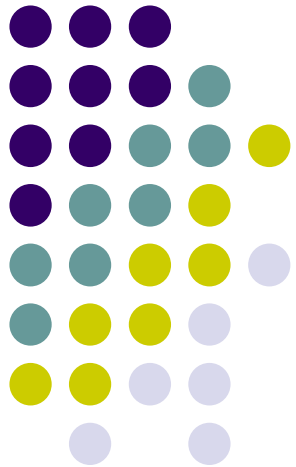
Προσεγγίζοντας τον Αντικειμενοστρεφή Προγραμματισμό μέσα από τη ΓΛΩΣΣΑ



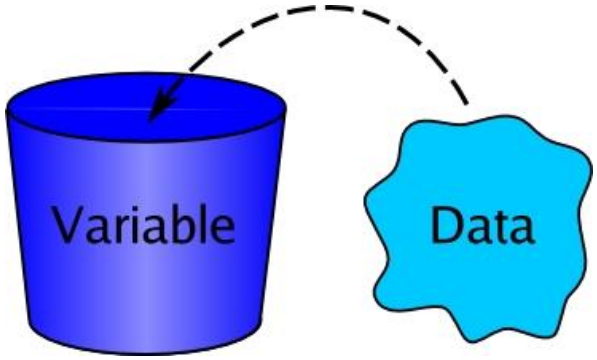
Ματθές Δημήτριος
38ο ΓΕΛ Αθηνών

ΓΛΩΣΣΑ

- Υποθετική γλώσσα προγραμματισμού
- Δημιουργήθηκε για διδακτικούς σκοπούς για τη διδασκαλία του μαθήματος ΑΕΠΠ
- Αποτελεί μετάφραση της γλώσσας προγραμματισμού PASCAL στα ελληνικά

Χαρακτηριστικά της ΓΛΩΣΣΑΣ

- Διαθέτει 4 τύπους μεταβλητών:
  - Ακέραιες
  - Πραγματικές
  - Λογικές
  - Χαρακτήρες (αλφαριθμητικές)
- Χρησιμοποιεί 2 τύπους υποπρογραμμάτων:
  - Διαδικασίες
  - Συναρτήσεις



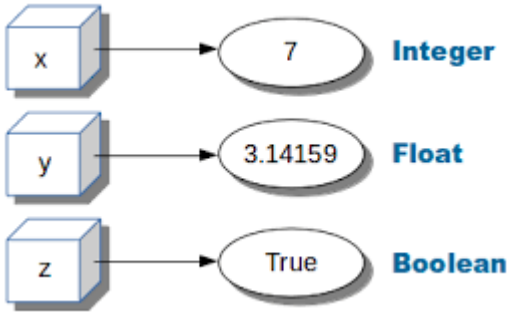
procedure function
function procedure
procedure function
function procedure
procedure function
function procedure
procedure function
function procedure

Χαρακτηριστικά της ΓΛΩΣΣΑΣ

- Εξαιτίας του τρόπου που δημιουργούνται και εκτελούνται τα προγράμματα γραμμένα ΓΛΩΣΣΑ την κατατάσσουμε στις **διαδικασιακές** (procedural) γλώσσες.
- Υποστηρίζει τον **δομημένο προγραμματισμό** (δομή ακολουθίας, επιλογής, επανάληψης) καθώς και τη **στατική δομή δεδομένων** που ονομάζεται **πίνακας**.

Δήλωση μεταβλητών στη ΓΛΩΣΣΑ

- Όσοι έχουν ασχοληθεί με τη ΓΛΩΣΣΑ γνωρίζουν ότι η δήλωση των μεταβλητών γίνεται με τρόπο όπως στο ακόλουθο παράδειγμα:



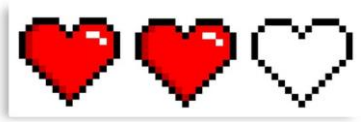

```
ΜΕΤΑΒΛΗΤΕΣ
   ΑΚΕΡΑΙΕΣ: χ, ψ, βαθμοί[30, 10]
   ΠΡΑΓΜΑΤΙΚΕΣ: α, β, μο[20]
   ΛΟΓΙΚΕΣ: βρέθηκε
   ΧΑΡΑΚΤΗΡΕΣ: όνομα, επώνυμο
```

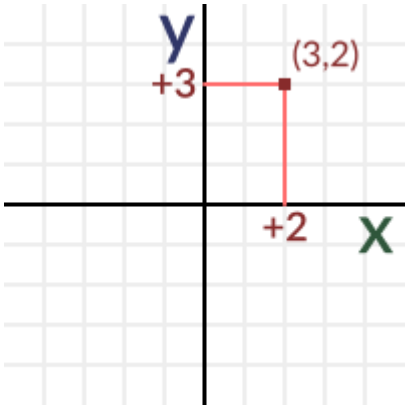
Δημιουργία ενός παιχνιδιού στη ΓΛΩΣΣΑ

- Προτού προσεγγίσουμε τον αντικειμενοστρεφή προγραμματισμό, ας θεωρήσουμε ότι θέλουμε να δημιουργήσουμε ένα παιχνίδι στη ΓΛΩΣΣΑ.
- Το παιχνίδι αυτό θα παίζεται από 2 παίκτες.

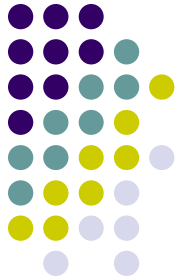
Δημιουργία ενός παιχνιδιού στη ΓΛΩΣΣΑ

- Κάθε παίκτης θα έχει τις εξής ιδιότητες / χαρακτηριστικά:
  - Έναν αριθμό «ζωών». Ο παίκτης που θα φτάσει πρώτος στις 0 ζωές θα χάνει και θα κερδίζει ο αντίπαλός του.
  - Ένα όνομα.
  - Ένα σκορ.
  - Ένα ζεύγος συντεταγμένων (χ, ψ) οι οποίες εκφράζουν το σημείο που ο παίκτης βρίσκεται στον δισδιάστατο χώρο στον οποίο κινείται.





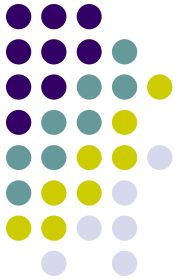
Δημιουργία ενός παιχνιδιού στη ΓΛΩΣΣΑ



- Επιπλέον, θα υπάρχει και μία διαδικασία με το όνομα **ΆλλαξεΣκορ(α)** η οποία θα δέχεται ως παράμετρο το σκορ ενός παίκτη και θα το αυξάνει κατά 1.
- Η δήλωση των προηγούμενων μεταβλητών σε ένα τέτοιο παιχνίδι θα μπορούσε να γίνει ως εξής:

```
ΜΕΤΑΒΛΗΤΕΣ
   ΑΚΕΡΑΙΕΣ: ζωές1, ζωές2, σκορ1, σκορ2
   ΠΡΑΓΜΑΤΙΚΕΣ: χ1, ψ1, χ2, ψ2
   ΧΑΡΑΚΤΗΡΕΣ: όνομα1, όνομα2
```

Δημιουργία ενός παιχνιδιού στη ΓΛΩΣΣΑ



- Ομοίως, μετά το τέλος του κυρίου προγράμματος, θα δημιουργούσαμε τη διαδικασία:

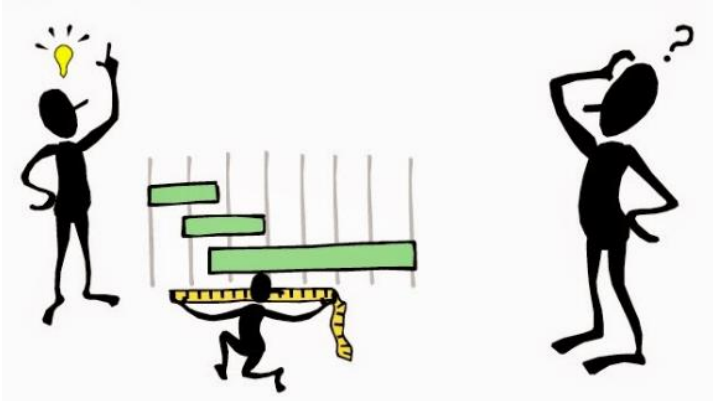
```
ΔΙΑΔΙΚΑΣΙΑ ΆλλαξεΣκορ(α)
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: α
ΑΡΧΗ
  α ← α + 1
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Δημιουργία ενός παιχνιδιού στη ΓΛΩΣΣΑ

- Ο απλός και διαδικασιακός αυτός τρόπος προγραμματισμού είναι αποδεκτός αφού υλοποιείται σχετικά εύκολα εφόσον οι παίκτες είναι λίγοι.
- Αναρωτιέται όμως κανείς τι θα γινόταν στην περίπτωση που οι παίκτες ήταν περισσότεροι (π.χ. 10)!
- Θα χρειαζόταν να δηλώσουμε τον αριθμό των παραπάνω μεταβλητών x10 φορές ή να χρησιμοποιήσουμε πίνακες.

Δημιουργία ενός παιχνιδιού στη ΓΛΩΣΣΑ

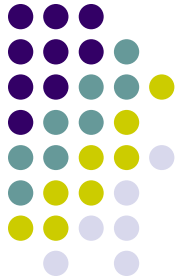
- Οι **ιδιότητες** κάθε παίκτη (μεταβλητές για τις ζωές, τις συντεταγμένες, τα σκορ, τα ονόματα) είναι «ανακατεμένες» μεταξύ τους και συνυπάρχουν όλες στο τμήμα δηλώσεων.
- Επιπλέον, η διαδικασία για την αλλαγή του σκορ βρίσκεται μετά το τέλος του κυρίου προγράμματος. Αν είχαμε κι άλλες διαδικασίες εκεί τότε θα ήταν «χαμένη» κάπου ανάμεσα σε αυτές.



Δημιουργία ενός παιχνιδιού στη ΓΛΩΣΣΑ

- Θα ήταν βολικό αν μπορούσαμε:
  - Να «ομαδοποιήσουμε» τις μεταβλητές και τα υποπρογράμματα που αφορούν τον κάθε παίκτη.
  - Να δημιουργήσουμε πολύ εύκολα και γρήγορα τους παίκτες του παιχνιδιού, ανεξάρτητα από τον αριθμό τους!

Η έννοια του αντικειμενοστρεφούς προγραμματισμού

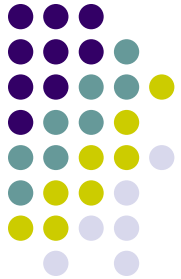


- Τη λύση στο πρόβλημα αυτό έρχεται να μας δώσει ο **αντικειμενοστρεφής προγραμματισμός**.
- Η βασική λογική του είναι ότι ορίζουμε μία **κλάση** (δηλαδή ένα γενικό πρότυπο) η οποία περιγράφει τις **ιδιότητες** (μεταβλητές) και τις **μεθόδους** (υποπρογράμματα) ενός **αντικειμένου**.

Η έννοια του αντικειμενοστρεφούς προγραμματισμού

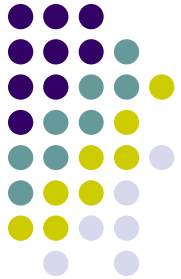
- Από αυτή την γενική **κλάση** μπορούμε να δημιουργήσουμε **αντικείμενα**.
- Π.χ. στην περίπτωση του παιχνιδιού μας θα μπορούσαμε να ορίσουμε μία **κλάση** με το όνομα **Παίκτης** η οποία περιγράφει τις μεταβλητές και τη διαδικασία που αναφέραμε προηγουμένως.
- Στη συνέχεια, βάσει της **κλάσης** αυτής, θα μπορούσαμε να δημιουργήσουμε τα **αντικείμενά** μας (π.χ. Παίκτης1, Παίκτης2 κλπ.)

Αντικειμενοστρεφής Προγραμματισμός και ΓΛΩΣΣΑ



- Δυστυχώς, η ΓΛΩΣΣΑ δεν διαθέτει χαρακτηριστικά αντικειμενοστρεφούς προγραμματισμού.
- Μπορούμε όμως να τον προσεγγίσουμε σε υποθετικό επίπεδο, βασιζόμενοι στην λογική που χρησιμοποιούν άλλες γλώσσες προγραμματισμού.

Αντικειμενοστρεφής Προγραμματισμός και ΓΛΩΣΣΑ



- **-ΑΝ-** λοιπόν υπήρχε ο αντικειμενοστρεφής προγραμματισμός στην ΓΛΩΣΣΑ, θα είχαμε τη δυνατότητα λίγο μετά το κυρίως πρόγραμμα να δημιουργήσουμε μία **κλάση** με τον ακόλουθο τρόπο:

Αντικειμενοστρεφής Προγραμματισμός και ΓΛΩΣΣΑ

Αρχή και τέλος της κλάσης

```
ΚΛΑΣΗ Παίκτης

  ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: ζωές, σκορ
    ΠΡΑΓΜΑΤΙΚΕΣ: χ, ψ
    ΧΑΡΑΚΤΗΡΕΣ: όνομα

  ΔΙΑΔΙΚΑΣΙΑ ΆλλαξεΣκορ
  ΑΡΧΗ
    σκορ ← σκορ + 1
  ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

ΤΕΛΟΣ_ΚΛΑΣΗΣ
```

Τμήμα δήλωσης των ιδιοτήτων (μεταβλητών) της κλάσης

Προσθήκη μιας νέας μεθόδου (διαδικασίας) στην κλάση

Σημείωση: Οι ιδιότητες της κλάσης είναι ορατές από τις μεθόδους που υπάρχουν σε αυτή!

Αντικειμενοστρεφής Προγραμματισμός και ΓΛΩΣΣΑ

- Η κλάση **Παίκτης** θα μπορούσε να απεικονιστεί γραφικά ως εξής:

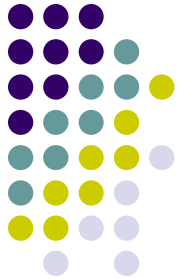
ΠΑΙΚΤΗΣ
ΑΚΕΡΑΙΕΣ: ζωές, σκορ **ΠΡΑΓΜΑΤΙΚΕΣ:** χ, ψ **ΧΑΡΑΚΤΗΡΕΣ:** όνομα
ΔΙΑΔΙΚΑΣΙΑ ΆλλαξεΣκορ

Αντικειμενοστρεφής Προγραμματισμός και ΓΛΩΣΣΑ

- Με την παραπάνω λογική, στο κυρίως πρόγραμμα και στο τμήμα δηλώσεων απλώς θα γράφαμε:

```
ΠΡΟΓΡΑΜΜΑ Παιχνίδι
ΜΕΤΑΒΛΗΤΕΣ
   ΠΑΙΚΤΗΣ: Παίκτης1, Παίκτης2
ΑΡΧΗ
!Εδώ μπαίνει ο κώδικας του παιχνιδιού
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

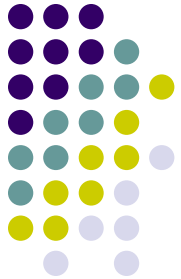
Αντικειμενοστρεφής Προγραμματισμός και ΓΛΩΣΣΑ



- Παρατηρούμε ότι αντί για τους γνωστούς τύπους των μεταβλητών, δηλώνουμε την κλάση **Παίκτης**. Βάσει αυτής δημιουργούμε 2 αντικείμενα με ονόματα **Παίκτης1** και **Παίκτης2**.
- Θα μπορούσαμε λοιπόν να θεωρήσουμε την κλάση **Παίκτης** ως έναν νέο τύπο δεδομένων και τα αντικείμενα **Παίκτης1** και **Παίκτης2** τις αντίστοιχες μεταβλητές του.

```
ΠΡΟΓΡΑΜΜΑ Παιχνίδι
ΜΕΤΑΒΛΗΤΕΣ
   ΠΑΙΚΤΗΣ: Παίκτης1, Παίκτης2
ΑΡΧΗ
!Εδώ μπαίνει ο κώδικας του παιχνιδιού
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

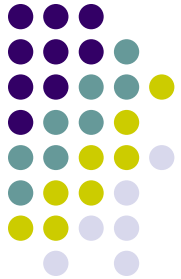
Αντικειμενοστρεφής Προγραμματισμός και ΓΛΩΣΣΑ



- Αν θέλουμε να θέσουμε τιμές στις ιδιότητες του Παίκτη1, το μόνο που έχουμε να κάνουμε είναι:

```
ΠΡΟΓΡΑΜΜΑ Παιχνίδι
ΜΕΤΑΒΛΗΤΕΣ
   ΠΑΙΚΤΗΣ: Παίκτης1, Παίκτης2
ΑΡΧΗ
   Παίκτης1.όνομα ← 'Γιώργος'
   Παίκτης1.ζωές ← 3
   Παίκτης1.σκορ ← 0
   Παίκτης1.χ ← 10.54
   Παίκτης1.ψ ← 20.33
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Αντικειμενοστρεφής Προγραμματισμός και ΓΛΩΣΣΑ



- Ομοίως, αν προσθέσουμε τιμές και στις ιδιότητες του Παίκτη2:

```
ΠΡΟΓΡΑΜΜΑ Παιχνίδι
ΜΕΤΑΒΛΗΤΕΣ
  ΠΑΙΚΤΗΣ: Παίκτης1, Παίκτης2
ΑΡΧΗ
  Παίκτης1.όνομα ← 'Γιώργος'
  Παίκτης1.ζωές ← 3
  Παίκτης1.σκορ ← 0
  Παίκτης1.χ ← 10.54
  Παίκτης1.ψ ← 20.33
  Παίκτης2.όνομα ← 'Νίκος'
  Παίκτης2.ζωές ← 3
  Παίκτης2.σκορ ← 0
  Παίκτης2.χ ← 60.89
  Παίκτης2.ψ ← 45.17
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Αντικειμενοστρεφής Προγραμματισμός και ΓΛΩΣΣΑ

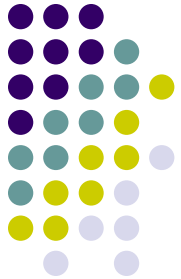
- Η εντολή:

```
ΓΡΑΨΕ 'Οι ζωές του Παίκτη 1 είναι: ', Παίκτης1.ζωές
```

θα εμφανίσει στην οθόνη:

```
Οι ζωές του Παίκτη 1 είναι: 3
```

Αντικειμενοστρεφής Προγραμματισμός και ΓΛΩΣΣΑ



- Ενώ αν στη συνέχεια εκτελεστούν μία φορά και οι επόμενες εντολές:

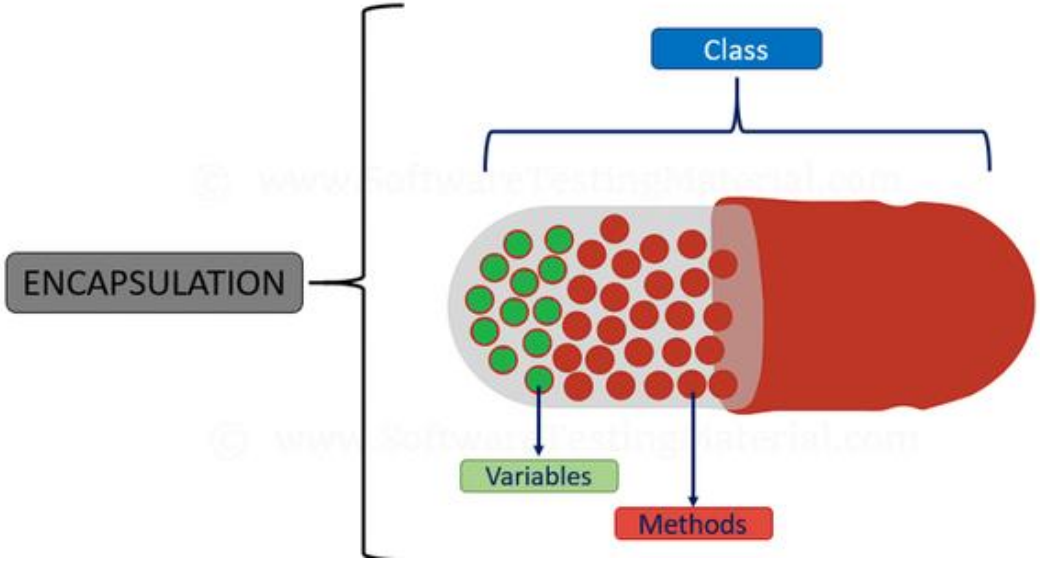
```
ΚΑΛΕΣΕ Παίκτης2.ΆλλαξεΣκορ
ΓΡΑΨΕ 'Το σκορ του Παίκτη 1 είναι: ', Παίκτης1.σκορ
ΓΡΑΨΕ 'Το σκορ του Παίκτη 2 είναι: ', Παίκτης2.σκορ
```

θα εμφανιστεί στην οθόνη:

```
Το σκορ του Παίκτη 1 είναι: 0
Το σκορ του Παίκτη 2 είναι: 1
```

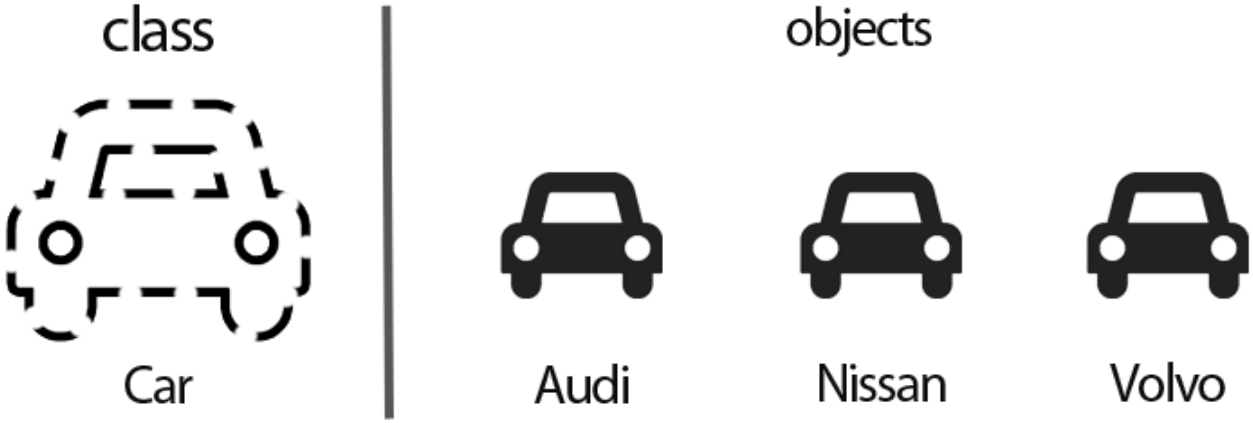
Ενθυλάκωση

- **Ενθυλάκωση** (encapsulation) είναι η δυνατότητα μιας **κλάσης** να συνδυάζει εσωτερικά **ιδιότητες** (μεταβλητές) και **μεθόδους** (υποπρογράμματα).



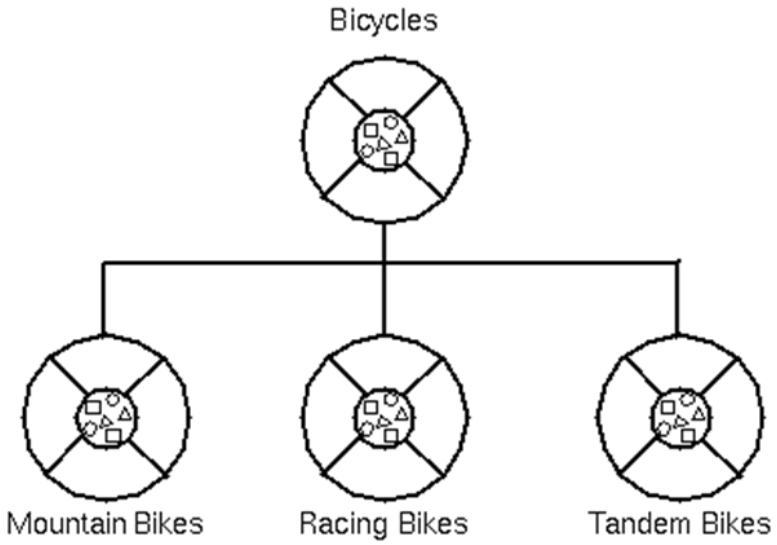
Ενθυλάκωση

- Έτσι, μπορούμε να πούμε ότι μία **κλάση** αποτελεί έναν «**αφαιρετικό τύπο δεδομένων»** (abstract) και μπορεί να παράγει ένα απεριόριστο πλήθος δομικά ίδιων **αντικειμένων**.



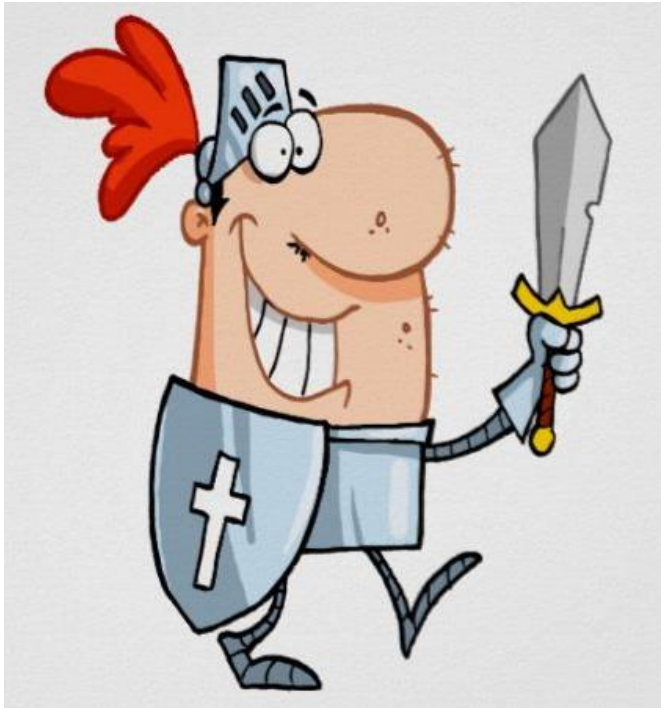
Κληρονομικότητα

- Υπάρχουν όμως περιπτώσεις που θέλουμε να δημιουργήσουμε μία νέα κλάση η οποία βασίζεται σε μία άλλη κλάση.
- Η νέα αυτή κλάση αφενός **κληρονομεί** τα στοιχεία της αρχικής κλάσης και αφετέρου προσθέτει νέες ιδιότητες και μεθόδους στα παραγόμενα αντικείμενα.
- Η νέα κλάση (**απόγονος**) αποτελεί στην ουσία μια «επέκταση» της δομής της αρχικής κλάσης (**πρόγονος**).



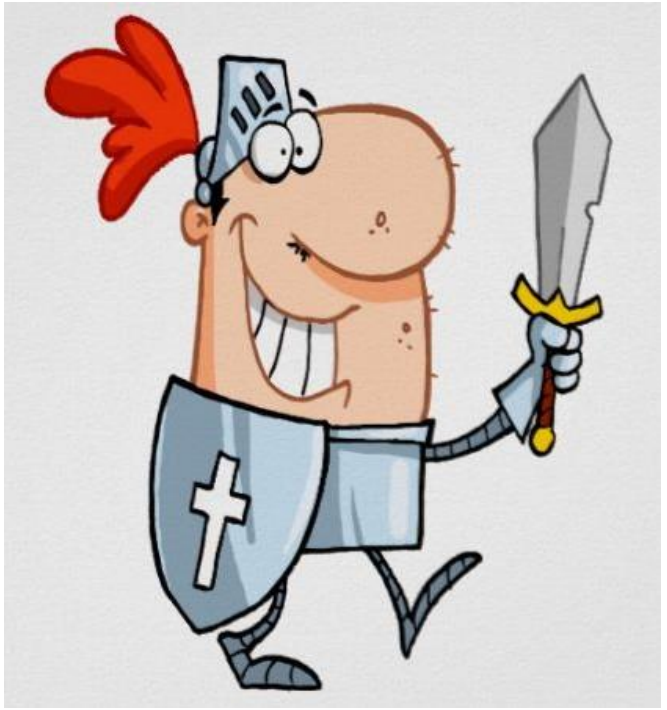
Κληρονομικότητα

- Ας δούμε πάλι το παιχνίδι στη ΓΛΩΣΣΑ.
- Ας υποθέσουμε ότι θέλουμε να έχουμε έναν νέο τύπο παίκτη ο οποίος διατηρεί τα χαρακτηριστικά της κλάσης **Παίκτης** αλλά παράλληλα είναι και **Ιππότης**, δηλαδή κρατάει κάποιες φορές κι ένα σπαθί.
- Επιπλέον, θέλουμε με κάποιο τρόπο να εμφανίζει κι ένα μήνυμα σχετικά με το αν κρατάει ή όχι το σπαθί.

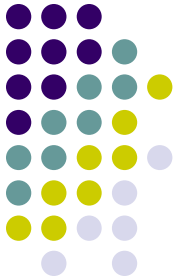


Κληρονομικότητα

- Για το γεγονός αν κρατάει ή όχι το σπαθί μπορούμε να χρησιμοποιήσουμε μία λογική μεταβλητή με όνομα **ΚρατάειΣπαθί**.
- Για το αντίστοιχο μήνυμα μπορούμε να καλούμε μία διαδικασία με όνομα **ΔείξεΟπλισμό** η οποία θα μας αναφέρει την κατάσταση του οπλισμού του ιππότη κάθε φορά.
- Για να γίνουν όλα αυτά πράξη θα προσθέταμε μετά το κυρίως πρόγραμμα και την κλάση **Παίκτης**, την επόμενη κλάση:



Κληρονομικότητα



```
ΚΛΑΣΗ Ιππότης : Παίκτης

  ΜΕΤΑΒΛΗΤΕΣ
    ΛΟΓΙΚΕΣ: ΚρατάειΣπαθί

  ΔΙΑΔΙΚΑΣΙΑ ΔείξεΟπλισμό
  ΑΡΧΗ
    ΑΝ ΚρατάειΣπαθί = ΑΛΗΘΗΣ ΤΟΤΕ
      ΓΡΑΨΕ 'Ο ιππότης κρατάει σπαθί'
    ΑΛΛΙΩΣ
      ΓΡΑΨΕ 'Ο ιππότης δεν κρατάει σπαθί'
    ΤΕΛΟΣ_ΑΝ
  ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

ΤΕΛΟΣ_ΚΛΑΣΗΣ
```

Η νέα κλάση **Ιππότης** βασίζεται στην κλάση **Παίκτης** και χρησιμοποιεί τις ιδιότητες και τις μεθόδους της

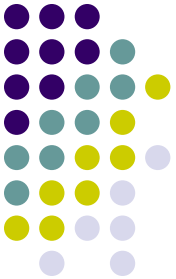
Προσθήκη ιδιότητας

Προσθήκη μεθόδου

Αρχή και τέλος της κλάσης

<u>Σημείωση</u>: Οι ιδιότητες της κλάσης είναι ορατές από τις μεθόδους που υπάρχουν σε αυτή!

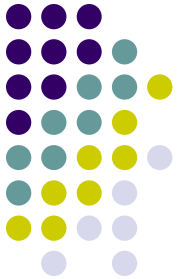
Κληρονομικότητα



- Αν τώρα λοιπόν δηλώναμε στην αρχή του προγράμματος τον παίκτη ως **Ιππότη** τότε θα μπορούσαμε να προσπελάσουμε όλες τις παλιές ιδιότητες του **Παίκτη** αλλά και τις νέες του **Ιππότη**:

```
ΠΡΟΓΡΑΜΜΑ Παιχνίδι
ΜΕΤΑΒΛΗΤΕΣ
   ΙΠΠΟΤΗΣ: Ιππότης1
ΑΡΧΗ
   Ιππότης1.όνομα ← 'Γιώργος'
   Ιππότης1.ζωές ← 3
   Ιππότης1.σκορ ← 0
   Ιππότης1.χ ← 10.54
   Ιππότης1.ψ ← 20.33
   Ιππότης1.ΚρατάειΣπαθί ← ΨΕΥΔΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Κληρονομικότητα



- Οι εντολές:

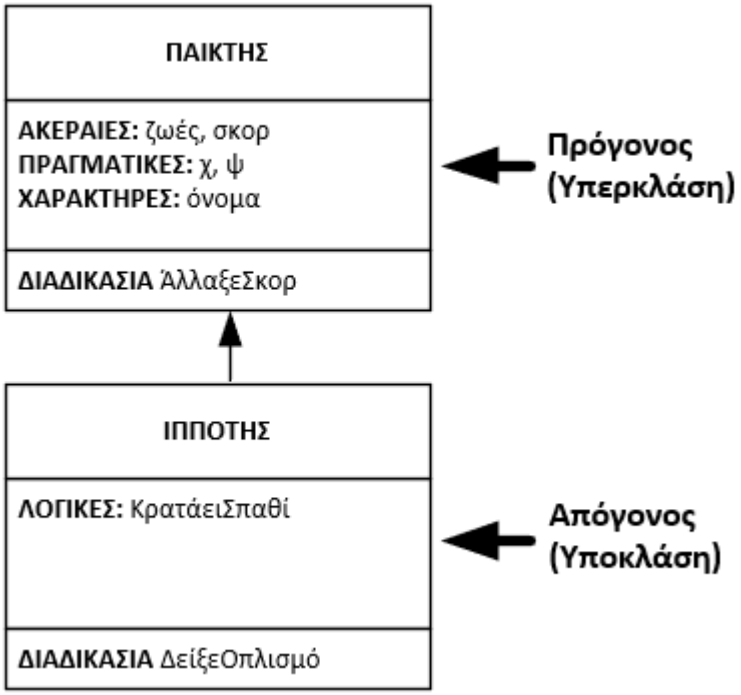
```
Ιππότης1.ΚρατάειΣπαθί ← ΑΛΗΘΗΣ
ΚΑΛΕΣΕ Ιππότης1.ΔείξεΟπλισμό
```

θα εμφανίσουν στην οθόνη το μήνυμα:

```
Ο ιππότης κρατάει σπαθί
```

Κληρονομικότητα

- Η δυνατότητα δημιουργίας ιεραρχιών αντικειμένων καλείται **κληρονομικότητα** (inheritance).
- Με βάση την κληρονομικότητα, μια κλάση μπορεί να περιγραφεί γενικά και στη συνέχεια μέσω αυτής της κλάσης μπορούν να οριστούν διάφορες υποκλάσεις αντικειμένων.
- Η κλάση **απόγονος** (υποκλάση) κληρονομεί και μπορεί να χρησιμοποιήσει όλες τις ιδιότητες και τις μεθόδους που περιέχει η κλάση **πρόγονος** (υπερκλάση).

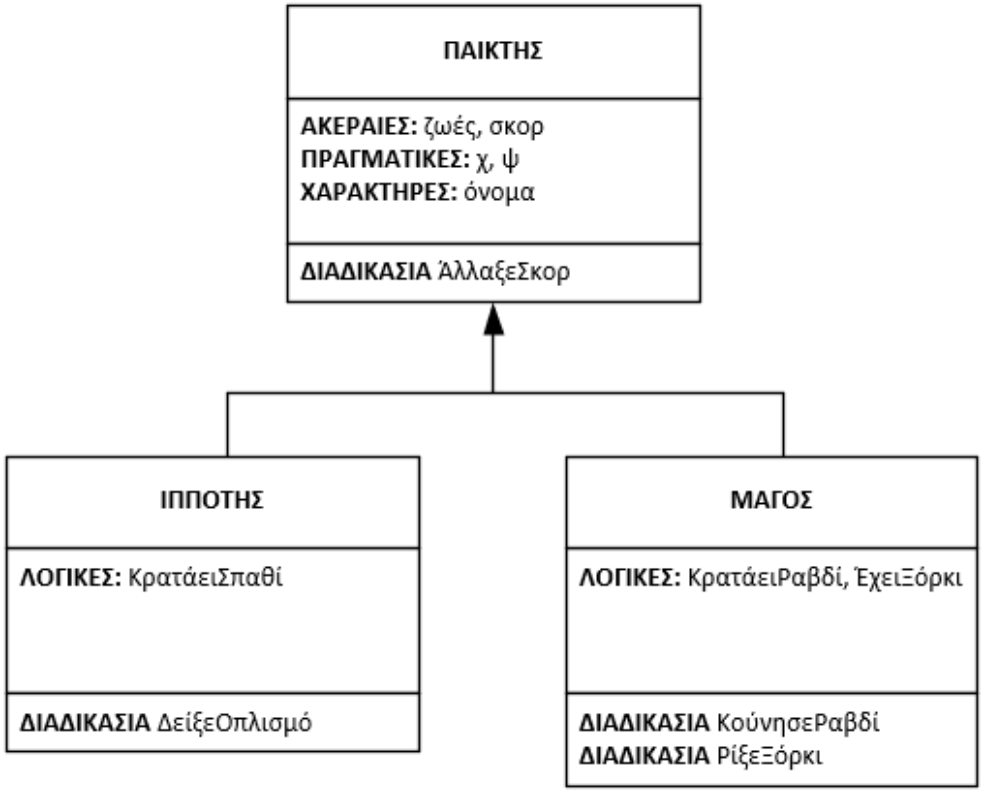


Κληρονομικότητα

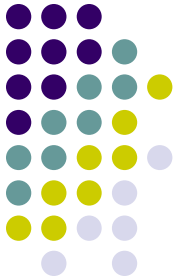
- Με την ίδια λογική μπορούμε να δημιουργήσουμε νέες «υπο-υποκλάσεις» που βασίζονται στις υποκλάσεις κ.ο.κ.

Κληρονομικότητα

- Ομοίως, από μία κλάση μπορούμε να έχουμε πολλαπλές υποκλάσεις/απογόνους

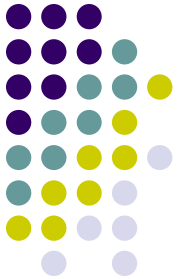


Πολυμορφισμός



- Όταν χρησιμοποιούμε κλάσεις δημιουργείται μερικές φορές η ανάγκη να αλλάξουμε τον τρόπο που κάποιες μέθοδοι αυτής λειτουργούν.
- Ας επιστρέψουμε στο παράδειγμα με το παιχνίδι κι ας δούμε την ανάγκη αυτή στην πράξη.
- Στην αρχική κλάση **Παίκτης** υπάρχει η μέθοδος **ΆλλαξεΣκορ** η οποία αυξάνει το σκορ του παίκτη κατά **1** κάθε φορά που καλείται.

Πολυμορφισμός



- Ας υποθέσουμε ότι θέλουμε να χρησιμοποιούμε τη συγκεκριμένη μέθοδο και στην περίπτωση που ο παίκτης μας είναι **Ιππότης**, με τη μόνη διαφορά ότι το σκορ θα πρέπει να αυξάνει κατά **10** αντί για **1** όπως εκείνο του **Παίκτη**.
- Για να το πετύχουμε αυτό, η τροποποίηση που πρέπει να κάνουμε στην κλάση **Ιππότης** είναι η ακόλουθη:

Πολυμορφισμός

```
ΚΛΑΣΗ Ιππότης : Παίκτης

  ΜΕΤΑΒΛΗΤΕΣ
    ΛΟΓΙΚΕΣ: ΚρατάειΣπαθί

  ΔΙΑΔΙΚΑΣΙΑ ΔείξεΟπλισμό
  ΑΡΧΗ
    ΑΝ ΚρατάειΣπαθί = ΑΛΗΘΗΣ ΤΟΤΕ
      ΓΡΑΨΕ 'Ο ιππότης κρατάει σπαθί'
    ΑΛΛΙΩΣ
      ΓΡΑΨΕ 'Ο ιππότης δεν κρατάει σπαθί'
    ΤΕΛΟΣ_ΑΝ
  ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

  ΔΙΑΔΙΚΑΣΙΑ ΆλλαξεΣκορ
  ΑΡΧΗ
    σκορ ← σκορ + 10
  ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

ΤΕΛΟΣ_ΚΛΑΣΗΣ
```

Η νέα κλάση **Ιππότης** βασίζεται στην κλάση **Παίκτης** και χρησιμοποιεί τις ιδιότητες και τις μεθόδους της

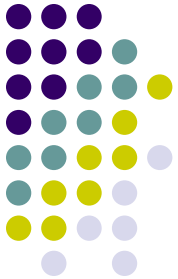
Η ιδιότητα **ΚρατάειΣπαθί** παραμένη ίδια

Η μέθοδος **ΔείξεΟπλισμό** παραμένει ίδια

Προσθέτουμε κι εδώ τη μέθοδο **ΆλλαξεΣκορ** για να επηρεάσουμε την συμπεριφορά της όταν καλείται από την κλάση **Ιππότης** και να αυξάνει το σκορ κατά 10

Αρχή και τέλος της κλάσης

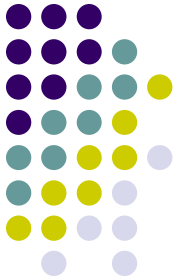
Πολυμορφισμός



- Αν δημιουργήσουμε στην αρχή του προγράμματος δύο αντικείμενα, ένα που να ανήκει στην κλάση **Παίκτης** κι ένα που να ανήκει στην κλάση **Ιππότης** και αρχικοποιήσουμε την ιδιότητα **σκορ** του καθενός, θα έχουμε:

```
ΠΡΟΓΡΑΜΜΑ Παιχνίδι
ΜΕΤΑΒΛΗΤΕΣ
   ΠΑΙΚΤΗΣ: Παίκτης1
   ΙΠΠΟΤΗΣ: Ιππότης1
ΑΡΧΗ
   Παίκτης1.σκορ ← 0
   Ιππότης1.σκορ ← 0
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Πολυμορφισμός



- Και αν στη συνέχεια εκτελέσουμε τις εντολές:

```
ΚΑΛΕΣΕ Παίκτης1.ΆλλαξεΣκορ
ΚΑΛΕΣΕ Ιππότης1.ΆλλαξεΣκορ
ΓΡΑΨΕ 'Το σκορ του παίκτη είναι: ', Παίκτης1.σκορ
ΓΡΑΨΕ 'Το σκορ του ιππότη είναι: ', Ιππότης1.σκορ
```

θα εμφανιστούν στην οθόνη τα μηνύματα:

```
Το σκορ του παίκτη είναι: 1
Το σκορ του ιππότη είναι: 10
```

Πολυμορφισμός - Ανασκόπηση

- **Πολυμορφισμός** (polymorphism) είναι μια ιδιότητα του αντικειμενοστρεφούς προγραμματισμού με την οποία **μια λειτουργία υλοποιείται με πολλούς διαφορετικούς τρόπους**.
- Ο πολυμορφισμός μας επιτρέπει να **επαναπροσδιορίσουμε τον τρόπο με τον οποίο λειτουργούν κάποια πράγματα,** είτε αλλάζοντας τον τρόπο λειτουργίας τους είτε αλλάζοντας τα εργαλεία τα οποία χρησιμοποιούνται για την επίτευξη του στόχου.

Συχνές ερωτήσεις – Απορίες

- **Ερώτηση**: Θα μπορούσαμε να δημιουργήσουμε έναν «πίνακα αντικειμένων»;

```
ΠΡΟΓΡΑΜΜΑ Παιχνίδι
ΜΕΤΑΒΛΗΤΕΣ
   ΠΑΙΚΤΗΣ: παίκτης[10]
   ΙΠΠΟΤΗΣ: ιππότης[20]
ΑΡΧΗ
   ! Κώδικας παιχνιδιού
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

- **Απάντηση**: Ναι!

Συχνές ερωτήσεις – Απορίες

- **Ερώτηση**: Θα μπορούσαμε να χρησιμοποιήσουμε και συναρτήσεις στις μεθόδους μίας κλάσης;

```
ΚΛΑΣΗ Παίκτης

  ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: ζωές, σκορ

  ΔΙΑΔΙΚΑΣΙΑ ΆλλαξεΣκορ
  ΑΡΧΗ
    σκορ ← σκορ + 1
  ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

  ΣΥΝΑΡΤΗΣΗ ΕίναιΖωντανός:ΛΟΓΙΚΗ
  ΑΡΧΗ
    ΑΝ ζωές > 0 ΤΟΤΕ
      ΕίναιΖωντανός ← ΑΛΗΘΗΣ
    ΑΛΛΙΩΣ
      ΕίναιΖωντανός ← ΨΕΥΔΗΣ
    ΤΕΛΟΣ_ΑΝ
  ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ

ΤΕΛΟΣ_ΚΛΑΣΗΣ
```

- **Απάντηση**: Ναι!

Τ Ε Λ Ο Σ

Ευχαριστώ για την προσοχή σας!