PRG1000B Grunnleggende programmering

Leksjon 2

Setninger og uttrykk

Del 2

Roy M. Istad, 2017

Lage egne navn i Java

Hva skal navnsettes?

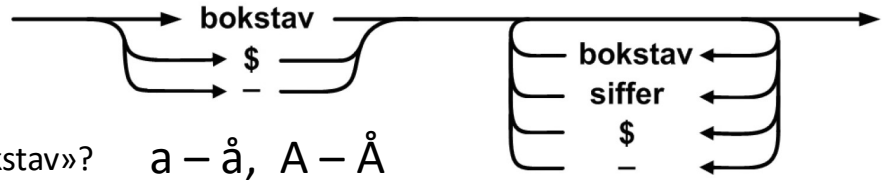
- Program (class)
- Metoder (main)
- Variabler og konstanter
  (konstant: "låst" variabel)
- Klasser (egne datatyper) og objekt (variabel av klasse-type)

Anbefalinger

- Bruk meningsfylte navn (både lettere å lese og forstå koden)
- Variabler og metoder starter med en liten bokstav
- Flere ord sammensatt til ett ved at påfølgende ord starter med stor bokstav
- Konstanter skrives med store bokstaver (med lav bindestrek som ordskiller)
- Klassenavn (program) starter med stor bokstav

Lage egne navn (ord) i Java

Syntaks for et lovlig Java-navn:



Hva menes med «bokstav»? a – å, A – Å

Hva menes med «siffer»? 0 – 9

Lovlige navn? `y` `etToTall` `MAX_VALUE` `år1`

Ulovlige navn? `2tall` `Q&A` `Ida-Marie` `Nei!`

Språkanbefaling: `enbil` → `enBil`, `mittprogram` → `MittProgram`
(*variabel*) (*klasse*)

Lage egne navn i Java

Variabeldeklarasjoner:

```
String fornavn;    // Alternativt (samtidig):
String etternavn;  // String fornavn, etternavn;
int antallBarn, antallEpler;
double mva = 0.25;
double nkrBeløp, usDollar;
```

Klassedeklarasjon (programnavn):

```
public class Kassalapp {
  // Skal ligge på filen: Kassalapp.java
}
```

Lage egne navn i Java

Konstant: Låst variabel – innsatt verdi kan ikke endres (`final`)

Konstantdeklarasjoner:

```
final double MVA = 0.25;      // momsen er nå 25%
final int MIN_ALDER = 18;     // aldersgrense for å kjøre bil
final char SKILLETEGN = '#'; // final: modifier/modifikator
```

Java-biblioteket har også konstanter, vi kan bruke verdier via navn, f.eks.

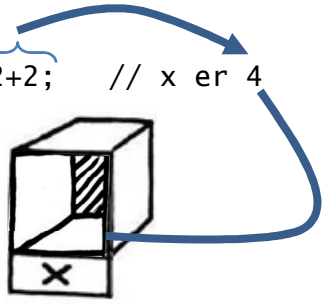
```
Math.PI // Slipper: 3.14159265…

Integer.MAX_VALUE
// Største heltallsverdi
```

Hvorfor bruke konstanter?

- gjør koden mer *lesbar*
- reduserer mulighetene for feil
- gjør program lettere å *endre* og lettere å vedlikeholde.

Uttrykk og verdier

```
int x = 2+2;    // x er 4
```



Anonym konstant:
Konkret verdi i koden.
Her heltallene 2, 3, 4 og 7

```
int y = x+3;    // x er 4, y er 7
```

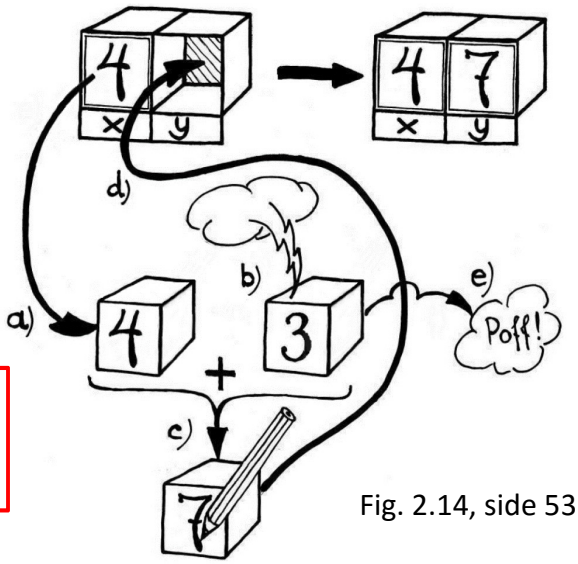
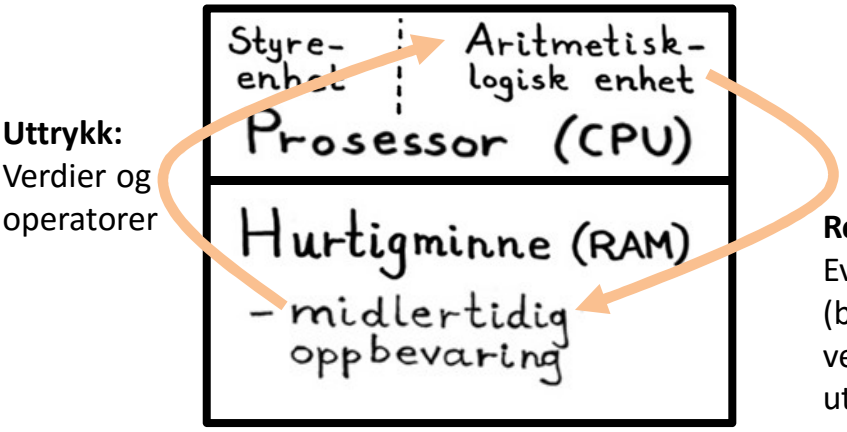


Fig. 2.14, side 53

Maskinkjerne - prinsippskisse



Uttrykk:
Verdier og operatorer

Resultat:
Evaluert (beregnet) verdi av uttrykk

Uttrykk og verdier

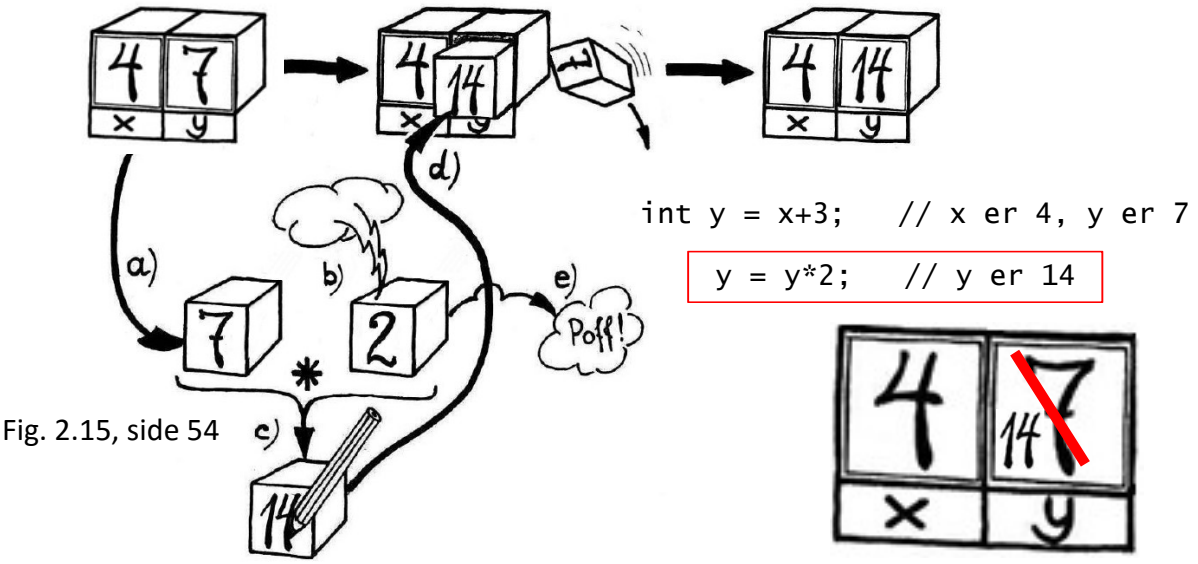


Fig. 2.15, side 54

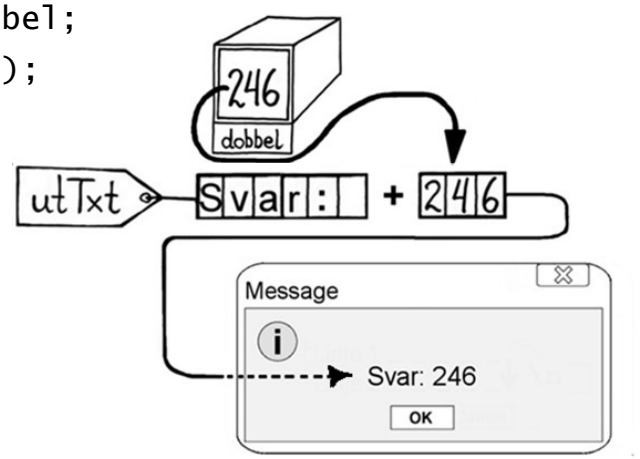
```
int y = x+3;   // x er 4, y er 7
y = y*2;   // y er 14
```

Utskrift via dialogvindu og `String`

```
String utTxt = "Svar: " + dobbel;
showMessageDialog(null, utTxt);
```

Teksten får et tillegg (skøytet på). Ikke + for addisjon mellom tall!

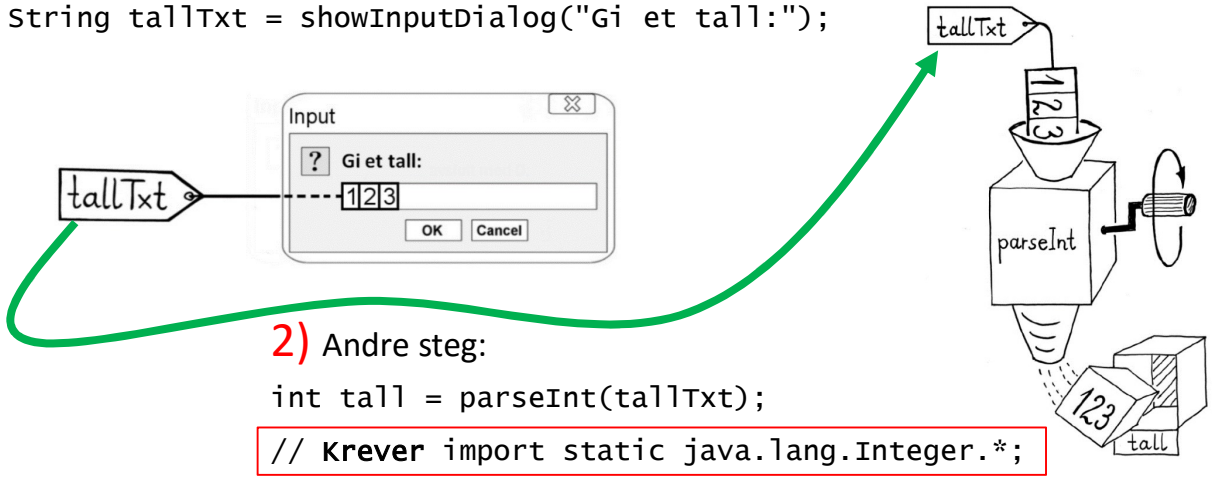
Verdien i variabelen dobbel blir automatisk omgjort (*konvertert*) til en tegnsekvens (ny String) før den skøytes på ledeteksten "Svar".



Innlesing via dialogvindu og `String`

1) Første steg:

```
String tallTxt = showInputDialog("Gi et tall:");
```



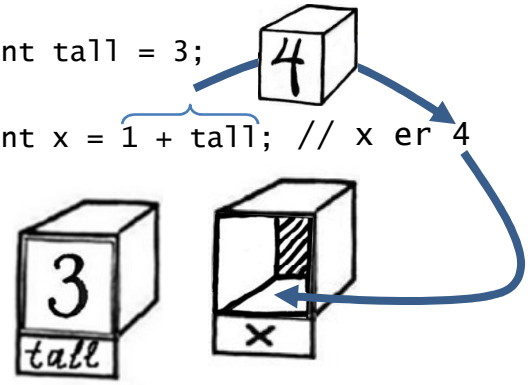
2) Andre steg:

```
int tall = parseInt(tallTxt);
```

```
// Krever import static java.lang.Integer.*;
```

Uttrykk, operatorer og verdier

```
int tall = 3;

int x = 1 + tall; // x er 4
```



Anonym konstant:
Konkret verdi i koden, her heltallene 1 og 3.

Uttrykk:
Variabler, verdier, konstanter og metodekall sammensatt med operatorer.

Verdier:
Anonyme konstanter, *evaluerte* uttrykk og metodekall.

Operatorer:
Spesielle symbol → utfører operasjoner (oppgaver) på/mellom verdier og uttrykk.

Eks. + – * /

Uttrykk, operatorer og verdier

Eksempel:

1) 12 + 6 / 3 – 1 = 12 + 2 – 1 = 13

2) (12 + 6) / 3 – 1 = 18 / 3 – 1 = 5

3) (12 + 6) / (3 – 1) = 18 / 2 = 9

Vi bruker parenteser for å styre/overstyre beregningsrekkefølgen.

NB!
Operatorene har ulik prioritet, de utføres i en bestemt rekkefølge.

F.eks. * og / utføres foran + og –

Primitive datatyper

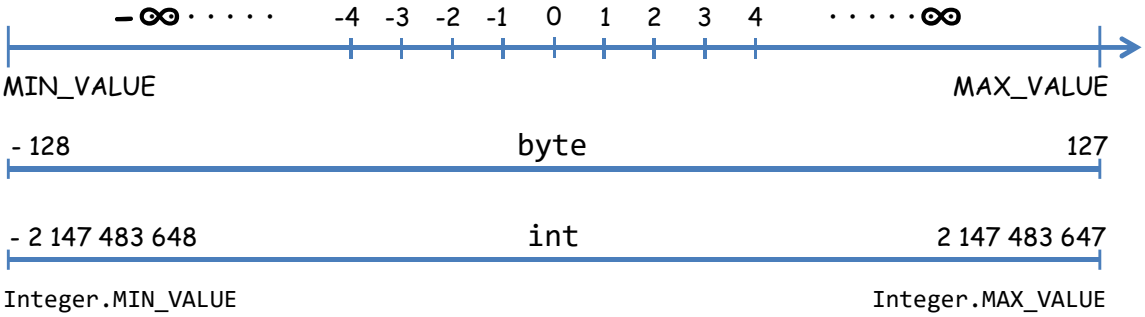
Datatype	Navn	Plass (bit)	Min.verdi	Maks.verdi
Enkelttegn	`char`	16	0	65535
Heltall	`int`	32	-2^{31}	$2^{31}-1$
Desimaltall	`double`	64	$4.9 \cdot 10^{-324}$	$1.8 \cdot 10^{+308}$
Logiske verdier	`boolean`	1	`false`	`true`

Primitive datatyper

Datatype	Navn	Plass (bit)	Min.verdi	Maks.verdi
Enkelttegn	`char`	16	0	65535
Heltall	`byte`	8	-128	127
Heltall	`short`	16	-32768	32767
Heltall	`int`	32	-2^{31}	$2^{31}-1$
Heltall	`long`	64	-2^{63}	$2^{63}-1$
Desimaltall	`float`	32	$1.4 \cdot 10^{-45}$	$3.4 \cdot 10^{+38}$
Desimaltall	`double`	64	$4.9 \cdot 10^{-324}$	$1.8 \cdot 10^{+308}$
Logiske verdier	`boolean`	1	`false`	`true`

Primitive datatyper

Heltallstype



Hva blir 1 500 000 000 + 1 500 000 000 i datatypen `int`?

Primitive datatyper

Heltallstype

`int`: Adderer tall til stadig større verdi . . .

integer overflow

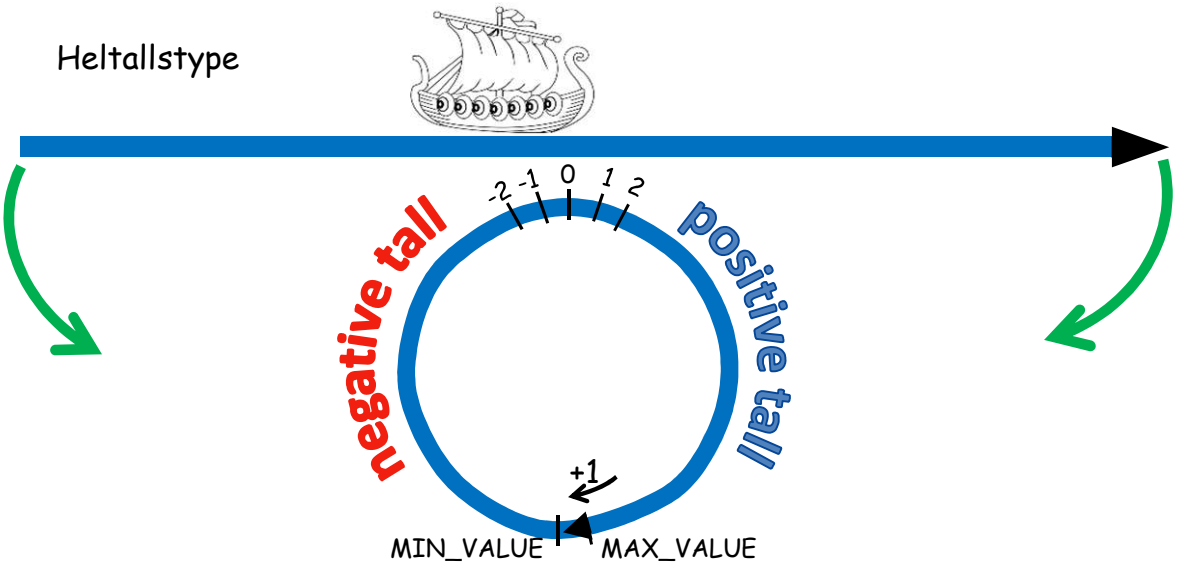
Eks. 1 000 000 000
\+ 1 000 000 000
\+ 1 000 000 000
= –1 294 967 296 ?

< 2 147 483 647
= MAX_VALUE

2 000 000 000 + 2 000 000 000 + 1 000 000 000 = 705 032 704 ?

Primitive datatyper

Heltallstype



Typeblanding og typetvang

Automatisk typeskifte:

```
int x = 5;
double y = x;  // 5 er et heltall!
out.println("y = " + y);  // gir ...?

// Hva skjer omvendt?
double z = 5;
int w = z;    // 5.0 er et desimaltall
out.println("w = " + w);  // gir ...?
```

`y = 5.0`

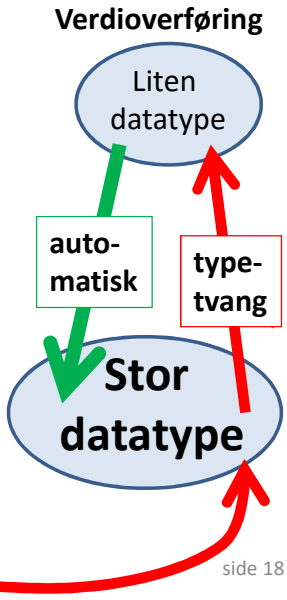
...incompatible types: possible lossy conversion...

Typetvang:

Datatype til beregningsresultat overstyres aktivt

```
double desimal = 2.8;
int hel = (int)desimal;
```

`(datatype)uttrykk`



Typeblanding: Datatypen til beregningsresultatet settes automatisk.

```
int    \
double  } { + - * / } { double \
                      \ int    } ⟶ double
```

```
int { + - * / } int ⟶ int
```

Hvordan skal vi få 2,5 eple pr barn som resultat?

Hvilken datatype må/bør variablene ha?

```
double lengde   =  3.5 + 6;
double brøk     =  12.9 / 3;
   int ePlePrBarn  =  10 / 4;
```

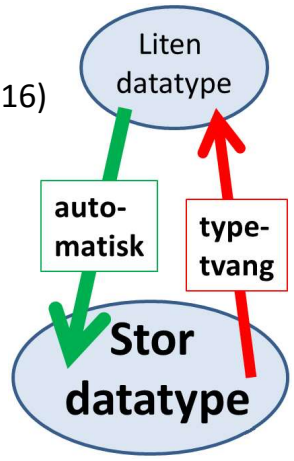
NB! Datatypen blir `double` når:

```
ePlePrBarn = 10/4.0;
ePlePrBarn = 10.0/4;
ePlePrBarn = (double)10/4;
```


Enkelttegn, `char` og Unicode

- Datatypen `char` brukes for å ta vare på ett enkelt tegn, f.eks. `'a'`
- Alle tegn har en tallkode i Unicode (side 405-406)
- Tegnet stor A har koden 65, stor B har 66 osv.

  Obs! De norske bokstavene ligger spredt utover (Å-197, Ø-216)



Eksempel på automatisk typeskifte:

```
char tegn = 'A';
int kodeNr = tegn; // kodeNr er nå 65
```

Eksempel på typetvang (*eksplisitt typeskifte*):

```
int kodeNr = 97;
char tegn = (char)kodeNr; // tegn er nå 'a'
```

side 405 - 406 **Unic**

kode	tegn	kode	tegn	kode	tegn
32		47	/	62	¿
33	!	48	0	63	?
34	″	49	1	64	@
35	#	50	2	65	A
36	$	51	3	66	B
37	%	52	4	67	C
38	&	53	5	68	D
39	′	54	6	69	E
40	(	55	7	70	F
41	)	56	8	71	G
42	*	57	9	72	H
43	+	58	:	73	I
44	,	59	;	74	J
45	-	60	¡	75	K
46	.	61	=	76	L

kode	tegn	kode	tegn	kode	tegn
77	M	94	^	111	o
78	N	95	_	112	p
79	O	96	‘	113	q
80	P	97	a	114	r
81	Q	98	b	115	s
82	R	99	c	116	t
83	S	100	d	117	u
84	T	101	e	118	v
85	U	102	f	119	w
86	V	103	g	120	x
87	W	104	h	121	y
88	X	105	i	122	z
89	Y	106	j	123	{
90	Z	107	k	124	—
91	[	108	l	125	}
92	\	109	m	126	~
93	]	110	n		

Særnorske bokstaver har fått tallkoder som bryter med alfabetisk sortering:

kode	tegn	kode	tegn	kode	tegn
197	Å	216	Ø	230	æ
198	Æ	229	å	248	ø

6108 Programmering i Java

Slutt på leksjon 2 – Del 2