Mikkel Andreasen

Resumé

I dette dokument vil vi komme over objektorienteret design, og nogle linjers kode.

Dokumentation af OOA og kode

Indhold

[Object-Oriented Design (OOD): 2](#_Toc73436267)

[Kort beskrivelse af diagrammer brugt i OOD: 2](#_Toc73436268)

[Klassediagrammer (KD) 2](#_Toc73436269)

[Sekvensdiagrammer (SD) 2](#_Toc73436270)

[Klasser: 3](#_Toc73436271)

[NewyorkerController 3](#_Toc73436272)

[Email 3](#_Toc73436273)

[PersonData 3](#_Toc73436274)

[CalculateOffer 3](#_Toc73436275)

[WallLayout 4](#_Toc73436276)

[Sketch 4](#_Toc73436277)

[ConnectToDB 4](#_Toc73436278)

# Object-Oriented Design (OOD):

### Kort beskrivelse af diagrammer brugt i OOD:

## Klassediagrammer (KD)

Klassediagrammet viser hvilket klasser der snakker sammen, samt giver et overblik over hvilke funktioner, den specifikke klasse, indeholder.

Klassediagrammet er særligt vigtigt, der det kan vise programmets kode, på mere brugervenligt niveau, uden man behøver at kigge source koden igennem, for at se hvordan programmet fungerer.

Dette gør det nemmere for folk, som ikke har været med til udviklingen af programmet, til at forstå, eller skrive videre på programmet.

## Sekvensdiagrammer (SD)

Sekvensdiagrammet fungerer lidt som SSD, men på et dybere niveau.

Systemsekvensdiagrammet, viser i hvilket sekvens de forskellige funktioner bliver udført, samt hvilke klasser, som funktionerne henvender sig til.

Dette giver personerne som skal kode programmet, et tydeligt overblik over hvilke funktioner, og klasser, der skal fremgå, i programmet. Dette kan være en yderst god ting at have, der den kan spare alle for meget tids spild, brugt på fejl rettelse.

# Klasser:

## NewyorkerController

klassen NewyorkerController, står for håndteringen af brugerens input af ”hight”, og ”width”, som funktionen createNewOffer, så tager og regner på. Dette resultat bliver så returneret, så andre klasser kan bruge udregningen.

## Email

Klassen Email, står for vidregivningen af persondata, samt data.

Funktionen createEmail tager personData, connectToDB, sketch, og price, som den så bruger til generering af Emails til NewYorker.

Den Email som der så bliver sendt, indeholder den persondata som kunden har indtastet, samt målene til den væg de har lavet, og en pris.

Funktionen createEmail tager også connectToDB med som input, der den skal kunne opbevare oplysningerne på den database, der så kan tilgås af NewYorker.

## PersonData

getName() : name

getEmail() : eMail

getPhoneNumber() : phoneNumber

Snakker til ConnectToDB

Klassen PersonData, står for håndtering af kundens indtastede oplysninger om sig selv, som den så klargøre til brug i andre klasser, ved at returnere dens udregninger.

## CalculateOffer

calculatePrice(totalPane int, totalFrame int) : price

rejectOffer() : RejectionMsg

Snakker til WallLayout

CalculateOffer er en Klassen, som står for udregningen af prisen på den væg, som kunden har designet.

Her er det så calculatePrice funktionen, der tager totalPane, og totalFrame, og regner med det. Dette bliver så returneret, så andre klasser kan bruge dataen. Hvis funktionen rejectOffer bliver kaldt, bliver den udregning ikke brugt, og kan overskrives af den næste beregning.

## WallLayout

createNewOffer(height double, width double)

calculatePane(height double) : totalPane

calculateFrame(width double) : totalFrame

## Sketch

updateSketch(wallLayout WallLayout) : sketch

Snakker til WallLayout

Sketch Klassen, står for håndterigen af alle former for billeder, og tegninger, af væggen.

updateSketch funktionens opgave, er at generere et billede af væggen, samt en udgave af dette billede, med mål, og materialer taget i brug. Dette bliver så returneret, så ander klasser kan bruge tegningerne.

## ConnectToDB

connectToDB()

selectFromPersonData(email String)

insertIntoPersonData(name String, phoneNumber String, eMail String)

deleteFromPersonData(email String)

Klassen ConnectToDB, står for at lave en forbindelse til den valgte database, hvor klassen PersonData så kan kommunikere med databasen.