

# ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

---

Πίνακες  
Κλάσεις και Αντικείμενα

# Μαθήματα από το πρώτο εργαστήριο

- Δημιουργία αντικειμένου Scanner
  - `Scanner input = new Scanner(System.in);`
  - Το αντικείμενο `input` είναι η σύνδεση του προγράμματος μας με το **πληκτρολόγιο**.
    - Έχουμε **ένα πληκτρολόγιο** θα δημιουργήσουμε **ένα αντικείμενο Scanner** το οποίο θα χρησιμοποιήσουμε για να διαβάσουμε οτιδήποτε πληκτρολογηθεί.
    - Δεν έχει νόημα να κάνουμε ένα αντικείμενο για κάθε μεταβλητή που διαβάζουμε.
  - Μέθοδοι της Scanner:
    - `next()`: **επιστρέφει το επόμενο String** από την είσοδο (όλοι οι χαρακτήρες από το σημείο που σταμάτησε την προηγούμενη φορά μέχρι να βρει white space: κενό, tab, αλλαγή γραμμής)
    - `nextInt()`: διαβάζει το επόμενο String και το μετατρέπει σε int και **επιστρέφει ένα int αριθμό**.
    - `nextDouble()`: διαβάζει το επόμενο String και το μετατρέπει σε double και **επιστρέφει τον double αριθμό**.
    - `nextLine()`: Διαβάζει ότι υπάρχει μέχρι να βρει **newline** και το επιστρέφει ως String.

# Μαθήματα από το πρώτο εργαστήριο

- Διάβασμα από την είσοδο:
  - Θέλουμε να διαβάσουμε ένα πραγματικό αριθμό ακολουθούμενο από ένα string.

ΣΩΣΤΟ!

```
Scanner in = new Scanner(System.in);  
double d = in.nextDouble();  
String s = in.next();
```

ΛΑΘΟΣ!

```
Scanner in = new Scanner(System.in);  
double d = in.nextDouble();  
String s = in.nextLine();
```

Το `nextLine()` δεν μας κάνει γιατί διαβάζει ότι ακολουθεί τον αριθμό μέχρι να βρει “\n”  
Αν πατήσουμε το `enter` μετά από τον ακέραιο, στην είσοδο μένει το κενό String και το “\n”  
Το `nextLine()` επιστρέφει λοιπόν το κενό String.

# Πίνακες

- Πολλές μεταβλητές του ίδιου τύπου μαζί.
  - `int [] myArray1 = {10,20};`
  - `int myArray2[] = new int[2];`
  - Δημιουργούν δύο πίνακες 2 θέσεων (`length = 2`) που κρατάνε ακέραιους
  - Οι πίνακες ορίζονται με ένα μέγεθος (`length`) και αυτό δεν αλλάζει

# Παράδειγμα με strings και πίνακες

- Φτιάξτε ένα πρόγραμμα που να διαβάζει μία γραμμή από κείμενο και να ψάχνει μία λέξη που δίνουμε σαν όρισμα μέσα σε αυτή τη γραμμή.

## ➤ `java LookFor hello`

- Περιμένει να διαβάσει μια γραμμή από κείμενο και ψάχνει τη λέξη `hello` μέσα στο κείμενο.

```
import java.util.Scanner;
```

```
class LookFor
```

```
{
```

```
    public static void main(String args[])
```

```
    {
```

```
        String name = "default";
```

```
        if (args.length == 1)
```

```
        {
```

```
            name = args[0];
```

```
        }
```

```
        Scanner input = new Scanner(System.in);
```

```
        String line = input.nextLine();
```

```
        String [] words = line.split(" ");
```

```
        for (int i =0; i < words.length; i ++)
```

```
        {
```

```
            if (name.equals(words[i])) {
```

```
                System.out.println(name + " found it at " + i);
```

```
            }
```

```
        }
```

```
    }
```

```
}
```

Τα command-line ορίσματα του προγράμματος αποθηκεύονται στον **πίνακα** από Strings που είναι όρισμα στην main()

Η μέθοδος split της κλάσης String με όρισμα ένα delimiter string σπάει το String με βάση το delimiter και επιστρέφει ένα πίνακα από Strings

Στην περίπτωση αυτή σπάμε το line με βάση το κενό και παίρνουμε τις λέξεις.

# ΔΗΜΙΟΥΡΓΩΝΤΑΣ ΔΙΚΕΣ ΜΑΣ ΚΛΑΣΕΙΣ ΚΑΙ ΑΝΤΙΚΕΙΜΕΝΑ

---

# Hello World

- Θα κάνουμε το ίδιο ακριβώς πρόγραμμα αλλά αυτή τη φορά θέλουμε «**κάποιος**» να πει το hello world.
  - Θέλουμε μια οντότητα που να μπορεί να πει κάτι
- Πως θα το κάνουμε?
  - Θα ορίσουμε μια **κλάση Person**.
  - Τα **αντικείμενα** αυτής της κλάσης θα μπορούν να **μιλήσουν**



# Hello World Revisited

```
class Person
```

Ορισμός κλάσης

```
{
```

```
    private String name = "Alice";
```

Ορισμός  
(και αρχικοποίηση) πεδίου

```
    public void sayHello()
```

Ορισμός μεθόδου

```
{
```

```
        System.out.println(name+": Hello World");
```

```
}
```

Χρήση πεδίου

```
}
```

```
class HelloWorldRevisited
```

```
{
```

```
    public static void main(String[] args)
```

```
{
```

```
        Person someone = new Person();
```

Ορισμός αντικειμένου

```
        someone.sayHello();
```

Κλήση μεθόδου

```
}
```

```
}
```

# Κλάσεις και αντικείμενα

- Ορισμός κλάσης:

```
class <Όνομα Κλάσης>
{
    <Ορισμός πεδίων κλάσης>

    <Ορισμός μεθόδων κλάσης>
}
```

- Ορισμός αντικειμένου:

```
[ <Όνομα Κλάσης> myObject = new <Όνομα Κλάσης> (); ]
```

- Ο ορισμός του αντικειμένου γίνεται συνήθως μέσα στη main ή μέσα στη μέθοδο μίας άλλης κλάσης που χρησιμοποιεί το αντικείμενο

# Τα keywords Public/Private

- Ότι είναι ορισμένο ως **public** σε μία κλάση **είναι προσβάσιμο** από μία άλλη κλάση που ορίζει ένα αντικείμενο τύπου Person
  - Π.χ., η μέθοδος **sayHello()** **είναι προσβάσιμη** από την κλάση **HelloWorldRevisited** μέσω του αντικειμένου **someone**.
- Ότι είναι ορισμένο ως **private** σε μία κλάση **δεν είναι προσβάσιμο** από μία άλλη κλάση
  - Π.χ., το πεδίο **name** **δεν είναι προσβάσιμο** από την κλάση **HelloWorldRevisited** μέσω του αντικειμένου **someone**.
- Μπορούμε να έχουμε public και private πεδία και μεθόδους.
  - Κανόνας: Τα **πεδία** τα ορίζουμε (σχεδόν) **ΠΑΝΤΑ private**.
  - Οι κλάσεις που χρειάζονται να καλούνται από **αντικείμενα** είναι **public** αυτές που είναι **βοηθητικές** είναι **private**.
- Τα πεδία και οι μέθοδοι μίας κλάσης, ανεξάρτητα αν είναι public ή private, είναι **προσβάσιμα** από όλες τις μεθόδους και τα αντικείμενα **της ίδιας κλάσης**
  - Π.χ., το πεδίο **name** είναι προσβάσιμο παντού μέσα στην κλάση **Person**.

# Παράδειγμα

- Θέλουμε ένα πρόγραμμα που να προσομοιώνει την κίνηση ενός αυτοκινήτου, το οποίο κινείται και τυπώνει τη θέση του.

# MovingCar

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

    public void move(){
        position += 1;
    }

    public void printPosition(){
        System.out.println("Car at position "+position);
    }
}

class MovingCar
{
    public static void main(String args[]){
        Car myCar = new Car();
        myCar.move();
        myCar.printPosition();
    }
}
```

# Μέθοδοι

- Οι μέθοδοι που έχουμε δει μέχρι τώρα είναι πολύ απλές
  - Δεν έχουν **παραμέτρους** (δεν παίρνουν **ορίσματα**)
  - Δεν **επιστρέφουν τιμή**

**void**: δεν επιστρέφει τιμή

Δεν παίρνει ορίσματα

```
public void move()  
{  
    position += 1;  
}
```

## Παράδειγμα 2

- Εκτός από την κίνηση κατά μία θέση θέλουμε να μπορούμε να κινούμε το όχημα όσες θέσεις θέλουμε είτε προς τα δεξιά (+) είτε προς τα αριστερά (-). Θα τυπώνεται η θέση σε κάθε κίνηση.

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

    public void move(){
        position += 1;
    }

    public void moveManySteps(int steps)
```

Παράμετρος της μεθόδου

```
    {
        int delta = 1;
```

```
        if (steps < 0){
            steps = -steps; delta = -1;
        }
        for (int i = 0; i < steps; i ++){
            position += delta;
            System.out.println("Car at position "+position);
        }
    }
```

Το πέρασμα των παραμέτρων γίνεται κατά τιμή (pass by value)

Τοπική μεταβλητή της μεθόδου

Η παράμετρος λειτουργεί ως τοπική μεταβλητή της συνάρτησης και χάνεται μετά την κλήση της μεθόδου. Η τιμή του ορίσματος δεν μεταβάλλεται

```
    public void printPosition(){
        System.out.println("Car at position "+position);
    }
}
```

```
class MovingCar2
{
    public static void main(String args[]){
        Car myCar = new Car();
        int steps = -10;
        myCar.moveManySteps(steps);
        System.out.println("--: " + steps);
    }
}
```

Όρισμα της μεθόδου

Τυπώνει --: -10



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

    public void move(){
        position += 1;
    }

    public void moveManySteps(int steps)
    {
        int delta = 1;
        if (steps < 0){
            steps = -steps; delta = -1;
        }
        for (int i = 0; i < steps; i ++){
            position += delta;
            printPosition();
        }
    }

    public void printPosition(){
        System.out.println("Car at position "+position);
    }
}

class MovingCar2
{
    public static void main(String args[]){
        Car myCar = new Car();
        int steps = -10;
        myCar.moveManySteps(steps);
        System.out.println("--: " + steps);
    }
}
```

Μπορούμε να κάνουμε την εκτύπωση καλώντας την printPosition()

# Τοπικές μεταβλητές

- Είδαμε πρώτη φορά τις **τοπικές μεταβλητές** όταν μιλήσαμε για μεταβλητές που ορίζονται μέσα σε ένα λογικό block.
  - Παρόμοια είναι και για τις μεταβλητές μιας **μεθόδου**.
- Τοπικές μεταβλητές μιας μεθόδου είναι οι μεταβλητές που ορίζονται **μέσα** στον κώδικα της μεθόδου
  - Περιλαμβάνουν και τις μεταβλητές που κρατάνε τις **παραμέτρους** της μεθόδου
- Οι μεταβλητές αυτές έχουν **εμβέλεια** μόνο **μέσα στην μέθοδο**
  - **Εξαφανίζονται** όταν **βγούμε** από τη μέθοδο.
- Αντιθέτως τα **πεδία** της κλάσης διατηρούνται όσο υπάρχει το **αντικείμενο**, και έχουν εμβέλεια σε **όλη** την κλάση

# Μέθοδοι που επιστρέφουν τιμές

- Μέχρι τώρα οι μέθοδοι που φτιάξαμε δεν επιστρέφουν τιμή
  - Είναι τύπου `void`.
- Σε πολλές περιπτώσεις θέλουμε η μέθοδος να μας **επιστρέφει τιμή**
  - Π.χ., μία μέθοδος που υπολογίζει το άθροισμα δύο αριθμών

# Παράδειγμα

- Το αυτοκίνητο μας δεν μπορεί να μετακινηθεί έξω από το διάστημα  $[-10, 10]$ . Θέλουμε η `move()` να μας επιστρέφει μια λογική τιμή αν η μετακίνηση έγινε η όχι.

```
import java.util.Scanner;

class Car
{
    private int position = 0;

    public boolean moveManySteps(int steps)
    {
        if ((position + steps < -10) || (position + steps > 10)){
            return false;
        }else{
            position += steps;
            return true;
        }
    }

    public void printPosition(){
        System.out.println("Car at position "+position);
    }
}

class MovingCar3
{
    public static void main(String args[]){
        Scanner input = new Scanner(System.in);
        Car myCar = new Car();
        int steps = input.nextInt();
        boolean carMoved = myCar.moveManySteps(steps);
        if (carMoved)
            myCar.printPosition();
        else
            System.out.println("Car could not move");
    }
}
```

# Η εντολή return

- Η εντολή **return** χρησιμοποιείται για να επιστρέψει μια τιμή μια μέθοδος.
- ΣΥΝΤΑΚΤΙΚΟ:
  - **return** <έκφραση>
- Αν έχουμε μια συνάρτηση που επιστρέφει τιμή τύπου **T**
  - Π.χ. **public double division(int x, int y)**
- η έκφραση στο return πρέπει να επιστρέφει μία τιμή τύπου **T**. (π.χ., **return x / (double) y**)
- **Κάθε μονοπάτι** εκτέλεσης του κώδικα θα πρέπει να επιστρέφει μια τιμή.
  - Η κλήση της return σε οποιοδήποτε σημείο του κώδικα **σταματάει την εκτέλεση** της μεθόδου και επιστρέφει τιμή.

```
import java.util.Scanner;

class Car
{
    private int position = 0;

    public boolean moveManySteps(int steps)
    {
        if ((position + steps < -10) || (position + steps > 10)){
            return false;
        }
        position += steps;
        return true;
    }

    public void printPosition(){
        System.out.println("Car at position "+position);
    }
}

class MovingCar3
{
    public static void main(String args[]){
        Scanner input = new Scanner(System.in);
        Car myCar = new Car();
        int steps = input.nextInt();
        boolean carMoved = myCar.moveManySteps(steps);
        if (carMoved)
            myCar.printPosition();
        else
            System.out.println("Car could not move");
    }
}
```

# Η εντολή return

- Μπορούμε να καλέσουμε την **return** και σε μία **void** μέθοδο
  - Χωρίς επιστρεφόμενη τιμή.
    - **return;**
  - Σταματάει την εκτέλεση της μεθόδου

```
public void printIfPositive()  
{  
    if (position < 0) {  
        return;  
    }  
    System.out.println("position = " + position);  
}
```



```

import java.util.Scanner;

class Car
{
    private int position = 0;

    public boolean moveManySteps(int steps)
    {
        if ((position + steps < -10) || (position + steps > 10)){
            return false;
        }
        position += steps;
        return true;
    }

    public void printPosition(){
        System.out.println("Car at position "+position);
    }
}

class MovingCar3
{
    public static void main(String args[]){
        Scanner input = new Scanner(System.in);
        Car myCar = new Car();
        int steps = input.nextInt();
        myCar.moveManySteps(steps);
        myCar.printPosition();
    }
}

```

Η moveManySteps επιστρέφει τιμή,  
αλλά η κλήση της την αγνοεί

Η printPosition θα επιστρέψει 0 αν δεν  
κινήθηκε το όχημα