UNIVERSITETET I OSLO

INF1000 – Uke 2

Dagens emner:
- Litt repetisjon
- Uttrykk
- Innlesing fra terminal
- Formatert utskrift

UNIVERSITETET I OSLO

Repetisjon – Program

- Program skrives i et programmeringsspråk
- Imperativ programmering: **Setninger** utføres i **sekvens**, ovenfra og nedover
- Variable **deklareres**
- Variable **tilordnes** verdier i setninger som kan inneholde **uttrykk**
- Program skrives i en fil, kompileres og kjøres.

UNIVERSITETET I OSLO

Repetisjon – Program

Alt inne i klasser

Prosedyren "main"

```
class Repetisjon {
   public static void main(String[] args){
       final double PI = 3.14;  // final betyr konstant
       double radius = 2.0;
       double areal;
       String FORTEKST =
              "Arealet til en sirkel med radius ";

       // Utregning
       areal = PI * radius * radius;

       System.out.println(FORTEKST + radius +
              " ER " + areal + ".");
   }
}
```

En kommentar

Setninger avsluttes med semikolon

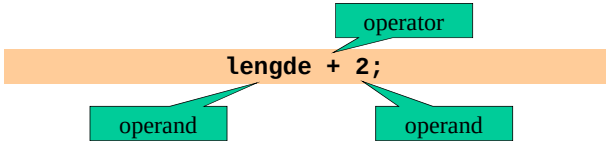
UNIVERSITETET I OSLO

Repetisjon – Variable og uttrykk

- En variabel er en plass i maskinens lager (minne)
- Variable må ha **navn**
  - Slik at vi kan angi i programmet vårt hvilken plass i lageret vi snakker om
- Variable må ha **type**
  - Så vi vet hvor stor plass variabelen tar og hva det er som ligger der

Uttrykk

- Operatorer
  - Utføres på en eller flere operander



- Metodekall
  - Operandene forekommer inne i parenteser

```
Math.ceil(x);
```

Utrykk

- Aritmetiske uttrykk:

```
2 * (3+4) / 1.5
2 / (12 + 34 - 2.3)
```

- Logiske uttrykk:

Uttrykket	har verdien true hvis	....og verdien false ellers
`x < y`	`x mindre enn y`	
`x <= y`	`x mindre enn eller lik y`	
`x == y`	`x lik y`	
`x !=y`	`x ikke lik y`	
`x > y`	`x større enn y`	
`x >= y`	`x større enn eller lik y`	
`! (x < y)`	`ikke x mindre enn y`	
`b1 && b2`	`både b1 og b2 sann`	
`b1 || b2`	`b1 eller b2 (eller begge) sann`	

Eksempel: Gangetabell

```
class Gangetabell {
   public static void main (String [] args) {
      System.out.println(1 * 8);
      System.out.println(2 * 8);
      System.out.println(3 * 8);
      System.out.println(4 * 8);
      System.out.println(5 * 8);
   }
}
```

KOMPILERING OG KJØRING

```
> javac Gangetabell.java
> java Gangetabell
8
16
24
32
40
```

Numeriske verdier - typer

Type	Lovlige verdier
byte	-128 til 127
short	-32 768 til 32 767
int	-2^{31} til $2^{31}-1$
long	-2^{63} til $2^{63}-1$
float	-3.4e38 til 3.4e38
double	-1.7e308 til 1.7e308

Tolking av literaler

- Tallene vi skriver i programmet vårt kalles literaler.
- De som kun inneholder tall tolkes som **int**.

```
int verdi = 12345;
```

- De som inneholder desimaltegn tolkes som **double**.

```
double verdi = 12.345;
```

long og float (Nb! Dette er en detalj)

- Dersom vi ønsker å skrive inn en long eller float må vi sette på et tegn bak tallet for å instruere kompilatoren.

```
long verdi1 = 12345678901234;  // Feil: Tallet blir
// tatt som integer, men det er for stort for integer

long verdi2 = 12345678901234L; // OK, L betyr ”long”
```

```
float verdi3 = 12.345;  //Feil: dataverdien er double.
// Den kan ikke automatisk konverteres til float

float verdi4 = 12.345F; // OK, F betyr ”float”
```

Typekonvertering

- Det er mulig å konvertere fra en datatype til en annen
- Enkelt fra heltall til et flyttall.
- Den andre veien må vi informere kompilatoren om
- Vi gjør det ved å sette typenavnet i parentes rett foran verdien vi ønsker å konvertere

```
double d = 3.14;
int i = (int) d;
int j = (int) 2.222;

// Men dette er altså ok
int x = 9;
double db = x;
```

Typekonvertering

- Utvidelse av typen:
  - Tillater flere dataverdier
  - Automatisk
- Innsnevring av typen
  - Tillater færre dataverdier
  - Eksplisitt
- Rekkefølgen (utvidelse mot høyre)

```
byte, short, int, long, float, double
```

Jobbe med heltall

- Heltall og heltall gir heltall
- Heltall og flyttall gir flyttall

```
int a  = 354 + 32 // a: 386
int b  = 354 - 32 // b: 332
int c  = 3 * 12   // c: 36
int d  = 5 / 2    // d: 2
```

- Heltallsdivisjon gir det største tallet som går opp i divisjonen

Heltallsdivsjon

- Heltallsdivisjon gir det største tallet som går opp i divisjonen (f.eks 13/5=2)
- Vi kan skrive et tall a som:
  a = k * b + rest

  der rest er den minste mulige verdien gitt k
  Eks: 13 = 2 * 5 + 3
- Resten finner vi med modula-operatoren (%)

```
int a  = 13
int b  = 5
int k  = 13 / 5;  //Nå er k lik 2
int r  = 13 % 5;  //Nå er r lik 3 (rest)
```

Divisjon og rest modulo 12

- Tenk deg at du skal "telle rundt klokka" gitt en urskive som viser 12 timer.
- Vi starter øverst og sier at klokka da er 0
- Hvis du teller 14 timer, har du gått rundt hele klokka 1 gang og klokka viser 2.
- Du har nå telt 14 modulo 12:
  - 14 / 12 er antall ganger du har gått rundt urskiven (dvs 1)
  - 14 % 12 er det tallet som urviseret står på (dvs 2).
- Nytt eksempel:
  - 25 / 12 == 2
  - 25 % 12 == 1

Evaluering av uttrykk

- Evaluering av uttrykk kan alltid styres med parenteser
- Av og til kan vi unngå å sette parenteser:
  - 2 + 5 * 4 - 3 **er lik** 2 + (5*4) – 3
  - b1 || b2 && b3 **er lik** b1 || (b2 && b3)
  - b1 && b2 = = b3 || b4 **er lik** (b1 && b2) = = (b3 || b4)
- Hovedregel: *Sett parenteser!*

Inkrementering og dekrementering

- Variablene som brukes i uttrykk endres ikke ved utregning, med to unntak.
- ++ og -- endrer verdien til variabelen med + eller minus 1
- ++antall Endrer antall for utrykket regnes ut
- antall++ Endrer antall etter at utrykket er regnet ut

```
int svar, i=0, j=0;
svar  = ++i; // i blir først 1. Så settes svar
             // lik i, altså 1
svar  = j++; // først settes svar lik j, altså 0.
             // Så økes j til 1.
// Dermed: i==1, j==1, svar==0
```

Oppgave

```
boolean b1, b2;
double x, y;
int z;

x = 45.33;
y = x + 1;
z = 0;
b1 = (x < y) && (z == 0);
b2 = false || b1;
```

Hva er verdien til b1 og til b2 etter at setningene over er utført?

Løsning

```
boolean b1, b2;
double x, y;
int z;

x = 45.33;     // x=45.33,    y=,           z=
y = x + 1;     // x=45.33,    y=46.33,      z=
z = 0;         // x=45.33,    y=46.33,      z=0


// x<y: true, z==0: true
b1 = (x < y) && (z == 0); // b1: true
b2 = false || b1;         // b2: true
```

Greit å vite

- Multiplikasjon må alltid angis eksplisitt med *:
  - `int prod = 10 a;` // feil!!
  - `int prod = 10 * a;` // riktig
- Det er forskjell på = og = = :
  - `=` brukes for å sette verdien til en variabel
  - `==` brukes for å sammenlikne to verdier
- Hvis vi har variabelen `boolean b` så er det ingen forskjell på
  `b == true`
  `b`
- Ekstra parenteser kan øke leseligheten for mennesker:
  `b = x == y;` betyr det samme som `b = (x == y);`

Kommentarer i programmer

- Kommentarer gjør programmene lettere å forstå
- De oversettes ikke: kompilatoren hopper over dem
- To typer kommentarer:

```
// Her er en kommentar som varer ut linja

/* Her er en kommentar som varer
        helt til hit */
```

- Gode programmer har kommentarer, men ikke på hver linje!
- Dere må kommentere programmene til oblig 2-4!

Hvorfor ikke alltid bruke double?

- Mens regning med heltall alltid er eksakt, er regning med desimaltall ikke:

```
double x = 0.1;
double y = (x + 1) - 1;
// Nå er verdien til x == y false!
```

- x og y er nesten like, men det er forskjell i et av desimalene langt ute
- x==y er dermed false.
- Når det er naturlig å bruke heltall bruker vi int!
- Når det er naturlig å bruke desimaltall bruker vi double

Hva er vitsen med class?

- En setning av typen

```
class {
  <...masse rart...>
}
```

kalles en klassedeklarasjon (eller bare klasse).

- Tenk på en klasse som en samling data (tall, tekst, bilder, osv) og operasjoner som vi ønsker å kunne utføre på dataene.
- Senere i kurset kommer hvert program til å bestå av mange klasser.

Presedensregler

- **Presedensregler** angir hvilke operatorer som har fortrinn (førsterett) ved utregning av sammensatte uttrykk.
- F.eks. blir * beregnet før +. Vi sier at
  - * har **høyere** presedens enn +
  - + har **lavere** presedens enn *
- Reglene står i læreboka!
- Tips: Bruk parenteser!

Aritmetisk uttrykk – skjulte parenteser

```
    2.0*3  + Math.ceil(7.23)    +  (2 + 3) * 3.5

    2.0*3  + (Math.ceil(7.23))  +  (2 + 3) * 3.5

   (2.0*3) + (Math.ceil(7.23))  +  (2 + 3) * 3.5

   (2.0*3) + (Math.ceil(7.23))  + ((2 + 3) * 3.5)

  ((2.0*3) + (Math.ceil(7.23))) + ((2 + 3) * 3.5)

 (((2.0*3) + (Math.ceil(7.23))) + ((2 + 3) * 3.5))
```

Aritmetisk uttrykk – utregning

```
 (((2.0*3) + (Math.ceil(7.23))) + ((2 + 3) * 3.5))
  ((6.0    + (Math.ceil(7.23))) + ((2 + 3) * 3.5))

  ((6.0    +  8.0)              + ((2 + 3) * 3.5))

   (14.0                        + ((2 + 3) * 3.5))

   (14.0                        + ( 5      * 3.5))

   (14.0                        +   17.5)

    31.5
```

Eksempel – Logisk uttrykk

```
int u=1,v=1;
boolean b = true;

boolean resultat =
    u>2 || v<2 &&  b == !(++u == v++);

// Hva blir resultat?
```

Presedensregler for logiske uttrykk

- Høyest: Metodekall
- ! (ikke)
- <, <=, >, >=
- ==, !=
- && (og)
- || (eller)

Logisk uttrykk – skjulte parenteser

```
   u>2  ||   v<2  &&  b ==  !( ++u  ==  v++)
   u>2  ||   v<2  &&  b ==  !((++u) == (v++))
   u>2  ||   v<2  &&  b == (!((++u) == (v++)))
  (u>2) ||   v<2  &&  b == (!((++u) == (v++)))
  (u>2) ||  (v<2) &&  b == (!((++u) == (v++)))
  (u>2) ||  (v<2) && (b == (!((++u) == (v++))))
  (u>2) || ((v<2) && (b == (!((++u) == (v++)))))
 ((u>2) || ((v<2) && (b == (!((++u) == (v++))))))
```

Logisk uttrykk – utregning

```
((u>2)  || ((v<2) && (b == (!((++u) == (v++))))))
( false || ((v<2) && (b == (!((++u) == (v++))))))
( false || (true  && (b == (!((++u) == (v++)))))) u: 1, v: 1
( false || (true  && (b == (!(    2 == (v++)))))) u: 2, v: 1
( false || (true  && (b == (!(    2 == 1 )))))  u: 2, v: 2
( false || (true  && (b == (! false ))))
( false || (true  && (b == true )))
( false || (true  && true ))
( false || true)
  true
```

Eksempel – String

```
int x = 2, y = 3;

String tekst = "2+3==" + (x + y) + "!=" + x + y;

System.out.println("tekst: " + tekst);

// Hva blir skrevet ut?
```

String – skjulte parenteser

```
     "2+3==" + (x + y)  + "!="  + x + y

(    "2+3==" + (x + y)) + "!="  + x + y

((   "2+3==" + (x + y)) + "!=") + x  + y

(((  "2+3==" + (x + y)) + "!=”) + x) + y

((((“2+3==" + (x + y)) + "!=”) + x) + y)
```

String – utregning av verdi

```
((("2+3==" + (x + y)) + "!=") + x) + y)

((("2+3==" +  5     ) + "!=") + x) + y)

(( "2+3==5"          + "!=") + x) + y)

(  "2+3==5!="                + x) + y)

   "2+3==5!=2"                  + y)

   "2+3==5!=23"
```

Presedens – Oppsummert

- Bruk parenteser for mye heller enn for lite
- Helst ikke bruk ++ og -- inne i lengre uttrykk
- Se på eksempler og oppgaver og øv på bruk av uttrykk

Matematiske funksjoner - Avrunding

```
class Avrunding {
  public static void main (String [] args) {
      double x = 0.53;
      // Avrund nedover:
      System.out.println((int) Math.floor(x));
      // Avrund oppover:
      System.out.println((int) Math.ceil(x));
      // Avrund til nærmeste heltall:
      System.out.println((int) Math.round(x));
  }
}
```

Resultat

```
$ javac Avrunding.java
$ java Avrunding
0
1
1
$
```

Kompileringsfeil og kjøretidsfeil

- Kompileringsfeil
  - Feil som oppdages av **javac**
  - Feilformulerte setninger
  - Feil type
  - Programmet blir ikke kompilert
  - Husk: Tidligere kompilerte utgaver kan ligge der
- Kjøretidsfeil
  - Feil som oppdages av **java**
  - Feil vi ikke kunne vite om før programmet ble kompilert
  - Programmet "krasjer"
- Designfeil
  - Bruk av feil formel eller fremgangsmåte. Resultatet blir feil.

Kompileringsfeil

```
class FeilType {
    public static void main(String[ ] args){
        int i = 123456;
        boolean b;
        b = i;
    }
}
```

```
$ javac FeilType.java
FeilType.java:5: incompatible types
found   : int
required: boolean
        b = i;
            ^
1 error
```

Kjøretidsfeil

```
class DivNull {
    public static void main(String[ ] args){
        int x = 7;
        int y = 0;
        int z = x / y;
    }
}
```

```
$ javac DivNull.java
$ java DivNull
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at DivNull.main(DivNull.java:5)
```

Hvordan løse oppgaver

1. **Se oppgaven utenfra:**
   1. Hva skal være inndata (input) til programmet?
   2. Hvordan skal programmet få tak i inndataene?
   3. Hva skal være utdata (output) fra programmet?
   4. Hvordan skal utdataene presenteres for brukeren?
2. **Hvordan transformere inndata til utdata?**
   1. Hvordan skal representeres (lagres)?
   2. Spesifiser en sekvens av trinn der:
      - hvert trinn gjør en enkel ting med dataene
      - hvert trinn er enkelt å programmere
3. **Skriv programkode (og test løsningen)**

Eksempel: Celsius og Fahrenheit

- Problem:
  I Norge angis vanligvis temperaturer i Celsius (C), mens man bl.a. i USA benytter Fahrenheit (F). F.eks. svarer 0 C til 32 F.

  Lag et program som lager en tabell som nedenfor (og med temperaturer i Fahrenheit fylt inn):

Celcius	Fahrenheit
-10.0	.....
0.0	.....
37.0	.....
100.0	.....

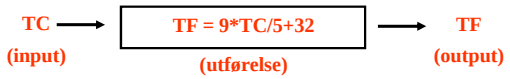
Hvilke data beskriver problemet?

- Inndata:
  - De fire Celcius-temperaturene -10, 0, 37 og 100 (desimaltall)
  - Vi tenker oss at temperaturene er gitt når vi skriver programmet. Senere skal vi se hvordan programmet kunne ha lest inndata fra terminal (fra brukeren).
- Utdata:
  - De tilsvarende (konverterte) Fahrenheit-temperaturene (desimaltall)
  - Skal skrives ut på skjermen i en tabell

Transformere inndata til utdata

- Vi må kjenne formelen for å regne om fra Celcius til Fahrenheit. La
  TC = Temperatur i Celcius
  TF = Temperatur i Fahrenheit
- Vi finner i et oppslagsverk at omregningsformelen er

  TF = 9 * TC / 5 + 32
- Dermed blir fremgangsmåten slik:



Programskisse – "Pseudokode"

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

      <sett TC lik -10>
      <regn ut TF>
      <skriv ut>

      <sett TC lik 0>
      <regn ut TF>
      <skriv ut>

      <sett TC lik 37>
      <regn ut TF>
      <skriv ut>
      <sett TC lik 100>
      <regn ut TF>
      <skriv ut>
   }
}
```

Ferdig program

```
class TemperaturKonvertering {
   public static void main (String[] args) {
        double TC, TF;
        System.out.println("Celcius Fahrenheit");

        TC = -10;
        TF = 9 * TC / 5 + 32;
        System.out.println(TC + " " + TF);

        TC = 0;
        TF = 9 * TC / 5 + 32;
        System.out.println(TC + " " + TF);

        TC = 37;
        TF = 9 * TC / 5 + 32;
        System.out.println(TC + " " + TF);

        TC = 100;
        TF = 9 * TC / 5 + 32;
        System.out.println(TC + " " + TF);
   }
}
```

Innlesning fra terminal

- Innlesning fra terminal kan gjøres på flere måter i Java. I INF1000 bruker vi pakken easyIO. Du må da skrive i toppen av programmet:

```
import easyIO.*;
```

- Inne i klassen skriver vi følgende før vi kan starte innlesning:

```
In tastatur = new In();
```

- Så kan vi lese inn fra terminal (=tastatur), f.eks. et heltall:

```
int radius;
System.out.print(”Oppgi radiusen: ”);
radius = tastatur.inInt();
```

Eksempel

Vi importerer pakken easyIO.

Vi oppretter en verktøykasse for lesing fra terminal og lager en variabel tast som blir vårt håndtak til denne verktøykassen.

I verktøykassen ligger det bl.a. en metode for å lese et heltall fra terminalen.

```
import easyIO.*;

class LesFraTerminal {
   public static void main (String [] args) {
        In tast = new In();
        System.out.print("Skriv et heltall: ");

        int k = tast.inInt();

        System.out.println("Du skrev: " + k);
   }
}
```

Resultat

```
$ javac LesFraTerminal.java
$ java LesFraTerminal
Skriv et heltall: 123
Du skrev: 123

$
```

Lesemetoder

```
// Opprette forbindelse med tastatur:
In tastatur = new In();

// Lese et heltall:
int k = tastatur.inInt();

// Lese et desimaltall:
double x = tastatur.inDouble();

// Lese et enkelt tegn:
char c = tastatur.inChar();

// Lese et enkelt ord:
String s = tastatur.inWord();

// Lese resten av linjen:
String s = tastatur.inLine();
```

Hvilken lesemetode skal jeg velge?

- Først:
  - importere easyIO og åpne forbindelse til tastaturet.
- Lese item for item:
  - For å lese et heltall: inInt() [egentlig: tastatur.inInt()]
  - For å lese et desimaltall: inDouble()
  - For å lese ett ord: inWord()
  - For å lese en hel linje (med minst ett tegn): inLine()
- Lese linje for linje:
  - Bruk readLine()
- Lese tegn for tegn:
  - For å lese neste tegn (også hvite tegn): inChar()

Hvordan lesemetodene virker

- Metodene **inInt()**, **inDouble()** og **inWord()** virker slik:
  - De hopper over eventuelle innledende blanke tegn.
  - De leser så alt fram til neste blanke tegn eller linjeskift. Dersom det som leses ikke er et heltall når inInt() brukes eller et desimaltall når inDouble() brukes, gis det en feilmelding og man får en ny sjanse (3 sjanser).
- Metoden **inChar()** virker slik:
  - Den leser neste tegn, enten det er et blankt tegn eller ikke.
- Metoden **inLine()** virker slik:
  - Den leser alt fram til slutten av linjen (inkludert blanke tegn), men ignorerer linjer hvor det kun står (igjen) et linjeskift.

Hvordan lesemetodene virker

Terminal-input: `_ _ x y z _ 1 6 1 2 7 5` `_ = blank`

`String s1 = tast.inWord();` `String s2 = tast.inWord();`	`s1: "xyz"` `s2: "161275"`
`String s1 = tast.inWord();` `int x = tast.inInt();`	`s1: "xyz"` `x : 161275`
`String s = tast.inLine();`	`s: "  xyz 161275"`

`char c1 = tast.inChar();`
`char c2 = tast.inChar();`
`char c3 = tast.inChar();`
	`c1: ' '`
	`c2: ' '`
	`c3: 'x'`

`int x = tast.inInt();`	`feilmelding`

Eksempel: lese data om en person

- Problem:
  Lag et program som leser fra terminal disse dataene om en person:
  - Navn
  - Yrke
  - Alder

  og som skriver ut dataene på skjermen etterpå.
- Framgangsmåte:
  - Vi bruker **inLine()** til å lese navn og yrke, og **inInt()** til å lese alder.

Ferdig program

```
import easyIO.*;
class LesDataOmPerson {
   public static void main (String [] args) {
        String navn, yrke;
        int alder;

        In tast = new In();
        System.out.print("Navn: ");
        navn = tast.inLine();

        System.out.print("Yrke: ");
        yrke = tast.inLine();

        System.out.print("Alder: ");
        alder = tast.inInt();

        System.out.print("Hei " + navn + ", du er " +
               alder);
        System.out.println(" år gammel og jobber som " +
               yrke);
   }
}
```

Et eksempel til

```
import easyIO.*;

class LesInnOrd {
   public static void main (String [] args) {
        In tastatur = new In();
        System.out.print("Skriv tre ord: ");
        String s1 = tastatur.inWord();
        String s2 = tastatur.inWord();
        String s3 = tastatur.inWord();

        System.out.println(
               "Du skrev disse ordene:");
        System.out.println("  " + s1);
        System.out.println("  " + s2);
        System.out.println("  " + s3);
   }
}
```

Formatert utskrift til skjerm

- Formatert utskrift vil si at vi angir nøyaktig hvordan utskriften skal se ut og plasseres på skjermen.
  - Kan gjøres "manuelt" med System.out.print(...), men det er upraktisk.
- Bedre: bruke en ferdiglaget pakke for slikt. I INF1000 bruker vi pakken easyIO. I toppen av programmet (**før** class) skriv:

  ```
  import easyIO.*;
  ```
- Inne i klassen skriver vi så:

  ```
  Out skjerm = new Out();
  ```
- Så kan vi skrive ut det vi ønsker, f.eks.:

  ```
  double pi = 3.1415926;
  skjerm.out(pi, 2, 6); // Skriv ut pi med 2 desimaler
                        // høyrejustert på 6 plasser.
  ```

Eksempel

Vi må først importere pakken easyIO.

```
import easyIO.*;
class FormatertUtskrift {
  public static void main (String [] args) {
      Out skjerm = new Out();

      int BREDDE1 = 20;
      int BREDDE2 = 30;

      skjerm.out("NAVN", BREDDE1);
      skjerm.outln("ADRESSE", BREDDE2);
      skjerm.out("Kristin Olsen", BREDDE1);
      skjerm.outln("Vassfaret 14, 0773 Oslo",
            BREDDE2);
  }
}
```

Vi oppretter en verktøykasse for skriving til terminal

I verktøykassen ligger det bl.a. verktøy (på java-språk: *metoder*) for å skrive til skjerm med og uten linjeskift til slutt.

Dukket objekter opp her?

Resultat

```
$ javac FormatertUtskrift.java
$ java FormatertUtskrift
NAVN                ADRESSE
Kristin Olsen       Vassfaret 14, 0773 Oslo
```

Tre måter å skrive ut på

- Uten formatering:

```
skjerm.out("Per Hansen");
skjerm.out(12345);
skjerm.out(3.1415, 2);
```

- Angi utskriftsbredde:

```
skjerm.out("Per Hansen", 15); // Bredde 15 tegn
skjerm.out(12345, 15);        // Bredde 15 tegn
skjerm.out(3.1415, 2, 15);    // Bredde 15 tegn,
                              // 2 desimaler
```

- Angi utskriftsbredde og justering:

```
skjerm.out("Per Hansen", 15, Out.RIGHT); // Høyrejuster
skjerm.out(12345, 15, Out.CENTER);       // Senterjuster
skjerm.out(3.1415, 2, 15, Out.LEFT);     // Venstrejuster
```