Cupcake projekt

Lars Peter Jørgensen

l-dat-da 0921a - Sem2

CPHBusiness Lyngby

Indholdsfortegnelse

Indledning

Projektets formål er at bruge den ny lærte viden om både front-end og back-end webudvikling, til at lave en semi-funktionel webshop for en fiktiv kunde. Kunden vil gerne have en webshop hvor en bruger kan bestille cupcakes med frit valg af topping og bund. Flere krav til webshoppens funktionalitet bliver stillet med *userstories*.

Baggrund

En virksomhed som heder Olsker Cupcakes, vil gerne have en webshop hvor kunderne kan bestille cupcakes, og virksomheden kan håndtere ordrerne. Kunden skal kunne oprette en konto for at kunne gemme ordre og betale for dem. Kunderne skal selv kunne vælge topping og bund på deres cupcakes. Ved et kundemøde har man fundet nogle *userstories* som beskriver nogle forskellige funktionaliteter webshopen skal have.

Teknologi valg

Java er anvendt som projektets programmingssprog og er brugt sammen med Maven til bl.a. at downloade nogle dependencies. Projektet er designet efter MVC strukturen. Jsp filer er blevet brugt som templates der bliver udfyldt med brugerens data, hver jsp fil her en korresponderende servlet class. Der er nogle ting som skal lagres i en database, bl.a. brugernes ordre, og til det bliver MySQL brugt.

Java servlets er en dependency som gør det nemt at lave webapplikationer med Java. Controlleren er delt op i *servlets* som har forskellige funktioner, baseret på hvilken side man ender på.

Tomcat er blevet brugt, som er en implementering af servlets, der her i projektet bliver brugt til at køre programmet som en web server.

Git er blevet brugt til at klone startkode repositoriet.

PlantUML er blevet brugt til at lave diverse diagrammer over projektet.

Paint.NET er blevet brugt til at lave nogle få justeringer på banneret, f.eks. en højere resolutions genskabelse af cupcake ikonet som var på banneret, og cupcake ikonet er rykket så der var plads til mere tekst.

Til websidens design er der blevet brugt Bootstrap 5.

Krav

Funktionelle krav

Sammen med kunden er der blevet udviklet nogle userstories, der er vist her.

1. Som kunde kan jeg bestille og betale cupcakes med en valgfri bund og top, sådan at jeg senere kan køre forbi butikken i Olsker og hente min ordre.

2. Som kunde kan jeg oprette en konto/profil for at kunne betale og gemme en ordre.

3. Som administrator kan jeg indsætte beløb på en kundes konto direkte i MySql, så en kunde kan betale for sine ordrer.

4. Som kunde kan jeg se mine valgte ordrelinier i en indkøbskurv, så jeg kan se den samlede pris.

5. Som kunde eller administrator kan jeg logge på systemet med email og kodeord. Når jeg er logget på, skal jeg kunne se min email på hver side (evt. i topmenuen, som vist på mockup'en).

6. Som administrator kan jeg se alle ordrer i systemet, så jeg kan se hvad der er blevet bestilt.

7. Som administrator kan jeg se alle kunder i systemet og deres ordrer, sådan at jeg kan følge op på ordrer og holde styr på mine kunder.

8. Som kunde kan jeg fjerne en ordre fra min indkøbskurv, så jeg kan justere min ordre.

9. Som administrator kan jeg fjerne en ordre, så systemet ikke kommer til at indeholde udgyldige ordrer. F.eks. hvis kunden aldrig har betalt.

Ikke-funktionelle krav

Udover de funktionelle krav så er der også blevet stillet yderligere krav til projektet:

1. Der laves en mockup i Draw.io eller lignende, som viser de websider den færdige løsning kommer til at bestå af.

2. Ordrer, kunder og øvrige data skal gemmes i en database.

3. Databasen skal normaliseres på 3. normalform med mindre andet giver bedre mening.

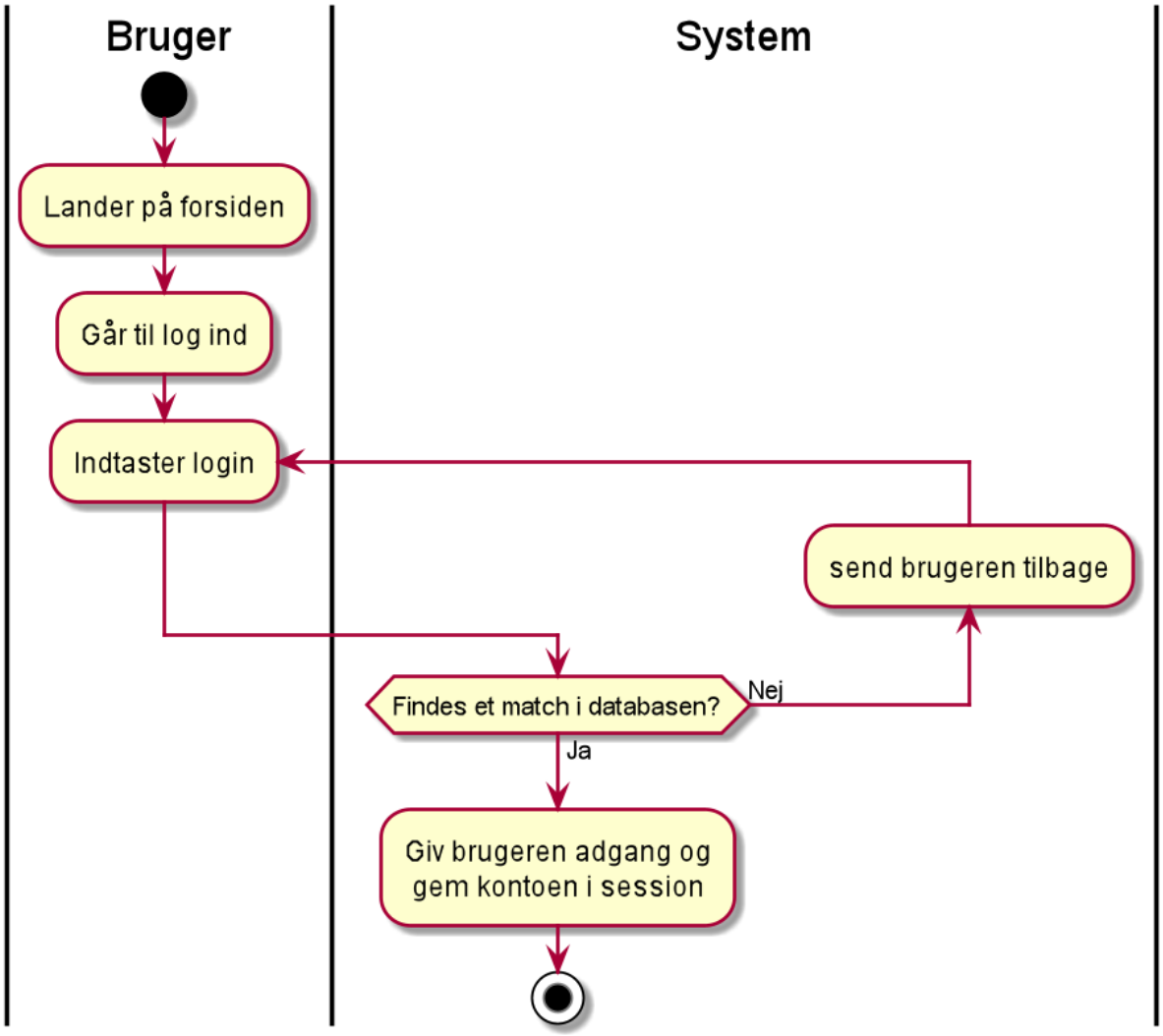
4. Kildekoden skal deles på GitHub.

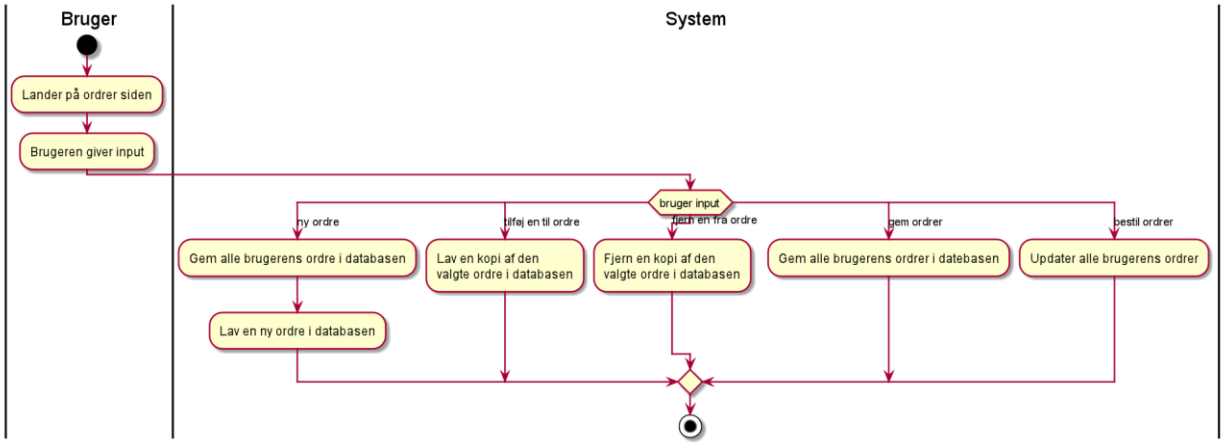
5. Det færdige produkt skal udvikles i Java, MySql, HTML, CSS, Twitter Bootstrap og køre på en Tomcat webcontainer.

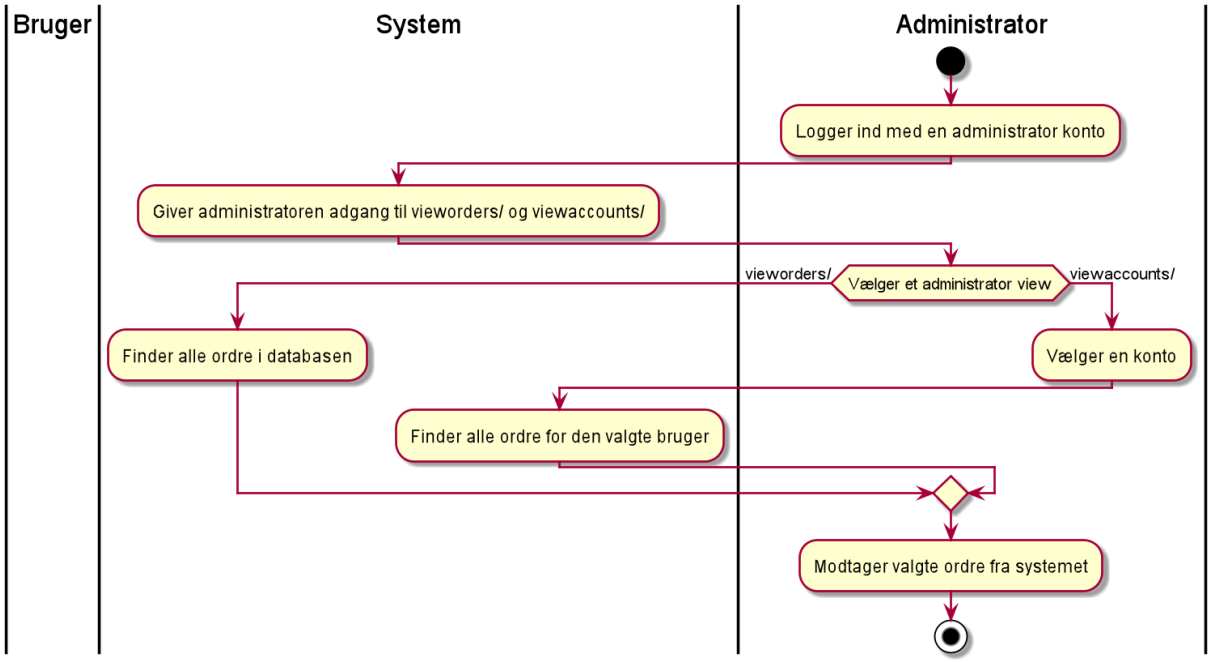
6. Løsningen skal udvikles med udgangspunkt i vores startkode.

Aktivitetsdiagram

I aktivitetsdiagramet er workflowet for forretningen dokumenteret. Man kan se at Systemet automatisere meget af arbejdet, og at brugeren ikke har en dialog med virksomheden men systemet.





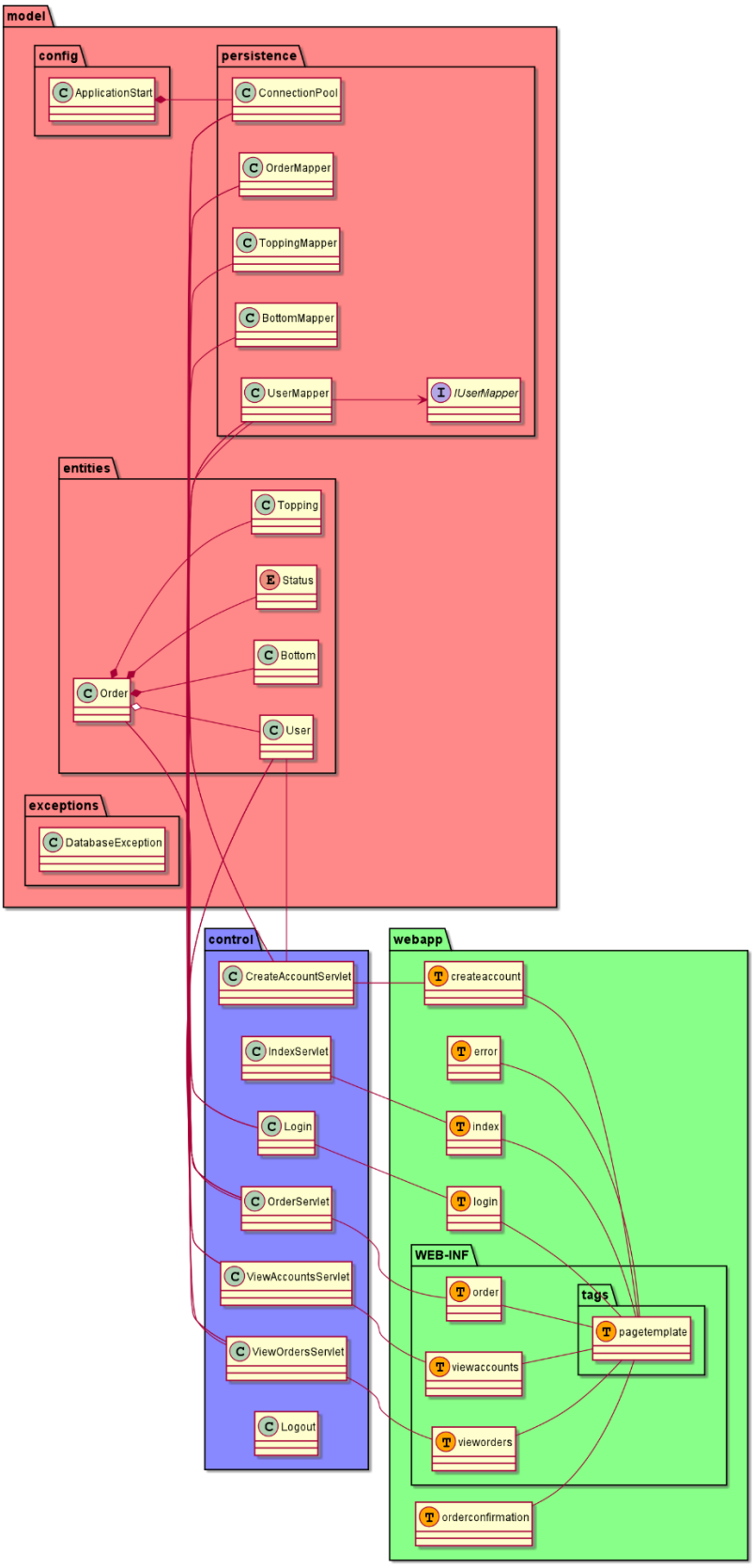


Domænemodel og ER diagram

Domænemodel

Projektet følger MVC strukturen, som også er repræsenteret i UML Domænemodelen. Det skal så siges at View'et er blevet navngivet *webapp*, men idéen er det samme. Domænemodelen er farvet for at nemmere kunne se opdelingen:

Rød til Model, Grøn til View, Blå til Control



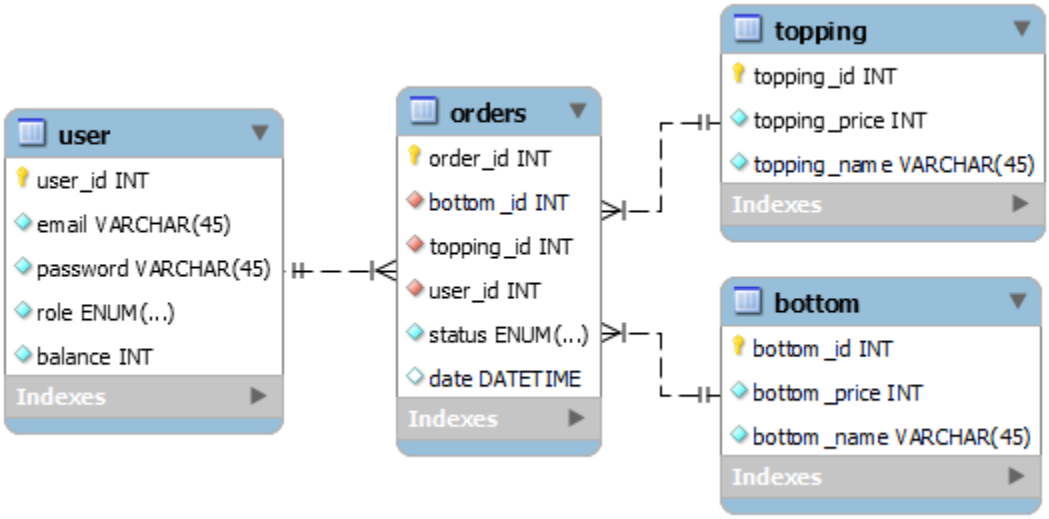
ER diagram

Database strukturen følger tredje normalform. En *order* bruger tre foreign keys fra tabelerne; user, topping og bottom. En *order* har også en status og en *date*.

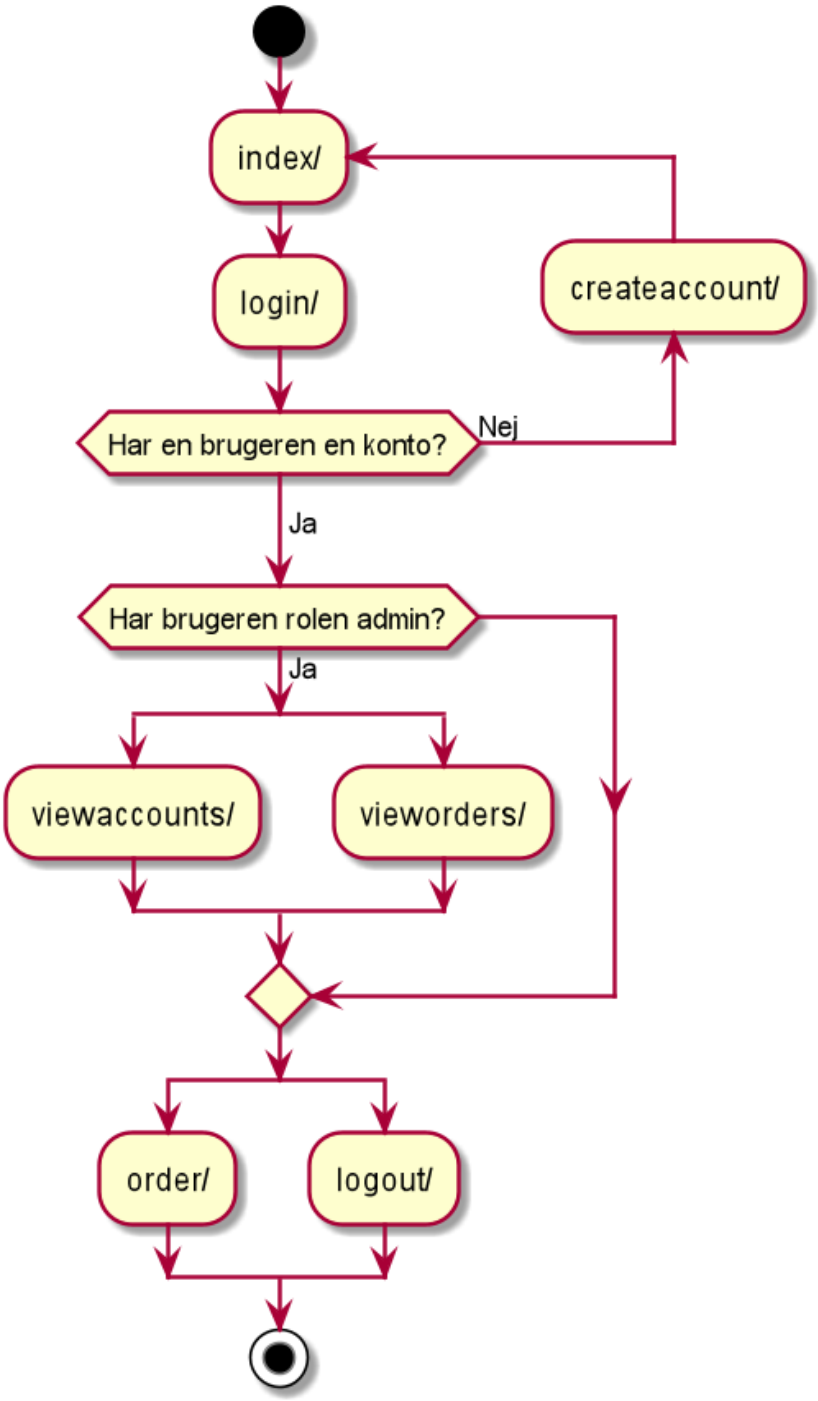
status er en enumerator, som indeholder CANCELLED, NOT_SUBMITTED, SUBMITTED, AWAITING_PICKUP og COMPLETED. Når ordrens status skal ændres så bliver *status* opdateret.

date viser datoen og tidspunktet for den sidste gang ordren fik blev ændret.

Tabellerne *topping* og *bottom* er næsten identiske, den eneste forskel er navnet og rækkerne de hver især indeholder.



Navigationsdiagram



Særlige forhold

Session

Når en bruger logger ind så bliver deres konto hentet fra databasen og gemt som et User objekt i session.

Der er ikke blev gjort noget for at øge sikkerheden i web applikationen.

Status på implementation

Alle diagrammer i det her dokument er blevet lavet som det sidste, så de repræsenterer alle den endelige udgave af projektet.

Bootstrap er blevet brugt til webshoppens design, og websiden er blevet testet og virker på små enheder. Det gjorde det meget nemt at lave et moderne design til websiden.

Plus og minus knapperne til ordrerne skulle have lavet og fjernet en kopi af en cupcake (man kan se det som antallet af et sæt cupcakes øges eller sænkes med en), men det blev ikke implementeret. Et workaround man kan bruge er at manuelt tilføje en ny cupcake og så ændre dens topping og bund til det samme som den man vil lave en kopi af, og så bliver den lagt sammen med den anden når brugeren gemmer ordrerne.

Der er nogle CRUD operationer som mangler i projektets mappers.

Der er ikke blevet lavet flere unittests andet end dem som tilhørte startkoden, dem skulle man nok have haft nogen flere af.

Der mangler nogle flere exceptions, der er kun DatabaseException og den beskriver ikke hvad der er gået galt, fordi det ikke er blevet specificeret ordentligt i koden. Exceptions bliver heller ikke ofte håndteret, men bare kastet videre.

Når man ”bygger” en cupcake så skal der være et billede af hvordan den kommer til at se ud. Der er kun et billede og det er når man vælger kombinationen ”Chocolate-Chocolate”. Det ville være bedst at man får virksomheden til at lave billederne, så derfor er der ikke blevet brugt tid på det.

Proces

Nogle metoder i de forskellige classes havde lidt for meget duplikeret kode, som man nok kunne have skåret ned på. F.eks. så er bottom.java og topping.java en nøjagtig kopi af hinanden, og man burde nok havet lavet en abstract class til dem.

Video Demo: https://youtu.be/dp_n20bRync