Design og Testbarhet

Gruppetime 15

Gruppe 3

agneshb@uio.no

I dag:

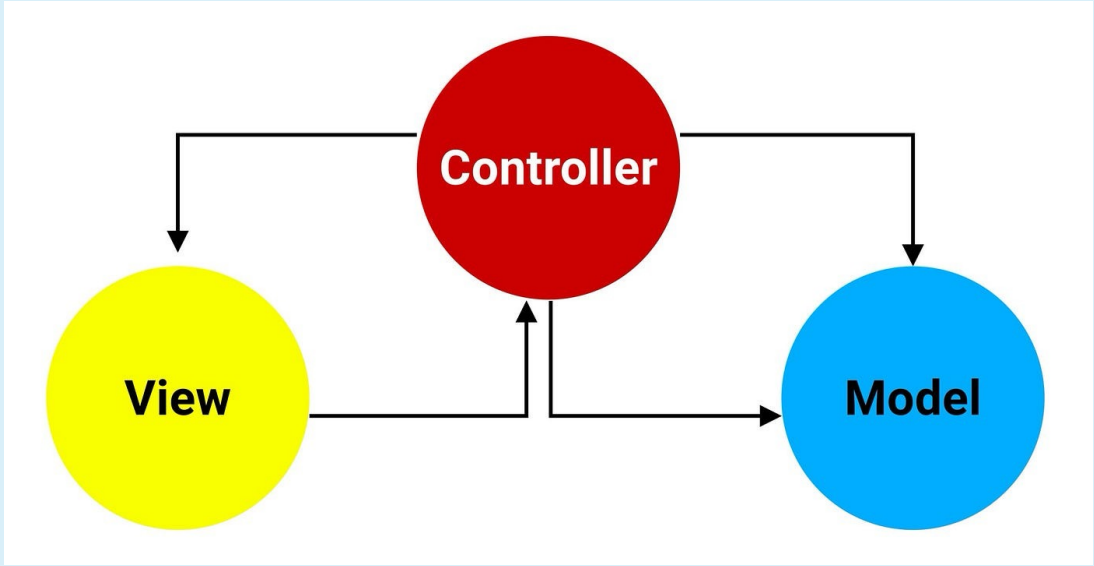
- SOLID-prinsippene
  - Eksempel MVC
- Testing av programvare

SOLID-prinsippene

- Single Responsibility Principle (SRP)
- Open/Closed Principle (OCP)
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

SOLID - Single Responsibility Principle (SRP)

- Klasser har ett veldefinert ansvarsområde.
- Prinsippet sikrer høy grad av kohesjon



MVC:

I MVC er hver komponent (Model, View, Controller) ansvarlig for en bestemt oppgave. Model håndterer data, View håndterer presentasjon og Controller delegerer mellom Model og View.

SOLID - Open/Closed Principle (OCP)

- Vi legger til ny funksjonalitet med ny kode (utvidelse) fremfor å endre eksisterende kode (modifisere).
- OCP er også knyttet til innkapsling og aksessmodifikatorer.
  - Public/private

MVC:

Tillater enkel utvidelse. For eksempel kan vi legge til nye visninger eller modeller uten å endre eksisterende kode.

SOLID - Liskov Substitution Principle (LSP)

- Objekter av superklasser bør kunne byttes ut med objekter av subklasser
- Arv og polymorfi.
- Gjelder også for objekter av klasser som implementerer samme interface/grensesnitt

MVC:

Kan erstatte eller endre visningen uten å påvirke resten av applikasjonen. Dette er fordi alle visninger skal følge et standard interface/grensesnitt.

SOLID - Interface Segregation Principle (ISP)

- Heller flere små interface/grensesnitt enn ett større → klassene trenger bare å implementere metodene de trenger.

MVC:

Kommuniserer komponentene gjennom veldefinerte grensesnitt. Dette betyr at komponentene ikke er tvunget til å implementere metoder de ikke bruker.

SOLI**D** - Dependency Inversion Principle (DIP)

- Objekter er avhengige av abstraksjoner (interface/grensesnitt) fremfor konkrete implementasjoner.
- Bidrar til lav kobling

MVC:

Høy-nivå moduler (som controller) er ikke avhengige av lav-nivå moduler (som model). Controller er i stedet avhengig av en abstraksjon av Model (et interface/grensesnitt).

Testing av programvare

Enhetstesting

- Tester objekter (moduler) i isolasjon, uavhengig av andre objekter, for å sikre at de mindre delene av programmet fungerer etter hensikten før vi setter delene sammen.
- Vi jobber etter AAA:
  - Arrange (testoppsett),
  - Act (testutførelse),
  - Assert (resultatsjekk).

Integrasjonstesting

- Tester sammenhengen eller samarbeidet (i.e. Integrasjonen) mellom objekter (moduler), for å sikre at større deler eller hele programmet fungerer etter hensikten før det kan brukes.