Uke 11

Designmønstre

Kort om designmønstre

- Handler om gjenbruk av kode, og få struktur
  - Unngår å pønske ut ny løsning hver gang
- Iterator, decorator, observer og singleton
- Kategorier
  - Behavioural, creational og structural

Iterator

- Beholder har iterator metode som gir iterator kan brukes til å iterere over beholderen
  - next() og hasNext()

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

        Iterator<String> iterator =
list.iterator();
        while(iterator.hasNext()) {

System.out.println(iterator.next());
        }}}
```

Observer

- Observatørobjekter og subjektobjekter
- Observatør får beskjed om endringer
- Mange observatører med liten kobling

Decorator

- Tilført og fjernet egenskaper under kjøring
  - Istedenfor arv som er statisk
- Sender inn objekt i konstruktøren som tar med seg tidligere egenskaper

Singleton

- Ett eneste objekt
  - F.eks sikkerhetskritisk ressurs f.eks database
  - Begrenset ressurs som gjør at den må være sentralisert
- Fører til global tilstand og mye kobling

```
public final class LoggerSingleton {
    private static LoggerSingleton instance =
null;
    private LoggerSingleton() {}

    // Method to get the single instance
    public static LoggerSingleton getInstance() {
        if (instance == null) instance = new
LoggerSingleton();
        return instance;
    }
    public void logMessage(String message) {
        System.out.println("Log Message: " +
message);
    }
}
```

Oppgaver / Trix