Η Εισαγωγή στον Αντικειμενοστραφή Προγραμματισμό: Προβλήματα και Μεθοδολογίες για την Αντιμετώπισή τους

Ξυνόγαλος Στέλιος
Διδάκτορας Εκπαιδευτικής Τεχνολογίας, Καθηγητής Πληροφορικής Β/θμιας Εκπ/σης
Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας
stelios@uom.gr

Σατρατζέμη Μάγια
Αναπληρώτρια Καθηγήτρια
Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας
maya@uom.gr

ΠΕΡΙΛΗΨΗ

Η διδασκαλία και εκμάθηση του αντικειμενοστραφούς προγραμματισμού, όπως προκύπτει από τη διεθνή βιβλιογραφία, παρουσιάζει αρκετές δυσκολίες. Στην παρούσα εργασία παρουσιάζεται μια σύνοψη των προβλημάτων που παρουσιάζονται στα μαθήματα εισαγωγής στον αντικειμενοστραφή προγραμματισμό και των δυσκολιών που αντιμετωπίζουν οι σπουδαστές. Επίσης, παρουσιάζονται κάποιες μεθοδολογίες για την αντιμετώπιση των καταγεγραμμένων προβλημάτων και γίνονται κάποιες προτάσεις για την αποτελεσματική αξιοποίηση αυτών των μεθοδολογιών.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: *αντικειμενοστραφής προγραμματισμός, μεθοδολογίες διδασκαλίας, προγραμματιστικός μικρόκοσμος, εκπαιδευτικό προγραμματιστικό περιβάλλον*

ΕΙΣΑΓΩΓΗ

Η εισαγωγή στον προγραμματισμό την τελευταία δεκαετία πραγματοποιείται, κατά κύριο λόγο, χρησιμοποιώντας το αντικειμενοστραφές παράδειγμα προγραμματισμού. Ακόμη όμως και στις περιπτώσεις που εξακολουθεί να χρησιμοποιείται το παράδειγμα του δομημένου προγραμματισμού για την εισαγωγή στον προγραμματισμό, στη συνέχεια, στα πλαίσια του προγράμματος σπουδών των εκπαιδευτικών ιδρυμάτων, διδάσκεται και ο αντικειμενοστραφής προγραμματισμός. Ανεξάρτητα από τη θέση του αντικειμενοστραφούς προγραμματισμού στο πρόγραμμα σπουδών, η διδασκαλία και η εκμάθησή του συνοδεύεται από ποικίλες δυσκολίες. Το γεγονός αυτό αποδίδεται συνήθως στο ότι η αντικειμενοστραφής τεχνική σχεδίασης είναι πιο αφηρημένη και πιο απαιτητική, όσον αφορά στις διαδικασίες της ανάλυσης και του σχεδιασμού, σε σχέση με την τεχνική του δομημένου προγραμματισμού. Ωστόσο, θα δείξουμε ότι το πρόβλημα έγκειται, κατά κύριο λόγο: στην *έλλειψη εκπαιδευτικών εργαλείων και διδακτικής εμπειρίας/γνώσης* για το αντικειμενοστραφές παράδειγμα προγραμματισμού, στη χρήση μιας μη αποτελεσματικής *μεθοδολογίας διδασκαλίας* και στη μη αξιοποίηση των αποτελεσμάτων των ερευνών σχετικών με τις δυσκολίες των αρχάριων κατά την εκμάθηση του αντικειμενοστραφούς προγραμματισμού.

Στην παρούσα εργασία γίνεται μια προσπάθεια σύνοψης της διεθνούς διδακτικής εμπειρίας σχετικά με τον αντικειμενοστραφή προγραμματισμό. Η σύνοψη αυτή συνίσταται στην παρουσίαση των προβλημάτων που συνοδεύουν τη διδασκαλία του αντικειμενοστραφούς προγραμματισμού και των δυσκολιών των σπουδαστών. Τη σύνοψη των προβλημάτων

διδασκαλίας ακολουθεί η περιγραφή των μεθοδολογιών που έχουν προταθεί για την αντιμετώπισή τους, καθώς επίσης και κάποιες προτάσεις για την παιδαγωγική αξιοποίησή τους.

ΠΡΟΒΛΗΜΑΤΑ ΣΤΗ ΔΙΔΑΣΚΑΛΙΑ ΤΟΥ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Στην ενότητα αυτή συνοψίζονται *γενικά προβλήματα* που έχουν καταγραφεί στη διεθνή βιβλιογραφία για τη διδασκαλία του αντικειμενοστραφούς προγραμματισμού.

Μετάβαση από το διαδικαστικό στον αντικειμενοστραφή προγραμματισμό

Αρκετές εργασίες έχουν εκπονηθεί με αντικείμενο τη θέση των διάφορων παραδειγμάτων προγραμματισμού στο πρόγραμμα σπουδών των Πανεπιστημιακών ιδρυμάτων. Οι εργασίες αυτές παρουσιάζουν απόψεις και εμπειρίες διδασκόντων σε Πανεπιστημιακά ιδρύματα και αναφέρονται κατά κύριο λόγο στον αντικειμενοστραφή και στον διαδικαστικό προγραμματισμό, καθώς και στη σειρά με την οποία πρέπει να διδάσκονται αυτά τα παραδείγματα προγραμματισμού. Αν και οι απόψεις διίστανται, η πλειοψηφία των διδασκόντων/ερευνητών (Decker & Hirshfield, 1994; Hadjerrouit, 1998; Tempte, 1991; Wick, 1995) πιστεύει ότι οι σπουδαστές αντιμετωπίζουν περισσότερες δυσκολίες όταν μεταβαίνουν από τον κατηγορηματικό (imperative) – διαδικαστικό (procedural) προγραμματισμό στον αντικειμενοστραφή, ενώ το αντίστροφο δεν ισχύει. Συγκεκριμένα, οι (Decker & Hirshfield, 1994) αναφέρουν ότι η μετάβαση είναι δύσκολη μόνο προς τη μια κατεύθυνση – από το διαδικαστικό στον αντικειμενοστραφή προγραμματισμό. Επιπλέον, ο αντικειμενοστραφής προγραμματισμός μπορεί να χρησιμοποιηθεί ως μέσο ανάπτυξης και διαδικαστικών ικανοτήτων *(procedural skills)*. Ο (Tempte, 1991) αναφέρει ότι παρά τους ισχυρισμούς ότι η αντικειμενοστραφής προσέγγιση επίλυσης προβλημάτων είναι πιο εύκολη - φυσική για την ακρίβεια-, ο νέος τρόπος σκέψης που απαιτεί δεν γίνεται εύκολα κατανοητός από σπουδαστές που έχουν εμπειρία στην επίλυση προβλημάτων με μια διαδικαστική γλώσσα προγραμματισμού. Ο (Handjerrouit, 1998; Handjerrouit, 1999) αναφέρει ότι οι σπουδαστές, οι οποίοι είχαν προηγούμενη εμπειρία με διαδικαστικές γλώσσες προγραμματισμού, κατά την εισαγωγή τους στη Java αντιμετώπισαν δυσκολίες με τις έννοιες του αντικειμενοστραφούς προγραμματισμού και συγκεκριμένα με την αξιοποίησή τους. Για παράδειγμα, οι σπουδαστές παρουσίασαν την τάση να χρησιμοποιούν τις μεθόδους ως διαδικασίες, αγνοώντας το ρόλο τους στα πλαίσια του αντικειμενοστραφούς προγραμματισμού.

Η αντικειμενοστραφής τεχνική ανάπτυξης προγραμμάτων είναι δύσκολη

Όπως αναφέρει ο (Handjerrouit, 1999) η εκμάθηση της τεχνικής ανάπτυξης προγραμμάτων που προτείνει το αντικειμενοστραφές παράδειγμα προγραμματισμού είναι δύσκολη για τους αρχάριους. Η δυσκολία έγκειται στο γεγονός ότι η τεχνική αυτή είναι πιο αφηρημένη από την τεχνική του δομημένου προγραμματισμού, απαιτεί νέους τρόπους σκέψης και είναι πιο απαιτητική όσον αφορά τις διαδικασίες της ανάλυσης και σχεδίασης. Οι (Ramalingam & Wiedenbeck, 1997) διεξήγαγαν μια εμπειρική μελέτη με στόχο να διερευνήσουν αν οι νοητές αναπαραστάσεις που δημιουργούν οι αρχάριοι προγραμματιστές για αντικειμενοστραφή και κατηγορηματικά προγράμματα διαφέρουν. Η μελέτη διεξήχθη στα πλαίσια ενός μαθήματος εισαγωγής στον προγραμματισμό που χρησιμοποιούσε ως γλώσσα προγραμματισμού τη C++. Οι 75 συμμετέχοντες μελέτησαν τρία κατηγορηματικά και τρία αντικειμενοστραφή προγράμματα – τα οποία ήταν απλά-, και απάντησαν σε ερωτήσεις που αφορούσαν στις αναπαραστάσεις που δημιούργησαν. Από τη μελέτη αυτή προέκυψε ότι τα νοητά μοντέλα που δημιουργούν οι φοιτητές για τα αντικειμενοστραφή και τα κατηγορηματικά προγράμματα διαφέρουν ουσιαστικά. Συγκεκριμένα, οι φοιτητές κατανόησαν σε μεγαλύτερο βαθμό τα κατηγορηματικά προγράμματα. Τα νοητά μοντέλα που δημιούργησαν οι φοιτητές για τα κατηγορηματικά προγράμματα επικεντρώθηκαν στις στοιχειώδης διαδικασίες (elementary operations) και τη ροή ελέγχου (control

flow) του προγράμματος που περιγράφονται από τον πηγαίο κώδικα, δημιούργησαν δηλαδή ένα μοντέλο για το πρόγραμμα (program model). Αντίθετα, τα νοητά μοντέλα που δημιούργησαν οι φοιτητές για τα αντικειμενοστραφή προγράμματα επικεντρώθηκαν περισσότερο στη ροή των δεδομένων (data flow) και τη λειτουργία του προγράμματος (function knowledge), δημιούργησαν δηλαδή ένα μοντέλο για τη λειτουργία του προγράμματος συνολικά (domain model).

Αδυναμία διδασκαλίας του παραδείγματος στα πλαίσια των μαθημάτων εισαγωγής στον προγραμματισμό

Οι (Decker & Hirshfield, 1994) αναφέρουν ότι το πρόβλημα αυτό δεν υφίσταται, ενώ οι (Brilliant & Wiseman, 1996) επισημαίνουν ότι από την ανάλυση των ερωτηματολογίων που συμπληρώθηκαν από διδάσκοντες σε διάφορα τμήματα Επιστήμης των Υπολογιστών προκύπτουν αντικρουόμενα αποτελέσματα. Ο (Handjerrouit, 1998) επισημαίνει ότι ένα μάθημα δεν είναι αρκετό για τη διδασκαλία του αντικειμενοστραφούς παραδείγματος προγραμματισμού, αλλά απαιτείται μια σειρά ανεξάρτητων μαθημάτων. Συγκεκριμένα, η εισαγωγή στο αντικειμενοστραφές παράδειγμα προγραμματισμού πρέπει να γίνεται αρκετά νωρίς και να ενισχύεται σε όλο το προπτυχιακό πρόγραμμα σπουδών.

Οι υπάρχουσες γλώσσες προγραμματισμού δεν είναι κατάλληλες για την εισαγωγή στο αντικειμενοστραφές παράδειγμα προγραμματισμού

Οι απόψεις των διδασκόντων/ερευνητών για το θέμα αυτό, σε αντίθεση με τα προηγούμενα, συμπίπτουν. Οι Dung Nguyen, Wallingford (Berman et al., 1994), Kolling, Koch και Rosenberg (Kolling et al., 1995) επισημαίνουν ότι οι σπουδαστές αντιμετωπίζουν δυσκολίες με τη σύνταξη της C++, ενώ πολύς χρόνος αφιερώνεται στην εκμάθησή της. Οι Kolling et al. μετά από μια αναφορά στα «μειονεκτήματα» των C++, Smalltalk, Eiffel και Sather καταλήγουν στο συμπέρασμα ότι καμία από τις παραπάνω γλώσσες δεν είναι κατάλληλη για τη διδασκαλία του αντικειμενοστραφούς προγραμματισμού (Kolling et al., 1995).

Τα εισαγωγικά μαθήματα προγραμματισμού διδάσκουν τους σπουδαστές να είναι μόνο «καταναλωτές» και όχι «παραγωγοί» επαναχρησιμοποιήσιμου λογισμικού

Ο Dung Nguyen επισημαίνει ότι οι σπουδαστές δυσκολεύονται να κατανοήσουν την απόκρυψη πληροφοριών (information hiding) όταν χρειάζεται να χρησιμοποιούν, να σχεδιάζουν και να υλοποιούν μια κλάση, με αποτέλεσμα να χρειάζεται ο διδάσκοντας να σχεδιάζει τις κλάσεις γι' αυτούς (Berman et al., 1994).

ΔΥΣΚΟΛΙΕΣ ΤΩΝ ΣΠΟΥΔΑΣΤΩΝ

Τα προβλήματα που παρουσιάζονται στα μαθήματα εισαγωγής στον αντικειμενοστραφή προγραμματισμό έχουν ως άμεση συνέπεια την εμφάνιση πληθώρας δυσκολιών. Τα στοιχεία προέρχονται, κατά κύριο λόγο, από συνεντεύξεις σπουδαστών (Aurora University & University of Kent at Canterbury) στα πλαίσια εισαγωγικών μαθημάτων στον αντικειμενοστραφή προγραμματισμό χρησιμοποιώντας τη Java ως γλώσσα προγραμματισμού (Fleury, 2000; Fleury, 2001; Carter & Fowler, 1998) και την παρατήρηση των σπουδαστών (The Open University, United Kingdom) στα πλαίσια του εξ' αποστάσεως προπτυχιακού εισαγωγικού μαθήματος "*Computing: an Object-oriented approach"* και του μεταπτυχιακού μαθήματος "*Object-oriented Software Technology"* (Holland et al., 1997). Οι σημαντικότερες δυσκολίες που αντιμετωπίζουν οι σπουδαστές κατά την εισαγωγή τους στον αντικειμενοστραφή προγραμματισμό είναι οι εξής:

- *Χρήση πολλαπλών κατασκευαστών* (Carter & Fowler, 1998).
- *Ταύτιση κλάσης/αντικειμένου*: οι Holland, Griffiths και Woodman, σε αντίθεση με τους Carter και Fowler (Carter & Fowler, 1998), αναφέρουν ότι κάποιοι σπουδαστές συγχέουν και δεν μπορούν να διαχωρίσουν τις κλάσεις από τα στιγμιότυπά τους (αντικείμενο κλάσης). Το

πρόβλημα είναι πιο έντονο όταν στα παραδείγματα χρησιμοποιείται μόνο ένα αντικείμενο της κάθε κλάσης (Holland et al., 1997).

- *Ταύτιση αντικειμένου/μεταβλητής*: αν στα παραδείγματα που παρουσιάζονται στα αρχικά μαθήματα χρησιμοποιείται μία μόνο μεταβλητή (instance variable) σε κάθε κλάση, είναι πολύ πιθανό να δημιουργηθεί η παρανόηση ότι κάθε αντικείμενο αποτελεί απλά ένα «περιτύλιγμα» μιας μεταβλητής.
- *Σύγχυση ταυτότητας/χαρακτηριστικού* (Identity/attribute confusion)*:* η σύγχυση αυτή μπορεί να συντελέσει στη δημιουργία αρκετών λανθασμένων αντιλήψεων (Holland et al., 1997): (1) μόνο μια μεταβλητή μπορεί να αναφέρεται σε ένα δεδομένο αντικείμενο μια δεδομένη στιγμή, (2) από τη στιγμή που μια μεταβλητή αναφέρεται σε ένα δεδομένο αντικείμενο, θα αναφέρεται πάντα σε αυτό, (3) αν υπάρχουν δύο διαφορετικές μεταβλητές, αυτές πρέπει να αναφέρονται σε δύο διαφορετικά αντικείμενα, (4) δύο αντικείμενα της ίδιας κλάσης με την ίδια κατάσταση είναι το ίδιο αντικείμενο, (5) δύο αντικείμενα που έχουν την ίδια τιμή για ένα χαρακτηριστικό (μεταβλητή) είναι το ίδιο αντικείμενο. Για παράδειγμα, αν έχουμε μια κλάση *Account* με μεταβλητές *name* και *balance* και υπάρχουν 2 αντικείμενα με τιμή "207/344915-72" για το χαρακτηριστικό *name*, αυτά αποτελούν το ίδιο αντικείμενο.
- *Τα αντικείμενα είναι απλές εγγραφές χωρίς συμπεριφορά*: είναι πολύ πιθανό οι σπουδαστές να μην καταλάβουν ότι η συμπεριφορά ενός αντικειμένου μπορεί να αλλάξει ουσιαστικά ανάλογα με την κατάστασή του (Holland et al., 1997).
- *Προαιρετική χρήση κατασκευαστών* (Constructors): ο μόνος λόγος χρήσης της ειδικής συνάρτησης μέλους για την κατασκευή ενός αντικειμένου, γνωστή ως constructor, είναι η αρχικοποίηση των μεταβλητών του αντικειμένου (Fleury, 2000). Η δημιουργία ενός αντικειμένου μπορεί να επιτευχθεί και με μια μέθοδο, έστω set_values, που δίνει αρχικές τιμές στις μεταβλητές του νέου αντικειμένου.
- *Μια μέθοδος μεταβάλει την κατάσταση ενός αντικειμένου απλά και μόνο με ανάθεση και όχι με αποστολή μηνυμάτων*: όταν τα παραδείγματα που χρησιμοποιούνται στα πρώτα μαθήματα βασίζονται σε αντικείμενα που η κατάστασή τους είναι αμετάβλητη, όπως για παράδειγμα αριθμητικά αντικείμενα, είναι δύσκολο να αποφευχθεί η εν λόγω παρανόηση. Ακόμη και έμπειροι σπουδαστές πέφτουν στην παγίδα (Holland et al., 1997).
- *Η χρήση dot operator είναι έγκυρη μόνο για μεθόδους* και όχι για μεταβλητές (Fleury, 2000).
- *Η μικρή έκταση ενός προγράμματος είναι προτιμότερη από την ενθυλάκωση*: πολλοί σπουδαστές θεωρούν τη μείωση του αριθμού των γραμμών κώδικα και των κλάσεων πιο σημαντική από την ενθυλάκωση. Αυτό τους διευκολύνει και στην ευκολότερη ανίχνευση των προγραμμάτων τους (Fleury, 2001).
- *Πολυμορφισμός υπό προϋπόθεση*: μέθοδοι διαφορετικών κλάσεων μπορούν να έχουν το ίδιο όνομα μόνο αν έχουν διαφορετικές υπογραφές (Fleury, 2000).

ΜΕΘΟΔΟΛΟΓΙΕΣ ΔΙΔΑΣΚΑΛΙΑΣ

Τα προβλήματα και οι δυσκολίες που παρουσιάζονται κατά την εισαγωγή στον αντικειμενοστραφή προγραμματισμό συχνά αποδίδονται στο γεγονός ότι η αντικειμενοστραφής τεχνική ανάπτυξης προγραμμάτων είναι πιο αφηρημένη από εκείνη του δομημένου προγραμματισμού, καθώς επίσης και πιο απαιτητική όσον αφορά στις διαδικασίες της ανάλυσης ενός προβλήματος και της σχεδίασης ενός αλγορίθμου για την επίλυσή του (Hadjerrouit, 1999). Ωστόσο, πιστεύουμε ότι το πρόβλημα έγκειται, κατά κύριο λόγο, στους εξής παράγοντες: (1) *έλλειψη εκπαιδευτικών εργαλείων και διδακτικής εμπειρίας/γνώσης* για το αντικειμενοστραφές παράδειγμα προγραμματισμού, και (2) *χρήση της κλασικής προσέγγισης διδασκαλίας* (Brusilovsky et al., 1997), η οποία συνίσταται στη χρήση μιας συμβατικής γλώσσας προγραμματισμού και ενός επαγγελματικού προγραμματιστικού περιβάλλοντος για τη γλώσσα αυτή. Η γλώσσα που χρησιμοποιείται είναι συνήθως η C++ και η Java, ενώ τα προβλήματα που επιλύονται αφορούν κατά κύριο λόγο στην επεξεργασία αριθμών και συμβόλων. Τα προβλήματα που προκύπτουν από

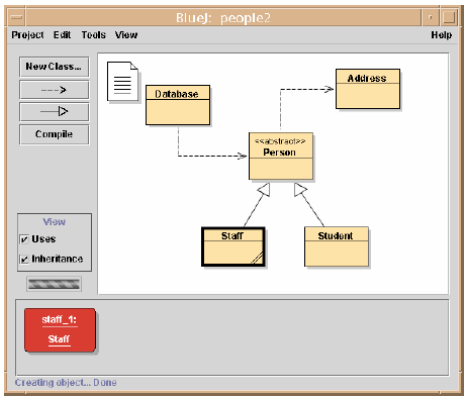
τη διδασκαλία του προγραμματισμού με την κλασική προσέγγιση είναι ποικίλα (Ξυνόγαλος, Σατρατζέμη & Δαγδιλέλης, 2000), ανεξάρτητα από το παράδειγμα προγραμματισμού. Για την αντιμετώπιση των προβλημάτων που παρουσιάζονται κατά την εισαγωγή στον αντικειμενοστραφή προγραμματισμό έχουν προταθεί οι παρακάτω μεθοδολογίες:

Παρουσίαση παραδειγμάτων και ανάθεση ασκήσεων ειδικά σχεδιασμένων ώστε να αποφευχθούν οι συνήθεις δυσκολίες

Οι (Holland et al., 1997) έχουν καταγράψει, όπως αναφέρθηκε στην προηγούμενη ενότητα, αρκετές δυσκολίες των σπουδαστών κατά την εισαγωγή τους στον αντικειμενοστραφή προγραμματισμό. Σύμφωνα με τους (Holland et al., 1997) το πρόβλημα μπορεί να αντιμετωπιστεί με την παρουσίαση παραδειγμάτων και την ανάθεση ασκήσεων ειδικά σχεδιασμένων για την αποφυγή των δυσκολιών οι οποίες έχουν καταγραφεί και οι οποίες είναι δύσκολο να αντιμετωπιστούν αν υιοθετηθούν από τους σπουδαστές. Για παράδειγμα, στα παραδείγματα που παρουσιάζονται και στις ασκήσεις που ανατίθενται στους σπουδαστές στα μαθήματα εισαγωγής στον αντικειμενοστραφή προγραμματισμό χρησιμοποιείται συνήθως, για λόγους απλότητας, ένα μόνο στιγμιότυπο (αντικείμενο) μιας κλάσης. Ως αποτέλεσμα αρκετοί σπουδαστές συγχέουν τις κλάσεις με τα στιγμιότυπά τους και δημιουργείται η γνωστή παρανόηση ότι κλάση και αντικείμενο είναι έννοιες ταυτόσημες. Για την αποφυγή αυτής της παρανόησης είναι καλό να χρησιμοποιούνται στα παραδείγματα και στις ασκήσεις αρκετά αντικείμενα της κάθε κλάσης.

Χρήση ενός εκπαιδευτικού προγραμματιστικού περιβάλλοντος

Οι (Kolling et al., 2003), όπως και αρκετοί άλλοι διδάσκοντες, αναφέρουν ότι η διδασκαλία του αντικειμενοστραφούς προγραμματισμού παρουσιάζει περισσότερες δυσκολίες σε σχέση με το δομημένο προγραμματισμό. Ωστόσο, οι (Kolling et al., 2003) θεωρούν ότι το πρόβλημα δεν έγκειται στο ότι ο αντικειμενοστραφής προγραμματισμός είναι εν γένει πιο πολύπλοκος. Αντίθετα, πιστεύουν ότι η διδασκαλία του καθίσταται πιο πολύπλοκη λόγω της έλλειψης κατάλληλων εργαλείων και παιδαγωγικής εμπειρίας για το συγκεκριμένο παράδειγμα. Τα σημαντικότερα προβλήματα που παρουσιάζουν τα περισσότερα περιβάλλοντα που χρησιμοποιούνται για τη διδασκαλία του αντικειμενοστραφούς προγραμματισμού είναι τα εξής: (1) το περιβάλλον δεν είναι αντικειμενοστραφές, (2) το περιβάλλον είναι πολύ πολύπλοκο, και (3) το περιβάλλον επικεντρώνεται στην ανάπτυξη γραφικών ενδιάμεσων και όχι στην αντικειμενοστρέφεια. Για την αντιμετώπιση των παραπάνω προβλημάτων οι (Kolling et al., 2003) ανέπτυξαν το εκπαιδευτικό περιβάλλον ***BlueJ***. Το BlueJ (http://www.bluej.org) είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον που χρησιμοποιεί ως γλώσσα προγραμματισμού την *Java*. Όταν ο σπουδαστής ανοίγει ένα project παρουσιάζεται στο βασικό παράθυρο του περιβάλλοντος (Εικόνα 1) ένα *διάγραμμα UML*, το οποίο οπτικοποιεί τη δομή της εφαρμογής. Ο σπουδαστής ξεκινάει με ένα σύνολο προκαθορισμένων κλάσεων, δημιουργεί αντικείμενα και καλεί τις διαθέσιμες μεθόδους προκειμένου να μελετήσει τη συμπεριφορά των αντικειμένων. Στη συνέχεια χρησιμοποιούνται υπάρχοντα προγράμματα για την παρουσίαση της σύνταξης της Java. Σε επόμενο στάδιο οι σπουδαστές επεκτείνουν υπάρχουσες κλάσεις



Εικόνα 1: Το βασικό παράθυρο του BlueJ.

υλοποιώντας ή προσθέτοντας δικές τους μεθόδους. Το επόμενο βήμα είναι η δημιουργία κλάσεων από τους σπουδαστές στα πλαίσια ενός υπάρχοντος project. Τέλος, οι σπουδαστές χωρίζονται σε ομάδες και δημιουργούν μια ολοκληρωμένη εφαρμογή.

Οι (Kolling & Rosenberg, 2001) πέραν της προτροπής για χρήση ενός εκπαιδευτικού περιβάλλοντος για τη διδασκαλία του αντικειμενοστραφούς προγραμματισμού, όπως το BlueJ που ανέπτυξαν, διατύπωσαν τις ακόλουθες *οδηγίες* για τη διδασκαλία της αντικειμενοστρέφειας με τη Java: (1) η διδασκαλία ξεκινάει με τα αντικείμενα, (2) μην ξεκινάς με κενή οθόνη, (3) μελέτησε κώδικα, (4) χρησιμοποίησε «μεγάλα» project, (4) μην ξεκινάς με τη συνάρτηση "main", (6) μην χρησιμοποιείς το γνωστό παράδειγμα "Hello World", (7) παρουσίασε τη δομή του προγράμματος, και (8) χρειάζεται προσοχή με το ενδιάμεσο χρήστη.

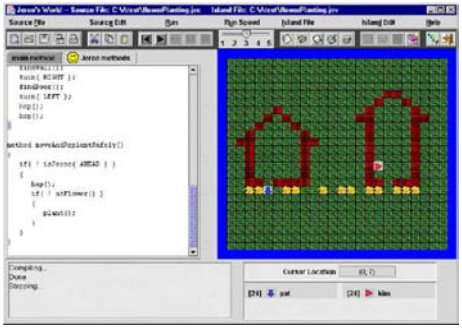
Χρήση ενός προγραμματιστικού μικρόκοσμου

Μια άλλη μεθοδολογία που έχει προταθεί από αρκετούς ερευνητές για την αντιμετώπιση των προβλημάτων που παρουσιάζονται κατά την εισαγωγή στον αντικειμενοστραφή προγραμματισμό είναι η χρήση προγραμματιστικών μικρόκοσμων. Οι προγραμματιστικοί μικρόκοσμοι, οι οποίοι σχεδιάζονται και αναπτύσσονται για εκπαιδευτικούς καθαρά σκοπούς, προσπαθούν να αντιμετωπίσουν τα προβλήματα που αναφέρθηκαν στην αρχή της ενότητας στο σύνολό τους:

- Βασίζονται συνήθως σε υπαρκτά φυσικά μοντέλα και χρησιμοποιούν μια *εκπαιδευτική γλώσσα προγραμματισμού*. Σε ορισμένες περιπτώσεις η γλώσσα αυτή μπορεί να αποτελεί υποσύνολο μιας συμβατικής γλώσσας ή/και να παρουσιάζει αρκετές ομοιότητες με αυτή.
- Είναι *ολοκληρωμένα προγραμματιστικά περιβάλλοντα* που χαρακτηρίζονται από *ευχρηστία* και ενσωματώνουν *τεχνολογίες για τη στήριξη των σπουδαστών*. Για παράδειγμα, η πλειοψηφία των προγραμματιστικών μικρόκοσμων ενσωματώνει δυνατότητες οπτικοποίησης.
- Στα πλαίσια ενός μικρόκοσμου επιλύονται, συνήθως, προβλήματα που βασίζονται σε εμπειρίες της καθημερινής ζωής.

Οι προγραμματιστικοί μικρόκοσμοι που έχουν αναπτυχθεί για την εισαγωγή στον αντικειμενοστραφή προγραμματισμό βασίζονται, κατά κύριο λόγο, στα ρομπότ Karel (Pattis et al., 1995) και Karel++ (Bergin et al., 1997). Τέτοιοι μικρόκοσμοι είναι οι JKarelRobot (Buck & Stucki, 2000), Jeroo (Sanders & Dorn, 2003) και ο objectKarel (Ξυνόγαλος, 2002). Και οι τρεις μικρόκοσμοι αποτελούν ολοκληρωμένα προγραμματιστικά περιβάλλοντα και παρέχουν τη δυνατότητα της *βήμα προς βήμα εκτέλεσης*.

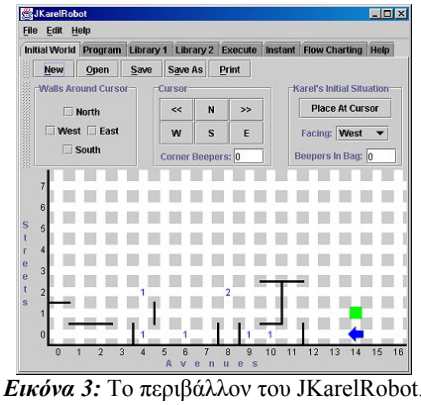
Το πλεονέκτημα του μικρόκοσμου ***Jeroo*** (http://info.nwmissouri.edu/~sanders/Jeroo/Jeroo.html) έγκειται στο γεγονός ότι χρησιμοποιεί ένα μόνο παράθυρο (Εικόνα 2). Η γλώσσα προγραμματισμού που χρησιμοποιεί έχει σύνταξη ανάλογη με εκείνη της C++ και της Java και υποστηρίζει την υλοποίηση *αναδρομικών μεθόδων*. Ωστόσο υπάρχουν αρκετοί περιορισμοί: (1) υπάρχει μία μόνο κλάση από την οποία μπορούν να δημιουργηθούν μόνο τέσσερα αντικείμενα, (2) δεν υποστηρίζεται η κληρονομικότητα, (3) ο σπουδαστής μπορεί να επεκτείνει την υπάρχουσα κλάση ορίζοντας void μεθόδους, όχι όμως και κατηγορήματα.



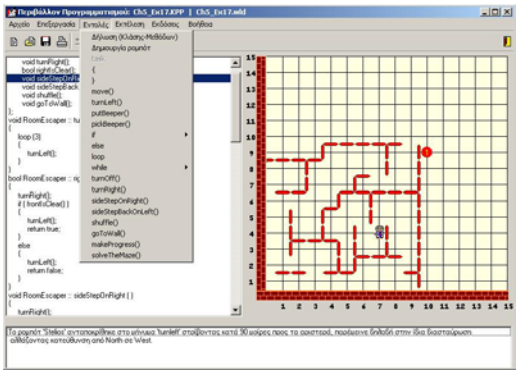
Εικόνα 2: Το παράθυρο του Jeroo.

Το βασικό πλεονέκτημα του προγραμματιστικού περιβάλλοντος ***JKarelRobot*** (http://math.otterbein.edu/JkarelRobot), το οποίο βασίζεται στο ρομπότ Karel, έγκειται στο

γεγονός ότι υποστηρίζει τρεις διαφορετικές γλώσσες προγραμματισμού: *Pascal*, *Java* και *Lisp*. Επίσης, ο JKarelRobot (Εικόνα 3) παρέχει τη δυνατότητα δημιουργίας *διαγραμμάτων ροής* για υπάρχοντα προγράμματα. Το μειονέκτημά του, όσον αφορά στον αντικειμενοστραφή προγραμματισμό, έγκειται στο γεγονός ότι επιτρέπει τη δημιουργία ενός μόνο ρομπότ (αντικειμένου) της κλάσης που δημιουργεί ο σπουδαστής, με αποτέλεσμα να ενισχύεται η δυσκολία διαχωρισμού των εννοιών της κλάσης και του αντικειμένου.



Εικόνα 3: Το περιβάλλον του JKarelRobot.



Εικόνα 4: Το περιβάλλον του objectKarel.

Ο ***objectKarel*** (Ξυνόγαλος, 2002), ο οποίος βασίζεται στο ρομπότ Karel++ και χρησιμοποιεί μια γλώσσα που μοιάζει με τη C++ και τη Java, ενσωματώνει μια σειρά *μαθημάτων* στα ελληνικά (και αγγλικά) και *δραστηριοτήτων* για την εξοικείωση του σπουδαστή με τις βασικές έννοιες του αντικειμενοστραφούς προγραμματισμού και τις βασικές δομές. Η ανάπτυξη των προγραμμάτων πραγματοποιείται χρησιμοποιώντας έναν *εκδότη δομής* (Εικόνα 4), δηλαδή μέσω ενός μενού και πλαισίων διαλόγου. Τέλος, ο objectKarel ενσωματώνει τη δυνατότητα της *επεξηγηματικής οπτικοποίησης*, την εμφάνιση δηλαδή επεξηγήσεων της τρέχουσας εντολής στα ελληνικά.

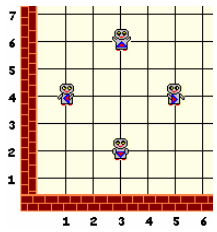
Η ΧΡΗΣΗ ΤΟΥ objectKarel ΣΤΗΝ ΤΑΞΗ

Στην ενότητα αυτή παρουσιάζονται τα συμπεράσματα από την πιλοτική εφαρμογή του objectKarel – σε 19 φοιτητές του τμήματος Εφαρμοσμένης Πληροφορικής - όσον αφορά στα προβλήματα που παρουσιάζονται κατά τη διδασκαλία του αντικειμενοστραφούς προγραμματισμού και τις δυσκολίες των σπουδαστών που παρουσιάστηκαν στις προηγούμενες ενότητες.

Δυνατότητα διδασκαλίας του παραδείγματος σε σύντομο χρονικό διάστημα

Από την πιλοτική εφαρμογή και αξιολόγηση του objectKarel έγινε εμφανές ότι η διδασκαλία των βασικών αρχών του αντικειμενοστραφούς προγραμματισμού μπορεί να επιτευχθεί σε πολύ σύντομο χρονικό διάστημα, σε αντίθεση με τα επαγγελματικά προγραμματιστικά περιβάλλοντα. Συγκεκριμένα, η διδασκαλία των εννοιών του αντικειμενοστραφούς προγραμματισμού πραγματοποιήθηκε σε 3 μαθήματα, ενώ στα 2 επόμενα μαθήματα πραγματοποιήθηκε η διδασκαλία των βασικών δομών επιλογής και επανάληψης. Τα 5 μαθήματα, διάρκειας 2 διδακτικών ωρών το κάθε ένα, πραγματοποιήθηκαν στο εργαστήριο. Σε κάθε ένα από τα αυτά χρησιμοποιήθηκε η θεωρία και οι δραστηριότητες που έχουν ενσωματωθεί στο περιβάλλον του objectKarel, ενώ στη συνέχεια οι φοιτητές έλυσαν σε ομάδες μία ή περισσότερες ασκήσεις, οι οποίες σχεδιάστηκαν έτσι ώστε να διαπιστώσουμε αν οι δυσκολίες που έχουν καταγραφεί στη διεθνή βιβλιογραφία παρουσιάζονται και στο περιβάλλον του objectKarel. Για παράδειγμα,

προκειμένου να διαπιστώσουμε αν οι φοιτητές δυσκολεύονται να διαχωρίσουν τις κλάσεις από τα στιγμιότυπά τους (αντικείμενα) τους ανατέθηκε η παρακάτω άσκηση:



*Να δημιουργήσετε μια νέα κλάση με όνομα **AugmentedRobot**, τα ρομπότ της οποίας θα είναι ικανά να ανταποκρίνονται στα μηνύματα:*

- ***turnRight**: στρίβοντας δεξιά κατά 90 μοίρες*
- ***turnAround**: κάνοντας στροφή 180 μοιρών*

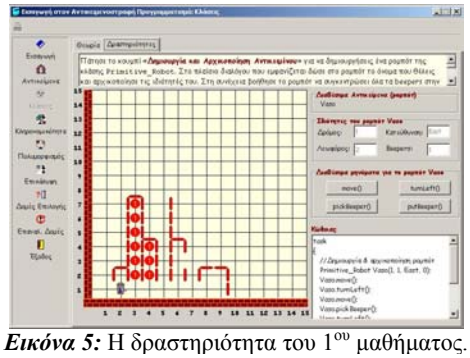
Στη συνέχεια να δημιουργήσετε τέσσερα ρομπότ, όπως φαίνεται στην διπλανή Εικόνα, και να στείλετε σε κάθε ένα από αυτά το μήνυμα turnRight. (Σημείωση: Η βασική κλάση Robot περιλαμβάνει τη μέθοδο turnLeft για στροφή προς τα αριστερά κατά 90 μοίρες.)

Για την παραπάνω άσκηση, όπως και για όλες τις ασκήσεις που ανατέθηκαν, χρησιμοποιήθηκε η *δυνατότητα της καταγραφής των ενεργειών* των σπουδαστών που έχει ενσωματωθεί στο περιβάλλον του objectKarel. Η μελέτη των διαδοχικών εκδόσεων των προγραμμάτων που ανάπτυξαν οι φοιτητές έδειξε ότι μία μόνο από τις 10 ομάδες αντιμετώπισε δυσκολία με το διαχωρισμό των εννοιών κλάσης και αντικειμένου, αφού πρότεινε τη δημιουργία 4 κλάσεων (μία για κάθε ρομπότ-αντικείμενο). Λεπτομερής περιγραφή των μαθημάτων, των διδακτικών στόχων και των συμπερασμάτων από τη μελέτη των ασκήσεων και της αξιολόγησης που ακολούθησε μετά το πέρας των 5 μαθημάτων δίνεται στο (Ξυνόγαλος, 2002), ενώ μια συνοπτική περιγραφή δίνεται στο (Ξυνόγαλος, 2004). Στον Πίνακα 1 παρουσιάζεται το περιεχόμενο των 5 μαθημάτων.

Μάθημα	**Έννοιες**
Αντικείμενα-κλάσεις	αντικείμενο, δημιουργία & αρχικοποίηση - χαρακτηριστικό & συμπεριφορά αντικειμένου, μήνυμα, μέθοδος, κλάση
Κληρονομικότητα	κληρονομικότητα, γονική κλάση, scope resolution operator, δήλωση κλάσης, ορισμός μεθόδου, επαναχρησιμοποιησημότητα
Πολυμορφισμός-επικάλυψη	πολυμορφισμός, επικάλυψη
Δομές επιλογής	if, if/else, εμφωλευμένες δομές, εντολή return, κατηγορήματα
Δομές επανάληψης	while, loop, εμφωλευμένες δομές, επιλογή κατάλληλων δομών (επιλογής ή/και επανάληψης) για ένα δεδομένο πρόβλημα

Πίνακας 1: Το περιεχόμενο των 5 μαθημάτων.

Οι σπουδαστές παρακινούνται μέσω της χρήσης εννοιών, πριν την υλοποίησή τους



Εικόνα 5: Η δραστηριότητα του 1ου μαθήματος.

Η παρουσίαση των εννοιών του αντικειμενοστραφούς προγραμματισμού δεν πραγματοποιείται, όπως συνήθως, παρέχοντας απλά κειμενικές και προφορικές περιγραφές. Στο περιβάλλον του objectKarel έχουν ενσωματωθεί *μαθησιακές δραστηριότητες* για την εξοικείωση των σπουδαστών με τις βασικές έννοιες πριν να ξεκινήσουν να αναπτύσσουν προγράμματα. Για παράδειγμα, στα πλαίσια της δραστηριότητας του 1ου μαθήματος που φαίνεται στην Εικόνα 5, οι φοιτητές στέλνουν στο ρομπότ τα κατάλληλα μηνύματα - προκειμένου να συγκεντρώσει τα αντικείμενα που υπάρχουν στον κόσμο του - πατώντας κουμπιά και όχι γράφοντας κώδικα, ενώ ταυτόχρονα παρατηρούν πως ανταποκρίνεται το ρομπότ, πως μεταβάλλονται οι τιμές των ιδιοτήτων του, καθώς επίσης και τη σύνταξη των ενεργειών τους στη γλώσσα προγραμματισμού. Οι στόχοι της δραστηριότητας αυτής είναι η

παρουσίαση και εξοικείωση με τις εξής έννοιες: δημιουργία και αρχικοποίηση των ιδιοτήτων ενός αντικειμένου, παρουσίαση του μηχανισμού αποστολής μηνυμάτων σε ένα αντικείμενο και του τρόπου ανταπόκρισής του σε αυτά, ένα αντικείμενο ανταποκρίνεται διαφορετικά σε ένα μήνυμα ανάλογα με την κατάστασή του, η κατάσταση ενός αντικειμένου μεταβάλλεται μέσω της εκτέλεσης των μεθόδων, παρουσίαση της σύνταξης της γλώσσας προγραμματισμού.

Οι σπουδαστές μαθαίνουν να είναι τόσο «καταναλωτές» όσο και «παραγωγοί» κώδικα που μπορεί να επαναχρησιμοποιηθεί

Στα πλαίσια των 5 μαθημάτων που πραγματοποιήθηκαν δόθηκε ιδιαίτερη βαρύτητα στην έννοια της επαναχρησιμοποιησιμότητας (reusability). Οι φοιτητές κλήθηκαν να μελετήσουν και να χρησιμοποιήσουν υπάρχουσες κλάσεις για να επιλύσουν συγκεκριμένα προβλήματα, να τροποιήσουν/επεκτείνουν κλάσεις, καθώς επίσης και να σχεδιάσουν και να υλοποιήσουν νέες κλάσεις. Στην πραγματικότητα, οι φοιτητές σχεδίασαν και υλοποίησαν σωστές κλάσεις για όλα σχεδόν τα προβλήματα που τους ανατέθηκαν. Επιπλέον, οι φοιτητές επαναχρησιμοποίησαν, εκτός από τις υπάρχουσες κλάσεις, και τις κλάσεις που δημιούργησαν οι ίδιοι.

Οι σπουδαστές δεν αντιμετώπισαν δυσκολίες με τις αντικειμενοστραφείς έννοιες

Από την ανάλυση των δεδομένων που συγκεντρώθηκαν στα 5 μαθήματα και το ερωτηματολόγιο που συμπλήρωσαν οι φοιτητές μετά από αυτά, προέκυψε ότι οι φοιτητές κατανόησαν τις βασικές έννοιες του αντικειμενοστραφούς προγραμματισμού (δες Πίνακα 1). Επιπλέον, οι ακόλουθες δυσκολίες που έχουν καταγραφεί στη βιβλιογραφία δεν παρουσιάστηκαν στο περιβάλλον του objectKarel: (1) ταύτιση αντικειμένου/κλάσης, (2) τα αντικείμενα είναι απλές εγγραφές χωρίς συμπεριφορά, (3) μέθοδοι διαφορετικών κλάσεων μπορούν να έχουν το ίδιο όνομα μόνο αν έχουν διαφορετικές υπογραφές.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η διδασκαλία και εκμάθηση του αντικειμενοστραφούς προγραμματισμού είναι γεγονός ότι παρουσιάζει αρκετές δυσκολίες. Το πρόβλημα πιστεύουμε ότι έγκειται, κατά κύριο λόγο, στην έλλειψη εκπαιδευτικών εργαλείων και διδακτικής εμπειρίας/γνώσης για τον αντικειμενοστραφή προγραμματισμό, στη χρησιμοποιούμενη μεθοδολογία διδασκαλίας και στη μη αξιοποίηση των αποτελεσμάτων των ερευνών σχετικών με τις δυσκολίες των αρχάριων στα πλαίσια του αντικειμενοστραφούς προγραμματισμού. Τα προβλήματα στη διδασκαλία του αντικειμενοστραφούς προγραμματισμού και οι δυσκολίες των σπουδαστών μπορούν να αντιμετωπιστούν διαμορφώνοντας κατάλληλες διδακτικές καταστάσεις. Η εισαγωγή στον αντικειμενοστραφή προγραμματισμό με ένα μικρόκοσμο, όπως προέκυψε και από την πιλοτική εφαρμογή του objectKarel, μπορεί να συμβάλει ουσιαστικά στην αντιμετώπιση των καταγεγραμμένων δυσκολιών, στην κατανόηση των βασικών εννοιών του αντικειμενοστραφούς προγραμματισμού και στην απόκτηση ενός ισχυρού θεωρητικού υπόβαθρου. Βέβαια, μετά από αυτή την εισαγωγή στην αντικειμενοστρέφεια, θα πρέπει αναπόφευκτα να ακολουθήσει η χρήση μιας συμβατικής γλώσσας προγραμματισμού. Η μετάβαση αυτή είναι σίγουρο ότι δεν θα είναι εύκολη. Οι σπουδαστές είναι σχεδόν σίγουρο ότι θα αντιμετωπίσουν δυσκολίες με τη σύνταξη της γλώσσας προγραμματισμού, με την έλλειψη του εκδότη δομής και γενικότερα με την πολυπλοκότητα του περιβάλλοντος. Για την αντιμετώπιση των δυσκολιών που σχετίζονται με το περιβάλλον μπορεί να χρησιμοποιηθεί ένα εκπαιδευτικό προγραμματιστικό περιβάλλον, όπως το BlueJ. Σε κάθε περίπτωση όμως, η διδασκαλία των εννοιών του αντικειμενοστραφούς προγραμματισμού πρέπει να γίνεται σταδιακά και με τη χρήση μαθησιακών δραστηριοτήτων που θα έχουν ως στόχο την κατανόηση των εννοιών πριν από την υλοποίησή τους. Τέλος, είναι απαραίτητη η παρουσίαση παραδειγμάτων και η ανάθεση ασκήσεων ειδικά σχεδιασμένων ώστε να αποφεύγονται οι ήδη καταγεγραμμένες δυσκολίες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Bergin, J., Stehlik, M., Roberts, J., & Pattis, R. (1997), Karel++ - A Gentle Introduction to the Art of Object-Oriented Programming, 2nd edn., New York: John Wiley and Sons.

Berman, A.M., Decker, R., Nguyen, D.X., Reid, R.J. & Wallingford, E. (1994), Using C++ in CS1/CS2, ACM SIGCSE Bulletin, Vol. 26, No. 1, 383-384.

Brilliant, S. & Wiseman, T. R. (1996), The First Programming Paradigm and Language Dilemma, *ACM SIGCSE Bulletin, Vol. 28, Issue 1*, 338-342.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A. & Miller P. (1997), Mini-languages: a way to learn programming principles, *Education and Information Technologies 2*, 65-83.

Buck, D., & Stucki, D.J. (2000), JKarelRobot: A Case Study in Supporting Levels of Cognitive Development in the Computer Science Curriculum, *ACM SIGCSE Bulletin, 33(1)*, 16-20.

Carter, J. & Fowler, A. (1998), Object Oriented Students?, *SIGCSE Bulletin, Vol. 28, No. 3,* 271.

Decker, R. & Hirshfield, S. (1994), The Top 10 Reasons Why Object-Oriented Programming Can't Be Taught In CS1, *ACM SIGCSE Bulletin, Vol. 26, No. 1*, 51-55.

Fleury, A. E. (2000), Programming in Java: student-constructed rules, *ACM SIGCSE Bulletin, Vol. 32, Issue 1*, 197-201.

Fleury, A. E. (2001), Encapsualtion and reuse as viewed by java students, *ACM SIGCSE Bulletin, Vol. 33, Issue 1*, 189-193.

Hadjerrouit, S. (1998), A Constructivist Framework for Integrating the Java Paradigm into the Undergraduate Curriculum, *ACM SIGCSE Bulletin, Vol. 30, Issue 3*, 105-107.

Hadjerrouit, S. (1999), A constructivist approach to object-oriented design and programming, *ACM SIGCSE Bulletin, Vol. 31, Issue 3*, 171-174.

Holland, S. Griffiths, R. & Woodman, M. (1997), Avoiding object misconceptions, *ACM SIGCSE Bulletin, Vol. 29, No. 1*, 131-134.

Kolling, M. & Rosenberg, J. (2001), Guidelines for Teaching Object Orientation with Java, *ACM SIGCSE Bulletin, Vol. 33 Issue 3*, 33-36.

Kolling, M., Koch, B. & Rosenberg, J. (1995), Requirements for a First Year Object-Oriented *Teaching Language, ACM SIGCSE Bulletin, Vol. 27, No. 1, 173-177.*

Kolling, M., Quig, B., Patterson, A., & Rosenberg, J. (2003), The BlueJ system and its pedagogy, *Journal of Computer Science Education, 13(4),* 249-268.

Pattis, R. E., Roberts, J. & Stehlik, M. (1995), Karel - The Robot, A Gentle Introduction to the Art of Programming, *2nd edn. New York, Wiley*.

Ramalingam, V. & Wiedenbeck, S. (1997), An Empirical Study of Novice Program Comprehension in the Imperative and Object-Oriented Styles, *Empirical Studies of Programmers: Seventh Workshop, ACM Press, New York*, 124-139.

Sanders, D., & Dorn, B. (2003), Jeroo: A Tool for Introducing Object-Oriented Programming, *ACM SIGCSE Bulletin, 35(1)*, 201-204.

Tempte, M C. (1991), Let's Begin Introducing the Object-Oriented Paradigm, *ACM SIGCSE Bulletin, Vol. 23, No. I*, 338-342.

Wick, M. (1995), On Using C++ and Object-Orientation in CS1: the Message is still more important than the Medium, *ACM SIGCSE Bulletin, Vol. 27, Issue 1*, 322-326.

Ξυνόγαλος, Σ. (2002), Εκπαιδευτική Τεχνολογία: Ένας Διδακτικός Μικρόκοσμος για την Εισαγωγή στον Αντικειμενοστραφή Προγραμματισμό, *Διδακτορική διατριβή, Τμήμα Εφ. Πληροφορικής Πανεπιστήμιο Μακεδονίας.*

Ξυνόγαλος, Σ. (2004), Πιλοτική Εφαρμογή και Αξιολόγηση του Προγραμματιστικού Περιβάλλοντος objectKarel, *Πρακτικά του 2ου Πανελλήνιου Συνεδρίου «Πληροφορική και Εκπαίδευση» του Σ.Ε.Π.Δ.Ε.Θ.* (υπό έκδοση), Θεσσαλονίκη, 20-22 Φεβρουαρίου 2004.

Ξυνόγαλος, Σ., Σατρατζέμη, Μ. & Δαγδιλέλης, Β. (2000), Η εισαγωγή στον προγραμματισμό: Διδακτικές Προσεγγίσεις και Εκπαιδευτικά Εργαλεία, *2ο Πανελλήνιο Συνέδριο «Οι Τεχνολογίες της Πληροφορίας και της Επικοινωνίας στην Εκπαίδευση*, Πάτρα, 13-15 Οκτωβρίου, 115-124.