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 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	•	•	•	•

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 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 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

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

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=-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

Komplekse problemer deler vi op i en række af trivielle problemer. Så løser vi de trivielle problemer i rækkefølge.

5 Funktioner i java

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 og Funktioner

Læs om while-løkker i kapitel 3.3 og om for-løkker i kapitel 3.4. Læs om funktioner i kapitel 3.5

Du kender nu til følgende datatyper og instruktioner:

Datatyper:

- float: kommatal.
- double: mere præcis end float.
- char: en enkelt karakter. Char er en unsigned byte og kan derfor ikke indeholde minustal.
- int: heltal.
- long: 2 gange så stor som en integer.
- boolean: sand eller falsk.

Instruktioner:

- size(); // sætter størrelsen for canvas
- line(); // tegner en linje
- stroke(); // farven på en streg
- strokeWeight(); // tykkelse af streg
- rect(); // tegner en rektangel
- circle(); // Tegner en cirkel
- arc(); // Tegner en bue
- fill(); // udfylder en figur med farve. [1]

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

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: 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 .

7.1 Opgave

Skriv et program hvor dit canvas er 800x800. Opret en funktion cirkel, som tegner en cirkel på dit canvas. Cirklen skal være rød. Ryd op efter dig. 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.0

7.2 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 (del 1; del 2; del 3) {
// Den sekvens af kode som løkken skal udføre
}
```

Et for-loop består af tre dele. Del 1 er en deklarering og initiering af vores index. Vi kalder index for i. Har vi brug for flere indlejret 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. For eksempel: i¡=10. 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 i–.

```
for (int i = 0; i < 10; i++){
  System.out.println(i);
}
```

Figur 5: For-løkke

7.3 Opgave

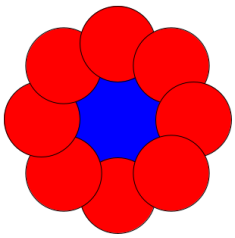
Tilpas din funktion firkant så den tegner et antal firkanter vertikalt. Hvor mange firkanter kan du få plads til?

7.4 Opgave

Omskriv din for-løkke til en while-løkke.

7.5 Opgave

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



Figur 6: Blomst

8 Betingelser/Forgreninger

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

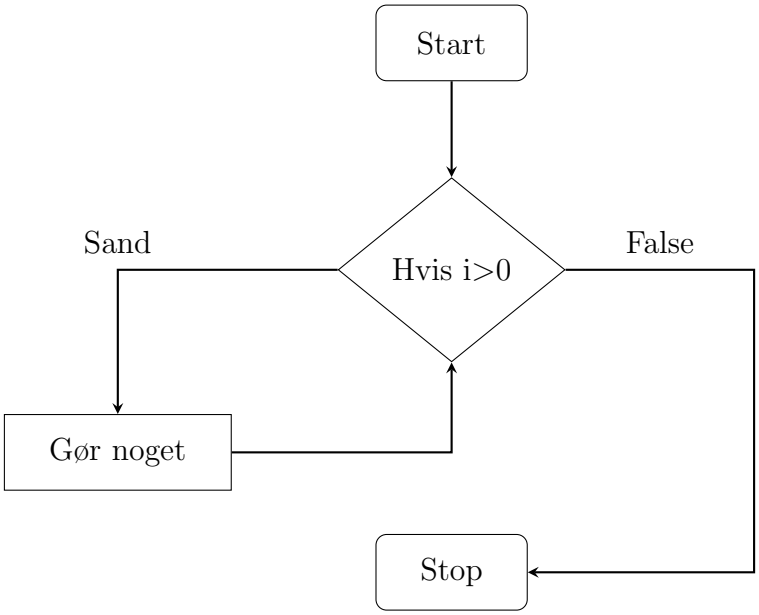
8.1 Operatorer til sammenligning

- 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: $a! = b$

8.2 Aritmetiske operatorer

- + 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
- % Modulus Returnere resten ved division x
- ++ Inkrement Øger værdien af en variabel med 1 ++x
- – Dekrement Mindsker værdien af en variabel med 1 –x

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



Figur 7: Rutediagram løkke

9 Lektion 4 uge 43

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

Computere arbejder med 0,1 og alt basserer sig på det binæretalsystem. Det betyder at alt funktionalitet baserer sig på booskalgebra, AND, OR og NOT. De nøgleord bruger vi også når vi programmerer.

I har lavet en liste med ord og udtryk:

1. Instruktion
2. Sekvens
3. Funktion
4. Kontrolstruktur
5. Betingelser
6. Forgrening
7. Løkke

8. Funktion
9. Initiering
10. Deklaration
11. Parameter
12. Cammelback notation
13. Variabel (https://data-flair.training/blogs/java-data-types/)
    (a) Ikke primitive datatyper
        i. String
        ii. Array
        iii. klasser
        iv. Interfaces
    (b) Primitive datatyper
        i. Int
        ii. Float
        iii. Char
        iv. Boolean
        v. Byte
        vi. Short
        vii. long
        viii. Double.

9.1 Opgave 1

Denne opgave handler de forskellige datatyper. Til dette skal du opstille en tese (et vildt, men kompetent gæt) for min og max værdi af hver primitiv data type. Skriv et program, som kan beregne den maksimale værdi for en datatype. Vi kalder dette den induktive metode (specialtilfælde), fordi vi leder efter en special værdi (sort svane). Find evt. inspiration i programmet testDatatyper, som du finder på github. Noter alle dine resultater. Brug nu den deduktive metode (logiske), og beregn den maksimale værdi for hver primitiv datatype ud fra hvor meget plads der allokeres i compueterens hukommelse til datatypen. F.eks allokeres der (sjovt nok) en byte til datatypen byte. Du kan her finde svaret https://data-flair.training/blogs/java-data-types/ Noter alle dine resultater og slut af med at sammenholde din tese med

resultatet af din induktive og deduktive metode og hvad der står i artiklen: https://data-flair.training/blogs/java-data-types/ Ekstra opgave: De to datatyper float og double er ikke lige nøjagtige. Det kan de se hved følgende opgave: Hvad giver kvadratroden af 2 gange med 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.

10 ROBO Code

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

10.1 Opgave 1

Opret en ny junior robot. Du skal overveje 4 ting:

1. Hvordan skal din robot bevæge sig på banen?
2. Hvad skal der ske når din robot opdager en fjende?
3. Hvad skal der ske når din robot bliver ramt?
4. Hvad skal der ske hvis din robot rammer væggen?

10.1.1 1 - Hvordan skal din robot bevæge sig på banen?

Din robot er programmeret til at gå 100 frem, dreje kanonen 360 grader og gå 100 tilbage og dreje kanonen 360 grader.

10.1.2 2 - Hvad skal der ske når din robot opdager en fjende?

step by step

som kun skal holde stille. Du kan kalde den target. robotten skal finde midten af skærmen og så holde stille.

10.2 Opgave 2

turnTo(360); back(fieldHeight); ahead(fieldHeight/2); turnTo(90); ahead(fieldWidth); back(fieldWidth/2);

Brug robotten SittingDuck til at skyde efter. Scan området med din radar. Drej kanonen i retning af målet. Fyr i mod målet.

10.3 Opgave 2

Udvid nu opgave 1, så at din kampvogn svinger kanonen 360 grader rund og her efter kører 100 pixels frem.

10.4 Opgave 3

Lav en strategi for hvad der skal ske hvis du kører ind i væggen.

10.5 Opgave 4

Lav en strategi for hvad der skal ske hvis du bliver ramt af en modstander

10.6 Opgave 5

Tilføj: public void onHitRobot() og lav en strategi for hvad du vil gøre hvis du rammer en modstander.

11 OOP

OOP betyder objekt orienteret programmering.

12 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 8.), 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 8: funktion som undersøger om alt lys er slukket