Qjava

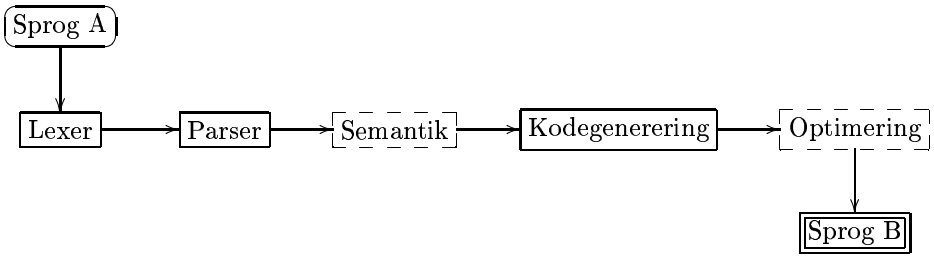
S	→	{ ⟨classdef⟩ }
classdef	→	"class" ⟨id⟩ "{" ⟨classcontents⟩ "}"
classcontents	→	{ (⟨vardef⟩ | ⟨fncdef⟩ | ";") }
vardef	→	⟨id⟩ ⟨id⟩ ";"
fncdef	→	"void" ⟨id⟩ "(" [⟨id⟩ ⟨id⟩ { "," ⟨id⟩ ⟨id⟩ }] ")" "{" ⟨sentences⟩ "}"
sentences	→	{ ⟨vardef⟩ | ⟨fnccall⟩|⟨if⟩|⟨while⟩|⟨break⟩ | ⟨return⟩ | ⟨assign⟩ | ";" }
fnccall	→	(⟨name⟩ | ⟨id⟩) "(" [⟨E⟩ { "," ⟨E⟩ }] ")" ";"
if	→	"if" "(" ⟨E⟩ ")" "{" ⟨sentences⟩ "}" "else" "{" [⟨sentences⟩] "}"
while	→	"while" "(" ⟨E⟩ ")" "{" [⟨sentences⟩] "}"
break	→	"break" ";"
return	→	"return" ["(" ")"] ";"
assign	→	(⟨name⟩ | ⟨id⟩) = ⟨E⟩ ";"
E	→	⟨E_1⟩ { "||" ⟨E_1⟩ }
E_1	→	⟨E_2⟩ { "&&" ⟨E_2⟩ }
E_2	→	⟨E_3⟩ { "|" ⟨E_3⟩ }
E_3	→	⟨E_4⟩ { "&" ⟨E_4⟩ }
E_4	→	⟨E_5⟩ { ("==" | "!=") ⟨E_5⟩ }
E_5	→	⟨E_6⟩ { ("<" | "<=") ⟨E_6⟩ }
E_6	→	⟨E_7⟩ { ("+" | "-") ⟨E_7⟩ }
E_7	→	⟨E_8⟩ { ("*" | "/" | "%") ⟨E_8⟩ }
E_8	→	⟨E_9⟩ | "new" ⟨id⟩ "()"
E_9	→	["!" | "-"] ⟨E_{10}⟩
E_{10}	→	⟨E_{11}⟩ | (⟨name⟩ | ⟨id⟩) ["(" ⟨E⟩ { "," ⟨E⟩ } ")"]
E_{11}	→	⟨number⟩ | "(" ⟨E⟩ ")"

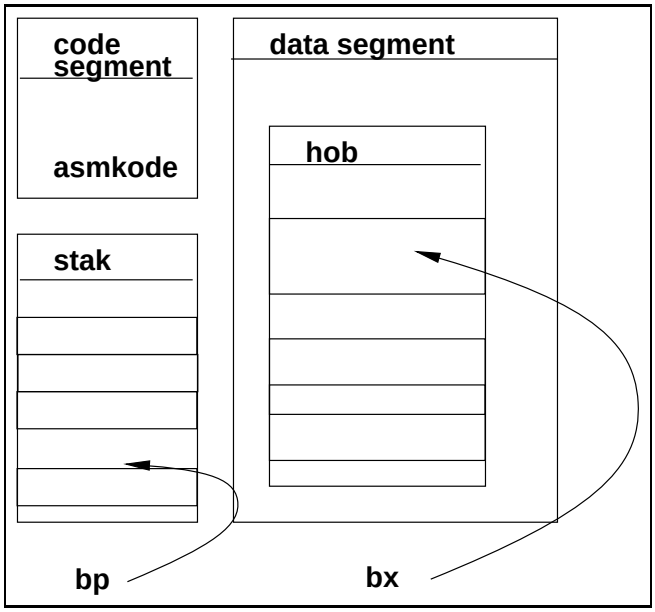
Begrænsninger$_1$

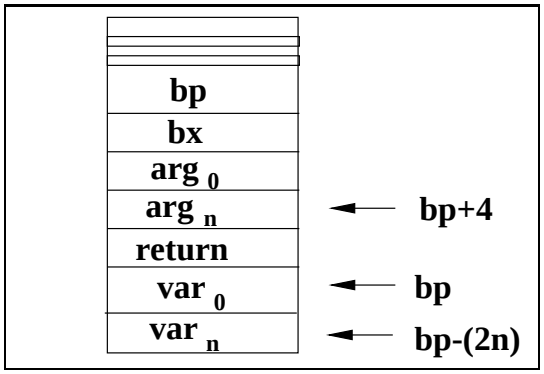
- Nestedede klassedefinitioner.
- Nedarvning ("extends").
- Access modifiers ("public", "private", ...)
- Ingen konstanter ("final").
- Ingen returværdi ved metodekald
- Operatorer og sætninger

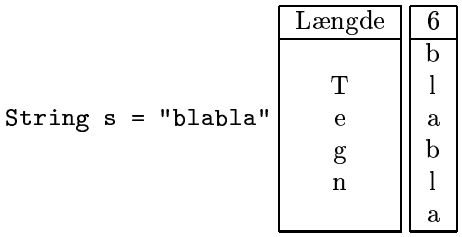
Begrænsninger$_2$

- Constructor
- Store programmer (pga. codesegment størrelse)
- Garbage collection + hobs faste størrelse $\Longrightarrow$ programmer kan ikke køre længe.
- `int` er 16 bit, og der understøttes ikke unicode karakterer.
- `while` og metoder kan kun have begrænset størrelse
- Begrænset antal standardfunktioner.


Sprog A
Lexer
Parser
Semantik
Kodegenerering
Optimering
Sprog B


code segment
asmkode
data segment
hob
stak
bp
bx


bp
bx
arg 0
arg n
return
var 0
var n
bp+4
bp
bp-(2n)



Optimering

Qjava	Qjava opt_1	Qjava opt_2	Qjava håndopt	JDK 1.3b	C	Assembler
11	10	8	6	9	8	5

```
class Prime
{
    static void main()
    {
        Prime ptr;
        ptr = new Prime();

        ptr.loop();
    }


    void loop()
    {
        int p; p = 3;

        while(p < 32000)
        {
            isPrime(p); p = p + 1;
        }
    }


    void isPrime(int p)
    {
        int i; i = 2;

        while(i < p)
        {
            if(p%i == 0){return; }
            else{i = i + 1; }
        }

        if(i == p){}
        else{}
    }
}
```

```
DOSSEG
.MODEL SMALL
.STACK 200h

.DATA

.CODE
     mov ax,@DATA
     mov ds,ax

    mov cx, 31999
again:
    mov si, 2               ; si = 2
primeTsTStart:
    cmp si, cx              ; while cx > si
    je primeTstEnd          ;

    mov ax, cx
    xor dx, dx
    div si
    cmp dx, 0               ; if cx % si == 0 then stop testing current number
    je primeTstEnd:
    inc si                  ; si++
    jmp primeTstStart
primeTstEnd:

    loop again              ; if cx > 0

    ; END OF SHOW
    mov ah,4ch
    int 21h

END
```