Software Modeling Dev with UML
CSP- 586
Assignment #4

Sivasenthil Namachivayan
A20391478

Table of Contents

Deliverable 1

UML Design Model/ Class Diagram

Deliverable 2

List of Design Patterns Used in Assignment 3

Factory Method Design Pattern
Observer Design Pattern

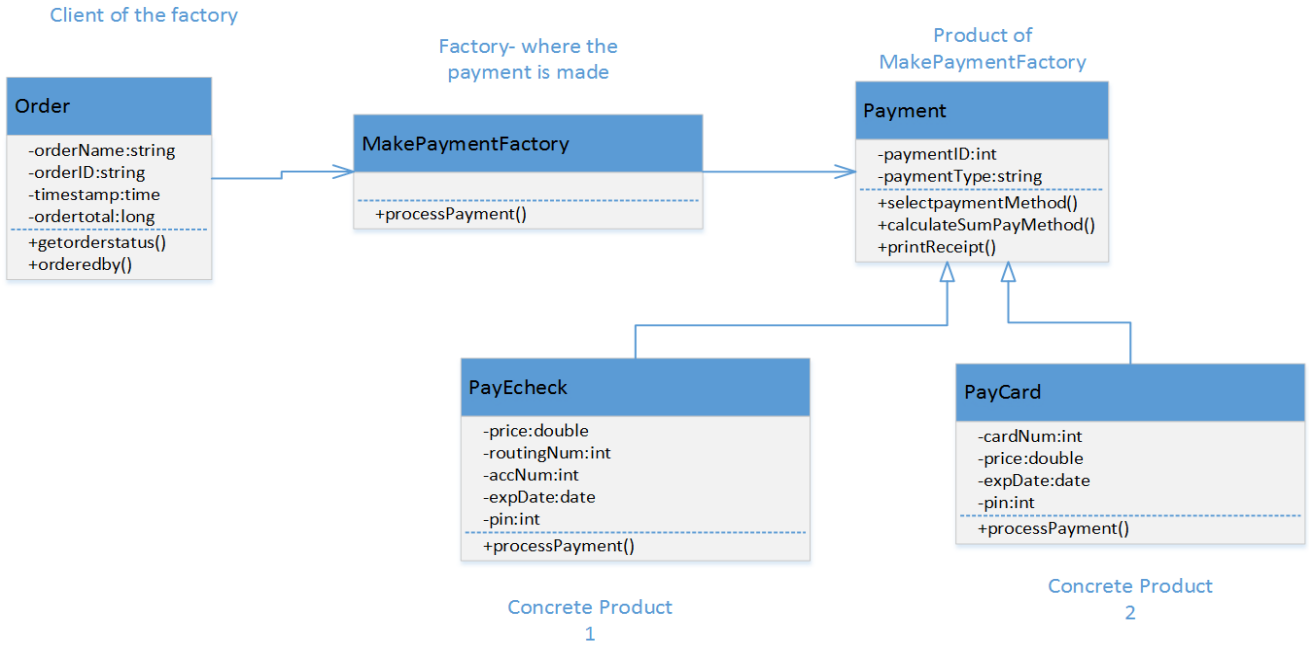
List of Design Patterns Used in Assignment 4

Abstract Factory Pattern
Composite Pattern
Iterator Pattern

Deliverable 3

Documentation of Design Pattern used in the Design Class Diagram in assignment 3.

Factory Method Design Pattern- **Implemented at Payment Process**



Class Order

This is the client of the factory. Order now goes through the *MakePaymentFactory* Class to get the instance of Payment

Class MakePaymentFactory

This class is declared with the *processPayment* method statically. This is the factory where the customer process payment. This is the only class referring to the concrete class *Payment*

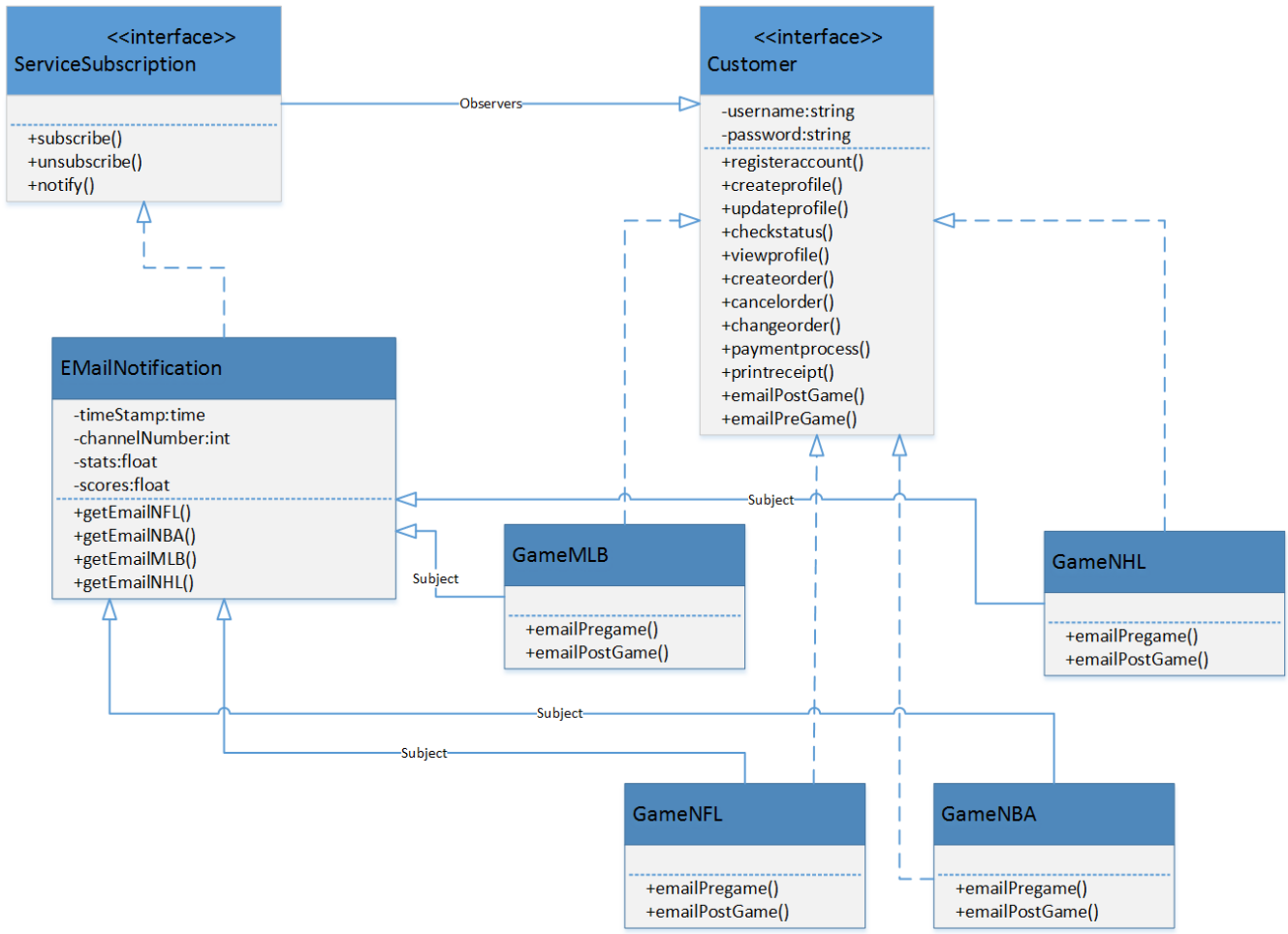
Class Payment

This is the product of the factory *MakePaymentFactory.* Here the Payment is defined as an abstract class

Classes PayEcheck & PayCard

These are the concrete products. Each Product needs to extend the abstract class *Payment* and be concrete. In this way it can be created by the factory and will be provided to the client when ever needed.

Observer Design Pattern- **Implemented at Email Notification**



Class ServiceSubscription

This is the subject interface in which the observers will subscribe or unsubscribe to get the notification of of various games. This is common for all the other services provided by the vendor and if the Email notification is subscribed the customer will get the Email pre and post games at free of cost where the checkout process will have $0 in the cart.

Class Customer

The Customer is the observer, where he can subscribe or unsubscribe for an Email notification.

Class EmailNotification

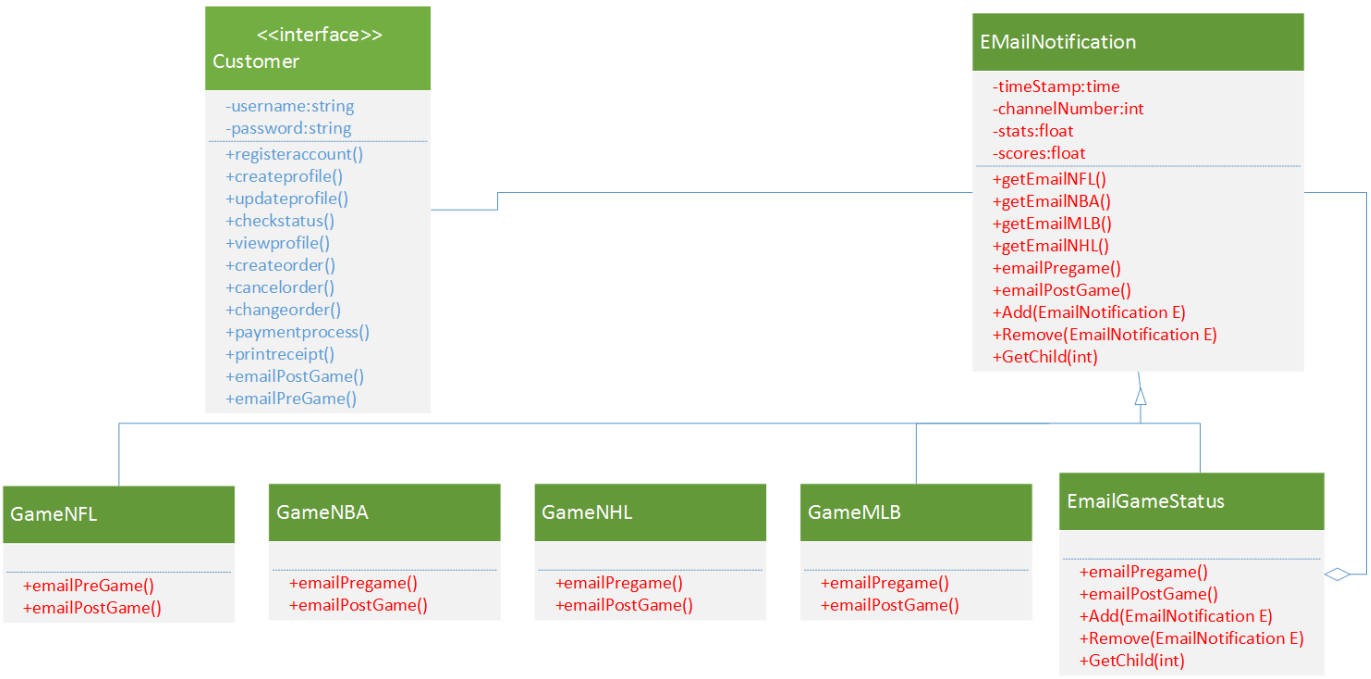
This class implement the subject interface *ServiceSubscription.*

Classes GameMLB, GameNHL, GameNFL and GameNBA

These are the sub classes of the super class *EmailNotification.*

Documentation of Design Pattern used in the Design Class Diagram in assignment 4

Composite Pattern- Implemented at Email Notifications



Class Customer - (Client)

Uses the EmailNotification(Component) interface to manipulate the objects in the composition

Class EmailNotification - (Component)

This Defines an interface for all the objects in the composition both the composite(EmailGameStatus) and leaf nodes(GameNFL,GameNBA, GameNHL, GameMLB). This component may implement a default behavior for Add(), remove(),getchild() and its operations

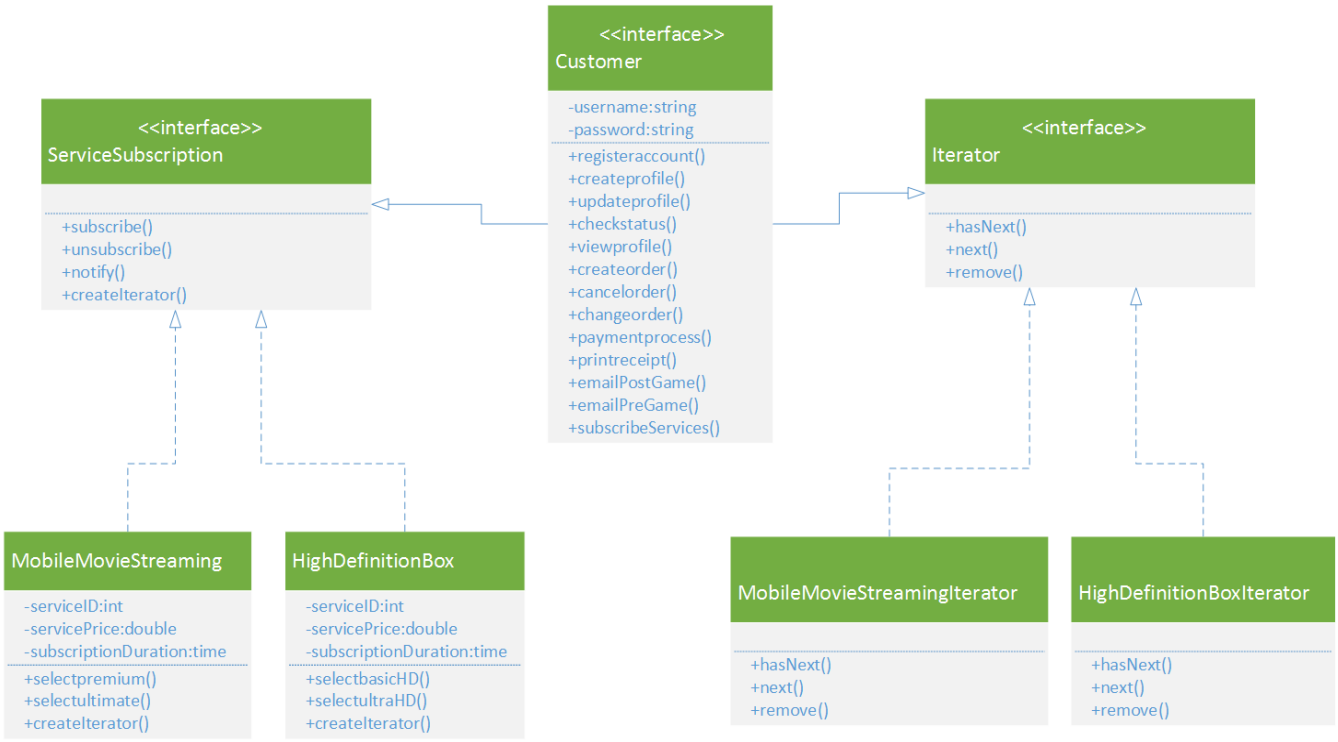
Class GameNFL, GameNBA, GameNHL and GameNLB(Leaf)

These leaves have no children. A Leaf defines the behaviour for the elements in the composition. This is done by the implementation of operations the class EmailGameStatus(composite) supports

Class EmailGameStatus

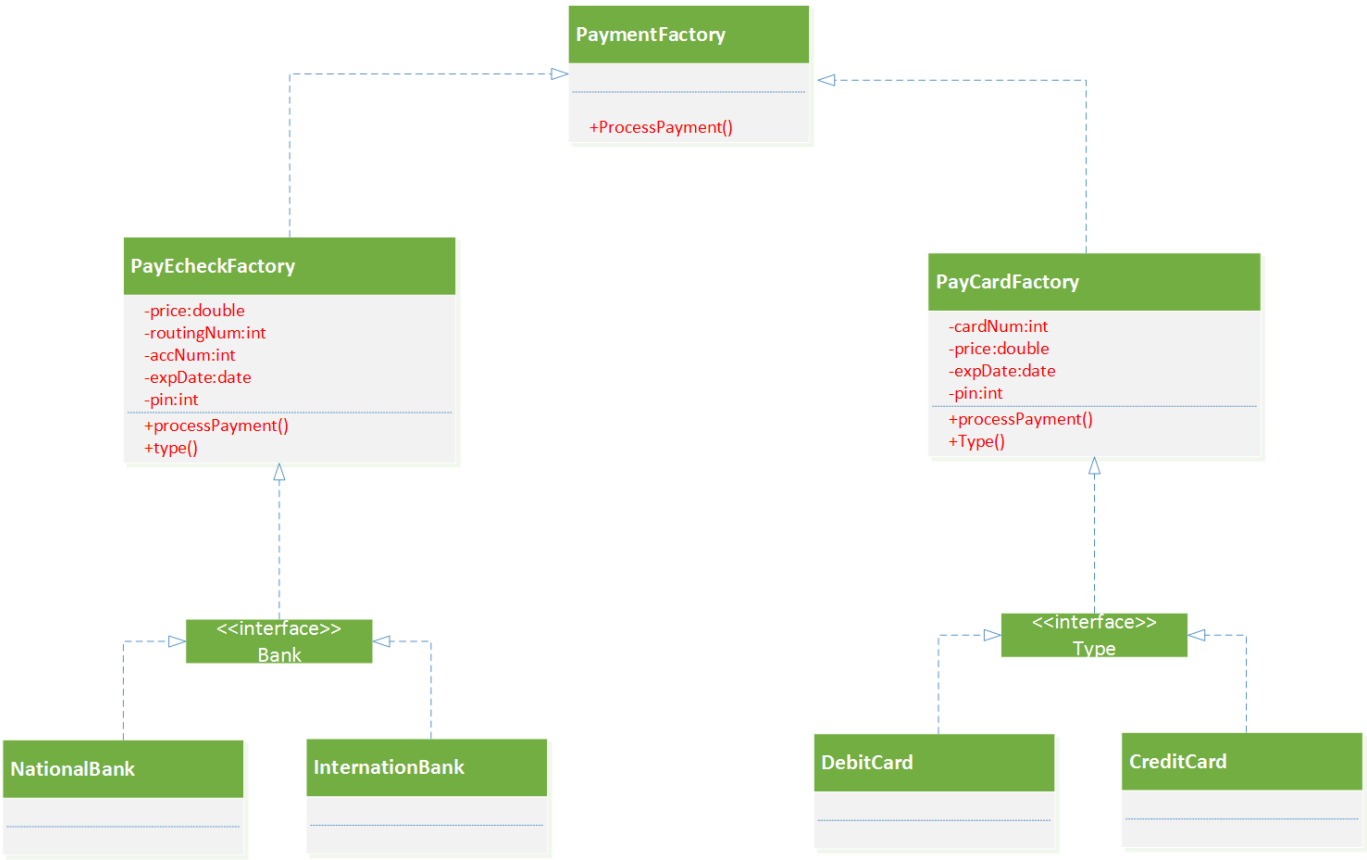
Its role is to define the behaviour of the components having children and to store child components. This implements the Leaf related operations.

Iterator Pattern- Implemented at 2 Services provided by the vendor



Class ServiceSubscription
This is the new ServiceSubscription interface. It specifies the new method createIterator().
Class Customer
Now Customer needs to be concerned about the Service Subscriptions and Iterators
Class Iterator
The Customer is been decoupled from the implementation of servicesubscriptions, so now we can use this iterator to iterate over any list of menu items without having to know about how the list of services are implemented
Class MobileMovieStreaming and Class HighDefinitionBox
These 2 classes not implement the servicesubscription interface, which means they need to implement the new createIterator() method. Here each concrete service is responsible for create their appropriate Service Iterators
Class MobileMovieStreamingIterator and Class HighDefinitionBoxIterator
Here classes returns respective iterator from its createiterator() method because that is the kind of iterator required iterate over its array of various services offered by these Servicesubscriptions.

AbstractFactory Pattern- Implemented at ModeOfPayment



Class PaymentFactory
This class provides provides an abstract interface for creating a family of payments

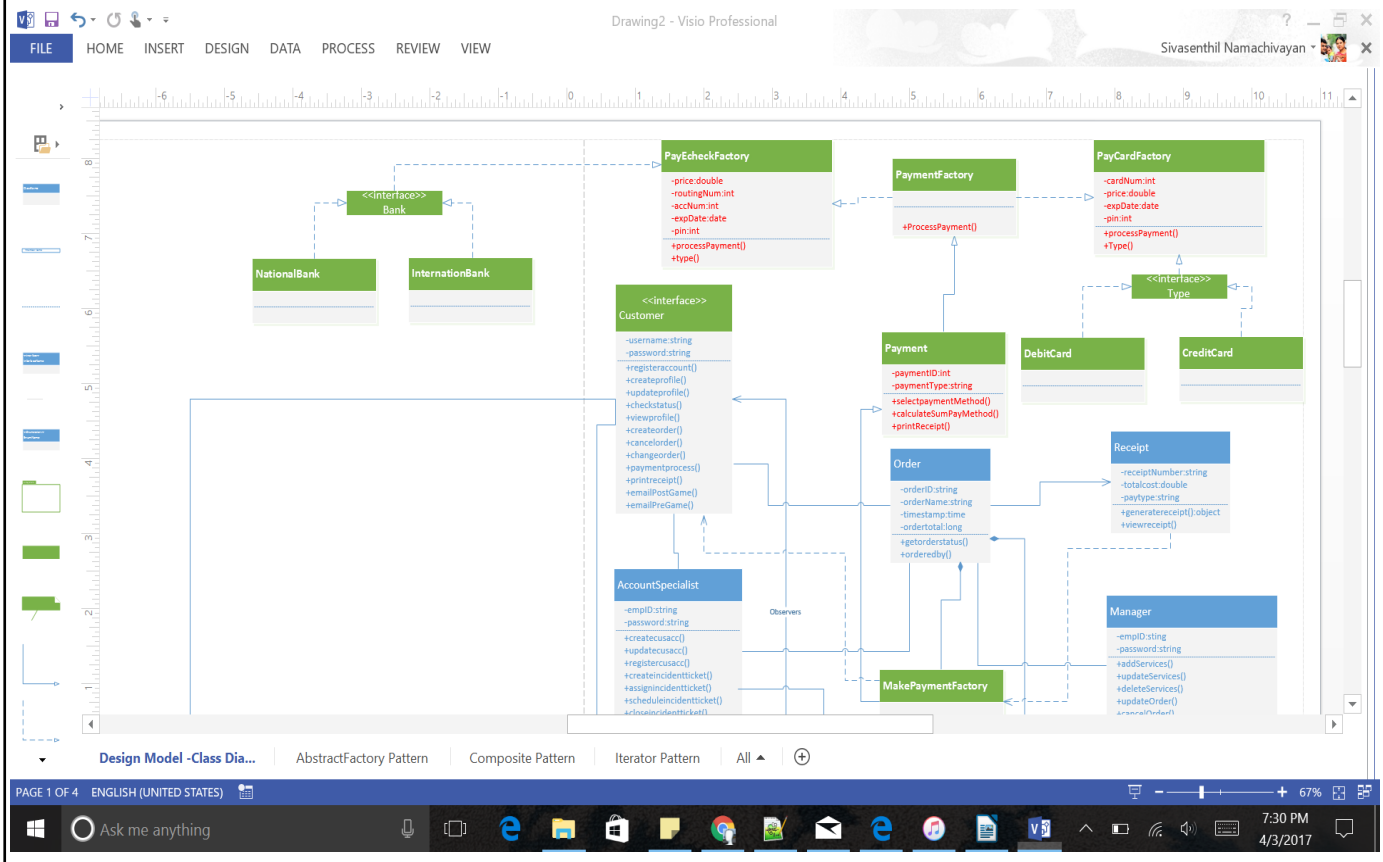
Class PayEcheckFactory and Class PayCardFactory
Each of these concrete subclassses create a family of payments for each type. The methods to process payment in an abstract factory are often implemented with a factory method.

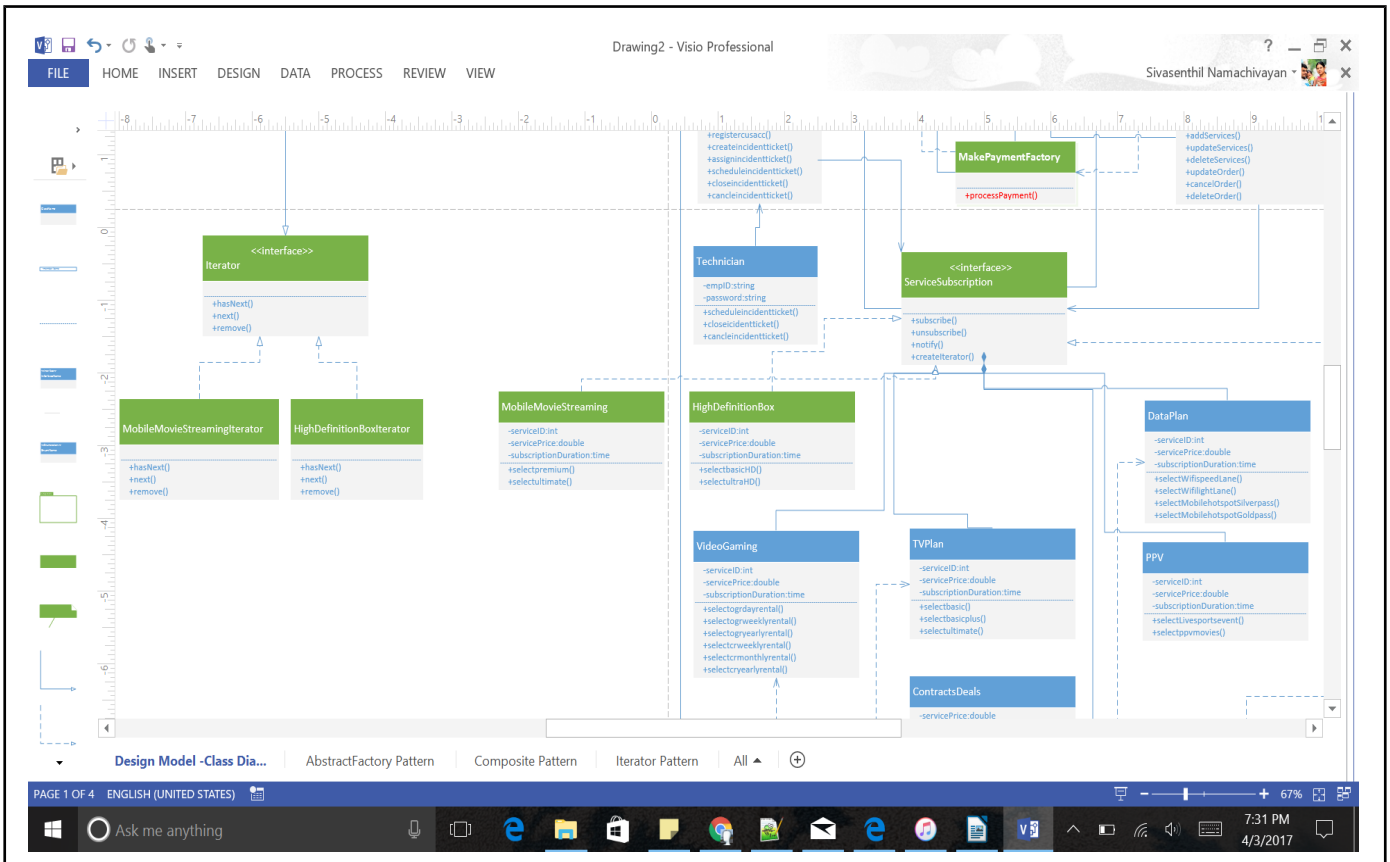
Class Bank and Class Type
Each Type represent the payment that is made by a factory method in the abstract factory

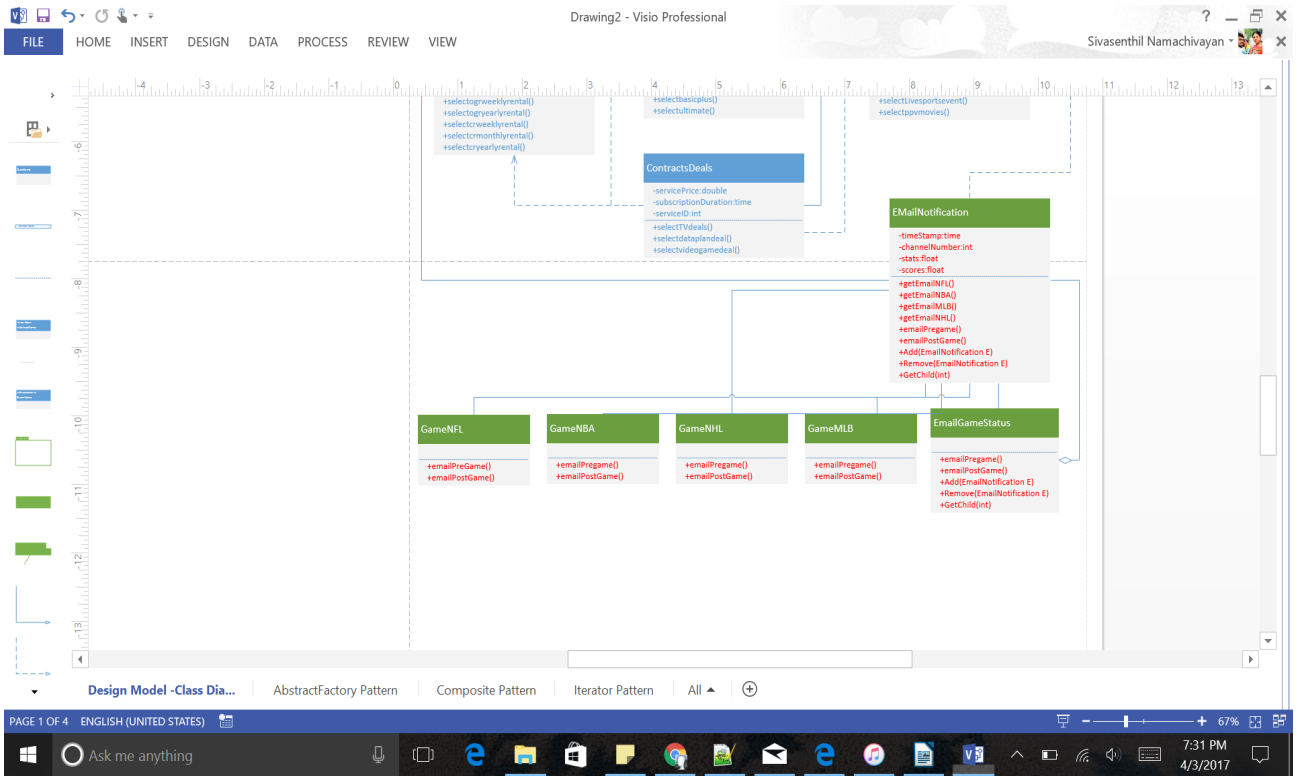
The payment types create parallel sets of payment families. Here we have payEcheckFactory family and PayCardFactory family.

Deliverable 4

Design Model Class Diagram using (Shift+PrtScr)




Drawing2 - Visio Professional
FILE
HOME
INSERT
DESIGN
DATA
PROCESS
REVIEW
VIEW
Sivasenthil Namachivayan
<<interface>>
Iterator
+hasNext()
+next()
+remove()
MobileMovieStreamingIterator
HighDefinitionBoxIterator
MobileMovieStreaming
-serviceID:int
-servicePrice:double
-subscriptionDuration:time
+selectpremium()
+selectultimate()
Technician
-empID:string
-password:string
+scheduleincidentticket()
+closeicidentticket()
+cancleincidentticket()
MakePaymentFactory
+processPayment()
<<interface>>
ServiceSubscription
+subscribe()
+unsubscribe()
+notify()
+createIterator()
HighDefinitionBox
+selectbasicHD()
+selectultraHD()
DataPlan
+selectWifispeedLane()
+selectWifilightLane()
+selectMobilehotspotSilverpass()
+selectMobilehotspotGoldpass()
VideoGaming
+selectogrdayrental()
+selectogrweeklyrental()
+selectogryearlyrental()
+selectcrweeklyrental()
+selectcrmonthlyrental()
+selectcryearlyrental()
TVPlan
+selectbasic()
+selectbasicplus()
+selectultimate()
PPV
+selectLivesportsevent()
+selectppvmovies()
ContractsDeals
-servicePrice:double
Design Model -Class Dia...
AbstractFactory Pattern
Composite Pattern
Iterator Pattern
All
PAGE 1 OF 4
ENGLISH (UNITED STATES)
67%
Ask me anything
7:31 PM
4/3/2017


Drawing2 - Visio Professional
FILE
HOME
INSERT
DESIGN
DATA
PROCESS
REVIEW
VIEW
Sivasenthil Namachivayan
ContractsDeals
-servicePrice:double
-subscriptionDuration:time
-serviceID:int
+selectTVdeals()
+selectdataplandeal()
+selectvideogamedeal()
EMailNotification
-timeStamp:time
-channelNumber:int
-stats:float
-scores:float
+getEmailNFL()
+getEmailNBA()
+getEmailMLB()
+getEmailNHL()
+emailPregame()
+emailPostGame()
+Add(EmailNotification E)
+Remove(EmailNotification E)
+GetChild(int)
GameNFL
+emailPreGame()
+emailPostGame()
GameNBA
+emailPregame()
+emailPostGame()
GameNHL
+emailPregame()
+emailPostGame()
GameMLB
+emailPregame()
+emailPostGame()
EmailGameStatus
+emailPregame()
+emailPostGame()
+Add(EmailNotification E)
+Remove(EmailNotification E)
+GetChild(int)
Design Model -Class Dia...
AbstractFactory Pattern
Composite Pattern
Iterator Pattern
All
PAGE 1 OF 4
ENGLISH (UNITED STATES)
67%
Ask me anything
7:31 PM
4/3/2017