Programutvikling: Obligatorisk 5, Designmønster

1) Kort sagt: Designmønster er objekt-orienterte løsninger på problemer som man ser at oppstår ofte. Designmønstre brukes mye i programutvikling, og vi går igjennom de mest brukte mønstrene.

2) Ved bruk av private konstruktør kan man ikke arve fra klassen!

3) STATE OG STRATEGY
Abstrakte metoder er måten man løser disse problemene på. Man lager en abstrakt klasse og implementerer dens abstrakte metoder inn i subklassene.

State: kunne tilby flere oppførsler for samme type objekt basert på tilstand.
Kan ha interface eller abstrakt klasse. Andre klasser implementerer den abstrakte metoden.

Strategy: kunne velge en spesifikk løsning på et problem ved kjøretid.
Kunne tilby flere typer lister, hvor man velger akkurat hvilken type liste ved kjøring av programmet.
Implementerer et interface med en algoritme. Algoritmen implementeres i forskjellige klasser og skal løse problemet.

Eksempel på State:
Vi har en brødskive, som ut i fra hvilket pålegg den får på seg har ulike tilstander.

```
public abstract class Brødskive{
        private String brødtype;

        public brødskive(String brødtype){
                this.brødtype = brødtype;
        }

        public abstract void taPåPålegg();
}
```

Eks. på en tilstand

```
public class syltetøy extends Brødskive{
        public syltetøy (String brødtype){
                super(brødtype);
        }
        public void taPåPålegg(){
                System.out.println «Brødskiven har på syltetøy»;
        }
}
```

Eksempel på Strategy:
Man har samtlige parametere man skal få inn og som skal sorteres. Men man vet ikke om det kommer inn tall eller bokstaver. Dersom det kun kommer inn bokstaver skal det sorteres etter bokstaver, dersom det kun kommer inn tall skal det sorteres etter tall. Da kan man bruke strategy til å ha to ulike klasser: der den ene sorterer alfabetisk, den andre etter kronologisk.

4)

Nyhetsbrev/Subjekt:

I konstruktøren: studenter = new ArrayList <Student>();
addObserver(Student o)
removeObserver(Student o)
setNews
notifyObserver(Student o)

Student/Observer:
public void update()

5) Composite lager en trestruktur. Øverst i trestrukturen har vi klassen Component, som igjen inneholder en abstrakt klasse "operation". Operation gjør en operasjon som er felles for alle objektene i dette systemet. En klasse kan da arve fra Composite, og kan ha barn. Barna er av typen component og blir lagt i en liste. Med metoden operation nå kan man løpe igjennom barna som et objekt. JavaFX bruker composite til å lage det som kalles scenegraf, som er en trestruktur med ulike noder.

6)