Worksheet 7: State

Updated: 21st October, 2014

1. Discussion: Simple State Diagrams

For each of the following cases, discuss how you would model the states. What features of the software are affected? What would the state diagram look like?

(a) "Saved" and "unsaved" documents in an office-type application (i.e. word processor, spreadsheet tool, drawing tool, etc.).

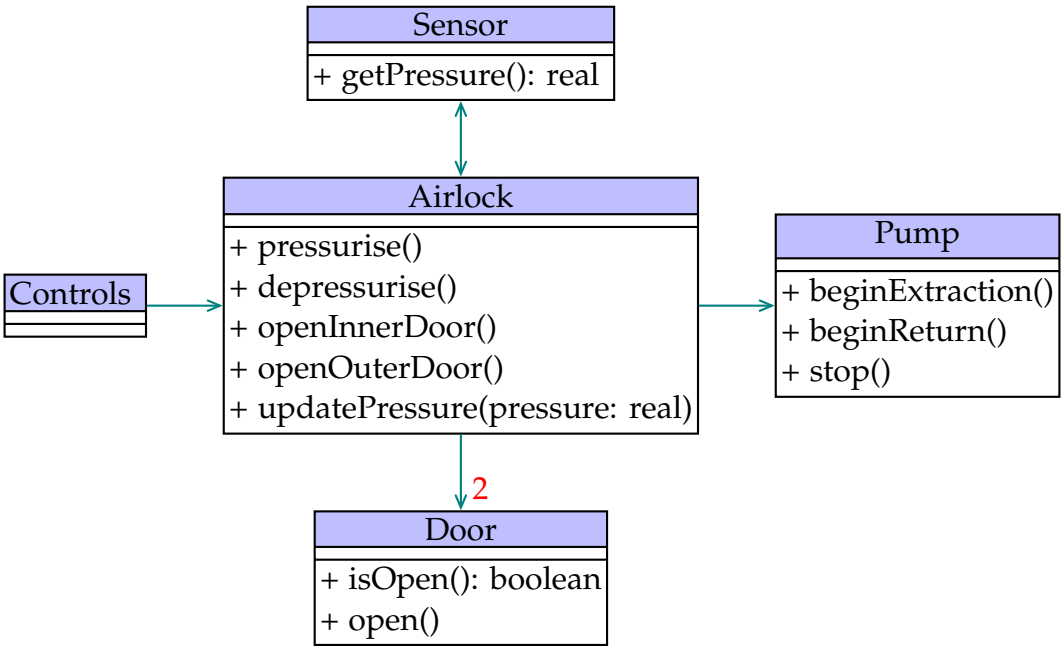
(b) A controller class for a blogging web application that manages different sets of user rights, depending on whether the user is "logged in" or not.

(c) A user interface with the ability to show different sets of controls: "simple", "intermediate" and "advanced" (as requested by the user).

There are a number of aspects of the software's functionality that may be important. See if you can think of them.

2. State Diagrams in Detail

Draw a UML statechart for the Airlock class in the following system, which manages an airlock in a space station.



To explain each class briefly:

Sensor — measures the pressure inside the airlock on demand. In addition, it calls updatePressure() every second (from its own thread).

Door — represents one of the two airlock doors (inner or outer). It can open the door, and retrieve the open status. Each door is closed manually by the astronaut.

Pump — controls the pump that extracts and returns the air from the airlock. It does one operation or the other until told to stop.

Controls — a simple user interface, consisting of four buttons. (Physically, these buttons are duplicated in three places: inside the airlock, outside the outer door, and outside the inner door.)

Airlock — the controller, to keep track of the airlock state in a safe manner.

At all times, at least one of the airlock doors must be closed. Otherwise the whole station would depressurise, and everyone would have a rather bad day.

The pressure inside the airlock is expected to be in the range 0 to about 110 kPa (kilopascals), since negative pressure is physically impossible [1], and 101.3 kPa is considered "standard" pressure. For our (simplified) purposes, anything above 90 kPa is considered acceptable for humans to breathe.

When an astronaut needs to perform maintenance to the exterior of the space station, they open the inner door of the airlock, float inside and close the door. Then the air is pumped out of the airlock and stored in a tank (so that we don't lose it). This takes a little while to happen. Then they open the outer door, exit, and close it behind them. When they need to return, the process happens in reverse (with the air being pumped back in).

The pumping process (either pressurising or depressurising the airlock) can be halted and reversed if needed.

The four control buttons result in the following methods being called:

- pressurise() begins pumping air back into the airlock.
- depressurise() begins pumping air out of the airlock.
- openInnerDoor() and openOuterDoor() are self-explanatory.

Each of these can only happen when it is safe. The airlock pressure must be less than 5 kPa before the outer door can be opened, and above 90 kPa before the inner door can be opened.

3. Basic State Implementation

Based on your state diagram, implement the Airlock class (in any language). Write it as a single self-contained class for now.

Note: You will need some sort of field (or fields) to keep track of which state you're in. Exactly how you do this is largely up to you, as long as the value(s) uniquely identify each possible state.

[1] Except during the first 10^{-36} seconds of the universe, when some unusual physics happened.

4. State Pattern

Based on your work so far, consider how to apply the State Pattern to this problem. Specifically, do the following:

(a) Extend the class diagram shown above to incorporate the State Pattern. (Hint: You'll need to keep all the current classes and relationships, and add some more, plus an interface.)

(b) Re-implement the Airlock class, and the other classes you've just proposed, in accordance with the State Pattern.

Note: Using the state pattern, you should find that there are fewer conditional (`if`/`switch`) statements than in the non-State Pattern implementation.

End of Worksheet