ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Δημιουργία Κλάσεων και Αντικειμένων

Κλάση

- Μια κλάση είναι μία αφηρημένη περιγραφή αντικειμένων με κοινά χαρακτηριστικά και κοινή συμπεριφορά.
  - Ένα καλούπι/πρότυπο που παράγει αντικείμενα
- Ένα αντικείμενο είναι ένα στιγμιότυπο μίας κλάσης.
- Η κλάση ορίζει τον τύπο του αντικειμένου.
  - Τα χαρακτηριστικά του αντικειμένου
  - Τις ενέργειες που μπορεί να επιτελέσει.

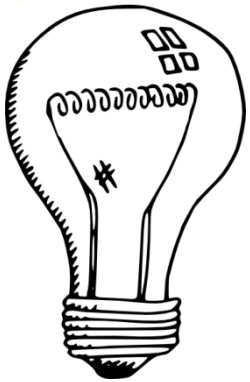
Πρακτικά στον κώδικα

- Μία κλάση Κ ορίζεται από
  - Κάποιες μεταβλητές τις οποίες ονομάζουμε πεδία
  - Κάποιες συναρτήσεις που τις ονομάζουμε μεθόδους.
    - Οι μέθοδοι «βλέπουν» τα πεδία της κλάσης

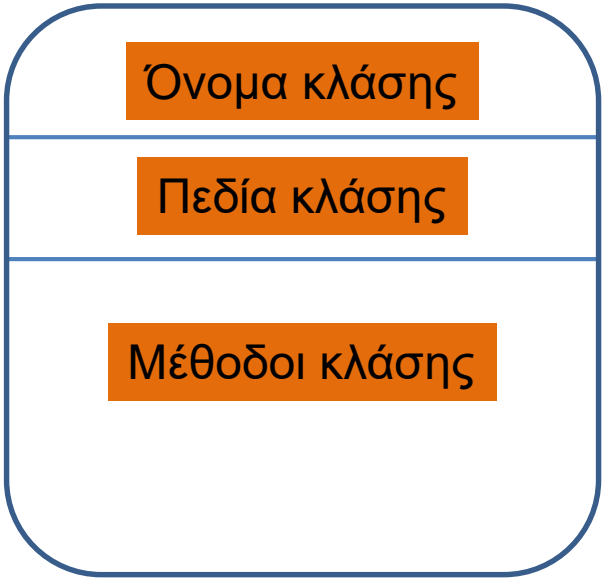
μέλη της κλάσης

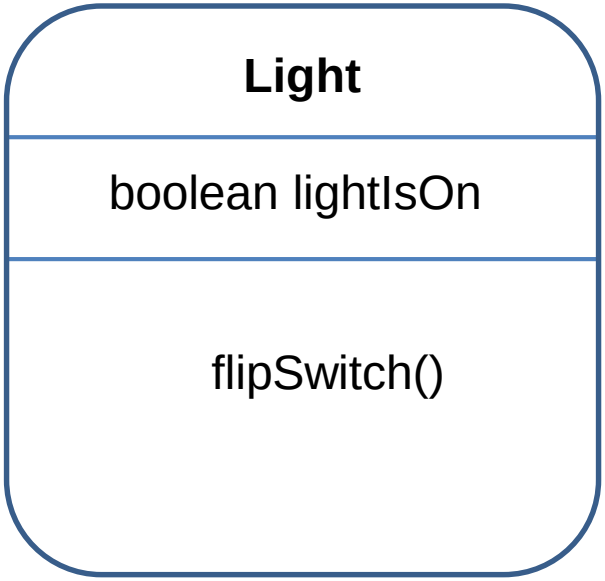
- Ένα αντικείμενο ορίζεται ως μια μεταβλητή τύπου Κ
  - Το αντικείμενο έχει συγκεκριμένες τιμές στα πεδία.
  - Στο πρόγραμμα έχουμε (συνήθως) πρόσβαση μόνο τις μεθόδους.
    - Μέσω των μεθόδων έχουμε πρόσβαση στα πεδία
  - Αν υπάρχουν κάποια πεδία στα οποία έχουμε πρόσβαση αυτά τα λέμε properties.

Δημιουργώντας φως



Θα φτιάξουμε μια κλάση που θα χειρίζεται ένα διακόπτη φωτός. Το φώς είναι είτε ανοιχτό είτε κλειστό και μπορούμε να ανοιγοκλείνουμε το φως





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

Ορισμός μελών κλάσης

- Ορισμός πεδίων
  - Τα πεδία είναι μεταβλητές και ορίζονται όπως οι υπόλοιπες μεταβλητές
    - Η μόνη διαφορά είναι ότι τα πεδία πρέπει να τα προσδιορίσουμε ως private ή public. Τα πεδία τα ορίζουμε πάντα private.

```
[private/public] <τυπος> <όνομα μεταβλητής> [ = τιμή];
```

- Ορισμός μεθόδων
  - Οι μέθοδοι είναι συναρτήσεις και έχουν το εξής ορισμό

```
[public/private] <τύπος> <όνομα συνάρτησης>([παράμετροι])
{
      <κώδικας συνάρτησης>
}
```

Light

```
class Light
{
        private boolean lightIsOn = false;

        public void flipSwitch()
        {
                lightIsOn = !lightIsOn;
        }
}

class HouseWithLights
{
        public static void main(String[] args)
        {
                Light bedroomLight = new Light();
                bedroomLight.flipSwitch();
        }
}
```

Ορισμός κλάσης

Ορισμός (και αρχικοποίηση) πεδίου

Ορισμός μεθόδου τύπου void χωρίς παραμέτρους

Χρήση πεδίου

Ορισμός αντικειμένου

Κλήση μεθόδου

Τα keywords Public/Private

- Ότι είναι ορισμένο ως public σε μία κλάση είναι προσβάσιμο από μία άλλη κλάση που ορίζει ένα αντικείμενο τύπου Light
  - Π.χ., η μέθοδος flipSwitch() είναι προσβάσιμη από την κλάση HouseWithLights μέσω του αντικειμένου bedroomLight.
- Ότι είναι ορισμένο ως private σε μία κλάση δεν είναι προσβάσιμο από μία άλλη κλάση
  - Π.χ., το πεδίο lightIsOn δεν είναι προσβάσιμο από την κλάση HouseWithLights μέσω του αντικειμένου bedroomLight.
- Μπορούμε να έχουμε public και private πεδία και μεθόδους.
  - Κανόνας: Τα πεδία τα ορίζουμε (σχεδόν) **ΠΑΝΤΑ** private.
  - Οι μέθοδοι 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 lightIsOn είναι προσβάσιμο παντού μέσα στην κλάση Light, και σε οποιοδήποτε άλλο αντικείμενο τύπου Light
  - Κάποιοι για να ξεχωρίζουν τα πεδία από άλλες μεταβλητές βάζουν '_' στην αρχή του ονόματος των πεδίων. Π.χ., _lightIsOn

```
class Light
{
        private boolean lightIsOn = false;

        public void flipSwitch(){
                lightIsOn = !lightIsOn;
        }

        public void printState(){
                if (lightIsOn){
                        System.out.println("The light is on");
                }else {
                        System.out.println("The light is off");
                }
        }
}
```

Η κατάσταση ενός αντικειμένου προσδιορίζεται από τις τιμές που έχουν τα πεδία της κλάσης

```
class HouseWithLights
{
        public static void main(String[] args){
                Light bedroomLight = new Light();
                bedroomLight.flipSwitch();
                bedroomLight.printState();
                Light kitchenLight = new Light();
                kitchenLight.flipSwitch();
                kitchenLight.printState();
        }
}
```

Η μόνη πρόσβαση που έχουμε στην κατάσταση του αντικειμένου είναι μέσω των μεθόδων της κλάσης.

Dimmer

- Θέλουμε ο διακόπτης μας να μας δίνει την δυνατότητα να αυξομειώνουμε την ένταση.
  - Τι επιπλέον πεδία πρέπει να προσθέσουμε?
  - Τι επιπλέον μεθόδους χρειαζόμαστε?

```
class DimmerLight
{
    private boolean lightIsOn = false;
    private int intensity = 100;

    public void flipSwitch(){
        lightIsOn = !lightIsOn;
    }

    public void dim(){
        if (intensity > 0){
            intensity --;
        }
    }

    public void birghten(){
        if (intensity < 100){
            intensity ++;
        }
    }

    public void printState(){
        if (lightIsOn){
            System.out.println("The light is ON with intensity " + intensity);
        }else{
            System.out.println("The light is OFF");
        }
    }
}
```

```
class HouseWithDimmerLights
{
    public static void main(String[] args){
        DimmerLight bedroomLight =
                                new DimmerLight();
        bedroomLight.flipSwitch();
        bedroomLight.dim();
        bedroomLight.printState();
    }
}
```

Dimmer

- Κάθε φορά που αυξάνουμε ή μειώνουμε την ένταση θέλουμε να μας λέει και την κατανάλωση
  - (Κατανάλωση = ένταση * 0.1 λεπτά/ώρα)

```
class DimmerLight2
{
    private boolean lightIsOn = false;
    private int intensity = 100;

    public void flipSwitch(){
        lightIsOn = !lightIsOn;
    }

    public void dim(){
        if (intensity > 0){
            intensity --;
        }
        double consumption = intensity *0.1;
        System.out.print("Consumption = "+consumption);
    }

    public void birghten(){
        if (intensity < 100){
            intensity ++;
        }
        double consumption = intensity *0.1;
        System.out.print("Consumption = "+consumption);
    }

    public void printState(){
        if (lightIsOn){
            System.out.println("The light is ON with intensity " + intensity);
        }else{
            System.out.println("The light is OFF");
        }
    }
}
```

Οι μεταβλητές consumption είναι τοπικές μεταβλητές

Υπάρχουν μόνο μέσα στις μεθόδους dim και brighten και όταν τελειώσει η κλήση τους εξαφανίζονται.

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

Παράδειγμα

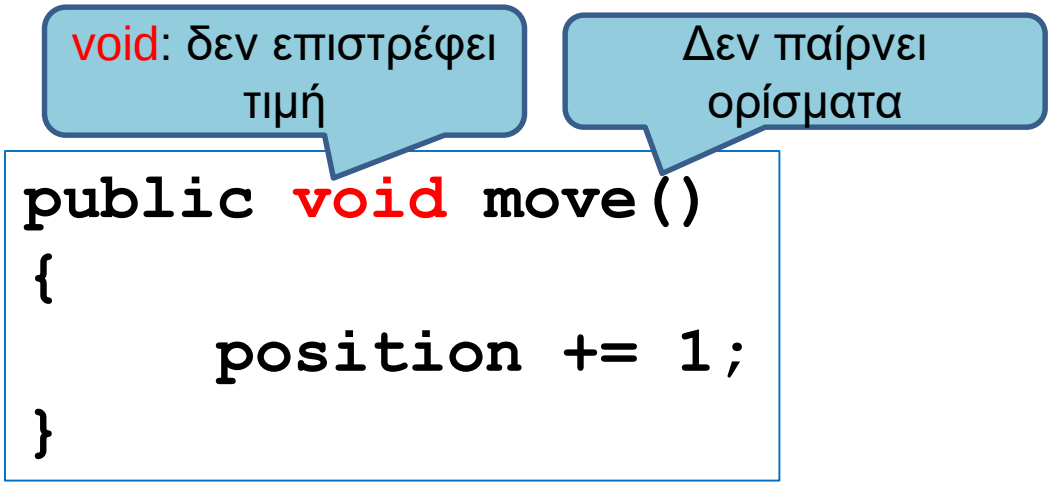
- Θέλουμε ένα πρόγραμμα που να προσομοιώνει την κίνηση ενός αυτοκινήτου, το οποίο κινείται και τυπώνει τη θέση του.

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



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 (-).

Παράμετροι

- Οι μέθοδοι μπορούν να έχουν παραμέτρους
  - Μας επιτρέπουν να περάσουμε τιμές στην μέθοδο μας

```
public void moveManySteps(int steps)
{
        position += steps;
}
```

Ορισμός παραμέτρου

- Μία παράμετρος ορίζεται όπως οποιαδήποτε άλλη μεταβλητή.
  - Πρέπει να έχει συγκεκριμένο τύπο και όνομα
  - Είναι τοπική μεταβλητή της μεθόδου

```
int x = 10;
myCar.moveManySteps(x);
myCar.moveManySteps(10);
```

Όρισμα στην κλήση της μεθόδου

- Όταν καλούμε την μέθοδο, περνάμε το όρισμα
  - Το όρισμα είναι μια έκφραση (κάτι που θα μπορούσε να είναι στο δεξί μέρος μιας ανάθεσης)
  - Θα πρέπει να συμφωνεί στον τύπο με την παράμετρο
  - Είναι σαν να κάνουμε ανάθεση `steps = x` ή `steps = 10`

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

    public void moveManySteps(int steps)
    {
        position += steps;
    }
}

class MovingCar2
{
    public static void main(String args[])
    {
        Car myCar = new Car();
        int x = 10;
        myCar.moveManySteps(x);
        myCar.moveManySteps(10);
        myCar.moveManySteps(2*x+10);
    }
}
```

Στον ορισμό της μεθόδου ορίζουμε και την παράμετρο της μεθόδου, όπως ορίζουμε μια μεταβλητή. Έχει ένα τύπο και ένα όνομα

Όταν καλούμε την μέθοδο περνάμε μια τιμή σαν όρισμα στην μέθοδο
Σαν όρισμα μπορεί να είναι μια οποιαδήποτε έκφραση.
Αρκεί ή αποτίμηση της έκφρασης να έχει τύπο συμβατό με αυτόν της παραμέτρου (int στην περίπτωση μας)

Κατά την κλήση της μεθόδου ουσιαστικά εκχωρείται η τιμή της έκφρασης στην μεταβλητή steps. Αυτό λέγεται και πέρασμα παραμέτρου.

Πέρασμα παραμέτρων

- Όταν καλούμε μια μέθοδο με μία τιμή σαν όρισμα, ουσιαστικά εκχωρούμε αυτή την τιμή στην παράμετρο της μεθόδου

Η κλήση

```
myCar.moveManySteps(2*x+10);
```

όπου η μεταβλητή x έχει την τιμή 10

Αποτιμάται η τιμή της έκφρασης και εκχωρείται

Ισοδυναμεί με τον κώδικα:

```
{
        int steps = 30;
        position += delta;
}
```

Το πέρασμα μεταβλητών με αυτό τον τρόπο λέγεται πέρασμα δια τιμής (pass by value). Η μέθοδος δεν έχει πρόσβαση στην μεταβλητή μόνο στην τιμή

Πέρασμα παραμέτρων δια τιμής

- Όταν το πέρασμα παραμέτρων γίνεται δια τιμής, το πρόγραμμα μας έχει πρόσβαση μόνο στην τιμή της παραμέτρου και όχι στην μεταβλητή που χρησιμοποιήσαμε στο όρισμα.
  - Σε όλες τις γλώσσες πλέον το πέρασμα παραμέτρων γίνεται δια τιμής

- Αν η παράμετρος είναι ένα αντικείμενο τα πράγματα γίνονται πιο σύνθετα
  - Η τιμή της μεταβλητής που έχουμε σαν παράμετρο είναι διεύθυνση μνήμης. Δεν μπορούμε να αλλάξουμε την διεύθυνση μνήμης αλλά μπορούμε να αλλάξουμε τα περιεχόμενα της.

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

    public void moveManySteps(int steps, String direction)
    {
       if (direction.equals("right"){ position += steps;}
       if (direction.equals("left") { position -= steps;}
    }
}

class MovingCar3
{
    public static void main(String args[]){
       Car myCar = new Car();
       myCar.moveManySteps(10, "left");
    }
}
```

Μέθοδος με πολλές παραμέτρους

Τα ορίσματα θα πρέπει να συμφωνούν με το πλήθος και τους τύπους των παραμέτρων στην αντίστοιχη θέση

Κλήση της μεθόδου

Τύποι παραμέτρων και ορισμάτων

- Οι παράμετροι μιας μεθόδου έχουν συγκεκριμένο τύπο
- Τα ορίσματα στην κλήση της μεθόδου θα πρέπει να συμφωνούν με τον τύπο της παραμέτρου, θέση προς θέση.
- Ισχύουν οι μετατροπές τύπου που ξέρουμε
  - byte → short → int → long → float → double

- Μία μέθοδος μπορεί να πάρει ως όρισμα και ένα αντικείμενο μιας κλάσης.
  - Το πώς δουλεύει αυτό θα το μάθουμε όταν μιλήσουμε για αναφορές.

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

Η εντολή return

- Η εντολή return χρησιμοποιείται για να επιστρέψει μια τιμή μια μέθοδος.
- Συντακτικό:
  - `return <έκφραση>`
- Ο τύπος της έκφρασης στην εντολή return θα πρέπει να είναι ίδιος (ή συμβατός) με τον τύπο της μεθόδου.
- 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.
  - Μπορούμε να το χρησιμοποιήσουμε αυτό για να απλοποιήσουμε τον κώδικα.

Παράδειγμα 3

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

```
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;
       }
    }
}
```

Όταν ορίζουμε μια μέθοδο που επιστρέφει τιμή θα πρέπει να ορίσουμε τον τύπο της τιμής που επιστρέφει.

Π.χ. αυτή η μέθοδος επιστρέφει τιμή boolean

Μια μέθοδος μπορεί να επιστρέφει και ένα αντικείμενο μιας κλάσης

Επιστρέφουμε μια τιμή μέσα στον κώδικα χρησιμοποιώντας την εντολή return.

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

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

Αν μπούμε μέσα στο if η return θα σταματήσει την εκτέλεση του κώδικα και θα μας βγάλει από την μέθοδο. Επιστρέφεται η τιμή false.

Δεν χρειάζεται πλέον το else

Ο τύπος μιας μεθόδου

- Μια μέθοδος που επιστρέφει τιμή ορίζεται με συγκεκριμένο τύπο. Π.χ.
  - `public boolean moveManySteps(int steps)`
  - `public double division(int x, int y)`
  - `public String getUserName()`
  - `public Car getCar()`

- 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 (συμβατού με το) T. (π.χ., `return x/(double)y`)

```
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 MovingCar4b{
    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");}
    }
}
```

Κλήση της μεθόδου

```
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 MovingCar4b{
    public static void main(String args[]){
        Scanner input = new Scanner(System.in);
        Car myCar = new Car();
        int steps = input.nextInt();
        if (myCar.moveManySteps(steps)) {
                myCar.printPosition();
        }
        else { System.out.println("Car could not move");}
    }
}
```

Κλήση της μεθόδου και χρήση του αποτελέσματος απευθείας μέσα στην συνθήκη. Δεν χρειάζεται να το αποθηκεύσουμε.

```
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 MovingCar4c
{
    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();
    }
}
```

Δεν είναι υποχρεωτικό να χρησιμοποιούμε πάντα την επιστρεφόμενη τιμή

Η moveManySteps επιστρέφει τιμή, αλλά η κλήση της την αγνοεί

Η printPosition θα επιστρέψει 0 αν δεν κινήθηκε το όχημα

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);
}
```

Η εντολή return

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

```
public void moveManySteps(int steps, String direction)
{
        if (steps < 0){
                return;
        }
        if (direction.equals("right"){ position += steps;}
        if (direction.equals("left") { position -= steps;}
}
```