Sequence Diagram

Written Date : September 02, 2016

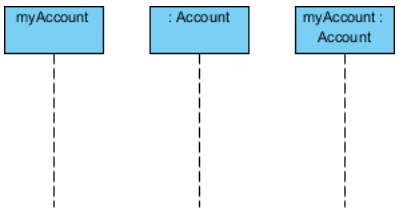
Both sequence diagrams and collaboration diagrams are kinds of interaction diagrams. Interaction diagrams address the dynamic view of a system. A sequence diagram is an interaction diagram that emphasizes the time-ordering of messages. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Typically, you'll use one sequence diagram to specify a use case's main flow, and variations of that diagram to specify a use case's exceptional flows.

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.

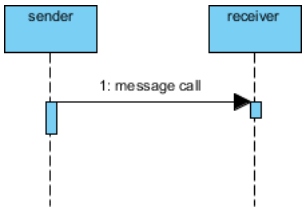
Everything in an object-oriented system is accomplished by objects. Objects take on the responsibility for things like managing data, moving data around in the system, responding to inquiries, and protecting the system. Objects work together by communicating or interacting with one another.

Message

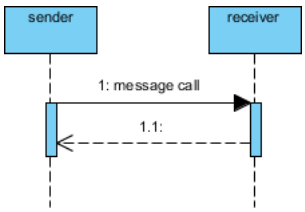
Messages depict the invocation of operations are shown horizontally. They are drawn from the sender to the receiver. Ordering is indicated by vertical position, with the first message shown at the top of the diagram, and the last message shown at the bottom. As a result, sequence numbers is optional.

The line type and arrowhead type indicates the type of message being used:

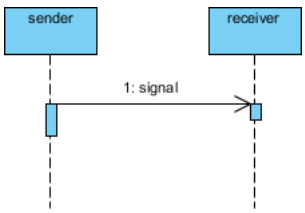
1. A **synchronous message** (typically an operation call) is shown as a solid line with a filled arrowhead. It is a regular message call used for normal communication between sender and receiver.



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



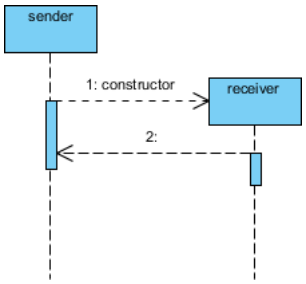
3. An **asynchronous message** has a solid line with an open arrowhead. A signal is an asynchronous message that has no reply.



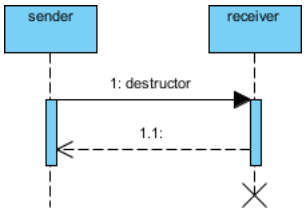
Creation and Destruction Messages

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.

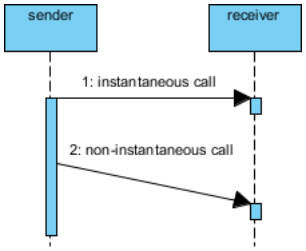


A **destructor message** destroys its receiver. There are other ways to indicate that a target is destroyed during an interaction. Only when a target's destruction is set to 'after destructor' do you have to use a destructor.



Non instantaneous message

Messages are often considered to be instantaneous, thus, the time it takes to arrive at the receiver is negligible. The messages are drawn as a horizontal arrow. To indicate that it takes a certain while before the receiver actually receives a message, a **slanted arrow is used**.



Focus of Control

Focus of Control represents the period during which an element is performing an operation. The top and the bottom of the of the rectangle are aligned with the initiation and the completion time respectively

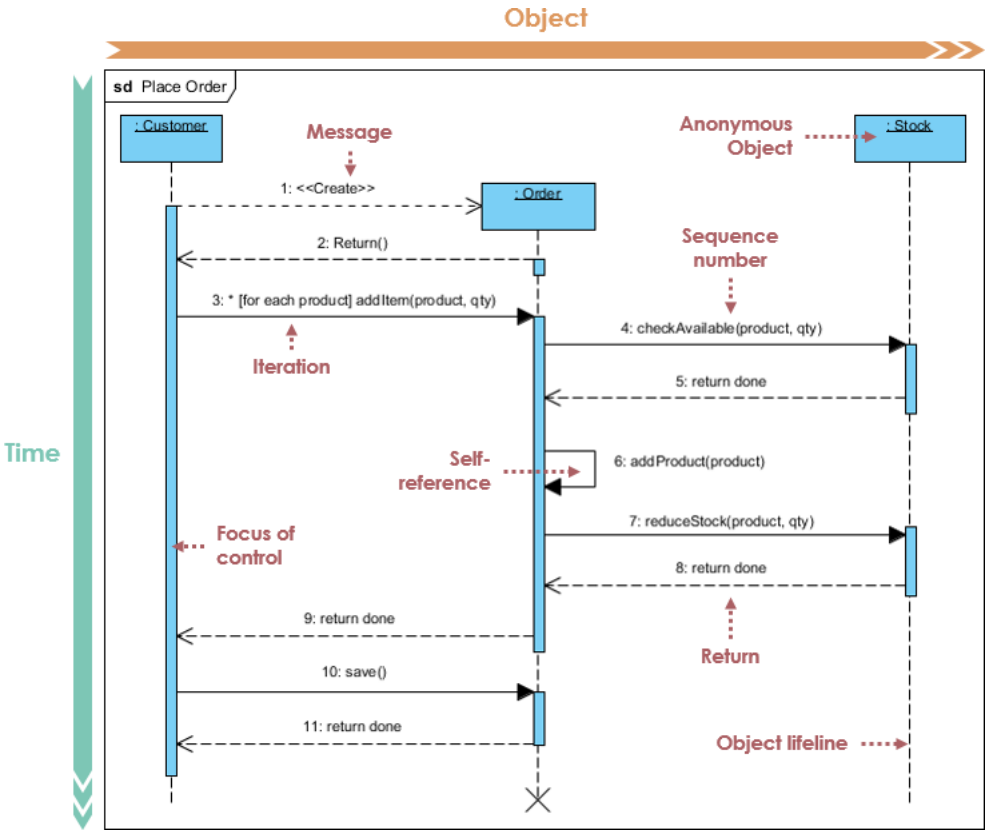
Iteration notation

Iteration notation represents a message is sent many times to multiple receiver objects, as would happen when you are iterating over a collection. You can show the basis of the iteration within brackets, such as *[for all order lines].

Example: Place Order

The example shows a Sequence diagram with three participating objects: Customer, Order, and the Stock. Without even knowing the notation formally, you can probably get a pretty good idea of what is going on.

1. Step 1 and 2: Customer creates an order.
2. Step 3: Customer add items to the order.
3. Step 4, 5: Each item is checked for availability in inventory.
4. Step 6, 7, 8 : If the product is available, it is added to the order.
5. Step 9 return
6. Step 10, 11: save and destroy order



Sequence Fragments

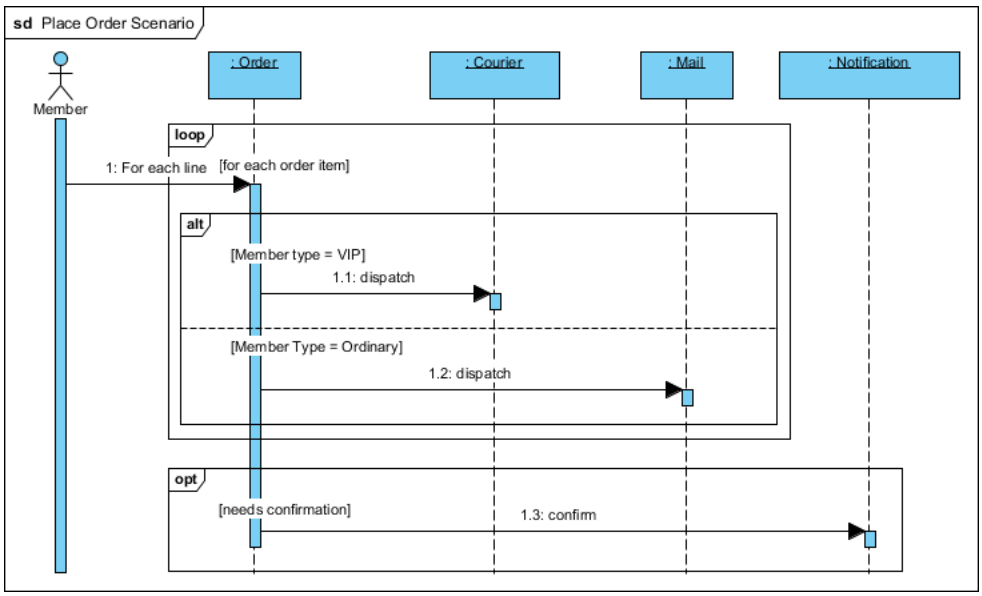
In a UML sequence diagram, combined fragments let you show loops, branches, and other alternatives. A combined fragment consists of one or more interaction operands, and each of these encloses one or more messages, interaction uses, or combined fragments.

A sequence fragment is represented as a box called a combined fragment, which encloses a portion of the interactions within a sequence diagram. The fragment operator (in the top left cornet) indicates the type of fragment. Fragment types include ref, assert, loop, break, alt, opt and neg, ref, sd.

Operator	Meaning
alt	Alternative multiple fragments: only the one whose condition is true will execute.
opt	**Optional**: the fragment executes only if the supplied condition is true. Equivalent to an alt only with one trace.
par	**Parallel**: each fragment is run in parallel.
loop	**Loop**: the fragment may execute multiple times, and the guard indicates the basis of iteration.
critical	**Critical region**: the fragment can have only one thread executing it at once.
neg	**Negative**: the fragment shows an invalid interaction.
ref	**Reference**: refers to an interaction defined on another diagram. The frame is drawn to cover the lifelines involved in the interaction. You can define parameters and a return value.
sd	**Sequence diagram**: used to surround an entire sequence diagram.

Example - Place Order Scenario

A member of a ship who would like to place an order online. The item ordered will be sent to the member either send by courier or by ordinary mail depending on she member status (VIP, Ordinary membership). Optionally, the shop will send the member a confirmation notification if the member opted for the notification option in the order.



Visual Paradigm supports sequence diagram and other UML diagram types. You can find all the tools you need in modeling the dynamic behaviors of a system using sequence diagram.

Try now for FREE

References

1. How to draw a Sequence Diagram in UML - Visual Paradigm User's Guide

Visual Paradigm home page
(https://www.visual-paradigm.com/)

Visual Paradigm handbooks
(https://www.visual-paradigm.com/learning/handbooks/)