

ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Κλάσεις και Αντικείμενα
Μέθοδοι

Παράδειγμα 1

- 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);  
    }  
}
```

Ορισμός κλάσης

Ορισμός (και αρχικοποίηση) πεδίου

Ορισμός μεθόδου

Χρήση πεδίου

```
class MovingCar
```

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

Ορισμός αντικειμένου

Κλήση μεθόδου

Παράδειγμα 2

- 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 moveManySteps(int steps)
    {
        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);
        }
    }

    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);
    }
}
```

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

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

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

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.

Παράδειγμα 3

- 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.

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");}
    }
}
```

Παράδειγμα 4

- Όταν καλούμε την συνάρτηση `move()` το όχημα μας θα κινείται ένα **τυχαίο αριθμό** από βήματα στο διάστημα $(-3,3)$

Υλοποίηση

- Θα φτιάξουμε μια **βοηθητική συνάρτηση** που θα μας **επιστρέφει** τον τυχαίο αριθμό από βήματα.

private: δεν χρειάζεται να φαίνεται έξω από την κλάση

Ο τύπος της **επιστρεφόμενης** τιμής

```
private int computeRandomSteps ()  
{  
    int radomSteps;  
    // do the computation  
  
    return randomSteps;  
}  
  
public void move() {  
    int steps = computeRandomSteps ();  
    moveManySteps (steps) ;  
}
```

Κλήση της συνάρτησης και χρήση της επιστρεφόμενης τιμής

```
import java.util.Random;

class Car
{
    private int MAX_VALUE = 3;
    private int position = 0;
    private Random randomGenerator = new Random();

    private int computeRandomSteps ()
    {
        int randomSteps = randomGenerator.nextInt(2*MAX_VALUE + 1) - MAX_VALUE;
        return randomSteps;
    }

    public void move(){
        int steps = computeRandomSteps ();
        moveManySteps(steps);
    }

    public void moveManySteps(int steps) { ... }

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

class MovingCar4
{
    public static void main(String args[]){
        Car myCar = new Car();
        myCar.move();
    }
}
```

Public/Private

- Ότι είναι ορισμένο ως **public** σε μία κλάση είναι προσβάσιμο από **οποιονδήποτε**.
 - Μπορούμε να καλέσουμε τις μεθόδους ορίζοντας ένα αντικείμενο της κλάσης
- Ότι είναι ορισμένο ως **private** σε μία κλάση είναι προσβάσιμο **μόνο** από την **ίδια κλάση**.
- Ο τροποποιητής **private** μας επιτρέπει την **απόκρυψη πληροφοριών** (**information hiding**).
 - Ο χρήστης της κλάσης **Car**, δεν χρειάζεται να ξέρει πως υλοποιείται η μέθοδος **computeRandomSteps** που υπολογίζει τον τυχαίο αριθμό των βημάτων.
 - Αν αποφασίσουμε να αλλάξουμε κάτι στη μέθοδο αυτό θα γίνει ως μέρος του επανασχεδιασμού της κλάσης **Car**. Κανείς άλλος δεν θα πρέπει να επηρεαστεί από την αλλαγή στον κώδικα.
- Τα **πεδία** μιας κλάσης τα ορίζουμε **πάντα private**.

Ενθυλάκωση

- Η ομαδοποίηση λογισμικού και δεδομένων σε μία οντότητα (κλάση και αντικείμενα της κλάσης) ώστε να είναι εύχρηστη μέσω ενός καλά ορισμένου **interface**, ενώ οι λεπτομέρειες υλοποίησης είναι κρυμμένες από τον χρήστη.
- **API** (Application Programming Interface)[‘Ει-Πι-Άι]
 - Μια περιγραφή για το πώς χρησιμοποιείται η κλάση μέσω των **public μεθόδων** της.
 - Java docs είναι ένα παράδειγμα.
 - Το API είναι αρκετό για να χρησιμοποιήσετε μια κλάση, δεν χρειάζεται να ξέρετε την υλοποίηση των μεθόδων.
- **ADT** (Abstract Data Type)
 - Ένας τύπος δεδομένων που ορίζεται χρησιμοποιώντας την αρχή της ενθυλάκωσης
 - Οι λίστες που χρησιμοποιήσατε στην Python είναι ένα παράδειγμα.
 - Δεδομένα και μέθοδοι.

Accessor and Mutator methods

- Πολλές φορές χρειαζόμαστε να **διαβάσουμε** ή να **αλλάξουμε** ένα πεδίο ενός αντικειμένου
 - Π.χ., να διαβάσουμε τη θέση του οχήματος, ή να τοποθετήσουμε το όχημα σε μια συγκεκριμένη θέση.
 - Πως θα το κάνουμε αφού τα πεδία είναι private?
- Ορίζουμε ειδικές μεθόδους
 - **Μέθοδος προσπέλασης** (**accessor** method) για διάβασμα
 - **Μέθοδος μεταλλαγής** (**mutator** method) για γράψιμο
- **Σύμβαση**: Στη Java η ονοματολογία των μεθόδων αυτών γίνεται με συγκεκριμένο τρόπο:
 - **get<ονομα μεταβλητης>** για την πρόσβαση
 - getPosition
 - **set<ονομα μεταβλητης>** για την μετάλλαξη
 - setPosition


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

    public void setPosition(int p){
        position = p;
    }

    public int getPosition(){
        return position;
    }
}
```

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

```
class MovingCar5
{
    public static void main(String args[]){
        Car myCar = new Car();
        myCar.setPosition(10);
        myCar.move();
        System.out.println(myCar.getPosition());
    }
}
```

Υπάρχουν περιπτώσεις που μπορεί να θέλουμε η συνάρτηση set να επιστρέφει **boolean** (true αν η ανάθεση έγινε επιτυχώς, false αλλιώς)

```

class Car
{
    private int position = 0;

    public void setPosition(int position) {
        this.position = position;
    }

    public int getPosition() {
        return position;
    }

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

class MovingCar5
{
    public static void main(String args[]) {
        Car myCar = new Car();
        myCar.setPosition(10);
        myCar.move();
        System.out.println(myCar.getPosition());
    }
}

```

Το **this.position** αναφέρεται στο πεδίο του αντικείμενου.
Το **position** αναφέρεται στην παράμετρο της συνάρτησης

Η κρυφή παράμετρος **this** προσδιορίζει το αντικείμενο που κάλεσε την μέθοδο

Έτσι μπορούμε να χρησιμοποιήσουμε το ίδιο όνομα μεταβλητής χωρίς να δημιουργείται σύγχυση

An encapsulated class

Implementation details hidden in the capsule:

Private instance variables
Private constants
Private methods
Bodies of public and private method definitions

Interface available to a programmer using the class:

Comments
Headings of public accessor, mutator, and other methods
Public defined constants

Programmer who uses the class

A class definition should have no public instance variables.

Παράδειγμα

- Μία κλάση που να αποθηκεύει ημερομηνίες
 - Η κλάση θα παίρνει την ημέρα, μήνα και χρόνο σαν νούμερα (π.χ., 13 3 2014) και θα μπορεί να τυπώνει την ημερομηνία με το όνομα του μήνα (π.χ., 13 Μαρτίου 2014)
 - Στο πρόγραμμα βάλετε μια ημερομηνία και τυπώστε την.