

# Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση

Τόμ. 1 (2006)

5ο Συνέδριο ΕΤΠΕ «Οι ΤΠΕ στην Εκπαίδευση»



Εκπαιδευτικά Προγραμματιστικά Περιβάλλοντα για τη Διδασκαλία του Αντικειμενοστραφούς Προγραμματισμού: μια επισκόπηση

Μάγια Σατρατζέμη, Στέλιος Ξυνόγαλος, Βασίλης Δαγδιλέλης

## Βιβλιογραφική αναφορά:

Σατρατζέμη Μ., Ξυνόγαλος Σ., & Δαγδιλέλης Β. (2026). Εκπαιδευτικά Προγραμματιστικά Περιβάλλοντα για τη Διδασκαλία του Αντικειμενοστραφούς Προγραμματισμού: μια επισκόπηση. *Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση*, 1, 899–906. ανακτήθηκε από <https://eproceedings.epublishing.ekt.gr/index.php/cetpe/article/view/9214>

## ■ ΕΚΠΑΙΔΕΥΤΙΚΑ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΑ ΠΕΡΙΒΑΛΛΟΝΤΑ ΓΙΑ ΤΗ ΔΙΔΑ- ΣΚΑΛΙΑ ΤΟΥ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ: ΜΙΑ ΕΠΙΣΚΟΠΗΣΗ

### **Μάγια Σατρατζέμη**

Τμήμα Εφαρμοσμένης Πληροφορικής  
Πανεπιστήμιο Μακεδονίας  
maya.uom.gr

### **Στέλιος Ξυνόγαλος**

Τμήμα Εφαρμοσμένης Πληροφορικής  
Πανεπιστήμιο Μακεδονίας  
stelios.uom.gr

### **Βασίλης Δαγδιλέλης**

Τμήμα Εκπαιδευτικής & Κοινωνικής Πολιτικής  
Πανεπιστήμιο Μακεδονίας  
dagdil.uom.gr

### **Περίληψη**

Η διδασκαλία σε αρχάριους προγραμματιστές των αρχών του ΟΟΠ με τη Java παρουσιάζει αρκετές δυσκολίες. Πολλοί ερευνητές κατέληξαν ότι οι δυσκολίες αυτές οφείλονται σε ένα μεγάλο βαθμό στα χρησιμοποιούμενα προγραμματιστικά περιβάλλοντα. Η διαπίστωση αυτή οδήγησε στην ανάπτυξη Εκπαιδευτικών Προγραμματιστικών Περιβαλλόντων (ΕΠΠ) προκειμένου να εξαλειφθούν αυτά τα προβλήματα. Στην εργασία αυτή επιχειρείται μια επισκόπηση πέντε ΕΠΠ προκειμένου να παρουσιασθούν τα χαρακτηριστικά τους καθώς και μια συγκριτική ανάλυση των κοινών χαρακτηριστικών αλλά και των διαφορών που συναντάμε σ' αυτά.

### **Λέξεις Κλειδιά**

Εκπαιδευτικά Προγραμματιστικά Περιβάλλοντα, αντικειμενοστραφής προγραμματισμός.

### **ΕΙΣΑΓΩΓΗ**

Έρευνες σχετικά με τη διδασκαλία του προγραμματισμού έδειξαν ότι οι αρχάριοι αντιμετωπίζουν προβλήματα κατά την εκμάθηση του προγραμματισμού (Milne I., Rowe G., 2002). Οι έρευνες αυτές έδειξαν ότι δυο είναι οι βασικότεροι παράγοντες δυσκολιών: *η γλώσσα προγραμματισμού και το επαγγελματικό προγραμματιστικό περιβάλλον* (Kölling, M., 1999). Η διαπίστωση αυτή οδήγησε πολλούς ερευνητές να ακολουθήσουν τις εξής λύσεις:

- Να επινοήσουν μικρογλώσσες που χρησιμοποιούν ένα υποσύνολο μιας διαδεδομένης γλώσσας προγραμματισμού (Brusilovsky P. et al, 1997). Στους

περισσότερους από αυτούς τους προγραμματιστικούς μικρόκοσμους υφίσταται κάποιο είδος πρωταγωνιστή, δηλαδή ένα είδος οντότητας με προγραμματιζόμενη συμπεριφορά. Οι σπουδαστές μαθαίνουν τις βασικές αρχές του προγραμματισμού καθοδηγώντας τον πρωταγωνιστή στην εκπλήρωση διαφόρων έργων.

- Να αναπτύξουν ειδικά σχεδιασμένα Εκπαιδευτικά Προγραμματιστικά Περιβάλλοντα (ΕΠΠ). Τα ΕΠΠ ενσωματώνουν δυνατότητες που δίνουν μεγαλύτερη έμφαση στις εκπαιδευτικές ανάγκες των αρχαρίων παρά στις επαγγελματικές ανάγκες των προγραμματιστών.

Στην εργασία αυτή θα ασχοληθούμε με ΕΠΠ που εμφανίσθηκαν τα τελευταία χρόνια με σκοπό να υποστηρίξουν το αντικειμενοστραφές παράδειγμα και χρησιμοποιούν τη Java. Η επισκόπηση αυτή δεν θα αναφερθεί και σε περιβάλλοντα που αναπτύχθηκαν στο παρελθόν για να υποστηρίξουν το διαδικαστικό παράδειγμα προγραμματισμού καθόσον α) σχετικές μελέτες για τα ΕΠΠ για διαδικαστικό προγραμματισμό υπάρχουν, ενώ αντίθετα για τα ΕΠΠ για αντικειμενοστραφή προγραμματισμό δεν έχουν διεξαχθεί β) έχει δημιουργηθεί πλέον ένας ικανοποιητικός αριθμός ΕΠΠ για τον αντικειμενοστραφή προγραμματισμό και κατά συνέπεια μπορούν να αποτελέσουν αντικείμενο μελέτης και συγκριτικής ανάλυσης. Στις ενότητες που ακολουθούν παρουσιάζονται τα μειονεκτήματα των επαγγελματικών περιβαλλόντων προγραμματισμού, στη συνέχεια επιχειρείται μια ανασκόπηση 5 ΕΠΠ και τέλος μια συγκριτική ανάλυση αυτών σε σχέση με τα βασικά χαρακτηριστικά γνωρίσματα που απαιτείται να διαθέτει ένα ΕΠΠ.

## **ΕΚΠΑΙΔΕΥΤΙΚΑ ΚΑΙ ΕΠΑΓΓΕΛΜΑΤΙΚΑ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΑ ΠΕΡΙΒΑΛΛΟΝΤΑ**

Αν και η Java έχει επικρατήσει ως γλώσσα για τη εισαγωγή στον αντικειμενοστραφή προγραμματισμό, παρ' όλα αυτά έχει χαρακτηριστικά τα οποία δυσκολεύουν τη διδασκαλία των αρχών του (Kölling, 1999, McIver, 2002). Η κονσόλα εισόδου/εξόδου (console I/O) της Java και η διασύνδεση της γραμμής εντολών (command line interface) είναι πολύπλοκες και τρομάζουν τους αρχάριους. Όταν χρησιμοποιείται για τη διδασκαλία της Java ένας εκδότης κειμένου (text editor) και μια γραμμή εντολών τότε οι σπουδαστές κουράζονται και αποθαρρύνονται από το μηχανισμό συγγραφής και εκτέλεσης του προγράμματος, με αποτέλεσμα να δυσκολεύονται να συγκεντρωθούν στο πραγματικό στόχο που είναι η σχεδίαση και ανάπτυξη ενός αντικειμενοστραφούς προγράμματος. Έτσι πολλοί εκπαιδευτικοί καταλήγουν να χρησιμοποιούν ένα επαγγελματικό Ολοκληρωμένο Περιβάλλον Ανάπτυξης (ΟΠΑ - Integrated Development Environment) προκειμένου να μειώσουν τα παραπάνω προβλήματα.

Τα επαγγελματικά ΟΠΑ σχεδιάσθηκαν για να υποστηρίξουν τη γρήγορη ανάπτυξη προγραμμάτων και κώδικα καλύτερης ποιότητας. Η βοήθεια συνίσταται στην ενσωμάτωση ενός «έξυπνου» εκδότη κειμένου που αναλύει το συντακτικό του προγράμματος και χρησιμοποιεί επισήμανση του κώδικα, υποστηρίζει αυτόματη στοίχιση του κειμένου του προγράμματος σύμφωνα με τη δομή του, καθώς επίσης και αυτόματη συμπλήρωση κώδικα (auto-completion). Τα ΟΠΑ επίσης απλοποιούν τη διαδικασία μεταγλώττισης και εκτέλεσης προγράμματος ενσωματώνοντας επιλογές μενού ή κουμπιά για

«Μεταγλώττιση» και «Εκτέλεση». Επίσης τα επαγγελματικά ΟΠΑ ενσωματώνουν ένα αποσφαλματωτή (source level debugger) ώστε όταν το πρόγραμμα παρουσιάζει λάθη κατά την εκτέλεση να μπορεί ο προγραμματιστής να ανιχνεύει και να διακόπτει την εκτέλεση του προγράμματος. Όλα αυτά τα χαρακτηριστικά των επαγγελματικών ΟΠΑ θα μπορούσε να θεωρήσει κανείς ότι θα βοηθήσουν και τους αρχάριους προγραμματιστές. Όμως οι ανάγκες των αρχάριων παρουσιάζουν ιδιαίτερες απαιτήσεις σε σχέση μ' αυτές των επαγγελματιών προγραμματιστών.

Το ενδιαμέσο του προγραμματιστικού περιβάλλοντος πρέπει να είναι απλό και διαισθητικό. Τα επαγγελματικά ΟΠΑ σχεδιάστηκαν ώστε να προσφέρουν δυνατότητες που χρειάζονται οι επαγγελματίες με αποτέλεσμα να διαθέτουν ένα πολύπλοκο γραφικό ενδιαμέσο (GUI), το οποίο αποπροσανατολίζει και κουράζει τους αρχάριους με την πληθώρα των επιλογών και των ρυθμίσεων.

Ένα επαγγελματικό ΟΠΑ δεν προσφέρει μηχανισμούς που μειώνουν τις δυσκολίες της Java. Ως δυσκολίες θεωρούμε την ανάγκη της χρήσης της `public static void main(String[] args)` για την εκκίνηση της εκτέλεσης μιας εφαρμογής java και η σύνταξη για τις λειτουργίες της κονσόλας εισόδου/εξόδου (console I/O). Ένα ΕΠΠ θα πρέπει να προσφέρει μηχανισμούς που ξεπερνούν αυτές τις δυσκολίες

Τα επαγγελματικά ΟΠΑ δεν αντανακλούν το αντικειμενοστραφές παράδειγμα προγραμματισμού. Οι αρχάριοι όταν χρησιμοποιούν αυτά τα περιβάλλοντα διαχειρίζονται γραμμές κώδικα και όχι κλάσεις και αντικείμενα. Στον αντικειμενοστραφή προγραμματισμό οι αφηρημένες έννοιες τις οποίες θα πρέπει να κατανοήσουν οι αρχάριοι είναι οι κλάσεις και τα αντικείμενα και κατά συνέπεια θα πρέπει το ΟΠΑ να διαθέτει ένα τρόπο που να επιτρέπει την άμεση διαχείριση αυτών των εννοιών μέσα από οπτικοποίηση (visualization) και όχι μόνο με τη σύνταξη γραμμών κώδικα.

Τα περισσότερα επαγγελματικά ΟΠΑ προσφέρουν δυνατότητες που επιτρέπουν τη γρήγορη ανάπτυξη γραφικών εφαρμογών «σέρνοντας και αφήνοντας» κουμπιά και έτσι προκαλούν σύγχυση στους αρχάριους θεωρώντας ότι οι αρχές του αντικειμενοστραφούς προγραμματισμού ισοδυναμούν με τη χρήση εργαλείων ανάπτυξης γραφικών ενδιάμεσων.

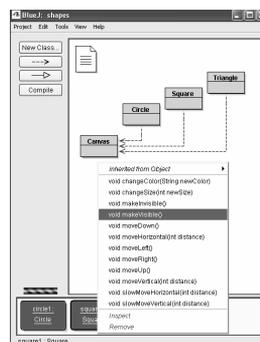
## ΕΠΙΣΚΟΠΗΣΗ ΕΠΠ

Στην ενότητα αυτή θα παρουσιάσουμε τα ΕΠΠ τα οποία υποστηρίζουν τη διδασκαλία της γλώσσας Java, διανέμονται ελεύθερα (Freeware) και ικανοποιούν τις ιδιαίτερες απαιτήσεις των αρχάριων. Στη μελέτη αυτή δεν θα αναφερθούμε στη ανάλυση προγραμματιστικών μικρόκοσμων που επίσης αναπτύχθηκαν για τον αντικειμενοστραφή προγραμματισμό. Τα ΕΠΠ που θα παρουσιάσουμε είναι: το BlueJ, DrJava, Ginipad, JGrasp, JCreator LE.

### **BlueJ** (<http://www.bluej.org/>)

Το γραφικό ενδιαμέσο του BlueJ (Kolling, M. et al., 2003) φαίνεται στο Σχήμα 1. Στο παράθυρο διαχείρισης του BlueJ παρουσιάζονται με οπτικό τρόπο τα αντικείμενα και οι κλάσεις, καθώς και η δομή της εφαρμογής με τη χρήση διαγραμμάτων UML. Χρησιμοποιώντας το αλληλεπιδραστικό ενδιαμέσο του BlueJ οι σπουδαστές μπορούν να δημιουργήσουν και να επιθεωρήσουν την κατάσταση των αντικειμένων και να καλέσουν μεθόδους. Ο συντάκτης

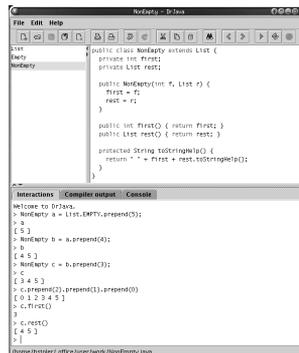
προγραμμάτων υποστηρίζει περιορισμένο χρωματισμό του κώδικα. Ο χρήστης μπορεί να ζητήσει να εμφανίζονται αριθμοί γραμμών στον πηγαίο κώδικα και υποστηρίζει αυτόματη στοίχιση και ταίριασμα αγκυλών. Σε περίπτωση συντακτικών λαθών αυτά εμφανίζονται στο κάτω μέρος του παραθύρου του συντάκτη. Δεν υπάρχει δυνατότητα αλληλεπίδρασης μεταξύ του μηνύματος λάθους και της γραμμής κώδικα. Το BlueJ παρέχει ένα οπτικό αποσφαλματωτή που επιτρέπει στο σπουδαστή να εισάγει σημεία διακοπής (breakpoints) και στη συνέχεια να εκτελέσει το πρόγραμμα βηματικά. Χρησιμοποιώντας το αναδυόμενο μενού μιας κλάσης ή ενός αντικειμένου ο χρήστης μπορεί να ζητήσει την εκτέλεση των μεθόδων και αυτόματα εμφανίζεται το παράθυρο του αποσφαλματωτή.



Σχήμα 1. BlueJ.

### DrJava (<http://drjava.org/>)

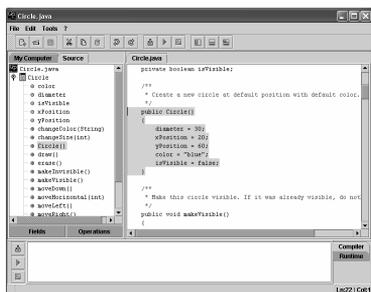
Το γραφικό ενδιάμεσο του DrJava (Allen E. et al. 2004) φαίνεται στο Σχήμα 2. Στην καρτέλα Interaction ο σπουδαστής μπορεί να εισάγει εκφράσεις σε Java και να δει άμεσα τα αποτελέσματά τους. Το DrJava παρέχει ένα απλό ενδιάμεσο για την εκτέλεση προγραμμάτων σε Java, εξαλείφοντας την ανάγκη της μεθόδου main(). Επιπλέον, το ενδιάμεσο παρέχει στους σπουδαστές τη δυνατότητα άμεσης παρατήρησης της συμπεριφοράς μεμονωμένων μεθόδων, ενισχύοντας την ιδέα ότι κάθε τμήμα ενός προγράμματος πρέπει να ελέγχεται χωριστά. Στην καρτέλα Compiler Output παρουσιάζονται τα λάθη που έχουν εντοπιστεί από τον μεταγλωττιστή. Κάνοντας κλικ σε οποιοδήποτε λάθος επισημαίνεται η αντίστοιχη γραμμή στον κώδικα. Το περιβάλλον του DrJava ενσωματώνει και ένα αποσφαλματωτή, τον οποίο πρέπει να ενεργοποιήσει ο χρήστης μέσω επιλογής μενού. Στην περίπτωση αυτή εμφανίζεται ένα τμήμα με πληροφορίες και ενεργοποιούνται αρκετές επιλογές στο μενού του αποσφαλματωτή. Ο χρήστης μπορεί να τοποθετήσει στον κώδικα Breakpoints και να επιλέξει από το μενού Step του αποσφαλματωτή κάποιον από τους διαθέσιμους τρόπους εκτέλεσης των μεθόδων (step into/out/over).



Σχήμα 2. DrJava.

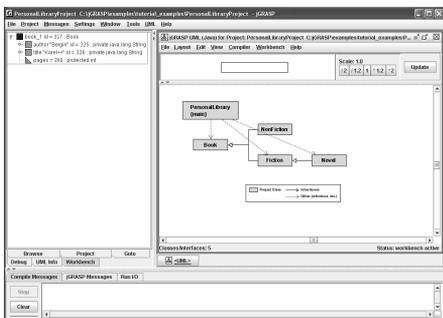
### Ginipad (<http://www.mokabyte.it/ginipad/english.htm>)

Το γραφικό ενδιάμεσο του Ginipad φαίνεται στο Σχήμα 3. Ο συντάκτης προγραμμάτων επισημαίνει στοιχεία του πηγαίου κώδικα με χρήση χρωμάτων και υποστηρίζει τη λειτουργία της αυτόματης ολοκλήρωσης κατά τη σύνταξη του πηγαίου κώδικα. Στο τμήμα παρουσίασης της δομής του κώδικα (Source Browser) παρέχεται μια παρουσίαση των κλάσεων, των μεθόδων, των πεδίων και των διασυνδέσεων του πηγαίου κώδικα με τη μορφή ενός δέντρου. Σε περίπτωση που ένα πρόγραμμα περιέχει λάθη ύστερα από τη μεταγλώττιση στο κάτω μέρος του περιβάλλοντος (Compiler pane) εμφανίζεται μία λίστα



Σχήμα 3. Το Ginipad.

με τα αντίστοιχα προγραμματιστικά λάθη, στην οποία ο χρήστης έχει τη δυνατότητα να επιλέξει κάποιο από αυτά και να μεταβεί στον πηγαίο κώδικα όπου βρίσκεται το συγκεκριμένο λάθος. Το Ginipad δεν διαθέτει αποσφαλματωτή με αποτέλεσμα να μην υποστηρίζει τη αποσφαλμάτωση ενός προγράμματος αλλά και την κατανόηση της εκτέλεσης ενός προγράμματος αφού δεν υπάρχει η δυνατότητα βηματικής εκτέλεσης.



Σχήμα 4. JGrasp.

**JGrasp** (<http://www.eng.auburn.edu/grasp/>)

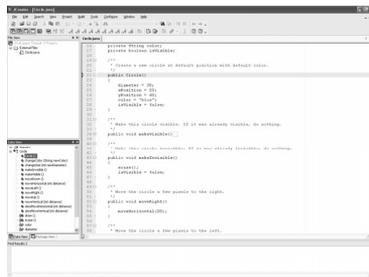
Το γραφικό ενδιάμεσο του JGrasp (Cross J. et al., 2002) φαίνεται στο Σχήμα 4. Ο συντάκτης προγραμμάτων υποστηρίζει επισήμανση στοιχείων του πηγαίου κώδικα με χρήση χρωμάτων. Επίσης, ο χρήστης μπορεί να ζητήσει να εμφανίζονται αριθμοί γραμμών στον πηγαίο κώδικα και να δει τη διαγραμματική μορφή του προγράμματος με τη χρήση των δια-

γραμμάτων δομής CSD. Με τη βοήθεια προτύπων προτείνεται κώδικας για τη δημιουργία κλάσης, μεθόδων, δομών ελέγχου και επανάληψης. Τα λάθη που αναφέρονται από τον μεταγλωττιστή παρουσιάζονται στην καρτέλα Compile Messages. Η περιγραφή του πρώτου λάθους που εντοπίζεται επισημαίνεται με αλλαγή του χρώματος ενώ ταυτόχρονα επισημαίνεται και η γραμμή του πηγαίου κώδικα όπου κατά πάσα πιθανότητα βρίσκεται το λάθος. Το JGrasp παρέχει ένα οπτικό αποσφαλματωτή που επιτρέπει στον σπουδαστή να εισάγει σημεία διακοπής και στη συνέχεια να εκτελέσει το πρόγραμμα βηματικά. Το JGrasp δημιουργεί διαγράμματα UML, τα οποία παράγονται για τις κλάσεις του τρέχοντος project. Ο σπουδαστής μπορεί να δημιουργήσει στιγμιότυπα για οποιαδήποτε κλάση του διαγράμματος UML, καθώς επίσης και στιγμιότυπα για οποιαδήποτε κλάση Java. Όταν δημιουργείται ένα αντικείμενο τότε αυτό εμφανίζεται στην καρτέλα workbench και ο σπουδαστής έχει τη δυνατότητα να το επιλέξει και να ενεργοποιήσει οποιαδήποτε μέθοδο ενός αντικειμένου μεμονωμένα, χωρίς να υπάρχει ανάγκη ανάπτυξης ενός προγράμματος προκειμένου να ελεγχθεί κάθε μέθοδος της κλάσης.

**JCreator LE** (<http://www.jcreator.com/index.htm>)

Το γραφικό ενδιάμεσο του JCreator φαίνεται στο Σχήμα 5 και αποτελείται από τέσσερα βασικά τμήματα: Το τμήμα File View που παριστάνει σε δενδρική μορφή όλα τα αρχεία που μπορεί ένα περιέχει ένα project. Επιτρέπει την προσθήκη νέων κλάσεων στο project μέσω προτύπων και πλαισίων διαλόγου. Το τμήμα Data View παρέχει μια σύνοψη της δομής της τρέχουσας κλάσης με τη μορφή ενός δένδρου και επιτρέπει τη προσθήκη μεθόδων και μεταβλητών μελών στην κλάση με τη βοήθεια πλαισίων διαλόγου. Και το τμήμα Code View.

Το JCreator παρέχει τη δυνατότητα εμφάνισης αριθμησης των γραμμών του πηγαίου κώδικα, αυτόματη στοίχιση, συμπλήρωση λέξεων κατά την πληκτρολόγηση και απόκρυψη τμημάτων κώδικα, για παράδειγμα του σώματος μιας μεθόδου. Το τμήμα που εκτείνεται κατά μήκος του κάτω μέρους του παραθύρου περιλαμβάνει πληροφορίες που σχετίζονται με αποτελέσματα εξόδου των εφαρμογών Java (General Output) καθώς επίσης και με αποτελέσματα εξόδου ύστερα από τη μεταγλώττιση ενός αρχείου (Build Output). Δεν περιλαμβάνει εργαλείο αποσφαλμάτωσης του κώδικα και κατά συνέπεια δεν υποστηρίζει βηματική εκτέλεση του κώδικα. Η έκδοση JCreator pro (shareware έκδοση) περιλαμβάνει αποσφαλματωτή και επιπλέον λειτουργίες.



Σχήμα 5. JCreator LE.

## ΣΥΓΚΡΙΤΙΚΗ ΑΝΑΛΥΣΗ

Οι βασικές κατηγορίες γνωρισμάτων οι οποίες θα πρέπει να λαμβάνονται υπόψη κατά τη συγκριτική ανάλυση ΕΠΠ (McIver, 2002) είναι οι εξής: *γραφικό ενδιαμέσο, συντάκτης, αποσφαλματωτής, μηνύματα λάθους, οπτικοποίηση, μεμονωμένη εκτέλεση μεθόδων και δημιουργία αντικειμένων, και αξιολόγηση από σπουδαστές*. Με βάση αυτά τα χαρακτηριστικά παραρνούσαμε στη συνέχεια τις ομοιότητες και διαφορές των παραπάνω 5 ΕΠΠ:

**Γραφικό ενδιαμέσο (GUI):** Από τα 5 περιβάλλοντα το BlueJ αλλά και το JGrasp αντανακλούν περισσότερο το παράδειγμα της γλώσσας προγραμματισμού, γιατί: δημιουργούν διαγράμματα UML που παρουσιάζουν τις συσχετίσεις των κλάσεων, μέσω άμεσης διαχείρισης των κλάσεων και των αντικειμένων επιτρέπουν τη δημιουργία αντικειμένων και τη κλήση μεθόδων χωρίς να χρειάζεται να συνταχθεί κώδικας, γεγονός που διευκολύνει, από τα πρώτα μαθήματα, στην παρουσίαση των βασικών εννοιών του αντικειμενοστραφούς προγραμματισμού, όπως: κλάση, ιδιότητες και συμπεριφορά αντικειμένου, αποστολή μηνυμάτων σε αντικείμενα, κληρονομικότητα κτλ.

**Συντάκτης (Editor):** Όλα τα περιβάλλοντα διαθέτουν τις βασικές λειτουργίες ενός συντάκτη κώδικα. Όλα τα περιβάλλοντα υποστηρίζουν επισήμανση των στοιχείων της γλώσσας με τη χρήση χρωμάτων και ρυθμίσεων των χαρακτηριστικών εμφάνισης του κώδικα, ενώ το BlueJ τα παρουσιάζει σε πιο περιορισμένο βαθμό. Το Ginipad δεν εμφανίζει αριθμηση των γραμμών κώδικα, ενώ το Ginipad και JGrasp δεν υποστηρίζουν ταίριασμα αγκυλών. Τα BlueJ, DrJava και Ginipad δεν υποστηρίζουν αναδίπλωση κώδικα. Δυνατότητα σχεδόν πλήρους ανάπτυξης κώδικα μέσω προτύπων διαθέτει το JGrasp, ενώ το BlueJ, Ginipad και JCreator μόνο σε επίπεδο κλάσης. Δυνατότητα για αυτόματη συμπλήρωση κώδικα κατά την ανάπτυξη των προγραμμάτων διαθέτουν μόνο τα JGrasp και JCreator.

**Μεταγλώττιση – μηνύματα λάθους:** Τα DrJava, Ginipad, JGrasp, JCreator επιτρέπουν την αλληλεπίδραση μεταξύ του μηνύματος λάθους και της αντίστοιχης γραμμής κώδικα ενώ το BlueJ όχι. Όμως, μόνο το BlueJ υποστηρίζει τον αρχάριο στον εντοπισμό του λάθους προσφέροντας βοήθεια με διευκρινιστικό κείμενο σχετικό με τις πιθανές αιτίες που ενδεχόμενα προκάλεσαν το

συντακτικό λάθος. Το BlueJ προσφέρει σημαντική βοήθεια στην κατανόηση των πιθανών αιτίων που προκάλεσαν το συντακτικό λάθος δίνοντας επεξηγήσεις, πράγμα που δεν συμβαίνει στα άλλα περιβάλλοντα που παρουσιάζουν τα μηνύματα λάθους με κωδικοποιημένη μορφή.

**Αποσφαλματωτής (Debugger):** Τα περιβάλλοντα Ginipad & JCreator δεν ενσωματώνουν αποσφαλματωτή. Στο DrJava η χρήση του αποσφαλματωτή είναι πολύπλοκη αφού μοιάζει μ' αυτή των επαγγελματικών περιβαλλόντων, μιας και απαιτεί ρυθμίσεις εκ μέρους του χρήστη για την εμφάνιση των σχετικών πληροφοριών.

**Οπτικοποίηση (Visualization):** Η δομή των κλάσεων (methods, fields) παρουσιάζεται στο Ginipad & JCreator. Τα BlueJ & JGrasp παρουσιάζουν τις αλληλοεξαρτήσεις των κλάσεων με τη βοήθεια των διαγραμμάτων UML. Το BlueJ επιτρέπει τη δημιουργία του διαγράμματος UML με άμεση διαχείριση εικονιδίων αλλά και αντίστροφα, ενώ το JGrasp μόνο με βάση τη δομή του project. Η παρακολούθηση της συμπεριφοράς ενός αντικειμένου γίνεται μόνο στα BlueJ & JGrasp. Επιπλέον, το JGrasp δημιουργεί διαγράμματα CSD που επιτρέπουν τη γραφική αναπαράσταση της δομής του κώδικα. Μόνο το BlueJ προσφέρει τη γραφική αναπαράσταση των αντικειμένων, ενώ το JGrasp με μορφή κειμένου.

**Μεμονωμένη εκτέλεση μεθόδων – δημιουργία αντικειμένων:** Τα BlueJ & JGrasp επιτρέπουν τη μεμονωμένη εκτέλεση μεθόδων (και δημιουργία αντικειμένων) με τη βοήθεια γραφικού ενδιαμέσου και άμεση διαχείριση στις κλάσεις του διαγράμματος UML. Επιπλέον στο BlueJ τα αντικείμενα αναπαρίστανται με γραφικό τρόπο. Στο DrJava υπάρχει η δυνατότητα εκτέλεσης των μεθόδων γράφοντας εκφράσεις και εντολές.

**Αξιολόγηση:** Αποτελέσματα χρήσης στην τάξη παρουσιάζονται μόνο για το BlueJ (Van Haaster, Hagan, 2004) ενώ συγκριτικά αποτελέσματα χρήσης των παραπάνω περιβαλλόντων δεν αναφέρονται στη βιβλιογραφία.

## ΣΥΜΠΕΡΑΣΜΑΤΑ

Από την παραπάνω παρουσίαση και συγκριτική ανάλυση των χαρακτηριστικών των ΕΠΠ γίνεται φανερό ότι από τα 5 ΕΠΠ το BlueJ και JGrasp ενσωματώνουν δυνατότητες που πιστεύουμε ότι βοηθούν τον αρχάριο προγραμματιστή αλλά και επιτρέπουν στον εκπαιδευτικό να διδάξει τις αρχές του ΟΟΠ με μια διαφορετική προσέγγιση απ' αυτή που απαιτούσε το διαδικαστικό παράδειγμα προγραμματισμού. Η προσέγγιση διδασκαλίας του αντικειμενοστραφούς παραδείγματος προγραμματισμού απαιτεί την παρουσίαση και κατανόηση των εννοιών: κλάση, αντικείμενο, κληρονομικότητα από την αρχή. Πιστεύουμε ότι σ' αυτό μπορούν να συνδράμουν αποτελεσματικά και τα δυο αυτά περιβάλλοντα μέσω των διαγραμμάτων UML, της αναπαράστασης των αντικειμένων (ιδιαίτερα το BlueJ επιτρέπει την οπτική αναπαράσταση αντικειμένου), της μεμονωμένης εκτέλεσης μεθόδων και δημιουργίας αντικειμένων με ιδιαίτερα εύκολο τρόπο καθώς και του αποσφαλματωτή που ενεργοποιείται χωρίς ιδιαίτερες ρυθμίσεις. Βέβαια η διαπίστωση κατά πόσον ένα ΕΠΠ πραγματοποιεί το σκοπό για τον οποίο κατασκευάστηκε καθώς και η υπεροχή ενός ΕΠΠ σε σχέση με ένα άλλο, επιτυγχάνεται μόνο με τη βοήθεια συγκριτικής αξιολόγησης από τους χρήστες για τους οποίους κατασκευάστηκε, δηλαδή τους αρχάριους προγραμματιστές. Η συγκριτική αξιολόγηση των ΕΠΠ παρέχει σημαντικές πληροφορίες για τη σχεδίαση νέων συστημάτων και την επιλογή από τα υπάρ-

χοντα. Δυστυχώς, μέχρι στιγμής χρειάζεται αρκετή προσπάθεια σ' αυτόν τον τομέα και ιδιαίτερα στην ανάπτυξη μιας κοινά αποδεκτής τεχνικής αξιολόγησης (McIver, 2002) που θα επιτρέψει τη συγκριτική ανάλυση διαφορετικών ΕΠΠ, έτσι ώστε να μπορούν να συλλέγονται περισσότερες πληροφορίες που θα επιτρέπουν μια συζήτηση σχετικά με την επιλογή του προγραμματιστικού περιβάλλοντος που είναι πιο κατάλληλο για μια εισαγωγή στον αντικειμενοστραφή προγραμματισμό.

## ΒΙΒΛΙΟΓΡΑΦΙΑ

- Allen E., Cartwright R., Stoler B., (2004), DrJava: a lightweight pedagogic environment for Java, *Proceedings of the SIGCSE 2004*, 137-141.
- Brusilovsky, P., Calabrese, E., Hvorecky, E., Kouchnirenko, A., & Miller, P., (1997), Mini-languages: A Way to Learn Programming Principles, *Education and Information Technologies*, 2(1), 65-83.
- Cross J., T. Hendrix D., Barowski L., (2002) Using The Debugger As An Integral Part Of Teaching Cs1, *32nd ASEE/IEEE Frontiers in Education Conference*, Boston.
- Kölling, M., (1999) The Problem of Teaching Object-Oriented Programming, Part 2: Environments, *Journal of Object-Oriented Programming*, Vol. 11 No. 9, 6-12.
- Kölling, M., Quig, B., Patterson, A. and Rosenberg, J., (2003), The BlueJ system and its pedagogy, *Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology*, Vol 13, No 4.
- McIver, L., (2002), Evaluating Languages and Environments for Novice Programmers, *Fourteenth Annual Workshop of the Psychology of Programming Interest Group (PPIG 2002)*, Middlesex, UK, 100-110.
- Milne, I., Rowe, G., (2002), Difficulties in Learning and Teaching Programming - Views of Students and Tutors, *Education and Information Technologies*, 7:1, Kluwer Academic Publishers, 55-66.
- Van Haaster, K. and Hagan, D., (2004), Teaching and Learning with BlueJ: an Evaluation of a Pedagogical Tool, *Information Science & Information Technology Education Joint Conference*, Rockhampton, QLD, Australia, 456-470.