# Experiment 6: Sketch Sequence diagram for the project

**Learning Objective:** Students will able to draw Sequence diagram for the project

**Tools:** Dia, StarUML

**Theory:**

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

**Sequence Diagram representation**

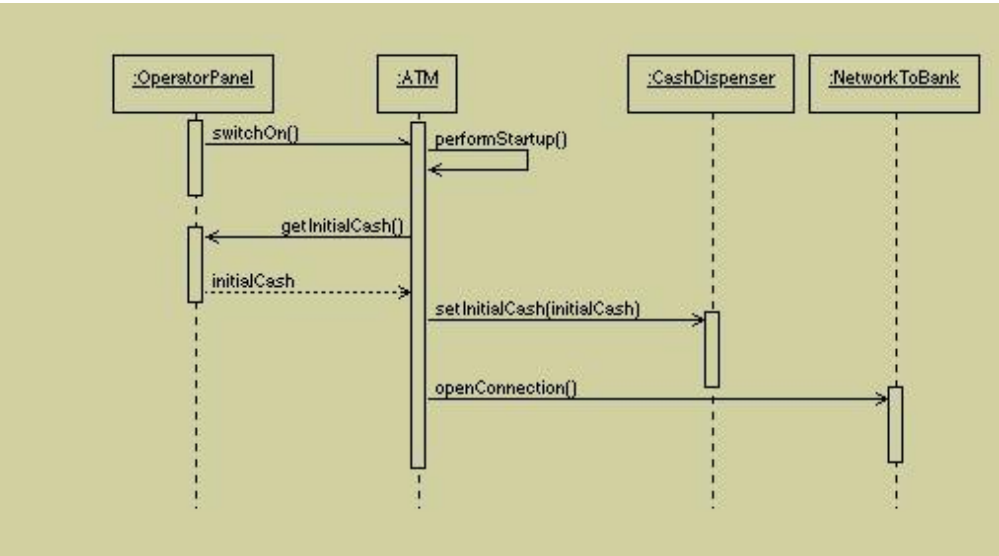
Call Message: A message defines a particular communication between Lifelines of an Interaction.

Destroy Message: Destroy message is a kind of message that represents the request of destroying the lifecycle of target lifeline.

Lifeline: A lifeline represents an individual participant in the Interaction.

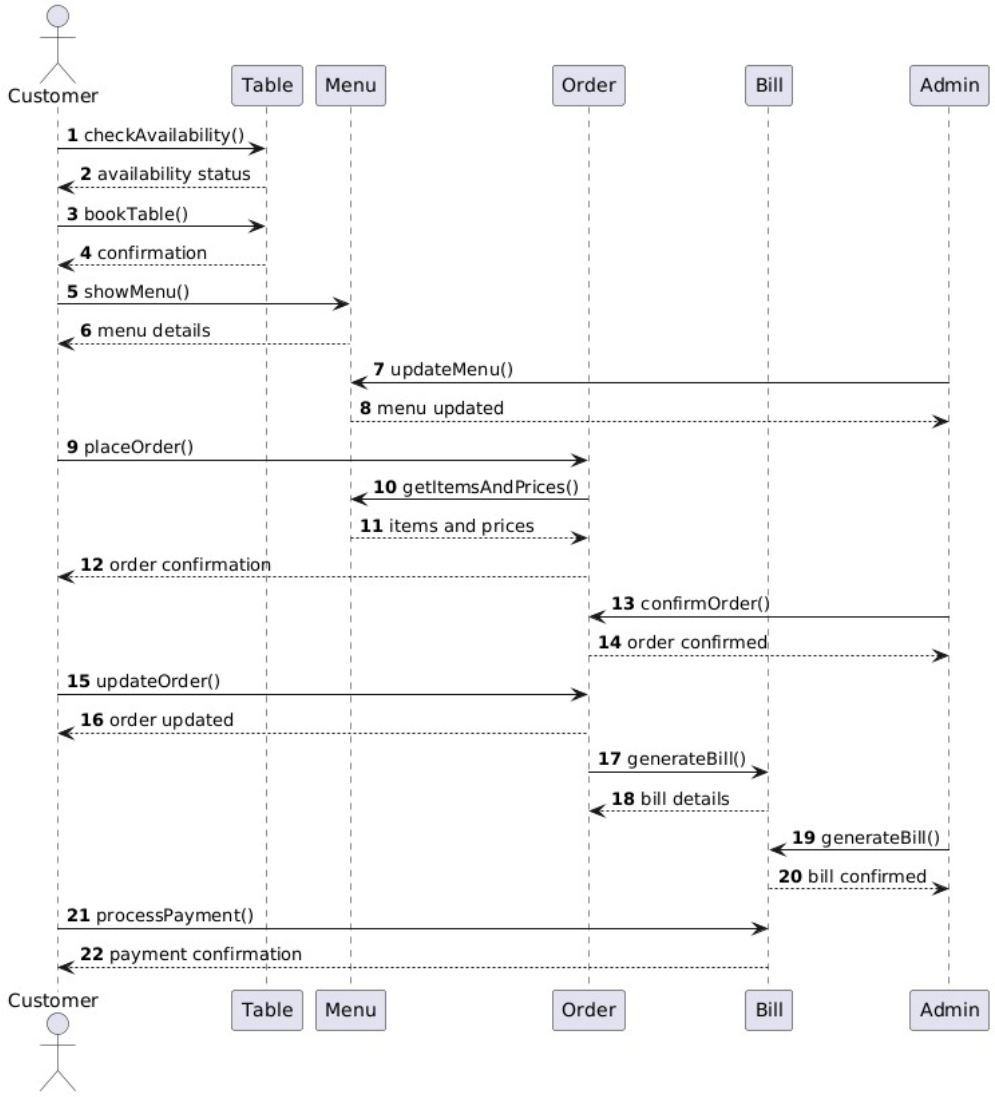
Recursive Message: Recursive message is a kind of message that represents the invocation of message of the same lifeline. Its target points to an activation on top of the activation where the message was invoked from.

**Sequence Diagram: Example for ATM System startup**




It is clear that sequence charts have a number of very powerful advantages. They clearly depict the sequence of events, show when objects are created and destroyed, are excellent at depicting concurrent operations, and are invaluable for hunting down race conditions. However, with all their advantages, they are not perfect tools. They take up a lot of space, and do not present the interrelationships between the collaborating objects very well.

## Result and Discussion:




Sequence Diagram Description

**1. Table Booking**

The Customer checks the availability of a table by calling checkAvailability().
The Table responds with the availability status.
If available, the Customer books the table by calling bookTable(), and the Table confirms the booking.

**2. Menu Viewing and Updating**

The Customer requests the menu by calling showMenu() on the Menu component.
The Menu responds by displaying available items and prices.
Meanwhile, the Admin updates the menu using updateMenu(), ensuring the latest offerings are available.

**3. Placing an Order**

The Customer places an order by calling placeOrder() on the Order component.
The Order fetches item details and prices from the Menu.

The Menu returns the requested information, and the Order confirms the order with the Customer.

**4. Order Verification by Admin**
The Admin verifies the order by calling confirmOrder() on the Order component.
The Order confirms the verification back to the Admin.

**5. Updating an Order**
If needed, the Customer can modify their order by calling updateOrder() on the Order component.
The Order updates the changes and confirms them with the Customer.

**6. Bill Generation and Payment**
The Order generates a bill by calling generateBill() on the Bill component.
The Bill returns the bill details to the Order.
The Admin also calls generateBill() to verify and confirm the final bill.
The Customer makes a payment by calling processPayment() on the Bill.
The Bill processes the payment and confirms the transaction with the Customer.

## Learning Outcomes:

LO1: Identify the classes and objects.
LO2: Identify the interactions between the objects
LO3: Develop a sequence diagram for different scenarios

**Course Outcomes:** Upon completion of the course students will be able to draw the sequence diagram for the project.

## Conclusion:

For Faculty Use:

| **Correction Parameters** | **Formative Assessment [40%]** | **Timely completion of Practical [ 40%]** | **Attendance / Learning Attitude [20%]** | |
|---|---|---|---|---|
| **Marks Obtained** | | | | |