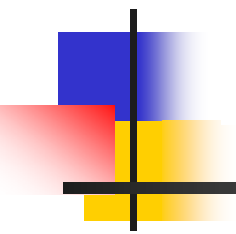


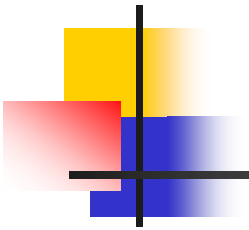
INF 1000 (uke 2)
Variabler, tilordninger og uttrykk

Grunnkurs i programmering

Institutt for Informatikk

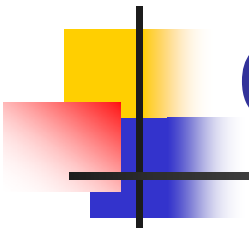
Universitet i Oslo

Anja Bråthen Kristoffersen og Are Magnus Bruaset



I dag skal vi se på...

- Flere praktiske opplysninger
- Litt repetisjon
- Hva er en variabel i et program?
  - Deklarasjoner og variabeltyper
  - Tilordning
- Flere detaljer i Java-språket



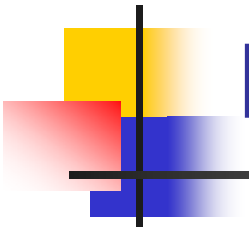
Gruppene starter denne uken!

- Hjemmesida
  http://www.ifi.uio.no/inf1000/v06
  - Sjekk tider for din gruppe
  - Ukeoppgaver (fra bok + ekstra)

Klart for første oblig

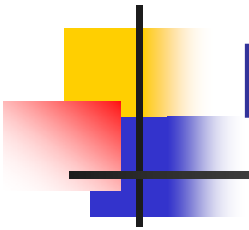
- Tilgjengelig på hjemmesida
- Frist 3. februar kl. 16.00

- Alle obligene er lagt ut på hjemmesida.
  - På hver oblig står det når du har lært nok for å løse obligen samt innleveringsdato.



Rep: Programmering

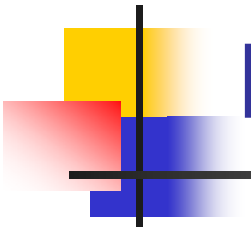
- Et **program** er en **rekkefølge av enkle ordrer**
- Ordrene utføres av datamaskinen - en for en, fra første til siste ordre
- Vi må spesifisere ordrene i et **programmeringsspråk**



Rep: Java

- Java-programmer skrives i Java-språket
- Streng, men enkel grammatikk
  - Må følges 100%
- Må oversettes til maskinlesbar kode
  - Gjøres ved å kompilere java-programmet

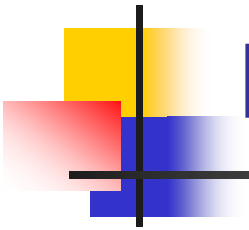
```
class Utskrift {
	public static void main(String[] args) {
		System.out.println("Beethoven komponerte Skjebnesymfonien");
	}
}
```



Rep: Java-programmer

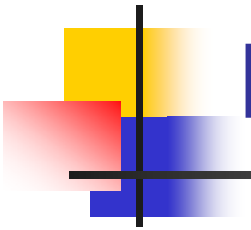
- Java-kode skrives inn i en editor (Emacs)
- Java-kode (”.java”-fil) oversettes av kompilatoren (”javac”) til maskinlesbar kode (”.class”-fil)
- Kompilert kode (i ”.class”-fil) kjøres av kjøreprogrammet (”java”)

Prosessen (editering – kompilering – kjøring) er stort sett den samme for alle språk!



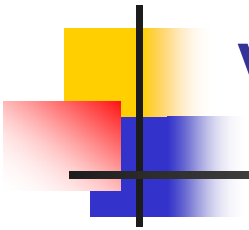
Nytt stoff

- Variabler
- Tilordninger
- Uttrykk



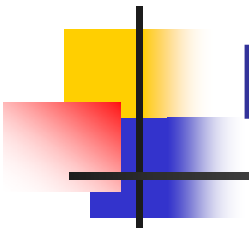
Data

- Alle problemer innholder **data** som beskriver problemet
  - Søking i Ibsens skuespill?
    - Trenger å lagre en rekke **tekster** i hovedhukommelsen
  - Lage værmelding?
    - Trenger å lagre en rekke **tall** i hovedhukommelsen (bl.a data fra vær-observasjonene)
- Slike data (tall, tekster,..) må vi ha i programmet vårt
- Setter av egne, navngitte ”plasser” i programmet



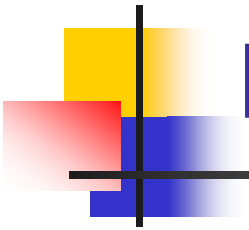
Variabler

- Disse ”plassene” kalles **variabler** og har tre egenskaper:
  - Et **navn** (entydig)
  - En **type** (hva slags data kan variabelen lagre? F.eks. en tekst, et heltall, et desimaltall,...)
  - Et **data-innhold,** en **verdi** som variabelen skal lagre (ikke entydig, kan endres senere i programmet)



Program = Data + Handlinger

- Setninger i et program er av tre typer:
  - Sett av plass til data
  - Gjør noe med data (regne, skrive ut,...)
  - Kommentarer som gjør det lettere å lese programmet

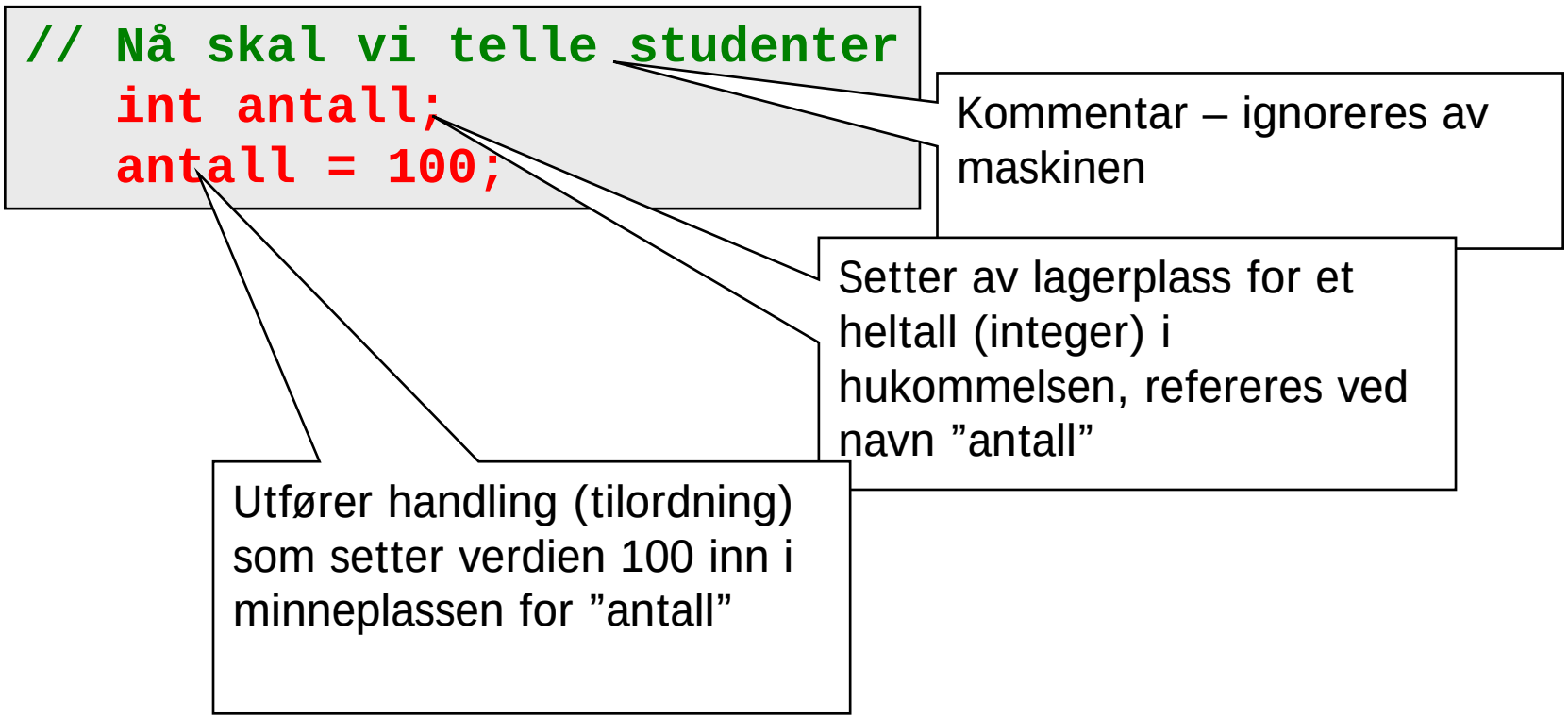


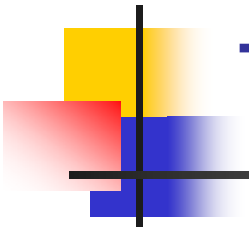
Kommentarer

- Ignoreres av oversetter og kjøreprogram (utføres ikke)
- I Java:
  - Linjekommentarer startet med to skråstreker `//`
    - Alt **etter** `//` på en linje blir kommentaren
  - Avsnittskommentar starter med `/*` og slutter med `*/`.
    - Alt **mellom** `/*` og `*/` blir kommentaren uansett hvor mange linjer kommentaren er.

Data + handlinger (+ kommentarer)

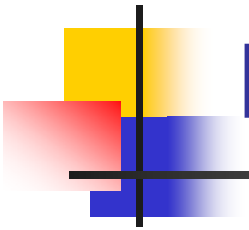
- Eksempel (del av et program):





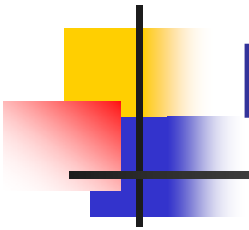
To enkle programmer

- To eksempler på mer eller mindre enkle Java-programmer
  - ✓ Renteberegning
  - ✓ Forslag til fornavn
- Mer om reglene for å lage (enkle) Java program



Programvareutvikling - oversikt

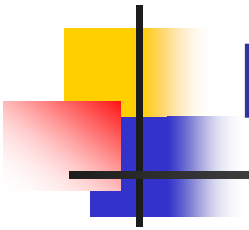
1. Først har vi et problem vi skal løse (en oppgave)
2. Finn en fremgangsmåte (=algoritme) for problemet
3. Hvilke data beskriver problemet / algoritmen?
4. Skriv et Java-program, syntaktisk korrekt
   - Kompiler og rett opp eventuelle feilmeldinger
5. Test ut programmet, sjekk at det gir riktig svar
   - Gjøres ved å kjøre programmet på små eksempler hvor vi vet svaret, programmet kjøres først når det er ferdig testet på de ekte dataene.



Problem 1: Renteberegning

1. Du setter 120 000 kr. i banken. Du får 6.25% rente første året, og 7 % rente andre året.

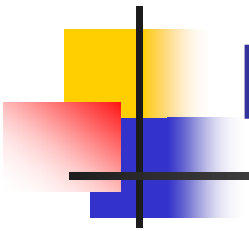
   Skriv ut hvor mye du har i banken etter første og andre året



Renteberegning fortsatt

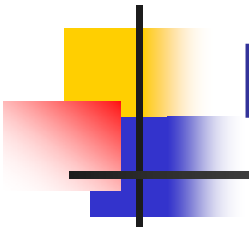
Hvordan løse det:

2. Kan vi en formel for renteberegning?
3. Hvilke data beskriver problemet ?
4. Skrive kode
5. Test



Problemanalyse

- Renteberegningsformel:
  - ny kapital = gammel kapital * (1+rente)
- Hvilke data har vi i problemet ?
  - Innskudd, dvs gammel kapital
  - Rentesats for 1. og 2. år



Har forståelsen, trenger program

Program:

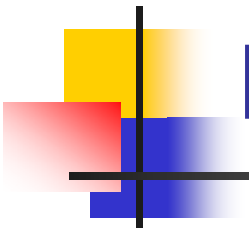
1. Deklarere data
2. Skrive handlingssetninger, inkludert utskrift
3. Skrive kommentarer i koden

Rente-programmet, fyll ut

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

      …

  }
}
```

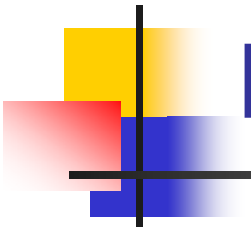
Program 1 – Rente-programmet

```
public class Rente
{
  public static void  main ( String[] args) {
    double      rente;
    double      kapital = 120000;

    // rente første år
    rente = 6.25/100;
    kapital = kapital + kapital*rente;
    System.out.println(”Kapital etter første år: " + kapital);

    // rente andre år
    rente = 7.0/100;
    kapital = kapital + kapital*rente;
    System.out.println(”Kapital etter andre år: " + kapital);
  }
}
```



Kompilering og testing

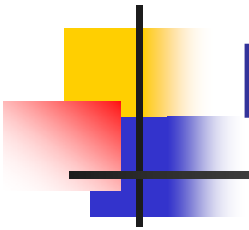
```
>javac Rente.java

>java Rente
Kapital etter første år: 127500.0
Kapital etter andre år: 136425.0
```



Problem 2: Navneforslag

- Herr Jensen er nybakt far og vil at junior skal ha tre fornavn: ”Kai”, ”Ole” og ”Georg”.

  Lag en nummerert liste over alle mulige navn-kombinasjoner av disse slik at herr Jensen kan velge den beste

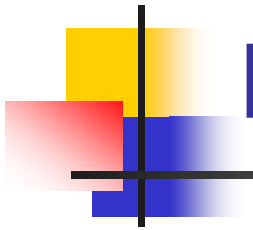


Problemanalyse

- Har vi her en metode?
- Hvor mange muligheter er det?
- Hvilke data trenger vi?
  - Navnene?

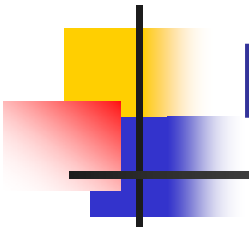
Program 2 - Navn

```
public class Navn
{
  public static void  main ( String[] args) {
    String k = ”Kai”;
    String o = ”Ole”;
    String g = ”Georg”;

    System.out.println(” ”);

      …

  }
}
```



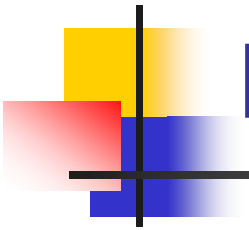
Program 2 - Navn

```
public class Navn
{
  public static void  main ( String[] args)  {
    String k = ”Kai”;
    String o = ”Ole”;
    String g = ”Georg”;

    // det er 6 mulige kombinasjoner av 3 navn

    System.out.println(”1. ” + k + ” ” + o + ” ”+ g + ” Jensen”);
    System.out.println(”2. ” + k + ” ” + g + ” ”+ o + ” Jensen”);
    System.out.println(”3. ” + o + ” ” + k + ” ”+ g + ” Jensen”);
    System.out.println(”4. ” + o + ” ” + g + ” ”+ k + ” Jensen”);
    System.out.println(”5. ” + g + ” ” + o + ” ”+ k + ” Jensen”);
    System.out.println(”6. ” + g + ” ” + k + ” ”+ o + ” Jensen”);
  }
}
```



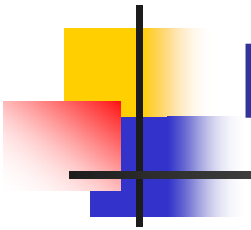
Kompilering og testing

```
>javac Navn.java

>java Navn
1. Kai Ole Georg Jensen
2. Kai Georg Ole Jensen
3. Ole Kai Georg Jensen
4. Ole Georg Kai Jensen
5. Georg Ole Kai Jensen
6. Georg Kai Ole Jensen
```



Litt mer Java

- Deklarasjoner
  - **`double`**
  - **`int`**
- Handlingssetninger, tilordning
- Utskrifts-setningen
  - **`System.out.println( ..... );`**



Heltall og desimaltall (flyttall)

- Datamaskiner håndterer disse på forskjellig måte
- Heltall er alltid eksakte, mens desimaltall har bare en viss nøyaktighet

Deklarasjoner – double, int

- Desimaltall deklareres med

  `double`

  - Eksempel:

    `double høyde, volum;`

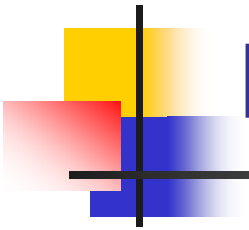
- Heltall deklareres med

  `int`

  - Eksempel:

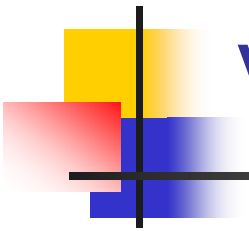
    `int antallBiler;`

    `int antallFuglereder;`



Deklarasjoner - double, int

- En deklarasjon lager en **variabel**
  - Setter av **plass** i hurtiglageret
  - I den plassen kan vi lagre verdier av den deklarerte **typen** (dvs. int, double,...)
  - Variabelen (lagerplassen) får det **navnet** vi gir den i deklarasjonen. Navnet er entydig.

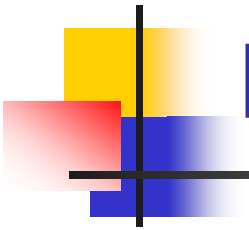


Variabler - konvensjoner

- Variabler har liten bokstav først i navnet
- Består navnet av flere ord skrives det slik:

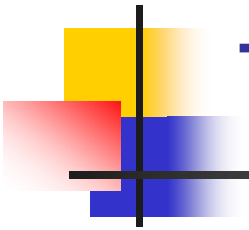
```
int antKunder, antallBokserMaling;
```

- Når en variabel først er deklarert, kan den brukes mange ganger, og verdien kan endres mange ganger



Handlingssetninger, tilordning

- Tilordningsetninger (v.s. = h.s.;) gir en ny verdi til en variabel
- På venstre side (v.s.) står navnet til en variabel
- Tegnet **=** leses: *”settes lik”*
- På høyre side (h.s.) står et regnestykke, verdien regnes ut



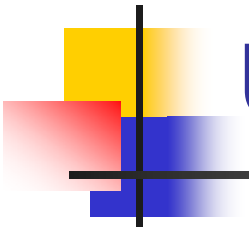
Tilordning, fortsatt

- Forekommer det navn på variabler i regnestykket, bruker vi verdien av disse i utregningen
- Eksempel:

```
int i, j, k = 2;
i = 14;
j = i + 22;
j =  j + 1;
i = i + k;
j = 10* j + i;
```

Spesialnotasjon: ofte brukte operasjoner

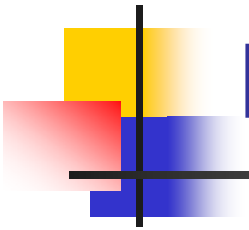
```
j++;      //det samme som j = j+1;
j--;      //det samme som j = j-1;
j *= 22; //det samme som j = j*22;
j += a;  //det samme som j = j+a;
j -= 14; //det samme som j = j-14;
  ................
(flere etter samme mønster)
```



Utskrifts-setningen

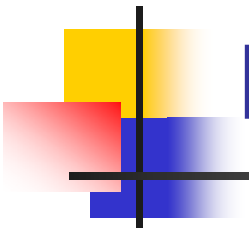
```
System.out.println( ”Litt utskrift:” + i  + ”    ” + s + dd );
```

- Det som står inne i parentesen skrives ut på skjermen
- Flere ledd kan skjøtes sammen med +



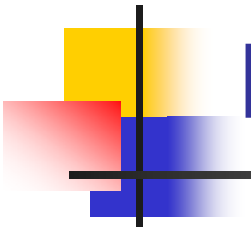
Mer utskrift

- Det som skjøtes sammen kan være:
  - **`String`**-konstanter eks: ”Renta er”, ”Forslag til navn:”
  - **`String`** deklarasjoner
    - Eksempel: **`String s = ”Simen”;`**
  - **`int`** (heltall) – blir da omgjort til **`String`**
    - **`int i`** har fått verdien 1023. Da blir **`i`** omgjort til en **`String`** bestående av sifrene ’1’,’0’,’2’,’3’
  - **`double`** (desimaltall) blir omgjordt til **`String`** med sifre før og etter punktum (evt. til en **`String`** med eksponent hvis tallet er for stort eller lite)
  - Eksempel: **`double dd = 0.2345;`**



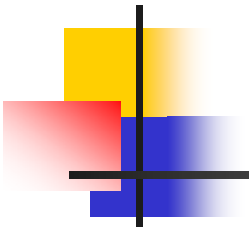
Flere Java-detaljer

- Alle de 50 ordene i Java (reserverte ord)
- Konstanter
- Uttrykk
- Logiske uttrykk
- Pakker og Java klassebibliotek
- Tre typer feil



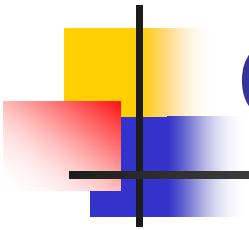
De 50 reserverte ordene i Java

`abstract`	`default`	`goto`	`new`	`synchronized`
`boolean`	`do`	`if`	`package`	`this`
`break`	`double`	`implements`	`private`	`throw`
`byte`	`else`	`import`	`protected`	`throws`
`case`	`extends`	`instanceof`	`public`	`transient`
`catch`	`false`	`int`	`return`	`true`
`char`	`final`	`interface`	`short`	`try`
`class`	`finally`	`long`	`static`	`void`
`const`	`float`	`native`	`super`	`volatile`
`continue`	`for`	`null`	`switch`	`while`



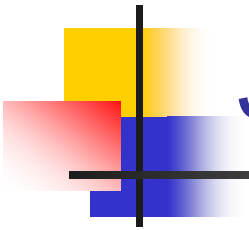
Addisjon (+) har mange virkemåter

- Mellom to heltall gir ”+” en heltallsaddisjon,
  `2 + 2`
- Mellom to flytende tall gir ”+” en desimaltallsaddisjon,
  `2.5 + 3.14`
- Mellom et heltall og et flytende tall gir ”+” en desimaltalls-addisjon: **`2 + 3.14`**
- Mellom to tekststrenger gir ”+” en sammenskjøting (konkatenering) av strengene
  `”Hallo ” + ”verden”`



Overlasting av operatorer

At en operator har ulik virkning avhengig av operandene, kalles **overlasting** (av operatoren)

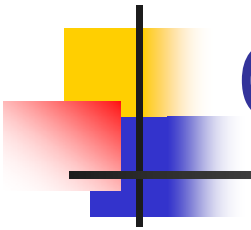


Java er et sterkt typet språk

- Variablene deklareres til å lagre verdier av en bestemt type
  - Eksempel: `int sum;`
- Oversetteren kontrollerer
  - At verdiene som inngår i regnestykker er av riktig type
  - At typen av uttrykket på høyre side av en tilordning stemmer med typen av variabelen på venstre side
  - Eksempel:

```
int sum;

sum = 3.14 + 0.2;
```

gir syntaksfeil



Casting

- Konvertering (”casting”) av en verdi fra en type til en annen er av og til mulig (detaljer senere)
  - Implisitt:

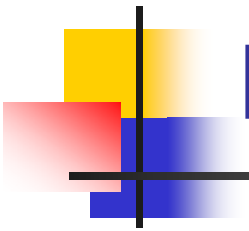
```
double ti;
ti = 10;
```

  - Eksplisitt:

```
int sum;
sum = (int) 3.14;
```

Deklarasjoner og variabeltyper

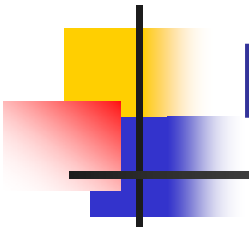
Type	Forklaring	Eksempel	Lovlige verdier
byte	heltall	`byte b = 101;`	-128 til 127
short	heltall	`short s = -10002;`	-32768 til 32767
int	heltall	`int i = 0;`	-2147483648 til 2147483647
long	heltall	`long l = 2000000L;`	-9223372036854775808L til 9223372036854775807L
float	desimaltall	`float f = 0.25F;`	ca ±3.40282347 E+38F (6 til 7 signifikante desimaler)
double	desimaltall	`double d = 3.14;`	ca ±1.7976931348623157 0 E+308 (15 signifikante desimaler
char	tegn	`char c =’a’;`	alle ASCII tegn
boolean	sannhetsverdi	`boolean b = true;`	true, false



Navn på variable

- Alltid liten forbokstav
- Alltid kun et ord
- Må begynne på en bokstav
- Kan så inneholde bokstaver, tall, ”_”
- Bruk stor bokstav ved ny stavelse, eksempel:

```
int antallDagerIUka = 7;
double x1, x2, sirkelAreal;
```



Konstanter

```
final int DUSIN = 12;
final int SNES = 20;
final int GROSS = 144;
final int MAX_ELEVER_PR_KLASSE = 20;
```

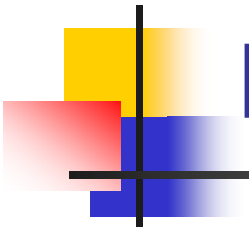
- Kan ikke endres etter at de har fått første verdi pga. Java-ordet **final**
- Bruker bare store bokstaver (for å vise at dette er konstanter)

Evaluering av numeriske uttrykk

- Eksempel:

```
double x = 3 * (a+1.3)/4+ 4 – a+b/c;
```

- Regnes ut fra venstre mot høyre, men:
  - Først metodekall
  - Så regnes all parenteser ut
  - Så alle ++ og – –
  - Så * og /
  - Så + og –



Parenteser

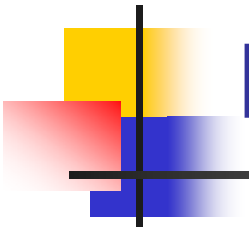
- Bruk heller parenteser hvis det er vanskelig

```
double x = 3 *( (a+1.3)/4 )– a +(b/c);
```

Logiske variabler og uttrykk

```
boolean c, b;
b = i < 5;            // i er mindre enn 5
c = (j != 5);         // j inneholder ikke verdien 5;

System.out.println("Er i større enn 5" + b);
```

Operator	Beskrivelse	Eksempel
&&	Og (sann hvis begge ledd sanne)	`b = (x<y) && (y<z);`
||	Eller (sann hvis minst ett ledd er sant)	`b = (x<y) || (y<z);`
!	Ikke (snur sannhetsverdien)	`b = !(x<y);`
< og >	Mindre enn, større enn	`b = x < y;`
<= og >=	Mindre enn eller lik, større enn eller lik	`b = x <= y;`
==	Er lik	`b = (x==y);`
!=	Er ikke lik	`b = (x != y);`

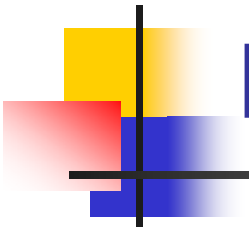


Pakker og Java-klassebibliotek

```
double d = Math.sqrt(23);
```

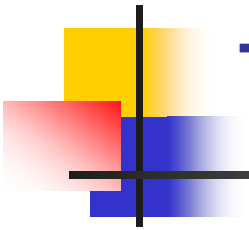
- I et eget Java-bibliotek er det over 2000 ferdigprogrammerte klasser
- Med unntak av en samling av mye brukte klasser, (pakken ”java.lang”), må vi importere en slik samling av klasser hvis vi vil bruke noen av dem

Pakker og Java-klassebibliotek



- Vi har hittil brukt bibliotekskklassene : **`Math`** og **`System`** som begge er i **`"java.lang"`**
- Skal vi bruke noen andre klasser i andre pakker må disse importeres.
- I toppen av programmet må vi da ha en import-setning:
  - Eksempel:

```
import easyIO.*;
```



Tre muligheter for feil

1. Syntaks-feil
   - Vi skriver programmet galt, grammatikk-feil
     - Finnes av kompilatoren, javac
2. Algoritme-feil
   - Vi har brukt gal løsningsmetode. Programmet løser ikke problemet
     - Kan finnes ved å konstruere små eksempler hvor løsningen er kjent. Kjør programmet på disse.
3. Kjøre-feil
   - Programmet stopper under utførelse

Feil nr. 1 – hvilken ?

```
// Program  med FEIL !
public class EnkeltProgram1
{
  public static void  main ( String[] args) {
    int     radius;
    double  areal;

    radius = 4;
    areal = 3.14 * radius * radius

    System.out.println("Sirkel, radius: " + radius
      + ", areal: " + areal);
  }
}
```

Oversetterens svar på feil nr 1

```
>

>javac EnkeltProgram1.java
EnkeltProgram1.java:10: ';' expected
^
1 error

>
```

Feil nr 2 – hvilken ?

```
//  Program  med FEIL !

public class EnkeltProgram2
{
  public static void  main ( String[] args) {
    int     radius;
    double  areal ;

    radius = 4;
    areal = 3.14 * radius;

    System.out.println("Sirkel, radius: " + radius
      + " , areal: " + areal);
  }
}
```

Kompilerering og test av EnkeltProgram2

```
>javac  EnkeltProgram2.java
>java  EnkeltProgram2
Sirkel, radius: 4, areal: 12.56
>
```

Feil i formelen for areal av sirkel, skal være

areal = 3.14*radius*radius

Feil nr. 3 – hvilke(n) ?

```
//  Program  med FEIL !
public class EnkeltProgram3
{
  public static void  main ( String[] args) {
    int   radius = 0;
    double  areal ;

    areal =  radius/radius*3.14;

    System.out.println("Sirkel, radius: " + radius
      + ", areal: " + areal);
  }
}
```

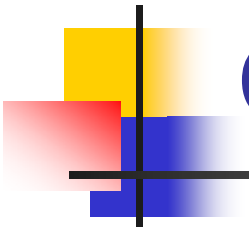
Feil 3

```
>javac Enkeltprogram3.java

>java EnkeltProgram3
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at EnkeltProgram3.main(Enkeltprogram3.java:8)

>
```

Feil i formelen for areal av sirkel, skal være

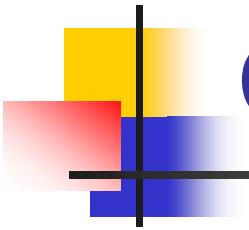
areal = 3.14*radius*radius

Divisjon med null, udefinert operasjon!



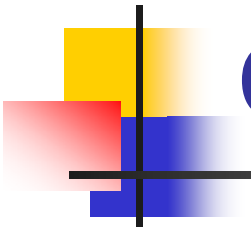
Oppsummering - variabler

- En variabel deklareres i programmet, og vi får da en plass i lageret med
  - Et navn
  - En type
  - En verdi



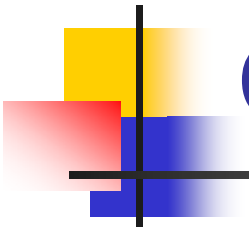
Oppsummering - variabelnavn

- Navn på variabler bør begynne med liten bokstav
- Stor bokstav for hvert ord i variabelnavnet (**`antallBarn`**)
- Variabelnavnet kan ikke være likt noen av de reserverte ordene i Java-språket



Oppsummering - tilordning

- Eksempel: **`i = j*3 +i ;`**
- Venstre side (**`i`**) er navnet på en variabel som skal få ny verdi
- Høyresiden **`(j*3 +i)`** er et regnestykke
  - Variabelnavn (som **`j`** og **`i`**) erstattes med deres nåværende verdi



Oppsummering - uttrykk

- Tre typer uttrykk som ofte forekommer:
  - Numeriske uttrykk
  - Logiske uttrykk
  - Streng-uttrykk