# Styklisten

### 1.0 – introduktion

Styklisten i vores projekt er en liste over hvor mange og hvilke materialer som skal bruges til at bygge den carport som en kunder gerne vil bestille. Genereringen af denne liste sker automatisk og processen for det går fra databasen, gennem backend og ud til frontend af vores program. Listen indeholder alt lige fra træ og bygge komponenter, tag elementer og de værtøjsdele som skal bruges for at sætte det hele sammen. Vores stykliste indeholder også den samlede omkostning og salgs pris for sit indhold.

### 1.1 – opbygning

Styklisten eksisterer som tre hashmap attributer på vores offerRequest klasse. Disse hashmaps bliver fyldt op med objekter af henholdsvis komponenter(Component), tagelementer(RoofUnit) og værtøjsdele(Part) i deres Map-key og antallet af denne, som integer, bliver sat i Map-value.

### 1.2 – opfyldning af lister

OfferRequest klassen indeholder et hav af metoder, hvis formål det er at tage imod diverse mål fra konfiguraionen og så ud fra den individuelle metode, skabe et objekt af den rigtige type, med de rigtige attributer og det rigtige antal. Objektet som skabes indsættes som sagt i den dertilhørende Hashmap liste, som Key-værdi og antallet af objektet som skal bruges, tilføjes som Value-værdi.

### 1.3 – data fra database

Når vi opretter objekter til vores stykliste, så bruger vi data fra databasen. Vi beder vores Mapper metoder på DBAccess laget om at hente denne data og sende os et objekt som vi kan tage fat i og bearbejde, inden vi putter det i vores hashmap liste.

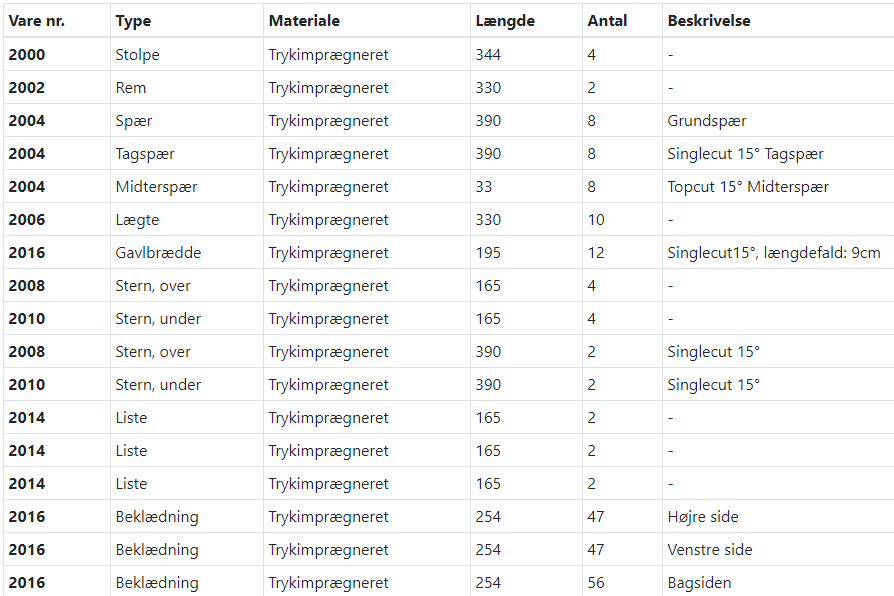
### 1.4 – data til hjemmeside

Hashmap attributterne på bestillingsobjektet(OfferRequest) er nu fyldt med objekter(hhv. Component, RoofUnit, Part). Når en ansat åbner en bestilling på hjemmesiden, vil vores front-end kigge listerne igennem og indsætte de specifikke data som styklisten skal indeholde.

### 1.5 – resultat

Denne proces gør det muligt at starte med en database record som ser sådan ud:

Og ende med et billede hos den ansatte som ser sådan ud:



## 1.6.0 – kodeeksempler

Her viser vi et par udvalgte kodestykker som viser hvordan dele af programmet fungerer.

## 1.6.1 – mapper metode

I denne kode har vi en metode i PartMapper klassen.

Metoden tager imod en string fra metodekaldet, som beskriver hvilken væktøjdsdel som vi vil have fat i.

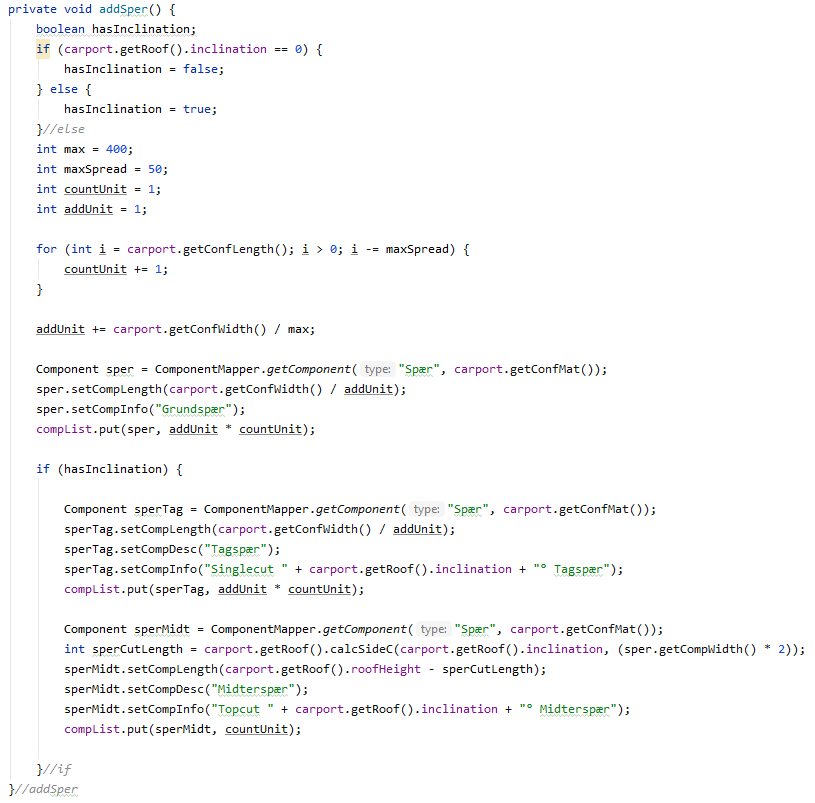
Koden sender et sql request databasen, som sender resultset tilbage til metoden.

Metoden piller resultset’et fra hinanden og indsætter værdierne i de dertil egnede variabler.

Disse vabler bruges når metoden nederst opretter et objekt af klassen Part.

Metoden sender objektet tilbage til metodekaldet.

## 1.6.2 – Tilføj komponent



Her har vi en metode som tilføjer en komponent type som objekt til vores komponentliste og et antal.

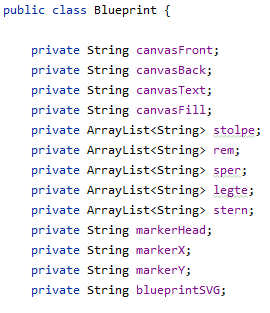
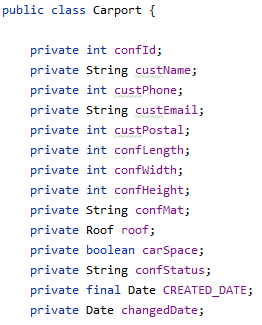
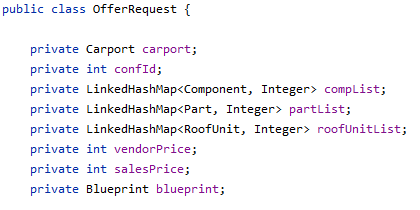
Metoden er skræddersyet til at tilføje spær til komponentlisten, så vi er i stand til at indsætte standard værdier som er relevante for netop spær.

Vi har en ’if condition’ som aktiveres hvis carporten har tagrejsning.

Generelt kan man se hvordan metoden spørger en Mapper metode om at få et komponent objekt.  
Dette objekt bliver så ændret i længden så det passer til konfigurationen og i beskrivelsen.

Den færdige komponent sættes så ind i mappet, med det samlede antal som Value.

## 1.6.3 – Færdige klasser

Til slut står vi med et OfferRequest objekt som kan sendes videre til LogicFacade klassen(hvis metoder bad om objektet til at begynde me). OfferRequestet har yderligere to objekter i sig, nemlig Carport og Blueprint.