Funktionale Sprachen: LISP

Strukturen:

(defstruct *name* [komponenten])

generiert automatisch

- make-*name*
- copy-*name*
- *name*-p
- *name-komp1, name-komp2,…*

Funktionale Sprachen: LISP

Beispiel:

```
(defstruct point
        x
        y
        z)
```

```
(make-point   :x  2  :y 3  :z  5)              → Punkt (2,3,5)
(setf p1 (make-point   :x  2  :y 3  :z  5))    → #S(POINT :x 2 :y 3 :z 5)
(setf p2 (copy-point p1))                       → Punkt P2 Kopie von P1
(point-p p2)                                    → T
```

Funktionale Sprachen: LISP

Beispiel (Fortsetzung):

```
(defun ursprung-dist   (p)
        (sqrt  (+  (*  (point-x  p) (point-x  p))
                   (*  (point-y  p) (point-y  p))
                   (*  (point-z  p) (point-z  p))  )))
```

```
(ursprung-dist  p1)                    → 6.164414
```

Funktionale Sprachen: LISP

Objekte: Klassendefinitionen ähnlich, ABER…

```
(defclass  name ()
        (attr1
         attr2
         …))
```

- [optional]: Vaterklassen in () auflisten
- Attribute heißen Slots
- Zugriffsfunktionen: nicht automatisch generiert
- aber es existieren: (make-instance …), (slot-value …), (with-slots ….)

Funktionale Sprachen: LISP

Beispiel:

```
(defclass  point ()
        (x
         y
         z))

(make-instance  ´point)                          → erzeugt Punkt-Objekt
(setf p1 (make-instance ´point))                 → #<POINT #x000333D73B48>
(setf (slot-value p1 ´x) 2  (slot-value p1 ´y) 3  (slot-value p1 ´z) 5)
(defun urspr-dist  (point)
        (with-slots (x y z) point
                (sqrt (+  (* x x) (* y y) (* z z)))))
(urspr-dist   p1)                                → 6.164414
```

Funktionale Sprachen: LISP

Beispiel (Erweiterung):

```
(defclass  point ()
        (x :accessor pointx :initarg :x
         y :accessor pointy :initarg :y
         z :accessor pointz :initform 0
        farbe :reader getfarbe :allocation :class))
```

:accessor erzeugt Funktionen (pointx) und (setf pointx) als Getter- und Setter-Funktionen

:reader erzeugt nur Getter-Funktion

:initarg

:initform erlauben in make-instance Initialisierung der Attribute mittels Parameter :x bzw :y bzw. mittels vorgegebenen Wert

:allocation :class Klassenattribut (Default :instance)

Funktionale Sprachen: LISP

Methoden:

```
(defmethod urspr-dist (point)
        ….)

(defmethod  urspr-dist ((p point))
        (sqrt (+  (* pointx pointx)
                  (* pointy pointy)
                  (* pointz pointz))))
```

- analog zu (defun …)- Funktionen
- oder mit (var class) –Angabe in Parameterliste: Methode wird nur ausgeführt, wenn akt. Parameter Instanz der Klasse