# Διάλεξη 5: Κλάσεις και Αντικείμενα

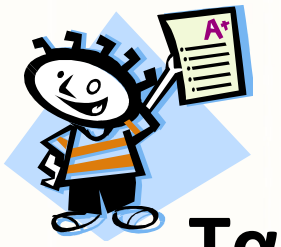
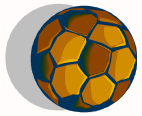
**Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:**
Εισαγωγή στις έννοιες:

- Κλάσεις και Αντικείμενα
- Κατασκευή, Πρόσβαση Αντικειμένων
- Διαχείριση Μνήμης, Garbage collector

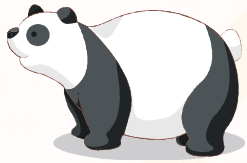
## Διδάσκων: Παναγιώτης Ανδρέου

# Αντικείμενα

- Τα αντικείμενα είναι ο φυσικός τρόπος να αναπαραστήσεις τον πραγματικό κόσμο σε ένα σύστημα
- Τα αντικείμενα είναι εύκολα στην κατανόηση
- Τα αντικείμενα έχουν φέρει την επανάσταση στην ανάπτυξη συστημάτων
- Οι εφαρμογές με αντικείμενα είναι εύκολες στην κατανόηση και στην συντήρηση


**Τα πάντα**



**είναι αντικείμενα!!!**





# Κατηγορίες Αντικειμένων

- **Χειροπιαστά αντικείμενα**
  Πραγματικά αντικείμενα που υπάρχουν στον φυσικό κόσμο
- **Ρόλοι**
  Σκοπός ή ανάθεση ενέργειας ενός άτομο, εξοπλισμού ή οργανισμού
- **Περιστατικά**
  Συμβαίνει ένα φαινόμενο, συμβάν (event)
- **Αλληλεπιδράσεις**
  Αποτέλεσμα σχέσεις μεταξύ αντικειμένων
- **Προδιαγραφές**
  Αντιπροσωπεύουν κανόνες, πρότυπα ή κριτήρια

## Τι είναι ένα αντικείμενο;

- Οτιδήποτε στον πραγματικό κόσμο
- Κάτι χειροπιαστό π.χ., ο υπολογιστής
- Μία διαδικασία, π.χ., δοκιμή μίας μηχανής
- Μία σχέση, π.χ., ένα συμβόλαιο
- Κάτι θεωρητικό, π.χ., ένας πίνακας

# Οτιδήποτε είναι «ουσιαστικό» … είναι αντικείμενο

# Το μοντέλο του αντικειμένου

- **Βρίσκεται στο κέντρο**
- **Είναι ο τελικός αποδέκτης των πάντων**
- **Είναι το σημείο εκκίνησης για την υλοποίηση**
- **Βασίζεται στην αφαιρετικότητα (abstraction)**
  Νοητική διαδικασία που επιτρέπει να αντιλαμβανόμαστε και να δομούμε τη γνώση για διάφορες έννοιες του πραγματικού κόσμου σε διάφορα επίπεδα και σε διάφορους τομείς (domains)

# Πραγματικός Κόσμος: Αντικείμενα και Τομείς

- **Τομέας (domain):** ξεχωριστό μέρος του πραγματικού κόσμου που περιλαμβάνει τις δικές του οντότητες
- Καθετί είναι αντικείμενα και **κάθε αντικείμενο ανήκει σε κάποιο τομέα**
- **Τομέας ενός προβλήματος:** περιοχή που απευθύνεται το πρόβλημα
- Ο στόχος είναι να καθοριστεί η εμβέλεια και τα όρια του τομέα προβλήματος
- Η κατανόηση του τομέα ενός προβλήματος είναι απαραίτητη για την μοντελοποίηση του συστήματος/εφαρμογής

# Αναπαράσταση Αντικειμένων

- Κάθε αντικείμενο έχει ένα **όνομα**, π.χ., μπάλα, αυτοκίνητο, κύκλος
- Τα αντικείμενα έχουν **χαρακτηριστικά που καθορίζουν την κατάστασή** τους, π.χ., βάρος, μήκος, αριθμός θέσεων
- Τα αντικείμενα έχουν **λειτουργίες** οι οποίες καθορίζουν, τι μπορεί να κάνει ένα αντικείμενο σε κάποιο άλλο αντικείμενο, τι μπορεί να συμβεί σε ένα αντικείμενο, π.χ., πετάω, ξεκινώ, σταματώ, μετρώ.

## Ορολογία αντικειμενοστρεφή προγραμματισμού

- **Ταυτότητα** (identity): Το "κλειδί" μέσω του οποίου μπορούμε να αποκτήσουμε πρόσβαση στο αντικείμενο → **Όνομα/ Διεύθυνση Μνήμης**
- **Κατάσταση** (state): Το σύνολο των δεδομένων που αποθηκεύονται στο εσωτερικό του αντικειμένου → **Πεδία/Τιμές Δεδομένων**
- **Συμπεριφορά** (behavior): Το σύνολο των ενεργειών (μεθόδων) που μπορεί να διεκπεραιώσει το αντικείμενο: συναρτήσεις που διαθέτει → **Λειτουργίες/ Μέθοδοι**

# Σχέσεις μεταξύ αντικειμένων

- **Κατάταξη (classification)**
  Σε ποια ομάδα ανήκει το κάθε αντικείμενο
- **Γενίκευση/Ειδίκευση (generalization/specialization)**
  Ποια άλλα αντικείμενα είναι παρόμοια με κάποιο αντικείμενο
- **Συνάθροιση (aggregation)**
  Ποια αντικείμενα «καλύπτονται» από ένα συγκεκριμένο αντικείμενο
- **Συσχέτιση (association)**
  Ειδική σχέση που επιτρέπει σε ένα αντικείμενο να προκαλέσει ένα άλλο αντικείμενο να εκτελέσει κάποια ενέργεια εκ μέρους του

# Κλάσεις και Αντικείμενα

- **Μία κλάση είναι ένα πρότυπο για τη δημιουργία παρόμοιων αντικειμένων.**
- **Μία κλάση είναι όπως ένα αρχιτεκτονικό σχέδιο**: περιγράφει τις ιδιότητες ενός αντικειμένου
- Όλα τα αντικείμενα σε μία κλάση έχουν τα ίδια χαρακτηριστικά, τις ίδιες μεθόδους αλλά όχι τις ίδιες τιμές.
- Από μία κλάση μπορούν να δημιουργηθούν πολλά στιγμιότυπα (αντικείμενα)

**Κλάση**

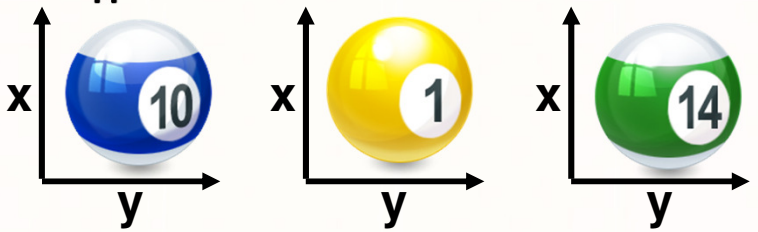
**Αντικείμενα (Στιγμιότυπα)**

```
public class Anthopolis{
…

Anthopolis Giannis;

Anthopolis Athos;
```

# Πραγματικός Κόσμος: Κλάσεις και Αντικείμενα

- Το σύνολο όλων των μπάλων μπιλιάρδου είναι μία κλάση

- Όλες οι μπάλες έχουν τα ίδια χαρακτηριστικά όπως μέγεθος, χρώμα, βάρος




- Όλες οι μπάλες υποστηρίζουν κάποιες ενέργειες π.χ., κτύπημα, τρύπα


- Διαφορετικές μπάλες (instances) μπορούν να έχουν διαφορετικό μέγεθος, διαφορετικό χρώμα και διαφορετικό βάρος

# Παραδείγματα: Κλάση

```
class Circle {
  //Η ακτίνα αυτού του κύκλου
  double radius = 1.0;

  //Δημιούργησε ένα αντικείμενο τύπου κύκλος
  Circle () {
  };

  //Δημιούργησε ένα αντικείμενο τύπου κύκλος
  //με συγκεκριμένη ακτίνα
  Circle (double newRadius) {
    radius = newRadius;
  };

  //Επέστρεψε το εμβαδό αυτού του κύκλου
  double getArea () {
    return radius * radius * π;
  }

}
```

**Δεδομένα/ Μεταβλητές**

**Κατασκευαστές**

**Μέθοδοι**

# Κλάσεις για τον προγραμματιστή

- Οι κλάσεις επιτρέπουν την δημιουργία νέων τύπων (παρόμοια με τις δομές (structs) στην C
  Παράδειγμα: Δημιουργία κλάσης για δημιουργία αντικειμένων που αναπαραστούν σημεία Χ,Υ

```
class Point {
  int x;
  int y;
}
```

- Αντίθετα με την C, τώρα κάθε δομή δηλ. κλάση μπορεί να περιλαμβάνει τις δικές τις ενέργειες (methods)
  →User-defined Abstract Data Types

```
class Point {
  int x;
  int y;

  int distance(
    Point other) {
    return sqrt(
     (this.x-other.x)² +
     (this.y-other.y)²;
  }
}
```

- **Η δημιουργία αντικειμενοστρεφή προγραμμάτων είναι η δημιουργία ADTs**

# Παραδείγματα: Αντικείμενα

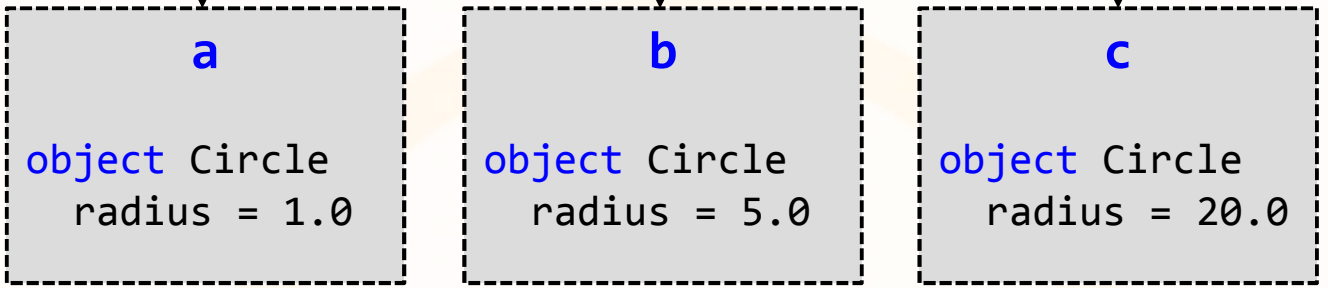
## Κλάση Circle

```
class Circle {
  //Η ακτίνα αυτού του κύκλου
  double radius = 1.0;

  //Δημιούργησε ένα αντικείμενο τύπου κύκλος
  Circle () {
  };

  //Δημιούργησε ένα αντικείμενο τύπου κύκλος
  //με συγκεκριμένη ακτίνα
  Circle (double newRadius) {
    radius = newRadius;
  };

  //Επέστρεψε το εμβαδό αυτού του κύκλου
  double getArea () {
    return radius * radius * π;
  }

}
```

## Κλάση Test: περιλαμβάνει την μέθοδο main

```
public class TestCircle{

  public static void main(String[] args){
    Circle a = new Circle();
    Circle b = new Circle(5);
    Circle c = new Circle(20);

…
```

**3 αντικείμενα της κλάσης Circle**



# Κατασκευή αντικειμένων

- Η δήλωση ενός αντικειμένου κάποιου τύπου, π.χ., Circle a δεν σημαίνει και δέσμευσή του στη μνήμη.
- Η κατασκευή/δημιουργία ενός νέου αντικειμένου γίνεται με τη χρήση της εντολής **new**.
  - `Circle a = new Circle();`
  - `Circle a = new Circle(5.0);`
  - `String s = new String(‘‘asdf’’);` ⇔ `String s = ‘‘asdf’’;`
- Με την κλήση του new, εκτελούνται κάποιες ειδικές μέθοδοι που ονομάζονται **κατασκευαστές (constructors)** και είναι υπεύθυνοι για την δημιουργία του αντικειμένου και αρχικοποίηση των πεδίων του.
- Παράδειγμα:

```
class Circle {
  …
  Circle () {
  };
  …
```

# Κατασκευή αντικειμένων (συν.) – Constructors

- Οι κατασκευαστές ΠΡΕΠΕΙ να έχουν το ίδιο όνομα με την κλάση.
- Μπορούν να υπάρχουν πολλοί κατασκευαστές με το ίδιο όνομα (overloading) αλλά όχι με την ίδια υπογραφή → διαφορετικό αριθμό και τύπο παραμέτρων.
- Οι κατασκευαστές ΔΕΝ επιστρέφουν κάτι.
- Μπορεί να δέχονται ή να μην δέχονται παραμέτρους
  - Χωρίς παραμέτρους (No-arg constructor), π.χ., `Circle(){ … }` Ονομάζεται επίσης και default constructor
  - Με παραμέτρους, π.χ., `Circle(double newRadius){ … }`
- Σε περίπτωση που δεν δηλωθεί constructor, δημιουργείται αυτόματα ο default (no-args) constructor.
- Μετά το κάλεσμα του constructor, μπορεί να γίνει η “διασύνδεσή” της μεταβλητής με τη μνήμη που έχει δεσμευτεί για το αντικείμενο.

# Πρόσβαση στα πεδία/δεδομένα αντικειμένων

- Η πρόσβαση στα πεδία και τις μεθόδους των αντικειμένων, γίνεται με τη χρήση της τελείας (.).
- Παραδείγματα: `(Circle a = new Circle(5.0);)`
  - Πρόσβαση στα δεδομένα: `a.radius`
  - Πρόσβαση στα δεδομένα: `a.getArea();`
- **Ερώτηση:** Τι γίνεται στις περιπτώσεις που ο κατασκευαστής δεν δίνει τιμές στα πεδία του αντικειμένου;
- **Απάντηση:** Τα πεδία αρχικοποιούνται με τιμές default:
  - Boolean → false
  - Strings → null
  - Αριθμητικοί τύποι (π.χ., int, float) → 0 ή 0.0
  - Char → '\u0000'
- **ΠΡΟΣΟΧΗ: Η αρχικοποίηση με default δεν ισχύει για μεταβλητές που δηλώνονται μέσα σε μία μέθοδο**

```
class Circle {
  double radius = 1.0;

  Circle () {
  };
  …

}
```

```
Test() {
  int x;
  System.out.println(x);
```

COMPILE ERROR!!

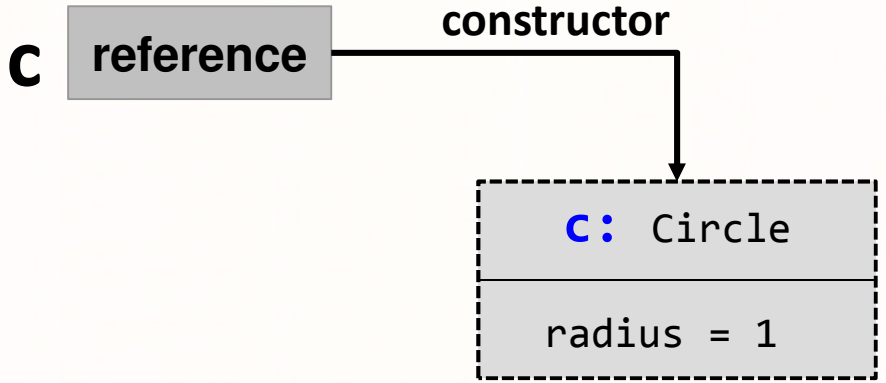
# Ανάθεση

- Η ανάθεση σε μεταβλητές αρχέγονων τύπων είναι απλή και ακολουθεί γνωστούς κανόνες.
- Η ανάθεση ενός αντικειμένου **A** σε ένα άλλο **B**, σημαίνει την ανάθεση του χειριστηρίου (**handle**) **α,** του **A,** στο χειριστήριο **b** του **B: b = a;**
- Αποτέλεσμα της ανάθεσης αυτής είναι το b και το a να δείχνουν στο ίδιο αντικείμενο **(A),** ενώ η πρόσβαση στο αντικείμενο **B** έχει χαθεί.
- Το φαινόμενο αυτό λέγεται **aliasing (**ψευδωνυμία) και είναι αποτέλεσμα του πως η Java χειρίζεται τα αντικείμενα.
- Η **ψευδωνυμία** προκύπτει και όταν περνάμε αντικείμενα σαν **παραμέτρους** σε μια μέθοδο, π.χ., f( Circle a ) { … }
- Σε ορισμένες γλώσσες η μέθοδος **f()** θα έκανε ένα αντίγραφο του ορίσματος **Circle a,** μέσα στο πεδίο ισχύος (scope) της (**pass-by-value**).
- Στην Java όμως περνάμε το χειριστήριο σαν παράμετρο (**pass-by-reference**), κι έτσι μέσα στην **f()** αλλάζουμε το ίδιο το αντικείμενο.

# Ανάθεση: Αρχέγονοι Τύποι vs. Τύποι Αντικειμένων

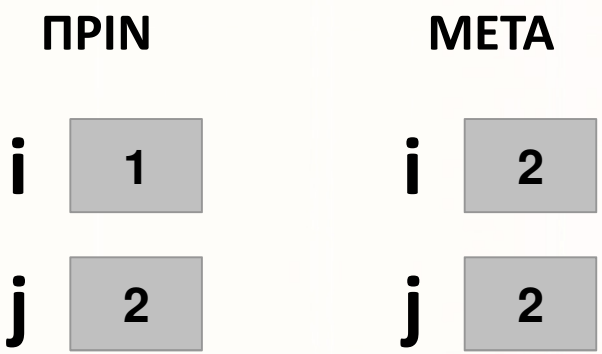
**Primitive type** e.g., int i = 1


**Object (reference) type** e.g., Circle c



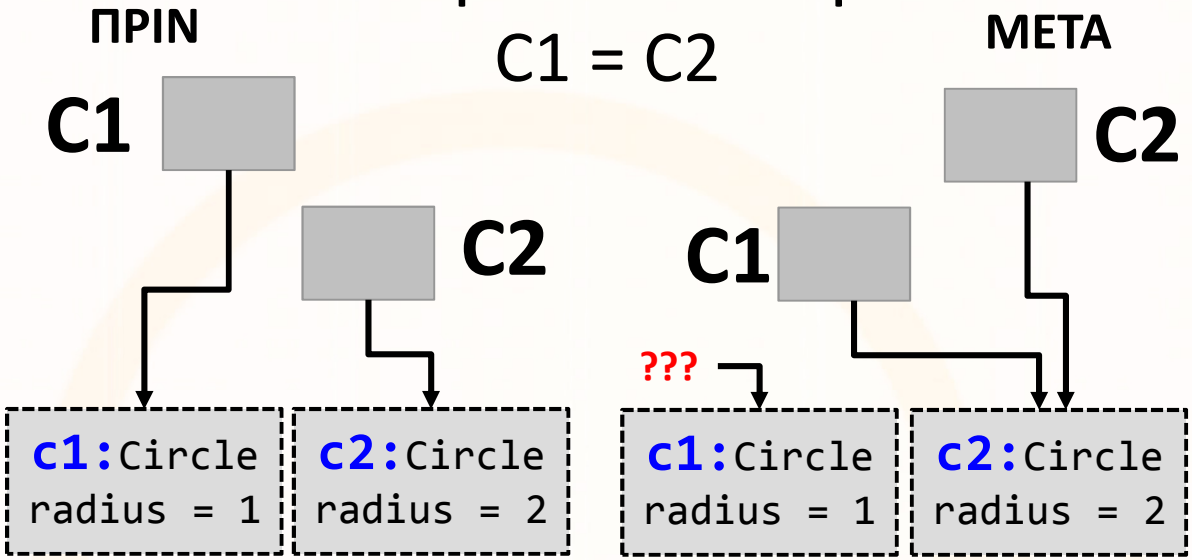

## Ανάθεση σε Πρωτόγονο Τύπο

i=j




## Ανάθεση σε Τύπο Αντικείμενο

C1 = C2



# Αποκομιστής Σκυβάλων (garbage collector)

- Στο προηγούμενο παράδειγμα, μετά την ανάθεση c1=c2 δεν υπάρχει μεταβλητή που να δείχνει στο αντικείμενο του c1.
- Το αντικείμενο αυτό είναι για τα σκουπίδια ☺
- Η JVM συλλέγει αυτόματα τα σκουπίδια με τον garbage collector.
  - Αυτό σημαίνει αυτόματη διαχείριση μνήμης αντίθετα με C++.
  - Επίσης σημαίνει περισσότερο χρόνο για προγραμματισμό
- Ένα αντικείμενο θεωρείται επιλέξιμο από τον garbage collector εάν:
  - Όλες οι αναφορές του είναι null
  - Το αντικείμενο είναι δημιουργημένο σε ένα block και η αναφορά είναι out-of-scope μόλις τελειώσει η εκτέλεση του block
  - Το αντικείμενο έχει αναφορά από ένα αντικείμενο πατέρα μόνο, και όλες οι αναφορές στον πατέρα είναι null (αυτό ισχύει αναδρομικά)
  - Το αντικείμενο αναφέρεται μόνο μέσω ενός WeakHashMap
- Ο garbage collector έχει άμεση σχέση με τα διάφορες στάδια του κύκλος ζωής ενός αντικειμένου

# Κύκλος Ζωής αντικειμένου

Τυπικά στάδια ζωής αντικειμένου:

1. **Created:** Δέσμευση μνήμης, κάλεσμα constructor (ή/και superclass constructor), αρχικοποίηση των μεταβλητών
2. **In use (strongly reachable):** Τα αντικείμενα που έχουν τουλάχιστο ένα strong reference
3. **Invisible:** Τα αντικείμενα που ΔΕΝ έχουν κανένα strong reference αλλά μπορεί να υπάρχουν weak references
4. **Unreachable:** Τα αντικείμενα που ΔΕΝ έχουν κανένα strong ref.
5. **Collected:** Ο garbage collector αναγνωρίζει ένα unreachable αντικείμενο
6. **Finalized:** Τα αντικείμενα έχουν εκτελέσει τη finalize method
7. **Deallocated:** Το JVM είναι έτοιμο για αποδέσμευση της μνήμης του αντικειμένου

Πως διαχειρίζεται το JVM τη μνήμη για το κάθε στάδιο ζωής;

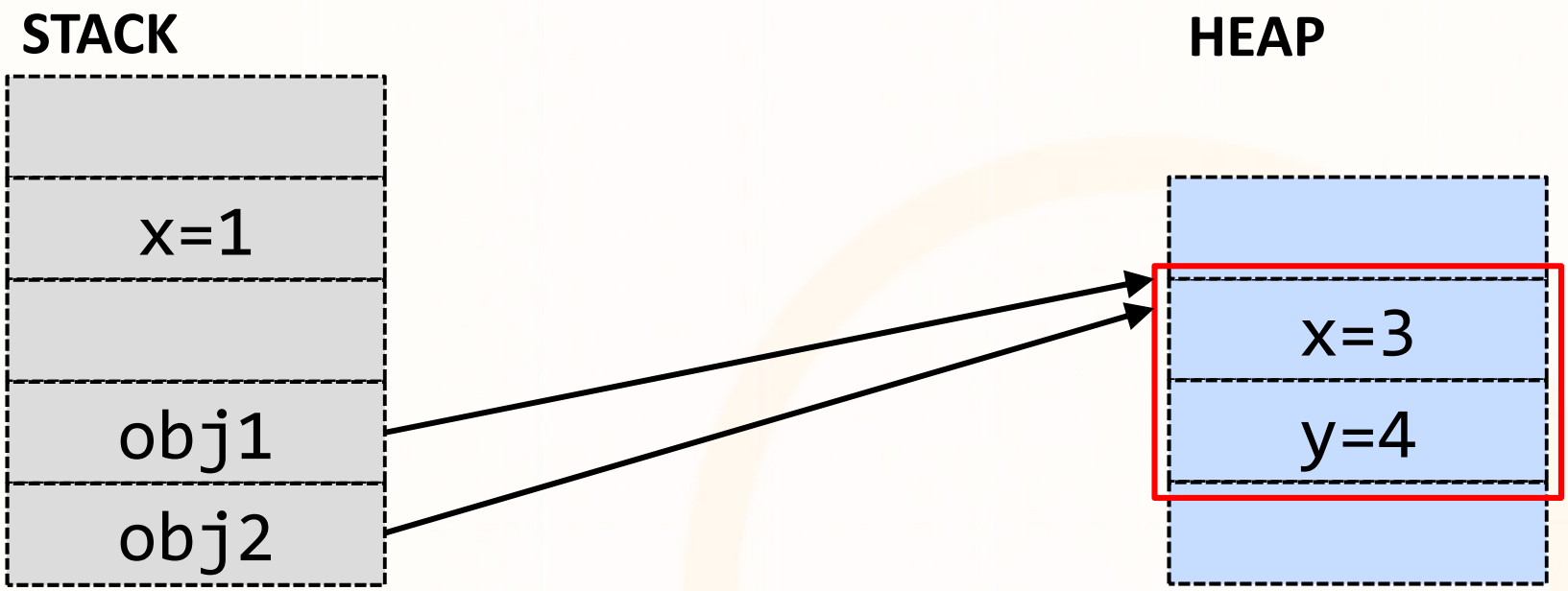
# Διαχείριση Μνήμης

- **Στοίβα (stack)**
  - Μέρος της μνήμης που παρέχει ταχύτητα.
  - Μεγαλώνει/Μικραίνει ανάλογα με τις μεθόδους που καλούνται
  - Αποθηκεύει τους τοπικούς (method/block) αρχέγονους τύπος
  - Αποθηκεύει τις τοπικές (method/block) reference μεταβλητές
- **Σωρός (Heap)**
  - Πιο αργό μέρος της μνήμης που παρέχει όμως χωρητικότητα
  - Αποθηκεύει τις μεταβλητές του αντικειμένου (instanse variables)
  - Αποθηκεύει αντικείμενα που δημιουργούνται με το new
  - Μεγαλώνει/Μικραίνει ανάλογα με τα αντικείμενα που δημιουργούνται
- **Στατική περιοχή (Static Area) (μέρος του Heap)**
  - Αποθηκεύει global και static μεταβλητές

# Διαχείριση Μνήμης (συν.)

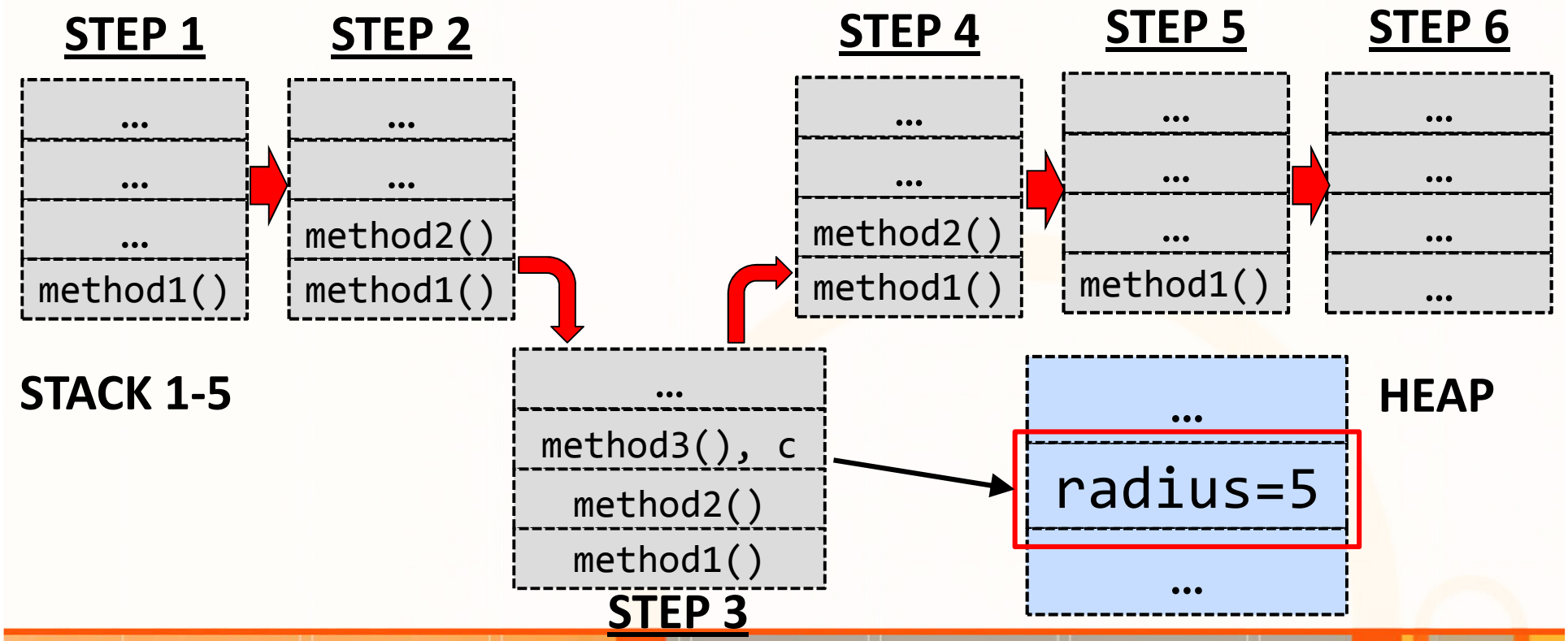
```
class Obj{
  int x;   int y;
  Obj (int a, int y) {
    x=a;
    y=b;
  }
}
```

```
…
{
 int x = 1;
 Obj obj1 = new Obj(3, 4);
 Obj obj2 = obj1;
}
…
```

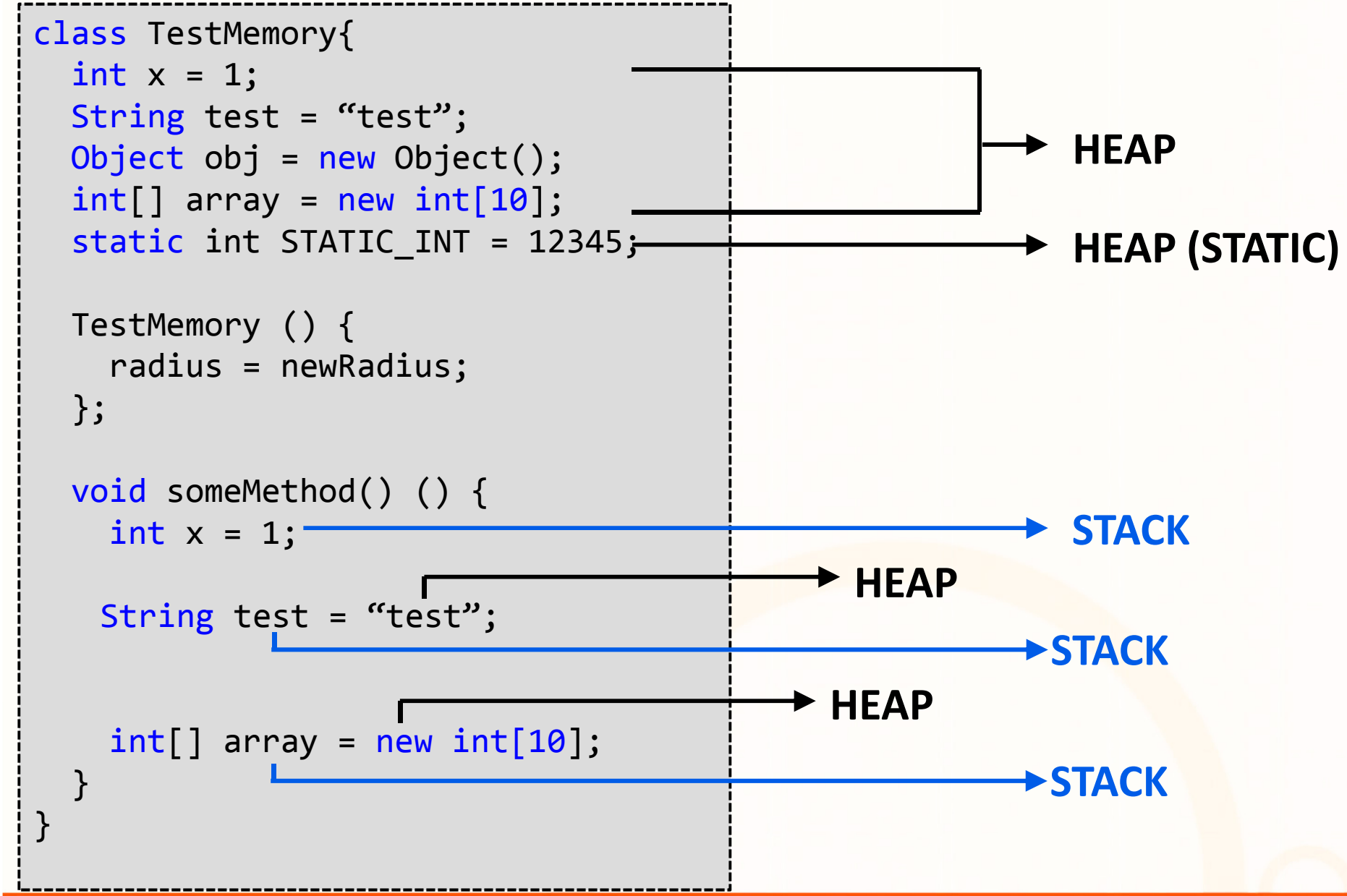


# Διαχείριση Μνήμης (συν.)

```
class StackMethods{
  void method1(){ method2(); }
  void method2(){ method3(); }
  void method3(){ Circle c = new Circle( 5 ); }
}
```



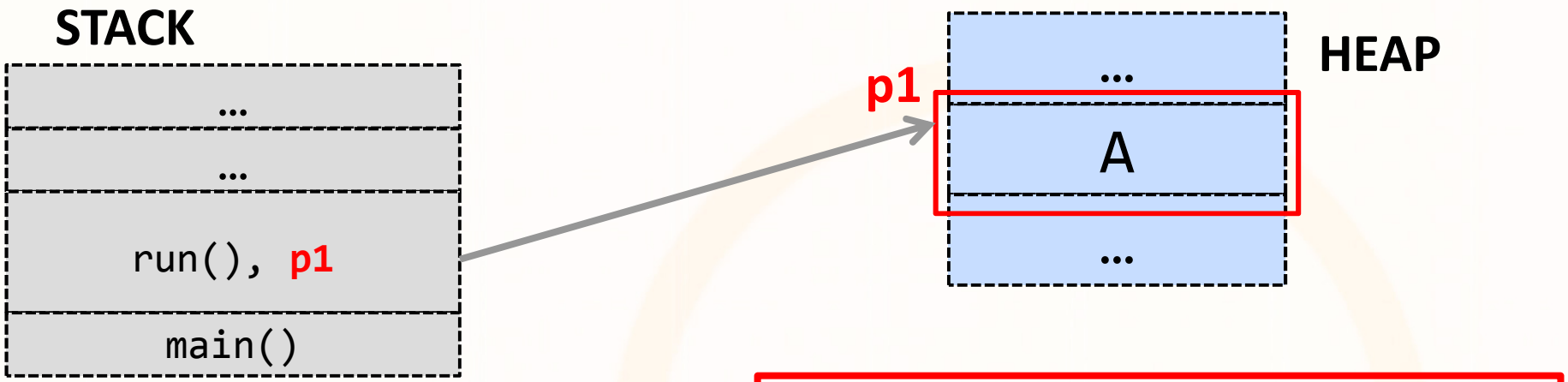
# Διαχείριση Μνήμης (συν.)



# Αποκομιστής Σκυβάλων (garbage collector) (συν.)

- Ερώτηση 1. Τι θα γίνει μόλις τελειώσει η εκτέλεση του try…catch;

```
public void run() {
  try {
     person p1 = new person("A");
     p1.showDetails();
  } catch (Exception ex) { }
  while( true ) { //do something }
}
```

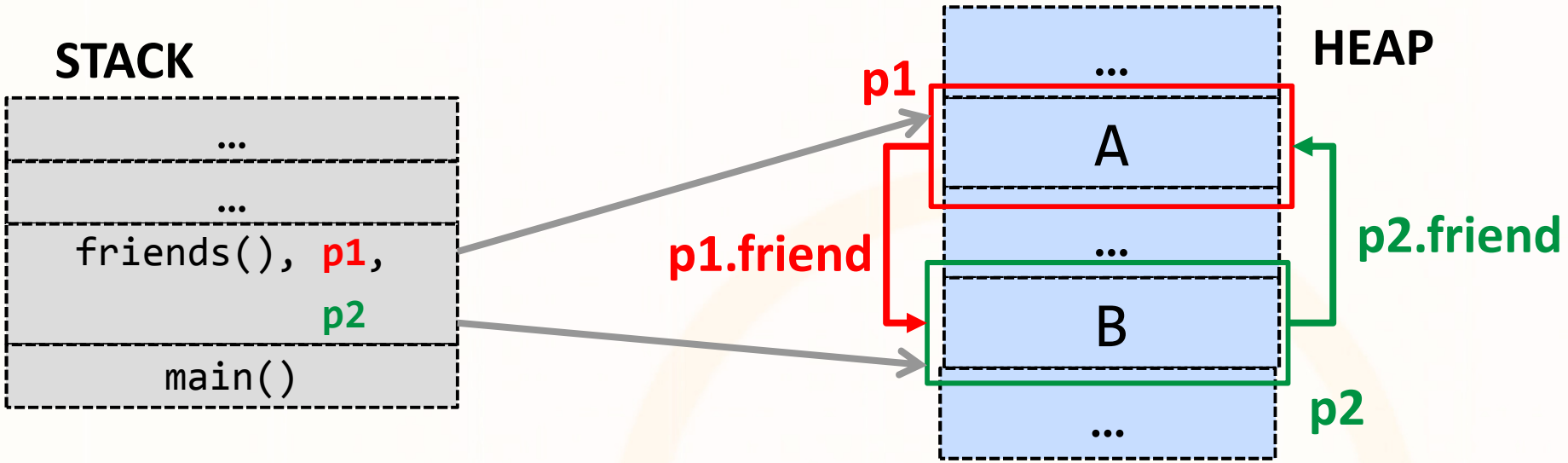



**Η μεταβλητή p1 γίνεται invinsible**

# Αποκομιστής Σκυβάλων (garbage collector) (συν.)

- Ερώτηση 2. Τι θα γίνει μόλις τελειώσει η εκτέλεση της friends();

```
public void friends {
   person p1 = new person("A");
   person p2 = new person("B");
   p1.friend = p2;
   p2.friend = p1;
}
```




**Η μεταβλητές p1 και p2 γίνονται unreachable**