IN1010 uke 9

Gruppe 2

Agenda

- Menti
- Litt om designmønstre
- ☕ Pause 13:00-13:15

- Jobbe med oppgaver/trix/oblig!

Menti

Designmønstre

Iterator-mønsteret

- 2 metoder i Iterator: hasNext() og next()
- Brukes for å gå gjennom array/liste

```java
ArrayList<String> list = new ArrayList<>();
list.add(e:"En"); list.add(e:"To"); list.add(e:"Tre");

// ALTERNATIV 1
Iterator<String> iterator = list.iterator();
while(iterator.hasNext()) {
    System.out.println(iterator.next());
}

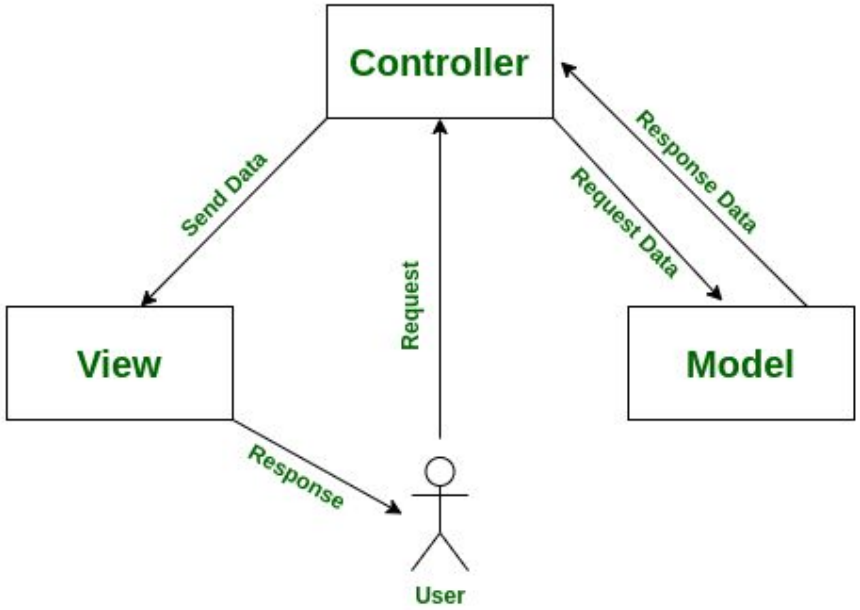
// ALTERNATIV 2
for (String e : list) {
    System.out.println(e);
}
```

Enkleste måten å bruke:
For-each loop vil automatisk bruke iteratoren for å gå gjennom (kaller på hasNext() og next())

Observer-mønsteret

- Observatør-objekter observerer subjekt-objekter & får beskjed om endringer

- Eksempel: Model-View-Controller (MVC)
- Eksempler på bruk: nyhetsfeed, følgere får oppdateringer i sosiale medier



Decorator-mønsteret

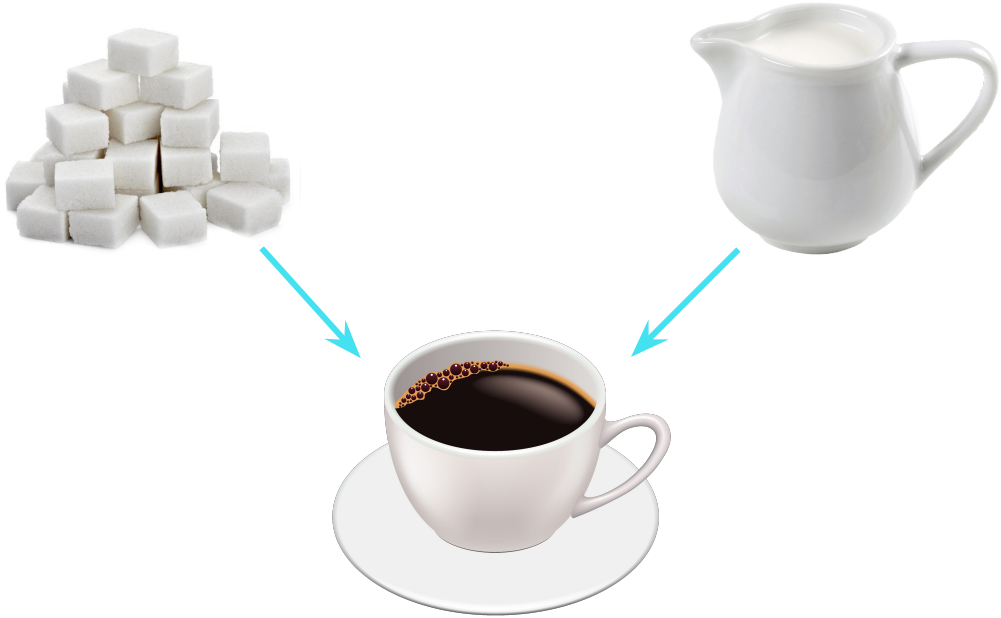
- Legger til/fjerner egenskaper til et objekt under kjøring

- Eksempler på bruk: produkter med tilvalg, f.eks. kafé eller takeaway

- Kan lage decorator for hver ekstra-ting, og så legge på disse som man vil (“wrapper” objekter i decorators)

Decorator-mønsteret

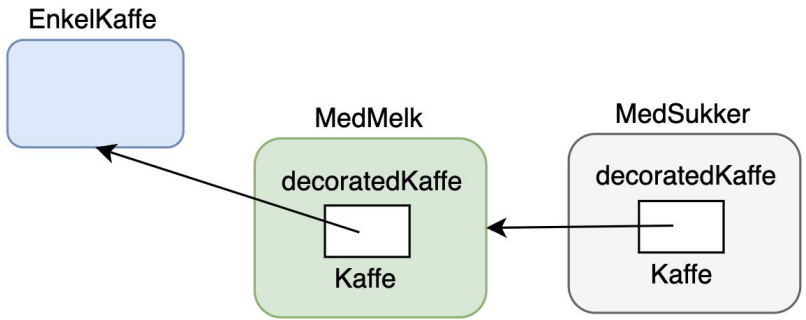
- ***Kodeeksempel***: Kaffe med ulike tilvalg

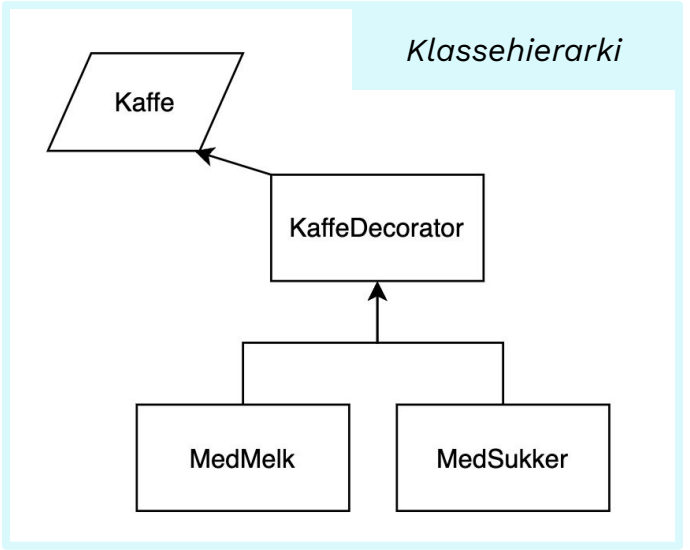


Se gruppesiden for uke 9 for hele koden

Decorator-mønsteret

Fortsetting på kaffe-eksempelet






```
Kaffe enkelKaffe = new EnkelKaffe();
Kaffe melkKaffe = new MedMelk(enkelKaffe);
Kaffe sukkerMelkKaffe = new MedSukker(melkKaffe);

System.out.println(sukkerMelkKaffe.hentBeskrivelse()
                   + " Pris: " + sukkerMelkKaffe.pris() + " kr");
```

Singleton-mønsteret

- Singleton = ett eneste objekt av en gitt type i programmet

- Eksempler på bruk:
  - 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

Singleton-mønsteret

Eksempelkode: Singleton brukt til terminal-logging

```java
public final class LoggerSingleton {
    // Nøyaktig én instans
    private static LoggerSingleton instance = null;
    // Privat konstruktør -> kan ikke lage nytt objekt i noen andre klasser
    private LoggerSingleton() {}


    // Metode for å lage nøyaktig en instanse av objektet, om ikke laget før
    public static LoggerSingleton getInstance() {
        if (instance == null) instance = new LoggerSingleton();
        return instance;
    }
    // Metode som logger meldinger i terminalen
    public void logMessage(String message) {
        System.out.println("Log Message: " + message);
    }
}
```

Jobb med oppgaver!

IN1010 Emnesiden → Grupper →
Gruppe 2 → Uke9

Husk å ta pause ☕