INF1000 (Uke 4)
Mer om forgreninger, While-løkker

Grunnkurs i programmering

Institutt for Informatikk

Universitet i Oslo

Are Magnus Bruaset og Anja B. Kristoffersen

I dag

- Repetisjon
  - easyIO
  - Enkle if-setninger
- Mer om forgrening
- While-løkker

Tre måter å lese fra terminal

- Først: importere easyIO og åpne forbindelse til tastaturet
- Lese item for item:
  - For å lese et heltall: inInt()
  - For å lese et desimaltall: inDouble()
  - For å lese ett ord: inWord()
  - For å lese alle ord: inWord("\n")

Tre måter å lese fra terminal

- Lese linje for linje:
  - For å lese resten av linja: inLine()
- Lese tegn for tegn:
  - For å lese neste tegn (også hvite tegn): inChar()

Heltallsdivisjon

- Java konverterer ikke fra heltall til desimaltall når to heltall adderes, subtraheres, multipliseres eller divideres:

  - 234 + 63 : heltall (int)
  - 235 - 23 : heltall (int)
  - 631 * 367 : heltall (int)
  - 7 / 2 : heltall (int)

Heltallsdivisjon

- Legg spesielt merke heltallsdivisjonen:

Når to heltall divideres på hverandre i Java blir resultatet et heltall, selv om vanlige divisjonsregler tilsier noe annet. Dette kalles heltallsdivisjon, og resultatet er det samme som om vi fulgte vanlige divisjonsregler og så avrundet nedover til nærmeste heltall. Dvs (7/2) = = (int) (7.0/2.0) = = 3.

Oppgave

- Avgjør i hvert tilfelle hvilken datatype resultatet har:

Uttrykk	Datatype
2 + 6 * 3	int
14.2 + 6	double
3/2 + 4	int
"Vekt: " + 25 + " kg"	String
"" + 17.4	String
Math.ceil(5.3) + (int) 3.25	double

Repetisjon: If-setninger

- If-setninger uten else-gren:

```
if (logisk uttrykk){
  /* Her kommer de instruksjonene
     som skal utføres når det logiske
     uttrykket er sant (true) */
}
```

Repetisjon: If-setninger

- If-setninger med else-gren:

```
if (logisk uttrykk){
    /* Her kommer de instruksjonene
       som skal utføres når det
       logiske uttrykket er sant (true) */
} else {
    /* Her kommer de instruksjonene
       som skal utføres når det
       logiske uttrykket er usant (false) */
}
```


If-setninger kan settes sammen

- Sammensetning av flere if-setninger:

```
if (x < 0) {
  System.out.print("Tallet er negativt");
} else if (x == 0) {
  System.out.print("Tallet er null");
} else {
  System.out.print("Tallet er positivt");
}
```


Eksempel: Adgang til studier

- Problem:

  For å få adgang til universitetet i Ruritania må man oppfylle begge disse kravene:

  alder $\geq$ 18

  karaktersnitt > 4.8

  Lag et program som, gitt en persons alder og karaktersnitt, skriver ut på skjerm om personen får adgang til universitetet eller ikke.

Løsnings-skisse

```
import easyIO.*;

class AdgangTilUniversitetet {
  public static void main (String [] args) {
    <deklarasjoner>

    <les alder og karaktersnitt fra terminal>

    <avgjør om personen kan få adgang og
     skriv ut resultatet>
  }
}
```

Ting vi må ta stilling til:

- Hvilke variable trenger vi? Hva slags type skal de ha?
- Hvordan avgjør vi om personen skal få adgang eller ikke?
- Hvordan skal utskriften se ut?

Løsning

```
import easyIO.*;

class AdgangTilUniversitetet {
  public static void main (String [] args) {









  }
}
```



```
import easyIO.*;

class AdgangTilUniversitetet {
  public static void main (String [] args) {

    int    alder;
    double karSnitt;

    Out skjerm = new Out();
    In  tast   = new In ();

    skjerm.out("Alder: ");          alder    = tast.inInt();
    skjerm.out("Karaktersnitt: ");  karSnitt = tast.inDouble();

    if (alder >= 18) {
      if (karSnitt > 4.8)
        skjerm.outln("Personen kan tas opp");
      else
        skjerm.outln("Personen kan ikke tas opp");
    } else
      skjerm.outln("Personen kan ikke tas opp");
  }
}
```

Oppgave

- Hva blir skrevet ut av dette programmet ? (les programmet nøye)

```
class IfTest {
  public static void main (String [] args) {
    String s = "Petter";

    if (s.equals("Jens"));
    {
      System.out.println("Ordet var " + s);
    }
  }
}
```


Svar:

```
sjau Uke4/IfTest2>java IfTest2
B
```

Oppgave 2

Hva blir skrevet ut av dette programmet?

```
class IfTest2 {
  public static void main (String [] args) {
    double x = -0.5;
    double y = 0.5;

    if (Math.ceil(x) == Math.ceil(y)) {
      System.out.println("A");
    }
    if ((int) x == (int) y) {
      System.out.println("B");
    }
    if (x < y) {
      if (x < 0) {
        if (y < 0) {
          System.out.println("C");
        }
      } else {
        System.out.println("D");
      }
    }
  }
}
```

Alternativ til if-else: switch

- En sammensetning av flere if-setninger kan i noen tilfeller erstattes med en switch-setning:

```
switch (uttrykk) {
  case verdi1:
     <instruksjoner>
     break;
  …
  case verdiN:
     <instruksjoner>
     break;
  default:
     <instruksjoner>
  }
```

Et uttrykk som gir en verdi som er av en av typene char eller int (evt. byte eller short)

Break

- Nøkkelordet break avbryter utførelsen av switch-setningen
- Når break mangler, fortsetter utførelsen på neste linje (det er sjelden ønskelig)

Eksempel

```
class BrukAvSwitch {
  public static void main (String [] args) {
    char c = 'x';

    switch(c) {
      case 'a':
        System.out.println("Tegnet var en a");
        break;
      case 'b':
        System.out.println("Tegnet var en b");
        break;
      default :
        System.out.println("Tegnet var ikke a eller b");
    }
  }
}
```


Oppgave

- Hva blir skrevet ut av dette programmet?

```
class Divisjon {
  public static void main (String [] args) {
    if (1/2 > 0) {
      System.out.println("A");
    } else {
      System.out.println("B");
    }
  }
}
```

Blokker

- En blokk er en samling instruksjoner omgitt av krøllparenteser:

```
{
 instruksjon 1;
 instruksjon 2;
 ....
 instruksjon n;
}
```

- Alle steder i et Java-program hvor det kan stå en instruksjon, kan vi om ønskelig i stedet sette en blokk

Eksempel

```
class Blokker {
   public static void main (String [] args) {
     double x = -10.2;
     double absx;

     if (x < 0) {
        absx = -x;
        System.out.println("Absoluttverdien er: " + absx);
     } else {
        absx = x;
        System.out.println("Absoluttverdien er: " + absx);
     }
   }
}
```

her er en blokk

her er en blokk

Deklarasjoner inne i blokker

- Vi har lov til å deklarere variable inne i en blokk, forutsatt at de ikke allerede er deklarert utenfor blokken
- Eksempel:

```
double x = 0.3;
if (x < 0) {
 double y; // y er deklarert inne i en blokk
 y = -x;
}
```


Deklarasjoner inne i blokker

- En variabel deklarert inne i en blokk (f.eks. y ovenfor) slutter å eksistere når blokken er utført
- Derfor er dette ulovlig:

```
double x = 0.3;
if (x < 0) {
 double y;
 y = -x;
}
x = y;   // Ulovlig, siden y ikke eksisterer her
```

While-setninger

- Vi kan få utført en instruksjon (eller en blokk) mange ganger ved hjelp av en while-setning (også kalt while-løkke):

```
while (logisk uttrykk) {
  setning 1;
  setning 2;
  .....
  setning n;
}
```


While-setninger

- Virkemåte:
  1. Det logiske uttrykket regnes ut
  2. Hvis uttrykket er sant (true), utføres setning 1,2,...,n, og deretter går vi til punkt 1 ovenfor (NB: det logiske uttrykket kan da ha skiftet verdi!)
  3. Hvis uttrykket er usant (false), avsluttes while-setningen


Eksempel på while-setning

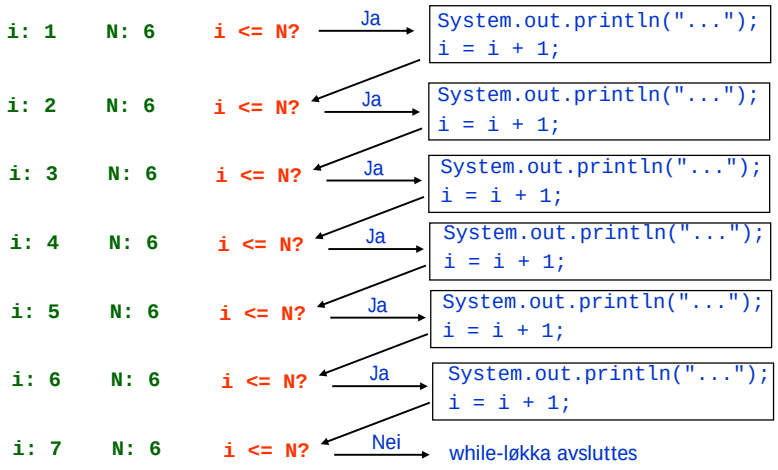
```
class SkrivLinjer {
   public static void main (String [] args) {
     int N = 6;
     int i = 1;

     while (i <= N) {
       System.out.println("Jeg må lese til eksamen");
       i = i + 1;
     }

     System.out.println("Nå er while-løkka ferdig");

   }
}
```


Hva skjer når while-setningen utføres?



Kompilering og kjøring

```
> javac SkrivLinjer.java
> java SkrivLinjer
Jeg må lese til eksamen
Jeg må lese til eksamen
Jeg må lese til eksamen
Jeg må lese til eksamen
Jeg må lese til eksamen
Jeg må lese til eksamen
Nå er while-løkka ferdig
```


Evig løkke

Dersom testen i while-løkka aldri blir usann (false), vil utførelsen av while-løkka aldri stoppe. Dette kalles en evig løkke

```
class EvigLokke1 {
  public static void main (String [] args) {
    while (true) {
      System.out.println("INF 1000");
    }
  }
}
```

```
class EvigLokke2 {
  public static void main (String [] args) {
    int i = 1, j = 2;
    while (i < j) {
      System.out.println("Nå er i < j");
    }
  }
}
```


Kompilering og kjøring

```
> javac
EvigLokke1.java
> java
EvigLokke1
INF 1000
INF 1000
INF 1000
INF 1000
INF 1000
INF 1000
     ...
     ...
     ...
  (osv)
```

```
> javac
EvigLokke2.java
> java
EvigLokke2
Nå er i < j
Nå er i < j
Nå er i < j
Nå er i < j
Nå er i < j
Nå er i < j
     ...
     ...
     ...
  (osv)
```


Oppgave

- Hva blir utskriften fra dette programmet?

```
class LokkeTest {
  public static void main (String [] args) {
    int i = 3;

    while (i > 0) {
      System.out.print("Nå er i = ");
      System.out.println(i);
      i = i - 1;
    }
  }
}
```

Oppgave 2

- Hva blir utskriften fra dette programmet?

```
class LokkeTest2 {
   public static void main (String [] args) {
     int i = 1;

     while (i < 4) {
       System.out.print("Nå er i = ");
       System.out.println(i);
       i = i + 1;
     }
   }
}
```


Eksempel: Gangetabell

- Problem:

  Lag et program som bruker en while-løkke til å beregne "tregangen" og lage en slik utskrift på skjermen:

```
1 * 3 = 3
2 * 3 = 6
3 * 3 = 9
4 * 3 = 12
5 * 3 = 15
6 * 3 = 18
7 * 3 = 21
8 * 3 = 24
9 * 3 = 27
10 * 3 = 30
```


Løsnings-skisse

```
class TreGangen {
  public static void main (String[] args) {

    <deklarasjoner>

    <initialisering av variable>

    <while-løkke, hvor hvert gjennomløp
     regner ut og skriver ut en ny linje
     på skjermen>

  }
}
```


While-løkka

```
while (ikke ferdig) {
   <finn svaret på neste regnestykke>
   <skriv ut svaret>
}
```

For å finne svaret på neste regnestykke, må vi holde rede på de to tallene som skal ganges sammen. Vi trenger bare en variabel for dette, siden annen faktor alltid er lik 3. Vi lager også en variabel for svaret:

```
int k, svar;
```

Vi initialiserer slik:

```
k = 1;
```

Dermed blir svaret på neste regnestykke:

```
svar = k * 3;
```

while-løkka *forts.*

```
while (ikke ferdig) {
  svar = k * 3;
  System.out.print(k + " * 3 = " + svar);
```

Vi må i tillegg huske å endre verdien til k i slutten av hvert gjennomløp (ellers gjør vi samme regnestykke igjen og igjen):

```
k = k + 1;
```

Hvor lenge skal while-løkka løpe ? Vi kan bruke denne testen:

```
k <= 10
```


Ferdig program

```
class TreGangen {
  public static void main (String[] args) {
    int k=1, svar;

    while (k <= 10) {
      svar = k * 3;
      System.out.println(k + " * 3 = " + svar);
      k = k + 1;
    }

  }
}
```


Kompilering og test

```
> javac TreGangen.java
> java TreGangen
1 * 3 = 3
2 * 3 = 6
3 * 3 = 9
4 * 3 = 12
5 * 3 = 15
6 * 3 = 18
7 * 3 = 21
8 * 3 = 24
9 * 3 = 27
10 * 3 = 30
```


Eksempel: stolpediagram

- Problem:

  Anta f.eks. at fire mobiltelefoner koster henholdsvis 600, 300, 900 og 1200 kroner. Det kan vi visualisere slik:

  mobil 1: ******
  mobil 2: ***
  mobil 3: *********
  mobil 4: ************

  Antall stjerner er i hvert tilfelle lik prisen / 100.

  Vi skal lage et program som lager et slikt stolpediagram for to mobiltelefoner, gitt prisen på de to telefonene.

Hvilke data beskriver problemet?

- Input:
  - Prisen til de to telefonene (to desimaltall)
- Output:
  - Antall stjerner som skal skrives ut for hver av telefonene (to heltall)

Fremgangsmåte

- Antall stjerner som skal skrives ut for hver telefon er prisen delt på 100. Vi må også avrunde til et heltall

  Dette kan vi f.eks. gjøre slik:

```
antall1 = (int) (pris1 / 100);
antall2 = (int) (pris2 / 100);
```

- For å få skrevet ut riktig antall stjerner bruker vi en while-løkke:
  - Hvert gjennomløp av løkka skriver ut en enkelt stjerne
  - Vi teller opp antall gjennomløp av løkka (= antall stjerner) og stopper når antallet er det vi ønsker

Løsnings-skisse

```
class Stolpediagram {
  public static void main (String[] args) {
    <deklarasjoner>

    <les inn pris1 og pris2 fra terminal>

    antall1 = (int) (pris1 / 100);
    <skriv ut "mobil 1: ">
    <skriv ut antall1 stjerner>

    antall2 = (int) (pris2 / 100);
    <skriv ut "mobil 2: ">
    <skriv ut antall2 stjerner>

  }
}
```


Hvordan skrive ut antall1 stjerner?

Vi kan bruke en while-løkke:

```
int i = 0;
while (i < antall1) {
  System.out.print("*");
  i = i + 1;
}
```

En ting mangler: trenger et linjeskift etter at alle stjernene er skrevet ut (slik at neste stolpediagram kommer på ny linje). Vi kan gjøre dette slik (etter while-løkka):

```
System.out.println(""); // Denne lager linjeskift
```

```
import easyIO.*;

class StolpeDiagram {
  public static void main (String[] args) {
    int antall1, antall2, i;
    In tastatur = new In();

    System.out.print("Pris på telefon 1: ");
    double pris1 = tastatur.inDouble();
    System.out.print("Pris på telefon 2: ");
    double pris2 = tastatur.inDouble();

    System.out.print("mobil 1: ");
    antall1 = (int) (pris1 / 100);
    i = 0;
    while (i < antall1) {
      System.out.print("*");
      i = i + 1;
    }
    System.out.println("");   // Start ny linje på skjermen

    System.out.print("mobil 2: ");
    antall2 = (int) (pris2 / 100);
    i = 0;
    while (i < antall2) {
      System.out.print("*");
      i = i + 1;
    }
    System.out.println("");   // Start ny linje på skjermen
  }
}
```

Kompilering og test

```
> javac Stolpediagram.java
> java Stolpediagram
Pris på telefon 1: 800
Pris på telefon 2: 1200
mobil 1: ********
mobil 2: ************
```


Eksempel: Finne gjennomsnitt

- Problem:

  Lag et program som leser en rekke desimaltall fra terminal, helt til brukeren oppgir tallet 999 som betyr at innlesningen skal avsluttes.

  Programmet skal deretter regne ut gjennomsnittet av de innleste verdiene og skrive ut svaret på skjermen.

Løsnings-skisse

- Framgangsmåte:
  - Vi bruker inDouble() til å lese desimaltallene
  - Vi lager en while-løkke for innlesningen, slik at innlesningen kan foretas så mange ganger det er ønsket
  - Hver gang vi leser en ny lovlig verdi, adderer vi det til summen av de foregående verdiene. Vi holder også rede på hvor mange verdier som er innlest
  - Testen i while-løkka sørger for stopp når siste innleste verdi er 999
  - Etter while-løkka deler vi summen på antall innleste verdier, og skriver ut svaret på skjermen

While-løkka

```
sum = 0;
x = 0;

while (x != 999)
{
   <les x fra terminal>

   if (x != 999)
   {
      sum = sum + x;
      antall = antall + 1;
   }

}
```

Sett x lik en hvilken som helst verdi forskjellig fra 999

Så lenge vi ikke har lest stoppsignalet 999, fortsetter vi innlesningen

Vi oppdaterer variablene sum og antall

Når vi kommer hit, *kan* vi risikere at *ingen* verdier er lest (kun 999), så vi vet ikke om sum/antall er veldefinert

Ferdig program

```
import easyIO.*;

class FinnGjennomsnitt {
  public static void main (String[] args) {
    double x = 0, sum = 0;
    int antall = 0;
    In tast = new In();
    Out skjerm = new Out();

    while (x != 999) {
      skjerm.out("Oppgi et desimaltall (999 for å avslutte): ");
      x = tast.inDouble();
      if (x != 999) {
        sum = sum + x;
        antall = antall + 1;
      }
    }

    if (antall == 0) {
      skjerm.outln("Ingen verdier ble oppgitt!");
    } else {
      skjerm.out("Gjennomsnitt: ");
      skjerm.outln(sum/antall, 2);
    }

  }
}
```

Eksempel: Innlesning med sjekk

- Problem:

  Lag et program som leser et heltall mellom 1 og 100 fra terminal.

  Hvis det innleste tallet ikke ligger i det lovlige intervallet, skal programmet be om nytt tall.

Løsnings-skisse

- Framgangsmåte:
  - Vi bruker metoden inInt() til å lese heltallet fra terminal
  - Vi legger selve innlesningen inni en while-løkke, slik at innlesningen om nødvendig kan utføres flere ganger
  - Testen øverst i while-løkka må være true når vi første gang kommer til while-løkka, og vi må sørge for at verdien blir false straks vi har lest en lovlig verdi (slik at videre innlesning stopper)

Programskisse

```
import easyIO.*;

class LesVerdi {
  public static void main (String[] args) {

    <deklarasjoner>

    boolean fortsett = true;

    while (fortsett)
    {
      <les inn heltall fra terminal>

      <sett fortsett lik false hvis lovlig verdi,
       og gi feilmelding hvis ulovlig verdi>
    }

    <skriv ut verdien>
  }
}
```


Ferdig program

```
import easyIO.*;

class LesVerdi {
  public static void main (String[] args) {
    int verdi = 0;
    boolean fortsett = true;
    In tast = new In();

    while (fortsett)
    {
      System.out.print("Oppgi verdi (1,2,...,100): ");
      verdi = tast.inInt();

      if (verdi >= 1 && verdi <= 100)
      {
        fortsett = false;
      }
      else
      {
        System.out.println("Ulovlig verdi! Prøv igjen!");
      }
    }

    System.out.println("Du oppga verdien " + verdi);
  }
}
```