Indhold

1 Det binære talsystem

Modsætningen til digital er analog. Analog er en trinløs eller glidende overgang fra en tilstand til en anden. Digital er en trinvis overgang, fra 0 til 1. Hvor en analog værdi kan være mange værdier er der kun to digitale værdier, 0 og 1.

Det kan godt være abstrakt at skulle forstå, at Snapchat, Instagram og Netflix i bund og grund kun er en række af nuller og ettaller. Men lad os starte et sted som i måske kender. Regnbuens spektrum strækker sig fra blå over grøn, gul til rød. Står vi og iagttager regnbuen vil vi kunne se tusinde af farver. For at computeren kan forstå det, bliver vi nød til at give hver enkel farve et unikt nummer. Hurtigt vil vi kunne se, at vores liste over numre vokser og bliver lang. Vi løber hurtigt ind i problemet, at der ikke er mere plads på vores A4 side. Vi løber tør for plads fordi tal fylder! Etterne flyder 1 ciffer, tierne fylder 2 cifre, hundrederne fylder 3 cifre og så videre. Vi har altså ikke nok plads/hukommelse, til at registrere alle farver på et stykke papir og vi må derfor beslutte os for hvor mange stykker papir vi vil bruge på at registrere vores farver. Der er 40 linjer på et stykke A4 papir derfor vil to sider give 80 forskellige farver og tre sider give 120 farver. Sådan er det også i computeren. Men da computeren er digital har vi kun adgang til ciffrene 0 og 1. Lad os for et øjeblik vende tilbage til titalssystemet. 0 er ingen ting, men placerer vi det bagved et andet ciffer tidobler vi cifferets værdi. Det betyder, at det ikke er cifferet, men cifferets placering som er afgørende for dets værdi. Sjovt nok læser vi tal fra højre mod venstre og ikke som vi læser tekster, fra venstre mod højre. Så når vi taler om tal, siger vi, at den første plads er alle etterne, den anden plads er aller tierne, den tredje plads er alle hundrederne osv. Så titals systemet består af 10 frskellige cifre 0-9, og det er cifrets position som bestemmer dets værdi.
Dette kan vi udtrykke matematisk. 10 er grundtallet i talsystemet, eksponenten angiver tallets placering/værdi (0=ettere, 1=tierne, 2= hundrederne) og 1 er cifferet vi ønsker at kende værdien for. Cifrets værdi er $10^n * tallet$

Forstil dig nu, at du i stedet for 10 cifre kun har to cifre, 0 og 1. Det fungerer på helt samme måde, men i stedet for ettere, tiere, hundrede og tusinder. Har vi alle 1'er, 2'er,4'er,8'er, 16'er, 32'er, 64'er, 128'er, 256'er, 512'er, 1024'er også videre.Det er altså cifrets placering som er afgærende for tallets værdi. F.eks. er 1010 binært lig med den decimale værdi 10. 1'er er der ikke nogen af, 2'er er der en af, 4'er er der ikke nogen af, men der er én 8'er: $2 + 8 = 10$.

Dette kan vi igen udtrykke matematisk. 2 er grundtallet i talsystemet, eksponenten angiver tallets placering/værdi (0=1'er, 1=2'er, 2= 4'er, 3=8'er) og 1 er cifferet vi ønsker at kender værdien for. Da vi i det binære talsystem

1022				
	tusinderne	hundrederne	tierne	ettere
0	•	$10^2 * 0 = 0$	•	•
1	$10^3 * 1 = 1000$	•	•	•
2	•	•	$10^1 * 2 = 20$	$10^1 * 2 = 2$
3	•	•	•	•
4	•	•	•	•
5	•	•	•	•
6	•	•	•	•
7	•	•	•	•
8	•	•	•	•
9	•	•	•	•

Figur 1: En beregning af værdien 1022 i titalssystemet

10				
	8'er	4'er	2'er	1'er
0	•	•	•	•
1	$2^3 = 8$	•	$2^1 = 2$	

Figur 2: En beregning af værdien 10 i totalssystemet

ikke har mere end to cifre, er der ingen grund til at gange med cifrets værdi. Cifrets værdi er derfor $= 2^n$

1.1 Bytes og Bits

Hvis du kan forholde dig til analogen om der på en A4 side er 40 linjer, så vil du måske forstå at en byte har 8 linjer eller celler. Vi kalder en celle for en bit. Hver celle repræsenterer en fordobling af den foregående værdi.

Hvis du kigger på figur 3, vil du se at der er 8 kolonner og to rækker. Den øverste række viser alle 1'erne, 2'erne, 4'erne, 8'erne osv. Rækken neden under viser om bitten er sat. Vi lægger alle værdier sammen på celler hvor bitten er sat for at finde den decimale værdi. I mit eksempel er det tilfældet for cellen som repræsentere væriden 1 og cellen med værdien 4. Derfor er den decimale værdi: $1 + 4 = 5$

128	64	32	16	8	4	2	1
0	0	0	0	0	1	0	1

Figur 3: En byte består af 8 bits, her er værdien 5

128	64	32	16	8	4	2	1
0	0	1	0	1	0	1	0

Figur 4: En byte med værdien 42

1.2 Opave

Der var en gang for længe længe side. Før man kendte til Snapchat, Facebook og Instagram, ja faktisk før internettet og telefoni! Da boede der, i en lille skøn dal, en ung smuk kvinde som var gift med en tyk, grim mand. Manden var gammel og tjente til dagen af vejen ved at slibe knive i byen. Hver dag, når klokken slog 8 slag, stod manden op og besluttede sig for, hvornår han ville drage ind til byen for at slibe knive. Og hver dag, når manden stod op, fortalte han kvinden hvornår han ville tage afsted. Nogle gange kl 9 andre gange kl 11. Aldrig var to dage ens. Kvinden var forelsket. Ikke i den gamle mand, men i en ung smuk og stæk mand som hyrder får oppe i bjergene. Hyrden kunne gå oppe i bjergene, og med længsel se ned på gården med de fire små vinduer, hvor den unge smukke kvinde og den gamle tykke mand boede. Kvinden havde en aftale med hyrden. Hver dag, når manden stod op og fortalte hvornår han ville tage afsted, så ville sætte lys i vinduerne for på den måde at signalere til hyrden hvornår banen var fri. Satte hun lys i det første vindue skulle han komme kl 1. Satte hun lys i det andet vindue, skulle han komme kl 2. Satte hun lys i det tredje vindue skulle han komme kl: 4 og var der lys i det fjerde vindue, var der fri bare kl.8. Når hyrden så kom ned fra bjerget, havde de en time til at hygge sig i. Derfor hedder det den dag i dag hyrdetimen!

I hvilke vinduer skulle kvinden tænde lys, hvis hyrden skulle komme kl 3 eller kl 5 eller kl 10?

1.3 Den maksimale værdi for en byte

En byte består normalt af 8 bit. Den maksimale værdi en byte kan repræsentere er: $128+64+32+16+8+4+2+1 = 255$ men antallet af forskællige værdier er 256! Fordi 0 tæller også med som værdi. Det betyder, at vi i en byte på 8 bit kan vælge, at hver værdi kan repræsentere en af regnbuen farver. Det gør det muligt at have 256 forskellige farver.

1.4 Paritets bit

Hvis den første bit (den længst til højre) i en byte er sat, så ved vi, at tallet er ulige. Hvis den ikke er sat, er tallet lige. Vi kalder denne bit for paritets

bit eller the least significant bit.

1.5 The most significant bit

Hvis en byte er signed, så betyder det at den kan bruges til både positive og negative værdier. The most significant bit, er den bit som har den største værdi, dvs. den bit som er længst til venstre. Denne bit benyttes til at angive om det er et positivt eller negativ tal. Der med er den maksimale værdi 127 og den mindste -128 og antallet af forskellige værdier 256. Du kan afprøve det med følgende lille java program.

```
byte b=127;
for (int i = 0; i<256; i++){
  b++;
  println(b);
}
```

Figur 5: Et eksempel på min og max værdi af en byte

2 Primitive datatyper

Det er nemt at se forskel på primitive og ikke primitive[1] datatyper i Java. Primitive datatyper staves med lille begyndelses bogstav, ikke primitive datatyper staves med stort begyndelses bogstav. Primitive datatyper kan repræsenteres i en eller flere bytes og kan sammenlignes med en operator '=='. Ikke primitive datatyper er klasser som har tilknyttet funktioner. Hver gang man møder en ikke primitiv datatype, bør man se i dokumentationen, om der er metoder som man kan bruge. Ikke primitive datatyper kan ikke sammenlignes direkte men kun ved hjælp af en funktion fx. equals().

Tabel 1: Liste over datatyper.

Ikke primitive	Primitive
String	integer
Array	float
Klasser	char
Interfaces	boolean
	byte
	short
	long
	double

Denne opgave handler om at forstå de forskellige datatyper.

2.1 Opgave

Undersøg for hver primitiv datatype, hvor meget plads (i bytes) hver datatype bruger. Det kan du finde her: https://data-flair.training/blogs/java-data-types/. Eksempel: Datatypen integer fylder 4 bytes og at den maksimale værdi er:$2^{31} - 1 = 2.147.483.648$. Noter dine resultater for hver af de 8 primitive datatyper.

2.2 Opgave

Skriv et program som beviser hvilke minimums- og maksimumsværdier for de primitive datatype kan indeholde. Find evt. inspiration i programmet i figur 5 på side: 6. Noter dine resultater for hver af de 8 primitive datatyper.

[1] https://data-flair.training/blogs/java-data-types/

2.3 Opgave

De to datatyper float og double er ikke lige nøjagtige. Det kan de se ved følgende opgave: Hvad giver kvadratroden af 2 gange med kvadratroden af 2? ($\sqrt{2} * \sqrt{2}$). Lav et først et program med sqrt() som returnerer en float og herefter med Math.sqrt() som returnerer en double.

- Slå funktionen sqrt() op i Processings dokumentation. Deklarer en variabel f af type float og initier med værdien af sqrt(2). Brug kommandoen println(f); til at udskrive værdien af f.
- Du kan ikke slå funktionen Math.sqrt() op i Processings dokumentation. Men den fungerer på helt samme måde som sqrt(), den returnerer bare en double. Deklarer en variabel d af type double og initier med værdien af Math.sqrt(2). Brug kommandoen println(d); til at udskrive værdien af d.

Forklar forskellen på de to resultater.

3 Ikke-primitive datatyper

Vi skal i dette kapitel arbejde med ikke-primitive datatyper. Det gælder for de primitive datatyper at de kan repræsenteres i en byte. Ikke primitive datatyper kræver meget mere plads. Primitive datatyper er foruddefineret i java, det er ikke primitive datatyper ikke, på nær String. Ikke primitive datatyper kan bruges til at kalde funktioner og bestemte operationer.

Den ikke primitive datatype String, er en række af karrakterer (Chars). Der er til datatypen knyttet en række metoder:

- toUpperCase() Gør alle karakterer til store bogstaver
- toLowerCase() Gør alle karakterer til små bogstaver
- substring() Returnerer en ny streng som er en del at den orginale streng.
- length() Returnerer en heltalsværdi som repræsenterer antallet af karakterer i strengen.
- indexOf() Returnerer indeks af den første forekomst af substreng i strengen.
- equals() Sammenligner to strenge
- charAt() Returnerer karakteren på den af index angivne plads.

Eksempler på andre ikke primitive datatyper: Strings, Arrays, Classes, Interface, etc.

3.1 Opgave

Lav et program som klipper i vores streng. Den skal tage fra position 83 og til slut. Udskriv den nye streng.

3.2 Opgave

Lav et program som klipper i vores streng. Den skal tage fra position 83 og til 85. Udskriv den nye streng.

3.3 Opgave

Lav hele sætningen om til store bogstaver og udskrive den.

3.4 Opgave

Lav hele sætningen om til små bogstaver og udskriv den.

3.5 Opgave

Lav opgaver på codingbat.com: String 1

4 Sådan spiser man en elefant

Når vi står foran en udfordring, hvor der ikke er en oplagt løsning, kalder vi det for et komplekst problem. Komplekse problemer kan løses ved at vi deler problemet op i en række af trivielle problemer. Et trivielt problem er et problem, hvor løsningen er åbenlys. Så løser vi de trivielle problemer i rækkefølge og løser på den måde det komplekse problem.

5 Sekventiel/procedural -programmering

Læs kapitel 3.1 i Systimebogen.
Denne opgave handler om sekventiel og procedural programmering. Sekventiel programmering, betyder at instruktionernes rækkefølge ikke er ligegyldig. Procedural programmering betyder at vi kan gentage en sekvens af kode flere gangen selv om vi kun skriver koden én gang. Vi kan altså opdele vores program i forskellige funktioner og kalde funktionen når vi har brug for den. Det ville være smart med en procedure som kan beregne en vares pris med moms.

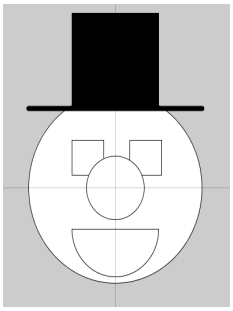
Det vigtige i denne opgave er, at du skal lære at bruge dokumentationen til processing/Java.

Herudover vil du blive introduceret til rutediagrammer. Et værktøj som skal hjælpe dig med at strukturere dine programmer.

Der er altså hele tre ting som kræver din opmærksomhed!

5.1 Opgave

Du skal lave en kopi af min tegnig: opg1-hoejhat.pdf. Det primære fokus i denne opgave er, at bruge dokumentationen i processing. Der findes i processing en lang række forud definerede procedurer/funktioner. Vi kan se at det er en funktion fordi er altid er () bagefter. For eksempel size(); Size er en funktion som kræver to parameter, brede og højde. size(400,600); Men hvordan finder man ud af hvor mange parameter og hvilke man kan bruge til en funktion? Det gør man ved at kigge i dokumentationen til processing. Du finder alle funktioner i processings reference guide: processing.org/reference.



Figur 6: Højhat

Du kan bruge disse otte instruktioner for at kunne lave tegningen.

- size();
- line();
- strokeWeight();
- rect();
- square();
- circle();
- arc();
- fill();

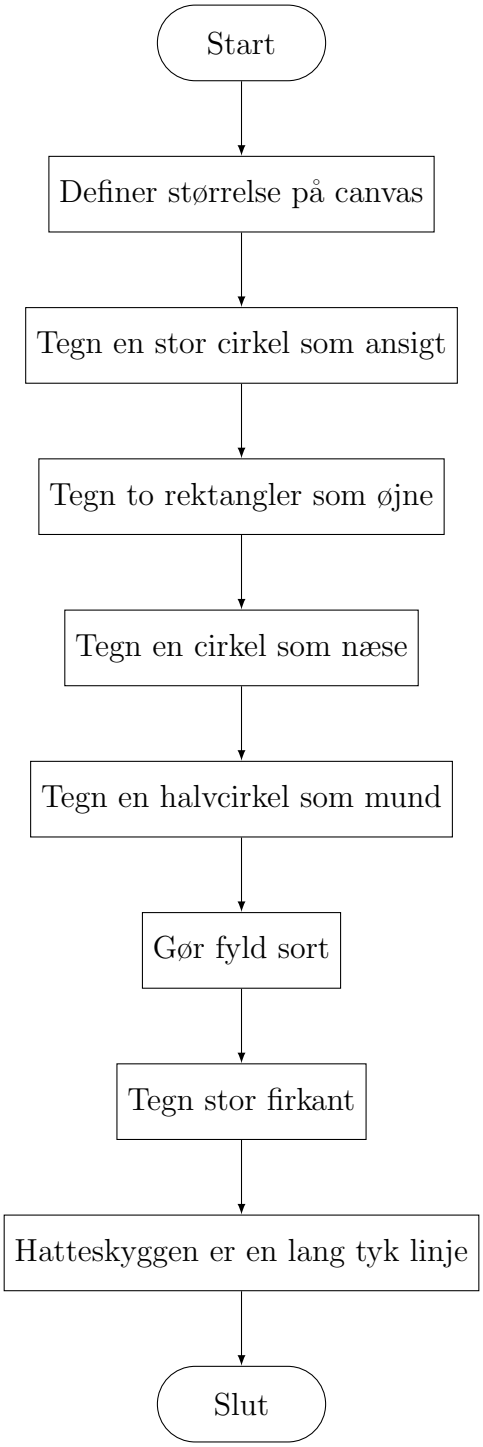
Brug Processings dokumentation: processing.org/reference, for at finde ud af hvilke parameter de forskellige funktioner skal have.

- canvas (vindue som processing åbner) kan have størrelsen $400, 600$
- strokeWeight() er tykkelsen på stregen.
- fill() udfylder figuren med en RGB-farve.

Husk at koordinaterne $0, 0$ er øverste venstrehjørne (normalt vil det være nederste venstre) og er efter princippet: "hen ad vejen, ned til stegen". X,Y.

Tip: Vi er vant til at bruge grader til at måle en vinkel, men kan også bruge radianer. Denne artikel forklarer radianer: webmatematik.dk/lektioner/matematik-a/trigonometri

Brug rutediagrammet i figur 7 for at skrive koden:



Figur 7: Rutediagram

Hvad sker der, hvis du bytter om på rækkefølgen? Altså hvis du starter med øjne,næse og mund og så tegner ansigtet.

5.2 Opgave

Du har nu en fornemmelse af hvad sekventiel programmering er. Men en af styrkerne i Java er, at det understøtter produceral programmering. Hver procedure programmeres sekventielt. Vi kalder også procedurer for funktioner. I Proccesing skal der altid være to funktioner. Setup og draw. Setup bruger vi til f.eks. at sætte størrelse på canvas eller andre initieringer. Draw er hoveddelen. Proceduren (vi kalder det også en funktion) looper 60 gange i sekundet. Vi kan selv definere flere funktioner.

Tilføj nu følgende linjer til dit program og flyt alt dit kode ned i proceduren/funktionen hoved.

```
void setup(){
  size(480, 120);
}

void draw(){
  head();
}

void head(){
  // Skriv din kode her. I mellem de to tuborg paranteser.
}
```

5.3 Opgave

Del dit program yderligere op. Således at én funktion tegner hatten. En anden tegner øjne, en tegner munden og den sidste tegner ansigtet. Kald funktionerne for: void hat(){} ,void eyes(){} ,void mouth(){}, void face(){}, husk at fjerne funktionen head(), når du er færdig.

5.4 Opgave

Gør nu dit canvas så stort at der er plads til to hoveder ved siden af hinanden. Tilføj to parametere til dine funktioner, en float x og en float y koordinat, udfra hvilke du kan tegne alle dine elementer af ansigtet. Ret dine linjer til så de tager udgangspunk i dine x og y koordinater. Er du rigtig god, kaler du dine funktioner med de samme værdier!

5.5 Opgave

Tegn figuren færdig med krop, arme og ben i hver deres funktion.

6 Primitiv Animation

Det vigtige i denne opgave er, at du skal lære at bruge dokumentationen til processing/Java. Herudover skal du bruge rutediagram til at strukturere din kode. Tænk på animation som stop motion teknik.

Du skal bruge nogle nye instruktioner

- frameRate()
- frameCount()
- noLoop()
- rotate()
- translate()
- popMatrix()
- pullMatrix()

Brug Processings dokumentation: processing.org/reference, for at finde ud af, hvad de forskellige funktioner gør og hvilke parameter de forskellige funktioner skal have.

6.1 Opgave

Lav et program, hvor et hjul, med minimum tre eger, ruller over skærmen fra venstre mod højre. Start med at lave et rute diagram.

6.2 Opgave

Udvid programmet så når hjulet forsvinder i højre side, dukker det op på den anden side igen.

6.3 Opgave

Lav programmet om så når hjulet rammer væggen, at det stopper og ruller tilbage hvor det kom fra.

6.4 Opgave

Lav et program som lave en primitiv animation af en mand som kan gå. Start med at lave et rute diagram.

Du kan finde inspiration i mine to eksempler på Github.

7 Løkker

Læs om while-løkker i kapitel 3.3 og om for-løkker i kapitel 3.4. En løkke er en sekvens som skal gentages et antal gange. Kender vi antalltet af itereationer benytter vi en for-løkke, kender vi ikke antallet af iterationer benytter vi en while løkke.

- Vask disse 5 tallerkener op. For-løkke
- Hvor længe skal jeg lede efter min nøgle? Til den er fundet. While løkke.

7.1 While-Løkke

While loop bruger vi når vi ikke ved hvor mange iterationer vi skal bruge. En iteration er hver gang løkken gentager sig selv.

Vi kan oversætte det engelske ord while, med så længe.

```
boolean found=false;

while(!found){
        // set found til true for at stoppe

}
```

Figur 8: While løkke

Hvis vi opretter en variabel af typen boolean og kalder den found og initierer den med værdien falsk. Så kan vi benytte os af en negering (!)(ikke) og bruge betingelsen while(!found). Det læses "while not found". Hvilket giver god mening at vi skal blive ved med at lede efter bilnøglen til vi finde den. Tænker vi over det skal while-løkken have en sand betingelse, men værdien for found er falsk. Men !found gør den sand. I det øjeblik vi finder hvad vi leder efter sætter vi found til at være true og så vil !found blive false og løkken stopper.

7.2 For-løkke

For-løkker bruges når vi kender antallet af iterationer. For eksemple hvis vi skal løbe alle bogstaver igennem i en streng eller objekter i et array. Fordelen ved en for-løkker er at der er indbygget en tæller som vi kan bruge.

```
for (int i=0;i<str.lenght();i=i+1){
        // kode som gentages
}
```

Figur 9: For-løkke

En for-løkke består af tre dele. Del 1 er en deklarering og initiering af vores index/tæller. Vi kalder index for i. Har vi brug for flere indlejrede løkker, følger vi alfabetet og kalder den næste j så k etc. Del 2 er den betingelse som skal være opfyldt for at løkken bliver udført. Betingelsen knytter sig typisk til vores index i og længden af den streng eller array vi benytter. For eksempel: $i <= 10$ eller $i < str.length()$. Del tre beskriver den handling som skal ske med i efter hver iteration. Vi kan tælle i op med 1; i++. Eller trække en fra med $i--$.

7.3 Enhanced loop

Enhanced loop benyttes til at iterere gennem en liste af objekter og udfører en instruktion for hvert element i listen.

```
for (Particle part : particles) {
  part.display();
}
```

Figur 10: Enhanced-løkke

Vi har talt om at string har et index. Med en løkke kan vi gennemløbe alle positioner af en streg og se på det enkelte bogstav på den i'ne-plads.

Slå op i dokumentationen for processing og læs om string(). Nederst er der en række metoder som kan benyttes sammen med en string. Nå en datatype starter med stort bogstav, vil der altid være metoder i kan bruge. Man bruger metoderne sammen med variabelnavnet. F.eks. hvis str er defineret som datatype String, så kan man skrive: str.charAt(i);

7.4 Opgave

Lav henholdsvis en for-løkke og en while-løkken som udskriver de 5 første bogstaver i sætningen "Hej med dig!". Tip, brug funktionen charAt() sammen

med din tæller i.

7.5 Opgave

Lav henholdsvis en for-løkke og en while-løkken som skal finde alle e'er i sætningen: "Dette er en sætning som indeholder mange e'er. Men hvor mange er der?"Løkken skal udskrive alle e'er og udskrive hvor mange e'er som er fundet. Tilsidst udskrives længden af overstående sætning.

7.6 Opgave

Du skal lave et while-loop som sammenligner alle bogstaver i en streng med et 'Z'. Når betingelsen er sand skal løkken stoppe. Udskriv "Fundet"til consollen sammen med index for bogstavet 'Z'. Brug denne String "I Afrika lever der mange dyr på savannen, et af den er zebraen. Zebraen er et flok dyr."Tip: du kan med fordel bruge str.toLowerCase() hvis din streng hedder str, ellers skal du lede efter både 'z' og 'Z'. For at sammenligne en character kan du bruge betingelsen: str.charAt(i)=='Z' eller str.charAt(i)=='z'.

7.7 Opgave

Denne opgaver ligner den tidligere men der er forskel. Du skal lave et for-loop som tæller alle 'Z' eller 'z' i en streng. Udskriv "antal a z'er: "til consollen sammen med antallet af z'er. Brug denne String "I Afrika lever der mange dyr på savannen, et af den er zebraen. Zebraen er et flok dyr."

7.8 Opgave

Omskriv nu de to opgaver så du bruger for-løkke i while opgaven og visa versa.

8 Funktioner

Læs om funktioner i kapitel 3.5.
Vi skal arbejde med funktioner. Du kender dem fra matematikken $f(x)$, hvor en funktion beregner/returnerer en værdi. Vi bruger funktioner når vi skal udføre den samme sekvens flere gange, med forskellige variabler. Funktioner i Java, består af 4 dele.

Navn: Funktioner skal navngives med et ikke reserveret ord (ord som er brugt i forvejen), men hvor navnet er sigende for funktionens funktion. I

mit eksempel er navenet "minFunktion". Vi bruger navnet når vi skal bruge funktionen.

Returdatatype: Funktioner skal deklareres efter returdatatype. Funktionen kunne returnere en int, float, string eller char, men altid kun én ting. Hvis funktionen ikke returnerer noget, skal den defineres som void.

Parameter: Funktionen kan modtage en lang række forskellige parameter. En funktions paramenter skal angives i parantesen efternavenet samtidig med at vi deklarerer datatypen. Man kan deklarere mange parametere til sin funktion. Parameter er kun gyldige lokalt. Det vil sige at du kan ikke bruge en variabel du har deklareret i andre funktioner uden at skulle deklarere dem igen.

Kode: En funktions kode skal skrives imellem de to tuborgparanteser .

```
float pris=20.0f;

void setup(){
        size(400,400);
        pris=beregnMoms(pris);
        udskrivPris(pris);
}

//funktion som beregner moms
float beregnMoms(float pris){
        return pris*1.25;
}
// funktion som udskriver pris
void udskrivPris(float pris){
        println("Pris incl moms: "+pris);
}
```

Figur 11: Funktion

8.1 Opgave

Skriv et program hvor dit canvas er 800x800. Opret en funktion som tegner en cirkel i midten af dit canvas. Cirklen skal være rød. Navngiv din funktion så den fortæller hvad funktionen gør. Opret en funktion som tegner en firkant. Firkanten skal være blå. Navngiv din funktion så den fortæller hvad funktionen gør.

8.2 Opgave

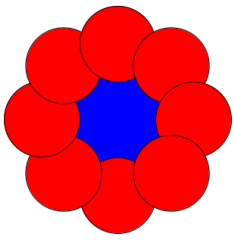
Tilpas din funktion så den tegner 8 cirkler. Brug et for loop og brug dit index i til at gange din x koordinat i cirklen, således at din cirkel ikke bliver tegnet det samme sted.

```
for (int i = 0; i < 10; i++){
// flyt cirkel med brug af i.
   circle(0+i*20,0,20);
}
```

Figur 12: For-løkke

8.3 Opgave

Lave en blomst af cirkler. Placer en firkant i midten og lav kronbladene af 8 cirkler. Du kan måske bruge funktionerne pushMatrix(), popMatrix, translate() og rotate()?



Figur 13: Blomst

9 Betingelser/Forgreninger

Læs om Betingelser/forgreninger i kapitel 3.2 i Systime bogen.

9.1 Sammenlignings operatorer

Sammenlignings operatorer		
Operator	Beskrivelse	Eksempel
$<$	Mindre end:	$a < b$
$<=$	Mindre end eller lig med:	$a <= b$
$>$	Større end:	$a > b$
$>=$	Større end eller lig med:	$a >= b$
$==$	Er lig med:	$a == b$
$!=$	Forskellig fra:	$a! = b$

9.2 Aritmetiske operatorer

Aritmetiske operatorer			
Operator	Navn	Beskrivelse	Eksempel
+	Addition	Lægger to værdier sammen.	x+y
−	Subtraktion	Trækker to værdier fra hianden	x - y
*	Multiplikation	Ganger to værdier	x * y
/	Division	Dividerer en værdi med den anden	x / y
++	Inkrement	Øger værdien af en variabel med 1	x++
−−	Dekrement	Mindsker værdien af en variabel med 1	x−−

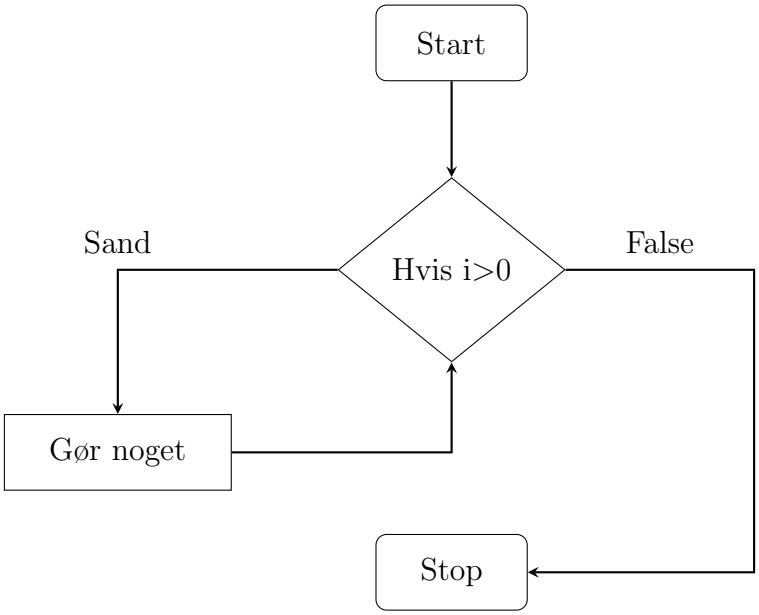
9.3 Initierings operatorer

Initierings operatorer		
Operator	Eksempel	Det samme som
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
>>=	$x >>= 3$	$x = x >> 3$
<<=	$x <<= 3$	$x = x << 3$

9.4 Logiske operatorer

Logiske operatorer			
Operator	Navn	Beskrivelse	Eksempel
&&	And	Sand hvis begge betingelser er sande	$x < 5 \&\& x < 10$
||	Or	Sand hvis en betingelser er sand	$x < 5 \|\| x < 4$
!	Not	Negation, returnerer falsk hvis betingelsen er sand	$!(x < 5 \&\& x < 10)$

```
if (betingelse) {
// Den sekvens af kode som løkken skal udføre hvis betingelsen er sand.
}
```



Figur 14: Rutediagram løkke

9.5 Opgave

Lav henholdsvis en for-løkke og en while-løkken som skal gennem løbes 30 gange. Start med i=0; og tæl i op hver gang du løber gennem løkken. Den skal kun udskrive i, når i er mellem værdien 10 til og med 20.

9.6 Opgave

Lav et program som udskriver bogstaverne fra position: 39,19,41,6,4, 16,6,4,16,35,95,41,48,95 til skærmen. Brug sætningen "Løkken skal udskrive alle e'er og tilsidst udskrive hvor mange e'er som er fundet. Ja, så er det rigtigt :)"

10 Ordliste

Vi trækker håndbremsen, stopper op, og reflekterer over hvad vi har lært 'so far'.

I har lavet en liste med ord og udtryk. Tjek din liste om du mangler disse ord.

- Instruktion - enkelt programlinije eller kommando
- Funktion - en isoleret sekvens som returnerer et produkt af en funktion
- Kontrolstruktur
  - Sekvens - en række af instruktioner
  - Forgrening/Betingelser
  - Løkke
- Variabel - værdi som kan ændres under afviling
- Parameter - en variabel som sendes til en funktion
- Konstant - En værdi som ikke kan ændres
- Deklaration - bestemme datatype til en variabel
- Initiering - tildele en værdi til en variabel
- Cammelback notation - en måde at skrive variabel navne på.

11 Optimering af kode og brug af funktioner

Opgaven handler om optimering af kode og brug af funktioner.

Læs og forstå koden i mappen atArbejdeMedEnStreng.

11.1 Opgave 1

Optimer og omskriv den kode du finder i mappen atArbejdeMedEnStreng, herunder opret relevante funktioner og optimer output. f.eks.: lav deklaration og initiering i samme linje. lav while om til for loop. slet variabler du ikke skal bruge.

11.2 Opgave 2

Opret nu to funktioner. En som kan udskrive et tal, og en som kan udskrive et bogstav. Brug disse funktioner i koden til dit output. void printCharToConsole(char c) void printResultToConsole(int i) Reglen er, at så snart man har mere en 7 linjer kode, så skal man overveje at lave en funktion.

Husk, når du vil bruge funktioner så skal det hele være funktioner. void setup() void draw()

12 Funktioner, arrays og datatyper

Denne opgave handler om at bruge funktioner, arrays og datatyper.

12.1 Opgave

1. Lav et program som kan beregne din alder i hele år. Følg min psudokode:

   (a) setup: canvas størrelse og noLooop;

   (b) draw - mit hovedprogram Opret integerværdi med et årstal - myBirthYear Kald funktionen howOld() med parameteren myBirthYear, udskriv returværdigen fra howOld().

   (c) lav funktionen howOld: int howOld(int myBirthYear) hent det aktuelle årstal ved at bruge processing funktionen year() træk nu de to tal fra hianden. returner resultatet

2. Udvid programmet til også at kunne regne din alder ud i hele måneder. brug følgende psudokode:

   I funtionen draw() - hovedprogrammet: Opret en konstant med din fødselsmåned - int myBirthMonth = 3 (marts);
   Kald funktionen howOldInMonths() med parameteren myBirthMonth udskriv resultatet af howOldInMonths()

3. Opret en funktion howOldInMonths() som modtager parameteren int myBirthMonth og myBirthYear, og returnerer en int.

   hvis myBirthMonth er større end month() så har du ikke haft fødselsdag så skal du returnere month()+((howOld()-1)*12)

   hvis month() er større en myBirthMonth() så har du haft fødselsdag så skal du returnere month()-myBirthMonth+(howOld()*12)

   Funktionen skal returnere en integer værdi med antallet af måneder man er gammel.

4. Opret et array af int, som indeholder antallet af dage i en måned.Lav et loop som kan beregne summen af dage mellem to datoer. Start med at lave psudokode.

13 Fejlfinding

Denne opgave handler om at finde og rette fejl.
Vi arbejder med datatyper og løkker.

13.1 Debugger

13.2 Opgave 1

Opgaverne står i filerne.

14 OOP

OOP betyder objekt orienteret programmering.

15 Opgave

Læringsmål: Du skal i denne opgave arbejde med klasser, arrays og input fra mus. På github, kan du finde et program house, med en klasse Room. Programmet tegner et hus. For hvert objekt af klassen Room, du opretter, tegner programmet et ekstra værelse. Et værelse er 100x100 pixels. Der er i alt plads til tre rum i bredden og tre rum i højden.

1. Undersøg programmet house. Hvor deklarerer jeg en variabel af datatypen Room, og hvad hedder min variabel?
2. Hvor mange funktioner er der i programmet house?
3. Når jeg oprettet objektet, altså initierer variablen med en værdi, hvor mang parameter skal der så bruges?
4. Undersøg klassen Room, attributter.
   (a) Hvor mange er der og hvad bruges de til?
   (b) Hvor mange er konstanter og hvor mange er variabler?
   (c) Hvad er sammenhængen mellem antallet af parameter og klassens tilstand?
5. Find i programmet house, den eller de linijer kode som tegner taget på huset. Dette ansvar bør ligge i klassen. Opret en funktion drawRoof() i klassen Room og flyt de linijer kode over i klassen.
6. For at kunne håndtere flere rum, kan vi oprette et array. Ændre variablen house, til et array af datatype Room. Jeg kunne godt tænke mig to etager, så derfor skal længden på house være 6. Room[] house = new Room[6];
7. Tilføj nu følgende rum: Kitchen, Livingroom, Toilet, Bedroom og Bathroom. Husk at korrigere X og Y positionen.
8. For at få programmet til at udskrive alle rum, skal du loop'e igennem dit array og kalde "house[i].drawRoom();"brug "for(int i = 0;i<house.length;i++)"som løkke struktur.

9. Find ud af, hvad du skal ændre i funktionen mouseClicked() for at kunne tænde og slukke lyset i alle rum.

10. Hvor vil du tilføje funktionen checkHouse(); (se Figure 15.), som kan finde ud af om du kan gå fra huset med ro i sindet og vide at alt lys er slukket?

11. I mousedClicked() er det funktionen selv, som udskriver om lyset er tændt eller slukket. Dette ansvar vil jeg gerne have skrevet over i klassen.

```
void checkHouse(){
    boolean found=false;
    for (int i=0; i< house.length; i++) {
        if (house[i].isLightOn() == true) {
            println ("WOW turn off the light in the "+house[i].getRoomName());
            found = true;
        }
    }
    if (!found) {
        println("All right! You'r ready to go!");
    } else {
        println("you forgot something");
    }
}
```

Figur 15: funktion som undersøger om alt lys er slukket