Μετάφραση Γλωσσών Αντικειμενοστρεφούς Προγραμματισμού

Διαλέξεις στο μάθημα: Μεταφραστές ΙΙ

Γεώργιος Μανής

Αντικειμενοστρεφής προγραμματισμός

- **αντικείμενα** (objects)
  - κατάσταση (δεδομένα)
  - συμπεριφορά (λειτουργίες)
- **κελυφοποίηση** (encapsulation)
  - αφαιρετική από κοινού αντιμετώπιση της κατάστασης και τις συμπεριφοράς των αντικειμένων
- **κλάσεις** (classes)
  - αντικείμενα με την ίδια συμπεριφορά
  - αντικείμενα της ίδιας κλάσης μπορεί να διαφέρουν μόνο κατά την κατάσταση
- **κληρονομικότητα** (inheritance)
  - μία κλάση μπορεί να οριστεί σαν επέκταση μιας άλλης κλάσης αποκτώντας τις μεθόδους και ιδιότητες της γονικής τους κλάσης προσθέτοντας και δικές της

Αντικειμενοστρεφής προγραμματισμός

- **υπερφόρτωση μεθόδου** (method overloading)
  - στην ίδια ή σε διαφορετικές κλάσεις υπάρχουν μέθοδοι με το ίδιο όνομα και διαφορετικά ορίσματα
- **υποσκέλιση μεθόδου** (method overriding)
  - μία θυγατρική κλάση και η γονική της έχουν μία μέθοδο ομώνυμη και με τα ίδια ορίσματα.
- **πολυμορφισμός** (polymorphism)
  - ο μεταγλωττιστής αποφασίζει ποια μέθοδο θα καλέσει, βασισμένος στον τύπο του τρέχοντος αντικειμένου
- **αφηρημένη κλάση** (abstract class)
  - μία κλάση που ορίζεται μόνο για να κληρονομηθεί σε θυγατρικές υποκλάσεις και δεν υπάρχουν δικά της στιγμιότυπα (αντικείμενα)

Αντικείμενα και μέθοδοι

- Τα **αντικείμενα** διαφέρουν από τις εγγραφές καθώς οι εγγραφές περιέχουν μόνο δεδομένα ενώ τα αντικείμενα περιέχουν και μεθόδους
- Οι **μέθοδοι** είναι οι διαδικασίες και οι συναρτήσεις που εφαρμόζονται στα δεδομένα ενός αντικειμένου και αφορούν συγκεκριμένη εμφάνιση ενός αντικειμένου
- Η **μεταγλώττιση** μπορεί να γίνει ακριβώς με τον ίδιο τρόπο που θα γινόταν σε μία συνάρτηση ή διαδικασία εάν αυτή είχε ένα ακόμα όρισμα: το όνομα του αντικειμένου
- Η παράσταση ενός αντικειμένου στη **μνήμη** δεν διαφέρει από αυτή μιας εγγραφής αφού απαιτείται χώρος μόνο για τα δεδομένα

Αντικείμενα και μέθοδοι

```
class Shape
        var posX, posY: real;

procedure move (dx,dy:real)
        begin
        posX:=posX+dx;
        posY:=posY+dy;
        end
end

var s: Shape;
```

Shape
posX
posY

```
Shape struct
        posX:real;
        posY: real
end
var Shape s;
```

Η παράσταση ενός αντικειμένου στη μνήμη με τα δεδομένα δεν διαφέρει από αυτή μιας εγγραφής αφού απαιτείται χώρος μόνο για τα δεδομένα και όχι για τον κώδικα

Αντικείμενα και μέθοδοι

```
class Shape
        var posX, posY: real;

procedure move (dx,dy:real)
        begin
        posX:=posX+dx;
        posY:=posY+dy;
        end
end

var s: Shape;
```

```
procedure Shape@move (var self: Shape; dx,dy:real)
        begin
        self.posX:=self.posX+dx;
        self.posY:=self.posY+dy;
        end
```

- **η μεταγλώττιση μπορεί να γίνει ακριβώς με τον ίδιο τρόπο που θα γινόταν σε μία συνάρτηση ή διαδικασία εάν αυτή είχε ένα ακόμα όρισμα: το όνομα του αντικειμένου**
- **το όνομα της διαδικασίας περιέχει το όνομα της κλάσης έτσι ώστε περισσότερες κλάσεις να μπορούν να ορίζουν μεθόδους με το ίδιο όνομα**
- **με τον χαρακτήρα @ εξασφαλίζεται η μοναδικότητα των ονομάτων (δεν αποτελεί νόμιμο χαρακτήρα για τη γλώσσα)**
- **Η κλήση s.move(x,y) γίνεται Shape@move(s,x,y)**

Κατασκευαστές και καταστροφείς

- καλούνται κατά την **κατασκευή** και **καταστροφή** των αντικειμένων αντίστοιχα
- οι **κατασκευαστές** (constructors)
  - αρχικοποιούν ένα αντικείμενο αμέσως μετά τη δέσμευση μνήμης
  - υλοποιούνται σαν μία μέθοδο που δεν επιστρέφει αποτέλεσμα
- οι **καταστροφείς** (destructors)
  - καλούνται αμέσως πριν την αποδέσμευση της μνήμης
  - υλοποιούνται σαν μία μέθοδο που δεν επιστρέφει αποτέλεσμα ούτε έχει παραμέτρους

Κατασκευαστές και καταστροφείς

```
class Shape
        var posX, posY: real;
        procedure constructor (dx,dy:real)
            begin
            posX:=dx;
            posY:=dy;
            end
        procedure destructor (I)
           begin
           …
           end
end
```

```
procedure Shape@constructor (var self:Shape;x.y:real);
        begin
        self.posX:=x;
        self.posY:=Y;
        end

procedure Shape@destructor(var self:Shape)
        begin
        …
        end;
```

Η ακόλουθη δήλωση μεταβλητής var s:Shape(10,20) **έχει σαν συνέπεια στην αρχή της δομικής μονάδας που βρίσκεται η δήλωση αυτή να προστεθεί μία κλήση στον κατασκευαστή** Shape@constructor (s, 10, 20) **και στο τέλος στον καταστροφέα** Shape@destructor(s)

Σε περίπτωση που τα αντικείμενα κατασκευάζονται και καταστρέφονται δυναμικά πρέπει να χρησιμοποιούνται και οι εντολές δυναμικής εκχώρησης και επιστροφής μνήμης

Κληρονομικότητα

- **ιεραρχίες κλάσεων** για επαναχρησιμοποίηση κώδικα
- στην απλή κληρονομικότητα κάθε κλάση μπορεί να χρησιμοποιήσει τα **χαρακτηριστικά μιας άλλης κλάσης**, όχι όμως περισσοτέρων από μιας
- η κλάση που κληρονομεί τα χαρακτηριστικά λέγεται **παραγόμενη κλάση** (derived class) ή **υποκλάση** (subclass)
- η κλάση που κληροδοτεί λέγεται **υπερκλάση** (superclass) ή **βασική κλάση** (base class)
- η παραγόμενη κλάση μπορεί εκτός από τα χαρακτηριστικά που κληρονομεί να **ορίζει δικά της πεδία και μεθόδους**

Κληρονομικότητα

```
class Shape
        var posX, posY: real;
end

class Circle extends Shape
        var radius:real;

        procedure scale (s:real)
        begin
                        radius := radius
        * s
        end
end
```

Shape
posX
posY

Circle
posX
posY
radius

Κληρονομικότητα

```
class Shape
        var posX, posY: real;
end


class Line extends Shape
        var endX,endY;

        procedure move (dx,dy:real)
        begin
                        posX:=posX+dx;
                        posY:=posY+dy;
                        endX:=endX+dx;
                        endY:=endY+dy;
        end
end
```

Shape
posX
posY

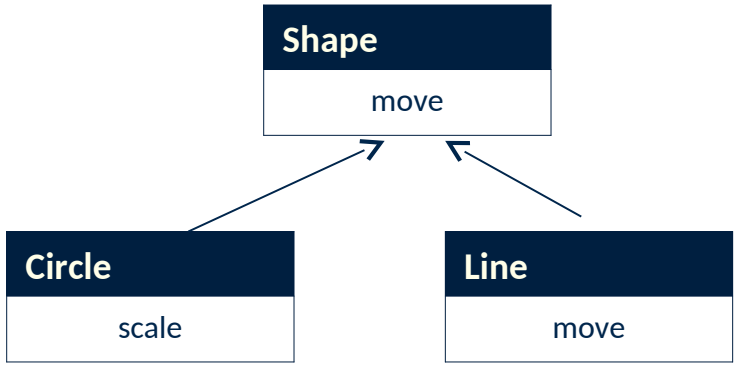
Line
posX
posY
endX
endY

Για να είναι ευκολότερο τα πεδία της βασικής κλάσης να χρησιμοποιούνται από την παραγόμενη κλάση πρέπει τα πεδία να έχουν τις ίδιες σχετικές διευθύνσεις με αυτά της βασικής κλάσης. Δηλαδή τα κληρονομούμενα πεδία να τοποθετούνται στην αρχή της μνήμης που δεσμεύεται για την παραγόμενη κλάση και τα υπόλοιπα πεδία να ακολουθούν

Στατικό δέσιμο μεθόδων

- σε κάθε κλήση μεθόδου πρέπει να είναι γνωστό σε ποια κλάση ανήκει η μέθοδος
- αν η αναζήτηση γίνεται από τον μεταγλωττιστή βάσει του τύπου του αντικειμένου πάνω στο οποίο καλείται η μέθοδος τότε λέμε ότι η γλώσσα υποστηρίζει **στατικό δέσιμο μεθόδων** (static binding)
- η μέθοδος **αναζητείται** πρώτα στην κλάση στην οποία ανήκει το αντικείμενο στο οποίο εφαρμόζεται
- **αν δε βρεθεί** τότε αναζητείται στην υπερκλάση αυτής
- αν δε βρεθεί και στην υπερκλάση τότε συνεχίζουμε στην υπερκλάση της υπερκλάσης και η αναζήτηση συνεχίζεται μέχρι τη **ρίζα** της ιεραρχίας
- αν και εκεί δε βρεθεί τότε εμφανίζουμε **μήνυμα λάθους**

Στατικό δέσιμο μεθόδων



Παράδειγμα:

```
var   s:Shape(10,20)
      c:Circle(30,30,10)
      i:Line(10,20,30,30)

s.move (5,5)      --->    Shape@move(s,5,5)
c.move(5,5)       --->    Shape@move(c,5,5)

l.move(5,5)       --->    Line@move(l,5,5)
c.scale(2)        --->    Circle@scale(c,2)
```

Η Circle δεν ορίζει τη μέθοδο move και κατά συνέπεια αναζητεί τη μέθοδο στην κλάση Shape και καλεί εκεί την move

Η κλάση Line επαναορίζει αυτή τη μέθοδο και άρα η κλήση της l.move καλεί τη δική της μέθοδο

Δυναμικό δέσιμο μεθόδων

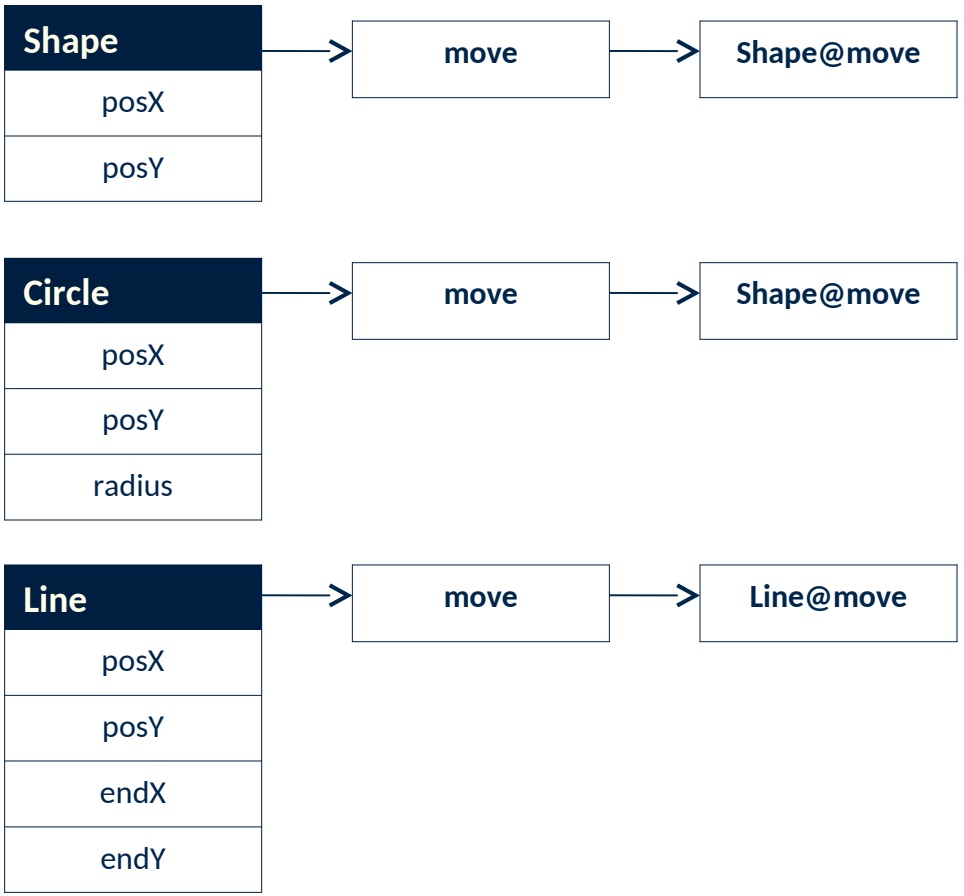
- **Το δυναμικό δέσιμο** μεθόδων (dymamic binding) χρησιμοποιείται από όλες τις αντικειμενοστρεφείς γλώσσες σήμερα
- λέγεται και **πολυμορφισμός υποτύπων** (subtype polymorphism) και καταχρηστικά **πολυμορφισμός**
- αναζήτηση της συνάρτησης που πρέπει να κληθεί **σε χρόνο εκτέλεσης**
- δεν είναι πάντοτε εύκολο να αποφασιστεί **σε χρόνο μεταγλώττισης** ποια μέθοδος πρέπει να κληθεί (π.χ. απαιτείται αποδεικτοδότηση ενός δείκτη)
- απαιτείται **διαφορετική** παράσταση των αντικειμένων στη μνήμη
- ένας πίνακας περιέχει τις διευθύνσεις των δυναμικών μεθόδων, **ονομάζεται πίνακας ανταπόκρισης μεθόδων** (method dispatch table) ή **περιγραφέας κλάσης** (class descriptor)

Δυναμικό δέσιμο

```
class Shape
        var posX, posY: real;
        dynamic procedure move (dx,dy:real)
        begin
                        posX:=posX+dx;
                         posY:=posY+dy;
        end
end


class Line extends Shape
        var endX,endY;
        dynamic procedure move (dx,dy:real)
        begin
                        super.move(dx,dy)
                        endX:=endX+dx;
                        endY:=endY+dy;
        end
end
```

p^.move(5,5)



Αφηρημένες κλάσεις

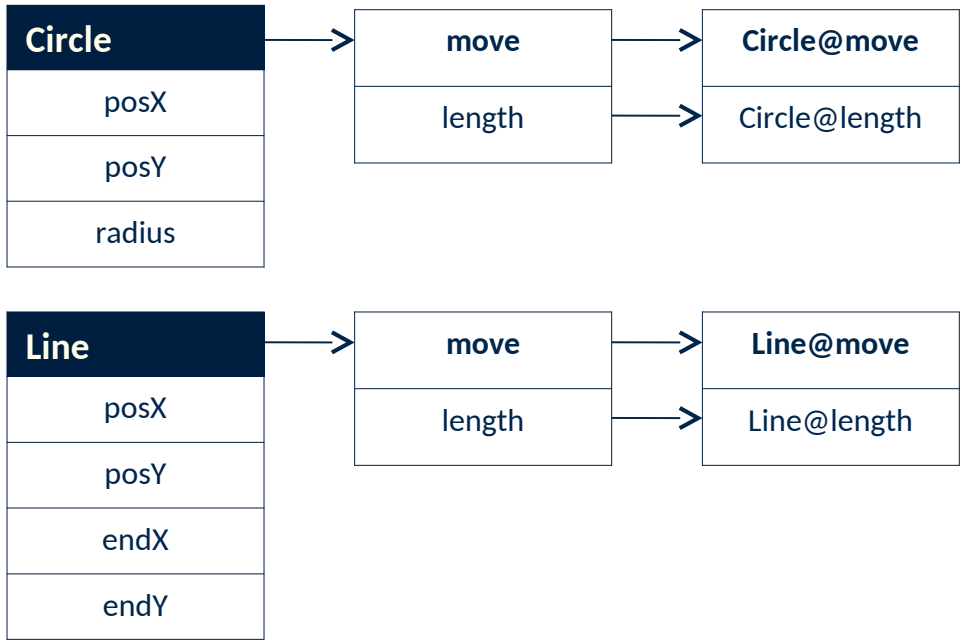
- κλάσεις που περιέχουν μεθόδους, ο **κώδικας** των οποίων δηλώνεται σε κάποια υποκλάση
- ονομάζονται **αφηρημένες κλάσεις** (abstract classes)
- δεν είναι δυνατόν να κατασκευαστούν **αντικείμενα** αυτών των κλάσεων
- δεν απαιτείται να κατασκευαστούν **πίνακες ανταπόκρισης μεθόδων** για αυτές

Αφηρημένες κλάσεις

```
abstract class Shape
        ...
        function length(): real
end

class Line extends Shape
        ...
        function length(): real
         ...
end

class Circle extends Shape
        ..
        function length(): real
         ...
end
```

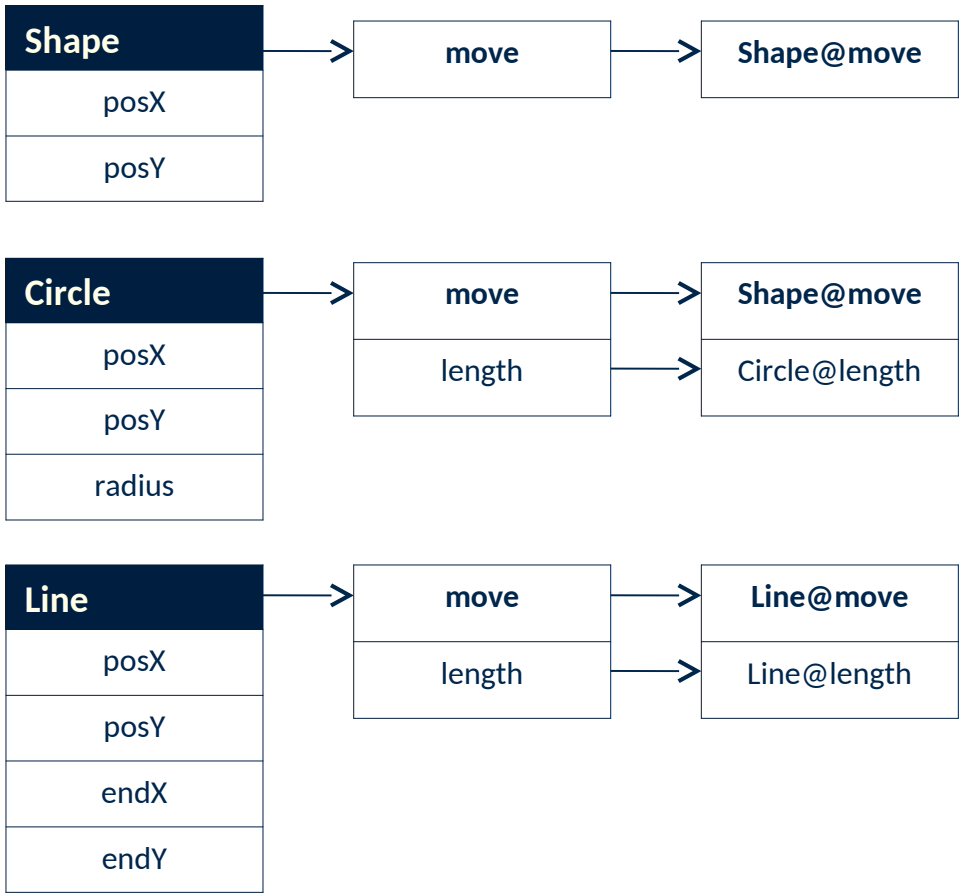


Αφηρημένες κλάσεις

```
class Shape
        ...
        abstract function length(): real
end

class Line extends Shape
        ...
        function length(): real
         ...
end

class Circle extends Shape
        ..
        function length(): real
         ...
end
```



Ευχαριστώ !!!