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 tusindvis af farver. For at computeren kan forstå det, bliver vi nødt til at give hver enkelt farve et unikt nummer. Hurtigt vil vi kunne se, at vores liste over farver 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! Ettallerne fylder 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, og registrerer vi én farve per linje 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 cifrene 0 og 1.
Lad os for et øjeblik vende tilbage til titalsystemet. 0 er ingenting, 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 ettallerne, den anden plads er alle tierne, den tredje plads er alle hundrederne osv.Så talsystemet består af 10 forskellige cifre, 0-9, og det er cifrets position, der 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: $10^n * tallet$.

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

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

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'ere, 2'ere, 4'ere, 8'ere, 16'ere, 32'ere, 64'ere, 128'ere, 256'ere, 512'ere, 1024'ere også videre. Det er altså cifrets placering, som er afgørende for tallets værdi. F.eks. er 1010 binært lig med den decimale værdi 10. 1'ere er der ikke nogen af, 2'ere er der en af, 4'ere 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 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 analogien om, at 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 nedenunder viser, om bittet er sat. Vi lægger alle værdier sammen i celler, hvor bittet er sat, for at finde den decimale værdi. I mit eksempel er det tilfældet for cellen, som repræsenterer værdien 1, og cellen med værdien 4. Derfor er den decimale værdi: $1 + 4 = 5$

1.2 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 forskellige værdier er 256! Fordi 0 tæller også med som værdi. Det betyder, at vi i en byte på 8

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

Figure 4: En byte med værdien 42

bit kan vælge, 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 (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.

1.4 The most significant bit

Hvis en byte er signed, 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 negativt tal. Dermed er den maksimale værdi 127 og den mindste -128, og antallet af forskellige værdier er 256. Du kan afprøve det med følgende lille Java-program.

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

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

1.5 Opgave

Der var en gang for længe siden. 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 stærk 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 hun 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. 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?

14 Objektorienteret programmering

OOP betyder objektorienteret programmering og er et programmeringsparadigme. Det tager udgangspunkt i et fysisk objekt, som kan være en bil, person, hus, kuglepen osv. Men det kan også være immaterielt, som en helligdag eller rettigheder. For at kunne oprette en instans af et objekt, skal vi bruge en beskrivelse af objektets egenskaber og funktionalitet. En personbils egenskaber er f.eks. farve, mærke, motorstørrelse osv. En personbils funktionalitet er, at den kan køre fremad, bakke, dreje osv. Denne beskrivelse kalder vi en klasse. Man kan bruge UML-diagrammer til at beskrive klasser. 25

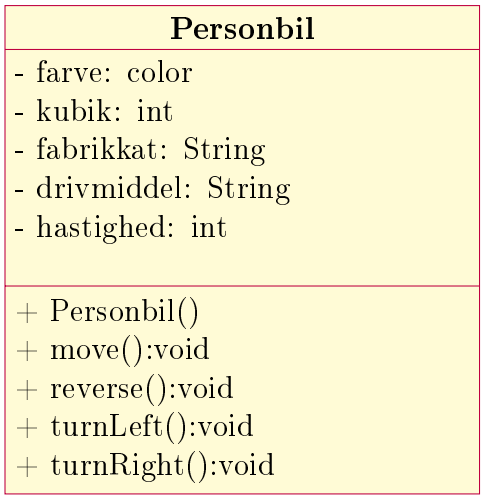

Figure 25: Personbil

Når vi programmerer, forsøger vi at skabe en matematisk model af virkeligheden. Objekter giver os mulighed for at lave vores egne datatyper.

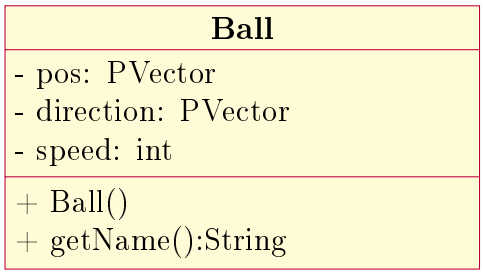

Figure 26: Class diagram

14.1 Nedarvning

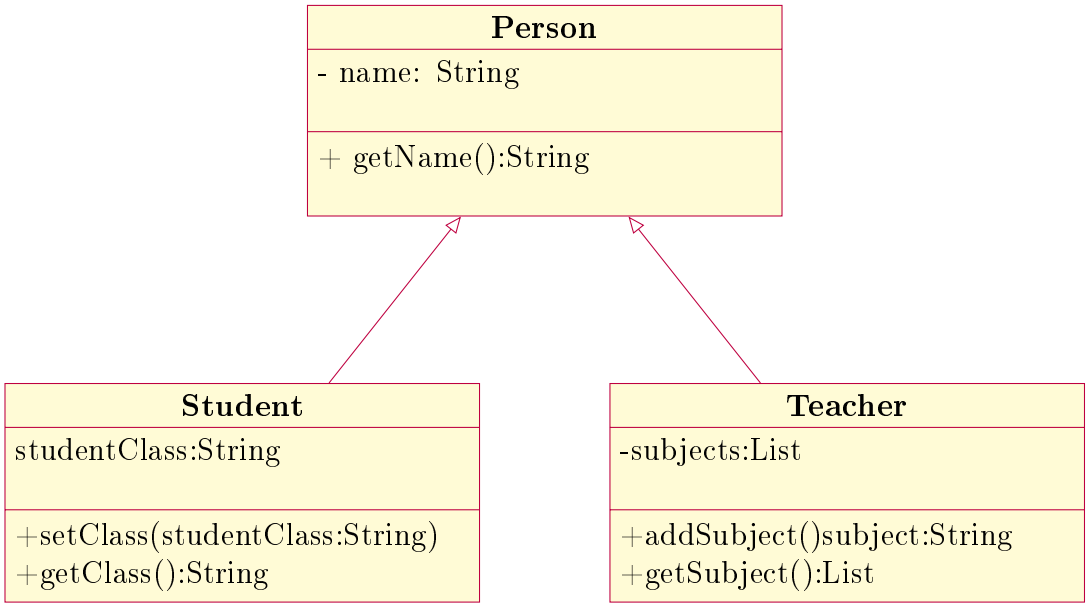

Figure 27: Class diagram nedarvning

14.2 Opgave

Her er et eksempel på en klasse. Firkant.pde. Klasser er ikke-primitive datatyper og skrives altid med stort.

Klassen Firkant, kan generere en tilfældig x og y koordinat. Klassen kan også tegne en firkant på dit canvas.

En klasse er en skabelon.

```
class navn-paa-klassen{

        klassens tilstand / attributter
        (variabler-kan initieres gennem konstroktor eller setters)

        klassen konstruktor
        (hvad skal der ske i de sekundt at vi initierer klassen)

        klassens metoder
        (getters/setters og alle andre metoder)
}
```

Figure 28: Skabelon til en klasse

Hvis du kigger i mappen klasse, så er der to filer. klasse.pde, er den fil som bruger klassen, og klassen er Firkant.pde.

Vi deklarerer en variable af datatypen integer på følgende måde: int x; -på samme måde gør vi med klasser: Firkant f; på denne måde laver vi en spand

som kan indeholde firkanter. Men for at kunne bruge spanden skal vi have noget i den. Vi initierer på følgende måde: f = new firkant();

Nu kan vi tilgå både metode og attributter ved at skrive f.generate(); eller f.x = 0; I vores tilfælde giver det ingen mening at ændre på klassens attributter, fordi klassen selv genererer nogle tilfældige værdier. Reglen er, at klassen skal være autonom. Dvs. at der ikke må være andre som ændrer på klassens tilstand end klassen selv.

14.2.1 Opgave

Ændre klassen Firkant således, at firkantens tilfældig x og y position generes i konstruktøren og ikke skal kaldes fra hovedprogrammet.

14.2.2 Opgave

Tilføj en attribut farve, til klassen. Farven skal være tilfældig. Brug RGB farver. Hvis du ikke kan huske hvad komandoen hedder, kan slå det op i dokumentationen.

14.2.3 Opgave

Tilføj en metode som kan tegne en cirkel i stedet for en firkant.

14.2.4 Opgave

ændre hovedprogrammet klasse.pde så den skiftevis tegner en firkant og en cirkel.

14.2.5 Opgave

Det er jo lidt dumt at have en klasse som hedder firkant, hvis den tegner en cirkel. Så lave en ny klasse så du deler firkant og cirkel hver for sig. Tilpas klasse.pde til dine nye klasser.

14.2.6 Opgave

Udvid med en trekant og en rektangel.

14.2.7 Opgave

lav så figuerne bliver af variabel størrelse.

14.2.8 Opgave

opload dit program til din github.

14.3 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 14.3.), 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 ("turn off light in"+
                 house[i].getRoomName());
                 found = true;
          }
        }
         if (!found){
                 println("All right! You're ready to go!");
        } else {
                 println("you forgot something");
        }
}
```

Figure 29: funktion som undersøger om alt lys er slukket