1 Hva er et designmønster?

Systematisk gir navn, motiverer og forklarer et generelt design som hjelper et kjent design problem i objekt orienterte systemer. En løsning.

Antpattern

- En ofte brukt men dårlig løsning på et designproblem
- Systemutvikling er en gren i utvikling og tidligere gode design patterns kan over tid bli ansett som antipatterns
- Singleton, et GoF, kan i noen ses på som et antipattern fordi det introduserer global tilstand. Gliobal tilstand er stort sett uønsket
- Det er uenigheter

Designmønstre

- Å ha kunnskap om designmønster gir oss mulighet til å snakke om kodedesign på en bedre måte, siden man relatere til eksisterende praksiser
- Noen ganger bedre å bruke støtte i programmeringsårket
- Angular bruker MVC(designmønster)

Forskjellige designmønster

- DAO / Repository
- Middleware / Pipeline / Meditator
- Service / Facade
- Adapter / Listener
- Observable / Listener
- Dependency Injection / IoC
- Singleton
- Polymorfisme / Mocking

MiddleWare / Pipeline / Decorator / Wrapper
Et objekt likt interface som orginialen men som tillegger ekstra funksjonalitet Sentralt i AOP Legger typisk til generelle mekanismer som logg inn, sikkerhet.

Adapter

- Definerer et grensesnitt på et høyere nivå som gjør det et subsystem

- Service i ReactJS hvor vi legger kall i backed i en serviceclasse.

Repository, DAO

- Definerer et grensesnitt på et høyere nivå som gjør det et subsystem
- per entitet i en domenemodell
- kan kobmineres med DI

Singleton er en statisk klasse der du kan få tilgang til alt

MVC

- Model holde data
- View sørger for presentasjon
- Controller utfører funksjonalitet

Service / Facade

- Et objekt som konverterer et grensesnitt til et annet
- For eksempel eksterne systemer
- Eller mot en database (DataAccessObject / Repository)

Coupling

- Handler om hvor stor grad en modul har koblinger til, har kunnskap om eller avhenger av andre moduler.
- Endring i moduler kan blir vanskelig for andre moduler.
- Isolert sett blir en modul vanskeligere å forstå
- Bedre med løse koblinger(færre konsekvenser, lettere å gjenbruke, lettere å teste)

Cohesion

- Hvor fokusert er en modul?
- Har en modul en tydelig og avgrenset oppgave?

Polymorfisme

- Separering av interface og implementasjon
- Hvis vi seperarer en klasse i en interface og en eller flere implementasjoner kan velge blandt disse under kjøring
- Dette er essensielt ved testing. Vi kan ikke alltid bruke reelle ressurser.

Dependency Injection IoC

- I stedet for at et objekt oppretter sine avhengigheter selv, får de dem servert av klienten
- Med dependency injection blir avhengigheter tydliggjort i grensesnittet(interface). Det blir letter å forstå koden.
- Med DI blir alle injections laget i klienten, og sender dem nedover treet.
- SPRING og Angular er DI rammeverk

Mocking

- Man kan bruke mock objekter for å etterligne reelle objekter
- crash test dummy
- brukes om reelle objekter er upraktisk å bruke i en test
- Mock-rammeverk: for å lage mock objekter raskt og effektivt
-
  - Mockito
  - JMockit
  - PowerMock