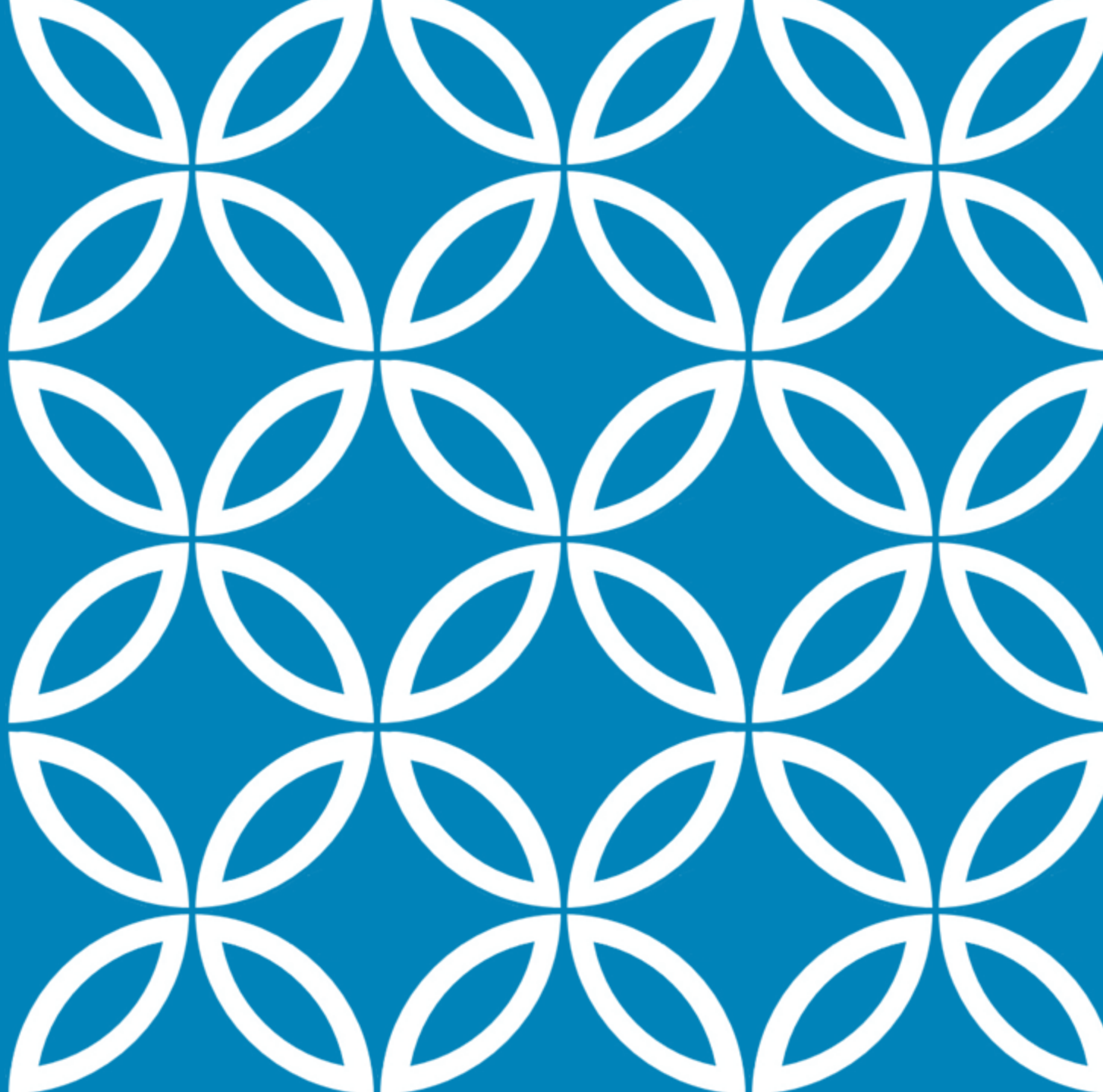
DESIGNMØNSTRE

agneshb@uio.no



I DAG

Ulike designmønstre:

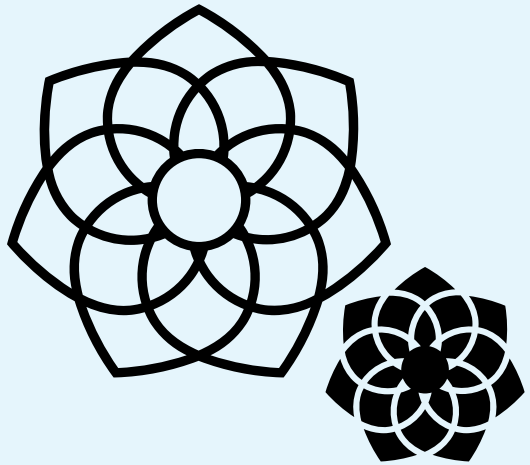
- Iterator
- Observer
- Singleton
- Decorator

Ser på kodeeksempler på disse

DESIGNMØNSTRE

Gjenbrukbare måter å strukturere programmer på

Forskjellige mønstre har fordeler og ulemper avhengig av hva vi ønsker å oppnå i programmene våre.



ITERATOR

Iterable-grensesnittet med metodene hasNext() og next()

Eksempel: Iterator-mønsteret med kall på iterator, hasNext og next

Eksempel: skrive ut array- eller listeelementer med for-løkke

```java
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());
}
```

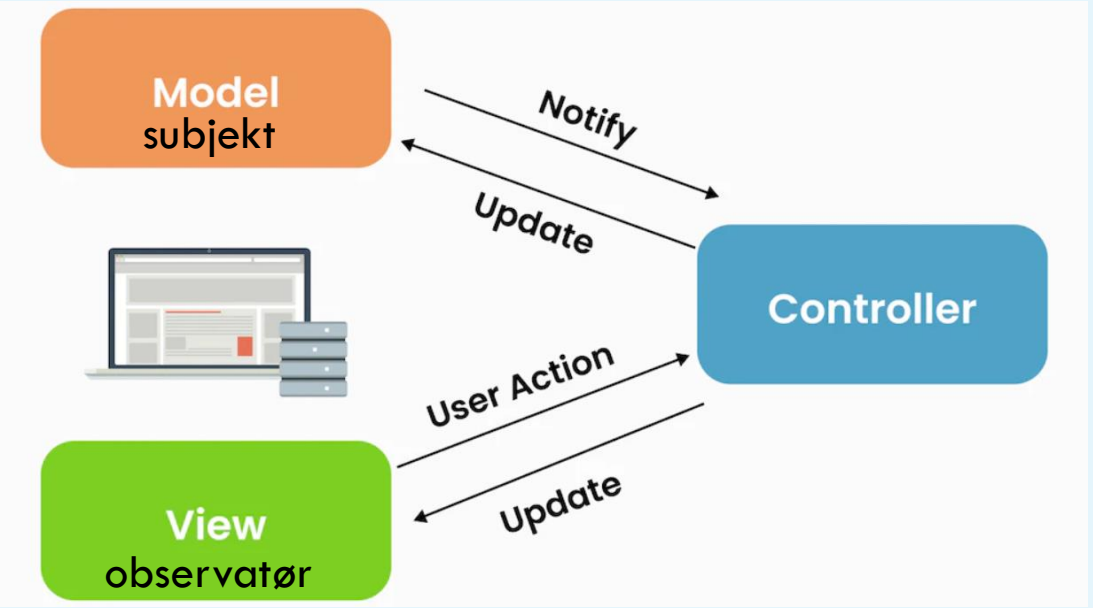
```java
for (String string : list) {
    System.out.println(string);
}
```

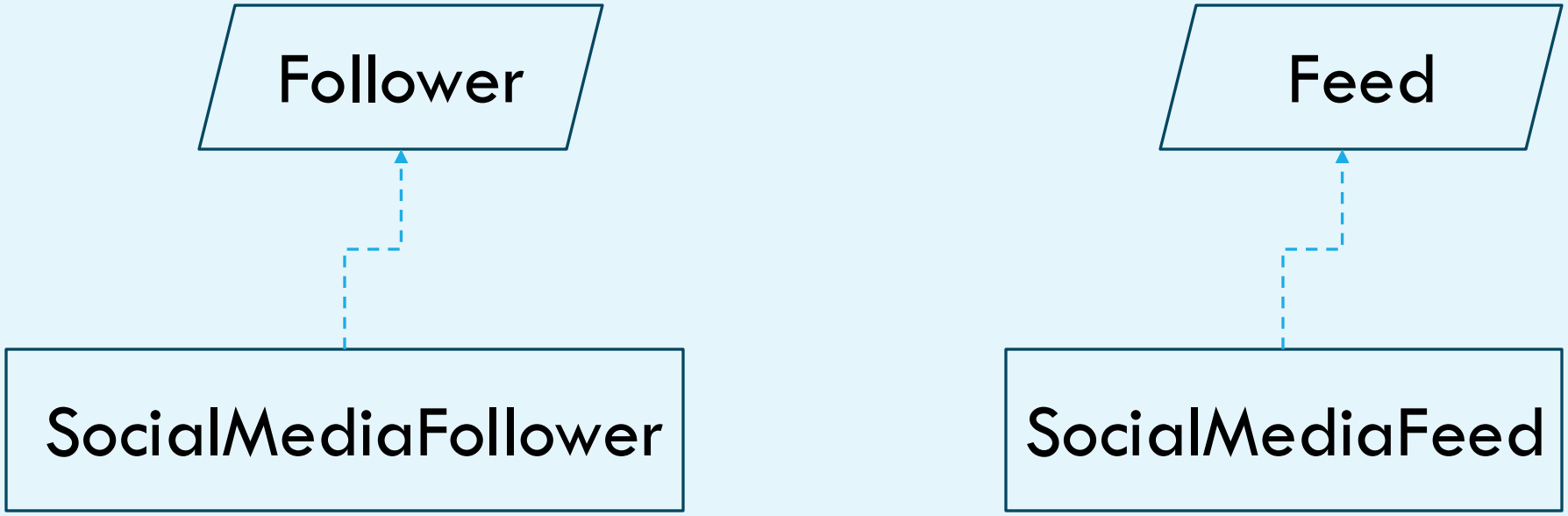
OBSERVER

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

Model-View-Controller: View endres når Model endres (MVC brukt i GUI)

Eksempler på bruk: nyhetsfeed, følgere får oppdateringer i sosiale medier



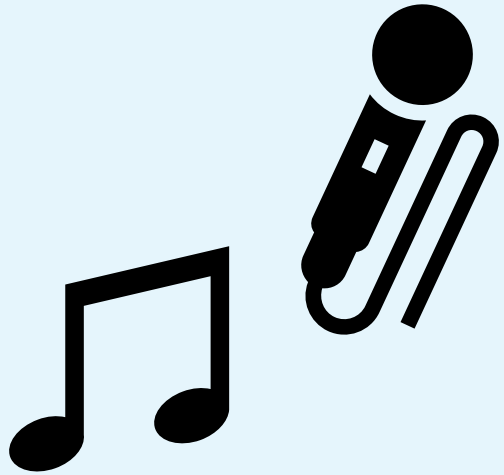

Follower
SocialMediaFollower
Feed
SocialMediaFeed

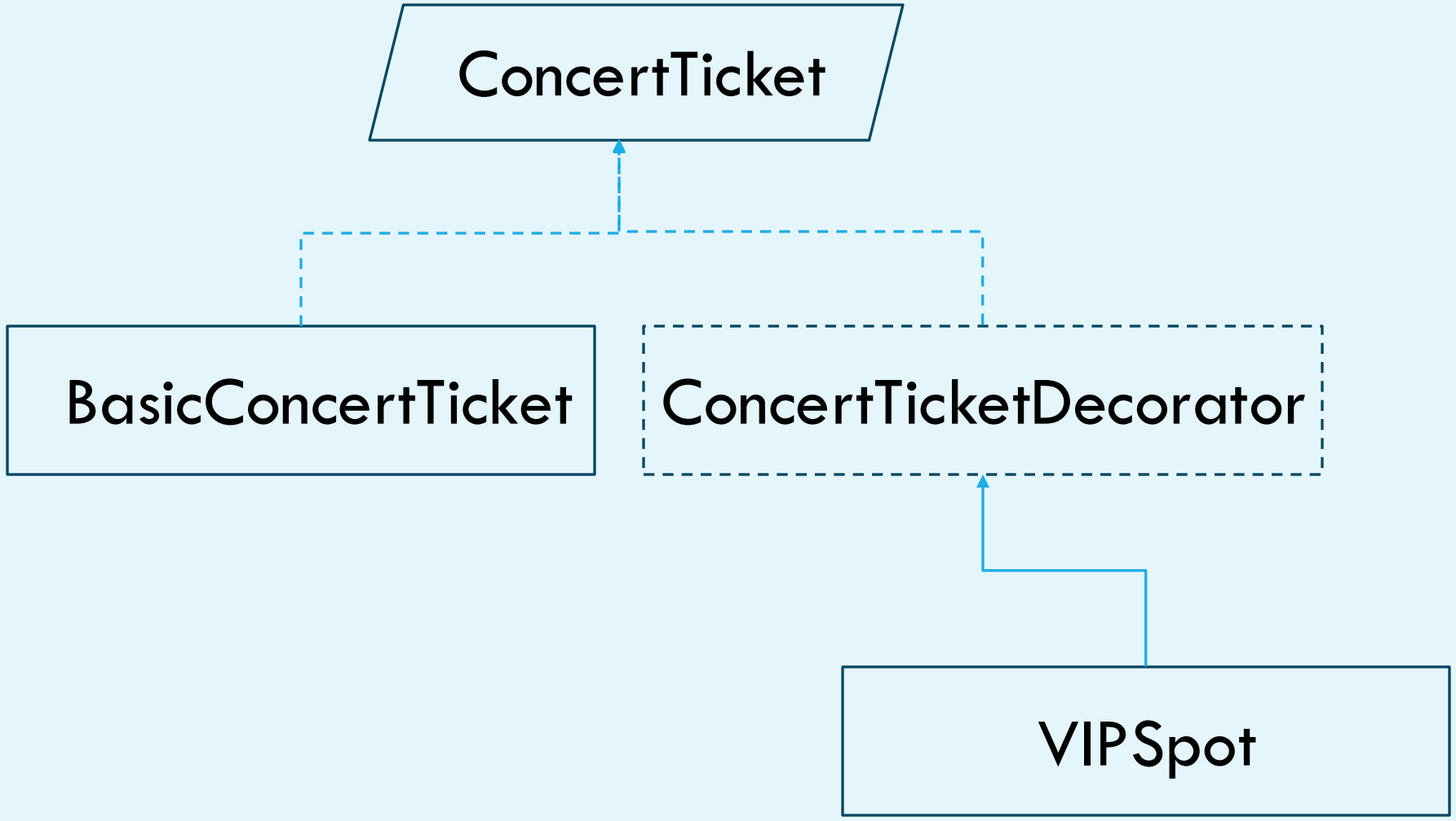
DECORATOR

Et objekt får tilført egenskaper eller fjernet egenskaper under kjøring

Eksempler på bruk:

- produkter med tilvalg, f.eks. kafé eller takeaway
- konsertbillett dekorert med VIP-tilleggPizza som man kan bestille med pepperoni eller ekstra ost




ConcertTicket
BasicConcertTicket
ConcertTicketDecorator
VIPSpot

SINGLETON

Ett eneste objekt av en gitt type i programmet

Kontrollere tilgang til sikkerhetskritiske ressurser

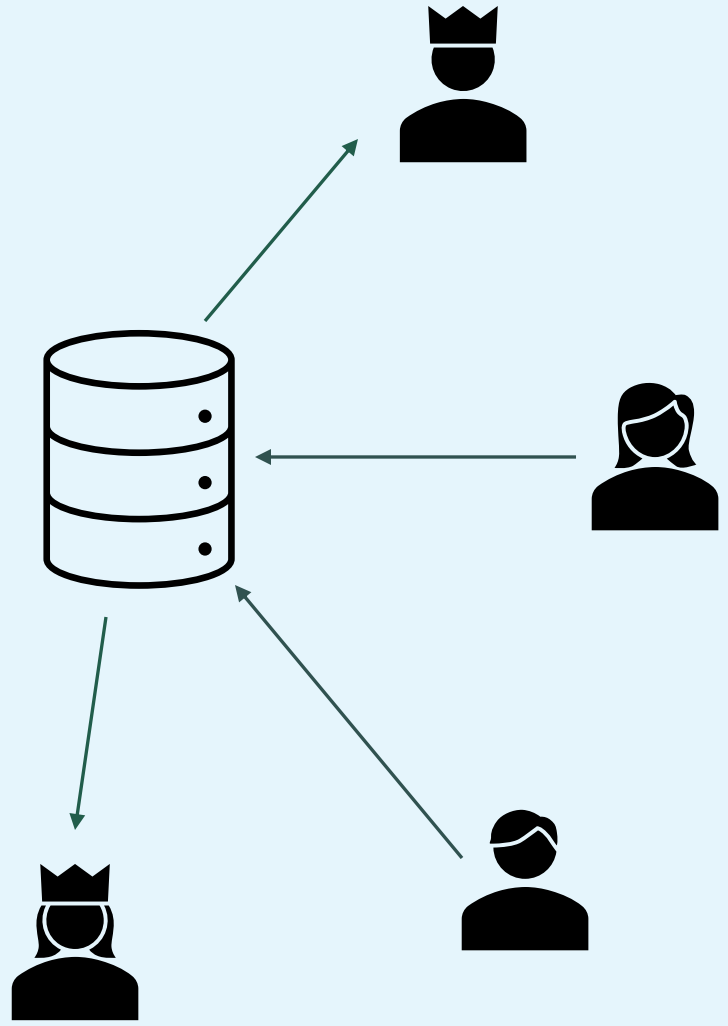
- Eks: database, bank

Begrense ressurskrevende objekter til én instans

- Eks: sentralisert logging av hendelser under programkjøring

Global tilstand i programmet

Trådsikkerhet er ofte en forutsetning når vi bruker Singleton-mønsteret



EKSEMPEL SINGLETON

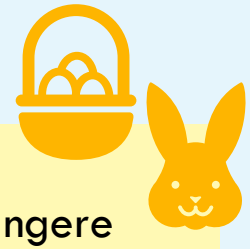
```java
public final class LoggerSingleton {
    // Single instance
    private static LoggerSingleton instance = null;

    // 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;
    }

    // Method to demonstrate Singleton usage logging messages to the terminal
    public void logMessage(String message) {
        System.out.println("Log Message: " + message);
    }
}
```

JOBBE MED OPPGAVER



God påske!
Ingen forelesningere eller gruppetimer neste uke

Seminaroppgaver ligger på semestersiden → grupper → gruppe 3 → uke 11