

Programmering B

Eksamensopgave



What can you say: "Oki!"

Deltagere: Kacper Kelner, Otto Esmann Jensen & Isabella Amalie Bang Eriksen

Skole: Aarhus Gymnasium C

Fag: Programmering B

Klasse: 18xar2

Vejleder: Mirsad Kadribasic

Dato for aflevering: onsdag d. 03/06/2020 kl. 14:00

1. Resume	3
2. Problemformulering	3
3. Metoder	4
3.1 Agile vs. vandfald	4
3.2 Objekt orienteret analyse (OOA) og objekt orienteret design (OOD)	4
3.3 Matematisk modellering	4
4. Programbeskrivelse, Funktionalitet, Brugergrænseflader, Udvalgt kode (se eksempel fil)	5
5. Pseudokode	6
6. Flowchart	7
7. Class diagram	10
8. Class og funktionsbeskrivelse	10
9. Test	11
10. Konklusion	11
11. Bilag	11
11.1 Code med "style" og kommentar	11
11.2 Litteraturliste	11

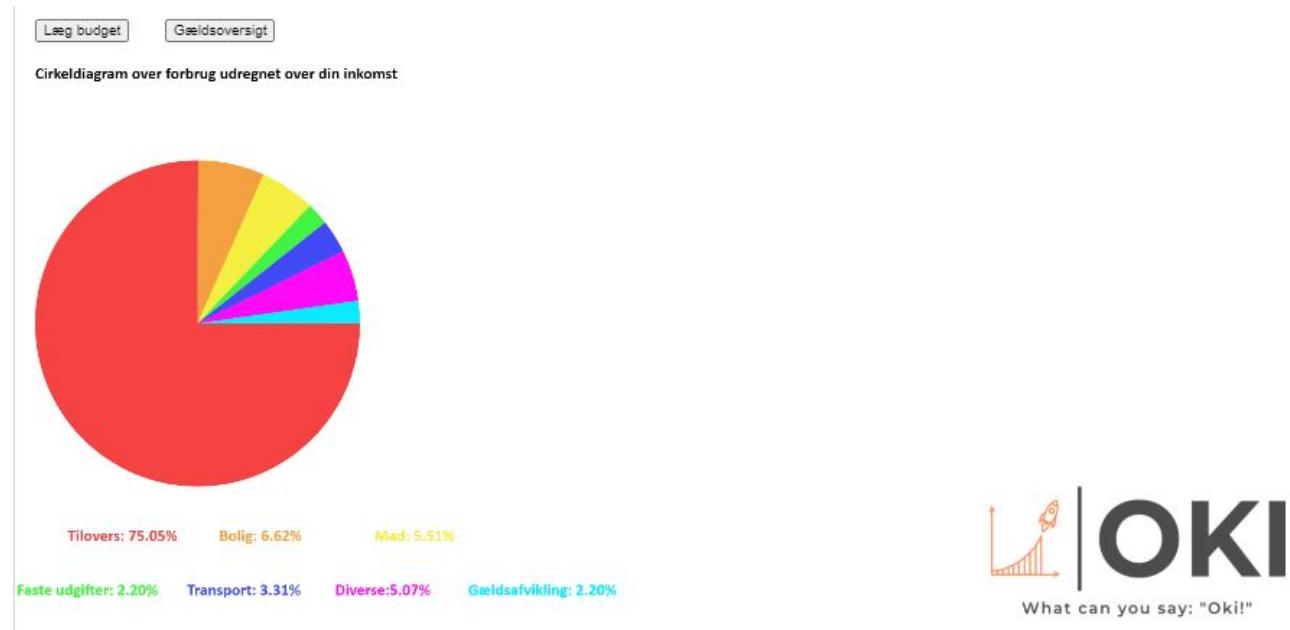
1. Resume

Denne opgave er udarbejdet i programmering B i sproget p5.js, hvor vi også har brugt html og javascript.

Programmet er lavet til at holde styr på brugerens økonomi, så man får styr på sine penge. Programmet genererer statistikker over brugerens økonomi ud fra, hvordan brugeren har indskrevet sin indkomst og forbrug.

Programmet er målrettet efter dem, som af en eller anden grund ikke har overskud i sin økonomi. Programmet skal hjælpe med at skabe overblik og beregne et budget ud fra brugerens økonomi og ønsker.

Her er et skærmbillede fra programmet:



2. Problemformulering

Økonomiske udfordringer har været et omfattende problem for både erhverv og private under corona krisen. Dette gør, at en stor del af samfundet og indbyggerne mister penge. Derfor har vores gruppe udarbejdet et program, som beregner statistikker, budget mm. ud fra indkomst, forbrug, opsparing og gæld. Dette skaber en række udfordringer, som skal løses, da årsprøven kræver en række krav, som programmet skal opfylde.

- Hvordan designes og programmeres opgaven, så det skaber den mest mulige oversigt over brugerens økonomi?
- Hvordan kodes programmet, så det opfylder gestaltlovene bedst muligt?

3. Metoder

3.1 Agile vs. vandfald

Ind for programmering er der som regel tale om 2 forskellige hoved metoder at udarbejde sit program på. Den første er Agile og den anden er vandfald metode. I dette projekt har vores gruppe valgt at arbejde med “agile metoden”. Grunden til dette valgt var at, vores gruppe havde en overordnet ide hvori den var der mange forskellige ting at gøre sig an på. Dette gjorde, at vi ikke mente, at vi skulle skabe et nøjagtigt planlægning af, hvordan programmet skulle fungere sammen lige med det samme. I stedet valgte vi at starte med vores hoved ide og undervejs tilføj vi nye elementer samt ideer som, kom frem ved selve skrivning af programmet. Vi udarbejdede idéen om et program, som holder styr på ens økonomi ved hjælp af en “brainstorm”, hvor efter vi programmerede små dele, som blev sendt til vores vejleder. Ud fra denne idé lavede vi designs, kravspecifikation og test, der blev brugt som en skabelon for at kunne kode programmet. I løbet af projektet udarbejde vi nye designs, krav osv., som efterfølgende blev brugt til at kode den endelige version af programmet.

3.2 Objekt orienteret analyse (OOA) og objekt orienteret design (OOD)

I vores program var det relevant at anvende objekt orienteret programmering (OOA). Objektorienteret programmering er især relevant i et program hvor, der er tale om mange forskellige elementer, som har de samme overordnede attributter, med måske nogle små forskelligheder. Og dette skete i vores tilfælde, da vi bruger mange knapper samt, input bokse. Blev det meget oplagt at bruge (OOA). Anvendelse af objekt orienteret programmering kan ses boede ved “knapperne” så som “input bokse” som befinner sig i vores program.

3.3 Matematisk modellering

Igenom programmet bliver der brugt forskellige matematiske udregninger. Disse udregning bliver brugt i forbindelse med økonomisk beregning såsom, Lægning af personens budget og circle diagrammet.

Læg budget:

Vi udregner, hvor meget brugeren har tilbage af sin løn efter at have udfyldt de forskellige bokse.

$$I_{tilbage} = I_{total} - (kr_{bolig} + kr_{mad} + kr_{fasteudgifter} + kr_{transport} + kr_{diverse} + kr_{gældsafvikling})$$

Class circle Diagram:

I class circle Diagram opdeler vi personens udgifter i forskellige kategori, hvor efter vi udregner vi dem i procent dele i forhold til helheden. Måden som vi gøre dette på ser således ud:

$$ProcentDel = \frac{forbrug_{information}}{alleforbrug_{infromatoner}} \cdot 100$$

4. Programbeskrivelse, Funktionalitet, Brugergrænseflader, Udvalgt kode (se eksempel fil)

Vores program er et program der har til formålet at hjælpe den enkelte mennesker med at, holde styr på sit eget økonomi. Dette har vi gjort ved at, visualisere hans månedligt forbrug samt indkomst ved hjælp af fx cirkeldiagrammer. Der over har vi give brugeren muligheden for at skabe overblik over sine løn samt låns renter.

Objekt orienteret programmering (OOP):

I vores program har vi benyttet os af den såkaldte objekt orienteret programmering. Vores program er hoved opbygget omkring 2 klasser (classes). Den første klasse er (box-class).

Denne klasse har til formålet at skabe alle boxe der bliver brugt til alle inputs-felter

igenmenn programmet. Som vi kan

observere i klassen bliver der

modtaget 4 forskellige attributter x,
y, text og output. De første 2 af
dem det ville sige (x,y) bliver brugt

til at placere vores boks i et

```
35 //Klasse (box) skaber box for input-felter og login knapper
36 class box
37 {
38     constructor(x, y, text, output)
39     {
40         this.x = x;
41         this.y = y;
42         this.text = text;
43         this.output = output;
44     }
}
```

bestemt sted på skærmen. Dette fungere præcis ligesomt i et koordinatsystem. Text værdi holder styr på hvilket navnet fremkommer på den enkelte knap. Og til sidst har vi output der er den værdi der holder styr på, hvad er det endeligt knappen skal gøre. Dette gøre den ved hjælp af clickLogin() funktion der består af en række if-statements. Som kan ses til højre. Disse if-statements laver om på den globale variabel "screen", denne variabel bliver så genbrugt, i function check(), som indeholder endnu flere if-statements. Der holder styr på hvilket skærm vi befinner sig på, og hvad er det endeligt der skal fremkomme på den. Et eksempel på dette kan observeres på den nederste billede på højre side.

Den anden og side klasse der bliver brugt i vores program er cirkelDiagram. Der bliver brugt til at skabe

```
47     clickLogin()
48     {
49         if (this.output == 0)
50         {
51             screen = 1
52         }
53     }
54
55     if (this.output == 1)
56     {
57         screen = 2
58     }
59
60     if (this.output == 2)
61     {
62         screen = 3
63     }
64 }
```

```
210 if (screen == 3) // Læg budget-siden
211 {
212     removeElements();
213     rect(0, 0, width, height)
214     budget();
215
216     fill("#222222");
217     textSize(20);
218     text("Månedlig indkomst", 20, 45)
219     text("Bolig", 20, 125)
220     text("Mad", 20, 205)
221     text("Faste udgifter", 20, 285)
222     text("Transport", 20, 365)
223     text("Diverse", 20, 445)
224     text("Gældsaftvilkning", 20, 525)
225     text("Lån", 20, 605)
226     text("Tilskud", 20, 685)
227     text("Sparekonto", 20, 765)
228     text("Bil", 20, 845)
229     text("Tilskud", 20, 925)
230 }
```

vores cirkeldiagrammer. Denne klasse skal modtage 4 værdier x og y der bedstemer cikreldiagramets placering i forhold til koordinatsystemet. Attributen split er en liste, der definerer hvordan pengene skal opdeles. Den indeholder værdiernes procentdel af det fulde budget. Vi opretter en metode for at tegne selve cirkeldiagrammet. Vi ganger hundreddedelene i split-attributen med 360 for at få hvilken vinkel der udgør de individuelle procenter i en hel cirkel. Et for-loop går i gennem listen og danner et arc for hver procentdel med dennes vinkel, til det tilsammen danner en cirkel. Da arc-funktionen i p5 kræver både fra- og til-vinkel, gemmer vi for hvert led i listen den forrige vinkel i en attribut, der hedder lastAngle. Vi danner så arc fra sidste vinkel til nuværende vinkel. Vi tilføjer en farve til dette arc ved at have en attribut kaldet colors som indeholder 6 forskellige farver, som for-loopet også vil gennemgå og bruge farver fra det nuværende led. Ned under kan der ses attribute lastAngle samt arc-funktionen.

```
461  createCircleDiagram()
462  {
463    for (this.i = 0; this.i <= this.amount - 1; this.i += 1)
464    {
465      forbrug.split = [nfc(forbruginfo[0]/hun*100,2), nfc(forbruginfo[1]/hun*100,2), nfc(forbruginfo[2]/hun*100,2),
466      nfc(forbruginfo[3]/hun*100,2), nfc(forbruginfo[4]/hun*100,2), nfc(forbruginfo[5]/hun*100,2),
467      nfc(forbruginfo[6]/hun*100,2)]
468      this.angle = radians(360*(this.split[this.i]/100))
469      fill(this.colors[this.i])
470      arc(this.x + 150, this.y, 300, 300, this.lastAngle, this.lastAngle + this.angle)
471      this.lastAngle += this.angle
472      textSize(15)
473      textStyle('Calibri');
474      textStyle(BOLD)
475      //text(this.label[this.i], (((this.x + 400) / this.amount) * this.i) + 25, 500)
476      if (this.i < 3)
477      {
478        text(this.label[this.i] + this.split[this.i] + "%", ((this.x + 400)/3) * this.i + 50, 500)
479      }
480      else
481      {
482        text(this.label[this.i] + this.split[this.i] + "%", ((this.x + 400)/3) * (this.i - 3), 550)
483      }
484      fill(color(255,255,255))
485    }
```

5. Pseudokode

Vores program bruges til at holde orden i økonomien ved, at brugeren selv indtaster ens indkomst og forbrug. Derefter bearbejder programmet ens data og udarbejder et budget.

Login:

Det første der ses når man åbner programmet er login siden, hvor man enten kan logge ind og sign ind - altså oprette en bruger. Når man opretter en bruger gemmer programmet brugeren i en json fil i en database.

Forsiden:

Den næste side viser forsiden, hvor der ses 2 knapper, som hedder "Læg budget" og "Gældsoversigt".

Læg budget:

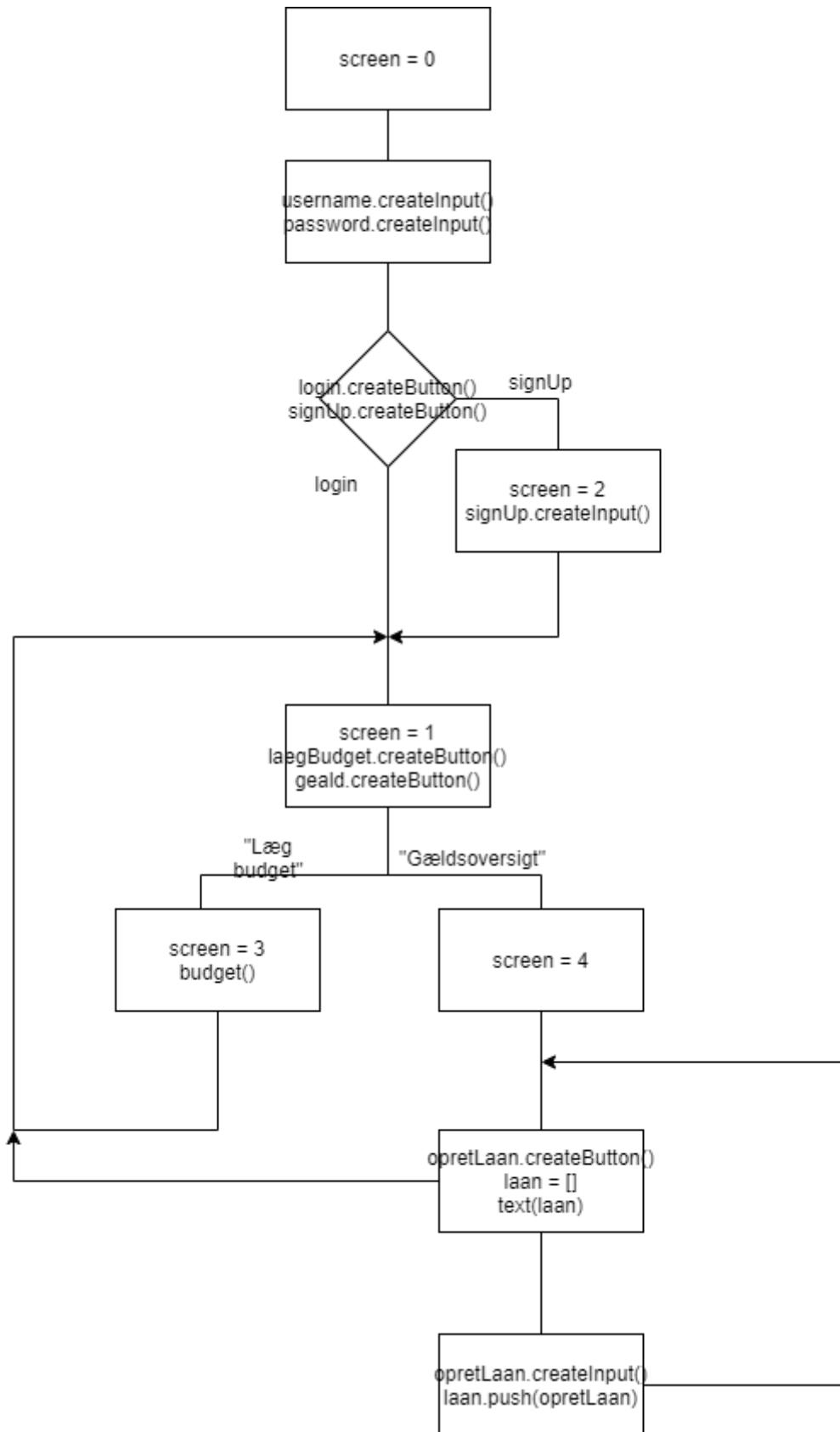
Når man klikker på “Læg budget” vises en ny side, hvor man indtaster ens månedlig indkomst og derefter ens forbrug, som er opdelt i følgende: Bolig, Mad, Faste udgifter, Transport, Diverse og Gældsafvikling. Efter man har indtastet beløbene, beregner programmet, hvor mange penge man har tilbage af sin indkomst. Derefter trykker man på knappen “Tilbage”, som tager en tilbage til forsiden, hvor på forsiden der vises et cirkeldiagram og procent opdelingen af, hvordan man bruger sine penge.

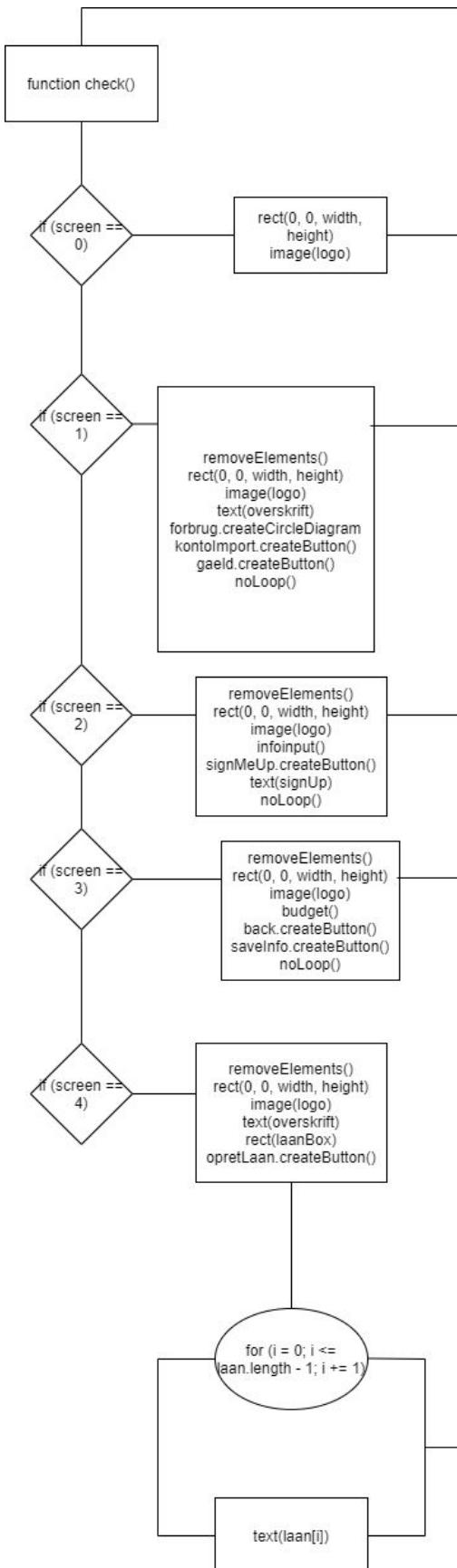
Gældsoversigt:

Når man klikker på “Gældsoversigt” vises en side, som holder oversigt over ens lån.

6. Flowchart

Til højre ses et overordnet flowchart om hele vores programs struktur.





7. Class diagram

Herunder ses 2 klasse diagrammer over de 2 klasser, som er i vores kode. Hver klasse har deres egen boks, som er inddelt i 3 felter. Det øverste felt indeholder navnet på klassen. Feltet i midten indeholder klassens attributter. Det nederste felt indeholder klassen funktioner.

box	circleDiagram
<ul style="list-style-type: none">- this.x- this.y- this.text- this.output	<ul style="list-style-type: none">- this.x- this.y- this.split- this.label- this.colors- this.lastAngle- this.amount
<ul style="list-style-type: none">+ constructor(x, y, text, output)+ clickLogin()+ createInput()+ createButton()	<ul style="list-style-type: none">+ constructor(x, y, split, label)+ createCircleDiagram()

8. Class og funktionsbeskrivelse

Funktioner fra p5js:

Gennem vores program bruger vi funktioner der allerede er et del af programmeringssproget (p5js). Disse funktioner er draw() og setup(). Funktionen draw() fungere som et loop der bliver opdateret 60 gange i sekundet. Hvor imod setup() er en funktion der bliver opdateret kun en gang, ved selve starten af programmet. Og derefter bliver den slet ikke opdateret.

Funktion check():

Denne funktion fungere som et slags tjekker, der er skabt til at tjekke hvilket skærm vi befinder sig på. Og på det samme tidspunkt oprette alle de relevante inputs, text, image, box osv. Det er også derfor den befinner sig ind for draw().

Funktion infoinput ()

Funktion infoinput skaber input felterne for vores Sign up-side hvor man indsætter informationer omkring sit navn, efternavn, email@ osv. Der ud over bruger den også funktionen

Funktion jsonFil()

jsonFil() er en funktion der opretter samt downloader et Json Fill der indeholder informationer omkring personen, såsom navn, efternavn, email og adgangskoden.

9. Test

Programmet blev programmeret og testet i p5.js ved, at vi programmerede hver for sig og tjekkede for fejl, hvorefter vi sammensatte koden og tjekkede for fejl igen. Vi brugte console.log til at tjekke om programmet gik ind i funktionerne.

10. Konklusion

Gennem dette projekt har vi udviklet et program, som holder overblik over den private økonomi. Det lykkedes os at programmere et program, hvor vi først designede dets udseende, og derefter udarbejde koden. Programmet skaber et godt overblik over ens økonomi med et cirkeldiagram og procenter, som er nemt at overskue. Desuden opfylder dette gestaltlovene, her ses to eksempler.

Figur og baggrund:

Vi har gjort det nemt at overskue forskellen på baggrund og forgrund. I baggrunden ses vores logo og den hvide farve, som er generel for alle vores side. Forgrunden på alle siderne ses statistikker, tal, knapper m.m., og de forveksles ikke med baggrunden.

Loven om nærhed:

Knapperne holdes tæt sammen i programmet, hvilket gør brugeren ikke bliver forvirret over, hvor de resterende knapper er. På forsiden ses cirkeldiagrammet og procenterne, som hører til. De står tæt sammen og gør ikke programmet forvirrende.

11. Bilag

11.1 Code med "style" og kommentar

```
// Objekter og variabler bliver skabt  
  
let input,input2,button  
  
let json = {};//Ny JSON fil
```

```
let savefile = true;

let screen = 0

let information = []

let forbruginfo = []

let info = true

let width = 1200

let height = 600

let logo;

let laan = [];

let forbrug1 = 0

let hun = 0

// Setup-funktionen kører kun én gang ved starten af programmet. Indeholder
// Login-knapper og inputbokse.

function setup() {

  createCanvas(width, height);

  logo = loadImage('unknown (1).png')

  username = new box((width/2)-100, 300, "Username")

  password = new box((width/2)-100, 330, "Password")

  login = new box((width/2)-45, 360, "Login", 0)
```

```
signup = new box((width/2)-51, 390, "Sign up", 1)
```

```
backTo1 = new box(1100, 20, "Tilbage")
```

```
username.createInput()
```

```
password.createInput()
```

```
login.createButton()
```

```
signup.createButton()
```

```
}
```

```
//Klasse (box) skaber box for input-felter og login knapper
```

```
class box
```

```
{
```

```
constructor(x, y, text,output)
```

```
{
```

```
    this.x = x;
```

```
    this.y = y;
```

```
    this.text = text;
```

```
this.output = output;  
}
```

//clickLogin formål er at tjekke hvilket skærm der skal frem visses. Variablen output bestemmer hvilken skærm, der fremkommer.

```
clickLogin()  
{  
    if (this.output == 0)  
    {  
        screen = 1  
        //console.log("hello")  
    }  
  
    if (this.output == 1)  
    {  
        screen = 2  
  
    }  
  
    if (this.output == 2)  
    {  
        screen = 3  
    }  
}
```

```
if (this.output == 3)
```

```
{
```

```
screen = 4
```

```
}
```

```
if (this.output == 4)
```

```
{
```

```
screen = 5
```

```
}
```

```
if (this.output == 5)
```

```
{
```

```
screen = 4
```

```
}
```

```
if (this.output == 6)
```

```
{
```

```
screen = 7
```

```
}
```

```
if (this.output == 7)
```

```
{
```

```
screen = 8
```

```
}
```

```
}
```

```
//createInput skaber alle inputfelter, der er i programmet.
```

```
createInput()
```

```
{
```

```
    this.input = createInput(this.text)
```

```
    this.input.position(this.x,this.y)
```

```
}
```

```
// CreateButton skaber alle knapper der bliver brugt i programmet.
```

```
createButton()
```

```
{
```

```
    this.button = createButton(this.text)
```

```
    this.button.position(this.x,this.y)
```

```
    this.button.mousePressed(() => {this.clickLogin()});
```

```
}
```

```
}
```

// Funktion jsonFil bliver brugt til at skabe en JSON-fil. Som indeholder information om selve brugeren der opretter sig ind gennem (Sign-up).

```
function jsonFil()  
{  
    json.name = firstname.input.value();  
    json.lastname = lastname.input.value();  
    json.email = email.input.value();  
    json.password = password.input.value();  
    saveJSON(json, "konto.json")  
}
```

//Function check kører i et loop og tjekker hvilken skærm, vi befinder os i. Derudover opstiller den alle de ting, som skal bruges på de enkelte skærme.

// I funktionen kan der observeres if-statements, der tjekker hvilken skærm, der vises.

```
function check()  
{  
    if (screen == 0) // Login-skærmen  
    {
```

```
rect(0, 0, width, height)  
image(logo, 380, 75, 0, 0)  
}  
  
if (screen == 1) // Forsiden
```

```
{  
removeElements();  
fill(255)  
stroke(2)  
rect(0, 0, width, height)  
noStroke()  
fill(color(0,0,0))  
text("Cirkeldiagram over forbrug udregnet over din inkomst", 20, 75)
```

```
forbrug = new circleDiagram(20, height / 2, [forbruginfo[0]/hun*100,  
forbruginfo[1]/hun*100, forbruginfo[2]/hun*100, forbruginfo[3]/hun*100,  
forbruginfo[4]/hun*100, forbruginfo[5]/hun*100, forbruginfo[6]/hun*100], ["Tilovers: ","Bolig:  
", " Mad: "," Faste udgifter: "," Transport: "," Diverse:","Gældsaftvikling: "])
```

```
forbrug.createCircleDiagram()
```

```
kontoimport = new box(20, 20, "Læg budget", 2)
```

kontoimport.createButton()

```
//kontoimport = new box(500, 200, "Importér konto", 3)  
//kontoimport.createButton()
```

```
//fremtidbudget = new box(500, 250, "Fremtidigt budget", 4)  
//fremtidbudget.createButton()
```

```
gaeld = new box(140, 20, "Gældsoversigt", 5)  
gaeld.createButton()
```

```
//hvormaar = new box(500, 350, "Hvornår har jeg råd til...?", 6)  
//hvormaar.createButton()
```

```
image(logo, 900, 450, 275, 120)
```

```
}
```

```
if (screen == 2) // Sign up-siden  
{  
    removeElements();  
    rect(0, 0, width, height)  
    infoinput()
```

```
signmeup = new box(260, 350, "Sign me up now", 1)
signmeup.createButton()
signmeup.button.size(280,50)

textSize(20)
text("First name",20,58)
text("Last name",20,158)
text("Email",20,258)
text("Password",20,358)
text("Repeat password",20,458)

image(logo, 900, 450, 275, 120)

noLoop()

}

if (screen == 3) // Læg budget-siden
{
  removeElements();
  rect(0, 0, width, height)
  budget()
```

```
fill('#222222');

textSize(20)

text("Månedlig indkomst",20,45)

text("Bolig",20,125)

text("Mad",20,205)

text("Faste udgifter",20,285)

text("Transport",20,365)

text("Diverse",20,445)

text("Gældsaftvikling",20,525)

text("Hvad har du tilbage af din inkomst:",450,150)
```

```
line(450, 155, 740, 155);
```

```
// Skabber kanppen der sender dig tilbage til Menu skærm
```

```
backTo1.createButton()

backTo1.button.size(80,50)
```

```
//Skaber kanppen der gemmer din penge forbrugt og gemmer den i et liste

saveinfo = new box(280,270, "Gem data")

saveinfo.createButton()

saveinfo.button.size(60,40)
```

```
image(logo, 900, 450, 275, 120)
```

```
noLoop()
```

```
}
```

```
if (screen == 4) // Gældsoversigt-siden
```

```
{
```

```
removeElements();
```

```
fill((255, 255, 255))
```

```
stroke(0)
```

```
rect(0, 0, width, height)
```

```
fill((0, 0, 0))
```

```
textSize(25)
```

```
text("Gældsoversigt", 50, 50)
```

```
mineLaan = []
```

```
noStroke()  
fill(200)  
rect(75, 75, 400, 500, 20)
```

```
redigerLaan = new box(500, 100, "Rediger låن")  
redigerLaan.createButton()  
redigerLaan.button.size(300, 75)
```

```
nytLaan = new box(500, 200, "Opret nyt lån", 7)  
nytLaan.createButton()  
nytLaan.button.size(300, 75)
```

```
gaeldorfri = new box(825, 200, "Hvornår er jeg gældfri?")  
gaeldorfri.createButton()  
gaeldorfri.button.size(300, 75)
```

```
fill(255)  
stroke(0)  
rect(500, 300, 625, 275)
```

```
if (laan.length > 0)  
{
```

```
for (i = 0; i <= laan.length - 1; i += 1)

{
    fill(120)
    textSize(25)
    text("Lån " + (i+1), 100, 100 + 50 * i)

    textSize(15)
    text("Stiftelse: " + laan[i][0] + " kr.", 175, 90 + 50 * i)
    text("Lån: " + laan[i][1] + " kr.", 175, 105 + 50 * i)
    text("Afdrag: " + laan[i][2] + " kr.", 275, 90 + 50 * i)
    text("Rente: " + laan[i][3] + " %", 275, 105 + 50 * i)
    text("Gebyr: " + laan[i][4] + " kr.", 375, 90 + 50 * i)
    text("Tid: " + laan[i][5] + " ", 375, 105 + 50 * i)

    fill(255)

}

}

//if (mouseX > 75 && mouseX < 475 && mouseY > 75 && mouseY < 105)
//{
//rect(75, 75, 400, 30)
//}
```

```
backTo1.createButton()  
  
backTo1.button.size(80,50)  
  
noStroke()  
  
  
tint(255, 126)  
  
image(logo, 900, 450, 275, 120)  
  
  
}  
  
  
if (screen == 7) // Hvornår har jeg råd til..?  
{  
  
removeElements();  
  
rect(0, 0, width, height)  
  
  
image(logo, 900, 450, 275, 120)  
  
}  
  
  
if (screen == 8) // Opret lån  
{  
  
removeElements();  
  
rect(0, 0, width, height)
```

tint(255, 126)

image(logo, 900, 450, 275, 120)

oprettelsesKrav = ["Stiftelsesgebyr", "Lån", "Afdrag", "Rente", "Gebyr", "Løbetid for
lånet"]

oprettelsesKravInput = []

fill(0)

textSize(25)

text("Oprettelse af lån", 500, 60)

text(" kr.", 805, 100)

text(" kr.", 805, 175)

text(" kr.", 805, 250)

text(" %", 805, 325)

text(" kr.", 805, 400)

text(" måneder", 805, 475)

for (i = 0; i <= 5; i += 1)

{

textSize(20)

text(oprettelsesKrav[i], 400, 100 + 75 * i)

oprettelsesKravInput[i] = new box(650, (100 + 75 * i)-20, "0")

```
oprettelsesKravInput[i].createInput()
```

```
}
```

```
gemLaan = new box(550, 500, "Gem lån")  
gemLaan.createButton()  
gemLaan.button.size(120, 60)
```

```
backTo1.createButton()
```

```
backTo1.button.size(80,50)
```

```
noLoop()
```

```
fill(255)
```

```
}
```

```
}
```

```
// Function infoinput laver inputfilterne for Sign up-siden
```

```
function infoinput()  
{  
    x = 20
```

y = 60

```
firstname = new box(x,y, "")  
lastname = new box(x, y + 100, "")  
email = new box(x, y + 200, "")  
password = new box(x, y + 300, "")  
repeat = new box(x, y + 400, "")
```

```
firstname.createInput()  
lastname.createInput()  
email.createInput()  
password.createInput()  
repeat.createInput()  
  
}
```

// Function budget skaber inputfelterne for budget-siden

function budget()

{

x = 20

y = 50

indkomst = new box(x,y, "")

```
bolig = new box(x, y + 80, "")
```

```
mad = new box(x, y + 160, "")
```

```
fasteUdgifter = new box(x, y + 240, "")
```

```
transport = new box(x, y + 320, "")
```

```
diverse = new box(x, y + 400, "")
```

```
galdsafvikling = new box(x, y + 480, "")
```

```
//resten = new box (515,170,forbrug1)
```

```
indkomst.createInput()
```

```
bolig.createInput()
```

```
mad.createInput()
```

```
fasteUdgifter.createInput()
```

```
transport.createInput()
```

```
diverse.createInput()
```

```
galdsafvikling.createInput()
```

```
//resten.createInput()
```

```
textSize(32);  
  
text(forbrug1 + "Kr", 550, 190);  
  
}  
  
// I draw kalder vi kun check-funktionen  
  
function draw() {  
  
background(220);  
  
check()  
  
console.log(screen)  
  
}
```

```
// I klassen circleDiagram skaber vi et cirkeldiagram for forbrugernes afgifter.  
  
class circleDiagram  
  
{  
  
//I constructoren definerer vi attributerne.  
  
constructor(x, y, split, label)  
  
{
```

```
this.x = x

this.y = y

this.split = sort(split)

print(this.split)

this.label = label

this.colors = [color(245, 66, 66), color(245, 161, 66), color(245, 239, 66), color(66, 245, 69), color(66, 75, 245), color(255, 10, 247), color(10, 235, 255)]

this.lastAngle = 0

this.amount = this.split.length

}

// I createCircleDiagram skaber vi selve diagrammet.

createCircleDiagram()

{

    if (forbrug1 >= 0)

    {

        for (this.i = 0; this.i <= this.amount - 1; this.i += 1)

        {

            forbrug.split = [nfc(forbruginfo[0]/hun*100,2), nfc(forbruginfo[1]/hun*100,2),
nfc(forbruginfo[2]/hun*100,2), nfc(forbruginfo[3]/hun*100,2), nfc(forbruginfo[4]/hun*100,2),
nfc(forbruginfo[5]/hun*100,2), nfc(forbruginfo[6]/hun*100,2)]

            this.angle = radians(360*(this.split[this.i]/100))

            fill(this.colors[this.i])

            arc(this.x + 150, this.y, 300, 300, this.lastAngle, this.lastAngle + this.angle)

            this.lastAngle += this.angle
        }
    }
}
```

```
textSize(15)

textFont('Calibri');

 textStyle(BOLD)

//text(this.label[this.i], (((this.x + 400) / this.amount) * this.i) + 25, 500)

if (this.i < 3)

{

text(this.label[this.i] + this.split[this.i] + "%", ((this.x + 400)/3) * this.i + 50, 500)

}

else

{

text(this.label[this.i] + this.split[this.i] + "%", ((this.x + 400)/3) * (this.i - 3), 550)

}

fill(color(255,255,255))

}

}

else

{

fill(color(245, 66, 75))

text("Dit forbrug er større end din indkomst. Ændr dit forbrug.", this.x, this.y)

fill(255)

}
```

}

}

// Funktionen mousePressed tjekker, om vi trykker på de forskellige knapper, vi bruger igennem programet. Dette gør vi ved hjælp af if-statements, som kan ses ned under.

```
function mousePressed(){
```

```
    if (screen == 2 && savefile == true && mouseX > 260 && mouseX < 540 && mouseY > 350 && mouseY < 400 && mouseIsPressed)
```

```
{
```

```
    jsonFil()
```

```
    savefile = false;
```

```
    information = [firstname.input.value() ,lastname.input.value(), email.input.value(),  
password.input.value()]
```

```
    screen = 1
```

```
loop();
```

```
}
```

```
if (screen == 3 && mouseX > 1100 && mouseX < 1200 && mouseY > 20 && mouseY < 70 && mouselsPressed)
```

```
{
```

```
    screen = 1
```

```
    loop();
```

```
}
```

```
if ( screen == 3 && mouseX > 280 && mouseX < 340 && mouseY > 270 && mouseY < 310 && mouselsPressed)
```

```
{
```

```
// Sakber et liste med informationer omkring dit månedlig indkomst
```

```
forbruginfo = [float(indkomst.input.value()), float(bolig.input.value()),  
float(mad.input.value()), float(fasteUdgifter.input.value()), float(transport.input.value()),  
float(diverse.input.value()), float(galdsafvikling.input.value())]
```

```
console.log(forbruginfo)
```

```
// Minusser alle din informationer omkring din mådelig forburg fra dit månedelig  
indskomst således at, du få information omkring hvor, mange penge du har tilbage.
```

```
forbrug1 = forbruginfo[0] - forbruginfo[1] - forbruginfo[2] - forbruginfo[3] -  
forbruginfo[4] - forbruginfo[5] - forbruginfo[6]
```

//kalder budget funktionene for at opdater vore cirkel diagram.

budget()

//skaber et variebel der bruges til at opsamle boede din mådedelig forbrug samt dit indkomst.

hun = forbruginfo[0] + forbruginfo[1] + forbruginfo[2] + forbruginfo[3] + forbruginfo[4]
+ forbruginfo[5] + forbruginfo[6]

//Skaber et liste hvor i den bliver der regnet din mådedelig forbrug og indkomst om til procent dele. Ud fra "hun" det ville sige ud fra den samlede økonimi.

forbrug.split = [nfc(forbruginfo[0]/hun*100), nfc(forbruginfo[1]/hun*100),
nfc(forbruginfo[2]/hun*100), nfc(forbruginfo[3]/hun*100), nfc(forbruginfo[4]/hun*100),
nfc(forbruginfo[5]/hun*100), nfc(forbruginfo[6]/hun*100,10)]

console.log(forbrug.split)

}

if (screen == 1 && mouseX > 20 && mouseX < 70 && mouseY > 20 && mouseY < 40 && mouseIsPressed)

{

screen = 3

loop();

}

if (screen == 8 && mouseX > 550 && mouseX < 670 && mouseY > 500 && mouseY < 560 && mouseIsPressed)

{

```
laan.push([oprettelsesKravInput[0].input.value(), oprettelsesKravInput[1].input.value(),
oprettelsesKravInput[2].input.value(), oprettelsesKravInput[3].input.value(),
oprettelsesKravInput[4].input.value(), oprettelsesKravInput[5].input.value()])
```

```
console.log(laan)
```

```
screen = 4
```

```
loop();
```

```
}
```

```
if (screen == 8 && mouseX > 1100 && mouseX < 1200 && mouseY > 20 && mouseY <
70 && mouseIsPressed)
```

```
{
```

```
screen = 4
```

```
loop();
```

```
}
```

```
if (screen == 4 && mouseX > 1100 && mouseX < 1200 && mouseY > 20 && mouseY <
70 && mouseIsPressed)
```

```
{
```

```
screen = 1
```

```
textSize(25)
```

```
}
```

```
}
```

11.2 Litteraturliste

Idé til gælds statistik:

<https://laaneberegneren.dk/laaneberegner/>

Renters rente formel:

<https://bestaamatematik.dk/faa-styr-paa-renters-rente-beregning-kapitalfremskrivning/>

Logo:

https://www.wix.com/logobrand/wix-logo-maker?utm_source=google&utm_medium=cpc&utm_campaign=1624436120^62150769059&experiment_id=wix%20logo^e^310341988693^&gclid=Cj0KCQjw-_j1BRDkARIIsAJcfmTGR3qpiFweEL-gSPDsr2j2vFaSfGQ8G054X9MPRQ3huGvbQMqq1mEaAiFIEALw_wcB