2. GoF State: Nested States

- Brug af nested states til at håndtere komplekse state machines.
- Eksempel: Lommelygte med tilstande som OFF, ON, Low og High, kontrolleret af PWR og MODE knapper.

3. GoF State: Orthogonal States

- Håndtering af orthogonal states ved at kombinere flere state machines.
- To implementeringsstrategier:
    . Sammenfald til en enkelt state machine.
    . Opret to separate state machines og lad konteksten holde en reference til begge.

4. Implementation af Orthogonal States

- **FlashLight Class:**
    - Interface og funktioner til at håndtere tilstande og handlinger.
- **IntensityState Class:**
    - Metoder til at håndtere tilstandsovergange for lysintensitet.
- **ColorState Class:**
    - Metoder til at håndtere tilstandsovergange for lysfarver.

5. State vs. Strategy

- **State Pattern:**
    - Bruges til at modellere et system med tilstande.
    - Overgange mellem tilstande håndteres af tilstandene selv.
- **Strategy Pattern:**
    - Bruges til at ændre adfærden af en del af et program.
    - Klienten kan ændre algoritmen ved kørselstidspunkt.

6. Begrænsninger ved State Pattern

- Beskriver potentielle udfordringer ved implementering af komplekse state machines.

Noter til "8.1 - Patterns - State"

1. Introduktion til State Machines

- Forelæser: Claudio Gomes
- Version: 1.0.2

2. State Machines

- Model af et system med en endelig mængde af tilstande.
- Systemet kan skifte mellem tilstande, når begivenheder opstår.
- Eksempel: Lyskontakt med tilstandene ON og OFF.

3. Implementering af State Machines

- Tre almindelige implementeringer:

. Switch-case.
. State/event tabeller.
. GoF State Pattern.

4. Eksempel på Switch-case Implementering

- Implementering af en lommelygte med forskellige tilstande ved hjælp af switch-case strukturer.

5. GoF State Pattern

- Når state machines bliver komplekse, bliver switch-case og tabelimplementeringer rodet og svære at teste.
- Introduktion af GoF State Pattern for at håndtere komplekse state machines.

6. Struktur og Overgange i State Pattern

- **Context Class:**
  - Ejer state machine og indeholder event handlers og handlinger.
- **State Class:**
  - Abstrakt klasse med standardimplementering af event handlers.
- **Concrete State Classes:**
  - Implementerer specifikke tilstande og deres event handlers.

7. Fordele ved GoF State Pattern

- Hver tilstand implementeres som en underklasse af en fælles abstrakt state klasse.
- Konteksten refererer til én state objekt ad gangen og delegerer state-afhængig adfærd til dette objekt.
- State klasserne holdes statsløse og kan implementeres som singleton.

Kodeeksempler i C++

GoF Template Method

AbstractClass og ConcreteClass:

```
#include <iostream>

class AbstractClass {
public:
    void TemplateMethod() {
        MethodX();
        MethodA();
        MethodB();
        MethodY();
    }
protected:
    virtual void MethodA() = 0;
    virtual void MethodB() = 0;
private:
    void MethodX() { std::cout << "MethodX called\n"; }
    void MethodY() { std::cout << "MethodY called\n"; }
};
```

```
class ConcreteClass1 : public AbstractClass {
protected:
    void MethodA() override
```