ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Αναφορές

Αντικείμενα ως επιστρεφόμενες τιμές

Πίνακες με αντικείμενα

Βαθιά και ρηχά αντίγραφα

Copy Constructor

Η αναφορά this

Σύνθεση και αναφορές

ΑΝΤΙΚΕΙΜΕΝΑ ΩΣ ΕΠΙΣΤΡΕΦΟΜΕΝΕΣ ΤΙΜΕΣ

Επιστροφή αντικειμένων

- Ένα αντικείμενο που δημιουργούμε μέσα σε μία μέθοδο μπορούμε να το διατηρήσουμε και μετά το τέλος της μεθόδου αν κρατήσουμε μια αναφορά σε αυτό.
- Ένας τρόπος να γίνει αυτό είναι αν η μέθοδος επιστρέφει το αντικείμενο (δηλαδή την αναφορά σε αυτό) που δημιουργήσαμε

Η κλάση Date

```
class Date
{
    private int day = 1;
    private int month = 1;
    private int year = 2015;
    private String[] monthStrings =
                    {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
                     "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

    public Date(int day, int month, int year){
       this.day = day;
       this.month = month;
       this.year = year;
    }

    public String toString(){
       return day + " " + monthNames[month-1] + " " + year;
    }
}
```

Θέλω η κλάση να μπορεί να μου επιστρέφει μια νέα ημερομηνία αλλά ένα χρόνο μετά. Πως μπορώ να το κάνω?

Η κλάση Date

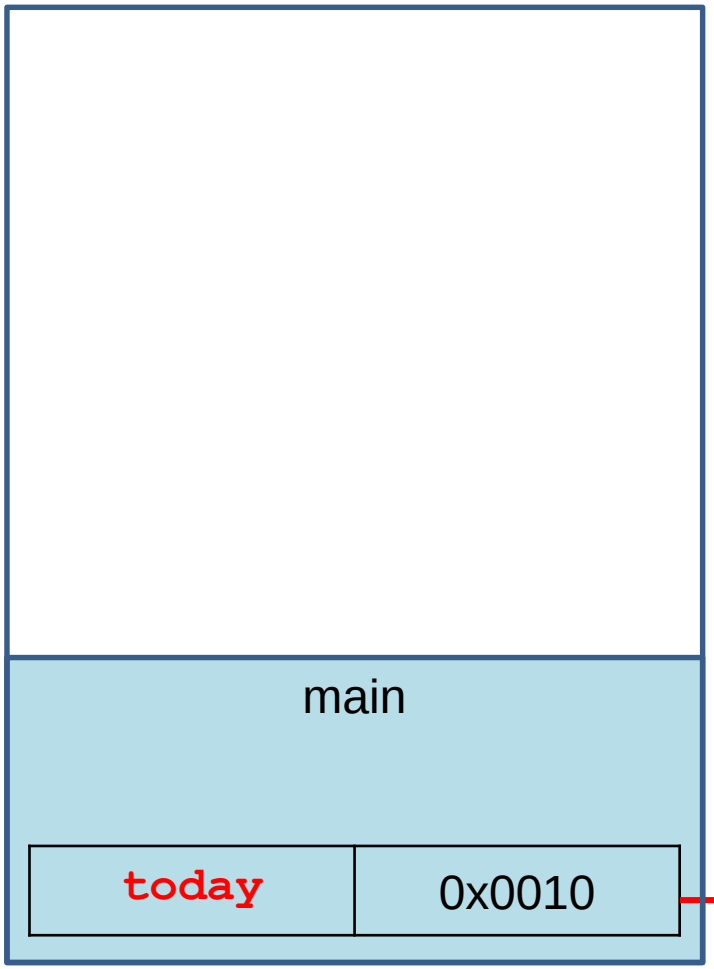
```
class Date
{
    private int day = 1;
    private int month = 1;
    private int year = 2014;
    private String[] monthStrings =
                        {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
                        "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

    public Date(int day, int month, int year){
        this.day = day; this.month = month; this.year = year;
    }

    public String toString(){
        return day + " " + monthNames[month-1] + " " + year;
    }

    public Date nextYear(){
        Date nextYearDate = new Date(day,month,year+1);
        return nextYearDate;
    }
}
```

Η κλάση nextYear() επιστρέφει ένα νέο αντικείμενο Date με την ημερομηνία ένα χρόνο μετά.

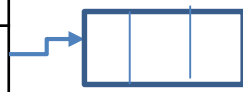
```
class DateExample
{
    public static void main(String args[]){
      Date today = new Date(16,4,2018);
      System.out.println(today);
      Date todayNextYear = today.nextYear();
      System.out.println(todayNextYear);
    }
}
```

Εξέλιξη του προγράμματος

```
class DateExample
{
  public static void main(String args[]){
    Date today = new Date(16,4,2018);
    System.out.println(today);
    Date todayNextYear = today.nextYear();
    System.out.println(todayNextYear);
  }
}
```



day	16
month	4
year	2018
monthStrings	0x0050



Εξέλιξη του προγράμματος

```
public Date nextYear(){
    Date nextYearDate =
         new Date(day,month,year+1);
    return nextYearDate;
}
```

Η πρόσβαση στα πεδία day, month, year είναι μέσω της μεταβλητής this

today.nextYear()

nextYearDate	0x0200
this	0x0010

main

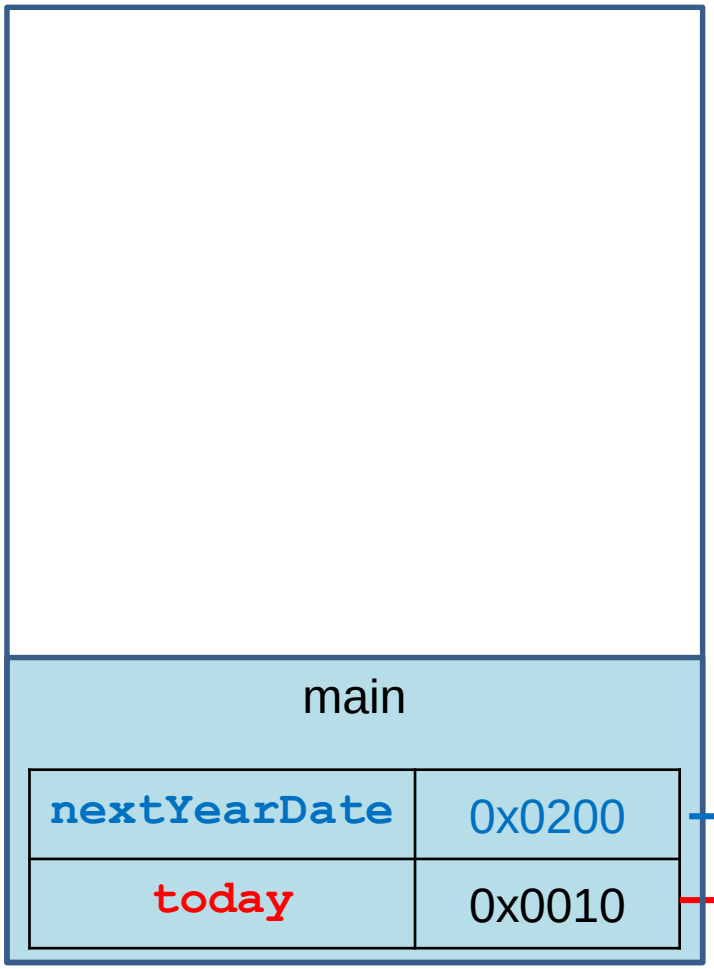
today	0x0010

day	16
month	4
year	2019
monthStrings	0x0250

day	16
month	4
year	2018
monthStrings	0x0050

Εξέλιξη του προγράμματος

```
class DateExample
{
   public static void main(String args[]){
      Date today = new Date(16,4,2018);
      System.out.println(today);
      Date todayNextYear = today.nextYear();
      System.out.println(todayNextYear);
   }
}
```



day	16
month	4
year	2019
monthStrings	0x0250

day	16
month	4
year	2018
monthStrings	0x0050

Η κλάση Date

```
class Date
{
    private int day = 1;
    private int month = 1;
    private int year = 2014;
    private String[] monthStrings =
                        {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
                        "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

    public Date(int day, int month, int year){
        this.day = day; this.month = month; this.year = year;
    }

    public String toString(){
        return day + " " + monthNames[month-1] + " " + year;
    }

    public Date nextYear(){
        return new Date(day,month,year+1);
    }
}
```

Τι γίνεται αν η ημερομηνία είναι 29/2?

Μπορούμε να επιστρέψουμε το αντικείμενο που δημιουργούμε κατευθείαν ως επιστρεφόμενη τιμή (παρομοίως και ως όρισμα σε μέθοδο)

Η κλάση Date

```
class Date
{
    private int day = 1;
    private int month = 1;
    private int year = 2014;
    private String[] monthStrings =
                        {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
                        "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

    public Date(int day, int month, int year){
        this.day = day; this.month = month; this.year = year;
    }

    public String toString(){
        return day + " " + monthNames[month-1] + " " + year;
    }

    public Date nextYear(){
        if (day == 29 && month == 2){
            return null;
        }
        return new Date(day,month,year+1);
    }
```

Τι γίνεται αν η ημερομηνία είναι 29/2?

Η τιμή null: Μία κενή αναφορά. Η τιμή μπορεί να χρησιμοποιηθεί σαν μια default τιμή, ή σαν ένδειξη λάθους (στην περίπτωση αυτή ότι δεν μπορούμε να δημιουργήσουμε το αντικείμενο)

```
class DateExample
{
    public static void main(String args[]){
     Date today = new Date(3,4,2014);
     System.out.println(today);
     Date todayNextYear = today.nextYear();
     if( todayNextYear != null){
         System.out.println(todayNextYear);
     }
    }
}
```

Προσοχή: Η χρήση του null για έλεγχο λάθους σημαίνει ότι όποτε χρησιμοποιούμε την μέθοδο θα πρέπει να προσέχουμε αν η επιστρεφόμενη τιμή είναι null. Δεν είναι καλή λύση, και αργότερα θα μάθουμε για εξαιρέσεις για να χειριζόμαστε τέτοια προβλήματα.

ΠΙΝΑΚΕΣ ΑΠΟ ΑΝΤΙΚΕΙΜΕΝΑ

Πίνακες από αντικείμενα

- Όπως ορίζουμε πίνακες από πρωταρχικούς τύπους μπορούμε να ορίσουμε και πίνακες από αντικείμενα
  - `Person[] array = new Person[3];`
  - Ορίζει ένα πίνακα με τρία αντικείμενα τύπου Person
  - Ουσιαστικά ένα πίνακα με αναφορές.

- Όταν ορίζουμε ένα πίνακα από αντικείμενα πρέπει να είμαστε προσεκτικοί να δεσμεύουμε σωστά τη μνήμη.

Παράδειγμα

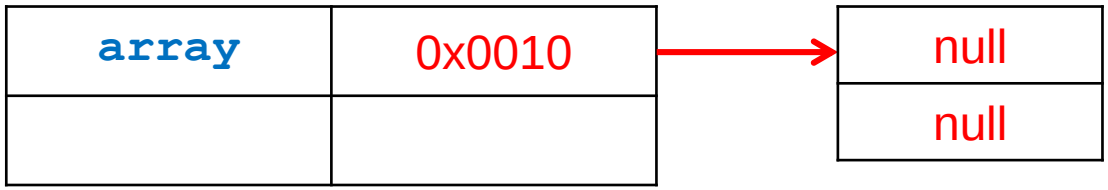
```
Person[] array;
```

array	null

- Η εντολή αυτή θα δημιουργήσει μια μεταβλητή με το όνομα array η οποία κάποια στιγμή θα δείχνει σε ένα πίνακα με Person. Για την ώρα είναι null.

Παράδειγμα

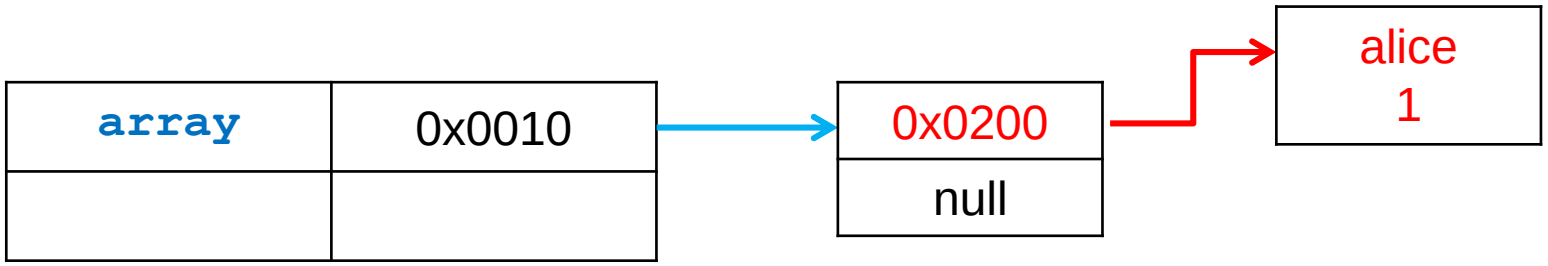
```
Person[] array;
array = new Person[2];
```



- Η εντολή new θα δεσμεύσει δύο θέσεις μνήμης στο heap για να κρατήσουν δύο αναφορές τύπου Person. Εφόσον δεν έχουμε δημιουργήσει τις μεταβλητές ακόμη, αυτές θα είναι null.

Παράδειγμα

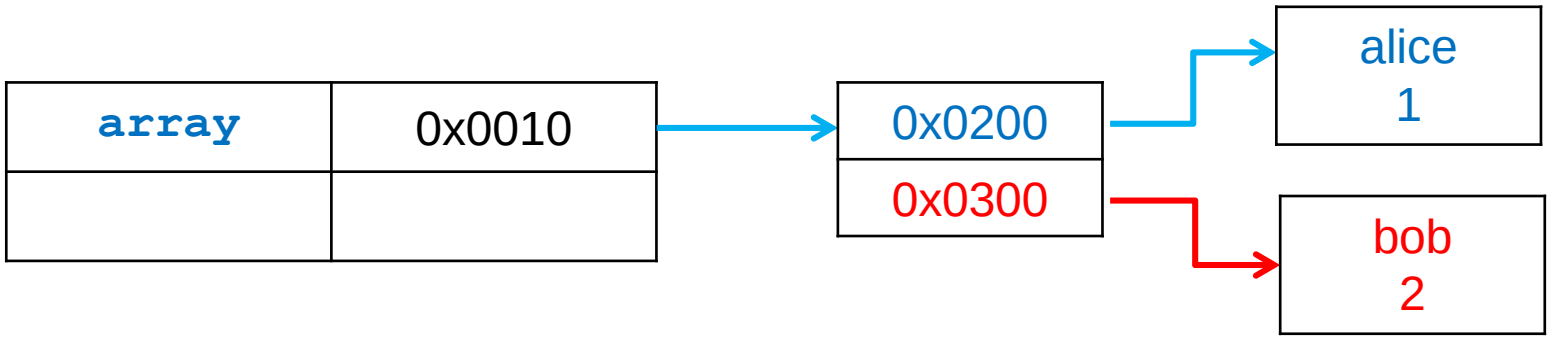
```
Person[] array;
array = new Person[2];
array[0] = new Person("alice", 1);
```



- Η νέα εντολή new θα δεσμεύσει χώρο για ένα Person. Δημιουργείται το αντικείμενο και η αναφορά αποθηκεύεται στην πρώτη θέση του πίνακα array.

Παράδειγμα

```
Person[] array;
array = new Person[2];
array[0] = new Person("alice", 1);
array[1] = new Person("bob", 1);
```



- Η νέα εντολή new θα δεσμεύσει χώρο για άλλο ένα Person. Δημιουργείται το αντικείμενο και η αναφορά αποθηκεύεται στην δεύτερη θέση του πίνακα array.

Πίνακες από πίνακες

- Οι δισδιάστατοι πίνακες είναι ουσιαστικά πίνακες από αντικείμενα, όπου τα αντικείμενα είναι πάλι πίνακες
- Π.χ., έτσι δεσμεύουμε πίνακα ακεραίων 5×5

```
int[][] array;
array = new int[5][];
for (int i=0; i<5; i++){
     array[i] = new int[5];
}
```

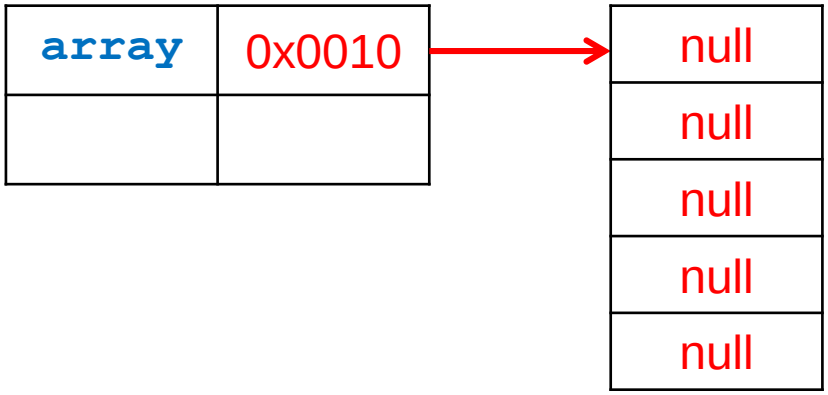
Παράδειγμα

```
int[][] array;
```

array	null

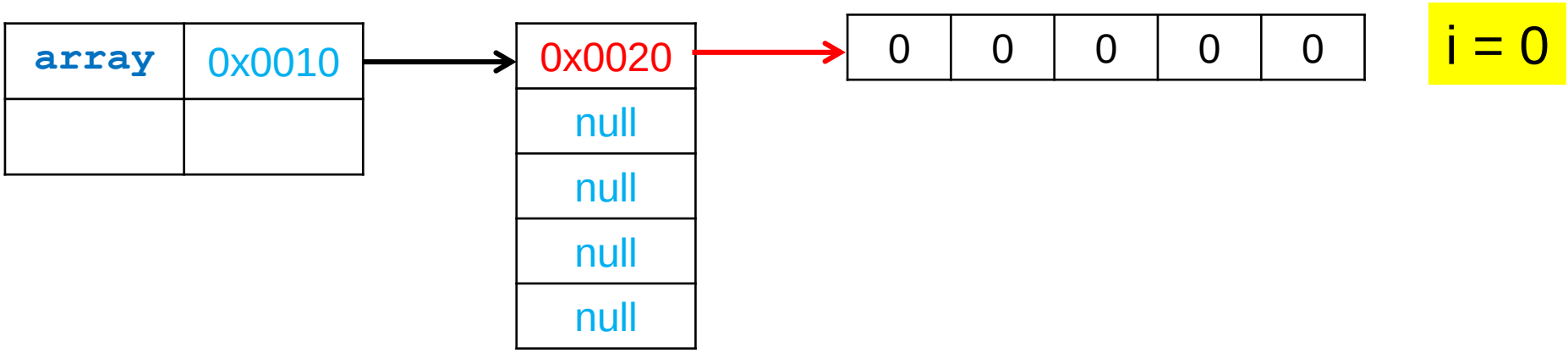
Παράδειγμα

```
int[][] array;
array = new int[5][];
```



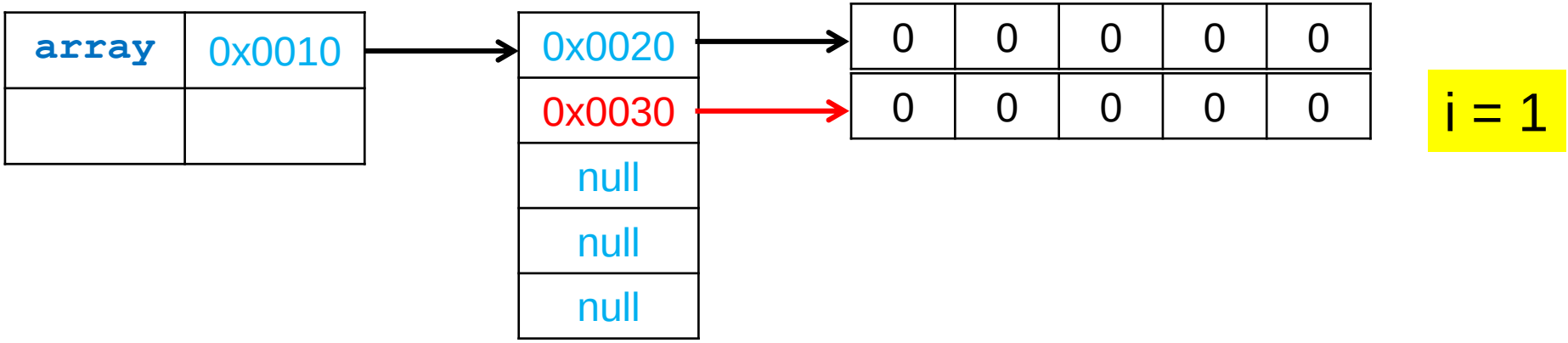
Παράδειγμα

```
int[][] array;
array = new int[5][];
for (int i=0; i<5; i++){
      array[i] = new int[5];
}
```



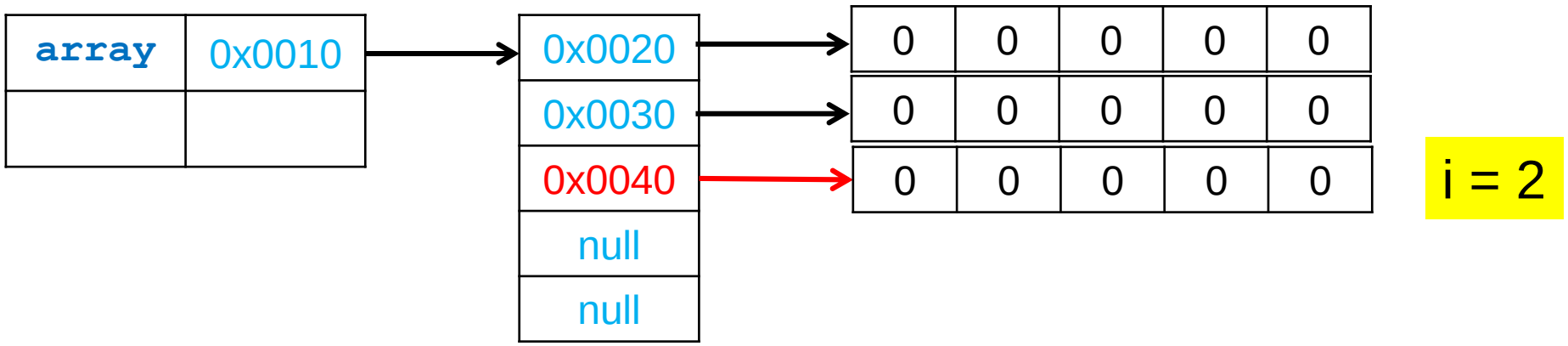
Παράδειγμα

```
int[][] array;
array = new int[5][];
for (int i=0; i<5; i++){
      array[i] = new int[5];
}
```



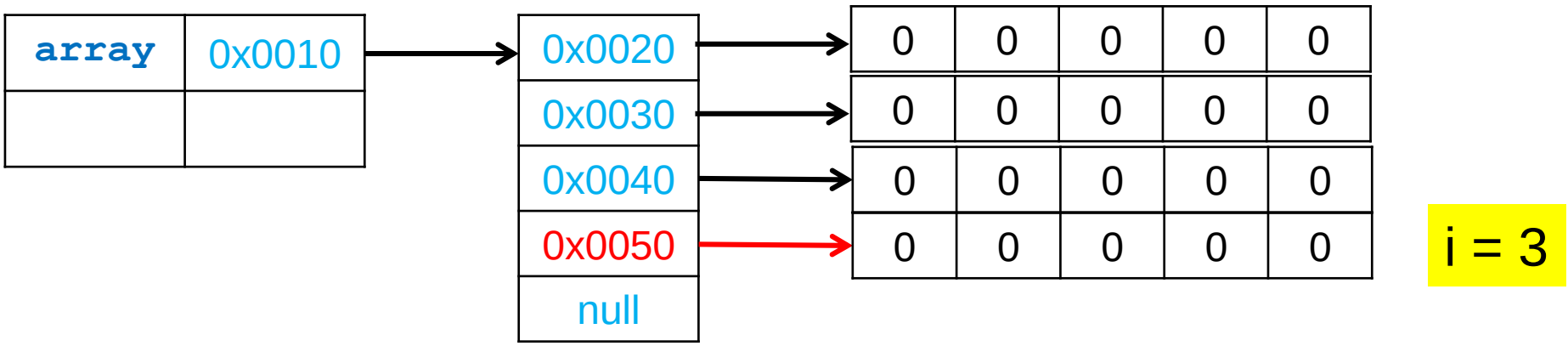
Παράδειγμα

```
int[][] array;
array = new int[5][];
for (int i=0; i<5; i++){
      array[i] = new int[5];
}
```



Παράδειγμα

```
int[][] array;
array = new int[5][];
for (int i=0; i<5; i++){
    array[i] = new int[5];
}
```



Παράδειγμα

```
int[][] array;
array = new int[5][];
for (int i=0; i<5; i++){
      array[i] = new int[5];
}
```

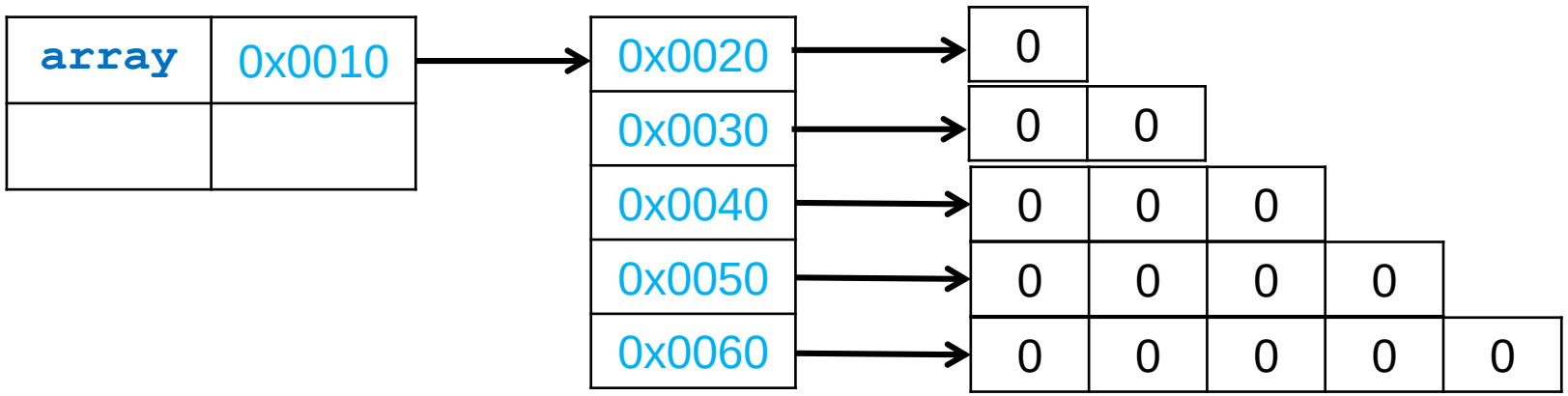
Πίνακες από πίνακες

- Μπορεί ο δισδιάστατος μας πίνακας να είναι ασύμμετρος.
- Π.χ., έτσι ορίζουμε ένα διαγώνιο πίνακα.

```
int[][] array;
array = new int[5][];
for (int i=0; i<5; i++){
     array[i] = new int[i+1];
}
```

Παράδειγμα

```
int[][] array;
array = new int[5][];
for (int i=0; i<5; i++){
     array[i] = new int[i+1];
}
```



ΒΑΘΕΙΑ ΚΑΙ ΡΗΧΑ ΑΝΤΙΓΡΑΦΑ COPY CONSTRUCTOR

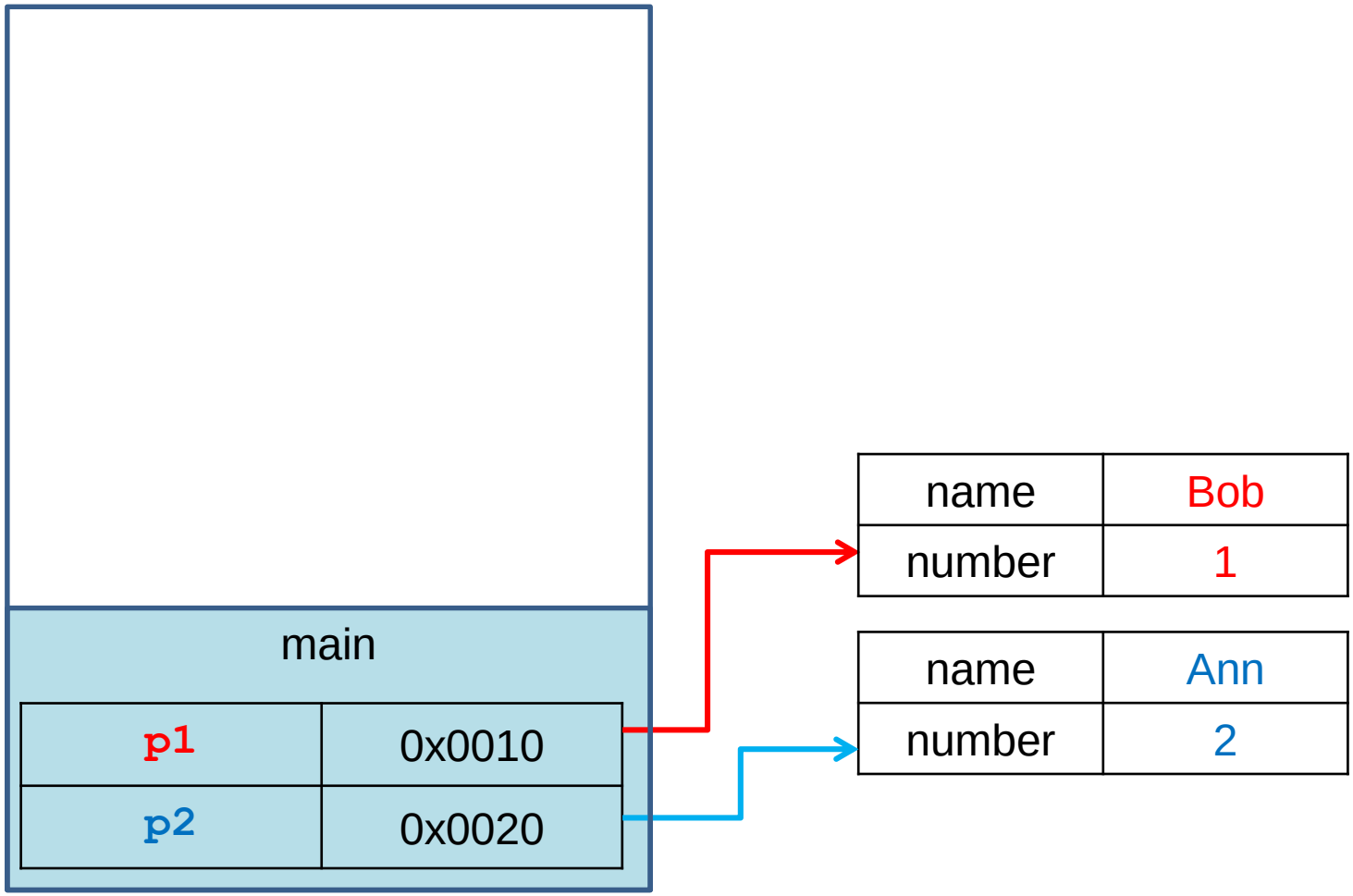
```
public class Person
{
    private String name;
    private int number;

    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 Person copier() {
        Person newPerson = new Person(this.name,this.number);
        return newPerson;
    }

}
```

Παράδειγμα

```
public class ClassParameterDemo2
{
    public static void main(String[] args)
    {
        Person p1 = new Person("Bob", 1);
        Person p2 = new Person("Ann", 2);
        p1 = p2.copier();
        System.out.println(p1);
     }
}
```

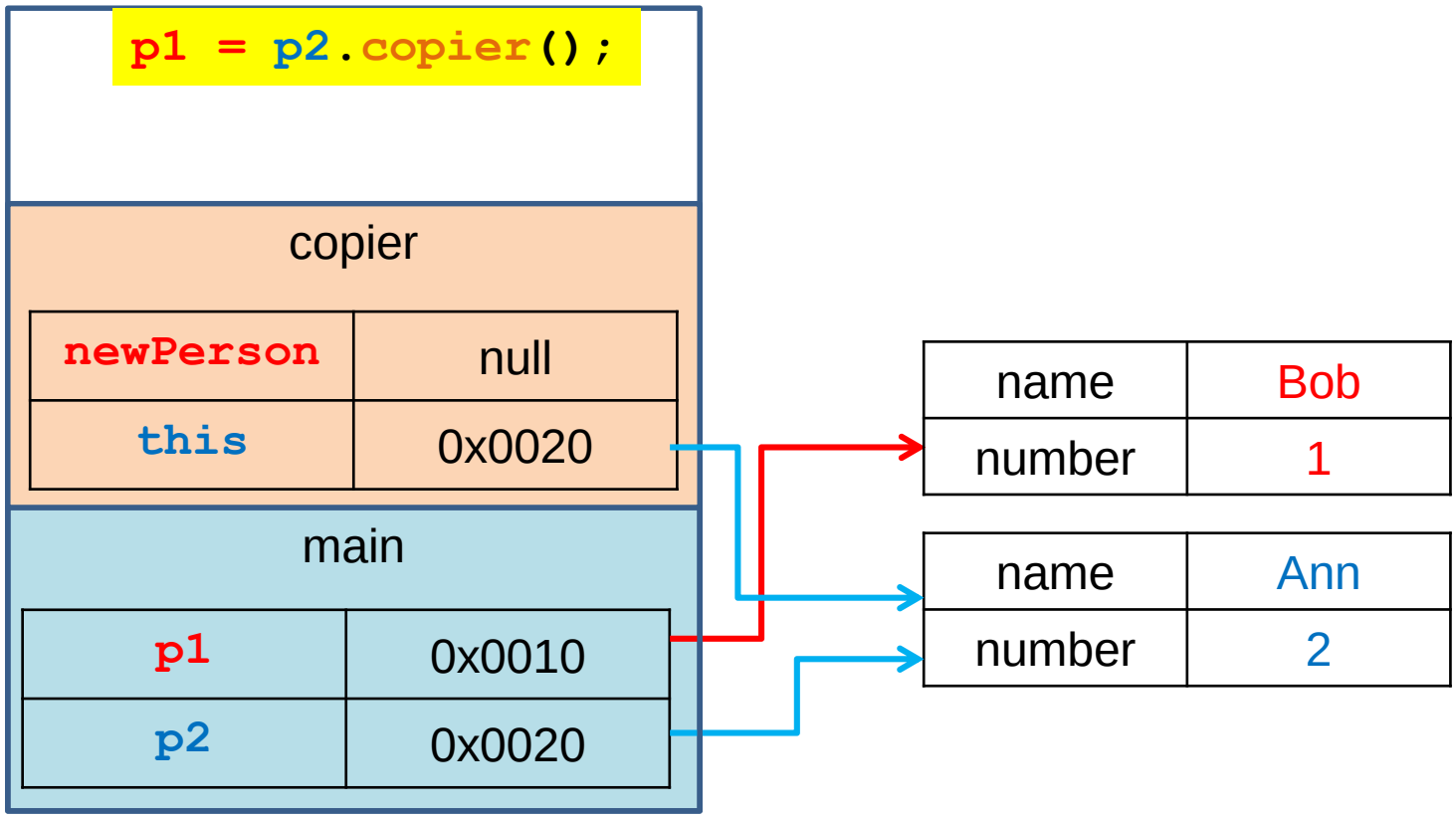
Τι θα τυπώσει?

Εξέλιξη του προγράμματος



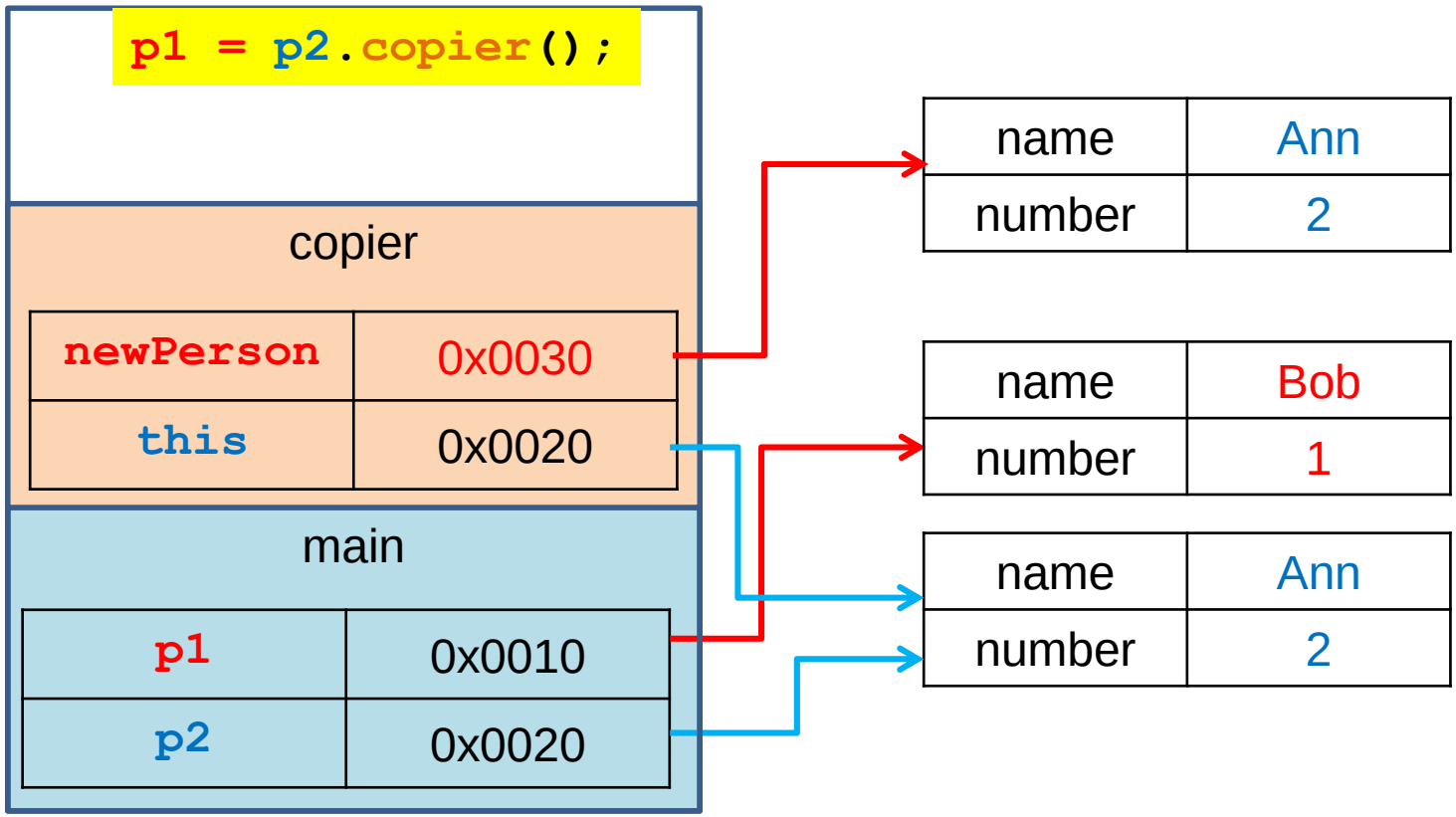
Εξέλιξη του προγράμματος

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



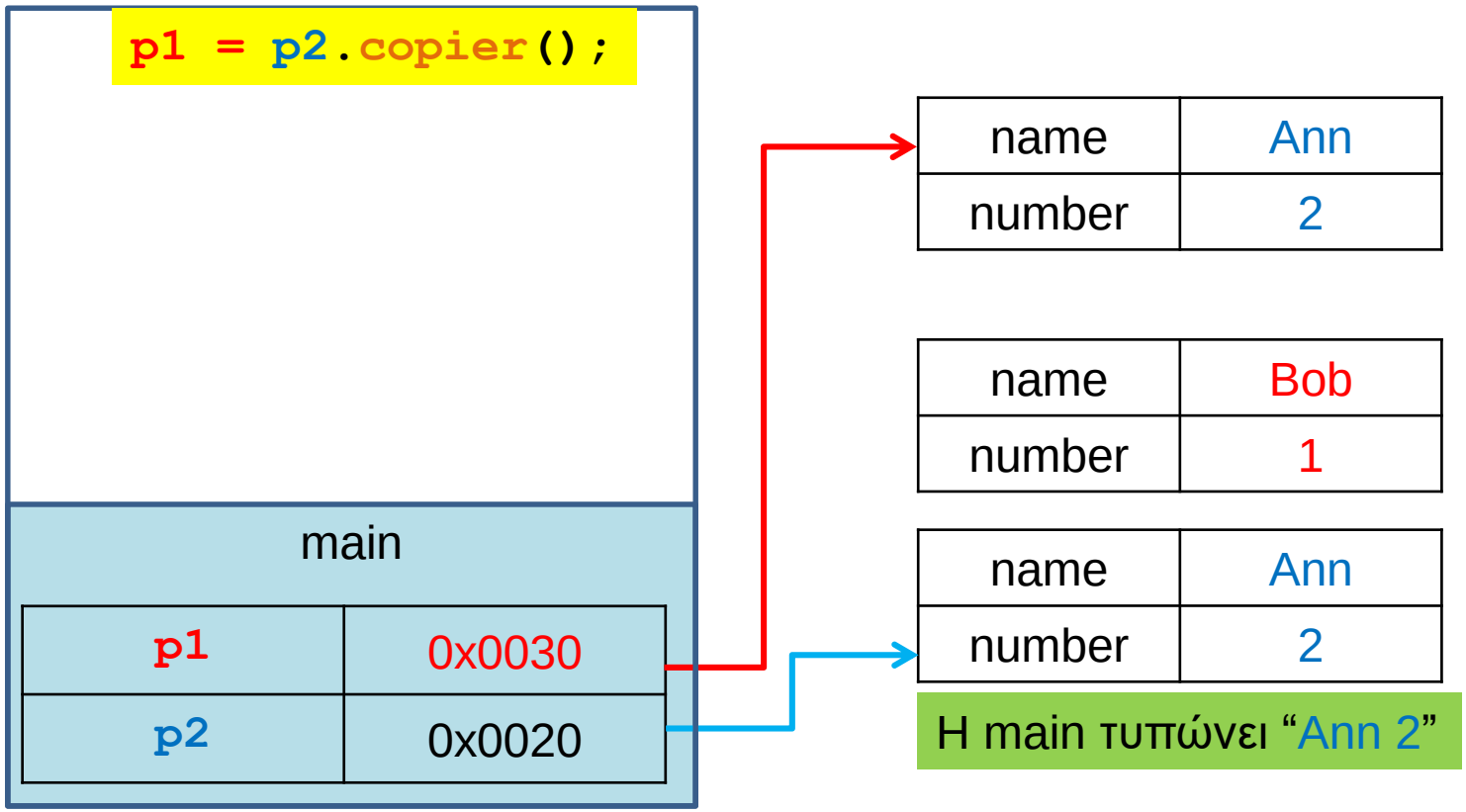
Εξέλιξη του προγράμματος

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



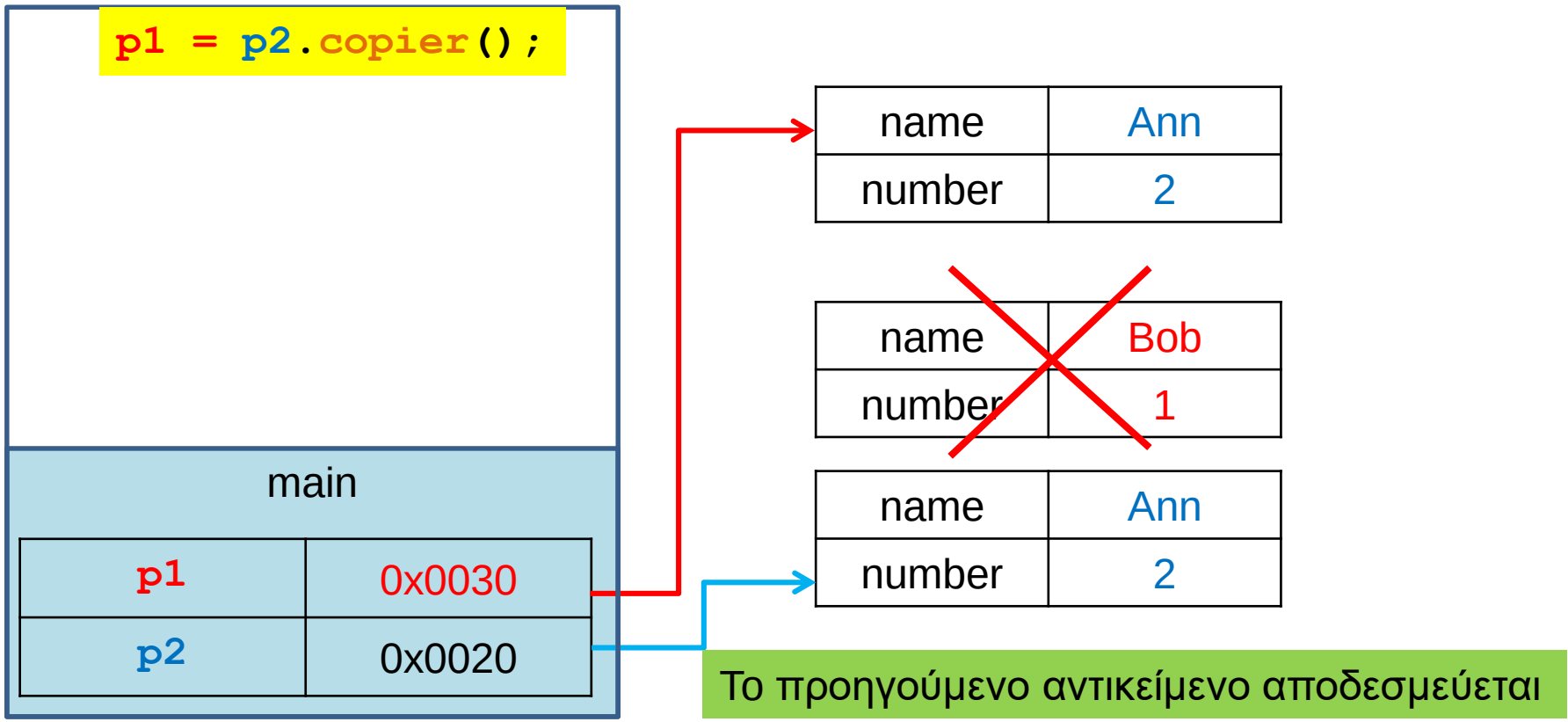
Εξέλιξη του προγράμματος

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



Εξέλιξη του προγράμματος

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



Δημιουργία αντιγράφων

- Η μέθοδος copier όπως την ορίσαμε πριν δημιουργεί ένα καινούριο αντικείμενο που είναι αντίγραφο αυτού που έκανε την κλήση.
- Στην περίπτωση μας το αντικείμενο έχει μόνο πεδία που είναι πρωταρχικού τύπου ή μη μεταλλάξιμα αντικείμενα. Γενικά ένα αντικείμενο μπορεί να έχει ως πεδία άλλα αντικείμενα (δηλαδή αναφορές).
- Στην περίπτωση αυτή η δημιουργία αντιγράφου θα πρέπει να γίνεται με πολύ προσοχή!

```
class Car
{
    private int dim;
    private int[] position;

    public Car(int d){
       dim = d; position = new int[d];
    }

    public void move(){
       for (int i=0; i < dim; i++){position[i] ++;}
    }

    public Car copy(){
       Car newCar = new Car(this.dim);
       newCar.position = this.position;
       return newCar;
    }

    public String toString(){
       String output = "";
       for (int i=0; i < dim; i++){output = output + position[i] + " ";}
       return output;
    }

     public static void main(String args[]){
          Car car1 = new Car(2);
          car1.move();
          Car car2 = car1.copy();
          car2.move();
          System.out.println(car1);
    }
}
```

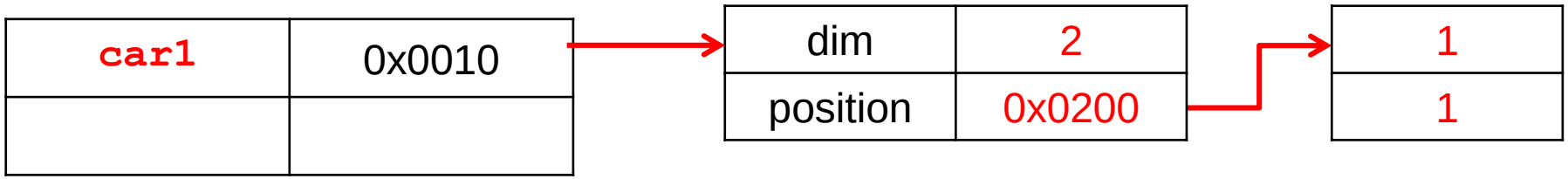
Το Car κινείται σε 1 ή 2 διαστάσεις
Χρειαζόμαστε ένα πίνακα για την θέση του

Η copy δημιουργεί και επιστρέφει ένα νέο Car

Τι θα τυπώσει η main?

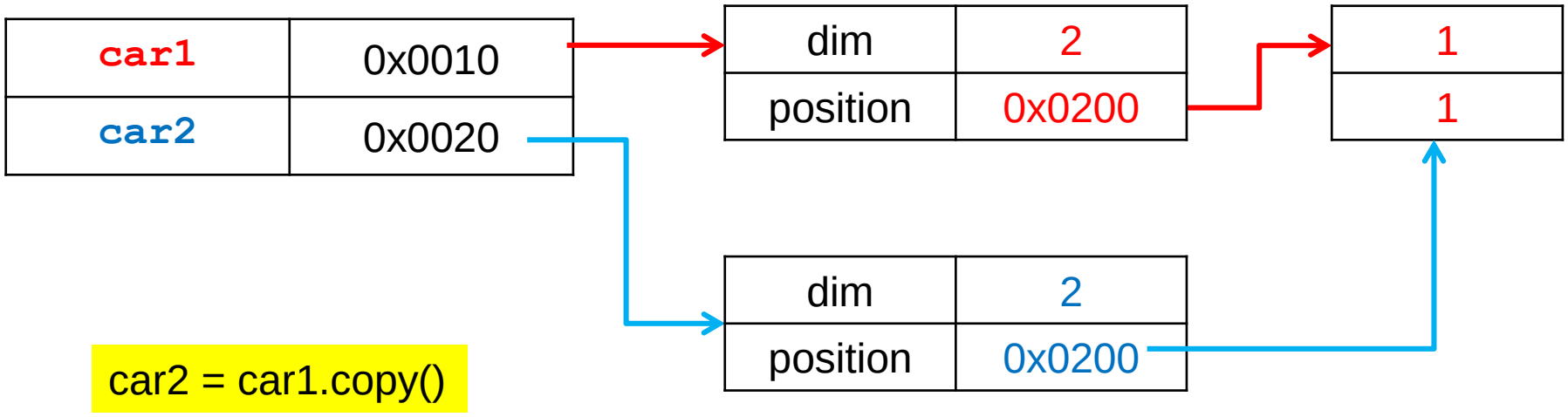
Ρηχά Αντίγραφα

- Η copy όπως την έχουμε ορίσει δημιουργεί ένα ρηχό αντίγραφο του αντικειμένου
  - Αντιγράφει τις αναφορές στα αντικείμενα και όχι τα περιεχόμενα των αντικειμένων



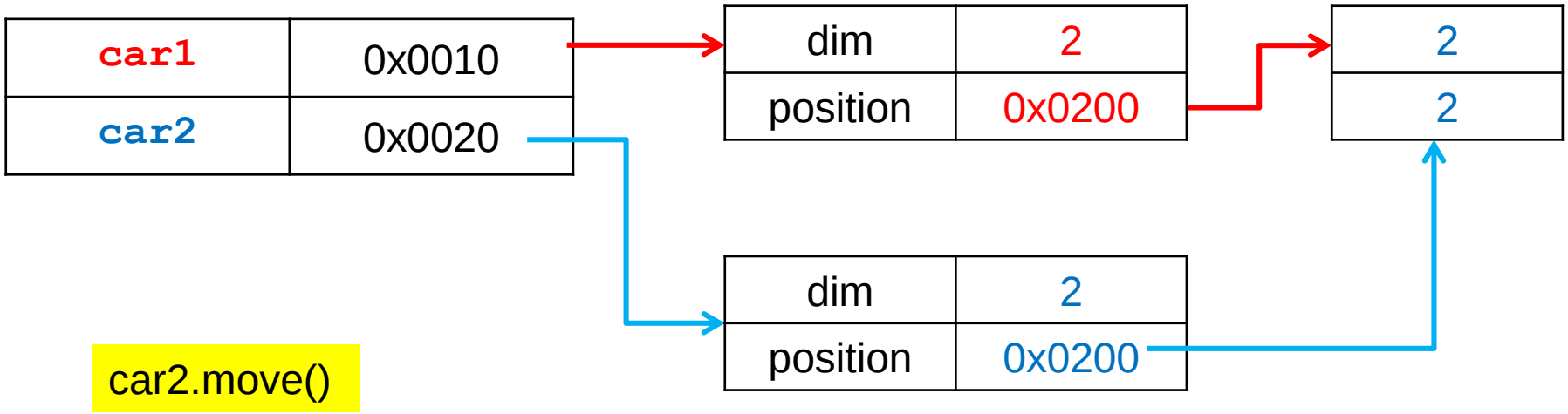
Ρηχά Αντίγραφα

- Η copy όπως την έχουμε ορίσει δημιουργεί ένα ρηχό αντίγραφο του αντικειμένου
  - Αντιγράφει τις αναφορές στα αντικείμενα και όχι τα περιεχόμενα των αντικειμένων



Ρηχά Αντίγραφα

- Η copy όπως την έχουμε ορίσει δημιουργεί ένα ρηχό αντίγραφο του αντικειμένου
  - Αντιγράφει τις αναφορές στα αντικείμενα και όχι τα περιεχόμενα των αντικειμένων

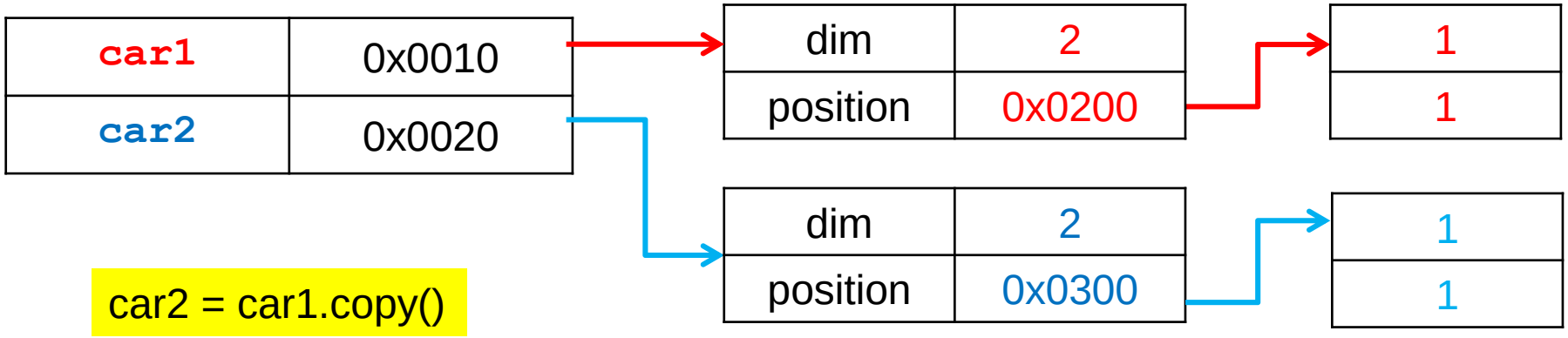


Μετακινείται και το car1 αλλά αυτό δεν είναι επιθυμητό.

Βαθύ αντίγραφο

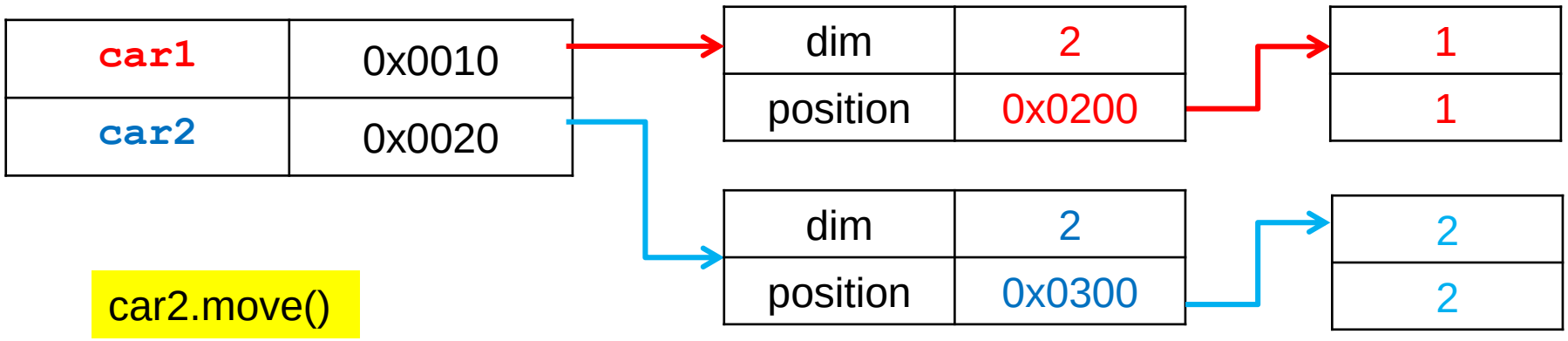
- Τις περισσότερες φορές θέλουμε να κάνουμε ένα βαθύ αντίγραφο του αντικειμένου, όπου για κάθε αντικείμενο μέσα στο αντίγραφο δεσμεύουμε νέα μνήμη

```
public Car copy(){
    Car newCar = new Car(this.dim);
    for (int i=0; i<dim; i++){
        newCar.position[i] = this.position[i];
    }
    return newCar;
}
```



Βαθύ αντίγραφο

- Το βαθύ αντίγραφο του car1 είναι πλέον ένα ανεξάρτητο αντικείμενο.



Η μετακίνηση του car2 δεν επηρεάζει το car1

Copy Constructor

- Ένας Constructor που παίρνει σαν όρισμα ένα αντικείμενο του ίδιου τύπου και δημιουργεί ένα αντίγραφο
  - `public Car(Car other)`

- Ο copy constructor έχει δύο λειτουργίες:
  - Δεσμεύει τη μνήμη για το αντικείμενο
  - Αντιγράφει τις τιμές του αντικειμένου-ορίσματος.
- Πάντα πρέπει να δημιουργούμε ένα βαθύ αντίγραφο του αντικειμένου

Copy Constructor για την Car

```
public Car(Car other)
{
    this.dim = other.dim;
    position = new int[this.dim];
    for (int i = 0; i < this.dim; i ++){
        this.position[i] = other.position[i];
    }
}
```

Δημιουργεί βαθύ αντίγραφο:
Δεσμεύουμε καινούριο πίνακα και αντιγράφουμε μία-μία τις τιμές

Κλήση:

```
Car car1 = new Car(2);
Car car2 = new Car(car1);
```

Φωλιασμένος Copy Constructor

- Αν μια κλάση έχει πεδία αντικείμενα από μία άλλη κλάση, τότε όταν καλούμε τον copy constructor θα πρέπει να έχουμε ορίσει copy constructor και για τις κλάσεις των αντικειμένων-πεδίων.

Παράδειγμα

```
class CarDriver
{
    private int position;
    private Person driver;

    public CarDriver(CarDriver other){
        this.position = other.position;
        driver = new Person(other.driver);
    }
}
```

Καλεί την copy constructor της Person

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

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

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

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

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

    public boolean equals(Person other){
        return (this.name.equals(other.name) && this.number == other.number;
    }
}
```

Η ΑΝΑΦΟΡΑ THIS

Η μεταβλητή this

- Η μεταβλητή (παράμετρος) this
  - Μια κρυφή παράμετρος η οποία περνάει σε κάθε μέθοδο και κρατάει μια αναφορά στο αντικείμενο κλήσης (το αντικείμενο που καλεί την μέθοδο).
- Την χρησιμοποιήσαμε για να διαφοροποιήσουμε τα πεδία του αντικειμένου από παραμέτρους με το ίδιο όνομα

```
class Person
{
        private String name;
        private int age;

        public Person(String name, int age){
                this.name = name;
                this.age = age;
        }
}
```

Η μεταβλητή this

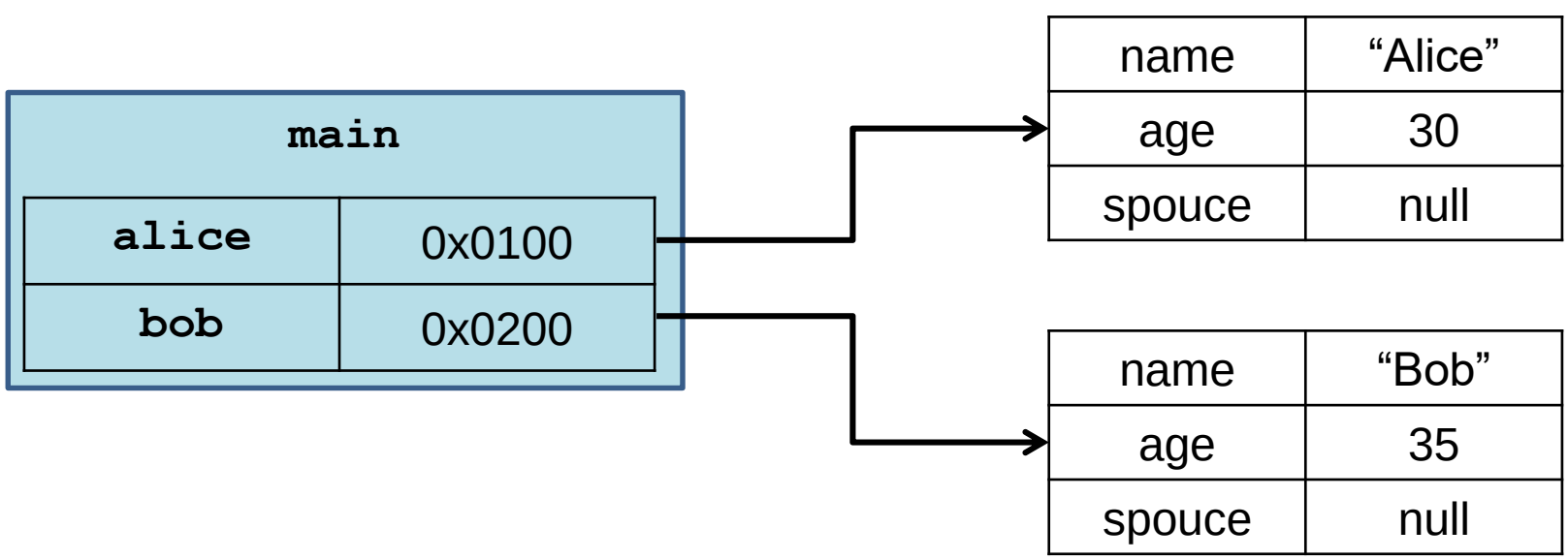
- Εκτός από αυτή την χρήση, μπορούμε να χρησιμοποιήσουμε την μεταβλητή this σαν οποιαδήποτε άλλη μεταβλητή
  - Μπορούμε να την επιστρέψουμε σε κάποια μέθοδο.
  - Μπορούμε να την αναθέσουμε σε κάποια μεταβλητή
  - Μπορούμε να την περάσουμε σαν παράμετρο σε κάποια μέθοδο
- Αυτό είναι χρήσιμο όταν χρειαζόμαστε μια αναφορά στο αντικείμενο κλήσης.

```
class Person
{
    private String name;
    private int age;
    private Person spouce;

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    public Person getOlderPerson(Person other){
        if (this.age > other.age){
            return this;
        }
        return other;
    }

    public void marry(Person other){
        this.spouce = other;
        other.spouce = this;
    }
}
```

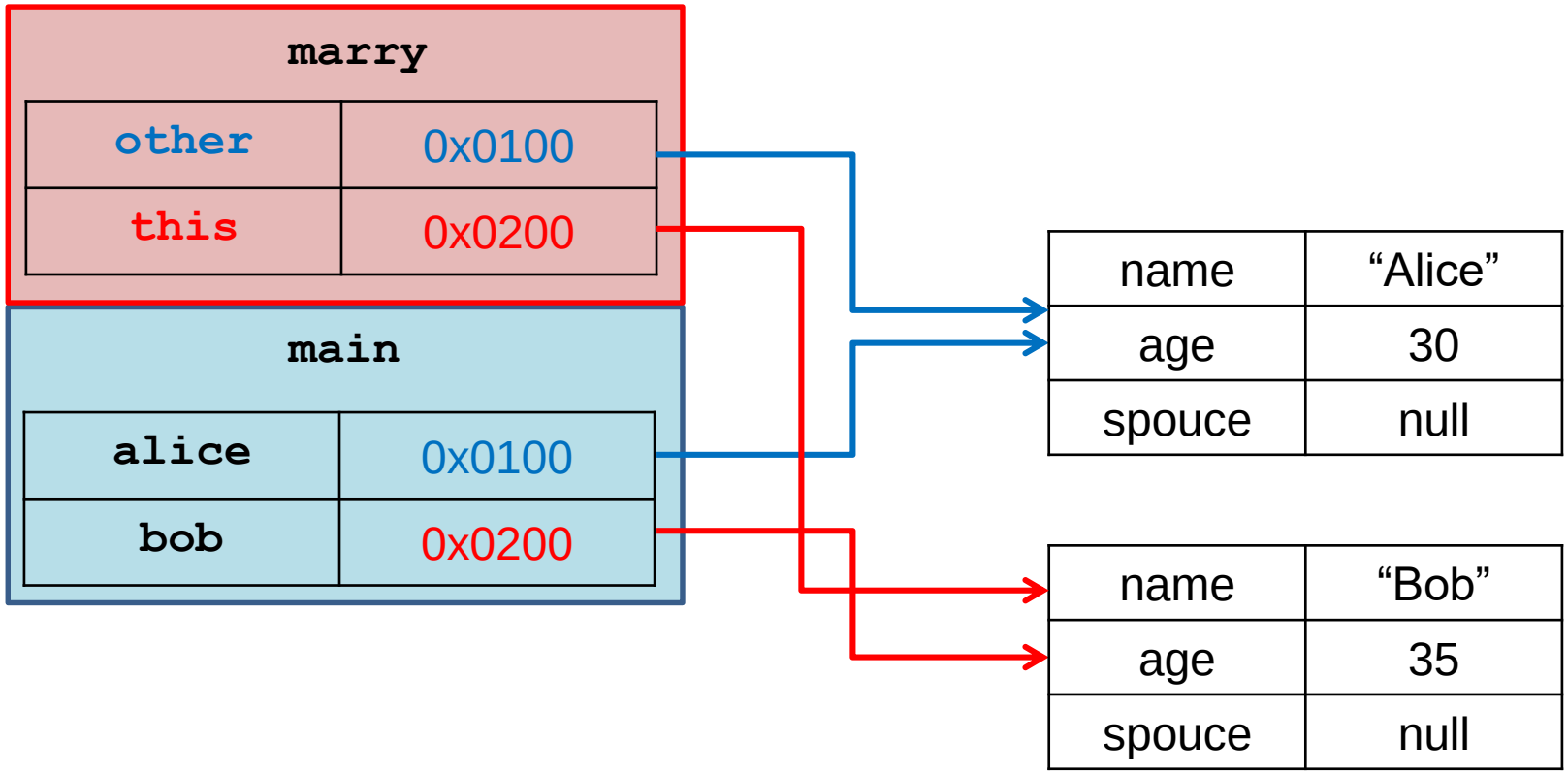
Η μέθοδος μας επιστρέφει μια αναφορά σε αντικείμενο Person
Αν η ηλικία του ατόμου που καλεί την μέθοδο είναι μεγαλύτερη αυτού που περνάμε σαν όρισμα επιστρέφουμε την αναφορά this
Αλλιώς επιστρέφουμε την αναφορά other.

Θέτουμε τον/την σύζυγο του other να δείχνει στο αντικείμενο this

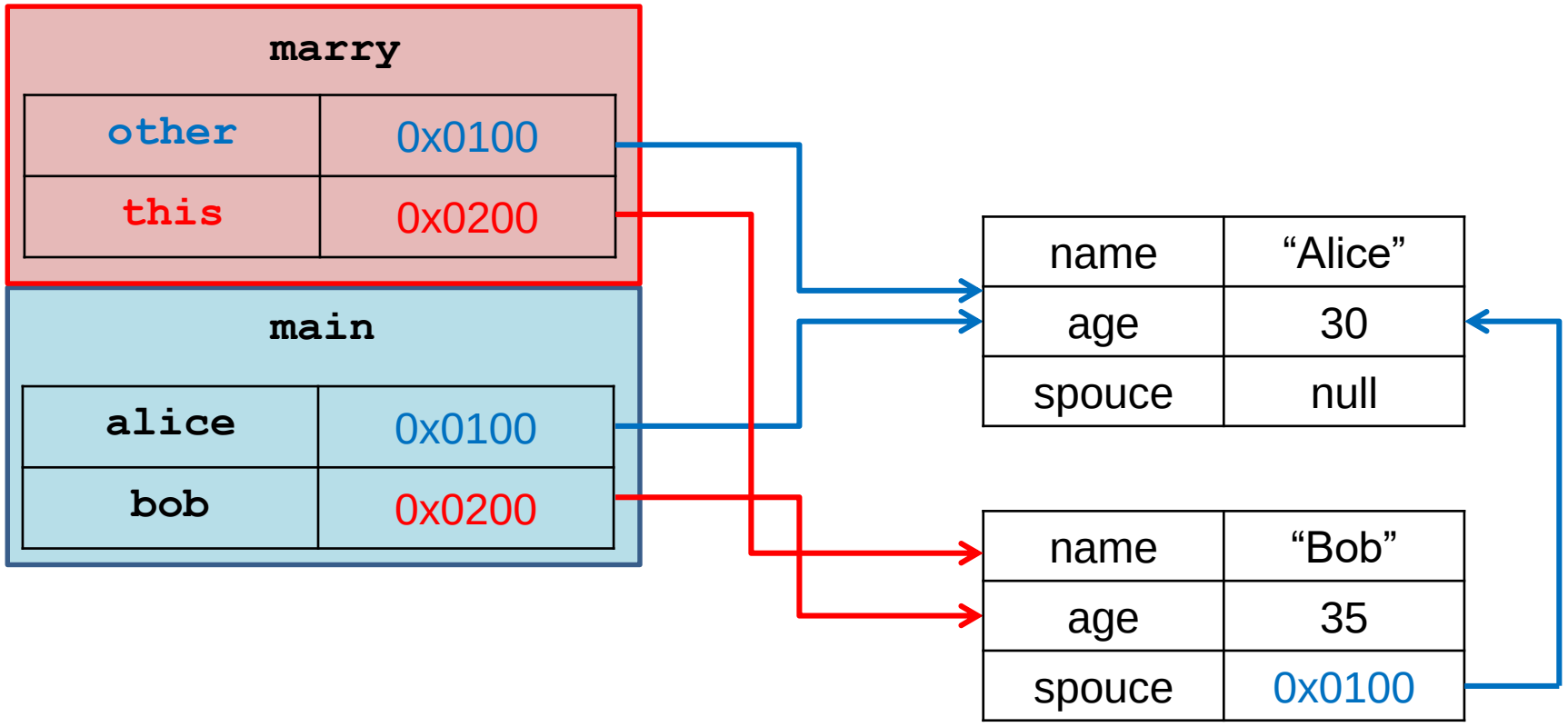
```
class MarriageTest1
{
        public static void main(String[] args){
                Person alice = new Person("Alice", 30);
                Person bob = new Person("Bob", 35);
                Person older = bob.getOlderPerson(alice);
                bob.marry(alice);
        }
}
```


main
alice
0x0100
bob
0x0200
name
“Alice”
age
30
spouce
null
name
“Bob”
age
35
spouce
null

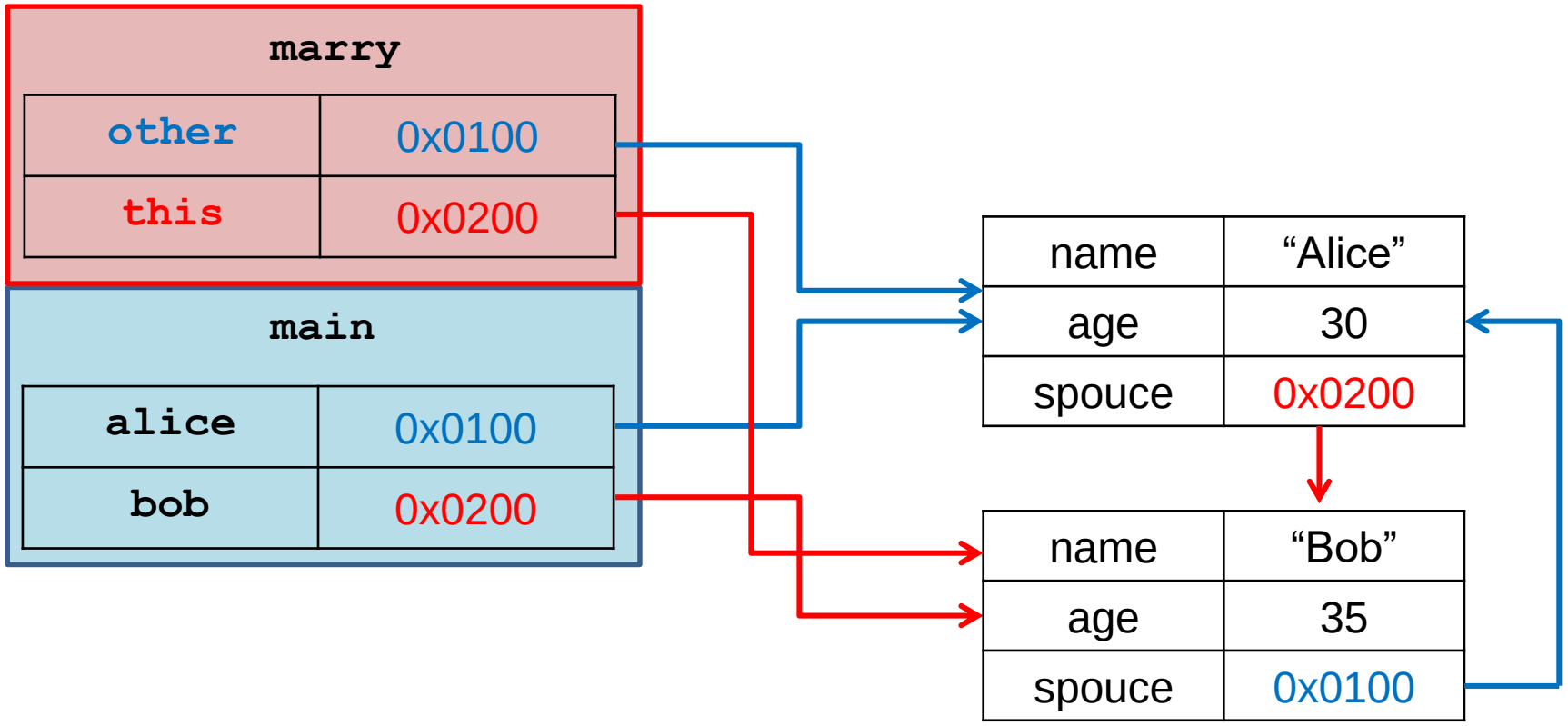
```
bob.marry(alice);
```



```
public void marry(Person other){
        this.spouce = other;
        other.spouce = this;
}
```



```
public void marry(Person other){
        this.spouce = other;
        other.spouce = this;
}
```



Παρένθεση: Τα περιεχόμενα ενός ArrayList μπορεί να είναι οτιδήποτε, π.χ., στην περίπτωση αυτή πίνακες από αντικείμενα Person

```
class Registry
{
        private ArrayList<Person[]> marriages =
                new ArrayList<Person[]>();

        public void addMariage(Person A, Person B){
                Person[] couple = new Person[2];
                couple[0] = A;
                couple[1] = B;
                marriages.add(couple);
        }

}
```

```
class Person
{
    private String name;
    private int age;
    private Person spouce;

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    public Person getOlderPerson(Person other){
        if (this.age > other.age){
            return this;
        }
        return other;
    }

    public void marry(Person other, Registry reg){
        this.spouce = other;
        other.spouce = this;
        reg.addMarriage(this,other)
    }
}
```

Προσθέτουμε ένα νέο ζευγάρι με τα αντικείμενα this και other

```
class MarriageTest2
{
        public static void main(String[] args){
                Person alice = new Person("Alice", 30);
                Person bob = new Person("Bob", 35);
                Registry cityReg = new Registry();
                Person older = bob.getOlderPerson(alice);
                bob.marry(alice,cityReg);
        }
}
```

ΣΥΝΘΕΣΗ

Σύνθεση

- Μια κλάση μπορεί να χρησιμοποιεί αντικείμενα άλλων κλάσεων.
  - Αυτή η διαδικασία ονομάζεται σύνθεση.

- Όταν έχουμε σύνθεση κλάσεων πρέπει να είμαστε προσεκτικοί για το πώς μεταβάλλονται τα περιεχόμενα αντικειμένων στα οποία δείχνουν πολλαπλές αναφορές.

Παραδείγματα

- Τι γίνεται αν έχουμε ένα constructor που παίρνει όρισμα ένα πίνακα?
  - `public Car(int[] position)`
  - Αν ο πίνακας αλλάξει μέσα στην main θα αλλάξει και στο αντικείμενο.

- Τι γίνεται αν στο κάνουμε τον πίνακα null στην main?

```
class CarArray
{
    private int[] position;
    private int dim;

    public CarArray(int[] array){
        dim = array.length;
        position = array;
    }

    public void move(){
        for (int i=0; i < dim; i++){
            position[i] ++;
        }
    }

    public String toString(){
        String output = "";
        for (int i=0; i < dim; i++){
            output = output + position[i] + " ";
        }
        return output;
    }

    public static void main(String args[]){
        int[] pos = {1,2};
        CarArray myCar = new CarArray(pos);
        myCar.move();
        pos[0] ++ ;
        pos = null;
    }
}
```

```
System.out.println(pos[0]+ " " + pos[1]);
System.out.println(myCar);
System.out.println(myCar);
```

Τι θα τυπώσει?

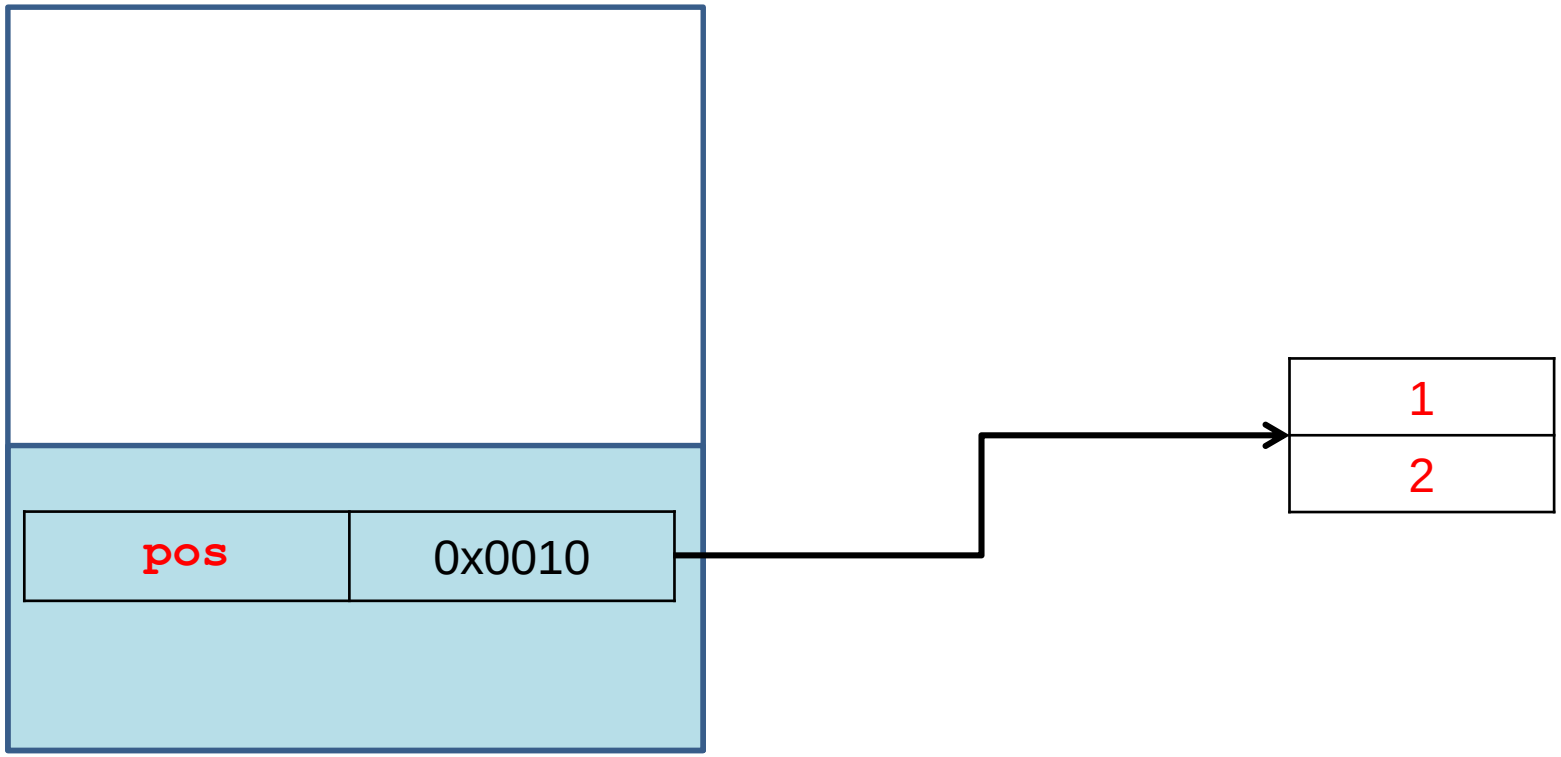
Η αλλαγή στα περιεχόμενα του πίνακα στο αντικείμενο myCar αλλάζει και τα περιεχόμενα του pos

Η αλλαγή στα περιεχόμενα του pos αλλάζει και τα περιεχόμενα του πίνακα στο αντικείμενο myCar

Η αλλαγή στην τιμή του pos δεν αλλάζει τα περιεχόμενα το πεδίο στο αντικείμενο myCar

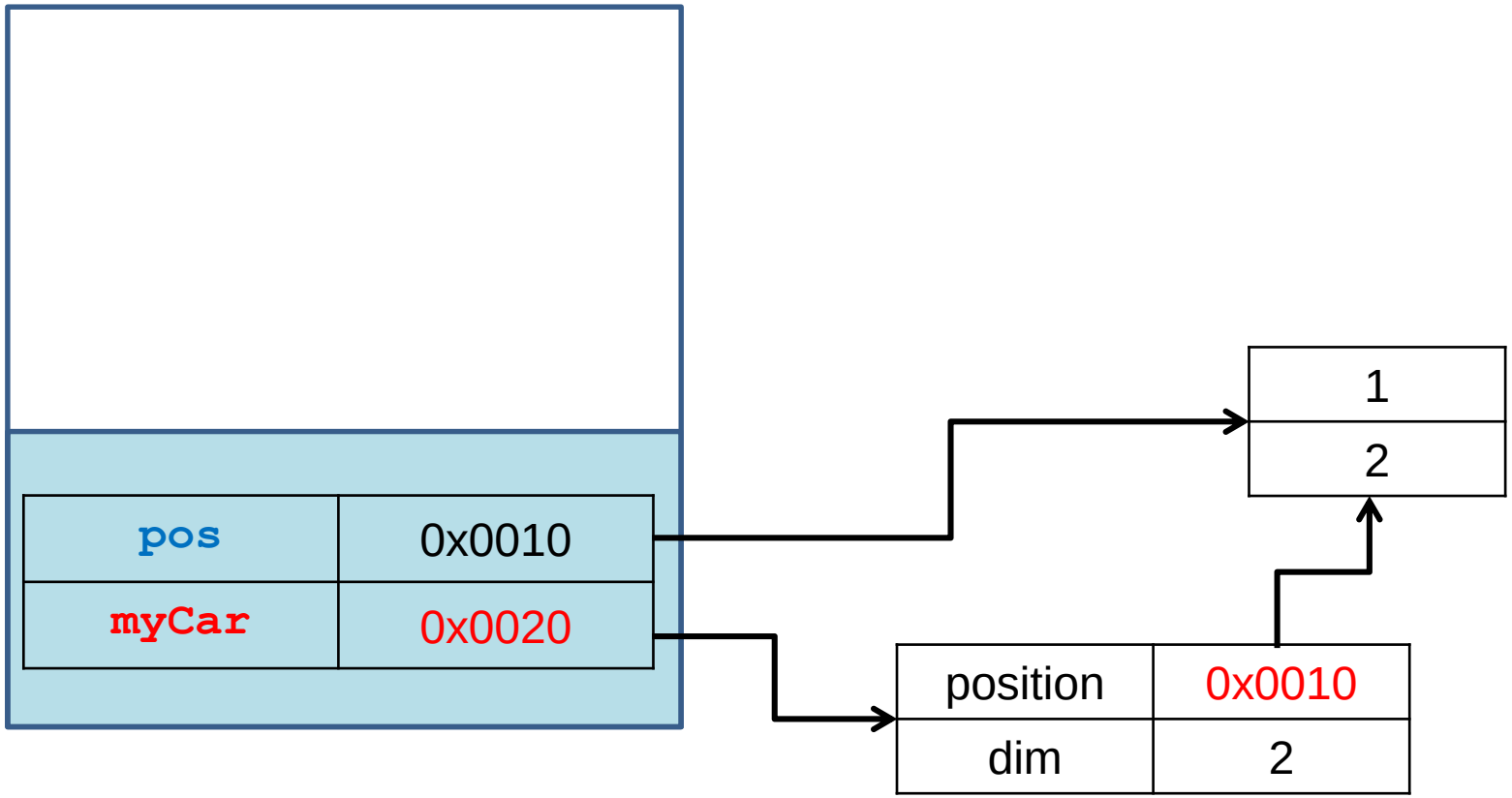
Εκτέλεση

```
int[] pos = {1,2}
```



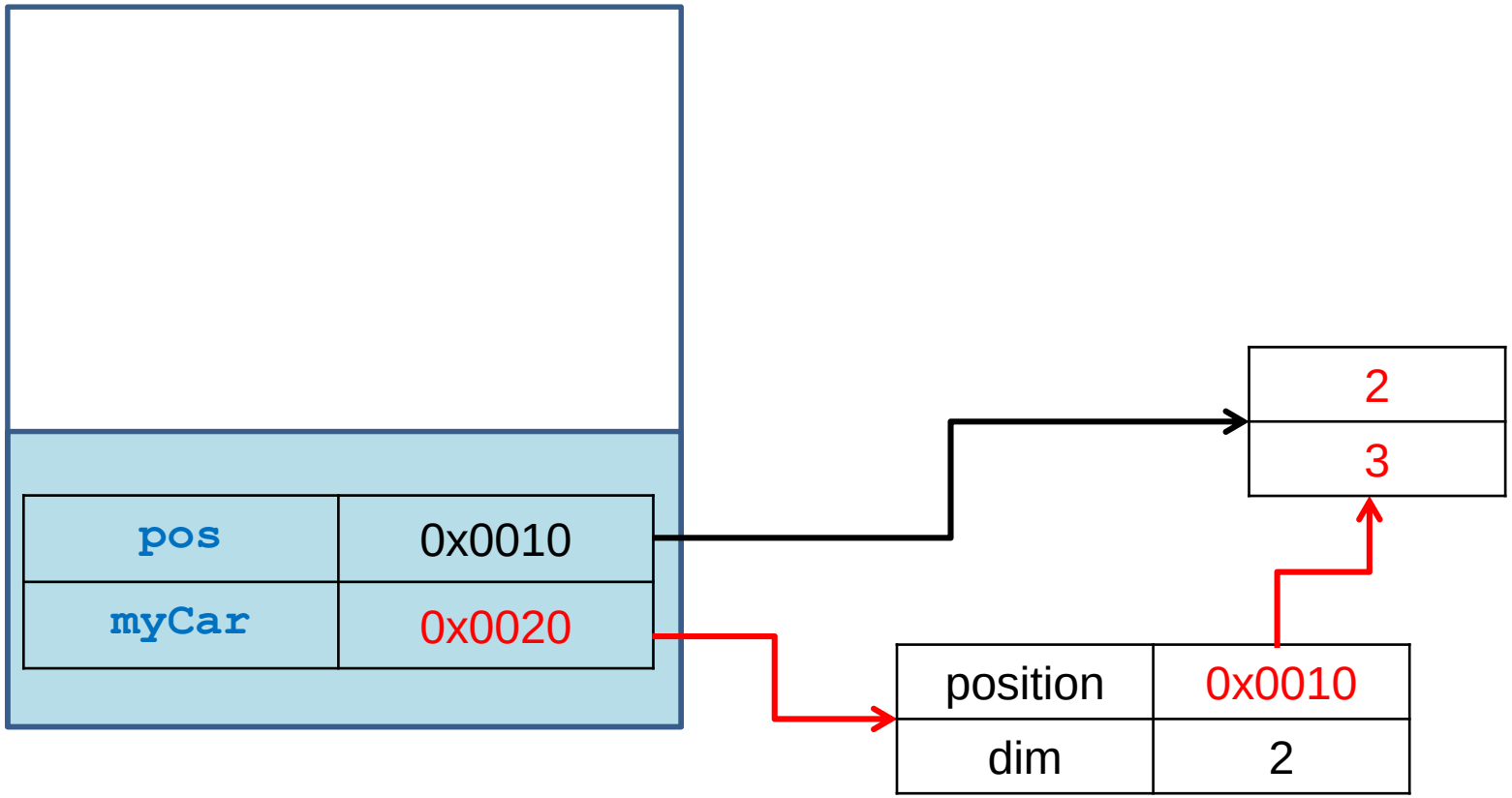
Εκτέλεση

```
int[] pos = {1,2}
CarArray myCar = new CarArray(pos);
```



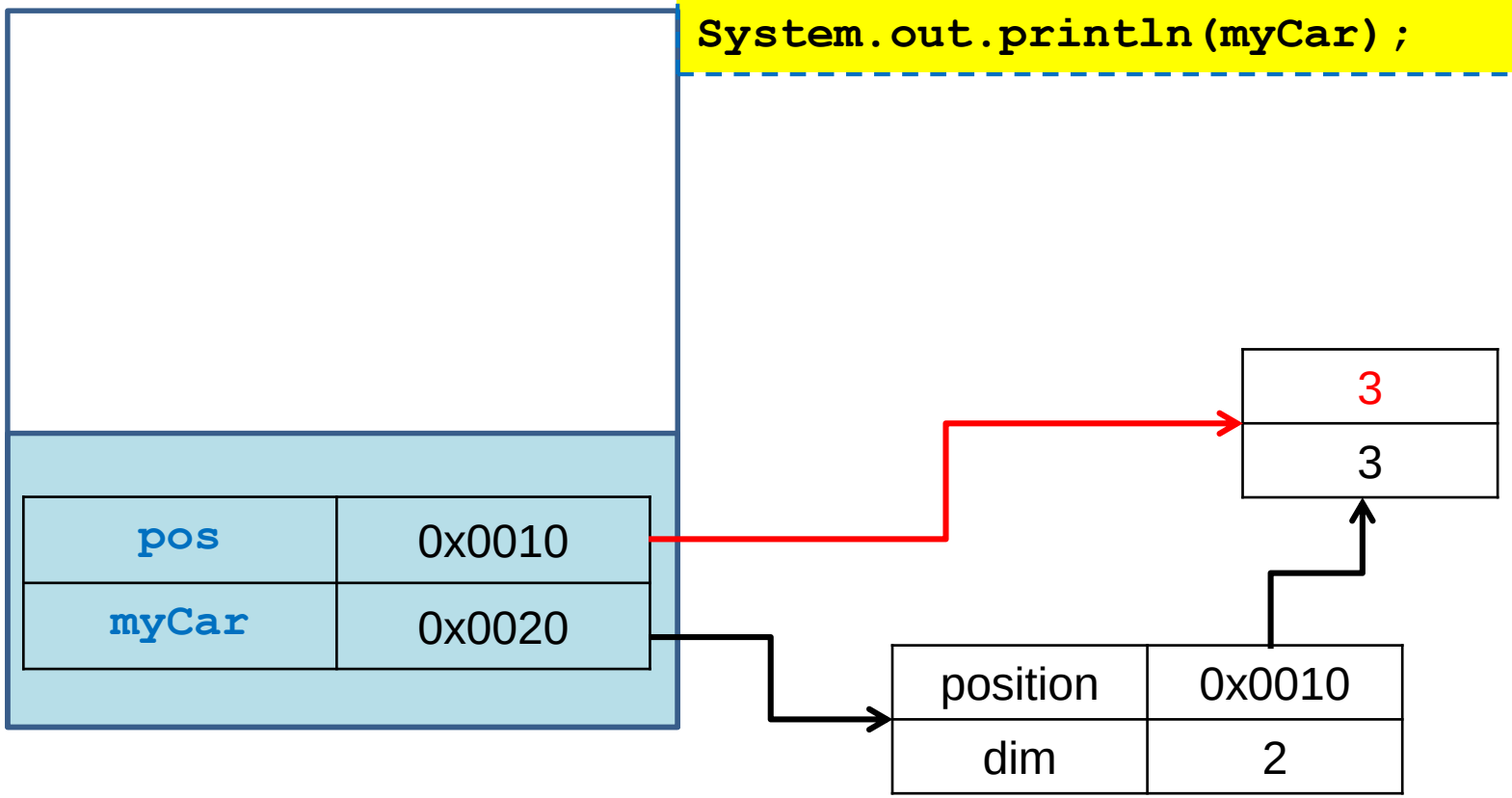
Εκτέλεση

```
int[] pos = {1,2}
CarArray myCar = new CarArray(pos);
myCar.move();
System.out.println(pos[0]+" "+pos[1]);
```



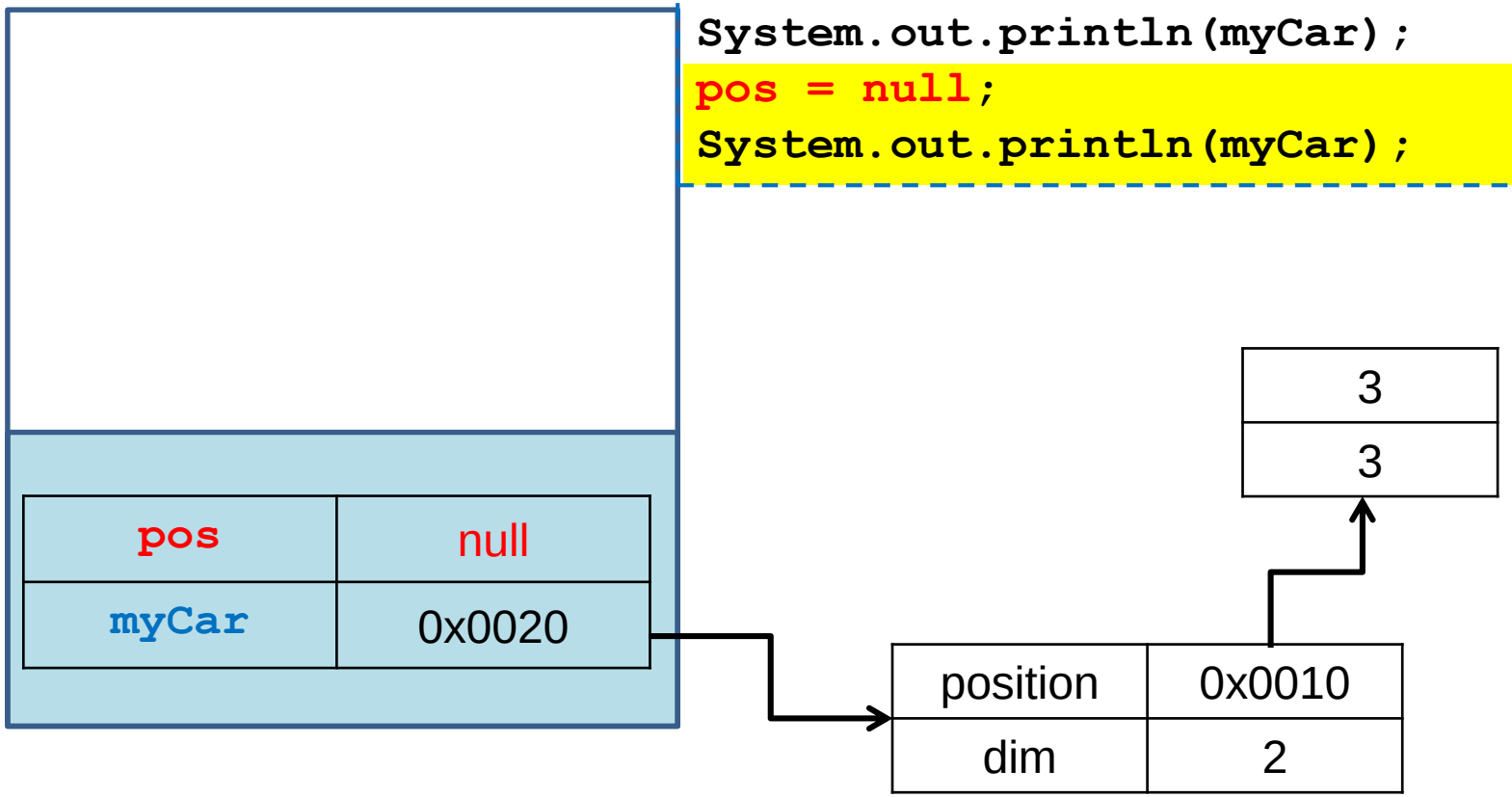
Εκτέλεση

```
int[] pos = {1,2}
CarArray myCar = new CarArray(pos);
myCar.move();
System.out.println(pos[0]+" "+pos[1]);
pos[0] ++ ;
System.out.println(myCar);
```



Εκτέλεση

```
int[] pos = {1,2}
CarArray myCar = new CarArray(pos);
myCar.move();
System.out.println(pos[0]+" "+pos[1]);
pos[0] ++ ;
System.out.println(myCar);
pos = null;
System.out.println(myCar);
```



```
class Person
{
  private String name;

  public Person(String name){
    this.name = name;
  }

  public String getName(){
    return name;
  }

  public void setName(String name)
  {
    this.name = name;
  }
}
```

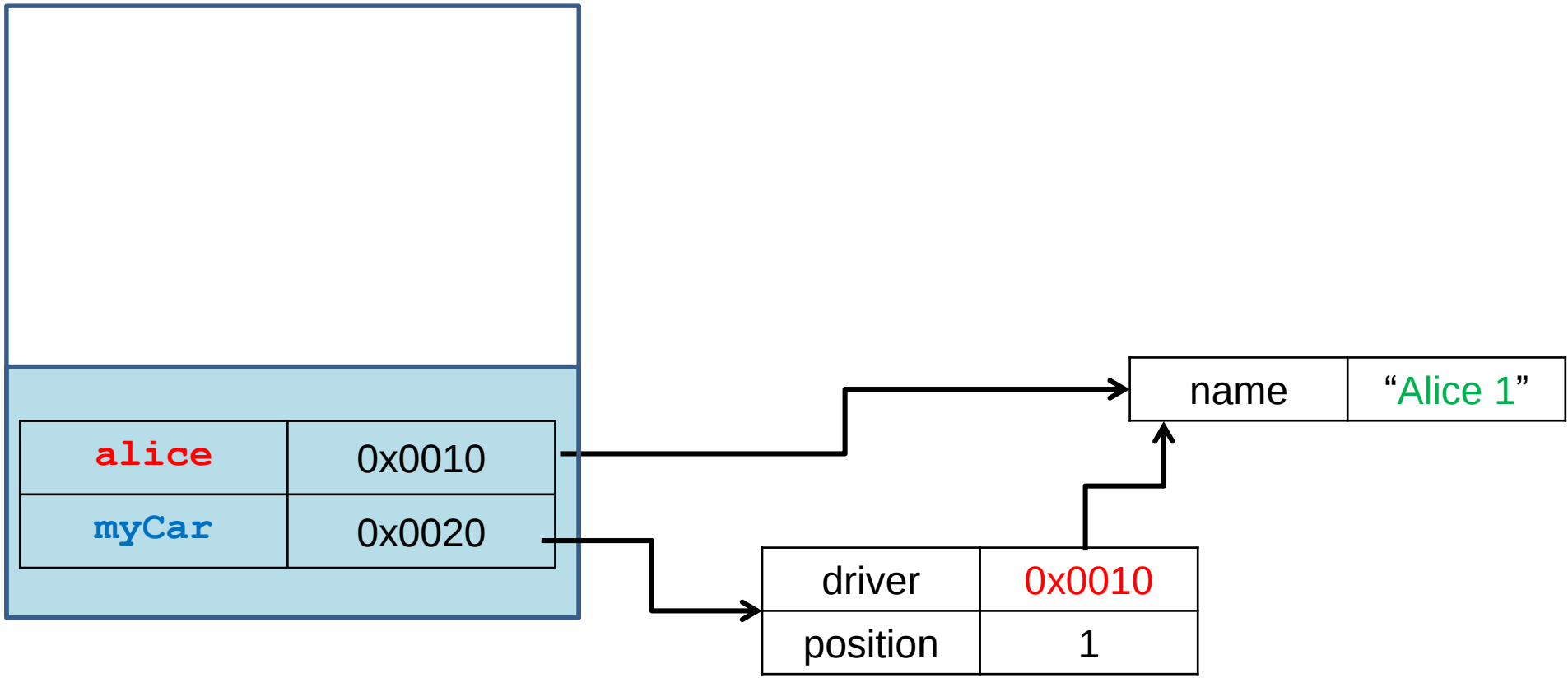
```
class Car
{
  private int position = 0;
  private Person driver;

  public Car(int position, Person driver){
    this.position = position;
    this.driver = driver;
  }

  public Person getDriver(){
    return driver;
  }
}
```

```
class MovingCarDriver3
{
  public static void main(String args[]){
    Person alice = new Person("Alice 1");
    Car myCar = new Car(1, alice);
    alice.setName("Alice 2");
    System.out.println(myCar.getDriver().getName());
    alice = new Person("Alice 3");
    System.out.println(myCar.getDriver().getName());
  }
}
```

Τι θα τυπώσει?

Εκτέλεση

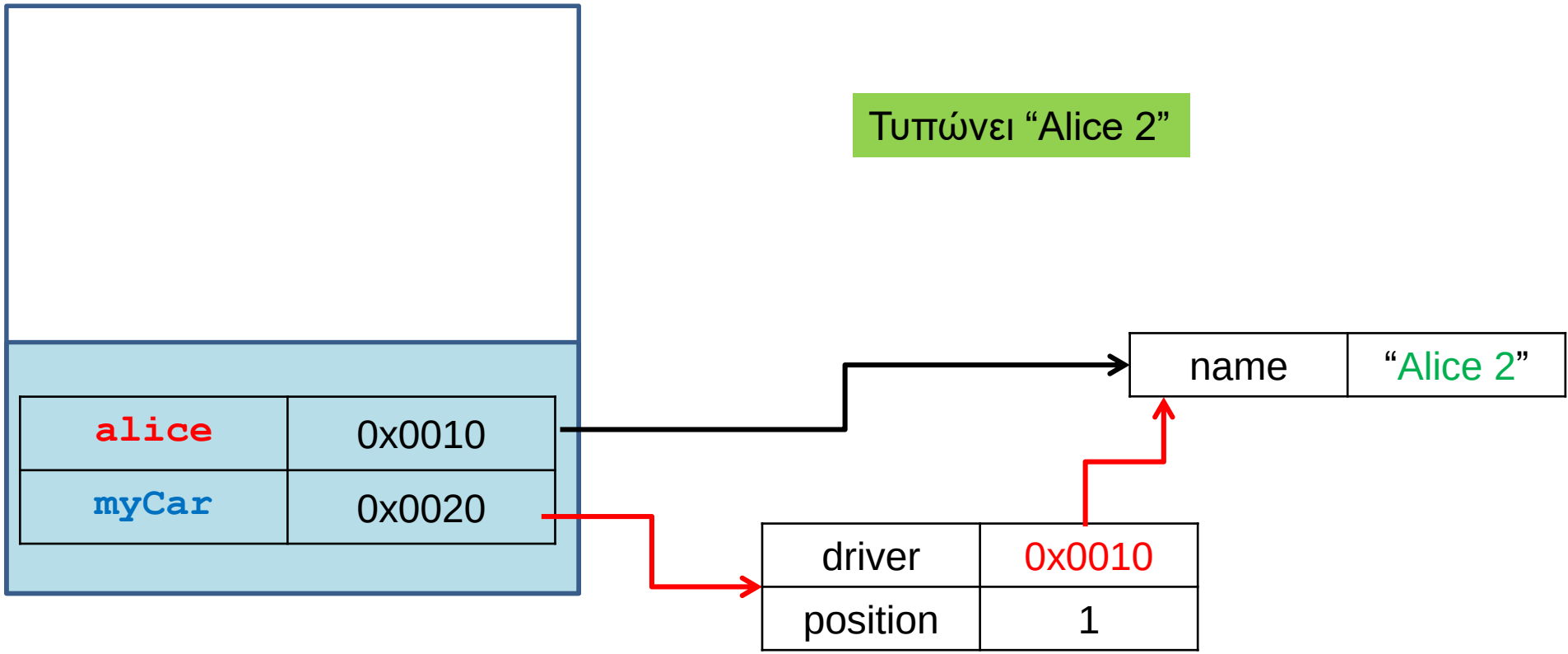
```
Person alice = new Person("Alice 1");
Car myCar = new Car(1, alice);
```



Εκτέλεση

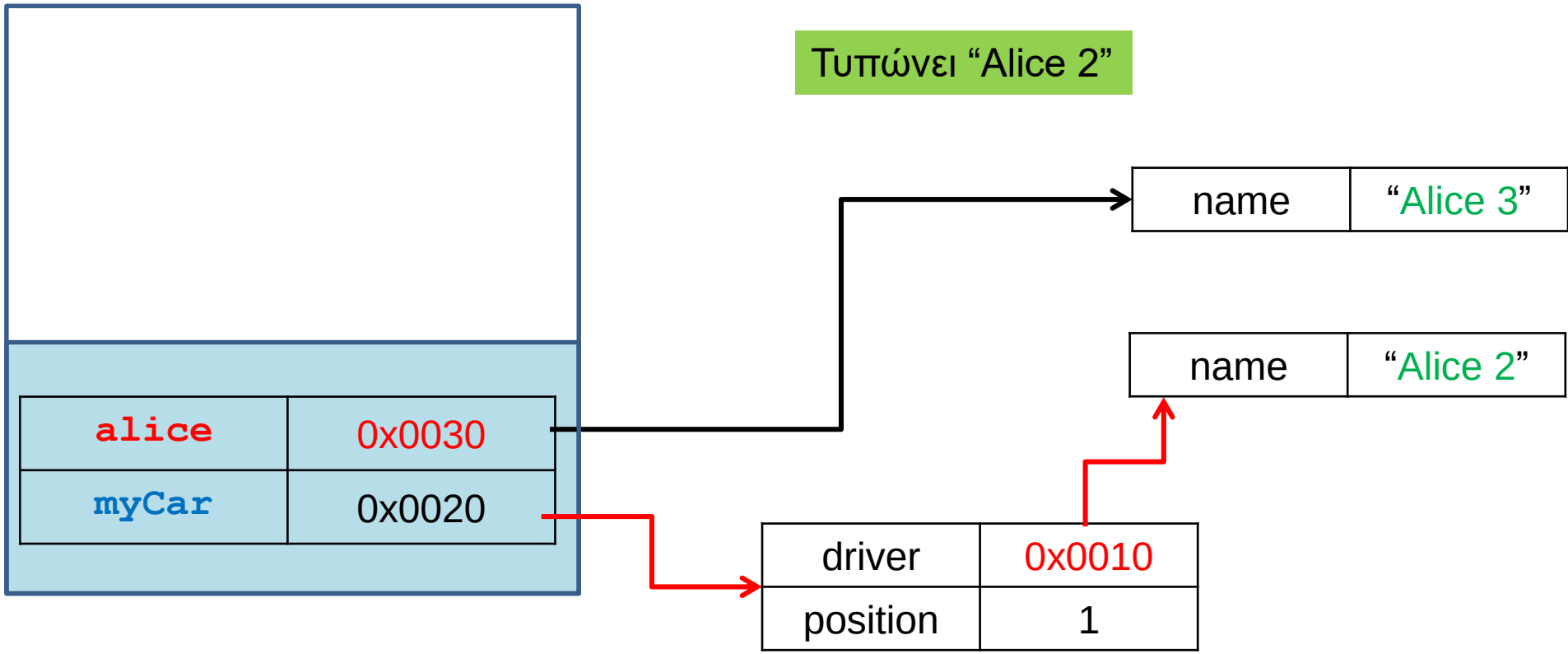
```
alice.setName("Alice 2");
System.out.println(myCar.getDriver().getName());
```

Τυπώνει “Alice 2”



Εκτέλεση

```
alice = new Person("Alice 3");
System.out.println(myCar.getDriver().getName());
```



Αντικείμενα μέσα σε αντικείμενα

- Ορίζουμε κλάσεις για να ορίσουμε τύπους δεδομένων τους οποίους χρειαζόμαστε
  - Π.χ., ο τύπος δεδομένων Person για να μπορούμε να χειριζόμαστε πληροφορίες για ένα άτομο, και ο τύπος δεδομένων Car που κρατάει πληροφορία για το αυτοκίνητο.
- Τους τύπους δεδομένων που ορίζουμε τους χρησιμοποιούμε για να δημιουργήσουμε μεταβλητές (αντικείμενα).
- Τα αντικείμενα μπορεί να είναι πεδία άλλων κλάσεων
  - Π.χ., η κλάση Car έχει ένα πεδίο τύπου Person
- Μία κλάση χρησιμοποιεί αντικείμενα άλλων κλάσεων και έτσι συνθέτουμε πιο περίπλοκους τύπους δεδομένων.