ΚΕΦΑΛΑΙΟ

1

Δημιουργία προγραμμάτων Java

Μαθησιακά αποτελέσματα:

- Ορισμός βασικής ορολογίας προγραμματισμού
- Σύγκριση διαδικαστικού και αντικειμενοστρεφούς προγραμματισμού
- Περιγραφή χαρακτηριστικών της γλώσσας προγραμματισμού Java
- Ανάλυση μιας εφαρμογής Java που παράγει έξοδο κονσόλας
- Μεταγλώττιση κλάσης Java και διόρθωση συντακτικών σφαλμάτων
- Εκτέλεση εφαρμογής Java και διόρθωση λογικών σφαλμάτων
- Προσθήκη σχολίων σε κλάση Java
- Δημιουργία εφαρμογής Java που παράγει έξοδο GUI
- Εύρεση βοήθειας

Ορισμός βασικής ορολογίας προγραμματισμού

Πρόγραμμα υπολογιστή (computer program) είναι ένα σύνολο οδηγιών που γράφετε για να πείτε στον υπολογιστή τι πρέπει να κάνει. Ο εξοπλισμός του υπολογιστή, όπως η οθόνη ή το πληκτρολόγιο, είναι το **υλικό** (hardware), ενώ τα προγράμματα είναι το **λογισμικό** (software). Το πρόγραμμα που εκτελεί μια εργασία για κάποιο χρήστη (όπως ο υπολογισμός και η εκτύπωση επιταγών μισθοδοσίας, η επεξεργασία κειμένου ή ένα παιχνίδι) είναι **λογισμικό εφαρμογής** (application software), ενώ το πρόγραμμα που διαχειρίζεται τον ίδιο τον υπολογιστή (όπως τα Windows ή το Linux) είναι **λογισμικό συστήματος** (system software). Η **λογική** (logic) πίσω από οποιοδήποτε πρόγραμμα υπολογιστή, είτε πρόκειται για πρόγραμμα εφαρμογών είτε για πρόγραμμα συστήματος, καθορίζει την ακριβή σειρά των οδηγιών που απαιτούνται για την παραγωγή των επιθυμητών αποτελεσμάτων. Μεγάλο μέρος αυτού του βιβλίου περιγράφει τον τρόπο ανάπτυξης της λογικής που απαιτείται για τη δημιουργία λογισμικού εφαρμογής.

Όλα τα προγράμματα υπολογιστή μετατρέπονται τελικά σε γλώσσα μηχανής. Η **γλώσσα μηχανής** (machine language), ή αλλιώς **κώδικας μηχανής** (machine code), είναι το πιο βασικό σύνολο οδηγιών που μπορεί να εκτελέσει ένας υπολογιστής. Κάθε τύπος επεξεργαστή (το εσωτερικό υλικό που χειρίζεται τις οδηγίες του υπολογιστή) έχει το δικό του σύνολο οδηγιών σε γλώσσα μηχανής. Οι προγραμματιστές περιγράφουν συχνά τη γλώσσα μηχανής χρησιμοποιώντας το 1 και το 0 για να αναπαραστήσουν τα διακοπτικά (on/off) λογικά κυκλώματα των συστημάτων υπολογιστών.

Το σύστημα που χρησιμοποιεί μόνο 1 και 0 είναι το *δυαδικό σύστημα αρίθμησης*. Στο Παράρτημα Β περιγράφουμε αναλυτικά το δυαδικό σύστημα. Αργότερα σ' αυτό το κεφάλαιο θα μάθετε ότι *κώδικας byte (bytecode)* είναι το όνομα για τον κώδικα που δημιουργείται όταν προγράμματα Java μετατρέπονται σε γλώσσα μηχανής.

Η γλώσσα μηχανής είναι μια **γλώσσα προγραμματισμού χαμηλού επιπέδου** (low-level programming language), δηλαδή μια γλώσσα που βρίσκεται «κοντά» στα κυκλώματα του επεξεργαστή ενός υπολογιστή. Οι γλώσσες χαμηλού επιπέδου απαιτούν από εσάς να χρησιμοποιείτε διευθύνσεις μνήμης σε συγκεκριμένες μηχανές όταν δημιουργείτε εντολές. Αυτό σημαίνει ότι οι γλώσσες χαμηλού επιπέδου είναι δύσχρηστες και πρέπει να προσαρμόζονται για κάθε τύπο μηχανής στην οποία εκτελείται ένα πρόγραμμα.

Ευτυχώς, ο προγραμματισμός έχει εξελιχθεί σε μια πιο εύκολη εργασία χάρη στην ανάπτυξη γλωσσών προγραμματισμού υψηλού επιπέδου. Η **γλώσσα προγραμματισμού υψηλού επιπέδου** (high-level programming language) σας επιτρέπει να χρησιμοποιείτε ένα λεξιλόγιο εύχρηστων όρων, όπως *διάβασε*, *γράψε* ή *πρόσθεσε*, αντί για τις ακολουθίες των 1 και 0 που εκτελούν τελικά αυτές τις λειτουργίες. Οι γλώσσες υψηλού επιπέδου σάς επιτρέπουν επίσης να αντιστοιχίζετε κατανοητά ονόματα μίας μόνο λέξης σε περιοχές της μνήμης ενός υπολογιστή όπου αποθηκεύετε δεδομένα. Αυτό σημαίνει ότι μπορείτε να χρησιμοποιήσετε μνημονικά ονόματα όπως hoursWorked ή rateOfPay, αντί να είστε αναγκασμένοι να θυμάστε τις θέσεις τους στη μνήμη. Σήμερα οι προγραμματιστές έχουν στη διάθεσή τους περισσότερες από 2.000 γλώσσες προγραμματισμού υψηλού επιπέδου και η Java είναι μία από αυτές.

Κάθε γλώσσα υψηλού επιπέδου έχει τη δική της **σύνταξη** (syntax), δηλαδή κανόνες σχετικά με το πώς συνδυάζονται σωστά μεταξύ τους τα στοιχεία της γλώσσας ώστε να συνθέτουν χρήσιμες λειτουργίες. Για παράδειγμα, ανάλογα με τη συγκεκριμένη γλώσσα υψηλού επιπέδου, θα μπορούσατε να χρησιμοποιήσετε το ρήμα *εκτύπωσε* ή *γράψε* προκειμένου να εμφανίσετε μία έξοδο. Όλες οι γλώσσες διαθέτουν ένα συγκεκριμένο, περιορισμένο λεξιλόγιο [τις **δεσμευμένες λέξεις** (keywords) της γλώσσας] και ένα συγκεκριμένο σύνολο κανόνων που υπαγορεύουν τη χρήση του συγκεκριμένου λεξιλογίου. Όταν μαθαίνετε μια γλώσσα προγραμματισμού υπολογιστών, όπως την Java, τη C++ ή τη Visual Basic, στην πραγματικότητα μαθαίνετε το λεξιλόγιο και τη σύνταξη για τη συγκεκριμένη γλώσσα.

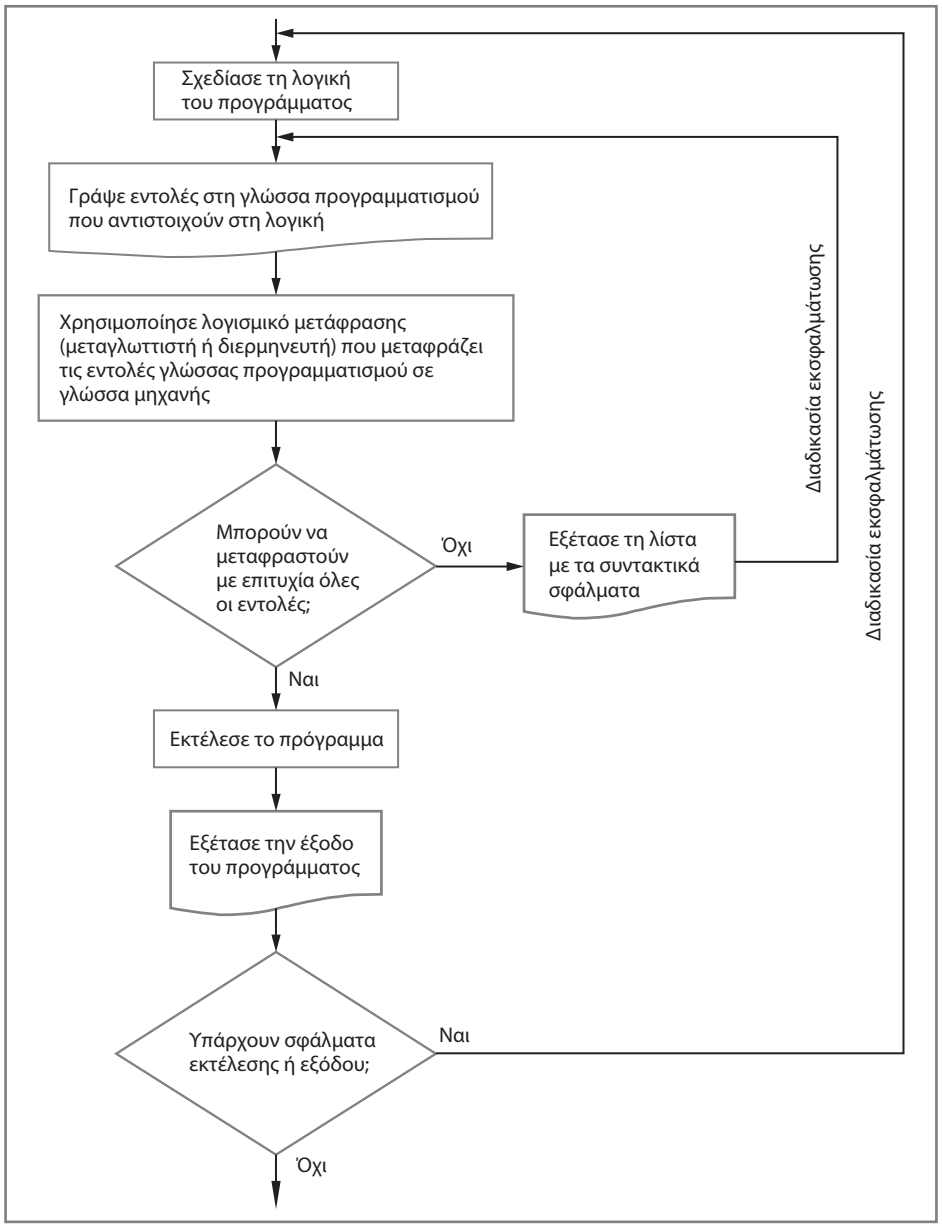
Με τη χρήση μιας γλώσσας προγραμματισμού, οι προγραμματιστές γράφουν μια σειρά από **εντολές προγραμμάτων** (program statements ή commands), οι οποίες κατορθώνουν να εκπληρώσουν τις εργασίες που ζητούνται από το πρόγραμμα. Συχνά δίνονται με προστακτική μορφή στον υπολογιστή, όπως «εμφάνισε αυτή τη λέξη» ή «πρόσθεσε αυτούς τους δύο αριθμούς».

Αφού γράψουν τις εντολές των προγραμμάτων, οι προγραμματιστές γλωσσών υψηλού επιπέδου χρησιμοποιούν ένα άλλο πρόγραμμα που ονομάζεται **μεταγλωττιστής** (compiler) ή **διερμηνευτής** (interpreter) το

οποίο μεταφράζει τις εντολές τους σε γλώσσα μηχανής. Ο μεταγλωττιστής μεταφράζει ολόκληρο το πρόγραμμα πριν **εκτελέσει** (execute) οποιαδήποτε εντολή, ενώ ο διερμηνευτής μεταφράζει μία εντολή προγράμματος κάθε φορά, εκτελώντας κάθε εντολή αμέσως μόλις τη μεταφράσει.

Το αν θα χρησιμοποιείτε μεταγλωττιστή ή διερμηνευτή εξαρτάται πολλές φορές από τη γλώσσα προγραμματισμού που χρησιμοποιείτε. Για παράδειγμα, η C++ είναι μια μεταγλωττιζόμενη γλώσσα και η Visual Basic είναι μια διερμηνευόμενη γλώσσα. Κάθε τύπος μεταφραστή έχει τους υποστηρικτές του – τα προγράμματα που γράφονται σε μεταγλωττιζόμενες γλώσσες εκτελούνται πιο γρήγορα, ενώ για τα προγράμματα που γράφονται σε διερμηνευόμενες γλώσσες, η ανάπτυξη και η εκσφαλμάτωση είναι πιο εύκολες. Η Java χρησιμοποιεί τα καλύτερα στοιχεία από τις δύο τεχνολογίες: έναν μεταγλωττιστή για τη μετάφραση των εντολών προγραμματισμού και έναν διερμηνευτή για την ανάγνωση του μεταγλωττισμένου κώδικα γραμμή προς γραμμή όταν εκτελείται το πρόγραμμα [όπως λέμε, **κατά την εκτέλεση** (at run time)].

Οι μεταγλωττιστές και οι διερμηνευτές εμφανίζουν ένα ή περισσότερα μηνύματα σφάλματος κάθε φορά που συναντούν μια μη έγκυρη εντολή προγράμματος – δηλαδή μια εντολή που περιέχει **συντακτικό σφάλμα** (syntax error) ή χρησιμοποιεί με λάθος τρόπο τη γλώσσα. Παραδείγματα συντακτικών σφαλμάτων περιλαμβάνουν



Εικόνα 1-1
Η διαδικασία ανάπτυξης προγραμμάτων

τη λάθος γραφή μιας δεσμευμένης λέξης ή την παράλειψη κάποιας λέξης που είναι υποχρεωτική για μια εντολή. Όταν ανιχνεύεται συντακτικό σφάλμα, ο προγραμματιστής μπορεί να το διορθώσει και να επιχειρήσει ξανά τη μετάφραση. Η διόρθωση όλων των συντακτικών σφαλμάτων είναι το πρώτο μέρος της διαδικασίας **εκσφαλμάτωσης** (debugging) ενός προγράμματος – της εξάλειψης όλων των **σφαλμάτων** (bugs) και των ατελειών από το πρόγραμμα. Η Εικόνα 1-1 παρουσιάζει τα βήματα που ακολουθεί ο προγραμματιστής κατά την ανάπτυξη ενός εκτελέσιμου προγράμματος. Θα μάθετε περισσότερα για την εκσφαλμάτωση των προγραμμάτων Java παρακάτω σ' αυτό το κεφάλαιο.

Όπως δείχνει η Εικόνα 1-1, θα μπορούσατε να γράψετε ένα πρόγραμμα με σωστή σύνταξη, αλλά με λογικά σφάλματα. **Λογικό σφάλμα** (logic error) είναι η ατέλεια που επιτρέπει την εκτέλεση ενός προγράμματος, αλλά δεν του επιτρέπει να εκτελεστεί σωστά. Η ορθή λογική συνεπάγεται ότι όλες οι κατάλληλες εντολές θα δίνονται με την κατάλληλη σειρά μέσα στο πρόγραμμα. Παραδείγματα λογικών σφαλμάτων είναι ο πολλαπλασιασμός δύο τιμών ενώ θέλετε να τις διαιρέσετε ή η έξοδος αποτελεσμάτων πριν αποκτηθεί η κατάλληλη είσοδος. Όταν αναπτύσσετε ένα πρόγραμμα μεγάλου μεγέθους, θα πρέπει να σχεδιάσετε τη λογική του πριν ξεκινήσετε να γράφετε τις εντολές του.

Η διόρθωση λογικών σφαλμάτων είναι πολύ πιο δύσκολη από τη διόρθωση συντακτικών σφαλμάτων. Τα συντακτικά σφάλματα εντοπίζονται από τον μεταφραστή της γλώσσας όταν μεταγλωττίζετε ένα πρόγραμμα. Όμως το πρόγραμμα μπορεί να μην έχει συντακτικά σφάλματα και να εκτελείται, ενώ εξακολουθεί να έχει λογικά σφάλματα. Υπάρχουν φορές που τα λογικά σφάλματα αναγνωρίζονται μόνο κατά την εξέταση της εξόδου ενός προγράμματος. Για παράδειγμα, αν γνωρίζετε ότι ο μισθός ενός υπαλλήλου περιέχει την τιμή 4.000, αλλά όταν εξετάζετε την έξοδο του προγράμματος μισθοδοσίας βλέπετε ότι περιέχει την τιμή 40, τότε αντιλαμβάνεστε ότι υπάρχει λογικό σφάλμα. Ίσως εκτελέστηκε μια λάθος πράξη ή ενδεχομένως η έξοδος προβάλλει τις ώρες εργασίας αντί του πληρωτέου μισθού. Όταν η έξοδος είναι λανθασμένη, ο προγραμματιστής πρέπει να εξετάσει προσεκτικά όλες τις εντολές μέσα στο πρόγραμμα, να αναθεωρήσει ή να μετακινήσει τις προβληματικές εντολές και να μεταφράσει και δοκιμάσει το πρόγραμμα ξανά.

Απλώς και μόνο επειδή ένα πρόγραμμα παράγει σωστή έξοδο, δεν συνεπάγεται ότι δεν περιέχει λογικά σφάλματα. Για παράδειγμα, έστω ότι ένα πρόγραμμα πρέπει να πολλαπλασιάζει δύο τιμές που εισάγει ο χρήστης, ότι ο χρήστης εισήγαγε τον αριθμό 2 και για τις δύο τιμές και ότι η έξοδος είναι 4. Το πρόγραμμα θα μπορούσε πιθανώς να έχει προσθέσει κατά λάθος τις τιμές. Ο προγραμματιστής θα ανακάλυπτε το λογικό σφάλμα μόνο εισάγοντας διαφορετικές τιμές, όπως 5 και 7, και με την επανεξέταση του αποτελέσματος.

Οι προγραμματιστές αναφέρονται σε ορισμένα λογικά σφάλματα ως **σημασιολογικά σφάλματα** (semantic error). Για παράδειγμα, αν κάνετε ορθογραφικό λάθος σε μια λέξη γλώσσας προγραμματισμού, διαπράττετε συντακτικό σφάλμα, αλλά αν χρησιμοποιήσετε τη σωστή λέξη σε λάθος πλαίσιο χρήσης, διαπράττετε σημασιολογικό σφάλμα.

ΔΥΟ ΑΛΗΘΕΙΕΣ ΚΙ ΕΝΑ ΨΕΜΑ

Ορισμός βασικής ορολογίας προγραμματισμού

Σε κάθε πλαίσιο «Δύο αλήθειες κι ένα ψέμα», δύο από τις αριθμημένες προτάσεις είναι αληθείς και μία είναι ψευδής. Αναγνωρίστε την ψευδή πρόταση και εξηγήστε γιατί είναι ψευδής.

1. Αντίθετα από μια γλώσσα προγραμματισμού χαμηλού επιπέδου, η γλώσσα προγραμματισμού υψηλού επιπέδου σάς επιτρέπει να χρησιμοποιείτε λεξιλόγιο εύχρηστων όρων αντί για τις ακολουθίες ανοιχτών και κλειστών διακοπτών (λογικών κυκλωμάτων) που εκτελούν τις αντίστοιχες εργασίες.
2. Έχετε συντακτικό σφάλμα όταν χρησιμοποιείτε με λανθασμένο τρόπο τη γλώσσα προγραμματισμού. Ο εντοπισμός και η επιδιόρθωση όλων των συντακτικών σφαλμάτων αποτελούν τμήματα της διαδικασίας εκσφαλμάτωσης ενός προγράμματος.
3. Τα λογικά σφάλματα εντοπίζονται σχετικά εύκολα, επειδή το λογισμικό που μεταφράζει ένα πρόγραμμα βρίσκει αυτόματα όλα τα λογικά σφάλματα.

Η ψευδής πρόταση είναι η #3. Ο μεταφραστής μιας γλώσσας βρίσκει τα συντακτικά σφάλματα, αλλά τα λογικά σφάλματα συνήθως ανακαλύπτονται μόνο με εξέταση της εξόδου ενός προγράμματος.

Σύγκριση διαδικαστικού και αντικειμενοστρεφούς προγραμματισμού

Δύο δημοφιλείς προσεγγίσεις για τη δημιουργία προγραμμάτων υπολογιστή είναι ο διαδικαστικός προγραμματισμός και ο αντικειμενοστρεφής προγραμματισμός.

Διαδικαστικός προγραμματισμός

Ο **διαδικαστικός προγραμματισμός** (procedural programming) είναι το είδος προγραμματισμού στο οποίο οι πράξεις εκτελούνται η μία μετά την άλλη, διαδοχικά. Σε διαδικαστικές εφαρμογές, ορίζετε ονόματα για θέσεις της μνήμης υπολογιστή οι οποίες μπορούν να διατηρούν τιμές –για παράδειγμα, αριθμούς και κείμενο– σε ηλεκτρονική μορφή. Οι επώνυμες θέσεις της μνήμης ονομάζονται **μεταβλητές** (variables) επειδή περιέχουν τιμές που θα μπορούσαν να αλλάξουν. Για παράδειγμα, ένα πρόγραμμα μισθοδοσίας μπορεί να περιέχει μια μεταβλητή με το όνομα rateOfPay. Η θέση μνήμης στην οποία παραπέμπει το όνομα rateOfPay μπορεί να περιέχει διαφορετικές τιμές (διαφορετική τιμή για κάθε υπάλληλο της εταιρείας) σε διαφορετικές χρονικές στιγμές. Κατά την εκτέλεση του προγράμματος μισθοδοσίας, για κάθε τιμή που αποθηκεύεται στη θέση μνήμης με το όνομα rateOfPay μπορεί να εκτελούνται πολλές πράξεις – για παράδειγμα, η τιμή μπορεί να διαβαστεί από συσκευή εξόδου, να πολλαπλασιαστεί επί μία άλλη μεταβλητή που αναπαριστά ώρες εργασίας και να εκτυπωθεί σε χαρτί.

Για πρακτικούς λόγους, οι μεμονωμένες πράξεις που χρησιμοποιούνται σε ένα πρόγραμμα υπολογιστή ομαδοποιούνται συχνά σε λογικές μονάδες, τις **διαδικασίες** (procedures). Για παράδειγμα, μια σειρά από τέσσερις ή πέντε συγκρίσεις και πράξεις που καθορίζουν συνολικά τον φόρο παρακράτησης για ένα πρόσωπο θα μπορούσαν να ομαδοποιηθούν ως μια διαδικασία με το όνομα calculateFederalWithholding. Ένα διαδικαστικό πρόγραμμα ορίζει τις θέσεις μνήμης των μεταβλητών και έπειτα καλεί μια σειρά από διαδικασίες για είσοδο, επεξεργασία και έξοδο των τιμών που αποθηκεύονται σ' αυτές τις θέσεις. Όταν το πρόγραμμα **καλεί μια διαδικασία** (call a procedure), η τρέχουσα λογική ροή εκτέλεσης του προγράμματος εγκαταλείπεται προσωρινά, ώστε να υπάρχει η δυνατότητα να εκτελεστούν οι εντολές της διαδικασίας. Ένα μόνο διαδικαστικό πρόγραμμα μπορεί να περιέχει εκατοντάδες μεταβλητές και κλήσεις διαδικασιών. Οι διαδικασίες αναφέρονται επίσης ως *ενότητες, μέθοδοι, λειτουργίες* και *υπορουτίνες*. Οι χρήστες διαφορετικών γλωσσών προγραμματισμού συνηθίζουν να χρησιμοποιούν διαφορετική ορολογία. Όπως θα μάθετε αργότερα σ' αυτό το κεφάλαιο, οι προγραμματιστές Java χρησιμοποιούν συχνότερα τον όρο *μέθοδος*.

Αντικειμενοστρεφής προγραμματισμός

Ο αντικειμενοστρεφής προγραμματισμός είναι επέκταση του διαδικαστικού προγραμματισμού, με μια ελαφρώς διαφορετική προσέγγιση στη σύνταξη προγραμμάτων. Η σύνταξη **αντικειμενοστρεφών προγραμμάτων** (object-oriented programs) περιλαμβάνει:

- Τη δημιουργία κλάσεων, οι οποίες αποτελούν πρότυπα για την κατασκευή αντικειμένων
- Τη δημιουργία αντικειμένων, τα οποία είναι συγκεκριμένα στιγμιότυπα αυτών των κλάσεων
- Τη δημιουργία εφαρμογών που χειρίζονται ή χρησιμοποιούν αυτά τα αντικείμενα

Οι προγραμματιστές χρησιμοποιούν τα αρχικά *OO* για τον όρο *αντικειμενοστρεφής* (object-oriented), ενώ ο αντικειμενοστρεφής προγραμματισμός απαντά ως *OOP* (object-oriented programming).

Αρχικά ο αντικειμενοστρεφής προγραμματισμός χρησιμοποιούνταν πιο συχνά για δύο βασικές κατηγορίες εφαρμογών:

- **Προσομοιώσεις σε υπολογιστές** (computer simulations), οι οποίες επιχειρούν να μιμηθούν πραγματικές δραστηριότητες, έτσι ώστε οι διεργασίες τους να μπορούν να βελτιωθούν ή οι χρήστες να είναι σε θέση να κατανοήσουν καλύτερα πώς λειτουργούν οι πραγματικές διεργασίες
- **Γραφικές διεπαφές χρήστη** ή **GUI** (από τις λέξεις *graphical user interfaces*), οι οποίες επιτρέπουν στους χρήστες να αλληλεπιδρούν με ένα πρόγραμμα σε γραφικό περιβάλλον

Είναι εύκολο να σκεφτούμε αντικείμενα σε αυτές τις δύο κατηγορίες εφαρμογών. Για παράδειγμα, ένας δήμος ίσως θέλει να αναπτύξει ένα πρόγραμμα που προσομοιώνει κυκλοφοριακά μοτίβα, έτσι ώστε να προσπαθήσει να καταστρώσει σχέδια καταπολέμησης της κυκλοφοριακής συμφόρησης. Οι προγραμματιστές θα δημιουργούσαν κλάσεις για αντικείμενα όπως τα αυτοκίνητα και οι πεζοί, τα οποία θα περιείχαν τα δικά τους δεδομένα και κανόνες συμπεριφοράς. Για παράδειγμα, κάθε αυτοκίνητο έχει μια ταχύτητα και μια μέθοδο για μεταβολή αυτής της ταχύτητας. Τα συγκεκριμένα στιγμιότυπα αυτοκινήτων θα μπορούσαν να τεθούν σε κίνηση, ώστε να δημιουργηθεί προσομοίωση μιας πραγματικής πόλης σε ώρα αιχμής.

Η δημιουργία περιβάλλοντος GUI για χρήστες είναι επίσης μια χρήση που μοιάζει φυσική για τον αντικειμενοστρεφή προγραμματισμό. Είναι εύκολο να σκεφτούμε τα στοιχεία που ένας χρήστης χειρίζεται σε μια οθόνη υπολογιστή, όπως κουμπιά και γραμμές κύλισης, ως στοιχεία που θυμίζουν πραγματικά αντικείμενα. Κάθε αντικείμενο GUI περιέχει δεδομένα – για παράδειγμα, το κουμπί σε μια οθόνη έχει συγκεκριμένο μέγεθος και χρώμα. Κάθε αντικείμενο περιέχει επίσης συμπεριφορές – για παράδειγμα, σε κάθε κουμπί ο χρήστης μπορεί να κάνει κλικ και το κουμπί αντιδρά με συγκεκριμένο τρόπο. Υπάρχουν αρκετοί που θεωρούν ότι ο όρος *αντικειμενοστρεφής προγραμματισμός* είναι συνώνυμος με τον προγραμματισμό GUI, αλλά ο αντικειμενοστρεφής προγραμματισμός σημαίνει περισσότερα πράγματα. Μολονότι πολλά προγράμματα GUI είναι αντικειμενοστρεφή, δεν χρησιμοποιούν όλα τα αντικειμενοστρεφή προγράμματα αντικείμενα GUI. Οι σύγχρονες επιχειρήσεις χρησιμοποιούν τεχνικές αντικειμενοστρεφούς σχεδίασης κατά την ανάπτυξη διαφόρων επαγγελματικών εφαρμογών, είτε είναι εφαρμογές GUI είτε όχι. Στα πρώτα 13 κεφάλαια αυτού του βιβλίου, θα μάθετε αντικειμενοστρεφείς τεχνικές που ταιριάζουν σε οποιονδήποτε τύπο προγράμματος, και στα τελευταία κεφάλαια θα εφαρμόσετε όσα μάθατε γι' αυτές τις τεχνικές ειδικά σε εφαρμογές GUI.

Η κατανόηση του αντικειμενοστρεφούς προγραμματισμού προϋποθέτει την κατανόηση τριών βασικών εννοιών:

- Ενθυλάκωση όπως εφαρμόζεται σε κλάσεις με τη μορφή αντικειμένων
- Κληρονομικότητα
- Πολυμορφισμός

Κλάσεις, αντικείμενα και ενθυλάκωση

Στην αντικειμενοστρεφή ορολογία, η **κλάση** (class) είναι ο όρος που περιγράφει μια ομάδα ή συλλογή αντικειμένων με κοινές ιδιότητες. Με τον ίδιο τρόπο που ένα σχέδιο προϋπάρχει του σπιτιού που κατασκευάζεται με βάση αυτό το σχέδιο, και με τον ίδιο τρόπο που μια συνταγή προϋπάρχει οποιουδήποτε φαγητού ψήνεται με βάση αυτή τη συνταγή, ο ορισμός μιας κλάσης προϋπάρχει οποιουδήποτε αντικειμένου δημιουργείται με βάση αυτό τον ορισμό. Ο **ορισμός κλάσης** (class definition) περιγράφει ποια χαρακτηριστικά θα έχουν και τι θα μπορούν να κάνουν τα αντικείμενα. Τα **χαρακτηριστικά** (attributes) είναι τα γνωρίσματα που καθορίζουν ένα αντικείμενο – οι **ιδιότητες** (properties) του αντικειμένου. Όταν μαθαίνετε μια γλώσσα προγραμματισμού όπως η Java, μαθαίνετε να χρησιμοποιείτε δύο τύπους κλάσεων: εκείνες που έχουν ήδη αναπτυχθεί από τους δημιουργούς της γλώσσας και τις δικές σας καινούριες, προσαρμοσμένες κλάσεις.

Αντικείμενο (object) είναι το συγκεκριμένο **στιγμιότυπο** (instance) μιας κλάσης. Η δημιουργία ενός στιγμιοτύπου ονομάζεται **υποστασιοποίηση** (instantiation). Μπορείτε να δημιουργείτε αντικείμενα από κλάσεις που γράφετε και από κλάσεις άλλων προγραμματιστών, συμπεριλαμβανομένων των δημιουργών της Java. Οι τιμές που περιέχουν οι ιδιότητες ενός αντικειμένου πολλές φορές διαφοροποιούν τα στιγμιότυπα της ίδιας κλάσης μεταξύ τους. Για παράδειγμα, η κλάση Automobile περιγράφει πώς είναι τα αντικείμενα Automobile. Μερικές από τις ιδιότητες της κλάσης Automobile είναι η μάρκα, το μοντέλο, το έτος και το χρώμα. Κάθε αντικείμενο Automobile έχει τα ίδια χαρακτηριστικά, αλλά όχι απαραίτητα τις ίδιες τιμές γι' αυτά τα συγκεκριμένα χαρακτηριστικά. Ένα αυτοκίνητο (Automobile) μπορεί να είναι λευκό Ford Taurus του 2010, και ένα άλλο μπορεί να είναι κόκκινο Chevrolet Camaro του 2015. Ομοίως, ο σκύλος σας έχει τις ιδιότητες όλων των σκυλιών (της κλάσης Dog), όπως ράτσα, όνομα, ηλικία και το κατά πόσο έχει κάνει όλα τα εμβόλια που πρέπει. Οι τιμές των ιδιοτήτων ενός αντικειμένου αναφέρονται ως **κατάσταση** (state) του αντικειμένου. Με άλλα λόγια, τα αντικείμενα αντιστοιχούν κατά κάποιο τρόπο στα ουσιαστικά, ενώ τα χαρακτηριστικά αντιστοιχούν στα επίθετα που περιγράφουν τα ουσιαστικά.

Όταν καταλαβαίνετε την κλάση ενός αντικειμένου, καταλαβαίνετε ποια είναι τα χαρακτηριστικά του αντικει-

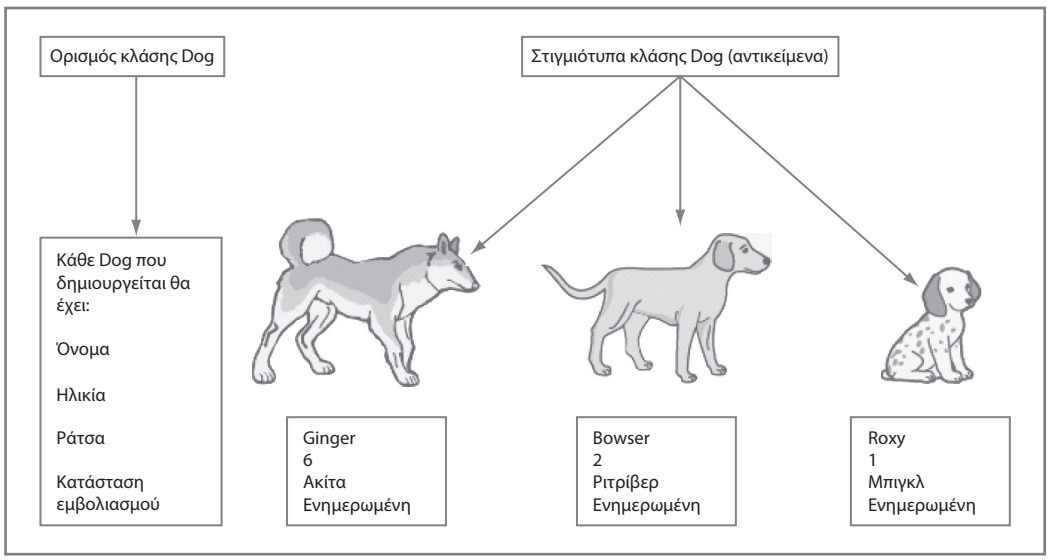
μένου. Αν ο φίλος σας αγοράσει ένα Automobile, γνωρίζετε ότι αυτό έχει όνομα μοντέλου, ενώ αν ο φίλος σας πάρει ένα Dog, γνωρίζετε ότι ο σκύλος έχει ράτσα. Αν γνωρίζετε ποια χαρακτηριστικά υπάρχουν για κλάσεις, είστε σε θέση να θέσετε τα κατάλληλα ερωτήματα για τις καταστάσεις ή τις τιμές αυτών των χαρακτηριστικών. Για παράδειγμα, θα μπορούσατε να ρωτήσετε για την κατανάλωση καυσίμων του αυτοκινήτου, αλλά δεν θα ρωτούσατε αν το αυτοκίνητο έχει κάνει εμβόλια. Ομοίως, σε ένα λειτουργικό περιβάλλον GUI, περιμένετε από κάθε στοιχείο να διαθέτει συγκεκριμένα χαρακτηριστικά και μεθόδους, όπως ένα παράθυρο με γραμμή τίτλου και κουμπί τερματισμού, επειδή κάθε στοιχείο διαθέτει αυτές τις ιδιότητες ως μέλος της γενικής κλάσης στοιχείων GUI. Η Εικόνα 1-2 παρουσιάζει τη σχέση ορισμένων αντικειμένων Dog με την κλάση Dog.

Βάσει σύμβασης, οι προγραμματιστές που χρησιμοποιούν την Java ξεκινούν τα ονόματα των κλάσεών τους με κεφαλαίο γράμμα. Ως εκ τούτου, η κλάση που ορίζει τα χαρακτηριστικά και τις μεθόδους ενός αυτοκινήτου μάλλον θα ονομάζεται Automobile, ενώ η κλάση για τους σκύλους μάλλον θα έχει το όνομα Dog. Ωστόσο, η πιστή τήρηση αυτής της σύμβασης δεν είναι υποχρεωτική για τη δημιουργία ενός λειτουργικού προγράμματος.

Πέρα από τον ορισμό ιδιοτήτων, οι κλάσεις ορίζουν μεθόδους που μπορούν να χρησιμοποιούν τα αντικείμενά τους. **Μέθοδος** (method) είναι το ανεξάρτητο κομμάτι κώδικα προγράμματος που εκτελεί κάποια ενέργεια, όπως τη διαδικασία σε κάποιο διαδικαστικό πρόγραμμα. Ένα Automobile, για παράδειγμα, μπορεί να έχει μεθόδους για εμπρόσθια κίνηση, για όπισθεν κίνηση και για τον προσδιορισμό της κατάστασης του ντεπόζιτου. Ομοίως, ένα Dog μπορεί να έχει μεθόδους για περίπατο, διατροφή και ορισμό του ονόματός του, ενώ τα στοιχεία ενός προγράμματος GUI μπορεί να έχουν μεθόδους για τη μεγιστοποίηση και ελαχιστοποίησή τους, αλλά και για τον καθορισμό του μεγέθους τους. Με άλλα λόγια, αν τα αντικείμενα μοιάζουν με ουσιαστικά, οι μέθοδοι μοιάζουν με ρήματα.

Σε αντικειμενοστρεφείς κλάσεις, τα χαρακτηριστικά και οι μέθοδοι ενθυλακώνονται σε αντικείμενα. Η **ενθυλάκωση** (encapsulation) αναφέρεται σε δύο στενά συνδεδεμένες αντικειμενοστρεφείς έννοιες:

- Ενθυλάκωση είναι ο εγκλεισμός δεδομένων και μεθόδων μέσα σε ένα αντικείμενο. Η ενθυλάκωση σας επιτρέπει να χειρίζεστε όλες τις μεθόδους και τα δεδομένα ενός αντικειμένου ως μία ενιαία οντότητα. Όπως ένας σκύλος περιέχει όλα τα χαρακτηριστικά και τις ικανότητές του, το ίδιο συμβαίνει με το αντικείμενο Dog ενός προγράμματος.
- Η ενθυλάκωση αναφέρεται επίσης στην απόκρυψη των δεδομένων και των μεθόδων ενός αντικειμένου από εξωτερικές πηγές. Η απόκρυψη των δεδομένων ονομάζεται γενικά *απόκρυψη πληροφορίας* (*information hiding*), ενώ η απόκρυψη του τρόπου λειτουργίας των μεθόδων είναι η *απόκρυψη υλοποίησης* (*implementation hiding*) – θα μάθετε περισσότερα για αμφότερους τους όρους στο κεφάλαιο «Χρήση μεθόδων, κλάσεων και αντικειμένων». Η ενθυλάκωση σας δίνει τη δυνατότητα να αποκρύβετε συγκεκριμένα χαρακτηριστικά και μεθόδους αντικειμένων από εξωτερικές πηγές και διατηρεί δεδομένα και μεθόδους ασφαλή από απρόσεκτες μεταβολές.



Εικόνα 1-2 Ορισμός κλάσης Dog και ορισμένα αντικείμενα που δημιουργούνται απ' αυτή

Αν οι μέθοδοι ενός αντικειμένου είναι καλογραμμένες, ο χρήστης δεν θα χρειάζεται να γνωρίζει λεπτομέρειες χαμηλού επιπέδου για τον τρόπο εκτέλεσης των μεθόδων και απλώς θα πρέπει να κατανοεί τη διεπαφή ή την αλληλεπίδραση μεταξύ μεθόδου και αντικειμένου. Για παράδειγμα, αν μπορείτε να γεμίσετε το Automobile σας με βενζίνη, αυτό συμβαίνει επειδή κατανοείτε τη διεπαφή ανάμεσα στο ακροφύσιο της αντλίας βενζίνης και την υποδοχή της δεξαμενής καυσίμου στο όχημά σας. Δεν χρειάζεται να γνωρίζετε τη μηχανική λειτουργία της αντλίας ή πού ακριβώς βρίσκεται το ντεπόζιτο εντός του οχήματός σας. Αν μπορείτε να διαβάσετε το ταχύμετρό σας, δεν έχει σημασία πώς υπολογίζεται το νούμερο που βλέπετε. Αν μάλιστα κάποιος κατασκευάσει μια καλύτερη συσκευή προσδιορισμού της ταχύτητας, με μεγαλύτερη ακρίβεια, και την τοποθετήσει στο Automobile σας, δεν χρειάζεται να το γνωρίζετε, ούτε θα σας ενδιέφερε πώς λειτουργεί, αρκεί η διεπαφή σας με αυτή να παραμείνει ίδια. Οι ίδιες αρχές ισχύουν για καλά κατασκευασμένες κλάσεις που χρησιμοποιούνται σε αντικειμενοστρεφή προγράμματα – για τα προγράμματα τα οποία χρησιμοποιούν κλάσεις, αρκεί να είναι γνωστές οι διεπαφές τους.

Κληρονομικότητα και πολυμορφισμός

Μια σημαντική δυνατότητα του σχεδιασμού αντικειμενοστρεφών προγραμμάτων είναι η **κληρονομικότητα** (inheritance) – η ικανότητα δημιουργίας κλάσεων που μοιράζονται τα χαρακτηριστικά και τις μεθόδους υφιστάμενων κλάσεων με πιο συγκεκριμένα χαρακτηριστικά. Για παράδειγμα, το Automobile (αυτοκίνητο) είναι μια κλάση, και όλα τα αντικείμενα της κλάσης Automobile μοιράζονται πολλά χαρακτηριστικά και δυνατότητες. Το Convertible (καμπριολέ) είναι μια κλάση η οποία κληρονομεί χαρακτηριστικά από την κλάση Automobile –Convertible είναι ένας τύπος Automobile που έχει και μπορεί να κάνει ό,τι κάνει και ένα «απλό» Automobile– με τη διαφορά ότι διαθέτει την πρόσθετη ικανότητα να αφαιρεί τον ουρανό του. [Με την ίδια λογική, η κλάση αυτοκινήτων Automobile κληρονομεί από την κλάση Vehicle (όχημα)]. Το Convertible δεν είναι αντικείμενο – είναι κλάση. Ένα συγκεκριμένο στιγμιότυπο της κλάσης Convertible είναι αντικείμενο – για παράδειγμα, το my1967 BlueMustangConvertible.

Η κληρονομικότητα συμβάλλει στην κατανόηση των πραγματικών αντικειμένων. Για παράδειγμα, την πρώτη φορά που βλέπετε ένα καμπριολέ αυτοκίνητο, γνωρίζετε πώς λειτουργεί η ανάφλεξη, η πέδηση, το κλείδωμα των θυρών και άλλα συστήματα, επειδή συνειδητοποιείτε ότι το καμπριολέ είναι τύπος αυτοκινήτου, άρα πρέπει να σας ενδιαφέρουν μόνο τα χαρακτηριστικά και οι μέθοδοι που είναι «νέα» σε ένα καμπριολέ. Τα πλεονεκτήματα στον προγραμματισμό είναι τα ίδια – μπορείτε να κατασκευάσετε νέες κλάσεις με βάση υπάρχουσες και να επικεντρωθείτε στις εξειδικευμένες δυνατότητες που θα προσθέσετε.

Μια τελική σημαντική έννοια στην αντικειμενοστρεφή ορολογία είναι ο **πολυμορφισμός** (polymorphism). Ουσιαστικά ο πολυμορφισμός αναφέρεται σε «πολλές μορφές» – περιγράφει τη δυνατότητα που έχουν οι γλώσσες να ερμηνεύουν σωστά την ίδια γλώσσα ή σύμβολο σε διαφορετικές καταστάσεις, ανάλογα με το πλαίσιο αναφοράς. Για παράδειγμα, αν και οι κλάσεις Automobile, Sailboat και Airplane (αυτοκίνητο, σκάφος και αεροπλάνο, αντίστοιχα) κληρονομούν όλες από την κλάση Vehicle (όχημα), οι μέθοδοι turn και stop (στροφή και στάμάτημα) λειτουργούν διαφορετικά για τα στιγμιότυπα καθεμιάς από αυτές τις κλάσεις. Τα πλεονεκτήματα του πολυμορφισμού θα διαφανούν όταν θα αρχίσετε να δημιουργείτε εφαρμογές GUI που περιέχουν δυνατότητες όπως παράθυρα, κουμπιά και γραμμές μενού. Σε μια εφαρμογή GUI, είναι πιο εύκολο να θυμάστε ένα όνομα μεθόδου, όπως setColor ή setHeight, κι αυτό να λειτουργεί πάντα σωστά με τις κατάλληλες προσαρμογές, ανεξάρτητα από το είδος του αντικειμένου που τροποποιείτε.

Όταν βλέπετε ένα συν (+) ανάμεσα σε δύο αριθμούς, αντιλαμβάνεστε ότι οι δύο αριθμοί προστίθενται. Όταν θα δείτε το ίδιο σύμβολο σκαλισμένο σε ένα δέντρο ανάμεσα σε δύο ονόματα, αντιλαμβάνεστε ότι αυτά τα ονόματα συνδέονται ερωτικά. Επειδή το σύμβολο έχει διαφορετικό νόημα ανάλογα με το πλαίσιο στο οποίο χρησιμοποιείται, είναι πολυμορφικό. Στα Κεφάλαια 10 και 11 θα βρείτε περισσότερες πληροφορίες για την κληρονομικότητα και τον πολυμορφισμό και πώς υλοποιούνται στην Java.

ΔΥΟ ΑΛΗΘΕΙΕΣ ΚΙ ΕΝΑ ΨΕΜΑ

Σύγκριση διαδικαστικού και αντικειμενοστρεφούς προγραμματισμού

1. Στιγμιότυπο μιας κλάσης είναι αντικείμενο το οποίο έχει δημιουργηθεί και διαθέτει τα χαρακτηριστικά και τις μεθόδους που περιγράφονται στον ορισμό της κλάσης.
2. Η ενθυλάκωση προστατεύει τα δεδομένα κρύβοντάς τα μέσα σε ένα αντικείμενο.
3. Πολυμορφισμός είναι η ικανότητα δημιουργίας κλάσεων που μοιράζονται τα χαρακτηριστικά και τις μεθόδους υφιστάμενων κλάσεων, αλλά με πιο συγκεκριμένες δυνατότητες.

Η ψευδής πρόταση είναι η #3. Η κληρονομικότητα είναι η ικανότητα δημιουργίας κλάσεων που μοιράζονται τα χαρακτηριστικά και τις μεθόδους υφιστάμενων κλάσεων, αλλά με πιο συγκεκριμένες δυνατότητες, ενώ ο πολυμορφισμός περιγράφει την ικανότητα χρήσης ενός όρου για την έναρξη πολλών ενεργειών.

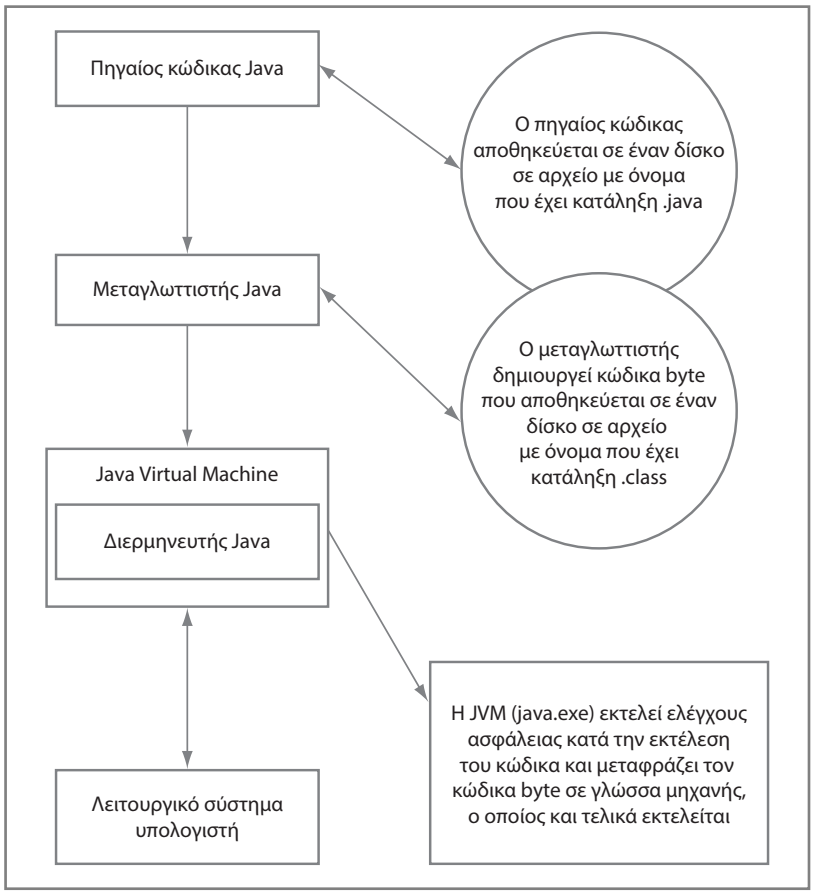
Περιγραφή χαρακτηριστικών της γλώσσας προγραμματισμού Java

Η **Java** αναπτύχθηκε από τη Sun Microsystems ως αντικειμενοστρεφής γλώσσα για επαγγελματικές εφαρμογές γενικής χρήσης και για αλληλοεπιδραστικές εφαρμογές Διαδικτύου που βασίζονται στον Παγκόσμιο Ιστό (η Sun εξαγοράστηκε αργότερα από την Oracle Corporation). Μερικά από τα πλεονεκτήματα που κάνουν την Java τόσο δημοφιλή είναι τα χαρακτηριστικά ασφάλειας εκτέλεσης των προγραμμάτων που παρέχει, αλλά και το γεγονός ότι είναι **αρχιτεκτονικά ουδέτερη** (architecturally neutral): Αντίθετα από άλλες γλώσσες, μπορείτε να χρησιμοποιείτε την Java για προγράμματα που μπορούν να εκτελούνται σε οποιοδήποτε λειτουργικό σύστημα (όπως Windows, Mac OS ή Linux) ή συσκευή (όπως PC, τηλέφωνα και tablet).

Η Java μπορεί να εκτελεστεί σε μεγάλη ποικιλία υπολογιστών και συσκευών επειδή δεν εκτελεί εντολές απευθείας στον υπολογιστή. Αντίθετα, η Java εκτελείται σε έναν υποθετικό, εικονικό υπολογιστή, την **Java Virtual Machine** (JVM). Με τον όρο *εικονικό* αναφερόμαστε σε κάτι που δεν αποτελεί φυσική οντότητα η οποία κατασκευάζεται από κάποιο υλικό, αλλά σε κάτι που αποτελείται μόνο από λογισμικό.

Η Εικόνα 1-3 παρουσιάζει το περιβάλλον της Java. Οι εντολές προγραμματισμού που γράφονται σε μια γλώσσα προγραμματισμού υψηλού επιπέδου είναι ο **πηγαίος κώδικας** (source code). Όταν γράφετε ένα πρόγραμμα Java, πρώτα κατασκευάζετε τον πηγαίο κώδικα σε πρόγραμμα επεξεργασίας κειμένου όπως το Notepad, σε περιβάλλον ανάπτυξης ή σε επεξεργαστή πηγαίου κώδικα όπως το **jGRASP**, το οποίο μπορείτε να λάβετε δωρεάν από το Διαδίκτυο. **Περιβάλλον ανάπτυξης** (development environment) είναι το σύνολο εργαλείων με τα οποία μπορείτε να γράφετε προγράμματα, καθώς σας παρέχουν διάφορες δυνατότητες όπως η εμφάνιση των δεσμευμένων λέξεων της γλώσσας με διαφορετικό χρώμα. Οι εντολές αποθηκεύονται σε ένα αρχείο και, έπειτα, ο μεταγλωττιστής Java μετατρέπει τον πηγαίο κώδικα σε **κώδικα byte** (bytecode). Αμέσως μετά, ένα πρόγραμμα που ονομάζεται **διερμηνευτής Java** (Java interpreter) ελέγχει τον κώδικα byte και επικοινωνεί με το λειτουργικό σύστημα, ώστε οι οδηγίες του κώδικα byte να εκτελεστούν γραμμή προς γραμμή από την Java Virtual Machine. Επειδή το πρόγραμμα Java δεν έρχεται σε επαφή με το λειτουργικό σύστημα, είναι απομονωμένο από το συγκεκριμένο υλικό στο οποίο εκτελείται κάθε φορά. Χάρη σ' αυτή την απομόνωση, η JVM παρέχει προστασία από εισβολείς που θα μπορούσαν να προσπελάσουν το υλικό του υπολογιστή σας μέσα από το λειτουργικό σύστημα, και κατά συνέπεια η Java θεωρείται πιο ασφαλής από άλλες γλώσσες. Υπάρχει ακόμα ένα πλεονέκτημα που παρέχει η JVM, το οποίο έχει ως θετικό αποτέλεσμα λιγότερη δουλειά για τους προγραμματιστές – με άλλες γλώσσες προγραμματισμού, οι προμηθευτές λογισμικού πρέπει συνήθως να παράγουν πολλαπλές εκδόσεις του ίδιου προϊόντος (μια έκδοση για Windows, μια έκδοση για Macintosh, μια έκδοση για UNIX, μια έκδοση για Linux, κ.λπ.) ώστε όλοι οι χρήστες να μπορούν να εκτελέσουν το πρόγραμμα. Με την Java, μία μόνο έκδοση του προγράμματος μπορεί να εκτελεστεί σε όλες αυτές τις πλατφόρμες. Η Sun Microsystems μάλιστα υιοθέτησε ένα σλόγκαν που περιγράφει αυτή τη δυνατότητα των προγραμμάτων Java να λειτουργούν σωστά σε πολλαπλές πλατφόρμες: **«Write once, run anywhere»** (WORA, *γράψε μία φορά, εκτέλεσε παντού*).

Η Java είναι επίσης πιο απλή στη χρήση της από πολλές άλλες αντικειμενοστρεφείς γλώσσες. Η Java σχεδιάστηκε με βάση τη C++. Μολονότι καμία από τις δύο γλώσσες δεν διαβάζεται ή δεν γίνεται κατανοητή εύκολα με την πρώτη ματιά, η Java καταφέρνει να απαλείψει ορισμένα από τα πιο δυσνόητα χαρακτηριστικά της C++, όπως τους δείκτες και την πολλαπλή κληρονομικότητα.



Εικόνα 1-3
Το περιβάλλον της Java

Τύποι προγραμμάτων Java

Μπορείτε να γράψετε δύο είδη προγραμμάτων χρησιμοποιώντας την Java:

- Οι **μικροεφαρμογές** (applets) είναι προγράμματα που ενσωματώνονται σε μια ιστοσελίδα. Μπορείτε να μάθετε περισσότερα για τις μικροεφαρμογές σε ειδική ενότητα στο τέλος του κεφαλαίου.
- Οι **εφαρμογές Java** είναι αυτόνομα προγράμματα. Οι εφαρμογές Java μπορούν να χωριστούν σε δύο κατηγορίες, στις **εφαρμογές κονσόλας** (console applications), οι οποίες υποστηρίζουν έξοδο χαρακτήρων ή κειμένου σε οθόνη υπολογιστή, και στις **παραθυρικές εφαρμογές** (windowed applications), οι οποίες δημιουργούν ένα GUI με στοιχεία όπως μενού, γραμμές εργαλείων και παράθυρα διαλόγου. Οι εφαρμογές κονσόλας είναι οι εφαρμογές που δημιουργούνται πιο εύκολα απ' όλες τις άλλες – θα αρχίσετε να τις χρησιμοποιείτε στην επόμενη ενότητα.

ΔΥΟ ΑΛΗΘΕΙΕΣ ΚΙ ΕΝΑ ΨΕΜΑ

Περιγραφή χαρακτηριστικών της γλώσσας προγραμματισμού Java

1. Η Java αναπτύχθηκε έτσι ώστε να είναι αρχιτεκτονικά ουδέτερη, που σημαίνει ότι οποιοσδήποτε μπορεί να κατασκευάσει μια εφαρμογή χωρίς πολλές γνώσεις.
2. Αφού γράψετε ένα πρόγραμμα Java, ο μεταγλωττιστής μετατρέπει τον πηγαίο κώδικα σε δυαδικό πρόγραμμα, τον κώδικα byte.
3. Τα προγράμματα Java που ενσωματώνονται σε μια ιστοσελίδα ονομάζονται μικροεφαρμογές, ενώ τα αυτόνομα προγράμματα ονομάζονται εφαρμογές Java.

Η ψευδής πρόταση είναι η #1. Η Java αναπτύχθηκε έτσι ώστε να είναι αρχιτεκτονικά ουδέτερη, που σημαίνει ότι μπορείτε να χρησιμοποιείτε την Java για να γράφετε προγράμματα που θα εκτελούνται σε οποιαδήποτε πλατφόρμα.

Ανάλυση μιας εφαρμογής Java που παράγει έξοδο κονσόλας

Με μια πρώτη ματιά, ακόμα και η απλούστερη εφαρμογή Java περιλαμβάνει αρκετές εντολές και σύνταξη που θα μπορούσε να προκαλέσει σύγχυση. Δείτε την εφαρμογή στην Εικόνα 1-4. Αυτό το πρόγραμμα αποτελείται από επτά γραμμές και η μοναδική εργασία που κάνει είναι να εμφανίζει τη φράση «First Java application» στην οθόνη.

```
public class First
{
   public static void  main(String[] args)
   {
      System.out.println("First Java application");
   }
}
```

Εικόνα 1-4
Η κλάση First

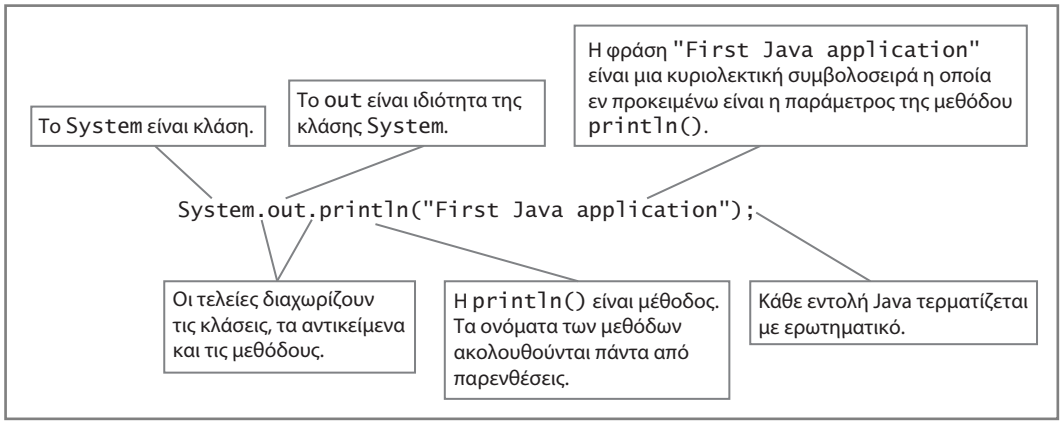
Σε κώδικα προγράμματος σε εικόνες του βιβλίου, οι δεσμευμένες λέξεις Java, όπως και τα true, false και null εμφανίζονται με διαφορετικό τρόπο από τα άλλα στοιχεία του προγράμματος. Παρακάτω σ' αυτό το κεφάλαιο παραθέτουμε μια ολοκληρωμένη λίστα με τις δεσμευμένες λέξεις της Java.

Ο κώδικας για κάθε ολοκληρωμένο πρόγραμμα που παρουσιάζουμε σ' αυτό το βιβλίο είναι διαθέσιμος σε ένα σύνολο από αρχεία για φοιτητές, που μπορείτε να κατεβάσετε στον υπολογιστή σας, ώστε να είστε σε θέση να εκτελείτε τα προγράμματα στον υπολογιστή σας.

Η εντολή που εμφανίζει την έξοδο

Μολονότι το πρόγραμμα στην Εικόνα 1-4 εκτείνεται σε αρκετές γραμμές, περιέχει μόνο μία εντολή προγραμματισμού Java. Η εντολή System.out.println("First Java application"); είναι αυτή που παράγει έργο σ' αυτό το πρόγραμμα. Όπως όλες οι εντολές Java, η εν λόγω εντολή τερματίζεται με ένα ερωτηματικό. Οι περισσότερες εντολές προγραμματισμού Java μπορούν να καταλαμβάνουν όσες γραμμές επιθυμείτε, αρκεί να εισάγετε τις αλλαγές γραμμών στα κατάλληλα σημεία. Για παράδειγμα, στο πρόγραμμα της Εικόνας 1-4, θα μπορούσατε να εισάγετε μια αλλαγή γραμμής πριν ή μετά την παρένθεση ανοίγματος ή πριν ή μετά την παρένθεση τερματισμού. Συνήθως όμως, αν η εντολή είναι μικρή, την τοποθετείτε σε μία μόνο γραμμή.

Το κείμενο «First Java application» είναι μια **κυριολεκτική συμβολοσειρά** (literal string) χαρακτήρων – μια σειρά χαρακτήρων που εμφανίζεται στην έξοδο ακριβώς όπως την εισάγετε. Οποιαδήποτε κυριολεκτική συμβολοσειρά στην Java γράφεται μέσα σε διπλά εισαγωγικά. Στην Java, μια κυριολεκτική συμβολοσειρά δεν μπορεί να χωριστεί και να εισαχθεί σε πολλαπλές γραμμές. Η Εικόνα 1-5 παρουσιάζει αυτή τη συμβολοσειρά και τα άλλα τμήματα της εντολής.



Εικόνα 1-5
Ανατομία μιας εντολής Java