INF1000 – Uke 2

Variable, enkle datatyper og tilordning

Oversikt

- Litt repetisjon
  - Datamaskinen
  - Programmeringsspråk
  - Kompilering og kjøring av programmer
- Variabler, deklarasjoner og typer
- Tilordning
- Uttrykk
  - Verdier
  - Evaluering
- Matematiske funksjoner i Java

Repetisjon - Introduksjon

- Forstå problemer og metoder i informatikk
- **Lære å programmere**
- Mye å gjøre.
  - Både ukeoppgaver og obligatoriske oppgaver må gjøres!
- Maskiner og innlogging
- Filer

Repetisjon - Program

- Maskinen har et begrenset sett med funksjoner
  - Legge sammen tall
  - Flytte data
- Et program er en oppskrift for hva maskinen skal gjøre
- Skriver inn programmet i en fil
- Programmer som eksisterer i maskinen oversetter vårt program til instruksjoner (”maskinspråk”)
- Kompilering og kjøring

Repetisjon – Eksempel

Alt inne i klasser

Prosedyren ”main”

```
class Repetisjon {

    public static void main(String[] args){
        // Skriver ut en linje.

        System.out.println(
                "I dag snakker vi om variable og uttrykk.”
        );
    }
}
```

En kommentar

Setninger avsluttes med semikolon

En tekst eller ”String”

Kompilering og kjøring

```
class MittProgram {
    public static void main(String[] args){
        System.out.println("");
    }
}
```

Java Programtekst

Fil: MittProgram.java

$ javac MittProgram.java

Fil: MittProgram.class

```
Êþº¾??-?
???
??<init>?()V?Code?LineNumberTable?main?([Ljava/lang/String;)V?
SourceFile?MittProgram.java???
MittProgram?java/lang/Object?
??????????????????????*·?±???????????? ??
??????????<±???????
????????
????
```

Kompilert Javaprogram

Kompilering og kjøring

```
Êþº¾??-?
???
??<init>?()V?Code?LineNumberTable?main?([Ljava/lang/String;)V?
SourceFile?MittProgram.java???
MittProgram?java/lang/Object?
??????????????????????*·?±???????????? ??
??????????<±???????
????????
????
```

Kompilert Javaprogram

Fil: MittProgram.class

$ java MittProgram

Programmet kjører

Variable – Programmer og data

- Programmet bearbeider data
- Oppgaver som søk, sortering, beregning
  - Summere regning
  - Finne studenten med best gjennomsnittskarakter
  - Finne billigste flybillett
  - Regne ut hvordan været blir i morgen
- Vi må sette av plass til dataene

Variabel – En plass i lageret

- En plass i maskinens lager (minne) kan ses på som
  - en skuff i en kommode, eller som
  - en biloppstillingsplass på en parkeringsplass
- De kan ha forskjellige størrelser avhengig størrelsen på det dataelementet vi skal legge der
- Variabler må ha **navn**
  - Slik at vi kan angi i programmet vårt hvilken plass i lageret vi snakker om
- Variabler må ha **type**
  - Så vi vet hvor stor plass variabelen tar og hva det er som ligger der

Variable – Deklarasjon

- Deklarasjon
  - Angi navn og type til en variabel
  - Er en setning i programmet

```
int radius;
```

Type

Navn

Variable - Typer

- Velger en type som passer til det vi skal gjøre
- Heltall
  - Alder
  - Antall
- Flyttall
  - Vekt
  - Pris
- Sannhetsverdi
  - Er på?
  - Er tom?
- Tekster
  - Navn
  - Adresse

Variable – Primitive typer

Type	Forklaring	Eksempel
byte	heltall	byte b = 9;
short	heltall	short s = 256;
int	heltall	int i = 76234
long	heltall	long l = **12345690L**
float	desimaltall	float f = **2.5F**
double	desimaltall	double d = 3.14
char	ett tegn	char c = **'9';**
boolean	sannhetsverdi	boolean b = true;

UNIVERSITETET I OSLO

Variable - String

- Brukes ofte på samme måte som de primitive, men er ikke en primitiv type
- Skrives inne i ””
  - For eksempel: ”Fredrik Sørensen”, ”INF1000”

UNIVERSITETET I OSLO

Variable – Flere i samme setning

```
int lengde, bredde, høyde;
String navn, adresse;
```

er det samme som

```
int lengde;
int bredde;
int høyde;

String navn;
String adresse;
```

UNIVERSITETET I OSLO

Variable – Tilordning

- Variable har ingen verdi i utgangspunktet
- Verdien settes med en tilordningssetning
- En variabel kan tilordnes flere ganger
- En variabel kan tilordnes hvor som helst etter at den er deklarert

UNIVERSITETET I OSLO

Variable – Tilordning

```
// Deklarasjon
int lengde;
String navn;

// Tilordning
lengde = 34;
navn = ”Fredrik Sørensen”;

// Ny tilordning, endrer verdien
lengde = 38;

// Bruk (avlesing)
lengde = lengde + 2;
```

Betyr ”settes lik”. Det er ikke en likning.

Variable – Tilordning og avlesing

- Opprette en lagerplass i maskinen til et heltall:

```
int lengde;
int svar;
```

- Fylle plassen med en verdi (et heltall):

```
lengde = 38;
```

Tilordner verdien til variabelen

- Avlese (eller bruke) verdien:

```
svar = lengde * 2;
```

Variable – Hva skjedde der?

svar = lengde + 2;

1. **Verdien som ligger i variabelen lengde hentes fram (f.eks 7)**
2. **En ny verdi regnes ut, som beskrevet av uttrykket "lengde + 2" (f.eks 7+2=9)**
3. **Variabelen svar settes til denne nye verdien (f.eks 9)**

- **For eksempel vil lengde være 7 og svar 9 at setningen er utført**

Variable – Må tilordnes før avlesing

- En variabel som ikke er tilordnet kan ikke avleses
- Gir feil når vi forsøker å kompilere programmet

Forsøker å lese en variable som ikke er tilordnet

```
int lengde;
lengde = lengde * 2;
```

Kompilerer programmet

```
$ javac TilordningAvlesing.java
```

```
TilordningAvlesing.java:7: variable lengde might not have been initialized
      lengde = lengde + 1;
               ^
1 error
```

Vi får en feilmelding

Variable – Kombinert deklarasjon og tilordning

- **Vi kan tilordne en verdi direkte ved deklarasjonen.**

```
int lengde  = 36;

String navn  = "Fredrik Sørensen";
```

UNIVERSITETET I OSLO

Typer ved tilordning

- Ved tilordning må typen til verdien være den samme som typen til variabelen.

```
int verdi1 = 12345; // OK
int verdi2 = 2.0;   // Gir kompileringsfeil

double verdi3 = 2.0;   // OK
double verdi4 = true;  // Gir kompileringsfeil
```

UNIVERSITETET I OSLO

Variable – Regne ut areal

```
public class Areal {

   public static void main(String[] args){
      double pi = 3.14;
      double radius = 2.0;

      // Regner ut arealet og skriver ut resultatet
      double areal = pi * radius * radius;
      System.out.println("Arealet av en sirkel med radius " + radius + " er " + areal);

      // Øker arealet med 2 og gjentar oppgaven.
      radius = radius + 2;
      areal = pi * radius * radius;
      System.out.println("Arealet av en sirkel med radius " + radius + " er " + areal);
   }

}
```

UNIVERSITETET I OSLO

Variable – Bruk av String

```
System.out.println(”2” + ”3”);
// Resultat: 23
System.out.println(”2” + 3);
// Resultat: 23
System.out.println(”2 + 3”);
// Resultat: 2 + 3
System.out.println(2 + 3);
// Resultat: 5
```

UNIVERSITETET I OSLO

Variable – Flere tilordnings-operasjoner

- Det finnes flere tilordningsoperasjoner
- Forkortelser for noe vi skriver ofte

```
lengde = lengde + 2;
```

kan også skrives

```
lengde += 2;
```

UNIVERSITETET I OSLO

Variable – Flere tilordnings-operasjoner

- **Det finnes tilsvarende for de andre regneoperasjonene**

```
lengde -= 2; // Trekke fra 2

lengde *= 2; // Multiplisere med 2

lengde /= 2; // Dele på 2
```

UNIVERSITETET I OSLO

Eksempel – Bytte om variabler

- Anta at vi har disse instruksjonene:

```
int første, andre;
første = 65;
andre = 77;
```

- Hvordan kan vi bytte om verdiene i de to variablene?
- Vi forsøker dette:

```
første = andre;
andre = første;
```

- Vil dette virke?
  Se boksen til høyre

Når vi har utført …	så er verdien til:
`første = 65;` `andre = 77;`	`første: 65` `andre : 77`
`første = andre;`	`første: 77` `andre : 77`
`andre = første;`	`første: 77` `andre : 77`

UNIVERSITETET I OSLO

Eksempel – Bytte om variabler

- Problemet var at vi mistet den opprinnelige verdien til første når vi utførte

```
første = andre;
```

- Vi kan løse problemet ved å ta vare på den opprinnelige verdien i en tredje variabel. Alle instruksjonene:

```
int første, andre, hjelpevar;
første = 65;
andre = 77;
hjelpevar = første;
første = andre;
andre = hjelpevar;
```

- Vi sjekker at det virker

Når vi har utført …	så er verdien til:

`første = 65;`
`andre = 77;`
	`første: 65`
	`andre : 77`
	`hjelepevar : --`

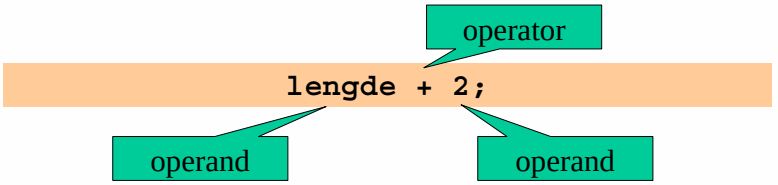
`hjelpevar = første;`
`første = andre;`
	`første: 77`
	`andre : 77`
	`hjelepevar : 65`

`andre = hjelpevar ;`	`første: 77`
	`andre : 65`
	`hjelepevar : 65`

UNIVERSITETET I OSLO

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
- For eksempel er det helt uproblematisk å konvertere et heltall til et flyttall.
- Den andre veien derimot 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!*

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!

UNIVERSITETET I OSLO

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
```

UNIVERSITETET I OSLO

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?

UNIVERSITETET I OSLO

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
```

UNIVERSITETET I OSLO

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);`

UNIVERSITETET I OSLO

Kommentarer i programmer

- For å lage programmene mer forståelige, legger vi inn kommentarer i programteksten
- Kommentarer 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 – bruk kommentarer når det er ting dere ønsker å rette oppmerksomheten mot, f.eks. sentrale punkter i programmet eller spesielt vanskelige ting.
- Det er en forutsetning at dere kommenterer programmene dere leverer som besvarelse på de obligatoriske oppgavene (oblig 2-4).

UNIVERSITETET I OSLO

Hvorfor ikke alltid bruke double?

- Mens regning med heltall alltid er eksakt, er regning med desimaltall ikke det - maskinen kan gjøre avrundingsfeil, slik som her:

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

- Verdiene til x og y er nesten like, men fordi det er en forskjell i et av desimalene langt ute blir x==y false. Slike avrundingsfeil betyr ofte veldig lite, men du kan ikke stole på at alle desimalene er korrekte når du regner med double.
- Det tar mer plass i hukommelsen å holde en double-verdi enn å holde en int-verdi.
- Det kan ta mer tid å gjøre beregninger med desimaltall enn med heltall.
- Konklusjon: når det er naturlig å bruke heltall bruker du int og når det er naturlig å bruke desimaltall bruker du double!

UNIVERSITETET I OSLO

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. Hver klasse har sitt ansvarsområde: å utføre visse oppgaver, håndtere visse typer data, eller begge deler. Dette gjør bl.a. programmene oversiktlige og gjør det lettere å bruke biter av programmet på nytt i andre sammenhenger.