Programare funcţională – Laboratorul 12

CLOS (Common Lisp Object System)

Isabela Drămnesc

May 15, 2012

1 Concepte

- Clase, Instante
- Proprietati
- Metode, Metode generice
- Superclase, Precedenta
- Mostenire

2 Clase şi instanţe:

```
> (use-package :clos)

; definirea unei clase om care are componentele: locuieste,
; lucreaza, casatorit

> (defclass om ()
    (locuieste lucreaza casatorit))

; pentru a crea o instanta a clasei om

> (setq isabela (make-instance 'om))

> (describe isabela)

#<OM #x19F33DED> is an instance of the CLOS class #1=#<STANDARD

-CLASS OM>.
Slots:
  LOCUIESTE    unbound
  LUCREAZA     unbound
  CASATORIT    unbound
```

3 Slot Properties

Modificarea si accesarea valorilor componentelor:

```
; folosim :initform pentru a da o valoare implicita unei
; componente

> (defclass om ()
      ((locuieste :initform 'Timisoara)
       (lucreaza :initform 'UVT)
       (casatorit :initform '??)))

> (setq isabela (make-instance 'om))

> (describe isabela)

; daca dorim sa setam o valoare particulara si in mod explicit
; utilizam cuvantul cheie :initarg

> (defclass om ()
     ((locuieste :initform 'Timisoara)
      (lucreaza :initform 'UVT :initarg :lucreaza_in_proba)
      (casatorit :initform 'nu :initarg :casatorit_initial)))

> (setq ionela (make-instance 'om :lucreaza_in_proba
'Alcatel))

> (describe ionela)

> (setq ionut (make-instance 'om :lucreaza_in_proba 'Conti
:casatorit_initial 'da))

> (describe ionut)

> (setq irina (make-instance 'om :casatorit_initial 'da))

> (describe irina)

;; accesarea valorii unei componente se face utilizand
; slot-value

> (setq florian (make-instance 'om))

> (slot-value florian 'locuieste)

> (setf (slot-value florian 'locuieste) 'Cluj)

> (describe florian)

;;; citirea, scrierea sau accesul la o anumita componenta
; se poate face mai usor definind: reader, writer, accessor

> (defclass dreptunghi ()
      ((lungime :reader lungime? :writer lungime! :accessor
```

```
lung)
        latime))

> (setq figura1 (make-instance 'dreptunghi))

> (describe figura1)

> (setf (lung figura1) 10)

> (lungime? figura1)

> (lung figura1)

> (describe figura1)

;;; si o alta optiune de a modifica valorile componentelor unui
; obiect -- declararea lor ca fiind alocate pe clasa si nu pe
; instanta

> (defclass student_la_info_UVT ()
      ((adresa_UVT :initform 'V_Parvan :allocation :class)
       (are_cursuri :initform 'informatica)))

> (setq student1 (make-instance 'student_la_info_UVT))

> (describe student1)

> (setq student2 (make-instance 'student_la_info_UVT))

> (describe student2)

> (setf (slot-value student1 'adresa_UVT) 'Bogdanesti)

> (describe student2)

> (describe student1)
```

4 Superclase, Precedenta, Mostenire

```
;;;; mostenirea proprietatilor in CLOS

> (defclass inginer (om) ())

> (defclass dezvoltator_soft (inginer) ())

> (defclass doctor (om) ())

> (defclass doctor_dezv_soft (doctor dezvoltator_soft) ())
```

```
> (subtypep 'doctor_dezv_soft 'inginer)

> (setq dan (make-instance 'doctor))

> (subtypep (type-of dan) 'om)

> (subtypep (type-of dan) 'inginer)


;; pentru a stabili ordinea in care sunt mostenite valorile
; componentelor, se construieste pentru fiecare clasa o lista
; de precedenta a claselor

> (clos::class-precedence-list (find-class 'doctor_dezv_soft))
(#<STANDARD-CLASS DOCTOR_DEZV_SOFT> #<STANDARD-CLASS DOCTOR>
 #<STANDARD-CLASS DEZVOLTATOR_SOFT> #<STANDARD-CLASS INGINER>
 #<STANDARD-CLASS OM :VERSION 2> #<STANDARD-CLASS STANDARD-
OBJECT>
 #<BUILT-IN-CLASS T>)

> (clos::class-direct-superclasses (find-class 'doctor_dezv_soft))
```

4.1 Ierarhie de clase:

```
;;;; exemplu:

> (defclass c1 ()
     ((s1 :initform 1 :initarg :1s1 :accessor a1s1)))

> (defclass c2 (c1)
    ((s1 :initform 2 :initarg :2s1 :accessor a2s1)))

> (defclass c3 (c1)
     ((s1 :initform 3 :initarg :3s1 :accessor a3s1)))

> (defclass c4 (c1)
     ((s1 :initform 4 :initarg :4s1 :accessor a4s1)))

> (defclass c5 (c1)
       ((s1 :initform 5 :initarg :5s1 :accessor :a5s1)))

> (defclass c6 (c2 c3)
       ((s1 :accessor a6s1)))

> (defclass c7 (c4 c5)
        ((s1 :initform 7 :accessor a7s1)))

> (defclass c8 (c6 c7)
         ((s1 :accessor a8s1)))
```

```
;;; lista de precedenta a claselor pentru clasa c8 este:

> (clos::class-precedence-list (find-class 'c8))
(#<STANDARD-CLASS C8> #<STANDARD-CLASS C6> #<STANDARD-CLASS C2>
 #<STANDARD-CLASS C3> #<STANDARD-CLASS C7> #<STANDARD-CLASS C4>
 #<STANDARD-CLASS C5> #<STANDARD-CLASS C1> #<STANDARD-CLASS
STANDARD-OBJECT>
 #<BUILT-IN-CLASS T>)

> (setq instance-of-c8 (make-instance 'c8))

> (slot-value instance-of-c8 's1)
2    ;valoare mostenita din c2 (valoarea definita prima in lista
     ; de precedenta a claselor)

;; toate specificatiile :accessor si :initarg sunt mostenite si
; pot fi folosite in clasa c8

> (a1s1 instance-of-c8)

> (a8s1 instance-of-c8)

> (setq instance-of-c8-2 (make-instance 'c8 :4s1 29))

> (a1s1 instance-of-c8-2)
```

5 Metode, Metode generice

Definirea metodelor:

```
> (defmethod vorbeste ((el om) ceva)
     (format t "~% locuiesc in ~A lucrez la ~a si vorbesc ~A"
      (slot-value el 'locuieste)
      (slot-value el 'lucreaza)
       ceva)
    'end)

> (vorbeste ionut 'despre_LISP)

> (vorbeste ionela 'oare_ce?)

> (defclass catel ()
    (culoare rasa))

> (setq cutzu (make-instance 'catel))

> (defmethod vorbeste ((el catel) ceva) (print 'hamham))

> (vorbeste cutzu 'ceva)
```

```
;;; metodele pot fi mostenite

> (setq danut (make-instance 'doctor_dezv_soft
:lucreaza_in_proba 'La-PC))

> (describe danut)

> (vorbeste danut 'ce_vreau_eu)

;;;; desi metoda vorbeste nu e definita in clasa
; doctor_dezv_soft, ea este mostenita din clasa om

> (vorbeste ionela 'mult)

;; multimea metodelor care au acelasi nume formeaza o functie
; generica. Fiecare lista de metode aplicabile este o submultime a
; metodelor unei anumite functii generice.

; particularizarea unei metode specifica unei anumite instante:

> (defmethod vorbeste ((el (eql ionut)) ceva)
          (declare (ignore ceva))
        (call-next-method)
       (print 'ma_casatoresc)
      'out)

[87]> (vorbeste ionut 'ceva)

; call-next-method apeleaza urmatoarea metoda aplicabila pentru
; apelul dat din lista de metode aplicabile si este o modalitate
; de combinare a metodelor.
; Pentru a vedea lista acestor metode studiem exemplul urmator:

> (defmethod mmm ((el om)) (print 'om))

> (defmethod mmm ((el inginer))
                 (print 'inginer)
                 (call-next-method))

> (defmethod mmm ((el doctor))
       (print 'doctor)
       (call-next-method))

> (defmethod mmm ((el doctor_dezv_soft))
              (print 'doctor)
            (call-next-method))

> (mmm ionut)

> (mmm dan)
```

```
> (mmm danut)

> (defmethod se_casatoreste ((el om) (ea om))
       (setf (slot-value ea 'locuieste) (slot-value el
'locuieste)
             (slot-value el 'casatorit) 'da
             (slot-value ea 'casatorit) 'da))

> (describe ionela)

> (describe danut)

> (se_casatoreste danut ionela)

> (describe ionela)

> (describe danut)
```

6 Exercitiu din [St.Trausan-Matu]

```
> (use-package :clos)

> (defclass obiect_fizic ()
    ((material :initarg :material)
     (culoare :initarg :culoare)))

> (defclass obiect_sferic (obiect_fizic)
    ((raza :initarg :raza)))

> (defmethod volum ((x obiect_sferic))
    (* 4 pi (expt (slot-value x 'raza) 3)))

> (defclass obiect_cubic (obiect_fizic)
     ((latura :initarg :latura)))

> (defmethod volum ((x obiect_cubic))
     (expt (slot-value x 'latura) 3))

> (defclass obiect_din_plastic (obiect_fizic)
     ((material :initform 'plastic)))

> (defclass obiect_din_fier (obiect_fizic)
     ((material :initform 'fier)))

> (defclass minge (obiect_sferic obiect_din_plastic) ())

> (setf minge1 (make-instance 'minge :culoare 'rosu :raza 2)
```

```
                minge2 (make-instance 'minge :culoare 'alb :raza 3))

> (slot-value minge2 'material)

> (volum minge1)

> (volum minge2)

> (describe minge1)

> (defclass cub (obiect_cubic obiect_din_fier) ())

> (setf cub1 (make-instance 'cub :latura 10 :culoare
'violet))

> (describe cub1)

> (volum cub1)
```

7 Tema

1. Folosindu-vă de exerciţiile de mai sus creaţi un program cu următoarele cerinţe:

- o clasă FiguraGeometrica cu componentele: nume, o metoda generica pentru calculul ariei şi o metodă pentru afişarea figurii;
- o clasă Cerc care moşteneşte numele si calculul ariei de la clasa FiguraGeometrica şi are în plus componenta razacercului şi o metodă pentru afişare;
- o clasă Triunghi care moşteneşte numele si calculul ariei de la clasa FiguraGeometrica şi are în plus o metodă pentru afişare;
- o clasă Dreptunghi: lungime, latime, o metodă pentru calculul ariei;
- o clasă Patrat: latura pătratului va fi setată cu lăţimea dreptunghiului, o metodă pentru calculul ariei şi o metodă de afişare.
- creaţi cel puţin două instanţe pentru fiecare clasă;
- transmiteţi mesaje între obiectele claselor;

Notă: Termen de realizare: laboratorul următor.