Εισαγωγή στη γλώσσα προγραμματισμού Java

Σημειώσεις Σεμιναρίου

Περιεχόμενα

Ενότητα 1 – Βασικά Στοιχεία της Γλώσσας

1.1 Εισαγωγή

Καλώς ήρθατε στο μάθημα ηλεκτρονικής διδασκαλίας «Εισαγωγή στη Γλώσσα Προγραμματισμού Java» του Εργαστηρίου Τεχνολογίας Πολυμέσων του Ε.Μ.Π. Σκοπός του μαθήματος είναι η εκμάθηση των βασικών δυνατοτήτων της Java στους εκπαιδευόμενους μέσα από μία σειρά 10 εκπαιδευτικών ενοτήτων οι οποίες καλύπτουν όλα αυτά που θα πρέπει να γνωρίζει κάποιος για να ξεκινήσει τον προγραμματισμό εφαρμογών σε Java.

Η οργάνωση της διδασκαλίας του μαθήματος έχει γίνει με τη θεώρηση πως οι εκπαιδευόμενοι δεν έχουν πρότερη επαφή με τη Java ή κάποια άλλη γλώσσα προγραμματισμού και άρα, η διδασκαλία ξεκινάει από το μηδέν. Λαμβάνοντας υπ' όψιν πως ο κάθε εκπαιδευόμενος έχει διαφορετικές γνώσεις και υπόβαθρο είναι προφανές πως θα υπάρξουν περιπτώσεις όπου κάποιος εκπαιδευόμενος με εμπειρία στη Java ή σε κάποια άλλη γλώσσα προγραμματισμού ίσως αισθανθεί πως το επίπεδο του μαθήματος είναι κατώτερο του αναμενομένου. Στον αντίποδα, κάποιος εκπαιδευόμενος με ελάχιστη ή καθόλου εμπειρία σε γλώσσες προγραμματισμού ίσως αισθανθεί να βομβαρδίζεται από τον όγκο των πληροφοριών που περιέχονται στις ενότητες του μαθήματος και πιθανώς να απογοητευτεί. Στη μεν πρώτη περίπτωση συστήνουμε υπομονή μέχρι την ενότητα 4 (αντικειμενοστρεφής προγραμματισμός) ενώ στη δεύτερη συστήνουμε επιμονή, προσπάθεια και καθημερινή ενασχόληση.

Ιδιαίτερη έμφαση δίνεται στη σωστή εκμάθηση της γλώσσας και για το λόγο αυτό τόσο ο κώδικας που θα συναντήσετε στις σημειώσεις και τις παρουσιάσεις ακολουθεί πιστά τους κανόνες σωστής πρακτικής που έχουν θεσπιστεί από την προγραμματιστική κοινότητα για τη δημιουργία «σωστών» και ευανάγνωστων προγραμμάτων. Οι κανόνες αυτοί θα σας παρουσιαστούν αναλυτικά κατά τη διάρκεια του μαθήματος και σας προτρέπουμε να τους υιοθετήσετε ώστε να μάθετε να γράφετε σωστά δομημένο κώδικα.

Τέλος, από το Χειμερινό διδακτικό εξάμηνο του 2009 δίνεται η δυνατότητα στους εκπαιδευόμενους μετά την ολοκλήρωση της παρακολούθησης του σεμιναρίου και για όσους το επιθυμούν, να κάνουν ένα ακόμα βήμα προς την κατεύθυνση της επίσημης πιστοποίησης της Sun Microsystems για τη Java, συμμετέχοντας και περνώντας επιτυχώς το αντίστοιχο τεστ (περισσότερες λεπτομέρειες σχετικά με το τεστ της Java θα βρείτε στην υποενότητα 1.4). Έτσι λοιπόν, στη διδακτέα ύλη έχουν ενταχθεί και καλύπτονται πλήρως όλα τα απαιτούμενα συστατικά που θα πρέπει να γνωρίζει ο υποψήφιος, όπως αυτά έχουν καθοριστεί από τη Sun Microsystems.

1.2 Ορολογία

Ένα μεγάλο και σημαντικό κεφάλαιο στην εκμάθηση μιας γλώσσας προγραμματισμού είναι η γνώση της ορολογίας. Η Java δεν αποτελεί εξαίρεση και μάλιστα περιέχει έναν αρκετά μεγάλο αριθμό όρων τους οποίους θα διδαχτείτε στην πορεία του μαθήματος. Οι εξελίξεις στο χώρο της πληροφορικής έρχονται από το εξωτερικό και συνεπώς η ορολογία είναι εκφρασμένη στην Αγγλική γλώσσα. Από την άλλη πλευρά, οι μεταφραστές των ξενόγλωσσων βιβλίων στα Ελληνικά συνήθως μεταφράζουν τον

Αγγλικό όρο αντιστοιχίζοντάς τον με κάποια Ελληνική λέξη, που δυστυχώς στις περισσότερες περιπτώσεις δεν είναι η απολύτως σωστή. Υπάρχουν αμέτρητα τέτοια παραδείγματα, ένα από αυτά τυγχάνει να είναι και το βιβλίο που προμηθευτήκατε, το οποίο ανεξάρτητα από το συγκεκριμένο θέμα θεωρείται κορυφαίο στο είδος του.
Η πρακτική που θα ακολουθηθεί στη διδασκαλία του μαθήματος είναι να σας παρουσιάζεται η ορολογία στα Αγγλικά αλλά και στα Ελληνικά, χρησιμοποιώντας όμως τον σωστό όρο ακόμη κι αν έρχεται σε αντίθεση με αυτόν που μπορεί να συναντήσετε στο βιβλίο σας. Η προσωπική μου συμβουλή σε όσους επιθυμούν να ασχοληθούν επαγγελματικά με τον προγραμματισμό είναι να δώσουν έμφαση στην εκμάθηση των Αγγλικών όρων. Όσοι από εσάς εργάζονται στο χώρο της πληροφορικής θα γνωρίζουν πως η συντριπτική πλειονότητα των Μηχανικών Λογισμικού αλλά και των λοιπών ειδικοτήτων χρησιμοποιούν στις μεταξύ τους συνομιλίες την Αγγλική ορολογία, πράγμα απολύτως φυσιολογικό.

1.3 Συμβάσεις

Στην παρούσα υποενότητα σας παρουσιάζονται οι συμβάσεις που χρησιμοποιούνται στις σημειώσεις αυτές για διευκόλυνσή σας.
Κανόνας σωστής πρακτικής: Χρησιμοποιώντας τη συγκεκριμένη σύμβαση θα σας παρουσιάζονται οι κανόνες σωστής πρακτική, τους οποίους, όπως ήδη αναφέρθηκε, κρίνεται σκόπιμο να υιοθετήσετε.
Συμβουλή για το διαγώνισμα της Sun: Χρησιμοποιώντας τη συγκεκριμένη σύμβαση θα σας δίνονται συμβουλές σχετικές με το διαγώνισμα πιστοποίησης της Sun Microsystems. Οι συμβουλές αυτές είναι χρήσιμες μόνο σε όσους σκοπεύουν να λάβουν μέρος στη συγκεκριμένη εξέταση.
Σημείωση: Με τη σύμβαση αυτή θα σας δίνονται διευκρινήσεις επάνω σε συγκεκριμένα θέματα.

1.4 Η Εξέταση Πιστοποίησης Της Sun Microsystems

Η Sun Microsystems προσφέρει τη δυνατότητα επίσημης πιστοποίησης γνώσεων στη γλώσσα Java, όπως άλλωστε και οι περισσότερες εταιρείες που παράγουν τεχνολογία (Cisco, Microsoft κλπ). Συγκεκριμένα, παρέχονται οκτώ διαφορετικά διαγωνίσματα τα οποία οδηγούν στην αντίστοιχη πιστοποίηση και διαφοροποιούνται ως προς το επίπεδο δυσκολίας αλλά και το αντικείμενο εξέτασης. Αναλυτικά οι πιθανές πιστοποιήσεις που μπορεί κάποιος προγραμματιστής να κατοχυρώσει είναι οι εξής:

- **Sun Certified Java Associate (SCJA)**: Εισαγωγική πιστοποίηση, το διαγώνισμα της οποίας εξετάζει τις στοιχειώδεις γνώσεις στη γλώσσα Java. Δεν έχει μεγάλη πρακτική αξία.
- **Sun Certified Java Programmer (SCJP)**: Το αμέσως επόμενο επίπεδο πιστοποίησης, το διαγώνισμα της οποίας εξετάζει τις βασικές γνώσεις του υποψηφίου στη Java SE 6 και την ικανότητά του να χρησιμοποιεί σωστά τα εργαλεία της γλώσσας. Είναι η πιστοποίηση την οποία θα είστε σε θέση να διεκδικήσετε ολοκληρώνοντας επιτυχώς το μάθημα αυτό και έχοντας κατανοήσει το σύνολο της διδακτέας ύλης. Η συγκεκριμένη πιστοποίηση χαίρει μεγάλης αποδοχής στην αγορά εργασίας και αποτελεί σίγουρα ένα πρόσθετο εφόδιο σε όποιον την κατέχει.

- **Sun Certified Java Developer (SCJD)**: Ένα βήμα ανώτερο από την SCJP, η συγκεκριμένη πιστοποίηση απευθύνεται σε όσους επιθυμούν να έχουν επίσημη απόδειξη για την ικανότητά τους να υλοποιούν προχωρημένες εφαρμογές στη Java SE 6. Για την επίτευξη της πιστοποίησης αυτής εκτός από το ηλεκτρονικό διαγώνισμα απαιτείται και η εκπόνηση μιας εργασίας. Επίσης, ως προαπαιτούμενο, ο κάθε υποψήφιος θα πρέπει να κατέχει πιστοποίηση SCJP για την έκδοση 6 ή κάποια άλλη προγενέστερη έκδοση της Java.
- **Sun Certified Web Component Developer (SCWCD)**: Η πιστοποίηση αυτή είναι για την Enterprise έκδοση της Java (Java EE) και συνεπώς απευθύνεται σε developers διαδικτυακών εφαρμογών που χρησιμοποιούν τη συγκεκριμένη τεχνολογία. Και εδώ, η SCJP είναι προαπαιτούμενη.
- **Sun Certified Business Component Developer (SCBCD)**: Η συγκεκριμένη πιστοποίηση είναι ένα σκαλί ανώτερη από την προαναφερθείσα μιας το διαγώνισμα εξετάζει την ικανότητα του υποψηφίου στη δημιουργία ολοκληρωμένων Java EE εφαρμογών χρησιμοποιώντας την τεχνολογία των Enterprise JavaBeans. Προαπαιτούμενη η SCJP.
- **Sun Certified Developer For Java Web Services (SCDJWS)**: Πιστοποίηση για developers υπηρεσιών ιστού (web services) χρησιμοποιώντας τις τεχνολογίες της Java EE. Και για τη συγκεκριμένη πιστοποίηση η SCJP είναι προαπαιτούμενη.
- **Sun Certified Mobile Application Developer (SCMAD)**: Πιστοποίηση για developers εφαρμογών για κινητά και μικροσυσκευές χρησιμοποιώντας την τεχνολογία Java ME. Για κάθε υποψήφιο η πιστοποίηση SCJP είναι προαπαιτούμενη.
- **Sun Certified Enterprise Architect (SCEA)**: Η ανώτερη πιστοποίηση της Sun που μπορεί να πετύχει κάποιος developer και εξετάζει την ικανότητα του υποψηφίου να σχεδιάζει και να υλοποιεί πολύπλοκες επιχειρησιακές εφαρμογές κάνοντας χρήση των τεχνολογιών της Java EE. Δεν υπάρχουν προαπαιτούμενα αλλά η εξέταση είναι αρκετά δύσκολη και περιλαμβάνει τρία διαφορετικά παραδοτέα: ένα ηλεκτρονικό διαγώνισμα πολλαπλής επιλογής, μία εργασία και ένα διαγώνισμα τύπου essay.

Από τις παραπάνω πιστοποιήσεις, εκείνη που καλύπτεται από τη διδακτέα ύλη του μαθήματος και που μας ενδιαφέρει είναι η SCJP. Για την επίτευξη της συγκεκριμένης πιστοποίησης ο υποψήφιος καλείται να περάσει επιτυχώς ένα ηλεκτρονικό διαγώνισμα με ερωτήσεις πολλαπλής επιλογής.
Το διαγώνισμα λαμβάνει χώρα σε κάποιο εγκεκριμένο εξεταστικό κέντρο (VUE ή Prometric), συγκεκριμένη μέρα και ώρα που έχει συμφωνηθεί μεταξύ του υποψηφίου και των εξεταστών.
Για όσους δεν έχουν προηγούμενη πιστοποίηση SCJP σε προγενέστερη έκδοση της Java ο κωδικός του διαγωνίσματος είναι ο CX-310-065. Η διάρκεια του συγκεκριμένου διαγωνίσματος είναι 210 λεπτά, μέσα στα οποία ο υποψήφιος καλείται να απαντήσει 72 ερωτήσεις πολλαπλής επιλογής με συνήθως 5 πιθανές απαντήσεις. Το σκορ βάσης είναι το 65%, δηλαδή θα πρέπει να απαντηθούν σωστά 47 από τις 72 ερωτήσεις.
Αντίστοιχα, αν κάποιος κατέχει πιστοποίηση SCJP σε προγενέστερη έκδοση της Java, μπορεί να εξεταστεί στο διαγώνισμα με κωδικό CX-310-066, το οποίο είναι διαγώνισμα αναβάθμισης και περιλαμβάνει λιγότερες ερωτήσεις αλλά και μικρότερη χρονική διάρκεια (48 ερωτήσεις σε 150 λεπτά).
Το σκορ βάσης σε αυτήν την περίπτωση είναι το 66% δηλαδή θα πρέπει να απαντηθούν σωστά 32 από τις 48 ερωτήσεις.
Το αποτέλεσμα της εξέτασης γίνεται άμεσα γνωστό στον υποψήφιο με το πέρας της διαδικασίας. Θα πρέπει να τονιστεί σε αυτό το σημείο πως τα διαγωνίσματα διατίθενται στην Αγγλική γλώσσα. Επίσης, κατά τη διάρκεια του διαγωνίσματος όπως είναι φυσικό, ο υποψήφιος δε μπορεί να κάνει χρήση

ηλεκτρονικών βοηθημάτων (PC, κινητό κλπ) και το μόνο που μπορεί να χρησιμοποιήσει είναι κόλλες αναφοράς και στυλό που του παρέχονται από το εξεταστικό κέντρο.
Το επίπεδο του διαγωνίσματος είναι αυτό ακριβώς που πρέπει να είναι, που σημαίνει πως μόνο οι καλά προετοιμασμένοι υποψήφιοι καταφέρνουν να το περάσουν επιτυχώς. Πλημμελώς προετοιμασμένοι υποψήφιοι δεν έχουν καμία τύχη και συνεπώς αν κάποιος νιώθει πως δεν έχει προετοιμαστεί κατάλληλα θα ήταν σοφό εκ μέρους του να καλέσει το εξεταστικό κέντρο λίγες ημέρες πριν την εξέταση και να την επαναπρογραμματίσει για μεταγενέστερη ημερομηνία.
Οι ερωτήσεις που περιέχονται στο διαγώνισμα είναι όλων των ειδών, υπάρχουν εύκολες ερωτήσεις, υπάρχουν ερωτήσεις παγίδες (trick questions) και τέλος υπάρχουν και δύσκολες ερωτήσεις που για να απαντηθούν σωστά θα πρέπει ο υποψήφιος να έχει εμβαθύνει αρκετά στη γνώση της γλώσσας. Ως αποτέλεσμα, το τελικό σκορ του κάθε υποψηφίου αντικατοπτρίζει πλήρως τις γνώσεις του και τις δεξιότητές του στη Java.
Εκτιμώ πως όσοι εργάζονται ως προγραμματιστές αξίζει να διεκδικήσουν τη συγκεκριμένη πιστοποίηση η οποία θα τους φανεί εξαιρετικά χρήσιμη στην αγορά εργασίας. Το μάθημα που επιλέξατε να παρακολουθήσετε θα σας προετοιμάσει κατάλληλα για το διαγώνισμα και θα σας παρέχει όλο το απαραίτητο υλικό, συμπεριλαμβανομένων και δοκιμαστικών ηλεκτρονικών διαγωνισμάτων που μπορείτε να εγκαταστήσετε στον υπολογιστή σας.

1.5 Πηγές Γνώσης

1. Βιβλιογραφία
   Το βιβλίο που έχετε ήδη προμηθευτεί είναι ένα πολύ καλό βιβλίο για εκμάθηση της Java από αρχάριους και θα σας βοηθήσει αρκετά στην πορεία του εξαμήνου. Στην αγορά υπάρχει υπερπληθώρα επιλογών για όποιον ενδιαφέρεται να προμηθευτεί έναν ακόμα τίτλο. Ένα από τα καλύτερα βιβλία (ξενόγλωσσο) που έχουν γραφτεί για την Java είναι το:
   - *Effective Java (2nd Edition), Joshua Bloch, Pearson Education, 2008*

   Για όσους ενδιαφέρονται να συμμετέχουν στο τεστ πιστοποίησης της Sun, ένας δεύτερος εξειδικευμένος τίτλος θα βοηθήσει πάρα πολύ. Τα παρακάτω βιβλία έχουν γραφτεί με γνώμονα την προετοιμασία του υποψηφίου για το τεστ και αποτελούν εξαιρετικό βοήθημα.
   - *SCJP Sun Certified Programmer for Java 6 Study Guide*, *Katherine Sierra, McGraw-Hill, 2008*
   - *Programmer's Guide to Java SCJP Certification: A Comprehensive Primer, Khalid Mughal, Addison Wesley, 2008*
2. Online παραδείγματα
   Στο διαδίκτυο οι πηγές που σχετίζονται με την εκμάθηση της Java είναι πραγματικά αμέτρητες. Ένα καλό site με tutorial στα Ελληνικά είναι το:
   http://conta.uom.gr/conta/ekpaideysh/seminaria/Tutorials-Java.html
3. Websites
   Το site της Sun Microsystems για τη Java ήταν και παραμένει το σημείο αναφοράς σε ό,τι έχει να κάνει με τη γλώσσα Java. Στο site αυτό θα βρείτε κάθε είδους υλικό συμπεριλαμβανομένων online παραδειγμάτων, άρθρων, case studies, software, τεκμηρίωσης κλπ. Η διεύθυνση είναι:
   http://java.sun.com

1.6 Ιστορία της Java

Η ανάπτυξη της Java ξεκίνησε το 1991, όταν στην Sun Microsystems βρίσκονταν σε αναζήτηση ενός κατάλληλου εργαλείου που θα μπορούσε να χρησιμοποιηθεί ως πλατφόρμα ανάπτυξης λογισμικού για «έξυπνες» μικροσυσκευές. Οι αρχικοί πειραματισμοί έγιναν από τον James Gosling (ο πατέρας της Java) κάνοντας χρήση της C++ η οποία όμως έδειχνε να είναι ακατάλληλη.
Ως συνέπεια, ο Gosling αποφάσισε να πειραματιστεί με τη δημιουργία μιας νέας γλώσσας η οποία θα έφερε αρκετά από τα χαρακτηριστικά της C++ και θα προσέθετε τα χαρακτηριστικά εκείνα που χρειάζονταν για τον προγραμματισμό μικροσυσκευών. Ύστερα από μία σειρά ενδιάμεσων γλώσσων (C++ ++) καταλήγει στην Oak (βελανιδιά). Η γλώσσα ονομάστηκε έτσι από μία βελανιδιά που βρισκόταν έξω από το γραφτείο του και που έβλεπε καθημερινά.
Η συγκεκριμένη γλώσσα διατηρούσε μεγάλη συγγένεια με τη C++ αλλά είχε πιο έντονο αντικειμενοστρεφή χαρακτήρα και χαρακτηριζόταν από την απλότητά της. Όταν λίγο αργότερα η ομάδα ανάπτυξης της Oak ενημερώνεται πως το συγκεκριμένο όνομα είναι ήδη κατοχυρωμένο αναγκάζεται να την μετονομάσει σε Java (είδος καφέ). Το όνομα προέκυψε από τις συχνές συναντήσεις τους σε τοπικές καφετέριες.
Τη δεδομένη χρονική περίοδο το project δείχνει να βρίσκεται σε αδιέξοδο μιας και ο αρχικός στόχος που ήταν η προώθηση μικροσυσκευών δεν έχει κεντρίσει το ενδιαφέρον των επενδυτών και συνεπώς δεν υπάρχει χρηματοδότηση, ενώ η αγορά την περίοδο εκείνη δείχνει να κινείται σε διαφορετικές κατευθύνσεις. Το ηθικό της ομάδας βρίσκεται στο ναδίρ και ενώ η Sun είναι έτοιμη να εγκαταλείψει το project, μια ευτυχής συγκυρία δημιουργείται με το ξεπέταγμα και την ευρεία διάδοση του internet. Οι ιθύνοντες του project βλέπουν αμέσως την ευκαιρία που τους παρουσιάζεται και κάνουν στροφή στο project προσαρμόζοντας τη γλώσσα στις νέες απαιτήσεις που δημιουργούνται για την ανάπτυξη διαδικτυακών εφαρμογών.
Έτσι, το 1996 έχουμε την πρώτη επίσημη παρουσίαση της Java από την Sun Microsystems, όπου τυγχάνει ευρείας αποδοχής. Γνωρίζοντας πως η δημιουργία προτύπου είναι απαραίτητη για κάθε σοβαρή γλώσσα προγραμματισμού, η Sun προσεγγίζει την επιτροπή ISO με σκοπό τη δημιουργία ενός τέτοιου standard για τη Java, αλλά για άγνωστους λόγους σύντομα αποσύρεται. Από εκείνη τη στιγμή και έως σήμερα, η Java παραμένει αυτό που στη βιομηχανία ονομάζεται de facto standard.
Το 1998 παρουσιάζεται η έκδοση 2 της Java η οποία περιέχει αρκετές προσθήκες σε σχέση με την πρώτη έκδοση. Από πολλούς προγραμματιστές ακόμη και σήμερα, η συγκεκριμένη έκδοση θεωρείται και η καλύτερη.
Η επόμενη έκδοση της Java θα καθυστερήσει έξι χρόνια, αλλά το 2004 θα γίνει η επίσημη παρουσίαση της έκδοσης 5 με την κωδική ονομασία Tiger. Δύο χρόνια αργότερα (2006) λανσάρεται η Java 6 με κωδική ονομασία Mustang, που είναι και η πιο πρόσφατη.
Τέλος, το 2007 η Sun Microsystems αποφασίζει να κάνει όλον τον πηγαίο κώδικα της Java open source με μοναδική εξαίρεση ένα μικρό κομμάτι, του οποίου τα πνευματικά δικαιώματα δεν κατείχε η ίδια.

1.7 Χαρακτηριστικά της Java

Τα πιο βασικά χαρακτηριστικά της Java είναι τα ακόλουθα:

- Είναι σχετικά απλή: Το συγκεκριμένο χαρακτηριστικό αναφέρεται σε πολλές πηγές σχετικές με τη γλώσσα Java, αλλά είναι καθαρά υποκειμενικό. Η Java είναι όσο απλή μπορεί να είναι

μία γλώσσα προγραμματισμού. Συγκρινόμενη με τη C++, είναι απλούστερη μιας και έχει εξαλειφθεί η χρήση δεικτών (pointers) ενώ η διαχείριση της μνήμης γίνεται από την ίδια τη γλώσσα.

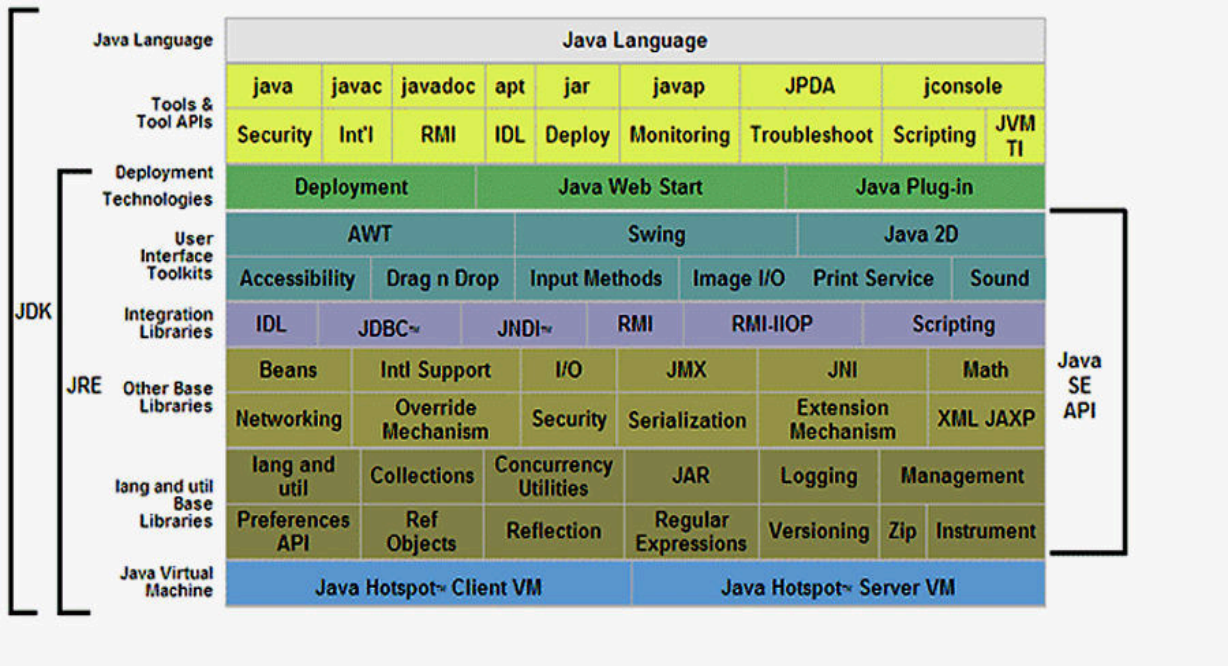
- Είναι μεταγλωττιζόμενη (compiled) και ερμηνευόμενη (interpreted): Σε αντίθεση με τις C/C++ που είναι compiled γλώσσες, η διαδικασία παραγωγής εκτελέσιμου κώδικα στη Java κάνει χρήση και των δύο αυτών τεχνικών. Η διαδικασία αυτή περιγράφεται αναλυτικά σε επόμενη υποενότητα.
- Είναι αμιγώς αντικειμενοστρεφής (pure OOD): Η Java υποστηρίζει αποκλειστικά το μοντέλο αντικειμενοστρεφούς προγραμματισμού (object-oriented paradigm) και δεν υπάρχει δυνατότητα χρήσης της σύμφωνα με κάποιο άλλο μοντέλο (π.χ. διαδικαστικό).
- Είναι φορητή σε επίπεδο μεταγλωττισμένου κώδικα: Ένα από τα πιο ισχυρά χαρακτηριστικά της Java. Αυτό πρακτικά σημαίνει πως ένας προγραμματιστής μπορεί να γράψει και μεταγλωττίσει ένα πρόγραμμα π.χ. σε Windows και στη συνέχεια να πάρει το αρχείο που παράχθηκε από τη μεταγλώττιση και να το τρέξει σε ένα μηχάνημα Unix χωρίς καμία αλλαγή. Απαραίτητη προϋπόθεση είναι στο μηχάνημα αυτό (Unix) να είναι εγκατεστημένο το αντίστοιχο JRE.
- Κάνει αυστηρό έλεγχο τύπων (strongly typed): Η Java απαιτεί από τον προγραμματιστή να κάνει σωστή χρήση των τύπων (περισσότερα για τους τύπους στην ενότητα 2) και δεν επιτρέπει αυθαίρετες μετατροπές όπως οι C/C++.
- Είναι γλώσσα υψηλού επιπέδου: Η εκμάθηση της Java και η σύνταξη κώδικα είναι σχετικά απλή μιας και η γλώσσα κάνει χρήση λέξεων που βρίσκονται πιο κοντά στη φυσική γλώσσα (Αγγλικά) παρά στη γλώσσα μηχανής.
- Παρέχει υψηλό επίπεδο ασφάλειας: Η εκτέλεση προγραμμάτων ελέγχεται από μηχανισμούς ασφαλείας που αποτρέπουν την κακόβουλου κώδικα.
- Υποστηρίζει πολυμέσα: Η Java είναι από τις ελάχιστες γλώσσες της κατηγορίας που παρέχουν έμφυτη υποστήριξη για την ανάπτυξη πολυμεσικών (multimedia) εφαρμογών.
- Είναι κατάλληλη για προγραμματισμό δικτυακών εφαρμογών: Όπως αναφέρθηκε σε προηγούμενη υποενότητα, η Java διευκολύνει την υλοποίηση τόσο δικτυακών (network) όσο και διαδικτυακών (web) εφαρμογών.
- Υποστηρίζει πολυνηματική επεξεργασία (multi-threaded processing): Και στην περίπτωση αυτή, η Java είναι μία από τις ελάχιστες γλώσσες της κατηγορίας της που παρέχει έμφυτη υποστήριξη για την ανάπτυξη multi-threaded εφαρμογών.
- Κάνει αυτόματη διαχείριση μνήμης: Στη Java, η διαχείριση της μνήμης ελέγχεται αποκλειστικά από αυτήν μέσω ενός υποπρογράμματος που ονομάζεται garbage collector (αποκομιστής απορριμάτων) και ο προγραμματιστής δεν εμπλέκεται ποτέ στη διαδικασία αυτή.
- Είναι δυναμική: Προσαρμόζεται εύκολα σε διαφορετικά περιβάλλοντα και απαιτήσεις και είναι ιδανική για τη διασύνδεση και επικοινωνία ετερογενών συστημάτων. Η Java ενημερώνεται συνεχώς ενσωματώνοντας και υποστηρίζοντας τις τελευταίες τεχνολογικές εξελίξεις.
- Κάνει αποκλειστική χρήση της δυναμικής διασύνδεσης (dynamic binding): Στη Java η διασύνδεση των δεδομένων και των μεθόδων που αυτά υποστηρίζουν γίνεται κατά την εκτέλεση του προγράμματος (run-time).

Γενικά, η φιλοσοφία της γλώσσας όσον αφορά το προγραμματιστικό μοντέλο που υιοθετεί είναι πως ο προγραμματιστής θα πρέπει να προστατεύεται από τη γλώσσα και τα προγράμματα να είναι ασφαλή και αξιόπιστα.

Η Java έχει αντλήσει όλη τη συσσωρευμένη γνώση και εμπειρία προγραμματισμού συστημάτων και έχει ενσωματώσει τις σωστές πρακτικές προγραμματισμού με αποτέλεσμα να θεωρείται σήμερα από πολλούς η πιο ολοκληρωμένη και καλά δομημένη γλώσσα της αγοράς. Δεν είναι άλλωστε τυχαίο πως είχε ευρεία αποδοχή αμέσως μετά από το λανσάρισμά της ενώ στις ημέρες μας η συντριπτική πλειονότητα των business projects παγκοσμίως υλοποιούνται σε Java. Εξαίρεση αποτελούν οι εφαρμογές όπου η ταχύτητα εκτέλεσης αποτελεί παράγοντα υψίστης σημασίας όπως π.χ. games, λειτουργικά συστήματα, βάσεις δεδομένων κλπ.
Η ταχύτητα παραμένει η Αχίλλειος πτέρνα της Java παρά τις διάφορες βελτιώσεις που έχουν γίνει κατά καιρούς ενώ το παράδοξο είναι πως αυτή η συγκεκριμένη αδυναμία είναι συνέπεια ενός από τα πιο ισχυρά χαρακτηριστικά της γλώσσας, της φορητότητας. Το γεγονός πως η Java παράγει ενδιάμεσο κώδικα ο οποίος στη συνέχεια ερμηνεύεται σε κώδικα μηχανής και όχι απ' ευθείας κώδικα μηχανής όπως η C++ χαρίζει μεν την ιδιότητα της φορητότητας στη γλώσσα, κάνοντάς την όμως να υστερεί σε ταχύτητα.

1.8 Αρχιτεκτονική της Java SE 6

Για να εκτελέσουμε ένα πρόγραμμα γραμμένο σε Java σε κάποιον υπολογιστή, είναι απαραίτητο στον υπολογιστή αυτόν να είναι εγκατεστημένο τουλάχιστον το αντίστοιχο JRE (Java Runtime Environment) για τον επεξεργαστή που διαθέτει και το λειτουργικό σύστημα που χρησιμοποιεί. Στο σχήμα 1 διακρίνονται τα συστατικά της αρχιτεκτονικής που χρησιμοποιούν τόσο το JRE της έκδοσης 6 της Java, όσο και το JDK (Java Development Kit).



Σχήμα 1

Το SDK και το JRE διατίθενται ξεχωριστά και το μεν πρώτο απευθύνεται σε όσους ενδιαφέρονται να προγραμματίσουν και να τρέξουν Java προγράμματα στον υπολογιστή τους, το δε δεύτερο σε όσους

απλά θέλουν να έχουν τη δυνατότητα να τρέχουν προγράμματα γραμμένα σε Java στο μηχάνημά τους. Όπως φαίνεται και από το σχήμα, το JDK περιέχει και το JRE, είναι δηλαδή υπερσύνολό του.
Ας δούμε όμως τα πιο σημαντικά κομμάτια της αρχιτεκτονικής της Java.

Java Virtual Machine (JVM): Η εικονική μηχανή (virtual machine) είναι ίσως το πιο σημαντικό συστατικό της αρχιτεκτονικής μιας και είναι υπεύθυνη για την εκτέλεση του ενδιάμεσου κώδικα (bytecode) και τη μετατροπή του στη γλώσσα που 'καταλαβαίνει' το λειτουργικό σύστημα και ο επεξεργαστής του συγκεκριμένου μηχανήματος. Ορισμένες από της πολλές λειτουργίες του JVM είναι το φόρτωμα των κλάσεων στη μνήμη από τον class loader, ο έλεγχος εγκυρότητας του bytecode και η αποτροπή εκτέλεσης κακόβουλου κώδικα από τον bytecode verifier, η διαχείριση της μνήμης, η λειτουργία του μεταγλωττιστή τελευταίας στιγμής (JIT compiler), ο αυτόματος χειρισμός εξαιρέσεων (automated exception handling) κλπ.

Java SE API (Application Programming Interface): Ένα σετ κλάσεις και interfaces που παρέχουν όλη τη λειτουργικότητα της γλώσσας.

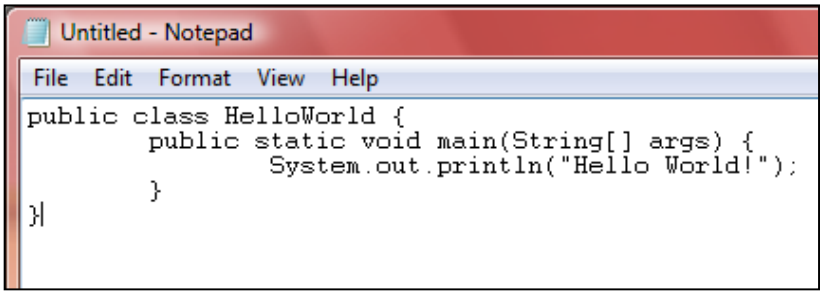
Deployment Technologies: Τεχνολογίες που διευκολύνουν το deployment (εγκατάσταση και εκτέλεση) εφαρμογών Java είτε αυτόνομα (stand-alone) είτε μέσα από κάποιον browser (π.χ. εκτέλεση κάποιου applet).

Tools: Ένα σετ από εργαλεία για τον προγραμματιστή που περιέχονται μόνο στο JDK. Τα πιο σημαντικά από αυτά είναι:

- **javac**: Ο compiler της Java
- **java**: Ο διερμηνέας της Java
- **javadoc**: Το εργαλείο που δημιουργεί HTML τεκμηρίωση σε στυλ API
- **jar**: Το εργαλείο που δημιουργεί και διαχειρίζεται αρχεία τύπου JAR
- **jdb**: Ο Java debugger
- **javap**: Disassembler αρχείων τύπου .class

1.9 Κύκλος Υλοποίησης

Ο κύκλος υλοποίησης στη Java αρχίζει με τη σύνταξη του κώδικα από τον προγραμματιστή και την αποθήκευσή του σε ένα αρχείο πηγαίου κώδικα (source code file). Ένα από τα χαρακτηριστικά της Java όπως και των περισσοτέρων γλώσσων προγραμματισμού είναι η μη απαίτηση συγκεκριμένου λογισμικού για τη δημιουργία προγραμμάτων. Πράγματι, τα μόνα που χρειαζόμαστε για να γράψουμε ένα απλό πρόγραμμα σε κάποιον υπολογιστή είναι ένας οποιοσδήποτε κειμενογράφος (π.χ. notepad) και ένα πρόσφατο JDK.



Σχήμα 2

Στο σχήμα 2 γίνεται χρήση του notepad των Windows για τη σύνταξη του κώδικα μιας απλής εφαρμογής, του διάσημου Hello world. Προσπαθήστε να ακολουθήσετε ένα ένα τα βήματα που περιγράφονται ώστε να γράψετε και να τρέξετε το πρώτο σας πρόγραμμα στη Java. Προσέξτε κατά την πληκτρολόγηση του κώδικα να μην κάνετε συντακτικά λάθη και ο κώδικάς σας να είναι ακριβώς ίδιος με αυτόν του σχήματος 2. Αποθηκεύστε το αρχείο με το όνομα *HelloWorld.java*.
Έχοντας γράψει τον κώδικα, το επόμενο βήμα είναι να τον μεταγλωττίσουμε. Ανοίξτε τη γραμμή εντολών (command) και θέστε ως τοποθεσία τον φάκελο που περιέχει το αρχείο που μόλις δημιουργήσατε. Στη συνέχεια πληκτρολογήστε

```
javac HelloWorld.java
```

και πατήστε Enter. Αυτό δίνει εντολή καλεί τον compiler της Java να ελέγξει το αρχείο μας για σφάλματα. Αν υπάρχουν σφάλματα (συντακτικά λάθη κλπ) τότε ο compiler θα μας τα υποδείξει, διαφορετικά θα παράξει ένα αρχείο με όνομα *HelloWorld.class*. Αν η απόκριση του συστήματος στην εντολή που δώσατε είναι η **`javac: file not found`** τότε δεν έχετε θέσει σωστά το command να βλέπει τον φάκελο που περιέχει το αρχείο πηγαίου κώδικα και ο compiler δεν το βρίσκει.
Το αρχείο *HelloWorld.class* είναι αυτό που περιέχει τον ενδιάμεσο κώδικα (bytecode). Για να εκτελέσετε το πρόγραμμά σας πληκτρολογήστε στο command

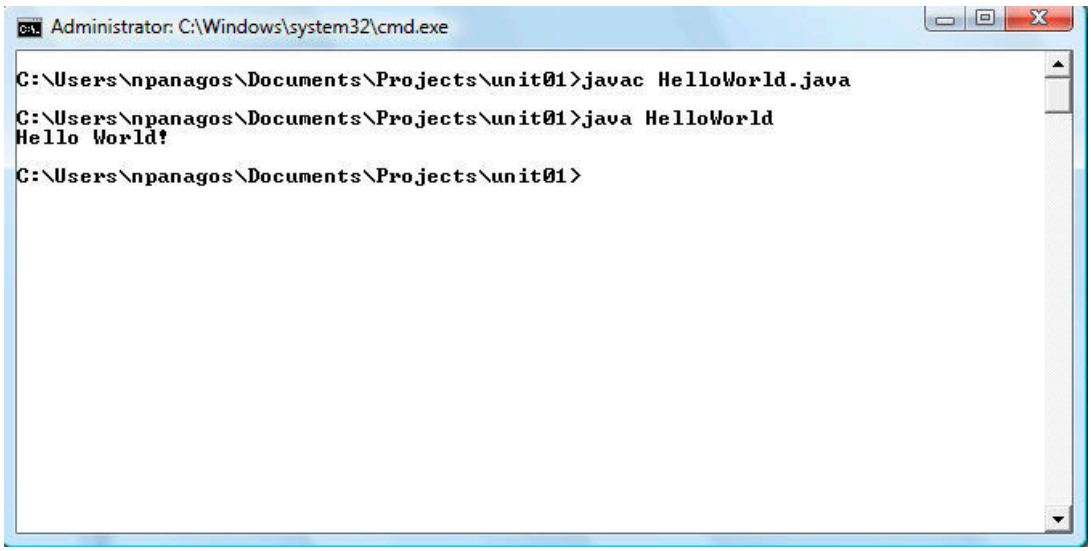
```
java HelloWorld
```

και πατήστε Enter. Το πρόγραμμά σας θα εκτελεστεί και θα πρέπει η έξοδός του να είναι αντίστοιχη με αυτή του σχήματος 3, που δείχνει όλη τη διαδικασία μεταγλώττισης και εκτέλεσης.
Ανακεφαλαιώντας λοιπόν, η εντολή για να μεταγλωττίσουμε ένα αρχείο πηγαίου κώδικα από το command είναι η:

```
javac onoma_arxeiou.java
```

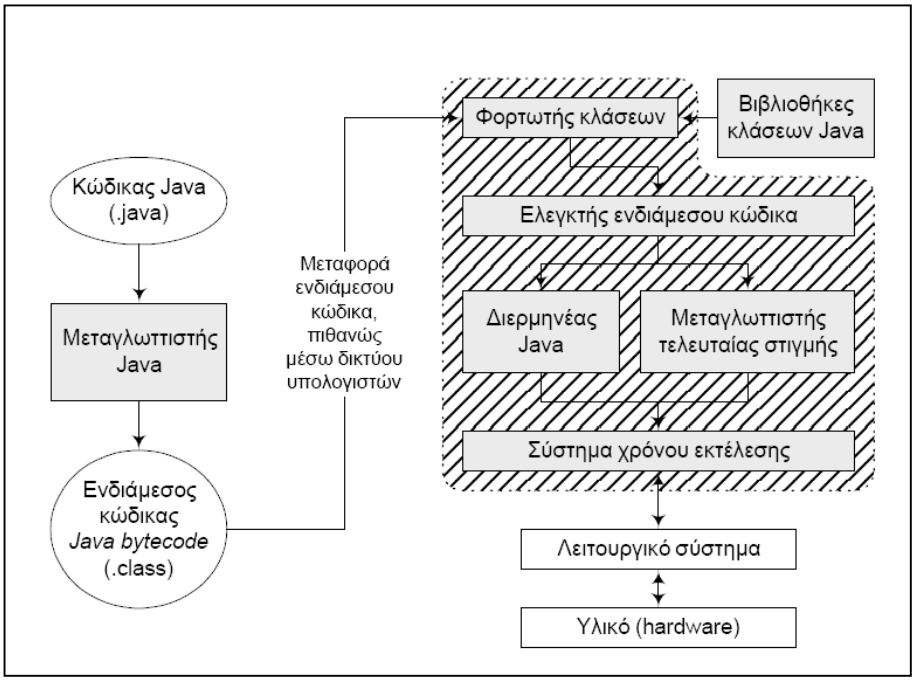
Για να εκτελέσουμε ένα αρχείο ενδιάμεσου κώδικα, προσέχουμε πως η εντολή είναι:

```
java onoma_klashs
```



Σχήμα 3

Στο σχήμα 4 απεικονίζεται όλη η παραπάνω διαδικασία με τα επιμέρους της βήματα. Το αρχείο πηγαίου κώδικα μεταγλωττίζεται και παράγεται ένα νέο αρχείο ενδιάμεσου κώδικα.
Στη συνέχεια το πρόγραμμα εκτελείται και η διαδικασία που ακολουθείται είναι αρχικά να φορτωθούν οι κλάσεις από τον class loader, στη συνέχεια να γίνει ο έλεγχος εγκυρότητας από τον bytecode verifier και τέλος να ξεκινήσει η μετατροπή του bytecode σε κώδικα μηχανής από τον διερμηνέα της Java. Μία από τις πρόσφατες προσθήκες στην JVM είναι ο μεταγλωττιστής τελευταίας στιγμής (JIT compiler) ο οποίος βοηθάει στην πιο γρήγορη εκτέλεση του κώδικα με το να μεταγλωττίζει κομμάτια που επαναλαμβάνονται. Η JVM φαίνεται στο γραμμοσκιασμένο κομμάτι του σχήματος.



Σχήμα 4

1.10 Integrated Development Environments (IDEs)

Φανταστείτε τώρα πως δουλεύετε σε ένα project που περιλαμβάνει δεκάδες κλάσεις και μερικές χιλιάδες γραμμές κώδικα. Είναι προφανές πως θα ήταν αδύνατο να υλοποιήσετε μια εφαρμογή ακολουθώντας τη διαδικασία που περιγράφηκε στην προηγούμενη υποενότητα, δηλαδή χρησιμοποιώντας έναν απλό κειμενογράφο και τη γραμμή εντολών. Ο απλός κειμενογράφος δε 'γνωρίζει' τη σύνταξη της γλώσσας και δε μπορεί να κάνει διαχωρισμό μεταξύ λέξεων που έχουν ειδική σημασία για τη γλώσσα και του λοιπού κώδικα.
Ο προγραμματιστής θα έπρεπε να συντάξει τον κώδικα χωρίς κανένα ουσιαστικό βοήθημα από τον κειμενογράφο, στη συνέχεια να τον μεταγλωττίσει φεύγοντας από τον κειμενογράφο και πηγαίνοντας σε κάποιο άλλο παράθυρο (αυτό του command), ενώ στην περίπτωση που υπήρχαν λάθη, θα έπρεπε να τα εντοπίσει και να τα διορθώσει ελέγχοντας μία μία τις λέξεις του κώδικα που βρίσκονται κοντά στη γραμμή που του υπέδειξε ο compiler. Κάτι τέτοιο είναι εντελώς αντιπαραγωγικό και φυσικά, για αρκετές δεκαετίες τώρα η ανάπτυξη εφαρμογών γίνεται με τη βοήθεια εξειδικευμένων εφαρμογών

που ονομάζονται Ενιαία Περιβάλλοντα Υλοποίησης (Integrated Development Environments) ή απλά IDEs.

Τα IDEs κάνουν τη ζωή του developer πιο εύκολη προσφέροντάς του μεγάλη ποικιλία βοηθημάτων τόσο σε βασικό όσο και σε σε πιο προχωρημένο επίπεδο. Έτσι λοιπόν, ακόμη και τα πιο απλά IDEs προσφέρουν έναν editor που 'γνωρίζει' τις λέξεις που έχουν ειδική σημασία για τη γλώσσα, προβάλλει με ξεχωριστό χρώμα τα διάφορα στοιχεία του κώδικα (δεσμευμένες λέξεις, σχόλια, αλφαριθμητικά) και μαρκάρει τις γραμμές που του υποδεικνύει ο compiler πως υπάρχουν σφάλματα. Επίσης, αναλαμβάνουν τη διαχείριση των αρχείων μας και μας επιτρέπουν να δουλεύουμε επάνω σε διαφορετικά projects ταυτόχρονα.

Το πιο ισχυρό χαρακτηριστικό τους όμως είναι η δυνατότητα που δίνουν στον προγραμματιστή να εκτελεί όλα τα βήματα του κύκλου υλοποίησης μέσα από ένα και μόνο περιβάλλον, αυτό του IDE. Τόσο η σύνταξη, η μεταγλώττιση η εκτέλεση και αποσφαλματοποίηση του κώδικα (debugging) γίνονται μέσα από το ίδιο περιβάλλον και όπως είναι φυσικό αυτό αυξάνει την παραγωγικότητα των ομάδων υλοποίησης.

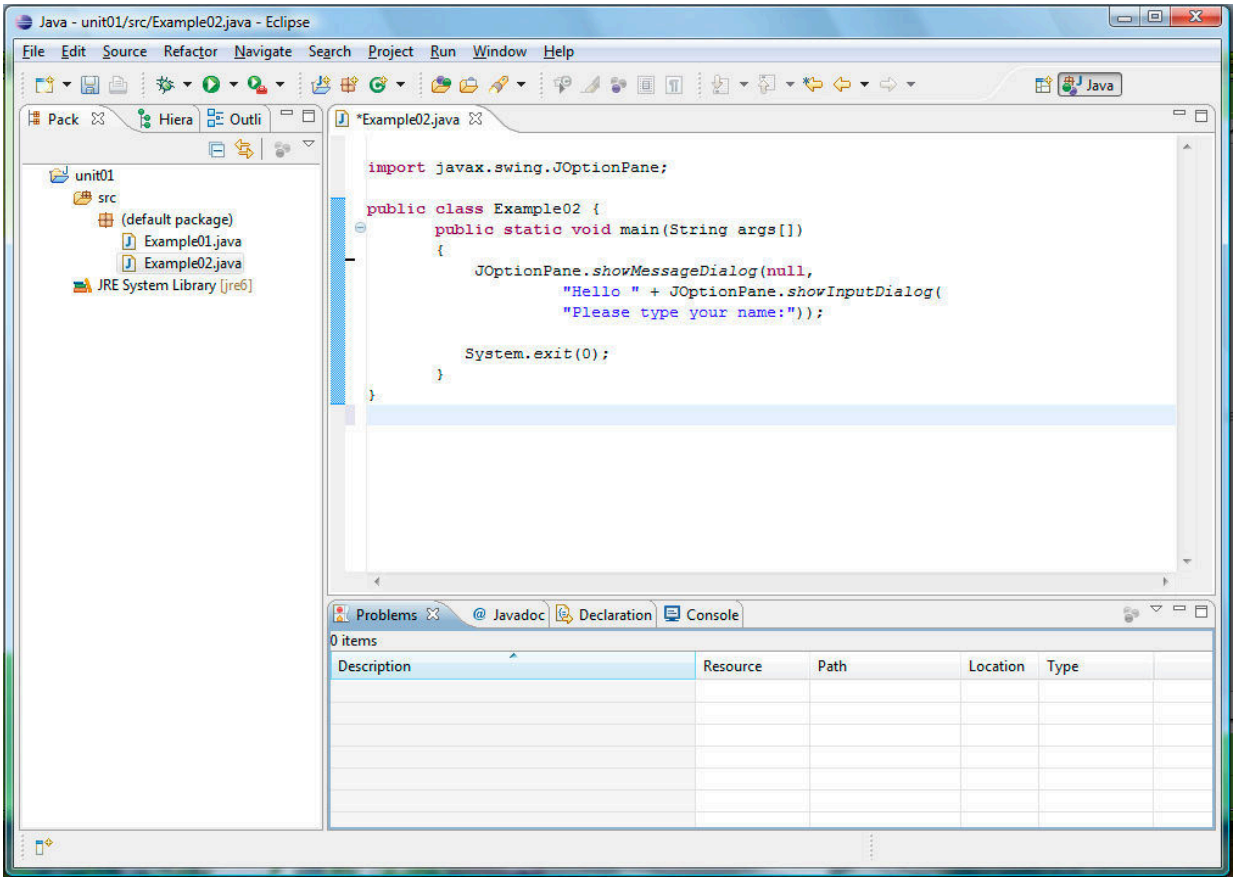
Στα προχωρημένα βοηθήματα των IDEs ανήκουν ο έλεγχος εκδόσεων (version control), το refactoring, η διατήρηση αρχείου ιστορικότητας, οι συμβουλές προς τον προγραμματιστή κατά τη σύνταξη με τη μορφή smart tags, οι προτάσεις για την επιδιόρθωση λαθών κλπ. Στις μέρες μας είναι αυτονόητο για κάποιον που θέλει να ασχοληθεί σοβαρά με τον προγραμματισμό πως θα χρησιμοποιήσει ένα τέτοιο IDE για τη συγκεκριμένη γλώσσα. Δεδομένης της ευρείας διάδοσης της Java, υπάρχουν αρκετά IDEs για να επιλέξει κανείς, τα πιο δημοφιλή και ποιοτικά από αυτά θεωρούνται το Eclipse και το NetBeans. Και τα δύο αυτά IDEs είναι open source που σημαίνει πως μπορούμε να τα προμηθευτούμε και να τα χρησιμοποιήσουμε εντελώς δωρεάν. Τόσο το Eclipse όσο και το NetBeans σας παρέχονται στο συνοδευτικό CD μαζί με οδηγίες για την εγκατάστασή τους και τη βασική χρήση τους. Και τα δύο αυτά περιβάλλοντα είναι εξαιρετικά ποιοτικά και μπορούν να χρησιμοποιηθούν όχι μόνο για ανάπτυξη σε Java αλλά και σε πολλές άλλες γλώσσες, γεγονός που τα κάνει ακόμα περισσότερο ελκυστικά.

Το Eclipse είναι προγενέστερο του NetBeans και έχει φτάσει σε ένα πολύ ικανοποιητικό επίπεδο ωρίμανσης, αν και ανά τακτά χρονικά διαστήματα λανσάρονται νέες εκδόσεις που το βελτιώνουν και του προσθέτουν λειτουργικότητα. Είναι το εργαλείο επιλογής της πλειονότητας των Java developers παγκοσμίως, συμπεριλαμβανομένου και του γράφοντος. Για τον λόγο αυτόν, τα διάφορα screenshots που θα συναντήσετε στις σημειώσεις σας αλλά και στις παρουσιάσεις θα προέρχονται από το συγκεκριμένο περιβάλλον. Διατίθεται τόσο για την πλατφόρμα των Windows όσο και για άλλες πλατφόρμες όπως το Linux και το Unix. Στο σχήμα 5 φαίνεται ένα screenshot του Eclipse.

Το NetBeans αποτελεί το αντίπαλο δέος του Eclipse αλλά ακόμη δεν έχει φτάσει στο απαιτούμενο επίπεδο ωρίμανσης μιας και πρόκειται για μεταγενέστερο project. Παρόλα αυτά πρόκειται για ένα επίσης πολύ καλό IDE και δεδομένου πως από το 2008 η ίδια η Sun το προτείνει για την ανάπτυξη Java εφαρμογών και το διαθέτει από το ίδιο της το site, αφήνει υποσχέσεις για το μέλλον. Όπως το Eclipse, έτσι και το NetBeans διατίθεται για διαφορετικά λειτουργικά συστήματα.

Άλλα γνωστά IDEs για ανάπτυξη σε Java είναι ο JDeveloper της Oracle (open source) και ο JBuilder της Codegear (πρώην Borland) το οποίο όμως είναι εμπορικό. Η επιλογή είναι δική σας.

<u>Συμβουλή για το διαγώνισμα της Sun</u>: Παρόλο που θα χρησιμοποιείτε κάποιο από τα δύο IDEs που σας προτάθηκαν για την ανάπτυξη των εφαρμογών σας και που αυτοματοποιούν τη διαδικασία μεταγλώττισης, θα πρέπει οπωσδήποτε να γνωρίζετε για το διαγώνισμα της Sun τη διαδικασία όπως περιγράφηκε στην υποενότητα 1.9, δηλαδή τη χρήση των **`javac`** *και* **`java`***.*



Σχήμα 5

1.11 Δομή Αρχείων στη Java

Σε αντίθεση με τις C/C++ όπου έχουμε αρχεία πηγαίου κώδικα αλλά και βιβλιοθήκες, στη Java υπάρχουν μόνο αρχεία πηγαίου κώδικα. Τα αρχεία αυτά έχουν πάντοτε κατάληξη .java, π.χ. *MyClass.java*.

Κανόνας σωστής πρακτικής: Κάθε αρχείο πηγαίου κώδικα θα πρέπει να περιέχει μία μόνο κλάση. Το αρχείο θα πρέπει να έχει ακριβώς το ίδιο όνομα με την κλάση.

Σημείωση: Η Java είναι case sensitive δηλαδή κάνει διαχωρισμό μεταξύ πεζών και κεφαλαίων χαρακτήρων. Αυτό σημαίνει πως για τη Java οι λέξεις myClass και MyClass αναφέρονται σε ξεχωριστές οντότητες. Βάσει αυτού και του παραπάνω κανόνα σωστής πρακτικής, αν δημιουργήσετε ένα αρχείο πηγαίου κώδικα στο IDE που χρησιμοποιείτε και το ονομάσετε MyClass.java, θα πρέπει το όνομα της κλάσης που περιέχει να είναι το MyClass.

Γενικά, για την ονομασία αρχείων πηγαίου κώδικα στη Java ισχύουν οι εξής κανόνες σωστής πρακτικής:

- Τα ονόματα μπορούν να περιέχουν μόνο αλφαριθμητικούς χαρακτήρες. Περίεργα σύμβολα όπως τα * - & (^ ~ ! κλπ απαγορεύονται. Το μόνο σύμβολο που επιτρέπεται είναι το underscore (_)
- Τα ονόματα θα πρέπει πάντα να ξεκινούν με γράμμα και όχι με αριθμό
- Τα ονόματα δε θα πρέπει να περιέχουν κενά (spaces). Αν κάποιο όνομα αποτελείται από δύο ή περισσότερες λέξεις, μπορείτε να τις διαχωρίσετε είτε με τη χρήση underscores είτε χρησιμοποιώντας κεφαλαίο χαρακτήρα στο πρώτο γράμμα κάθε λέξης. Για παράδειγμα, αν θέλετε να δημιουργήσετε ένα αρχείο που θα φιλοξενήσει μία κλάση που φορτώνει φωτογραφίες στη μνήμη (Image Loader) τα ονόματα ImageLoader και Image_Loader είναι τα πλέον κατάλληλα και πλήρως εναρμονισμένα με το συγκεκριμένο κανόνα.
- Τα ονόματα θα πρέπει να χρησιμοποιούν αποκλειστικά το Αγγλικό αλφάβητο.

Ένα Java project μπορεί να περιέχει πολλά αρχεία πηγαίου κώδικα. Για είναι εκτελέσιμο, θα πρέπει σε τουλάχιστον ένα από τα αρχεία του project να περιέχεται μία κεντρική μέθοδος, η οποία αναλύεται διεξοδικά στην υποενότητα 1.13.
Ένα τυπικό αρχείο πηγαίου κώδικα Java λοιπόν, μπορεί να περιέχει τα εξής:

- Δήλωση πακέτου
- Εντολές import
- Ορισμό κλάσης ή interface
- Σχόλια
- Μία κεντρική μέθοδο

Στο σχήμα 6 απεικονίζεται ένα απλό αρχείο πηγαίου κώδικα Java και υποδεικνύονται τα διάφορα συστατικά του. Κάθε ένα από αυτά θα το μελετήσουμε αναλυτικά στις ενότητες που ακολουθούν.

Σχήμα 6

1.12 Σχόλια (Comments)

Τα σχόλια είναι το πρώτο συστατικό της γλώσσας που διδάσκεστε. Είναι κομμάτια κειμένου που δεν έχουν καμία πρακτική αξία όσον αφορά στη λειτουργικότητα του προγράμματος αλλά παρέχουν σημαντική βοήθεια σε οποιονδήποτε διαβάσει τον κώδικα διευκολύνοντάς τον να τον κατανοήσει. Ο λόγος ύπαρξης των σχολίων λοιπόν είναι η τεκμηρίωση του κώδικα.
Για να καταλάβετε καλύτερα τη σημασία των σχολίων, υποθέστε πως ασχολείστε με τη δημιουργία και υλοποίηση ενός πολύπλοκου αλγορίθμου που θα χρησιμοποιηθεί σε κάποιο πρόγραμμα. Το πρόγραμμα ολοκληρώνεται και παραδίδεται κανονικά. Μετά από έναν χρόνο, ο project manager σας λέει πως θα πρέπει να κάνετε μικροδιορθώσεις στον αλγόριθμο ώστε να είναι περισσότερο

αποτελεσματικός. Χωρίς την ύπαρξη εύστοχων σχολίων σε καίρια σημεία του κώδικα, θα χρειαζόσασταν αρκετό χρόνο διαβάζοντας τον κώδικα ώστε να θυμηθείτε πως ακριβώς είχατε σκεφτεί όταν τον υλοποιούσατε. Ένα ακόμη πιο τρομακτικό σενάριο (για τον προγραμματιστή που δε γνωρίζει τι τον περιμένει) είναι αυτό όπου έχετε αποχωρήσει από την εταιρεία και ο κώδικάς σας δίνεται για βελτιστοποίηση σε κάποιον προγραμματιστή που τον βλέπει για πρώτη φορά!
Έχοντας κατανοήσει τη σημασία των σχολίων, ας δούμε τη σύνταξή τους. Στη Java υπάρχουν δύο τύποι σχολίων. Τα σχόλια που μπορούν να εκτείνονται σε περισσότερες της μιας γραμμές και αυτά της μίας γραμμής.
Τα σχόλια της πρώτης κατηγορίας ξεκινούν με την ακολουθία χαρακτήρων **/*** (κάθετος-αστεράκι) και τερματίζουν με την ακολουθία χαρακτήρων ***/** (αστεράκι-κάθετος). Για παράδειγμα, το σχόλια που ακολουθούν είναι και τα δύο συντακτικά ορθά.

```
/* auto einai ena syntaktika ortho sxolio */

/*
 * opws
 * kai
 * ayto
 */
```

Οτιδήποτε βρίσκεται μεταξύ των χαρακτήρων έναρξης και τερματισμού των σχολίων αγνοείται παντελώς από τον compiler, πράγμα που σημαίνει πως μπορείτε να γράψετε οτιδήποτε θελήσετε, σχόλια στα Αγγλικά, σχόλια σε Greeklish, σχόλια στα Ελληνικά κλπ.
Τα σχόλια μίας γραμμής ξεκινούν με την ακολουθία χαρακτήρων **//** (2 κάθετοι) και τελειώνουν στο τέλος της τρέχουσας γραμμής χωρίς να χρειάζεται να τερματιστούν. Το ακόλουθο για παράδειγμα είναι ένα σωστό σχόλιο μίας γραμμής:

```
// this is a single line comment
```

Τα σχόλια συνήθως προβάλλονται στους editors των IDEs με πράσινο χρώμα. Η σωστή πρακτική για τη χρήση σχολίων προτείνει τη χρήση τους με μέτρο, θα πρέπει δηλαδή να χρησιμοποιούνται μόνο εκεί που πραγματικά χρειάζονται. Κώδικας που δεν περιέχει καθόλου σχόλια είναι αρκετά δυσνόητος αλλά στον αντίποδα, κώδικας που περιέχει υπερβολικό αριθμό σχολίων γίνεται εξαιρετικά δυσανάγνωστος και κουραστικός.
Τέλος, με τη χρήση ειδικών σχολίων (**/** ***/**) και ειδικών tags που ονομάζονται annotations και σε συνδυασμό με το εργαλείο javadoc του JDK είναι δυνατόν να παράξουμε ένα HTML reference manual για τον κώδικά μας, στο στυλ του Java API reference manual. Η συγκεκριμένη λειτουργία δεν συμπεριλαμάνεται στην ύλη του διαγωνίσματος της Sun και σας παρουσιάζεται απλά ενημερωτικά. Στο παράδειγμα που ακολουθεί γίνεται χρήση τέτοιων annotations εντός των σχολίων.

```
/**
* This class represents a jpg image
*
* @author Nikos Panagos
* @version 1.0, March 23, 2008
* @see server
*/
```

Συμβουλή για το διαγώνισμα της Sun: Αν και τα σχόλια αποτελούν ένα από τα πιο εύκολα κομμάτια της ύλης, θα πρέπει να είστε σε θέση να ξεχωρίζετε σχόλια που είναι σωστά γραμμένα από σχόλια που θα προκαλέσουν συντακτικά λάθη.

1.13 Κεντρική Μέθοδος (`main`)

Όπως αναφέρθηκε ήδη, για να είναι ένα πρόγραμμα εκτελέσιμο, θα πρέπει οπωσδήποτε σε κάποιο από τα αρχεία πηγαίου κώδικα από τα οποία αποτελείται να υπάρχει τουλάχιστον μία κεντρική μέθοδος (main method). Αυτή είναι μία ακόμη διαφορά της Java από τις C/C++ μιας και στη Java μπορούμε να έχουμε περισσότερες της μιας κεντρικές μεθόδους (σε διαφορετικά βέβαια αρχεία) και ο χρήστης να επιλέξει ποια θα είναι αυτή που θα εκτελεστεί, ενώ στις C/C++ θα πρέπει να υπάρχει αποκλειστικά μία κεντρική συνάρτηση ανά project.
Η κεντρική μέθοδος αποτελεί το σημείο ειδόδου στο πρόγραμμά μας και όπως όλες οι μέθοδοι περικλείεται σε ένα ζεύγος από άγκιστρα (**{ }**) και περιέχει μία σειρά από εντολές.
Όταν δώσουμε εντολή στο διερμηνέα της Java να εκτελέσει το πρόγραμμά μας, π.χ. εκτελώντας

```
java MyClass
```

ο διερμηνέας θα ξεκινήσει να εκτελεί τις εντολές που περιέχονται στη **`main`** αρχίζοντας από αυτήν που βρίσκεται αμέσως μετά το άγκιστρο έναρξης και καταλήγοντας αφού εκτελέσει και αυτήν που βρίσκεται αμέσως πριν το άγκιστρο τερματισμού. Η εντολές εκτελούνται η μία μετά την άλλη με τη σειρά που είναι γραμμένες (σειριακά – sequentially).
Στη Java η κεντρική μέθοδος έχει πάντοτε τη μορφή:

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

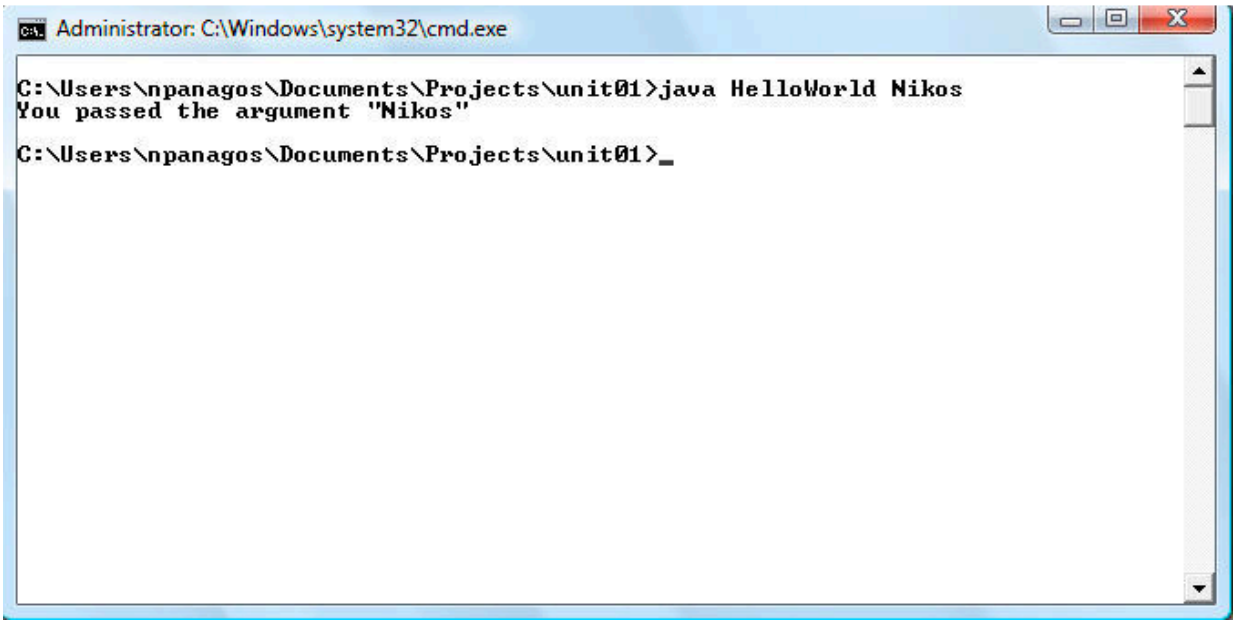
Ως εντολή ορίζουμε στη Java μία 'έκφραση' που ξεκινάει από την αρχή μιας γραμμής και τερματίζει με το χαρακτήρα ; (ερωτηματικό – semi-colon) όντας παράλληλα συντακτικά ορθή. Σύμφωνα με την ισχύουσα πρακτική, θα πρέπει στα προγράμματά μας να έχουμε μία εντολή ανά γραμμή. Μία τέτοια εντολή λοιπόν μπορεί να είναι η δήλωση μιας μεταβλητής, η ορθή χρήση μιας δεσμευμένης λέξης της γλώσσας, η κλήση μιας μεθόδου, μία αριθμητική πράξη κλπ. Μία ομάδα εντολών που περικλείονται σε ένα ζεύγος αγκίστρων (**{ }**) ονομάζεται μπλοκ εντολών.
Ένα ακόμη χαρακτηριστικό της κεντρικής μεθόδου στη Java είναι η δυνατότητα που παρέχει να περνάμε παραμέτρους (command line arguments) κατά την εντολή εκτέλεσης του προγράμματος, οι οποίες μπορεί να χρησιμοποιηθούν από τον κώδικα για την επίτευξη συγκεκριμένης λειτουργικότητας. Στον κώδικα του σχήματος 7, για να εκτελεστεί σωστά το πρόγραμμα θα πρέπει όταν κληθεί ο διερμηνέας να του περάσουμε και μία ακόμα παράμετρο την οποία στη συνέχεια θα χρησιμοποιήσει για να μας προβάλλει το μήνυμα:

```
You passed the argument "[παράμετρος]"
```

Ένα δείγμα εξόδου του προγράμματος φαίνεται στο σχήμα 8.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("You passed the argument \"" + args[0] + "\"");
    }
}
```

Σχήμα 7



Σχήμα 8

Συμβουλή για το διαγώνισμα της Sun: Για το διαγώνισμα της Sun θα πρέπει να γνωρίζετε τη σωστή μορφή της κεντρικής μεθόδου και τη δυνατότητά χρήσης της σε συνδυασμό με παραμέτρους. Αν και όπως αναφέρθηκε η κεντρική μέθοδος έχει σχεδόν πάντα τη μορφή
`public static void main(String[] args)`
υπάρχουν και δύο παραλλαγές που μπορεί να συναντήσετε, οι οποίες είναι εξίσου έγκυρες. Αυτές ειναι οι:

`public static void main(String args[])` *και η*
`public static void main(String[] something)`

Στη μέν πρώτη περίπτωση είναι απόλυτα έγκυρο να γράψουμε τις αγκύλες μετά από το όνομα της παραμέτρου ενώ στη δεύτερη, μπορούμε να δώσουμε όποιο όνομα θέλουμε σε παραμέτρους μεθόδων. Αντίστοιχα έγκυρη θα ήταν και η σύνταξη που χρησιμοποιεί τον συνδυασμό και των δύο αυτών παραλλαγών, δηλαδή τις αγκύλες μετά το όνομα της παραμέτρου, καθώς και διαφορετικό όνομα παραμέτρου, π.χ.

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

1.14 Δεσμευμένες Λεξεις της Java

Η ακόλουθη λίστα περιέχει τις δεσμευμένες λέξεις στη Java. Μέχρι το τέλος του σεμιναρίου θα τις έχουμε χρησιμοποιήσει σχεδόν όλες.

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

* Δε χρησιμοποιείται
** Εντάχθηκε στην έκδοση 1.2
*** Εντάχθηκε στην έκδοση 1.4
**** Εντάχθηκε στην έκδοση 5

*Συμβουλή για το διαγώνισμα της Sun: Στο διαγώνισμα θα υπάρχει τουλάχιστον μία ερώτηση επάνω στις δεσμευμένες λέξεις και άρα θα πρέπει να τις γνωρίζετε (θεωρείται εύκολη ερώτηση). Προσοχή, στις λέξεις **goto** και **const** οι οποίες αν και έχουν δεσμευτεί από τη Java ίσως για λόγους συμβατότητας με τη C++, δε χρησιμοποιούνται και δε θεωρούνται έγκυρες. Αν τις χρησιμοποιήσετε σε κάποιο πρόγραμμά σας, ο compiler θα απαιτήσει να τις σβήσετε. Ανήκουν πάντως στις δεσμευμένες λέξεις.*

1.15 Απλή Έξοδος στην Κονσόλα

Μία από τις πιο κοινές λειτουργίες των προγραμμάτων είναι η προβολή μηνυμάτων ή αποτελεσμάτων στο χρήστη. Για εφαρμογές γραμμής εντολών, η συγκεκριμένη λειτουργία επιτυγχάνεται με τη χρήση του αντικειμένου **System.out** που είναι το βασικό αντικείμενο εξόδου για τη Java. Η χρήση του είναι πολύ απλή μιας και δεν απαιτείται η εισαγωγή κάποιου πακέτου και στην πιο απλή μορφή της εμφανίζει μέσω της κλήσης μιας εκ των **println** ή **print** κάποιο αλφαριθμητικό στην κονσόλα.
Η **println** εμφανίζει το αλφαριθμητικό που της περνάμε ως παράμετρο στην κονσόλα και κατεβαίνει στην επόμενη γραμμή ενώ η **print** έχει ακριβώς την ίδια λειτουργία χωρίς όμως να κατέβει στην επόμενη γραμμή. Διαδοχικές κλήσεις της **print** δηλαδή εξακολουθούν να εμφανίζουν

τα μηνύματα στην ίδια γραμμή μέχρις ότου ο προγραμματιστής εισάγει έναν ειδικό χαρακτήρα που θα πληροφορήσει τον compiler να κατέβει στην επόμενη γραμμή (**\n**) ή καλέσει την **println**.
Χρησιμοποιώντας τις μεθόδους αυτές σε συνδυασμό με τον τελεστή **+** μπορούμε εκτός από αλφαριθμητικά να εμφανίζουμε στην κονσόλα και τιμές μεταβλητών. Ο ακόλουθος κώδικας για παράδειγμα θα εμφανίσει στην κονσόλα το μήνυμα **x = 5.**

```
...
int x = 5;
System.out.println("x = " + x);
...
```

Η έξοδος στην κονσόλα εξετάζεται με περισσότερη λεπτομέρεια στην ενότητα 7, αλλά η αναφορά της στην παρούσα υποενότητα ήταν απαραίτητη, δεδομένου πως θα σας χρειαστεί για την υλοποίηση των εργασιών σας.

1.16 Ακολουθίες Διαφυγής (Escape Sequences)

Σε όλες τις γλώσσες προγραμματισμού υπάρχουν κάποια σύμβολα που έχουν ειδική σημασία για τη γλώσσα. Χαρακτηριστικό παράδειγμα στη Java είναι τα διπλά εισαγωγικά (") που σηματοδοτούν την αρχή και το τέλος ενός αλφαριθμητικού (string). Το γεγονός αυτό, ότι δηλαδή τα σύμβολα αυτά έχουν ειδική σημασία για τη γλώσσα δημιουργεί προβλήματα στις περιπτώσεις που θέλουμε να τα χρησιμοποιήσουμε με διαφορετικό τρόπο από αυτόν που αποτελεί την πρωταρχική τους λειτουργία.
Για παράδειγμα, ας υποθέσουμε πως θέλετε σε κάποιο πρόγραμμά σας να εμφανίσετε το γνωστό Σαιξπηρικό ερώτημα "To be or not to be?" (μαζί με τα εισαγωγικά).
Αν χρησιμοποιήσετε την εντολή

```
System.out.println("To be or not to be?");
```

θα προβληθεί μεν το μήνυμα αλλά χωρίς τα εισαγωγικά γιατί αυτά όπως είπαμε στη συγκεκριμένη εντολή ορίζουν την αρχή και το τέλος του αλφαριθμητικού.
Αν τώρα δοκιμάσετε την εντολή

```
System.out.println(""To be or not to be?"");
```

θα διαπιστώσετε πως η συγκεκριμένη γραμμή προκαλεί σφάλμα κατά τη μεταγλώττιση. Πράγματι, ο compiler ερμηνεύει το πρώτο ζεύγος εισαγωγικών ως ένα κενό αλφαριθμητικό, ακολουθεί μία σειρά λέξεων τις οποίες ο compiler δεν αναγνωρίζει (η φράση 'To be or not to be' δεν έχει κανένα νόημα για αυτόν) και τέλος υπάρχει ένα ακόμη κενό αλφαριθμητικό.
Για την αντιμετώπιση τέτοιων καταστάσεων υπάρχουν σε κάθε γλώσσα οι λεγόμενες ακολουθίες διαφυγής (escape sequences). Πρόκειται για ακολουθίες χαρακτήρων που όταν ο compiler τις 'συναντήσει' γνωρίζει πως να τις χειριστεί, π.χ. καταλαβαίνει πως ο προγραμματιστής επιθυμεί να συμπεριλάβει διπλά εισαγωγικά σε κάποιο αλφαριθμητικό κλπ. Τέτοιες ακολουθίες διαφυγής υπάρχουν για όλους τους χαρακτήρες με ειδικό νόημα για τη γλώσσα και οι αντίστοιχες της Java συνοψίζονται στον πίνακα 1.

Ακολουθία Διαφυγής	Χαρακτήρας
\n	newline
\t	tab
\b	backspace
\f	form feed
\r	return
\"	" (διπλά εισαγωγικά)
\'	' (μονό εισαγωγικό)
****	\ (back slash)
\uDDDD	Ο χαρακτήρας Unicode με κωδικό DDDD (DDDD είναι 4ψήφιος δεκαεξαδικός αριθμός)

Πίνακας 1

Από τον πίνακα βλέπουμε πως για τα διπλά εισαγωγικά, η ακολουθία διαφυγής είναι η **\"**. Επιστρέφοντας λοιπόν στο παράδειγμά μας, για να επιτύχουμε την προβολή του επιθυμητού μηνύματος θα γράφαμε

```
System.out.println("\"To be or not to be?\"");
```

Η συγκεκριμένη ακολουθία διαφυγής χρησιμοποιήθηκε και στο παράδειγμα της υποενότητας 1.13 (κεντρική μέθοδος με παραμέτρους) όπου μπορείτε να μελετήσετε τη χρήση της σε ένα ολοκληρωμένο πρόγραμμα. Από τις ακολουθίες διαφυγής του πίνακα 1, οι πιο κοινές πλην των διπλών εισαγωγικών είναι το newline που όπως είπαμε κατεβάζει τον κέρσορα στην επόμενη γραμμή, το tab, το return και η έκφραση αναπαράστασης ενός κωδικού Unicode.

1.17 Πακέτα (Packages)

Τα πακέτα είναι το δεύτερο συστατικό της γλώσσας που διδάσκεστε στην πρώτη αυτή ενότητα. Πρόκειται για ένα ιδιαίτερα σημαντικό κομμάτι της γλώσσας μιας και παρέχει στους προγραμματιστές έναν μηχανισμό ιεραρχικής οργάνωσης των κλάσεων και διευκολύνει την κατάτμηση του κώδικα (modularity). Φανταστείτε τα πακέτα σαν κουτιά που μέσα τους περιέχουν κλάσεις, interfaces (οι κλάσεις και τα interfaces καλύπτονται διεξοδικά στις ενότητες 4 και 5) καθώς και άλλα πακέτα.
Όλες οι κλάσεις της Java είναι οργανωμένες σε τέτοια πακέτα ενώ παράλληλα δίνεται η δυνατότητα στον προγραμματιστή να δημιουργήσει δικά του, όπως θα δούμε στη συνέχεια. Κάθε πακέτο έχει συγκεκριμένο όνομα γραμμένο κατά σύμβαση με πεζούς πάντοτε χαρακτήρες ώστε να ξεχωρίζει από τα ονόματα των κλάσεων και μπορεί να περιέχει όπως προαναφέρθηκε κλάσεις, interfaces και άλλα πακέτα. Όταν αναφερόμαστε σε κάποιο πακέτο ξεκινάμε από το κορυφαίο στην ιεραρχία και

προχωράμε στο πακέτο που επιθυμούμε διαχωρίζοντάς τα με τελείες. Για παράδειγμα με την έκφραση

```
java.util
```

αναφερόμαστε στο πακέτο *util* που περιέχεται στο πακέτο *java*. Το πακέτο *java* βρίσκεται στην κορυφή της ιεραρχίας. Αν θέλουμε να χρησιμοποιήσουμε στον κώδικά μας μία κλάση που περιέχεται σε κάποιο πακέτο, θα πρέπει να πληκτρολογήσουμε το πλήρες όνομά της (fully qualified name) ώστε ο compiler να μπορέσει να την εντοπίσει. Για παράδειγμα η εντολή

```
java.util.Stack s = new java.util.Stack();
```

υποδεικνύει στον compiler να δημιουργήσει ένα αντικείμενο της κλάσης **Stack** που περιέχεται στο πακέτο *util*, το οποίο με τη σειρά του περιέχεται στο πακέτο *java* που βρίσκεται στην κορυφή της ιεραρχίας. Η αναφορά χρησιμοποιώντας το πλήρες όνομα δεν είναι απαραίτητη μόνο στην περίπτωση που οι δύο κλάσεις βρίσκονται στο ίδιο πακέτο και άρα 'βλέπουν' η μία την άλλη.
Είναι προφανές πως η χρήση του πλήρους ονόματος απαιτεί αρκετή πληκτρολόγηση από πλευράς προγραμματιστή και για την καταπολέμηση του συγκεκριμένου προβλήματος η ομάδα της Java δημιούργησε τη λύση του **import**. Κάνοντας χρήση της δεσμευμένης λέξης **import** σε συνδυασμό με το πλήρες όνομα της κλάσης που θέλουμε να χρησιμοποιήσουμε στον κώδικά μας δε χρειάζεται πλέον να αναφερόμαστε σε αυτήν με το πλήρες όνομα και αρκεί να χρησιμοποιήσουμε το απλό όνομά της. Το προηγούμενο παράδειγμα δηλαδή, κάνοντας χρήση του **import** θα μετατρεπόταν ως εξής:

```
import java.util.Stack;
…
Stack s = new Stack();
```

Η χρήση **import** δεν έχει καμία αρνητική συνέπεια όσον αφορά το μέγεθος του προγράμματός μας (δεν εισάγεται ο κώδικας της κλάσης) και μπορείτε να το χρησιμοποιείτε κατά βούληση. Ένας άλλος τρόπος χρήσης του **import** είναι σε συνδυασμό με το χαρακτήρα μπαλαντέρ (wildcard) όπως φαίνεται στην έκφραση που ακολουθεί:

```
import java.util.*;
```

Η συγκεκριμένη εντολή υποδεικνύει στον compiler να συμπεριλάβει τα ονόματα όλων των κλάσεων και interfaces που περιέχονται στο πακέτο *java.util* (όχι όμως τα ονόματα των πακέτων που μπορεί να περιέχονται σε αυτό). Αν και κάτι τέτοιο δείχνει βολικό, είναι προτιμότερο να είσαστε συγκεκριμένοι όταν υποδεικνύετε στον compiler ποιες κλάσεις θέλετε να εισάγετε και να χρησιμοποιείτε το χαρακτήρα μπαλαντέρ μόνο όταν θέλετε να εισάγετε πολλές κλάσεις από το ίδιο πακέτο.
Ας εξετάσουμε όμως για ποιο λόγο είναι χρήσιμα τα πακέτα. Ας υποθέσουμε πως μία ομάδα προγραμματιστών δουλεύει σε ένα μεγάλο project και πως ο προγραμματιστής Α έχει δημιουργήσει μία κλάση που παρέχει βοηθητικές λειτουργίες για την επεξεργασία ψηφιακών φωτογραφιών και την έχει ονομάσει **Helper**. Εντελώς συμπτωματικά, ο προγραμματιστής Β έχει δημιουργήσει και αυτός μια κλάση που παρέχει βοηθητικές λειτουργίες για την επεξεργασία ψηφιακού ήχου και την έχει ονομάσει επίσης **Helper**. Στο συγκεκριμένο σενάριο, όταν τα κομμάτια κώδικα της ομάδας

συντίθονταν για να ξεκινήσουν τα τεστ, ο compiler δε θα μπορούσε να κάνει διαχωρισμό μεταξύ των δύο κλάσεων με το ίδιο όνομα και η μεταγλώττιση θα αποτύγχανε.
Η λύση στο συγκεκριμένο πρόβλημα έρχεται με τη χρήση πακέτων, αυτός άλλωστε είναι και ο λόγος για τον οποίον δημιουργήθηκαν. Η Java μας δίνει τη δυνατότητα να φτιάξουμε δικά μας πακέτα χρησιμοποιώντας τη δεσμευμένη λέξη **package** σε συνδυασμό με ένα όνομα της επιλογής μας. Για παράδειγμα η εντολή

```
package sound;
```

υποδεικνύει στον compiler να τοποθετήσει την κλάση που περιέχεται στο συγκεκριμένο αρχείο πηγαίου κώδικα μέσα στο πακέτο *sound*. Αντίστοιχα, η εντολή

```
package elearning.lesson;
```

τοποθετεί την κλάση που περιέχεται στο συγκεκριμένο αρχείο μέσα στο πακέτο *lesson*, το οποίο με τη σειρά του περιέχεται στο πακέτο *elearning* που βρίσκεται στην κορυφή της ιεραρχίας.
Επιστρέφοντας στο σενάριο των δύο προγραμματιστών και των κλάσεων με τα ίδια ονόματα, μια ενδεικτική λύση θα ήταν ο ένας να τοποθετήσει την κλάση του σε ένα πακέτο με το όνομα *imaging* και ο άλλος σε ένα πακέτο με το όνομα *sound*. Με τον τρόπο αυτό, από οποιοδήποτε σημείο του κώδικα θα μπορούσαμε να υποδείξουμε ρητά στον compiler σε ποια από τις δύο αυτές κλάσεις αναφερόμαστε.
Θα πρέπει να τονιστεί στο σημείο αυτό πως σε περιπτώσεις σαν αυτήν (δύο ή περισσοτέρων κλάσεων με το ίδιο όνομα σε διαφορετικά πακέτα) όταν κάνουμε χρήση **import** θα πρέπει να πληκτρολογούμε το πλήρες όνομα και όχι να χρησιμοποιούμε το χαρακτήρα μπαλαντέρ, δηλαδή

```
import imaging.Helper;
```

και

```
import sound.Helper;
```

Σε ένα αρχείο πηγαίου κώδικα, η εντολή ορισμού πακέτου (**package**) τοποθετείται πάντοτε στην πρώτη γραμμή και στη συνέχεια ακολουθούν τα **import** statements. Τέλος, ακολουθεί η δήλωση της κλάσης, όπως φαίνεται στο ακόλουθο απόσπασμα:

```
package elearning.sound;

import java.util.*;

public class Helper {

      ...

}
```

Το τελευταίο αλλά πολύ σημαντικό ζήτημα που θα μας απασχολήσει σχετικά με τα πακέτα είναι αυτό του *default* πακέτου. Εάν όταν δημιουργούμε μία κλάση δεν την τοποθετήσουμε σε κάποιο πακέτο, τότε η Java την τοποθετεί αυτόματα σε ένα 'ειδικό' πακέτο που ονομάζεται *default* πακέτο. Ο

συγκεκριμένος διακανονισμός λειτουργεί απροβλημάτιστα για μικρά προγράμματα όπως αυτά που θα υλοποιήσετε στην παρούσα ενότητα, όχι όμως όταν ασχοληθείτε με μεγαλύτερα projects όπου γίνεται χρήση πολλών κλάσεων που μπορεί να προέρχονται και από διαφορετικές πηγές.
Συγκεκριμένα, το πρόβλημα έγκειται στο γεγονός πως καμία κλάση που ανήκει στο *default* πακέτο δε μπορεί να γίνει import (και άρα να χρησιμοποιηθεί) από κώδικα που βρίσκεται σε οποιαδήποτε κλάση άλλου πακέτου. Με τον τρόπο αυτόν λοιπόν, η Java σχεδόν μας εξαναγκάζει να οργανώνουμε και να τοποθετούμε τις κλάσεις μας σε πακέτα. Τα IDEs μάλιστα όπως το Eclipse και το NetBeans μας ενημερώνουν με αντίστοιχο προειδοποιητικό μήνυμα αν στο διάλογο δημιουργίας μιας νέας κλάσης αφήσουμε το πεδίο ονομασίας πακέτου κενό. Όπως αναφέρθηκε προηγουμένως, για μικρά προγράμματα μπορείτε να κάνετε χρήση του default πακέτου χωρίς να έχετε προβλήματα, αλλά θα ήταν καλό να μάθετε από τώρα να τοποθετείτε τις κλάσεις πάντοτε σε κάποιο πακέτο.

1.18 Κοινοί Διάλογοι (Common Dialogs)

Στην τελευταία αυτή υποενότητα θα ανεφερθούμε συνοπτικά στους πιο χρήσιμους από τους κοινούς διαλόγους (common dialogs) που διαθέτει η Java και που θα χρησιμοποιήσετε στις εργασίες σας για είσοδο και έξοδο. Οι διάλογοι δε συμπεριλαμβάνονται στην εξεταστέα ύλη του διαγωνίσματος της Sun.
Όλες οι γλώσσες προγραμματισμού που παρέχουν τη δυνατότητα δημιουργίας παραθυρικών εφαρμογών διαθέτουν μία σειρά διαλόγων που ονομάζονται κοινοί διάλογοι (common dialogs) και διευκολύνουν την εισαγωγή δεδομένων από το χρήστη, την προβολή μηνυμάτων κλπ. Η Java δε θα μπορούσε να αποτελεί εξαίρεση και περιέχει τέτοιους διαλόγους στο πακέτο *javax.swing*, στο οποίο βρίσκονται όλες οι κλάσεις για τη δημιουργία GUI (Graphical User Interface). Τις δυνατότητες του πακέτου *javax.swing* θα τις εξετάσουμε στην ενότητα 10. Οι δύο διάλογοι που θα χρησιμοποιήσουμε είναι ο διάλογος προβολής μηνυμάτων στο χρήστη και ο διάλογος εισαγωγής δεδομένων.
Ο πρώτος εμφανίζεται καλώντας τη μέθοδο **`showMessageDialog()`** του αντικειμένου **`JOptionPane`**, και έχει τη μορφή του σχήματος 9 (σημ: τα αντικείμενα και οι μέθοδοι θα σας παρουσιαστούν λεπτομερώς στην ενότητα 4. Αν οι όροι αυτοί σας είναι άγνωστοι είναι φυσιολογικό και δε θα πρέπει να σας ανησυχεί).
Ανοίξτε το IDE της επιλογής σας και ορίστε ένα νέο project. Στο project αυτό δημιουργήστε μία νέα κλάση και ονομάστε την **`Example01`**. Στη συνέχεια πληκτρολογήστε τον ακόλουθο κώδικα, αποθηκεύστε το αρχείο και εκτελέστε το. Θα πρέπει να πάρετε την έξοδο του σχήματος 9.

```
import javax.swing.JOptionPane; // program uses JOptionPane

public class Example01 {
   public static void main(String args[]){
      JOptionPane.showMessageDialog(null,
                  "Welcome\nto\nJava\nProgramming!");

      System.exit(0);  // terminate application with window
   }
}
```