7. Patterns 5: Model-View-Controller (MVC) og Model-View-Presenter (MVP)

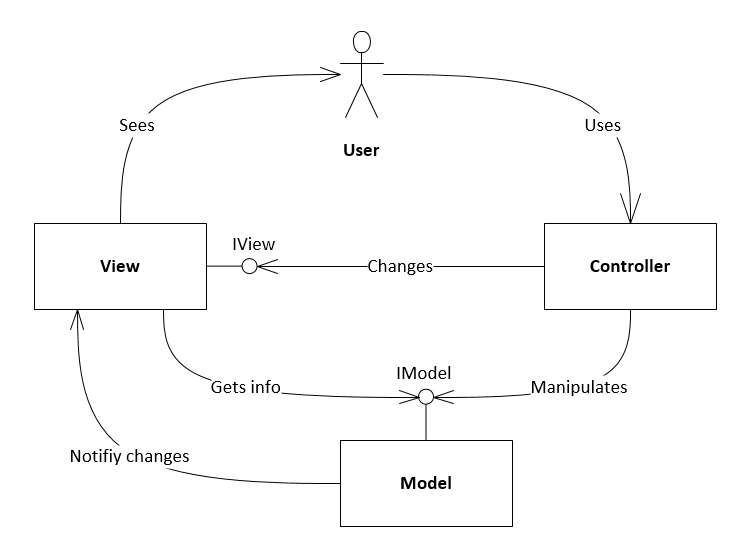
# Redegør for, hvad et software design pattern er.

Design Pattern kan beskriver et problem der opstår igen og igen og derefter beskriver kernen i en løsning til det problem. Det der gør det til et mønster er så at man kan bruge løsningen om og om igen, uden at gøre samme.

Når vi anvender designregler skal det nemlig altid tilpasses. Et design pattern er derfor aldrig en færdig løsning på et problem, men en skabelon til løsning af et problem.

# Redegør for Model-View-Control mønstret og dets variationer.

Dette pattern forsøger at adskille GUI logik og forretnings logik ved at opdele koden i dele der hvert har forskellige ansvar.



Model-View-Controller:

* **Model** er forretningslogikken som er defineret af kravene for systemet.

Det er her alt applikationens logik bor det er også her vi har data tilgang, når der ændres i modellen notificere modellen view’et om at der er sket ændringer den er observable.

Derved er modellen også frakoblet fra GUI’en så den kan bruges andre steder.

* **Controllers** opgave er at beslutte hvilke ændringer der skal ske i modellen ud fra brugerens handlinger.

Hvis der trykkes på en knap kan controlleren fortælle view at det skal ændre sig den kan også ændre i modellen og lade denne opdatere view.

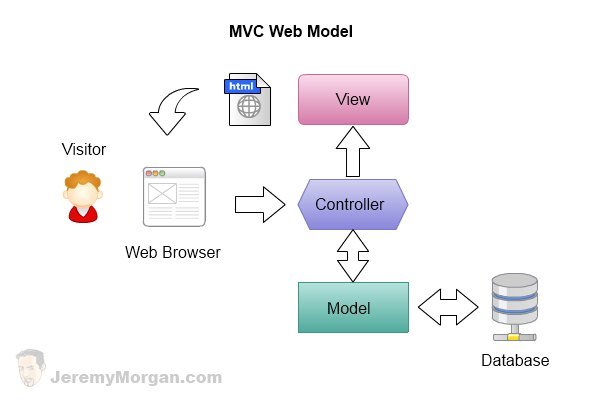
* **Views** opgave er at præsentere modellen for brugeren

View kan være implementeret med stratefy pattern hvor controller så vil være den der giver view’et den strategy. View kan også være Composite da det kan bestå af flere elementer som er ”Composites” der indeholder ”leafs” eller andre ”composites”

MVC er godt til at vise den samme model på flere måder da den kan have forskellige views.

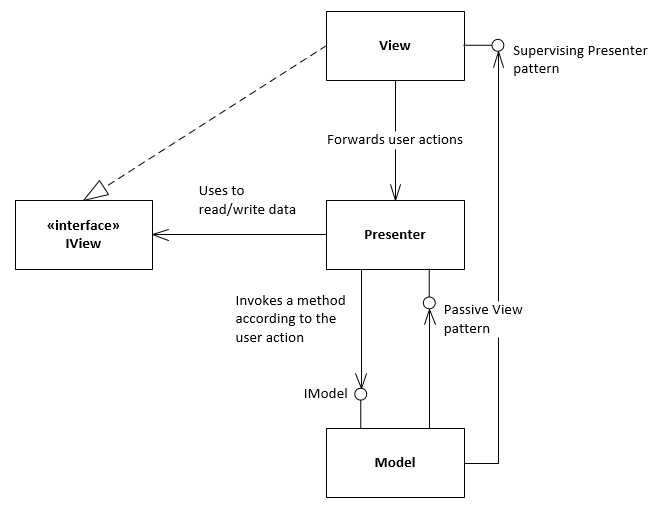
## Varianter

Model 2 er en variant der er tilpasset til brug i web øjemed. Her er indskudt en front controller som tager sig af kommunikationen til bruger. View vil i dette tilfælde bestå af HTML og javascript som vil blive ændret når modellen opdateres det foregår bare igennem controllers da view ikke kan have direkte kontakt til model da de er adskilte over web.



# Redegør for Model-View-Presenter mønstret og dets variationer.

Model view presenter er en videre udvikling af MVC. Den findes i flere varianter, Passive view og Supervising presenter.



For **passive view** gælder:

* **View**

View skal være så dumt som muligt, det består af en række kontroller/elementer/ect. som viser modellen, bruger interaktion går fra view og ned i presenter. Dvs. det skal ikke tage stilling til noget men bare være præsentation

* **Model**

Er repræsentationen af data (ligesom MVC).

* **Presenter**

Presenter manipulere modellen ud fra de events der kommer fra view. Presenter opdatere View igennem IView interfacet står også for formatering. GUI logikken bor i presenter.

For **Supervising Presenter** gælder:

* **View**

Her tillades det at lave databinding ned i modellen. Ellers virker view som før.

* **Presenter**

Her skal presenter stadig kommunikere med modellen om hvad der er sket i view.

Der skal ikke laves så meget kode i presenteren i denne verison men det er svære at teste og der er mindre indkapsling.