Uke 11 – programeringsmønstre

in1010 - gruppe2

8 april 2025

this week>

- Litt om programmeringsmønstre
  - Behavioral
  - Creational
  - Structural
- Seminaroppgaver / innlevering 5

Hva er programmeringsmønstre?

- Programmeringsmønstre eller design patterns er **gjenbrukbare måter** å strukturere programmer på.
- De ulike typene har sine **fordeler og ulemper**, avhengig av målet vårt.

De forskjellige programmeringstypene:

- **Behavioral** - hvordan disse objektene kommuniserer og samhandler med hverandre
- **Creational** - kontroll over hvordan objektene lages
- **Structural** - hvordan man strukturer klasser og objekter

Behavioral: iterator

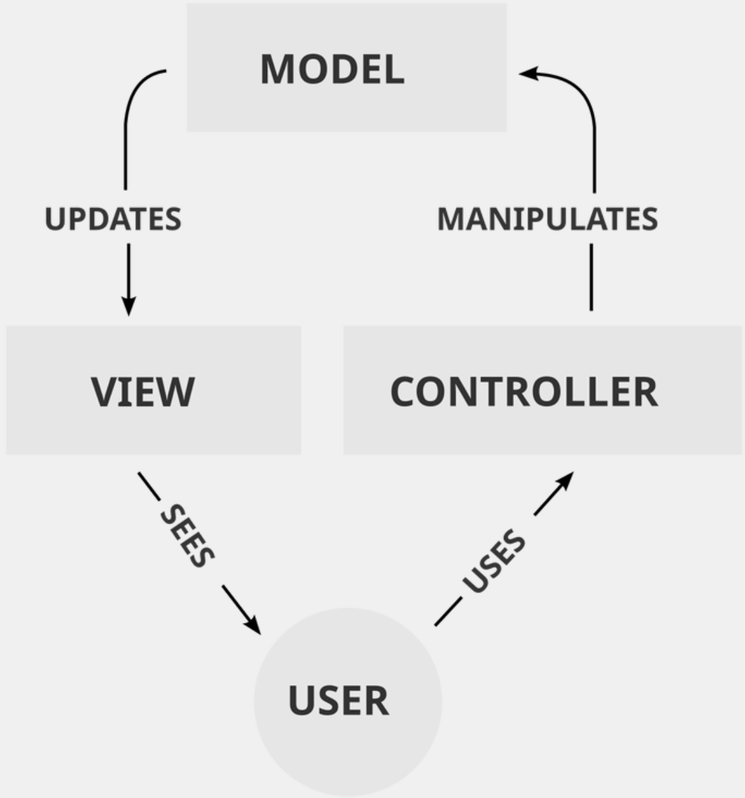
- Iterator
  - En metode for å navigere gjennom en samling av objeketer.
  - Bruker hasNext(), next()

```java
import java.util.ArrayList; import java.util.Iterator; import java.util.List;
public class IteratorExample {
    Run | Debug
    public static void main(String[] args) {
        List<String> list = new ArrayList<>();
        list.add(e:"One"); list.add(e:"Two"); list.add(e:"Three");

        Iterator<String> iterator = list.iterator();
        while(iterator.hasNext()) {
            System.out.println(iterator.next());
        }}}
```

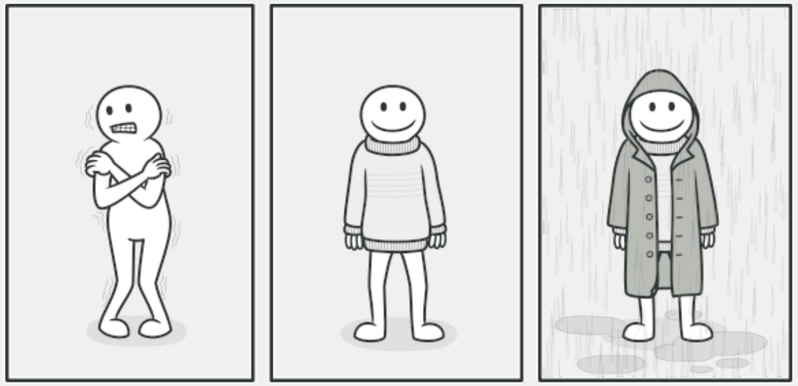
Behavioral: observer

- Observerer endringer til andre objekter.
- Den blir oppdatert når den merker en endring på objektet.
- Eksempel:
  - MVC - model-view-controller
    - View observerer model
    - Hver endring i model, så oppdateres view seg
  - Nyhetsoppdatering
    - Hver gang det kommer en nyhet, så får følgerne oppdatering på sosiale medier



Structural: Decorator

- Legge til eller fjerne funksjonalitet i koden uten å måtte rekonstruere koden
- Hver decorator har en referanse til objektet de decorer til.
- Eksempel
  - Produkter med tilvalg
  - Når du bestiller en kaffe, kan du velge om du vil legge til ekstra ting i kaffen som melk, sukker, smakstilsetning etc



eksempel på en decoratør

```java
// A ConcertTicket implementation
class BasicConcertTicket implements ConcertTicket {
    @Override
    public String getDescription() {
        return "Concert ticket";
    }

    @Override
    public double getCost() {
        return 450.0;
    }
}


// The ConcertTicket decorator
abstract class ConcertTicketDecorator implements ConcertTicket {
    private final ConcertTicket decoratedTicket;

    public ConcertTicketDecorator(ConcertTicket decoratedTicket){
        this.decoratedTicket = decoratedTicket;
    }

    public String getDescription() {
        return decoratedTicket.getDescription();
    }

    public double getCost() {
        return decoratedTicket.getCost();
    }
```

lagrer instans av klassen den decorater til

Tillegg funksjonene som blir lagt til

Creational: singleton

- Kun en instanst av en klasse.
- Kontrollere tilgang til sikkerhetskritiske ressurser,
  - f.eks. en database
- Begrense ressurskrevende objekter til én instans,
  - f.eks. sentralisert logging av ønskede og uønskede hendelser under programkjøring

```java
public final class LoggerSingleton {
    // Single instance
    private static LoggerSingleton instance = null;
    // Prevent instantiation from other classes by making constructor private
    private LoggerSingleton() {}


    // Method to get the single instance
    public static LoggerSingleton getInstance() {
        if (instance == null) instance = new LoggerSingleton();
        return instance;
    }
```

Seminaroppgaver

eller Innlevering 5