

11.

Σύγχρονα προγραμματιστικά περιβάλλοντα



### Εισαγωγή

Η ανάγκη δημιουργίας σύγχρονων περιβαλλόντων εργασίας, προσανατολισμένων στη φιλικότητα επικοινωνίας με τον χρήστη, έφερε δραστικές αλλαγές στον προγραμματισμό και τα προγραμματιστικά εργαλεία. Σε ένα σύγχρονο προγραμματιστικό περιβάλλον οι έννοιες του αντικειμενοστραφούς, του οδηγούμενου από γεγονότα προγραμματισμού και της οπτικής σχεδίασης είναι βασικά χαρακτηριστικά. Παρέχονται μια σειρά από γραφικά εργαλεία για την πλήρη υποστήριξη και ευκολότερη και ταχύτερη ανάπτυξη προγραμμάτων. Ιδιαίτερο γνώρισμα των περιβαλλόντων αυτών είναι η έμφαση που δίνεται στο γραφικό περιβάλλον και στην επικοινωνία της εφαρμογής με το χρήστη. Ακόμη οι εφαρμογές δεν είναι απομονωμένες, αλλά συνεργάζονται με άλλες εφαρμογές και ετερογενή περιβάλλοντα.



### Διδακτικοί στόχοι

Στόχοι του κεφαλαίου αυτού είναι οι μαθητές:

- ⇒ να ορίζουν τις έννοιες αντικείμενο, ιδιότητα, γεγονός, μέθοδος,
- ⇒ να διατυπώνουν τη μεθοδολογία σχεδιασμού μιας εφαρμογής σε ένα σύγχρονο προγραμματιστικό περιβάλλον,
- ⇒ να αναλύουν μια εφαρμογή μέσα από αντικείμενα και γεγονότα,
- ⇒ να κατασκευάζουν απλά προγράμματα σε σύγχρονα προγραμματιστικά περιβάλλοντα συνδυάζοντάς τα με τις γνώσεις τους στον τμηματικό και στον δομημένο προγραμματισμό.



### Προερωτήσεις

- ✓ ποιες είναι οι τεχνικές προγραμματισμού που εφαρμόζονται σε ένα σύγχρονο προγραμματιστικό περιβάλλον;
- ✓ είναι εφικτό να χρησιμοποιήσουμε διαφορετικές τεχνικές προγραμματισμού στην ανάπτυξη μιας εφαρμογής;
- ✓ η εφαρμογή νέων τεχνικών προγραμματισμού καταργεί τις υπάρχουσες;
- ✓ όταν αναφερόμαστε σε εφαρμογή νέων τεχνικών προγραμματισμού κατ' ανάγκη δυσκολεύεται το έργο του προγραμματιστή;
- ✓ ποια τα πλεονεκτήματα ενός σύγχρονου προγραμματιστικού περιβάλλοντος;
- ✓ είναι εφικτό να επιτύχουμε τη συνεργασία διαφορετικών εφαρμογών σε ένα σύγχρονο περιβάλλον εργασίας;

### 11.1. Αντικειμενοστραφής προγραμματισμός

Τα κλασικά προγραμματιστικά περιβάλλοντα εργασίας, οδήγησαν στην ανάπτυξη πολύπλοκων προϊόντων λογισμικού, που η κατασκευή τους και η συντήρησή τους απαιτούσε ειδικούς με βαθιές γνώσεις σε επιμέρους θέματα, όπως τεχνικές διαχείρισης μνήμης, πρότυπα επικοινωνίας, ειδικές γλώσσες προγραμματισμού και λειτουργικά συστήματα. Η πολυπλοκότητα κατασκευής του λογισμικού δημιούργησε ταυτόχρονα προβληματισμούς και δρομολόγησε απόπειρες εξεύρεσης φιλικότερων τεχνικών και μεθόδων σχεδιασμού προγραμμάτων. Από την άλλη πλευρά η μοντελοποίηση του ανθρώπινου συλλογισμού μέσω της τεχνητής νοημοσύνης, έκανε φανερή την ανάγκη της δημιουργίας ενός διαφορετικού προγραμματιστικού περιβάλλοντος που θα ομαδοποιούσε στην ίδια οντότητα όλες τις πληροφορίες και ιδιότητες που αφορούσαν στην ίδια έννοια.

Ο αντικειμενοστραφής σχεδιασμός προγραμμάτων γεννιέται από την προσπάθεια αντιμετώπισης αυτών ακριβώς των ζητημάτων. Χρησιμοποιώντας τον όρο **αντικειμενοστραφής προγραμματισμός** (object - oriented programming) δεν αναφερόμαστε σε κάποιο συγκεκριμένο προϊόν ή μια γλώσσα προγραμματισμού, αλλά σε ένα διαφορετικό τρόπο προσέγγισης προβλημάτων με τον υπολογιστή.

Ο αντικειμενοστραφής προγραμματισμός ή η αντικειμενοστραφής σχεδίαση προέκυψε από τη στιγμή που απαντήθηκε διαφορετικά – από ότι απαντιόταν μέχρι τότε - το ερώτημα *“Η δομή των προγραμμάτων είναι προτιμότερο να στηρίζεται στις “ενέργειες” ή στα δεδομένα;”*. Η νέα απάντηση που δόθηκε, η απάντηση *“δεδομένα”*, προσδιορίζει και τη βασική διαφορά ανάμεσα στις παραδοσιακές προγραμματιστικές τεχνικές και στην αντικειμενοστραφή προσέγγιση η οποία περιγράφει *“ενέργειες”* (επεξεργασία) που εφαρμόζονται πάνω σε δεδομένα.

Η αντικειμενοστραφής σχεδίαση λοιπόν εκλαμβάνει σαν πρωτεύοντα δομικά στοιχεία ενός προγράμματος τα δεδομένα, από τα οποία δημιουργούνται με κατάλληλη μορφοποίηση τα αντικείμενα. Αυτή η σχεδίαση αποδείχθηκε ότι επιφέρει καλύτερα αποτελέσματα, αφού τα προγράμματα που δημιουργούνται είναι περισσότερο ευέλικτα, επαναχρησιμοποιήσιμα και περισσότερο φιλικά.

Οι θεμελιώδεις αρχές του αντικειμενοστραφούς προγραμματισμού, πηγάζουν από τον καθημερινό μας φυσικό κόσμο. Στηρίζονται στο γεγονός, ότι για να μπορέσει κάποιος να κατανοήσει άγνωστες σε αυτόν έννοιες, θα πρέπει να καθοδηγηθεί μέσω της προσομοίωσης των άγνωστων αυτών εννοιών αντιστοιχίζοντας αυτές σε πρακτικές γνώσεις και εικόνες από το πε-



Μεταξύ των πρώτων αντικειμενοστραφών γλωσσών προγραμματισμού που εμφανίστηκαν είναι η Simula και η Smalltalk.

ριβάλλον του, τις οποίες γνωρίζει και μπορεί πολύ εύκολα να χειριστεί. Σύμφωνα με την αντικειμενοστραφή θεωρεία ανάπτυξης εφαρμογών, η προσέγγιση κάθε προβλήματος πρέπει να γίνεται με φυσική ερμηνεία και να μην στηρίζεται σε πολύπλοκα τεχνικά ζητήματα.

Σύγχρονα αντικειμενοστραφή προγραμματιστικά περιβάλλοντα είναι η Visual C++, η Java, η Visual Basic, το Delphi.

### 11.1.1. Αντικείμενα

Βασικά στοιχεία του αντικειμενοστραφούς προγραμματισμού αποτελούν τα **αντικείμενα** (objects). Αν ρίξουμε μια ματιά γύρω μας, θα παρατηρήσουμε ότι το περιβάλλον μας αποτελείται από αντικείμενα, τα οποία μπορούμε πολύ εύκολα να αντιληφθούμε και να χειριστούμε. Ως παραδείγματα αντικειμένων μπορούμε να αναφέρουμε ένα αυτοκίνητο, μια μπάλα καλαθοσφαίρισης, ένα παγκάκι, ένα πορτοκάλι και έναν άνθρωπο (σχήμα 11.1).



Σχ. 11.1. Αντικείμενα από το φυσικό περιβάλλον

Κάθε αντικείμενο είναι ευδιάκριτο και αυτόνομο, ενώ το σύνολο των χαρακτηριστικών του προσδιορίζουν με λεπτομέρεια τη φυσική του υπόσταση. Για παράδειγμα, μια μπάλα καλαθοσφαίρισης έχει σχήμα στρογγυλό, το υλικό κατασκευής της είναι το πλαστικό και το χρώμα της είναι πορτοκαλί. Ο άνθρωπος έχει επίσης χαρακτηριστικά όπως όνομα, βάρος, ημερομηνία γέννησης, χρώμα ματιών κ.λπ. Είναι εμφανές ότι τα χαρακτηριστικά ενός αντικειμένου καθορίζονται από τις τιμές των επιμέρους ιδιοτήτων τους. Για παράδειγμα, το χρώμα της μπάλας καλαθοσφαίρισης είναι πορτοκαλί, το σχήμα της είναι στρογγυλό, το χρώμα των ματιών του ανθρώπου είναι καστανό.

Αν παρατηρήσουμε λεπτομερέστερα ένα αντικείμενο, θα διαπιστώσουμε ότι περιέχει και κανόνες συμπεριφοράς, “γνωρίζει” δηλαδή πως πρέπει να αντιδράσει όταν μια ενέργεια ασκηθεί επάνω του. Αν ρίξουμε τη μπάλα καλαθοσφαίρισης από κάποιο ύψος, αυτή θα αναπηδήσει στο πάτωμα, αν την πιέσουμε θα αλλοιωθεί το σχήμα της (σχήμα 11.2).

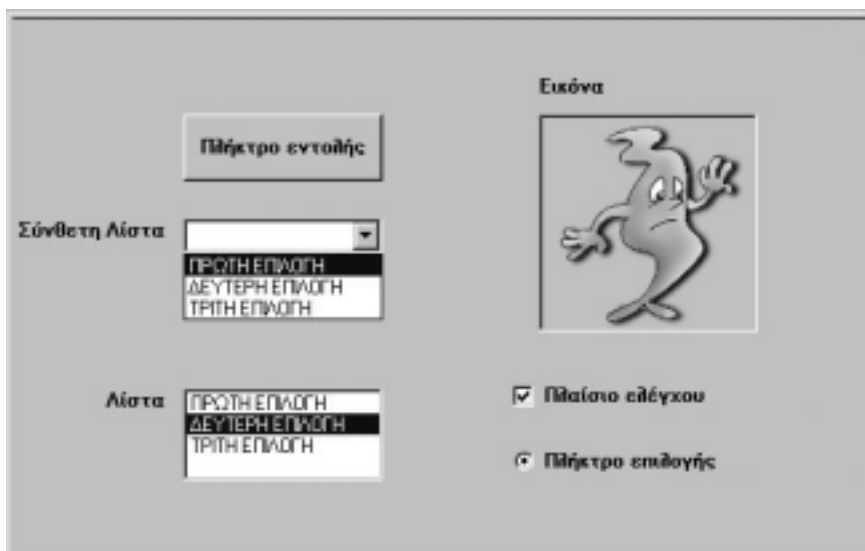


**Σχ. 11.2 . Συμπεριφορά αντικειμένου μπάλας καλαθοσφαίρισης**

Πως είναι όμως δυνατό να συνδυάσουμε τον καθημερινό μας κόσμο με τον κόσμο της Πληροφορικής; Μα φυσικά με την προσομοίωση των συστατικών μιας εφαρμογής ως φυσικά αντικείμενα.

Σε μια εφαρμογή, ένα αντικείμενο είναι ο ομαδοποιημένος συνδυασμός κώδικα και δεδομένων, τα οποία έχουμε τη δυνατότητα να τα χειριστούμε ενιαία. Από την μεριά ενός προγραμματιστή, τα δεδομένα (data) αποτελούν τα χαρακτηριστικά και οι ενέργειες (operations) καθορίζουν τη συμπεριφορά ενός αντικειμένου. Οι ενέργειες στον αντικειμενοστραφή προγραμματισμό αναφέρονται και ως **μέθοδοι** (methods).

Το “χτίσιμο” μιας αντικειμενοστραφούς εφαρμογής, επιτυγχάνεται με τη δημιουργία και το χειρισμό αντικειμένων. Σε μερικά αντικειμενοστραφή προγραμματιστικά περιβάλλοντα τα αντικείμενα της εφαρμογής μπορούν να δημιουργούνται είτε μέσω κώδικα είτε με τη βοήθεια κατάλληλων γραφικών εργαλείων, σε άλλα πάλι, δημιουργούνται μόνο μέσω κώδικα. Τα αντικείμενα μιας εφαρμογής μπορεί να είναι είτε γραφικά, είτε όχι. Μερικά γραφικά αντικείμενα που υποστηρίζουν πολλά αντικειμενοστραφή προγραμματιστικά περιβάλλοντα, όπως μια φόρμα, ένα πλήκτρο εντολής, ένα πλαίσιο λίστας, ένα πλαίσιο σύνθετης λίστας, ένα πλαίσιο ελέγχου, ένα πλήκτρο εντολής και μια εικόνα φαίνονται στο σχήμα 11.3.



Σχ. 11.3 . Γραφικά αντικείμενα μιας εφαρμογής

Πολλά από τα σύγχρονα προγραμματιστικά αντικειμενοστραφή περιβάλλοντα, προσφέρουν ένα αριθμό προκαθορισμένων αντικειμένων, τα οποία μπορούν να αξιοποιηθούν μέσα στην εφαρμογή. Φυσικά δεν περιορίζουν τον προγραμματιστή στο χειρισμό αυτών και μόνο των προκαθορισμένων αντικειμένων, αλλά του επιτρέπουν, μέσω εντολών προγράμματος, το σχεδιασμό νέων αντικειμένων προσαρμοσμένων στις δικές του ανάγκες.

Σε μια εφαρμογή μπορούμε να συμπεριλάβουμε και μη γραφικά αντικείμενα. Αυτό σημαίνει ότι το αντικείμενο υπάρχει, αλλά δεν έχει εξωτερικά χαρακτηριστικά. Τέτοια ειδικά αντικείμενα σε μια εφαρμογή μπορεί να προσφέρονται έτοιμα από το περιβάλλον ανάπτυξης, αλλά μπορεί και ο προγραμματιστής να δημιουργήσει και καινούργια αντικείμενα, μέσα από τμήματα κώδικα.

### 11.1.2. Κλάσεις

Όπως ήδη αναφέραμε ένα αντικείμενο είναι αυτόνομο, έχει δηλαδή τη δική του φυσική υπόσταση και ταυτότητα. Αυτό πρακτικά σημαίνει, ότι ακόμη και αν δύο αντικείμενα έχουν ακριβώς τις ίδιες τιμές στις ιδιότητές τους εξακολουθούν να παραμένουν δύο ανεξάρτητα αντικείμενα. Για παράδειγμα δύο μπάλες καλαθοσφαίρισης έχουν τα ίδια χαρακτηριστικά, κάθε φορά όμως σε ένα παιχνίδι χρησιμοποιείται μόνο μια από αυτές. Μπο-

ρούμε να χαρακτηρίσουμε την έννοια μπάλα καλαθοσφαίρισης ως το πρότυπο δημιουργίας ανεξάρτητων αντικειμένων μπάλας. Κάθε ανεξάρτητο αντικείμενο, αποτελεί ένα στιγμιότυπο (instance) του γενικού τύπου μπάλα καλαθοσφαίρισης. Ο γενικός τύπος ενός αντικειμένου καλείται **κλάση** (class) και καθορίζει τις αρχικές ιδιότητες και τη συμπεριφορά κάθε αντικείμενου που προέρχεται από αυτή.

Σε ένα αντικειμενοστραφές προγραμματιστικό περιβάλλον η υποστήριξη κλάσεων αποτελεί κυρίαρχο στοιχείο. Η κλάση είναι η στατική περιγραφή ενός συνόλου αντικειμένων. Όλα τα αντικείμενα δημιουργούνται ως ακριβή αντίγραφα της κλάσης τους. Για παράδειγμα, διαφορετικά αντικείμενα εικόνας, αποτελούν στιγμιότυπα της κλάσης εικόνα.

### 11.1.3. Ιδιότητες

Κατά τη δημιουργία τους τα διαφορετικά αυτά αντικείμενα έχουν τις ίδιες ιδιότητες και αντιδρούν με τον ίδιο τρόπο σε όποιο μήνυμα ανιχνευτεί από το περιβάλλον τους. Καθένα από τα αντικείμενα εικόνας κληρονομεί τις αρχικές ιδιότητες και την συμπεριφορά του από την κλάση εικόνα. Ωστόσο ο προγραμματιστής έχει τη δυνατότητα, μετατρέποντας τις τιμές των **ιδιοτήτων** τους (attributes ή properties), να διαμορφώσει διαφορετικά την εξωτερική εμφάνιση των αντικειμένων. Μπορεί επίσης με τη συγγραφή των κατάλληλων εντολών κώδικα να αντιστοιχίσει διαφορετικές εργασίες σε καθένα από αυτά τα αντικείμενα εικόνας.

Όπως ακριβώς είναι δυνατόν στο φυσικό μας κόσμο να μετατρέψουμε τα χαρακτηριστικά ενός αντικειμένου για παράδειγμα να αλλάξουμε το χρώμα ενός ποδηλάτου, ή τη θέση ενός φωτιστικού, έτσι και σε μια αντικειμενοστραφή εφαρμογή έχουμε τη δυνατότητα ή να αλλάξουμε το χρώμα, τη θέση, το μέγεθος ή όποιο άλλο χαρακτηριστικό του αντικείμενου επιθυμούμε.

Σε ένα γραφικό αντικειμενοστραφές περιβάλλον ανάπτυξης εφαρμογών, οι ιδιότητες των αντικειμένων είναι δυνατό να πάρουν αρχικές τιμές κατά το σχεδιασμό, αλλά τις περισσότερες φορές έχουμε τη δυνατότητα να τις μετατρέψουμε και κατά την εκτέλεση της εφαρμογής.

Κατά το σχεδιασμό μιας εφαρμογής πολλά από τα σύγχρονα περιβάλλοντα ανάπτυξης εφαρμογών μας παρέχουν κατάλληλα εργαλεία, όπως **παράθυρο ιδιοτήτων** (properties window), που εμφανίζουν τη τιμή κάθε ιδιότητας ενός αντικειμένου και μας επιτρέπουν τη μετατροπή τους.

Όταν θέλουμε να μετατρέψουμε τη τιμή μιας ιδιότητας κατά την εκτέλεση μιας εφαρμογής, επειδή δεν έχουμε πρόσβαση στα εργαλεία που μας



*Στη Smalltalk, αντικειμενοστραφή γλώσσα προγραμματισμού, καθετί, συμπεριλαμβανομένων των κλάσεων, θεωρείται αντικείμενο. Η κλάση εκλαμβάνεται σαν ένα στιγμιότυπο μιας κλάσης ανωτέρου επιπέδου που καλείται μετακλάση (metaclass).*

## Χαρακτηριστικά αντικειμενοστραφών περιβαλλόντων ανάπτυξης εφαρμογών

Το βασικότερο χαρακτηριστικό ενός αντικειμενοστραφούς περιβάλλοντος ανάπτυξης εφαρμογών είναι, όπως ήδη προαναφέραμε, η **κλάση**. Ένα πραγματικά αντικειμενοστραφές περιβάλλον ανάπτυξης εφαρμογών πρέπει να διαθέτει ακόμη τα παρακάτω χαρακτηριστικά :

**Υποστήριξη αφηρημένων τύπων (Data abstraction).** Κάθε αντικείμενο περιγράφεται ως υλοποίηση αφηρημένων τύπων. Κάθε αφηρημένος τύπος έχει ως δομικά του στοιχεία δεδομένα και συγκεκριμένους κανόνες συμπεριφοράς. Ένα αντικείμενο περιγράφεται από συγκεκριμένες ιδιότητες και κατά την εκτέλεση της εφαρμογής λαμβάνει μόνο τα μηνύματα που το αφορούν αγνοώντας οτιδήποτε άλλο συμβαίνει στο περιβάλλον του. Αυτό εξασφαλίζει ότι κάθε τμήμα του προγράμματος επηρεάζει μόνο τα αντικείμενα που το αφορούν και έχουμε πλήρη έλεγχο της εφαρμογής.

**Ενθυλάκωση (Encapsulation).** Όπως έχουμε ήδη αναφέρει, σε μια αντικειμενοστραφή εφαρμογή κάθε αντικείμενο αποτελεί ξεχωριστή οντότητα και περιέχει ενσωματωμένα τα χαρακτηριστικά (δεδομένα) και τους κανόνες συμπεριφοράς του (μεθόδους). Η δυνατότητα ενός αντικειμένου να συνδυάζει εσωτερικά τα δεδομένα και τις μεθόδους χειρισμού του καλείται ενθυλάκωση. Την ενθυλάκωση μπορούμε να την παρομοιάσουμε σαν ένα κέλυφος που υπάρχει γύρω από κάθε αντικείμενο και διαχωρίζει τον εσωτερικό από τον εξωτερικό του κόσμο. Με την ενθυλάκωση ο χειρισμός κάθε αντικειμένου είναι φυσικά πιο εύχρηστος και ασφαλής, γιατί τα περιεχόμενά του προστατεύονται και είναι δυνατό να μεταβληθούν μόνο με την αλλαγή των τιμών των ιδιοτήτων του ή με την δράση των μεθόδων που υποστηρίζει.

**Κληρονομικότητα (Inheritance).** Χαρακτηριστικά και μέθοδοι μπορούν να είναι κοινά σε διαφορετικές κλάσεις που είναι ιεραρχικά συνδεδεμένες. Μια κλάση μπορεί να περιγραφεί γενικά και στη συνέχεια μέσω αυτής της κλάσης να οριστούν υποκλάσεις (subclasses) αντικειμένων. Η κλάση απόγονος (υποκλάση) κληρονομεί και μπορεί να χρησιμοποιήσει όλα τα δεδομένα και τις μεθόδους που περιέχει η κλάση πρόγονος. Για παράδειγμα αν περιγράψουμε μια γενική κλάση γεωμετρικό σχήμα, μπορούμε στη συνέχεια να δημιουργήσουμε από αυτή υποκλάσεις αντικειμένων τριγώνου, τετραγώνου, κ.λπ. Η δυνατότητα δημιουργίας ιεραρχιών αντικειμένων καλείται κληρονομικότητα.



**Πολυμορφισμός (Polymorphism).** Με το χαρακτηριστικό του πολυμορφισμού παρέχεται η δυνατότητα στο ίδιο αντικείμενο να αναφέρεται, στο επίπεδο εκτέλεσης της εφαρμογής, σε διαφορετικές κλάσεις, να επιδρά διαφορετικά σε διαφορετικά αντικείμενα. Δύο ή περισσότερες κλάσεις αντικειμένων μπορούν να υποστηρίξουν συμπεριφορές με κοινό όνομα και ίδιο βασικό σκοπό αλλά με διαφορετική εφαρμογή. Για παράδειγμα σε ένα αυτοκίνητο τα πεντάλ του γκαζιού και του φρένου υποστηρίζουν τη μέθοδο πάτησε. Με τη μέθοδο πάτησε ο οδηγός δίνει μια εντολή για την εκτέλεση μιας εργασίας, αλλά η εφαρμογή της μεθόδου είναι διαφορετική σε κάθε πεντάλ. Όταν ο οδηγός πατήσει το γκάζι, το αυτοκίνητο αναπτύσσει ταχύτητα, ενώ το πάτημα του φρένου το επιβραδύνει. Η ίδια συμπεριφορά του οδηγού επιφέρει διαφορετικό αποτέλεσμα. Με τη χρήση του πολυμορφισμού ο προγραμματιστής απαλλάσσεται από τη σύνταξη και την επανάληψη πολύπλοκων δομών ελέγχου μέσα στην εφαρμογή.

παρέχει το περιβάλλον ανάπτυξης, πρέπει να επέμβουμε δυναμικά με τη χρήση εντολών κώδικα. Στα περισσότερα αντικειμενοστραφή προγραμματιστικά περιβάλλοντα, όπως η Visual C++ ή η Visual Basic, μια εντολή απόδοσης τιμής σε ιδιότητα έχει τη μορφή :

Αντικείμενο.Ιδιότητα = Τιμή

Στην παραπάνω εντολή ο όρος Αντικείμενο αντιπροσωπεύεται από μια ιδιότητα η οποία χαρακτηρίζει μοναδικά το επιθυμητό αντικείμενο. Συνήθως αυτή η ιδιότητα είναι το Όνομα του αντικειμένου.

#### 11.1.4. Μέθοδοι

Σε πολλά αντικειμενοστραφή προγραμματιστικά περιβάλλοντα, όπως η Smalltalk και η Visual Basic, ένα αντικείμενο είναι δυνατόν εκτός από ιδιότητες να περιέχει και μεθόδους (methods). Μια μέθοδος είναι η εφαρμογή μιας ενέργειας επάνω σε ένα αντικείμενο. Οι μέθοδοι είναι εσωτερικά στοιχεία του αντικειμένου όπως και οι ιδιότητες.

Όλα τα αντικείμενα μιας κλάσης υποστηρίζουν τις ίδιες μεθόδους. Το αποτέλεσμα μιας μεθόδου επάνω σε ένα αντικείμενο συνήθως επηρεάζει κάποιες από τις τιμές των ιδιοτήτων του. Οι μέθοδοι μπορούν να ενεργήσουν σε ένα αντικείμενο μόνο κατά την εκτέλεση της εφαρμογής, άρα μέσα από εντολές κώδικα. Η σύνταξη μιας εντολής εκτέλεσης μεθόδου είναι :

Αντικείμενο.Μέθοδος

## 11.2. Οδηγούμενος από γεγονότα προγραμματισμός

Ο **οδηγούμενος από γεγονότα προγραμματισμός** (event-driven programming) είναι μια μεθοδολογία προγραμματισμού που μας επιτρέπει, σε συνδυασμό με τον αντικειμενοστραφή προγραμματισμό, να εκμεταλλευτούμε τα πλεονεκτήματα των σύγχρονων περιβαλλόντων εργασίας.



*Ένα εξωτερικό μήνυμα είναι δυνατό να προκαλέσει περισσότερα του ενός γεγονότων στην εφαρμογή.*

Για να μπορέσουμε να κατανοήσουμε καλύτερα τα πλεονεκτήματα του οδηγούμενου από γεγονότα προγραμματισμού, θα κάνουμε πρώτα μια σύγκρισή του με το δομημένο προγραμματισμό, που αποτελεί την παραδοσιακή μορφή προγραμματισμού.

Σε μια εφαρμογή που έχει αναπτυχθεί με τη φιλοσοφία του δομημένου προγραμματισμού, η εκτέλεσή της ξεκινά από την αρχική εντολή του προγράμματος και η ροή εκτέλεσής της είναι καθορισμένη από τις διαδικασίες και τις συναρτήσεις που περιλαμβάνει το πρόγραμμα. Σύμφωνα με τα παραπάνω το δομικό στοιχείο του προγράμματος αποτελούν οι διαδικασίες και οι συναρτήσεις. Σε όλη τη διάρκεια της εκτέλεσης της εφαρμογής, το πρόγραμμα διατηρεί τον έλεγχό της, ενώ στο χρήστη έχει ανατεθεί δευτερεύων ρόλος και απλά πληκτρολογεί κάποια δεδομένα, όταν το πρόγραμμα το απαιτεί.

Αντίθετα με τον οδηγούμενο από γεγονότα προγραμματισμό, ο χρήστης με την έναρξη της εκτέλεσης της εφαρμογής αποκτά τον έλεγχό της και αποφασίζει, χρησιμοποιώντας το πληκτρολόγιο ή το ποντίκι, ποιο τμήμα του προγράμματος θα εκτελεστεί. Κάθε ενέργεια του χρήστη δημιουργεί μηνύματα με τη μορφή γεγονότων, τα οποία αντιλαμβάνεται το πρόγραμμα και ανταποκρίνεται σε αυτά.



**Γεγονός** (event) είναι μια ενέργεια που αναγνωρίζεται από την εφαρμογή, και συγκεκριμένα από τα αντικείμενά της, και την αναγκάζει να ανταποκριθεί. Η πρόκληση ενός γεγονότος μπορεί να γίνει είτε από το χρήστη με τη χρήση του πληκτρολογίου ή του ποντικιού, είτε από το σύστημα.

Παραδείγματα γεγονότων που προκαλούνται από το χρήστη είναι το πάτημα κάποιου πλήκτρου εντολής, η επιλογή ενός στοιχείου μίας λίστας, η πληκτρολόγηση κάποιων χαρακτήρων σε ένα πλαίσιο κειμένου, η επιλογή ενός αντικειμένου μενού επιλογών κ.λπ. Ως γεγονότα που προκαλούνται από το σύστημα μπορούμε να αναφέρουμε την αποστολή μηνυμάτων από το ρολόι του υπολογιστή ή από άλλες εφαρμογές και ο τερματισμός του συστήματος.

### 11.2.1. Διαδικασίες

Σε πολλά από τα σύγχρονα προγραμματιστικά περιβάλλοντα, ο τρόπος που ο προγραμματιστής καθορίζει την εκτέλεση του προγράμματος, είναι με συγγραφή των κατάλληλων εντολών κώδικα που ενεργοποιούν τα αντικείμενα της εφαρμογής και τα υποχρεώνουν να αντιδράσουν στα γεγονότα που προκαλούνται στο περιβάλλον τους. Ο κώδικας για κάθε γεγονός που πρόκειται να συμβεί, σε άλλα προγραμματιστικά περιβάλλοντα γράφεται με τη μορφή ανεξάρτητων τμημάτων κώδικα του προγράμματος, που καλούνται **διαδικασίες γεγονότων** (event procedures), ενώ σε κάποια άλλα προγραμματιστικά περιβάλλοντα γράφεται ενιαία.

Κάθε αντικείμενο της εφαρμογής αναγνωρίζει ένα συγκεκριμένο αριθμό γεγονότων. Κατά συνέπεια, για εκείνη την κατηγορία των προγραμματιστικών περιβαλλόντων που υποστηρίζουν διαδικασίες γεγονότων, αυτές καθορίζονται από την ενέργεια και το αντικείμενο που τις υποστηρίζει.

Η σύνταξη μιας διαδικασίας γεγονότος είναι :

**ΔΙΑΔΙΚΑΣΙΑ** Αντικείμενο\_Γεγονός

...

Εντολές κώδικα

...

**ΤΕΛΟΣ ΔΙΑΔΙΚΑΣΙΑΣ**

Το μεγάλο πλεονέκτημα των σύγχρονων περιβαλλόντων προγραμματισμού, είναι ότι κάθε αντικείμενο αναγνωρίζει αυτόματα τα γεγονότα που το αφορούν και ξέρει πως θα αντιδράσει σε αυτά. Ο προγραμματιστής σε κανένα σημείο του προγράμματος δεν χρειάζεται να ασχοληθεί με τον τρόπο που το αντικείμενο θα ανιχνεύσει από το περιβάλλον το γεγονός, ούτε με την κλήση του αντίστοιχου κατάλληλου κώδικα, η οποία θα γίνει αυτόματα από το αντικείμενο.

Σε μια εφαρμογή οδηγούμενη από τα γεγονότα δεν συμπεριλαμβάνουμε μόνο διαδικασίες γεγονότων, αλλά έχουμε τη δυνατότητα να συμπεριλάβουμε και **γενικές διαδικασίες** ή **συναρτήσεις** που καλούνται μέσα από διαδικασίες γεγονότων.

Για παράδειγμα αν θέλουμε σε πολλά σημεία ενός προγράμματος να γίνει ο ίδιος υπολογισμός, αποφεύγουμε να χρησιμοποιούμε τις ίδιες εντολές προγράμματος σε κάθε διαδικασία γεγονότος, αλλά δημιουργούμε μια γενική διαδικασία Υπολογισμός εφαρμόζοντας τις αρχές του δομημένου προγραμματισμού :



*Η έννοια της διαδικασίας δεν απαντάται με την ίδια ονομασία σε όλα τα σύγχρονα προγραμματιστικά περιβάλλοντα. Για παράδειγμα, στη Visual C++ οι διαδικασίες ονομάζονται συναρτήσεις (functions).*

**ΔΙΑΔΙΚΑΣΙΑ** Υπολογισμός

...

Εντολές κώδικα

...

**ΤΕΛΟΣ ΔΙΑΔΙΚΑΣΙΑΣ**

Στη συνέχεια μέσα από κάθε διαδικασία γεγονός στο σημείο που θέλουμε να εκτελεστεί ο συγκεκριμένος υπολογισμός καλούμε τη γενική διαδικασία :

**ΔΙΑΔΙΚΑΣΙΑ** Αντικείμενο\_Γεγονός

...

Κάλεσε Υπολογισμός ' *Κλήση της γενικής διαδικασίας*

...

**ΤΕΛΟΣ ΔΙΑΔΙΚΑΣΙΑΣ**

Παρατηρούμε λοιπόν ότι για την ανάπτυξη μιας σύγχρονης εφαρμογής πρέπει να εφαρμόσουμε και τις τρεις τεχνικές προγραμματισμού που έχουμε αναπτύξει. Σε μια σύγχρονη εφαρμογή συνδυάζουμε τον **τμηματικό** (modular) προγραμματισμό, αξιοποιώντας τις τεχνικές του αντικειμενοστραφούς και του οδηγούμενου από γεγονότα προγραμματισμού, αλλά εξακολουθούμε να χρησιμοποιούμε και το **δομημένο** (structural) προγραμματισμό, αφού τα περιεχόμενα των διαδικασιών γεγονότων είναι δομημένες εντολές κώδικα. Η εφαρμογή του δομημένου προγραμματισμού φυσικά δε διαφοροποιείται και ισχύουν όσα γνωρίζουμε για τις μεταβλητές, τις δομές ελέγχου, τους πίνακες κ.ά. Τέλος μέσα σε ένα πρόγραμμα έχουμε τη δυνατότητα χρησιμοποίησης γενικών διαδικασιών και συναρτήσεων που επίσης στηρίζονται στις αρχές του δομημένου προγραμματισμού.

### 11.2.2. Ροή εκτέλεσης εφαρμογής

Αν προσπαθήσουμε να αναλύσουμε τη ροή εκτέλεσης μιας εφαρμογής που ακολουθεί την τεχνική του οδηγούμενου από γεγονότα προγραμματισμού και υποστηρίζει διαδικασίες γεγονότων, μπορούμε να τη παρουσιάσουμε με τα παρακάτω βήματα :

1. Ο χρήστης ξεκινά την εκτέλεση της εφαρμογής.
2. Τα αντικείμενα της εφαρμογής περνούν σε κατάσταση αναμονής γεγονότων.
3. Μόλις ένα αντικείμενο αναγνωρίσει μέσα στο περιβάλλον του κάποιο γεγονός που το αφορά, καλεί την αντίστοιχη διαδικασία γεγονότος.

4. Αν στη διαδικασία γεγονότος έχουμε συμπεριλάβει εντολές κώδικα, εκτελούνται διαφορετικά η εφαρμογή αγνοεί το γεγονός.
5. Τα αντικείμενα της εφαρμογής περνούν πάλι σε κατάσταση αναμονής γεγονότων (Βήμα 2).

Ο προγραμματιστής δεν είναι απαραίτητο να συμπεριλάβει εντολές κώδικα σε κάθε διαδικασία γεγονότος που υποστηρίζουν τα αντικείμενα της εφαρμογής. Απλά επιλέγει αυτές για τις οποίες τον ενδιαφέρει το πρόγραμμα να αντιδρά. Αν μια διαδικασία γεγονότος δεν περιέχει καμία εντολή κώδικα, όπως είναι φυσικό αυτό το γεγονός αγνοείται τελείως από την εφαρμογή, σαν να μην συνέβη ποτέ.



*Ένα γεγονός είναι δυνατό να προκληθεί και μέσα από εντολές κώδικα. Η επιτυχία μιας οδηγούμενης από γεγονότα εφαρμογής εξαρτάται από τον τρόπο που ο προγραμματιστής χρησιμοποιεί τα γεγονότα. Η πρόκληση γεγονότων μέσα από κώδικα αποτελεί μια εξυπνη τεχνική.*

### 11.3. Υλοποίηση εφαρμογών σε σύγχρονο προγραμματιστικό περιβάλλον

Στο παρελθόν ένας προγραμματιστής κατά το σχεδιασμό της εφαρμογής έδινε ιδιαίτερη έμφαση στην ορθή υλοποίηση του αλγορίθμου του προγράμματος και δεν ασχολούνταν σχεδόν καθόλου με τον τρόπο επικοινωνίας του χρήστη με την εφαρμογή, αφού άλλωστε το περιβάλλον εργασίας δεν του παρείχε και ιδιαίτερες δυνατότητες επάνω σε αυτό το τμήμα του προγράμματος.

Αντιθέτως σε ένα σύγχρονο περιβάλλον εργασίας δίνεται ιδιαίτερη έμφαση στον τρόπο που επικοινωνεί ο χρήστης με το σύστημα. Κατά συνέπεια ο προγραμματιστής είναι υποχρεωμένος να ασχοληθεί ιδιαίτερα με τον τρόπο επικοινωνίας του χρήστη με την εφαρμογή, τη **διεπαφή χρήστη** (user interface). Κατά το σχεδιασμό της διεπαφής χρήστη, ο προγραμματιστής πρέπει να αντιλαμβάνεται την εφαρμογή από την πλευρά του χρήστη. Ένας προγραμματιστής για να δημιουργήσει μια σύγχρονη, φιλική και ευέλικτη εφαρμογή θα πρέπει να δώσει οπτική μορφή στο περιβάλλον εργασίας που παρέχει στο χρήστη. Τα σύγχρονα προγραμματιστικά περιβάλλοντα παρέχουν στον προγραμματιστή όλα εκείνα τα εργαλεία που χρειάζεται για να δημιουργήσει ένα τέτοιο περιβάλλον. Και φυσικά τα εργαλεία αυτά είναι τα αντικείμενα.

Στον κόσμο των αντικειμένων, ένας προγραμματιστής είναι φυσικό να ξεκινά το σχεδιασμό της εφαρμογής από τα αντικείμενα που πρόκειται να χρησιμοποιήσει. Η σωστή επιλογή των αντικειμένων εξασφαλίζει σε μεγάλο βαθμό και την επιτυχία εκτέλεσης της εφαρμογής.

Μετά από την επιλογή των κατάλληλων αντικειμένων, ο προγραμματιστής γνωρίζει ποιες θα είναι οι πιθανές ενέργειες του χρήστη ή του συστήματος επί της εφαρμογής. Επομένως είναι ευδιάκριτο ποια είναι τα γεγονότα που είναι δυνατό να ανιχνεύσουν τα επιλεγμένα αντικείμενα. Από τα πιθανά γεγονότα που θα συμβούν στο περιβάλλον της εφαρμογής, ο προγραμματιστής επιλέγει αυτά που θέλει να αξιοποιήσει μέσα από την εφαρμογή του και γράφει τον κατάλληλο κώδικα.

Συνοψίζοντας τα παραπάνω μπορούμε να περιγράψουμε την ανάπτυξη μιας εφαρμογής σε ένα σύγχρονο προγραμματιστικό περιβάλλον σαν διαδικασία τριών βημάτων :

1. Σχεδιασμός του τρόπου επικοινωνίας χρήστη-εφαρμογής, επιλέγοντας τα κατάλληλα αντικείμενα.
2. Καθορισμός της αρχικής συμπεριφοράς των αντικειμένων μέσω των ιδιοτήτων που υποστηρίζουν.
3. Δημιουργία και εκσφαλμάτωση κώδικα.

### Παράδειγμα

Για να γίνει κατανοητός ο τρόπος υλοποίησης μιας εφαρμογής σε ένα σύγχρονο προγραμματιστικό περιβάλλον, θα τον περιγράψουμε, θεωρώντας ένα υποθετικό προγραμματιστικό περιβάλλον, παρουσιάζοντας ένα απλό παράδειγμα και περιγράφοντας αναλυτικά όλα τα βήματα σχεδιασμού.

Το πρόβλημα συνίσταται στη δημιουργία μιας εφαρμογής που θα διαθέτει πλήκτρα, το πάτημα των οποίων θα έχει σαν αποτέλεσμα να εμφανίζεται καθένα από τα τρία παραπάνω περιγραφόμενα βήματα σχεδιασμού μιας εφαρμογής σε ένα σύγχρονο προγραμματιστικό περιβάλλον.

### Δημιουργία της διεπαφής χρήστη

Το πρώτο στοιχείο που πρέπει να απασχολήσει ένα προγραμματιστή, είναι ο τρόπος επικοινωνίας του χρήστη με την εφαρμογή. Για το σχεδιασμό λοιπόν της διεπαφής χρήστη πρέπει να επιλέξουμε τα κατάλληλα αντικείμενα. Στο παράδειγμά μας θα χρησιμοποιήσουμε :

- ⇒ ένα αντικείμενο **φόρμα** το οποίο είναι το παράθυρο μέσα στο οποίο θα εκτελείται η εφαρμογή,
- ⇒ αντικείμενα **πλήκτρα εντολής** με τα οποία ο χρήστης θα καθοδηγεί την εκτέλεσή της ,

⇒ αντικείμενα **ετικέτες** που θα εμφανίζουν τις πληροφορίες.

Αρχικά δημιουργούμε μια φόρμα και στη συνέχεια τοποθετούμε επάνω της τρία πλήκτρα εντολής. Τα τρία πλήκτρα εντολής τα επιλέγουμε από την εργαλειοθήκη (toolbox) που μας παρέχει το υποθετικό περιβάλλον προγραμματισμού στο οποίο βρισκόμαστε.

Τα τρία πλήκτρα εντολής θα δίνουν τη δυνατότητα στο χρήστη να επιλέγει το βήμα της περιγραφής ανάπτυξης μιας εφαρμογής σε ένα σύγχρονο προγραμματιστικό περιβάλλον, που θέλει να εμφανιστεί. Για να είναι τα πλήκτρα κατατοπιστικά θα εμφανίσουμε επάνω σε κάθε ένα τον αριθμό του βήματος που πρόκειται να παρουσιάζεται. Ακόμη πρέπει να αποδώσουμε σε καθένα από τα πλήκτρα ένα μοναδικό όνομα, με το οποίο θα επικοινωνούμε μαζί του μέσα από το κώδικα της εφαρμογής. Η μετατροπή αυτών των χαρακτηριστικών μπορεί να γίνει αλλάζοντας τις τιμές ιδιοτήτων των αντικειμένων.

Για το υποθετικό σύγχρονο προγραμματιστικό περιβάλλον που βρισκόμαστε, κάνουμε τις παρακάτω παραδοχές :

- ⇒ η απόδοση τιμών στις ιδιότητες των αντικειμένων γίνεται με χρήση κατάλληλου εργαλείου (παραθύρο ιδιοτήτων) που διαθέτει,
- ⇒ η απόδοση προγραμματιστικού ονόματος σε ένα αντικείμενο πλήκτρο εντολής γίνεται μέσω της ιδιότητάς του **Όνομα**,
- ⇒ η απόδοση λεκτικού πάνω σε ένα αντικείμενο πλήκτρο εντολής γίνεται μέσω της ιδιότητάς του **Τίτλος**,
- ⇒ η απόδοση γραφικού πάνω σε ένα αντικείμενο πλήκτρο εντολής γίνεται μέσω της ιδιότητάς του **Εικόνα**,

Για τα τρία πλήκτρα εντολής πρέπει να αντιστοιχήσουμε τις παρακάτω τιμές στις προαναφερθείσες ιδιότητες :

Αντικείμενο	Ιδιότητα	Τιμή
Πλήκτρο εντολής 1	Όνομα	ΠλήκτροΕντολήςΒήμα1
	Τίτλος	1
Πλήκτρο εντολής 2	Όνομα	ΠλήκτροΕντολήςΒήμα2
	Τίτλος	2
Πλήκτρο εντολής 3	Όνομα	ΠλήκτροΕντολήςΒήμα3
	Τίτλος	3



Η τιμή στην ιδιότητα Όνομα, είναι σωστό να προσδιορίζει τη κλάση από την οποία προέρχεται το αντικείμενο και το ρόλο του μέσα στην εφαρμογή. Για παράδειγμα με την τιμή ΠλήκτροΕντολήςΒήμα1, γίνεται αντιληπτό ότι αναφερόμαστε σε ένα πλήκτρο εντολής, που εμφανίζει το πρώτο βήμα.

Στη συνέχεια ακολουθώντας την ίδια διαδικασία, τοποθετούμε τέσσερα αντικείμενα Ετικέτα επάνω στη φόρμα. Η πρώτη ετικέτα θα περιγράφει το σκοπό της εφαρμογής και οι τρεις επόμενες τα βήματα σχεδιασμού της εφαρμογής. Τέλος στην εφαρμογή θα τοποθετήσουμε ένα ακόμη πλήκτρο εντολής που θα του αποδώσουμε γραφική μορφή, και με το οποίο θα τερματίζεται η εκτέλεση της εφαρμογής.

Για τα εργαλεία Ετικέτα στο υποθετικό προγραμματιστικό περιβάλλον που βρισκόμαστε, κάνουμε τις εξής παραδοχές :

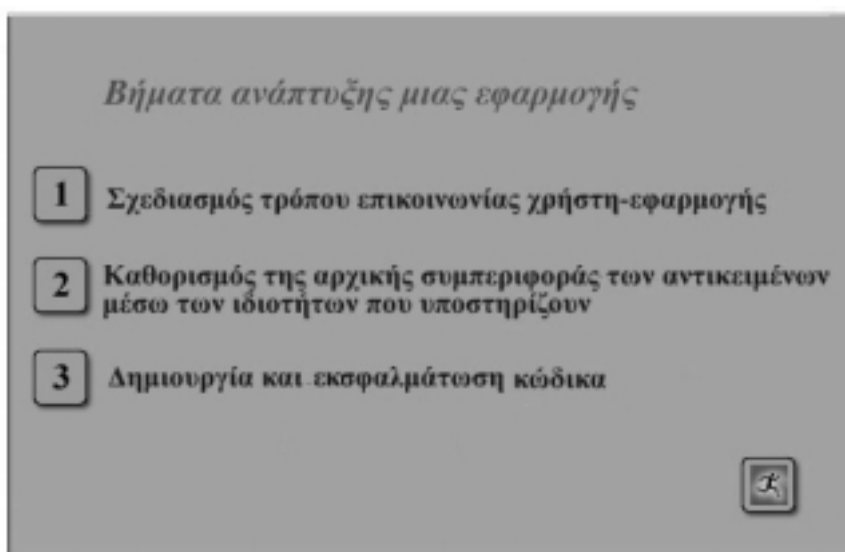
- ⇒ έχουν τη δυνατότητα να εμφανίζουν κείμενο που καταχωρείται κατά τη διάρκεια σχεδίασης της εφαρμογής, αλλά ο χρήστης δεν έχει τη δυνατότητα να το τροποποιήσει κατά τη διάρκεια εκτέλεσης της εφαρμογής,
- ⇒ η καταχώριση κειμένου σε ένα αντικείμενο ετικέτα γίνεται μέσω της ιδιότητάς της **Τίτλος**,
- ⇒ η απόδοση προγραμματιστικού ονόματος σε ένα αντικείμενο ετικέτα γίνεται μέσω της ιδιότητάς της **Όνομα**,
- ⇒ η εμφάνιση ή όχι του κειμένου που περιλαμβάνει ένα αντικείμενο ετικέτα εξαρτάται από την τιμή που έχει η ιδιότητά του **Ορατό** και η οποία δέχεται τις τιμές Αληθής ή Ψευδής.

Στα υπόλοιπα αντικείμενα της εφαρμογής θα αποδώσουμε τις παρακάτω τιμές :

Αντικείμενο	Ιδιότητα	Τιμή
Ετικέτα 0	Όνομα	ΕτικέταΤίτλος
	Ορατό	Αληθής
	Τίτλος	Βήματα ανάπτυξης μιας εφαρμογής
Ετικέτα 1	Όνομα	ΕτικέταΒήμα1
	Ορατό	Ψευδής
	Τίτλος	Σχεδιασμός του τρόπου επικοινωνίας χρήστη - εφαρμογής, επιλέγοντας τα κατάλληλα αντικείμενα
Ετικέτα 2	Όνομα	ΕτικέταΒήμα2
	Ορατό	Ψευδής
	Τίτλος	Καθορισμός της αρχικής συμπεριφοράς των αντικειμένων μέσω των ιδιοτήτων που τα χαρακτηρίζουν



Αντικείμενο	Ιδιότητα	Τιμή
Ετικέτα 3	Όνομα	ΕτικέταΒήμα3
	Ορατό	Ψευδής
	Τίτλος	Δημιουργία και εκσφαλμάτωση κώδικα
Πλήκτρο εντολής 4	Όνομα	ΠλήκτροΕντολήςΤέλος
	Εικόνα	(εικονίδιο εξόδου)



Σε κάθε παράθυρο μιας εφαρμογής πρέπει να δίνεται στο χρήστη ένας εμφανής τρόπος εξόδου από την εφαρμογή, όπως γίνεται στο παράδειγμά μας με το πλήκτρο εντολής ΠλήκτροΕντολήςΤέλος. Με αυτή τη τεχνική δημιουργούμε φιλικές εφαρμογές και δίνουμε τη δυνατότητα χειρισμού τους ακόμη και σε αρχάριους χρήστες.

Σχ. 11.4. Η διασύνδεση του χρήστη με την εφαρμογή στο παράδειγμά μας

Μετά από την αντιστοίχιση όλων των ιδιοτήτων έχουμε ολοκληρώσει τη σχεδίαση της διασύνδεσης του χρήστη με την εφαρμογή, έτσι όπως φαίνεται στο σχήμα 11.4.

Είναι χαρακτηριστικό ότι στο υποθετικό σύγχρονο προγραμματιστικό περιβάλλον που βρισκόμαστε, η υλοποίηση της επικοινωνίας του χρήστη με την εφαρμογή έχει ολοκληρωθεί χωρίς να έχουμε γράψει ούτε μια εντολή κώδικα. Απλά έχουμε επιλέξει τα αντικείμενα και αποδώσαμε τις κατάλληλες τιμές σε ορισμένες από τις ιδιοτητές τους. Παρόμοια όμως συμβαίνει και σε πολλά πραγματικά προγραμματιστικά περιβάλλοντα.

### Ο κώδικας

Το τελευταίο βήμα που πρέπει να εκτελέσουμε για να ολοκληρωθεί ο σχεδιασμός της εφαρμογής, είναι η προσθήκη των εντολών κώδικα. Μέσα από τις εντολές κώδικα θα εκτελούμε τις εργασίες της εφαρμογής.

Στο παράδειγμά μας όλες οι παρεμβάσεις στην εκτέλεση της εφαρμογής, από την πλευρά του χρήστη, γίνονται μέσω των πλήκτρων εντολής.

Στο υποθετικό σύγχρονο προγραμματιστικό περιβάλλον που βρισκόμαστε κάνουμε την εξής παραδοχή αναφορικά με τα προκαλούμενα γεγονότα από τις ενέργειες του χρήστη :

⇒ Το γεγονός που προκαλείται μόλις ο χρήστης πατήσει το αριστερό πλήκτρο του ποντικιού, όταν ο δείκτης του ποντικιού βρίσκεται πάνω από κάποιο πλήκτρο εντολής, είναι το γεγονός **Κλικ**.

Επομένως ο κώδικας της εφαρμογής πρέπει να γραφτεί στις διαδικασίες γεγονότων Κλικ των αντικειμένων πλήκτρα εντολής.

Το πάτημα κάθε πλήκτρου εντολής εμφάνισης των βημάτων, πρέπει κάθε φορά να εμφανίζει το μήνυμα που αντιστοιχεί σε αυτό και να αποκρύπτει τις υπόλοιπες ετικέτες βημάτων.

Ο κώδικας των τριών πλήκτρων εντολής λοιπόν θα είναι :

**ΔΙΑΔΙΚΑΣΙΑ** ΠλήκτροΕντολήςΒήμα1\_Κλικ()

ΕτικέταΒήμα1.Όρατο = Αληθής

ΕτικέταΒήμα2.Όρατο = Ψευδής

ΕτικέταΒήμα3.Όρατο = Ψευδής

**ΤΕΛΟΣ ΔΙΑΔΙΚΑΣΙΑΣ**

**ΔΙΑΔΙΚΑΣΙΑ** ΠλήκτροΕντολήςΒήμα2\_Κλικ()

ΕτικέταΒήμα1.Όρατο = Ψευδής

ΕτικέταΒήμα2.Όρατο = Αληθής

ΕτικέταΒήμα3.Όρατο = Ψευδής

**ΤΕΛΟΣ ΔΙΑΔΙΚΑΣΙΑΣ**

**ΔΙΑΔΙΚΑΣΙΑ** ΠλήκτροΕντολήςΒήμα3\_Κλικ()

ΕτικέταΒήμα1.Όρατο = Ψευδής

ΕτικέταΒήμα2.Όρατο = Ψευδής

ΕτικέταΒήμα3.Όρατο = Αληθής

**ΤΕΛΟΣ ΔΙΑΔΙΚΑΣΙΑΣ**

Τέλος στην διαδικασία γεγονότος Κλικ του πλήκτρου τερματισμού ΠλήκτροΕντολήςΤέλος της εφαρμογής πρέπει να συμπεριλάβουμε την εντολή με την οποία τερματίζεται η εκτέλεση της εφαρμογής :

**ΔΙΑΔΙΚΑΣΙΑ** ΠλήκτροΕντολήςΤέλος\_Κλικ()

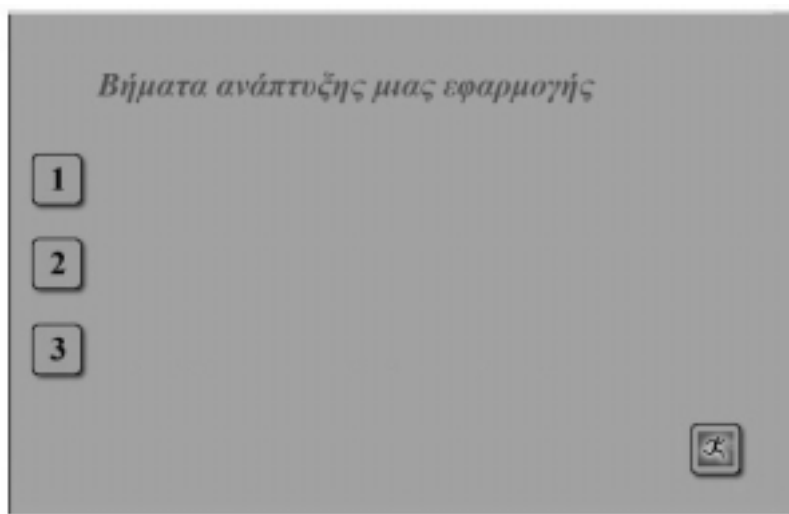
Εντολή τερματισμού εφαρμογής

**ΤΕΛΟΣ ΔΙΑΔΙΚΑΣΙΑΣ**

Στο παράδειγμά μας, χρησιμοποιήσαμε μόνο το γεγονός Κλικ, για λόγους απλότητας. Η εφαρμογή αν υλοποιούταν σε πραγματικό προγραμματιστικό περιβάλλον θα παρουσίαζε πολλά κοινά σημεία με αυτά που παρουσιάστηκαν στο υποθετικό. Θα είχε βέβαια και πολλά περιθώρια βελτίωσης και πολλούς εναλλακτικούς τρόπους προσέγγισης, αλλά σκοπός είναι η εισαγωγή στη φιλοσοφία και στη μεθοδολογία προγραμματισμού και όχι η επίδειξη δυνατοτήτων ενός σύγχρονου, έστω και υποθετικού, προγραμματιστικού περιβάλλοντος.

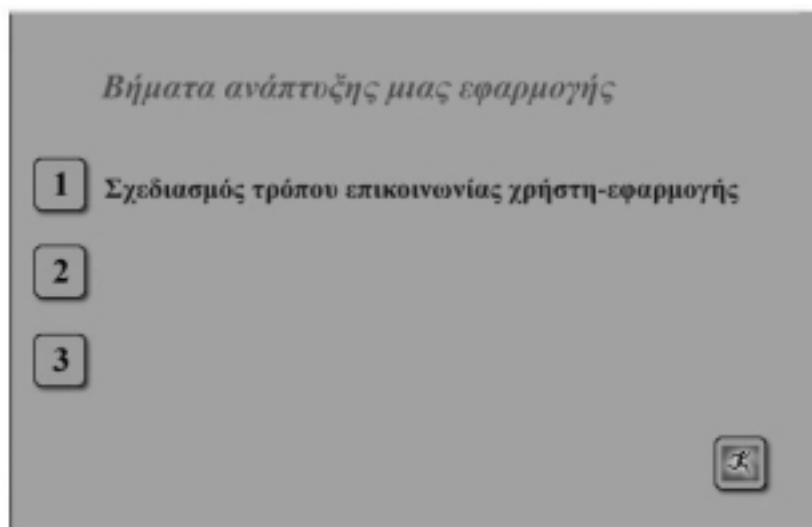
### Η εκτέλεση

Μόλις ολοκληρώσουμε την προσθήκη του κώδικα στις διαδικασίες γεγονότων που θέλουμε να αντιδρά η εφαρμογή, είμαστε σε θέση να εκκινήσουμε την εκτέλεσή της στο υποθετικό προγραμματιστικό περιβάλλον μας. Μόλις ξεκινήσει η εφαρμογή θα εμφανίζεται η αρχική της οθόνη (σχήμα 11.5).



Σχ. 11.5. Η πρώτη οθόνη του παραδείγματος σε μορφή εκτέλεσης

Στη συνέχεια ο χρήστης έχει τη δυνατότητα πατώντας κάποιο από τα τρία αριθμημένα πλήκτρα να εμφανίσει κάποιο από τα τρία βήματα σχεδιασμού της εφαρμογής (σχήμα 11.6).



Σχ. 11.6. Ένα από τα τρία βήματα εκτέλεσης της εφαρμογής

Χρησιμοποιώντας το πλήκτρο εντολής με το γραφικό, ο χρήστης έχει τη δυνατότητα κάθε στιγμή να διακόψει την εκτέλεση της εφαρμογής.

#### 11.4. Στοιχεία γραφικού προγραμματιστικού περιβάλλοντος

Η φιλικότητα και η ευελιξία μιας εφαρμογής καθορίζεται κυρίως από το τρόπο που επικοινωνεί με το χρήστη και συνεργάζεται με το περιβάλλον εργασίας. Το περιβάλλον εργασίας μιας εφαρμογής δεν πρέπει να είναι κουραστικό και δύσχρηστο. Αντίθετα πρέπει να είναι ευχάριστο, να καθοδηγεί και να διευκολύνει το χρήστη στην εργασία που θέλει να επιτελέσει.

Όλα τα σύγχρονα προγραμματιστικά εργαλεία, όπως η Visual C++, το Delphi ή η Visual Basic, μας προσφέρουν τη δυνατότητα να δίνουμε στις εφαρμογές μας τέτοια χαρακτηριστικά μέσα από τη χρησιμοποίηση των γραφικών. Με τον προγραμματισμό σε γραφικό προγραμματιστικό περιβάλλον, παρέχοντας ένα εύχρηστο περιβάλλον εργασίας στους χρήστες, αυξάνουμε τις δυνατότητές τους και μεγιστοποιούμε τα αποτελέσματα. Επιπλέον, μειώνεται δραστικά ο χρόνος και το κόστος εκμάθησης των χρηστών σε μια καινούργια εφαρμογή.

Ένα γραφικό προγραμματιστικό περιβάλλον μας δίνει ευελιξία και διαφορετικές δυνατότητες δημιουργίας γραφικού τρόπου επικοινωνίας του χρήστη με την εφαρμογή, αλλά ταυτόχρονα μας καθοδηγεί να δημιουργήσουμε γνώριμες εφαρμογές, αφού τα αντικείμενα, όπως πλήκτρα εντολής, λίστες, πλαίσια ελέγχου, πλήκτρα επιλογής, που χρησιμοποιούμε στις δικές μας εφαρμογές είναι ήδη γνωστά στους χρήστες από εφαρμογές γενικής χρήσης. Έτσι οι εφαρμογές μας, διατηρούν κάποια φιλοσοφία που είναι ήδη γνωστή στους χρήστες από άλλες εφαρμογές, όπως λογιστικά φύλλα, επεξεργαστές κειμένου ή βάσεις δεδομένων, που χρησιμοποιούν παρόμοια γραφικά αντικείμενα.

Ακόμα, ένα γραφικό προγραμματιστικό περιβάλλον προσφέρει στον προγραμματιστή τα κατάλληλα εργαλεία για γρήγορη ανάπτυξη εφαρμογών. Έτοιμα εργαλεία που παρέχονται από πολλά προγραμματιστικά περιβάλλοντα, όπως εργαλειοθήκες (toolboxes), εργαλεία εκσφαλμάτωσης (debugging tools), εύχρηστα παράθυρα συγγραφής κώδικα (code windows), γεννήτριες εκτυπωτικών αναφορών (report generators) συνθέτουν ένα περιβάλλον που διευκολύνει τον προγραμματιστή στο έργο του. Ακόμη διάφοροι οδηγοί εκτέλεσης εργασιών (wizards) τον βοηθούν στην εκτέλεση συχνών εργασιών, όπως τη δημιουργία φορμών ή εκτυπώσεων. Τέλος ένα σύγχρονο προγραμματιστικό εργαλείο, επιτρέπει στον προγραμματιστή να εκμεταλλευτεί τις ήδη υπάρχουσες εφαρμογές του συστήματος, χωρίς να χρειάζεται να δημιουργεί τμήματα κώδικα στην εφαρμογή του για την εκτέλεση εργασιών που ήδη εκτελεί σωστά κάποια άλλη εφαρμογή.

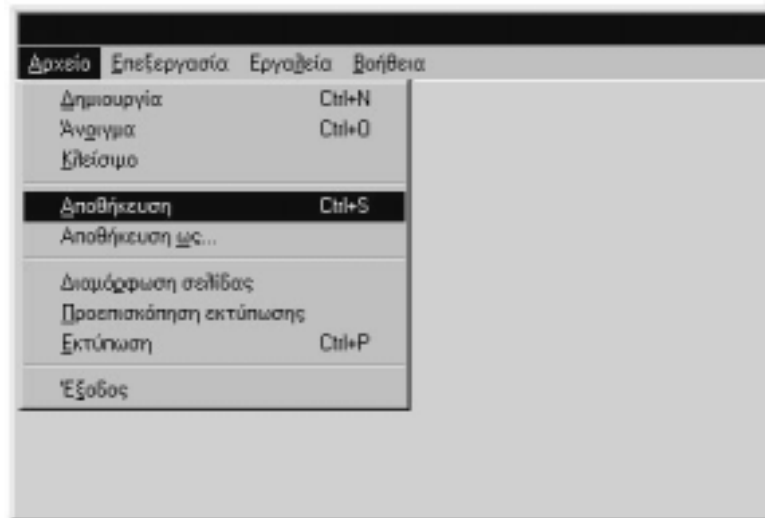
Το αποτέλεσμα εργασίας με ένα γραφικό προγραμματιστικό εργαλείο, είναι η γρήγορη υλοποίηση δυναμικών εφαρμογών που επικοινωνούν φιλικά με το χρήστη και συνεργάζονται αρμονικά με το περιβάλλον και τις υπόλοιπες εφαρμογές.

Βασικά χαρακτηριστικά στοιχεία των γραφικών προγραμματιστικών περιβαλλόντων είναι τα μενού επιλογών και τα πλαίσια διαλόγου.

### 11.4.1 Μενού επιλογών

Τα **μενού επιλογών** (menus) μας επιτρέπουν, κατά την εκτέλεση μιας εφαρμογής, τη γρήγορη πρόσβαση σε μια εύκολα προσπελάσιμη λίστα επιλογών. Οι περισσότεροι χρήστες γνωρίζουν ήδη τις δυνατότητες των μενού επιλογών αφού τα έχουν ήδη χρησιμοποιήσει μέσα από γνωστές εφαρμογές γενικής χρήσης.

Τα κλασικά μενού επιλογών εμφανίζονται στη **γραμμή μενού** (menu bar) ενός παραθύρου (σχήμα 11.7).

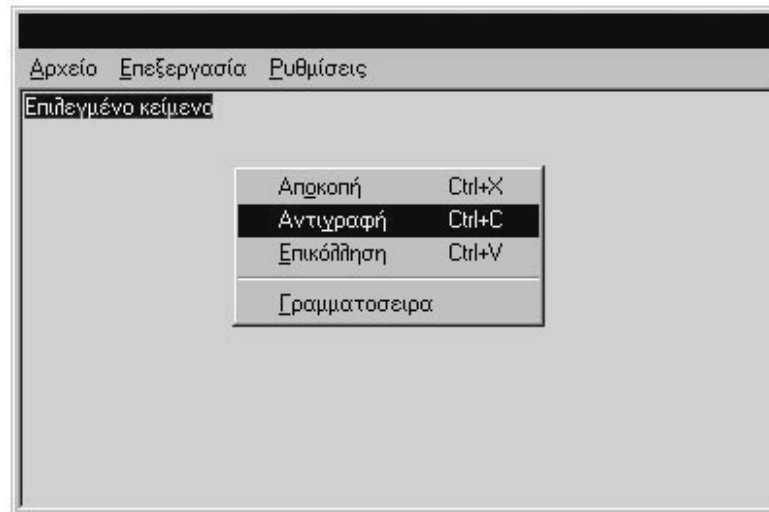


Σχ. 11.7. Ένα κλασικό μενού επιλογών



Η υλοποίηση ενός μενού επιλογών σε αρκετά από τα σύγχρονα προγραμματιστικά περιβάλλοντα γίνεται μέσα από σχεδιαστικά εργαλεία που προσφέρουν, τους Επεξεργαστές μενού επιλογών (Menu Editor).

Πολλά περιβάλλοντα παρέχουν τη δυνατότητα εμφάνισης μενού επιλογών με τη μορφή **πτυσσόμενων μενού** (popup menu) (σχήμα 11.8). Τα πτυσσόμενα μενού επιλογών, συνήθως εμφανίζονται με το πάτημα του δεξιού πλήκτρου του ποντικιού και τα χρησιμοποιούμε για τη εμφάνιση σύντομων επιλογών σε συγκεκριμένα σημεία της εφαρμογής.



Σχ. 11.8. Ένα πτυσσόμενο μενού επιλογών

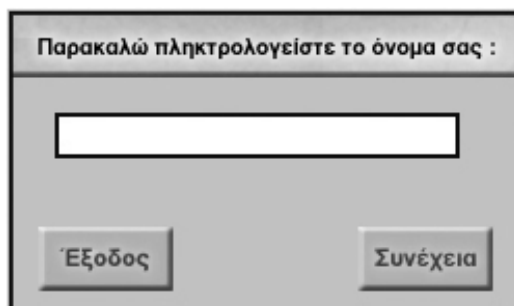


Σχ. 11.9. Μια γραμμή εργαλείων

Στις εφαρμογές μας αν θέλουμε να δώσουμε ένα σύντομο τρόπο πρόσβασης στις πιο συχνά εκτελέσιμες επιλογές ενός μενού, ο καλύτερος τρόπος είναι η δημιουργία **γραμμών εργαλείων** (toolbars) (σχήμα 11.9). Μια γραμμή εργαλείων μπορεί να υλοποιηθεί πολύ εύκολα με την ομαδοποίηση αντικειμένων εικόνας ή πλήκτρων εντολής που τα τοποθετούμε κάτω από τη γραμμή μενού και τον αντίστοιχο κώδικα.

#### 11.4.2 Πηαίσια διαλόγου

Τα **πλαίσια διαλόγου** (dialog boxes) (σχήμα 11.10) εξασφαλίζουν ένα άμεσο διάλογο της εφαρμογής με το χρήστη. Πολλές φορές κατά την εκτέλεση μιας εφαρμογής είναι αναγκαίο ο χρήστης να επικοινωνεί με την εφαρμογή. Μέσα από τα πλαίσια διαλόγου είναι δυνατό να εμφανιστούν μηνύματα προς το χρήστη, αλλά και ο χρήστης μπορεί να εισάγει κάποια δεδομένα στην εφαρμογή.



Σχ. 11.10. Ένα πλαίσιο διαλόγου



*Η σχεδίαση ενός πλαισίου διαλόγου πρέπει να αντιμετωπίζεται από την πλευρά του χρήστη. Πρέπει να χρησιμοποιούνται κατανοητά μηνύματα και να εμφανίζονται οι κατάλληλες επιλογές για την περίπτωση.*

Τα περισσότερα σύγχρονα προγραμματιστικά εργαλεία παρέχουν στο προγραμματιστή έτοιμα προκαθορισμένα πλαίσια διαλόγου που μπορεί να χρησιμοποιήσει στις εφαρμογές του. Τα προκαθορισμένα πλαίσια διαλόγου, έχουν το πλεονέκτημα ότι διατηρούν την ίδια φιλοσοφία με το περιβάλλον εργασίας, αφού είναι πανομοιότυπα με τα πλαίσια που εμφανίζει το σύστημα όταν πρέπει να επικοινωνήσει με το χρήστη, για παράδειγμα σε περίπτωση λάθους. Σε ένα προκαθορισμένο πλαίσιο διαλόγου έχουμε τη δυνατότητα να εμφανίσουμε διάφορα πλήκτρα, όπως πλήκτρο επικύρω-

σης (OK), ακύρωσης (Cancel) και επανάληψης (Retry). Η εμφάνιση των προκαθορισμένων πλαισίων διαλόγου γίνεται μέσω συναρτήσεων της γλώσσας προγραμματισμού.

Αν κανένα από τα προκαθορισμένα πλαίσια διαλόγου, που μας προσφέρει το περιβάλλον προγραμματισμού δεν καλύπτει κάποια ανάγκη της εφαρμογής μας ή δεν μας εκφράζει η τυποποίησή τους, έχουμε φυσικά τη δυνατότητα τοποθετώντας αντικείμενα επάνω σε μια φόρμα να δημιουργήσουμε τα δικά μας πλαίσια διαλόγου.

### 11.5. Επικοινωνία με άλλες εφαρμογές

Οι προγραμματιστές στον παραδοσιακό προγραμματισμό τις περισσότερες φορές αγνοούσαν υπάρχουσες εφαρμογές, γιατί το περιβάλλον εργασίας καθιστούσε δύσκολη τη συνεργασία τους. Έτσι κατέφευγαν στη συγγραφή τμημάτων κώδικα για την εκτέλεση εργασιών, που ήδη εκτελούσε ορθά μια άλλη εφαρμογή.

Όμως τα σύγχρονα περιβάλλοντα εργασίας παρέχουν τους μηχανισμούς για υψηλού επιπέδου επικοινωνία μεταξύ διαφορετικών εφαρμογών, του λειτουργικού συστήματος και της εφαρμογής μας και μας επιτρέπουν έτσι την εκμετάλλευση έτοιμων βιβλιοθηκών προγραμμάτων, ελαττώνοντας και τις απαιτήσεις σε συγγραφή κώδικα.

Επιπλέον μέσα από την εφαρμογή μας μπορούμε να έχουμε πρόσβαση και επικοινωνία με άλλου τύπου εφαρμογές για την ανάκτηση και ανάλυση δεδομένων διαφορετικής μορφής. Συγκεκριμένα, μέσα από μια εφαρμογή μας, μπορούμε να επεξεργαστούμε τα δεδομένα ενός λογιστικού φύλλου, να μορφοποιήσουμε και να εκτυπώσουμε τμήματα κειμένου που έχουν γραφτεί με έναν επεξεργαστή κειμένου, να χρησιμοποιήσουμε ένα πρόγραμμα επικοινωνίας για να στείλουμε ή να δεχτούμε μηνύματα, να αναλύσουμε τα δεδομένα μιας υπάρχουσας βάσης δεδομένων, να απεικονίσουμε γραφικά τα δεδομένα μας που προέρχονται από ένα πρόγραμμα παρουσιάσεων (σχήμα 11.11).

Οι σύγχρονες τεχνολογίες σύνδεσης, μας επιτρέπουν μέσα από τη δική μας εφαρμογή την αξιοποίηση των δεδομένων μιας άλλης ανεξάρτητης εξωτερικής εφαρμογής. Υπάρχουν δύο τρόποι διασύνδεσης εφαρμογών, φαινομενικά ίδιοι από την πλευρά του χρήστη αλλά με τεχνικές διαφοροποιήσεις.



*Ο προγραμματιστής σε ένα σύγχρονο προγραμματιστικό περιβάλλον, αντιμετωπίζει κάθε ολοκληρωμένο πρόβλημα σαν μια ανεξάρτητη εφαρμογή. Αν κάποια από τις εφαρμογές που ήδη κατέχει μπορεί να επιλύσει το πρόβλημα, τότε έχει την ευχέρεια να τη χρησιμοποιήσει. Αυτός ο τρόπος αντιμετώπισης καλείται εφαρμογοκεντρικός (document-centered view) και αποτελεί ένα υψηλού επιπέδου σύστημα πολυ-διεργασίας ή συνεπιτέλεσης (multitasking).*





Σχ. 11.11. Συνεργασία διαφορετικών τύπων εφαρμογών

Με τον πρώτο τρόπο, έχουμε τη δυνατότητα **διασύνδεσης** (linking) των δεδομένων μιας άλλης εφαρμογής με τη δική μας εφαρμογή. Τα δεδομένα της άλλης εφαρμογής που χρησιμοποιούμε στη δική μας, βρίσκονται σε κάποιο ξεχωριστό αρχείο του συστήματος. Με την τεχνική αυτή, η εφαρμογή μας καταφεύγει στο αρχείο και αντλεί τα δεδομένα, όποτε τα χρειαστεί, χωρίς να επιβαρύνεται σε χωρητικότητα. Τα δεδομένα αυτά της άλλης εφαρμογής παραμένουν προσπελάσιμα σε οποιαδήποτε τρίτη εφαρμογή μπορεί να τα χρειαστεί. Για παράδειγμα ένα αρχείο εικόνας είναι δυνατό να προσπελαίνεται από την εφαρμογή δημιουργίας του φυσικά, και ταυτόχρονα να είναι διασυνδεδεμένο τόσο στην εφαρμογή μας, όσο και σε κάποια τρίτη εφαρμογή. Κάθε χρονική στιγμή και στις τρεις εφαρμογές θα εμφανίζεται το πιο πρόσφατα ενημερωμένο αρχείο εικόνας.

Η τεχνική της διασύνδεσης, είναι ιδιαίτερα χρήσιμη για την υποστήριξη συνδέσεων πραγματικού χρόνου (real-time) μεταξύ εφαρμογών και για την κοινή προσπέλαση δεδομένων από διαφορετικές εφαρμογές με άμεση ενημέρωση των τροποποιήσεων του αντικειμένου. Μειονέκτημα αυτής της τεχνικής, είναι ότι πιθανή μετακίνηση των δεδομένων από το σημείο του συστήματος στο οποίο βρίσκονται, έχει σαν αποτέλεσμα την αδυναμία πρόσβασης της εφαρμογής μας στα δεδομένα αυτά, αν προηγουμένως δεν γίνει ενημέρωση του συνδέσμου: Χαρακτηριστικό του τρόπου αυτού σύνδεσης είναι ότι η τροποποίηση των δεδομένων της εξωτερικής εφαρμογής μπορεί

να γίνει μόνο μέσα από την εφαρμογή που τα δημιούργησε με ενεργοποίησή της είτε εξωτερικά, είτε εσωτερικά από την εφαρμογή μας

Ο δεύτερος τρόπος μας επιτρέπει την **ενσωμάτωση** (embedding) των δεδομένων μέσα στη δική μας εφαρμογή. Με την τεχνική αυτή εξασφαλίζεται ο αποκλειστικός έλεγχος των δεδομένων μέσα από την εφαρμογή μας. Τα δεδομένα δημιουργούνται μέσα από την εφαρμογή μας ή αντιγράφονται από κάποιο υπάρχον αρχείο και αποθηκεύονται στην εφαρμογή μας. Αυτό πρακτικά σημαίνει επιβάρυνση της εφαρμογής μας από την άποψη πως το αρχείο που δημιουργείται έχει μεγάλο μέγεθος, αλλά ταυτόχρονα εξασφαλίζεται η ασφάλεια των δεδομένων, αφού δεν επιτρέπεται να τροποποιηθούν από κάποια άλλη εφαρμογή.

Με την τεχνική της ενσωμάτωσης δύο εκδοχές είναι δυνατές. Τα δεδομένα να δημιουργηθούν εκείνη τη στιγμή ή να είναι ήδη έτοιμα. Στην πρώτη περίπτωση τα δεδομένα δημιουργούνται με τη χρήση μιας άλλης εφαρμογής που ενεργοποιείται μέσα από τη δική μας και ενσωματώνονται κατ' ευθείαν στην εφαρμογή μας. Για παράδειγμα, με την ενεργοποίηση μέσα από την εφαρμογή μας και τη χρήση ενός προγράμματος επεξεργασίας εικόνας, μπορούμε να δημιουργήσουμε ένα γραφικό που θα ενσωματωθεί στα δεδομένα της εφαρμογής μας. Σε αυτήν την περίπτωση δεν δημιουργείται κάποιο ξεχωριστό αρχείο στο σύστημα και επιπλέον δεν υπάρχει η δυνατότητα πρόσβασης από τη μεριά μιας τρίτης εφαρμογής σε αυτό το γραφικό.



*Η βασική διαφορά μεταξύ των τεχνικών διασύνδεσης και ενσωμάτωσης είναι ο τρόπος αποθήκευσης των συνδεδεμένων δεδομένων.*

Στη δεύτερη περίπτωση, υπάρχει ανεξάρτητο αρχείο στο σύστημα, από το οποίο και προέρχονται τα δεδομένα που χρησιμοποιούμε στην εφαρμογή μας. Αυτό βέβαια δεν σημαίνει ότι το ανεξάρτητο αυτό αρχείο δεσμεύεται από την εφαρμογή μας. Ένα αντίγραφο του ενσωματώνεται στο αρχείο της εφαρμογής μας. Αν για παράδειγμα έχουμε διασυνδέσει την εφαρμογή μας με ένα λογιστικό φύλλο και κάποια στιγμή θέλουμε να επιφέρουμε κάποιες αλλαγές, ενεργοποιούμε μέσα από την εφαρμογή μας την εφαρμογή επεξεργασίας λογιστικών φύλλων και πραγματοποιούμε τις επιθυμητές αλλαγές. Αν στην συνέχεια “ανοίξουμε” το ανεξάρτητο αρχείο του λογιστικού φύλλου που υπάρχει στο σύστημα, θα διαπιστώσουμε ότι οι αλλαγές αυτές δεν έχουν καταγραφεί σε αυτό.

Ανακεφαλαιώνοντας, τα δεδομένα που συνδέονται με τη εφαρμογή μας με τη τεχνική της διασύνδεσης αποθηκεύονται στην εφαρμογή η οποία τα δημιούργησε, ενώ αντίθετα τα δεδομένα που συνδέονται στην εφαρμογή μας με τη τεχνική της ενσωμάτωσης, περιέχονται στην εφαρμογή μας και αποθηκεύονται μαζί της.

## Ανακεφαλαίωση

Σε ένα σύγχρονο προγραμματιστικό περιβάλλον όπως είναι αυτά του αντικειμενοστραφούς και του οδηγούμενου από γεγονότα προγραμματισμού, προσφέρονται νέοι τρόποι υλοποίησης των εφαρμογών, νέα εργαλεία και τεχνικές, αυξημένο γραφικό περιβάλλον εργασίας, αλλά παράλληλα οι κλασικές τεχνικές του τμηματικού και του δομημένου προγραμματισμού είναι πάντοτε απαραίτητες και εφαρμόσιμες.

Μπορεί κάποιος να αναρωτηθεί, αν ένα σύγχρονο περιβάλλον κάνει δύσκολη τη ζωή των προγραμματιστών. Η απάντηση σίγουρα είναι αρνητική, αφού ο προγραμματιστής έχει πλέον ως εφόδιά του ένα σύνολο βοηθημάτων που τον επικουρούν, τον επιβλέπουν και τον διορθώνουν σε όλη τη διάρκεια της προγραμματιστικής εργασίας του. Φυσικά στην αρχή, ένας προγραμματιστής θα δυσκολευτεί με το τρόπο που πρέπει να συνδυάσει όλες τις νέες τεχνικές προγραμματισμού, αλλά πολύ σύντομα θα εξοικειωθεί και θα σκέφτεται κάθε εφαρμογή του με όρους αντικειμένων και γεγονότων.

Το μεγαλύτερο πλεονέκτημα ενός σύγχρονου προγραμματιστικού περιβάλλοντος, είναι ότι μας επιτρέπει να δημιουργήσουμε στις εφαρμογές μας ένα φιλικό, εύχρηστο και ευχάριστο περιβάλλον εργασίας για κάθε χρήστη. Τα γραφικά αντικείμενα που μας παρέχουν τα περιβάλλοντα αυτά συνδράμουν προς αυτήν την κατεύθυνση.

Σε ένα μοντέρνο και ολοκληρωμένο περιβάλλον, έχουμε τη δυνατότητα να επιτύχουμε την αρμονική συνεργασία διαφορετικών εφαρμογών. Μέσα από τη δική μας εφαρμογή μπορούμε να εμφανίσουμε και να επεξεργαστούμε δεδομένα από διαφορετικές πηγές. Επικοινωνώντας με τις υπόλοιπες εφαρμογές του συστήματος γλιτώνουμε χρόνο και κόστος, αλλά κυρίως εκμεταλλευόμαστε εφαρμογές που ήδη λειτουργούν σωστά και ο χρήστης είναι εξοικειωμένος με την χρήση τους.

## Λέξεις κλειδιά

Αντικειμενοστραφής προγραμματισμός, οδηγούμενος από γεγονότα προγραμματισμός, αντικείμενα, κλάσεις, μέθοδοι, γεγονότα, τρόπος επικοινωνίας χρήστη-εφαρμογής, γραφικό περιβάλλον επικοινωνίας, επικοινωνία εφαρμογών, μενού επιλογών, πλαίσια διαλόγου, διασύνδεση, ενσωμάτωση.





### Ερωτήσεις - Θέματα για συζήτηση

1. Να γίνει συζήτηση σχετικά με τα πλεονεκτήματα που μας προσφέρουν τα σύγχρονα περιβάλλοντα προγραμματισμού.
2. Περιγράψτε τη μεθοδολογία ανάπτυξης μιας σύγχρονης εφαρμογής.
3. Περιγράψτε τη ροή εκτέλεσης μιας εφαρμογής που διέπεται από τις αρχές του αντικειμενοστραφή και του οδηγούμενου από γεγονότα προγραμματισμού.
4. Περιγράψτε γραφικά αντικείμενα που είναι δυνατό να χρησιμοποιήσετε στις εφαρμογές σας.
5. Δώστε τον ορισμό του αντικειμένου.
6. Δώστε παραδείγματα αντικειμένων σε προγραμματιστικό επίπεδο.
7. Τι είναι οι μέθοδοι;
8. Δώστε τον ορισμό του γεγονότος.
9. Από που αντλούν τα αντικείμενα τις αρχικές ιδιότητες και την συμπεριφορά τους;
10. Περιγράψτε παραδείγματα μενού επιλογών που είναι δυνατό να χρησιμοποιήσετε σε μια εφαρμογή;
11. Ποια είναι τα πλεονεκτήματα των προκαθορισμένων πλαισίων διαλόγου;
12. Ποια η διαφορά ενός ενσωματωμένου από ένα συνδεδεμένο αντικείμενο;
13. Να γίνουν εκτιμήσεις για τις μελλοντικές εξελίξεις στο χώρο του προγραμματισμού.

### Βιβλιογραφία



1. Object-Oriented modeling and Design, Prentice Hall, New York, 1991.
2. G. Masini, A. Napoli, D. Colnet, D. Lionard, K. Tombre : Les langages á objets, InterEditions, Paris, 1989.
3. Gary Entsminger: Secrets of the Visual Basic for Windows, SAMS Publishing, Indiana USA, 1992.
4. Bertrand Meyer : Object-oriented Software Construction, Prentice Hall, London, 1988.

5. Παν. Πολίτης-Ηλ. Γιαννόπουλος: Προγραμματισμός με τη Visual Basic 4.0, Εκδόσεις Νέων Τεχνολογιών, Αθήνα, 1997.

## Διευθύνσεις Διαδικτύου

⇒ <http://iamwww.unibe.ch/~scg/Ooinfo/>

Ενημερωμένη ιστοσελίδα που περιέχει ευρετήριο για πηγές πληροφόρησης στο Internet, σχετικές με τις αντικειμενοστραφείς γλώσσες προγραμματισμού και τα συστήματα που εφαρμόζονται.

⇒ <http://whatis.com/oop.htm>

Σε αυτή την ιστοσελίδα περιγράφεται ο αντικειμενοστραφής προγραμματισμός και αναλύεται η φιλοσοφία του.

⇒ <http://oop.cs.technion.ac.il>

Περιέχει τεχνικά πληροφοριακά δελτία για τον αντικειμενοστραφή προγραμματισμό.

⇒ [www.oop.com/frameworks/delphi/](http://www.oop.com/frameworks/delphi/)

Αναλύει την εφαρμογή του αντικειμενοστραφή προγραμματισμού στο περιβάλλον Delphi.

⇒ <http://www.humboldt.edu/~jsb7/C/WIN95/events.shtml>

Σε αυτή την ιστοσελίδα περιγράφεται η εφαρμογή του οδηγούμενου από τα γεγονότα προγραμματισμού.

⇒ <http://midrangecomputing.com/mc/97/05/>

Σύνοψη του οδηγούμενου από τα γεγονότα προγραμματισμού και σύγκρισή του με παραδοσιακές μεθόδους προγραμματισμού.

⇒ <http://service.shu.ac.uk/schools/sci/mathsjw/jw/vbintro/>

Εισαγωγικές έννοιες για τον οπτικό προγραμματισμό και την εφαρμογή του στα περιβάλλοντα προγραμματισμού Visual Basic και Delphi.

⇒ <http://www.cnet.com/Resources/Info/Glossary/Terms/ole.html>

Ευρετήριο όρων σχετικά με την τεχνική OLE (Object Linking and Embedding).

