ΤΕΙ ΠΑΤΡΑΣ
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΕΠΙΧΕΙΡΗΜΑΤΙΚΟΥ ΣΧΕΔΙΑΣΜΟΥ ΚΑΙ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΕΚΠΑΙΔΕΥΤΙΚΟΥ ΥΛΙΚΟΥ ΓΙΑ ΤΙΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΟΡΓΑΝΩΣΗ ΑΡΧΕΙΩΝ – ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JAVA

ΣΠΟΥΔΑΣΤΡΙΕΣ:
ΑΒΡΑΑΜ ΜΑΡΙΑ
ΔΗΜΗΤΡΟΥΚΑ ΜΑΡΙΑ
ΜΠΕΡΤΑΚΗ ΔΗΜΗΤΡΑ

ΕΙΣΗΓΗΤΗΣ:
Δρ ΜΠΑΚΑΛΗΣ ΑΡΙΣΤΕΙΔΗΣ

ΠΑΤΡΑ 2008

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Όταν αναλάβαμε να διεκπεραιώσουμε το θέμα της ανάπτυξης εκπαιδευτικού υλικού για τη γλώσσα προγραμματισμού Java, στα πλαίσια του μαθήματος των Δομών Δεδομένων και Οργάνωσης Αρχείων, αναλάβαμε και τη μεγάλη πρόκληση της εφαρμογής μιας καινοτομικής ιδέας, πάνω στην οποία οι γνώσεις μας ήταν από ανεπαρκείς έως ανύπαρκτες. Περάσαμε ατελείωτες ώρες μελετώντας τη Java και προσπαθώντας να βρούμε τρόπους για την υλοποίηση αυτού του θέματος. Το μεγαλύτερο στοίχημα ήταν ότι έπρεπε να παρουσιάσουμε με τον απλούστερο δυνατό τρόπο εξειδικευμένους όρους και ορισμούς ώστε να κάνουμε τη Java προσιτή σε όλους. Ελπίζουμε το αποτέλεσμα να είναι ικανοποιητικό, όχι μόνο για τους αρχάριους αλλά και για τους γνώστες της γλώσσας προγραμματισμού Java.

Σκοπός μας είναι το εγχειρίδιο αυτό να αποτελέσει ένα χρήσιμο εργαλείο στα χέρια οποιουδήποτε έρχεται για πρώτη φορά σε επαφή με τη Java. Για το λόγο αυτό, προσπαθήσαμε να χρησιμοποιήσουμε όσο το δυνατόν απλές, κατανοητές εκφράσεις, προσθέτοντας και ένα λεξιλόγιο στο τέλος της εργασίας μας, στο οποίο επεξηγούνται άγνωστοι όροι.

Για την προσπάθεια αυτή οφείλουμε να ευχαριστήσουμε τον καθηγητή και εισηγητή της πτυχιακής μας κύριο Μπακάλη Αριστείδη, για τις συμβουλές και την βοήθεια του, καθώς και όσους μας συμπαραστάθηκαν και μας βοήθησαν στην προσπάθεια μας αυτή.

ΕΙΣΑΓΩΓΗ

Ξεκινώντας κάποιος να ασχοληθεί με τη Java (ελληνιστί Ιάβα) θα κάνει την εξής απλή ερώτηση: Τι είναι η Java;

Κατά πρώτον η Java είναι ένα νησί της Ινδονησίας και κατά δεύτερον είναι μια γλώσσα προγραμματισμού που σχεδιάστηκε τη δεκαετία του ’90, ώστε να μπορεί ο κώδικας της να εκτελείται σε οποιοδήποτε μηχάνημα, σε ότι πλατφόρμα και αν τρέχει, χωρίς τροποποιήσεις. Διακρίνεται από το χαρακτηριστικό "WORA" (Write Once, Run Anywhere).

Επιπλέον, η Java είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την Sun Microsystems για να χρησιμοποιείται σε δίκτυα και ειδικά στο World Wide Web, δηλαδή στο Internet. Έχει αλλάξει διάφορες μορφές και ονομασίες στην διάρκεια του χρόνου μέχρι να φτάσει στην σημερινή της μορφή. Το γεγονός ότι χρησιμοποιεί αντικείμενα την καθιστά πιο εύχρηστη από τις υπόλοιπες γλώσσες προγραμματισμού και γι’ αυτό είναι πιο διαδεδομένη.

Γενικά, η Java είναι μια ασφαλής και ισχυρή γλώσσα προγραμματισμού με πολλές δυνατότητες και μεγάλο λεξιλόγιο. Απαιτείται αρκετός χρόνος για να ασχοληθεί κάποιος επαγγελματικά με αυτήν την γλώσσα λόγω του μεγέθους της. Υπάρχει πάντως μια βασική δομή, η οποία αν κατανοηθεί πλήρως, διευκολύνει σε μεγάλο βαθμό την γρήγορη ανάπτυξη αξιόπιστων εφαρμογών.

Η Java προκάλεσε ίσως το μεγαλύτερο ενδιαφέρον σε σύγκριση με οποιαδήποτε άλλη εξέλιξη στον κόσμο του Internet. Είναι η πρώτη που κατάφερε να συμπεριλάβει ήχο και κίνηση σε μια ιστοσελίδα. Επιτρέπει στους χρήστες να αλληλεπιδρούν (interact) με την ιστοσελίδα. Εκτός από το να διαβάζει απλά και ίσως να συμπληρώνει μία φόρμα, ο χρήστης μπορεί να παίξει παιχνίδια, να συνομιλήσει, να λαμβάνει συνεχώς τις πιο πρόσφατες πληροφορίες και πολλά άλλα.

Μερικές από τις δυνατότητες της Java:

- Ήχος ο οποίος εκτελείται όποτε ο χρήστης φορτώνει μία σελίδα
- Μουσική που παίζει στο background μιας σελίδας

- Δημιουργία κινουμένων σχεδίων
- Βίντεο
- Παιχνίδια με πολυμέσα

Ο κώδικας σε Java μπορεί να εισαχθεί και να εκτελεστεί, χωρίς καμία τροποποίηση, σε έναν PC που χρησιμοποιεί είτε MS-DOS ή Windows, σε Macintosh αλλά και σε μηχανή Unix.
Στο εγχειρίδιο αυτό θα βρείτε πλούσιο και εικονογραφημένο υλικό για τη Java δομημένο και προσαρμοσμένο στις ανάγκες του αρχάριου αναγνώστη. Ξεκινώντας από το που θα βρείτε τη Java μέχρι τη δημιουργία εμπλουτισμένων προγραμμάτων, το εγχειρίδιο αυτό θα σας οδηγήσει βήμα βήμα στην κατανόηση και χρήση αυτής της γλώσσας προγραμματισμού.

Η δομή του εγχειριδίου είναι τέτοια που να βοηθάει τον αναγνώστη στην καλύτερη δυνατή κατανόηση του. Πριν από κάθε κεφάλαιο υπάρχει ένας συνοπτικός πρόλογος που προετοιμάζει τον αναγνώστη για το τι πρόκειται να διαβάσει με σχόλια και οδηγίες για το κεφάλαιο που ακολουθεί. Η ύλη είναι χωρισμένη σε κεφάλαια ανάλογα το θέμα τους, με πολλά παραδείγματα και ασκήσεις. Στο τέλος κάθε κεφαλαίου υπάρχουν επαναληπτικές ερωτήσεις και ασκήσεις για την αυτοαξιολόγηση του αναγνώστη.

Στις τελευταίες σελίδες του εγχειριδίου υπάρχουν πολλές συγκεντρωτικές σελίδες με παραδείγματα και ασκήσεις οι οποίες αποτελούν συμπλήρωμα της ύλης που υπάρχει σε κάθε κεφάλαιο. Σε αυτές τις σελίδες μπορείτε να ανατρέξετε για να βρείτε επιπλέον παραδείγματα αλλά και ασκήσεις για να εφαρμόσετε τις γνώσεις σας. Ακόμα θα βρείτε και ένα λεξιλόγιο ελληνικών αλλά και ξένων όρων, για την ερμηνεία της ορολογίας της Java.

Ελπίζουμε το εγχειρίδιο αυτό να γίνει το δεξί σας χέρι. Καλή ανάγνωση!

ΑΒΡΑΑΜ ΜΑΡΙΑ
ΔΗΜΗΤΡΟΥΚΑ ΜΑΡΙΑ
ΜΠΕΡΤΑΚΗ ΔΗΜΗΤΡΑ

Σε αυτό το κεφάλαιο θα κάνουμε μια θεωρητική εισαγωγή στη Java. Αρχικά, θα «ρίξουμε» μια ματιά στην ιστορία της Java, παρακολουθώντας βήμα βήμα την πορεία της ομάδας Green και της Sun Microsystems στη υλοποίηση καινοτομικών ιδεών που άλλαξαν την ιστορία της τεχνολογίας. Θα μάθουμε πως γεννήθηκε η ιδέα του αντικειμενοστρεφούς προγραμματισμού και πως φτάσαμε από την Oak στη σημερινή μορφή της Java.

Στη συνέχεια θα κάνουμε μια θεωρητική προσέγγιση στον τρόπο λειτουργίας, τα χαρακτηριστικά, τα εργαλεία και τις εφαρμογές της Java.

Αυτό το κεφάλαιο είναι ο πρώτος κρίκος της αλυσίδας της Java. Γι΄ αυτό θα πρέπει να δωθεί η απαραίτητη σημασία για να κατανοηθούν κάποιοι όροι που θα τους συναντάμε συχνά από δω και πέρα.

ΚΕΦΑΛΑΙΟ 1

1) Η ΙΣΤΟΡΙΑ ΤΗΣ JAVA

Η Java παρουσιάστηκε όταν μία ομάδα ερευνητών, η οποία προσπαθούσε να αναπτύξει ενσωματωμένο λογισμικό (embedded software) για έξυπνες καταναλωτικές συσκευές, διαπίστωσε ότι οι ήδη υπάρχουσες γλώσσες C και C++ δεν ανταποκρίνονταν στις απαιτήσεις της.

Το 1990, ο μηχανικός λογισμικού της Sun Microsystems Patric Naughton που ασχολούταν εκείνη την εποχή με την υποστήριξη όλων των διαφορετικών APIs (Abstract Programming Interfaces) που είχε στη διάθεσή της η Sun Microsystems, σκεφτόταν να μετακινηθεί στην ανταγωνιστική εταιρία NeXT μετά από αντίστοιχη πρόταση. Όταν ανακοινώθηκε η απόφασή του για αποχώρηση από τη Sun στον τότε γενικό διευθυντή της Scott McNealy, και φίλο του Naughton, αυτός δεν τη δέχθηκε και του έδωσε την ευκαιρία να φτιάξει μία λίστα με τα παράπονα του για την εταιρία, καθώς επίσης και τις προτάσεις του σχετικά με το τι ήταν αναγκαίο να γίνει.

Ο Naughton έφτιαξε αυτή τη λίστα με το σκεπτικό ότι τίποτα από αυτά που θα παρατηρούσε και θα πρότεινε δεν θα ενδιέφερε το διευθυντή του. Στο γράμμα που έστειλε στο γενικό διευθυντή αργότερα, υπογράμμιζε πως αν η εταιρία αποφασίσει να προγραμματίσει την τακτική της στο μέλλον, πολλοί προγραμματιστές που απασχολούνταν για την υποστήριξη των Windows Systems (X-Windows, Motif κλπ.) θα μπορούσαν να χρησιμοποιηθούν πιο αποδοτικά. Οι προτάσεις του Naughton ήταν αντικειμενικές και γι΄ αυτό το λόγο έκαναν εντύπωση. Το γράμμα αυτό, συζητήθηκε από πολλούς μηχανικούς λογισμικού που εργάζονταν στην εταιρία και οι οποίοι υποστήριξαν φανερά τον Naughton και τις ιδέες του. Η υποστήριξη αυτή των συναδέλφων του, έκανε τους διευθυντές της εταιρίας να λάβουν σοβαρά υπόψιν τους τις ιδέες του.

Τελικά η εταιρία πρότεινε στον Naughton να δημιουργήσει μία ομάδα έρευνας από μηχανικούς και προγραμματιστές για να ασχοληθούν με την εφεύρεση μιας καινοτομικής ιδέας πάνω σε οποιοδήποτε τομέα επιθυμούσαν.

Η ομάδα του Naughton τελικά δημιουργήθηκε με την ονομασία Green και απέκτησε απόλυτη ανεξαρτησία κινήσεων. Τα πρώτα βήματα της ομάδας Green κινήθηκαν γύρω από τις παιχνιδομηχανές που κυκλοφορούσαν τότε στο εμπόριο και είχαν γνωρίσει τεράστια επιτυχία όπως τα Nintendo Game Boy και Sega. Το πρόβλημα που εμφάνιζαν τότε οι μηχανές αυτές ήταν οι περιορισμένες δυνατότητες επέκτασης τους. Το περιορισμένο εύρος εφαρμογών των επεξεργαστών τους, δεν επέτρεπε την ανάπτυξη πρωτοποριακών ιδεών. Έτσι, το στοίχημα της ομάδας Green ήταν να δημιουργήσει κάτι καινοτομικό στο χώρο των παιχνιδομηχανών.

Η Sun παρέδωσε κι άλλα προνόμια στην ομάδα Green μετατρέποντας την σε αυτόνομη εταιρία, θυγατρική της Sun, με την επωνυμία First Person. Η νεοσυσταθείσα εταιρία είχε πολλές νέες ιδέες αλλά δεν ήξερε ακόμη πώς να τις χρησιμοποιήσει για να κάνει ένα άνοιγμα προς την αγορά. Μία ιδέα ήταν η ανερχόμενη τεχνολογία της αλληλεπιδραστικής τηλεόρασης (interactive television), η οποία όμως δεν καρποφόρησε κυρίως λόγω λανθασμένων διοικητικών χειρισμών σχετικά με τα πνευματικά δικαιώματα, οπότε και εγκαταλείφθηκε η ιδέα.

Την ίδια εποχή στα εργαστήρια της Sun γεννιόταν μία νέα γλώσσα προγραμματισμού που ο James Gosling, προϊστάμενος του Naughton, ονόμασε Oak (oak=βελανιδιά), εμπνευσμένος από τη βελανιδιά που έβλεπε έξω από το παράθυρό του. Αυτή η γλώσσα προγραμματισμού αποσκοπούσε στο να χρησιμοποιήσει τα χαρακτηριστικά όλων των παιχνιδομηχανών χειρός και να δημιουργήσει μία νέα, ενιαία πλατφόρμα υλοποίησης προγραμμάτων.

Καθώς η υλοποίηση της γλώσσας Oak συνεχίζονταν, η ομάδα Green συνέχιζε ανεξάρτητα την εκτεταμένη έρευνα για την προσέλκυση του κοινού προς τα ηλεκτρονικά παιχνίδια και τον τρόπο που αλληλεπιδρούσαν με τα μηχανήματα.

Τελικά, μετά από πολλές έρευνες η ομάδα δημιούργησε μία μικρή, κινητή συσκευή (ακριβώς όπως και το Gameboy) με το κωδικό όνομα “*7” (star seven). Το *7 παρουσίαζε έναν κινούμενο χαρακτήρα που σχεδιάστηκε από τον Joe Palrang και ονομάστηκε Duke, ο οποίος βοηθούσε τους χρήστες να εξερευνήσουν μία απλή στη χρήση αλλά πλούσια σε γραφικά διεπαφή. Η ιδέα πάνω από στην οποία κινήθηκε η υλοποίηση ήταν ότι μία διεπαφή χρήστη θα πρέπει να είναι φιλική και να ψυχαγωγεί

ταυτόχρονα το χρήστη πάνω σε μία εύκολα μετακινούμενη, προσωπική συσκευή. Ο Duke αργότερα θα γινόταν η μασκότ και το σήμα κατατεθέν της Java.

*Η αρχική συσκευή του *7*

*Μία οθόνη του *7, που δείχνει μία πειραματική διεπαφή χρήστη*

*Ο Duke, ο χαρακτήρας που σχεδιάστηκε αρχικά για το *7 και κατέληξε μασκότ της Java*

Έτσι, τον Αύγουστο του 1991 εμφανίζεται μια νέα αντικειμενοστραφής γλώσσα με το όνομα Oak (Object Application Kernel), η οποία προστίθεται στην λίστα των γλωσσών προγραμματισμού που προσφέρουν μόνο ουσιαστική υποστήριξη σε εφαρμογές τύπου πελάτη-εξυπηρετητή (client-server).

Τον Απρίλιο του 1993 η γλώσσα κάνει τα πρώτα της βήματα στον χώρο του διαδικτύου με πολύ θετικά αποτελέσματα. Το περιστατικό αυτό παρακινεί την Sun στην χρηματοδότηση για την ανάπτυξη της γλώσσας, αν και τον Αύγουστο του προηγούμενου έτους η πρώτη προσπάθεια για την πώλησή της καταλήγει σε αποτυχία.

Παράλληλα, το Εθνικό Κέντρο για Εφαρμογές Υπερυπολογιστών (National Center for Supercomputing Applications) εισήγαγε στην αγορά το Mosaic, την πρώτη εφαρμογή πλοήγησης (browser) για τον Παγκόσμιο Ιστό (Web). Η συνέχεια, ήταν η καταιγιστική νέα τεχνολογία του Παγκόσμιου Ιστού η οποία αύξανε την κίνηση στο Διαδίκτυο (Internet) όλο και περισσότερο. Με δεδομένη αυτή την κατεύθυνση της αγοράς προς

τα δίκτυα, η First Person αποφάσισε στις αρχές του 1994 να επικεντρώσει τις προσπάθειές της προς τις εφαρμογές πολυμέσων συνεχούς σύνδεσης (online multimedia).

Τη στγμή που η γλώσσα Oak ήταν έτοιμη να παρουσιαστεί ήρθε η ιδέα να δημιουργηθεί ένα λειτουργικό σύστημα γραμμένο σε Oak. Πραγματικά, η κίνηση αυτή έγινε και στη συνέχεια ο Naughton πρότεινε την ελεύθερη διάθεση του κώδικα στο Internet, ιδέα που επίσης υιοθετήθηκε με τη δημιουργία ενός μεταφραστή για Oak γραμμένο στην ίδια τη γλώσσα. Για την υποστήριξη του νέου λειτουργικού συστήματος, ο Naughton, συνεργάστηκε με την ομάδα ανάπτυξης της Oak, δημιούργησαν την πρώτη εφαρμογή πλοήγησης για τον Παγκόσμιο ιστό γραμμένη σε Oak, και την ονόμασαν WebRunner. Το πρώτο applet στην ιστορία δημιουργήθηκε τότε και παρουσίαζε τον Duke να χαιρετά τους δημιουργούς του.

Η Sun υποστήριξε την απόφαση να μοιραστεί ελεύθερα ο κώδικας της Oak στο Internet αλλά αφού πρώτα άλλαξε το όνομά της σε Java για περισσότερο εμπορικούς λόγους. Άλλα ονόματα που είχαν προταθεί για τη Java ήταν Neon, Lyric, Pepper και Silk.

Στα μέσα του 1994 αναπτύσσεται το πρώτο δοκιμαστικό πρόγραμμα πλοήγησης με Java υπό το όνομα WebRunner (η πρώτη έκδοση του προγράμματος πλοήγησης HotJava της Sun), ενώ το φθινόπωρο του ίδιου έτους ο Van Hoff πραγματοποιεί με Java τον πρώτο Java διερμηνευτή (ο προηγούμενος Java διερμηνευτής είχε αναλυθεί από τον Gosling σε C).

Τον επόμενο χειμώνα, τον Ιανουάριο του 1995, η γλώσσα παίρνει την σημερινή της ονομασία και η πρώτη έγκυρη και αυθεντική τεκμηρίωση εμφανίζεται με την μορφή ενός ‘white paper’ [Sun 95]. Τον Μάιο του 1995 η Sun παρουσιάζει επίσημα την Java και το HotJava. Παράλληλα, η Netscape αγοράζει άδεια χρήσης της Java και περιλαμβάνει την γλώσσα στην δεύτερη έκδοση του προγράμματος πλοήγησης Netscape. Η πορεία της γλώσσας πλέον είναι ανοδική αφού, όλοι οι μεγάλοι κατασκευαστές λογισμικού γνωστοποιούν την τελική τους κρίση να μεταχειριστούν την Java, με κυριότερη την απόφαση της Microsoft τον Δεκέμβριο του 1995. Έτσι,

γίνεται φανερό ότι η Java είναι η γλώσσα προγραμματισμού που προηγείται και θα προηγείται για αρκετά χρόνια ακόμα.

Διαφημιστικό του WebRunner

Το μόνο που χρειαζόταν πλέον η τεράστια κοινότητα του Διαδικτύου ήταν μία πλατφόρμα που να τρέχει τις εφαρμογές που έχουν γραφτεί σε Java. Αυτή δεν ήταν άλλη από τον WebRunner ο οποίος επίσης μετονομάστηκε σε HotJava λόγω νομικών προβλημάτων με τα κατατεθέντα σήματα. Η συνέχεια ήρθε με την υποστήριξη από τη Netscape της Java στη δημοφιλή εφαρμογή πλοήγησής της (Navigator).

Από τότε μέχρι σήμερα η Java έχει επεκταθεί πάρα πολύ, κάνοντας τον Duke μία από τις πιο δημοφιλείς φυσιογνωμίες στο Διαδίκτυο. Από την εποχή εκείνη έχουν συμβεί πάρα πολλά στο Διαδίκτυο και στην τεχνολογία για επιχειρήσεις, και αυτά είναι μόνο η αρχή: το JDK™, το sandbox, τα applets, χιλιάδες προσπάθειες προσανατολισμένες σε τεχνολογίες Java, η αρχιτεκτονική JavaBeans™, το Java Studio™, το Netscape Communicator, χιλιάδες παροχείς υπηρεσιών Διαδικτύου, 60 εκατομμύρια χρήστες του Διαδικτύου, 56K και καλωδιακά modems, το ηλεκτρονικό εμπόριο, τα servlets, οι Java Foundation Classes, η ψηφίδες Enterprise JavaBeans™, το JavaOS for Business™, και δεσμεύσεις από μεγάλες εταιρίες όπως η IBM.

Τα παραπάνω είναι κάτι περισσότερο από έναν απλό κατάλογο προϊόντων λογισμικού και γεγονότων. Είναι μία ένδειξη του πόσο γρήγορα αναπτύχθηκε αυτή η τεχνολογία κατά τη διάρκεια των τελευταίων ετών. Η τεχνολογία της Java είναι ένα

από τα σπουδαιότερα υπολογιστικά περιβάλλοντα αυτή τη στιγμή, ένας ποταμός που έχει πλημμυρίσει τις όχθες του και κατευθύνεται τώρα προς οποιεσδήποτε βιομηχανίες ή επιχειρήσεις.

Στους μήνες που ακολούθησαν την αρχική ανακοίνωση της Sun, η τιμή της μετοχής της ανέβηκε στα ύψη, πράγμα για το οποίο εν μέρει ευθύνεται και η τεχνολογία Java. Κατά το τέλος του πρώτου πλήρους χρόνου τεχνολογίας Java, το τμήμα JavaSoft είχε παραχωρήσει 38 άδειες και είχε προσελκύσει 6.000 άτομα που ασχολούνται με την ανάπτυξη λογισμικού στην εμπορική έκθεσή της, το πρώτο συνέδριο JavaOne Developer Conference™. Κατά το τέλος του δευτέρου χρόνου, η JavaSoft είχε παραχωρήσει σχεδόν 100 άδειες και το συνέδριο JavaOne προσέλκυε 10.000 άτομα.

Στον εορτασμό των τρίτων γενεθλίων της τεχνολογίας Java, το μετονομασμένο τμήμα Java της Sun απασχολούσε 800 άτομα, υποστηριζόταν από 150 άδειες και πολλές εκατοντάδες χιλιάδων μηχανικών λογισμικού παγκοσμίως. Κατάφερε να φιλοξενήσει 15.000 άτομα στο JavaOne (η μεγαλύτερη σύναξη προγραμματιστών που έγινε ποτέ παγκοσμίως), κατάφερε να υπερκεράσει τον προηγούμενο ετήσιο στόχο της για 10.000 downloads του JDK σε μία μέρα, και υπερηφανευόταν για μία ιδιόκτητη πολυάσχολη γραμμή T3.

2) ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ JAVA

Η διαφοροποίηση της Java έγκειται στα ιδιαίτερα χαρακτηριστικά της ως προς τη μορφή, τη διαδραστικότητα της με το χρήστη, την αποτελεσματικότητα και τις πολλαπλές εφαρμογές της. Τα βασικά χαρακτηριστικά της είναι τα εξής:

Α) **Απλή**: Με το χαρακτηρισμό απλή εννοούμε τη χρήση λιγότερων χαρακτήρων, που θα κάνουν τη χρήση της Java πιο εύκολη. Στην προσπάθεια δημιουργίας της Java λήφθηκε σοβαρά υπόψη η απλότητα. Η νέα γλώσσα δε θα έπρεπε να αγνοεί την πραγματικότητα στον προγραμματισμό: οι περισσότεροι προγραμματιστές γνώριζαν και χρησιμοποιούσαν τη C ενώ όσοι ενδιαφέρονταν για αντικειμενοστραφείς εφαρμογές, τη C++. Δε θα ήταν επομένως ώριμο να εισάγουν μία εντελώς νέα γλώσσα χωρίς καμία σχέση με αυτό που γνώριζαν πολύ καλά οι προγραμματιστές.

Ωστόσο, καταβλήθηκε προσπάθεια ώστε η Java να απαλλαγεί από χαρακτηριστικά των C/C++ που είτε δε χρησιμοποιούταν ευρέως, είτε περισσότερο δυσκόλευαν τον προγραμματιστή, παρά του πρόσφεραν επιπλέον δυνατότητες. Τέτοια χαρακτηριστικά είναι, ενδεικτικά, η υπερφόρτωση τελεστών (operator overloading) και η πολλαπλή κληρονομικότητα (multiple inheritance) από τη C++.

Επιπλέον το κυριότερο πρόβλημα της C είναι η αποδοτική διαχείριση της μνήμης. Η δέσμευση ή αποδέσμευση μνήμης είναι μία ιδιαίτερα επίπονη διαδικασία που αποσπά τον προγραμματιστή από το σκοπό του, απασχολώντας τον με ζητήματα ρουτίνας. Η Java για το λόγο αυτό εισήγαγε την αυτόματη συλλογή «απορριμμάτων» (garbage collection), δηλαδή την αυτόματη αποδέσμευση μνήμης στην οποία δεν υπάρχουν αναφορές (referenced). Πλέον ο προγραμματιστής απαλλάσσεται από την ανάγκη να γνωρίζει και να ασχολείται με τη διαχείριση της μνήμης στην πλατφόρμα στην οποία εργάζεται.

Τέλος, ένα άλλο χαρακτηριστικό που πηγάζει από την απλότητα αφορά στο μέγεθος. Ένας από τους στόχους δημιουργίας της Java ήταν να μπορεί να τρέχει σε μικρές μηχανές χειρός. Έτσι, οι βιβλιοθήκες της καθώς και όλα τα προγράμματα γραμμένα σε αυτή είναι αρκετά μικρά, πράγμα που τις δίνει μεγάλο πλεονέκτημα σε ένα ανομοιογενές δικτυακό περιβάλλον στο οποίο έχει σημασία ο χρόνος που απαιτείται προκειμένου να «κατέβει» η εφαρμογή.

Β) **Αντικειμενοστραφής**: Τα δεδομένα και οι μέθοδοι συνεργάζονται για να αναδείξουν αντικείμενα που θα έχουν και δεδομένα και μεθόδους. Η Java ακολουθεί τη παράδοση της Smalltalk, μια γλώσσα που απαιτεί τη χρήση αντικειμένων και μόνο, σε αντίθεση με άλλες που επιτρέπουν στον προγραμματιστή να ελιχθεί χρησιμοποιώντας πιο κλασσικές μεθόδους στο ίδιο πρόγραμμα. Η έννοια του αντικειμενοστραφούς (object oriented) είναι από τις πιο πολυχρησιμοποιημένες στη βιομηχανία παραγωγής λογισμικού. Εντούτοις, αντιπροσωπεύει μία από τις δυναμικότερες έννοιες στον προγραμματισμό, υλοποιώντας καθαρά και αποδοτικά τις διεπαφές και τις επαναχρησιμοποιούμενες ψηφίδες (components) λογισμικού.

Ο αντικειμενοστραφής σχεδιασμός είναι γενικότερα μία τεχνική που επικεντρώνει στο σχεδιασμό με αντικείμενα – δεδομένα (objects) και στη διεπαφή αυτών προς τα έξω

παρά με τα εργαλεία που θα χρησιμοποιηθούν για τη δημιουργία τους. Είναι περισσότερο ενδιαφέρον, δηλαδή, οι ιδιότητες ενός αντικειμένου και πώς αυτό αλληλεπιδρά με άλλα αντικείμενα στον ίδιο ή σε κάποιο άλλο χώρο (πρόγραμμα ή πλατφόρμα). Είναι ουσιαστικά αυτό που κάνει η τεχνολογία plug-and-play.

Η ιδέα για την υλοποίηση των αντικειμενοστραφών χαρακτηριστικών της Java προήλθαν ουσιαστικά αυτούσια από τη C++, ενώ χρησιμοποιήθηκαν και επεκτάσεις από την Objective C για τη δυναμικότερη ανάλυση των μεθόδων-συναρτήσεων (dynamic method resolution).

Γ***)*** **Δυναμική** : Ένα αυτοδύναμο πρόγραμμα πρέπει να μην είναι απρόβλεπτο και να μην αποτυγχάνει σε σφάλματα του προγραμματιστή. Ο προγραμματιστής εμφανίζει προβλήματα διαχείρισης της μνήμης, τα οποία η Java προσπαθεί να επιλύει γρήγορα. Η Java είναι κατά πολλές έννοιες πιο δυναμική από ότι η C και η C++. Σχεδιάστηκε ώστε να προσαρμόζεται εύκολα σε ένα διαρκώς εξελισσόμενο περιβάλλον. Για παράδειγμα ας υποθέσουμε ότι δύο εταιρίες παράγουν λογισμικό σε C++.

Έστω, ότι η Α εταιρία παράγει μία βιβλιοθήκη κλάσεων την οποία χρησιμοποιεί η εταιρία Β για την παραγωγή του τελικού προϊόντος. Αν κάποια στιγμή η Α αποφασίσει να βελτιώσει τη βιβλιοθήκη της, τότε η Β θα πρέπει να πάρει να ανανεωμένη βιβλιοθήκη και να την χρησιμοποιήσει για να κάνει εκ νέου μετάφραση στο λογισμικό της και να το διανείμει πάλι στην αγορά. Με άλλα λόγια δεν είναι άμεσα χρησιμοποιήσιμη η νέα βιβλιοθήκη, πράγμα που έρχεται σε αντίθεση με την έννοια του πραγματικού αντικειμενοστραφούς προγραμματισμού.

Αυτό που κάνει η Java, είναι να διασυνδέει τα επιμέρους κομμάτια κατά τη διάρκεια της εκτέλεσης. Οι βιβλιοθήκες μπορούν να περιέχουν ακόμα και νέες εντελώς μεθόδους χωρίς καμία επίπτωση σε προγράμματα που έχουν ήδη διανεμηθεί.
Μία διεπαφή (interface) είναι ένα σύνολο μεθόδων (το αντίστοιχο των συναρτήσεων στη C) που ένα αντικείμενο πρέπει να περιέχει αλλά δεν το ενδιαφέρει η υλοποίησή τους. Με αυτόν τον τρόπο μία κλάση που υλοποιεί (implements) μία διεπαφή, είναι σίγουρο ότι υλοποιεί και τις αντίστοιχες μεθόδους, που υπάρχουν σε αυτό. Σε αντίθεση με τις διεπαφές, η κληρονομικότητα (inheritance) περνάει στις κλάσεις –

παιδιά πέρα από το σύνολο των μεθόδων και τις υλοποιήσεις τους στην κλάση – πατέρα (superclass). Η Java υποστηρίζει πολλαπλές διεπαφές αλλά απλή κληρονομικότητα στις κλάσεις της. Κατά συνέπεια, μία κλάση μπορεί να κληρονομεί τις μεθόδους μόνο ενός πατέρα, σε αντίθεση με τη C++, η οποία επιτρέπει πολλαπλή κληρονομικότητα, αλλά μπορεί να υλοποιεί οσεσδήποτε διεπαφές. Οι διεπαφές εξασφαλίζουν μεγαλύτερη ευελιξία, και επαναχρησιμοποίηση αφού διασυνδέουν αντικείμενα ανάλογα με το τι πρέπει να κάνουν ή να υλοποιούν, παρά με τον πώς τα υλοποιούν.

Επιπρόσθετα, οι κλάσεις κατά τη φάση της εκτέλεσης πάντα έχουν την αντιπροσώπευσή τους (representation). Υπάρχει μία κλάση και ένα ή περισσότερα στιγμιότυπά (instances) της που περιέχουν πληροφορίες της εκτέλεσης. Αν προσπαθήσει κανείς να αποκτήσει ένα δείκτη σε αυτό το στιγμιότυπο είναι πάντα γνωστός ο τύπος του αντικειμένου στο οποίο δείχνει. Αντίθετα με τη C++, όπου ο προγραμματιστής δεν ξέρει αυτόν τον τύπο και κάνει αυθαίρετα προσαρμογή τύπου (type cast), χωρίς καμία εγγύηση ότι είναι και το σωστό, η Java κάνει πάντα έλεγχο τύπων κατά τη μεταγλώττιση και την εκτέλεση, έτσι ώστε να αποφευχθούν τέτοιες προβληματικές καταστάσεις. Η διαφορά μεταξύ της Java και της C++, όσον αφορά στο χαρακτηριστικό των τύπων, είναι ότι στη μεν πρώτη ο χρήσης εμπιστεύεται το μεταφραστή ότι δεν θα περάσει κανένα λάθος τύπων, στη δε δεύτερη, ο μεταφραστής εμπιστεύεται τον προγραμματιστή ότι δε θα του περάσει λάθος προσαρμογή τύπου.

Δ) **Ασφαλής**: Τα συστήματα ασφαλείας της Java εξασφαλίζουν ότι ο κώδικας που έχει γραφτεί μια φορά στη Java, δεν είναι εύκολο να παραποιηθεί. Ακόμα έχει σχεδιαστεί για να ελαχιστοποιεί την πιθανότητα προσβολής του συστήματος. Ένας από τους στόχους της Java είναι να μπορεί να τρέχει πάνω από δικτυακά και κατανεμημένα περιβάλλοντα. Με τα γνωστά προβλήματα ασφαλείας στα δίκτυα, θα έπρεπε να ενσωματωθούν κάποιες τεχνικές που θα εξασφάλιζαν την ασφάλεια των προγραμμάτων και των δεδομένων τους στο δίκτυο.

Πραγματικά, η Java εφοδιάστηκε με έναν ισχυρό μηχανισμό ελέγχου. Όταν ένα applet, δηλαδή μία Java εφαρμογή, φθάνει στον προορισμό της, γίνεται τοπικά έλεγχος για να διαπιστωθεί αν πραγματικά είναι η εφαρμογή που ζητήθηκε κι αν έχει

όλα τα απαιτούμενα διακριτικά γνωρίσματα. Οι τεχνικές ελέγχου γνησιότητας βασίζονται στη μέθοδο κρυπτογράφησης δημοσίου κλειδιού (public key encryption).

Όσον αφορά στο θέμα της ασφάλειας, αξίζει να σημειωθούν και τα χαρακτηριστικά που αναφέρθηκαν και σε προηγούμενη παράγραφο. Με την απαγόρευση της πρόσβασης άλλων αντικειμένων στους πόρους ενός αντικειμένου, ουσιαστικά εξαλείφεται ο κίνδυνος των ιών και των αλλοιωμένων δεδομένων (corrupted data).

Ε) **Ουδέτερη αρχιτεκτονική**: Ο κώδικας της Java μπορεί να γραφτεί και να τρέξει σε οποιοδήποτε υπολογιστή, χωρίς να υπάρξουν διαφορές. Η Java σχεδιάστηκε για να υποστηρίζει εφαρμογές πάνω από δίκτυο αλλά ως γνωστό ένα δίκτυο περιλαμβάνει ένα μεγάλο σύνολο διαφορετικών μηχανών και λειτουργικών συστημάτων. Για να εξασφαλιστεί το ότι η ίδια εφαρμογή θα τρέχει ομοιογενώς σε αυτό το περιβάλλον, ο μεταγλωττιστής παράγει έναν ανεξάρτητο μηχανής αντικείμενο κώδικα (object code), ο οποίος με τη βοήθεια του Συστήματος Εκτέλεσης Java (Java Runtime System – JRE ή Java Virtual Machine – JVM) μπορεί να τρέξει σε πολλούς διαφορετικούς επεξεργαστές.

Το παραπάνω γεγονός όμως δεν είναι μόνο χρήσιμο στην περίπτωση των δικτύων και των δικτυακών εφαρμογών αλλά και στις τοπικές εφαρμογές. Μέχρι τώρα, με την μεγάλη ποικιλία επεξεργαστών και μηχανημάτων που κυκλοφορούν στην αγορά, οι κατασκευαστές πακέτων λογισμικού είναι υποχρεωμένοι από τους νόμους της αγοράς να παράγουν το ίδιο προϊόν για πολλές διαφορετικές πλατφόρμες, όπως για παράδειγμα για UNIX, IBM PC, Apple Macintosh. Όσο περισσότερο προχωρά η αγορά επεξεργαστών, τόσο πιο πολύ εντείνεται το πρόβλημα. Με τη Java, η φιλοσοφία είναι ότι η εφαρμογή γράφεται μία φορά σε οποιαδήποτε πλατφόρμα και έπειτα θα μπορεί να τρέχει οπουδήποτε (WORA – Write Once, Run Anywhere).

Ο μεταφραστής της Java παράγει εντολές bytecode, οι οποίες είναι εντολές που δεν έχουν να κάνουν με κάποια συγκεκριμένη μηχανή αλλά με την Εικονική Μηχανή Java (Java Virtual Machine – JVM). Στη συνέχεια καθώς εκτελείται ο κώδικας αυτός γίνεται και η μετατροπή σε κώδικα συγκεκριμένο για την κάθε μηχανή.

ΣΤ) **Κατανεμημένη**: Η Java μπορεί να δεχτεί δεδομένα όλων των τύπων και των μεγεθών. Ακόμα ένα Java πρόγραμμα είναι δυνατό να μεταφερθεί από το δίκτυο και να εκτελεστεί. Επίσης, μπορούν διαφορετικά κομμάτια του προγράμματος να έρθουν από διαφορετικές ιστοσελίδες.

Ζ) **Υψηλή απόδοση**: Τα προγράμματα της Java τρέχουν αργά. Χρησιμοποιεί Just In Time (JIT) συντάκτες (compilers) για να τρέχει πιο γρήγορα. Πέρα από την πολύ καλή απόδοση που επιτυγχάνεται με τη χρήση bytecodes, πολλές φορές είναι αναγκαία ακόμα υψηλότερη απόδοση. Για το λόγο αυτό, τα bytecodes μπορούν να μεταφράζονται σε εντολές μηχανής για τη συγκεκριμένη κεντρική μονάδα επεξεργασίας (Central Processing Unit – CPU), κατά τη μεταφορά και εκτέλεσή τους (on the fly).

Επιπλέον, επειδή η λογική του σχεδιασμού της παραγωγής εντολών μηχανής κατά αποδοτικό τρόπο ήταν πρωταρχική, η διαδικασία είναι εξαιρετικά απλή και αποδοτική. Μάλιστα, ο μεταφραστής κάνει αυτόματα δέσμευση καταχωρητών (register allocation) και βελτιστοποίηση κατά τη διάρκεια παραγωγής των bytecodes.

Σε μεταφράσιμο (interpreted) κώδικα, έχουμε περίπου 300.000 κλήσεις μεθόδων το δευτερόλεπτο σε ένα Sun SPARCStation 10. Έτσι η απόδοση των bytecodes που μετατρέπονται σε εντολές μηχανής είναι ασύγκριτη με αυτή των C/C++.

Η) **Πολυνηματικότητα (multithreading):** καλείται η τεχνική δημιουργίας εφαρμογών που βασίζονται στα πολλαπλά «νήματα» (threads). Ένα νήμα είναι μία ακολουθία εντολών η οποία μπορεί να εκτελείται παράλληλα με άλλες παρόμοιες. Είναι δηλαδή ένα είδος παραλληλισμού του κώδικα, ο οποίος παρ’ όλα αυτά μπορεί να τρέχει και σε έναν επεξεργαστή. Η δυσκολία του να γραφούν προγράμματα που να αντιμετωπίζουν καταστάσεις όπου πολλά πράγματα ταυτόχρονα πρέπει να συμβαίνουν, ανάγεται στη δυσκολία εξαγωγής ενός παράλληλου προγράμματος από το αντίστοιχο σειριακό, με τα αντίστοιχα προβλήματα συγχρονισμού, αδιεξόδων, αποτυχιών.

Η Java, παρ’ όλα αυτά, έχει ένα ισχυρό σύνολο αρχών συγχρονισμού που βασίζεται στο *monitor and condition variable paradigm* το οποίο αναφέρθηκε για πρώτη φορά

από τον *[Hoare, C.A.R., 1974]*. Με τη χρήση αυτών των τεχνικών μέσα στην ίδια τη γλώσσα, τα νήματα γίνονται ευκολότερα στη χρήση τους. Η όλη υλοποίηση βασίστηκε στο σύστημα Cedar/Mesa της Xerox.

Άλλα χαρακτηριστικά της πολυνηματικότητας είναι η καλύτερη αλληλεπίδραση πραγματικού χρόνου. Σε αυτόνομα περιβάλλοντα Java, η απόκριση προσεγγίζει πολύ τις απαιτήσεις απόκρισης πραγματικού χρόνου. Αν όμως αυτό το περιβάλλον βρίσκεται κάτω από ένα διαφορετικό λειτουργικό σύστημα (π.χ. UNIX, Windows, Macintosh) η απόδοση αυτή φυσικό είναι να παρουσιάζεται μειωμένη.

Θ) **Δημιουργεί ανεξάρτητες εφαρμογές και applets**. (τα applets είναι προγράμματα που βρίσκονται σε html ιστοσελίδες και εκτελούνται από τον Web Browser)

Ι) **Είναι διερμηνευτική γλώσσα**: Δηλαδή ο java compiler παράγει κώδικα σε μορφή ψευδοκώδικα (bytecode) το οποίο δεν τρέχει από μόνο του σε καμία μηχανή. Για να εκτελεστεί απαιτείται ένας διερμηνέας (interpreter) για να μετατρέψει τον ψευδοκώδικα σε πραγματικό, εκτελέσιμο κώδικα. Αυτό το χαρακτηριστικό δίνει τη δυνατότητα στον ψευδοκώδικα να μπορεί να τρέξει σε οποιοδήποτε μηχάνημα, με οποιοδήποτε λειτουργικό, αρκεί να έχει εγκατασταθεί ένας java interpreter. Ο bytecode κώδικας, δηλαδή το αποτέλεσμα της διαδικασίας της μεταγλώττισης, μεταφράζεται σε τοπικές εντολές μηχανής και δεν αποθηκεύεται πουθενά. Ο κώδικας αυτός μεταφράζεται στη συνέχεια (interpreted) με όλα τα γνωστά πλεονεκτήματα που έχει αυτό το γεγονός στην απόδοση του προγράμματος.

Η διασύνδεση (linking) που είναι η υπεύθυνη διαδικασία για τη σύνθεση των επιμέρους κομματιών (modules) του προγράμματος έχει μία σχετικά εύκολη δουλειά, με αποτέλεσμα να μην υπάρχει μεγάλος φόρτος στη φάση αυτή. Επιπρόσθετα, λόγω αυτής της μεταφράσιμης φύσης του κώδικα, η αποσφαλμάτωση (debugging) είναι μία σχετικά εύκολη διαδικασία, ακόμα και στο ανομοιογενές περιβάλλον ενός δικτύου.

Κ) **Υποστηρίζει εφαρμογές πολυμέσων (multimedia)**: Δηλαδή η Java παρέχει ευκολίες στη δημιουργία multimedia εφαρμογών. Αυτό επιτυγχάνεται με την ευελιξία της και με τις συνεχώς εμπλουτιζόμενες βιβλιοθήκες της.

Λ) **Επεκτασιμότητα:** Η Java μπορεί να συνδεθεί με τμήμα πηγαίου κώδικα που βρίσκεται τοπικά, σε γλώσσες όπως η C.Αυτό σημαίνει ότι έχει μεγάλες πιθανότητες να αναδειχτεί σε γλώσσα γενικής χρήσης.

Μ) **Ταχύτητα:** Η Java είναι πολύ γρήγορη από οποιαδήποτε γλώσσα τύπου Script διότι ο κώδικας έχει ήδη εν μέρει μεταγλωττιστεί.Η καθυστέρηση στην εκτέλεση οφείλεται κυρίως στη σύνδεση και του compilation του byte code.Ο κώδικας σε C είνα συνήθως κατά 20 φορές πιο γρήγορος .

Ν) **Δικτυακή Γνώση (Network-savvy):** Η Java έχει ενσωματωμένη μία μεγάλη συλλογή από βιβλιοθήκες που υλοποιούν άμεσα τα γνωστότερα TCP/IP πρωτόκολλα, όπως το HTTP και το FTP. Αυτό κάνει τη δημιουργία δικτυακών συνδέσεων ευκολότερη από αυτή των C/C++. Οι εφαρμογές της Java μπορούν να προσπελάσουν μία οποιαδήποτε άλλη εφαρμογή ή γενικότερα πόρο στο δίκτυο με χρήση των URLs (Uniform Resource Locators) και με την ίδια ευκολία που μπορεί κάποιος προγραμματιστής να προσπελάσει το τοπικό σύστημα αρχείων.

Ξ) **Ευρωστία (Robust)**: Η Java σχεδιάστηκε ως μία γλώσσα της οποίας τα προγράμματα πρέπει να είναι αξιόπιστα από διάφορες απόψεις. Έτσι, δίνεται ιδιαίτερη έμφαση στον έλεγχο κατά τη μεταγλώττιση πιθανών αιτιών που θα μπορούσαν να προκαλέσουν λάθος. Επιπλέον, γίνεται κατά την εκτέλεση του προγράμματος ένας δυναμικός έλεγχος για να εντοπιστούν άλλα λάθη εκτέλεσης.

Για να γίνει σαφές το πρόβλημα που παρουσιάζουν άλλες γλώσσες, αναφέρουμε ενδεικτικά το πρόβλημα του μεταφραστή της C, όσον αφορά στη δυνατότητα έμμεσης δήλωσης συναρτήσεων. Σε μεγάλης κλίμακας εφαρμογές μία δηλωμένη συνάρτηση μπορεί, κατά λάθος, να θεωρηθεί και υλοποιημένη χωρίς στην πραγματικότητα να υπάρχει κώδικας για τη συγκεκριμένη συνάρτηση ή να είναι προγενέστερη έκδοση της ίδιας, καταλήγοντας σε λάθος κατά την εκτέλεση (runtime error). Σε γλώσσες που χρησιμοποιούν τύπους, όπως η C++, ο έλεγχος τύπων και δηλώσεων γίνεται κατά τη φάση της μεταγλώττισης. Ωστόσο ο μεταγλωττιστής της C++ δεν είναι απαλλαγμένος από όλες τις επιρροές της C.
Η ουσιαστικότερη όμως διαφορά των Java και C/C++ είναι ότι η μεν πρώτη χρησιμοποιεί ένα μοντέλο υλοποίησης των δεικτών που εξαλείφει την πιθανότητα

λανθασμένης εγγραφής ή αλλοίωσης των δεδομένων στη μνήμη, οι άλλες χρησιμοποιούν την τεχνική αριθμητικής των δεικτών (pointer arithmetic – οι δείκτες αντιμετωπίζονται σαν διευθύνσεις μνήμης πάνω στις οποίες μπορούν να εφαρμοστούν όλες οι επιτρεπτές αριθμητικές πράξεις όπως πρόσθεση, αφαίρεση, μετατόπιση). Για τον ίδιο σκοπό η Java χρησιμοποιεί πραγματικούς πίνακες (true arrays) που εγγυώνται ότι γίνεται πάντα ο έλεγχος του δείκτη και δε θα γίνει απόπειρα εγγραφής σε περιοχή μνήμης πέρα από τα όρια του πίνακα. Επίσης δεν υποστηρίζει την μετατροπή αριθμού σε δείκτη (κάτι που περιπλέκει αρκετές φορές τους προγραμματιστές).

Με δεδομένο ότι δε θα γίνει λάθος στο χειρισμό της μνήμης, ο προγραμματιστής μπορεί να είναι ήσυχος ότι κανείς δε θα μπορέσει να του αλλοιώσει τα δεδομένα και επομένως δε θα χρειαστεί να λάβει επιπλέον μέτρα ασφάλειας. Τέλος, ο μεταφραστής δεν αφήνει μεγάλα περιθώρια για επιλογές στον προγραμματιστή όσον αφορά σε τύπους συναρτήσεων. Αυτό το φαινομενικά αρνητικό χαρακτηριστικό, εξασφαλίζει ότι λάθη τύπων θα ελεγχθούν και θα βρεθούν κατά τη διάρκεια της μεταγλώττισης και στο εξής δε θα πρέπει να προβληματίζουν τον προγραμματιστή ως πιθανές αιτίες λαθών.

3) Εκδόσεις της Java

Οι εκδόσεις (version) της Java που έχουν κυκλοφορήσει ως τώρα, ξεκινάνε από τη Java 1.0, η οποία εμφανίστηκε το 1995. Έπειτα ακολούθησε η Java 1.1 το 1996, με μερικές μικρές τροποποιήσεις. Οι επόμενες εκδόσεις από την 1.1 ως την 1.6 βασιζόταν πάνω στην ίδια σύνταξη και στις ίδιες βιβλιοθήκες με μερικές μόνο βελτιώσεις.

4) Τα εργαλεία της Java

Παρακάτω κατανέμονται τα εργαλεία της Java που περιέχονται στην έκδοση 1.0.2.
της Java και του Java Developers Kit (JDK) της Sun Microsystemstm .

Javac: Είναι ο compiler της Java. Η χρήση του στο command-line είναι:
javac <όνομα αρχείου>.

Εδώ να σημειώσουμε ότι το javac δεν παράγει ένα αρχείο με όλον τον κώδικα, αλλά χωριστό αρχείο για κάθε κλάση. Τα αρχεία των κλάσεων ονομάζονται: <όνομα κλάσης> .class.

Java: Είναι ο interpreter της Java. Η χρήση του είναι η εξής: java <κλάση>, πχ java myClass και όχι java myClass.class.

Javaw: (MONO στα Windows 95/NT) Είναι παρόμοιο με το java με μόνη την διαφορά ότι δεν χρειάζεται shell για να τρέξει.

Jdb: Είναι ο Java debuger.
Javah: Κατασκευάζει C files και stub files για κάποια κλάση. Αυτά τα αρχεία είναι απαραίτητα όταν θέλουμε να υλοποιήσουμε κάποιες από τις μεθόδους της κλάσης σε C, πράγμα πολύ σπάνιο.

Javap: Είναι ο Java disassembler.

Javadoc: Είναι ένα πρόγραμμα για αυτόματη κατασκευή documentation. Είναι αρκετά χρήσιμο στην κατασκευή βοηθημάτων και τεχνικών αναφορών για εφαρμογές οποιουδήποτε μεγέθους.

Appletviewer: Είναι ένα πρόγραμμα το οποίο μας επιτρέπει να τρέχουμε και να χρησιμοποιούμε τα διάφορα applets σε Java. Οι stand-alone εφαρμογές, ωστόσο, δεν τρέχουν στον appletviewer αλλά κατευθείαν στον java ή javaw.

5) Τι μπορούμε να κάνουμε με τη Java

Με την Java μπορούμε να δημιουργήσουμε τριών ειδών προγράμματα: τις εφαρμογές, τα applets και τα beans. Οι εφαρμογές είναι αυτόνομα προγράμματα. Χρησιμοποιούμε την Java για να δημιουργήσουμε οποιαδήποτε εφαρμογή, όπως για

παράδειγμα ένα πρόγραμμα επεξεργασίας γραφικών, αλλά κυρίως για την δημιουργία εφαρμογών για τα δίκτυα υπολογιστών. Ένα πρόγραμμα εγκαθίσταται σε έναν κεντρικό υπολογιστή, ενώ η εκτέλεσή του μπορεί να πραγματοποιηθεί από οποιονδήποτε υπολογιστή του δικτύου. Με αυτόν τον τρόπο η εγκατάσταση γίνεται πιο εύκολη και πιο οικονομική. Για την εκτέλεσή της απαιτείται ο Java διερμηνευτής (interpreter). Τα applets είναι προγράμματα ενσωματωμένα σε αρχεία HTML (Hypertext Markup Language, Γλώσσα Σημείωσης Υπερ-κειμένου) που δημιουργούν σελίδες για το Web. Είναι μικρά σε μέγεθος ώστε η αναμονή για την μεταφορά τους να διαρκεί όσο το δυνατό λιγότερο. Εξυπηρετούν περιορισμένους σκοπούς, όπως είναι η δημιουργία εντυπωσιακών εφέ κίνησης για να κάνουν μια εγκατάσταση πιο ελκυστική για τους επισκέπτες του διαδικτύου, αλλά φυσικά χρησιμοποιούνται και για πιο σοβαρές εφαρμογές. Με τα applets υπάρχει αλληλεπίδραση με τον επισκέπτη σε πραγματικό χρόνο. Για παράδειγμα, για την παραγγελία ενός βιβλίου ο επισκέπτης εισάγει τα στοιχεία του και αμέσως λαμβάνει μήνυμα σχετικό με την ενέργεια που μόλις έκανε. Τα beans προστέθηκαν πρόσφατα στην Java. Ο κύριος στόχος είναι, ένα bean να επιτελεί μια συγκεκριμένη και περιορισμένη λειτουργία και να εκτελείται όποτε αυτό ζητηθεί από άλλα beans, εφαρμογές ή applets. Για να αρχίσουμε να προγραμματίζουμε με Java, είναι απαραίτητο να κατανοήσουμε τα παρακάτω:

- Βασικές αρχές για τον προγραμματισμό και ιδιαίτερα για τον αντικειμενοστραφή προγραμματισμό.
- Πώς να ετοιμάσουμε το σύστημά μας για να δημιουργήσουμε προγράμματα σε Java.
- Την βασική δομή των εφαρμογών και των applets.
- Τον τρόπο αποθήκευσης δεδομένων σε ένα πρόγραμμα.
- Πώς να εξαναγκάζουμε ένα πρόγραμμα να επαναλαμβάνει ενέργειες ή να τις πραγματοποιεί κάτω από συγκεκριμένες συνθήκες.
- Τις τεχνικές χρήσης των δομών της Java σε προγράμματα.

6) Εφαρμογές της Java

Η εικονική μηχανή της Java: Αφού γραφεί κάποιο πρόγραμμα σε Java τότε μεταγλωτίζεται μέσω του εργαλείου javac, το οποίο παράγει έναν αριθμό από αρχεία .class (=bytecode). Το bytecode είναι η μορφή που παίρνει ο πηγαίος κώδικας της

Java όταν μεταγλωττιστεί. Όταν προσπαθήσουμε λοιπόν να εκτελέσουμε την εφαρμογή μας το Java Virtual Machine που πρέπει να είναι εγκατεστημένο στο μηχάνημά μας, θα αναλάβει να διαβάσει τα αρχεία .class και να τα μεταφράσει σε γλώσσα και εντολές μηχανής (assembly) που υποστηρίζει το λειτουργικό μας σύστημα και ο επεξεργαστής μας, έτσι ώστε να εκτελεστεί (να σημειώσουμε εδώ ότι αυτό συμβαίνει με την παραδοσιακή Εικονική Μηχανή **(Virtual Machine)** . Πιο σύγχρονες εφαρμογές της εικονικής Μηχανής μπορούν και μεταγλωττίζουν πολύχρηστα τμήματα bytecode απ' ευθείας σε εγγενή κώδικα (native code) με αποτέλεσμα να βελτιώνεται η ταχύτητα). Χωρίς αυτό δε θα ηταν δυνατή η εκτέλεση λογισμικού γραμμένου σε Java. Πρέπει να πούμε οτι η JVM ειναι λογισμικό εξαρτημένο από την πλατφόρμα, δηλαδή για κάθε είδος λειτουργικού συστήματος και αρχιτεκτονικής επεξεργαστή υπάρχει διαφορετική έκδοση του. Έτσι υπάρχουν διαφορετικές JVM για Windows, Linux, Unix, Macintosh, κινητά τηλέφωνα, παιχνιδομηχανές κλπ.

Οτιδήποτε θέλει να κάνει ο προγραμματιστής (ή ο χρήστης) γίνεται μέσω της εικονικής μηχανής. Αυτό βοηθάει στο να υπάρχει μεγαλύτερη ασφάλεια στο σύστημα γιατί η εικονική μηχανή είναι υπεύθυνη για την επικοινωνία χρήστη - υπολογιστή. Ο προγραμματιστής δεν μπορεί να γράψει κώδικα ο οποίος θα έχει καταστροφικά αποτελέσματα για τον υπολογιστή γιατί η εικονική μηχανή θα τον ανιχνεύσει και δε θα επιτρέψει να εκτελεστεί. Από την άλλη μεριά ούτε ο χρήστης μπορεί να κατεβάσει «κακό» κώδικα από το δίκτυο και να τον εκτελέσει. Αυτό είναι ιδιαίτερα χρήσιμο για μεγάλα κατανεμημένα συστήματα όπου πολλοί χρήστες χρησιμοποιούν το ίδιο πρόγραμμα συγχρόνως.

Ο συλλέκτης απορριμάτων (Garbage Collector): Ακόμα μία ιδέα που βρίσκεται πίσω από τη Java είναι η ύπαρξη του **συλλέκτη απορριμάτων** (Garbage Collector). Συλλογή απορριμάτων είναι μία κοινή ονομασία που χρησιμοποιείται στον τομέα της πληροφορικής για να δηλώσει την ελευθέρωση τμημάτων μνήμης από δεδομένα που δε χρειάζονται και δε χρησιμοποιούνται άλλο. Αυτή η απελευθέρωση μνήμης στη Java είναι αυτόματη και γίνεται μέσω του συλλέκτη απορριμάτων. Υπεύθυνη για αυτό είναι και πάλι η εικονική μηχανή η οποία μόλις «καταλάβει» ότι ο σωρός (heap) της μνήμης (στη Java η συντριπτική πλειοψηφία των αντικειμένων αποθηκεύονται στο

σωρό σε αντίθεση με τη C++ όπου αποθηκεύονται κυρίως στη στοίβα - stack) κοντεύει να γεμίσει ενεργοποιεί το συλλέκτη απορριμάτων.

Έτσι ο προγραμματιστής δε χρειάζεται να ανησυχεί για το πότε και αν θα ελευθερώσει ένα συγκεκριμένο τμήμα της μνήμης, ούτε και για δείκτες (pointers) που αναφέρονται σε άδειο χώρο μνήμης. Αυτό είναι ιδιαίτερα σημαντικό αν σκεφτούμε ότι ένα μεγάλο ποσοστό κατάρρευσης των προγραμμάτων οφείλονται σε λανθασμένο χειρισμό της μνήμης.

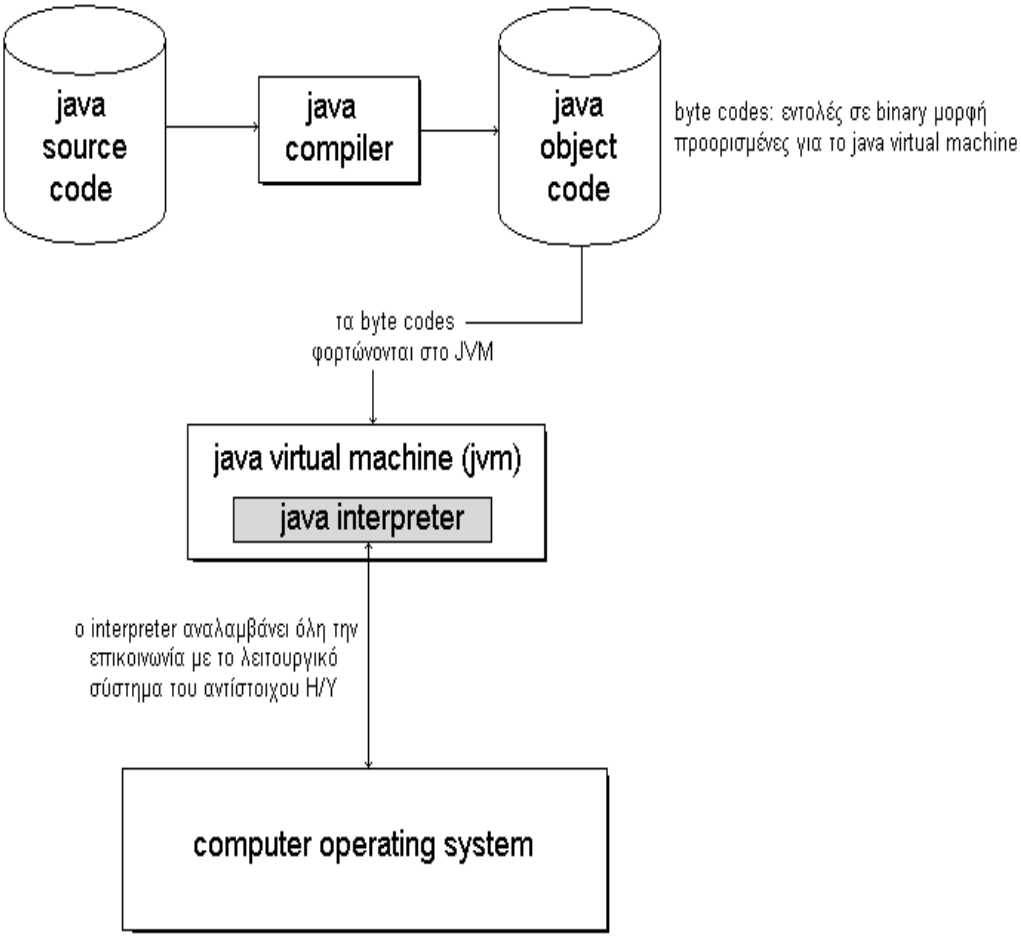
Ερμηνευτές Java και προγράμματα περιήγησης: Ένα πρόγραμμα περιήγησης μπορεί να περιέχει έναν ή περισσότερους ερμηνευτές.Ένα συμβατικό πρόγραμμα περιήγησης περιέχει ένα ερμηνευτή HTML ο οποίος χρησιμοποιείται για να εμφανίζει στην οθόνη στατικά ή δυναμικά έγγραφα. Ένα πρόγραμμα περιήγησης που χρησιμοποιεί την Java περιέχει δύο ερμηνευτές:ένα ερμηνευτή HTML, και ακόμα ένα για τα εφαρμογίδια.

Ένας ερμηνευτής Java είναι σύνθετο πρόγραμμα.Η καρδιά του ερμηνευτή είναι ένας απλός βρόχος ο οποίος προσομειώνει έναν υπολογιστή. Ο ερμηνευτής διατηρεί ένα δείκτη εντολής ο οποίος αρχικά δείχνει στην αρχή του εφαρμογιδίου. Σε κάθε επανάληψη του βρόχου, ο ερμηνευτής προσκομίζει τον κώδικα bytecode που βρίσκει στην διεύθυνση στην οποία δείχνει ο δείκτης εντολής. Έπειτα, ο ερμηνευτής αποκωδικοποιεί την εντολή και πραγματοποιεί την καθορισμένη πράξη. Για παράδειγμα, αν ο ερμηνευτής βρει τον bytecode add (πρόσθεση) και δύο ακέραιους τελεστέους, προσθέτει τους δύο ακέραιους, ενημερώνει το δείκτη εντολής, και συνεχίζει με την επόμενη επανάληψη του βρόχου.

Εκτός από ένα βασικό αποκωδικοποιητή εντολών, έναν ερμηνευτή Java πρέπει να περιλαμβάνει υποστήριξη για το περιβάλλον χρόνου εκτέλεσης της Java. Δηλαδή, ένας ερμηνευτής Java πρέπει να είναι σε θέση να εμφανίζει γραφικά στην οθόνη του χρήστη, να προσπελάζει το Internet, και να πραγματοποιεί είσοδο/έξοδο. Ακόμα, ο ερμηνευτής πρέπει να είναι σχεδιασμένος έτσι ώστε να επιτρέπει σε ένα εφαρμογίδιο να χρησιμοποιεί υπηρεσίες του προγράμματος περιήγησης για να ανακαλεί και να εμφανίζει στατικά και δυναμικά έγγραφα. Επομένως, ο ερμηνευτής Java που

περιέχεται σε ένα πρόγραμμα περιήγησης πρέπει να είναι σε θέση να επικοινωνεί με τον πελάτη HTTP και τον ερμηνευτή HTML του προγράμματος περιήγησης.

Μεταγλώτιση ενός προγράμματος Java: Ένα εφαρμογίδιο Java πρέπει να μεταγλωττιστεί και να αποθηκευτεί σε ένα διακομιστή του Ιστού προτού μπορέσει ένα πρόγραμμα περιήγησης να το κατεβάσει και να το εκτελέσει.Η τεχνολογία Java περιλαμβάνει ένα μεταγλωττιστή που ονομάζεται Javac. Ο προγραμματιστής εκτελεί τον Javac για να μεταγλωττίσει ένα πηγαίο πρόγραμμα Java στην αναπαράσταση bytecode της γλώσσας.



Η είσοδος για τον μεταγλωττιστή Javac είναι ένα πηγαίο πρόγραμμα Java. Το όνομα του αρχείου εισόδου πρέπει να τελειώνει με το επίθεμα . Java. Ο μεταγλωττιστής Javac επαληθεύει αν το πηγαίο πρόγραμμα είναι συντακτικά σωστό, μεταφράζει το πρόγραμμα στην αναπαράσταση bytecode, και στέλνει την έξοδο σε ένα αρχείο με επίθεμα .class.

Ένα πρόγραμμα Java αποτελείται από μια ακολουθία δηλώσεων.Ένα στοιχείο που δηλώνεται ως puplic (δημόσιο) εξάγεται ώστε να είναι διαθέσιμο και σε άλλα εφαρμογίδια, ένα στοιχείο που δηλώνεται ως private (ιδιωτικό) δεν εξάγεται, και δε μπορούν να γίνονται εξωτερικές αναφορές σε αυτό. Ένα αρχείο πηγαίου κώδικα πρέπει να περιέχει ακριβώς μια τάξη puplic, της οποίας το όνομα πρέπει να είναι ίδιο με το πρόθεμα του ονόματος του αρχείου του πηγαίου κώδικα. Δηλαδή, η δημόσια τάξη του αρχείου d.java πρέπει να ονομάζεται d. Επομένως ένα αρχείο που περιέχει το μεταγλωττισμένο bytecode για την τάξη d ονομάζεται d. class.

Παράδειγμα:Ένα αρχείο πηγαίου κώδικα A.java περιέχει τις παρακάτω δηλώσεις:

```
puplic class A { ... }
class B { ... }
class C { ... }
```

Το πρόγραμμα δηλώνει την τάξη A ως δημόσια (puplic), ενώ οι τάξεις B και C είναι ιδιωτικές. Όταν μεταγλωττίζεται το A.java, ο μεταγλωττιστής Javac παράγει το αρχείο εξόδου A. Class.

Το περιβάλλον του χρόνου εκτέλεσης της: Η τεχνολογία Java ορίζει ένα περιβάλλον χρόνου εκτέλεσης στο οποίο εκτελούνται τα προγράμματα της γλώσσας. Το περιβάλλον του χρόνου εκτέλεσης χαρακτηρίζεται από τα παρακάτω:

- Ερμηνευόμενη εκτέλεση. Αν και ένα πρόγραμμα Java μπορεί να μεταγλωττιστεί σε κώδικα κατάλληλο για ένα συγκεκριμένο υπολογιστή, ο σκοπός των σχεδιαστών ήταν να χρησιμοποιείται η γλώσσα ως ερμηνευτής. Δηλαδή, σε αντίθεση με ένα συμβατικό μεταγλωττιστή, ο οποίος μεταφράζει ένα πηγαίο πρόγραμμα σε δυαδικό πρόγραμμα αντικειμενικού κώδικα για τη συγκεκριμένη αρχιτεχτονική, ένας μεταγλωττιστής Java μεταφράζει ένα πρόγραμμα Java σε μια ανεξάρτητη από τον υπολογιστή δυαδική αναπαράσταση, που ονομάζεται αναπαράσταση bytecode της Java. Ένα πρόγραμμα που ονομάζεται ερμηνευτής διαβάζει την αναπαράσταση bytecode και ερμηνεύει τις εντολές.
- Αυτόματη περισυλλογή σκουπιδιών. Η διαχείριση της μνήμης στην Java διαφέρει ριζικά από τη διαχείριση μνήμης που χρησιμοποιείται σε γλώσσες όπως η C και η C++. Αντί να απαιτείται από ένα πρόγραμμα να καλεί διαδικασίες όπως οι malloc και free για να κατανέμει και να απελευθερώνει μνήμη, το σύστημα χρόνου

εκτέλεσης της Java παρέχει αυτόματη περισυλλογή σκουπιδιών. Δηλαδή, ένα πρόγραμμα μπορεί να βασίζεται στο σύστημα χρόνου εκτέλεσης για να βρίσκει και να ανακτά τη μνήμη που δε χρησιμοποιείται πια.Ουσιαστικά, όταν ένα πρόγραμμα καταργεί όλες τις αναφορές σε ένα αντικείμενο, ο συλλεκτής σκουπιδιών ανακτά τη μνήμη που έχει αποδοθεί σε αυτό το αντικείμενο.

- Πολυνηματική εκτέλεση.Το σύστημα χρόνου εκτέλεσης της Java παρέχει υποστήριξη για την ταυτόχρονη εκτέλεση νημάτων.Ουσιαστικά, το σύστημα του χρόνου εκτέλεσης περιέχει τα τμήματα ενός κειτουργικού συστήματος τα οποία χειρίζονται το χρονοπρογραμματισμό και τη θεματική εναλλαγή.Καθώς η CPU εναλλάσσεται μεταξύ ενός συνόλου από τρέχοντα ενεργά νήματα, το σύστημα χρόνου εκτέλεσης επιτρέπει να εκτελούνται όλα τα νήματα.
- Πρόσβαση στο Internet.Το σύστημα χρόνου εκτέλεσης της Java περιλαμβάνει μια βιβλιοθήκη υποδοχών την οποία χρησιμοποιεί ένα πρόγραμμα για να αποκτήσει πρόσβαση στο Internet.Συγκεκριμένα, ένα πρόγραμμα Java μπορεί να γίνει πελάτης-το πρόγραμμα Java μπορεί να χρησιμοποιήσει το TCP ή το UDP για να έρθει σε επαφή με ένα μακρινό διακομιστή.
- Υποστήριξη γραφικών.Επειδή ένα εφαρμογίδιο πρέπει να έχει τη δυνατότητα να ελέγχει την οθόνη, το σύστημα του χρόνου εκτέλεσης της Java περιλαμβάνει βασικούς μηχανισμούς οι οποίοι επιτρέπουν σε ένα πρόγραμμα να δημιουργεί στην οθόνη παράθυρα που περιέχουν κείμενο ή γραφικά.

Ανεξαρτησία από τον υπολογιστή και φοριτότητα: Η γλώσσα και το σύστημα του χρόνου εκτέλεσης της Java έχουν σχεδιαστεό για να κάνουν την τεχνολογία Java ανεξάρτητη από το υλικό των υπολογιστών.Για παράδειγμα, η γλώσσα δεν περιέχει κανένα χαρακτηριστικό που ορίζεται από την υλοποίηση. Αντίθετα, το εγχειρίδιο αναφοράς της γλώσσας ορίζει τη σημασία του κάθε τελεστή και της κάθε πρότασης της γλώσσας μονοσήμαντα. Ακόμα, η αναπαράσταση bytecode της Java είναι ανεξάρτητη από το υποκείμενο υλικό. Επομένως, αφού ένα πρόγραμμα Java μεταγλωττιστεί στην αναπαράσταση bytecode, παράγει ακριβώς την ίδια έξοδο σε οποιονδήποτε υπολογιστή.

Η χρήση μιας δυαδικής αναπαράστασης ανεξάρτητης από τον υπολογιστή και ενός ερμηνευτή κάνει τα προγράμματα Java φορητά μεταξύ των διαφόρων αρχιτεκτονικών υπολογιστών. Φανταστείτε ένα πρόγραμμα Java που έχει μεταγλωττιστεί σε ένα

υπολογιστή με επεξεργαστή Pentium, βρίσκεται αποθηκευμένο σε ένα διακομηστή Ιστού που χρησιμοποιεί επεξεργαστή Sparc, και εκτελείται σε κάποια άλλη αρχιτεκτονική. Επίσης, η ίδια έξοδος θα προκύψει αν το μεταγλωττισμένο πρόγραμμα εκτελεστεί σε ένα άλλο υπολογιστή.

Το να διατηρούνται τα εφαρμογίδια αναξέρτητα από το υποκείμενο υλικό των υπολογιστών είναι απαραίτητο, για 3 λόγους. Πρώτον, οι χρήστες του Internet έχουν πολλούς τύπους υπολογιστών. Η ανεξαρτησία από τον υπολογιστή επιτρέπει σε ένα πρόγραμμα περιήγησης που εκτελείται σε οποιαδήποτε μάρκα υπολογιστή να κατεβάζει και να εκτελεί ένα αντίγραφο του ενεργού εγγράφου. Δεύτερον, η ανεξαρτησία από τον υπολογιστή εξασφαλίζει ότι το έγγραφο θα παράγει σωστή έξοδο για όλα τα προγράμματα περιήγησης. Τρίτον, η ανεξαρτησία από τον υπολογιστή μειώνει ριζικά το κόστος δημιουργίας των εγγράφων, επειδή ένας προγραμματιστής μπορεί να κατασκευάζει και να δοκιμάζειμόνο μια έκδοση ενός ενεργού εγγράφου, αντί για μια έκδοση για κάθε τύπο υπολογιστή.

Η βιβλιοθήκη Java: Το τρίτο συστατικό μέρος της τεχνολογίας Java είναι μια βιβλιοθήκη ορισμένων τάξεων. Η βιβλιοθήκη είναι εκτεταμένη-περιέχει δεκάδες ορισμούς τάξεων, με περίπου 2000 ξεχωριστές μεθόδους. Ακόμα σημαντικότερο, η βιβλιοθήκη περιέχει τάξεις οι οποίες καλύπτουν ένα ευρύ φάσμα αναγκών. Για παράδειγμα, η βιβλιοθήκη περιέχει τάξεις για τα παρακάτω:

- Χειρισμός γραφικών.Οι ρουτίνες χειρισμού γραφικών επιτρέπουν σε ένα πρόγραμμα Java να ασκεί ακριβή έλεγχο στα περιεχόμενα και την εμφάνιση της οθόνης του χρήστη (π.χ υπάρχουν στη βιβλιοθήκη υπηρεσίες που επιτρέπουν σε ένα εφαρμογίδιο να εμφανίζει κείμενο, γραφικά, εικόνες, ή πλαίσια διαλόγου).
- Χειρισμός εξαιρέσεων.Όταν παρουσιάζεται μια απροσδόκητη συνθήκη ή σφάλμα σε ένα πρόγραμμα Java, το περιβάλλον χρόνου εκτέλεσης προκαλεί μια εξαίρεση.Η βιβλιοθήκη της Java περιλαμβάνει ένα σύνολο τάξεων οι οποίες διευκολύνουν το χειρισμό των εξαιρέσεων.
- Σύλληψη συμβάντων.Συμβάντα προκύπτουν όταν ένας χρήστης πατά ή αφήνει ένα πλήκτρο του ποντικιού ή του πληκτρολογίου.Η βιβλιοθήκη περιέχει τάξεις οι οποίες επιτρέπουν σε ένα πρόγραμμα Java να συλλαμβάνει και να χειρίζεται τέτοια συμβάντα.

- Είσοδος/Έξοδος αρχείων.Ένα εφαρμογίδιο χρησιμοποιεί υπηρεσίες εισόδου/εξόδου αρχείων για να χειρίζεται αρχεία στον τοπικό υπολογιστή.Η είσοδος/έξοδος αρχείων χρησιμοποιείται από ένα πρόγραμμα που αποθηκεύουνμακροπρόθεσμες πληροφορίες κατάστασης.
- Πρόσβαση στο σύστημα χρόνου εκτέλεσης. Η βιβλιοθήκη της Java περιέχει τάξεις τις οποίες μπορεί να χρησιμοποιεί ένα εφαρμογίδιο για να αποκτά πρόσβαση σε υπηρεσίες του περιβάλλοντος του χρόνου εκτέλεσης.Για παράδειγμα, ένα εκτελούμενο πρόγραμμα μπορεί να ζητήσει να δημιουργηθεί ένα νήμα.

Χρήση γραφικών Java σε ένα συγκεκριμένο υπολογιστή: Επειδή η εργαλειοθήκη AWT έχει σχεδιαστεί έτσι ώστε να είναι ανεξάρτητη από το υλικό και το σύστημα γραφικών των υπολογιστών οποιασδήποτε συγκεκριμένης εταιρείας η Java δε χρειάζεται να χρησιμοποιεί απευθείας το υλικό απεικόνισης. Η Java μπορεί να εκτελείται πάνω από τα συμβατικά συστήματα παραθύρων.Για παράδειγμα, αν ένας χρήστης εκτελεί ένα πρόγραμμα περιήγησης σε ένα υπολογιστή που χρησιμοποεί το X Window System, ένα εφαρμογίδιο Java χρησιμοποιεί το X για να δημιουργεί και να χειρίζεται παράθυρα στην οθόνη. Αν ένας άλλος χρήστης εκτελεί ένα πρόγραμμα περιήγησης με διαφορετικό σύστημα παραθύρων για να δημιουργεί και να χειρίζεται παράθυρα στην οθόνη.

Ένα εφαρμογίδιο μπορεί να δουλεύει σε πολλά συστήματα παραθύρων μέσω μιας σημαντικής αντιστοιχίας η οποία είναι ενσωματωμένη στο περιβάλλον χρόνου εκτέλεσης, και σε ένα σύνολο ενδιάμεσων συναρτήσεων. Ένα περιβάλλον χρόνου εκτέλεσης της Java περιλαμβάνει ένα ενδιάμεσο επίπεδο λογισμικού, όπου κάθε συνάρτηση του χρησιμοποίει το σύστημα παραθύρων του υπολογιστή για να υλοποιεί μια συγκεκριμένη πράξη γραφικών της Java.Πριν εκτελεστεί ένα εφαρμογίδιο, το περιβάλλον χρόνου εκτέλεσης πραγματοποιεί μια αντιστοιχία μεταξύ της κάθε μεθόδου γραφικών της Java και της κατάλληλης ενδιάμεσης συνάρτησης. Όταν ένα εφαρμογίδιο ζητήσει μια πράξη της εργαλειοθήκης AWT, ο έλεγχος περνά σε μια μέθοδο της βιβλιοθήκης της Java.Στη συνέχει η μέθοδος της βιβλιοθήκης μεταβιβάζει τον έλεγχο στη κατάλληλη ενδιάμεση συνάρτηση, η οποία χρησιμοποιεί το σύστημα παραθύρων του υπολογιστή για να πραγματοποιήσει την πράξη. Η έννοια αυτή ονομάζεται εργοστάσιο.

ΕΠΑΝΑΛΗΠΤΙΚΕΣ ΕΡΩΤΗΣΕΙΣ 1^{ΟΥ} ΚΕΦΑΛΑΙΟΥ

Απαντώντας αυτές τις ερωτήσεις θα βεβαιωθείτε ότι έχετε κατανοήσει πλήρως αυτά που διαβάσατε και είναι μια καλή ευκαιρία να λυθούν τυχόν απορίες. Σκοπός των επαναληπτικών ερωτήσεων είναι η αυτοαξιολόγηση του αναγνώστη.

<u>Ερωτήσεις κρίσεως</u>

Ερώτηση 1: Ποιο από τα χαρακτηριστικό της Java πιστεύετε ότι την έκανε τόσο δημοφιλή;

_____

_____

_____

Ερώτηση 2: Θα μπορούσε κάποιος να γράψει κώδικα Java σε περιβάλλον Unix και έπειτα να τον μεταφράσει σε περιβάλλον Windows; και γιατί;

_____

_____

_____

Ερώτηση 3: Για ποιο λόγο πιστεύετε ότι δόθηκε στη Java ο χαρακτηρισμός απλή;

_____

_____

_____

Ερωτήσεις πολλαπλής επιλογής

1. Η Java είναι:

α. Μεταφραστής κώδικα προγραμμάτων.

β. Γλώσσα κατασκευής ιστοσελίδων.

γ. Γλώσσα κατασκευής προγραμμάτων.

δ. Όλα τα προηγούμενα.

2. Η Java χαρακτηρίζεται ως:

α. Απλή, δυναμική και ασφαλής.

β. Αργή και δυσλειτουργική.

γ. Γρήγορη και πολυνηματική με μη ερμηνευτικό κώδικα.

δ. Το α και το γ.

Ερωτήσεις τύπου σωστό- λάθος.

1. Ο Java compiler ή Javac μεταφράζει τον ψευδοκώδικα σε αναγνώσιμο για τον χρήστη κώδικα και το αντίθετο.

_____

_____

2. Η Virtal Machine της Java προστατεύει τον υπολογιστή από την εισβολή επικίνδυνου κώδκα.

_____

_____

3. Το JDK , το JIT και το Jdb είναι μεταφραστές της Java.

_____

Το 2^{ο} κεφάλαιο είναι ίσως το σημαντικότερο και το πιο ενδιαφέρον. Σε αυτή την ενότητα θα παρακολουθήσετε βήμα προς βήμα πως γίνεται η εγκατάσταση του περιβάλλοντος του Net Beans στον υπολογιστή σας, μέσα από το οποίο θα έχετε τη δυνατότητα να δημιουργήσετε τα δικά σας προγράμματα.

Το πιο σημαντικό είναι ότι σε αυτό το κεφάλαιο θα προχωρήσουμε και στη δημιουργία του πρώτου μας προγράμματος. Με απλά παραδείγματα και ασκήσεις τελειώνοντας το τρίτο κεφάλαιο θα πρέπει να έχετε μάθει να φτιάχνετε απλά προγράμματα μόνοι σας.

ΚΕΦΑΛΑΙΟ 2

1) Εγκατάσταση της Java

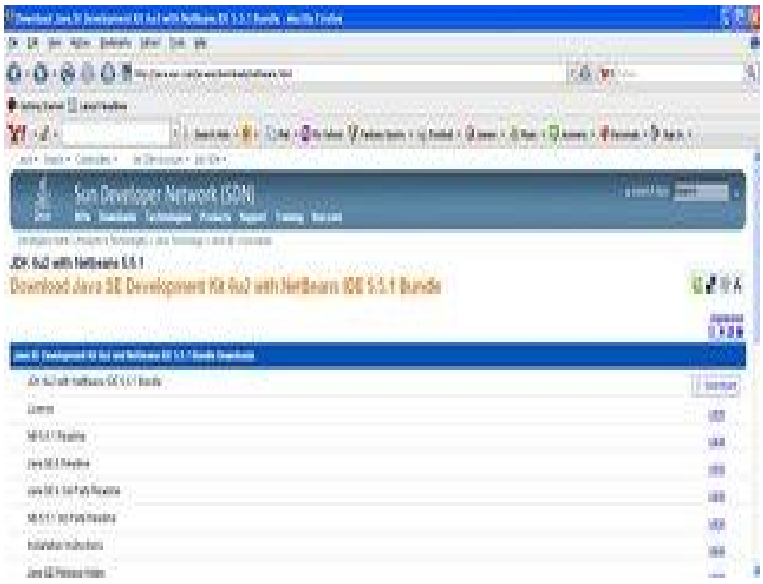
Οι εκδόσεις της Java σε διαφορετικά στάδια ολοκλήρωσης διατίθενται από τη Sun for Windows 95 και Windows NT for X86, Unix και MacOS 7.5. Μέχρι στιγμής δεν υπάρχουν εκδόσεις της Java για τα MIPS, Alpha or PowerPC based NT, Windows 3.1, Amiga. Το βασικό περιβάλλον της Java αποτελείται από έναν web browser, ο οποίος μπορεί να εκτελεί τις μίνι εφαρμογές της Java, έναν compiler που μετατρέπει τον πηγαίο κώδικα της Java σε κώδικα byte, κι έναν μεταφραστή της Java για να εκτελεί τα προγράμματα. Αυτά είναι τα τρία συστατικά-κλειδιά ενός περιβάλλοντος Java. Επίσης απαραίτητος είναι ένας text editor όπως το Brief ή το BBEdit.

Η Sun διαθέτει το Java Developers Kit (JDK). Περιέχει έναν applet viewer όπου θα μπορείτε να δείτε και να ελέγξετε τις εφαρμογές σας. Το JDK περιλαμβάνει επίσης τον javac compiler, τον java interpreter, τον javaprof profiler, τον Java debugger και περιορισμένα κείμενα. Τα περισσότερα από τα κείμενα για το API και τη βιβλιοθήκη κλάσης είναι στο web site της Sun.

Ας δούμε τώρα βήμα βήμα πως θα κάνετε την εγκατάσταση της Java στον υπολογιστή σας. Το πρώτο βήμα είναι να κατεβάσουμε από την Sun Microsystems το JDK που είναι ο compiler μαζί με το γραφικό περιβάλλον του NetBeans. Το ολοκληρωμένο πακέτο υπάρχει στην εξής ακόλουθη ηλεκτρονική διεύθυνση:

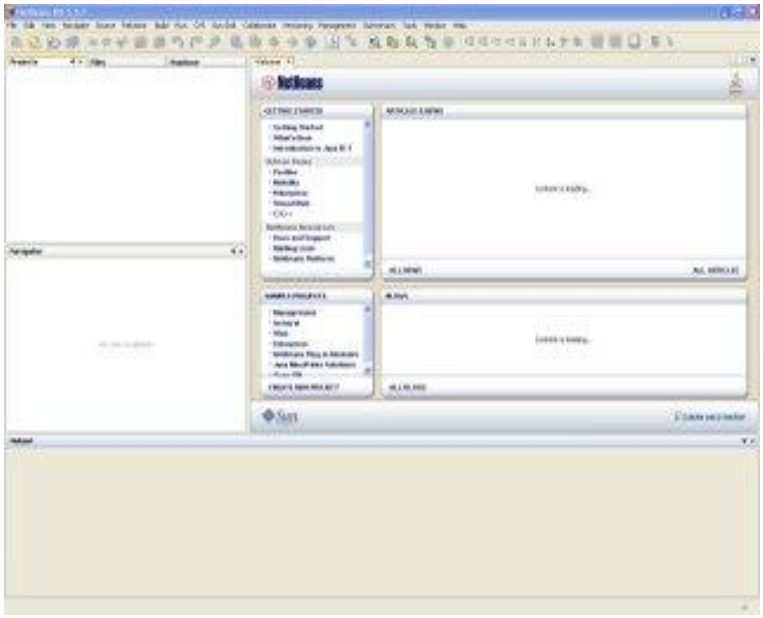
http://java.sun.com/javase/downloads/netbeans.html

Η ιστοσελίδα που θα ανοίξετε θα είναι όπως η παρακάτω.



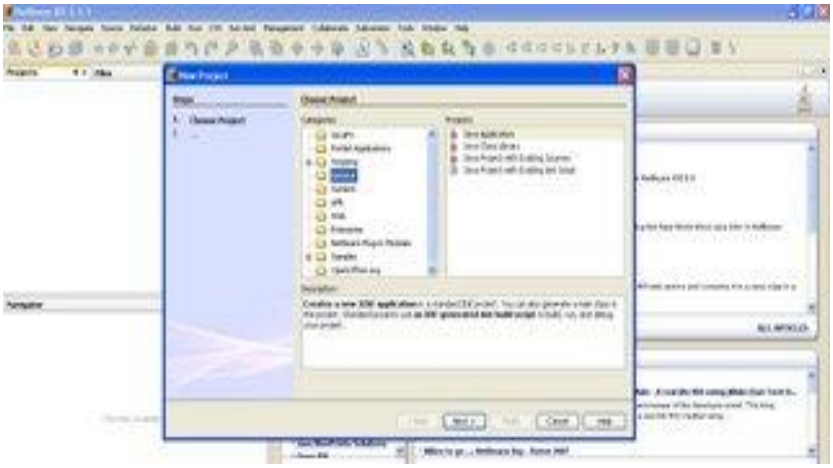
Εικόνα 1

Θα πρέπει τώρα για να ολοκληρωθεί η διαδικασία κατεβάσματος του προγράμματος να πατήσετε το κουμπί “download” για να δέχτηκε τους ορούς από την Sun Microsystems. Το αρχείο το όποιο θα κατέβει στον υπολογιστή σας θα έχει μέγεθος περίπου 149 MB. Κάνετε διπλό κλικ επάνω του και δέχεστε όλες τις επιλογές πατώντας “next” σε κάθε παραθυράκι που θα σας εμφανίζει. Το πρόγραμμα αρχικά θα εγκαταστήσει το JKD και μετά το γραφικό περιβάλλον του NetBeans. Όταν ολοκληρωθεί η εγκατάσταση του περιβάλλοντος του Net Beans τότε η οθόνη σας θα πρέπει να είναι όπως η εικόνα 2.



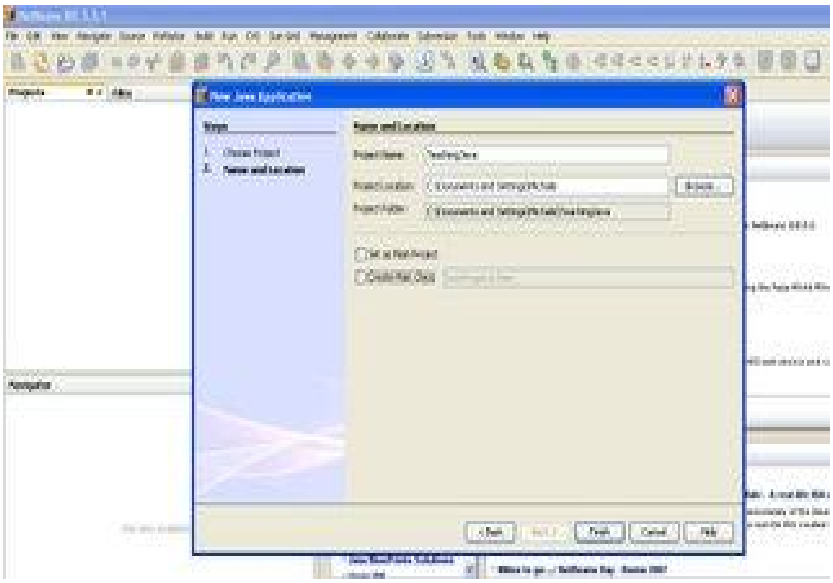
Εικόνα 2

Το NetBeans δουλεύει κάτω από την έννοια του project. Δηλαδή πριν ξεκινήσει το πρώτο πρόγραμμα πρέπει να οριστεί ένα project κάτω από το όποιο θα γραφούν όλες οι εφαρμογές. Από το βασικό μενού πατάτε την επιλογή File, διαλέγετε την επιλογή New Project και στο παράθυρο που εμφανίζεται επιλέγετε το φάκελο General από τα αριστερά, Java Application από τα δεξιά, και έπειτα πατάτε το κουμπί next.



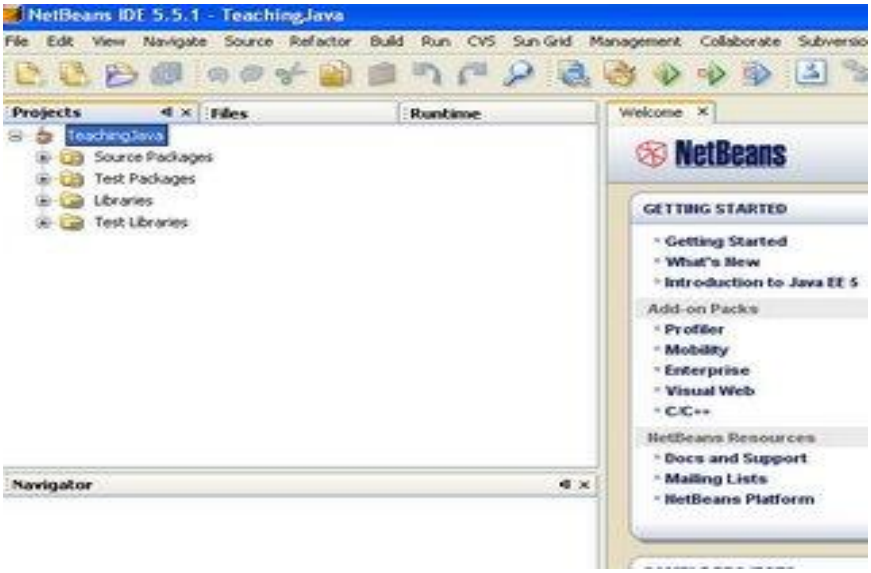
Εικόνα 3

Στο τελευταίο παράθυρο επιλογών που εμφανίζεται, στο Project Name δίνετε ένα όνομα της αρεσκείας σας (έστω TeachingJava, δίνοντας όμως μεγάλη προσοχή στο διαχωρισμό πεζών-κεφαλαίων), αφού αποδεχτείτε τις Project Location και Project Folder επιλογές, βγάζετε τα δυο πράσινα ckecks από τις επιλογές Set As Main Project και Create Main Class έτσι ώστε να είναι καθαρά.



Εικόνα 4

Πατώντας το κουμπί Finish θα δημιουργηθεί ένα project με το εικονάκι του φλιτζανιού καφέ το όποιο θα περιέχει κάθε πρόγραμμα που θα γράψετε.

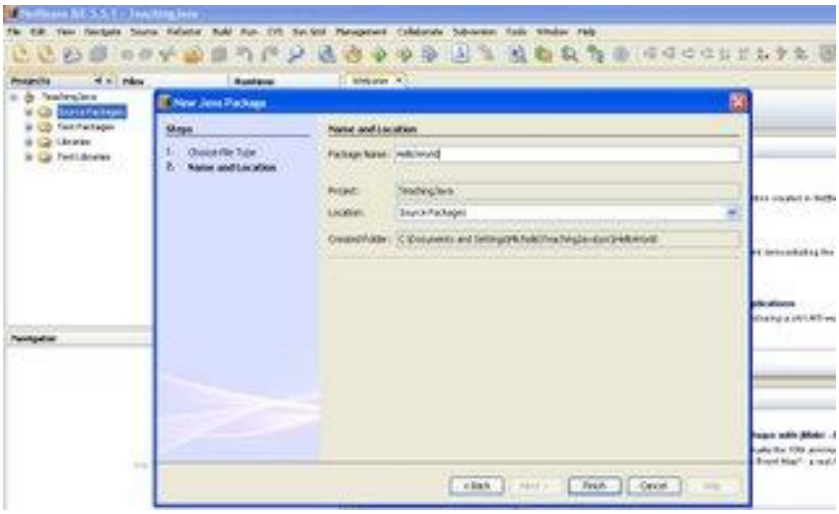


Εικόνα 5

Αυτή ήταν η διαδικασία της δημιουργίας ενός project, η οποία δε θα χρειαστεί να επαναληφθεί εάν πρόκειται να ασχοληθείτε με τη δημιουργία ερασιτεχνικών προγραμμάτων.

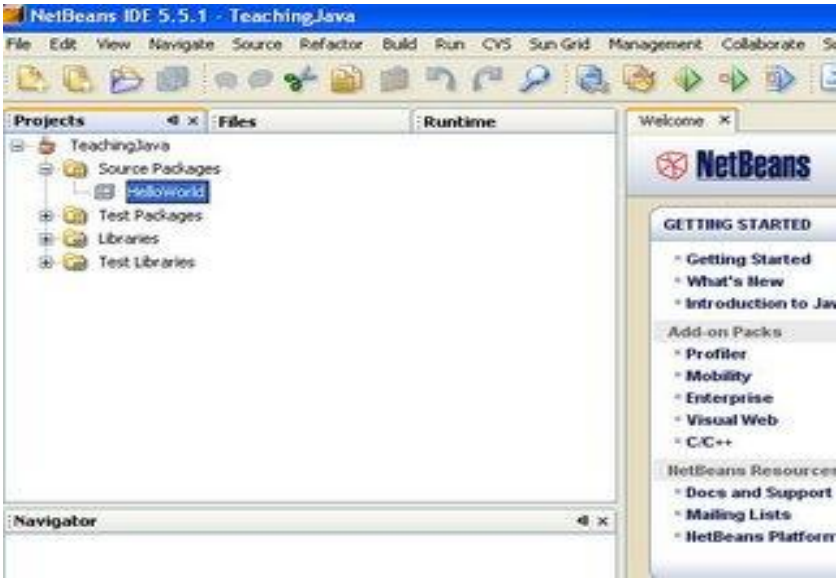
Επόμενο βήμα και ποιό ουσιώδες, είναι η δημιουργία πακέτου. Ένα πακέτο περιέχει όλα εκείνα τα αρχεία που συνεργάζονται μαζί για να κάνουν μια εφαρμογή να τρέξει. Για να καταλαβαίνει όμως το NetBeans ποια αρχεία ανήκουν μαζί στην ιδία εφαρμογή, έτσι ώστε να τα βρίσκει εύκολα και να τα συνδυάζει για να δίνει το επιθυμητό αποτέλεσμα, τα περικλείει στην έννοια του πακέτου.

Για τη δημιουργία πακέτου, κάνουμε δεξί κλικ στο εικονίδιο Source Packages, επιλεγούμε New, και μετά Java Package. Στο παράθυρο που ανοίγει πρέπει να δώσετε ένα όνομα στο πακέτο (για λόγους ευκολίας μπορείτε να δώσετε στο πακέτο το ίδιο όνομα με το πρόγραμμα βάζοντας το κάθε πρόγραμμα στο δικό του πακέτο). Όποτε ονομάζετε το πακέτο HelloWorld και πατάτε Finish.



Εικόνα 6

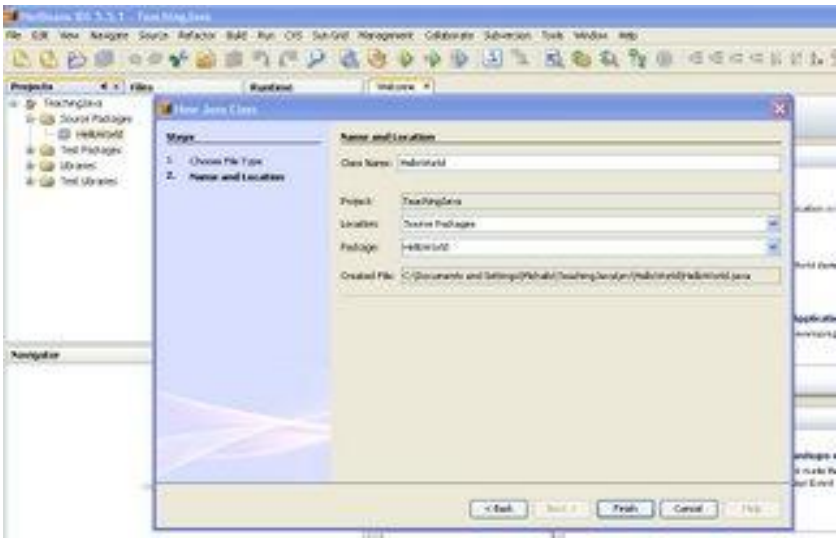
Το πακέτο HelloWorld βρίσκεται ήδη κάτω από το Source Packages και είναι έτοιμο για να γράψετε το πρώτο σας πρόγραμμα.



Εικόνα 7

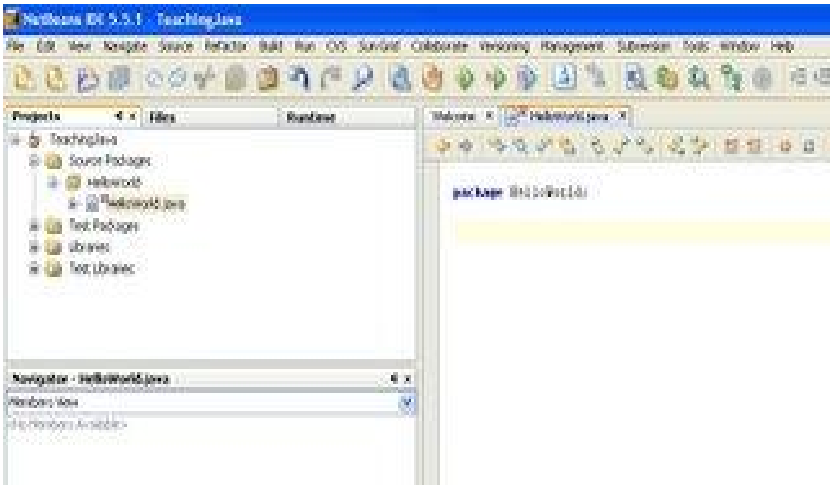
Κάνετε δεξί κλικ επάνω στο πακέτο HelloWorld που δημιουργήσατε, επιλέγετε New, και μετά Java class. Στην επιλογή Class Name βάζετε το όνομα του προγράμματος το όποιο είναι HelloWorld. Αυτό συμβαίνει γιατί κάθε πρόγραμμα αποτελεί και μια κλάση.

Όπως αναφέρθηκε και προηγουμένως, πρέπει να δοθεί μεγάλη προσοχή στο διαχωρισμό πεζών κεφαλαίων όταν δίνετε ονόματα σε προγράμματα ή πακέτα, διότι η Java είναι πολύ ευαίσθητη με τα πεζά και κεφάλαια γράμματα. Όπως γράψετε το όνομα του προγράμματος στο παραθυράκι που έχετε ανοιχτό τώρα, έτσι πρέπει να το γράψετε και μέσα στο πρόγραμμα. Το HelloWorld με την λέξη helloworld δεν είναι το ίδιο για την Java. Τα βλέπει σαν δυο διαφορετικά ονόματα! Στη συνέχεια αφήνετε τις υπόλοιπες επιλογές όπως είναι και πατάτε Finish.



Εικόνα 8

Το κείμενο που εμφανίζεται είναι ένας σκελετός για τη δημιουργία προγραμμάτων, τον οποίο δίνει έτοιμο η Java για να διευκολύνει τους χρήστες της. Για να γράψετε κώδικα ξεκινάτε από τον κενό χώρο κάτω από το κείμενο.



Εικόνα 9

2) Διαδικασία ανάπτυξης της εφαρμογής

Κάθε υπολογιστής μπορεί να αναγνωρίσει και να καταλάβει μία μορφή προγράμματος και αυτή είναι ο κώδικας μηχανής (machine code). Ο κώδικας αυτός αποτελείται από δυαδικούς αριθμούς, τους οποίους ο επεξεργαστής του υπολογιστή μεταφράζει σε εντολές, που είναι ιδιαίτερα απλές, σε τιμές και σε διευθύνσεις μνήμης.

Οι περισσότερες γλώσσες προγραμματισμού χρησιμοποιούν λέξεις και δομές που είναι κατανοητές από τους ανθρώπους. Ο προγραμματιστής καταγράφει εντολές σε ένα αρχείο κειμένου, που είναι γνωστός ως πηγαίος κώδικας (source code). Η διαδικασία ανάπτυξης αυτόνομης εφαρμογής σε ένα τυπικό περιβάλλον ανάπτυξης Java εφαρμογών, περιλαμβάνει τρεις ενέργειες:

Συγγραφή του πηγαίου κώδικα: Για την συγγραφή του πηγαίου κώδικα μπορούμε να χρησιμοποιήσουμε έναν editor της αρεσκείας μας. Για παράδειγμα, στο Unix μπορούμε να χρησιμοποιήσουμε τον emacs, ενώ στα Windows 95 και Windows NT το Notepad ή τον DOSEdit. Ολοκληρωμένα περιβάλλοντα ανάπτυξης όπως το Visual J++ της Microsoft ή το Visual Cafe της Symantec έχουν τους δικούς τους εξειδικευμένους editors.
Προσοχή: Έχει σημασία το όνομα του αρχείου να ταιριάζει με το όνομα της τάξης καθώς και η προέκταση του αρχείου να είναι *.java*. Αν ο editor προσθέτει διαφορετική επέκταση πρέπει να την διορθώσουμε.

Μεταγλώττιση του πηγαίου κώδικα: Η μεταγλώττιση γίνεται από τον Java μεταγλωττιστή που έχει το όνομα javac. Ο μεταγλωττιστής μεταφράζει τον Java πηγαίο κώδικα σε bytecodes (η γλώσσα που κατανοεί και εκτελεί ο Java διερμηνευτής). Ο μεταγλωττιστής δεν δημιουργεί εκτελέσιμο κώδικα της μηχανής εκτέλεσης, αλλά μιας υποθετικής μηχανής που έχει σαν εκτελέσιμο κώδικα τον bytecodes. Κάθε μηχανή που διαθέτει Java διερμηνευτή μπορεί να εκτελέσει bytecodes. Για τη μεταγλώττιση δίνουμε την εντολή:
java <όνομα προγράμματος>.java
στη γραμμή εντολών του συστήματος (DOS prompt για Windows 95 ή NT και shell prompt για Unix). Αν υπάρχουν λάθη, ο μεταγλωττιστής τα εντοπίζει και τα αναφέρει.

Αν δεν υπάρχουν λάθη, δημιουργεί ένα αρχείο με το όνομα του προγράμματος και με επέκταση .class, που περιέχει την αναπαράσταση του προγράμματος σε bytecodes και αποτελεί το εκτελέσιμο πρόγραμμα.

Εκτέλεση του προγράμματος: Η εκτέλεση γίνεται με την χρήση του Java διερμηνευτή, τον οποίο ενεργοποιούμε με την εντολή

java <όνομα προγράμματος>

Δεν απαιτείται η προσθήκη της επέκτασης .class.

Με την εκτέλεση του Java διερμηνευτή το σύστημα κατανοεί και εκτελεί bytecodes. Ο **Java διερμηνευτής** αποτελείται από τα εξής τρία τμήματα:

- **class loader:**

Αναλαμβάνει να φορτώσει στην μνήμη του υπολογιστή
τον bytecode κώδικα είτε από το τοπικό σύστημα αρχείων είτε διαμέσου του δικτύου

- **bytecode verifier:**

Αναλαμβάνει να ελέγξει κάθε τμήμα (κλάση) bytecode κώδικα που προέρχεται από το δίκτυο, για την διασφάλιση της μη παράβασης των περιορισμών ασφαλείας που ορίζει το Java περιβάλλον. Είναι ευνόητο ότι το Java περιβάλλον πρέπει να διαθέτει περιορισμούς που θα απαγορεύει την πρόκληση ζημιών στο σύστημά μας από εφαρμογές που κατεβάζουμε από το δίκτυο, όταν αυτές εκτελούνται

- **interpreter:**

Αυτό το μέρος διερμηνεύει τον bytecode κώδικα, εντολή προς εντολή, σε γλώσσα μηχανής που εκτελείται από την CPU του συστήματος.

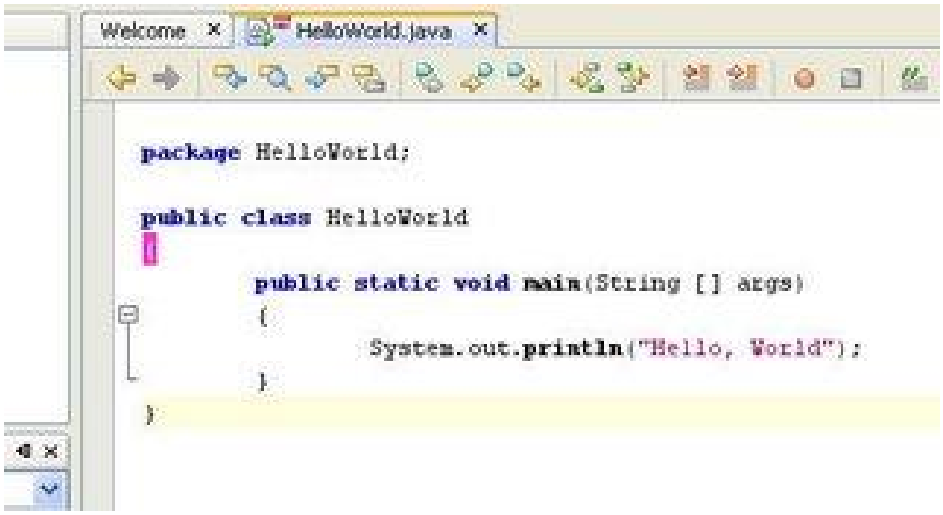
Με το τέλος κάθε τμήματος, υπάρχει περίπτωση να εντοπιστούν λάθη, τα οποία αναφέρονται από το σύστημα, διορθώνονται από τον προγραμματιστή και η διαδικασία επαναλαμβάνεται έως ότου έχουμε επιτυχές αποτέλεσμα.

3) Το πρώτο πρόγραμμα σε Java

Ήρθε η ώρα να γράψετε το πρώτο σας πρόγραμμα πιο πάνω αναφέραμε τα βήματα αλλά θα τα ξαναδούμε. Τώρα ας γράψουμε το πρώτο μας πρόγραμμα (κάτω από την γραμμή package HelloWorld) όπως το βλέπεται παρακάτω Δίνοντας πάντα προσοχή στην εναλλαγή πεζών κεφαλαίων.

```
public class HelloWorld

{

public static void main(String [ ] args)

{

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

}

}
```

Το ολοκληρωμένο πρόγραμμα φαίνεται στην παρακάτω οθόνη.

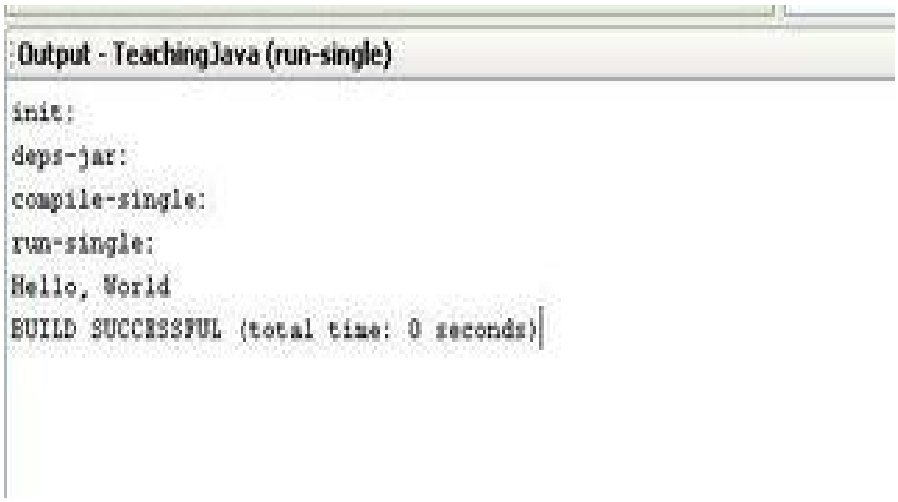


Εικόνα 10

Είσαστε ένα βήμα πριν τρέξετε το πρόγραμμα σας. Αν κοιτάξετε προσέχτηκα αριστερά, κάτω από το πακέτο HelloWorld, υπάρχει ένα αρχείο, το όποιο είναι το πρόγραμμα που μόλις γράψατε. Κάνετε δεξί κλικ επάνω του και από τις επιλογές που εμφανίζονται, διαλέξτε πρώτα το “compile file”. Αν δεν έχετε λάθη στο πρόγραμμα σας τότε στο κάτω μέρος της οθόνης θα δείτε ότι η διαδικασία έλεγχου για τυχών λάθη ήταν επιτυχής (BUILD SUCCESSFUL) και το πρόγραμμα σας είναι άριστα γραμμένο.

Εικόνα 11

Κάνετε πάλι δεξί κλικ επάνω στο πρόγραμμα HelloWorld.java κάτω σας το πακέτο HelloWorld και σας τη φορά διαλέγουμε την επιλογή “Run File”. Το αποτέλεσμα θα το δείτε πάλι στο κάτω μέρος σας οθόνης σας.



Εικόνα 12

Αν όλα έχουν πάει καλά με την εγκατάσταση του NetBeans, τότε ήδη έχετε τρέξει το πρώτο σας πρόγραμμα σε γλώσσα προγραμματισμού java. Ξεκινήστε τώρα να αναλύσετε τις εντολές και την θεωρία στις οποίες θα στηριχτείτε για να δημιουργήσετε χωρίς λάθη και κατά μεγάλο βαθμό πολύ αποδοτικές εφαρμογές.

Στον προγραμματισμό, ανεξάρτητου ποια γλώσσα χρησιμοποιούμε, πάντα περιοριζόμαστε από κάποιους όρους και κανόνες που υπάρχουν. Στην Java, ένας από αυτούς τους περιορισμούς είναι η μη χρησιμοποίηση των keywords (λέξεις-κλειδιά που χρησιμοποιεί αποκλειστικά η java για να εκτελέσει συγκεκριμένες

εντολές). Για παράδειγμα, δεν μπορείτε να χρησιμοποιήσετε την λέξη public σαν όνομα δικής σας μεταβλητής στο πρόγραμμα σας, γιατί η συγκεκριμένη λέξη χρησιμοποιείται από την java σαν ειδικός Όρος που δείχνει το είδος πρόσβασης σε μια κλάση. Ο πινάκας που ακολουθεί δείχνει όλες τις λέξεις που ο προγραμματιστής δεν μπορεί να χρησιμοποιήσει για ονόματα μεταβλητών.

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

Επίσης υπάρχουν ακόμα τρεις λέξεις στην Java που είναι δεσμευμένες: true, false, και null.

Μεταβλητές ονομάζονται οι λέξεις που φτιάχνει ο χρηστής μέσα στο πρόγραμμα του και μπορούν να περνούν διαφορές τιμές. Παράδειγμα, x = 5 ή bankAccount=500. Οι χ και BankAccount είναι μεταβλητές που όπως τρέχει το πρόγραμμα θα πάρουν διαφορές τιμές. Πρέπει να οριστούν οι κανόνες ονομασίας μιας μεταβλητής.

Πρώτος κανόνας, μια μεταβλητής δεν μπορεί να έχει το ίδιο όνομα με καμία από τις ήδη αναφερόμενες παραπάνω προκαθορισμένες λέξεις κλειδιά.

Δεύτερος κανόνας, μια μεταβλητής αποτελείται από αλφαβητικούς χαρακτήρες, αριθμούς (από 0 μέχρι 9), το underscore (_) σύμβολο, η το σύμβολο του δολαρίου ($).

Τρίτος και τελευταίος κανόνας είναι η απαγόρευση να χρησιμοποιηθεί αριθμός σαν ο πρώτος χαρακτήρας. Για να δώσουμε ένα πιο ξεκάθαρο παράδειγμα η μεταβλητής με το όνομα Skywalker1 είναι δεκτή, ενώ η μεταβλητής 1Floor δεν είναι γιατί ξεκάνει το όνομα με αριθμούς.

Φυσικά οι μεταβλητές θα χρησιμοποιηθούν ίσως περισσότερο από μια φορά μέσα στο ίδιο πρόγραμμα. Λογικό θα ήταν λοιπόν, να δίναμε ονόματα στις μεταβλητές που έχουν μια λογική έννοια. Για παράδειγμα, αν θέλουμε να ονομάσουμε μια μεταβλητής η οποιαδήποτε παίρνει σαν τιμή ημερομηνίες, το όνομα Date θα ήταν το καταλληλότερο. Οπότε τώρα γνωρίζοντας τους κανόνες μπορείτε ν δώσετε ένα όνομα στη μεταβλητή σας.

Σε περίπτωση που δεν τρέξει το πρόγραμμα σας ακολουθήστε τα εξής βήματα:

1. Εφόσον το πρόβλημα είναι στο compilation, δηλαδή έχετε κάποιο συντακτικό λάθος κοιτάξτε το μήνυμα που σας βγάζει. Γενικά τα διαγνωστικά μηνύματα στη Java είναι αρκετά διαφωτιστικά και σας λένε σε ποια γραμμή ακριβώς υπάρχει το λάθος. Προσπαθήστε να ακολουθήσετε τις συμβουλές που σας δίνει η Java. Αν ο compiler διαμαρτύρεται συνέχεια λέγοντας πως η συγκεκριμένη κλάση πρέπει π.χ. να είναι abstract το πιθανότερο είναι πως έχει δίκιο.
2. Στο 99% των περιπτώσεων έχετε ξεχάσει κάποιο ερωτηματικό, έχετε βάλει κάποια παραπάνω ή μία λιγότερη παρένθεση, έχετε ξεχάσει το άγκιστρο που κλείνει το block του κώδικα } ή έχετε γράψει κάποιο όνομα μεταβλητής με λάθος τρόπο (θυμηθείτε πως η Java είναι case sensitive).
3. Στις υπόλοιπες περιπτώσεις αυτό που συμβαίνει είναι πως συνήθως η Java δε βρίσκει την κλάση που της ζητάτε να χρησιμοποιήσει. Βεβαιωθείτε πως στο αρχείο που χρησιμοποιείτε κάποια κλάση αν αυτή δεν είναι στο standard package java.lang πρέπει να κάνετε import το package στο οποίο ανήκει. Π.χ. η κλάση Vector ανήκει στο java.util, οπότε και στην κορυφή του αρχείου που χρησιμοποιείτε την κλάση αυτή θα πρέπει να γράψετε import java.util.*;

4. Αν ο compiler συνεχίσει να παραπονιέται πως δε βρίσκει κάποια κλάση ελέγξτε αν είσαστε στο σωστό directory για να κάνετε compile ή για να τρέξετε το πρόγραμμα, και αν παρόλα αυτά συνεχίσει τα παράπονα ελέγξτε πως έχετε θέσει σωστά το environment variable CLASSPATH (για λεπτομέρειες για το CLASSPATH κοιτάξτε τις πληροφορίες για **installation** της Java)
5. Προσέξτε πώς χρησιμοποιείτε τα exceptions και από πού αυτά ξεκινούν και που καταλήγουν. Αν μία εντολή έχει τη δυνατότητα να κάνει throw κάποιο exception η Java θα σας ζαλίσει γιατι σας υποχρεώνει είτε να το χειριστείτε π.χ. με ένα γενικό try {...} catch (Exception e) {...} μπλοκ, είτε να δηλώσετε πως η συνάρτηση στην οποία βρίσκεται η εντολή αυτή έχει τη δυνατότητα να κάνει και αυτή throw ένα exception.
6. Αν στο πρόγραμμά σας υπάρχουν λογικά λάθη, δηλαδή το πρόγραμμα δουλεύει χωρίς να τερματίζεται χωρίς λόγο, αλλά από την άλλη δε σας βγάζει τα αναμενόμενα αποτελέσματα, δοκιμάστε να βρείτε με το μυαλό σας το σημείο στο οποίο μπορεί να υπάρχει το λογικό λάθος. Ξανασκεφτείτε τον αλγόριθμο που δώσατε και ξανακοιτάξτε αυτό που γράψατε. Δοκιμάστε να πάρετε πληροφορίες από το πρόγραμμά σας βάζοντας εντολές System.out.println() σε κατάλληλα σημεία πριν καταφύγετε σε βαριά περιβάλλοντα debugging.
7. Βεβαιωθείτε πως γνωρίζετε κάθε στιγμή το πού θα πρέπει να είναι ο έλεγχος του προγράμματος καθώς επίσης και το ποια συνάρτηση ακριβώς εκτελείται όταν είστε σε μία κλάση η οποία κληρονομεί από άλλη μία, αυτή με τη σειρά της από άλλη κ.λπ.
8. Δοκιμάστε να ακολουθήσετε μία διαφορετική προσέγγιση σε αυτό που προσπαθείτε να υλοποιήσετε. Ίσως να μην έχετε καταλάβει πώς ακριβώς δουλεύει κάποια κλάση που έχει η Java, να έχετε σχεδιάσει κάτι με διαφορετικό τρόπο από αυτό που θα περίμενε κανείς ή να μην είναι τόσο απλό να εκφράσετε αυτό που έχετε στο μυαλό σας.
9. Προσέξτε το σύνηθες λάθος στα for, while π.χ. ο παρακάτω κώδικας δε θα τυπώσει ποτέ 100 αριθμούς και μπορείτε να φάτε αρκετή ώρα ψάχνοντας το γιατί ...

```
for (i = 0; i < 100; i++); // όχι ερωτηματικό!!!
    System.out.println(i);
```

10. Μην ξεχνάτε να φτιάχετε τους κατάλληλους constructors για τις κλάσεις που δημιουργήσατε. Επίσης μην ξεχνάτε εκεί μέσα να δημιουργείτε τα αντικείμενα τα οποία αποτελούν πεδία στην κλάση σας. Ένα Null Pointer Exception σημαίνει πως κάτι δεν έχετε δημιουργήσει ή αρχικοποιήσει σωστά.

4) Μέθοδοι

Η Java χρησιμοποιεί μεθόδους για να τρέξει τις κλάσεις. Με την εντολή public, το κείμενο που ακολουθεί θα εμφανίζεται. Με την εντολή static το κείμενο ανήκει περισσότερο στην κλάση και λιγότερο στο αντικείμενο, ενώ με την εντολή void δεν μεταβάλει καμιά μεταβλητή.
Public static void main (string *args[1])

Κλήση μεθόδων

- Καλούμε τις μεθόδους ενός αντικειμένου με τη σύνταξη
  αντικείμενο.μέθοδος(...)

Παράδειγμα

```
import java.util.*;

class TestCal {
	public static void main(String argv[]) {
		GregorianCalendar then = new GregorianCalendar();

		then.set(1973, 10, 17);
		System.out.println(then.getTimeInMillis());

		GregorianCalendar now = new GregorianCalendar();
		Date d = new Date();
		now.setTime(d);
		System.out.println(now.getTimeInMillis());
	}
}
```

Μέθοδοι κατασκευαστές

- Σε κάθε κλάση μπορούν να οριστούν μέθοδοι κατασκευαστές (*constructors*) με όνομα ίδιο με αυτό της κλάσης.

- Η μέθοδος κατασκευής καλείται κάθε φορά που δημιουργείται ένα νέο αντικείμενο (με τη χρήση new ObjectType).
- Η μέθοδος καταστροφής καλείται κάθε φορά που παύει να υπάρχει ένα αντικείμενο δηλαδή αντίστοιχα όταν τελειώνει το πρόγραμμα, όταν η ροή βγαίνει από το τοπικό τμήμα, ή όταν το σύστημα αποκομιδής αχρήστων της Java καταστρέφει το συγκεκριμένο αντικείμενο.
- Το όρισμα που δηλώνουμε στη μέθοδο κατασκευής επιτρέπει προσδιορισμό ιδιοτήτων του αντικειμένου που δημιουργούμε (π.χ. τον αριθμό στοιχείων σε έναν πίνακα) ή αρχικών τιμών.
- Για το ίδιο αντικείμενο μπορεί να έχουν οριστεί πολλές μεθόδοι κατασκευής με διαφορετικά ορίσματα.
- Η κλήση της συνάρτησης κατασκευής μπορεί να γίνει κατά τον ορισμό μιας μεταβλητής με τη μορφή "κλάση μεταβλητή = new κλάση(όρισμα)".

Παράδειγμα

```
import java.util.*;

class TestCal {
	public static void main(String argv[]) {
		GregorianCalendar then = new GregorianCalendar(1973, 10, 17);
		System.out.println(then.getTimeInMillis());
	}
}
```

Ιδιότητες και μέθοδοι κλάσης

- Σε μια κλάση μπορούν να δηλωθούν (με τον προσδιορισμό static) (πεδία) τα οποία υπάρχουν μόνο μια φορά για όλη την κλάση.
- Επίσης μπορούν να δηλωθούν αντίστοιχες μέθοδοι που μπορούν να κληθούν με τη σύνταξη κλάση.συνάρτηση.
- Τα πεδία αυτά χρησιμοποιούνται για την επεξεργασία στοιχείων που αφορούν ολόκληρη την κλάση και όχι τα αντικείμενά της.

Παράδειγμα

```
/* Compile with javac -encoding cp1253 FindDay.java */
import java.util.*;
import java.io.*;

class FindDay {
	/** Return the name of the given numeric week day */
	public static String weekDayName(int weekNumber) {
		switch (weekNumber) {
		case Calendar.MONDAY: return "Δευτέρα";
		case Calendar.TUESDAY: return "Τρίτη";
		case Calendar.WEDNESDAY: return "Τετάρτη";
		case Calendar.THURSDAY: return "Πέμπτη";
		case Calendar.FRIDAY: return "Παρασκευή";
		case Calendar.SATURDAY: return "Σάββατο";
		case Calendar.SUNDAY: return "Κυριακή";
		}
		return null;
	}

	public static void main(String args[]) throws Exception {
		// Are appropriate arguments given?
		if (args.length != 3) {
			System.err.println("usage: FindDay year month day");
			System.exit(1);
		}

		// Parse year, month, day
		int year = Integer.parseInt(args[0]);
		int month = Integer.parseInt(args[1]) - 1;
		int monthDay = Integer.parseInt(args[2]);

		// Set the calendar and calculate the day name
		GregorianCalendar d = new GregorianCalendar(year, month, monthDay);
		String dn = weekDayName(d.get(Calendar.DAY_OF_WEEK));
```

```
            // Create an output channel for Greek characters
            PrintWriter out = new PrintWriter(
                  new OutputStreamWriter(System.out, "cp737"), true);
            // Print the day name
            out.println(dn);
      }
}
```

Μέθοδοι όλων των αντικειμένων

Όλες οι κλάσεις της Java υποστηρίζουν αντικείμενα με τις παρακάτω μεθόδους:

Object clone()

Επιστρέφει ένα αντίγραφο του αντικειμένου

String to String()

Επιστρέφει μια συμβολοσειρά που παριστάνει το αντικείμενο

boolean equals(Object obj)

Επιστρέφει true αν το αντικείμενο είναι "ίσο" με το obj

5) Αριθμοί και σύμβολα

Οι αριθμοί ξεκινούν με ένα από τα δέκα ψηφία. Οι ακέραιοι, όπως 1,1432 και 1134224, αντιμετωπίζονται ως ακέραιοι χωρίς κάποιους επιπρόσθετους χαρακτήρες. Οι μεγάλοι ακέραιοι, οι αριθμοί κινητής υποδιαστολής και οι οκταδικές και δεκαεξαδικές τιμές χρειάζονται επιπρόσθετα γράμματα για να ερμηνευθούν από τον compiler. Για να τα αναγνωρίσει πρέπει να χρησιμοποιήσει το γράμμα f μετά τον αριθμό π.χ x=3,1415f. Οι αριθμοί που γράφονται με εκθετικό τρόπο χρησιμοποιούν είτε το γράμμα e είτε το E π.χ x=6,023e23f.Οι μεγάλοι ακέραιοι ακολουθούνται από ένα l ή L π.χ longvar=132312312412L.Οι δεκαεξαδικοί ξεκινούν με 0x ή 0X και έχουν τη μορφή x=0xDEAD14.Οι οκταδικοί ξεκινούν με ένα 0. Αυτά όμως θα τα δούμε πιο αναλυτικά στο επόμενο κεφάλαιο.

Τύπος	**Μέγεθος**	**Εύρος**	**Σύνταξη**
byte	8 bits	-2 ως 2 -1	121
short	16 bits	-2 ως 2 -1	4023

int	32 bits	-2 ως 2 -1	412235
long	64 bits	-2 ως 2 -1	622312882881
float	xx bits	-άπειρο ως άπειρο	6,023e23i
double	xx bits	-άπειρο ως άπειρο	1,23414234324D

Στην Java, ο αριθμός των bits που χρησιμοποιούνται για έναν αριθμό τύπου int είναι σταθερός. Υπάρχουν επίσης τιμές με τη μορφή γραμμάτων, οι οποίες χρησιμοποιούνται με χαρακτήρες(σε απλά εισαγωγικά) και συμβολοσειρές(σε διπλά εισαγωγικά).

Ειδικοί χαρακτήρες

Σύμβολο	Χαρακτήρας	Σύμβολο	Χαρακτήρας
b	backspace	‘	απλά εισαγωγικά
n	αλλαγή γραμμής	“	διπλά εισαγωγικά
r	carriage return	\	ανάποδη κάθετος
t	στηλοθέτης	u	έκδοση
f	form feet		

Μερικές συμβολοσειρές μπορεί να έχουν την παρακάτω μορφή:
message =Bob was ‘strange’;
message=Name t Rank t Serial Number;
message=Macintosh is a trademark(2122)of Apple.;

Στο πρώτο παράδειγμα τοποθετούνται απλά εισαγωγικά εντός της συμβολοσειράς, στο δεύτερο χρησιμοποιούνται στηλοθέτεςγια τη δημιουργία επικεφαλίδων στηλών και στο τρίτο ένας χαρακτήρας UNICODE για το σήμα κατατεθέν.

6) Δεδομένα στη Java

Η σύνταξη της είναι σχεδόν εξ ολοκλήρου δανεισμένη από τη C και τη C++. Ο run-time interpreter της Java είναι σημαντικά διαφορετικός, όσον αφορά τον τρόπο με τον

οποίο διαχειρίζεται τα δεδομένα. Βρίσκεται πλησιέστερα στη LISP από οποιαδήποτε έκδοση της C.

Η μεγαλύτερη διαφορά είναι ότι ο μηχανισμός run- time της Java από της C παρακολουθεί τη χρήση της μνήμης για λογαριασμό του προγραμματιστή. Άλλες διαφορές είναι ότι η Java είναι πολύ αυστηρότερη αντικειμενοστρεφής γλώσσα. Μερικές μορφές δεδομένων δεν είναι διαθέσιμες. Άν θέλετε να δημιουργήσετε μια δομή δεν μπορείτε να το κάνετε απευθείας. Πρέπει να δημιουργήσετε μια κλάση και να προσθέσετε σε αυτή μεταβλητές-δείγματα.

ΕΠΑΝΑΛΗΠΤΙΚΕΣ ΕΡΩΤΗΣΕΙΣ 2^{ΟΥ} ΚΕΦΑΛΑΙΟΥ

Ερωτήσεις:

1. Ποιος ο ρόλος του project στη δημιουργία ενός προγράμματος; Θα μπορούσατε σε ένα project να αποθηκεύσετε περισσότερα από ένα προγράμματα;

_____

_____

_____

2. Είναι απαραίτητη η δημιουργία κλάσης για τη δημιουργία ενός προγράμματος; Θα μπορούσαν να αποθηκευτούν δύο διαφορετικές κλάσεις στο ίδιο πακέτο;

_____

_____

_____

3. Ποια είναι τα εργαλεία από τα οποία αποτελείτε το βασικό περιβάλλον της Java;

_____

_____

_____

4. Θα ήταν ορθό να ονομάσετε μία κλάση null;

_____

_____

_____

Ασκήσεις:

1. Δημιουργήστε ένα project με την ονομασία ergasia. Σε αυτό αποθηκεύστε μία κλάση, αφού δημιουργήσετε το πακέτο, και δώστε της το όνομα teipatras

2. Μέσα στην κλάση δημιουργήστε ένα πρόγραμμα με το όνομα teipatras.

3. Γράψτε μια εφαρμογή που να εμφανίζει το όνομα και τη διεύθυνση στην οθόνη, με μια ξεχωριστή γραμμή για κάθε τμήμα της διεύθυνσης.(Χρήση των εργαλείων javac-java).

```
Class Address
{
 puplic static void main(String[ ] args)
{
  System.out.println("ΕΚΔΟΣΕΙΣ Κ");
  System.out.println("ΚΟΡΙΝΘΟΥ");
 System.out.println("GR 60 ΑΘΗΝΑ");
System.out.println ("GREECE");
  }
}
```

Στο τρίτο κεφάλαιο θα κάνουμε μια εισαγωγή στον αντικειμενοστραφή προγραμματισμό. Θα επεξηγήσουμε την έννοια του αντικειμένου στο πρόγραμμα και πως αυτά αλληλεπιδρούν για την παραγωγή ενός προγράμματος.

Θα δούμε ακόμα τους τύπους δεδομένων της Java για να γίνει πιο κατανοητός ο τρόπος γραφής και χρήσης του κώδικά της.

Έπειτα θα δούμε τις κλάσεις των αντικειμένων στη Java. Από τη δημιουργία μιας κλάσης με τον κατάλληλο κώδικα μέχρι τη λειτουργία τους στην κατασκευή προγραμμάτων.

ΚΕΦΑΛΑΙΟ 3

1) Αντικείμενα στο λογισμικό

Το λογισμικό αποτελείται από αντικείμενα, όπως και ο πραγματικός κόσμος. Τα αντικείμενα του πραγματικού κόσμου περιγράφονται από *χαρακτηριστικά* και *ιδιότητες*. *Χαρακτηριστικά* των αντικειμένων, είναι τα ιδιαίτερα χαρακτηριστικά που έχει ένα αντικείμενο, ενώ οι *ιδιότητες* αφορούν την ταξινόμηση, η οποία δείχνει τους διαφορετικούς τύπους αντικειμένων, την αφαίρεση, στην οποία εξειδικεύουμε τα κριτήρια για το διαχωρισμό των αντικειμένων και την κληρονομικότητα, δηλαδή τη μεταφορά των χαρακτηριστικών στις κατώτερες ιεραρχικά κλάσεις.

Τα αντικείμενα στο λογισμικό περιγράφονται από τις σχέσεις ανάμεσα στα αντικείμενα διαφορετικών κλάσεων. Σ' ένα αντικειμενοστρεφές πρόγραμμα, τα αντικείμενα ανταλλάσουν μηνύματα μεταξύ τους. Τα μηνύματα μπορούν να είναι είτε ***εντολής*** είτε ***ερώτησης***. Τα μηνύματα αποστέλλονται χρησιμοποιώντας το όνομα του αντικειμένου-παραλήπτη. Ανάμεσα στο όνομα του παραλήπτη και το μήνυμα μπαίνει μία τελεία. Η γενική μορφή ενός Java μηνύματος είναι:

object_name. message name (…);

Στα εισαγωγικά μπορούν να περιλαμβάνονται παράμετροι με επιπλέον πληροφορίες για το μήνυμα αν ζητούνται, διαφορετικά είναι άδεια.

Επικοινωνία μπορεί επίσης να υπάρχει μεταξύ ανθρώπου και αντικειμένων. Σ' ένα σύστημα (use case) μηνύματα ανταλλάσσονται μεταξύ του ανθρώπου-χρήστη και των αντικειμένων. Το Java μήνυμα που αποστέλλει ο χρήστης είναι της μορφής:

object. message ( );

Παράδειγμα:

Αν θελήσουμε να βάλουμε αυτόματο χρονόμετρο στο κλιματιστικό μας, τότε θα στείλουμε ένα μήνυμα στο μετρητή του κλιματιστικού, το οποίο θα αποτελεί και το αντικείμενο με το οποίο θα ανταλλάξουμε μηνύματα. Έτσι ως χρήστης θα στείλουμε το μήνυμα :

κλιματιστικό_μετρητής .χρονομέτρηση( );

Το μήνυμα αυτό είναι ερωτηματικό γιατί περιμένει απάντηση.

Μπορούμε όμως στο μήνυμα να βάλλουμε και κάποιες παραμέτρους. Για παράδειγμα, μπορούμε να στείλουμε ένα μήνυμα στο κλιματιστικό, που να του ζητάει να λειτουργήσει μόνο για 30 λεπτά. Το μήνυμα που θα στείλουμε στο μετρητή του κλιματιστικού θα είναι:

κλιματιστικό_μετρητής.χρονομέτρηση(30);

Αν όμως θέλουμε να χρησιμοποιήσουμε και άλλες μονάδες του χρόνου, για να ζητήσουμε από το μετρητή να λειτουργήσει, όπως για παράδειγμα αν θέλουμε να λειτουργήσει για μια ώρα και 15 λεπτά, το μήνυμα που θα στείλουμε θα είναι:

κλιματιστικό_μετρητής.χρονομέτρηση(1,15,0);

Δημιουργία νέων αντικειμένων

Κάθε αντικείμενο της κλάσης έχει ένα δικό του αντίγραφο των πεδίων που έχουμε ορίσει.

Σε αντίθεση με τους βασικούς τύπους της Java (int, boolean, double) που ο ορισμός τους καθορίζει και την περιοχή της μνήμης που φυλάσσονται τα αντίστοιχα στοιχεία, οι μεταβλητές που ορίζονται ως αντικείμενα περιέχουν απλώς έναν δείκτη σε περιοχή της μνήμης που πιθανώς να περιέχει τα στοιχεία για το συγκεκριμένο αντικείμενο.

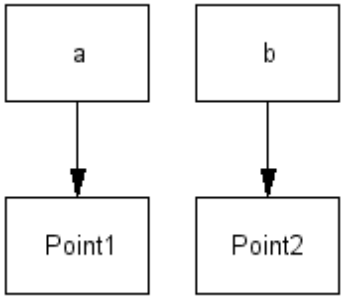
Αφού ορίσουμε μια μεταβλητή ως αντικείμενο, πρέπει να καθορίσουμε και την περιοχή της μνήμης στην οποία θα φυλάσσονται τα πεδία του. Αυτή μπορεί να είναι:

Νέα περιοχή μνήμης οριζόμενη με τη σύνταξη

```
objectVariable = new ObjectType();
```

Παράδειγμα:

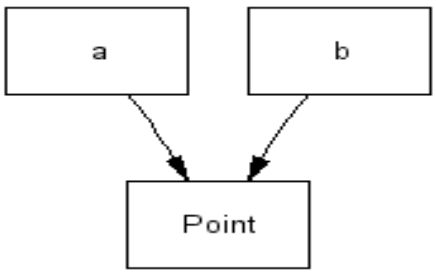
```
Point a, b;
a = new Point();
b = new Point();
```



Αντιγράφοντας στο συγκεκριμένο αντικείμενο κάποιο άλλο το οποίο έχει ήδη οριστεί:
Παράδειγμα:

```
Point a, b;
a = new Point();
b = a;
```

Στην περίπτωση αυτή, τα δύο αντικείμενα μοιράζονται (*share*) την ίδια περιοχή της



μνήμης.

2) Αντικειμενοστραφής προγραμματισμός

Ο αντικειμενοστραφής προγραμματισμός, (OOP - Object Oriented Programming), είναι μια προγραμματιστική φιλοσοφία όπως και ο προστακτικός ή ο λογικός προγραμματισμός.

Σύμφωνα με τον OOP ένα πρόγραμμα δεν αποτελείται από τα δεδομένα και τον κώδικα που τα επεξεργάζεται αλλά από αντικείμενα (objects) τα οποία εμπεριέχουν

τα δεδομένα και τα οποία (αντικείμενα) ανταλλάσουν μεταξύ τους πληροφορίες και μηνύματα προκειμένου να επιτευχθεί ο στόχος του προγράμματος. Για παράδειγμα έστω ότι έχουμε ένα σύστημα που περιλαμβάνει μια βάση δεδομένων καθώς και user interfaces με τα οποία οι χρήστες επικοινωνούν και αλληλεπιδρούν με την Β.Δ.. Μια αντικειμενοστραφής (Object Oriented - OO) γλώσσα προγραμματισμού, μπορεί να δημιουργεί ταχύτερα καλύτερο και περισσότερο αξιόπιστο λογισμικό. Υπάρχει σημαντική διαφορά μεταξύ των παραδοσιακών και των αντικειμενοστραφών γλωσσών προγραμματισμού.

Εάν θέλουμε να δημιουργήσουμε μια παραδοσιακή γλώσσα, θα πρέπει να γράψουμε κώδικα για την σχεδίαση του πλήκτρου στην οθόνη, για την εμφάνιση και κίνηση του δείκτη του ποντικιού, για την παρακολούθηση του πότε θα γίνει η ενεργοποίηση, για την σχεδίαση του πλήκτρου όταν αυτό είναι πατημένο και στο τέλος για τις ενέργειες που θα πρέπει να γίνουν μόλις πατηθεί το πλήκτρο. Και αν θέλουμε να δημιουργήσουμε και δεύτερο πλήκτρο, όλα αυτά θα χρειαστεί να τα ξανακάνουμε. Ενώ, με την Java, όπως και με τις περισσότερες αντικειμενοστραφείς γλώσσες προγραμματισμού, επειδή γνωρίζουν τι είναι πλήκτρο οθόνης, την εμφάνισή του και πως αλληλεπιδρά με το ποντίκι, το μόνο που έχουμε να κάνουμε είναι να καθορίσουμε την θέση του στην οθόνη, τι να γράφει πάνω του και τι να κάνει όταν ενεργοποιείται.

3) Γιατί είναι η Java Αντικειμενοστρεφής;

Η Java είναι μια απλή γλώσσα προγραμματισμού, γι’ αυτό χρησιμοποιεί αντικείμενα που βοηθούν στη μείωση της πολυπλοκότητας του λογισμικού τη στιγμή που οι απαιτήσεις για τις λειτουργίες του λογισμικού είναι αυξημένης πολυπλοκότητας. Οι αντικειμενοστρεφείς τεχνικές είναι απαραίτητες για τη δημιουργία κώδικα σε Intenet και πολυμέσα, στο πλαίσιο των οποίων έχει σχεδιαστεί η Java.

4) Αντικείμενα δεδομένων

Τα αντικείμενα δεδομένων είναι σύνολα από αριθμούς ή και χαρακτήρες τα οποία αποθηκεύονται με τη μορφή δεδομένων. Η Java παρέχει αυτές τις βασικές κλάσεις,

που περιέχουν μια ευρεία γκάμα μεθόδων που μπορούν να φανούν αρκετά χρήσιμες:

Βασικός τύπος	**Σχετική κλάση**
boolean	Boolean
char	Character
int	Integer
long	Long
float	Float
double	Double

Οι Boolean χαρακτήρες αποθηκεύονται ως δεδομένα. Οι intValue(), longValue, floatValue, doublevalue επιστρέφουν την τιμή που είναι αποθηκευμένη στο αντικείμενο του κατάλληλου τύπου. Αν κριθεί κατάλληλο, η Java θα στρογγυλοποιήσει τους αριθμούς.

5) Συμβολοσειρές

Είναι αντικείμενα με τη δική τους κλάση και συλλογή μεθόδων, είναι όμως αρκετά διαφορετικά από όλα τα αντικείμενα που θα δημιουργήσετε. Πολλές από τις μεθόδους για τη δημιουργία και το χειρισμό συμβολοσειρών αποτελούν τμήμα της βασικής σύνταξης της Java. Έτσι μπορείτε να πληκτρολογήσετε ‘pre’+fix’, για να ενώσετε δύο συμβολοσειρές.

Χειρισμός: Οι βασικοί τελεστές που χρησιμοποιούνται για το χειρισμό των συμβολοσειρών είναι το σύμβολο ισότητας(=), πρόσθεσης(+), και ο συνδυασμός τους (+=).Το σύμβολο πρόσθεσης αποτελεί τη συντομογραφία της συνένωσης συμβολοσειρών και χρησιμοποιείται για την ένωση 2 συμβολοσειρών:

```
String a, b, c, d;
a=’alpha’;
b=’beta’;
c=a+””’+b
d=a;
d+=””’;
```

d+=b;

Τόσο η c όσο και η d περιέχουν στο τέλος τις ίδιες πληροφορίες. Μπορείτε ακόμα να αναμίξετε και αριθμούς, τους οποίους η Java θα μετατρέψει αυτόματα στο σωστό τύπο. Η συμπεριφορά αυτή είναι κάπως ανορθόδοξη, επειδή η Java κανονικά επιμένει στους σαφείς καθορισμούς τύπων και οποιασδήποτε μετατροπής από ένα τύπο σε άλλον.

Παράδειγμα:

int age=10;

String message;

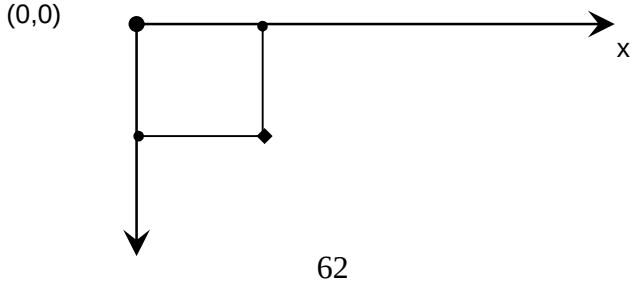
message='Bob is'+'years old today.Habby Birthday.';

Εδώ ο compiler της Java καλεί μυστικά τη μέθοδο toString η οποία είναι διαθέσιμη σε πολλούς από τους πιο συχνά χρησιμοποιούμενους τύπους και αντικείμενα. Στη κλάση String, μετατρέπει τα δεδομένα σε συμβολοσειρά. Σε άλλες κλάσεις μετατρέπει τα δεδομένα απλά στο τύπο τους.

6) Γραφικά στη Java

Το πακέτα γραφικών της Java αποτελεί ένα αρκετά άμεσο σύστημα σχεδίασης, το οποίο πρέπει να είναι οικείο σε οποιονδήποτε έχει γράψει προγράμματα για μικρουπολογιστές.Οι ρουτίνες γραφικών της Java είναι παρόμοιες με αυτές του QuickDraw της Apple ή με το Windows toolkit της Microsoft, χωρίς όμως να είναι τόσο εκτενείς.

Η Java παρέχει τη δυνατότητα αλλαγής χρωμάτων και σχεδιασμού γραμμών, παραλληλογράμμων και ελλείψεων. Η μόνη σημαντική διαφορά ανάμεσα στις μεθόδους γραφικών της Java και στις άλλες είναι η αρχή των αξόνων (0,0) της Java στο σύστημα συντεταγμένων βρίσκεται στην επάνω αριστερή γωνία. Ο άξονας των x αυξάνεται οριζόντια από τα αριστερά προς τα δεξιά, αλλά ο άξονας y αυξάνει από επάνω προς τα κάτω.



y

Java.awt.Graphics: Το πακέτο γραφικών Graphics της Java περιλαμβάνει πολλές διαφορετικές μεθόδους σχεδιασμού γραφικών, οι οποίες ανήκουν στη κλάση Graphics. Σε πολλές περιπτώσεις στη μέθοδο paint του applet σας θα δοθεί ένα αντικείμενο από τη κλάση Graphics.Αν θελήσετε να σχεδιάσετε σε αυτό, θα εκτελέσετε κάποιες μεθόδους της κλάσης. Αν θέλετε να δημιουργήσετε τα δικά σας αντικείμενα Graphics αυτό θα γίνει σε περιπτώσεις που επιθυμείτε υψηλή απόδοση για τη δημιουργία καλής οπτικής εντύπωσης σε κάποια animation.

Η μέθοδος drawLine: Αν θέλετε να σχεδιάσετε μια γραμμή, η μέθοδος αυτή δέχεται 4 παραμέτρους: τις συντεταγμένες x και y της αρχής, και τις συντεταγμένες x και y του τέλους.

```
import java.awt. Graphics;
import java.applet.*;
puplic class g1 extends Applet{
puplic void paint(Graphics g){
for(int i=0;i<100;i+=10){
g.drawLine(i,i,i+10,i);
}
}
}
```

Οι μέθοδοι drawRect και fillRect: Με την java toolkit μπορούν να σχεδιαστούν παραλληλόγραμμα, να γεμίσουν με χρώμα, να στρογγυλοποιηθούν και να αποκτήσουν μια ψευδο-3D μορφή.

Οι μέθοδοι drawPolygon και fillPolygon: Μπορείτε να δημιουργήσετε πολύγωνα στην οθόνη, κατασκευάζοντας ένα πίνακα των συντεταγμένων x και ένα πίνακα των συντεταγμένων y.Αυτοί καθορίζουν τις γωνίες του πολυγώνου.Οι γωνίες δεν θα ενωθούν μεταξύ τους από την drawPolygon, αν δεν προσθέσετε το τελευταίο σημείο

του πίνακα.Η fillPolygon θα κλείσει το περίγραμμα του πολυγώνου, έτσι ώστε να έχει τη δυνατότητα να το γεμίσει με χρώμα.

Οι μέθοδοι drawOval και drawArc: Μπορείτε να σχεδιάσετε ελλείψεις και τμήματα ελλείψεων, χρησιμοποιώντας τη μέθοδο drawOval και drawArc, και να τα γεμίσετε με χρώμα χρησιμοποιώντας τις μεθόδους fillOval και fillArc αντίστοιχα. Οι παράμετροι που δέχονται οι drawOval και drawArc είναι ουσιαστικέ οι συντεταγμένες του μικρότερου παραλληλόγραμμου που περιβάλλει την έλλειψη. Οι παράμετροι που χρησιμοποιούνται είναι οι συντεταγμένες άνω και αριστερά, καθώς και το ύψος και το πλάτος.

Η μέθοδος drawArc δέχεται δύο ακόμα παραμέτρους οι οποίες υποδεικνύουν την έναρξη και το μήκος του τόξου, μετρημένοι σε μοίρες. Η τιμή μηδέν βρίσκεται στο άνω κεντρικό σημείο και οι τιμές αυξάνουν σύμφωνα με τη φορά του ρολογιού. Έτσι η τιμή των 90 μοιρών βρίσκεται στη δεξιά πλευρά του κύκλου.

Παράδειγμα

```
import java.awt. Graphics;
import java.applet.*;
puplic class g4 extends Applet{
puplic void paint(Graphics g){
for(int i=0;i<140;i+=20){
g.drawOval (i,5,15,45);
}
for(int i=0;i<140;i+=20){
g. fillOval (i,55,15,45);
        }
        for(int i=0;i<140;i+=20){
        g. drawArc (i,100,20,45, i+10, i+100);
        }
        for(int i=0;i<140;i+=20){
        g. fill Arc (i,50,20,45, i+10, i+100);
        }
      }
```

}

7) Χρώματα

Η Java προσφέρει στον προγραμματιστή τη δυνατότητα να διαλέξει χρώματα, χρησιμοποιώντας ένα βασικό μοντέλο το οποίο είναι κοινό στα περισσότερα μηχανήματα. Κάθε χρώμα κατασκευάζεται από ένα συνδυασμό κόκκινου, πράσινου και μπλε(μοντέλο red, green, blue ή RGB).Η ένταση του κάθε τμήματος καθορίζεται από ένα byte, το οποίο μπορεί να πάρει τιμές από 0 έως 255.Ο συνδυασμός (255, 255, 255) δίνει το λευκό χρώμα, ο συνδυασμός (255,0, 0) καθαρό κόκκινο, ο συνδυασμός (0, 255, 0) καθαρό πράσινο και ο συνδυασμός(0, 0, 255) καθαρό μπλε. Ο συνδυασμός (0, 0, 0) δίνει το μαύρο χρώμα.

Πίνακας χρωματικών τιμών

Object Name	*Red*	*Green*	*Blue*	*Red*	*Green*	*Blue*	*Object Name*
Color.cyan	0	255	255	0	0	255	Color.blue
Color.black	0	0	0	0	255	0	Color.green
Color.gray	128	128	128	192	192	192	Color.lightgray
Color.dark Gray	64	64	64	0	0	0	Color.white
Color.red	255	0	0	255	255	0	Color.yellow
Color.magenta	255	0	255	0	255	255	Color.cyan
Color.pink	255	175	175	255	200	0	

							Color.orange

8) Τεχνολογία στην Java

Η Java χρησιμοποιεί τον όρο applet (εφαρμογίδιο) για να αναφέρεται στα προγράμματα ενεργών εγγράφων για να κάνει διάκριση μεταξύ των ενεργών εγγράφων και των συμβατικών προγραμμάτων των υπολογιστών. Η τεχνολογία Java περιλαμβάνει 3 βασικά συστατικά μέρη τα οποία έχουν σχέση με τα εφαρμογίδια:

- Γλώσσα προγραμματισμού. Η τεχνολογία Java περιλαμβάνει μια νέα γλώσσα προγραμματισμού, η οποία μπορεί να χρησιμοποιείται ως γλώσσα πηγαίου κώδικα και για συμβατικά προγράμματα υπολογιστών και για εφαρμογίδια Java.
- Περιβάλλον χρόνου εκτέλεσης. Το σύστημα Java ορίζει το περιβάλλον χρόνου εκτέλεσης το οποίο παρέχει τις υπηρεσίες που χρειάζονται για να εκτελείται ένα πρόγραμμα Java.
- Βιβλιοθήκη τάξεων.Για να είναι πιο εύκολη η συγγραφή εφαρμογιδίων, η Java παρέχει μια μεγάλη βιβλιοθήκη, η οποία παρέχει λογισμικό που καλύπτει τις περιέχει λογισμικό που καλύπτει τις περισσότερες εργασίες που πραγματοποιεί ένα εφαρμογίδιο.

Στην πράξη, η διάκριση μεταξύ των συστατικών μερών της τεχνολογίας Java μπορεί να είναι δύσκολη, επειδή είναι σχεδιασμένα να δουλεύουν μαζί. Για παράδειγμα, η γλώσσα περιέχει δυνατότητες που εξαρτώνται από την υποστήριξη του περιβάλλοντος χρόνου εκτέλεσης, και η βιβλιοθήκη περιέχει λογισμικό που παρέχει μια διασύνδεση για υπηρεσίες του περιβάλλοντος χρόνου εκτέλεσης.

9) Αντικείμενα – Κλάσεις

Ένα αντικείμενο ορίζεται ως ένα σύνολο από πεδία (*fields*) (ή μεταβλητές υπόστασης (*instance variables*) ή ιδιότητες (*properties*)) και μεθόδους (*methods*). Και οι δύο κατηγορίες ονομάζονται μέλη (*members*) του αντικειμένου. Τα πεδία περιέχουν δεδομένα σχετικά με την κατάσταση του αντικειμένου. Οι μέθοδοι περιέχουν κώδικα που επιτρέπει την πρόσβαση και την αλλαγή των ιδιοτήτων του αντικειμένου

Τα πεδία και οι μέθοδοι ενός συγκεκριμένου αντικειμένου ονομάζονται πεδία και ιδιότητες της υπόστασης (*instance*) του αντικειμένου. Αντικείμενα του ίδιου τύπου ομαδοποιούνται σε μια κλάση κλάση (*class*) (ή τάξη). Στην κλάση αυτή ορίζουμε τα πεδία και τις μεθόδους του κάθε αντικειμένου. Κάθε κλάση αποτελεί έναν νέο τύπο με βάση τον οποίο μπορούμε να ορίζουμε μεταβλητές

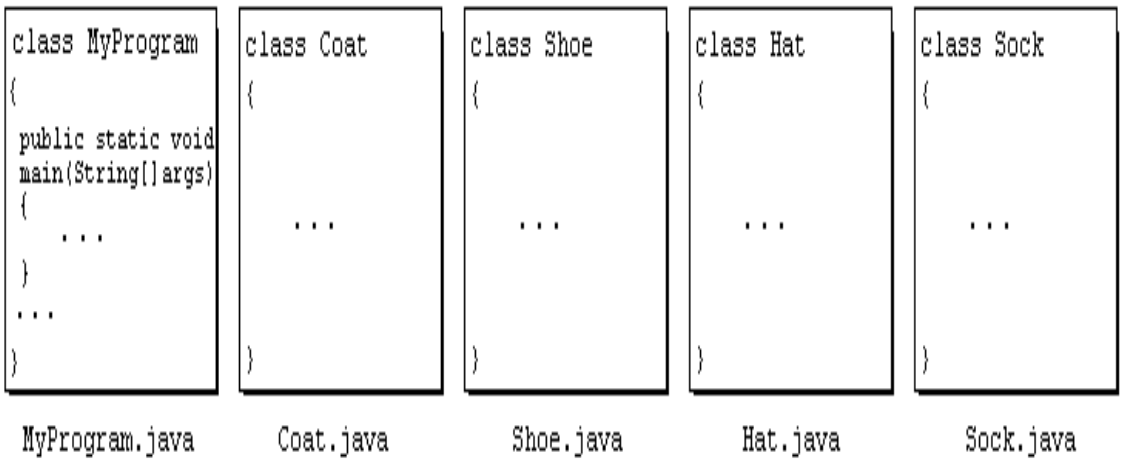
```
import java.util.*;

class TestCal {
    public static void main(String argv[]) {
        GregorianCalendar now, yesterday;
    }
}
```

Σε μια αντικειμενοστραφή γλώσσα προγραμματισμού υπάρχουν έτοιμα επαναχρησιμοποιήσιμα τμήματα κώδικα που λέγονται αντικείμενα. Ένα αντικείμενο αποτελείται από μεταβλητές και μεθόδους. Όταν χρησιμοποιούμε απλές μεταβλητές,

μπορούμε να χρησιμοποιήσουμε διάφορες εκφράσεις για να αλλάξουμε την τιμή τους.

Μερικές μέθοδοι, που είναι ενσωματωμένες ρουτίνες, χρησιμοποιούνται για την αλλαγή των τιμών των μεταβλητών, ενώ άλλες ελέγχουν τις δραστηριότητές του. Είναι ένας ασφαλής τρόπος για να πραγματοποιηθούν αυτές οι αλλαγές, καθώς εξασφαλίζεται ότι η νέα τιμή θα είναι μέσα στα αποδεκτά όρια και ότι δεν θα δημιουργεί παρενέργειες σε άλλες ιδιότητες του αντικειμένου.
Υπάρχει η δυνατότητα της ένθεσης (encapsulation). Οι μεταβλητές ενός αντικειμένου προστατεύονται από την άμεση επαφή με τον έξω κόσμο και μπορούν να αλλαχθούν μόνο μέσα από τις μεθόδους του αντικειμένου.
ΣΗΜΕΙΩΣΗ: Μερικά αντικείμενα της Java, όπως πλήκτρα ή άλλα συστατικά της οθόνης, αποτελούν σαφώς αντικείμενα. Τα αντικείμενα όμως μπορεί να μην είναι ορατά. Οι δομές δεδομένων είναι αντικείμενα.

Οι κλάσεις είναι σχέδια που ακολουθούνται κάθε φορά που θέλουμε να κατασκευάσουμε κάτι, και απλώς καθορίζουν κάποια μεταβλητά στοιχεία.
Κάθε αντικείμενο αποτελεί ένα *στιγμιότυπο* μιας κλάσης. Η διαδικασία δημιουργίας ενός αντικειμένου ονομάζεται *αρχικοποίηση στιγμιότυπου*. Η διαφορά που υπάρχει μεταξύ των κλάσεων και των αντικειμένων, συχνά είναι ασαφής για τον παρακάτω λόγο.



10) Κλάσεις αντικειμένων στη Java

Παρακάτω παρατίθεται περιγραφικά ο κώδικας για τη δημιουργία της βασικής δομής μιας κλάσης στη Java. Τα περιεχόμενα εντός των εισαγωγικών < >, περιγράφουν την κλάση και μεταβάλλονται από το χρήστη ανάλογα τα στοιχεία της κλάσης που θέλουμε να δημιουργήσουμε.

class <class_name> {

<data_type or class_name> <variables or objects>;

<returned type> <method_name> (<parameter list>) {

<method body>

}

<data_type or class_name> <variables or objects>;

<returned type> <method_name> (<parameter list>) {

<method body>

}

Συγκεκριμένα, μπορούμε παρακάτω να δούμε τη δημιουργία μια κλάσης για τη μισθοδοσία των εργαζομένων μίας επιχείρησης:

Παράδειγμα:

class Salaries {

int num_of_Employees; // πεδίο

int countEmployees() { // μέθοδος

return num_of_Employees;

}

Employees e1, e2,…,en; // πεδία

Employee getEmployee(int num) { // μέθοδος

return ....;

}

}

Η μεταβλητή num_of_Employees που εισήχθη πρώτη, είναι μία μεταβλητή τύπου int (integer). Employees είναι το όνομα μιας κλάσης και τα e1, e2… en είναι αντικείμενα της κλάσης Employees. Ακόμα παρατηρούμε ότι οι μέθοδοι μπορούν να έχουν επιστρεφόμενο τύπο (return) έναν απλό τύπο δεδομένων ή και μία κλάση. Η δομή της κλάσης δεν είναι προκαθορισμένη. Δηλαδή, τα πεδία και οι μέθοδοι μπορούν να δηλωθούν με όποια σειρά θέλει ο χρήστης, ακόμα και να χρησιμοποιηθούν πριν ακόμα δηλωθούν. Παρ' όλ' αυτά τα πεδία τοποθετούνται στην αρχή της κλάσης για να είναι ο κώδικας πιο ευανάγνωστος.

Ορατότητα

Όταν χρησιμοποιούμε μεταβλητές μιας κλάσης μέσα σε μια άλλη, τότε μπορούμε να χρησιμοποιήσουμε όλα τα πεδία και όλες τις μεθόδους που έχουν οριστεί με δημόσιο (*public*) προσδιοριστή.

Αντίθετα, δεν μπορούμε να χρησιμοποιήσουμε σε άλλες κλάσεις πεδία και μεθόδους που έχουν οριστεί με ιδιωτικό (*private*) ή προστατευόμενο (*protected*) ή προσδιοριστή. Αυτές μπορούν να χρησιμοποιηθούν μόνο μέσα από μεθόδους της κλάσης στην οποία ορίζονται.

Στο παρακάτω παράδειγμα στην κλάση Point τα στοιχεία public είναι τα x, y, και moveToCenter(). Αντίθετα τα μέλη (ιδιότητες) της κλάσης visible και serialNumber και η μέθοδος setpos δεν είναι ορατά και προσβάσιμα παρά μόνο από τις συναρτήσεις (μεθόδους) της κλάσης.

Αρχείο Point.java

```
import gr.aueb.dds.BIO;

class Point {
    // Public fields
    public int x, y;
    private boolean visible;
    private int serialNumber;
```

```
	// Private method
	private void setpos(int sx, int sy) {
		x = sx;
		y = sy;
	}
	// Public methd
	public void moveToCenter() {
		setpos(0, 0);
	}
}
```

Αρχείο TestPoint.java

```
import gr.aueb.dds.BIO;
class TestPoint {
	public static void main(String args[])
	{
		Point a;

		a = new Point();
		a.moveToCenter();
		// Use public field
		a.x = 10;
	}
}
```

Ορισμός πεδίων υπόστασης

Τα πεδία υπόστασης ορίζονται σαν να ήταν μεταβλητές, μέσα στο σώμα της κλάσης.

```
class Point {
	/* Fields */

	/** Point x coordinate */
	private int x;
	/** Point y coordinate */
```

```
    private int y;
}
```

- Καλό είναι τα πεδία να ορίζονται με τον προσδιοριστή private έτσι ώστε να μην είναι ορατά σε άλλες κλάσεις.
- Στη Java προτρέπουμε τα ονόματα των πεδίων να αρχίζουν με πεζό γράμμα.

Ορισμός μεθόδων υπόστασης

- Οι μέθοδοι υπόστασης ορίζονται ως συναρτήσεις μέσα στο σώμα της κλάσης.

```
class Point {
    /* Fields */

    /** Point x coordinate */
    private int x;
    /** Point y coordinate */
    private int y;

    /* Methods */
    /** Return the x coordinate */
    public int getx() { return x; }
   /** Return the y coordinate */
    public int gety() { return y; }
    /** Set the point position to the specified coordinates */
    public void setpos(int sx, int sy) {
        x = sx;
        y = sy;
    }
   /** Return string representation */
    public String toString() {
        return "x=" + new Integer(x).toString() +
            " y=" + new Integer(y).toString();
    }
}
```

- Ο προσδριοριστής static που δίνουμε στη συνάρτηση main φυσικά δε δίνεται στην περίπτωση του ορισμού των παραπάνω μεθόδων μια και πρόκειται για μεθόδους υπόστασης και όχι για μεθόδους κλάσης
- Στη Java προτρέπουμε τα ονόματα των μεθόδων να αρχίζουν με πεζό γράμμα

Πρόσβαση στα μέλη της κλάσης

- Μέσα στο σώμα των μεθόδων υπόστασης της κλάσης έχουμε άμεση πρόσβαση σε όλες τις μεθόδους και τα πεδία με απλή χρήση του ονόματός τους
- Όταν αναφερόμαστε σε μια μέθοδο ή πεδίο εξυπακούεται πως αναφερόμαστε στο αντικείμενο για το οποίο κλήθηκε η δική μας μέθοδος
- Στο αντικείμενο αυτό μπορούμε να αναφερθούμε και με το όνομα this.
- Έτσι οι ο ορισμός

```
class Point {
    /** Set the point position to the specified coordinates */
    public void setpos(int sx, int sy) {
        x = sx;
        y = sy;
    }
}
```

είναι ίδιος με τον ορισμό

```
class Point {
    /** Set the point position to the specified coordinates */
    public void setpos(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

Από μεθόδους κλάσης (static) δε μπορούμε να έχουμε άμεση πρόσβαση σε μεθόδους και πεδία υπόστασης μια και οι μέθοδοι κλάσης δεν καλούνται για ένα συγκεκριμένο αντικείμενο.

Μέθοδοι κατασκευής

- Σε κάθε κλάση μπορούν να οριστούν μέθοδοι κατασκευής με όνομα ίδιο με αυτό της κλάσης.
- Η μέθοδος κατασκευής καλείται κάθε φορά που δημιουργείται ένα νέο αντικείμενο (με τη χρήση new ObjectType).
- Το όρισμα που δηλώνουμε στη μέθοδο κατασκευής επιτρέπει προσδιορισμό ιδιοτήτων του αντικειμένου που δημιουργούμε (π.χ. τον αριθμό στοιχείων σε έναν πίνακα) ή αρχικών τιμών.
- Για το ίδιο αντικείμενο μπορούμε να ορίσουμε πολλές μεθόδους κατασκευής με διαφορετικά ορίσματα.
- Η μέθοδος κατασκευής δε χρειάζεται κάποιον προσδιοριστή (static, public, void) στον ορισμό της.
- Ρόλος της μεθόδου κατασκευής είναι συνήθως να δίνει αρχικές τιμές στα πεδία της κλάσης και να αποκτά πρόσβαση σε εξωτερικούς πόρους που απαιτούνται για τη λειτουργία της.

Παράδειγμα:

```
class Point {
    /* Fields */

    /** Point x coordinate */
    private int x;
    /** Point y coordinate */
    private int y;

    /* Methods */
    /** Default constructor */
    Point() { x = y = 0; }
    /** Constructor with coordinates */
    Point(int x, int y) {
        this.x = x;
        this.y = y;
```

```
    }
}
```

Μέθοδοι καταστροφής

- Σε κάθε κλάση μπορεί να οριστεί μια μέθοδος καταστροφής (*destructor*) με όνομα finalize.
- Η μέθοδος καταστροφής καλείται κάθε φορά που παύει να υπάρχει ένα αντικείμενο δηλαδή αντίστοιχα όταν τελειώνει το πρόγραμμα, όταν η ροή βγαίνει από το τοπικό τμήμα, ή όταν το σύστημα αποκομιδής αχρήστων της Java καταστρέφει το συγκεκριμένο αντικείμενο.
- Η μέθοδος καταστροφής δε δέχεται κάποιο όρισμα.
- Η μέθοδος καταστροφής δε χρειάζεται κάποιον προσδιοριστή (static, public, void).
- Ρόλος της μεθόδου καταστροφής είναι να αποδεσμεύει εξωτερικούς πόρους που δέσμευσε η μέθοδος κατασκευής.

Παράδειγμα:

```
class Point {
    /* Fields */

    /** Point x coordinate */
    private int x;
    /** Point y coordinate */
    private int y;

    /* Methods */
    /** Default constructor */
    Point() { x = y = 0; }
    /** Constructor with coordinates */
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
```

```
    /** Point destructor */
    public void finalize() {
        System.out.println("Another point (" + this + ") bites the dust");
    }

    /** Test method */
    static public void main(String args[]) {
        Point a = new Point(1, 2);
        Point b = new Point(5, 8);
        a = b;
        // Force the garbage collector to run
        System.gc();
    }
}
```

Έλεγχος πρόσβασης

Κάθε μέθοδος ή πεδίο μπορεί να έχει ως προσδιοριστή
public
Το μέλος είναι ορατό σε όλες τις άλλες κλάσεις
private
Το μέλος είναι ορατό μόνο στις μεθόδους της δικής μας κλάσης
Μια καλοσχεδιασμένη κλάση έχει:

- Όλα τα μέλη υπόστασης ορισμένα ώς private.
- Όλες τις μεθόδους που δεν έχουν σχεδιαστεί για χρήση έξω από την κλάση ορισμένες ως private.

11) Ιδιότητες και μέθοδοι κλάσης

- Σε μια κλάση μπορούν να δηλωθούν (με τον προσδιορισμό static) πεδία τα οποία υπάρχουν μόνο μια φορά για όλη την κλάση, καθώς και αντίστοιχες μέθοδοι.
- Τα μέλη αυτά χρησιμοποιούνται για την επεξεργασία στοιχείων που αφορούν ολόκληρη την κλάση και όχι τα αντικείμενά της.

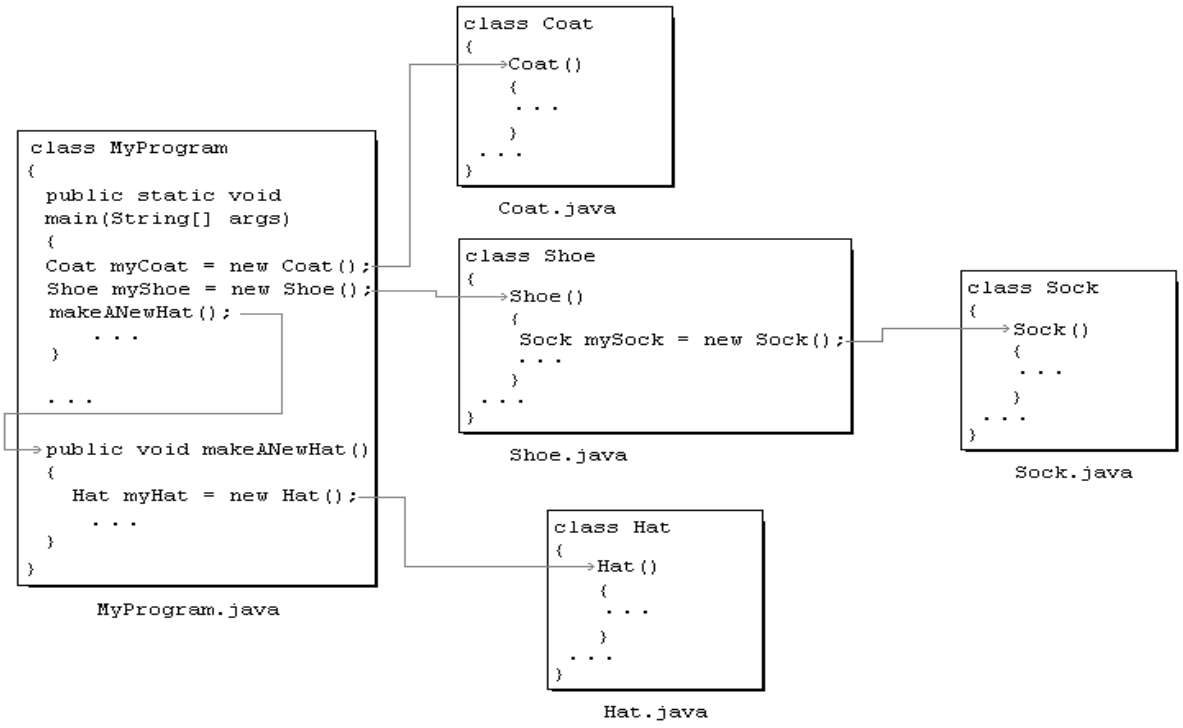
- Οι συναρτήσεις που έχουν οριστεί static δεν έχουν πρόσβαση σε μη static μεταβλητές ούτε στη μεταβλητή this.
- Το παρακάτω παράδειγμα ορίζει έναν μετρητή numPoints που μετρά πόσα σημεία είναι ενεργά καθώς και την αντίστοιχη συνάρτηση πρόσβασης getNumPoints:

```
class Point {
    private int x, y;
    /** Count number of points used */
    private static int numPoints = 0;
    /** Point constructor */
    Point(int ix, int iy) {
        x = ix;
        y = iy;
        numPoints++;          // Adjust points counter
    }
    /** Point constructor */
    Point() {
        x = y = 0;
        numPoints++;          // Adjust points counter
    }
    // Point destructor
    public void finalize() {
        numPoints--;          // Adjust points counter
    }

    // Return number of points currently used
    public static int getNumPoints() {
        return numPoints;
    }
    static public void main(String args[]) {
        Point a = new Point(1, 2);
        Point b = new Point(5, 8);
        Point c = new Point(7, 2);
```

```
            a = b;
            System.out.println(getNumPoints() + " points are alive now");
            System.out.println("Garbage collecting");
            // Force the garbage collector to run
            System.gc();
            System.out.println(Point.getNumPoints() + " points are alive now");
        }
}
```

Η μέθοδος main

Κάθε μία από τις παραπάνω κλάσεις έχει διαφορετικές ιδιότητες, αφορούν δηλαδή στο πως συμπεριφέρεται κάποιο αντικείμενο που θα δημιουργηθεί από αυτές. Στο σύνολό τους αποτελούν μία εφαρμογή. Μία από αυτές, η κλάση MyProgram, μεταξύ άλλων περιέχει και τη μέθοδο main. Είναι λοιπόν αυτή που θα κληθεί όταν εκτελεστεί η εφαρμογή. Κατά συνέπεια η κλάση MyProgram είναι υπεύθυνη για τη δημιουργία αντικειμένων των υπολοίπων κλάσεων. Αυτό μπορεί να γίνει έμμεσα ή άμεσα. Έτσι, μία πιθανή εκδοχή είναι η παρακάτω:



Η κλάση MyProgram δημιουργεί μέσα από τη μέθοδο main ένα αντικείμενο myCoat της κλάσης Coat και ένα αντικείμενο myShoe της κλάσης Shoe, η μέθοδος constructor της οποίας δημιουργεί εκτός των άλλων και ένα αντικείμενο mySock της κλάσης Sock. Στη συνέχεια καλείται η μέθοδος makeANewHat η οποία δημιουργεί ένα αντικείμενο myHat της κλάσης Hat. Παρατηρούμε δηλαδή πως η κλάση MyProgram είναι τελικά υπεύθυνη για την δημιουργία αντικείμενων των υπολοίπων κλάσεων, είτε αυτό γίνεται με άμεσο (περιπτώσεις myCoat, myShoe, myHat) είτε με έμμεσο τρόπο (περίπτωση mySock). Η μεταγλώττιση λοιπόν της παραπάνω εφαρμογής αφορά την κλάση που περιέχει τη μέθοδο main. Θα δηλώναμε δηλαδή:

>javac MyProgram.java

Αυτό αρκεί για να μεταγλωττιστούν και οι υπόλοιπες εφόσον γίνεται έμμεση ή άμεση αναφορά σε αυτές. Σημειώνεται τέλος πως όλα τα αρχεία πηγαίου κώδικα που αποτελούν μία κοινή εφαρμογή, πρέπει απαραιτήτως να βρίσκονται στον ίδιο φάκελο.

Το σχεδιαστικό πρότυπο singleton

Το σχεδιαστικό πρότυπο (*design pattern*) *singleton* επιτρέπει τον ελεγχόμενο ορισμό ενός μοναδικού αντικειμένου μιας κλάσης.

Παράδειγμα:

```
public class Company {
    /** Single company instance */
    private static final Company instance = new Company();

    /** Return the single company instance */
    public static Company getInstance() {
        return instance;
    }

    /** Supress public default constructor to assure non-instantiation */
private Company() {
```

```
            // Never invoked from outside the class
        }
    }
```

Πρόσβαση στο μοναδικό αντικείμενο Company έχουμε μόνο με τη μέθοδο getInstance()

Η κλάση Stack

Η κλάση

Stack <E>

υλοποιεί μια δομή δεδομένων με το χαρακτηριστικό τύπο πρόσβασης τελευταίο μέσα πρώτο έξω (*last in first out*) (LIFO). Η δομή αυτή ονομάζεται στοίβα (*stack*). Η κλάση Stack υλοποιεί τη διεπαφή Collection και ορίζει μεταξύ άλλων τις παρακάτω μεθόδους:

boolean empty()

Αληθές αν η στοίβα είναι άδεια

E push(E item)

Προσθέτει το αντικείμενο item στην κορυφή της στοίβας

E pop()

Αφαιρεί και επιστρέφει το αντικείμενο από την κορυφή της στοίβας

E peek()

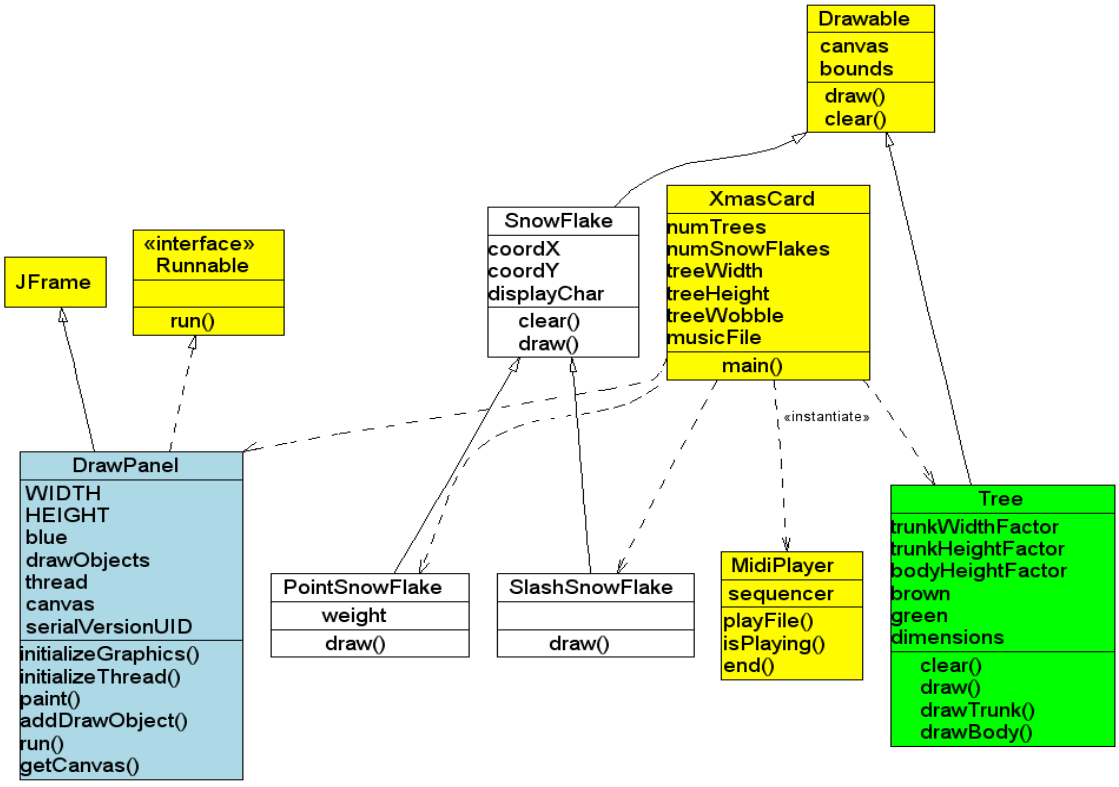
Επιστρέφει το αντικείμενο από την κορυφή της στοίβας

Παράδειγμα: οι πύργοι του Ανόι

```
import java.util.*;
public class Hanoi {
    static Stack A, B, C;
    /** Display the contents of a collection with a given name */
    static void showCollection(String name, Collection c) {
        Iterator i;

        System.out.print(name + ": ");
        for (i = c.iterator(); i.hasNext(); )
            System.out.print(i.next() + " ");
        System.out.println();
```

```
    }

    /** Display the hanoi towers */
    static void showConfiguration() {
        showCollection("A", A);
        showCollection("B", B);
        showCollection("C", C);
        System.out.println();
    }
    /** Move n blocks from to using tmp */
    static void move(int n, Stack from, Stack to, Stack tmp) {
        if (n == 1) {
            to.push(from.pop());
            showConfiguration();
        } else {
            move(n - 1, from, tmp, to);
            to.push(from.pop());
            showConfiguration();
            move(n - 1, tmp, to, from);
        }
    }
    public static void main(String args[]) {
        final int N = 4;
        A = new Stack();
        B = new Stack();
        C = new Stack();

        for (int i = N; i > 0; i--)
            A.push(new Integer(i));
        showConfiguration();
        move(N, A, C, B);
    }
}
```

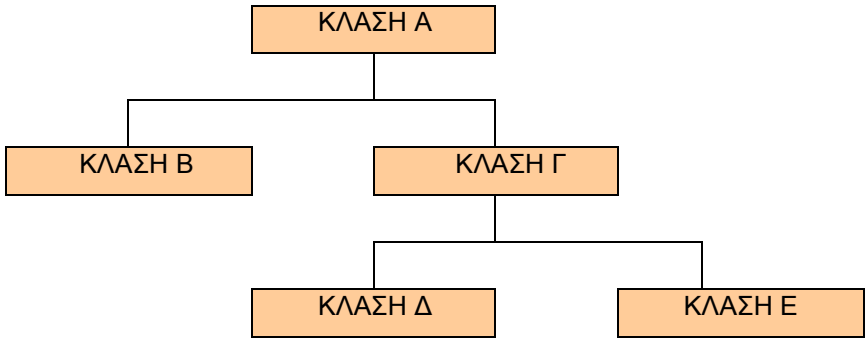
Διάγραμμα των κλάσεων



12) Η κληρονομικότητα των κλάσεων

Μια κλάση B, λέγεται υποκλάση (subclass), ή παράγωγη κλάση ή παιδί της κλάσης A, όταν κληρονομεί από αυτήν τις μεθόδους και τα δεδομένα της. Ενώ υπερκλάση (superclass), ή παράγουσα κλάση ή πατέρας της B, λέγεται μία κλάση A, οι οποία παραδίδει τις ιδιότητες της σε μία κλάση B. Η διαδικασία αυτή λέγεται subclassing. Μια υπερκλάση μπορεί να απευθύνεται

σε περισσότερες από μία κλάσεις. Δηλαδή η κληρονομικότητα δημιουργεί μια δενδρική ιεραρχική δομή μεταξύ των κλάσεων.



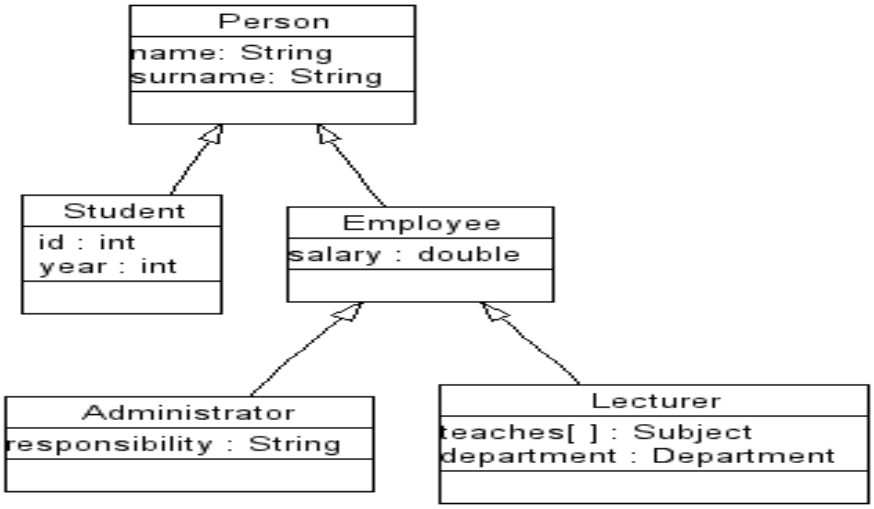
Ας πάρουμε, για παράδειγμα, μια εταιρεία κατασκευής αυτοκινήτων. Τα νέα μοντέλα θα τα παράγουν σε διάφορες εκδόσεις και θα επιλέξουν να βάλουν 2 ή 4 πόρτες στο καθένα. Η διαδικασία ξεκινάει με ένα σχέδιο που καλύπτει τα κοινά στοιχεία. Στην συνέχεια, συμπληρώνουν σε αυτό τα ειδικά στοιχεία που χαρακτηρίζουν ποίας έκδοσης είναι και δημιουργούνται οι εκδόσεις με τις 2 ή τις 4 πόρτες. Αφού συμπληρωθεί το βασικό σχέδιο, όλα τα υπόλοιπα κληρονομούν τα χαρακτηριστικά του, αλλά φυσικά προσθέτουν και πολλά επιπλέον στοιχεία.

Συχνά μια σειρά κλάσεων αντικειμένων μπορεί να μοντελοποιηθεί με τη μορφή μιας ιεραρχίας.

Για παράδειγμα:

Όλα τα μέλη της Πανεπιστημιακής Κοινότητας είναι φυσικά πρόσωπα και έχουν ως ιδιότητες το όνομα και το επώνυμό τους. Οι φοιτητές έχουν ακόμα ως ιδιότητα τον αριθμό μητρώου τους (id) και το έτος που φοιτούν. Όσοι έχουν σχέση εργασίας με το Πανεπιστήμιο έχουν ως ιδιότητα το μισθό τους. Το διοικητικό προσωπικό έχει ως πρόσθετη ιδιότητα τον τομέα ευθύνης του. Οι διδάσκοντες έχουν ως πρόσθετη ιδιότητα το τμήμα τους και τα μαθήματα που διδάσκουν.

Οι σχέσεις αυτών των κλάσεων μπορούν να παρασταθούν στο παρακάτω διάγραμμα.



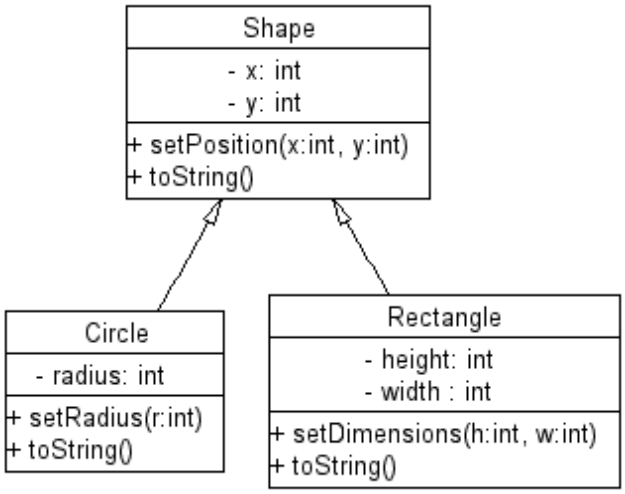
Κάθε κλάση κληρονομεί τις ιδιότητες της μητρικής της κλάσης.

Παράδειγμα με κληρονομικότητα:

```
        public String toString() {
                return super.toString() + ": Circle(" + radius + ")";
        }
}

class Rectangle extends Shape {
        private int height, width;
        public void setDimensions(int h, int w) {
                height = h;
                width = w;
        }
        public String toString() {
                return super.toString() + ": Rectangle(" + height + " x " + width +
")";
        }
}

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

```
    {
        Circle c = new Circle();
        Rectangle r = new Rectangle();
        r.setPosition(1, 2);
        r.setDimensions(50, 50);
        c.setPosition(3, 4);
        c.setRadius(10);
        System.out.println(r);
        System.out.println(c);
    }
}
```

Η παραπάνω διάταξη μπορεί να παρασταθεί σχηματικά ως εξής:



Ο προσδιοριστής protected

Μπορούμε να ορίσουμε μια κατηγορία μεθόδων και ιδιοτήτων με τον προσδιορισμό protected. Αυτές είναι προσβάσιμες από τις υποκλάσεις της κλάσης μας, αλλά όχι από άλλες συναρτήσεις έξω από την ιεραρχία της κλάσης. Παράδειγμα:

```
class Shape {
    private int x, y;           // Position
    protected int getX() { return x; }
    protected int getY() { return y; }
    public void setPosition(int px, int py) {
        x = px;
```

```
        y = py;
    }
    public String toString() {
        return "Shape(" + x + ", " + y + ")";
    }
}
```

Δυναμική διεκπεραίωση

Μια αναφορά (αντικείμενο) σε μια υποκλάση μπορεί αυτόματα να μετατραπεί σε αναφορά (αντικείμενο) της βασικής της κλάσης. Αυτό επιτρέπεται διότι κάθε αντικείμενο μιας υποκλάσης είναι (*is a*) και αντικείμενο της βασικής κλάσης. Το αντικείμενο συνεχίζει να διατηρεί τις ιδιότητες της υποκλάσης μετά τη μετατροπή και μπορεί να μετατραπεί πίσω στην ίδια υποκλάση με τη χρήση ρητού τελεστή μετατροπής (cast). Παράδειγμα:

```
static public void main(String args[])
{
    Rectangle r = new Rectangle();
    Shape s;

    r.setPosition(1, 2);
    r.setDimensions(50, 50);
    s = r;
    s.setPosition(10, 20);
    System.out.println(r);
    System.out.println(s);
    r = (Rectangle)s;
}
```

Οι υποκλάσεις μιας κλάσης μπορούν να αντικαταστήσουν μια μέθοδό της με μια που θα ορίσουν αυτές.

Όταν κληθεί η μέθοδος που έχει αντικατασταθεί από μια υποκλάση μέσω ενός αντικειμένου της βασικής κλάσης το οποίο έχει προέλθει από

αντικείμενο κάποιας υποκλάσης τότε θα κληθεί η αντίστοιχη μέθοδος της υποκλάσης από την οποία έχει προέλθει το αντικείμενο.

Πλεονεκτήματα της δυναμικής διεκπεραίωσης

Η δυνατότητα της δυναμικής διεκπεραίωσης (*dynamic dispatch*) επιτρέπει: το δυναμικό καθορισμό της συμπεριφοράς ενός αντικειμένου ανάλογα με την κλάση του *κατά την εκτέλεση του προγράμματος*, την αλλαγή της συμπεριφοράς μιας παλιάς κλάσης από μια νεώτερη (υποκλάση της) και την ενοποιημένη διαχείριση διαφορετικών αντικειμένων μέσω της βασικής τους κλάσης. Η δυνατότητα αυτή προάγει τη Java από γλώσσα που υποστηρίζει τα αντικείμενα σε αντικειμενοστρεφή γλώσσα.

Αφηρημένες κλάσεις

Αν μια μέθοδος μιας κλάσης οριστεί με τον προσδιοριστή *abstract* και χωρίς σώμα τότε η συνάρτηση αυτή είναι ιδεατή (*abstract*) δηλαδή δεν έχει υλοποίηση στη συγκεκριμένη κλάση. Παράδειγμα:

```
abstract class document {
    public abstract String identify();
}
```

Αν μια κλάση περιέχει (ή έχει κληρονομήσει) τουλάχιστον μια ιδεατή μέθοδο τότε και αυτή πρέπει να οριστεί με τον προσδιοριστή abstract και ονομάζεται ιδεατή (*abstract*).

Οι ιδεατές κλάσεις χρησιμοποιούνται μόνο για να ορίσουν γενικές έννοιες από τις οποίες εκπορεύονται συγκεκριμένες κλάσεις.

Σε αντικείμενα μιας ιδεατής κλάσης μπορούν να ανατεθούν αντικείμενα από μη ιδεατές υποκλάσεις της. Τα ίδια όμως τα αντικείμενα μιας ιδεατής κλάσης δεν μπορούν ποτέ να δημιουργηθούν αυτούσια.

Για παράδειγμα, ένας σχεδιασμός του πληροφοριακού συστήματος του Πανεπιστημίου μπορεί να ορίσει την ιδεατή κλάση person ως βασική κλάση

για τις υποκλάσεις student, employee και visitor. Αν και δε θα μπορεί να ένα νέο αντικείμενο με βάση την αφηρημένη κλάση person, αυτή μπορεί να περιέχει ορισμένα βασικά χαρακτηριστικά όπως birth_date και να επιβάλει την υλοποίηση συγκεκριμένων μεθόδων όπως home_page_URL() ορίζοντάς τις ως ιδεατές.

Παράδειγμα

Το παρακάτω παράδειγμα ορίζει τη βασική κλάση Shape και τις υποκλάσεις της Circle και Rectangle. Η μέθοδος area μπορεί να οριστεί για τις υποκλάσεις και κατά την εκτέλεση του προγράμματος να εκτελεστεί η σωστή έκδοσή της. Η συνάρτηση toString της Shape εκμεταλλεύεται τη δυνατότητα αυτή και μπορεί να κληθεί (και να δουλέψει σωστά) με όρισμα οποιαδήποτε από τις υποκλάσεις της shape.

```
abstract class Shape {
	private double x, y;		// Position
	protected double getX() { return x; }
	protected double getY() { return y; }
	public void setposition(double px, double py) {
		x = px;
		y = py;
	}
	public abstract double area();
	public String toString() {
		return "Shape(x=" + x + ", y=" + y + ", area=" + area() + ")";
	}
}

class Circle extends Shape {
	private double radius;
	public void setradius(double r) {
		radius = r;
	}
	public double area() {
		return 2 * Math.PI * radius * radius;
```

```
        }
        public String toString() {
                return super.toString() + ": Circle(" + radius + ")";
        }
}

class Rectangle extends Shape {
        private double height, width;
        public void setdimensions(double h, double w) {
                height = h;
                width = w;
        }
        public double area() {
                return height * width;
        }
        public String toString() {
                return super.toString() + ": Rectangle(" + height + " x " + width + ")";
        }
}

class Test {
        static public void main(String args[])
        {
                Circle c = new Circle();
                Rectangle r = new Rectangle();
                Shape s[] = new Shape[2];
                s[0] = r;
                r.setposition(1, 2);
                r.setdimensions(50, 50);

                s[1] = c;
                c.setposition(3, 4);
                c.setradius(10);
                for (int i = 0; i < s.length; i++)
```

```
            System.out.println(s[i]);
        }
}
```

Το παραπάνω πρόγραμμα θα τυπώσει:

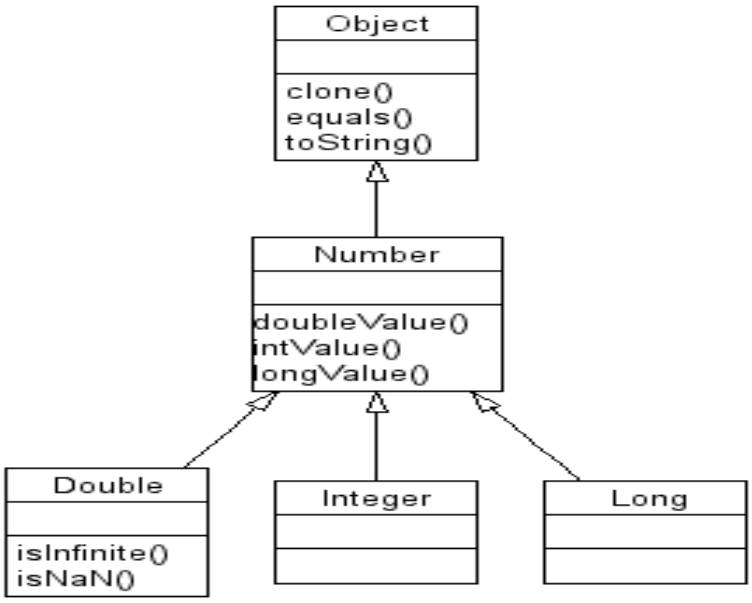
Shape(x=1.0, y=2.0, area=2500.0): Rectangle(50.0 x 50.0)

Shape(x=3.0, y=4.0, area=628.3185307179587): Circle(10.0)

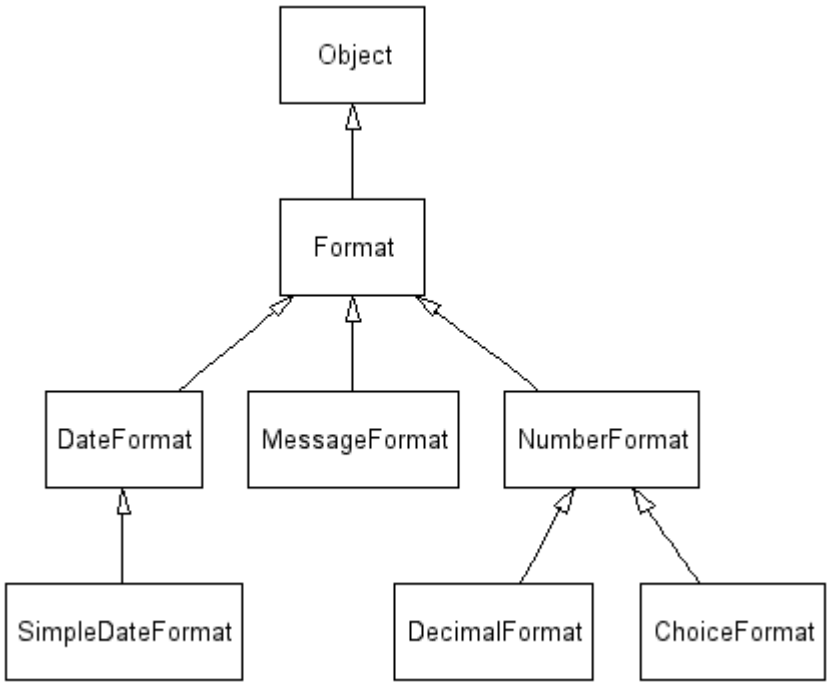
Η ιεραρχία κλάσεων στη βιβλιοθήκη της Java

Όλες οι κλάσεις της Java έχουν ως βασική κλάση την κλάση Object. Η βιβλιοθήκη της Java χρησιμοποιεί την κληρονομικότητα για να οργανώσει την ιεραρχία των κλάσεων που υποστηρίζει.

Παράδειγμα από την παράσταση αριθμών



Παράδειγμα από τις κλάσεις για το σχηματισμό μηνυμάτων

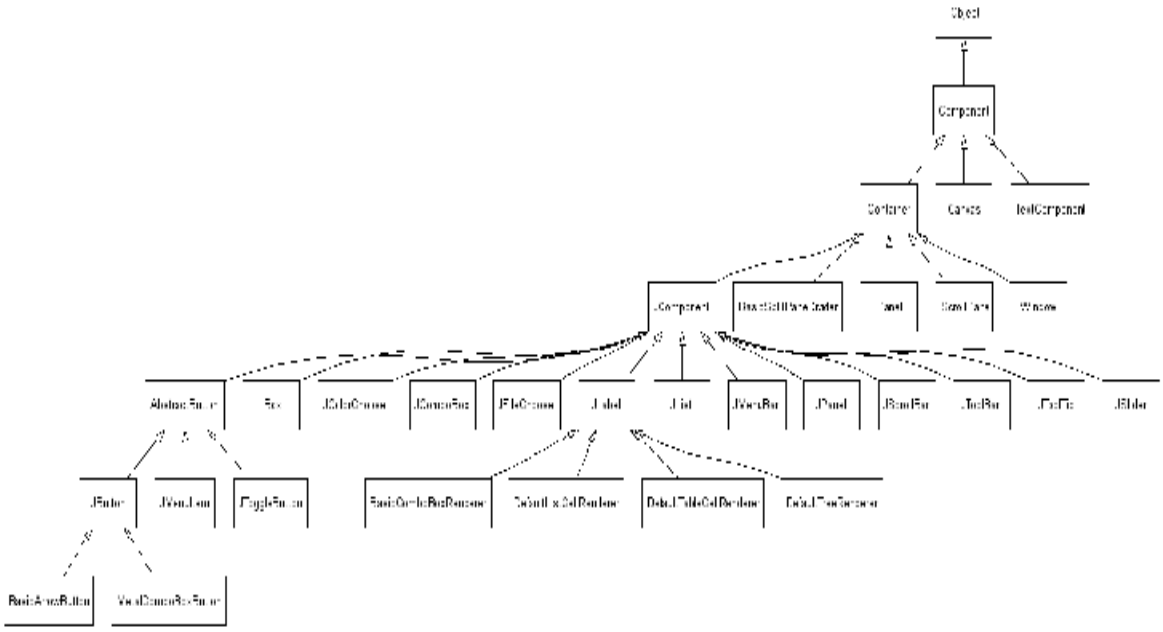


Παράδειγμα χρήσης:

```
Object[] arguments = {
    new Integer(7),
    new Date(System.currentTimeMillis()),
    "a disturbance in the Force"
};
String result = MessageFormat.format(
    "At {1,time} on {1,date}, there was {2} on planet {0,number,integer}.",
    arguments);
```

At 12:30 PM on Jul 3, 2053, there was a disturbance in the Force on planet 7.

Τμήμα της ιεραρχίας Swing



13) Σφάλματα και εξαιρέσεις στη Java

Μια εξαίρεση είναι ένα σήμα πως έχει συμβεί κάποιο μη κανονικό συμβάν (π.χ. λάθος). Η μη κανονική κατάσταση προκαλεί μια εξαίρεση. Στο πρόγραμμά μας μπορούμε να ορίσουμε κώδικα που θα συλλάβει την εξαίρεση. Με τη βοήθεια των εξαιρέσεων μπορούμε να ξεχωρίσουμε στο πρόγραμμά μας το χειρισμό των λαθών από τον υπόλοιπο κώδικα.

Παράσταση εξαιρέσεων

Στη Java οι εξαιρέσεις παριστάνονται ως υποκλάσεις της κλάσης Throwable:
Error: Λάθη που δεν επιτρέπουν την επανάκτηση της λειτουργίας
Exception : Λάθη που μπορεί να επιτρέπουν την επανάκτηση της λειτουργίας

Οι εφαρμογές μπορούν να ορίζουν νέες υποκλάσεις με τις δικές τους εξαιρέσεις. Κάθε κλάση εξαίρεσης χρειάζεται να έχει μόνο ως μέλη δύο μεθόδους κατασκευαστές: μία χωρίς όρισμα και μία με όρισμα συμβολοσειρά

Παράδειγμα:

```
import java.io.*;
public class NonGreekCharacterException extends IOException {
```

```
public NonGreekCharacterException() {}
public NonGreekCharacterException(String msg) {
super(msg);
 }
          }
```

Πως δημιουργούνται οι εξαιρέσεις;

Στη γλώσσα Java μια εξαίρεση μπορεί να δημιουργηθεί ως εξής

Πρώτον, με την εντολή throw .

```
if (ammount < 0)
throw new NegativeAmmountException();
```

Δεύτερον, με την κλήση μιας μεθόδου που προκαλεί την εξαίρεση. Για παράδειγμα η μέθοδος print μπορεί να προκαλέσει την NullPointerException.

Τρίτον με την εκτέλεση κώδικα που δημιουργεί εξαίρεση. Για παράδειγμα ο παρακάτω κώδικας θα δημιουργήσει μια εξαίρεση ArrayIndexOutOfBoundsException.

```
int a[] = new int[10];
a[15] = 42;
```

Πως χειριζόμαστε τις εξαιρέσεις;

Σε μια μέθοδο της Java μια εξαίρεση μπορούμε:

Τη χειριζόμαστε τοπικά, τη δηλώνουμε σαν δική μας μέθοδο που δημιουργεί την συγκεκριμένη εξαίρεση και την αγνοούμε αν είναι τύπου Error ή RuntimeException

Τοπικός χειρισμός των εξαιρέσεων

Ο τοπικός χειρισμός γίνεται με μπλοκ try/catch/finally:

```
try {
   // Κώδικας που μπορεί να δημιουργήσει την εξαίρεση
} catch (ExceptionClass_1 e) {
      // Κώδικας που χειρίζεται την ExceptionClass_1
} catch (ExceptionClass_2 e) {
  // Κώδικας που χειρίζεται την ExceptionClass_2
```

```
} finally {
  // Κώδικας που εκτελείται πάντα στο τέλος
}
```

Παράδειγμα από τη μέθοδο readInt της βιβλιοθήκης BIO:

```
public static int readInt() {
 int i = 0;
  try {
  i = Integer.parseInt(readString());
  } catch (NumberFormatException e) {
 System.err.println("Error reading Integer: " + e);
 System.exit(1); }
 return (i);
}
```

Πως δηλώνουμε τις εξαιρέσεις;

Αν μια δική μας μέθοδος δε χειρίζεται μια εξαίρεση που μπορείρ να δημιουργηθεί από μια μέθοδο που εμείς καλούμε ή προκαλεί ρητά μια εξαίρεση με τη χρήση της throw, τότε πρέπει να το δηλώσουμε με τη σύνταξη

```
myMethodName( /* ... */) throws Exception_1, Exception_2 { }
```

Παράδειγμα:

```
public static int parsePositiveInt(String s) throws
NumberFormatException {
    int i = parseInt(s);
    if (i < 0)
        throw new NumberFormatException();
```

Ισχυρισμοί εξαιρέσεων

Ένας ισχυρισμός (*assertion*) μας επιτρέπει να τεκμηριώσουμε στον κώδικα του προγράμματος την άποψή μας για τον τρόπο λειτουργίας του. Ο ισχυρισμός έχει τυπικά τη μορφή ενός κατηγορήματος που θεωρούμε πως στη συγκεκριμένη στιγμή θα είναι αληθές. Παράδειγμα:

```
b = a / 2;
assert (a >= 0 && b <= a) || (a < 0 && b > a);
```

Ο ισχυρισμός επιτρέπει σε κάποιον που διαβάζει τον κώδικά μας να καταλάβει καλύτερα πως λειτουργεί ο κώδικας. Επιπλέον ο μεταγλωττιστής μπορεί να δημιουργήσει κώδικα που ελέγχει τους ισχυρισμούς κατά την εκτέλεση του προγράμματος.

Όσο ο κώδικας βρίσκεται στο στάδιο των δοκιμών ενεργοποιούμε τον έλεγχο τον ισχυρισμών για να εντοπίσουμε λάθη.

Όταν ο κώδικας εκτελείται σε περιβάλλον παραγωγής απενεργοποιούμε τον έλεγχο των ισχυρισμών για να μην έχουμε αρνητικές επιπτώσεις στην ταχύτητα εκτέλεσης του προγράμματος. Με τη χρήση ισχυρισμών μπορούμε να ορίσουμε

Προϋποθέσεις (*preconditions*)

Συνθήκες που πρέπει να ισχύουν πριν την εκτέλεση κάποιου τμήματος κώδικα (π.χ. μεθόδου).

Μετασυνθήκες (*postcondition*)
Συνθήκες που ξέρουμε πως θα ισχύουν μετά την εκτέλεση κάποιου τμήματος κώδικα.

Στη Java ο ισχυρισμός δηλώνεται με τη δεσμευμένη λέξη assert την οποία ακολουθεί με λογική τιμή που κανονικά πρέπει να είναι αληθής. Μετά τη λογική τιμή μπορούμε να προσθέσουμε και μια συμβολοσειρά που θα τυπωθεί στην οθόνη αν ο ισχυρισμός αποτύχει (η συνθήκη βρεθεί ψευδής) Αν η συνθήκη κατά την εκτέλεση του προγράμματος είνα ψευδής, τότε δημιουργείται μια εξαίρεση τύπου AssertionError Όταν χρησιμοποιούμε ισχυρισμούς τους ενεργοποιούμε κατά την εκτέλεση να το εκτελούμε με την εντολή:

java -ea ClassName

Παράδειγμα:

/*

```
 * Run With java -ea FindMax
 */

class FindMin {

	/** Return the minimum number in non-empty array v */
	public static int findMin(int v[]) {
		int min = Integer.MAX_VALUE;

		// Precondition: v[] is not empty
		assert v.length > 0 : "v[] is empty";
		// Precondition: min <= v[i] for every i
		for (int i = 0; i < v.length; i++)
			assert min <= v[i] : "Found value < MAX_VALUE";

		// Locate the real minimum value
		for (int i = 0; i > v.length; i++)
			if (v[i] < min)
				min = v[i];

		// Postcondition: min <= v[i] for every i
		for (int i = 0; i < v.length; i++)
			assert min <= v[i] : "Found value > MAX_VALUE";
		return min;
	}

	// Test harness
	public static void main(String argv[]) {
		int t[] = new int[5];

		t[0] = 4;
		t[1] = -4;
		t[2] = 145;
		t[3] = 0;
```

```
        t[4] = Integer.MAX_VALUE;
        System.out.println("Min value is " + findMin(t));
    }
}
```

14) ΟΡΙΣΜΑΤΑ ΚΑΙ ΒΡΟΓΧΟΙ

Η Εντολή IF

Ακόμα και τα πιο ασήμαντα προγράμματα πρέπει να παίρνουν αποφάσεις. Πρέπει να ελέγχουν κάποιες συνθήκες και να λειτουργούν διαφορετικά, βασιζόμενα σ' αυτές τις συνθήκες. Αυτό είναι συνηθισμένο στην πραγματική ζωή. Για παράδειγμα, βγάζετε το χέρι σας έξω από το παράθυρο για να διαπιστώσετε αν βρέχει. Αν βρέχει παίρνετε μαζί σας ομπρέλα, διαφορετικά όχι.

Όλες οι γλώσσες προγραμματισμού έχουν μορφές δηλώσεων if που επιτρέπουν να εξετάζουμε συνθήκες. Στον προηγούμενο κώδικα θα έπρεπε να είχαμε ελέγξει αν υπήρχαν command line arguments προτού προσπαθήσουμε να τα χρησιμοποιήσουμε.
Όλοι οι πίνακες έχουν μήκη και γι' αυτό χρησιμοποιούμε τη μεταβλητή arrayname.length. Ελέγχουμε το μήκος του args παρακάτω:

```
// This is the Hello program in Java
class Hello {

   public static void main (String args[]) {

        /* Now let's say hello */
        System.out.print("Hello ");
        if (args.length > 0) {
          System.out.println(args[0]);
        }
     }
   }
```

Μεταγλωτίστε και τρέξτε το πρόγραμμα, δίνοντας διαφορετικές εισόδους κάθε φορά. Θα πρέπει να παρατηρήσετε ότι δεν είναι πια ένα ArrayIndexOutOfBoundsException αν δεν δώσετε command line arguments.

Αυτό που κάναμε ήταν να βάλουμε τη δήλωση System.out.println(args[0]) σε έναν έλεγχο υπόθεσης if (args.length>0){ }. Ο κώδικας μεταξύ των αγκίστρων System.out.println(args[0]) τώρα πια εκτελείται αν και μόνο αν το μήκος των args είναι μεγαλύτερο από το 0. Στην Java χρησιμοποιούμε το >,που σημαίνει μεγαλύτερο από, το < που σημαίνει μικρότερο από, το <= και το >=.

Θα περιμένατε ότι ο έλεγχος της ισότητας δύο αριθμών πραγματοποιείται με το σύμβολο =. Εμείς ήδη χρησιμοποιήσαμε το σύμβολο = για να θέσουμε τιμές σε μία μεταβλητή. Γι' αυτό χρειαζόμαστε ένα καινούριο σύμβολο για να ελέγχουμε την ισότητα. Έτσι η Java δανείζεται από τη C τον συμβολισμό ==.

Δεν είναι ασυνήθιστο ακόμα και για τους πιο έμπειρους προγραμματιστές να γράφουν == όταν εννοούν = και το αντίστροφο. Αυτό είναι ένα πολύ συνηθισμένο λάθος στα προγράμματα της C. Ευτυχώς στη Java δεν επιτρέπεται να χρησιμοποιούμε το == και το = στα ίδια σημεία. Έτσι ο compiler μπορεί να αντιληφθεί το λάθος κι εσείς να το διορθώσετε, προτού τρέξετε το πρόγραμμα.

Όλες οι δηλώσεις συνθήκης στη Java ζητούν τιμές boolean κι αυτές επιστρέφουν οι τελεστές ==, >, <,>=,<=. Boolean είναι μια τιμή που είναι true ή false. Αν θέλετε να θέσετε μία μεταβλητή boolean σε ένα πρόγραμμα Java, πρέπει να χρησιμοποιήσετε τις σταθερές true και false. Το false δεν είναι 0 και το true δεν είναι όχι 0, όπως στη C.

Οι έμπειροι προγραμματιστές πιθανόν να επισημάνουν ότι υπάρχει μια εναλλακτική μέθοδος να χειριστούμε το ArrayIndexOutOfBoundsException με τις δηλώσεις try και catch. Θα επιστρέψουμε σ' αυτό σύντομα.

Η εντολή ELSE

Ίσως να παρατηρήσατε ένα μικρό σφάλμα (cosmetic bug) στο προηγούμενο πρόγραμμα. Ένα cosmetic bug δεν σπάει το πρόγραμμα ή το σύστημα, ούτε παράγει λανθασμένα αποτελέσματα αλλά απλά ενοχλεί.

Το cosmetic bug εδώ ήταν ότι αν δεν περιλαμβάναμε κανένα command line argument, το πρόγραμμα δεν θα έσπαζε, αλλά θα τύπωνε το «Hello» και δεν θα άλλαζε γραμμή. Το πρόβλημα ήταν ότι χρησιμοποιήσαμε System.out.print και όχι

System.out.println. Δεν υπήρχε χαρακτήρας τέλους γραμμής. Ήταν σαν να πληκτρολογήσαμε αυτό που θέλαμε χωρίς να πατήσουμε το enter.
Αυτό θα μπορούσε να διορθωθεί αν τοποθετούσαμε το System.out.println(«»); στο τέλος της μεθόδου main αλλά έτσι θα είχαμε πολλά end-of-lines αν ο χρήστης πληκτρολογούσε ένα όνομα. Θα μπορούσαμε να προσθέσουμε μία δήλωση if. Έτσι θα είχαμε:

```
// This is the Hello program in Java
class Hello {

    public static void main (String args[]) {

      /* Now let's say hello */
      System.out.print("Hello ");
      if (args.length > 0) {
        System.out.println(args[0]);
      }
      if (args.length <= 0) {
        System.out.println("whoever you are");
      }
  }
}
```

Αυτό διορθώνει το σφάλμα, αλλά είναι δύσκολο να διαβαστεί και να εκτελεστεί ο κώδικας. Είναι εύκολο να χάσουμε μια πιθανή περίπτωση. Για παράδειγμα, μπορεί να ελέγξουμε αν το args.length είναι μικρότερο από το 0 και να αφήσουμε την πιο ενδιαφέρουσα περίπτωση δηλαδή όταν το args.length είναι ίσο με το 0. Αυτό που χρειαζόμαστε είναι μια δήλωση else. Ακολουθεί η σωστή λύση:

```
// This is the Hello program in Java
class Hello {
    public static void main (String args[]) {
        /* Now let's say hello */
      System.out.print("Hello ");
      if (args.length > 0) {
        System.out.println(args[0]);
      }
```

```
    else {
      System.out.println("whoever you are");
    }
  }
}
```

Τώρα που το Hello τουλάχιστον αποφεύγει το ArrayIndexOutOfBoundsException, δεν έχουμε τελειώσει ακόμα. Το java Hello και το Java Hello Rusty δουλεύουν, αλλά αν πληκτρολογήσουμε java Hello Elliotte Rusty Harold, η Java τυπώνει μόνο Hello Elliotte. Ας το διορθώσουμε.

Δεν περιοριζόμαστε σε δύο περιπτώσεις. Μπορούμε να συνδυάσουμε ένα else και ένα if δημιουργώντας ένα else if και να το χρησιμοποιήσουμε για να εξετάσουμε μια πληθώρα από αμοιβαία αποκλειόμενες πιθανότητες. Για παράδειγμα, ακολουθεί μία έκδοση του προγράμματος Hello που χειρίζεται τέσσερα ονόματα :

```
// This is the Hello program in Java
class Hello {
   public static void main (String args[]) {
        /* Now let's say hello */
     System.out.print("Hello ");
     if (args.length == 0) {
       System.out.print("whoever you are");
     }
     else if (args.length == 1) {
       System.out.println(args[0]);
     }
     else if (args.length == 2) {
       System.out.print(args[0]);
       System.out.print(" ");
       System.out.print(args[1]);
     }
     else if (args.length == 3) {
       System.out.print(args[0]);
       System.out.print(" ");
       System.out.print(args[1]);
       System.out.print(" ");
```

```
      System.out.print(args[2]);
    }
    else if (args.length == 4) {
      System.out.print(args[0]);
      System.out.print(" ");
      System.out.print(args[1]);
      System.out.print(" ");
      System.out.print(args[2]);
      System.out.print(" ");
      System.out.print(args[3]);
    }
    else {
      System.out.print(args[0]);
      System.out.print(" ");
      System.out.print(args[1]);
      System.out.print(" ");
      System.out.print(args[2]);
      System.out.print(" ");
      System.out.print(args[3]);
      System.out.print(" and all the rest!");
     }
    System.out.println();
  }
}
```

Ένας μη έμπειρος προγραμματιστής της Java θα έγραφε κώδικα σαν τον παραπάνω. Ένας από τους λόγους που κάνουν αυτή τη λύση δύσχρηστη, είναι ότι χρησιμοποιούμε για κάθε μεταβλητή διαφορετική δήλωση που θα την τυπώνει. Η Java παρόλα’ αυτά τυπώνει πολλαπλά πράγματα με μία δήλωση. Αντί να περιλαμβάνει μόνο ένα όνομα στο print argument, μπορεί να περιλαμβάνει πολλά που να χωρίζονται μεταξύ τους με +. Αυτά μπορούν να περιλαμβάνουν μεταβλητές όπως args[0] και σταθερά αλφαριθμητικά όπως «and all the rest!». Για παράδειγμα, το τελευταίο block θα μπορούσε να γραφεί :

```
else {
```

```
  System.out.print(args[0] + " " + args[1] + " " + args[2] + " " + args[3] + " and all the rest!");
}
```

Αυτή η σύνταξη είναι πιο εύκολο να διαβαστεί και να γραφτεί αλλά παραμένει δύσχρηστη στην περίπτωση που τα command line arguments αυξηθούν. Στην επόμενη ενότητα θα δούμε πώς να χειριζόμαστε πάνω από δύο δισεκατομμύρια command line arguments με απλό τρόπο.

white-do

Η Java προσφέρει δυο βασικές εντολές κατασκευής βρόχων, γνωστές ως white-do
Η μόνη τους διαφορά είναι ότι ο βρόχος white έχει τον έλεγχο στη κορυφή του μπλοκ κώδικα και ο βρόχος do στο κάτω μέρος του κώδικα. Σε μερικές περιπτώσεις χρησιμοποιείται ο white, όταν χρειαστεί να ελέγξετε την έκφραση,
πριν εκτελέσετε κώδικα μέσα στο σώμα του βρόχου. Σε άλλες περιπτώσεις θα θελήσετε να εκτελέσετε αυτόματα το βρόχο μια φορά και αυτό ακριβώς προσφέρει ο βρόχος do.

Η μορφή του βρόχου white είναι κάπως έτσι:

```
x=100
white(x>0){
System.out.printlin(x+bottles of beer on the wall.'')
System.out.printlin(x+bottles of beer'');
System.out.printlin(Take one dawn,pass it around.''
System.out.printlin('--x+ bottles of beer.-');
}
```

Ο βρόχος αυτό θα ξεκινήσει με τον έλεγχο (x>0)και, αν αυτός έχει τιμή true, θα εκτελέσει το σώμα. Έπειτα θα επιστρέψει στην κορυφή και θα εκτελέσει ξανά τον έλεγχο. Ο βρόχος do εκτελεί πάντοτε το σώμα του βρόχου, πριν από τον έλεγχο. Λειτουργικά , αυτό δημιουργεί διαφορά μόνο στο πρώτο πέρασμα του βρόχου. Το πρώτο αυτό πέρασμα θα εκτελείται πάντοτε .Η πρόταση ελέγχου , η οποία ελέγχει τη συνέχεια της εκτέλεσης του βρόχου, θα λειτουργήσει έπειτα με τον ίδιο τρόπο.

Παράδειγμα

```
do{
System.out.println(‘’T  minus + time+seconds to liftoff.’’) ;
Time--;
}while(time>0);
```

break

Είναι σχεδιασμένη για να προσφέρει απλές πόρτες διαφυγής στους προγραμματιστές βρόχων. Δημιουργεί άμεση έξοδο από το βρόχο, χωρίς να εκτελεστεί το υπόλοιπο σώμα κώδικα ή να γίνει ο τελικός έλεγχος.

Παράδειγμα

```
i=0;
while(I<100){
if(ItemArray[i].Foundlt)break;
i++;
}
if(ItemArray[i].Foundlt)
System.out.println(It was found in slot:’+i);
else
System.out.println(‘’Not found.’’);
```

15) Πολυμορφισμός

Πολυμορφισμός είναι η δυνατότητα που υπάρχει να δημιουργείται ένα νέο αντικείμενο με διαφορετικές μορφές. Θα μπορούσαμε, για παράδειγμα, να γράψουμε:

```
νέο τετράγωνο= Τετράγωνο()

νέο τετράγωνο= Τετράγωνο(50,100)

νέο τετράγωνο= Τετράγωνο(50,100,25,25)
```

Η πρώτη εντολή δημιουργεί στην επάνω αριστερή γωνία της οθόνης ένα τετράγωνο μηδενικού ύψους και πλάτους, έτοιμο να μετακινηθεί ή να μεγαλώσει αργότερα. Η δεύτερη εντολή δημιουργεί ένα τετράγωνο 50 επί 100

κουκκίδων, ξανά στην επάνω αριστερή γωνία της οθόνης. Τέλος, η τρίτη εντολή δημιουργεί ένα τετράγωνο 50 επί 100 κουκκίδων που απέχει όμως από το επάνω και αριστερό μέρος της οθόνης 25 κουκκίδες.

16) Υπερφόρτωση (Overloading)

Πριν περάσουμε στους τρόπους δημιουργίας αντικειμένων ας δούμε το παρακάτω κομμάτι κώδικα, το οποίο είναι νόμιμο.

```
class number {
int a;
void add(int x) {
a += x;
}
void add(float f) {
a += ((int) f);
}
void add(number n) {
a += n.get_a();
}
int get_a() {
return a;
}
}
```

Βλέπουμε δηλαδή ότι η συνάρτηση void add(...) χρησιμοποιείται πολλές φορές (3) αλλά κάθε φορά με διαφορετική λίστα παραμέτρων. Αυτή η επαναχρησιμοποίηση του ίδιου ονομάτος λέγεται **υπερφόρτωση** (overloading) του ονόματος αυτού.

Στις object oriented (αντικειμενοσταφείς) γλώσσες επιτρέπεται ο ορισμός μίας μεθόδου δύο ή περισσότερες φορές αρκεί να έχουν διαφορετικές λίστες παραμέτρων. Παράδειγμα :
aMethod(), aMethod(type1), aMethod(type2), aMethod(type1, type2), aMethod(type2, type1).

Όλες οι παραπάνω μεθόδοι είναι διαφορετικές μεταξύ τους. Έτσι, κατά την κλήση μιάς μεθόδου πρέπει κατ' αρχήν να εξετάζεται το όνομά της και έπειτα να εξετάζεται και η λίστα των παραμέτρων της, όσον αφορά το πλήθος, τον τύπο και τη σειρά των παραμέτρων. Αυτό είναι μια μορφή πολυμορφισμού.

17) Κατασκευαστές (Constructors)

Κάθε κλάση διαθέτει τουλάχιστον μία μέθοδο η οποία εκτελείται κατά τη δημιουργία ενός στιγμιοτύπου της, (δηλ. κατά τη δημιουργία ενός object της κλάσης αυτής). Αυτές οι μέθοδοι λέγονται κατασκευαστές (constructors) της κλάσης. Σκοπός των constructors είναι η δέσμευση μνήμης για την κατασκευή του αντικειμένου καθώς και η διενέργεια κατάλληλων αρχικοποιήσεων. Ο προγραμματιστής μπορεί να ορίσει πολλούς constructors για μία κλάση αλλά για τη δημιουργία ενός αντικειμένου (στιγμιοτύπου) μπορεί να καλέσει μόνο έναν από αυτούς. Φυσικά η επιλογή είναι ελεύθερη. Αν όμως ο προγραμματιστής δεν ορίσει κανένα constructor τότε η γλώσσα καλεί έναν constructor της υπερκλάσης (κάποιον που δεν θέλει παραμέτρους) ή δίνει μήνυμα λάθους αν δεν βρει κάποιον τέτοιο.

Ένα θέμα που προκύπτει είναι το πώς ορίζονται οι constructors. Στην JAVA ορίζονται όπως ακριβώς και οι απλές μέθοδοι με μόνη τη διαφορά ότι έχουν το ίδιο όνομα με την κλάση, πχ :

```
class A {
int n;
A() { // Constructor - 1
n=0;
}
A(int x) { // Constructor - 2
n = x;
}
}
```

Σε αυτήν την περίπτωση οι constructors ξεχωρίζονται μεταξύ τους από τις λίστες των παραμέτρων τους. Επίσης, οι constructors δεν έχουν επιστρεφόμενο τύπο, ούτε void. (Βλέπε το παραπάνω παράδειγμα).

Η κατασκευή ενός νέου αντικειμένου γίνεται ως εξής:

A a;

*a = **new** A();*

ή

*a = **new** A(5);*

18) Διεπαφές

- Μια διεπαφή (*interface*) (ή διασύνδεση σε άλλα ελληνικά βιβλία) είναι μια μονάδα σχεδιασμού
- Η διεπαφή ορίζει μεθόδους και σταθερές που μπορεί μια κλάση να υλοποιήσει.
- Ο ορισμός μιας διεπαφής είναι παρόμοιος με τον ορισμό μιας κλάσης. Καμιά όμως από τις μεθόδους δεν έχει σώμα. Παράδειγμα, κινητήρας εσωτερικής καύσης.

```
interface InternalCombustionEngine {
 public void start();
 public void stop();
public void setThrottle(int throttleLevel);
 public int getRPM();
}
```

- Όλες οι μέθοδοι σε μια διεπαφή υπονοούνται ως abstract.
- Καμία μέθοδος δε μπορεί να είναι static.
- Τα πεδία που ορίζονται σε μια διεπαφή πρέπει να είναι static και final.
- Τα πεδία αυτά χρησιμοποιούνται για τον ορισμό σταθερών των κλήσεων των μεθόδων.
- Μια κλάση υλοποιεί μια διεπαφή με τον ορισμό implements.
- Η κλάση πρέπει να τις υλοποιήσει όλες τις μεθόδους της διεπαφής, ή να οριστεί ως abstract.
- Μια διεπαφή μπορεί να υλοποιηθεί από πολλές διαφορετικές κλάσεις.

Παράδειγμα:

Τετράχρονος κινητήρας

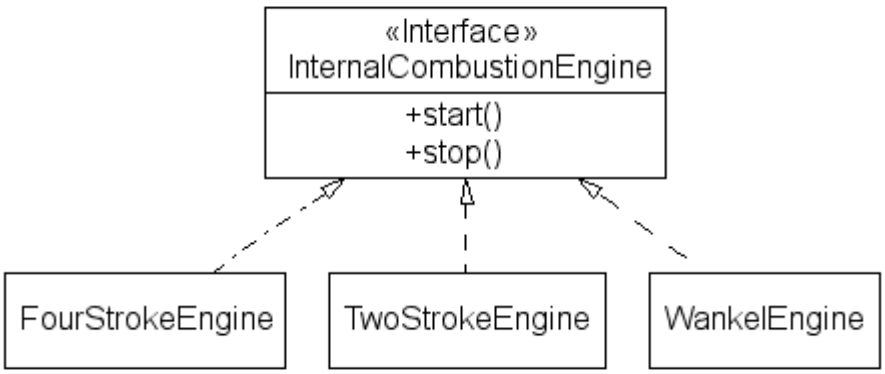
class FourStrokeEngine **implements** InternalCombustionEngine {}

Δίχρονος κινητήρας

class TwoStrokeEngine **implements** InternalCombustionEngine {}

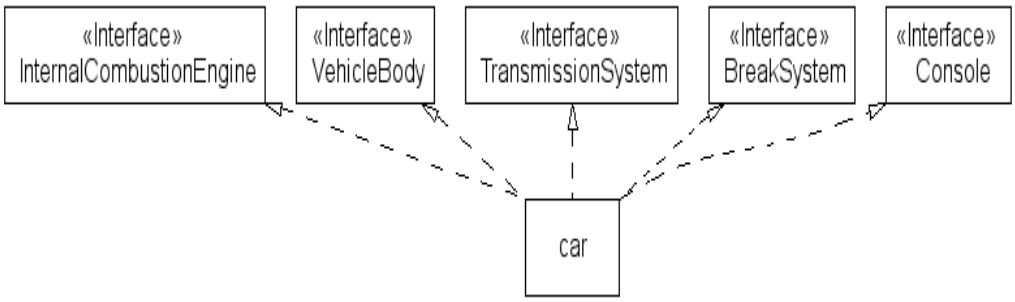
Κινητήρας Wankel

class WankelEngine **implements** InternalCombustionEngine {}



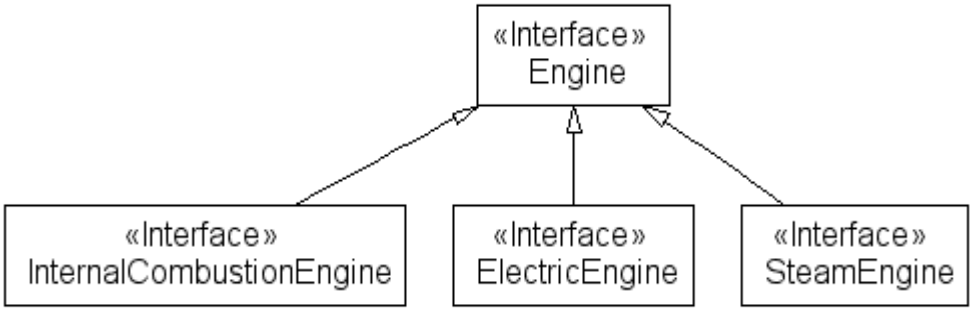
- Μια κλάση μπορεί να υλοποιήσει πολλές διεπαφές.

class Car **implements**
VehicleBody,
InternalCombustionEngine,
TransmissionSystem,
BreakSystem,
Console
{
}
class Truck **implements**
VehicleBody,
InternalCombustionEngine,
TransmissionSystem,
BreakSystem,
Console,
Container
{
}



Με την υποστήριξη πολλαπλών διεπαφών από μια κλάση, μπορούμε στη Java να υλοποιήσουμε σχέδια που βασίζονται σε πολλαπλή κληρονομικότητα (*multiple inheritance*) . Οι διεπαφές μπορούν να επεκταθούν, ακριβώς όπως και οι κλάσεις.

interface Engine {}
interface InternalCombustionEngine **extends** Engine {}
interface ElectricEngine **extends** Engine {}
interface SteamEngine **extends** Engine {}



<u>Η διεπαφή Collection</u>

Η γενικευμένη διεπαφή
Collection <E>
ορίζει μεταξύ άλλων τις παρακάτω μεθόδους:
void clear()
Αφαιρεί όλα τα αντικείμενα από τη συλλογή
boolean add(E o)
Προσθήκη στοιχείου (επιστρέφει αληθές αν η συλλογή μεταβλήθηκε)
boolean addAll(Collection <? extends E> c)
Προσθήκη συλλογής (επιστρέφει αληθές αν η συλλογή μεταβλήθηκε)

boolean remove(Object o)

Αφαίρεση στοιχείου (επιστρέφει αληθές αν η συλλογή μεταβλήθηκε)

boolean removeAll(Collection <?> c)

Αφαίρεση συλλογής (επιστρέφει αληθές αν η συλλογή μεταβλήθηκε)

boolean contains(Object o)

Επιστρέφει αληθές αν το στοιχείο περιέχεται

boolean containsAll(Collection <?> c)

Επιστρέφει αληθές αν τα στοιχεία της συλλογής περιέχονται

Iterator <E> iterator()

Επιστρέφει έναν επαναλήπτη για διάσχιση της συλλογής

boolean isEmpty()

Επιστρέφει αληθές αν η συλλογή είναι άδεια

int size()

Επιστρέφει τον αριθμό των στοιχείων στη συλλογή

Παράδειγμα της διεπαφής Collection

```
import java.util.*;

public class CollectionTest {
	static void testContain(Collection c, Object o) {
		System.out.print("The collection " + c);
		if (c.contains(o))
			System.out.print(" contains");
		else
			System.out.print(" does not contain");
		System.out.println(" the object " + o);
	}

	public static void main(String args[]) {
		LinkedList <String> list = new LinkedList <String>();

		list.add("Petros");
		list.add("Maria");

		testContain(list, "Maria");
```

```
            testContain(list, "John");
            testContain(list, "Petros");
            System.out.println("Number of elements = " + list.size());
        }
}
```

Η διεπαφή Iterator

Η διεπαφή

Iterator <E>

υποστηρίζει τη διάσχιση και μεταβολή συλλογών. Ορίζει τις παρακάτω μεθόδους:

boolean hasNext()

Επιστρέφει αληθές αν υπάρχει επόμενο στοιχείο

Object next()

Επιστρέφει το επόμενο στοιχείο

void remove()

Διαγράφει το τελευταίο στοιχείο που επιστρέφτηκε

Η διεπαφή ListIterator επεκτείνει την Iterator με τις παρακάτω μεθόδους:

boolean hasPrevious()

Επιστρέφει αληθές αν υπάρχει προηγούμενο στοιχείο

Object previous()

Επιστρέφει το προηγούμενο στοιχείο

void add(Object o)

Προσθέτει ένα στοιχείο

void set(Object o)

Αντικαθιστά το τελευταίο στοιχείο που επιστρέφτηκε

Παράδειγμα της διεπαφής Iterator

```
import java.util.*;
public class IteratorTest {
        static void fill(Collection <String> c) {
                c.add("Kerkyra");
                c.add("Zakynthos");
                c.add("Kythira");
                c.add("Santorini");
```

```
            c.add("Dilos");
            c.add("Samos");
            c.add("Rodos");
            c.add("Kastelorizo");
        }

        static void printAll(Collection <?> c) {
            Iterator <?> i;
            for (i = c.iterator(); i.hasNext(); )
                System.out.print(i.next() + " ");
        }

        public static void main(String args[]) {
            HashSet <String> hash = new HashSet<String> ();
            LinkedList <String> list = new LinkedList<String> ();
            fill(hash);
            fill(list);
            System.out.println("Hash contains:");
            printAll(hash);
            System.out.println("\nList contains:");
            printAll(list);
        }
}
```

Διάσχιση δομών με την εντολή foreach

Μπορούμε εύκολα να διασχίσουμε τα μέλη συλλογών που υλοποιούν τη διεπαφή Iterable μέσω της εντολής foreach.

Η χρήση της foreach υποστηρίζεται ακόμα και για τους πίνακες της Java.

Η εντολή συντάσσεται ως εξής

for (Τύπος μεταβλητή : συλλογή)

    εντολή

Η μεταβλητή λαμβάνει διαδοχικά ως τιμή όλες τις τιμές της συλλογής.

Παράδειγμα διάσχισης συλλογής:

static void printAll(Collection <?> c) {

```
        for (Object o : c)
                System.out.print(o + " ");
}
```

Παράδειγμα διάσχισης πίνακα:

```
class Echo {
        public static void main(String args[]) {
                for (String s : args)
                        System.out.print(s + " ");
                System.out.println();
        }
}
```

Παράδειγμα: διάσχιση της ακολουθίας Fibonacci

```
import java.util.Iterator;
import java.math.BigInteger;

/**
 * An Iterable interface over the Fibonacci sequence.
 * @author Diomidis Spinellis
 */
class FibonacciSequence implements Iterable<BigInteger> {

        /** The iterator over the Fibonacci sequence. */
        private class  FibonacciIterator implements Iterator<BigInteger> {
                /** Integer n-2 of the series. */
                private BigInteger n0 = null;
                /** Integer n-1 of the series. */
                private BigInteger n1 = null;

                /**
                 * Return true.
                 * The FibonacciSequence sequence is infinite.
                 */
                public boolean hasNext() { return true; }
```

```
        /** Return the next FibonacciSequence integer. */
        public BigInteger next() {
                if (n0 == null) {
                        n0 = BigInteger.ONE;
                        return n0;
                } else if (n1 == null) {
                        n1 = BigInteger.ONE;
                        return n1;
                } else {
                        BigInteger r = n0.add(n1);
                        n0 = n1;
                        n1 = r;
                        return r;
                }
        }

        /**
         * Remove an element.
         * Nothing to see here; move on.
         */
        public void remove() {
                throw new UnsupportedOperationException();
        }
}

/** Return an iterator for the FibonacciSequence series. */
public Iterator<BigInteger> iterator() {
        return new FibonacciIterator();
}

/** A simple test harness. */
public static void main(String argv[]) {
        FibonacciSequence fib = new FibonacciSequence();
```

```
            for (BigInteger i : fib)
                System.out.println(i);
        }
}
```

Η διεπαφή Map

Η διεπαφή

Map <K, V>

ορίζει μια απεικόνιση από ένα κλειδί (*key*) K σε μια τιμή (*value*) V. Η διεπαφή ορίζει μεταξύ άλλων τις παρακάτω μεθόδους:

void clear()

Αφαιρεί όλα τα αντικείμενα από τη συλλογή

Object put(Object key, Object value)

Προσθήκη στοιχείου (επιστρέφει την προηγούμενη τιμή ή null)

void putAll(Map <? extends K, ? extends V> m)

Προσθήκη στοιχείων

boolean remove(Object key)

Αφαίρεση στοιχείου (επιστρέφει την προηγούμενη τιμή ή null)

V get(Object key)

Επιστρέφει την τιμή του στοιχείου με το κλειδί key

boolean containsKey(Object key)

Επιστρέφει αληθές αν το στοιχείο με το κλειδί key περιέχεται

boolean containsValue(Object value)

Επιστρέφει αληθές αν το στοιχείο με την τιμή value περιέχεται (αργή)

Set<K> keySet()

Επιστρέφει το σύνολο των κλειδιών

Collection<V> values()

Επιστρέφει μια συλλογή με τις αποθηκευμένες τιμές

boolean isEmpty()

Επιστρέφει αληθές αν η συλλογή είναι άδεια

int size()

Επιστρέφει τον αριθμό των στοιχείων στη συλλογή

19) Πακέτα

Σε μεγάλα προγράμματα η πληθώρα των κλάσεων που ορίζονται οδηγεί συχνά σε ρύπανση του χώρου ονοματοδοσίας (*namespace polution*). Αυτό συμβαίνει όταν το ίδιο όνομα χρησιμοποιείται με διαφορετικούς τρόπους σε δύο διαφορετικά τμήματα του προγράμματος. Τοποθετώντας την υλοποίηση σε ένα πακέτο (*package*) μπορούμε να περιορίσουμε το χώρο στον οποίο είναι ορατά τα ονόματα της υλοποίησής μας να ομαδοποιήσουμε σχετικές κλάσεις να επιτρέψουμε σε τρίτους να χρησιμοποιήσουν τις δικές μας κλάσεις ακόμα και αν το όνομά τους ταυτίζεται με άλλες. Ο ορισμός μιας κλάσης σε ένα πακέτο γίνεται αν στο αντίστοιχο αρχείο προσθέσουμε τη λέξη package και το όνομα του πακέτου

```
package gr.aueb.dmst.dds;
class Shape {}
```

Έχει επικρατήσει η ονομασία πακέτων να βασίζεται στο διαδικτυακό όνομα του οργανισμού. Με τον τρόπο αυτό ελαχιστοποιούμε την πιθανότητα σύγκρουσης ονομάτων (*name collision*). Για να χρησιμοποιήσουμε μια κλάση που βρίσκεται μέσα σε ένα πακέτο μπορούμε. Να προτάξουμε το όνομα του πακέτου πριν την κλάση

```
class DrawingEditor {
        gr.aueb.dmst.dds.Shape s;
}
```

Να δηλώσουμε στην αρχή του αρχείου μας πως θέλουμε να κάνουμε ορατές όλες τις κλάσεις του πακέτου με το ονομά τους, χωρίς πρόθεμα. Αυτό γίνεται με την εντολή import.

```
import gr.aueb.dmst.dds;

class DrawingEditor {
        Shape s;
}
```

Μπορούμε επίσης να εισάγουμε ονόματα κλάσεων από πολλά πακέτα στο πρόγραμμά μας με τη σύνταξη import όνομα.*

```
import gr.aueb.dmst.*;
```

Κλάσεις ή πεδίο (μέθοδοι και ιδιότητες) που δεν έχουν οριστεί public φαίνονται μόνο μέσα στο πακέτο στο οποίο ορίζονται. Με τον τρόπο αυτό μπορούμε να προστατεύσουμε τμήματα της υλοποίησής μας από εξωτερικές παρεμβάσεις. Κώδικας που δε βρίσκεται σε ένα συγκεκριμένο πακέτο θεωρείται πως βρίσκεται στο ανώνυμο πακέτο.

ΕΠΑΝΑΛΗΠΤΙΚΕΣ ΕΡΩΤΗΣΕΙΣ 3^{ΟΥ} ΚΕΦΑΛΑΙΟΥ

Οι παρακάτω ερωτήσεις θα σας δώσουν την ευκαιρία να σκεφτείτε πάνω σε αυτά που διαβάσατε στο 2^{ο} κεφάλαιο και να βεβαιωθείτε ότι τα έχετε κατανοήσει.

ΕΡΩΤΗΣΕΙΣ:

1.Θα μπορούσατε ως χρήστης να ανταλλάξετε μηνύματα με ένα αντικείμενο λογισμικού;

_____

_____

2. Ένα μήνυμα λογισμικού είτε είναι εντολής είτε ερώτησης περιλαμβάνει στο τέλος εισαγωγικά στα οποία ορίζονται οι παράμετροι του μηνύματος. Εάν το μήνυμα δεν περιλαμβάνει παραμέτρους θα μπορεί να αναγνωστεί από τον παραλήπτη ή με απλά λόγια, αν στείλουμε ένα μήνυμα με κενά εισαγωγικά, θα είναι αυτό ορθό;

_____

_____

_____

_____

3. Θα ήταν δυνατό για μία μη αντικειμενοστραφή γλώσσα να δημιουργήσει κώδικα για το ιντερνέτ;

_____

_____

_____

ΑΣΚΗΣΕΙΣ

1. Έστω ότι θέλετε να βάλετε σε λειτουργία το ξυπνητήρι του κινητού σας για να χτυπήσει στις 07:30 πμ.. Γράψτε το μήνυμα με το οποίο θα δίνατε την εντολή στο ξυπνητήρι θα χτυπήσει τη συγκεκριμένη ώρα.
Έπειτα γράψτε το μήνυμα με το οποίο θα ζητούσατε από το ξυπνητήρι να επαναλάβει το χτύπημα 2,5 λεπτά αργότερα.

2. Ποια είναι τα τρία βασικά χρώματα που χρησιμοποιούνται το λογισμικό και ποιοί οι συνδυασμοί τους; Πώς θα γράφατε το χρώμα μωβ;

3. Να γίνει μία τάξη Book η οποία θα χρησιμοποιηθεί για τα βιβλία μιας βιβλιοθήκης. Για κάθε βιβλίο να δηλώσετε τις μεταβλητές title (String) με τον τίτλο του βιβλίου, author (επίσης String) για τον συγγραφέα και firstEdition (int) που είναι το έτος της πρώτης έκδοσης. Όλες οι μεταβλητές θα πρέπει να είναι ιδιωτικές (private).
Να δώσετε μεθόδους get και set για όλες τις ιδιότητες που αναφέρθηκαν παραπάνω.
Στη συνέχεια δημιουργείστε ένα πρόγραμμα το οποίο θα δημιουργήσει ένα πίνακα τριών βιβλίων. Τέλος η main θα πρέπει να ταξινομήσει τον πίνακα αυτών των βιβλίων χρησιμοποιώντας οποιαδήποτε μέθοδο ταξινόμησης θέλετε, με αύξουσα σειρά ως προς το έτος πρώτης έκδοσης. Αφού η main κάνει τη ταξινόμηση θα πρέπει να εμφανίσει τα τρία βιβλία με ένα String του στυλ "Βιβλίο: <όνομα-βιβλίου>, Συγγραφέας: <όνομα-συγγραφέα>, Έτος πρώτης έκδοσης: <έτος-πρώτης-έκδοσης>.

4. Στην προηγούμενη άσκηση η τάξη Book έχει την ιδιότητα Author για τον συγγραφέα, την οποία προηγουμένως υλοποιήσαμε ως String. Να δηλώσετε μία τάξη Author η οποία θα έχει την ιδιότητα name για το όνομα του συγγραφέα που θα είναι String και get και set μεθόδους γι' αυτή την ιδιότητα. Να συμπεριλάβετε στην τάξη του συγγραφέα και ένα πίνακα 20 το πολύ βιβλίων στον οποίο θα τοποθετήσουμε τα βιβλία του συγγραφέα. Τα βιβλία θα

είναι αντικείμενα της τάξης Book της προηγούμενης άσκησης. Από την τάξη Book της προηγούμενης άσκησης να αλλάξετε την ιδιότητα Author και τις μεθόδους setAuthor και getAuthor έτσι ώστε να είναι και να επενεργούν πάνω σε ένα αντικείμενο τύπου Author. Για την τάξη Author θα πρέπει επίσης να συμπεριλάβετε δύο μεθόδους:

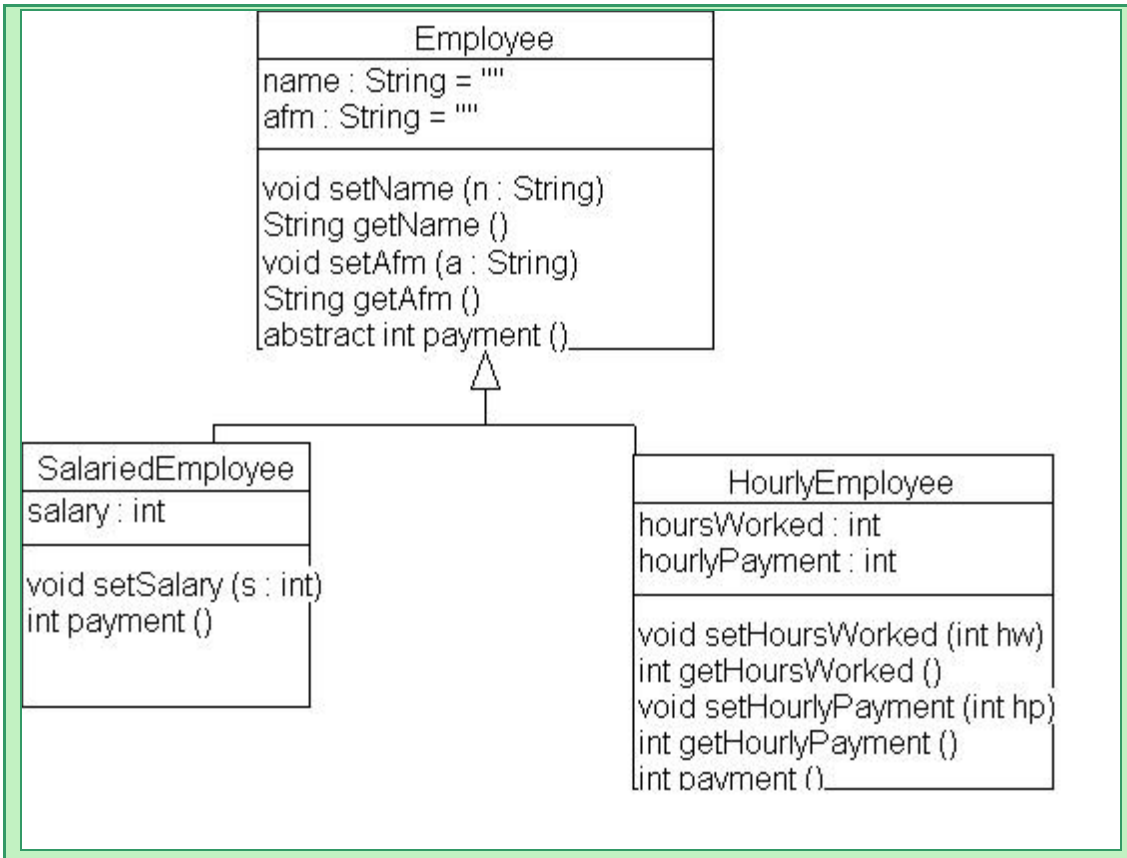
Την public void setter(int index, Book b) throws ArrayIndexOutOfBoundsException η οποία θα θέτει στην θέση index του πίνακα των βιβλίων ενός συγγραφέα το βιβλίο που δίνεται σαν δεύτερη παράμετρο. Η μέθοδος θα πρέπει να προκαλεί την εξαίρεση ArrayIndexOutOfBoundsException στην περίπτωση που η θέση index είναι εκτός των ορίων του πίνακα.

Την public Book[] getter() η οποία επιστρέφει τον πίνακα των βιβλίων του συγγραφέα.

Αφού δηλώσετε τις ποιο πάνω τάξεις κάνετε πρόγραμμα στο οποίο θα δημιουργήσετε ένα αντικείμενο τύπου Author με τον αγαπημένο σας συγγραφέα. Στη συνέχεια θα δημιουργήσετε δύο βιβλία του και θα κάνετε τις απαραίτητες συσχετίσεις. Τέλος θα εμφανίσετε λίστα με τα βιβλία του συγγραφέα, χρησιμοποιώντας την getter μέθοδο για τον συγγραφέα αυτό.

5. Να γίνει Interface CryingObject το οποίο θα διαθέτει τη μέθοδο public void cry(). Στη συνέχεια δημιουργείστε την τάξη Baby που θα υλοποιεί το Interface αυτό. Η μέθοδος cry για το Baby απλά θα εμφανίζει στην οθόνη το μήνυμα "Crying all night long!". Επίσης δημιουργείστε τάξη BluesSinger η οποία επίσης θα υλοποιεί το InterfaceCryingObject και της οποίας η μέθοδος θα εμφανίζει το μήνυμα "Crying when singing!".
Στη συνέχεια δημιουργείστε τάξη Main με μέθοδο main η οποία θα δημιουργεί πίνακα τύπου CryingObject με δύο θέσεις . Στη πρώτη θέση τοποθετείστε ένα νέο αντικείμενο της τάξης Baby ενώ στη δεύτερη τοποθετείστε ένα αντικείμενο της τάξης BluesSinger. Τέλος, η main θα εκτελεί βρόχο (loop) στον οποίο θα καλεί τη μέθοδο cry σε όλα τα αντικείμενα του πίνακά της.

6. Να γίνει πρόγραμμα με τις ακόλουθες τάξεις όπως φαίνονται στο UML διάγραμμα που δίνεται:



Η τάξη Employee είναι υπερτάξη των τάξεων SalariedEmployee και HourlyEmployee. Η υπερτάξη έχει τις ιδιότητες name (όνομα) και afm (ΑΦΜ) και μεθόδους για να θέτουμε και να ανακτούμε τις ιδιότητες αυτές. Επίσης έχει την αφηρημένη (abstract) μέθοδο payment(). Επομένως και η τάξη Employee θα πρέπει να δηλωθεί ως abstract.

Η υποτάξη SalariedEmployee (τακτικός υπάλληλος) έχει επίσης την ιδιότητα salary και μία set μέθοδο για να θέτει αυτή την ιδιότητα. Επίσης η μέθοδος δηλώνει την αφηρημένη τάξη payment της υπερτάξης έτσι ώστε να επιστρέφει τον salary.

Η υποτάξη HourlyEmployee (ωρομίσθιος υπάλληλος) έχει επίσης τις ιδιότητες hoursWorked (ώρες εργασίας) και hourlyPayment (ωρομίσθιο). Επίσης δηλώνει τις μεθόδους get και set γι' αυτές τις ιδιότητες.Τέλος, δηλώνει την μέθοδο payment που είναι ο ορισμός της abstract μεθόδου της υπερτάξης. Η μέθοδος αυτή θα επιστρέφει το γινόμενο των ωρών εργασίας επί του

ωρομισθίου.

Να κατασκευάσετε την πιο πάνω ιεραρχία τάξεων και στη συνέχεια να κάνετε μία τάξη Main με μία συνάρτηση main. Η main θα έχει ένα πίνακα δυο θέσεων ο οποίος θα είναι τύπου Employee. Στην θέση 0 του πίνακα να δημιουργήσετε και να βάλετε ένα αντικείμενο SalariedEmployee και στη θέση 1 του πίνακα να δημιουργήσετε και να βάλετε ένα αντικείμενο HourlyEmployee.

Στη συνέχεια να θέσετε τα στοιχεία του πρώτου υπαλλήλου και τον μισθό του και τα στοιχεία του δεύτερου υπαλλήλου συμπεριλαμβανομένων και των ωρών εργασίας και του ωρομισθίου. Τέλος, η main θα πρέπει να εκτελεί ένα βρόχο στον οποίο θα εμφανίσει τον μισθό των δύο υπαλλήλων καλώντας τη μέθοδο payment στα αντικείμενα του πίνακά της.

Στο 4^{ο} κεφάλαιο θα δούμε αναλυτικά τις μεταβλητές στη Java. Αρχικά θα μάθουμε πως ορίζονται: τι είναι μια μεταβλητή και ποιες λέξεις κλειδιά χρησιμοποιούνται για τον ορισμό της. Ποιους τύπους δεδομένων μπορούν να πάρουν και ποιοι είναι αυτοί, μέσα από αναλυτικούς πίνακες και παραδείγματα. Πως ονομάζουμε μια μεταβλητή και ποιοί περιορισμοί ισχύουν. Ποιά είναι η εμβέλεια τους και τέλος τις τελικές μεταβλητές και τη χρησιμότητα τους σε ένα πρόγραμμα.

Ακόμα, μέσα από παραδείγματα θα δείτε πως κατασκευάζουμε ένα πρόγραμμα με μεταβλητές.

ΚΕΦΑΛΑΙΟ 4

1) Ορισμός μεταβλητών

Οι μεταβλητές είναι οι θέσεις μνήμης στις οποίες ένα πρόγραμμα τοποθετεί τα δεδομένα του κατά τη διάρκεια της λειτουργίας του. Σ' αυτό το κεφάλαιο θα μάθουμε πως δηλώνουμε μεταβλητές, ποιοί είναι οι τύποι των μεταβλητών στη γλώσσα Java, πως πρέπει να είναι το όνομα μιας μεταβλητής καθώς και ποια είναι η εμβέλεια της μεταβλητής.

Έχει ήδη γίνει αναφορά στο προηγούμενο κεφάλαιο για τις μεταβλητές της Java. Σε αυτό όμως το κεφάλαιο θα δούμε πιο αναλυτικά τι είναι οι μεταβλητές στη Java και πως χρησιμοποιούνται. Συνεχίζοντας την ανάλυση για τον ορισμό και δημιουργία μιας μεταβλητής, θα γίνει αναφορά στα οκτώ διαφορετικά είδη τιμών τα οποία θα χρησιμοποιηθούν κατά την εγγραφή του κώδικα. Αυτή είναι αναγκαία διαδικασία γιατί η Java απαιτεί από τον προγραμματιστή να ορίσει λεπτομερώς το όνομα και το είδος τιμών που θα πάρει μια μεταβλητή πριν τη χρήση της στο πρόγραμμα. Αν πρέπει να οριστεί, για παράδειγμα μια μεταβλητή που συμβολίζει το πλήθος των ωρών που έχει μια ημέρα, τότε θα ορίζαμε την μεταβλητή ως:

```
int day=24;
```

Η παραπάνω γραμμή κώδικα ειδοποιεί το υπόλοιπο πρόγραμμα ότι υπάρχει μια μεταβλητή με το όνομα «day» η οποία θα παίρνει ακεραίους αριθμούς «int», και στο συγκεκριμένο κώδικα έχει αρχική τιμή «24». Η λέξη int περιορίζει το είδος των τιμών που μπορεί να πάρει η μεταβλητή και το είδος των πράξεων που μπορούν να πραγματοποιηθούν με αυτή. Εκτός από την λέξη-κλειδί int, η Java υποστηρίζει ακόμα 7 είδους τιμών που τις ονομάζει primitive data types και θα τους δούμε παρακάτω.

2) Τύποι μεταβλητών

Στους παρακάτω δύο πίνακες παραθέτουμε τους τύπους δεδομένων που θα συναντήσετε στη Java. Στον πρώτο πίνακα παρουσιάζονται περιγραφικά οι τύποι δεδομένων, το μέγεθος τους σε bits και το εύρος των τιμών που παίρνουν. Ενώ στο δεύτερο παρουσιάζεται πιο αναλυτικά το είδος των δεομένων κάθε τύπου.

Data Type	Size	Min Value	Max Value
byte	8 bits	-128	127
short	16 bits	-32768	32767
int	32 bits	-2147483648	2147483647
long	64 bits	-9223372036854775808	9223372036854775808
float	32 bits	±1.40239846E-45	±3.40282347E+8
double	64 bits	±4.94065645841246544E-324	4.94065645841246544E-324
char	16 bits	\u0000	\uFFFF
boolean	n/a	true or false	

Πίνακας 1

Τύπος	**Μέγεθος και μορφή**	**Περιγραφή**
(ακέραιες μεταβλητές)		
byte	8 bit σε μορφή συμπληρώματος ως προς 2	Ακέραιος μήκους ενός byte
short	16 bit σε μορφή συμπληρώματος ως προς 2	Ακέραιος μικρού μήκους
int	32 bit σε μορφή συμπληρώματος ως προς 2	Ακέραιος
long	64 bit σε μορφή συμπληρώματος ως προς 2	Ακέραιος μεγάλου μήκους
(πραγματικές μεταβλητές)		
float	32 bit σε μορφή IEEE 754	Πραγματικός απλής ακρίβειας
double	64 bit σε μορφή IEEE 754	Πραγματικός διπλής ακρίβειας
(άλλοι τύποι)		

char	16-bit Unicode χαρακτήρας	Ένας χαρακτήρας
boolean	true ή false	Μία boolean τιμή αληθής ή ψευδής

Πίνακας 2

Παρατηρήστε ότι το μέγεθος του εύρους για κάθε data type ορίζεται αυστηρά. Για παράδειγμα, μια μεταβλητή είδους int έχει εύρος ακέραιων αριθμών μεγέθους 32-bit. Ο λόγος που η Java είναι τόσο απόλυτη με τον ορισμό για κάθε data type, είναι επειδή κάθε java πρόγραμμα τρέχει πάνω στο JVM και όχι στο λειτουργικό σύστημα. Το λειτουργικό σύστημα, όποιο και να είναι, δεν επηρεάζει με κανένα τρόπο το εύρος η το μέγεθος των αριθμών.

Παράδειγμα με μεταβλητές

```
puplic class NumberLab extends Applet{
        byte a;
        short b;
        int c;
        long d;
        float e;
        double f;
        char g;
        boolean h;
        puplic void init(){
        d=9000000323;
        c=(int) d;
        System.out.println('c='+c);
        b=(short) c;
        System.out.println ('b='+b);
        a=(byte) b;
        System.out.println('a='+a);
        e=(float) b;
        System.out.println('e='+e);
        e=(float) d;
```

```
System.out.println('e='+e);
f=(double) d;
System.out.println('f='+f);
g=a;
b=(short) g;
System.out.println('b='+b);
}
}
```

Αποτέλεσμα:c=410065731, b=6979, a=67, e=6979, e=4,10066e+08, f=4,10066e+08, g=C, b=97

Ας επεξηγήσουμε όμως πως γίνεται ο ορισμός μιας μεταβλητής. Ορίζοντας int day = 24;, το πρόγραμμα κρατεί στην μνήμη του υπολογιστή 32-bits χώρο που τον ονομάζει day και περιμένει να τον γεμίσει με κάποιον αριθμό. Στην συγκεκριμένη περίπτωση με τον αριθμό 24.

Βάζοντας μαζί όλα αυτά τα βήματα για τον ολοκληρωμένο ορισμό μιας μεταβλητής, η σειρά θα είναι ως εξής: Βήμα πρώτο, ονομάζουμε την μεταβλητή μας ακλουθώντας τους κανόνες ονομασίας και αποφεύγοντας τη χρήση λέξεων κλειδιά. Βήμα δεύτερο, ορίζουμε το είδος τιμών που θα δέχεται η μεταβλητή. Παρακάτω δίνονται ορισμένα παραδείγματα. Υπενθυμίζουμε ότι μια μεταβλητή int μπορεί να δεχτεί μεγαλύτερο εύρος αριθμών από μια μεταβλητή short.

```
short x;

int age;

float salary;
```

Για να μάθουμε τις αρχικές τιμές που θα πάρει μια μεταβλητή από την αρχή του προγράμματος, τότε μπορούμε να ορίσουμε μια αρχική τιμή στην μεταβλητή με τον assignment operator «=».

```
int x = 22;
```

Αν προσπαθήσουμε να βάλουμε λάθος τιμή στην μεταβλητή χ όπως π.χ. δεκαδικούς (14.7) η java θα παραπονεθεί γιατί απλούστατα εξ ορισμού μια int μεταβλητή δεν μπορεί να δέχεται τίποτα άλλο εκτός από ακέραιους αριθμούς.

Τώρα πολλοί από εσάς ίσως να αναρωτιέστε τι συμβαίνει με τις μεταβλητές όταν δεν ξέρουμε την αρχική τους τιμή και απλά τις ορίζουμε. Τι συμβαίνει με το μέρος της μνήμης που έχει αποθηκεύσει το όνομα αλλά δεν έχει τιμή;

Σε περίπτωση που δεν είναι γνωστή η αρχική τιμή της μεταβλητής, η Java έχει φροντίσει ώστε εάν ο προγραμματιστής δεν γνωρίζει τις αρχικές τιμές, τότε μπορεί απλά να δώσει την αριθμητική τιμή μηδέν σαν αρχική τιμή, αλλιώς η java θα το κάνει αυτόματα. Ο λόγος που συμβαίνει αυτό, είναι για να μην αποκτήσει η μεταβλητή λάθος τιμή από μονή της παίρνοντας την τυχαία τιμή που υπάρχει στην μνήμη του υπολογιστή εκείνη την ώρα. Για να μην πάρουμε λοιπόν κάποια τυχαία τιμή, η java μηδενίζει τις τιμές από μόνη της.

Στον παρακάτω πίνακα φαίνονται οι τιμές που δίνει αυτόματα η Java στις μεταβλητές

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Πίνακας 3

Ξεκινώντας την ανάλυση των ειδών των μεταβλητών, θα αναφερθούμε πρώτα στα τέσσερα ειδή ακέραιων που υποστηρίζει η java τα οποια είναι byte, short, int και long. Και τα τέσσερα αυτά είδη είναι signed primitive date types (όπως αναφέρθηκε και νωρίτερα), δηλαδή το εύρος των τιμών τους καλύπτει και το αρνητικό και το θετικό φάσμα αριθμών.

Για να δηλώσουμε μεταβλητές στην Java και να δώσουμε και αρχικές τιμές, μπορούμε να γράψουμε τα εξής :

```
int a;

float p;

float weight = 90;

char initial = 'A';

String name = "George";

boolean married = true;
```

3) Ονόματα μεταβλητών

Ένα πρόγραμμα Java αναφέρεται σε μία μεταβλητή με το όνομά της.Ένα όνομα μιας μεταβλητής θα πρέπει να ικανοποιεί τους ακόλουθους περιορισμούς:

1) Πρέπει να είναι ένα έγκυρο αναγνωριστικό της Java αποτελούμενο από μία σειρά Unicode χαρακτήρων.
2) Δεν πρέπει να είναι δεσμευμένη λέξη της Java (keyword).
3) Δεν πρέπει να υπάρχει άλλο ίδιο όνομα μεταβλητής στην περιοχή εμβέλειας της μεταβλητής. Στη Java συνηθίζονται τα ακόλουθα:

Τα ονόματα των τάξεων ξεκινάνε με κεφαλαία γράμματα Τα ονόματα των μεταβλητών ξεκινάνε με πεζά γράμματα Στη περίπτωση που έχουμε αναγνωριστικό πολλών λέξεων τότε ενώνουμε τις λέξεις γράφοντας με κεφαλαίο το πρώτο γράμμα κάθε λέξης.

4) Εμβέλεια μεταβλητών (scope)

Η εμβέλεια μίας μεταβλητής είναι η περιοχή εκείνη του προγράμματός μας στην οποία η μεταβλητή είναι ορατή και καθορίζει πότε η μεταβλητή δημιουργείται και καταστρέφεται. Υπάρχουν τέσσερα ήδη μεταβλητών με αντίστοιχα τέσσερις περιοχές εμβέλειας:

Μεταβλητές μέλη με εμβέλεια την τάξη.

Τοπικές μεταβλητές με εμβέλεια την ομάδα εντολών στην οποία δηλώνονται.

Παράμετροι μεθόδων με εμβέλεια την μέθοδο μέσα στην οποία δηλώνονται

και Παράμετροι χειριστών εξαιρέσεων με εμβέλεια την ομάδα εντολών του χειριστή εξαιρέσεων

5) Τελικές μεταβλητές και παράμετροι

Μπορούμε να δηλώσουμε μία μεταβλητή ή μία παράμετρος σε μία μέθοδο ως τελική (final).Η δήλωση final έχει ως αποτέλεσμα την απαγόρευση της αλλαγής της τιμής στην εμβέλεια στην οποία ανήκει η μεταβλητή. Δηλαδή η μεταβλητή είναι στη πραγματικότητα σταθερά για την εμβέλεια αυτή.

Στην περίπτωση της μεταβλητής η λέξη final θα πρέπει λογικά να χρησιμοποιηθεί με κάποια αρχικοποίηση για την απόδοση της αρχικής και ταυτόχρονα οριστικής τιμής στη μεταβλητή, όπως στο ακόλουθο παράδειγμα:

```
int interestRate = 0.15;
```

Μπορούμε όμως να αναβάλλουμε την απόδοση της αρχικής τιμής και γι’ αργότερα αν αυτό είναι αναγκαίο. Η αρχική τιμή φυσικά μπορεί να αποδοθεί μόνο μία φορά. Μία τελική μεταβλητή στην οποία δεν έχει αποδοθεί ακόμα τιμή ονομάζεται κενή (blank).

Στην περίπτωση μιας παραμέτρου σε μία μέθοδο η δήλωση final έχει το νόημα ότι η συγκεκριμένη παράμετρος δεν μπορεί να αλλάξει τιμή μέσα στη μέθοδο αλλά θα διατηρηθεί η τιμή που θα περαστεί από το καλούν τμήμα του προγράμματος.

Για παράδειγμα η μέθοδος withdraw δεν μπορεί να μεταβάλλει τη παράμετρο money , διότι η παράμετρος αυτή έχει δηλωθεί ως τελική (final). Μπορεί μόνο να την χρησιμοποιήσει.

Συχνά στο πρόγραμμά μας χρησιμοποιούμε αριθμητικές ή άλλες σταθερές (*constants*).

Αν οι σταθερές εμφανίζονται μέσα στον κώδικα αυτό τον κάνει δυσνόητο και δύσκολο να συντηρηθεί.

```
    double area = 6 * length * width;
```

Με τον προσδιοριστή final μπορούμε να ορίσουμε μεταβλητές των οποίων η τιμή δε θα αλλάξει κατά τη διάρκεια εκτέλεσης του προγράμματος.

```
final int CUBE_FACETS = 6;
double area = CUBE_FACETS * length * width;
```

Αντίστοιχα μπορεί μια κλάση να ορίσει δημόσιες σταθερές ιδιότητες ως μέλη για να μπορούν να χρησιμοποιηθούν από άλλες κλάσεις.

```
class PhysicalConstants {
// Speed of light in vacuum (m/s)
public static final double c = 299792458;
// Permeability of vaccum (N/A/A)
public static final double mu0 = 4 * Math.PI * 10e-7;
// Newtonian constant of gravitation (m*m*m/kg/s/s)
public static final double G = 6.67259e-11;
// Planck constant (J*s)
public static final double h = 6.6260755e-34;
// Elementary charge (C)
public static final double e = 1.60217733e-19;
}
```

Οι αριθμητικές σταθερές τυπικά ονομάζονται με κεφαλαία γράμματα.

Τελικές κλάσεις και μέθοδοι

Με τον προδιοριστή final μπορούμε να περιορίσουμε τη χρήση κληρονομικότητας σε κλάσεις και μεθόδους.

Μια κλάση με τον προσδιοριστή final δεν μπορεί να επεκταθεί.

```
final class PasswordChecker {
// Class contents here
}
```

Μια μέθοδος με τον προσδιοριστή final δεν μπορεί να αντικατασταθεί σε μια υποκλάση.

```
class Password {
final boolean isPasswordValid() { /* ... */ }
}
```

Με τον τρόπο αυτό μπορούμε να είμαστε σίγουροι πως μια συγκεκριμένη υλοποίηση στην οποία βασίζεται το πρόγραμμά μας δε θα αλλάξει από άλλες κλάσεις.
Τελικές κλάσεις και μεταβλητές επιτρέπουν στο μεταγλωττιστή να πραγματοποιήσει και ορισμένες βελτιστοποιήσεις στον παραγώμενο κώδικα.

Η απόφαση όμως για τη χρήση του προσδιοριστή πρέπει να βασίζεται αποκλειστικά στο σχεδιασμό του προγράμματος.

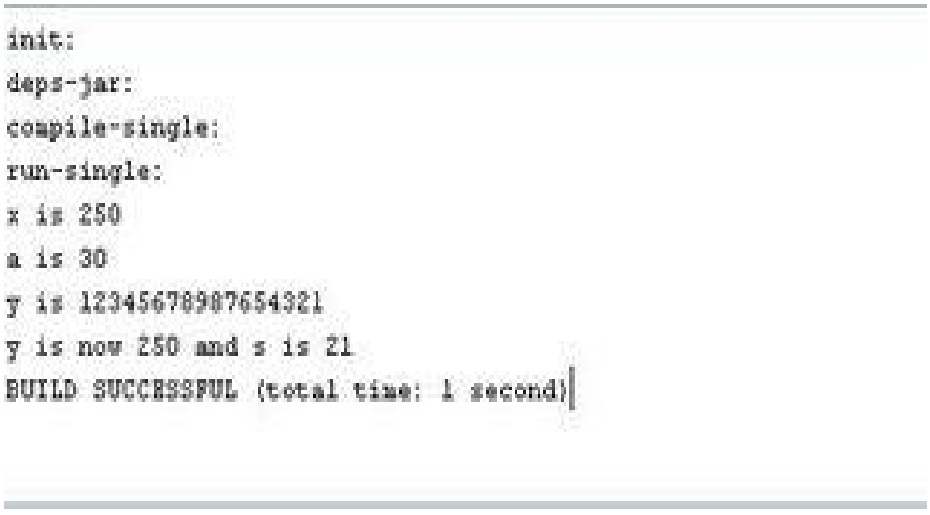
6) Πρόγραμμα με μεταβλητές

Ας δούμε όμως καλύτερα τι γίνεται στην πράξη. Θα γράψουμε ένα πρόγραμμα και θα το αναλύσουμε γραμμή προς γραμμή. Ξεκινάτε με τη διαδικασία στο NetBeans. Πρώτα θα δημιουργήσετε ένα πακέτο με το όνομα IntExample κάνοντας δεξί κλικ επάνω στο Source Packages του project που έχετε δημιουργήσει. Μετά θα κάνετε δεξί κλικ επάνω στο πακέτο και θα διαλέξετε την επιλογή Java Class. Πρέπει να ονομάσετε την κλάση σας ακριβώς όπως το πρόγραμμα σας. Γράφετε το παρακάτω πρόγραμμα. Αν το έχετε γράψει σωστά, κάνετε δεξί κλικ επάνω στο αριστερό παραθυράκι και διαλέγετε την επιλογή compile, και μετά την ολοκλήρωση της διαδικασίας του compilation κάνετε πάλι κάνετε δεξί κλικ επάνω στο αρχείο και διαλέγετε την επιλογή Run.

```
 public class IntExample
 {
 public static void main(String [] args)
{
 int x = 250;
 System.out.println("x is " + x);
```

```
 short a, b, c;
 c = 21;
 b = 9;
 a = (short) (b + c);
 System.out.println("a is " + a);

 long y = 12345678987654321L;
 System.out.println("y is " + y);

 y = x;
 byte s;
 s = (byte) c;
 System.out.println("y is now " + y + " and s is " + s);
 }
}
```

Αν το πρόγραμμα τρέξει χωρίς κανένα απολύτως πρόβλημα το αποτέλεσμα στην οθόνη σας πρέπει να είναι το εξής:



Ανάλυση:

Κάθε πρόγραμμα java ξεκάνει πάντα με τον ορισμό μιας κλάσης ανεξάρτητου αν είναι εκτελέσιμη η όχι. Η λέξη public υποδηλώνει ότι στην κλάση έχουν πρόσβαση οποιοσδήποτε από άλλη κλάση από το ίδιο πακέτο, η ακόμα και από το ίδιο project.

Με άλλα λόγια ορίζει ελεύθερη πρόσβαση στην κλάση από οπουδήποτε. Μπορούμε να το θεωρήσουμε σαν μια universal πρόσβαση.

Η λέξη class περνάει το μήνυμα στον compiler ότι το αρχείο το όποιο θα δημιουργηθεί θα είναι μια κλάση. IntExample είναι το όνομα της κλάσης το όποιο θα πρέπει να είναι ακριβώς το ίδιο με το όποιο αποθηκεύεται το αρχείο στον σκληρό δίσκο. Το άγκιστρο που ανοίγει θα περιλάβει ότι κώδικα γράψουμε μέσα στην συγκεκριμένη κλάση. Το ζεύγος του θα είναι βασικά και το τελευταίο άγκιστρο που θα κλείσει στο τέλος του αρχείου.
Ότι κώδικα περικλείουμε μέσα στην μέθοδο main (δηλαδή μέσα στα άγκιστρα της main) θα εκτελεστεί. Μια κλάση χωρίς την main δεν είναι εκτελέσιμη. Με αλλά λόγια, ένα java προγραμμα δεν μπορεί να τρέξει χωρίς να υπάρχει η μέθοδος main.

Στην Πέμπτη γραμμή ορίζετε μια μεταβλητή με το όνομα x η οποία θα δέχεται integer (ακεραίους) αριθμούς και της δίνουμε αρχική τιμή 250. Το semicolon (;) είναι αναγκαίο στην σύνταξη της java γιατί υποδηλώνει μια ολοκληρωμένη πρόταση προγραμματισμού η αλλιώς statement όπως είναι η αγγλική ονομασία της.

Η έκτη γραμμή απλά δείχνει στην οθόνη το εξής μήνυμα “ x is 250 ”. Αν κοιτάξετε πάλι το αποτέλεσμα στο NetBeans όταν τρέξατε το πρόγραμμα θα δείτε ότι είναι η πρώτη γραμμή στο αποτέλεσμα που παρουσιάζει το output παράθυρο κάτω αριστερά στο Netbeans περιβάλλον.

Η System.out.println είναι ο τρόπος που παρουσιάζεται το αποτέλεσμα μας στην κονσόλα. Στην παρένθεση μέσα ότι περικλείεται σε διπλά " " σύμβολα σημαίνει ότι θα εμφανιστεί όπως ακριβώς είναι γραμμένο. Ενώ το +x μετά από το κλείσιμο των " " σημαίνει ότι θα εμφανίσει δίπλα στο μήνυμα " x is" ότι ακριβώς είναι η τρέχουσα τιμή του χ που έχει οριστεί στο πρόγραμμα. Εδώ αξίζει να σημειωθεί το γεγονός ότι επειδή το + σύμβολο ενώνει από τα αριστερά μια σειρά χαρακτήρων String (δηλαδή παραπάνω από έναν χαρακτήρα κλεισμένους μέσα σε διπλά " ") και από τα δεξιά integer (ακεραίους), τότε μετατρέπει επίσης τους ακεραίους σε String για να μπορέσει να δείξει το αποτέλεσμα. Στις σειρές από οχτώ ως έντεκα συμβαίνει το εξής: ενώ ορίζουμε τις τρεις μεταβλητές a, b και c να δέχονται short (16 bit) ακεραίους, θα περιμέναμε με την απλή λογική ότι short + short να μας έδινε short σαν αποτέλεσμα. Όμως αυτό δεν συμβαίνει, και ο λόγος είναι πολύ απλός. Η java όπως ήδη έχουμε

αναφέρει έχει σαν default ακέραιο τους integer αριθμούς. Οπότε, κάθε πράξη που βλέπει μεταξύ ακέραιων προσπαθεί να δώσει το αποτέλεσμα σαν integer. Για να πάρουμε το σωστό αποτέλεσμα από μια απλή πρόσθεση, χρησιμοποιούμε τη μέθοδο μετατροπής που λέγεται casting. Με άλλα λόγια, μπροστά από την πράξη που χρειάζεται μετατροπή, ανοίγουμε παρένθεση και γράφουμε σε ποιο είδος αριθμών θέλουμε να μετατρέψουμε το αποτέλεσμα πριν δοθεί με το σύμβολο της ισότητας στην μεταβλητή a. Άρα, στον παραπάνω κώδικα, η μετατροπή της πρόσθεσης των αριθμών b+c από short σε integer γίνεται πριν δοθεί το αποτέλεσμα στην μεταβλητή a που περιμένει να της δοθεί ένας short αριθμός σαν απάντηση.

Στη δωδέκατη σειρά ζητάμε να δούμε την τρέχουσα τιμή που τελικά κατέχει η μεταβλητή a, και το πετυχαίνουμε χρησιμοποιώντας την println μέθοδο.

Ενώ στις σειρές δεκατέσσερα και δεκαπέντε ορίζουμε μια μεταβλητή y να δέχεται long τιμές. Ο τρόπος που το ορίζουμε αυτό είναι ακριβώς ο ίδιος όπως κάναμε και πρωτύτερα με την int μεταβλητή. Όμως, υπάρχει μια μικρή λεπτομέρεια. Όποτε ορίζουμε έναν long αριθμό πρεπει στο τέλος του αριθμού να βάλουμε το L για να μπορεσει να καταλαβει ο compiler της java ότι ο αριθμός είναι long. Η System.out.println("y is " + y) νομίζω έχετε ήδη καταλάβει ότι απλά θα μας δείξει το μήνυμα "y is 12345678987654321". Μπορείτε να βεβαιωθείτε κοιτώντας άλλη μια φορά το αποτέλεσμα στο Netbeans.

Στη δέκατη έβδομη σειρά το σύμβολο "=" έχει την ιδιότητα να περάσει την τιμή που έχει η μεταβλητή από τα δεξιά, στην αριστερή μεταβλητή. Δηλαδή η μεταβλητή y έχει την ίδια τιμή με το x, δηλαδή 250.

Στη δέκατη ένατη σειρά κάνουμε το casting. Προσπαθούμε να περάσουμε μια τιμή που ορίζεται σαν short (16 bits) , σε μια τιμή που είναι byte (8 bits). Με άλλα λόγια, προσπαθούμε να περάσουμε ένα μεγαλύτερο εύρος τιμών, σε ένα μικρότερο. Σε αυτές τις περιπτώσεις αν η Java δει ότι ένας αριθμός μικρότερος σε bits όπως ένας byte, πάει να περάσει την τιμή μου σε μια μεταβλητή που περιγράφεται από ένα data type μεγαλύτερο σε bits, τότε κάνει το casting προς τα πάνω μονή της.

Κλείνοντας το πρόγραμμα, ζητάμε να δούμε την τελική τιμή των μεταβλητών y και s.

ΕΠΑΝΑΛΗΠΤΙΚΕΣ ΕΡΩΤΗΣΕΙΣ 4^{ΟΥ} ΚΕΦΑΛΑΙΟΥ

Ερωτήσεις:

1. Ποιοί είναι οι 7 τύποι δεδομένων των μεταβλητών της Java και πόσα bits μέγεθος έχουν ο κάθε ένας;

_____

_____

_____

2. Γράψτε σε φθίνουσα σειρά τους τύπους δεδομένων ανάλογα το μέγεθός τους.

_____

_____

_____

3. Με ποιό τρόπο ορίζουμε αρχική τιμή σε μία μεταβλητή;

_____

_____

4. Θα μπορούσατε να ορίσετε αρχική τιμή σε μία μεταβλητή short την τιμή 0,5;

_____

_____

5. Σε περίπτωση που δεν ορίσετε αρχική τιμή σε μία μεταβλητή, τι τιμή θα έχει αυτή η μεταβλητή;

_____

_____

6. Ποιές από τις παρακάτω λέξεις θα μπορούσαν να είναι ονόματα μεταβλητών;

A. profit

Β. long

Γ. Salary of Emploees

Δ. PlithosArithmon

Ε. athrismaArithmon

Ασκήσεις:

1. Προσπαθήστε να γράψετε το παρακάτω πρόγραμμα στο NetBeans με τον τρόπο που έχουμε εξηγήσει, δημιουργώντας μία νέα κλάση (FirstExercise) στο πακέτο που έχετε ήδη φτιάξει.

```
public class FirstExercise
 {
 public static void main(String [] args)
{
int x = 120;
 System.out.println("x is " + x);
short y,z,t;
 y=5;
z= 20;
t= (short) (y+z);
 System.out.println("t is " + t);
k = (x-t);
 System.out.println("k is " + k);
 }
}
```

2. Προσπαθήστε τώρα να δημιουργήσετε ένα πρόγραμμα που να υπολογίζει το εμβαδόν ενός ισόπλευρου τετραγώνου με πλευρά 5 εκατοστά. Την κλάση που θα δημιουργήσετε ονομάστε την EmvadonTetragonou ενώ τις short μεταβλητές για την πλευρά και το εμβαδόν του τετραγώνου ονομάστε τις a και e αντίστοιχα.

3. Με τον ίδιο τρόπο υπολογίστε και την περίμετρο ενός ορθογωνίου με πλευρές a= 5 και b=10. Ονομάστε την κλάση PerimetrosOrthogoniou και τη μεταβλητή της περιμέτρου με p.

4. Να γίνει πρόγραμμα StringUpper.java το οποίο θα έχει πίνακα με 10 String με Αγγλικές λέξεις. Το πρόγραμμα θα πρέπει να διατρέχει τον πίνακα και να μετατρέπει κάθε String στο array αυτό σε ένα String με κεφαλαία γράμματα. Να χρησιμοποιήσετε τη συνάρτηση toUpperCase της τάξης String.

5. Να γίνει πρόγραμμα SortArray.java το οποίο χρησιμοποιεί τον αλγόριθμο της ταξινόμησης με τη μέθοδο της φυσαλίδας για την αύξουσα ταξινόμηση ενός πίνακα 10 ακεραίων.

6.Να γίνει πρόγραμμα SeekNumber.java το οποίο χρησιμοποιεί τον αλγόριθμο της δυαδικής αναζήτησης για να αναζητήσει ένα ακέραιο που δίνεται σαν παράμετρος στο πρόγραμμα σε ένα πίνακα ακεραίων ο οποίος είναι ταξινομημένος.

7. Γράψτε ένα πρόγραμμα που θα υπολογίζει την ηλικία ενός ανθρώπου σε ημέρες, λαμβάνοντας την ηλικία του σε χρόνια, μήνες και ημέρες.Υποθέστε ότι κάθε μήνας έχει 30 μέρες.

```
Class Ex20
{
 puplic static void main(String[ ] args)
{
  int day=15    //ηλικία σε μέρες, μήνες, έτη
int month=5
int year=80
int age;
age=365*year+30*month+day;
  System.out.println("Είστε''+age+"ημερών");
  }
}
```

8. Μετατρέψτε το παραπάνω πρόγραμμα έτσι ώστε να δέχεται την ημερομηνία γέννησης ενός ανθρώπου και τη σημερινή ημερομηνία και στη συνέχεια να υπολογίζει την ηλικά του.

```
Class Ex21
{
  puplic static void main(String[ ] args)
{

   int bday=5;    // ημερομηνία γέννησης
   int bmonth=9;
   int byear=84;
   int day=29;   //σημερινή ημερομηνία
   int month=2;
   int year=08;

   int birthday, today, age;

   birthday=365*byear+30*bmonth+bday;
   today=365*year+30*month+day;
   age=today-birthday;


  System.out.println("Είστε''+age+"ημερών");

  }
}
```

Στο 5^{ο} κεφάλαιο θα μιλήσουμε για τους τελεστές. Αφού ορίσαμε τις μεταβλητές και είδαμε πως χρησιμοποιούνται, είναι καιρός να δούμε πως κάνουμε πράξεις με αριθμούς.

Για να κάνουμε όμως πράξεις πρέπει να δηλώσουμε τις σχέσεις ανάμεσα στις μεταβλητές. Τα πέντε είδη τελεστών που θα δούμε θα μας βοηθήσουν να ορίσουμε τις σχέσεις αυτές.

Πρώτα θα δούμε τους αριθμητικούς τελεστές, με τους οποίους θα ορίσουμε τις πράξεις πρόσθεσης, αφαίρεσης, πολλαπλασιασμού και διαίρεσης και τα σύμβολα που χρησιμοποιούνται για τν κάθε πράξη. Έπειτα θα συνεχίσουμε με τους τελεστές σύγκρισης, που θα ορίσουν τις σχέσεις ισότητας ή ανισότητας ανάμεσα στις μεταβλητές. Με τους λογικούς τελεστές θα ορίσουμε τη λογική επαλήθευση των τελεστών. Στη συνέχεια με τους δυαδικούς τελεστές θα ορίσουμε τις μεταβλητές σε επίπεδο bits. Τέλος με τους τελεστές καταχώρισης θα μάθουμε τη χρήση ορισμένων συμβόλων για να ορίζουμε τις πράξεις.

ΚΕΦΑΛΑΙΟ 5

1) ΤΕΛΕΣΤΕΣ

Οι τελεστές είναι σύμβολα τα οποία συμβολίζουν την τέλεση μιας λειτουργίας σε ένα, δύο ή και τρεις τελεστές (operators). Υπάρχουν δηλαδή τρία είδη τελεστών ως προς τον αριθμό των τελεστών στους οποίους επενεργούν:

Μοναδιαίοι τελεστές (Unary Operators): Αυτοί επενεργούν σε ένα και μόνο τελεστή. Για παράδειγμα ο τελεστής ++ αυξάνει κατά 1 τον τελεστή του όπως στην έκφραση ++count.

Δυαδικοί τελεστές (Binary Operators): Αυτοί επενεργούν σε δύο τελεστές. Για παράδειγμα ο τελεστής + της αριθμητικής πρόσθεσης απαιτεί δύο τελεστές όπως στην έκφραση op1 + op2.

Τριαδικοί τελεστές (Ternary Operators): Αυτοί επενεργούν σε τρεις τελεστές. Η Java έχει μόνο ένα τέτοιο τελεστή τη φραση if που συμβολίζεται με το ?, όπως στην ακόλουθη έκφραση a>4?a-4:4

2) ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ

Οι αριθμητικοί τελεστές χρησιμοποιούνται για τις αριθμητικές πράξεις και είναι αυτοί που φαίνονται στον ακόλουθο πίνακα.

Τελεστής	Χρήση	Περιγραφή
+	a+b	Το άθροισμα των a και b
-	a-b	Η διαφορά των a και b
*	a*b	Το γινόμενο των a και b
/	a / b	Το πηλίκο των a και b
%	a%b	Το υπόλοιπο της διαίρεσης του a δια του b.
++	a++ ή ++a	Αυξάνει τη μεταβλητή a κατά 1
--	a-- ή --a	Μειώνει τη μεταβλητή a κατά 1

Μερικές παρατηρήσεις για τους αριθμητικούς τελεστές είναι οι ακόλουθες:

Ο τελεστής + στη Java έχει επεκταθεί έτσι ώστε να έχει και το νόημα της συνένωσης για τα Strings. Έτσι για παράδειγμα οι ακόλουθες εντολές θα εμφανίσουν στην οθόνη

το μήνυμα των «Αρ. Χαρκτήρων: 10». Παρατηρείστε επίσης την σιωπηλή μετατροπή του count από int σε String για να συνενωθεί με τη σταθερά «Αρ. Χαρακτήρων»

```
int count = 20;
System.out.println("Αρ. Χαρακτήρων: "+count);
```

- Ο τελεστής ++ όπως και ο --, μπορούν να εφαρμοστούν είτε από τα αριστερά (προ-αύξηση & προ-μείωση) είτε από τα δεξιά (μετά-αύξηση & μετά-μείωση).

  Προ-αύξηση και προ-μείωση σημαίνει ότι αν η συγκεκριμένη έκφραση αποτελεί τμήμα μιας ευρύτερης έκφρασης πρώτα θα υπολογιστεί η νέα τιμή της μεταβλητής που αυξάνουμε ή μειώνουμε και μετά θα υπολογιστεί η έκφραση με τη χρήση της νέας τιμής, όπως στο παράδειγμα:

  ```
  int x=20;
  y = ++x; //Στο y θα εισαχθεί η τιμή 21, και φυσικά το x θα γίνει 21
  ```

  Μετά-αύξηση και μετά-μείωση σημαίνει ότι αν η συγκεκριμένη έκφραση αποτελεί τμήμα μιας ευρύτερης έκφρασης πρώτα θα υπολογιστεί η έκφραση με βάση τη τιμή εκείνη τη στιγμή που έχει η μεταβλητή που αυξάνουμε ή μειώνουμε και μετά θα αυξηθεί ή θα μειωθεί η μεταβλητή όπως στο παράδειγμα:

  ```
  int x=20;
  y = x++; //Στο y θα εισαχθεί η τιμή 20, και φυσικά το x θα γίνει 21
  ```

3) ΤΕΛΕΣΤΕΣ ΣΥΓΚΡΙΣΗΣ

Οι τελεστές σύγκρισης που δίνονται στη συνέχεια συγκρίνουν δύο τιμές και καθορίζουν τη σχέση μεταξύ τους. Η έκφραση που περιέχει ένας τελεστής σύγκρισης μπορεί να είναι true ή false αν επαληθεύεται ή όχι η έκφραση.

Τελεστής	Χρήση	Περιγραφή
>	a>b	true αν το a είναι μεγαλύτερο του b
>=	a>=b	true αν το a είναι μεγαλύτερο ή ίσο του b

<	a<b	true αν το a είναι μικρότερο του b
<=	a<=b	true αν το a είναι μικρότερο ή ίσο του b
==	a==b	true αν το a είναι ίσο με το b
!=	a!=b	true αν το a είναι διάφορο του b

4) ΛΟΓΙΚΟΙ ΤΕΛΕΣΤΕΣ

Τους λογικούς τελεστές συνήθως τους χρησιμοποιούμε με τους τελεστές σύγκρισης για την κατασκευή ποιο πολύπλοκων λογικών εκφράσεων, εκφράσεων δηλαδή που μπορεί να είναι true ή false. Για παράδειγμα η ακόλουθη έκφραση θα είναι true αν η τιμή του x είναι μεταξύ 10 και 100:
x>=10 && x<=100

Τελεστής	Χρήση	Περιγραφή
&&	x && y	true αν και η x και η y είναι true. Η y δεν αποτιμάται αν η x είναι false.
||	x || y	false αν και η x και η y είναι false. Η y δεν αποτιμάται αν η x είναι true
!	!x	Η λογική άρνηση της έκφρασης x
&	x&y	Το ίδιο με τον && μόνο που η y θα αποτιμηθεί ακόμα και αν η x είναι false
|	x|y	Το ίδιο με τον || μόνο που η y θα αποτιμηθεί ακόμα και αν η x είναι true.

Μερικές φορές είναι επιθυμητό να αποτιμηθεί και το δεύτερο μέρος μιας σύνθετης λογικής έκφρασης ακόμα και αν έχει υπολογιστεί η τιμή αλήθειας της έκφρασης. Για παράδειγμα φανταστείτε την ακόλουθη έκφραση:

x>0 && someFunc(x)>10

Αν θέλουμε σ' αυτό το παράδειγμα να κληθεί οπωσδήποτε η συνάρτηση someFunc τότε θα πρέπει να χρησιμοποιήσουμε τον & αντί του &&, γιατί με τον && αν το x <=0 δεν θα αποτιμηθεί το δεύτερο μέρος της έκφρασης μια και η τιμή αλήθειας της έκφρασης θα είναι ούτως ή άλλως false, αφού το πρώτο μέρος της έκφρασης θα είναι false.

Ανάλογη είναι και η χρήση του τελεστή | σε σχέση με τον ||. Για παράδειγμα στην έκφραση

x>0 | someFunc(x)>10
θα κληθεί η someFunc ακόμα και αν το x είναι θετικό

5) ΔΥΑΔΙΚΟΙ ΤΕΛΕΣΤΕΣ

Με τους τελεστές αυτούς μπορούμε και κάνουμε πράξεις σε επίπεδο bit.

Τελεστής	Χρήση	Περιγραφή
>>	a >> b	Κάνει ολίσθηση b bits δεξιά στο a
<<	a<<b	Κάνει ολίσθηση b bits αριστερά στο a
>>>	a >>> b	Το ίδιο με τον >> αλλά χωρίς πρόσημο
&	a&b	Δυαδικό και των a και b
|	a | b	Δυαδικό ή των a και b
^	a ^ b	Αποκλειστικό ‘η (XOR) των a και b
~	~a	Δυαδικό συμπλήρωμα του a

Οι τελεστές δυαδικών πράξεων χρησιμοποιούνται όταν θέλουμε να επέμβουμε στην τιμή μιας μεταβλητής σε επίπεδο bit.
Για παράδειγμα έστω ότι έχουμε δηλώσει τις ακόλουθες σταθερές για να δηλώσουμε τα χαρακτηριστικά ενός κινητού:

final int multicolor = 1;
final int bluetooth = 2;

Στη συνέχεια δηλώνουμε μία μεταβλητή int, έστω την int mobproperties = 0 και στη συνέχεια να αποδώσουμε σε αυτή τη μεταβλητή ταυτόχρονα δύο ή και τρεις ιδιότητες χρησιμοποιώντας τον τελεστή **|** όπως παρακάτω:

mobproperties = multicolor **|** bluetooth;

Σε άλλο σημείο του προγράμματος θα μπορούσαμε να ελέγξουμε με την χρήση του δυαδικού & αν ένα κινητό έχει bluetooth ως εξής:

if ((mobproperties & bluetooth) == bluetooth) …

*AND=και τα δύο, OR=το ένα ή το άλλο ή και τα δύο, XOR=το ένα ή το άλλο αποκλειστικά

6) ΤΕΛΕΣΤΕΣ ΚΑΤΑΧΩΡΙΣΗΣ

Ο τελεστής καταχώρισης τιμής σε μεταβλητή είναι ο =. Ο = αποδίδει την τιμή της έκφρασης στα δεξιά του στη μεταβλητή που βρίσκεται στα αριστερά του, όπως στην εντολή:

```
x = y+20;
```

όπου θα υπολογιστεί η έκφραση y+20 και στη συνέχεια θα καταχωρηθεί η τιμή αυτή στη μεταβλητή x.

Στη Java χρησιμοποιούνται και άλλοι τελεστές απόδοσης τιμής όπου είναι ουσιαστικά συντομεύσεις για απόδοση τιμής σε μεταβλητή με ταυτόχρονη αύξηση, μείωση κτλ αυτής της μεταβλητής.
Για παράδειγμα η έκφραση με τον τελεστή +=:

```
x += 20;
```

είναι ταυτόσημη με την έκφραση: x = x+20;

Αντίστοιχη σημασία έχουν και οι τελεστές: -= *= /= %= &= |= ^= <<= >>= >>>=

ΕΠΑΝΑΛΗΠΤΙΚΕΣ ΕΡΩΤΗΣΕΙΣ 5^{ΟΥ} ΚΕΦΑΛΑΙΟΥ

Ερωτήσεις:

1. Ποια είναι η χρησιμότητα των τελεστών;

_____

_____

_____

2. Γράψτε δίπλα από κάθε σύμβολο το γράμμα Μ αν ανήκει στους Μοναδιαίους ή το Δ αν ανήκει στους Δυαδικούς τελεστές.

Α. *
Β. %
Γ. ++
Δ. >=
Ε. ==
Στ. !
Ζ. |
Η. &

3. Γράψτε τις εξής εκφράσεις χρησιμοποιώντας τους κατάλληλους τελεστές.

- **Το πηλίκο των x και y είναι ίσο με 5**
- **Η y είναι ίση με τη x προσαυξημένη κατά 1**
- **Το γινόμενο των x και y είναι ίσο με το 0 αν το x ή το y είναι ίσο με 0**
- **Το άθροισμα των x και y είναι μεγαλύτερο του 0, αν το x ή y είναι διάφορο του μηδενός.**

_____

_____

_____

_____

4. Απαντήστε τα παρακάτω ερωτήματα αν ισχύουν τα εξής:

X== a+b

Y==(c*b)/2

K==X –Y/X+10*b

a==2*b, b>=0, c==b/2

α. Ο Κ είναι θετικός ή αρνητικός αριθμός;

β. Αν b<=0 απαντήστε στην α ερώτηση

γ. Μπορεί το a να πάρει αρνητική τιμή;

_____

_____

_____

5. Ποιόν τελεστή θα επιλέγατε για να δηλώσετε την αποκλειστική επιλογή του x ή του y;

_____

_____

6. Έστω ότι έχετε τα εξής δεδομένα: y=x+1 όπου x>0, y>0, x&&y. Αν στο x δοθεί η τιμή -1 ποιο θα είναι το αποτέλεσμα;

_____

_____

_____

_____

Σε αυτό το κεφάλαιο θα δούμε τους πίνακες στη Java. Πως δηλώνουμε ένα πίνακα στη Java και πως τον κατασκευάζουμε. Καθώς και τις μεθόδους και τις μεταβλητές που χρησιμοποιούμε στους πίνακες.

Ακόμα θα δούμε τις λίστες. Πως δημιουργούνται, ποιος είναι ο σκοπός και η χρησιμότητα τους. Με τη χρήση παραδειγμάτων θα δούμε την κατασκευή πινάκων και λιστών βήμα βήμα.

ΚΕΦΑΛΑΙΟ 6

1) Πίνακες

Στα σημαντικά προβλήματα που αφορούν τους υπολογιστές συχνά χρειάζεται να αποθηκεύουμε λίστες αντικειμένων. Συχνά αυτά τα αντικείμενα καθορίζονται σειριακά και αναφέρονται στη θέση τους στη λίστα. Πολλές φορές η σειρά αυτή είναι φυσική, όπως η σειρά των δέκα πρώτων ανθρώπων που φτάνουν στην τράπεζα. Το πρώτο άτομο θα είναι το αντικείμενο 1 στη λίστα, το δεύτερο άτομο που έφτασε θα είναι το αντικείμενο 2 στη λίστα κτλ. Σε άλλες περιπτώσεις η σειρά αυτή δεν έχει καμία σημασία, όπως στο RAM configuration problem όπου το να έχουμε 4 MB SIMM στη slot A και 8 MB SIMM στη slot B είναι ακριβώς το ίδιο με το να έχουμε 8 MB SIMM στη slot A και 4 MB SIMM στην slot B.

Υπάρχουν πολλοί τρόποι για να αποθηκεύσουμε λίστες, όπως συνδεδεμένες λίστες, σύνολα, δέντρα και πίνακες. Ποιο θα προτιμήσετε εξαρτάται από τις απαιτήσεις της εφαρμογής σας και τη φύση των δεδομένων σας. Η Java παρέχει κλάσεις για πολλούς απ' αυτούς τους τρόπους αποθήκευσης δεδομένων.

Οι πίνακες (Arrays) είναι πιθανότατα ο αρχαιότερος και πιο αποτελεσματικός τρόπος αποθήκευσης συνόλων μεταβλητών. Ένας πίνακας είναι ένα σύνολο μεταβλητών που μοιράζονται το ίδιο όνομα και συντάσσονται με τη σειρά από το 0 μέχρι το πλήθος των μεταβλητών μειωμένο κατά 1. Ο αριθμός των μεταβλητών που μπορεί να αποθηκευτεί ονομάζεται διάσταση του πίνακα. Κάθε μεταβλητή ονομάζεται στοιχείο του πίνακα.

Ένας πίνακας μπορεί να παρασταθεί σαν στήλη δεδομένων όπως πιο κάτω:

I[0]	4
I[1]	2
I[2]	76
I[3]	-90
I[4]	6

Σ’ αυτή την περίπτωση βλέπουμε έναν ακέραιο πίνακα που ονομάζεται I με 5 στοιχεία. Ο τύπος του πίνακα είναι int και έχει διάσταση 5.

2) ΔΗΜΙΟΥΡΓΙΑ ΠΙΝΑΚΩΝ

Υπάρχουν τρία στάδια για τη δημιουργία πινάκων:

1. Να δηλώσουμε τον πίνακα
2. Να ορίσουμε τον πίνακα
3. Να αρχικοποιήσουμε τον πίνακα

Δήλωση Πινάκων. Όπως και οι άλλες μεταβλητές στη Java, ένας πίνακας πρέπει να έχει ένα συγκεκριμένο τύπο όπως byte, int, string ή double. Μόνο μεταβλητές του αντίστοιχου τύπου μπορούν να αποθηκευτούν σε έναν πίνακα. Για παράδειγμα δεν μπορούμε να έχουμε έναν πίνακα που να αποθηκεύει και ints και strings.
Όπως και οι άλλες μεταβλητές στη Java, έτσι και οι πίνακες πρέπει να δηλώνονται. Όταν δηλώνουμε έναν πίνακα βάζουμε στον τύπο και το [] που δείχνει ότι η μεταβλητή είναι ένας πίνακας. Ακολουθούν μερικά παραδείγματα:

```
int[] k;
float[] yt;
String[] names;
```

Με άλλα λόγια δηλώνουμε έναν πίνακα όπως δηλώνουμε οποιαδήποτε άλλη μεταβλητή μόνο που βάζουμε και αγκύλες στο τέλος του τύπου της μεταβλητής.

Ορισμός Πινάκων. Για να δημιουργήσουμε έναν πίνακα χρησιμοποιούμε τον τελεστή new. Πρέπει να πούμε στον compiler πόσα στοιχεία θα αποθηκευτούν στον πίνακα. Πιο κάτω βλέπουμε πώς δημιουργούμε τις μεταβλητές:

```
k = new int[3];
yt = new float[7];
names = new String[50];
```

Τα νούμερα στις αγκύλες καθορίζουν τη διάσταση του πίνακα, δηλαδή πόσες θέσεις έχει. Ο k μπορεί να κρατήσει τρία ints, ο yt μπορεί να κρατήσει επτά floats και ο names μπορεί να κρατήσει πενήντα strings. Συνήθως αυτό ονομάζεται ορισμός του πίνακα αφού καθορίζει το τμήμα της RAM που χρειάζεται ο πίνακας.

Εδώ θα δούμε για πρώτη φορά και τον τελεστή new. Ο new είναι δεσμευμένη λέξη στη Java και χρησιμοποιείται όχι μόνο για να ορίσει πίνακες αλλά και άλλα είδη αντικειμένων.

Αρχικοποίηση Πινάκων. Κάθε στοιχείο του πίνακα προσδιορίζεται από το όνομα του πίνακα και από έναν ακέραιο που φανερώνει τη θέση του στον πίνακα. Οι αριθμοί που χρησιμοποιούνται ονομάζονται δείκτες του πίνακα. Οι δείκτες είναι συνεχόμενοι αριθμοί που ξεκινούν από το 0. Γι' αυτό και ο πίνακας k έχει τα στοιχεία k[0], k[1] και k[2]. Εφόσον ξεκινάμε από το 0 δεν υπάρχει k[3] και αν προσπαθήσουμε να έχουμε πρόσβαση σ' αυτό θα δημιουργήσουμε ένα ArrayIndexOutOfBoundsException.

Παρακάτω θα δούμε πώς αποθηκεύουμε τιμές στους πίνακες:

```
k[0] = 2;
k[1] = 5;
k[2] = -2;
yt[6] = 7.5f;
names[4] = "Fred";
```

Αυτό το βήμα ονομάζεται αρχικοποίηση του πίνακα ή καλύτερα αρχικοποίηση των στοιχείων του πίνακα.

Ακόμα και για τους πίνακες μεσαίου μεγέθους είναι σπάνιο να αρχικοποιούμε κάθε στοιχείο ξεχωριστά. Συχνά είναι πιο εύκολο να χρησιμοποιούμε τον βρόγχο for. Για παράδειγμα ακολουθεί ένα loop που γεμίζει έναν πίνακα με τα τετράγωνα των αριθμών από το 0 ως το 100.

```
float[] squares = new float[101];
for (int i=0, i <= 100; i++) {
squares[i] = i*i;
}
```

Θα πρέπει να προσέξετε δύο σημεία του κώδικα:

- Εφόσον οι δείκτες ξεκινούν από το 0, χρειαζόμαστε 101 στοιχεία αν θέλουμε να συμπεριλάβουμε και το τετράγωνο του 100.
- Παρόλο που το i είναι int, γίνεται float όταν αποθηκεύει τετράγωνο αφού δηλώσαμε τα τετράγωνα σαν πίνακα float.

Ο τρόπος για να αποφύγουμε το πρώτο σημείο είναι το παράδειγμα που ακολουθεί:

```
float[] squares = new float[101];
for (int i=0, i < squares.length; i++) {
squares[i] = i*i;
```

}

Σημειώστε ότι το <= γίνεται <.

Συντομεύσεις (Shortcuts)

Πιθανόν να φαίνεται δύσκολη η δημιουργία ενός πίνακα, ειδικά σε κάποιον που έχει μάθει να χειρίζεται γλώσσες όπως η Fortran. Ευτυχώς η Java παρέχει πολλές ευκολίες για τη δήλωση, τον ορισμό και την αρχικοποίηση πινάκων.

Μπορούμε να δηλώσουμε και να ορίσουμε έναν πίνακα την ίδια στιγμή:

```
int[] k = new int[3];
float[] yt = new float[7];
String[] names = new String[50];
```

Μπορούμε ακόμα να δηλώσουμε, να ορίσουμε και να αρχικοποιήσουμε έναν πίνακα την ίδια στιγμή δημιουργώντας μια λίστα με τις αρχικές τιμές μέσα σε αγκύλες όπως πιο κάτω:

```
int[] k = {1, 2, 3};
float[] yt = {0.0f, 1.2f, 3.4f, -9.87f, 65.4f, 0.0f, 567.9f};
```

3) ΜΕΤΡΗΣΗ ΨΗΦΙΩΝ (COUNTING DIGITS)

Οι κύριες μέθοδοι μιας εφαρμογής αποθηκεύουν τα command line arguments σ'έναν πίνακα από αλφαριθμητικά που ονομάζεται args.

Ας θεωρήσουμε μια κλάση που μετρά πόσες φορές επαναλαμβάνονται τα ψηφία 0-9 στο δεκαδικό μέρος του αριθμού pi για παράδειγμα. Τα ψηφία θα πρέπει να επαναλαμβάνονται με την ίδια συχνότητα μετά από ένα μεγάλο χρονικό διάστημα.

Αυτό θα το δούμε δημιουργώντας έναν πίνακα από 10 longs που ονομάζεται ndigit. Το μηδενικό στοιχείο του ndigit θα ανιχνεύει τον αριθμό των μηδενικών, το πρώτο στοιχείο του ndigit θα ανιχνεύει τον αριθμό των άσσων και ούτω καθεξής. Ελέγχουμε τον random number generator της Java για να δούμε αν παράγει τυχαίους αριθμούς.

```
import java.util.*;
class RandomTest {
  public static void main (String args[]) {
    int[] ndigits = new int[10];
    double x;
    int n;
```

```
    Random myRandom = new Random();
      // Initialize the array
    for (int i = 0; i < 10; i++) {
      ndigits[i] = 0;
    }

    // Test the random number generator a whole lot
    for (long i=0; i < 100000; i++) {
      // generate a new random number between 0 and 9
      x = myRandom.nextDouble() * 10.0;
      n = (int) x;
      //count the digits in the random number
      ndigits[n]++;
    }

    // Print the results
    for (int i = 0; i <= 9; i++) {
      System.out.println(i+": " + ndigits[i]);
    }
  }
}
```

Έχουμε τρεις βρόγχους for στον παρακάτω κώδικα, έναν για να αρχικοποιήσουμε τον πίνακα, έναν για να πραγματοποιήσουμε τον επιθυμητό υπολογισμό και έναν τρίτο για να τυπώνει τα αποτελέσματα. Αυτό είναι αρκετά συνηθισμένο σε κώδικες που χρησιμοποιούν πίνακες.

4) ΔΙΣΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ (TWO DIMENSIONAL ARRAYS)

Ο πιο συνηθισμένος τύπος πολυδιάστατου πίνακα είναι ο δισδιάστατος. Ένας δισδιάστατος πίνακας σαν πίνακας τιμών είναι όπως ο παρακάτω:

c0	c1	c2	c3	
r0	0	1	2	3

r1	1	2	3	4
r2	2	3	4	5
r3	3	4	5	6
r4	4	5	6	7

Εδώ έχουμε έναν πίνακα με 5 γραμμές και 4 στήλες. Έχει συνολικά 20 στοιχεία. Έχουμε πει ότι έχει διαστάσεις 5x4 , όχι 20. Αυτός ο πίνακας δεν είναι ίδιος με τον 4x5 πίνακα που ακολουθεί πιο κάτω:

c0	c1	c2	c3	c4
r0	0	1	2	3
r1	1	2	3	4
r2	2	3	4	5
r3	3	4	5	6

Πρέπει να χρησιμοποιούμε 2 αριθμούς για να αναγνωρίσουμε τη θέση σε ένα δισδιάστατο πίνακα. Αυτές είναι οι θέσεις των γραμμών και των στηλών που βρίσκονται τα στοιχεία. Για παράδειγμα αν ο παραπάνω πίνακας λέγεται J τότε J[0][0] είναι 0, J[0][1] είναι 1, J[0][2] είναι 2, J[0][3] είναι 3, J[1][0] είναι 1 κτλ.
Εδώ βλέπουμε πώς αναφέρονται τα στοιχεία ενός 4x5 πίνακα που ονομάζεται M.

M[0][0]				
M[0][1]	M[0][2]	M[0][3]	M[0][4]	
M[1][0]	M[1][1]	M[1][2]	M[1][3]	M[1][4]
M[2][0]	M[2][1]	M[2][2]	M[2][3]	M[2][4]
M[3][0]	M[3][1]	M[3][2]	M[3][3]	M[3][4]

Οι δισδιάστατοι πίνακες δηλώνονται, ορίζονται και αρχικοποιούνται όπως και οι μονοδιάστατοι πίνακες. Γι' αυτό πρέπει να καθορίζουμε δύο διαστάσεις αντί για μία και γι' αυτό χρησιμοποιούμε δύο φωλιασμένα for για να γεμίσουμε τον πίνακα.
Στα παραπάνω παραδείγματα οι πίνακες γεμίζονται με το άθροισμα των δεικτών των γραμμών και των στηλών. Βλέπετε τον κώδικα που δημιουργεί και γεμίζει έναν τέτοιο πίνακα:

```
class FillArray {
  public static void main (String args[]) {
      int[][] M;
    M = new int[4][5];
      for (int row=0; row < 4; row++) {
      for (int col=0; col < 5; col++) {
        M[row][col] = row+col;
      }
    }
    }
  }
```

Ο αλγόριθμος που θα χρησιμοποιήσετε για να γεμίσετε έναν πίνακα εξαρτάται από τη χρήση για την οποία θέλετε τον πίνακα. Ακολουθεί ένα πρόγραμμα που υπολογίζει την ταυτοτική μήτρα για μια δεδομένη διάσταση.

```
class IDMatrix {
  public static void main (String args[]) {
      double[][] ID;
    ID = new double[4][4];
      for (int row=0; row < 4; row++) {
      for (int col=0; col < 4; col++) {
        if (row != col) {
          ID[row][col]=0.0;
        }
        else {
          ID[row][col] = 1.0;
        }
      }
    }
```

```
  }
 }
```

Στους δισδιάστατους πίνακες το λάθος ArrayIndexOutOfBounds συμβαίνει όταν υπερβαίνουμε το μέγιστο δείκτη γραμμής ή στήλης.

5) ΠΟΛΥΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ (MULTIDIMENSIONAL ARRAYS)

Η Java μας επιτρέπει να έχουμε πίνακες τριών, τεσσάρων ή και περισσότερων διαστάσεων. Είναι πολύ πιθανό αν χρειάζεστε έναν πίνακα με περισσότερες από 3 διαστάσεις να χρησιμοποιήσετε λανθασμένη δομή δεδομένων. Ακόμα και οι τρισδιάστατοι πίνακες είναι σπάνιοι έξω από τις επιστημονικές και μηχανικές εφαρμογές.

Η σύνταξη για τους τρισδιάστατους πίνακες είναι μια επέκταση αυτής που χρησιμοποιήσαμε στους δισδιάστατους. Ακολουθεί ένα πρόγραμμα που δηλώνει, ορίζει και αρχικοποιεί έναν τρισδιάστατο πίνακα:

```
class Fill3DArray {
  public static void main (String args[]) {
      int[][][] M;
    M = new int[4][5][3];
      for (int row=0; row < 4; row++) {
      for (int col=0; col < 5; col++) {
        for (int ver=0; ver < 3; ver++) {
          M[row][col][ver] = row+col+ver;
        }
      }
    }
  }
}
```

Χρειαζόμαστε τρία φωλιασμένα for για να χειριστούμε την επιπλέον διάσταση. Η σύνταξη για ακόμα μεγαλύτερες διαστάσεις είναι παρόμοια. Απλά προσθέτουμε άλλο ένα ζευγάρι αγκίστρων και μια ακόμα διάσταση.

6) ΜΗ ΙΣΟΖΥΓΙΣΜΕΝΟΙ ΠΙΝΑΚΕΣ (UNBALANCED ARRAYS)

Όπως και στη C, έτσι και στη Java δεν έχουμε πραγματικούς πολυδιάστατους πίνακες. Στη θέση τους η Java χρησιμοποιεί πίνακες πινάκων. Αυτό σημαίνει ότι είναι

πολύ πιθανό να έχουμε μη ισοζυγισμένους πίνακες. Ένας μη ισοζυγισμένος πίνακας είναι ένας πολυδιάστατος πίνακας που οι διαστάσεις δεν είναι ίδιες για όλες τις γραμμές. Στις περισσότερες εφαρμογές αυτό πρέπει να το αποφεύγουμε.

7) ΠΙΝΑΚΕΣ ΔΕΔΟΜΕΝΩΝ

Οι πίνακες αποτελούν μια από τις πιο χρησιμοποιούμενες δομές δεδομένων στον προγραμματισμό. Η υλοποίηση τους στη Java, είναι επιφανειακά , παρόμοια με αυτή της C στη σύνταξη. Υπάρχουν όμως και αρκετές λειτουργικές διαφορές μεγάλης σημασίας. Η σημαντικότερη διαφορά είναι ότι η Java απαιτεί οι πίνακες να έχουν ένα συγκεκριμένο μήκος, το οποίο δεν μπορεί ποτέ να παραβιαστεί. Η έκφραση arr[i] σημαίνει ξεκίνησε από τη βάση του πίνακα arr και μετακινήσου i μονάδες προς τα εμπρός . Αν ο πίνακας αποτελούνταν από 10 μονάδες και το i ήταν ίσο με 100, τότε θα μπορούσατε να προσπελάσετε απαγορευμένες περιοχές στη μνήμη. Η Java παρακολουθεί τα όρια του πίνακα κατά την εκτέλεση και ειδοποιεί για σφάλμα, αν βρει κάποιο τέτοιο πρόβλημα.

Παραδείγματα:

```
int a[]=new int[10];
String b[]=new string[10];
Foo[]c=new foo[10];
Bar d[];
```

Ο κώδικας αυτός δημιουργεί 3 καινούριους πίνακες και τους δίνει ένα αρχικό μήκος για την αποθήκευση δεδομένων. Στο τέταρτο παράδειγμα ,δεν έχει ακόμα καθοριστεί κάποιο μήκος για τον πίνακα d. Οι αριθμοί 0-9 μπορούν να χρησιμοποιηθούν για τη πρόσβαση στα διάφορα τμήματα των πινάκων a, b και c. Ο πρώτος πίνακας με όνομα a, περιέχει ακέραιους. Οι b και c, είναι πίνακες δεικτών αντικειμένων. Τα άγκιστρα για τη δήλωση του πίνακα μπορούν να τοποθετηθούν σε δύο διαφορετικές θέσεις. Η πρώτη θέση ,η οποία χρησιμοποιείται στη δήλωση των πινάκων a και b είναι οικεία στους προγραμματιστές της C. Η δεύτερη θέση έχει σκοπό τη μεγαλύτερη εναρμόνιση των δηλώσεων αυτών με τον τρόπο με τον οποίο καθορίζονται τα αντικείμενα στη Java.

Οι πίνακες 2 διαστάσεων δημιουργούνται με τη μορφή πινάκων που περιέχουν μονοδιάστατους πίνακες. Οι πίνακες αυτοί μπορούν να δημιουργηθούν με κώδικα, όπως ο παρακάτω:

```
Int ticTacToeBoard[] []=new int[3] [3];
Int AgeWeight[] []=new int[100] [];
Int AgeWeight[0] []=new int[3];
Int AgeWeight[1] []=new int[5];
//…
```

Οι πίνακες στοιχείων (arrays) μας επιτρέπουν να αποθηκεύουμε και να χειριζόμαστε μαζική πληροφορία. Μπορούμε να τους χρησιμοποιήσουμε όταν έχουμε πολλά δεδομένα ιδίου τύπου, π.χ. καταλόγους ονομάτων, κ.α. Οι βασικές αρχές που διέπουν τους πίνακες στοιχείων είναι απλές. Αντί να έχουμε πολλές μεταβλητές ιδίου τύπου με διαφορετικό όνομα η κάθε μια, έχουμε μόνο ένα όνομα που αναφέρεται σε ολόκληρο το σύνολο και στο οποίο κάθε ξεχωριστό στοιχείο χαρακτηρίζεται από ένα δείκτη (δηλαδή την θέση του μέσα στο σύνολο). Ο δείκτης αυτός γράφεται μέσα σε αγκύλες [] μετά το όνομα του πίνακα στοιχείων. Καθώς ο δείκτης μπορεί να είναι μια μεταβλητή, υπάρχει ένας απλός τρόπος για να προσπελάσουμε οποιοδήποτε ή όλα τα στοιχεία του πίνακα.

Ένα παράδειγμα για την χρήση των πινάκων στοιχείων είναι το εξής: Υποθέτουμε ότι έχουμε έναν πίνακα στοιχείων με 1000 αριθμούς με το όνομα num και μια μεταβλητή με το όνομα count. Αν μπορούσαμε να δώσουμε στην μεταβλητή count όλες τις τιμές από το 1 έως το 1000 και μπορούσαμε να επαναλαμβάνουμε την γραμμή

System.out.println(num[count]);

τότε θα εμφανιζόντουσαν όλοι οι αριθμοί του πίνακα στοιχείων ο ένας κάτω από τον άλλον.

8) ΑΝΑΖΗΤΗΣΗ (SEARCHING)

Είναι πολύ συνηθισμένο να ψάχνουμε σ’ έναν πίνακα μια συγκεκριμένη τιμή. Μερικές φορές αυτή η τιμή είναι γνωστή εκ των προτέρων. Άλλες φορές αναζητάμε το μικρότερο ή το μεγαλύτερο στοιχείο.

Εκτός από την περίπτωση που ξέρουμε κάτι για το περιεχόμενο ενός πίνακα, σε όλες τις άλλες περιπτώσεις ο πιο γρήγορος αλγόριθμος αναζήτησης ενός πίνακα είναι η γραμμική αναζήτηση. Χρησιμοποιούμε έναν βρόγχο for και βλέπουμε κάθε στοιχείο του πίνακα μέχρι να βρούμε το στοιχείο που θέλουμε. Ακολουθεί μια απλή μέθοδος που τυπώνει το μεγαλύτερο και το μικρότερο στοιχείο ενός πίνακα.

```
  static void printLargestAndSmallestElements (int[] n) {
    int max = n[0];
```

```
    int min = n[0];

    for (int i=1; i < n.length; i++) {
      if (max < n[i]) {
        max = n[i];
      }
      if (min > n[i]) {
        min = n[i];
      }
    }
    System.out.println("Maximum: " + max);
    System.out.println("Minimum: " + min);
       return;
    }
```

Αν πρόκειται να ψάξετε έναν πίνακα πολλές φορές, πιθανόν να θελήσετε να τον ταξινομήσετε πρώτα. Θα δούμε τους αλγόριθμους ταξινόμησης στο επόμενο κεφάλαιο.

9) ΤΑΞΙΝΟΜΗΣΗ (SORTING)

Όλοι οι αλγόριθμοι ταξινόμησης βασίζονται σε δύο θεμελιώδεις λειτουργίες: τη σύγκριση και τη μετακίνηση (swapping). Η σύγκριση είναι απλή. Το swapping είναι λίγο πιο πολύπλοκο. Θεωρείστε το ακόλουθο πρόβλημα. Θέλουμε να μεταφέρουμε την τιμή από το a στο b. Οι πιο πολλοί θα πρότειναν μια λύση σαν την ακόλουθη:

```
class Swap1 {
  public static void main(String args[]) {
    int a = 1;
    int b = 2;
    System.out.println("a = "+a);
    System.out.println("b = "+b);
      // swap a and b
    a = b;
    b = a;
    System.out.println("a = "+a);
    System.out.println("b = "+b);
```

```
  }
}
```

Αυτή θα δημιουργούσε την παρακάτω έξοδο:

```
a = 1
b = 2
a = 2
b = 2
```

Δεν περιμέναμε όμως αυτό! Το πρόβλημα είναι ότι χάσαμε την τιμή 1 καθώς βάλαμε την τιμή του b στο a. Για να διορθωθεί αυτό πρέπει να εισάγουμε μια τρίτη μεταβλητή, την temp, η οποία θα κρατάει την αρχική τιμή του a.

```
class Swap2 {

  public static void main(String args[]) {
   int a = 1;
   int b = 2;
   int temp;
   System.out.println("a = "+a);
   System.out.println("b = "+b);

   // swap a and b
      temp = a;
   a = b;
   b = temp;
   System.out.println("a = "+a);
   System.out.println("b = "+b);
  }
}
```

Αυτός ο κώδικας παράγει το αποτέλεσμα που περιμέναμε.

```
a = 1
b = 2
a = 2
b = 1
```

Bubble Sort

Τώρα που είδαμε πώς να μετακινούμε τις τιμές 2 μεταβλητών, ας προχωρήσουμε στην ταξινόμηση. Υπάρχουν πολλοί διαφορετικοί αλγόριθμοι ταξινόμησης. Ένας από τους πιο απλούς και δημοφιλής αλγόριθμους είναι ο bubble sort. Η έννοια του bubble sort είναι να ξεκινήσουμε από την κορυφή του πίνακα. Συγκρίνουμε κάθε στοιχείο με το επόμενο στοιχείο. Αν είναι μεγαλύτερο από αυτό τότε τα μεταθέτουμε. Αυτή τη διαδικασία την κάνουμε όσες φορές χρειαστεί. Η μικρότερη τιμή εμφανίζεται στην κορυφή του πίνακα ενώ η μεγαλύτερη στο τέλος. Ακολουθεί ο κώδικας:

```
import java.util.*;
class BubbleSort {
  public static void main(String args[]) {
     int[] n;
   n = new int[10];
   Random myRand = new Random();
     // initialize the array
   for (int i = 0; i < 10; i++) {
     n[i] = myRand.nextInt();
   }

   // print the array's initial order
   System.out.println("Before sorting:");
   for (int i = 0; i < 10; i++) {
     System.out.println("n["+i+"] = " + n[i]);
               }

   boolean sorted = false;
   // sort the array
   while (!sorted) {
     sorted = true;
     for (int i=0; i < 9; i++) {
       if (n[i] > n[i+1]) {
         int temp = n[i];
         n[i] = n[i+1];
         n[i+1] = temp;
         sorted = false;
```

```
      }
    }
  }

  // print the sorted array
  System.out.println();
  System.out.println("After sorting:");
   for (int i = 0; i < 10; i++) {
     System.out.println("n["+i+"] = " + n[i]);
   }
  }
}
```

Σ’ αυτήν την περίπτωση ταξινομήσαμε έναν πίνακα με αύξουσα σειρά. Το μικρότερο στοιχείο μπαίνει πρώτο. Θα ήταν εύκολο να το αλλάξουμε σε ταξινόμηση με φθίνουσα σειρά.

Τρόποι χειρισμού Δυναμικών Πινάκων

```
//χρήση της ArrayList
import java.util.ArrayList;
class testArrayList {
public static void main ( String [ ] args ) {
ArrayList list1 = new ArrayList () ;
// προσθήκη στοιχείων στη λίστα
for ( int i = 5; i > 0; i-- ) {
list1.add ( 0, "List item " + i ) ;
}
// εκτύπωση λίστας
System.out.println ( "list1 size: " + list1.size ( ) ) ;
for ( int i = 0; i < list1.size ( ) ; i++ ) {
System.out.println ( list1.get ( i ) ) ;
}
}
}
```

10) Λίστες

Μια λίστα είναι μια κυλιόμενη λίστα από συμβολοσειρές που ορίζονται στο java.awt.List. Θα δημιουργήσουμε μια καινούργια λίστα, ακριβώς όπως δημιουργούμε ένα αντικείμενο. Ο συγκεκριμένος κατασκευαστής constructor) που χρησιμοποιούμε, ψάχνει για έναν int (έτσι είναι το όνομα των ορατών γραμμών) και έναν boolean, το οποίο λέει αν επιτρέπονται ή όχι πολλαπλές επιλογές. Θα ζητήσουμε 25 γραμμές και καθόλου πολλαπλές επιλογές.

```
List theList;
theList = new List(25, false);
```

Θα προσθέσουμε strings στην λίστα χρησιμοποιώντας την μέθοδο addItem από το List.

```
theList.addItem("This is a list item");
```

Τέλος χρειάζεται να προσθέσουμε αυτήν την λίστα στο applet μας(πιο συγκεκριμένα στο applet container).Αυτό το κάνουμε με τη γραμμή: add(theList); στην init μέθοδο. Αυτό είναι όλο. Μπορούμε να χρησιμοποιήσουμε το ίδιο applet με αυτό που χρησιμοποιήσαμε πριν, με αυτές τις μικρές αλλαγές.

```
import java.applet.Applet;
import java.awt.*;

public class EventList extends Applet {
  List theList;
  public void init() {
    theList = new List(25, false);
    add(theList);
    theList.addItem("init event");
  }

  public void paint(Graphics g) {
    theList.addItem("paint event");
  }

  public void start() {
    theList.addItem("start event");
```

```
}

public void destroy() {
  theList.addItem("destroy event");
}

public void update(Graphics g) {
  theList.addItem("update event");
}

public boolean mouseUp(Event e, int x, int y) {
  theList.addItem("mouseUp event");
  return false;
}

public boolean mouseDown(Event e, int x, int y) {
 theList.addItem("mouseDown");
 return false;
}

public boolean mouseDrag(Event e, int x, int y) {
  theList.addItem("mouseDrag event");
      return false;
}
  public boolean mouseMove(Event e, int x, int y) {
  theList.addItem("mouseMove event");
  return false;
}

public boolean mouseEnter(Event e, int x, int y) {
  theList.addItem("mouseEnter event");
  return false;
}
```

```
 public boolean mouseExit(Event e, int x, int y) {
    theList.addItem("mouseExit event");
    return false;
  }

  public void getFocus() {
    theList.addItem("getFocus event");
  }

  public void gotFocus() {
    theList.addItem("gotFocus event");
  }
    public void lostFocus() {
    theList.addItem("lostFocus event");
  }

  public boolean keyDown(Event e, int x) {
    theList.addItem("keyDown event");
    return true;
  }
}
```

11) ΠΩΣ ΝΑ ΧΡΗΣΙΜΟΠΟΙΕΙΤΕ ΤΙΣ ΛΙΣΤΕΣ

```
import java.util.*;
public class UseArrayList {
public static void main ( String [ ] args ) {
//How to Use ArrayList
//Begin
ArrayList arraylist = new ArrayList ( ) ;
String
arr [ ] = { "india","","three",null,"new Integer ( 1 ) " } ;
//Different ways to Add Elements to the arraylist
for ( int i=0;i < arr.length;i++ )
{
```

```
arraylist.add ( i,arr [ i ] ) ;
}
arraylist.add ( new Float ( 3.5 ) ) ;
arraylist.add ( new StringBuffer ( "IN BUFFER" ) ) ;
arraylist.add ( null ) ;
arraylist.add ( new Float ( 3.5 ) ) ;
String aa="Software Engineer";
arraylist.add ( 1,aa ) ;
//To see the Elements of the arraylist
System.out.println ( "The elements of arraylist are " ) ;
for ( int i=0;i < arraylist.size ( ) ;i++ )
{
System.out.println ( "
"+ ( i+1 ) +" ) "+arraylist.get ( i ) ) ;
}
System.out.println ( "The size of arraylist
="+arraylist.size ( ) ) ;
System.out.println ( "The arraylist is Empty?
="+arraylist.isEmpty ( ) ) ;
//Set the element at position one
arraylist.set ( 1,"J2EE Programmer" ) ;
//Converts arraylist to array
System.out.println ( "The elements of arraylist To array are
" ) ;
Object list2array [ ] = arraylist.toArray ( ) ;
for ( int i=0;i < list2array.length;i++ )
{
System.out.println ( "
"+ ( i+1 ) +" ) "+list2array [ i ] ) ;
}
//Checks whether this object is present in the arraylist or not
System.out.println ( "Contains this \"one\" value
="+arraylist.contains ( "one" ) ) ;
System.out.println ( "Contains this \"null\" value
```

```
="+arraylist.contains ( null ) ) ;
//Return the Index of the object
//arraylist is zero based, so it will give position 4
System.out.println ( "IndexOf \"null\" value
="+arraylist.indexOf ( null ) ) ;
//Gives the Last Occurance of the Object
System.out.println ( "LastIndexOf \"new Float ( 3.5 ) \" value
="+arraylist.lastIndexOf ( new Float ( 3.5 ) ) ) ;
//Remove the elements from the arraylist
arraylist.remove ( 2 ) ;
System.out.println ( "The size of arraylist
="+arraylist.size ( ) ) ;
//arraylist.removeRange ( ) ; See removeRange method in the
same site
arraylist.clear ( ) ;
System.out.println ( "The size of arraylist
="+arraylist.size ( ) ) ;
//End of ArrayList
}
}
```

ΠΑΡΑΔΕΙΓΜΑ ΛΙΣΤΑΣ ΜΕ ΑΝΤΙΚΕΙΜΕΝΑ

```
/**
* The ArrayList implements a growable array.
* Insertions are always done at the end.
*/
public class ArrayList<AnyType> extends AbstractCollection<AnyType>
implements List<AnyType>
{
/**
* Construct an empty ArrayList.
*/
public ArrayList( )
{
clear( );
```

```
}
/**
* Construct an ArrayList with same items as another Collection.
*/
public ArrayList( Collection<? extends AnyType> other )
{
clear( );
for( AnyType obj : other )
add( obj );
}
/**
* Returns the number of items in this collection.
* @return the number of items in this collection.
*/
public int size( )
{
return theSize;
}
/**
* Returns the item at position idx.
* @param idx the index to search in.
* @throws ArrayIndexOutOfBoundsException if index is out of
range.
*/
public AnyType get( int idx )
{
if( idx < 0 || idx >= size( ) )
throw new ArrayIndexOutOfBoundsException( "Index " + idx
+ "; size " + size( ) );
return theItems[ idx ];
}
/**
* Changes the item at position idx.
* @param idx the index to change.
```

```
* @param newVal the new value.
* @return the old value.
* @throws ArrayIndexOutOfBoundsException if index is out of
range.
*/
public AnyType set( int idx, AnyType newVal )
{
if( idx < 0 || idx >= size( ) )
throw new ArrayIndexOutOfBoundsException( "Index " + idx
+ "; size " + size( ) );
AnyType old = theItems[ idx ];
theItems[ idx ] = newVal;
return old;
}
/**
* Tests if some item is in this collection.
* @param x any object.
* @return true if this collection contains an item equal to x.
*/
public boolean contains( Object x )
{
return findPos( x ) != NOT_FOUND;
}
/**
* Returns the position of first item matching x in this
collection,
* or NOT_FOUND if not found.
* @param x any object.
* @return the position of first item matching x in this
collection,
* or NOT_FOUND if not found.
*/
private int findPos( Object x )
{
```

```
for( int i = 0; i < size( ); i++ )
if( x == null )
{
if( theItems[ i ] == null )
return i;
}
else if( x.equals( theItems[ i ] ) )
return i;
return NOT_FOUND;
}
/**
* Adds an item to this collection, at the end.
* @param x any object.
* @return true.
*/
public boolean add( AnyType x )
{
if( theItems.length == size( ) )
{
AnyType [ ] old = theItems;
theItems = (AnyType []) new Object[ theItems.length * 2 +
1 ];
for( int i = 0; i < size( ); i++ )
theItems[ i ] = old[ i ];
}
theItems[ theSize++ ] = x;
modCount++;
return true;
}
/**
* Removes an item from this collection.
* @param x any object.
* @return true if this item was removed from the collection.
*/
```

```
public boolean remove( Object x )
{
int pos = findPos( x );
if( pos == NOT_FOUND )
return false;
else
{
remove( pos );
return true;
}
}
/**
Removes an item from this collection.
* @param idx the index of the object.
* @return the item was removed from the collection.
*/
public AnyType remove( int idx )
{
AnyType removedItem = theItems[ idx ];
for( int i = idx; i < size( ) - 1; i++ )
theItems[ i ] = theItems[ i + 1 ];
theSize--;
modCount++;
return removedItem;
}
/**
* Change the size of this collection to zero.
*/
public void clear( )
{
theSize = 0;
theItems = (AnyType []) new Object[ DEFAULT_CAPACITY ];
modCount++;
}
```

```
/**
* Obtains an Iterator object used to traverse the collection.
* @return an iterator positioned prior to the first element.
*/
public Iterator<AnyType> iterator( )
{
return new ArrayListIterator( 0 );
}
/**
* Obtains a ListIterator object used to traverse the collection
bidirectionally.
* @return an iterator positioned prior to the requested element.
* @param idx the index to start the iterator. Use size() to do
complete
* reverse traversal. Use 0 to do complete forward traversal.
* @throws IndexOutOfBoundsException if idx is not between 0 and
size(), inclusive.
*/
public ListIterator<AnyType> listIterator( int idx )
{
return new ArrayListIterator( idx );
}
/**
* This is the implementation of the ArrayListIterator.
* It maintains a notion of a current position and of
* course the implicit reference to the ArrayList.
*/
private class ArrayListIterator implements ListIterator<AnyType>
{
private int current;
private int expectedModCount = modCount;
private boolean nextCompleted = false;
private boolean prevCompleted = false;
ArrayListIterator( int pos )
```

```
{
if( pos < 0 || pos > size( ) )
throw new IndexOutOfBoundsException( );
current = pos;
}
public boolean hasNext( )
{
if( expectedModCount != modCount )
throw new ConcurrentModificationException( );
return current < size( );
}
public boolean hasPrevious( )
{
if( expectedModCount != modCount )
throw new ConcurrentModificationException( );
return current > 0;
}
public AnyType next( )
{
if( !hasNext( ) )
throw new NoSuchElementException( );
nextCompleted = true;
prevCompleted = false;
return theItems[ current++ ];
}
public AnyType previous( )
{
if( !hasPrevious( ) )
throw new NoSuchElementException( );
prevCompleted = true;
nextCompleted = false;
return theItems[ --current ];
}
public void remove( )
```

```
{
if( expectedModCount != modCount )
throw new ConcurrentModificationException( );
if( nextCompleted )
ArrayList.this.remove( --current );
else if( prevCompleted )
ArrayList.this.remove( current );
else
throw new IllegalStateException( );
prevCompleted = nextCompleted = false;
expectedModCount++;
}
}
private static final int DEFAULT_CAPACITY = 10;
private static final int NOT_FOUND = -1;
private AnyType [ ] theItems;
private int theSize;
private int modCount = 0;
}
```

ΕΠΑΝΑΛΗΠΤΙΚΕΣ ΕΡΩΤΗΣΕΙΣ 6^{ΟΥ} ΚΕΦΑΛΑΙΟΥ

Ερωτήσεις:

1. Τι είναι ένας πίνακας στη Java;

2. Ποια είδη πίνακα υπάρχουν;

3. Τι είναι η αρχικοποίηση πινάκων;

4. ποια είναι η χρησιμότητα των shortcuts;

5. Τι είναι η ταξινόμηση;

6. Τι είναι η Λίστα;

Ασκήσεις:

1)Άσκηση με Ταξινόμηση Στοιχείων Πίνακα

Να γραφεί ένα πρόγραμμα που θα τοπεθετεί σε πίνακα και θα τα εκτυπώνει ταξινομημένα.

2) Έστω η παρακάτω κλάση Product. Δημιουργήστε μια κλάση PCStore, η οποία έχει μια λίστα από προϊόντα. Έπειτα δημιουργήστε μια μέθοδο, η οποία προσθέτει αντικείμενα Product στη λίστα.

```
/**
* <p>Title: </p>
*
* <p>Description: </p>
*
* <p>Copyright: Copyright (c) 2007</p>
*
* <p>Company: </p>
*
* @author not attributable
* @version 1.0
*/
public class Product {
private String περιγραφή;
private int id;
```

```
private double τιμή, έκπτωση;
public Product(int id, String περιγραφή, double τιμή, double έκπτωση) {
this.id=id;
this.περιγραφή=περιγραφή;
this.έκπτωση=έκπτωση;
this.τιμή=τιμή;
}
public String toString()
{
String detail="Προϊόν: "+περιγραφή+"\nΤιμή:
"+τιμή+"\nΈκπτωση:"+έκπτωση*100+"%";
return detail;
}
}
```

3) Έστω η παρακάτω κλάση Product. Δημιουργήστε μια κλάση PCStore, η οποία έχει μια λίστα από τομείς. Ο κάθε τομέας έχει ένα όνομα, έναν υπεύθυνο και μια λίστα από προϊόντα. Έπειτα δημιουργήστε μια μέθοδο, η οποία προσθέτει αντικείμενα Product στη λίστα. Και μια μέθοδο που προθέτει τομείς στη λίστα. Για κάθε λίστα να δημιουργήσετε μεθόδους που εκτυπώνουν τα στοιχεία των λιστών.

```
/**
* <p>Title: </p>
*
* <p>Description: </p>
*
* <p>Copyright: Copyright (c) 2007</p>
*
* <p>Company: </p>
*
* @author not attributable
* @version 1.0
*/
```

```
public class Product {
private String περιγραφή;
private int id;
private double τιμή, έκπτωση;
public Product(int id, String περιγραφή, double τιμή, double έκπτωση) {
this.id=id;
this.περιγραφή=περιγραφή;
this.έκπτωση=έκπτωση;
this.τιμή=τιμή;
}
public String toString()
{
String detail="Προϊόν: "+περιγραφή+"\nΤιμή:
"+τιμή+"\n΄Εκπτωση:"+έκπτωση*100+"%";
return detail;
}
}
```

4) Να γραφεί ένα πρόγραμμα που θα κάνει χρήση του πίνακα Vector

Σε αυτό το κεφάλαιο θα μιλήσουμε για τα νήματα, τη χρήση παραμετρικών μεθόδων, τις γενικεύσεις κλάσεων και τον πολυμορφισμό.

Με παραδείγματα θα δούμε τη χρησιμότητα των νημάτων στα προγράμματα της Java.

ΚΕΦΑΛΑΙΟ 7

1) Γενικεύσεις και νήματα

Παραμετρικοί τύποι

Με τη χρήση παραμετρικών τύπων (parameterized types) μπορούμε να επιτρέψουμε την εύκολη έκφραση νέων σύνθετων τύπων μέσα στο πρόγραμμά μας. Πώς; Γνωρίζουμε ότι με παραμέτρους σε μεθόδους μπορούμε να αλλάζουμε τη συμπεριφορά του προγράμματος κατά την εκτέλεσή του.

```
static int square(int x) {
	return x * x;
}
```

Αντίστοιχα, με παραμετρικούς τύπους μπορούμε να αλλάζουμε τη συμπεριφορά του κώδικά μας κατά τη μεταγλώττισή του.

```
class Pair <E1, E2> {
	private E1 element1;
	private E2 element2;
	public Pair(E1 e1, E2 e2) {
		element1 = e1;
		element2 = e2;
	}
	public E1 getFirst() {
		return element1;
	}
	public E2 getSecond() {
		return element2;
	}
	public String toString() {
		return "(" + element1.toString() + ", " + element2.toString() + ")";
	}
}
class Sock {}
class Man {}
```

```
class Woman {}

class Test {
        public static void main(String args[]) {
                Pair <Sock, Sock> pairOfSocks;
                Pair <Man, Woman> churchMarriedCouple;
        }
}
```

2) Παραμετρικοί τύποι στη Java

- Στη Java ορίζουμε παραμετρικούς τύπους με το όνομα μιας κλάσης ή διεπαφής ακολουθούμενο από μια λίστα από ορίσματα.

```
Pair <Sock, Sock>
Pair <Man, Woman>
Vector <String>
Collection <Integer>
Collection <Pair <Sock, Sock>>
```

- Παράμετρος ενός τύπου επιτρέπεται να είναι κάποιος τύπος αναφοράς (reference type) (κλάση, διεπαφή, ή πίνακας) ή ένας τύπος μπαλαντέρ (wildcard).
- Ο τύπος μπαλαντέρ γράφεται με ένα ? και μπορεί να ακολουθείται από ένα από τους περιορισμούς extends ή super και το όνομα ενός τύπου. Παραδείγματα:

```
Collection <?>
Collection <? extends E>
Collection <? super E>
```

(Δε θα επεκταθούμε στους τύπους αυτούς.)

- Με βάση τους παραμετρικούς τύπους ορίζουμε γενικεύσεις (generic) κλάσεων και μεθόδων.

Παράδειγμα γενίκευσης

```
class LinkedList <E> {
	/** Node's value */
	private E value;
	/** Next node */
	private LinkedList <E> next;

	/** Construct a list with a single element v */
	LinkedList(E v) {
		value = v;
		next = null;
	}

	/** Return a list with element n added to it */
	public LinkedList <E> add(E v) {
		LinkedList <E> n = new LinkedList <E>(v);
		n.next = this;
		return n;
	}

	/** Return a string representation of the list */
	public String toString() {
		String me = value.toString();

		/* Recursive implementation */
		if (next == null)
			return me;
		else
			return me + " -> " + next;
	}
	/** Test harness */
	static public void main(String args[]) {
		LinkedList <Integer> intList = new LinkedList <Integer>(0);
```

```
            intList = intList.add(1);
            intList = intList.add(18);
            intList = intList.add(45);
            for (int i = 0; i < 5; i++)
                  intList = intList.add(i * 10);
            System.out.println(intList);
      }
}
```

3) Νήματα

Ένα νήμα (thread) παριστάνει μια ροή εκτέλεσης του προγράμματος.

Ένα πρόγραμμα μπορεί να εκτελεί πολλά νήματα ταυτόχρονα. Με τον τρόπο αυτό μπορούμε να υλοποιήσουμε εφαρμογές που απαιτούν ταυτόχρονη ή ψευδοταυτόχρονη πολλαπλή επεξεργασία (π.χ. εξυπηρετητές (servers) ή παιγνίδια), να ελλατώσουμε το χρόνο που περιμένει ο χρήστης για την ολοκλήρωση μιας διεργασίας, να εκμεταλλευτούμε καλύτερα τους πόρους πολλαπλών επεξεργαστών.

Όταν υλοποιούμε κώδικα με νήματα δημιουργούνται συχνά προβλήματα συγχρονισμού που πρέπει να αντιμετωπίσουμε. Διαφορετικά νήματα μπορούν να έχουν πρόσβαση στην ίδια μεταβλητή ή πόρο του συστήματος με τρόπο που να δημιουργεί λάθη. Στην προσπάθεια να λύσουμε το παραπάνω πρόβλημα κλειδώνοντας την εκτέλεση κάποιων νημάτων μπορεί να καταλήξουμε σε αδιέξοδο (deadlock). Τα στοιχεία αυτά δεν καλύπτονται στο συγκεκριμένο μάθημα. Αντί για νήματα μπορούμε συχνά να χρησιμοποιήσουμε

πολλαπλές διεργασίες

γεγονότα (events)

ασύγχρονες (asynchronous) διεπαφές

4) Νήματα στη Java

Στη Java κάθε νήμα παριστάνεται ως ένα αντικείμενο μιας κλάσης που υλοποιεί τη διεπαφή Runnable.

Η διεπαφή Runnable απαιτεί την υλοποίηση μιας και μόνο μεθόδου

public void run()

Η μέθοδος run περιέχει τον κώδικα που εκτελεί το νήμα.

Μέσα στη μέθοδο run πρέπει να αντιμετωπίσουμε την εξαίρεση InterruptedException ή οποία μπορεί να συμβεί αν ένα άλλο νήμα τερματίσει τη λειτουργία μας.

Για να δημιουργήσουμε ένα νήμα αρκεί να περάσουμε ένα αντικείμενο της κλάσης που έχουμε ορίσει στον κατασκευαστή της κλάσης Thread

```
MyThreadClass tc = new MyThreadClass();

Thread t = new Thread(tc);
```

Για να αρχίσει η εκτέλεση του νήματος πρέπει να καλέσουμε τη μέθοδο start του αντίστοιχου αντικειμένου Thread.

```
MyThreadClass tc = new MyThreadClass();

Thread t = new Thread(tc);

t.start();
```

Μπορούμε να περιμένουμε τον τερματισμό ενός νήματος, καλώντας τη μέθοδο join ενός αντικειμένου Thread.

Η εκτέλεση του προγράμματός μας τερματίζεται όταν τερματίσουν την εκτέλεσή τους όλα τα νήματα εκτέλεσης.

Παράδειγμα χρήσης νημάτων

```
public class CoctailGuest implements Runnable {

    /** What the guest will say */
```

```
private String mumble;

/** How many seconds will he/she pause before speaking */

private int pause;

/** How long the guest will stay */

private int stay;

/** How long the guest has stayed */

private int hereFor;



/** Constructor */

public CoctailGuest(String mumble, int pause, int stay) {

        this.mumble = mumble;

        this.pause = pause;

        this.stay = stay;

        hereFor = 0;

}

/** Execution method */

public void run() {

        try {

                while (hereFor < stay) {

                        Thread.sleep(pause * 1000);
```

```
                hereFor += pause;
                System.out.println(mumble);
            }
        } catch (InterruptedException e) {
            System.out.println("Something has come up; got to go.");
            return;
        } finally {
            System.out.println("Good bye.");
        }
    }
    public static void main(String args[]) {
        final int NGUEST = 6;
        CoctailGuest guest[] = new CoctailGuest[NGUEST];
        Thread thread[] = new Thread[NGUEST];

        int i = 0;
        guest[i++] = new CoctailGuest("Can I have another drink?", 8, 30);
        guest[i++] = new CoctailGuest("Nice food!", 7, 120);
        guest[i++] = new CoctailGuest("Ha ha ha...", 3, 120);
        guest[i++] = new CoctailGuest("Hi, I am Maria.", 5, 60);
```

```
            guest[i++] = new CoctailGuest("Hello, I am Petros.", 15, 60);


            // Create the threads
            for (i = 0; i < NGUEST; i++)
                thread[i] = new Thread(guest[i]);
            // Start the threads
            for (i = 0; i < NGUEST; i++)
                thread[i].start();
        }
}
```

ΕΠΑΝΑΛΗΠΤΙΚΕΣ ΕΡΩΤΗΣΕΙΣ 7^{ΟΥ} ΚΕΦΑΛΑΙΟΥ

Ερωτήσεις:

1) Πως χρησιμοποιούμε τους παραμετρικούς τύπους;

2) Πως ορίζονται στη Java οι παραμετρικοί τύποι;

3) Ποια είναι τα είδη πολυμορφισμού;

4) Τι είναι τα νήματα και πως εκτελούνται;

5) Τι είναι η διεπαφή Runnable;

Εδώ ήρθε η ώρα να απαντήσουμε στο ερώτημα γιατί τη Java; Ποιος ο λόγος να επιλέξει κάποιος τη Java έναντι των άλλων γλωσσών προγραμματισμού;

Δεν είναι ο ρόλος μας να σας πείσουμε να απορρίψετε τις άλλες γλώσσες και να στραφείτε αποκλειστικά στη Java. Αυτό που προσπαθήσαμε να κάνουμε σε αυτό το κεφάλαιο είναι να συγκρίνουμε με αντικειμενικά κριτήρια τις υπάρχουσες γλώσσες προγραμματισμού, ώστε να ολοκληρωθούν οι γνώσεις πάνω σε αυτό το θέμα.

Η σύγκριση ξεκινάει με τη Java και τη JavaScript και συνεχίζει με τις C και C++.

ΚΕΦΑΛΑΙΟ 8

1) Σύγκριση της Java με άλλες γλώσσες προγραμματισμού

Ομοιότητες με τη C++

Η Java έχει μεγάλη ομοιότητα με τη C++ αφού διατηρεί πολλές εννοιολογικές δομές της C++, καθώς και μεγάλο μέρος της σύνταξης της. Για παράδειγμα, οι περισσότεροι τύποι προτάσεων της Java υπάρχουν και στη C++.Ακόμα , ο μηχανισμός τύπων της Java χρησιμοποιεί μια ιεραρχία τάξεων.

Οι βασικές διαφορές μεταξύ των δύο οφείλονται στο ότι οι σχεδιαστές της Java θέλησαν να κάνουν τον προγραμματισμό ευκολότερο και τα προγράμματα πιο αξιόπιστα. Συνεπώς, η Java δημιουργήθηκε για να μειώσει σε κάποιο βαθμό τη πολυπλοκότητα της C++.Ειδικότερα, η Java παραλείπει κάποια χαρακτηριστικά της γλώσσας C++ τα οποία χρησιμοποιούνται σπάνια, παρανοούνται εύκολα ή ρέπουν σε σφάλματα. Για παράδειγμα, η Java δεν υποστηρίζει την υπερφόρτωση τελεστών, τη πολλαπλή κληρονομικότητα, ή τις εκτεταμένες αυτόματες αναγκαστικές μετατροπές δεδομένων. Σε μερικές περιπτώσεις, η Java διατηρεί χαρακτηριστικά της C++, αλλά περιορίζει τη χρήση τους.

Αντιθέσεις με C, C++

Στις γλώσσες προγραμματισμού C και C++, ο compiler μετατρέπει το αρχείο με τις εντολές σε γλώσσα μηχανής για το συγκεκριμένο λειτουργικό στο οποίο γίνεται η διαδικασία και για το συγκεκριμένο CPU του υπολογιστή. Που σημαίνει ότι, μεταφέροντας το πρόγραμμα μας σε άλλο υπολογιστή και ακόμα χειρότερα σε άλλο λειτουργικό, θα πρέπει να κάνουμε την διαδικασία του compilation ξανά. Στην java όμως δεν συμβαίνει αυτό, διότι είναι η γλώσσα που γράφεται μία φορά και χρησιμοποιείται παντού.

Με απλά λόγια:

- Η Java δεν επιτρέπει πολλαπλή κληρονομικότητα (διατήρηση απλότητας)
- Κάθε κλάση μπορεί να κληρονομεί το πολύ από μία άλλη κλάση («extends»)
- Interfaces: Δομές που περιέχουν «μη υλοποιημένες» συναρτήσεις, χωρίς μεταβλητές-μέλη και δεν είναι κλάσεις
- Μία κλάση μπορεί να κάνει «implement» περισσότερα από ένα interface

- Η Java θεωρείται εν γένει απλούστερη γλώσσα από τη C++.

- Όλες οι Java μέθοδοι είναι όπως οι virtual της C++.
- Η Java δεν υποστηρίζει δείκτες (pointers).
- Η Java δεν υποστηρίζει defines, typedefs ή preprocessor. Οπότε, δε χρειάζεται ούτε αρχεία κεφαλίδας (header files).
- Στη Java δεν υποστηρίζονται καθολικές μεταβλητές. Εναλλακτικά: “static”
- Στη Java δεν επιτρέπονται συναρτήσεις εκτός κλάσεων (stand-alone functions).

Αντιστοίχηση μεταβλητών ανάμεσα σε Java και Pascal

Τα χαρακτηριστικά των αντικειμένων και οτιδήποτε άλλο θέλουμε να αποθηκεύσουμε κάπου πριν το χειριστούμε, αποθηκεύονται σε *μεταβλητές*, ανάλογα με την τιμή τους, (κείμενο, ακέραιος ή δεκαδικός αριθμός κτλ)

Τύπος	**Μέγεθος bytes** 3)	**Pascal**	**Java**	**Τιμές**4)
μικρός ακέραιος	1	shortint	short	-128..127
ακέραιος	2	integer	int	-32768..32767
μεγάλος ακέραιος	4	longint	long	-2δις έως 2δις5)
πραγματικός	6	real	float	όλοι6)
χαρακτήρας	1	char	char	
κείμενο	255	string	String	έως 255 χαρακτήρες7)
λογικός	1 8)	boolean	boolean	true ή false
μικρός ακέραιος+	1	byte	byte	0 .. 2559)
ακέραιος+	2	word		0 .. 65535

Για να μην μπλεχτούμε στις διαφορές μεταξύ γλωσσών προγραμματισμού, στους ακεραίους δεν ασχολούμαστε με τους θετικούς και ξεχνάμε ότι η java μας επιτρέπει μεγαλύτερες τιμές.

2) Γλώσσες προγραμματισμού βασισμένες στη Java

Επιλογή ανάμεσα σε Java και JavaScript

Κάποια πράγματα μπορούν να γίνουν με μια από τις δύο γλώσσες και άλλα μπορούν να γίνουν εξίσου εύκολα και με τις δύο. Μερικές ιδέες ανάπτυξης εφαρμογών και η αντίστοιχη επιλογή ανάμεσα στις δύο γλώσσες είναι:

JavaScript

- Απλή φόρμα: Μπορείτε εύκολα να δημιουργήσετε φόρμες με τη βοήθεια της HTML και να προσθέσετε το κώδικα ο οποίος θα εκτελείται κάθε φορά που το περιεχόμενο ενός αντικειμένου της φόρμας μεταβάλλεται για διευκόλυνση στην αυτοματοποίηση των υπολογισμών κατά τη συμπλήρωση της φορολογικής δήλωσης.

- Κόσμοι VRML: Μπορείτε να δημιουργήσετε εύκολα μια περιγραφή ενός τρισδιάστατου κόσμου σε γλώσσα VRML και να την εισάγετε στο VRML plug-in που διαθέτουν οι σύγχρονοι browsers.

Java

- Τρισδιάστατοι κόσμοι: Η SUN μαζί με το JDK διανέμει μια εκπληκτική εφαρμογή σχεδιασμού τρισδιάστατων μοντέλων για μόρια (χημικά). Διατίθεται ο πηγαίος κώδικας όλων των ρουτινών υπολογισμού, υλοποίησης και απεικόνισης των θέσεων τρισδιάστατων αντικειμένων σε χαμηλό επίπεδο.

- Animation**:** Για να σχεδιάσετε μέσα σε πλαίσιο, στο εσωτερικό μιας σελίδας στον browser. Η δυνατότητα αυτή είναι αναγκαία , προκειμένου να δημιουργήσετε κινούμενες εικόνες.

<u>Διαφορές στη σύνταξη</u>

Απλότητα: Δεν υπάρχουν αρχεία ορισμών, επικεφαλίδας, δομές ή μικροεντολές με τα οποία μπορεί να ‘παίξει” κανείς. Υπάρχει συνήθως ένας μόνο τρόπος για τη περίπτωση μιας συγκεκριμένης εργασίας και αυτό απλοποιεί τη διαδικασία

συγγραφής του κώδικα. Αν και υπάρχουν φορές που αυτό μπορεί να μην είναι βολικό, η απλότητα αυτή αποτελεί συνήθως καθαρό κέρδος.

Πλήρης αντικειμενοστρέφεια. Η θεμελιώδης κατασκευή είναι η κλάση η οποία διαθέτει δεδομένα και μεθόδους που επεξεργάζονται τα δεδομένα αυτά.

Χρήση συντομεύσεων. Η Java δε διαθέτει πολλές από τις επιπρόσθετες συντομεύσεις της C, C++.Πολλές από τις απλές συντομεύσεις της C, C++ έχουν διατηρηθεί και οι σχεδιαστές της Java έκαναν πολλές πρακτικές παραχωρήσεις στους εργαζομένους προγραμματιστές.

Η JavaFX

Το JavaFX Script, μια νέα γλώσσα προγραμματισμού δεσμών ενεργειών (script), προσφέρει στους προγραμματιστές Java τη δύναμη να δημιουργούν γρήγορα εμπλουτισμένες σε περιεχόμενο εφαρμογές για την ευρύτερη δυνατή γκάμα προϊόντων όπως μεταξύ άλλων κινητών τηλεφώνων, αποκωδικοποιητών τηλεόρασης, επιτραπέζιων υπολογιστών, ακόμη και δίσκων Blu-ray. Οι δημιουργοί περιεχομένου έχουν πλέον έναν απλό τρόπο να δημιουργούν περιεχόμενο για οποιαδήποτε καταναλωτική συσκευή που βασίζεται στη Java.
Οι εφαρμογές που γράφονται με το JavaFX Script αξιοποιούν το πλεονέκτημα των καθιερωμένων πλέον διακριτικών γνωρισμάτων της τεχνολογίας Java, όπως π.χ. φορητότητα ("γράφονται μία φορά, εκτελούνται οπουδήποτε), ασφάλεια εφαρμογών, διανομή παντού και εταιρική συνδεσιμότητα.

Επιπλέον, το JavaFX Script μπορεί να υποστηρίξει περιβάλλοντα χρήστη οποιουδήποτε μεγέθους και πολυπλοκότητας. Γράφεται στατικά και μπορεί να εκμεταλλευθεί τα προγραμματιστικά μοντέλα της Java, όπως π.χ. τις δυνατότητες διάρθρωσης, επαναχρησιμοποίησης και "ενθυλάκωσης" (encapsulation) κώδικα που καθιστούν δυνατή τη δόμηση και συντήρηση μεγάλων προγραμμάτων σε JavaFX Script.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΚΑΙ ΕΦΑΡΜΟΓΕΣ ΣΕ JAVA

Σε αυτή την ενότητα θα βρείτε εφαρμογές και επιπλέον πληροφορίες απ' όλα τα κεφάλαια για την καλύτερη κατανόηση όσων μάθατε, καθώς και επαναληπτικές ερωτήσεις και ασκήσεις αυτοαξιολόγησης.

Επαναληπτικές Ερωτήσεις

1. Πότε δημιουργήθηκε η γλώσσα προγραμματισμού Java;
2. Ποια εταιρεία δημιούργησε την Java;
3. Τι είναι πηγαίος κώδικας;
4. Από πού προέρχεται ο πηγαίος κώδικας;
5. Τι είδους αρχεία περιέχουν τον πηγαίο κώδικα Java;
6. Τι είναι ο κώδικας byte;
7. Από πού προέρχεται ο κώδικας byte;
8. Τι είδους αρχεία περιέχουν κώδικα byte;
9. Ποια προγράμματα ονομάζονται *φορητά* στον προγραμματισμό;
10. Σε τι διαφέρει ο κώδικας byte της Java από άλλες γλώσσες προγραμματισμού χαμηλού επιπέδου;
11. Σε τι διαφέρει ένας μεταγλωττιστής από έναν διερμηνευτή;
12. Τι είναι η εικονική μηχανή Java;
13. Τι είναι *εφαρμογή*;
14. Τι είναι ο *υπεύθυνος ανάπτυξης εφαρμογών*;
15. Τι σημαίνει Java API;
16. Τι σημαίνει IDE;
17. Τι σημαίνει SDK;
18. Τι σημαίνει JIT;
19. Τι σημαίνει JVM;
20. Ποιες είναι οι διαφορές ανάμεσα στις μεταβλητές και τα αντικείμενα;
21. Ποιοι είναι οι οχτώ στοιχειώδεις τύποι της Java;
22. Τι είναι τύπος αναφοράς;
23. Τι είναι υπερχείλιση ακέραιων αριθμών;

Εφαρμογή 1

Η πρώτη μας εφαρμογή σε Java. Σε αυτή την εφαρμογή θα δείτε ένα απλό παράδειγμα δημιουργίας ενός απλού προγράμματος σε Java. Η διαδικασία που θα ακολουθήσετε είναι η ίδια που έχουμε περιγράψει στο τρίτο κεφάλαιο.
Αν έχετε δημιουργήσει πακέτο στο NetBeans δεν χρειάζεται να δημιουργήσετε καινούργιο. Στο πακέτο που είδη έχετε ήδη ανοίξει θα δημιουργήσετε μια νέα κλάση με το όνομα HelloHellas.

Ακολουθεί η πρώτη εφαρμογή σε Java.

```
public class HelloHellas
{
	public static void main (String[ ] arguments)
	{
		System.out.println("Hello from Hellas");
		// Η πρώτη μας εφαρμογή σε Java
	}
}
```

Η εντολή class ορίζει το όνομα μιας τάξης (κλάσης, class), που εδώ είναι το HelloHellas και πρέπει να είναι υποχρεωτικά το ίδιο και με τα ίδια πεζά και κεφαλαία γράμματα με το όνομα του πηγαίου αρχείου (προγράμματος) της Java, δηλ. HelloHellas.java.

Η συνάρτηση main() πρέπει να υπάρχει υποχρεωτικά σε κάθε απλή εφαρμογή (όχι μικροεφαρμογή) της Java και είναι η πρώτη εντολή που αναζητά η Java κατά την μεταγλώττιση και εκτέλεση της εφαρμογής. Με τις αγκύλες { και }, μπορούμε να χωρίσουμε τον κώδικα του προγράμματος σε λογικές ενότητες (blocks).

Για σχόλια (comments) μίας γραμμής, μπορούμε να χρησιμοποιήσουμε τους χαρακτήρες //, ενώ για σχόλια πολλών γραμμών τους χαρακτήρες /* ή /** για την αρχή του σχολίου και τους χαρακτήρες */ για το τέλος του σχολίου. Οι χαρακτήρες /** και */ αποτελούν ένα σχόλιο τεκμηρίωσης (documentation comment), που το χρησιμοποιεί το εργαλείο (tool) javadoc όταν ετοιμάζει την αυτόματη δημιουργία τεκμηρίωσης (documentation).

Αν μεταγλωττίσετε και εκτελέσετε με επιτυχία το παραπάνω πρόγραμμα, θα πρέπει να δείτε στην οθόνη το εξής μήνυμα :

Hello from Hellas

Εφαρμογή 2

Ακολουθεί μία απλή εφαρμογή Applet με χρήση γραφικών.

```
public class weight extends java.applet.Applet
{
	int varos = 0;
	public void init()
	{
		varos = 100;
	}
	public void paint(java.awt.Graphics g)
	{
		g.drawString("Το βάρος είναι : " + varos + " κιλά", 10, 20);
	}
}
```

Σ' αυτό το πρόγραμμα δηλώνετε μια μεταβλητή με όνομα varos, που είναι ακέραιος (integer), και η έξοδός αποτελείται από ενωμένα strings και την τιμή της μεταβλητής varos. Η μέθοδος init() καλείται μόνο μία φορά όταν εκτελείται το applet και χρησιμοποιείται για να ορίσει μεταβλητές που χρειάζονται στο applet.

Η μέθοδος paint() είναι υποχρεωτική σε κάθε applet και χρησιμοποιείται για να μπορούμε να εμφανίζουμε ο,τιδήποτε θέλουμε σ' ένα applet. Χρησιμοποιεί ένα όρισμα (argument), το οποίο είναι το αντικείμενο Graphics του πακέτου java.awt, που εδώ το ονομάσαμε g.

Με την εντολή (συνάρτηση ή μέθοδο) drawString(), που ανήκει στο αντικείμενο Graphics, μπορούμε να εμφανίσουμε ένα μήνυμα στην οθόνη σε συγκεκριμένο σημείο x=10 και y=20, όπου η θέση 0, 0 είναι η πάνω αριστερή γωνία του παραθύρου εμφάνισης του applet μέσα στην ιστοσελίδα.

Εφαρμογή 3

Εφαρμογή με Strings

Για να συγκρίνουμε strings στην Java χρησιμοποιούμε την μέθοδο equals().

Για να βρούμε το μήκος ενός string στην Java χρησιμοποιούμε την μέθοδο length().

Για να μετατρέψουμε ένα string σε κεφαλαία ή πεζά γράμματα χρησιμοποιούμε τις μεθόδους toUpperCase() και toLowerCase() αντίστοιχα.

Ακολουθεί το παράδειγμα :

```
Public class StringApplets
{
	public static void main (String[ ] arguments)
	{
		String firstName = "Manuel";
		String secondName = "Christo";
		System.out.println("Είναι τα ονόματα ίδια; " +
firstName.equals(secondName));
		System.out.println("Το πρώτο όνομα έχει " + firstName.length() + "
γράμματα");
		System.out.println("Το πρώτο όνομα με κεφαλαία : " +
firstName.toUpperCase());
		System.out.println("Το δεύτερο όνομα με πεζά : " +
secondName.toLowerCase());
	}
}
```

Εφαρμογή 4

Εφαρμογή με την εντολή If.

Η εντολή if μπορεί να πάρει τις εξής μορφές στην Java :

```
int balance = -1;
if (balance < 0)
```

```
        System.out.println("Ο Λογαριασμός σας δεν έχει αντίκρυσμα");
 int balance = 3;
if (balance < 0)
        System.out.println("Ο Λογαριασμός σας δεν έχει αντίκρυσμα");
else
        System.out.println("Τα διαθέσιμα κονδύλια είναι : " + balance);
 int balance = 0;
if (balance < 0)
        System.out.println("Ο Λογαριασμός σας δεν έχει αντίκρυσμα");
else if (balance = = 0)
        System.out.println("Δεν υπάρχουν κονδύλια στον Λογαριασμός σας");
else
        System.out.println("Τα διαθέσιμα κονδύλια είναι : " + balance);
```

Εφαρμογή 5

Εφαρμογή με την εντολή Switch

Εκτός από τις φωλιασμένες εντολές if-else, ένας άλλος τρόπος για να ελέγξουμε πολλές διαφορετικές καταστάσεις είναι να χρησιμοποιήσουμε την εντολή switch.

```
public class Months
{
        public static void main (String[ ] arguments)
        {
                int month = 0;
                if (argument.length > 0)
                        month = Integer.parseInt(arguments[0]);
                switch (month)
                {
                        case (1):
                                System.out.println("Ιανουάριος");
                                break;
                        case (2):
                                System.out.println("Φεβρουάριος");
```

```
                break;
            case (3):
                System.out.println("Μάρτιος");
                break;
            ...
            default:
                System.out.println("Μάρτιος");
        }
    }
}
```

Εφαρμογή 6

Εφαρμογή με την εντολή For

Ο βρόχος (loop) for ζητάει από τον υπολογιστή να εκτελέσει μια συγκεκριμένη εργασία έναν ορισμένο αριθμό φορών.

```
public class Numbers01
{
    public static void main(String[ ] arguments)
    {
        for (int number = 0; number < 100; number++)
        {
            System.out.println("Αριθμός : " + number + ".");
        }
    }
}
```

Εφαρμογή 7

Εφαρμογή με την εντολή While

Ο βρόχος while ελέγχει μια συνθήκη και αν αυτή είναι αληθής (true), επαναλαμβάνει μια ομάδα εντολών.

```
public class Numbers02
```

```
{
	public static void main(String[ ] arguments)
	{
		int number = 0;
		while (number < 100)
		{
			System.out.println("Αριθμός : " + number + ".");
			number++;
		}
	}

}
```

Εφαρμογή 8

Εφαρμογή με την εντολή do… while.

Η Εντολή Do ... While. Ο βρόχος do ... while εκτελεί μια ομάδα εντολών μία φορά τουλάχιστον και μετά ελέγχει μια συνθήκη και αν αυτή είναι αληθής (true), τότε επαναλαμβάνει την ομάδα εντολών.

```
public class Numbers03
{
	public static void main(String[ ] arguments)
	{
		int number = 0;
		do
		{
			System.out.println("Αριθμός : " + number + ".");
			number++;
		}
		while (number < 100);
	}
}
```

Εφαρμογή 9

Εφαρμογή με την εντολή Break

Για να βγούμε από έναν βρόχο, ο κανονικός τρόπος είναι να γίνει ψευδής (false) η συνθήκη ελέγχου του βρόχου, υπάρχουν όμως περιπτώσεις που θέλουμε να βγούμε νωρίτερα από τον βρόχο και προς τον σκοπό αυτό χρησιμοποιούμε την εντολή break.

```
public class Numbers04
{
	public static void main(String[ ] arguments)
	{
		int number = 0;
		do
		{
			number++;
			System.out.println("Αριθμός : " + number + ".");
			if (number = = 50)
				break;
		}
		while (number < 100);
	}
}
```

Εφαρμογή 10

Εφαρμογές με πίνακες (Arrays) στην Java

Μπορούμε να δημιουργήσουμε πίνακες (arrays) γι' όλα τα είδη των πληροφοριών που μπορούν να αποθηκευθούν σαν μεταβλητές, ως εξής :

```
String[ ] babyName;
int[ ] babyAge;
boolean[ ] babyWeight;
```

Οι παραπάνω εντολές δημιούργησαν τρεις πίνακες αλλά δεν έχει αποθηκευθεί ακόμα τίποτα μέσα τους. Για να γίνει αυτό, πρέπει να χρησιμοποιήσουμε την εντολή (τελεστή) new και να καθορίσουμε πόσα στοιχεία θα έχει ο πίνακας, ως εξής :

```
String[ ] babyName = new String[40];
```

Μπορούμε να εκχωρήσουμε και τιμές μαζί με την δήλωση ενός πίνακα, ως εξής :

```
int[ ] babyAge = {2, 5, 4, 3, 1};
```

Οι τιμές που θα καταχωρηθούν στον πίνακα βρίσκονται ανάμεσα στους χαρακτήρες { και } και χωρίζονται με κόμματα. Ακόμη, όλα τα στοιχεία του πίνακα πρέπει να είναι του ίδιου τύπου και δεν καθορίζουμε τον αριθμό τους.

Τα στοιχεία ενός πίνακα αριθμούνται από το 0 έως το n-1, όπου το n είναι το πλήθος των στοιχείων του πίνακα. Για να βρούμε το πλήθος των στοιχείων ενός πίνακα, μπορούμε να χρησιμοποιήσουμε την ιδιότητα length, ως εξής :

```
int[ ] babyAge = {2, 5, 4, 3, 1};
System.out.println("Υπάρχουν " + babyAge.length + " μωρά");
String[ ] Names = {"Anna", "Georgia", "Maria", "Alexia"};
for (int count = 0; count < Names.length; count++)
{
      System.out.println("Ονομα : " + Names[count]);
}
```

Για να δημιουργήσουμε έναν πίνακα δύο διαστάσεων, δίνουμε την εξής εντολή :

```
String[ ] babyDetail = new String[5][5];
```

Εφαρμογή 11

Εφαρμογή με κληρονομικότητα.

Μια τάξη που κληρονομεί χαρακτηριστικά από μια άλλη τάξη καλείται δευτερεύουσα τάξη (subclass) και η τάξη από την οποία κληρονομεί καλείται υπερτάξη (superclass). Για παράδειγμα, θα δημιουργήσουμε μια τάξη με όνομα mammal (θηλαστικό), η οποία θα έχει την ιδιότητα (attribute) name και την μέθοδο (method) sleep().

Μετά, χρησιμοποιώντας την λέξη κλειδί extends, θα δημιουργήσουμε μια νέα τάξη με όνομα cat, η οποία θα κληρονομεί όλα τα χαρακτηριστικά της τάξης mammal αλλά θα έχει και μια επιπλέον δική της μέθοδο με όνομα speak().

```
public class mammal
{
        String name;
        public void sleep()
        {
                System.out.println("snore ... snore ...");
        }
}
public class cat extends mammal
{
        public void speak()
        {
                System.out.println("νιάου νιάου");
        }
}
```

Εχουμε τώρα μια δευτερεύουσα τάξη με όνομα cat της υπερτάξης mammal, όπου η τάξη cat έχει την ιδιότητα name και την μέθοδο sleep() που έχει κληρονομήσει από την υπερτάξη mammal καθώς και την δική της μέθοδο speak().

Ακολουθεί έναα παράδειγμα που δημιουργεί αντικείμενα από την παραπάνω τάξη cat και χρησιμοποιεί τα χαρακτηριστικά της.

```
class newcat
{
        public static void main(String arguments[ ])
        {
                cat catty = new cat();
                catty.name = "Azor";
                System.out.println("O Azor νιαουρίζει");
                catty.speak();
                System.out.println("O Azor κοιμάται");
```

```
            catty.sleep();
        }
}
```

Η συμπεριφορά και οι ιδιότητες μιας τάξης αποτελούν τον συνδυασμό της δικής της συμπεριφοράς και ιδιοτήτων καθώς και της συμπεριφοράς και των ιδιοτήτων που έχει κληρονομήσει απ' όλες τις υπερτάξεις της. Δύο μέθοδοι μπορούν να έχουν το ίδιο όνομα αν έχουν διαφορετικό αριθμό ορισμάτων ή ορίσματα διαφορετικού τύπου.

Μέσα σε μια δευτερεύουσα τάξη μπορούμε να ξαναορίσουμε και να αντικαταστήσουμε έτσι μια μέθοδο που έχει ήδη οριστεί σε μια υπερτάξη. Αυτό αποκαλείται υπερκάλυψη (overriding) της μεθόδου.

Εφαρμογή 12

Εφαρμογές (Applets).

Όλα τα applets αποτελούν δευτερεύουσες τάξεις (subclasses) της τάξης Applet, η οποία είναι μέρος του πακέτου java.applet. Δεν υπάρχει καμία κύρια μέθοδος main() σ' ένα applet της Java, όπως συμβαίνει με τις εφαρμογές, και έτσι δεν υπάρχει ένα καθορισμένο σημείο εκκίνησης για το πρόγραμμα. Αντίθετα, ένα applet περιέχει μια ομάδα από στάνταρτ μεθόδους που είναι έτοιμες να αντιμετωπίσουν συγκεκριμένα συμβάντα καθώς εκτελείται το applet.

Σ' αντίθεση με τις εφαρμογές, οι τάξεις των applets πρέπει να είναι public για να μπορούν να δουλέψουν και δηλώνονται ως εξής :

```
public class lesson01 extends java.applet.Applet
{
    ...
}
```

Η παραπάνω τάξη κληρονομεί (extends) όλες τις μεθόδους που εμφανίζονται αυτόματα όταν χρειασθεί και που είναι οι init(), paint(), start(), stop() και destroy(). Αν θέλουμε να συμβεί κάτι σ' ένα applet, θα πρέπει να υπερκαλύψουμε (override) αυτές τις μεθόδους και οι μέθοδοι που υπερκαλύπτουμε συχνότερα είναι οι paint() και init().

Η μέθοδος paint() αποτελεί μέρος κάθε applet που θα δημιουργούμε και χωρίς αυτήν δεν μπορούμε να εμφανίσουμε τίποτα στην οθόνη. Χρησιμοποιεί ένα όρισμα της τάξης Graphics και η σύνταξή της είναι ως εξής :

```
public class paint(Graphics screen)
{
   ...
}
```

Θα πρέπει να έχουμε προσθέσει νωρίτερα την εντολή import java.awt.Graphics στην αρχή του κώδικα και πριν από τον ορισμό της τάξης.

Η μέθοδος init() καλείται μόνο μία φορά όταν εκτελείται το applet και χρησιμοποιείται για να ορίσει μεταβλητές που χρειάζονται στο applet, όπως για παράδειγμα γραμματοσειρές (fonts), χρώματα (colors) κ.ά.

Η μέθοδος start() καλείται όταν το applet αρχίζει να εκτελείται και για να κληθεί για δεύτερη φορά, πρέπει να σταματήσει η εκτέλεση του applet.

Η μέθοδος stop() καλείται όταν σταματάει η εκτέλεση του applet, δηλ. όταν φεύγουμε από την ιστοσελίδα.

Η μέθοδος destroy() καλείται αμέσως πριν τερματισθεί το applet.

Για να τοποθετήσουμε ένα applet σε μια ιστοσελίδα, θα πρέπει να χρησιμοποιήσουμε τα tags <applet> και </applet>, ως εξής :

<applet code="lesson01.class" codebase="java" height=100 width=250> Πρέπει να έχετε έναν φυλλομετρητή που να υποστηρίζει Java </applet>

Η ιδιότητα code αναφέρει το όνομα του applet που θα εκτελεσθεί, η codebase τον φάκελο όπου υπάρχει ο κώδικας του applet αν δεν είναι ο τρέχον φάκελος και οι height και width το ύψος και το πλάτος αντίστοιχα του ορθογωνίου που θα εμφανίσει το applet στην οθόνη. Ανάμεσα στα tags <applet> και </applet> μπορούμε να γράψουμε ένα κείμενο για να εμφανισθεί σε φυλλομετρητές που δεν υποστηρίζουν Java.

Ακολουθεί ένα απλό παράδειγμα applet.

```
import java.awt.Graphics;
public class message extends java.applet.Applet
{
	int Value;
	public void init()
	{
		Value = 0;
	}
	public void paint(Graphics screen)
	{
		screen.drawString("Η αξία του αυτοκινήτου είναι : " + Value, 10, 30);
	}
}
```

Η μέθοδος drawString() χρησιμοποιείται για να εμφανίσουμε μηνύματα σ' ένα applet της Java αντί για την System.out.println().

Μέσα σ' ένα applet μπορούμε να χρησιμοποιήσουμε την μέθοδο repaint() για να καλέσουμε ξανά την μέθοδο paint() στην περίπτωση που έχει γίνει κάποια αλλαγή στην τιμή κάποιας μεταβλητής. Για παράδειγμα, μπορούμε να ξαναγράψουμε την παραπάνω μικροεφαρμογή αλλάζοντας μόνο την μέθοδο paint(), ως εξής :

```
	public void paint(Graphics screen)
	{
		screen.drawString("Η αξία του αυτοκινήτου είναι : " + Value, 10, 30);
		Value++;
		repaint();
	}
```

Η τιμή του Value θα αλλάζει συνέχεια και θα καλούμε την μέθοδο repaint(), που στην ουσία είναι η ίδια η μέθοδος paint(), για να εμφανισθεί η νέα τιμή της Value.

Εφαρμογή 13

Πέρασμα Παραμέτρων σε Applets

Μπορούμε να διαβιβάσουμε παραμέτρους σ' ένα applet χρησιμοποιώντας το tag <param> και τις ιδιότητές του (attributes) name και value. Μπορούμε να έχουμε περισσότερα από ένα tag <param> σ' ένα applet, τα οποία πρέπει να βρίσκονται ανάμεσα στα tags <applet> και </applet>. Η ιδιότητα name χρησιμοποιείται για να ξεχωρίσει η παράμετρος ενώ η ιδιότητα value δίνει μια τιμή στην παράμετρο.

Με την μέθοδο getParameter() της τάξης Applet μπορούμε να διαβάσουμε την τιμή μιας παραμέτρου ενός tag <param>. Σαν όρισμα στην μέθοδο δίνουμε το όνομα της παραμέτρου όπως ορίζεται στην ιδιότητα name. Για παράδειγμα, αν έχουμε τα εξής σ' ένα tag <applet> :

```
<applet>
   ...
   <param name ="text" value="Home">
   ...
</applet>
```

Τότε με την παρακάτω εντολή μέσα από τον κώδικα του applet της Java :

```
   String buttonText = getParameter("text")
```

θα καταχωρηθεί η τιμή "Home" στο string buttonText.

Επειδή η μέθοδος getParameter() επιστρέφει όλες τις παραμέτρους σαν strings, θα πρέπει να κάνουμε μετατροπές τύπου, ως εξής :

```
   <param name="age" value="36">
   ...
   int age;
   String ageParam = getParameter("age");
   age = Integer.parseInt(ageParam);
```

Αν μια παράμετρος δεν υπάρχει, τότε η μέθοδος getParameter() επιστρέφει μια τιμή null.

```
   String pname;
   String nameParam = getParameter("username");
   if (nameParam != null)
```

```
    pname = nameParam;
...
screen.drawString("Ονομα χρήστη : " + pname, 10, 30);
```

Εφαρμογή 14

Η Τάξη Font (Γραμματοσειρές) Για να μπορέσουμε να εμφανίσουμε μια γραμματοσειρά στην Java, πρέπει να γνωρίζουμε τα εξής στοιχεία γι' αυτήν :

- Τύπος (Arial, Courier κλπ)
- Στυλ (bold, italic, plain)
- Μέγεθος σε points

Για να δημιουργήσουμε ένα δικό μας αντικείμενο γραμματοσειράς, χρησιμοποιούμε την τάξη Font με τον τελεστή new και περνάμε τρία ορίσματα, όπως δηλώθηκαν παραπάνω. Μπορούμε να χρησιμοποιήσουμε τις σταθερές Font.PLAIN, Font.BOLD, Font.ITALIC και Font.BOLD+Font.ITALIC.

Με την μέθοδο setFont() ορίζουμε μια γραμματοσειρά ενώ με την μέθοδο getFont() μπορούμε να δούμε ποια γραμματοσειρά χρησιμοποιείται εκείνη τη στιγμή. Τα στοιχεία της τρέχουσας γραμματοσειράς χρησιμοποιούνται για την εμφάνιση του κειμένου στην οθόνη.

Ακολουθεί ένα παράδειγμα.

```
import java.awt.*;
public class Fonts extends java.applet.Applet
{
    public void paint(Graphics screen)
    {
        Font currentFont = new Font("TimesRoman", Font.PLAIN, 20);
        screen.setFont(currentFont);
        screen.drawString("Γραμματοσειρά Plain Times Roman 20 point", 10, 30);
        Font combFont = new Font("TimesRoman", Font.BOLD+Font.ITALIC, 40);
        screen.setFont(combFont);
```

```
        screen.drawString("Γραμματοσειρά Bold+Italic Times Roman 40 point", 10,
50);
    }
}
```

Εφαρμογή 15

Εφαρμογή με την τάξη Color (Χρώματα)

Μπορούμε να χρησιμοποιήσουμε δύο τρόπους για να ορίσουμε το χρώμα που θέλουμε να εμφανίσουμε σ' ένα applet. Ο πρώτος χρησιμοποιεί μια σταθερή μεταβλητή της τάξης Color και ο δεύτερος ορίζει νέα χρώματα με βάση τις RGB (red, green, bule) τιμές τους από 0-255.

Μπορούμε να ορίσουμε το χρώμα του φόντου της εφαρμογής με την μέθοδο setBackground() και το χρώμα του κειμένου της εφαρμογής με την μέθοδο setColor(). Ακολουθεί ένα παράδειγμα.

```
import java.awt.*;
public class Colors extends java.applet.Applet
{
    public void paint(Graphics screen)
    {
        setBackground(Color.green);
        screen.drawString("Χρώμα φόντου μαύρο", 10, 30);
        screen.setColor(Color.blue);
        screen.drawString(Χρώμα κειμένου κόκκινο", 10, 50);
        Color mystery = new Color(100, 200, 50);
        setBackground(mystery);
        screen.drawString("Χρώμα φόντου τυχαίο", 10, 70);
    }
}
```

Εφαρμογή 16

Εφαρμογή με χρήση Γραφικών

Στην Java μπορούμε να σχεδιάσουμε τα εξής γραφικά :

Γραμμές

Ορθογώνια και Τετράγωνα

Κύκλους και Ελλείψεις

Τόξα

Πολύγωνα

Ολα τα παραπάνω να είναι γεμισμένα με χρώμα ή άδεια

Για να σχεδιάσουμε ένα γραφικό στην Java χρησιμοποιούμε μια μέθοδο της τάξης Graphics.

Για να σχεδιάσουμε μια γραμμή χρησιμοποιούμε την μέθοδο drawLine() ως εξής :

```
screen.setColor(Color.red);
screen.drawLine(100, 150, 170, 200);
```

Η παραπάνω εντολή σχεδιάζει μια γραμμή με πλάτος 1 pixel και χρώμα κόκκινο από την θέση (100, 150) της οθόνης μέχρι την θέση (170, 200).

Για να σχεδιάσουμε ένα ορθογώνιο ή τετράγωνο έχουμε πολλές επιλογές :

Η μέθοδος drawRect() για απλό ορθογώνιο, μη γεμισμένο και χωρίς στρογγυλεμένες γωνίες :

```
screen.drawRect(100, 150, 200, 250)
```

Η παραπάνω εντολή σχεδιάζει ένα ορθογώνιο με πάνω αριστερά κορυφή στο σημείο 100, 150, πλάτος 200 και ύψος 250.

Η μέθοδος fillRect() για γεμισμένο ορθογώνιο :

```
screen.setColor(Color.blue);
screen.fillRect(100, 150, 200, 250)
```

Η παραπάνω εντολή σχεδιάζει ένα ορθογώνιο γεμισμένο με μπλε χρώμα, με πάνω αριστερά κορυφή στο σημείο 100, 150, πλάτος 200 και ύψος 250.

Η μέθοδος drawRoundRect() για ορθογώνιο με στρογγυλεμένες γωνίες :

```
screen.setColor(Color.green);
screen.drawRoundRect(100, 150, 200, 250, 15, 15)
```

Η μέθοδος fillRoundRect() για γεμισμένο ορθογώνιο με στρογγυλεμένες γωνίες :

```
screen.setColor(Color.yellow);
screen.fillRoundRect(100, 150, 200, 250, 15, 15)
```

Οι μέθοδοι drawRoundRect() και fillRoundRect() έχουν δύο επιπλέον παραμέτρους που είναι η απόσταση από την γωνία που θέλουμε να ξεκινάει η καμπύλη.

Για να σχεδιάσουμε μια έλλειψη ή έναν κύκλο χρησιμοποιούμε την μέθοδο drawOval() ως εξής :

```
screen.setColor(Color.red);
screen.drawOval(100, 150, 20, 30);
```

Η παραπάνω εντολή σχεδιάζει μια έλλειψη με κόκκινο χρώμα, με πάνω αριστερά κορυφή στο σημείο 100, 150, πλάτος 20 και ύψος 30.

Η μέθοδος fillOval() γεμίζει την έλλειψη με χρώμα :

```
screen.setColor(Color.yellow);
screen.fillOval(100, 150, 20, 30);
```

Για να σχεδιάσουμε ένα τόξο χρησιμοποιούμε την μέθοδο drawArc() ως εξής :

```
screen.setColor(Color.blue);
screen.drawArc(100, 150, 30, 20, 45, 180);
```

Η παραπάνω εντολή σχεδιάζει ένα τόξο έλλειψης με μπλε χρώμα, με πάνω αριστερά κορυφή στο σημείο 100, 150, πλάτος 20, ύψος 30, γωνία εκκίνησης του τόξου στις 45 μοίρες και γωνία του τόξου 180 μοίρες με φορά αντίθετη αυτής των δεικτών του ρολογιού.

Η μέθοδος fillArc() γεμίζει το τόξο με χρώμα :

```
screen.setColor(Color.green);

screen.fillArc(100, 150, 20, 20, 90, -180);
```

Για να σχεδιάσουμε ένα πολύγωνο, πρέπει πρώτα να δημιουργήσουμε έναν πίνακα (array) ακεραίων που θα περιέχει τις x συντεταγμένες και έναν άλλον πίνακα ακεραίων που θα περιέχει τις y συντεταγμένες των κορυφών του πολυγώνου. Χρησιμοποιούμε την μέθοδο drawPolygon(), η οποία παίρνει σαν παραμέτρους τους δύο πίνακες με τις συντεταγμένες καθώς και έναν ακέραιο αριθμό που παριστάνει τον αριθμό των κορυφών του πολυγώνου.

```
int[] xPoints = {100, 150, 200, 250};
int[] yPoints = {50, 100, 150, 300};
int point = 4;
screen.setColor(Color.red);
screen.drawPolygon(xPoints, yPoints, points);
```

Η μέθοδος fillPolygon() γεμίζει το πολύγωνο με χρώμα :

```
screen.fillPolygon(xPoints, yPoints, points);
```

Εφαρμογή 17

Εφαρμογή με πλήκτρα εντολής

Για να μπορέσουμε να δημιουργήσουμε ένα συστατικό, όπως ένα πλήκτρο εντολής (button) σ' ένα πρόγραμμα της Java, δημιουργούμε πρώτα το αντικείμενο και μετά χρησιμοποιούμε την μέθοδο add() για να το προσθέσουμε σ' ένα υπάρχον συστατικό.

Ενα συστατικό Button είναι ένα πλήκτρο με δυνατότητα κλικ και με μια ετικέτα (label, caption) που περιγράφει τι θα συμβεί αν κάνουμε κλικ στο πλήκτρο. Για να προσθέσουμε ένα button σ' ένα applet, γράφουμε τον εξής κώδικα :

```
import java.awt.*;
public class lesson20 extends java.applet.Applet
{
	Button helloButton = new Button("Hello");
	public void init()
	{
		add(helloButton);
	}
}
```

Εφαρμογή 18

Εφαρμογή με ετικέτες και πλαίσια κειμένου

Μια ετικέτα (Label) εμφανίζει ένα μήνυμα (string) που δεν μπορεί να αλλαχθεί από τον χρήστη, ενώ ένα πλαίσιο κειμένου (TextField) είναι μια περιοχή όπου ο χρήστης μπορεί να καταχωρήσει μία γραμμή κειμένου. Μπορούμε να δημιουργήσουμε ετικέτες και πλαίσια κειμένου σ' ένα applet της Java ως εξής :

```
import java.awt.*;
public class lesson21 extends java.applet.Applet
```

```
{
    Label emailLabel = new Label("Διεύθυνση e-mail : ", Label.RIGHT);
    TextField emailAddress = new TextField(30);
    public void init()
    {
        add(emailLabel);
        add(emailAddress);
    }
}
```

Εφαρμογή 19

Εφαρμογή με Πλαίσια Ελέγχου και Πλήκτρα Επιλογής

Ενα πλαίσιο ελέγχου (checkbox) είναι ένα τετράγωνο συστατικό που μπορούμε να επιλέξουμε ανεξάρτητα από άλλα πλαίσια ελέγχου, ενώ ένα πλήκτρο επιλογής (radio button) είναι ένα κυκλικό συστατικό όπου μπορούμε να επιλέξουμε ένα και μόνο ένα από μια ομάδα πλήκτρων επιλογής. Η Java χαρακτηρίζει και τα πλαίσια ελέγχου και τα πλήκτρα επιλογής σαν CheckBox και τα διαχωρίζει αν κατά την στιγμή της δημιουργίας τους ανήκουν ή όχι σε μια ομάδα (CheckboxGroup).

Για να δημιουργήσουμε ένα πλαίσιο ελέγχου σ' ένα applet γράφουμε τον εξής κώδικα :

```
import java.awt.*;
public class lesson22a extends java.applet.Applet
{
    Checkbox emailMe = new Checkbox("Στείλτε e-mail", true);
    public void init()
    {
        add(emailMe);
    }
}
```

Μπορούμε να προσθέσουμε δύο ορίσματα κατά την δημιουργία ενός πλαισίου ελέγχου, όπου το πρώτο είναι μια επεξηγηματική γραμμή κειμένου που θα εμφανισθεί δίπλα στο πλαίσιο ελέγχου και το δεύτερο είναι μια τιμή boolean που

διευκρινίζει αν το πλαίσιο ελέγχου θα είναι επιλεγμένο (true) ή όχι (false) όταν εκτελεσθεί η εφαρμογή.

Για να δημιουργήσουμε πλήκτρα επιλογής σ' ένα applet πρέπει πρώτα να δημιουργήσουμε μια ομάδα πλαισίων ελέγχου με το αντικείμενο CheckboxGroup και μετά κατά την δημιουργία των πλαισίων ελέγχου να χρησιμοποιήσουμε και μια τρίτη παράμετρο που να δηλώνει την ομάδα στην οποία θα ανήκει αυτό το πλαίσιο ελέγχου, άρα στην ουσία δημιουργούμε ένα πλήκτρο επιλογής. Γράφουμε τον εξής κώδικα :

```
import java.awt.*;
public class lesson22b extends java.applet.Applet
{
	CheckboxGroup cards = new CheckboxGroup();
	Checkbox visacard = new Checkbox("Visa Card", true, cards);
	Checkbox mastercard = new Checkbox("Master Card", false, cards);
	Checkbox dinerscard = new Checkbox("Diners Card", false, cards);
	public void init()
	{
		add(visacard);
		add(mastercard);
		add(dinerscard);
	}
}
```

Δημιουργήσαμε ένα checkbox group με όνομα cards και όλα τα πλαίσια ελέγχου (στην ουσία πλήκτρα επιλογής) που δημιουργήσαμε προστέθηκαν σ' αυτή την ομάδα (group).

Εφαρμογή 20

Εφαρμογή με Λίστες Επιλογής

Μια λίστα επιλογής (choice) είναι μια πτυσσόμενη λίστα επιλογών από τις οποίες μόνο μία μπορεί να είναι επιλεγμένη. Είναι γνωστή και σαν πτυσσόμενο ή σύνθετο

πλαίσιο (combo box). Για να δημιουργήσουμε μια λίστα επιλογής, δημιουργούμε πρώτα το αντικείμενο και μετά προσθέτουμε μία-μία τις επιλογές, ως εξής :

```
import java.awt.*;
public class lesson23 extends java.applet.Applet
{
	Choice sport = new Choice();
	public void init()
	{
		sport.add("Μπάσκετ");
		sport.add("Βόλλεϋ");
		sport.add("Ποδόσφαιρο");
		sport.add("Κολύμβηση");
		add.sport();
	}
}
```

Δημιουργήσαμε μια λίστα επιλογής με τέσσερα αθλήματα.

Εφαρμογή 21

Εφαρμογή με Περιοχές Κειμένου

Μια περιοχή κειμένου (TextArea) είναι ένα πεδίο κειμένου όπου μπορούμε να καταχωρήσουμε περισσότερες από μία γραμμές κειμένου. Σε μια περιοχή κειμένου πρέπει να καθορίσουμε το πλάτος και το ύψος της και μπορούμε να τοποθετήσουμε και ένα αρχικό κείμενο μαζί την δήλωσή της.

Μπορούμε να δημιουργήσουμε μια περιοχή κειμένου σ' ένα applet της Java ως εξής :

```
import java.awt.*;
public class lesson24 extends java.applet.Applet
{
	TextArea comments = new TextArea("Florina per sempre", 10, 20);
	public void init()
	{
		add(comments);
```

```
    }
}
```

Εφαρμογή 22

Εφαρμογή με Layout Managers

Τα αντικείμενα Panel ανήκουν στην κατηγορία των αντικειμένων που είναι γνωστά σαν υποδοχείς (containers). Τα αντικείμενα αυτά δεν εμφανίζουν τίποτα και χρησιμοποιούνται για να περιέχουν άλλα αντικείμενα.

Το προκαθορισμένο layout manager είναι η τάξη FlowLayout που έχουμε χρησιμοποιήσει σ' όλα τα παραδείγματά μας μέχρι τώρα, χωρίς να το έχουμε δηλώσει ρητά. Σύμφωνα μ' αυτό το layout manager, τα αντικείμενα τοποθετούνται σε μια περιοχή από αριστερά προς τα δεξιά και στην επόμενη γραμμή όταν δεν υπάρχει άλλος χώρος.

Εφαρμογή 23

Εφαρμογή με GridLayout Manager

Η τάξη GridLayout οργανώνει τα συστατικά σ' έναν υποδοχέα (container) μ' έναν συγκεκριμένο αριθμό γραμμών και στηλών, ως εξής :

```
GridLayout ourLayout = new GridLayout(3, 4);
setLayout(ourLayout);
```

Η παραπάνω διάταξη ορίζει την τοποθέτηση των αντικειμένων σ' ένα πλέγμα (grid) με 3 γραμμές και 4 στήλες.

Εφαρμογή 24

Εφαρμογή με BorderLayout Manager

Η τάξη BorderLayout τακτοποιεί τα συστατικά σε πέντες περιοχές, όπου το τέσσερις χαρακτηρίζονται από τις θέσεις της πυξίδας, δηλ. North, South, East, West και μια για το κέντρο (Center), ως εξής :

```
import java.awt.*;
```

```
public class lesson27 extends java.applet.Applet
{
	Button oneButton = new Button("Ενα");
	Button twoButton = new Button("Δύο");
	Button threeButton = new Button("Τρία");
	Button fourButton = new Button("Τέσσερα");
	Button fiveButton = new Button("Πέντε");
	public void init()
	{
		BorderLayout ourLayout = new BorderLayout();
		setLayout(ourLayout);
		add(oneButton, "North");
		add(twoButton, "South");
		add(threeButton, "East");
		add(fourButton, "West");
		add(fiveButton, "Center");
	}
}
```

Εφαρμογή 25

Τα Συμβάντα Χρήστη

Η απόκριση στα συμβάντα χρήστη (user events) σ' ένα πρόγραμμα της Java απαιτεί την χρήση ενός ή περισσοτέρων EventListener interfaces. Οι τάξεις ακρόασης (listening classes) αποτελούν μέρος του πακέτου java.awt.event και μπορούμε να τις κάνουμε διαθέσιμες με την εξής εντολή :

```
import java.awt.event.*;
```

Πρέπει επίσης να χρησιμοποιήσουμε την εντολή implements για να δηλώσουμε ότι θα χρησιμοποιήσουμε ένα ή περισσότερα listening interfaces. Η επόμενη εντολή δημιουργεί μια τάξη που χρησιμοποιεί το ActionListener, το οποίο είναι ένα interface που χρησιμοποιείται με πλήκτρα εντολής (buttons) και άλλα συστατικά.

```
public class Graph extends java.applet.Applet implements ActionListener
```

```
{
    ...
}
```

Το EventListener interface δίνει την δυνατότητα σ' ένα συστατικό ενός γραφικού interface χρήστη να παράγει συμβάντα χρήστη. Ενα πρόγραμμα πρέπει να περιλαμβάνει ένα listener interface για κάθε τύπο συστατικού που επιθυμεί να ακούσει, ως εξής :

```
public class Graph3d extends java.applet.Applet implements ActionListener, MouseListener
{
    ...
}
```

Αφού έχουμε δημιουργήσει το interface που απαιτείται για ένα συγκεκριμένο συστατικό, θα πρέπει να ορίσουμε αυτό το συστατικό ώστε να δημιουργεί συμβάντα χρήστη. Για να κάνουμε ένα πλήκτρο εντολής (button) να παράγει ένα συμβάν, χρησιμοποιούμε την μέθοδο addActionListener(), ως εξής :

```
Button button1 = new Button("Εκκίνηση");

button1.addActionListener(this);
```

Οταν παράγεται ένα συμβάν χρήστη από ένα συστατικό που περιέχει έναν listener, θα κληθεί αυτόματα μια μέθοδος. Αυτές οι μέθοδοι πρέπει να βρίσκονται στην τάξη που καθορίσθηκε όταν ο listener προστέθηκε στο συστατικό.

Ο κάθε listener διαθέτει διαφορετικές μεθόδους που καλούνται για να συλλάβουν τα συμβάντά τους. Για παράδειγμα, το ActionListener interface στέλνει συμβάντα σε μια τάξη που καλείται actionPerformed(), ως εξής :

```
void public actionPerformed(ActionEvent e)
{
    ...
}
```

Ολα τα συμβάντα ενέργειας που στέλνονται σ' αυτό το πρόγραμμα θα πάνε σ' αυτήν την μέθοδο. Στην συγκεκριμένη περίπτωση στέλνουμε ένα αντικείμενο με όνομα e της τάξης ActionEvent στην μέθοδο actionPerformed(). Υπάρχουν πολλές διαφορετικές τάξεις αντικειμένων που αντιπροσωπεύουν τα συμβάντα χρήστη που μπορούν να σταλούν σ' ένα πρόγραμμα.

Αυτές οι τάξεις έχουν μεθόδους που μπορούμε να χρησιμοποιήσουμε για να βρούμε ποιο συστατικό προκάλεσε την εκτέλεση του συμβάντος. Στην μέθοδο actionPerformed() μπορούμε να αναγνωρίσουμε το συστατικό αυτό με την εξής εντολή :

```
String cmd = e.getActionCommand();
```

Η μέθοδος getActionCommand() επιστρέφει ένα string και αν το συστατικό είναι ένα πλήκτρο εντολής (button), η τιμή του string θα είναι η ετικέτα (label, caption) του πλήκτρου εντολής, ενώ αν το συστατικό είναι ένα πλαίσιο κειμένου (text field), η τιμή του string θα είναι το κείμενο που έχει καταχωρηθεί στο πλαίσιο κειμένου.

Με την μέθοδο getSource() μπορούμε να μάθουμε το αντικείμενο που προκάλεσε το συμβάν, ως εξής :

```
Object source = e.getSource();
```

Εφαρμογή 26

Συμβάντα με Πλαίσια Ελέγχου και Λίστες Επιλογής

Τα πλαίσια ελέγχου και οι λίστες επιλογής απαιτούν το ItemListener interface. Για να κάνουμε ένα απ' αυτά τα συστατικά να δημιουργεί συμβάντα, χρησιμοποιούμε την μέθοδο addItemListener(). Οι επόμενες εντολές δημιουργούν ένα πλαίσιο ελέγχου με όνομα visaCard και το κάνουν να στέλνει συμβάντα χρήστη κάθε φορά που κάνουμε κλικ πάνω του.

```
Checkbox visaCard = new Checkbox("Visa Card", true);
visaCard.addItemListener(this);
```

Τα συμβάντα λαμβάνονται από την μέθοδο itemStateChanged(), η οποία παίρνει σαν όρισμα ένα αντικείμενο ItemEvent. Για να βρούμε ποιο αντικείμενο προκάλεσε το συμβάν, χρησιμοποιούμε την μέθοδο getItem().

Για να βρούμε αν ένα πλαίσιο ελέγχου είναι επιλεγμένο ή όχι, χρησιμοποιούμε την μέθοδο getStateChange() με τις σταθερές ItemEvent.SELECTED και ItemEvent.DESELECTED, ως εξής :

```
int status = visaCard.getStateChanged();
if (status = = ItemEvent.SELECTED)
{
      // το στοιχείο είναι επιλεγμένο
}
else
{
      // το στοιχείο δεν είναι επιλεγμένο
}
```

Για να βρούμε την τιμή που έχει επιλεγεί σ' ένα αντικείμενο λίστας, χρησιμοποιούμε την μέθοδο getItem() και μετατρέπουμε την τιμή που επιστρέφει σε string, ως εξής :

```
Object which = visaCard.getItem();

String answer = (String) which;
```

Εφαρμογή 27

Συμβάντα με Πλαίσια Κειμένου

Αν θέλουμε να ελέγξουμε αν έχει λάβει χώρα ένα συμβάν σ' ένα πλαίσιο κειμένου, όπως αλλαγή της τιμής του, χρησιμοποιούμε το TextListener interface. Η μέθοδος addTextListener() κάνει ένα συστατικό κειμένου να μπορεί να δημιουργεί συμβάντα χρήστη και η μέθοδος textValueChanged() λαμβάνει τα συμβάντα, ως εξής :

```
public void textValueChanged(TextEvent txt)
{
      Object source = txt.getSource();
```

```
        if (source = = address)
        {
            String newAddress = address.getText();
        }
        else
        {
            String newZipCode = zipCode.getText();
        }
    }
```

Εφαρμογή 28

Γράψτε και εκτελέστε ένα πρόγραμμα Java, το οποίο να δίνει ως αρχική τιμή σε ένα αντικείμενο String το μικρό σας όνομα και στην συνέχεια να το τυπώνει σε τρεις διαφορετικές γραμμές.

Λύση:

```
public class PrintName
   { public static void main (String[] args)
     { String name = “Αθηνά”;
       System.out.println (name) ;
       System.out.println (name) ;
       System.out.println (name) ;
        }
}
```

Εφαρμογή 29

Γράψτε και εκτελέστε ένα πρόγραμμα Java, το οποίο να δίνει ως αρχική τιμή σε ένα αντικείμενο String το μικρό σας όνομα και στην συνέχεια να το τυπώνει τρεις φορές στην ίδια γραμμή, με ενδιάμεσα κενά, όπως στο επόμενο παράδειγμα

Αθηνά Αθηνά Αθηνά

Λύση:

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

```
{ String name = “Αθηνά”;
  System.out.println (name+” “+name+” “+name);
}
     }
```

Εφαρμογή 30

Γράψτε και εκτελέστε ένα πρόγραμμα Java, το οποίο να ζητάει από τον χρήστη ξεχωριστά το επώνυμο και το μικρό του όνομα και στην συνέχεια να τυπώνει έναν χαιρετισμό όπως ο επόμενος

Πληκτρολόγησε το επώνυμό σου: Χατζάκη

Πληκτρολόγησε το μικρό σου όνομα: Αθηνά

Γεια σου, Αθηνά Χατζάκη

Τα έντονα γράμματα υποδηλώνουν την είσοδο του χρήστη.

Λύση:

```
public class PrintName
   { public static void main (String[] args)
     { final int LEN = 20 ;
       byte[] buffer1 = new byte [LEN] ;
       System.out.print (“Πληκτρολόγησε το επώνυμό σου: ”);
       try ( System.in.read(buffer1, 0, LEN) ; )
       catch (Exception e) {}
       String lname = new String(buffer1) ;
       byte[] buffer2 = new byte [LEN] ;
       System.out.print (“Πληκτρολόγησε το μικρό σου όνομα: ”);
       try ( System.in.read(buffer2, 0, LEN) ; )
       catch (Exception e) {}
       String fname = new String(buffer2) ;
System.out.println(“Γεια σου, “ + fname.trim() + “ “ + lname.trim()) ;
     }
   }
```

Εφαρμογή 31

Να γίνει πρόγραμμα StringUpper.java το οποίο θα έχει πίνακα με 10 String με Αγγλικές λέξεις. Το πρόγραμμα θα πρέπει να διατρέχει τον πίνακα και να μετατρέπει κάθε String στο array αυτό σε ένα String με κεφαλαία γράμματα. Να χρησιμοποιήσετε τη συνάρτηση toUpperCase της τάξης String.

Λύση:

```
class StringUpper {
  public static void main(String[] args) {
    String[] arrayOfWords = {  "backspace", "deletes",
                               "the", "character", "to",
                               "the", "left", "of", "the", "cursor"};
    for (int i=0; i<arrayOfWords.length; i++) {
      arrayOfWords[i] = arrayOfWords[i].toUpperCase();
      System.out.println(arrayOfWords[i]);
    }
  }
}
```

Εφαρμογή 32

Να γίνει πρόγραμμα SortArray.java το οποίο χρησιμοποιεί τον αλγόριθμο της ταξινόμησης με τη μέθοδο της φυσαλίδας για την αύξουσα ταξινόμηση ενός πίνακα 10 ακεραίων.

Λύση:

```
class SortArray {
  public static void main(String[] args)
  {
    int[] a = {10, 5, 109, 200, 8, 764, 32, 87, 87, 9};

    for (int i=0; i<a.length-1; i++)
      for (int j=i+1; j<a.length; j++)
        if (a[i] > a[j]) {
          int k = a[i];
          a[i] = a[j];
```

```
            a[j] = k;
          }

    //Emfanisi tou apotelesmatos
    for (int i=0; i<a.length; i++)
      System.out.println("a["+i+"] = "+a[i]);
  }
}
```

Εφαρμογή 32

Να γίνει πρόγραμμα SeekNumber.java το οποίο χρησιμοποιεί τον αλγόριθμο της δυαδικής αναζήτησης για να αναζητήσει ένα ακέραιο που δίνεται σαν παράμετρος στο πρόγραμμα σε ένα πίνακα ακεραίων ο οποίος είναι ταξινομημένος.

Λύση:

```
class SeekNumber {
  public static void main(String[] args)
  {
    int a[] = { 1, 9, 10, 23, 45, 67, 90, 95, 105, 180 };
    int left, right, mid=0;
    if (args.length != 1) {
      System.out.println("Usage: java SeekNumber int");
      System.exit(0);
    }
    int x = Integer.parseInt(args[0]);
    left = 0;
    right = a.length-1;
    boolean found = false;
    while (!found && (left<=right))
    {
      mid = (left+right)/2;
      if (a[mid] == x)
        found = true;
      else
        if (a[mid] < x)
```

```
            left = mid + 1;
          else
            right = mid - 1;
      }
      if (found)
        System.out.println(x+" was found in position a["+mid+"]");
      else
        System.out.println(x+" was not found");
    }
}
```

Εφαρμογή 33

Να γίνει μία τάξη *Book* η οποία θα χρησιμοποιηθεί για τα βιβλία μιας βιβλιοθήκης. Για κάθε βιβλίο να δηλώσετε τις μεταβλητές *title* (*String)* με τον τίτλο του βιβλίου, *author* (επίσης *String)* για τον συγγραφέα και *firstEdition (int)* που είναι το έτος της πρώτης έκδοσης. Όλες οι μεταβλητές θα πρέπει να είναι ιδιωτικές *(private).* Να δώσετε μεθόδους *get* και *set* για όλες τις ιδιότητες που αναφέρθηκαν παραπάνω. Στη συνέχεια δημιουργείστε ένα πρόγραμμα το οποίο θα δημιουργήσει ένα πίνακα τριών βιβλίων. Τέλος η *main* θα πρέπει να ταξινομήσει τον πίνακα αυτών των βιβλίων χρησιμοποιώντας οποιαδήποτε μέθοδο ταξινόμησης θέλετε, με αύξουσα σειρά ως προς το έτος πρώτης έκδοσης. Αφού η *main* κάνει τη ταξινόμηση θα πρέπει να εμφανίσει τα τρία βιβλία με ένα String του στυλ "Βιβλίο: <όνομα-βιβλίου>, Συγγραφέας: <όνομα-συγγραφέα>, Έτος πρώτης έκδοσης: <έτος-πρώτης-έκδοσης>.

Λύση:

```
public class Main
{
        public static void main (String[] args)
        {
                //Dimioyrgia toy pinaka 3 books
                Book books[] = new Book[3];
                for (int i=0; i<3; i++)
                        books[i] = new Book();
```

```
        //Stoixeia gia to 1o vivlio
        books[0].setAuthor("Jaworski Jamie");
        books[0].setTitle("Java 2 Unleashed");
        books[0].setFirstEdition(1999);

        //Stoixeia gia to 2o vivlio
        books[1].setAuthor("Stroustrup Bjarne");
        books[1].setTitle("The C++ Programming Language");
        books[1].setFirstEdition(1991);

        //Stoixeia gia to 3o vivlio
        books[2].setAuthor("Szyperski Clemens");
        books[2].setTitle("Component Software - Beyond O-O Programming");
        books[2].setFirstEdition(1997);

        //Taxinomisi twn vivliwn me ti bubble sort
        int count = books.length;
        for (int i=1; i<count; i++)
                for (int j=count-1; j>=i; j--)
                        //An xreiazetai kane swap
                        if (books[j-1].getFirstEdition() >
                                books[j].getFirstEdition()) {
                                Book tempBook = books[j-1];
                                books[j-1] = books[j];
                                books[j] = tempBook;
                        }

        //emfanise ta vivlia
        for (int i=0; i<books.length; i++)
                System.out.println("Syggrafeas: "+books[i].getAuthor()+
                                                ", Vivlio: "+books[i].getTitle()+
                                                ", Etos Prwtis Ekdosis:
"+books[i].getFirstEdition());
        }
```

}

Εφαρμογή 34

Στην προηγούμενη άσκηση η τάξη *Book* έχει την ιδιότητα *Author* για τον συγγραφέα, την οποία προηγουμένως υλοποιήσαμε ως *String.* Να δηλώσετε μία τάξη *Author* η οποία θα έχει την ιδιότητα *name* για το όνομα του συγγραφέα που θα είναι *String* και *get* και *set* μεθόδους γι' αυτή την ιδιότητα. Να συμπεριλάβετε στην τάξη του συγγραφέα και ένα πίνακα 20 το πολύ βιβλίων στον οποίο θα τοποθετήσουμε τα βιβλία του συγγραφέα. Τα βιβλία θα είναι αντικείμενα της τάξης *Book* της προηγούμενης άσκησης. Από την τάξη *Book* της προηγούμενης άσκησης να αλλάξετε την ιδιότητα *Author* και τις μεθόδους *setAuthor* και *getAuthor* έτσι ώστε να είναι και να επενεργούν πάνω σε ένα αντικείμενο τύπου *Author.*
Για την τάξη *Author* θα πρέπει επίσης να συμπεριλάβετε δύο μεθόδους:

- Την *public void setter(int index, Book b) throws ArrayIndexOutOfBoundsException* η οποία θα θέτει στην θέση *index* του πίνακα των βιβλίων ενός συγγραφέα το βιβλίο που δίνεται σαν δεύτερη παράμετρο. Η μέθοδος θα πρέπει να προκαλεί την εξαίρεση *ArrayIndexOutOfBoundsException* στην περίπτωση που η θέση *index* είναι εκτός των ορίων του πίνακα.
- Την *public Book[] getter()* η οποία επιστρέφει τον πίνακα των βιβλίων του συγγραφέα.

Αφού δηλώσετε τις ποιο πάνω τάξεις κάνετε πρόγραμμα στο οποίο θα δημιουργήσετε ένα αντικείμενο τύπου *Author* με τον αγαπημένο σας συγγραφέα. Στη συνέχεια θα δημιουργήσετε δύο βιβλία του και θα κάνετε τις απαραίτητες συσχετίσεις. Τέλος θα εμφανίσετε λίστα με τα βιβλία του συγγραφέα, χρησιμοποιώντας την *getter* μέθοδο για τον συγγραφέα αυτό.
Λύση:

```
public class Main
{

    public static void main (String[] args)
    {
        //Dimiourgia enos siggrafea
```

```
        Author booch = new Author();
        booch.setName("Booch G.");

        //Dimiourgia tou 1ou vivliou
        Book b1 = new Book();
        b1.setTitle("Software Components with Ada");
        b1.setFirstEdition(1987);

        //Dimiourgia tou 2ou vivliou
        Book b2 = new Book();
        b2.setTitle("Object Oriented Analysis and Design, 2nd Ed.");
        b2.setFirstEdition(1994);

        //Sysxetisi toy siggrafea me ta vivlia toy
        booch.setter(0, b1);
        booch.setter(1, b2);

        //Sysxetisi toy vivlioy me ton siggrafea toy
        b1.setAuthor(booch);
        b2.setAuthor(booch);
        //Emfanisi twn vivliwn toy siggrafea
        Book b[] = booch.getter();
        for (int i=0; i<b.length; i++)
            if (b[i] != null)
                System.out.println(b[i].getTitle());
    }
}
```

Εφαρμογή 35

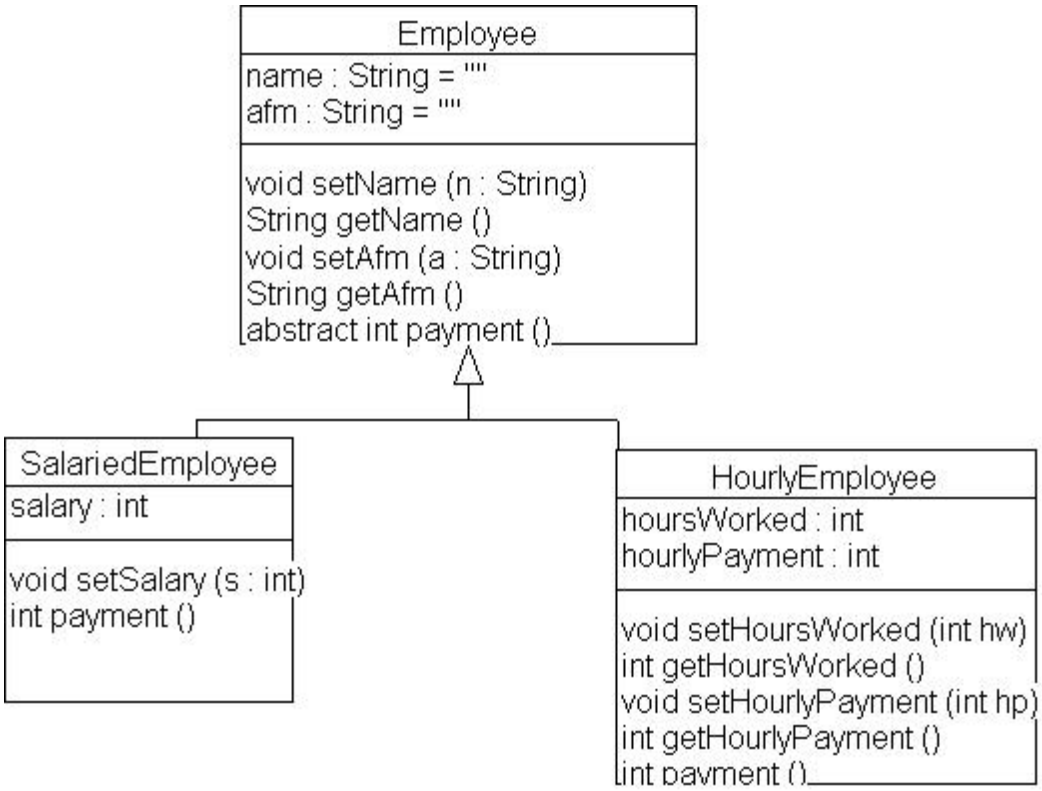
Να γίνει *Interface CryingObject* το οποίο θα διαθέτει τη μέθοδο *public void cry()*. Στη συνέχεια δημιουργείστε την τάξη *Baby* που θα υλοποιεί το *Interface* αυτό. Η μέθοδος *cry* για το *Baby* απλά θα εμφανίζει στην οθόνη το μήνυμα "Crying all night long!". Επίσης δημιουργείστε τάξη *BluesSinger* η οποία επίσης θα υλοποιεί το

InterfaceCryingObject και της οποίας η μέθοδος θα εμφανίζει το μήνυμα "Crying when singing!".
Στη συνέχεια δημιουργείστε τάξη *Main* με μέθοδο *main* η οποία θα δημιουργεί πίνακα τύπου *CryingObject* με δύο θέσεις . Στη πρώτη θέση τοποθετείστε ένα νέο αντικείμενο της τάξης *Baby* ενώ στη δεύτερη τοποθετείστε ένα αντικείμενο της τάξης *BluesSinger*. Τέλος, η *main* θα εκτελεί βρόχο (loop) στον οποίο θα καλεί τη μέθοδο *cry* σε όλα τα αντικείμενα του πίνακά της.
Λύση:

```
public class Main
{
	public static void main (String[] args)
	{
		CryingObject co[] = new CryingObject[2];
		//dimiourgia toy Baby
		Baby b = new Baby();

		//dimiourgia toy BluesSinger
		BluesSinger bs = new BluesSinger();

		//topothetisi twn antikeimenwn ston pinaka
		co[0] = b;
		co[1] = bs;
		for (int i=0; i<co.length; i++)
		co[i].cry();
	}
}
```

Εφαρμογή 36

Να γίνει πρόγραμμα με τις ακόλουθες τάξεις όπως φαίνονται στο UML διάγραμμα που δίνεται:



Η τάξη *Employee* είναι υπερτάξη των τάξεων *SalariedEmployee* και *HourlyEmployee.* Η υπερτάξη έχει τις ιδιότητες *name* (όνομα) και *afm* (ΑΦΜ) και μεθόδους για να θέτουμε και να ανακτούμε τις ιδιότητες αυτές. Επίσης έχει την αφηρημένη (abstract) μέθοδο *payment()*. Επομένως και η τάξη *Employee* θα πρέπει να δηλωθεί ως *abstract*.

Η υποτάξη *SalariedEmployee (τακτικός υπάλληλος)* έχει επίσης την ιδιότητα *salary* και μία *set* μέθοδο για να θέτει αυτή την ιδιότητα. Επίσης η μέθοδος δηλώνει την αφηρημένη τάξη *payment* της υπερτάξης έτσι ώστε να επιστρέφει τον *salary*.

Η υποτάξη *HourlyEmployee (ωρομίσθιος υπάλληλος)* έχει επίσης τις ιδιότητες *hoursWorked* (ώρες εργασίας) και *hourlyPayment* (ωρομίσθιο). Επίσης δηλώνει τις μεθόδους *get* και *set* γι' αυτές τις ιδιότητες.Τέλος, δηλώνει την μέθοδο *payment* που είναι ο ορισμός της *abstract* μεθόδου της υπερτάξης. Η μέθοδος αυτή θα επιστρέφει το γινόμενο των ωρών εργασίας επί του ωρομισθίου.

Να κατασκευάσετε την πιο πάνω ιεραρχία τάξεων και στη συνέχεια να κάνετε μία τάξη *Main* με μία συνάρτηση *main.* Η *main* θα έχει ένα πίνακα δυο θέσεων ο οποίος θα είναι τύπου *Employee*. Στην θέση 0 του πίνακα να δημιουργήσετε και να βάλετε ένα αντικείμενο *SalariedEmployee* και στη θέση 1 του πίνακα να δημιουργήσετε και να βάλετε ένα αντικείμενο *HourlyEmployee*. Στη συνέχεια να θέσετε τα στοιχεία του

πρώτου υπαλλήλου και τον μισθό του και τα στοιχεία του δεύτερου υπαλλήλου συμπεριλαμβανομένων και των ωρών εργασίας και του ωρομισθίου. Τέλος, η *main* θα πρέπει να εκτελεί ένα βρόχο στον οποίο θα εμφανίσει τον μισθό των δύο υπαλλήλων καλώντας τη μέθοδο *payment* στα αντικείμενα του πίνακά της.

Λύση:

```
public class Main
{
	public static void main (String[] args)
	{
		Employee emp[] = new Employee[2];

		//O prwtos ypalilos plirwnetai me to mina
		emp[0] = new SalariedEmployee();
		emp[0].setName("Mikelis");
		emp[0].setAfm("777777");
		((SalariedEmployee) emp[0]).setSalary(300000);

		//O deyteros ypallilos einai oromisthios
		emp[1] = new HourlyEmployee();
		emp[1].setName("Karamitros");
		emp[1].setAfm("888888");
		((HourlyEmployee) emp[1]).setHourlyPayment(4000);
		((HourlyEmployee) emp[1]).setHoursWorked(30);

		/*Twra anatrexoume ton pinaka kai kaloyme tin
		  methodo payment() se kathe employee gia na
		  emfanisoyme tin plirwmi toy
		*/
		for (int i=0; i<emp.length; i++)
			System.out.println("Employee's #"+i+
							" salary is "+
							emp[i].payment());
	}
}
```

ΑΠΑΝΤΗΣΕΙΣ ΑΣΚΗΣΕΩΝ

ΕΠΑΝΑΛΗΨΗ 3^{ΟΥ} ΚΕΦΑΛΑΙΟΥ

Άσκηση 1:

Το μήνυμα που θα στέλναμε σε αυτή την περίπτωση είναι:

κινητό_ξυπνητήρι.ενεργοποίηση(7,30,0);

Ενώ για την επανάληψη του χτυπήματος 2,5 λεπτά αργότερα θα γράψουμε:

κινητό_ξυπνητήρι.επανάληψη(7,32,30);

Σημείωση: Οι εντολές ενεργοποίηση και επανάληψη είναι προαιρετικές. Μπορείτε να χρησιμοποιήσετε οποιεσδήποτε άλλες λέξεις σας εκφράζουν.

Άσκηση 2:

Κόκκινο (255,0,0), πράσινο (0,255,0) και μπλε (0,0,255) ενώ για το μωβ ο συνδυασμός είναι (255,0,255) αφού το μωβ είναι μείξη του κόκκινου με το μπλε.

Άσκηση 3:

```
public class Book
{
private String title;
private String author;
private int firstEdition;
public Book()
{
title = "";
author="";
firstEdition=0;
}

public void setTitle(String t)
        {
                title = t;
```

```
	}

	public String getTitle()
	{
		return title;
	}

	public void setAuthor(String a)
	{
		author = a;
	}
	public String getAuthor()
	{
		return author;
	}

	public void setFirstEdition(int fe)
	{
		firstEdition = fe;
	}

	public int getFirstEdition()
	{
		return firstEdition;
	}
}
```

Άσκηση 4.

```
public class Book
{
		private String title;
		private int firstEdition;
		private Author author;
```

```
        public Book()
        {
                title = "";
                author=null;
                firstEdition=0;
        }

public void setTitle(String t)
{
        title = t;
}

public String getTitle()
{
        return title;
}

public void setAuthor(Author a)
{
        author = a;
}

public Author getAuthor()
{
        return author;
}

public void setFirstEdition(int fe)
{
        firstEdition = fe;
}

public int getFirstEdition()
{
```

```
                return firstEdition;
        }
}
```

Άσκηση 5.

```
public class BluesSinger implements CryingObject
{
        public void cry()
        {
                System.out.println("Crying when singing!");
        }
}
```

Άσκηση 6.

```
public class Main
{
        public static void main (String[] args)
        {
                Employee emp[] = new Employee[2];

                //O prwtos ypalilos plirwnetai me to mina
                emp[0] = new SalariedEmployee();
                emp[0].setName("Mikelis");
                emp[0].setAfm("777777");
                ((SalariedEmployee) emp[0]).setSalary(300000);

                //O deyteros ypallilos einai oromisthios
                emp[1] = new HourlyEmployee();
                emp[1].setName("Karamitros");
                emp[1].setAfm("888888");
                ((HourlyEmployee) emp[1]).setHourlyPayment(4000);
                ((HourlyEmployee) emp[1]).setHoursWorked(30);

                /*Twra anatrexoume ton pinaka kai kaloyme tin
```

```
                methodo payment() se kathe employee gia na
                emfanisoyme tin plirwmi toy
              */
              for (int i=0; i<emp.length; i++)
                    System.out.println("Employee's #"+i+
                                                  " salary is "+
                                                  emp[i].payment());
        }
}
```

ΕΠΑΝΑΛΗΨΗ 4^{ΟΥ} ΚΕΦΑΛΑΙΟΥ

Άσκηση 1.

Το αποτέλεσμα που θα βγάλει το NetBeans θα πρέπει να είναι:

X=120

Y= 25

K=95

Άσκηση 2.

Το πρόγραμμα που θα γράψετε θα είναι ως εξής:

```
public class EmvadonTetragonou
 {
 public static void main(String [] args)
{
short a;
 a=5;
System.out.println("a is " + a);
 e= (short) (a*a);
 System.out.println("e is " + e);
 }
}
```

Άσκηση 3.

Το πρόγραμμα για τον υπολογισμό της περιμέτρου θα είναι

```
public class PerimetrosOrthogoniou
 {
 public static void main(String [] args)
{
short a, b;
 a=5;
System.out.println("a is " + a);
 b=10;
System.ut.println("b is " + b);
 e= (short) (2*a+2*b);
 System.out.println("e is " + e);
 }
}
```

Άσκηση 4

```
class StringUpper {
   public static void main(String[] args) {
      String[] arrayOfWords = {  "backspace", "deletes",
                                 "the", "character", "to",
                                 "the", "left", "of", "the", "cursor"};
      for (int i=0; i<arrayOfWords.length; i++) {
         arrayOfWords[i] = arrayOfWords[i].toUpperCase();
         System.out.println(arrayOfWords[i]);
      }
   }
}
```

Άσκηση 5

```
class SortArray {
   public static void main(String[] args)
   {
      int[] a = {10, 5, 109, 200, 8, 764, 32, 87, 87, 9};
```

```
    for (int i=0; i<a.length-1; i++)
      for (int j=i+1; j<a.length; j++)
        if (a[i] > a[j]) {
          int k = a[i];
          a[i] = a[j];
          a[j] = k;
        }

    //Emfanisi tou apotelesmatos
    for (int i=0; i<a.length; i++)
      System.out.println("a["+i+"] = "+a[i]);
  }
}
```

Άσκηση 6

```
class SeekNumber {
  public static void main(String[] args)
  {
    int a[] = { 1, 9, 10, 23, 45, 67, 90, 95, 105, 180 };
    int left, right, mid=0;
    if (args.length != 1) {
      System.out.println("Usage: java SeekNumber int");
      System.exit(0);
    }

    int x = Integer.parseInt(args[0]);
    left = 0;
    right = a.length-1;
    boolean found = false;
    while (!found && (left<=right))
    {
      mid = (left+right)/2;
      if (a[mid] == x)
        found = true;
```

```
        else
          if (a[mid] < x)
            left = mid + 1;
          else
            right = mid - 1;
      }
      if (found)
        System.out.println(x+" was found in position a["+mid+"]");
      else
        System.out.println(x+" was not found");
    }
}
```

ΕΠΑΝΑΛΗΨΗ 6^{ΟΥ} ΚΕΦΑΛΑΙΟΥ

Άσκηση 1:

Υποδειγματική Λύση

```
import java.io.*;
public class review1 {
/**
* @param args
*/
public static void main(String[] args)throws IOException {
String names [] = {"Alex","Tom","Mike","Nick","Maria"};
System.out.println("1.)Print the length of the
table\nLength of table is : "+names.length+"\n\n2.)Print the data of
the table");
for (int i=0; i<names.length; i++)
{
System.out.println("Id "+i+" of table has :
"+names[i]);
}
System.out.println("\n3.)Sort the data of the table and
```

```
print");
for (int i=0; i<names.length-1; i++)
{
for(int j=i+1; j<=names.length-1; j++)
{
if (names[i].compareTo(names[j])>0)
{
String tmp=names[i];
names[i]=names[j];
names[j]=tmp;
}
}
}
for (int i=0; i<names.length; i++)
{
System.out.println("Id "+i+" of table has :
"+names[i]);
}
System.out.println("\n * * * End of program * * *\n");
}
}
```

Άσκηση 2:

Υποδειγματική Λύση

```
import java.util.Vector;
/**
* <p>Title: </p>
*
* <p>Description: </p>
*
* <p>Copyright: Copyright (c) 2007</p>
*
* <p>Company: </p>
*
```

- @author not attributable

Άσκηση 3:

Υποδειγματική Λύση

```
import java.util.Vector;
/**
* <p>Title: </p>
*
* <p>Description: </p>
*
```

- Copyright: Copyright (c) 2007

Άσκηση 4:

Υποδειγματική Λύση

```
/*
```

size(): Επιστρέφει το πλήθος των στοιχείων του Vector.

elementAt(int index): Επιστρέφει το στοιχείο που βρίσκεται στη θέση index.

firstElement(): Επιστρέφει το πρώτο στοιχείο του vector.

lastElement():Επιστρέφει το τελευταίο στοιχείο του vector.

removeElementAt(int index): Διαγράφει το στοιχείο που βρίσκεται στη θέση index.

hasMoreElements(): Ελέγχει εάν ο πίνακας έχει και άλλα στοιχεία ή όχι

nextElement(): Επιστρέφει το επόμενο στοιχείο

```
*/
import java.util.*;
public class VectorDemo{
public static void main(String[] args){
Vector<Object> vector = new Vector<Object>();
int primitiveType = 10;
Integer wrapperType = new Integer(20);
String str = "tapan joshi";
vector.add(primitiveType);
vector.add(wrapperType);
vector.add(str);
vector.add(2, new Integer(30));
System.out.println("the elements of vector: " +
```

```
vector);
System.out.println("The size of vector are: " +
vector.size());
System.out.println("The elements at position 2 is: "
+ vector.elementAt(2));
System.out.println("The first element of vector is: "
+ vector.firstElement());
System.out.println("The last element of vector is: "
+ vector.lastElement());
vector.removeElementAt(2);
Enumeration e=vector.elements();
System.out.println("The elements of vector: " +
vector);
while(e.hasMoreElements()){
System.out.println("The elements are: " +
e.nextElement());
}
}
}
```

ΛΕΞΙΛΟΓΙΟ

ΕΛΛΗΝΙΚΟΙ ΟΡΟΙ

Αντικειμενοστρέφεια: Η Java ακολουθεί τη παράδοση της Smalltalk, μια γλώσσα που απαιτεί τη χρήση αντικειμένων και μόνο, σε αντίθεση με άλλες που επιτρέπουν στον προγραμματιστή να ελιχθεί χρησιμοποιώντας πιο κλασσικές μεθόδους στο ίδιο πρόγραμμα.

Αριθμοί και σύμβολα: Οι αριθμοί ξεκινούν με ένα από τα δέκα ψηφία.Οι ακέραιοι, όπως 1,1432 και 1134224, αντιμετωπίζονται ως ακέραιοι χωρίς κάποιους επιπρόσθετους χαρακτήρες.

Οι μεγάλοι ακέραιοι, οι αριθμοί κινητής υποδιαστολής και οι οκταδικές και δεκαεξαδικές τιμές χρειάζονται επιπρόσθετα γράμματα για να ερμηνευθούν από τον compiler.Για να τα αναγνωρίσει πρέπει να χρησιμοποιήσει το γράμμα f μετά τον αριθμό πχ x=3,1415f.

Οι αριθμοί που γράφονται με εκθετικό τρόπο χρησιμοποιούν είτε το γράμμα e είτε το E πχ x=6,023e23f.Οι μεγάλοι ακέραιοι ακολουθούνται από ένα l ή L πχ longvar=132312312412L.Οι δεκαεξαδικοί ξεκινούν με 0x ή 0X και έχουν τη μορφή x=0xDEAD14.Οι οκταδικοί ξεκινούν με ένα 0.

Τύπος	**Μέγεθος**	**Εύρος**	**Σύνταξη**
byte	8 bits	-2 ως 2 -1	121
short	16 bits	-2 ως 2 -1	4023
int	32 bits	-2 ως 2 -1	412235
long	64 bits	-2 ως 2 -1	622312882881
float	xx bits	-άπειρο ως άπειρο	6,023e23i
double	xx bits	-άπειρο ως άπειρο	1,23414234324D

Στην Java, ο αριθμός των bits που χρησιμοποιούνται για έναν αριθμό τύπου int είναι σταθερός. Υπάρχουν επίσης τιμές με τη μορφή γραμμάτων, οι οποίες

χρησιμοποιούνται με χαρακτήρες(σε απλά εισαγωγικά) και συμβολοσειρές(σε διπλά εισαγωγικά).
Ειδικοί χαρακτήρες:

Σύμβολο	Χαρακτήρας	Σύμβολο	Χαρακτήρας
b	backspace	‘	απλά εισαγωγικά
n	αλλαγή γραμμής	“	διπλά εισαγωγικά
r	carriage return	\	ανάποδη κάθετος
t	στηλοθέτης	u	έκδοση
f	form feet		

Μερικές συμβολοσειρές μπορεί να έχουν την παρακάτω μορφή:
message =Bob was ‘strange’;
message=Name t Rank t Serial Number;
message=Macintosh is a trademark(2122)of Apple.;

Στο πρώτο παράδειγμα τοποθετούνται απλά εισαγωγικά εντός της συμβολοσειράς, στο δεύτερο χρησιμοποιούνται στηλοθέτες για τη δημιουργία επικεφαλίδων στηλών και στο τρίτο ένας χαρακτήρας UNICODE για το σήμα κατατεθέν.

Αποτελεσματικότητα: Η Java σχεδιάστηκε έτσι, ώστε να τρέχει σε μηχανές που δε διαθέτουν μεγάλη μνήμη .Έτσι 4 Mbytes μνήμης θεωρούνται αρκετά για να την υποστηρίξουν μαζί με ένα ελαφρύ λειτουργικό σύστημα.

Δυναμική διασύνδεση: Η Java συνδέεται δυναμικά που σημαίνει ότι δεν αναζητά μια μέθοδο προτού χρειαστεί να την εκτελέσει. Το χαρακτηριστικό αυτό αποδεικνύεται ιδιαίτερα χρήσιμο, διότι επιτρέπει στα προγράμματα της Java να χρησιμοποιούν μεθόδους που βρίσκονται αποθηκευμένες τοπικά στους browsers.

Επεκτασιμότητα: Η Java μπορεί να συνδεθεί με τμήμα πηγαίου κώδικα που βρίσκεται τοπικά, σε γλώσσες όπως η C.Αυτό σημαίνει ότι έχει μεγάλες πιθανότητες να αναδειχτεί σε γλώσσα γενικής χρήσης.

Ισχυρός καθορισμός τύπων: Σε κάθε σημείο του κώδικα πρέπει να καθορίζεται ο τύπος του αντικειμένου του οποίου γίνεται χρήση.

Ταχύτητα: Η Java είναι πολύ γρήγορη από οποιαδήποτε γλώσσα τύπου Script διότι ο κώδικας έχει ήδη εν μέρει μεταγλωττιστεί. Η καθυστέρηση στην εκτέλεση οφείλεται κυρίως στη σύνδεση καις το compilation του byte code.Ο κώδικας σε C είνα συνήθως κατά 20 φορές πιο γρήγορος .

Ονόματα: Τα βασικά στοιχεία κατασκευής κάθε προγράμματος είναι τα ονόματα που δίνονται στις κλάσεις, στις μεταβλητές και στους αριθμούς που χρησιμοποιούνται ως σταθερές.

Τα ονόματα στη Java μπορούν να αποτελούνται από οποιονδήποτε συνδυασμό χαρακτήρων και αριθμών, ο οποίος ξεκινά είτε με κάποιο γράμμα είτε με το σύμβολο του δολαρίου ή της υπογράμμισης.

Η Java ξεχωρίζει τα κεφαλαία από τα πεζά γράμματα. Μερικοί Java compilers αναγνωρίζουν το πρότυπο UNICODE, μια προέκταση 16-bit του προτύπου 8- bit ACII, το οποίο σχεδιάστηκε για να περιλαμβάνει σύμβολα για όλες τις γλώσσες του κόσμου.

Τη στιγμή της συγγραφής , οι περισσότεροι χρήστες της Java δημιουργούν τα προγράμματα τους χρησιμοποιώντας απλά προγράμματα επεξεργασίας κειμένου, όπως ο emacs, το BBEdit ή ακόμα και το Microsoft Write.

Πολλοί προγραμματιστές της Java χρησιμοποιούν κάποιες τυποποιημένες συμβάσεις, όταν διαλέγουν τα ονόματα για τις μεταβλητές τους. Οι κλάσεις ξεκινούν με κεφαλαίο γράμμα ενώ οι μέθοδοι και οι μεταβλητές με πεζό.

Υπάρχουν 47 λέξεις οι οποίες είναι δεσμευμένες από τον compiler για την υπόδειξη λειτουργιών και μπορούν να χρησιμοποιηθούν ως ονόματα κλάσεων, μεταβλητών ή σταθερών.

- abstract Κλάσεις και μεθόσοι μπορούν να οριατούν ως αφηρημένες, αν ο καθορισμός κάποιας μεθόδου δεν βρίσκεται στην τρέχουσα κλήση.
- char Ένας από τους βασικούς τύπους που χρησιμοποιούνται για τη τοποθέτηση χαρακτήρων.
- byte Ένας από τους βασικούς τύπους.Κρατάει 8 bits

- int Ένας από τους βασικούς τύπους
- if Η πρώτη λέξη κλειδί σε μια λειτουργία απόφασης με διακλαδώσεις
- long Ένας από τους βασικούς τύπους.Ένας μεγαλύτερος ακέραιος.
- null Ο μηδενικός δείκτης.
- for Η έναρξη του απλού βρόχου
- finally Τμήμα του ορισμού case.
- switch Τμήμα μιας πρότασης case.

ΑΓΓΛΙΚΟΙ ΟΡΟΙ

Animation: *Java*.Για να σχεδιάσετε μέσα σε πλάισιο, στο εσωτερικό μιας σελίδας στον browser.Η δυνατότητα αυτή είναι αναγκαία , προκειμένου να δημιουργήσετε κινούμενες εικόνες.

Applet : Είναι ένα μικρό πρόγραμμα, που είναι σχεδιασμένο για να εκτελείται μέσα από μια άλλη εφαρμογή. Σε αντίθεση με τις εφαρμογές, τα applets δεν μπορούν να εκτελεσθούν απευθείας από το λειτουργικό σύστημα.

Τα applets γράφονται στη γλώσσα προγραμματισμού Java, είναι συνήθως μικρά σε μέγεθος και προορίζονται για να ενσωματωθούν σε μια ιστοσελίδα. Ο κώδικας τους περιλαμβάνεται ανάμεσα στα tags <applet> ... </applet>. Ο αρχικός κώδικας είναι γραμμένος σ' ένα αρχείο με επέκταση **.java**, γίνεται μεταγλώττιση (compilation) και προκύπτει ένα αρχείο με το ίδιο όνομα αλλά με την επέκταση **.class**, το οποίο χρησιμοποιούμε στα tags <applet> της HTML.

Ο χρήστης δεν μπορεί να δει τον πρωτογενή κώδικα ενός applet, αλλά μόνο να το εκτελέσει. Ένα applet μπορεί επίσης να δεχθεί και ορισμένες παραμέτρους, όπως τον κώδικα (.class) που θα πρέπει να εκτελέσει (code), το πλάτος (width) και το ύψος (height) του παραθύρου που θα δημιουργηθεί μέσα στην ιστοσελίδα για την εκτέλεση του applet κ.ά.

```
<body>
   <applet code=clicker.class width=600 height=400>
   </applet>
</body>
...
<body>
   <applet code="fprotate.class" codebase="_fpclass/" width="282" height="200">
     <param name="rotatoreffect" value="dissolve">
     <param name="time" value="3">
     <param name="url" value valuetype="ref">
     <param name="image1" value="small-florina1.jpg" valuetype="ref">
     <param name="image2" value="small-florina2.jpg" valuetype="ref">
     <param name="image3" value="small-florina3.jpg" valuetype="ref">
   </applet>
</body>
```

Application: Αποδίδεται στα ελληνικά με τον όρο εφαρμογή και είναι ένα πρόγραμμα ή ομάδα προγραμμάτων που είναι σχεδιασμένα για τους τελικούς χρήστες (end users).

AWT (Abstract Windows Toolkit): Είναι το API της Java API το οποίο δίνει τη δυνατότητα στους προγραμματιστές να αναπτύξουν εφαρμογές σε Java με συστατικά GUI, όπως παράθυρα (windows), πλήκτρα (buttons) και γραμμές κύλισης (scroll bars).

Browser: Είναι γνωστό και ως Web browser και αποδίδεται στα ελληνικά με τον όρο Φυλλομετρητής ή Πρόγραμμα Περιήγησης Ιστοσελίδων ή και Πρόγραμμα Ανάγνωσης Ιστοσελίδων. Είναι ένα πρόγραμμα (εφαρμογή) που χρησιμοποιείται για τον εντοπισμό και την εμφάνιση των ιστοσελίδων (Web pages). Οι πιο γνωστοί browsers είναι ο Microsoft Internet Explorer, ο Netscape Navigator, ο Opera και ο Mozilla FireFox.

Class: Αποδίδεται στα ελληνικά με τον όρο Τάξη ή Κλάση και είναι μια κατηγορία αντικειμένων (objects) ή και το ίδιο το αρχείο του applet. Για παράδειγμα, μπορούμε

να έχουμε μια τάξη με όνομα shape που να περιέχει αντικείμενα που να είναι κύκλοι, ορθογώνια και τρίγωνα.

Compilation: Αποδίδεται στα ελληνικά με τον όρο Μεταγλώττιση και είναι η μετατροπή ενός προγράμματος που είναι γραμμένο σε μια γλώσσα υψηλού επιπέδου (high-level programming language) από τον πηγαίο κώδικα (source code), που είναι κατανοητός από τον άνθρωπο, σε κώδικα γλώσσας μηχανής ή αντικείμενο κώδικα (object code), που είναι κατανοητός μόνο από τον υπολογιστή. Υπάρχουν ειδικά προγράμματα, οι λεγόμενοι compilers (μεταγλωττιστές) για τη μετάφραση (μετατροπή) ενός προγράμματος σε κώδικα γλώσσας μηχανής.

Components: ψηφίδες λογισμικού

Constructor: Κατασκευαστής (κλάσης)

Dynamic method resolution: Δυναμική μέθοδος επίλυσης

Embedded software: λογισμικό ενσωματωμένο σε συσκευές

HTML (HyperText Markup Language): Αποδίδεται στα ελληνικά με τον όρο Γλώσσα Σήμανσης Υπερκειμένου και είναι η γλώσσα συγγραφής που χρησιμοποιούμε για να δημιουργούμε ιστοσελίδες (Web pages) για δημοσίευση στον Παγκόσμιο Ιστό (World Wide Web).

Interface: Διεπαφή

JAR (Java Archive): Είναι μια μορφή αρχείου που χρησιμοποιείται για να συνενώσει όλα τα συστατικά (components) που απαιτούνται από ένα applet της Java. Τα αρχεία JAR απλοποιούν το φόρτωμα (download) των applets εφόσον όλα τα συστατικά (αρχεία .class, εικόνες, ήχοι κ.ά.) μπορούν να πακεταρισθούν σ' ένα μόνο αρχείο.

Java: Είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου (high-level programming language) που αναπτύχθηκε από την εταιρεία Sun Microsystems. Εϊναι μια αντικειμενοστραφής γλώσσα (object-oriented language) που μοιάζει πολύ με την C++, αλλά είναι πιο απλοποιημένη ώστε να μην συμβαίνουν συνήθη προγραμματιστικά λάθη.

JavaBean: Είναι μια προδιαγραφή που αναπτύχθηκε από την εταιρεία Sun Microsystems.και η οποία ορίζει το πώς αλληλεπιδρούν τα αντικείμενα της Java. Ένα αντικείμενο που συμμορφώνεται μ' αυτήν την προδιαγραφή αποκαλείται ένα JavaBean.

Java bytecode: Ψευδοκώδικας της Java

Java interpreter: Διερμηνευτής της Java

JavaScript : Είναι μια γλώσσα συγγραφής σεναρίων (scripting language) που αναπτύχθηκε από την εταιρεία Netscape ώστε να μπορούν οι συγγραφείς ιστοσελίδων (Web authors) να σχεδιάζουν διαδραστικές ιστοσελίδες (interactive sites).
Είναι μια ουσία μια επέκταση της γλώσσας σήμανσης υπερκειμένου HTML καθώς περιέχει τις γνωστές εντολές if, switch, for κ.ά. των γλωσσών προγραμματισμού που δεν περιέχονται στην απλή HTML.
Ο κώδικας της JavaScript ενσωματώνεται μέσα στον κώδικα της HTML με τα ειδικά tags <script language=JavaScript> ... </script>. Ανταγωνιστική της JavaScript είναι η γλώσσα συγγραφής σεναρίων VBScript της εταιρείας Microsoft.

JDBC (Java DataBase Connectivity): Είναι ένα API της Java που δίνει τη δυνατότητα στα προγράμματα της Java να εκτελούν εντολές της SQL. Στην ουσία αποτελεί έναν σύνδεσμο της Java με μια βάση δεδομένων ώστε να μπορούν τα προγράμματά της να αλληλεπιδρούν και να ανταλλάσσουν δεδομένα με μια βάση δεδομένων που είναι συμβατή με την SQL.

JDK (Java Development Kit): Είναι το πακέτο ανάπτυξης λογισμικού (SDK - Software Development Kit) για τη δημιουργία προγραμμάτων σε γλώσσα προγραμματισμού Java. Δηλαδή, είναι ο compiler της java, το προγραμμα που χρειάζεται να κατεβάσουμε από την Sun Microsystems για να μπορέσουμε να προγραμματίσουμε σε Java.

JIT (Just-In-Time): Είναι ένας μεταγλωττιστής (compiler) που μετατρέπει τον bytecode της Java σε εντολές γλώσσας μηχανής.

JVM (Java Virtual Machine): Εϊναι το περιβάλλον ανάπτυξης της Java. Για παράδειγμα, τα applets της Java εκτελούνται σε μια εικονική μηχανή (JVM) η οποία δεν έχει καμία πρόσβαση στο κύριο λειτουργικό σύστημα.

Multithreading: Λέγεται η τεχνική δημιουργίας εφαρμογών που βασίζονται στα πολλαπλά νήματα (πολυνηματικότητα).

Object(s): Αντικείμενα στο λογισμικό ή στον πραγματικό κόσμο. Τα αντικείμενα του πραγματικού κόσμου περιγράφονται από χαρακτηριστικά και ιδιότητες. Τα αντικείμενα στο λογισμικό περιγράφονται από τις σχέσεις ανάμεσα στα αντικείμενα διαφορετικών κλάσεων.

Object Oriented Programming (OOP): Αποδίδεται στα ελληνικά με τον όρο Αντικειμενοστραφής και είναι ένα είδος προγραμματισμού στο οποίο δεν ορίζουμε μόνο τους τύπους των δεδομένων (data types) αλλά και τις ιδιότητες (properties) των αντικειμένων (objects) καθώς και τις μεθόδους (methods) που μπορούν να εφαρμοσθούν σ' αυτά.

Parameter: Είναι ένα χαρακτηριστικό που έχει ένα όνομα και μια τιμή και που χρησιμοποιείται για να αλλάζει (προσαρμόζεται) η συμπεριφορά ενός προγράμματος (εφαρμογής). Για παράδειγμα, οι δύο επόμενες παράμετροι ορίζουν το είδος του εφέ η πρώτη και ένα χρονικό διάστημα η δεύτερη.

```
<param name="rotatoreffect" value="dissolve">
<param name="time" value="3">
```

Servlet: Είναι ένα applet που εκτελείται σ' έναν server. Ο όρος συνήθως αναφέρεται σ' ένα applet της Java που εκτελείται μέσα σ' ένα περιβάλλον Web server.
Οι ιστοσελίδες που περιέχουν κώδικα JSP (JavaServer Pages), δηλ. ανάμιξη κώδικα HTML με κώδικα Java, μετατρέπονται (μεταφράζονται ή μεταγλωττίζονται) σε Servlets πριν εκτελεσθούν στον server. Ένα Servlet, στη γενική του μορφή, είναι μια τάξη (class) της Java που υλοποιεί (implements) το interface Servlet και δέχεται αιτήσεις (requests) και παράγει (δημιουργεί) αποκρίσεις (responses).
Οι αιτήσεις μπορεί να προέρχονται από τάξεις της Java, από Web clients ή και από άλλα Servlets. Όταν υλοποιούμε ένα interface λέμε ότι η τάξη μάς παρέχει υλοποιήσεις για τις μεθόδους που είναι δηλωμένες στο interface. Συνεπώς, όταν

υλοποιούμε το interface Servlet δηλώνουμε ότι ο κώδικάς μας θα παρέχει υλοποιήσεις για τις μεθόδους που βρίσκονται στο interface Servlet.
Αφού τα JSPs (ιστοσελίδες με JSP κώδικα) μετατρέπονται σε Servlets, τότε θα μπορεί να ρωτήσει κάποιος γιατί να πρέπει να μάθουμε και να χρησιμοποιούμε και τα δύο;
Ο βασικότερος λόγος για το γιατί πρέπει να χρησιμοποιούμε τα JSPs είναι ότι μπορούμε να γράψουμε ευκολότερα τον κώδικα, ενώ αν ασχοληθούμε με το γράψιμο του κώδικα σε Servlet, θα πρέπει να μάθουμε να δημιουργούμε τάξεις (classes) και κληρονομικότητα (inheritance) σε γλώσσα Java, κάτι όχι ιδιαίτερα επιθυμητό από τους περισσότερους. Αν και το αποτέλεσμα είναι το ίδιο, απαιτείται πολύ περισσότερη δουλειά για να γράψουμε κώδικα για Servlets και επίσης πολύ καλή γνώση της Java, κάτι που δεν είναι απαραίτητο αν θέλουμε να γράψουμε σε JSP κώδικα.

Superclass: Υπερκλάση ή κλάση πατέρας. Η μητρική (γενική) κλάση που μπορεί να έχει πολλές θυγατρικές κλάσεις, πιο εξειδικευμένες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

ΕΛΛΗΝΙΚΗ ΒΙΒΛΙΟΓΡΑΦΙΑ:

- Τσουροπλής Αθαν., Κλημόπουλος Στερ., Εισαγωγή Στην Πληροφορική, εκδόσεις νέων τεχνολογιών, 2000.
- Δουλέψτε με την Java, Stephen R. Davis, Εκδόσεις Κλειδάριθμος, Αθήνα 1996
- Java ο εύκολος τρόπος, P.K. McBride, Εκδόσεις Δίαυλος, Αθήνα 1998
- Από την C στην Java, Κλεάνθης Χρ. Θραμπολίδης, Εκδόσεις Τζιόλα, Θεσσαλονίκη 1999
- Αντικειμενοστραφής προγραμματισμός: Από την C στην Java, Κλεάνθης Χρ. Θραμπολίδης, Εκδόσεις Τζιόλα, Θεσσαλονίκη 2002
- Οδηγός της JavaScript, John Pollock, Εκδόσεις Γκιούρδας, Αθήνα 2001
- Το επίσημο βιβλίο της Netscape για την JavaScript, Kent John, Εκδόσεις Ιων, Αθήνα 1997
- Προγραμματίστε σε Java και JavaScript, Peter Wayner, Εκδόσεις Anubis, Αθήνα 1998
- Πλήρες εγχειρίδιο της Javascript / Martin Webb ; Απόδοση: Μαίρη Γκλαβά, Webb, Martin, Γκλαβά Μαίρη, Γκιούρδας, Γκιούρδας Αθήνα , 2001
- Η γλώσσα JavaScript, Λιακέας, Γιώργος, Αθήνα : Κλειδάριθμος, c2002
- Θεωρία και προβλήματα προγραμματισμού Java, John R. Hubbard, Ράπτη, Άννα, Αθήνα : Κλειδάριθμος, c1999
- Γιώργος Λιακέας *Εισαγωγή στην Java*, Εκδόσεις Κλειδάριθμος 2001.
- Γιάννη Κάβουρα. *Προγραμματισμός με Java.* Εκδόσεις Κλειθάριθμος, Αθήνα 2003
- Harvey M. Deitel και Paul J. Deitel. *Java Προγραμματισμός,* Εκδόσεις Μ. Γκιούρδας, Αθήνα 2005.

ΞΕΝΗ ΒΙΒΛΙΟΓΡΑΦΙΑ:

- Andrew S. Tanenbaum (Author), Maarten van Steen (Author), Distributed Systems: Principles and Paradigms, Prentice Hall (January 15, 2002)

- W. Richard Stevens (Author), UNIX Network Programming, Volume 2: Interprocess Communications (2nd Edition), Prentice Hall PTR; 2nd edition (August 25, 1998)
- Jeff Magee (Author), Jeff Kramer (Author), Concurrency: State Models & Java Programs, John Wiley & Sons; Book and CD-ROM edition (April 16, 1999**)**
- Jason Pritchard (Author), COM and CORBA(R) Side by Side: Architectures, Strategies, and Implementations, Addison-Wesley Pub Co; 1st edition (July 15, 1999)
- David G. Messerschmitt, Networked Applications: A Guide to the New Computing Infrastructure**,** Morgan Kaufmann; 1st edition (April 15, 1999)
- Programming with Java / Julia Case Bradley, Anita C. Millspaugh, Bradley, Julia Case, Millspaugh, Anita, MCGRAW-HILL. IRWIN, Boston : McGraw-Hill/Irwin, c2002
- Java : how to program, P.J. Deitel, H.M.Deitel, New Jersey : Pearson, c2007

ΗΛΕΚΤΡΟΝΙΚΗ ΒΙΒΛΙΟΓΡΑΦΙΑ:

- el.wikipedia.com
- www.cs.teilar.gr
- www.emp.gr
- www.uom.gr
- www.in.gr
- www.microsoft.com
- Επίσημο site της Java: http://www.javasoft.com/
- Java 2 SE 1.4.2_04: http://java.sun.com/j2se/1.4.2/download.html
- Sun's Java tutorial on-line: http://java.sun.com/docs/books/tutorial/
- Java API: http://java.sun.com/j2se/1.4.2/docs/api/
- MS VisualJ++ Home page: http://msdn.microsoft.com/visualj/
- MS VisualJ++ Programmer's guide: http://msdn.microsoft.com/library/default.asp?URL=/library/devprods/vs6/visualj/vjcore/vjovrprogrammersguide.htm

- Object Oriented Programming Concepts: http://java.sun.com/docs/books/tutorial/java/concepts/
- On-line help: http://java.sun.com/j2se/1.3/docs/api/index.html
- Red Hat. Πληροφορίες και downloads για το λειτουργικό σύστημα Linux Red Hat.
- GNU Emacs. Πληροφορίες και downloads για τον επεξεργαστή κειμένου Emacs.
- Cygwin is a UNIX environment for Windows. Πληροφορίες και downloads .
- GNU Emacs for Windows. Πληροφορίες και downloads για τον επεξεργαστή κειμένου Emacs στα Windows.