Κατασκευή Προγράμματος

Στο σημείο αυτό έχουμε ορίσει τα περισσότερα από τα βασικά εργαλεία για να γράψουμε ένα πλήρες πρόγραμμα σε Java. Δεν έχουμε αναφέρει όμως τη δομή ενός προγράμματος, τον τύπο αρχείων που θα χρησιμοποιηθούν ή πως θα τα μεταγλωττίσουμε και εκτελέσουμε.

Η ρουτίνα main() Στη Java, όπως ακριβώς και στις γλώσσες στις οποίες βασίστηκε, τις C/C++, κάθε πρόγραμμα ξεκινά την εκτέλεσή του από την ρουτίνα/μέθοδο main(). Αντίθετα με τις παραπάνω γλώσσες όμως, η main() δεν βρίσκεται σε κάποιο αρχείο μόνη της, αλλά ως μέθοδος σε κάποια κλάση.

Ακολουθεί ένα παράδειγμα προγράμματος σε Java:

```
 class Example

{

 public static void main(String args[])

 {

 System.out.println(“hello everyone”);

}

 }
```

Οι λέξεις public, static και void έχουν ειδική σημασία, καθώς επίσης και η παράμετρος args[]. Όσον αφορά τη void έχουμε ήδη αναφέρει ότι χρησιμοποιείται όταν η μέθοδος δεν επιστρέφει κάποιο αποτέλεσμα.

Στην περίπτωση της main() σημαίνει ότι το πρόγραμμα δεν επιστρέφει κάποιο κωδικό επιτυχίας στο λειτουργικό σύστημα.

Η public αφορά την πρόσβαση σε κάποια μεταβλητή ή μέθοδο μιας κλάσης και σημαίνει ότι το στοιχείο που χαρακτηρίζει είναι προσβάσιμο από οποιοδήποτε άλλο αντικείμενο, ακόμη και εκτός της ίδιας της κλάσης.

Για τη main() είναι απαραίτητος χαρακτηρισμός γιατί εκτελείται από κώδικα εκτός της ίδιας της κλάσης (το λειτουργικό σύστημα).

Τέλος, η static χαρακτηρίζει δεδομένα ή μεθόδους τα οποία θέλουμε να καλέσουμε πριν αυτά δημιουργηθούν σε κάποιο αντικείμενο. Διαφορετικά, θα ήταν αναγκαία η δημιουργία ενός αντικειμένου της κλάσης Example με τη new, για να μπορέσουμε να καλέσουμε τη main().

Αλλά κάτι τέτοιο δεν είναι δυνατό, γιατί θα έπρεπε να εκτελέσουμε κώδικα πρίν εκτελεστεί η main(), αλλά η main() είναι αυτή που εκτελείται πρώτη σε ένα πρόγραμμα Java.

Αρχεία .java

Ο πηγαίος κώδικας ενός προγράμματος (source code) ή μονάδα μεταγλώττισης (compilation unit) όπως καλείται στη Java, περιέχεται σε αρχεία των οποίων η κατάληξη είναι .java.

Το όνομα του αρχείου πρίν την κατάληξη πρέπει να είναι το ίδιο με το όνομα της κλάσης την οποία περιέχει. Σε άλλες γλώσσες προγραμματισμού, κάτι τέτοιο δεν είναι αναγκαίο, αλλά για την Java είναι απαραίτητο για επιτυχημένη μεταγλώττιση του πηγαίου κώδικα. Η μεταγλώττιση γίνεται χρησιμοποιώντας την εντολή javac (java compiler), ως εξής:

(source code directory) #javac Example.java (UNIX/Linux)

C:\(source code directory)\> javac Example.java (Windows)

Η εντολή αυτή θα παράγει τον εκτελέσιμο κώδικα της Java (το java bytecode, όπως αναφέραμε στην αρχή του παρόντος κειμένου), υπό τη μορφή αρχείου με το ίδιο όνομα αλλά με διαφορετική κατάληξη. Η κατάληξη του java bytecode είναι .class.

Αρχεία .class

Τα αρχεία .class είναι αυτά που εκτελούνται μέσω του Java Virtual Machine (JVM). Η κλήση του JVM δε γίνεται άμεσα, αλλά μέσω της εντολής java.

(source code directory)# java Example (UNIX/Linux)

hello everyone

C:\(source code directory)\> java Example (Windows)

hello everyone

Σημειώστε ότι δε προσθέτουμε την κατάληξη .class στο όνομα του αρχείου Example. Για κάθε αρχείο .java, ο compiler της java δημιουργεί ένα αντίστοιχο αρχείο .class. Για την εκτέλεση ενός προγράμματος απαιτούνται όλα τα αρχεία .class που αντιστοιχούν στα αρχεία .java που χρησιμοποιεί το πρόγραμμα.

Αρχεία .jar

Στην περίπτωση που χρησιμοποιούμε πολλές κλάσεις σε ένα πρόγραμμα, δηλαδή πολλά αρχεία .java και κατά συνέπεια πολλά αρχεία .class, για να εκτελέσουμε ένα αρχείο σε ένα άλλο υπολογιστή, θα έπρεπε να μεταφέρουμε όλα τα αρχεία .class και να τα εκτελέσουμε επί τόπου. Κάτι τέτοιο δεν είναι ιδιαίτερα βολικό ειδικά για μεγάλο αριθμό αρχείων. Για το σκοπό αυτό, η Java παρέχει ένα σύστημα ομαδοποίησης των αρχείων .class σε ένα αρχείο JAR (Java Archive) με αντίστοιχη κατάληξη .jar.

Η δημιουργία ενός τέτοιου αρχείου είναι αρκετά απλή και χρησιμοποιεί την εντολή jar (με τον ίδιο τροπο σε Windows και UNIX/Linux):

C:\(source code directory) # jar cvf classes.jar MyClass1.class

Another.class

Η εκτέλεση του κώδικα που βρίσκεται στο αρχείο classes.jar γίνεται με τον εξίσου απλό τρόπο:

C:\(source code directory)\> java classes.jar

Στην πραγματικότητα τα αρχεία jar δεν είναι παρά συμπιεσμένα αρχεία zip με άλλη κατάληξη και επιπλέον πληροφορίες.

Μέχρι τώρα παρουσιάσαμε ορισμένα παραδείγματα προγραμμάτων σε Java αλλά δεν αναλύσαμε περισσότερο τη λειτουργία τους και πολύ περισσότερο τη ροή τους. Η ροή ενός προγράμματος αναφέρεται στην διαδοχή της εκτέλεσης των εντολών στον επεξεργαστή (είτε είναι πραγματικός είτε εικονικός στη περίπτωση της Java με το JVM).

Η κατανόηση και εμπέδωση του προγραμματισμού ως έννοια, προϋποθέτει την αναγνώριση της ροής ενός προγράμματος από τη μελέτη του πηγαίου του κώδικα. Από τη μελέτη αυτή μπορούμε να βγάλουμε τα εξής συμπεράσματα:

• Αναγνώριση των κλάσεων που χρησιμοποιεί το πρόγραμμα και των σχέσεων μεταξύ τους.

• Πληροφορίες για τα δεδομένα και τους τύπους δεδομένων που χρησιμοποιούνται σε κάθε κλάση.

• Πληροφορίες για τις συναρτήσεις/μεθόδους που χρησιμοποιεί η κάθε κλάση.

• Πληροφορίες για τη σειρά κλήσης κάθε μεθόδου των κλάσεων.

• Πληροφορίες για τη δομή της κάθε μεθόδου και πιθανές παραμετροποιήσεις της λειτουργίας των.

• Από ποιά κλάση ξεκινάει το πρόγραμμα (δηλαδή σε ποιά κλάση περιέχεται η μέθοδος main()).

Απώτερος σκοπός είναι να συλλάβουμε τη γενική εικόνα της λειτουργίας του προγράμματος. Θα θεωρήσουμε το ακόλουθο παράδειγμα για καλύτερη κατανόηση των προαναφερθέντων σημείων:

```
1 class ArgsExample {

2 public static void main(String args[]) {

3 for (int i=0; i < args.length; i++) {

4 if (args[i].equals(“-file”) == true) {

5 if (i+1 < args.length)
```

6 System.out.println(“FILE: “+args[i+1]);

7 }

8 }

9 }

10 }

Στις γραμμές 1-3 η διαδικασία είναι γνωστή: ορίζεται η κλάση, δηλώνεται η βασική ρουτίνα main() και ξεκινάει το loop της εντολής for. Στη γραμμή 4 χρησιμοποιούμε την μέθοδο equals() της κλάσης String για να ελέγξουμε μία-μία τις παραμέτρους args[] αν κάποια είναι ίση με “-file”. Αν ισχύει κάτι τέτοιο (δηλαδή αν η equals() επιστρέψει true) τότε βεβαιωνόμαστε ότι υπάρχει επόμενη παράμετρος (το i+1 είναι μικρότερο του μεγέθους του πίνακα, γραμμή 5) και τυπώνουμε την επόμενη παράμετρο (γραμμή 6). Ο έλεγχος συνεχίζεται για τις υπόλοιπες παραμέτρους args[]. Αφού μεταγλωττίσουμε το πρόγραμμά μας με τη javac, έστω ότι το εκτελούμε με κάποιες παραμέτρους στη γραμμή εντολών:

javac ArgsExample.java

java ArgsExample one -file two three -file

FILE: two