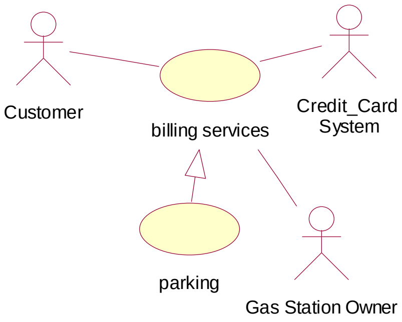
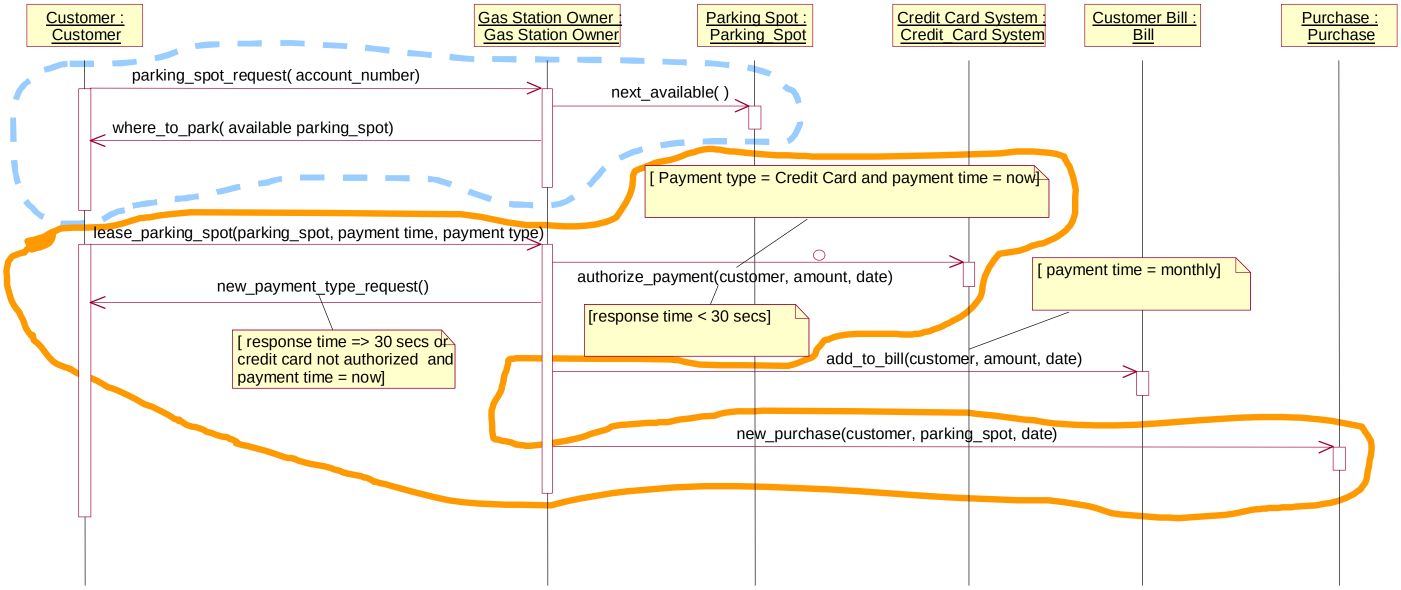
The Examples of Documents to Be Read

Example 1: A use case for an automated system at a gas station, describing how a customer purchases a parking spot. Note that "time of payment is the same as purchase time" is a *condition*; it describes what must be true for the functionality to be executed. "The Customer can only wait for 30 seconds for the authorization process" imposes a *constraint* that must be always be true for system functionality.

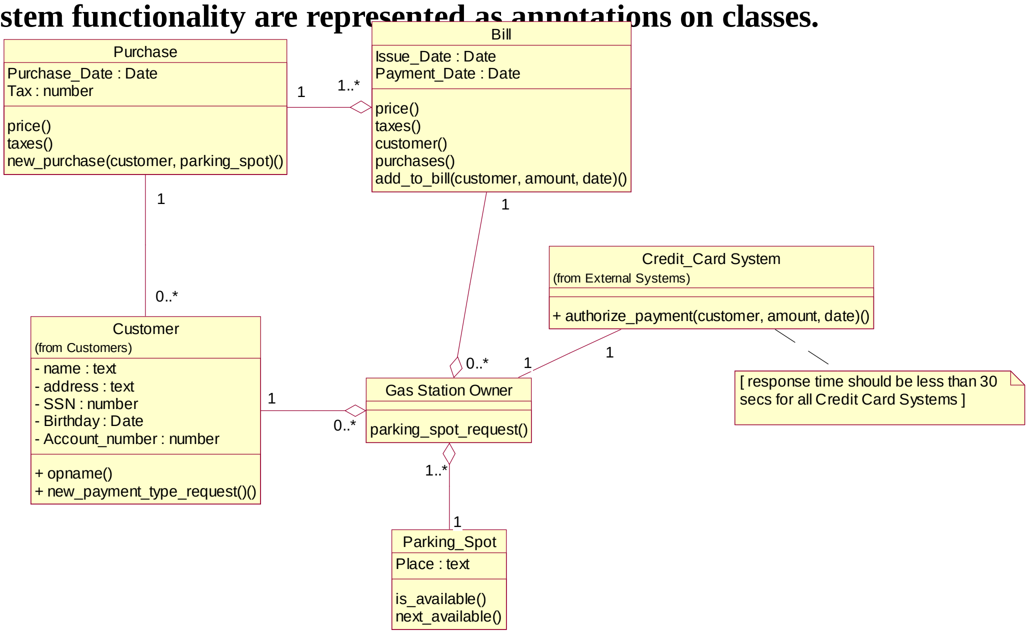


A customer, giving his account_number, asks the Gas Station Owner for an available parking spot to park his car.
To get an available parking spot Gas Station Owner searches for the next parking place available. With this information the customer can confirm the lease of the parking place. The time of payment (time of purchase or a monthly paper bill) and how the service should be paid (by cash, personal check or credit card).
If the time of payment is the same as the purchase time and Customer decides to pay by Credit Card then Credit Card system should be used. The Customer can only wait for 30 seconds for the authorization process otherwise this payment should be made by cash or personal check to avoid other Customers waiting on the lane. The Gas Station Owner should ask the Customer for a new payment type.
It allows the Gas Station Owner to mark a new service purchase for this Customer at this date.

Example 2: A sequence diagram for the automated gas station system, capturing how classes collaborate to perform the functionality described in Example 1. Combinations of messages that form system *services* have been marked. Conditions and constraints are included as annotations on the diagram. “Response time < 30 secs” represents a nonfunctional *constraint* on the way certain functionality has to be implemented. “Payment time = monthly” is an example of a *condition* that must be true for a particular message to be executed; in this case, the system variable “payment time” must have the value “monthly.”


Customer : Customer
Gas Station Owner : Gas Station Owner
Parking Spot : Parking_Spot
Credit Card System : Credit_Card System
Customer Bill : Bill
Purchase : Purchase
parking_spot_request(account_number)
next_available( )
where_to_park(available parking_spot)
[Payment type = Credit Card and payment time = now]
lease_parking_spot(parking_spot, payment time, payment type)
authorize_payment(customer, amount, date)
[payment time = monthly]
new_payment_type_request()
[response time < 30 secs]
[response time => 30 secs or credit card not authorized and payment time = now]
add_to_bill(customer, amount, date)
new_purchase(customer, parking_spot, date)

Example 3: The class diagram for the classes described in Example 2. Note that *constraints* on system functionality are represented as annotations on classes.



Example 4: Requirements descriptions and Class descriptions used to show how conditions and constraints should be considered while reading both documents. Observe the relationship between both documents shown by the underlined information.

Requirement Description

1 – A customer has the option to be billed automatically at the time of purchase (of gas, car maintenance or parking spots) or to be sent a monthly paper bill. Customers can pay via cash, credit card or personal check. Gas Station services have a fixed price (gas: US$ 1.09 gallon, car maintenance: US$ 150.00 and parking spot: US$ 5.00 per day). The tax is 5% added to the final price of the purchase. Sometimes, the Gas Station owner can define discounts to those prices.

Class Description

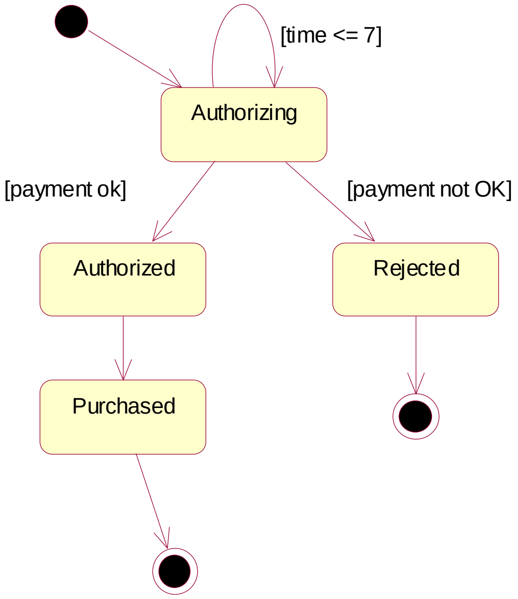
```
Class name:     Purchase
    Category:           Customers
    External Documents:
    Export Control:  Public
    Cardinality:            n
    Hierarchy:
      Superclasses:  none
    Public Interface:
      Operations:
                price
                taxes
    Private Interface:
      Attributes:
                Purchase_Date : Date
                Tax : number
              Service: Services
    Implementation:
      Attributes:
                Purchase_Date : Date
                Tax : number = 0.05
   Operation name:      price
    Public member of:        Purchase
    Concurrency:     Sequential
       Return   (1 + tax) * service->price
    Operation name:  taxes
    Public member of:        Purchase
    Concurrency:     Sequential
       Return tax * service->price

Class name:      Services
    Category:           Services
    External Documents:
    Export Control:  Public
    Cardinality:            n
    Hierarchy:
      Superclasses:  none
   Public Interface:
      Operations:
                price
    Private Interface:
      Attributes:
                Discount_Rate : number
              Price : number
    Implementation:
      Attributes:
                Discount_Rate : number
              Price: number
    Operation name:      price
    Public member of:        Services
    Concurrency:     Sequential
       Return  (1 - discount_rate ) * price

Class name:     Car_Maintenance
    Category:           Services
    External Documents:
    Export Control:  Public
    Cardinality:            n
    Hierarchy:
      Superclasses:  Services
   Public Interface:
      Operations:
                price
    Private Interface:
      Attributes:
```

```
                Price : number
Implementation:
   Attributes:
                Price : number = 150.00
```

Example 5: A state diagram for the "gas station owner" class from the automated gas station system. An associated sequence diagram is shown in Example 2.


[time <= 7]
Authorizing
[payment ok]
[payment not OK]
Authorized
Rejected
Purchased