Αντικειμενοστρεφής σχεδίαση

Μετάφραση του κεφαλαίου 6 από το βιβλίο
The Essence of Program Design, των Douglas Bell, Ian Morrey, John Pugh, Prentice Hall, 1977

1 Εισαγωγή

Ξεκινάμε παρουσιάζοντας τις αρχές του αντικειμενοστρεφούς (Ο-Ο) προγραμματισμού. Εξετάζουμε τα ιδιαίτερα χαρακτηριστικά του και πώς αυτά υποστηρίζονται από τις αντικειμενοστρεφείς γλώσσες προγραμματισμού. Αργότερα θα δούμε πώς σχεδιάζουμε αντικειμενοστρεφή προγράμματα.

Αν νιώθετε ότι είστε εξοικειωμένοι με τις έννοιες του αντικειμενοστρεφούς προγραμματισμού, όπως κλάσεις και κληρονομικότητα, ίσως θα θέλατε να προσπεράσετε τις πρώτες ενότητες μέχρι να φτάσετε στο «Εισαγωγή στην Αντικειμενοστρεφή Σχεδίαση».

2 Αντικειμενοστρεφείς έννοιες

Τα τελευταία χρόνια αρκετές αντικειμενοστρεφείς γλώσσες έχουν διαδοθεί ευρέως. Μπορούν να χωριστούν σε δύο ομάδες – αμιγείς Ο-Ο γλώσσες (πχ Smalltalk, Eiffel και Java) και υβριδικές γλώσσες (πχ C++, Objective-C, CLOS και αντικειμενοστρεφείς διάλεκτοι της Pascal). Οι υβριδικές γλώσσες είναι επεκτάσεις παραδοσιακών γλωσσών και συνεπώς υποστηρίζουν το παραδοσιακό, αλλά και το Ο-Ο στιλ προγραμματισμού, ενώ οι αμιγείς γλώσσες έχουν σχεδιαστεί ειδικά για να υποστηρίζουν την Ο-Ο φιλοσοφία ανάπτυξης.

Οι Ο-Ο γλώσσες υποστηρίζουν τις ακόλουθες έννοιες:

- αντικείμενα και ενθυλάκωση
- πολυμορφισμό και δυναμική δέσμευση
- κλάσεις
- κληρονομικότητα και υποκλάσεις
- βιβλιοθήκες κλάσεων

Ας τις εξετάσουμε μία-προς-μία.

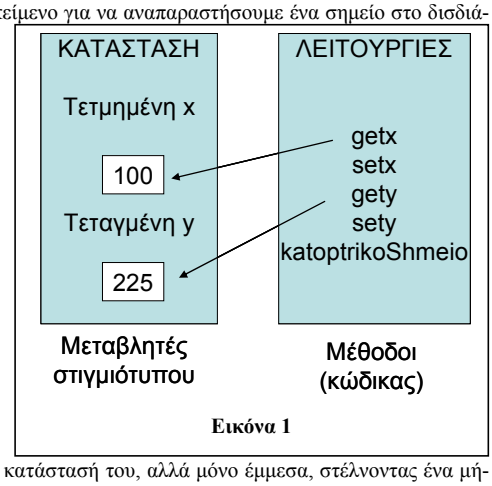
3 Αντικείμενα και ενθυλάκωση

Αντίθετα από τις παραδοσιακές γλώσσες, όπου διαδικασίες και δεδομένα θεωρούνται σαν ξεχωριστές οντότητες, οι Ο-Ο γλώσσες υιοθετούν την άποψη ότι πρέπει να χρησιμοποιηθεί μία αδιαχώριστη οντότητα, το *αντικείμενο*, για να ενθυλακώσει τα δεδομένα και τις διαδικασίες. Τα δεδομένα που περιέχονται σε ένα αντικείμενο μπορούν να τροποποιηθούν με τις διαδικασίες (που συνήθως αποκαλούνται *μέθοδοι*) που είναι μέρος αυτού του αντικειμένου.

Στον Ο-Ο προγραμματισμό, η κλήση μιας μεθόδου συχνά αναφέρεται ως *πέρασμα μηνύματος* σε ένα αντικείμενο. Ένα πρόγραμμα αποτελείται από μια συλλογή αντικειμένων τα οποία περνούν μηνύματα το ένα στο άλλο. Τα αντικείμενα ανταποκρίνονται στα μηνύματα, παράγοντας και νέα αντικείμενα ως αποτέλεσμα. Κάθε αντικείμενο μπορούμε να το σκεφτούμε σαν ένα μικρό εικονικό επεξεργαστή του οποίου η συμπεριφορά ορίζεται μόνο από το πώς ανταποκρίνεται στη λήψη μηνυμάτων. Ανακεφαλαιώνοντας, ένα Ο-Ο σύστημα μπορούμε να το φανταστούμε σαν μια συλλογή αντικειμένων που επικοινωνούν (συνεργάζονται) για να επιτύχουν κάποιο αποτέλεσμα.

Ας υποθέσουμε, σαν παράδειγμα, ότι θέλουμε να δημιουργήσουμε ένα αντικείμενο για να αναπαραστήσουμε ένα σημείο στο δισδιάστατο χώρο. Το αντικείμενο 'Σημείο' απεικονίζεται στην Εικόνα 1. Δύο χαρακτηριστικά ή ιδιότητες αποτελούν την κατάστασή του - η τετμημένη x (με τιμή 100) και η τεταγμένη y (με τιμή 225). Οι χρήστες επικοινωνούν με το αντικείμενο στέλνοντας μηνύματα σε αυτό. Ένα αντικείμενο 'Σημείο' μπορεί, για παράδειγμα, να ανταποκριθεί σε (ή να υποστηρίζει το πρωτόκολλο για) μηνύματα που:

- ανακαλούν την τετμημένη x
- ανακαλούν την τεταγμένη y
- επιστρέφουν ένα νέο σημείο (y, x) ίσο με το κατοπτρικό του σημείου (x, y) που έλαβε το μήνυμα



Εικόνα 1

Τα αντικείμενα υιοθετούν μια θεμελιώδη αρχή για την δόμηση των συστημάτων λογισμικού – την *απόκρυψη πληροφορίας*. Η ιδέα πίσω από την απόκρυψη πληροφορίας είναι ότι οι χρήστες ενός αντικειμένου δεν χρειάζεται (και δεν θα έπρεπε) να έχουν πρόσβαση στην αναπαράστασή του ή στην υλοποίηση των λειτουργιών του. Είναι χρήσιμο να σκεφτούμε τα αντικείμενα σαν να παρέχουν δύο όψεις του εαυτού τους: μια για τους δυνητικούς χρήστες ή πελάτες του αντικειμένου, και μια άλλη για αυτούς που υλοποιούν το αντικείμενο. Οι χρήστες του αντικειμένου μπορούν να αλλάξουν την κατάστασή του, αλλά μόνο έμμεσα, στέλνοντας ένα μήνυμα στο αντικείμενο. Το μεγαλύτερο πλεονέκτημα της προσέγγισης αυτής είναι ότι επιτρέπει σε αυτόν που υλοποιεί το αντικείμενο να αλλάξει την αναπαράστασή του κατά τρόπο που να είναι διαφανής στους χρήστες. Οι χρήστες του αντικειμένου δεν χρειάζεται να αντιληφθούν την αλλαγή. Αυτός ο διαχωρισμός της δημόσιας διεπαφής του αντικειμένου από την εσωτερική υλοποίησή του είναι απαραίτητος για την παραγωγή συντηρήσιμου και επαναχρησιμοποιήσιμου λογισμικού.

Στο παραπάνω παράδειγμα, δεν θα υπήρχε κανένας αντίκτυπος στους χρήστες αν, για παράδειγμα, τα σημεία υλοποιούνταν έτσι ώστε η κατάστασή τους να τηρείται σε πολικές (ακτίνα r, γωνία θ), και όχι σε καρτεσιανές (x, y) συντεταγμένες. Σίγουρα, η υλοποίηση της μεθόδου που επιστρέφει την τετμημένη x ενός σημείου, πρέπει να αλλάξει, αλλά αυτή η αλλαγή δεν θα επηρεάσει τους χρήστες. Ακόμα θα στέλνουν το ίδιο μήνυμα για να ανακτήσουν αυτήν την πληροφορία.

Ένα Ο-Ο πρόγραμμα δουλεύει με αντικείμενα τα οποία στέλνουν μηνύματα σε άλλα αντικείμενα. Όταν ένα αντικείμενο λαμβάνει ένα μήνυμα, προσδιορίζει αν έχει κατάλληλη μέθοδο που να του επιτρέπει να ανταποκριθεί. Ο ορισμός της μεθόδου περιγράφει πώς το αντικείμενο θα αντιδράσει με τη λήψη του μηνύματος. Σε μη Ο-Ο ορολογία, θα μιλούσαμε για το σύνολο των λειτουργιών και διαδικασιών που παρέχονται από το αντικείμενο. Έτσι για να ανακεφαλαιώσουμε την ορολογία:

Μέθοδος: Συνώνυμο της διαδικασίας ή λειτουργίας. Καλείται όταν ένα μήνυμα λαμβάνεται από ένα αντικείμενο.
Πρωτόκολλο: Το σύνολο των μηνυμάτων στα οποία ανταποκρίνεται ένα αντικείμενο.

Για να εκτελεστεί μια λειτουργία, μιλάμε για αποστολή ενός μηνύματος σε ένα αντικείμενο, αντί για την κλήση μιας διαδικασίας. Το αντικείμενο που λαμβάνει το μήνυμα αναφέρεται ως παραλήπτης (ή αποδέκτης). Έτσι, το μήνυμα `είσαιΚόκκινη?` μπορεί να σταλεί σε μια μπάλα σε ένα πρόγραμμα βιντεοπαιχνιδιού. Παρά τη νέα ορολογία, το αποτέλεσμα της αποστολής ενός μηνύματος σε ένα αντικείμενο ταυτίζεται με μια παραδοσιακή κλήση λειτουργίας, όπου το αντικείμενο που λαμβάνει το μήνυμα θα ήταν μια παράμετρος της λειτουργίας. Το αποτέλεσμα της αποστολής ενός μηνύματος στο αντικείμενο είναι να καλέσει την κατάλληλη μέθοδο, η οποία μετά επιστρέφει ένα αντικείμενο σαν αποτέλεσμα. Στην περίπτωση του μηνύματος `είσαιΚόκκινη?`, το αποτέλεσμα που επιστρέφεται θα είναι ένα αντικείμενο που μπορεί να τιμή αληθές ή ψευδές.

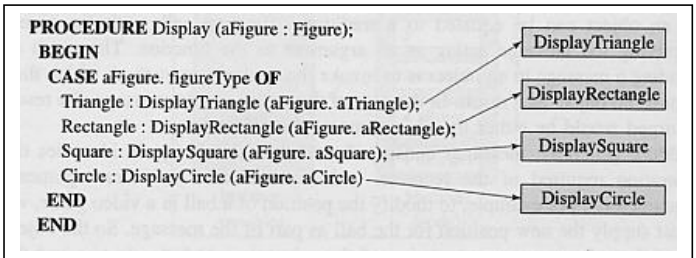
Γενικότερα, τα μηνύματα αποτελούνται από έναν επιλογέα μέσω του οποίου αναγνωρίζεται η εργασία που απαιτείται από τον παραλήπτη και ένα σύνολο από μηδέν ή παραπάνω παραμέτρους. Για παράδειγμα, για να αλλάξουμε τη θέση μιας μπάλας σε ένα βιντεοπαιχνίδι, πρέπει να παρέχουμε την νέα θέση της μπάλας σαν μέρος του μηνύματος. Έτσι το αντικείμενο `μπάλα1` λαμβάνει ένα μήνυμα που αποτελείται από τον επιλογέα `άλλαξεΘέση` και την παράμετρο `μιαΝέαΘέση`.

4 Πολυμορφισμός και δυναμική δέσμευση

Ένα από τα πιο σημαντικά χαρακτηριστικά του Ο-Ο προγραμματισμού είναι ότι η ερμηνεία ενός μηνύματος είναι στα χέρια του παραλήπτη, δηλαδή το ίδιο μήνυμα μπορεί να ερμηνευτεί από διαφορετικούς παραλήπτες με διαφορετικούς τρόπους. Μέθοδοι που έχουν αυτή την ιδιότητα λέγονται *πολυμορφικές*. Τα μηνύματα είναι λοιπόν κλήσεις διαδικασίας, με το πρόσθετο χαρακτηριστικό ότι η πραγματική διαδικασία που θα ξεκινήσει δεν είναι καθορισμένη μέχρι το μήνυμα να σταλεί σε ένα συγκεκριμένο παραλήπτη. Αυτή η διαδικασία της καθυστέρησης του καθορισμού της πραγματικής μεθόδου μέχρι το χρόνο εκτέλεσής της, ονομάζεται *δυναμική δέσμευση* (dynamic binding)

Ένα σημαντικό πλεονέκτημα του πολυμορφισμού και της δυναμικής δέσμευσης είναι ότι επιτρέπει την υπερφόρτωση ονομάτων, δηλαδή το ίδιο όνομα μπορεί να χρησιμοποιηθεί στο σύστημα για να δηλώσει μία κοινώς χρησιμοποιούμενη και καλά κατανοητή εργασία. Αυτή η συνέπεια στην ονομασία των λειτουργιών μειώνει το πλήθος των διαφορετικών ονομάτων, ειδικά σε μεγάλα συστήματα.

Ας δούμε ακόμα ένα παράδειγμα της χρησιμότητας πολυμορφισμού και δυναμικής δέσμευσης όπου θα συγκρίνουμε την δυναμική δέσμευση με την στατική που υπάρχει στις παραδοσιακές γλώσσες. Ας θεωρήσουμε μια εφαρμογή που διαχειρίζεται (πχ σχεδιάζει, κλπ) διάφορα είδη γεωμετρικών σχημάτων, όπως τετράγωνα, τρίγωνα, ορθογώνια και κύκλους. Για να συλλάβουμε την αφηρημένη έννοια του σχήματος με τύπους δεδομένων σε μια παραδοσιακή γλώσσα, όπως η Pascal, πρέπει να ορίσουμε μια εγγραφή με μεταβλητό πεδίο (variant) μέσω του οποίου θα διακρίνονται οι διαφορετικοί τύποι σχημάτων. Στη γλώσσα C, αυτό θα ήταν ένα πεδίο σε μία δομή (struct) που θα διαχώριζε τους διαφορετικούς τύπους σχημάτων. Μετά θα υλοποιούσαμε μια λειτουργία προβολής της εγγραφής. Για να αναλύσουμε την υλοποίηση στα συνθετικά της μέρη, θα μπορούσαμε να παρέχουμε διαφορετικές διαδικασίες για κάθε σχήμα. Σε αυτή την περίπτωση, η διαδικασία θα χρησιμοποιούσε κάποιο είδος εντολής πολλαπλής επιλογής (case) για να καθορίσει τον τύπο σχήματος που εμπλέκεται, ώστε να καλέσει τη σωστή διαδικασία για το συγκεκριμένο σχήμα (δείτε Εικόνα 2). Επειδή η συσχέτιση μεταξύ κάθε διαδικασίας σχετιζόμενης με ένα σχήμα και του τύπου της παραμέτρου πρέπει να γίνει κατά τη μεταγλώττιση, η σύνδεση των δυο τους ονομάζεται *στατική δέσμευση* (static binding).

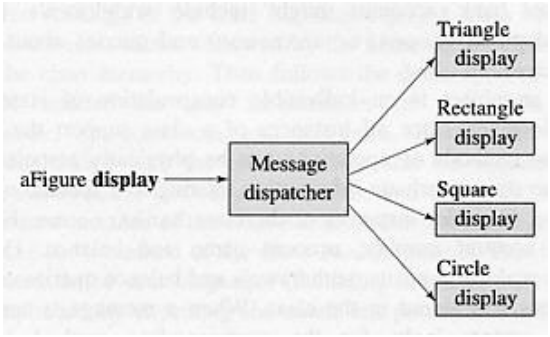


Εικόνα 2

Με τον ίδιο τρόπο θα μπορούσαμε να ορίσουμε λειτουργίες για τη μετακίνηση ή τον υπολογισμό του εμβαδού κάθε σχήματος. Αυτές οι διαδικασίες θα είχαν την ίδια λογική πολλαπλής επιλογής `case` με την παραπάνω διαδικασία προβολής.

Σε μία Ο-Ο γλώσσα με δυναμική δέσμευση, πάλι πρέπει να υλοποιήσουμε μία μέθοδο προβολής για καθένα από τα τρίγωνα, τετράγωνα, ορθογώνια και κύκλους. Ωστόσο, μπορούμε να χρησιμοποιήσουμε το ίδιο όνομα σε όλες τις περιπτώσεις. Επιπλέον, δεν είναι πια ευθύνη του προγραμματιστή να καθορίσει την σωστή μέθοδο που πρέπει να κληθεί. Ο προγραμματιστής μπορεί να στείλει το μήνυμα προβολής σε οποιοδήποτε σχήμα. Ανάλογα με τον τύπο του σχήματος, θα εκτελεστεί η σωστή μέθοδος προβολής από το μήνυμα του αποστολέα του συστήματος (δείτε Εικόνα 3).

Η Ο-Ο λύση είναι καταλληλότερη όταν έχουμε συχνές αλλαγές και επιθυμούμε να επαναχρησιμοποιούμε τον κώδικα. Ας υποθέσουμε ότι θέλουμε να επεκτείνουμε το παράδειγμα με τα σχήματα ώστε να συμπεριλάβουμε άλλον ένα τύπο σχήματος, το πεντάγωνο. Θα θέλαμε να εφοδιάσουμε τα πεντάγωνα με όλες τις λειτουργίες που υποστηρίζονται από τα άλλα σχήματα. Στην παραδοσιακή λύση, πρέπει να τροποποιήσουμε όλες τις λειτουργίες συμπεριλαμβάνοντας μια περίπτωση στην πολλαπλή επιλογή για το πεντάγωνο. Σε κάθε περίπτωση πρέπει να προστίθεται μια νέα επιλογή. Σε ένα μεγάλο σύστημα, με αυτό το σενάριο είναι εύκολο να προκύψουν λάθη. Υπάρχουν πιθανότητες να παραλείψουμε μία ή περισσότερες από τις απαραίτητες αλλαγές. Σε ένα Ο-Ο σύστημα, οι απαιτούμενες αλλαγές είναι εντοπισμένες στον κώδικα που υλοποιεί το νέο σχήμα, το πεντάγωνο – και απλώς υλοποιούμε αυτόν τον κώδικα, χωρίς να αλλάζουμε τίποτε άλλο.



Εικόνα 3

Περιληπτικά, οι έννοιες του πολυμορφισμού και της δυναμικής δέσμευσης επιτρέπουν τη χρήση γενικού, επαναχρησιμοποιήσιμου κώδικα.

5 Κλάσεις

Μια σημαντική ικανότητα που έχουμε είναι να ομαδοποιούμε και να κατηγοριοποιούμε. Οι ελέφαντες, οι τίγρεις, οι πολικές αρκούδες, τα άλογα και οι αγελάδες είναι θηλαστικά. Ο μόλυβδος, το ασήμι και ο λευκόχρυσος είναι όλα μέταλλα. Οι τρεχούμενοι λογαριασμοί, οι κλειστές καταθέσεις με προθεσμία, είναι τύποι τραπεζικών λογαριασμών, κοκ. Μέσω της ομαδοποίησης είμαστε σε θέση να συσχετίζουμε κοινά χαρακτηριστικά σε όλα τα μέλη μιας κλάσης. Όλα τα θηλαστικά είναι σπονδυλωτά (έχουν ραχοκοκαλιά), είναι θερμόαιμα και έχουν τρίχωμα στο σώμα τους. Όλα τα μέταλλα έχουν ατομικά βάρη και όλοι οι τραπεζικοί λογαριασμοί έχουν υπόλοιπο.

Στον Ο-Ο προγραμματισμό, η *κλάση* είναι η αφαίρεση που συλλαμβάνει τις ιδιότητες και τις λειτουργίες που είναι κοινές για ένα σύνολο αντικειμένων. Μια κλάση περιγράφει την αναπαράσταση και το πρωτόκολλο μηνυμάτων που ακολουθεί κάθε μέλος της, ή στην ορολογία του Ο-Ο προγραμματισμού, το *στιγμιότυπο* της κλάσης. Στις περισσότερες Ο-Ο γλώσσες, κάθε αντικείμενο είναι στιγμιότυπο κάποιας κλάσης.

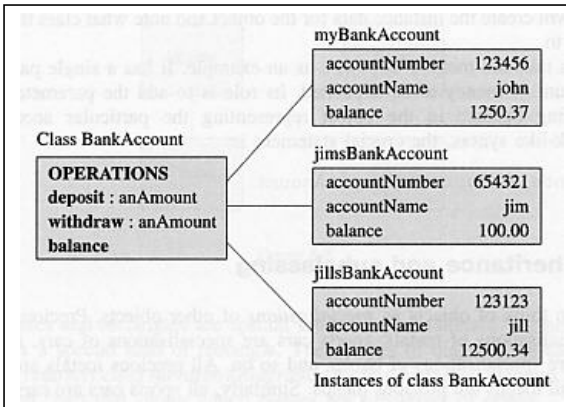
Συνοψίζοντας, η κλάση είναι μια περιγραφή ενός συνόλου αντικειμένων με παρόμοια χαρακτηριστικά, ιδιότητες και συμπεριφορά.

Ένα στιγμιότυπο είναι ένα μεμονωμένο αντικείμενο που περιγράφεται από την κλάση του και είναι μέλος της. Ας εξετάσουμε μια απλή τραπεζική εφαρμογή στην οποία επιθυμούμε να υποστηρίξουμε ένα σύνολο τραπεζικών λογαριασμών και να εφαρμόσουμε κοινές λειτουργίες, όπως ερώτηση υπολοίπου, κατάθεση και ανάληψη. Τα δεδομένα που συσχετίζονται με κάθε ξεχωριστό τραπεζικό λογαριασμό θα αποτελούνται από τουλάχιστον έναν αριθμό λογαριασμού, ένα όνομα λογαριασμού και ένα υπόλοιπο. Η απεικόνιση του αντικειμένου μπορεί να θεωρηθεί ως μια εγγραφή (record) της Pascal, ή ως μια δομή (struct) της γλώσσας C - μια συλλογή δηλαδή, από ετερογενή στοιχεία ή πεδία. Τα πεδία ενός αντικειμένου αναφέρονται ως *μεταβλητές στιγμιότυπου*, δεδομένου ότι θα απαντώνται σε κάθε στιγμιότυπο και θα μπορούν να τροποποιηθούν. Όλοι οι τραπεζικοί λογαριασμοί επομένως έχουν τρεις μεταβλητές στιγμιότυπου: αριθμό λογαριασμού, όνομα λογαριασμού και υπόλοιπο. Κατά συνέπεια:

Μεταβλητή στιγμιότυπου: Δεδομένα που είναι συστατικό μέρος ή πεδίο ενός αντικειμένου.

Οι λειτουργίες στους τραπεζικούς λογαριασμούς θα μπορούσαν να περιλαμβάνουν αναλήψεις, καταθέσεις, και ερωτήσεις για το υπόλοιπό τους.

Συνεπώς ένα αντικείμενο ενθυλακώνει κατάσταση (δεδομένα) και λειτουργίες. Ωστόσο, δεδομένου ότι όλα τα στιγμιότυπα μιας κλάσης υποστηρίζουν το ίδιο σύνολο λειτουργιών, οι μέθοδοι ή λειτουργίες μπορούν να συνδεθούν με την κλάση. Μόνο η κατάσταση ή ιδιωτικές πληροφορίες που σχετίζονται με ένα συγκεκριμένο αντικείμενο ανήκουν στην μεταβλητή στιγμιότυπου. Ας εξετάσουμε στιγμιότυπα της κλάσης `BankAccount`. Κάθε στιγμιότυπο έχει τον δικό του αριθμό, όνομα, και υπόλοιπο. Εντούτοις, οι λειτουργίες για την πραγματοποίηση καταθέσεων, αναλήψεων και ερωτήσεων υπολοίπου μπορούν να είναι κοινές για όλα τα στιγμιότυπα και να αποθηκεύονται στην κλάση. Όταν ένα μήνυμα στέλνεται σε έναν τραπεζικό λογαριασμό, το σύστημα ψάχνει για την αντίστοιχη μέθοδο στην κλάση `BankAccount`. Η Εικόνα 4 επεξηγεί τις κοινές λειτουργίες που συνδέονται με την κλάση `BankAccount` και τρία στιγμιότυπα καθένα από τα οποία διατηρεί την δική του κατάσταση. Αυτή η φυσική, σε αντίθεση με τη λογική, άποψη των κλάσεων και των στιγμιότυπων μας οδηγεί στους ακόλουθους εναλλακτικούς ορισμούς:



Εικόνα 4

Κλάση: Αποθήκη για μεθόδους (λειτουργίες) που τις μοιράζονται όλα τα στιγμιότυπα (αντικείμενα) που ανήκουν σε εκείνη την κλάση - ένας μηχανισμός κοινοκτημοσύνης κώδικα.
Στιγμιότυπο: Αποθήκη για δεδομένα που περιγράφουν την κατάσταση ενός συγκεκριμένου μέλους μιας κλάσης.

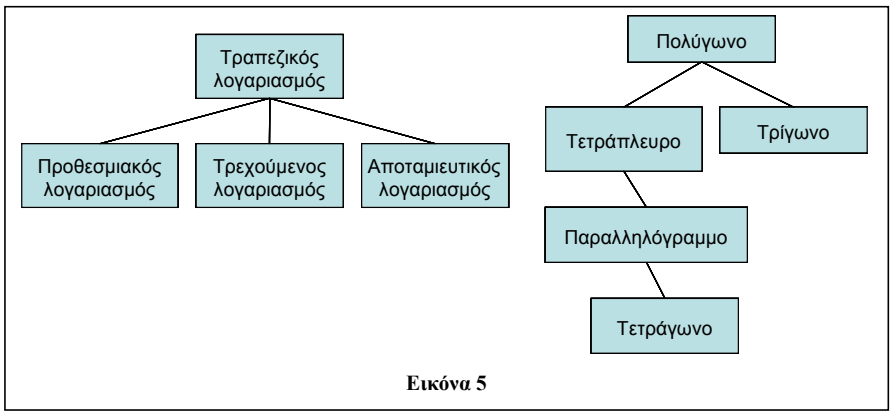
Ας περιγράψουμε την υλοποίηση της κλάσης `BankAccount`. Μια κλάση αρχίζει με τη δήλωση του ονόματος κλάσης. Κατόπιν ακολουθεί η δήλωση των μεταβλητών που απαιτούνται για οποιοδήποτε στιγμιότυπο της κλάσης. Αυτές είναι: `accountNumber`, `accountName`, και `balance`. Στο τέλος προσδιορίζονται οι μέθοδοι, κάθε μία από τις οποίες διαθέτει ένα όνομα μεθόδου και τα απαραίτητα ορίσματα. Αυτές είναι οι `openAccount`, `withdraw` , `deposit` και `queryBalance`.

Για να δημιουργηθεί ένα νέο στιγμιότυπο της κλάσης `BankAccount` πρέπει να σταλεί ένα μήνυμα στη μέθοδο `OpenAccount`. Έτσι δημιουργείται ένα νέο αντικείμενο. Σε αρκετές γλώσσες (συμπεριλαμβανομένων των Smalltalk, C++ και Java) αυτό επιτυγχάνεται με τη λέξη `new`. Στο παράδειγμα, το μήνυμα που στέλνεται στη μέθοδο `OpenAccount` θα περιλαμβάνει το όνομα του λογαριασμού, τον αριθμό λογαριασμού και το αρχικό υπόλοιπο ως παραμέτρους. Το σύστημα θα δημιουργήσει τα δεδομένα στιγμιότυπου για το αντικείμενο και θα σημειώσει σε ποια κλάση ανήκει.

Ας δούμε τώρα τη μέθοδο `deposit` για παράδειγμα. Έχει μια παράμετρο, το χρηματικό ποσό που πρόκειται να κατατεθεί. Ο ρόλος της μεθόδου είναι να προσθέσει την παράμετρο στο υπάρχον υπόλοιπο του αντικειμένου που αντιπροσωπεύει το συγκεκριμένο λογαριασμό.

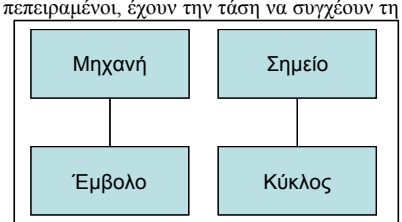
6 Κληρονομικότητα και υποκλάσεις

Συχνά σκεφτόμαστε τα αντικείμενα ως *εξειδικεύσεις* άλλων αντικειμένων. Τα πολύτιμα μέταλλα είναι υποκατηγορία των μετάλλων, τα αγωνιστικά αυτοκίνητα είναι υποκατηγορία των αυτοκινήτων, τα ρομαντικά μυθιστορήματα είναι υποκατηγορία των βιβλίων, κ.ο.κ. Όλα τα πολύτιμα μέταλλα είναι μέταλλα αλλά όλα τα μέταλλα δεν είναι πολύτιμα. Ομοίως, όλα τα αγωνιστικά αυτοκίνητα είναι αυτοκίνητα και όλα τα ρομαντικά μυθιστορήματα είναι βιβλία, αλλά το αντίστροφο δεν ισχύει. Επεκτείνοντας αυτή την έννοια, μπορούμε να θεωρήσουμε μια κλάση αντικειμένων ως *υποκλάση* κάποιας άλλης. Μπορούμε επίσης να μιλήσουμε για μια κλάση που είναι *υπέρ-κλάση* κάποιας άλλης. Συνοψίζοντας, μπορούμε να δημιουργήσουμε ιεραρχίες κλάσεων που να βασίζονται σε σχέσεις 'είναι-ένα'. Στην Εικόνα 5 οι προθεσμιακοί, τρεχούμενοι και αποταμιευτικοί λογαριασμοί είναι όλοι τραπεζικοί λογαριασμοί. Ομοίως, στην Εικόνα 5, τα τετράπλευρα και τα τρίγωνα είναι πολύγωνα και τα τετράγωνα και τα παραλληλόγραμμα είναι ειδικές περιπτώσεις τετραπλεύρων. Επιπλέον, ένα τετράγωνο είναι ειδική περίπτωση ενός ορθογωνίου. Αυτοί οι τύποι διαγραμμάτων ονομάζονται (για προφανείς λόγους) *διαγράμματα ιεραρχίας κλάσεων*. Το διάγραμμα ιεραρχίας κλάσεων είναι ένα δενδρικό διάγραμμα που εμφανίζει ποιες κλάσεις είναι υποκλάσεις άλλων.



Εικόνα 5

Είναι απαραίτητο να επιδειχθεί λίγη προσοχή εδώ. Αρχάριοι, μερικές φορές ακόμα και πεπειραμένοι, έχουν την τάση να συγχέουν τη σχέση 'είναι-ένα' με τη σχέση 'αποτελείται-από'. Δύο παραδείγματα ιεραρχίας 'αποτελείται-από' εμφανίζονται στην Εικόνα 6. Το έμβολο είναι ένα στοιχείο της μηχανής, δεν είναι μια μηχανή. Ομοίως, ένας κύκλος δεν είναι ένα σημείο. Το έναυσμα για να περιγραφεί ο κύκλος ως υποκλάση του σημείου μπορεί να είναι ότι ένα σημείο απαιτείται για το κέντρο του κύκλου. Ωστόσο, με αυτόν τον συλλογισμό, ο κύκλος θα έπρεπε επίσης να κληρονομήσει και από την κλάση αριθμός, επειδή ένας αριθμός απαιτείται για την ακτίνα του κύκλου.



Εικόνα 6

Εξειδίκευση και γενίκευση

Τι σημαίνει ότι μια κλάση είναι μια υποκλάση κάποιας άλλης; Διαισθητικά, εννοούμε ότι η υποκλάση έχει όλα τα χαρακτηριστικά της γενικότερης κλάσης, αλλά την επεκτείνει κατά κάποιο τρόπο. Τα πολύτιμα μέταλλα έχουν όλα τα χαρακτηριστικά των μετάλλων αλλά, επιπλέον, μπορούν να διακριθούν από τα υπόλοιπα μέταλλα με βάση τη νομισματική τους αξία. Ομοίως, τα τετράπλευρα είναι ειδικές περιπτώσεις πολυγώνων με τέσσερις πλευρές. Τα πολύγωνα μπορούν να έχουν οποιοδήποτε αριθμό πλευρών. Τα τετράγωνα είναι ειδικές περιπτώσεις των τετραπλεύρων όπου και οι τέσσερις πλευρές έχουν ίσο μήκος, και οι προσκείμενες πλευρές είναι κάθετες η μια στην άλλη. Εφαρμόζοντας ανάποδα αυτά τα επιχειρήματα, μπορούμε να περιγράψουμε την υπέρ-κλάση μιας κλάσης ως γενίκευση της κλάσης.

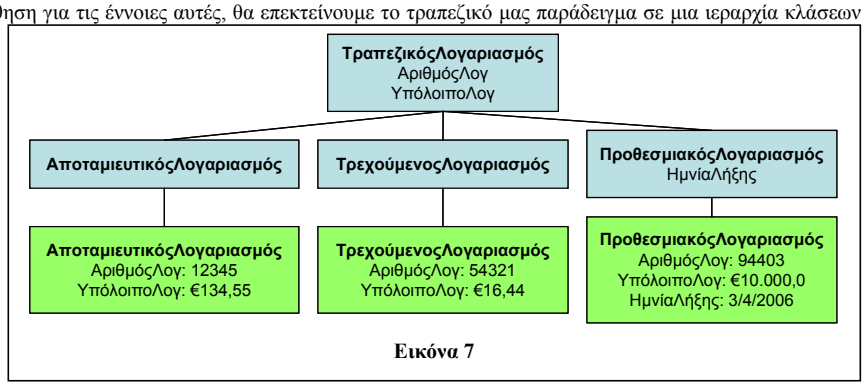
Ένας από τους καλύτερους τρόπους να περιγραφεί κάτι καινούργιο σε κάποιον άλλο είναι να περιγραφεί με όρους κάποιου με το οποίο είναι παρόμοιο, δηλαδή περιγράφοντας πώς αυτό διαφέρει από κάτι γνωστό. Ένα παράδειγμα είναι ότι η ζέβρα είναι ένα άλογο με ρίγες! Αυτός ο συνοπτικός ορισμός περιέχει ένα μεγάλο ποσό πληροφοριών σε κάποιον σχετικό με τα άλογα.

Οι αντικειμενοστρεφείς γλώσσες υποστηρίζουν τις έννοιες της εξειδίκευσης και της περιγραφής διαφορών. Οι κλάσεις οργανώνονται ιεραρχικά σε σχέσεις εξειδίκευσης. Όταν μια κλάση είναι μια υποκλάση κάποιας άλλης, θεωρείται δεδομένο ότι θα *κληρονομήσει* την αναπαράσταση και τη συμπεριφορά της υπέρ-κλάσης της. Χάρη στο μοίρασμα που επιτυγχάνεται λόγω κληρονομικότητας, η νέα κλάση πρέπει να περιγράψει, μόνο, κατά τι διαφέρει από την υπέρ-κλάση. Συνοψίζοντας:

Υποκλάση: Κλάση που κληρονομεί τις μεθόδους και την αναπαράσταση από μια υπάρχουσα κλάση.

Υπέρ-κλάση: Κλάση από την οποία μια άλλη κλάση κληρονομεί αναπαράσταση και μεθόδους.

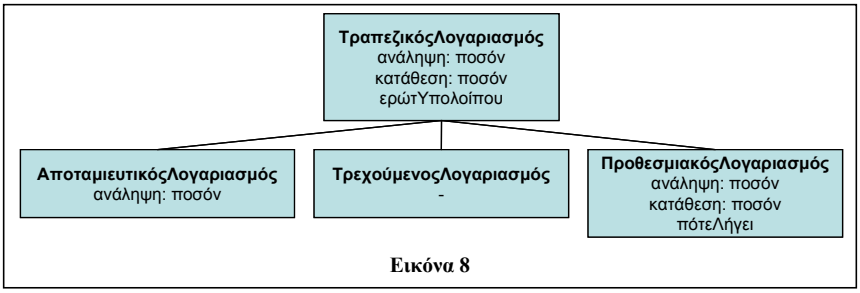
Για να έχουμε μια καλύτερη αίσθηση για τις έννοιες αυτές, θα επεκτείνουμε το τραπεζικό μας παράδειγμα σε μια ιεραρχία κλάσεων τραπεζικών λογαριασμών, όπως φαίνεται στην Εικόνα 7. Για λόγους απλότητας, έχουμε περιορίσει το πρόβλημα μόνο στα στοιχειώδη. Ας υποθέσουμε ότι όλοι οι τραπεζικοί λογαριασμοί έχουν έναν αριθμό λογαριασμού και ένα υπόλοιπο και, επιπλέον, οι προθεσμιακοί λογαριασμοί έχουν μια ημερομηνία λήξης. Η κλάση 'ΤραπεζικόςΛογαριασμός', επομένως, έχει δύο μεταβλητές στιγμιότυπου, τις 'ΑριθμόςΛογ' και 'ΥπόλοιποΛογ', και όλες οι τρεις υποκλάσεις κληρονομούν αυτή την αναπαράσταση, έτσι ώστε όλα τα στιγμιότυπα έχουν τουλάχιστον αυτά τα δύο πεδία. Οι υποκλάσεις 'ΤρεχούμενοςΛογαριασμός' και 'ΑποταμιευτικόςΛογαριασμός' δεν προσθέτουν καμία μεταβλητή επιπλέον. Η κλάση 'ΠροθεσμιακόςΛογαριασμός', εντούτοις, παρουσιάζει μια επιπλέον μεταβλητή στιγμιότυπου την 'ΗμνίαΛήξης', δίνοντας στους κλειστούς λογαριασμούς χρονικό όριο. Γενικά, οι υποκλάσεις μπορούν να προσθέσουν καινούργιες μεταβλητές, αλλά δε μπορούν ποτέ να αφαιρέσουν. Ωστόσο, οι υποκλάσεις μπορούν να προσθέσουν μεθόδους με ίδια ονόματα για να υπερισχύσουν των μεθόδων που κληρονομούνται από την υπέρ-κλάση.



Εικόνα 7

Όλοι οι τύποι τραπεζικών λογαριασμών υποστηρίζουν λειτουργίες για ερώτηση υπολοίπου ενός λογαριασμού. Εάν η λειτουργία έχει τις πανομοιότυπες υλοποιήσεις σε κάθε υποκλάση, μπορούμε να υλοποιήσουμε τη λειτουργία μία και μόνο φορά στην κοινή υπέρ-κλάση 'ΤραπεζικόςΛογαριασμός' και οι τρεις υποκλάσεις θα την κληρονομήσουν. Άλλες λειτουργίες, όπως η ερώτηση για την προθεσμία ενός λογαριασμού, θα πρέπει να είναι συγκεκριμένες στην κλάση 'ΠροθεσμιακόςΛογαριασμός'.

Σε μερικές περιπτώσεις, μια κοινή λειτουργία που ίσως να επιθυμούσαμε να εφαρμόσουμε μία και μόνο φορά σε μία υπέρ-κλάση θα πρέπει να επαναληφθεί στις υποκλάσεις, εάν η υλοποίησή της εξαρτάται από τον συγκεκριμένο τύπο λογαριασμού. Παραδείγματος χάριν, η λειτουργία της κατάθεσης (τοποθετώ χρήματα σε λογαριασμό) μπορεί να είναι κοινή για τους τρεχούμενους και τους αποταμιευτικούς λογαριασμούς, αλλά να απαιτείται διαφορετική υλοποίηση για τους προθεσμιακούς λογαριασμούς. Η Εικόνα 8 απεικονίζει μια σχεδίαση για την τοποθέτηση των λειτουργιών μέσα στην ιεραρχία των κλάσεων. Οι λειτουργίες πρέπει να τοποθετούνται όσο το δυνατόν υψηλότερα στην ιεραρχία, έτσι ώστε μπορούν να μοιράζονται σε όσο το δυνατόν περισσότερες υποκλάσεις.



Εικόνα 8

Όταν ένα μήνυμα στέλνεται σε ένα αντικείμενο, το σύστημα κοιτάζει αρχικά για μια μέθοδο με το ίδιο όνομα στην κλάση του αντικειμένου. Εάν βρεθεί η μέθοδος, εκτελείται. Ειδάλλως, συνεχίζεται η αναζήτηση στην υπέρ-κλάση κ.ο.κ. Τελικά, θα βρεθεί μια μέθοδος και θα εκτελεσθεί ή θα φτάσουμε στην κορυφή της ιεραρχίας. Τότε ανακοινώνεται λάθος, δεδομένου ότι δεν υπάρχει καμία μέθοδος που να μπορεί να ανταποκριθεί στο μήνυμα. Για να διευκρινισθεί αυτή η διαδικασία αναζήτησης, μελετήστε το παράδειγμα στην Εικόνα 8 για να προσδιορίσετε ποια μέθοδος εκτελείται πραγματικά.

Ανακεφαλαιώνοντας, μια νέα κλάση μπορεί να διαφοροποιείται από τις υπέρ-κλάσεις της με διάφορους τρόπους. Ειδικότερα, η νέα κλάση μπορεί να κάνει ένα ή περισσότερα από τα εξής:

- Να υποστηρίξει πρόσθετες λειτουργίες, πέρα από αυτές που κληρονομούνται.
- Να υποστηρίξει νέες υλοποιήσεις λειτουργιών που έτσι και αλλιώς κληρονομεί.
- Να υπερκαλύπτει υπάρχουσες λειτουργίες που υποστηρίζονται από την υπέρ-κλάση, αλλά είναι ακατάλληλες για τη νέα κλάση προσθέτοντας μια λειτουργία που σηματοδοτεί σφάλμα.

- Να περιέχει ένα περιορισμένο υποσύνολο από τις μεταβλητές στιγμιότυπου της κλάσης
- Να προσθέτει επιπλέον δεδομένα.

Παρατηρήστε ότι σκοπός της εξειδίκευσης και της κληρονομικότητας που έχουμε περιγράψει είναι, σε αντίθεση με την δική μας διαισθητική άποψη, ότι μια υποκλάση είναι επέκταση της υπέρ-κλάσης της

Θεωρήσαμε ως δεδομένο ότι μια κλάση μπορεί να κληρονομήσει μόνο από μια υπέρ-κλάση, οπότε το διάγραμμα ιεραρχίας κλάσεων είναι πάντα ένα δέντρο. Αυτή λέγεται *απλή κληρονομικότητα.* Κάποιες γλώσσες, όπως η C++, επιτρέπουν την *πολλαπλή κληρονομικότητα,* στην οποία μια κλάση μπορεί να κληρονομεί από δυο ή περισσότερες υπέρ-κλάσεις. Η πολλαπλή κληρονομικότητα είναι προφανώς πιο μπερδεμένη, αλλά πιο ευέλικτη. Υπάρχει διαμάχη ανάμεσα στην αντικειμενοστρεφή κοινότητα για το αν η πολλαπλή κληρονομικότητα είναι επιθυμητή, και μια πρόσφατη γλώσσα, η Java, δεν την υποστηρίζει. Εδώ θα περιορίσουμε την προσοχή μας στην απλή κληρονομικότητα.

7 Βιβλιοθήκες κλάσεων

Οι αντικειμενοστρεφείς βιβλιοθήκες κλάσεων είναι πιο ευέλικτες από τις παραδοσιακές βιβλιοθήκες αριθμητικών και στατιστικών υπορουτινών της FORTRAN ή της C. Οι παραδοσιακές βιβλιοθήκες είναι στατικές με την έννοια ότι δεν είναι δυνατό να επαναγράφεται ή να επεκτείνεται μια ρουτίνα που δεν λειτουργεί ακριβώς με τον τρόπο που επιθυμούμε. Αντίθετα, μια Ο-Ο βιβλιοθήκη επιτρέπει προσαρμογή και επέκταση με τη βοήθεια υποκλάσεων.

Για παράδειγμα, η βιβλιοθήκη κλάσεων της Smalltalk περιλαμβάνει περίπου 150 κλάσεις. Αυτό είναι και πλεονέκτημα, αλλά και μειονέκτημα. Πλεονέκτημα είναι γιατί καινούριες εφαρμογές μπορούν να εκμεταλλευτούν αυτή τη βιβλιοθήκη και να αποφύγουμε να «ανακαλύψουμε τον τροχό». Αρχικά όμως, είναι και μειονέκτημα γιατί ο προγραμματιστής πρέπει να μάθει το περιεχόμενο της βιβλιοθήκης και να μπορεί να διακρίνει μεταξύ των σχετικών συνόλων κλάσεων που φαινομενικά παρέχουν σχεδόν την ίδια λειτουργία. Για παράδειγμα, η βιβλιοθήκη κλάσεων της Smalltalk παρέχει πολλά είδη κλάσεων που υλοποιούν συλλογές.

Η βιβλιοθήκη κλάσεων της Smalltalk έχει λειτουργήσει ως μοντέλο για πολλές βιβλιοθήκες Ο-Ο γλωσσών. Οι σημαντικότερες κλάσεις μπορούν να ταξινομηθούν ως εξής:

Βασικές κλάσεις: `Object`, `Boolean`, `Character`, `Integer`, `Float`, `Fraction`, `Date`, `Time`, κλπ

Κλάσεις συλλογών: `Dictionary`, `Array`, `String`, `Symbol`, `Set`, `OrderedCollection`, `SortedCollection`, κλπ

Κλάσεις Γραφικών: `Point`, `Rectangle`, `Bitmap`, `Pen`, κλπ. Είναι κλάσεις που παρέχουν πρόσβαση σε υπορουτίνες γραφικών.

Κλάσεις γραφικών διεπαφών: Κλάσεις που υποστηρίζουν παράθυρα, παράθυρα διαλόγου, μενού, πλήκτρα, σύνταξη κειμένου, κλπ

Εργαλεία: Κλάσεις που πλοηγούν μέσα στην βιβλιοθήκη κλάσεων, εκτελούν επιθεώρηση αντικειμένων, εκσφαλμάτωση πηγαίου κώδικα, διαχείριση αλλαγών, κλπ

Περιληπτικά, οι αντικειμενοστρεφείς γλώσσες ενθαρρύνουν την παραγωγή των επαναχρησιμοποιήσιμων βιβλιοθηκών κλάσεων. Υπάρχουσες βιβλιοθήκες υποστηρίζουν βασικούς τύπους δεδομένων, γραφικά, διεπαφές χρήστη και εργαλεία υποστήριξης προγραμματισμού. Αλλά όλο και περισσότερο εμφανίζονται βιβλιοθήκες για συγκεκριμένους τομείς εφαρμογών.

8 Περίληψη των εννοιών του αντικειμενοστρεφούς προγραμματισμού

Ο Ο-Ο προγραμματισμός χαρακτηρίζεται από:

Προγραμματισμός με αντικείμενα: Τα αντικείμενα έχουν κατάσταση και μπορούν να απαντούν σε ερωτήσεις για τον εαυτό τους. Τα αντικείμενα είναι τύποι δεδομένων. Ενθυλακώνουν κατάσταση ή αναπαράσταση ενός αντικειμένου μαζί με λειτουργίες πάνω σε αυτή την κατάσταση και υποστηρίζουν την αρχή της *απόκρυψης πληροφορίας*.

Προγραμματισμός με προσομοίωση: Οι εφαρμογές σχεδιάζονται και υλοποιούνται ως προσομοίωση του προβλήματος που επιλύουν. Τα αντικείμενα μοντελοποιούν οντότητες στον πραγματικό κόσμο. Αυτό το είδος προγραμματισμού αναφέρεται συχνά ως *ανθρωπομορφικός προγραμματισμός*.

Πέρασμα μηνυμάτων: Οι μέθοδοι καθορίζουν το πώς ένα αντικείμενο θα ανταποκριθεί σε ένα δοσμένο μήνυμα.

Προγραμματισμός χρησιμοποιώντας πολυμορφισμό: Τα μηνύματα ερμηνεύονται με διαφορετικούς τρόπους από διαφορετικούς παραλήπτες.

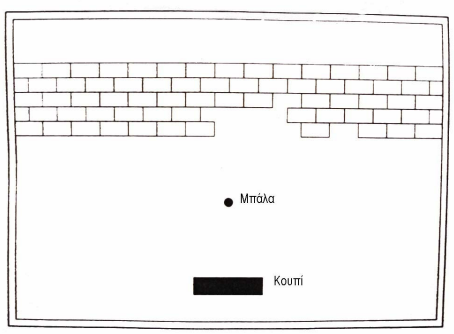
Προγραμματισμός χρησιμοποιώντας κληρονομικότητα: Η διαμοίραση του κώδικα επιτυγχάνεται μέσω της κληρονομικότητας αναπαράστασης και μεθόδων από τη μια κλάση ενός αντικειμένου σε άλλη. Οι νέες κλάσεις καθορίζονται ως εξειδικεύσεις των κλάσεων που υπάρχουν.

9 Εισαγωγή στην αντικειμενοστρεφή σχεδίαση

Έχουμε δει ότι ο Ο-Ο προγραμματισμός παρέχει στο σχεδιαστή ένα σύνολο ισχυρών εννοιών και εργαλείων. Ο σχεδιαστής ενός Ο-Ο προγράμματος είναι να βρει ποιες κλάσεις, αντικείμενα και μέθοδοι απαιτούνται για το πρόγραμμα. Ο σχεδιαστής πρέπει επίσης να μελετήσει πώς καθεμιά από αυτές τις κλάσεις και αντικείμενα χρησιμοποιεί τα υπόλοιπα. Το τελικό προϊόν της σχεδίασης μπορεί να τεκμηριωθεί χρησιμοποιώντας δύο συμβολισμούς:

1. Διαγράμματα ιεραρχίας κλάσεων, που εμφανίζουν τις κλάσεις στο πρόγραμμα και πώς συσχετίζονται ως υποκλάσεις και υπέρ-κλάσεις.
2. Λίστες CRC (Class-Responsibility-Collaborators), οι οποίες εμφανίζουν για κάθε κλάση τι παρέχει και ποιες άλλες κλάσεις χρησιμοποιεί.

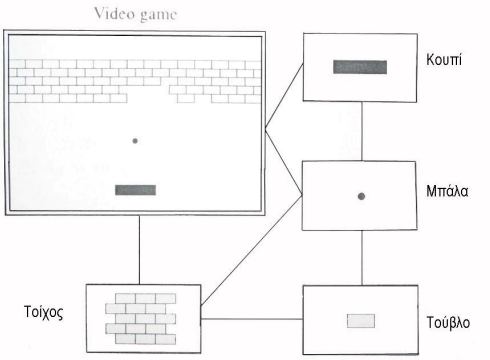
Η Ο-Ο σχεδίαση και προγραμματισμός μπορεί να περιγραφτεί ως προγραμματισμός με προσομοίωση. Η αλληγορία βασίζεται στην ‘μεταφορά’ των φυσικών ή ιδεατών αντικειμένων του πραγματικού κόσμου σε αντικείμενα του προγράμματος. Για παράδειγμα, αντικείμενα είναι οι πελάτες σε μια επιχείρηση, τα τρόφιμα σε μια υπεραγορά, ή τα διάφορα ανταλλακτικά σε ένα εργοστάσιο. Προσπαθούμε να ‘ενσαρκώσουμε’ τα αντικείμενα από το πρόβλημα σε υπολογιστικά μοντέλα, δίνοντας στα αντικείμενα στο πρόγραμμά μας τα ίδια χαρακτηριστικά και δυνατότητες που έχουν στον πραγματικό κόσμο. Κατά συνέπεια χτίζουμε ένα μοντέλο ή μια προσομοίωση του προβλήματος που θέλουμε να λύσουμε. Όταν τα αντικείμενα της υλοποίησης έχουν άμεση αντιστοίχιση με τα αντικείμενα του προβλήματος, είναι πιο εύκολο να κατανοήσουμε και να χρησιμοποιήσουμε το λογισμικό.



Εικόνα 9

Για να πάρετε μια αίσθηση για το σχεδιασμό με αντικείμενα, θα δούμε ένα παράδειγμα με το παιχνίδι στην Εικόνα 9.

Εάν προσεγγίσουμε το πρόβλημα χρησιμοποιώντας Ο-Ο σχεδίαση, η πρώτη μας έγνοια είναι να προσπαθήσουμε να προσδιορίσουμε τα αντικείμενα που περιλαμβάνονται στο πρόβλημα και όχι τις λειτουργίες. Τα ευκολότερα αντικείμενα που μπορούμε να προσδιορίσουμε είναι εκείνα που υπάρχουν σε αντίστοιχη μορφή και στον πραγματικό κόσμο. Στο παράδειγμα, τα αντικείμενα είναι τα τούβλα, ο τοίχος, οι πλευρές, το κουπί, η μπάλα και το ίδιο το βιντεοπαιχνίδι, όπως φαίνονται στην Εικόνα 10.



Εικόνα 10

Τα αντικείμενα έχουν *κατάσταση* (πληροφορίες) και *συμπεριφορά* (διαδικασίες). Μια μπάλα μπορεί να αναπαρασταθεί από μια ακτίνα και μια θέση (Εικόνα 11). Ένα κουπί, μια πλευρά ή ένα τούβλο μπορούν να περιγραφούν από τη θέση τους, το πλάτος και το ύψος. Ανάλογα, η κατάσταση ενός παιχνιδιού αποτελείται από μια μπάλα, ένα κουπί, έναν τοίχο τούβλων και τέσσερις πλευρές. Τα αντικείμενα επίσης υποστηρίζουν μια συμπεριφορά – λειτουργίες που μπορεί να εφαρμοστούν για να τροποποιήσουν ή να «προβάλουν» τα δεδομένα τους. Για παράδειγμα, μια μπάλα ανταποκρίνεται στα αιτήματα να αναφέρει ή να τροποποιήσει τη θέση της. Ομοίως, μια μπάλα μπορεί να ερωτηθεί εάν βρίσκεται πίσω από το κουπί παιχνιδιών ή εάν συγκρούεται με οποιαδήποτε από τα άλλα στοιχεία του παιχνιδιού. Μπορούμε να εκτελέσουμε παρόμοιες αναλύσεις πάνω στα άλλα αντικείμενα του παιχνιδιού.

Videogame	**Μπάλα**	**Τούβλο**	**Πλευρές**	**Κουπί**
Μπάλα	Θέση	Θέση	Θέση	Θέση
Κουπί	ακτίνα	Πλάτος	Πλάτος	Πλάτος
Πλευρές		Ύψος	Ύψος	Ύψος
τοίχος				

Εικόνα 11

Σημαντικό είναι να αναγνωρίσουμε τις συσχετίσεις μεταξύ των αντικειμένων. Στο παράδειγμα με το παιχνίδι, το κουπί και η μπάλα συσχετίζονται. Όταν το κουπί πετύχει τη μπάλα, η μπάλα θα αλλάξει κατεύθυνση. Παρόμοια είναι και η περίπτωση όταν η μπάλα πετύχει τον τοίχο. Επίσης μπορούμε να αναγνωρίσουμε και συσχετίσεις του τύπου ‘αποτελείται-από’. Τέτοια σχέση υπάρχει μεταξύ των τούβλων και του τοίχου - ο τοίχος είναι κατασκευασμένος από τούβλα. Με αυτό τον τρόπο μπορούμε να σχηματίσουμε μια εικόνα για το πώς αλληλεπιδρούν τα διάφορα μέρη του παιχνιδιού. Ουσιαστικά, η Ο-Ο σχεδίαση θεωρεί ότι μια εφαρμογή είναι μια συλλογή από συνεργαζόμενα αντικείμενα.

Αντί να «ανακαλύπτουμε τα αντικείμενα» στο χώρο του προβλήματος, συνηθίζεται να «αναζητούμε τις κλάσεις». Οι κλάσεις είναι ο θεμελιώδης μηχανισμός αφαίρεσης στην Ο-Ο σχεδίαση. Αναγνωρίζουν το γεγονός ότι συλλογές από σχετιζόμενα αντικείμενα μοιράζονται κοινά χαρακτηριστικά και συμπεριφορά. Για παράδειγμα, όλα τα τούβλα στον τοίχο του παραδείγματός μας με το παιχνίδι παρουσιάζουν την ίδια συμπεριφορά. Το κάθε τούβλο ξεχωριστά ανήκει στην κλάση ‘Τούβλο’. Όλα τα ξεχωριστά είδη της κάθε κλάσης ακολουθούν την κοινή δομή και συμπεριφορά που περιγράφεται από την κλάση. Το κάθε είδος ξεχωριστά διατηρεί τη δική του κατάσταση. Κάθε τούβλο, για παράδειγμα, διατηρεί τη θέση του στον τοίχο.

10 Σχεδίαση CRC

Η μέθοδος Class-Responsibility-Collaborators (CRC – Κλάσεις-Ευθύνες-Συνεργάτες) προτάθηκε το 1989[1]. Είναι μία τεχνική σχεδίασης που χρησιμοποιείται ως αφετηρία για την Ο-Ο σχεδίαση μιας εφαρμογής. Υπάρχουν τρία βασικά βήματα σε μία CRC σχεδίαση:

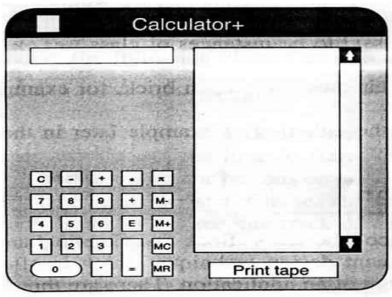
1) Βρίσκουμε τις κλάσεις, δηλαδή προσδιορίζουμε τα αντικείμενα που ενέχονται και τις κλάσεις τους.
2) Καθορίζουμε για ποια θέματα είναι αρμόδιες , δηλαδή τι κάνουν (συμπεριφορά).
3) Καθορίζουμε τους συνεργάτες τους, δηλαδή άλλες κλάσεις που τις χρειάζονται για να φέρουν σε πέρας τις δουλειές τους.

Για να τεκμηριωθεί η σχεδίαση και να περιοριστούν οι λεπτομέρειες που μπορεί να παρουσιαστούν, υπάρχουν λίστες που καταγράφονται πάνω σε CRC φόρμες, οι οποίες συχνά χρησιμοποιούνται για να τεκμηριώσουν κάθε κλάση. Η εμφάνιση μιας CRC φόρμας φαίνεται στην Εικόνα 12. Είναι βέβαια φυσικό να χρησιμοποιείται ένα εργαλείο λογισμικού για την τήρηση αυτών των λιστών.

Όνομα κλάσης	
Ευθύνες	Συνεργάτες
•…	•…
•…	•…
•…	•…
•…	•…

Εικόνα 12

Για παράδειγμα θα σχεδιάσουμε με την τεχνική CRC μια αριθμομηχανή της οποίας η διεπαφή με το χρήστη φαίνεται στην Εικόνα 13. Θα θέλατε ίσως να επιχειρήσετε μια δική σας σχεδίαση πριν διαβάσετε παρακάτω.



Εικόνα 13

Η περιγραφή για την αριθμομηχανή ακολουθεί:
Η αριθμομηχανή αποτελείται από έναν αριθμό πλήκτρων και μία οθόνη. Υπάρχει ένα πλήκτρο για κάθε ψηφίο (από το 0 μέχρι το 9) το οποίο επιτρέπει την εισαγωγή αριθμών που προβάλλονται στην οθόνη. Υπάρχουν πλήκτρα με τα οποία γίνονται υπολογισμοί – πρόσθεση, διαίρεση κλπ. Το αποτέλεσμα της πράξης παρουσιάζεται στην οθόνη.

Η αριθμομηχανή έχει ακόμη ένα μια προσομοιωνόμενη χαρτοταινία που μπορεί να κινηθεί προς τα πάνω ή προς τα κάτω χρησιμοποιώντας μια ράβδο κύλισης. Ένα πλήκτρο επιτρέπει την εκτύπωση των περιεχομένων της χαρτοταινίας.

10.1 Βρίσκοντας τις κλάσεις

Ο προσδιορισμός των κλάσεων (και των αντικειμένων) είναι το κλειδί για την Ο-Ο σχεδίαση. Μία συνηθισμένη μέθοδος για να προσδιορίσουμε αρχικά τις κλάσεις είναι να εξετάσουμε την προδιαγραφή του προβλήματος. Μία σχεδίαση φυσιολογικά ξεκινά από έναν σχετικά λεπτομερή καθορισμό του προβλήματος. Η προδιαγραφή είναι σημαντική για κάθε σχεδίαση, αλλά ειδικά για την Ο-Ο σχεδίαση είναι κρίσιμη. Διαβάστε την προδιαγραφή και εντοπίστε τα ουσιαστικά. Τα ουσιαστικά είναι καλοί υποψήφιοι για τα αντικείμενα μέσα στο πρόγραμμα.

Η διαδικασία της ονομασίας των κλάσεων είναι πολύ σημαντική. Γενικά τα ονόματα των κλάσεων θα είναι ουσιαστικά και πρέπει να χρησιμοποιούνται όροι του προβλήματος.

Δεν υπάρχει συγκεκριμένη φόρμουλα για να βρούμε τη σωστή συλλογή των αντικειμένων που μοντελοποιούν το πρόβλημα. Επιπλέον, η πορεία είναι διερευνητική και αναδρομική. Είναι συνηθισμένο σε μια σχεδίαση να προτείνονται κάποιες κλάσεις, αλλά έπειτα να απορρίπτονται, καθώς η σχεδίαση εξελίσσεται. Επίσης, η ανάγκη για κάποιες κλάσεις δεν εμφανίζεται μέχρι τη φάση της υλοποίησης. Το να προσδιορίσει κανείς τις κλάσεις (και τα αντικείμενα) απαιτεί διορατικότητα, γνώση και κατανόηση του προβλήματος.

Στο παράδειγμα με την αριθμομηχανή προτείνονται οι κλάσεις `Calculator`, `Display`, `Button`, `PrintTape`.

Είναι σημαντικό στις αρχικές φάσεις της σχεδίασης να μην προχωρούμε σε πολλές λεπτομέρειες στην σχεδίαση. Ακόμη δεν είναι η κατάλληλη στιγμή να αρχίσουμε να λαμβάνουμε υπόψη διαφορετικούς τύπους πλήκτρων, πχ πλήκτρα ψηφίων, πλήκτρα πράξεων ή πλήκτρα εκτύπωσης. Αυτό είναι ένα σημαντικό θέμα, αλλά θα πρέπει να το αντιμετωπίσουμε αργότερα. Οι κλάσεις θα πρέπει να είναι κλάσεις για τους 'χρήστες'. Κλάσεις για τους 'προγραμματιστές', όπως στοίβες που τηρούν μερικά αποτελέσματα των υπολογισμών, ή μία αριθμητική μονάδα για την εκτέλεση υπολογισμών, είναι σαφώς ακατάλληλες σε αυτή τη φάση.

10.2 Καθορίζοντας τις ευθύνες των κλάσεων

Το δεύτερο βήμα είναι να καθορίσουμε τι υποτίθεται ότι πρέπει να κάνουν οι κλάσεις, δηλαδή ποια είναι η συμπεριφορά τους. Ο σκοπός είναι να καθορίσουμε το ρόλο και τα καθήκοντα κάθε κλάσης μέσα στο σύστημα σε μία ή δύο σύντομες αγγλικές προτάσεις. Ενώ οι κλάσεις είναι ουσιαστικά, οι ευθύνες είναι ρήματα. Πάλι διαβάζουμε την προδιαγραφή για να τις βρούμε. Για κάθε μία από τις κλάσεις που προσδιορίσαμε νωρίτερα, φτιάχνουμε μια λίστα ενεργειών .

`Calculator`

- Προβάλει τον εαυτό του μεταβιβάζοντας τα αιτήματα σε πλήκτρα, οθόνη και χαρτοταινία.
- Υπολογίζει άθροισμα, γινόμενο, διαφορά κλπ μεταξύ δύο αριθμών.

[1] Beck K, Cunnigham W, A laboratory for teaching object-oriented thinking, *ACM Sigplan Notices*, sel. 1-6, Οκτ. 1989.

- Προβάλει τα αποτελέσματα ενός υπολογισμού ζητώντας από την οθόνη να το κάνει.
- Προβάλει τον υπολογισμό στην χαρτοταινία ζητώντας από την ίδια να το κάνει.
- Εκτυπώνει τα περιεχόμενα της χαρτοταινίας ζητώντας της να το κάνει.

`Display`
- Προβάλει οτιδήποτε του ζητηθεί αλλά δε μπορεί να υπολογίσει τίποτα.

`Button`
- Ενημερώνει την αριθμομηχανή κάθε φορά που πιέζεται.
- Διατηρεί το ψηφίο ή τη λειτουργία που αντιπροσωπεύει.
- Προβάλει τον εαυτό του.

`PrintTape`
- Προβάλει οτιδήποτε της ζητηθεί να εμφανίσει πάνω σε μια εξομοίωση της χαρτοταινίας, η οποία μπορεί να μετακινηθεί προς τα πάνω και προς τα κάτω.
- Εάν ζητηθεί, εκτυπώνει τα περιεχόμενα της χαρτοταινίας.

10.3 Καθορίζοντας τους συνεργάτες

Τέλος, για κάθε κλάση καθορίζουμε τους συνεργάτες, δηλαδή αυτές τις κλάσεις χωρίς των οποίων τη βοήθεια δε θα ήταν δυνατό για την κλάση να ανταποκριθεί στις ευθύνες της. Μια συνεργασία υποδηλώνει μια αμφίδρομη σχέση, αλλά συχνά η σχέση είναι μονόδρομη. Μια κλάση ζητά από μια κλάση-συνεργάτη να εκτελέσει κάποια εργασία. Ο όρος συνεργάτης μπορεί να αντικατασταθεί με τον όρο ‘βοηθός’. Για το πρόγραμμα της αριθμομηχανής, οι συνεργάτες για κάθε κλάση είναι:

`Calculator`
- `Display, Button, PrintTape`
- Μια αριθμομηχανή γνωρίζει για την οθόνη, τα πλήκτρα και την χαρτοταινία, καθώς χρειάζεται να είναι ικανή να τους λέει να προβάλλονται.

`Button`
- `Calculator`
- Ένα πλήκτρο θα πρέπει να γνωρίζει την αριθμομηχανή, καθώς θα πρέπει να λέει στην αριθμομηχανή ότι έχει πατηθεί. Κάθε πλήκτρο πρέπει επίσης να έχει μια ένδειξη έτσι ώστε να μπορεί να διακρίνεται από τα άλλα πλήκτρα. Η ένδειξη του πλήκτρου θα είναι μια συμβολοσειρά.

`Display`
- -
- Η οθόνη δεν έχει συνεργάτες. Δουλεύει ανεξάρτητα όταν τροφοδοτείται με ένα αποτέλεσμα για να το προβάλει.

`PrintTape`
- `Printer`
- Η χαρτοταινία μπορεί να εμφανίσει οποιαδήποτε πληροφορία της ζητηθεί πάνω στην προσομοίωση χαρτιού, αλλά θα πρέπει να ζητήσει από ένα πραγματικό εκτυπωτή να εκτυπώσει τα περιεχόμενά της.

Ο προσδιορισμός των συνεργατών δίνει μια ιδέα στο πώς επικοινωνούν μεταξύ τους τα αντικείμενα των κλάσεων για να εκτελέσουν κάποια καθήκοντα. Έτσι ξεκινάμε να έχουμε μια αίσθηση για τη ροή μηνυμάτων μεταξύ των αντικειμένων. Μπορούμε ήδη να καταλάβουμε ότι ένα πλήκτρο στέλνει μήνυμα πληροφορώντας την αριθμομηχανή ότι έχει πιεσθεί και η αριθμομηχανή πρέπει να στείλει μηνύματα στην οθόνη και την χαρτοταινία για να προβάλουν κάτι. Η Εικόνα 14 δείχνει μια συμπληρωμένη φόρμα CRC για την κλάση `Button`.

Κλάση `Button`	
Ευθύνες •Μπορεί να πει στην Αριθμομηχανή ότι έχει πατηθεί •Γνωρίζει ποιο ψηφίο αντιπροσωπεύει •Μπορεί να προβάλει τον εαυτό του	Συνεργάτες •Αριθμομηχανή

Εικόνα 14

Μέχρι τώρα έχουμε ολοκληρώσει μία Ο-Ο σχεδίαση για το πρόγραμμα της αριθμομηχανής. Έχουμε προσδιορίσει τις υποψήφιες κλάσεις για το πρόγραμμα και έχουμε ένα σύνολο από φόρμες CRC που εμφανίζουν τις αλληλεπιδράσεις μεταξύ των κλάσεων. Αργότερα θα δούμε πώς μπορούμε να βελτιώσουμε τη σχεδίαση.

Συνοπτικά , η σχεδίαση CRC εξασφαλίζει χρήσιμες πληροφορίες σχετικά με τις κλάσεις και τις μεθόδους και αναγνωρίζει πιθανούς τρόπους διάσπασης ενός προβλήματος. Η σχεδίαση CRC τείνει να είναι μια πολύ επαναληπτική διεργασία. Αναγνωρίζονται οι υποψήφιες κλάσεις. Μετά κάποιες γίνονται δεκτές, ενώ άλλες απορρίπτονται – ίσως επειδή δεν έχουν ευθύνες ή επειδή οι ευθύνες τους υπάγονται σε κάποια άλλη κλάση). Οι ευθύνες τείνουν να μεταναστεύουν από τη μια κλάση στην άλλη καθώς κατανοούμε καλύτερα τα αντικείμενα και τον ρόλο τους μέσα στο πρόβλημα.

11 Λεπτομερειακή σχεδίαση

Η σχεδίαση CRC είναι μια διερευνητική διαδικασία η οποία μπορεί να αποτελέσει τη βάση μιας Ο-Ο σχεδίασης. Εδώ παρουσιάζουμε τεχνικές που προχωρούν μετά το CRC και επιτρέπουν στο σχεδιαστή να αναλύσει, να διευκρινίσει και να προσδιορίσει περαιτέρω ένα σχεδιασμό. Αυτές οι τεχνικές είναι:

- Λεπτομερειακός προσδιορισμός ευθυνών με περιπτώσεις-χρήσης (use-cases)
- Προσδιορισμός σχέσεων μεταξύ κλάσεων
- Οργάνωση κλάσεων σε ιεραρχίες
- Ομαδοποίηση ευθυνών
- Αναζήτηση επαναχρησιμοποιήσιμων πλαισίων σχεδίασης

Λεπτομερειακός προσδιορισμός ευθυνών με περιπτώσεις-χρήσης

Η σχεδίαση CRC περιγράφει σε συντομία τις ευθύνες της κλάσης σε φυσική γλώσσα. Τελικά, αυτές οι ευθύνες πρέπει να αναλυθούν, ώστε να μπορούμε με ακρίβεια να περιγράψουμε τις υπηρεσίες που προσφέρει κάθε κλάση. Για κάθε κλάση πρέπει να αναγνωρίσουμε την κατάσταση (δεδομένα), που τηρείται από κάθε στιγμιότυπο και το πρωτόκολλο των μηνυμάτων στα οποία απαντά η κλάση.

Οι *περιπτώσεις-χρήσης* (use-cases, μερικές φορές λέγονται και εκτελέσιμα σενάρια) είναι ένας απλός μηχανισμός για να μάθουμε περισσότερα για τον τρόπο αλληλεπίδρασης και επικοινωνίας των αντικειμένων. Η ιδέα είναι να διαλέξουμε κάποια αντιπροσωπευτικά παραδείγματα του τρόπου που θα χρησιμοποιηθεί η εφαρμογή και να ακολουθήσουμε το μονοπάτι των μηνυμάτων πού στέλνονται. Για παράδειγμα, στην αριθμομηχανή μια περίπτωση-χρήσης μπορεί να είναι ότι ο χρήστης πληκτρολογεί «10 + 1 =». Οι περιπτώσεις-χρήσης συλλαμβάνουν την κατάσταση, τη συμπεριφορά και την κυκλοφορία μηνυμάτων στην εφαρμογή. Συγκεκριμένα, συλλαμβάνουν

- τα αντικείμενα που εμπλέκονται στην περίπτωση-χρήσης,
- τα σημαντικότερα μηνύματα που ανταλλάσσονται ανάμεσα σε αντικείμενα, και
- τη ροή ελέγχου.

Γενικά, οι περιπτώσεις-χρήσης επικεντρώνονται στη συμπεριφορά σε κάποιο επίπεδο αφαίρεσης, δηλαδή δεν αποσκοπούν στην απογραφή όλων των μηνυμάτων.

Θα παρακολουθήσουμε την περίπτωση-χρήσης «10 + 1 =» για την αριθμομηχανή, αγνοώντας την αλληλεπίδραση με την αριθμομηχανή.

1. Πρώτα, πάτησε το 1. Το πλήκτρο με τον αριθμό 1 λέει στην αριθμομηχανή «το 1 έχει πατηθεί». Η αριθμομηχανή αποθηκεύει το 1 και λέει στην οθόνη «πρόβαλε τον αριθμό 1».
2. Μετά, πάτησε το 0. Το πλήκτρο με τον αριθμό 0 λέει στην αριθμομηχανή «το 0 έχει πατηθεί». Η αριθμομηχανή αποθηκεύει το 0 επισυνάπτοντάς το στο προηγούμενο ψηφίο, για να πάρουμε τον αριθμό 10 και λέει στην οθόνη «πρόβαλε τον αριθμό 1».
3. Μετά, πάτησε το +. Το πλήκτρο με το σύμβολο + λέει στην αριθμομηχανή «το + έχει πατηθεί». Η αριθμομηχανή αποθηκεύει το + (δεν μπορεί να προσθέσει ακόμη γιατί ο δεύτερος όρος της πρόσθεσης δεν υπάρχει).
4. Μετά, πάτησε το 1. Το πλήκτρο με τον αριθμό 1 λέει στην αριθμομηχανή «το 1 έχει πατηθεί». Η αριθμομηχανή αποθηκεύει το 1 σε μία περιοχή διαφορετική από εκεί που έχει αποθηκευτεί το 10 (προφανώς, χρειαζόμαστε ονόματα - θα τους ονομάσουμε όρος1, πράξη και όρος2. Μέχρι τώρα, ο όρος1 είναι το 10, η πράξη είναι το σύμβολο + και ο όρος2 είναι το 1). Μετά η αριθμομηχανή λέει στην οθόνη «δείξε το 1».
5. Τέλος, πάτησε το =. Το πλήκτρο με το σύμβολο = λέει στην αριθμομηχανή «το = έχει πατηθεί». Η αριθμομηχανή εκτελεί «όρος1 πράξη όρος2», και λέει στην οθόνη «δείξε το αποτέλεσμα».

Τι έχουμε μάθει από μια τέτοια περίπτωση-χρήσης; Η κατάσταση που πρέπει να διατηρείται από την αριθμομηχανή είναι πιο συγκεκριμένη - χρειάζεται να τηρούμε τα όρος1, πράξη, όρος2. Επίσης, αναγνωρίσαμε το πρωτόκολλο για διάφορα αντικείμενα. Για παράδειγμα, η οθόνη της αριθμομηχανής πρέπει να υποστηρίζει μηνύματα όπως:

- `display: aString`
- `appendToDisplay: aString`

Επίσης, θα ήταν άκομψο να αποκρίνεται η αριθμομηχανή ξεχωριστά σε κάθε ψηφίο, πχ σκεφτείτε 0,1,2… Θα χρειαστεί να αναπτύξουμε ένα γενικό πρωτόκολλο όπου η ετικέτα του κουμπιού που πατήθηκε να περνάει ως κομμάτι του μηνύματος: `digitButton: aButtonLabel`

Η ανάλυση της περίπτωσης-χρήσης συνεχίζεται διαλέγοντας άλλα αντιπροσωπευτικά παραδείγματα για να εκθέσουμε περισσότερες λεπτομέρειες της κατάστασης και της συμπεριφοράς των κλάσεων που εμπλέκονται σε μια εφαρμογή.

Συνοψίζοντας, οι περιπτώσεις-χρήσης βοηθούν στον προσδιορισμό της κατάστασης και της συμπεριφοράς των κλάσεων σε μία εφαρμογή και να δίνουν μια καλύτερη εικόνα για την αλληλεπίδραση των αντικειμένων σε μια εφαρμογή.

Προσδιορισμός σχέσεων μεταξύ κλάσεων, οργάνωση κλάσεων σε ιεραρχίες και ομαδοποίηση ευθυνών

Αξίζει να επαναλάβουμε ότι ένας από τους βασικούς στόχους του αντικειμενοστρεφούς προγραμματισμού είναι να παράγει γενικά, επαναχρησιμοποιήσιμα τμήματα λογισμικού, τα οποία μπορούν να επαναχρησιμοποιηθούν στην εφαρμογή για την οποία κατασκευάστηκαν, αλλά και σε άλλες μελλοντικές εφαρμογές. Οι έννοιες της κληρονομικότητας και της εξειδίκευσης επιτρέπουν:

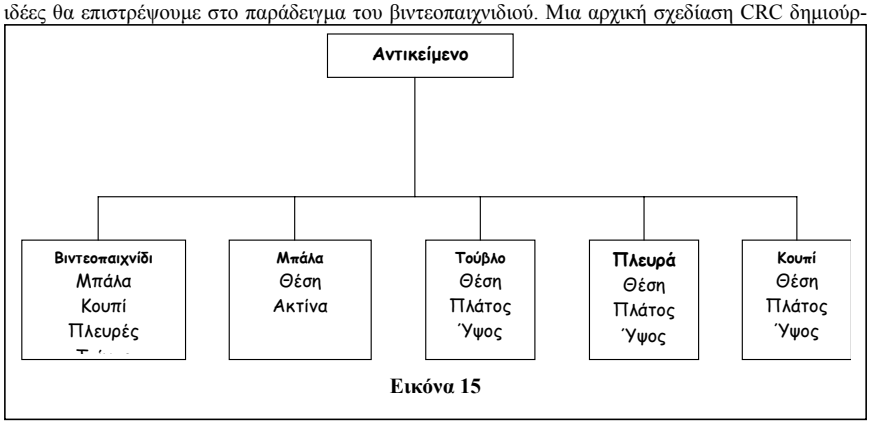
- Νέες κλάσεις να περιγραφούν ως επεκτάσεις ή εξειδικεύσεις κλάσεων που ήδη υπάρχουν.
- Υποκλάσεις να κληρονομήσουν τη συμπεριφορά και την κατάσταση των υπέρ-κλάσεων.

Τα πλεονεκτήματα της κληρονομικότητας και της εξειδίκευσης είναι:

- Στο επίπεδο του σχεδιασμού, σύντομοι και ξεκάθαροι ορισμοί – οι υποκλάσεις περιγράφονται μόνο στα σημεία που διαφέρουν από τις υπέρ-κλάσεις τους.
- Στο επίπεδο της υλοποίησης, η επαναχρησιμοποίηση κώδικα, αντί για την αντιγραφή του – οι υποκλάσεις επαναχρησιμοποιούν τον υπάρχοντα κώδικα των υπέρ-κλάσεων.

Περιγράψαμε τους μηχανισμούς που υλοποιεί αυτές τις έννοιες νωρίτερα. Αυτές οι έννοιες εισάγουν νέες διαστάσεις στη διαδικασία σχεδίασης. Ένας βασικός σχεδιαστικός στόχος είναι η ομαδοποίηση των ευθυνών μέσω μιας ιεραρχίας κλάσεων. Μια ευθύνη της υπέρ-κλάσης μπορεί να μοιραστεί σε κάθε υποκλάση της.

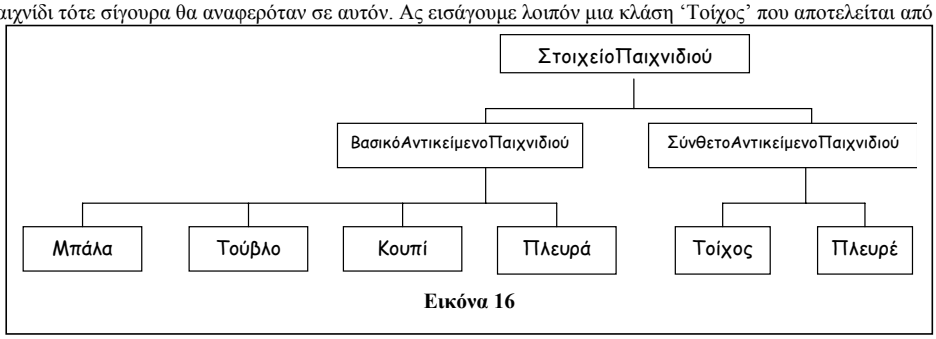
Για να εξερευνήσουμε αυτές τις ιδέες θα επιστρέψουμε στο παράδειγμα του βιντεοπαιχνιδιού. Μια αρχική σχεδίαση CRC δημιούργησε την ιεραρχία κλάσεων που φαίνεται στην Εικόνα 15. Θα υποθέσουμε ότι έχουμε απλή κληρονομικότητα (οι υποκλάσεις μπορούν να έχουν μία μόνο υπέρ-κλάση). Υπάρχει η κλάση 'Αντικείμενο' την οποία κληρονομούν όλες οι υπόλοιπες. Προς το παρόν, οι κλάσεις που συνιστούν το πρόγραμμα του βιντεοπαιχνιδιού είναι απολύτως ανεξάρτητες μεταξύ τους. Θα δούμε σε λίγο πώς μπορούμε να ανακαλύψουμε ομοιότητες μεταξύ κλάσεων και επομένως κληρονομικότητα.



Εικόνα 15

Χωρίς καν να σκεφτούμε τα πρωτόκολλα μηνυμάτων, είναι προφανές ότι όλα τα αντικείμενα που απαρτίζουν το παιχνίδι έχουν κάποια θέση και μέγεθος. Γι' αυτό ας εισάγουμε μια νέα κλάση, 'ΣτοιχείοΠαιχνιδιου', η οποία θα χειριστεί αυτές τις ευθύνες για λογαριασμό των υποκλάσεών της. Όμως πρέπει πρώτα να γενικεύσουμε τον ορισμό του μεγέθους – η μπάλα έχει ακτίνα αλλά άλλα στοιχεία έχουν πλάτος και ύψος. Γενικεύουμε λοιπόν το μέγεθος ως μια περιοχή – έτσι ο ίδιος ορισμός θα χρησιμοποιηθεί για όλα τα στοιχεία. Σημειώστε ότι η 'ΣτοιχείοΠαιχνιδιού' καλείται *αφηρημένη* κλάση. Δεν θα δημιουργήσουμε στιγμιότυπα αυτής της κλάσης – ο ρόλος της είναι να συλλαμβάνει την αφηρημένη έννοια της θέσης / μεγέθους. Θα υπάρξουν άλλες ευθύνες οι οποίες θα είναι συγκεκριμένες για κάθε υποκλάση.

Μπορούμε να βελτιώσουμε περαιτέρω το σχεδιασμό; Τι άλλες αφαιρέσεις θα ήταν χρήσιμες; Υπάρχουν τουλάχιστο δυο όψεις σε αυτό το ζήτημα. Μερικά από τα εξαρτήματα του παιχνιδιού είναι κινούμενα (η μπάλα και το κουπί), ενώ άλλα δεν είναι κινούμενα (τα τούβλα και οι πλευρές). Θα πρέπει να ορίσουμε κλάσεις 'ΚινούμενοΣτοιχείο' και 'ΑκίνητοΣτοιχείο'; Πώς κρίνουμε εάν αυτές οι κλάσεις είναι χρήσιμες ή όχι; Είναι σαφές ότι εάν λίγες ευθύνες μπορούν να μετατοπιστούν στην καινούργια αφηρημένη κλάση, τότε δεν έχουμε κερδίσει πολλά. Ίσως υπάρχει μια καλύτερη αφαίρεση. Η τρέχουσα σχεδίαση δεν έχει την έννοια του τοίχου, αλλά εάν κάποιος περιέγραφε το παιχνίδι τότε σίγουρα θα αναφερόταν σε αυτόν. Ας εισάγουμε λοιπόν μια κλάση 'Τοίχος' που αποτελείται από τούβλα. Χωρίς να προχωρούμε σε περισσότερες εξηγήσεις, προσέξτε μια σχεδίαση που αναπαρίσταται στην Εικόνα 16.



Εικόνα 16

Συνοψίζοντας, όταν επεξεργαζόμαστε μια Ο-Ο σχεδίαση οφείλουμε να προσδιορίζουμε τις σχέσεις μεταξύ των κλάσεων και να κατανέμουμε συμπεριφορές στις κλάσεις μέσα σε ένα σχήμα ιεραρχικής οργάνωσης. Σημαντικές σχέσεις μεταξύ κλάσεων τις οποίες έπρεπε να εντοπίσουμε είναι:

- Σχέσεις 'είναι-ένα'. Τις χειριζόμαστε με σχέσεις υπέρ-κλάση / υποκλάση. Για παράδειγμα μπάλες, τούβλα, κουπιά και πλευρές είναι όλα βασικά αντικείμενα του παιχνιδιού.
- Σχέσεις 'είναι-παρόμοιο'. Αναγνωρίζουμε ότι αντικείμενα όπως μπάλες, τούβλα, κλπ έχουν όλα ιδιότητες, όπως θέση και μέγεθος.
- Σχέσεις 'είναι-τμήμα'. Το κουπί, η μπάλα και τα τούβλα είναι στοιχεία του βιντεοπαιχνιδιού. Παρομοίως, ο τοίχος αποτελείται από τούβλα.

Η κατανόηση των σχέσεων των κλάσεων είναι κρίσιμη ώστε να παραχθεί ποιοτική αντικειμενοστρεφής σχεδίαση.

Αναζήτηση επαναχρησιμοποιήσιμων πλαισίων σχεδίασης

Η επαναχρησιμοποίηση μπορεί να επιτευχθεί με την χρήση κληρονομικότητας και αφηρημένων κλάσεων. Ένας απώτερος σκοπός και πολύ πιο δύσκολος είναι να προσδιορίσουμε επαναχρησιμοποιήσιμα πλαίσια σχεδίασης, δηλαδή υποσυστήματα συνεργαζόμενων κλάσεων που μπορούν να επαναχρησιμοποιηθούν για συγκεκριμένα πεδία εφαρμογών.

Ίσως το παλαιότερο παράδειγμα ενός πλαισίου σχεδίασης είναι η αρχιτεκτονική μοντέλου - όψης - ελεγκτή (Model-View-Controller, MVC) που εισήχθηκε από τη Smalltalk για την υλοποίηση γραφικών διεπαφών (Graphical User Interfaces, GUI). Τα συστατικά της γραφικής διεπαφής διαχωρίζονται στα τρία τμήματα της αρχιτεκτονικής. Το μοντέλο περιλαμβάνει δεδομένα, κλάσεις και αντικείμενα για την εφαρμογή, η όψη προβάλει τις πληροφορίες του μοντέλου και ο ελεγκτής χειρίζεται τα συμβάντα του ποντικιού και του πληκτρολογίου. Αυτή η αρχιτεκτονική έχει μερικά αξιοσημείωτα χαρακτηριστικά. Επιτρέπει πολλαπλές όψεις του ίδιου μοντέλου, καθώς επίσης και την χρησιμοποίηση αυτών σαν μέρη κατασκευής μέσα σε μεγαλύτερα τμήματα, δηλαδή καινούργια είδη όψεων μπορούν να κατασκευαστούν χρησιμοποιώντας ήδη υπάρχουσες όψεις ως υπό-όψεις. Η βιβλιοθήκη της Smalltalk περιλαμβάνει μια μεγάλη βιβλιοθήκη όψεων και ελεγκτών με συμπεριφορά σε κείμενο, γραφικά και λίστες για παράδειγμα. Αυτά μπορούν να εξειδικευτούν ακόμα περισσότερο από τους προγραμματιστές.

Η σημαντική συνεισφορά της αρχιτεκτονικής MVC δεν είναι τα τρία τμήματά της, αλλά η αλληλεπιδράσεις μεταξύ τους, και η αναγνώριση ότι αυτό το πλαίσιο αλληλεπίδρασης είναι επαναχρησιμοποιήσιμο.

Επαναχρησιμοποιήσιμα πλαίσια σχεδίασης προορίζονται για συγκεκριμένα πεδία εφαρμογών, όπως διεπαφές χρήστη, συστήματα αρχείων και δρομολόγηση διεργασιών. Η αναγνώριση αυτών των οικογενειών συνεργαζομένων κλάσεων προσθέτει μια άλλη διάσταση - πρόκληση στην διαδικασία σχεδίασης.

12 Η σχεδίαση είναι προγραμματισμός και ο προγραμματισμός είναι σχεδίαση

Η αντικειμενοστρεφής σχεδίαση είναι επαναληπτική διαδικασία. Σε κάθε επανάληψη, η σχεδίαση εξελίσσεται. Ενώ στην παραδοσιακή ανάπτυξη λογισμικού, σχεδίαση και υλοποίηση θεωρούνται ξεχωριστές κατά βάση δραστηριότητες, στην Ο-Ο μέθοδο οι δύο δραστηριότητες είναι συχνά δυσδιάκριτες. Αυτό συμβαίνει για τρεις κυρίως λόγους:

6. Η πρωτοτυποποίηση είναι ένα αναπόσπαστο κομμάτι της Ο-Ο σχεδίασης και υλοποίησης. Κατασκευάζουμε πρωτότυπα διότι αναγνωρίζουμε ότι στις περισσότερες περιπτώσεις οι απαιτήσεις για ένα σύστημα είναι τουλάχιστον ασαφείς ή όχι πλήρως κατανοητές. Είναι μία διερευνητική διαδικασία, με την οποία ελέγχουμε την εγκυρότητα της ανάλυσης, της σχεδίασης και των επιλογών διεπαφής χρήστη. Η πρωτοτυποποίηση υπαγορεύει ότι η σχεδίαση και η υλοποίηση προχωρούν σε μεγάλα – αλυσιδωτά επαναλαμβανόμενα βήματα.
7. Οι δραστηριότητες που συμβαίνουν κατά τη διάρκεια της βελτίωσης μιας Ο-Ο σχεδίασης είναι παρόμοιες και στη σχεδίαση ενός λειτουργικού πρωτοτύπου. Ακόμη, δραστηριότητες όπως η αναδιοργάνωση της ιεραρχίας κλάσεων συμβαίνουν τόσο κατά την υλοποίηση του πρωτοτύπου, όσο και κατά τη σχεδίαση.
8. Οι σχεδιαστές πρέπει να είναι πολύ πιο ενημερωμένοι σε ό,τι αφορά το περιβάλλον ανάπτυξης, εξαιτίας των επιπτώσεων που έχουν οι επαναχρησιμοποιημένες βιβλιοθήκες κλάσεων στη σχεδίαση.

Ο Meyer[2] υποστηρίζει ότι σχεδίαση και υλοποίηση είναι στιγμιότυπα της ίδιας δραστηριότητας - του προγραμματισμού.

13 Μεθοδολογίες αντικειμενοστρεφούς σχεδίασης

Στη λογοτεχνία έχουν παρουσιαστεί αρκετές μεθοδολογίες Ο-Ο σχεδίασης, πράγμα που αντανακλά τη σχετική ανωριμότητα της επιστημονικής αυτής περιοχής σε σύγκριση με τη δομημένη σχεδίαση. Γενικά είναι αποδεκτό ότι ο Ο-Ο προγραμματισμός είναι πιο κατανοητός από την Ο-Ο σχεδίαση, που με τη σειρά της είναι καλύτερα κατανοητή από την Ο-Ο ανάλυση που με τη σειρά της είναι καλύτερα κατανοητή από την Ο-Ο διαχείριση έργου.

Προσεγγίσαμε τη σχεδίαση κυρίως με βάση τη μεθοδολογία που ονομάζεται οδηγούμενα-από-ευθύνες σχεδίαση (responsibility-driven design[3]). Άλλες γνωστές μεθοδολογίες περιλαμβάνουν object-oriented design, HOOP, OOSD. Καθεμιά από τις μεθοδολογίες χρησιμοποιεί (διαφορετική) διαγραμματική απεικόνιση για να παρουσιάζει την τελική σχεδίαση.

14 Συζήτηση και αξιολόγηση

Σε ότι αφορά την επιρροή στους προγραμματιστές, ο Ο-Ο προγραμματισμός προβλεπόταν να είναι τη δεκαετία του 1990, αυτό που ήταν ο δομημένος προγραμματισμός στη δεκαετία του 1970. Αυτό πράγματι συνέβη - η Ο-Ο σχεδίαση και προγραμματισμός αποτελεί την κυρίαρχη προσέγγιση στην σύγχρονη ανάπτυξη λογισμικού. Σε αυτό βοήθησε η κυκλοφορία και ευρεία χρήση γλωσσών προγραμματισμού που υποστηρίζουν το Ο-Ο στιλ, όπως οι C++, Ada, Smalltalk και Java.

Τα ισχυρά σημεία της Ο-Ο προσέγγισης είναι:

- η φυσικότητα της μοντελοποίηση της εφαρμογής με αντικείμενα, κλάσεις και μεθόδους
- η αρθρωσιμότητα (σε συνδυασμό με την απόκρυψη πληροφορίας) που προσφέρουν οι κλάσεις
- η ευκολία επαναχρησιμοποίησης στοιχείων λογισμικού λόγω κληρονομικότητας

Τις τελευταίες δεκαετίες η Ο-Ο μεθοδολογία έχει κυριαρχήσει στη σχεδίαση προγράμματος και στην ανάπτυξη λογισμικού.

---

[2] Meyer B, *Object-Oriented Software construction,* Prentice-Hall, 1988.

[3] Wirfs-Brock R J and Johnson Ralph E, 'Surveying current research in O-O design, *Communications of the ACM,* vol. 33, no. 9, pp.104-124, Sept. 1990.

Περίληψη αξιολόγησης

Ιδιαίτερα χαρακτηριστικά και πλεονεκτήματα
Ξεφεύγει από τις προηγούμενες προσεγγίσεις της σχεδίασης παρέχοντας μια σφαιρική άποψη της σχέσης μεταξύ δεδομένων και λειτουργιών. Εξασφαλίζει μέγιστη επαναχρησιμοποίηση μέσω του μηχανισμού της κληρονομικότητας.

Μειονεκτήματα
Η σχεδίαση δεν απορρέει από καλά καθορισμένα βήματα. Απαιτείται πιο βαθιά και δημιουργική προσέγγιση.

Φιλοσοφία / Προοπτική
Η θεωρία πίσω από τη μέθοδο είναι ότι η δομή του προγράμματος πρέπει να είναι ένα μοντέλο ή μία προσομοίωση της περιοχής του προβλήματος. Ένα τέτοιο μοντέλο εκφράζεται μέσω αντικειμένων που αντιπροσωπεύουν πιστά τα στοιχεία της εφαρμογής.

Συστηματική;
Αρκετά. Η αναγνώριση αντικειμένων, κλάσεων και κληρονομικότητας γίνεται κατά περίπτωση. Πληθώρα τύπων διαγραμμάτων βοηθούν στη διαδικασία σχεδίασης και την τεκμηρίωση των σχεδιαστικών επιλογών.

Κατάλληλες εφαρμογές
Όλες. Η Ο-Ο σχεδίαση ταιριάζει ιδιαίτερα σε προγράμματα που οδηγούνται από συμβάντα. Τέτοια είναι τα προγράμματα που αναφέρθηκαν ως παραδείγματα. Το πρόγραμμα ανταποκρίνεται σε διάφορα συμβάντα που προκαλεί ο χρήστης μέσω του πληκτρολογίου και του ποντικιού. Κάθε συμβάν εξυπηρετείται με κλήση μιας μεθόδου του κατάλληλου αντικειμένου.

Ακατάλληλες εφαρμογές
Καμία.

Πάνω-προς-κάτω;
Όχι. Όταν ένα πρόγραμμα έχει σχεδιαστεί χρησιμοποιώντας την Ο-Ο μέθοδο είναι ένα δίκτυο αντικειμένων χωρίς προφανή ιεραρχία. Κατά τη σχεδίαση αναγνωρίζονται οι ιεραρχίες κλάσεων, αλλά αυτή η διαδικασία δεν είναι απαραίτητο να εκτελείται από πάνω προς τα κάτω.

Καλό για μεγάλης κλίμακας σχεδιασμό;
Ναι. Η Ο-Ο σχεδίαση έχει εφαρμοστεί με επιτυχία σε πολλά μεσαία και μεγάλα προγράμματα και συστήματα λογισμικού. Πιο συγκεκριμένα η έμφαση που δίνεται στην επαναχρησιμοποίηση βοηθάει ώστε νέα και μεγάλα προγράμματα να μπορούν να σχεδιαστούν εύκολα.

Καλό για μικρής κλίμακας σχεδιασμό;
Όχι. Το τελικό προϊόν της Ο-Ο σχεδίασης είναι η δομή ενός προγράμματος σε μεγάλη κλίμακα, εκφρασμένη με κλάσεις, αντικείμενα και μεθόδους. Ο λεπτομερής σχεδιασμός των μεθόδων πρέπει να γίνει χρησιμοποιώντας άλλη προσέγγιση.

Υποστήριξη εργαλείων
Διατίθενται εργαλεία που υποστηρίξουν τη δημιουργία, σύνταξη, επόπτευση και μετασχηματισμό διαφόρων τύπων διαγραμμάτων που χρησιμοποιεί η μέθοδος.

15 Περίληψη

Η Ο-Ο σχεδίαση χαρακτηρίζεται από:

- **Αναζήτηση των κλάσεων:** Αναγνωρίζονται οι κλάσεις (και τα αντικείμενα) που αρμόζουν για συγκεκριμένο πεδίο προβλημάτων.
- **Λεπτομερή ορισμό ευθυνών των κλάσεων:** Καθορίζεται ο ρόλος και οι ευθύνες κάθε κλάσης μέσα στο σύστημα.
- **Προσδιορισμό συνεργατών κάθε κλάσης:** Καθορίζεται εκείνες οι κλάσεις χωρίς τη συνδρομή των οποίων μια κλάση δεν μπορεί να φέρει εις πέρας τις ευθύνες της.
- **Εκλέπτυνση ευθυνών με περιπτώσεις-χρήσης:** Μαθαίνουμε πώς συμπεριφέρονται και επικοινωνούν τα αντικείμενα χρησιμοποιώντας σενάρια εκτέλεσης.
- **Αναγνώριση σχέσεων μεταξύ κλάσεων:** Προχωρούμε σε ποιο λεπτομερειακή σχεδίαση αναγνωρίζοντας σημαντικές σχέσεις μεταξύ κλάσεων, όπως ‘είναι-ένα’, ‘είναι-τμήμα’.
- **Οργάνωση των κλάσεων σε ιεραρχίες:** Ανακαλύπτουμε σχέσεις ‘είναι-ένα’ ανάμεσα σε κλάσεις χρησιμοποιώντας τις έννοιες της εξειδίκευσης και της γενίκευσης. Μεταφράζουμε τις ιεραρχίες κλάσεων σε επαναχρησιμοποιούμενο κώδικα μέσω κληρονομικότητας και αφηρημένων κλάσεων.
- **Ομαδοποίηση ευθυνών:** Μεταφέρουμε συμπεριφορά από μια κλάση σε άλλη υλοποιώντας συμπεριφορές έτσι ώστε κοινός κώδικας να μοιράζεται από όλες τις υποκλάσεις.
- **Αναζήτηση επαναχρησιμοποιήσιμων πλαισίων σχεδίασης:** Αναγνωρίζουμε υποσυστήματα συνεργαζόμενων κλάσεων που αλληλεπιδρούν με τρόπο που μπορεί να επαναχρησιμοποιηθεί σε κάποια ανάλογη περίπτωση.

Κλείνοντας, είναι σημαντικό να αντιληφθούμε ότι τα Ο-Ο συστήματα είναι ακόμα σε, σχετικά, βρεφικό στάδιο. Δεν υπάρχει μια ευρέως αποδεκτή μεθοδολογία Ο-Ο σχεδίασης.

Ασκήσεις

1. Μπορεί η αντικειμενοστρεφής σχεδίαση να χαρακτηρισθεί ως πάνω-προς-κάτω ή κάτω-προς-πάνω διαδικασία;
2. Αν ο προγραμματισμός και η σχεδίαση είναι δύο όψεις του ίδιου νομίσματος, όπως διατείνεται η Ο-Ο σχεδίαση, αυτό σημαίνει ότι όλοι οι σχεδιαστές πρέπει να είναι επίσης προγραμματιστές;
3. Σε ποιο βαθμό επηρεάζεται η Ο-Ο σχεδίαση από τις βιβλιοθήκες κλάσεων που διατίθενται; Σε ποιο βαθμό οι σχεδιαστές πρέπει να γνωρίζουν τις διαθέσιμες βιβλιοθήκες;
4. Συνεχίστε τη σχεδίαση της αριθμομηχανής που περουσιάστηκε. Συγκεκριμένα:
   - Σχεδιάστε περισσότερες περιπτώσεις-χρήσης.
   - Ομαδοποιήστε τις κλάσεις σε μια ιεραρχία
5. Τι χαρακτηριστικά ή ενδείξεις πρέπει να χρησιμοποιήσουμε για να προσδιορίσουμε πιθανά ελαττώματα σε μια Ο-Ο σχεδίαση; Για παράδειγμα, μια κλάση με υπερβολικό πλήθος ευθυνών θα έπρεπε να επανεξεταστεί.
6. Εφαρμόστε τις τεχνικές που παρουσιάστηκαν για να σχεδιάσετε μια ταμειολογιστική μηχανή (ΑΤΜ) τραπέζης. Το ΑΤΜ ανταποκρίνεται σε ακολουθίες εντολών. Υποθέστε ότι το ΑΤΜ είναι ικανό να εκτελεί τα παρακάτω:
   - Αναγνωρίζει κάθε χρήστη μέσω μιας προσωπικής κάρτας και ενός συγκεκριμένου κωδικού που εισάγει ο χρήστης.
   - Κάνει καταθέσεις και αναλήψεις σε λογαριασμούς.
   - Μεταφέρει κεφάλαια μεταξύ δύο λογαριασμών.
   - Προβάλει το υπόλοιπο ενός λογαριασμού.