Software Design

Aflevering 1 – Memento Design Pattern

Gruppe 8

Navn	Studienummer	AU-ID
Mathilde Juncker Schougaard	202307406	AU749350
Sanne Yding Feddersen	202004994	AU670076
Saga Noelle Bahn	202304998	AU753698

Memento Design Pattern

Memento er et adfærdsmæssigt designmønster. Dette mønster bruges til at gemme og genoprette et tidligere stadie af et objekt, uden direkte at offentliggøre detaljerne bag implementationen. Designmønsteret bruges f.eks. i forbindelse med fortryd knapper i tekstredigeringsprogrammer og at hente tidligere gem fra videospil (Shvets, u.d.).

Memento mønstret består af fire komponenter, *Originator*, *Memento*, *Caretaker* og *Client*.

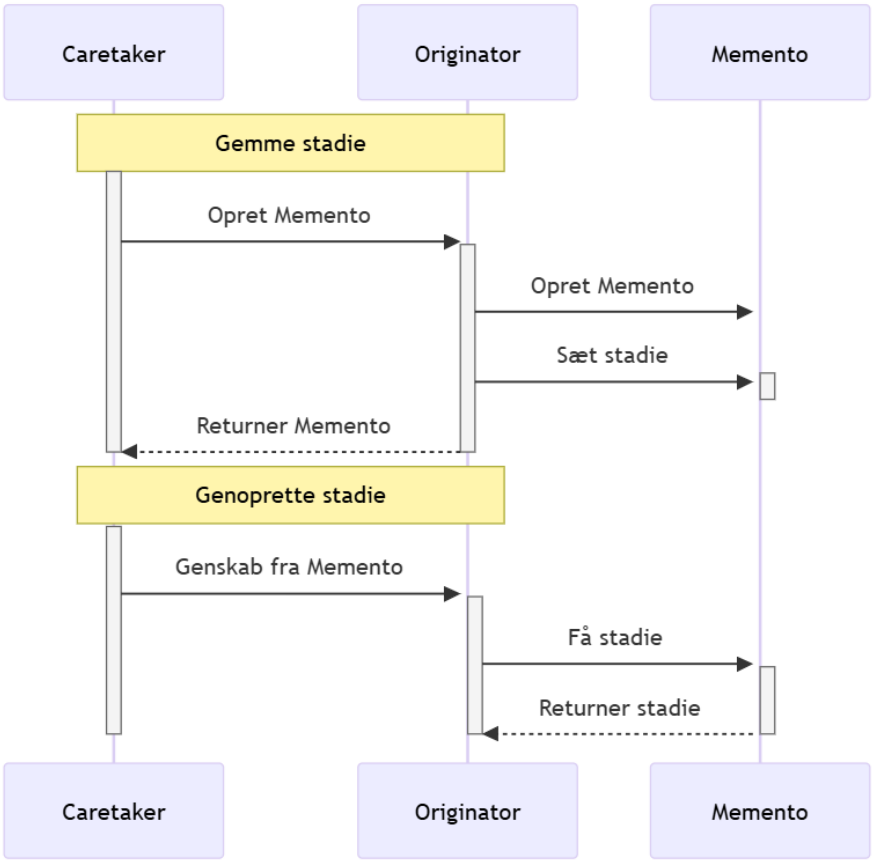
> *Originator* står for at fastslå og vedligeholde objektets stadie. Det er også denne komponent, der opretter og kommunikerer med mementoobjekterne, der opbevarer snapshottet af stadiet og har metoderne til at sætte og hente de forskellige stadier.
>
> *Memento* er selve objektet er opbevarer snapshottet af originatoreren i en tidsperiode. Denne beskytter snapshottet mod at andre får adgang til den, så det kun er originatoreren der har adgang.
>
> *Caretaker* er ansvarlig for at holde styr på de forskellige mementoobjekter. Klassen kender ikke selv til indholdet af dem, men kan anmode om mementos fra originatoreren, således stadiet enten kan gemmes eller genoprettes.
>
> *Client* repræsenterer den del af systemet, der interagerer med originatoreren og caretakeren, således en specielle funktionalitet opnås (Geeks, 2024).

Fordele

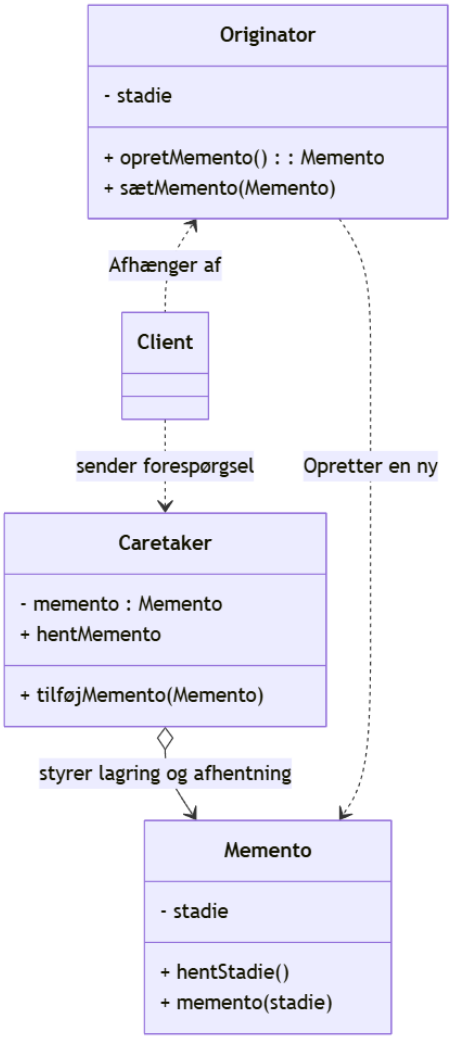
En af fordelene ved mementomønsteret er at mønsteret kan oprette et snapshot af objektets stadie uden at bryde indkapslingen. Dette gør det også mulig at simplificere originatorerens kode, da denne ikke skal bekymre sig om stadiets historie, men i stedet kan lade caretakeren tag sig af det (Shvets, u.d.). Dette gør det muligt for mønstret at oprette en fortryd funktion. Desuden er det også denne funktionalitet, der gør det muligt for f.eks. videospil at benytte sig af snapshots til checkpoints (Geeks, 2024).

Ulemper

Selvom designmønsteret har mange fordele, er der selvfølgelig også nogle ulemper. En af disse er at programmet hurtigt kan komme til at bruge en masse RAM, hvis der oprettes mementos for ofte. Derudover er det ikke muligt at garantere at stadiet med mementoet forbliver urørt i alle programmeringssprog. Dette drejer sig f.eks. om PHP, Python og JavaScript (Shvets, u.d.). Yderligere kan mønstret være overflødigt, hvis de tidligere stadier af objektet er nemme at genskabe. Der er også risiko for at koden bliver mere kompleks og uigennemskuelig, især hvis fortryd ikke er en nødvendig funktion for programmet (Geeks, 2024).



Figur 1: Sekvensdiagram efter (Geeks, 2024).



Figur 2: Klassediagram over Memento efter: (Geeks, 2024).

Sammenligning med State Pattern

State-mønsteret er også et adfærdsmæssigt designmønster, der bruges til at ændre objektets opførsel dynamisk baseret på dets tilstand. Hvor Memento fokuserer på at gemme og gendanne objektets stadier, ændrer State objektets adfærd ved at skifte tilstand. Memento bruges til fortrydelse og tilstandsgendannelse, mens State ændrer objektets reaktioner uden at gemme tidligere stadier. I modsætning til Memento, der arbejder med snapshots, bruges State til at repræsentere forskellige adfærdsmønstre i objektet. Eksempelvis kan State Pattern bruges i en medieafspiller, hvor den kan håndtere funktioner som ”Play” og ”Pause”, hvor adfærden af programmet ændres med tilstanden. Memento kan derimod bruges til gemme hvor positionen på afspilningen er og dermed gendanne den senere.

Sammenligning med Command Pattern

Command-mønsteret er ligesom memento et adfærdsmæssigt designmønster. Modsat memento bruges Command-mønsteret til at indkapsle anmodninger som objekter. Hvor Memento fokuserer på at gemme og gendanne objektets tilstand uden at afsløre implementeringsdetaljer, bruges Command

til at udføre, fortryde og logge handlinger. Command kan oprette en fortryd-funktion ved at udføre modsatte handlinger, hvorimod Memento gendanner en tidligere tilstand uden at ændre objektet. Command er derfor mere fleksibelt til komplekse handlinger, mens Memento er bedre til at bevare objektets historik. Eksempelvis kan memento gemme hele dokumentets tilstand i et tekstredigeringsprogram, hvorimod Command gemmer individuelle handlinger og dermed kan fortryde dem én ad gangen.

Konklusion

Memento Design Pattern er god, når der er behov for at gemme og gendanne tidligere tilstande af et objekt, uden at skulle rode med dets interne struktur. Mønsteret gør det muligt, at implementere funktioner som undo/redo eller save/load, hvilket gør det brugbart i f.eks. spil og grafiske applikationer.

En af fordelene ved Memento err, at det bevarer objektets integritet ved at opretholde encapsulation, samtidig med at det gør det muligt at navigere mellem forskellige versioner af data. Til gengæld kan det godt blive lidt tungt, hvis man gemmer mange snapshots af store objekter – det kan hurtigt sluge en masse hukommelse.

Sammenlignet med Command Pattern, der gemmer handlinger i stedet for tilstande, og State Pattern, der ændrer objektets adfærd afhængigt af dets tilstand, så tilbyder Memento en lidt anderledes tilgang til at arbejde med historikstyring.

Samlet set er Memento Pattern en god løsning, når applikationer kræver historikstyring, man skal bare være opmærksom på, hvor meget hukommelse det kan kræve.