ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Σύνθεση αντικειμένων

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?

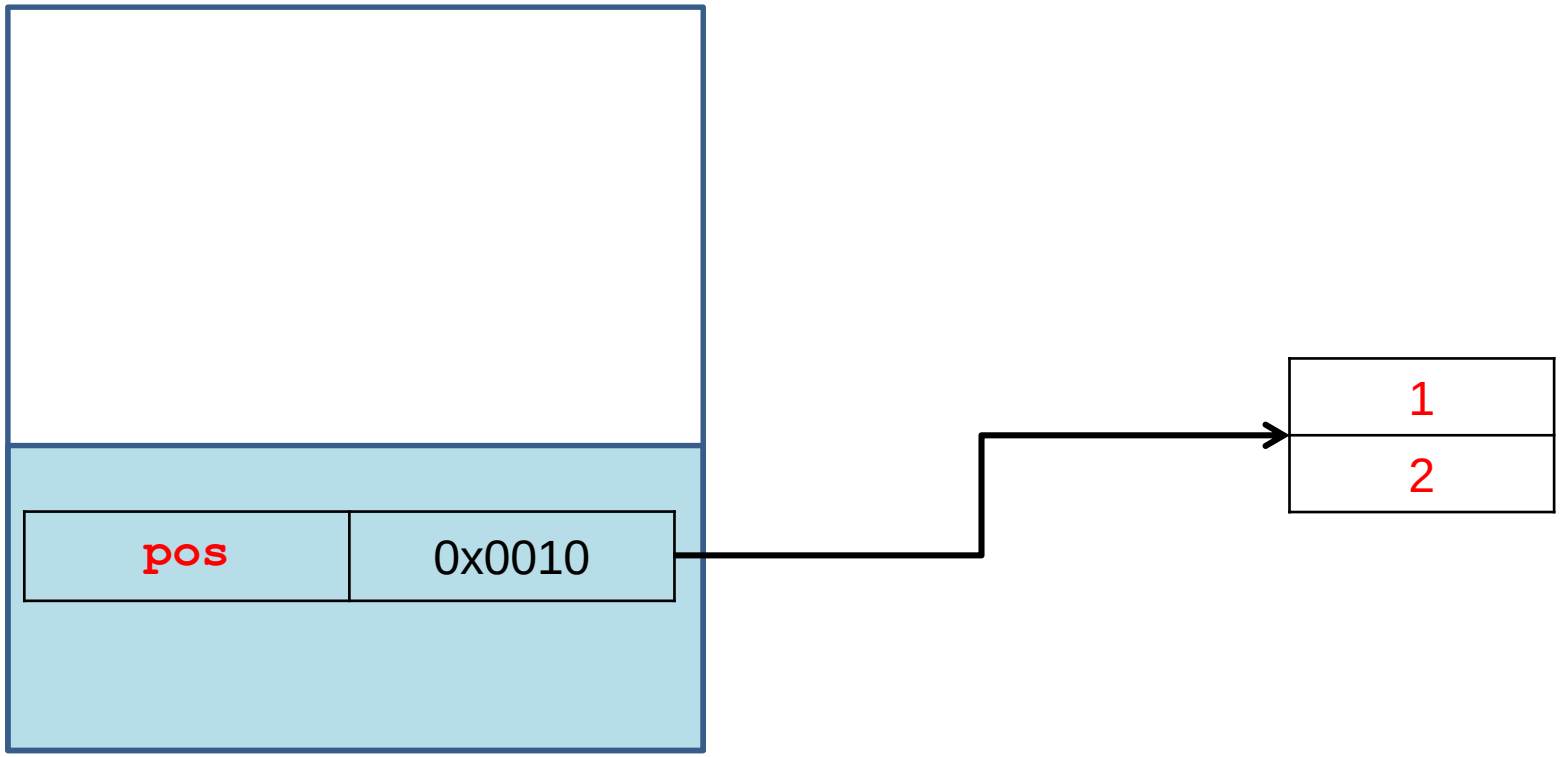
Η αλλαγή στα περιεχόμενα του πίνακα στο αντικείμενο myCar αλλάζει και τα περιεχόμενα του pos

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

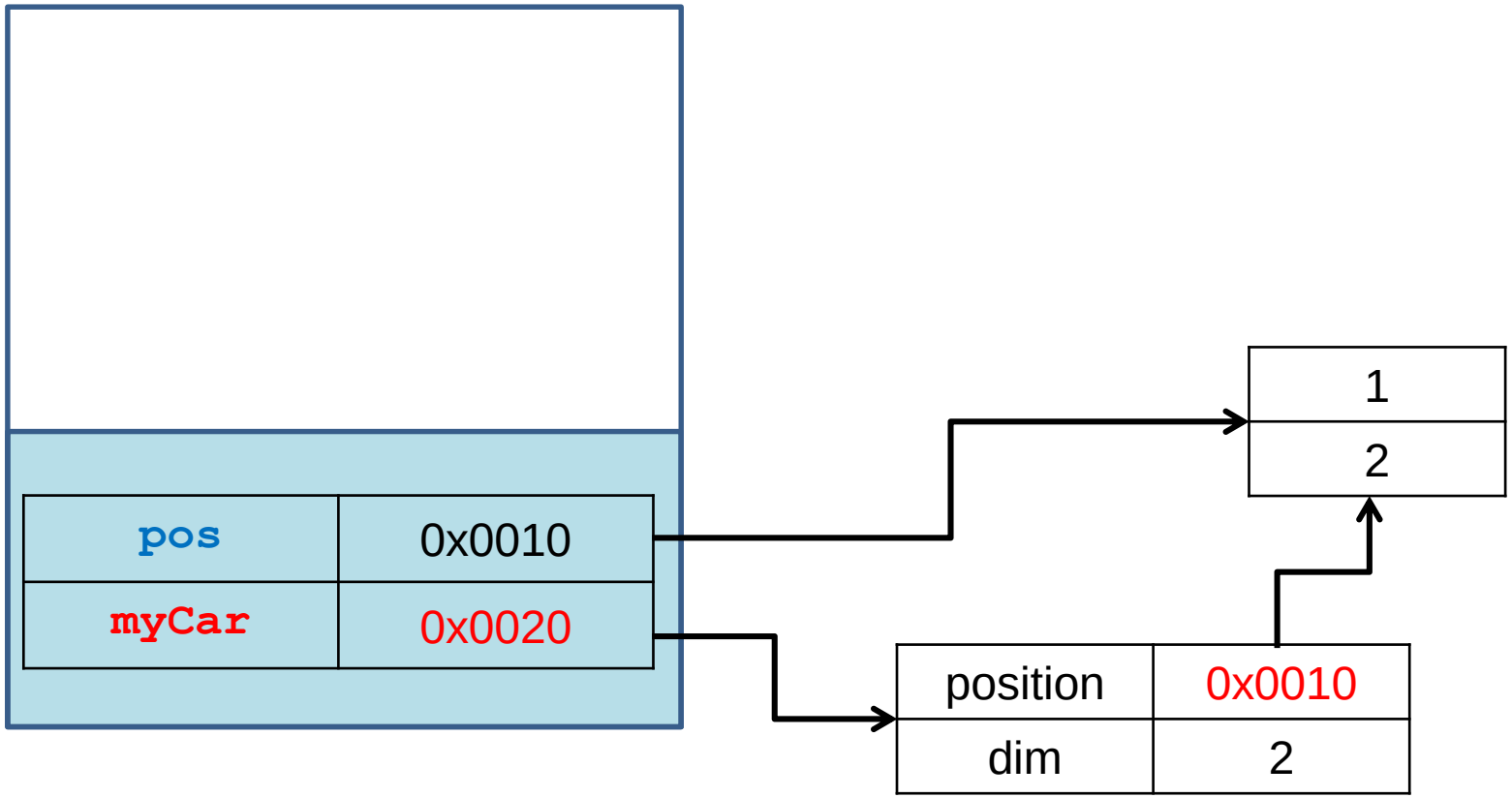
Εκτέλεση

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



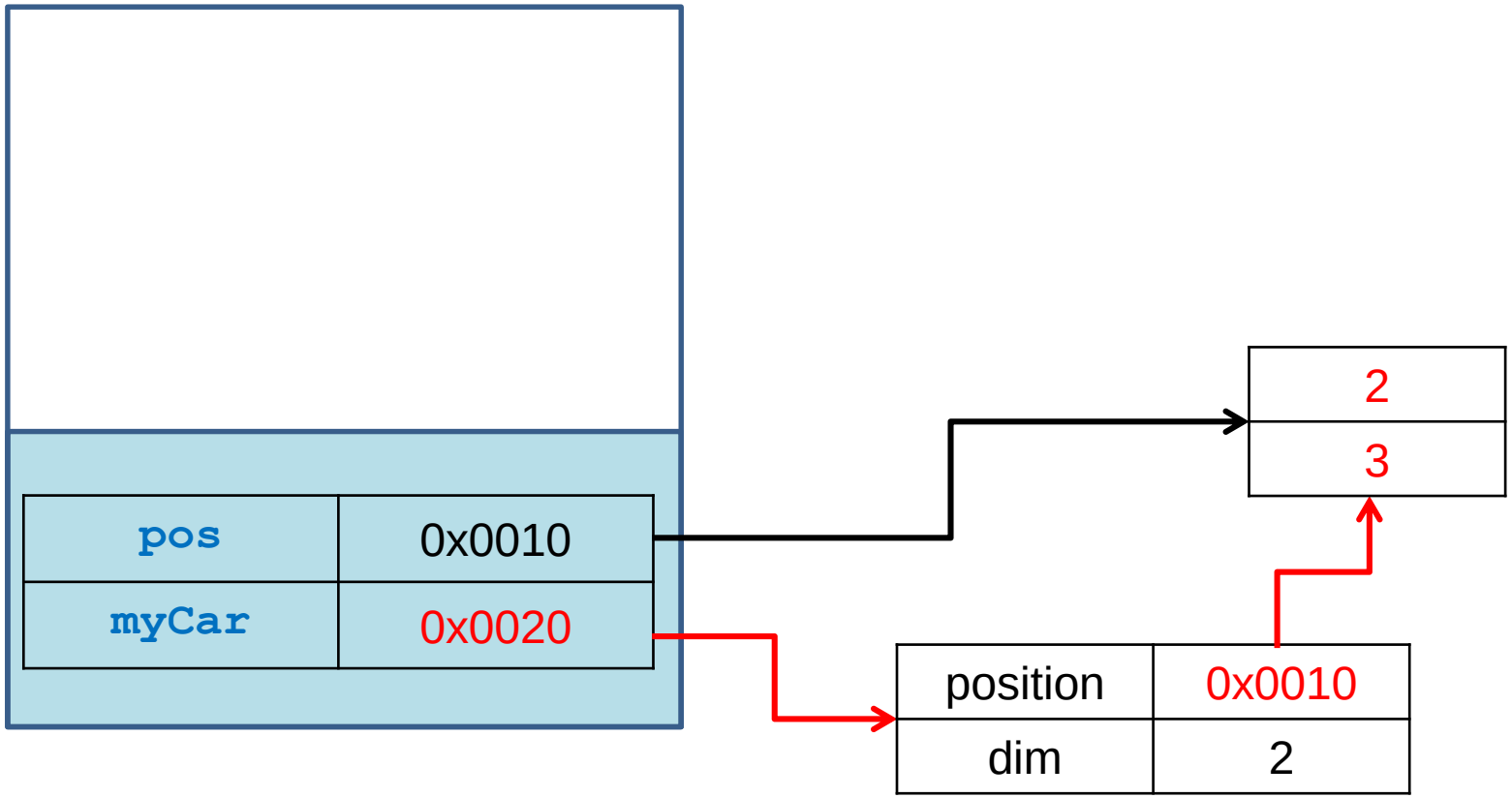
Εκτέλεση

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



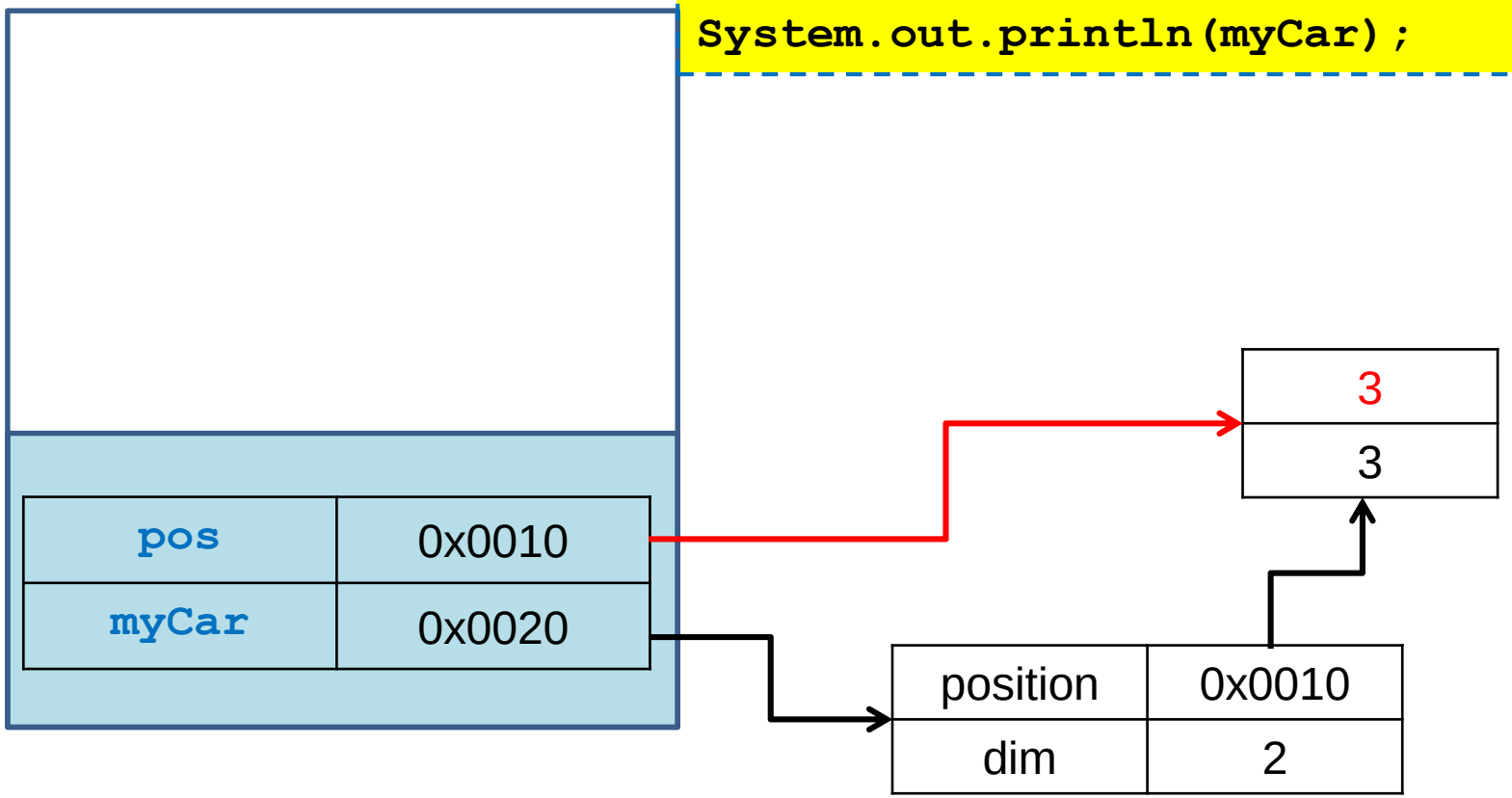
Εκτέλεση

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



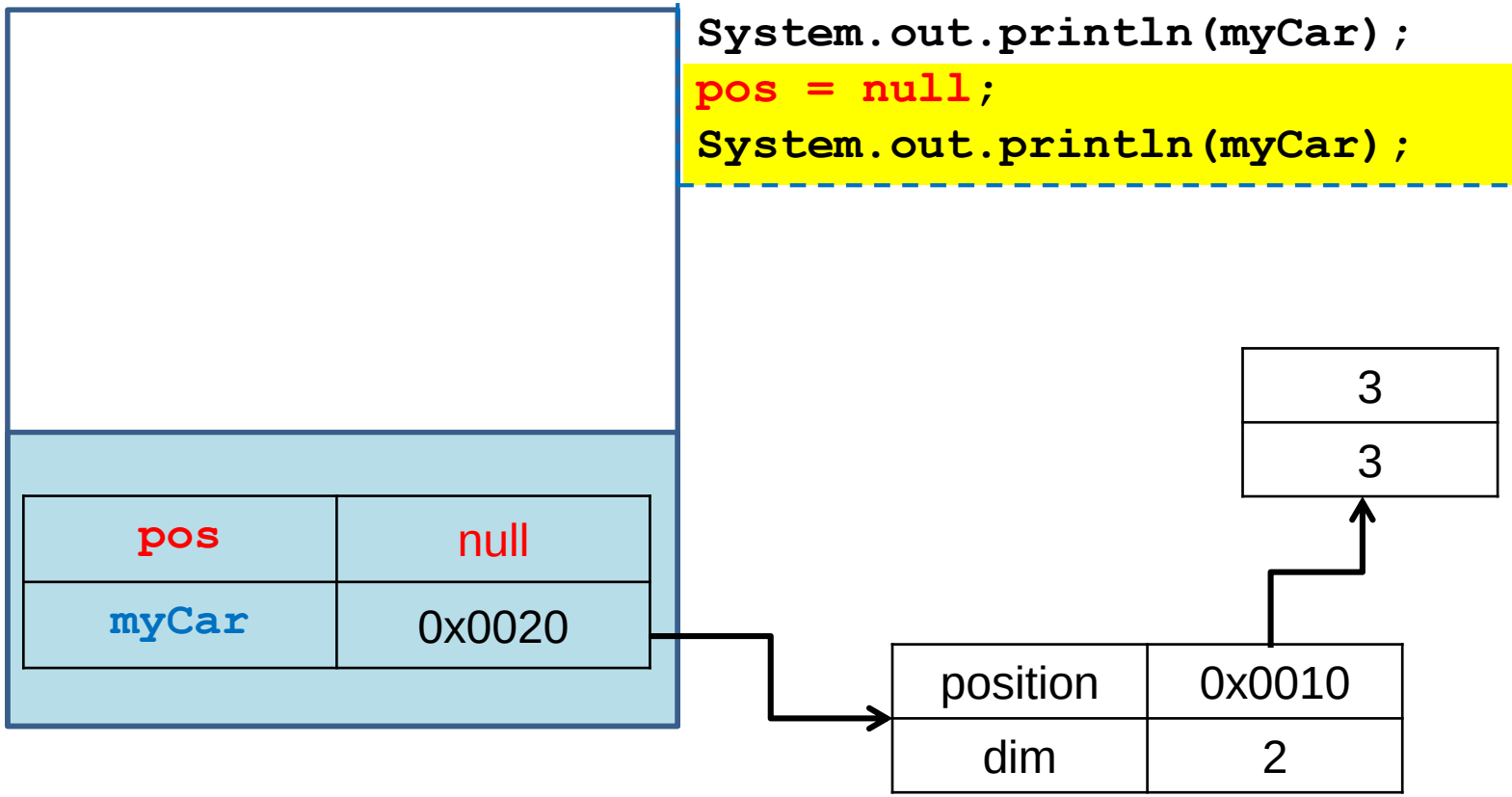
Εκτέλεση

```
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;
  }
}
```

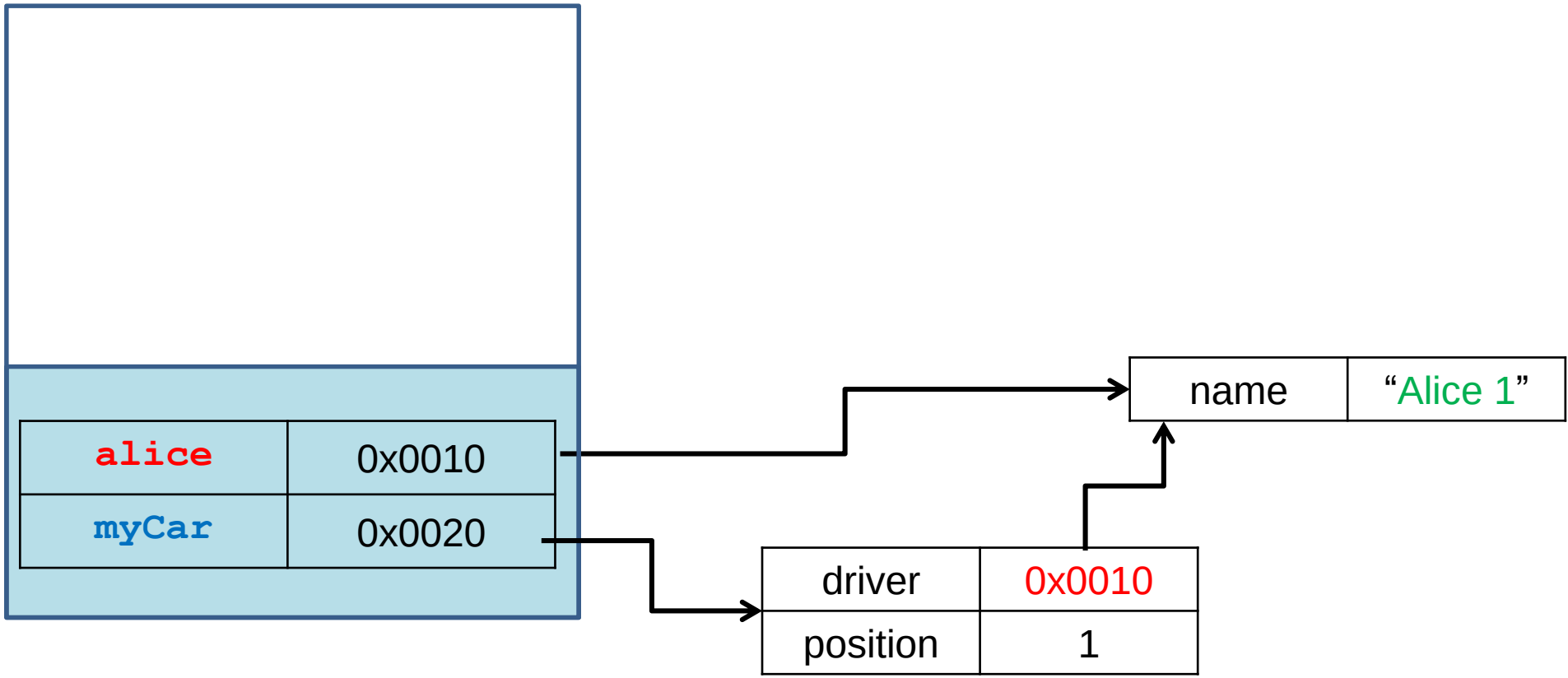
```
class Car
{
  private int position = 0;
  private Person driver;

  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

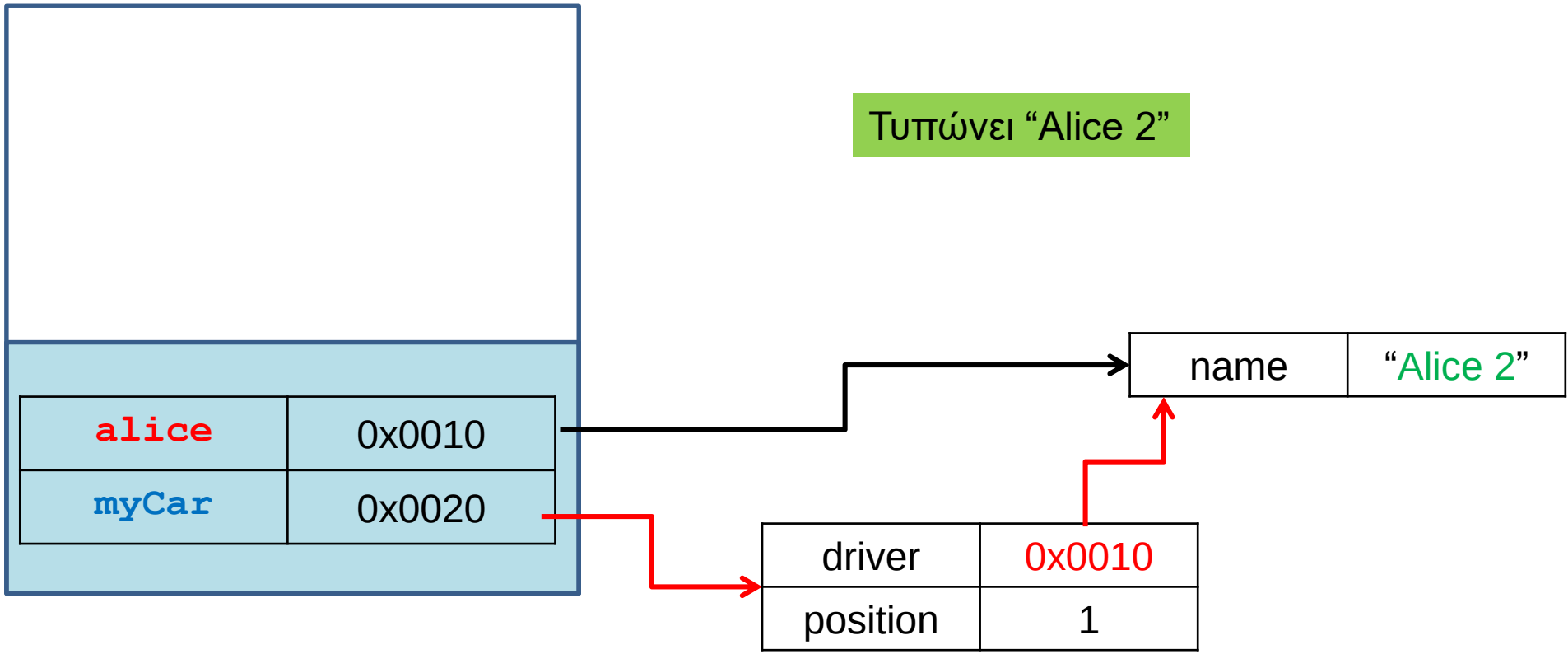
```
Person alice = new Person("Alice 1");
Car myCar = new Car(1, alice);
```



Εκτέλεση

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

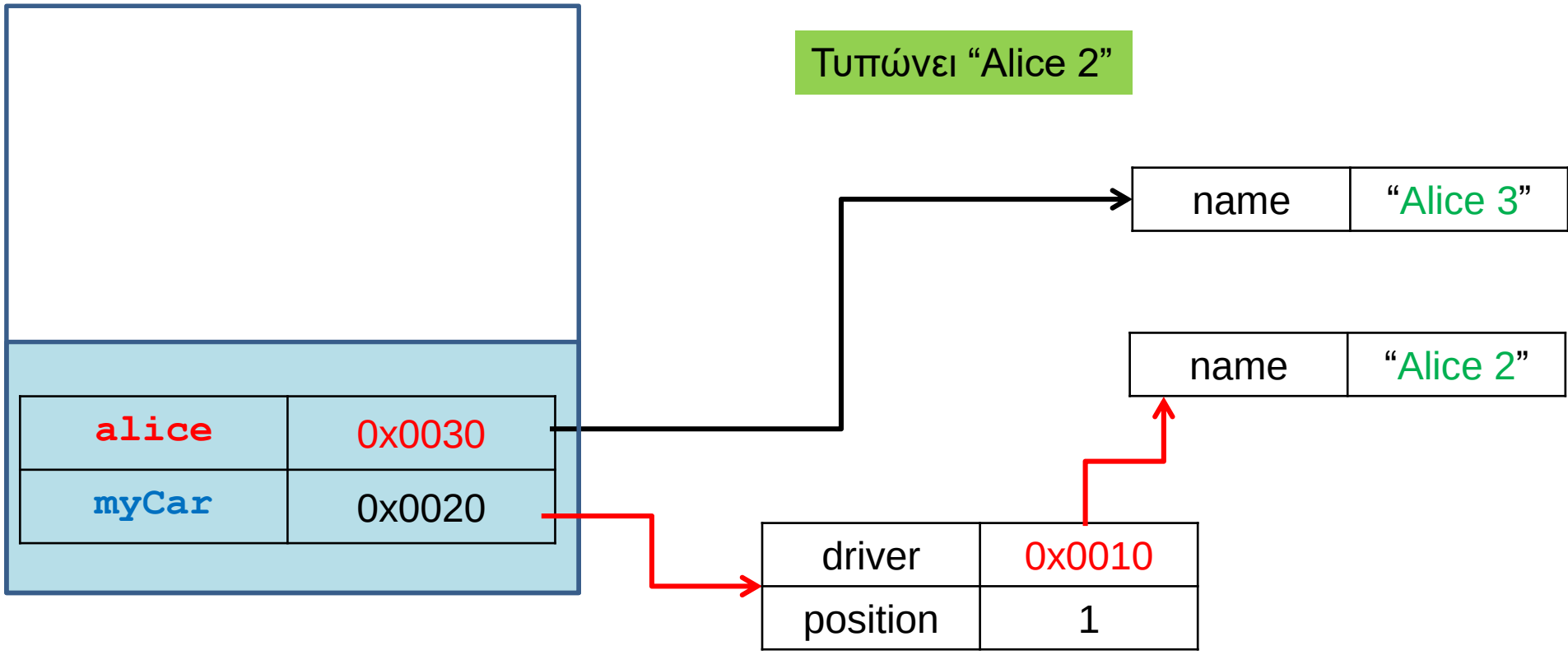
Τυπώνει “Alice 2”



Εκτέλεση

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

Τυπώνει “Alice 2”

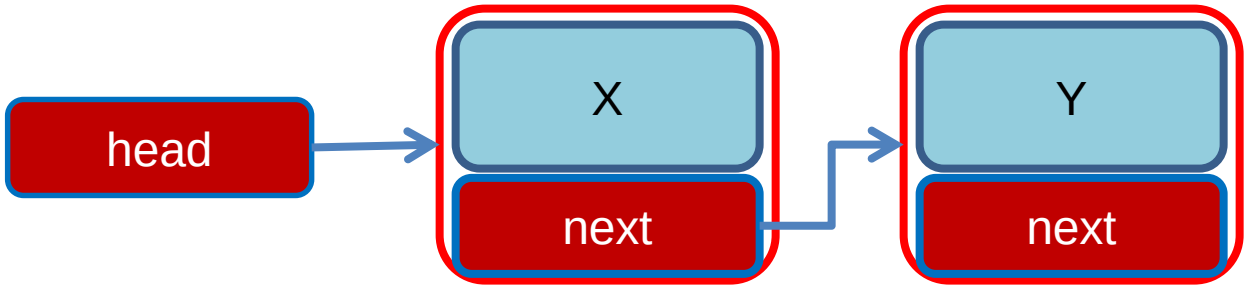


Αντικείμενα μέσα σε αντικείμενα

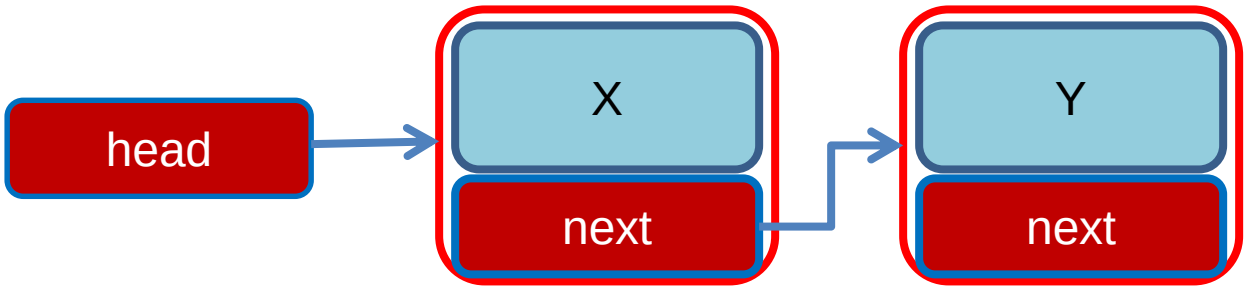
- Ορίζουμε κλάσεις για να ορίσουμε τύπους δεδομένων τους οποίους χρειαζόμαστε
  - Π.χ., ο τύπος δεδομένων Person για να μπορούμε να χειριζόμαστε πληροφορίες για ένα άτομο, και ο τύπος δεδομένων Car που κρατάει πληροφορία για το αυτοκίνητο.
- Τους τύπους δεδομένων που ορίζουμε τους χρησιμοποιούμε για 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.

Παράδειγμα

- Υλοποιήστε το Stack που φτιάξαμε στα προηγούμενα μαθήματα ώστε να μην έχει περιορισμό στο μέγεθος (capacity).
- Βασική ιδέα:
  - Δημιουργούμε στοιχεία της στοίβας και τα συνδέουμε το ένα να δείχνει στο άλλο.
  - Χρειάζεται να ξέρουμε και την κορυφή της στοίβας.

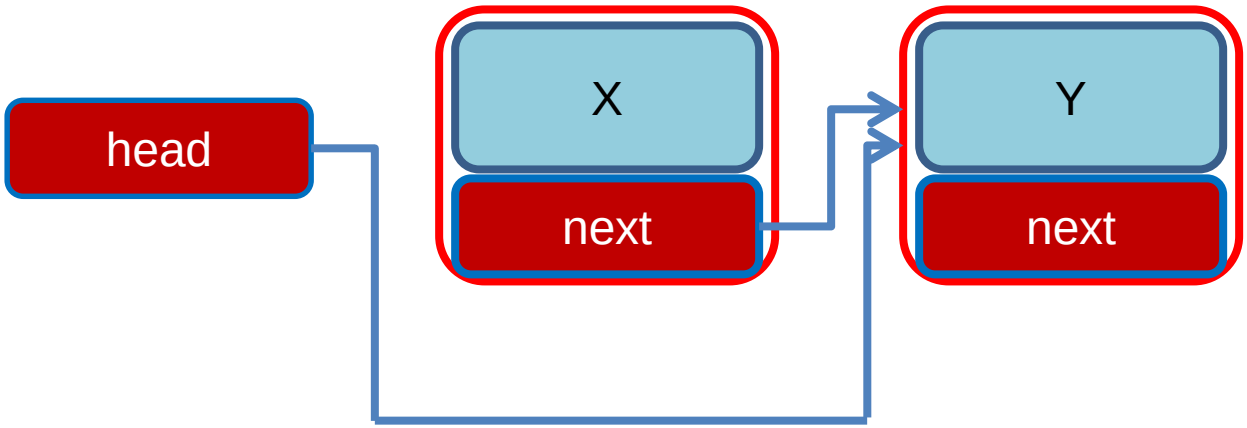


Στοίβα



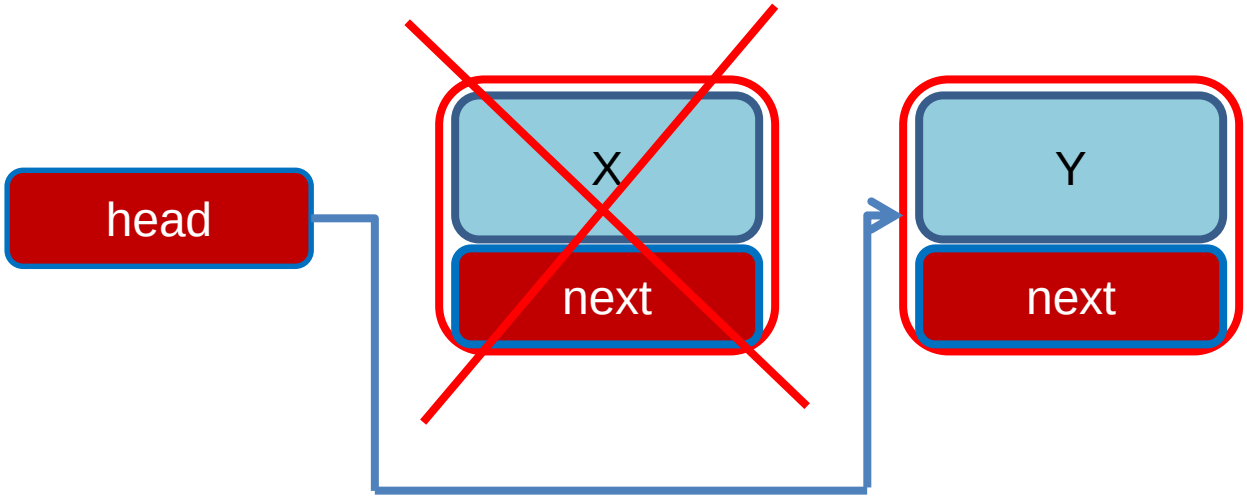
Pop(): Αφαιρεί το στοιχείο στην κορυφή της στοίβας και επιστρέφει την τιμή του (X στο παράδειγμα μας)

Στοίβα



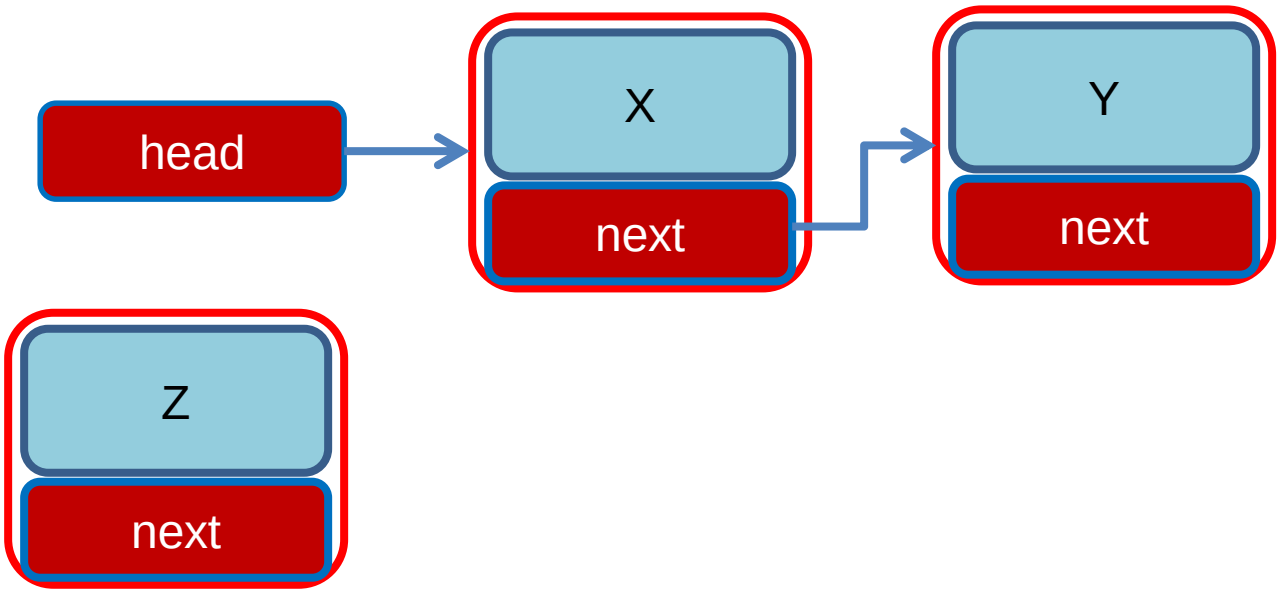
Pop(): Αφαιρεί το στοιχείο στην κορυφή της στοίβας και επιστρέφει την τιμή του (X στο παράδειγμα μας)

Στοίβα



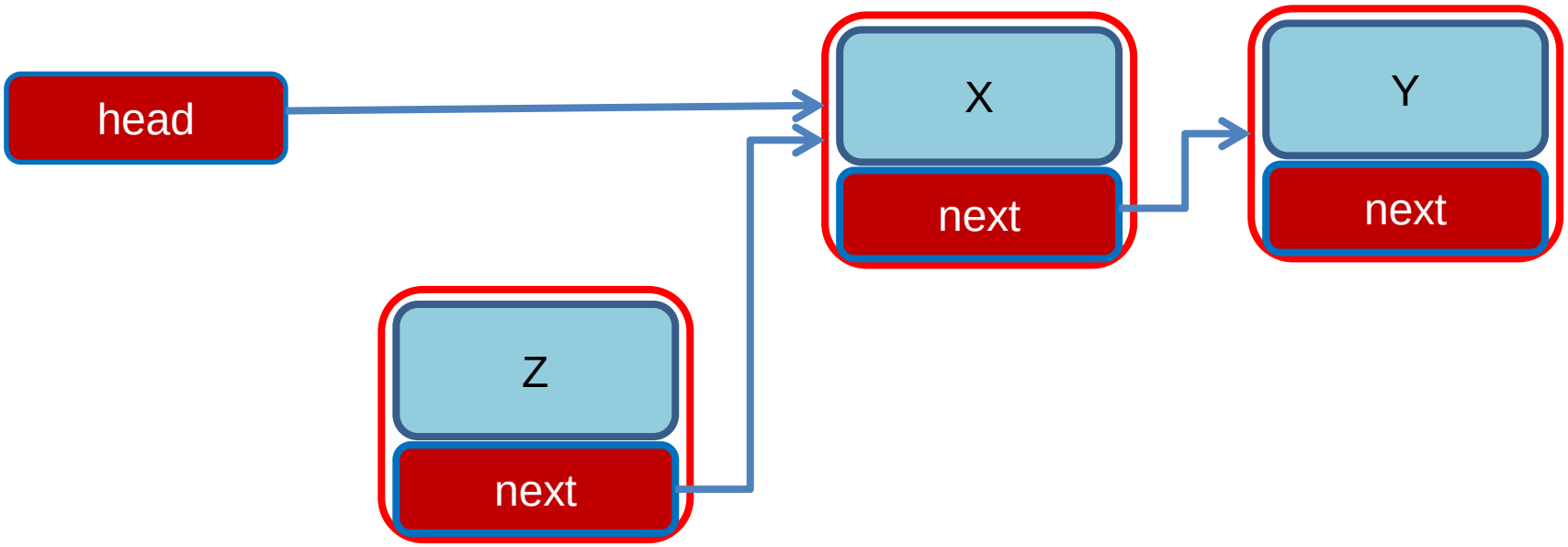
Pop(): Αφαιρεί το στοιχείο στην κορυφή της στοίβας και επιστρέφει την τιμή του (X στο παράδειγμα μας)

Στοίβα



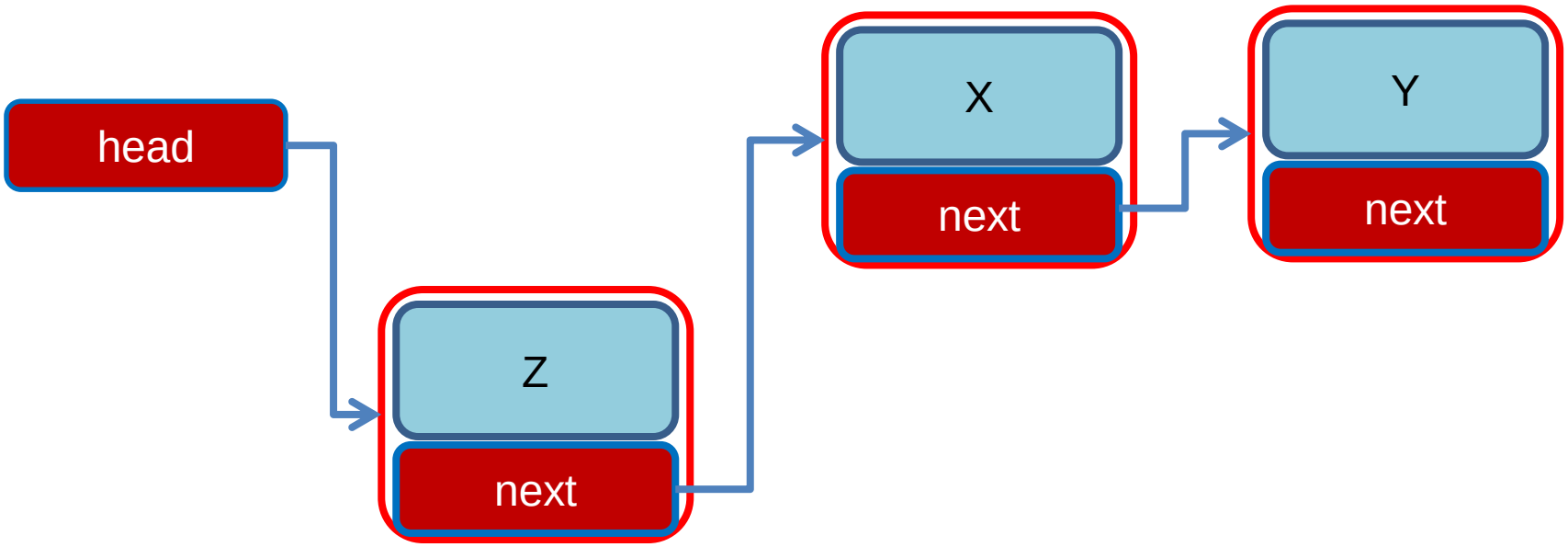
Push(Z): Προσθέτει την τιμή Z στην κορυφή της στοίβας

Στοίβα



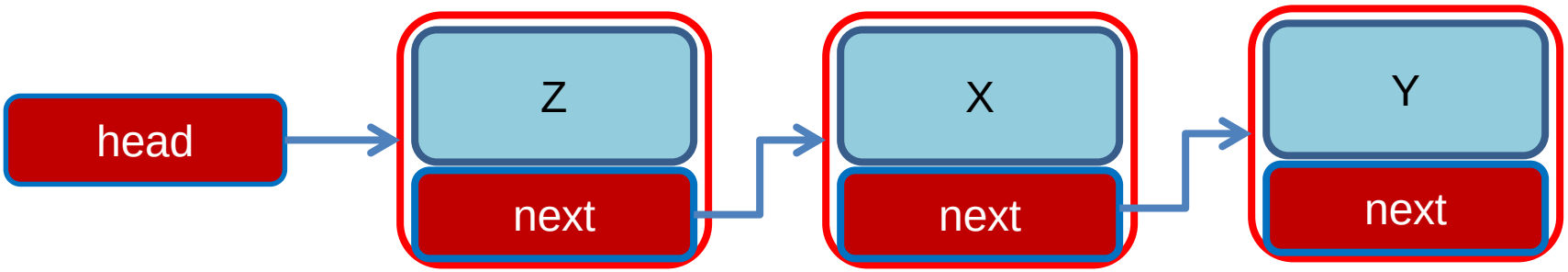
Push(Z): Προσθέτει την τιμή Z στην κορυφή της στοίβας

Στοίβα



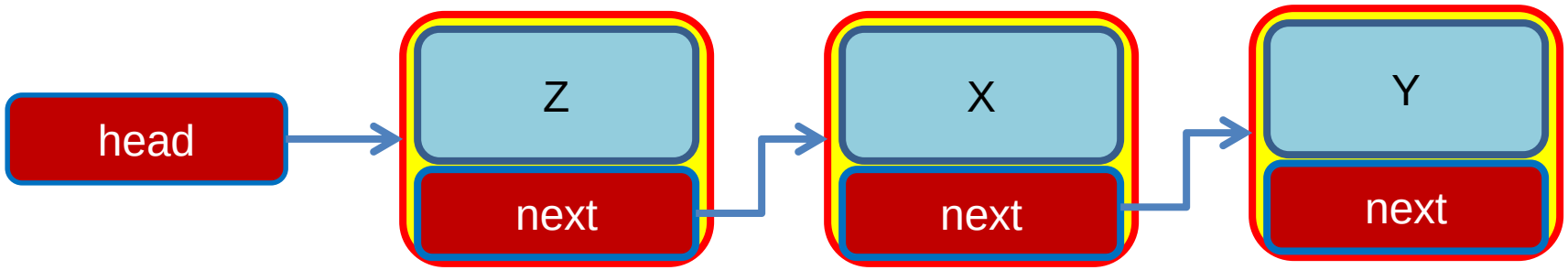
Push(Z): Προσθέτει την τιμή Z στην κορυφή της στοίβας

Στοίβα



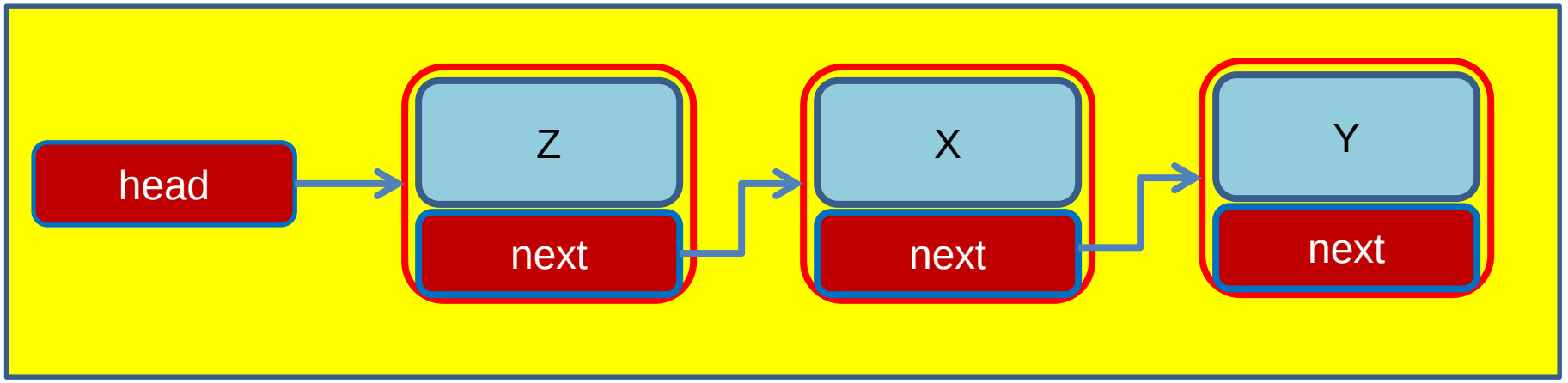
Push(Z): Προσθέτει την τιμή Z στην κορυφή της στοίβας

Στοίβα - Υλοποίηση



- Θα ορίσουμε StackElement μια κλάση που κρατάει το κάθε στοιχείο της στοίβας.

Στοίβα - Υλοποίηση



- Θα ορίσουμε StackElement μια κλάση που κρατάει το κάθε στοιχείο της στοίβας.
- Και μια κλάση Stack που υλοποιεί την στοίβα και όλες τις λειτουργίες της

```
class StackElement
{
        private int value;
        private StackElement next = null;

        public StackElement(int value){
                this.value = value;
        }

        public int getValue(){
                return value;
        }

        public StackElement getNext(){
                return next;
        }

        public void setNext(StackElement element){
                next = element;
        }

}
```

Το επόμενο στοιχείο

Επιστρέφει αντικείμενο

```
class Stack
{
        private StackElement head;
        private int size = 0;

        public int pop(){
                if (size == 0){ // head == null
                        System.out.println("Pop from empty stack");
                        System.exit(-1);
                }
                int value = head.getValue();
                head = head.getNext();
                size --;
                return value;
        }

        public void push(int value){
                StackElement element = new StackElement(value);
                element.setNext(head);
                head = element;
                size ++;
        }

}
```

Το πρώτο στοιχείο της στοίβας μας φτάνει για τα βρούμε όλα

Σταματάει την εκτέλεση του προγράμματος

Τα αντικείμενα τύπου StackElement δημιουργούνται μέσα στην Stack.

Μέθοδος toString()

```
public String toString(){
     String returnString = "";
     IntStackElement e = head;
     for (int i = 0; i < size; i ++){
        returnString = returnString + e.getValue() + " ";
        e = e.getNext();
     }
     return returnString;
}
```

Χρειαζόμαστε μία StackElement μεταβλητή για να διατρέξει τα στοιχεία της στοίβας

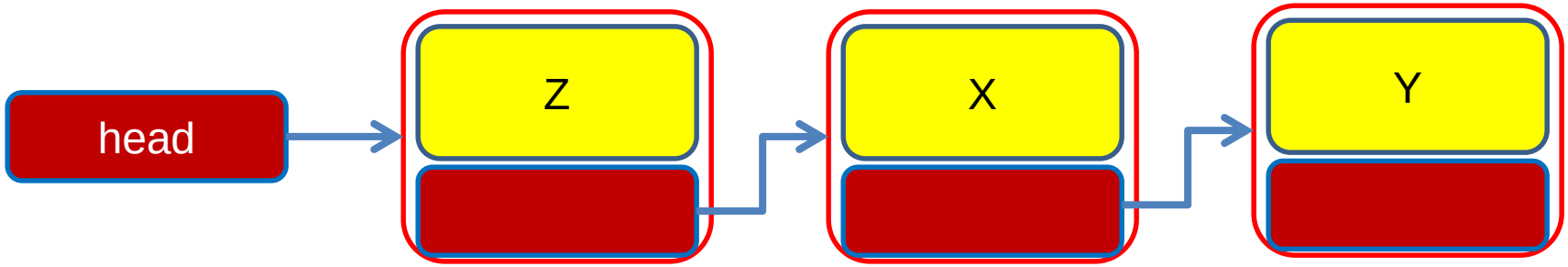
Εναλλακτική υλοποίηση

```
public String toString(){
     String returnString = "";
     for (IntStackElement e = head;
          e != null;
          e = e.getNext()){
        returnString = returnString + e.getValue() + " ";
     }
     return returnString;
}
```

for-loop που δεν διατρέχει ακεραίους

```
class StackExample
{
        public static void main(String[] args){
                Stack s = new Stack();
                s.push(3);
                s.push(2);
                s.push(1);
                System.out.println(s.pop());
                System.out.println(s.pop());
                System.out.println(s.pop());
                System.out.println(s.pop());
        }
}
```

Στοίβα - Υλοποίηση



- Τα X,Y,Z μπορεί να είναι δεδομένα οποιουδήποτε τύπου ή κλάσης. Π.χ. αντί για ακέραιους θα μπορούσαμε να έχουμε αντικείμενα τύπου Person.

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

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

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

```
class PersonStackElement
{
        private Person value;
        private PersonStackElement next;

        public PersonStackElement(Person val){
                value = val;
        }

        public void setNext(PersonStackElement element){
                next = element;
        }

        public PersonStackElement getNext(){
                return next;
        }

        public Person getValue(){
                return value;
        }

}
```

Ο constructor παίρνει σαν όρισμα το αντικείμενο που έχει ήδη δημιουργηθεί

Το αντικείμενο το χειριζόμαστε σαν μια οποιαδήποτε μεταβλητή

```
class Stack
{
        private PersonStackElement head;
        private int size = 0;

        public Person pop(){
                if (size == 0){ // head == null
                        System.out.println("Pop from empty stack");
                        return null;
                }
                int value = head.getValue();
                head = head.getNext();
                size --;
                return value;
        }

        public void push(Person value){
                StackElement element = new StackElement(value);
                element.setNext(head);
                head = element;
                size ++;
        }

}
```

Η pop πλέον επιστρέφει μεταβλητή τύπου Person

Επιστρέφουμε null για να σηματοδοτήσουμε ότι έγινε λάθος (όχι απαραίτητα ο καλύτερος τρόπος να το κάνουμε αυτό)

```
class StackExample
{
	public static void main(String[] args){
		PersonStack stack = new PersonStack();
		Person alice = new Person("Alice", 1);
		stack.push(alice);
		Person bob = new Person("Bob",2);
		stack.push(bob);
		Person charlie = new Person("Charlie",3);
		stack.push(charlie);
		System.out.println(stack.pop());
		System.out.println(stack.pop());
		System.out.println(stack.pop());
	}
}
```

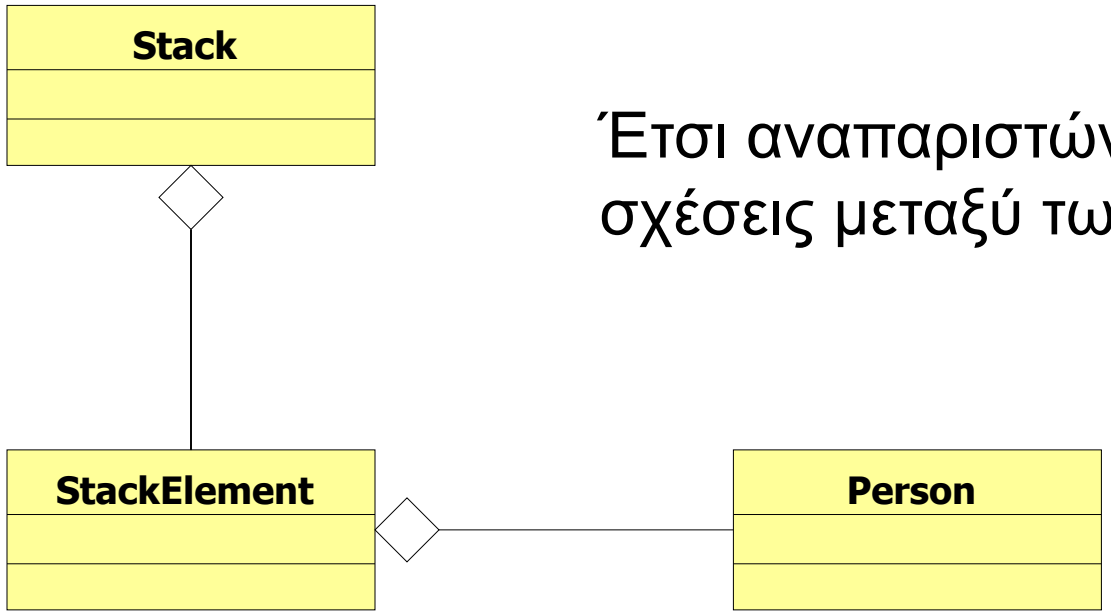
Προσοχή! Αν καλέσουμε άλλη μια φορά την pop θα πάρουμε runtime error γιατί προσπαθούμε να προσπελάσουμε null αναφορά

Σχέσεις μεταξύ κλάσεων

- Στο παράδειγμα με τη στοίβα έχουμε τρείς διαφορετικές κλάσεις (Person, StackElement, Stack) τις οποίες συσχετίζονται μεταξύ τους με διαφορετικούς τρόπους.
- Μπορεί να υπάρχουν πολλές διαφορετικές σχέσεις μεταξύ κλάσεων.
  - Στην περίπτωση μας, η μία κλάση ορίζεται χρησιμοποιώντας αντικείμενα της άλλης
- Αυτού του είδους τη σχέση την λέμε σχέση σύνθεσης
  - Μερικές φορές την ξεχωρίζουμε σε σχέση σύνθεσης (composition) και συνάθροισης (aggregation).

Η UML γλώσσα

- Η UML (Unified Modeling Language) είναι μια γλώσσα για να περιγράφουμε και να καταλαβαίνουμε τον κώδικα μας.
- Τα UML διαγράμματα παρέχουν μια οπτικοποίηση των σχέσεων μεταξύ των κλάσεων.



Έτσι αναπαριστώνται οι σχέσεις μεταξύ των κλάσεων

Σχέσεις κλάσεων

- Όταν έχουμε κλάσεις που έχουν αντικείμενα άλλων κλάσεων ένα θέμα που προκύπτει είναι πότε και πού θα γίνεται η δημιουργία των αντικειμένων και πότε η καταστροφή τους
  - Πιο σημαντικό σε γλώσσες που δεν έχουν garbage collector.
- Π.χ., τα αντικείμενα τύπου StackElement στο προηγούμενο παράδειγμα δημιουργούνται μέσα στην κλάση Stack, και καταστρέφονται μέσα στην Stack, ή αν η Stack καταστραφεί.
  - Αλλαγές σε StackElement αντικείμενα γίνονται μόνο μέσα στην Stack
- Τα αντικείμενα τύπου Person που χρησιμοποιούνται στην StackElement δημιουργούνται εκτός της κλάσης και μπορεί να υπάρχουν αφού καταστραφεί η κλάση.
  - Αλλαγές στα αντικείμενα Person επηρεάζουν και τα περιεχόμενα της Stack και τούμπαλιν.
- Συχνά οι σχέσεις του δεύτερου τύπου λέγονται σχέσεις συνάθροισης, ενώ του πρώτου σχέσεις σύνθεσης.

Σχέση συνάθροισης – Aggregation

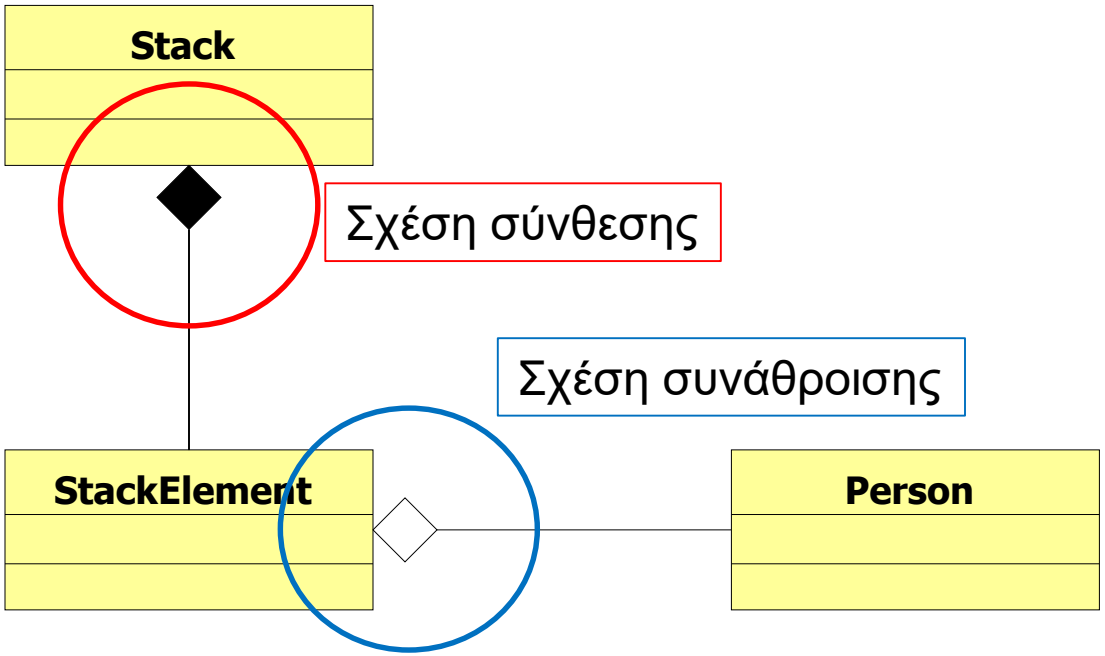
- Η κλάση X έχει σχέση συνάθροισης με την κλάση Y, αν αντικείμενο/α της κλάσης Y ανήκουν στο αντικείμενο της κλάσης X.
  - Τα αντικείμενα της κλάσης Y έχουν υπόσταση και εκτός της κλάσης X.
  - Όταν καταστρέφεται ένα αντικείμενο της κλάσης X δεν καταστρέφονται απαραίτητα και τα αντικείμενα της κλάσης Y.
- Παραδείγματα:
  - Σε έναν άνθρωπο μπορεί να ανήκει ένα αυτοκίνητο, ρούχα, κλπ.
  - Ένα κτήριο μπορεί να έχει μέσα ανθρώπους, έπιπλα, κλπ.
- Στην περίπτωση μας η κλάση StackElement έχει σχέση συνάθροισης με την κλάση Person.

Σχέση σύνθεσης – Composition

- Η κλάση X έχει σχέση σύνθεσης με την κλάση Y, αν το αντικείμενο της κλάσης X αποτελείται από αντικείμενα της κλάσης Y.
  - Τα αντικείμενα της κλάσης Y δεν υπάρχουν εκτός της κλάσης X.
  - Η κλάση X δημιουργεί τα αντικείμενα της κλάσης Y, και καταστρέφονται όταν καταστρέφεται το αντικείμενο της κλάσης X.
- Παραδείγματα:
  - Ένας άνθρωπος αποτελείται από μέρη του σώματος: κεφάλι, πόδια, χέρια κλπ.
  - Ένα κτήριο αποτελείται από τοίχους, δωμάτια, πόρτες, κλπ.
- Στην περίπτωση μας η κλάση Stack έχει σχέση σύνθεσης με την κλάση StackElement.

UML διαγράμματα

- Για να ξεχωρίζουν μεταξύ τους (κάποιες φορές) αναπαριστώνται διαφορετικά στα UML διαγράμματα.



Aggregation and Composition

- Το αν θα είναι μια σχέση, σχέση συνάθροισης ή σύνθεσης εξαρτάται κατά πολύ και από την υλοποίηση μας και τον σχεδιασμό.
  - Π.χ., σε ένα διαφορετικό πρόγραμμα μπορεί να επαναχρησιμοποιούμε το StackElement.
  - Π.χ., σε μία διαφορετική εφαρμογή, τα ανθρώπινα όργανα υπάρχουν και χωρίς τον άνθρωπο.

Προσοχή!

- Ο διαχωρισμός σε σχέσεις συνάθροισης και σύνθεσης είναι ως ένα βαθμό ένας φορμαλισμός.
  - Μην «κολλήσετε» προσπαθώντας να ορίσετε την σχέση.
  - Το σημαντικό είναι όταν δημιουργείτε το πρόγραμμα σας να σκεφτείτε ποιες κλάσεις χρειάζονται τα αντικείμενα που δημιουργούνται και πότε πρέπει να δημιουργηθούν μέσα στον κώδικα, και ποιες κλάσεις επηρεάζονται όταν αλλάζουν.
  - Δεν υπάρχει χρυσός κανόνας. Γενικά το πώς θα σχεδιαστεί το πρόγραμμα είναι κάτι που μπορεί να γίνει με πολλούς τρόπους συνήθως. Διαλέξτε αυτόν που θα κάνει το πρόγραμμα πιο απλό, ευανάγνωστο, εύκολο να επεκταθεί, να ξαναχρησιμοποιηθεί και να διατηρηθεί.