# Behavior-based Model

State Machine Diagram

# How much is enough?

- We have done Use Case model (Scenario-based model)
- We have completed Class diagram (Class-based model)

- Aren’t those requirement modeling representations enough?

# Behavioral Model

- dynamic behavior of the system
- function of specific events and time; indicates how software will respond to external events or stimuli

# Creating a Behavioral Model

1. evaluate all use cases to fully understand the sequence of interaction within the system,
2. identify events that drive the interaction sequence and understand how these events relate to specific objects,
3. build a state machine diagram for the system,
4. review the behavioral model to verify accuracy and consistency.

# Identifying Events

- Scenarios in use case description represent sequence of activities that involves actors and the system
- The actor and the system exchange information in the sequence of activities
- The exchange of information is an event.
- Some event have an explicit impact on the flow of control.

# Identifying Events

*The homeowner uses the keypad to key in a four-digit password. The password is compared with the valid password stored in the system. If the password is incorrect, the control panel will beep once and reset itself for additional input. If the password is correct, the control panel awaits further action.*

# Event / Stimulus

- A discrete signal that happens at a point in time
- Hardware Interrupt? Message Calls?
- Caused by communication between classes within or external
- May cause a change in state
- May trigger actions
- May have associated conditions

# State of a Class

- A *passive state* is simply the current status of all of an object's attributes
  - For example: Attributes of Student class
    - Student ID, name, enrolled date,…..
- The *active state* of an object indicates the current status of the object as it undergoes a continuing transformation or processing.
  - For example: Status of Student class
    - New, Enrolled, Suspended, Graduated

# Event and State

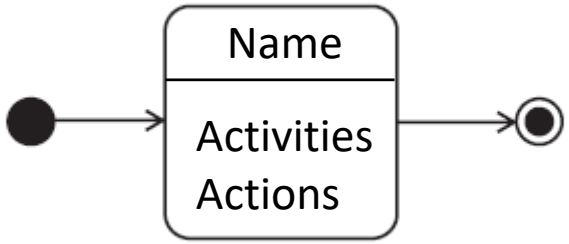
- An *event* (sometimes called a *trigger* ) must occur to force an object to make a transition from one active state to another.

# State Machine Diagram

- Objects in the system change their state in response to events
  - When you turn the ignition button, your car's engine change from stop to start
  - After certain amount of time, the toaster eject your bread

- UML state machine diagram documents these kinds of changes.
  - It presents the states an object can be in along with the transitions between the states
  - It shows the starting point and endpoint of a sequence of state change

# Notation



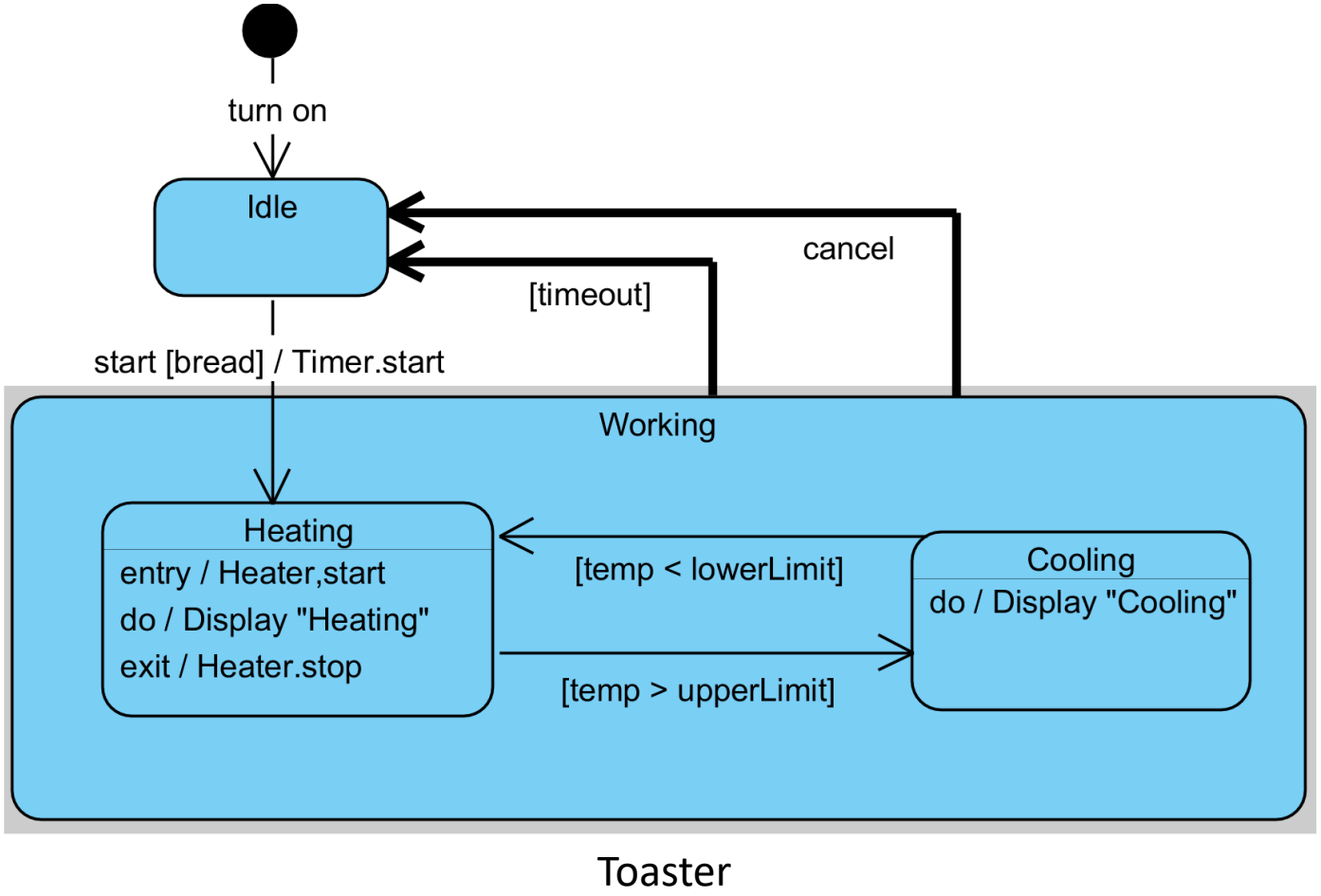

- Start and Terminate
- State
- Transition
- Action and activities

# Start / Terminate State

- Start State: Creation of object
- Terminate State: Destruction of object

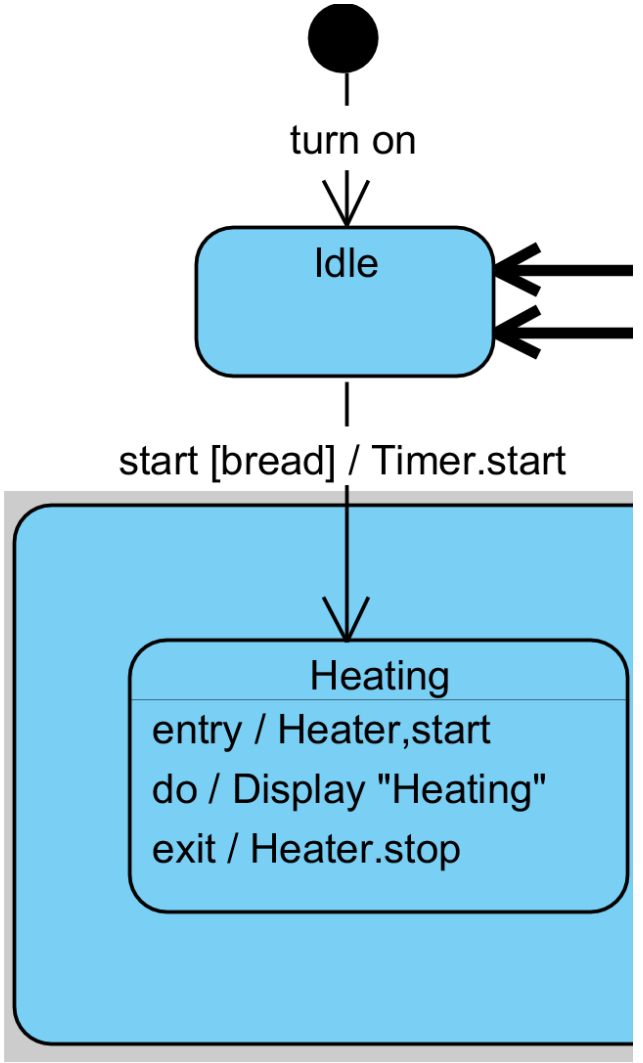
# State

- Each class has one or more states
- Class behave differently depending on each state
- Class can change from one state to the next through Transition
- Each state must have incoming and outgoing transitions

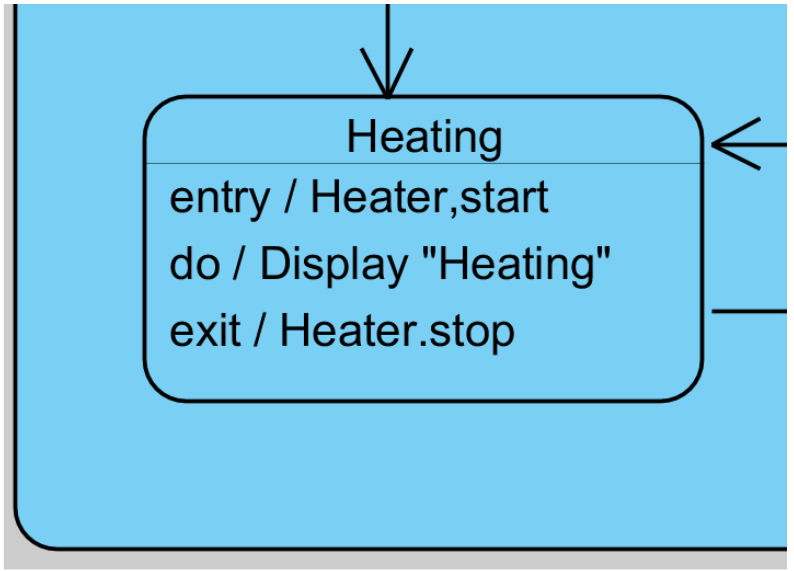


# The transition

- Change from one state to the other
- What caused transition?
  - Event
  - Condition met during an event
- Actions may be executed during transition



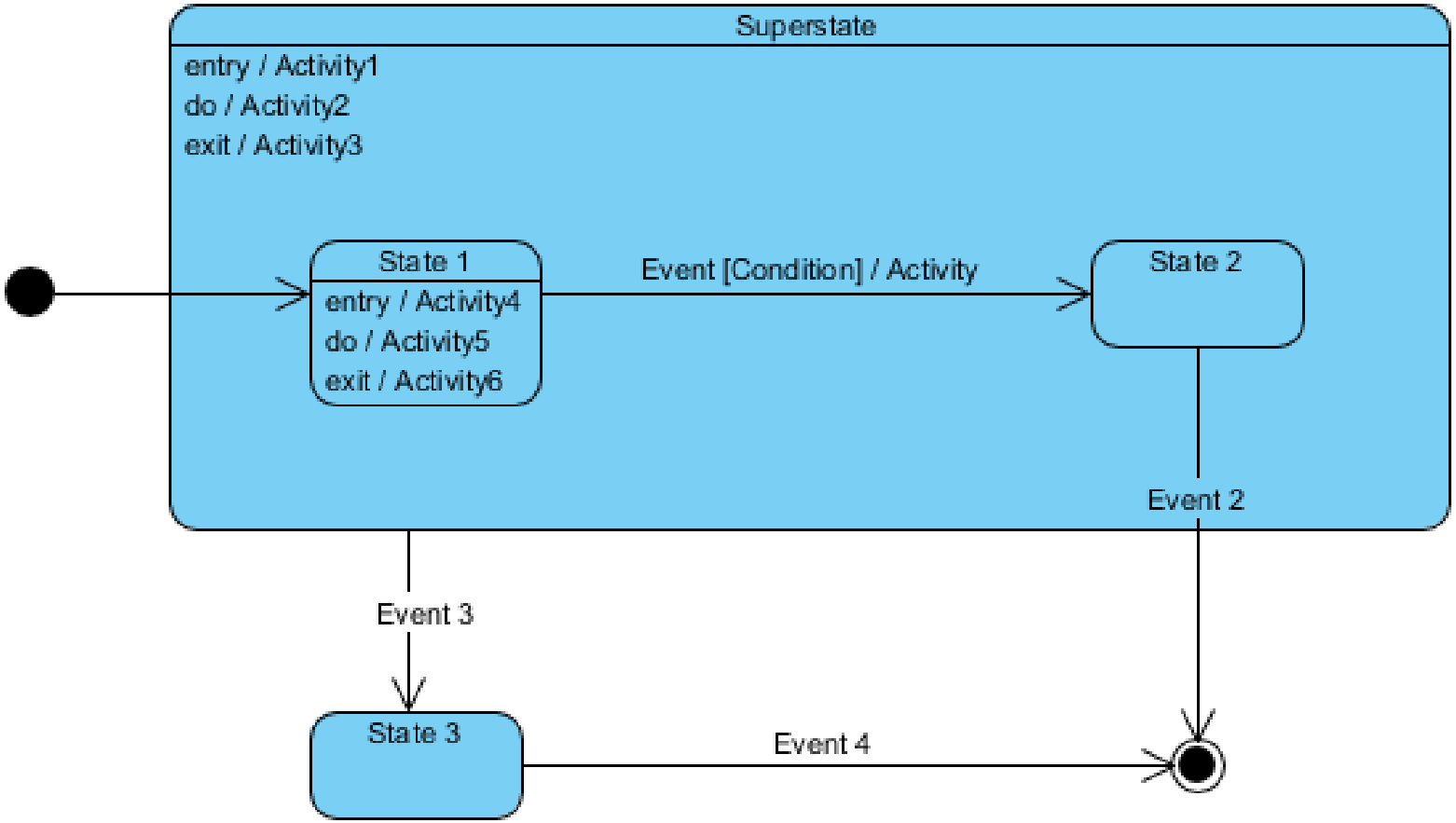
# Action and activity




- 3 categories of activities
  - entry / action
  - do / activity
  - exit / action
- entry/action: discrete action, perform only once at the entrance to the state
- do/activity: continuous activity throughout the state live span
- exit/action: discrete action, perform only once at the exit of the state

# State Hierarchy

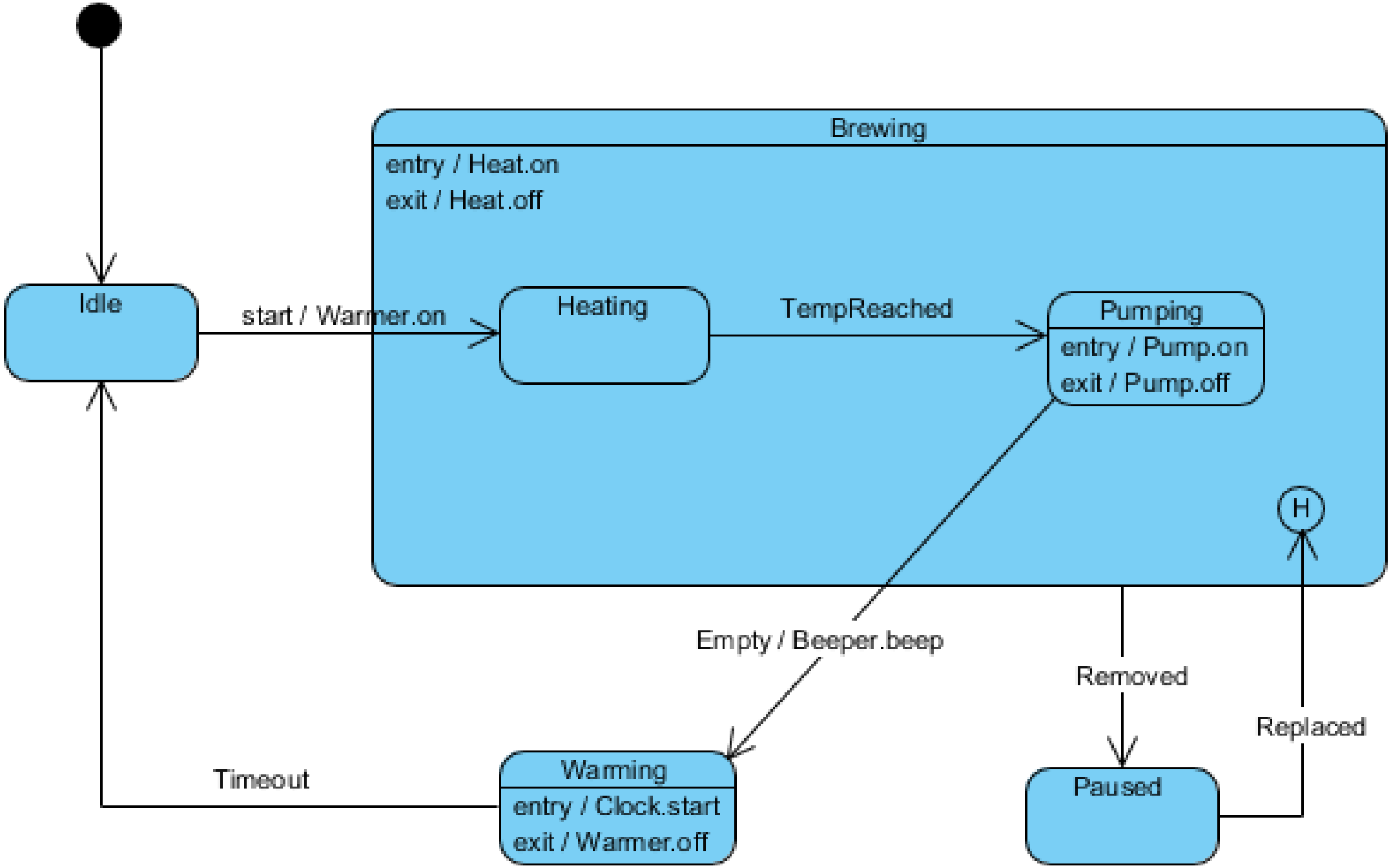
- Superstate / Substate



# Example: Modeling an automatic coffee maker


- Person fills with coffee grounds & water filled
- Person puts carafe in place
- Person presses the Start button
- System starts brewing coffee
  - Heating element and carafe warmer turned on
  - When brewing temp reached, pump is started
  - Stop pump, heating element, and sound beeper when water level is empty
- Automated timer begins when brewing is complete
- When timer elapses, turn off carafe warmer.

Source from: Rob Pettit, 2005


Brewing
entry / Heat.on
exit / Heat.off
Idle
start / Warmer.on
Heating
TempReached
Pumping
entry / Pump.on
exit / Pump.off
H
Empty / Beeper.beep
Removed
Replaced
Timeout
Warming
entry / Clock.start
exit / Warmer.off
Paused