CSP 586: Software Modeling and Development with UML

WideCast Cable & Internet Provider

* List of Actors:

1. Managers
2. Account Specialists (Customer Support)

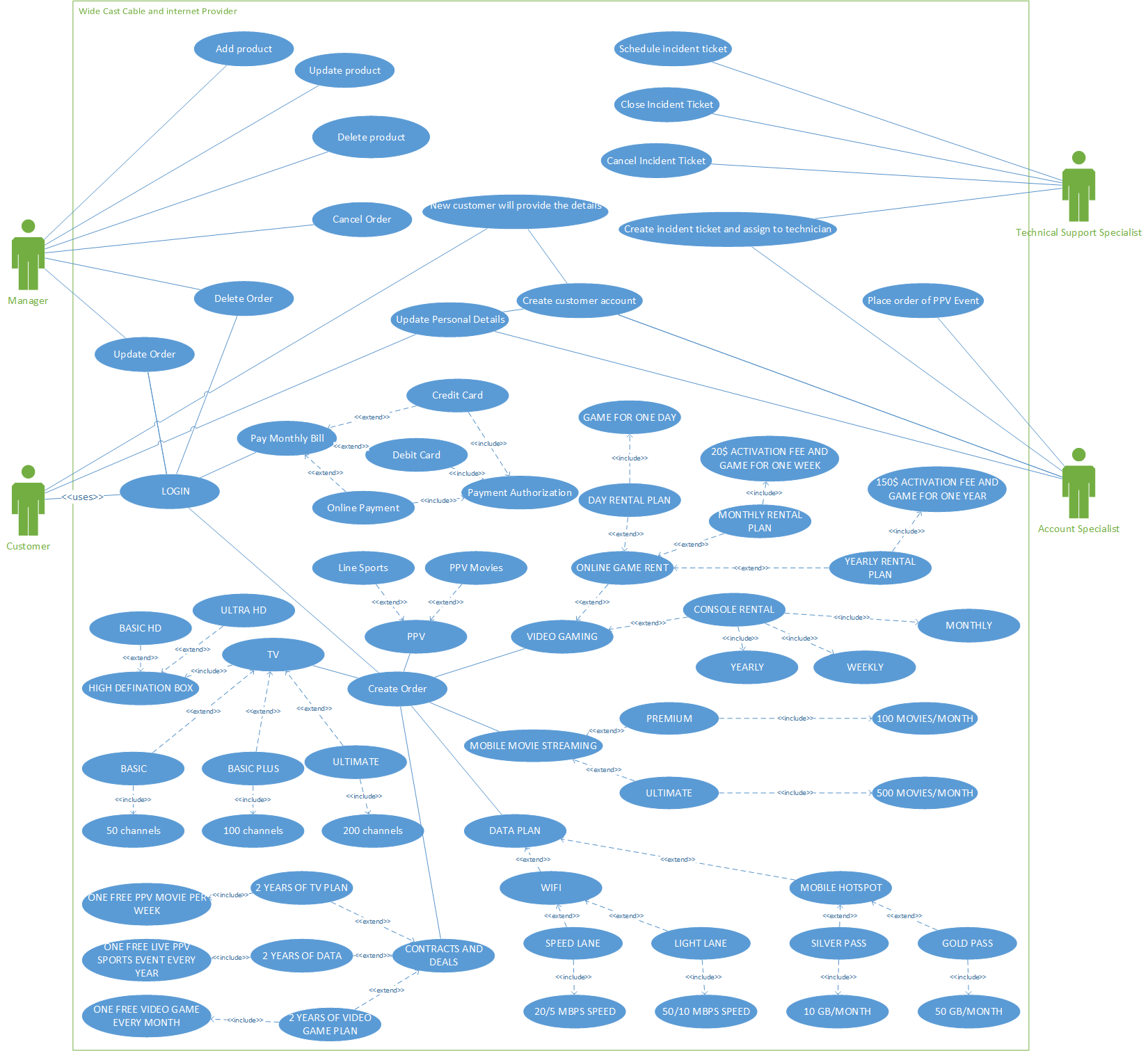
3. Technical Support Specialists (Technician)

4. Customers

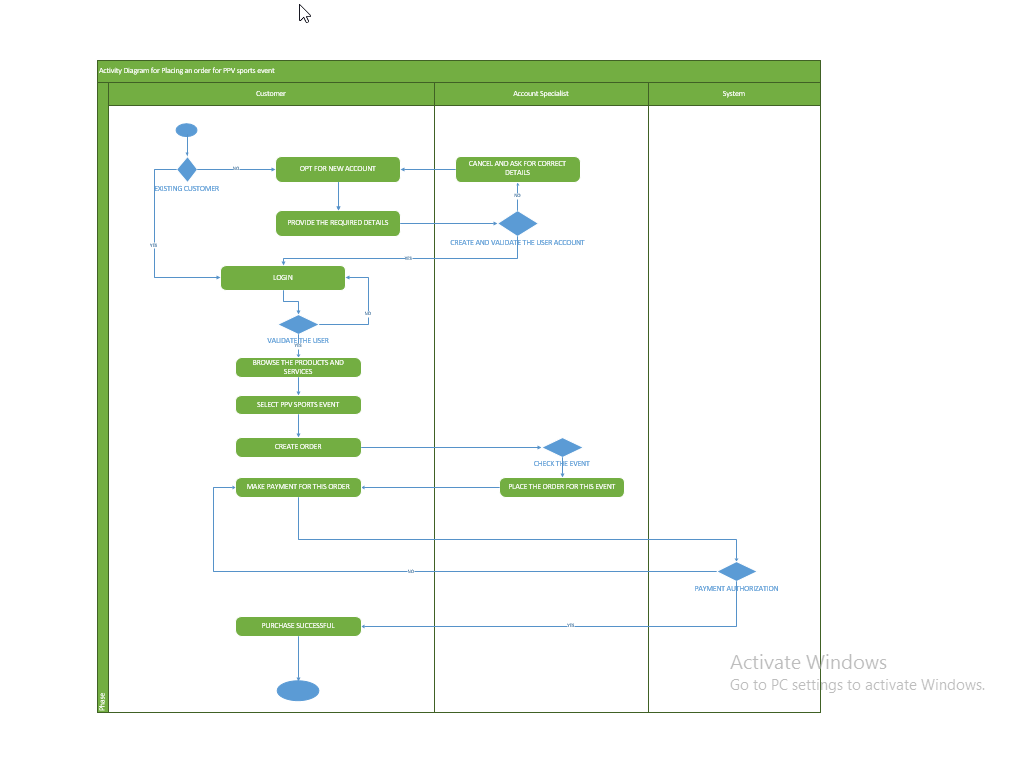
* List of Use Cases:

|  |  |
| --- | --- |
| Managers | 1. Add any products or services offered 2. Update any products or services offered 3. Delete any products or services offered 4. Cancel any type of order 5. Update any type of order 6. Delete any type of order |
| Account Specialists (Customer Support) | 1. Create an account for a new customer   1. 2. Update customer account 2. 3. Create an incident ticket and assign it to a Technician 3. 4. Place an order of a PPV event |
| Customer | 1. Update Personal Account Information (credit card, personal info, etc.) 2. Pay Monthly Bill 3. Create an order for any products/services 4. Change an order for any products/services 5. Cancel an order for any of the  products/services 6. Login with the credentials 7. If it’s a new customer, he will request for a new account. |
| Technical Support Specialists (Technician) | 1. Schedule incident ticket 2. Close incident ticket 3. Cancel incident ticket |
| System for payment authorization | 1. Validate the payment paid |

* Use Case Diagram



* Activity Diagram For Placing order for PPV Sports Event

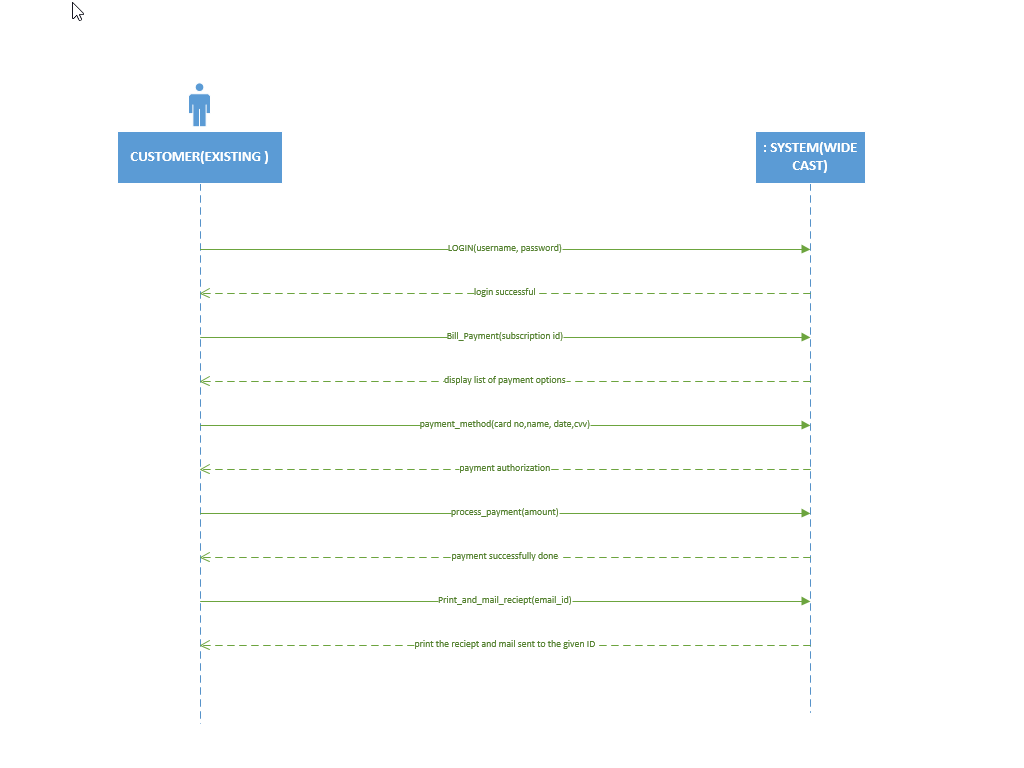


* Fully Dressed Format of the Use Case

|  |  |
| --- | --- |
| Use Case Name | Order a PPV Sport Event |
| Scope | WideCast Cable and Internet Provider |
| Level | User Level Goal |
| Primary Actor | Customer |
| Stakeholders and Interests | **Account Specialists**   * To create new accounts for the new customers with the details provided from him. * To place an order for PPV Sports Event when the customer orders for it.   **System for Payment Authorization**   * System will verify the payment made. |
| Preconditions | * If it’s a new customer, then he should provide the required details to the Account Specialists and get his account created. * If it’s an existing customer, then he should have correct login details. |
| Success Guarantee | * While placing the order for the PPV Event, that event should be available for that fixed date. * The Payment made should be authorized and successful. |
| Main Success Scenario | 1. The new customers will request for the new account which will be created by the account specialist. 2. The existing customer will login with his credentials. 3. He will then browse through the products and services offered. 4. He will select the PPV Sports Service. 5. He must be sure that its available on that day. 6. He will create the order for that event. 7. The account specialist will acknowledge the above request and places the order for the sports event after verifying its availability. 8. The customer should make the payment 9. The system will verify the payment made. |
| Extensions | Can’t cancel the order placed for PPV Sports Event |
| Special Requirements | Internet Connection should be there. |
| Technology and Data Variation List | * System to do the payment authorization * Internet connection to view or login to the website. |
| Frequency of Occurrence | Occasional based on the customer’s interest to see the Sports event and if the sports event is on that day. |
| Miscellaneous | If you select for 2-year plan of data plan, then you get one live PPV Sports event free every year. So, you can also opt for this plan. |

1. Any Three System Sequence Diagrams for the use-case diagram you created in

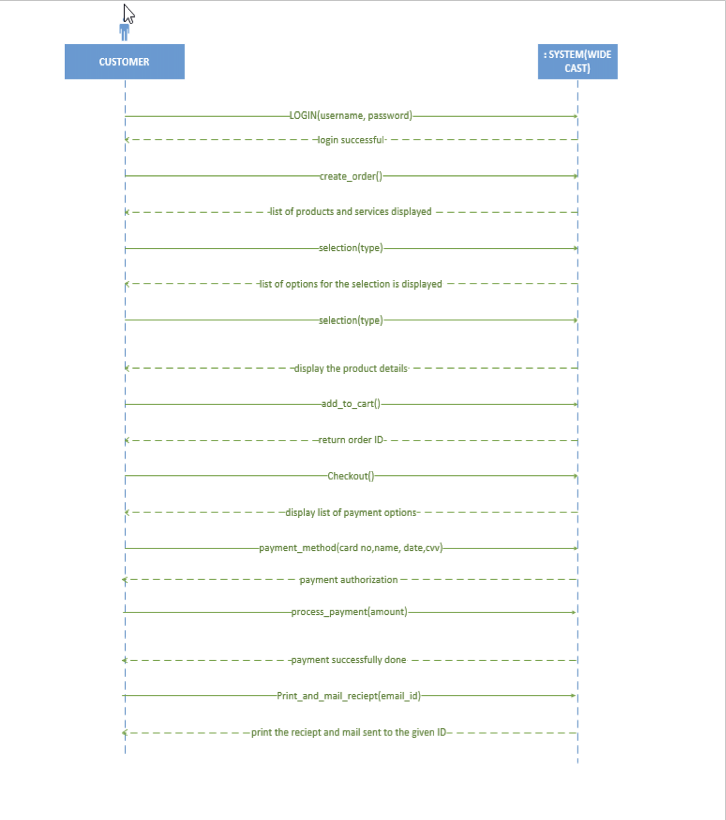
A. PAY MONTHLY BILLS

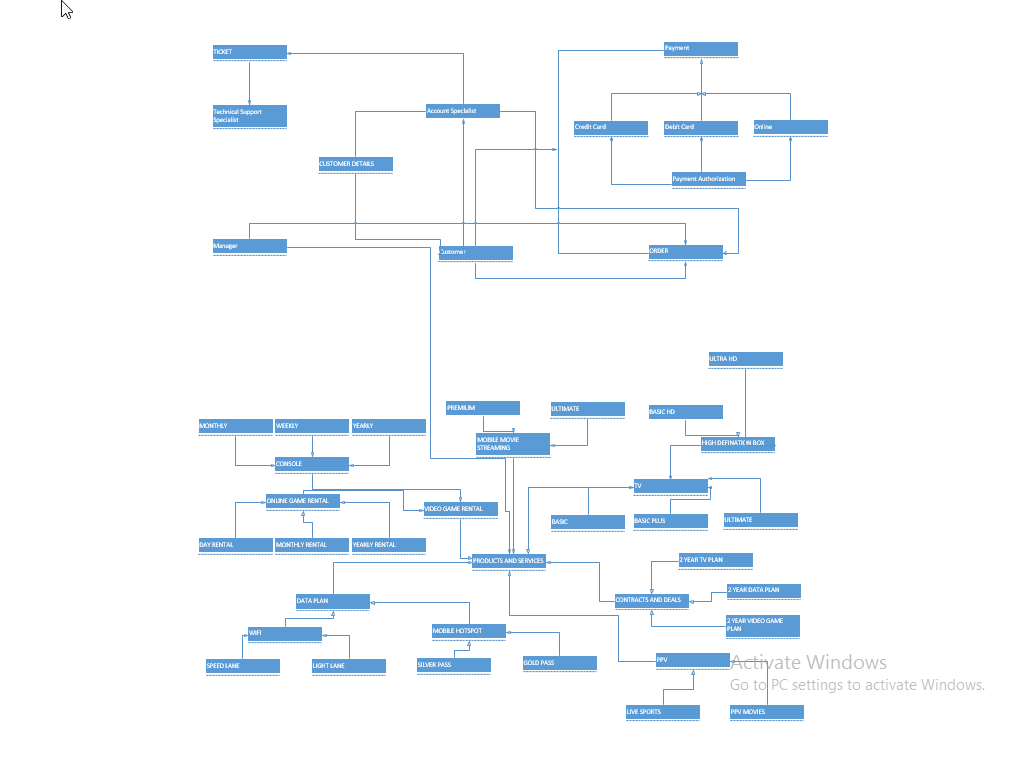
B. INCIDENT TICKET PROCESS



C. CREATE ORDER BY THE CUSTOMER



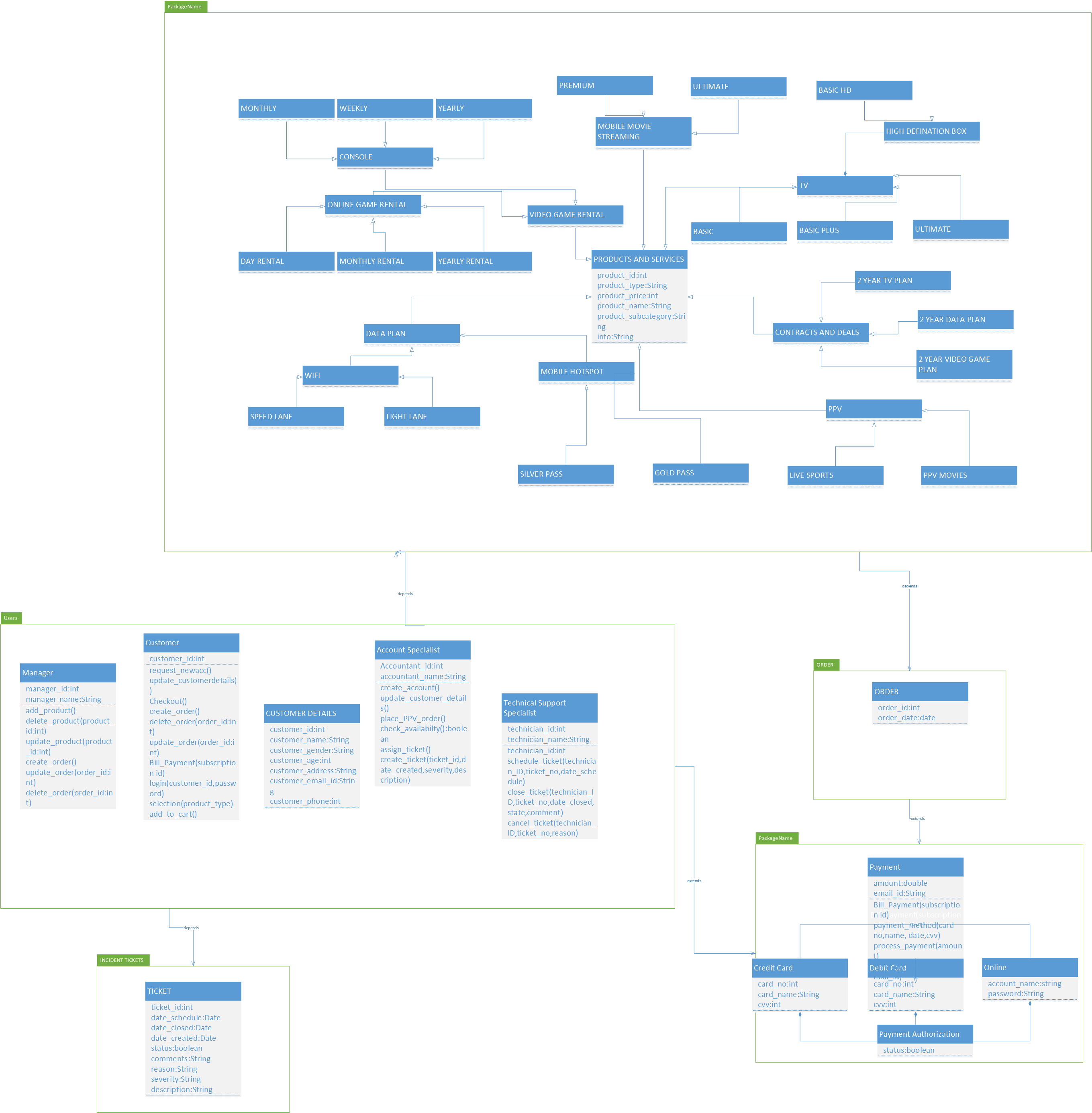
2. The Class diagram for your analysis model



3. The Class diagram and package diagram for your design model CLASS DIAGRAM :

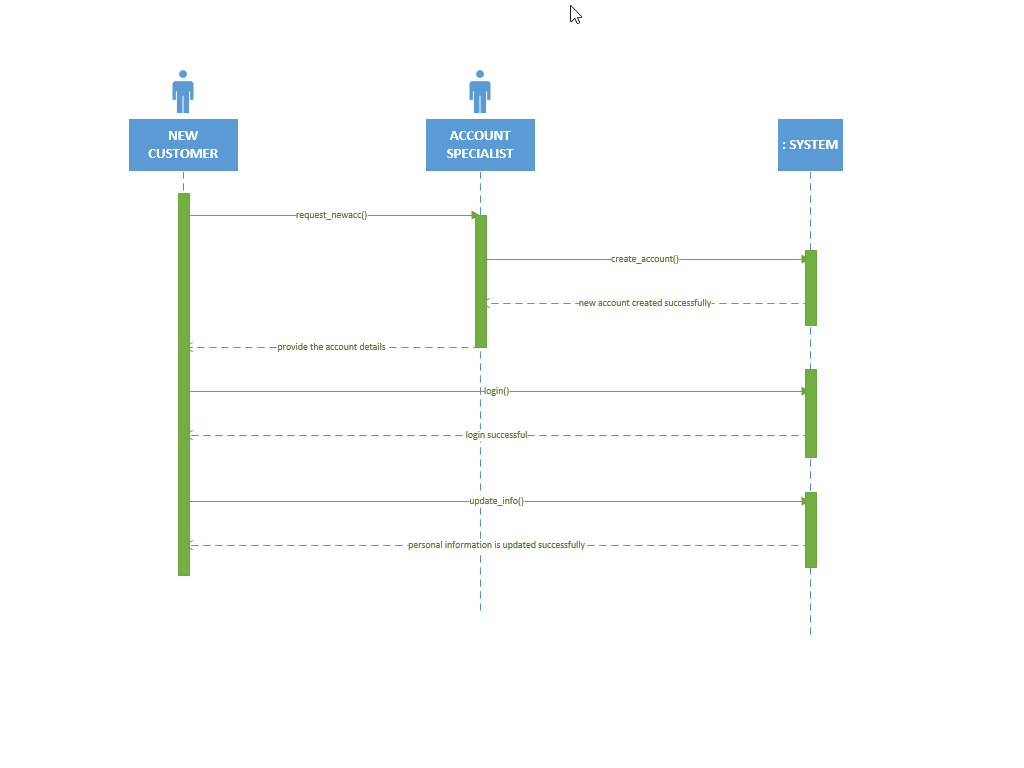


PACKAGE DIAGRAM

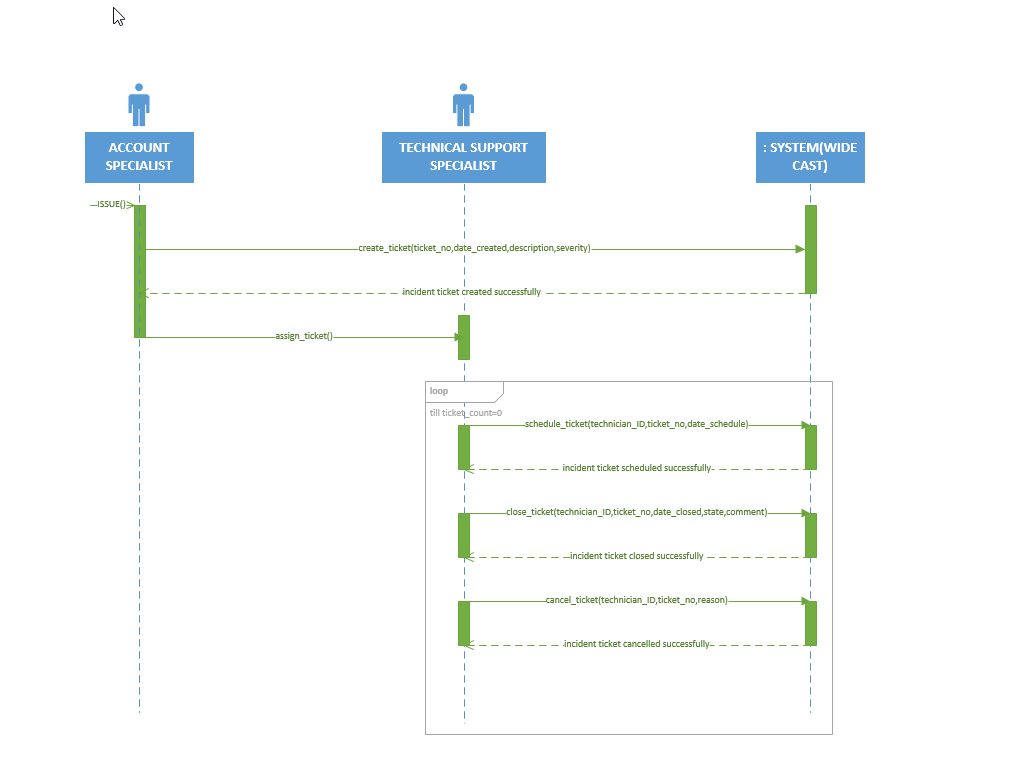


4. Any Five sequence interaction diagrams that you can pick form the list of potential sequence interaction diagrams for your design model.

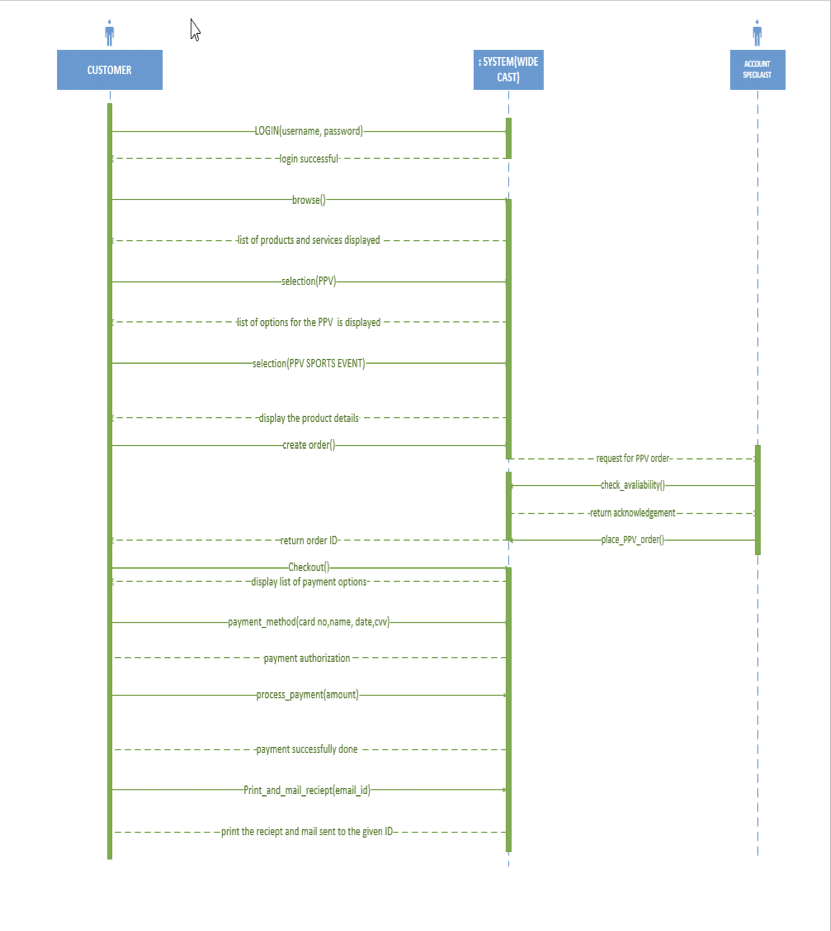
A. CREATION OF NEW CUSTOMER ACCOUNT



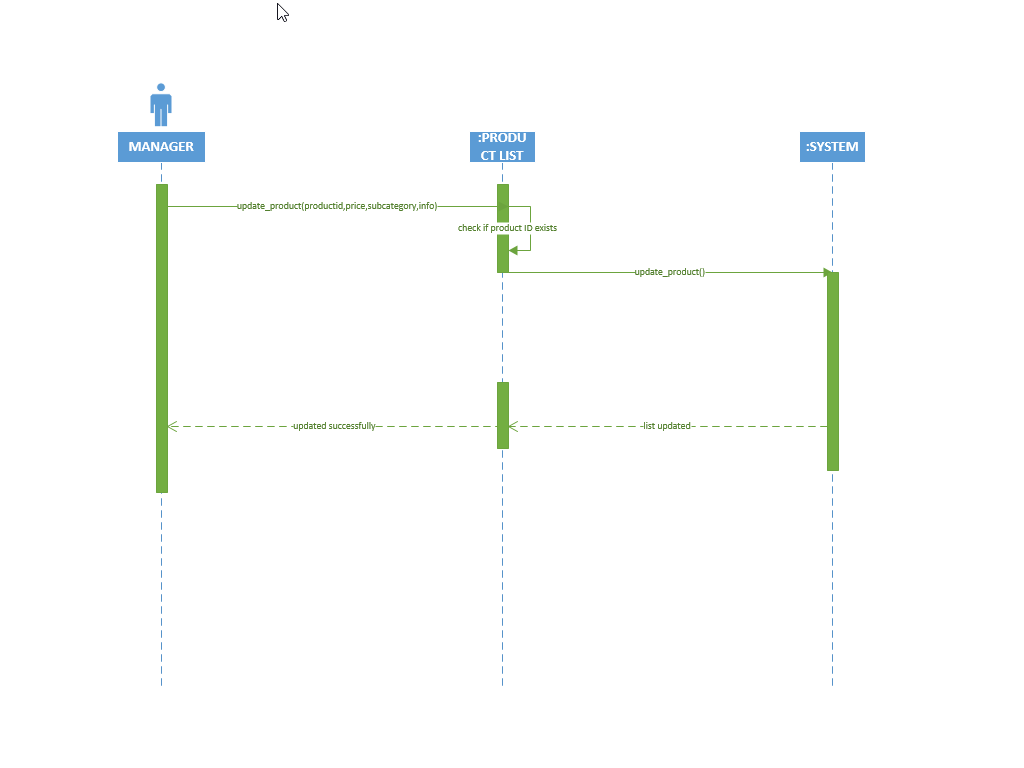
B. INCIDENT TICKET CREATION AND RESOLUTION



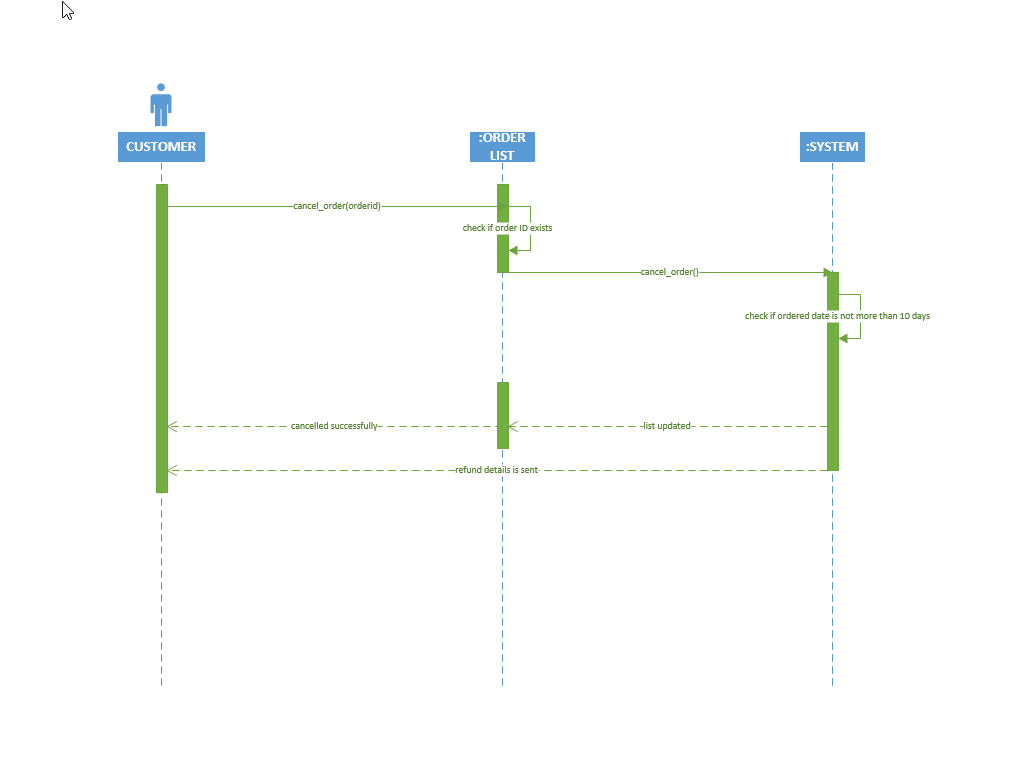
C. ORDER PPV SPORTS EVENT BY THE CUSTOMER



D. UPDATE PRODUCT BY THE MANAGER



E. CANCEL ORDER BY THE CUSTOMER



1. Complete list of classes
2. Technical Support Specialist

2. Manager

3. Customer

    a. Customer details

4. Account Specialist

5. Ticket

6. Payment

    a. Payment Strategy

    b. Credit Card Strategy

    c. Debit Card Strategy

d. Online Strategy

7. Order

8. Subscribes

9. Subscriber

10. NFL

11. MLB

12. NHL

13. NBA  
14. TV

    a. Basic

    b. Basic Plus

    c. Ultimate

    d. HD Box

        - Basic HD

        - Ultra HD

15. Mobile Movie Streaming

    a. Ultimate

    b. Premium

16. Contract & Deals

    a. 2 Year TV Plan

    b. 2 Year Data Plan

    c. 2 Year Video game Plan

17. PPV

    a. Line Sports

   b. PPV Movie

18. Data Plan

    a. WiFi

      - Speed Lane

      - Light Lane

     b. Mobile Hotspot

      - Silver Pass

      - Gold Pass

19. Video Game Rental

    a. Online Game Rental

      - Day Rental

      - Monthly Rental

      - Yearly Rental

    b. Console

      - Monthly

      - Weekly

      - Yearly

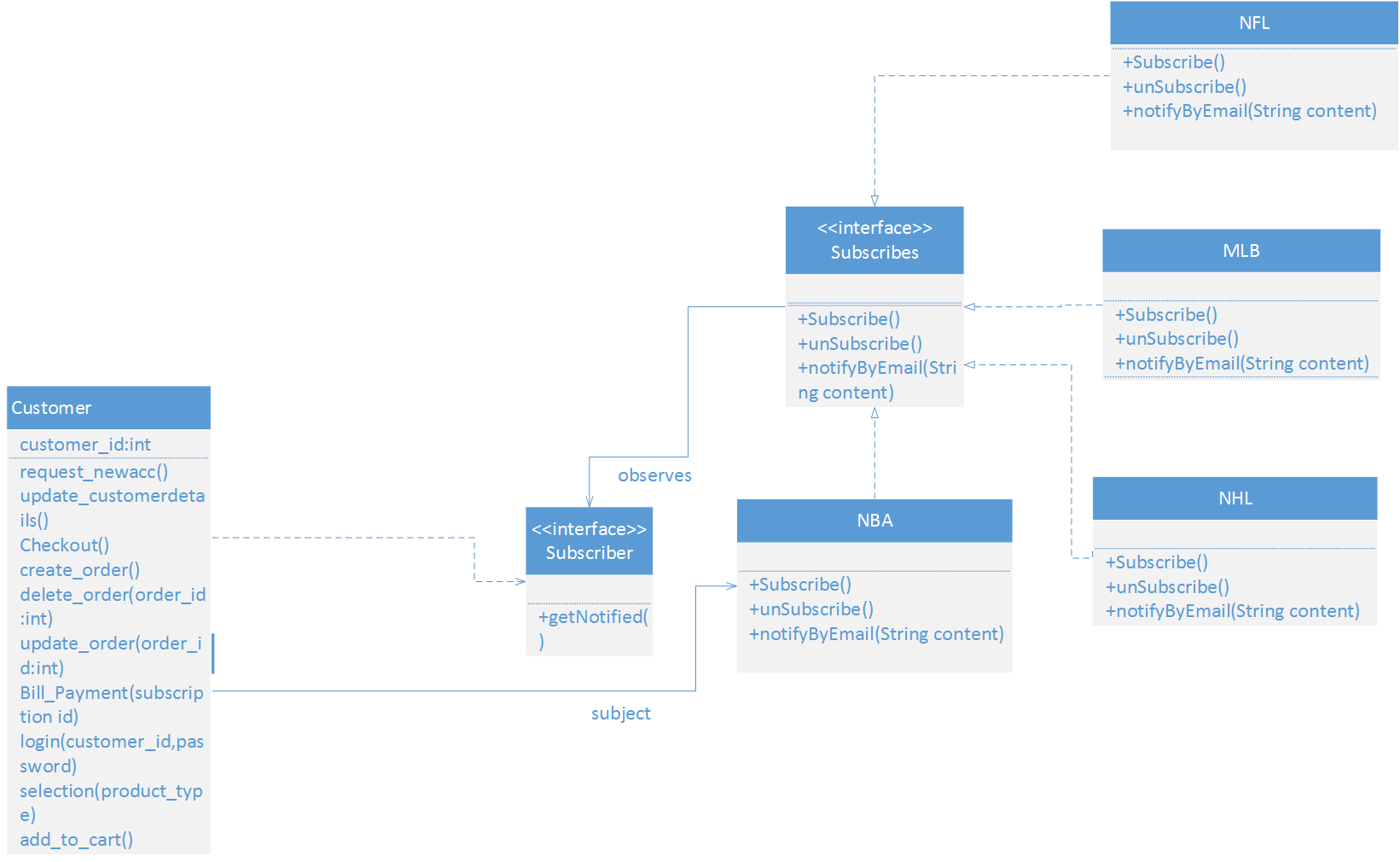
1. Complete UML Design Model/class diagram



1. List of the Design patterns used

* Observer Design pattern
* Strategy Design Pattern

1. Documentation of the design pattern
2. Observer Design pattern



From the definition given, observer design pattern defines one to many dependency between object so that when one object changes state, all its dependents are notified and updated automatically. This type is behavioral pattern.

In my design model, the customer is considered as the observer and the subscription is considered to the Subject Event.

The customers can register to get the subscriptions for the given list of four games. In this, the customers use the subscribes interface to register as observers and to remove themselves from the list.

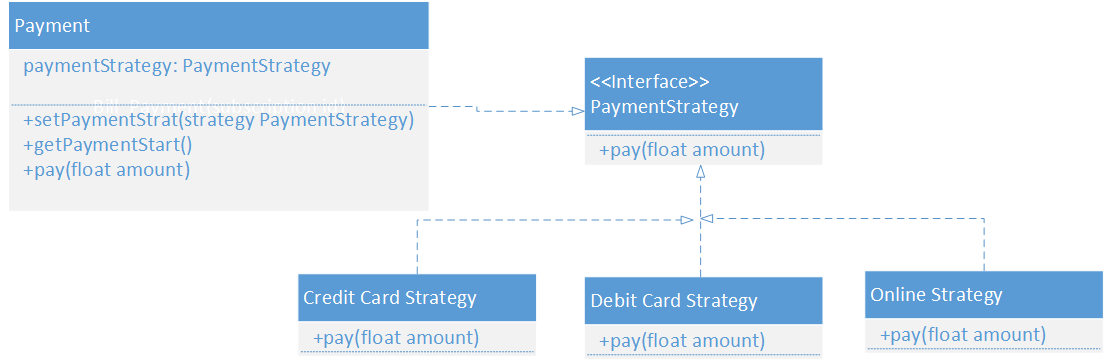
Each subscribes interface can have many customers(observer)

All the customers should implement the subscriber interface to receive the interface.

And customers class is any class that implements the observer interface.

NFL, NHL, NBA, MLB are all considered as the concrete subject that have methods to set and get the state and these implement the subject interface.

1. Strategy Design Pattern



In Strategy pattern, a behavior or its algorithm can be changed at run time. This type of design pattern is behavior pattern.

In Strategy pattern, we create objects which represent various strategies and a context object whose behavior varies as per its strategy object. The strategy object changes the executing algorithm of the context object.

The Interface entity represents either an abstract base class, or the method signature expected by the client. So, in this case, the customer who is the client strategies the payment type. Payment class is the interface of the payment strategy. It gives the option of credit card, debit card or online.

1. Design Model class diagram



1. List of the Design pattern(s) used in:

* Assignment 3

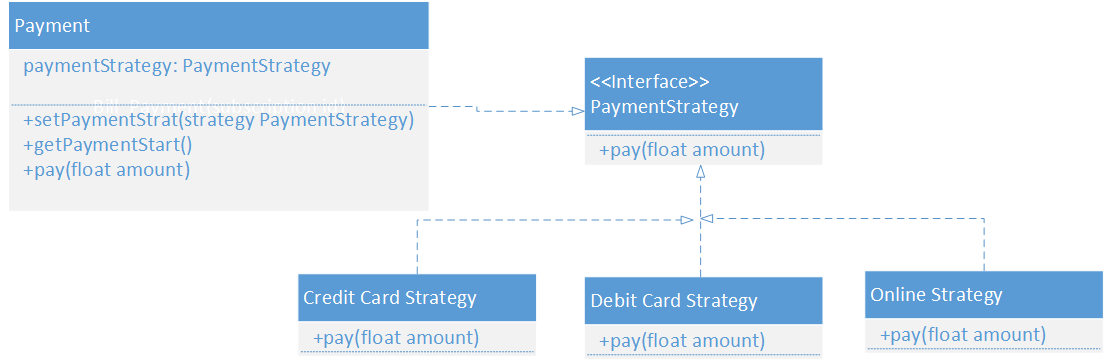
1. Observer Design pattern
2. Strategy Design Pattern

* Assignment 4

1. Strategy Design Pattern
2. Factory Design Pattern
3. Iterator Design Pattern
4. Singleton Design Pattern

[NOTE: I have implemented four design pattern this time since in previous assignment Strategy Design Pattern was not considered (was not taught back then)So total count of design pattern is five]

1. Documentation
2. STRATEGY DESIGN PATTERN

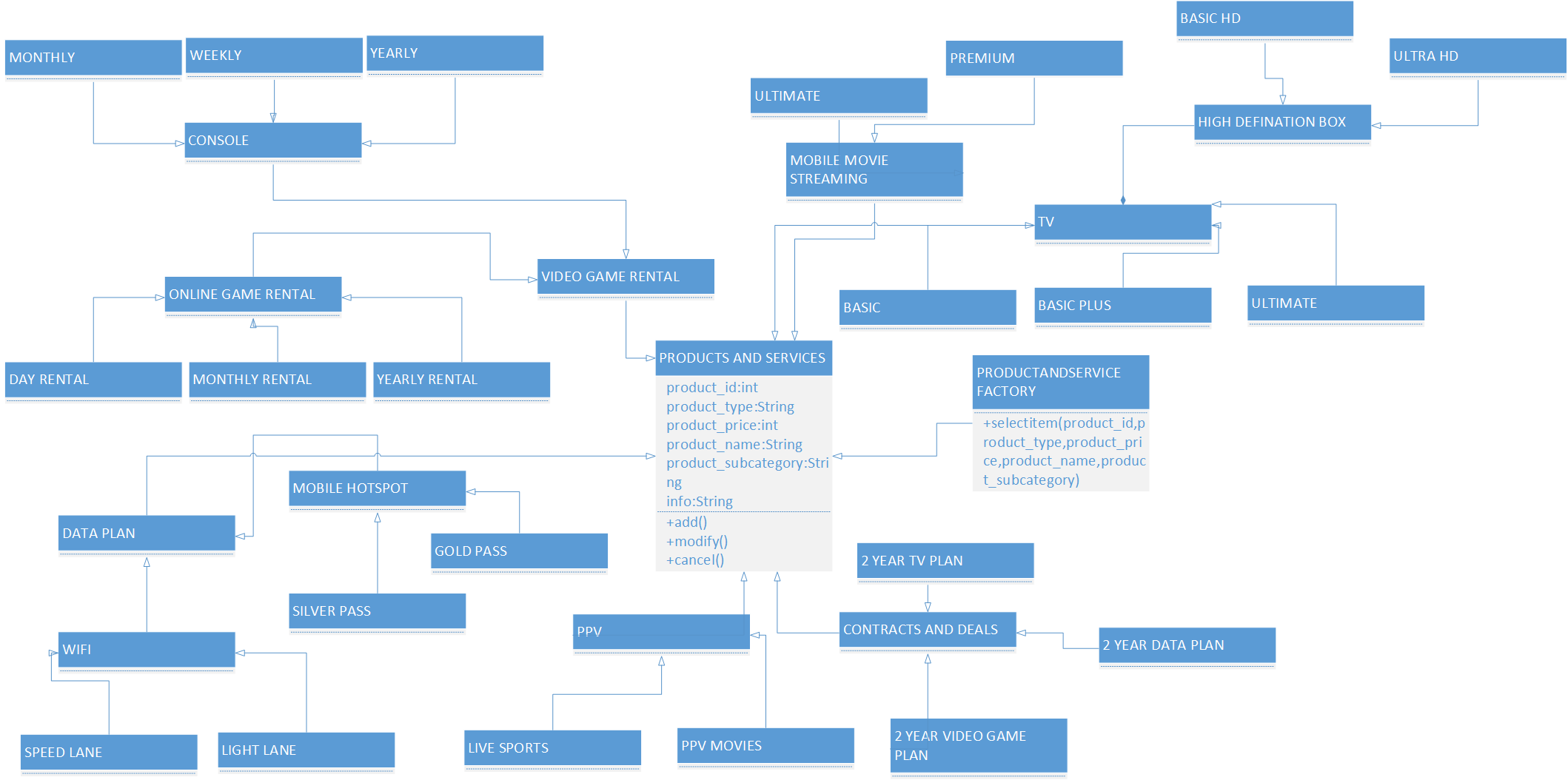


In Strategy pattern, a behavior or its algorithm can be changed at run time. This type of design pattern is behavior pattern.

In Strategy pattern, we create objects which represent various strategies and a context object whose behavior varies as per its strategy object. The strategy object changes the executing algorithm of the context object.

The Interface entity represents either an abstract base class, or the method signature expected by the client. So, in this case, the customer who is the client strategies the payment type. Payment class is the interface of the payment strategy. It gives the option of credit card, debit card or online.

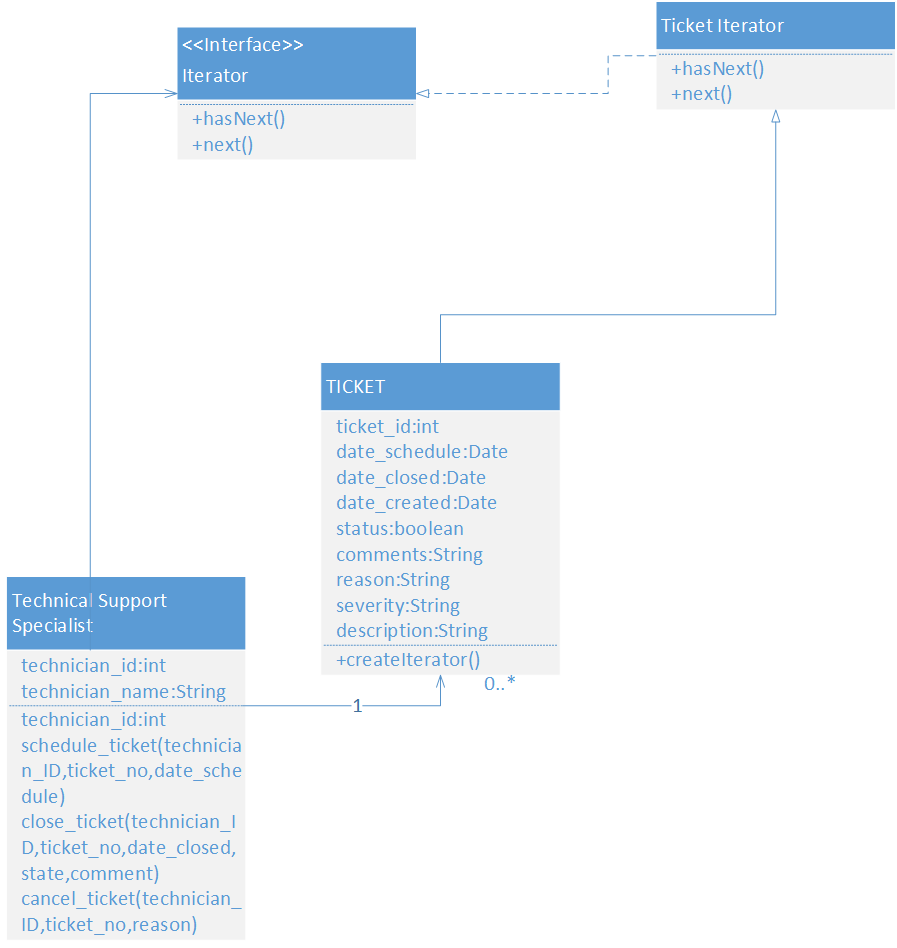
1. **FACTORY METHOD DESIGN PATTERN**

****

It defines an interface for creating an object, but lets the subclass to instantiate. The factory method design pattern is used to model the services provided by the online service.

In the above design, the ProductandService factory is newly implemented. The selectItem method consists of the attributes for the product. In the product and services class, it consists of all the products and services offered.

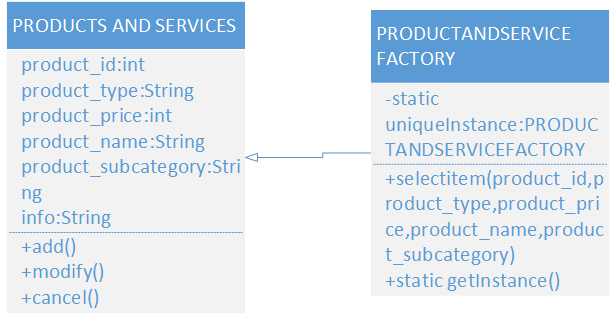
1. ITERATOR DESIGN PATTERN



Iterator design pattern allows traversal of the elements of an aggregate without exposing the underlying implementation. In this, we implement iterator interface that provides the interface that all iterators must implement and set of methods for traversing over elements of a collection.

Here the concrete Iterator is the Ticket Iterator which responsible for the managing the current position of the iteration. The client in this scenario is Technical support specialist who is associated with tickets operation. The ticket class is the collection or the aggregates. In the ticket class, we have the createIterator method which is responsible for creating the iterations.

1. SINGLETON DESIGN PATTERN



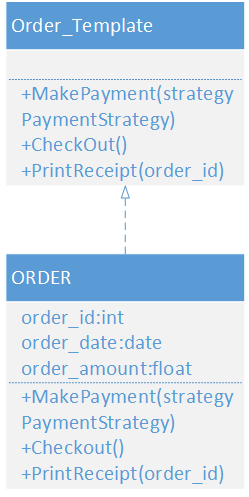
Singleton design pattern ensures a class has only one object, and provides a global point of access to it. It belongs to creational design patterns. The class implementing the singleton pattern is PRODUCTANDSERVICE FACTORY which is a general-purpose class.

In this design pattern, a unique instance class variable and static public method getInstance is implemented. This getInstance() method is the class method which can be accessed anywhere.

The Additional two design pattern implemented are:

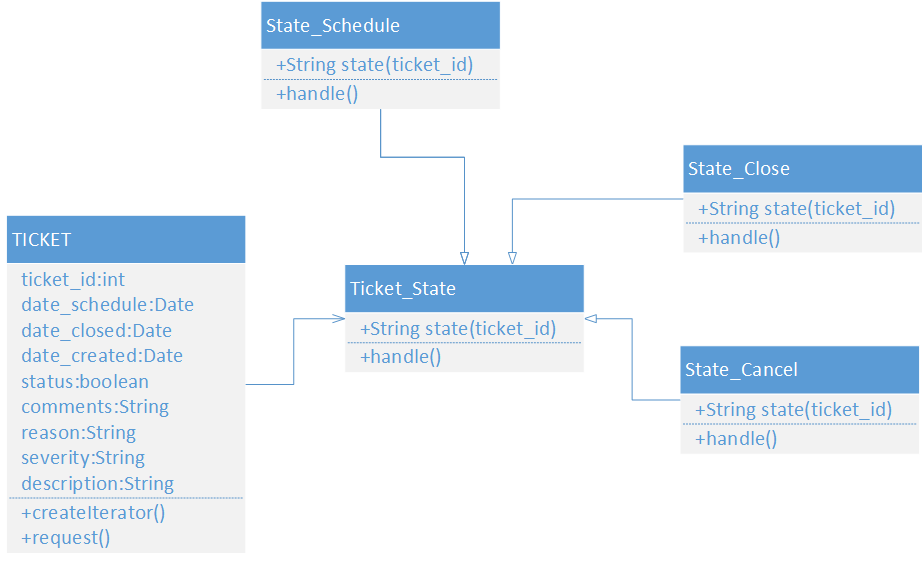
1. Template Method Design Pattern
2. State Design Pattern
3. Template Method Design Pattern

It defines the skeleton of an algorithm in a method, deferring some steps to subclasses. It lets subclasses redefine certain steps of an algorithm without changing the algorithm’s structure. The template method makes use of the primitive operations to implement an algorithm. It is decoupled from the actual implementation of these operation. The order class is concrete class which implements the methods in the template class. The order template class has the abstract methods needed while placing an order and making payment.



1. State design pattern

State Pattern allows an object to alter its behavior when its internal state changes. The Object will appear to change its class. The ticket class is the context class that can have many internal states. In the context class, there is a request() which is delegated to the state to handle. The State class defines the behavior of the context class. All the states are interchangeable and many states can exist. The states handle requests from the context. Each state gives its own implementation for a request. In this way, when the context changes state, its behavior will change as well.



UPDATED DESIGN CLASS DIAGRAM:



MVC ARCHITECTURAL PATTERN

