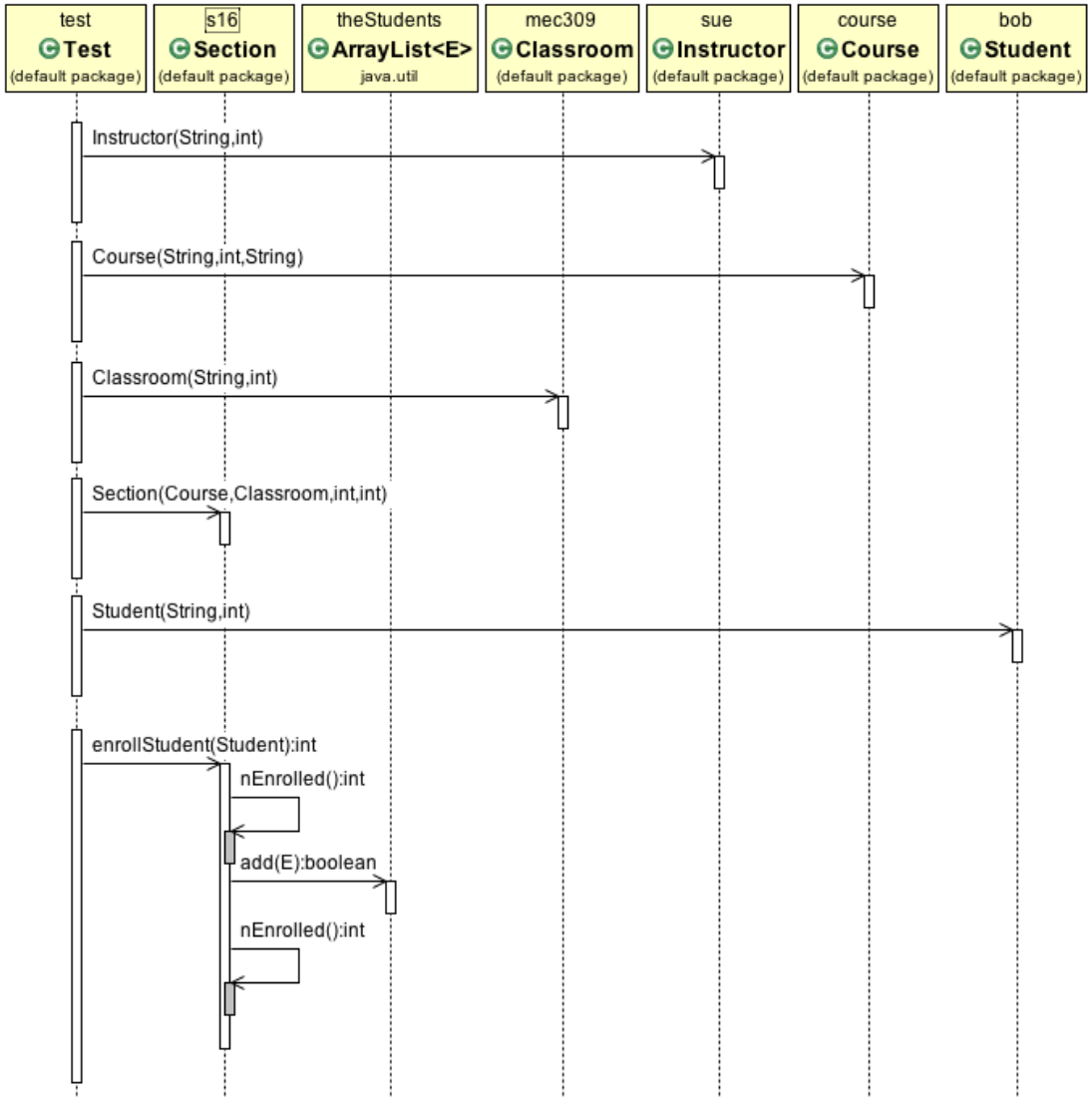
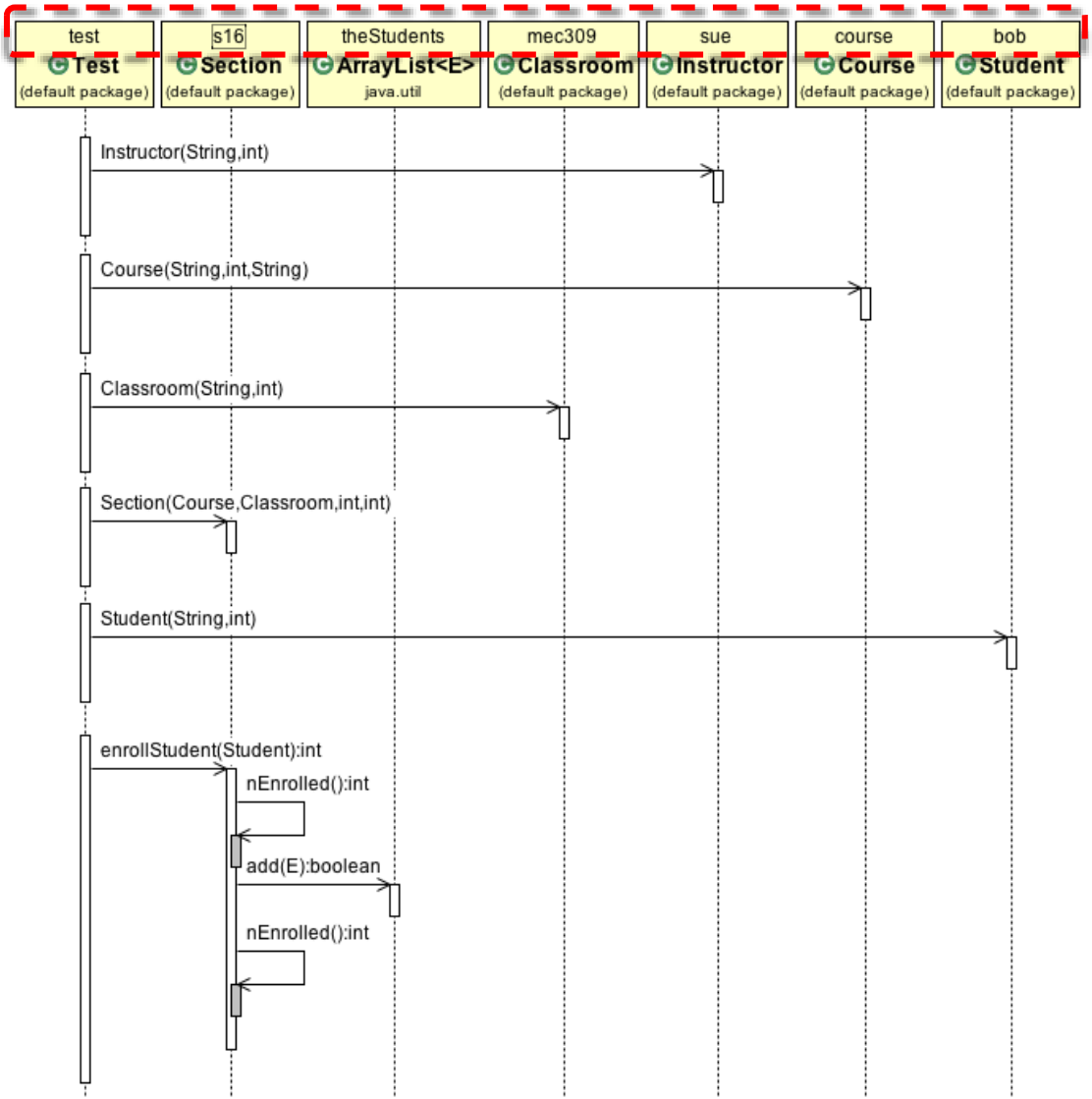
UML Sequence Diagrams

Sequence Diagram Example



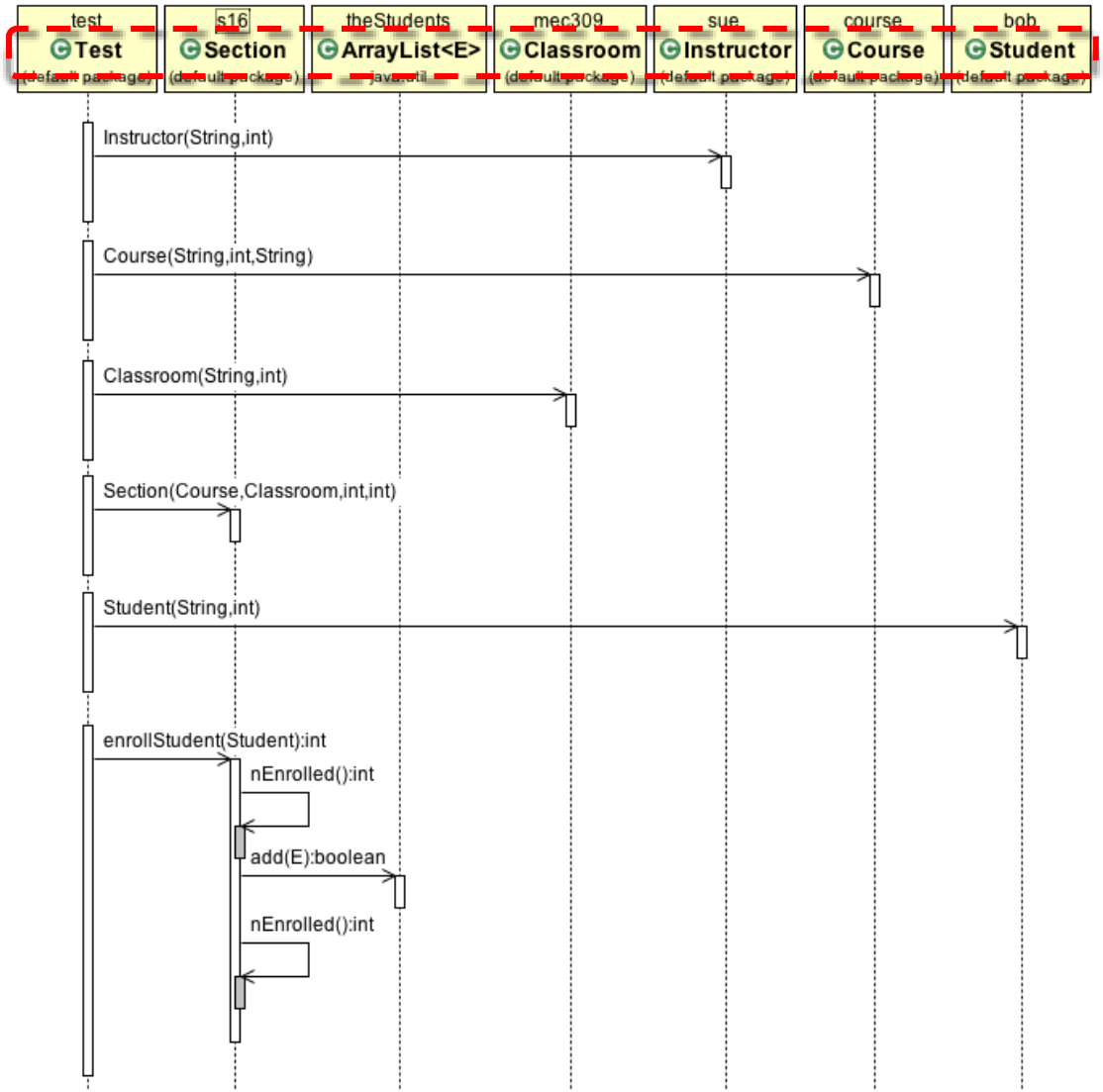
Thoughts?

Sequence Diagram Example



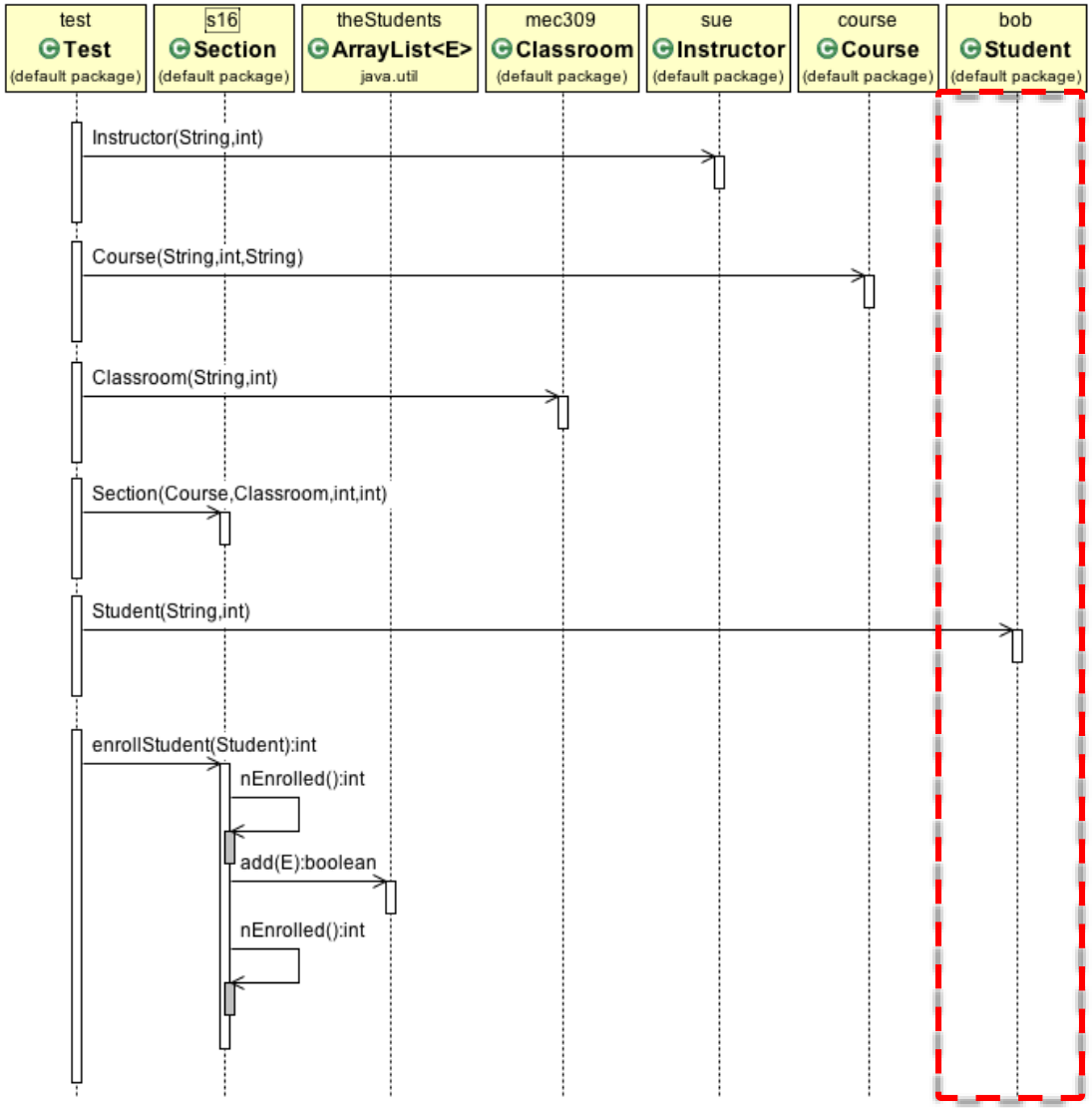
Objects / instances

Sequence Diagram Example



Class names

Sequence Diagram Example



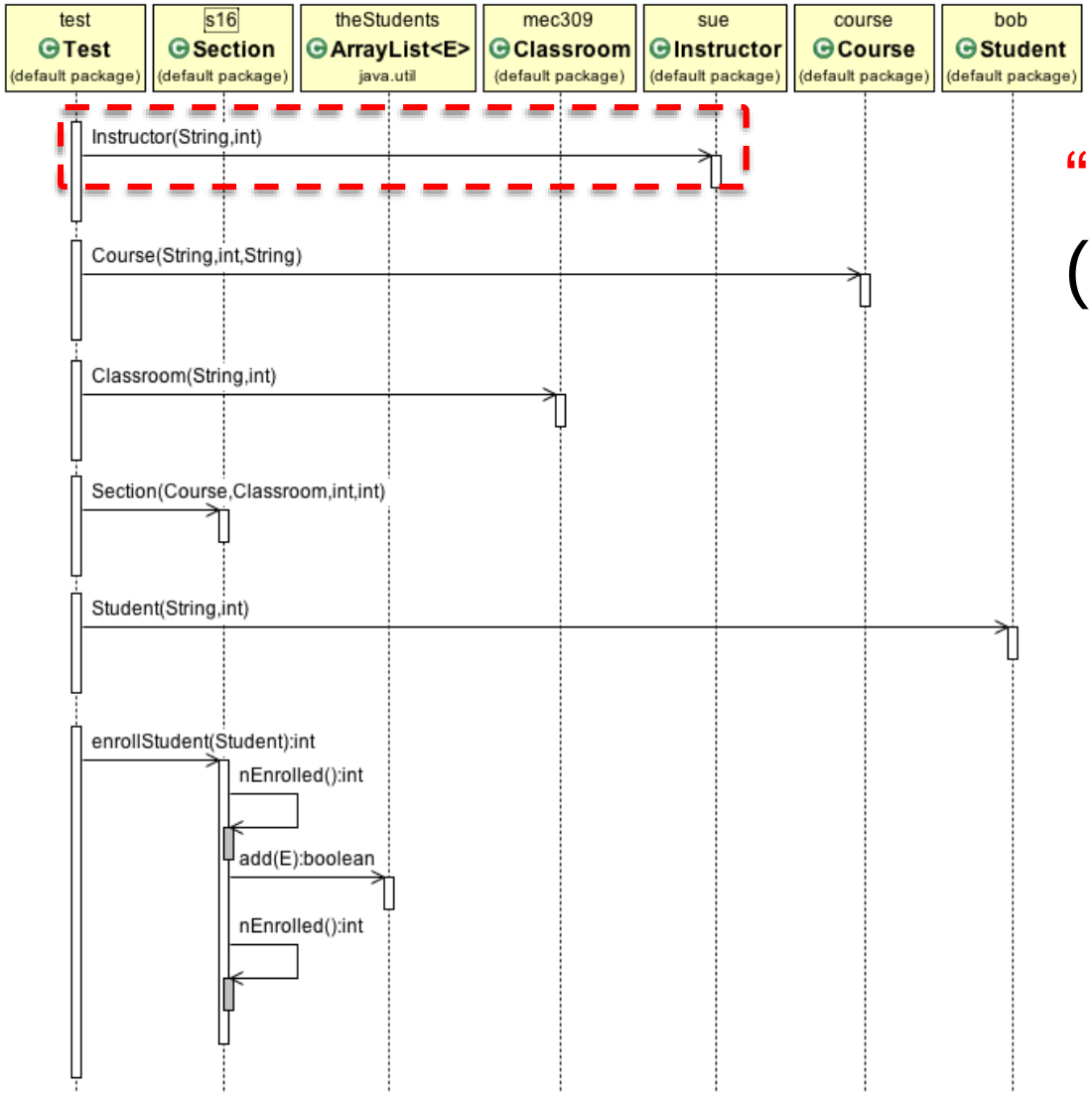
"Time/lifeline"

- from "early" (top)
- to "late" (bottom)

Sequence Diagram Example

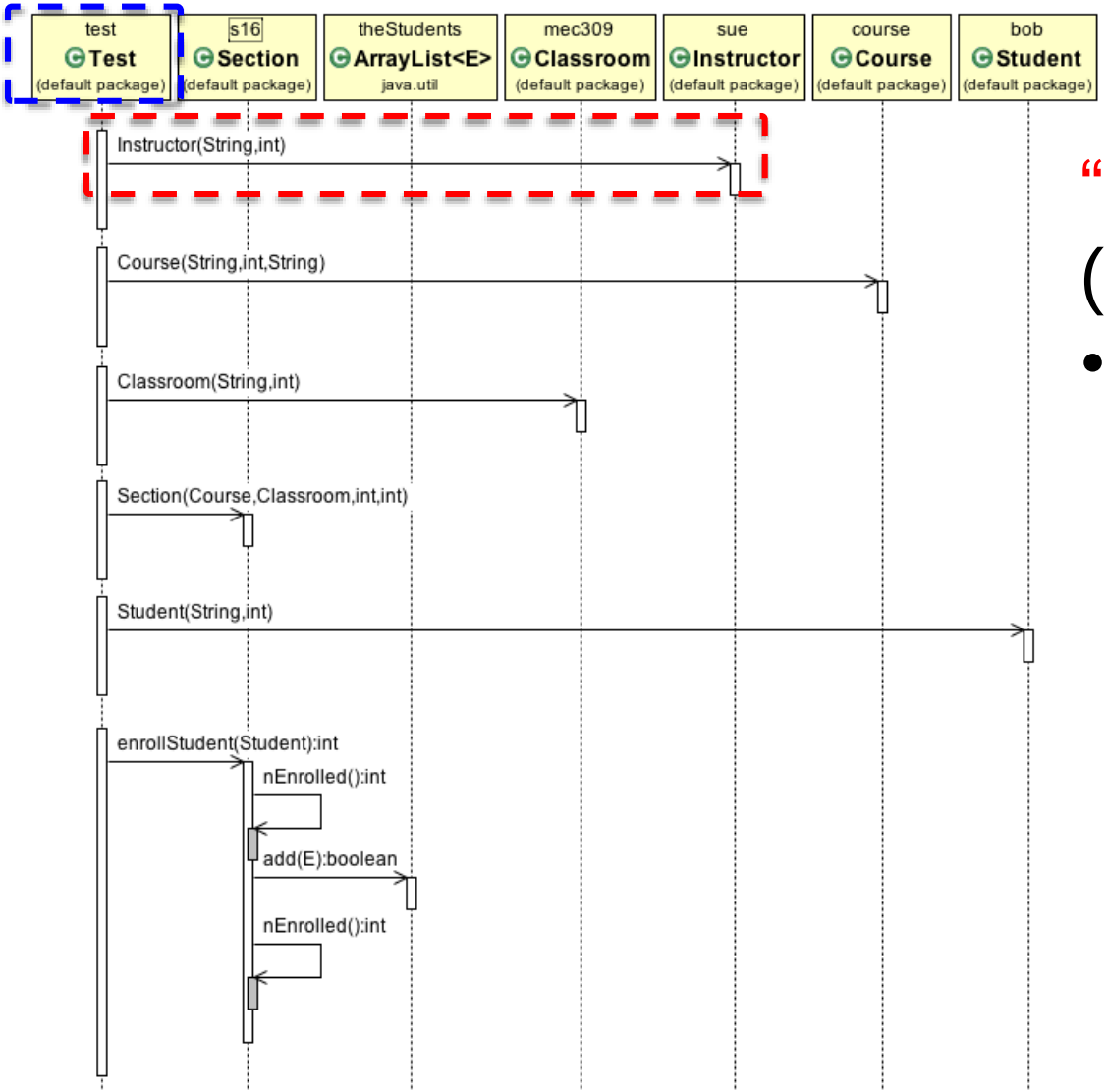
"Execution/Processing Time" of method for object (aka Activation boxes or method-call boxes)

Sequence Diagram Example



"Message Passing"
(i.e., method call)

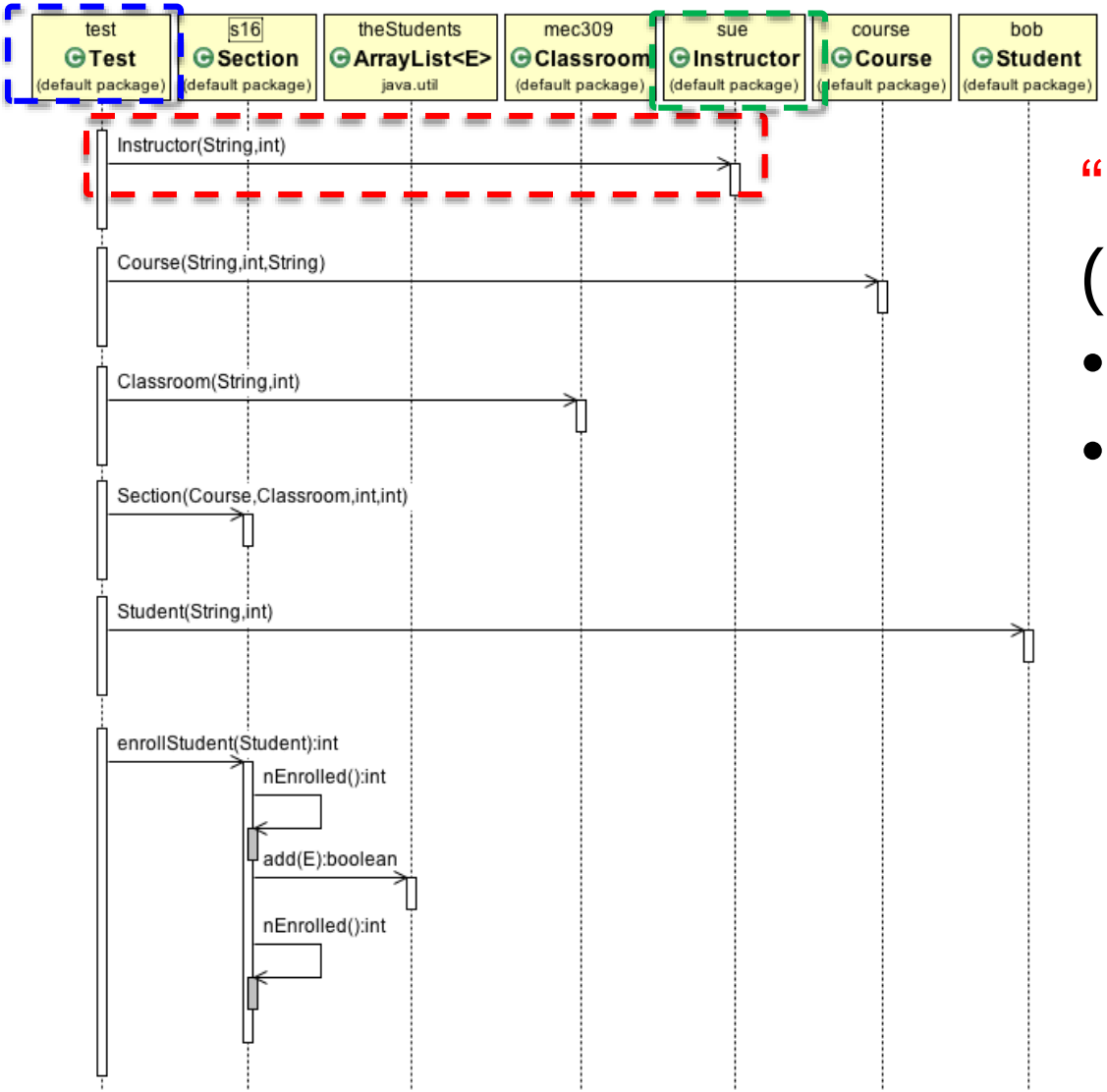
Sequence Diagram Example



"Message Passing"
(i.e., method call)

- from caller

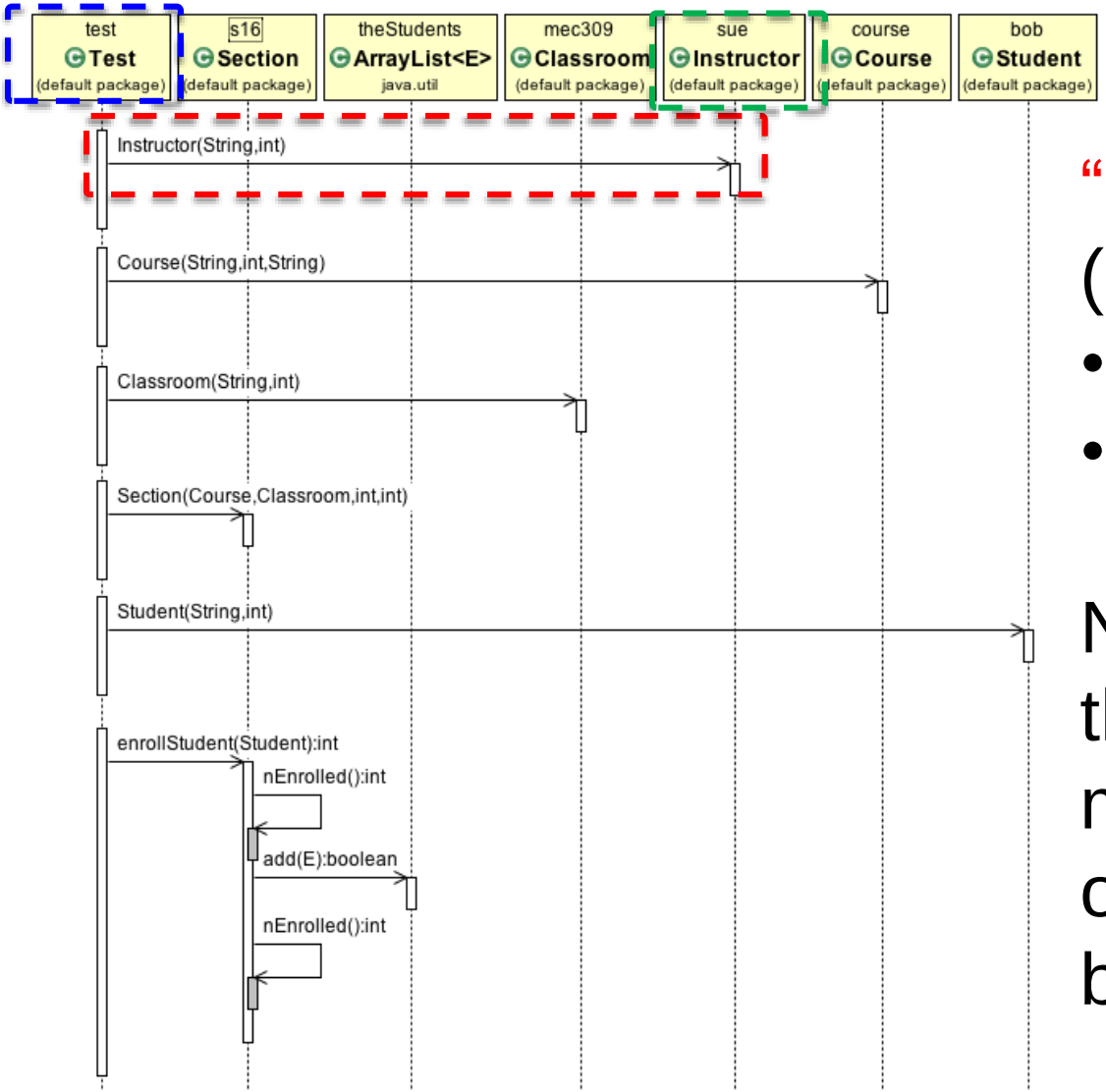
Sequence Diagram Example



"Message Passing"
(i.e., method call)

- from caller
- to callee

Sequence Diagram Example



"Message Passing"
(i.e., method call)

- from caller
- to callee

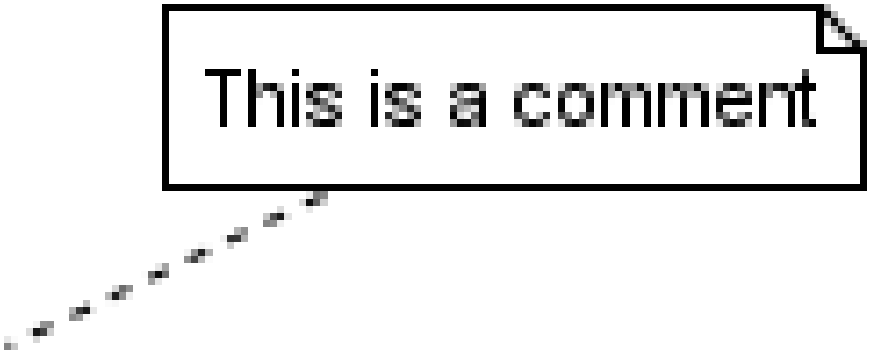
NOTE:
the name of the method that appears on the arrow label belongs to the callee

Sequence Diagrams

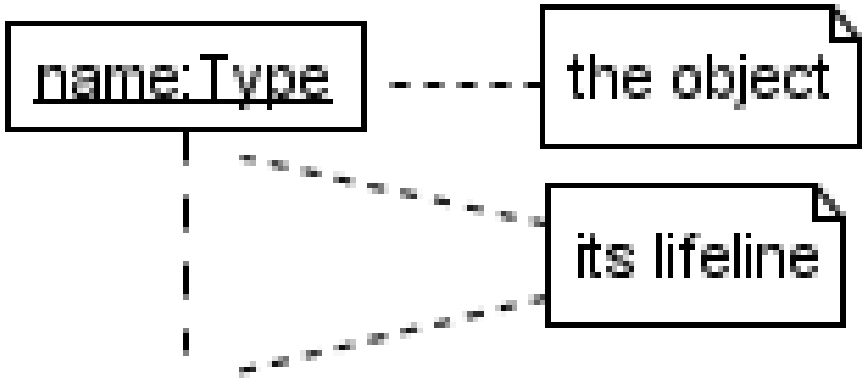
- Illustrates interactions between objects
- Illustrates dynamic behavior
  - When new objects are instantiated
  - Method invocations
  - Order of operations
- Can also illustrate when objects are destroyed

Sequence Diagrams Notations*

- Comments



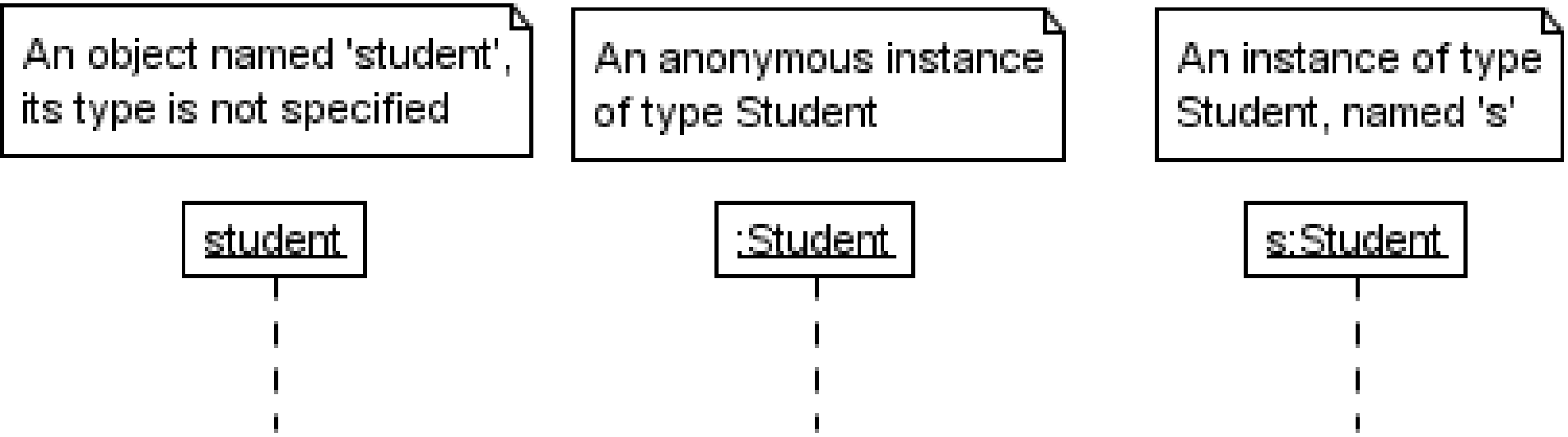
- Basic Notation



*http://www.tracemodeler.com/articles/a_quick_introduction_to_uml_sequence_diagrams/

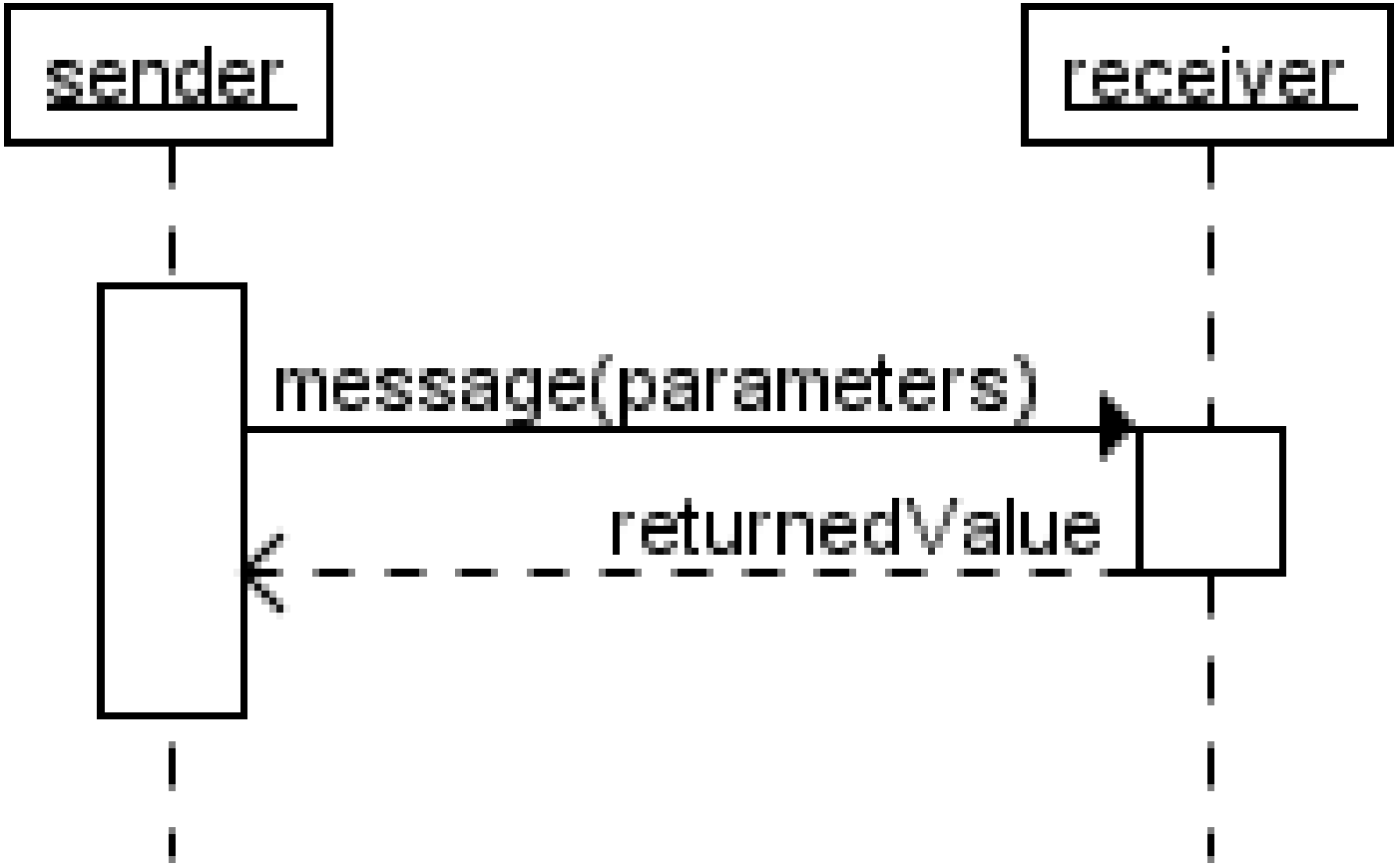
Sequence Diagrams Notations

- Object notations following the "object:class" template



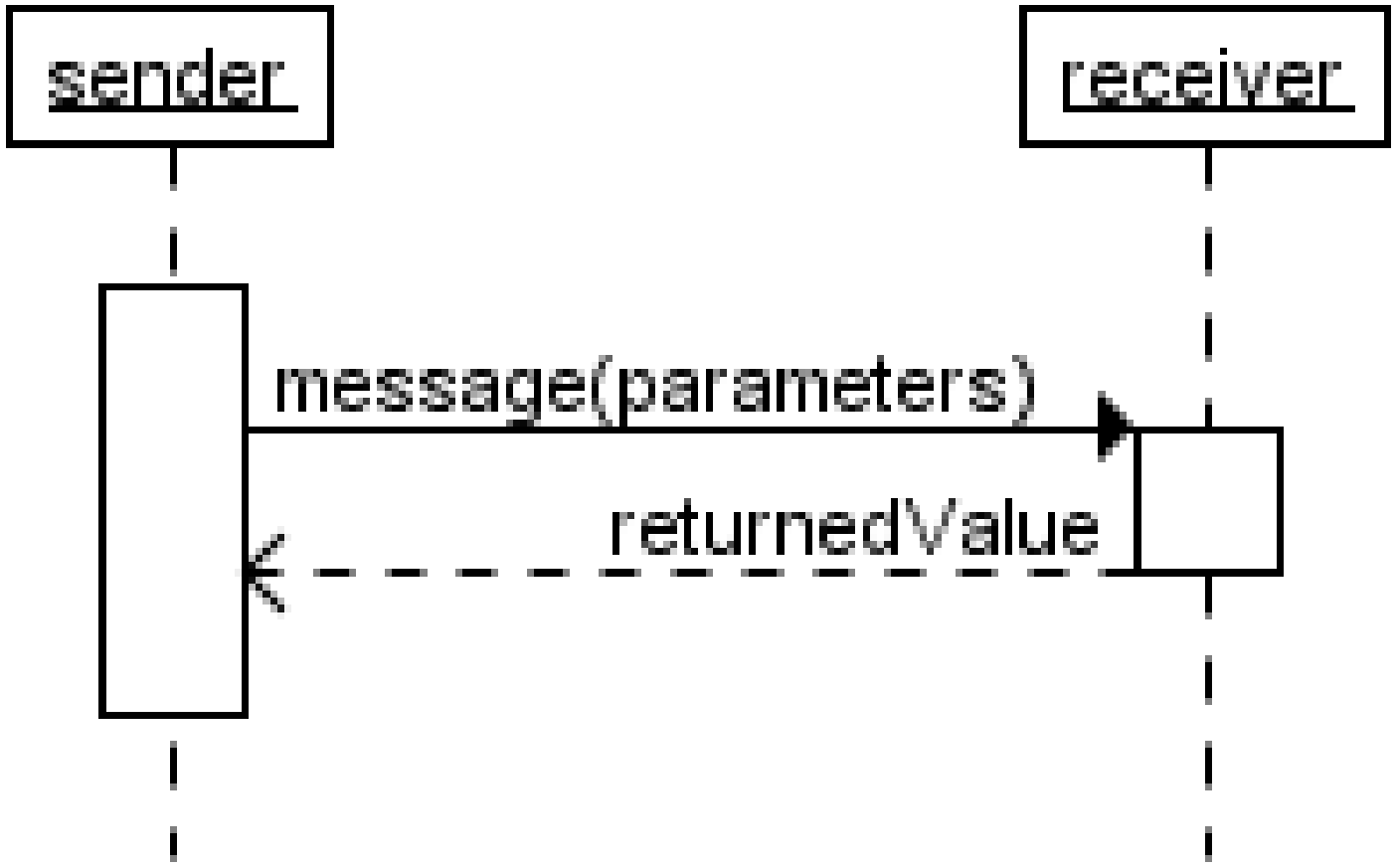
Sequence Diagrams Notations

- Synchronous message and returned value



Sequence Diagrams Notations

- Synchronous message and returned value



- If there is no return value, the dashed arrow can be omitted for clarity/readability

Sequence Diagrams Notations

- Synchronous message

- Asynchronous message

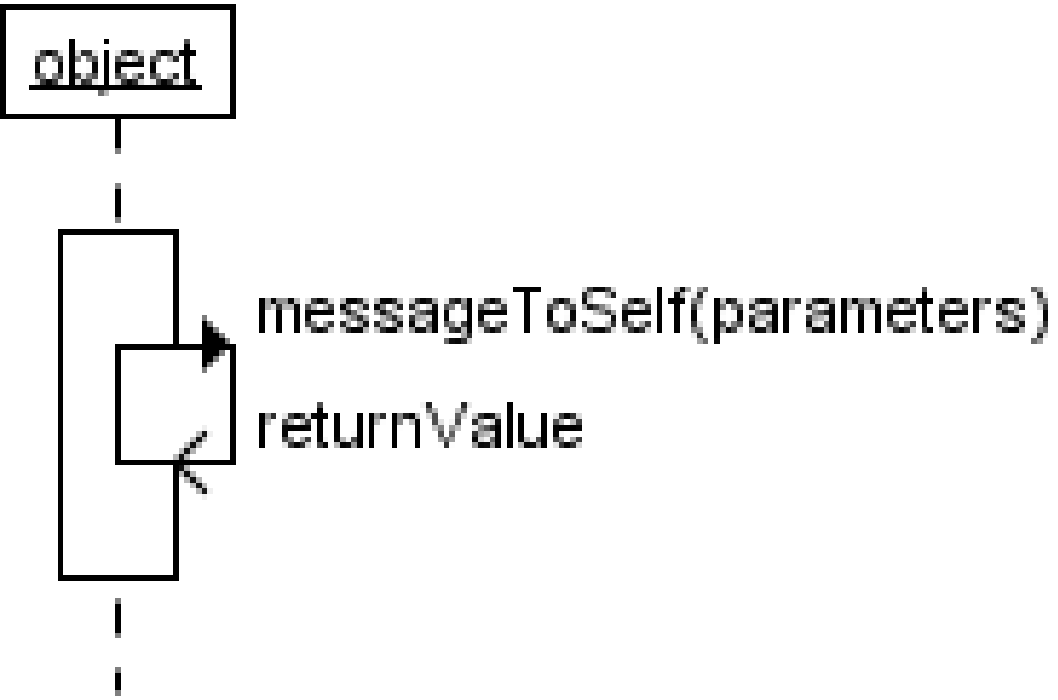
Sequence Diagrams Notations

- Synchronous message

- Asynchronous message (open arrow head)

Sequence Diagrams Notations

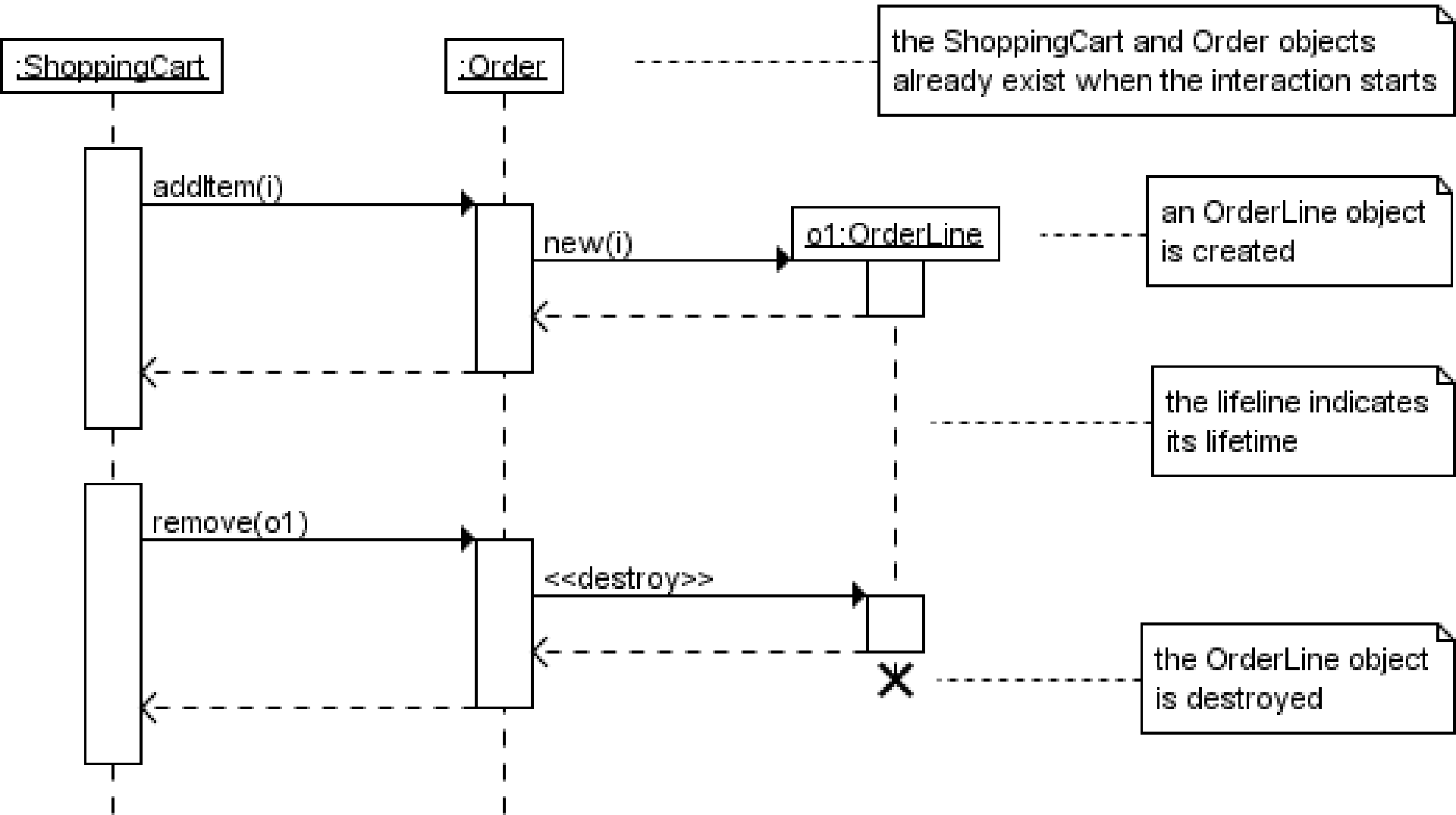
- Message to self



a message to self
and its return value

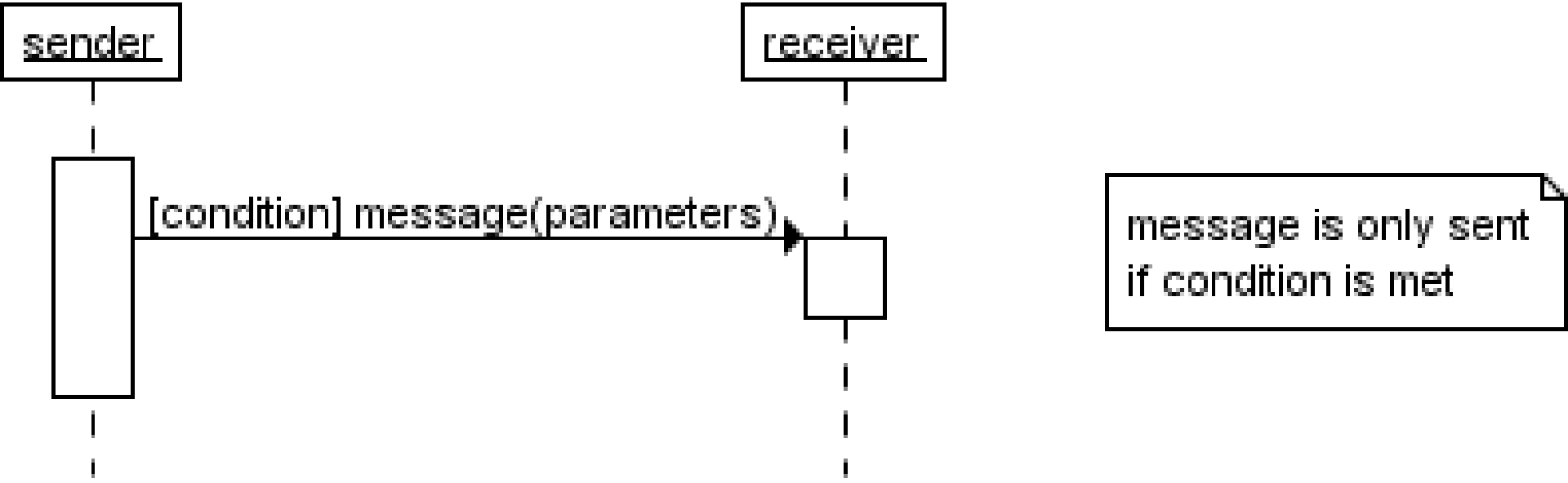
Sequence Diagrams Notations

- Creation and destruction



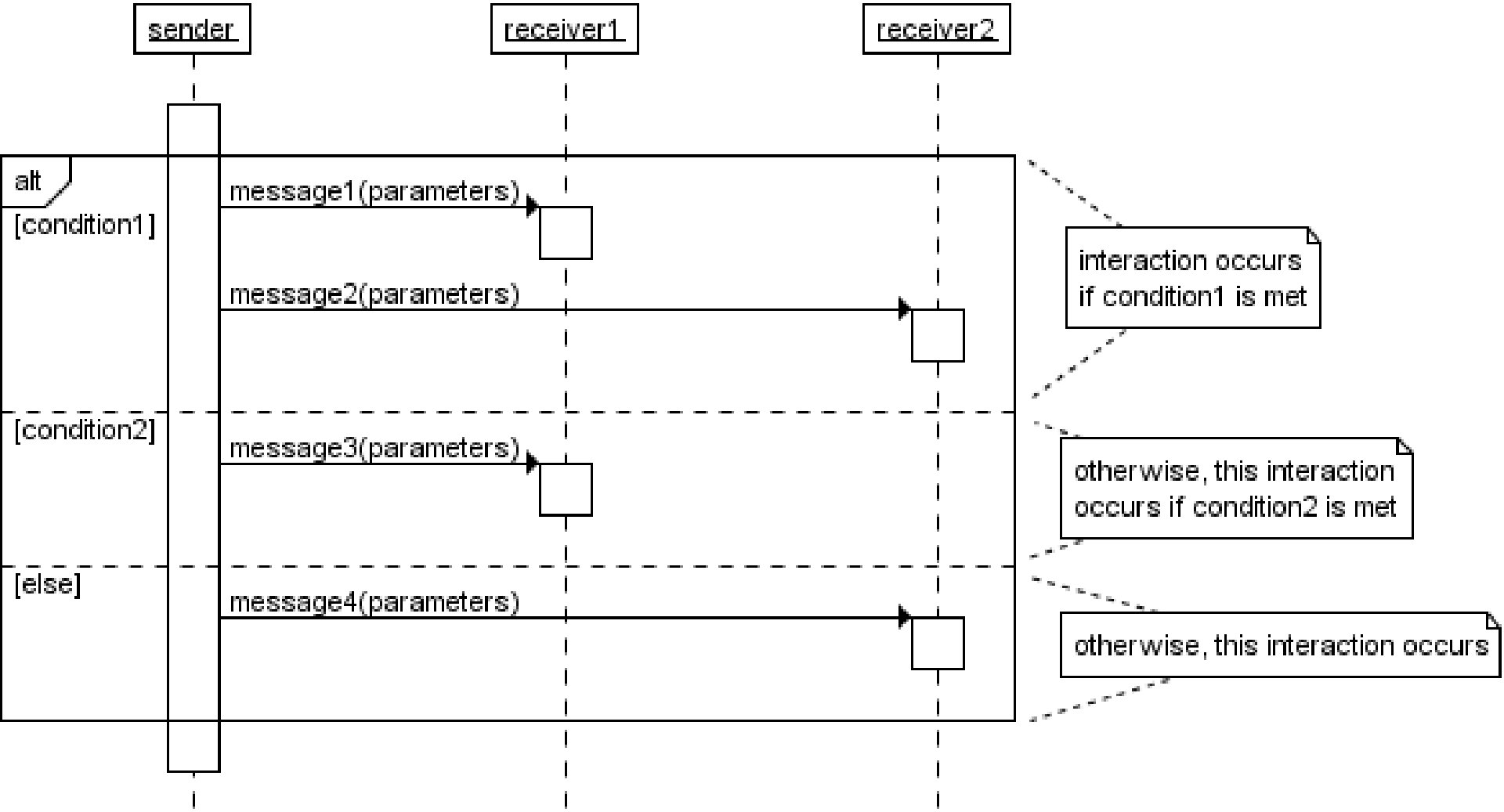
Sequence Diagrams Notations

- Conditional interaction



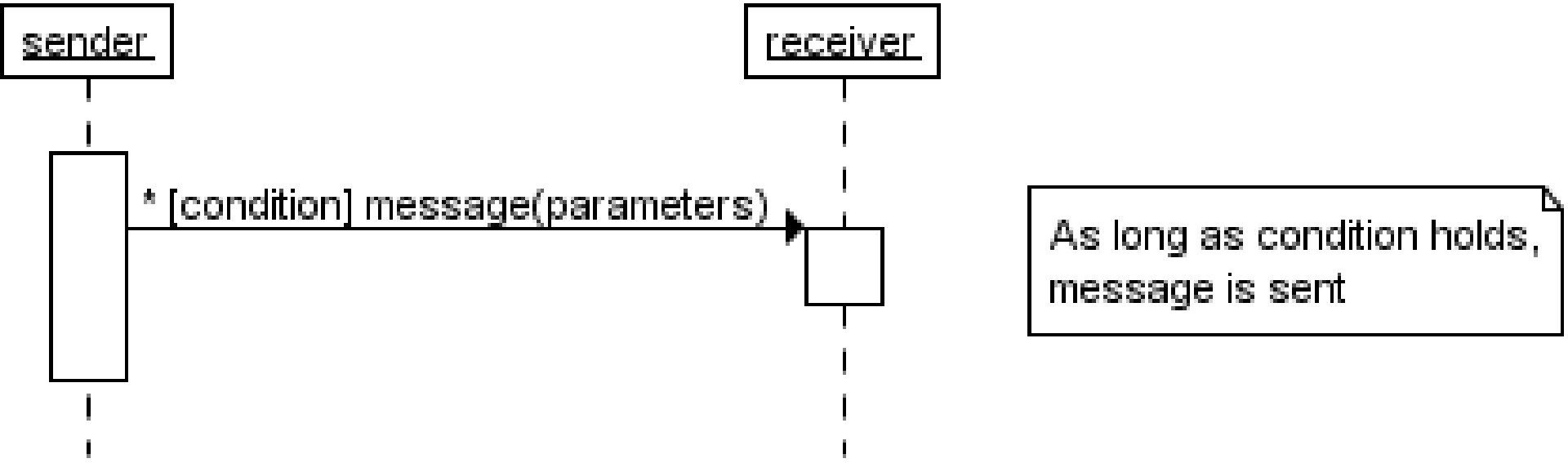
Sequence Diagrams Notations

- Conditional interaction



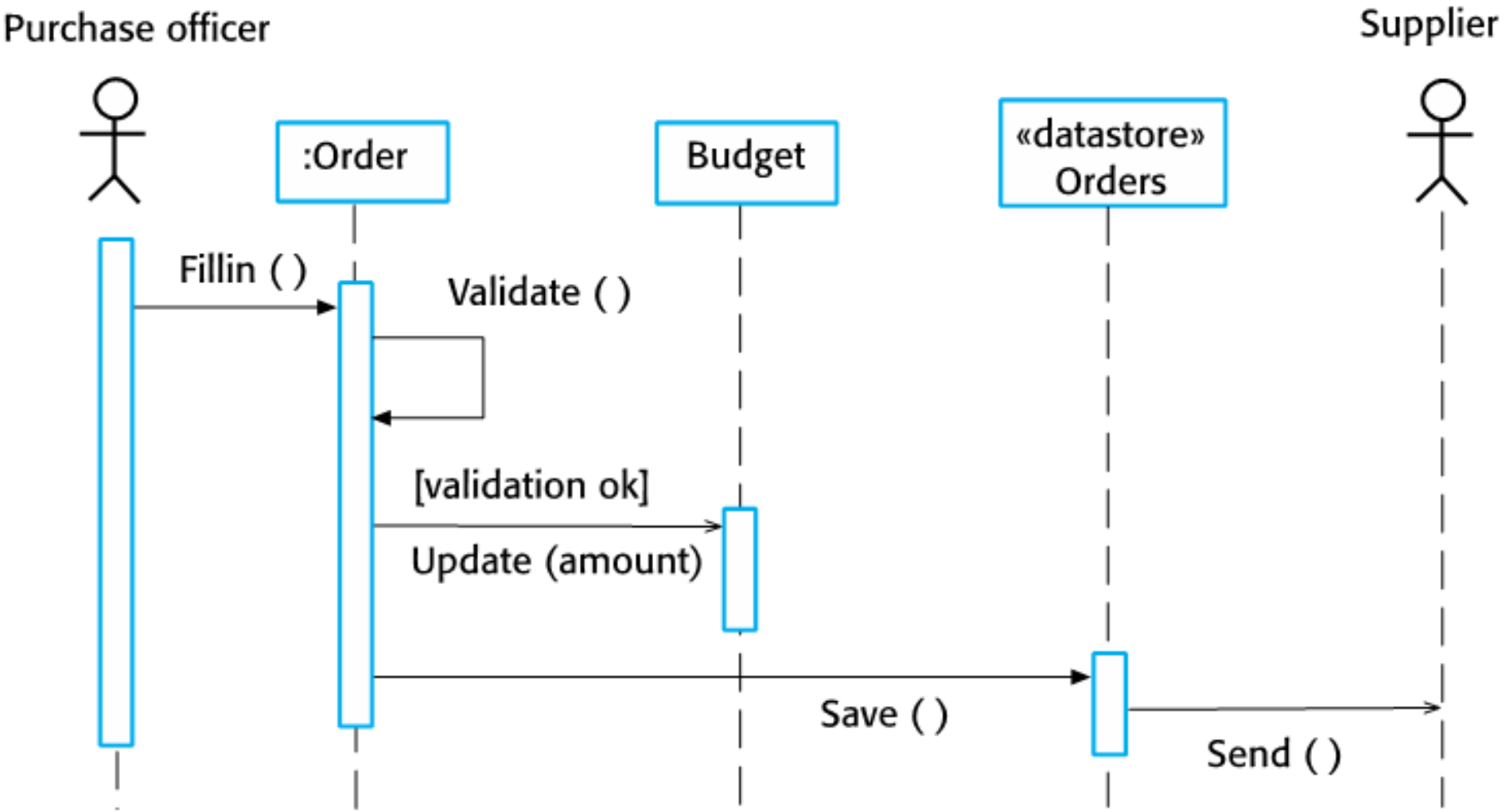
Sequence Diagrams Notations

- Repeated interaction (*)

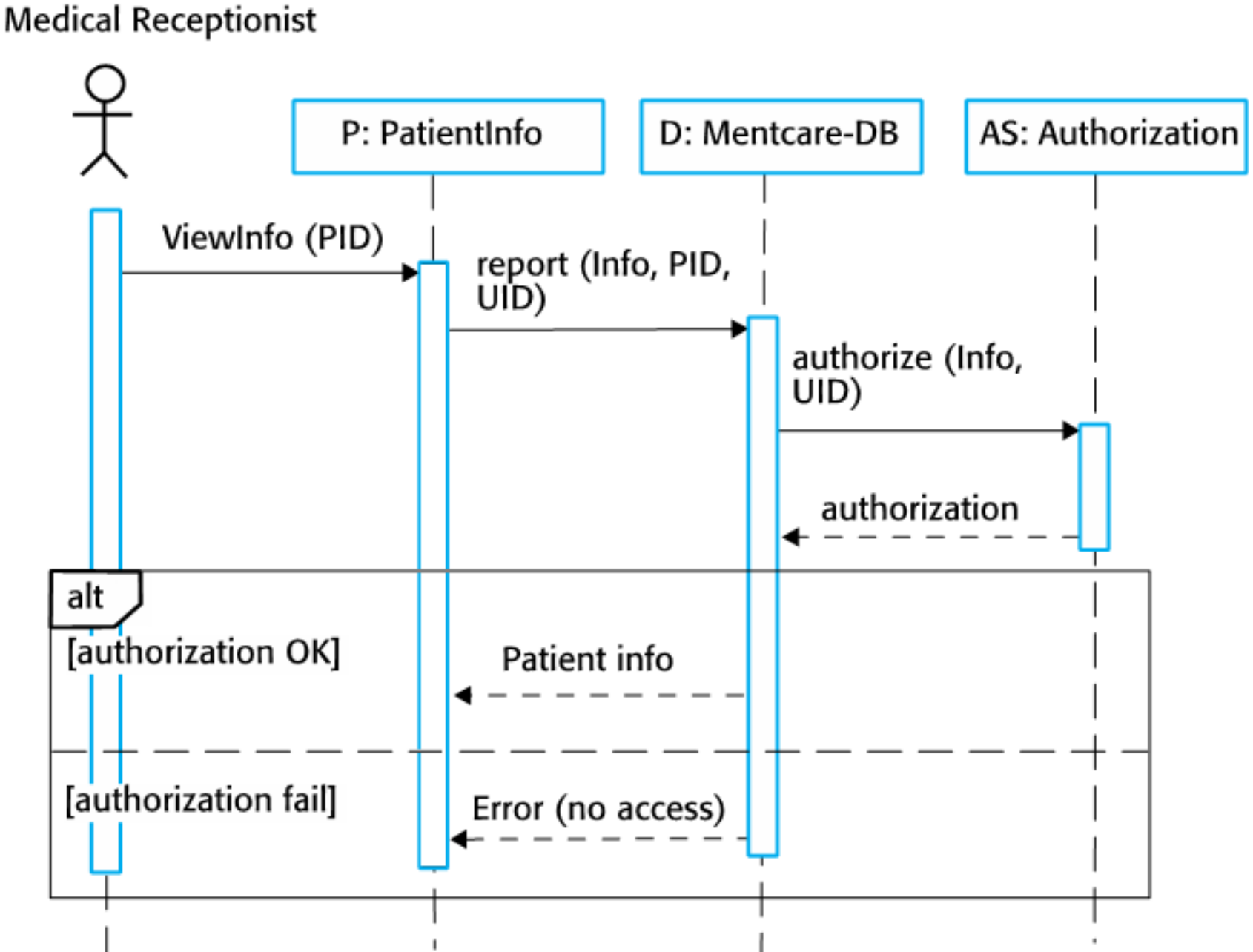


Sequence Diagrams Examples

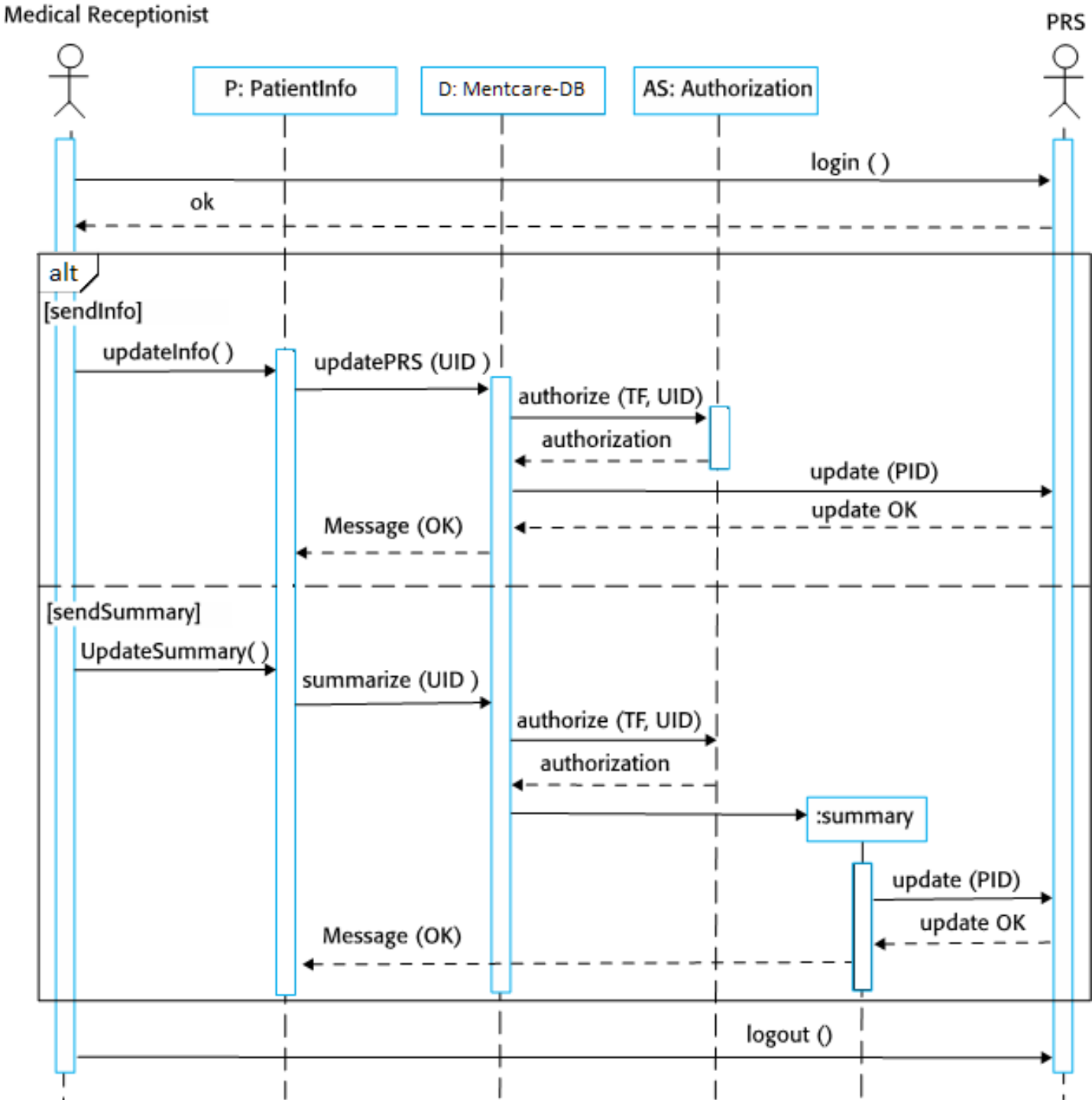
Order Processing



View Patient Information



Transfer Data



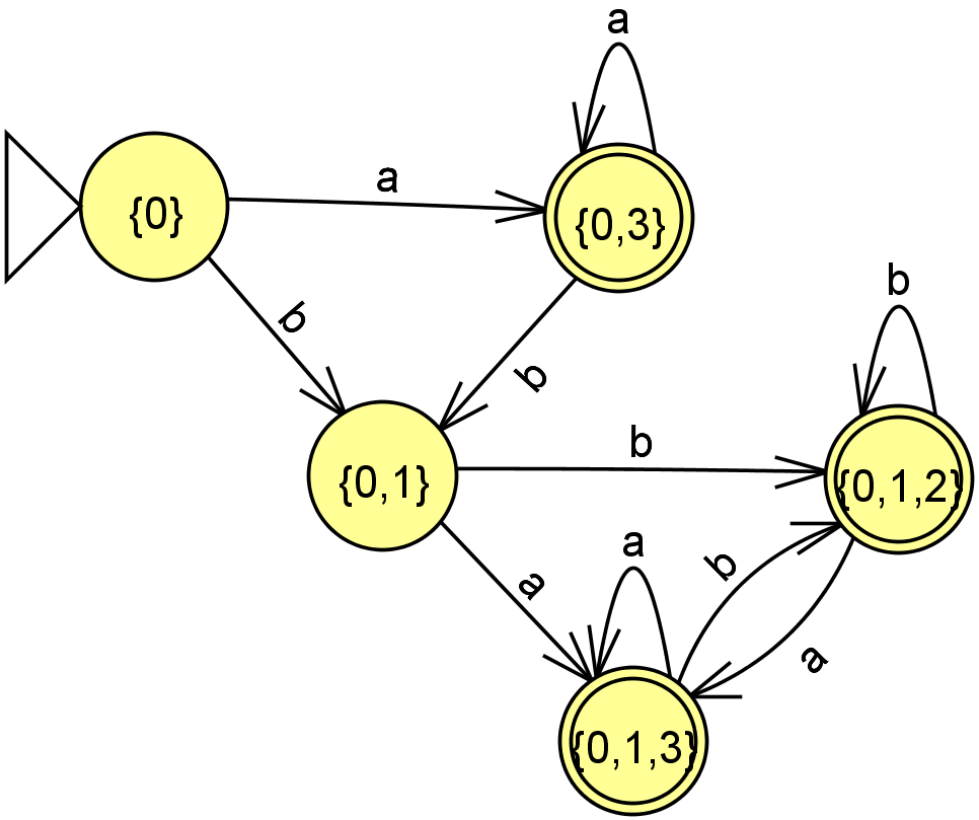
UML State Charts

UML State Chart

- AKA *UML State Machine*
- State Chart is an aid for the internal design of a class
- Illustrates how an object responds to events

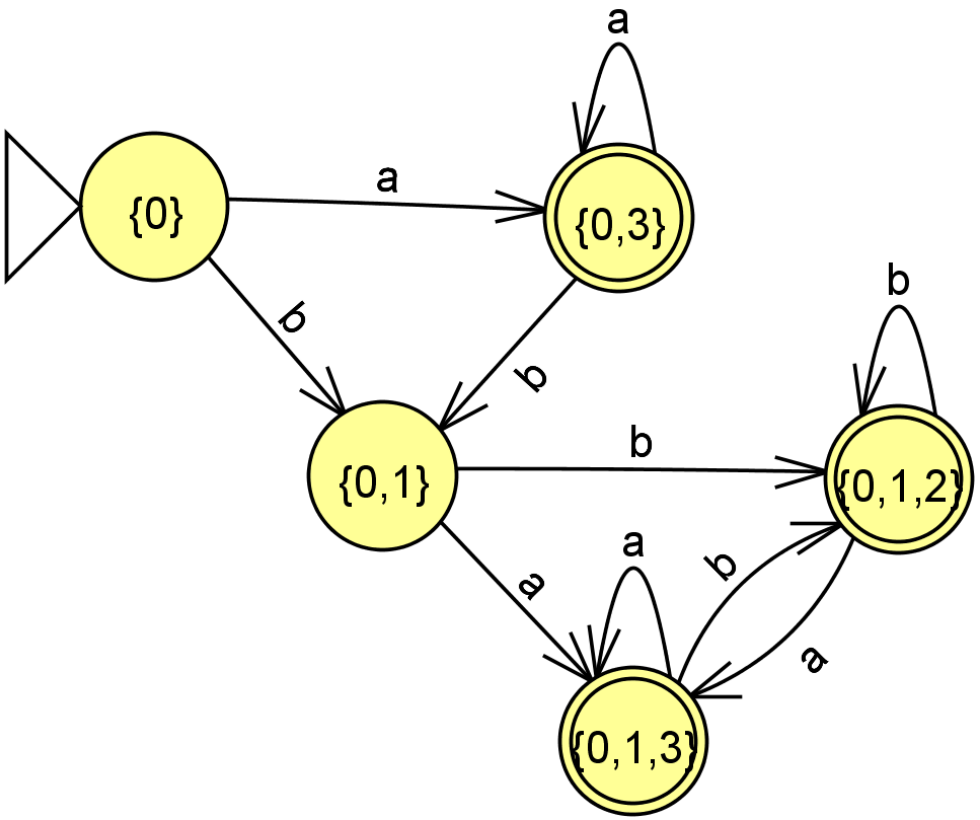
UML State Chart

- Related to a State Diagram for a Finite State Automaton in CS361



UML State Chart

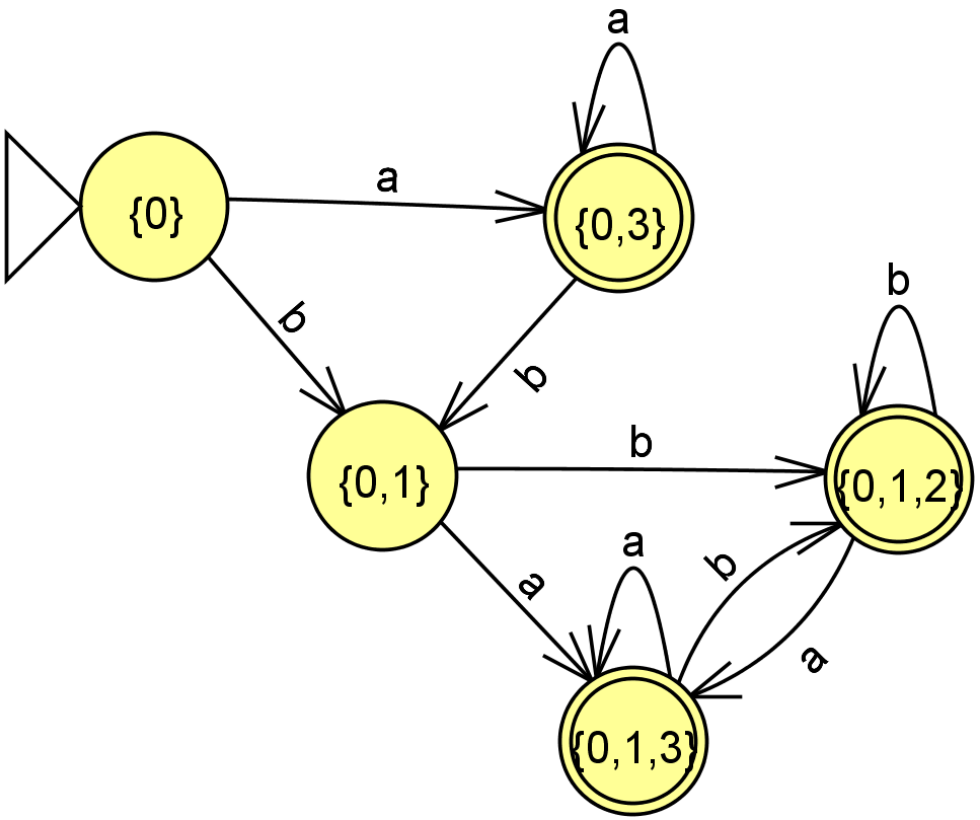
- Related to a State Diagram for a Finite State Automaton in CS361



- Examples of accepted inputs:
- Examples of rejected inputs:

UML State Chart

- Related to a State Diagram for a Finite State Automaton in CS361



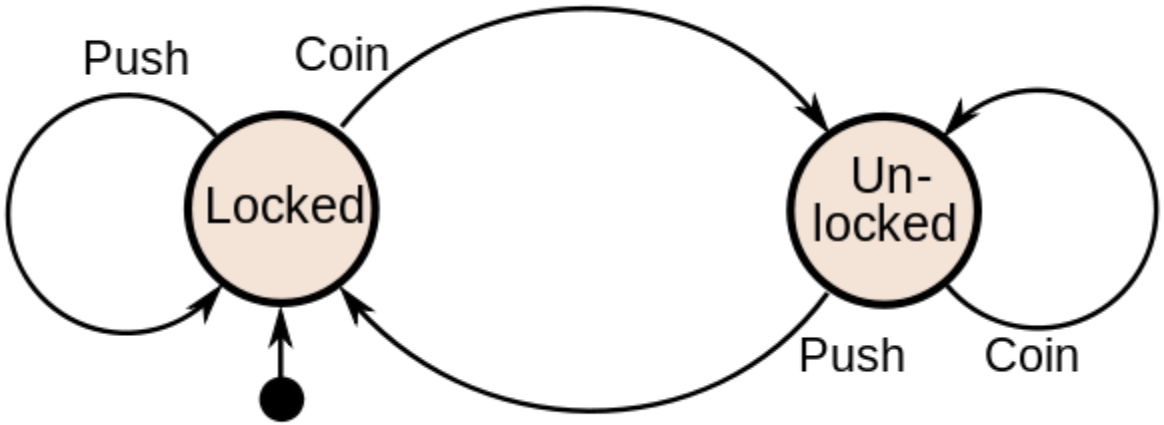
- Examples of accepted inputs:
  - a, aa, bbb
- Examples of rejected inputs:
  - $b, ab, aaaab$

Design a "Finite State Automaton"-like diagram for modeling the CAPSLOCK button on a keyboard (whiteboard only)

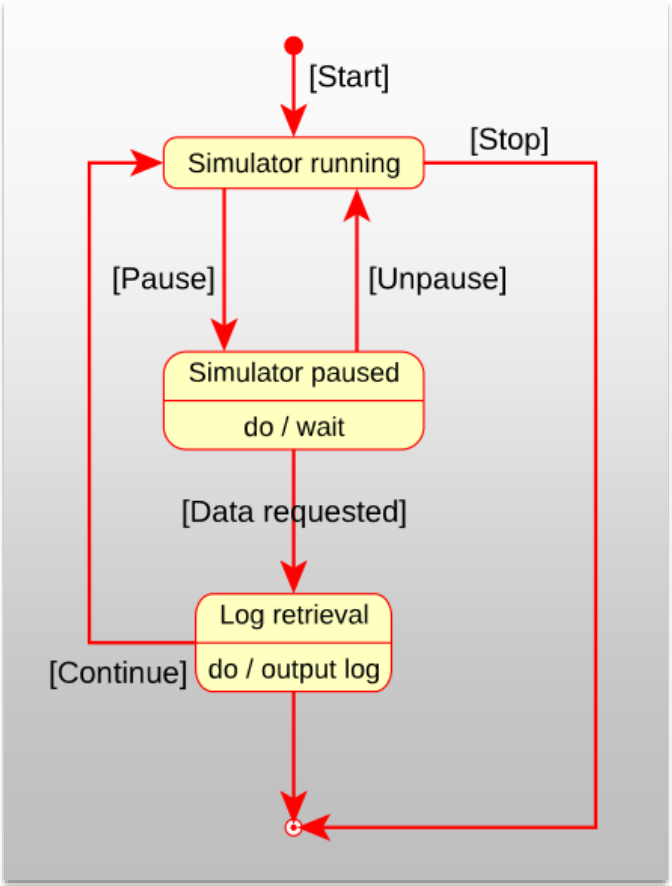
- Represent the states
- Represent the transitions

Example State Diagram (turnstile)

Example State Diagram (turnstile)



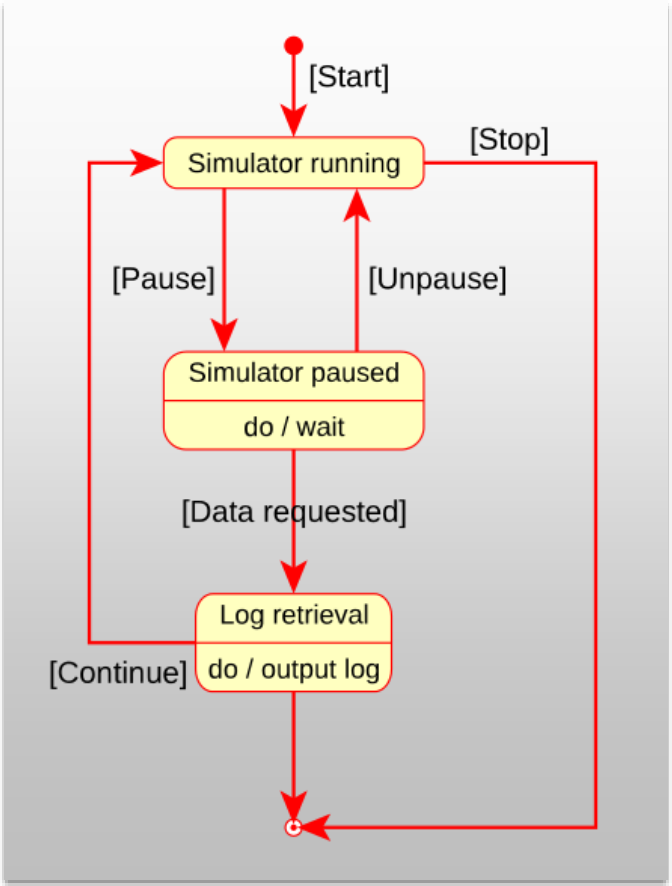
UML Statechart: Basic Structure



https://en.wikipedia.org/wiki/File:UML_State_diagram.svg

UML Statechart: Basic Structure

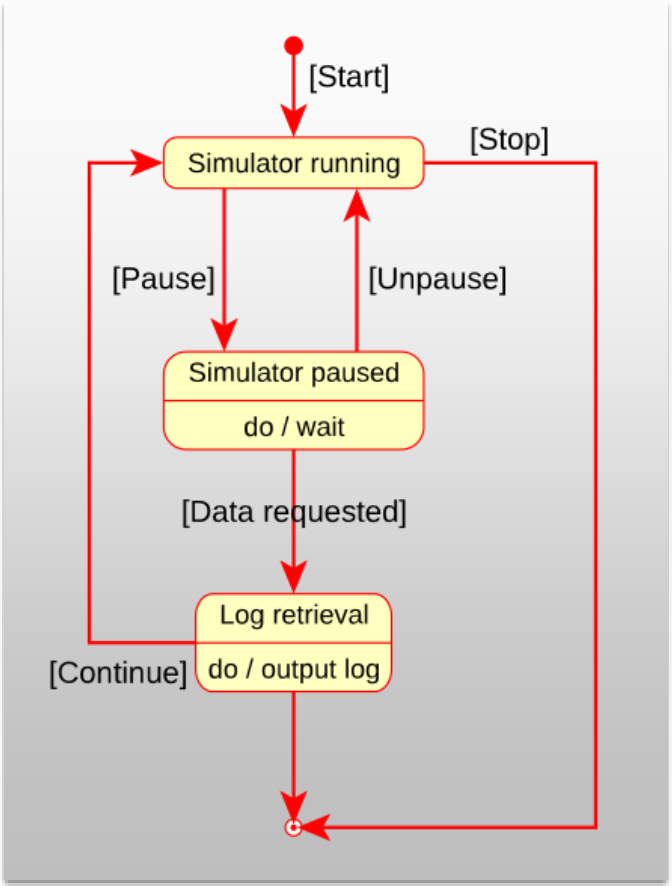
- Filled circle: Origin of the initial (object constructor's) transition



https://en.wikipedia.org/wiki/File:UML_State_diagram.svg

UML Statechart: Basic Structure

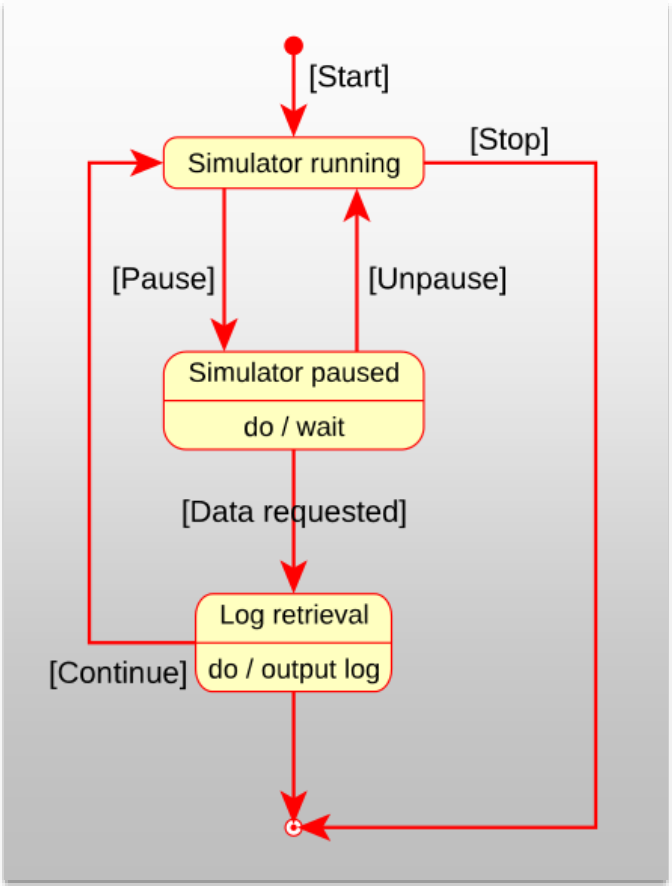
- Filled circle: Origin of the initial (object constructor's) transition
- Rounded rectangles: Object states
  - Name appears on top line
  - Optional activities appear in bottom line



https://en.wikipedia.org/wiki/File:UML_State_diagram.svg

UML Statechart: Basic Structure

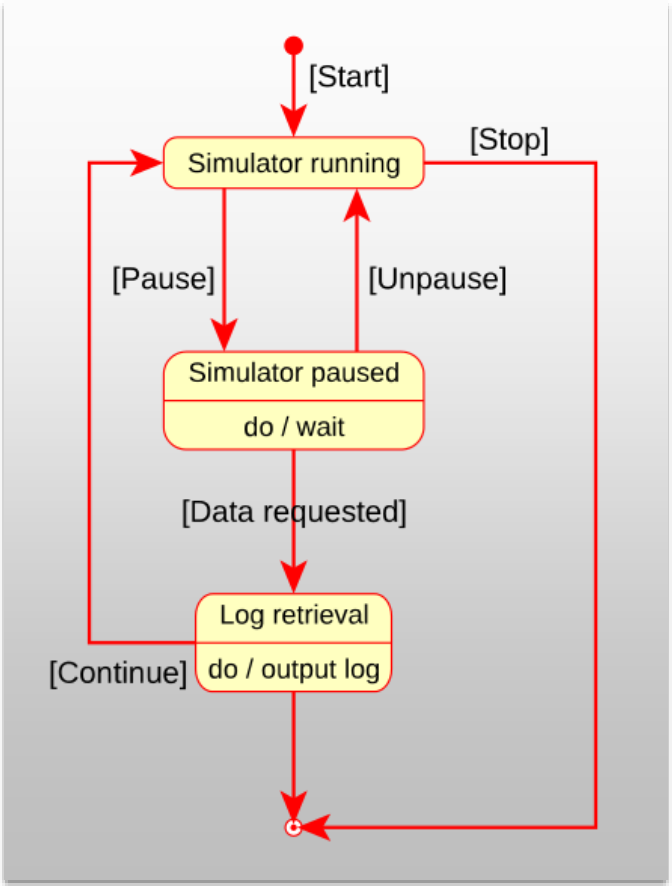
- Filled circle: Origin of the initial (object constructor's) transition
- Rounded rectangles: Object states
  - Name appears on top line
  - Optional activities appear in bottom line
- Arrows: State transitions
  - Optional event name: `Panic`
  - Optional guard: `[nZombies>0]`
  - Optional action: `/runAway()`



https://en.wikipedia.org/wiki/File:UML_State_diagram.svg

UML Statechart: Basic Structure

- Filled circle: Origin of the initial (object constructor's) transition
- Rounded rectangles: Object states
  - Name appears on top line
  - Optional activities appear in bottom line
- Arrows: State transitions
  - Optional event name: `Panic`
  - Optional guard: `[nZombies>0]`
  - Optional action: `/runAway()`
- Hollow circle: Final state (uncommon)



https://en.wikipedia.org/wiki/File:UML_State_diagram.svg

State Chart Basic Concepts

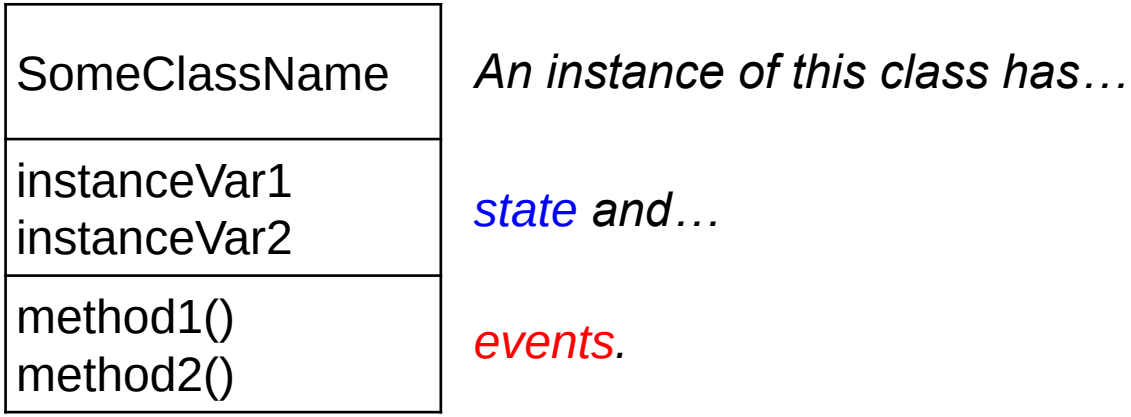
- The *state* of an object is defined by its *instance variables*

- An *event* is triggered by invoking an object's *method*

State Chart Basic Concepts

- The *state* of an object is defined by its *instance variables*

- An *event* is triggered by invoking an object's *method*



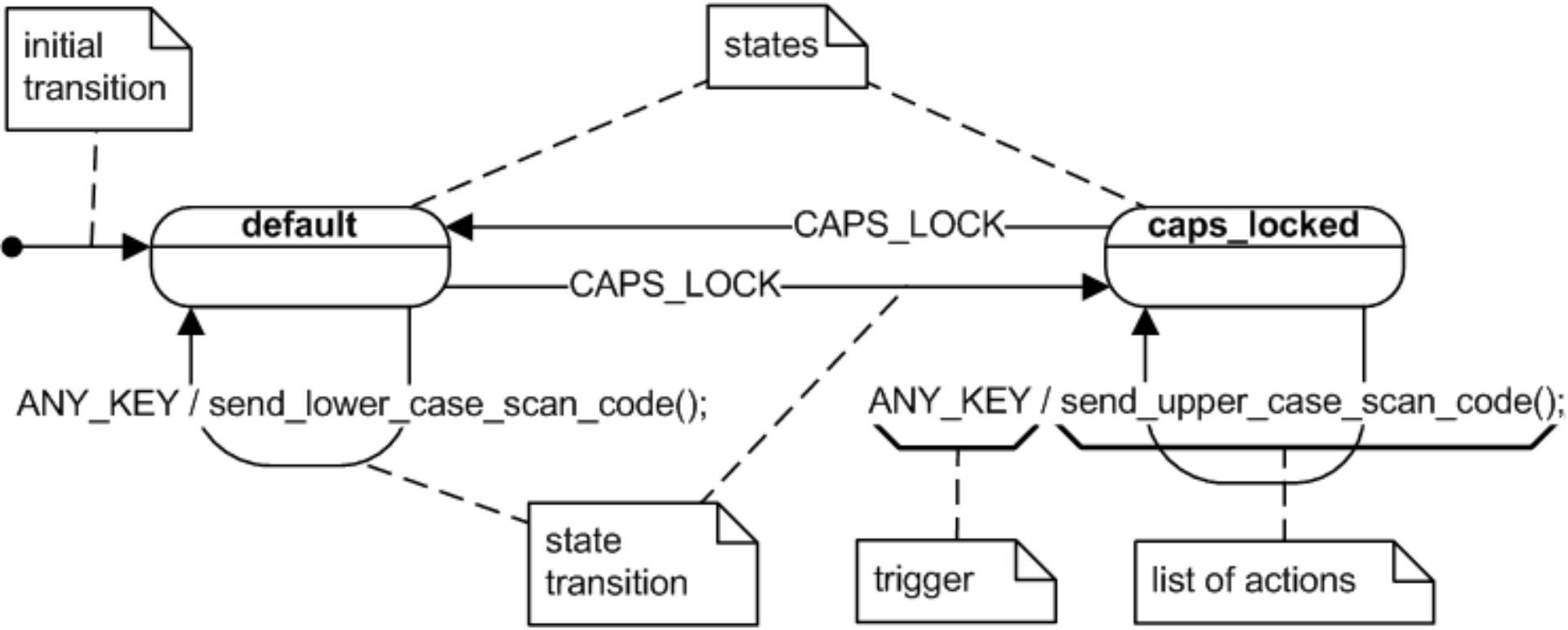
A Few Details about *State*

- Any object (having one or more instance variables) is in exactly one state at any given time
- You may think of that state as the bits in the complete set of instance variables
- An object's initial state is established by its constructor

Some Details About Events

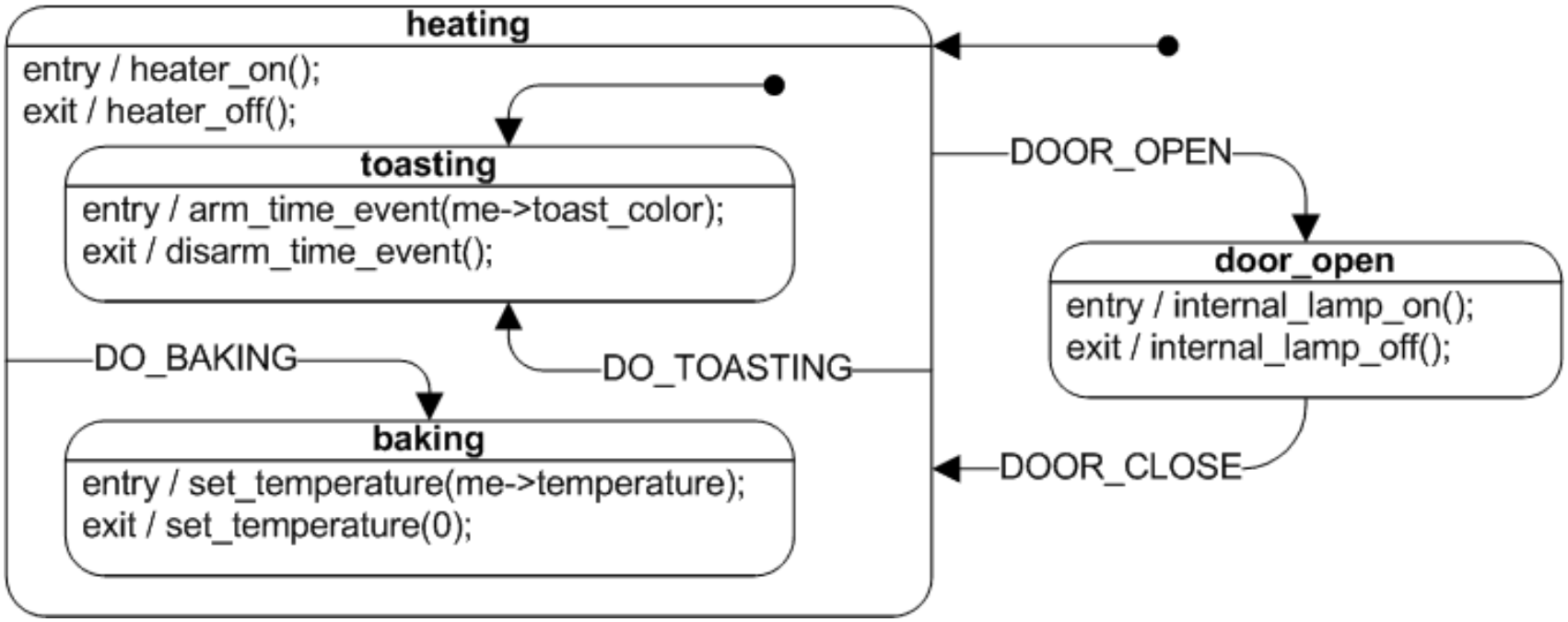
- Changes in state are known as state transitions
- Events may trigger state transitions

UML Statechart: Realistic Keyboard Example



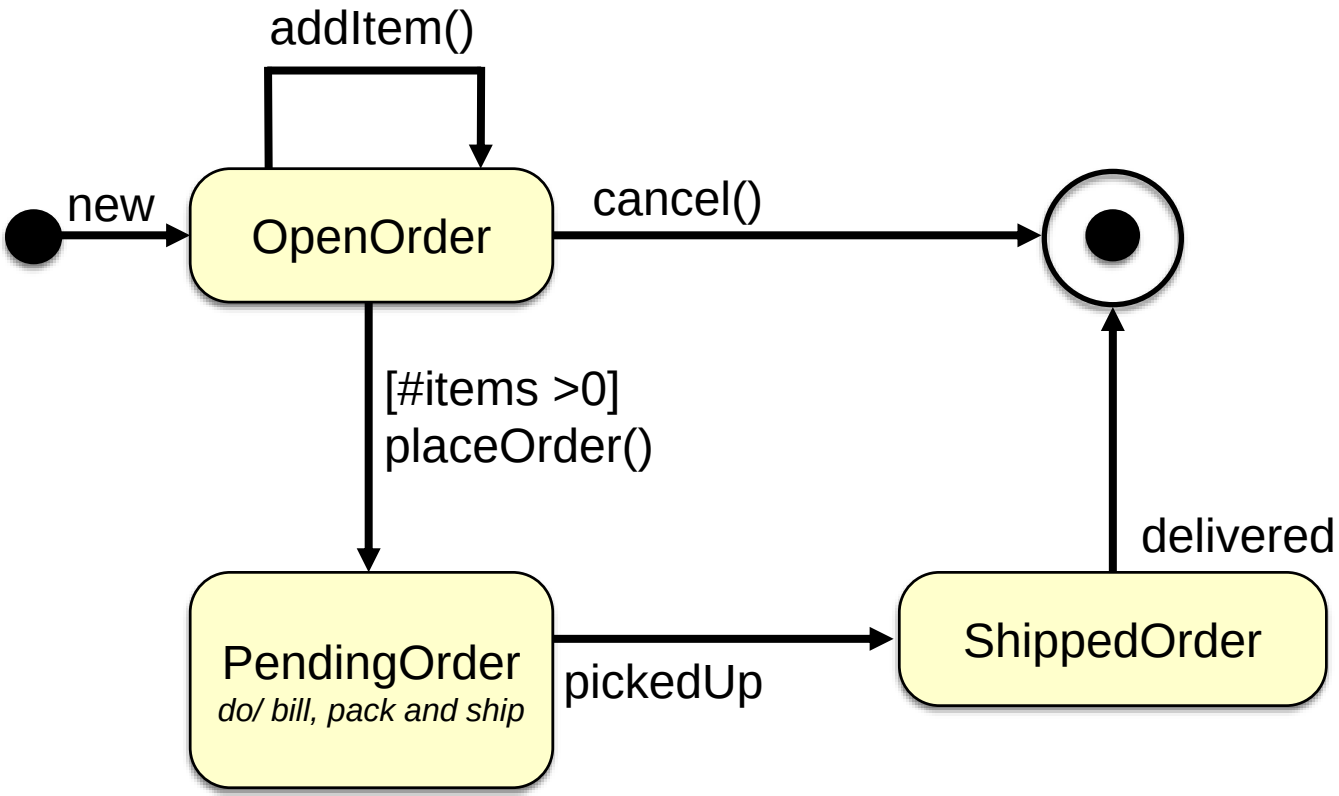
https://en.wikipedia.org/wiki/File:UML_state_machine_Fig1.png

UML Statechart example: Toaster Oven

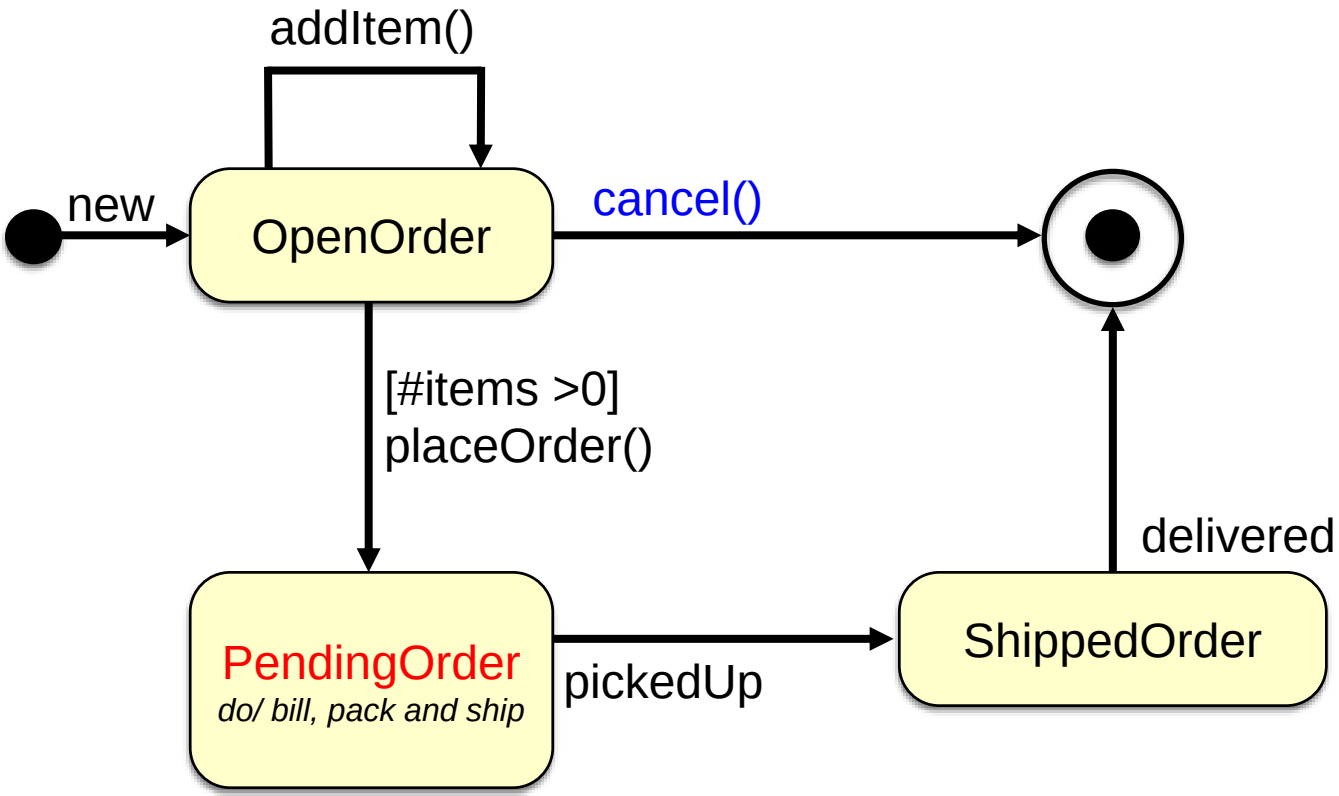


https://upload.wikimedia.org/wikipedia/commons/2/20/UML_state_machine_Fig5.png

Example State Chart

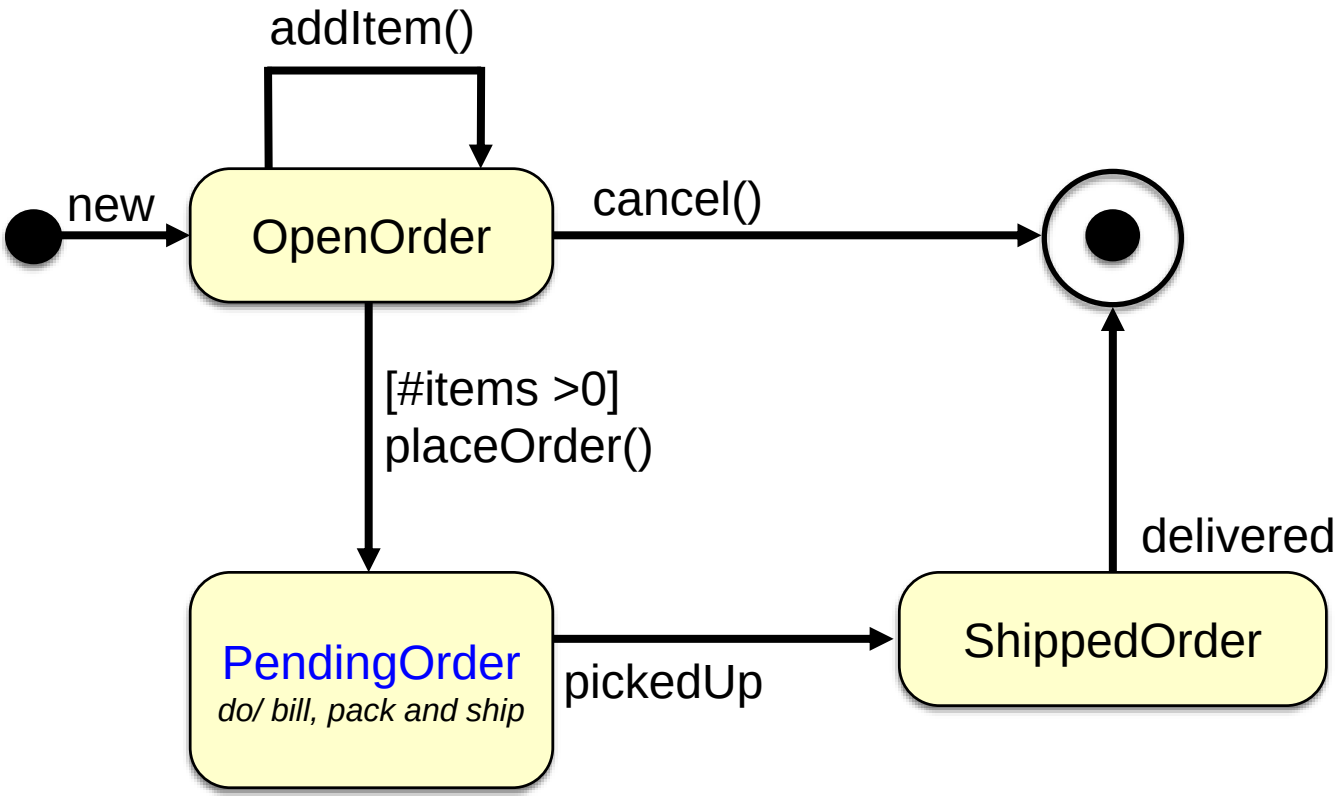


Example State Chart



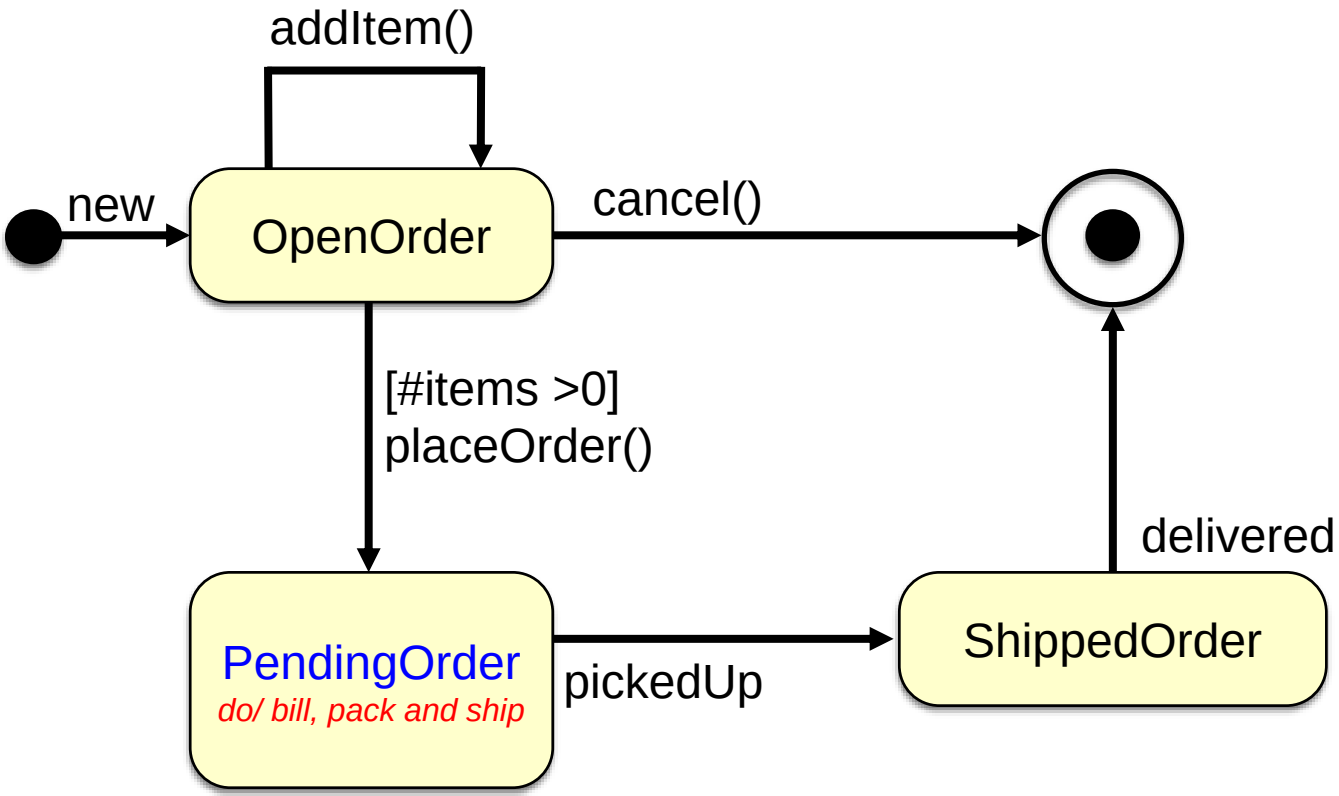
- The chart illustrates the option to `cancel()` an open order
  - And our inability to cancel an order once it has been placed

Example State Chart



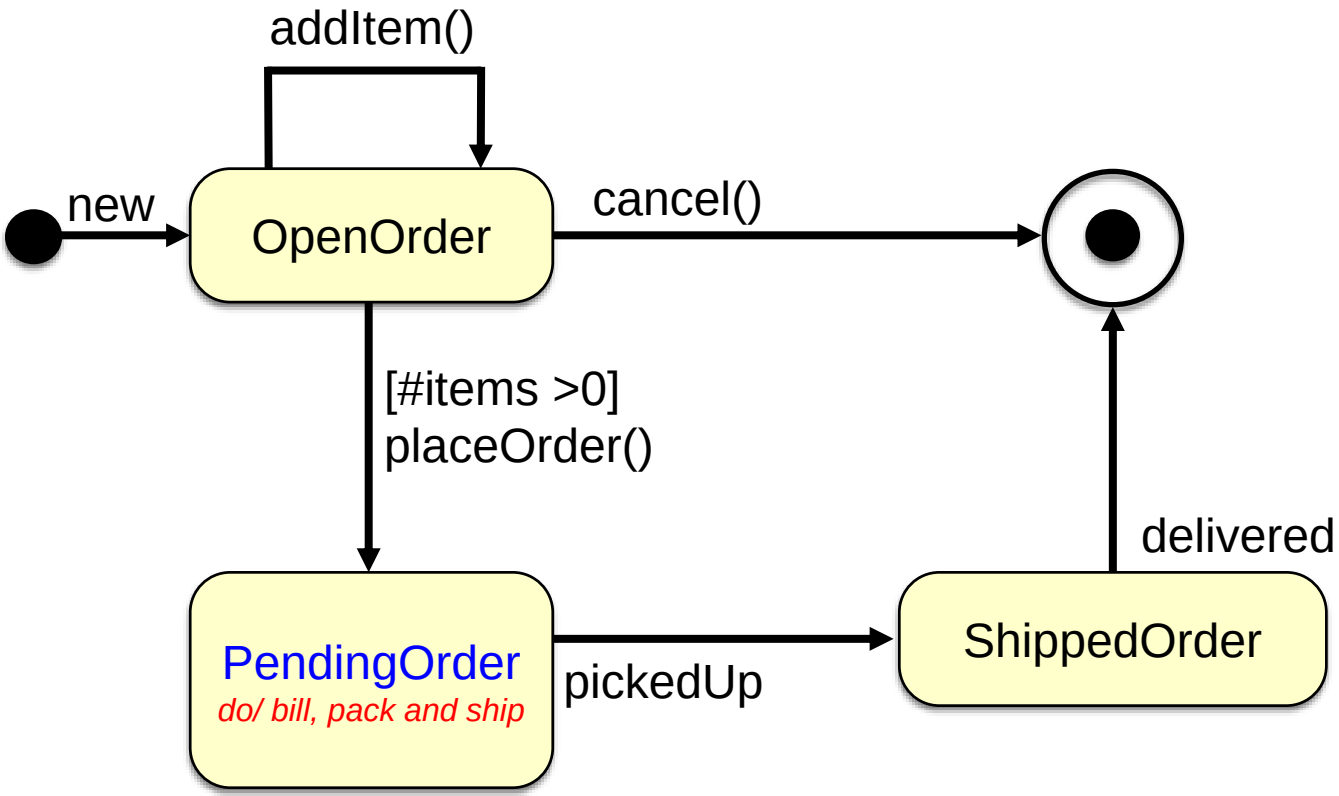
- A `PendingOrder` has been placed by the Customer but has not been picked-up by a carrier (e.g., UPS)

Example State Chart



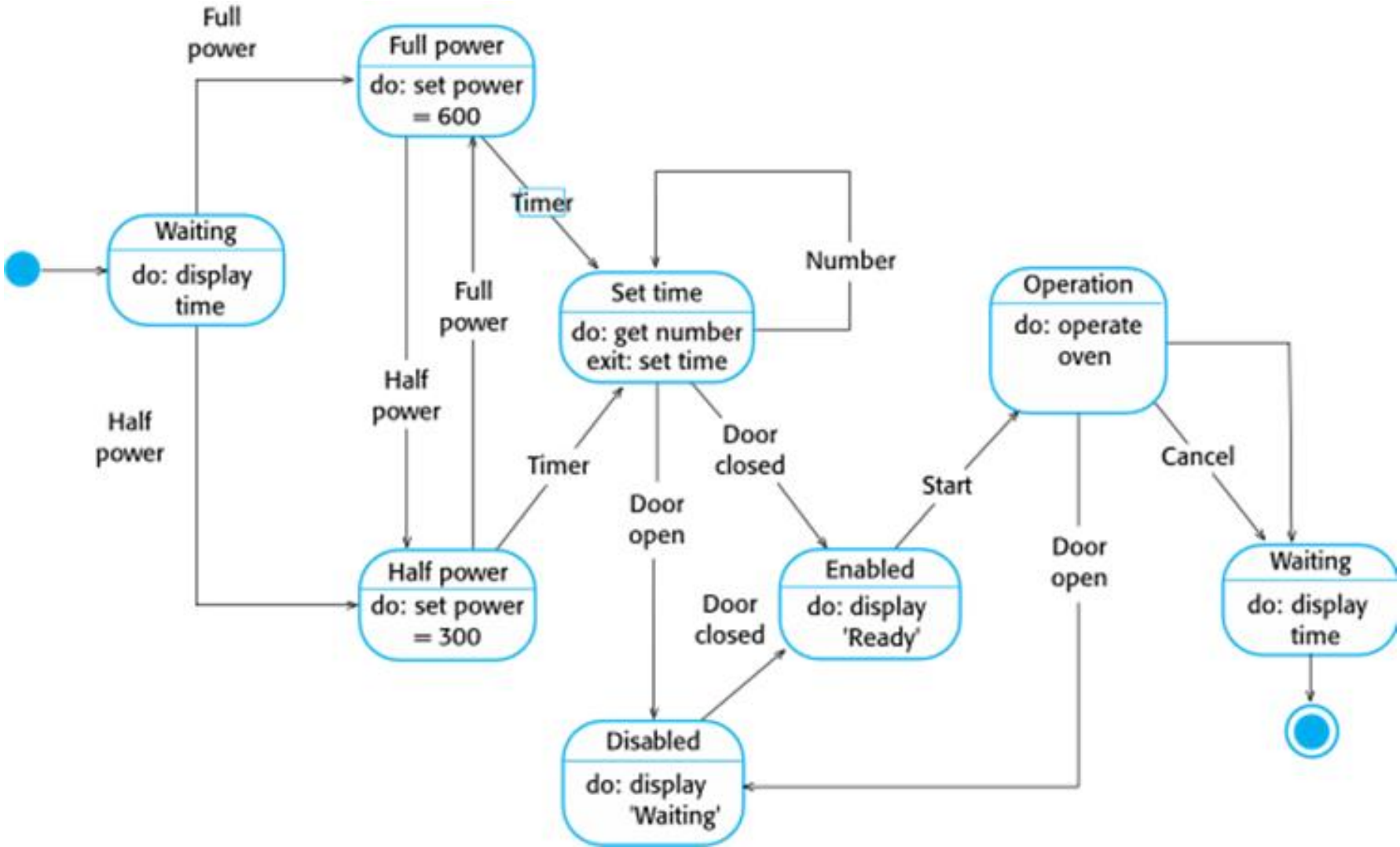
- `PendingOrder` departs significantly from CS361. It contains *activities* (see *do/*activity) that are not modeled here in detail:

Example State Chart

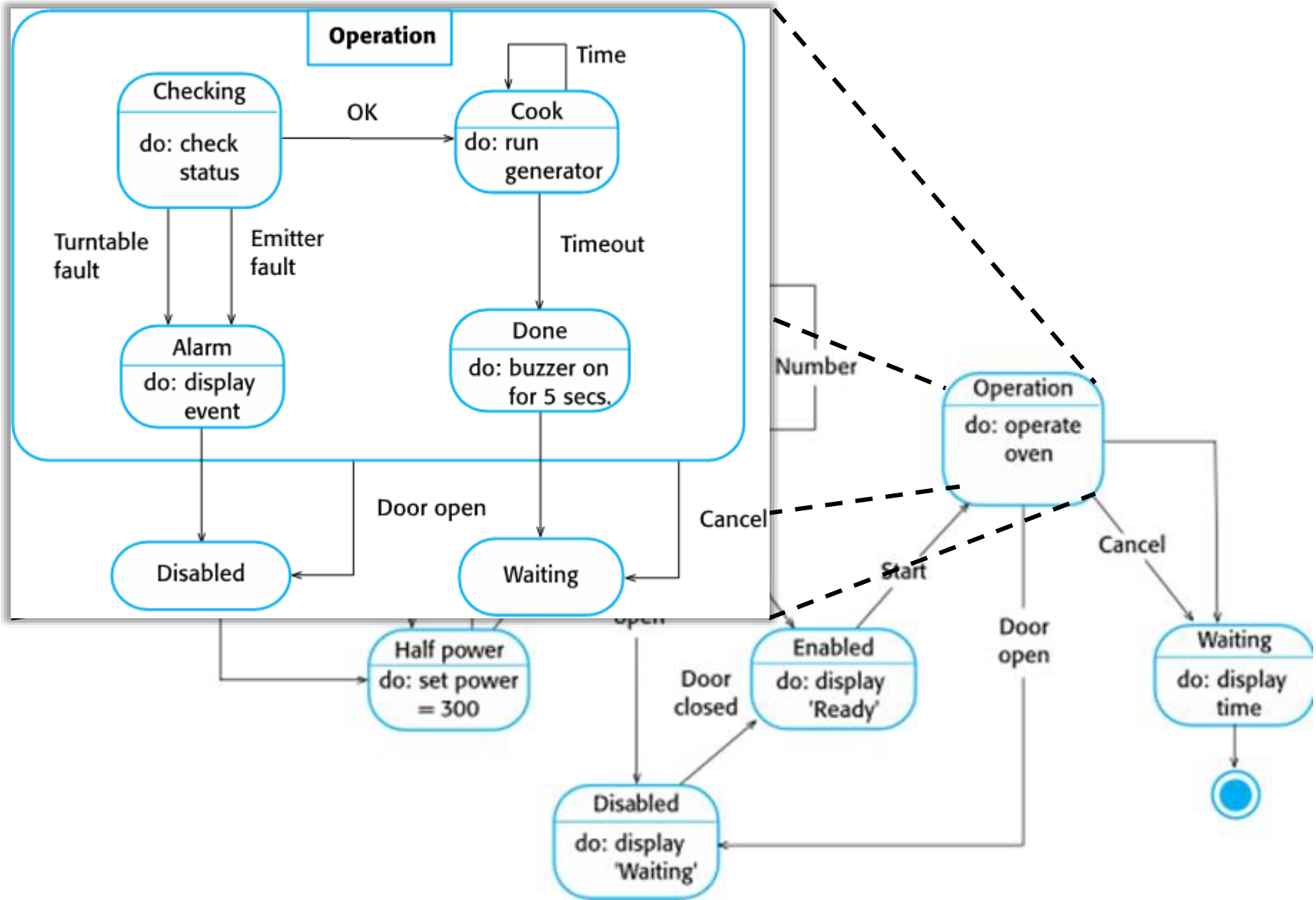


- `PendingOrder` departs significantly from CS361. It contains *activities* (see *do/*activity) that are not modeled here in detail:
  - We billed the Customer's credit card
  - We printed the pick list for the warehouse workers
  - We printed the shipping label (and paid the carrier)
  - The real-world order awaits pick-up on our dock

Example State Chart – Microwave Oven



Example State Chart – Microwave Oven



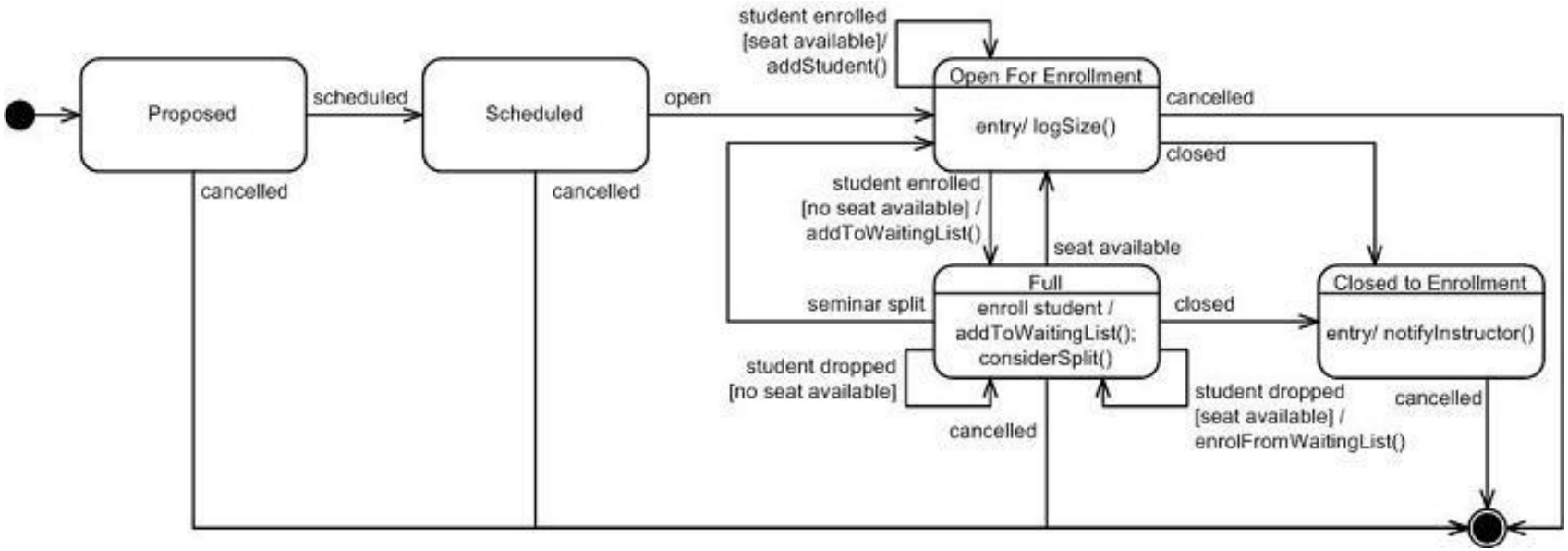
Example State Chart – Microwave Oven

State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows ‘Half power’.
Full power	The oven power is set to 600 watts. The display shows ‘Full power’.
Set time	The cooking time is set to the user’s input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows ‘Not ready’.
Enabled	Oven operation is enabled. Interior oven light is off. Display shows ‘Ready to cook’.
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for five seconds. Oven light is on. Display shows ‘Cooking complete’ while buzzer is sounding.

Example State Chart – Microwave Oven

Stimulus/Transition	Description
Half power	The user has pressed the half-power button.
Full power	The user has pressed the full-power button.
Timer	The user has pressed one of the timer buttons.
Number	The user has pressed a numeric key.
Door open	The oven door switch is not closed.
Door closed	The oven door switch is closed.
Start	The user has pressed the Start button.
Cancel	The user has pressed the Cancel button.

Example – seminar during registration

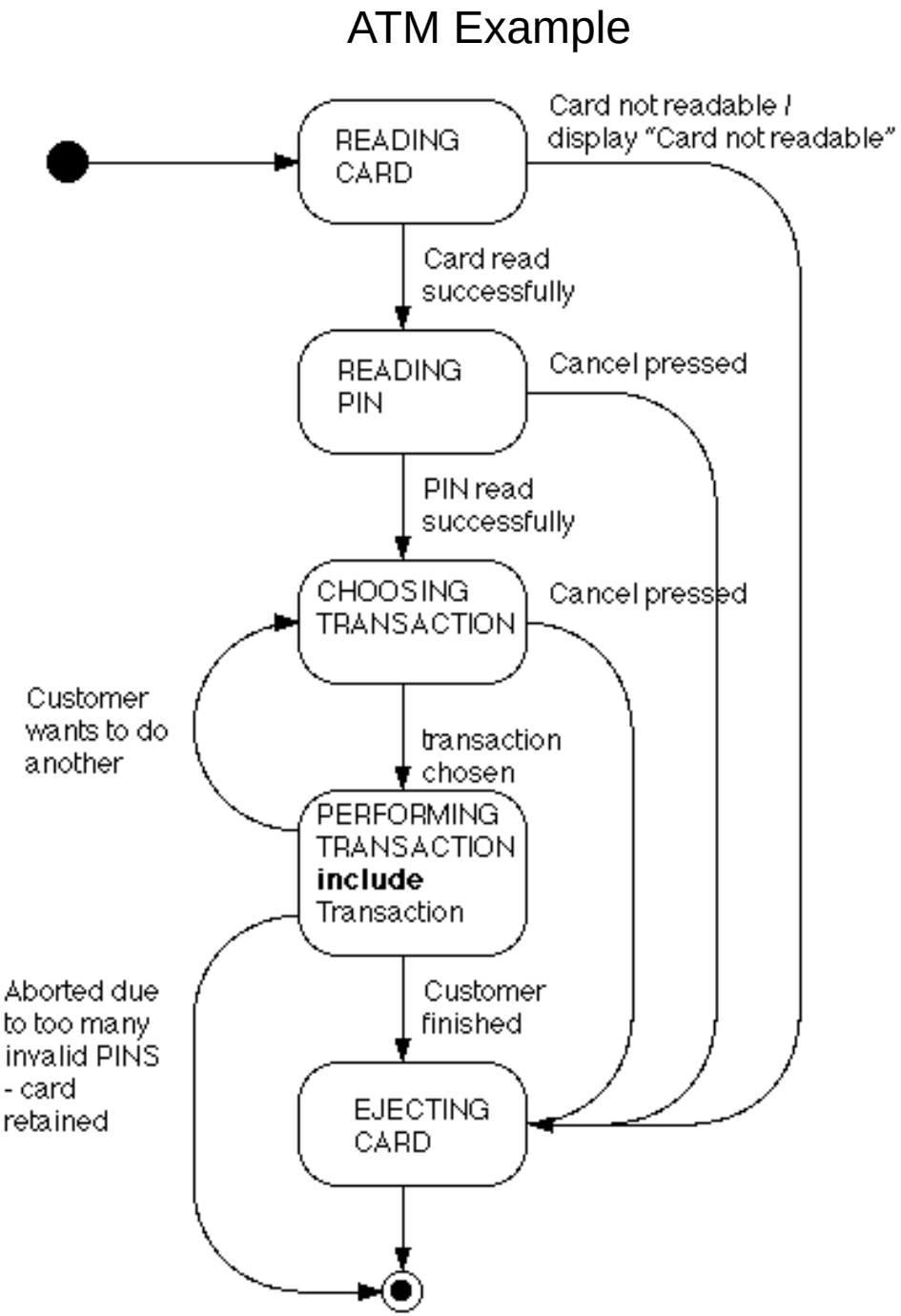


When to Use UML State Charts?

When to Use UML State Charts?

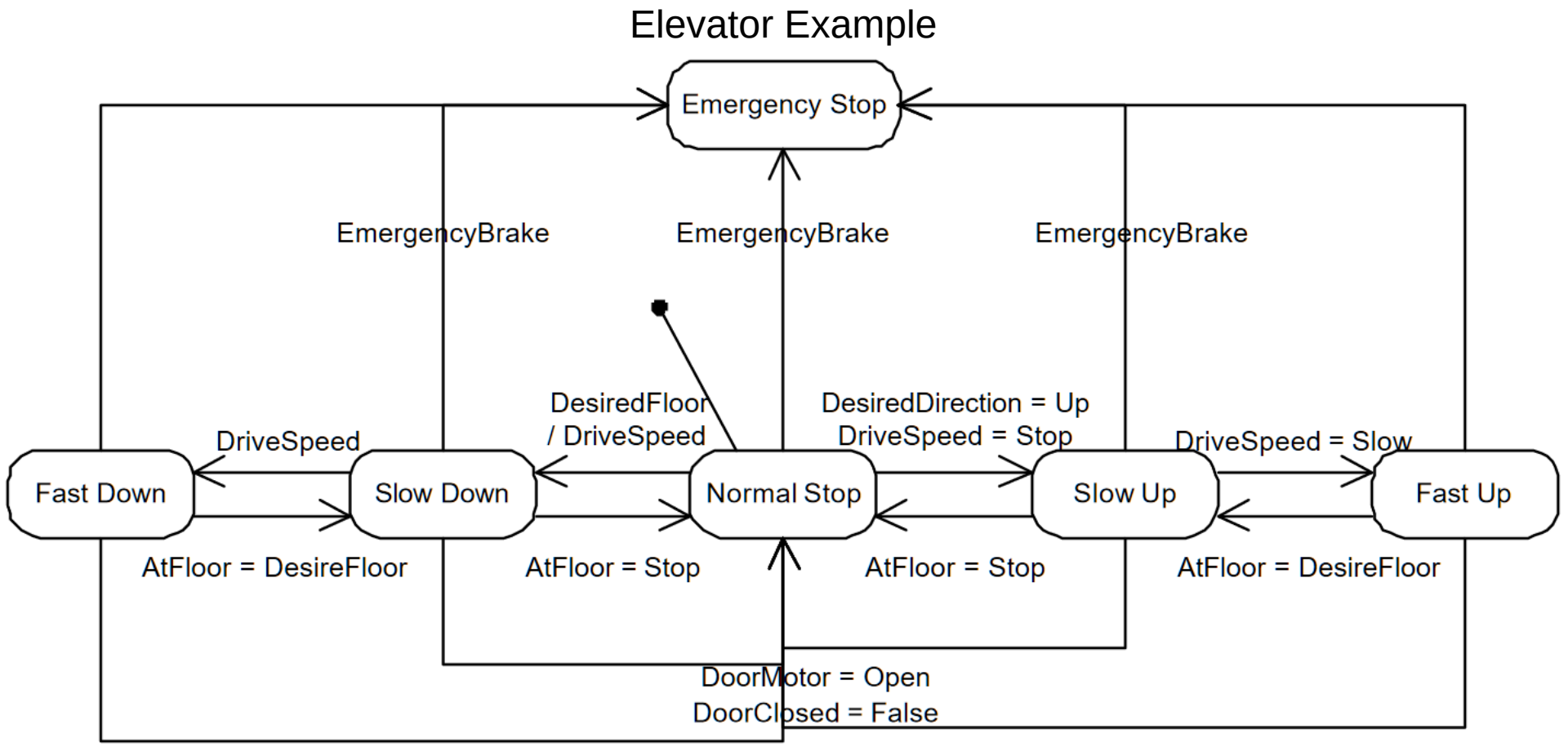
- Design
  - When you're working by yourself on a complex class
    - Not all methods are supported in all states
    - Action taken by a method depends upon state
  - When you're explaining how to use a complex class to another developer
- Documentation
  - Complex classes (if you need a state chart to design a class, those who follow you will benefit from it as well)
  - Software developed on contract

UML State Charts are particularly useful for "hardware"-related projects or projects with hardware components



http://www.math-cs.gordon.edu/courses/cs211/ATMExample/Statecharts.html

UML State Charts are particularly useful for "hardware"-related projects or projects with hardware components



https://www.cs.cmu.edu/~luluo/Courses/18540PhDreport.pdf

Popular UML Diagrams

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- State Machine Diagram

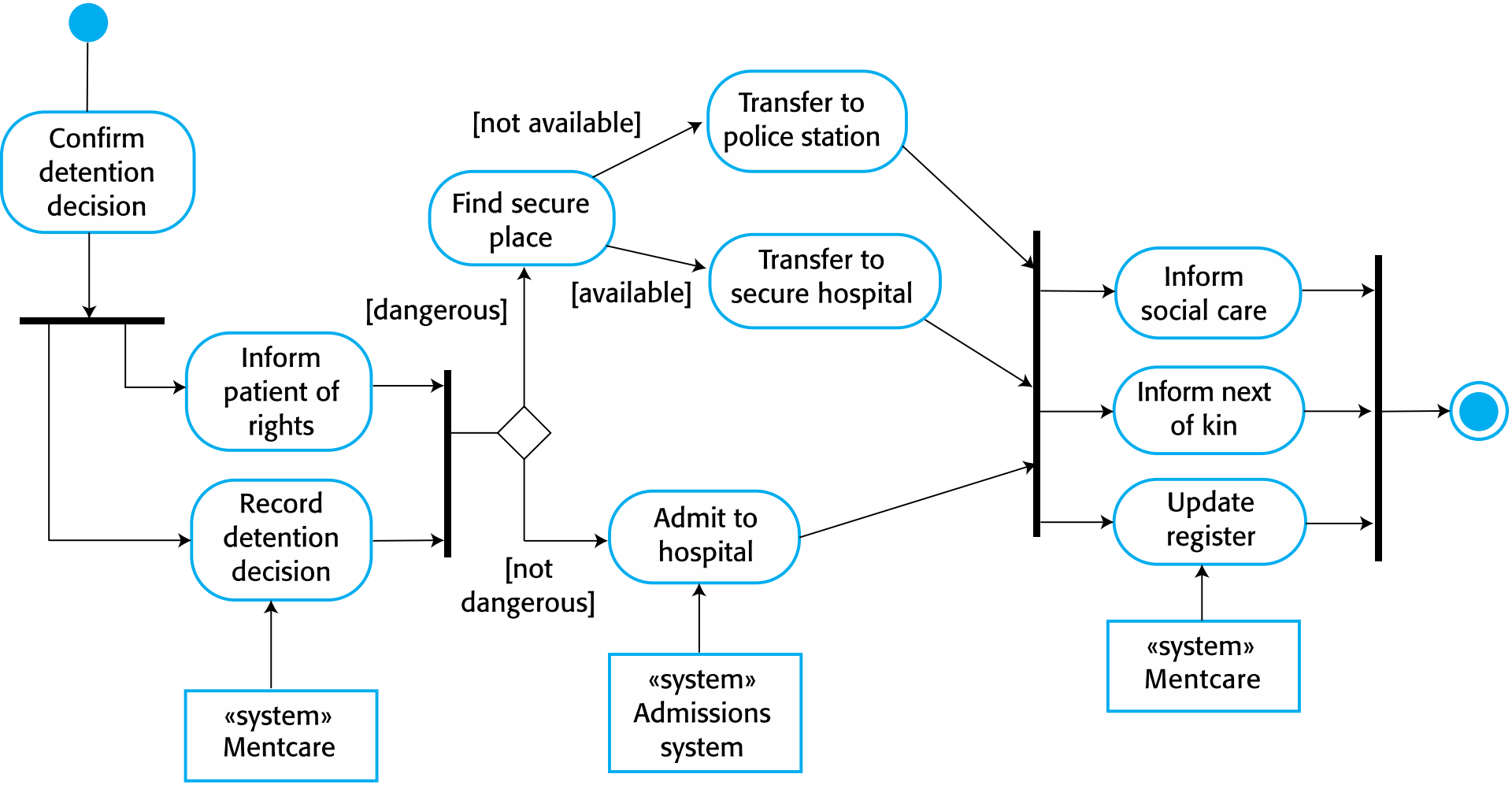
Popular UML Diagrams

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- State Machine Diagram

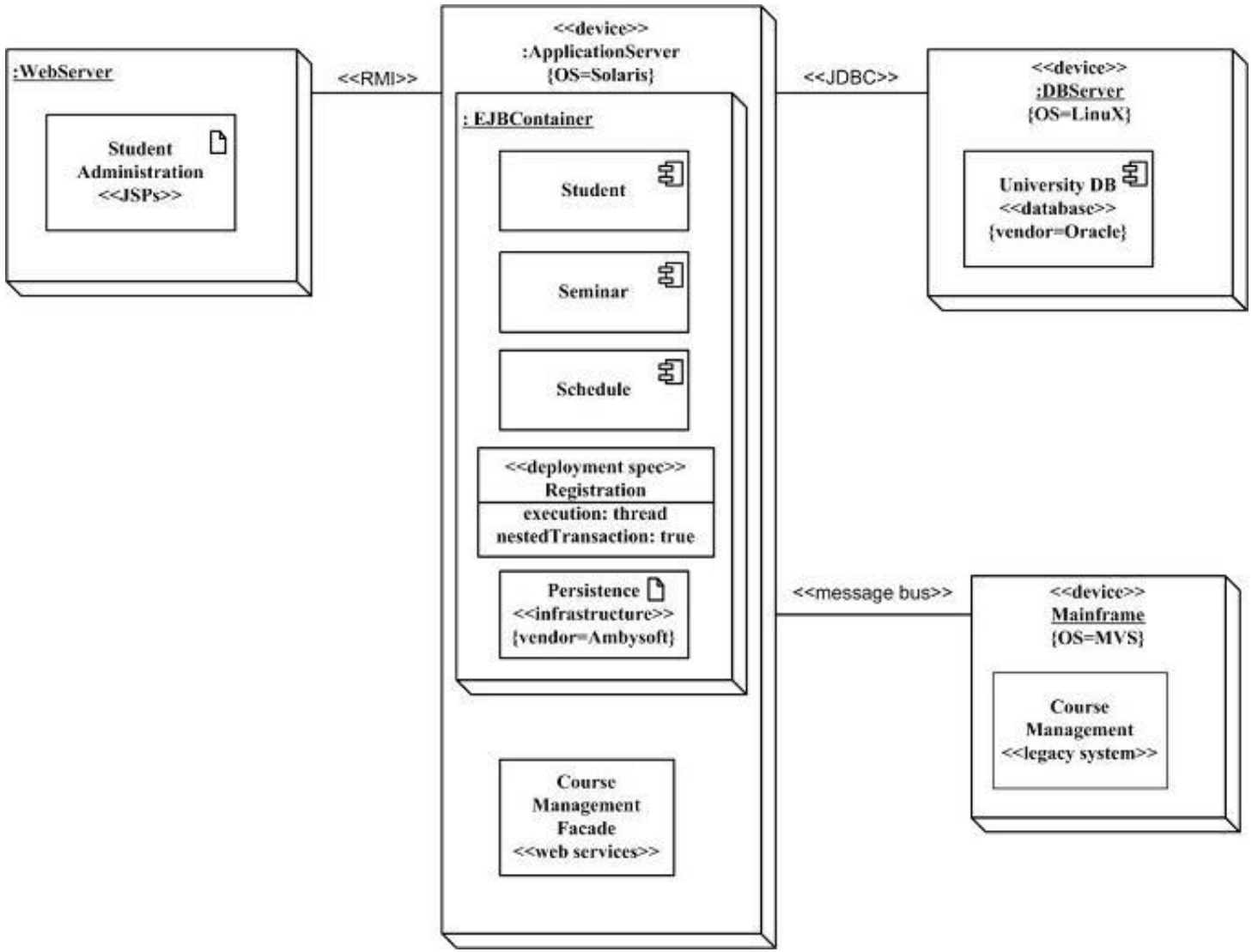
Other UML Diagrams

- Activity Diagram
- Deployment Diagram
- Object Diagram
- Package Diagram
- Component Diagram
- Profile Diagram
- Communication Diagram
- Timing Diagram
- Composite Structure Diagram
- Interaction Overview Diagram

UML Activity Diagram Example
(Process model of involuntary detention)



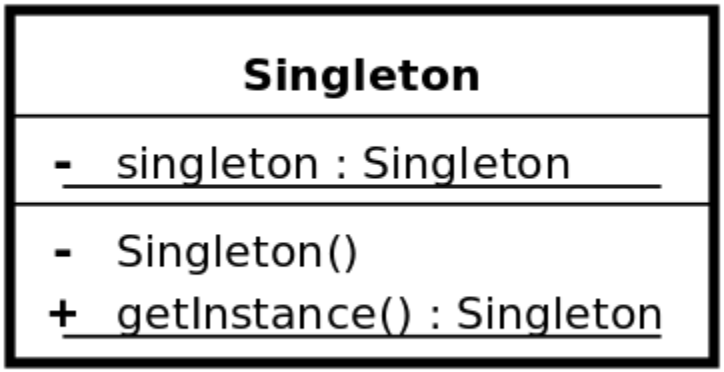
UML Deployment Diagram Example
(University Information System)



http://www.agilemodeling.com/artifacts/deploymentDiagram.htm

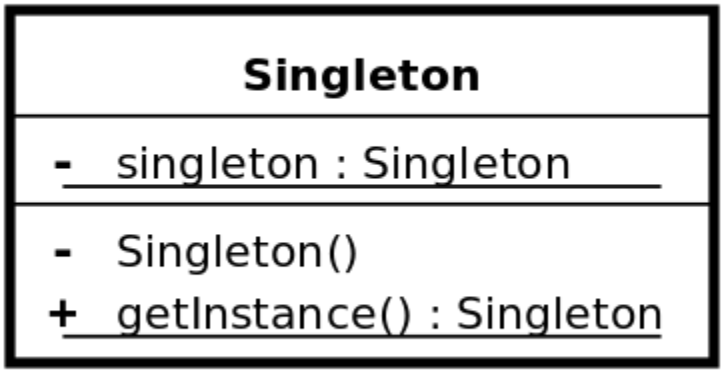
Example Design Patterns

Singleton Design Patterns



https://en.wikipedia.org/wiki/Singleton_pattern

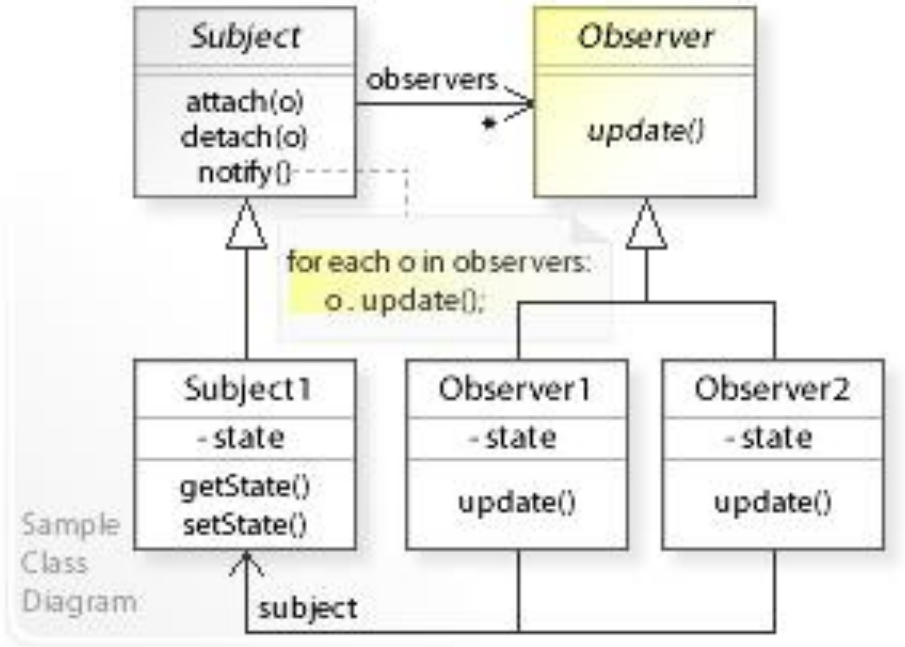
Singleton Design Patterns

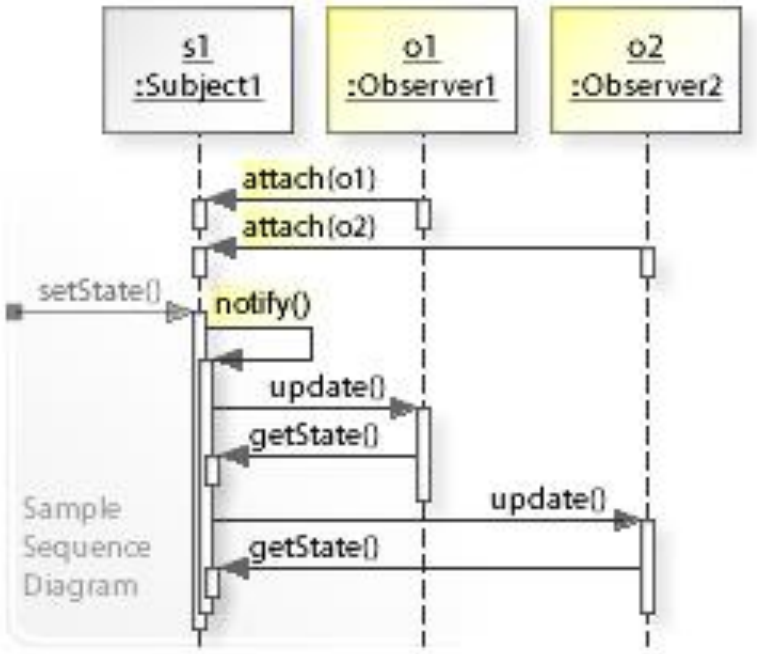



```
public final class Singleton {
    private static final Singleton INSTANCE = new Singleton();

    private Singleton() {}

    public static Singleton getInstance() {
        return INSTANCE;
    }
}
```

https://en.wikipedia.org/wiki/Singleton_pattern

Observer Design Pattern

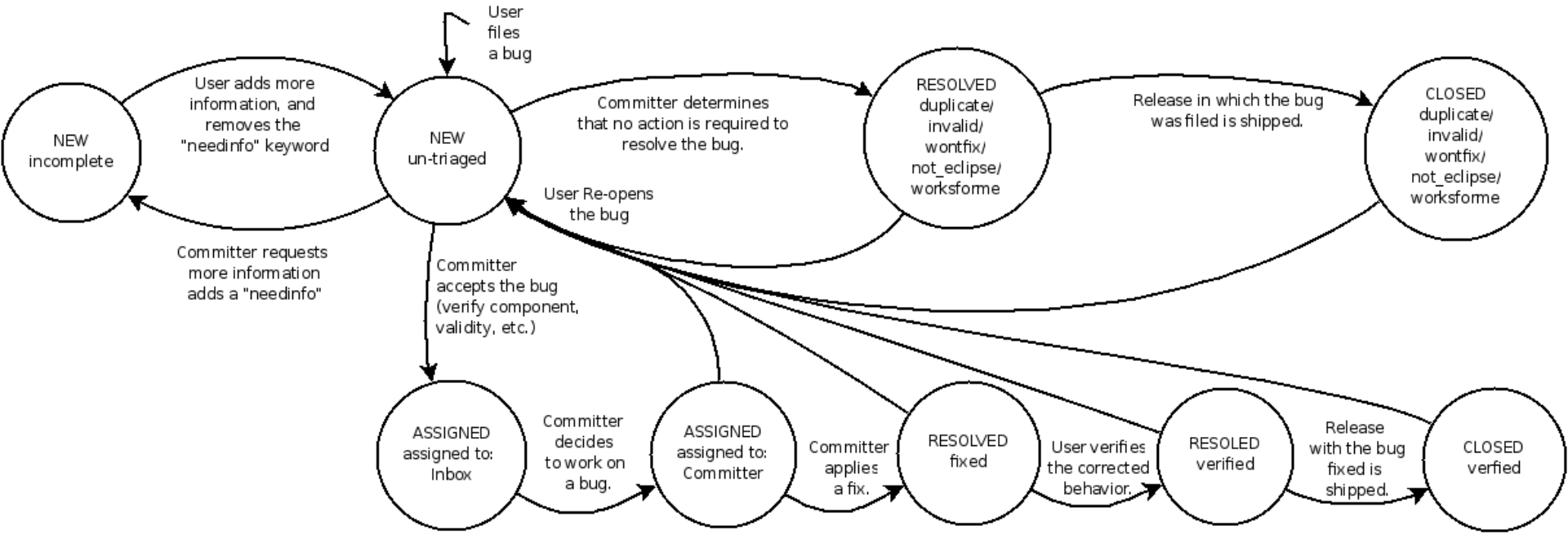




https://en.wikipedia.org/wiki/Observer_pattern

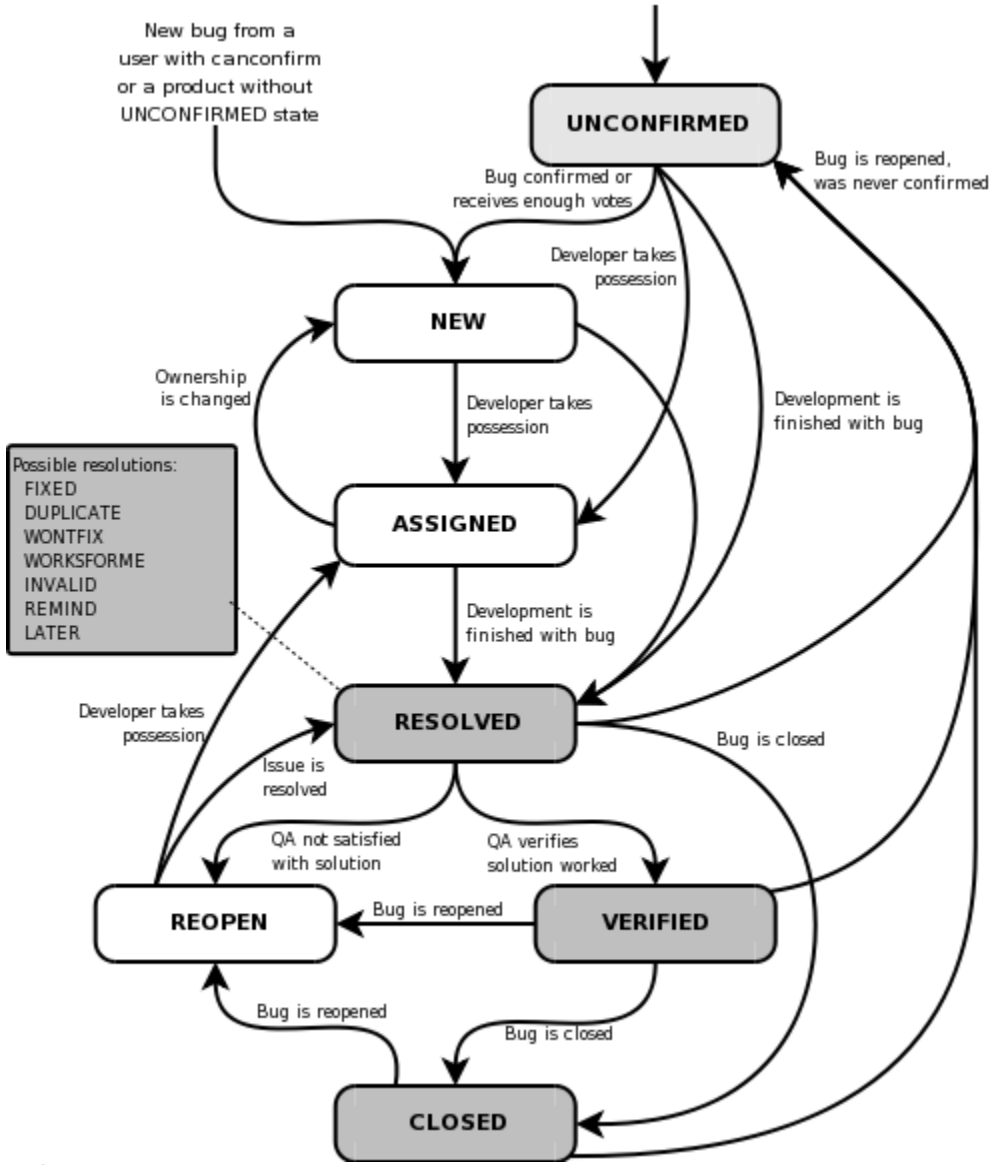
How to model the status of a bug report?

Eclipse Issue (e.g., bug) Tracking Lifecycle



https://wiki.eclipse.org/CDT/Bugs

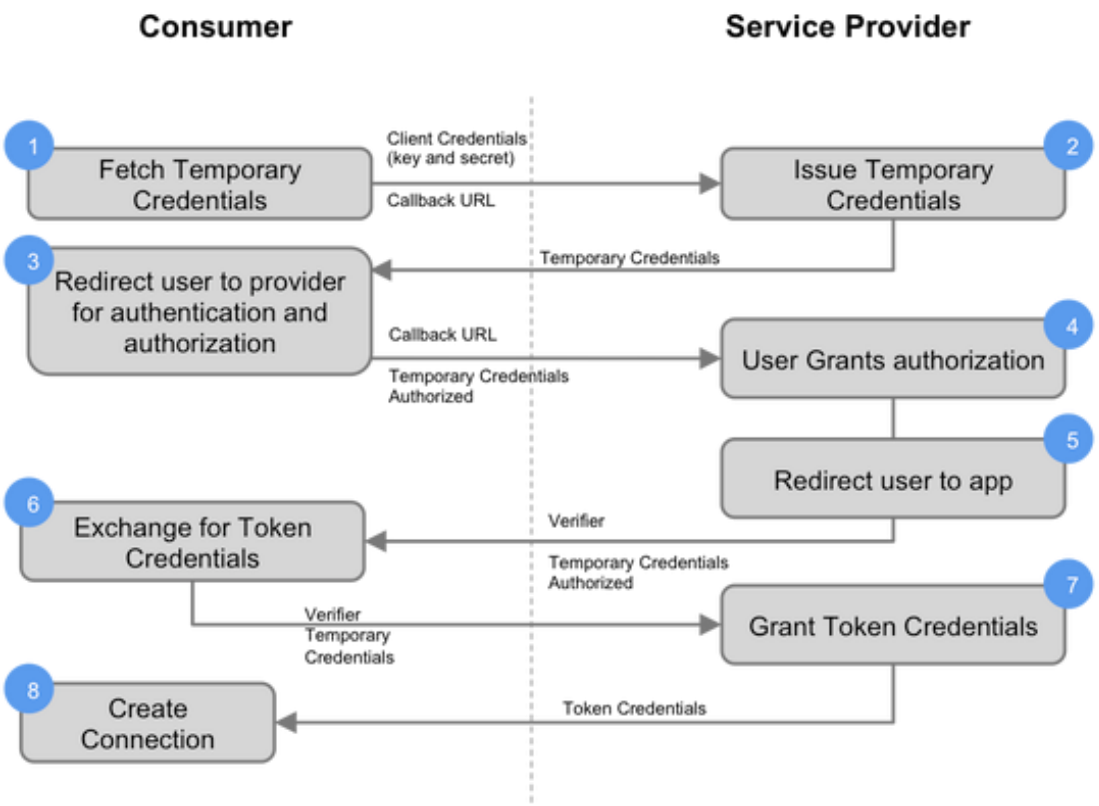
Bugzilla – Life Cycle of a Bug



https://www.bugzilla.org/docs/2.18/html/lifecycle.html

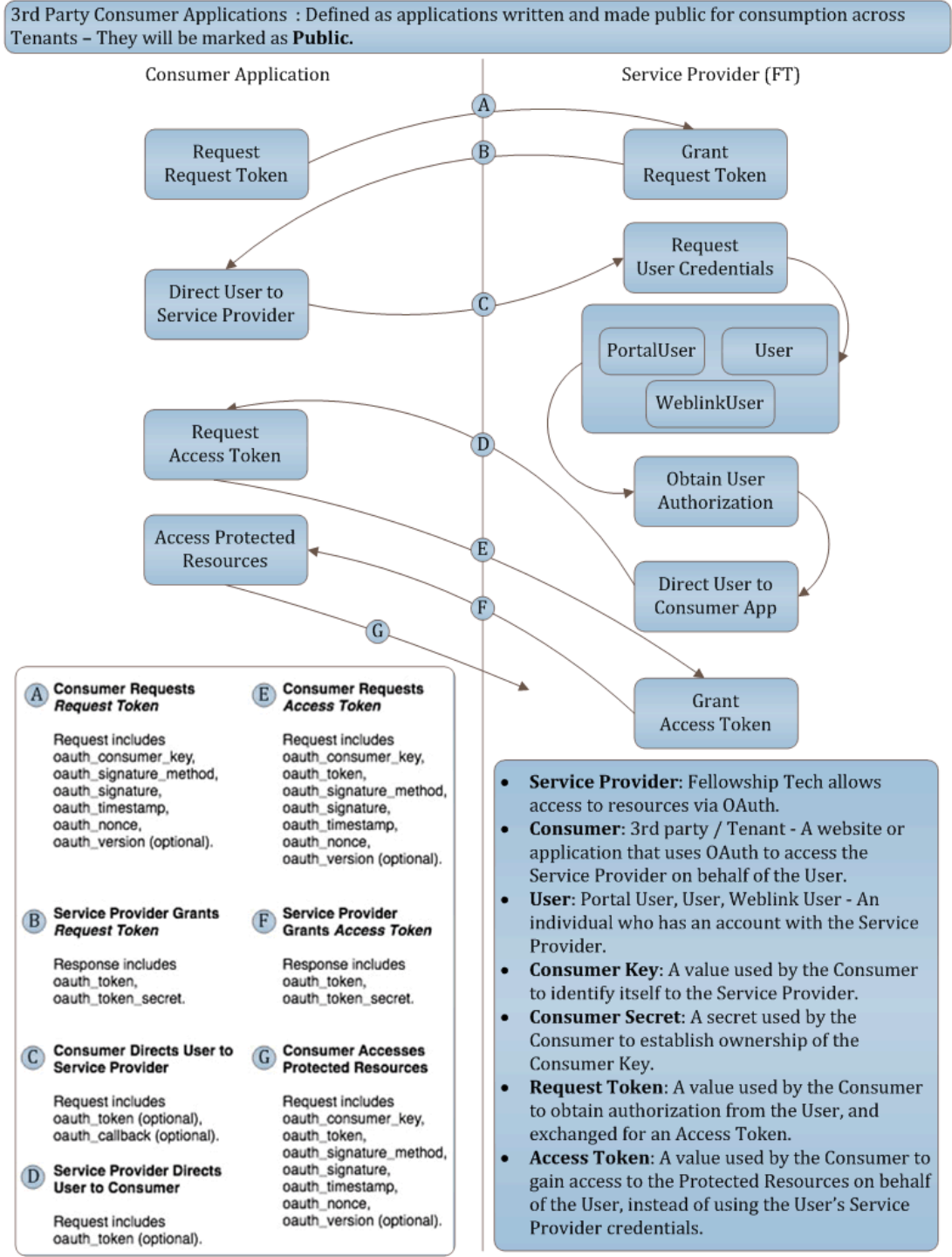
Examples of "UML"-like diagram in the "real-world"

OAuth 1.0



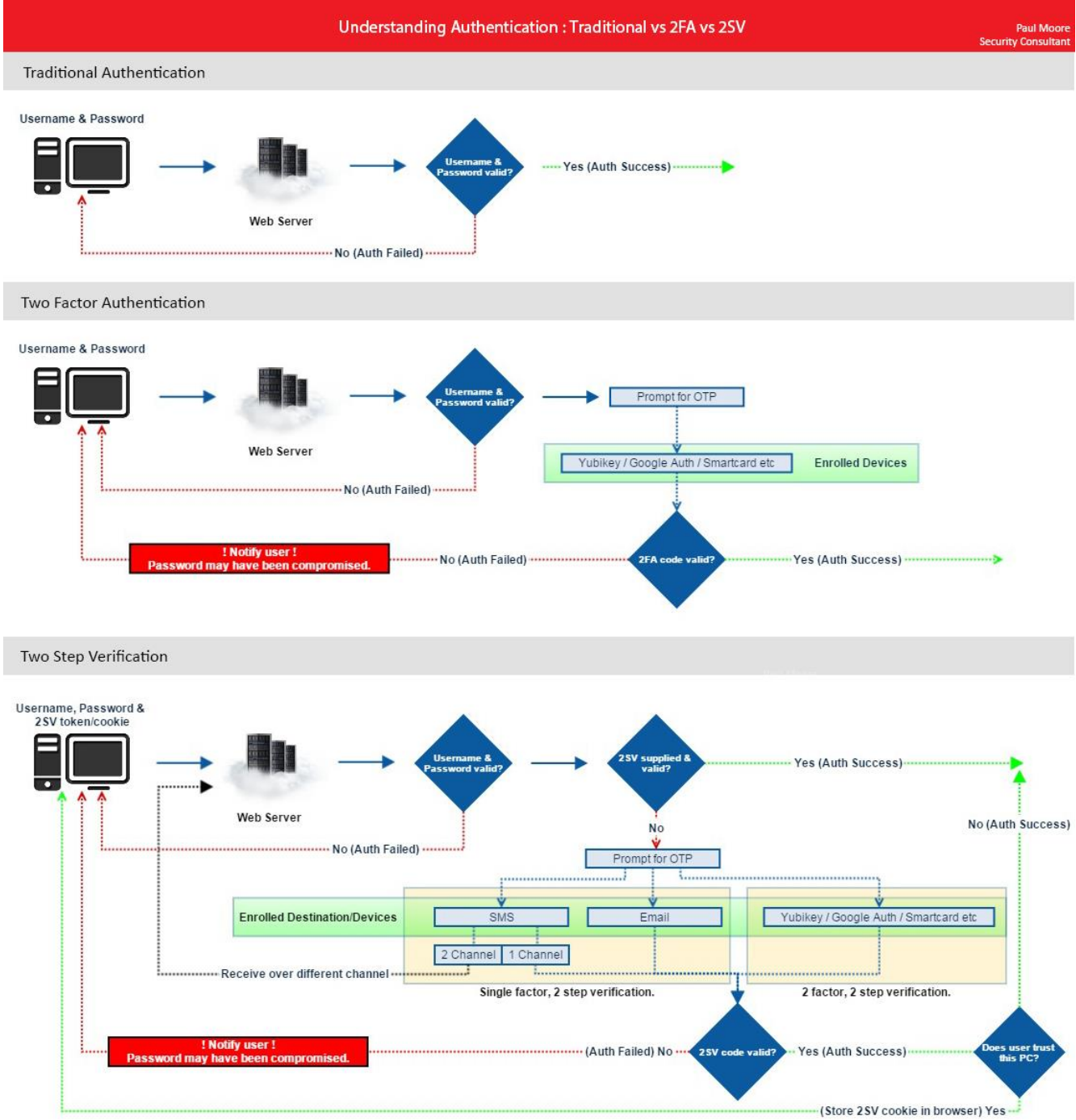
https://deepakpol.wordpress.com/2015/07/16/oauthorization/

Examples of "UML"-like diagram in the "real-world"



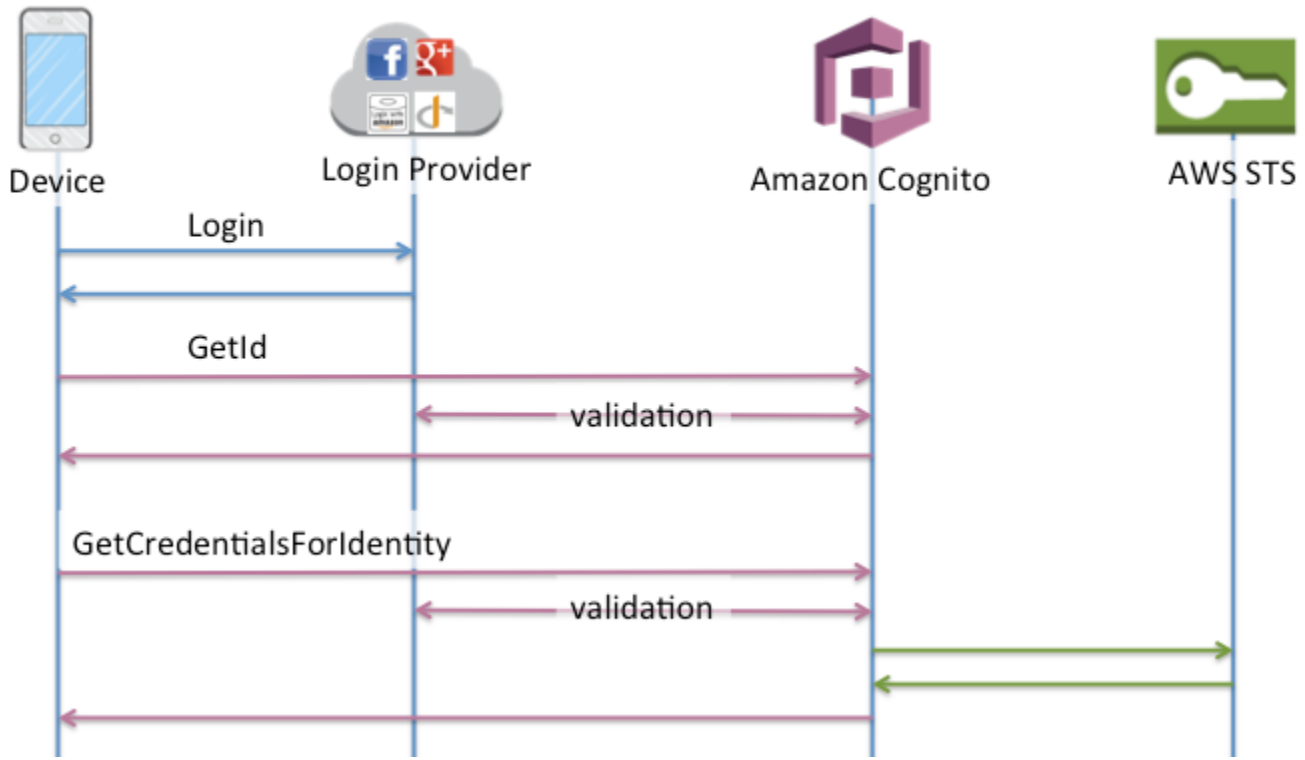
https://developer.fellowshipone.com/docs/v1/Util/AuthDocs.help

Examples of "UML"-like diagram in the "real-world"



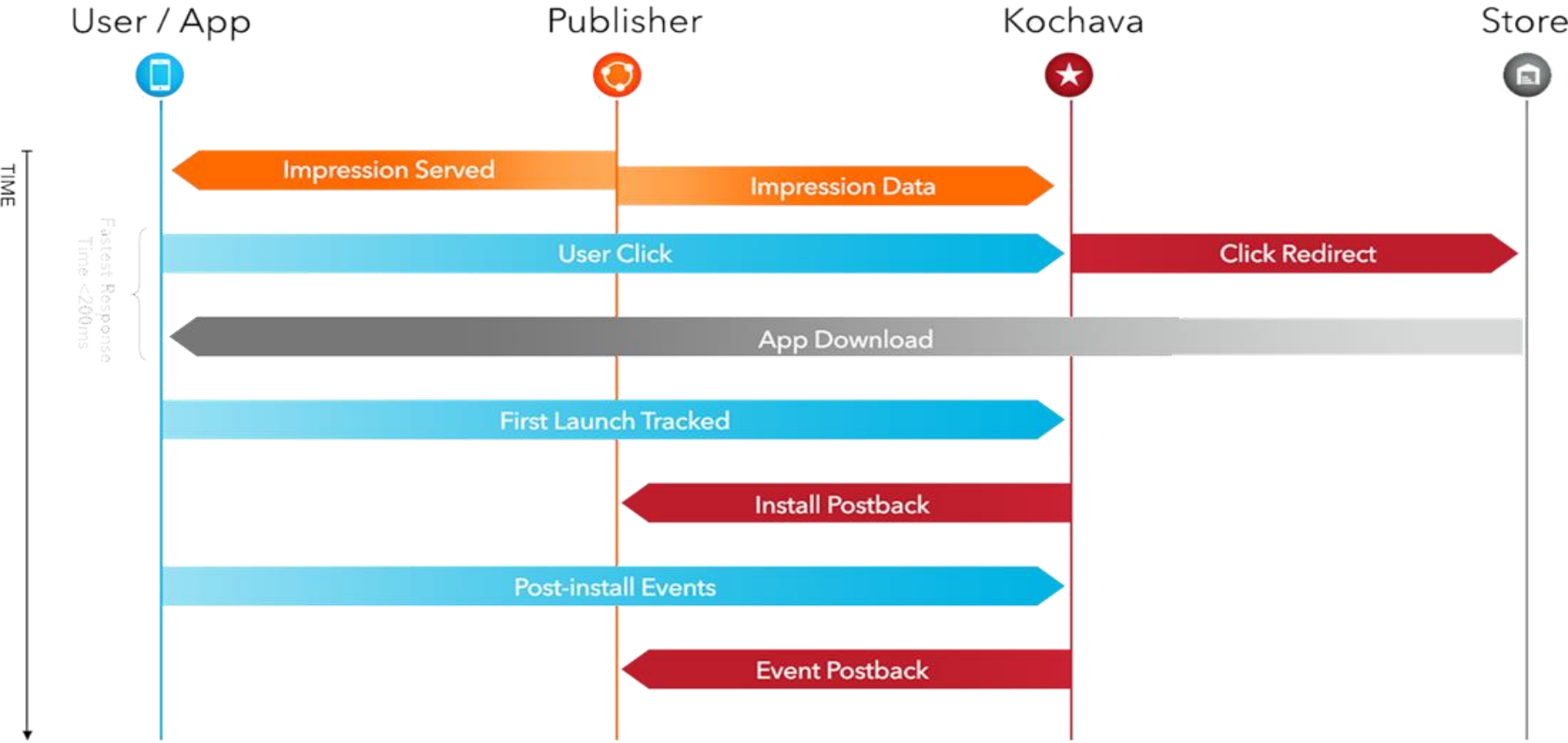
https://paul.reviews/the-difference-between-two-factor-and-two-step-authentication/

Examples of "UML"-like diagram in the "real-world"



http://docs.aws.amazon.com/cognito/latest/developerguide/authentication-flow.html

Examples of "UML"-like diagram in the "real-world"



State Machines on any Ruby Class

```ruby
class Vehicle
  attr_accessor :seatbelt_on, :time_used, :auto_shop_busy

  state_machine :state, :initial => :parked do
    before_transition :parked => any - :parked, :do => :put_on_seatbelt

    after_transition :on => :crash, :do => :tow
    after_transition :on => :repair, :do => :fix
    after_transition any => :parked do |vehicle, transition|
      vehicle.seatbelt_on = false
    end

    after_failure :on => :ignite, :do => :log_start_failure

    around_transition do |vehicle, transition, block|
      start = Time.now
      block.call
      vehicle.time_used += Time.now - start
    end

    event :park do
      transition [:idling, :first_gear] => :parked
    end

    event :ignite do
      transition :stalled => same, :parked => :idling
    end

    event :idle do
      transition :first_gear => :idling
    end
```

- https://github.com/pluginaweek/state_machine