# plainSekFlowcharts

I lektion 5 og 6 er der brugt nogle små diagrammer til grafisk at illustrere kontrolstrukturerne. Sådanne diagrammer hedder ”*flowcharts*” på engelsk og flowdiagrammer på dansk.

Meningen er at give et overblik over opbygningen af program­met eller en del af programmet.

Som med alt andet nyt skal man lige vænne sig til udseende, men ellers kan sådanne flowdiagrammer læses og forstås af ikke-programmeringskyndige.

Det kan være til stor hjælp, hvis du vil have andre mennesker til at sætte sig ind i dit program, fx for at hjælpe, inspirere, rette eller på anden måde give et bidrag. Da diagrammerne stort set er uafhængige af programmeringssproget udvider det læserskaren voldsomt.

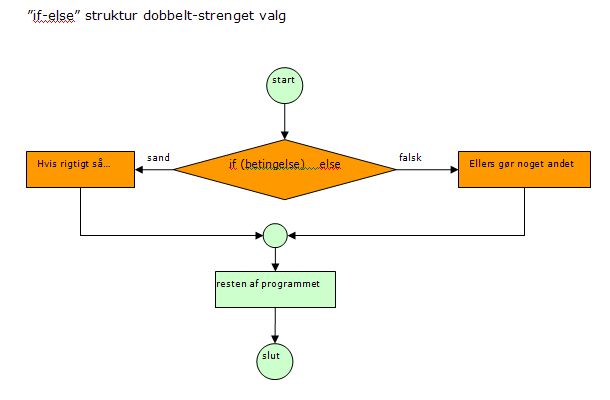
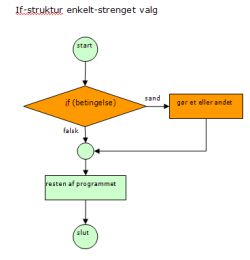
**Simpelt program:**

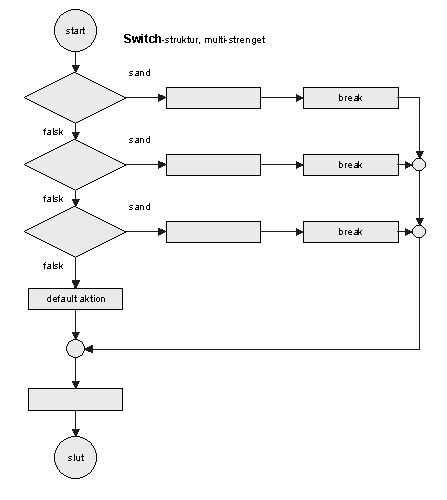
Et simpelt program kan fx være sekventielt. Her afvikles programmet i den rækkefølge koden gennemlæser programmet: linie 1, linie 2, linie 3, linie 4 osv. Det kan illustreres som vist her til højre. Bemærk at teksten ikke behøver at være programmering men alm. forståelige stikord.

**Program med valg:**

Det er næsten umuligt at skrive et program uden at der skal træffes valg.

Her er vist en oversigt over de to mest brugte valg-strukturer fra Java:



Endelig findes også valgstrukturen   
”switch” som kan give valgmulighed mellem et vilkårligt antal af valg­mu­lig­heder.  
 Switch-sætningen er ikke gennemgået/vist i tidligere lektioner.

Syntaksen er vist herunder. Som valgmuligheder kan indsættes heltal og ”char” (karakterer – enkelttegn)

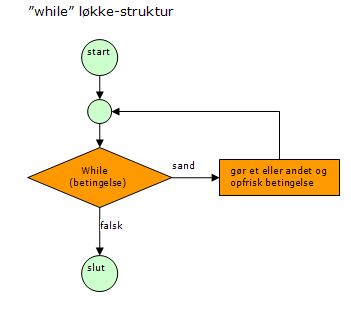
F.x. 1,2,3,…. og a,b,c,…z,A,B,C…,Z.

int mulighed

switch (mulighed)  
{  
 case 1:  
 System.out.println ("Et 1-tal");  
 break;  
 case 2:  
 System.out.println ("Et 2-tal");  
 break;  
 case 3:  
 System.out.println ("Magiske tre");  
 break;  
 case 4:  
 System.out.println ("Fortsaette?");  
 break;  
 default:  
 System.out.println("Defaulttekst");  
 }

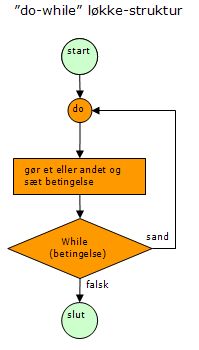
**Program med gentagelser:**

Programmer med loops kan også illustreres med flowdiagrammer. Her er tre gentagelsesstrukturer fra Java: ”while”, ”do-while” og ”for”

”While”: Dette loop kører så længe

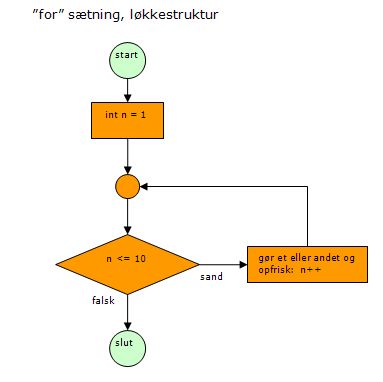
betingelsen i *while (betingelse)* er sand.

Programmøren skal være omhyggelig med at sikre, at man kan komme ud af loopet ved at sørge for at betingelsen ændres inde i loopet. Alternativt må programmøren lave en ”nødudgang”: fx. at man maximalt må forsøge sig 10 gange.

”do-while”: Dette loop sikrer at noget udføres mindst én gang. Loopet køres kun så længe

betingelsen i *while (betingelse)* er sand.

Igen skal programmøren være omhyggelig med at opfriske betingelsen, så vi ikke får en uendelig løkke.



For-sætningen kører et forudbestemt antal gange.

Start, slut og trin-værdi er samlet i én og samme linie

**for (n=1;n<=10; n++)**

Dette er en stor fordel, da genopfriskningen ikke glemmes ved ”copy-paste” genbrug.

**Program med metoder:**

I nogle programmet sker afviklingen ikke sekventielt.

Når vi skriver programmer med metoder kan afviklingen af programmet springe rundt mellem de enkelte metoder, når disse kaldes.

Metoder er et program i programmet. Hver gang metoden bliver kaldt, springer programmet frem til metoden og afvikler denne.

I andre programmeringssprog f.eks. C++ kaldes metoder for funk­tio­ner.

I sådanne tilfælde kan det være en fordel at opdele programmet i flere flowdiagrammer, så hvert enkelt diagram forbliver overskueligt.

Lav gerne et diagram for main-metoden og et diagram for hver metode.

Lad os finde et gammelt program med metoder frem fra lektion 7 og vise det som flowdiagram.

public class PasswordCheck3

{

public static void main(String[] args) //Main-metoden afvikler programmet

{

double tal=8.45;

boolean personOK= false;

personOK = godkendBruger();

if (personOK)

udfoerSpaendendeProgram(tal);

else

System.out.println("\nNul Putte! Du maa ikke vaere med.");

}

//Metode der godkender password og returnerer sand eller falsk

public static boolean godkendBruger()

{

String password = "fake";

int count=1;

System.out.print("\n\nIndtast password: ");

password = Keyboard.readString();

/\*To betingelser som begge skal være opfyldt:

forkert password og under 10 forsøg \*/

while (!password.equals("fake") && (count<10))

{

System.out.println("\nDu har indtastet forkert password. ");

System.out.print("Proev igen. Indtast password: ");

password = Keyboard.readString();

count=count+1;

}

if (password.equals("fake")) return true;

else return false;

}

//Metode der ikke har nogen returværdi - void

public static void udfoerSpaendendeProgram(double tal)

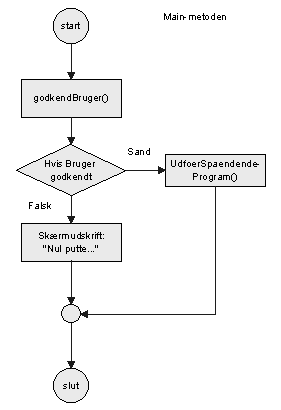
{

System.out.println(”\n\nTillykke du er godkendt”);

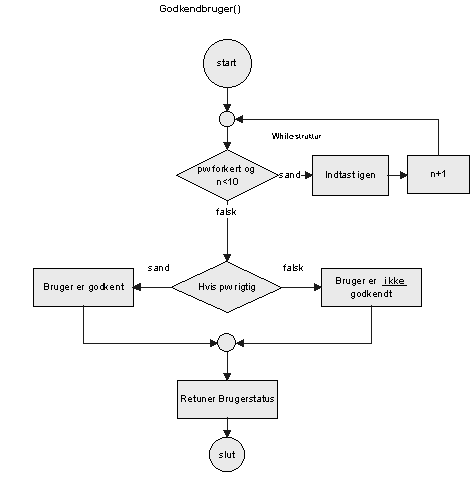
System.out.println(” Du kan nu veksle euro til kurs: ” + tal);

}

} // slut på klassen



Her er main-metoden.   
  
Bemærk at under-metoder  
blot er kaldt uden at vise  
hvad der sker i detaljer.



Her kommer metoden   
*GodkendBruger()*.  
  
Metoden   
*UdfoerSpaendendeProgram()*   
er ikke vist som flowdiagram, da  
der i dette program/metode kun er  
en simpel skærmudskrift.

På denne måde kan komplicerede diagrammer opbygges eller splittes ad. Husk at hovedformålet med et flowdiagram er at vise flowet gennem programmet i en overskuelig form.

Derfor ”Keep it simple”. \**KISS =keep it simple stupid*

Teksten inde i de enkelte komponenter kan være mere eller mindre programspecifik.  
I eksemplet herover er valgt ikke-program sprog.

I lektion 6 og 7 er der valgt Java-syntaks, da formålet netop var at vise denne.

Afhængig af hvem man kommunikerer med, vælger man sprog.   
Flowdiagrammer er netop beregnet til at kommunikerer med. Ved at vælge ikke-programmeringssprog kan ikke-programmører også sættes ind i programmets virkemåde og give sin kritik tilkende. (Kritik kan jo også være positiv og konstruktiv).

**Opgave 12:** Lav et eller flere diagrammer til at illustrere et par af opgaverne fra lektion 6-11.