**Mediator Pattern**

I4SWD-01

**Gruppe 02**

201404118 Anders W. Birkelund

201271001 Rune D. Rask

201405166 Jonas R. Hartogsohn

Indhold

[Introduktion 3](#_Toc482177835)

[Formål 3](#_Toc482177836)

[Type 3](#_Toc482177837)

[Struktur 3](#_Toc482177838)

[Dynamik 3](#_Toc482177839)

[Konsekvenser 3](#_Toc482177840)

[Sammenligning 3](#_Toc482177841)

[Observer 3](#_Toc482177842)

[Implementering 3](#_Toc482177843)

[Konklusion 3](#_Toc482177844)

# Introduktion

## Formål

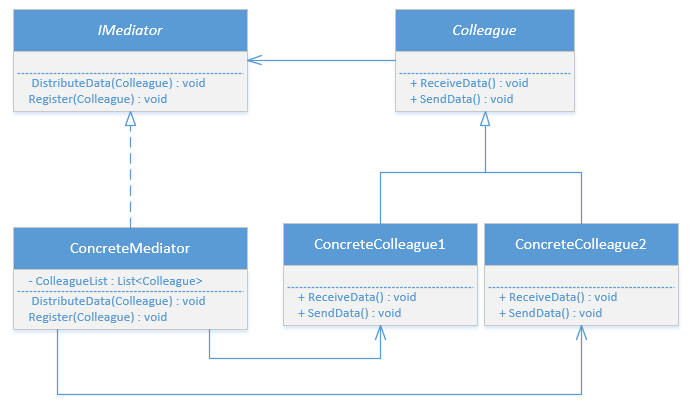
Mediator-mønstret benyttes til at nedsætte koblingen i et system, hvor mange komponenter skal kommunikere med hinanden på forskellig vis. I det givne system ville koblingen forhøjes markant ved tilføjelse af flere komponenter, og derved bliver systemet utroligt svært at vedligeholde. Et sådant komplekst kommunikationsnetværk forsimples ved at introducere mønstret, hvor al kommunikation foregår via et Mediator-objekt, der står for at facilitere kommunikationen mellem komponenterne.[[1]](#footnote-1)

Gang of Four deler deres design mønstre ind i tre grupper: Creational-, Structural- og Behavioral patterns. Mediator-mønstret er et Behavioral Pattern, da det står for kommunikationen og interaktionen mellem objekter.

## Struktur

Mediator-mønstret består af følgende klasse:

* **IMediator** er et interface til den konkrete Mediator-klasse. Her deklareres de generelle metoder for en Mediator.
* **Colleague** er en abstract klasse, hvori de overordnede metoder for Colleagues defineres.
* **ConcreteMediator** implementerer IMediator-interfacet og metoderne heri.
* **ConcreteColleague** nedarver fra Colleague og overrider disse metoder.



ConcreteColleague registrerer sig selv i Mediatoren via Register-metoden. Herved kender Mediator alle Colleagues i form af Colleague-typen. Når en ConcreteColleague skal sende data, gøres dette via Mediatoren, der distribuerer denne data. Skal den pågældende data fordeles til bestemte Colleagues, defineres denne logik i Mediatoren.

## Fordele

Ved at benytte Mediator-mønstret, opfylder man Open/Close-princippet fra SOLID, da man ved at nedsætte den høje kobling opnår at objekternes afhængigheder bliver forsimplet. Dette er en klar fordel, når systemet skal videreudvikles, da man herved får muligheden for at udskifte komponenterne uden det har indflydelse på de andre objekter.

## Konsekvenser

Når man benytter sig af Mediatormønstret, skal man være opmærksom på ikke at lave en gudeklasse, med kendskab til alle objekter i systemet og med høj kompleksitet. I scenarier hvor flere forskellige typer subklasser, som skal benytte sig af Mediatoren, vil det være nødvendigt at have logik, som håndterer de individuelle typer. Dette kan medfølge kompleks og uoverskuelig kode. I sådanne tilfælde kan det være nødvendigt oprette flere typer Mediatorer, som håndterer kommunikation mellem hver de forskellige typer klasser.

# Sammenligning

## Observer

# Implementering

# Konklusion

1. http://www.codeproject.com/Articles/186187/Mediator-Design-Pattern [↑](#footnote-ref-1)