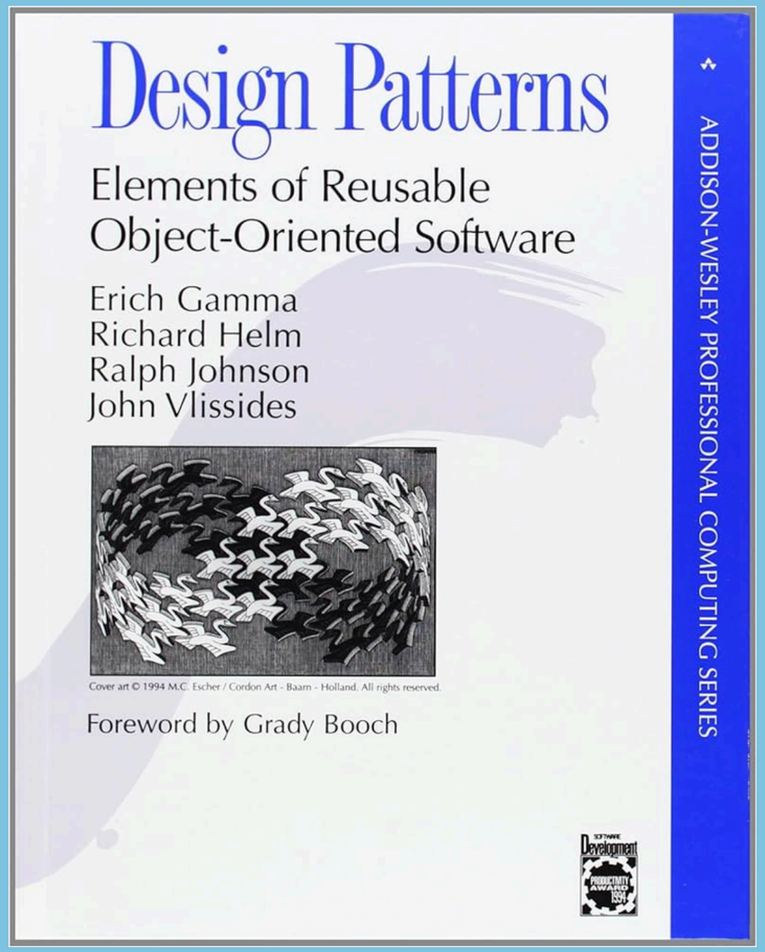
Uke 11 - Designmønstre

Gruppe 5 - jonasbny@ifi.uio.no

Agenda



Designmønstre (design patterns)

Behavioral
(Iterator, Observer)

Creational
(Singleton)

Structural
(Decorator)

ukas shoutout

oppgaver

Design mønstre (design patterns)

Generelt er designmønstre gjenbrukbare måter å strukturere programmer på, og forskjellige mønstre har fordeler og ulemper avhengig av hva vi ønsker å oppnå i programmene våre.

Når man har programmert en stund vil man kunne begynne å kjenne igjen problemer man har støtt på før.

Da kan det være lurt å gjenbruke løsninger fra tidligere erfaring.

Vi “gjenbruker” kunnskap

Design mønstre

I boken Design Patterns nevnes det 23 mønster som bredt representerer løsninger på designproblemer.

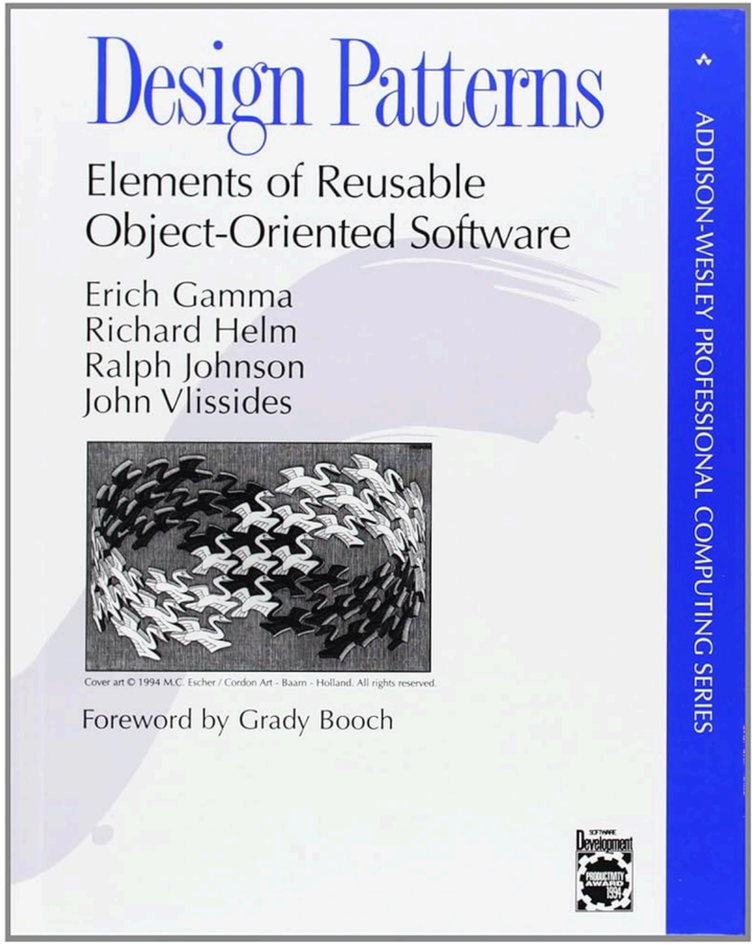
Vi skal se på 4 av de:

Iterator

Observer

Singleton

Decorator



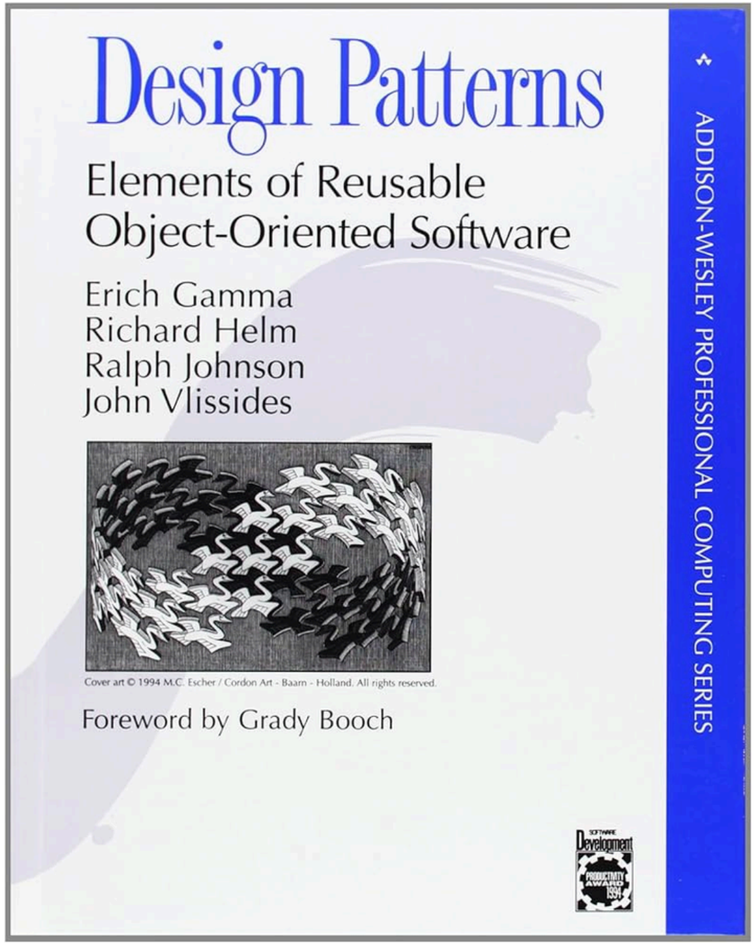
Klassifisering

Vi deler mønstrene opp i tre typer

Behavioral
- Iterator, observer

Creational
- Singleton

Structural
- Decorator

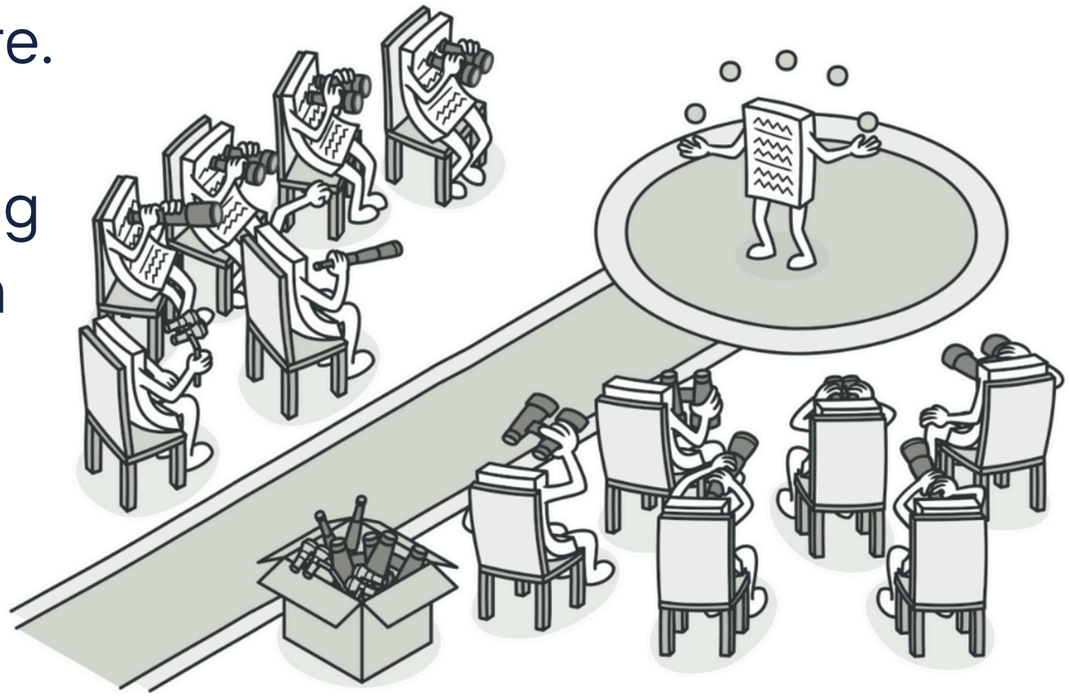


Behavioral

Hvordan objekter kommuniserer med hverandre.

identifiserer hvordan objekter kommuniserer, og hvordan slik kommunikasjon kan realiseres som klasser og objekter.

Her er det mønster som:
command, strategy, **observer**, **iterator**, state, template method, visitor, memento, c.o.r

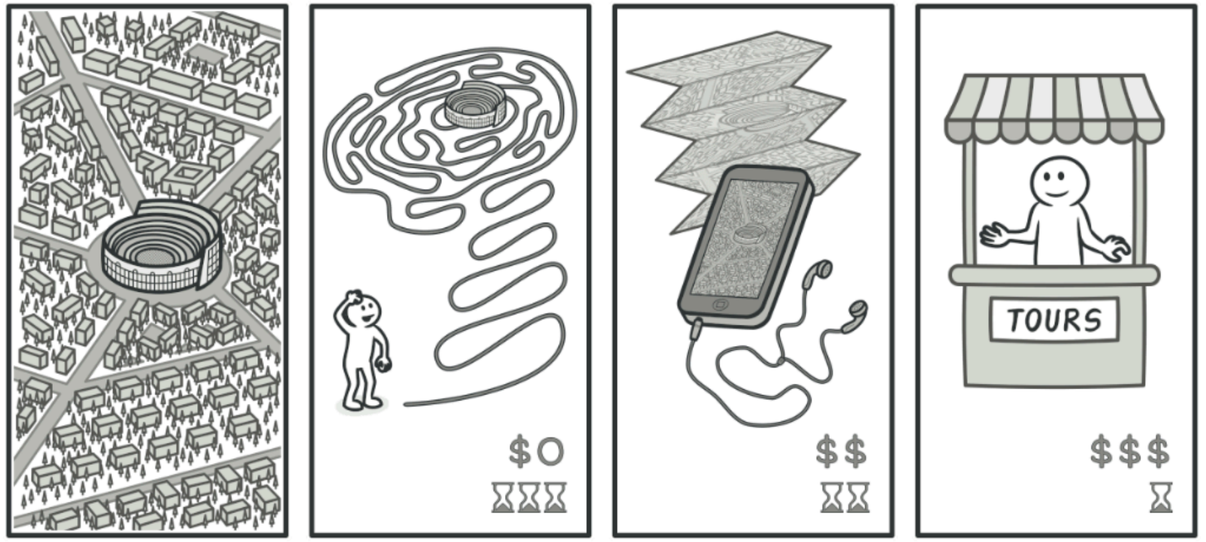


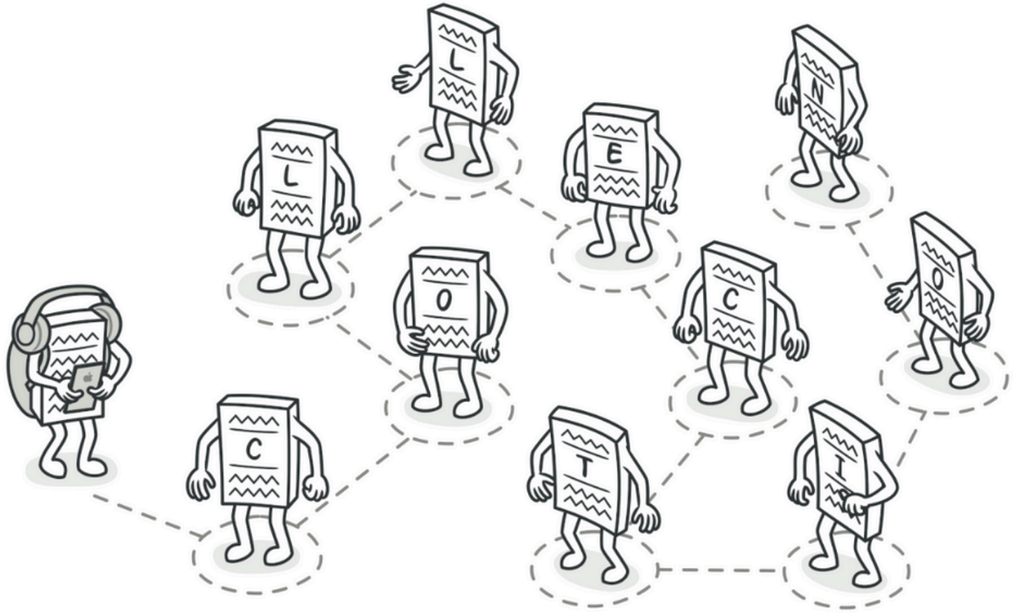
Iterator

Definerer hvordan verdier i et objekt skal itereres gjennom

Gjør det mulig å iterere gjennom samlinger på en enhetlig måte.

Skjuler den interne strukturen til samlingen.





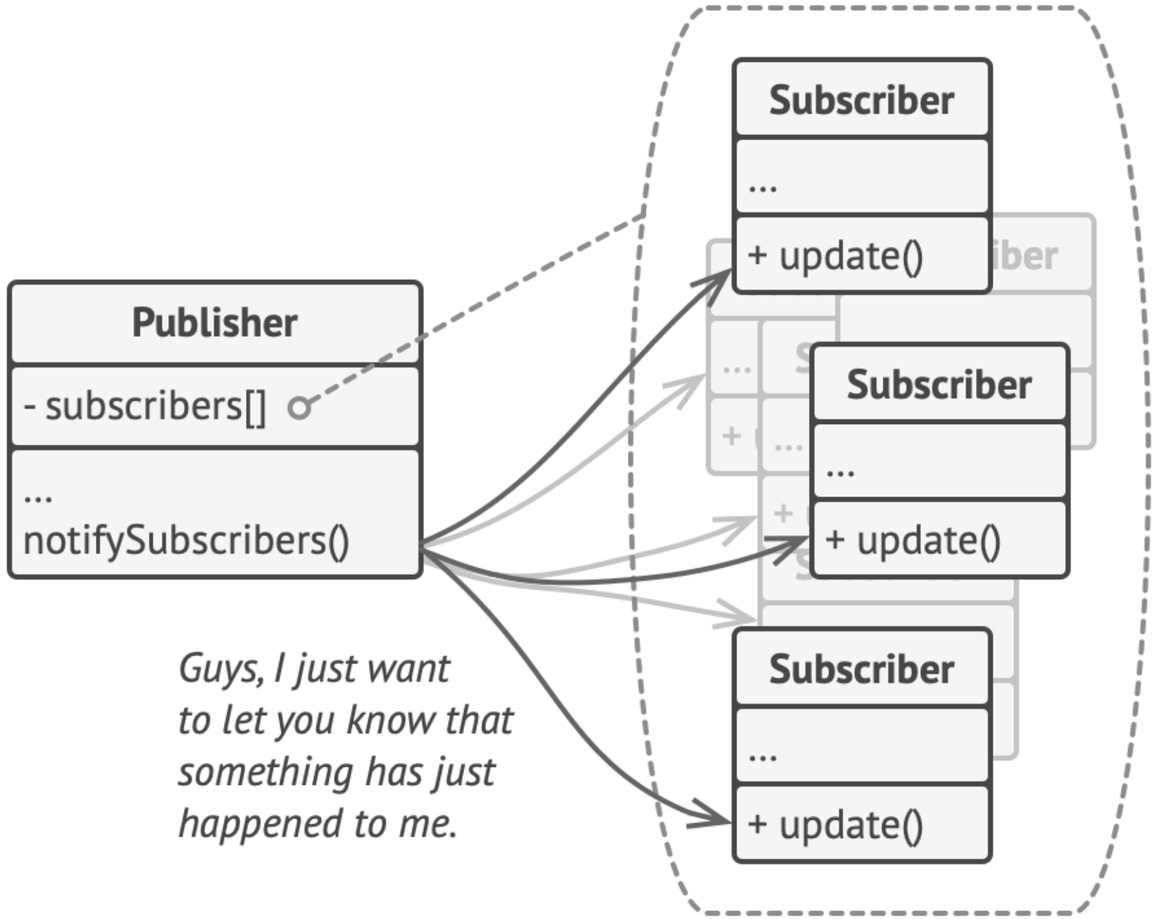
Observer

subjekt og observatør forhold eller en til mange

Observatør ønsker å få beskjed når noe endrer seg hos *subjektet*

subjekt kan observeres av en eller flere *observatører*

VG eksempel
publisher (subject) og subscriber (observer)



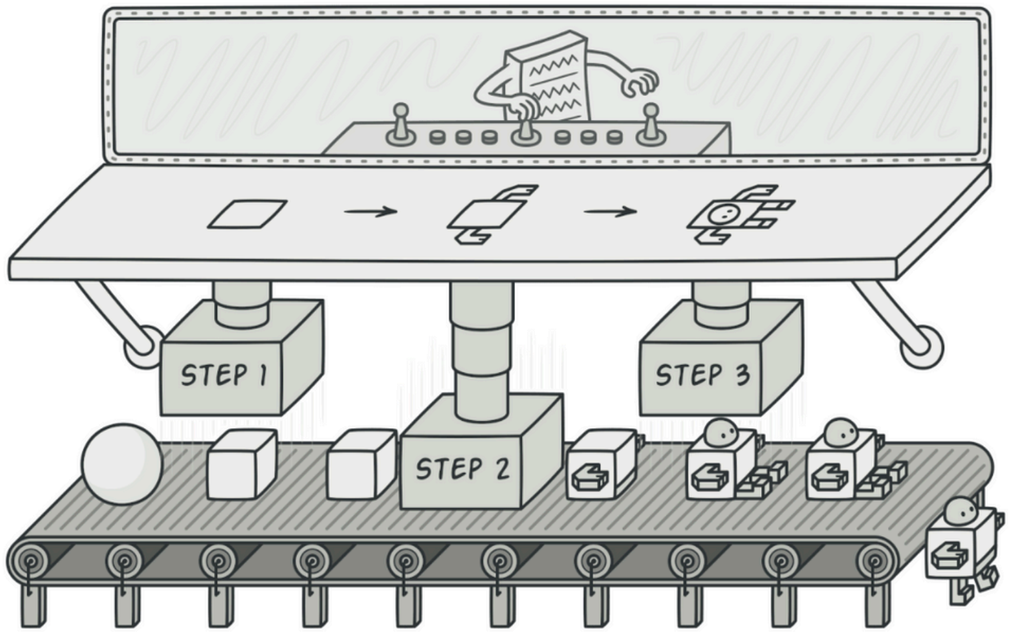
Creational

Denne klassen med designmønstre handler i stor graf om å **opprette** objekter på ulike måter

kontroll over hvordan objekter lages, utifra gitte forutsetninger.

creational patterns gir oss objekt “lage” mekanismer for å øke fleksibilitet og gjenbruk av kode

Her er det mønster som:
Factory method, abstract factory, **singleton**, builder, prototype

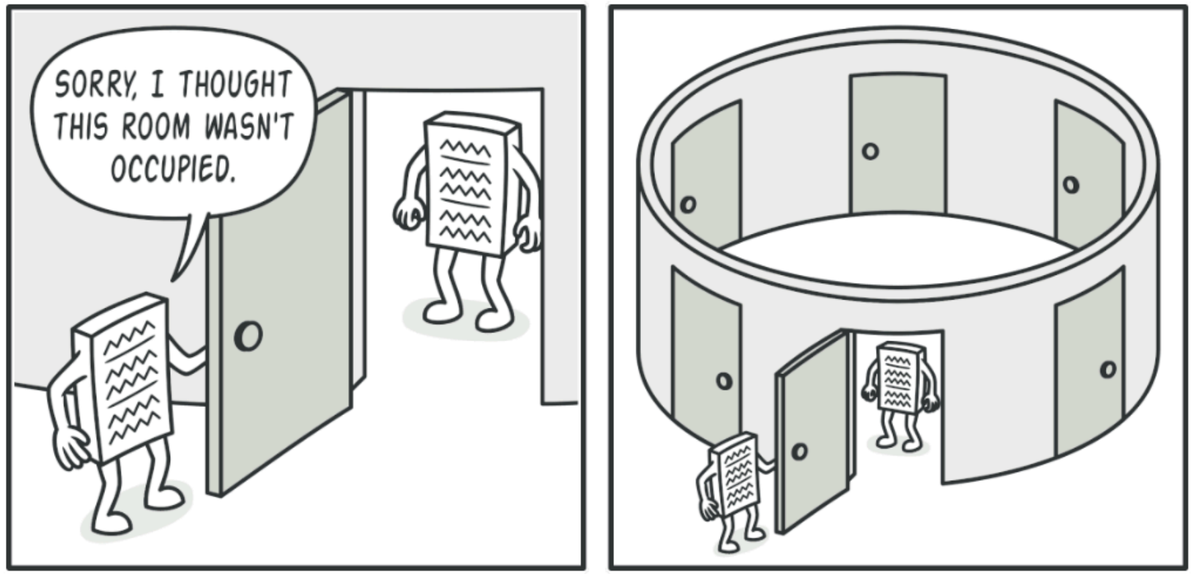


Singleton

En klasse som bare kan ha en enkelt instans av seg

Mønstre handler om å passe på at det finnes en, og kun en instans av en gitt ting

Eksempel
Dersom vi vil opprette en tilkobling til en database i et program



Clients may not even realize that they're working with the same object all the time.

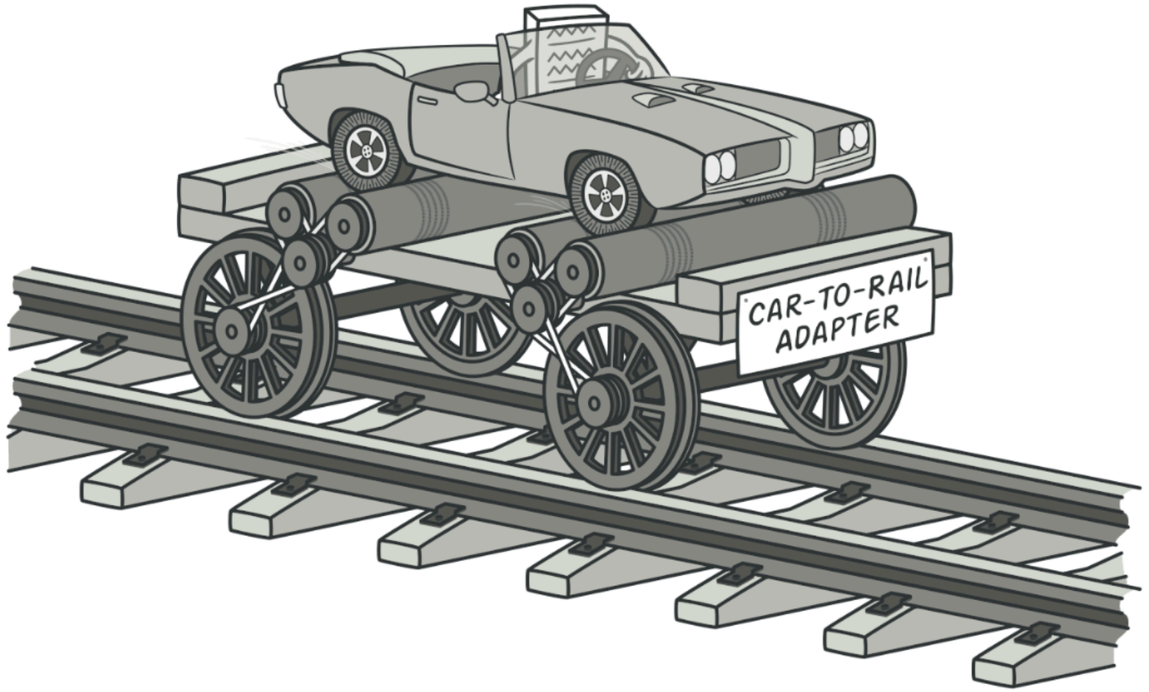
Structural

Hvordan objekter relaterer til hverandre.

Hvordan man kan strukturere klasser og objekter for å løse gitte former for problemer.

Eksempel
Vi har sett på forskjellige wrapper klasser i Java, som lar primitive typer som int og boolean samarbeide med objekter

Her er det mønster som:
Adapter, **decorator**, composite, facade, proxy, flyweight, bridge

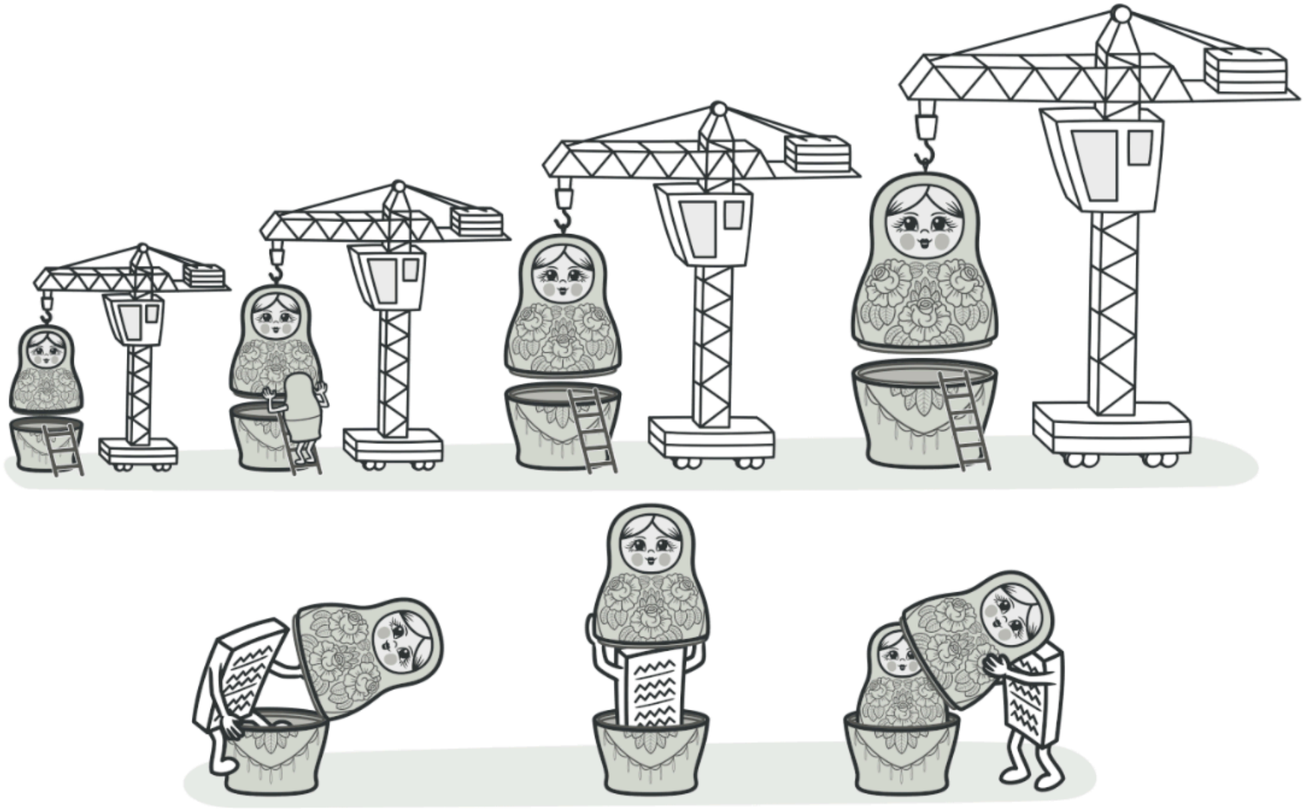


Decorator

Et objekt får tilført egenskaper eller fjernet egenskaper under kjøring

eksempel
produkter med tilvalg

- festivalbillett med VIP-tillegg



Med arv kan du ikke endre en oppførsel til et eksisterende objekt “in runtime”, man kan bare erstatte et helt objekt med et annet

Oppsummert

1. **Designmønster handler om velkjente måter å løse vanlige problemer på**

2. **Når man har programmert en stund vil man kunne begynne å kjenne igjen problemer man har støtt på før. Da kan det være lurt å gjenbruke løsninger fra tidligere erfaring.**

3. **Blir mere relevant i for eksempel IN2000, så ikke glem de!**

4. **Refactoring.guru**

   https://refactoring.guru/design-patterns

Ukas shoutout

Chèlbè -Lou Phelps

produsert av KAYTRANADA

box fresh, kom ut på fredag



Oppgaver

