

2. ΕΙΣΑΓΩΓΗ ΣΤΗ JAVA

Η γλώσσα προγραμματισμού Java

Βασικό συντακτικό, ορισμός μεταβλητών,
έλεγχος ροής

Η εξέλιξη των γλωσσών προγραμματισμού

- Η εξέλιξη των γλωσσών προγραμματισμού είναι μια διαδικασία **αφαίρεσης**
 - Στην αρχή ένα πρόγραμμα ήταν μια σειρά από εντολές σε γλώσσα μηχανής.
 - Με τον **Διαδικασιακό Προγραμματισμό (procedural programming)**, ένα πρόγραμμα έγινε μια συλλογή από **διαδικασίες** που η μία καλεί την άλλη.
 - Στον **Συναρτησιακό Προγραμματισμό (functional programming)** ένα πρόγραμμα είναι μια συλλογή από **συναρτήσεις** όπου η μία εφαρμόζεται πάνω στην άλλη.
 - Στον **Λογικό Προγραμματισμό (logic programming)** ένα πρόγραμμα είναι μια συλλογή από **κανόνες** και **γεγονότα**.
 - Στον **Αντικειμενοστραφή Προγραμματισμό (object oriented programming)** ένα πρόγραμμα είναι μια συλλογή από **κλάσεις** και **αντικείμενα** όπου το ένα μιλάει με το άλλο

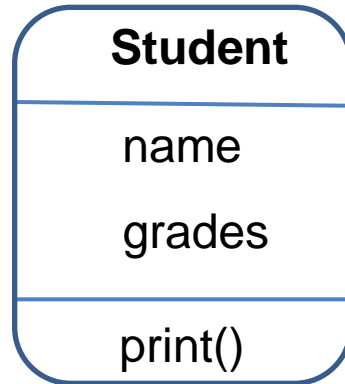
Αντικειμενοστραφής Προγραμματισμός

- Οι πέντε αρχές του Allan Kay:
 - Τα πάντα είναι **αντικείμενα**.
 - Ένα πρόγραμμα είναι μια **συλλογή** από **αντικείμενα** όπου το ένα λέει στο άλλο τι να κάνει.
 - Κάθε αντικείμενο έχει δικιά του **μνήμη** και αποτελείται από **άλλα αντικείμενα**.
 - Κάθε αντικείμενο έχει ένα συγκεκριμένο **τύπο**.
 - Τύπος = **Κλάση**
 - Αντικείμενα του **ίδιου τύπου** μπορούν να δεχτούν **τα ίδια μηνύματα**
 - Δηλαδή έχουν τις **ίδιες λειτουργίες**

Κλάσεις και Αντικείμενα

Κλάση

Μια αφηρημένη περιγραφή μιας οντότητας



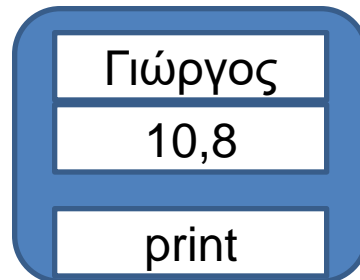
Όνομα κλάσης

Πεδία κλάσης: Ιδιότητες/Χαρακτηριστικά

Μέθοδοι κλάσης: λειτουργίες

Αντικείμενα

Student **studentGeorge**:



Ένα συγκεκριμένο στιγμιότυπο της αφηρημένης κλάσης

Πρόσβαση στο αντικείμενο μόνο μέσω κλήσεων των μεθόδων:

```
studentGeorge.print()
```

Τυπώνει τις πληροφορίες για το αντικείμενο.

Σύντομη ιστορία του Αντικειμενοστραφούς Προγραμματισμού

- Η πρώτη γλώσσα που χρησιμοποίησε τις έννοιες της κλάσης και του αντικειμένου θεωρείται η **SIMULA** (1960s)
 - Γλώσσα για προσομοιώσεις συστημάτων
- Εμπνευσμένος από την SIMULA ο **Allan Kay** δημιούργησε στην HP την γλώσσα **SmallTalk** με στόχο μια γλώσσα που να υποστηρίζει γραφικά (1970s)
 - Ήταν αυτός που εισήγαγε την έννοια «**Αντικειμενοστραφής Προγραμματισμός**» (**Object Oriented Programming**)
 - Το 2003 βραβεύτηκε με το Turing Award
- Οι ιδέες του αντικειμενοστραφούς προγραμματισμού άρχισαν να εισάγονται σε πολλές υπάρχουσες ή νέες γλώσσες. Ο **Bjorn Stroustrup** δημιούργησε την **C++** (1980s)
- Η Sun δημιούργησε την γλώσσα **Java** η οποία βρίσκει εφαρμογή σε ανάπτυξη εφαρμογών στο διαδίκτυο (1990s)
 - Ακολούθησε η Microsoft με την .NET πλατφόρμα και τις γλώσσες **Visual Basic** και **C#**

Ιστορία της Java

- Ο **Patrick Naughton** απειλεί την Sun ότι θα φύγει.
- Τον βάζουν σε μία ομάδα αποτελούμενη από τους **James Gosling** και **Mike Sheridan** για να σχεδιάσουν τον προγραμματισμό των έξυπνων συσκευών της επόμενης γενιάς.
 - The **Green project**.
- Ο Gosling συνειδητοποιεί ότι η C++ δεν είναι αρκετά αξιόπιστη για να δουλεύει σε συσκευές περιορισμένων δυνατοτήτων και με διάφορες αρχιτεκτονικές.
 - Δημιουργεί τη γλώσσα **Oak**
- Το 1992 η ομάδα κάνει ένα demo μιας συσκευής **PDA, *7 (star 7)**
 - Δημιουργείται η θυγατρική εταιρία **FirstPerson Inc**
- Η δημιουργία των έξυπνων συσκευών αποτυγχάνει και η ομάδα (μαζί με τον **Eric Schmidt**) επικεντρώνεται στην εφαρμογή της πλατφόρμας στο **Internet**.
 - Ο Naughton φτιάχνει τον **WebRunner browser** (μετα **HotJava**)
 - Η γλώσσα μετονομάζεται σε **Java** και το ενδιαφέρον επικεντρώνεται σε εφαρμογές που τρέχουν μέσα στον browser.
- Ο **Marc Andersen** ανακοινώνει ότι ο **Netscape browser** θα υποστηρίξει Java μικροεφαρμογές (applets)

Java

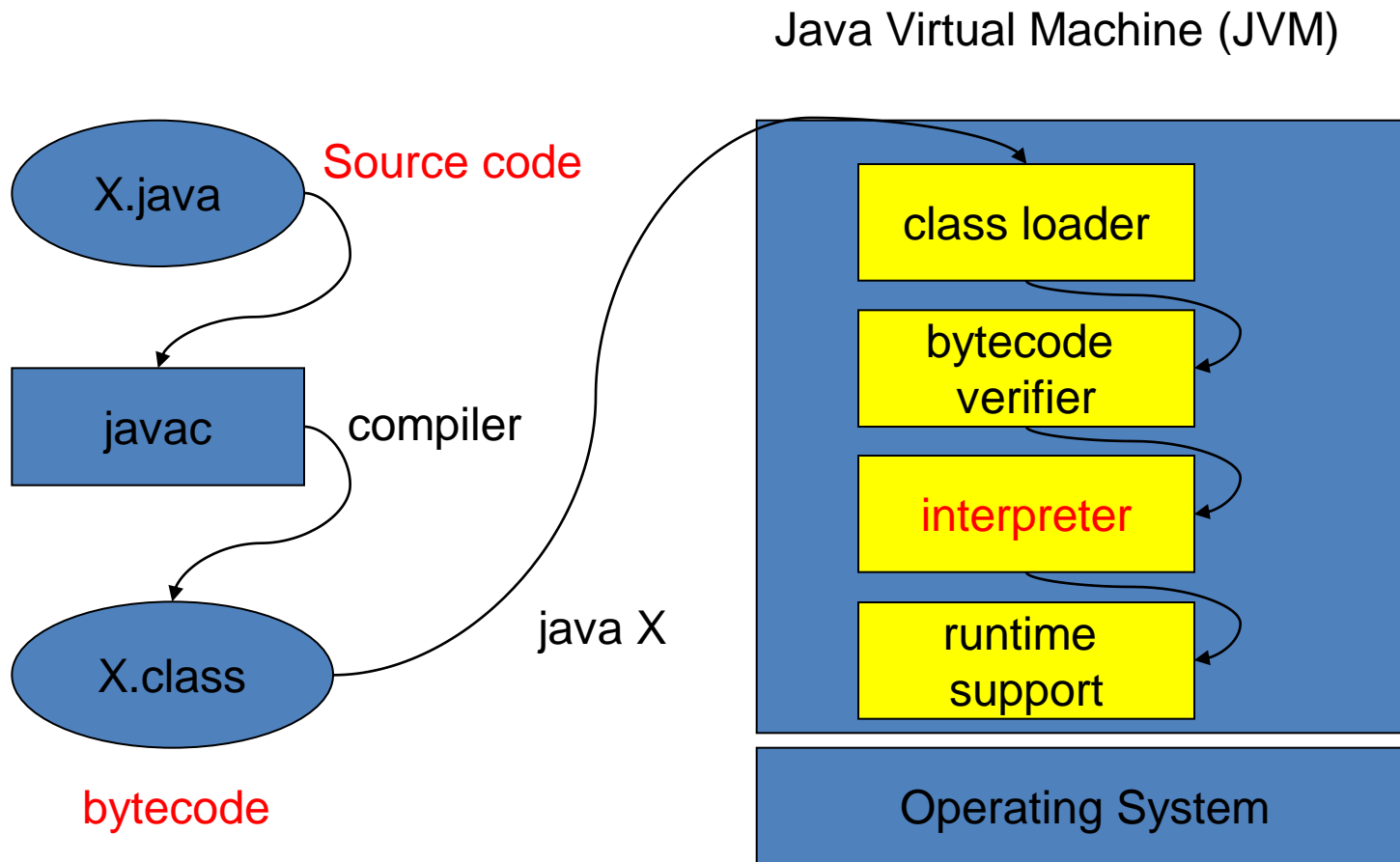
- Η Java είχε τους εξής στόχους:
 - "simple, object-oriented and familiar"
 - "robust and secure"
 - "architecture-neutral and portable"
 - "high performance"
 - "interpreted, threaded, and dynamic"

Java

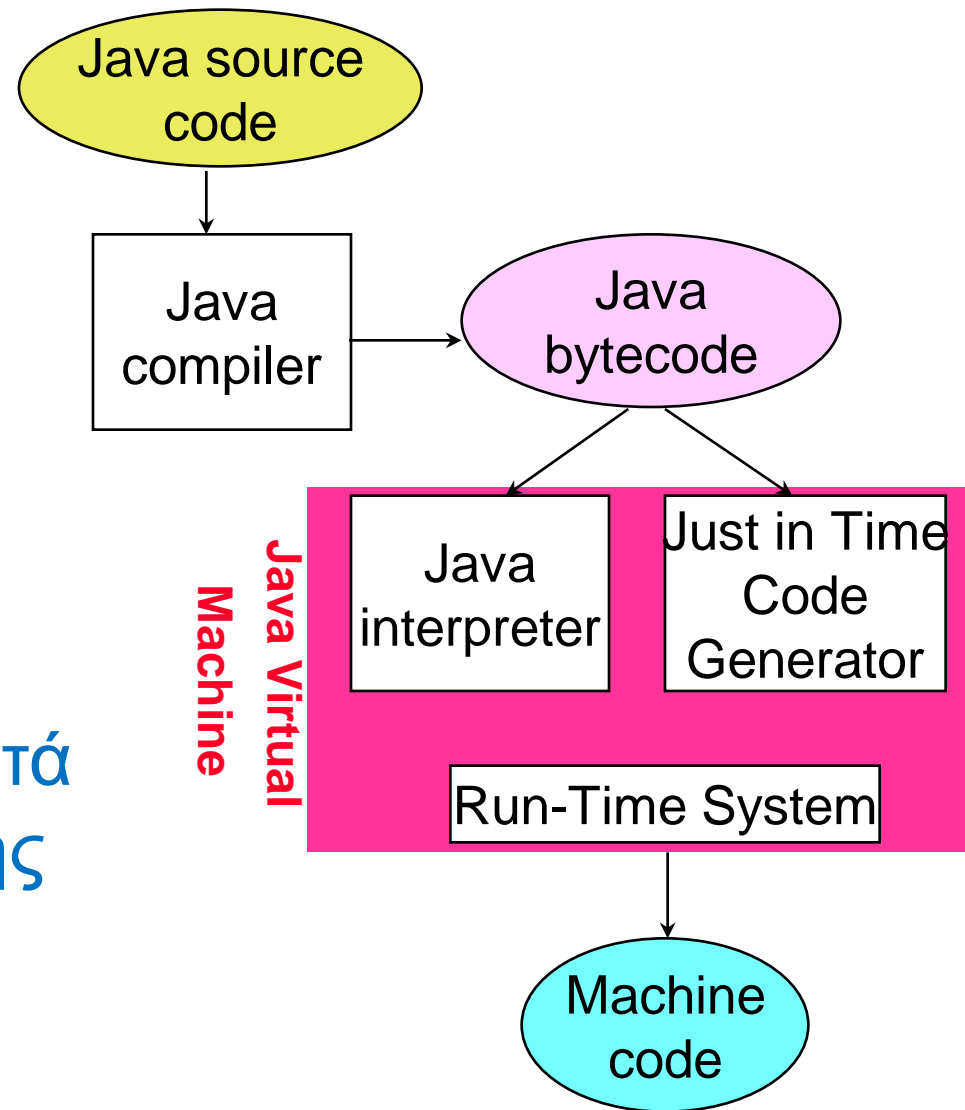
- Η Java είχε τους εξής στόχους:
 - "simple, object-oriented and familiar"
 - "robust and secure"
 - "architecture-neutral and portable"
 - "high performance"
 - "interpreted, threaded, and dynamic"

“architecture-neutral and portable”

- Το μεγαλύτερο πλεονέκτημα της Java είναι η **μεταφερισιμότητα (portability)**: ο κώδικας μπορεί να τρέξει πάνω σε οποιαδήποτε πλατφόρμα.
 - **Write-Once-Run-Anywhere** μοντέλο, σε αντίθεση με το σύνηθες **Write-Once-Compile-Anywhere** μοντέλο.
- Αυτό επιτυγχάνεται δημιουργώντας ένα **ενδιάμεσο κώδικα (bytecode)** ο οποίος μετά τρέχει πάνω σε μια **εικονική μηχανή (Java Virtual Machine)** η οποία το μεταφράζει σε **γλώσσα μηχανής**.
 - Οι προγραμματιστές πλέον γράφουν κώδικα για την εικονική μηχανή, η οποία δημιουργείται **για οποιαδήποτε πλατφόρμα**.



- **Just in Time (JIT) code generator (compiler)** βελτιώνει την απόδοση των Java Applications μεταφράζοντας (compiling) bytecode σε machine code **πριν ή κατά τη διάρκεια της εκτέλεσης**

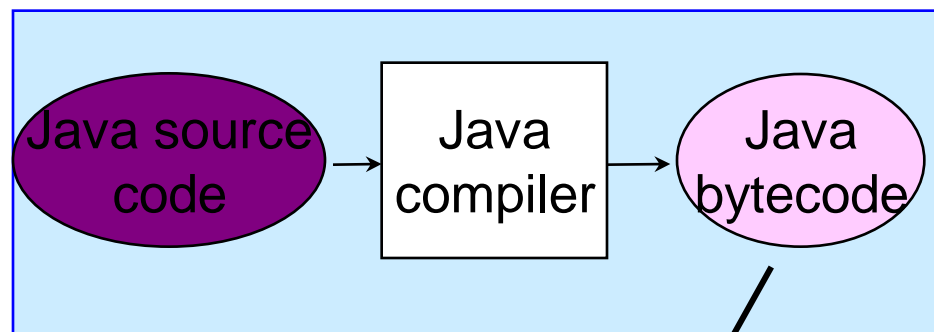


Java και το Internet

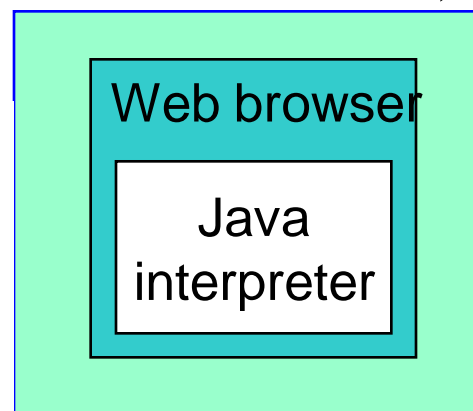
- Η προσέγγιση της Java είχε μεγάλη επιτυχία για **Web εφαρμογές**, όπου έχουμε ένα τεράστιο κατακευματισμένο **client-server** μοντέλο με πολλές διαφορετικές αρχιτεκτονικές
 - **Client-side programming**: Αντί να κάνει όλη τη δουλειά ο server για την δημιουργία της σελίδας κάποια από την επεξεργασία των δεδομένων γίνεται στη μηχανή του client.
 - **Web Applets**: κώδικας ο οποίος κατεβαίνει μαζί με τη Web σελίδα και τρέχει στη μηχανή του client. Είναι πολύ σημαντικό στην περίπτωση αυτή ο κώδικας να είναι portable.
 - **Server-side programming**: μία web σελίδα μπορεί να είναι το αποτέλεσμα ενός προγράμματος που συνδυάζει δυναμικά δεδομένα και είσοδο του χρήστη.
 - **Java Service Pages (JSPs)**: Η λύση της Java. Γίνεται compiled σε **servlets** και τρέχει στη μεριά του server.

Java Applets

- Το Web Browser software περιλαμβάνει ένα **JVM**
 - ◆ **Φορτώνει** τον java byte code από τον remote υπολογιστή
 - ◆ **Τρέχει** τοπικά το Java πρόγραμμα μέσα στο παράθυρο του Browser



Remote computer



Local computer

"simple, object-oriented and familiar"

- **Familiar:** Η Java είχε ως έμπνευση της την C++, και δανείζεται αρκετά από τα χαρακτηριστικά της.
- **Object-oriented:** Η Java είναι «**ΠΙΟ αντικειμενοστραφής**» από την C++ η οποία προσπαθεί να μείνει συμβατή με την C
 - Στην Java **τα πάντα** είναι **αντικείμενα**
- **Simple:** Η Java δίνει λιγότερο έλεγχο στο χρήστη, αλλά κάνει τη ζωή του πιο εύκολη. Η **διαχείριση της μνήμης** γίνεται **αυτόματα**.
 - Η γλώσσα φροντίζει να κάνει πιο εύκολο και πιο σταθερό (robust) τον προγραμματισμό παρότι αυτό μπορεί να έχει αποτέλεσμα τα προγράμματα να γίνονται **πιο αργά**.

HELLO WORLD

Το πρώτο μας πρόγραμμα σε Java

Java Installation

- Για να μπορείτε να μεταγλωττίσετε και να τρέξετε Java προγράμματα στον υπολογιστή σας θα πρέπει να **εγκαταστήσετε** την Java.
 - Θα κάνετε download and install από τη σελίδα της Oracle.
 - Ψάξετε “**download java**” ή “**install java**” για οδηγίες
 - <https://www.java.com/en/download/>
 - https://java.com/en/download/help/windows_manual_download.xml
- Υπάρχει περίπτωση μετά την εγκατάσταση να πρέπει να προσθέσετε το **path στο directory** στο οποίο εγκαταστάθηκε η Java στο **Path environmental variable**
 - Συνήθως αυτό γίνεται αυτόματα.

Δομή ενός απλού Java προγράμματος

- Το **όνομα** του αρχείου που κρατάει το πρόγραμμα είναι **X.java** (όπου **X** το όνομα του προγράμματος)
 - Στο παράδειγμα μας ονομάζουμε το πρόγραμμα μας: **HelloWorld.java**
- Μέσα στο πρόγραμμα μας πρέπει να έχουμε μια **κλάση** με το όνομα **X**.
 - **class X** (**class HelloWorld** στο παράδειγμα μας)
- Η κλάση **X** θα πρέπει να περιέχει μια **μέθοδο main** η οποία είναι το **σημείο εκκίνησης** του προγράμματος μας
 - **public static void main(String[] args)**

File HelloWorld.java

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

Το **όνομα του .java αρχείου** και το **όνομα της κλάσης** (που περιέχει την μέθοδο main) θα πρέπει να είναι τα **ίδια** (αν η κλάση έχει οριστεί public)

Μεταγλώττιση – Compiling

Η μεταγλώττιση γίνεται με την εντολή **javac**

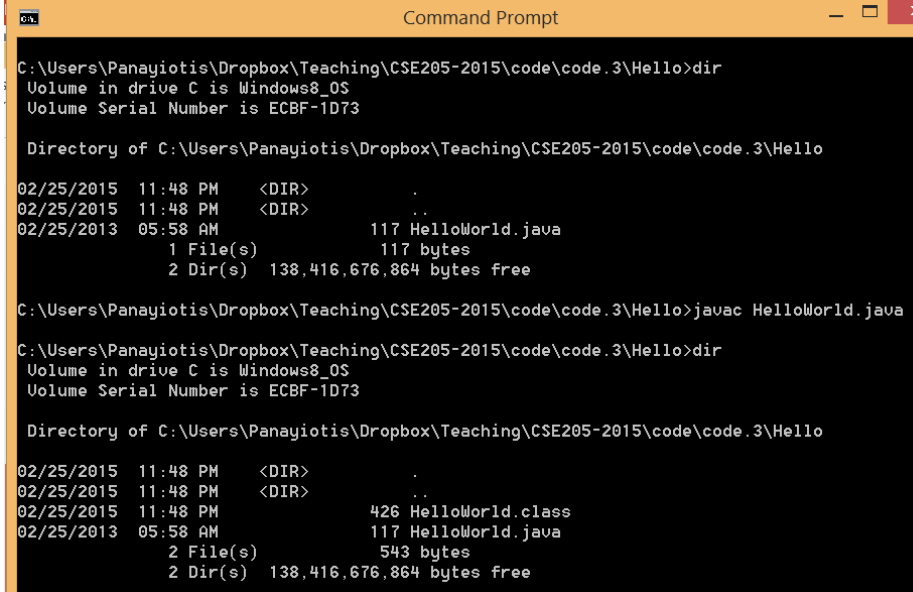
- `javac <.java αρχείο>`

Π.χ.

```
➤ javac HelloWorld.java
```

Το αποτέλεσμα είναι η δημιουργία ενός **.class** αρχείου που περιέχει τον ενδιαμέσο κώδικα (bytecode)

Το αρχείο **HelloWorld.class** στο παράδειγμα μας



```
Command Prompt
C:\Users\Panayiotis\Dropbox\Teaching\CSE205-2015\code\code.3\Hello>dir
Volume in drive C is Windows8_OS
Volume Serial Number is ECBF-1D73

Directory of C:\Users\Panayiotis\Dropbox\Teaching\CSE205-2015\code\code.3\Hello

02/25/2015  11:48 PM  <DIR>          .
02/25/2015  11:48 PM  <DIR>          ..
02/25/2013  05:58 AM             117 HelloWorld.java
                1 File(s)      117 bytes
                2 Dir(s)  138,416,676,864 bytes free

C:\Users\Panayiotis\Dropbox\Teaching\CSE205-2015\code\code.3\Hello>javac HelloWorld.java

C:\Users\Panayiotis\Dropbox\Teaching\CSE205-2015\code\code.3\Hello>dir
Volume in drive C is Windows8_OS
Volume Serial Number is ECBF-1D73

Directory of C:\Users\Panayiotis\Dropbox\Teaching\CSE205-2015\code\code.3\Hello

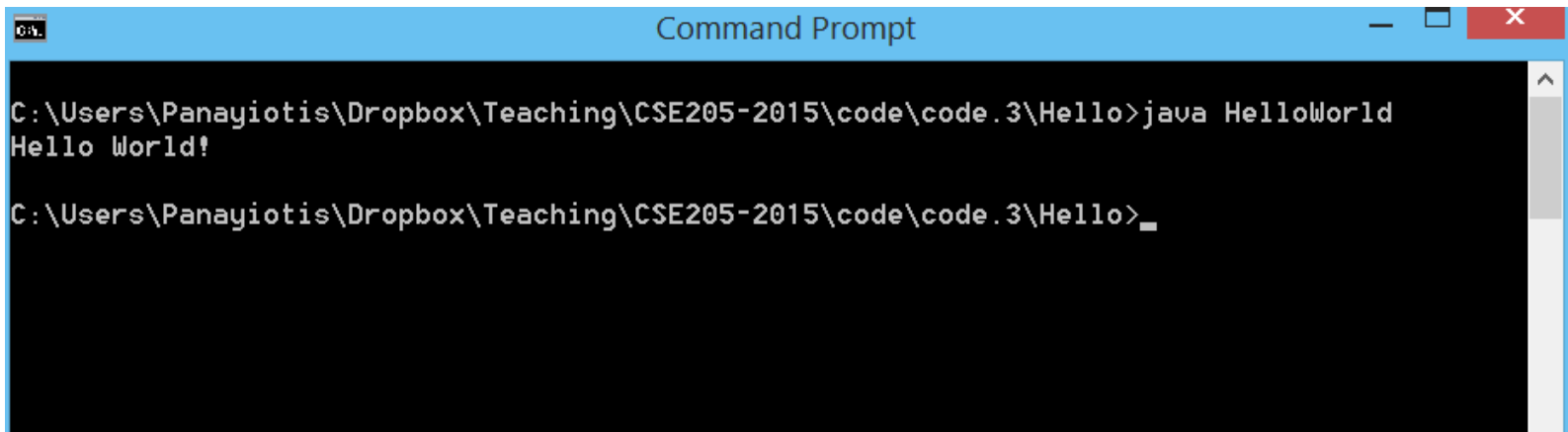
02/25/2015  11:48 PM  <DIR>          .
02/25/2015  11:48 PM  <DIR>          ..
02/25/2015  11:48 PM             426 HelloWorld.class
02/25/2013  05:58 AM             117 HelloWorld.java
                2 File(s)      543 bytes
                2 Dir(s)  138,416,676,864 bytes free
```

Εκτέλεση - Running

- Η εκτέλεση του κώδικα γίνεται με την εντολή **java**
 - `java <όνομα αρχείου χωρίς επίθεμα>`

➤ **java HelloWorld**

Χωρίς κανένα επίθεμα!



```
Command Prompt
C:\Users\Panayiotis\Dropbox\Teaching\CSE205-2015\code\code.3\Hello>java HelloWorld
Hello World!
C:\Users\Panayiotis\Dropbox\Teaching\CSE205-2015\code\code.3\Hello>_
```

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

Λέξεις σε κόκκινο: δεσμευμένες λέξεις

Ορίζει την
κλάση

Όνομα της κλάσης

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

```
class HelloWorld
```

```
{
```

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

```
    {
```

```
        // print message
```

```
        System.out.println("Hello world!");
```

```
    }
```

```
}
```

Τα άγκιστρα { ... } ορίζουν ένα **λογικό block** του κώδικα

- Αυτό μπορεί να είναι **μία κλάση**, **μία συνάρτηση**, **ένα if statement**
- Οι μεταβλητές που ορίζουμε μέσα σε ένα λογικό block είναι **τοπικές**, έχουν **εμβέλεια** μόνο μέσα στο block
- Αντίστοιχο των tabs στην Python, εδώ δεν χρειάζονται αλλά είναι καλό να τα βάζουμε για να διαβάζεται ο κώδικας πιο εύκολα.

Ορισμός της συνάρτησης main

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


Ορισμός της συνάρτησης main

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

public, static: θα τα εξηγήσουμε αργότερα

Ορισμός της συνάρτησης main

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

Το τι επιστρέφει η μέθοδος

void: Η μέθοδος δεν επιστρέφει τίποτα.

Ορισμός της συνάρτησης main

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

Το όνομα της μεθόδου

- **main**: ειδική περίπτωση που σηματοδοτεί το σημείο εκκίνησης του προγράμματος.

Ορισμός της συνάρτησης main

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

Ορίσματα της μεθόδου

- Ένας **πίνακας** από **Strings** που αντιστοιχούν στις **παραμέτρους** με τις οποίες τρέχουμε το πρόγραμμα.

Η κλάση String

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

- **String**: κλάση που χειρίζεται τα **αλφαριθμητικά**.
- Στη Java χρειάζεται να ορίσουμε τον **τύπο** της κάθε μεταβλητής
- **Strongly typed language**

Σχόλια!

```
/**  
 * A class that prints a message "hello world"  
 **/
```

```
class HelloWorld
```

```
{
```

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

```
    {
```

```
        // print message
```

```
        System.out.println("Hello world!");
```

```
    }
```

```
}
```

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

Κάθε εντολή στη Java πρέπει να τερματίζει με το ;

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

Αντικείμενο **System.out**
Ορίζει το ρεύμα εξόδου

Μέθοδος println:
Τυπώνει το String αντικείμενο που
δίνεται ως όρισμα και αλλάζει γραμμή


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

To "Hello World" είναι ένα αντικείμενο της κλάσης String

Programming Style

Το όνομα της κλάσης ξεκινάει με κεφαλαίο και χρησιμοποιούμε την **CamelCase** σύμβαση

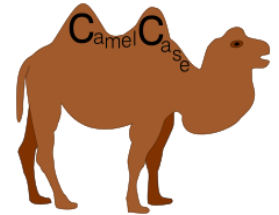
```
class HelloWorld
{
    public static void main(String[] args)
    {
        // print message
        System.out.println("Hello world!");
    }
}
```

Στοίχιση του κώδικα:

- Οι εντολές μέσα σε ένα block του κώδικα ξεκινάνε ένα tab πιο μπροστά από το προηγούμενο.
- Όλες οι εντολές σε ένα block είναι στοιχισμένες
- Τα άγκιστρα είναι στοιχισμένα με την εντολή που ορίζει το block

Programming Style: Ονόματα

- Τα ονόματα των **κλάσεων** ξεκινάνε με κεφαλαίο, τα ονόματα των **πεδίων**, **μεθόδων** και **αντικειμένων** με μικρό.
 - Π.χ., **HelloWorld**, **position**, **print**
- Κάποιες συμβάσεις ξεκινούν τα ονόματα πεδίων με ‘**_**’ για να ξεχωρίζουν από τις μεθόδους
 - Π.χ., **_position**
- Χρησιμοποιούμε **ολόκληρες λέξεις** (και συνδυασμούς τους) για τα ονόματα
 - Δεν πειράζει αν βγαίνουν μεγάλα ονόματα
- Χρησιμοποιούμε το **CamelCase Style**
 - Όταν για ένα όνομα έχουμε πάνω από μία λέξη, τις συνενώνουμε και στο σημείο συνένωσης κάνουμε το πρώτο γράμμα της λέξης κεφαλαίο
 - **printName** όχι **print_name**
- Χρησιμοποιούμε **κεφαλαία** και ‘**_**’ για τις **σταθερές**.
 - Π.χ., **PI_NUMBER**



Παράδειγμα 2

- Φτιάξτε ένα πρόγραμμα που τυπώνει το λόγο δύο ακεραίων.

Division.java

Ισοδύναμο με το `String[] args`

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

Ορισμός/Δήλωση μεταβλητών

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

Δήλωση δύο ακεραίων μεταβλητών και μιας πραγματικής μεταβλητής

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

Ορισμός/Δήλωση μεταβλητών

- Ορισμός μεταβλητής

```
<τυπος> <όνομα μεταβλητής> [ = τιμή ] ;
```

- Ο ορισμός της μεταβλητής γίνεται **μόνο μία φορά**, πριν ή όταν θα την χρησιμοποιήσουμε για πρώτη φορά.
- Ο τύπος της μεταβλητής είναι είτε ένας πρωταρχικός τύπος, είτε μια υπάρχουσα ή νέα κλάση

- Παραδείγματα

```
int enumerator = 32;
```

Ορισμός μεταβλητής με αρχικοποίηση

```
int denominator = 10;
```

```
String myString;
```

Ορισμός μεταβλητής χωρίς αρχικοποίηση. Η Java θα δώσει default τιμές ("", false)

```
boolean b;
```

```
double division = enumerator / (double) denominator;
```

Ορισμός μεταβλητής με αρχικοποίηση με ανάθεση

Πρωταρχικοί τύποι

Όνομα τύπου	Τιμή	Μνήμη
boolean	true/false	1 byte
char	Χαρακτήρας (Unicode)	2 bytes
byte	Ακέραιος	1 byte
short	Ακέραιος	2 bytes
int	Ακέραιος	4 bytes
long	Ακέραιος	8 bytes
float	Πραγματικός	4 bytes
double	Πραγματικός	8 bytes

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

Πρωταρχικοί τύποι

Όνομα τύπου	Τιμή	Μνήμη
boolean	true/false	1 byte
char	Χαρακτήρας (Unicode)	2 bytes
byte	Ακέραιος	1 byte
short	Ακέραιος	2 bytes
int	Ακέραιος	4 bytes
long	Ακέραιος	8 bytes
float	Πραγματικός	4 bytes
double	Πραγματικός	8 bytes

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

Ανάθεση

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

μεταβλητή

Έκφραση που αποτιμάται σε μια τιμή

Ο τύπος της έκφρασης και της μεταβλητής θα πρέπει να συμφωνούν

Ανάθεση: αποτίμηση της τιμής της έκφρασης στο δεξιό μέλος του “=” και μετά ανάθεση της τιμής στην μεταβλητή στο αριστερό μέλος
Το αριστερό μέλος είναι **πάντα** μεταβλητή

Casting

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

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

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

Αναθέσεις

- Στην ανάθεση κατά κανόνα, η τιμή του δεξιού μέρους θα πρέπει να είναι **ίδιου τύπου** με την μεταβλητή του αριστερού μέρους.
- Υπάρχουν εξαιρέσεις όταν υπάρχει **συμβατότητα** μεταξύ τύπων
- **byte** → **short** → **int** → **long** → **float** → **double**
 - Μια τιμή τύπου **T** μπορούμε να την αναθέσουμε σε μια μεταβλητή τύπου που εμφανίζεται **δεξιά του T**.
- (Σε αντίθεση με την C) ο τύπος `boolean` δεν είναι συμβατός με τους ακέραιους.

Διαχείριση αλφαριθμητικών

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

Ο τελεστής “+” μεταξύ αντικείμενων της κλάσης String **συνενώνει** (concatenates) τα δύο String.

Μεταξύ ενός String και ενός βασικού τύπου, ο βασικός τύπος **μετατρέπεται** σε String και γίνεται η συνένωση