Programmering B Rapport

ELEV: Nelisa Chehrehgosha og Mie Norre Engemann  
FAG: Programmering B  
KLASSE: 2.R  
SKOLE: Aarhus Gymnasium C  
VEJLEDER: Mirsad Kadribasic  
DATO FOR AFLEVERING: 2/05/2022

Indholdsfortegnelse

[1 Abstract 3](#_Toc102416261)

[2 Problemforulering 4](#_Toc102416262)

[3 Programmet 5](#_Toc102416263)

[library 5](#_Toc102416264)

[3.1 design 7](#_Toc102416265)

[3.1.2 pseudokode 7](#_Toc102416266)

[3.1.3 Formelregner 11](#_Toc102416267)

[3.1.4 GUI 12](#_Toc102416268)

[3.1.5 scrum 13](#_Toc102416269)

[3.2 dokumentation 14](#_Toc102416270)

[3.2.1 kode kommentar 14](#_Toc102416271)

[3.2.2 kode stil 14](#_Toc102416272)

[3.2.3 funktion beskrivelser 15](#_Toc102416273)

[3.2.4 klasse diagram 15](#_Toc102416274)

[3.3 deling 16](#_Toc102416275)

[3.3.1 github 16](#_Toc102416276)

[4 Test 17](#_Toc102416277)

[5 Konklusion 18](#_Toc102416278)

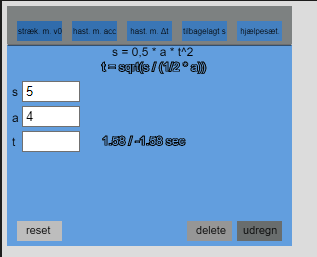
[6 Bilag 19](#_Toc102416279)

# 1 Abstract

For denne opgave er der lavet et program, som led i programmeringseksamen i programmering B. Programmet er skrevet i JavaScript på p5.js.

Programmet er en formelregner, som udregner fysik formler omhandlende bevægelse ud fra de inputs den får af brugeren.

Programmet er målrettet opgaveregning og undervisningsbrug. Programmet regner først og fremmest ens opgaver ud fra givet værdier, men det giver også brugeren den isoleret formel, som så giver brugeren et indblik i hvad der bliver regnet og hvorfor man får det svar man gør. Tanken er at man lærer lidt hver gang man bruger formelregneren ud fra et matematisk udgangspunkt, ved at man kan se sammenhængen mellem formel og isoleret formel, hvilket også kan hjælpe en med bedre at forstå fysikken bag.

Nedenunder er vist et skærmbillede fra programmet, som viser hvordan svaret vil se ud på en af formlerne med vilkårlige indsatte værdier.  
  


Skærmbillede

# 2 Problemforulering

Meget fysikforståelse og mange fysikopgaver bygger på et fundament af matematik. På den måde kan en god matematik indsigt hjælpe med en bedre forståelse af fysikken, og også til regning af fysikopgaver er denne indsigt nødvendig. Men siden man husker og lærer bedst ved at øve sig, så må det mest optimale være at få matematik og fysisk forståelsen til at arbejde sammen i konkrete opgaver.  
Hvordan laver man et program som formår dette uden det bliver nytteløst, når man først har forstået fysikken og matematikken bag?  
Og hvordan gør man samtidig programmet forståeligt nok for modtageren, så de ikke skal bruge alt deres energi på at forstå selve programmet?

Disse forudsætninger er afgørende for om programmets kerne lykkes.

# 3 Programmet

FlowChart

Programmet består af en menu, hvori man kan vælge imellem de forskellige formler inden for fysikemnet ’bevægelse’ vha. knapper, samt et ”regnefelt”, hvor man kan indskrive sine kendte værdier og få givet den isoleret formel og beregnede manglende værdi.  
Til højre vises et FlowChart for den ikke-interaktive del af programmet, altså hvad der sker når programmet startes.

Det består af tegning af menuen med sine formelknapper og tegning af ”regnefeltet” med reset-, delete- og udregningsknap.

## library

For at kunne importere lyde er der brugt library. Lyde kan bruges på mange forskellige måder alt efter hvilket program det er. Lyde er blevet brugt til alle knapperne i dette program, og de afspilles, når der trykkes på knapperne. Vi deler de fire forskellige knapper op i fire klasser ud fra, hvad de laver. Hver slags knap har egen lyd, som hænger sammen med knappens funktion, eksempelvis vil nogen kunne kende lyden for delete-knappen fra MacBooks.  
Der er fire lyde i alt. For formlerne, skal man vide om der bliver klikket på, derfor blev der brugt en grundlæggende keyboardlyd. Udregn-knappen skal have en dejlig lyd for at indikere, at løsningen er blevet beregnet. Vi brugte en reverse lydeffekt til reset-knappen for at gøre det klart, hvad den gør.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Text

Description automatically generatedLaver en funktion for lyd

Kode : preload af lyd

Kalder mySoundUdregn i udregn klassen Kalder mySoundDelete i delete klassen

Text

Description automatically generatedText

Description automatically generated

Kode : afspilning af DeleteLyd

Kode : afspilning af UdregnLyd

*Graphical user interface, text

Description automatically generated*Text

Description automatically generatedKalder mySoundReset i reset klassen Kalder mySoundKnap i button klassen

Kode : afspilning af FormelknapLyd

Kode : afspilning af ResetLyd



## 3.1 design

### 3.1.2 pseudokode

**Main (sketch)**Set up  
Definer skærm

Intansiere formelknapper (kalde klassen Button) med parameter for rgb-værdier, værdier til påvirkning af knap-placering og knap-navn

Intansiere reset, delete og udregn knap (kalde deres klasser 1 gang) med parameter for rgb-værdier og værdier til påvirkning af knap-placering

Intansiere formler (kalde klassen formula) med parameter for rgb-værdier, værdier til påvirkning af knap-placering og formel-nummer

Rect(Tegn menu)  
Rect(Regnefelt)

Draw  
Kald OverKnap Funktion fra Button-objekterne

Kald OverKnap Funktion fra Reset, Delete og Udregn objekterne

Kald formula Funktion fra Formulas-objekterne

Mouse Pressed Funktion  
 kald On\_Off funktion fra Button-objekterne

kald On\_Off funktion fra Reset, Delete og Udregn objekterne

kald On\_Off funktion fra Formulas-objekterne

**OverKlasse (parent klasse for alle klasser)**constructor (parametre…)  
this.parametre = parametre;

lav universelle værdier ift alle knapper

OverKnap Funktion  
 if (mus over knap)  
 påvirk rgb-værdier med tal mellem 0-1  
 else  
 farv med rgb-værdier  
  
 rect(knap)

On\_Off Funktion (bliver kun kaldt gennem anden klasse, hvis mousePressed)  
 if (mus over knap)  
 return true  
 else  
 return false

**Button (extends OverKlasse)**constructor (parametre…, unikParameter)  
super(parametre…)

this. unikParameter = unikParameter

OverKnap Funktion  
 super.OverKnap()  
  
 text(”knappe navn”, på knap)

On\_Off Funktion (bliver kun kaldt gennem anden klasse, hvis mousePressed)  
 if (super.On\_Off == true)  
 afspil lyd

**Delete/Reset/Udregn (extends OverKlasse)**constructor (parametre…)  
super(parametre…)

OverKnap Funktion  
 super.OverKnap()  
  
 text(”knappe navn”, på knap)

On\_Off Funktion (bliver kun kaldt gennem anden klasse, hvis mousePressed)  
 if (super.On\_Off == true)  
 afspil lyd  
 return boolean = true

**Formulas (extends OverKlasse)**constructor (parametre…, unikParameter)  
super(parametre…)

this. unikParameter = unikParameter

lav universelle værdier bla. Ift. placering af svar

lav inputfelter  
giv inputfelter position udenfor canvas

On\_Off Funktion (bliver kun kaldt gennem anden klasse, hvis mousePressed)  
 if (super.On\_Off == true)  
 if (formel-nummer == første)  
 reset regnefelt  
 text(”formel”, regnefelt)  
 giv det nødvendige antal inputfelter position indenfor canvas  
 text(”værdi/symbol for inputfelt”, side af inputfelt)

if (formel-nummer == anden)  
 (….samme som for ’første’, men tilpasset ’anden’….)

if (formel-nummer == tredje)  
 (….samme som for ’første’, men tilpasset ’tredje’….)

*\*\*\*ovenstående fortsættes til ikke flere formler er\*\*\**

formula Funktion (boolean fra Reset, Delete og Udregn)  
 this.parametre = parametre;

if (boolean Delete == true)  
 fjern inputfelters indskrevne værdier  
 fjern isoleret formel og svar

if (boolean Reset == true)  
 fjern inputfelters indskrevne værdier  
 fjern formel, isoleret formel, svar og inputfelters værdi/symbol  
 giv inputfelter position udenfor canvas

if (formel == første && Boolean Udregn == true)  
 kald funktion for udregning af ’første’ formel dvs. beregning2()

if (formel == anden && Boolean Udregn == true)  
 kald funktion for udregning af ’anden’ formel dvs. beregning2()

*\*\*\*ovenstående fortsættes til ikke flere formler er\*\*\**

Beregning1 Funktion  
 this.inputfelterVærdier = inputfelterVærdier;

If(isNaN(x-inputfelterVærdier) && isNaN(resterende inputfelterVærdier) == false)  
 udregn ukendte værdi  
 text(”ukendte værdi”, side af tilhørende inputfelt)  
 text(”isoleret formel ift ukendt værdi”, under oprindelige formel)

*\*\*\*ovenstående fortsættes til ikke flere inputfelterVærdier er\*\*\**

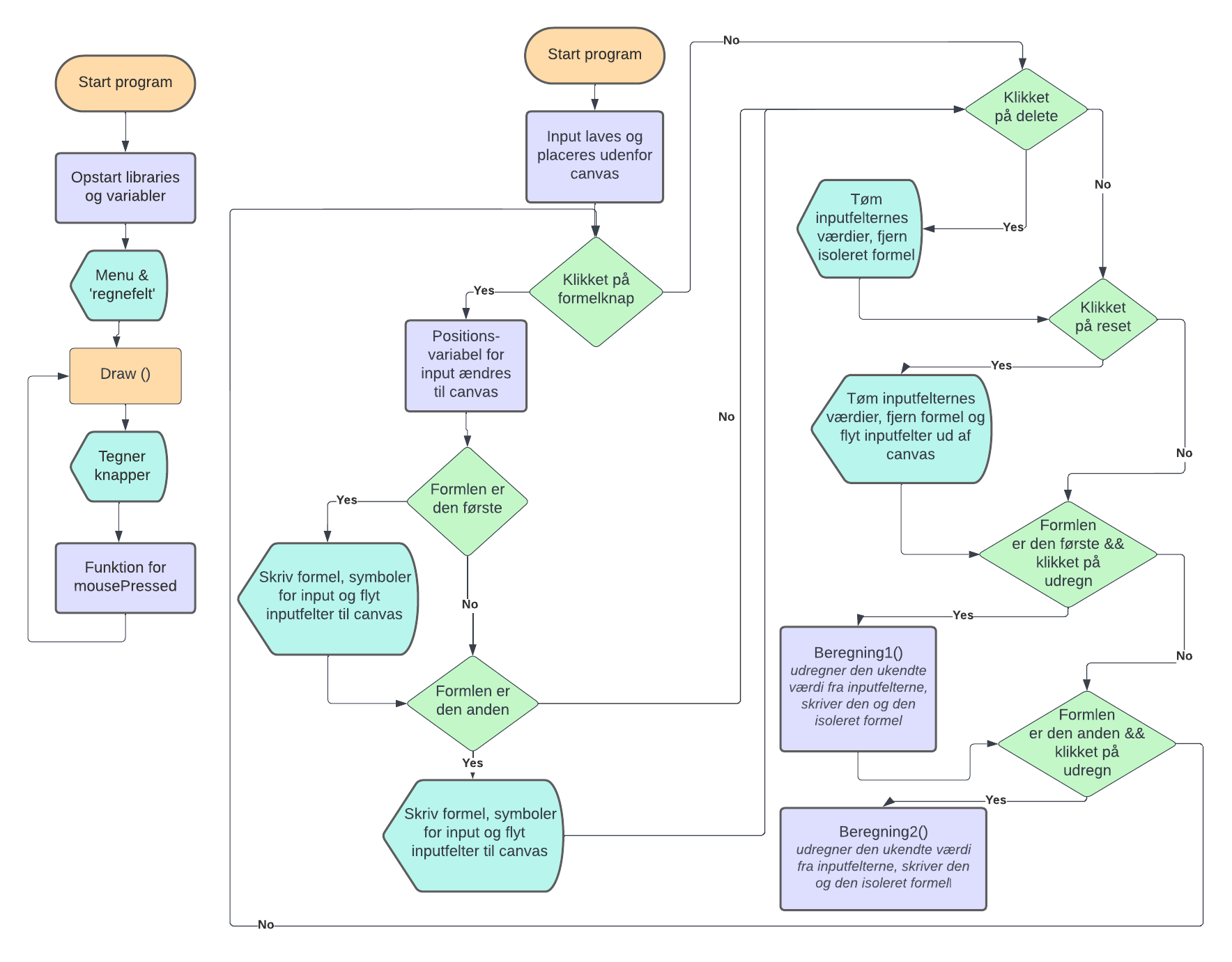
Beregning\* Funktion  
 this.inputfelterVærdier = inputfelterVærdier;

If(isNaN(x-inputfelterVærdier) && isNaN(resterende inputfelterVærdier) == false)  
 udregn ukendte værdi  
 text(”ukendte værdi”, side af tilhørende inputfelt)  
 text(”isoleret formel ift ukendt værdi”, under oprindelige formel)

*\*\*\*ovenstående fortsættes til ikke flere inputfelterVærdier er\*\*\**

*\*\*\*ovenstående funktioner fortsætter til ikke flere formler der skal udregnes er\*\*\**

### 3.1.3 Formelregner

Længere nede er et FlowChart over, hvad der sker i klassen Formulas, når man starter med at interagere med programmet. Der er fokuseret på Formulas klassen, da det er den som fungerer som selve lommeregneren, og så modtager den værdier fra hhv. Delete, Reset og Udregn klasserne, som fortæller om der er trykket på dem. På den måde inkluderer Formulas klassen næsten alt, bort set fra GUI delen af knapperne, hvilket bl.a. også ses i at funktionen OverKnap, der står for GUI’en i de andre klasser, aldrig bliver kaldt i Formulas. 

FlowChart : Formulas

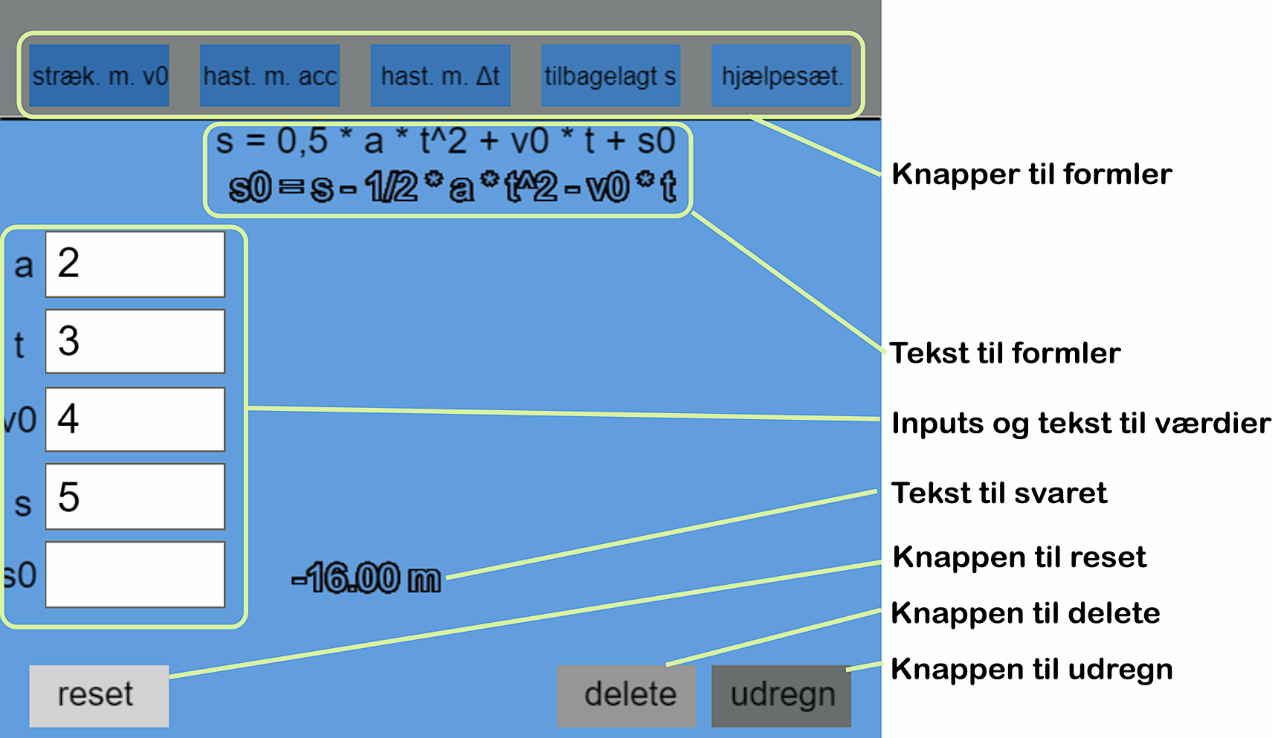
\* I FlowChartet oven over er der taget udgangspunkt i et program med 2 formler. Vores program har 5 formler, men for forståelses skyld er der kun taget udgangspunkt i 2, siden der intet nyt kommer med de tre sidste formlers indblanding.

### 3.1.4 GUI

GUI, Graphical User Interface, er det som brugeren ser, når de benytter og interagerer med et program. Det er en vigtig del af programmet at formidle det til brugeren sådan at det er letforståeligt og intuitivt at bruge.  
GUI bruges bl.a. i operativsystemer og installeret software, hvor grafiske komponenter som ikoner, billeder og knapper benyttes til at hjælpe brugeren gennem interaktion med en applikations funktionalitet.

#### Knapper

Når programmet er startet, så har brugeren 5 formel-knapper at trykke på, og et tryk på dem vil klargøre formelregneren til en udregning af den formel, derudover er der også 3 andre knapper for hhv. reset, delete og udregn.  
Ved disse knapper ses GUI‘en ved at rektanglet, som knappen består af, bliver mørkere i farven, hvis man har musen over den. På den måde gør man det klart for brugeren, hvilken knap de befinder sig på. Det gør alt sammen at hvad egentligt bare er et rektangel på skærmen, kommer til at virke som en mere intuitiv knap.  
I 3.2.3 beskrives funktionen OverKnap(), der står for knappernes GUI-element.

Her er en gennemgang af hvordan programmet ser ud for brugeren, når de har udregnet en ukendt værdi.  


Skærmbillede

På baggrund af skærmbilledet ovenover kan man også se, hvordan programmet er blevet gjort mere overskueligt for brugeren ved at menuen med formel-knapper har egne farver og baggrund, og ”regnefeltet” også har egne farver.

### 3.1.5 scrum

Her er vores tidsplan over formelregner lavet med Trello.

<https://trello.com/invite/b/XEB5Xn0L/1e184173d18e1ccc867ff3f601394c8e/formelregner>

Graphical user interface, application

Description automatically generated

Skærmbillede : tidsplan

.

## 3.2 dokumentation

### Et billede, der indeholder tekst Automatisk genereret beskrivelse3.2.1 kode kommentar

Kode : kode kommentar

Til højre ses et eksempel på, hvordan kode-kommentare er blevet brugt gennem projektet til at gøre det lettere og hurtigere at forstå koden, fordi man så har kendt formålet.  
Dette er især vigtigt med sådanne kommentare, når man er flere om at programmere et program, så man også kan forstå og bruge den andens kode uden total forvirring.

Kommentarerne kan benyttes til flere formål, de kan eks. bruges til at forklare hvad bestemte linjer og funktioner gør i programmet, som i eksemplet til højre.

### Et billede, der indeholder tekst Automatisk genereret beskrivelse3.2.2 kode stil

Kode : kode stil

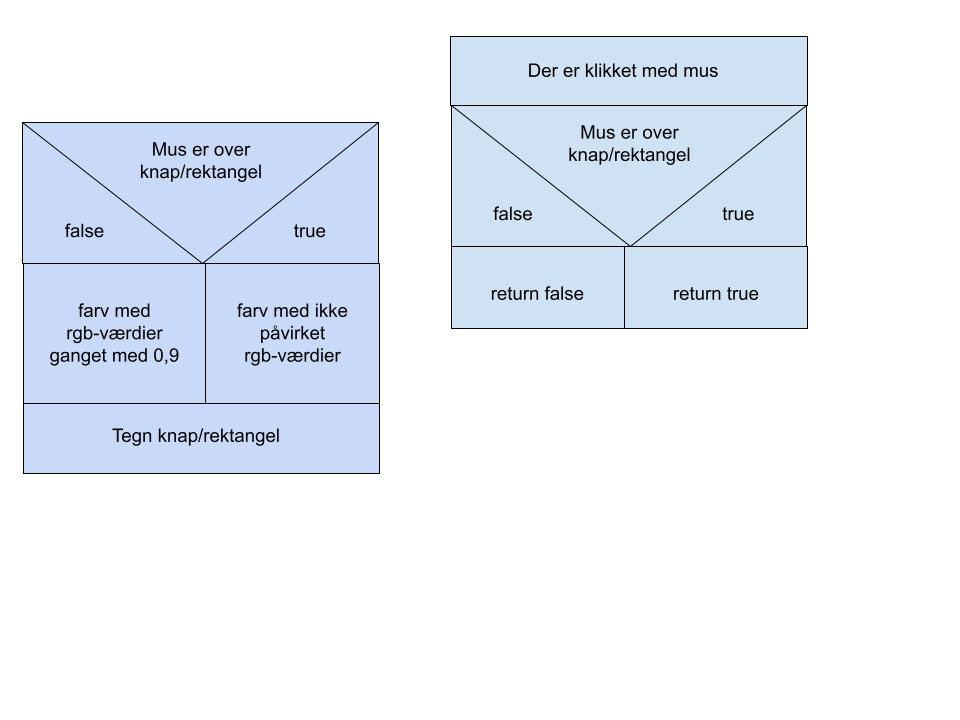
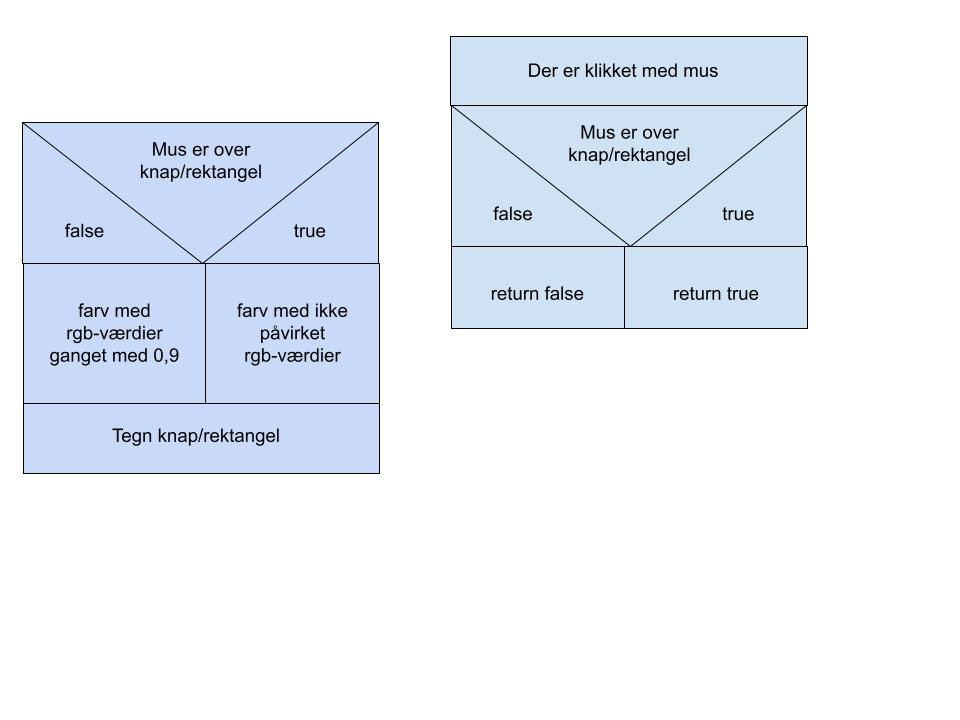
Til højre ses et eksempel på, hvordan vi har sat koden op med indryk, parenteser og kode-kommentare.

Sådan har vi sat koden op i alle klasser, fordi det skaber et godt overblik og er lettere for øjet at forholde sig til.  
Eksempelvis gør de indrykket parenteser og linjer, at man får et hurtigere overblik over hvad hører til hvad. Det kan man bl.a. se i eksemplet ved at men ikke er i tvivl om hvad gør ind under if-sætningen if(this.formel == ”første”).

### 3.2.3 funktion beskrivelser

Overklassens (parent klassen) funktioner, OverKnap() og On\_Off(), bliver brugt i alle andre klasser, derfor bliver de her beskrevet lidt nærmere med et Nassi-Schneidermann diagram hver.

Nassi-Schneidermann diagram for *OverKnap() (GUI)* Nassi-Schneidermann diagram for *On\_Off()*

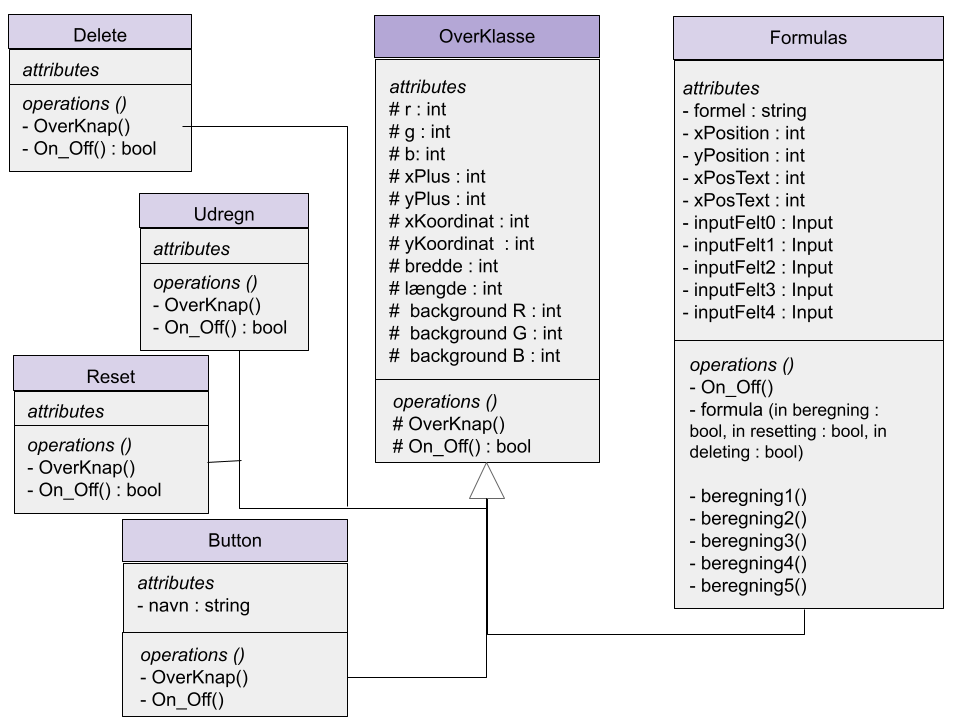


Figur 1: Nassi-Schneidermann diagram On\_Off

Figur : Nassi-Schneidermann diagram OverKnap

### 3.2.4 klasse diagram

Dette program har 1 parent klasse og 5 child klasser, og de er alle beskrevet i klassediagrammet nedenunder.



Figur : klasse diagram

## 3.3 deling

### 3.3.1 github

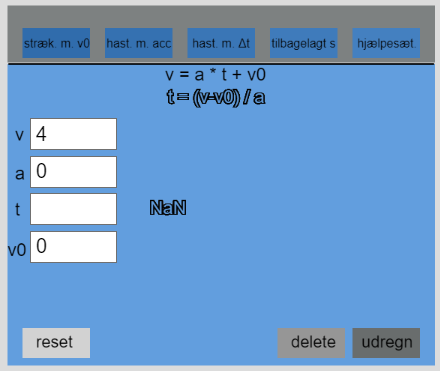
For at vi kunne arbejde sammen brugte vi GitHub til at dele opdateringer af de ting, vi har ændret i koden. Dette hjalp os til at kunne arbejde separat og nemt dele det op mellem os uden at skulle bekymre os om at noget kode vil gå tabt.

A screenshot of a computer screen

Description automatically generated with medium confidence

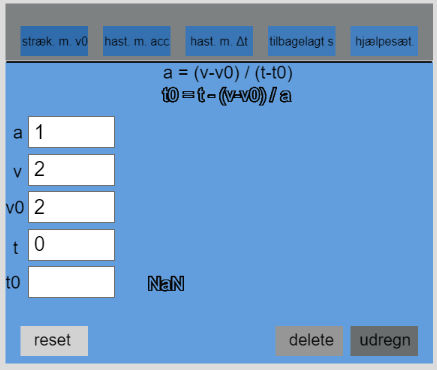
Skærmbillede : GitHub

# 4 Test

Da programmet skal kunne udregne den ukendte værdi i en formel, så må den af gode grunde også, som med al matematik, følger de matematiske regler der er.  
Eksempelvis kan man ikke dividere med 0 eller tage kvadratroden af et negativt tal, men det er vigtigt at fortælle brugeren hvorfor de ikke kan få et svar, hvis de forsøger at dividere med nul. Men da programmet også skal bruges til at skabe en bedre forståelse for både matematik og fysik, så giver programmet ikke bare svaret. I så fald man forsøger at dividere med nul, vil programmet skrive ’NaN’ (Not a number) uden for den ukendte værdi, og så må man jo ty til den isoleret formel for at forstå hvorfor dette er tilfældet - sådan ses det til højre. Forsøger man at tage kvadratroden af et negativt tal vil samme ske.  
Dertil er der også fulgt op på matematikken sådan, at man får både en positiv og en negativ version af det tal som en kvadratrod giver, samt at en andengradsligning, hvis diskriminanten er nul eller større, giver to værdier.

Skærmbillede : division med nul

Derfor blev der ført flere tests af de forskellige isolerede formler og ikke- isolerede formler.

Et andet eksempel på matematikkens begrænsninger, som måtte kodes ind ses nederst til højre. Indsætter man værdierne fra inputfelterne i den isolerede formel, så får man nul. Alligevel siger formelregneren NaN, fordi man i den oprindelige formel også har en division, hvor selvsamme værdier vil resultere i et forsøg på at dividere med nul.

Skærmbillede : division med nul

# 5 Konklusion

Et program er lavet, som kan br

Meget fysikforståelse og mange fysikopgaver bygger på et fundament af matematik. På den måde kan en god matematik indsigt hjælpe med en bedre forståelse af fysikken, og også til regning af fysikopgaver er denne indsigt nødvendig. Men siden man husker og lærer bedst ved at øve sig, så må det mest optimale være at få matematik og fysisk forståelsen til at arbejde sammen i konkrete opgaver.  
Hvordan laver man et program som formår dette uden det bliver nytteløst, når man først har forstået fysikken og matematikken bag?  
Og hvordan gør man samtidig programmet forståeligt nok for modtageren, så de ikke skal bruge alt deres energi på at forstå selve programmet?

Disse forudsætninger er afgørende for om programmets kerne lykkes.

# 6 Bilag