Løkker og arrayer

Løse problemer med programmering

INF1000, uke3
Geir Kjetil Sandve

Hva vi har lært så langt

- Variabler og uttrykk
- Beslutninger
- Kontrollflyt og feilmeldinger
- Metoder og parametre

Fokus i dag

- Repetert kjøring (løkker)
- Holde på mange verdier (arrayer)
- Løse problemer vha programmering

Repetert kjøring

- {u3Lokker1.java}:
  program vi så på også i forrige uke
- Programmet har to problem:
  - Repetert kode gjør det vanskeligere å ha oversikt og gjøre endringer
  - Koden var fastlåst til å spørre nøyaktig 3 ganger om alder

Repetert kjøring

- Forrige uke reduserte vi repetisjon ved å flytte kode til en metode
  - Vi trengte fremdeles 3 kall,
    og låste dermed også til å spørre 3 ganger
- Denne uka skal vi se på mekanismer for å repetere utførelse, uten å måtte repetere kode eller kall

Et eksempel på repetert kjøring

- {U3RiktigPlussing.java}

Repetert kjøring (løkke): **while**

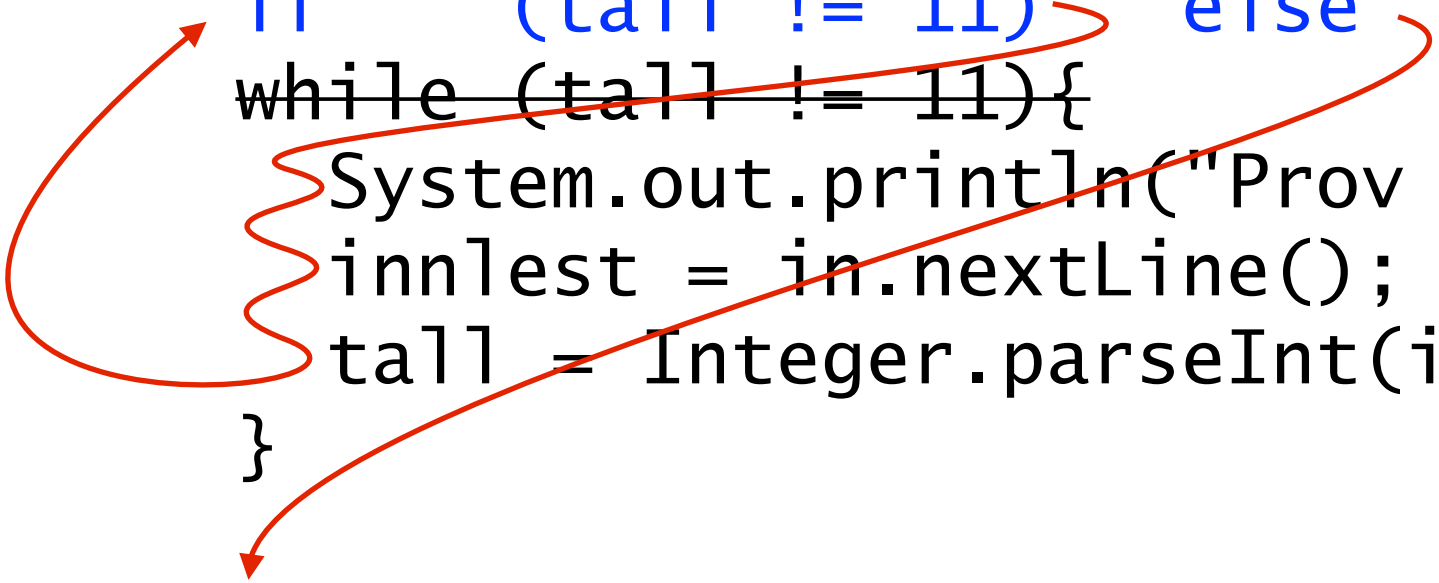
- Syntaksen er veldig rett frem:
  - while (something) {do1; do2; ...}
  - while (tall<100) {System.out.println(tall); tall+=5;}
- En slags if med tilbakekobling:
  - Nesten som if, bare at man kjører innholdet mange ganger - helt til something ikke lenger er sant

While som en **if** med tilbakekobling

```
System.out.println("Hva er 4+7?");
innlest = in.nextLine();
tall = Integer.parseInt(innlest);


while (tall != 11){
  System.out.println("Prov igjen!");
  innlest = in.nextLine();
  tall = Integer.parseInt(innlest);
}
```

While som en **if** med tilbakekobling

```
System.out.println("Hva er 4+7?");
innlest = in.nextLine();
tall = Integer.parseInt(innlest);

if    (tall != 11)  else
while (tall != 11){
   System.out.println("Prov igjen!");
   innlest = in.nextLine();
   tall = Integer.parseInt(innlest);
}
```



Eksempelet vi startet med

- {u3Lokker2.java}: repetert spørring frem til brukeren oppgir negativ alder

Merk den presise rekkefølgen ting blir gjort!

- while (something) {do1; do2; ...}
  - Man sjekker,
    kjører hele blokka,
    og går så tilbake til sjekken igjen
  - Sjekk something,
    do1, do2,
    sjekk something igjen,
    do1, do2 ...

Merk den presise rekkefølgen ting blir gjort!

- while (something) {do1; do2; ...}
- Det blir altså **ikke** sjekket noe mellom do1 og do2
  - Det som sjekkes er oppfylt når man starter å kjøre kodeblokka
  - .. men det kan slutte å være oppfylt underveis i blokka

Finjustering av når noe sjekkes og brukes

- {u3Lokker3.java}: Innlesning av alder lagt sist i løkka, slik at verdi sjekkes like etter innlesning
  - Unngår å skrive alderskommentar etter terminerende input (-1) fra bruker
- Man må legge en ekstra linje med innlesning før selve løkka begynner
- Det finnes også en alternativ while-versjon:
  - **do-while** kan i noen tilfeller være praktisk, men av liten betydning

En liten oppgave

- Skriv kode (det essensielle) som regner ut summen av tallene fra 1 til 100 *(1+2+3...+100)*
  - Prøv alene med blyant og papir i 3 minutt
  - Diskutér med nabo i 3 minutt
- {U3SumTallrekkeVhaWhile.java}

En noe mer kompleks problemstilling

- Problemstilling:
  dersom man triller to terninger, hva er sannsynligheten for å få minst 6 i sum?
- {U3TerningSumVhaWhile.java}

Initialiser-sjekk-inkrementer

- Et vanlig mønster knyttet til løkker:
  - Gi startverdi til variabel (initialiser)
  - Gå i løkke inntil variabelen passerer en gitt verdi
  - Inkrementer variabel i slutten av løkka

Initialiser-sjekk-inkrementer: for

- {U3SumTallrekkeVhaFor.java}

Initialiser-sjekk-inkrementer

```
int sum = 0;
int tall = 1; intialiser

while (tall<=100){ sjekk
  sum+=tall;
  tall +=1; inkrementer
}
System.out.println(sum);
```

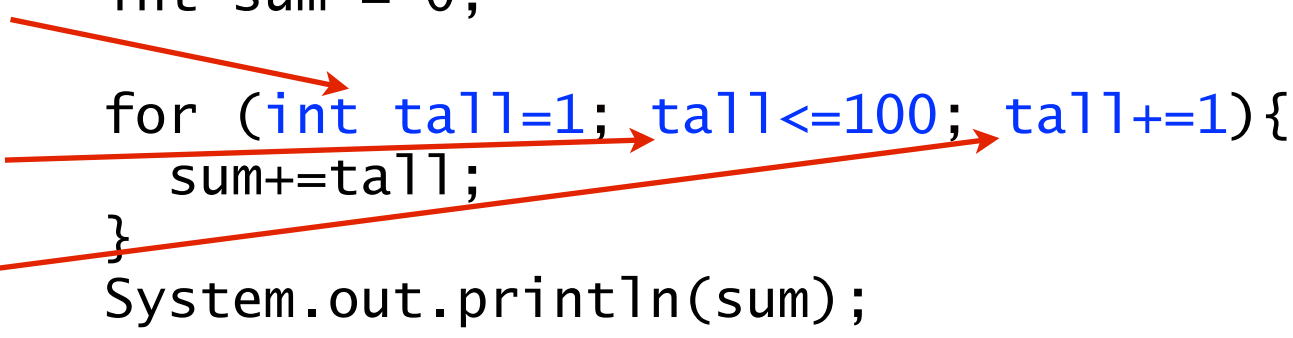
Initialiser-sjekk-inkrementer

While

```
int sum = 0;
int tall = 1; intialiser

while (tall<=100){ sjekk
  sum+=tall;
  tall +=1; inkrementer
}
System.out.println(sum);
```

For

```
int sum = 0;

for (int tall=1; tall<=100; tall+=1){
  sum+=tall;
}
System.out.println(sum);
```



Initialiser-sjekk-inkrementer

While

```
int terning1, terning2;
int antSuksesser=0;
int teller=0; intialiser

while (teller<1000){ sjekk
  teller +=1; inkrementer

  terning1=sjanse.nextInt(6)+1;
  terning2=sjanse.nextInt(6)+1;

  if (terning1+terning2 >= 6){
    antSuksesser += 1;
  }
}
System.out.println(
  antSuksesser/1000.0);
```

For

```
int terning1, terning2;
int antSuksesser=0;

for (int teller=0;teller<1000;
teller+=1){

terning1=sjanse.nextInt(6)+1;

terning2=sjanse.nextInt(6)+1;

  if (terning1+terning2 >= 6){
    antSuksesser += 1;
  }
}
System.out.println(
  antSuksesser/1000.0);
```

Initialiser-sjekk-inkrementer: **for**

- Litt mer kompleks syntaks:
  - for (initialization; condition; update) {do1; do2; ...}
- For-uttrykket inneholder tre helt ulike ting:
  - Initialization: kjøres bare én gang, før selve løkka
  - Condition: sjekkes hver runde før kodeblokka kjøres (tilsvarer uttrykket inni while)
  - Update (inkrementering): kjøres i slutten av løkka *(legges til i slutten av kodeblokka)*

Initialiser-sjekk-inkrementer: for

- **for** er egentlig ikke mer enn en forkortet skrivemåte
  - Alt man kan gjøre vha **for** kan man også gjøre vha en tilsvarende **while**
  - Eneste formål er å forkorte og tydeliggjøre en typisk bruksmåte av (while-) løkker
  - Den typiske bruken er noe som skal repeteres et fastlagt antall ganger
  - I praksis bruker man gjerne **for** oftere enn **while**

Sjonglere med flere verdier

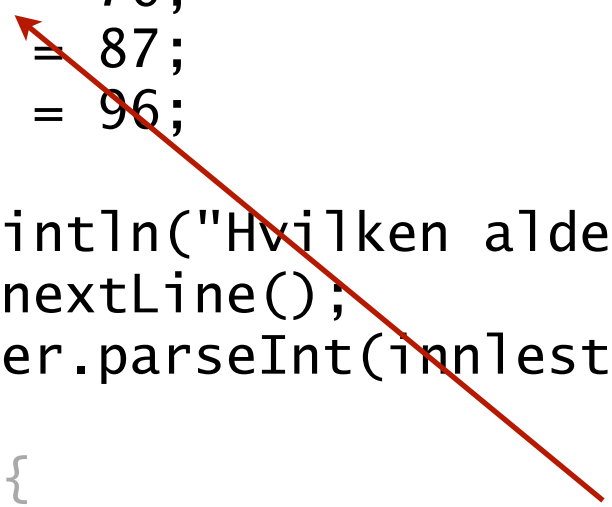
- {HoydeGittAlder1.java}

Finne verdien vi trenger direkte

```
int hoydeAar0 = 50;
int hoydeAar1 = 76;
int hoydeAar2 = 87;
int hoydeAar3 = 96;

System.out.println("Hvilken alder vil du vite om");
innlest = in.nextLine();
alder = Integer.parseInt(innlest);

if (alder==0){
  System.out.println(hoydeAar0);
} else if (alder==1){
  System.out.println(hoydeAar1);
} else if (alder==2){
  System.out.println(hoydeAar2);
```

Finne verdien vi trenger direkte

```
int hoydeAar0 = 50;
int hoydeAar1 = 76;
int hoydeAar2 = 87;
int hoydeAar3 = 96;

System.out.println("Hvilken alder vil du vite om");
innlest = in.nextLine();
alder = Integer.parseInt(innlest);

if (alder==0){
  System.out.println(hoydeAaralder);
} else if (alder==1){
  System.out.println(hoydeAar1);
} else if (alder==2){
  System.out.println(hoydeAar2);
```



Vi kan slå opp verdien vi trenger direkte!

- Det vi ønsket:
  - hoydeAar**alder**
- Syntaks i java:
  - hoydeAar[alder];
- Og før dette må vi definere hoydeAar som en array:
  - int[] hoydeAar = {50, 76, 87, 96};

Håndtere høydene i en array

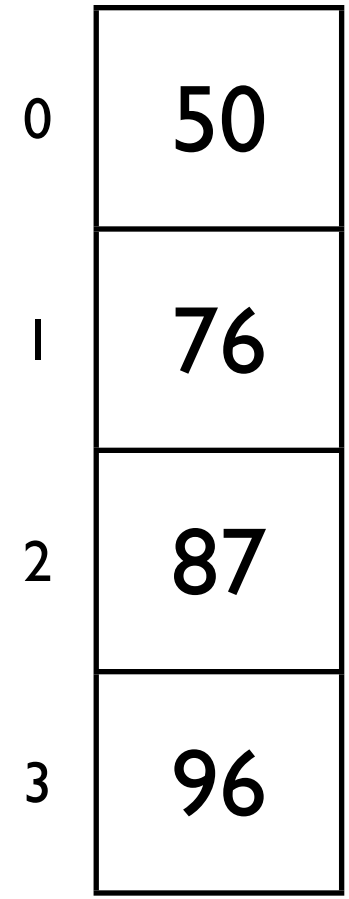
- {HoydeGittAlder2.java}

Array

- Definere array:

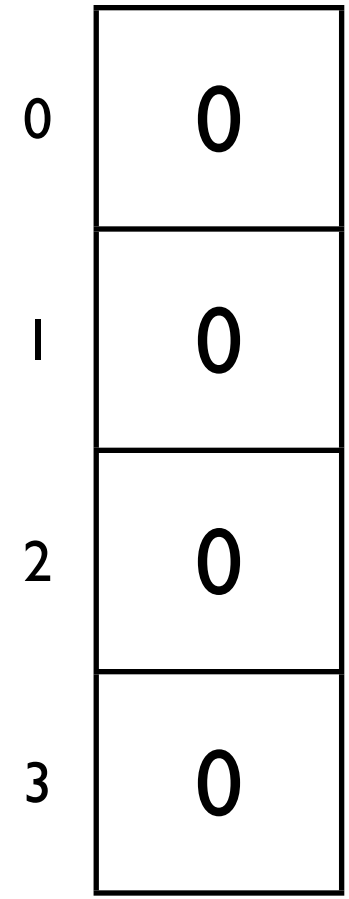
Array

- Definere array:
  - int[] hoydeAar = {50, 76, 87, 96};



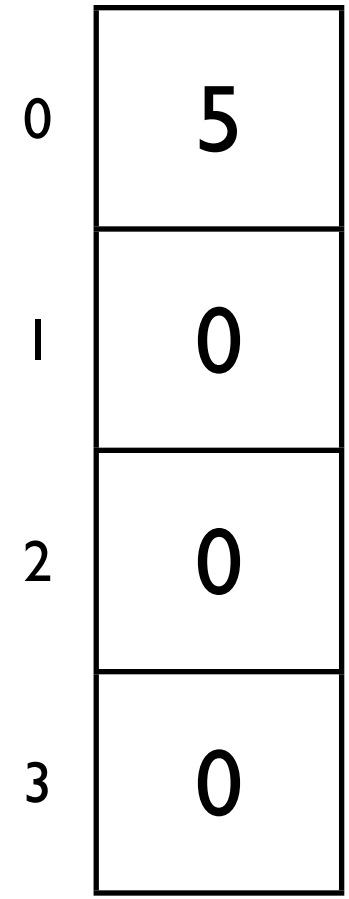
Array

- Definere array:
  - int[] hoydeAar = {50, 76, 87, 96};
  - int[] hoydeAar = new int[4];



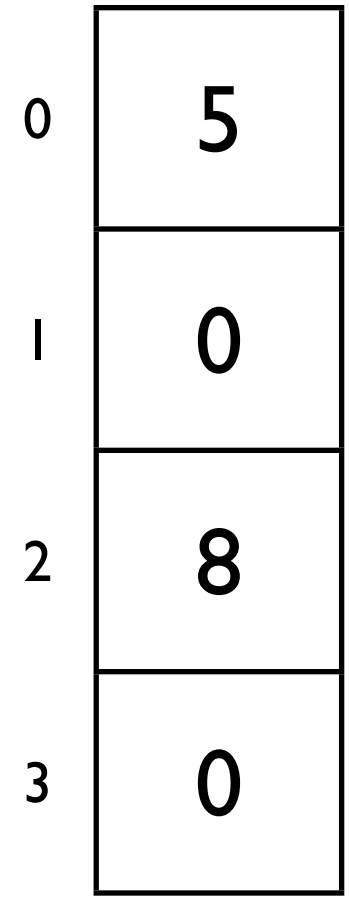
Array

- Definere array:
  - int[] hoydeAar = {50, 76, 87, 96};
  - int[] hoydeAar = new int[4];
- Sette en enkeltverdi:
  - hoydeAar[0] = 5;



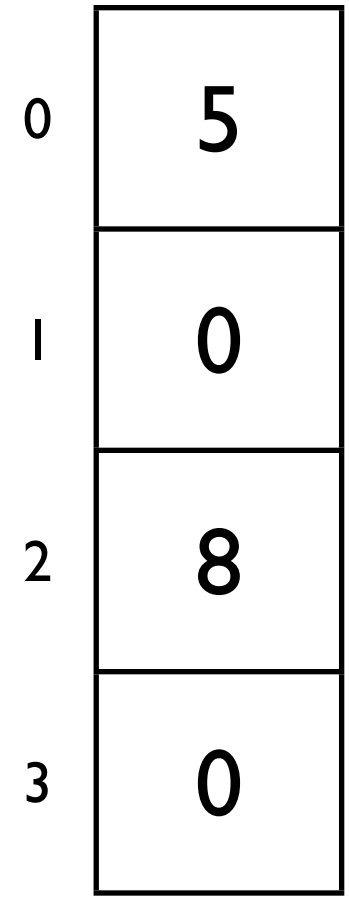
Array

- Definere array:
  - int[] hoydeAar = {50, 76, 87, 96};
  - int[] hoydeAar = new int[4];
- Sette en enkeltverdi:
  - hoydeAar[0] = 5;
  - hoydeAar[2] = 8;



Array

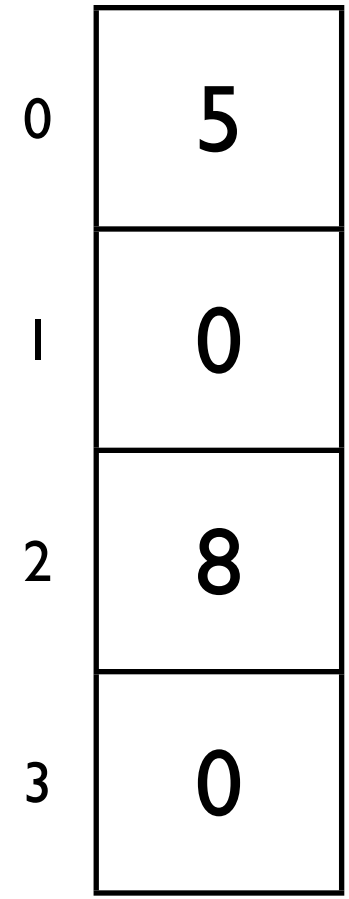
- Definere array:
  - int[] hoydeAar = {50, 76, 87, 96};
  - int[] hoydeAar = new int[4];
- Sette en enkeltverdi:
  - hoydeAar[0] = 5;
  - hoydeAar[2] = 8;
- Bruke enkeltverdi
  - System.out.println(hoydeAar[0]);



Array

- Definere array:
  - int[] hoydeAar = {50, 76, 87, 96};
  - int[] hoydeAar = new int[4];
- Sette en enkeltverdi:
  - hoydeAar[0] = 5;
  - hoydeAar[2] = 8;
- Bruke enkeltverdi
  - System.out.println(hoydeAar[0]);

5



Array - merk

- Størrelsen på en array kan bestemmes av en annen variabel
  - int hvorMangeHoyder = Integer.parseInt(in.nextLine());
  - int[] hoydeAar = new int[hvorMangeHoyder];
- Alle verdier må være samme type,
  men man kan bestemme hvilken
  - String[] hoydeAar = {"uhyre liten", "bitteliten", "liten"}
- En tom array fylles med default-verdier
  - int[] hoydeAar = new int[4];
  - *er samme som:* int[] hoydeAar = {0,0,0,0}

Array og løkker

- Array og løkker opptrer ofte sammen
  - hvorfor tror du det er slik?
    - Filosofer i tre minutt - eventuelt diskuter med nabo
- Noen forslag:
    - Arrayer tillater en løkke å operere på mange verdier - typisk én verdi for hver gjennomkjøring av løkka
    - Løkker kan oppsummere innholdet i en array - regne ut sum av verdiene, finne minste verdi osv.

Et typisk eksempel

- {U3SumAvArray.java}

Et ekte problem
(på en travel morgen)

- Problemstilling:
  - Jeg har en kopp hvor det ligger 4 løse kontaktlinser, 2 for venstre øye og 2 for høyre øye
  - Av ren latskap grabber jeg to vilkårlige linser fra koppen (av de 4) og håper at jeg har grabbet et for venstre og et for høyre (ellers må jeg lete videre).
  - Hva er sannsynligheten for at jeg fikk en for hvert øye?
- {U3_SimuleringLinser1.java-U3_SimuleringLinser4.java}

Oppsummering

- Løkker gjør at kodelinjer kan kjøres flere ganger
  - F.eks. gjøre lignende type utregning på ulike verdier
- Arrayer gjør det mulig å jobbe med mange verdier
  - Kan slå opp direkte på verdien i en bestemt posisjon
- Løkker og arrayer jobber godt sammen
  - Kan generere og oppsummere store mengder verdier
- Dere har nå verktøykassen for å løse skikkelige problemstillinger!