

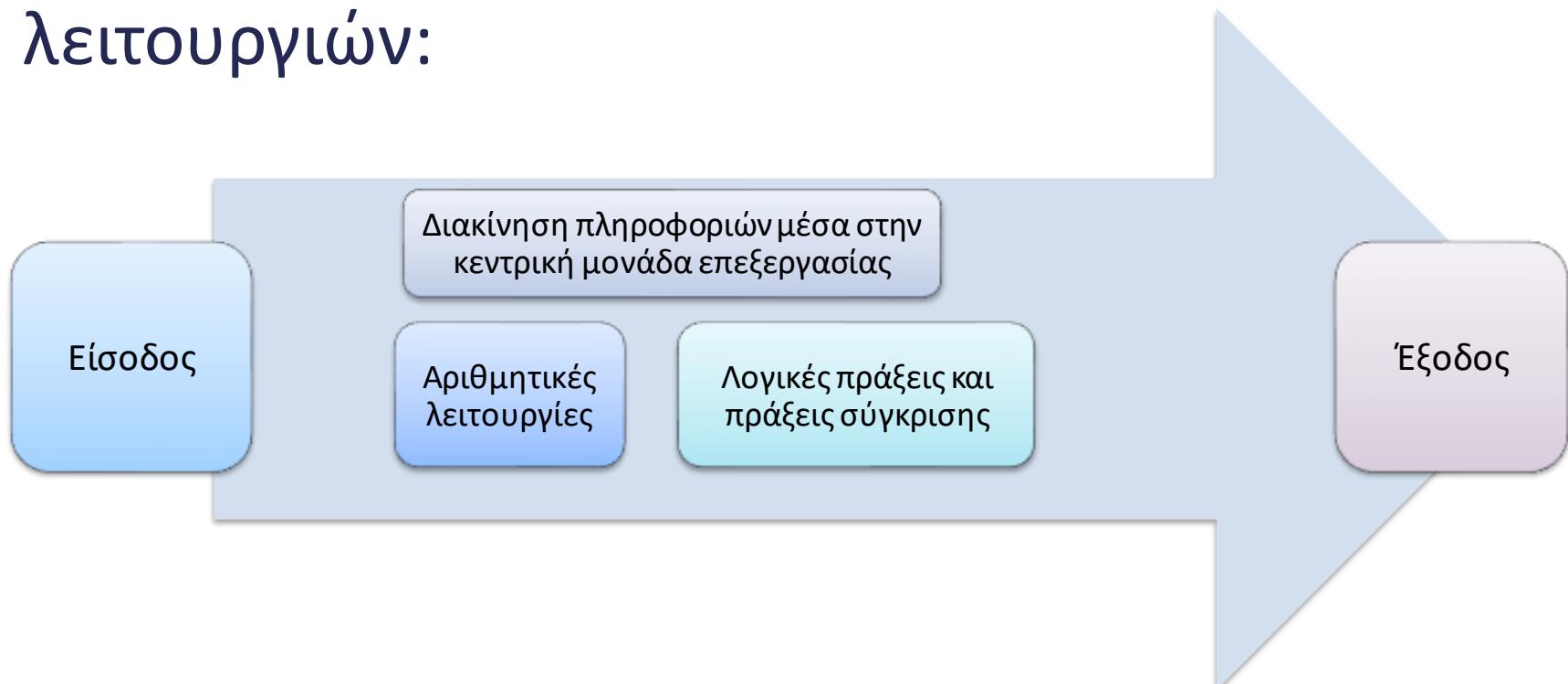
2α. Εισαγωγή στον Αντικειμενοστραφή Προγραμματισμό

1. Βασικές έννοιες του Προγραμματισμού
2. Γενιές γλωσσών προγραμματισμού
3. Είδη Προγραμματισμού
4. Αντικειμενοστραφής προγραμματισμός



Προγραμματισμός Η/Υ

Βασίζεται στην επίλυση προβλημάτων μέσω της συνδυασμένης χρήσης τεσσάρων θεμελιωδών λειτουργιών:



Βασικές έννοιες προγραμματισμού

Πρόγραμμα – Ένα σύνολο εντολών που καθοδηγούν τον υπολογιστή στην εκτέλεση συγκεκριμένων λειτουργιών.

Γλώσσα Προγραμματισμού Υψηλού Επιπέδου – Γλώσσα προγραμματισμού το σύνολο των συμβόλων της οποίας προσιδιάζει στην φυσική (ανθρώπινη) γλώσσα.

Συντακτικό – Οι κανόνες που καθορίζουν τον τρόπο συνδυασμού των διαφόρων συμβόλων της γλώσσας προγραμματισμού έτσι ώστε να σχηματιστεί το πρόγραμμα.

Γλώσσα Μηχανής – Γλώσσα προγραμματισμού το σύνολο των συμβόλων της οποίας είναι άμεσα κατανοητό από τον επεξεργαστή του υπολογιστή.

Γενιές γλωσσών προγραμματισμού

Πρώτη γενιά:
Γλώσσες μηχανής
(0,1)

Δεύτερη γενιά:
Συμβολικές
γλώσσες (get,
store)

γλώσσες «χαμηλού επιπέδου»
(low level languages)

Τρίτη γενιά:
Γλώσσες υψηλού
επιπέδου (Pascal,
C, ADA, LOGO,
COBOL)

- *Procedural*
- *Interpreters & Compilers*

Τέταρτη γενιά:
Ειδικές γλώσσες
για την
εξυπηρέτηση
συγκεκριμένων
αναγκών (SQL)

- *Object-Oriented*
- *Queries*

5^η Γενιά;

Επεξεργασία κώδικα υψηλού επιπέδου

Μεταγλωττιστής (Compiler)

Μετατροπή από κώδικα υψηλού επιπέδου σε γλώσσα μηχανής. Εκτέλεση του προγράμματος που προκύπτει.

Διερμηνευτής (Interpreter)

Ο κώδικας υψηλού επιπέδου εκτελείται γραμμή προς γραμμή κατά τη διάρκεια εκτέλεσης του προγράμματος.

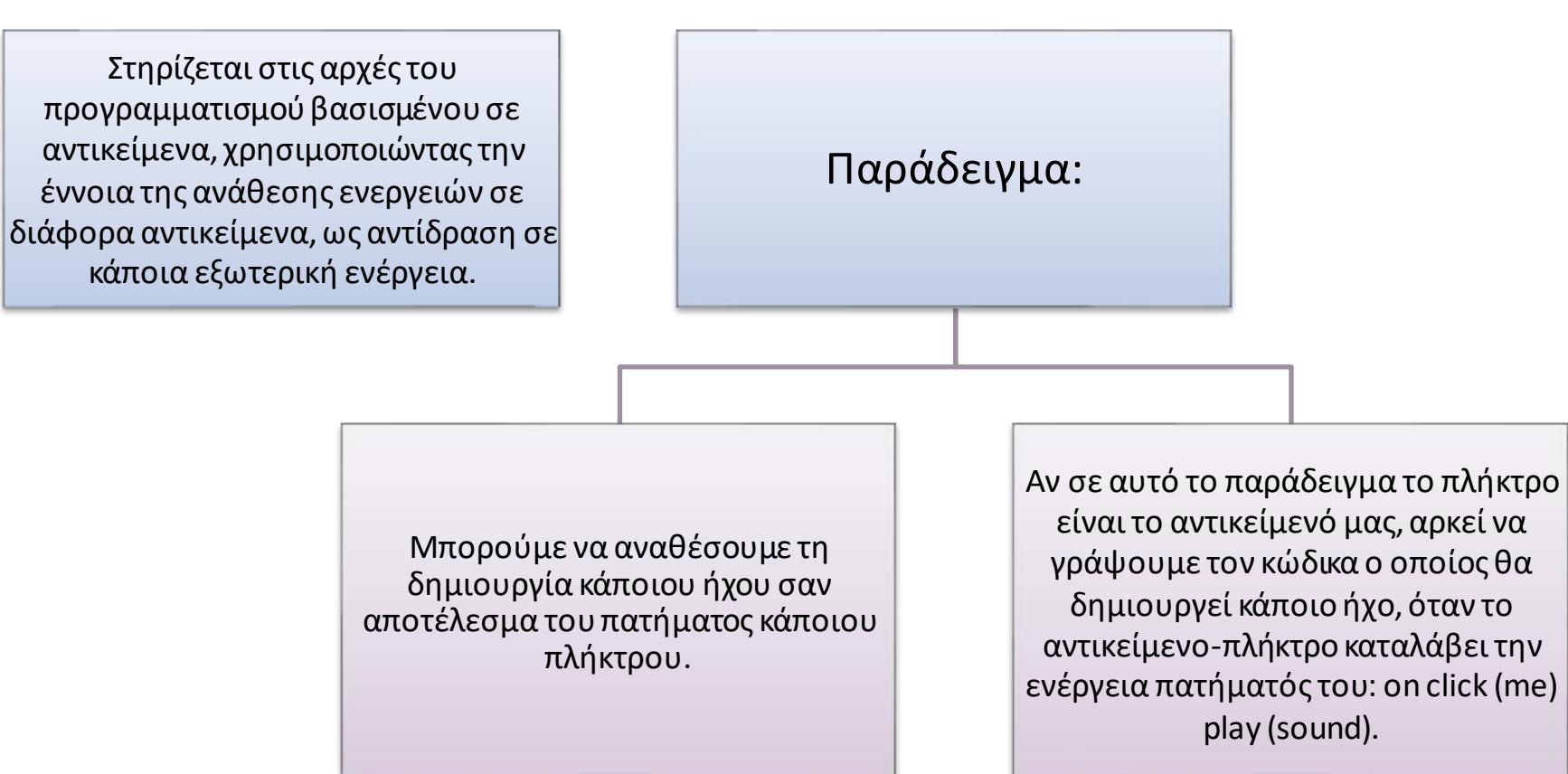
Διαδικασιακός (δομημένος) Προγραμματισμός

Δημιουργία μιας **διαδικασίας** ή αλλιώς ενός συνόλου εργασιών οι οποίες δίνονται με καθορισμένο τρόπο και σειρά στο πρόγραμμα για την εκπλήρωση ενός συγκεκριμένου στόχου.

Μια διαδικασία, από τη στιγμή που θα περιγραφεί μπορεί να χρησιμοποιηθεί και σαν **τμήμα** μιας άλλης διαδικασίας, χτίζοντας τελικά ένα σύνολο ενεργειών το οποίο υλοποιεί τον αλγόριθμο του προγράμματος που θέλουμε να δημιουργήσουμε.

Οι περισσότερες γλώσσες προγραμματισμού που χρησιμοποιούνται σήμερα για εφαρμογές γενικού τύπου είναι **γλώσσες διαδικασιακές**.

Προγραμματισμός βασισμένος σε γεγονότα (Event Driven Programming)



Αντικειμενοστραφής Προγραμματισμός (Object Oriented Programming)

Βασίζεται στη λογική ότι ένα πρόγραμμα είναι ένα σύνολο από **αντικείμενα** τα οποία επικοινωνούν και παράγουν τα επιδιωκόμενα αποτελέσματα.

Τα αντικείμενα αυτά έχουν δύο βασικά στοιχεία: **μεθόδους** και **ιδιότητες** (δεδομένα).

Η δημιουργία των προγραμμάτων βασίζεται στην εύρεση ή δημιουργία των **κατάλληλων** αντικειμένων και του ορισμού του τρόπου **λειτουργίας** και **αλληλεπίδρασης** μεταξύ τους για την επίτευξη του στόχου του προγράμματος.

Με τη χρήση των αντικειμένων υπάρχει η δυνατότητα για την **διανομή** των αντικειμένων και την **επαναχρησιμοποίησή** τους σε άλλα προγράμματα, διευκολύνοντας έτσι τη διαδικασία του προγραμματισμού.

Αντικειμενοστραφής Προγραμματισμός

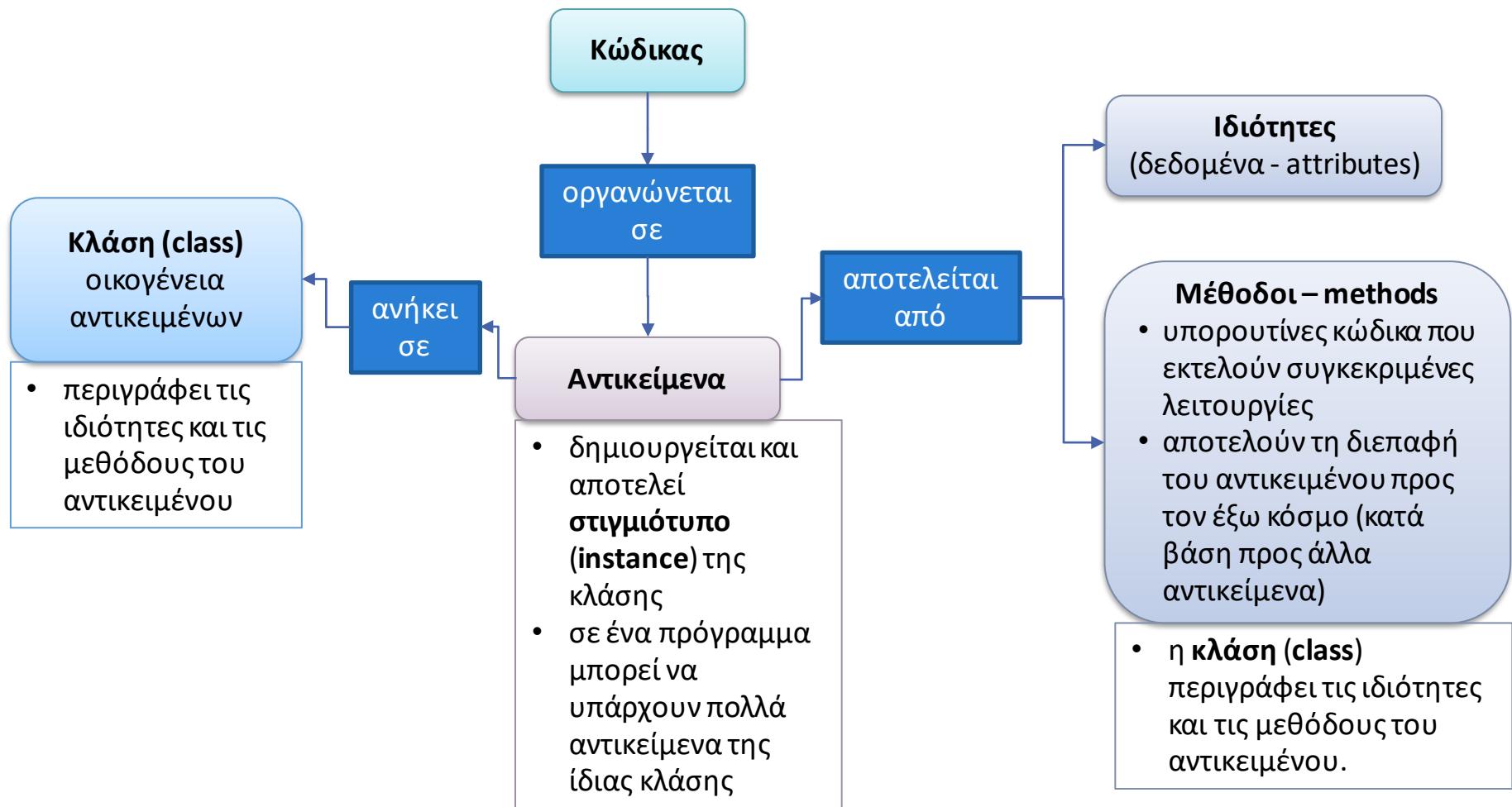
Τι σημαίνει;

- Ο κώδικας είναι οργανωμένος σε κλάσεις αντικειμένων.
- Δημιουργία αντικειμένων βασισμένων στις παραπάνω κλάσεις.
- Δεν υπάρχουν:
 - Μεμονωμένες συναρτήσεις.
 - Μεμονωμένα τμήματα κώδικα.
- Η εκτέλεση του προγράμματος πραγματοποιείται με αρχικοποίηση αντικειμένων που ανταλλάσσουν μηνύματα (δυνατότητα για προγραμματισμό βασισμένο σε γεγονότα= event driven).

Αντικειμενοστραφείς γλώσσες προγραμματισμού

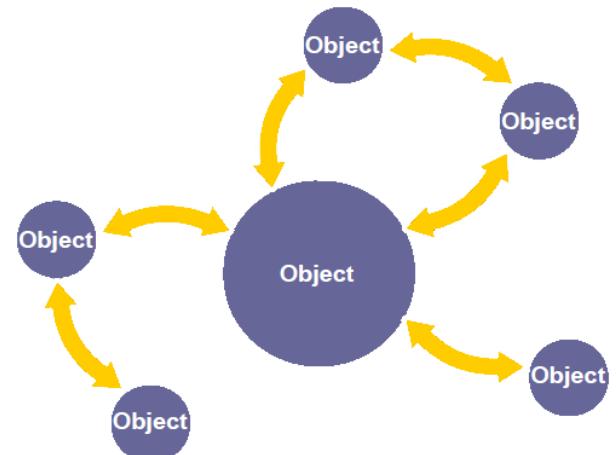


Αντικείμενα, Κλάσεις, Μέθοδοι, Ιδιότητες



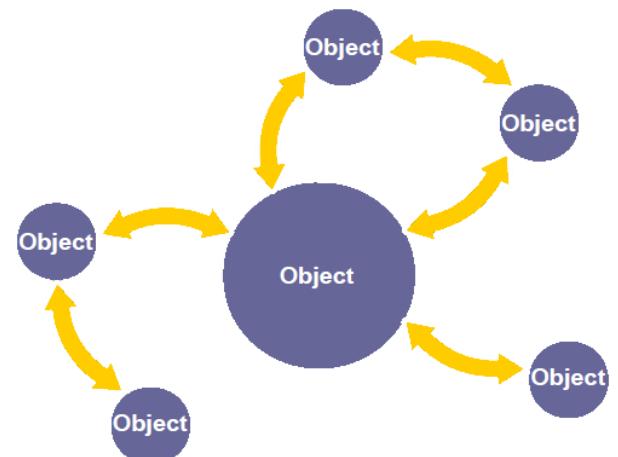
Διαδικαστικές Vs Αντικειμενοστραφείς

- Διαδικαστικές Γλώσσες
 - Διαβάζουν μία γραμμή τη φορά
 - η γλώσσα C είναι διαδικαστική
- Αντικειμενοστραφείς Γλώσσες
 - Διαβάζουν μία γραμμή τη φορά
 - Μοντελοποιούν αντικείμενα μέσω κώδικα
 - Δίνουν έμφαση στην αλληλεπίδραση αντικειμένων
 - Επιτρέπουν την αλληλεπίδραση χωρίς προκαθορισμένη σειρά
 - Η Java και η C ++ είναι αντικειμενοστραφείς γλώσσες



Προγραμματίζοντας με αντικείμενα

- Όταν έχετε μια ιδέα ή απαίτηση για ένα πρόγραμμα ...
 - Εξετάστε το είδος των αντικειμένων που μπορεί να υπάρχουν σε αυτό το πρόγραμμα.
 - Εξετάστε τις ιδιότητες και τις συμπεριφορές αυτών των τύπων αντικειμένων.
 - Εξετάστε πως αλληλεπιδρούν τα αντικείμενα.



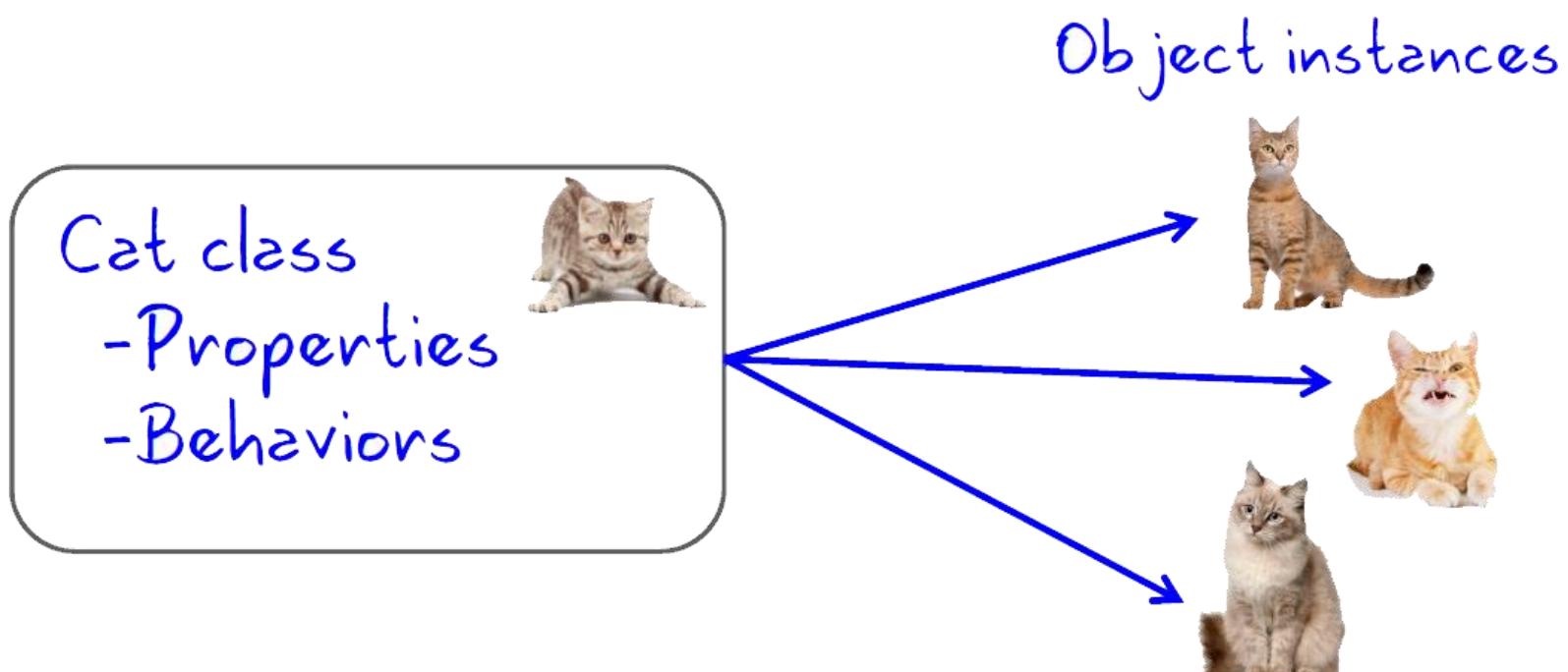
Παράδειγμα

- Ιδιότητες:
 - Όνομα
 - Ηλικία
 - Ράτσα
 - Αγαπημένο φαγητό
- Συμπεριφορές:
 - νιαούρισμα
 - Παίζω
 - Πλύση
 - Τρώω
 - Κυνήγι



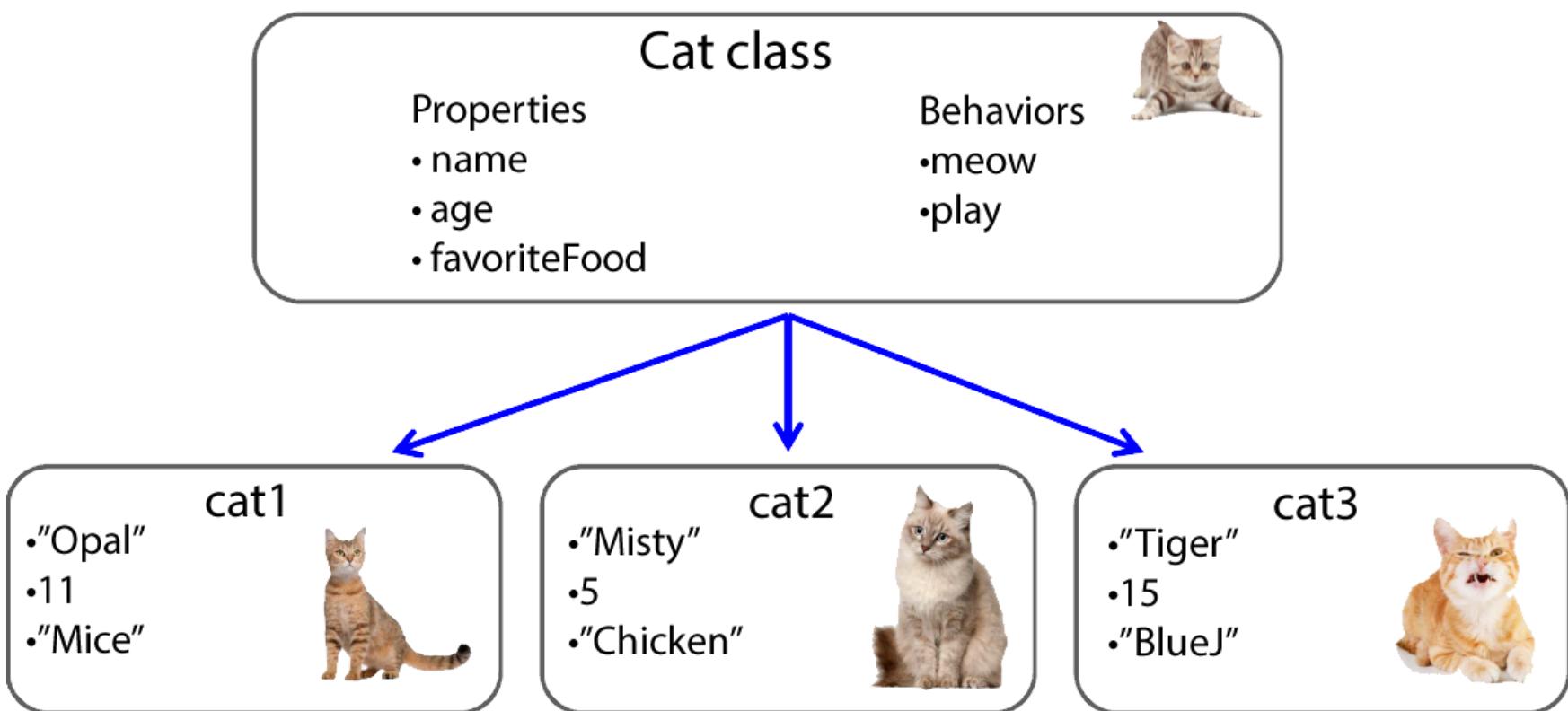
Κλάσεις (Classes) & Στιγμιότυπα (Instances)

- Κλάση = Ιδιότητες (properties) + Συμπεριφορές (behaviors)
- Η κλάση χρησιμοποιείται ως **Σχέδιο** ή **Συνταγή** ή «**Καλούπι**» για τη δημιουργία στιγμιότυπων αντικειμένων.



Κλάσεις (Classes) & Στιγμιότυπα (Instances)

- Όλα τα στιγμιότυπα «γάτας» μπορούν να νιαουρίζουν, παιζουν, τρώνε.



Μεθοδολογία ανάπτυξης προγραμμάτων σε αντικειμενοστραφές περιβάλλον

Μελετάμε το
πρόβλημα

Το αποσυνθέτουμε σε
κλάσεις που αντιστοιχούν
σε οντότητες του
πραγματικού κόσμου

- Επαναχρησιμοποίηση
έτοιμων κλάσεων
- Κάθε κλάση έχει μεθόδους
οι οποίες καλούν μεθόδους
άλλων κλάσεων.

Δημιουργούμε
αντικείμενα ανάλογα
με το πρόγραμμα που
θέλουμε να
αναπτύξουμε και τα
βάζουμε να
αλληλεπιδρούν

Χαρακτηριστικά Αντικειμενοστραφών Γλωσσών

Αφαίρεση (Abstraction)

- Ο προγραμματιστής δουλεύει σε υψηλό επίπεδο, με λιγότερες λεπτομέρειες και σαφώς καθορισμένες διεπαφές

Ενθυλάκωση (Encapsulation)

- Προστασία των δεδομένων και των μεθόδων

Κληρονομικότητα (Inheritance)

- Επιτρέπει την δημιουργία νέων κλάσεων βάσει προηγουμένων

Πολυμορφισμός (Polymorphism)

- Χρήσιμη ιδιότητα που πρέπει να υπάρχει σε κάθε αντικειμενοστραφή γλώσσα και επιτρέπει την αλλαγή της συμπεριφοράς κλάσεων

Οφέλη αντικειμενοστραφούς προγραμματισμού

- Ευκολότερη συντήρηση του κώδικα.
- Βελτίωση της αξιοπιστίας.
 - Άλλαγές στην εσωτερική δομή μιας μεθόδου δεν επηρεάζουν τυχόν άλλες κλάσεις που τυχόν την χρησιμοποιούν.
 - Ομοίως αλλαγές στην εσωτερική δομή ενός αντικειμένου επηρεάζουν μόνο το εν λόγω αντικείμενο.
- Περισσότερες δυνατότητες για επαναχρησιμοποίηση κώδικα.
- Δυνατότητα για εργασία σε ομάδες.
 - Δεν απαιτείται η μια ομάδα ανάπτυξης να ξέρει τη δομή των κλάσεων που δημιουργούνται από μια άλλη ομάδα. Αρκεί να ξέρει ποιες μεθόδους να καλέσει.

Ενότητα 2β: Εισαγωγή στη Java

1. Ιστορία Java
2. Προϊόντα & Τεχνολογίες Java
3. Platform-Dependent και Cross-Platform Independent γλώσσες
4. Java Runtime Environment (JRE)
5. Java Virtual Machine (JVM)
6. Java Development Kit (JDK)
7. Integrated Development Environment (IDE)
8. Διαδικασίας Εκτέλεσης προγράμματος Java
9. Ορολογία-λεξικό Java



Γιατί Java;

- Παγκόσμιο πρότυπο για την ανάπτυξη «ενσωματωμένων» και «φορητών» εφαρμογών, παιχνιδιών, περιεχόμενο Ιστού αλλά και επιχειρησιακό λογισμικό.
 1. Object-oriented
 2. Cross-platform
 3. Internet/Web based
 4. Free-Open Source
 5. Reusable



Java@Desktops

- Java ~ 1,1 δις
- JRE ~ 930 εκ. (downloads/year)
- JDK ~ 9,5 εκ. (downloads/year)



Java@Mobiles

- ~3 εκ. συσκευές



Java TV & Card

- 100% blu-ray players



#1 Development Platform in the Cloud

~30 Χρόνια Java

- 1995: Πρώτη έκδοση της Java



Ιστορία

- Ξεκίνησε το 1990 με το όνομα OAK από την εταιρεία Sun Microsystems
 - Διασύνδεση συσκευών με διαφορετικές CPU
 - Σχεδιάστηκε έχοντας κατά νου τη διαδραστική τηλεόραση!
 - Αρχηγός της ομάδας ανάπτυξης (Green Team) ήταν ο James Gosling
- Η έναρξη του WWW της έδωσε μεγάλη ώθηση
 - Ανάπτυξη Web Multimedia Components για ιστοσελίδες (Java Applets - small applications)
- 1995 → Java
 - Ενσωματώθηκε στον Netscape Navigator
- "Write Once, Run Anywhere" (WORA)
- Το όνομα προέρχεται:
 - Από μια ποικιλία καφέ που πουλούσε η τοπική καφετέρια (?)
 - Από κάποια από τα ονόματα των δημιουργών της γλώσσας (?): (James Gosling, Arthur Van Hoff, Andy Bechtolsheim)
- Πέρασε στην κυριότητα της Oracle το 2010



James Gosling
The Father of Java

Java version history

Release	Year
JDK Beta	1995
JDK 1.0	1996
JDK 1.1	1997
J2SE 1.2	1998
J2SE 1.3	2000
J2SE 1.4	2002
J2SE 5.0	2004
Java SE 6	2006
Java SE 7	2011
Java SE 8	2014
Java SE 9	2017
Java SE 10 (18.3)	2018


ORACLE

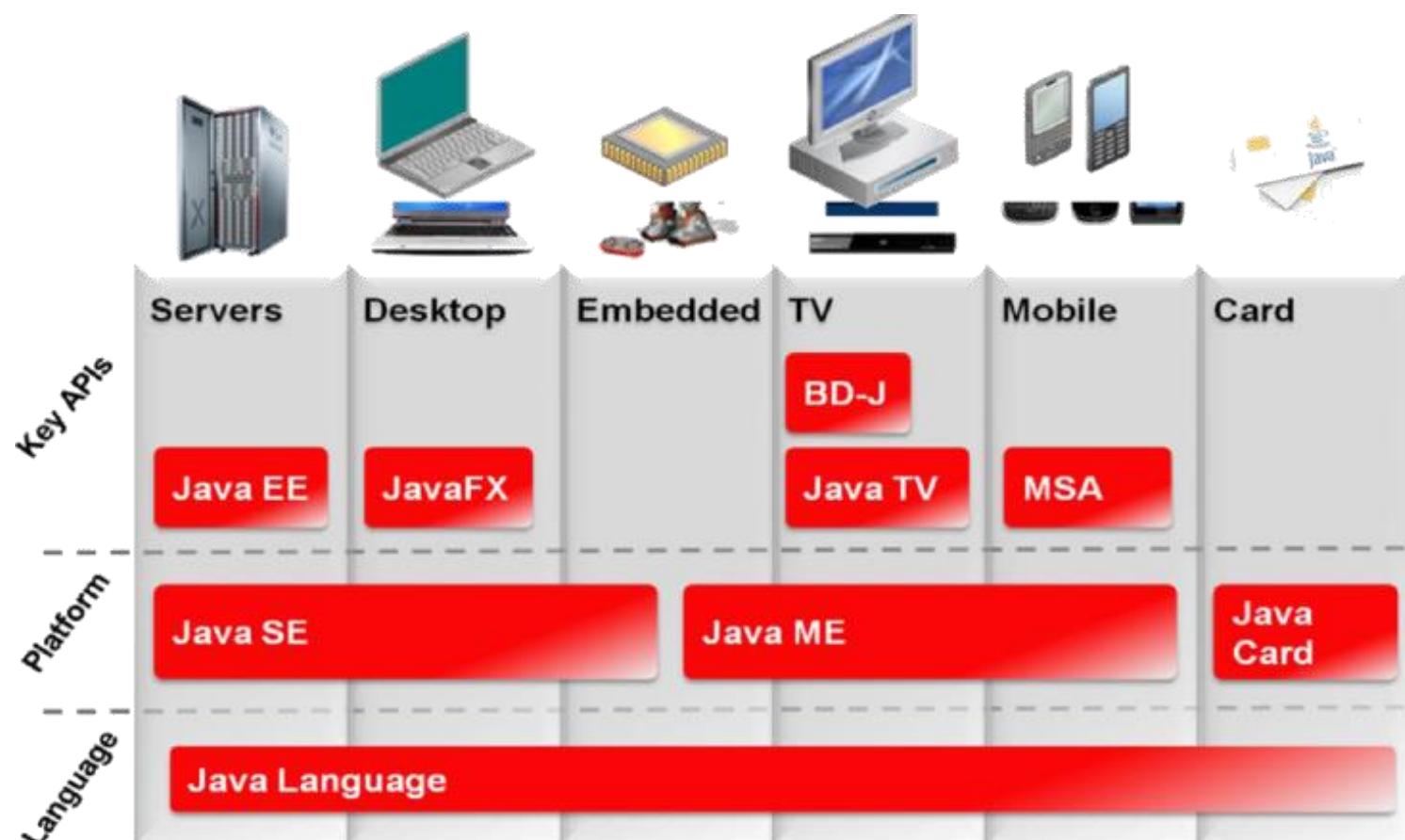
Duke
the Java Maskot

Source: https://en.wikipedia.org/wiki/Java_version_history

Ομάδες προϊόντων Java

1. Java Platform, Standard Edition (Java SE): Για desktops
2. Java Platform, Enterprise Edition (Java EE): Για μεγάλες επιχειρησιακές server-side/client-side κατανεμημένες εφαρμογές π.χ. για e-commerce εταιρίας
3. Java Platform, Micro Edition (Java ME): Για κινητά, PDAs, Smart TV, Smart cards, Raspberry Pi, κ.α.
4. Java Card: Για έξυπνες κάρτες (ταυτότητα, ασφάλεια, συναλλαγές, κάρτες SIM)

Συσκευές & Προϊόντα Java



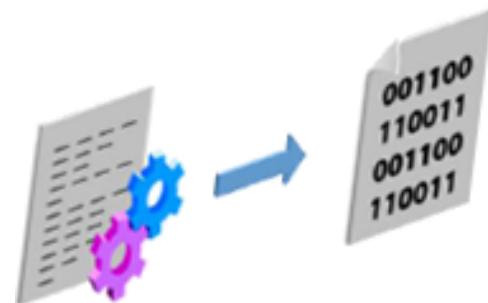
Notes: BD-J: Blu-ray Disk Java

MSA: Mobile Service Architecture

Java Programming

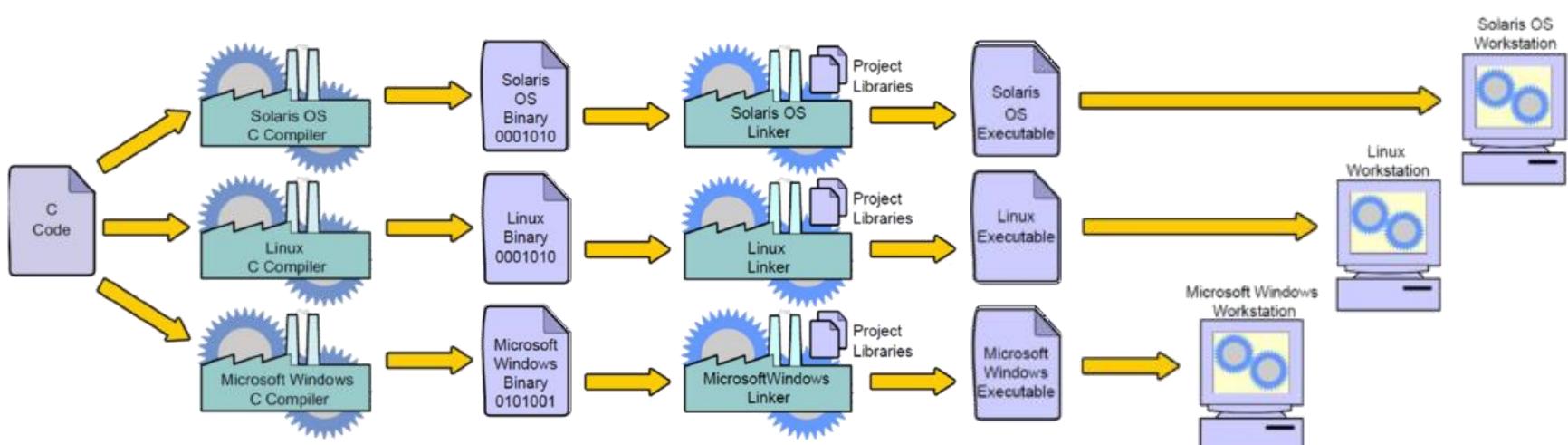
Τι είναι ένα πρόγραμμα;

- Σύνολο εντολών το οποίο τρέχει σε έναν Η/Υ ή οποιαδήποτε άλλη ψηφιακή συσκευή
 - Σε επίπεδο μηχανής → δυαδικές εντολές (11001010001,...)
(Κώδικας μηχανής)
- Τα περισσότερα προγράμματα είναι γραμμένα σε γλώσσα υψηλού επιπέδου (high-level code)
→ Πρέπει να μεταφραστούν σε γλώσσα μηχανής



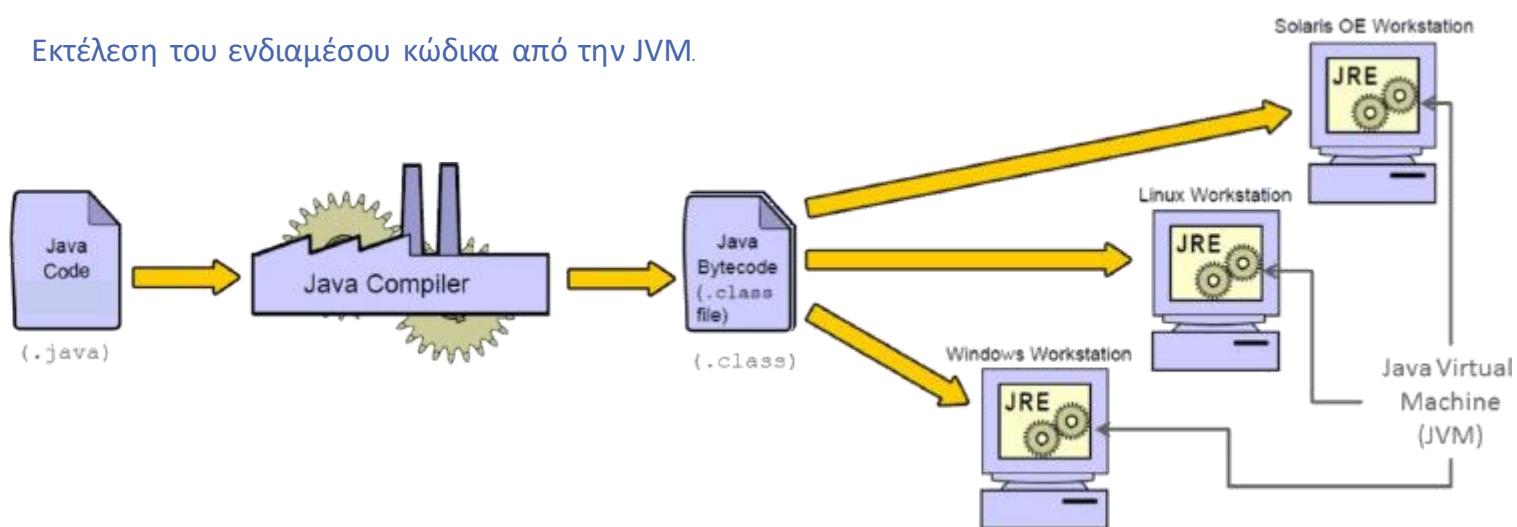
Platform-dependent γλώσσες

- Συνήθως οι γλώσσες (π.χ. C++) απαιτούν μεταγλώττιση (compiling) και σύνδεση (linking) του προγράμματος μέσω ενός εκτελέσιμου προγράμματος (executable) το οποίο εξαρτάται από την εκάστοτε πλατφόρμα (platform-dependent)



Java: Cross-Platform Independent γλώσσα

- Η Java εκτελείται σε έναν υποθετικό υπολογιστή που ονομάζεται Java Virtual Machine (JVM)
- Βήματα δημιουργίας και εκτέλεσης προγραμμάτων στη Java
 - Συγγραφή κώδικα.
 - Μεταγλώττιση σε ενδιάμεσο κώδικα (bytecode).
 - Εκτέλεση του ενδιαμέσου κώδικα από την JVM.



Java Runtime Environment (JRE)



- Η Java Virtual Machine (JVM):
 - Διερμηνεύει τον κώδικα Java
 - Φορτώνει τις κλάσεις Java (classes)
 - Εκτελεί τα προγράμματα Java
- Ένα πρόγραμμα Java, χρειάζεται επιπλέον ένα σύνολο από στάνταρ Java class βιβλιοθήκες (libraries)

**Java
Runtime
Environment
(JRE)**

Java Runtime Environment (JRE)



- Περιλαμβάνει
 - Java Virtual Machine (JVM)
 - Java class libraries

- Σκοπός
 - Διαβάζει bytecode (.class)
 - Τρέχει τον ίδιο κώδικα (bytecode) παντού με JVM

Java Development Kit (JDK)



- Περιλαμβάνει
 - JRE 
 - Java Compiler
 - Πρόσθετα εργαλεία
- Σκοπός
 - Μεταγλωττίζει (compile) bytecode (.java → .class)

Integrated Development Environment (IDE)



- Παρέχει
 - Εξειδικευμένο επεξεργαστή κειμένου (text editor)
 - Εντοπισμό σφαλμάτων (debugging)
 - Διαχείριση Projects
- Παραδείγματα



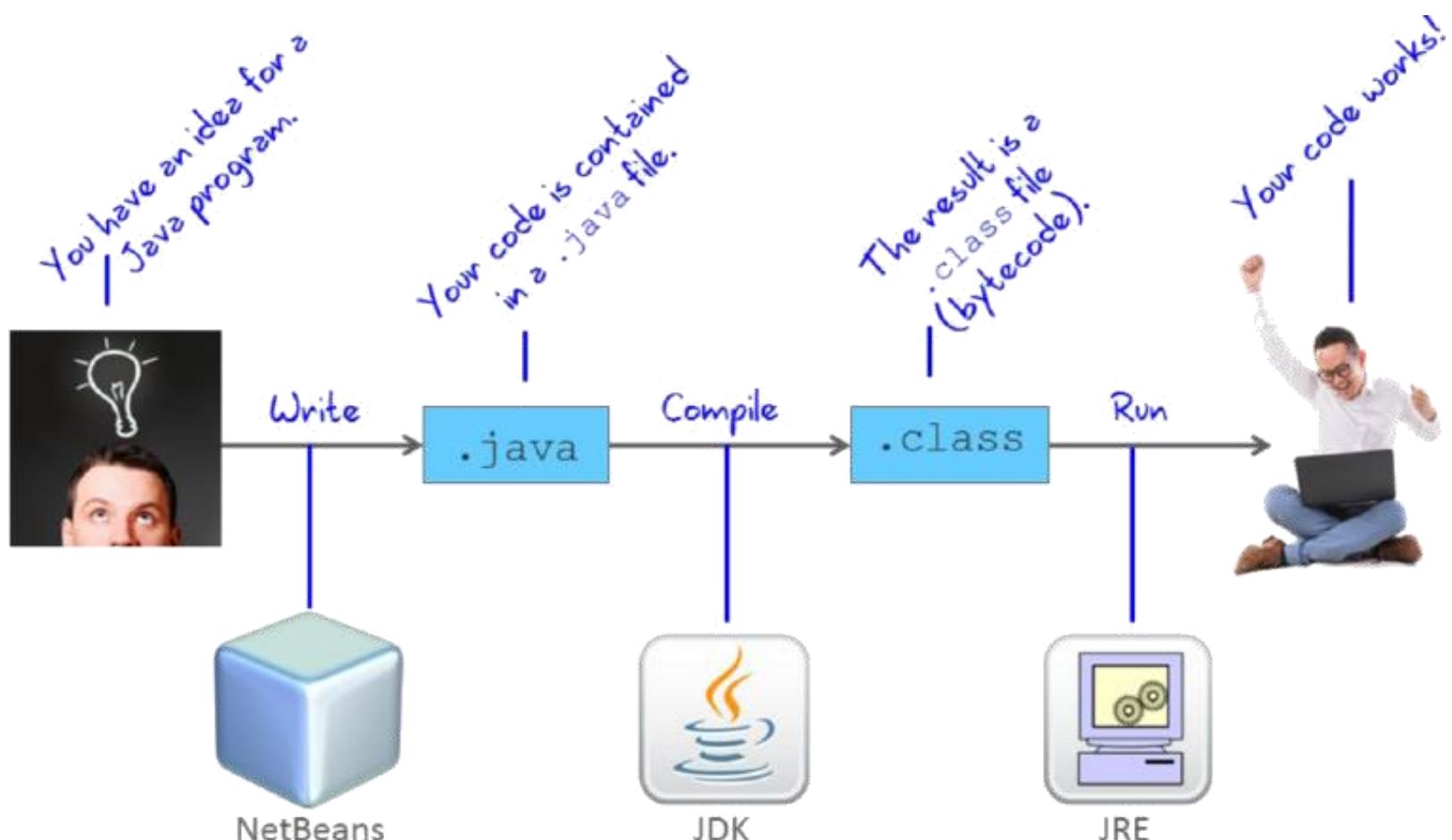
Κατηγορίες Προγραμμάτων Java

- **Java applications** – Αυτόνομα προγράμματα.
- **Java applets** – Προγράμματα ενσωματωμένα σε Ιστοσελίδα.
- **Java servlets** – Προγράμματα που εκτελούνται από διακομιστές Παγκοσμίου Ιστού (Web Servers) στο πλαίσια εφαρμογών client-server.
- **Enterprise Java Beans** – Προγράμματα για ανάπτυξη εταιρικών εφαρμογών με την αρχιτεκτονική τριών επιπέδων (3-tier) J2EE.
- **Java για κινητά**
 - J2ME
- **JavaCard**
 - Έξυπνες κάρτες που προγραμματίζονται σε Java
 - Πιστωτικές
 - ATM
 - Ηλεκτρονικές κλειδαριές κ.α.

Ένα απλό πρόγραμμα σε Java: Hello.java

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

Εκτέλεση προγράμματος Java



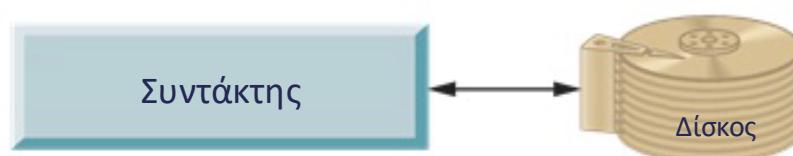
Εκτέλεση προγράμματος Java

- Συνήθως τα προγράμματα στη Java ακολουθούν πέντε στάδια:



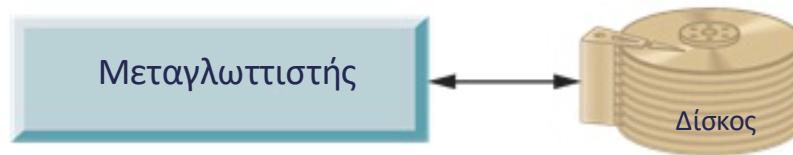
Εκτέλεση προγράμματος Java

Στάδιο 1:
Επεξεργασία



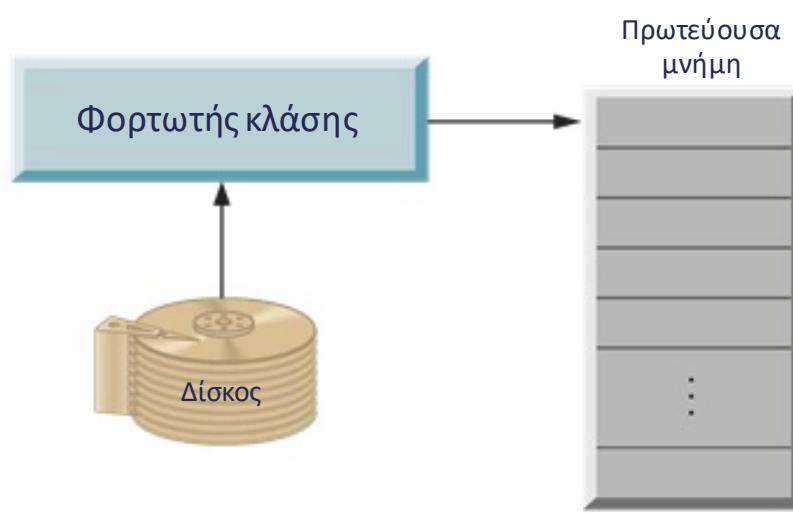
Το πρόγραμμα δημιουργείται με έναν συντάκτη και αποθηκεύεται στον δίσκο σε ένα αρχείο με κατάληξη **.java**

Στάδιο 2:
Μεταγλωττιση



Ο μεταγλωττιστής δημιουργεί bytecodes και τα αποθηκεύει στον δίσκο σε ένα αρχείο με κατάληξη **.class**

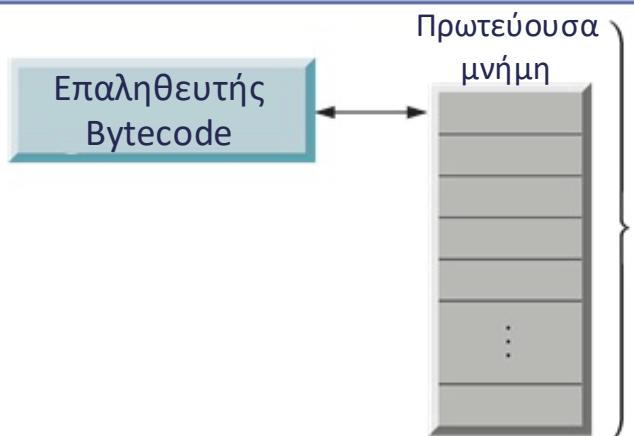
Στάδιο 3:
Φόρτωση



Ο φορτωτής κλάσης διαβάζει τα **.class** αρχεία που περιέχουντα bytecodes από τον δίσκο και μεταφέρει τα bytecodes αυτά στη μνήμη

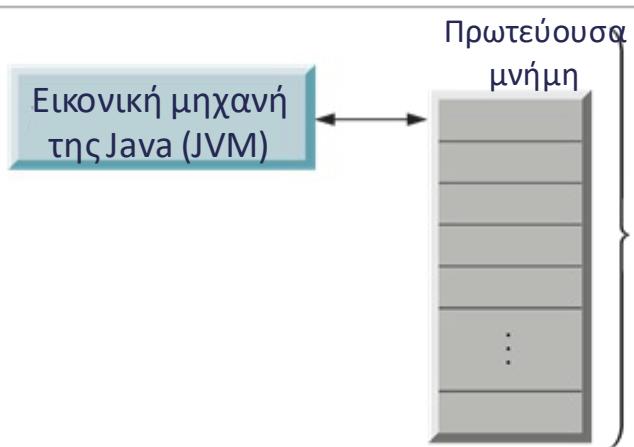
Εκτέλεση προγράμματος Java

Στάδιο 4:
Επαλήθευση



Ο bytecode επαληθευτής, επιβεβαιώνει πως όλα τα bytecodes είναι έγκυρα και δεν παραβιάζουν τους περιορισμούς ασφαλείας της Java.

Στάδιο 5:
Εκτέλεση



Για την εκτέλεση του προγράμματος η εικονική μηχανή της Java (JVM) διαβάζει τα bytecodes και ακριβώς όταν χρειάζεται (just-in-time, JIT) τα μεταγλωττίζει (μεταφράζει) σε μια γλώσσα που ο υπολογιστής μπορεί να καταλάβει. Κατά τη διάρκεια της εκτέλεσης του προγράμματος, αποθηκεύει δεδομένα στην κύρια μνήμη.

Δημιουργία κώδικα Java

- Το στάδιο 1 αποτελείται από την επεξεργασία ενός αρχείου με ένα πρόγραμμα σύνταξης (ή απλά συντάκτης).
 - Γράφουμε ένα πρόγραμμα σε Java (πηγαίος κώδικας) με τον συντάκτη (editor)
 - Πραγματοποιούμε τις απαραίτητες διορθώσεις
 - Σώζουμε το πρόγραμμα
- 'Ενα αρχείο με την κατάληξη .java υποδεικνύει ότι το αρχείο περιέχει πηγαίο κώδικα Java*
- Συντάκτες σε Linux: vi, emacs, jedit, geany, gedit ...
 - Συντάκτες σε Windows: Notepad, Notepad++ (www.notepad-plus-plus.org), EditPlus (www.editplus.com), TextPad (www.textpad.com) , jEdit (www.jedit.org) ...

Μεταγλώττιση κώδικα σε Java

- Στάδιο 2
 - Με χρήση της εντολής javac (ο μεταγλωττιστής Java) μεταγλωττίζουμε το πρόγραμμα. Παράδειγμα: για μεταγλώττιση του προγράμματος με όνομα Welcome.java, πληκτρολογούμε
 - javac Welcome.java*
 - Αν το πρόγραμμα μεταγλωττιστεί επιτυχώς, δημιουργείται ένα .class αρχείο με όνομα Welcome.class που περιέχει την μεταγλωττισμένη έκδοση του προγράμματος.

Μεταγλώττιση κώδικα σε Java

- Ο μεταγλωττιστής μεταφράζει τον πηγαίο κώδικα Java σε bytecodes που αντιπροσωπεύουν τις διεργασίες προς εκτέλεση.
- Τα bytecodes εκτελούνται από την Εικονική Μηχανή της Java (JVM)
 - Τα Bytecodes είναι ανεξάρτητα από την πλατφόρμα
 - Τα Bytecodes είναι φορητά - μπορούν να εκτελεστούν σε οποιαδήποτε πλατφόρμα περιέχει μία εικονική μηχανή της Java (JVM) που καταλαβαίνει την έκδοση της Java στην οποία μεταγλωττίστηκαν.
- Η JVM επικαλείται από την εντολή `java` . Για παράδειγμα για να εκτελέσουμε μία Java εφαρμογή με όνομα `Welcome`, θα πληκτρολογούσαμε την εντολή:

`java Welcome`

Φόρτωση κλάσεων σε Java

- Στάδιο 3
 - Η JVM μεταφέρει το πρόγραμμα στην μνήμη για εκτέλεση
 - Ο φορτωτής των κλάσεων (class loader) λαμβάνει τα .class αρχεία που περιέχουν τα bytecodes του προγράμματος και τα μεταφέρει στην κύρια μνήμη.
 - Φορτώνει επίσης όσα .class αρχεία παρέχονται από την Java τα οποία χρησιμοποιεί το πρόγραμμά μας.
 - Τα .class αρχεία μπορούν να φορτωθούν από έναν δίσκο στο σύστημά μας ή και μέσω δικτύου.

Επαλήθευση bytecode σε Java

- Στάδιο 4
 - Καθώς οι κλάσεις φορτώνονται, ο επαληθευτής bytecode (Bytecode verifier) εξετάζει τα bytecodes τους: Εξασφαλίζει ότι είναι έγκυρα και δεν παραβιάζουν τους περιορισμούς ασφαλείας της Java.
 - Η Java επιβάλει ισχυρή ασφάλεια ώστε να εξασφαλίσει ότι τα προγράμματα Java που φθάνουν από το δίκτυο δεν βλάπτουν τα αρχεία του συστήματός μας (όπως κάνουν οι ιοί στους υπολογιστές).

Σφάλματα στη Java

- Συντακτικά σφάλματα – Προκύπτουν όταν παραβιάζουμε τους κανόνες δόμησης του προγράμματος.
 - Ανιχνεύονται πάντοτε από το πρόγραμμα μεταγλώττισης.
- Σφάλματα εκτέλεσης – Λάθη εξαιτίας των οποίων το πρόγραμμα παράγει λανθασμένα αποτελέσματα ή διακόπτεται.
 - Η διακοπή εκτέλεσης στην Java συνοδεύεται από τη δημιουργία κατάστασης εξαίρεσης (exception).
- Λογικά σφάλματα – Λάθη στη λογική δημιουργίας του προγράμματος
 - Οδηγούν σε συμπεριφορά του προγράμματος αντίθετη με την αναμενόμενη.
 - Μπορεί να μην οδηγήσουν σε κάποιο πρόβλημα για αρκετό καιρό, και να διαφύγουν της προσοχής μας.

Το «λεξικό» της γλώσσας

abstract	assert	boolean	break	byte	case
catch	char	class	const	continue	default
double	do	else	enum	extends	false
final	finally	float	for	goto	if
implements	import	instanceof	int	interface	long
native	new	null	package	private	protected
public	return	short	static	strictfp	super
switch	synchronized	this	throw	throws	transient
true	try	void	volatile	while	

Το «λεξικό» της γλώσσας

abstract	assert	boolean	break	byte	case
catch	char	class	const	continue	default
double	do	else	enum	extends	false
final	finally	float	for	goto	if
implements	import	instanceof	int	interface	long
native	new	null	package	private	protected
public	return	short	static	strictfp	super
switch	synchronized	this	throw	throws	transient
true	try	void	volatile	while	

String?

Όνομασία μεταβλητών

- Ονόματα έως 255 χαρακτήρες
- Μπορούν να αρχίζουν από γράμμα, \$, _ (_hello, \$hello)
- Υποστηρίζονται Unicode χαρακτήρες (안녕하세요)
- Δεν μπορούμε να χρησιμοποιήσουμε κενά ή σύμβολα (hello!, hello there)
- Είναι case sensitive
- Δεν μπορούμε να χρησιμοποιήσουμε δεσμευμένες λέξεις

Χρήση ονομάτων στη Java

- Τα ονόματα των Packages πρέπει να είναι με μικρούς χαρακτήρες (lowercase).
 - Σε μικρά projects με λίγα packages μπορούμε να δίνουμε μικρά (αλλά πάντα με νόημα) ονόματα: package pokeranalyzer, package mycalculator
 - Σε εταιρίες λογισμικού και μεγάλα projects óπου τα packages μπορεί να εισαχθούν σε άλλες κλάσεις, τα ονόματα κανονικά υποδιαιρούνται. Συνήθως ξεκινάμε με το όνομα της εταιρίας πριν περάσουμε σε διαίρεση σε layers ή features: package com.mycompany.utilities, package org.bobscompany.application.userinterface
- Κλάσεις: Τα ονόματα πρέπει να είναι σε CamelCase.
 - Προσπαθούμε να χρησιμοποιούμε ουσιαστικά, αφού μια κλάση συνήθως εκφράζει κάτι από τον πραγματικό κόσμο: class Customer, class Account
- Διεπαφές: Τα ονόματα πρέπει να είναι σε CamelCase.
 - Συνήθως έχουν ένα όνομα που περιγράφει την πράξη που μπορεί να πραγματοποιηθεί από την κλάση: interface Comparable, interface Enumerable
 - Σημείωση: Κάποιοι προγραμματιστές προτιμούν να ξεχωρίζουν τις διεπαφές, ξεκινώντας το όνομα με "I": interface Icomparable, interface IComparable

Χρήση ονομάτων στη Java

- **Μέθοδοι:** Τα ονόματα πρέπει να είναι σε mixed case.
 - Χρησιμοποιούμε ρήματα για να περιγράψουμε το τι κάνει η μέθοδος: `void calculateTax()`, `string getSurname()`.
- **Μεταβλητές:** Τα ονόματα πρέπει να είναι σε mixed case.
 - Τα ονόματα πρέπει να εκφράζουν τις τιμές που παίρνει η μεταβλητή: `string firstName`, `int orderNumber`.
 - Χρησιμοποιούμε πολύ μικρά ονόματα όταν οι μεταβλητές έχουν πολύ μικρό χρόνο ζωής, όπως μέσα σε for loops: `for (int i=0; i<20;i++)`.
- **Σταθερές:** Τα ονόματα πρέπει να είναι με κεφαλαίους χαρακτήρες: `static final int DEFAULT_WIDTH`, `static final int MAX_HEIGHT`.

Μεταβλητές

- Τοποθεσίες στη μνήμη του υπολογιστή, τις οποίες έχουμε ονομάσει και στις οποίες αποθηκεύεται μία τιμή.

Παράδειγμα:

```
String a = "a";  
String b = "letter b";  
int k = 15;  
a = "letter a";  
String c = a + "and" + b;
```

Μεταβλητές

- Θυμηθείτε τις μεταβλητές σε μια εξίσωση.
- Μπορούμε να τις δώσουμε οποιαδήποτε αξία.
- Παρομοίως και στη Java:

```
String x = "Alex";  
System.out.println("My name is " +x);
```



"My name is Alex"

Έστω $y=-2x+5$

Εάν $x=0$, τότε $y=5$

Εάν $x=2$, τότε $y=1$

Γιατί μεταβλητές;

- Χωρίς μεταβλητές

```
System.out.println("My name is Alex");
System.out.println("Alex is so cool!");
System.out.println("Hooray Alex!");
System.out.println("Please enjoy Alex
Appreciation "
+ "Day! My name is Alex. I know how excited "
+ "everyone is to start appreciating Alex on Alex"
+ "Appreciation Day! Alex, Alex, Alex! Yay"
+ "Alex!!! That's me! Alex is the best date ever!");
```

- Με μεταβλητές

```
String x = "Sam";
System.out.println("My name is "+x);
System.out.println(x + " is so cool!");
System.out.println("Hooray "+x +"!");
System.out.println("Please enjoy "+x +
Appreciation "
+ "Day! My name is "+x +". I know how excited "
+ "everyone is to start appreciating "+x +" on "+x +
"Appreciation Day! "+x +"," +x +"," +x +"! Yay"+ x
+"!!! That's me! "+x +" is the best date ever!");
```

Τι αλλαγές πρέπει να κάνουμε στον παραπάνω κώδικα ώστε να εκτυπώνει το αποτέλεσμα και για την Rita, και στις 2 περιπτώσεις.

Τύποι Δεδομένων

- Όλοι οι τύποι..

	Τύπος τιμών	Μήκος σε bytes	Πεδίο τιμών	Αρχική τιμή
byte	Ακέραιος (Integer)	1	-128 ... 127	0
short		2	-32768 ... 32767	0
int		4	$-2^{31} \dots 2^{31}-1 / 0 \dots 2^{32}-1$ (Java SE8)	0
long		8	$-2^{63} \dots 2^{63}-1 / 0 \dots 2^{64}-1$ (Java SE8)	0
float	Κινητής υποδιαστολής (floating point number)	4	single-precision 32-bit IEEE 754 floating point	0
double		8	double-precision 64-bit IEEE 754 floating point	0
char	Χαρακτήρας (Single character)	2	\u0000' (0) ... '\uffff' (65.535)	0
boolean	Λογική τιμή	?	false - true	False

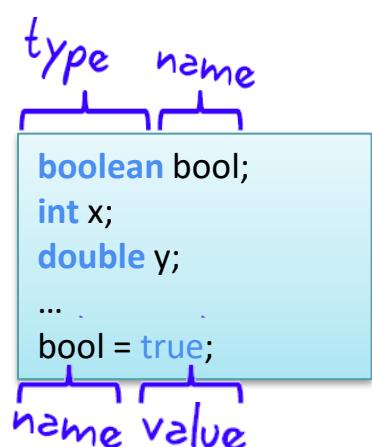
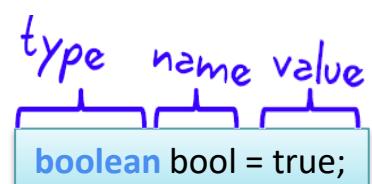
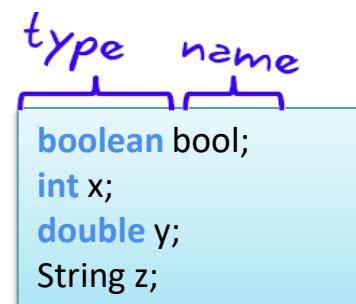
Τύποι Δεδομένων

- Συνήθεις τύποι..

Type	Keyword	Example Values
Boolean	boolean	true, false
Integer	int	1, -10, 20000
Double	double	1.0, -10.0005, 3.141
Char	char	'a', '&'
String	String	"Alex", "I ate too much dinner."

Δήλωση μεταβλητής

- Πρέπει να δηλώσετε τον τύπο της μεταβλητής χρησιμοποιώντας λέξεις-κλειδιά.
- Αφού δηλώσετε μια μεταβλητή ...
 - Αυτή η μεταβλητή υπάρχει..
 - Δεν χρειάζεται να τη δηλώσετε ξανά.
- Μπορείτε να δηλώσετε και ορίσετε μια μεταβλητή σε μία γραμμή:
- ή να δηλώσετε μια μεταβλητή σε μία γραμμή και να την ορίσετε σε μία άλλη



Δήλωση μεταβλητής – Συνήθη λάθη

1. Ακατάλληλες τιμές σε μία μεταβλητή

```
int intVar1 = true;  
int z = "Puppies!";
```

2. Μη δήλωση μεταβλητής

```
intVar3 = 3; //intVar3 ?
```

3. Λάθος γραφή μεταβλητής

```
double doubleVar2;  
doublevAr2 = 2.1;
```

4. Διπλή δήλωση της ίδια μεταβλητής

```
double doubleVar3;  
double doubleVar3 = 3.1;
```

5. Χρήση μεταβλητής χωρίς προηγούμενη εκχώρηση τιμής

```
double doubleVar4;  
System.out.println (doubleVar4);
```

```
y=-2x+5  
x="Puppies!";  
y=-2 ("Puppies "+)+5  
y=???
```

Παράλειψη αρχικοποίησης

→ Είναι σωστό το παρακάτω;

```
double doubleVar1, doubleVar2, doubleVar3 = 3.1;
```

Όνομασία Μεταβλητών

```
int dsfdsfspoop = 20;           // «Τι θέλει να πει ο ποιητής;»  
int x = 20;                     /* συνήθως μόνο για testing ή δείκτης επανάληψης για μικρούς  
βρόγχους (loops) */  
  
int x = 20;                     // Ποιο x?  
double x = 22.0;  
System.out.println(x);  
  
boolean 1337Hacker = true;      /* οι μεταβλητές δεν μπορούν να ξεκινούν από αριθμό */  
  
int continue= 20;               /* Οι δεσμευμένες λέξεις δεν μπορούν να χρησιμοποιηθούν για  
όνομα μεταβλητής - για αυτό και έχουν χρώμα μπλε στο  
NetBeans */
```

Καλά Όνόματα!

- myVariable ✓
- int studentAge = 20; ✓
- String myCatchPhrase = "Enjoy Alex Appreciation Day!"; ✓

Μεταβλητές και Σταθερές

- **Μία μεταβλητή**
 - Μπορεί να παίρνει διαφορετικές τιμές κατά τη διάρκεια του προγράμματος.
 - Μπορεί να δηλωθεί σε κάποιο σημείο και να πάρει (και να ξαναπάρει) τιμή αλλού.
 - **Μία σταθερά**
 - Παίρνει μία τιμή (μπορεί και μετά τη δήλωσή της).
 - Δηλώνεται σε ένα σημείο του κώδικα και δεν μπορεί να ξαναδηλωθεί.
 - Η σύμβαση λέει ότι πρέπει να χρησιμοποιούνται κεφαλαία γράμματα και το σύμβολο _ για την ονομασία της.
- int numberOfHoursInADay= 67;
numberOfHoursInADay= 24;
- final int NUMBER_OF_HOURS_IN_A_DAY = 24;

Αριθμητικά δεδομένα

Ακέραιοι

Πότε τους χρησιμοποιούμε;

Τύπος	Μήκος	Πιθανές τιμές	Ελάχιστη τιμή	Μέγιστη τιμή
Ποτέ → Byte	8 bits	2^8 , ή... 256	-2^7 , ή... -128	2^7-1 , ή... 127
Ποτέ → short	16 bits	2^{16} , ή... 65.535	-2^{15} , ή... -32.768	$2^{15}-1$, ή... 32.767
Συχνά → int	32 bits	2^{32} , ή... 4.294.967.296	-2^{31} , ή... -2.147.483.648	$2^{31}-1$, ή... 2.147.483.647
Σπάνια → long	64 bits	2^{64} , ή... 18.446.744.073.709.551.616	-2^{63} , ή... -9.223.372.036.854.775.808L	$2^{63}-1$, ή... 9.223.372.036.854.775.807L

Μαθηματικοί τελεστές

Πράξη	Τελεστής	Παράδειγμα	Σχόλια
Πρόσθεση	+	sum = num1 + num2;	If num1 is 10 and num2 is 2, sum is 12
Αφαίρεση	-	diff = num1 - num2;	If num1 is 10 and num2 is 2, diff is 8
Πολλαπλασιασμός	*	prod = num1 * num2;	If num1 is 10 and num2 is 2, prod is 20
Διαίρεση	/	quot = num1 / num2;	If num1 is 31 and num2 is 6, quot is 5 → Το υπόλοιπο απορρίπτεται → Η διαίρεση με 0 επιστρέφει σφάλμα
Υπόλοιπο	%	num1 = 31; num2 = 6; mod = num1 % num2; mod is 1	Το υπόλοιπο του πρώτου αριθμού διαιρούμενο με τον δεύτερο αριθμό

Εκχώρηση αριθμών

- Βρείτε το x?

Εντολές

```
int x = 20;
x = 25;
x = 5 + 3;
System.out.println(x);
```

Τιμές του x

20
25
8

Οθόνη

8

```
int x = 20;
x = 25;
x = 5 + 3;
x = x + 1;
x += 1;
x++;
System.out.println(x);
```

20
25
8
9
10
11

11

συντόμευση += για αύξηση κατά y
συντόμευση ++ για αύξηση κατά 1

Εντολές

```
int y = 20;
int x = y;
y++;
System.out.println(x);
System.out.println(y);
```

Τιμές του x, y

x	y
20	20
	21

Οθόνη

20
21

Η αλλαγή του y δεν επηρεάζει το x

Τελεστές και εκχωρήσεις

Πράξη	Τελεστής	Παράδειγμα	Αποτέλεσμα
Εκχώρηση	=	a = b	a = 2
Πρόσθεση & εκχώρηση	+=	a += b	a = 8
Αφαίρεση & εκχώρηση	-=	a -= b	a = 4
Πολλαπλασιασμός & εκχώρηση	*=	a *= b	a = 12
Διαίρεση & εκχώρηση	/=	a /= b	a = 3
Αύξηση κατά 1	++	a++; ($\Rightarrow a = a + 1;$)	a = 7
Προ-αύξηση (++ μεταβλητή)		b= ++a;	a=7 b=7
Μετα-αύξηση (μεταβλητή++)		b= a++;	b=6 a=7
Μείωση κατά 1	--	a--; ($\Rightarrow a = a - 1;$)	a = 5
Προ-μείωση (-- μεταβλητή)		b= --a;	a=5 b=5
Μετα-μείωση (μεταβλητή--)		b= a--;	b=6 a=5

Τι εμφανίζει το παρακάτω πρόγραμμα;

```
1. int count=15;
2. int a, b, c, d;
3. a = count++;
4. b = count;
5. c = ++count;
6. d = count;
7. System.out.println(a + ", " + b + ", " + c + ", " + d);
```

count	a	b	c	d
15				
16	15			
		16		
17			17	
				17

Output
15, 16, 17, 17

Πραγματικοί

Τύπος	Μήκος	Παράδειγμα	Πότε χρησιμοποιούνται;
float	32 bits	public float pi = 3.141592F;	Σχεδόν ποτέ
double	64 bits	public double pi = 3.141592;	Συχνά

- Προσέξτε τα παρακάτω:

(α)

```
int x = 9/2;  
System.out.println(x); //prints 4
```

(β)

```
double x = 9/2;  
System.out.println(x); //prints 4.0
```

- Γιατί συμβαίνει αυτό;

Πραγματικοί

Τύπος	Μήκος	Παράδειγμα	Πότε χρησιμοποιούνται;
float	32 bits	public float pi = 3.141592F;	Σχεδόν ποτέ
double	64 bits	public double pi = 3.141592;	Συχνά

- Προσέξτε τα παρακάτω:

(α)

```
int x = 9/2;  
System.out.println(x); //prints 4
```

(β)

```
double x = 9/2;  
System.out.println(x); //prints 4.0
```

- Γιατί συμβαίνει αυτό;

- Στο (α) η Java κόβει το .5
- Στο (β) μετατρέπει τον x σε double.

Η διαίρεση ("/") λειτουργεί διαφορετικά στους ακέραιους και στους πραγματικούς (οι ακέραιοι αριθμοί δεν **στρογγυλοποιούνται**, αλλά **περικόπτονται**)

Πραγματικοί

Τύπος	Μήκος	Παράδειγμα	Πότε χρησιμοποιούνται;
float	32 bits	public float pi = 3.141592F;	Σχεδόν ποτέ
double	64 bits	public double pi = 3.141592;	Συχνά

- Προσέξτε τα παρακάτω:

(α)

```
int x = 9/2;
System.out.println(x); //prints 4
```

(β)

```
double x = 9/2;
System.out.println(x); //prints 4.0
```

- Γιατί συμβαίνει αυτό;

- Στο (α) η Java κόβει το .5
- Στο (β) μετατρέπει τον x σε double.

Η διαίρεση ("/") λειτουργεί διαφορετικά στους ακέραιους και στους πραγματικούς (οι ακέραιοι αριθμοί δεν **στρογγυλοποιούνται**, αλλά **περικόπτονται**)

- Λύση:

- Χρήση double στην πράξη

(γ)

```
double x = 9/2.0;
System.out.println(x); //prints 4.5
```

Προτεραιότητας Εκτέλεσης Πράξεων

Οι πράξεις εκτελούνται κατά προτεραιότητα με την ακόλουθη σειρά.

1. Παρενθέσεις '()'
2. Αύξηση/μείωση '++' ή '--'
3. Πολλαπλασιασμός/διαίρεση '*' ή '/' (από αριστερά προς τα δεξιά)
4. Πρόσθεση/αφαίρεση '+' ή '-' (από αριστερά προς τα δεξιά)

Τελεστές της ίδιας προτεραιότητας εκτελούνται από αριστερά προς τα δεξιά.

Προτεραιότητας Εκτέλεσης Πράξεων: Παραδείγματα

```
int x = 10 +20 +30 / 3;
```

```
int x = 10 +20 +(30 / 3);
```

```
int x = (10 +20 +30) / 3;
```

```
double x = 3 / 2 + 1;
```

```
double y = 3 / (2+1);
```

```
int x = (((25 -5) * 4) / (2 -10)) + 4;
```

```
int x = 25 -5 * 4 / 2 -10 + 4;
```

Προτεραιότητας Εκτέλεσης Πράξεων:

Παραδείγματα

int x = 10 +20 +30 / 3; x=40

int x = 10 +20 +(30 / 3); x=40

int x = (10 +20 +30) / 3; x=20

double x = 3 / 2 + 1; x = 2.0

double y = 3 / (2+1); y = 1.0

int x = (((25 -5) * 4) / (2 -10)) + 4; x = -6

int x = 25 -5 * 4 / 2 -10 + 4; x = 9

Χαρακτήρες

Τύπος δεδομένων

Τύπος	Μήκος	Πιθανές τιμές
char	16 bits	'a..z', 'A..Z', ...

- π.χ. `char shirtSize = 'M';` // μονά εισαγωγικά για απλούς χαρακτήρες
- Μπορείτε να ενώσετε χαρακτήρες για να δημιουργήσετε προτάσεις.

```
char letter1 = 'H';
char letter2 = 'e';
char letter3 = 'l';
char letter4 = 'l';
char letter5 = 'o';
System.out.println(letter1 +letter2 +letter3 +letter4 +letter5);
// Οι μεγάλες προτάσεις θα ήταν λίγο επώδυνο να κωδικοποιηθούν
```

Τύπος δεδομένων

Τύπος	Μήκος	Πιθανές τιμές
char	16 bits	'a..z', 'A..Z', ...

- π.χ. `char shirtSize = 'M';` // μονά εισαγωγικά για απλούς χαρακτήρες
- Μπορείτε να ενώσετε χαρακτήρες για να δημιουργήσετε προτάσεις.

```
char letter1 = 'H';
char letter2 = 'e';
char letter3 = 'l';
char letter4 = 'l';
char letter5 = 'o';
System.out.println(letter1 +letter2 +letter3 +letter4 +letter5);
// Οι μεγάλες προτάσεις θα ήταν λίγο επώδυνο να κωδικοποιηθούν
```

- Όμως υπάρχει και καλύτερος τρόπος

```
String greeting = "Hello World!";
//Παρατηρήστε τα διπλά εισαγωγικά
System.out.println(greeting);
```

Χαρακτήρες (chars) Vs Συμβολοσειρές (strings)

- Οι χαρακτήρες (chars):
 - είναι για ένα μόνο χαρακτήρα
 - χρησιμοποιούν μονά εισαγωγικά
 - δεν μπορούν να χειριστούν πολλούς χαρακτήρες
 - char είναι δεσμευμένη λέξη της Java (γίνεται μπλε)
 - `char shirt1Size = 'S';` 
 - `char shirt2Size = 'M';` 
 - `char shirt3Size = 'L';` 
 - `char shirt4Size = 'XL';` 
 - `char shirt5Size = "XXL";` 
- Οι συμβολοσειρές (strings)
 - είναι για πολλούς χαρακτήρες
 - χρησιμοποιούν διπλά εισαγωγικά
 - είναι αντικείμενα ειδικού σκοπού, όχι δεσμευμένη λέξη (έχει κεφαλαίο το πρώτο γράμμα)
 - όπως όλα τα αντικείμενα έχουν **ιδιότητες** (String properties) & **συμπεριφορές** (String behaviors)
 - `String shirt6Size = "XXXL";` 

Παραδείγματα

- Δήλωση και αρχικοποίηση μία μεταβλητής

```
int intVar = 300;
```

```
String stringVar = "Three Hundred";
```

- Δήλωση και αρχικοποίηση πολλών μεταβλητών

```
int x, y, z;
```

```
String xString, yString, zString;
```

- Αρχικοποίηση δηλωμένων μεταβλητών

```
xString = "One";
```

Δηλώστε και χρησιμοποιήστε τα Strings όπως
τους άλλους τύπους δεδομένων τη Java

Συνδυασμός Strings

- ```
String combinedLiterals = "I want to"+ " buy a shirt.";
System.out.println(combinedLiterals);
```

 → I want to buy a shirt.
- ```
String var1 = "This shirt has";  
String var2 = " too many buttons."  
String combinedVariables = var1 + var2;  
System.out.println(combinedVariables);
```

 → This shirt has too many buttons.
- ```
String greet1 = "Hello";
String greet2 = "World";
String message1 = greet1 +" "+greet2 +"!";
String message2 = greet1 +" "+greet2 +" "+2016 +"!";
System.out.println(message2);
System.out.println(greet1 +" "+greet2 +" "+2016 +"!");
```

 → Hello World 2016!  
→ Hello World 2016!

# Strings & Numbers

- Οι συμβολοσειρές (Strings) μπορούν να περιέχουν αριθμούς:
  - String totalPrice = "Total: \$" + 3;
  - System.out.println(totalPrice); //Total: \$3
- Προσοχή όμως:
  - String totalPrice = "Total: \$" + 3 + 2 + 1;
  - System.out.println(totalPrice); //Total: \$321
- Καλύτερα παρενθέσεις για αριθμούς:
  - String totalPrice = "Total: \$" +(3 + 2 + 1);
  - System.out.println(totalPrice); //Total: \$6

# *The Underscore “\_”*

- Με την Java SE7, μπορείτε να χρησιμοποιείτε **underscore “\_”** για να κάνετε μεγάλους αριθμούς πιο ευανάγνωστους:
  - `int x = 123456789;` → `int x = 123_456_789;`

# Ακολουθίες διαφυγής

- Ένας χαρακτήρας μετά από το '\' ονομάζεται ειδικός χαρακτήρας και δημιουργεί μία ακολουθία διαφυγής (escape sequence) με ιδιαίτερη σημασία για τον μεταγλωττιστή.
- Παραδείγματα
  - `System.out.println("This is the first line. \n" + "This is the second line.");`
  - `System.out.println("The cat said \"Meow!\\\" to me.");`

| Ειδικός χαρακτήρας | Περιγραφή        |
|--------------------|------------------|
| \t                 | new tab          |
| \b                 | backspace        |
| \n                 | new line         |
| \r                 | Αλλαγή γραμμής   |
| \f                 | Form feed        |
| '                  | Απλό εισαγωγικό  |
| "                  | Διπλά εισαγωγικά |
| \\"                | Backslash        |

Output:  
This is the first line.  
This is the second line.

Output:  
The cat said "Meow!" to me.

# Εκτύπωση αλλαγής γραμμής println

- Όταν γράφουμε κείμενο σε νέα γραμμή, δεν σημαίνει και ότι θα εκτυπωθεί σε μια νέα γραμμή:

```
System.out.println("This is the first line."
+ "This is NOT the second line.");
```

Output  
This is the first line. This is NOT the second line.

- Αυτό γίνεται με 2 τρόπους:

- Escape sequences:

```
System.out.println("This is the first line. \n"
+ "This is the second line.");
```

- Println:

```
System.out.println("This is the first line.");
System.out.println("This is the second line.");
```

Output  
This is the first line.  
This is the second line.

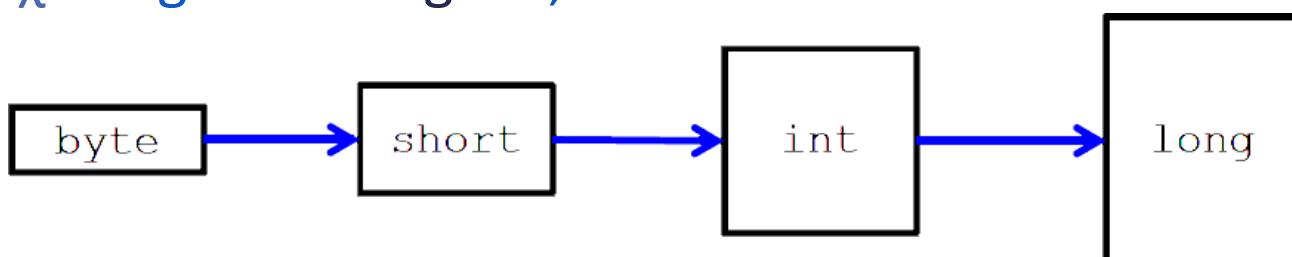
# Ιεραρχία τύπων δεδομένων και implicit typecast

- Οι τύποι δεδομένων στη Java είναι ιεραρχημένοι.
- Το νόημα είναι να μπορεί ένας αριθμός που ανήκει σε τύπο πιο ψηλά στην ιεραρχία, να συγκρατήσει την ακρίβεια του χαμηλότερου στη μετατροπή (άρα όσο πιο ψηλά, τόσο μεγαλύτερη ακρίβεια ενός τύπου)
- Η ιεραρχία είναι:
  1. double
  2. float
  3. long
  4. int
  5. short
  6. byte
- Implicit typecast: Αναβαθμίζει τον τύπο, σύμφωνα με την ιεραρχία

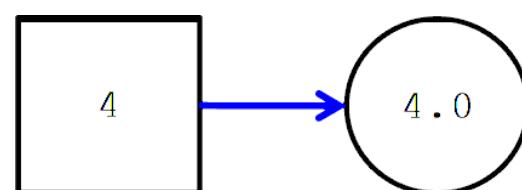
# Προβιβασμοί (Promotion)/ implicit typecast

- Αυτόματοι

- Εκχώρηση μικρότερου τύπου σε μεγαλύτερο,  
π.χ. `long intToLong = 6;`



- Εκχώρηση ακεραίου σε αριθμό κινητής υποδιαστολής, π.χ. `double intToDouble = 4;`



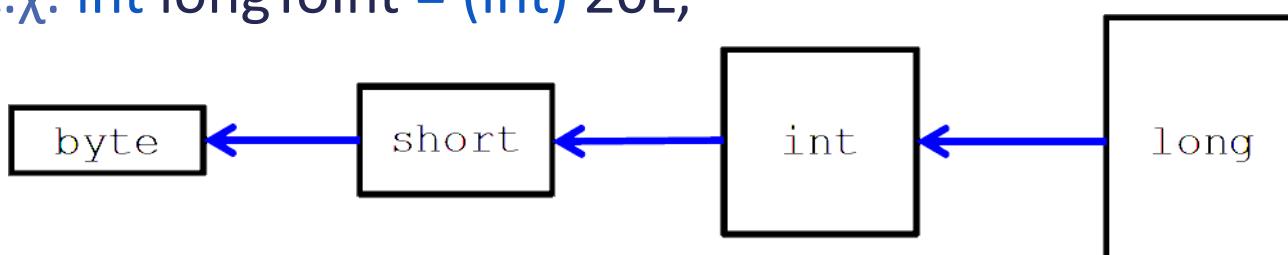
# Προβιβασμοί (Promotion): Προβλήματα

| Πιθανό πρόβλημα                                                               | Λύση                                                                           |
|-------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| <pre>int num1 = 55555; int num2 = 66666; long num3; num3 = num1 * num2;</pre> | <pre>int num1 = 55555; long num2 = 66666; long num3; num3 = num1 * num2;</pre> |
| Λάθος → //num3 is -591337666                                                  | //num3 is 3703629630 ←Σωστό                                                    |

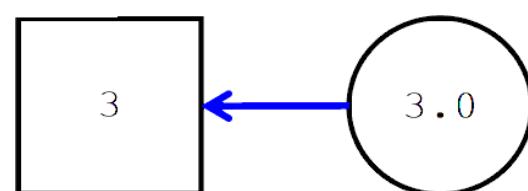
| Πιθανό πρόβλημα                                                         | Λύση                                                                       |
|-------------------------------------------------------------------------|----------------------------------------------------------------------------|
| <pre>int num1 = 7; int num2 = 2; double num3; num3 = num1 / num2;</pre> | <pre>int num1 = 7; double num2 = 2; double num3; num3 = num1 / num2;</pre> |
| Λάθος → //num3 is 3.0                                                   | //num3 is 3.5 ←Σωστό                                                       |

# Μετατροπές (Casting)

- Πότε:
  - Εκχώρηση μεγαλύτερου τύπου σε μικρότερο,  
π.χ. `int longToInt = (int) 20L;`



- Εκχώρηση αριθμού κινητής υποδιαστολής σε ακέραιο  
π.χ. `short doubleToShort = (short) 3.0;`



# Μετατροπές (Casting): Προβλήματα

| Πιθανό πρόβλημα                                                                 | Λύση                                                                                        |
|---------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <pre>int myInt;<br/>long myLong = 123987654321L;<br/>myInt = (int)myLong;</pre> | <pre>int myInt;<br/>long myLong = 99L;<br/>myInt = (int)myLong;</pre>                       |
| <b>Λάθος →</b> // Number is "chopped"<br>// myInt is -566397263                 | // myInt is 99<br>//no data loss, only zeroes.<br><span style="color: green;">←Σωστό</span> |

# Παράδειγμα

Ποιά από τα παρακάτω επιτρέπονται;

- **int x = 3.5;**
- **float x = 3;**
- **long i = 3;**
- **byte x = 155;**
- **double d = 3.14159F;**

# Παραδείγματα

Ποιά από τα παρακάτω επιτρέπονται;

- **int x = 3.5;**

Δεν επιτρέπεται γιατί το 3.5 δεν χωράει στο int

- **float x = 3;**

Επιτρέπεται γιατί το 3 είναι τύπου int που που χωράει στο float

- **long i = 3;**

Επιτρέπεται γιατί το 3 είναι τύπου int που που χωράει στο long

- **byte x = 155;**

Δεν επιτρέπεται γιατί το 155 είναι πολύ μεγάλο για να χωρέσει ένα byte  
( $<=127$ )

- **double d = 3.14159F;**

Επιτρέπεται γιατί το 3.14159F είναι τύπου float, που χωράει στο double

# Μετατροπή συμβολοσειρών σε αριθμητικά δεδομένα

- Η μετατροπή κειμένου σε αριθμό είναι μορφή ανάλυσης (parsing).

- String to an `int`:

```
int intVar1 = Integer.parseInt("100");
```

- String to a `double`:

```
double doubleVar2 = Double.parseDouble("2.72");
```

# Εισαγωγή δεδομένων από χρήστη

- Υπάρχουν πολλοί τρόποι για να πάρουμε δεδομένα από χρήστες (user input):
  - Κουμπιά (φυσικά ή εικονικά)
  - Αναγνώριση φωνής
  - Παράθυρα διαλόγου
- Η Java προσφέρει διάφορους τρόπους όπως:
  - JavaFX (διάδοχος του Swing)
  - Scanner

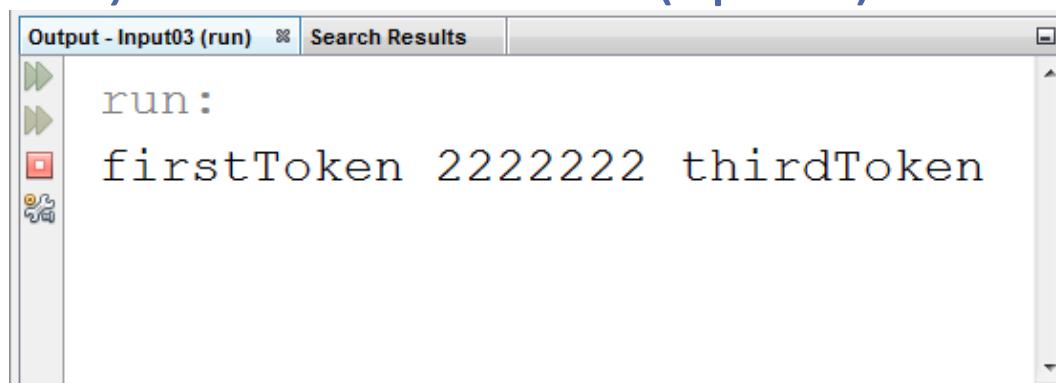
# Εισαγωγή δεδομένων με αντικείμενα Scanner

- Ένα αντικείμενο Scanner ανοίγει μια ροή για τη συλλογή δεδομένων εισόδου:
  - System.in διαβάζει για να ροές δεδομένων εισόδου από το πληκτρολόγιο.
  - To Scanner μπορεί επίσης να χρησιμοποιηθεί χωρίς IDE.
- Καλό είναι να κλείνετε τη ροή εισόδου όταν τελειώσετε.

```
public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);
 System.out.println("Hello, world!");
 System.out.println("Hello, world!");
 sc.close();
}
```

# Ανάγνωση εισόδου με Scanner

- Τα αντικείμενα Scanner αναζητούν αναζήτηση για λεκτικές μονάδες (tokens).
- Τα tokens διαχωρίζονται με έναν συγκεκριμένο οριοθέτη (delimiter).
  - Συνήθως είναι ένα “κενό” (Space).



The screenshot shows an IDE's output window titled "Output - Input03 (run)". The window displays the results of a Java program using the Scanner class. The output is as follows:

```
run:
firstToken 2222222 thirdToken
```

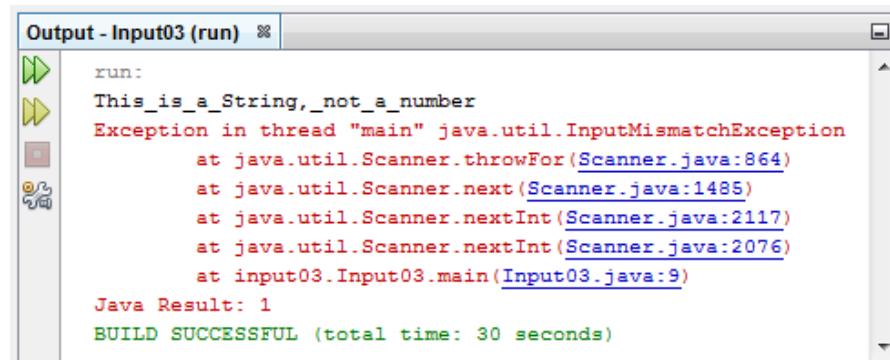
The output consists of two lines. The first line starts with "run:" followed by a colon. The second line contains three tokens: "firstToken", "2222222", and "thirdToken".

# Μέθοδοι της κλάσης Scanner

- `nextInt ()`: διαβάζει το επόμενο token ως int.
- `nextDouble ()`: διαβάζει το επόμενο token ως πραγματικό.
- `next()`: διαβάζει το επόμενο token ως String
- `nextLine()`: διαβάζει την είσοδο ως String μέχρι ο χρήστης να αλλάξει γραμμή

```
public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);
 int x = sc.nextInt();
 double y = sc.nextDouble();
 String z = sc.next();
 String input = sc.nextLine();
 sc.close();
}
```

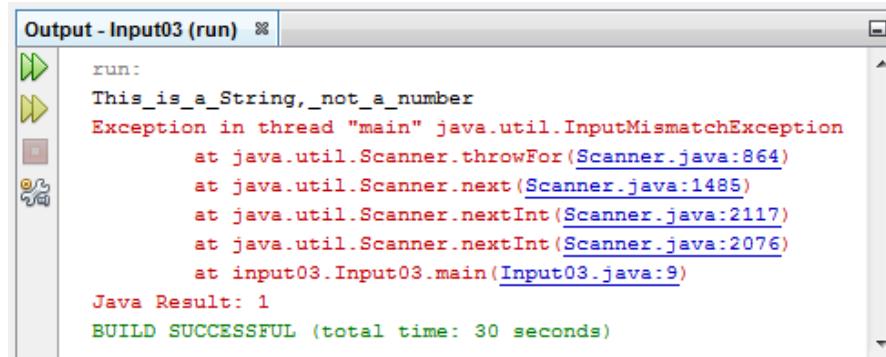
# Πιθανά λάθη: InputMismatch Exception



- Συμβαίνει επειδή τα δεδομένα εισόδου δεν μπορούν να αναλυθούν (parse) όπως ήταν αναμενόμενο, π.χ.

```
public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);
 System.out.println(sc.nextInt());
 sc.close();
}
```

# Πιθανά λάθη: IllegalStateException



- Συμβαίνει επειδή τα δεδομένα εισόδου προσπελάστηκαν αφού είχε κλείσει η ροή εισόδου, π.χ.

```
public static void main(String[] args) {
 Scanner sc = new Scanner(System.in);
 sc.close();
 System.out.println(sc.nextInt());
}
```

# Κλάσεις (Classes)

- Κατηγορίες Κλάσεων:
  1. Κλάσεις που θα γράψετε εσείς..
  2. Κλάσεις που έχει γράψει κάποιος άλλος..
  3. Κλάσεις που ανήκουν στην Java

# Χρήση μεθόδων

- Εάν ...
  - Πολλές παρόμοιες γραμμές κώδικα  
(συμπεριλαμβανομένων υπολογισμών).
  - Πρέπει να περιγράψουμε τη συμπεριφορά ενός αντικειμένου.

# Κύρια Μέθοδος (main)

- Γνωστή και ως **οδηγός** (driver):
  - Καθοδηγεί τα συμβάντα (events) ενός προγράμματος.
  - Παρέχει πρόσβαση σε πεδία και μεθόδους ή σε άλλες κλάσεις.
- Δεν περιγράφει τη συμπεριφορά κάποιου συγκεκριμένου αντικειμένου.
  - Πρέπει να είναι ξεχωριστά από τις διάφορες κλάσεις αντικειμένων.
  - Μόνο μία κύρια μέθοδο για κάθε εφαρμογή.

# Δημιουργία κλάσης αντικειμένων

```
public class Calculator{
 public static void main(String args[]){
 double tax = 0.05;
 double tip = 0.15;
 double person1 = 10;
 double total1 = person1*(1+tax+tip);
 System.out.println(total1);
 }
}
```

Βήματα:

# Δημιουργία κλάσης αντικειμένων

```
public class Calculator{
 //Fields
 public double tax = 0.05;
 public double tip= 0.15;
 public double originalPrice= 10;

 public static void main(String args[]){
 //double tax = 0.05;
 //double tip = 0.15;
 //double person1 = 10;
 double total1 = originalPrice *(1 +tax +tip);
 System.out.println(total1); //}
}
```



Βήματα:

1. Μετακίνηση πεδίων από την κύρια μέθοδο

# Δημιουργία κλάσης αντικειμένων

```
public class Calculator{
 //Fields
 public double tax = 0.05;
 public double tip= 0.15;
 public double originalPrice= 10;

 //Methods
 public void findTotal(){
 double total1 = originalPrice *(1 +tax +tip);
 System.out.println(total1);
 }

 public static void main(String args[]){
 //double tax = 0.05;
 //double tip = 0.15;
 //double person1 = 10;
 //double total1 = person1*(1 +tax +tip);
 //System.out.println(total1);
 }
}
```



## Βήματα:

1. Μετακίνηση πεδίων από την κύρια μέθοδο
2. Μετακίνηση επαναλαμβανόμενων συμπεριφορών από την κύρια μέθοδο

# Δημιουργία κλάσης αντικειμένων

```
public class Calculator{
 //Fields
 public double tax = 0.05;
 public double tip= 0.15;
 public double originalPrice= 10;

 //Methods
 public void findTotal(){
 double total1 = originalPrice *(1 +tax +tip);
 System.out.println(total1);
 }
 //public static void main(String args[]){
 // double tax = 0.05;
 // double tip = 0.15;
 // double person1 = 10;
 // double total1 = person1*(1 +tax +tip);
 // System.out.println(total1);
 //}
}
```

## Βήματα:

1. Μετακίνηση πεδίων από την κύρια μέθοδο
2. Μετακίνηση επαναλαμβανόμενων συμπεριφορών από την κύρια μέθοδο
3. Κατάργηση κύριας μεθόδου

# Δημιουργία κλάσης αντικειμένων

```
public class Calculator{
 //Fields
 public double tax = 0.05;
 public double tip= 0.15;
 public double originalPrice= 10;

 //Methods
 public void findTotal(){
 double total1 = originalPrice *(1 +tax +tip);
 System.out.println(total1);
 }
}
```

Βήματα:

1. Μετακίνηση πεδίων από την κύρια μέθοδο
2. Μετακίνηση επαναλαμβανόμενων συμπεριφορών από την κύρια μέθοδο
3. Κατάργηση κύριας μεθόδου

**Τέλος!**

# Δημιουργία κλάσης αντικειμένων

```
public class Calculator{
 //Fields
 public double tax = 0.05;
 public double tip= 0.15;
 public double originalPrice= 10;

 //Methods
 public void findTotal(){
 double total1 = originalPrice *(1 +tax +tip);
 System.out.println(total1);
 }
}
```

Βήματα:

1. Μετακίνηση πεδίων από την κύρια μέθοδο
2. Μετακίνηση επαναλαμβανόμενων συμπεριφορών από την κύρια μέθοδο
3. Κατάργηση κύριας μεθόδου

**Τέλος!**

**Και η κύρια main() μέθοδος;;;**

# Δημιουργία κλάσης αντικειμένων

```
public class Calculator{
 //Fields
 public double tax = 0.05;
 public double tip= 0.15;
 public double originalPrice= 10;

 //Methods
 public void findTotal(){
 double total1 = originalPrice *(1 +tax +tip);
 System.out.println(total1);
 }
}

public class CalculatorTest {
 public static void main(String args[]){
 //Create Calculator object instance
 Calculator calc= new Calculator();
 calc.tip= 0.10; //Altering a field
 calc.findTotal(); //Calling a method
 }
}
```

Βήματα:

1. Μετακίνηση πεδίων από την κύρια μέθοδο
2. Μετακίνηση επαναλαμβανόμενων συμπεριφορών από την κύρια μέθοδο
3. Κατάργηση κύριας μεθόδου

**Τέλος!**

**Και η κύρια main() μέθοδος;;;**

Σε μια άλλη κλάση, π.χ. test class η οποία:

- Δημιουργεί αντικείμενα
- Καλεί πεδία και μεθόδους ενός αντικειμένου χρησιμοποιώντας την τελεία “”

# Μεταβλητές αντικειμένων

- Όπως και στους βασικούς τύπους (primitives), για τα αντικείμενα χρησιμοποιούνται μεταβλητές.
- Για τον ορισμό και την αρχικοποίησή τους απαιτείται η λέξη `new`.
  - Αυτό ονομάζεται instantiating (δημιουργία αντικειμένου).
  - Στα αντικείμενα τύπου **String**, δεν απαιτείται instantiating.

```
int age = 22;
String str = "Happy Birthday!";
Scanner sc = new Scanner();
Calculator calc = new Calculator();
```



A diagram illustrating the components of a variable. It shows three horizontal arrows pointing downwards from the code above. The first arrow is under "int", the second under "sc", and the third under "calc". Below these arrows, the words "type", "name", and "value" are written in a stylized font, corresponding to each arrow respectively.

type      name      value

# Ο τελεστής ":"

- Τοποθετείται μετά το όνομα μιας μεταβλητής **αντικειμένου** για πρόσβαση στα **πεδία** ή τις **μεθόδους** του.

```
public class CalculatorTest {
 public static void main(String args[]) {
 Calculator calc = new Calculator();
 calc.printTip(); //prints 0.15
 calc.tip = 0.10;
 calc.printTip(); //prints 0.10
 }
}
```

```
public class Calculator{
 public double tip= 0.15; //initialized value 0.15
 public void printTip(){
 System.out.println(tip);
 }
}
```

# Ορίσματα Μεθόδων (arguments)

- Στο πρόγραμμα **Calculator**:

```
public class Calculator{
 public double tip= 0.15;//initialized value 0.15
 public void printTip(){
 System.out.println(tip);
 }
}
```

- Χρειάζονται 2 γραμμές για κάθε άτομο!
- Επίσης είναι επικίνδυνο να γράφουμε κώδικα ο οποίος αλλάζει πεδία απευθείας!
- Μπορούμε ωστόσο να περάσουμε **ορίσματα (arguments)** σε μία μέθοδο

# Ορίσματα (arguments) & Παράμετροι (parameters)

- **Όρισμα** είναι μια τιμή που περνά κατά τη κλήση μιας μεθόδου:

```
Calculator calc = new Calculator();
calc.calculate(3, 2.0); // εκτυπώνει 1.5
```

Arguments

- **Παράμετρος** είναι μια μεταβλητή που ορίζεται στη δήλωση μεθόδου:

```
public void calculate (int x, double y) {
 System.out.println(x/y);
}
```

Parameters

# Ορίσματα Μεθόδων (arguments)

- Η μέθοδος **calculate** είναι γραμμένη για να δεχθεί δύο ορίσματα:

- Το 1<sup>ο</sup> τύπου **int**
- Το 2<sup>ο</sup> τύπου **double**,

```
public void calculate(int x, double y){
 System.out.println(x/y); //prints 1.5
}
```

- Κλήση **calculate** →

```
Calculator calc = new Calculator();
calc.calculate(3, 2.0);
```

- Τι θα γίνει εάν αλλάξουμε τη σειρά;

```
Calculator calc = new Calculator();
calc.calculate(2.0, 3);
```

- Compiler error:

- int x cannot be assigned a double value!
- Η σειρά των ορισμάτων έχει σημασία!

# Παραδείγματα

```
public void calculate0() {
 System.out.println("No parameters");
}
```

```
public void calculate1(int x) {
 System.out.println(x/2.0);
}
```

```
public void calculate2(int x, double y) {
 System.out.println(x/y);
}
```

```
public void calculate3(int x, double y, int z) {
 System.out.println(x/y +z);
}
```

# Παραδείγματα

```
public void calculate0() {
 System.out.println("No parameters");
}
```

```
public void calculate1(int x) { calculate1(3)
 System.out.println(x/2.0); → 1.5
}
```

```
public void calculate2(int x, double y) {
 System.out.println(x/y); calculate2(3,2.0)
} → 1.5
```

```
public void calculate3(int x, double y, int z) {
 System.out.println(x/y +z); calculate3(3, 2.0, 2)
} → 3.5
```

# Εμβέλεια (scope)

- Οι παράμετροι της μεθόδου είναι μεταβλητές που έχουν εμβέλεια μέσα στη μέθοδο {μπλοκ κώδικα μεθόδου}

```
public void calculate2(int x, double y) {
 System.out.println(x/y);
}
```

Εμβέλεια  
του x

```
public void calculate() {
 System.out.println(x/2.0);
}
```

Εκτός εμβέλειας του x  
//τι είναι το x;

# Μεταβλητές εκτός εμβέλειας

- Πως θα βγάλω το **total** έξω από τη μέθοδο;

```
public class Calculator {
 public double tax = 0.05;
 public double tip= 0.15;

 public void findTotal(double price, String name){
 double total = price*(1+tax+tip);
 System.out.println(name +": $" +total);
 }
}
```

```
public class CalculatorTest{
 public static void main(String args[]){
 Calculator calc = new Calculator();
 System.out.println(calc.findTotal(10)+ calc.findTotal(12));
 }
}
```

# Μεταβλητές εκτός εμβέλειας

- Πως θα βγάλω το **total** έξω από τη μέθοδο;

```
public class Calculator {
 public double tax = 0.05;
 public double tip= 0.15;

 public void findTotal(double price, String name){
 double total = price*(1+tax+tip);
 System.out.println(name +": $" +total);
 }
}
```

```
public class CalculatorTest{
 public static void main(String args[]){
 Calculator calc = new Calculator();
 System.out.println(calc.findTotal(10)+ calc.findTotal(12));
 }
}
```

- Όμως:     '**void**' type not allowed here //μήνυμα λάθους

# Τύποι μεθόδου

- Οι μέθοδοι τύπου **void** δεν επιστρέφουν τιμές (δεν αποθηκεύονται τιμές μετά την κλήση μιας **void** μεθόδου).
- Οι μέθοδοι, όπως και οι μεταβλητές, μπορούν και επιστρέφουν τιμές διαφόρων τύπων π.χ. byte, short, int, long, double, String, boolean, κ.α.:
- Πως:
  - Δηλώστε τη μέθοδο να είναι κάποιου τύπου, και όχι **void**.
  - Χρησιμοποιήστε τη λέξη-κλειδί **return** μέσα σε μια μέθοδο, ακολουθούμενη από μια τιμή.

```
public String returnString() {
 return ("Hello");
}
```

```
public int sum(int x, int y) {
 return(x + y);
}
```

```
public boolean isGreater(int x, int y) {
 return(x > y);
}
```

# Χρήση παραμέτρων

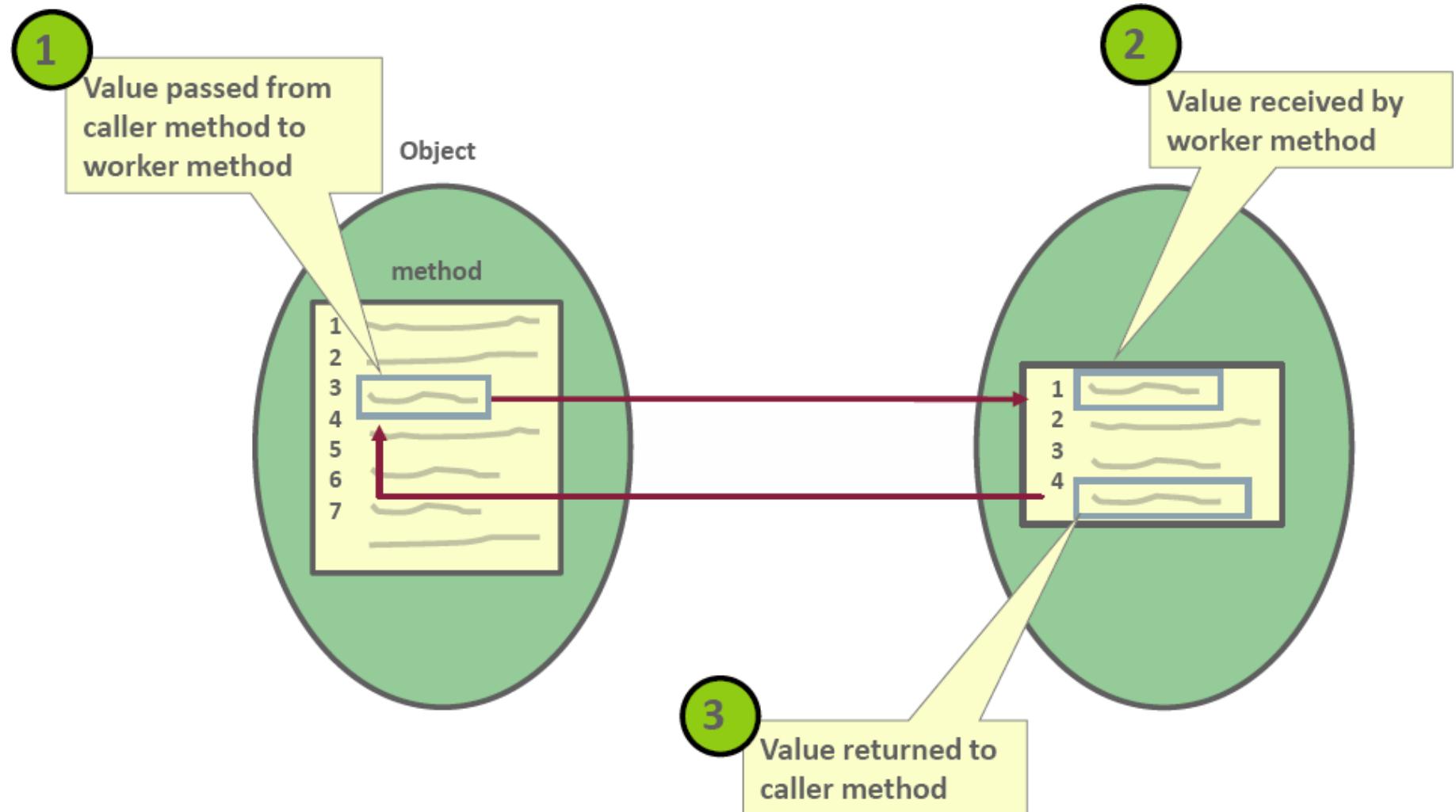
- Τι διαφορά έχουν τα παρακάτω:

```
public static void main(String[] args){
 int num1 = 1, num2 = 2;
 int result = num1 + num2;
 System.out.println(result);
}
```

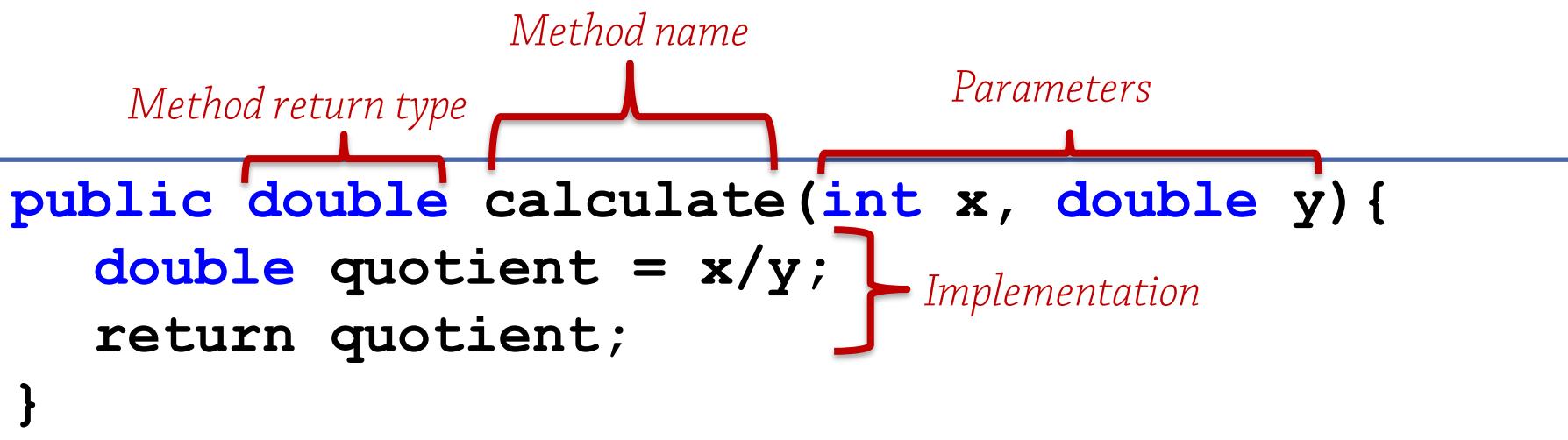
```
public class Athroisma {
 public static int sum(int x, int y) {
 return(x+y);
 }

 public static void main(String[] args) {
 int num1 = 1, num2 = 2;
 int result = sum(num1,num2);
 System.out.println(result);
 }
}
```

# Χρήση παραμέτρων



# Συνοψίζοντας...



```
Method name: calculate
Method return type: double
Parameters: int x, double y
Implementation: quotient = x/y; return quotient;
```

```
public double calculate(int x, double y) {
 double quotient = x/y;
 return quotient;
}
```

The diagram illustrates the structure of a Java method. A blue horizontal bar spans the width of the code. Red curly braces are placed above the code to identify its components: 'Method name' covers the entire method header 'calculate(int x, double y)', 'Method return type' covers the word 'double' before the header, 'Parameters' covers the parameters 'int x, double y', and 'Implementation' covers the body of the method starting with 'double quotient = x/y;'.

# Βιβλιοθήκες Java

# Βιβλιοθήκες Java

- Γιατί θα πρέπει να ανακαλύψουμε τον τροχό;

Αντί να ξαναγράψουμε τον ίδιο κώδικα Java για διάφορα προγράμματα...

...μπορούμε να χρησιμοποιήσουμε τη βιβλιοθήκη που παρέχεται από την Java, η οποία οργανώνει τον κώδικα που χρησιμοποιείται συχνά.

Αυτή η βιβλιοθήκη καλείται βιβλιοθήκη κλάσης Java.

Η τεκμηρίωση της βιβλιοθήκης:

<https://docs.oracle.com/javase/8/docs/api/>

# Πακέτα στη βιβλιοθήκη κλάσεων Java

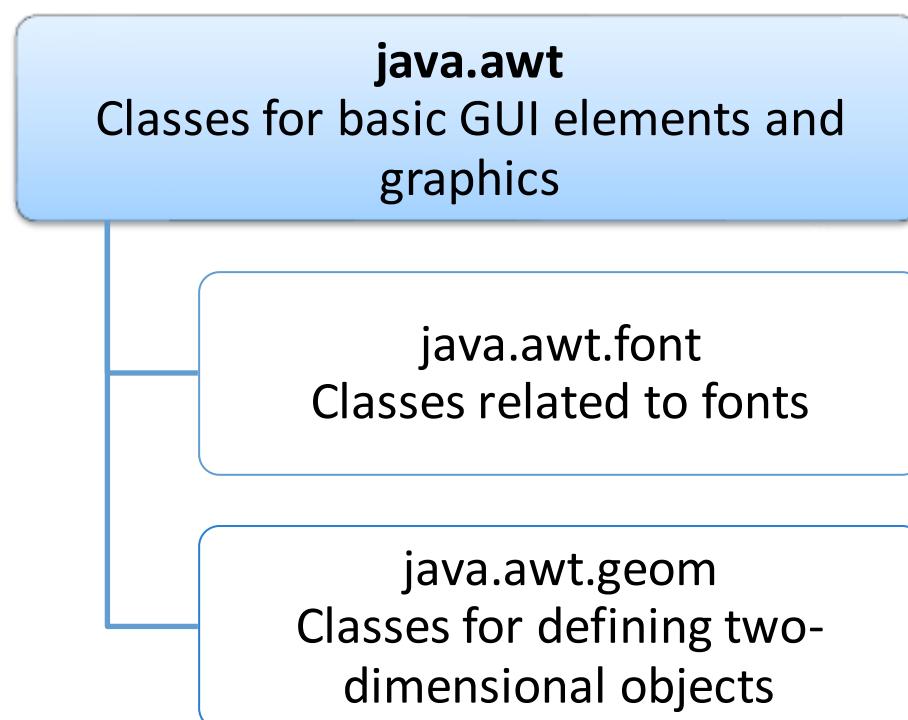
Οι κλάσεις της βιβλιοθήκης κλάσεων Java είναι οργανωμένες σε πακέτα.

Ένα πακέτο περιέχει μια ομάδα σχετικών κλάσεων.

| Πακέτο (Package)   | Σκοπός (Περιεχόμενο)                                                     |
|--------------------|--------------------------------------------------------------------------|
| <b>java.lang</b>   | Περιέχει κλάσεις που είναι θεμελιώδεις για το σχεδιασμό της γλώσσας Java |
| <b>javax.swing</b> | Περιέχει κλάσεις για την κατασκευή στοιχείων GUI                         |
| <b>java.net</b>    | Περιέχει κλάσεις για εφαρμογές δικτύωσης                                 |
| <b>java.time</b>   | Περιέχει κλάσεις για ημερομηνίες, ώρες, στιγμιότυπα και διάρκεια         |

# Πακέτα στη βιβλιοθήκη κλάσεων Java

Η τεράστια συλλογή των κλάσεων Java είναι οργανωμένη σε ιεραρχία δένδρου, η οποία επιτρέπει τα πακέτα (packages) να χωριστούν σε υποπακέτα (subpackages)



# Χρήση κλάσης πακέτου

- Για πρόσβαση σε μία κλάση ενός πακέτου θα πρέπει να οριστεί το πλήρες όνομα:

— π.χ. java.util.Scanner  
Package      Class Name

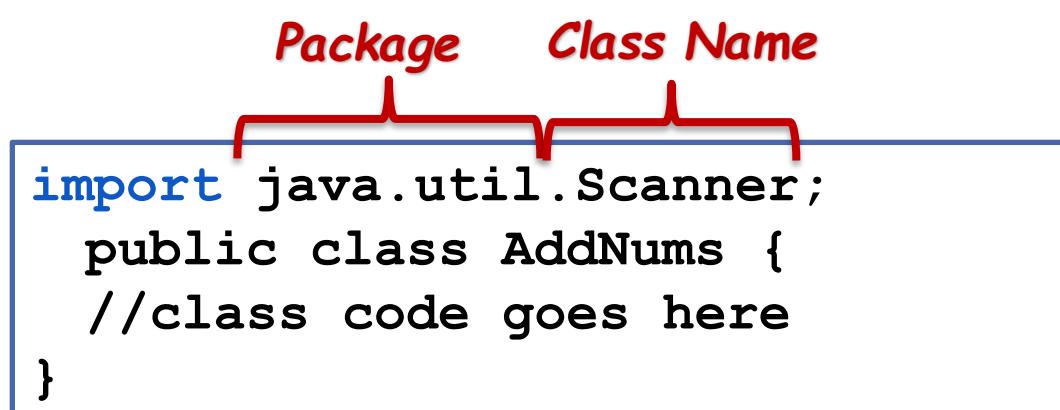
**Ωστόσο, έτσι δημιουργούνται  
μεγάλα ονόματα!**

```
public static void main(String[] args) {
 int num;
 java.util.Scanner keyboard = new java.util.Scanner(System.in);
 System.out.print("Enter a number");
 num= keyboard.nextInt();
 System.out.println("The number you entered is " + num);
}
```

# Χρήση *import*

- Μπορούμε να αποφύγουμε το πλήρες όνομα μίας κλάσης κατηγορίας με την εντολή *import* (import package.className), π.χ.

*Package*      *Class Name*



```
import java.util.Scanner;
public class AddNums {
 //class code goes here
}
```

# Πρόσβαση σε όλες τις κλάσεις ενός πακέτου

- Μπορείτε να εισαγάγετε όλες τις κλάσεις ενός πακέτου χρησιμοποιώντας τον χαρακτήρα "\*" (wildcard) στην εντολή εισαγωγής import.

```
import java.util.Date;
import java.util.Calendar;

public class DisplayDate {
 //class definition here
}
```

# Πρόσβαση σε όλες τις κλάσεις ενός πακέτου

- Μπορείτε να εισαγάγετε όλες τις κλάσεις ενός πακέτου χρησιμοποιώντας τον χαρακτήρα "\*" (wildcard) στην εντολή εισαγωγής import.

```
import java.util.*;

public class DisplayDate {
 //class definition here
}
```

# java.lang

- Τα παραπάνω δεν ισχύουν για τις κλάσεις του πακέτου java.lang...

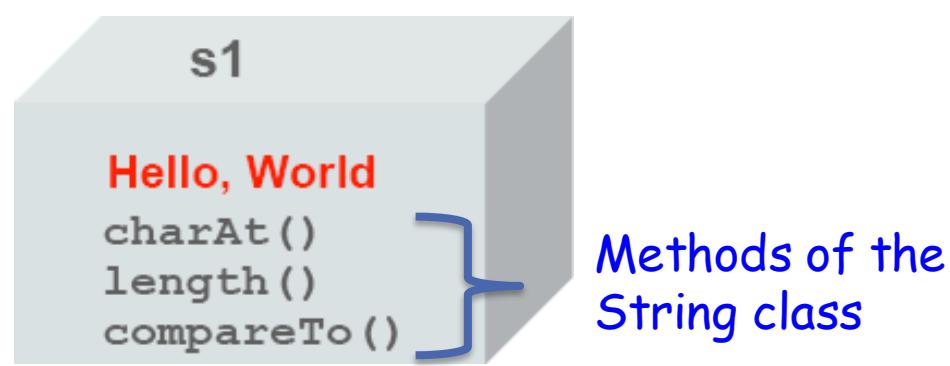
```
public class DisplayOutput {
 public static void main(String[] args) {
 System.out.println("Hello, how are you today?");
 }
}
```

...καθώς το java.lang εισάγεται **αυτόματα** σε όλα τα προγράμματα Java.

# Η κλάση String

# Strings

- Ακολουθίες χαρακτήρων (συμβολοσειρές)
  - “AaBb123,...?, etc.”
- Δεν αποτελούν μεταβλητές ενός βασικού τύπου μεταβλητών δεδομένων (primitive data type) όπως π.χ int, double, boolean, κ.α.
- Άλλα, αντικείμενα της κλάσης java.lang.String:
  - Π.χ. String s1= “Hello, World”;
- Τεκμηρίωση της κλάσης String:
  - <https://docs.oracle.com/javase/8/docs/api/>



# Τεκμηρίωση της κλάσης String

public int charAt(String str)

Return type of  
the method

Name of the  
method

Data type of the parameter that  
must be passed into the method

| Method Summary    |                                                                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Modifier and Type | Method and Description                                                                                                                        |
| char              | <b>charAt</b> (int index)<br>Returns the char value at the specified index.                                                                   |
| int               | <b>codePointAt</b> (int index)<br>Returns the character (Unicode code point) at the specified index.                                          |
| int               | <b>codePointBefore</b> (int index)<br>Returns the character (Unicode code point) before the specified index.                                  |
| int               | <b>codePointCount</b> (int beginIndex, int endIndex)<br>Returns the number of Unicode code points in the specified text range of this String. |
| int               | <b>compareTo</b> (String anotherString)<br>Compares two strings lexicographically.                                                            |
| int               | <b>compareToIgnoreCase</b> (String str)<br>Compares two strings lexicographically, ignoring case differences.                                 |
| String            | <b>concat</b> (String str)<br>Concatenates the specified string to the end of this string.                                                    |

# Μέθοδοι (χαρακτηριστικές) του αντικειμένου String

| Μέθοδος                                          | Λειτουργία                                                                                                                                                                                                                           |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| charAt(int index)                                | Επιστρέφει το χαρακτήρα στη θέση index                                                                                                                                                                                               |
| compareTo(String anotherString)                  | Συγκρίνει δύο strings. Επιστρέφει:<br>< 0, όταν string που καλεί τη μέθοδο είναι λεξικογραφικά πρώτο.<br>== 0, όταν τα 2 strings είναι λεξικογραφικά ίδια<br>> 0, όταν η παράμετρος που περνάει στη μέθοδο είναι λεξικογραφικά πρώτη |
| contains(CharSequence s)                         | Επιστρέφει true μόνο εάν το string περιέχει τη σειρά χαρακτήρων s                                                                                                                                                                    |
| indexOf(int ch)                                  | Επιστρέφει τη θέση στην οποία θα συναντήσει για πρώτη φορά το χαρακτήρα ch στο string                                                                                                                                                |
| isEmpty()                                        | Επιστρέφει true, εάν το string είναι κενό (δηλαδή έχει μηδενικό μήκος)                                                                                                                                                               |
| Length()                                         | Επιστρέφει το μήκος ενός string                                                                                                                                                                                                      |
| replace(char oldChar, char newChar)              | Αντικαθιστά όλες τις εμφανίζεις του oldChar με newChar.                                                                                                                                                                              |
| replaceFirst(String pattern, String replacement) | Αντικαθιστά μόνο την πρώτη εμφάνιση του Stringpattern στο String                                                                                                                                                                     |
| toLowerCase() / toUpperCase()                    | Μετατρέπει όλους τους χαρακτήρες στο string σε lowercase/uppercase                                                                                                                                                                   |
| str.substring(int beginIdx)                      | Επιστρέφει το substring από το beginIdx ως το τέλος του str                                                                                                                                                                          |
| str.substring(int beginIdx,int endIdx)           | Επιστρέφει το substring από το beginIdx ως το endIdx                                                                                                                                                                                 |

# Παραδείγματα

| Κώδικας Java                                                                                                                                                                                                           | Αποτέλεσμα |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| <pre>String name= "Mike.W"; System.out.println(name.length());</pre>                                                                                                                                                   |            |
| <pre>String str= "Hello, World"; str.indexOf(char l)</pre>                                                                                                                                                             |            |
| <pre>String str="Susan"; System.out.println(str.charAt(0)); System.out.println(str.charAt(3));</pre>                                                                                                                   |            |
| <pre>String phoneNum= "404-543-2345"; int idx1 = phoneNum.indexOf('-'); System.out.println("index of first dash: "+ idx1); int idx2 = phoneNum.indexOf('-', idx1); System.out.println("second dash idx: "+idx2);</pre> |            |
| <pre>String greeting = "Hello, World!"; String sub = greeting.substring(0, 5); String tail = greeting.substring(7);</pre>                                                                                              |            |

# Παραδείγματα

| Κώδικας Java                                                                                                                                                                                                           | Αποτέλεσμα                                   |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|
| <pre>String name= "Mike.W"; System.out.println(name.length());</pre>                                                                                                                                                   | 6                                            |
| <pre>String str= "Hello, World"; str.indexOf(char l)</pre>                                                                                                                                                             | 2                                            |
| <pre>String str="Susan"; System.out.println(str.charAt(0)); System.out.println(str.charAt(3));</pre>                                                                                                                   | S<br>a                                       |
| <pre>String phoneNum= "404-543-2345"; int idx1 = phoneNum.indexOf('-'); System.out.println("index of first dash: "+ idx1); int idx2 = phoneNum.indexOf('-', idx1); System.out.println("second dash idx: "+idx2);</pre> | index of first dash: 3<br>second dash idx: 7 |
| <pre>String greeting = "Hello, World!"; String sub = greeting.substring(0, 5); String tail = greeting.substring(7);</pre>                                                                                              | Hello<br>World!                              |

# Παραδείγματα

| Κώδικας Java                                                                                                                                                                                 | Αποτέλεσμα                                                                                  |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <pre>String str= "Using String replace to replace character"; String newString =str.replace("r", "R"); System.out.println(newString);</pre>                                                  | Using StRing Replace to Replace chaRacteR                                                   |
| <pre>String replace = "String replace with replaceFirst"; String newString = replace.replaceFirst("re", "RE"); System.out.println(newString);</pre>                                          | String REplacewith replaceFirst                                                             |
| <pre>String s1 = "Susan"; String s2 = "Susan"; String s3 = "Robert"; System.out.println(s1.compareTo(s2)); System.out.println(s1.compareTo(s3)); System.out.println(s3.compareTo(s1));</pre> | 0 // s1 είναι ίδιο με s2<br>1 // το 'S' ακολουθεί το 'R'<br>-1 // το 'R' προηγείται του 'S' |

# StringBuffer

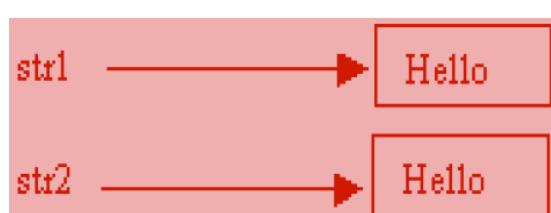
- Για να έχουμε τη δυνατότητα να τροποποιήσουμε ένα αλφαριθμητικό, χρησιμοποιούμε την κλάση StringBuffer.
  - Τα αντικείμενα τύπου StringBuffer μπορούν να περιέχουν αχρησιμοποίητο χώρο για χαρακτήρες (ενώ τα τύπου String δεν μπορούν).
  - StringBuffer sb = new StringBuffer(10) → κρατάει χώρο για 10 χαρακτήρες στο αντικείμενο sb.
- Οι μέθοδοι length() και capacity() μας δείχνουν το μήκος και τη χωρητικότητα ενός αντικειμένου τύπου StringBuffer.
- Μπορούμε να μετατρέψουμε ένα αντικείμενο StringBuffer σε String, με χρήση της μεθόδου toString().
- Η StringBuffer παρέχει και τη δυνατότητα αντιστροφής χαρακτήρων με τη μέθοδο reverse().

# Tα Strings είναι αμετάβλητα

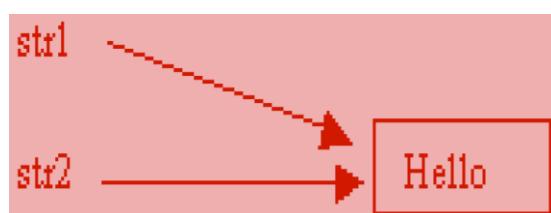
- Το String object είναι αμετάβλητο, δηλαδή, αφού δημιουργηθεί, η τιμή του δεν μπορεί να αλλάξει.
- Για αυτό και η Java μπορεί να τα επεξεργαστεί αποτελεσματικά.
- Για παράδειγμα
  - Εάν

```
String str1 = "Hello";
String str2 = "Hello";
```

– τότε



– ωστόσο



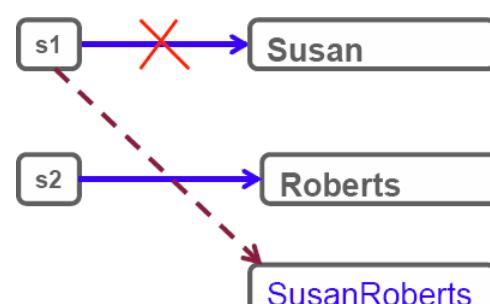
Το JRE γνωρίζει ότι τα δύο Strings είναι ίδια οπότε δεσμεύει την ίδια θέση μνήμης και για τα δύο αντικείμενα.

# Συνένωση (Concatenating) Strings

- Μπορεί να γίνει με δύο τρόπους:

- με τον τελεστή “+”

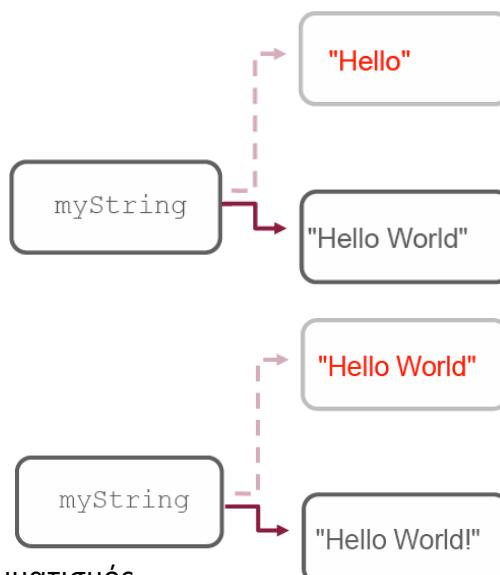
```
String s1 = "Susan";
String s2 = "Roberts";
s1= s1+s2;
System.out.println(s1);
```



- με τη μέθοδο concat ()

```
String myString = "Hello";
myString = myString.concat("World");
...

myString = myString + "!"
```



# concat () Vs “+”

- Ο τελεστής “+”:
  - Μπορεί να λειτουργήσει μεταξύ ενός string αντικειμένου και των char, int, double ή float τύπων μεταβλητών.
  - Μετατρέπει την τιμή σε string πριν από την ένωση/άθροισμα.
- Η μέθοδος concat ():
  - Μπορεί να καλείται μόνο σε strings.
  - Ελέγχει για συμβατότητα των τύπων δεδομένων και αναπαράγει σφάλμα μεταγλώττισης (compile time error) αν δεν ταιριάζουν.

# Η κλάση Random



# Τυχαίοι Αριθμοί

- Η κλάση `java.util.Random` χρησιμοποιείται για την παραγωγή τυχαίων αριθμών για διάφορες εφαρμογές π.χ. μοίρασμα τράπουλας, λοταρία, κ.α.
- Περιέχει μεθόδους που επιστρέφουν τυχαίους αριθμούς τύπου `double`, `boolean`, `float`, `long`.



| Method                              | Output                                                      |
|-------------------------------------|-------------------------------------------------------------|
| <code>boolean nextBoolean();</code> | True/false τιμή                                             |
| <code>int nextInt()</code>          | Integer τιμή μεταξύ Integer.MIN_VALUE και Integer.MAX_VALUE |
| <code>long nextLong()</code>        | Long τιμή μεταξύ Long.MIN_VALUE και Long.MAX_VALUE          |
| <code>float nextFloat()</code>      | Float number $\geq 0.0$ και $< 1.0$                         |
| <code>double nextDouble()</code>    | Double number $\geq 0.0$ και $< 1.0$                        |

# Παραδείγματα

```
import java.util.Random;
public class RandomNum{
 public static void main(String[] args){
 Random rndNum= new Random();
 int randomNum= rndNum.nextInt();
 System.out.println("Random Number: " + randomNum);
 }
}
```

Output

Random Number: 1660093261

Εκτυπώνει έναν τυχαίο  
ακέραιο αριθμό κάθε φορά

```
import java.util.Random;
public class RandomNum{
 public static void main(String[] args) {
 Random num= new Random();
 System.out.println("Random Number 1: "+num.nextInt());
 System.out.println("Random Number 2: "+num.nextInt());
 System.out.println("Random Number 3: "+num.nextInt());
 System.out.println("Random Number 4: "+num.nextInt());
 System.out.println("Random Number 5: "+num.nextInt());
 }
}
```

Output

Random Number 1: 1814918663  
Random Number 2: -944814285  
Random Number 3: 767298538  
Random Number 4: 762007235  
Random Number 5: 1220127792

Εκτυπώνει μία ακολουθία  
τυχαίων ακεραίων αριθμών κάθε  
φορά

# Παραδείγματα

```
import java.util.Random;
public class RandomNum{
 public static void main(String[] args){
 Random num = new Random();
 double randomDouble = num.nextDouble();
 System.out.println("Random Number: " + randomDouble);
 }
}
```



Random Number: 0.8502641005640065



Εκτυπώνει έναν τυχαίο  
πραγματικό αριθμό κάθε φορά

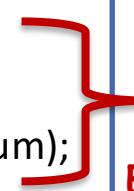
# Παραδείγματα

```
import java.util.Random;
public class RandomNum{
 public static void main(String[] args) {

 Random rand1 = new Random();
 int randomnum = rand1.nextInt(20);
 System.out.println("Random Number: " + randomnum);

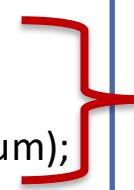
 Random rand2 = new Random();
 randomnum = rand2.nextInt(40)+1;
 System.out.println("Random Number: " + randomnum);

 Random rand3 = new Random();
 randomnum = rand3.nextInt(31)+5;
 System.out.println("Random Number: " + randomnum);
 }
}
```



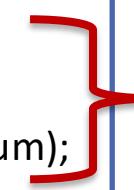
Random Number: 17

ΕΚΤΥΠΩΝΕΙ έναν τυχαίο ακέραιο αριθμό από 0 έως 19



Random Number: 29

ΕΚΤΥΠΩΝΕΙ έναν τυχαίο ακέραιο αριθμό από 1 έως 40



Random Number: 24

ΕΚΤΥΠΩΝΕΙ έναν τυχαίο ακέραιο αριθμό από 5 έως 35

# Γεννήτρια τυχαίων αριθμών

- Μερικές φορές χρειάζεται να δημιουργηθεί η ίδια ακολουθία τυχαίων αριθμών κάθε φορά, χρησιμοποιώντας μια αρχική σταθερή τιμή που ονομάζεται γεννήτρια.
- Όταν δημιουργείτε ένα αντικείμενο της κλάσης Random, περάστε έναν σταθερό ακέραιο για να καθορίσετε τη γεννήτρια:
  - `Random rndNumbers = new Random(20L);`
- Γεννήτρια (seed)
- Με τη μέθοδο setSeed(), μπορείτε να αλλάξετε τη γεννήτρια.
- Ωστόσο, κάθε φορά που ορίζετε την ίδια γεννήτρια, επιστρέφεται η ίδια τυχαία ακολουθία.

# Παράδειγμα

```
public static void main(String[] args) {
 Random rand = new Random(20L);
 System.out.println("Random Number 1: " + rand.nextInt(100));
 System.out.println("Random Number 2: " + rand.nextInt(100));
 System.out.println("Random Number 3: " + rand.nextInt(100));
 System.out.println("Changing seed to change to sequence");
 rand.setSeed(5L);
 System.out.println("Random Number 4: " + rand.nextInt(100));
 System.out.println("Random Number 5: " + rand.nextInt(100));
 System.out.println("Random Number 6: " + rand.nextInt(100));
 System.out.println("Setting seed 20 produce previous
 sequence");
 rand.setSeed(20L);
 System.out.println("Random Number 7: " + rand.nextInt(100));
 System.out.println("Random Number 8: " + rand.nextInt(100));
 System.out.println("Random Number 9: " + rand.nextInt(100));
}
```



Random Number 1: 53  
Random Number 2: 36  
Random Number 3: 1  
Changing seed to change to sequence  
Random Number 4: 87  
Random Number 5: 92  
Random Number 6: 74  
Setting seed 20 produce previous sequence  
Random Number 7: 53  
Random Number 8: 36  
Random Number 9: 1

# Η κλάση Math

$$\pi$$

# java.lang.Math

- Η κλάση Math περιέχει μεθόδους για την εκτέλεση μαθηματικών συναρτήσεων και πράξεων π.χ. μέγιστο/ελάχιστο δύο τιμών, στρογγυλοποίηση, λογαριθμικές/τριγωνομετρικές πράξεις, τετραγωνική ρίζα, κ.α.
- Τεκμηρίωση της κλάσης Math: Περιλαμβάνεται στο πακέτο java.lang
- <http://docs.oracle.com/javase/8/docs/api/index.html>

# Ενδεικτικές Μέθοδοι της κλάσης Math

| Method Name         | Description                                |
|---------------------|--------------------------------------------|
| abs(value)          | απόλυτη τιμή                               |
| ceil(value)         | στρογγυλοποίηση (προς τα πάνω)             |
| cos(value)          | συνημίτονο (σε rads)                       |
| floor(value)        | στρογγυλοποίηση (προς τα κάτω)             |
| log(value)          | λογάριθμος                                 |
| log10(value)        | δεκαδικός λογάριθμος                       |
| max(value1, value2) | μεγαλύτερη τιμή από δύο τιμές              |
| min(value1, value2) | μικρότερη τιμή από δύο τιμές               |
| pow(base, exponent) | εκθετική (exponent) τιμή της βάσης (base)  |
| random()            | πραγματικός τυχαίος αριθμός μεταξύ 0 και 1 |
| round(value)        | πλησιέστερος ακέραιος αριθμός              |
| sin(value)          | ημίτονο (σε rads)                          |
| sqrt(value)         | τετραγωνική ρίζα                           |

# Μέθοδοι της κλάσης Math

- Οι μέθοδοι της κλάσης Math είναι **static**
  - Μπορούν να χρησιμοποιηθούν μέσω του ονόματος της κλάσης.
  - Δεν χρειάζεται να δημιουργηθεί αντικείμενο του Math class, πρώτα, όπως κάναμε στην Random.
  - Σύνταξη: Math.methodName(parameters)

```
Math.sqrt(121.0) ; // δεν χρειάζεται να δημιουργηθεί
 // αντικείμενο Math πρώτα
```

```
Random rndNum= new Random() ; // πρέπει να δημιουργηθεί
int randomNum = rndNum.nextInt() ; // αντικείμενο Random πρώτα
```

# Μέθοδοι της κλάσης Math

- Υπολογίζουν κάποιο αριθμητικό αποτέλεσμα αλλά δεν το εκτυπώνουν:

```
public static void main(String[] args) {
 Math.sqrt(121.0); //δεν εκτυπώνει κάτι
}
```

- ...συνεπώς:

```
public static void main(String[] args) {
 System.out.println("Square root: "+ Math.sqrt(121.0)); //εκτυπώνει 11.0
}
```

- ..ή

```
public static void main(String[] args) {
 double sqroot= Math.sqrt(121.0); // αποθηκεύει και
 System.out.println("Square root: "+sqroot); //εκτυπώνει 11.0
}
```

# Τι εκτυπώνει το παρακάτω;

```
public class Calculate{
 public static void main(String[] args) {
 double result = Math.min(3, 7) + Math.abs(-50);
 System.out.println("Result is " + result);
 }
}
```

# Τι εκτυπώνει το παρακάτω;

```
public class Calculate{
 public static void main(String[] args) {
 double result = Math.min(3, 7) + Math.abs(-50);
 System.out.println("Result is " + result);
 }
}
```

Result is 53.0

# Τι υπολογίζουν τα παρακάτω;

- `Math.abs (-1.23)`
- `Math.pow (3, 2)`
- `Math.sqrt (121.0)`
- `Math.sqrt (256.0)`
- `Math.abs (Math.min (-3, -5))`

# Τι υπολογίζουν τα παρακάτω;

- |                                |      |
|--------------------------------|------|
| • Math.abs (-1.23)             | 1.23 |
| • Math.pow (3, 2)              | 9.0  |
| • Math.sqrt (121.0)            | 11.0 |
| • Math.sqrt (256.0)            | 16.0 |
| • Math.abs (Math.min (-3, -5)) | 5.0  |

# Σταθερές

- Η κλάση Math περιέχει δύο σταθερές:

| Field   | Description  |
|---------|--------------|
| Math.E  | 2.7182818... |
| Math.PI | 3.1415926... |

- Το PI σε ένα πρόγραμμα, χρησιμοποιείται ως εξής:
  - Math.PI

# Υπολογισμός επιφάνειας κύκλου

```
import java.util.Scanner;
public class AreaOfCircle {
 public static void main(String args[]) {
 Scanner sc = new Scanner(System.in);
 System.out.print("Enter the radius: ");
 double radius = sc.nextDouble();
 double area = Math.PI* radius * radius;
 System.out.println("The area of circle is: " + area);
 }
}
```

# Δομές επιλογής

- Παράδειγμα
  - Ας πούμε ότι οδηγείτε για το Πανεπιστήμιο. Σταματάτε σε μια διασταύρωση όπου θα πρέπει να πάρετε μία απόφαση:
    - Να στρίψετε αριστερά,
    - ή δεξιά,για να φτάσετε στον προορισμό σας;



Υπάρχουν μόνο δύο απαντήσεις σε κάθε μία από τις ερωτήσεις: **ναι** ή **όχι**

# Μεταβλητές Boolean

- Στη Java, οι τιμές για τον τύπο **boolean** είναι

- **Αληθής (true)**
- **Ψευδής (false)**

```
public static void main(String args[]) {
 boolean passed, largeVenue, grade;
 passed = true; } δήλωση
 largeVenue= false; } εκχώρηση τιμής
 grade = passed; }
 System.out.println(passed); } εκτύπωση τιμών
 System.out.println(largeVenue); }
 System.out.println(grade); }
}
```

- Αντιστοίχιση σε μια μεταβλητή τύπου boolean:

```
int x=5;
System.out.println(x == 5);
boolean isFive = x == 5; //η isFive είναι true
x = 4;
boolean isFive = x == 5; //η isFive είναι false
```

# Παράδειγμα: Τι εκτυπώνει το παρακάτω;

```
boolean res1 = 24 == 15;
System.out.println("res1: " + res1);
int value1 = 15;
int value2 = 24;
boolean res2 = value1 == value2;
System.out.println("res2: " + res2);
```

# Παράδειγμα: Τι εκτυπώνει το παρακάτω;

```
boolean res1 = 24 == 15;
System.out.println("res1: " + res1);
int value1 = 15;
int value2 = 24;
boolean res2 = value1 == value2;
System.out.println("res2: " + res2);
```



res1: false  
res2: false

# Σχεσιακοί τελεστές

| Συνθήκη                   | Τελεστής           | Παράδειγμα                                        |
|---------------------------|--------------------|---------------------------------------------------|
| Είναι ίσο με              | <code>==</code>    | <code>int i=1;</code><br><code>(i == 1)</code>    |
| Δεν είναι ίσο με          | <code>!=</code>    | <code>int i=2;</code><br><code>(i != 1)</code>    |
| Είναι μικρότερο από       | <code>&lt;</code>  | <code>int i=0;</code><br><code>(i &lt; 1)</code>  |
| Είναι μικρότερο ή ίσο με  | <code>&lt;=</code> | <code>int i=1;</code><br><code>(i &lt;= 1)</code> |
| Είναι μεγαλύτερο από      | <code>&gt;</code>  | <code>int i=2;</code><br><code>(i &gt; 1)</code>  |
| Είναι μεγαλύτερο ή ίσο με | <code>&gt;=</code> | <code>int i=1;</code><br><code>(i &gt;= 1)</code> |

# Παράδειγμα

```
public static void main(String args[]) {
 int a = 10;
 int b = 20;
 System.out.println(a == b);
 System.out.println(a != b);
 System.out.println(a > b);
 System.out.println(a < b);
 System.out.println(b >= a);
 System.out.println(b <= a);
}
```

# Παράδειγμα

```
public static void main(String args[]) {
 int a = 10;
 int b = 20;
 System.out.println(a == b);
 System.out.println(a != b);
 System.out.println(a > b);
 System.out.println(a < b);
 System.out.println(b >= a);
 System.out.println(b <= a);
}
```

false  
true  
false  
true  
true  
false

## Προσοχή:

- = για εκχώρηση τιμής
- == για σύγκριση τιμών και επιστροφή boolean τιμής

# Έλεγχος Συνθήκης

- Επιλέγει ποια εντολή θα εκτελεστεί στη συνέχεια.
- Η απόφαση βασίζεται σε boolean εκφράσεις (συνθήκες) που αξιολογούνται ως αληθείς ή ψευδείς.
- Οι εντολές ελέγχου μπορεί να είναι της μορφής
  - **if**
  - **if/else**
  - **switch**

# Εντολή if

- Αποτελείται από μια έκφραση boolean που ακολουθείται από μία ή περισσότερες εντολές δηλώσεις.

*Boolean έκφραση*

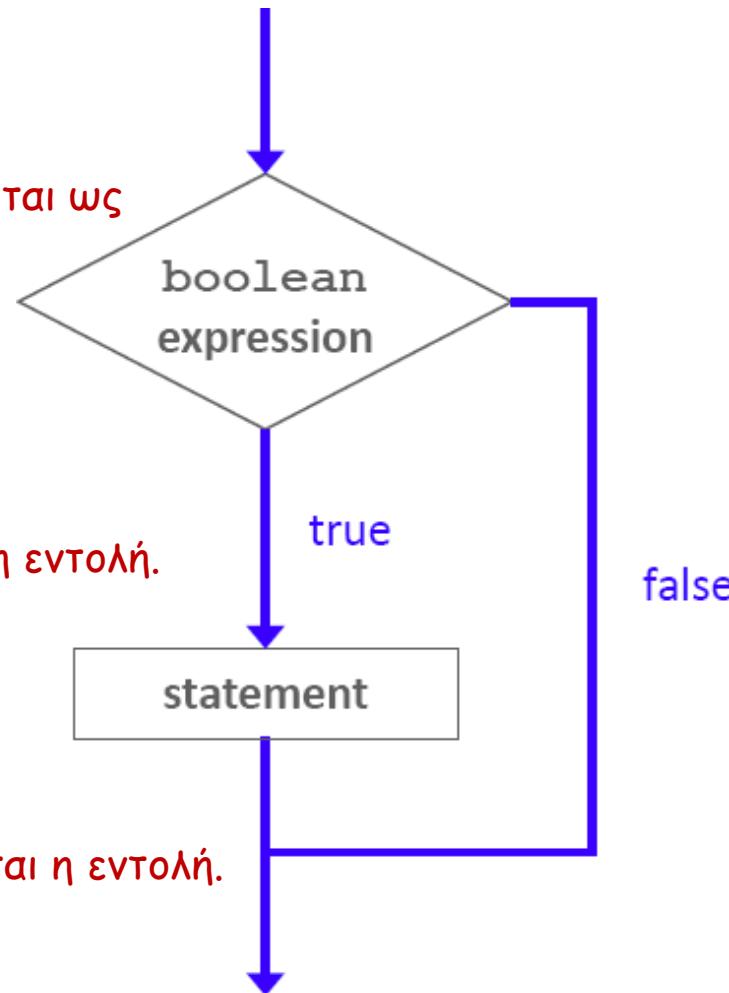
```
if (<some condition is true>) {
 // Οι εντολές θα εκτελεστούν αν
 // η boolean έκφραση είναι αληθής
}
```

# Διάγραμμα ροής

Η έκφραση boolean αξιολογείται ως αληθής ή ψευδής.

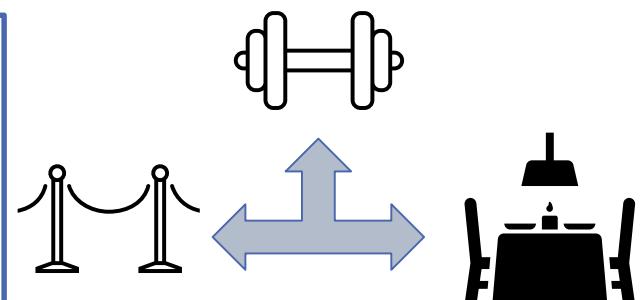
Αν ισχύει:  
Εκτελείται η εντολή.

Αν είναι ψευδής:  
Παραλείπεται η εντολή.



# Παράδειγμα: Που πάω για το γυμναστήριο;

```
public static void main(String args[]) {
 String left = "museum";
 String straight = "gym";
 String right = "restaurant";
 if (left == "gym") {
 System.out.println("Turn Left");
 }
 if (straight== "gym") {
 System.out.println("Drive Straight");
 }
 if (right == "gym") {
 System.out.println("Turn Right");
 }
}
```



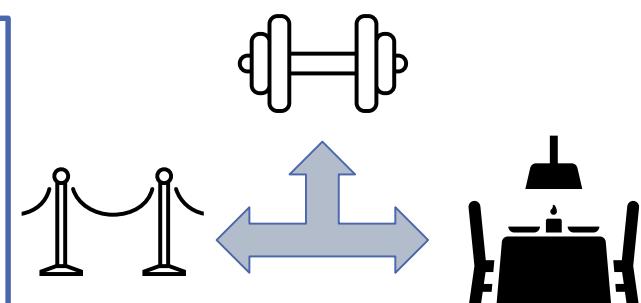
## **Παράδειγμα: Που πάω για το γυμναστήριο;**

```
public static void main(String args[]) {
 String left = "museum";
 String straight = "gym";
 String right = "restaurant";
 if (left == "gym") {
 System.out.println("Turn Left");
 }
 if (straight == "gym") {
 System.out.println("Drive Straight");
 }
 if (right == "gym") {
 System.out.println("Turn Right");
 }
}
```

False

True

False



## Drive Straight

## Άσκηση 5.2: ChkOddEven

- Ανοίξτε το ChkOddEven.java και υλοποιήστε τα εξής:
  - Εισάγετε έναν αριθμό μεταξύ 1 και 10.
  - Χρησιμοποιήστε if εντολή για να ελέγχετε εάν ένας αριθμός είναι μονός ή ζυγός.
- Το πρόγραμμα πρέπει να παράγει το ακόλουθο αποτέλεσμα:

```
run test
Enter a number:
[8]
The num 8 is even
```

# Εντολή if/else

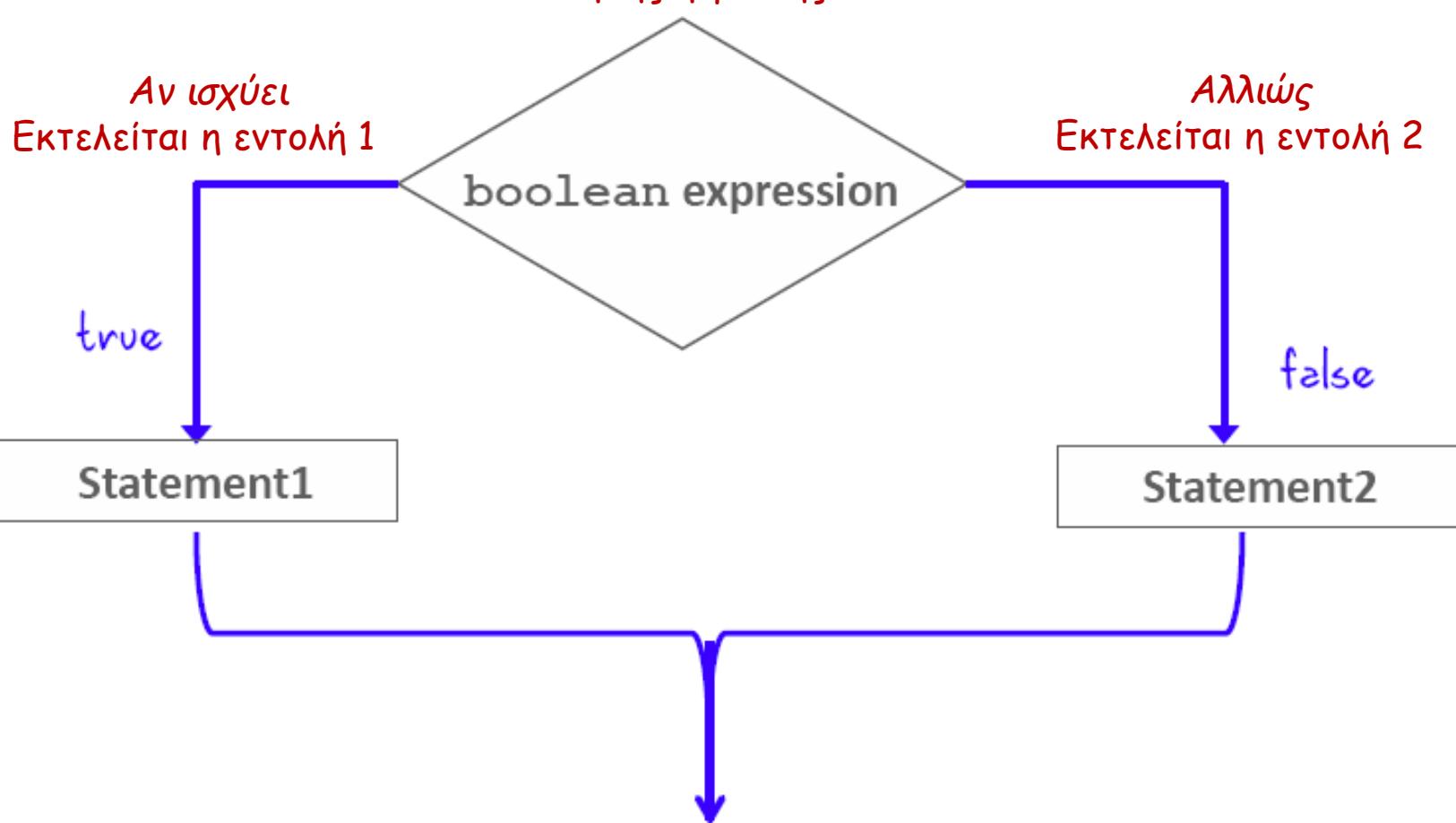
- Επιλογή μεταξύ 2 εναλλακτικών

*Boolean έκφραση*

```
if (<κάποια συνθήκη είναι αληθής>) {
 // κάνε κάτι } if block
}
else{
 // κάνε κάτι άλλο } else block
}
```

# Διάγραμμα ροής (διακλάδωση)

Η έκφραση boolean αξιολογείται ως  
αληθής ή ψευδής.



# Παράδειγμα: Τι εκτυπώνει το παρακάτω;

```
public class passMark {
 public static void test(int x){
 if (x >= 5){
 System.out.print("Βαθμός: "+x);
 System.out.println(" Πέρασες το μάθημα :-)");
 }
 else {
 System.out.print("Βαθμός: "+x);
 System.out.println(" Κόπηκες :-(");
 }
 }
 public static void main(String[] arguments){
 test(6);
 test(5);
 test(4);
 }
}
```

# Παράδειγμα: Τι εκτυπώνει το παρακάτω;

```
public class passMark {
 public static void test(int x){
 if (x >= 5){
 System.out.print("Βαθμός: "+x);
 System.out.println(" Πέρασες το μάθημα :-)");
 }
 else {
 System.out.print("Βαθμός: "+x);
 System.out.println(" Κόπηκες :-(");
 }
 }

 public static void main(String[] arguments){
 test(6);
 test(5);
 test(4);
 }
}
```

Βαθμός: 6 Πέρασες το μάθημα :-)  
Βαθμός: 5 Πέρασες το μάθημα :-)  
Βαθμός: 4 Κόπηκες :-)

# Σύγκριση Αντικειμένων

- Σχεσιακοί τελεστές == είναι χρήσιμοι για τη σύγκριση βασικών τύπων μεταβλητών.
- Προσοχή όμως στις μεταβλητές αντικειμένων.
- Για παράδειγμα, τι τυπώνουν τα παρακάτω;

```
int x = 3;
int y = 2;
int z = x + y;
boolean test = (z == x + y);
System.out.println(test);
```

```
String x = "Ora";
String y = "cle";
String z = x + y;
boolean test = (z == x + y);
System.out.println(test);
```

# Σύγκριση Αντικειμένων

- Σχεσιακοί τελεστές == είναι χρήσιμοι για τη σύγκριση βασικών τύπων μεταβλητών.
- Προσοχή όμως στις μεταβλητές αντικειμένων.
- Για παράδειγμα, τι τυπώνουν τα παρακάτω;

```
int x = 3;
int y = 2;
int z = x + y;
boolean test = (z == x + y);
System.out.println(test);
```



true

```
String x = "Ora";
String y = "cle";
String z = x + y;
boolean test = (z == x + y);
System.out.println(test);
```



false

# Σύγκριση Αντικειμένων

- Σχεσιακοί τελεστές == είναι χρήσιμοι για τη σύγκριση βασικών τύπων μεταβλητών.
- Προσοχή όμως στις μεταβλητές αντικειμένων.
- Για παράδειγμα, τι τυπώνουν τα παρακάτω;

```
int x = 3;
int y = 2;
int z = x + y;
boolean test = (z == x + y);
System.out.println(test);
```



true

```
String x = "Ora";
String y = "cle";
String z = x + y;
boolean test = (z == x + y);
System.out.println(test);
```



false

Γιατί:

# Σύγκριση Αντικειμένων

- Σχεσιακοί τελεστές == είναι χρήσιμοι για τη σύγκριση βασικών τύπων μεταβλητών.
- Προσοχή όμως στις μεταβλητές αντικειμένων.
- Για παράδειγμα, τι τυπώνουν τα παρακάτω;

```
int x = 3;
int y = 2;
int z = x + y;
boolean test = (z == x + y);
System.out.println(test);
```



true

Συγκρίνει τις  
τιμές των  
μεταβλητών

```
String x = "Ora";
String y = "cle";
String z = x + y;
boolean test = (z == x + y);
System.out.println(test);
```



false

Συγκρίνει τις  
θέσεις των  
αντικειμένων  
στη Μνήμη

# Σύγκριση Αντικειμένων

- Δεν θα πρέπει να συγκρίνουμε Strings χρησιμοποιώντας “==”,
- ...αλλά χρησιμοποιώντας τη μέθοδο equals() (ανήκει στην κλάση java.lang.String).
- Δέχεται μία παράμετρο τύπου String και ελέγχει αν το περιεχόμενο των Strings είναι όμοιο, επιστρέφοντας μία boolean τιμή.

```
String x = "Ora";
String y = "cle";
String z = x +y;
boolean test = z.equals(x + y);
System.out.println(test);
```



true

- Επίσης, υπάρχει και η παρόμοια μέθοδος: equalsIgnoreCase()

# Λογικοί τελεστές

# Λογικοί τελεστές

- Ένας πιο εύκολος τρόπος για τις πολλαπλές συνθήκες είναι οι **λογικοί τελεστές**.

| Τελεστής | Περιγραφή  |
|----------|------------|
| &&       | Λογικό ΚΑΙ |
|          | Λογικό Ή   |
| !        | Άρνηση     |

- Μπορείτε να τους χρησιμοποιήσετε για να συνδυάσετε πολλαπλές εκφράσεις boolean σε μια.

# Λογικοί τελεστές: Παραδείγματα

| Λειτουργία                                       | Τελεστής | Παράδειγμα                                                          |
|--------------------------------------------------|----------|---------------------------------------------------------------------|
| Πρέπει να ισχύουν και οι 2 συνθήκες              | &&       | <pre>int i = 2; int j = 8; ((i &lt; 1) &amp;&amp; (j &gt; 6))</pre> |
| Πρέπει να ισχύει 1 από τις 2 ή και οι 2 συνθήκες |          | <pre>int i = 2; int j = 8; ((i &lt; 1)    (j &gt; 10))</pre>        |
| Πρέπει να μην ισχύει η συνθήκη                   | !        | <pre>int i = 2; (!(i &lt; 3))</pre>                                 |

# Στο προηγούμενο παράδειγμα..

```
public static void main(String[] args) {
 int numberDaysAbsent= 0;
 int grade = 95;
 if (grade >= 88 && numberDaysAbsent== 0) {
 System.out.println("Κέρδισες την Υποτροφία!");
 }
 else {
 System.out.println("Δεν μπορείς να πάρεις υποτροφία");
 }
}
```

# Παράλειψη ελέγχων

- Στον `&&` τελεστή, αν η 1<sup>η</sup> έκφραση (στα αριστερά) είναι ψευδής, δεν χρειάζεται να αξιολογηθεί η 2<sup>η</sup> έκφραση (στα δεξιά).
  - Π.χ.

```
boolean b = (x != 0) && ((y / x) > 2);
```

αριστερή  
έκφραση

δεξιά  
έκφραση

η Java δεν  
αξιολογεί το  
 $((y / x) > 2)$

- Αντίστοιχα, στον `||` τελεστή, αν η 1<sup>η</sup> έκφραση (στα αριστερά) είναι αληθής, δεν χρειάζεται να αξιολογηθεί η 2<sup>η</sup> έκφραση (στα δεξιά).

```
boolean b = (x <= 10) || (x > 20);
```

αριστερή  
έκφραση

δεξιά  
έκφραση

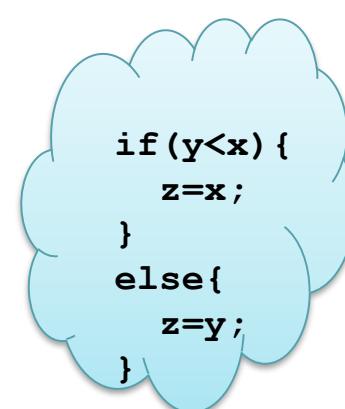
η Java δεν  
αξιολογεί το  
 $(x > 20)$

# Τριαδικός τελεστής (Ternary Conditional Operator)

- `result=condition ? value1 : value2`
  - Εάν η συνθήκη (condition) είναι αληθής,
    - εκχώρησε `result = value1`
  - Διαφορετικά,
    - `result = value2`
- Παράδειγμα

`int x = 2, y = 5, z = 0;`

`z = (y < x) ? x : y;`



```
if (y < x) {
 z=x;
}
else{
 z=y;
}
```

Μειονεκτήματα:

- Μπορούμε να έχουμε μόνο 2 αποτελέσματα
- Οι `value1` και `value2` πρέπει να είναι του ίδιου τύπου

# Τριαδικός τελεστής (Ternary Conditional Operator)

- Πως θα μπορούσαμε να υλοποιήσουμε το παρακάτω με τριαδικό τελεστή;

```
public static void main(String args[]) {
 int numberOfGoals = 5;
 String s;
 if (numberOfGoals == 1) {
 s = "goal";
 }
 else {
 s = "goals";
 }
 System.out.println("I scored " + numberOfGoals + " " + s);
}
```

# Τριαδικός τελεστής (Ternary Conditional Operator)

- Πως θα μπορούσαμε να υλοποιήσουμε το παρακάτω με τριαδικό τελεστή;

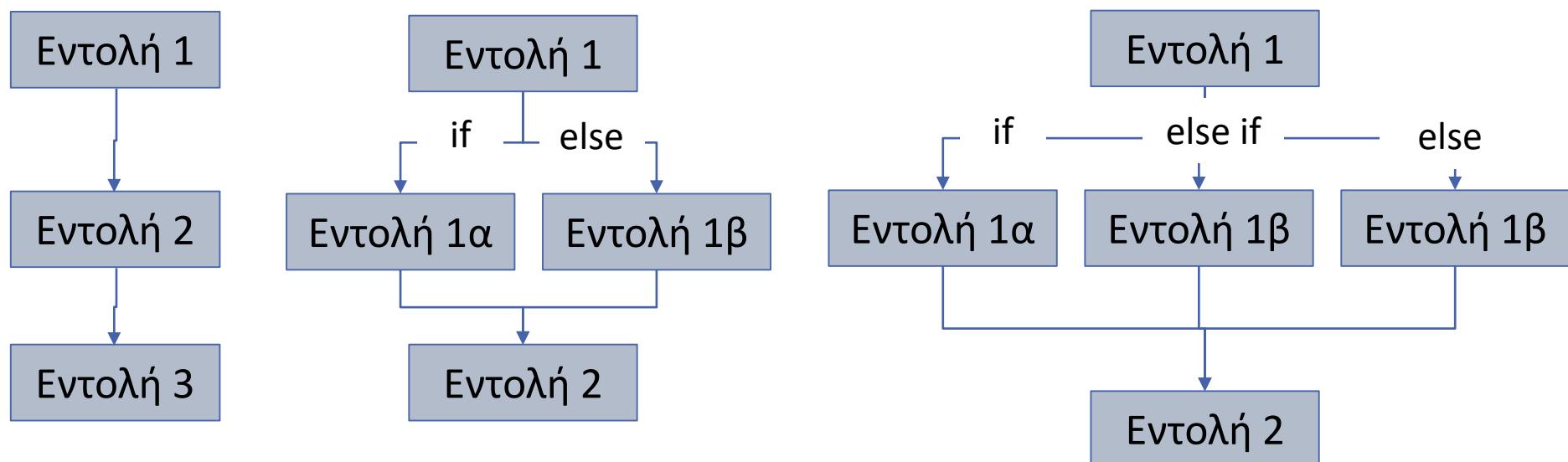
```
public static void main(String args[]) {
 int numberOfGoals = 5;
 String s;
 if (numberOfGoals == 1) {
 s = "goal";
 }
 else {
 s = "goals";
 }
 System.out.println("I scored " + numberOfGoals + " " + s);
}
```

Αλλάζουμε το  
if/else

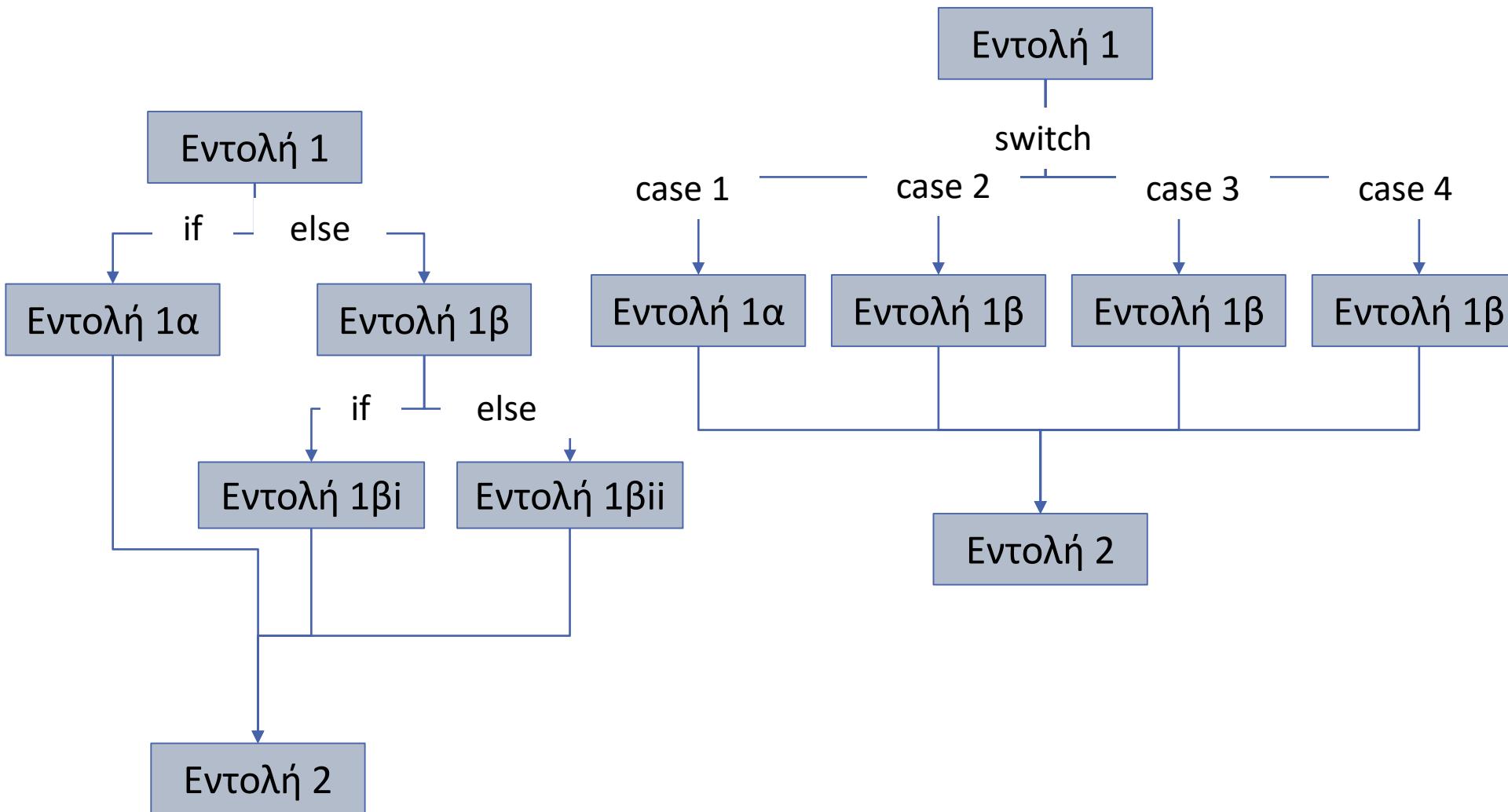
```
System.out.println("I scored "+numberOfGoals + " "
+ (numberOfGoals==1 ? "goal": "goals"));
```

# Άλλες δομές If/else

# Δομές Επιλογής-Διαγράμματα ροής



# Δομές Επιλογής-Διαγράμματα ροής



# Διασυνδεμένη if (Chain if)

- Συνδέει πολλές συνθήκες
- Δυσδιάκριτη στην ανάγνωση κώδικα
- Σύνταξη

```
if(<condition1>){
 //code_block1
}
elseif(<condition2>){
 // code_block2
else{
 // default_code
}
```

# Διασυνδεμένη if (Chain if)

## Παράδειγμα

```
public static void main(String args[]) {
 double income = 30000, tax;
 if (income <= 15000) {
 tax = 0;
 }
 else if (income <= 25000) {
 tax = 0.05 * (income -15000);
 }
 else{
 tax = 0.05 * (income -(25000 -15000));
 tax += 0.10 * (income -25000);
 }
}
```

# Εμφωλιασμένη if (Nested if)

- Μια εντολή if μπορεί να υπάρχει μέσα στο σώμα μιας άλλης εντολής if.

```
if(tvType == "color") {
 if (size == 14) {
 discPercent = 8;
 }
 else {
 discPercent = 10;
 }
}
```

η εντολή else συνδυάζεται  
με την εσωτερική εντολή  
if (size == 14)

```
if(tvType == "color") {
 if (size == 14) {
 discPercent = 8;
 }
}
else {
 discPercent = 10;
}
```

η εντολή else συνδυάζεται  
με την εξωτερική εντολή  
if (tvType == "color")

# Δομή Επιλογής Switch

# Μελέτη Περίπτωσης

- Έστω ότι εργάζεστε σε ένα Δημοτικό.
- Να γράψετε ένα πρόγραμμα που να εκτυπώνει το όνομα της βαθμίδας του κάθε μαθητή ανάλογα με την ηλικία του, για τις 4 πρώτες τάξεις:
  - 6: Α Δημοτικού
  - 7: Β Δημοτικού
  - 8: Γ Δημοτικού,
  - 9: Δ Δημοτικού,

```
public static void main(String args[]) {
 Scanner in = new Scanner(System.in);
 System.out.println("Ποια είναι η ηλικία σου?");
 int grade = in.nextInt();
 if(grade == 6){
 System.out.println("Α Δημοτικού");
 }
 else if (grade == 7) {
 System.out.println("Β Δημοτικού");
 }
 else if (grade == 8) {
 System.out.println("Γ Δημοτικού");
 }
 else if(grade == 9) {
 System.out.println("Δ Δημοτικού");
 }
 else if (grade == 10) {
 System.out.println("Ε Δημοτικού");
 }
 else if(grade == 11) {
 System.out.println("ΣΤ Δημοτικού");
 }
 else {
 System.out.println("Δεν ανήκεις στο
Δημοτικό");
 }
}
```

**Πρώτη υλοποίηση: if / else εντολή**

```
public static void main(String args[]) {
 Scanner in = new Scanner(System.in);
 System.out.println("Ποια είναι η ηλικία σου?");
 int grade = in.nextInt();
 if(grade == 6){
 System.out.println("Α Δημοτικού");
 }
 else if (grade == 7) {
 System.out.println("Β Δημοτικού");
 }
 else if (grade == 8) {
 System.out.println("Γ Δημοτικού");
 }
 else if(grade == 9) {
 System.out.println("Δ Δημοτικού");
 }
 else if (grade == 10) {
 System.out.println("Ε Δημοτικού");
 }
 else if(grade == 11) {
 System.out.println("ΣΤ Δημοτικού");
 }
 else {
 System.out.println("Δεν ανήκεις στο
Δημοτικό");
 }
}
```

**Πρώτη υλοποίηση: if / else εντολή**

```
public static void main(String args[]) {
 Scanner in = new Scanner(System.in);
 System.out.println("Ποια είναι η ηλικία σου?");
 int grade = in.nextInt();
 switch(grade) {
 case 6:
 System.out.println("Α Δημοτικού");
 break;
 case 7:
 System.out.println("Β Δημοτικού");
 break;
 case 8:
 System.out.println("Γ Δημοτικού");
 break;
 case 9:
 System.out.println("Δ Δημοτικού");
 break;
 case 10:
 System.out.println("Ε Δημοτικού");
 break;
 case 11:
 System.out.println("ΣΤ Δημοτικού");
 break;
 default:
 System.out.println("Δεν ανήκεις στο Δημοτικό");
 break;
 }
}
```

**Δεύτερη υλοποίηση: switch εντολή**

# Η εντολή switch

- Είναι πιο ευανάγνωστη και συντηρήσιμη
- Μπορεί να χρησιμοποιηθεί σε πολύπλοκες διακλαδώσεις
- Προσφέρει καλύτερη απόδοση (σε σύγκριση με την *if / else*)

# Η εντολή switch: Σύνταξη

```
Switch <έκφραση> {
 case : <τιμή 1>
 //εντολές
 break;
 case : <τιμή 1>
 //εντολές
 break;
 default :
 //εντολές
}
...
```



- Η <έκφραση> μπορεί να πάρει μία μοναδική τιμή τύπου int, short, byte, char ή String
- Όχι όμως εύρος τιμών



- Η **break** χρησιμοποιείται ως η τελευταία εντολή σε κάθε case block
- Μεταφέρει τον έλεγχο έξω από την Switch

# Switch Fall Through

- Κατάσταση που συμβαίνει εάν δεν υπάρχουν εντολές break στο τέλος κάθε case εντολής.
- Σε αυτή την περίπτωση, όλες οι εντολές εκτελούνται σειριακά, ανεξάρτητα από την <τιμή 1> της κάθε μίας case, μέχρι να υπάρξει εντολή break.

# Switch Fall Through: Παράδειγμα

```
public static void main(String args[]) {
 int month = 12;
 switch (month) {
 case 2:
 System.out.println("28 days (29 in leap years)");
 break;
 case 4:
 case 6:
 case 9:
 case 11:
 System.out.println("30 days");
 break;
 case 1:
 case 3:
 case 5:
 case 7:
 case 8:
 case 12:
 System.out.println("31 days");
 break;
 default:
 System.out.println("Illegal month number");
 break;
 }
}
```