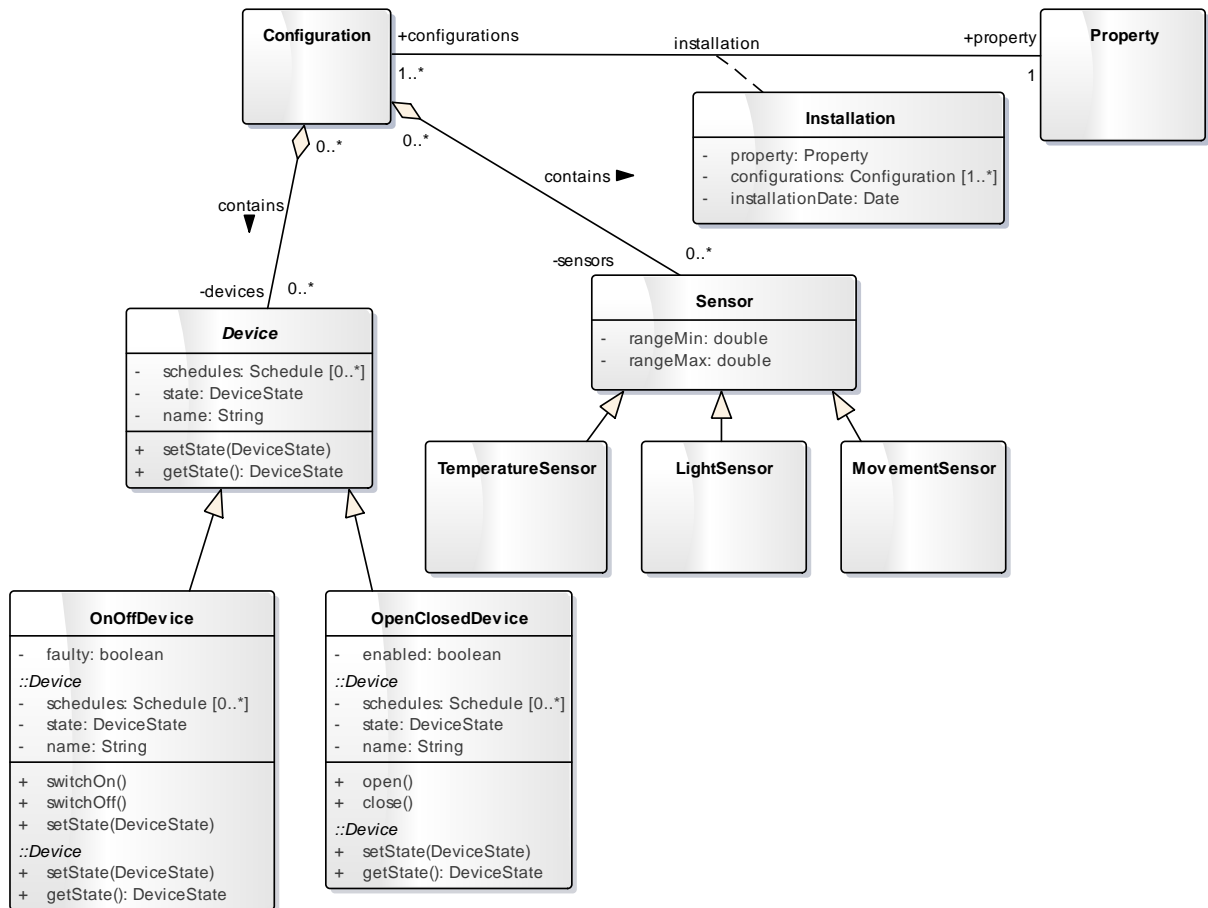# Learning UML

O'Reilly

Exercises – Interfaces and Enumerations

If you have followed along, your *home network* diagram should look similar to the one in the movie. If not, first create a diagram that looks like this.

Now do the following:

Make some space in the diagram by moving *Sensor* and its subclasses down to the right.

Add a new interface called *Switchable*. Add the operations *enable()* and *disable()*, both with no parameters and no return type.

Add realization relationships between *Sensor* and *Switchable* and between *Device* and *Switchable*. When you do so, select the option to add the operations to the two classes.

Set the class *Sensor* to be an abstract class.

Add the enumeration *DeviceState* to the diagram with the values *On*, *Off*, *Open* and *Closed*.

Your *home network* diagram should now have these elements in it:
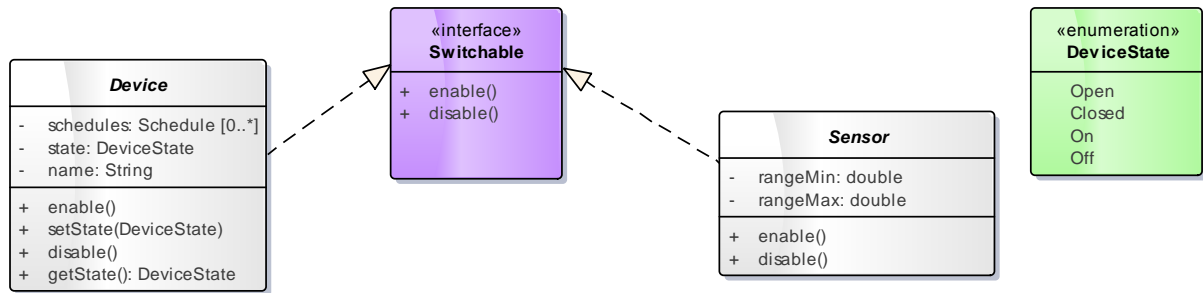
Go to the *events* diagram.

Your diagram should look something like this.
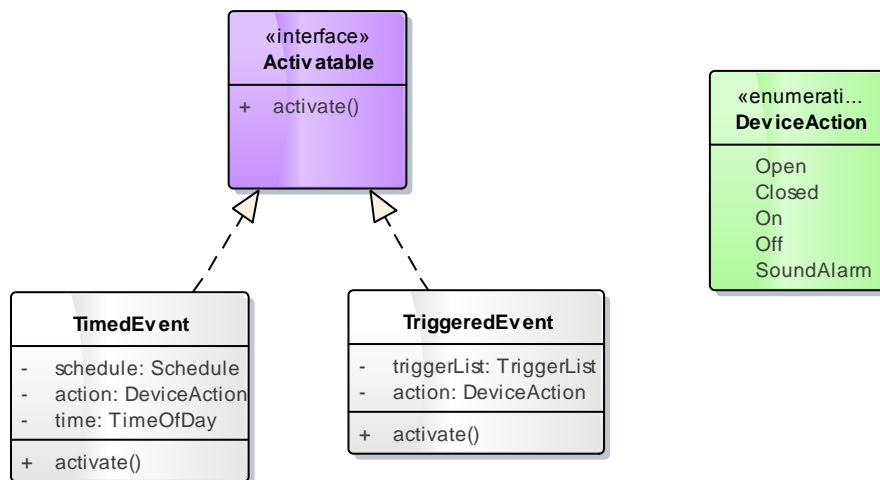
Edit the values *Closed, On* and *Off* to be *Close*, *SwitchOn* and *SwitchOff*. This way these are all actions and distinct from the states in *DeviceState*.

Your diagram should now look like this.

Go to the *domain* diagram, which should look like this.

Schedule

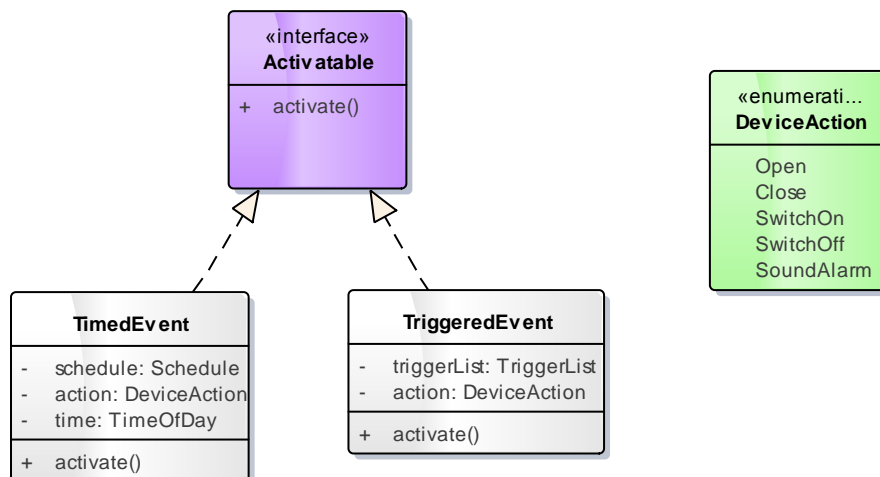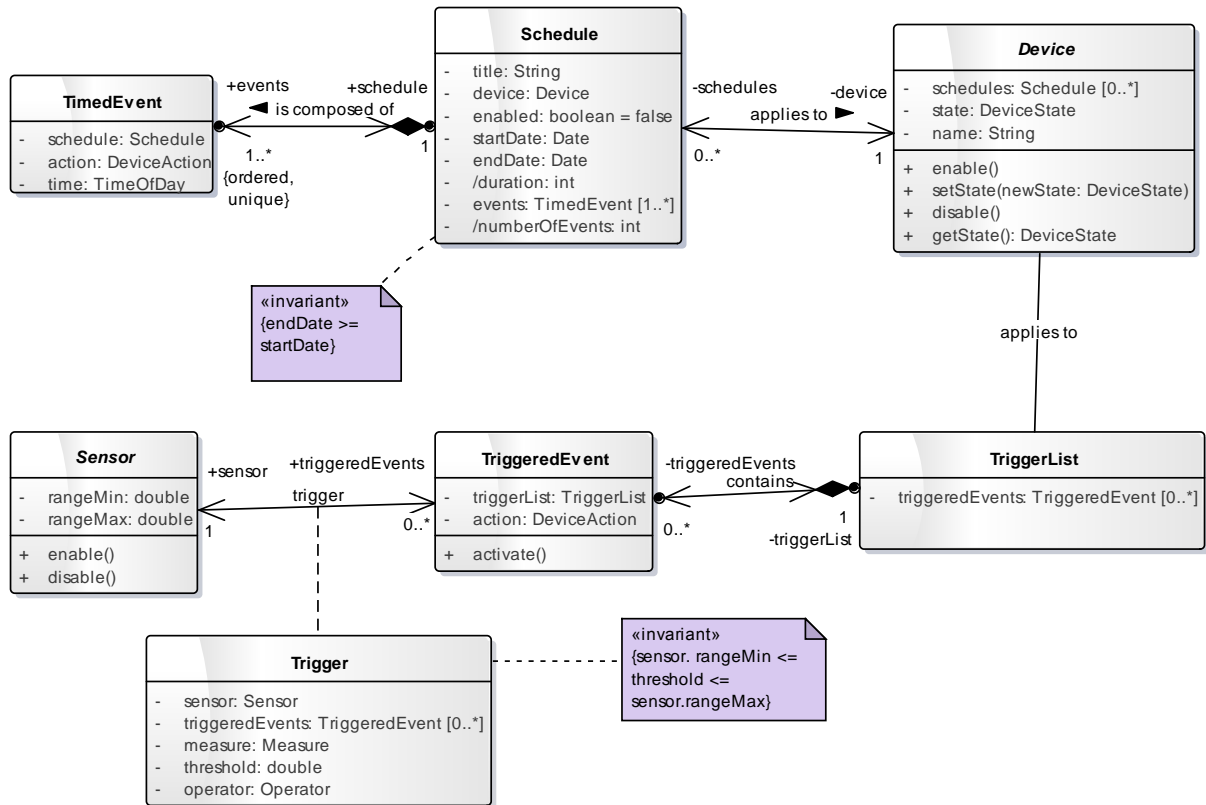| Schedule |
|---|
| - title: String |
| - device: Device |
| - enabled: boolean = false |
| - startDate: Date |
| - endDate: Date |
| - /duration: int |
| - events: TimedEvent [1..*] |
| - /numberOfEvents: int |

TimedEvent

TimedEvent
- schedule: Schedule
- action: DeviceAction
- time: TimeOfDay

+events +schedule

◄ is composed of

1..*
{ordered, unique}

1

Device

Device
- schedules: Schedule [0..*]
- state: DeviceState
- name: String
+ enable()
+ setState(newState: DeviceState)
+ disable()
+ getState(): DeviceState

-schedules -device

applies to ▶

0..*   1

«invariant»
{endDate >= startDate}

applies to

Sensor

Sensor
- rangeMin: double
- rangeMax: double
+ enable()
+ disable()

+sensor +triggeredEvents

trigger

1   0..*

TriggeredEvent

TriggeredEvent
- triggerList: TriggerList
- action: DeviceAction
+ activate()

-triggeredEvents
contains

1   0..*
-triggerList

TriggerList

TriggerList
- triggeredEvents: TriggeredEvent [0..*]

Trigger

Trigger
- sensor: Sensor
- triggeredEvents: TriggeredEvent [0..*]
- measure: Measure
- threshold: double
- operator: Operator

«invariant»
{sensor. rangeMin <= threshold <= sensor.rangeMax}

Add the enumeration *Operator* with the values =, >, <, >=, <=. Your enumeration should look like this.

«enumeration» **Operator**
=
>
<
>=
<=