5-1/ J2

```
prog := d ...   e
 d := define (f  x ...) e
e := v  | (e ...)  | (if e e e)
   | x
v := num | bool | prim | f
x ∈ some set of variable names
f ∈ some set of function names
```

```
(define (Double x) (+ x x))
(Double (+ (Double 1) 3)) ——
(define (Quad y) (Double (Double y)))
(Quad (+ 1 (Double 3)))
```

5-2)

$$f(x) = 1 + x$$

$$f(3) = 1 + 3 = 4$$

$$f(3+4) = f(7) = 1 + 7 = 8$$

functions give us variable and <u>substitution</u>

$$\text{subst} : x \times v \times e \to e$$

$$\text{subst}\ x\ x_v\ v = v$$

$$\text{subst}\ x\ x_v\ x = x_v$$

$$y = y$$

$$(\text{if}\ e_c\ e_t\ e_f) = \text{if}\ (\text{subst}\ x\ x_v\ e_c)$$

$$(\text{subst}\ x\ x_v\ e_t)$$

$$(\text{subst}\ x\ x_v\ e_f)$$

5-3/ $\text{interp} : e \Rightarrow v$ $\Delta = \text{map}(f, d)$

$\text{interp} : \Delta \times e \Rightarrow v$

$\text{interp}\ \Delta\ f(v_0, \ldots, v_n) = \text{interp}\ \Delta\ e_{body}'$

where $e_{body}' = \overrightarrow{\text{subst}}\ (x_0 \ldots x_n)\ (v_0 \ldots v_n)\ e_{body}$

$\text{define}\ f(x_0, \ldots, x_n)\ e_{body} = \Delta[f]$

$CK_0 : e \to v$

5-4/ CK_1 $\qquad$ $CK_1 : \Delta \times e \to v$

7 $\langle v_n, \text{kapp}((v_0 \ldots, f), (), k) \rangle$

$\mapsto \langle \text{ebody}' , k \rangle$

where $\text{ebody}' = \overrightarrow{\text{subst}}\ (x_0 \ldots x_n)\ (v_0 \ldots v_n)\ \text{ebody}$

$\text{define } f(x_0 \ldots x_n)\ \text{ebody} = \Delta(f)$

5-5/ (define (Double x) (+ x x))

(Double (Double 1))

↓ inject

Δ = [Double ↦ define (Double (x) (+ x x)]

< Double (Double 1), kret > ↓ 4

< Double, kapp () ((Double 1)) kret > ↓ 5

< Double 1, kapp (Double) () kret > ↓ 4

< Double, kapp () (1) (kapp (Double) () kret) > ↓ 5

< 1, kapp (Double) () (kapp (Double) () kret) > ↓ 7

subst (+ x x) (x) (1) → (+ 1 1)

< (+ 1 1), kapp (Double) () kret) > ↓ 4,5,5,6

< 2 , kapp (Double) () kret) > ↓ 7

< (+ 2 2), kret > ↓ 4,5,5,6

< 4, kret > ⟶ 4

5-6/ the K structure is implemented by a stack in many languages

```
class Test {
  static int F( int x) { return F(x); }
  public static void main () { F(0); } }
```

"stack overflow ..." F, F, F, F, F, ...

```
(define (F x) (F x))
(F 0)
```

$\Delta = [F \mapsto \text{define } F(x)\ (F x)]$

< F 0, kret >

< F, kapp () (0) kret >

< 0, kapp (F) () kret >

$\Delta(F) = \text{define } F(x)\ (F\ x)$

< subst (x) (0) (F x), kret >

< F 0, kret >

"tail-call optimization"

proper function call implementation