Προχωρημένα θέματα

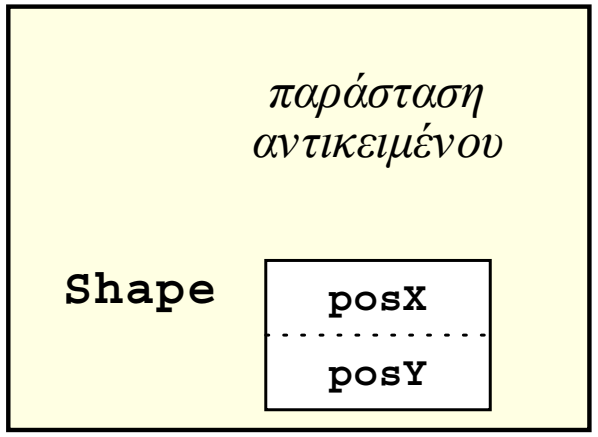
- Μεταγλώττιση αντικειμενοστρεφών γλωσσών
- Χαρακτηριστικά του αντικειμενοστρεφούς προγραμματισμού
  - λογισμικό οργανωμένο ως σύνολο από αλληλεπιδρώντα αντικείμενα
  - αντικείμενο / κλάση
  - κελυφοποίηση (encapsulation)
  - κληρονομικότητα (inheritance)
  - πολυμορφισμός υποτύπων (subtype polymorphism)

Αντικείμενα και μέθοδοι (i)

- Αντικείμενο = εγγραφή (record) + μέθοδοι
- Αναπαράσταση αντικειμένων: ίδια με την αναπαράσταση εγγραφών
  - Δεσμεύεται χώρος για τα πεδία του αντικειμένου
  - Οι μέθοδοι δε συμπεριλαμβάνονται στην αναπαράσταση
  - Οι κλήσεις μεθόδων μεταφράζονται σε κλήσεις συναρτήσεων με μια επιπλέον παράμετρο: ένα δείκτη στο αντικείμενο όπου αναφέρονται

Αντικείμενα και μέθοδοι (ii)

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

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



⇓

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

s.move(1, 2) ⇒ Shape@move(s, 1, 2)

Κατασκευαστές & καταστροφείς

- Ίδια αντιμετώπιση με τις μεθόδους

```
class Shape
  ...
  procedure constructor (x, y : real);
  ...
  procedure destructor ();
  ...
end;
```

⇓

```
procedure Shape@constructor (var self : Shape;
                             x, y : real);
procedure Shape@destructor  (var self : Shape);
```

```
var s : Shape(10, 20);  ⇒
```

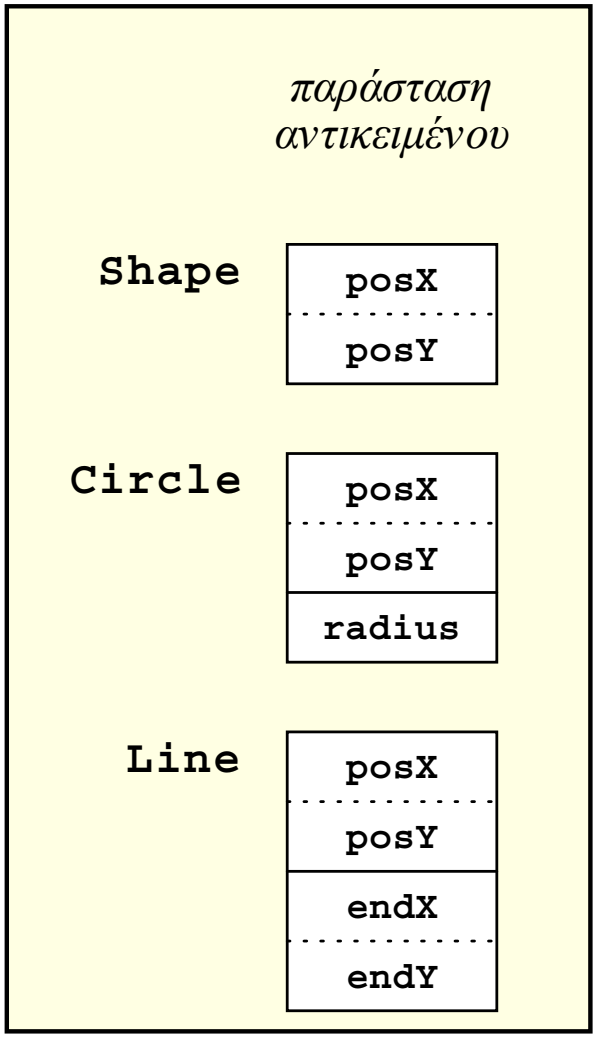
```
Shape@constructor(s, 10, 20);
...
Shape@destructor(s);
```

Απλή κληρονομικότητα (i)

- Υποκλάση / υπερκλάση
- Επισκίαση μεθόδων (method override)
- Υποτύποι (subtypes)
- Αναπαράσταση αντικειμένων
  - Τα πεδία της υπερκλάσης τοποθετούνται πρώτα (στην αρχή του αντικειμένου)
  - Αυτό εφαρμόζεται αναδρομικά
  - Τα πεδία της υποκλάσης ακολουθούν

Απλή κληρονομικότητα (ii)

```
class Circle extends Shape
  var radius : real;
  procedure scale (s : real);
  ...
end;

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



Απλή κληρονομικότητα (iii)

- Στατικό δέσιμο μεθόδων (static binding)

```
var s : Shape(10, 20);
    c : Circle(30, 30, 10);
    l : Line(10, 20, 30, 30);
    p : ^Shape;

...
p := @l;

s.move(5, 5);
c.move(5, 5);
l.move(5, 5);
c.scale(2);
p^.move(5, 5);
```

$\Rightarrow$

```
Shape@move(s, 5, 5);
Shape@move(c, 5, 5);
Line@move(l, 5, 5);
Circle@scale(c, 2);
Shape@move(p^, 5, 5)
```

Πολυμορφισμός υποτύπων (i)

- Δυναμικό δέσιμο μεθόδων (dynamic binding)
  - Όταν γίνεται κλήση μεθόδων μέσω δεικτών ή αναφορών σε αντικείμενα της υπερκλάσης, καλούνται οι αντίστοιχες μέθοδοι των υποκλάσεων
  - Πίνακας ανταπόκρισης μεθόδων (method dispatch table) ή περιγραφέας κλάσης (class descriptor)
  - Κοινός για όλα τα αντικείμενα μιας κλάσης

Πολυμορφισμός υποτύπων (ii)

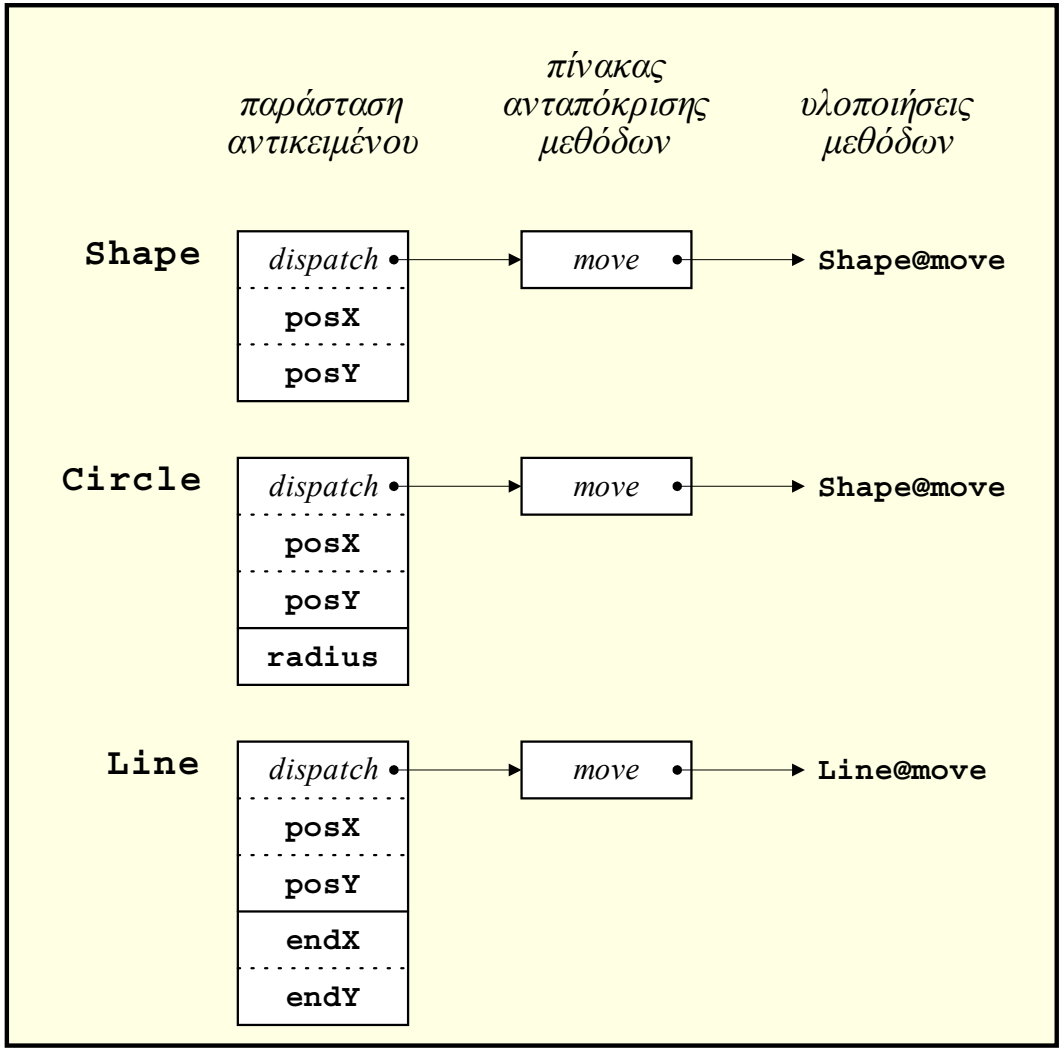
```
class Shape
  var posX, posY : real;
  dynamic procedure move (dx, dy : real);
end;

class Line extends Shape
  var endX, endY : real;
  dynamic procedure move (dx, dy : real);
end;

var p : ^Shape;

...
p := @l;

p^.move(5, 5)
```

Πολυμορφισμός υποτύπων (iii)

```
var p : ^Shape;

...
p := @l;

p^.move(5, 5)
```

```
f := p^.dispatch^.move;
f^(p^, 5, 5)
```

- Ομοίως για αφηρημένες (abstract) μεθόδους



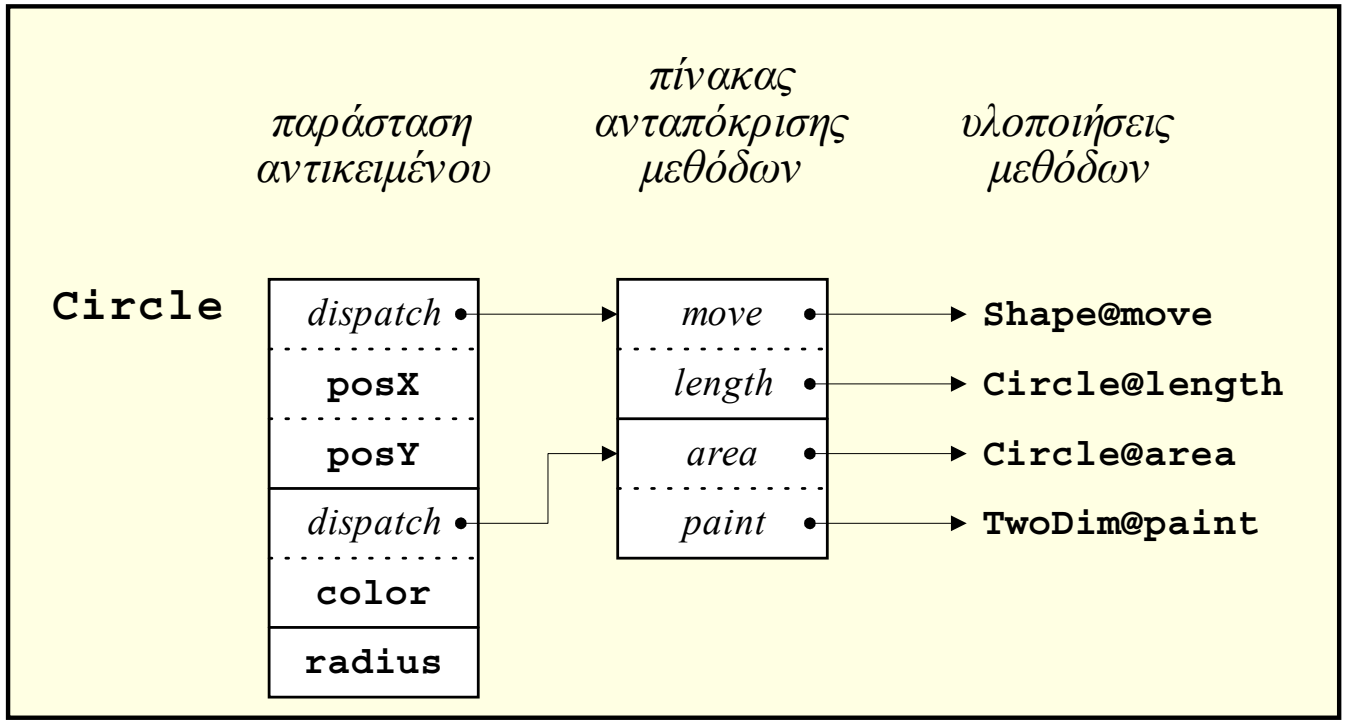
Πολλαπλή κληρονομικότητα (i)

- Πρόβλημα με την αναπαράσταση των αντικειμένων σε συνδυασμό με τους υποτύπους

```
class TwoDim
   var color : ColorType;
   abstract function area () : real;
   dynamic procedure paint (c : ColorType);
end;

class Circle extends Shape, TwoDim
   ...
   dynamic function area () : real;
end;
```

Πολλαπλή κληρονομικότητα (ii)

```
var c  : Circle(...);
    p1 : ^Shape;
    p2 : ^TwoDim;

...
p1 := @c;
p2 := @c
```



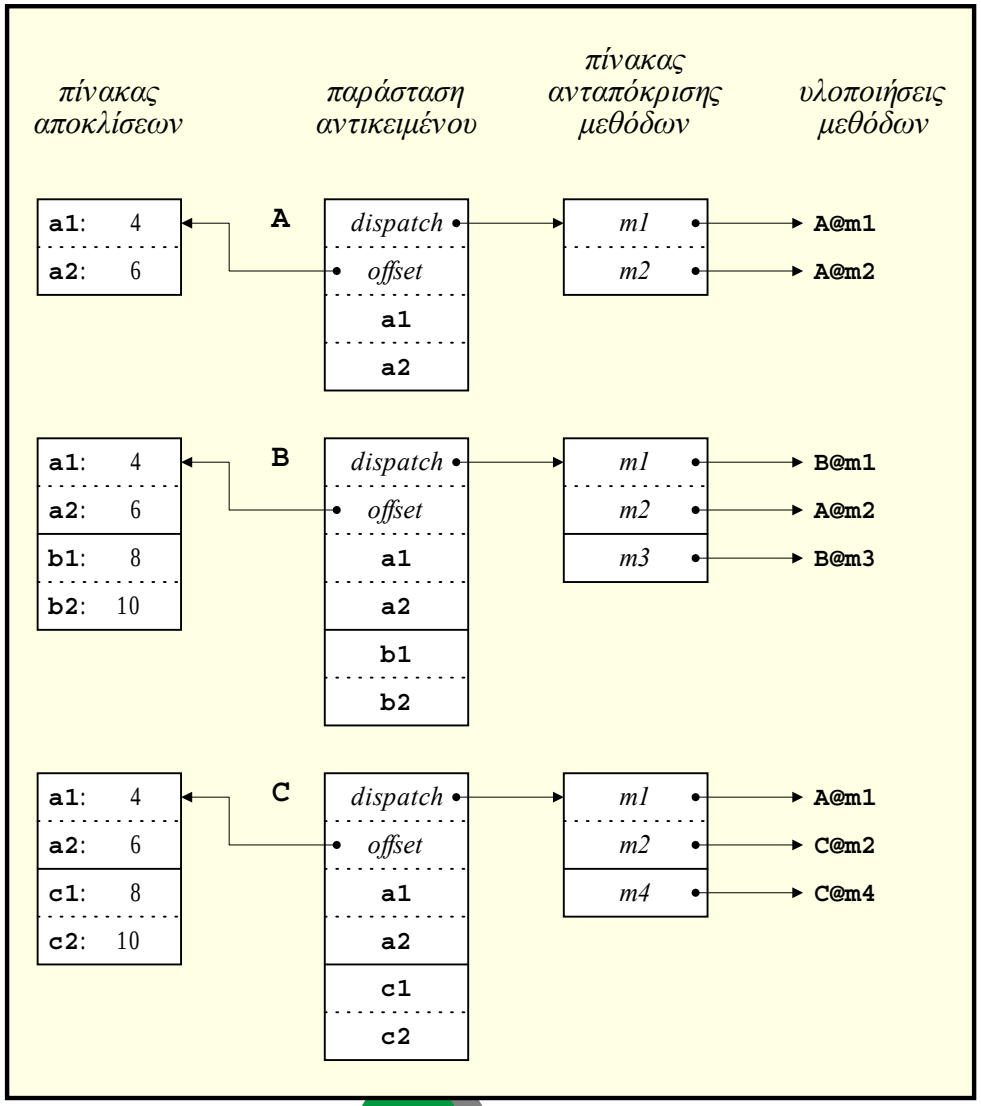
Πολλαπλή κληρονομικότητα (iii)

- Εξαρτημένη πολλαπλή κληρονομικότητα

```
class A
   var a1, a2 : integer;
   dynamic procedure m1 ();
   dynamic procedure m2 ();
end;

class B extends A
   var b1, b2 : integer;
   dynamic procedure m1 ();
   dynamic procedure m3 ();
end;
```

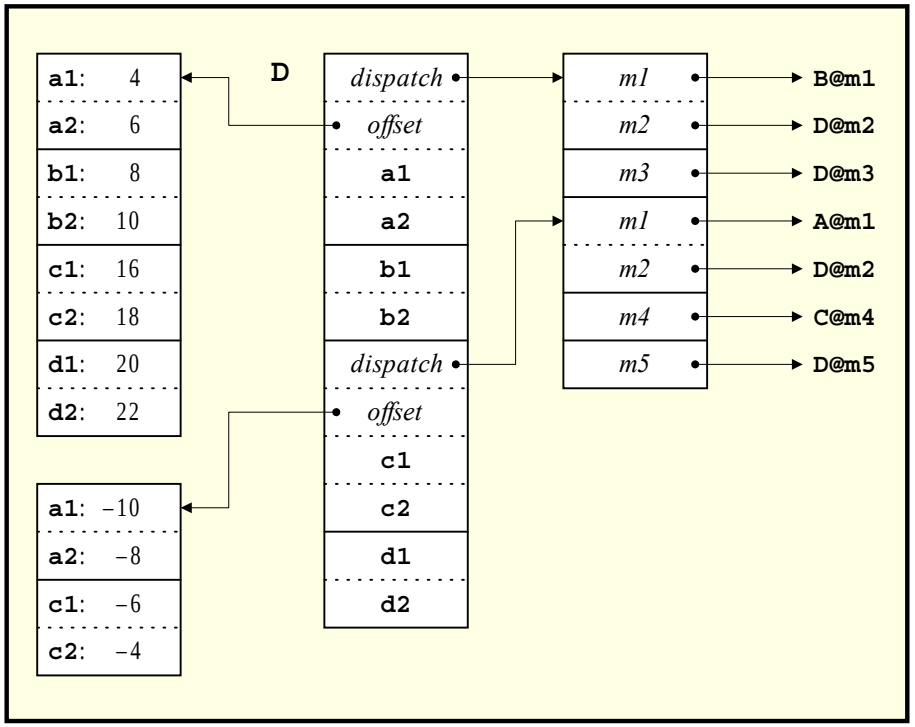
```
class C extends A
   var c1, c2 : integer;
   dynamic procedure m2 ();
   dynamic procedure m4 ();
end;

class D extends B, C
   var d1, d2 : integer;
   dynamic procedure m2 ();
   dynamic procedure m3 ();
   dynamic procedure m5 ();
end;
```

Πολλαπλή κληρονομικότητα (iv)



- Πίνακας αποκλίσεων (offset table)

```
p^.a1    ⇒
(p + p^.offset^.a1)^
```



Έλεγχος υποτύπων (i)

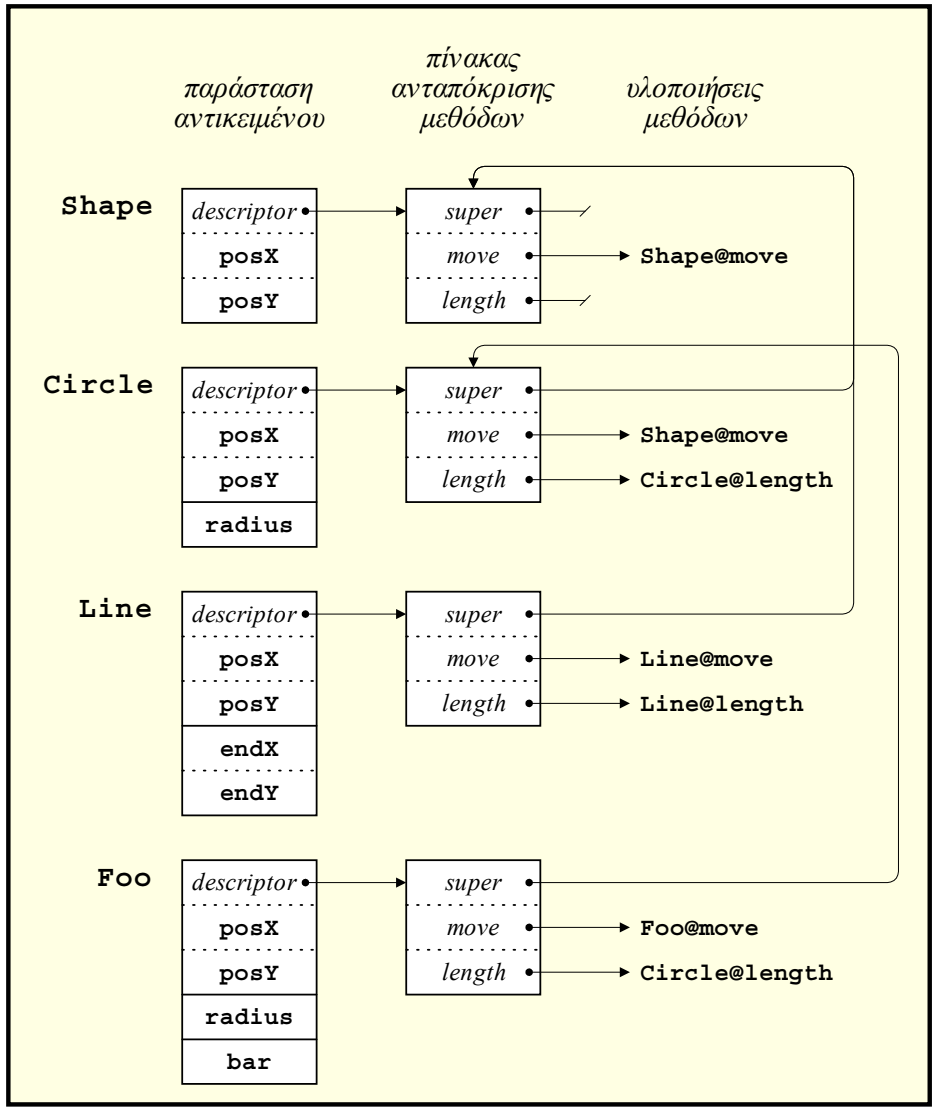
Ερώτηση 1: σε ποια κλάση ανήκει το αντικείμενο X;
Απάντηση: απλώς σύγκρινε τους περιγραφείς κλάσης

Ερώτηση 2: ανήκει το αντικείμενο X στην κλάση $\mathcal{C}$;
Απάντηση: ...λίγο πιο δύσκολη...

```
var p : ^Shape;
...
if p^ instanceOf Circle then
   ...1...
else
   ...2...
```

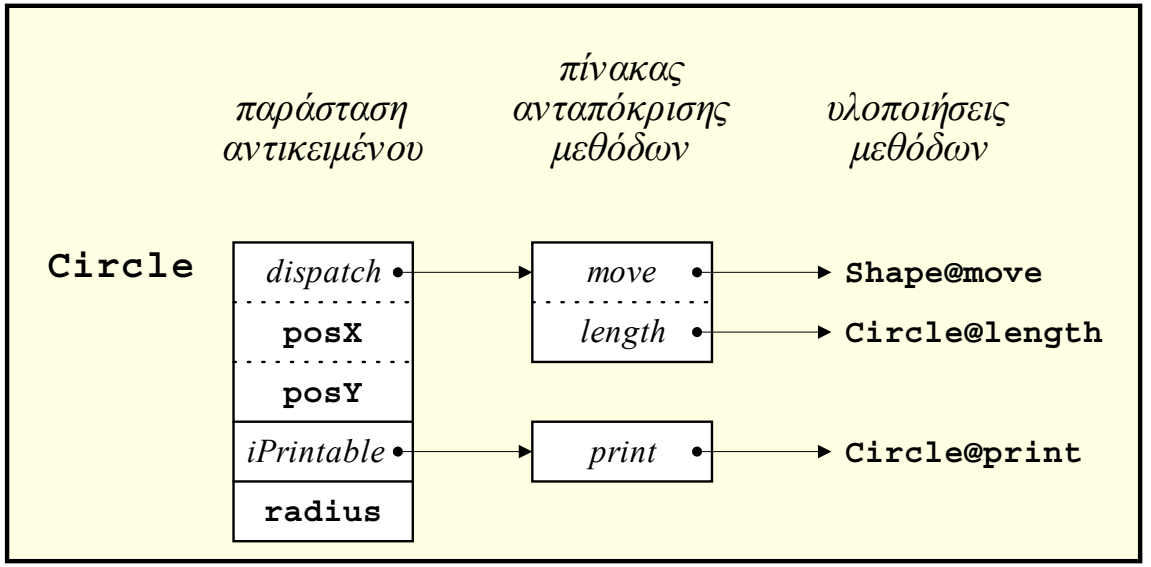
Έλεγχος υποτύπων (ii)

```
  d := p^.descriptor;
loop:
  if d = Circle@descriptor
  then begin
    ...1...
    goto next
  end
  else if d = nil then
  begin
    ...2...
    goto next
  end
  else
    d = d^.super;
  goto loop;
next:
```



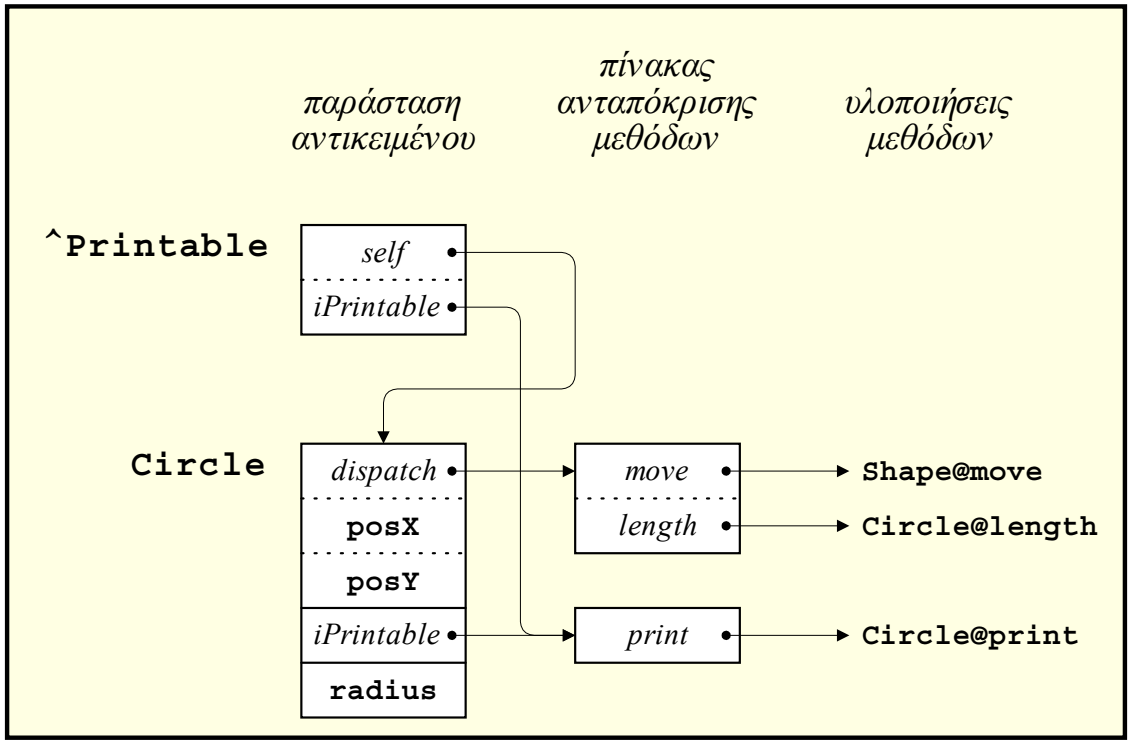
Διαπροσωπείες (i)

```
interface Printable
   dynamic procedure print ();
end;

class Circle extends Shape implements Printable
   ...
   dynamic procedure print ();
end;
```



Διαπροσωπείες (ii)

```
var c : Circle(...);
    p : ^Printable;

...
p := @c;

p^.print()
```



⇓

```
p.self := @c;
p.iPrintable := c.iPrintable
```