ΚΕΦΑΛΑΙΟ 1

Δημιουργία προγραμμάτων Java

Μαθησιακά αποτελέσματα:

Όταν ολοκληρώσετε αυτό το κεφάλαιο, θα είστε ικανοί:

- **1.1** Να κατανοείτε τη βασική ορολογία προγραμματισμού
- **1.2** Να συγκρίνετε τον διαδικαστικό με τον αντικειμενοστρεφή προγραμματισμό
- **1.3** Να περιγράφετε τα χαρακτηριστικά της γλώσσας προγραμματισμού Java
- **1.4** Να αναλύετε μια εφαρμογής Java που παράγει έξοδο κονσόλας
- **1.5** Να μεταγλωττίζετε μια κλάση Java και να διορθώνετε συντακτικά σφάλματα
- **1.6** Να εκτελείτε μια εφαρμογή Java και να διορθώνετε λογικά σφάλματα
- **1.7** Να προσθέτετε σχόλια σε κλάση Java
- **1.8** Να δημιουργείτε εφαρμογή Java που παράγει έξοδο GUI
- **1.9** Να βρίσκετε και να συμβουλεύεστε υποστηρικτικό υλικό για περαιτέρω ανάπτυξη των δεξιοτήτων προγραμματισμού Java

1.1 Βασική ορολογία προγραμματισμού

Είναι γεγονός ότι βλέπουμε πολλούς υπολογιστές καθημερινά. Μπορεί να υπάρχει ένας φορητός υπολογιστής στο γραφείο, ενώ επίσης υπάρχουν υπολογιστές στο κινητό, στο αυτοκίνητο και ίσως στον θερμοστάτη, στο πλυντήριο και στην ηλεκτρική σκούπα. Μαθαίνοντας την ορολογία των υπολογιστών και κάποια βασικά στοιχεία προγραμματισμού, κατανοούμε επίσης τη λειτουργία των συσκευών, αναπτύσσουμε κριτική σκέψη και μαθαίνουμε να επικοινωνούμε καλύτερα. Θα αποκτήσετε όλες αυτές τις ικανότητες καθώς προχωράτε στη μελέτη του βιβλίου αυτού.

Τα συστήματα υπολογιστών αποτελούνται από το υλικό και το λογισμικό.

- Τα μηχανικά μέρη του υπολογιστή, όπως η οθόνη ή το πληκτρολόγιο, είναι το **υλικό** (hardware).
- Τα προγράμματα είναι το **λογισμικό** (software). **Πρόγραμμα υπολογιστή** (computer program), ή απλώς πρόγραμμα (program), είναι ένα σύνολο οδηγιών που γράφουμε για να πούμε στον υπολογιστή τι πρέπει να κάνει.

Το λογισμικό μπορεί να χωριστεί σε δύο ευρείες κατηγορίες:

- Το πρόγραμμα που εκτελεί μια εργασία για κάποιον χρήστη (όπως είναι ο υπολογισμός της μισθοδοσίας και η εκτύπωση της σχετικής απόδειξης, η επεξεργασία κειμένου ή ένα παιχνίδι) είναι το **λογισμικό μιας**

εφαρμογής (application software). Τα προγράμματα που συγκροτούν το λογισμικό μιας εφαρμογής λέγονται εφαρμογές (applications ή apps, για συντομία).

- Το πρόγραμμα που διαχειρίζεται τον ίδιο τον υπολογιστή (όπως τα Windows ή το Linux) είναι το **λογισμικό του 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 λογισμικών εφαρμογής.

Προγράμματα υπολογιστή μπορούν να γραφούν σε γλώσσες προγραμματισμού υψηλού ή χαμηλού επιπέδου:

- Οι **γλώσσες προγραμματισμού υψηλού επιπέδου** (high-level programming languages), όπως είναι οι Java, Visual Basic, C++, ή C#, επιτρέπουν τη χρήση ενός λεξιλογίου εύχρηστων όρων, όπως *διάβασε, γράψε* ή *πρόσθεσε*.
- Οι **γλώσσες προγραμματισμού χαμηλού επιπέδου** (low-level programming languages) είναι γλώσσες που βρίσκονται «κοντά» στα κυκλώματα του επεξεργαστή ενός υπολογιστή. Αυτό σημαίνει ότι οι γλώσσες χαμηλού επιπέδου είναι δύσχρηστες και πρέπει να προσαρμόζονται για κάθε τύπο μηχανής στην οποία εκτελείται ένα πρόγραμμα.

Όλα τα προγράμματα υπολογιστή, ακόμα και τα προγράμματα που χρησιμοποιούν γλώσσα προγραμματισμού υψηλού επιπέδου , μετατρέπονται τελικά στη γλώσσα προγραμματισμού πιο χαμηλού επιπέδου, που είναι η 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.

Σημείωση

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

Κάθε γλώσσα προγραμματισμού έχει το δικό της **συντακτικό** (syntax), δηλαδή κανόνες σχετικά με τον τρόπο με τον οποίο τα στοιχεία της γλώσσας συνδέονται σωστά ώστε να συνθέτουν χρήσιμες εντολές. Για παράδειγμα, ανάλογα με τη συγκεκριμένη γλώσσα υψηλού επιπέδου, θα μπορούσε να χρησιμοποιηθεί το ρήμα *εκτύπωσε* ή *γράψε* προκειμένου να εμφανιστεί ένα αποτέλεσμα. 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 μαθαίνουμε μια γλώσσα προγραμματισμού υπολογιστών, όπως την Java, τη C++ ή τη Visual Basic, στην πραγματικότητα μαθαίνουμε το λεξιλόγιο και τη σύνταξη για τη συγκεκριμένη γλώσσα.

Με τη χρήση μιας γλώσσας προγραμματισμού, οι προγραμματιστές γράφουν μια σειρά από **εντολές** (program statements), οι οποίες είναι παρόμοιες με προτάσεις στην αγγλική γλώσσα. Οι εντολές επιτελούν τις απαιτούμενες λειτουργίες του προγράμματος. Συχνά δίνονται με προστακτική μορφή στον υπολογιστή, όπως «εμφάνισε αυτή τη λέξη» ή «πρόσθεσε αυτούς τους δύο αριθμούς».

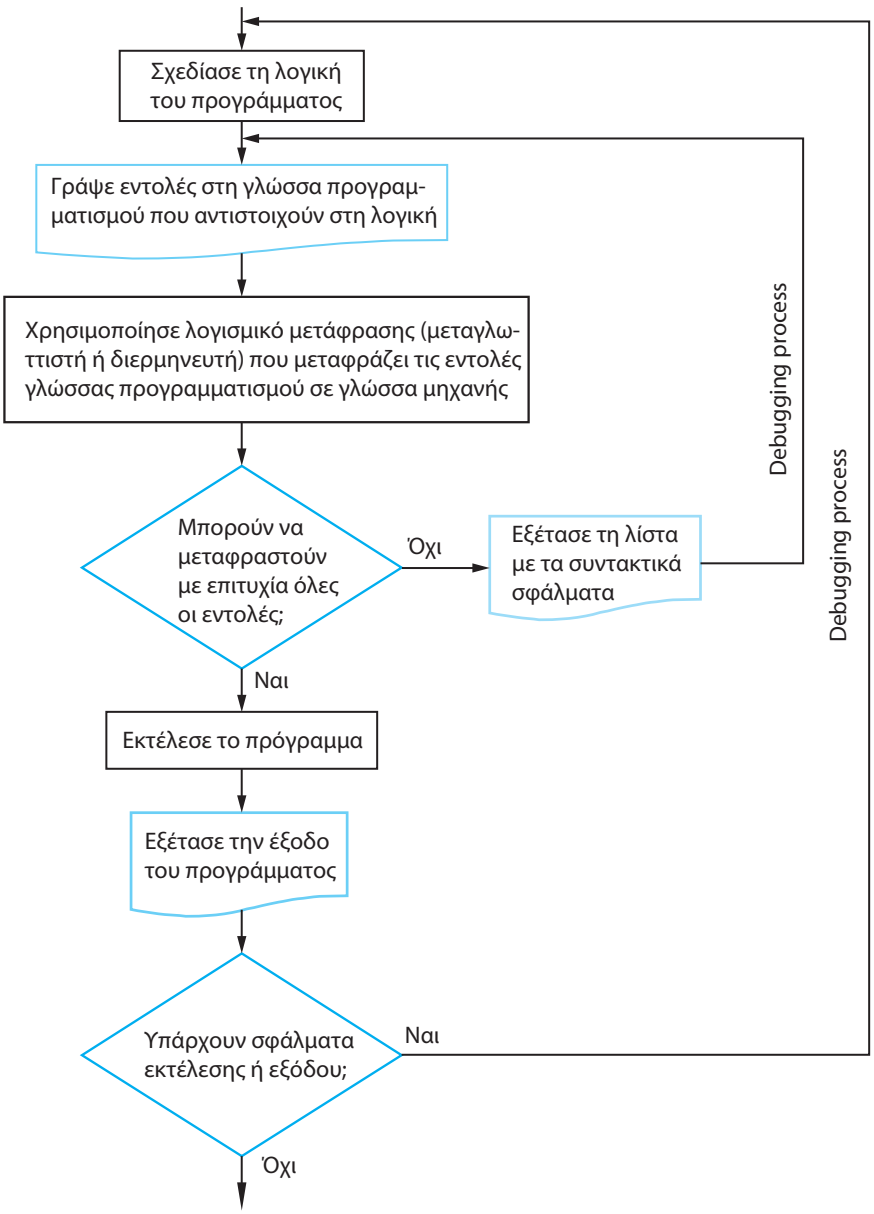
Μετά την εγγραφή των εντολών σε γλώσσα προγραμματισμού υψηλού επιπέδου, οι προγραμματιστές χρησιμοποιούν ένα άλλο πρόγραμμα που ονομάζεται **μεταγλωττιστής** (compiler) ή **διερμηνευτής** (interpreter) το οποίο μεταφράζει τις εντολές σε γλώσσα μηχανής. Ο μεταγλωττιστής μεταφράζει ολόκληρο το πρόγραμμα και μετά το **εκτελεί** (execute), ενώ ο διερμηνευτής μεταφράζει και εκτελεί κάθε εντολή αμέσως μόλις τη μεταφράσει.

Σημείωση

Η χρήση μεταγλωττιστή ή διερμηνευτή εξαρτάται πολλές φορές από τη γλώσσα προγραμματισμού που επιλέγεται. Για παράδειγμα, η C++ είναι μια μεταγλωττιζόμενη γλώσσα ένω η Visual Basic είναι μια διερμηνευόμενη γλώσσα. Κάθε τύπος μεταφραστή έχει τα πλεονεκτήματά 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 τη

Εικόνα 1-1 Η διαδικασία ανάπτυξης προγραμμάτων



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 παρουσιάσουμε αναλυτικότερα την εκσφαλμάτωση των προγραμμάτων Java παρακάτω σε αυτό το κεφάλαιο.

Όπως δείχνει η Εικόνα 1-1, θα μπορούσαμε να γράψουμε ένα πρόγραμμα, το οποίο μεταγλωττίζεται επιτυχώς (δηλαδή δεν περιέχει συντακτικά σφάλματα), αλλά ίσως εξακολουθεί να μην αποτελεί ορθό πρόγραμμα, επειδή μπορεί να περιέχει ένα ή περισσότερα 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 θέλουμε να τις διαιρέσουμε ή η έξοδος αποτελεσμάτων πριν αποκτηθεί η κατάλληλη είσοδος. Κατά την ανάπτυξη ενός προγράμματος μεγάλου μεγέθους, θα πρέπει να σχεδιάζουμε τη λογική του πριν αρχίσουμε να γράφουμε 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 δομής του προγράμματός μας (όταν την εξέταση αυτή την εκτελεί μια ομάδα προγραμματιστών από κοινού, αυτό ονομάζεται δομημένη περιήγηση), αλλά μερικές φορές τα λογικά σφάλματα αναγνωρίζονται μόνο κατά την εξέταση της εξόδου ενός προγράμματος. Για παράδειγμα, αν γνωρίζουμε ότι ο μισθός ενός υπαλλήλου περιέχει την τιμή 4.000, αλλά όταν εξετάζουμε την έξοδο του προγράμματος μισθοδοσίας διαπιστώνουμε ότι περιέχει την τιμή 40, αντιλαμβανόμαστε 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, καθώς και να μεταφράσει και να εκτελέσει το πρόγραμμα ξανά.

Σημείωση

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.

Σημείωση

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

Δύο αλήθειες κι ένα ψέμα | Βασική ορολογία 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 επιτρέπει τη χρήση ενός λεξιλογίου εύχρηστων όρων αντί για τις ακολουθίες ανοιχτών και κλειστών διακοπτών (λογικών κυκλωμάτων) που εκτελούν τις αντίστοιχες εργασίες.

2. Προκύπτει συντακτικό σφάλμα όταν χρησιμοποιούμε τη γλώσσα προγραμματισμού με λανθασμένο τρόπο. 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 μπορεί να υπάρχουν λογικά σφάλματα σε ένα πρόγραμμα χωρίς συντακτικά λάθη.

1.2 Σύγκριση διαδικαστικού και αντικειμενοστρεφούς προγραμματισμού

Όλοι οι προγραμματιστές έρχονται αντιμέτωποι με συντακτικά και λογικά σφάλματα με τον ίδιο περίπου τρόπο, αλλά μπορούν να ακολουθήσουν διαφορετικές προσεγγίσεις στη διαδικασία προγραμματισμού. Ο διαδικαστικός και ο αντικειμενοστρεφής προγραμματισμός περιγράφουν δύο διαφορετικές προσεγγίσεις στη συγγραφή προγραμμάτων.

Διαδικαστικός προγραμματισμός

Ο **διαδικαστικός προγραμματισμός** (procedural programming) είναι το είδος προγραμματισμού στον οποίο οι πράξεις εκτελούνται η μία μετά την άλλη, διαδοχικά. Ο τυπικός διαδικαστικός προγραμματισμός ορίζει και χρησιμοποιεί επώνυμες θέσεις της μνήμης του υπολογιστή – κάθε τέτοια θέση, όπου αποθηκεύονται προσωρινά τα δεδομένα, ονομάζεται **μεταβλητή** (variable). Για παράδειγμα, τα δεδομένα ενός προγράμματος μισθοδοσίας μπορούν να διαβαστούν από μια συσκευή εισόδου και να αποθηκευτούν σε μια θέση μνήμης που ο προγραμματιστής έχει ονομάσει rateOfPay. Η τιμή της μεταβλητής μπορεί να χρησιμοποιηθεί σε μια αριθμητική εντολή, ως η βάση για μια απόφαση, να σταλεί σε μια συσκευή εξόδου ή να εκτελέσει άλλες λειτουργίες. Τα δεδομένα που είναι αποθηκευμένα σε μια μεταβλητή μπορούν να αλλάξουν, ή να ποικίλλουν, κατά τη διάρκεια της εκτέλεσης ενός προγράμματος.

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 την ονομασία calculateFederalWithholding (). (Στο βιβλίο μετά από κάθε ονομασία διαδικασίας θα εμφανίζονται τα σύμβολα της παρένθεσης). Καθώς ένα διαδικαστικό πρόγραμμα εκτελεί τις εντολές του, μπορεί κάποιες φορές να σταματήσει τη ροή του προγράμματος για να καλέσει μια διαδικασία (call a procedure). Όταν το πρόγραμμα **καλεί μια διαδικασία**, 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 κλήσεις διαδικασιών. Οι διαδικασίες αναφέρονται επίσης ως *ενότητες, μέθοδοι, λειτουργίες* και *υπορουτίνες*. Οι χρήστες διαφορετικών γλωσσών προγραμματισμού συνηθίζουν να χρησιμοποιούν διαφορετική ορολογία. Όπως θα αναφέρουμε αργότερα σε αυτό το κεφάλαιο, οι προγραμματιστές Java χρησιμοποιούν συχνότερα τον όρο *μέθοδος*.

Αντικειμενοστρεφής προγραμματισμός

Ο αντικειμενοστρεφής προγραμματισμός είναι μια επέκταση του διαδικαστικού προγραμματισμού, στην οποία υιοθετείται 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 δείγματα αυτών των κλάσεων
- Τη δημιουργία εφαρμογών που χειρίζονται ή χρησιμοποιούν αυτά τα αντικείμενα