ΕΚΠΑΙΔΕΥΤΙΚΟ ΕΓΧΕΙΡΙΔΙΟ

GETTING STARTED

SDC A.E.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Η Γλώσσα προγραμματισμού JAVA

Στις αρχές της δεκαετίας του 90, μία ομάδα προγραμματιστών, της εταιρείας Sun Microsystems, ανέλαβαν να δημιουργήσουν προγράμματα, με τα οποία θα μπορούσαν να κάνουν καλύτερη τη χρήση και τη λειτουργία των οικιακών ηλεκτρονικών συσκευών. Χρησιμοποίησαν στη αρχή σαν γλώσσα προγραμματισμού τη C++. Επειδή όμως, η C++ τους δυσκόλευε σαν γλώσσα σε θέματα όπως διαχείριση μνήμης, πολλαπλής κληρονομικότητας κλπ, αποφάσισαν ότι έπρεπε να δημιουργηθεί μία νέα γλώσσα που θα κρατούσε τα καλά χαρακτηριστικά της C++, αλλά θα πρόσθετε καινούργιες δυνατότητες και θα ήταν αρκετά αξιόπιστη.
Στη πορεία διαπιστώθηκε ότι το όλο εγχείρημα δεν θα παρουσίαζε εμπορική επιτυχία. Εκείνη τη εποχή, το Internet αρχίζει να ξεφεύγει από τη κατάσταση του απλού κειμένου και αρχίζουν να χρησιμοποιούνται γραφικά μέσω του παγκόσμιου ιστού. Η εταιρεία αντιλαμβάνεται ότι η νέα γλώσσα ταιριάζει τέλεια στη νέα κατάσταση που διαμορφώνεται και αποφασίζει να διαθέσει τη γλώσσα δωρεάν, ώστε να τυποποιηθεί. Παρουσιάζεται επίσημα από τη εταιρεία Sun, και δημιουργείται το Hot Java, ένα πρόγραμμα πλοήγησης στο Internet. Τότε δημιουργείται το Netscape Navigator με υποστήριξη Java, και η εταιρεία Microsoft παρουσιάζει τον Internet Explorer 3.0 με υποστήριξη Java. Έτσι η Java βρήκε τη θέση της στο χώρο.

Τα *βασικά χαρακτηριστικά* της είναι :

- Είναι σχετικά απλή.
- Είναι *αντικειμενοστρεφής (Object Oriented)*. Η Java χρησιμοποιεί κλάσεις για να οργανώσει τον κώδικα σε λογικές ενότητες. Το πρόγραμμα δημιουργεί αντικείμενα, τα οποία έχουν δύο συνιστώσες : τις μεταβλητές και τις μεθόδους. Οι μεταβλητές περιγράφουν τι είναι τα αντικείμενα, ενώ οι μέθοδοι περιγράφουν τι κάνει το αντικείμενο.
- Είναι *μεταγλωττιζόμενη (compiled)*, αλλά μπορεί να χαρακτηριστεί και διερμηνευόμενη (*interpreted*).
- Είναι *ασφαλής*. Έχει σχεδιαστεί με τέτοιο τρόπο, ώστε να παρέχει ασφάλεια εκτέλεσης του κώδικα στο δίκτυο.

- Υποστηρίζει *multithreading*. Ένα πρόγραμμα Java, έχει τη δυνατότητα να περιλαμβάνει πολλές ξεχωριστές διαδικασίες, οι οποίες να εκτελούνται συνεχώς και ανεξάρτητα η μία από τη άλλη. Για παράδειγμα, μπορεί από ένα πρόγραμμα να μεταδίδεται μία εικόνα και συγχρόνως ο χρήστης να εισάγει στοιχεία από το πληκτρολόγιο.
- Δημιουργεί *γρήγορο κώδικα*. Πολλές εταιρείες έχουν αναπτύξει μεταγλωττιστές οι οποίοι το κώδικα byte σε κώδικα γλώσσας μηχανής.
- Δημιουργούμε *δυναμικές ιστοσελίδες*. Συγκεκριμένα μπορούμε :
    - Να παίξουμε ήχο ή βίντεο.
    - Να δημιουργήσουμε κίνηση σε πραγματικό χρόνο.
    - Να δημιουργήσουμε φόρμες εισαγωγής στοιχείων από τους χρήστες.
    - Να χρησιμοποιήσουμε διανυσματικά γραφικά αντί των bitmap εικόνων.
    - Να φτιάξουμε παιχνίδια και εφαρμογές πραγματικού χρόνου, στα οποία να συμμετέχουν πολλοί χρήστες συγχρόνως.

Η Java δημιουργεί *δύο τύπους προγραμμάτων* :

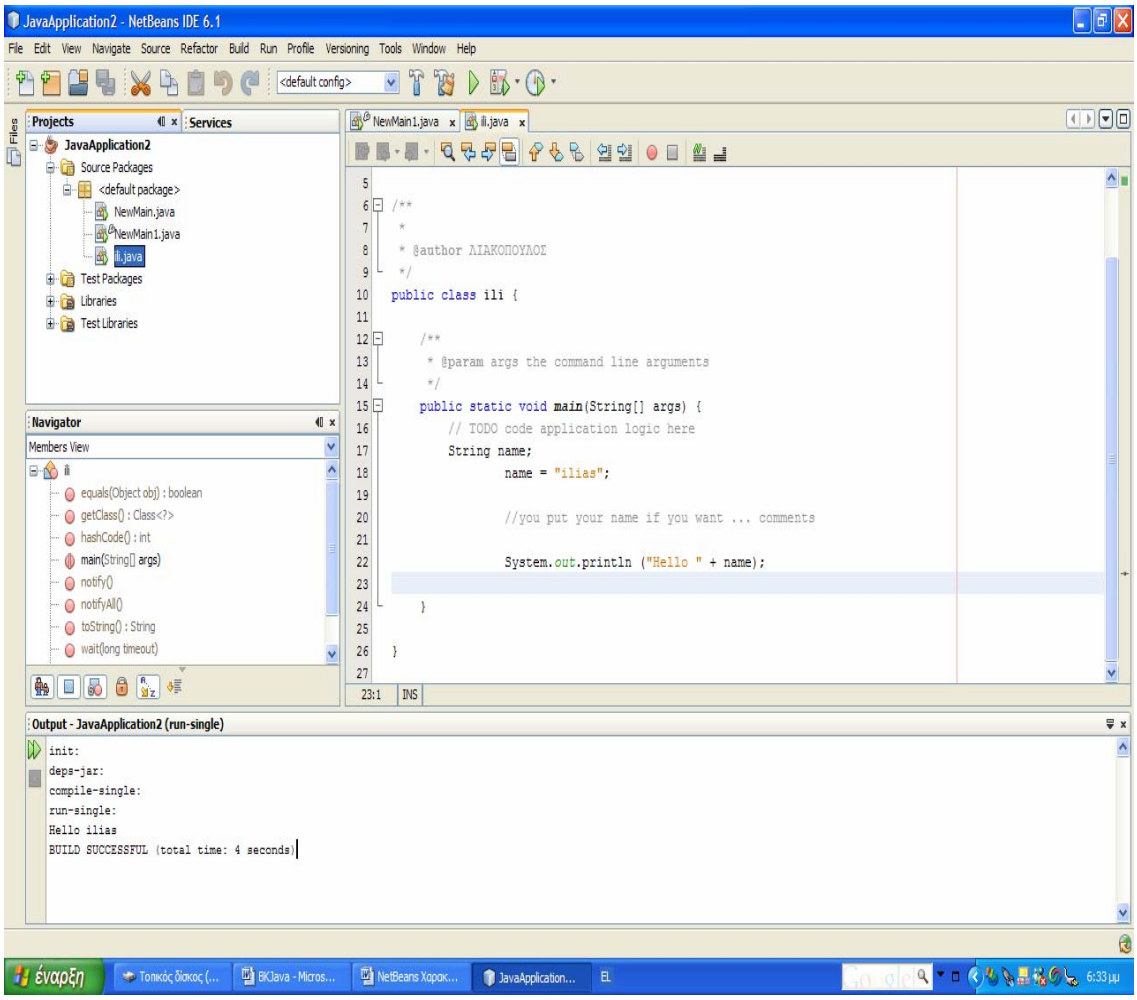
- Ο ένας τύπος είναι τα ***applet***, δηλαδή η δημιουργία δυναμικών ιστοσελίδων.
- Ο άλλος τύπος είναι οι **εφαρμογές (applications)**. Οι εφαρμογές διαιρούνται σε δύο κατηγορίες :
    - Εφαρμογές κονσόλας (console applications), οι οποίες δεν υποστηρίζουν γραφικά και στα Windows τρέχουν σε ένα παράθυρο MS-DOS.
    - Παραθυρικές εφαρμογές (Window Java applications), οι οποίες διαθέτουν πολλά παράθυρα και γενικά χρησιμοποιούν όλα τα στοιχεία ενός γραφικού περιβάλλοντος επικοινωνίας με το χρήστη (GUI).

Διαδικασία εκκίνησης της Java και το NetBeans IDE

Για να μπορέσουμε να γράψουμε προγράμματα σε Java, χρειάζεται να διαθέτουμε το ειδικό πακέτο ανάπτυξης εφαρμογών (**Java development kit ή JDK)**.
Το **JDK** είναι μία γλώσσα προγραμματισμού. Συγχρόνως περιέχει μεταγλωττιστή, διερμηνευτή, αποσφαλματωτή (debugger), αποσυμβολομεταφραστή (disassembler).
Αρχικά θα πρέπει να κατεβάσουμε από τη σελίδα της Sun Microsystems το κατάλληλο JDK, το οποίο θα πρέπει να είναι έκδοσης JDK 1.5.5_0 ή μεταγενέστερο, γιατί το NetBeans το απαιτεί.
Η διαδικασία που θα ακολουθηθεί είναι :

Πηγαίνετε στη σελίδα της Sun → **http://sun.com** , μετά στη επιλογή downloads και από τη συγκεκριμένη σελίδα επιλέξτε να κατεβάσετε το JDK 6. Αφού ολοκληρωθεί το κατέβασμα, τότε εγκαταστήστε το JDK 6 στο υπολογιστή σας, ακολουθώντας βήμα - βήμα τη διαδικασία που προσδιορίζετε από τις οθόνες εγκατάστασης.

Αφού ολοκληρωθεί η εγκατάσταση του JDK 6, το επόμενο βήμα είναι να εγκαταστήσετε το NetBeans IDE που σας παρέχεται. Το περιβάλλον του NetBeans IDE φαίνεται παρακάτω :


JavaApplication2 - NetBeans IDE 6.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
<default config>
Projects
Services
JavaApplication2
Source Packages
<default package>
NewMain.java
NewMain1.java
ili.java
Test Packages
Libraries
Test Libraries
Navigator
Members View
ili
equals(Object obj) : boolean
getClass() : Class<?>
hashCode() : int
main(String[] args)
notify()
notifyAll()
toString() : String
wait(long timeout)
NewMain1.java
ili.java
/**
*
* @author ΛΙΑΚΟΠΟΥΛΟΣ
*/
public class ili {
/**
* @param args the command line arguments
*/
public static void main(String[] args) {
// TODO code application logic here
String name;
name = "ilias";
//you put your name if you want ... comments
System.out.println ("Hello " + name);
}
}
23:1
INS
Output - JavaApplication2 (run-single)
init:
deps-jar:
compile-single:
run-single:
Hello ilias
BUILD SUCCESSFUL (total time: 4 seconds)
έναρξη
Τοπικός δίσκος (...
BKJava - Micros...
NetBeans Χαρακ...
JavaApplication...
EL
6:33 μμ

Ένα δωρεάν, ανοιχτού κώδικα, Περιβάλλον για την Ολοκληρωμένη Ανάπτυξη λογισμικού. Παίρνετε όλα τα εργαλεία που χρειάζεστε για να δημιουργήσετε μία επαγγελματική επιφάνεια εργασίας, προγράμματα για επιχειρήσεις, εφαρμογές web και κινητών συσκευών με τη γλώσσα Java, C/C++, και Ruby. Το NetBeans IDE είναι εύκολο να εγκατασταθεί και να χρησιμοποιηθεί κατευθείαν από το κουτί και να τρέχει σε πολλές πλατφόρμες συμπεριλαμβανομένων των Windows, Linux, Mac OS X και Solaris.

Το NetBeans IDE 6.1 παρέχει αρκετά νέα χαρακτηριστικά και βελτιώσεις, όπως η επεξεργασία με τα πλούσια χαρακτηριστικά του JavaScript, και την ενίσχυση της σύνδεσης με MySQL. Επίσης, αυτή η έκδοση παρέχει βελτιωμένη απόδοση, ειδικά ταχύτερη εκκίνηση (έως 40%), χαμηλότερη κατανάλωση μνήμης και τη βελτίωση της ανταπόκρισης κατά τη διάρκεια της εργασίας με μεγάλα έργα.

Τα *βασικά χαρακτηριστικά* του NetBeans IDE είναι :

1. **Έργα από το κουτί**

   Απλά κατεβάστε και να εγκαταστήστε το NetBeans IDE και είστε έτοιμοι για ξεκίνημα. Με το μικρό του μέγεθος, η εγκατάσταση είναι απλή. Όλα τα IDE εργαλεία και λειτουργίες του έχουν ενσωματωθεί πλήρως - δεν χρειάζεται κανείς να κυνηγάει για plug-ins - και συνεργάζονται στενά όταν ξεκινήσει το IDE.

2. **Ελεύθερο και ανοικτού κώδικα**

   Όταν χρησιμοποιείτε το NetBeans IDE, εντάσσεσθε σε μια δυναμική, ανοικτή πηγή κώδικα, με χιλιάδες χρήστες έτοιμους να βοηθήσουν και να συνεισφέρουν. Υπάρχουν συζητήσεις για το

έργο NetBeans σε mailing lists, blogs για PlanetNetBeans, και απαντήσεις σε συχνές ερωτήσεις (FAQs).

3. **Ισχυρό GUI Builder**

   Η GUI Builder υποστηρίζει ένα περίτεχνο και απλοποιημένο Swing Application Framework και Beans Binding. Μπορείτε να στήσετε GUI's εύκολα και γρήγορα.

4. **Υποστήριξη για Java προτύπων και πλατφορμών**

   Το IDE παρέχει ολοκληρωμένες λύσεις για όλες τις πλατφόρμες ανάπτυξης Java.

   - Πλήρης υποστήριξη **Java** σε κινητές συσκευές. Πλήρες περιβάλλον για τη δημιουργία, τη δοκιμή και την εκτέλεση εφαρμογών για κινητές συσκευές (PDAs, κινητά τηλέφωνα, κλπ). Με preprocessor blocks, μπορείτε εύκολα να χειρίζεσθε θέματα κατακερματισμού.
   - Υποστήριξη **Java Enterprise Edition (EE) 5:** Είναι η πρώτη ελεύθερη, ανοικτού κώδικα IDE που υποστηρίζει Java EE 5. Επίσης υποστηρίζει Java EE 5 με σχεσιακά αντικείμενα χαρτογράφησης, χρησιμοποιώντας Java Persistence API.
   - Υποστήριξη του **Java Standard Edition (SE):** Μπορείτε να αναπτύξετε εφαρμογές χρησιμοποιώντας τα νεότερα Java SE πρότυπα.

5. **Profiling και εργαλεία εντοπισμού σφαλμάτων**

   Με το NetBeans IDE profiler, έχετε πληροφορίες σε πραγματικό χρόνο για χρήση μνήμης και πιθανής συμφόρησης της απόδοσης του συστήματος. Επιπλέον, μπορείτε να δημιουργήσετε ειδικά εργαλεία κώδικα για να αποφευχθεί η υποβάθμιση των επιδόσεων κατά τη διάρκεια Profiling. Το εργαλείο Heap Walker σας βοηθά

να αξιολογήσετε τα περιεχόμενα του Java heap και να βρείτε διαρροές μνήμης.

6. **Υποστήριξη Ruby και Ruby on Rails**

   Και τα δύο, Ruby και JRuby on Rails είναι διαθέσιμα. Μπορείτε να μεταβείτε εύκολα μεταξύ των δύο. Οι προηγμένες δυνατότητες επεξεργασίας Ruby, καθιστά εύκολη την δημιουργία και την τροποποίηση εφαρμογών Ruby.

7. **Υποστήριξη Service Oriented Architecture (SOA)**

   Υποστηρίζει SOA σε σύνθετες εφαρμογές και εργαλεία, όπως BPEL, WSDL, XSD.

8. **Επεκτάσιμη πλατφόρμα**

   Ξεκινήστε με αυτή την επεκτάσιμη πλατφόρμα και προσθέστε τα δικά σας χαρακτηριστικά και επεκτάσεις στο NetBeans IDE. Κατασκευάστε μία εφαρμογή παρόμοια της IDE, κρατώντας μόνο τα χαρακτηριστικά που θέλετε. Επεκτείνοντας την πλατφόρμα, εξοικονομεί χρόνο και βελτιστοποιεί τις επιδόσεις.

9. **Προσαρμοζόμενα Έργα**

   Μέσω του NetBeans IDE, που βασίζεται σε βιομηχανικά πρότυπα, όπως Apache Ant, Maven, και rake, μπορείτε εύκολα να προσαρμόσετε τα έργα σας και να προσθέσετε λειτουργικότητα. Μπορείτε να κατασκευάσετε, να τρέξετε, και να αναπτύξετε έργα σε διακομιστές εκτός του IDE.

10. **Υποστήριξη Visual Web**

    Το NetBeans IDE παρέχει ένα οπτικό περιβάλλον ανάπτυξης, εργαλεία, καθώς και drag and drop στοιχεία που απλουστεύσουν την ανάπτυξη ιστοσελίδων.

11. **Υποστήριξη κώδικα εκτός της Java**

    Δεν περιορίζεσθε στη γλώσσα προγραμματισμού Java. Μπορείτε να συμπεριλάβετε πολλές άλλες γλώσσες προγραμματισμού, όπως C++, C, γλώσσες προγραμματισμού όπως η JavaScript, Ruby, κλπ. Ακόμη πιο συναρπαστικό, είναι να καθορίσετε τη δική σας γλώσσα και να την συμπεριλάβετε στα έργα σας.

12. **Τεχνική Υποστήριξη**

    Για οποιαδήποτε βοήθεια μπορείτε να απευθυνθείτε στην Sun Developer support packages, που προσφέρει συμβουλές προγραμματισμού και υποστήριξη λογισμικού.

Δημιουργία μιας εφαρμογής Java

Για να δημιουργήσεις μια εφαρμογή Java πρέπει να ακολουθήσεις τα εξής βήματα :

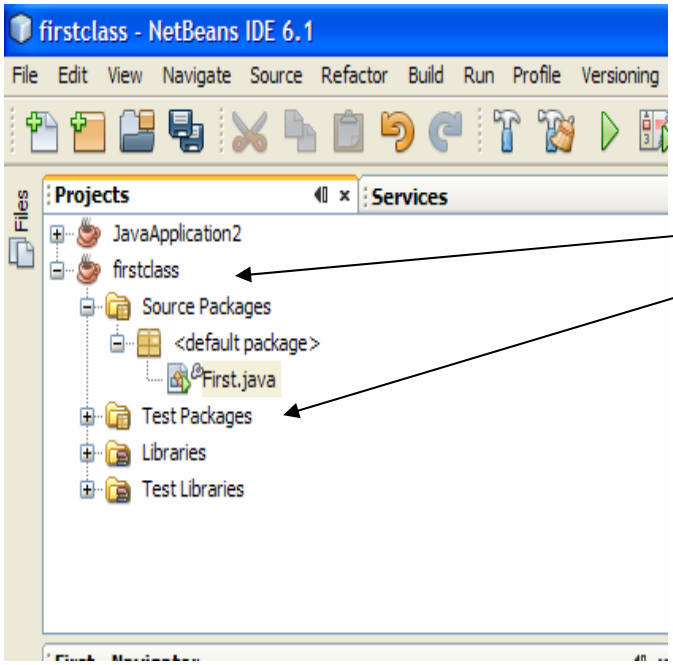
Αρχικά πρέπει να δημιουργηθεί ένα project το οποίο θα περιέχει όλα τα αρχεία (εφαρμογές) του project.

Για τη δημιουργία του project κάνετε τα εξής :

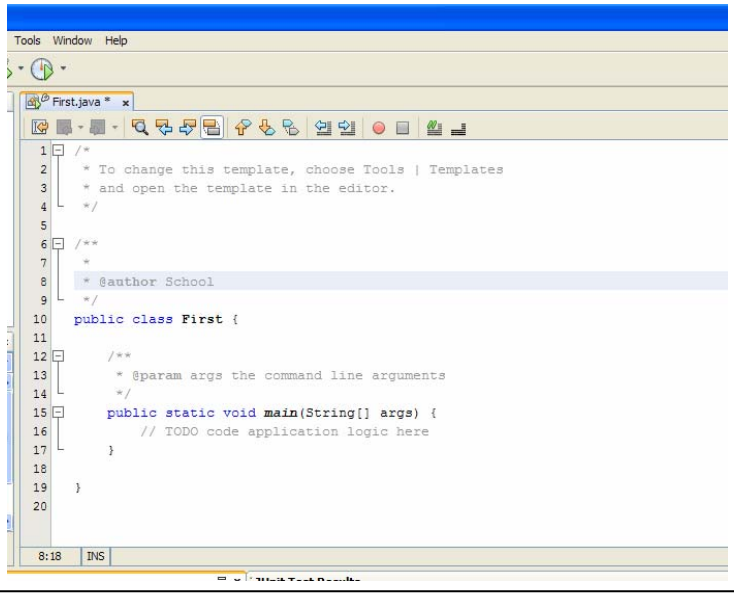
File new project → στη καρτέλα categories επιλέγεις Java → στη καρτέλα projects επιλέγεις Java Applications → κλικ Next. Στη επόμενη καρτέλα που ανοίγει επιλέγεις : Project Name, Project Location, Project Folder. Απενεργοποιείς τις επιλογές Create Main Class και Set as Main Project → κλικ στο Finish.

Αμέσως μετά πρέπει να δημιουργηθεί αρχείο της εφαρμογής μέσα στο project, το οποίο από μόνο θα έχει προέκταση .java. Για να δημιουργηθεί κάνεις τα εξής :

File new File → στη καρτέλα που ανοίγει, στο project επιλέγεις το project στο οποίο θα καταχωρηθεί το αρχείο, στη επιλογή categories επιλέγεις Java, και στη καρτέλα file types επιλέγεις Java main Class. → κλικ Next . Στη επόμενη καρτέλα επιλέγεις όνομα για τη κλάση και μετά κλικ στο Finish. Αφού ολοκληρωθεί επιτυχώς η παραπάνω διαδικασία, έχετε τη παρακάτω οθόνη :



Βλέπετε το project firstclass που δημιουργήθηκε, καθώς και τη εφαρμογή First.java.

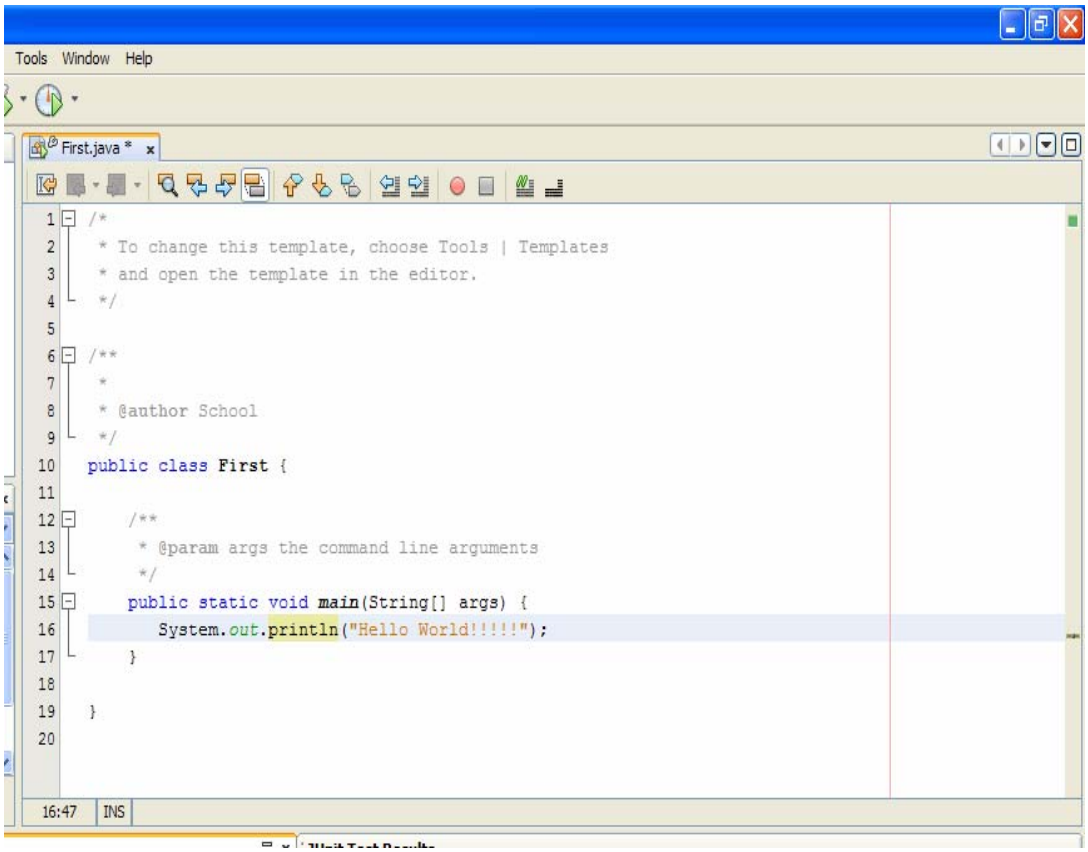


Βλέπετε το αρχείο First.java, στο οποίο είμαστε έτοιμοι να πληκτρολογήσουμε τον κώδικα του προγράμματος.

Πληκτρολογούμε τη εντολή που ακολουθεί στο αρχείο First.java.

System.out.println ("Hello World!!!!!!!");

Πρέπει να δώσουμε προσοχή στη πληκτρολόγηση γιατί στη Java έχουν διαφορά τα κεφαλαία από τα πεζά. Επίσης, κάθε εντολή στη Java ολοκληρώνεται με το ερωτηματικό (;). Οπότε το αρχείο First. java διαμορφώνεται ως εξής :



Στο σημείο αυτό είναι απαραίτητο να εξηγήσουμε ότι βλέπουμε στο ορισμό του προγράμματος.

Class First → Ορισμός της κλάσης First. Ο ορισμός περιλαμβάνει μόνο μία μέθοδο, τη main().

Οι αγκύλες {} → πηγαίνουν ανά ζεύγη και περικλείουν αυτόνομα κομμάτια κώδικα.

Public → Η λέξη public δηλώνει ότι η μέθοδος είναι προσπελάσιμη από παντού.

static → Η λέξη static δηλώνει ότι η μέθοδος είναι προσπελάσιμη ακόμη και αν δεν έχουν δημιουργηθεί αντικείμενα της κλάσης.

Void → σημαίνει ότι δεν επιστρέφει καμιά τιμή.
Main () → Η βασική μέθοδος.

Μέσα στη Main() υπάρχει η εντολή : System.out.println("Hello World!!");

System → Είναι το όνομα μίας βασικής κλάσης, η οποία περιλαμβάνει αντικείμενα και μεταβλητές, για υποστήριξη εισαγωγής δεδομένων από το πληκτρολόγιο. Επίσης χρησιμοποιείται και για την έξοδο χαρακτήρων στην οθόνη.

out → Το αντικείμενο out δηλώνει σαν τυπική έξοδο την οθόνη και είναι μέλος της κλάσης System.

println () → Είναι μέθοδος του αντικειμένου out και τυπώνει στην οθόνη την φράση που βρίσκεται μέσα στην παρένθεση. Προσοχή στο ερωτηματικό (;), το οποίο πρέπει να είναι στο τέλος κάθε εντολής.

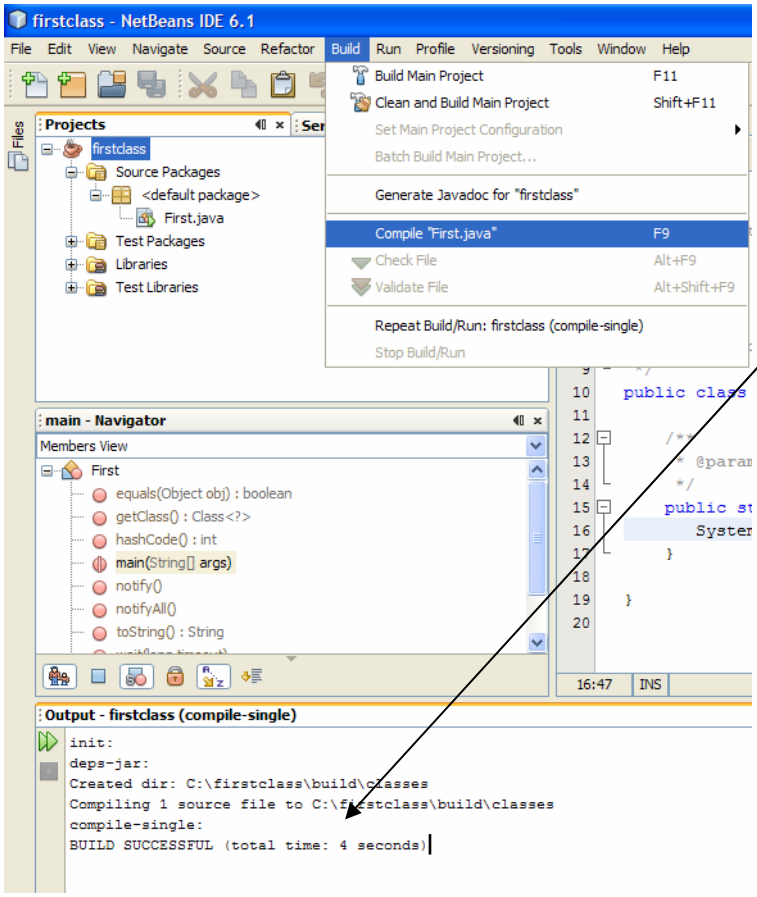
Μεταγλώττιση ενός προγράμματος

Τρόπος Α'

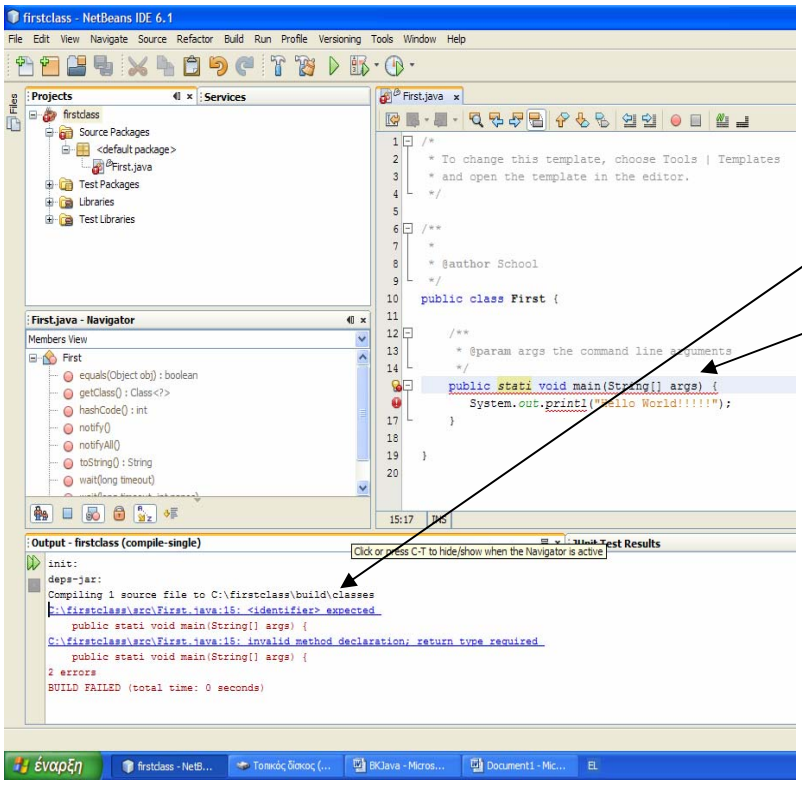
Χρησιμοποιώντας το NetBeans μεταγλωττίζουμε ένα πρόγραμμα με τη ακόλουθη διαδικασία :

Στη γραμμή μενού → κλικ στη επιλογή Build → κλικ στη επιλογή Compile "First.java". Αν η διαδικασία είναι επιτυχής, δηλαδή δεν υπάρχουν συντακτικά λάθη, τότε στη οθόνη του NetBeans στη περιοχή output, στο κάτω μέρος της οθόνης, παίρνεις το μήνυμα BUILD SUCCESSFUL.
Αν υπάρχουν λάθη, τότε στη περιοχή του output εμφανίζεται ο συνολικός αριθμός των λαθών και συγχρόνως στη περιοχή του προγράμματος υπογραμμίζονται με κόκκινο χρώμα τα λάθη τα οποία εντοπίστηκαν.

Η παραπάνω διαδικασία φαίνεται στις παρακάτω οθόνες :



Διαδικασία επιτυχούς μεταγλώττισης του προγράμματος First.java.



Διαδικασία μεταγλώττισης με λάθη στο κώδικα του προγράμματος

Αν η διαδικασία μεταγλώττισης είναι επιτυχής τότε δημιουργείται το First.class αρχείο, που είναι ένα αρχείο που περιέχει κώδικα Byte.

Που βρίσκονται τα αρχεία First.java και First.class ;

Αν υποθέσουμε ότι σαν όνομα project δώσαμε firstclass, και σαν project Location δώσαμε το δίσκο (c:\) , τότε μετά τη μεταγλώττιση στο c:\ δημιουργήθηκε φάκελος με το όνομα του project δηλαδή firstclass.
Το First.java βρίσκεται στο path → c:\firstclass\src.
Το First.class βρίσκεται στο Path → c:\firstclass\built\classes.

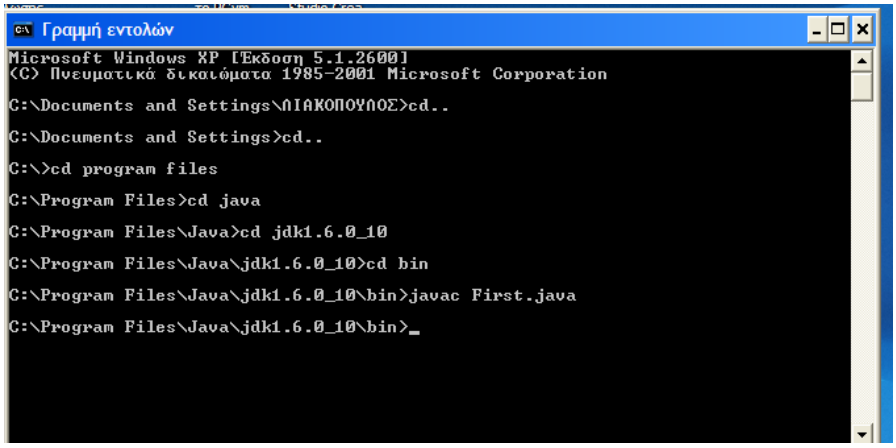
Τρόπος Β'

Μπορούμε να τρέξουμε το πρόγραμμα μας από τη γραμμή εντολών του MS-DOS. Ανοίγουμε ένα τέτοιο παράθυρο με το εξής τρόπο :

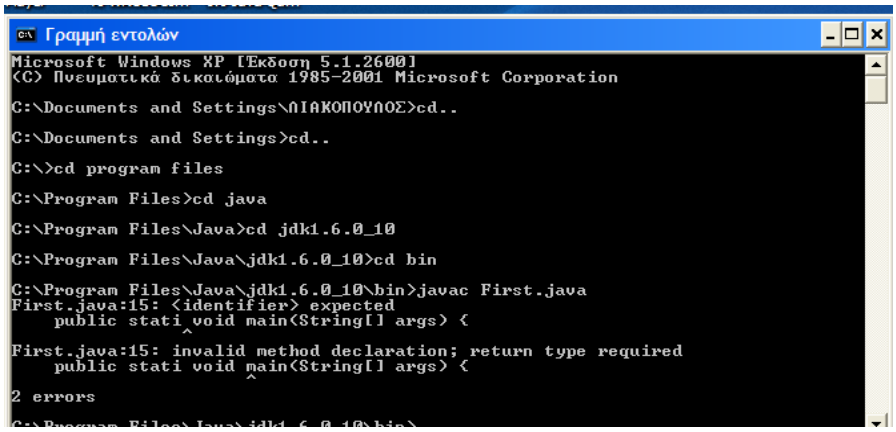
Έναρξη → Προγράμματα → βοηθήματα → Γραμμή Εντολών.
Το αρχείο μας First.java, πρέπει να το αποθηκεύσετε στο φάκελο bin του JDK, δηλαδή στο path : c:\program files\java\jdk1.6.0_10\bin. Η εντολή μεταγλώττισης ενός προγράμματος είναι :

Javac First.java

Αν δεν υπάρχουν συντακτικά λάθη τότε εμφανίζεται η παρακάτω οθόνη:



Αν υπάρχουν συντακτικά λάθη, τότε στη οθόνη της γραμμής εντολών εμφανίζονται τα λάθη όπως φαίνονται στη οθόνη που ακολουθεί :

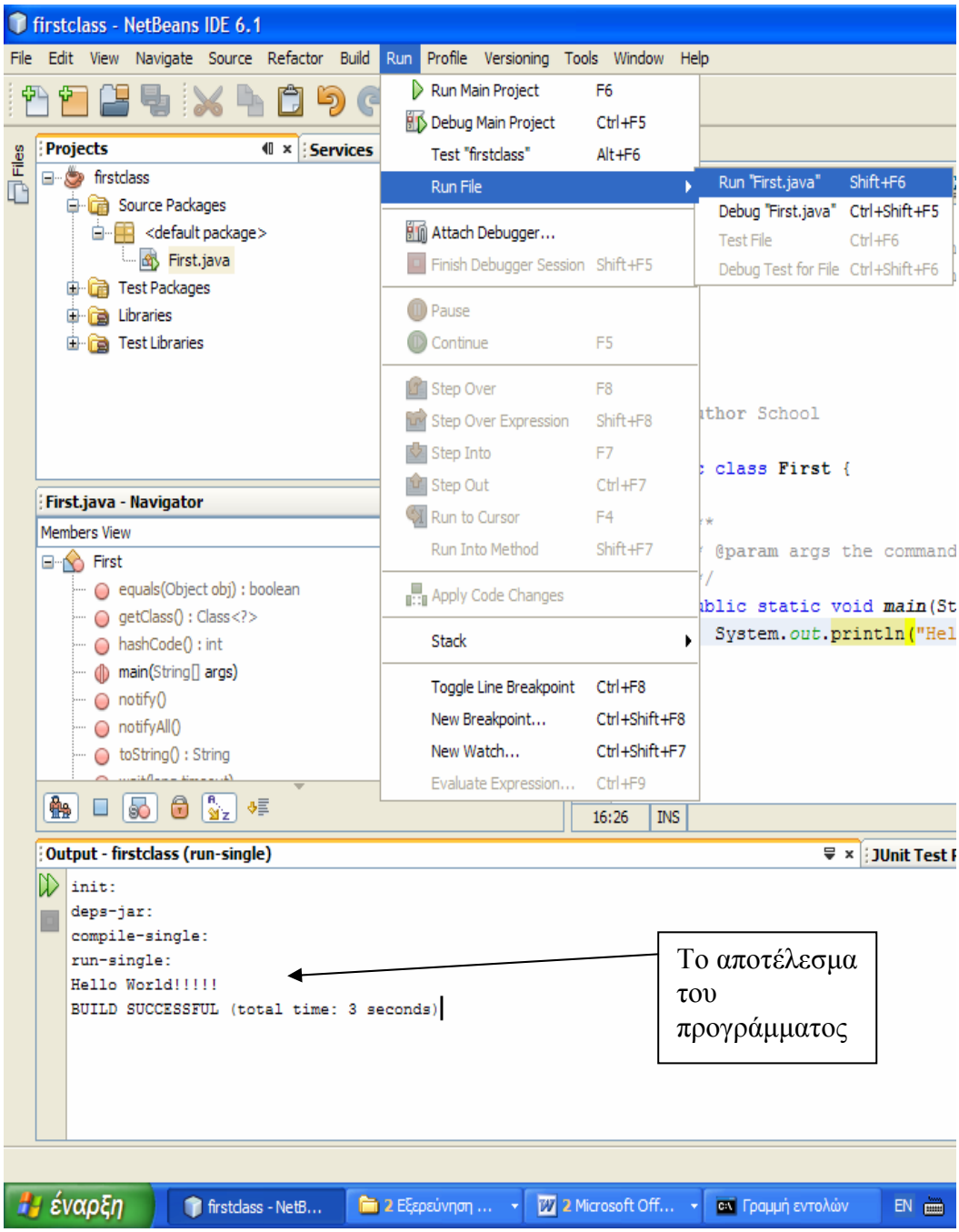


Εκτέλεση ενός προγράμματος

Τρόπος Α'

Για να εκτελέσετε ένα πρόγραμμα από το NetBeans πρέπει :

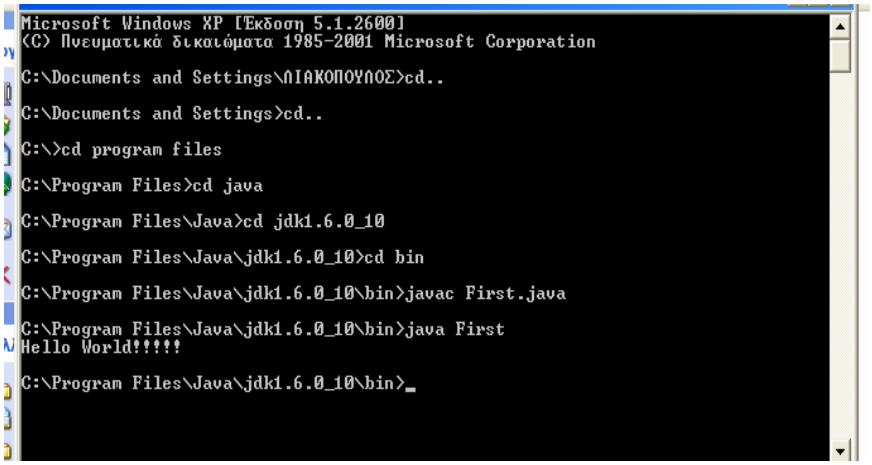
Run → Run File → Run "First.java". Παρακάτω φαίνεται η διαδικασία εκτέλεσης, και το αποτέλεσμα στη περιοχή του output.


firstclass - NetBeans IDE 6.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Run Main Project F6
Debug Main Project Ctrl+F5
Test "firstclass" Alt+F6
Run File
Run "First.java" Shift+F6
Debug "First.java" Ctrl+Shift+F5
Test File Ctrl+F6
Debug Test for File Ctrl+Shift+F6
Attach Debugger...
Finish Debugger Session Shift+F5
Pause
Continue F5
Step Over F8
Step Over Expression Shift+F8
Step Into F7
Step Out Ctrl+F7
Run to Cursor F4
Run Into Method Shift+F7
Apply Code Changes
Stack
Toggle Line Breakpoint Ctrl+F8
New Breakpoint... Ctrl+Shift+F8
New Watch... Ctrl+Shift+F7
Evaluate Expression... Ctrl+F9
Projects
Services
firstclass
Source Packages
<default package>
First.java
Test Packages
Libraries
Test Libraries
First.java - Navigator
Members View
First
equals(Object obj) : boolean
getClass() : Class<?>
hashCode() : int
main(String[] args)
notify()
notifyAll()
toString() : String
16:26
INS
Output - firstclass (run-single)
init:
deps-jar:
compile-single:
run-single:
Hello World!!!!!
BUILD SUCCESSFUL (total time: 3 seconds)
Το αποτέλεσμα του προγράμματος
έναρξη
firstclass - NetB...
2 Εξερεύνηση ...
2 Microsoft Off...
Γραμμή εντολών
EN

Τρόπος Β'

Από το περιβάλλον του MS-DOS, για να εκτελέσουμε το First.class αρχείο, γράφουμε στο σημείο των εντολών την εξής εντολή :

Java First (Δεν χρειάζεται να γράψουμε τη προέκταση του αρχείου First.class). Το αποτέλεσμα φαίνεται στη οθόνη που ακολουθεί :



Εκτέλεση προγράμματος όπου απαιτείται εισαγωγή στοιχείων από το πληκτρολόγιο

Ενδέχεται να χρειαστεί να εισάγουμε δεδομένα από το πληκτρολόγιο, κατά το χρόνο εκτέλεσης ενός προγράμματος. Αυτό γίνεται προσθέτοντας παραμέτρους στη εντολή του διερμηνευτή java, δηλαδή

Java όνομα προγράμματος όρισμα1 όρισμα2 όρισμα3 ……..

Η Java αποθηκεύει αυτά τα ορίσματα σε ένα πίνακα με το όνομα args. Το πρώτο όρισμα αντιστοιχεί στο στοιχείο args[0], το δεύτερο στο στοιχείο args[1] κλπ. Αν ο κώδικας του προγράμματος είναι ο παρακάτω, στη

οθόνη που ακολουθεί, φαίνεται η διαδικασία μεταγλώττισης και εκτέλεσης.

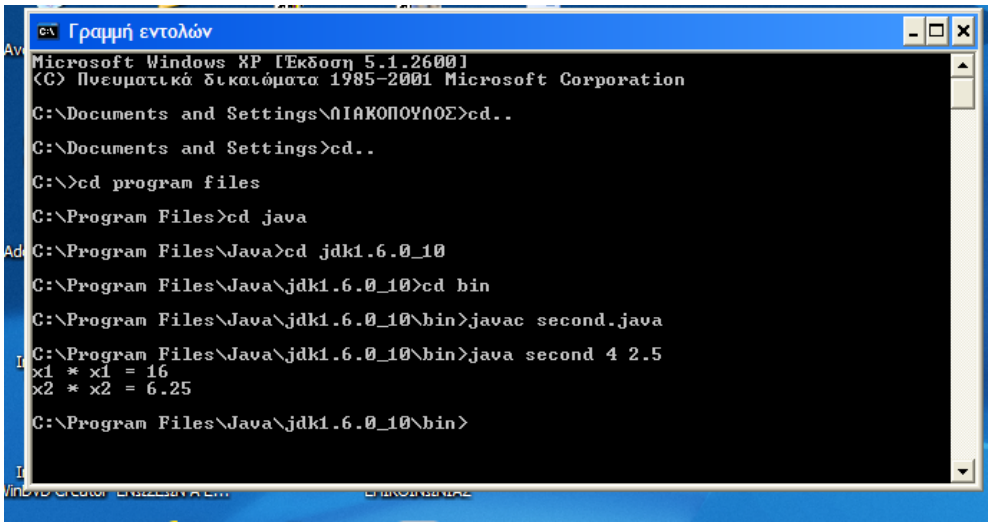
```
public class second
{
        public static void main (String [] args)

            {
                    int x1;
                    double x2;

                    x1= Integer.valueOf (args [0]).intValue ();
                    x1 = x1* x1;

                    x2= Double.valueOf (args[1]).doubleValue ();
                    x2=Math.pow(x2,2);

                    System.out.println ("x1 * x1 = " + x1);
                    System.out.println ("x2 * x2 = " + x2);
                }
}
```



ΕΝΟΤΗΤΑ 1

Δεδομένα, μεταβλητές, τύποι δεδομένων

Βασικές έννοιες

- Δεδομένα - Μεταβλητή
- Ακέραιοι τύποι δεδομένων (byte, short, int, long)
- Αριθμοί κινητής υποδιαστολής (float, double)
- Χαρακτήρες - αλφαριθμητικά
- Λογικές μεταβλητές.

Σχέδιο μαθήματος.

Δεδομένα-Μεταβλητές

- Στη εισαγωγή του μαθήματος, και κατόπιν συζήτησης με τους μαθητές, να αποσαφηνιστούν οι βασικές έννοιες του *input, process και output.*

- Να συνειδητοποιήσουν τι είναι τα *δεδομένα* , τι είναι οι μεταβλητές, πως τις ορίζουμε. Ως *μεταβλητή* στη Java, θεωρούμε μία περιοχής μνήμης στη οποία έχουμε δώσει ένα όνομα και τη χρησιμοποιούμε για αποθήκευση δεδομένων.

- Να καταλάβουν ότι προτού χρησιμοποιήσουμε μία μεταβλητή πρέπει να δηλώσουμε το όνομα της και το τύπο της.

- Η δήλωση μίας μεταβλητής γίνεται συνήθως στη αρχή του προγράμματος , μπορεί όμως να γίνει και σε άλλες θέσεις. Στη αρχή της δήλωσης μπαίνει ο τύπος του δεδομένου (int για ακεραίους, string για αλφαριθμητικό , Boolean για λογικές μεταβλητές) και στη συνέχεια το όνομα της μεταβλητής το οποίο επιλέγουμε εμείς.

- Σαν όνομα μιας μεταβλητής μπορούμε να έχουμε όσους χαρακτήρες θέλουμε, αλλά πρέπει να αρχίζει με γράμμα, ή δολάριο ($), ή το χαρακτήρα υπογράμμισης (_) και να μην περιέχει κενά. Δεν πρέπει να αρχίζει με αριθμό, ούτε να είναι ίδιο με κάποια από τις δεσμευμένες λέξεις της Java.

- Πρέπει να επισημάνουμε ότι στη Java έχουν διαφορά τα κεφαλαία από τα πεζά γράμματα στη δήλωση μιας μεταβλητής.

- Να καταλάβουν γιατί έχουμε διαφορετικούς τύπους δεδομένων και τη διαφορά της έννοιας χαρακτήρας και αλφαριθμητικό.

Ακέραιοι τύποι δεδομένων (byte, short, int, long)

- Οι μεταβλητές που κρατούν ακέραιους τύπους δεδομένων είναι τεσσάρων τύπων : **byte, short, int, long**
  Byte : από -128 έως 127 και δεσμεύει στη μνήμη 8 bits.
  Short : από -32768 έως 32767 και δεσμεύει 16 bits.
  Int : από -2147483648 έως 2147483647 και δεσμεύει 32 bits.
  Long : τεράστιος ακέραιος και δεσμεύει 64 bits.
  Πχ : int number; byte cnt = 100; short price = -100;

Αριθμοί κινητής υποδιαστολής (float, double)

- Αριθμοί που δεν είναι ακέραιοι και περιλαμβάνουν και ακέραιο και δεκαδικό μέρος, και μπορεί να είναι και θετικοί και αρνητικοί. Ανάλογα με τη ακρίβεια που παρέχουν, οι αριθμοί κινητής υποδιαστολής χωρίζονται σε :
  Float: από $-3.4X10^{38}$ έως $3.4X10^{38}$ και δεσμεύει 4 Bytes.
  Double: από $-1.7X10^{308}$ έως $1.7X10^{308}$ και δεσμεύει 8 Bytes.
  Πχ : float num; double num1 = 132.25;

Χαρακτήρες - Αλφαριθμητικά

- Ένα δεδομένο τύπου *χαρακτήρα* περιλαμβάνει έναν οποιοδήποτε χαρακτήρα. Ο χαρακτήρας πρέπει να περικλείεται σε απλά εισαγωγικά.
  Ως *αλφαριθμητικό* δεδομένο εννοούμε ένα σύνολο από χαρακτήρες. Στη Java, το αλφαριθμητικό είναι ένα αντικείμενο της κλάσης String και πρέπει να περικλείεται σε διπλά εισαγωγικά.
  Πχ : **char** onechar = ‘b’; **Sting** name = “Peter”;

Λογικές μεταβλητές

- Πολλές φορές πρέπει να αποφασίσουμε αν μία συνθήκη ισχύει ή όχι. Τότε χρησιμοποιούμε μεταβλητές οι οποίες παίρνουν *λογικές* (*boolean*) τιμές. Οι τιμές αυτές είναι **true** και **false**.
  Πχ boolean state = false;

ΕΝΟΤΗΤΑ 2

Τελεστές, εκχωρήσεις, μετατροπές δεδομένων

Βασικές έννοιες

- Τελεστές αριθμητικών πράξεων (+ , - , * , / , modulo (%))
- Εκχωρήσεις τιμών σε μεταβλητές
- Τελεστές σύγκρισης (== , != , < , > , <= , >=)
- Μοναδιαίοι τελεστές αύξησης και μείωσης (++ , --)
- Λογικοί τελεστές (AND , OR , NOT , XOR με τα σύμβολα τους &&, || , ! , ^ αντίστοιχα)
- Αρχικές τιμές για τις μεταβλητές
- Μετατροπή τύπου δεδομένων ((επιθυμητός τύπος) μεταβλητή)
- Μαθηματικές συναρτήσεις (Η κλάση Math). Βασικές συναρτήσεις όπως : abs, ceil, exp, log, max, min, pow, round, random, sqrt
- Μετατροπή αλφαριθμητικών σε αριθμούς με τις μεθόδους α)valueOf(αλφαριθμητικό) και β) intValue(), longValue(), floatValue(), doubleValue()
- Εισαγωγή δεδομένων από το πληκτρολόγιο κατά το χρόνο εκτέλεσης.

Σχέδιο μαθήματος.

*Τελεστές αριθμητικών πράξεων (+ , - , * , / , modulo (%))*

- Οι τελεστές αριθμητικών πράξεων είναι : πρόσθεσης (+), αφαίρεσης (-), πολλαπλασιασμού (*), διαίρεσης (/), και υπολοίπου

ή modulo (%). Οι τελεστές *, /, % έχουν υψηλότερη προτεραιότητα , ενώ +, - , έχουν χαμηλότερη προτεραιότητα.

Εκχωρήσεις τιμών σε μεταβλητές

- Όταν δηλώνουμε z = z + 3 αυτό σημαίνει ότι υπολογίζεται το δεξί μέρος και το αποτέλεσμα καταχωρείται σαν νέα τιμή του z.

  Η σχέση x = x + y , είναι ισοδύναμη με x += y.
  Η σχέση x = x – y , είναι ισοδύναμη με x -= y.
  Η σχέση x = x * y , είναι ισοδύναμη με x *= y.
  Η σχέση x = x / y , είναι ισοδύναμη με x /= y.

Τελεστές σύγκρισης (== , != , < , > , <= , >=)

- Όλοι οι τελεστές σύγκρισης δέχονται λογικές τιμές (true / false). Αυτοί είναι : ίσον → = = , Διάφορον → != , μικρότερο → < , μεγαλύτερο → > , μικρότερο ή ίσο → <= , μεγαλύτερο ή ίσο → >=. Μεγάλη προσοχή να δοθεί στο τελεστή ισότητας (= =) για να αποφευχθεί η σύγχυση με το τελεστή εκχώρησης (=).

Μοναδιαίοι τελεστές αύξησης και μείωσης (++ , --)

- Με παραδείγματα να δουν την χρήση των μοναδιαίων τελεστών αύξησης και μείωσης , δηλαδή τη διαφορά ανάμεσα στις εντολές y = x++; και y = ++x, y = x- -και y = - - x;
  Πχ : αν υποθέσουμε ότι η μεταβλητή χ τύπου int έχει αρχική τιμή 20, και κάνουμε τη δήλωση y = x++; Η μεταβλητή y θα πάρει τη τιμή 20, ενώ στη συνέχεια η μεταβλητή χ θα αυξήσει τη τιμή της και θα γίνει 21.
  Πχ : : αν υποθέσουμε ότι η μεταβλητή χ τύπου int έχει αρχική τιμή 20, και κάνουμε τη δήλωση y = ++χ; Η μεταβλητή Χ θα αυξήσει πρώτα τη τιμή της και θα γίνει 21 και μετά η τιμή 21 θα καταχωρηθεί στη μεταβλητή Υ. Και οι δύο μεταβλητές μετά τη εκτέλεση της εντολής θα έχουν τη ίδια τιμή.

Λογικοί τελεστές (AND , OR , NOT , XOR)

- Οι **λογικοί τελεστές** είναι οι ακόλουθοι:

Τελεστής	**Περιγραφή**
&&	Λογικός τελεστής AND
||	Λογικός τελεστής OR
!	Λογικός τελεστής NEGATION

Τους συσχετιστικούς τελεστές, τους τελεστές ισότητας και τους λογικούς τελεστές τους συναντάμε κυρίως στις εντολές **if** , **for** , **while**, **do**.
Οι παραπάνω τελεστές χρησιμοποιούνται για συγκρίσεις μεταξύ αριθμών, μεταβλητών και παραστάσεων.
Εάν η σύγκριση είναι **αληθής** τότε το αποτέλεσμα είναι 1 διαφορετικά εάν είναι **ψευδής** τότε το αποτέλεσμα είναι μηδέν.

Αρχικές τιμές για τις μεταβλητές

- Σε όσες μεταβλητές δεν δίνουμε αρχική τιμή όταν τις δηλώνουμε, η Java δίνει από μόνη της μία αρχική τιμή η οποία εξαρτάται από τον τύπο της μεταβλητής. Αυτές είναι :

Τύπος	**Αρχική τιμή**	**Τύπος**	**Αρχική τιμή**
byte	0	short	0
int	0	long	0L
float	0.0f	double	0.0d
char	‘\u0000	boolean	false
Σε αντικείμενο	null		

Μετατροπή τύπου δεδομένων

- *Μετατροπή* δεδομένων εννοούμε τη διαδικασία με τη οποία μετατρέπουμε ένα τύπο δεδομένου σε ένα άλλο. Αυτό γίνεται

τοποθετώντας τον τύπο που θέλουμε να γίνει η μεταβλητή σε παρένθεση, πριν από το όνομα της μεταβλητής.
Δηλαδή (επιθυμητός τύπος) μεταβλητή.
Η περίπτωση μετατροπής τύπου δεδομένων (λ.χ. από τύπο int σε double), θα μπορούσε να γίνει σε περίπτωση πράξεων όπως η διαίρεση. Αντίθετα, μια μετατροπή από double σε int θα είχε σαν αποτέλεσμα στρογγυλοποίηση του αριθμού και θα υπήρχε πρόβλημα στη ακρίβεια αναπαράστασης του

Μαθηματικές συναρτήσεις (Η κλάση Math)

- Η κλάση Math περιέχει μεθόδους με τις οποίες μπορούμε να κάνουμε βασικές πράξεις με εκθετικά , λογαρίθμους, τετραγωνικές ρίζες και τριγωνομετρικές συναρτήσεις. Παραδείγματα:
  double c=Math.pow (a,b); αντιστοιχεί σε $c = a^b$
  double b=Math.cos (a); αντιστοιχεί σε b = cos (a)
  double b=Math.abs (a); αντιστοιχεί σε b = |a|
  Ιστοσελίδα που αφορά την κλάση Math και κάθε άλλη κλάση:
  http://java.sun.com/j2se/1.5.0/docs/api
  http://java.sun.com/javase/6/docs/api

Μετατροπή αλφαριθμητικών σε αριθμούς

- Πολλές φορές χρειάζεται ένα αλφαριθμητικό να μετατραπεί σε αριθμητικό δεδομένο. Για παράδειγμα , το αλφαριθμητικό ‘’15’’ να μετατραπεί σε ακέραιο τύπο int , πρέπει να γράψουμε :

  Int i=Integer.valueOf (“15’’).intValue ();
  Πχ : το αλφαριθμητικό ‘’15.3’’ για να μετατραπεί σε αριθμό τύπου float πρέπει :
  float f=Float.valueOf (“15.3’’).floatValue ();

Εισαγωγή δεδομένων από το πληκτρολόγιο κατά το χρόνο εκτέλεσης.

- Μπορεί να χρειαστεί να εισάγουμε δεδομένα από το πληκτρολόγιο κατά το χρόνο εκτέλεσης ενός προγράμματος. Αυτό γίνεται προσθέτοντας ορίσματα στη εντολή του διερμηνευτή Java όνομα_προγράμματος έτσι ώστε να γίνει :

Java όνομα_προγράμματος όρισμα1 όρισμα2 ……

ΕΝΟΤΗΤΑ 3

Δομές διακλάδωσης

Βασικές έννοιες

- Αποφάσεις – η δομή if – ο όρος else.
- Ένθετα if.
- Ο τελεστής ? :
- Η δομή πολλαπλής διακλάδωσης switch.

Σχέδιο μαθήματος.

Αποφάσεις – η δομή if – ο όρος else.

- Σε όλα σχεδόν τα προγράμματα πρέπει να ληφθεί μία απόφαση. Στο σημείο αυτό διακόπτεται η κανονική ροή του προγράμματος και ανάλογα με τη τιμή μιας έκφρασης, πραγματοποιείται μία *διακλάδωση υπό συνθήκη*. Η πιο συνηθισμένη εντολή διακλάδωσης είναι η δομή if. Στο σημείο αυτό θα πρέπει να εξηγηθεί η δομή if, καθώς και ο όρος else.

  Η γενική μορφή της εντολής **if-else** είναι η ακόλουθη:

```
if (παράσταση) {
  εντολή_1;
  εντολή_2;
   .
}
else {
  εντολή_A;
  εντολή_B;
  .
}
```

  Η απλή μορφή της εντολής **if** είναι η ακόλουθη:

```
if (παράσταση) {
   εντολή_1;
   εντολή_2;
   .
   .
}
```

Η μορφή της εντολής **if** με μία μόνο εντολή είναι η ακόλουθη:

```
if (παράσταση)
  εντολή_1;
```

Η δομή διακλάδωσης περιέχει επίσης και το όρο **else if,** οπότε σχηματίζεται μία δομή **if….else….if.** Η γενική μορφή της εντολής **if-else** συμπεριλαμβανομένης και της **else-if** είναι:

```
if (παράσταση_1) {
   εντολή_1_1;
   εντολή_1_2;
   .
}
else if(παράσταση_2) {
   εντολή_2_1;
   εντολή_2_2;
   .
}
else if(παράσταση_3) {
  εντολή_3_1;
  εντολή_3_2;
  .
}
else {
  εντολή_A;
  εντολή_B;
  .
}
```

- Ένθετα if. Σε πολλές περιπτώσεις η εκτέλεση ενός if ή του else μπορεί να οδηγήσει σε ένα άλλο if. Έχουμε δηλαδή τη περίπτωση ενός *ένθετου (nested) if*. Όταν έχουμε τέτοιες περιπτώσεις, τότε το κάθε else ανήκει στο πλησιέστερο if.

Ο τελεστής ? :

Ο **τριαδικός τελεστής ?:** παρέχει απλά έναν εναλλακτικό τρόπο γραφής της απλής εντολής if και έχει την ακόλουθη σύνταξη:

παράσταση_1 ? παράσταση_2 : παράσταση_3

και αντιστοιχεί στην ακόλουθη σύνταξη της εντολής if:

```
if  (παράσταση_1 )
   παράσταση_2;
else
   παράσταση_3;
```

Πχ : min = a < b ? a : b;

Η δομή πολλαπλής διακλάδωσης switch.

- Υπάρχουν περιπτώσεις που πρέπει να ελέγξουμε τη τιμή μίας μεταβλητής, η οποία έχει τη δυνατότητα να πάρει διαφορετικές τιμές και ανάλογα να εκτελέσει μία σειρά από ενέργειες. Αν και θα μπορούσαμε να το υλοποιήσουμε με μια σειρά από if .. else if …. Else, μπορούμε πιο βολικά να χρησιμοποιήσουμε τη δομή *πολλαπλής διακλάδωσης switch*.

  Η γενική μορφή της εντολής **switch** είναι η ακόλουθη:

```
switch (παράσταση) {
  case σταθερή-παράσταση_1 :
    εντολή_1_1;
    εντολή_1_2;
    .
    break;
  case σταθερή-παράσταση_2 :
    εντολή_2_1;
    εντολή_2_2;
    .
    break;
  .
  .
  default:
    εντολή_Α;
```

εντολή_Β;
.
break;
}

Πρώτα υπολογίζεται η ***παράσταση*** και στη συνέχεια εκτελείται η περίπτωση (**case**) της οποίας η ***σταθερή-παράσταση*** ταυτίζεται με αυτή.

ΕΝΟΤΗΤΑ 4

Εντολές Επανάληψης

Βασικές έννοιες

- Εντολές επανάληψης – βρόχοι
- Ο βρόχος for
- Ο βρόχος while
- Ο βρόχος do – while
- Μετατροπή του for σε while και do – while
- ατέρμονα βρόχος
- Διαφορά χρήσης while και do – while.
- Ένθετοι βρόχοι
- Η εντολή continue σε ένα βρόχο.
- Η εντολή break σε ένα βρόχο και στο switch.

Σχέδιο μαθήματος.

- Ο βρόχος επανάληψης , είναι μία δομή η οποία μας επιτρέπει να επαναλαμβάνουμε μια σειρά από εντολές. Θα πρέπει να επισημανθούν οι δύο βασικές περιπτώσεις επαναλήψεων. Αν γνωρίζουμε εκ των προτέρων το πλήθος των επαναλήψεων που πρέπει να γίνουν, τότε χρησιμοποιούμε το βρόγχο επανάληψης *for* ενώ αν δεν γνωρίζουμε από πριν το πλήθος των επαναλήψεων τότε χρησιμοποιούμε τους βρόχους *while ή do – while*.

Ο βρόχος for

- Η γενική σύνταξη της εντολής **for** είναι η ακόλουθη:

for (*παράσταση_1*;*παράσταση_2*;*παράσταση_3*){
εντολή_1;

εντολή_2;
..
}

Ο βρόχος while

- Η γενική σύνταξη της εντολής **while** είναι η ακόλουθη:

while (*παράσταση*){
εντολή_1;
εντολή_2;
..
}

Ο βρόχος do - while

- Η γενική σύνταξη της εντολής **do-while** είναι η ακόλουθη:

Do {
εντολή_1;
εντολή_2;
..
} while(*παράσταση*);

Μετατροπή του for σε while και do – while

Η εντολή **for** μπορεί να γραφεί χρησιμοποιώντας την εντολή **while** ως:

***παράσταση_1*;**
while(*παράσταση_2*){
εντολή_1;
εντολή_2;
.
***παράσταση_3*;**
}

Ατέρμονας βρόχος

Η δημιουργία ενός *ατέρμονα βρόχου* γίνεται ως εξής:
for (;;){

εντολή_1;
εντολή_2;
.
}

Διαφορά χρήσης while και do – while

- Να επισημανθεί η σημαντική διαφορά χρήσης των δύο βρόχων *while και do – while.*
  Ο βρόχος while δημιουργεί μία επανάληψη μίας ή περισσοτέρων εντολών, με τη διαφορά ότι ο αριθμός των επαναλήψεων δεν είναι προκαθορισμένος αλλά εξαρτάται από μία συνθήκη. Αν η συνθήκη είναι αληθής τότε ο βρόχος εξακολουθεί να εκτελείται. Αν η συνθήκη γίνει ψευδής, τότε ο βρόχος σταματά να εκτελείται και η ροή συνεχίζεται με τις επόμενες εντολές του προγράμματος. Εννοείται, ότι αν η συνθήκη που καθορίζει την εκτέλεση του βρόχου βρεθεί ψευδής στην αρχή, τότε ο βρόχος αυτός δεν θα εκτελεστεί καμία φορά.
  Ο βρόχος do – while είναι παρόμοιος με το βρόχο while, με τη διαφορά ότι η λογική συνθήκη δεν ελέγχεται στην αρχή του βρόχου, αλλά στο τέλος. Αυτό εξασφαλίζει την εκτέλεση του βρόχου τουλάχιστον μία φορά, ακόμη και αν η λογική συνθήκη είναι ψευδής.

Ένθετοι βρόχοι

- Να εξηγηθεί με παραδείγματα η ροή εκτέλεσης προγράμματος χρησιμοποιώντας *ένθετα (nested) βρόχους*. Μπορούμε να τοποθετήσουμε ένα βρόχο μέσε σε ένα άλλο, αρκεί να προσέξουμε τα εξής : α) ο εσωτερικός βρόχος πρέπει να περιέχεται εξ ολοκλήρου στον εξωτερικό. β) Δεν μπορούμε να μπούμε στο εσωτερικό του βρόχου αν δεν περάσουμε από τη πρώτη εντολή του. γ) Δεν μπορούμε να χρησιμοποιήσουμε την ίδια μεταβλητή ως απαριθμητή σε δύο ή περισσότερους βρόχους, που ο ένας εμπεριέχεται στον άλλο.

Η εντολή continue σε ένα βρόχο

- Υπάρχουν περιπτώσεις, όπου θέλουμε να μην εκτελεστεί το περιεχόμενο ενός βρόχου για μία συγκεκριμένη τιμή της συνθήκης

ή του μετρητή. Χρησιμοποιούμε τότε την εντολή continue. Σε περίπτωση που έχουμε ένθετους βρόχους και θέλουμε να ξεφύγουμε από το εσωτερικό για ορισμένες τιμές του μετρητή ή της συνθήκης μεταφέροντας τη ροή στον εξωτερικό βρόχο, τότε χρησιμοποιούμε τη εντολή continue με ετικέτα (label).

Η εντολή break σε ένα βρόχο και στο switch

- Η εντολή break στη δομή switch χρησιμοποιείται για να βγούμε από τη πολλαπλή διακλάδωση και να συνεχίσουμε με την επόμενη εντολή. Ομοίως μπορεί να χρησιμοποιηθεί για έξοδο από ένα βρόχο. Μπορούμε να χρησιμοποιήσουμε το break με τη βοήθεια μιας συνθήκης για να βγούμε από ένα βρόχο. Επίσης αν θέλουμε να βγούμε με μιας από δύο βρόχους, τότε μπορούμε να χρησιμοποιήσουμε το break μαζί με ετικέτα.

ΕΝΟΤΗΤΑ 5

Πίνακες

Βασικές έννοιες

- Πίνακες με μία διάσταση.
- Δήλωση, καταχώρηση και απόδοση αρχικών τιμών στον πίνακα.
- Πίνακες πολλαπλών διαστάσεων
- Ταξινόμηση.

Σχέδιο μαθήματος.

Πίνακες με μία διάσταση

- Να επισημανθεί ότι ο πίνακας έχει τη δυνατότητα να καταχωρεί στοιχεία του ίδιου τύπου. Αντί λοιπόν να ορίσουμε ξεχωριστή μεταβλητή για να κρατά κάθε μία από τις τιμές αυτού του τύπου, είναι προτιμότερο να δημιουργούμε μία *μεταβλητή με δείκτη* ή έναν *πίνακα (array)*. Τα στοιχεία του πίνακα σημειώνονται με κοινό όνομα, που συνοδεύεται από ένα *δείκτη (index or subscript)*. Ο δείκτης είναι ένας ακέραιος αριθμός τοποθετημένος μέσα σε

αγκύλες λ.χ. A[5]. Η δήλωση αυτή μας δείχνει το έκτο στοιχείο του πίνακα, αφού το πρώτο στοιχείο είναι πάντα αυτό με δείκτη 0.

Δήλωση, καταχώρηση και απόδοση αρχικών τιμών στον πίνακα

- Ο πίνακας πρέπει να έχει ένα συγκεκριμένο τύπο, όπως byte, int, float, Sting κτλ. Και όλα τα στοιχεία του πίνακα πρέπει να είναι του ίδιου τύπου. Δεν μπορούμε σε ένα πίνακα να αποθηκεύουμε δεδομένα τύπου int και double. Η δήλωση ενός πίνακα είναι :

  Float [] number; ή float number [];

  Για να δηλώσουμε στο μεταγλωττιστή ότι ο πίνακας number θέλουμε να έχει πέντε στοιχεία και να δεσμευτεί αντίστοιχος χώρος για αυτά στη μνήμη πρέπει να γράψουμε :

  Number = new float [5];

  Βέβαια θα μπορούσαμε να κάνουμε δήλωση και καταχώρηση σε μια εντολή όπως :

  Float number [] = new float [5];

  Θα μπορούσαμε να αποδώσουμε αρχικές τιμές στο πίνακα ως εξής :

  Number [3] = 3.0;

  Για να τυπώσουμε τη τιμή που έχει το τέταρτο στοιχείο του πίνακα τότε : System.out.println (number[3]);

  Για να αρχικοποιήσουμε τον πίνακα σε 0 θα είχαμε το εξής :

```
For (int i = 0; i<=4; i++)
    Number[i] = 0.0;
```

Πίνακες πολλαπλών διαστάσεων

- Ένα πίνακα με δύο διαστάσεις μπορούμε να τον φανταστούμε σαν ένα πλαίσιο αποτελούμενο από γραμμές και στήλες. Για να προσδιοριστεί η θέση κάθε στοιχείου απαιτούνται δύο αριθμοί, ένας για τη γραμμή και ένας για τη στήλη. Οι πίνακες δύο ή και περισσοτέρων διαστάσεων δηλώνονται, καταχωρούνται και παίρνουν αρχικές τιμές με τον ίδιο τρόπο όπως και οι πίνακες μιας διάστασης. Η εργασία αυτή γίνεται με τη χρήση ανάλογου αριθμού ένθετων for ή while. Πχ

  - Ορισμός πίνακα → int [][]m;
  - Καταχώρηση πίνακα → m = new [4][5];

- Εισαγωγή στοιχείων στο πίνακα :

```
For (int row = 0; row < 4; row++)
  {
    For (int col = 0; col<5; col++)
      {
        m [row][col] = row + col;
      }
  }
```

- Εκτύπωση στοιχείων πίνακα :

```
For (int row = 0; row < 4; row++)
  {
    For (int col = 0; col<5; col++)
      System.out.print (m [row][col] + " ");
    System.out.println ();
  }
```

Ταξινόμηση

- Έμφαση θα πρέπει να δοθεί στη ταξινόμηση των δεδομένων του πίνακα. Με τον όρο ταξινόμηση εννοούμε τη διαδικασία με την οποία τοποθετούμε μία ομάδα από δεδομένα σε αύξουσα ή φθίνουσα σειρά. Υπάρχουν διάφορες μέθοδοι ταξινόμησης *(sorting)*. Να αναφερθούν οι δύο πιο απλές, αλλά συγχρόνως και οι πιο αργές που είναι :
  - *Ταξινόμηση με επιλογή* και
  - *Ταξινόμηση με τη μέθοδο της φυσαλίδας.*

ΕΝΟΤΗΤΑ 6

Μέθοδοι

Βασικές έννοιες

- Τι είναι οι μέθοδοι.
- Πώς δηλώνουμε μια μέθοδο.
- Πώς γίνεται η επιστροφή της τιμής μιας μεθόδου.
- Πώς ο χρήστης καλεί μία μέθοδο.
- Τι είναι η αναδρομική μέθοδος.

Σχέδιο μαθήματος.

Δήλωση – επιστροφή τιμής – κάλεσμα μεθόδου

- Ο καλύτερος τρόπος για να κατασκευάσει κάποιος ένα πρόγραμμα, είναι να το φτιάξει χρησιμοποιώντας μικρά κομμάτια κώδικα. Στη Java υπάρχουν δύο τρόποι για να το κατασκευάσει κανείς. Ο ένας είναι οι *μέθοδοι (methods)* , που θα εξηγηθούν στο σημείο αυτό, και ο δεύτερος τρόπος είναι οι *κλάσεις(classes)* , που θα εξηγηθούν αργότερα. Με τον όρο *μέθοδο (method)* εννοούμε ένα σύνολο από δηλώσεις και εντολές οι οποίες ομαδοποιούνται ώστε να αποτελέσουν ένα ανεξάρτητο μίνι πρόγραμμα. Μια μέθοδος *δηλώνεται* με το ακόλουθο τρόπο :

```
Τιμή_επιστροφής   ονομα_μεθόδου (λίστα παραμέτρων)
  {
       δηλώσεις και εντολές
  }
```

Το όνομα της μεθόδου πρέπει να είναι οποιοδήποτε έγκυρο όνομα. Η τιμή_επιστροφής είναι ο τύπος δεδομένου που επιστρέφει η μέθοδος σε αυτόν που την καλεί. Εάν η μέθοδος δεν επιστρέφει καμία τιμή τότε στη θέση της τιμής επιστροφής πρέπει να μπει η λέξη *void*. Οι δηλώσεις και οι εντολές οι οποίες ανήκουν στη μέθοδο πρέπει να περικλείονται σε άγκιστρα.

Για να *καλέσει* τη μέθοδο ο χρήστης, αρκεί να καλέσει το όνομα της στο σημείο που τη χρειάζεται, τοποθετώντας τα κατάλληλα ορίσματα. Μόλις η μέθοδος τελειώσει, η εκτέλεση του προγράμματος συνεχίζεται από το σημείο ακριβώς που έγινε η κλήση της μεθόδου. Αν η μέθοδος δεν επιστρέφει αποτέλεσμα, η επιστροφή στο κυρίως πρόγραμμα γίνεται μετά το δεξί άγκιστρο ή με τη κλήση της εντολής *return*. Αν η μέθοδος επιστρέφει αποτέλεσμα, η επιστροφή στο κυρίως πρόγραμμα γίνεται με την εντολή *return έκφραση*.

Μία μέθοδος, μπορεί από το εσωτερικό της να καλεί και άλλες μεθόδους.

Αναδρομική μέθοδος

- Ειδική περίπτωση μεθόδου είναι η αναδρομική μέθοδος. Αναδρομική (recursive) λέγεται η μέθοδος η οποία καλεί τον εαυτό της. Παράδειγμα χρήσης ακολουθεί.

ΕΝΟΤΗΤΑ 7

Αλφαριθμητικά

Βασικές έννοιες

- Τι είναι τα αλφαριθμητικά
- Πράξεις με αλφαριθμητικά
- Υποαλφαριθμητικά.
- Προσπέλαση μεμονωμένων χαρακτήρων ενός αλφαριθμητικού.
- Εύρεση του μήκους του.
- Αντικατάσταση χαρακτήρων σε αλφαριθμητικό.
- Εντοπισμός χαρακτήρων σε αλφαριθμητικό.
- Μετατροπή άλλων τύπων δεδομένων σε αλφαριθμητικά(η μέθοδος valueOf).
- Μετατροπή αλφαριθμητικών σε άλλους τύπους δεδομένων.

Σχέδιο μαθήματος.

Τι είναι τα αλφαριθμητικά

- Με τον όρο *αλφαριθμητικά* (*string*) εννοούμε μία σειρά από χαρακτήρες πχ ένα όνομα. Κάθε αλφαριθμητικό περικλείεται μέσα σε διπλά εισαγωγικά. Υπάρχουν δύο κλάσεις διαχείρισης αλφαριθμητικών : η *String* και η *String Buffer*.
  Μία μεταβλητή αλφαριθμητικού τύπου είναι ένα αντικείμενο της κλάσης String. Δηλώνουμε και δίνουμε αρχική τιμή σε μεταβλητή ως εξής : String mystring = “hello”;

  Ομοίως μεταβλητές που κρατούν πίνακες όπως :
  String car = new String [5];

Πράξεις με αλφαριθμητικά

Για *συνένωση* δύο ή περισσοτέρων αλφαριθμητικών χρησιμοποιούμε τον τελεστή +. Μπορούμε να συνενώσουμε αλφαριθμητικές μεταβλητές, με αλφαριθμητικά δεδομένα τοποθετημένα σε εισαγωγικά πχ :
System.out.println (first + " , " + second);

Για *έλεγχο ισότητας* δύο αλφαριθμητικών χρησιμοποιείται η μέθοδος *equals* πχ :
If (string1.equals (string2))
Αν θέλουμε να μη λαμβάνεται κατά τη σύγκριση υπ' όψη το αν οι χαρακτήρες είναι κεφαλαίο ή πεζοί, τότε χρησιμοποιούμε τη μέθοδο *equalsIgnoreCase()* αντί για τη equals.

Η κλάση String περιέχει μεθόδους με τις οποίες μπορούμε να *μετατρέψουμε τα κεφαλαία γράμματα σε πεζά και αντίστροφα*.

Η μέθοδος *toLowerCase ()* → μετατρέπει σε πεζά γράμματα.
Η μέθοδος *to UpperCase ()* → μετατρέπει σε κεφαλαία γράμματα.
Πχ : String str1 = "Anna";
String str2 = str1.toUpperCase ();
String str3 = str2.toLowerCase ();

Υποαλφαριθμητικά

- Με τον όρο *υπααλφαριθμητικό (substring)* εννοούμε ένα αλφαριθμητικό, το οποίο βρίσκεται μέσα σε ένα αλφαριθμητικό μεγαλύτερο ή ίσο από αυτό. Για τη εξαγωγή ενός αλφαριθμητικού από ένα άλλο χρησιμοποιούμε τη μέθοδο substring().
  Η σύνταξη της είναι : String substring (int αρχή , int τέλος)
  Το όρισμα 'αρχή' προσδιορίζει το δείκτη για το πρώτο χαρακτήρα του αλφαριθμητικού, ενώ ο ακέραιος 'τέλος' προσδιορίζει το δείκτη μιας θέσης μετά τον τελευταίο χαρακτήρα του υποαλφαριθμητικού μέσα στο αλφαριθμητικό. Πχ

  String str1 = "Today is a nice day";
  String str2 = str1.substring (3, 5);
  System.out.println (str2);

Προσπέλαση μεμονωμένων χαρακτήρων ενός αλφαριθμητικού

- Πολλές φορές χρειαζόμαστε να πάρουμε μεμονωμένους χαρακτήρες από ένα αλφαριθμητικό. Χρησιμοποιούμε μια μεταβλητή τύπου int για να προσδιορίσουμε τη θέση του χαρακτήρα σε συνδυασμό με τη μέθοδο *charAt()* πχ :

```
String str1 = “Today is a nice day”;
System.out.println (“character at position 4is : ” + str1.charAt (4));
```

Εύρεση του μήκους του

- Η μέθοδος length(), μία χρήσιμη μέθοδος, για να μπορέσουμε να βρούμε το μήκος του αλφαριθμητικού. Πχ ¨

```
String str = “I like football”;
System.out.println (“the string contains = ” + str.length() +
“characters”);
```

Αντικατάσταση χαρακτήρων σε αλφαριθμητικό

- Η μέθοδος *replace()* χρησιμοποιείται για να αντικατασταθούν κάποιοι χαρακτήρες ενός αλφαριθμητικού με άλλους. Η μέθοδος replace() δέχεται δύο παραμέτρους. Ο πρώτος είναι ο χαρακτήρας που πρόκειται να αντικατασταθεί, και ο δεύτερος ο χαρακτήρας που θα πάρει τη θέση του. Να σημειωθεί ότι αντικαθίστανται όλοι οι χαρακτήρες που είναι ίδιοι με τον χαρακτήρα προς αντικατάσταση. Πχ :

```
String str1 = “Today is a nice day”;
String str2 = str1.replace(‘a’, ‘o’);
```

Εντοπισμός χαρακτήρων σε αλφαριθμητικό

- Οι μέθοδοι *indexOf() και lastIndexOf()* είναι δύο μέθοδοι που χρησιμοποιούνται για τον εντοπισμό χαρακτήρων μέσα σε ένα αλφαριθμητικό.

  indexOf (int ch) → Επιστρέφει τη τιμή του δείκτη για τη πρώτη θέση του χαρακτήρα ch μέσα στο αλφαριθμητικό.

  indexOf (int ch, int index) → Όπως η παραπάνω, με τη διαφορά ότι η ανίχνευση αρχίζει από τη θέση που δείχνει το index.

  indexOf (String str) → Επιστρέφει τη θέση του δείκτη όπου συναντάται για πρώτη φορά το str μέσα στο αλφαριθμητικό.

  indexOf (String str, int index) → Όπως η παραπάνω, με τη διαφορά ότι η ανίχνευση αρχίζει από τη θέση που δείχνει το index.

  Η μέθοδος indexOf() θα ψάξει στο αλφαριθμητικό από αριστερά προς τα δεξιά, ενώ η lastIndexOf() δουλεύει και συντάσσεται όπως και η indexOf(), με τη διαφορά ότι θα ψάξει στο αλφαριθμητικό από το τέλος προς τη αρχή του. Πχ :

```
String str1 = “Today is a nice day”;
Int i=0; int j=0;
I = str1.indexOf (‘a’);
J = str1.indexOf (‘a’, i+1);
```

Μετατροπή άλλων τύπων δεδομένων σε αλφαριθμητικά

- Στη Java έχουμε τη δυνατότητα να μετατρέψουμε διάφορους τύπους δεδομένων σε αλφαριθμητικά με τη μέθοδο *valueOf()*. Πχ :

```
Char ch = ‘a’;
Int n = 10;
Double pi = 3.1415;
String str1 = String.valueOf (ch);
String str2 = String.valueOf (n);
```

String str3 = String.valueOf (pi);

Μετατροπή αλφαριθμητικών σε άλλους τύπους δεδομένων

- Η μέθοδος *parseInt()* διαβάζει το περιεχόμενο του αλφαριθμητικού, το μετατρέπει σε ακέραιο, και στη συνέχεια το καταχωρεί σε κάποια μεταβλητή. Πχ :

  String str1 = "1234";
  Int a = Integer.parseInt (str1);
  System.out.println ("str =" + str1);
  System.out.println ("a =" + a);

  Όλοι οι βασικοί τύποι δεδομένων έχουν αντίστοιχες κλάσεις συσκευαστές και είναι : Double, Float, Long, Integer, Short, Byte, Character, Boolean.

ΕΝΟΤΗΤΑ 8

Κλάση StringBuffer

Βασικές έννοιες

- Η κλάση Sting Buffer.
- Οι μέθοδοι capacity(), append(), length() της StringBuffer().
- Αντικατάσταση χαρακτήρων σε αντικείμενο της κλάσης StringBuffer.
- Μετατροπή αντικειμένου StringBuffer σε αλφαριθμητικό.
- Αντιστροφή αλφαριθμητικού (η μέθοδος reverse).

Σχέδιο μαθήματος.

Η κλάση Sting Buffer

- Τα αντικείμενα της κλάσης String δεν μπορούν να αλλάξουν. Δεν μπορούμε δηλαδή να αλλάξουμε ένα αλφαριθμητικό μέσα στο ίδιο το αντικείμενο. Για να γίνει αυτό η κλάση StringBuffer χρησιμοποιείται για αλφαριθμητικά που μεταβάλλονται.

Οι μέθοδοι capacity(), append(), length()

- Βασικές μέθοδοι της κλάσης StringBuffer είναι :
  Length () → μας δίνει το χρησιμοποιούμενο μήκος
  Capacity () → μας δίνει τη συνολική χωρητικότητα του.
  Append () → προσθέτουμε χαρακτήρες στο αντικείμενο. Πχ :

```
StringBuffer a = new StringBuffer (50);
a.append (“Today is ”);
System.out.println (a);
System.out.println (a.length ());
System.out.println (a.capacity ());
a.append (“Sunday”);
System.out.println (a);
```

Αντικατάσταση χαρακτήρων σε αντικείμενο της κλάσης StringBuffer ()

- Οι μέθοδοι *setCharAt()* και *insert()* χρησιμοποιούνται για να κάνουμε προσθήκες στο αλφαριθμητικό περιεχόμενο ενός αντικειμένου τύπου StringBuffer. Πχ :
  StringBuffer a = new StringBuffer ("Today is Sunday");
  a.insert (3,'a');
  System.out.println (a);
  a.setCharAt (4,'b');

Μετατροπή αντικειμένου StringBuffer() σε αλφαριθμητικό

- Αντίστροφα, ενδέχεται ένα αντικείμενο της κλάσης StringBuffer να θέλουμε να μετατραπεί σε αλφαριθμητικό, για αυτό χρησιμοποιούμε τη μέθοδο *toString()*. *Πχ*

  StringBuffer a = new StringBuffer ("Today is Sunday");
  String str = a.toString ();

Αντιστροφή αλφαριθμητικού

- Η μέθοδος *reverse()* της κλάσης StringBuffer, χρησιμοποιείται για να γίνει αντιστροφή ενός αλφαριθμητικού τύπου StringBuffer. Πχ :

  StringBuffer a = new StringBuffer ("Today is Sunday");
  a.reverse ();
  System.out.println (a);

ΕΝΟΤΗΤΑ 9

Κλάσεις και αντικείμενα

Βασικές έννοιες

- Τι είναι οι κλάσεις και πως ορίζονται.
- Τι είναι οι κατασκευαστές (constructor).
- Πώς δημιουργούνται τα αντικείμενα.
- Πολλαπλοί κατασκευαστές (constructors).
- Τι είναι τα πακέτα – πως δημιουργούνται.
- Ενθυλάκωση.
- Πολυμορφισμός.
- Κληρονομικότητα.

Σχέδιο μαθήματος.

Τι είναι οι κλάσεις και πως ορίζονται

- Η κλάση είναι μία γενική φόρμα για την κατασκευή αντικειμένων που έχουν παρόμοια χαρακτηριστικά. Αφού ορίσουμε τη κλάση , μετά μπορούμε να χρησιμοποιήσουμε τον ορισμό της για να δημιουργήσουμε ***αντικείμενα.*** Κάθε αντικείμενο λέγεται και *στιγμιότυπο* της κλάσης. Στη Java υπάρχουν έτοιμες κλάσεις, αλλά μπορεί ο χρήστης να δημιουργήσει και τις δικές του κλάσεις. Δύο είναι τα στοιχεία που περιλαμβάνονται στο ορισμό μιας κλάσης.

  A) **Οι *μεταβλητές (variables)***, οι οποίες διαφοροποιούν το ένα αντικείμενο της κλάσης από το άλλο και

  B) **Οι *μέθοδοι (methods),***οι οποίες καθορίζουν τι μπορούμε να κάνουμε στα αντικείμενα μιας κλάσης.

  Για να *ορίσουμε μια κλάση*, χρησιμοποιούμε τη λέξη class ακολουθούμενη από το όνομα που επιλέγουμε για τη κλάση. Πχ :

```
Class circle
  {
    }
```

Την αποθηκεύουμε με το όνομα circle.java και όταν τη μεταγλωττίσουμε δημιουργείται το αρχείο circle.class. Ακολουθεί ένα παράδειγμα για να κατανοήσουμε το ορισμό της κλάσης :

```
Class circle
 {
   Static double pi = 3.14;
   Static int count = 0;
   Double radius;
   Double x;
   Double y;
 }
```

Στο σημείο αυτό οι μαθητές θα πρέπει να κατανοήσουν τις ακόλουθες έννοιες που αφορούν τις μεθόδους και τις μεταβλητές που ορίζονται μέσα σε μια κλάση.

Οι μεταβλητές της κλάσης ανήκουν σε δύο κατηγορίες :

A) Οι μεταβλητές pi και count είναι μεταβλητές κλάσης και δηλώνονται με τη λέξη ***static*** . Μια *μεταβλητή κλάσης* υπάρχει έστω και δεν έχει δημιουργηθεί ακόμη κανένα αντικείμενο. Όταν το αντικείμενο δημιουργηθεί, τότε περιέχει τη τιμή του pi και της count. Εάν οι τιμές αυτές αλλάξουν, τότε οι νέες τιμές τοποθετούνται σε όλα τα αντικείμενα της κλάσης.

B) Οι μεταβλητές radius, x, y λέγονται *μεταβλητές στιγμιότυπου*. Σε κάθε αντικείμενο της κλάσης που δημιουργείται, οι μεταβλητές αυτές υπάρχουν, αλλά έχουν διαφορετική τιμή από αντικείμενο σε αντικείμενο.

Ανάλογα με τις μεταβλητές, υπάρχουν και δύο κατηγορίες μεθόδων, οι *μέθοδοι κλάσης* και οι *μέθοδοι στιγμιότυπου*.

A) Οι *μέθοδοι κλάσης* δηλώνονται με τη λέξη static και μπορεί να εκτελεστούν ακόμη και αν δεν έχει δημιουργηθεί κανένα αντικείμενο της κλάσης.
B) Οι *μέθοδοι στιγμιότυπου* για να εκτελεστούν πρέπει να υπάρχουν αντικείμενα. Αν δεν υπάρχει αντικείμενο, δεν έχει νόημα να εκτελεστεί μία τέτοια μέθοδος.

Παρατήρηση : Λόγω της σπουδαιότητας της συγκεκριμένης ενότητας, να γίνουν πολλά παραδείγματα από το τετράδιο μαθητή,

ώστε να κατανοήσουν καλύτερα τις κλάσεις και τη κληρονομικότητα.

*Η **μεταβλητή this*** : Κάθε μέθοδος στιγμιότυπου διαθέτει μία αναφορά με το όνομα this, η οποία αναφέρεται στο τρέχον αντικείμενο, για το οποίο έχει κληθεί η μέθοδος. Κάθε φορά στην μνήμη υπάρχει μόνο ένα αντίγραφο της μεθόδου. Χρησιμοποιώντας λοιπόν την λέξη this, μπορούμε να κάνουμε την μέθοδο να εργάζεται κάθε φορά για διαφορετικό αντικείμενο. Επειδή το this αναφέρεται στο τρέχον αντικείμενο της κλάσης, χρησιμοποιείται μόνο για τις μεθόδους στιγμιότυπου και όχι για τις μεθόδους κλάσης.

Τι είναι οι κατασκευαστές (constructor)

- *Οι **κατασκευαστές (constructors)*** ορίζονται μέσα στην κλάση. Ο κατασκευαστής έχει το ίδιο όνομα με την κλάση και χρησιμοποιείται για να δώσει αρχικές τιμές σε ένα νέο αντικείμενο. Τα βασικά του χαρακτηριστικά που τον *διαφοροποιούν* από τις άλλες μεθόδους είναι :

Α) Δεν επιστρέφει ποτέ τιμή και δεν βάζουμε μπροστά του τύπο επιστροφής, αλλά ούτε και την λέξη void.
Β) Έχει το ίδιο όνομα με την κλάση.

Ο κατασκευαστής μπορεί να έχει όσες παραμέτρους θέλουμε, ενδεχομένως όμως και καμία. Ομοίως μπορούμε να έχουμε περισσότερους από έναν κατασκευαστές για την ίδια κλάση, καθένας από τους οποίους να έχει διαφορετική λίστα παραμέτρων. Αυτό προκύπτει από την ανάγκη να δημιουργούμε αντικείμενα με διαφορετικά χαρακτηριστικά. Οι διαφορετικοί κατασκευαστές έχουν απαραίτητα το ίδιο όνομα, αλλά διαφορετική λίστα παραμέτρων.

Πώς δημιουργούνται τα αντικείμενα

- Ακολουθεί ένα παράδειγμα δημιουργίας ενός κατασκευαστή και δημιουργίας ενός αντικειμένου.

```
Class Circle
{
```

```
   static double Pi=3.14;
   static int count=0;
   double radius;
   double  x;
   double  y;

  //constructor
 Circle (double r1, double x1, double y1)
   {
    Radius=r1;
    x  =  x1;
    y  =  y1;
    ++ count;
}
}
```

Η δήλωση της μεταβλητής που θα κρατήσει το αντικείμενο είναι → circle troxos; Δεν έχει δημιουργηθεί ακόμα το αντικείμενο. Για την δημιουργία του πρέπει να γράψουμε :

Troxos = new circle (40.0, 5.0, 4.0);

Με την δήλωση αυτή δώσαμε τιμές στο r1, x1, y1, και φτιάξαμε ένα αντικείμενο troxos ακτίνα 40.0 και συντεταγμένες (5.0, 4.0) στο σύστημα αξόνων.

Το ερώτημα που θα πρέπει να τεθεί στους μαθητές σε αυτό το σημείο είναι το εξής :
Δημιουργήσαμε μία κλάση. Χρησιμοποιώντας το κατασκευαστή (constructor) κατασκευάζουμε ένα ή πολλά αντικείμενα. Πώς θα χρησιμοποιηθεί η κλάση που δημιουργήσαμε; Σε ποιο σημείο στο πρόγραμμα μας θα κατασκευάσουμε τα αντικείμενα μας για να τα επεξεργαστούμε;
Λόγω της μεγάλης σημασίας που έχει η απάντηση στα παραπάνω ερωτήματα, η απάντηση θα δοθεί στο τετράδιο του μαθητή, όπου με παραδείγματα οι μαθητές θα καταλάβουν τη διαδικασία.

Πολλαπλοί κατασκευαστές (constructors)

Σε μία κλάση μπορούμε να ορίσουμε περισσότερους από ένα κατασκευαστές (constructors). Αυτό οφείλεται στη ανάγκη να δημιουργούμε αντικείμενα με διαφορετικά χαρακτηριστικά. Οι

διαφορετικοί κατασκευαστές έχουν υποχρεωτικά το ίδιο όνομα, αλλά διαφορετική λίστα παραμέτρων.

Τι είναι τα πακέτα – πως δημιουργούνται

- *Ένα πακέτο (package)* είναι μία συλλογή από κλάσεις, οι οποίες μπορούν να προσθέσουν δυνατότητες στην java. Ο κώδικας που βρίσκεται σε ένα πακέτο μπορεί να ξαναχρησιμοποιηθεί σε διαφορετικά προγράμματα. Όλες οι κλάσεις που έχουν χρησιμοποιηθεί μέχρι στιγμής, βρίσκονται στο πακέτο java.lang. Για να χρησιμοποιήσουμε μία μέθοδο ή μία κλάση ενός πακέτου, υπάρχουν δύο τρόποι :

  A) Ο πρώτος τρόπος είναι να χρησιμοποιήσουμε το *πλήρες όνομα του πακέτου*, κατά τον ορισμό του αντικειμένου της κλάσης. Π.χ.
  Java.util.Date thisDate = new java.util.Date ();

  Εδώ χρησιμοποιήσαμε την κλάση Date που βρίσκεται στο πακέτο util και δημιουργήσαμε το αντικείμενο της κλάσης thisDate.

  B) Ο Δεύτερος τρόπος είναι να εισάγουμε την κλάση στην αρχή του προγράμματος, με την λέξη ***import*** και κατόπιν να χρησιμοποιήσουμε την κλάση μέσα στο πρόγραμμα, όπως φαίνεται παρακάτω :

  ```
  Import java.util.Date;
  Date thisDate = new Date ();
  ```

  Πολλές φορές όμως, αντί να εισάγουμε μία κλάση είναι προτιμότερο να εισάγουμε ολόκληρο το πακέτο, χρησιμοποιώντας ένα αστερίσκο (*), π.χ. **import java.util.*;**

  Τα πιο σημαντικά πακέτα που προσφέρονται με την γλώσσα της java είναι τα εξής :

  Java.lang → Περιέχει τις βασικές κλάσεις της java. Δεν χρειάζεται να κάνεις import. Μία σημαντική κλάση του πακέτου είναι η κλάση Math.
  Java.awt → Οι κλάσεις υποστηρίζουν το γραφικό περιβάλλον GUI.

Java.awt.event	→	Χειρισμός γεγονότων (event handling)
Java.io	→	Κλάσεις για είσοδο – έξοδο δεδομένων
Java.applet	→	Περιέχει κλάσεις με τις οποίες μπορούμε να γράψουμε προγράμματα που να ενσωματώνονται σε ιστοσελίδες.
Java.security	→	Κλάσεις για αύξηση της ασφάλειας, χρησιμοποιώντας μεθόδους κρυπτογράφησης.
Java.awt.image	→	Κλάσεις για επεξεργασία εικόνας
Javax.swing	→	Περιέχει κλάσεις και διασυνδέσεις για την υλοποίηση του νέου περιβάλλοντος επικοινωνίας με τον χρήστη.
Java.util	→	Κλάσεις για χειρισμό ημερομηνιών, καθώς και χειρισμό αλφαριθμητικών.

Πως μπορούμε να πακετάρουμε τις δικές μας κλάσεις ?

Εάν θέλουμε οι κλάσεις και οι μέθοδοι του πακέτου να είναι *προσπελάσιμες έξω από αυτό*, θα πρέπει να τις δηλώνουμε χρησιμοποιώντας την λέξη ***public***.

Για να *προσθέσουμε μία κλάση σε ένα πακέτο* πριν ακόμα την ορίσουμε, γράφουμε την δήλωση :

Package όνομα-πακέτου;

Έτσι εάν είχαμε μία κλάση circle και θέλαμε να την τοποθετήσουμε σε ένα πακέτο με όνομα history θα γράφαμε:

```
package history;

public class circle
{
εντολές
}
```

Παράδειγμα δημιουργίας πακέτου θα δοθεί στο τετράδιο μαθητή στην αντίστοιχη ενότητα.

Τα βασικά χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού

- Τα βασικά χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού υλοποιούν τρεις αρχές :

  Ενθυλάκωση (encapsulation)
  Πολυμορφισμό (polymorphism)
  Κληρονομικότητα (inheritance)

Ενθυλάκωση

- *Ενθυλάκωση (encapsulation)*. Η ενθυλάκωση παρουσιάζει την εξωτερική όψη ενός αντικειμένου και αποκρύπτει ότι υπάρχει μέσα του. Αυτό επιτρέπει στον προγραμματιστή να τροποποιεί και να βελτιώνει τους αλγορίθμους οι οποίοι υπάρχουν σε μία κλάση, χωρίς να ανησυχεί μήπως προκαλέσει απρόβλεπτα αποτελέσματα. Τα δεδομένα και οι μέθοδοι μίας κλάσης, μπορούν να δηλωθούν ως **public**, ως **private** και ως **protected**.

  Public → Οι μεταβλητές, οι μέθοδοι και οι κλάσεις που δηλώνονται ως public είναι δυνατόν να προσπελαστούν από οποιαδήποτε άλλη κλάση όπου και αν βρίσκεται αυτή.

  Private → Όσες μεταβλητές και μέθοδοι χαρακτηρίζονται ως private μπορούν να προσπελαστούν και να χρησιμοποιηθούν μόνο από το ίδιο το αντικείμενο. Δεν μπορούν να προσπελαστούν από πουθενά έξω από αυτό.

  Protected → Οι μεταβλητές και οι μέθοδοι που έχουν δηλωθεί ως protected σε μία κλάση μπορούν να προσπελαστούν από όλες τις κλάσεις που βρίσκονται στο ίδιο πακέτο ή από τις υποκλάσεις (subclasses) αυτής της κλάσης που θα δούμε αργότερα.

Ο παρακάτω πίνακας καθορίζει πλήρως την ενθυλάκωση :

Δυνατότητα προσπέλασης	Public	Protected	Χωρίς προσδιοριστή	private
Από την ίδια κλάση	√	√	√	√
Από τις κλάσεις του ίδιου πακέτου	√	√	√	-
Από οποιοδήποτε κλάση εκτός του πακέτου όπου ανήκει η κλάση	√	-	-	-
Από μία υποκλάση του ίδιου πακέτου	√	√	√	-
Από μία υποκλάση έξω από το πακέτο όπου ανήκει η κλάση	√	√	-	-

Πολυμορφισμός

- *Πολυμορφισμός (*polymorphism*)*. Πολυμορφισμό έχουμε στην περίπτωση που η ίδια η μέθοδος ή ο ίδιος τελεστής χρησιμοποιείται με διαφορετικούς τύπους δεδομένων. Παράδειγμα Πολυμορφισμού είναι ο τελεστής της πρόσθεσης (+). Μπορεί να χρησιμοποιηθεί τόσο για την πρόσθεση δύο αριθμών όσο και για την συνένωση δύο αλφαριθμητικών. Ομοίως ο Πολυμορφισμός χρησιμοποιείται και στις κλάσεις. Παράδειγμα Πολυμορφισμού κλάσεων είναι όταν έχουμε δύο κατασκευαστές (constructor) εκ των οποίων ο ένας έχει τρεις παραμέτρους και ο άλλος ένα. Η κατάσταση σύμφωνα με την οποία μπορούμε να έχουμε περισσότερους από ένα κατασκευαστές με το ίδιο όνομα, αλλά τον καθένα με τις δικές του παραμέτρους, ονομάζεται *υπερφόρτωση κατασκευαστή*. Το φαινόμενο της υπερφόρτωσης ισχύει και στις μεθόδους. Μπορούμε δηλαδή να δημιουργήσουμε μεθόδους με το ίδιο όνομα, αλλά με διαφορετικό ορισμό και άλλες παραμέτρους. Η java έχει την δυνατότητα να τις ξεχωρίζει και κάθε φορά να εκτελεί την μέθοδο που πρέπει, από τον αριθμό και τον τύπο των παραμέτρων τους.

Κληρονομικότητα

- *Κληρονομικότητα (inheritance)*. Η Κληρονομικότητα είναι μία από τις βασικές ιδέες του αντικειμενοστραφούς

προγραμματισμού. Σύμφωνα με την ιδέα αυτή, ένα αντικείμενο μπορεί να κληρονομεί τα δεδομένα και τις μεθόδους ενός άλλου αντικειμένου, καθώς επίσης να τους επιφέρει τροποποιήσεις. Στην java κάθε κλάση έχει μία *υπερκλάση (**super class**)*, η οποία βρίσκεται ιεραρχικά πάνω από αυτή. Μπορεί επίσης να έχει μία ή περισσότερες *υποκλάσεις (**subclasses**),* οι οποίες βρίσκονται σε κατώτερο επίπεδο ιεραρχίας. Οι υποκλάσεις κληρονομούν αυτόματα όλες τις μεταβλητές και τις μεθόδους από τις υπερκλάσεις τους. Στην κορυφή όλων των κλάσεων βρίσκεται η κλάση *Object*. Όλες οι κλάσεις κληρονομούν από αυτή την υπερκλάση. Έτσι όταν ορίζουμε μία κλάση για την οποία δεν υπάρχει ορισμός υπερκλάσης, η java θεωρεί αυτόματα ότι η οριζόμενη κλάση είναι υποκλάση της Object. Για να δηλώσουμε ότι μία κλάση είναι υποκλάση κάποιας άλλης, χρησιμοποιούμε την λέξη ***extends***. π.χ.

```
Class Car extends motorVehicle
{
…..
 }
```

Σύμφωνα με την παραπάνω δήλωση η κλάση Car εκτός από τις ιδιότητες οι οποίες ορίζονται μέσα στο εσωτερικό της, κληρονομεί (άρα μπορεί να χρησιμοποιήσει) και τις ιδιότητες της κλάσης motorVehicle, η οποία είναι υπερκλάση. Με την ίδια λογική η κληρονομικότητα μπορεί να εκτείνεται και σε περισσότερα από ένα επίπεδα (multilevel inheritance).

Η δεσμευμένη λέξη της java, ***super,*** χρησιμοποιείται ως εξής:

Super → Χρησιμοποιείται για την κλήση των μελών της υπερκλάσης μίας κλάσης.

Super () → Καλεί τον κατασκευαστή της υπερκλάσης.

Super.f() → Καλεί την συνάρτηση f (), η οποία έχει οριστεί στην υπερκλάση.

Public final class MyClass → Σε περίπτωση κατά την οποία θέλουμε μία κλάση να μην δημιουργεί υποκλάσεις, τότε δηλώνουμε την κλάση αυτή ως τελική (final).

Public final double area () → Η δήλωση αυτή γίνεται για να μην είναι δυνατόν η μέθοδος αυτή να ακυρωθεί.

Public final double pi=3.14; → Η δήλωση αυτή χαρακτηρίζει *και* μία μεταβλητή καθιστώντας την με αυτόν τον τρόπο σταθερά. Οποιαδήποτε προσπάθεια τροποποίησης της τιμής του pi θα δημιουργήσει ένα σφάλμα μεταγλώττισης.

Public abstract getname () → Μία μέθοδος αφηρημένου τύπου σε μία κλάση απλώς ορίζεται χωρίς να υλοποιείται.

Κασταμονής 99α & Μακρυγιάννη, 142 35 Ν. Ιωνία
Τηλ. 210.2719100, Fax : 210-2718133
www.sdc.gr