Köst

Hvað er

1+true

1+2.0

1+2

1+2+”?”

1+(2+”?”)

Köst

Kast er þegar við breytum gildi af einu tagi yfir í gildi af öðru tagi.

- óbein: þegar hlutirnir gerast á bak við tjöldin
- bein: sérstaklega tekið fram í kóða með kasti eða föllum

expression	*expression type*	*expression value*
`"1234" + 99`	`String`	`"123499"`
`Integer.parseInt("123")`	`int`	`123`
`(int) 2.71828`	`int`	`2`
`Math.round(2.71828)`	`long`	`3`
`(int) Math.round(2.71828)`	`int`	`3`
`(int) Math.round(3.14159)`	`int`	`3`
`11 * 0.3`	`double`	`3.3`
`(int) 11 * 0.3`	`double`	`3.3`
`11 * (int) 0.3`	`int`	`0`
`(int) (11 * 0.3)`	`int`	`3`

Köst

Hvað er?

5 / 3

5.0 / 3

5 / 3.0

((double) 5) / 3

5 / ((double) 3)

(double) 5 / 3

(double) (5/3)

Köst

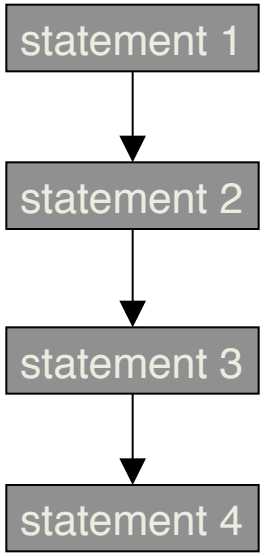
Hvaða gildi eru notuð?

- int -> double, heiltalan sem rauntala
- double -> int, námundað í átt að 0
- int -> String, strengurinn með stafina f. heiltöluna
- double -> String
- int,float -> boolean, ekki hægt, notið samanburð

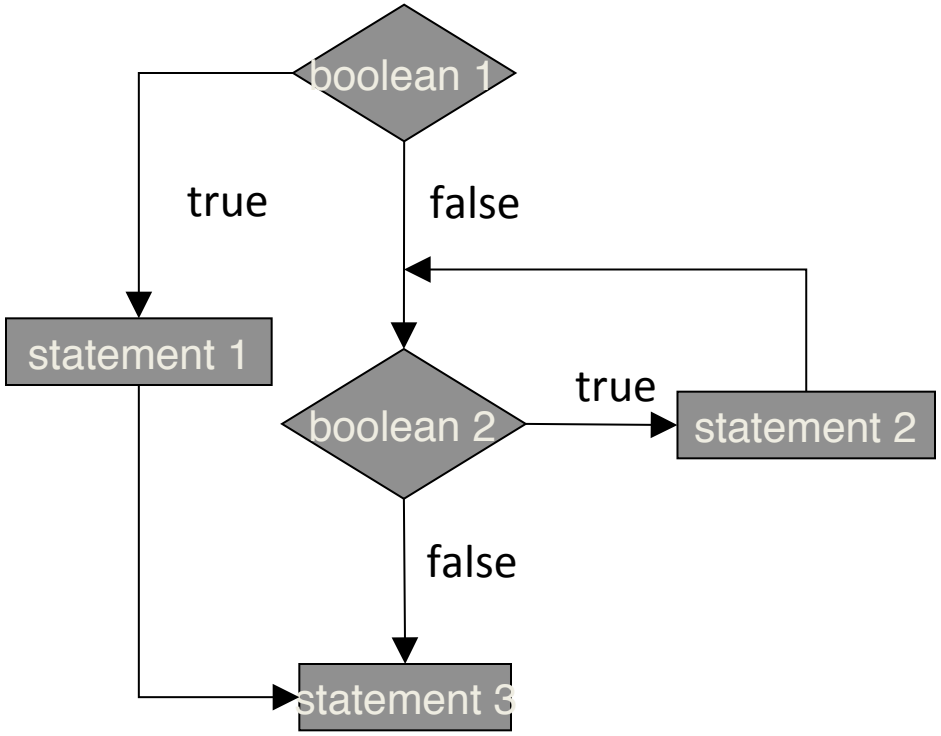
Samantekt

- Öll gildi í Java hafa sín eigin gagnatög
- Breytur í Java þarf að skilgreina fyrirfram með gagnatögum
- Hægt að breyta á milli með köstum
- Klassískar prófspurningar

Stýriskipanir (control flow) leyfa okkur að breyta hvaða skipanir eru framkvæmdar



Einfalt línulegt forrit



stýriskipanir leyfa flóknari forrit

Setningar vs. segðir

Segð (expression) er kóði sem skilar gildi.

- int a = 0;
- System.out.println(“a = “+a);
- boolean b = (a == 0);

Setning (statement) er minnsta eining forrits sem keyrð er

- int a = 0; // gildingarsetning (assignment statement)
- System.out.println(“Hello”); // fallakall (function call)
- boolean b; // skilgreiningarsetning (declaration statement)

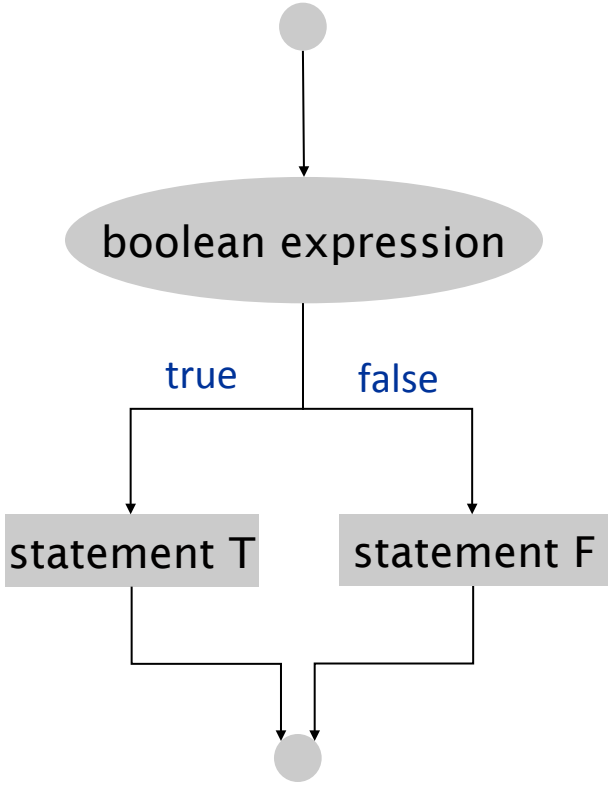
if stýrisetningin

if stýrisetningin er notuð til að skipta upp flæði

- boolean expression er segð sem skilar boolean gildi
- statement T er keyrð ef að gildið er satt
- statement F er keyrð ef að gildið er ósatt

```
if (boolean expression) {
    statement T;
}
else {
    statement F;
}
```

statement T og F mega vera runur af setningum

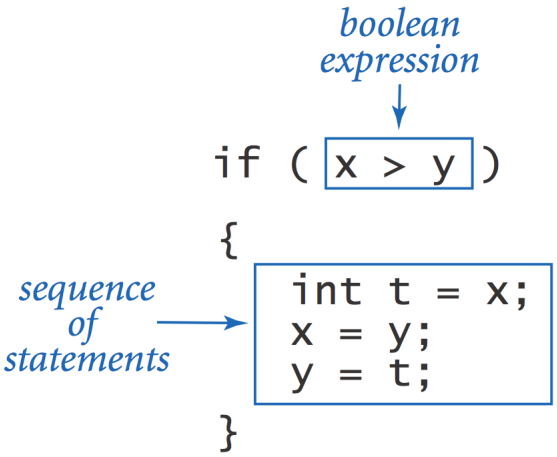


if stýrisetningin

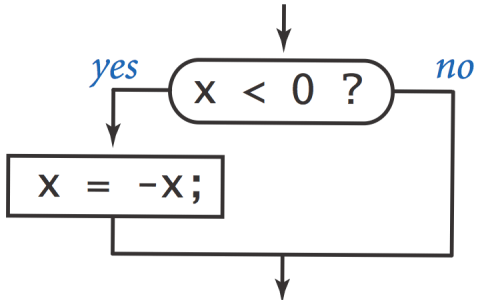
`if` stýrisetningin er notuð til að skipta upp flæði

- `boolean expression` er segð sem skilar boolean gildi
- statement T er keyrð ef að gildið er satt
- statement F er keyrð ef að gildið er ósatt

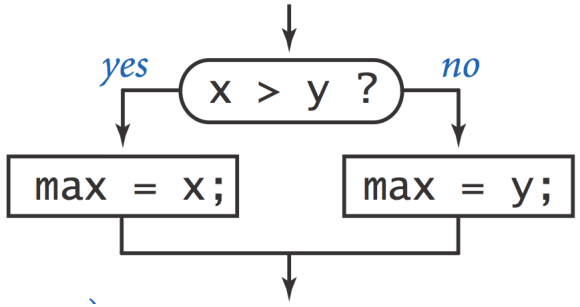
x og y eru heiltölur




```
if (x < 0) x = -x;
```




```
if (x > y) max = x;
else       max = y;
```



if dæmi

Math.random() skilar slembigildi í [0,1]

```
public class Flip {
   public static void main(String[] args) {
      if (Math.random() < 0.5) System.out.println("Heads");
      else                     System.out.println("Tails");
   }
}
```

```
% java Flip
Heads

% java Flip
Heads

% java Flip
Tails

% java Flip
Heads
```

if dæmi

absolute value	`if (x < 0) x = -x;`

put x *and* y *into sorted order*	`if (x > y)`
	`{`
	`  int t = x;`
	`  y = x;`
	`  x = t;`
	`}`

maximum of x *and* y	`if (x > y) max = x;` `else  max = y;`
error check for division operation	`if (den == 0) System.out.println("Division by zero");` `else  System.out.println("Quotient = " + num/den);`

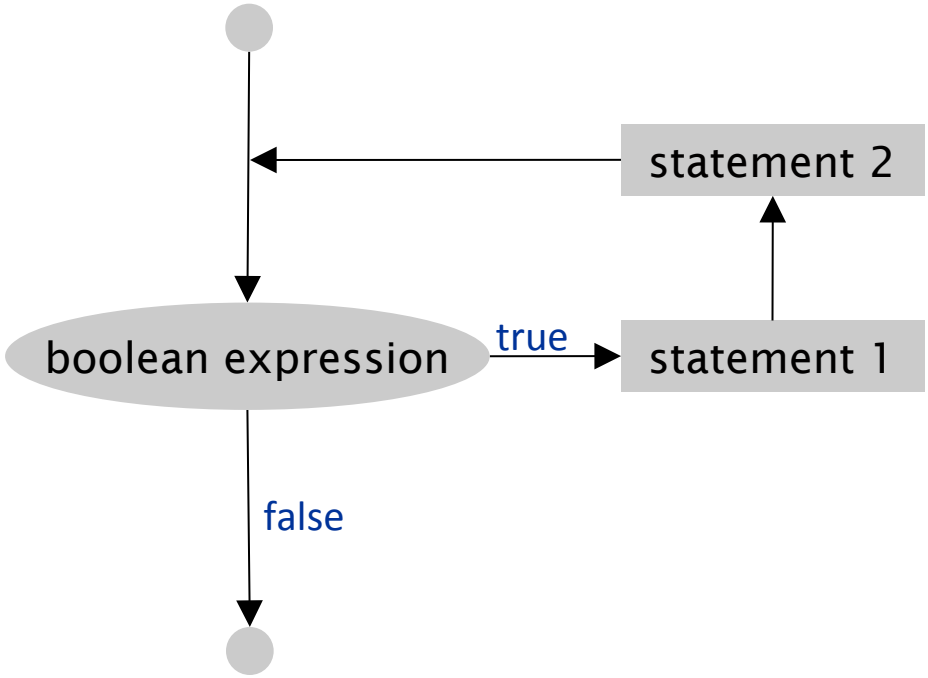
error check for quadratic formula	`double discriminant = b*b - 4.0*c;`
	`if (discriminant < 0.0)`
	`{`
	`  System.out.println("No real roots");`
	`}`
	`else`
	`{`
	`  System.out.println((-b + Math.sqrt(discriminant))/2.0);`
	`  System.out.println((-b - Math.sqrt(discriminant))/2.0);`
	`}`

While

while setningin er notuð til að endurtaka setningar

- Reikna út boolean segð
- ef satt þá framkvæmum við skipanir
- Enturtakist

```
while (boolean expression) {
    statement 1;
    statement 2;
}
```



- Gæti verið endurtekið 0 sinnum eða endalaust

while dæmi

Reiknar út veldi af 2 upp í N

```
int i = 0;
int v = 1;
while (i <= N) {
    System.out.println(i + " " + v);
    i = i + 1;
    v = 2 * v;
}
```

i	v	i <= N
0	1	true
1	2	true
2	4	true
3	8	true
4	16	true
5	32	true
6	64	true
7	128	false

```
0 1
1 2
2 4
3 8
4 16
5 32
6 64
```

N = 6

Kvaðratrót með Newton-Raphson aðferð

Newton-Raphson aðferð við að reikna út kvaðratrót af c:

- setjum $t_0 = c$
- Endurtökum þar til $t_i = c/t_i$ með nægri nákvæmni
- setjum $t_{i+1} = (t_i + c/t_i)/2$

$$
\begin{aligned}
t_0 &= && = && 2.0 \\
t_1 &= \tfrac{1}{2}(t_0 + \tfrac{2}{t_0}) && = && 1.5 \\
t_2 &= \tfrac{1}{2}(t_1 + \tfrac{2}{t_1}) && = && 1.4166666666666665 \\
t_3 &= \tfrac{1}{2}(t_2 + \tfrac{2}{t_2}) && = && 1.4142156862745097 \\
t_4 &= \tfrac{1}{2}(t_3 + \tfrac{2}{t_3}) && = && 1.4142135623746899 \\
t_5 &= \tfrac{1}{2}(t_4 + \tfrac{2}{t_4}) && = && 1.414213562373095
\end{aligned}
$$

Newton-Raphson

```
public class Sqrt {
   public static void main(String[] args) {
      double epsilon = 1e-15;
      double c = Double.parseDouble(args[0]);
      double t = c;
      while (Math.abs(t - c/t) > t*epsilon) {
         t = (c/t + t) / 2.0;
      }
      System.out.println(t);
   }
}
```

- Notum Math.abs() > t*epsilon til að meta nákvæmni
- Skrifum yfir t í hverri ítrun, notum ekki t_i og t_{i+1}

While

Við notum while til að keyra setningar aftur og aftur. Hvað með 10 sinnum? eða N sinnum?

```
int i = 1;
while (i <= 10 ) {
    ...
    i += 1;
}
```

```
int i = 0;
while (i < N ) {
    ...
    i += 1;
}
```

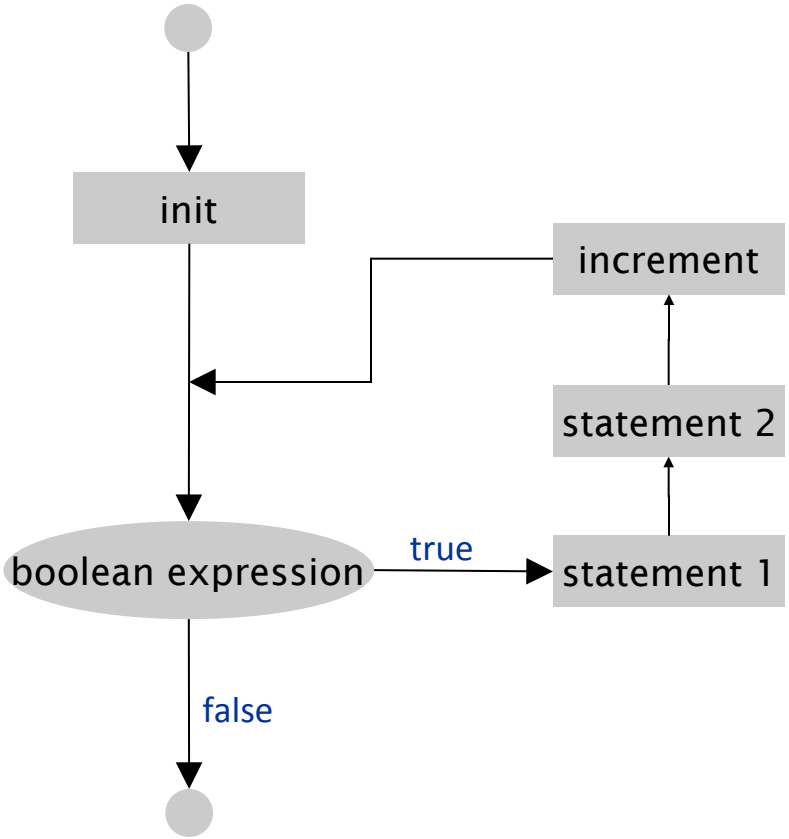
Einfaldara að nota `for`-lykkjur.

Notum while þegar við vitum ekki fyrirfram hvenær við klárum

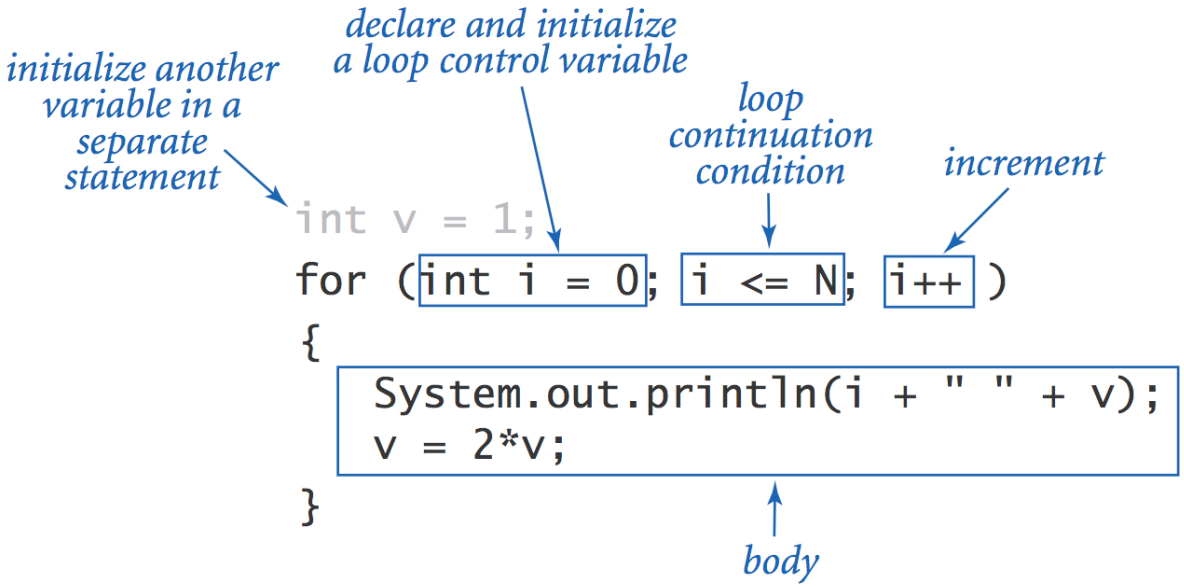
For lykkjur er einfaldari útgáfa af while lykkju

- Keyrum init setningu fyrst
- Athugum hvort skilyrðið er uppfyllt
- Síðan er for-blokkin keyrð
- Loks increment setning keyrð
- Endurtekið

```
for (init; boolean expression; increment) {
    statement 1;
    statement 2;
}
```



for lykkjur



Hvað prentast út þegar N = 5?

for og while lykkjur dæmi

print largest power of two less than or equal to N	`int v = 1;`
	`while (v <= N/2)`
	`  v = 2*v;`
	`System.out.println(v);`

compute a finite sum (1 + 2 + . . . + N)	`int sum = 0;`
	`for (int i = 1; i <= N; i++)`
	`  sum += i;`
	`System.out.println(sum);`

compute a finite product (N! = 1 × 2 × . . . × N)	`int product = 1;`
	`for (int i = 1; i <= N; i++)`
	`  product *= i;`
	`System.out.println(product);`

print a table of function values	`for (int i = 0; i <= N; i++)` `  System.out.println(i + " " + 2*Math.PI*i/N);`

Faldaðar for lykkjur

```
**********1
** * * * *2
* *   *   * 3
** *    *   4
*    *     *5
***   *     6
*      *    7
** *    *   8
* *      * 9
**   *     *10
```

```java
public class DivisorPattern {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);

        for (int i = 1; i <= N; i++) {
            for (int j = 1; j <= N; j++) {
                if ((i % j == 0) || (j % i == 0)) {
                    System.out.print("*");
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println(i);
        }
    }
}
```