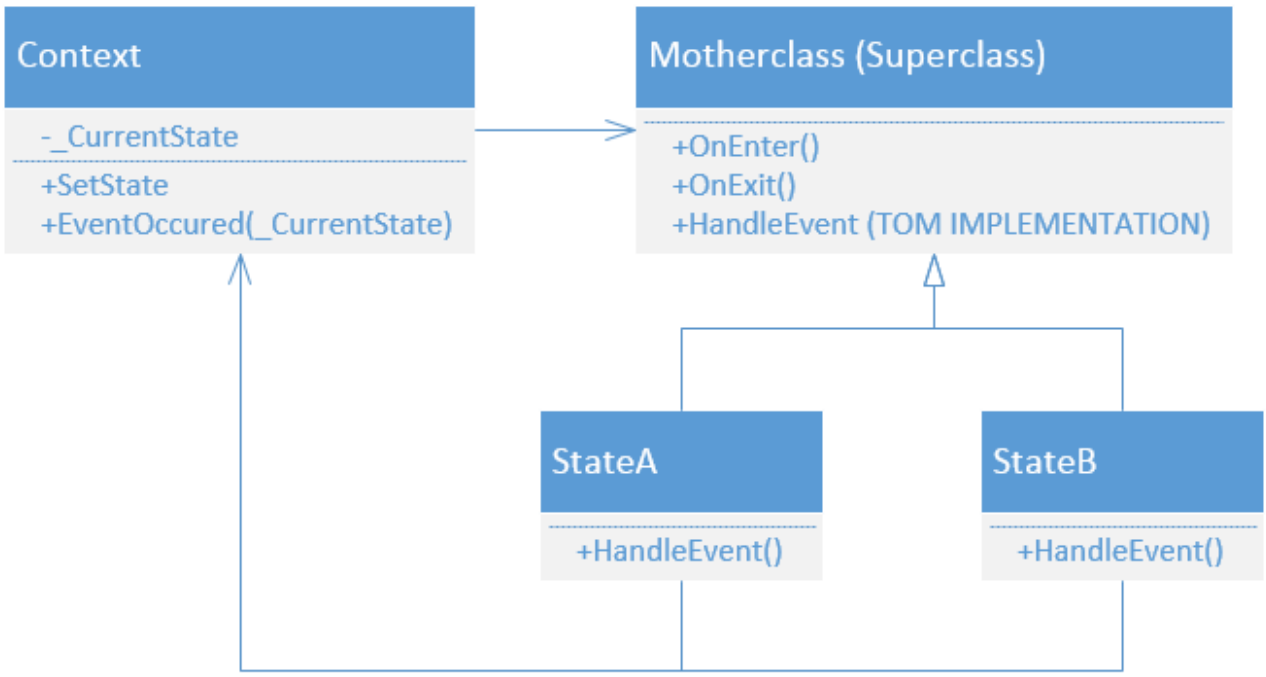
6 – State pattern

- **Redegør for hvad et software design pattern er.**
- **Redegør for strukturen i GoF state pattern.**
- **Sammenlign switchcase-implementering med GoF state.**
- **Redegør for fordele og ulemper ved GoF State.**
- **Redegør for hvordan et UML statemachine diagram mapper til GoF State.**

Software design pattern:

Et design pattern er kernen i en løsning til et problem, som opstår gentagende gange. Årsagen til, at det er et mønster, er, at problemet og løsningen af problemet kan anvendes igen og igen uden, at man gør det samme 2 gange.
Da design patterns altid skal tilpasses vores problem i vores kontekst. Et design pattern er derfor aldrig en færdig løsning på et problem, men en skabelon til løsning af et problem.

Struktur State pattern:



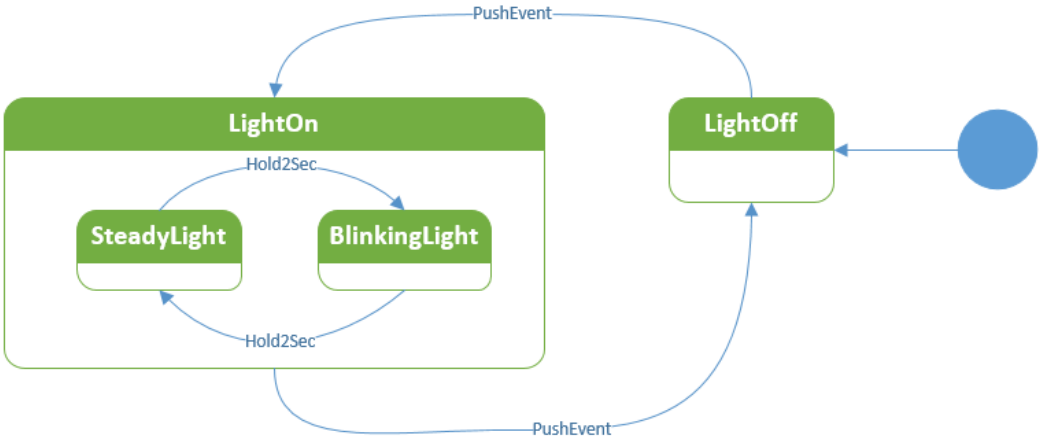
Context anvender metoden EventOccured() og sender currentState med til motherclass. MotherClass implementation i HandleEvent er tom. Alt efter hvilket state, som current state er, bliver eventet behandlet i enten StateA eller StateB.

MINDER OM STRATEGY PATTERN, fordi der bliver valgt en bestemt måde at håndter event på alt efter hvilket state man er i. (minder om hvilken strategi man anvender til bestemte events).

Sammenligning med SwichCase:

Eksempel:

En lampe kan være tændt eller slukket (2 states). 2 korrespondance afbryder (kun 1 event - trykket).



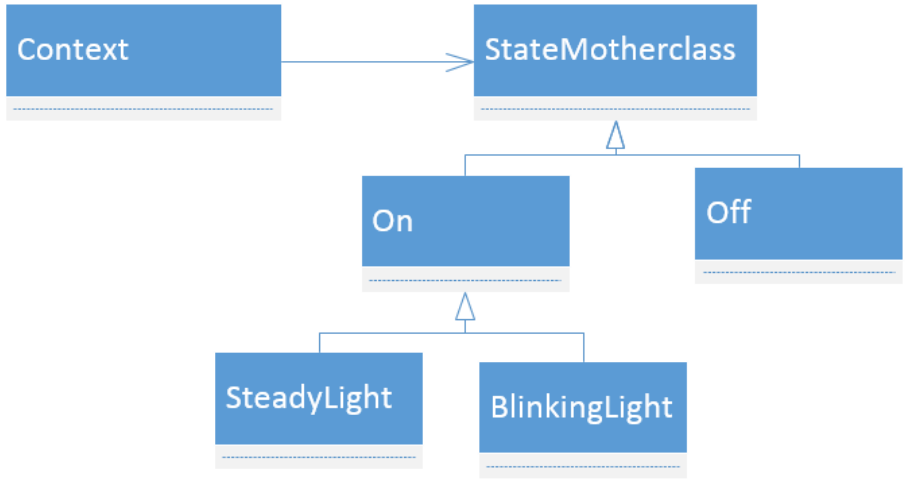

```
switch(state)
    {
        case LightOn:
        switch(state)
        {
            case SteadyLight:
            { ....}

            case BlinkingLight:
            { ....}
        }
        case LightOff:
        TurnOffLight();
    }
```

For hver case med **NESTED STATES** skal der oprettes en switchcase for det **NESTED STATE**.

Ved mange states **->** mange switchcases. Endnu flere nested states.

State Pattern:



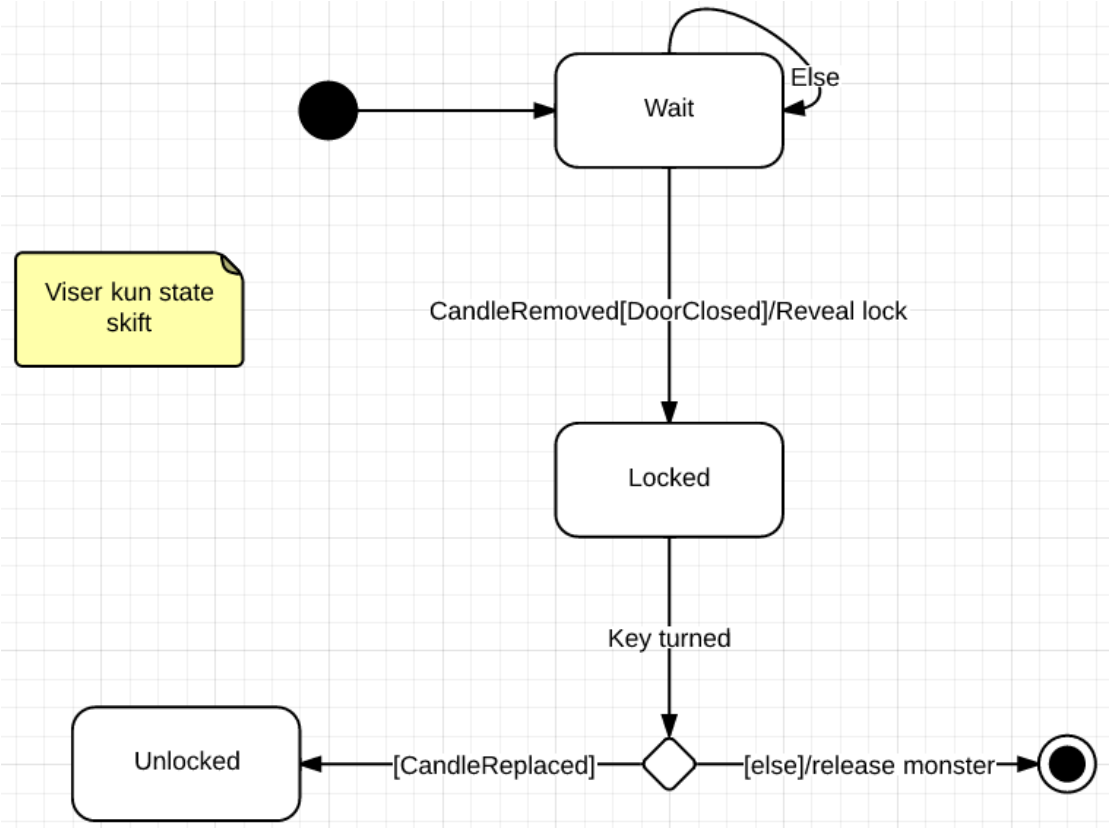
I state pattern opnår man en klar adskillelse af event og state. Da hver state kan implementere handling for hver event.

I Nested Swich opnår man ikke det samme, da eventhandlingen kan ligge sammen med state logikken. Optimeret kan der været lavet controller til eventhandling. Lille opdeling men ikke optimal.

I State pattern er det nemt at tilføje nye states – ikke gældende for swich, da der skal oprettes nye swichcases for alle event.

Swichcase kan anvendes til små state-lignede programmer.

Mapning fra UML State Diagram til State Pattern



- Hver *firkant* bliver til et *state* dvs. sub-klasse af IChestState
    - Wait, Locked, Unlocked
- *Events* viser hvilke funktioner interfacet IChestState skal kunne håndtere, da eneste funktioner der kan ændre stadie!
    - CandleRemoved, Key Turned
- *Guards* viser *logik* som må implementeres i de enkelte konkrete states
- *Activity* fx Reveal Lock, Er kald de konkrete states må kalde tilbage på Chest,
- *Tilstandsskift* viser hvornår vi skal sætte *CurrentState = new xx*