Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση

Τόμ. 1 (2008)

4ο Συνέδριο Διδακτική Πληροφορικής

Μελέτη των Δυσκολιών των Φοιτητών για την Έννοια του «Αντικειμένου» στον Αντικειμενοστραφή Προγραμματισμό

Στ. Ξυνόγαλος

Βιβλιογραφική αναφορά:

Ξυνόγαλος Σ. (2023). Μελέτη των Δυσκολιών των Φοιτητών για την Έννοια του «Αντικειμένου» στον Αντικειμενοστραφή Προγραμματισμό . *Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση*, *1*, 091–100. ανακτήθηκε από https://eproceedings.epublishing.ekt.gr/index.php/cetpe/article/view/5854

Μελέτη των Δυσκολιών των Φοιτητών για την Έννοια του «Αντικειμένου» στον Αντικειμενοστραφή Προγραμματισμό

Στ. Ξυνόγαλος

Τμήμα Διοίκησης Τεχνολογίας
Πανεπιστήμιο Μακεδονίας
stelios@uom.gr

Περίληψη

Η διδασκαλία και η εκμάθηση των αρχών του αντικειμενοστραφούς προγραμματισμού, όπως προκύπτει από σχετικές μελέτες, παρουσιάζει αρκετές δυσκολίες. Στην παρούσα εργασία παρουσιάζονται οι δυσκολίες προπτυχιακών φοιτητών που αφορούν σε μια θεμελιώδη έννοια του αντικειμενοστραφούς προγραμματισμού – την έννοια του αντικειμένου. Τα αποτελέσματα βασίζονται σε μια γραπτή εξέταση που πραγματοποιήθηκε στο πλαίσιο ενός μαθήματος αντικειμενοστραφούς προγραμματισμού που χρησιμοποιεί το εκπαιδευτικό προγραμματιστικό περιβάλλον BlueJ. Η διδακτική προσέγγιση που υιοθετήθηκε είναι αυτή των δημιουργών του περιβάλλοντος, με ορισμένες ουσιαστικές παρεμβάσεις που έγιναν μετά την πρώτη εφαρμογή και αξιολόγηση της διδασκαλίας.

Λέξεις κλειδιά: *αντικειμενοστραφής προγραμματισμός, εκπαιδευτικό προγραμματιστικό περιβάλλον, διδακτική προσέγγιση.*

Abstract

Teaching and learning object-oriented programming (OOP), according to relevant studies, is accompanied with several difficulties. In this paper we present the difficulties of undergraduate students that refer to a fundamental concept for OOP – the concept of object. The results are based on a written exam that took place in the context of an OOP course that uses the educational programming environment of BlueJ. The didactic approach adopted was that of the creators of the environment, with some important interventions which took place after the first application and evaluation of the teaching.

Keywords: *object-oriented programming, educational programming environment, didactic approach*.

1. Εισαγωγή

Οι δυσκολίες που αντιμετωπίζουν οι σπουδαστές κατά την εισαγωγή τους στον αντικειμενοστραφή προγραμματισμό έχουν αποτελέσει αντικείμενο μελέτης πολλών ερευνητών (Carter & Fowler, 1998; Fleury, 2000; Hristova, 2003; Topor, 2002; Truong et al., 2004; Ziring, 2001). Ορισμένες από τις δυσκολίες αυτές φαίνεται να είναι εγγενείς στην εκμάθηση των αρχών του αντικειμενοστραφούς προγραμματισμού, ενώ άλλες δεν έχουν τόσο έντονο χαρακτήρα. Ο βαθμός

εμφάνισης τους μάλιστα σε μια διδασκαλία αποτελεί σε κάποιο βαθμό μέτρο αξιολόγησης της διδακτικής αποτελεσματικότητας της. Μια δημοφιλής διδακτική προσέγγιση που φαίνεται να υιοθετούν αρκετοί διδάσκοντες είναι εκείνη που βασίζεται στο εκπαιδευτικό προγραμματιστικό περιβάλλον BlueJ (Kölling et al., 2003), τις διδακτικές οδηγίες που έχουν διατυπώσει οι δημιουργοί του (Kölling & Rosenberg, 2001) και - σε ορισμένες περιπτώσεις - το συνοδευτικό βιβλίο *"Objects First with Java: A practical introduction using BlueJ"* (Barnes & Kölling, 2004).

Στην παρούσα εργασία παρουσιάζονται τα αποτελέσματα ενός μαθήματος εισαγωγής στον αντικειμενοστραφή προγραμματισμό που βασίζεται στη σειρά μαθημάτων που προτείνουν οι Barnes και Kölling στο βιβλίο τους (Barnes & Kölling, 2004) και στο εκπαιδευτικό προγραμματιστικό περιβάλλον BlueJ. Συγκεκριμένα, παρουσιάζονται οι δυσκολίες των φοιτητών που σχετίζονται με τη θεμελιώδη έννοια του αντικειμένου σε συνδυασμό με σχετικά αποτελέσματα που καταγράφονται στη βιβλιογραφία.

2. Περιγραφή της διδακτικής προσέγγισης

Το περιβάλλον BlueJ, αν και υποστηρίζει την ανάπτυξη οποιασδήποτε αντικειμενοστραφούς εφαρμογής σε Java, έχει έντονο εκπαιδευτικό χαρακτήρα. Χρησιμοποιώντας το αλληλεπιδραστικό ενδιάμεσο του BlueJ οι σπουδαστές μπορούν γρήγορα και εύκολα να δημιουργήσουν αντικείμενα, να καλέσουν τις μεθόδους τους μέσω ενός αναδυόμενου μενού και να ελέγξουν ανά πάσα στιγμή την κατάσταση τους χωρίς να χρειάζεται να γράψουν κώδικα. Το περιβάλλον συνοδεύεται από ένα μεγάλο πλήθος εφαρμογών που μπορούν να χρησιμοποιήσουν οι σπουδαστές. Ο μηχανισμός αυτός επιτρέπει στον διδάσκοντα να καθυστερήσει την παρουσίαση άλλων τεχνολογιών ενδιαμέσων και να πραγματοποιήσει μια διδασκαλία που θα βασίζεται πραγματικά σε μια προσέγγιση «πρώτα-τα-αντικείμενα».

Η προσέγγιση αυτή έχει υιοθετηθεί και στο βιβλίο *"Objects First with Java: A practical introduction using BlueJ"* (Barnes & Kölling, 2004), το οποίο άλλωστε επιτάσσει τη χρήση του σε συνδυασμό με το περιβάλλον BlueJ. Η εν λόγω διδακτική προσέγγιση μπορεί να συνοψιστεί ως εξής:

"Πρώτα-τα-αντικείμενα": ο σπουδαστής δημιουργεί αντικείμενα και καλεί τις μεθόδους τους από το πρώτο μάθημα.
Επαναληπτική προσέγγιση: επαναλαμβανόμενη παρουσίαση εννοιών.
Παρουσίαση των αρχών του αντικειμενοστραφούς προγραμματισμού γενικά και όχι λεπτομέρειες της γλώσσας Java ειδικά.
Οργάνωση της ύλης βάσει των θεμελιωδών εργασιών ανάπτυξης μιας αντικειμενοστραφούς εφαρμογής και όχι των δομών της γλώσσας Java.
Καθοδηγούμενη από έτοιμα project προσέγγιση.
Η διδακτική προσέγγιση στην οποία βασίζεται το βιβλίο, εκτός από την προσέγγιση «πρώτα-τα αντικείμενα» υιοθετεί στοιχεία και από την προσέγγιση «πρώτα-το μοντέλο» (model-first) (Bennedsen & Caspersen, 2004; Georgantaki & Retalis,

2007), αφού η ύλη οργανώνεται βάσει των θεμελιωδών εργασιών ανάπτυξης μιας εφαρμογής που βασίζεται στο αντικειμενοστραφές μοντέλο και όχι στις δομές της γλώσσας. Επίσης, στο βιβλίο και κυρίως στη διδασκαλία που περιγράφεται στην παρούσα εργασία ακολουθείται η εξής διαδικασία για την παρουσίαση νέων εννοιών: (1) περιγράφεται ένα πρόβλημα που για τη λύση του απαιτεί τη δημιουργία ενός μοντέλου κάποιου υπαρκτού, συνήθως, συστήματος του κόσμου, (2) παρουσιάζονται βασικές έννοιες από τη περιοχή του προβλήματος και οι συσχετίσεις τους, (3) δημιουργείται ένα νοητό μοντέλο του συστήματος και αναγνωρίζονται τα τμήματα από τα οποία είναι δομημένο το μοντέλο και κατ' επέκταση τα αντικείμενα που απαρτίζουν το πεδίο ορισμού του προβλήματος, (4) το προαναφερθέν νοητό μοντέλο παρουσιάζεται με τη μορφή ενός απλοποιημένου διαγράμματος UML όπου φαίνονται οι απαιτούμενες κλάσεις και οι συσχετίσεις τους, (5) υλοποιούνται οι κλάσεις, εισάγοντας - στα πλαίσια της ανάγκης υλοποίησης του μοντέλου - νέες έννοιες και δομές, (6) παρέχονται σχεδιαστικά πρότυπα και πρότυπα κώδικα για την εφαρμογή τους σε αντίστοιχες περιπτώσεις.

Το περιβάλλον BlueJ και η σειρά μαθημάτων που προτείνεται στο προαναφερθέν βιβλίο χρησιμοποιούνται εδώ και τρία χρόνια για τη διδασκαλία ενός μαθήματος αντικειμενοστραφούς προγραμματισμού σε φοιτητές του Τμήματος Διοίκησης Τεχνολογίας του Πανεπιστημίου Μακεδονίας. Η διδασκαλία που πραγματοποιήθηκε την πρώτη χρονιά βασίστηκε στη διδακτική προσέγγιση που προτείνουν οι συγγραφείς του βιβλίου και δημιουργοί του περιβάλλοντος BlueJ. Η αξιολόγηση του μαθήματος έδειξε ότι η συγκεκριμένη διδακτική προσέγγιση αποτελεί μια καλή πρόταση για την εισαγωγή στον αντικειμενοστραφή προγραμματισμό. Ωστόσο, δεν είναι απαλλαγμένη από δυσκολίες και προβλήματα, κάποια από τα οποία θεωρούμε ότι οφείλονται στην προτεινόμενη διδακτική προσέγγιση (Xinogalos, Satratzemi & Dagdilelis, 2006). Το συγκεκριμένο μάθημα αναδιοργανώθηκε και διδάχθηκε για δεύτερη χρονιά κάνοντας κάποιες παρεμβάσεις στη διδακτική προσέγγιση των Barnes & Kölling (Xinogalos, Satratzemi & Dagdilelis, 2007): (1) *Δημιουργία αντικειμένων και κλήση μεθόδων γράφοντας κώδικα (χρήση της main)* από το 4ο μάθημα αντί για τα τελευταία μαθήματα, (2) *Χρήση των χαρακτηριστικών του BlueJ με μεγαλύτερη προσοχή και όχι καταχρηστικά,* (3) *Σχεδίαση διδακτικών καταστάσεων και εργασιών βάσει των ευρημάτων της αξιολόγησης της πρώτης εφαρμογής των μαθημάτων*. Στις επόμενες ενότητες παρουσιάζονται κάποια αποτελέσματα από αυτή την αναθεωρημένη διδασκαλία του μαθήματος.

3. Περιγραφή της μελέτης

Τα στοιχεία που παρουσιάζονται στην εργασία προέρχονται από μια εξέταση που συμμετείχε στη διαμόρφωση του τελικού βαθμού των φοιτητών και η οποία έλαβε μέρος μετά τη διεξαγωγή των έξι πρώτων μαθημάτων: (1ο) *Εισαγωγικές έννοιες*: αντικείμενα, κλάσεις, μέθοδοι, το περιβάλλον BlueJ, (2ο) *Ορισμός κλάσεων*: πεδία, κατασκευαστές, μέθοδοι, (3ο) *Αλληλεπίδραση αντικειμένων*: δημιουργία αντικειμένων

από αντικείμενα, πολλαπλοί κατασκευαστές, (4ο) *Στατικές μέθοδοι*: η μέθοδος main, εκτέλεση χωρίς το BlueJ, (5ο & 6ο) *Ομαδοποίηση αντικειμένων*: συλλογές ευέλικτου μεγέθους (`ArrayList`) και συλλογές προκαθορισμένου μεγέθους (πίνακες).

Στα πλαίσια της γραπτής εξέτασης που πραγματοποιήθηκε, δόθηκε στους φοιτητές ο σκελετός δύο κλάσεων: της κλάσης `Candidate` για την αναπαράσταση υποψηφίων για τα πτυχία αγγλικών Cambridge First Certificate και Certificate of Proficiency, και της κλάσης `ExaminationBook`, για την ομαδοποίηση των υποψηφίων για τα προαναφερθέντα πτυχία σε δύο διαφορετικές ομάδες. Τους ζητήθηκε να ορίσουν κατασκευαστές και μεθόδους για την υλοποίηση της συμπεριφοράς των αντικειμένων των κλάσεων, να δημιουργήσουν αντικείμενα και να καλέσουν τις μεθόδους τους.

```
public class Candidate
{
    private String name;
    private String title;
    private int money; . . .
}
```

```
public class ExaminationBook
{
   private ArrayList FCE;
   private ArrayList PCE;
   . . .
}
```

Λόγω της περιορισμένης έκτασης της εργασίας, θα παρουσιάσουμε μόνο τα αποτελέσματα που σχετίζονται άμεσα με την έννοια του αντικειμένου, όπως πολλαπλοί κατασκευαστές, δημιουργία αντικειμένου, και προσπέλαση των πεδίων ενός αντικειμένου.

4. Τα αποτελέσματα της μελέτης

4.1 Πολλαπλοί κατασκευαστές

Στα πλαίσια της γραπτής εξέτασης ζητήθηκε από τους φοιτητές να ορίσουν στην κλάση `Candidate` δύο κατασκευαστές, από τους οποίους: ο πρώτος θα δέχεται μέσω μιας παραμέτρου με όνομα `name` τιμή για το πεδίο `name` και θα αρχικοποιεί κατάλληλα τα άλλα πεδία, και ο δεύτερος θα δέχεται μέσω παραμέτρων τιμές για όλα τα πεδία. Βασικός στόχος του ερωτήματος αυτού ήταν η διερεύνηση του βαθμού στον οποίο εμφανίζονται στη συγκεκριμένη διδασκαλία δυσκολίες και παρανοήσεις που έχουν καταγραφεί στη βιβλιογραφία σχετικά με τους κατασκευαστές:

Η χρήση πολλαπλών κατασκευαστών προκαλεί σύγχυση στους φοιτητές (Carter & Fowler, 1998).
Οι αρχάριοι προγραμματιστές συχνά δηλώνουν ένα κατασκευαστή ως μέθοδο (Ziring, 2001).
Συχνά, οι αρχάριοι πραγματοποιούν υπολογισμούς στον κατασκευαστή μιας κλάσης (Topor, 2002).
Συγχέονται μεταβλητές στιγμιοτύπου και τοπικές μεταβλητές (Truong, 2004) και επισκιάζεται μια μεταβλητή στιγμιοτύπου με τη δήλωση σε μια μέθοδο της κλάσης μιας τοπικής μεταβλητής που έχει το ίδιο όνομα (Ziring, 2001).

Στον Πίνακα 1 παρουσιάζονται τα στοιχεία που αφορούν στη δήλωση των πολλαπλών κατασκευαστών και στον Πίνακα 2 τα λάθη των φοιτητών σε κάθε ένα από τους δύο κατασκευαστές. Σε όλους τους πίνακες τα ποσοστά των σωστών απαντήσεων και των διαφόρων λαθών αναφέρονται στον αριθμό των φοιτητών που απάντησαν στη συγκεκριμένη ερώτηση και όχι στο σύνολο των 64 φοιτητών που συμμετείχαν στην εξέταση.

Πίνακας 1: *Ορισμός πολλαπλών κατασκευαστών.*

Δεν ορίστηκε κανένας κατασκευαστής	11%
Ορίστηκαν σωστά και οι δύο κατασκευαστές	32%
Ορίστηκε σωστά ο ένας από τους δύο κατασκευαστές	30%
Ορίστηκε (είτε σωστά είτε λανθασμένα) ο ένας μόνο από τους δύο	16%
Δόθηκε λάθος όνομα στον ένα από τους δύο κατασκευαστές	7%
Δόθηκε λάθος όνομα και στους δύο κατασκευαστές	2%

Πίνακας 2: *Λάθη στον ορισμό πολλαπλών κατασκευαστών.*

	Κατασκευαστής με χρήση του `this`	Κατασκευαστής με πέρασμα παραμέτρων
Δεν ορίστηκε	9%	2%
Σωστός	35%	58%
Λάθη		
Παράμετρος	21%	
- Διαφορετικό όνομα από αυτό που ζητήθηκε	19%	
- Λείπει η παράμετρος	2%	
Αρχικοποίηση πεδίων/χαρακτηριστικών	44%	33%
- Ανάθεση των τιμών μη ορισμένων αναγνωριστικών	11%	7%
- Ανάθεση μιας λανθασμένης σταθερής τιμής	18%	
- Αν και χρησιμοποιούνται παράμετροι στα πεδία ανατίθενται σταθερές τιμές		5%

Οι σημαντικότερες δυσκολίες που αφορούν στον ορισμό των κατασκευαστών είναι:

Αρκετοί φοιτητές δυσκολεύονται να ορίσουν δύο ή και περισσότερους κατασκευαστές στην ίδια κλάση και ορίζουν ένα μόνο κατασκευαστή ή δεν δίνουν το ίδιο όνομα στους κατασκευαστές όπως απαιτείται (Πίνακας 1).

Η χρήση του `this` δυσκολεύει τους φοιτητές. Ωστόσο, στον 1^{ο} κατασκευαστή που απαιτούσε τη χρήση του, οι φοιτητές δεν επισκίασαν τη μεταβλητή στιγμιοτύπου/πεδίο (Truong, 2004; Ziring, 2001), αλλά επέλεξαν να δώσουν άλλο όνομα στην παράμετρο προκειμένου να αποφύγουν τη χρήση του `this`, ή

θεωρώντας ότι δεν μπορεί να δηλωθεί παράμετρος που έχει το ίδιο όνομα με ένα πεδίο (Πίνακας 2).
Δυσκολίες αντιμετωπίζουν οι φοιτητές και με την αρχικοποίηση των πεδίων ενός αντικειμένου σε ένα κατασκευαστή (Πίνακας 2). Τα πιο συχνά λάθη είναι η ανάθεση των τιμών μη ορισμένων αναγνωριστικών στη περίπτωση που ζητείται ένα πεδίο να ενημερώνεται μέσω παραμέτρου και η ανάθεση λανθασμένων σταθερών τιμών στα υπόλοιπα.

4.2 Προσπέλαση των πεδίων ενός αντικειμένου

Για τη διερεύνηση των δυσκολιών που αντιμετωπίζουν οι φοιτητές με την προσπέλαση των πεδίων ενός αντικειμένου, τους ζητήθηκε: α) να ορίσουν μεθόδους *set* που θα δέχονται μέσω παραμέτρου μία τιμή και θα ενημερώνουν το αντίστοιχο πεδίο, β) να ορίσουν μεθόδους *get* για την επιστροφή των τιμών των πεδίων, γ) να προσπελάσουν από άλλη κλάση ή από τη μέθοδο `main` τα `private` πεδία αντικειμένων της κλάσης `Candidate` για να κάνουν κάποιο έλεγχο της τιμής τους, να τη χρησιμοποιήσουν σε υπολογισμούς, να την εμφανίσουν ή να την μεταβάλουν.

Σχετικές δυσκολίες που έχουν καταγραφεί στη βιβλιογραφία είναι οι εξής:

Σε μία non-void μέθοδο, η οποία θα έπρεπε να επιστρέφει μια τιμή ενός συγκεκριμένου τύπου, λείπει η εντολή `return` (Hristova, 2003).
Ένα άλλο σχετικό λάθος αφορά στη δήλωση λανθασμένου τύπου επιστρεφόμενης τιμής για μια non-void μέθοδο (Ziring, 2001).
Σύγχυση μεταξύ της δήλωσης των παραμέτρων στον ορισμό μιας μεθόδου και του περάσματος παραμέτρων στην ενεργοποίησή της (Hristova, 2003).
Κλήση μεθόδων με λάθος ορίσματα (arguments) (Hristova, 2003).
Παράλειψη των παρενθέσεων μετά την κλήση μιας μεθόδου (Hristova, 2003).
Χρήση ενός κενού μετά από την τελεία κατά την κλήση μιας συγκεκριμένης μεθόδου (Hristova, 2003).
Κλήση μιας μη-στατικής μεθόδου από τη μέθοδο `main` απευθείας και χωρίς τη δημιουργία αντικειμένου και κλήσης μέσω αυτού (Ziring, 2001).
Μια μέθοδος που επιστρέφει κάποια τιμή ενεργοποιείται ως κενή (void) μέθοδος ή ως εντολή, με αποτέλεσμα η τιμή που επιστρέφεται από τη μέθοδο να μην χρησιμοποιείται. Επίσης, συχνά υπάρχει ασυμβατότητα μεταξύ του τύπου της επιστρεφόμενης τιμής μιας μεθόδου και του τύπου της μεταβλητής στην οποία αποθηκεύεται η επιστρεφόμενη τιμή (Hristova, 2003).
Τα αποτελέσματα από τις απαντήσεις των φοιτητών παρουσιάζονται στους Πίνακες 3 και 4 και συνοψίζονται ως εξής:

Οι δυσκολίες που αντιμετωπίζουν οι φοιτητές με τις μεθόδους *set* και *get* αφορούν κατά κύριο λόγο στη δήλωση του επιστρεφόμενου τύπου, και σε μικρότερο βαθμό στη δήλωση των παραμέτρων και στη χρήση της εντολής `return` (Πίνακας 3).

Βέβαια, οι δυσκολίες αυτές μπορούν να χαρακτηριστούν ως τυπικές, αφού εμφανίζονται ανεξάρτητα από το παράδειγμα προγραμματισμού που χρησιμοποιείται.

Πίνακας 3: *Ορισμός μεθόδων set και get*

	Μέθοδοι *set*	**Μέθοδοι *get***
Δεν ορίστηκαν	19%	16%
Σωστός ορισμός	56%	72%
Λάθη		
Επιστρεφόμενος τύπος	17%	19%
- χρησιμοποιήθηκε ο τύπος της παραμέτρου	12%	
- παραλείπεται	4%	2%
- `int` σε όλες τις περιπτώσεις	2%	9%
- `void`		6%
Παράμετρος	8%	
- λείπει ο τύπος	4%	
- λείπει η παράμετρος	2%	
- χρησιμοποιείται παράμετρος χωρίς λόγο		4%
- χρησιμοποιείται το ίδιο όνομα με τη μέθοδο	2%	
Εντολή ανάθεσης	33%	
- Χρησιμοποιείται μια εντολή του τύπου: `<πεδίο>.setTitle(title1 ή <πεδίο>)`	12%	
Εντολή `return`		9%
- επιστρέφονται λάθος τιμές		6%
- λείπει		2%
- `name.get(); Money.get();`		2%

Πίνακας 4: *Προσπέλαση private πεδίων από άλλη κλάση και τη μέθοδο main*

	Προσπέλαση από άλλη κλάση	**main**
Απευθείας προσπέλαση του πεδίου χωρίς την ύπαρξη στιγμιοτύπου: *πεδίο*	41% - 47%	42%
Απευθείας προσπέλαση του πεδίου *<αντικείμενο>.πεδίο*	5% - 3%	
Προσπέλαση του ίδιου του στιγμιοτύπου αντί του πεδίου του	16% - 0%	
Λείπουν τα ορίσματα σε μεθόδους *set*		21%
Οι μέθοδοι καλούνται για τις κλάσεις και όχι για τα αντικείμενα (σύγχυση αντικειμένου-κλάσης)		6%
Χρήση του ονόματος του πεδίου που πρόκειται να ενημερωθεί ως όρισμα σε μεθόδους *set*		6%

Ένας μεγάλος αριθμός φοιτητών προσπελάζει απευθείας τα `private` πεδία ενός αντικειμένου από μια εξωτερική κλάση ή τη `main`, χωρίς στιγμιότυπο (Πίνακας 4). Η συμπεριφορά αυτή ενδεχομένως αποτελεί γενίκευση του γεγονότος ότι οι φοιτητές προσπελάζουν απευθείας τα πεδία που ορίζονται σε μια κλάση κατά την ανάπτυξη των μεθόδων της και δυσκολίας κατανόησης του γεγονότος ότι μπορούν να υπάρχουν ταυτόχρονα περισσότερα από ένα αντικείμενα της ίδιας κλάσης και δεν αρκεί να αναφέρουμε μόνο το όνομα του πεδίου που θέλουμε να προσπελάσουμε.

4.3 Δημιουργία αντικειμένων

Εκτός από τις δυσκολίες που αντιμετωπίζουν οι φοιτητές με τον ορισμό (πολλαπλών) κατασκευαστών, δυσκολίες καταγράφονται και κατά την κλήση τους για τη δημιουργία αντικειμένων. Στα πλαίσια της παρούσας μελέτης ζητήθηκε από τους φοιτητές να δημιουργήσουν αντικείμενα χρησιμοποιώντας και τους δύο κατασκευαστές, προκειμένου να διερευνηθεί ποιες από τις καταγεγραμμένες δυσκολίες εμφανίζονται στη συγκεκριμένη διδασκαλία:

Η χρήση των κατασκευαστών είναι προαιρετική, αφού ο μόνος λόγος χρήσης τους 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.
Στην κλήση του κατασκευαστή χρησιμοποιούνται ως ορίσματα μη ορισμένες μεταβλητές (Topor, 2002).
Στον Πίνακα 5 καταγράφονται τα λάθη των φοιτητών που αφορούν στην κλήση των 2 κατασκευαστών της κλάσης `Candidate` για τη δημιουργία 2 αντικειμένων.

Πίνακας 5: *Δημιουργία αντικειμένων.*

Δεν απάντησε	48%
Σωστή απάντηση	61%
Σύγχυση του ορισμού με την κλήση του κατασκευαστή: `new Candidate(String name, String title, int money)`	9%
Στην κλήση του κατασκευαστή χρησιμοποιούνται ως ορίσματα τα πεδία	6%
Λείπουν τα ορίσματα στην κλήση του κατασκευαστή	15%
Δεν χρησιμοποιείται το `new`	6%

Όπως φαίνεται στον Πίνακα 5, ορισμένοι φοιτητές παραλείπουν όλα ή κάποια από τα ορίσματα κατά την κλήση ενός κατασκευαστή και συγχέουν τη δήλωση των παραμέτρων στον ορισμό του και του περάσματος παραμέτρων στην ενεργοποίησή του. Βέβαια, η δυσκολία δεν αφορά ειδικά στους κατασκευαστές, αλλά γενικότερα στην κλήση μεθόδων (Hristova, 2003).

5. Συμπεράσματα

Η βασικότερη έννοια του αντικειμενοστραφούς προγραμματισμού είναι η έννοια του αντικειμένου, η οποία δεν γίνεται εύκολα κατανοητή από τους σπουδαστές. Πολλές από τις δυσκολίες που σχετίζονται με την έννοια αυτή μπορούν να αντιμετωπιστούν σε μεγάλο βαθμό χρησιμοποιώντας το περιβάλλον BlueJ.

Στη διδασκαλία που περιγράψαμε το περιβάλλον BlueJ χρησιμοποιήθηκε εκτεταμένα στα τρία πρώτα μαθήματα για την εξοικείωση των φοιτητών με την έννοια του αντικειμένου μέσω ειδικά σχεδιασμένων δραστηριοτήτων. Η χρήση των δυνατοτήτων του περιβάλλοντος BlueJ δεν επεκτάθηκε σε όλα τα μαθήματα, ή τουλάχιστον δεν αποτέλεσε το μοναδικό μέσο ελέγχου των προγραμμάτων που ανέπτυξαν οι φοιτητές, μιας και η αξιολόγηση της πρώτης διδασκαλίας έδειξε ότι η παρατεταμένη χρήση των τεχνικών οπτικοποίησης και άμεσης διαχείρισης του περιβάλλοντος BlueJ δημιουργεί προβλήματα όταν οι φοιτητές καλούνται να επιτύχουν τα ίδια αποτελέσματα γράφοντας κώδικα. Στο τέταρτο μάθημα παρουσιάστηκε η στατική μέθοδος `main` και το περιβάλλον JCreator και οι φοιτητές ξεκίνησαν να δημιουργούν αντικείμενα και να καλούν μεθόδους γράφοντας κώδικα. Στα επόμενα μαθήματα, οι φοιτητές είχαν τη δυνατότητα επιλογής του περιβάλλοντος που επιθυμούσαν για την επίλυση των ασκήσεων.

Κατά τη διάρκεια των διαλέξεων και των εργαστηριακών δραστηριοτήτων οι φοιτητές φάνηκε να κατανοούν τις σχετικές έννοιες. Ωστόσο, η γραπτή εξέταση έδειξε ότι η υλοποίηση τους δεν είναι το ίδιο εύκολη. Για την αντιμετώπιση των δυσκολιών απαιτείται σχεδίαση επιπρόσθετων δραστηριοτήτων. Για την κατανόηση του αποτελέσματος που θα έχει ένας κατασκευαστής ο οποίος αναθέτει λανθασμένες αρχικές τιμές σε κάποια πεδία μπορεί, για παράδειγμα, να χρησιμοποιηθεί ο ενσωματωμένος αποσφαλματωτής για τη βηματική εκτέλεση του κατασκευαστή με ταυτόχρονη παρακολούθηση των αλλαγών των τιμών των πεδίων. Κατά την εκτέλεση των μεθόδων ενός αντικειμένου πρέπει πάντα να είναι ορατό το παράθυρο επιθεώρησης της κατάστασής του, ενώ σε περιπτώσεις όπου δεν γίνεται κατανοητό πως αλλάζει η κατάσταση ενός αντικειμένου με την εκτέλεση των μεθόδων θα πρέπει αυτές να εκτελούνται βηματικά. Χρήσιμο είναι επίσης να παρουσιάζονται παραδείγματα λανθασμένου κώδικα και να εντοπίζονται και να αναλύονται τα λάθη τους. Πολλές φορές οι φοιτητές υιοθετούν δικούς τους κανόνες και παρανοήσεις οι οποίες είναι απαραίτητο να καταπολεμηθούν πριν να εδραιωθούν. Η υπάρχουσα γνώση για τις παρανοήσεις και τις δυσκολίες μπορούν να βοηθήσουν ουσιαστικά τον διδάσκοντα στη σχεδίαση μιας πιο αποτελεσματικής διδασκαλίας.

Τέλος, σκοπεύουμε να τροποποιήσουμε/εμπλουτίσουμε τις ατομικές εργασίες που ανατίθενται στους φοιτητές, έτσι ώστε να εφαρμόσουν κατά την εκπόνησή τους τη στρατηγική «πρώτα-το-μοντέλο» που εφαρμόζεται κατά τη διδασκαλία (ενότητα 2), αλλά συνήθως αγνοείται κατά την εκπόνηση εργασιών μιας και θωρείται χρονοβόρα στρατηγική. Για να επιτευχθεί αυτό θα ζητηθεί από τους φοιτητές, στα πλαίσια των

εργασιών που υποβάλουν: να απαντούν σε ερωτήσεις που αφορούν στην περιγραφή σημείων-κλειδιών του μοντέλου και στις μεταξύ τους συσχετίσεις, να περιγράφουν το μοντέλο με απλοποιημένα διαγράμματα UML, να προσδιορίζουν τις τεχνικές που θα εφαρμόσουν (π.χ. αλληλεπίδραση αντικειμένων, κληρονομικότητα) και τέλος να υλοποιούν το μοντέλο.

Βιβλιογραφία

Barnes, D. & Kölling, M., *Objects First with Java: A practical introduction using BlueJ*, Prentice Hall, 2004.

Bennedsen, J. & Caspersen, M. (2004), Programming in Context – A Model-First Approach to CS1, *Proceedings of SIGCSE '0*4, 477-481.

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

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

Georgantaki, S. & Retalis, S. (2007), A Learning Research Informed Design and Evaluation of a Web-enhanced Object Oriented Programming Seminar, *Journal of Information Systems Education*, Vol. 18, Number 2, 243-254.

Hristova, M., Misra, A., Rutter, M. & Mercuri, R. (2003). Identifying and Correcting Java Programming Errors for Introductory Computer Science Students. *ACM SIGCSE Bulletin*, 35 (1), 153-156.

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

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

Topor, R. W. (2002), *CIT1104 Programming II: Common (Java) programming errors*, http://www.cit.gu.edu.au/~rwt/p2.02.1/errors.html (Accessed June 2005).

Truong, N., Roe, P. & Bancroft, P. (2004), Static Analysis of Students Java Programs, *Proceedings of the 6th Australian Computing Education Conference*, 317-325.

Xinogalos, S., Satratzemi, M. & Dagdilelis, V. (2006), Studying Students' Difficulties in an OOP Course Based on BlueJ, *9th IASTED International Conference on Computers and Advanced Technology in Education*, 4-6 October 2006, Lima, Peru, 82-87.

Xinogalos, S., Satratzemi, M. & Dagdilelis, V. (2007), Re-designing an OOP course based on BlueJ, *Proceedings of the 7th IEEE International Conference on Advanced Learning Technologies*, 18-20 July 2007, Niigata, Japan, 660-664.

Ziring, N. (2001). *Novice Java Programmers' Favorite Mistakes* Page, http://users.erols. com/ziring/java-npm.html#item9 (Accessed May 2005).