Contents

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 spectrum strækker sig fra blå over grøn, gul til rød. Står vi og iagtager 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 plad på vores A4 side. Vi løber tør for plads fordi tal fylder! Etterne flyder 1 cifer, tierne fylder 2 cifre, hundrederne fylder 3 cifre osv. 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 talne 0 og 1. Lad os for et øjeblik vende tilbage til titalssystemet. 0 er ingen ting, men placerer vi det bagved et andet cifer tidobler vi ciferets 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 tredije plads er alle hundredene osv. Så titals systemet består af 10 frskellige cifre 0-9, og det er ciffrets position som bestemmer dets værdi.
Dette kan vi udtrykke matematisk. 10 er grundtallet i talsystemet, exponenten angiver tallets placering/værdi (0=ettere, 1=tierere, 2= hundredere) og 1 er cifferet vi ønsker at kende værdien for. Ciferets 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 ciferets plasering 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, exponenten 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	•	•	•	•

Figure 1: En beregning af værdien 1022 i titalssystemet

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

Figure 2: En beregning af værdien 10 i totalssystemet

ikke har mere end to cifre, er der ingen grund til at gange med cifferetsværdi. Ciferets 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

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

Figure 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

Figure 4: En byte med værdien 42

den decimale værdi: $1 + 4 = 5$

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 hyrdede 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 tredije vindue skulle han komme kl: 4 og var der lys i det fjere 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?

1.2 Den maksimale værdi for en byte

En byte består normalt af 8 bit. Der findes også bytes på 6 eller 12 bits ligesom at de i amerika ikke bruger A4 papir, men letter format. 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 sige, at hver værdi kan repræsentere en af regnbuen farver. Det gør det muligt at have 256 forskellige farver.

1.3 Paritets bit

Hvis den første bit 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.

1.4 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=-128; for (int i = 0; i¡260; i++) println(b); b++;

1.5 Når du køber øl hos købmanden

2 Processing editor

3 Sektiventiel programmering

4 Hvordan man spiser en elefant

5 Funktioner i java

6 Lektion 1

Denne opgave handler om sekventiel programmering. Det betyder at instruktionernes rækkefølge ikke er ligegyldig.

6.1 Opgave

Du skal lave en kopi af min tegnig: opg1-hoejhat.pdf. Det primære fokus i denne opgave er, at bruge dokumentationen i processing.

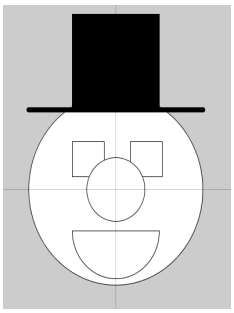

Figure 5: hoejhat

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

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

Brug processing dokumentationen: processing.org/reference, for at finde ud af hvilke parameter de forskellige funtioner skal have.

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

Brug rutediagrammet i figur 6 for at skrive koden:

6.2 Opgave

1. Hvad sker der, hvis du bytter om på rækkefølgen? Altså hvis du starter med øjne,næse og mund og så tegner ansigtet.
2. Undersøg i processings dokumentation, om du kan finde funktioner som kan begregne:
   - En funktion som kan beregne potensen af en given værdi, x^y
   - En funtion som kan beregne kvadratroden af en given værdi, $\sqrt{x}$
3. Dette er en opgave i at bruge koordinatsystemet med en variabel. Lav et program som kan beregne længden af C i en retvinklet trekant, og som tegner trekanten på skærmen og udskriver længden til consollen.

Du kan bruge dette program med kommentarer som jeg har lavet. Læst først alle linjer og prøv at forstå hvad de gør. Selve opgaven er de nederste tre linjer.

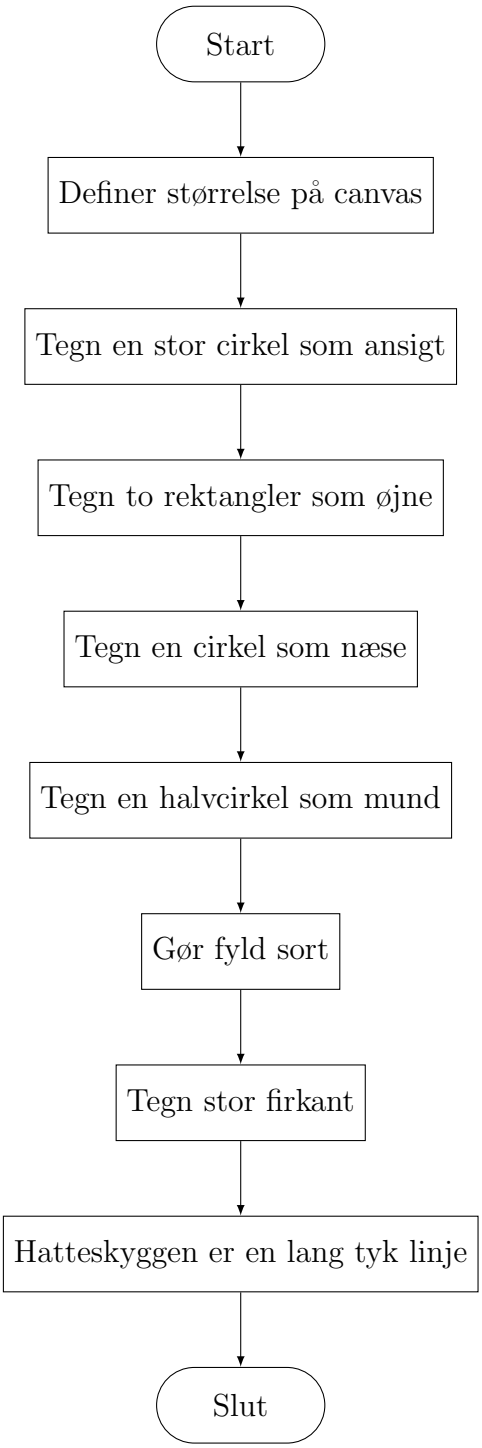

Figure 6: Rutediagram

```
//Variable deklaration. Vi navngiver og bestemmer typen for en variabel.
// float betyder at variablen kan indeholde et kommatal. Det skal vi fordi pow() forlanger at
float a;
float b;
float c;
// Initiering af variablen. Vi tilføjer en værdi til variablen
a = 120;
b = 180;
c = 0;
// angiver størrelsen af canvas
size(800,600);
// pytagoras beregning af C med funktionen pow()
c=sqrt(pow(a,2) + pow(b,2));
//udskriv den beregnede længde for C til consol
println(c);
// tegn linijerne på skærmen - du skal udfylde alle x'er med den rigtige værdi. Hint: man kø
simple matematiske operationer som f.eks. + - * eller /. f.eks. 20+a
//Hvis du starter i koordinaten 20,20 kan du ved hjælp at længden af a og b finde de sidste
line (20,20,x,x);
line (20,x,x,x);
line (x,x,x,x);
```

Figure 7: Program som beregner en side af en trekant

7 Lektion 2

Du kender nu til følgende datatyper og instruktioner:

Datatyper:

- float: kommatal.
- char: en enkelt karakter.
- String: et array af char.

Instruktioner:

- size(); // sætter størrelsen for canvas
- line(); // tegner en linje

- strokeWeight(); // tykkelse af streg
- rect(); // tegner en rektangel
- circle(); // Tegner en cirkel
- arc(); // Tegner en bue
- fill(); // udfylder en figur med farve. [1]

I denne opgave skal du bruge datatypen integer. Integers er heltal som f.eks: 0,1,2,3,4,5... i java deklarerer vi variablen x af datatypen Integer som:

```
int x;
```

Eksempel på en deklaration og initiering i samme linje:

```
int x = 0;
```

Vi skal også 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: Ligesom variabler skal deklareres så skal funtioner deklarere deres 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. Void betyder ugyldig eller udløbet.
- 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.

[1] Find rgb farver her: www.rapidtables.com/web/color/RGB_Color.html

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

HUSK: Hvis du vil bruge en funktion i processing skal du også bruge void setup(); og void d

7.1 Opgave

Se på programmet fourTimesAAlien (hentes på GITHUB).

1. Find funktionen frameFunction. På hvilken linje (nummer), deklarerers frameFunction() ?
2. Fra hvilke linjenumre, bliver funktionen kaldt?
3. Med hvor mange parameter kalder jeg funktionen?
4. Opret nu dine egne funktioner som laver:
   (a) Hoved med øjne
   (b) Kroppen med ben
   (c) Skirver et nyt navn
   (d) Kald dine tre nye funktioner fra draw() og se om programmet stadigvæk tegner en alien i det øverste felt til venstre.
   (e) Opret en ny funktion som kalder hoved og krop og ret draw() til. Din funktion skal tegne en alien i alle firkanter. Du skal tilpasse parameterne og du finder nok ud af at det har noget med x,y pos at gøre :)

7.2 Opgave

Du skal lave et kasseapperat. Apperatet skal have nogle forskellige funktioner som returnerer forskellige værdier. Når en funktion returnerer en værdi skal vi have en beholder som kan indeholde den værdi som kommer retur. F.eks. en moms funktion returnerer en float.

//deklaration af variablen float belobMedMoms=0; // kald funktionen og hold beholderen beloebMedMoms klar til at modtage retur værdien fra funktionen. belobMedMoms = beregMoms(belobUdenMoms);

Hent programmet kasseapparat fra Github

(a) Se på funktion som beregner moms. Den modtager en integer værdig som parameter og returnerer en float(kommatal). Du beregner momsen ved at gange med 1,25. Forklar alle linjer i programmet.

(b) Lav en ny funktion som trækker moms fra. Den modtager en float værdi som parameter og returnerer en float. Man trækker momsen fra ved at gange med 0,8.

(c) Lav en ny funktion som kan udskrive en bon.

Tak for dit køb Jens, Du har købt for 125 kroner.

Beløb uden moms: 100 Beløb med moms: 125 Momsbeløbet udgør: 25

(d) Slå day() op i dokumentationen og tilføj følgende linje til din bon Dato: 16/9 2020 kl 13:00

ingen mellemrum imellem talne :)

8 Lektion 3

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

I har lavet en liste med ord og udtryk:

- 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

Table 1: Liste over datatyper.

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

- Cammelback notation - en måde at skrive variabel navne på.

Det er nemt at se forskel på primitive og ikke primitive[2] variabler i Java. Primitive datatyper staves med lille begyndelses bogstav, ikke primitive datatyper staves med stort begyndelses bogstav. Primitive datatyper kan repræsenteres i en byte og kan sammenlignes med en operator '=='. Ikke primitive datatyer 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().

Denne opgave handler om at forstå de forskellige datatyper.

8.1 Opgave

1. Undersøg for hver primitiv datatype, hvor meget plads (bytes) der allokeres i hukommelsen når man deklarerer en variabel af datatypen. Det kan i finde her: https://data-flair.training/blogs/java-data-types/. En integer fylder 4 bytes og at den maksimale værdi er: $2^{31} = 2.147.483.648$. Noter alle dine resultater

2. Skriv et program som ved hjælp af funktioner, beviser hvilke minimums- og maksimumsværdier for de primitive datatype kan indeholde.Find evt. inspiration i programmet testDatatyper, som du finder på github. Noter alle dine resultater

3. De to datatyper float og double er ikke lige nøjagtige. Det kan de se hved følgende opgave: Hvad giver kvadratroden af 2 gange med

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

kvadratroden af 2? Lav et først et program med sqrt() som returnerer en float og herefter med Math.squrt() som returnerer en double. Forklar forskellen på de to funktioner og redegør for resultatet af de to instruktioner.

9 Lektion 4

Vi har nu talt om de forskellige datatyper, vi har talt om floating point og vi ved hvad overflow er.

I denne opgave skal du arbejde med løkker. For og While løkker. 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.

Hvis vi har en variable, som vi vil bruge som en tæller, af type integer, har vi talt om to måder at bruge variablen på.

metode 1:

```
int i;
i= i +1;
```

metode 2:

```
int i;
i++;
```

Begge metoder lægger værdien 1 til variablen i. Men hvorfor er det vigtigt at vide, hvordan man lægger 1 til en værdi? Det er det fordi vi bruger en tæller når vi løber igennem en løkke. Den tæller hvor mange gange vi har udført instrutionen. Det kan være smart at bruge i forbindelse med streng operationer.

Løkker Der findes to former for løkker. While og For.

while (condition) // code block to be executed

for (statement 1; statement 2; statement 3) // code block to be executed

While-løkke Oversat, betyder while, sålænge - så længe betingelsen ikke er sand, så skal løkken udføres. Se på dette eksempel:

boolean found = false; // en variable til at teste på int i=0; println(”så løber vi igennem while-løkken”); while (!found) println(i);

if (i==9) // test om betingelsen er opfyldt found=true; //vi har fundet den rigtige værdi og gør nu found true. i++; // tæl i en op

Hvis vi ikke sætter sand til at være true, så vil løkken køre for evigt. Vi skal altså kunne forudse et tidspunkt, hvor vi kan sætte found til at være sand, for at det er en fordel at bruge en while struktur. Jeg ved, hvis jeg starter med 0 og lægger 1 til, så vil jeg på et tidspunt ramme 9.

Alternativet er en for-løkke For løkker bruger vi når vi ved, hvor mange gange vi skal genneløbe løkken maksimalt.

println("så løber vi igennem for-løkken"); for (int i =0; i¡10; i++) println(i);

Opgave:

Vi har talt om at string har et index. Hvis vi har en løkke kan vi gennemløbe en streg og se på det enkelte bogstav på i-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 datatype starter med stort bogstav, vil der altid være metoder i kan bruge. Man bruger metoderne sammen med variabel navnet. F.eks. hvis str er defineret som datatype String, så kan man skrive: str.charAt(i);

1) Lav henholdsvis en for-løkke og en while-løkken som udskriver den 5 karakter i sætningen "Hej med dig!". Tip, brug charAt() sammen med din tæller i.

2) 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.

3) 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 tilsidst udskrive hvor mange e'er som er fundet.

4) 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 :)" 5) Lav et program som kan udskrive længden af overstående sætning. 6) Lav et program som klipper i vores streng. Den skal tage fra position 83 og til slut. Udskriv den nye streng. 7) Lav et program som klipper i vores streng. Den skal tage fra position 83 og til 85. Udskriv den nye streng. 8) Lav hele sætningen om til store bogstaver og udskrive den. 9) Lav hele sætningen om til små bogstaver og udskrive den. 10) læs og forstå følgende program:

String str2 = "Nej"; if (str2.equals("ja")) println ("Det gør den!"); else println ("Det gør den ikke!");

a) hvad udskriver programmet? b) hvordan kan du få programmet til at udskrive det modsatte af, hvad den gør nu?

10 Lektion 5

Opgaven handler om optimering af kode og brug af funktioner.

Læs og forstå koden i mappen atArbejdeMedEnStreng.

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.

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()

11 Lektion 6

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

11.1 Opgave

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

setup: canvas størrelse og noLooop;

draw - mit hovedprogram opret integerværdi med et årstal - myBirthYear kald funktionen howOld() med parameteren myBirthYear udskriv returværdigen fra howOld().

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 draw - mit hovedprogram: Opret en konstant med din fødselsmåned - int myBirthMonth = 3 (marts) Kald funktionen howOldInMonths() med parameteren myBirthMonth udskriv resultatet af howOldInMonths()

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.

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

12 ROBO Code

Offensiv strategi - en agressiv robot som er opsøgende og konfronterende. Deffensiv strategi - en passiv robot som er forsvarende og undvigende.

Opgave 1 Opret en junior robot, som kun skal holde stille. Brug robotten SittingDuck til at skyde efter. Scan området med din radar. Drej kanonen i retning af målet. Fyr i mod målet. Opgave 2 Udvid nu opgave 1, så at din kampvogn svinger kanonen 360 grader rund og her efter kører 100 pixels frem. Opgave 3 Lav en strategi for hvad der skal ske hvis du kører ind i væggen. Opgave 4 Lav en strategi for hvad der skal ske hvis du bliver ramt af en modstander Opgave 5 Tilføj: public void onHitRobot() og lav en strategi for hvad du vil gøre hvis du rammer en modstander.

13 Github

Vi skal arbejde med versions styring. 1. Gå til: github.com og opret en konto. 2. Installer Github Desktop: https://desktop.github.com/ bemærk hvis du bruger mac eller linux skal du vælge en anden distribution.

5. Læs denne artikkel: https://www.version2.dk/artikel/saadan-kommer-du-gang-med-at-laere-git-1073040 3. Opret en ny folder på din computer. 6. Du skal lave en clone (kopi) af mit repository https://github.com/ChrisBruhn/heiku-digt.git som du gemmer i den folder du lige har lavet. Github guider dig. 4. Du skal skrive et Heiku digt. Åben den fil du har fået fra mit repository og tilføj dit digt. Hvis du ikke ved hvad det er, kan du læse om det her: https://da.wikipedia.org/wiki/Haiku 7. Opret et merge-request.

Git-begreber: Repository: Et repository er det sted, du lagrer dit projekt. Det kan være en lokal mappe eller en ekstern server eller tjeneste som Github.

Add: Tilføjer filer eller ændringer til det næste commit.

Commit: Et commit er som et save game i et computerspil. Når du har nået et punkt, hvor det er en god idé at lave et save game, du kan vende

tilbage til, så laver du et commit. Det gemmer de ændringer, du har lavet indtil videre.

Push: Når de ændringer, du har samlet i et commit, skal sendes til dit repository, skal du lave et push.

Branch: En branch er noget du bruger, når du har brug for at arbejde på en kopi af projektet og ikke vil gemme dine ændringer direkte i 'master'-kopien. Du skifter til en branch med checkout.

Pull: Når du skal hente og integrere de ændringer i projektet, som ligger i repository'et, så laver du et pull.

Merge: Når Git skal samle ændringer, bliver dit commit merget. Det kan give en konflikt, hvis to har ændret de samme steder i en fil.

Daniel Shiffman, som også har lavet proccesing, har lavet nogle gode videoer om github.

Introduction - Git and GitHub for Poets her

Branches - Git and GitHub for Poets her

Forks and Pull Requests - Git and GitHub for Poets her