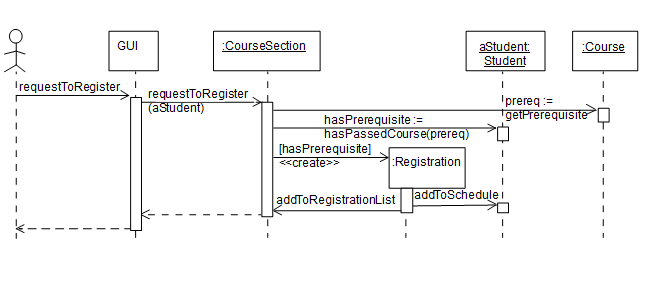
Interaction Diagrams

**-Model the dynamic aspects of a software system - visualize how the system runs - after use case & class diagrams**

**Sequence Diagram:**

Object Method from target object being called



Actor

(External entity)

Instance created

X

Lifeline

Activation Box (shown when active) End of object’s life

**[Condition]** **\*[iterations]** **sequence\_number:** **return\_variable:=** method\_name (**args**)

**List of arguments**

**Name of the variable returned by the operation**

**(int) orders which method call is done first (top to bottom)**

**(int) how many times this method will be called**

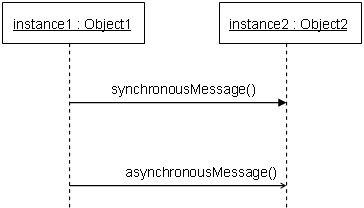
**Condition that needs to be satisfied**

**else skip**

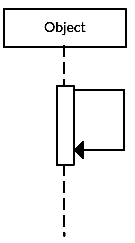
Note:

All of these are

optional



Synchronous: sender must wait for the response



of the receiver (frozen state)

Reflexive message: When an object sends a message to itself

(Calls one of its own methods)

Note: A return message is used to indicate that the message receiver is done processing the message and is returning control over to the message caller. Return messages are optional notation pieces.



Tip: You can avoid cluttering up your diagrams by minimizing the use of return messages since the return value can be specified in the initial message arrow itself.

**Fragments:**

|  |  |
| --- | --- |
| **Operator** | **Meaning**  **opt**  [x == 5] |
| alt | **Alternate**: only one fragment will execute |
| opt | **Optional**: the fragment executes only  if the supplied condition is true |
| loop | **Loop**: the fragment may execute multiple times |
| ref | **Reference**: refers to an interaction defined  on another diagram |

**alt**

**alt**

**[x>y]**

**[x<y]**

**Why we use them:**

A good sequence diagram is still above the level of the real code (not all code is drawn on diagram)

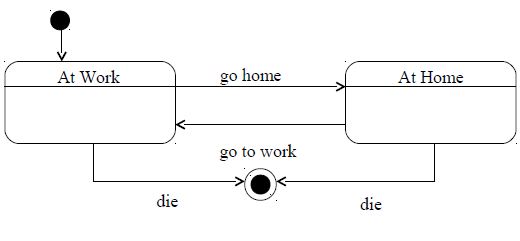
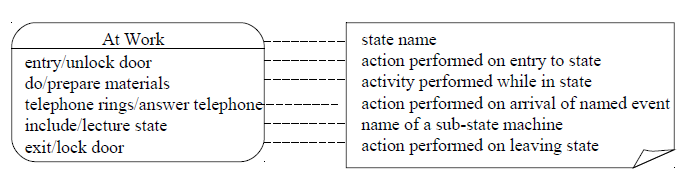
Sequence diagrams are language-agnostic (can be implemented in many different languages)

Non-coders can read and write sequence diagrams.

It is easier to do sequence diagrams as a team.

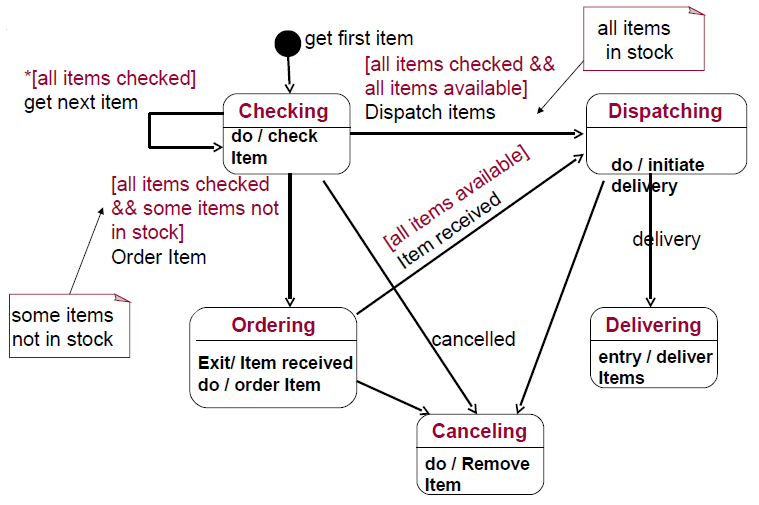
Can see many objects/classes at a time on same page (visual bandwidth)

**State Diagram**



Initial state

end state



comment

Super-States:

