ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Αναφορές

Αντικείμενα ως ορίσματα

Η μνήμη του υπολογιστή

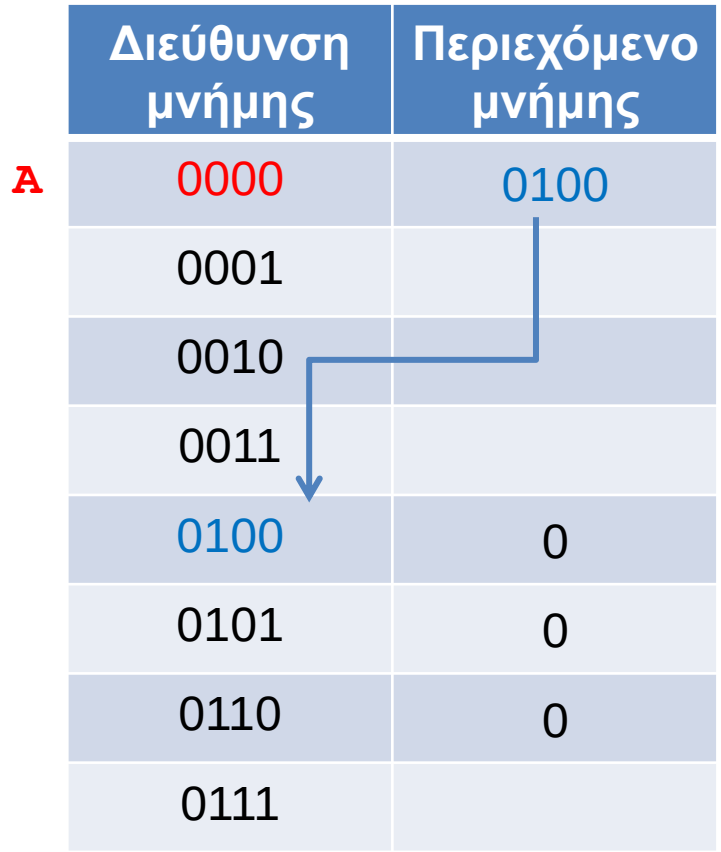
- Η κύρια μνήμη (main memory - RAM) του υπολογιστή κρατάει τα δεδομένα για την εκτέλεση των προγραμμάτων.
  - Η μνήμη είναι προσωρινή, τα δεδομένα χάνονται όταν ολοκληρωθεί το πρόγραμμα.
- Κάθε μεταβλητή πιάνει διαφορετικό χώρο στη μνήμη ανάλογα με τον τύπο της
  - Μπορούμε να προσπελάσουμε την μεταβλητή αν ξέρουμε τη διεύθυνση του πρώτου byte και το μέγεθος της.
  - Άρα η θέση μνήμης της μεταβλητής αποτελείται από μία διεύθυνση και το μέγεθος.
- Από εδώ και πέρα θα χρησιμοποιούμε τον όρο θέση μνήμης ανεξάρτητα από το μέγεθος της μεταβλητής.

Διεύθυνση μνήμης	Περιεχόμενο μνήμης
0x0000	‘a’
0x0100	5
0x0110	8.5
0x0111	“bob”
0x1000	0x1111
0101	
0110	
0x1111	

Αποθήκευση αντικειμένων

- Οι θέσεις μνήμης των αντικειμένων κρατάνε μια διεύθυνση στο χώρο στον οποίο αποθηκεύεται το αντικείμενο
- Η διεύθυνση αυτή λέγεται αναφορά.
- Ο χώρος μνήμης του αντικειμένου δεσμεύεται με την εντολή new.

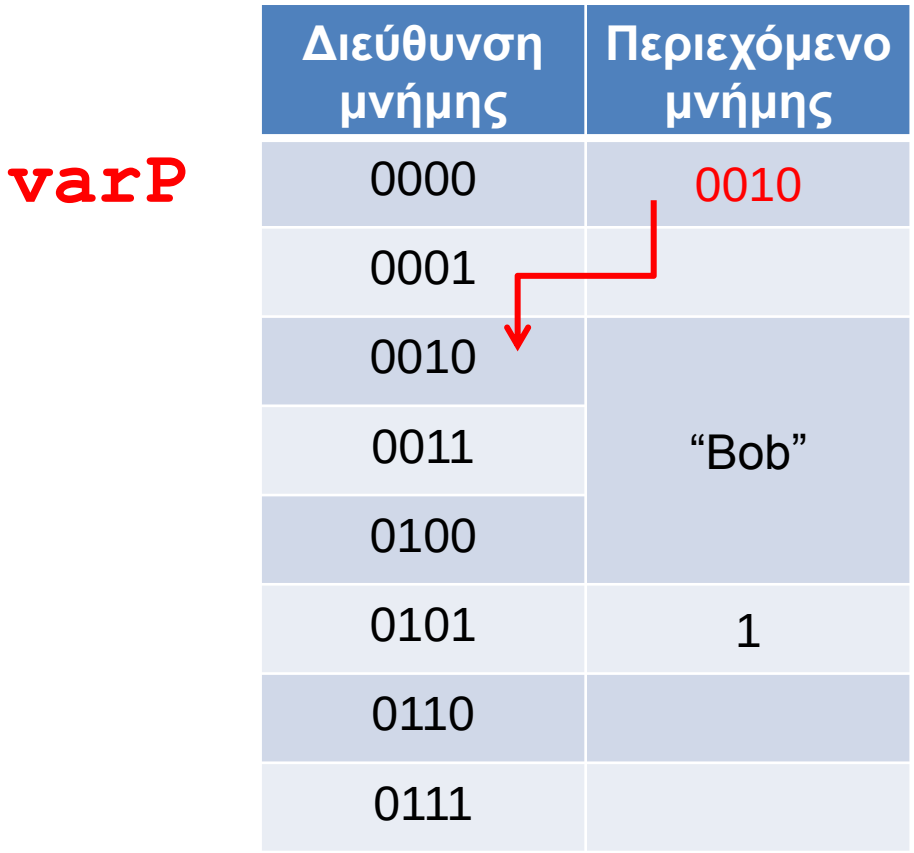
```
int[] A = new int[3];
```



```
public class Person
{
    private String name;
    private int number;

    public Person(String initName, int initNumber){
        name = initName;
        number = initNumber;
    }

    public void set(String newName, int newNumber){
        name = newName;
        number = newNumber;
    }

    public String toString( ){
        return (name + " " + number);
    }
}
```

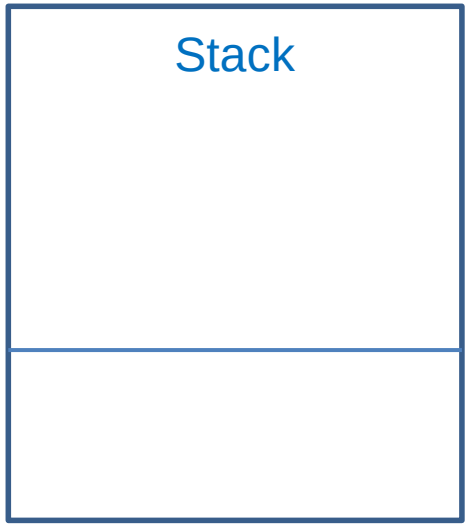
Παράδειγμα

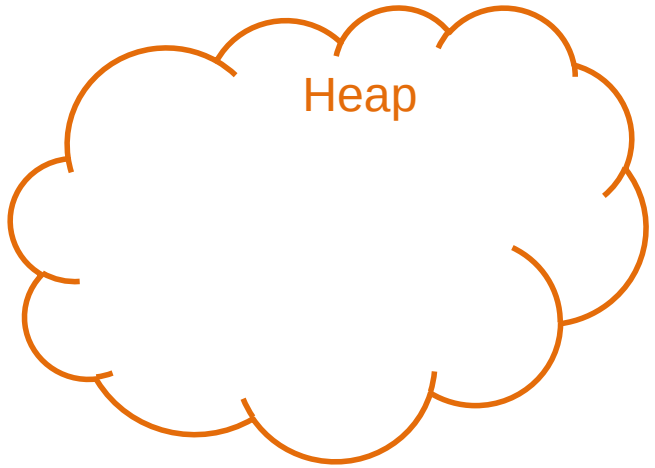
```
Person varP = new Person("Bob", 1);
```



Διαχείριση μνήμης από το JVM

- Η μνήμη χωρίζεται σε δύο τμήματα
  - Τη στοίβα (stack) που χρησιμοποιείται για να κρατάει πληροφορία για τις τοπικές μεταβλητές κάθε μεθόδου/block.
  - Το σωρό (heap) που χρησιμοποιείται για να δεσμεύουμε μνήμη για τα αντικείμενα



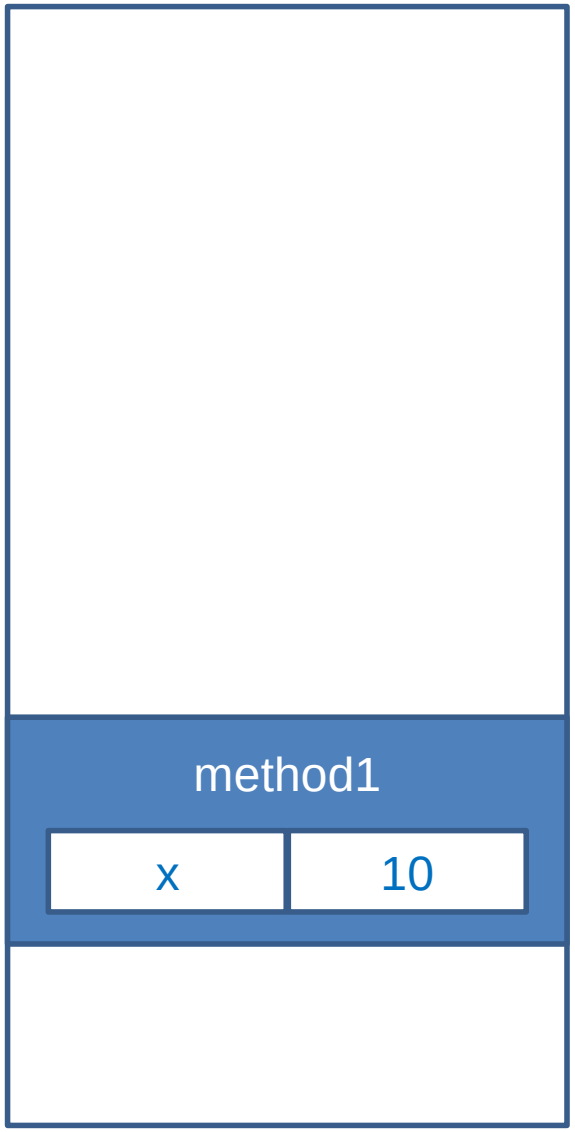


Stack

- Κάθε φορά που καλείται μία μέθοδος, δημιουργείται ένα «πλαίσιο» (frame) για την μέθοδο στη στοίβα
  - Δημιουργείται ένας χώρος μνήμης που αποθηκεύει τις παραμέτρους και τις τοπικές μεταβλητές της μεθόδου.
- Αν η μέθοδος καλέσει μία άλλη μέθοδο θα δημιουργηθεί ένα νέο πλαίσιο και θα τοποθετηθεί (push) στην κορυφή της στοίβας.
- Όταν βγούμε από την μέθοδο το πλαίσιο αφαιρείται (pop) από την κορυφή της στοίβας και επιστρέφουμε στην προηγούμενη μέθοδο
- Στη βάση της στοίβας είναι η μέθοδος main.

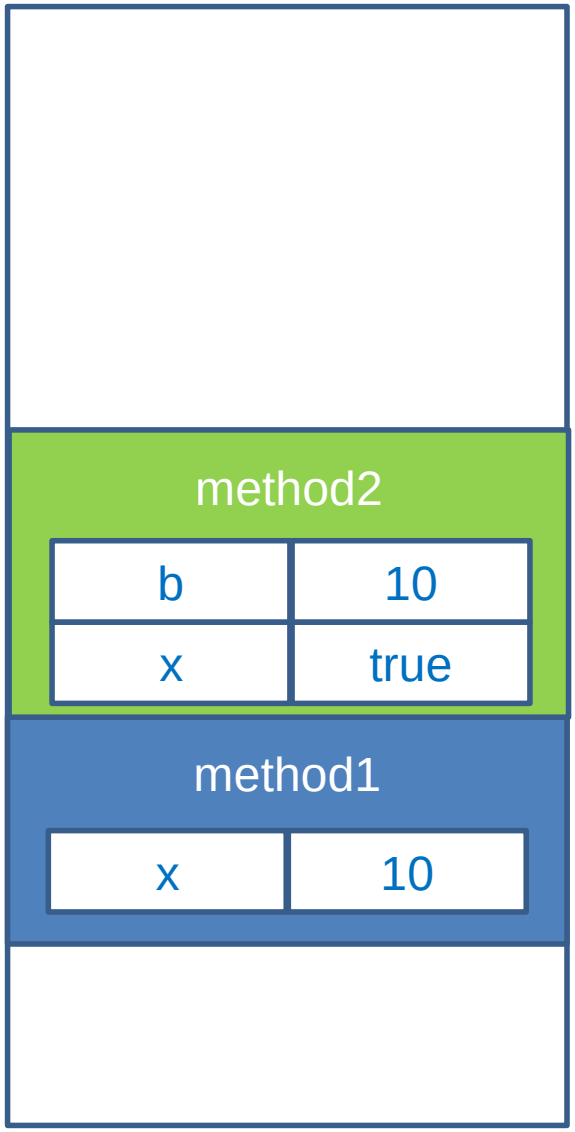
Παράδειγμα

```
public void method1(){
   int x = 10;
   method2(x);
}
```



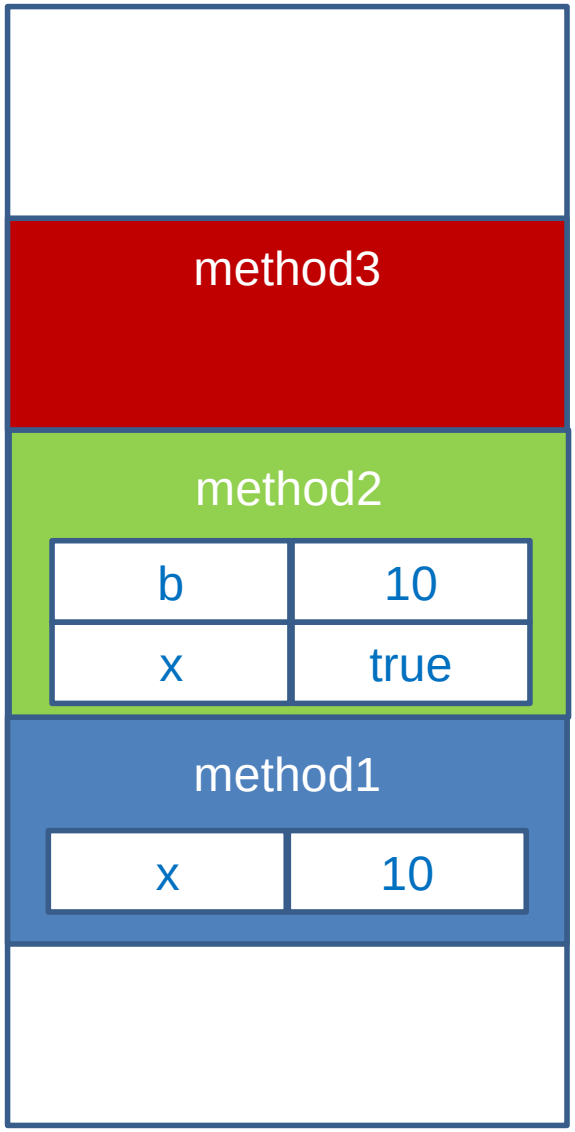
Παράδειγμα

```
public void method1(){
   int x = 10;
   method2(x);
}

public void method2(int b){
   boolean x = true;
   method3();
}
```



Παράδειγμα

```
public void method1(){
   int x = 10;
   method2(x);
}

public void method2(int b){
   boolean x = true;
   method3();
}

public void method3()
{…}
```

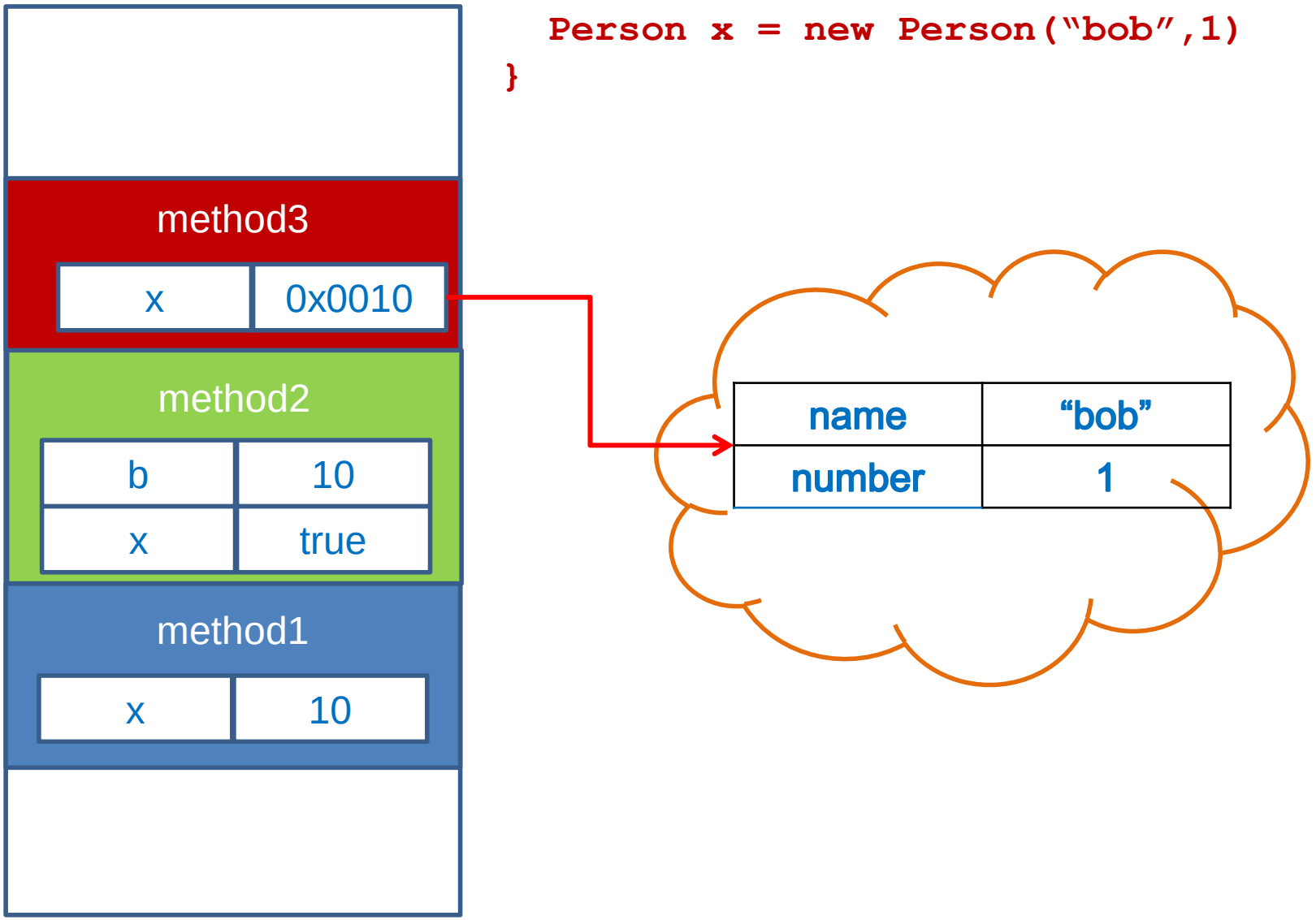


Heap

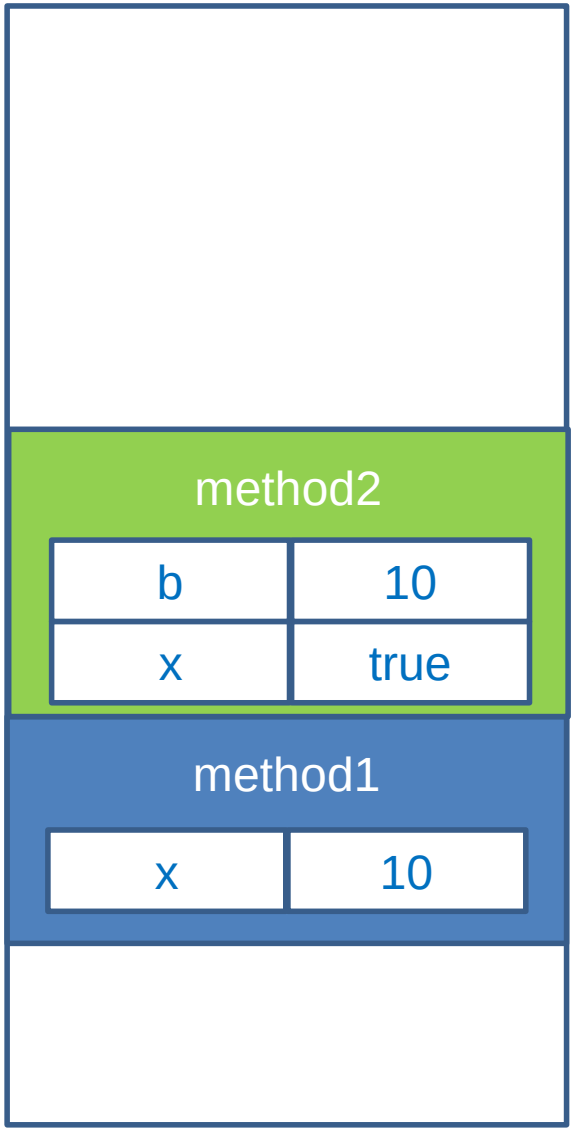
- Όταν μέσα σε μία μέθοδο δημιουργούμε ένα αντικείμενο με την new γίνονται τα εξής
  - στο πλαίσιο (frame) της μεθόδου (στη στοίβα) υπάρχει μια τοπική μεταβλητή που κρατάει την αναφορά στο αντικείμενο
  - Η κλήση της new δεσμεύει χώρο μνήμης στο σωρό (heap) για να κρατήσει τα πεδία του αντικειμένου.
  - Η αναφορά δείχνει στη θέση μνήμης που δεσμεύτηκε.

Παράδειγμα

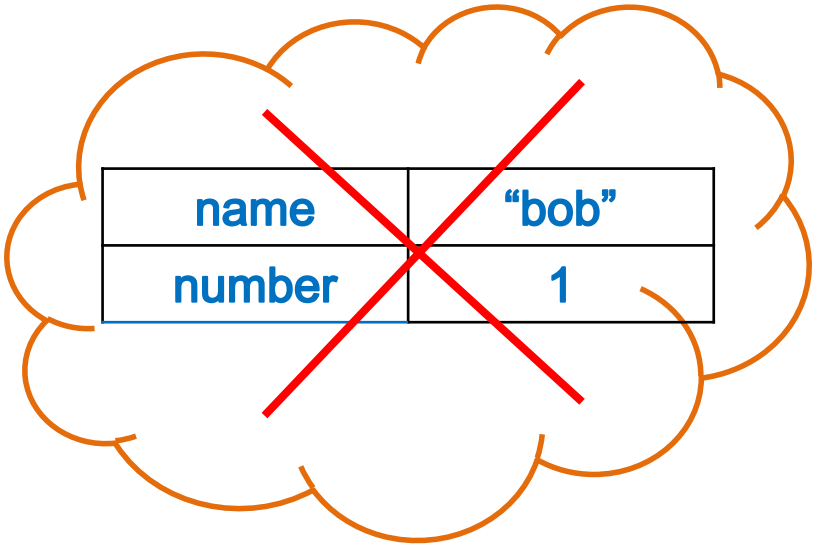
```
public void method3()
{
   Person x = new Person("bob",1)
}
```



Παράδειγμα



Όταν επιστρέφουμε από την μέθοδο method3 η αναφορά προς το αντικείμενο Person παύει να υπάρχει.



Αν δεν υπάρχουν άλλες αναφορές στο αντικείμενο τότε ο garbage collector αποδεσμεύει τη μνήμη του αντικειμένου

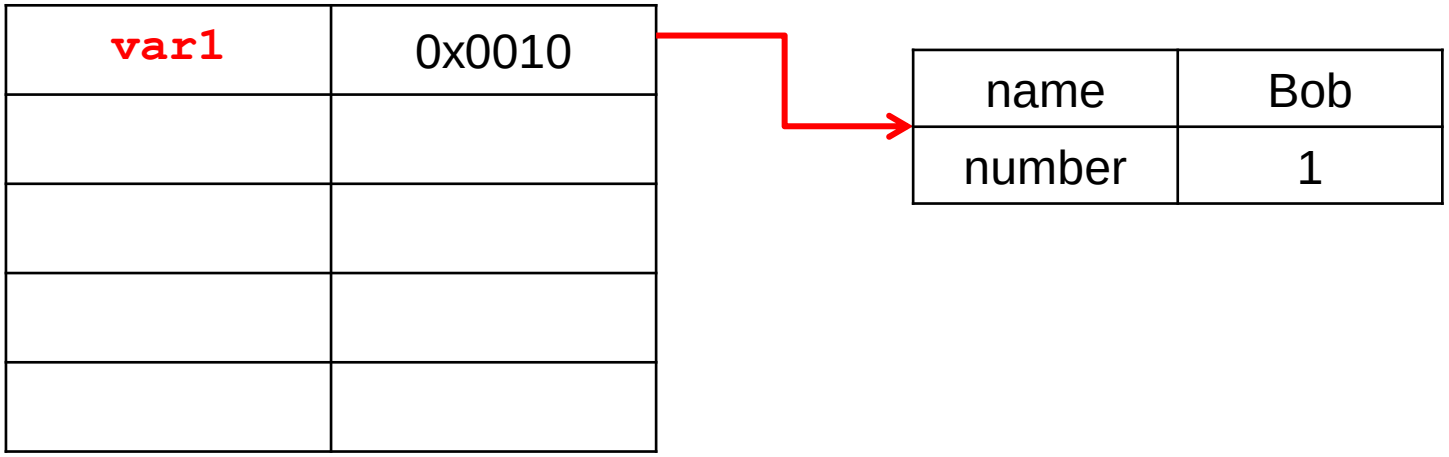
Αναθέσεις μεταξύ αντικειμένων

Τι θα τυπώσει το παρακάτω πρόγραμμα?

```
Person var1 = new Person("Bob", 1);
Person var2;
var2 = var1;
var2.set("Ann",2);
System.out.println(var1);
```

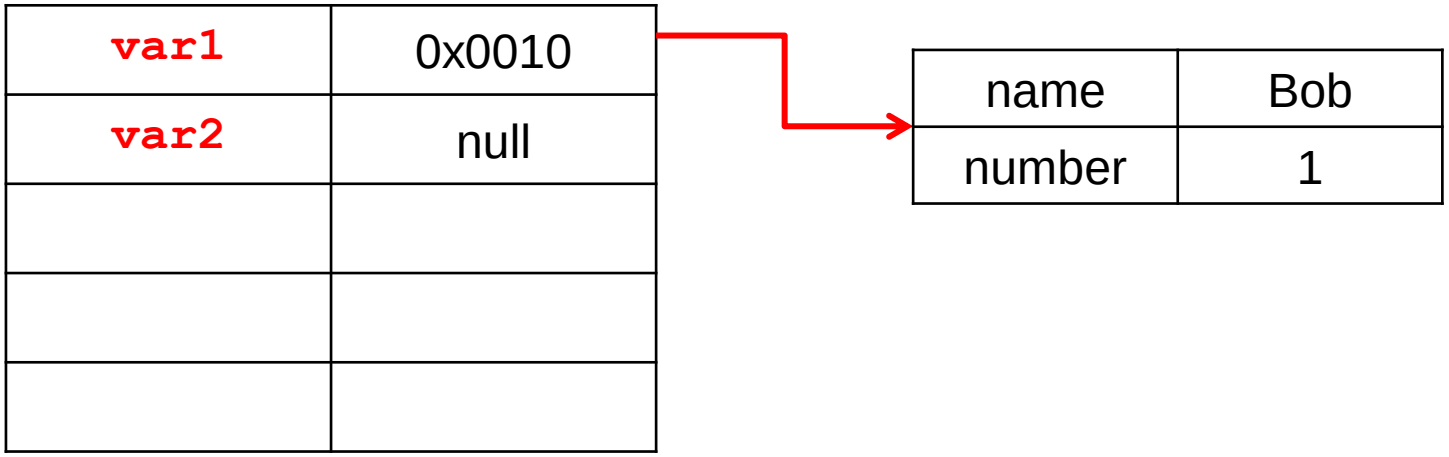
Αναθέσεις μεταξύ αντικειμένων

```
Person var1 = new Person("Bob", 1);
Person var2;
var2 = var1;
var2.set("Ann",2);
System.out.println(var1);
```



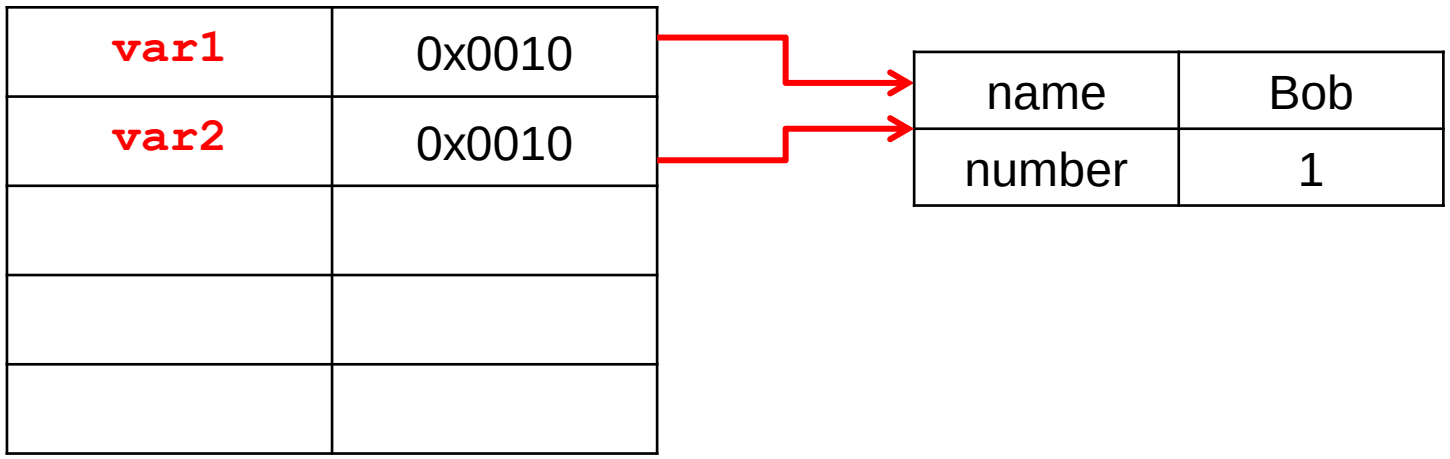
Αναθέσεις μεταξύ αντικειμένων

```
Person var1 = new Person("Bob", 1);
Person var2;
var2 = var1;
var2.set("Ann",2);
System.out.println(var1);
```



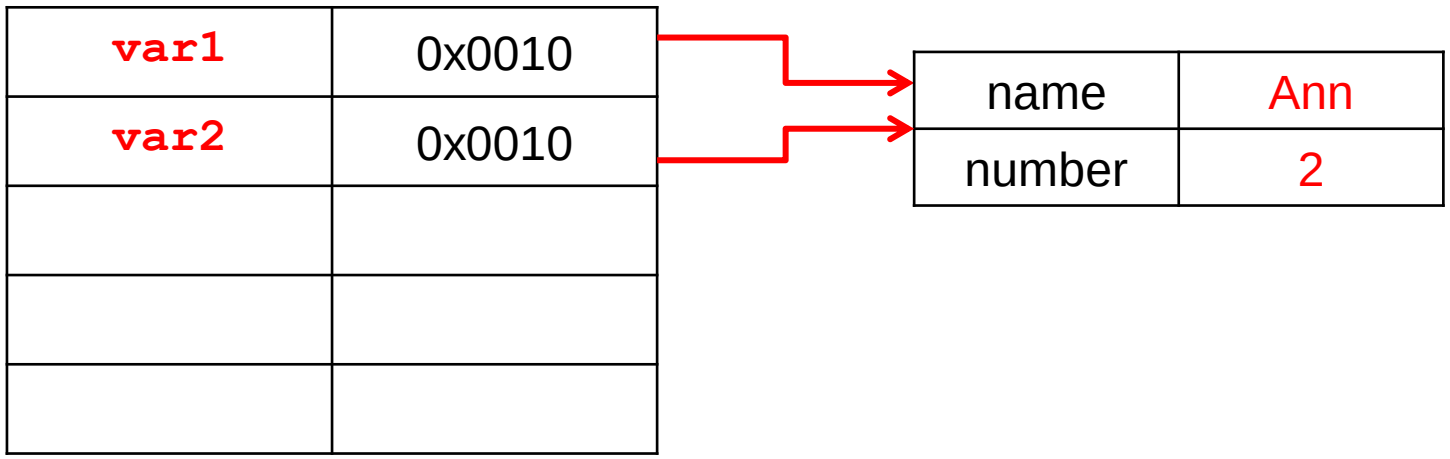
Αναθέσεις μεταξύ αντικειμένων

```
Person var1 = new Person("Bob", 1);
Person var2;
var2 = var1;
var2.set("Ann",2);
System.out.println(var1);
```



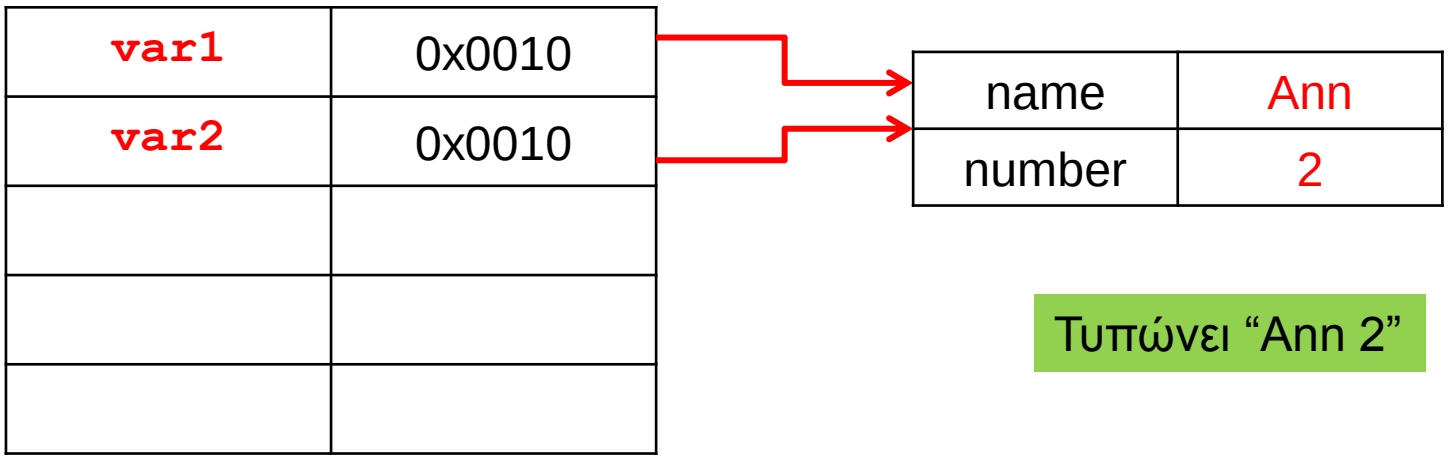
Αναθέσεις μεταξύ αντικειμένων

```
Person var1 = new Person("Bob", 1);
Person var2;
var2 = var1;
var2.set("Ann",2);
System.out.println(var1);
```



Αναθέσεις μεταξύ αντικειμένων

```
Person var1 = new Person("Bob", 1);
Person var2;
var2 = var1;
var2.set("Ann",2);
System.out.println(var1);
```



Τυπώνει “Ann 2”

Αντικείμενα ως παράμετροι

- Όταν περνάμε παραμέτρους σε μία μέθοδο το πέρασμα γίνεται πάντα δια τιμής (call-by-value)
  - Δηλαδή απλά περνάμε τα περιεχόμενα της θέσης μνήμης της συγκεκριμένης μεταβλητής.
  - Για μεταβλητές πρωταρχικού τύπου, αλλαγές στην τιμή της παραμέτρου δεν αλλάζουν την μεταβλητή που περάσαμε σαν όρισμα.

- Τι γίνεται όμως αν η παράμετρος είναι ένα αντικείμενο?
  - Τα περιεχόμενα της θέσης μνήμης μιας μεταβλητής-αντικείμενο είναι μια αναφορά.
  - Αν μέσα στην μέθοδο αλλάξουν τα περιεχόμενα του αντικειμένου (εκεί που δείχνει η αναφορά) τότε αλλάζει και η μεταβλητή που περάσαμε.

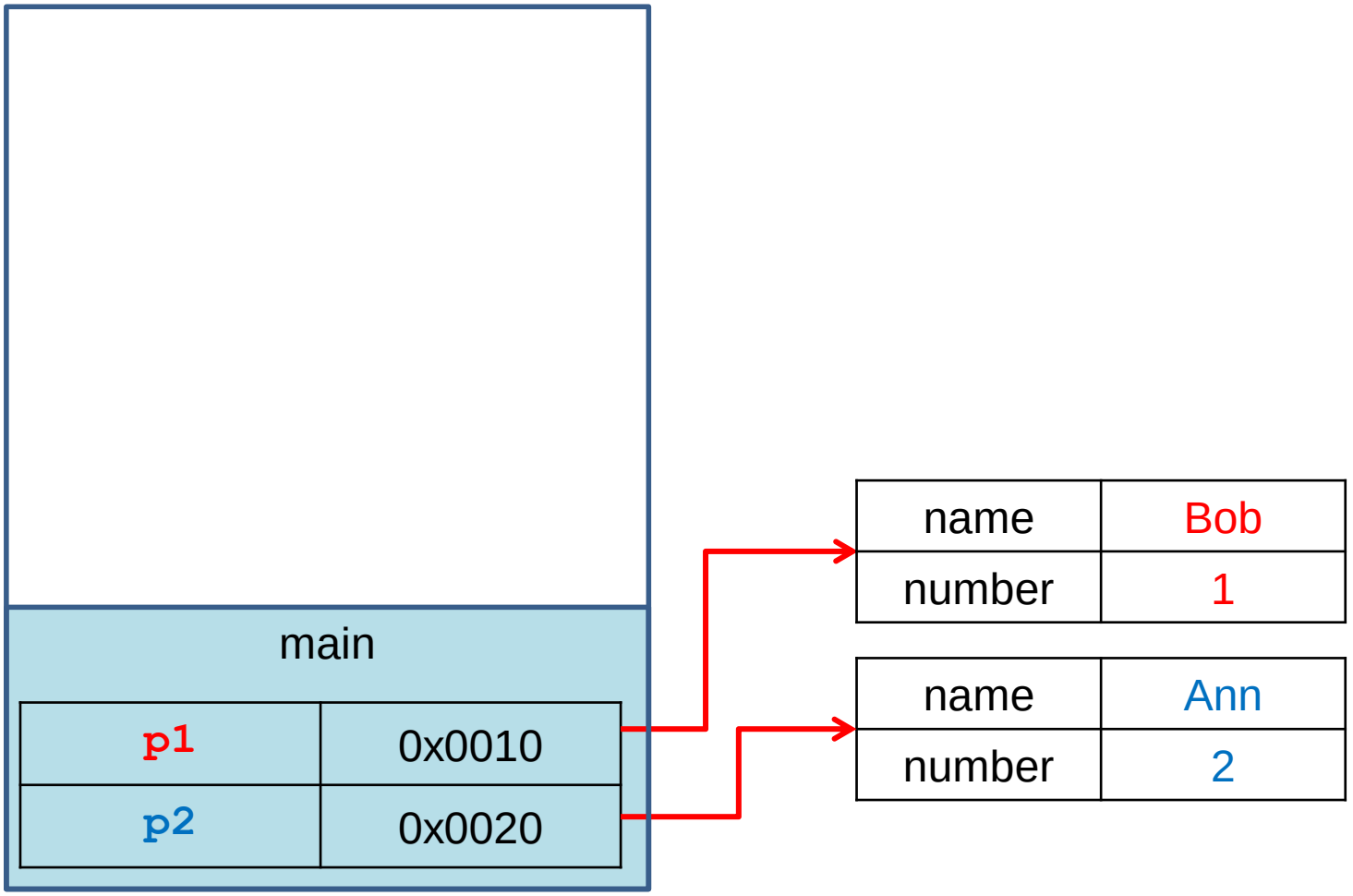
```
public class Person
{
    private String name;
    private int number;

    public Person(String initName, int initNumber){
        name = initName;
        number = initNumber;
    }

    public void set(String newName, int newNumber){
        name = newName;
        number = newNumber;
    }

    public String toString( ){
        return (name + " " + number);
    }

    public void copier( Person other) {
        other.name = this.name;
        other.number = this.number;
    }

}
```

Παράδειγμα

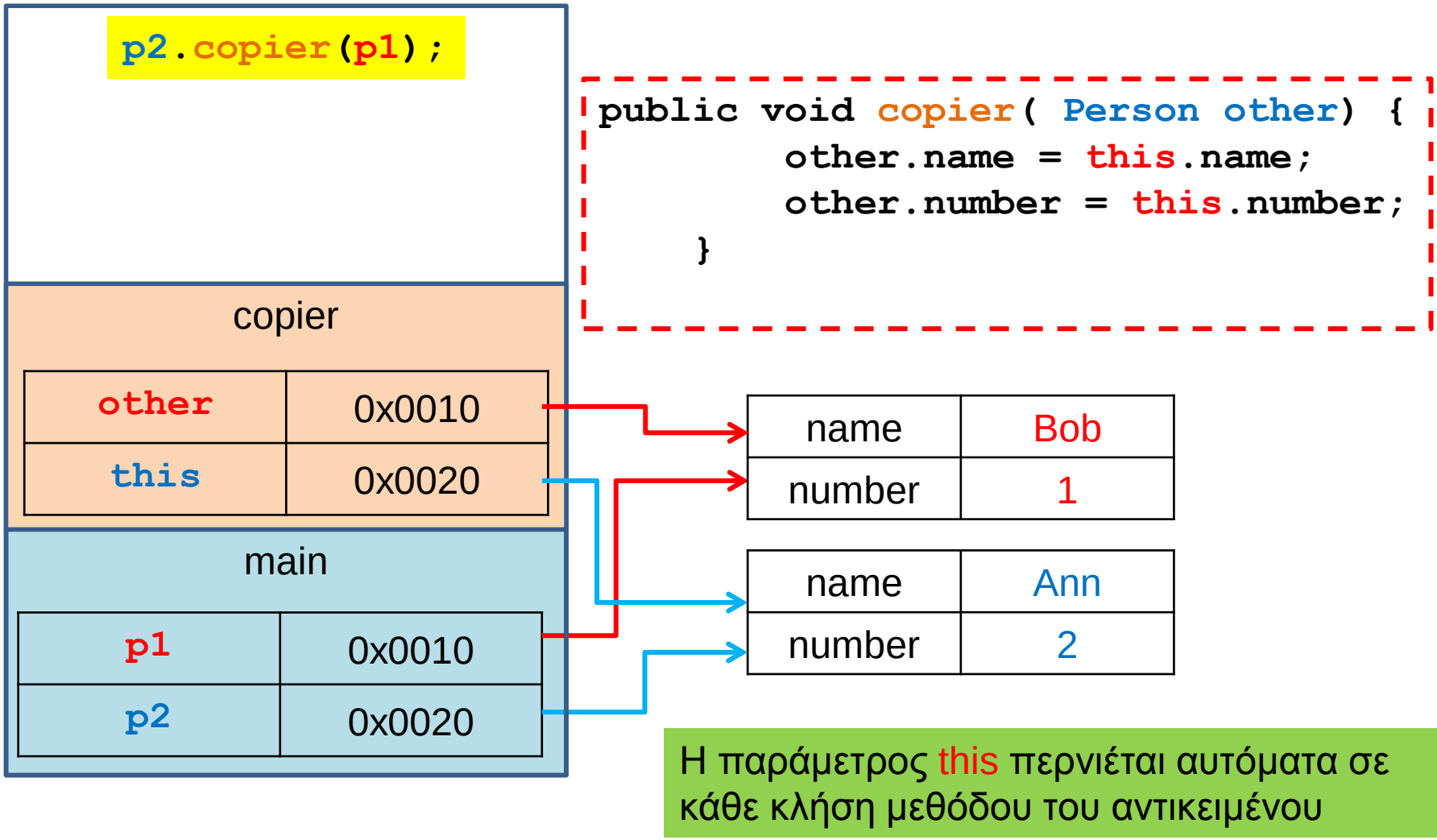
```
public class ClassParameterDemo
{
    public static void main(String[] args)
    {
        Person p1 = new Person("Bob", 1);
        Person p2 = new Person("Ann", 2);
        p2.copier(p1);
        System.out.println(p1);
     }
}
```

Τι θα τυπώσει?

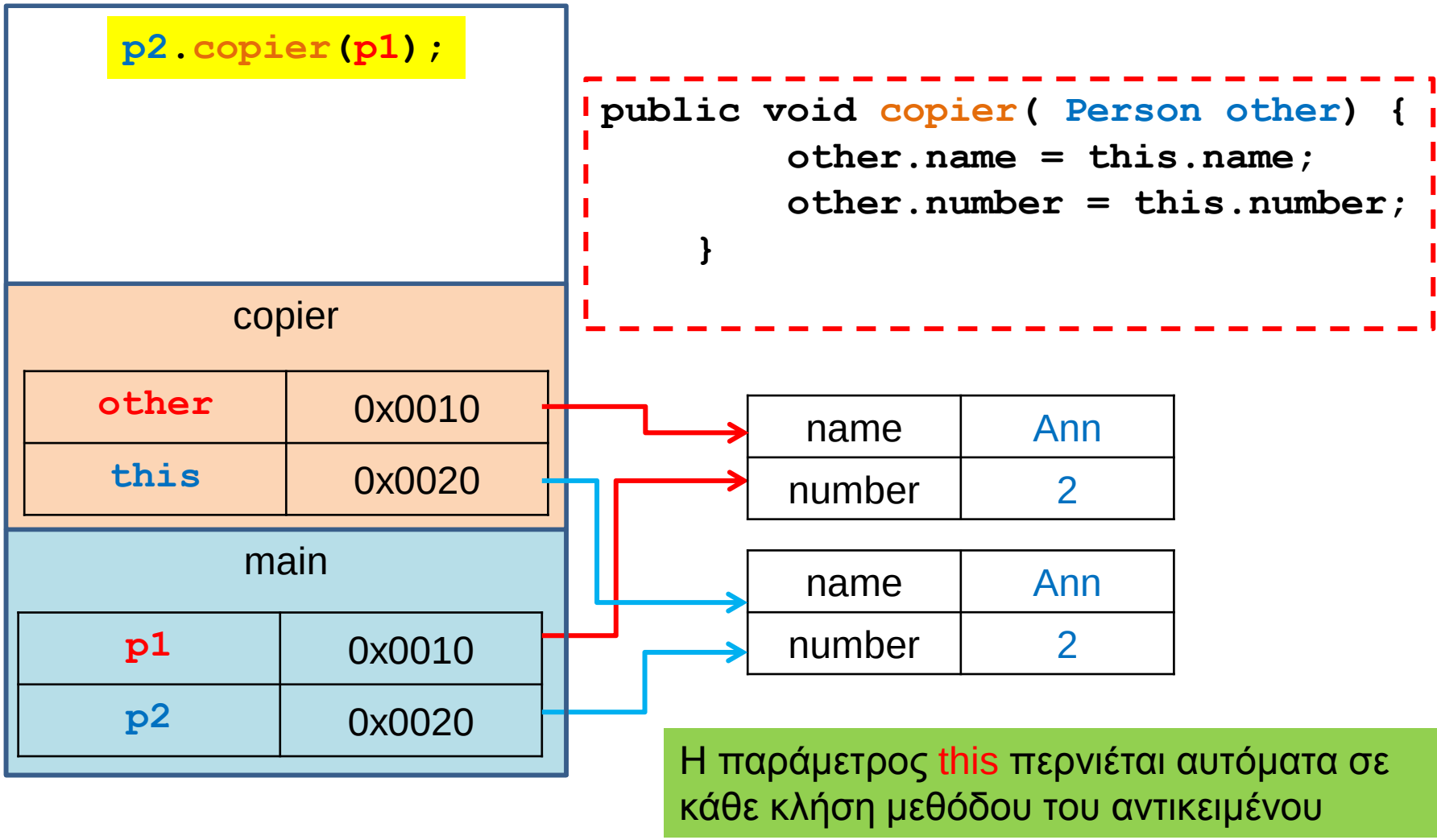
Εξέλιξη του προγράμματος



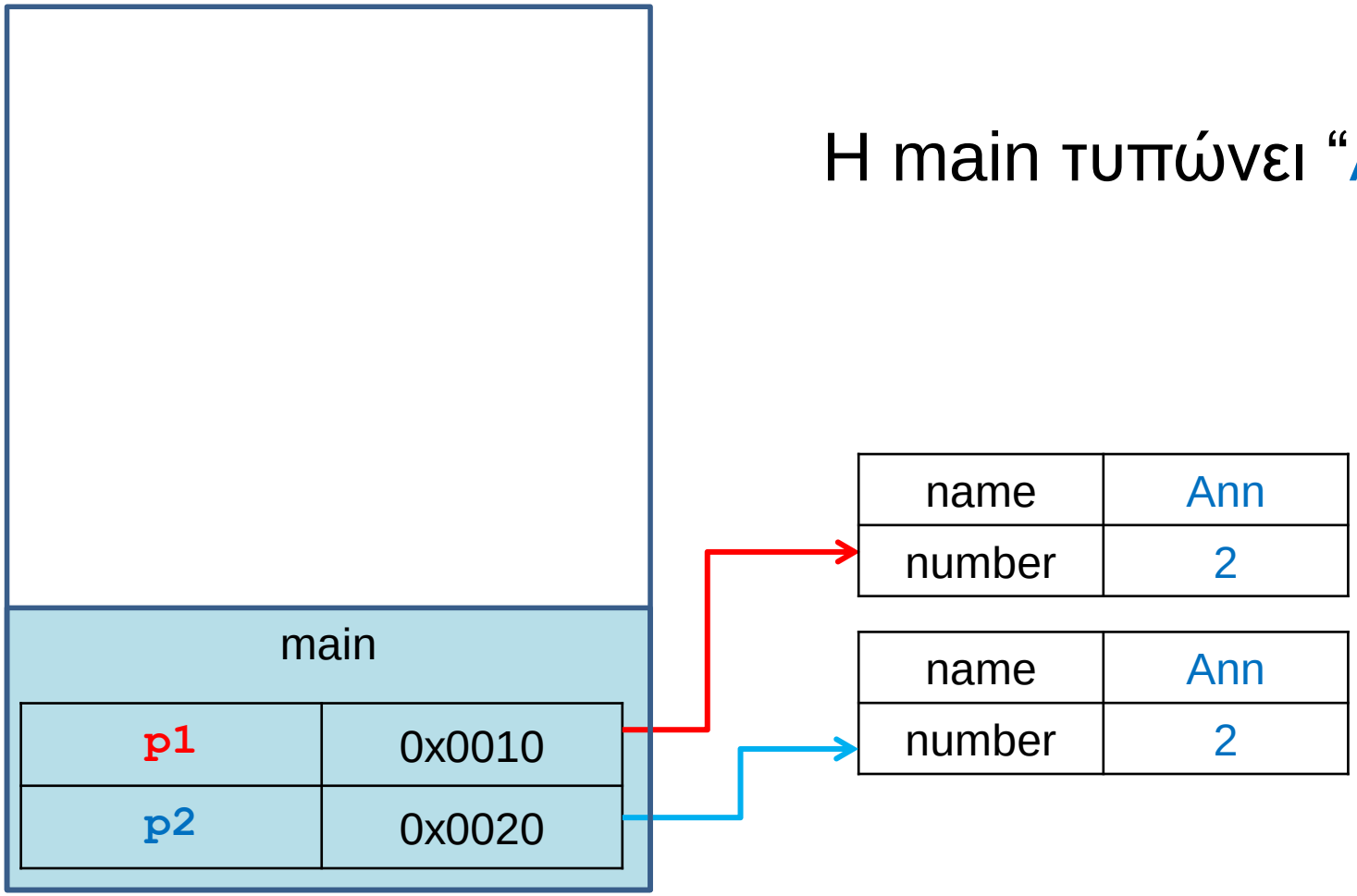
Εξέλιξη του προγράμματος



Εξέλιξη του προγράμματος



Εξέλιξη του προγράμματος



Η main τυπώνει “Ann 2”

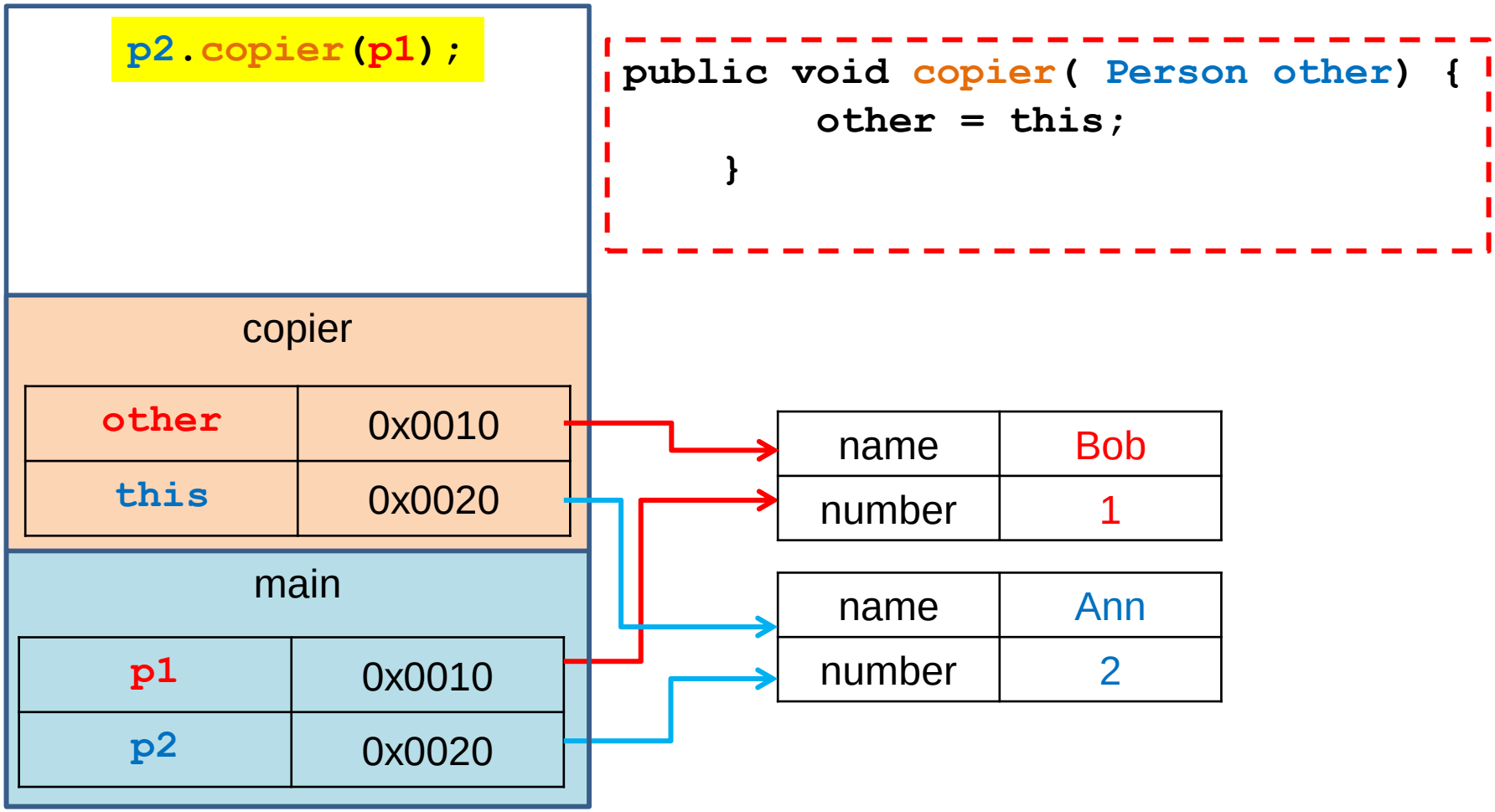
Μια άλλη υλοποίηση της copier

```
public void copier( Person other) {
        other = this;
    }
```

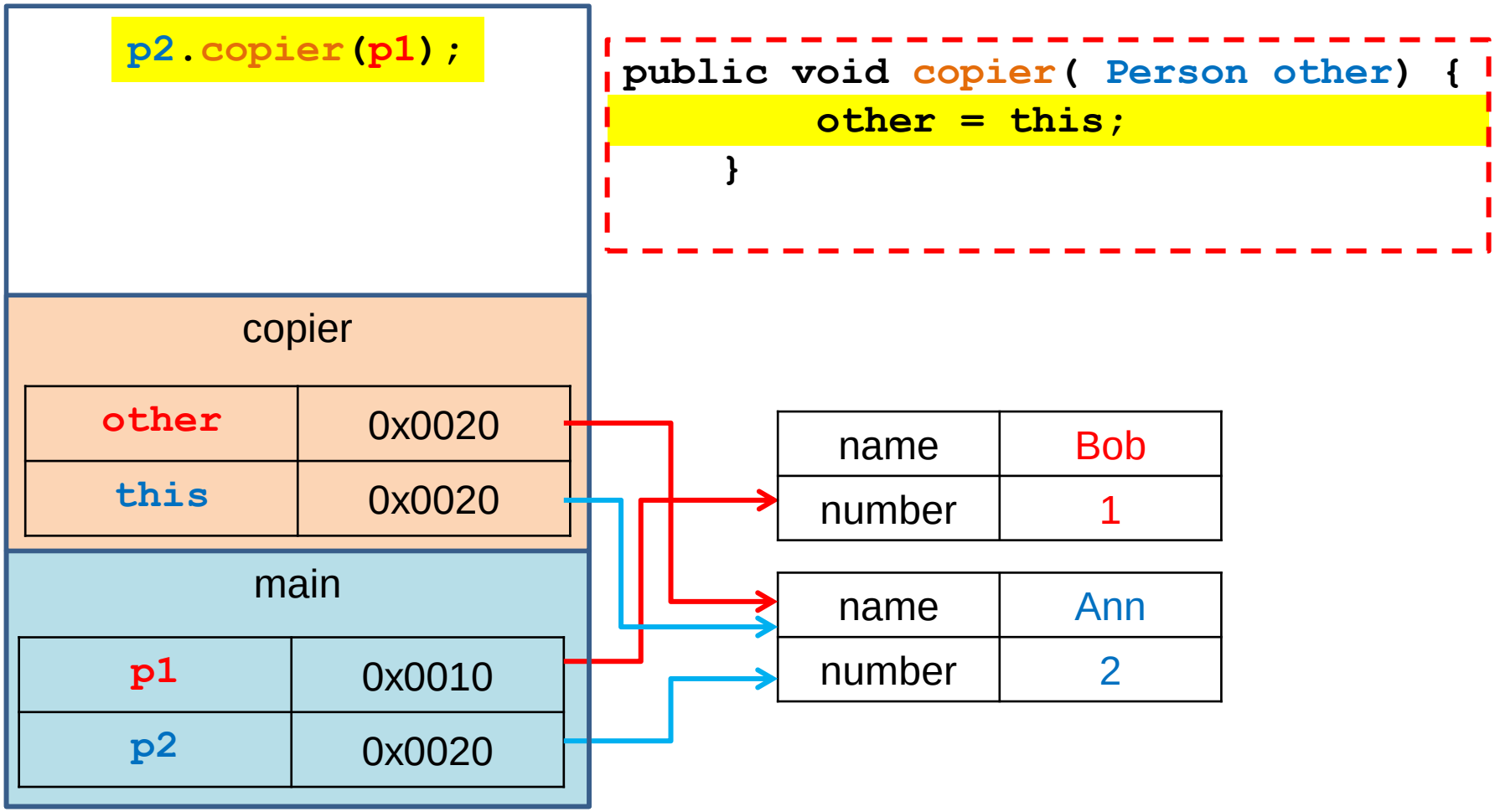
```
public class ClassParameterDemo
{
    public static void main(String[] args)
    {
        Person p1 = new Person("Bob", 1);
        Person p2 = new Person("Ann", 2);
        p2.copier(p1);
        System.out.println(p1);
     }
}
```

Τι θα τυπώσει?

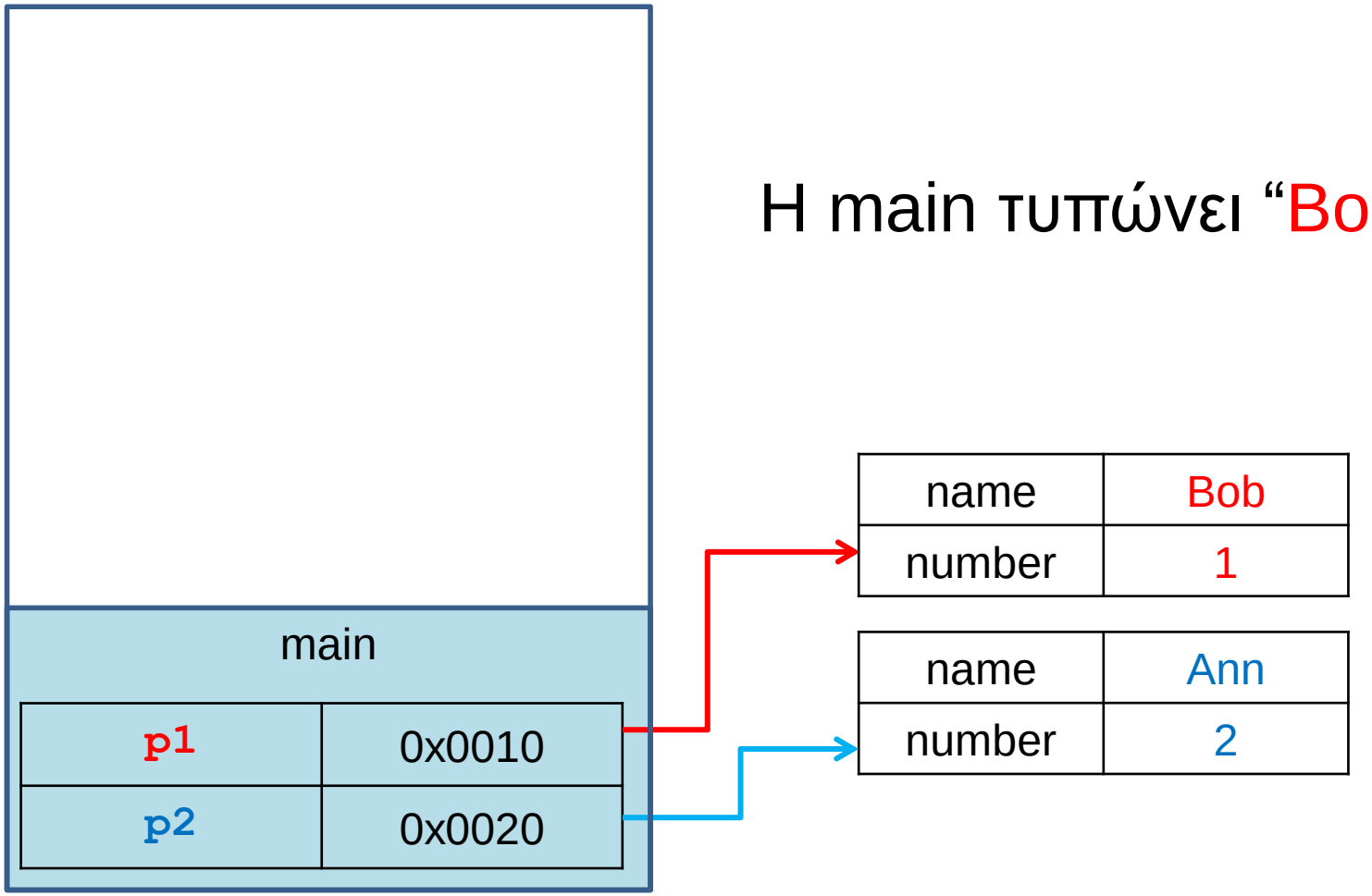
Εξέλιξη του προγράμματος



Εξέλιξη του προγράμματος



Εξέλιξη του προγράμματος



Η main τυπώνει “Bob 1”

Μια ακόμη υλοποίηση της copier

```
public void copier( Person other) {
        other = new Person(this.name, this.number);
    }
```

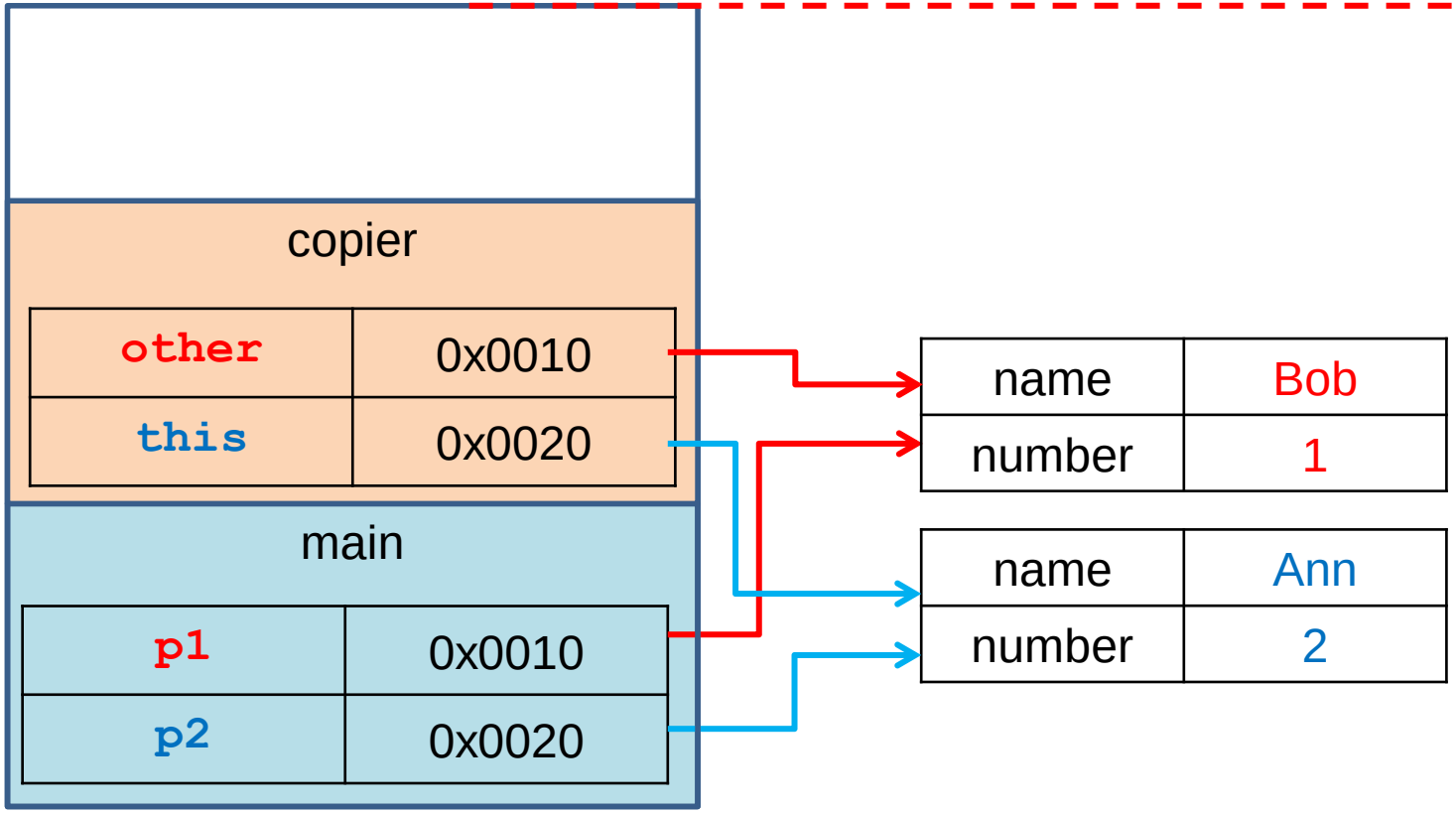
```
public class ClassParameterDemo
{
    public static void main(String[] args)
    {
        Person p1 = new Person("Bob", 1);
        Person p2 = new Person("Ann", 2);
        p2.copier(p1);
        System.out.println(p1);
     }
}
```

Τι θα τυπώσει?

Εξέλιξη του προγράμματος

```
p2.copier(p1);
```

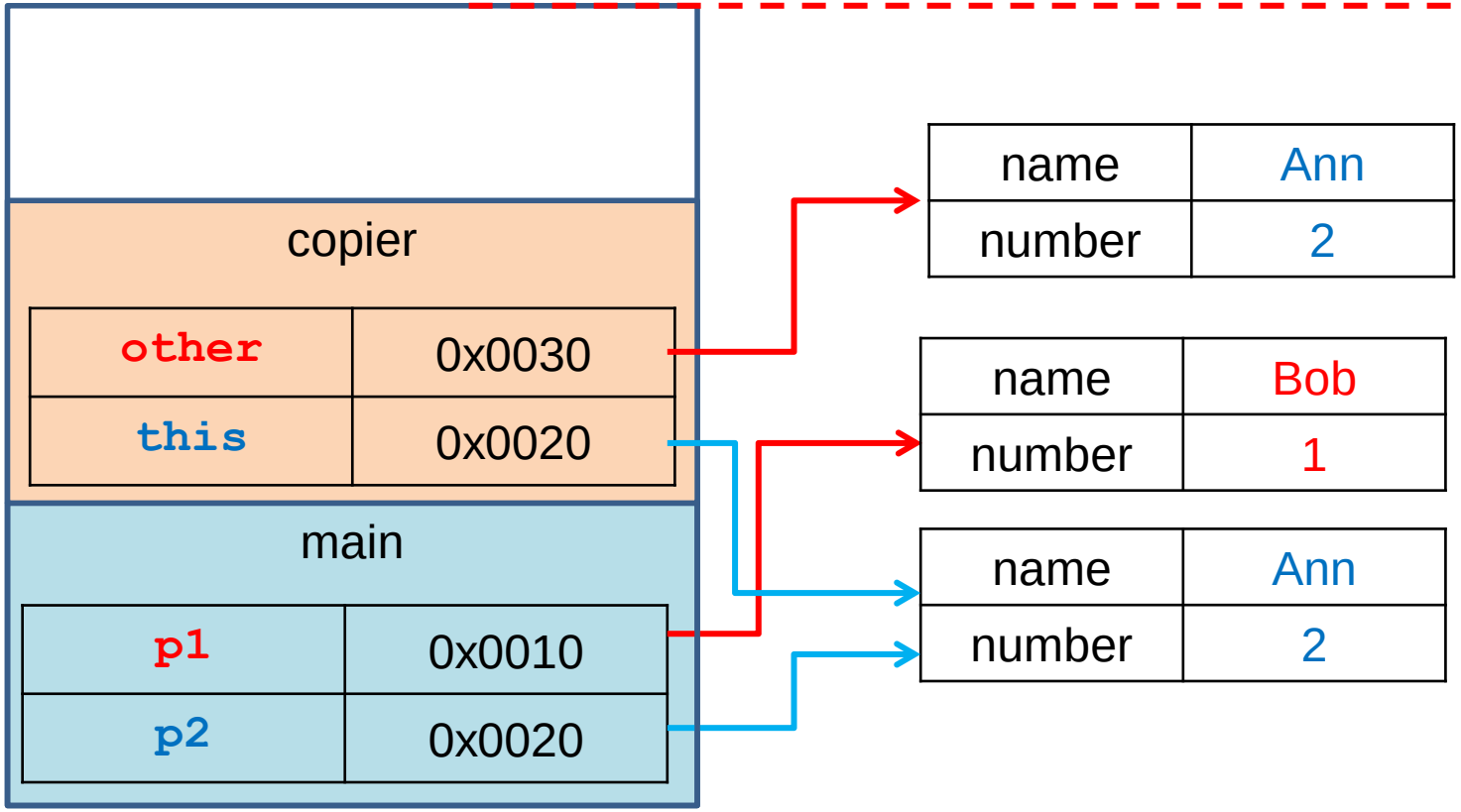
```
public void copier( Person other) {
 other = new Person(this.name, this.number);
}
```



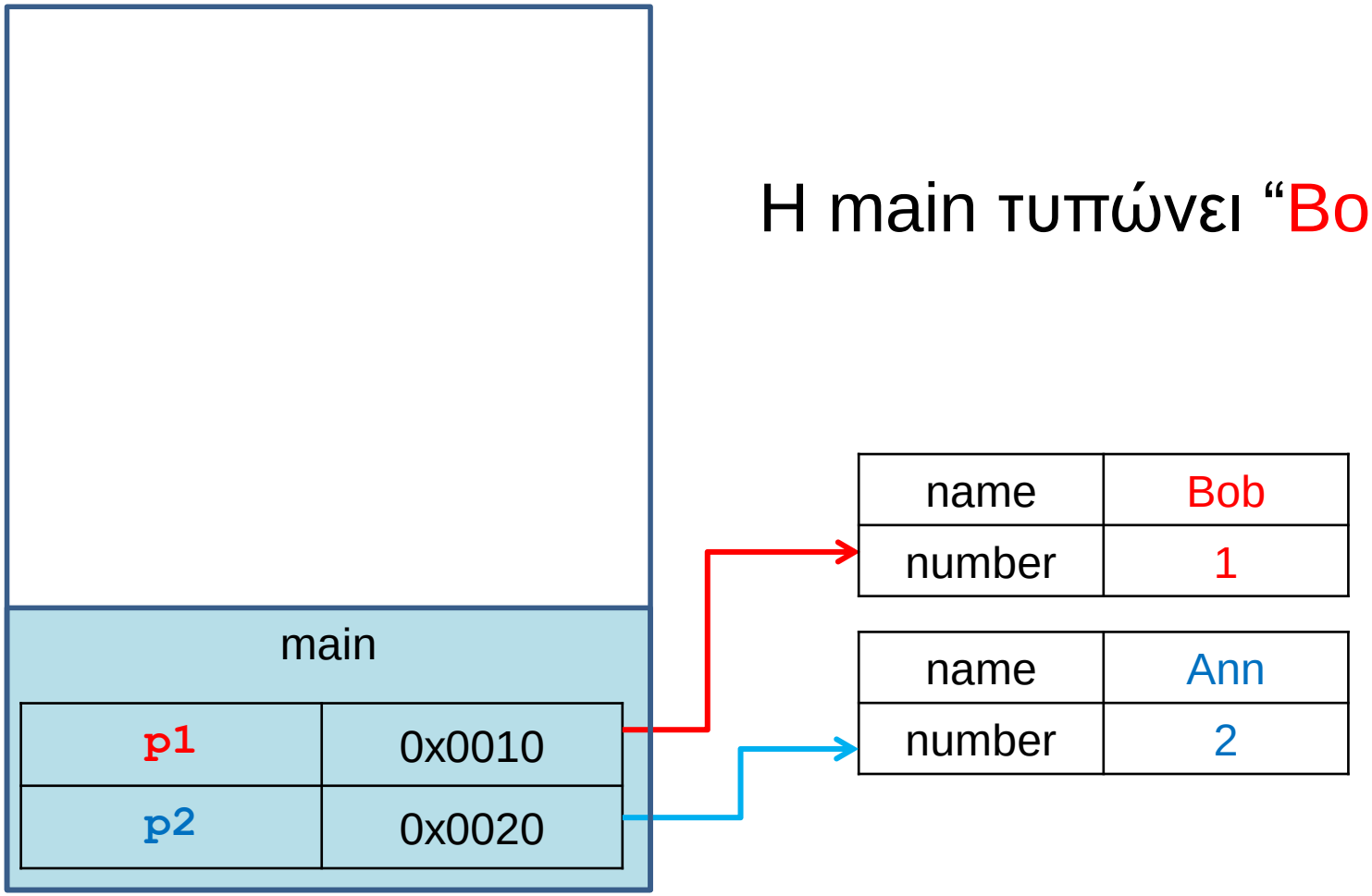
Εξέλιξη του προγράμματος

```
p2.copier(p1);
```

```
public void copier( Person other) {
  other = new Person(this.name, this.number);
}
```



Εξέλιξη του προγράμματος



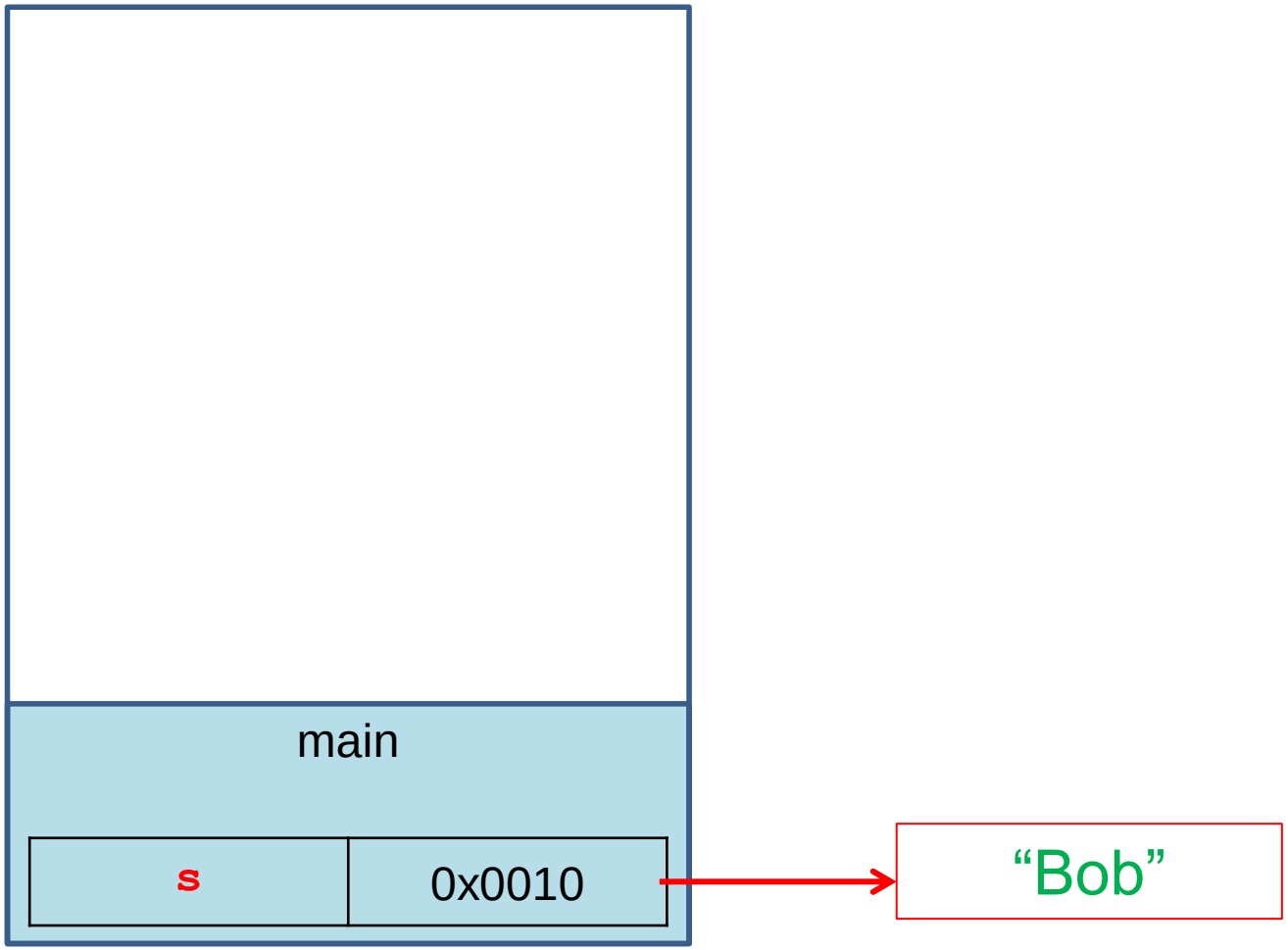
Η main τυπώνει “Bob 1”

Άλλο ένα παράδειγμα

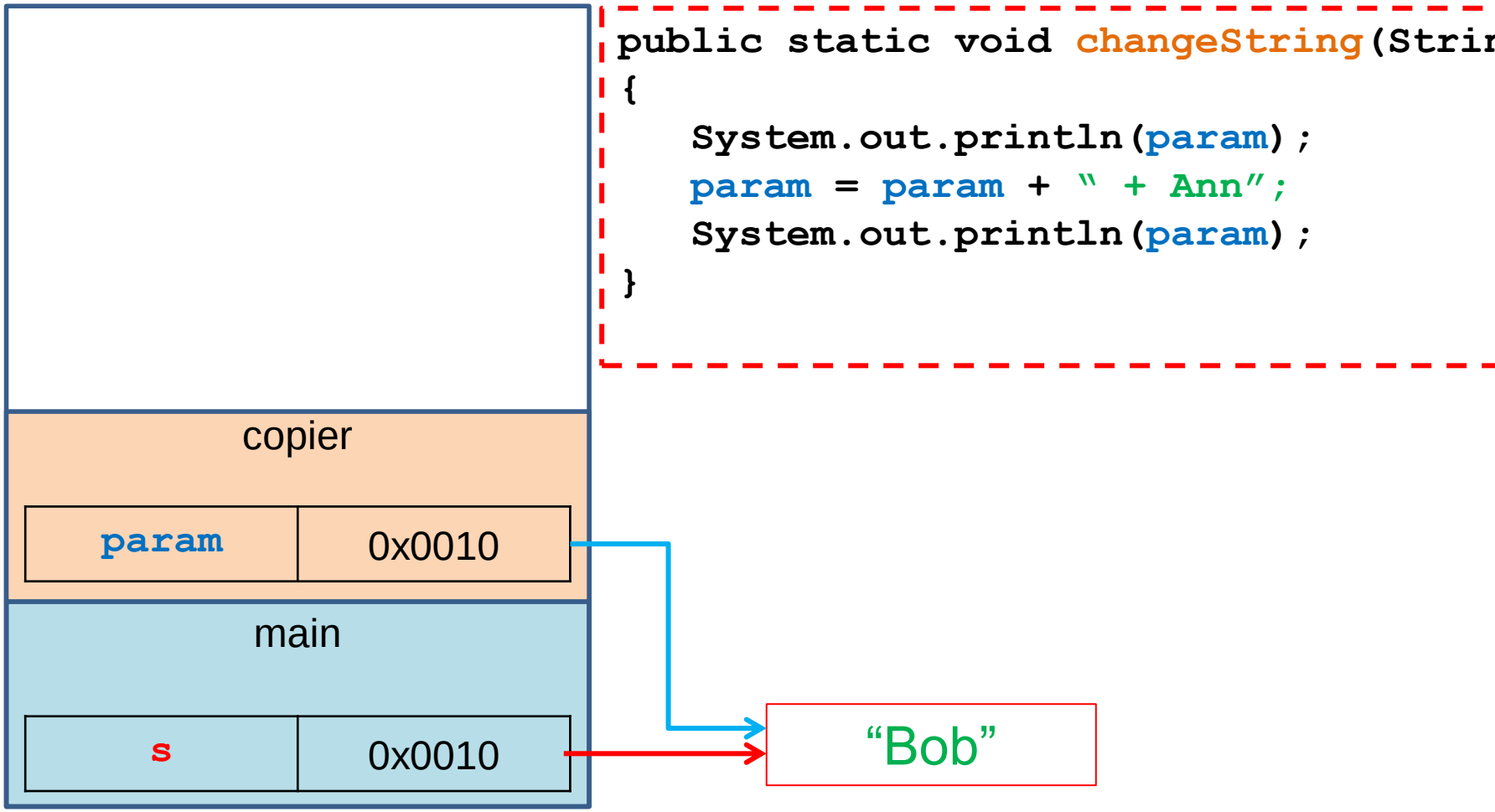
```
public class StringParameterDemo
{
    public static void main(String[] args)
    {
        String s = "Bob";
        changeString(s);
        System.out.println(s);
     }

    public static void changeString(String param)
    {
        System.out.println(param);
        param = param + " + Ann";
        System.out.println(param);
    }
}
```

Τι θα τυπώσει?

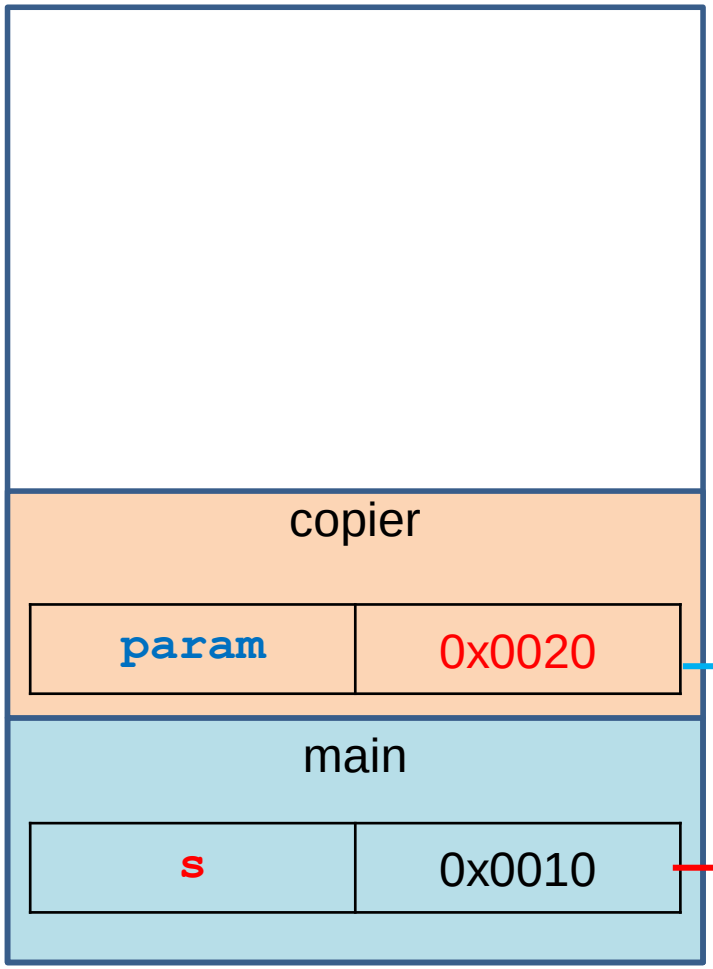
Εξέλιξη του προγράμματος



Εξέλιξη του προγράμματος



Εξέλιξη του προγράμματος




```
public static void changeString(Strir
{
    System.out.println(param);
    param = param + " + Ann";
    System.out.println(param);
}
```

Τα Strings είναι μη μεταλλάξιμα αντικείμενα (immutable objects)

"Bob + Ann"

"Bob"

Η αλλαγή σε ένα String έχει ως αποτέλεσμα τη δημιουργία ενός καινούριου αντικειμένου