Designmønstre

Gruppetime 9

agneshb@uio.no

I dag

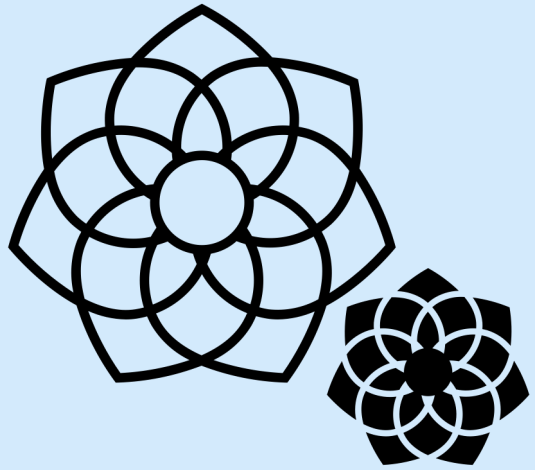
Ulike designmønstre:

- 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()
- Eksempler på bruk: skrive ut array- eller listeelementer med for-løkke
- Eksempelkode: Iterator-mønsteret med kall på iterator, hasNext og next

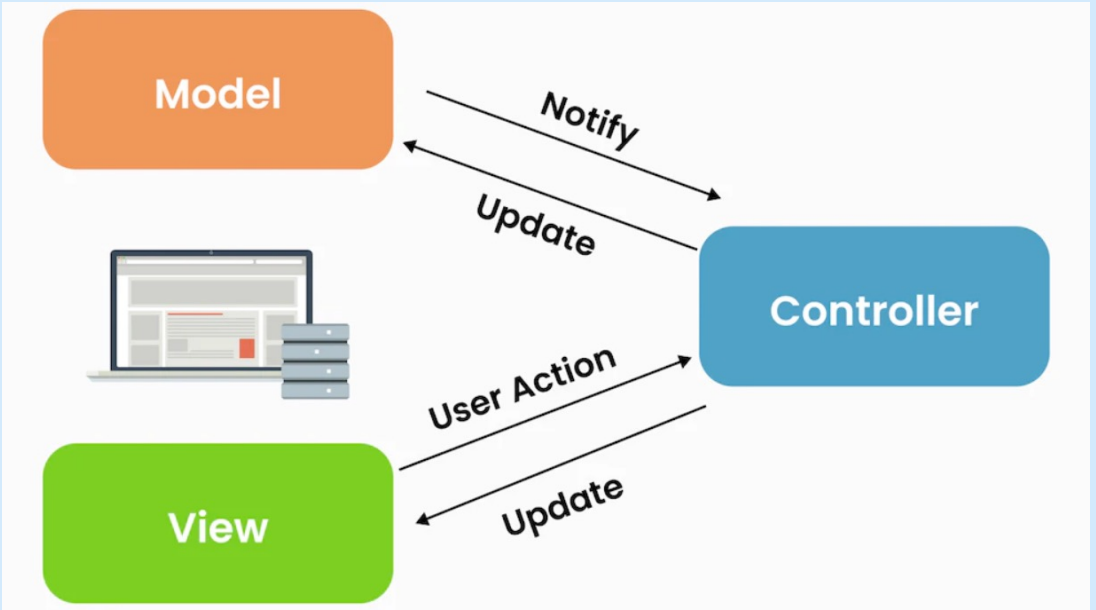
```java
List<String> list = new ArrayList<>();
list.add(e:"One");
list.add(e:"Two");
list.add(e:"Three");

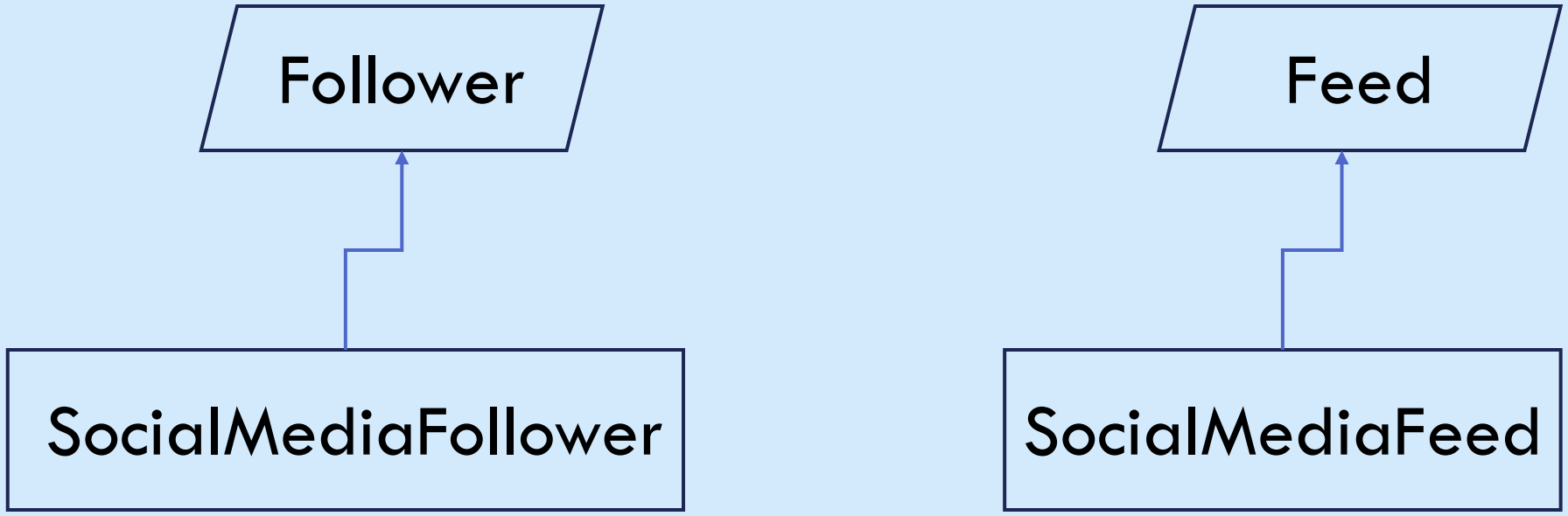
Iterator<String> iterator = list.iterator();
while(iterator.hasNext()) {
    System.out.println(iterator.next());
}
```

```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



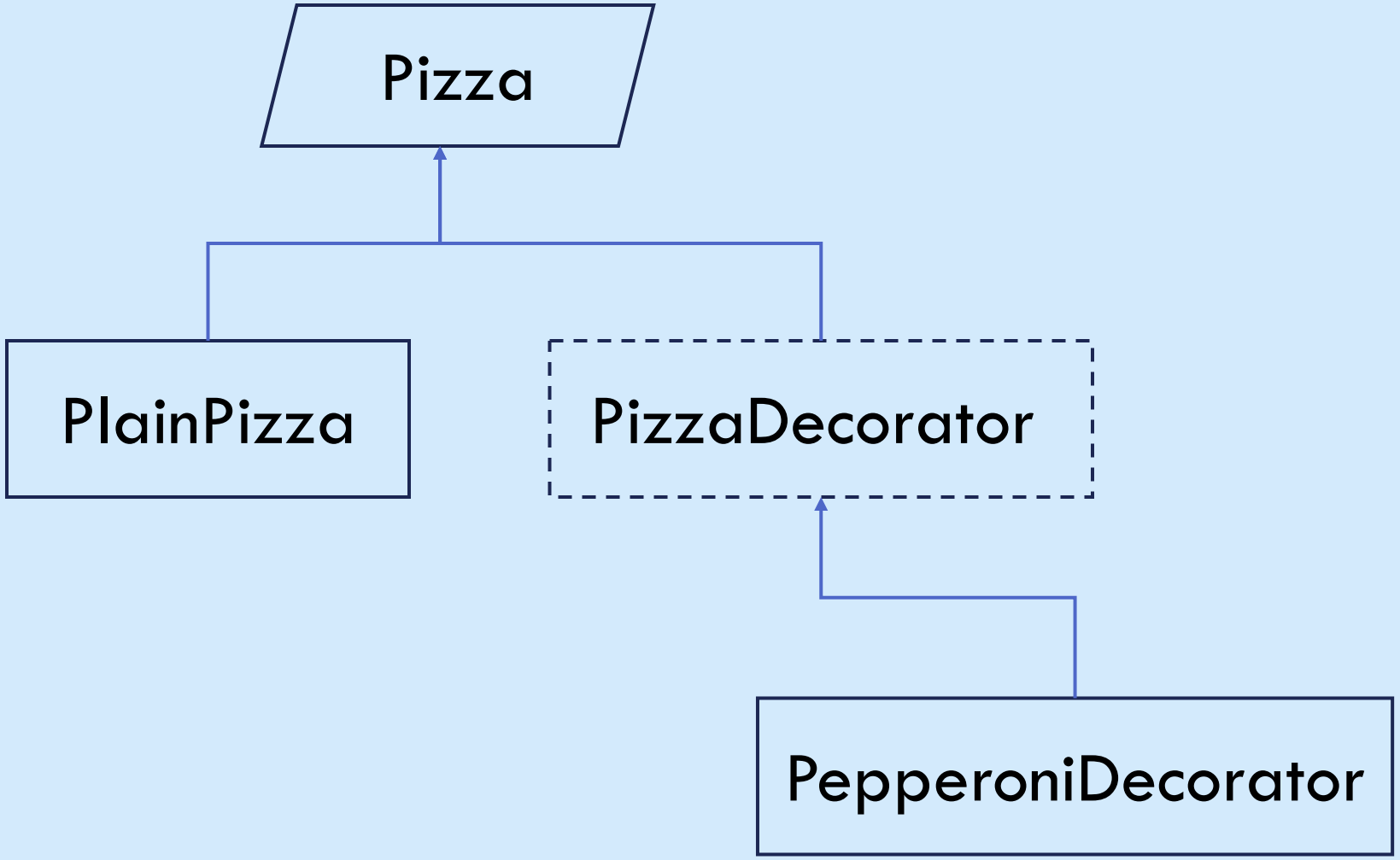

Follower
SocialMediaFollower
Feed
SocialMediaFeed

Decorator

- Et objekt får tilført egenskaper eller fjernet egenskaper under kjøring
- Eksempler på bruk:
  - produkter med tilvalg, f.eks. kafé eller takeaway
  - Pizza som man kan bestille med pepperoni eller ekstra ost

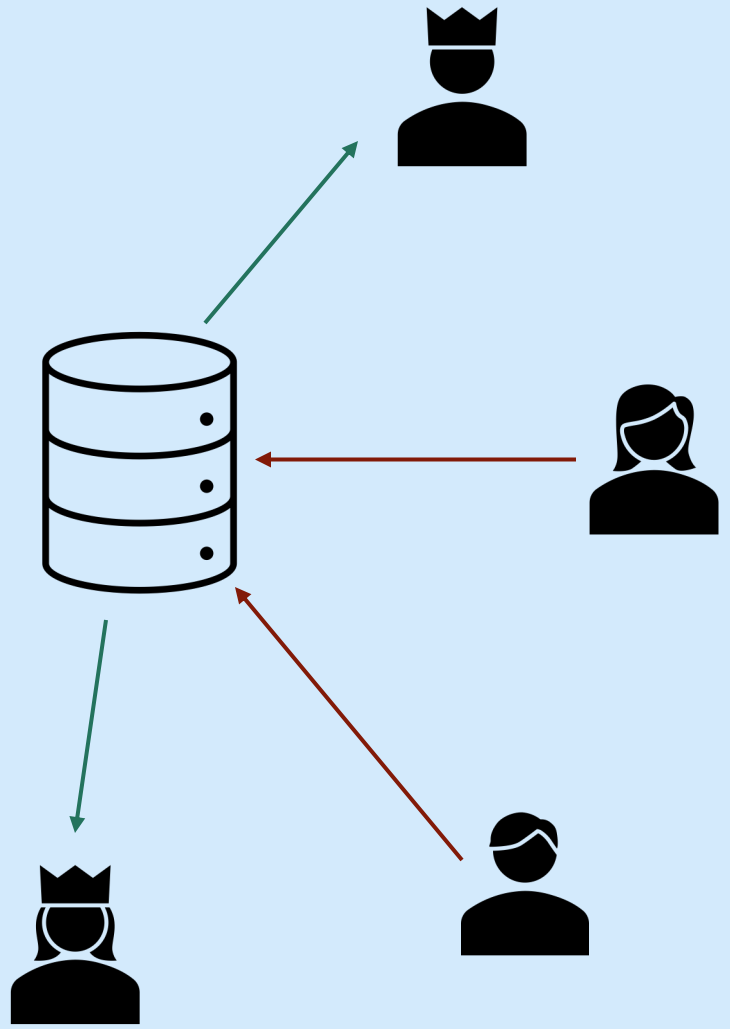





Pizza
PlainPizza
PizzaDecorator
PepperoniDecorator

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
  - Kommer tilbake til dette



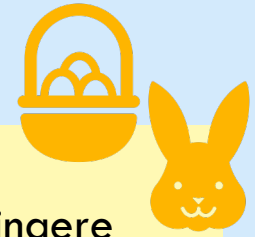
Eksempel Singleton

```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);
    }
}
```



God påske!
Ingen forelesningere eller gruppetimer neste uke

Jobbe med oppgaver

Innlevering med rekursjon

Seminaroppgaver ligger på semestersiden → grupper → gruppe 3 → uke 9