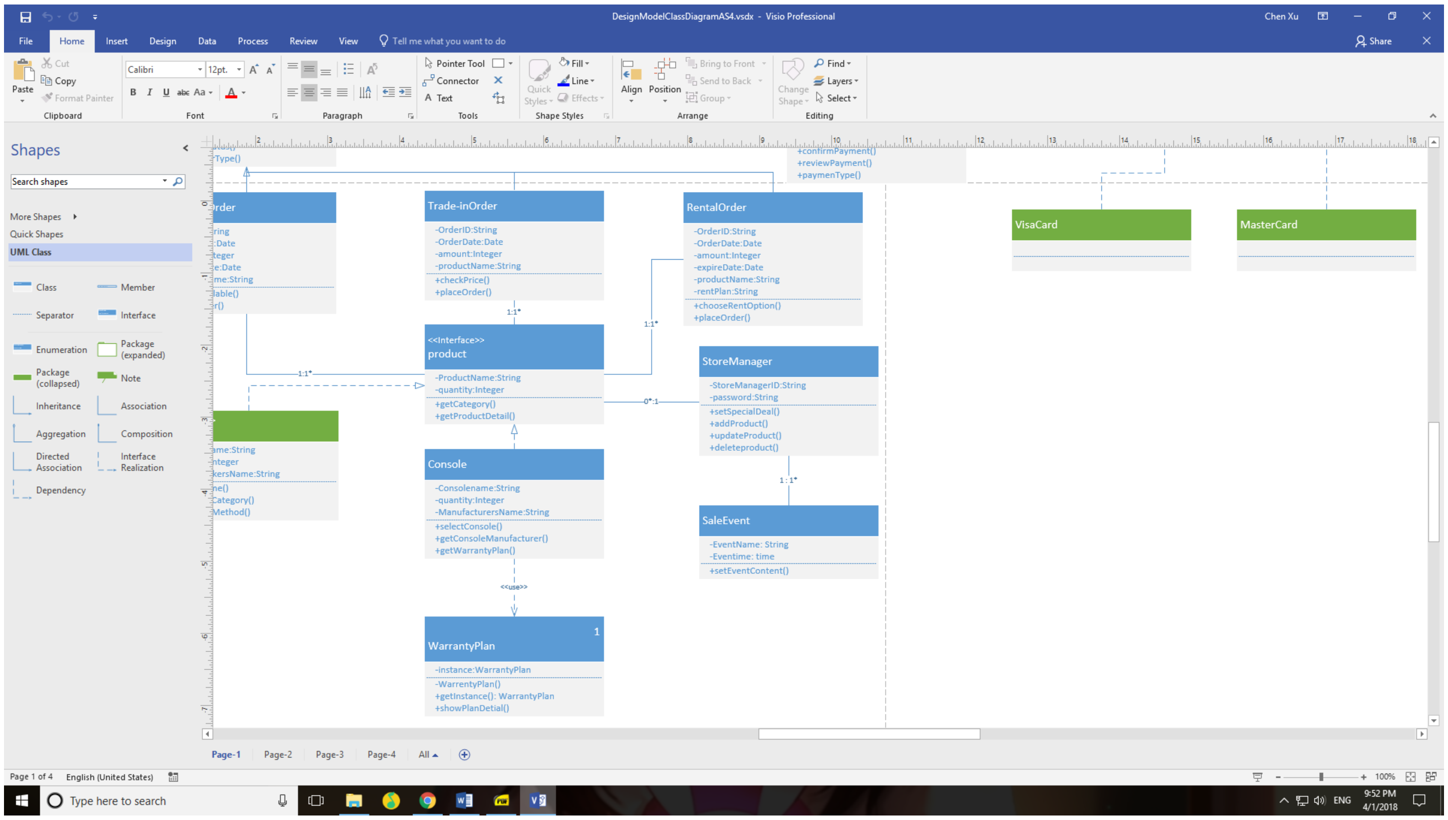
II. EAGame, 2KGame and ActivisionGame are subclasses that have defer part . They help template class to instantiated different category instances.

4. Capture design model class diagram(s)






DesignModelClassDiagramAS4.vsdx - Visio Professional
selectOrderType()
+confirmPayment()
+reviewPayment()
+paymenType()
Pre-orderOrder
-OrderID:String
-OrderDate:Date
-amount:Integer
-releaseDate:Date
-productName:String
+checkAvailable()
+placeOrder()
Trade-inOrder
-OrderID:String
-OrderDate:Date
-amount:Integer
-productName:String
+checkPrice()
+placeOrder()
RentalOrder
-OrderID:String
-OrderDate:Date
-amount:Integer
-expireDate:Date
-productName:String
-rentPlan:String
+chooseRentOption()
+placeOrder()
EAGame
-ProductName:String
-quantity:Integer
-Game MakersName:String
+selectGame()
+getGameCategory()
2KGame
-ProductName:String
-quantity:Integer
-Game MakersName:String
+selectGame()
+getGameCategory()
ActivisionGame
-ProductName:String
-quantity:Integer
-Game MakersName:String
+selectGame()
+getGameCategory()
<<abstract>>
Game
-ProductName:String
-quantity:Integer
-Game MakersName:String
+selectGame()
+getGameCategory()
+templateMethod()
<<Interface>>
product
-ProductName:String
-quantity:Integer
+getCategory()
+getProductDetail()
1:1*
1:1*
1:1*
0*:1
StoreManager
-StoreManagerID:String
-password:String
+setSpecialDeal()
+addProduct()
+updateProduct()
+deleteproduct()
1 : 1*
SaleEvent
-EventName: String
-Eventime: time
+setEventContent()
Console
-Consolename:String
-quantity:Integer
-ManufacturersName:String
+selectConsole()
+getConsoleManufacturer()
+getWarrantyPlan()
<<use>>
WarrantyPlan
1
-instance:WarrantyPlan
-WarrentyPlan()
+getInstance(): WarrantyPlan
+showPlanDetial()
Page-1
Page-2
Page-3
Page-4
All
Page 1 of 4
English (United States)
100%
9:51 PM
4/1/2018


DesignModelClassDiagramAS4.vsdx - Visio Professional
+confirmPayment()
+reviewPayment()
+paymenType()
VisaCard
MasterCard
Trade-inOrder
RentalOrder
<<Interface>>
product
StoreManager
SaleEvent
Console
WarrantyPlan
<<use>>
1:1*
1:1*
1:1*
0*:1
1 : 1*
Page-1
Page-2
Page-3
Page-4
All
Page 1 of 4
English (United States)
100%
9:52 PM
4/1/2018