INF 1000 – høsten 2011
Uke 2 – 30. september

Grunnkurs i Objektorientert Programmering

Institutt for Informatikk

Universitetet i Oslo

Siri Moe Jensen og Arne Maus

INF1000 undervisningen

- Forelesningene: Første introduksjon til ukens pensum
  - Prinsipper og hovedtrekk – hva trenger du å forstå
- Gruppeundervisning: Anvendelse (oppgaveløsning), detaljer og eksempler, gjenta/ forklare ved behov
- Eget arbeid: Lesing og *programmering* (oppgaver)

- Ekstra tilbud fra Studielaben
  - INF1000-lab fredag 14:15-18:00 på Modula med gruppelærer
    - NB: Denne uken torsdag 12:15 – 16:00 på Chill pga undervisningsfri fredag
  - Studieorakler og termvakter ved akutte problemer
  - Helpdesk for egen laptop
  - Ekstra kapasitet? Følg kurs i Sonen, oppsøk student-org

Innhold

- Et Java-program – og kommentarer
- Variable, datatyper, uttrykk
- Mer om utskrift til skjerm
- Programblokker
- Forgreninger i programmet
  - if
  - if-else
- Feil i programmer
- Hvordan løse programmeringsoppgaver

Mål for uke 2:

* Java: Variable, uttrykk og forgreninger (Kap 2 + 4.1-4.2)
* Programmering: Skrive, kompilere, teste og rette programmer

Siste program fra forrige uke…

```
class Utskrift2 {
   public static void main(String[] args) {
        System.out.println("Arne har aldri komponert en symfoni");
        System.out.println("Beethoven komponerte Skjebnesymfonien");
        System.out.println(" ----------******----------");
   }
}
```

Kompilering og kjøring:

```
>javac Utskrift2.java

>java Utskrift2
Arne har aldri komponert en symfoni
Beethoven komponerte Skjebnesymfonien
 ----------******----------
```

Kommentarer i programmene

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

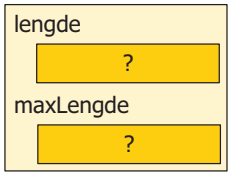
```
// Her er en kommentar som varer ut linja

/* 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!

Variabel: En plass i hukommelsen

- For å kunne huske og forandre på data, trenger programmet *variable* å lagre dem i.
- Vi reserverer plass i hukommelsen ved å *deklarere* en variabel – da oppgir vi *type* og *navn*

```
int lengde;
int maxLengde;
```



- Typen avgjør *hva slags verdi* som kan lagres i variabelen: int er en type for å lagre heltall
- Du bestemmer navnet: start med (helst liten) bokstav, deretter tall og bokstaver

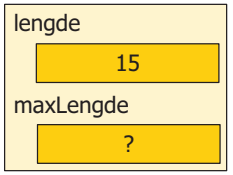
Variable: Tilordninger

- Når variabelen er deklarert, kan vi plassere en verdi av riktig type der
- Dette kalles en *tilordning*

```
lengde = 15;
```

Les:

"lengde settes lik 15"



- Kan også gjøres sammen med deklarasjonen

```
int lengde = 15;
```


Variable - avlesing

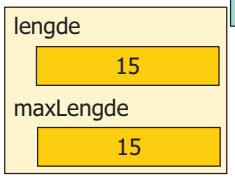
- Verdien i en variabel kan leses av og for eksempel benyttes for å gi verdi til en annen variabel ved en tilordning

```
int lengde;
int maxLengde = 15;

lengde = maxLengde;
```

Variabelen lengde..

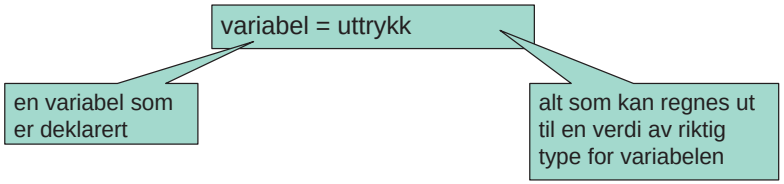
..settes lik ..

..*verdien* i variabelen maxLengde



Uttrykk

- Den generelle formen for en tilordning er



- *Uttrykket* på høyre side leses av og beregnes først – deretter plasseres resultatet som verdi i variabelen på venstre side
- Eksempler:

```
int ant = 0;
ant = ant + 1;
```


Flere datatyper i Java

Datatype	Beskrivelse	Eksempel
int	heltall	int k = 3;
double	desimaltall	double x = 3.14;
boolean	boolsk (logisk) verdi	boolean b = true;
char	tegn	char c = '@';
String	tekst	String s = "Hei på deg";

Det finnes flere (short, long, byte, ..) som du gjerne kan lese om i boken, men som ikke er sentrale å kjenne til i dette kurset.

Numeriske datatyper

- int, double, float, short og long er numeriske datatyper i Java
- Vi vil benytte int og double
- Mulige verdier i en int-variabel er heltall
  - -2
  - 0
  - 72636
- Mulige verdier i en double-variabel er desimaltall (flyttall):
  - 3.14
  - 0.00045
  - -3.72
  - 5.00000 (kan godt skrives som 5 i programmet ditt)

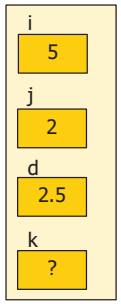
Numeriske uttrykk

- Kan stå på høyresiden av tilordning til variabel av samme numeriske type (f. eks. int eller double)
  - Numeriske *literaler* – brukes direkte i programmet
    - 3.14
    - 0
  - Numeriske *konstanter* – deklareres med fast verdi
    - final int MAX_LENGDE = 10
    - final double KORT_PI = 3.14
  - Numeriske variable
  - Operasjoner med numerisk resultat

Numeriske operasjoner

- Numeriske operasjoner med egne operatorer (symboler):
  - De 4 regneartene: + - * /
  - Inkrementering og dekrementering: ++ --
- Utført på kun heltall blir resultatet alltid heltall
- Utført på desimaltall kan resultatet bli desimaltall

```
int i = 10/2;          // gir 5
int j = 10/4;          // gir 2 (heltallsdivisjon)

double d = 10/4.0;     // gir 2.5
int k = 10/4.0;        // kan ikke lagres i int =>kompilatorfeil
```

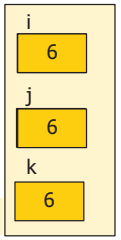


Presedens

- Operasjonene i et uttrykk utføres i en nøye bestemt rekkefølge
- Rekkefølgen bestemmes først av presendensregler, eks
  - først innholdet i parenteser, deretter
  - ++ og -- (eneste operatorer som endret variablene i et uttrykk!)
  - * og /
  - + og -
- For operasjoner med samme presedens: Fra venstre mot høyre

```
int i = 1 + 2*2;
int j = i++;        // i leses til j, deretter økes i med 1
int k = ++j;        // j økes med 1, detter leses verdien til k
```

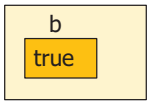
Resultat etter de 3 programsetningene:



Datatypen boolean

- I programmer har vi ofte bruk for å holde rede på om noe er sant eller usant, f.eks om (x > 0)
- Java har derfor en egen boolsk (logisk) datatype som bare har to lovlige verdier:
  - true
  - false

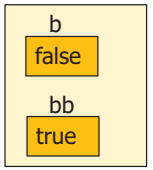
```
boolean b = true;          // Her får b verdien true
```



Boolske (logiske) uttrykk

- Logiske uttrykk gir boolsk verdi som resultat
- Logiske operasjoner:
  - (0 == 5) *tester* om 0 har samme verdi som 5!!
  - Brukes for sammenligning av to uttrykk av samme type
  - Ikke å forveksle med = som brukes for tilordning!
  - Andre logiske operatorer (eksempler senere):
    - < og >
    - <= og >=
    - !=
    - && og ||

```
boolean b = (0==5);          // Her får b verdien false
boolean bb = (0 <= 5);       // Her får bb verdien true
```



Variabeltypen String

- String-variable *peker på* en tekststreng

```
String s = "Hei på deg";
```

- To tekster kan konkateneres (spleises) ved hjelp av tegnet +

```
String t = s + ", Siri!";        // t peker nå på teksten "Hei på deg, Siri!"

System.out.println (t);          // skriver dette ut på skjermen
```

- Merk at operatoren + dermed har to ulike betydninger – avhengig av om (minst) ett av argumentene er en tekst

Utskrift fra variable

- Vi har brukt variable i tilordning av verdi til andre variable
- Verdien i en variabel kan også skrives ut på skjerm
- System.out.println () kan skrive ut både numeriske verdier, tegn og tekst-verdier

```
class Utskrift {
   public static void main (String [] args) {
            String s = "Hei! ";
            int i = 16;
            char c = '#';
            System.out.println (s);
            System.out.println (i);
            System.out.println (c);
   }
}
```

Kjøre-eksempel

```
>
>javac Utskrift.java
>java Utskrift
Hei!
16
#
>
```


Flere verdier på samme linje

- To måter å skrive ut flere verdier på samme linje:
  - Bruk System.out.print () for hver verdi untatt den siste
  - Oppgi flere verdier med + mellom i System.out.println ()

```
class Utskrift {
   public static void main (String [] args) {
            System.out.print ("En forelesningstime varer ");
            System.out.println (" 45 minutter.");
   }
}
```

Kjøre-eksempel (likt for begge)

```
>javac Utskrift.java
>java Utskrift
En forelesningstime varer 45 minutter.
>
```

```
class Utskrift {
   public static void main (String [] args) {
            int timeLen = 45;
            System.out.println ("En forelesningstime varer " + timeLen + " minutter.");
   }
}
```


Ting å passe på

- Variable kan ikke få navn med andre tegn enn tall og bokstaver, og må begynne med bokstav (vanligvis liten)
- De kan ikke hete det samme som reserverte Java-ord
- Vi kan ikke deklarere flere variable med samme navn
- Variable kan ikke leses før de er tilordnet en verdi
- Tips for å skjønne hva et program gjør: Tegn variablene, og oppdater verdier deres etter hvert som programsetningene utføres!

Blokker i programmer

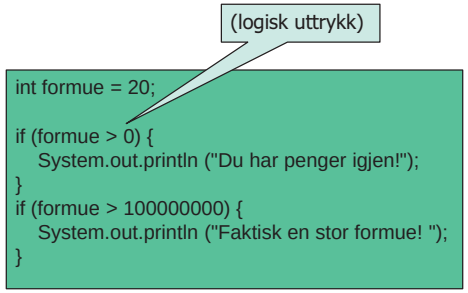
- En *programblokk* er en samling programsetninger omsluttet av krøllparenteser

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

- Blokker kan ligge inne i blokker (se over)
- En variabels *skop* er fra deklarasjonsstedet til blokken avsluttes
- Variable er ikke *definerte* (synlige) utenfor sine skop

Forgreninger: if

- Ofte vil vi at maskinen skal utføre ulike instruksjoner avhengig av for eksempel en verdi i en variabel
- Da kan vi bruke if-konstruksjonen i Java for å bestemme *under kjøring* om en eller flere setninger skal utføres



Kjøre-eksempel

```
>javac Utskrift.java
>java Utskrift
>Du har penger igjen!
>
```


Flere eksempler på tester

- Andre eksempler på logiske uttrykk (betingelser) å teste på:

```
if (!false) {
            System.out.println ("Utføres alltid")
}
```

```
if (true) {
            System.out.println ("Utføres alltid")
}
```

```
if (0 != 0) {
            System.out.println ("Utføres aldri");
}
```

```
int i = 5;
if ((i < 10) && (i > 10)) {
            System.out.println ("Utføres aldri");
}
```


Forgreninger: if - else

- Når man ønsker alternative setninger utført hvis betingelsen er usann brukes konstruksjonen *if-else*

```
int formue = 20;

if (formue > 100000000) {
   System.out.println ("Faktisk en stor formue! ");
} else if (formue > 1000000) {
   System.out.println ("Faktisk ganske romslig!");
} else {
 System.out.println ("Men ikke svært mye!");
}
```

- Slike if-else-setninger kan kobles i kjede med flere mulige utfall

Program-eksempler med feil (1)

```
class Eksempel1 {
  public static void main (String [] args) {
    boolean b;
    b = 2;            // Her prøver vi å sette b lik et heltall
  }
}
```

Feil under kompilering:

```
> javac Eksempel1.java
Eksempel1.java:4: incompatible types
found : int
required: boolean
            b = 2;     // Her prøver vi å sette b lik et heltall
                ^
1 error
>
```


Program-eksempler med feil (2)

```
class Eksempel2 {
    public static void main (String [] args) {
            int x = 3;
            int y = 0;
            int z = x / y;        // Heltallsdivisjon med null
    }
}
```

Feil under kjøring:

```
>java Eksempel2
Exception in thread "main" java.lang.ArithmeticException: / by zero
            at Eksempel2.main(Eksempel2.java:5)
```


Program-eksempler med feil (3)

```
class Eksempel3 {
    public static void main (String [] args) {
            int x = 2;
            int y = 4;

            x = y; //prøver å bytte verdier mellom x og y
            y = x; //når vi leser fra y er gammel verdi overskrevet

            System.out.println ("x=" + x + " og y=" + y);
    }
}
```

Logisk feil:

```
>java Eksempel3
x=4 og y=4
>
```


Hvordan løse oppgaver

1. Se oppgaven utenfra:
   - Hva skal være inndata (input) til programmet?
   - Hvordan skal programmet få tak i inndataene?
   - Hva skal være utdata (output) fra programmet?
   - Hvordan skal utdataene presenteres for brukeren?
2. Hvordan transformere inndata til utdata?
   - Hvordan skal dataene representeres (lagres)?
   - 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: Regn ut summen av to tall

- Inndata: I dette eksempelet bruker vi konstantene 123456 og 99999, men for større nytte ville vi latt brukeren skrive inn verdien som skal adderes
- Utdata er summen av tallene – vi må altså addere dem
- Vi ønsker å presentere resultatet sammen med de to adderte tallene på brukerens skjerm:

  Summen av <tall1> og <tall2> er <sum>

Programskisse – "Pseudokode"

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

<sett tall1 lik 123456>
<sett tall2 lik 99999>

<regn ut summen>

<skriv ut>
}
}
```

- Tegn gjerne variablene og følg med på verdiene de får når du utfører handlingene en etter en – simuler maskinen!

Ferdig program

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

            int tall1, tall2, sum;

            tall1 = 123456;
            tall2 = 99999;

            sum = tall1 + tall2;

            System.out.println ("Summen av " + tall1 + " og " + tall2 + " er: " + sum);

   }
}
```

- NB: Test programmet ved å kjøre det!

Oppgave

- Lag et program der du deklarerer og gir verdi til en heltallsvariabel og en logisk variabel. Eksperimenter med ulike verdier i disse når programmet er ferdig skrevet.
- Programmet skal teste verdien i heltallsvariabelen, og skrive ut riktig ukedag hvis tallet er mellom 1 og 7 (1= mandag etc).
- Hvis tallet er under 1 eller over 7, skrives det ut en beskjed om dette.
- Hvis tallet er 3 skal det i tillegg testes om verdien i den logiske variabelen er true – i så fall skrives setningen "En ekstra bra onsdag" ut i tillegg til bare ukedagen.

➢ Her får du testet if-else i flere ledd, samt en if-setning inni en annen.