Software Design - Obligatorisk Opgave

Om Mediator Pattern

Bjørn Nørgaard Sørensen
Stud.nr: 201370248

Joachim Dam Andersen
Stud.nr: 201370031

Dennis Tychsen
Stud.nr: 201311503

2. December 2015

Indholdsfortegnelse

1 Problem

Når objekters funktionalitet distribueres ud mellem hinanden, vil der opstå høj kobling, og masser af interkonnektivitet. I et mediator pattern oprettes et separat mediator-objekt, som står for at kontrollere objekters interaktioner med hinanden.

2 Løsning

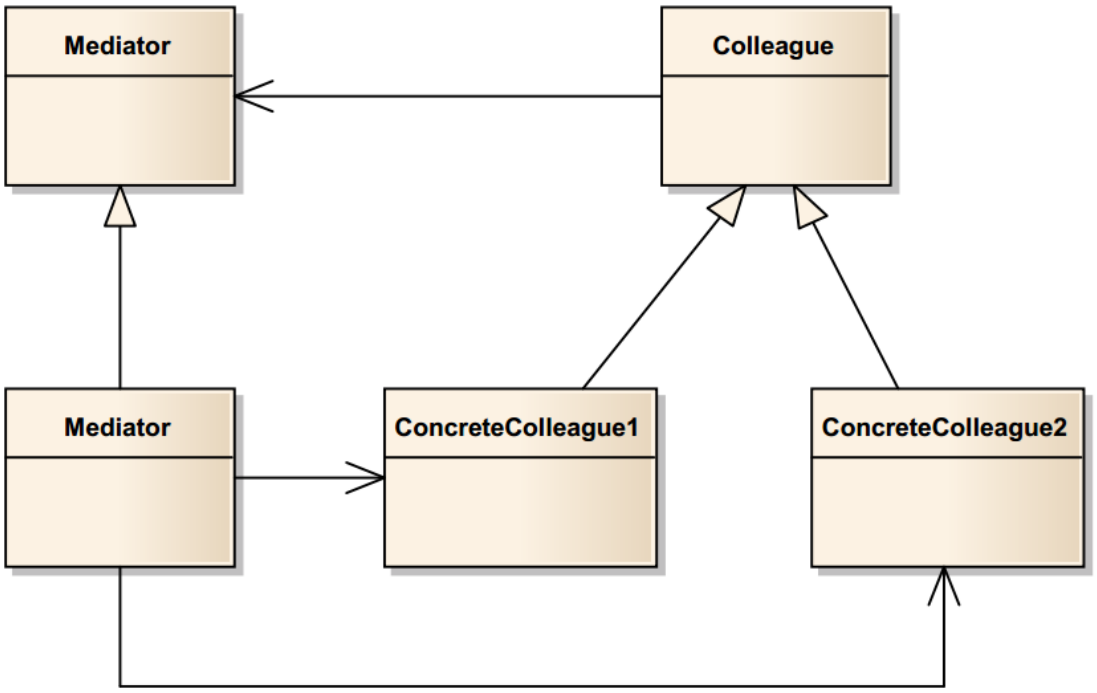

Figure 1: Generelt klassediagram om Mediator pattern.

Brugen af et mediator pattern begrænser mængden af afledte klasser i et system, i og med mediatoren centraliserer funktionalitet der ellers ville være spredt ud på mange klasser. Ved at pakke objekters interkonnektivitet ind i et mediator pattern, får man samtidig skabt et ekstra abtraktionsniveau der gør funktionalitet mere overskuelig.

3 Eksempel

Definition og identifikation af deltagende klassers type:

- Mediator.
  - Definerer et interface til kommunikation with “Colleague objekter”.
- Concrete mediator.
  - Implementerer kommunikationsinterfacet, ved at koordinere Colleague objekter.
- Colleague klasser.

– Hver colleague klasse kender sit mediator object.
– Når colleaguen vil snakke med en anden klasse, kommunikeres der udelukkende gennem mediatoren.

4 Sammenligning

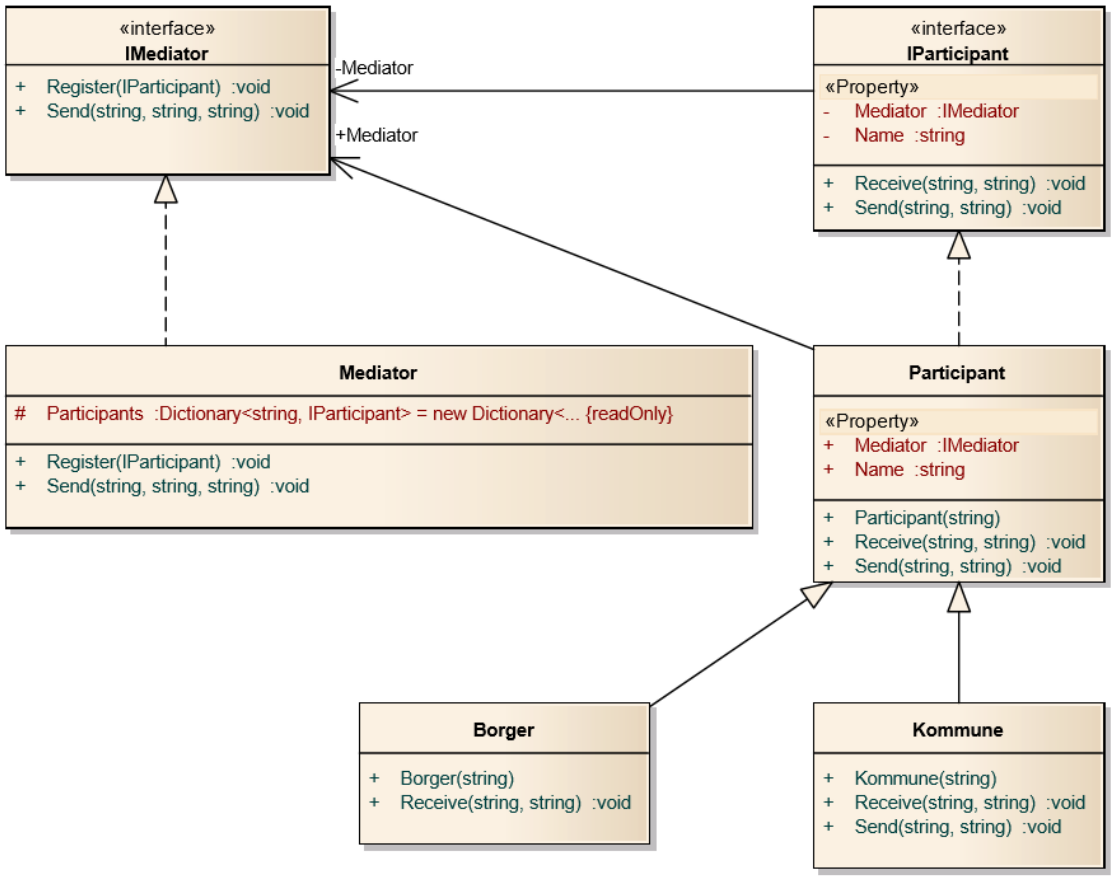

Figure 2: Klassediagram for vores eksempel på Mediator pattern.

Open-Closed Mediator pattern mindsker koblingen i programmet. Dette gør, at alle klasser blot skal bruge én reference (til mediatoren), i stedet for at have adskillige referencer til mange forskellige klasser. Dette er godt for Open-Closed princippet, da programmet er åbent for tilføjelser, men lukket for modifikationer. I princippet kan et objekt fjernes fra programmet, uden at programmet går i stykker. Dette skyldes, at mediatoren blot vil undlade at sende en besked, i stedet for, at der er NullPointer referencer i koden.

Single-Responsibility SRP betyder, at alle klasser kun skal stå for at udføre én ting, altså have et enkelt ansvar. Derudover skal hver klasse også kun have en grund til at ændre sig. Mediatoren overholder SRP, da dens eneste (og fornemmeste) opgave er at sende beskeder fra A til B, altså at "Mediate". Selve funktionaliteten der skal udføres, når der modtages en besked ligger ved de individuelle objekter der modtager beskederne fra mediatoren.

5 Konklusion

Ud fra journalen kan det konkluderes, at Mediator pattern er godt at bruge, i tilfælde, hvor der opstår mange forbindelser mellem mange forskellige klasser. Ved hjælp af mediatoren centraliseres referencerne til de forskellige objekter ét sted, og samler kald til mediatoren, i stedet for enkelte klasser.