

# Design Pattern

Peter Levinsky, IT Roskilde

03.03.2023

# Design Pattern - Beskrivelse

**Navn** – fælles betegnelsen – fag ord

**Problem** – beskrivelse af problemet

**Løsning** – KUN! Design løsning (UML diagrammer)

## Design Pattern – GRASP (General Responsibility Assignment Software Patterns)

- Information Expert
- Creator Pattern
- Controller
- Low Coupling
- High Cohesion

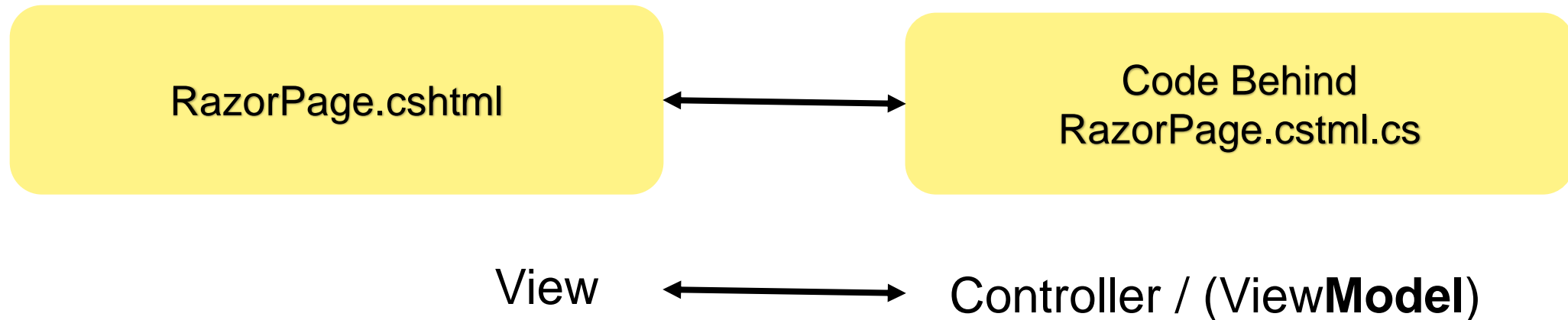
# Design Pattern – GRASP (General Responsibility Assignment Software Patterns)

- **Information Expert**  
Hvem skal man spørge for at få information → den klasse der har data
- **Creator Pattern**  
Hvem kan oprette et objekt ->
  - a) Dem der har information til at initialisere objektet
  - b) Ved Komposite den klasse der består af objekterne

# Design Pattern – GRASP (General Responsibility Assignment Software Patterns)

- **Controller**

What first object beyond the UI layer receives and coordinates ("controls") a system operation?



# Design Pattern – GRASP (General Responsibility Assignment Software Patterns)

- **Low Coupling**

How to support low dependency, low change impact, and increased reuse? ->

Assign a responsibility so that coupling remains low

*Indication of High Coupling– many relations in a DCD*

- **High Cohesion**

How to keep objects focused, understandable, and manageable, and as a side effect, support

Low Coupling? ->

Assign a responsibility so that cohesion remains high.

*Indication of Low Cohesion– few relations in a DCD*

# Design Pattern – Pattern fra 1.semester

- Singleton
  - kun ét objekt (kun set indirekte)
- Controller
  - bag en RazorPage (view) er en Controller (code behind)

# Design Pattern – Kategorier

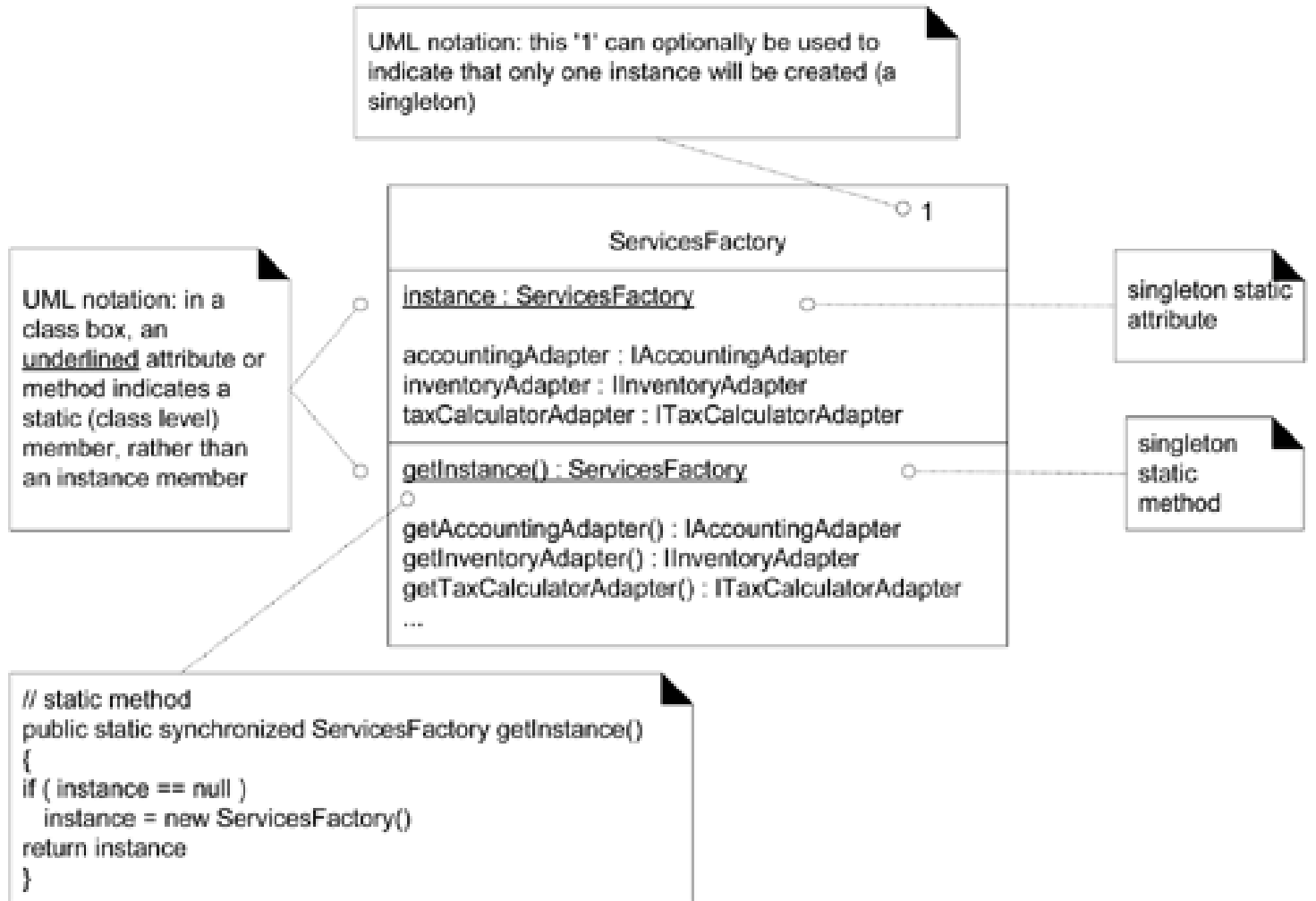
- **Creational Patterns**
  - Factory, Abstract Factory, Singleton ...
- **Structural Patterns**
  - Adaptor, Proxy, Decorator ...
- **Behavioral Patterns**
  - Observer, Template, Chain of Responsibility, Strategi, State ...



# Design Pattern – Creational Patterns

## Singleton

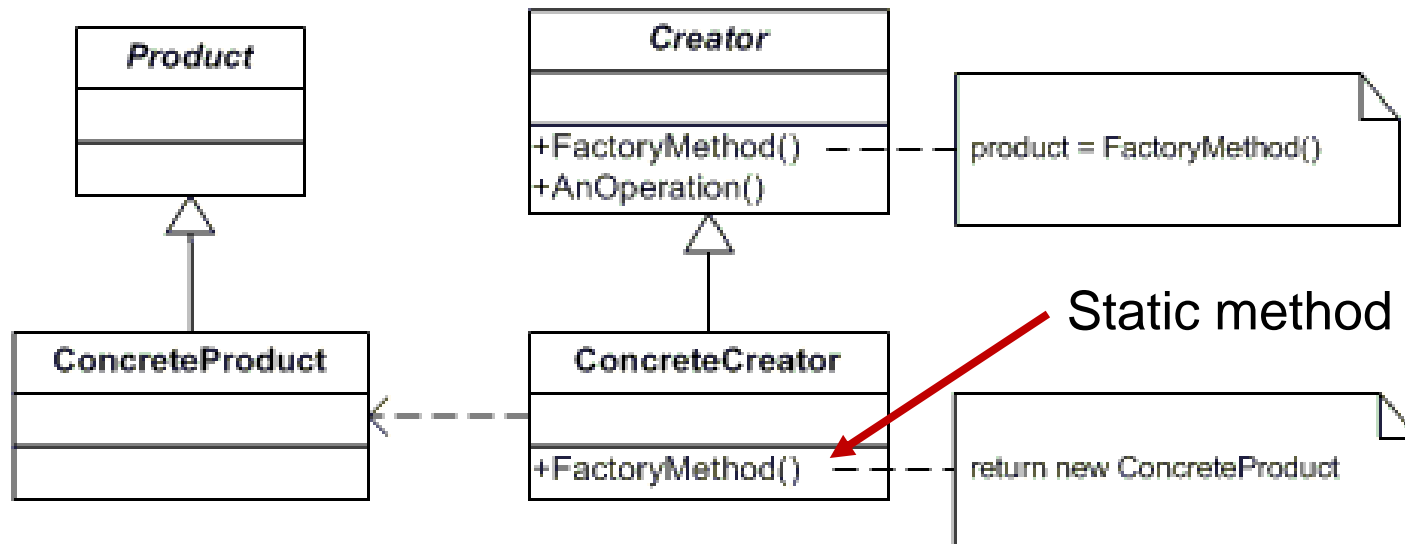
- Problem: Exactly one instance of a class is allowed.
- Løsning:
  1. PRIVATE constructor
  2. static instance field
  3. static read property



# Design Pattern – Creational Patterns

## Factory

- **Problem:** Who should be responsible for creating objects when there are special considerations, such as complex creation logic, a desire to separate the creation responsibilities for better cohesion, and so forth?
- **Løsning:**



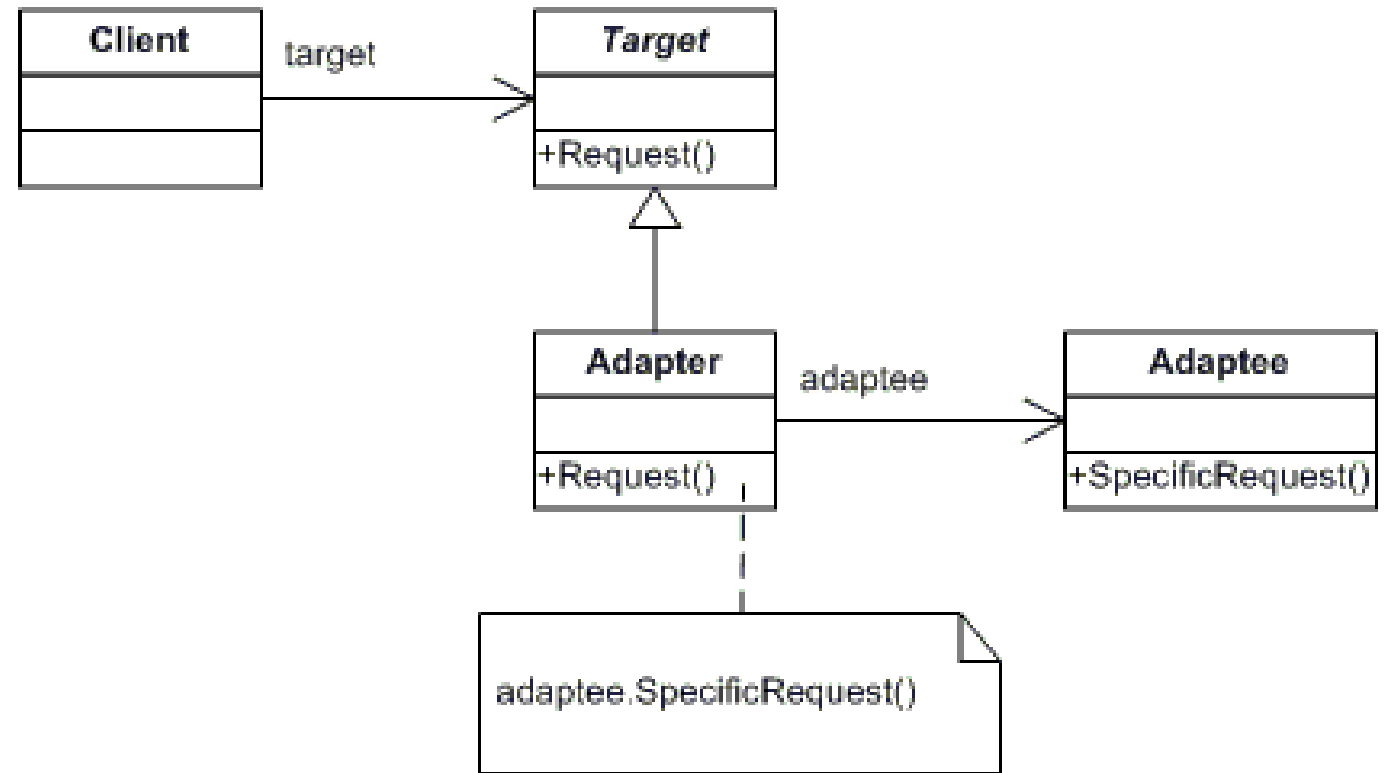
# Demo og opgaver

- Demo af Factory, Singleton og Abstract Factory
  
- Opgaverne Patterns.1 - Factory
- Ekstra opgave Patterns.2 – Abstract Factory
  
  
  
  
  
  
  
- Fortsætter med Structural Patterns kl 12:15

# Design Pattern – Structural Patterns

## Adaptor

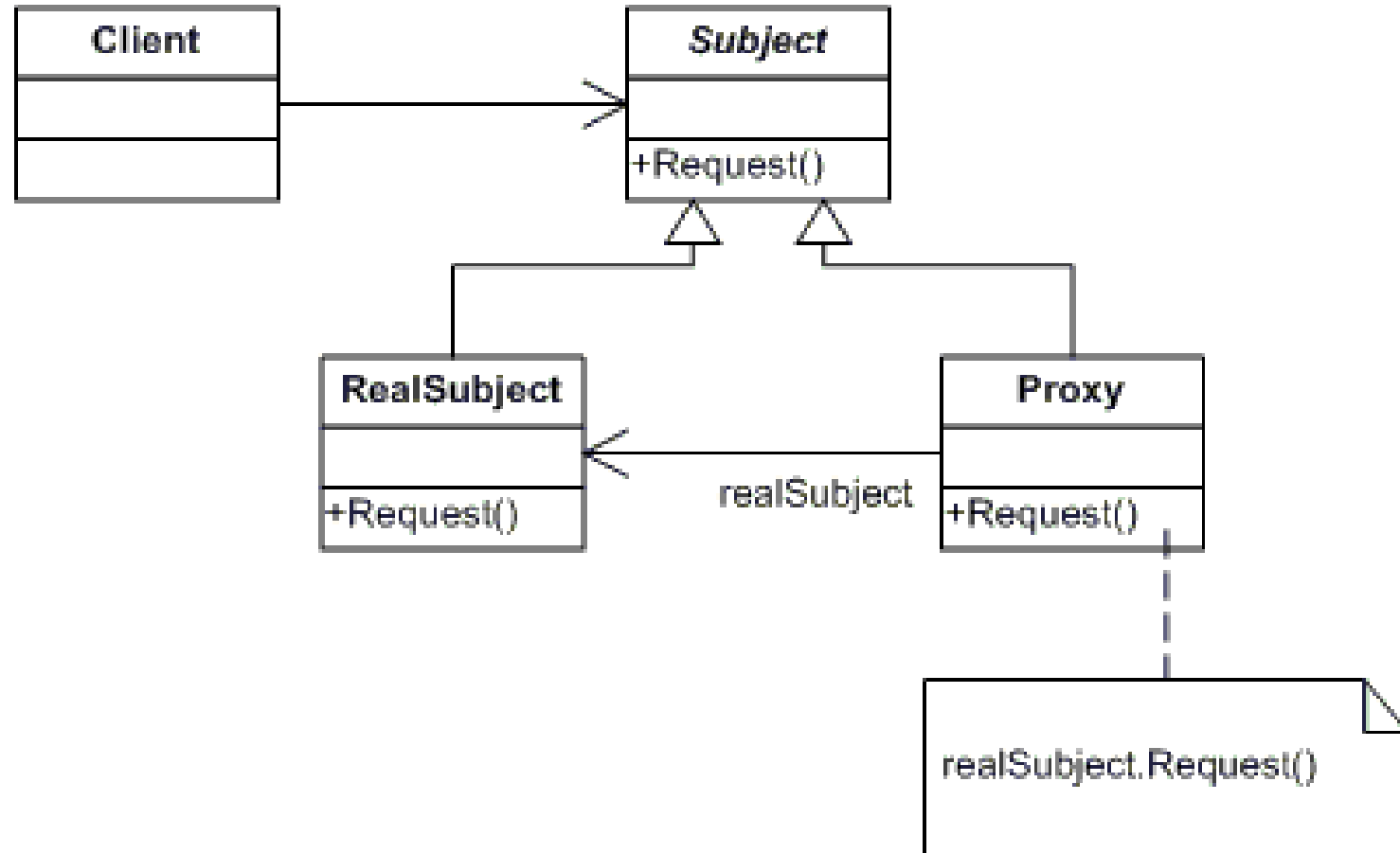
- Problem: How to resolve incompatible interfaces, or provide a stable interface to similar components with different interfaces?
- Løsning:



# Design Pattern – Structural Patterns

## Proxy

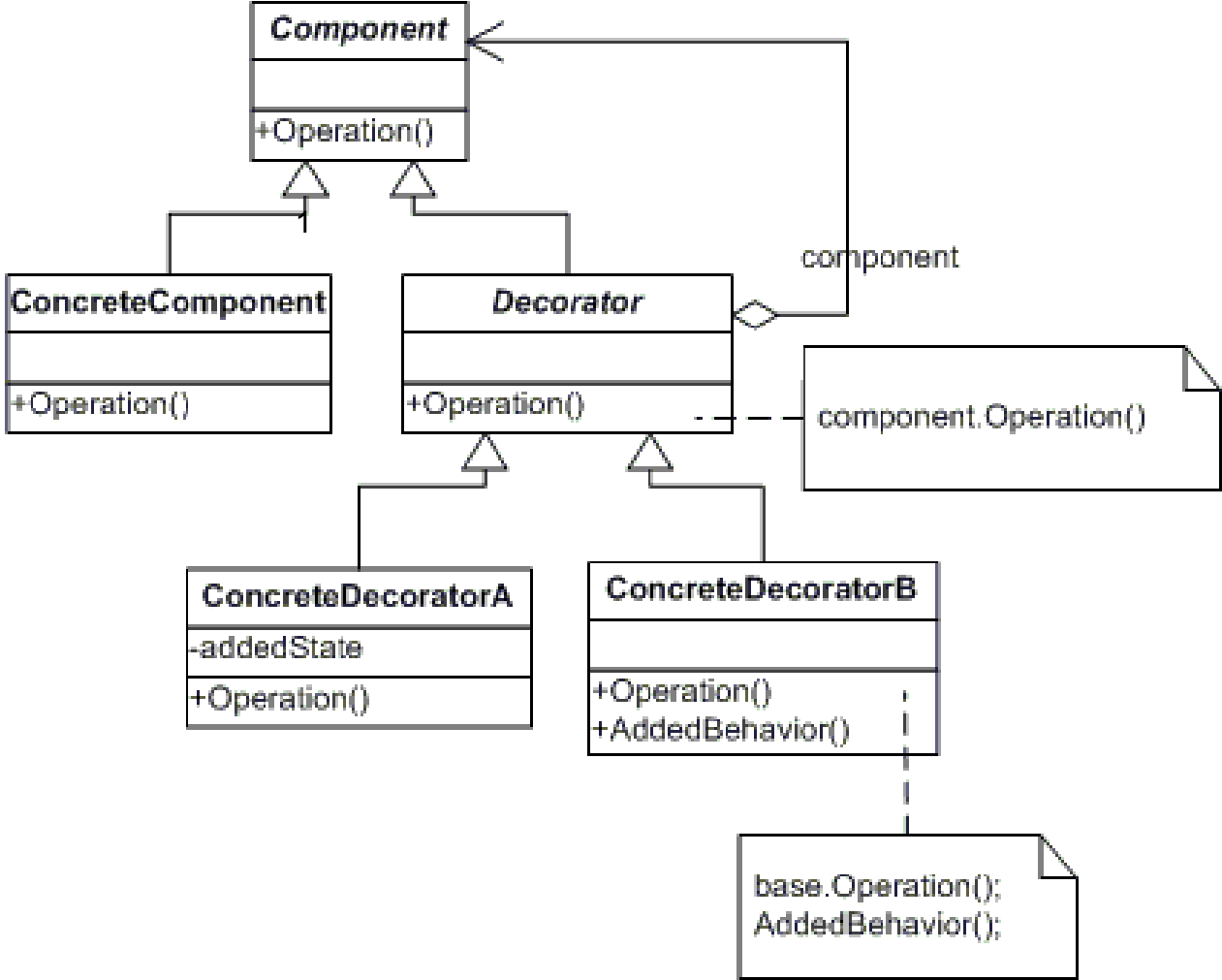
- Problem: How to provide a placeholder for another object to control access to it.
- Løsning:



# Design Pattern – Structural Patterns

## Decorator

- Problem: How to Attach additional responsibilities to an object dynamically
- Løsning:



# Demo og opgaver

- Demo af Adaptor, Facade, Proxy
  
- Opgaverne Patterns.3
- Ekstra Opgave Patterns.4