UML

# Objective:

To create the following UML:

* sequence diagram

for a real project or system.

# Theory:

**UML (UNIFIED MODELING LANGUAGE):**

**UML** (Unified Modeling Language) is a modeling language used by software developers. UML can be used to develop diagrams and provide users with ready-to-use, expressive modeling examples. Some UML tools generate program language code from UML. UML can be used for modeling a system independent of a platform language. UML is a graphical language for visualizing, specifying, constructing, and documenting information about software-intensive systems. UML gives a standard way to write a system model, covering conceptual ideas. With an understanding of modeling, the use and application of UML can make the software development process more efficient.

There are two categories of UML:

* behavior diagrams
* use case diagram
* sequence diagram
* structured diagrams
* class diagram
* **Sequence diagram:**

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

**COMPONENTs OF A SEQUENCE DIAGRAM:**

* Class Roles or Participants. Class roles describe the way an object will behave in context.
* Activation or Execution Occurrence. Activation boxes represent the time an object needs to complete a task.
* Messages.
* Lifelines.
* Destroying Objects.
* Loops.

**HOW TO CREATE A SEQUENCE DIAGRAM:**

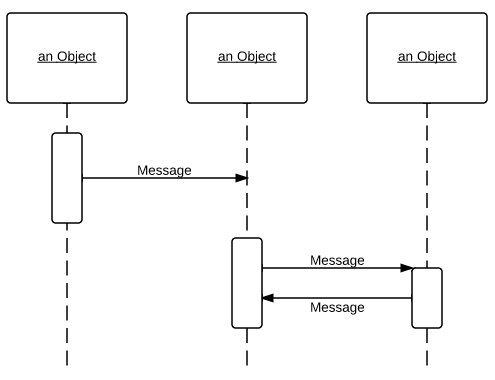
Sequence diagramming really is visual coding, even when you are modeling a usage scenario via a system-level sequence diagram.

While creating a sequence diagram, start by identifying the scope of what you are trying to model. You should typically tackle small usage scenarios at the system level or a single method/service at the detailed object level.

You should then work through the logic with at least one more person, laying out classifiers across the top as you need them. . The heart of the diagram is in the messages, which you add to the diagram one at a time as you work through the logic. You should rarely indicate return values, instead you should give messages intelligent names which often make it clear what is being returned.

It is interesting to note that as you sequence diagram you will identify new responsibilities for classes and objects, and, sometimes, even new classes. The implication is that you may want to update your class model appropriately, agile modelers will follow the practice Create Several Models in Parallel, something that CASE tools will do automatically. Remember, each message sent to a class invokes a static method/operation on that class each message sent to an object invokes an operation on that object.

Regarding style issues for sequence diagramming, prefer drawing messages going from left-to-right and return values from right-to-left, although that doesn’t always work with complex objects/classes. Justify the label on messages and return values, so they are closest to the arrowhead. Also prefer to layer the sequence diagrams: from left-to-right. indicate the actors, then the controller class(es), and then the user interface class(es), and, finally, the business class(es). During design, you probably need to add system and persistence classes, which you should usually put on the right-most side of sequence diagrams. Laying your sequence diagrams in this manner often makes them easier to read and also makes it easier to find layering logic problems, such as user interface classes directly accessing persistence.



**Exercise**

Draw sequence diagram for the following scenario:

Implement Sequence diagrams with C#

* Hotel management system

**Problem statement:**

A hotel has various types of rooms Dimensions of various: price, number of single beds, number of double beds. A database with a listing of all the rooms of the hotel is supplied. This database includes when the rooms have been booked.

People can look for availability on a website for certain types of room (room price), for a certain time of span. The systems check availability and returns the proposition that fits the reservation. If no exact match is found, something similar is proposed with at least the same person capacity.

* Airline reservation system

**Problem Statement:**

To develop a computerized meeting, the rising customer interest in booking online air travel reservations. The system should be convenient, user friendly and available via the internet.

The system should allow the users to view entire flights information of the airline, book tickets, view or, if required cancel current reservations and create member login for standalone users as well as agents.