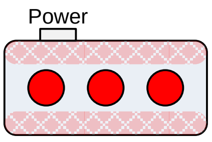
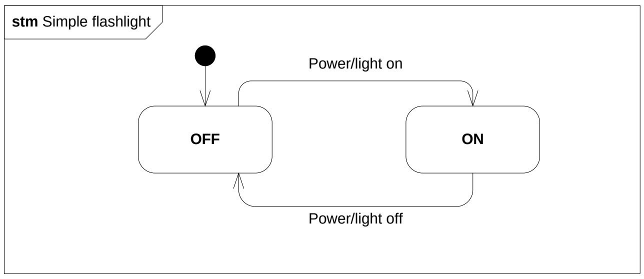
Exercise: Introduction to state patterns



In this exercise you will design and implement a simple state machine for a 3-LED bicycle flashlight which will gradually evolve to become more complex. Through this, you will learn how the UML/SysML state machine diagrams and the implementations are related.

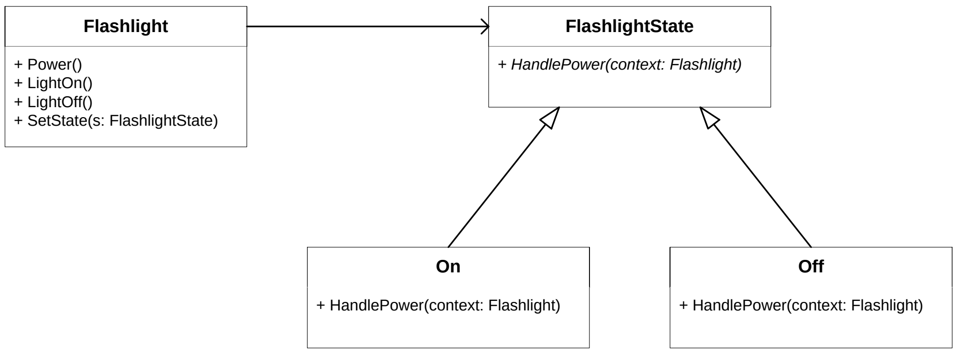
Exercise 1: Basic flashlight with switch/case

Implement the control mechanism of a basic version of 3-LED flashlight along the lines of the below state machine diagram. The implementation shall be done using the **switch/case** approach. Ensure that the names of the states, events and actions are reflected in your implementation. Run your state machine.



Exercise 2: Basic flashlight with GoF State

Create a new VS project under the same solution as above. In this project, implement the same sate machine as above, only this time using the **GoF State Pattern** along the lines given below. Ensure that you understand how the states map to classes in the pattern. Run your state machine.


Flashlight
+ Power()
+ LightOn()
+ LightOff()
+ SetState(s: FlashlightState)
FlashlightState
+ HandlePower(context: Flashlight)
On
+ HandlePower(context: Flashlight)
Off
+ HandlePower(context: Flashlight)

Exercise 3: Understand the call sequence in GoF State.

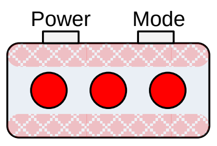
Assume your Flashlight is in the OFF state and receives a Power event. Draw a UML sequence diagram showing all calls back and forth between the Flashlight and FlashlightState objects.

Exercise 4: Compare implementation

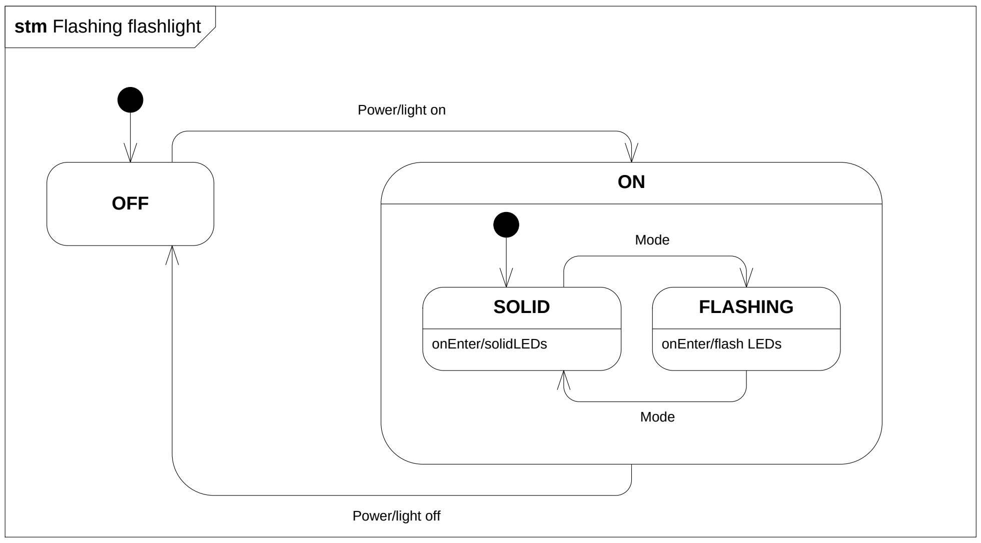
Compare the two implementations, the switch/case and the GoF State Pattern implementations. Which one do you prefer? Why? What do you think of the solutions in terms of extensibility, maintainability, and testability?

Exercise 5: Extend your implementations

Business is good, and a version 2 of the flashlight is designed. In the version, a Mode button is added. When the flashlight is on, pressing the Mode button will cause the flashlight to toggle between solid light and flashing light – yes, a very innovative and never-seen-before approach to bicycle flashlights! The design of the control mechanism is given below.



Extend both your switch/case and GoF State Pattern implementations to handle the new requirements. For the GoF State Pattern-implementation, be sure to draw your class diagram first and be sure you handle OnEnter correctly – if you do, the code becomes *very* elegant!



Exercise 6: Compare (again)

Again compare the two implementations, the switch/case and the GoF State Pattern implementations. Which one do you prefer now? Why? How easy is it to map the State Machine Diagram, including event handling, to the individual implementations now?

Exercise 7: Adhering to ISP

Right now, the Flashlight class has a lot of methods. Some are meant as event handlers, some are meant for the state objects so that they can cause the LEDs to turn on, off, and toggle. At the moment, the class violates ISP. Refactor the GoF State Pattern implementation so that this is fixed (hint: separate the interfaces needed for UI handling and states, respectively).