ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

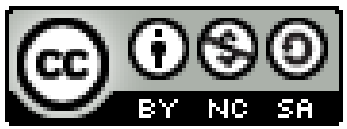
ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Συστήματα Γνώσης

Πρακτικό Κομμάτι Μαθήματος

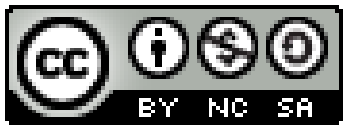
Η Αντικειμενοστραφής Γλώσσα Προγραμματισμού COOL του CLIPS

Νίκος Βασιλειάδης, Αναπλ. Καθηγητής
Τμήμα Πληροφορικής



Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Η Αντικειμενοστραφής Γλώσσα Προγραμματισμού COOL του CLIPS

Βασικά Χαρακτηριστικά COOL

- Αφαίρεση (abstraction)
- Εγκλεισμός (encapsulation)
- Κληρονομικότητα (inheritance)
- Πολυμορφισμός (polymorphism)
- Δυναμική δέσμευση (dynamic binding)

Αφαίρεση

- Ορισμός: Διαισθητική, υψηλού επιπέδου αναπαράσταση μίας σύνθετης έννοιας.
- Στην COOL, ο ορισμός νέων κλάσεων επιτρέπει την αφαίρεση νέων τύπων δεδομένων.
- Οι ιδιότητες (slots) και οι μέθοδοι (message-handlers) αυτών των κλάσεων περιγράφουν τις ιδιότητες (properties) και τη συμπεριφορά (behavior) μίας καινούριας ομάδας αντικειμένων.

Εγκλεισμός (1/2)

- Τα χαρακτηριστικά ενός αντικειμένου δεν είναι απευθείας προσβάσιμα στον υπόλοιπο κόσμο, δηλαδή στο υπόλοιπο πρόγραμμα.
  - Συνήθως η εσωτερική κατάσταση του αντικειμένου αποκρύπτεται.
  - Αυτή η ιδιότητα ονομάζεται εγκλεισμός των ιδιοτήτων του αντικειμένου.

Εγκλεισμός (2/2)

- Η COOL υποστηρίζει την ιδιότητα του εγκλεισμού, απαιτώντας την αποστολή μηνυμάτων (messages) για το χειρισμό στιγμιοτύπων (instances) των κλάσεων, ορισμένων από το χρήστη.
- Ένα στιγμιότυπο δε μπορεί να αποκριθεί σε ένα μήνυμα για το οποίο δεν έχει καθορισμένη μέθοδο.

Κληρονομικότητα (1/2)

- Οι κλάσεις είναι συνήθως οργανωμένες σε ιεραρχίες.
  - Οι πιο γενικές κλάσεις είναι τοποθετημένες ψηλά στην ιεραρχία, ενώ οι πιο συγκεκριμένες χαμηλότερα.
  - Οι κλάσεις που βρίσκονται ψηλά έχουν κάποια γενικά χαρακτηριστικά και μεθόδους τα οποία είναι κοινά για όλες τις κλάσεις που βρίσκονται χαμηλότερα στην ιεραρχία.

Κληρονομικότητα (2/2)

- Για την αποφυγή της επανάληψης ορισμού κοινών χαρακτηριστικών και μεθόδων υπάρχει η κληρονομικότητα.
  - Η δομή και η συμπεριφορά μιας γενικότερης κλάσης κληρονομείται στις περισσότερο συγκεκριμένες.

Πολλαπλή Κληρονομικότητα (1/2)

- Μία κλάση μπορεί να συνδέεται ιεραρχικά με περισσότερες της μίας γενικότερες κλάσεις.
  - Η κλάση που συνδέεται με πολλαπλές γενικότερες κλάσεις κληρονομεί χαρακτηριστικά και μεθόδους από όλες.

Πολλαπλή Κληρονομικότητα (2/2)

- Η COOL, χρησιμοποιώντας την υπάρχουσα ιεραρχία των κλάσεων, ορίζει τη ***λίστα προτεραιότητας κλάσεων*** (*class precedence list*) για μία νέα κλάση.
  - Αντικείμενα, τα οποία είναι στιγμιότυπα της νέας κλάσης, μπορούν να κληρονομήσουν ιδιότητες και συμπεριφορά από κάθε μία από τις κλάσεις της λίστας
  - Η λέξη ***προτεραιότητα*** υποδηλώνει ότι μία μέθοδος μίας κλάσης, που είναι πρώτη στη λίστα, υπερισχύει της ίδιας μεθόδου άλλης κλάσης που βρίσκεται πιο μετά στη λίστα.

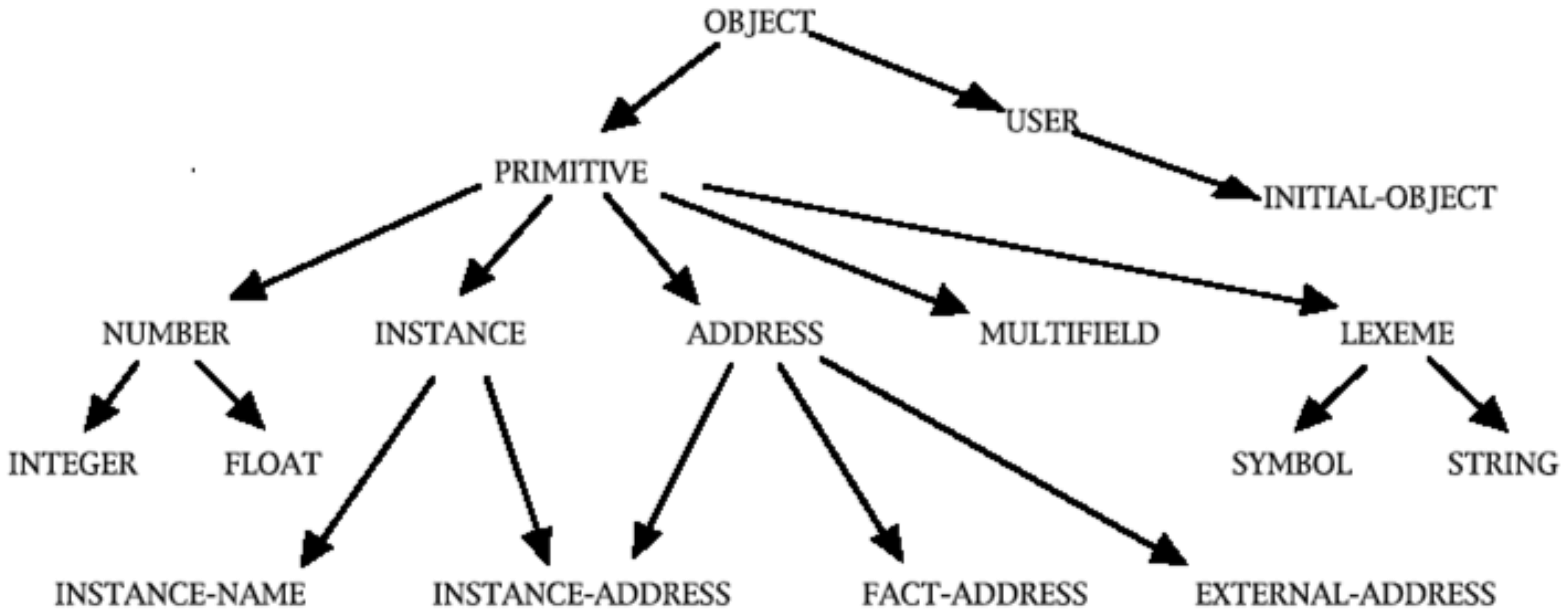
Πολυμορφισμός

- Ένα αντικείμενο μπορεί να απαντήσει σε ένα μήνυμα με ένα εντελώς διαφορετικό τρόπο από ό,τι ένα άλλο αντικείμενο.
- Αυτή η ιδιότητα ονομάζεται ***πολυμορφισμός***.
- Ο πολυμορφισμός επιτυγχάνεται επισυνάπτοντας μεθόδους, που έχουν το ίδιο όνομα αλλά εκτελούν διαφορετικές ενέργειες, στις κλάσεις των δύο αντικειμένων αντίστοιχα.

Δυναμική Δέσμευση

- Η ιδιότητα της ***δυναμικής δέσμευσης*** υποστηρίζεται στην COOL με την έννοια ότι μία αναφορά αντικειμένου (object reference) κατά την κλήση μίας συνάρτησης δεν έχει τιμή μέχρι τη στιγμή της εκτέλεσης.
- Για παράδειγμα, ένα όνομα στιγμιότυπου (instance-name) ή μία μεταβλητή μπορεί να αναφέρεται σε ένα αντικείμενο όταν στέλνεται ένα μήνυμα και να αναφέρεται σε ένα άλλο αντικείμενο κάποια άλλη στιγμή αργότερα.

Προκαθορισμένες Κλάσεις Συστήματος



Προκαθορισμένες Κλάσεις Συστήματος

- Δεν πρέπει να διαγραφούν ή να τροποποιηθούν
- Όλες οι κλάσεις είναι αφηρημένες (abstract), δηλαδή χρησιμοποιούνται μόνο για λόγους κληρονομικότητας.
  - Άμεσα αντικείμενα αυτών των κλάσεων δεν επιτρέπονται.
  - Εξαίρεση η **`INITIAL-OBJECT`**.

Προκαθορισμένες Κλάσεις Συστήματος

- Καμία από τις κλάσεις δεν έχει ιδιότητες και μεθόδους.
  - Η USER έχει μεθόδους.
  - Ο χρήστης μπορεί να *επισυνάψει* μεθόδους σε όλες τις κλάσεις συστήματος εκτός από τις κλάσεις **`INSTANCE`**, **`INSTANCE-ADDRESS`** και **`INSTANCE-NAME`**.

Προκαθορισμένες Κλάσεις Συστήματος

- Η **`OBJECT`** είναι υπερκλάση όλων των άλλων, συμπεριλαμβανομένων και των ορισμένων από το χρήστη κλάσεων.
- Όλες οι κλάσεις που ορίζονται από το χρήστη, θα έπρεπε, αλλά δεν απαιτείται, να κληρονομούν άμεσα ή έμμεσα από την κλάση **`USER`**.
  - Αυτή η κλάση έχει όλες τις βασικές μεθόδους του συστήματος, όπως αρχικοποίηση (initialization) και διαγραφή (deletion).

Ορισμός Κλάσεων

- Μέσω της συνάρτησης **`defclass`**
- Η **`defclass`** αποτελείται από 5 στοιχεία:
  - **όνομα**
  - **λίστα υπερκλάσεων** από την οποία η νέα κλάση κληρονομεί ιδιότητες και μεθόδους
  - **προσδιοριστής**, ο οποίος καθορίζει εάν θα δημιουργούνται **άμεσα αντικείμενα** της νέας κλάσης
  - **προσδιοριστής**, ο οποίος καθορίζει εάν τα αντικείμενα αυτής της κλάσης μπορούν να **ταιριάζουν** με πρότυπα αντικειμένων στο αριστερό μέρος των κανόνων
  - **λίστα ιδιοτήτων** που ορίζονται στη νέα κλάση

Σύνταξη defclass

```
(defclass <name> [<comment>]
    (is-a <superclass-name>+)
    [<role>]
    [<pattern-match-role>]
   <slot>*
    <handler-documentation>*
)
```

Παράδειγμα Ορισμού Κλάσεων

```
(defclass vehicle
     (is-a USER)
     (slot fuel-type (type SYMBOL))
     (slot tank-capacity (type INTEGER))
     (slot fuel-loaded (type INTEGER)))

(defclass car
     (is-a vehicle)
     (slot consumption-rate
          (type INTEGER))
     (slot reset-counter
          (type INTEGER)))
```

Λίστα Υπερκλάσεων

- Η λίστα υπερκλάσεων υποδηλώνει προτεραιό-τητα στην κληρονόμηση ιδιοτήτων και μεθόδων

```
(defclass A
    (is-a B C D)
    ...
)
```

- Προτεραιότητα στον ορισμό ιδιοτήτων και μεθόδων έχει η κλάση **A**
  - Στη συνέχεια η προτεραιότητα στην κληρονόμηση ορίζεται με τη σειρά **(B C D)**

Αφηρημένες / Συγκεκριμένες Κλάσεις

```
(defclass A
   (is-a ...)
   (role abstract/concrete)
```

- Μία αφηρημένη κλάση (**abstract**) χρησιμοποιείται μόνο για λόγους κληρονόμησης ιδιοτήτων και μεθόδων σε υποκλάσεις της, και δεν έχει άμεσα στιγμιότυπα.
  - Αν χρησιμοποιηθεί ποτέ η **make-instance** για μία τέτοια κλάση θα εμφανιστεί μήνυμα λάθους

Αφηρημένες / Συγκεκριμένες Κλάσεις

```
(defclass A
   (is-a ...)
   (role abstract/concrete)
```

- Μία συγκεκριμένη κλάση (**concrete**) μπορεί να έχει άμεσα στιγμιότυπα.
- Αν δεν υπάρχει ο συγκεκριμένος προσδιοριστής στον ορισμό μιας κλάσης, καθορίζεται μέσω της κληρονομικότητας.
  - Μία υποκλάση της κλάσης συστήματος **USER** θεωρείται **concrete** κλάση, αν δεν προσδιορίζεται ο ρόλος της.

Κλάσεις ταυτοποιήσιμες ή όχι

- Όταν μία κλάση χαρακτηρίζεται ως **`reactive`** τότε τα αντικείμενά της μπορούν να ταυτοποιηθούν στη συνθήκη ενός κανόνα.
- Όταν μία κλάση χαρακτηρίζεται ως **`non-reactive`** τότε τα αντικείμενά της **δεν** μπορούν να ταυτοποιηθούν στη συνθήκη ενός κανόνα, ακόμα και αν θεωρητικά «ταιριάζουν».

Κλάσεις ταυτοποιήσιμες ή όχι

- Μία **`abstract`** κλάση δεν μπορεί να είναι **`reactive`**.
  - Έτσι κι αλλιώς δεν έχει αντικείμενα!
- Αν δεν υπάρχει ο συγκεκριμένος προσδιοριστής, καθορίζεται μέσω της κληρονομικότητας.
  - Μία υποκλάση της κλάσης συστήματος **`USER`** θεωρείται **`reactive`** κλάση, αν δεν προσδιορίζεται ο ρόλος της, εκτός αν είναι **`abstract`**.

Ιδιότητες (slots)

- Οι ιδιότητες αποθηκεύουν τιμές που σχετίζονται με τα στιγμιότυπα-αντικείμενα κάθε κλάσης.
- Το όνομα του slot μπορεί να είναι οποιοδήποτε σύμβολο, εκτός από ***is-a*** και ***name*** που χρησιμοποιούνται στις συνθήκες των κανόνων.
- **Υπάρχουν 2 είδη slots:**
  - **Απλής τιμής (`slot`)**
  - **Πολλαπλών τιμών (`multislot`)**

Ιδιότητες (slots)

- Κάθε αντικείμενο έχει ένα αντίγραφο από τις ιδιότητες της άμεσης κλάσης του, καθώς και των ιδιοτήτων που κληρονομεί η κλάση.
- Τα slots κληρονομούνται από τις κλάσεις με τη σειρά που ορίζεται από την λίστα προτεραιότητας κλάσεων

Ιδιότητες (slots)

- Εάν 1 slot κληρονομείται από 2 διαφορετικές κλάσεις, τότε χρησιμοποιείται ο ορισμός από την πιο συγκεκριμένη κλάση, δηλαδή αυτή που βρίσκεται πιο αριστερά στη λίστα προτεραιότητας
- Υπάρχουν και slots που δεν κληρονο-μούνται λόγω της δήλωσης no-inherit
  - Εξαίρεση αποτελούν τα composite slots

Όψεις (facets) (1/2)

- Οι όψεις (*facets*) χαρακτηρίζουν τα slots.
  - Προκαθορισμένη τιμή (default)
  - Αποθήκευση τιμής (storage)
  - Πρόσβαση τιμής (access)
  - Προώθηση κληρονόμησης τιμής (inheritance propagation)
  - Κληρονόμηση όψεων (source)

Όψεις (facets) (2/2)

- Δυνατότητα ταυτοποίησης (pattern-matching)
- Ορατότητα τιμής της ιδιότητας σε υποκλάσεις
- Αυτόματη δημιουργία χειριστών-μηνυμάτων (message-handler) για πρόσβαση της ιδιότητας
- Αλλαγή μηνύματος για απόδοση τιμής στο slot
- Περιορισμοί (constraints)

Όψη Αποθήκευσης (*storage facet*)

- **`(storage local)`**
  - Η τιμή ενός slot για κάθε αντικείμενο αποθηκεύεται στο αντικείμενο
  - Κάθε αντικείμενο μπορεί να έχει διαφορετική τιμή
- **`(storage shared)`**
  - Η τιμή του slot για κάθε αντικείμενο αποθηκεύεται στην κλάση
  - Όλα τα αντικείμενα της κλάσης έχουν την ίδια τιμή
  - Όταν αλλάζει η τιμή για ένα αντικείμενο, αλλάζει για όλα

Όψη Πρόσβασης (*access facet*)

- **`(access read-write)`**
  - Το slot μπορεί να εγγραφεί και να αναγνωστεί
- **`(access read-only)`**
  - Το slot μπορεί μόνο να αναγνωστεί.
  - Η τιμή καθορίζεται μόνο με τη χρήση του **`default`**.
  - Ένα read-only slot με **`default`** τιμή ουσιαστικά είναι **`storage shared`**
- **`(access initialize-only)`**
  - Το slot μπορεί μόνο να αναγνωστεί.
  - Η τιμή καθορίζεται μόνο μία φορά, τη στιγμή που δημιουργείται το αντικείμενο με **make-instance**

Όψη προώθησης κληρονόμησης (*inheritance propagation facet*)

- `(propagation inherit)`
  - Το slot κληρονομείται από τις υποκλάσεις αυτή της κλάσης
- **`(propagation no-inherit)`**
  - Το slot **δεν** κληρονομείται από τις υποκλάσεις αυτή της κλάσης
  - Μόνο τα στιγμιότυπα της συγκεκριμένης κλάσης θα έχουν αυτό το slot

Όψη κληρονόμησης όψεων (source facet)

- Όταν κληρονομούνται τα slots κάποιας κλάσης, κληρονομούνται και τα facets της
- **`(source exclusive)`**
  - Στην πολλαπλή κληρονομικότητα κληρονομούνται τα facets της πιο συγκεκριμένης κλάσης
- **`(source composite)`**
  - Κληρονομούνται τα facets από όλες τις κλάσεις στην ιεραρχία και όχι μόνο από την πιο συγκεκριμένη
  - Πρακτικά, μπορούμε να επανακαθορίσουμε κάποιο facet στη συγκεκριμένη κλάση χωρίς να χρειαστεί να ορίσουμε ξανά όλο το slot
  - Π.χ. όταν θέλουμε να αλλάξουμε την default τιμή κάποιου slot και όχι να την κληρονομήσουμε

Όψη δυνατότητας ταυτοποίησης (pattern-match reactivity facet)

- Οποιαδήποτε αλλαγή τιμής ενός slot συνήθως προκαλεί ενεργοποίηση κάποιου κανόνα του οποίου η συνθήκη αναφέρεται σε αντικείμενα με το συγκεκριμένο slot
  - Είναι δυνατόν αυτό να αλλάξει
- **`(pattern-match reactive)`**
  - Το slot προκαλεί ενεργοποίηση κανόνα
- **`(pattern-match non-reactive)`**
  - Το slot **δεν** προκαλεί ενεργοποίηση κανόνα

Όψη ορατότητας (visibility facet)

- Η άμεση πρόσβαση στην τιμή ενός slot (παρακάμπτοντας την αποστολή μηνυμάτων) επιτρέπεται συνήθως μόνο σε χειριστές-μηνυμάτων της συγκεκριμένης κλάσης στην οποία ορίζεται το slot
  - Είναι δυνατόν αυτό να αλλάξει
- **`(visibility private)`**
  - Άμεση πρόσβαση μόνο στη συγκεκριμένη κλάση
- **`(visibility public)`**
  - Άμεση πρόσβαση στη συγκεκριμένη κλάση και στις υποκλάσεις που κληρονομούν το slot

Όψη δημιουργίας μηνυμάτων (create-accessor facet)

- Αυτόματη δημιουργία μηνυμάτων και χειριστών-μηνυμάτων για ανάγνωση/εγγραφή τιμών στα slots
- **`(create-accessor read)`**
  - Δημιουργείται ο χειριστής μηνύματος **`get-<slot-name>`**
  - Default όταν **`(access read-only)`**
- **`(create-accessor write)`**
  - Δημιουργείται ο χειριστής μηνύματος **`put-<slot-name>`**

Όψη δημιουργίας μηνυμάτων (create-accessor facet)

- `(create-accessor read-write)`
  - Δημιουργούνται και οι get- και τα put- χειριστές μηνυμάτων
  - Default όταν `(access read-write)`
- `(create-accessor ?NONE )`
  - Δεν δημιουργείται κανένας χειριστής μηνυμάτων
  - Default όταν `(access initialize-only)`

Όψη υπερκάλυψης μηνύματος (override-message facet)

- Υπάρχουν πολλές συναρτήσεις στην COOL οι οποίες χρησιμοποιούν τα μηνύματα **`put-<slot-name>`** για να αναθέσουν τιμές στα slots των αντικειμένων
  - **`make-instance`**, **`initialize-instance`**, **`message-modify-instance`**, **`message-duplicate-instance`**
- Ο χρήστης μπορεί να αλλάξει το όνομα του χειριστή-μηνύματος που χρησιμοποιούν αυτές οι συναρτήσεις με τη χρήση του facet

`(override-message <message-name>)`

Δημιουργία Στιγμιοτύπων

- Τα αντικείμενα, όπως και τα γεγονότα
  - Δημιουργούνται από το χρήστη
  - Διαγράφονται από την εντολή **`reset`**
  - Μπορούν να φορτωθούν από αρχείο
- Τα αντικείμενα δημιουργούνται με τη βοήθεια της συνάρτησης **`make-instance`**

```
(make-instance
   [<instance-name-expression>] of <class-name-
   expression>
   (<slot-name-expression> <expression>*)*
)
```

Ταυτότητα Αντικειμένου

- Όνομα αντικειμένου (**`instance-name`**)
  - Ένα σύμβολο που περικλείεται μέσα σε αγκύλες
  - Π.χ. **`[pump-1] [foo] [+++] [123-890]`**
  - Οι αγκύλες δεν αποτελούν μέρος του ονόματος αλλά ένδειξη πως πρόκειται για όνομα αντικειμένου

Ταυτότητα Αντικειμένου

- Διεύθυνση αντικειμένου (**`instance-address`**)
  - Πρόκειται για εσωτερική αναπαράσταση της διεύθυνσης του αντικειμένου στη μνήμη
  - Όταν τυπώνεται στην οθόνη έχει τη μορφή

    `<Instance-XXX>`

    όπου XXX είναι το όνομα του αντικειμένου
  - Έχουμε πρόσβαση σε αυτή μόνο μέσα από τις συνθήκες των κανόνων με τη βοήθεια της έκφρασης

    `?var <- (object … )`
  - Εναλλακτικά μπορούμε να μετατρέψουμε το όνομα ενός αντικειμένου σε διεύθυνση με τη χρήση της συνάρτησης **`instance-address`**

Ορισμός Στιγμιότυπου

```
CLIPS>(make-instance
          fiat_brava of car
        (fuel-type benzine)
        (tank-capacity 45)
        (fuel-loaded 30)
        (consumption-rate 10)
        (reset-counter 250))
[fiat_brava]
```

Δημιουργία Στιγμιοτύπων

- Η συνάρτηση **make-instance** επιστρέφει το όνομα του νέου αντικειμένου (αν όλα παν καλά) ή το σύμβολο **FALSE** αν υπάρχει αποτυχία
- Το όνομα του αντικειμένου **<instance-name-expression>** μπορεί να είναι είτε τύπου **SYMBOL** είτε τύπου **INSTANCE-NAME**
- Αν το όνομα δεν καθορίζεται στην **make-instance** τότε η CLIPS δημιουργεί ένα όνομα καλώντας τη συνάρτηση **gensym***
- Αν το αντικείμενο υπάρχει ήδη τότε η **make-instance** το διαγράφει πρώτα και το ξαναδημιουργεί με τις νέες τιμές

Δημιουργία Στιγμιοτύπων

- Με τη βοήθεια της έκφρασης **definstances** μπορούμε να ορίσουμε αντικείμενα μέσα σε ένα αρχείο και να τα δημιουργούμε κάθε φορά που εκτελείται η εντολή **reset**

```
(definstances <definstances-name>
  [<comment>]
  ([<instance-name-expression>] of
  <class-name-expression>
     (<slot-name-expression>
  <expression>*)*
  )*
)
```

Δημιουργία Στιγμιοτύπων

- Οι κλάσεις πρέπει να έχουν ήδη οριστεί
- Αν αποτύχει η δημιουργία κάποιου αντικειμένου, τότε τα υπόλοιπα αντικείμενα αγνοούνται
- Η **`definstances`** χρησιμοποιεί εσωτερικά τη συνάρτηση **`make-instance`**

Ορισμός Στιγμιότυπου

```
(definstances car
    (fiat_brava of car
        (fuel-type benzine)
        (tank-capacity 45)
        (fuel-loaded 30)
        (consumption-rate 10)
        (reset-counter 250)
    )
)
```

Χειρισμός Στιγμιοτύπων

- Χειριζόμαστε τα αντικείμενα με αποστολή μηνυμάτων σε αυτά με τη χρήση της συνάρτησης **send**
  - Παράμετροι: Αντικείμενο-παραλήπτης μηνύματος, όνομα μηνύματος, παράμετροι μηνύματος

```
(send <object-expression>
      <message-name-expression> <expression>*)
```

- Η συνάρτηση **send** επιστρέφει ως τιμή το αποτέλεσμα του μηνύματος

Μηνύματα

```
CLIPS> (send [fiat_brava] get-fuel-type)
benzine
CLIPS> (send [fiat_brava] put-fuel-type
  petroleum)
petroleum
CLIPS> (send [fiat_brava] print)
[fiat_brava] of car
(fuel-type petroleum)
(tank-capacity 45)
(fuel-loaded 30)
(consumption-rate 10)
(reset-counter 250)
```

Χειρισμός Στιγμιοτύπων

- Για να αλλάξουμε τιμές πολλών slots ταυτόχρονα σε ένα αντικείμενο χωρίς να χρειαστεί να γράψουμε πολλές εντολές **send**, χρησιμοποιούμε τη συνάρτηση **modify-instance**

```
(modify-instance <instance>
   (<slot-name-expression>
<expression>*)*)
```

- Η συνάρτηση εσωτερικά εκτελεί πολλές συναρτήσεις **send**
- Η συνάρτηση επιστρέφει **TRUE** εάν όλες οι αλλαγές τιμών έγιναν με επιτυχία, αλλιώς επιστρέφει **FALSE**

Μηνύματα

```
CLIPS> (modify-instance [fiat_brava]
          (reset-counter 0)
          (fuel-loaded 45))
TRUE
CLIPS> (send [fiat_brava] print)
[fiat_brava] of car
(fuel-type benzine)
(tank-capacity 45)
(fuel-loaded 45)
(consumption-rate 10)
(reset-counter 0)
```

Διαγραφή Αντικειμένων

```
CLIPS> (send [fiat_brava]
  delete)
TRUE
CLIPS> (send [fiat_brava] print)
[MSGPASS2] No such instance
  fiat_brava in function send.
FALSE
```

Ταυτοποίηση Αντικειμένων σε Κανόνες

- Τα αντικείμενα των κλάσεων που ορίζει ο χρήστης μπορούν να ταυτοποιηθούν στη συνθήκη ενός κανόνα

```
(object
  (is-a <constraint>) |
  (name <constraint>) |
  (<slot-name> <constraint>*)*
)
```

Ταυτοποίηση Αντικειμένων σε Κανόνες

- Ο περιορισμός **`is-a`** χρησιμεύει για να περιορίσουμε τα αντικείμενα που ταιριάζουν στη συνθήκη σε αυτά που ανήκουν σε μία συγκεκριμένη κλάση ή υποκλάσεις της.
- Ο περιορισμός **`name`** χρησιμεύει για να περιορίσουμε την ταυτοποίηση μόνο σε συγκεκριμένα αντικείμενα ή απλά για να μας επιστραφεί σε μεταβλητή το όνομα του αντικειμένου που ταίριαξε
- Οι υπόλοιποι περιορισμοί που ισχύουν για τα πρότυπα γεγονότων, ισχύουν και για τα αντικείμενα.

Χρήση αντικειμένων σε κανόνες

```
(defrule ask-data
    (object (is-a car)
        (name ?x)
        (fuel-type benzine))
  =>
    (printout t "Car: " ?x crlf))
```

- Εκτύπωση στην οθόνη:

```
Car: [fiat_brava]
```

INSTANCE-NAME

Χρήση αντικειμένων σε κανόνες

```
defrule ask-data1
     ?y <- (object (is-a car)
                   (fuel-type benzine))
  =>
     (printout t "Car: " ?y crlf))
```

- Εκτύπωση στην οθόνη:

```
Car: <Instance-fiat_brava>
```

INSTANCE-ADDRESS

Χρήση αντικειμένων σε κανόνες

```
(defrule ask-data2
    (object (is-a vehicle)
        (name ?x)
        (fuel-type benzine))
  =>
    (printout t "Vehicle: " ?x
  crlf))
```

- Εκτύπωση στην οθόνη:

```
Vehicle: [fiat_brava]
```

INSTANCE-NAME

Χρήση αντικειμένων σε κανόνες

```
(defrule ask-data3
    (object                    ;Δεν υπάρχει is-a
        (name ?x)
        (fuel-type benzine))
  =>
    (printout t "Object: " ?x
  crlf))
```

- Εκτύπωση στην οθόνη:

```
Object: [fiat_brava]
```

INSTANCE-NAME

Ταυτοποίηση Αντικειμένων σε Κανόνες

- Όταν ένα αντικείμενο δημιουργείται ή διαγράφε-ται επηρεάζονται όλοι οι κανόνες των οποίων η συνθήκη αναφέρεται στο αντικείμενο αυτό
- Όταν αλλάζει ένα slot επηρεάζονται μόνο οι κανόνες που αναφέρονται στο συγκεκριμένο slot και όχι κανόνες που αναφέρονται στο ίδιο αντικείμενο αλλά όχι στο συγκεκριμένο slot
  - Η σπουδαιότερη διαφορά με τα πρότυπα γεγονότων (πέρα από την ύπαρξη ιεραρχίας και κληρονομικότητας)

Χρήση αντικειμένων σε κανόνες

```
(defrule stopped-at-gas-station
  (goal fuel-reload)
  (object (is-a car)
     (name ?x)
     (tank-capacity ?c))
=>
  (send ?x put-reset-counter 0)
  (send ?x put-fuel-loaded ?c))
```

Χρήση αντικειμένων σε κανόνες

```
(defrule stopped-at-gas-station
  (goal fuel-reload)
  (object (is-a car)
     (name ?x)
     (tank-capacity ?c))
=>
  (modify-instance ?x
     (reset-counter 0)
     (fuel-loaded ?c)))
```

Διαφορά με templates

```
(defrule stopped-at-gas-station
  (goal fuel-reload)
  ?y <- (car   (name ?x)
               (tank-capacity ?c))
=>
  (modify ?y   (reset-counter 0)
               (fuel-loaded ?c)))
```

- Ο κανόνας θα εκτελούνταν επ’ άπειρο γιατί η αλλαγή κάποιων slots προκαλεί την διαγραφή του γεγονότος και την εισαγωγή νέου, με αποτέλεσμα το CLIPS να «νομίζει» ότι πρόκειται για ενεργοποίηση του κανόνα με νέα δεδομένα!

Ερωτήσεις και Ενέργειες σε Ομάδες Αντικειμένων

- Η COOL δίνει τη δυνατότητα εκτέλεσης ερωτήσεων και ενεργειών σε πολλά αντικείμενα μαζί βάσει συγκεκριμένων κριτηρίων αναζήτησης που θέτει ο χρήστης
  - Η δυνατότητα αυτή θυμίζει γλώσσα ερωταπαντήσεων (query language) των βάσεων δεδομένων (π.χ. SQL)
- Υπάρχουν 6 συναρτήσεις
  - Σε όλες, ο τρόπος με τον οποίο προσδιορίζεται η ομάδα των αντικειμένων στην οποία θα ενεργήσει η εντολή είναι κοινός

Ερωτήσεις και Ενέργειες σε Ομάδες Αντικειμένων

Συνάρτηση	Σκοπός
`any-instancep`	Ελέγχει αν υπάρχει έστω και μία ομάδα αντικειμένων που ικανοποιεί μια συνθήκη
`find-instance`	Επιστρέφει την πρώτη ομάδα αντικειμένων που ικανοποιεί μια συνθήκη
`find-all-instances`	Επιστρέφει όλες τις ομάδες αντικειμένων που ικανοποιούν μια συνθήκη

Ερωτήσεις και Ενέργειες σε Ομάδες Αντικειμένων

Συνάρτηση	Σκοπός
`do-for-instance`	Εκτελεί ένα σύνολο ενεργειών πάνω στην πρώτη ομάδα αντικειμένων που ικανοποιεί μια συνθήκη
`do-for-all-instances`	Εκτελεί ένα σύνολο ενεργειών πάνω σε κάθε ομάδα αντικειμένων που ικανοποιεί μια συνθήκη
`delayed-do-for-all-instances`	Πρώτα βρίσκει όλες τις ομάδες αντικειμένων που ικανοποιούν μια συνθήκη και στη συνέχεια εκτελεί ένα σύνολο ενεργειών σε αυτές

Συνάρτηση `any-instancep`

- Εφαρμόζει μια ερώτηση-συνθήκη σε κάθε ομάδα αντικειμένων που ταιριάζει με το πρότυπο που ορίζει ο χρήστης
  - Εάν υπάρχει κάποια ομάδα αντικειμένων που ικανοποιεί τη συνθήκη, τότε τερματίζει **αμέσως** επιστρέφοντας **`TRUE`**
  - Αλλιώς επιστρέφει **`FALSE`**

Συνάρτηση any-instancep

- **Σύνταξη**

```
(any-instancep
      <instance-set-template>
      <query>)
```

- **Παράδειγμα**
  - Υπάρχουν άνδρες ηλικίας άνω των 30?

```
CLIPS> (any-instancep
            ((?man MAN))          πρότυπο
            (> ?man:age 30))      συνθήκη
TRUE
```

Συνάρτηση `find-instance`

- Εφαρμόζει μια ερώτηση-συνθήκη σε κάθε ομάδα αντικειμένων που ταιριάζει με το πρότυπο που ορίζει ο χρήστης
  - Εάν υπάρχει κάποια ομάδα αντικειμένων που ικανοποιεί τη συνθήκη, τότε τερματίζει **αμέσως** επιστρέφοντας μία λίστα με αυτήν την ομάδα των αντικειμένων
  - Αλλιώς επιστρέφει μία λίστα μηδενικού μήκους

Συνάρτηση find-instance

- **Σύνταξη**

```
(find-instance
      <instance-set-template>
      <query>)
```

- **Παράδειγμα**
  - Βρες το πρώτο ζευγάρι ενός άνδρα και μιας γυναίκας με την ίδια ηλικία

```
CLIPS>(find-instance
            ((?m MAN) (?w WOMAN))
            (= ?m:age ?w:age))
([Man-1] [Woman-1])
```

πρότυπο

συνθήκη

Συνάρτηση `find-all-instances`

- Εφαρμόζει μια ερώτηση-συνθήκη σε κάθε ομάδα αντικειμένων που ταιριάζει με το πρότυπο που ορίζει ο χρήστης
  - Κάθε ομάδα αντικειμένων που ικανοποιεί τη συνθήκη αποθηκεύεται σε μία λίστα
  - Όταν εξαντληθούν όλες οι πιθανές ομάδες αντικειμένων, επιστρέφεται αυτή η λίστα

Συνάρτηση `find-all-instances`

- **Σύνταξη**

```
(find-all-instances
      <instance-set-template> <query>)
```

- **Παράδειγμα**
  - Βρες όλα τα ζευγάρια ενός άνδρα και μιας γυναίκας με την ίδια ηλικία

```
CLIPS> (find-all-instances
           ((?m MAN) (?w WOMAN))      πρότυπο
           (= ?m:age ?w:age))         συνθήκη
([Man-1] [Woman-1] [Man-2] [Woman-2])
```

Συνάρτηση `find-all-instances`

- Αν υπάρχουν ***n*** στιγμιότυπα σε κάθε ομάδα αντικειμένων και ***m*** ομάδες αντικειμένων που ικανοποιούν τη συνθήκη, τότε το μήκος της επιστρεφόμενης λίστας θα είναι ***n*** * ***m***
  - Τα πρώτα ***n*** στοιχεία ανήκουν στην πρώτη ομάδα αντικειμένων, κ.ο.κ.
  - Πρέπει να χρησιμοποιείται με προσοχή γιατί καταναλώνει πολύ μνήμη λόγω συνδυαστικής έκρηξης

Συνάρτηση `do-for-all-instances`

- Εφαρμόζει μια ερώτηση-συνθήκη σε κάθε ομάδα αντικειμένων που ταιριάζει με το πρότυπο που ορίζει ο χρήστης
  - Αν μία ομάδα αντικειμένων ικανοποιεί τη συνθήκη, τότε ένα σύνολο ενεργειών εκτελείται για αυτήν την ομάδα αντικειμένων
  - Το αποτέλεσμα της συνάρτησης είναι η τιμή της ενέργειας που εκτελείται για την τελευταία ομάδα αντικειμένων που ικανοποιεί τη συνθήκη
  - Αν καμία ομάδα αντικειμένων δεν ικανοποιεί τη συνθήκη, τότε επιστρέφει **`FALSE`**

Συνάρτηση do-for-all-instances

- **Σύνταξη**

```
(do-for-all-instances
  <instance-set-template>
  <query>
  <action>*)
```

Συνάρτηση do-for-all-instances

- **Παράδειγμα**
  - Τύπωσε όλες τις τριπλέτες διαφορετικών ανθρώπων οι οποίοι έχουν την ίδια ηλικία.

```
CLIPS> (do-for-all-instances
πρότυπο    ((?p1 PERSON)(?p2 PERSON)(?p3 PERSON))
συνθήκη    (= ?p1:age ?p2:age ?p3:age)
ενέργεια   (printout t ?p1 " " ?p2 " " ?p3 crlf))
[Girl-2] [Boy-3] [Boy-2]
[Girl-2] [Boy-4] [Boy-2]
[Girl-2] [Boy-4] [Boy-3]
[Boy-4] [Boy-3] [Boy-2]
...
```

Συνάρτηση do-for-all-instances

- Για να μην τυπώνονται τριπλέτες με 2 ή περισσότερα ίδια αντικείμενα

```
(do-for-all-instances
    ((?p1 PERSON)(?p2 PERSON)(?p3
PERSON))
    (and (= ?p1:age ?p2:age ?p3:age)
         (neq ?p1 ?p2)(neq ?p1 ?p3)(neq
?p2 ?p3))
    (printout t ?p1 " " ?p2 " " ?p3
crlf))
```

Συνάρτηση do-for-all-instances

- Για να μην τυπώνονται οι ίδιες τριπλέτες με όλους τις πιθανές διατάξεις των αντικειμένων

```
(do-for-all-instances
    ((?p1 PERSON)(?p2 PERSON)(?p3
PERSON))
    (and (= ?p1:age ?p2:age ?p3:age)
         (> (str-compare ?p1 ?p2) 0)
         (> (str-compare ?p2 ?p3) 0))
    (printout t ?p1 " " ?p2 " " ?p3
crlf))
```

Ορισμός Μεθόδων

- Για τον καθορισμό της συμπεριφοράς μίας κλάσης αντικειμένων σε απάντηση προς ένα συγκεκριμένο μήνυμα χρησιμοποιείται η ειδική συνάρτηση **`defmessage-handler`**
- Η υλοποίηση ενός μηνύματος αποτελείται από τμήματα διαδικαστικού κώδικα (procedural code), τα οποία ονομάζονται μέθοδοι ή χειριστές-μηνυμάτων (message-handlers)

Ορισμός Μεθόδων

- Στη λίστα προτεραιότητας κλάσεων μίας κλάσης, κάθε κλάση μπορεί να έχει μεθόδους για ένα μήνυμα
- Με αυτόν τον τρόπο, η κλάση και όλες οι υπερκλάσεις της μοιράζονται την εργασία του χειρισμού ενός μηνύματος
- Οι μέθοδοι κάθε κλάσης χειρίζονται εκείνο το τμήμα του μηνύματος που προορίζεται για αυτήν την κλάση

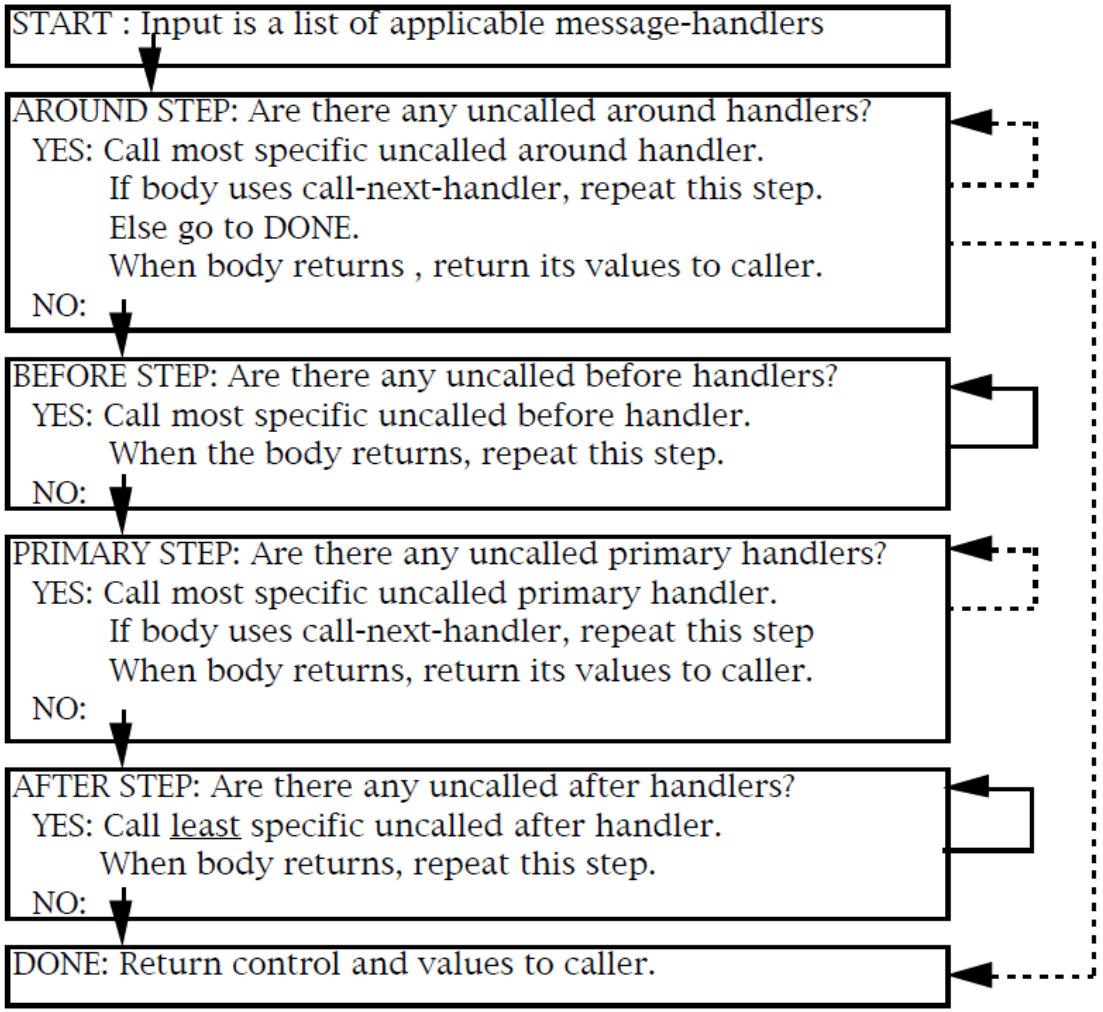
Κατηγορίες Μεθόδων (1/2)

- *primary*
  - Αποδίδουν την επιστρεφόμενη τιμή ενός μηνύματος
- *before*, *after*
  - Είναι βοηθητικές και χρησιμοποιούνται μόνο για παρενέργειες
  - Οι τιμές που επιστρέφουν αγνοούνται
  - Οι before μέθοδοι εκτελούνται πριν τις primary
  - Οι after μέθοδοι εκτελούνται μετά τις primary

Κατηγορίες Μεθόδων (2/2)

- *around*
  - Μπορούν να επιστρέψουν μία τιμή
  - Δημιουργούν ένα περιβάλλον για την εκτέλεση των υπόλοιπων μεθόδων
  - Ξεκινούν την εκτέλεση πριν από τις άλλες μεθόδους και συνεχίζουν αφού τελειώσουν όλες οι υπόλοιπες μέθοδοι

Εκτέλεση Μεθόδων



Συνάρτηση `defmessage-handler`

1. το όνομα μίας κλάσης στην οποία θα αποδοθεί η μέθοδος (πρέπει προηγουμένως να έχει οριστεί η κλάση)
2. το όνομα του μηνύματος στο οποίο η μέθοδος θα αποκριθεί
3. ένας προαιρετικός τύπος (η προκαθορισμένη τιμή είναι primary)
4. ένα προαιρετικό σχόλιο

Συνάρτηση defmessage-handler

5. μία λίστα παραμέτρων που θα περάσουν στη μέθοδο κατά τη διάρκεια της εκτέλεσης
6. μία προαιρετική παράμετρος, η οποία αφορά τις μεταβλητές πολλαπλών τιμών
7. μία σειρά εκφράσεων οι οποίες εκτελούνται στη σειρά όταν καλείται η μέθοδος.
   - Η επιστρεφόμενη τιμή της μεθόδου είναι ο υπολογισμός της τελευταίας έκφρασης.

Παράδειγμα Ορισμού Μεθόδου

```
                                1              2
(defmessage-handler car remaining-distance
                                                ()     5
      (- (* (/ ?self:fuel-loaded
               ?self:consumption-rate)
            100)
         ?self:reset-counter)                          7
)

CLIPS>(send [fiat_brava] remaining-distance)
50.0
```

Παράδειγμα Ορισμού Μεθόδου

- Για να χρησιμοποιηθεί η έκφραση `?self:fuel-loaded` στη μέθοδο **`remaining-distance`** πρέπει να οριστεί το slot **`fuel-loaded`** ως `public` στην κλάση **`vehicle`**

```
(defclass vehicle
     (is-a USER)
     (slot fuel-type (type SYMBOL))
     (slot tank-capacity (type INTEGER))
     (slot fuel-loaded
          (type INTEGER)
          (visibility public)))
```

Παράδειγμα Ορισμού Μεθόδου

- Αν δεν οριστεί public, τότε πρέπει να γίνει κλήση με αποστολή μηνύματος

```
(defmessage-handler car remaining-distance
                                        ()
      (- (* (/ (send ?self get-fuel-loaded)
               ?self:consumption-rate)
            100)
         ?self:reset-counter)
)
```

ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Τέλος Ενότητας

Επεξεργασία: Εμμανουήλ Ρήγας

Θεσσαλονίκη, 17/3/2014

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης