Utførelse av programmer, metoder og synlighet av variabler i JSP

Av Alf Inge Wang

1. Utførelse av programmer

Et dataprogram består oftest av en rekke programlinjer som gir instruksjoner til datamaskinen om hva som skal utføres. Slike instruksjoner brukes typisk til å:

- Opprette variabler
- Tilordne verdier til variabler
- Teste på verdi av variabler
- Utføre ulike beregninger
- Behandle ulike typer data
- Presentere data for bruker (lyd og bilde)
- Registrere og håndtere input fra bruker (mus, tastatur, joystick).

Når en datamaskin skal tolke et program er den normale framgangsmåten å begynne med første linje i programmet og gå igjennom programmet linje for linje. For små programmer kan hele programmet være samlet i en fil. For større programmer, fordeles programkoden i flere filer (men dette kan dere lære mer om i andre programmeringsfag). Program 1 under viser et enkelt JSP-program som skriver ut det største tallet av to tall.

Program 1. Programflyt til et enkelt JSP-program

```
1     <%
2     int x=1;
3     int y=5;
4     int svar;
5     if (x>y) {
6           svar=x;
7     } else {
8           svar=y;
9     }
10    out.println("Største tall: "+svar+"<br/>");
11    x=7;
12    y=3;
13    if (x>y) {
14          svar=x;
15    } else {
16          svar=y;
17    }
18    out.println("Største tall: "+svar+"<br/>");
19    %>
```

Programflyt

Linje 2
Linje 3
Linje 4
Linje 5
...
Linje 14
Linje 15
Linje 16
Linje 17
Linje 18

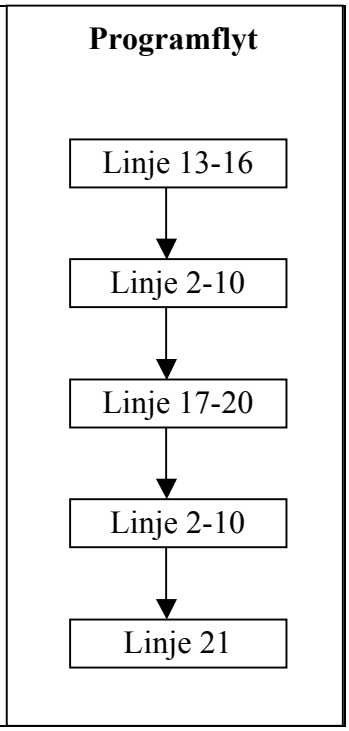
I kodeeksempelet over vil datamaskinen starte på linje 1 og gå igjennom alle programlinjene linje for linje til den kommer til programlinje 19). I programlinje 1 vil datamaskinen tolke "`<%`" og forstå at nå kommer det JSP-kode. I programlinjene 2-4 vil datamaskinen opprette tre variabler av typen heltall (int) og tilordne verdi til disse variablene. I programlinjene 5-9 vil datamaskinen sjekke hvem av variablene `x` og `y` som er størst og sette variabelen `svar` lik

den største av dem. Programlinje 10 vil skrive ut svaret til nettleseren. Programlinjene 11 og 12 tilordner nye verdier til `x` og `y` og deretter utføres den sammen sjekken om hvilken variabel som er størst og utskrift av denne. I programlinje 19, vil datamaskinene tolke "`%>`" til at nå er det slutt på JSP-koden. I dette eksempelet vil altså datamaskinen gå igjennom hele programmet linje for linje og endre på variabler avhengig av hvor langt man har kommet i kjøringen av programmet.

Program 2 (se under) gjør akkurat det samme som program 1, men her bruker vi en egen metode til å utføre testen av hvem av variablene som er størst ved hjelp av metoden `storst`.

Program 2. Programflyt til et JSP-program med metode

```
1     <%!
2     int storst(int x,int y) {
3     int svar;
4     if (x>y) {
5           svar=x;
6     } else {
7           svar=y;
8     }
9           return svar;
10    }
11    %>
12    <%
13    int x=1;
14    int y=5;
15    int svar;
16    svar=storst(x,y);
17    out.println("Største tall: "+svar+"<br/>");
18    x=7;
19    y=3;
20    svar=storst(x,y);
21    out.println("Største tall: "+svar+"<br/>");
22    %>
```



Kjøringa av program 2 foregår på en annen måte enn program 1. Her vil ikke datamaskinen utføre programlinjene 1-11 først. Når datamaskinene går igjennom program 2 vil den i programlinje 1 tolke "`<%!`" og forstå at her kommer en definisjon av en egen metode og ikke vanlig kode som skal utføres. Datamaskinen vil derfor gå igjennom definisjonen av metoden og sjekket at den er riktig definert, for deretter å huske metoden slik at den senere kan kalles av hovedprogrammet. Først i programlinje 12 vil den finne den normale JSP-merkelappen og starte utførelse av kodelinjene. I programlinjene 13-15 vil man opprette og tilordne de tre variablene `x,y` og `svar`. I programlinje 16 vil datamaskinen kalle metoden `storst` som er definert i programlinjene 2-10 . Datamaskinen vil derfor da gå igjennom linjene 2 til 10 med de verdiene man har satt for parametrene `x` og `y`, henholdsvis 1 og 5. Når datamaskinen kommer til linje 9 og `return` returnerer programmet til linje 16 med verdien til `svar.` Deretter vil programlinjene 17-19 bli utført. I programlinje 20, vil metoden `storst` på nytt bli kalt med verdiene 7 og 3 for `x` og `y`. Datamaskinen vil da igjen utføre programlinjene 2-10 og returnere svaret tilbake til variabelen `svar` i linje 20. Deretter vil programlinjene 21 og 22 bli utført.

Som vi ser av eksempelet i program 2, så vil ikke datamaskinen gå igjennom programmet direkte fra programlinje 1 til 22 når man har egendefinerte metoder. De programlinjene som beskriver metoden vil ikke bli kjørt før metoden blir kalt i programmet. Vi så også fra dette eksempelet at en metode kan kalles flere ganger, som fører til at de samme programlinjene blir utført flere ganger.

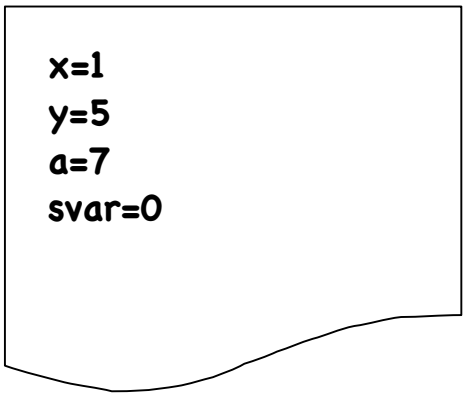
2. Synlighet av variabler

I enkle JSP-skript uten egendefinerte metoder kan du ikke definere flere variabel som har samme variabelnavn. Hvis vi ser på koden i Program 1, så deklareres variablene `x,y` og `svar` til å være av typen heltall i starten av koden. Det er da ikke mulig å deklarere nye variabler med navnene `x`, `y` eller `svar` uten at dette gjøres i en egen metode. I program 3 (se under), viser vi et JSP-skript som definerer de samme variablene flere ganger i samme fila.

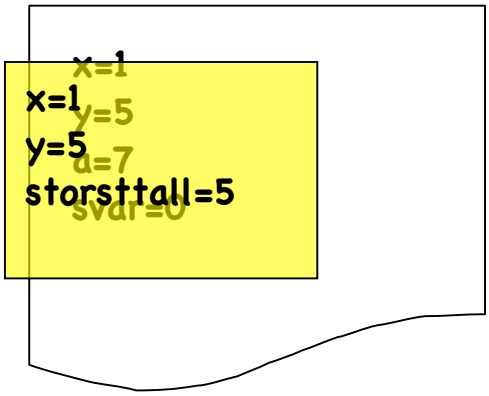
Program 3. Et JSP-program som viser synlighet av variabler

```
1    <%!
2    int storst(int x,int y) {
3         int storsttall;
4         if (x>y) {
5              storsttall=x;
6
7         } else {
8              storsttall=y;
9
10        }
11        return storsttall;
12   }
13   %>
14   <%
15   int x=1;
16   int y=5;
17   int a=7;
18   int svar=0;
19   svar=storst(x,y);
20   out.println("Største tall: "+svar+"<br/>");
21   %>
```

I program 3 deklareres `x` og `y` både i metoden `storst` og i selve programmet. Selv om variablene `x` og `y` har samme navn på begge plasser er de totalt uavhengige. For å forklare bruk av variabler på en enkel måte kan vi benytte metaforen huskelapp. En datamaskin trenger en huskelapp for å utføre sine programmer. På denne huskelappen noteres alt som trengs å holde orden på for å kunne kjøre programmet. En slik huskelapp vi inneholde alle variablene som blir brukt i et program. Hvis vi tar for oss den nederste koden i scriptlet-merkelappen (”<%”) i program 3, så vil denne huskelappen inneholde fire variabler: `x,y,a` og `svar`. Etter at programlinjene 15-18 har kjørt vil denne huskelappen se ut som vist på neste side.



I programlinje 19 kalles metode `storst` med parametrene `x` og `y`. Når en metode blir kalt i JSP vil ikke variablene bli overført direkte, men det er verdien eller innholdet til variablene som blir overført til metoden. I vårt tilfelle vil kallet av metoden tilsvare: `storst(1,5)`. Hva skjer med programmets huskelapp med alle variablene når vi kaller en metode? Saken er at metoder bruker sine egne "post-it-lapper" for å huske sine variabler. For program 3, vil denne post-it-lappen inneholde variablene `x,y` og `storsttall`. Selv om variablene `x` og `y` i metoden `storst` har samme navn som de variablene som finnes i hovedprogrammet har de ingen ting med hverandre å gjøre. I dette programmet vil også `x` og `y` få samme verdi etter som verdien av `x` og `y` i hovedprogrammet overføres til de nye variablene `x` og `y` i metoden `storst`. En måte å visualisere hva som skjer med variabler når man kaller en metode er å se for oss at post-it-lappen som metoden bruker for å huske sine variabler blir klistret over huskelappen for hovedprogrammet som illustrert nedenfor:



Bruken av post-it-lappen illustrerer at datamaskinen nå ikke kan se variablene i hovedprogrammet etter som den nå bruker en ny klistrelapp. Dette betyr at under kjøring av metoden `storst` vil datamaskinen ikke kunne bruke variablene `a` og `svar` som er definert i hovedprogrammet. Når datamaskinen er ferdig med å utføre metoden `storst` og returnerer det største tallet som er lagret i variabelen `storsttall` (5), vil post-it-lappen datamaskinen brukte for å huske variabler i metoden kastes! Dette betyr at hvis man kaller samme metode på nytt, vil datamaskinen **ikke** bruke den samme post-it-lappen på nytt og vil **ikke** huske verdiene av variablene (f.eks. at `storsttall` hadde verdien 5 sist gang). Ved et nytt metodekall, vil man bruke en ny post-it-lapp og alle variablene blir deklarert og gitt verdier helt på nytt. Når en datamaskin er ferdig med å utføre et metodekall og returnerer, vil den

bruke huskelappen som ligger under post-it-lappen som ble fjernet. Hvis man kaller et metode inne i en annen metode vil man få flere post-it-lapper oppå hverandre. Dette vil fungere akkurat på samme måte som illustrert ovenfor. Når man er ferdig med å utføre en metode, kaster man post-it-lappen og bruker lappen som ligger under. På denne måten kan man ha mange nivåer av metodekall som bruker egne post-it-lapper for å ta vare på metodens variabler. Man kan bruke metoden med lapper og post-it-lapper når man vil gå igjennom et program for å sjekke om programmet gjør oppgaven sin riktig.

På samme måte som man overfører parametere til en metoden, kan man også returnere parametere. For metoden `storst` er det spesifisert at en `int` skal returneres. Når metoden skriver `return storsttall` er det kun verdien til `storsttall` som blir overført tilbake til variabelen `svar` i hovedprogrammet. Vi har sett i dette eksempelet, at metoder har sine egne ”huskelapper” som kastes etter bruk og at den eneste måten å få overført noe mellom et kall av en metoden og selve metoden er igjennom parametere.