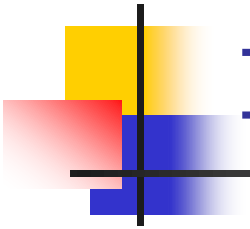
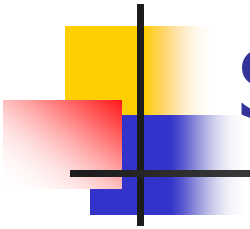
8. Dynamic Modeling using the Unified Modeling Language (UML) - **Sequence diagram**



Interaction Diagrams

- show how objects interact with one another
- UML supports two types of interaction diagrams
  - Sequence diagrams
  - Collaboration diagrams

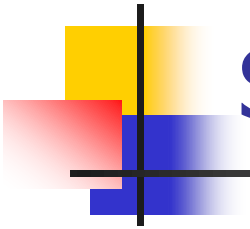


Sequence Diagram

A ***sequence diagram*** is an interaction diagram that emphasizes the time ordering of messages. It shows a set of objects and the messages sent and received by those objects.

Graphically, a sequence diagram is a table that shows objects arranged along the X axis and messages, ordered in increasing time, along the Y axis.

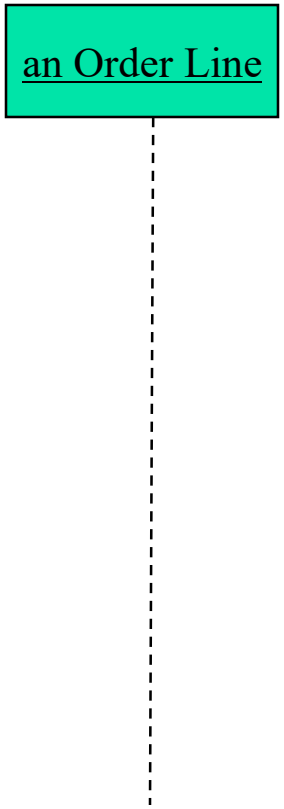
- *The UML User Guide, [Booch,99]*



Sequence Diagram

- A sequence diagram 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

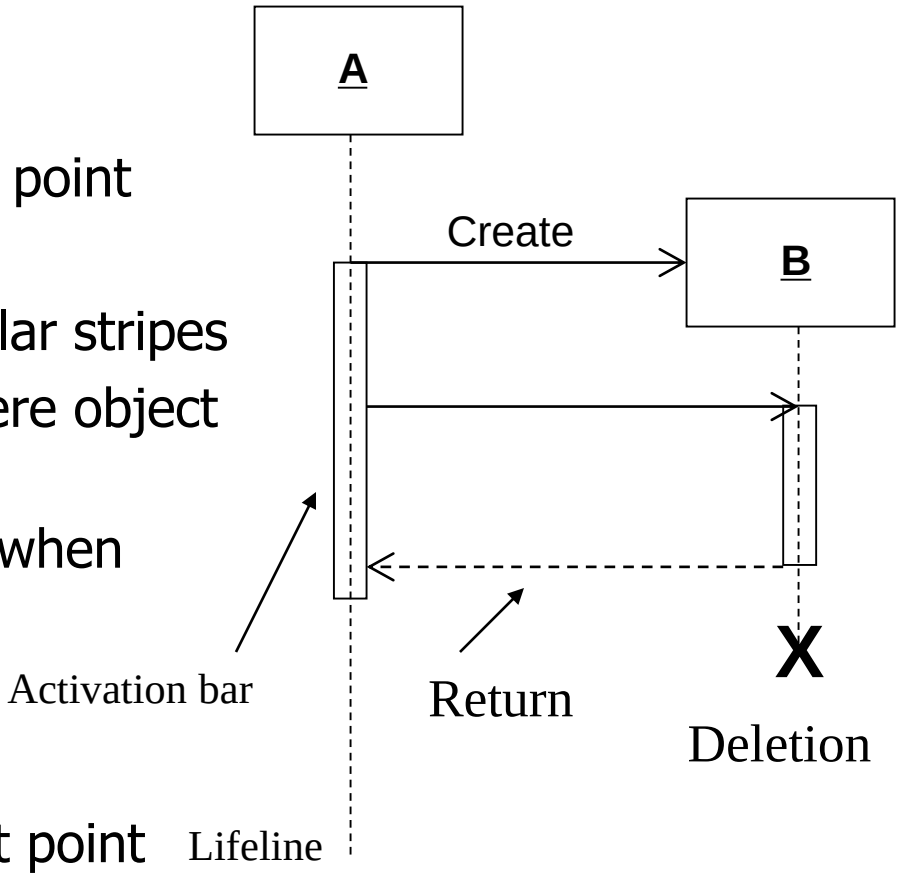
Sequence Diagram – Object Lifeline



An object in a sequence diagram is rendered as a box with a dashed line descending from it. The line is called the *object lifeline*, and it represents the existence of an object over a period of time.

Sequence Diagrams – Object Life Spans

- Creation
  - Create message
  - Object life starts at that point
- Activation
  - Symbolized by rectangular stripes
  - Place on the lifeline where object is activated.
  - Rectangle also denotes when object is deactivated.
- Deletion
  - Placing an 'X' on lifeline
  - Object's life ends at that point

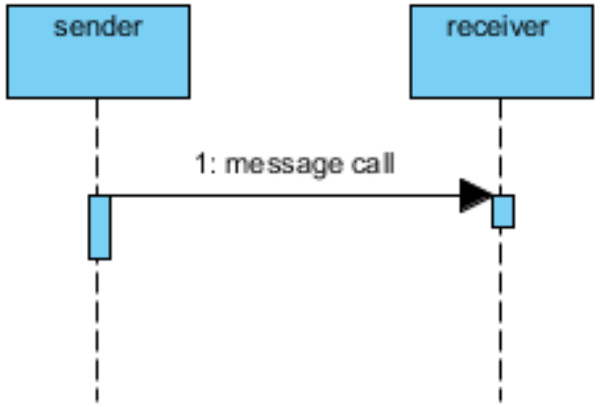


Sequence Diagram - Message

- 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.

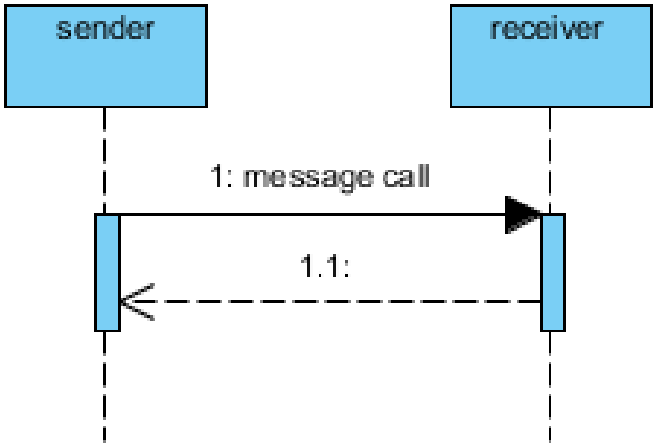
Sequence Diagram - Message

- 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.



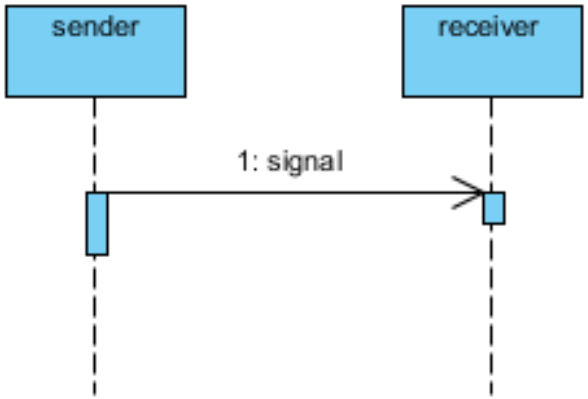
Sequence Diagram - Message

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



Sequence Diagram - Message

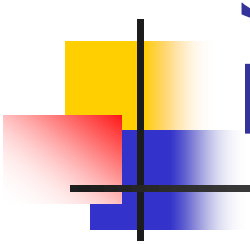
- An **asynchronous message** has a solid line with an open arrowhead. A signal is an asynchronous message that has no reply.



Sequence Diagram - Creation and Destruction Messages

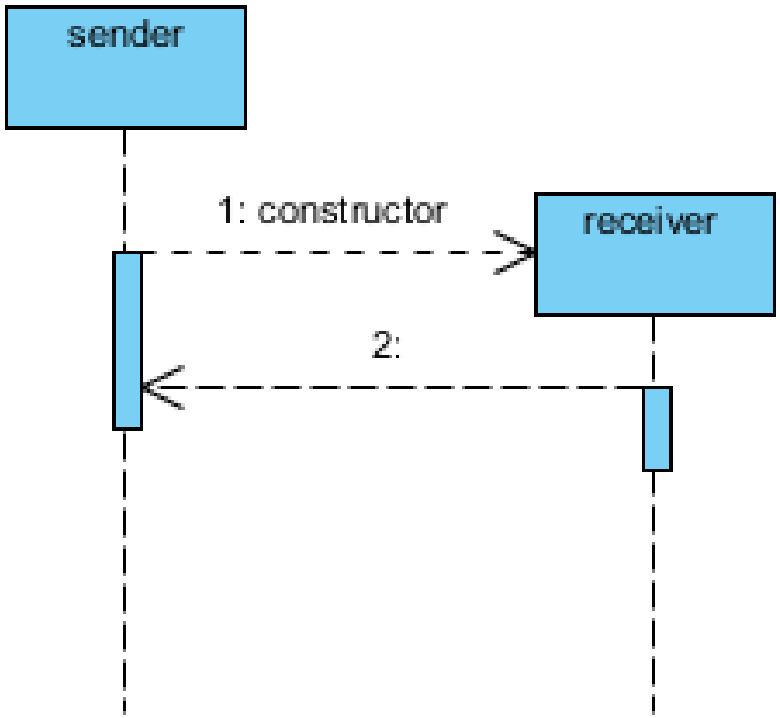
- Participants do not necessarily live for the entire duration of a sequence diagram's interaction. Participants can be created and destroyed according to the messages that are being passed.

Sequence Diagram - Creation and Destruction Messages

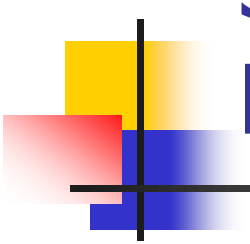


- A **constructor** message creates its receiver. The sender that already exist at the start of the interaction are placed at the top of the diagram. Targets that are created during the interaction by a constructor call are automatically placed further down the diagram.

Sequence Diagram - Creation and Destruction Messages

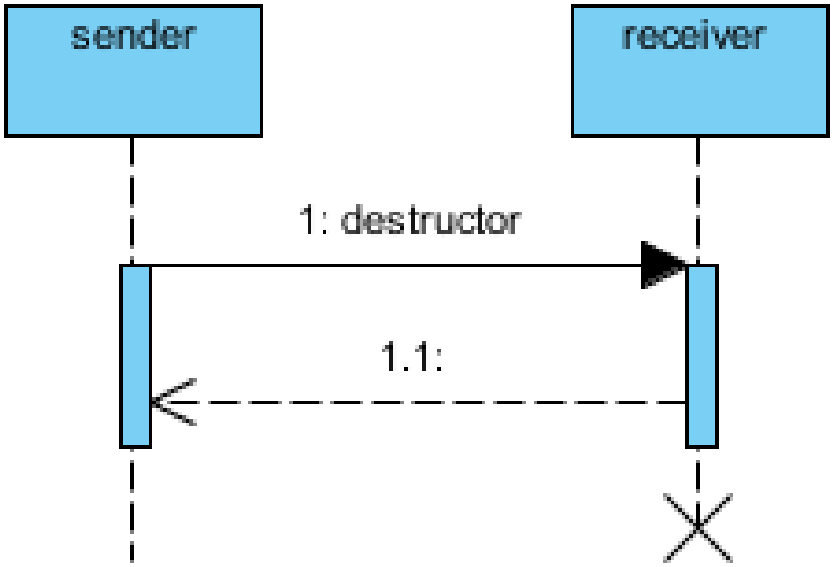


Sequence Diagram - Creation and Destruction Messages



- 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.

Sequence Diagram - Creation and Destruction Messages



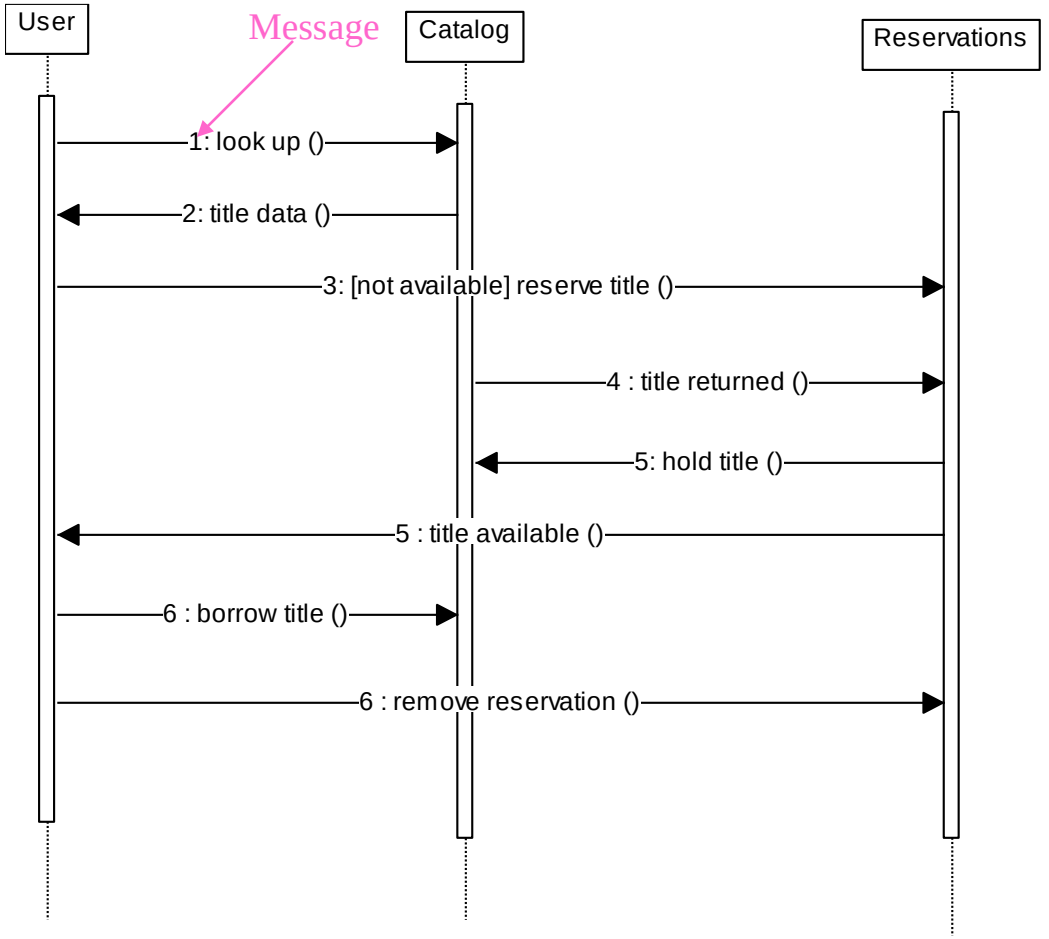
Sequence Diagram

•Sequence diagrams demonstrate the behavior of objects in a use case by describing the objects and the messages they pass.

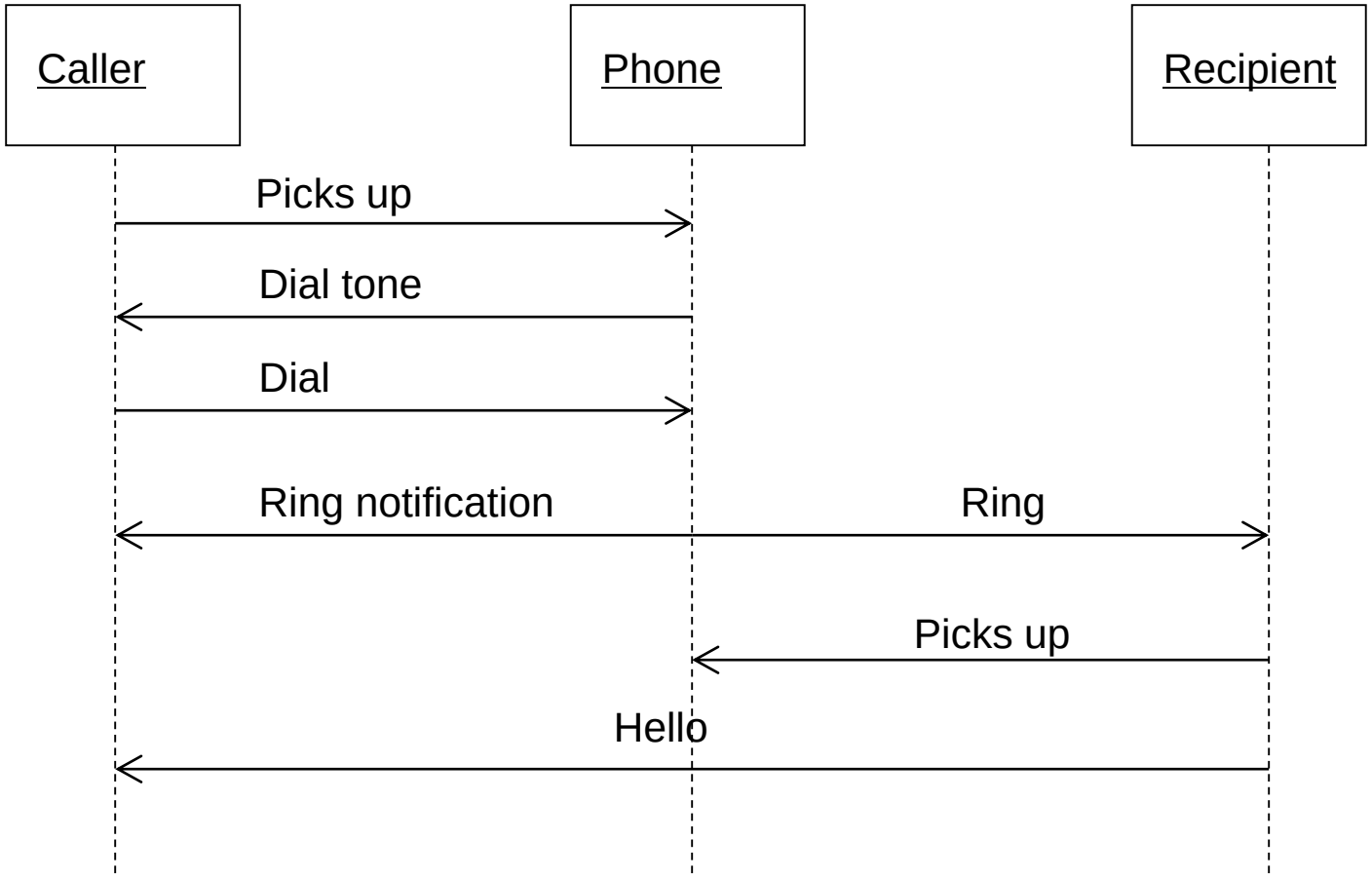
•The horizontal dimension shows the objects participating in the interaction.

•The vertical arrangement of messages indicates their order.

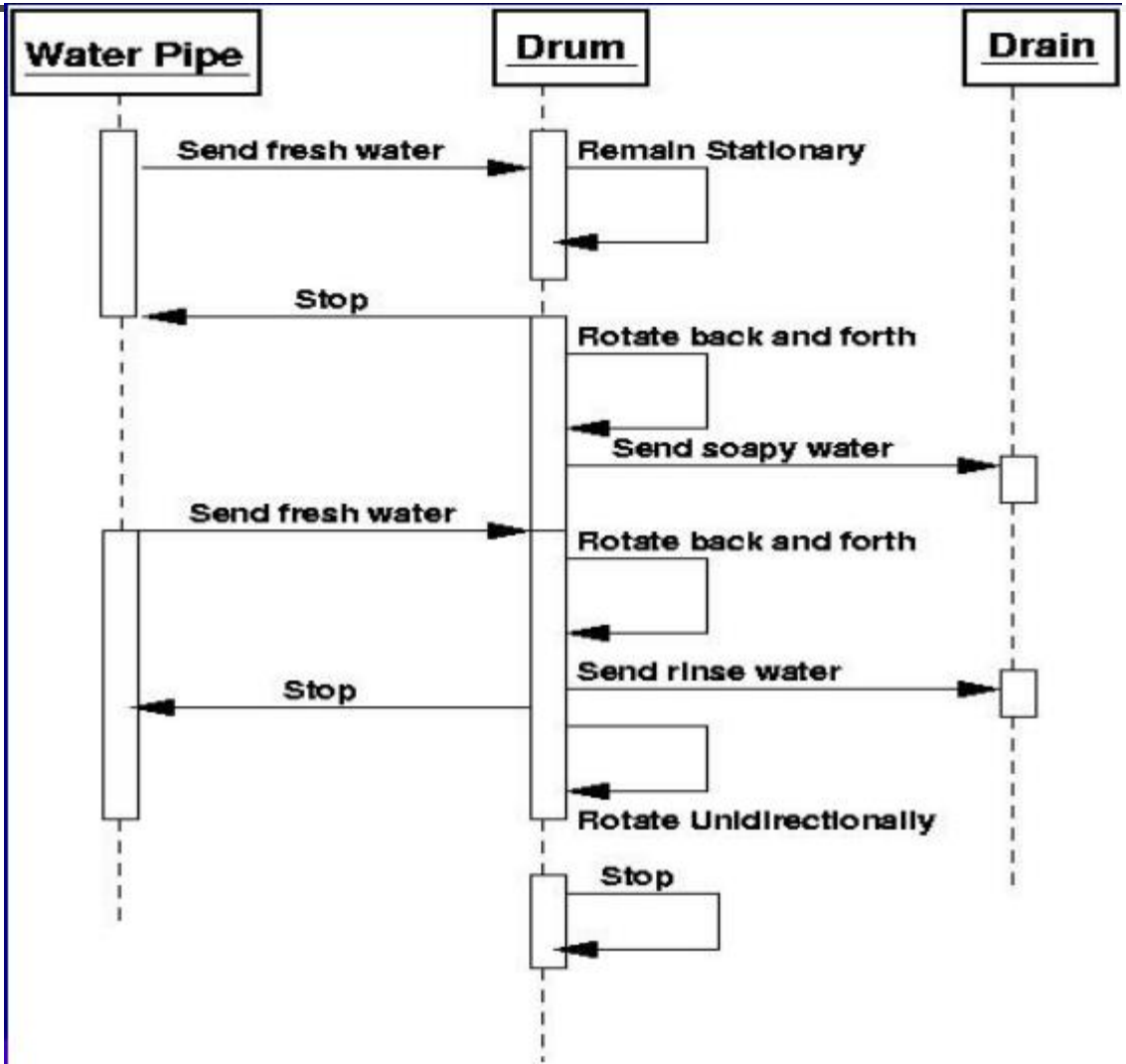
•The labels may contain the seq. # to indicate concurrency.



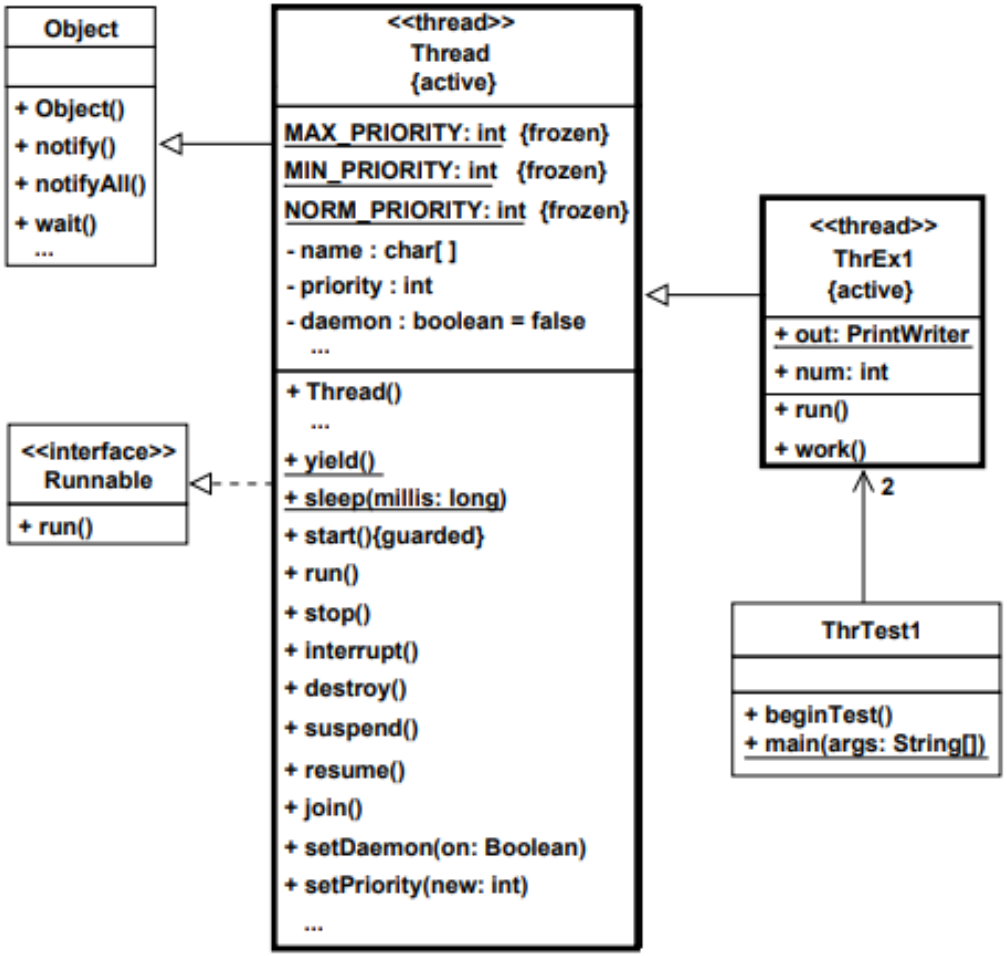
Sequence Diagram(make a phone call)



Sequence Diagram



Sequence Diagram – Example #1



Sequence Diagram – Example #1