Tegninger

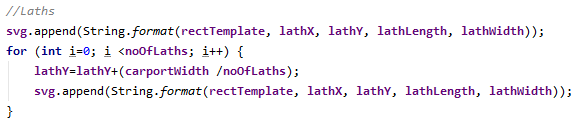
Vi har i projektet gjort brug af Scalable Vector Graphics også kalde SVG, som er vector baseret grafik i XML format. Det bruger vi i dette tilfælde til at tegne dynamiske tegninger ved hjælp af firkanter og linjer på en x/y akse. I modsætning til en almindelig x/y akse fungere SVG systemer lidt anderledes hvilket gør at man roteret aksen som vist på tegningen til højre. Det smarte ved SVG er at det skalerbart og kan vises i høj kvalitet i enhver opløsning, hvilket gør det ideelt til tegninger af carportene i vores projekt.

Kodeeksemplerne nedenunder er fra klassen Svg.java

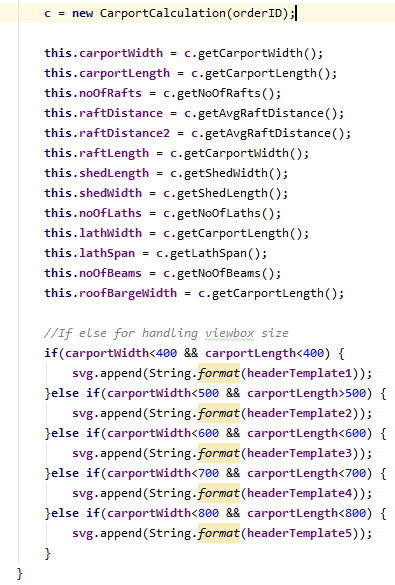
Måden hvorpå vi har bygget vores kode op omkring SVG motoren er ved hjælp af en constructor, en metode til at tegne og en StringBuilder hvor vi appender strenge som indeholder de SVG informationer der skal bruget for at fremvise de enkelte elementer i tegningen.

StringBuilder’en bliver altså fodret med en masse strenge som til sidst danner en sammenhængende SVG tegning på hjemmesiden.

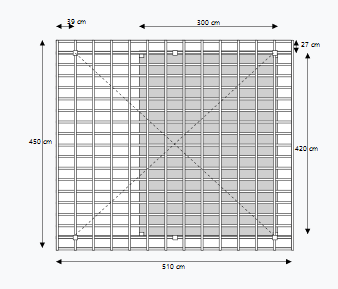


Ovenstående Streng er en af flere templates vi har lavet for at kunne generere de enkelte strenge såsom denne, der danner en firkant og forskyder den på canvas så det bliver placeret det rigtige sted. Det kunne f.eks. være en lægte som skal placeres på et tag.

I ovenstående eksempel laver vi et for loop for at tegne samtlige lægter på en gang, for så til sidst at appende den sammen med rectTemplate til stringbuilderen svg. På den måde bliver det tilføjet til de mange strenge som til sidst danner en stor streng med alt SVG indholdet.

For at bygge hele carporten har vi en metode kaldet addCarport() som indeholder alle vores strenge. Det er en stor metode som gør brug at samtlige templates samt variablerne fra CarportCalculations. Disse variabler bliver smidt ind på x og y’s pladser i de strenge der skal appendes til stringBuilderen. Derved for strengene de rigtige værdier med og derved SVG’en de rigtige mål.

For at kunne fylde vores SVG motor med alle de mål der skal til, for at tegne en carport, laves der en instans af CarportCalculation klassen. Herefter tildeler vi de lokale variabler i Svg klassen, værdier fra CarportCalculations, så vi får de dynamiske mål, som passer på kundes ønsker. Alt dette gjort i en constructor som også laver en headerTemplate baseret på carportstørrelsen således at vores viewbox på hjemmesiden ikke bliver tilpasset i størrelse.

Ret hurtigt rendte vi ind i et browserproblem med hensyn til kommaer og punktummer. Dette løste vi ved hjælp af en replace som tager alle kommaer og erstatter dem med punktummer for at tegningen kunne vises i en browser. Dernæst har vi nogle faste værdier i vores templates som igen skulle konverteres tilbage til kommaer for at de fungerede i SVG formatet i browseren.

På denne måde generere vi vores tegninger dynamisk i forhold til kundens indtastninger. Det smarte ved at gøre brug af SVG er at det som sagt tidligere er i XML format, og derfor ikke behøver at være en ekstern fil man hiver ind. Vi kan derimod generere dem dynamisk og skalerbart i selve koden, hvilket vi har formået at gøre ved hjælp af CarportCalculation klassen. Vi valgte at tegne 4 typer tegninger. Fra siden med farver, arbejdstegning fra siden, arbejdstegning oppe fra og arbejdstegning forfra. Dette gjorde vi ud fra de papire vi har fået af Fog som viser deres nuværende arbejdstegninger og hvad Fog finder vigtigt for kunden at vide. Vi valgte at tilføje en tegning fra siden med farver så kunden havde mulighed for at få visualiseret deres carport. Netop fordi at man som kunde kunne have svært ved at forstille sig den endelige udfald ud fra en arbejdstegning.