Φροντιστήριο JAVA

Οντοκεντρικός Προγραμματισμός (lect 1)

Ελένη Βογιατζάκη
evoyiatzaki@ceid.upatras.gr

Διαδικαστικά Μαθήματος

Θεωρία – Φροντιστήριο - Εργαστήριο

ΕΒΔΟΜΑΔΙΑΙΟ ΠΡΟΓΡΑΜΜΑ

- Θεωρία
  - Πέμπτη 9.00-12.00 (ΒΑ)
- Φροντιστήριο
  - Τετάρτη 12:00-14:00 (ΒΑ)

- Εργαστήριο (έναρξη: εβδομάδα 2/3/20)

Δευτέρα	Τρίτη	Τρίτη	Πέμπτη	Πέμπτη
16.00-18.00	09.00-11.00	11.00-13.00	12.00-14.00	16.00-18.00

Πρέπει να έχετε εγγραφεί σε μια από τις ομάδες στο eclass

ΕΡΓΑΣΤΗΡΙΟ

- 3 εργαστήρια (2 Java, 1 C++), 3 εβδομάδες το καθένα
- Κάθε εργαστήριο περιλαμβάνει:
  - Ένα σετ εργαστηριακών ασκήσεων εκτέλεσης (από φυλλάδιο)
  - Ένα σετ ασκήσεων σχεδίασης και εκτέλεσης
- Οι δύο πρώτες εβδομάδες κάθε εργαστηρίου είναι εβδομάδες προετοιμασίας/εξάσκησης
- Την τρίτη εβδομάδα παραδίδετε τεχνική αναφορά και για τα δύο σετ και εξετάζεστε προφορικά (πρόοδος)

Ημερομηνίες - Πρόγραμμα Διεξαγωγής Εργαστηρίων ανά Ομάδα (2019-2020)					
	Δευτέρα 16.00-18.00	**Τρίτη 09.00-11.00**	**Τρίτη 11.00-13.00**	**Πέμπτη 12.00-14.00**	**Πέμπτη 16.00-18.00**
Εβδ1: Εξάσκηση 1ο σετ (Java), ασκ. 1,2,3 φυλ.	Εβδομάδα 2 Μαρ-6 Μαρ				
Εβδ2: Εξάσκηση 1ο σετ (Java), ασκ. 1,2,3 φυλ.	Εβδομάδα 9 Μαρ-13 Μαρ				
Εβδ3: 1η Εξέταση Java (ασκ. 1,2,3 φυλ.-1ο σετ)	**Εβδομάδα 16 Μαρ-20 Μαρ**				
Εβδ4: Εξάσκηση 2ο σετ (Java), ασκ 4,5,6 φυλ.	Εβδομάδα 23 Μαρ-27 Μαρ				
Εβδ5: Εξάσκηση 2ο σετ (Java), ασκ 4,5,6 φυλ.	Εβδομάδα 30 Μαρ-3 Απρ				
Εβδ6: 2η Εξέταση Java (ασκ 4,5,6 φυλ.-2ο σετ)	**Εβδομάδα 6 Απρ-10 Απρ**				
	ΔΙΑΚΟΠΕΣ ΠΑΣΧΑ				
Εβδ7: Εξάσκηση 3ο σετ (C++), ασκ φυλ. C++	Εβδομάδα 27 Απρ-1 Μαϊου				
Εβδ8: Εξάσκηση 3ο σετ (C++), ασκ φυλ. C++	Εβδομάδα 4 Μαϊου-8 Μαϊου				
Εβδ9: Εξέταση 3ο σετ (C++)	**Εβδομάδα 11 Μαϊου-15 Μαϊου**				
Εβδ10: Επαναληπτικά εργαστήρια	Εβδομάδα 18 Μαϊου-22 Μαϊου				
Εβδ11: Εξέταση Εργασιών (Project)	**Εβδομάδα 1 Ιουνίου-5 Ιουνίου**				

ΑΞΙΟΛΟΓΗΣΗ

- Πρόοδοι στην ύλη του εργαστηρίου
  - Υποχρεωτικές; (χάνεται το αντίστοιχο ποσοστό σε περίπτωση μη προσέλευσης): **τρεις πρόοδοι + τελική εξέταση εργαστηρίου**
  - Βαρύτητα: **5%+5%+5%+10%=25%**
- Εργασία (Project)
  - Java και ένα μέρος C++ (15%)
- Γραπτή εξέταση
  - Βαρύτητα: 60%

ΑΞΙΟΛΟΓΗΣΗ

- Προϋπόθεση συμμετοχής στη γραπτή εξέταση
  - Να έχετε βαθμολογηθεί στην τελική εξέταση εργαστηρίου και σε 1 τουλάχιστον από τις 3 προόδους εργαστηρίου
- Προϋπόθεση προσμέτρησης εργαστηρίων και project
  - Να έχετε βαθμολογηθεί στην γραπτή εξέταση με τουλάχιστον 4.5

ΠΛΗΡΟΦΟΡΙΕΣ

Ιστοσελίδα εργαστηρίου
https://eclass.upatras.gr/courses/CEID1105/
(εδώ βρίσκεται αναρτημένο και το φυλλάδιο εργαστηρίου)

Δείτε προσεκτικά τη σελίδα ΠΛΗΡΟΦΟΡΙΕΣ

Στη σελίδα Έγγραφα θα αναρτώνται οι διαφάνειες

Contact info:
[Φροντιστήριο/Εργαστήριο]
Ελένη Βογιατζάκη , Δημήτρης Κουτσομητρόπουλος
evoyiatzaki@ceid.upatras.gr , kotsomit@ceid.upatras.gr

Δείτε το μάθημα στο eclass

Διαδικασία Εργαστηρίων

Μέθοδοι διδασκαλίας

Στο εργαστήριο έρχεστε κάθε εβδομάδα στο δίωρο που ανήκετε (βλ. Ομάδες Χρηστών) για να ασχοληθείτε με τις "ομάδες εργαστηριακών ασκήσεων" και τα αντίστοιχα "σετ ασκήσεων".

Συνολικά θα δοθούν τρια σετ εργαστηριακών ασκήσεων, τα δύο πρώτα πάνω σε JAVA και το τρίτο πάνω σε C++.

Εργαστηριακές Ασκήσεις JAVA:
Οι εργαστηριακές ασκήσεις βρίσκονται στο "Φυλλάδιο Εργαστηριακών Ασκήσεων JAVA". Από το φυλλάδιο αυτό θα γίνουν οι 6 πρώτες εργαστηριακές ασκήσεις, χωρισμένες σε δυο ομάδες: 1-2-3 και 4-5-6.

Εργαστηριακή Άσκηση C++:
Οι εργαστηριακές ασκήσεις βρίσκονται στο "Φυλλάδιο Εργαστηριακών Ασκήσεων C++".

Για κάθε μία από αυτές τις τρεις ομάδες εργαστηριακών ασκήσεων (2 Java, 1 C++) θα σας δίνεται και ένα αντίστοιχο σετ ασκήσεων. Για κάθε ομάδα εργαστηριακών ασκήσεων και αντίστοιχο σετ ασκήσεων έχετε δύο εβδομάδες στη διάθεσή σας για να ασχοληθείτε μαζί τους. Την τρίτη εβδομάδα θα έλθετε στο δίωρο που ανήκετε για εξέταση σ' αυτά. Το πρόγραμμα των εργαστηρίων για όλο το εξάμηνο θα βρίσκεται στον Ημερολόγιο. Οι ασκήσεις θα αναρτηθούν στην ενότητα 'Εργασίες' *(πρέπει να συνδεθείτε με τον λογαριασμό σας στο eclass)*, απο όπου θα γίνεται και η παράδοση τους.

Το αρχείο που θα παραδώσετε πρέπει να είναι **.rar ή .zip** και να περιέχει αναφορά pdf και κώδικα των ασκήσεων που υλοποιήσατε. Για τα Σετ ασκήσεων θα πρέπει να συμπεριλάβετε στην αναφορά και να περιλάβετε τον κώδικα και των αντίστοιχων εργαστηριακών ασκήσεων.
Για τις ασκήσεις σε Java εκτός της αναφοράς συμπεριλαμβάνετε τους φακέλους που δουλέψατε στο BlueJ (Αρκούν τα αρχεία .java και .bluej).

- Το αρχείο δεν θα πρέπει να ξεπερνά τα 3MB!
- Για τα πρότζεκτ, τα οποία είναι ομαδικά αρκεί να το παραδόσει ένα απο τα 3 μέλη της ομάδας.
- Στην αναφορά των Project πρέπει να αναγράφονται όλα τα ονόματα, τα ΑΜ και τα email των συνεργατών της ομάδας.

* Σε περίπτωση που χαθεί κάποια μέρα το εργαστήριο λόγω αργίας ή άλλου προβλήματος (π.χ. κλειστό υπολογιστικό λόγω απεργίας, κατάληψης ή εκλογών), οι φοιτητές μπορούν να προσέρχονται για επίλυση απορίων και εξέταση στα υπόλοιπα τμήματα της εβδομάδας.

C

Java

/** My first C program */

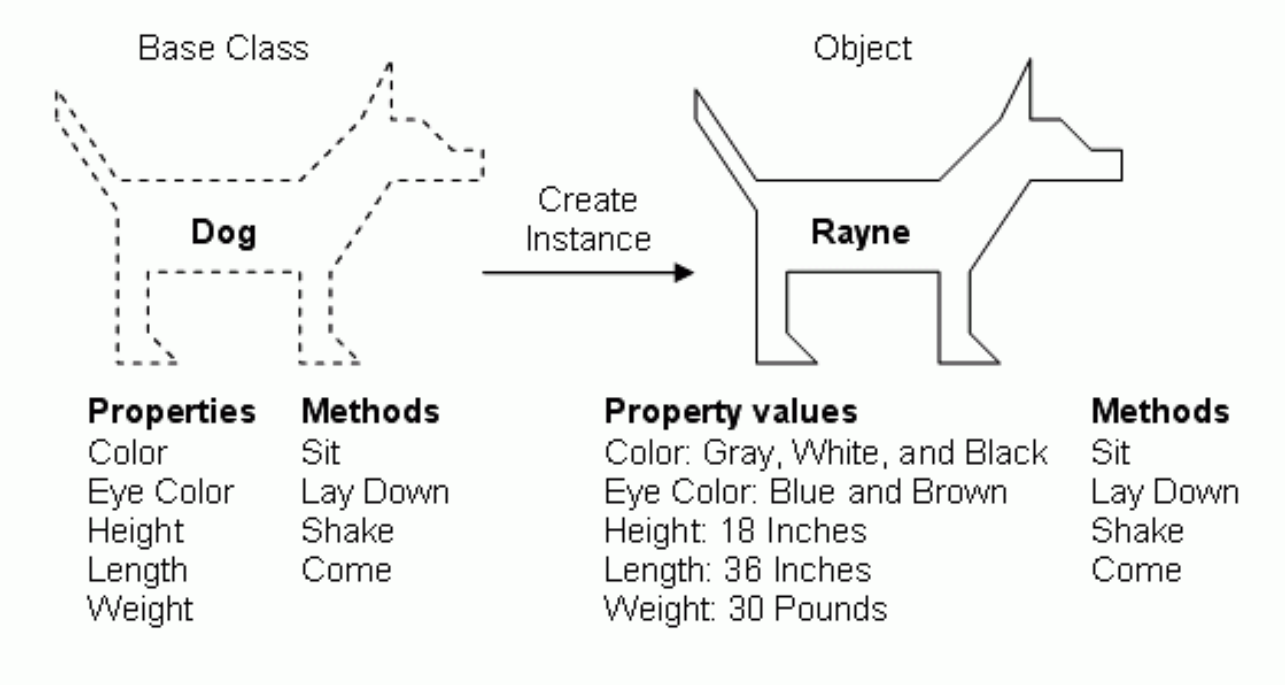
```
#include <stdio.h>
int main()

{
printf("Hello World\n");
printf(" Hello class 2019  \n ");
return 0;


}
```

```
public class HelloWorld
{    public static void main(String []args)
{
System.out.println("Hello World  \n");
 System.out.println("Hello class 2019  \n");
 }
}
```

```c
#include <stdio.h>
/* Created a structure here. The name of the structure is StudentData. */

struct StudentData
{
int stu_id;
int stu_grade; };

int main()
{
/* student is the variable of structure StudentData*/
struct StudentData student;

/*Assigning the values of each struct member here*/
student.stu_id = 1234;
student.stu_grade = 9;

/* Displaying the values of struct members */
printf("\nStudent Id is: %d", student.stu_id);
printf("\nStudent grade is: %d", student.stu_grade);

return 0; }
```

```java
public class Studentdata
{
   /* Created a structure here. The name of the structure is
StudentData. */
 int stu_id;
 int stu_grade;
    /*Constructor for objects of class studentdata
     */
    public Studentdata()
    {
   /*Assigning the values of each struct member here*/
       stu_id = 1234;
       stu_grade = 9;
    }

    public  void Printdata ()
    {
     /* Displaying the values of struct members */
    System.out.println("Student Id is: "+ stu_id);
    System.out.println("Student grade is: "+ stu_grade);
    }

    public static void main ()
    {
    Studentdata st = new Studentdata();
    st.Printdata();

    }
}
```

Αντικειμενοστρεφής τρόπος σκέψης



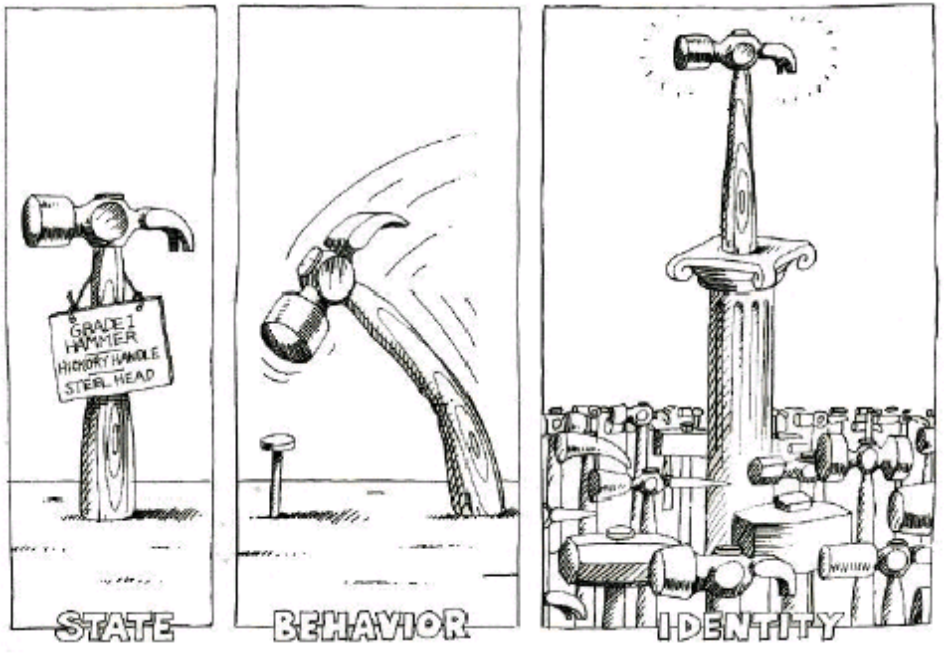
Βασικές αρχές αντικειμενοστρέφειας (αρχές του Allan Kay)

- Τα πάντα είναι αντικείμενα
- Κάθε αντικείμενο έχει δικιά του μνήμη
- Κάθε αντικείμενο έχει ένα συγκεκριμένο τύπο
  - Τύπος = Κλάση
- Τα αντικείμενα επικοινωνούν μέσω μηνυμάτων
- Αντικείμενα του ίδιου τύπου μπορούν να δεχτούν τα ίδια μηνύματα
  - Δηλαδή μπορούν να εκτελούν τις ίδιες λειτουργίες
- Ένα πρόγραμμα είναι μια συλλογή από αντικείμενα όπου το ένα λέει στο άλλο τι να κάνει

Αντικείμενο

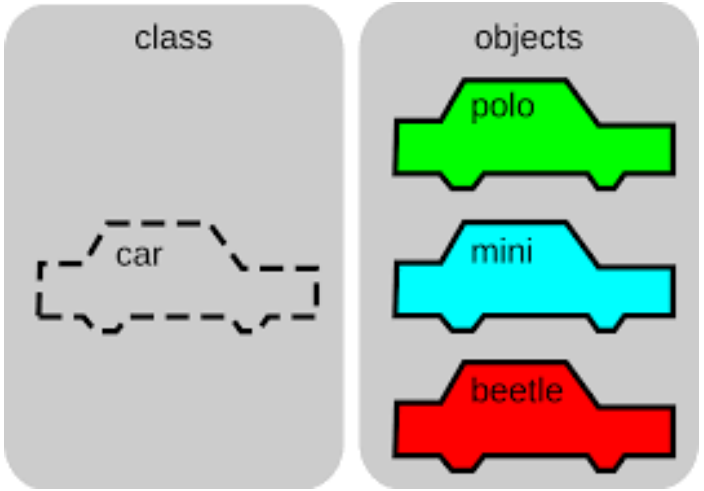
- Ένα αντικείμενο στον κώδικα αναπαριστά μια οντότητα (έννοια) και έχει:

- Μια κατάσταση, η οποία ορίζεται από ορισμένα χαρακτηριστικά
- Μια συμπεριφορά, η οποία ορίζεται από ορισμένες ενέργειες που μπορεί να εκτελέσει το αντικείμενο
- Μια ταυτότητα που το ξεχωρίζει από τα υπόλοιπα.



Παραδείγματα: ένας άνθρωπος, ένα πράγμα, ένα μέρος, μια υπηρεσία

Κλάσεις



- Κλάση: Μια αφηρημένη περιγραφή αντικειμένων με κοινά χαρακτηριστικά και κοινή συμπεριφορά
  - Ένα καλούπι που παράγει αντικείμενα
  - Ένα αντικείμενο είναι ένα στιγμιότυπο μίας κλάσης.

- Π.χ., η κλάση φοιτητής έχει τα γενικά χαρακτηριστικά (όνομα, βαθμοί) και τη συμπεριφορά print
  - Ο φοιτητής Χ είναι ένα αντικείμενο της κλάσης φοιτητής

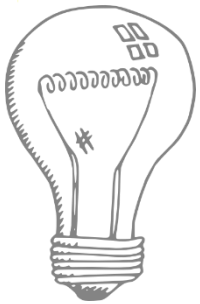
- Η κλάση Car έχει τα χαρακτηριστικά (brand, color) και τη συμπεριφορά (drive, stop)
  - Το αυτοκίνητο ΑΧΧ2013 είναι ένα αντικείμενο της κλάσης Car με κατάσταση τα χαρακτηριστικά (BMW, red)

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

- Ένα αντικείμενο ορίζεται ως μια μεταβλητή τύπου Κ
  - Στην Java (όπως και στις περισσότερες γλώσσες) όλες οι μεταβλητές έχουν ένα τύπο (“strongly typed”).
  - Το αντικείμενο που δημιουργείται παίρνει κάποιες τιμές στα πεδία της κλάσης και καταλαμβάνει κάποιο χώρο στη μνήμη.
    - Έτσι μετατρέπεται σε ένα φυσικό αντικείμενο με μοναδική ταυτότητα.

Διαχειρίζομαι τους λαμπτήρες σε ένα σπίτι



Θέλουμε να κάνουμε ένα πρόγραμμα που να διαχειρίζεται τους λαμπτήρες (φώτα) σε διάφορα δωμάτια και θα υλοποιεί και ένα dimmer

Αντικείμενα:

bedroomLight
kitchenLight

Light
intensity
on() off() dim() brighten()

Όνομα κλάσης

Πεδία κλάσης

Μέθοδοι κλάσης

Τα αντικείμενα δημιουργούνται σε άλλο σημείο του κώδικα το οποίο καλεί και τις μεθόδους

Light bedroomLight
Light kitchenLight

Η κλήση μιας μεθόδου σε ένα αντικείμενο μερικές φορές λέγεται και πέρασμα μηνύματος

Πλεονεκτήματα Object Orientation

- Τα αντικείμενα και οι κλάσεις μοντελοποιούν με φυσικό τρόπο τα αντικείμενα του κόσμου
- Τα πλεονεκτήματα είναι ότι αυτό κάνει τον κώδικα
  - πιο ευανάγνωστο
  - πιο τμηματοποιημένο
  - και πιο εύκολο να συντηρηθεί.

Παράδειγμα

- Θέλουμε να προσομοιώσουμε ένα αυτοκίνητο το οποίο κινείται πάνω σε μία ευθεία. Αρχικά ξεκινάει από τη θέση μηδέν. Σε κάθε χρονική στιγμή κινείται τυχαία κατά μία θέση είτε αριστερά είτε δεξιά. Σε κάθε βήμα τυπώνεται μια κουκίδα που δείχνει τη θέση του.

- Πώς θα λύσουμε αυτό το πρόβλημα?
  - Τι κλάσεις και τι αντικείμενα θα ορίσουμε?
  - Τι πεδία και τι μεθόδους θα έχουν?

Αντικειμενοστρεφής Σχεδίαση

- Οι οντότητες/έννοιες στον ορισμό του προβλήματος γίνονται κλάσεις και ορίζονται τα αντικείμενα που αναφέρονται στο πρόβλημα.
- Τα ρήματα γίνονται μέθοδοι
- Τα χαρακτηριστικά των αντικειμένων γίνονται πεδία
  - Τα πεδία μπορεί να είναι κι αυτά αντικείμενα.

- Δεν υπάρχει ένας μοναδικός τρόπος να μοντελοποιήσετε ένα πρόβλημα. Συνήθως όμως υπάρχει ένας που είναι καλύτερος από τους άλλους.

Παράδειγμα

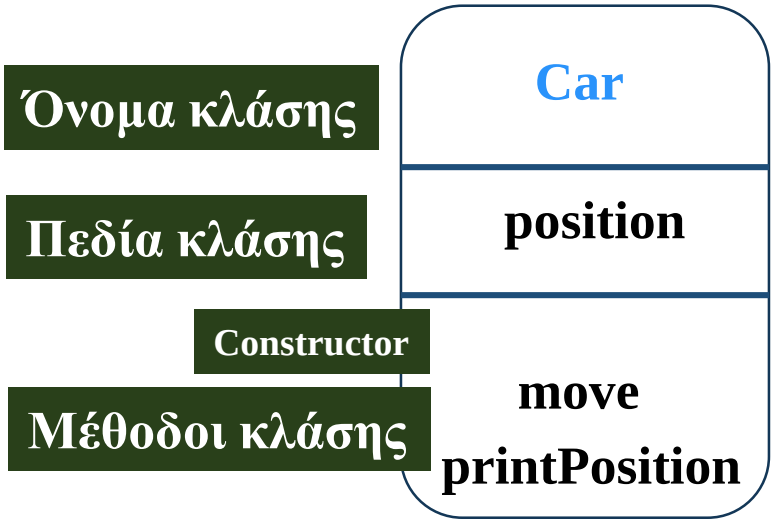
- Θέλουμε να προσομοιώσουμε ένα αυτοκίνητο το οποίο κινείται πάνω σε μία ευθεία. Αρχικά ξεκινάει από τη θέση μηδέν. Σε κάθε χρονική στιγμή κινείται τυχαία κατά μία θέση είτε αριστερά είτε δεξιά. Σε κάθε βήμα τυπώνεται μια κουκίδα που δείχνει τη θέση του.

- Πώς θα λύσουμε αυτό το πρόβλημα?
  - Τι κλάσεις και τι αντικείμενα θα ορίσουμε?
  - Τι πεδία και τι μεθόδους θα έχουν?

Στην πραγματικότητα ενώ μιλάω για αυτοκίνητο με ενδιαφέρει η θέση του.

Υλοποίηση

Αν έχω δύο αυτοκίνητα?



Programming Style

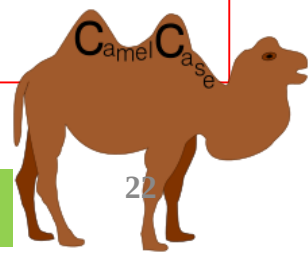
- Τα ονόματα των κλάσεων ξεκινάνε με κεφαλαίο, τα ονόματα των πεδίων, μεθόδων και αντικειμένων με μικρό.
- Χρησιμοποιούμε ολόκληρες λέξεις (και συνδυασμούς τους) για τα ονόματα
  - Δεν πειράζει αν βγαίνουν μεγάλα ονόματα
- Χρησιμοποιούμε το **CamelCase** Style
  - Όταν για ένα όνομα έχουμε πάνω από μία λέξη, τις συνενώνουμε και στο σημείο συνένωσης κάνουμε το πρώτο γράμμα της λέξης κεφαλαίο
  - printPosition όχι print_position
- Χρησιμοποιούμε **κεφαλαία** και '_' για τις **σταθερές**.

Πρόγραμμα
Δημιούργησε το αντικείμενο myCar τύπου Car

```
Car myCar = new Car()

myCar.printPosition()
For i = 1…10
    myCar.move()
    myCar.printPosition()
```

Λείπει κάτι?

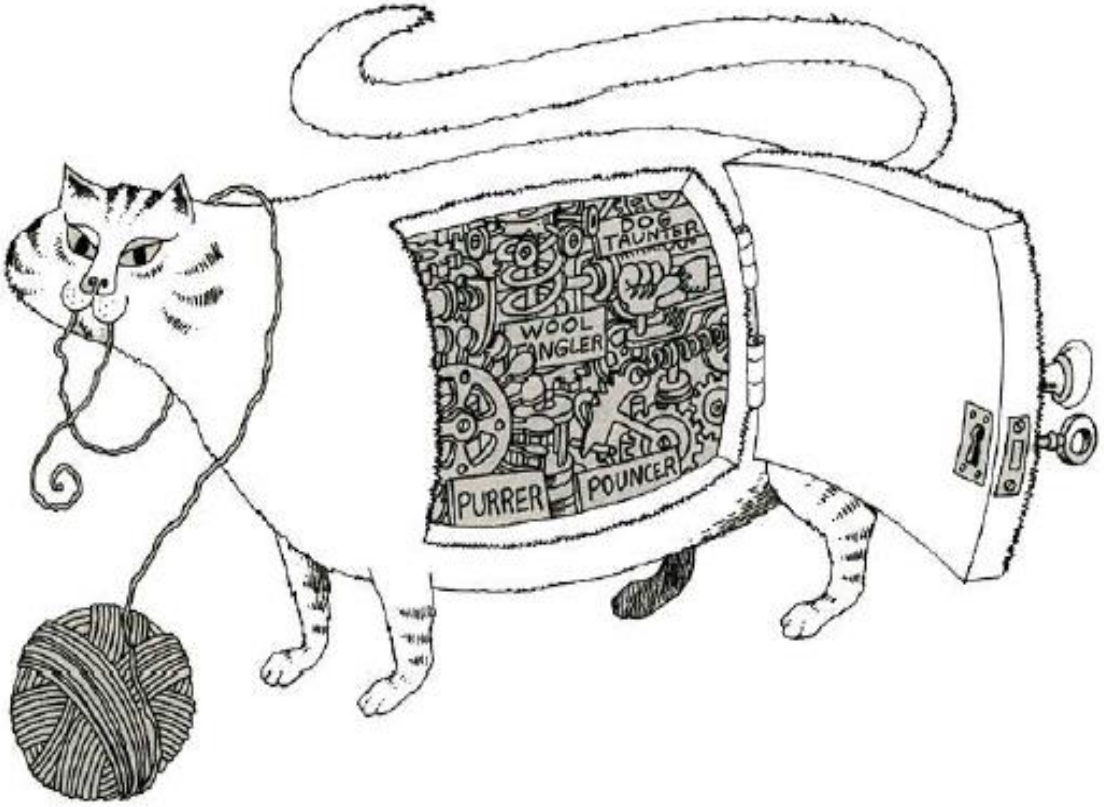
Αρχικοποίηση?



Απόκρυψη - Ενθυλάκωση

- Στο πρόγραμμα που κάναμε πριν δεν είχαμε πρόσβαση στην θέση του αυτοκινήτου
  - Μόνο οι μέθοδοι της κλάσης μπορούν να την αλλάξουν.
  - Γιατί?
    - Αν μπορούσε να αλλάζει σε πολλά σημεία στον κώδικα τότε κάποιες άλλες μέθοδοι θα μπορούσαν να το αλλάξουν και να δημιουργηθεί μπέρδεμα
    - Τώρα αλλάζει μόνο όταν είναι λογικό να αλλάξει – όταν κινηθεί το όχημα.
- Κάποιος που χρησιμοποιεί τις μεθόδους της κλάσης δεν ξέρει πως υλοποιούνται, γνωρίζει μόνο τι κάνουν και πως μπορεί να τις καλέσει
- Αυτή η αρχή λέγεται Ενθυλάκωση – Απόκρυψη Πληροφορίας

Ενθυλάκωση



- **Η στεγανοποίηση της κατάστασης και της συμπεριφοράς ώστε οι λεπτομέρειες της υλοποίησης να είναι κρυμμένες από το χρήστη του αντικειμένου.**

Παράδειγμα

- Τι γίνεται αν στο αρχικό μας παράδειγμα αν εκτός από αυτοκίνητο είχαμε και μία μηχανή? Η μηχανή κινείται με τον ίδιο τρόπο αλλά όταν τυπώνεται η θέση της αντί για κουκίδα (.) τυπώνεται ένα αστέρι (*).

- Τι κλάσεις πρέπει να ορίσουμε?

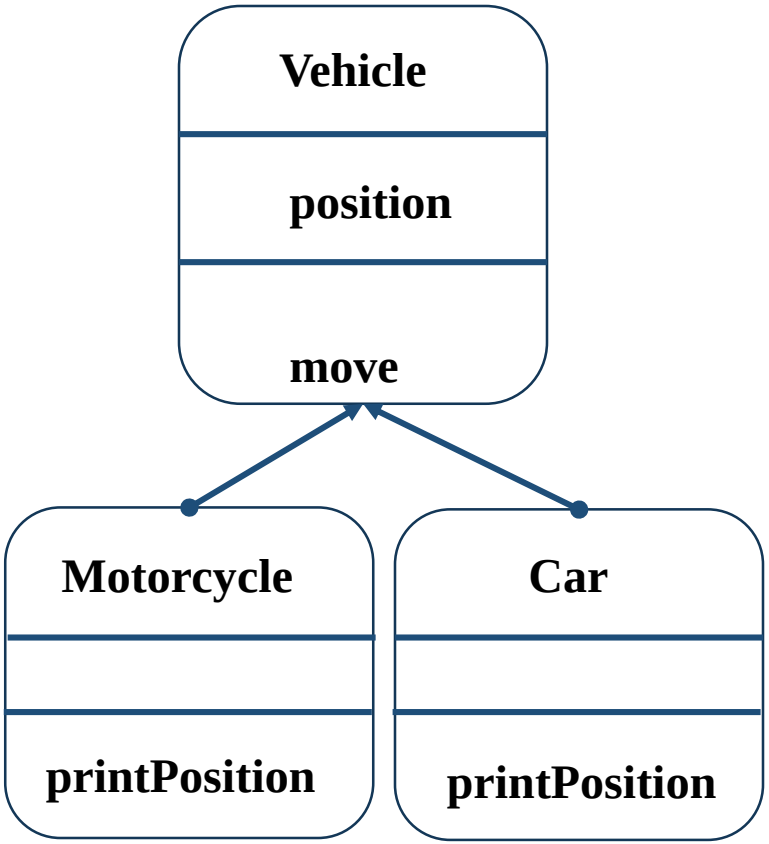
Μία λύση

- Μπορούμε να ορίσουμε ξανά από την αρχή μια καινούρια κλάση για τη μηχανή που να κάνει την ίδια κίνηση με το αυτοκίνητο (ίδια μέθοδο move) αλλά τυπώνεται διαφορετικά (διαφορετική μέθοδο printPosition)

- Τι προβλήματα έχει αυτό?
  - Επανάληψη του κώδικα (μπορεί να είναι μεγάλος και δύσκολος)
  - Πιθανότητα για λάθη
  - Πολύ δύσκολο να γίνουν αλλαγές (θα πρέπει να γίνονται σε δύο σημεία)

Κληρονομικότητα

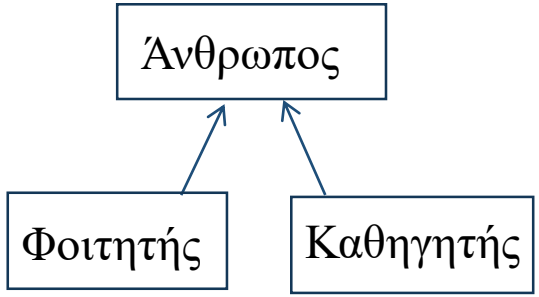
- Ορίζουμε μια κλάση Vehicle (όχημα) η οποία έχει θέση, και έχει κίνηση (μέθοδο move)

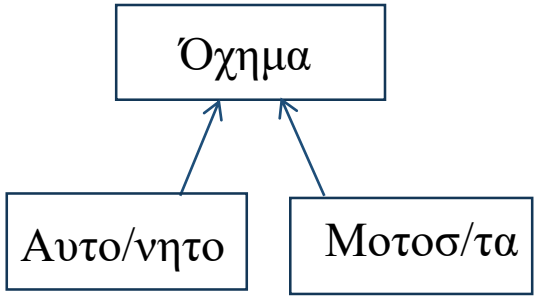



```
Πρόγραμμα
Car myCar = new Car()
Motorcycle myMoto = new Motorcycle()
myCar.printPosition()
myMoto.printPosition()
For i = 1…10
        myCar.move()
        myCar.printPosition()
        myMoto.move()
        myMoto.printPosition()
```

Κληρονομικότητα

- Οι κλάσεις μας επιτρέπουν να ορίσουμε μια ιεραρχία
  - Π.χ., και ο Φοιτητής και ο Καθηγητής ανήκουν στην κλάση Άνθρωπος.
  - Η κλάση Αυτοκίνητο ανήκει στην κλάση Όχημα η οποία περιέχει και την κλάση Μοτοσυκλέτα
- Οι κλάσεις πιο χαμηλά στην ιεραρχία κληρονομούν χαρακτηριστικά και συμπεριφορά από τις ανώτερες κλάσεις
  - Όλοι οι άνθρωποι έχουν όνομα
  - Όλα τα οχήματα έχουν μέθοδο drive, stop.





Οι κλάσεις σχηματίζουν ιεραρχία

- Οι κλάσεις δομούνται σε μια δενδρική ιεραρχία
- Η κλάση στη ρίζα της ιεραρχίας ονομάζεται Object
- Κάθε κλάση εκτός από την Object, έχει μια υπερκλάση (superclass)
- Μια κλάση μπορεί να έχει πολλούς προγόνους μέχρι την Object
- Όταν ορίζουμε μια κλάση, ορίζουμε την υπερκλάση της
  - Αν δεν ορίσουμε υπερκλάση θεωρείται η Object
- Κάθε κλάση μπορεί να έχει μία ή περισσότερες υποκλάσεις (subclasses)

Πολυμορφισμός

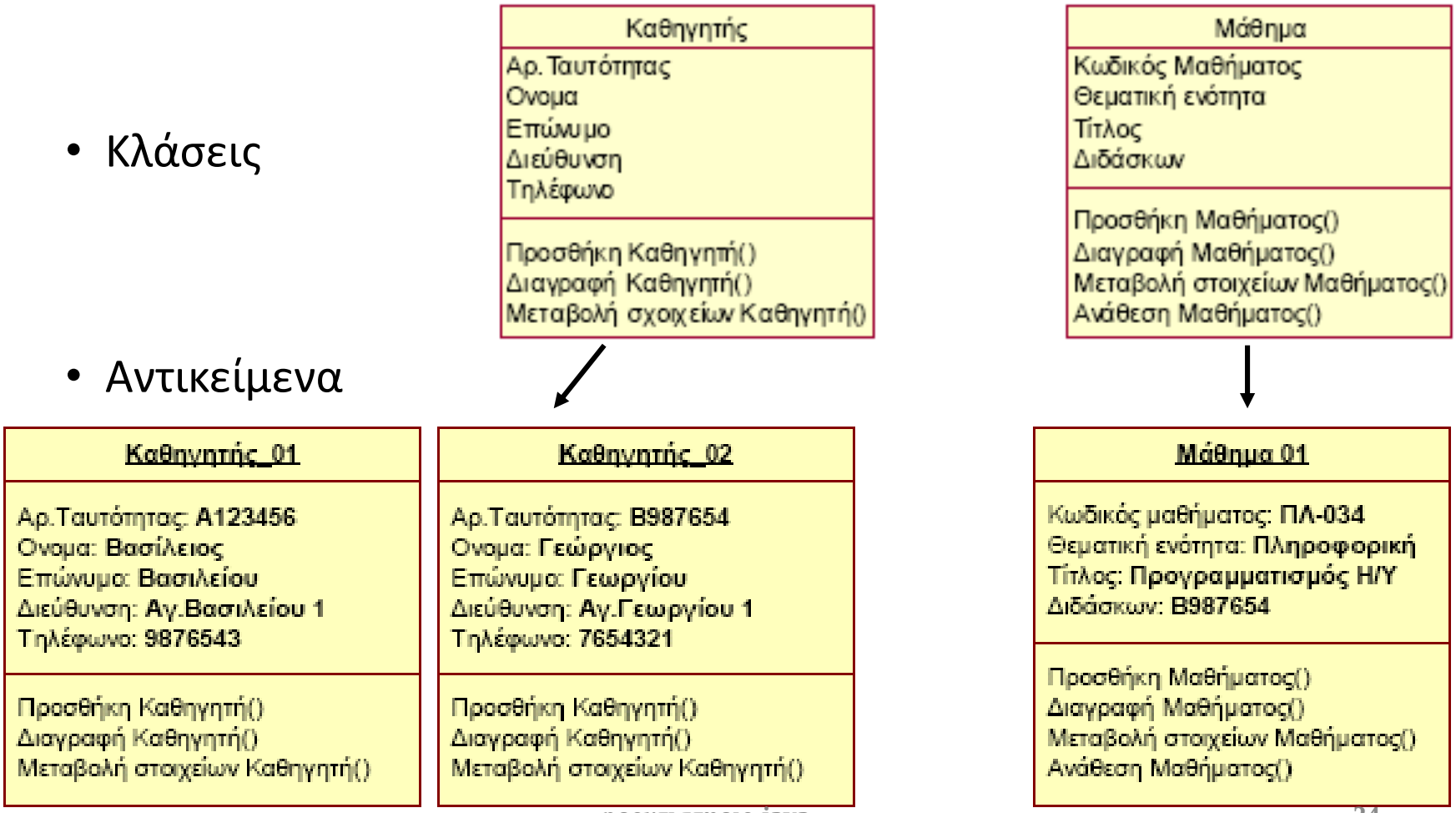
- Κλάσεις με κοινό πρόγονο έχουν κοινά χαρακτηριστικά, αλλά έχουν και διαφορές
  - Π.χ., είναι διαφορετικό το παρκάρισμα για ένα αυτοκίνητο και μια μηχανή
- Ο πολυμορφισμός μας επιτρέπει να δώσουμε μια κοινή συμπεριφορά σε κάθε κλάση (μια μέθοδο park), η οποία όμως υλοποιείται διαφορετικά για αντικείμενα διαφορετικών κλάσεων.
- Μπορούμε επίσης να ορίσουμε αφηρημένες κλάσεις, όπου προϋποθέτουμε μια συμπεριφορά και αυτή πρέπει να υλοποιηθεί σε χαμηλότερες κλάσεις διαφορετικά ανάλογα με τις ανάγκες μας

ΚΛΑΣΕΙΣ ΚΑΙ ΑΝΤΙΚΕΙΜΕΝΑ στη UML

UML: Μια επίσημη γλώσσα μοντελοποίησης

Κλάσεις και αντικείμενα στη UML

- Κλάσεις
- Αντικείμενα



Σχέσεις μεταξύ κλάσεων

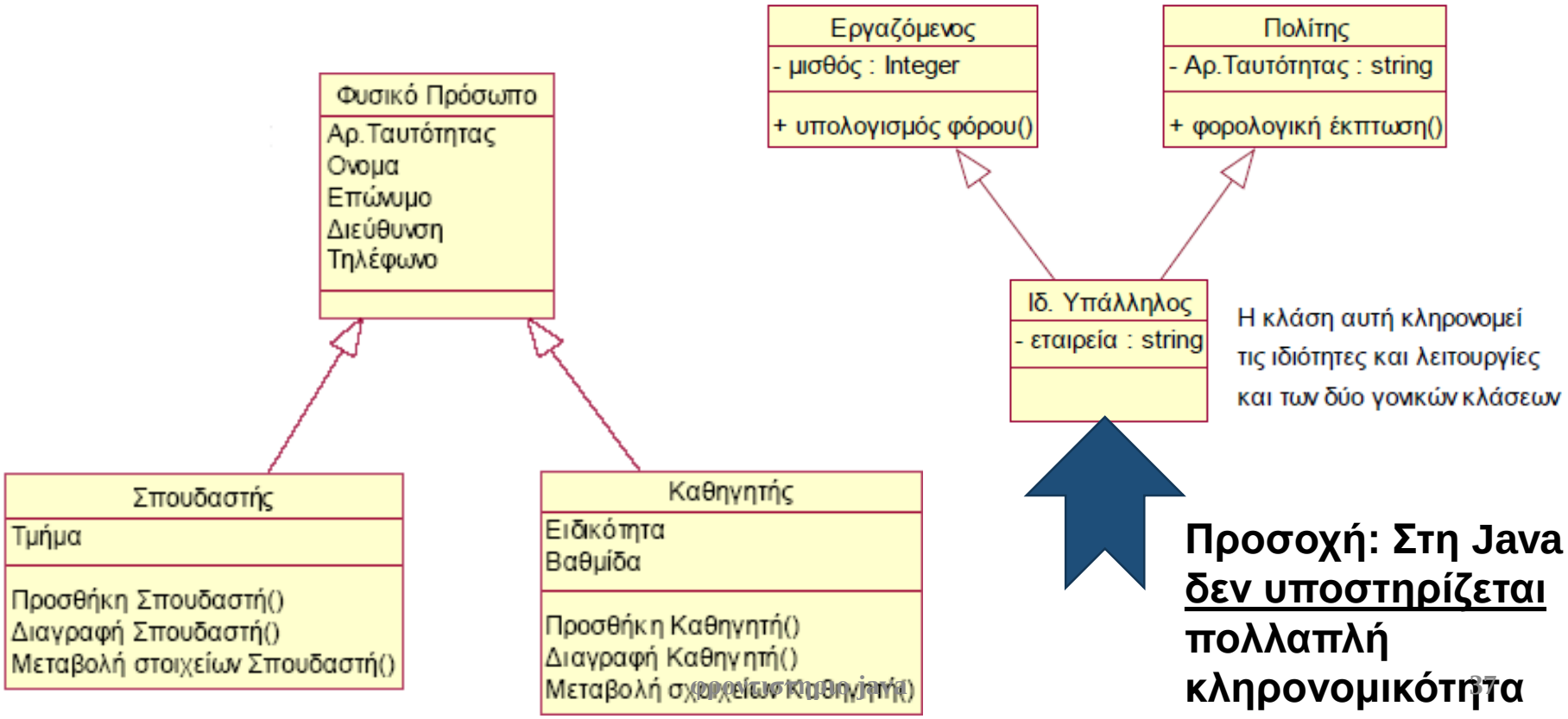
- Συσχέτιση (association): Μια γενική σχέση μεταξύ κλάσεων
  - Περιγραφή συσχέτισης (όνομα)
  - Πολλαπλότητα
  - Ρόλοι

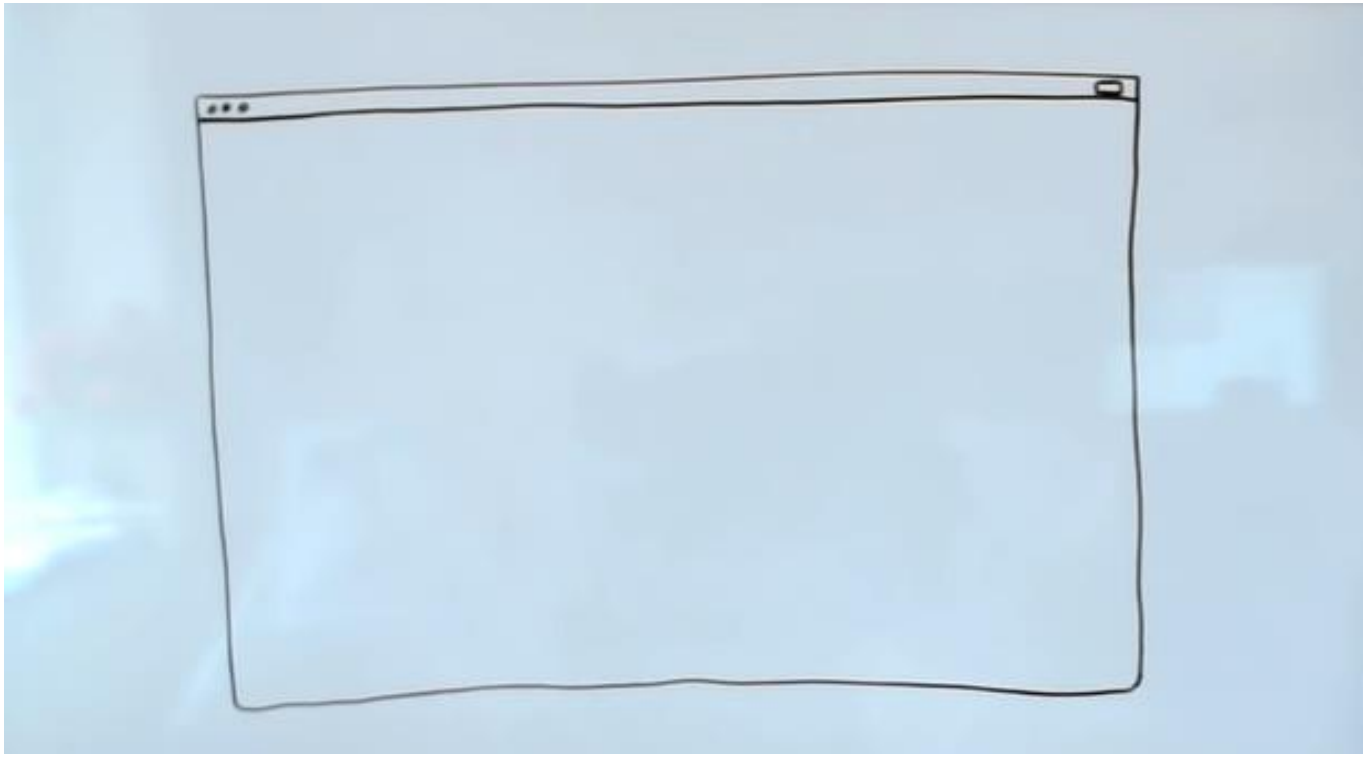
Κληρονομικότητα (1/2)

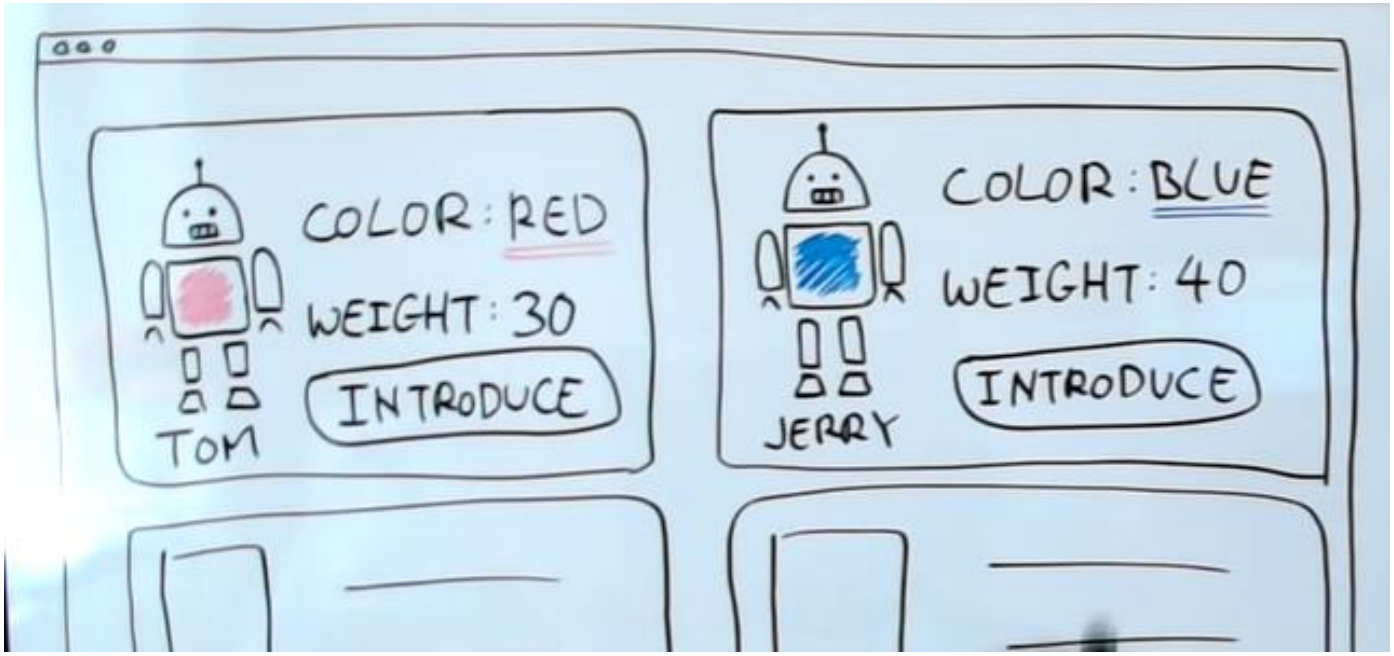
- ***"is a"***, ***"kind of"***
- ***Κληρονομικότητα*** ή ***γενίκευση*** (inheritance, generalisation): Η απόδοση χαρακτηριστικών από μια κλάση (πατέρας) σε άλλες (παιδιά).
  - Απλή: κάθε κλάση έχει έναν «πατέρα»
  - Πολλαπλή: κάθε κλάση έχει πάνω από έναν «πατέρα»
- Κληρονομικότητα και γενίκευση αποτελούν τις δύο όψεις ενός μηχανισμού ***ταξινόμησης*** (classification) οντοτήτων του πεδίου του προβλήματος:
  - Η κλάση-παιδί είναι εξειδίκευση της κλάσης-πατέρα
  - Η κλάση-πατέρας είναι γενίκευση της κλάσης-παιδί
- Οι υποκλάσεις προσφέρουν *εξειδικευμένη* συμπεριφορά από τα κοινά στοιχεία που προσφέρει η υπερκλάση.
- Επαναχρησιμοποίηση του κώδικα της υπερκλάσης.

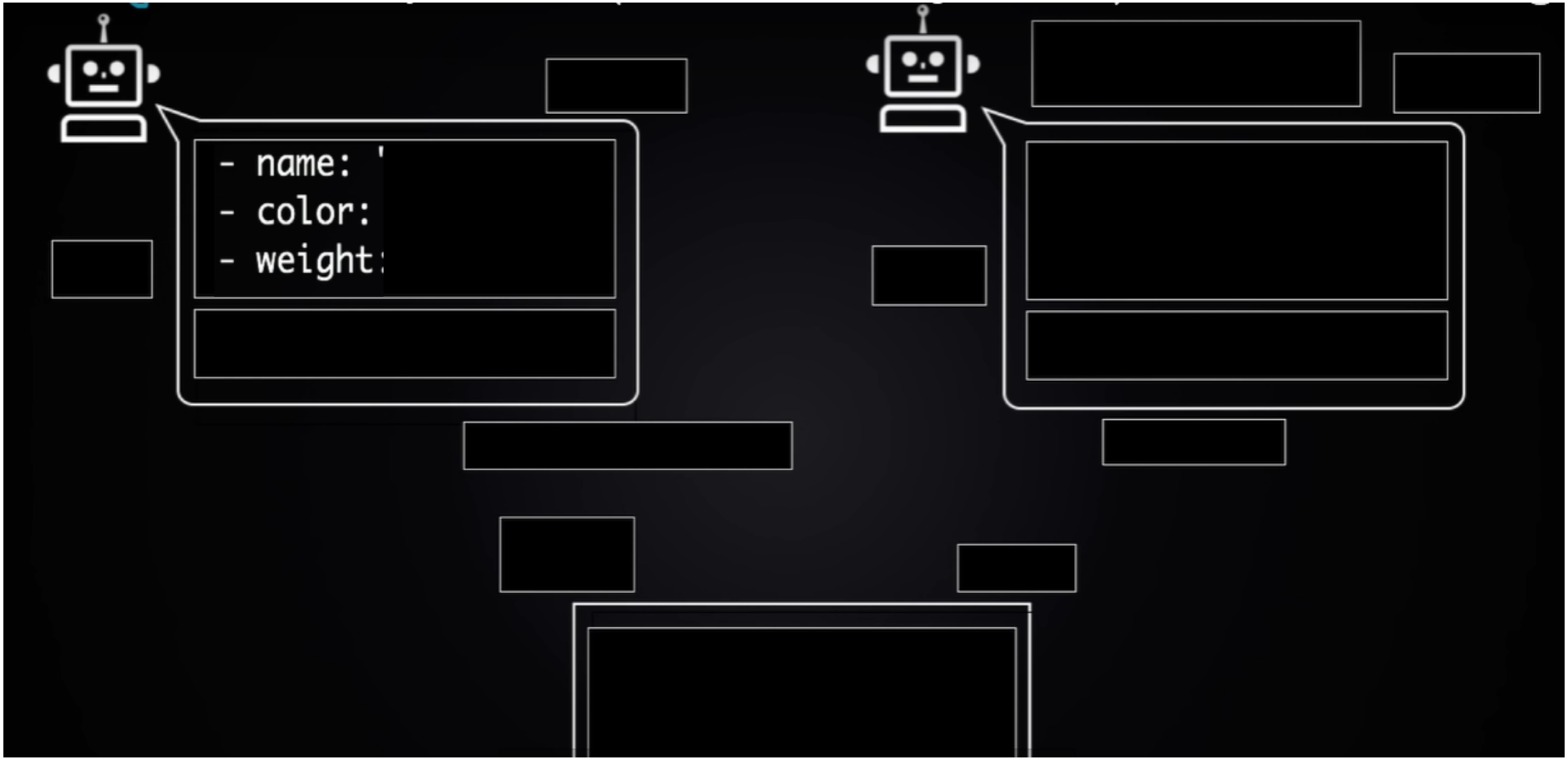
Κληρονομικότητα (2/2)

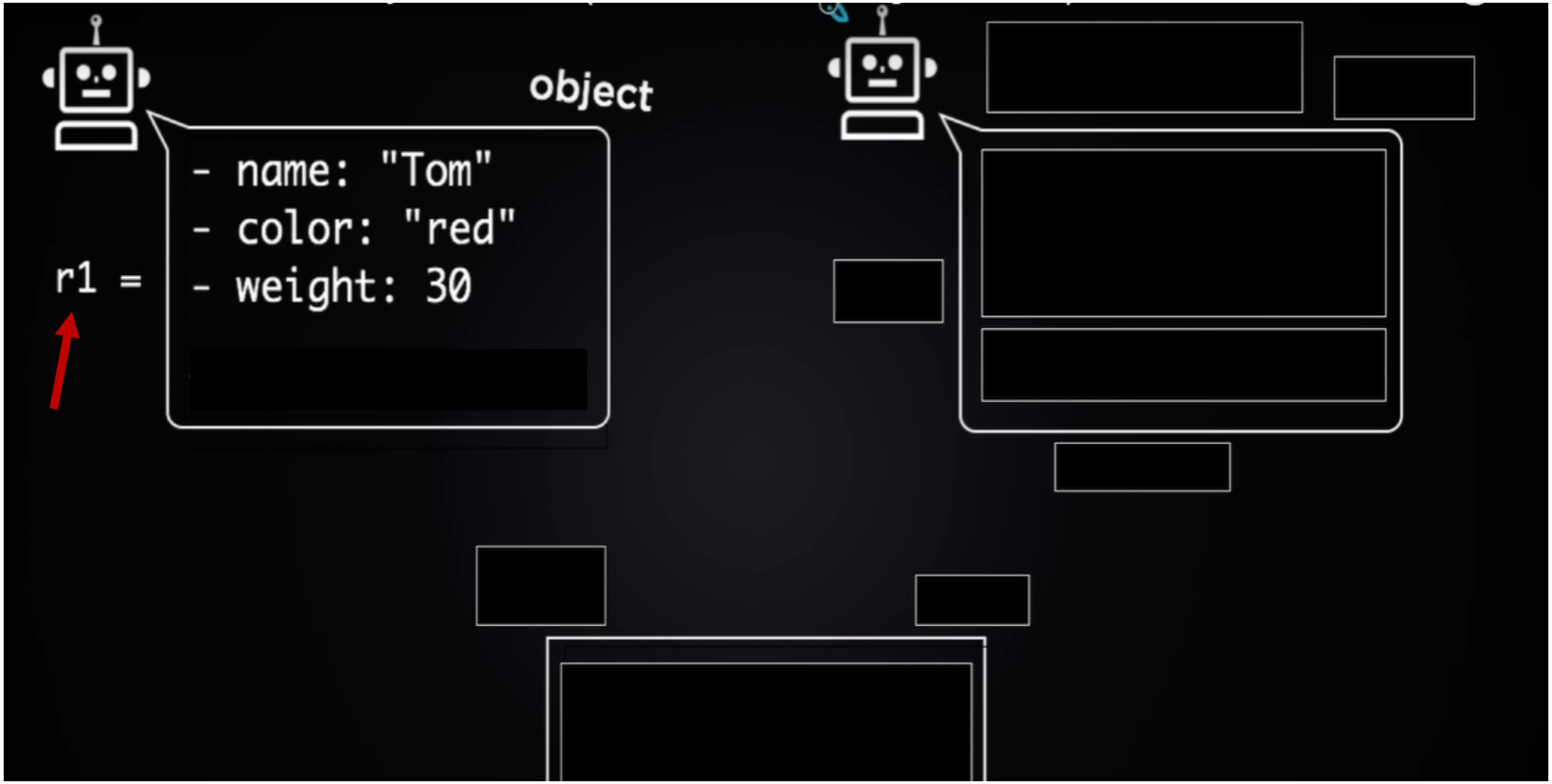
- Κάθε κλάση-παιδί έχει όλα τα χαρακτηριστικά της κλάσης-πατέρα και ορισμένα **επιπλέον**

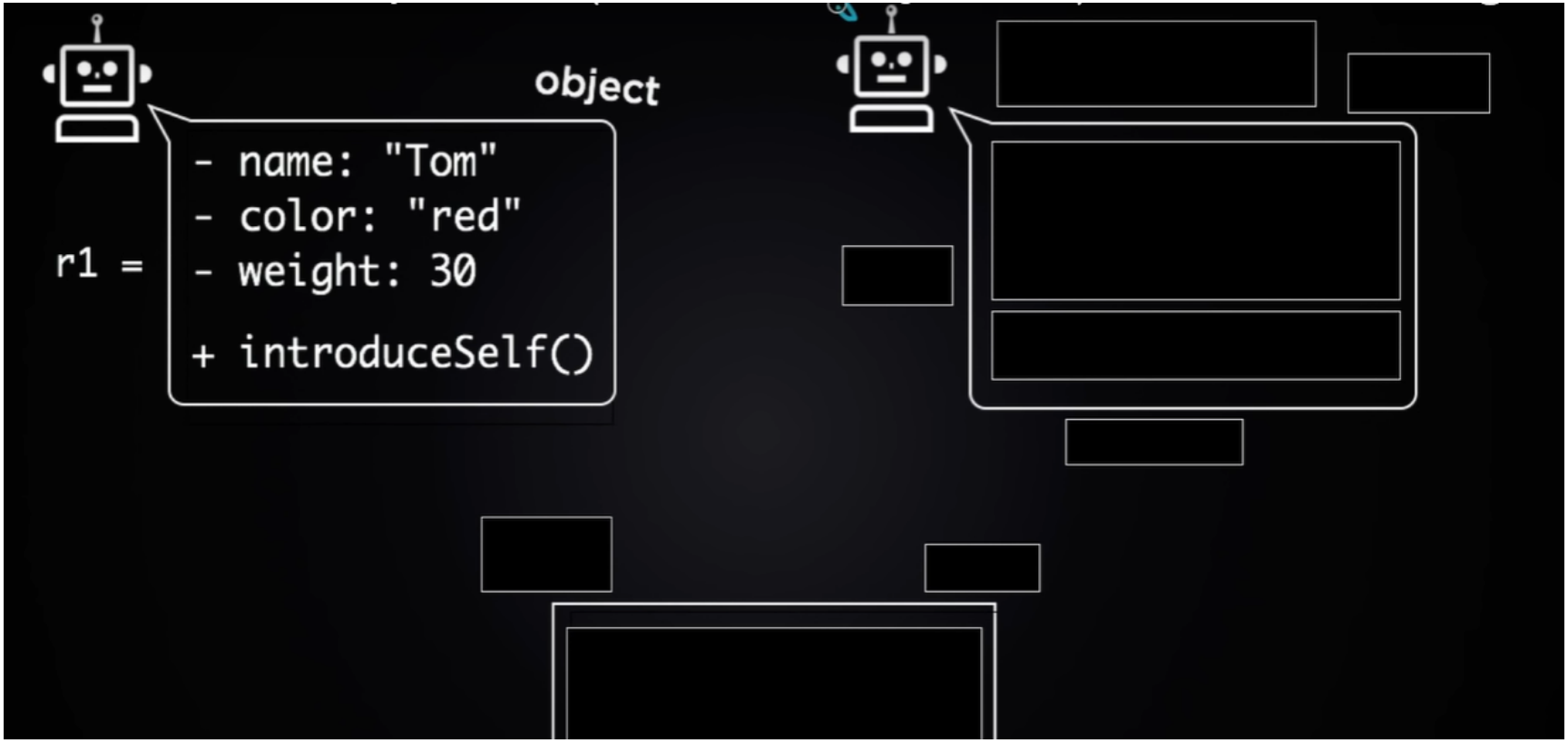


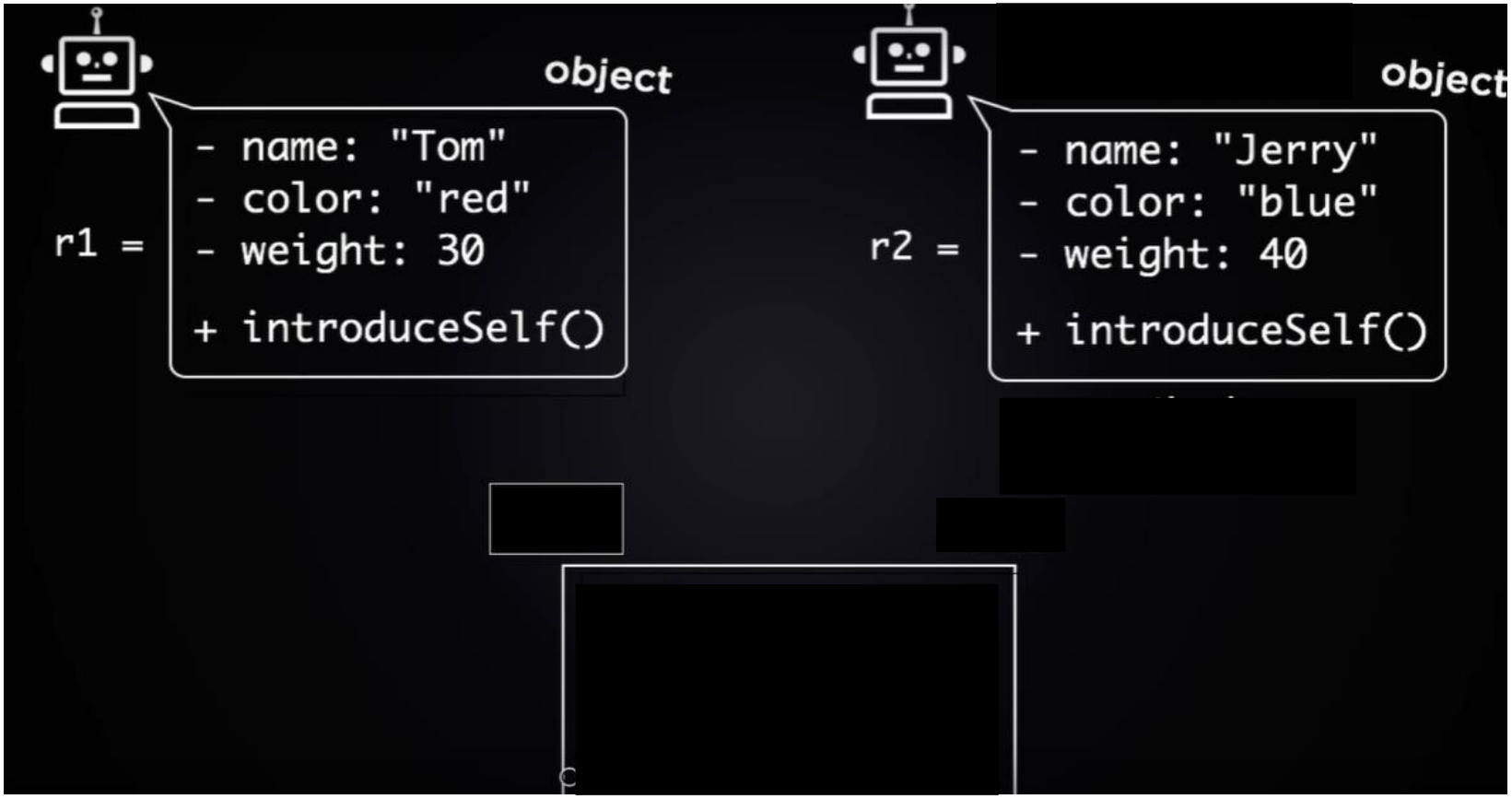


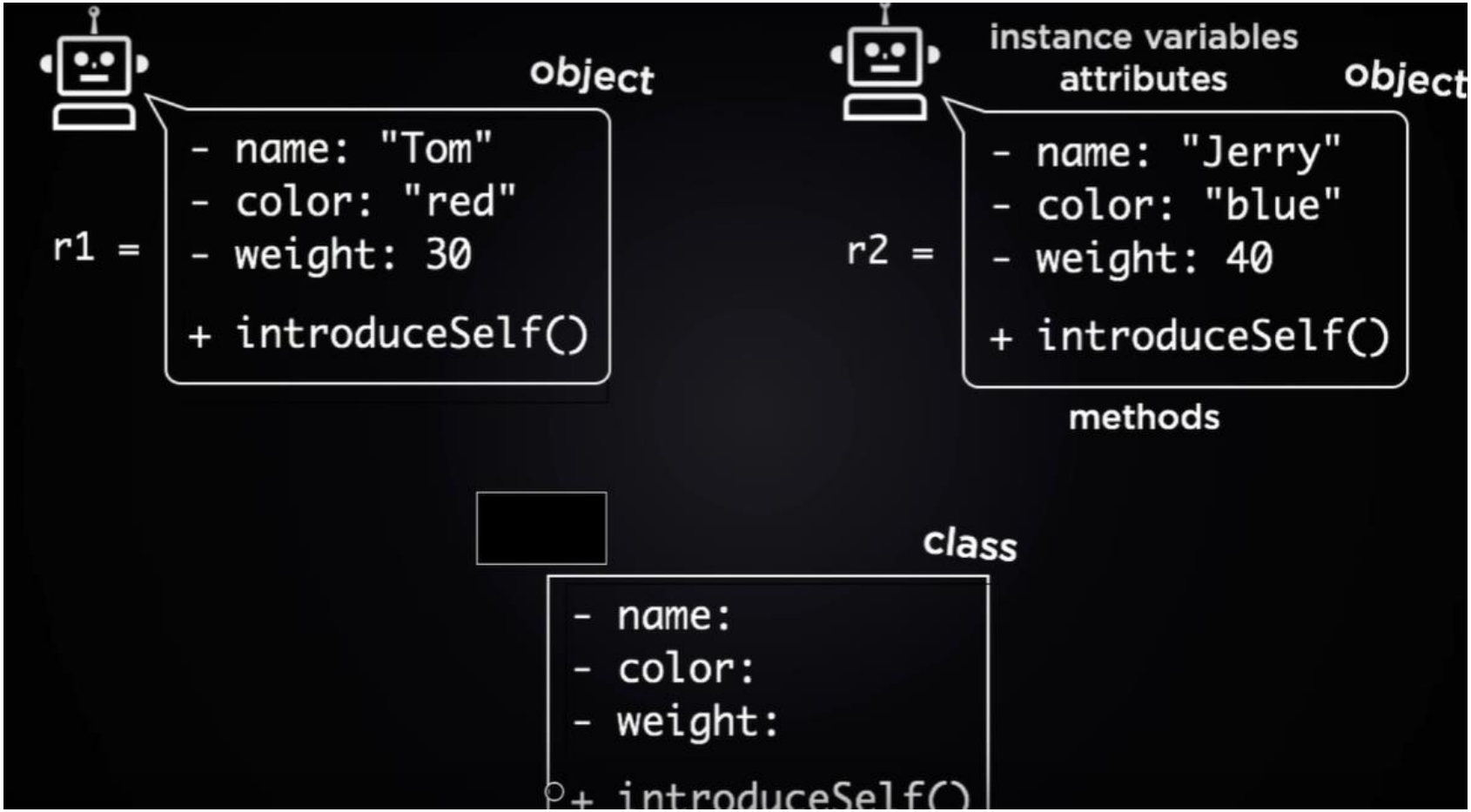

COLOR: RED
WEIGHT: 30
INTRODUCE
TOM
COLOR: BLUE
WEIGHT: 40
INTRODUCE
JERRY

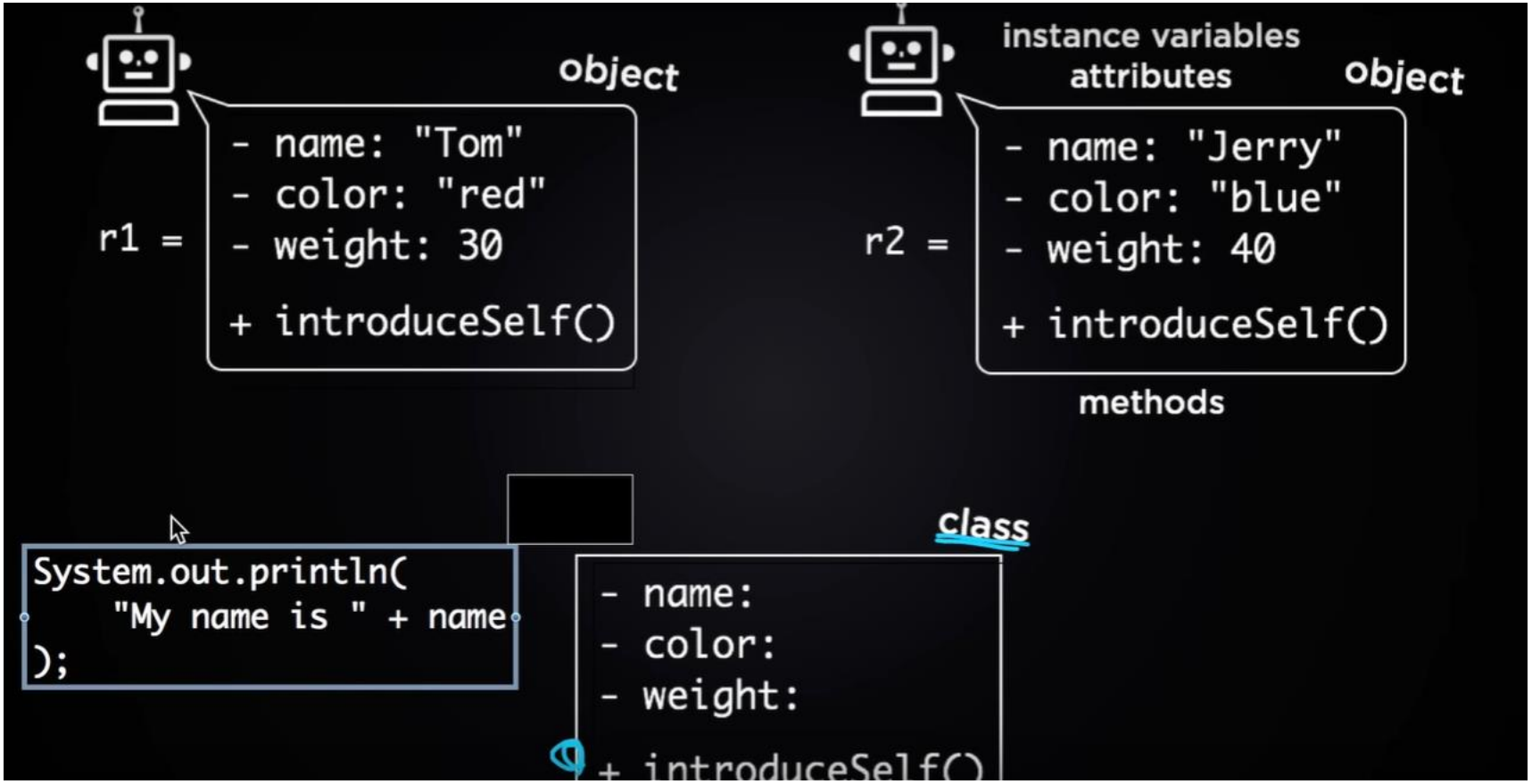

- name:
- color:
- weight:

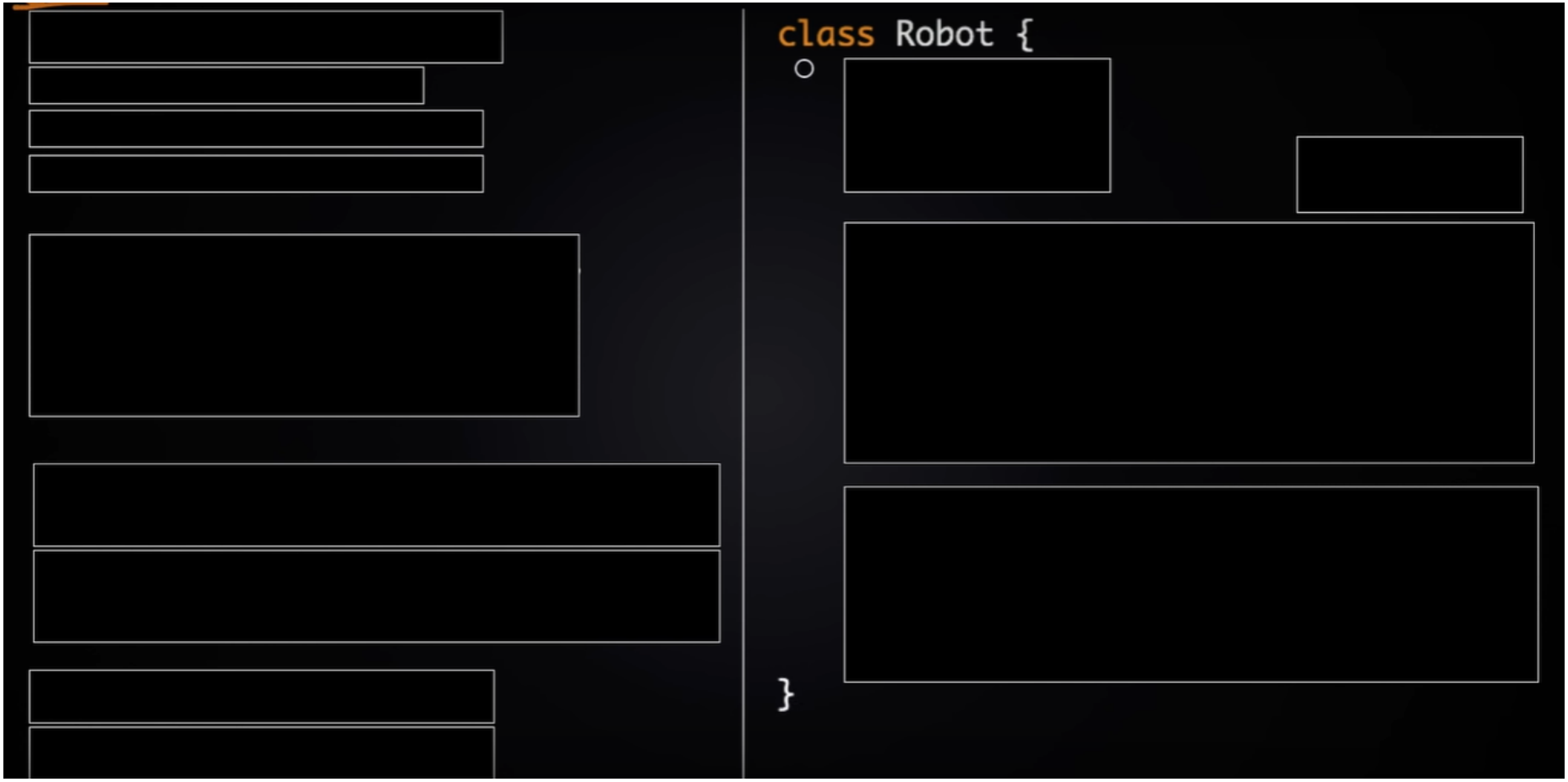

object
- name: "Tom"
- color: "red"
- weight: 30
r1 =


object
r1 =
- name: "Tom"
- color: "red"
- weight: 30
+ introduceSelf()


object
- name: "Tom"
- color: "red"
r1 =
- weight: 30
+ introduceSelf()
object
- name: "Jerry"
- color: "blue"
r2 =
- weight: 40
+ introduceSelf()


object
r1 =
- name: "Tom"
- color: "red"
- weight: 30
+ introduceSelf()
instance variables
attributes
object
r2 =
- name: "Jerry"
- color: "blue"
- weight: 40
+ introduceSelf()
methods
class
- name:
- color:
- weight:
+ introduceSelf()


object
- name: "Tom"
- color: "red"
- weight: 30
+ introduceSelf()
r1 =
instance variables
attributes
object
- name: "Jerry"
- color: "blue"
- weight: 40
+ introduceSelf()
r2 =
methods
System.out.println(
"My name is " + name
);
class
- name:
- color:
- weight:
+ introduceSelf()

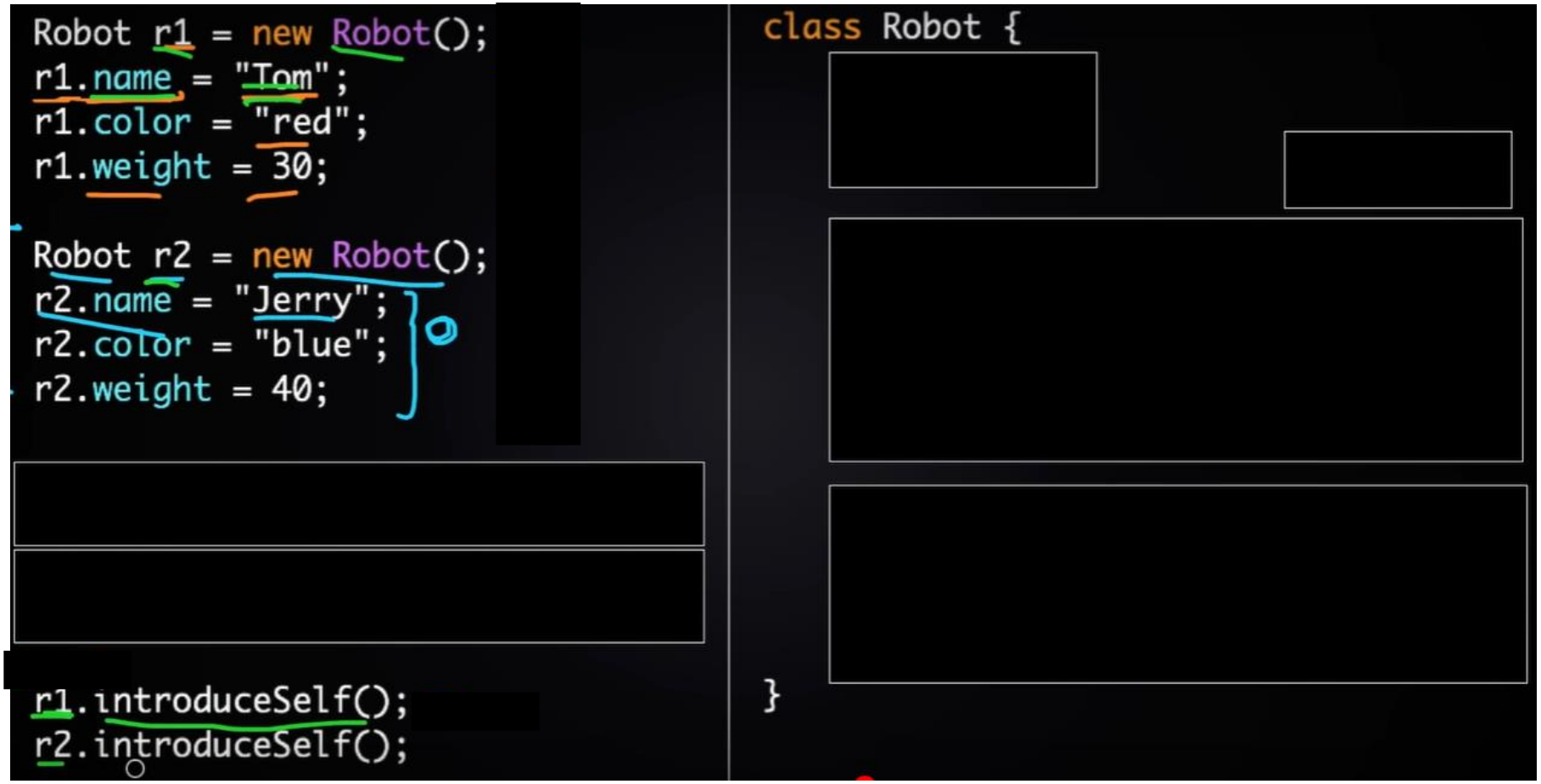

class Robot {
}

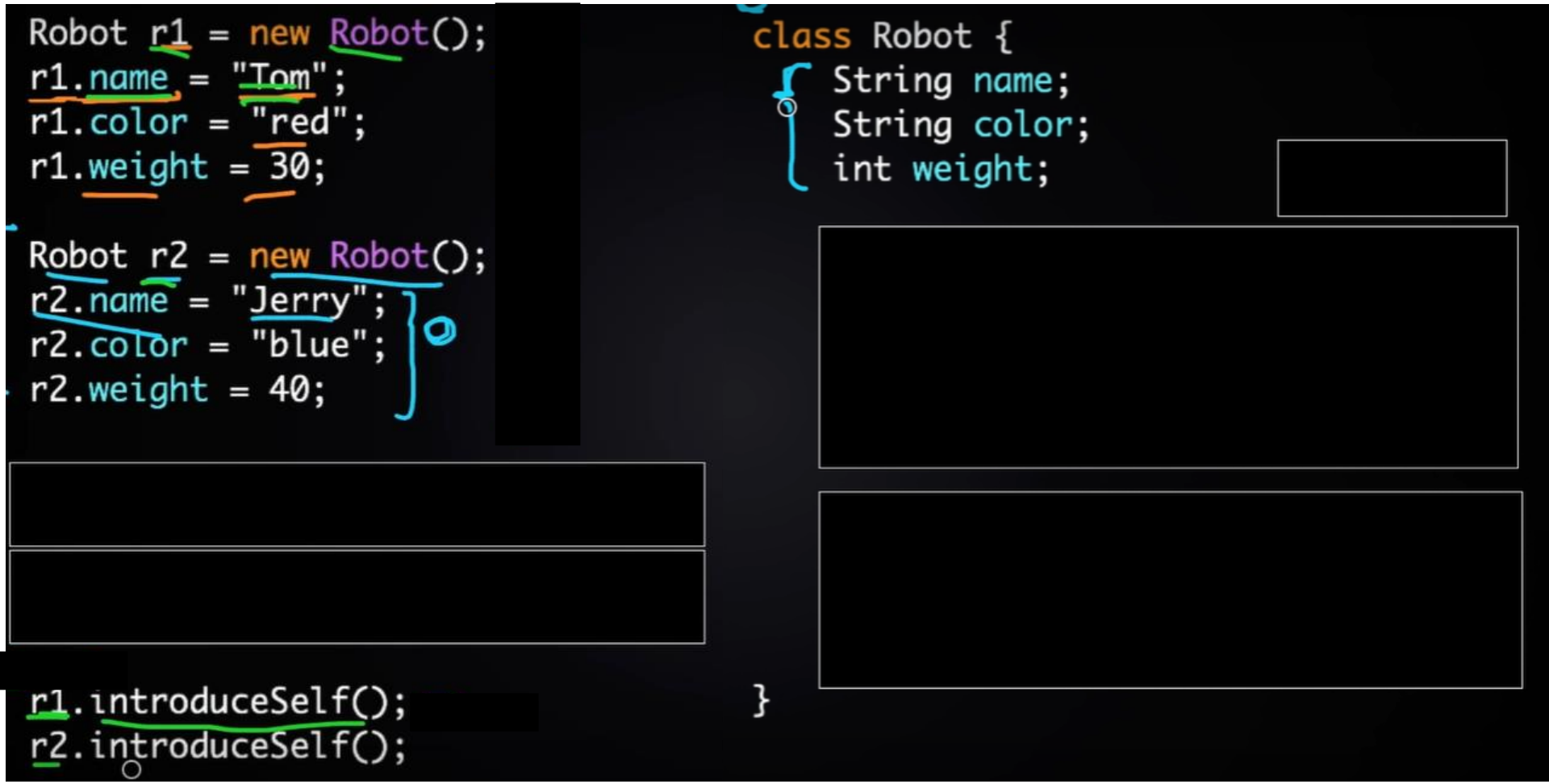
```java
Robot r1 = new Robot();
r1.name = "Tom";
r1.color = "red";
r1.weight = 30;
```

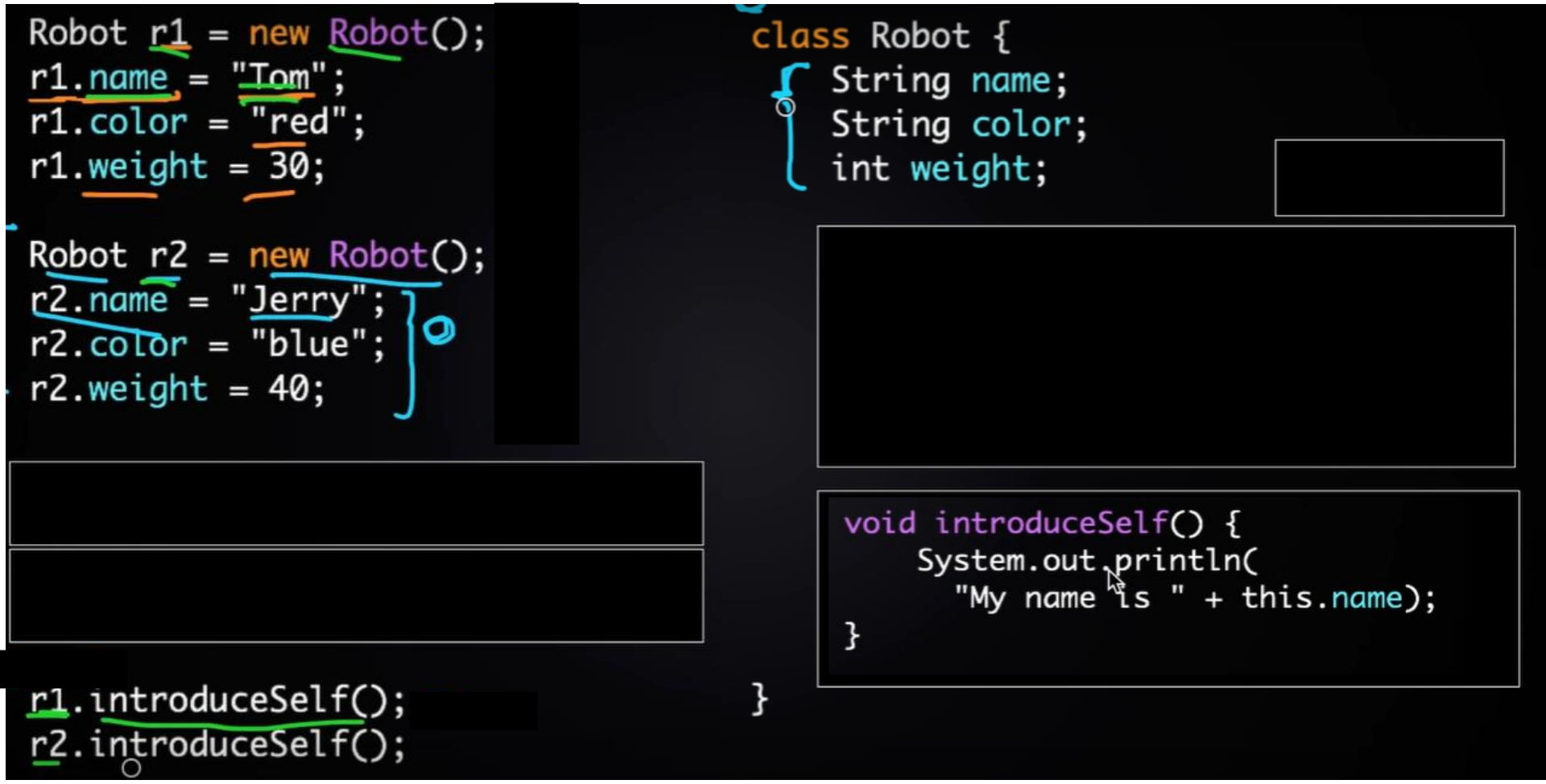
```java
class Robot {
}
```

```
Robot r1 = new Robot();
r1.name = "Tom";
r1.color = "red";
r1.weight = 30;

Robot r2 = new Robot();
r2.name = "Jerry";
r2.color = "blue";
r2.weight = 40;
```

```
class Robot {

}
```


Robot r1 = new Robot();
r1.name = "Tom";
r1.color = "red";
r1.weight = 30;
Robot r2 = new Robot();
r2.name = "Jerry";
r2.color = "blue";
r2.weight = 40;
r1.introduceSelf();
r2.introduceSelf();
class Robot {
}


Robot r1 = new Robot();
r1.name = "Tom";
r1.color = "red";
r1.weight = 30;
Robot r2 = new Robot();
r2.name = "Jerry";
r2.color = "blue";
r2.weight = 40;
r1.introduceSelf();
r2.introduceSelf();
class Robot {
String name;
String color;
int weight;
}


Robot r1 = new Robot();
r1.name = "Tom";
r1.color = "red";
r1.weight = 30;
Robot r2 = new Robot();
r2.name = "Jerry";
r2.color = "blue";
r2.weight = 40;
r1.introduceSelf();
r2.introduceSelf();
class Robot {
String name;
String color;
int weight;
void introduceSelf() {
System.out.println(
"My name is " + this.name);
}
}

```java
Robot r1 = new Robot();
r1.name = "Tom";
r1.color = "red";
r1.weight = 30;

Robot r2 = new Robot();
r2.name = "Jerry";
r2.color = "blue";
r2.weight = 40;

r1.introduceSelf();
r2.introduceSelf();
```

```java
class Robot {
    String name;
    String color;
    int weight;

    Robot(String n, String c, int w) {
        this.name = n;
        this.color = c;
        this.weight = w;
    }

    void introduceSelf() {
        System.out.println(
          "My name is " + this.name);
    }
}
```

constructor

```java
Robot r1 = new Robot();
r1.name = "Tom";
r1.color = "red";
r1.weight = 30;

Robot r2 = new Robot();
r2.name = "Jerry";
r2.color = "blue";
r2.weight = 40;

Robot r1 =
    new Robot("Tom", "red", 30);

r1.introduceSelf();
r2.introduceSelf();
```

```java
class Robot {
    String name;
    String color;
    int weight;

    Robot(String n, String c, int w) {
        this.name = n;
        this.color = c;
        this.weight = w;
    }

    void introduceSelf() {
        System.out.println(
          "My name is " + this.name);
    }
}
```

constructor

```java
Robot r1 = new Robot();
r1.name = "Tom";
r1.color = "red";
r1.weight = 30;

Robot r2 = new Robot();
r2.name = "Jerry";
r2.color = "blue";
r2.weight = 40;

Robot r1 =
    new Robot("Tom", "red", 30);
Robot r2 =
    new Robot("Jerry", "blue", 40);

r1.introduceSelf();
r2.introduceSelf();
```

```java
class Robot {
    String name;
    String color;
    int weight;

    Robot(String n, String c, int w) {
        this.name = n;
        this.color = c;
        this.weight = w;
    }

    void introduceSelf() {
        System.out.println(
          "My name is " + this.name);
    }
}
```

constructor

```java
Robot r1 = new Robot();
r1.name = "Tom";
r1.color = "red";
r1.weight = 30;

Robot r2 = new Robot();
r2.name = "Jerry";
r2.color = "blue";
r2.weight = 40;

Robot r1 =
    new Robot("Tom", "red", 30);
Robot r2 =
    new Robot("Jerry", "blue", 40);

r1.introduceSelf();
r2.introduceSelf();
```

```java
class Robot {
    String name;
    String color;
    int weight;

    Robot(String n, String c, int w) {
        this.name = n;
        this.color = c;
        this.weight = w;
    }

    void introduceSelf() {
        System.out.println(
          "My name is " + this.name);
    }
}
```

constructor

- https://www.youtube.com/watch?v=8yjkWGRlUmY

Εισαγωγή στη JAVA

Java (Εισαγωγή σε μία διαφάνεια…)

- Αναπτύχθηκε από τη Sun Microsystems
  - 1η έκδοση το 1995.
  - 2014 – έκδοση 8 (θα υποστηρίζεται μέχρι και Ιαν 2019)
  - Τελευταία έκδοση για download 9.0.4
  - Μάρτιος 2018 – έκδοση 10
- Δωρεάν διάθεση από http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Υλικό: http://www.oracle.com/technetwork/java/index.html
- Το συντακτικό βασίζεται σε C / C++

- Είναι platform-independent:
  - ο πηγαίος κώδικας μεταγλωττίζεται σε bytecode (ενδιάμεση μορφή εντολών μεταξύ γλώσσας υψηλού επιπέδου και γλώσσας μηχανής)
  - ένα πρόγραμμα σε bytecode μπορεί να εκτελεστεί σε οποιονδήποτε υπολογιστή έχει εγκατεστημένη τη Java Virtual Machine

Τεκμηρίωση

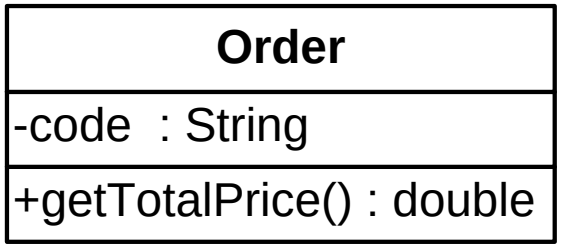
- Εξαιρετική τεκμηρίωση για τη Java από την Oracle:

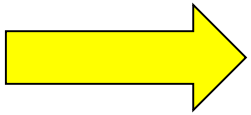
http://docs.oracle.com/javase/tutorial/

- Για οποιαδήποτε κλάση βιβλιοθηκών της Java (Java API) μία αναζήτηση στο Google του τύπου όνομαΚλάσης Java (π.χ. String Java)… συνήθως επιστρέφει ως πρώτο αποτέλεσμα την σελίδα της Oracle με την επίσημη τεκμηρίωση
  - Εκδόσεις API
    - Java SE (Standard Edition): Παρέχει όλη τη βασική λειτουργικότητα της Java για ανάπτυξη εφαρμογών, δικτύωση, ασφάλεια, πρόσβαση σε ΒΔ, GUI.
    - Java EE (Enterprise Edition): παρέχει κλάσεις βιβλιοθήκης (API) για εφαρμογές μεγάλης κλίμακας (multi-tiered, scalable, reliable, secure network applications)
    - Java ME (Micro Edition): Ανάπτυξη εφαρμογών σε φορητές συσκευές
    - JavaFX: Ανάπτυξη ‘ελαφρών’ διαδικτυακών εφαρμογών

Ορισμός κλάσης

- Κάθε κλάση κατά προτίμηση τοποθετείται σε ξεχωριστό αρχείο .java με όνομα το όνομα της κλάσης
- Ορατότητα ιδιοτήτων και μεθόδων
  -: ιδιωτική ορατότητα (private)
  +: δημόσια ορατότητα (public)





```
public class Order {

  private String code;

  public double getTotalPrice()
{
      //σώμα της μεθόδου
      //σχόλιο μιας γραμμής
      /* σχόλιο πολλών γραμμών
         με αυτό το σύμβολο */
}
```

Κατασκευαστές (Constructors)

- Κάθε κλάση διαθέτει μία «ειδική» μέθοδο που καλείται κατά την κατασκευή νέου αντικειμένου της κλάσης (κατασκευαστής – constructor)
- Αν δεν δηλωθεί κατασκευαστής, χρησιμοποιείται ο εξ' ορισμού κατασκευαστής που απλώς δημιουργεί το αντικείμενο χωρίς να αρχικοποιεί τις τιμές των ιδιοτήτων
- Ένας κατασκευαστής πρέπει υποχρεωτικά να έχει το όνομα της κλάσης
- Σε έναν κατασκευαστή με παραμέτρους μπορούμε να αποδώσουμε αρχικές τιμές σε ιδιότητες

Κατασκευαστές (Constructors)

κλάση

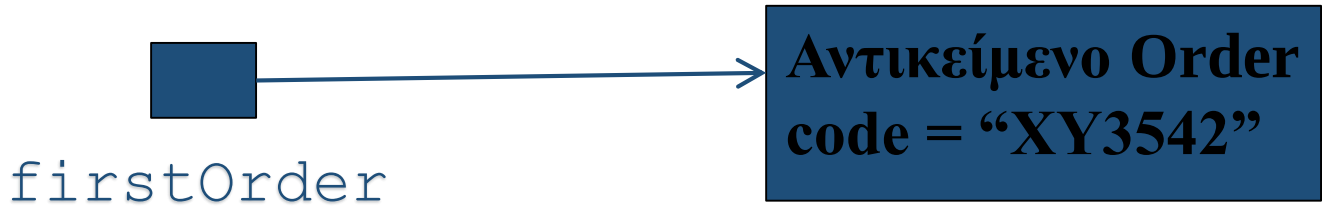
Order
- code : String
+ Order(text : String)

Constructor

```
public class Order {

   private String code;


   public Order(String text)  {

       code = text;

   }
}
```

Δημιουργία Αντικειμένων

- Δημιουργούμε ένα αντικείμενο κλάσης στη Java με χρήση της δεσμευμένης λέξης new και κλήση του κατασκευαστή της κλάσης
- Το νέο αντικείμενο το αναθέτουμε σε μία αναφορά (reference) του τύπου της κλάσης:
- Order firstOrder; **//δημιουργία αναφοράς προς αντικείμενο τύπου Order**
- firstOrder = new Order("XY3542"); **//ανάθεση τιμής στην αναφορά**
- Το firstOrder είναι μία αναφορά τύπου Order. κατ' ουσία πρόκειται για έναν δείκτη (pointer) προς αντικείμενο τύπου Order



- Καταχρηστικά, αναφερόμαστε συνήθως στο ίδιο το αντικείμενο ως firstOrder

Δημιουργία αντικειμένων

- Στη Java, τα πάντα είναι αντικείμενα (και κατά συνέπεια τα χειριζόμαστε μέσω αναφορών σε αυτά)
  - String name = “Markos”;
    το name είναι αναφορά προς αντικείμενο τύπου String
  - int[] anArray = new int[10];
    το anArray είναι αναφορά προς αντικείμενο τύπου int[], δηλ. πίνακας ακεραίων

- … εκτός από 8 Στοιχειώδεις τύπους που δεν είναι αντικείμενα:
  - byte, short, int, long, float, double, boolean, char

Βασικοί τύποι

Όνομα τύπου	Τιμή	Μνήμη	Τιμές
boolean	true/false	1 byte	true, false
char	Χαρακτήρας (Unicode)	2 bytes	Γράμματα, αριθμοί, σημεία στίξης και άλλα σύμβολα
byte	Ακέραιος	1 byte	-128 έως 127
short	Ακέραιος	2 bytes	-32.768 έως 32.767
int	Ακέραιος	4 bytes	-2.147.483.648 έως 2.147.483.647
long	Ακέραιος	8 bytes	-9,223,372,036,854,775,808 έως….
float	Πραγματικός	4 bytes	1,4E-45 έως $3,4^{E}$+38
double	Πραγματικός	8 bytes	4,9E-324 έως 1,7 E+308

Όταν ορίζουμε μια μεταβλητή δεσμεύεται ο αντίστοιχος χώρος στη μνήμη.
Το όνομα της μεταβλητής συνδέεται με αυτό το χώρο στη μνήμη.

Παράδειγμα ορισμού κλάσης

```
class Employee {

    private String name;

    private double salary;


    Employee (String n, double s) {
        name = n;
        salary = s;
    }

    void pay () {
        System.out.println("Pay the employee named " +
                                name + " $" + salary);
    }
    public String getName() { return name; } // getter
}
```

Παράδειγμα ορισμού κλάσης

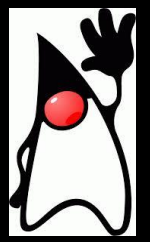
```
class Employee {
    // Πεδία ή ιδιότητες κλάσεις
    private String name;    // Σύμφωνα με τις μεθόδους που ορίζονται μετά, κάποιος εκτός κλάσης μπορεί να τη διαβάσει αλλά όχι αν την αλλάξει
    private double salary;  // Σύμφωνα με τις μεθόδους που ορίζονται μετά, κάποιος εκτός κλάσης δε μπορεί να τη διαβάσει ούτε να την αλλάξει
    // Κατασκευαστής (constructor)
    Employee (String n, double s) {
        name = n;
        salary = s;
    }
    // Μέθοδοι
    void pay () {
        System.out.println("Pay the employee named " +
                                name + " $" + salary);
    }
    public String getName()
            { return name; } // getter
}
```

Δημιουργία αντικειμένων

- Όλα τα προγράμματα Java **πρέπει να περιλαμβάνουν μία μέθοδο main από την οποία ξεκινά η εκτέλεση** του προγράμματος
  - Στη main μπορούμε να δημιουργήσουμε αντικείμενα των υπόλοιπων κλάσεων
- **Η μέθοδος main μπορεί να βρίσκεται σε οποιαδήποτε κλάση. Συνήθως την τοποθετούμε μέσα σε μια κλάση με όνομα Main**

```
public class Main{

   public static void main(String[] args) {

      //Δημιουργία αντικειμένων

   }
}
```

HELLO WORLD!



File HelloWorld.java

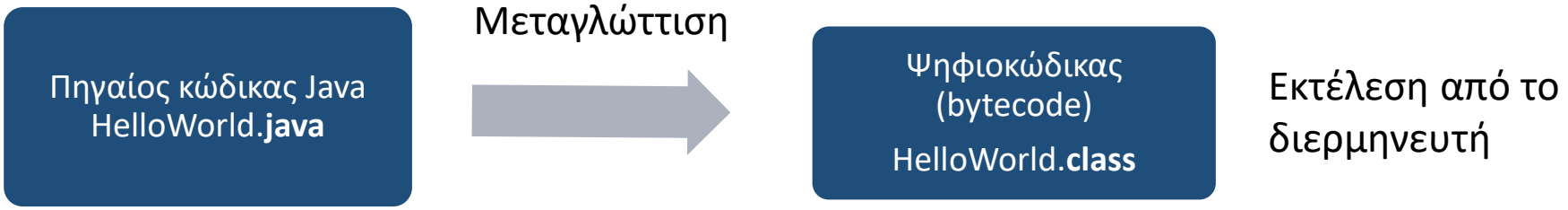
```
class HelloWorld
{
    public static void main(String args[])
    {
            // print message
            System.out.println("Hello world!");
      }
}
```

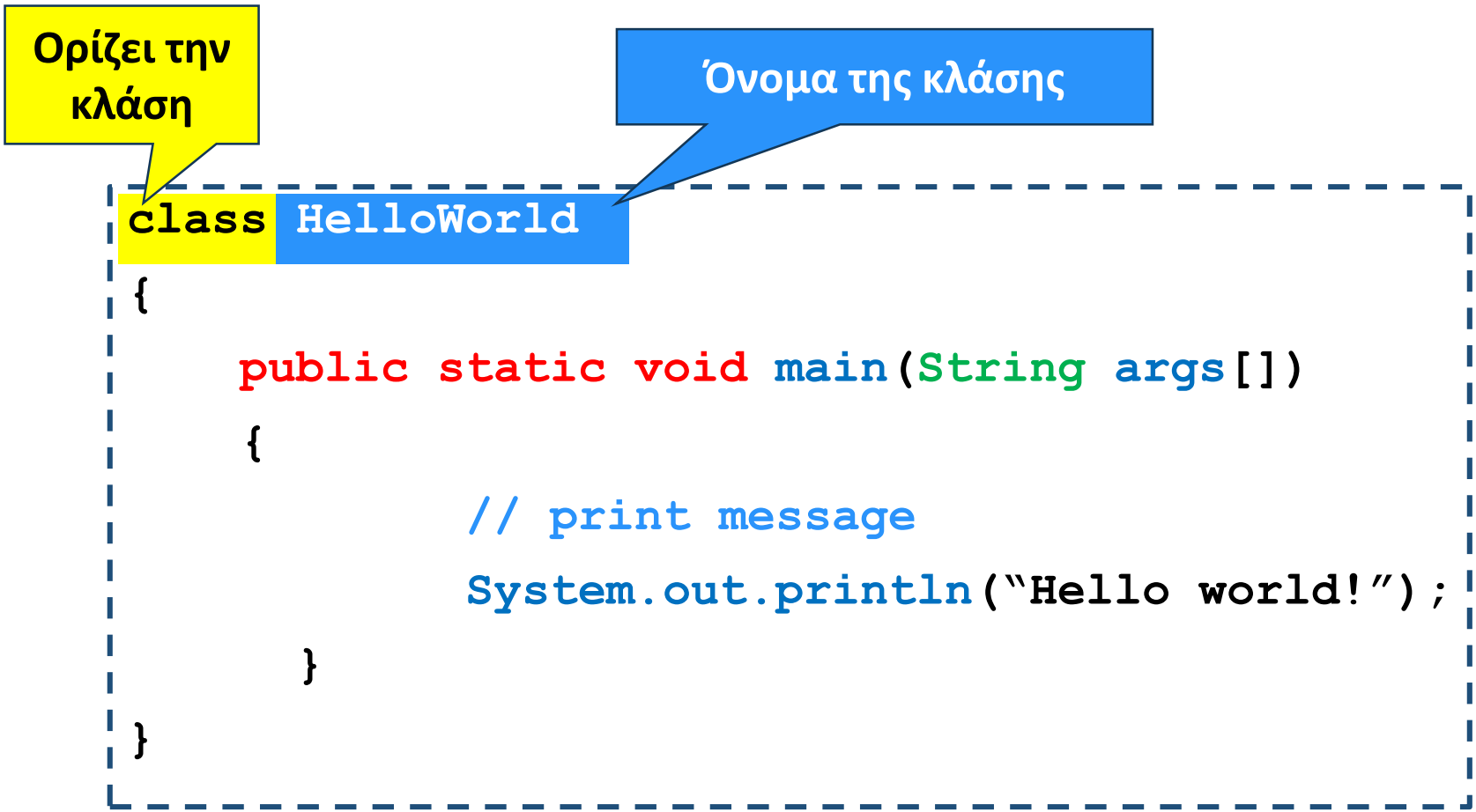
- `javac HelloWorld.java`
- `java HelloWorld`

Χωρίς καμία κατάληξη!

.java και .class

- Η Java είναι αρχιτεκτονικά ουδέτερη γλώσσα (Platform independent)
- Ένα αρχείο .class μπορεί να χρησιμοποιηθεί σε οποιοδήποτε υπολογιστή έχει εγκατεστημένη τη JVM
- Τα αρχεία .class έχουν πολύ μικρό μέγεθος
- Διερμηνευτής: JVM




Ορίζει την κλάση
Όνομα της κλάσης
class HelloWorld
{
public static void main(String args[])
{
// print message
System.out.println("Hello world!");
}
}

```
class HelloWorld
{
    public static void main(String args[])
    {
            // print message
            System.out.println("Hello world!");
    }
}
```

Τα άγκιστρα { … } ορίζουν ένα λογικό block του κώδικα

- Αυτό μπορεί να είναι μία κλάση, μία συνάρτηση, ένα if statement
- Οι μεταβλητές που ορίζουμε μέσα σε ένα λογικό block, έχουν εμβέλεια μέσα στο block

Ορισμός της μεθόδου main

```
class HelloWorld
{
    public static void main(String args[])
    {
            // print message
            System.out.println("Hello world!");
    }
}
```

public, static: θα τα εξηγήσουμε σε επόμενο μάθημα
void: Η μέθοδος δεν επιστρέφει τίποτα
main: σηματοδοτεί το σημείο εκκίνησης του προγράμματος

Ορισμός της μεθόδου main

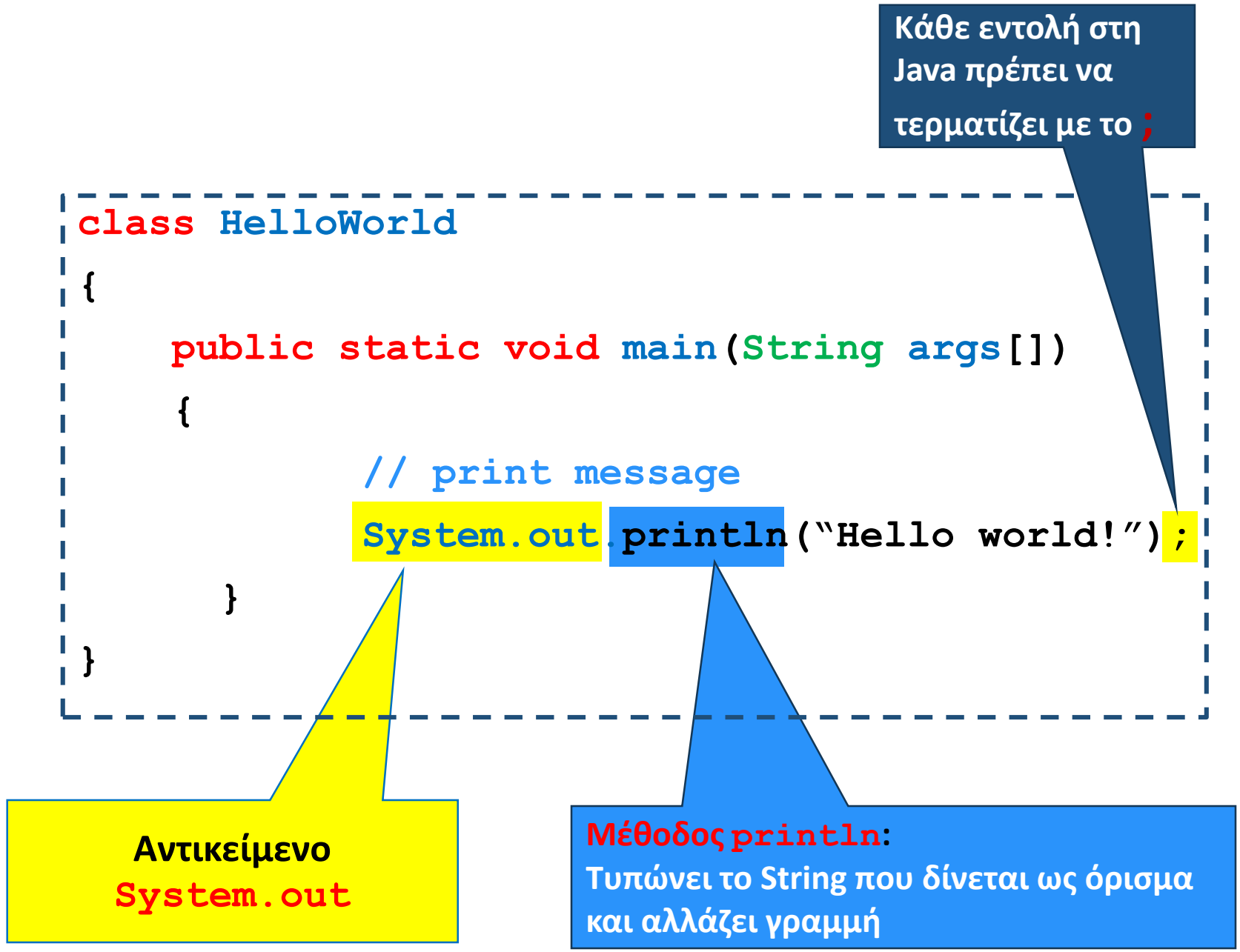
```
/**
* A class that prints a message "hello world"
*/
class HelloWorld
{
    public static void main(String args[])
    {
            // print message
            System.out.println("Hello world!");
    }
}
```

Ορίσματα της μεθόδου

- Ένας πίνακας από Strings που αντιστοιχούν στις παραμέτρους με τις οποίες τρέχουμε το πρόγραμμα.
- String: κλάση βιβλιοθήκης της Java που χειρίζεται τα αλφαριθμητικά

```
/**
* <h1>Hello, World!</h1>
* The HelloWorld program implements an application that
* simply displays "Hello World!" to the standard output.
* <p>
* Giving proper comments in your program makes it more
* user friendly and it is assumed as a high quality code.
*
*
* @author  Zara Ali
* @version 1.0
* @since   2014-03-31
*/
```

```
{
    public static void main(String args[])
    {
            // σχόλιο 1 γραμμής
            System.out.println("Hello world!");
    }
}
```



http://javapapers.com/core-java/system-out-println/

Παράδειγμα

- Φτιάξτε ένα πρόγραμμα που τυπώνει το αποτέλεσμα της διαίρεσης δύο ακεραίων.

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Ορισμός μεταβλητών

- Η Java είναι strongly typed γλώσσα: κάθε μεταβλητή θα πρέπει να έχει ένα τύπο
- Οι τύποι int και double είναι βασικοί τύποι (primitive types)
- Εκτός από τους βασικούς τύπους, όλοι οι άλλοι τύποι είναι κλάσεις

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division  = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Ανάθεση: αποτίμηση της τιμής της έκφρασης στο δεξιό μέλος του "=" και μετά ανάθεση της τιμής στην μεταβλητή στο αριστερό μέλος

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Μετατροπή τύπου: (double)denominator μετατρέπει την τιμή της μεταβλητής denominator σε double.

Αν δεν γίνει η μετατροπή, η διαίρεση μεταξύ ακεραίων μας δίνει πάντα ακέραιο.

Αναθέσεις

- Στην ανάθεση κατά κανόνα, η τιμή του δεξιού μέρους θα πρέπει να είναι ίδιου τύπου με την μεταβλητή του αριστερού μέρους.
- Υπάρχουν εξαιρέσεις όταν υπάρχει συμβατότητα μεταξύ τύπων

- byte → short → int → long → float → double
  - Μια τιμή τύπου T μπορούμε να την αναθέσουμε σε μια μεταβλητή τύπου που εμφανίζεται δεξιά του T

- (Σε αντίθεση με την C) ο τύπος boolean δεν είναι συμβατός με τους ακέραιους

Division.java

```
class Division
{
  public static void main(String args[])
  {
    int enumerator = 32;
    int denominator = 10;
    double division;
    division = enumerator/(double)denominator;
    System.out.println("Result = " + division);
  }
}
```

Ο τελεστής "+" μεταξύ αντικείμενων της κλάσης String συνενώνει (concatenates) τα δύο String.
Μεταξύ ενός String και ενός βασικού τύπου, ο βασικός τύπος μετατρέπεται σε String και γίνεται η συνένωση

Αλφαριθμητικά (strings)

- Η κλάση String είναι προκαθορισμένη κλάση της Java που μας επιτρέπει να χειριζόμαστε αλφαριθμητικά.
- Ο τελεστής "+" μας επιτρέπει την συνένωση
- Υπάρχουν πολλές χρήσιμες μέθοδοι της κλάσης String.
  - length(): μήκος του String
  - equals(String x): ελέγχει για ισότητα του αντικειμένου που κάλεσε την μέθοδο και του ορίσματος x.
  - trim(): αφαιρεί κενά στην αρχή και το τέλος του string.
  - Μέθοδοι για να βρεθεί ένα υπο-string μέσα σε ένα string.
  - κ.α.

Έξοδος στην οθόνη

- Μπορούμε να καλέσουμε τις μεθόδους του αντικειμένου System.out:
  - println(String s): για να τυπώσουμε ένα αλφαριθμητικό s και τον χαρακτήρα '\n' (αλλαγή γραμμής)
  - print(String s): τυπώνει το s αλλά δεν αλλάζει γραμμή
  - printf: Formatted output
    - printf("%d",myInt); // τυπώνει ένα ακέραιο
    - printf("%f",myDouble); // τυπώνει ένα πραγματικό
    - printf("%.2f",myDouble); // τυπώνει ένα πραγματικό με δύο δεκαδικά

Είσοδος από το πληκτρολόγιο

- Χρησιμοποιούμε την κλάση Scanner της Java
  - import java.util.Scanner;
- Αρχικοποιείται με το ρεύμα εισόδου:
  - Scanner in = new Scanner(System.in);
- Μπορούμε να καλέσουμε μεθόδους της Scanner για να διαβάσουμε κάτι από την είσοδο
  - nextLine(): διαβάζει μέχρι να βρει τον χαρακτήρα '\n'
  - next(): διαβάζει το επόμενο String
  - nextInt(): διαβάζει τον επόμενο int
  - nextDouble(): διαβάζει τον επόμενο double.

Παράδειγμα

```
import java.util.Scanner;


class TestIO
{
	public static void main(String args[])
	{
		Scanner input = new Scanner(System.in);
		String line = input.nextLine();
		System.out.println(line);
	}
}
```

new: δημιουργεί ένα αντικείμενο τύπου `Scanner` με το οποίο μπορούμε πλέον να διαβάζουμε από την είσοδο

Division.java με είσοδο από το χρήστη

```
import java.util.Scanner;
class Division {
  public static void main(String args[])   {
// Δημιουργεί το Scanner για να πάρει είσοδο από την κονσόλα
        Scanner input = new Scanner(System.in)
        int enumerator;
        int denominator;
        double division;
        System.out.println("Enter first integer:");
        enumerator = input.nextInt();
        System.out.println("Enter second integer:");
        denominator = input.nextInt();

        division = enumerator/(double)denominator;
        System.out.println("Result = " + division);
  }
}
```

Μέχρι στιγμής τίποτα αντικειμενοστραφές!

- Δεν κατασκευάσαμε κλάσεις και αντικείμενα
- Αν και χρησιμοποιήσαμε κάποιες κλάσεις βιβλιοθήκης της Java (String, Scanner) και κάποιες ‘έτοιμες’ μεθόδους
- Θα δούμε ένα παράδειγμα κώδικα σε δύο εκδοχές
  - μια μη αντικειμενοστρεφή
  - μια αντικειμενοστρεφή

Παράδειγμα

- Ζητούμενη λειτουργικότητα
  - Δίνουμε σαν είσοδο χαρακτηριστικά προϊόντων
    - Όνομα
    - Τιμή
    - Σκορ (σαν αξιολόγηση)
  - Αυτό γίνεται μέχρι να δηλώσουμε ότι δεν έχουμε άλλο προϊόν να εισάγουμε
  - Το πρόγραμμα υπολογίζει από όλα τα προϊόντα το καλύτερο (αυτό με τον μεγαλύτερο λόγο σκορ/τιμή)
  - Εμφανίζει τα στοιχεία του προϊόντος αυτού