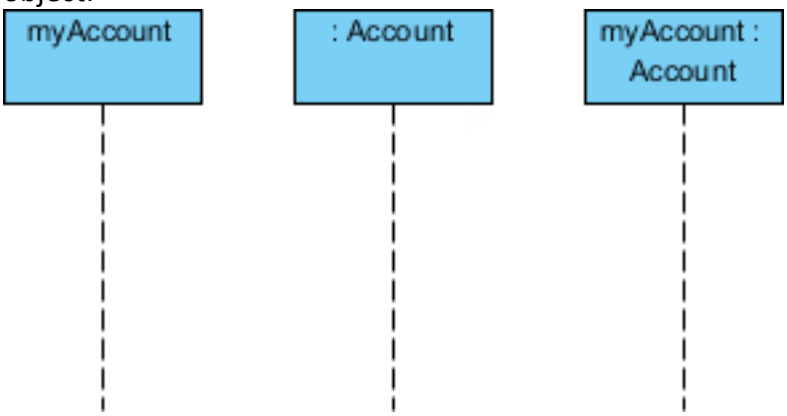
UML Sequence Diagrams Tutorial

Object

In the UML, an object in a sequence diagram is drawn as a rectangle containing the name of the object, underlined. An object can be named in one of three ways: the object name, the object name and its class, or just the class name (anonymous object). The three ways of naming an object are shown in Figure below.

Lifeline

Entities of participants in a collaboration (scenario) are written horizontally across the top of the diagram. A lifeline is represented by dashed vertical line drawn below each object. These indicate the existence of the object.



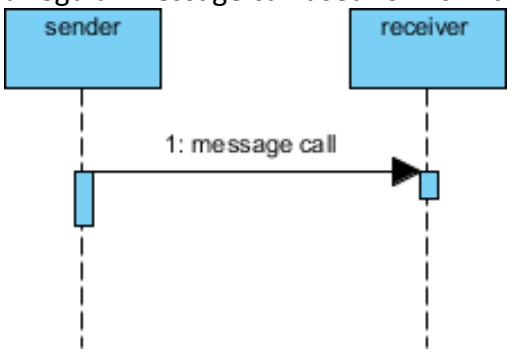
Object names can be specific (e.g., myAccount) or they can be general (e.g., myAccount :Account). Often, an anonymous object (:Account) may be used to represent any object in the class. Each object also has its timeline represented by a dashed line below the object. Messages between objects are represented by arrows that point from sender object to the receiver object.

Everything in an object-oriented system is accomplished by objects. Objects take on the responsibility for things like managing data, moving data around in the system, responding to inquiries, and protecting the system. Objects work together by communicating or interacting with one another.

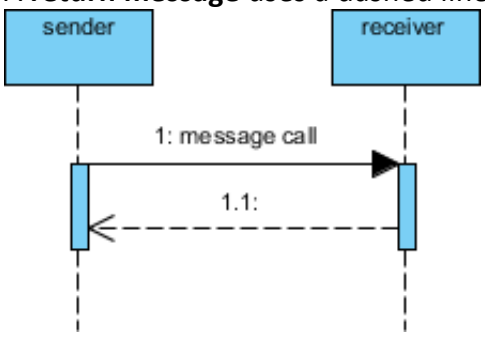
Message

Messages depict the invocation of operations are shown horizontally. They are drawn from the sender to the receiver. Ordering is indicated by vertical position, with the first message shown at the top of the diagram, and the last message shown at the bottom. As a result, sequence numbers is optional.
The line type and arrowhead type indicates the type of message being used:

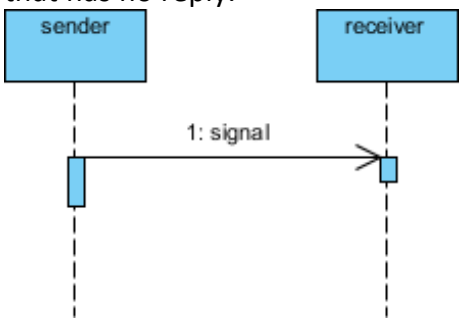
1. A **synchronous message** (typically an operation call) is shown as a solid line with a filled arrowhead. It is a regular message call used for normal communication between sender and receiver.



2. A **return message** uses a dashed line with an open arrowhead.



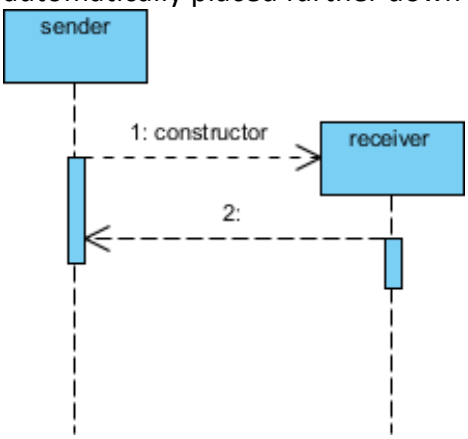
3. An **asynchronous message** has a solid line with an open arrowhead. A signal is an asynchronous message that has no reply.



Creation and Destruction Messages

Participants do not necessarily live for the entire duration of a sequence diagram's interaction. Participants can be created and destroyed according to the messages that are being passed.

A **constructor message** creates its receiver. The sender that already exist at the start of the interaction are placed at the top of the diagram. Targets that are created during the interaction by a constructor call are automatically placed further down the diagram.



A **destructor message** destroys its receiver. There are other ways to indicate that a target is destroyed during an interaction. Only when a target's destruction is set to 'after destructor' do you have to use a destructor.

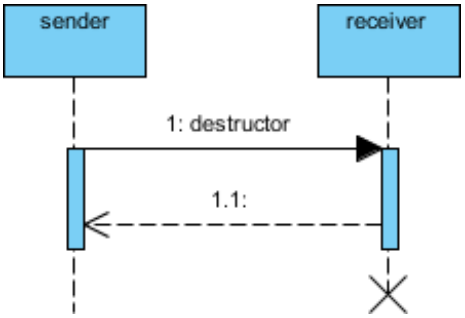


Figure 2. A sequence diagram that has incoming and outgoing messages

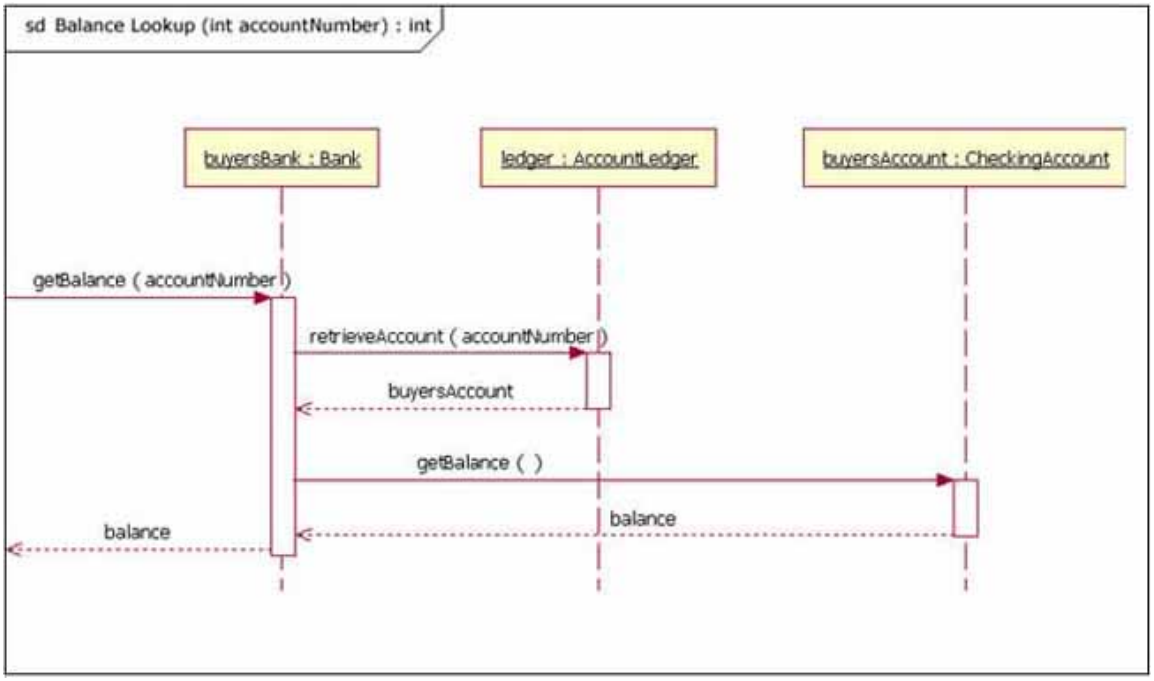


Figure 4. An example of messages being sent between objects

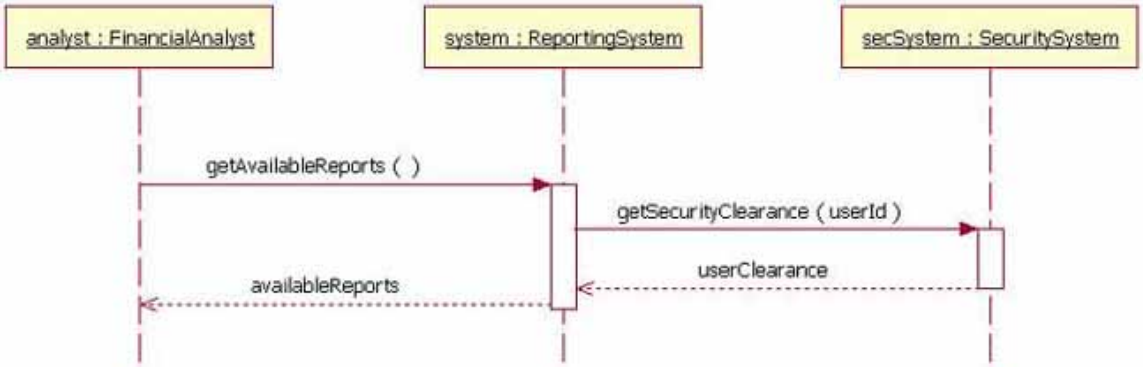


Figure 5. The system object calling its determineAvailableReports method

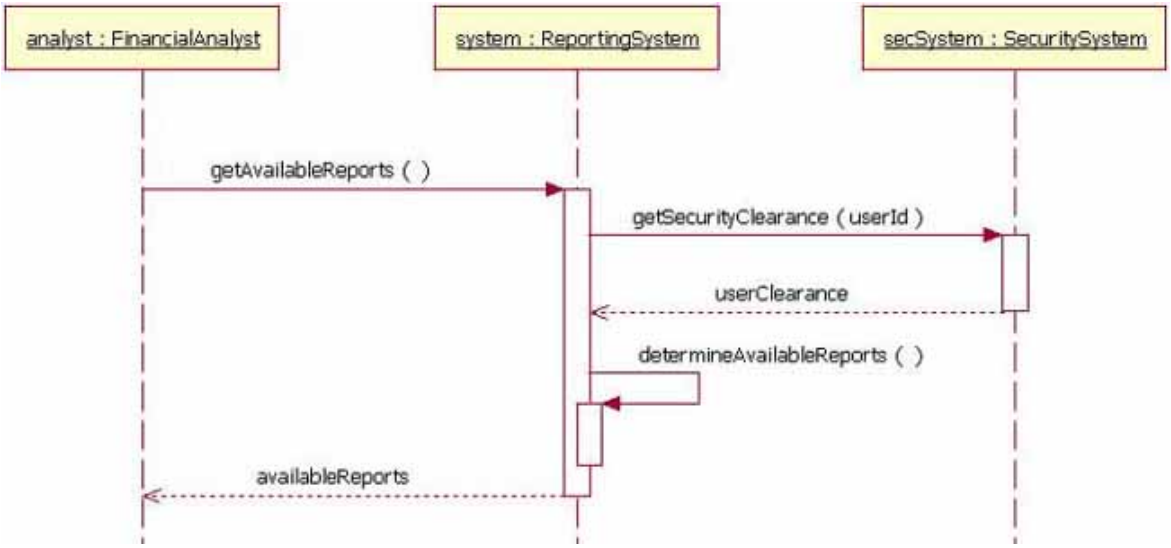


Figure 8. A sequence diagram fragment that contains an alternative combination fragment

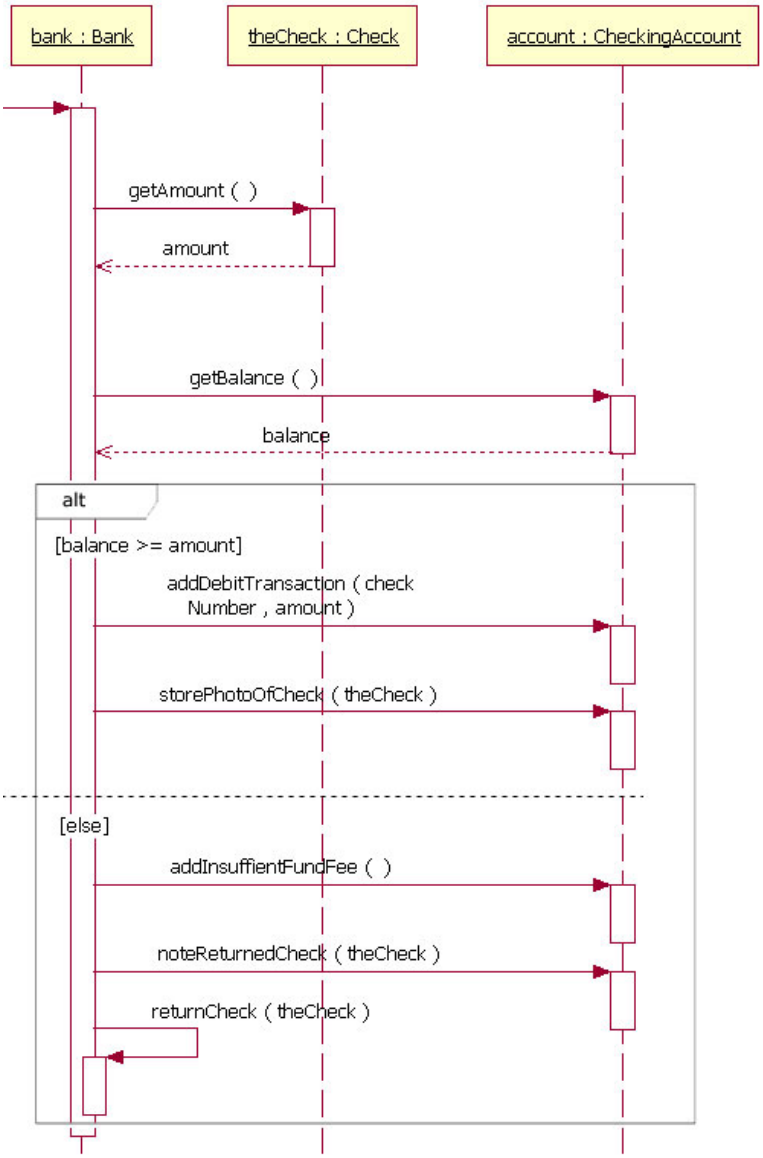
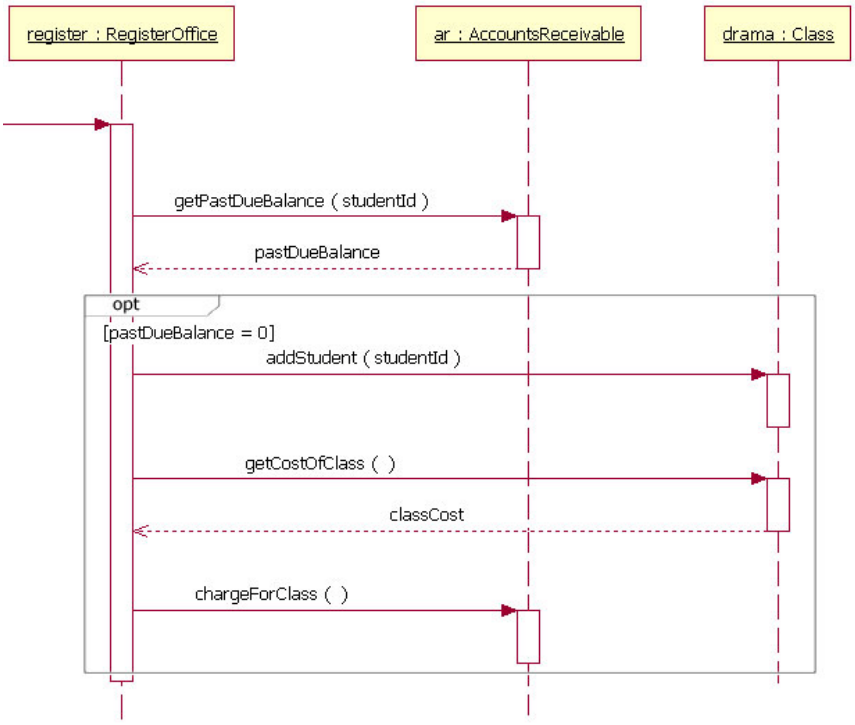
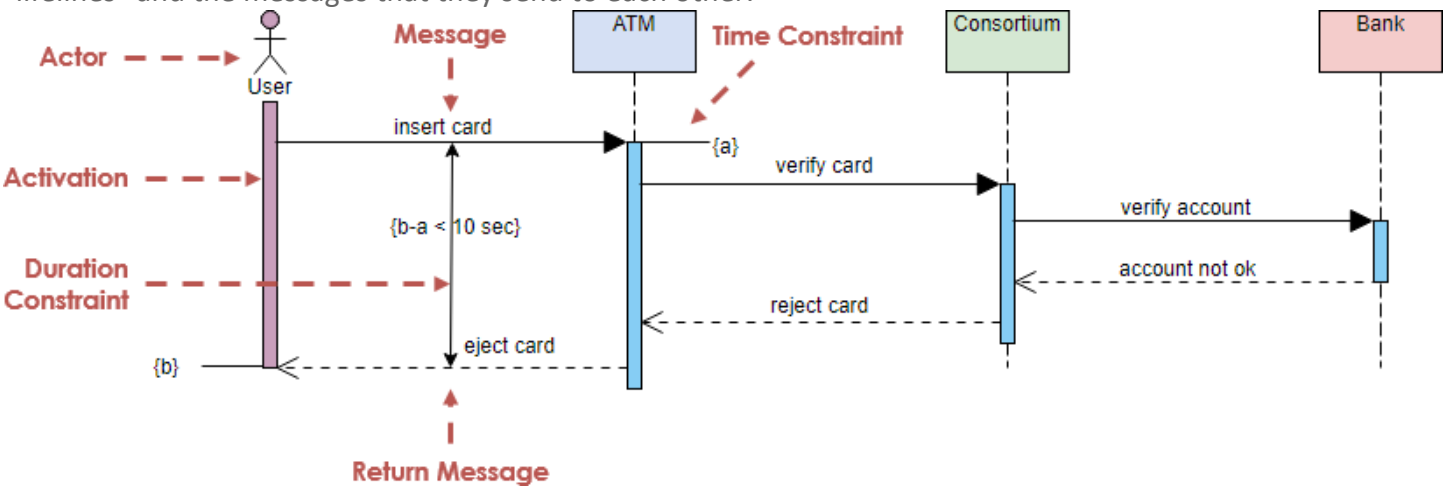


Figure 9. A sequence diagram fragment that includes an option combination fragment



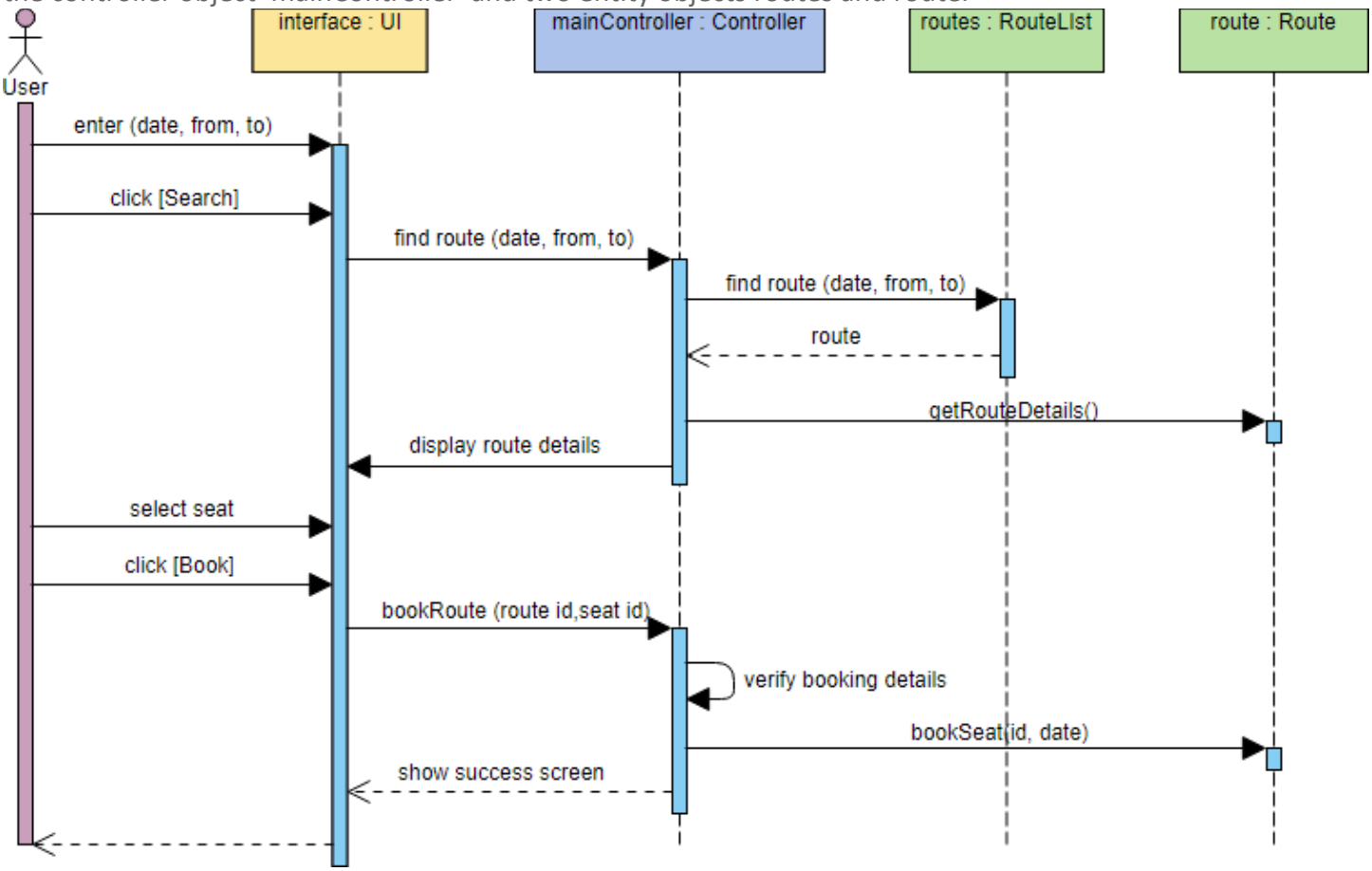
UML Sequence Diagram Tutorial

A **sequence diagram** describes an interaction among a set of objects participated in a collaboration (or scenario), arranged in a chronological order; it shows the objects participating in the interaction by their "lifelines" and the messages that they send to each other.

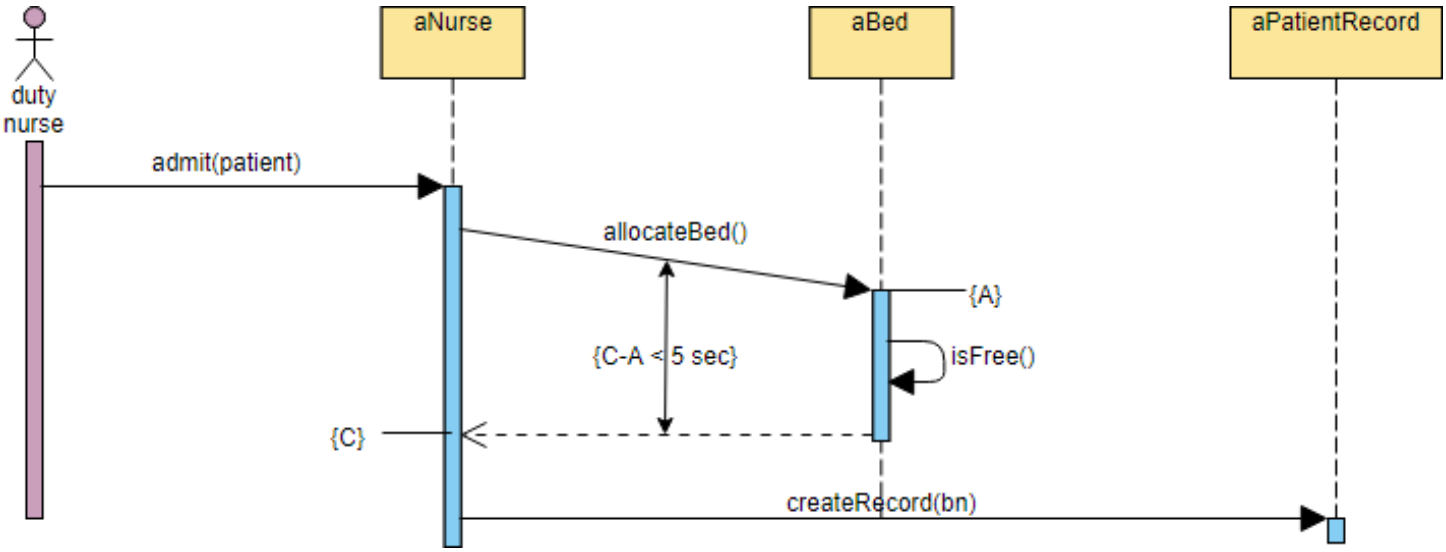


Sequence Diagram Examples

The sequence diagram example below shows the interactions between a user and a ticket booking system in booking a seat. It consists of mainly four parts: The actor, which is the user, the boundary object 'interface', the controller object 'mainController' and two entity objects routes and route.



The sequence diagram example below shows a patient admission process. It shows the use of timing and duration constraint.



The sequence diagram example shows how recursive message can be used in interaction modeling.