4 Σχεδίαση προγράμματος

4.1 Περί του τρόπου που ο προγραμματιστής δημιουργεί μια δομή για το πρόγραμμά του [5, σελ. 1-6]

- Από την πλευρά του χρήστη ή γενικότερα του πελάτη που έχει παραγγείλει την ανάπτυξη κάποιου προγράμματος, τα κριτήρια με βάση τα οποία αξιολογείται ένα πρόγραμμα είναι: (α) αξιοπιστία, (β) απλότητα, (γ) φιλικότητα, (δ) ευελιξία, και (ε) ταχύτητα.
  - Σημειώστε ότι η ταχύτητα ιεραρχείται ως το λιγότερο σημαντικό κριτήριο.
- Από την πλευρά της εταιρίας που αναπτύσσει ένα πρόγραμμα κύριοι στόχοι είναι: (α) αξιοπιστία, (β) εύκολη κατανόηση του κώδικα (για να είναι ευκολότερη η συντήρηση), (γ) κόστος < όφελος, (δ) προσαρμοστικότητα (για να μπορεί να απευθύνεται σε μεγαλύτερο μερίδιο αγοράς) και απόκρυψη πληροφορίας (information hiding).
  - Στις αντικειμενοστρεφείς γλώσσες προγραμματισμού, η απόκρυψη πληροφορίας επιτυγχάνεται μέσω των κλάσεων όπου γνωστοποιείται η δημόσια διεπαφή (ονόματα κλάσεων, σταθερών, μεθόδων, καθώς και παράμετροί τους και το τι κάνουν), ενώ ο τρόπος υλοποίησης παραμένει κλειστός[23].
- Γιατί να υπάρχει σχεδίαση προγράμματος;
  - Στοχεύει στην βελτίωση της αξιοπιστίας του προγράμματος. Η μειωμένη αξιοπιστία είναι εγγενές χαρακτηριστικό των πολύπλοκων συστημάτων.
  - Στοχεύει στη μείωση του κόστους. Η σχεδίαση είναι ένα πρόσθετο στάδιο στη διαδικασία ανάπτυξης λογισμικού και σε πρώτη θεώρηση αυξάνει το κόστος του έργου, αφού απαιτεί πόρους. Ωστόσο, μειώνει το συνολικό κόστος ιδιοκτησίας (Total Cost of Ownership, TCO), καθώς η διορθωτική και τροποποιητική συντήρηση του συστήματος γίνεται ευκολότερα.
- Χαρακτηριστικά σχεδίασης που είναι ανεξάρτητα από συγκεκριμένο πεδίο εφαρμογής:
  - Είναι δημιουργική.
  - Χρησιμοποιεί επιστημονική προσέγγιση όπου και όποτε χρειάζεται.
  - Οδηγεί σε τερπνό ή / και ωφέλιμο αποτέλεσμα.
- Μέθοδοι αποσύνθεσης (decomposition)
  - Πάνω-προς-κάτω (top-down): από το στόχο προκύπτουν τα γενικά στάδια. Αυτά αναλύονται περαιτέρω μέχρι να φτάσουμε σε απλές εργασίες που μπορούν να κωδικοποιηθούν. Το πρόγραμμα αναπτύσσεται ακολουθώντας δενδρική δομή, όπου στην κορυφή είναι ο κορμός που αντιστοιχεί στο άρθρωμα με το οποίο ξεκινάει η εκτέλεση του προγράμματος και από κάτω σαν κλαδιά είναι τα αρθρώματα που καλούνται για να υλοποιήσουν την λειτουργικότητα. Επικρατούσα, αλλά όχι μοναδική μέθοδος.
  - Κάτω-προς-πάνω (bottom-up): Ο προγραμματιστής ξεκινά υλοποιώντας τις βασικές λειτουργίες που πιστεύει ότι θα του χρειαστούν και μετά προσπαθεί να τις συνθέσει σε πιο σύνθετες λειτουργίες. Πάλι το πρόγραμμα έχει δενδρική δομή, αλλά η ανάπτυξή του ξεκινάει από τα φύλλα του και ανεβαίνει προς τα πάνω, προς τον κορμό.
- Διακρίνουμε επίπεδα λεπτομέρειας (granularity) στις ενότητες που οργανώνεται το πρόγραμμα:
  - Όταν επικεντρώνουμε την προσοχή μας στα αρθρώματα (υποπρογράμματα, κλάσεις, κλπ) στα οποία θα δομηθεί το πρόγραμμα, μιλάμε για **αρχιτεκτονική ή δομική σχεδίαση** (architectural design).
  - Όταν κατεβαίνουμε ένα σκαλί παρακάτω σε επίπεδο ανάλυσης, δηλαδή στο επίπεδο των εντολών πηγαίου κώδικα, μιλάμε για **λεπτομερή σχεδίαση** (detailed design).
- Ανάλογα με το πεδίο εφαρμογής του πληροφοριακού συστήματος που αναπτύσσεται, επιλέγεται η καταλληλότερη μεθοδολογία σχεδίασης (δείτε Εικόνα 57). Συχνά η επιλογή εξαρτάται από τις γνώσεις και την εμπειρία των ανθρώπων που συμμετέχουν στο έργο, ή και τα εργαλεία ανάπτυξης που χρησιμοποιούν (για παράδειγμα, γλώσσα προγραμματισμού, περιβάλλον ανάπτυξης, βάση δεδομένων).
- **Μέθοδος Jackson**: Αρχή της μεθόδου αυτής είναι ότι η δομή του προγράμματος πρέπει να ακολουθεί τη δομή των αρχείων πάνω στα οποία λειτουργεί το πρόγραμμα. Οι προδιαγραφές του προγράμματος καθορίζουν πώς τα αρχεία εισόδου μετασχηματίζονται στα αρχεία εξόδου.

Τεχνολογία	Βασικές έννοιες	Δομές
Δομημένος προγραμματισμός	Επιλογή δομών ελέγχου ροής προγράμματος	While, Repeat και άλλες επαναλήψεις, ομαδοποίηση εντολών (blocks)
Αρθρωτός προγραμματισμός	Απόκρυψη πληροφορίας	Αρθρώματα με καλώς ορισμένα πρωτόκολλα
Αφαίρεση δεδομένων	Απόκρυψη της αναπαράστασης των δεδομένων	Αφαιρετικοί τύποι δεδομένων (abstract data types)
Αντικειμενοστρεφής προγραμματισμός	Αντικείμενα με κληρονομικότητα	Κλάσεις, αντικείμενα, πολυμορφισμός

Εικόνα 57 Σύγκριση μεθοδολογιών σχεδίασης

[23] Βέβαια, με την εξάπλωση του μοντέλου του ανοικτού κώδικα (open source), αρκετά πληροφοριακά συστήματα παρέχουν πρόσβαση σε όλο τον πηγαίο κώδικά τους. Μάλιστα μερικές επιχειρήσεις / κυβερνήσεις απαιτούν να έχουν πρόσβαση σε συγκεκριμένα τουλάχιστον τμήματα του κώδικα πληροφοριακών συστημάτων που χρησιμοποιούν.

- Ταιριάζει ιδιαίτερα σε συστήματα επεξεργασίας δεδομένων (data processing) και ειδικότερα όπου η επεξεργασία γίνεται σειριακά.

4.2 Αντικειμενοστρεφής ανάπτυξη συστημάτων

- Η προσέγγιση αυτή αναλύεται στην αντικειμενοστρεφή ανάλυση που προσδιορίζει τα αντικείμενα που εμπλέκονται στις επιχειρηματικές διαδικασίες που πρέπει να υλοποιηθούν, στον αντικειμενοστρεφή σχεδιασμό που αποτυπώνει τα κύρια και βοηθητικά αντικείμενα που θα υλοποιηθούν και τις αλληλεπιδράσεις τους, και τέλος στον αντικειμενοστρεφή προγραμματισμό.
  - Εργαλεία που υποστηρίζουν αυτές τις φάσεις είναι οι διάφοροι τύποι διαγραμμάτων UML, οι αντικειμενοστρεφείς βάσεις δεδομένων και οι αντικειμενοστρεφείς γλώσσες προγραμματισμού.
- Η διαδικασία της αντικειμενοστρεφούς σχεδίασης ξεκινάει με την αναγνώριση των βασικών αντικειμένων που εμφανίζονται στο χώρο του προβλήματος που αντιμετωπίζεται. Έπειτα προσπαθούμε να διακρίνουμε τις σχέσεις των διαφόρων αντικειμένων μεταξύ τους. Οι δύο βασικότερες σχέσεις είναι η ιεραρχία γενίκευσης / ειδίκευσης (generalization / specialization) που ονομάζεται και ιεραρχία is-a και η ιεραρχία όλου-τμήματος (whole – part) που ονομάζεται και ιεραρχία has-a (δείτε και σελ. 27, παράγραφος «Μηχανισμός Μοντελοποίησης»).
  - Παράδειγμα: ένα όχημα είναι ένα αυτοκίνητο ή μοτοσικλέτα ή φορτηγό. Με τη σειρά του ένα αυτοκίνητο μπορεί να είναι ιδιωτικής ή δημόσιας χρήσης. Στην τελευταία περίπτωση μπορεί να είναι ταξί ή ενοικιαζόμενο, κοκ. Τα παραπάνω είναι παραδείγματα ιεραρχιών is-a που σταδιακά εξειδικεύουν το γενικό αντικείμενο «όχημα» προσθέτοντάς του χαρακτηριστικά και διαμορφώνοντάς του μια πιο συγκεκριμένη συμπεριφορά. Από την άλλη μεριά ένα όχημα έχει ρόδες, κινητήρα, σασί, κτλ. Ένας κινητήρας έχει κυλίνδρους, κοκ. Αυτά είναι παραδείγματα μιας ιεραρχίας has-a.

4.3 Σχολιασμός και τεκμηρίωση προγράμματος

- Συνήθως η τεκμηρίωση (documentation) συγγράφεται στο τέλος, αλλά αυτό είναι κακή πρακτική.
  - Στα σύγχρονα περιβάλλοντα ανάπτυξης ενσωματώνονται εργαλεία που παράγουν και συντηρούν την τεκμηρίωση του υπό ανάπτυξη προγράμματος. Δηλαδή η τεκμηρίωση αλλάζει παρακολουθώντας τις αλλαγές που γίνονται στον εκτελέσιμο κώδικα, και όχι εκ των υστέρων.

```
package somepackage;

/**
 * <p>Title: Πακέτο επίδειξης τεκμηριωσης</p>
 * <p>Description: Δεν κάνει τίποτα</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * @version 2.5
 */

/**
 * <p>Title: Μια κλάση του πακέτου</p>
 * <p>Description: Ούτε κι αυτή κάνει τίποτα</p>
 * @version 1.4
 */
public class KenhKlash {
  public static void main(String[] args) {
  }
}
```

somepackage

Class KenhKlash

public class **KenhKlash**

Title: Μια κλάση του πακέτου

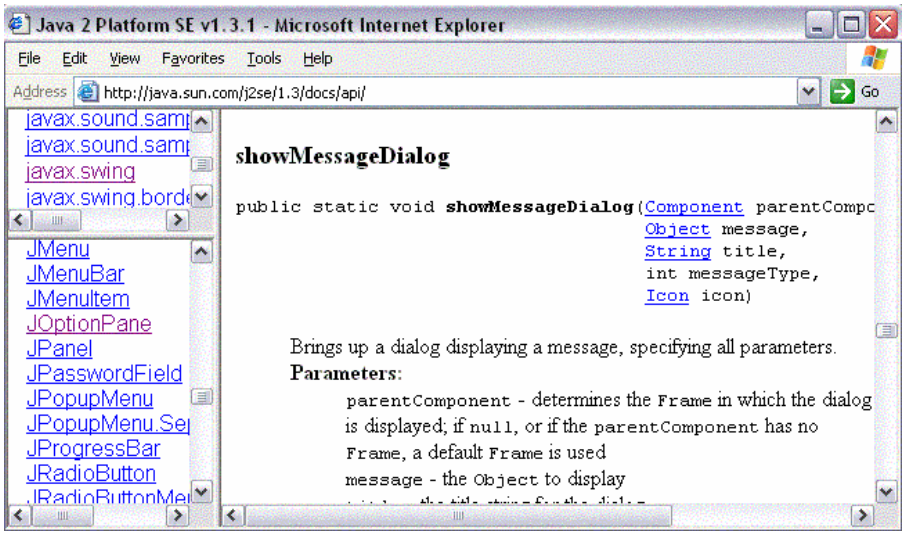
Description: Ούτε κι αυτή κάνει τίποτα

Version:

1.4

Εικόνα 58 Οι ετικέτες τεκμηρίωσης και το παραγόμενο αρχείο HTML

- Στη Java υπάρχει η δυνατότητα αυτοματοποιημένης παραγωγής της τεκμηρίωσης μέσω του εργαλείου javadoc. Αυτό το εργαλείο διαπερνά τον πηγαίο κώδικα και ανιχνεύει ειδικά διαμορφωμένα σχόλια που ξεκινούν με `/**`, είναι τοποθετημένα σε συγκεκριμένα σημεία του πηγαίου προγράμματος και περιλαμβάνουν τυποποιημένες ετικέτες, όπως `@version`, `@author`, `@param`, `@return`.[24] Από αυτά τα σχόλια φτιάχνει τυποποιημένες ιστοσελίδες HTML που περιέχουν τεκμηρίωση.
  - Με αυτό τον τρόπο έχει κατασκευαστεί και η «επίσημη» τεκμηρίωση του Java API, που είναι γνωστή ως Java Reference. Είναι σημαντικό κάθε προγραμματιστής



Εικόνα 59 Τεκμηρίωση του Java API

[24] Στο περιβάλλον ανάπτυξης Borland JBuilder, γράφοντας `/**` και Enter, πάνω από μια κλάση ή μια μέθοδο, εισάγεται αυτόματα ο σκελετός του κατάλληλου javadoc σχολίου.

Java να ανατρέχει στην τεκμηρίωση για να βρει πληροφορίες για τις βιβλιοθήκες – πακέτα που χρειάζεται να κάνει import στον κώδικά του, για τις κλάσεις και για τις μεθόδους, όπως τύπος παραμέτρων, λειτουργία που εκτελούν, κτλ.

- Η χρήση του javadoc επιβάλλεται όταν αναπτύσσετε κώδικα που πρόκειται να διανείμετε σε άλλους προγραμματιστές. Τα σχόλια javadoc εισάγονται πριν από την επικεφαλίδα της ενότητας την οποία τεκμηριώνουν, είτε αυτή είναι κλάση, είτε μέθοδος, είτε όνομα μεταβλητής / σταθεράς.

4.4 Διάρκεια ζωής και εμβέλεια των ονομάτων

- **Διάρκεια ζωής** (lifetime ή extent) είναι το διάστημα για το οποίο έχει εκχωρηθεί κάποιος χώρος στη μνήμη. Στη Java η ζωή ενός ονόματος ξεκινά την ώρα εκτέλεσης της δήλωσης του ονόματος και διαρκεί ως τον τερματισμό της εκτέλεσης του αρθρώματος στο οποίο ανήκει το όνομα (dynamic extent). Κατά την έξοδο από το άρθρωμα, το περιβάλλον εκτέλεσης επιστρέφει τον χώρο που είχε εκχωρηθεί για τα ονόματα στην ελεύθερη μνήμη. Κατ' εξαίρεση, τα ονόματα με στατική διάρκεια ζωής (static extent) διαρκούν έως τον τερματισμό του προγράμματος.
  - Οι περισσότερες γλώσσες προγραμματισμού έχουν υιοθετήσει τη δυναμική διάρκεια ζωής, καθώς επιτυγχάνει εξοικονόμηση μνήμης και ο προγραμματιστής δεν χρειάζεται να διαχειρίζεται την απελευθέρωση της δεσμευμένης μνήμης.
  - Ωστόσο, υπάρχουν προγράμματα που απαιτούν δυναμικές δομές δεδομένων, εάν οι απαιτήσεις σε αποθηκευτικό χώρο είναι ευμετάβλητες και τα δεδομένα μπορεί να συρρικνωθούν ή να επεκταθούν απότομα. Σε τέτοιες περιπτώσεις, ο προγραμματιστής αναλαμβάνει τον έλεγχο της δέσμευσης και αποδέσμευσης μνήμης.
- **Εμβέλεια** (scope) ενός ονόματος είναι το τμήμα του προγράμματος στο οποίο μπορούμε να προσπελάσουμε το όνομα. Μια τοπική μεταβλητή έχει εμβέλεια τη μέθοδο που είναι δηλωμένη (local scope). Τα ονόματα μεθόδων ή μεταβλητών στιγμιότυπου έχουν εμβέλεια ολόκληρη την κλάση τουλάχιστον (global scope).
  - Για λόγους ασφάλειας και εύκολης συντήρησης του προγράμματος προσπαθούμε να δίνουμε στα ονόματα την ελάχιστη εμβέλεια που χρειάζεται, δηλαδή τους επιτρέπουμε να είναι "ορατά" μόνον εκεί που είναι απαραίτητο. Ειδικά οι παγκόσμιες μεταβλητές είναι επικίνδυνη πρακτική.
  - Για τα ονόματα επιπέδου κλάσης, η εμβέλεια καθορίζεται με τις δεσμευμένες λέξεις `private`, `protected`, `public`. Ιδιωτικά (private) ονόματα έχουν την ελάχιστη εμβέλεια και είναι "ορατά" μόνο στο εσωτερικό της κλάσης. Αντίθετα, τα δημόσια (public) ονόματα είναι προσπελάσιμα από οπουδήποτε. Ένα ενδιάμεσο επίπεδο εμβέλειας προσφέρει η δήλωση `protected` – τα ονόματα είναι ορατά μέσα στην κλάση τους (τοπική εμβέλεια) και σε όλες τις κλάσεις που κληρονομούν την κλάση τους.
  - Η διάρκεια ζωής και η εμβέλεια ενός ονόματος δεν συμπίπτουν απαραίτητα. Για παράδειγμα, μια τοπική μεταβλητή σε μια μέθοδο μπορεί να επικαλύψει μια μεταβλητή στιγμιότυπου (instance variable). Παρότι η διάρκεια ζωής της μεταβλητή στιγμιότυπου δεν έχει λήξει όσο βρισκόμαστε στο εσωτερικό της μεθόδου, αναφορά στο όνομά της προκαλεί προσπέλαση στην τοπική μεταβλητή.
  - Για να αναφερθούμε σε ένα όνομα (μεταβλητή ή μέθοδο) του τρέχοντος αντικειμένου, χρησιμοποιούμε το προσδιοριστικό `this`[25]. Το `this` χρησιμοποιείται συχνά στους κατασκευαστές όπου υπάρχουν ομώνυμη παράμετρος που παρέχει την αρχική τιμή για το αντικείμενο και μεταβλητή στιγμιότυπου (δείτε το παράρτημα 2.15 Κλασματικοί αριθμοί).

4.5 Στυλ προγραμματισμού

- Μερικές συμβάσεις που βελτιώνουν τα χαρακτηριστικά του προγράμματος:
  - Χρησιμοποιήστε σταθερές και δηλώστε τις στην αρχή του προγράμματος
  - Στη δήλωση των μεταβλητών καταγράψτε τη χρήση τους και στην επικεφαλίδα των υπορουτινών τη λειτουργία τους
  - Ακολουθήστε ενιαία σύμβαση για την ονομασία μεταβλητών, πχ `sFirstName`, `iCountNumbers`, μεθόδων, πχ `getFinalValue`, `computeMaximum`, κλάσεων, πχ `MyTimer` και σταθερών `MAX_NUMBER_OF_STUDENTS`.
  - Χρησιμοποιήστε οδόντωση για να είναι ευκρινές το σώμα των εντολών.
- Κανόνες γραφής προγραμμάτων από την αναφορά αρ. 8, Kernigham, Plauger, *The elements of programming style,* McGraw-Hill, 2nd ed, 1978
  - Γράψε με σαφήνεια – μην προσπαθείς να είσαι πολύ έξυπνος.
  - Πες αυτό που θέλεις, απλά και άμεσα.
  - Χρησιμοποίησε όσο το δυνατόν περισσότερο προγράμματα βιβλιοθηκών – μην εφευρίσκεις τον τροχό.
  - Απόφευγε παγκόσμιες μεταβλητές.
  - Μην θυσιάζεις τη σαφήνεια για την ταχύτητα ή την οικονομία χώρου.
  - Να αντικαθιστάς παρόμοιες εκφράσεις που χρησιμοποιούνται συχνά με κλήσεις μιας συνάρτησης.
  - Βάζε παρενθέσεις αν δεν είσαι σίγουρος.
  - Να χρησιμοποιείς υποπρογράμματα και να ορίζεις σε σαφήνεια τις συνδέσεις τους.
  - Να ελέγχεις αν τα δεδομένα πληρούν τις προδιαγραφές που έχουν τεθεί, προτού να τα χρησιμοποιήσεις.
  - Να αρχικοποιείς τις μεταβλητές πριν να χρησιμοποιήσεις την τιμή τους.
  - Να δοκιμάζεις το πρόγραμμα όχι μόνο για τις τυπικές, αλλά και για τις οριακές εισόδους.

[25] Το `this` δεν μπορεί να χρησιμοποιηθεί με στατικά ονόματα (μεθόδους και μεταβλητές κλάσης).

4.6 Έλεγχος σύνταξης

- Μια γλώσσα (προγραμματισμού) ορίζεται τυπικά με βάση το συντακτικό της. Μια από τις λειτουργίες του μεταγλωττιστη είναι η συντακτική ανάλυση που ελέγχει αν το κείμενο του πηγαίου προγράμματος ακολουθεί του κανόνες σύνταξης της γλώσσας.
- Αλφάβητο μιας γλώσσας είναι οι αποδεκτοί χαρακτήρες που μπορούν να εμφανίζονται στον πηγαίο κώδικα.
  - Σε άλλες γλώσσες έχει σημασία η γραφή με κεφαλαία / πεζά γράμματα (πχ. Java, C) και σε άλλες όχι (Basic, Lisp, Pascal). Στη FORTRAN παλιότερα τα προγράμματα γραφόταν μόνο με κεφαλαία.
  - Οι περισσότερες γλώσσες θεωρούν ότι ο πηγαίος κώδικας είναι γραμμένος σε ASCII˙ η Java δέχεται χαρακτήρες Unicode.
- Κατά τη λεκτική ανάλυση αναγνωρίζονται οι «λέξεις», δηλαδή οι συνδυασμοί γραμμάτων που αποτελούν το πρόγραμμα. Βάση της λεκτικής ανάλυσης είναι τα κυριολεκτήματα. Κυριολέκτημα (literal) είναι μια λέξη που δεν αποαναφεροποιείται, πχ `13.5`, `"qwe rty"`, `1.2E-4`. Τα κυριολεκτήματα διαχωρίζονται από κενό διάστημα (white space): χαρακτήρες κενού (spaces), στηλοθέτες (tabs) ή χαρακτήρες αλλαγής γραμμής.
  - Διακρίνουμε επίσης τους τελεστές (operators) και τα ορίσματά τους (operands)
  - Δεσμευμένες λέξεις (reserved words) είναι τα κυριολεκτήματα που έχουν ειδικό νόημα στην κάθε γλώσσα και δεν μπορούν να χρησιμοποιηθούν για ονόματα. Παραδείγματα δεσμευμένων λέξεων της Java: **`if`**, **`while`**, **`char`**, **`switch`**, **`return`**.
  - Κανόνες για ονόματα (identifiers): συνήθως ξεκινούν από γράμμα και ακολουθούν γράμματα ή ψηφία. Επιτρέπονται και χαρακτήρες όπως `_ $`, ενώ απαγορεύονται τα σημεία στίξης. Σε μερικές γλώσσες επιβάλλεται όριο στο μήκος των ονομάτων, ενώ σε άλλες τα ονόματα έχουν απεριόριστο μήκος, αλλά ο μεταγλωττιστής λαμβάνει υπόψη του μόνον συγκεκριμένο αριθμό από τους πρώτους χαρακτήρες του ονόματος.
  - Οι προγραμματιστές κατά σύμβαση προθέτουν του ονόματος μιας μεταβλητής μια ένδειξη του τύπου της. Για παράδειγμα `iCounter`, `sEponymo`, `fTaxythta`, `bAnammeno`
- Μετά τη συντακτική ανάλυση έρχεται η σημασιολογική ανάλυση που αποδίδει το νόημα του προγράμματος (μέσω αναγωγής σε άλλη γλώσσα). Πρακτικά, το πρόγραμμα μεταφράζεται σε εντολές κάποιας απλούστερης γλώσσας που έχει αποδεδειγμένη σημασιολογία ή εκτελείται σε κάποια πραγματική ή προσομοιωνόμενη μηχανή.

4.7 Μετάφραση προγράμματος

- Ο πηγαίος κώδικας μεταφράζεται σε μια γλώσσα μηχανής μιας υπαρκτής (πχ για επεξεργαστή x86 ή Alpha) ή ιδεατής μηχανής (πχ Java Runtime Environment).
  - Ένα ενδιάμεσο βήμα μπορεί να είναι η μετάφραση σε συμβολική γλώσσα (assembly). Η assembly δεν εκτελείται παρά μόνον αν μετατραπεί σε γλώσσα μηχανής. Για την κατανόηση της assembly ο προγραμματιστής πρέπει να γνωρίζει την αρχιτεκτονική του υπολογιστή στόχου, πχ μέγεθος και πλήθος καταχωρητών (registers).
  - Οι optimizing compilers προσπαθούν να κατασκευάσουν πιο αποδοτικό κώδικα για γλώσσα μηχανής εκμεταλλευόμενοι ειδικές εντολές του μικροεπεξεργαστή ή των περιφερειακών διατάξεων και ιδιαιτερότητες του προγράμματος (πχ loop unfolding).

```
MEIWSH = FOROS + KRATHSEIS | LOAD FOROS     | 0101 10110010
                           | ADD KRATHSEIS  | 0100 01010101
                           | STORE MEIWSH   | 0001 00000001
```

Εικόνα 60 Πηγαίος κώδικας, assembly και γλώσσα μηχανής

4.8 Μεταφερσιμότητα προγράμματος

- Ανάπτυξη και εκτέλεση σε διαφορετικές υπολογιστικές πλατφόρμες
  - Παράδειγμα Java τηλεοράσεις και Linux ραδιοκασετόφωνα
- Εκτέλεση σε διαφορετικά λειτουργικά συστήματα
  - Η γλώσσα προγραμματισμού (πχ Java) υποβοηθά στην μεταφερσιμότητα, αλλά ο προγραμματιστής κατά περίπτωση πρέπει να αντιμετωπίζει ο ίδιος παραλλαγές στον μεταγλωττιστή ή στο περιβάλλον εκτέλεσης.
- Ιδεατές μηχανές
  - Παραδείγματα με Java Runtime Environment (byte code) και Visual Basic Virtual Machine. Η VB, εκτός από τη διερμηνεία του πηγαίου κώδικα, δίνει τη δυνατότητα μεταγλώττισης σε p-κώδικα και σε εγγενή κώδικα μηχανής.
  - Μειώνουν την ταχύτητα γιατί το πρόγραμμα δεν εκτελείται εγγενώς
  - Μεταγλώττιση Just-In-Time: Από τα παλιά χρόνια, τα προγράμματα σε διάτρητες καρτέλες υποβάλλονταν προς εκτέλεση και μεταφράζονταν πριν να εκτελεστούν. Μετά την εκτέλεση ο (εκτελέσιμος ή ενδιάμεσος) κώδικας χανόταν. Τώρα, επειδή δεν είναι γνωστή η μηχανή στόχος, αποστέλλεται το πηγαίο πρόγραμμα που μεταγλωττίζεται μόλις πριν την εκτέλεσή του στον πελάτη. Μετά την εκτέλεση, τόσο το πηγαίο, όσο και το εκτελέσιμο χάνονται.
  - Πληρέστεροι έλεγχοι runtime, πχ προσπέλαση στοιχείων πινάκων εκτός ορίων

4.9 Ασκήσεις

Α46. [Μέθοδος Jackson] Φτιάξτε πρόγραμμα που θα εμφανίζει ένα κείμενο ώστε η κάθε του γραμμή να έχει το πολύ *n* χαρακτήρες, όπου το *n* θα ορίζεται από το χρήστη.

Α47. [Μέθοδος Jackson] Φτιάξτε πρόγραμμα που θα δέχεται ως είσοδο τις ώρες ανατολής και δύσης του ηλίου για κάθε μήνα και θα τυπώνει ιστόγραμμα με τη διάρκεια της ημέρας.

Α48. [Μέθοδος Jackson] Φτιάξτε πρόγραμμα που θα δέχεται ως είσοδο τα αποτελέσματα ενός τυχαίου πειράματος (πχ το ρίξιμο ενός ζαριού) και θα υπολογίζει τη συχνότητα εμφάνισης του κάθε γεγονότος.

Α49. Χρησιμοποιήστε αναδρομή αντί για επανάληψη για να υπολογίσετε την αντιστροφή της συμβολοσειράς ως `reverse(s) = tail(x) . head(x)`, όπου head είναι ο πρώτος χαρακτήρας και tail όλη η συμβολοσειρά εκτός από τον πρώτο της χαρακτήρα.

Α50. Τροποποιήστε το παραπάνω αναδρομικό πρόγραμμα προσθέτοντας μια εντολή στην αρχή της αναδρομικής συνάρτησης οι οποία θα τυπώνει την τιμή των παραμέτρων και θα προσθέτει ένα επίπεδο οδόντωσης. Έτσι θα είναι άμεσα αντιληπτή η κατανόηση της ακολουθίας των αναδρομικών κλήσεων.

Α51. Τοποθετήστε οκτώ βασίλισσες σε μια σκακιέρα σε τέτοιες θέσεις ώστε καμιά να μην απειλείται, δηλαδή να μην υπάρχουν δύο στην ίδια γραμμή, στην ίδια στήλη ή στην ίδια διαγώνιο. Όταν βρείτε λύση στο πρόβλημα, οπτικοποιήστε την τυπώνοντας τις θέσεις των οκτώ βασιλισσών σε μια περιοχή κειμένου οκτώ γραμμών και στηλών. Η απλούστερη λύση είναι να παράγετε μηχανικά όλες τις πιθανές λύσεις (πόσες είναι;) και να ελέγχετε ποιες από αυτές αποτελούν λύση. Αυτό είναι ένα παράδειγμα εξαντλητικής αναζήτησης. Υπόδειξη για μια πιο αποδοτική λύση: τοποθετήστε μια βασίλισσα στην πρώτη γραμμή και σημειώστε σε έναν πίνακα τις θέσεις που απειλούνται. Τοποθετήστε τη δεύτερη βασίλισσα σε κάποια από τις θέσεις της δεύτερης γραμμής που δεν απειλούνται, κοκ.

Α52. [Το παιχνίδι της ζωής] Ο μαθηματικός John Conway χρησιμοποίησε το εξής μοντέλο για να μελετήσει τις αυξομειώσεις του πληθυσμού μιας αποικίας. Τα άτομα «ζουν» πάνω σε ένα πλέγμα Ν γραμμών και Μ στηλών και αναπαράγονται ακολουθώντας τους εξής κανόνες: (α) στις κενές κυψέλες του πλέγματος που έχουν τρεις γείτονες, γεννιέται ένα άτομο, (β) κάθε άτομο που έχει κανέναν ή ένα γείτονα, πεθαίνει από μοναξιά, (γ) κάθε άτομο που έχει δύο ή τρεις γείτονες επιζεί και στην επόμενη γενεά και (δ) κάθε άτομο με περισσότερους από 4 γείτονες πεθαίνει από υπερπληθυσμό. Θεωρήστε ότι όλες οι γεννήσεις και οι θάνατοι συμβαίνουν ταυτόχρονα. Φτιάξτε διεπαφή που θα προβάλλει την εξέλιξη της αποικίας, ξεκινώντας από μια αρχική κατανομή πληθυσμού στις κυψέλες του πλέγματος που θα ορίζει ο χρήστης. Χρησιμοποιήστε τον ψευδοκώδικα που φαίνεται στην Εικόνα 61 και κατασκευάστε μια ευκολόχρηστη διεπαφή χρήστη.

```
κατασκεύασε και αρχικοποίησε πλαίσιο για την τρέχουσα γενιά
κατασκεύασε πλαίσιο για την επόμενη γενιά
επαναλάμβανε μέχρι να σου πει ο χρήστης να σταματήσεις
  εμφάνισε τις κυψέλες του τρέχοντος πλαισίου
  επαναλάμβανε για όλες τις κυψέλες του τρέχοντος πλαισίου
    βρες το πλήθος των γειτόνων της κυψέλης
    επέλεξε ανάλογα με το πλήθος των γειτόνων
      αν είναι 0 ή 1 ή πάνω από 4,
        τότε η κυψέλη στο νέο πλαίσιο θα είναι κενή
      αν είναι 2,
        τότε η κυψέλη στο νέο πλαίσιο μένει όπως ήταν στο παλιό
      αν είναι 3,
        τότε η κυψέλη στο νέο πλαίσιο θα κατοικείται
    τέλος της επιλογής
  όριο της επανάληψης {για όλες τις κυψέλες}
  αντέγραψε το πλαίσιο της επόμενης γενιάς στην τρέχουσα γενιά
όριο της επανάληψης {μέχρι να σου πει ο χρήστης}
```

Εικόνα 61 Ψευδοκώδικας για το παιχνίδι της ζωής

4.10 Τραπεζικός Λογαριασμός

```
package simplebankaccount;

/**
 * <p> Title: Simple bank account </p>
 * <p> Description: Διαχειρίζεται το υπόλοιπο ενός τραπεζικού λογαριασμού.
 *     Μπορεις να καταθέσεις ή να αναλάβεις κάποιο ποσόν, φτάνει να επαρκεί το υπόλοιπο.
 * <p> Copyright: Freeware - No Copyright (c) 2003 </p>
 * <p> Company: Aegean University </p>
 * @author Ioanis Gaviotis
 * @version 1.0
 */
public class BankAccount {

  private double balance;

  public BankAccount() {
    balance = 0.0d;  //νεος λογαριασμος με υπολοιπο 0
  }

  /**
   * @param amount το (θετικό) ποσόν της κατάθεσης
   * @throws IllegalArgumentException αν το ποσόν είναι αρνητικό
   */
  public void deposit(double amount) throws IllegalArgumentException {
    if (amount >= 0.0d) balance += amount; //καταθεση
    else throw new IllegalArgumentException ("Amount must be positive.");
```

```
  }

  /**
   *
   * @param amount το (θετικό) ποσόν της ανάληψης
   * @throws IllegalArgumentException αν ποσόν είναι αρνητικό ή υπόλοιπο δεν επαρκεί
   */
  public void withdraw(double amount) throws IllegalArgumentException {
    if (amount >= 0.0d)
      if (amount <= balance) balance -= amount; //αναληψη
      else throw new IllegalArgumentException ("Balance not enough.");
    else throw new IllegalArgumentException ("Amount must be positive.");
  }

  /**
   * @return το υπόλοιπο του λογαριασμού
   */
  public double getBalance() {
    return balance;
  }

  /******************** για επίδειξη ***************************/
  public static void main(String[] args) {
    BankAccount myAccount = new BankAccount();
    boolean ok;
    try {
      myAccount.deposit(100.0);
      myAccount.withdraw(50.0);
      myAccount.withdraw(70.0);
      myAccount.deposit(-200.0);
    }
    catch (Throwable e) {
      System.out.println(e);
    }
    System.out.println(myAccount.getBalance());
  }
}
```