ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΠΛΗΡΟΦΟΡΙΚΗ ΙΙ

Ενότητα 1: Εισαγωγικές έννοιες

Μιχάλης Δρακόπουλος

Σχολή Θετικών επιστημών

Τμήμα Μαθηματικών

ΠΛΗΡΟΦΟΡΙΚΗ ΙΙ (Java)
Ενότητα 1

Αλγόριθμος: Βήμα προς βήμα διαδικασία για την επίλυση κάποιου προβλήματος. Το πλήθος των βημάτων πρέπει να είναι πεπερασμένο.

Αλλιώς: Πεπερασμένη ακολουθία ενεργειών που περιγράφει την επίλυση κάποιου προβλήματος.

Πρόγραμμα: Ακριβής διατύπωση ενός αλγορίθμου σε μια γλώσσα προγραμματισμού.

Βήματα στην υπολογιστική επίλυση ενός προβλήματος:

1) Ανάλυση δεδομένων του προβλήματος
2) Μαθηματική διατύπωση του προβλήματος
3) Σχεδιασμός κατάλληλου αλγορίθμου
4) Ανάπτυξη προγράμματος (αλγόριθμος → σε γλώσσα προγραμματισμού)
5) Εκτέλεση προγράμματος για συγκεκριμένα δεδομένα
6) Ερμηνεία αποτελεσμάτων

Γλώσσες προγραμματισμού

i) Γλώσσες υψηλού επιπέδου (Java, C, C++, Fortran, Pascal, Basic, κτλ.)
ii) Γλώσσες χαμηλού επιπέδου (γλώσσα μηχανής (Γ.Μ.), assembly)

Οι Η/Υ εκτελούν γλώσσα μηχανής (Γ.Μ.).

Γ.Μ.:

- διαφορετική για κάθε τύπο επεξεργαστή
- εντολές = αλληλουχίες από bits (0 και 1)

Assembly (λίγο ανώτερο επίπεδο από Γ.Μ.):

- διαφορετική για κάθε τύπο επεξεργαστή
- μνημονικά ονόματα αντί για bits

➢ Οι γλώσσες υψηλού επιπέδου (Γ.Υ.Ε.) μετατρέπονται σε γλώσσα μηχανής με κατάλληλα προγράμματα: τους **μεταγλωττιστές (compilers)**.

Γ.Υ.Ε. —*compiler*→ **Γ.Μ.**

➢ Οι γλώσσες υψηλού επιπέδου είναι ανεξάρτητες από τον τύπο επεξεργαστή, άρα ο ίδιος κώδικας ενός προγράμματος σε κάποια γλώσσα είναι ίδιος για κάθε Η/Υ.

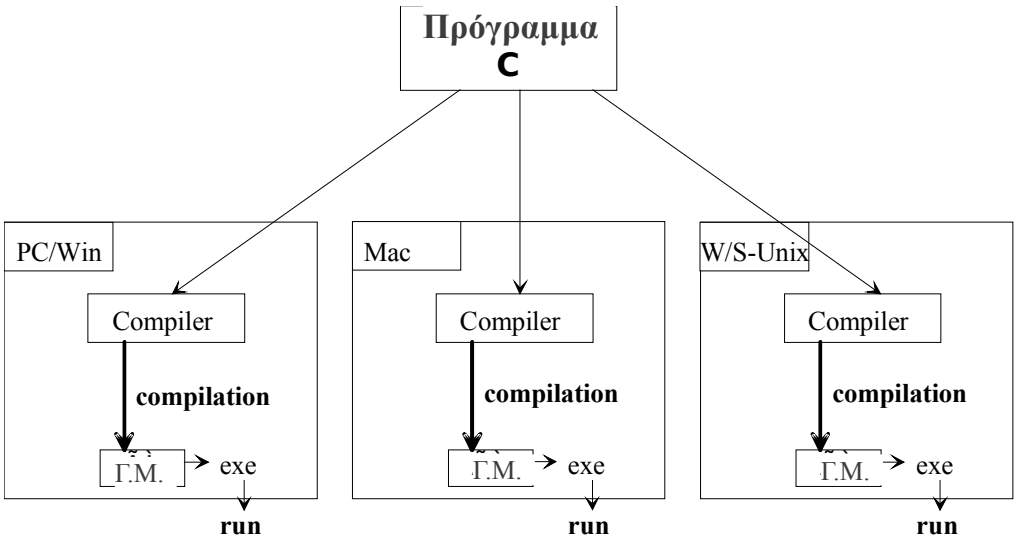
Όμως: ο μεταγλωττιστής κάποιας τέτοιας γλώσσας είναι διαφορετικός για κάθε τύπο επεξεργαστή. Άρα, ένα πρόγραμμα σε μια γλώσσα υψηλού επιπέδου x, μπορεί να είναι ίδιο για κάθε τύπο επεξεργαστή, όμως για να εκτελεσθεί χρειάζεται να μετατραπεί σε Γ.Μ. από τον αντίστοιχο μεταγλωττιστή της γλώσσας αυτής για τον συγκεκριμένο τύπο επεξεργαστή.

Σημείωση: Η μεταγλώττιση (compilation) ενός προγράμματος είναι συνήθως χρονοβόρα διαδικασία.

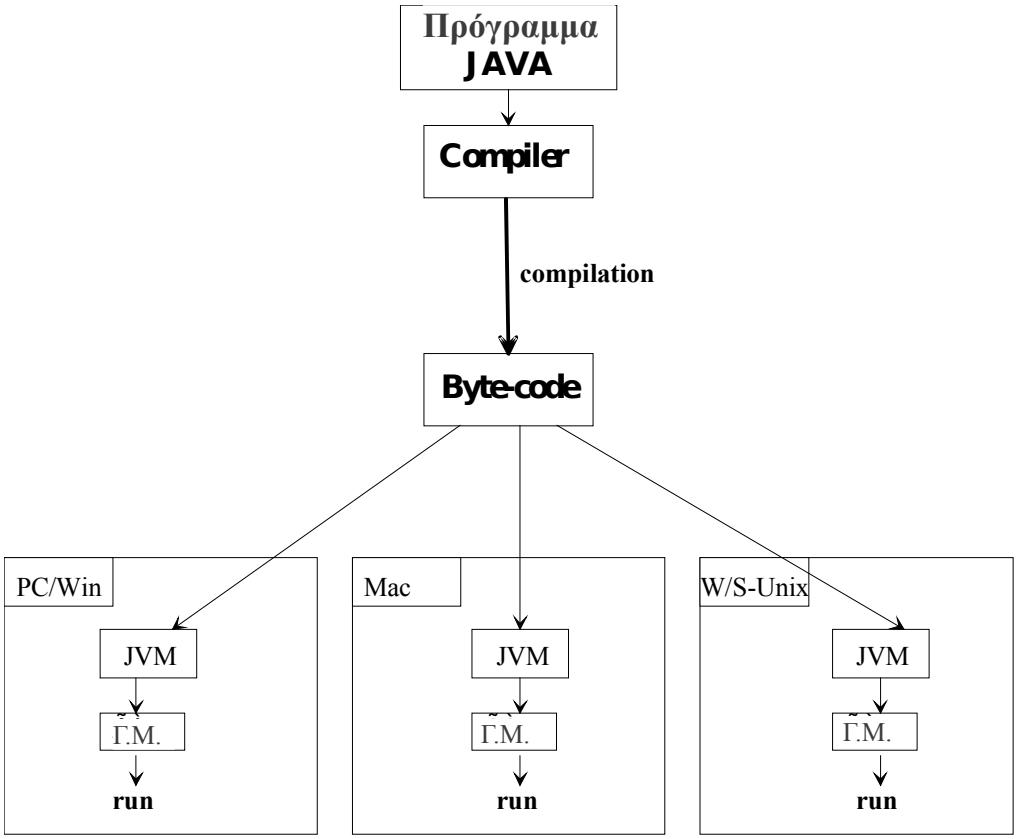
Το πλεονέκτημα της Java: Έχει **έναν** μεταγλωττιστή (compiler) για όλους τους τύπους επεξεργαστών. Ο compiler αυτός όμως **δεν** παράγει Γ.Μ., παράγει **byte-code**, μια «Γ.Μ.» για έναν υποθετικό (ιδεατό) υπολογιστή. Αυτός ο υποθετικός υπολογιστής (java virtual machine – JVM) είναι στην ουσία ένας διερμηνέας (interpreter) που μεταφράζει εντολή προς εντολή το byte-code σε Γ.Μ. του συγκεκριμένου επεξεργαστή στον οποίο τρέχει. Το JVM είναι διαφορετικό για κάθε επεξεργαστή, όμως είναι πολύ πιο απλό από έναν μεταγλωττιστή.

- Άρα, μπορεί η Java να προσθέτει ένα επιπλέον βήμα στην όλη διαδικασία, όμως πετυχαίνει τη μέγιστη **φορητότητα (portability)**, αφού χρειάζεται μόνο μία μεταγλώττιση (compilation) για να μπορεί να τρέξει σε οποιονδήποτε επεξεργαστή, ενώ η διερμηνεία είναι μια πολύ απλή διαδικασία.

Π.χ., για να τρέξει ένα πρόγραμμα C σε τρεις υπολογιστές που βασίζονται σε διαφορετικές πλατφόρμες (διαφορετικό επεξεργαστή ή/και λειτουργικό σύστημα), ακολουθείται η εξής διαδικασία:



Αντίστοιχα, για ένα πρόγραμμα Java η διαδικασία είναι η ακόλουθη:



Φαίνεται λοιπόν ότι, παρόλο που η Java προσθέτει ένα επιπλέον βήμα στην όλη διαδικασία, καταφέρνει να πραγματοποιεί *ένα μόνο compilation* του προγράμματος (που είναι η «επίπονη» διαδικασία, γι αυτό και φαίνεται στα σχήματα με μεγάλο βέλος), έναντι ενός compilation *για κάθε υπολογιστή* που απαιτεί η C. Αυτός είναι ο βασικός λόγος που η Java χρησιμοποιήθηκε και επικράτησε στο internet. Εάν κάποιος προσπαθούσε να υλοποιήσει τη διαδικασία της Java με κάποια άλλη γλώσσα προγραμματισμού (άρα έκανε το compilation αρχικά), θα έπρεπε **να ξέρει εκ των προτέρων** για τι είδους πλατφόρμα προορίζεται κάθε φορά το πρόγραμμα, κάτι που είναι εξαιρετικά ασύμφορο και πολλές φορές αδύνατο να πραγματοποιηθεί στο Internet.

Βασικά χαρακτηριστικά της Java:

1) Φορητότητα (portability): εκτέλεση μεταγλωττισμένου (compiled) κώδικα ανεξαρτήτως πλατφόρμας (→ Internet)
2) Αντικειμενοστραφής αλλά απλούστερη της C++
3) Μεγάλη βιβλιοθήκη έτοιμων κλάσεων
4) Χρησιμοποιεί στοιχεία της C
5) Ασφαλής
6) Κατανεμημένη (συντονισμός εκτέλεσης κώδικα διαφορετικών Η/Υ)

Ανάπτυξη εφαρμογής σε Java:

1) Κώδικας Java → κλάσεις → αρχεία. Π.χ., `Class1.java`

2) Compiler (μεταγλωττιστής)

`javac Class1.java` → `Class1.class` (byte-code)

3) Interpreter (διερμηνέας) (JVM)

`java Class1` → εκτέλεση του αρχείου Class1.class (αφού γίνει αυτόματα η διερμηνεία του από byte-code σε Γ.Μ.)

Μια εφαρμογή Java μπορεί να είναι είτε πρόγραμμα (οπότε τρέχει καλώντας τον JVM) είτε **applet**, δηλαδή ειδικό πρόγραμμα για το internet. Στην περίπτωση που μια εφαρμογή είναι applet, τρέχει είτε μέσω ενός web browser (π.χ. Firefox, Google Chrome, Opera, IE, κτλ.) είτε μέσω του appletviewer.

<u>Παράδειγμα μιας απλής εφαρμογής:</u> (Μία μόνο κλάση)

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

→ save: HelloWorld.java *(αποθήκευση του κώδικα σε αρχείο)*

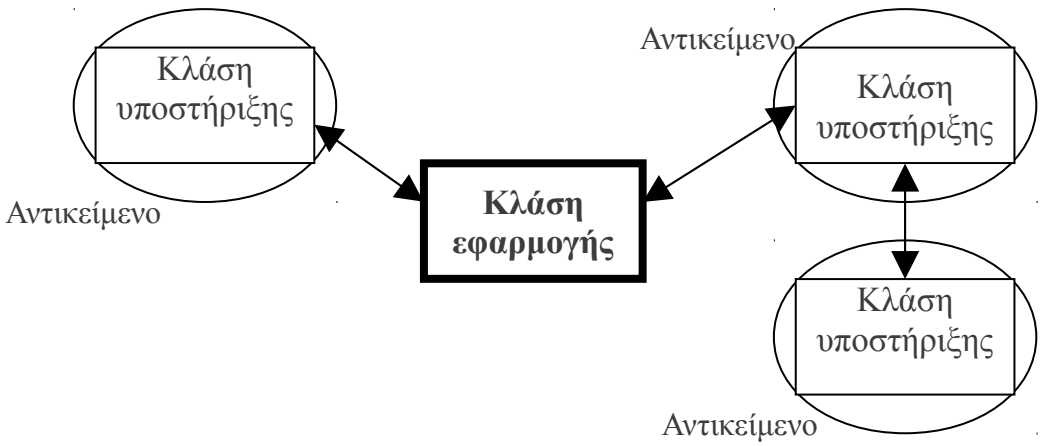
→ javac HelloWorld.java

→ java HelloWorld

Σε αυτό το σημείο το πρόγραμμα θα εκτελεσθεί και θα εμφανίσει στην οθόνη το μήνυμα:

Hello world!

Γενική μορφή ενός προγράμματος Java:



Ένα πρόγραμμα Java έχει απαραίτητα μία (και μόνο μία) κλάση εφαρμογής (η οποία περιέχει τη μέθοδο main) και *εάν* περιέχει και άλλες κλάσεις, τότε αυτές λέγονται κλάσεις υποστήριξης.

(Περισσότερα για τη γενική δομή ενός προγράμματος θα αναφερθούν σε επόμενες ενότητες, με την εισαγωγή στον αντικειμενοστραφή τρόπο προγραμματισμού).

Σφάλματα προγραμματισμού

α) Συντακτικό σφάλμα (syntax error) → κατά το compilation

β) Σφάλμα κατά την εκτέλεση (run-time error) → κατά τη διερμηνεία από το JVM (δηλ., περνάει το compilation αλλά η εκτέλεση του προγράμματος σταματάει.

γ) Λογικό σφάλμα (logic error / bug) → το πρόγραμμα εκτελείται αλλά το αποτέλεσμα είναι λάθος

- Δυσκολία εντοπισμού σφαλμάτων (συνήθως): **γ > β > α**
- Διαδικασία εντοπισμού και επίλυσης σφαλμάτων: **debugging**
- Διαδικασία ελέγχου κάποιων συγκεκριμένων σφαλμάτων (εξαιρέσεων): χειρισμός εξαιρέσεων (exceptions handling)

Μεταβλητές (variables)

Χρησιμοποιούνται για την αποθήκευση (φύλαξη) δεδομένων. Το στοιχείο το οποίο περιέχει μια μεταβλητή λέγεται **τιμή**.

Μια μεταβλητή έχει:
- ορατότητα (θα το δούμε αργότερα...)
- τύπο (type)
- όνομα (identifier – αναγνωριστικό)
- τιμή (value)

Η δημιουργία μιας μεταβλητής σε ένα πρόγραμμα γίνεται με τη δήλωσή της. Η σύνταξη μιας τέτοιας δήλωσης είναι:

```
<τύπος> <όνομα> [=τιμή];
```

(προς το παρόν παραλείπουμε την *ορατότητα*)

Στη διατύπωση ενός ορισμού σύνταξης μιας εντολής, το < ... > σημαίνει <u>υποχρεωτικό</u> μέρος της εντολής, ενώ το [...] σημαίνει <u>προαιρετικό</u>.

π.χ.,
```
int a = 5;
double b;
char answer;
```

- Άλλος τρόπος δήλωσης πολλών μεταβλητών μαζί (του ίδιου τύπου):

```
<τύπος> <μεταβλητή1>, <μεταβλητή2>, ... ;
```

π.χ., `int a1, a2, a3 = 5, a4;`

Τύποι (types) → πρωτογενείς (primitive types)
→ String (αλφαριθμητικά ή συμβολοσειρές)
→ κλάσεις → σύνθετη δομή σχεδιασμένη από τον προγραμματιστή

<u>Βασικοί κανόνες σύνταξης</u>

- Δεσμευμένες λέξεις (reserved): class, public, main, int, double, char, κτλ.
- Αναγνωριστικά (identifiers): ονόματα πραγμάτων (μεταβλητών, κλάσεων, μεθόδων, κτλ.)
  - ξεκινάνε με: γράμμα, _, #
  - περιέχουν: γράμματα, αριθμούς, _, # (όχι κενά, τελείες, *, κτλ.)
  - μέχρι 256 χαρακτήρες (συνήθως 1-15)
  - χωρίς κενά
  - υπάρχει διάκριση μεταξύ κεφαλαίων και μικρών

Συνήθως (κατά σύμβαση):
- ονόματα κλάσεων → ξεκινούν με Κεφαλαίο
- υπόλοιπα ονόματα (μεταβλητών, μεθόδων, κτλ) → ξεκινούν με μικρό
- σταθερές → όλα ΚΕΦΑΛΑΙΑ
- ξεχωρίζουμε λέξεις ενός ονόματος με Κεφαλαίο γράμμα και όχι με _
  π.χ., `numberOfBaskets` αντί για `number_of_baskets`

Π.χ. κάποια ονόματα μεταβλητών:

`my.class`: λάθος (περιέχει τελεία)
`public`: λάθος (δεσμευμένη λέξη)
`7eleven`: λάθος (ξεκινάει με αριθμό)
`Car`: σωστό, αλλά δε συνηθίζεται (μεταβλητή που ξεκινάει με Κεφαλαίο)

Πρωτογενείς τύποι μεταβλητών (primitive types)

byte	$-128 \leq$ ακέραιος ≤ 127
short	$-32768 \leq$ ακέραιος ≤ 32767
int	$-2^{31} \leq$ ακέραιος $\leq 2^{31}-1$ (δις)
long	$-2^{63} \leq$ ακέραιος $\leq 2^{63}-1$ (πεντάκις)
float	δεκαδικός, με εύρος: $\pm 10^{38}$ και ακρίβεια: $\pm 10^{-46}$
double	δεκαδικός, με εύρος: $\pm 10^{308}$ και ακρίβεια: $\pm 10^{-324}$
char	χαρακτήρας Unicode
boolean	true ή false

Θα χρησιμοποιούμε `int` για ακέραιους και `double` για πραγματικούς.

Παραδείγματα:

```
char symbol;
symbol = 'A';
System.out.println(symbol);
```

- Το σύμβολο "=" δεν είναι το μαθηματικό "=". Λέγεται **τελεστής εκχώρησης (assignment operator)**. Πραγματοποιεί *μεταβίβαση της τιμής στα δεξιά του στη μεταβλητή στα αριστερά του*.
  [Περισσότερες λεπτομέρειες θα αναφερθούν στην επόμενη ενότητα]

Άλλα παραδείγματα:

```
int a = 10;
boolean b = true;
double c = 4.2;
```

αλλά και:

```
a = a + 5;
```

```
int x = 1;
int y = 2;
x = y;   (το x γίνεται 2)
```

Τα δύο τελευταία παραδείγματα, αν και από μαθηματικής άποψης μπορούν να θεωρηθούν λάθος, σε μια γλώσσα προγραμματισμού είναι σωστά. Στο πρώτο, απλά αλλάζει η τιμή του `a` και από 10 γίνεται 15. Στο δεύτερο, η μεταβλητή `x` παίρνει την τιμή της μεταβλητής `y`, δηλαδή γίνεται ίση με 2.

- Για αλφαριθμητικά (αλληλουχία χαρακτήρων): Τύπος `String` (είναι κλάση – δεν είναι πρωτογενής τύπος μεταβλητών)

```
String s = "hello";
```

- Η τιμή σε μια μεταβλητή τύπου χαρακτήρα (`char`) δίνεται μέσα σε μονά εισαγωγικά (`'…'`) και είναι *ένας και μόνο ένας* χαρακτήρας.

- Η τιμή σε μια μεταβλητή τύπου `String` (αλφαριθμητικό) δίνεται μέσα σε διπλά εισαγωγικά (`"…"`) και μπορεί να είναι *από ένας μέχρι πολλοί χαρακτήρες* που μπορεί να έχουν και κενά μεταξύ τους *(περισσότερα σε επόμενο μάθημα)*.

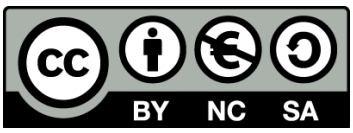
- Η τιμή μιας μεταβλητής `boolean` μπορεί να είναι αποκλειστικά `true` ή `false` και δίνεται *χωρίς εισαγωγικά*. Συνήθως η τιμή μιας τέτοιας μεταβλητής προκύπτει από το αποτέλεσμα κάποιας λογικής πράξης, όπως θα δούμε σε επόμενο μάθημα.

Σημειώματα

Σημείωμα Αναφοράς

Μιχάλης Δρακόπουλος. «Πληροφορική ΙΙ. Ενότητα 1: Εισαγωγικές έννοιες». Έκδοση: 1.0. Αθήνα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: http://opencourses.uoa.gr/courses/MATH106/.

Σημείωμα Αδειοδότησης



[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Διατήρηση Σημειωμάτων

- Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.