BUE

The British University in Egypt

الجامعة البريطانية في مصر

Software Engineering II

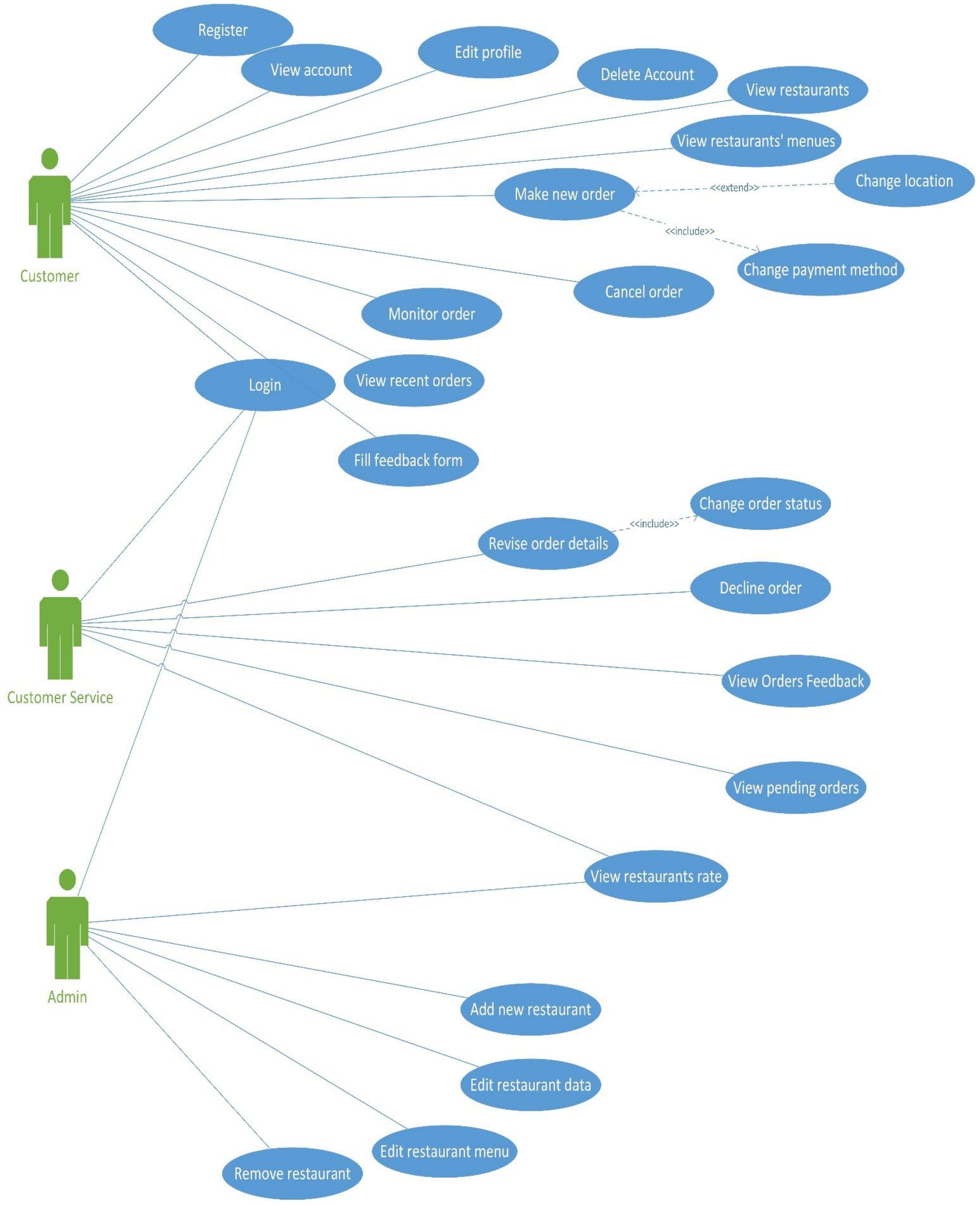
DR. ABEER HAMDY

FOOD ORDERING SERVICE

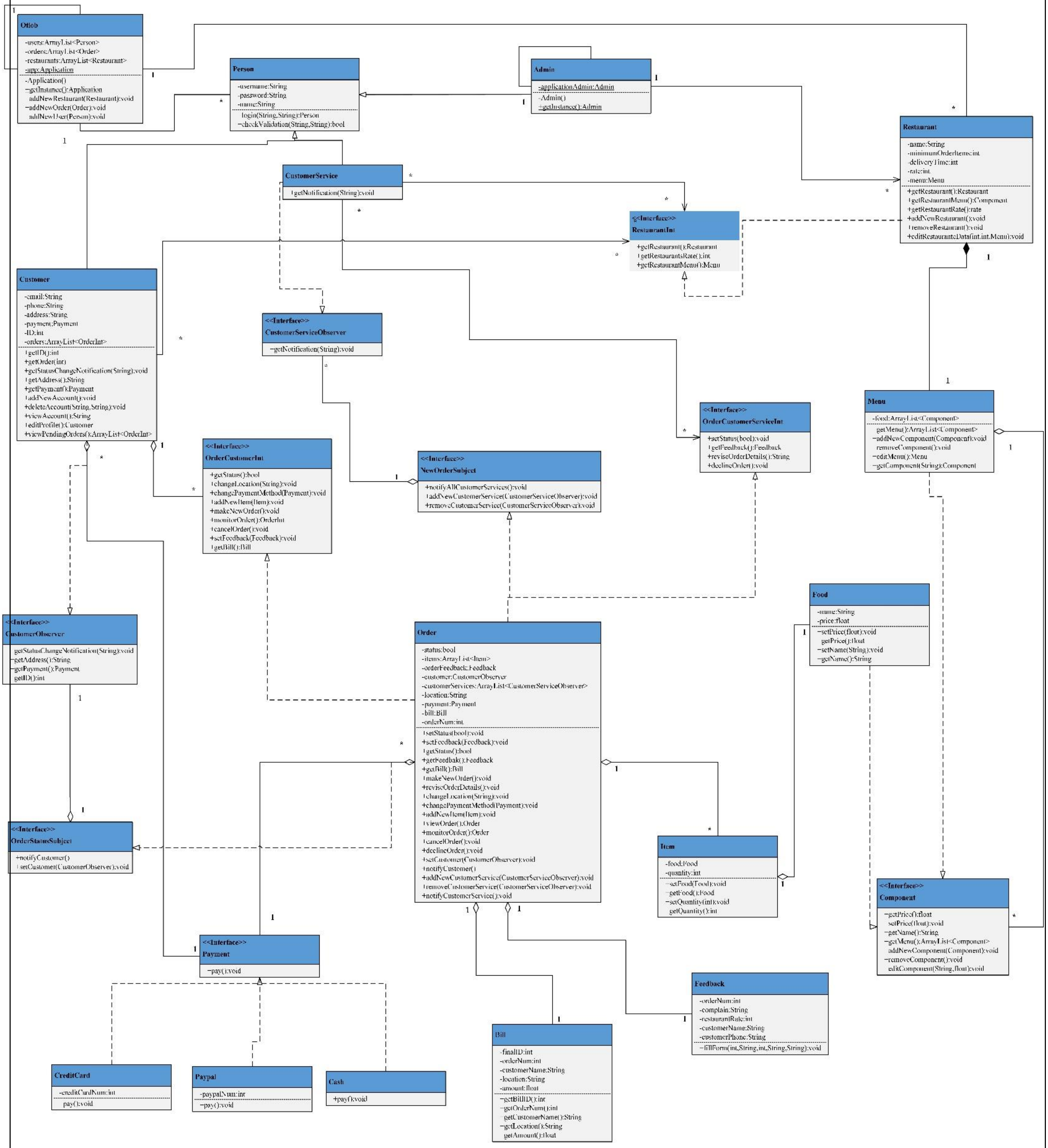
Name	ID
Gloria Ezzat	150753
Engy Samir	139394
Meran Nauman	148368
Wafaa Emad	147539

Table of contents

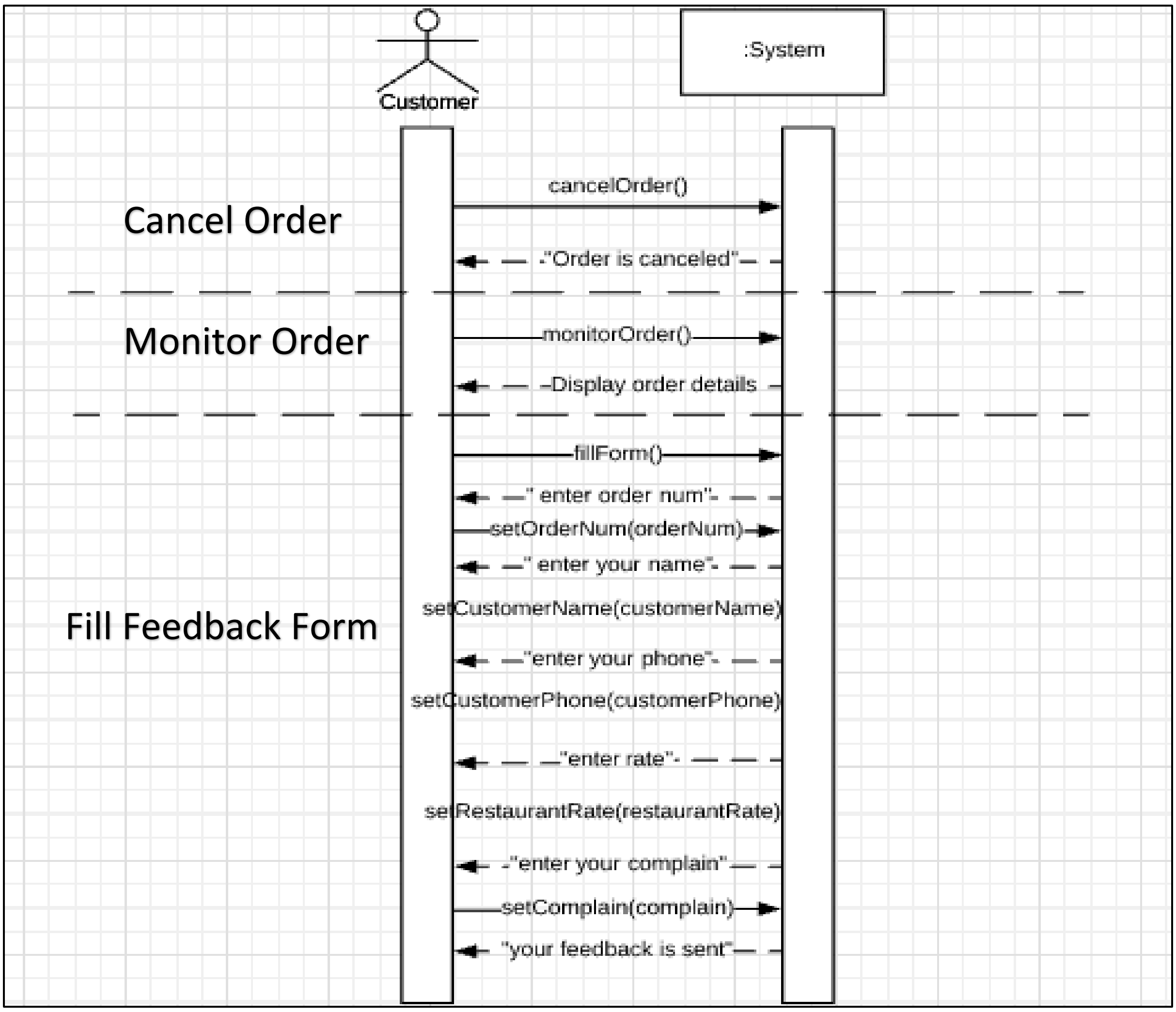
1- Use Cases (updated)

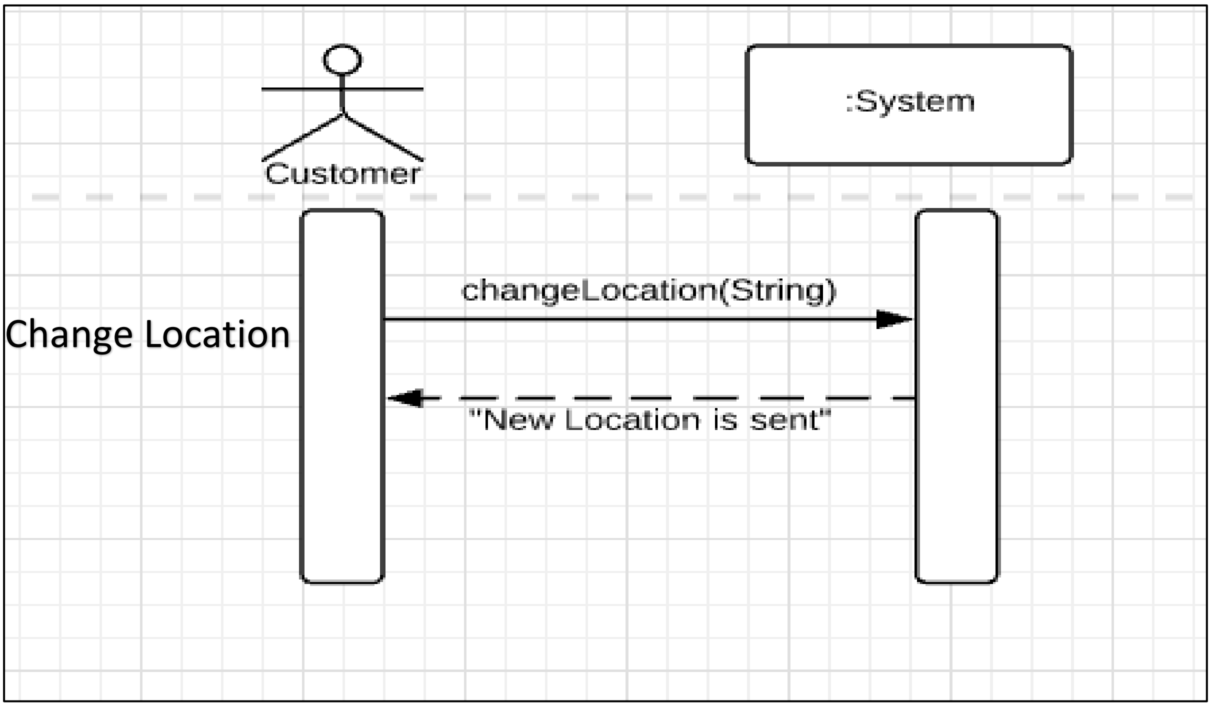


2- Class diagram

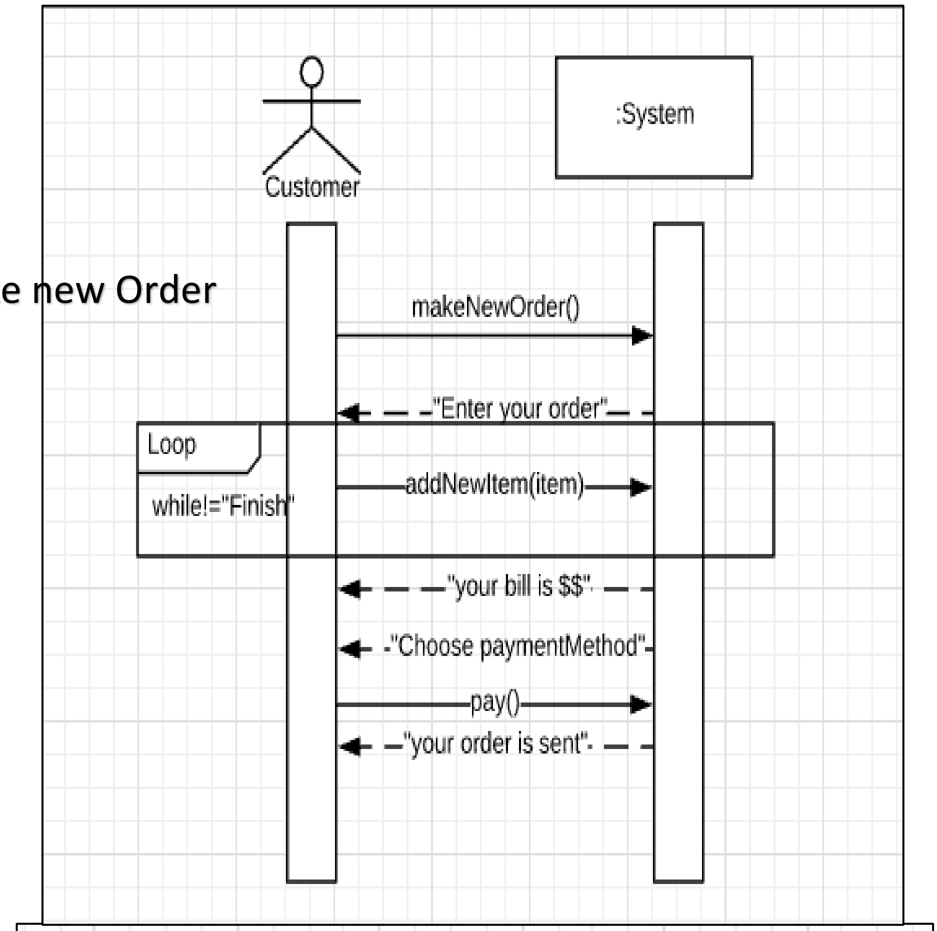


3- SSDs (System Sequence Diagrams)

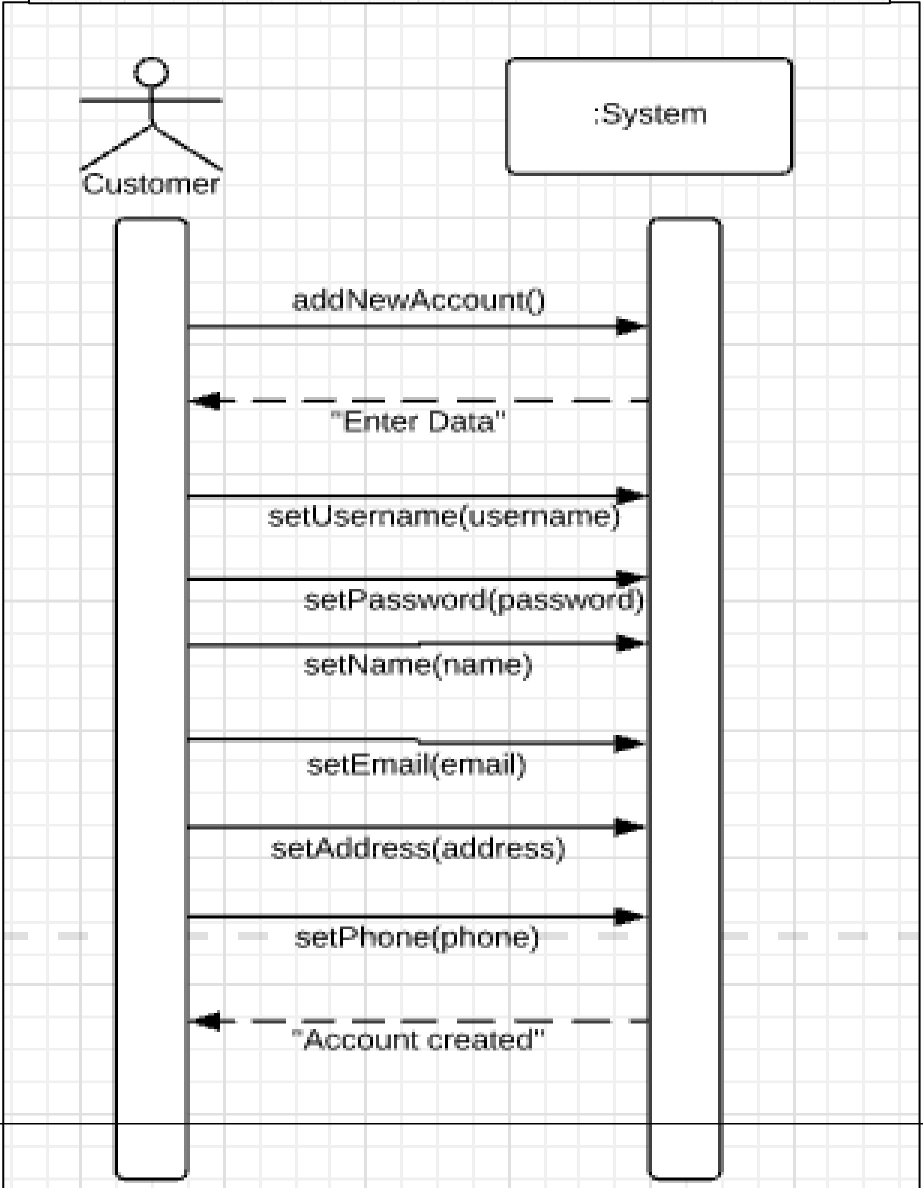


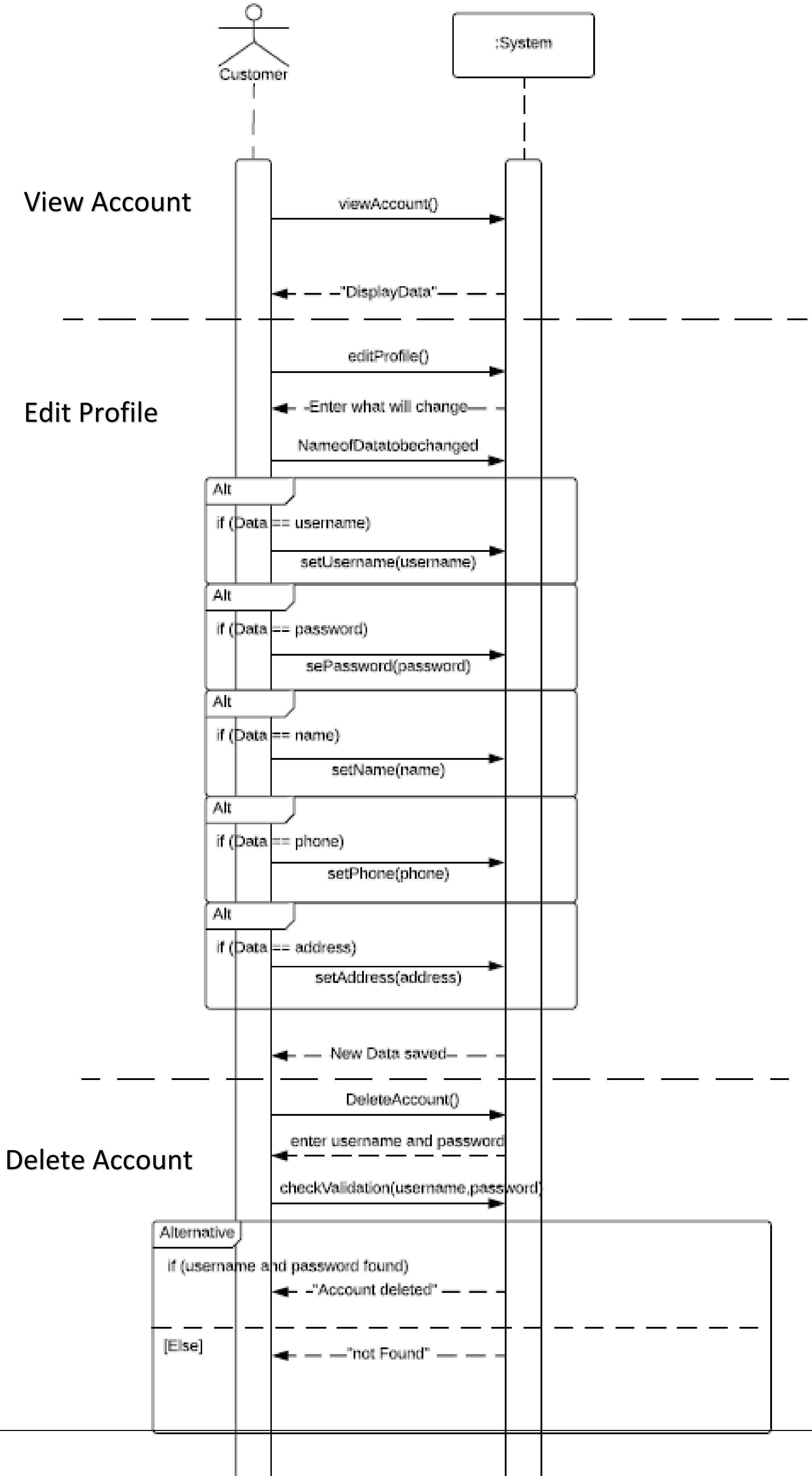


Make new Order



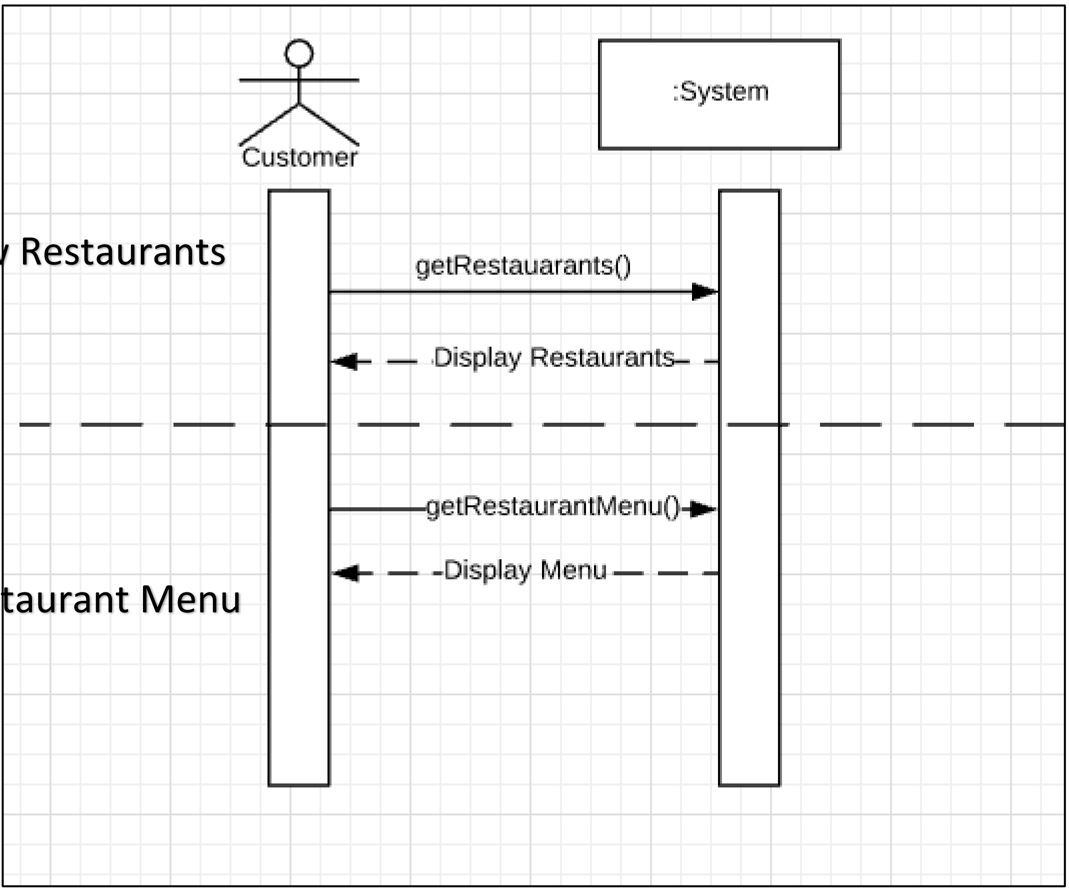
Register

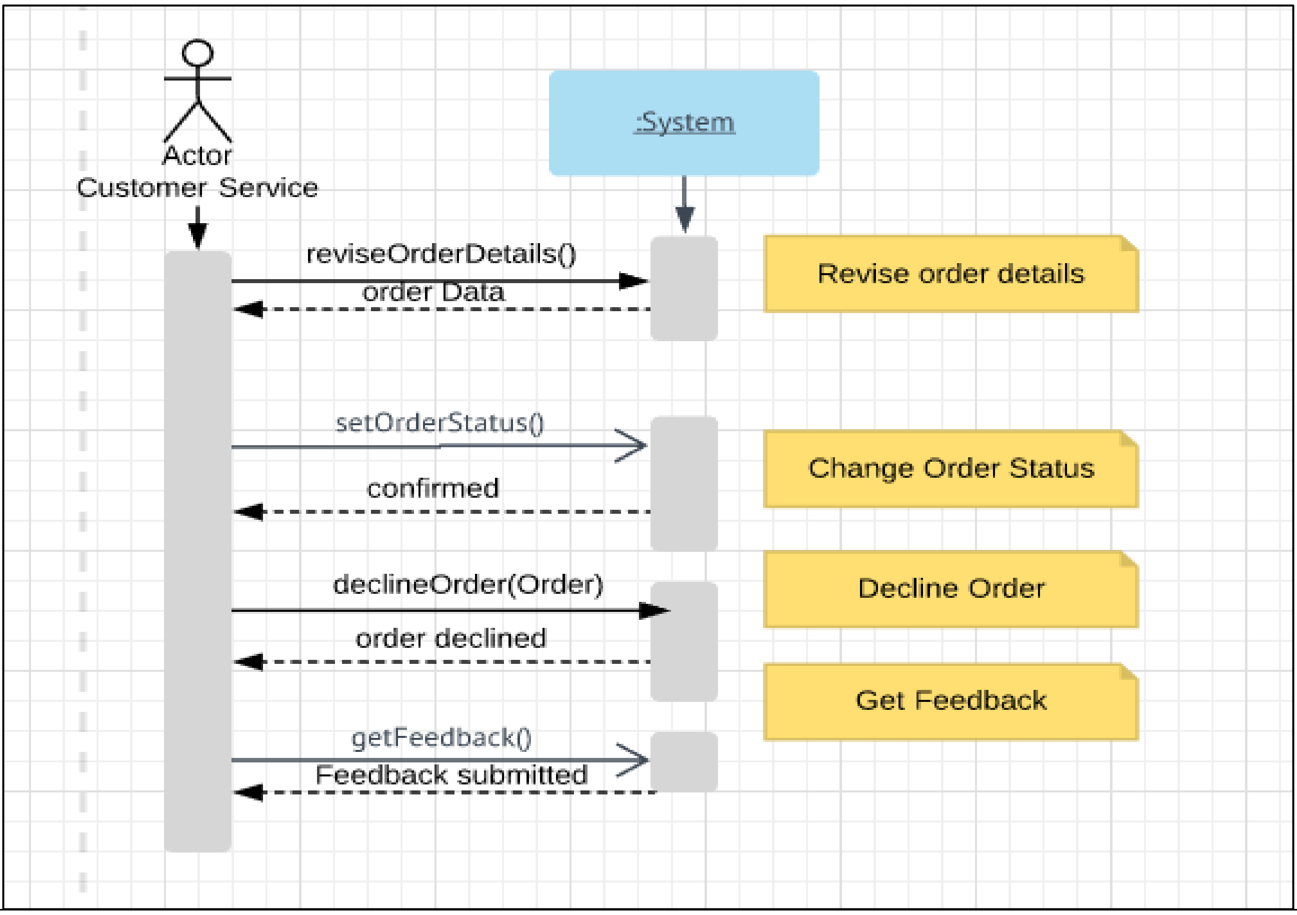


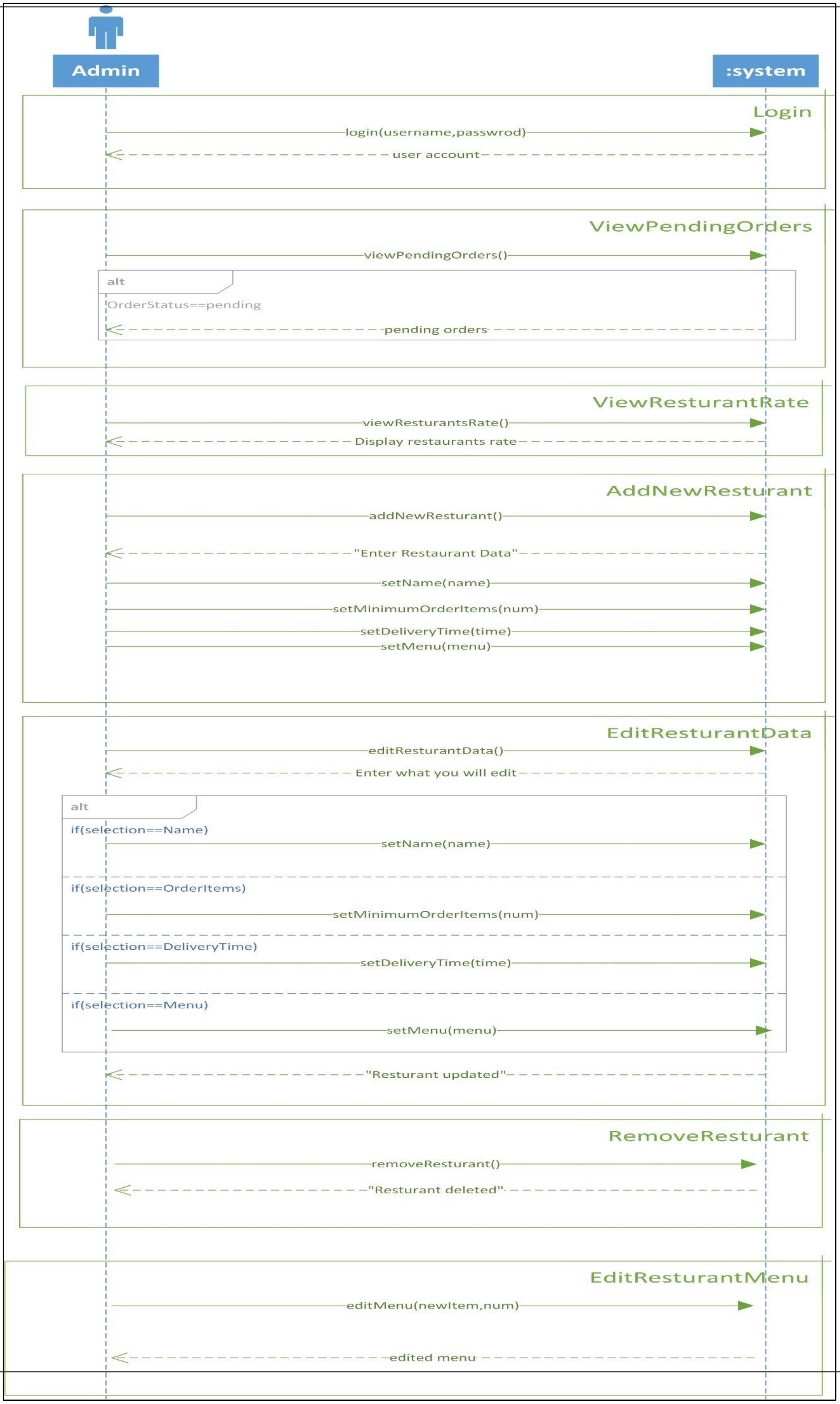

Customer
:System
View Account
viewAccount()
"DisplayData"
editProfile()
Edit Profile
Enter what will change
NameofDatatobechanged
Alt
if (Data == username)
setUsername(username)
Alt
if (Data == password)
sePassword(password)
Alt
if (Data == name)
setName(name)
Alt
if (Data == phone)
setPhone(phone)
Alt
if (Data == address)
setAddress(address)
New Data saved
DeleteAccount()
Delete Account
enter username and password
checkValidation(username,password)
Alternative
if (username and password found)
"Account deleted"
[Else]
"not Found"

View Restaurants

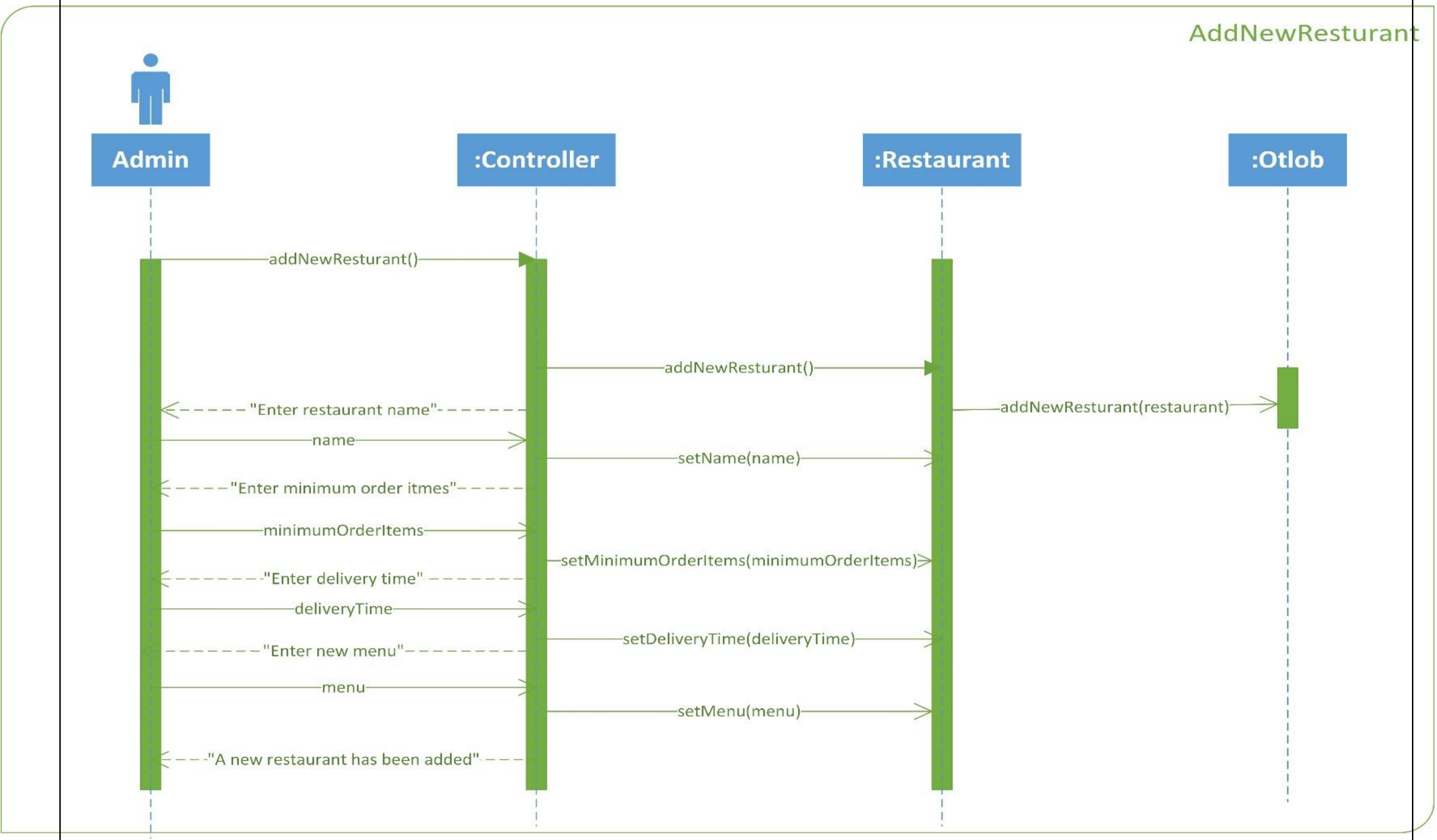
View Restaurant Menu

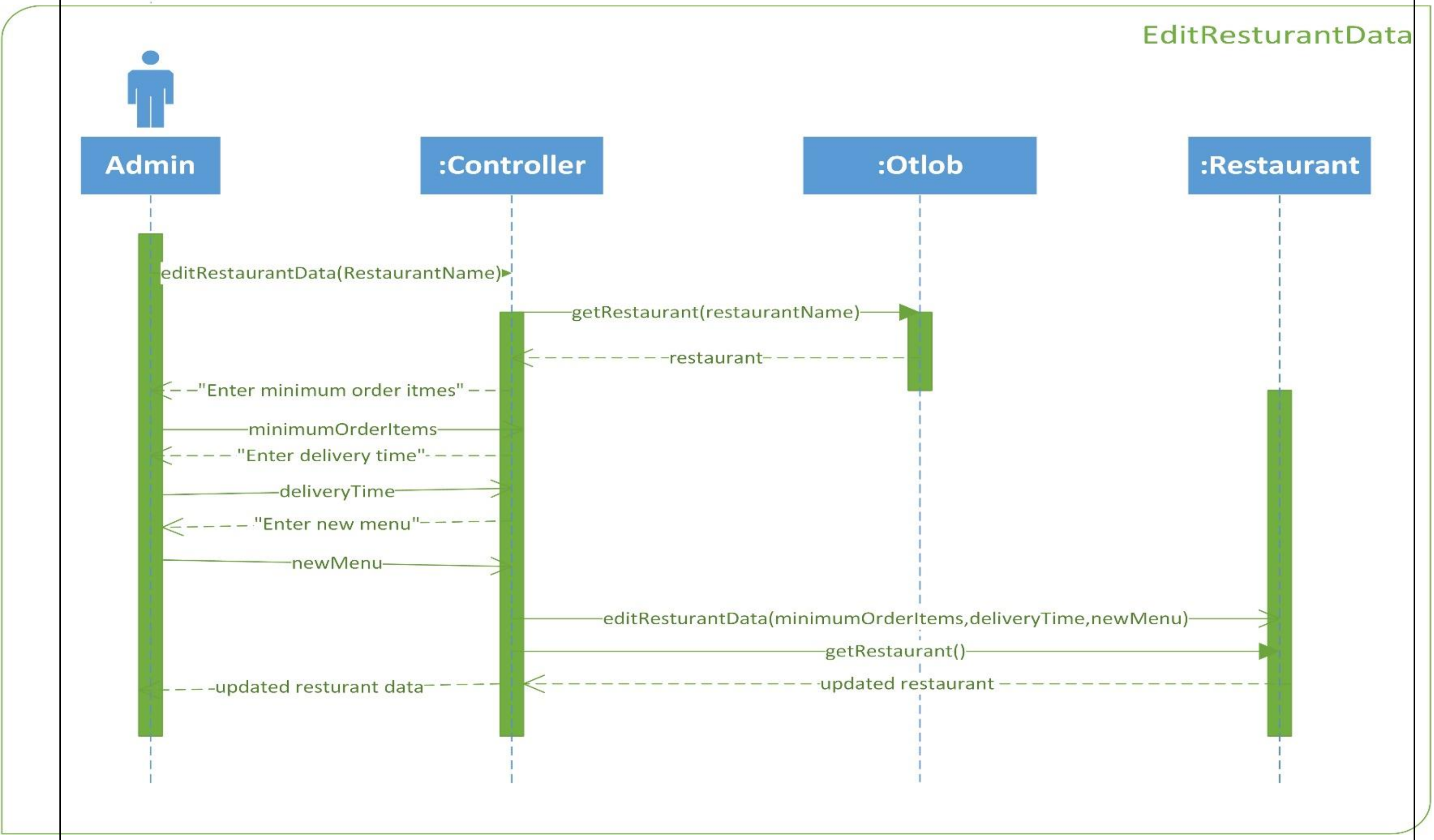


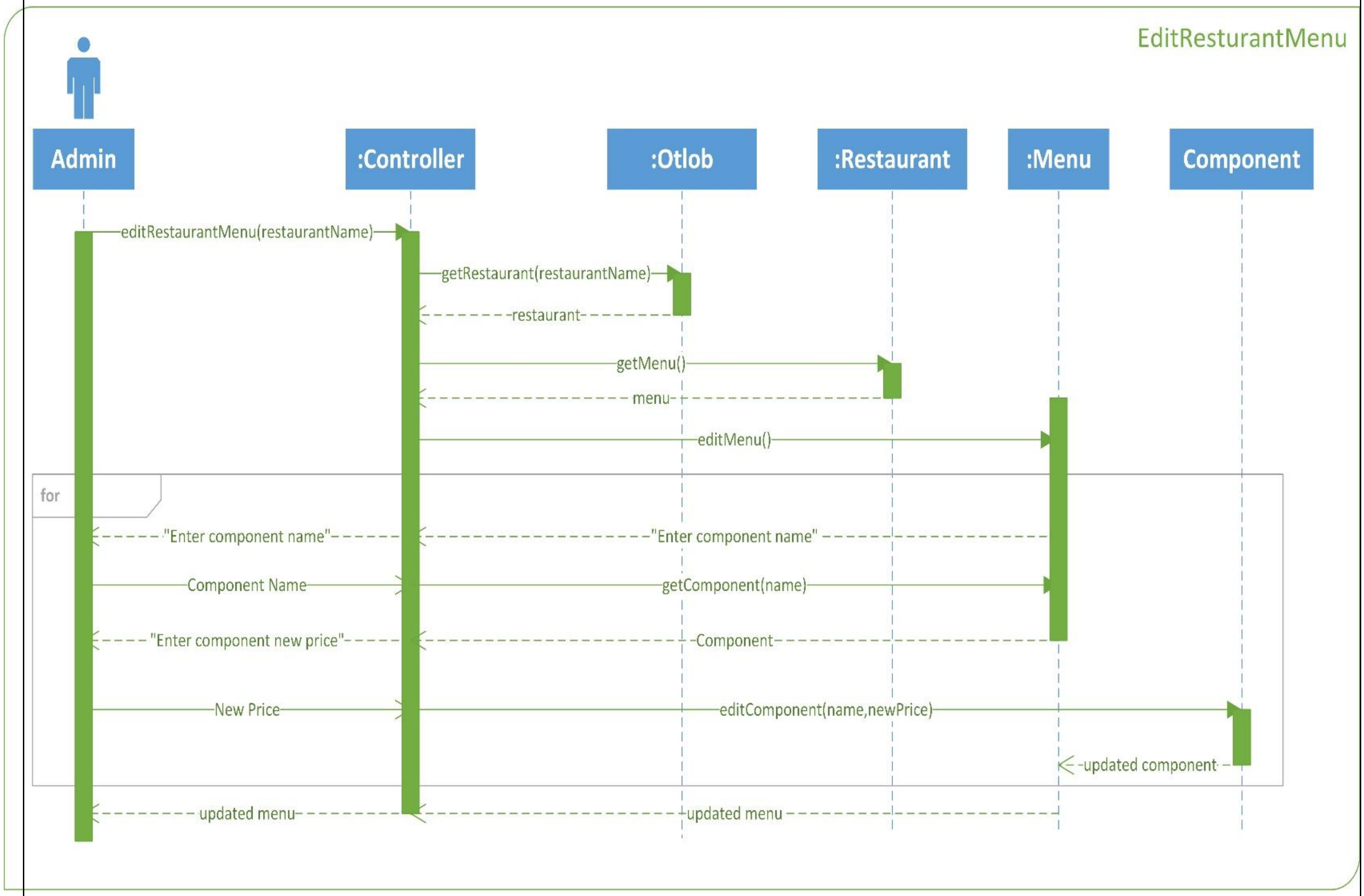


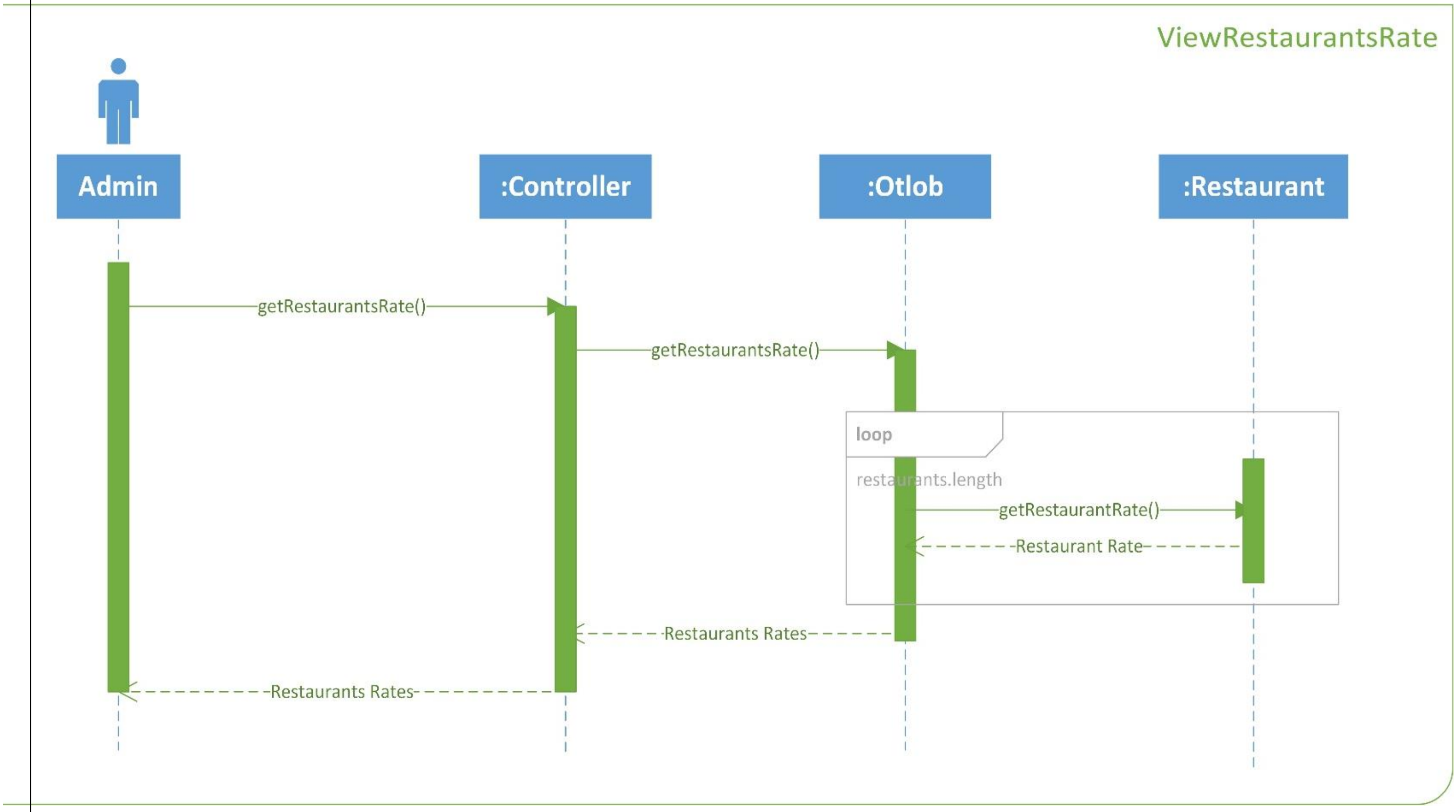

Admin
:system
Login
login(username,passwrod)
user account
ViewPendingOrders
viewPendingOrders()
alt
OrderStatus==pending
pending orders
ViewResturantRate
viewResturantsRate()
Display restaurants rate
AddNewResturant
addNewResturant()
"Enter Restaurant Data"
setName(name)
setMinimumOrderItems(num)
setDeliveryTime(time)
setMenu(menu)
EditResturantData
editResturantData()
Enter what you will edit
alt
if(selection==Name)
setName(name)
if(selection==OrderItems)
setMinimumOrderItems(num)
if(selection==DeliveryTime)
setDeliveryTime(time)
if(selection==Menu)
setMenu(menu)
"Resturant updated"
RemoveResturant
removeResturant()
"Resturant deleted"
EditResturantMenu
editMenu(newItem,num)
edited menu

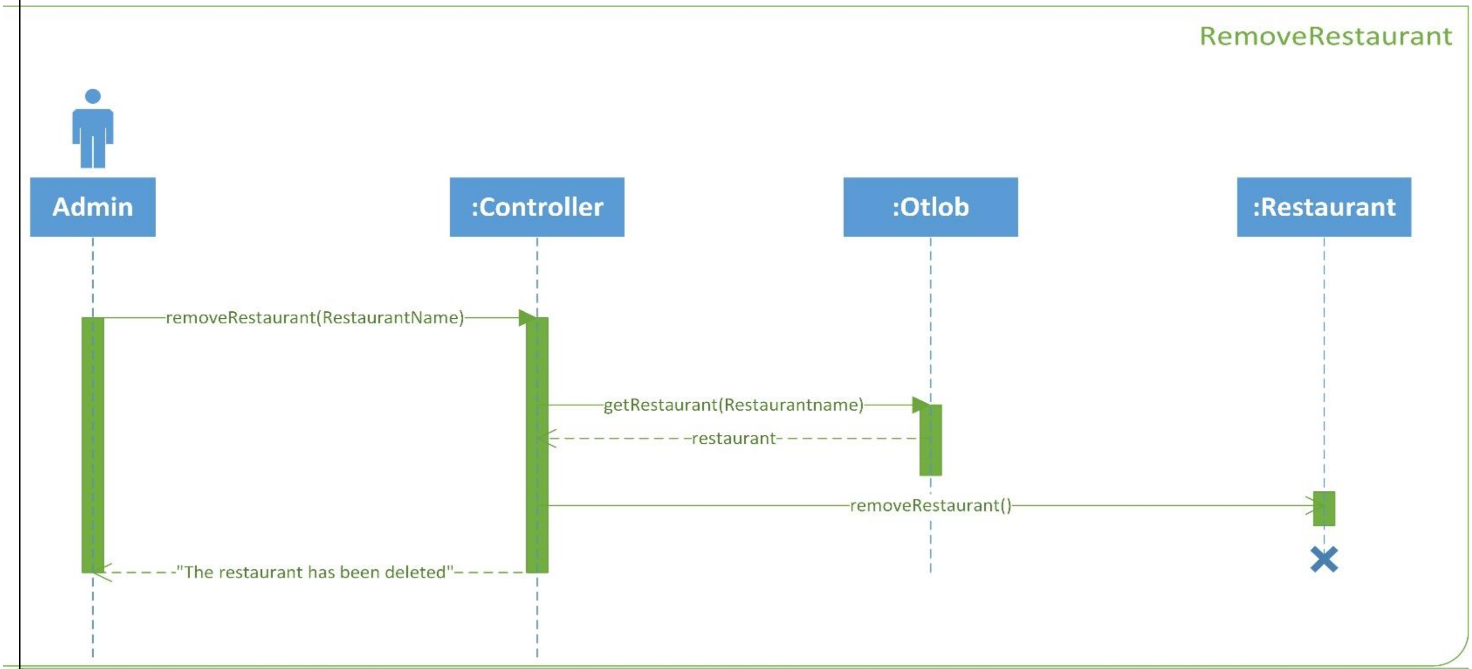
4- Sequence diagrams

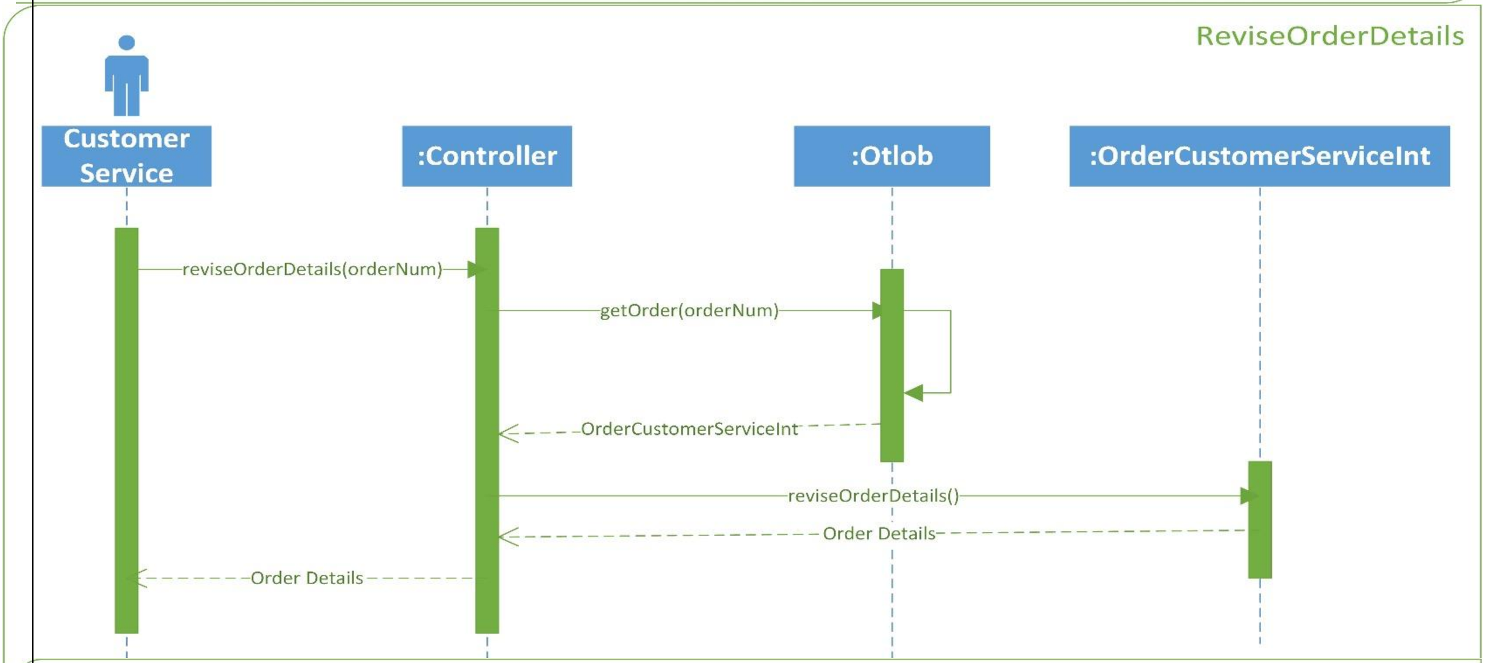


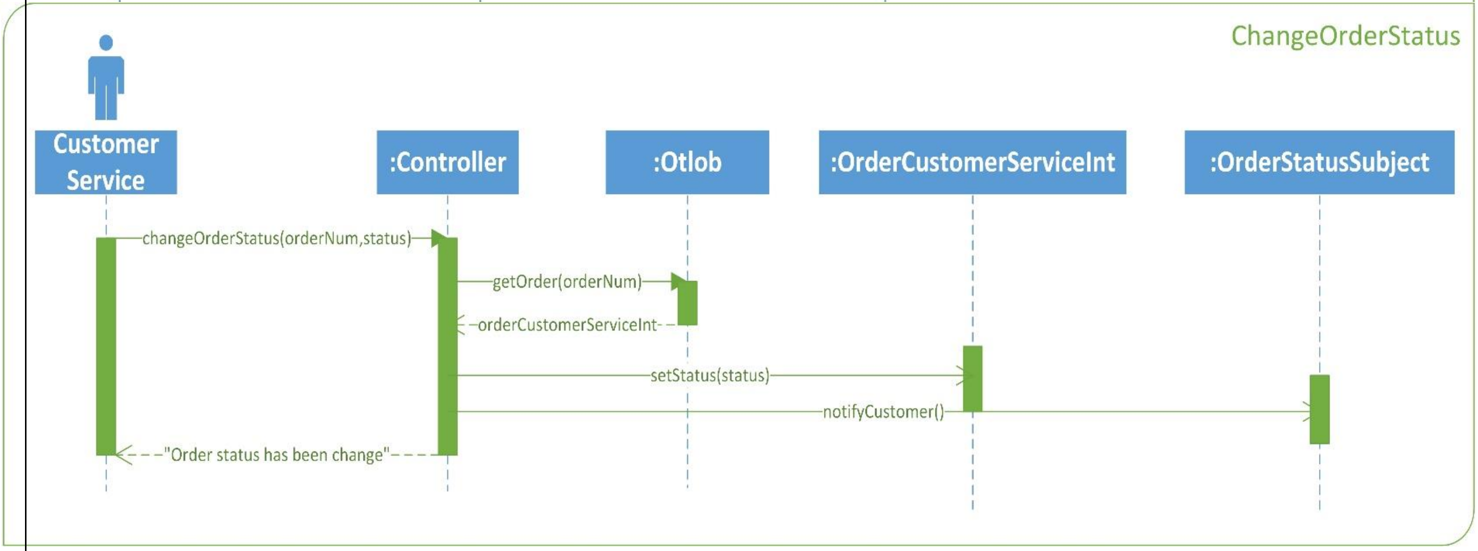


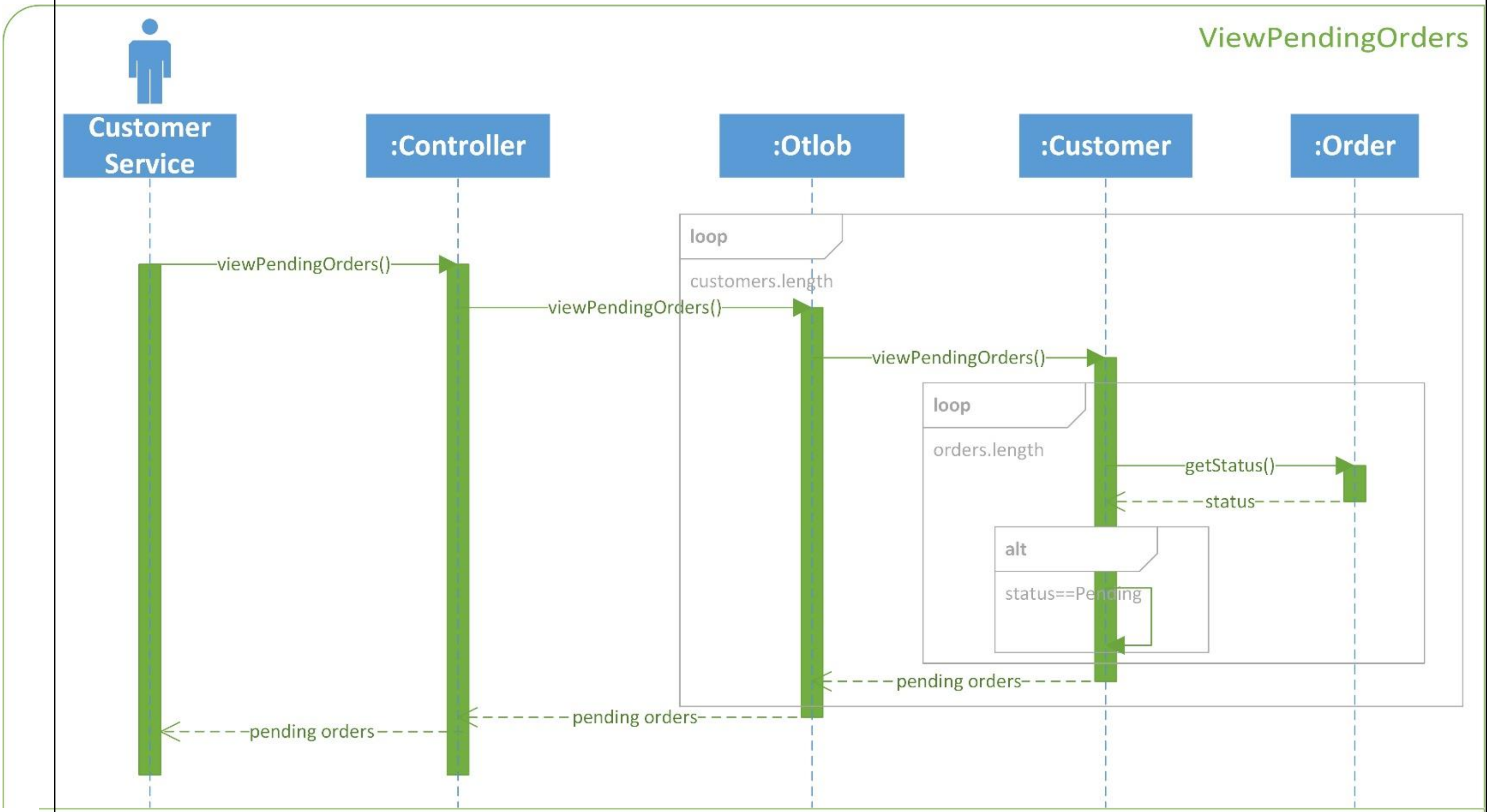

EditResturantMenu
Admin
:Controller
:Otlob
:Restaurant
:Menu
Component
editRestaurantMenu(restaurantName)
getRestaurant(restaurantName)
restaurant
getMenu()
menu
editMenu()
for
"Enter component name"
"Enter component name"
Component Name
getComponent(name)
"Enter component new price"
Component
New Price
editComponent(name,newPrice)
updated component
updated menu
updated menu

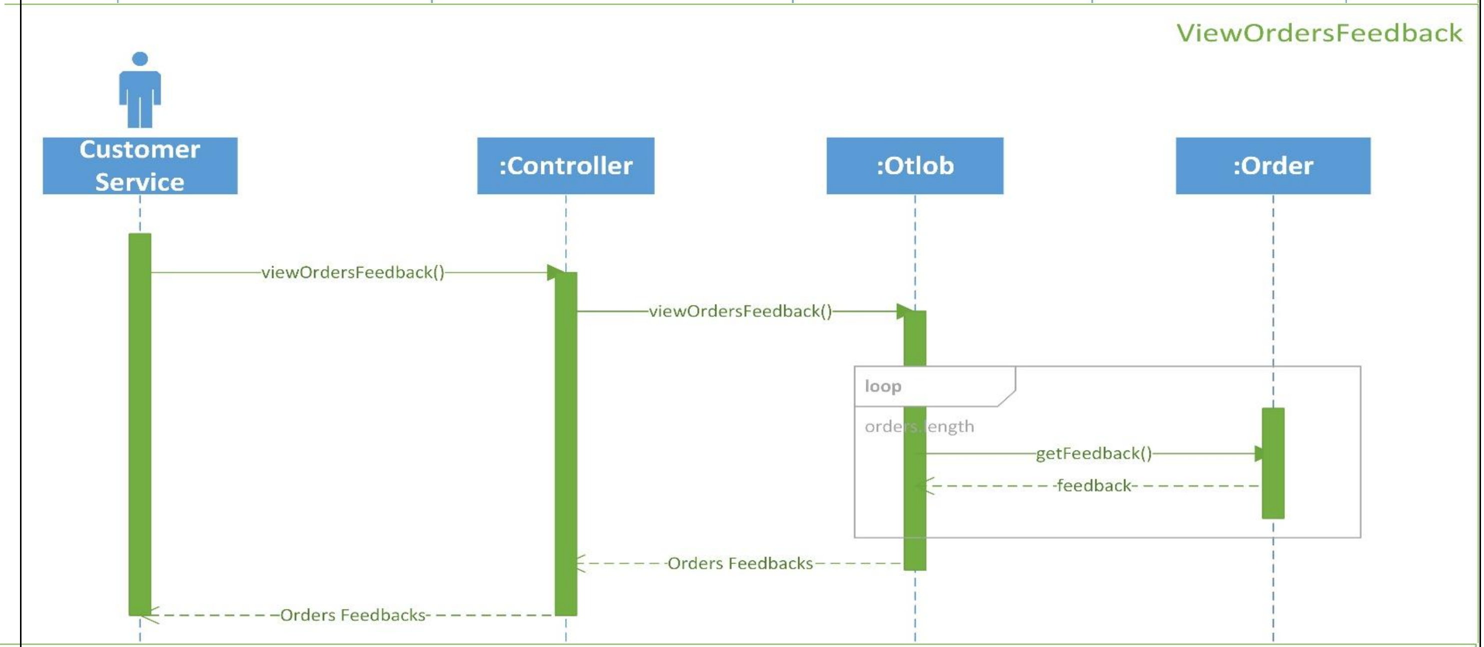

ViewRestaurantsRate
Admin
:Controller
:Otlob
:Restaurant
getRestaurantsRate()
getRestaurantsRate()
loop
restaurants.length
getRestaurantRate()
Restaurant Rate
Restaurants Rates
Restaurants Rates

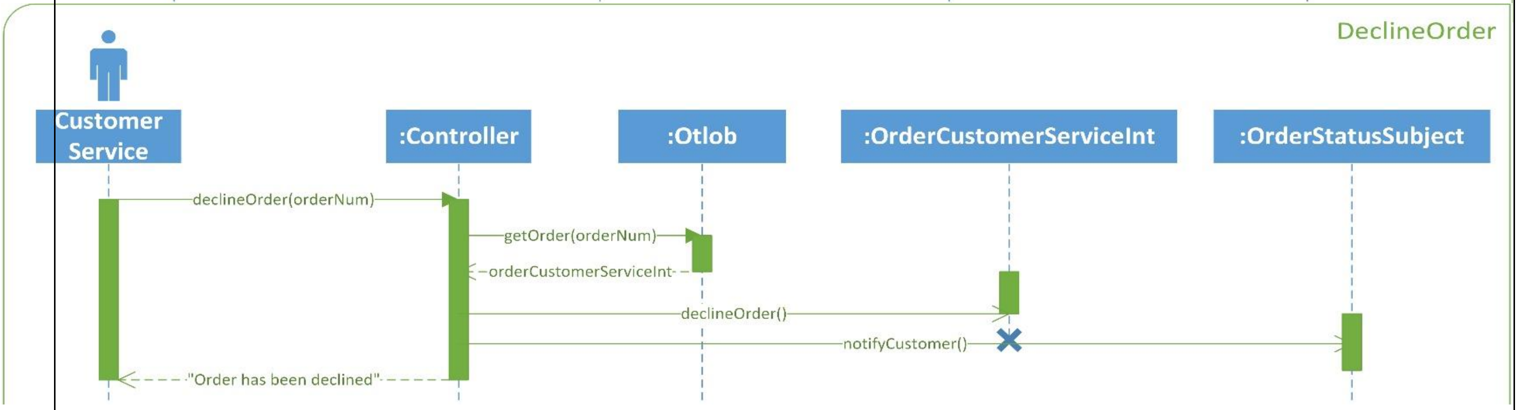

RemoveRestaurant
Admin
:Controller
:Otlob
:Restaurant
removeRestaurant(RestaurantName)
getRestaurant(Restaurantname)
restaurant
removeRestaurant()
"The restaurant has been deleted"

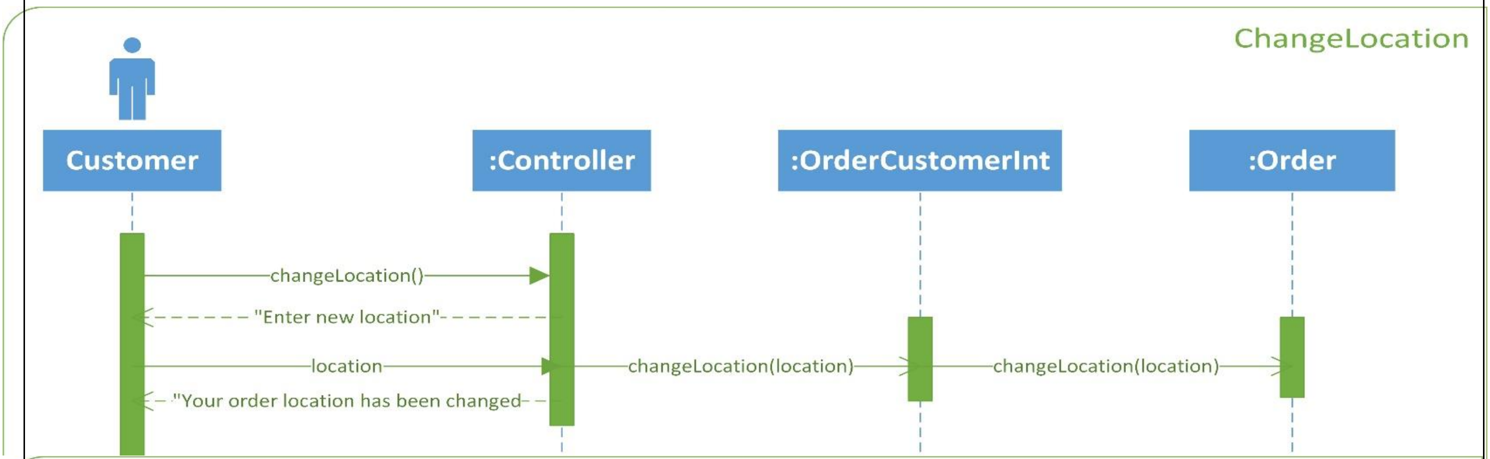

ReviseOrderDetails
Customer Service
:Controller
:Otlob
:OrderCustomerServiceInt
reviseOrderDetails(orderNum)
getOrder(orderNum)
OrderCustomerServiceInt
reviseOrderDetails()
Order Details
Order Details

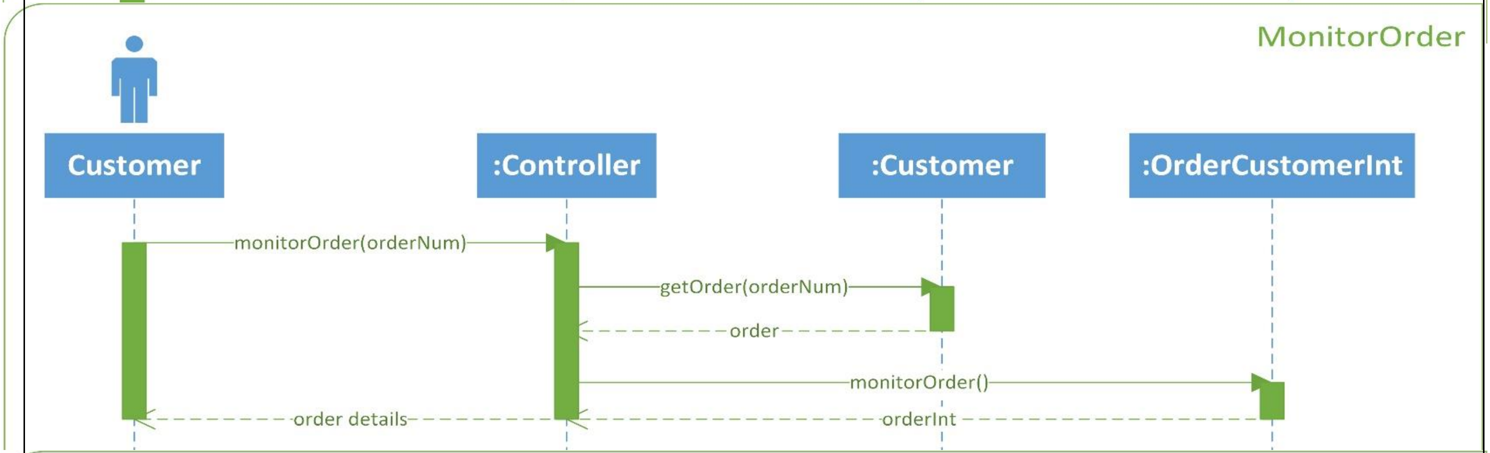

ChangeOrderStatus
Customer Service
:Controller
:Otlob
:OrderCustomerServiceInt
:OrderStatusSubject
changeOrderStatus(orderNum,status)
getOrder(orderNum)
orderCustomerServiceInt
setStatus(status)
notifyCustomer()
"Order status has been change"

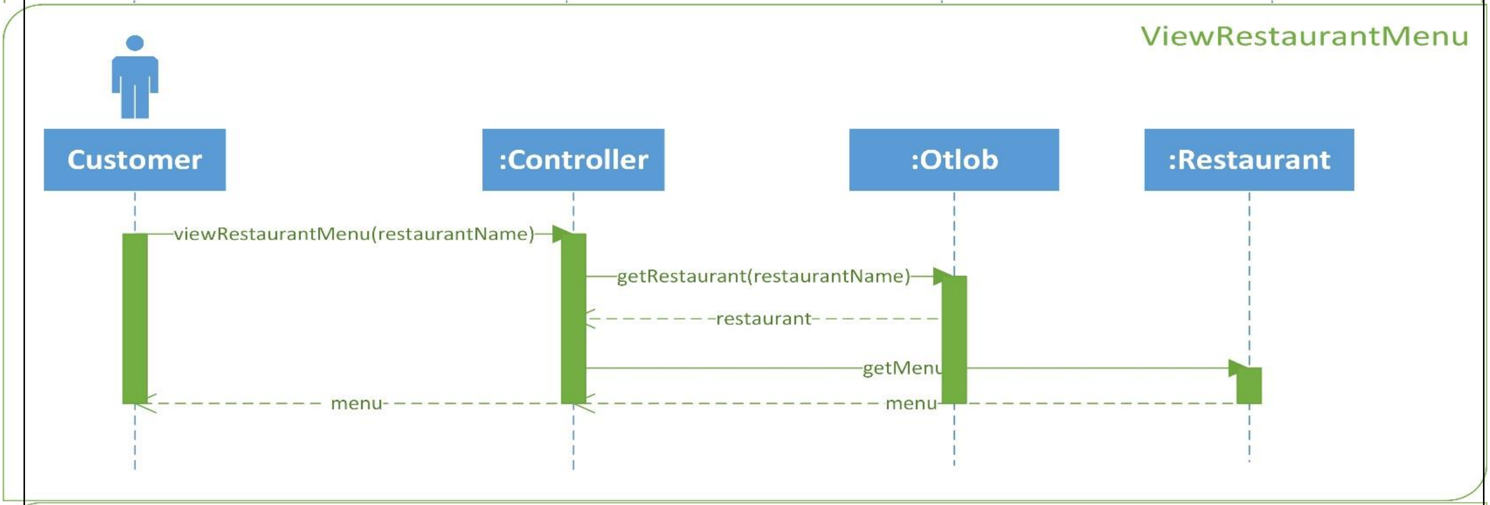

ViewPendingOrders
Customer Service
:Controller
:Otlob
:Customer
:Order
loop
customers.length
viewPendingOrders()
viewPendingOrders()
viewPendingOrders()
loop
orders.length
getStatus()
status
alt
status==Pending
pending orders
pending orders
pending orders

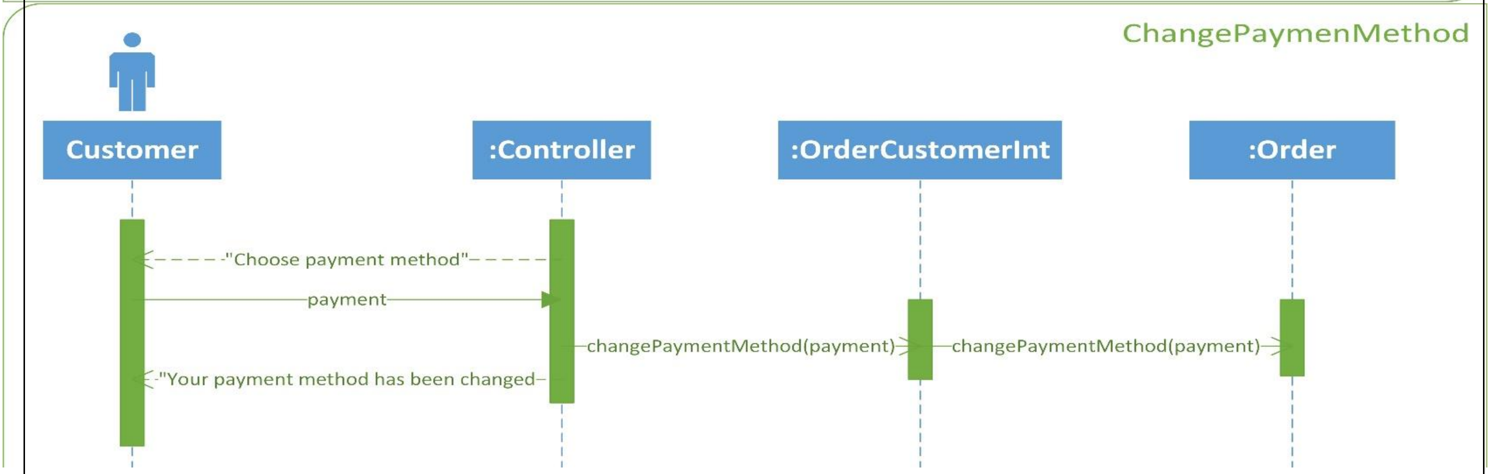

ViewOrdersFeedback
Customer Service
:Controller
:Otlob
:Order
viewOrdersFeedback()
viewOrdersFeedback()
loop
orders.length
getFeedback()
feedback
Orders Feedbacks
Orders Feedbacks

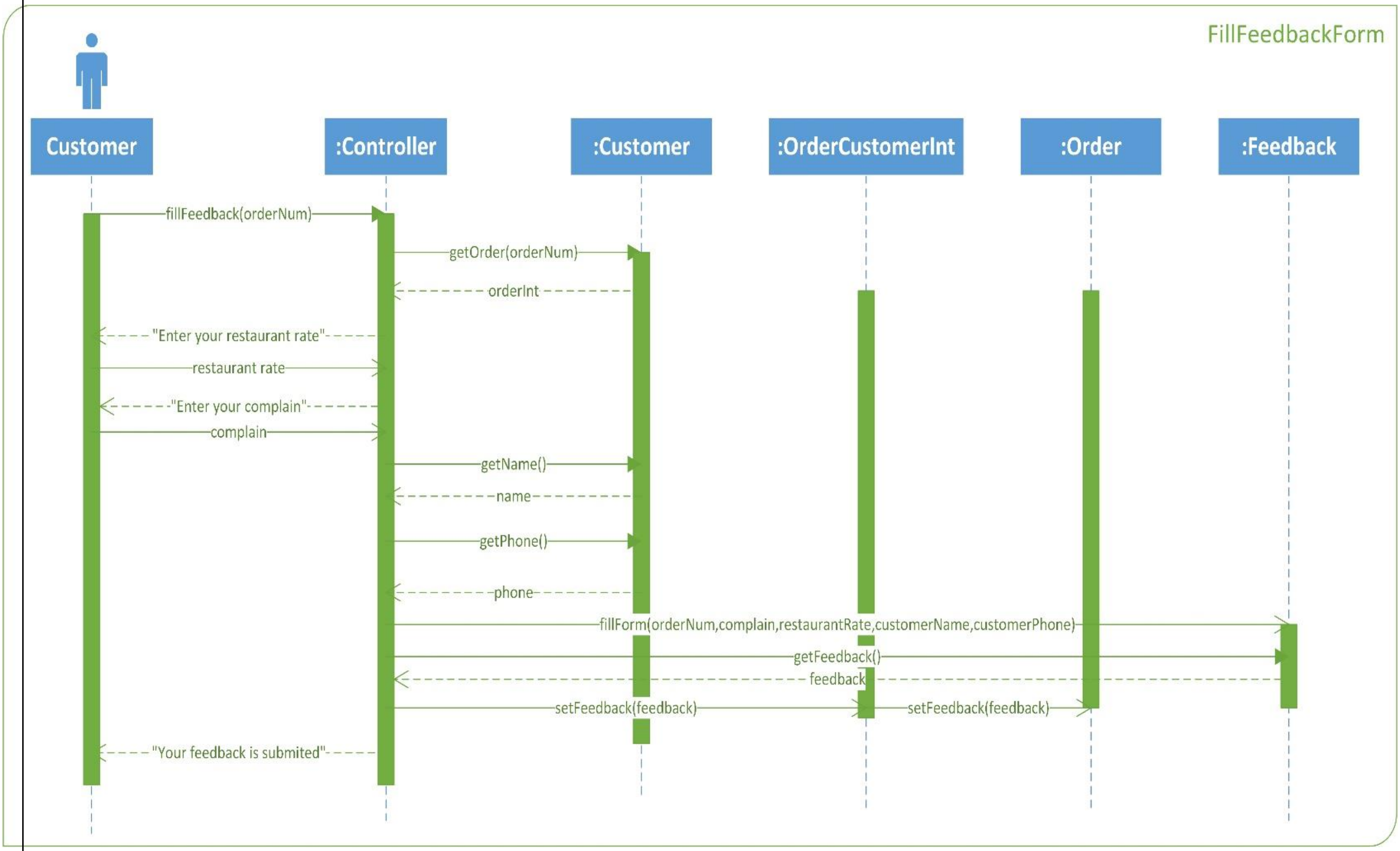

DeclineOrder
Customer Service
:Controller
:Otlob
:OrderCustomerServiceInt
:OrderStatusSubject
declineOrder(orderNum)
getOrder(orderNum)
orderCustomerServiceInt
declineOrder()
notifyCustomer()
"Order has been declined"

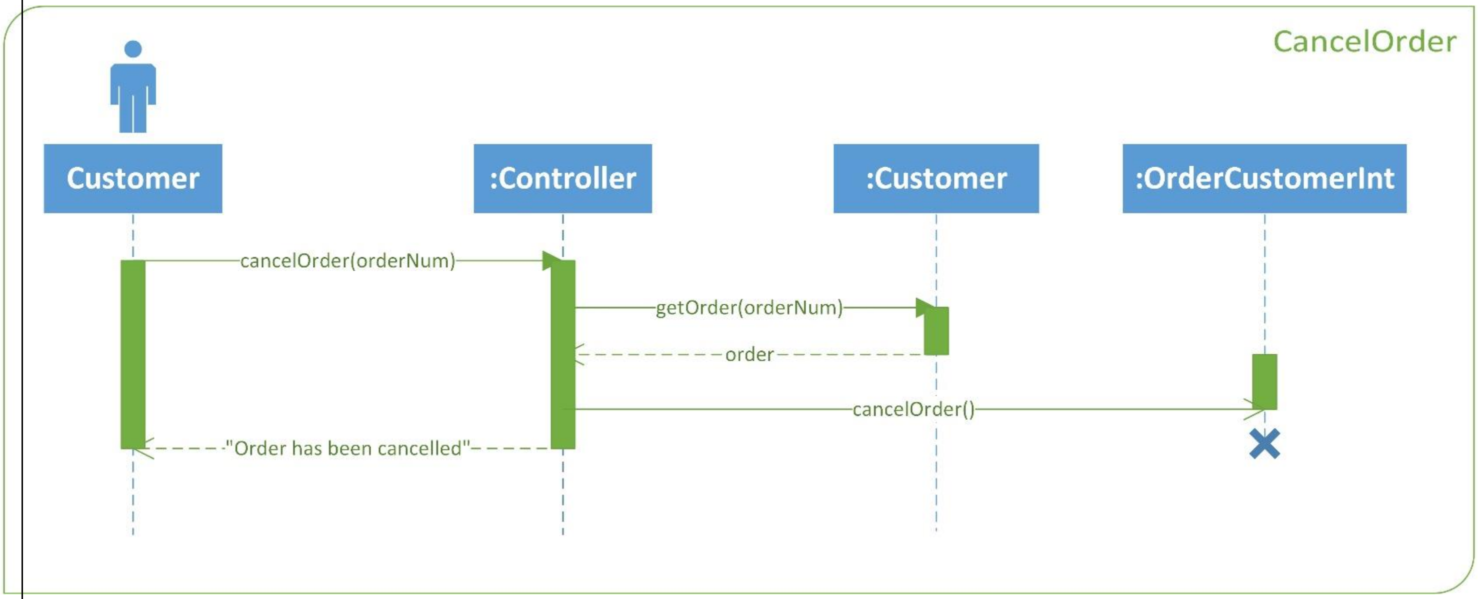

ChangeLocation
Customer
:Controller
:OrderCustomerInt
:Order
changeLocation()
"Enter new location"
location
changeLocation(location)
changeLocation(location)
"Your order location has been changed


MonitorOrder
Customer
:Controller
:Customer
:OrderCustomerInt
monitorOrder(orderNum)
getOrder(orderNum)
order
monitorOrder()
order details
orderInt

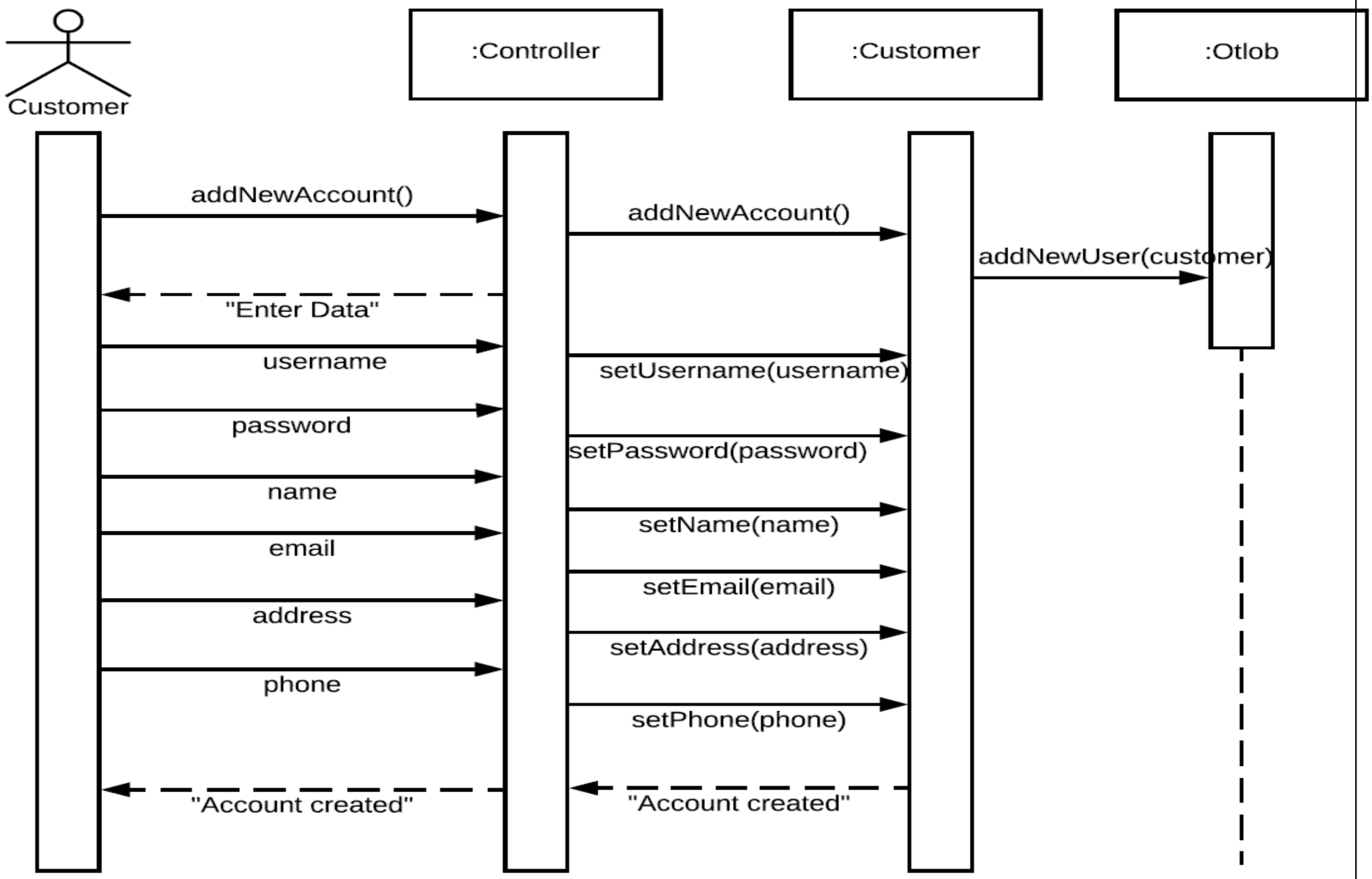

ViewRestaurantMenu
Customer
:Controller
:Otlob
:Restaurant
viewRestaurantMenu(restaurantName)
getRestaurant(restaurantName)
restaurant
getMenu
menu
menu


ChangePaymenMethod
Customer
:Controller
:OrderCustomerInt
:Order
"Choose payment method"
payment
changePaymentMethod(payment)
changePaymentMethod(payment)
"Your payment method has been changed

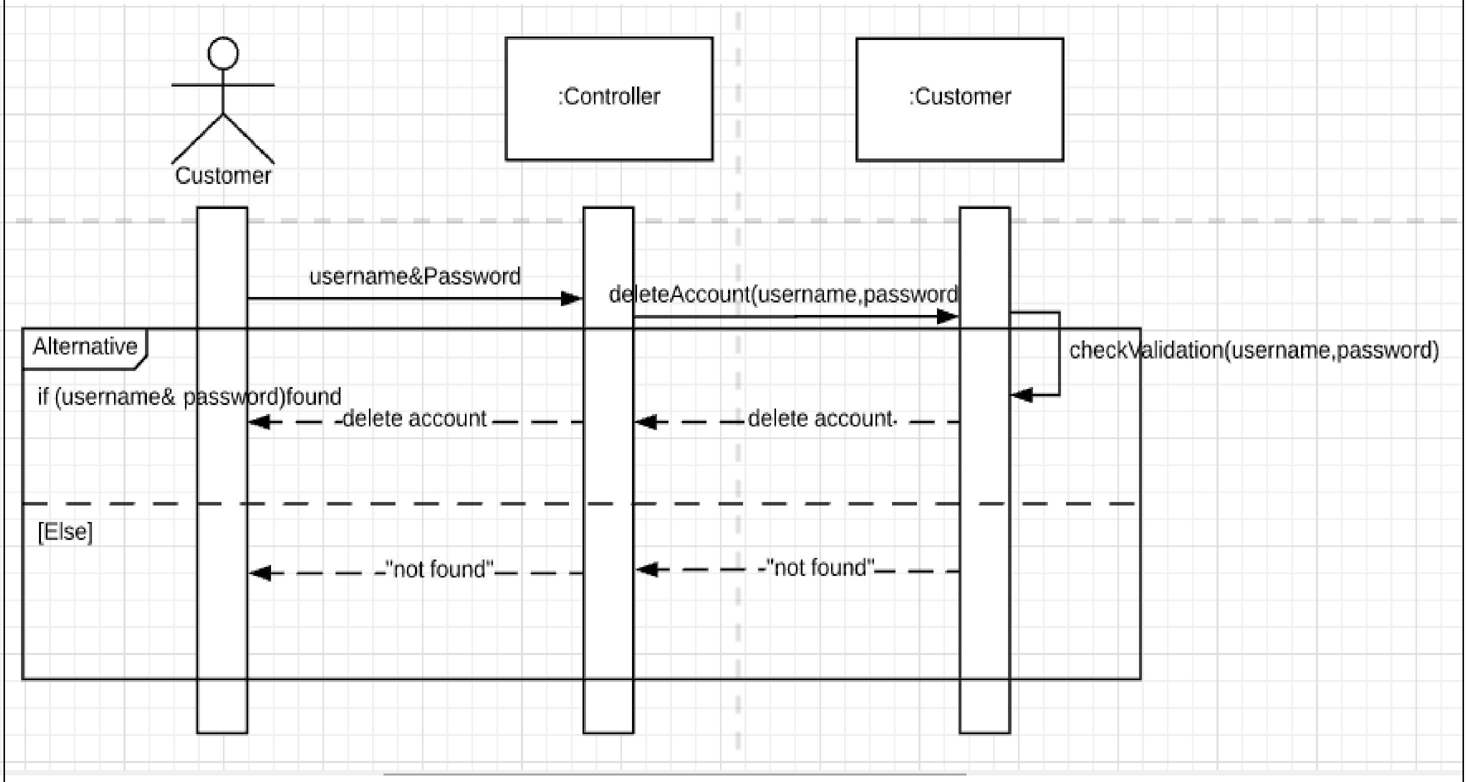

FillFeedbackForm
Customer
:Controller
:Customer
:OrderCustomerInt
:Order
:Feedback
fillFeedback(orderNum)
getOrder(orderNum)
orderInt
"Enter your restaurant rate"
restaurant rate
"Enter your complain"
complain
getName()
name
getPhone()
phone
fillForm(orderNum,complain,restaurantRate,customerName,customerPhone)
getFeedback()
feedback
setFeedback(feedback)
setFeedback(feedback)
"Your feedback is submited"


CancelOrder
Customer
:Controller
:Customer
:OrderCustomerInt
cancelOrder(orderNum)
getOrder(orderNum)
order
cancelOrder()
"Order has been cancelled"

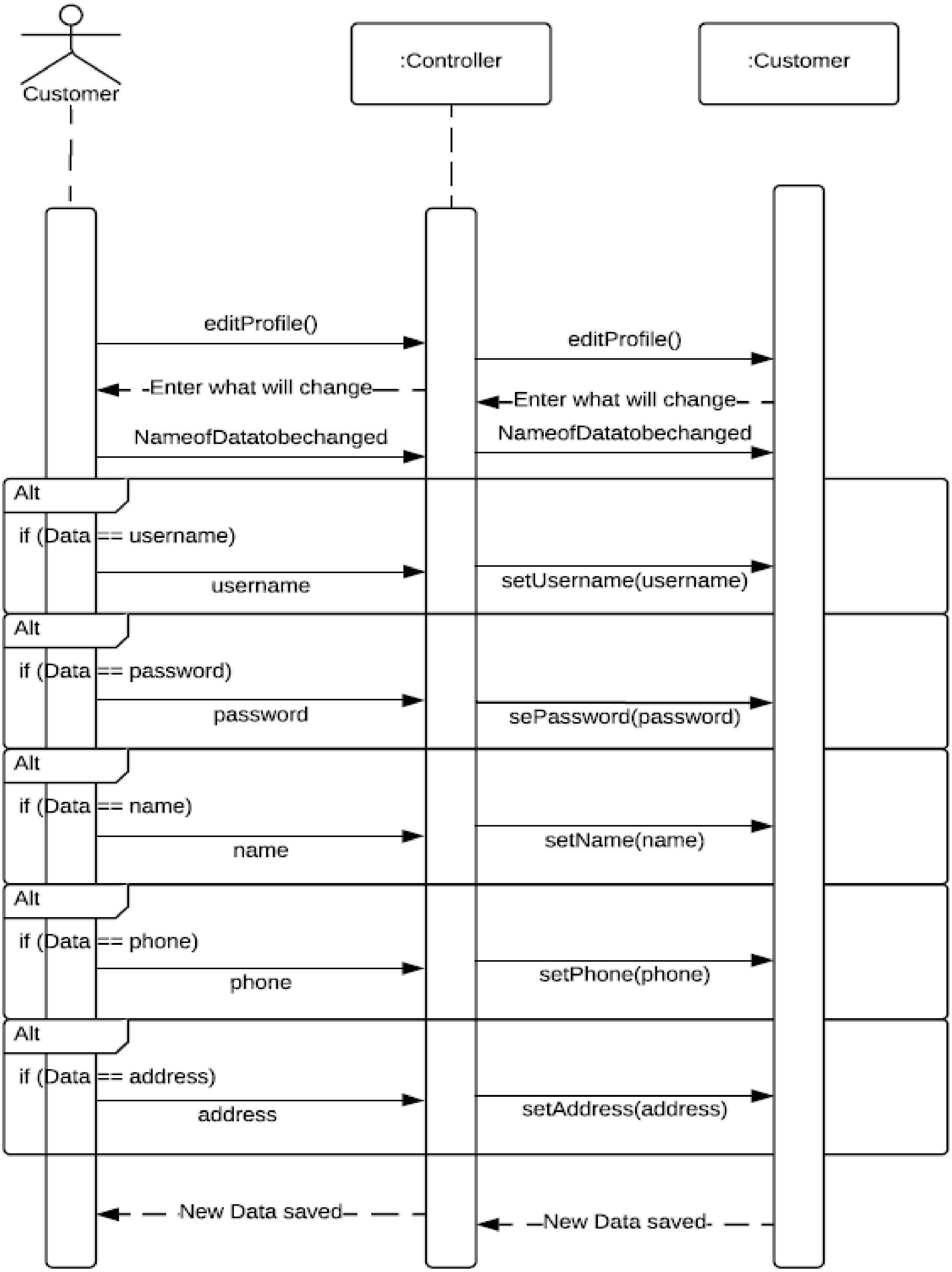
-Register



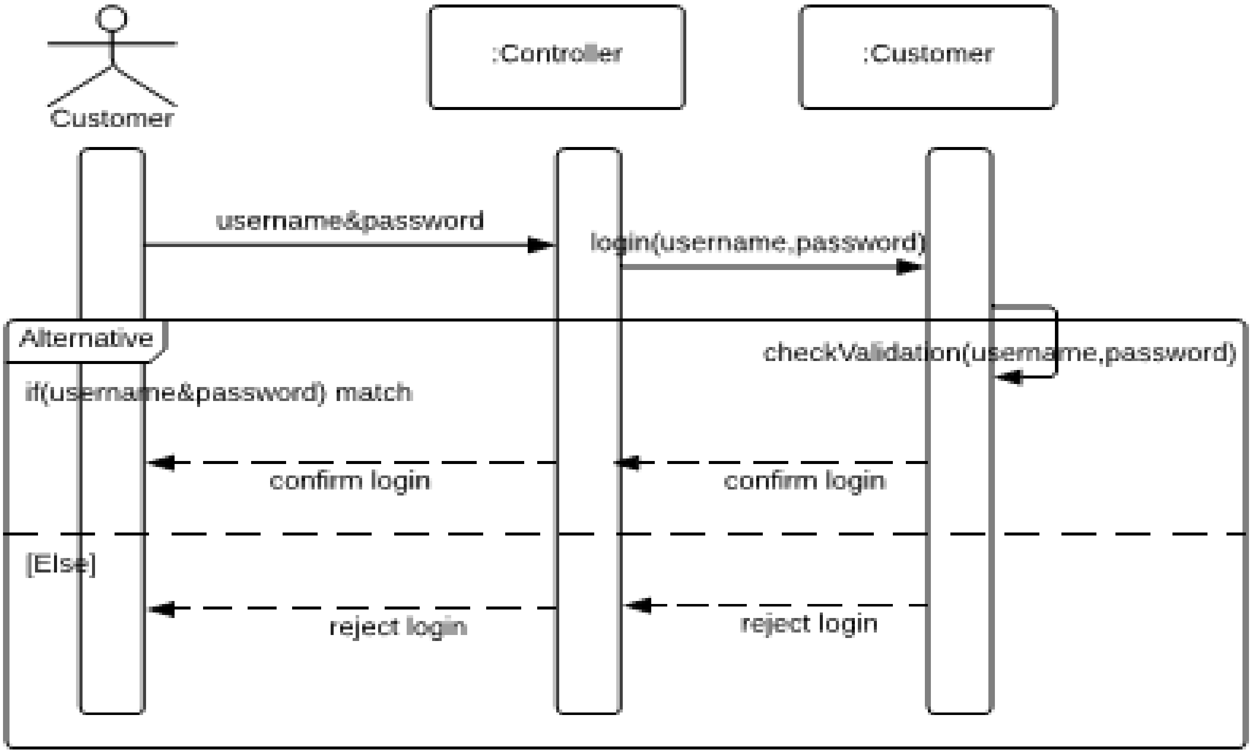
-Delete account



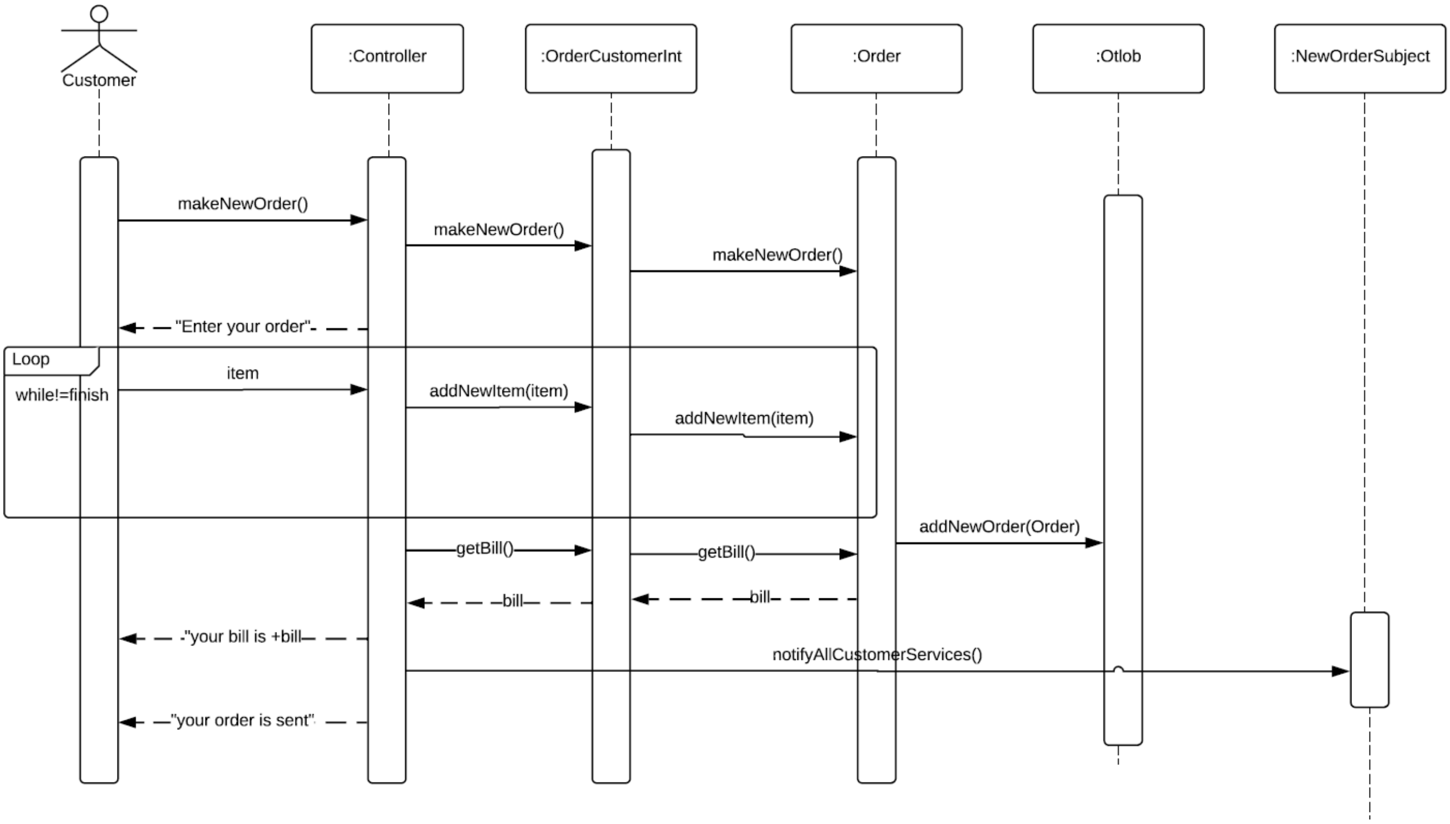
-Edit profile



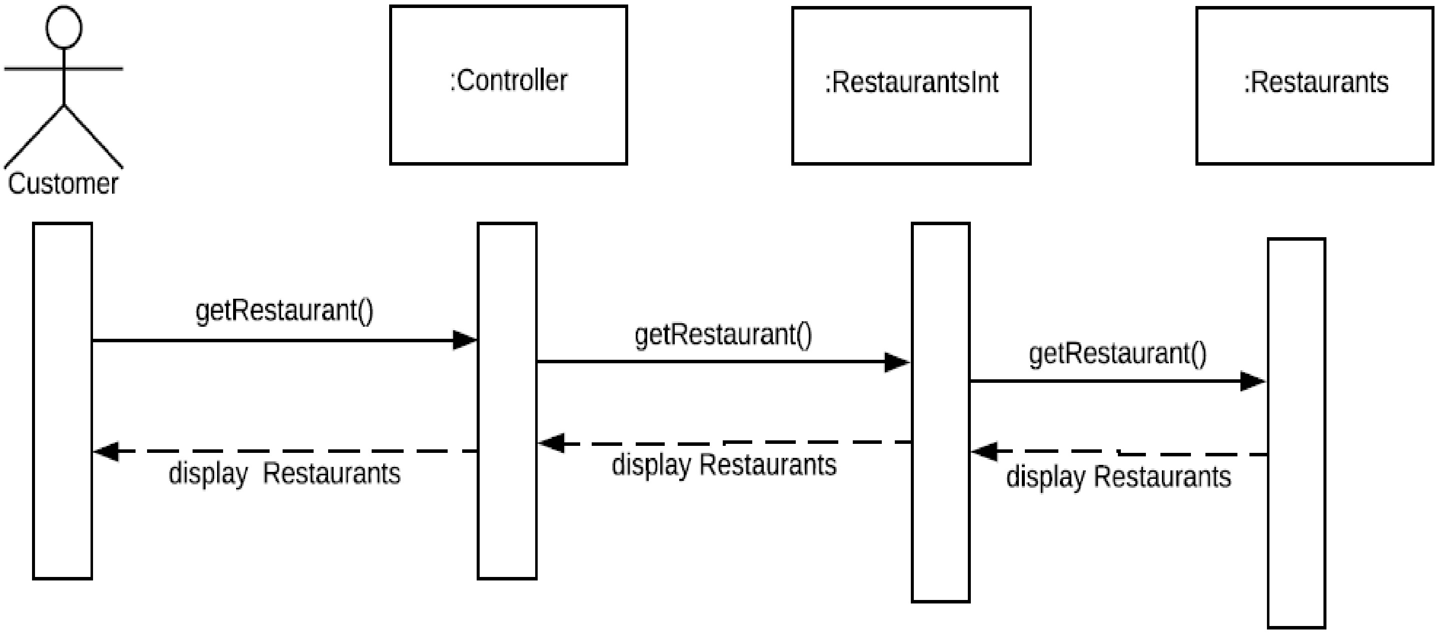
-Login



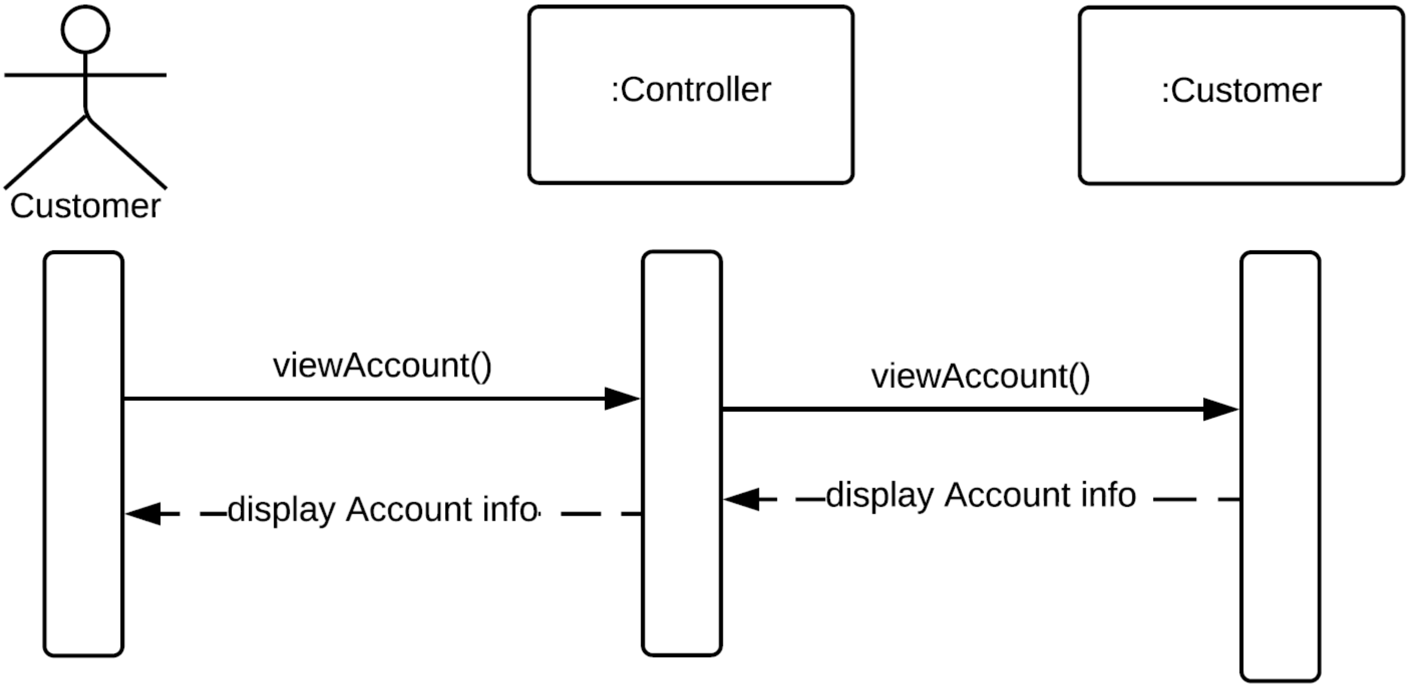
-Make new order



-View Restaurants



-View Account



5- Design patterns

- **Singleton**: Admin, Otlob

  In our system (otlob application) we only have one admin, so we will have only one instance for this class.

  Same for Otlob class, which is a class that holds all the users, restaurants and orders data.

- **Strategy**: Payment

  (PayPal, credit card, cash) classes have the same method with different implementation, so we used the strategy pattern.

- **Composite**: Menu, Food, Component

  A menu can have sub menu and the menu have array of food. So that the menu is treated uniformly like food.

- **Read-only**: -Order

  The order class is accessed by two users (customer, customer service), each user has access on specific methods, each one has his interface.

  -Restaurant

  The restaurant class is accessed by all users (customer, customer service, admin), each user has access on specific methods, admin has his own interface unlike customer and customer service who share the same interface.

- **Observer**: -Order, Customer

  When the order status is changed, or when the order is declined, a notification is sent to the customer.

  -Order, Customer Service

  When a new order is created, a notification is sent to the customer service.

- **Delegation**: Otlob

  This class contains objects of different classes and it uses their methods.

- **Immutable**: Bill

  No one can change the bill data.