ΕΚΠΑΙΔΕΥΣΗ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΜΕ ΧΡΗΣΗ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΚΥΡΑΝΟΣ ΗΛΙΑΣ ΑΜ:14

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων καθηγητής : Λαζαρίδης Βασίλειος,
Ζαχαροπουλος Βασίλειος

ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΠΟΛΥΜΕΣΩΝ

ΗΡΑΚΛΕΙΟ 2008

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Οι Αλγόριθμοι και οι Δομές Δεδομένων αποτελούν αντικείμενα της Επιστήμης των Υπολογιστών τα οποία, κατά γενική ομολογία, είναι δύσκολα στην κατανόησή τους. Κατά συνέπεια οι διδάσκοντες και οι σπουδαστές χρησιμοποιούνε συχνά εικόνες (σχήματα) που βοηθούνε τους μεν να τα διδάξουν αποδοτικότερα και τους δε να τα κατανοήσουν καλύτερα. Παραδείγματα τέτοιων εικόνων είναι: ένα διάγραμμα ροής που απεικονίζει τα βήματα της εκτέλεσης ενός αλγορίθμου, η σχεδίαση μιας συνδεδεμένης λίστας με σκοπό να εξηγηθούν οι λειτουργίες της, η σχεδίαση πλαισίων μιας στοίβας για την εξήγηση των αναδρομικών κλήσεων μιας συνάρτησης, η σχεδίαση ενός γραφήματος για την εξήγηση ενός αλγορίθμου γραφημάτων κ.λ.π. Όμως οι λειτουργίες μιας Δομής Δεδομένων ή ενός Αλγορίθμου είναι από τη φύση τους δυναμικές και συνεπώς η σχεδίαση μιας τέτοιας εικόνας στον πίνακα ή στο χαρτί δεν είναι ο πιο κατάλληλος τρόπος να απεικονιστούν. Ενδεικτικά αναφέρουμε το εξής σενάριο: ένας διδάσκων σχεδιάζει στον πίνακα μια συνδεδεμένη λίστα προσπαθώντας να εξηγήσει στους φοιτητές τις βασικές λειτουργίες της. Καθώς εισάγονται νέοι κόμβοι στη λίστα είναι πιθανόν να μην επαρκεί ο χώρος για να σχεδιαστούν. Στην περίπτωση αυτή η λίστα θα πρέπει να σχεδιαστεί από την αρχή ώστε να χωρέσει στο διαθέσιμο χώρο. Επίσης όταν διαγράφονται κάποιοι κόμβοι της λίστας, ο διδάσκων θα είναι αναγκασμένος να σβήνει κόμβους και να ξανασχεδιάζει κάποιους δείκτες (pointers). Το αποτέλεσμα αυτής της διαδικασίας είναι η δυσκολία στην παρακολούθηση των λειτουργιών μιας λίστας από τους φοιτητές.

Συγκεκριμένα το Jeliot μπορεί να διευκολύνει τους καθηγητές κατά πολύ στην διδασκαλία της γλώσσας προγραμματισμού Java αλλά και τους μαθητές να κατανοήσουν πολύ πιο εύκολα τον τρόπο λειτουργίας της.

Κεφάλαιο 1

Εισαγωγή

Η δυσκολία να κατανοηθούν έννοιες από τους φοιτητές οδηγούν τους εκπαιδευτικούς να αναζητούν νέους τρόπους διδασκαλίας. Ένα διαλογικό λογισμικό οπτικοποίησης δίνει μια πιθανή λύση στους παραδοσιακούς τρόπους διδασκαλίας και εκμάθησης. Η εφαρμογή οπτικοποίησης έχει το πλεονέκτημα της επανάληψης καθοδηγώντας τους φοιτητές βήμα-βήμα. Η δυνατότητα των υπολογιστών να αποθηκεύουν απέραντες ποσότητες πληροφοριών, τους καθιστά χρήσιμο εργαλείο ως βοηθητικό μέσο στην ενίσχυση της εκπαιδευτικής διδασκαλίας. Η χρήση των εικόνων διευκολύνει την επεξήγηση. Η χρήση της τεχνολογίας αυξήθηκε στην κοινωνία, όχι όμως και στη διδασκαλία. Εξαίρεση αποτελεί η εξ' αποστάσεων εκπαίδευση.

Οι υπολογιστές με ενσωματωμένες δυνατότητες χρήσης τεχνολογιών πολυμέσων έδωσαν την ευκαιρία στους εκπαιδευτές, αλλά και στους σχεδιαστές εκπαιδευτικού λογισμικού, να συμπεριλάβουν πρόσθετα μέσα για να ενισχύσουν την εκπαιδευτική διαδικασία όπως τον ήχο, την εικόνα, το βίντεο. Το εκπαιδευτικό λογισμικό βοηθά τους φοιτητές να μάθουν μέσω πειραματισμού, έναντι του συνήθως παθητικού περιβάλλοντος μιας παραδοσιακής διάλεξης στην τάξη. Ένα βασικό κομμάτι της εκπαίδευσης με χρήση των Ηλεκτρονικών Υπολογιστών είναι η διδασκαλία γλωσσών προγραμματισμού.

Η οπτικοποίηση προγράμματος είναι η απεικόνιση του κώδικα ενός προγράμματος με στατική ή με δυναμική μορφή. Στατική οπτικοποίηση είναι η παρουσίαση των περιεχομένων μιας λίστας ως διάγραμμα ενώ δυναμική οπτικοποίηση είναι το ίδιο διάγραμμα με τα περιεχόμενα και τα βέλη να αλλάζουν δυναμικά το περιεχόμενο της λίστας κατά την εκτέλεση του προγράμματος. Αναμένεται ότι η οπτικοποίηση θα βοηθήσει την διαδικασία κατανόησης καθώς ο εγκέφαλος θα συνοψίσει κάθε χαρακτήρα, λέξη, πρόταση ή μια παράσταση. Παιδαγωγικά είναι ασαφές πως μια δυναμική παρουσίαση μπορεί να διευκολύνει την εκμάθηση ή την επίλυση προβλημάτων και οι σχεδιαστές οπτικοποίησης προωθούν συστήματα απεικόνισης ανάλογα με την αισθητική τους ευχαρίστηση. Τα παιχνίδια των υπολογιστών θεωρείται ότι παρακινούν το φοιτητή στην εκμάθηση των υπολογιστών. Βασικά

χαρακτηριστικά είναι τρία γνωρίσματα: πρόκληση, φαντασία και περιέργεια. Οι σημαντικές πληροφορίες δίνονται χρησιμοποιώντας κάποιο σήμα το οποίο ενεργοποιεί ένα αισθητήριο κανάλι, αυξάνοντας την προσοχή. Τέτοια τεχνάσματα χρησιμοποιούνται στον σχεδιασμό συστημάτων οπτικοποίησης για να κατευθύνουν την προσοχή του φοιτητή. Στόχος είναι η βελτίωση της γνωστικής διαδικασίας.

Η ασύγχρονη τηλεκπαίδευση δεν απαιτεί την ταυτόχρονη συμμετοχή των μαθητών και των εισηγητών. Επιλεγούν μονοί το προσωπικό τους εκπαιδευτικό χρονικό πλαίσιο. Στο είδος της εκπαίδευσης αυτής ανήκει η Αυτοδιδασκαλία, η Ημιαυτόματη Εκπαίδευση και η Συνεργατική Εκπαίδευση.

Στην αυτοδιδασκαλία ο εκπαιδευόμενος εκπαιδεύεται μονός του χρησιμοποιώντας οποίο μέσο θέλει (βιβλία, Internet κ.α.), ενώ στην Ημιαυτόματη εκπαίδευση ισχύει το ίδιο με την Αυτοδιδασκαλία, μονό που υπάρχει και συγκεκριμένο χρονοδιάγραμμα. Στην Συνεργατική Εκπαίδευση ο εκπαιδευτής με τον εκπαιδευόμενο επικοινωνούν ασύγχρονα μεταξύ τους και χρησιμοποιούν χρονοδιάγραμμα παράδοσης των εργασιών. Στην ασύγχρονη τηλεκπαίδευση οι μαθητές δημιουργούν τις δικές τους ομάδες εργασίας και επικοινωνούν χωρίς τη μεσολάβηση των εκπαιδευτικών.

Πλεονεκτήματα της ασύγχρονης τηλεκπαίδευσης σε σχέση με την παραδοσιακή διδασκαλία:

- Χαμηλότερο κόστος - εξοικονόμηση χρόνου
- Ευκαιρία μάθησης για άτομα με δυσκολία μετακίνησης
- Ασύμμετρη διδασκαλία (χρόνος που εξυπηρετεί τον καθένα)
- Καλύτερη ποιότητα διδακτικού προσωπικού
- Ταχύτερη προσαρμογή στις ανάγκες των μαθητών
- Μεγαλύτερη οικογενειακή συνοχή

Μειονεκτήματα της ασύγχρονης τηλεκπαίδευσης:

- Απώλεια της επικοινωνίας μέσω προσωπικής επαφής
- Λιγότερο αυθόρμητη επικοινωνία
- Απαγόρευση μετάδοσης μέσω web θεμάτων copyright
- Χαμηλότερη κοινωνικοποίηση

- Δυσκολία ελέγχου της ταυτότητας του εκπαιδευομένου

Έχουν αναπτυχθεί διάφοροι τρόποι γραφικών απεικονίσεων. Έχει δηλωθεί ότι οι ανθρώπινες αισθήσεις είναι καταλληλότερες για τις οπτικές πληροφορίες. Οι περισσότεροι επαγγελματίες στους υπολογιστές χρησιμοποιούν την οπτικοποίηση ως εργαλείο. Η οπτικοποίηση λογισμικού ταιριάζει περισσότερο στον τρόπο λειτουργίας του ανθρωπίνου εγκεφάλου, κατά την επεξεργασία, την διαχείριση και αναγνώριση εικόνων και οπτικών δομών. Έχουν σχεδιαστεί πολλές ανεξάρτητες εφαρμογές που έχουν σκοπό την διδασκαλία γλωσσών προγραμματισμού, ορισμένα από αυτά τα συστήματα είναι το Dynalab,το PECAN, το FIELD, το ZSTEP95, το Leonardo και το JELIOT.

Το JELIOT είναι μια εφαρμογή δυναμικής αναπαράστασης της εκτέλεσης προγραμμάτων και μπορεί να χρησιμοποιηθεί για την εκμάθηση εισαγωγικών εννοιών προγραμματισμού. Είναι εξαιρετικά απλό έτσι ώστε να γίνεται αποδεκτό από αρχάριους προγραμματιστές και από τους καθηγητές τους οι οποίοι γλιτώνουν χρόνο καθώς δεν χρειάζεται να μάθουν πώς να προετοιμάζουν την δυναμική αναπαράσταση της εκτέλεσης ενός προγράμματος.

Το Jeliot έχει γραφτεί με τη χρήση της γλώσσας προγραμματισμού Java προκειμένου να αποκτήσει τη δυνατότητα εκτέλεσης σε ανεξάρτητο από λειτουργικό σύστημα περιβάλλον. Αναπαριστά δυναμικά προγράμματα που έχουν γραφτεί σε γλώσσα προγραμματισμού Java.

Τα βασικότερα μέρη του Jeliot είναι το τμήμα πηγαίου κώδικα το οποίο είναι το αριστερό τμήμα της οθόνης του γραφικού περιβάλλοντος που περιλαμβάνει έναν επεξεργαστή κειμένου όπου γίνεται η εισαγωγή του πηγαίου κώδικα του προγράμματος. Το πλαίσιο δυναμικής αναπαράστασης αποτελεί το κεντρικό τμήμα της εφαρμογής του Jeliot, είναι το τμήμα εκείνο όπου οποιαδήποτε δυναμική αναπαράσταση της εκτέλεσης του πηγαίου κώδικα λαμβάνει χώρα.

Η δομή της πτυχιακής εργασίας είναι:

Κεφάλαιο 1 : Εισαγωγή

Περιλαμβάνει την αρχική περιγραφή του θέματος της πτυχιακής εργασίας, καθορίζει το πλαίσιο διερεύνησης του θέματος της διδασκαλίας γλωσσών προγραμματισμού σε ένα ασύγχρονο περιβάλλον εκπαίδευσης.

Κεφάλαιο 2 : Εκπαίδευση και Υπολογιστές

Αναζητείται και παρουσιάζεται η δυνατότητα χρήσης ηλεκτρονικών υπολογιστών στην εκπαίδευση. Επίσης εισάγονται έννοιες όπως ασύγχρονη εκπαίδευση και τηλεκπαίδευση που περιγράφουν τα εκπαιδευτικά λογισμικά διδασκαλίας γλωσσών προγραμματισμού, παρουσιάζονται τα πλεονεκτήματα αλλά και τα μειονεκτήματα της χρήσης της τεχνολογίας του εκπαιδευτικού λογισμικού στην ασύγχρονη τελεκαπίδευση.

Κεφάλαιο 3 : Οπτικοποίηση Λογισμικού

Περιλαμβάνει εκτεταμένη αναφορά στην έννοια της οπτικοποίησης λογισμικού, τους τρόπους χρήσης της συγκεκριμένης τεχνολογίας στην ανάπτυξη λογισμικού, καθώς αποτελεί αναπόσπαστο τμήμα των εφαρμογών που αναπτύσσονται αποκλειστικά για να βοηθήσουν την εκπαιδευτική διαδικασία. Ακολουθεί η παρουσίαση μερικών εφαρμογών οπτικοποίησης λογισμικού από τα σημαντικότερα όπως το Dynalab,το PECAN, το FIELD, το ZSTEP95, το Leonardo και το JELIOT.

Κεφάλαιο 4 : Jeliot

Παρουσιάζει τις απαραίτητες οδηγίες εγκατάστασης του Jeliot το οποίο περιλαμβάνει δύο διαφορετικούς τρόπους χρήσης και εγκατάστασης: Το πρώτο χρησιμοποιεί το Windows Installer της Jeliot, ενώ το δεύτερο με τη χρήση της εκτελέσιμης διανομής του Jeliot. Γίνεται η πρώτη γνωριμία με το γραφικό περιβάλλον εργασίας (graphical user interface). Θα γίνει ανάλυση του μενού του προγράμματος, των κουμπιών εργαλειοθήκης και των κουμπιών που χρησιμοποιούνται για τον έλεγχο της δυναμικής αναπαράστασης του προγράμματος.

Κεφάλαιο 5 : Μαθήματα Jeliot

Το κεφάλαιο αυτό ασχολείται αποκλειστικά με την παρουσίαση του τρόπου διδασκαλίας με την εφαρμογή διδακτικών δραστηριοτήτων για την εκμάθηση της γλώσσας Java μέσω του προγράμματος Jeliot. Παρουσιάζονται σε μια σειρά μαθημάτων ξεκινώντας από την αρχή με τον τρόπο δήλωσης των μεταβλητών, μέχρι και τις πιο σύνθετες δομές ελέγχου. Τα παραδείγματα εφαρμογής του περιβάλλοντος εκπαίδευσης του Jeliot, περιλαμβάνουν επίσης τους τελεστές και τον ορισμό τους (αριθμητικοί τελεστές – παράδειγμα χρήσης αριθμητικών τελεστών, τελεστές σύγκρισης, λογικοί τελεστές – παράδειγμα χρήσης λογικών τελεστών, δυαδικοί τελεστές και τελεστές καταχώρησης). Την χρήση πινάκων (δήλωση και δημιουργία), και συμβολοσειρών (Strings) δήλωση και δημιουργία.

Οι δομές ελέγχου προγράμματος που περιγράφονται στα πλαίσια των μαθημάτων εκμάθησης της γλώσσας προγραμματισμού Java περιλαμβάνουν τις παρακάτω δομές:

- Δομές επιλογής
    - ✓ If - else
    - ✓ Switch - case
- Δομές επανάληψης
    - ✓ For
    - ✓ While
    - ✓ Do – while

Τις εντολές διακλάδωσης

- Break
- Continue
- Return

Κεφάλαιο 6 : Συμπεράσματα

Στο κεφάλαιο αυτό παρουσιάζονται τα συμπεράσματα και η βιβλιογραφία της πτυχιακής.

Κεφάλαιο 2

Υπολογιστές και Εκπαίδευση

Η συμβατική εκπαιδευτική μέθοδος παρουσιάζει πολλά μειονεκτήματα όταν χρησιμοποιείται για τη διδασκαλία δυναμικών διαδικασιών που υπάρχουν στην Επιστήμη της Πληροφορικής. Αντικείμενα όπως οι δομές δεδομένων και οι αλγόριθμοι απαιτούν την περιγραφή συνεχώς μεταβαλλόμενων πληροφοριών. Η παρουσίαση αυτών των αντικειμένων μπορεί να ολοκληρωθεί, εν μέρει, από έναν εκπαιδευτικό στον πίνακα χρησιμοποιώντας διαγράμματα και σχέδια, αλλά είναι δύσκολο να κατανοηθούν οι έννοιες που παρουσιάζονται από τους φοιτητές. Οι εκπαιδευτικοί συχνά αναζητούν νέους τρόπους για να συμπεριλάβουν στη διάλεξη τους προκειμένου να βελτιώσουν την παρουσίαση σύνθετων και δυναμικά εξελισσόμενων θεμάτων.

Ένα διαλογικό λογισμικό οπτικοποίησης παρέχει μια πιθανή λύση στους περιορισμούς των παραδοσιακών τρόπων διδασκαλίας και εκμάθησης. Μια τέτοια εφαρμογή μπορεί να παρουσιάσει τις δυναμικές πληροφορίες με τρόπο που είναι αδύνατο να παρουσιαστούν με τις παραδοσιακές μεθόδους. Η εφαρμογή οπτικοποίησης έχει ένα πολύ μεγάλο πλεονέκτημα αυτό της επανάληψης. Οι φοιτητές είναι σε θέση να επαναλάβουν τα παραδείγματα της διάλεξης ακριβώς με τον ίδιο τρόπο όπως ακριβώς αυτά παρουσιάστηκαν στην αίθουσα διδασκαλίας. Παρά τον περιορισμένο αριθμό παραδειγμάτων που μπορούν να παρουσιαστούν κατά τη διάρκεια μιας διάλεξης, οι φοιτητές με τη χρήση ενός λογισμικού οπτικοποίησης κατά τη διαδικασία της εκμάθησης, έχουν την ευκαιρία να διερευνήσουν εις βάθος το αντικείμενο της διδασκαλίας. Μια εφαρμογή οπτικοποίησης προσπαθεί να συλλάβει την προσοχή των φοιτητών και να τους καθοδηγήσει βήμα-βήμα στην καλύτερη κατανόηση.

Οι φοιτητές εμφανίζονται να κατανοούν τον τρόπο της παρουσίασης που πραγματοποιήθηκε κατά τη διάρκεια της διάλεξης, αλλά πρέπει να καταβάλλουν εκ νέου προσπάθεια για να κατανοήσουν πραγματικά τις πληροφορίες από τις σημειώσεις τους. Δεδομένου ότι οι σημειώσεις παρουσιάζουν μια στατική απεικόνιση

ενός δυναμικού γεγονότος, με αποτέλεσμα συχνά να γίνεται πηγή απογοήτευσης και παρανόησης. Αντίθετα από τους εκπαιδευτικούς, τα βιβλία δεν μπορούν να ρωτηθούν για πρόσθετες ή εναλλακτικές εξηγήσεις. Σαφώς, οι συμβατικές μέθοδοι διδασκαλίας συχνά δεν είναι ο πλέον αποδοτικός τρόπος για να διδαχτούν εξ ορισμού δυναμικά αντικείμενα.

Οι εκπαιδευτικοί συνεχώς επιδιώκουν αναζητώντας νέους τρόπους για να βελτιώσουν τη διδασκαλία, να διευκολύνουν τη μάθηση, και να κρατήσουν την προσοχή του ακροατηρίου τους αμείωτη. Η δυνατότητα των υπολογιστών να αποθηκεύουν απέραντες ποσότητες πληροφοριών, να προσομοιάσουν περιβάλλοντα και καταστάσεις που με άλλους τρόπους δεν θα ήταν διαθέσιμη, τους καθιστά ένα χρήσιμο εργαλείο ως βοηθητικό μέσο στην ενίσχυση της εκπαιδευτικής διαδικασίας.

Η χρήση των εικόνων και της δυναμικής παρουσίασης μεταβαλλόμενων γεγονότων ως εκπαιδευτικά βοηθήματα είναι αποδεκτή πρακτική. Τα εγχειρίδια γεμίζουν με εικόνες, ο εκπαιδευτικός συχνά σχεδιάζει διάφορα διαγράμματα στον πίνακα για να διευκολύνει την επεξήγηση. Η χρήση της δυναμικής παρουσίασης πηγαίνει ένα βήμα παραπέρα. Ενώ η στατική απεικόνιση έχει την δυνατότητα να παρέχει στον κόσμο την ουσία, του πως φαίνεται κάτι, ή την σύνθεσή του, η δυναμική παρουσίαση έχει καλύτερα αποτελέσματα, διότι παρουσιάζει την επεξήγηση δυναμικά εξελισσόμενων διαδικασιών.

Παρά την δραματική αύξηση στη χρήση της τεχνολογίας στην κοινωνία κατά τη διάρκεια των προηγούμενων δεκαετιών, η χρήση της τεχνολογίας στις αίθουσες διδασκαλίας δεν ακολούθησαν την ίδια συγκλονιστική εξέλιξη. Αν και η χρήση των υπολογιστών έχει ενσωματωθεί στα προγράμματα σπουδών, στις διαλέξεις χρησιμοποιείται περισσότερο ο πίνακας και η κιμωλία. Ο προβολέας (OH-projector) είναι ίσως το πιο συχνά χρησιμοποιημένος τεχνολογικός εξοπλισμός στις σημερινές αίθουσες διδασκαλίας.

Η απουσία της τεχνολογίας οφείλεται, εν μέρει, στις αποτυχημένες προσπάθειες, όπως η εισαγωγή της εκπαιδευτικής τηλεόρασης. Εξαίρεση στη χρήση της τεχνολογίας αποτελεί η εξ’ αποστάσεως εκπαίδευση. Είναι δύσκολο να προβλεφθεί το μέλλον της εκπαίδευσης, ίσως η αίθουσα διδασκαλίας ποτέ να μην γίνει ο χώρος

όπου η τεχνολογία θα παίζει σημαντικό ρόλο, στην πραγματικότητα όμως η ίδια η τάξη μπορεί να γίνει όλο και λιγότερο σημαντική στην εκπαίδευση στο μέλλον.

Εκτός από την αλλαγή του χαρακτήρα της παραδοσιακής τάξης, η τεχνολογία έχει την δυνατότητα να επηρεάσει και μια άλλη διάσταση της εκπαιδευτικής διαδικασίας, τα βιβλία. Η υλοποίηση φθηνών ηλεκτρονικών υπολογιστών με αυξημένες γραφικές δυνατότητες κατέστησε εφικτό στο εκπαιδευτικό λογισμικό να περιληφθούν και τα γραφικά μαζί με το κείμενο. Στο πρόσφατο παρελθόν οι ηλεκτρονικοί υπολογιστές με ενσωματωμένες δυνατότητες χρήσης τεχνολογιών πολυμέσων έδωσαν την ευκαιρία στους σχεδιαστές εκπαιδευτικού λογισμικού να περιλάβουν και τον ήχο, το βίντεο αλλά και την δυνατότητα των κινούμενων σχεδίων στις εφαρμογές τους. Παρά την δημιουργία και χρήση εφαρμογών εκπαιδευτικού λογισμικού για αρκετές δεκαετίες, η εκπαιδευτική διαδικασία δεν απέβαλε την ανάγκη για τα παραδοσιακά έντυπα υλικά. Ίσως το έντυπο υλικό να παραμείνει ένα βασικό στοιχείο και στο μέλλον της εκπαίδευσης. Η σημασία του εκπαιδευτικού λογισμικού περιορίζεται μόνο ως συμπληρωματικό εργαλείο. Γίνεται αποδεκτό ο βοηθητικός ρόλος του και στην τάξη αλλά και στο βιβλίο.

Το εκπαιδευτικό λογισμικό έχει τη δυνατότητα να παράσχει ένα διερευνητικό περιβάλλον, στο οποίο οι φοιτητές μπορούν να μάθουν μέσω πειραματισμού. Επιπλέον, αν το κατάλληλα σχεδιασμένο εκπαιδευτικό λογισμικό μπορεί να είναι ιδιαίτερα διαλογικό, έναντι του συνήθως παθητικού περιβάλλοντος μιας παραδοσιακής διάλεξης στην τάξη. Τέλος, με την ανάπτυξη τεχνολογιών πολυμέσων, το εκπαιδευτικό λογισμικό μπορεί να γίνει ένα ιδιαίτερα οπτικοποιημένο εκπαιδευτικό περιβάλλον εφόσον περιλαμβάνει και την δυνατότητα της δυναμικής παρουσίασης.

2.1 Οπτικοποίηση Προγραμμάτων

Ένα σημαντικό κομμάτι της εκπαίδευσης με την βοήθεια Η/Υ είναι η διδασκαλία του προγραμματισμού των Η/Υ. Η ανάπτυξη τέτοιων διδακτικών εφαρμογών βοηθούν τους φοιτητές στην εκμάθηση μιας γλώσσας προγραμματισμού για την επίλυση προβλημάτων. Οι περισσότεροι εκπαιδευτικοί κατά τη διάρκεια της διδασκαλίας προσπαθούν να προωθήσουν την ενεργή μάθηση, να λάβουν υπόψη τους τις ανάγκες των φοιτητών σε κάθε βήμα της εκπαιδευτικής διαδικασίας, να αλληλεπιδρούν με το

κοινό, να δημιουργούν νέα προβλήματα (τμήμα κώδικα), να το επιλύουν, για να εισπράξουν την ανατροφοδότηση της διαδικασίας με την συμμετοχή του κοινού.

Η οπτικοποίηση προγράμματος είναι η απεικόνιση του κώδικα ή των δομών δεδομένων ενός προγράμματος, είτε με στατική είτε με δυναμική μορφή. Ένα παράδειγμα στατικής οπτικοποίησης δομής δεδομένων είναι η παρουσίαση των περιεχομένων μιας συνδεμένης λίστας ως διάγραμμα, ενώ στην περίπτωση της δυναμικής οπτικοποίησης το ίδιο διάγραμμα με τα περιεχόμενα και τα βέλη αλλάζει δυναμικά το περιεχόμενο της λίστας κατά την εκτέλεση του προγράμματος. Ένα παράδειγμα στατικής οπτικοποίησης κώδικα είναι ο συνδυασμός της τεκμηρίωσης των προγραμμάτων και του πηγαίου κώδικα.

Μια εφαρμογή που έχει ως αντικείμενο τη δυναμική παρουσίαση προγράμματος (program animator) επιτρέπει την παρακολούθηση της εκτέλεσης του πηγαίου κώδικα ενός προγράμματος. Παρέχει μια άποψη της διαδικασίας εκτέλεσης του προγράμματος και εμφανίζει τις αλλαγές των μεταβλητών κατά τη διάρκεια της εκτέλεσης. Αυτή η διαδικασία μοιάζει με την αναζήτηση λαθών στο πρόγραμμα (debugging) που χρησιμοποιούνται για την ανάπτυξη λογισμικού, οι εφαρμογές δυναμικής παρουσίασης προγραμμάτων σχεδιάζονται ειδικά για εκπαιδευτικούς σκοπούς. Μερικά από τα μοναδικά χαρακτηριστικά γνωρίσματα αυτών των εφαρμογών περιλαμβάνει την δυνατότητα της αντίστροφης εκτέλεσης (reverse execution) σε οποιοδήποτε σημείο, την αυτόματη παρουσίαση των τιμών των μεταβλητών, και ειδικών τρόπων βηματικής εκτέλεσης ώστε να ενθαρρύνεται η πρόβλεψη της συμπεριφοράς του προγράμματος από τους φοιτητές.

Οι σχεδιαστές τέτοιων εφαρμογών ελπίζουν ότι η οπτικοποίηση ενός μέρους του δεδομένου προβλήματος, θα βοηθήσει τους χρήστες να αναπτύξουν πιο γρήγορα ένα πετυχημένο νοητικό μοντέλο. Γενικά, αναμένεται ότι η οπτικοποίηση θα βοηθήσει αρκετά τη διαδικασία κατανόησης. Η ανάγνωση πηγαίου κώδικα ή κειμένου είναι μια ειδική περίπτωση αυτής της οπτικής επεξεργασίας. Η λεπτομέρεια που περιλαμβάνεται είναι μεγάλη και ο εγκέφαλος θα συνοψίσει κάθε χαρακτήρα, λέξη, πρόταση σε μια εσωτερική έννοια ή μια παράσταση. Η οπτικοποίηση λογισμικού προσπαθεί να βοηθήσει την διαδικασία κατανόησης με την παροχή αυτών των αφαιρέσεων σε μια οπτική μορφή, μειώνοντας κατά συνέπεια το φορτίο ερμηνείας.

Η έρευνα δείχνει ότι οι αρχάριοι προγραμματιστές δεν έχουν ένα αποτελεσματικό πρότυπο ενός υπολογιστή, και ότι το πρότυπο αυτό πρέπει να διδαχθεί. Επιπλέον, οι λεπτομέρειες της εκτέλεσης ενός προγράμματος παραμένουν κρυμμένες στα τυπικά περιβάλλοντα προγραμματισμού - αυτό κρύβει την σημασιολογία των γλωσσών προγραμματισμού και συμβάλλει στη καμπύλη εκμάθησης των αρχάριων προγραμματιστών.

Όταν το αντικείμενο της οπτικοποίησης είναι το λογισμικό, δίνεται έμφαση στο ρόλο ολοκλήρωσης της διαδικασίας προγραμματισμού, αρχίζοντας από τις προδιαγραφές των βασικών απαιτήσεων. Αν δίνεται έμφαση κατά τη διαδικασία οπτικοποίησης σε θέματα υψηλού επιπέδου, και δεν ασχολείται με τις λεπτομέρειες εκτέλεσης μιλάμε για οπτικοποίηση αλγορίθμου. Η οπτικοποίηση προγράμματος χρησιμοποιεί τις κειμενικές (textual) και γραφικές αναπαραστάσεις για να μεταβιβάσει την συμπεριφορά κατά τη εκτέλεση σε πραγματικό χρόνο ενός προγράμματος.

Η οπτική αναπαράσταση της λειτουργίας ενός προγράμματος εξετάζει και τις δύο ανησυχίες: με την παρουσίαση της "ιδανικής" απεικόνισης, βοηθά τους αρχάριους στην κατασκευή του σωστού προτύπου ενός υπολογιστή, ή διαχωρίζει το πρότυπό του με το σωστό πρότυπο. Με το να εμφανίζει την επίδραση κάθε εντολής του προγράμματος σε αυτήν την απεικόνιση, εξηγεί την κρυμμένη σημασιολογία μιας γλώσσας προγραμματισμού. Διάφορα συστήματα έχουν δημιουργηθεί για την δυναμική παρουσίαση της λειτουργίας των προγραμμάτων. Μερικά από τα συστήματα αυτά παρουσιάζουν την εκτέλεση ενός προγράμματος πρωτίστως από την άποψη της ροής ελέγχου του προγράμματος , η οπτικοποίηση αποτελείται από την έμφαση στις γραμμές του κώδικα.

Μια αναπαράσταση με κίνηση είναι σε θέση να βοηθήσει, αν μειώσει την απαιτούμενη γνωστική προσπάθεια, αν αποδίδει μεγαλύτερη δυνατότητα επίλυσης προβλημάτων και αν διευκολύνει την διεξαγωγή σωστών ερμηνειών που βγαίνουν από την αναπαράσταση. Αυτό μας βοηθάει στην απόκτηση ενός πλαισίου που συνδέει την εξωτερική αναπαράσταση με τα εσωτερικά διανοητικά πρότυπα στον παρατηρητή και δείχνει τα παιδαγωγικά οφέλη που μπορούν να υπάρξουν.

Η σύνδεση της εξωτερικής και της εσωτερικής αναπαράστασης δεν αποτελεί από μόνη της μια αυτόματη διαδικασία. Η αναζήτηση ενός πλαισίου που να δείχνει τα

«σφάλματα ομοιότητας», δηλ. το ότι η εξωτερική αναπαράσταση που χρησιμοποιείται σε μια παρουσίαση ίσως να μην ταιριάζει με το εσωτερικό πρότυπο που δημιουργείται από το θεατή. Η σωστή σχεδίαση για την εξωτερική παρουσίαση είναι απαραίτητη ώστε η εσωτερίκευση στον θεατή να είναι σωστή. Αν η παρουσίαση είναι σωστή, μειώνεται η γνωστική προσπάθεια εκ μέρους του θεατή.

Η γνωστική προσπάθεια μειώνεται όσο η δυναμική παρουσίαση ξεκάθαρα εμποδίζει την παρακολούθηση πολλών λεπτομερών πληροφοριών και επιτρέπει τον φοιτητή να επικεντρωθεί στα πιο ουσιώδη σημεία. Τα νέα πρότυπα και χαρακτηριστικά δεν είναι εύκολα ορατά αν δεν πραγματοποιηθεί ανάλυση της συγκεκριμένης περιοχής του αλγορίθμου που παρουσιάζεται. Οι αρχάριοι δεν έχουν την ίδια εμπειρία και δεν βλέπουν τα ίδια οφέλη, ενώ βλέπουν την ίδια εξωτερική απεικόνιση το πρόβλημα της διαφοράς εμφανίζεται στην μεταφορά στα δικά τους εσωτερικά πρότυπα. Μπορεί το υπάρχον πλαίσιο γνώσης του αρχαρίου να εμποδίζει τις νέες πληροφορίες. Μπορεί η πληροφορία να παρουσιάζεται τόσο απλά ώστε ο παρατηρητής να προσέχει μόνο παθητικά την δυναμική παρουσίαση και δεν συνδέει τις νέες πληροφορίες με την υπάρχουσα γνώση. Η νέα απεικόνιση μπορεί απλά να μην είναι κατανοητή για τον αρχάριο.

Η μελέτη της δυναμικής παρουσίασης πρέπει να αντιμετωπιστεί με διαφορετικό τρόπο από ότι η στατικές εικόνες. Ειδικά το πώς καταλαβαίνουμε και το πώς κάνουμε συνδέσεις μεταξύ των στατικών και δυναμικών παρουσιάσεων που αντιπροσωπεύουν τις δυναμικές διαδικασίες των πραγματικών συστημάτων, ώστε να μπορούμε να διεξάγουμε συμπεράσματα και για τις δύο περιπτώσεις. Η άποψη της ανωτερότητας της δυναμικής παρουσίασης έναντι των στατικών εικόνων είναι η ορατή χρονική ανάλυση της μετακίνησης αντικειμένων, που καθιστά την αλλαγή των καταστάσεων προφανέστερη. Παιδαγωγικά, είναι ασαφές πως μια δυναμική παρουσίαση μπορεί να διευκολύνει την εκμάθηση ή την επίλυση προβλημάτων, δηλ. πως μπορεί να επιτευχθεί καλύτερο επίπεδο κατανόησης από την παρακολούθηση της δυναμικής παρουσίασης αντίθετα από το να φανταστούμε την κίνηση. Καλαίσθητες γραφικές αναπαραστάσεις, που χρησιμοποιεί νέους τρόπους και επιτρέπει στον καθένα να αντιληφτεί μια διαδικασία, οι σχεδιαστές συστημάτων οπτικοποίησης προωθούν συστήματα απεικόνισης ανάλογα με τη δική τους αισθητική ευχαρίστηση.

Ένα αισθητικά ευχάριστο διαλογικό γραφικό σύστημα, δεν σημαίνει απαραιτήτως ότι συμβάλλει και στην εκμάθηση.

Ένα περιβάλλον που πραγματικά παρακινεί επηρεάζει θετικά την εκμάθηση. Τα παιχνίδια των υπολογιστών μπορούν να θεωρηθούν λόγω των μοναδικών ικανοτήτων των υπολογιστών, ότι μπορούν να χρησιμοποιηθούν για την δημιουργία ενός περιβάλλοντος παρακίνησης. Από τη μελέτη της αλληλεπίδρασης με τα παιχνίδια υπολογιστών αποκτήθηκε η γνώση για τα βασικά χαρακτηριστικά γνωρίσματα ενός περιβάλλοντος παρακίνησης.

Βρέθηκαν ως βασικά χαρακτηριστικά τρία γενικά γνωρίσματα : πρόκληση (challenge), φαντασία (fantasy), περιέργεια (curiosity).

Πρόκληση: δεδομένου ότι ένα καλό εργαλείο σχεδιάζεται για να πετύχει στόχους, το οποίο ήδη έχει περιγραφεί ως εξωγενής στόχος, επιπλέον από την έκβαση του εξωτερικού στόχου που είναι αβέβαιος, το ίδιο το εργαλείο πρέπει να είναι αξιόπιστο και αποδοτικό. Η αυτοεκτίμηση διαδραματίζει ένα σημαντικό ρόλο στην πρόκληση, διότι ο αυτοσεβασμός κάποιου μπορεί να επιβραβευτεί θετικά ή αρνητικά σύμφωνα με την κατάλληλη ανάδραση που επιτυγχάνουν οι φοιτητές. Και η περίπτωση της αποτυχίας πρέπει να αντιμετωπιστεί κατά τέτοιο τρόπο ώστε η δυνατότητα της ζημιάς της αυτοσεβασμού να ελαχιστοποιηθεί.

Φαντασία: διανοητικές εικόνες των πραγμάτων, χωρίς την παρουσία αισθήσεων ή την πραγματική εμπειρία ενός ατόμου, καθορίζει τρεις διαστάσεις:

- *Εγγενής εναντίον εξωγενής* (*intrinsic vs. extrinsic*). Στην εξωγενή η φαντασία εξαρτάται από την χρήση των ικανοτήτων, ενώ στην εγγενή ότι δεν εξαρτάται μόνο από τις ικανότητες, αλλά η ικανότητες εξαρτώνται από την φαντασία.
- *Γνωστικές πτυχές* (*cognitive aspects*). Ο βαθμός που υιοθετούνται οι μεταφορές και αναλογίες που εμπλέκονται στην παλιά γνώση για να βοηθήσουν την απόκτηση νέων γνώσεων.
- *Συναισθηματικές πτυχές* (*emotional aspects*). Ο βαθμός στον οποίο εκπληρώνουν τις συναισθηματικές ανάγκες των φοιτητών

Περιέργεια: Προκειμένου να προκληθεί η περιέργεια σε ένα μαθησιακό περιβάλλον, δεν πρέπει ούτε να περιπλεχτεί ούτε να είναι απλή όσον αφορά την υπάρχουσα γνώση του μαθητευόμενου. Η περιέργεια διακρίνεται σε δύο τύπους στην αισθητήρια και στην γνωστική. Η αισθητήρια περιέργεια (sensory curiosity) περιλαμβάνει τον βαθμό προσέλκυση της προσοχής στις μεταβολές των χρωμάτων, του ήχου, ή άλλων ερεθισμάτων των αισθητηρίων οργάνων σε ένα περιβάλλον. Σε αντίθεση η γνωστική περιέργεια (cognitive curiosity) μπορεί να θεωρηθεί ως επιθυμία να προσκομίσει καλύτερη μορφή στις δομές γνώσης κάποιου, η επιθυμητή μορφή είναι μια κατάσταση όπου η γνώση είναι πλήρες, συνεπής και φειδωλή.

Στην επιστήμη της πληροφορικής και των μαθηματικών οι έννοιες παρουσιάζονται συχνά με τη χρήση εικόνων. Οι δομές δεδομένων, παρουσιάζονται χρησιμοποιώντας συγκεκριμένους τρόπους αναπαράστασης για να βοηθήσουν την κατανόηση, όπως τα δέντρα (trees) που παρουσιάζονται ως κόμβοι που συνδέονται με βέλη, όπως οι λίστες (lists) που παρουσιάζονται σαν τετράγωνα κουτιά που συνδέονται με δείκτες σε μια σειρά, όπως οι πίνακες (arrays) που παρουσιάζονται ως ένα κουτί που περιλαμβάνει μια σειρά από δεδομένα. Τέτοιες απεικονίσεις, που συνδέονται με τις λεκτικές εξηγήσεις, είναι ο καλύτερος τρόπος για να εξηγηθεί η δυναμική των αλγορίθμων. Μελέτες των ψυχολόγων που εξέτασαν τον τρόπο με τον οποίο οι άνθρωποι χτίζουν και διατηρούν τις έννοιες, υποστηρίζουν ότι η χρησιμοποίηση δυο ή περισσότερων διαφορετικών τρόπων παρουσίασης της ίδιας ιδέας βοηθάει στην απομνημόνευση και τον χειρισμό των εννοιών δημιουργικά.

Αυτό σημαίνει ότι μια λεκτική και μια οπτική κωδικοποίηση ενισχύει αμοιβαία η μια την άλλη και ο συνδυασμός τους είναι η καλύτερη προσέγγιση για λόγους διδασκαλίας. Υπάρχουν πολλές προσπάθειες που κατευθύνονται στο να εξερευνήσουν πόσο καλύτερα μεταβιβάζονται οι πληροφορίες από έναν υπολογιστή σε έναν φοιτητή προκειμένου να αποφευχθεί η οπτική υπερφόρτωση. Οι σημαντικές πληροφορίες μπορούν να δοθούν καλύτερα χρησιμοποιώντας κάποιο σήμα που ενεργοποιεί ένα τρίτο αισθητήριο κανάλι, διότι η προσοχή αυξάνεται από ένα σήμα που έρχεται σε αντίθεση με το φόντο. Τέτοια τεχνάσματα μπορούν να χρησιμοποιηθούν στον σχεδιασμό συστημάτων οπτικοποίησης των αλγορίθμων ως οπτικά σήματα που κατευθύνουν την προσοχή του φοιτητή στα ενδιαφέροντα μέρη της προσομοίωσης.

Εξετάζοντας γενικά τους αλγορίθμους, οι αλγόριθμοι που αλλάζουν συνεχώς τις καταστάσεις των δεδομένων καθώς εκτελούνται είναι τα πιο κατάλληλα για οπτικοποίηση. Το χρονικό διάστημα μεταξύ της αρχικής κατάστασης n, και της επόμενης κατάστασης μετά από την εκτέλεση μιας γραμμής του κώδικα όπου τα δεδομένα έχουν αλλάξει n+1, είναι το σημείο όπου ο παρατηρητής θέλει να δει τι έχει συμβεί και περισσότερο τον ενδιαφέρει η αλλαγή πως έχει συμβεί.

2.1.1 Ποιος θα χρησιμοποιήσει αυτά τα προγράμματα;

Αν και η απάντηση μπορεί να φαίνεται προφανή, αν αναφέρουμε ότι η μορφή της εφαρμογής είναι π.χ. applet, βασικά το μυαλό μας πάει στο μοναδικό τρόπο που μπορούμε να τα δούμε, στο Διαδίκτυο. Έτσι μπορούμε να πούμε ότι μόνο οι άνθρωποι που έχουν κάποια δυνατότητα πρόσβασης στο Διαδίκτυο θα μπορέσουν να το χρησιμοποιήσουν.

2.1.2 Ποια προαπαιτούμενη γνώση είναι απαραίτητη;

Μια άλλη παράμετρος που πρέπει να ληφθεί υπόψη είναι η δυνατότητα της κατανόησης νέων γνώσεων των φοιτητών. Η δυνατότητα χρησιμοποίησης μιας εφαρμογής οπτικοποίησης προγραμμάτων είναι ένα μέτρο το οποίο δείχνει την ευκολία εκμάθησης και χρήσης του συστήματος. Λόγω της μεγάλης απόκλισης στις δυνατότητες κατανόησης των φοιτητών, θα πρέπει να υποθέσουμε ότι η εφαρμογή σχεδιάζεται για αυτούς που έχουν τις λιγότερες δυνατότητες κατανόησης.

2.1.3 Τι θα αποκομίσει ο φοιτητής με τη χρησιμοποίηση της εφαρμογής οπτικοποίησης;

Ο θεμελιώδης στόχος που έχουν οι εφαρμογές οπτικοποίησης είναι η εκπαιδευτική αξία (educational value). Αλλά τι σημαίνει αυτό πραγματικά; Δεν υπάρχει λόγος να χρησιμοποιήσεις κάτι από το οποίο δεν μαθαίνεις τίποτα. Η πρόθεση αυτών των εφαρμογών είναι να βοηθήσει τους φοιτητές να κατανοήσουν πως λειτουργεί ένας αλγόριθμος ή ένα κομμάτι ενός αλγορίθμου βοηθώντας με την οπτική αναπαράσταση των εντολών του επαναλαμβανόμενου κώδικα. Αν ο φοιτητής μπορεί να κατανοήσει τη βασική ιδέα της λειτουργίας του αλγορίθμου βλέποντας κάποια οπτική αναπαράσταση της, τότε αποκτά την ικανότητα να μπορεί να επαναλάβει μόνος του τον αλγόριθμο και να δημιουργήσει τον παραλληλισμό της εκτέλεσης που

παρακολούθησε με τον τρόπο που αποτυπώθηκε η λειτουργία του αλγορίθμου στη μνήμη του. Με την αύξηση της αυτοπεποίθησης οι φοιτητές αισθάνονται πιο σίγουροι για τις δυνατότητές τους να εφαρμόσουν τις τεχνικές των αλγορίθμων σε καθημερινές καταστάσεις.

2.1.4 Ενδιαφέροντα γεγονότα

Οι περισσότερες επιδείξεις οπτικοποίησης προγράμματος (είτε είναι στατικές είτε είναι δυναμικές απεικονίσεις του κώδικα ή των δεδομένων) μπορούν να δημιουργηθούν αυτόματα, ενώ οι περισσότερες οπτικοποιήσεις αλγορίθμων (είτε είναι στατικές απεικονίσεις είτε είναι δυναμικές παρουσιάσεις αλγορίθμων) δεν μπορούν.

2.1.5 Τρόποι απεικόνισης της ανταλλαγής δύο στοιχείων

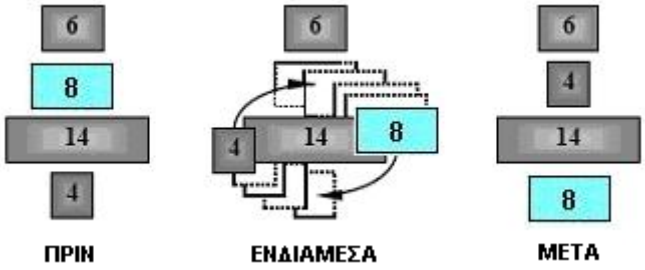
Μια απλή, στατική οπτικοποίηση παρουσιάζει την κατάσταση πριν και μετά από μια ενέργεια, δεν δίνει καμία διευκρίνιση του τρόπου με τον οποίο εκτελέσθηκε η ανταλλαγή των δεδομένων, Σχήμα 2.1.

Σχήμα 2.1: Στατική οπτικοποίηση. Παρουσίαση των δυο καταστάσεων πριν και μετά

Μια δυναμική παρουσίαση που χρησιμοποιεί το πρώτο αξίωμα (επεξηγεί την ροή των δεδομένων), δίνει τις πληροφορίες για αυτό που συμβαίνει: τι είναι η πηγή και ο προορισμός των δεδομένων, Σχήμα 2.2.

Σχήμα 2.2: Ομαλή δυναμική παρουσίαση. Παρουσίαση της ροής των δεδομένων

Μια δυναμική παρουσίαση σύμφωνα με τα τρία αξιώματα (Σχήμα 2.3) κρύβει το χωρίς ενδιαφέρον μεταβλητή (την βοηθητική (auxiliary) μεταβλητή, την εντολή της εκχώρησης), και παρουσιάζει μόνο αυτό που είναι το σημαντικότερο: την ανταλλαγή των τιμών των δυο δεδομένων συγχρόνως, με την παρουσίαση μιας συμμετρικής διαδικασίας. Είναι πιο κοντά στην έννοια αυτού που αποκαλούμε ανταλλαγή δεδομένων.



Σχήμα 2.3: Συγχρονισμένη δυναμική παρουσίαση. Παρουσιάζει την συμμετρική ανταλλαγή των δεδομένων

2.2 Ασύγχρονη Τηλεκπαίδευση

Η Ασύγχρονη Εκπαίδευση δεν απαιτεί την ταυτόχρονη συμμετοχή των μαθητών και των εισηγητών. Οι μαθητές δεν είναι ανάγκη να βρίσκονται συγκεντρωμένοι μαζί στον ίδιο χώρο ή την ίδια χρονική στιγμή. Αντίθετα, μπορούν να επιλέγουν μόνοι τους το προσωπικό τους εκπαιδευτικό χρονικό πλαίσιο και να συλλέγουν το εκπαιδευτικό υλικό σύμφωνα με αυτό. Η ασύγχρονη εκπαίδευση είναι περισσότερο ευέλικτη από την σύγχρονη. Στο είδος αυτό της εκπαίδευσης ανήκει η Αυτοδιδασκαλία, η Ημιαυτόνομη Εκπαίδευση και η Συνεργαζόμενη Εκπαίδευση.

Στην **Αυτοδιδασκαλία** ο εκπαιδευόμενος εκπαιδεύεται μόνος του χρησιμοποιώντας όποιο μέσο κρίνει αυτός κατάλληλο (βιβλία, Internet κλπ.).

Στην **Ημιαυτόνομη Εκπαίδευση** ισχύει ότι και στην Αυτοδιδασκαλία μόνο που υπάρχει και συγκεκριμένο χρονοδιάγραμμα επικοινωνίας με τον υπεύθυνο εκπαιδευτή είτε με φυσική παρουσία στην τάξη, είτε μέσω δικτύου (Internet, E-mail κλπ.) είτε μέσω audio ή/και video conference.

Στην **Συνεργαζόμενη Εκπαίδευση** (Collaborative) εκπαιδευτής και εκπαιδευόμενοι επικοινωνούν ασύγχρονα μεταξύ τους, οι εκπαιδευόμενοι μελετούν στον δικό τους χρόνο, ακολουθούν όμως ένα χρονοδιάγραμμα παράδοσης των εργασιών. Σήμερα, εξ' αιτίας της υψηλής απαιτούμενης δαπάνης για την εύρυθμη λειτουργία μιας υπηρεσίας τηλεκπαίδευσης με διαδραστική βιντεοδιάσκεψη (Interactive Videoconferencing), η οποία ανήκει στην σύγχρονη εκπαίδευση, οι περισσότερες εφαρμογές εκπαίδευσης εξ' αποστάσεως είτε αποφεύγουν τη χρήση παρόμοιων μεθόδων, είτε τις χρησιμοποιούν μόνο συμπληρωματικά. Ο κύριος όγκος του εκπαιδευτικού υλικού διατίθεται στο web (ως απλές σελίδες ή video) και η επικοινωνία του δασκάλου και των διδασκομένων ή των διδασκομένων μεταξύ τους γίνεται ασύγχρονα μέσω email, κάνοντας την εκπαιδευτική διαδικασία να θυμίζει μαθήματα δι' αλληλογραφίας.

Εδώ, ο εκπαιδευτής τοποθετεί κάθε φορά στο web ή αποστέλλει μέσω email την ύλη της ημέρας (ή της εβδομάδας, του μήνα κ.λπ. ανάλογα με τη συχνότητα των μαθημάτων) μαζί με ασκήσεις, ερωτήσεις και οτιδήποτε άλλο θεωρεί χρήσιμο για να κινήσει το ενδιαφέρον των μαθητών ή να ελέγξει την απόδοσή τους. Επίσης, μέσω email (και σπανιότερα μέσω chat) "συνομιλεί" με τους μαθητές του απαντώντας στις ερωτήσεις και τις απορίες τους. Τέλος, οι μαθητές ενθαρρύνονται να δημιουργήσουν τις δικές τους ομάδες εργασίας και να επικοινωνούν μεταξύ τους χωρίς τη μεσολάβηση του εκπαιδευτικού προσωπικού για αλληλοϋποστήριξη, ενημέρωση, ανταλλαγή απόψεων σχετικά με την ύλη ή τον τρόπο διδασκαλίας του μαθήματος κ.λ.π.

Στο ορατό μέλλον οι τεχνικές και οι οικονομικές εξελίξεις δεν προδιαγράφουν σημαντικές αλλαγές σε αυτό το μοντέλο ασύγχρονης τηλεκπαίδευσης το οποίο τείνει να εξελιχθεί στον κυρίαρχο τρόπο διδασκαλίας των επόμενων ετών ή ίσως των επόμενων δεκαετιών. Στις επόμενες παραγράφους θα δούμε τα πλεονεκτήματα και τα μειονεκτήματα της ασύγχρονης τηλεκπαίδευσης σε σχέση με την παραδοσιακή διδασκαλία

2.2.1 Πλεονεκτήματα της ασύγχρονης τηλεκπαίδευσης σε σχέση με την παραδοσιακή διδασκαλία:

1. **Χαμηλότερο κόστος**, αν και η αρχική προετοιμασία είναι αρκετά πιο χρονοβόρα από εκείνη μιας παραδοσιακής σειράς μαθημάτων, οι συντελεστές (εκπαιδευτής, βοηθοί κ.λ.π.) έχουν πολύ μικρότερο φόρτο εργασίας κατά την επανάληψη του ίδιου μαθήματος σε άλλες τάξεις. Επίσης, οι εκπαιδευόμενοι εξοικονομούν χρόνο, περιορίζοντας τις μετακινήσεις τους από και προς την αίθουσα διδασκαλίας. (Αυτός είναι ο κύριος λόγος για τον οποίο η τηλεκπαίδευση προτιμάται από τις επιχειρήσεις για την εκπαίδευση του προσωπικού τους, αλλά και από τους ίδιους τους εργαζόμενους.)
2. **Μεγαλύτερες ευκαιρίες για άτομα με δυσκολία μετακίνησης**, άτομα με ειδικές ανάγκες, αλλά και κάτοικοι απομονωμένων περιοχών αποκτούν πρόσβαση σε υπηρεσίες εκπαίδευσης οι οποίες ήταν απροσπέλαστες ή εξαιρετικά δαπανηρές με τις παραδοσιακές τεχνολογίες (ένας μαθητής από τα Ψαρά μπορεί να παρακολουθήσει μαθήματα ενός καθηγητή από την Κρήτη χωρίς να είναι απαραίτητη η μετάβασή του εκεί).
3. **Ασυμμετρική διδασκαλία**, κάθε εκπαιδευόμενος παρακολουθεί τα μαθήματα στη χρονική στιγμή που τον εξυπηρετεί καλύτερα. _εν μπορεί βέβαια να έχει απόλυτη ελευθερία (να παρακολουθεί όποτε θέλει). Του παρέχονται όμως περισσότερες επιλογές. Για παράδειγμα, αν τα μαθήματα ανανεώνονται (προστίθεται νέα ύλη) μια φορά την εβδομάδα, ο μαθητής μπορεί να εργαστεί πάνω στο νέο υλικό όποια στιγμή της εβδομάδας επιθυμεί.
4. **Καλύτερη ποιότητα διδακτικού προσωπικού**, προβλήματα μετακίνησης παρουσιάζονται όχι μόνο για τους διδασκόμενους αλλά και για τους εκπαιδευτές. Η τηλεκπαίδευση εξασφαλίζει για τον διοργανωτή του μαθήματος τους καλύτερους εκπαιδευτές οπουδήποτε στον κόσμο και αν βρίσκονται αυτοί. Επίσης, επιτρέπει τη διοργάνωση πιο εξειδικευμένων σεμιναρίων για πολύ ειδικά θέματα. (Π.χ. Ας υποθέσουμε πως απαιτούνται 100 μαθητές για να εξασφαλίσουν την οικονομική βιωσιμότητα ενός σεμιναρίου XML. Στην Αθήνα μπορεί να μην υπάρχουν πάνω από 50 ενδιαφερόμενοι, αλλά σίγουρα σε όλη την Ελλάδα και την Κύπρο θα μπορούσαμε να βρούμε 100 μαθητές για το αντικείμενο αυτό.)

5. **Ταχύτερη προσαρμογή στις ανάγκες των μαθητών και της αγοράς**, επειδή το διδακτικό υλικό είναι γραμμένο σε ηλεκτρονική μορφή είναι ευκολότερη η τροποποίηση του με βάση τις ιδιαίτερες ανάγκες των εκπαιδευομένων (π.χ. μεγαλύτερη έμφαση σε θέματα που απασχολούν τη συγκεκριμένη τάξη), τις πρόσφατες τεχνολογικές εξελίξεις ή την επικαιρότητα (ειδήσεις σχετικές με το μάθημα κ.λ.π.).
6. **Μεγαλύτερη οικογενειακή συνοχή** (στο απώτερο μέλλον), παράλληλα με την τηλεκπαίδευση παρατηρούμε και μια αυξανόμενη τάση για εργασία στο σπίτι (home office, τηλεργασία κ.λ.π.). Μπορούμε λοιπόν να υποθέσουμε πως σε μερικές δεκαετίες παιδιά και γονείς θα περνούν περισσότερο χρόνο μαζί μια και τόσο η εργασία των γονιών όσο και η εκπαίδευση γονιών και παιδιών θα πραγματοποιείται κυρίως στο σπίτι.

2.2.2 Μειονεκτήματα της ασύγχρονης τηλεκπαίδευσης:

1. **Απώλεια των μη γλωσσικών πλευρών της επικοινωνίας**, όσοι έχουν διδάξει σε μια τάξη γνωρίζουν πως μια απλή ματιά είναι αρκετή για να καταλάβει ο εκπαιδευτής ποιοι μαθητές βαριούνται ή είναι κουρασμένοι, πόσοι καταλαβαίνουν τι συζητείται, ποιοι κρατούν προσεκτικά σημειώσεις κ.λ.π. Ο τηλεκπαιδευτής στερείται αυτών των πληροφοριών και δυσκολεύεται περισσότερο να προσαρμόσει το μάθημά του στις ανάγκες των εκπαιδευόμενων.
2. **Λιγότερο αυθόρμητη επικοινωνία**, η τεχνολογία τηλεκπαίδευσης οικοδομεί ένα αόρατο εμπόδιο μεταξύ του δασκάλου και των μαθητών, περιορίζοντας την άμεση έκφραση των συναισθημάτων τους. (Ένα χαμόγελο ή ένα δηκτικό σχόλιο δεν μεταφέρονται αυτόματα μέσα στην "τάξη". Ο μαθητής ή ο δάσκαλος πρέπει να χρησιμοποιήσουν Η/Υ για να τα μεταδώσουν στους γύρω τους και φυσικά αυτή η τεχνική μεσολάβηση μειώνει τον αυθορμητισμό των ενεργειών τους.)
3. **Διαχείριση θεμάτων copyright**, ένα σοβαρό πρόβλημα δημιουργείται από την, ευρύτατα διαδεδομένη σήμερα, χρήση copyrighted υλικού για εκπαιδευτικούς σκοπούς. Η νομοθεσία επιτρέπει σε ένα καθηγητή να φωτοτυπήσει μερικά αποσπάσματα ενός βιβλίου και να τα μοιράσει στην τάξη του. Η τοποθέτηση όμως αυτού του υλικού στο web (έστω και σε site

προστατευμένο με password) θεωρείται δημόσια μετάδοση και δεν μπορεί να γίνει χωρίς αποζημίωση. Ένα άλλο πρόβλημα ανακύπτει από την ανάγκη προστασίας της εργασίας του εκπαιδευτή. Τα μαθήματα που δημιούργησε για την τάξη (στο web ή μέσω email) μπορούν εύκολα να αντιγραφούν και να αναμεταδοθούν σε όλο το δίκτυο χωρίς την άδειά του, ζημιώνοντάς τον οικονομικά. Το πρόβλημα αυτό είναι ήδη σημαντικό και προβλέπεται σύντομα να λάβει εκρηκτικές διαστάσεις.

4. **Χαμηλότερη κοινωνικοποίηση**, οι εκπαιδευόμενοι έχουν λιγότερες προσωπικές επαφές με το δάσκαλο και τους συμμαθητές τους. Υπάρχει λοιπόν ο κίνδυνος πως η εξ' αποστάσεως εκπαίδευση θα επηρεάσει αρνητικά τις ικανότητες των παιδιών στις διαπροσωπικές σχέσεις.
5. **Δυσκολία ελέγχου της ταυτότητας του εκπαιδευόμενου**, είναι γνωστό πως μερικοί δίδυμοι εκμεταλλεύονται την ομοιότητά τους για να δίνει ο ένας εξετάσεις στη θέση του άλλου. Αντίστοιχα φαινόμενα μπορούν να εμφανιστούν και στο χώρο της τηλεκπαίδευσης όπου η πλαστοπροσωπία είναι πολύ πιο εύκολη.

Κεφάλαιο 3

Οπτικοποίηση Λογισμικού

Η ιδέα της απεικόνισης προγραμμάτων Η/Υ είναι σχεδόν το ίδιο παλιά όσο και ο ίδιος ο προγραμματισμός.

3.1 Εισαγωγή στην οπτικοποίηση λογισμικού

Πολλοί διαφορετικοί τρόποι γραφικών απεικονίσεων έχουν αναπτυχθεί. Μαζί με την ανάπτυξη των γραφικών τερματικών, εμφανίστηκαν και συστήματα για τη δημιουργία οπτικοποιήσεων με τη βοήθεια των υπολογιστών από τις αρχές της δεκαετίας του ’50. Τα συστήματα χρησιμοποιούσαν υπολογιστές για την αυτόματη δημιουργία στατικών διαγραμμάτων από τα προγράμματα. Τα περισσότερα από εκείνα τα συστήματα βοηθούσαν για την επεξήγηση του κώδικα του προγράμματος. Η αυτοματοποίηση της οπτικοποίησης των δεδομένων των προγραμμάτων ήταν ένας πιο δύσκολος στόχος, αλλά μερικές προσπάθειες προς εκείνη την κατεύθυνση έγιναν επίσης αρκετά νωρίς. Με την εμφάνιση της τεχνολογίας των γραφικών στους Ηλεκτρονικούς Υπολογιστές στο τέλος της δεκαετίας του ’70, και στης αρχές της δεκαετίας του ’80 παρουσιάστηκε μια νέα ευκαιρία για την καλύτερη παρουσίαση των αλγορίθμων. Η δεκαετία του '80 ήταν μια πολύ ενεργή δεκαετία για το σχεδιασμό συστημάτων οπτικοποίησης λογισμικού, και η ανάπτυξη των συστημάτων συνεχίζεται μέχρι και σήμερα με αμείωτο ενδιαφέρον. Η δυνατότητα της εκτέλεσης προγραμμάτων σε ένα γραφικό πολυδιάστατο χώρο προκάλεσε τον ενθουσιασμό μέχρι το σημείο που τα οφέλη των απεικονίσεων εγκωμιάστηκαν απερίσκεπτα.

Συχνά δηλώθηκε ότι οι ανθρώπινες αισθήσεις είναι καταλληλότερες για τις οπτικές πληροφορίες, ή ότι η γραφική απεικόνιση κάνει την καλύτερη χρήση του ανθρώπινου εγκεφάλου επειδή διεγείρει και τα δύο ημισφαίρια του. Η γραφική απεικόνιση θεωρήθηκε πανάκεια για τα προβλήματα που υπάρχουν στην εκπαίδευση της επιστήμης της πληροφορικής. Κατά τις αρχές της δεκαετίας του ’90 η υπερβολή για την αξία της γραφικής απεικόνισης άρχισε να κοπάζει και υπήρξαν ακόμη και μερικές αρκετά αιχμηρές εκφράσεις προς την αντίθετη κατεύθυνση.

Στις αρχές της δεκαετίας του ‘80 οι ερευνητές άρχισαν να δημιουργούν συστήματα για να απεικονίσουν γραφικά, προγράμματα υπολογιστών και αλγορίθμους, χρησιμοποιώντας νέους τερματικούς σταθμούς με γραφικές δυνατότητες. Η παρουσίαση των αλγορίθμων μεταφέρθηκε από την έντυπη μορφή στις οθόνες των Η/Υ, με την εμφάνιση των πρώτων συστήματα Η/Υ που επέτρεπαν τη δημιουργία των αποκαλούμενων δυναμικών παρουσιάσεων αλγορίθμων, δηλ. την δυναμική απεικόνιση αλγορίθμων εν’ δράση.

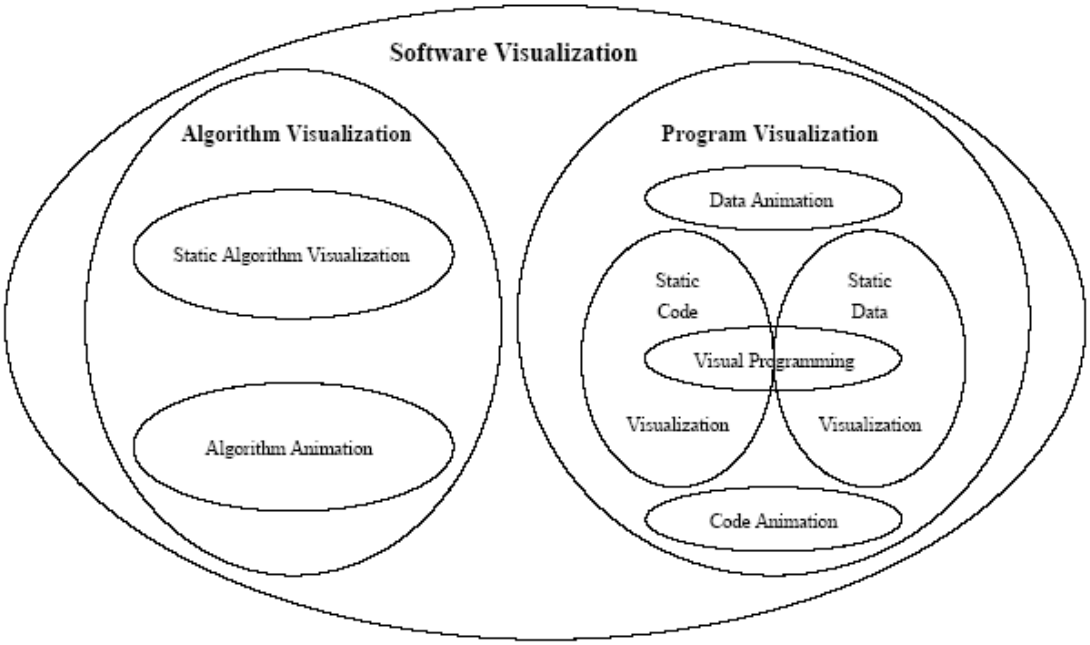
Η οπτικοποίηση στα συστήματα υπολογιστών είναι ένα κοινό φαινόμενο. Οι περισσότεροι επαγγελματίες στους υπολογιστές χρησιμοποιούν την οπτικοποίηση ως εργαλείο για να συνθέσουν ένα κατανοητό διανοητικό πρότυπο ενός συστήματος. Σε αυτό το περιβάλλον μια οπτικοποίηση δημιουργείται με τη νοητική απεικόνιση (φαντασία) του κάθε ατόμου, είναι η αφηρημένη κατανόηση ενός συστήματος με το οποίο οργανώνεται ορθολογικά η γνώση.

Η οπτικοποίηση λογισμικού (software visualization) ταιριάζει περισσότερο στον τρόπο λειτουργίας του ανθρώπινου εγκεφάλου, κατά την επεξεργασία, διαχείριση και αναγνώριση εικόνων και οπτικών δομών. Η οπτικοποίηση λογισμικού και οι διάφορες συνιστώσες της προσπαθούν να επιδράσουν σ’ αυτή τη έμφυτη ικανότητα δημιουργώντας γραφικές απεικονίσεις των συστημάτων λογισμικού. Οι σχεδιαστές τέτοιων εφαρμογών ελπίζουν ότι η οπτικοποίηση ενός μέρους του δεδομένου προβλήματος, θα βοηθήσει τους χρήστες να αναπτύξουν πιο γρήγορα ένα πετυχημένο νοητικό μοντέλο. Γενικά, αναμένεται ότι η οπτικοποίηση θα βοηθήσει αρκετά τη διαδικασία κατανόησης. Η ανάγνωση πηγαίου κώδικα ή κειμένου είναι μια ειδική περίπτωση αυτής της οπτικής επεξεργασίας. Η λεπτομέρεια που περιλαμβάνεται είναι μεγάλη και ο εγκέφαλος θα συνοψίσει κάθε χαρακτήρα, λέξη, πρόταση σε μια εσωτερική έννοια ή μια παράσταση. Η οπτικοποίηση λογισμικού προσπαθεί να βοηθήσει την διαδικασία κατανόησης με την παροχή αυτών των αφαιρέσεων σε μια οπτική μορφή, μειώνοντας κατά συνέπεια το φορτίο ερμηνείας.

Η προσπάθεια εύρεσης κατάλληλου ορισμού, και η μελέτη της σχετικής βιβλιογραφίας φέρνει μπροστά μας και άλλους όρους που σχετίζονται με την οπτικοποίηση λογισμικού, όπως η οπτικοποίηση προγραμμάτων (program visualization), η οπτικοποίηση αλγορίθμων (algorithm visualization), δυναμική

παρουσίαση αλγορίθμου (algorithm animation), και ο οπτικός προγραμματισμός (visual programming).

Η οπτικοποίηση λογισμικού είναι ένας γενικός όρος το οποίο έχει δυο βασικούς άξονες την οπτικοποίηση προγραμμάτων (program visualization) και την οπτικοποίηση αλγορίθμων (algorithm visualization), Σχήμα 3.1.



Σχήμα 3.1: Venn διάγραμμα που παρουσιάζει τις σχέσεις μεταξύ των διαφόρων περιοχών της οπτικοποίηση λογισμικού

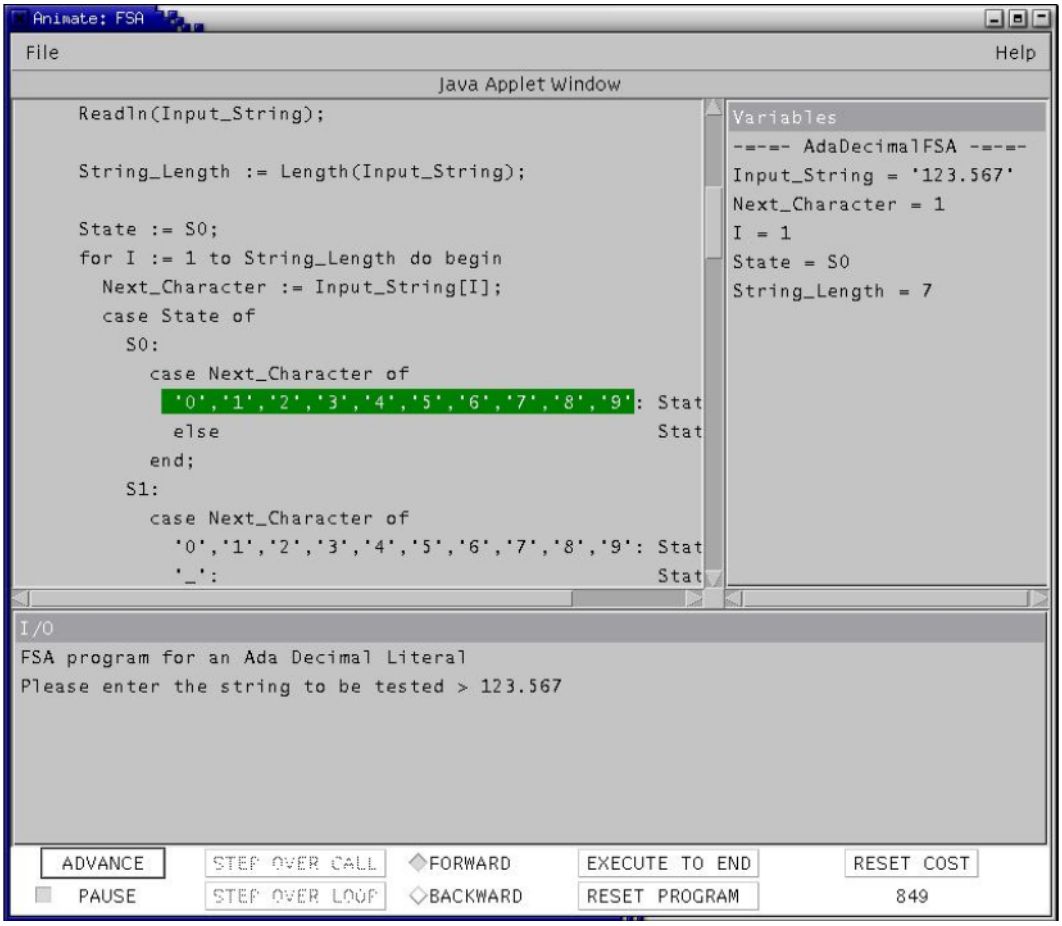
Μια εφαρμογή που έχει ως αντικείμενο τη *δυναμική παρουσίαση προγράμματος* (*program animator*) επιτρέπει την παρακολούθηση της εκτέλεσης του πηγαίου κώδικα ενός προγράμματος. Παρέχει μια άποψη της διαδικασίας εκτέλεσης του προγράμματος και εμφανίζει τις αλλαγές των μεταβλητών κατά τη διάρκεια της εκτέλεσης. Αυτή η διαδικασία μοιάζει με την *αναζήτηση λαθών στο πρόγραμμα* (*debugging*) που χρησιμοποιούνται για την ανάπτυξη λογισμικού, οι εφαρμογές δυναμικής παρουσίασης προγραμμάτων σχεδιάζονται ειδικά για εκπαιδευτικούς σκοπούς. Μερικά από τα μοναδικά χαρακτηριστικά γνωρίσματα αυτών των εφαρμογών περιλαμβάνει την δυνατότητα της *αντίστροφης εκτέλεσης* (*reverse execution*) σε οποιοδήποτε σημείο, την αυτόματη παρουσίαση των τιμών των μεταβλητών, και ειδικών τρόπων βηματικής εκτέλεσης ώστε να ενθαρρύνεται η πρόβλεψη της συμπεριφοράς του προγράμματος από τους φοιτητές. Όταν το

αντικείμενο της οπτικοποίησης είναι το λογισμικό, δίνεται έμφαση στο ρόλο ολοκλήρωσης της διαδικασίας προγραμματισμού, αρχίζοντας από τις προδιαγραφές των βασικών απαιτήσεων. Αν δίνεται έμφαση κατά τη διαδικασία οπτικοποίησης σε θέματα υψηλού επιπέδου, και δεν ασχολείται με τις λεπτομέρειες εκτέλεσης μιλάμε για οπτικοποίηση αλγορίθμου. Η οπτικοποίηση προγράμματος είναι η απεικόνιση του κώδικα ή των δομών δεδομένων ενός προγράμματος, είτε με στατική είτε με δυναμική μορφή.

Ένα παράδειγμα στατικής οπτικοποίησης δομής δεδομένων είναι η παρουσίαση των περιεχομένων μιας συνδεμένης λίστας ως διάγραμμα, ενώ στην περίπτωση της δυναμικής οπτικοποίησης το ίδιο διάγραμμα με τα περιεχόμενα και τα βέλη αλλάζει δυναμικά το περιεχόμενο της λίστας κατά την εκτέλεση του προγράμματος. Ένα παράδειγμα στατικής οπτικοποίησης κώδικα είναι ο συνδυασμός της τεκμηρίωσης των προγραμμάτων και του πηγαίου κώδικα.

3.1.1 DynaLab

Ο σχεδιασμός του συστήματος DynaLab (**Dyna**mic **Lab**oratory) ξεκίνησε με τη δημιουργία ενός ιδεατού υπολογιστή (virtual machine), που ονομάζεται E-Machine, το οποίο επιτρέπει και την αντίστροφη εκτέλεση των προγραμμάτων. Μόλις κατασκευάστηκε και ο εξομοιωτής του E-Machine, αναπτύχθηκαν και οι μεταγλωττιστές για τις γλώσσες προγραμματισμού C, Ada και Pascal. Σαν τελευταίο βήμα του συνολικού εγχειρήματος δημιουργήθηκε και η εφαρμογή δυναμικής παρουσίασης της εκτέλεσης για τις αντίστοιχες γλώσσες υψηλού επιπέδου, σε δυο διαφορετικές εκδόσεις η μια για περιβάλλον Unix και η άλλη για το περιβάλλον της Microsoft Windows. Μετά την εμφάνιση της Java, προκειμένου να μπορέσει να ξεφύγει από τον περιορισμό της εξάρτησης από λειτουργικά συστήματα (platform dependence), ο εξομοιωτής της E-Machine και το περιβάλλον δυναμικής παρουσίασης μεταφέρθηκαν από την αρχική υλοποίηση της γλώσσας C σε Java, Σχήμα 3.2.

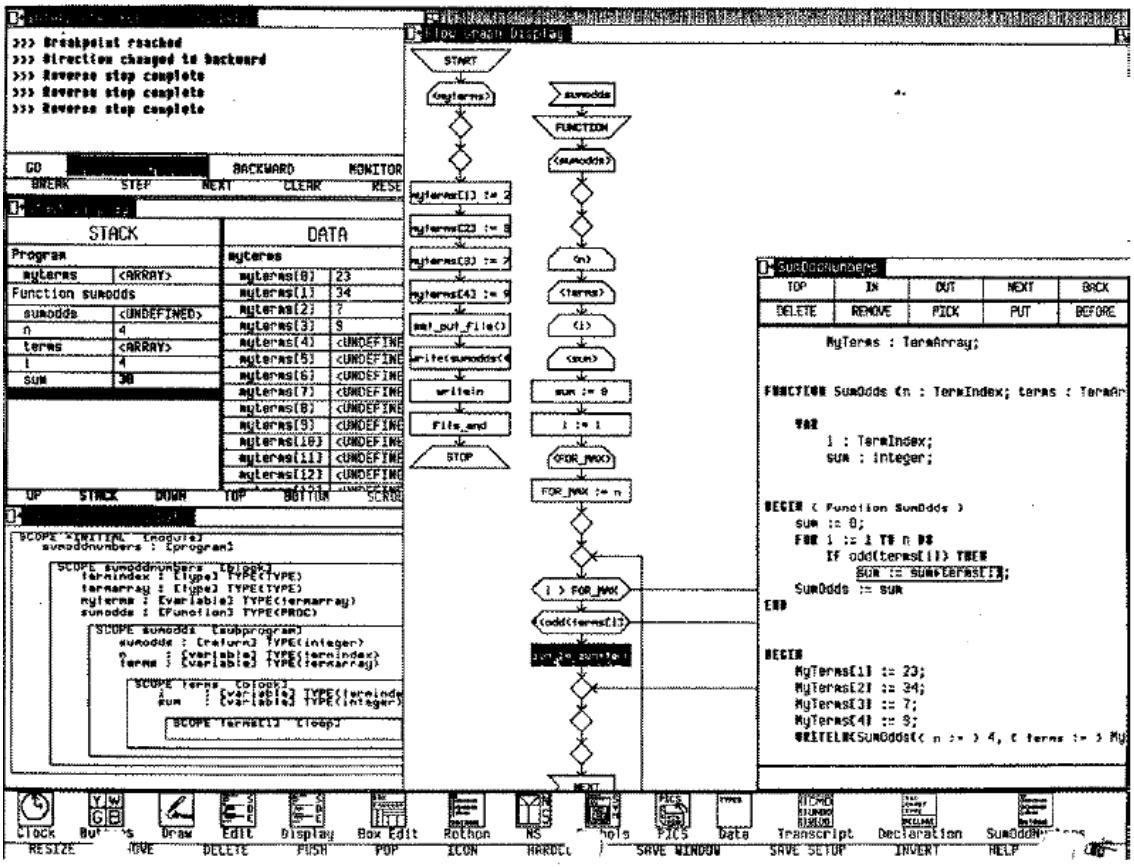


Σχήμα 3.2: Java έκδοση της εφαρμογής Dynalab

3.1.2 PECAN

Μια πρωτοποριακή προσπάθεια για την εξερεύνηση της δυνατότητας χρήσης των γραφικών των υπολογιστών (computer graphics) στην παρουσίαση της εκτέλεσης των προγραμμάτων ήταν το PECAN. Το PECAN περιλαμβάνει την προβολή πολλαπλών απόψεων του προγράμματος, Σχήμα 3.3. Όλες οι απόψεις προέρχονται από ένα κοινό δέντρο σύνταξης (syntax tree). Στις απόψεις του προγράμματος (program views) περιλαμβάνεται ένας συντάκτης κατευθυνόμενο από τη σύνταξη (syntax directed editor), πάνω δεξιά στο σχήμα και ένα Nassi-Shneiderman διάγραμμα που δημιουργείται αυτόματα, πάνω αριστερά στο σχήμα. Το Nassi-Shneiderman διάγραμμα είναι μια μορφή δομημένου διαγράμματος ροής (structured flow chart).

Η προβολή των δεδομένων που δημιουργήθηκαν από το χρήστη παρουσιάζονται στη μέση αριστερά στο σχήμα, και τέλος το διάγραμμα ροής που παρουσιάζει τον έλεγχο του προγράμματος, εμφανίζεται στη μέση του σχήματος.



Σχήμα 3.3: Στιγμιότυπο του συστήματος PECAN

3.1.3 FIELD

Το FIELD (Friendly Integrated Environment for Learning and Development) είναι ένα περιβάλλον προγραμματισμού που σχεδιάστηκε για εκπαιδευτική χρήση. Οι εκπαιδευόμενοι μπορούν να γράψουν προγράμματα σε γλώσσες όπως Pascal, Object-Oriented Pascal, C, και C++.

Το FIELD ενσωματώνει διάφορα εργαλεία προγραμματισμού (programming tools) που χρησιμοποιούν τον μηχανισμό βασισμένο σε μηνύματα (message-based mechanism). Αυτός ο μηχανισμός χρησιμοποιεί ένα κεντρικό εξυπηρετητή μηνυμάτων (message server) με τον οποίο επικοινωνούν τα εργαλεία. Όταν ένα εργαλείο τεθεί σε λειτουργία καταχωρεί στον εξυπηρετητή μηνυμάτων ένα υπόδειγμα το οποίο περιγράφει όλα τα μηνύματα που τον ενδιαφέρουν. Καθώς εκτελείται η

εφαρμογή, μηνύματα στέλνονται στο εξυπηρετητή μηνυμάτων τα οποία ανακατανέμονται σύμφωνα με τα υποδείγματα σε όλα τα εργαλεία που εκδήλωσαν ενδιαφέρον. Τα μηνύματα αποτελούνται από δύο βασικούς τύπους: την εντολή (command), η οποία ζητά κάποια λειτουργία από ένα άλλο εργαλείο, και την πληροφορία (information) που παρέχει δεδομένα από το ένα εργαλείο σε κάποιο άλλο που ενδεχομένως να το ενδιαφέρει.

Εκτός από την δυνατότητα παρακολούθησης της εκτέλεσης βήμα προς βήμα του προγράμματος, το FIELD παρέχει και άλλα εργαλεία οπτικοποίησης:

- Οπτικοποίηση δομών δεδομένων. Επιτρέπει την παρακολούθηση και αλλαγή των δομών δεδομένων του προγράμματος.
- Γράφημα κλήσεων. Παρουσιάζει την δομή των κλήσεων του προγράμματος και τονίζει τους κόμβους όταν είναι ενεργοί κατά τη διάρκεια της εκτέλεσης.
- Επίσης παρέχει διάφορα εργαλεία που μπορούν να παρουσιάσουν την κατανομή της μνήμης και τις προσβάσεις που γίνονται στα αρχεία.

3.1.4 ZStep 95

Η ZStep 95 είναι μια εφαρμογή δυναμικής παρουσίασης προγραμμάτων για ένα υποσύνολο της γλώσσας προγραμματισμού Lisp (Common Lisp). Επιτρέπει στους εκπαιδευόμενους την συγγραφή και βηματική εκτέλεση των προγραμμάτων, στην κανονική (προς τα εμπρός) αλλά και στην αντίστροφη εκτέλεση.

Η ZStep 95 έχει μερικά καινοτόμα χαρακτηριστικά για την απεικόνιση των εκφράσεων (expressions) και τις τιμές των μεταβλητών (variable values) ενός προγράμματος που εκτελείται. Παρά την προσπάθεια που πρέπει να καταβάλλει κάποιος να μοιράσει την προσοχή του ανάμεσα στην παρακολούθηση του πηγαίου κώδικα και του παραθύρου των τιμών των δεδομένων. Το περιεχόμενο του παράθυρο των τιμών συμβαδίζει με την εκτέλεση των εκφράσεων, παρέχει εκτός από την παρουσίαση της τρέχουσας αξίας των εκφράσεων αλλά και το κατάλογο των τιμών που έχει πάρει η έκφραση κατά τη διάρκεια της εκτέλεσης του προγράμματος.

3.1.5 LEONARDO

Το Leonardo παρέχει παρόμοιες λειτουργίες με το Dynalab. Επιτρέπει την συγγραφή και μεταγλώττιση προγραμμάτων στη γλώσσα προγραμματισμού C (ANSI C), με τη χρήση τυποποιημένων βιβλιοθηκών της C (standard C libraries). Τα προγράμματα εκτελούνται σε μια ειδική εικονική μηχανή (virtual machine), έτσι επιτρέπεται η αντίστροφη εκτέλεση σε οποιοδήποτε σημείο. Το LEONARDO δεν παρουσιάζει κατά τη διάρκεια της εκτέλεσης τις τιμές των μεταβλητών, αλλά περιλαμβάνει ένα γραφικό σύστημα το οποίο επιτρέπει την ενσωμάτωση των γραφικών παρουσιάσεων πάνω στα προγράμματα της γλώσσας C. Οι ειδικές εντολές μπορούν να τοποθετηθούν στα σχόλια του προγράμματος που κατευθύνουν το γραφικό σύστημα, για να λάβουν τις απαραίτητες ενέργειες για την παρουσίαση του προγράμματος που εκτελείται. Ένας προεπεξεργαστής (preprocessor), ενσωματώνει αυτές τις εντολές στον κώδικα του προγράμματος πριν από τη μεταγλώττιση του. Η βασική έκδοση του LEONARDO δημιουργήθηκε για υπολογιστές Apple Macintosh, Σχήμα 3.4.

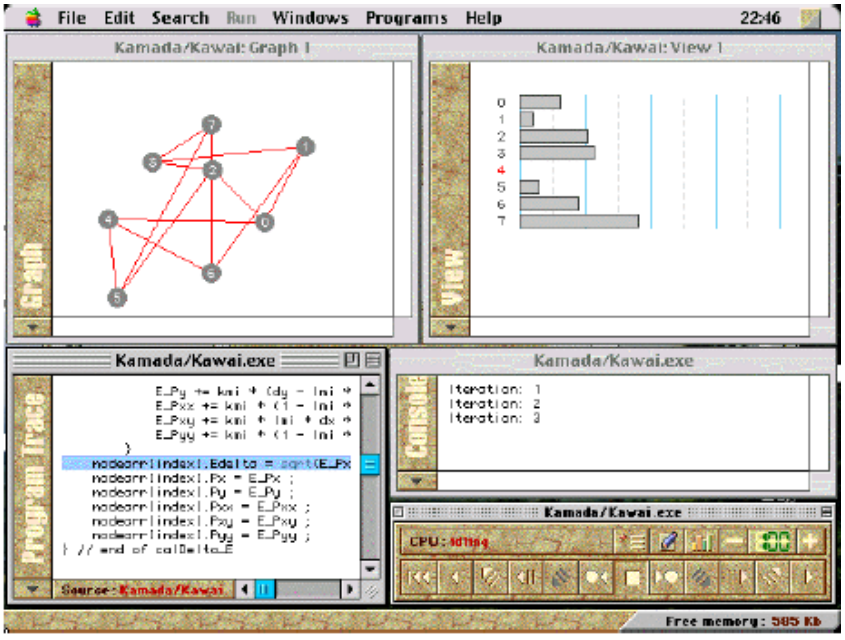
Το LEONARDO αρχικά προοριζόταν για την οπτικοποίηση αλγορίθμων γράφων (graph algorithms), αλλά αργότερα οι γραφικές δυνατότητες επεκτάθηκαν, με την προσθήκη και άλλων γραφικών αντικειμένων προκειμένου να χρησιμοποιηθεί και για γενικής χρήσης αλγορίθμους.

Τα κύρια χαρακτηριστικά γνωρίσματα του Leonardo είναι τα ακόλουθα:

- Οι δυναμικές παρουσιάσεις δημιουργούνται ως αντικατοπτρισμός των βασικών ιδιοτήτων των αλγορίθμων, το οποίο επιτυγχάνεται με την βοήθεια μιας λογικής γλώσσας οπτικοποίησης (logic visualization language), που ονομάζεται ALPHA και επιτρέπει την δήλωση της αντιστοίχισης μεταξύ των συγκεκριμένων και αφηρημένων δομών δεδομένων.
- Οποιαδήποτε στιγμή κατά τη διάρκεια της εκτέλεσης ενός προγράμματος το επόμενο βήμα μπορεί να οδηγήσει είτε στην επόμενη είτε στην προηγούμενη κατάσταση. Η δηλωτική προσέγγιση βασιζόμενη στην ALPHA και η πλήρης αντιστεψιμότητα (reversibility) του εικονικού υπολογιστή (virtual CPU), καθιστούν πιθανή την αντιστροφή της κατεύθυνσης της δυναμικής παρουσίασης χωρίς πρόσθετο κόστος, δηλ. χωρίς την ανάγκη αποθήκευσης πρόσθετων πληροφοριών.

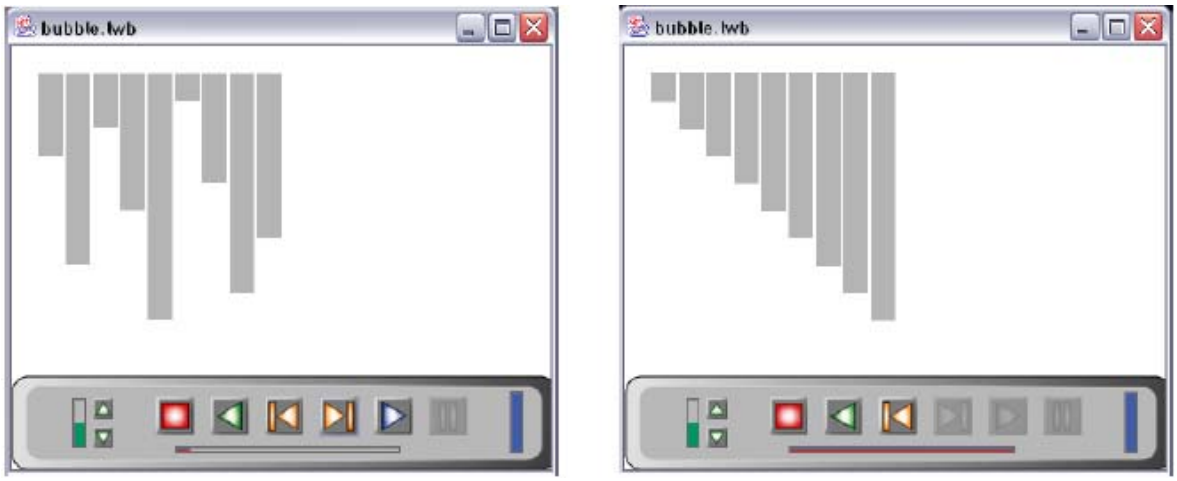
- Το σύστημα είναι σχεδιασμένο για να μην αποθαρρύνει κανέναν που να ενδιαφέρεται για την οπτικοποίηση του προγράμματός του. Διότι:

  - ✓ Τα οπτικά γεγονότα παράγονται αυτόματα, έτσι δεν είναι απαραίτητη η καλή γνώση του κώδικα του προγράμματος, έτσι ο φοιτητής απαλλάσσεται από το καθήκον εντοπισμού και ορισμού των ενδιαφερόντων γεγονότων.
  - ✓ Η αυξητική προσέγγιση (incremental approach), μπορεί να χρησιμοποιηθεί κατά τον ορισμό της δυναμικής παρουσίασης. Μια ελάχιστη ποσότητα πληροφοριών είναι απαραίτητη για την παραγωγή της βασικής απεικόνισης. Κατόπιν, είναι εύκολο να βελτιωθεί η παρουσίαση, προσθέτοντας, αφαιρώντας ή τροποποιώντας τους κανόνες οπτικοποίησης.
  - ✓ Ο πηγαίος κώδικας ενός προγράμματος μπορεί να χρησιμοποιηθεί ξανά, η δυναμική παρουσίαση των προγραμμάτων στο LEONARDO μπορεί να μεταγλωττιστεί και σε άλλο περιβάλλον που αγνοούν τους κανόνες οπτικοποίησης τα οποία υπάρχουν στο πρόγραμμα ως ειδικά σχόλια.
  - ✓ Το LEONARDO περιλαμβάνει ένα συντάκτη γραφικών (graph editor), που επιτρέπει τη δοκιμή των προγραμμάτων σε ένα φιλικό προς το χρήστη (user friendly) περιβάλλον.



Σχήμα 3.4: Το σύστημα δυναμικής παρουσίασης LEONARDO

Η έκδοση για το Internet της εφαρμογής ονομάζεται Leonardo Web, Σχήμα 3.5, είναι μια συλλογή εργαλείων βασισμένα στο Web (Web-based tools) για την δημιουργία και διασπορά των δυναμικών παρουσιάσεων για εκπαιδευτικούς σκοπούς για συστήματα τηλεκπαίδευσης (e-learning). Οι δυναμικές παρουσιάσεις μπορούν να δημιουργηθούν με τη χρήση ενός εξειδικευμένου οπτικού συντάκτη, και μπορούν να παρουσιαστούν με ένα Java ερμηνευτή (Java player), είτε ως αυτόνομη εφαρμογή (stand-alone application) είτε ως Java applet. Και οι δύο τρόποι παρέχουν τις ίδιες δυνατότητες επιλογής της εκτέλεσης της δυναμικής παρουσίασης: βήμα-βήμα ή συνεχόμενη. Για την υποστήριξη της απεικόνισης των εννοιών των αλγορίθμων, παρέχει μια βιβλιοθήκη που επιτρέπει την δημιουργία δυναμικών παρουσιάσεων απ' ευθείας από την Java υλοποίηση του αλγορίθμου σύμφωνα με ένα επιτακτικό κανόνα. Οι παρουσιάσεις μπορούν να περιλαμβάνουν κείμενο, γραφικά δυο διαστάσεων, εικόνες bitmap με ομαλές μεταβάσεις μεταξύ καταστάσεων.



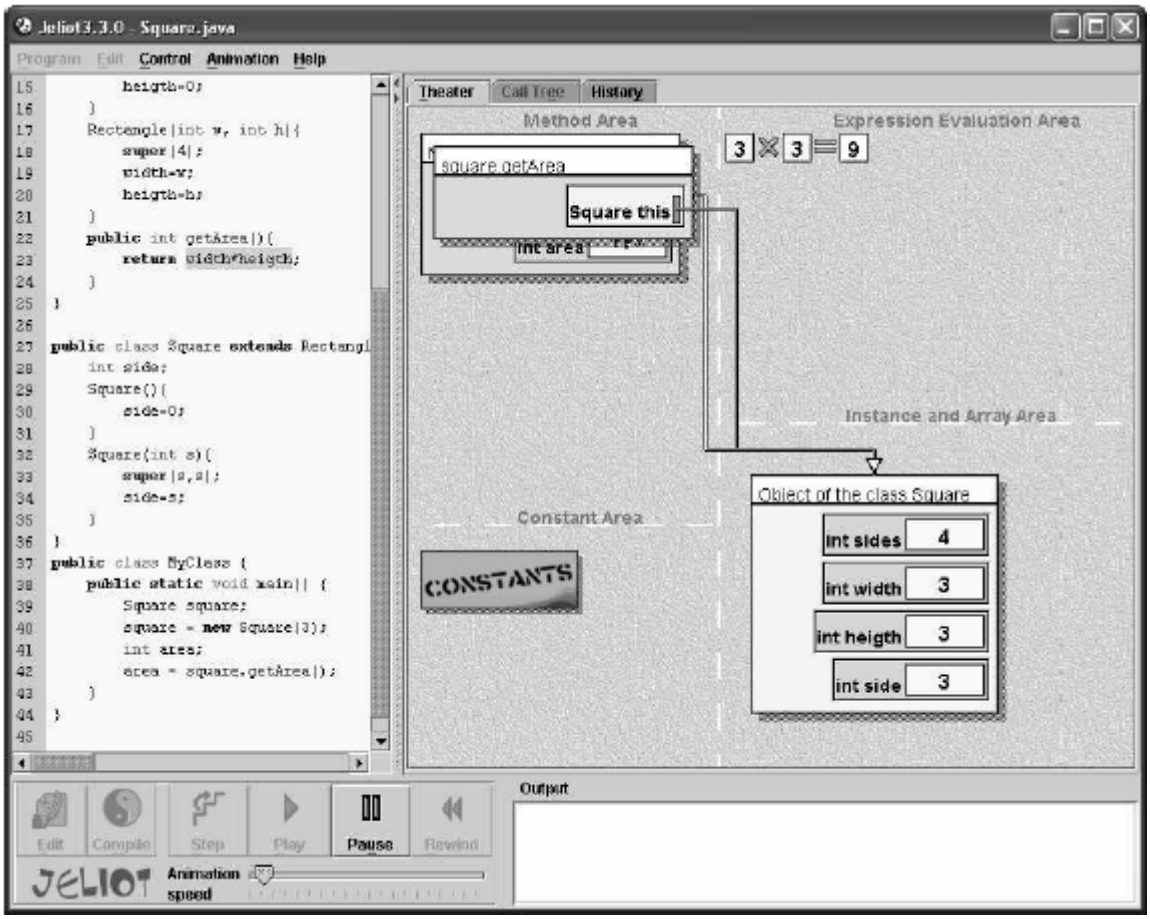
Σχήμα 3.5: Στιγμιότυπο από την δυναμική παρουσίαση του αλγορίθμου ταξινόμησης με τη μέθοδο της φυσαλίδας

Το σύστημα εκτέλεσης παρουσιάσεων του Leonardo Web, σχεδιάστηκε να είναι εύχρηστο και απλό. Το γραφικό περιβάλλον αλληλεπίδρασης του Player μοιάζει με ένα τυποποιημένο σύστημα ελέγχου βίντεο (VCR), Σχήμα 3.6. Τα κουμπιά ελέγχου επιτρέπουν την εκκίνηση, διακοπή, γρήγορη εκτέλεση (rewind), εκτέλεση της παρουσίασης εμπρός και πίσω.

Σχήμα 3.6: Το γραφικό σύστημα αλληλεπίδρασης του Leonardo Web

3.1.6 JELIOT

Η οικογένεια εφαρμογών Jeliot είναι μια συλλογή εργαλείων για την οπτικοποίηση προγραμμάτων και αλγορίθμων με στόχο την διδασκαλία προγραμματισμού, αλγορίθμων και δομών δεδομένων σε αρχάριους φοιτητές. Το τελευταίο μέλος της οικογενείας είναι το Jeliot 3. Είναι μια εφαρμογή οπτικοποίησης προγραμμάτων το οποίο βασίζεται στην αυτόματη δυναμική παρουσίαση των προγραμμάτων Java. Η έκδοση Jeliot 3 εξελίχτηκε από μια προηγούμενη έκδοση που ονομάζεται Jeliot 2000, αλλά συνεχίζει να έχει το ίδιο περιβάλλον αλληλεπίδρασης και στυλ οπτικοποίησης, Σχήμα 3.7. Η Jeliot 3 οπτικοποιεί τον τρόπο εκτέλεσης ενός προγράμματος Java παρουσιάζοντας την τρέχουσα κατάσταση του προγράμματος (π.χ. μεθόδους, μεταβλητές, και αντικείμενα), και την δυναμική παρουσίαση του υπολογισμού των εκφράσεων (expression evaluations) και βρόγχων (loops).



Σχήμα 3.7: Το περιβάλλον αλληλεπίδρασης του Jeliot 3

Κεφάλαιο 4

Jeliot

Το Jeliot είναι μία εφαρμογή δυναμικής αναπαράστασης της εκτέλεσης προγραμμάτων και μπορεί να χρησιμοποιηθεί για την εκμάθηση εισαγωγικών εννοιών προγραμματισμού. Η εκτέλεση του προγράμματος παρουσιάζεται αυτόματα δίχως πρόσθετων τροποποιήσεων ή σχολίων εκ μέρους του χρήστη. Ενώ κάτι τέτοιο περιορίζει την ευελιξία της διαδικασίας δυναμικής παρουσίασης της εκτέλεση του προγράμματος, αλλά συγχρόνως για τον ίδιο λόγο δηλ. δεν απαιτείται παρέμβαση του χρήστη για την δυναμική παρουσίαση το Jeliot είναι εξαιρετικά απλό έτσι ώστε να γίνεται εύκολα αποδεκτό από αρχαρίους προγραμματιστές, όπως επίσης και από τους καθηγητές τους οι οποίοι δεν χρειάζεται να σπαταλήσουν επιπλέον χρόνο για να μάθουν πώς να προετοιμάζουν την δυναμική αναπαράσταση της εκτέλεσης ενός προγράμματος.

Το Jeliot έχει γραφεί με τη χρήση της γλώσσας προγραμματισμού Java προκειμένου να αποκτήσει τη δυνατότητα εκτέλεσης σε ανεξάρτητο από λειτουργικό σύστημα περιβάλλον. Αναπαριστά δυναμικά προγράμματα που έχουν γραφτεί σε γλώσσα προγραμματισμού Java, αλλά κάτι τέτοιο δεν περιορίζει τη χρήση του αποκλειστικά σε μια σειρά μαθημάτων σχετικών με τη γλώσσα προγραμματισμού Java, αλλά μπορεί να χρησιμοποιηθεί και για την εκμάθηση και άλλων γλωσσών προγραμματισμού, όπως επί παραδείγματι η γλώσσα προγραμματισμού PASCAL.

4.1 Εγκατάσταση του Jeliot

Προτού ξεκινήσει οποιαδήποτε εγκατάσταση του Jeliot, θα πρέπει να ελεγχθεί εάν υπάρχει διαθέσιμο οποιοδήποτε από τα περιβάλλοντα εκτέλεση προγραμμάτων Java, όπως το Java Runtime Environment (JRE) της Sun και να καλύπτει τουλάχιστον την έκδοση 1.4 ή νεότερης έκδοσης. Προκειμένου να εξακριβωθεί εάν υπάρχει ήδη διαθέσιμο κάτι τέτοιο ή όχι, πρέπει να χρησιμοποιηθεί η εντολή: java.exe ή java. Εάν υπάρχει διαθέσιμη κάποια έκδοση της JRE, στη συνέχεια θα πρέπει να διαπιστωθεί και η έκδοση της, το οποίο μπορεί να επιτευχθεί με την χρήση της εντολής java –version στην κατάσταση εκτέλεσης εντολών. Στην περίπτωση που δεν υπάρχει ήδη

εγκαταστημένη έκδοση της JRE, θα πρέπει να ελεγχθούν οι διαθέσιμες εκδόσεις στη διεύθυνση: http:// java.sun.com, όπως και κάθε άλλη πληροφορία σχετικά με το πώς μπορεί να εγκατασταθεί η τελευταία έκδοση της Java.

Παρακάτω παρουσιάζονται οι οδηγίες εγκατάστασης του Jeliot το οποίο περιλαμβάνει δύο διαφορετικούς τρόπους χρήσης και εγκατάστασης:

Το πρώτο σχετίζεται με την χρήση του Windows Installer της Jeliot, ενώ το δεύτερο με τη χρήση της εκτελέσιμης διανομής του Jeliot.

Στους χρήστες των WINDOWS συστήνεται να επιλέξουν την έκδοση που χρησιμοποιεί το Windows Installer. Εάν όμως αυτό που επιθυμούμε είναι η χρήση της εφαρμογής του Jeliot, και όχι ο τρόπος που σχεδιάστηκε και δημιουργήθηκε το Jeliot οποία έκδοση και να επιλεγεί είναι κατάλληλη για την εκμάθηση βασικών εννοιών προγραμματισμού. Ωστόσο, για τους ενδιαφερομένους, επίσης διατίθεται και η Δυαδική Διανομή (Binary distribution).

4.1.1 Εγκατάσταση του Jeliot σε περιβάλλον Windows

Αυτός ο τρόπος εγκατάστασης χρησιμοποιεί το Windows Installer του Jeliot περιγράφουν. Το οποίο είναι ο καλύτερος τρόπος για τους χρήστες των WINDOWS καθώς είναι και ο ευκολότερος τρόπος να γίνει η εγκατάσταση. Ο τρόπος αυτός της εγκατάστασης δημιουργεί συντομεύσεις στο μενού Έναρξης όπως και στην Επιφάνεια Εργασίας για την ευκολότερη διαχείριση του Jeliot.

4.1.2 Μεταφορά των αρχείων του Jeliot σε ένα υπολογιστή

Η εφαρμογή του Jeliot είναι διαθέσιμη στην ιστοσελίδα του πανεπιστημίου Joensuu στη Φινλανδία **http://www.cs.joensuu.fi/jeliot/downloads.php** οι σελίδες που παρουσιάζονται καθοδηγούν το χρήστη ώστε να επιλέξει είτε την εκτέλεση της εφαρμογής Jeliot 3 Windows Installer ώστε να ξεκινήσει άμεσα η εγκατάσταση , είτε την δυνατότητα αποθήκευσης του αρχείου εγκατάστασης στον υπολογιστή του χρήστη. Ωστόσο, θα είναι προτιμότερο η αποθήκευση του αρχείου εγκατάστασης σε ένα τοπικό αποθηκευτικό μέσο σε περίπτωση που η πρώτη εγκατάσταση αποτύχει.

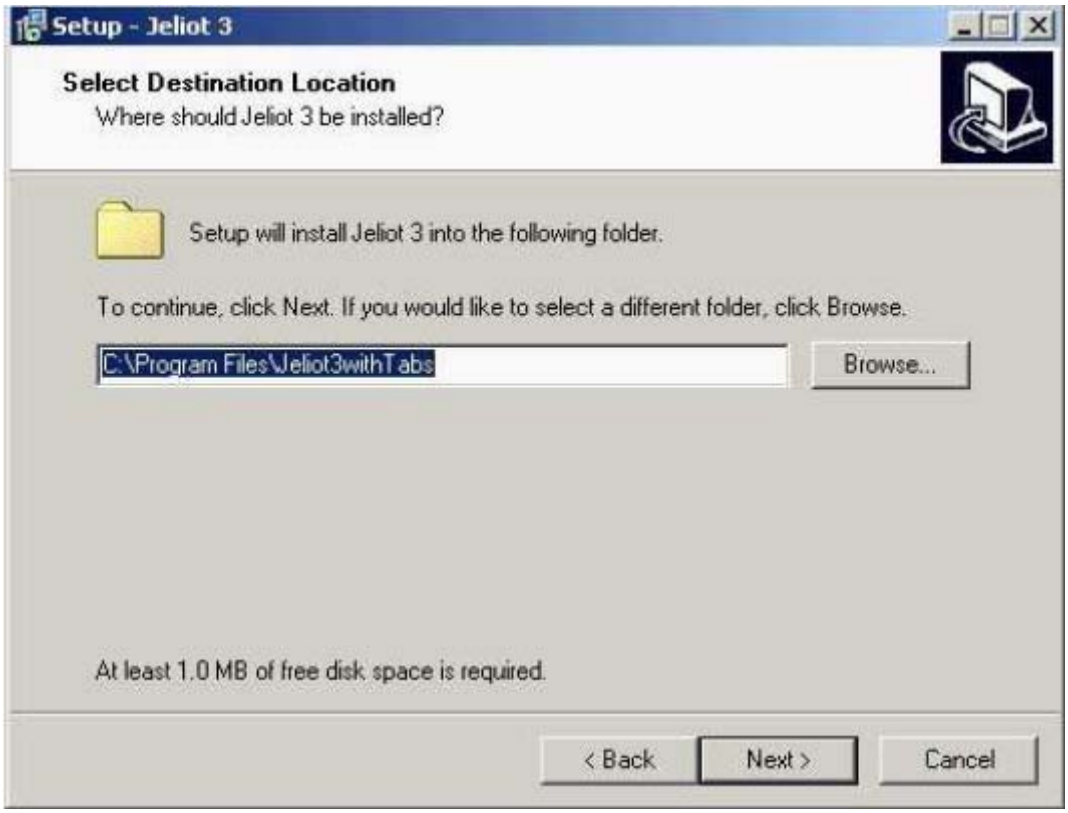
Το αρχείο που θα μεταφερθεί στον υπολογιστή του χρήστη έχει την ακόλουθη μορφή: Jeliot 3–N.exe όπου N είναι ο αριθμός της έκδοσης, για παράδειγμα, η έκδοση 2 preview 3, θα είναι το αρχείο : Jeliot3–2P3.exe.

4.1.3 Εγκατάσταση (INSTALLATION)

Κατά τη διάρκεια εκτέλεσης της εφαρμογής Jeliot3–N.exe, ένα παράθυρο υποδοχής θα εμφανιστεί όπως αυτό παρουσιάζεται στο Σχήμα 4.1.

Σχήμα 4.1: Διαδικασία της εγκατάστασης του Jeliot

Επιλέγοντας το κουμπί NEXT, η εκτέλεση της εφαρμογής θα μεταφερθεί στο επόμενο στάδιο όπως αυτό φαίνεται στο Σχήμα 4.2.



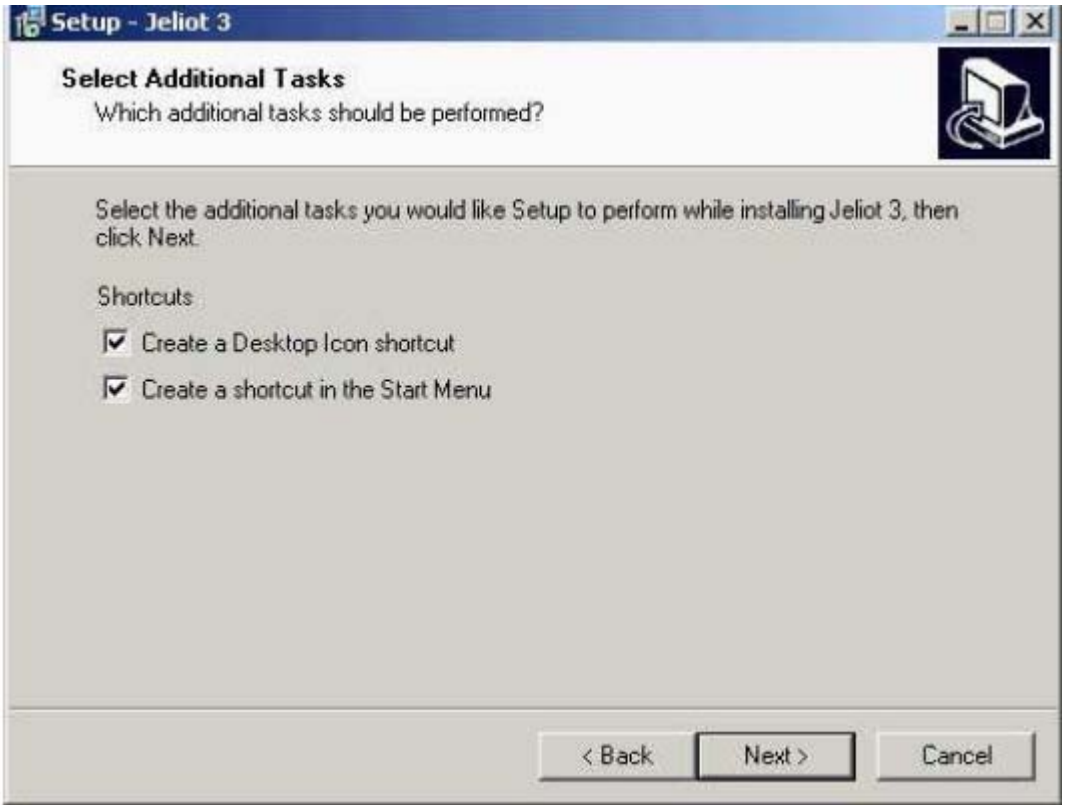
Σχήμα 4.2: Διαδικασία της εγκατάστασης του Jeliot

Στο παράθυρο αυτό δίνεται η δυνατότητα επιλογής του φακέλου εγκατάστασης της εφαρμογής του Jeliot. Η εγκατάσταση του Jeliot απαιτεί τουλάχιστον 1 MB διαθέσιμο χώρο στο αποθηκευτικό μέσο που επιλέχτηκε για εγκατάσταση. Για να συνεχιστεί η διαδικασία της εγκατάστασης επιλέγουμε και πάλι το κουμπί NEXT.

Το επόμενο βήμα της εγκατάσταση περιμένει να οριστεί το όνομα του φακέλου στο οποίο θα εγκατασταθεί το Jeliot, όπως αυτό παρουσιάζεται στο σχήμα 4.3, μετά τον ορισμό του φακέλου επιλέξτε το κουμπί NEXT.

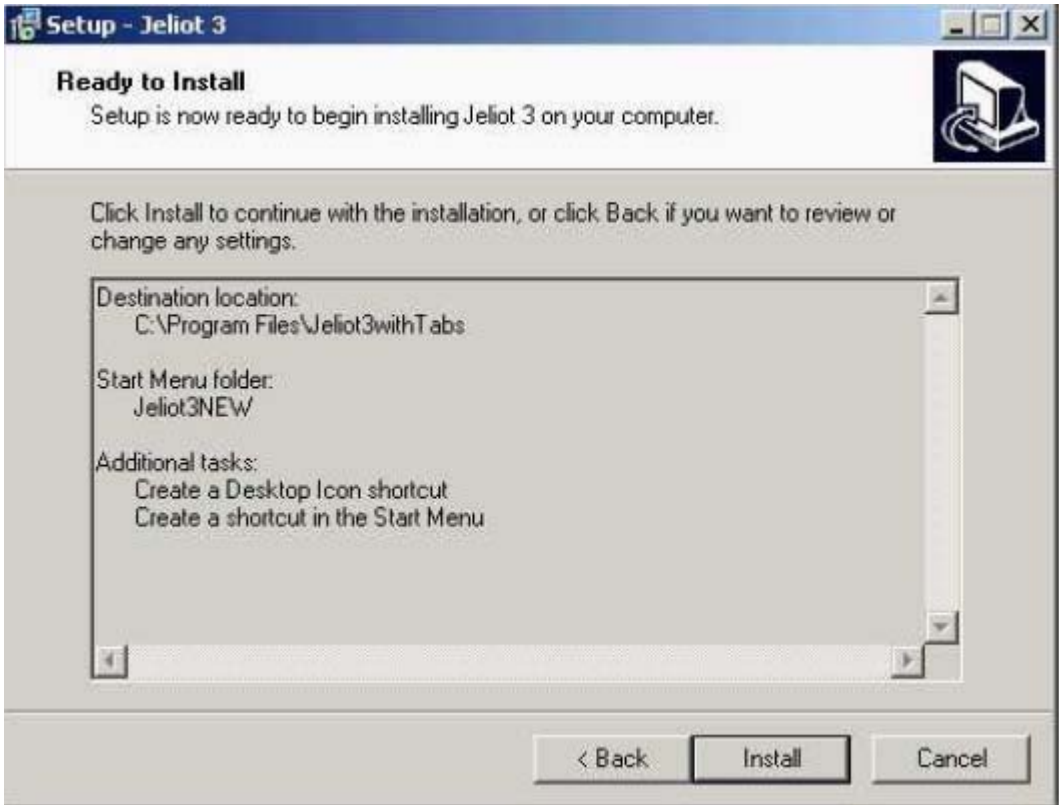
Σχήμα 4.3: Διαδικασία της εγκατάστασης του Jeliot

Το επόμενο βήμα της εγκατάστασης περιλαμβάνει την δυνατότητα εάν επιθυμείται η δημιουργία συντόμευσης (shortcuts) (Σχήμα 4.4). Επιλέξτε τουλάχιστον το Μενού Εκκίνησης και επιλέξτε το κουμπί NEXT.



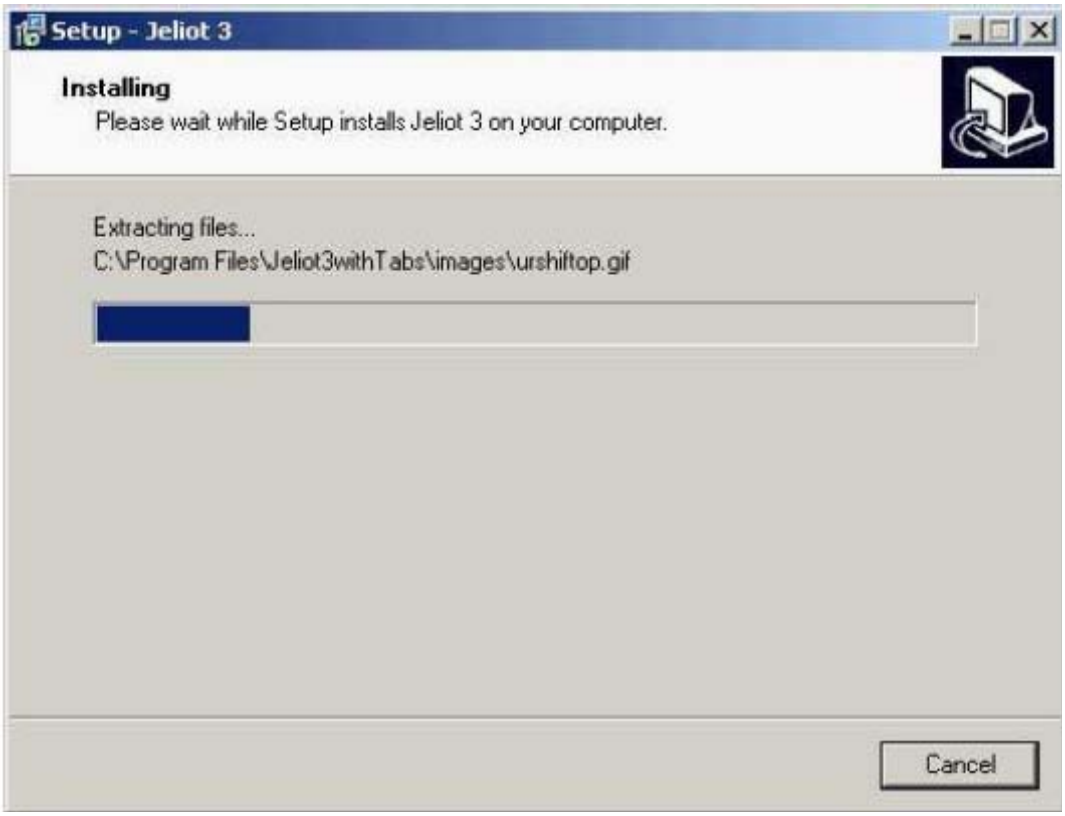
Σχήμα 4.4: Διαδικασία της εγκατάστασης του Jeliot

Το παράθυρο του σχήματος 4.5 ανακεφαλαιώνει τις επιλογές που έγιναν, επιλέγοντας το κουμπί της ΕΓΚΑΤΑΣΤΑΣΗΣ (Σχήμα 4.5), θα ξεκινήσει η εγκατάσταση.



Σχήμα 4.5: Διαδικασία της εγκατάστασης του Jeliot

Η διαδικασία της εγκατάστασης παρουσιάζεται στο σχήμα 4.6.



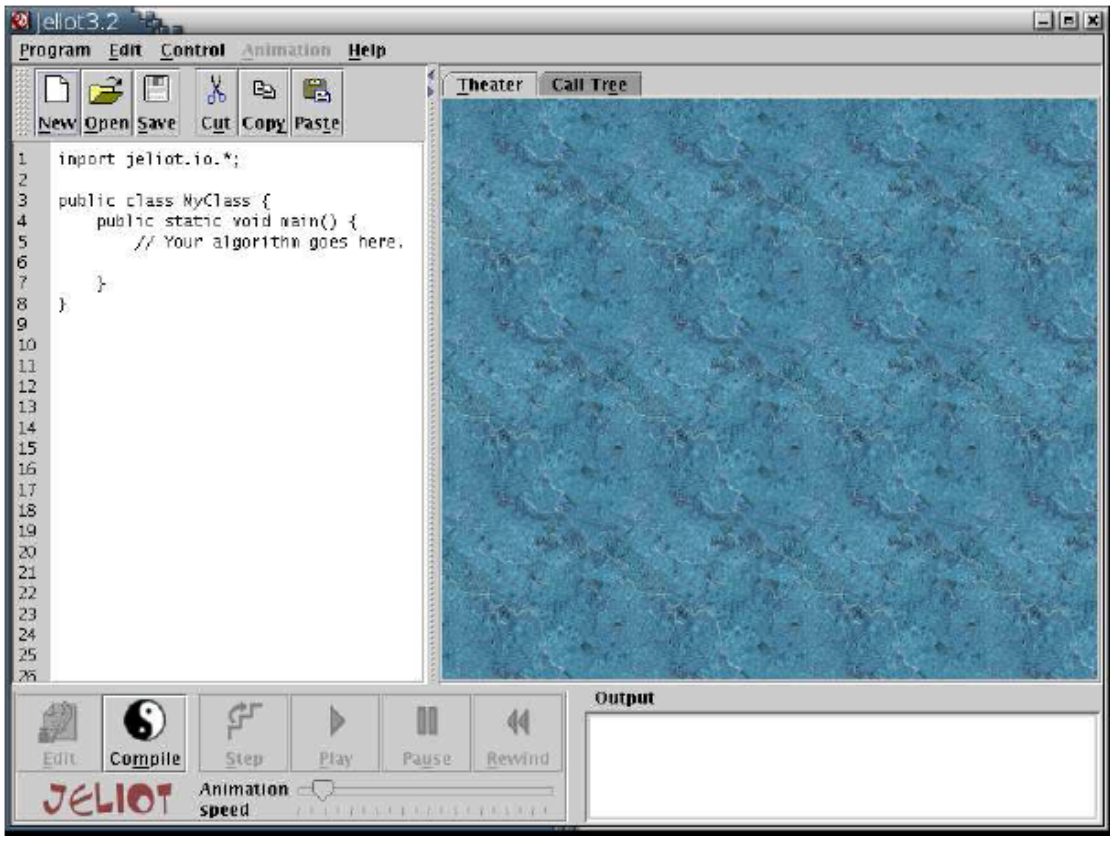
Σχήμα 4.6: Διαδικασία της εγκατάστασης του Jeliot

Ενώ στο τελικό παράθυρο ενημερώνεται ο χρήστης ότι η διαδικασία εγκατάστασης έχει ολοκληρωθεί με επιτυχία. Επιλέγοντας το κουμπί FINISH λήγει η εγκατάσταση. Από εδώ και στο εξής το Jeliot έχει εγκατασταθεί και είναι έτοιμο για χρήση.

4.1.4 Εκτέλεση του Jeliot

Εάν η εγκατάσταση μέσω του Windows Installer επιτεύχθηκε σωστά και εάν βέβαια έχετε κάνει τις επιλογές σας για την δημιουργία τρόπων συντόμευσης του μενού εκκίνησης (start-menu shortcuts), θα έπρεπε να έχετε το Jeliot διαθέσιμο στο μενού εκκίνησης των WINDOWS υπό μορφή φακέλου (folder) με το όνομα που εισάγατε κατά την διάρκεια της εγκατάστασης. Ανεξάρτητα με τις επιλογές σας, καλύτερα να έχετε και μια συντόμευση στην επιφάνεια εργασίας. Εάν ο installer απέτυχε να εισάγει τους τρόπους συντόμευσης, μπορείτε να κάνετε εκκίνηση στο Jeliot κάνοντας διπλό κλίκ επάνω στο Jeliot.bat μέσα στον κατάλογο(directory) στον οποίο εγκαταστήσατε το Jeliot.

Το βασικό παράθυρο του Jeliot θα πρέπει να έχει την παρακάτω μορφή, μετά την επιτυχή εγκατάσταση του προγράμματος (Σχήμα 4.7).



Σχήμα 4.7: Το εφαρμογή μετά την επιτυχή εγκατάσταση του προγράμματος

4.2 Εγκατάσταση της εκτελέσιμης έκδοσης (Executable Distribution)

Οι παρακάτω περιγραφή των βημάτων μπορεί να βοηθήσει στην εγκατάσταση της εκτελέσιμης έκδοσης του Jeliot σε περιβάλλοντα λειτουργικών συστημάτων όπως τα Windows, Linux, *nix η οποιουδήποτε άλλου λειτουργικού συστήματος, στο οποίο έχει εγκατασταθεί το περιβάλλον της Java.

4.2.1 Μεταφορά αρχείων (Downloading)

Η εφαρμογή του Jeliot είναι διαθέσιμη στην ιστοσελίδα του πανεπιστημίου Joensuu στη Φινλανδία **http://www.cs.joensuu.fi/jeliot/downloads.php** και επιλέξτε την εκτελέσιμη έκδοση Jeliot 3 executable distribution το οποίο είναι σε συμπιεσμένη μορφή (zipped file).

Το αρχείο που θα μεταφερθεί στον υπολογιστή του χρήστη έχει την ακόλουθη μορφή: Jeliot-3.zip όπου N είναι ο αριθμός της έκδοσης, για παράδειγμα, η έκδοση 2 preview 3, θα είναι το αρχείο : Jeliot3–2P3. zip.

4.2.2 Αποσυμπίεση (uncompression)

Η αποσυμπίεση του φακέλου που μόλις αποκτήσατε από την ιστοσελίδα θα έχει ως αποτέλεσμα την δημιουργία νέων φακέλων. Καταρχήν, θα πρέπει να δημιουργήσετε έναν κατάλογο για το Jeliot, για παράδειγμα c:/jeliot/ στο Windows ή /home/user/jeliot/ στο Linux/*nix, ο οποίος θα εξαρτάται από το λειτουργικό σας σύστημα και την πρόσβαση σε αυτό. Οποιοσδήποτε φάκελος ή κατάλογος είναι δεκτό, εφόσον γνωρίζετε πού βρίσκεται. Ακόμα και το πρόγραμμα αποσυμπίεσης μπορεί να δημιουργήσει έναν κατάλογο για τους φακέλους.

Επόμενο βήμα: αποσυμπιέστε το Jeliot3-N.zip στο φάκελο που δημιουργήσατε χρησιμοποιώντας

1. WinZip, ή κάποιο παραπλήσιο εργαλείο των Windows (διπλοπατήστε στον φάκελο Jeliot3-N.zip στον ίδιο κατάλογο και επιλέξτε αποσυμπιεση ‘extract’ από το πρόγραμμα. Επιλέξτε τον κατάλογο που μόλις δημιουργήσατε ως στόχο.
2. Δώστε την εντολή:unzip Jeliot 3-N.zip στο περιβάλλον Linux/*nix. Προσέξτε ότι η συγκεκριμένη εντολή θα αποσυμπιέσει τους φακέλους του καταλόγου μέσα στον οποίο βρίσκεστε και θα χρειαστεί τον φάκελο:Jeliot3-N.zip του ιδίου καταλόγου. Συγκεκριμενοποιήστε το πλήρες μονοπάτι (path) για να αποσυμπιέσετε το πακέτο από οπουδήποτε και χρησιμοποιήστε την επιλογή –d για να συγκεκριμενοποιήσετε τον κατάλογο-στόχο σας. Τώρα θα πρέπει να έχετε το Jeliot αποσυμπιεσμένο στον δίσκο. Για να το εξακριβώσετε, μπορείτε να ελέγξετε εάν οι φάκελοι: Jeliot.jar, jeliot.bat, jeliot.ico, license.txt και οι υποκατάλογοι: images, examples και docs όντως υπάρχουν στον κατάλογο τον οποίο δημιουργήσατε.

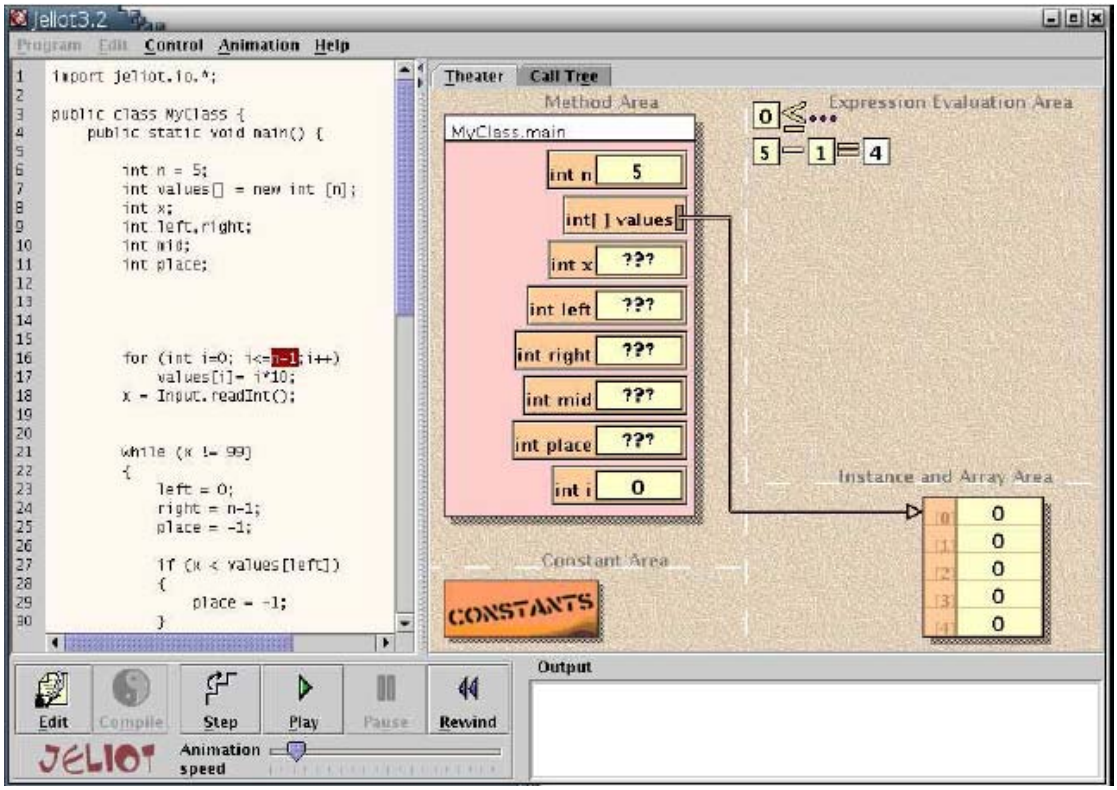
4.2.3 Εκτέλεση του Jeliot (running)

Προκειμένου να χρησιμοποιήσετε το Jeliot:

1. Στα Windows μπορείτε να χρησιμοποιήσετε: jeliot.bat. διπλοπατήστε στο εικονίδιο του Jeliot. Ίσως θα ήταν μια καλή ιδέα να δημιουργήσετε έναν τρόπο συντόμευσης για το Jeliot.bat με jeliot.ico
2. Στο Linux/*nix θα πρέπει να δώσετε την εντολή: java –jar jeliot.jar στον κατάλογο από όπου το Jeliot αποσυμπιέστηκε. Στην πραγματικότητα, αυτό ακριβώς είναι που κάνει το Jeliot.bat. Τώρα θα πρέπει το Jeliot 3 να υπάρχει στην οθόνη σας. Το σχήμα 4.7 δείχνει την οθόνη εκκίνησης του Jeliot.

4.3 Το Γραφικό περιβάλλον εργασίας (graphical user interface)

Το Jeliot χρησιμοποιείται πάντα μέσω του γραφικού περιβάλλοντος (GUI). Ένα στιγμιότυπο της εκτέλεσης της εφαρμογής μέσα στο γραφικό περιβάλλον φαίνεται παρακάτω στο σχήμα 4.8.



Σχήμα 4.8: Το Γραφικό περιβάλλον εργασίας (graphical user interface)

4.3.1 Τμήμα πηγαίου κώδικα (source frame)

Το αριστερό τμήμα της οθόνης του γραφικού περιβάλλοντος περιλαμβάνει έναν επεξεργαστή κειμένου, το τμήμα αυτό αποτελείται από αριθμημένες γραμμές και ονομάζεται τμήμα πηγαίου κώδικα όπου και συντάσσεται ο πηγαίος κώδικας του προγράμματος. Όταν ανοίγετε ή δημιουργείτε ένα νέο αρχείο, ενεργοποιείται το τμήμα του πηγαίου κώδικα, όπου δίνεται η δυνατότητα της επεξεργασίας του πηγαίου κώδικα. Επιλέγοντας το κουμπί: Compile (σύνταξη) θα μετακινηθεί η εργαλειοθήκη (toolbar) επεξεργασίας του πηγαίου κώδικα από το προσκήνιο και μετατρέπεται το τμήμα του πηγαίου κώδικα σε κατάσταση αποκλείοντας τη δυνατότητα επεξεργασίας του πηγαίου κώδικα. Ανοίγει η αυλαία του χώρου της δυναμικής παρουσίασης της εκτέλεσης του πηγαίου κώδικα.

Μετά την ολοκλήρωση της διαδικασίας αναπαράστασης της εκτέλεσης των εντολών του προγράμματος που υπάρχουν στο χώρο πηγαίου κώδικα, επιλέγοντας το κουμπί: Edit, το τμήμα του πηγαίου κώδικα θα επανέλθει στην κατάσταση διόρθωσης και πλέον δίνεται η δυνατότητα πάλι να επεξεργαστεί ο κώδικας του προγράμματος.

Κατά τη διάρκεια της δυναμικής αναπαράστασης, ο τμήμα του κώδικα (code) που βρίσκεται σε κατάσταση αναπαράστασης τονίζεται στο τμήμα του πηγαίου κώδικα. Για παράδειγμα, παρατηρώντας στο σχήμα 8 την εντολή που τονίζεται με κόκκινο χρώμα και παρατηρώντας συγχρόνως το τμήμα δυναμικής αναπαράστασης δείχνει τον υπολογισμό: n-1, η ίδια λειτουργία τονίζεται και στο τμήμα του πηγαίου κώδικα. Ένα άλλο παράδειγμα: ενώ εισάγεστε σε μία εντολή επανάληψης, το πλαίσιο δυναμικής αναπαράστασης δίνει μήνυμα για την συγκεκριμένη εντολή και το τμήμα του πηγαίου κώδικα θα υπογραμμίσει όλο το τμήμα που περιλαμβάνεται στην ειδική δομή της εντολής επανάληψης. Θα αντιληφθείτε πώς ακριβώς λειτουργεί μόλις το δείτε να συμβαίνει.

4.3.2 Πλαίσιο δυναμικής αναπαράστασης (animation frame)

Το τμήμα δυναμικής αναπαράστασης αποτελεί το βασικότερο κομμάτι της εφαρμογής του Jeliot. Είναι το τμήμα εκείνο όπου οποιαδήποτε δυναμική αναπαράσταση της εκτέλεσης του πηγαίου κώδικα λαμβάνει χώρα. Ενώ η εφαρμογή βρίσκεται σε κατάσταση εισαγωγής-διόρθωσης του πηγαίου κώδικα, το τμήμα αυτό

καλύπτεται από μία μπλε κουρτίνα. Όταν λοιπόν μετακινείστε στην κατάσταση δυναμικής αναπαράστασης, η κουρτίνα αυτή ανοίγει και αποκαλύπτει ένα φόντο απαλής καφετί απόχρωσης. Μόλις ξεκινήσετε την διαδικασία δυναμικής αναπαράστασης το πλαίσιο χωρίζεται σε τέσσερις ξεχωριστές περιοχές με άσπρες γραμμές με παύλες (dashed white lines). Οι περιοχές στη σειρά από αριστερά πρός τα δεξιά και από πάνω πρός τα κάτω είναι οι εξής: η περιοχή μεθόδων (Method Area), η περιοχή αξιολόγησης εκφράσεων (Express Evaluation) και τέλος, η περιοχή στιγμιότυπων και πινάκων (Instance and Array area).

(Method Area) Η περιοχή μεθόδων περιέχει πλαίσια ενεργοποίησης για όλες τις τρέχουσες μεθόδους που επεξεργάζονται. Όταν δεν μένει τίποτε άλλο πια στην περιοχή μεθόδων, δεν υπάρχει κάτι αναπαραστάσιμο. Τα πλαίσια ενεργοποίησης δίνονται ως κουτιά μέσα στα οποία υπάρχουν μεταβλητές (variables) που αναπαρίστανται ως μικρότερα κουτιά (subboxes). Οι μεταβλητές επιστροφής αναπαρίστανται ως ένα μεγαλύτερο κουτί το οποίο περιέχει την τιμή. Οι βασικοί τύποι μεταβλητών ή οι συμβολοσειρές (strings), η αξία παρατίθεται είτε συνεχόμενα με το όνομα είτε ως σύνδεσμος της περιοχής στιγμιότυπων και πινάκων (instance and array area) είτε τέλος ως το σύμβολο της γείωσης σε περίπτωση που είναι μηδέν (null).

(Expression Evaluation Area) Η περιοχή αξιολόγησης εκφράσεων είναι ακριβώς αυτό που λέει ο τίτλος της, μια περιοχή που παρουσιάζει την αξιολόγηση των εκφράσεων του πηγαίου κώδικα. Οποιαδήποτε εντολή εκτελείται, εδώ θα είναι το σημείο όπου θα πραγματοποιείται. Πληροφορίες για τα αποτελέσματα της αξιολόγησης εκφράσεων επίσης προβάλλονται εδώ, όπως επίσης και τα κουτιά διαλόγου για εισαγωγή του χρήστη.

Οποιαδήποτε στιγμή η εκτέλεση του πηγαίου κώδικα χρειαστεί κάποιες τιμές (literals), για να γίνει η δυναμική αναπαράσταση της το παίρνει από το κουτί των σταθερών τιμών (Constants Box) της περιοχής των σταθερών (constant area). Κάτι παρόμοιο διατίθεται για όλα τις τιμές, σε όποια μορφή και να παρουσιάζονται αυτά.

(Instance and Array area) Περιοχή Στιγμιότυπων και Πινάκων περιέχει δυναμικά καταχωρημένα αντικείμενα όπως στιγμιότυπα κλάσεων και πινάκων (arrays). Αυτές

συνδέονται σε πλαίσια ενεργοποίησης στην περιοχή μεθόδων (method area) μέσω συνδέσμων.

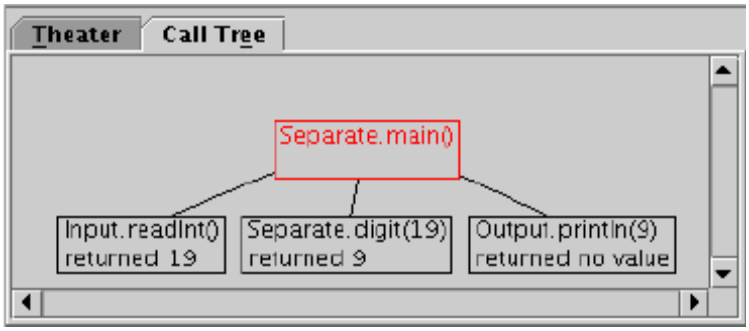
Τα αντικείμενα στο πλαίσιο αναπαράστασης ακολουθούν την παρακάτω χρωματική κωδικοποίηση:

Object	color	object	color
Methods	Pink box with white reader	Long	Light green
Floats	Purple	Strings	Red
Integers	Brown	Doubles	What is this color?
Chars	Green	Class objects	Yellow

Πίνακας 4.1: Χρωματική κωδικοποίηση στο πλαίσιο αναπαράστασης

4.3.3 Δέντρο κλήσης (Call Tree)

Το δέντρο κλήσης παρουσιάζει τις εκτελούμενες λειτουργίες κλήσης και τις αξίες επιστροφής τους, όπως ένα δέντρο, ClassName.main( ) όντας η φυσική ρίζα. Μπορείτε να δείτε το δέντρο κλήσης κάνοντας κλίκ επάνω στην επιλογή Call Tree tab στην κορυφή του πλαισίου δυναμικής αναπαράστασης. Το δέντρο κλήσης στη συνέχεια θα παρουσιαστεί στο τμήμα δυναμικής αναπαράστασης. Η επιστροφή πίσω στην δυναμική αναπαράσταση γίνεται κάνοντας κλίκ στην επιλογή Theater tab. Μπορείτε λοιπόν να περάσετε από την μία επιλογή στην άλλη όποτε εσείς το επιθυμείτε. Ωστόσο, τα κουτιά διαλόγου θα εμφανιστούν μόνον όταν εμφανιστεί η επιλογή theater. Θα πρέπει να το θυμάστε αυτό εάν επιθυμείτε να ακολουθήσετε την εκτέλεση (execution) από την εικόνα του δέντρου κλήσης. Ένα παράδειγμα του δέντρου κλήσης φαίνεται στο σχήμα 4.9.



Σχήμα 4.9: Δέντρο κλήσης (Call Tree)

4.3.4 Μενού (Menus)

Τα μενού στο Jeliot είναι αρκετά κατανοητά διότι είναι αυτό - επεξηγούμενα. Το μενού προγράμματος (Program menu) περιέχει εντολές για λειτουργίες φακέλου. Τα μενού :new, open, save, exit, και edit περιέχουν τις λειτουργίες του πίνακα (clipboard) cut (κόψε), copy (αντέγραψε), save (αποθήκευσε) και select all (επιλογή όλων). Μια γρήγορη περίληψη του τι ακριβώς κάνουν οι παραπάνω εντολές φαίνονται στον πίνακα 4.2. Μπορείτε επίσης να βρείτε τις περισσότερες από τις εντολές και στο : Toolbar στη κορυφή του πλαισίου πηγαίου κώδικα (source frame).

Το μενού Control (Έλεγχος) περιέχει τις εντολές : Edit και compile (Εκτέλεση). Το μενού Δυναμικής Αναπαράστασης (Animation) περιέχει εντολή για τον έλεγχο της ροής της δυναμικής αναπαράστασης. Ξανά, οι περισσότερες εντολές στα δυο αυτά μενού μπορούν να βρεθούν από το toolbar στο τέλος της οθόνης (σχήμα 4.8).

Επιπρόσθετα με τις εντολές που επίσης βρίσκονται τοποθετημένες στο toolbar, το μενού δυναμικής αναπαράστασης περιέχει δυο τις σημαντικότερες εντολές: Pause on message και Run until…. Η πρώτη εντολή αποτελεί ένα κουτί ελέγχου, το οποίο διακόψει την δυναμική αναπαράσταση σε οποιοδήποτε μήνυμα (such as/όπως :"continuing for loop") εάν αυτό είναι έτοιμο. Η δεύτερη εντολή θα ζητήσει έναν αριθμό γραμμής (linenumber) και θα τοποθετήσει εκεί ένα σημείο διακοπής της εκτέλεσης (breakpoint). Όταν η εκτέλεση φτάσει στην σειρά αυτή, τερματίζεται.

Σημειώστε ότι το μενού Δυναμικής Αναπαράστασης δεν λειτουργεί εφόσον βρίσκεται σε κατάσταση επεξεργασίας (editing state). Με τον ίδιο τρόπο τα μενού Program και edit δεν λειτουργούν σε κατάσταση δυναμικής αναπαράστασης.

COMMAND (ΕΝΤΟΛΗ)	DESCRIPTION (ΠΕΡΙΓΡΑΦΗ)
New	Εισάγει ένα άδειο περίγραμμα στο πλαίσιο πηγαίου κώδικα.
Open	Ανοίγει τον πίνακα για την επιλογή του αρχείου που θα ανοίξει το πλαίσιο πηγαίου κώδικα.
Save	Αποθηκεύει το πρόγραμμα του πλαισίου πηγαίου κώδικα σε επιλεγμένο φάκελο.
Exit	Έξοδος από το Jeliot.
Cut	Κόβει το επιλεγμένο κείμενο από το πλαίσιο πηγαίου κώδικα και το τοποθετεί επάνω στον πίνακα επικόλλησης (clipboard)
Copy	Αντιγράφει το επιλεγμένο κείμενο από το πλαίσιο πηγαίου κώδικα (source frame) στον πίνακα επικόλλησης (clipboard)
Paste	Τοποθετεί το περιεχόμενο του πίνακα επικόλλησης στο πλαίσιο πηγαίου κώδικα ξεκινώντας από την σειρά/γραμμή όπου βρίσκεται ο δείκτης θέσεις (cursor)
Select All	Επιλέγει ότι υπάρχει στο πλαίσιο πηγαίου κώδικα (source frame)

Πίνακας 4.2: Εντολές του προγράμματος με την αντίστοιχη περιγραφή

Οι εναλλακτικές επιλογές (options) περιέχουν διαφορετικές επιλογές οι οποίες δημιουργούνται για ξεχωριστές περιστάσεις :

- **Save Files In Unicode:** Εάν αντιμετωπίζετε προβλήματα με τους βασικούς χαρακτήρες κατά την αποθήκευση η εισαγωγή αρχείων ή κατά την διάρκεια της οπτικοποίησης, επιλέξτε την μέθοδο αποθήκευσης: Save Files In Unicode.
- **Save On Compilation:** Αποθηκεύστε ένα αρχείο που περιλαμβάνει τον πηγαίο κώδικα (source file) αυτομάτως πριν την μετάφραση του προγράμματος (compilation)
- **Show Strings as Objects:** Αλλάζει τον τρόπο παρουσίασης μιας μεταβλητής συμβολοσειράς (string value) σε αντικείμενο.
- **Do Garbage Collection:** Επιτρέπει την συλλογή σκουπιδιών κατά τη διάρκεια του προγράμματος οπτικοποίησης (visualization). Κάτι τέτοιο είναι ιδιαίτερος χρήσιμο όταν οι συμβολοσειρές (strings) φαίνονται ως αντικείμενα επειδή διαφορετικά η οθόνη μπορεί να καταρρεύσει η εκτέλεση του προγράμματος.
- **Ask For Command Line Parameters:** Όταν ξεκινά η δυναμική αναπαράσταση (animation), θα σας ζητηθούν οι παράμετροι για την κλήση της μεθόδου.
- **Ask For Method:** Όταν η δυναμική αναπαράσταση ξεκινά, θα σας ζητηθεί όνομα μεθόδου που θα καλέσετε αντί της βασικής μεθόδου.
- **Use Null Parameter To Call Main:** Εάν επιλεγεί η βασική (main) μέθοδος θα κληθεί με null παράμετρο, με αυτόν τον τρόπο αποτρέπεται η προσπάθεια δυναμικής αναπαράστασης ενός κενού πίνακα.
- **As Questions During Animation:** Όποτε κάποια έκφραση πρόκειται να αξιολογηθεί, ένα παράθυρο που εμφανίζεται και θα ζητήσει το αποτέλεσμα. Ωστόσο, πρόσφατες ερωτήσεις παράγονται μόνον για δηλώσεις αναθέσεων (assignment statements).
- **Pause On Message:** Αυτομάτως διακόπτεται η δυναμική αναπαράσταση για να παρουσιαστεί το μήνυμα που επεξηγεί την διαδικασία εκτέλεσης του προγράμματος.
- **Show History View:** Επιτρέπει ή όχι την παραγωγή ιστορικών εικόνων που δείχνουν τα στιγμιότυπα προηγούμενων αναπαραστάσεων εντός της σκηνής

αναπαράστασης. Σημειώστε ότι η παραγωγή ιστορικών εικόνων παίρνει κάποιο χρόνο και ίσως να επιβραδύνει την οπτικοποίηση. Εάν η αναπαράσταση είναι αργή καθιστά αδύνατη την επιλογή αυτή.

- **Select Font Of Code:** Επιλέξτε το φόντο για τον επεξεργαστή κώδικα (code editor).

Το τελευταίο μενού είναι το Help:Εκεί μπορείτε να βρείτε πληροφορίες για το Jeliot (about) όπως και για το έγγραφο Βοήθειας (Help). Το έγγραφο Βοήθειας ανοίγει ένα δικό του παράθυρο όταν το επιλέξετε από το μενού, η πατώντας το πλήκτρο F1.

4.3.5 Toolbar Buttons: Κουμπιά Εργαλειοθήκης

Σε κατάσταση επεξεργασίας (Edit), υπάρχει μια κινητή εργαλειοθήκη που περιλαμβάνει εντολές για την διαχείριση αρχείων και του πίνακα επικόλλησης (clipboard) στην κορυφή του πλαισίου του πηγαίου κώδικα (σχήμα 4.10). Αυτά λειτουργούν με τον ίδιο τρόπο όπως και οι εντολές από το μενού Program και Edit. Κατά τη διάρκεια μετάφρασης του πηγαίου κώδικα, η εργαλειοθήκη βρίσκεται κρυμμένη κάπου παρακάτω από το πλαίσιο του πηγαίου κώδικα, η απλά δεν φαίνεται όπως στην περίπτωση που το έχετε μετακινήσει και το έχετε αφήσει στο δικό του παράθυρο.

Σχήμα 4.10: Μενού προγράμματος και λειτουργίες φακέλου και πίνακα επικόλλησης της εργαλειοθήκης

4.3.6 Edit and Compile: Επεξεργασία και μετάφραση

Η εργαλειοθήκη φαίνεται στο σχήμα 4.11 έχει κουμπιά που λέγονται edit και compile. Με τα κουμπιά αυτά ελέγχετε αν το Jeliot είναι σε κατάσταση Επεξεργασίας η δυναμικής αναπαράστασης. Από την κατάσταση επεξεργασίας, μπορείτε να ξεκινήσετε την δυναμική αναπαράσταση κάνοντας κλικ επάνω στο compile. Μπορείτε να επιστρέψετε για να διορθώσετε τον πηγαίο κώδικα του προγράμματος οποιαδήποτε στιγμή από την κατάσταση δυναμικής αναπαράστασης επιλέγοντας Edit.

Σχήμα 4.11: Κουμπιά ελέγχου Επεξεργασίας, Μετάφρασης και Δυναμικής Αναπαράστασης

4.3.7 Animation Controls: Έλεγχος Δυναμικής Αναπαράστασης

Η δυναμική αναπαράσταση μπορεί να ελεγχθεί από τα κουμπιά VCR του σχήματος 4.11 στην κάτω δεξιά γωνία του παραθύρου του Jeliot. Ο πίνακας 4.3 περιγράφει την χρήση των κουμπιών ελέγχου της Δυναμικής Αναπαράστασης που μπορούν να επιλεγούν από την εργαλειοθήκη.

ΚΟΥΜΠΙ (BUTTON)	ΛΕΙΤΟΥΡΓΙΑ (FUNCTION)
Step	Συνεχίζει την δυναμική αναπαράσταση βήμα βήμα στην καθορισμένη ταχύτητα
Play	Συνεχίζει την δυναμική αναπαράσταση με καθορισμένη ταχύτητα έως ότου πατήσετε stop η τερματιστεί η εφαρμογή
Pause	Κάνει παύση στην συνεχόμενη δυναμική αναπαράσταση.
Rewind	Πάει πίσω στην αρχή του προγράμματος
Animation speed	Ρυθμίζει την ταχύτητα της δυναμικής αναπαράστασης. Προς αριστερά είναι πιο αργή, προς δεξιά είναι πιο γρήγορη. Μπορείτε επίσης να ελέγξετε την ταχύτητα της παρουσίασης σε κατάσταση αναπαράστασης βήμα βήμα από το μενού αναπαράστασης

Πίνακας 4.3: Εντολές της Εργαλειοθήκης της Δυναμικής Αναπαράστασης

4.3.8 Output Frame : Πλαίσιο Παρουσίασης Αποτελεσμάτων

Το πλαίσιο παρουσίασης βρίσκεται στην κάτω δεξιά γωνία της οθόνης. Όπως φαίνεται στο σχήμα 4.12, είναι ένα λευκό πλαίσιο κειμένου (textbox), στο οποίο παρουσιάζονται τα δεδομένα που παράγονται από το πρόγραμμα αναπαράστασης. Όλες οι τιμές που παράγονται και εμφανίζονται στο χώρο αυτό, χρησιμοποιούνται από το πλαίσιο δυναμικής αναπαράστασης και επιλέγονται για χρήση από ένα χεράκι που βγαίνει από το κουτί. Κάνοντας κλικ επάνω στο πλαίσιο παρουσίασης αποτελεσμάτων παρουσιάζεται ένα μενού με την μόνη επιλογή την δυνατότητα καθαρισμού της οθόνης. Μπορείτε να επιλέξετε την εναλλακτική αυτή επιλογή οποιαδήποτε στιγμή για να σβήσετε τα πάντα στο κουτί παρουσίασης αποτελεσμάτων (output box).

Σχήμα 4.12: Κουτί παρουσίασης αποτελεσμάτων, καθάρισμα εναλλακτικής επιλογής στο μενού και χεράκι επιλογής αξίας

4.3.9 Error Display : Παράσταση Λαθών

Στην περίπτωση που υπάρχει κάποιο λάθος στον κώδικα, το Jeliot στις περισσότερες φορές θα σας ειδοποιήσει σχετικά με αυτό, επιλέγοντας ένα από τα κουμπιά Play ή Step για πρώτη φορά. Εάν προσπαθείτε να χρησιμοποιήσετε μια μεταβλητή που είναι αδύνατον να χρησιμοποιηθεί, η δυναμική αναπαράσταση θα συνεχιστεί κανονικά έως ότου γίνει το λάθος, όπου και τερματίζετε παρουσιάζοντας ένα μήνυμα λάθους (error message).

Η ειδοποίηση για το λάθος θα παρουσιαστεί στο πλαίσιο δυναμικής αναπαράστασης. Κάνοντας κλικ στο OK, θα γυρίσετε πίσω στην οθόνη δυναμικής αναπαράστασης, αλλά δεν θα μπορείτε να συνεχίσετε με την αναπαράσταση έως ότου διορθώσετε τον

κώδικα που περιλαμβάνει το σφάλμα. Αυτό βέβαια εξακολουθεί να είναι δική σας δουλεία το Jeliot δεν μπορεί ακόμη να κάνει κάτι τέτοιο για έσας.

4.4 Java Issues: Θέματα σχετιζόμενα με την JAVA

Υπάρχουν δυο ασυμφωνίες μεταξύ Jeliot και Java.

1. Όλες οι κατηγορίες πρέπει να είναι σε ένα ενιαίο αρχείο.
2. Για I/O (είσοδο/έξοδο), εισάγετε το πακέτο : jeliot.io.*; το οποίο παρέχει τις μεθόδους :void Output.println( ), int Input.readInt( ), double Input.readDouble( ),char Input.readchar( ),String Input.readString( ). Τυπική διαδικασία εξόδου υποστηρίζεται εξ ορισμού.

Το Jeliot χρησιμοποιεί την Dynamic Java (**http://koala.ilog.fr/djava/**) σαν front-end και έτσι δέχεται σχεδόν όλα τα χαρακτηρίσθηκα της Java τα οποία θα θέλατε για να χρησιμοποιήσετε τον εισαγωγικό προγραμματισμό. Ωστόσο, η εφαρμογή της δυναμικής αναπαράστασης ίσως και να μην χρησιμοποιεί όλα αυτά τα χαρακτηρίσθηκα. Η νεότερη έκδοση της εφαρμογής περιλαμβάνει τα εξής χαρακτηριστικά:

- Τιμές για τύπους String, όλους τους βασικούς τύπους μεταβλητών και μονοδιάστατες πίνακες.
- Στατικές μεταβλητές.
- Εκφράσεις συμπεριλαμβανόμενων όλων των βασικών ή δυαδικών λειτουργιών εκτός της instanceof.
- Όλες τις δηλώσεις έλεγχου (if, while, etc.).
- Εκφράσεις σύγκρισης (exp?exp1:exp2).
- Κλήση μεθόδου, συμπεριλαμβανομένης της επαναλαμβανόμενης κλήσης.
- Κατασκευαστές, κατανομή των αντικειμένων και κλήση των μεθόδων στα αντικείμενα.

Δεν επιτρέπεται η χρήση των παρακάτω χαρακτηριστικών:

- Ανώτερες προσβάσεις πεδίων.
- Πίνακες που περιλαμβάνουν συστατικά που αναφέρονται σε άλλους τύπους (εκτός του τύπου string)
- Δυο ή και περισσότερες πολυδιάστατες παραστάσεις.
- Αρχικοποίηση Πινάκων.
- Οι μέθοδοι Java 2 SDK API δεν μπορούν να επιστρέψουν τύπους αντικειμένων (εκτός του τύπου String) or array types (π.χ. object.getClass( ) που επιστρέφουν ένα στιγμιότυπο μιας κλάσης.
- Η χρησιμοποιηθείσα μέθοδος κλάσεων hashCode( ) πρέπει να επιστρέφει πάντα μια μοναδική τιμή.

Κεφάλαιο 5

Μαθήματα Jeliot

Σ' αυτό το μάθημα θα κάνουμε την πρώτη μας εφαρμογή σε Jeliot η οποία (φυσικά) θα είναι το πρόγραμμα HelloWorld! και θα δούμε πως είναι μία εφαρμογή σε Jeliot.

5.1 Τι είναι μία εφαρμογή Jeliot

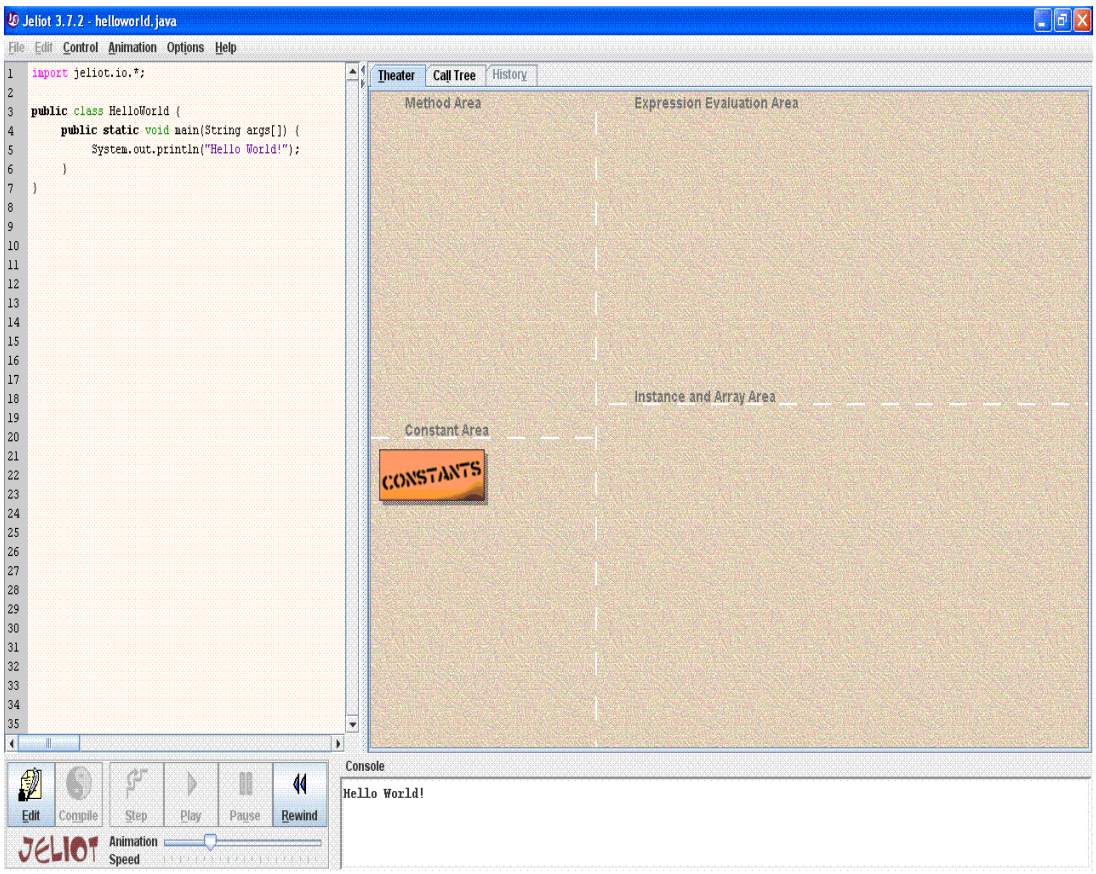
Μία Jeliot εφαρμογή θα πρέπει να έχει και μία συνάρτηση main (κύρια) που είναι και το σημείο εισόδου στην εφαρμογή (δηλαδή από εκεί ξεκινά το πρόγραμμα). Φυσικά μία συνάρτηση main θα πρέπει να βρίσκεται μέσα σε μία τάξη (class). Η Java είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού και επομένως δεν μπορούμε να έχουμε καμία συνάρτηση που δεν είναι τμήμα μιας τάξης στη Java. Θα πρέπει να έχουμε μία έστω τάξη στην οποία θα περιέχεται η main ακόμα και αν δεν την χρειαζόμαστε!

Η πρώτη μας εφαρμογή: Το πρόγραμμα "Hello World!"

Σ’ αυτό το σημείο θα κατασκευάσουμε την πρώτη μας εφαρμογή Jeliot που ο κώδικάς της είναι ο ακόλουθος:

```
public class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello World!!!");
    }
}
```

Το πρόγραμμα HelloWorld στην εφαρμογή Jeliot



Σχήμα 5.1: Το πρόγραμμα Hello World

Στο σχήμα 5.1 έχουμε το αποτέλεσμα του προγράμματος στην console της εφαρμογής Jeliot με το ακόλουθο : Hello World!

5.2 Μεταβλητές (variables)

Οι μεταβλητές είναι οι θέσεις μνήμης στις οποίες ένα πρόγραμμα τοποθετεί τα δεδομένα του κατά τη διάρκεια της λειτουργίας του. Σ' αυτό το εδάφιο θα μάθουμε πως δηλώνουμε μεταβλητές, ποιοι είναι οι τύποι των μεταβλητών στο Jeliot, πως πρέπει να είναι το όνομα μιας μεταβλητής και ποια είναι η εμβέλεια της μεταβλητής.

5.2.1 Δήλωση μεταβλητών

Η δήλωση μιας μεταβλητής έχει πάντοτε την μορφή:

<Τύπος Μεταβλητής> <Όνομα Μεταβλητής>;

Για παράδειγμα η ακόλουθη δήλωση αφορά μία ακέραια μεταβλητή που ονομάζεται count:

int count;

Προαιρετικά μπορεί να υπάρχει και μία αρχικοποίηση της μεταβλητής ταυτόχρονα με την δήλωσή της. Για παράδειγμα η ακόλουθη δήλωση αρχικοποιεί την count στη τιμή 0.

int count = 0;

Στη συνέχεια και για την περιοχή εμβέλειας της συγκεκριμένης μεταβλητής μπορούμε να χρησιμοποιήσουμε την count όπου επιτρέπεται να χρησιμοποιηθούν ακέραιες τιμές. Ο τύπος επομένως της μεταβλητής καθορίζει τις επιτρεπτές ενέργειες που μπορούν να γίνουν σε αυτή τη μεταβλητή.

5.2.2 Τύποι μεταβλητών

Οι τύποι των μεταβλητών στο Jeliot δίνονται από τον ακόλουθο πίνακα 5.1

Τύπος	Μέγεθος και μορφή	Περιγραφή
Ακέραιες μεταβλητές		
byte	8 bit σε μορφή συμπληρώματος ως προς 2	Ακέραιος μήκους ενός byte
short	16 bit σε μορφή συμπληρώματος ως προς 2	Ακέραιος μικρού μήκους
int	32 bit σε μορφή συμπληρώματος ως προς 2	Ακέραιος
long	64 bit σε μορφή συμπληρώματος ως προς 2	Ακέραιος μεγάλου μήκους
Πραγματικές μεταβλητές		
float	32 bit σε μορφή IEEE 754	Πραγματικός απλής ακρίβειας
double	64 bit σε μορφή IEEE 754	Πραγματικός διπλής ακρίβειας
Άλλοι Τύποι		
char	16-bit Unicode χαρακτήρας	Ένας χαρακτήρας
boolean	true ή false	Μια boolean τιμή αληθής ή ψευδής

Πίνακας 5.1: Τύποι μεταβλητών στο Jeliot

5.3 Τελεστές (operators)

Σ' αυτό το μάθημα θα αναφερθούμε στους τελεστές του Jeliot και στις έγκυρες χρήσεις τους.

5.3.1 Τι είναι οι τελεστές;

Οι τελεστές είναι σύμβολα τα οποία συμβολίζουν την τέλεση μιας λειτουργίας σε ένα, δύο ή και τρεις τελεστές (operators). Υπάρχουν λοιπόν τρία είδη τελεστών ως προς τον αριθμό των τελεστέων στους οποίους επενεργούν:

Μοναδιαίοι (unary) Τελεστές: Αυτοί επενεργούν σε ένα και μόνο τελεστέο. Για παράδειγμα ο τελεστής ++ αυξάνει κατά 1 τον τελεστέο του όπως στην έκφραση ++count.

Δυαδικοί (binary) Τελεστές: Αυτοί επενεργούν σε δύο τελεστέους. Για παράδειγμα ο τελεστής + της αριθμητικής πρόσθεσης απαιτεί δύο τελεστέους όπως στην έκφραση op1 + op2.

Τριαδικοί (ternary) Τελεστές: Αυτοί επενεργούν σε τρεις τελεστέους. Η Java έχει μόνο ένα τέτοιο τελεστή την έκφραση if που είναι ο :?, όπως στην ακόλουθη έκφραση a>4?a-4:4.

5.3.2 Αριθμητικοί τελεστές

Οι αριθμητικοί τελεστές χρησιμοποιούνται για τις αριθμητικές πράξεις και είναι αυτοί που φαίνονται στον ακόλουθο πίνακα 5.2.

Τελεστής	Χρήση	Περιγραφή
+	n1+n2	Αποτιμάται στο άθροισμα των n1 και n2
-	n1-n2	Αποτιμάται στη διαφορά των n1 και n2
*	n1*n2	Αποτιμάται στο γινόμενο των n1 και n2
/	n1 / n2	Αποτιμάται στο πηλίκο των n1 και n2
%	n1%n2	Αποτιμάται στο υπόλοιπο της διαίρεσης του n1 δια του n2
++	++n ***ή*** ++n	Αυξάνει την μεταβλητή n κατά 1
--	-- n ***ή*** n--	Μειώνει την μεταβλητή n κατά 1

Πίνακας 5.2: Αριθμητικοί τελεστές

Ο τελεστής ++ όπως και ο --, μπορούν να εφαρμοστούν είτε από τα αριστερά (προ-αύξηση & προ-μείωση) είτε από τα δεξιά (μετά-αύξηση & μετά-μείωση).

Προ-αύξηση και προ-μείωση σημαίνει ότι αν η συγκεκριμένη έκφραση αποτελεί τμήμα μιας ευρύτερης έκφρασης πρώτα θα υπολογιστεί η νέα τιμή της μεταβλητής που αυξάνουμε ή μειώνουμε και μετά θα υπολογιστεί η έκφραση με τη χρήση της νέας τιμής, όπως στο παράδειγμα:

int x=10;

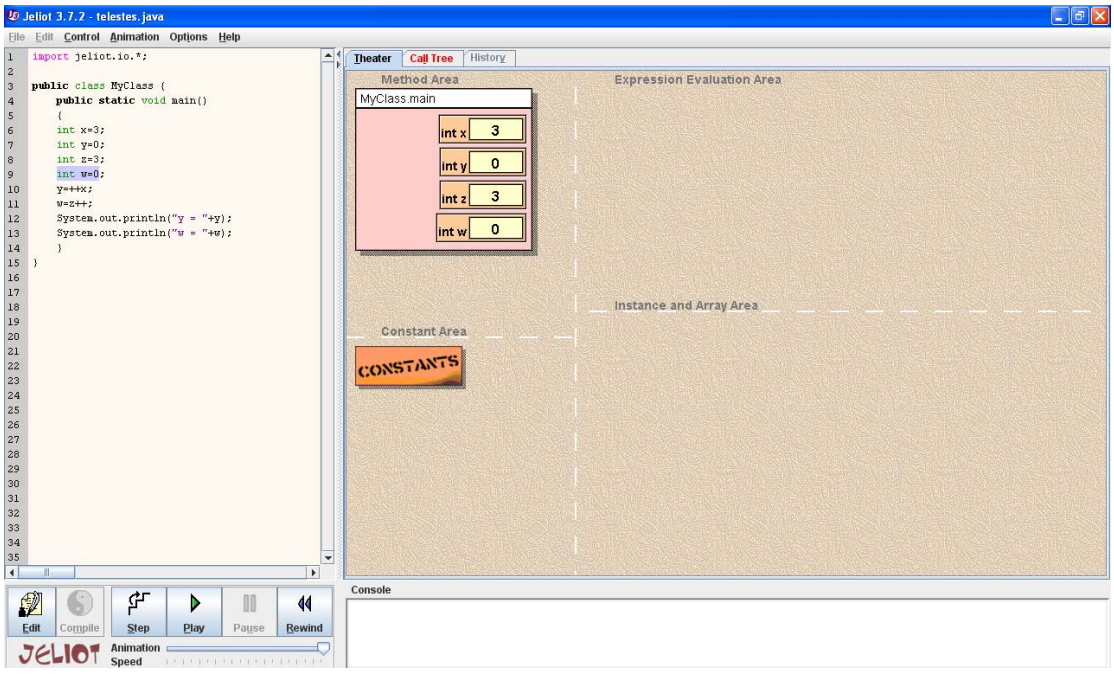
y = ++x; //Στο y θα εισαχθεί η τιμή 11, και φυσικά το x θα γίνει 11

Μετά-αύξηση και μετά-μείωση σημαίνει ότι αν η συγκεκριμένη έκφραση αποτελεί τμήμα μιας ευρύτερης έκφρασης πρώτα θα υπολογιστεί η έκφραση με βάση τη τιμή εκείνη τη στιγμή που έχει η μεταβλητή που αυξάνουμε ή μειώνουμε και μετά θα αυξηθεί ή θα μειωθεί η μεταβλητή όπως στο παράδειγμα:

int x=10;

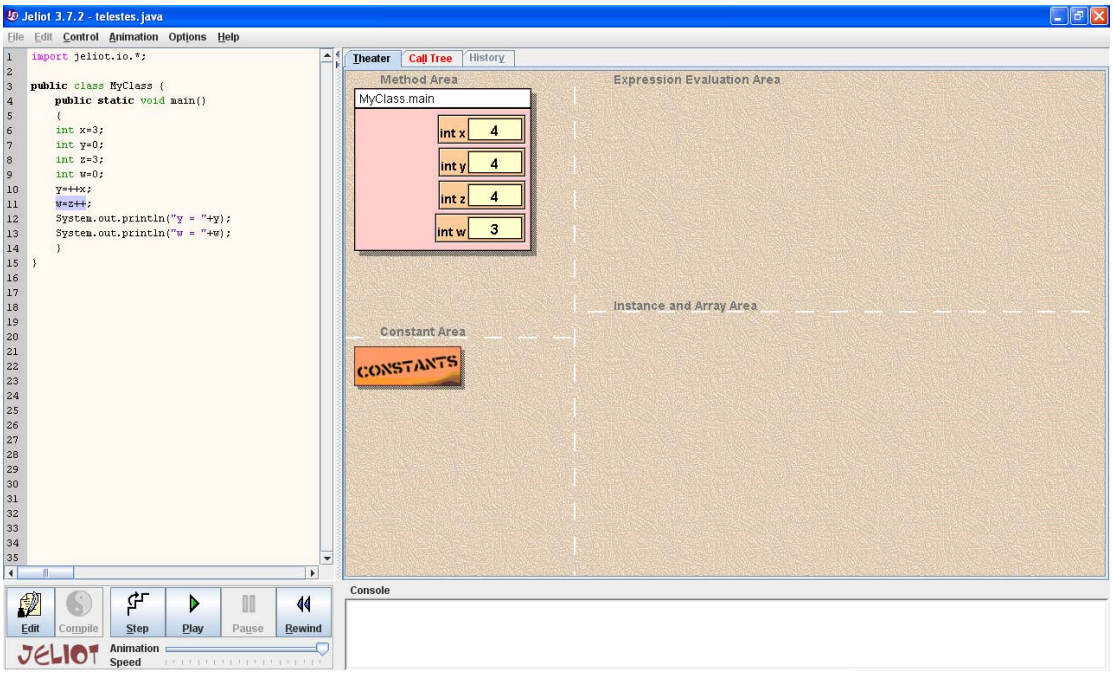
y = x++; //Στο y θα εισαχθεί η τιμή 10, και φυσικά το x θα γίνει 11

Ένα παράδειγμα στην εφαρμογή Jeliot



Σχήμα 5.2α: Παράδειγμα χρήσης αριθμητικών τελεστών

Στο σχήμα 5.2α δηλώνονται οι μεταβλητές και παίρνουν αρχικές τιμές x=3, y=0, z=3 και w=0

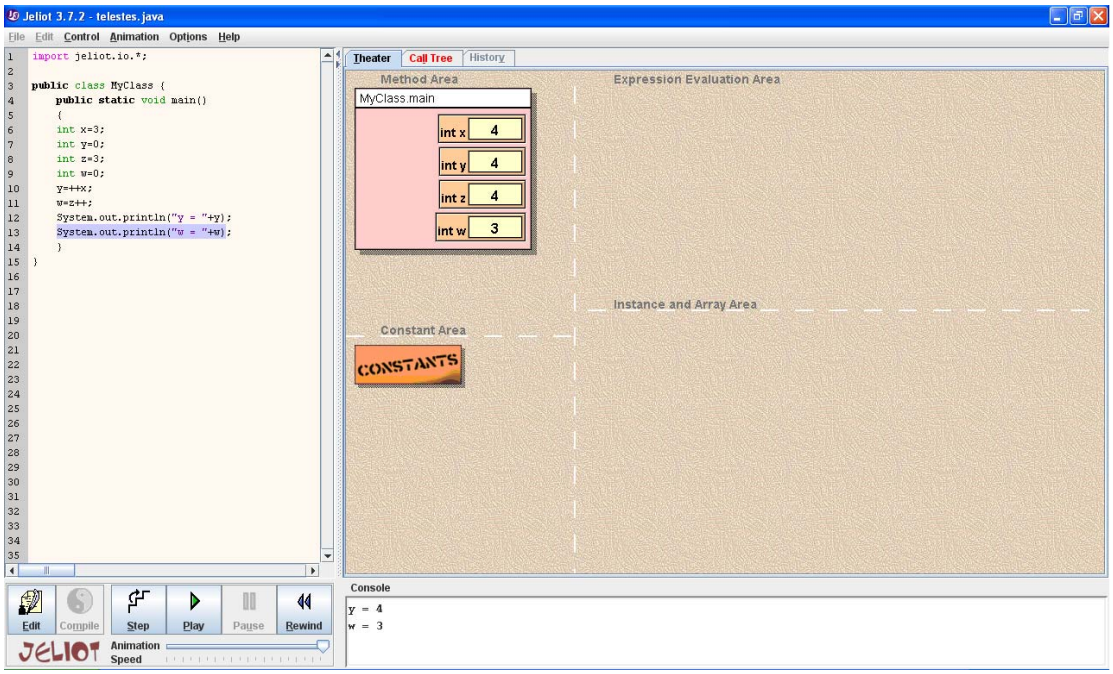


Σχήμα 5.2β: Παράδειγμα χρήσης αριθμητικών τελεστών

Στο σχήμα 5.2β οι εντολές :

y=++x; αυξάνεται πρώτα το x το οποίο γίνεται = 4 (x = 4) και στην συνέχεια το y παίρνει την τιμή του x (y = 4)

w=z++; το w παίρνει την τιμή του z (w = 3) και στην συνέχεια το z αυξάνεται κατά 1 (z = 4)



Σχήμα 5.2γ: Παράδειγμα χρήσης αριθμητικών τελεστών

Στο σχήμα 5.2γ έχουμε το αποτέλεσμα του προγράμματος στην console της εφαρμογής Jeliot με τα ακόλουθα : y=4, w=3

Κώδικας προγράμματος :

```
public class MyClass {
   public static void main()
   {
   int x=3;
   int y=0;
   int z=3;
   int w=0;
   y=++x;
   w=z++;
   System.out.println("y = "+y);
   System.out.println("w = "+w);
   }
}
```

5.3.3 Τελεστές σύγκρισης

Οι τελεστές σύγκρισης δίνονται στη συνέχεια στον πίνακα 5.3. Συγκρίνουν δύο τιμές και καθορίζουν τη σχέση μεταξύ τους. Η έκφραση που περιέχει ένα τελεστή σύγκρισης μπορεί να είναι true ή false

Τελεστής	Χρήση	Περιγραφή
>	n1>n2	true αν το n1 είναι μεγαλύτερο του n
>=	n1>=n2	true αν το n1 είναι μεγαλύτερο ή ίσο του n2
<	n1<n2	true αν το n1 είναι μικρότερο του n2
<=	n1<=n2	true αν το n1 είναι μικρότερο ή ίσο του n2
==	n1==n2	true αν το n1 είναι ίσο με το n2
!=	n1!=n2	true αν το n1 είναι διάφορο του n2

Πίνακας 5.3: Τελεστές σύγκρισης

5.3.4 Λογικοί τελεστές

Τους λογικούς τελεστές συνήθως τους χρησιμοποιούμε με τους τελεστές σύγκρισης για την κατασκευή ποιο πολύπλοκων λογικών εκφράσεων, εκφράσεων δηλαδή που μπορεί να είναι true ή false. Για παράδειγμα η ακόλουθη έκφραση θα είναι true αν η τιμή του x είναι μεταξύ 10 και 100:

x>=10 && x<=100

Οι λογικοί τελεστές είναι αυτοί που φαίνονται στον ακόλουθο πίνακα 5.4:

Τελεστής	Χρήση	Περιγραφή
&&	e1 && e2	true αν και η e1 και η e2 είναι true. Η e2 δεν αποτιμάται αν η e1 είναι false
||	e1 || e2	false αν και η e1 και η e2 είναι false. Η e2 δεν αποτιμάται αν η e1 είναι true
!	!e	Η λογική άρνηση της έκφρασης e
&	e1&e2	Το ίδιο με τον && μόνο που η e2 θα αποτιμηθεί ακόμα και αν η e1 είναι false
|	e1|e2	Το ίδιο με τον || μόνο που η e2 θα αποτιμηθεί ακόμα και αν η e1 είναι true

Πίνακας 5.4: Λογικοί τελεστές

Μερικές φορές είναι επιθυμητό να αποτιμηθεί και το δεύτερο μέρος μιας σύνθετης λογικής έκφρασης ακόμα και αν έχει υπολογιστεί η τιμή αλήθειας της έκφρασης.

Για παράδειγμα φανταστείτε την ακόλουθη έκφραση:

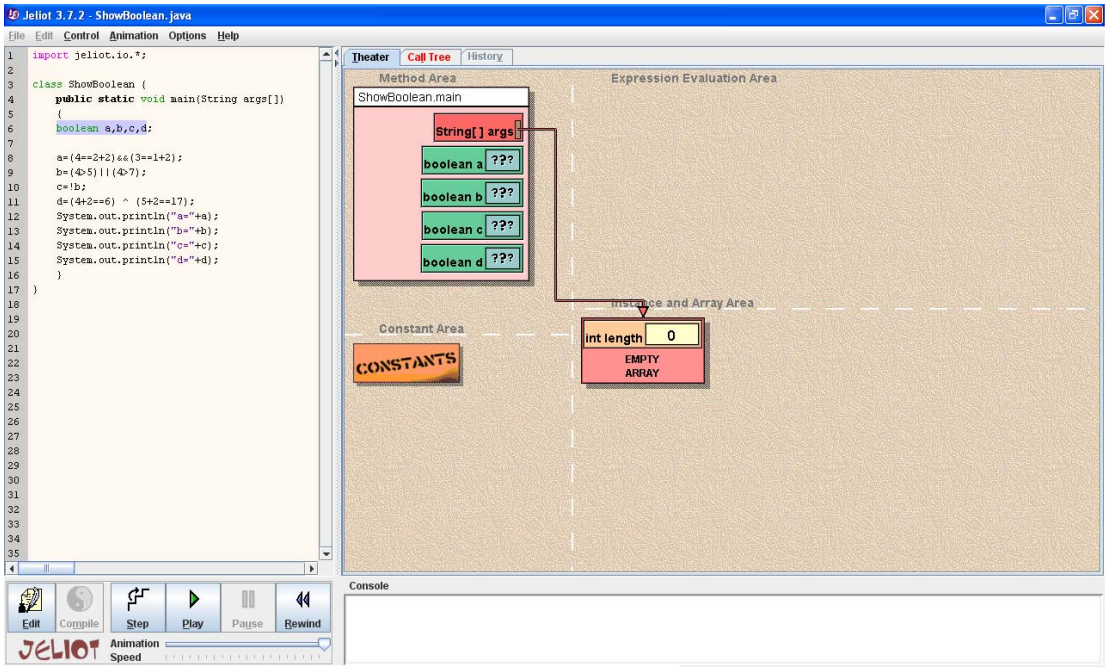
x>0 && someFunc(x)>10

Αν θέλουμε σ' αυτό το παράδειγμα να κληθεί οπωσδήποτε η συνάρτηση someFunc τότε θα πρέπει να χρησιμοποιήσουμε τον & αντί του &&, γιατί με τον && αν το x <=0 δεν θα αποτιμηθεί το δεύτερο μέρος της έκφρασης μια και η τιμή αλήθειας της έκφρασης θα είναι ούτως ή άλλως false, αφού το πρώτο μέρος της έκφρασης θα είναι false. Ανάλογη είναι και η χρήση του τελεστή | σε σχέση με τον ||. Για παράδειγμα στην έκφραση

x>0 | someFunc(x)>10

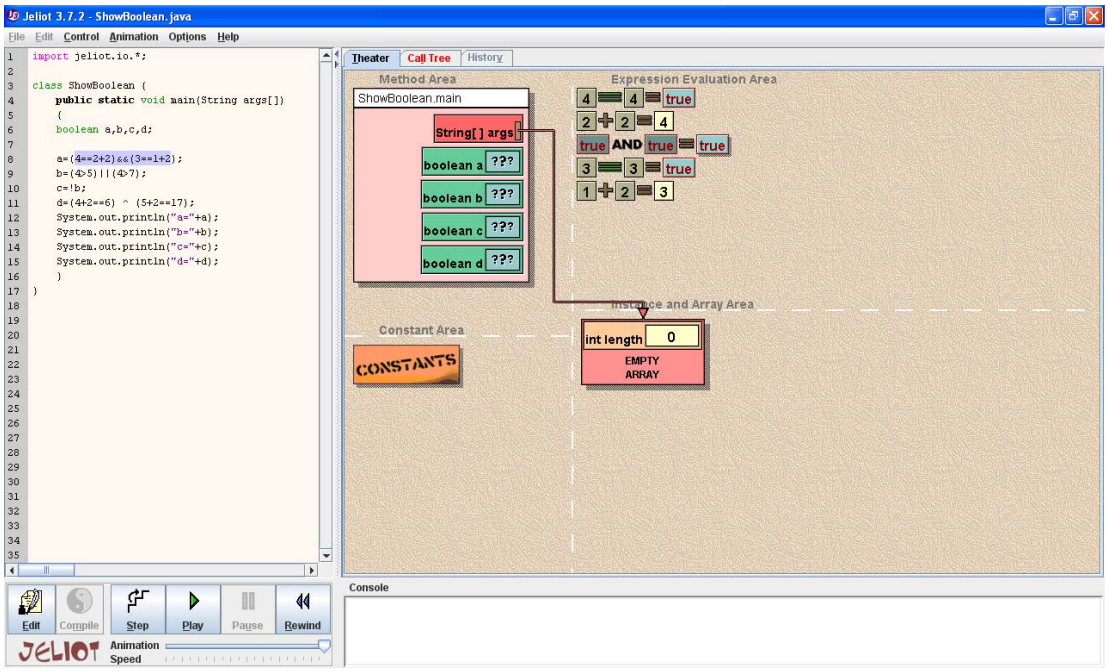
θα κληθεί η someFunc ακόμα και αν το x είναι θετικό

Ένα παράδειγμα στην εφαρμογή Jeliot



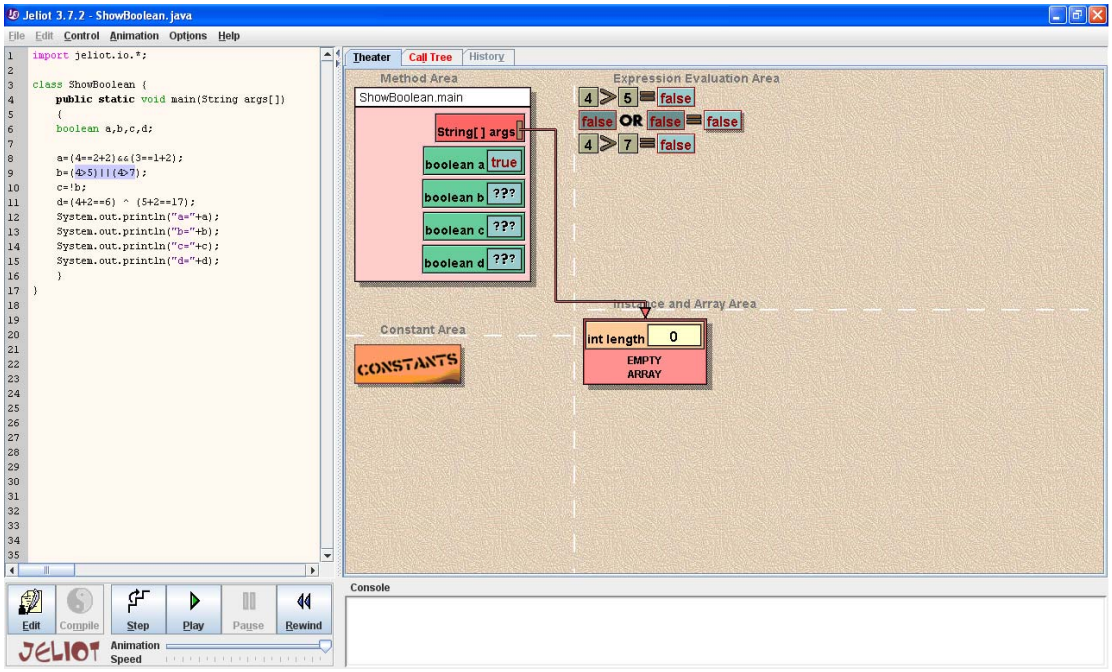
Σχήμα 5.3α: Παράδειγμα χρήσης λογικών τελεστών

Στο σχήμα 5.3α δηλώνονται οι μεταβλητές a, b, c, d σαν boolean.



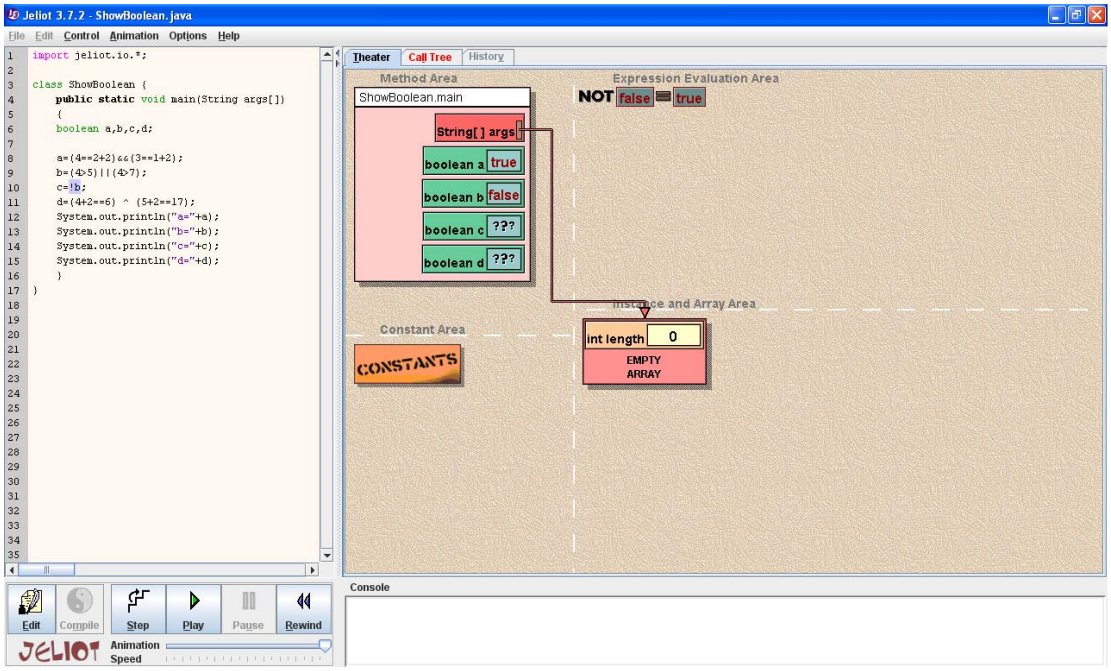
Σχήμα 5.3β: Παράδειγμα χρήσης λογικών τελεστών

Στο σχήμα 5.3β γίνεται έλεγχος της εντολής a=(4==2+2)&&(3==1+2); δηλαδή αν το (4=4 = true) και (3=3 = true). Το αποτέλεσμα είναι (true AND true) = true



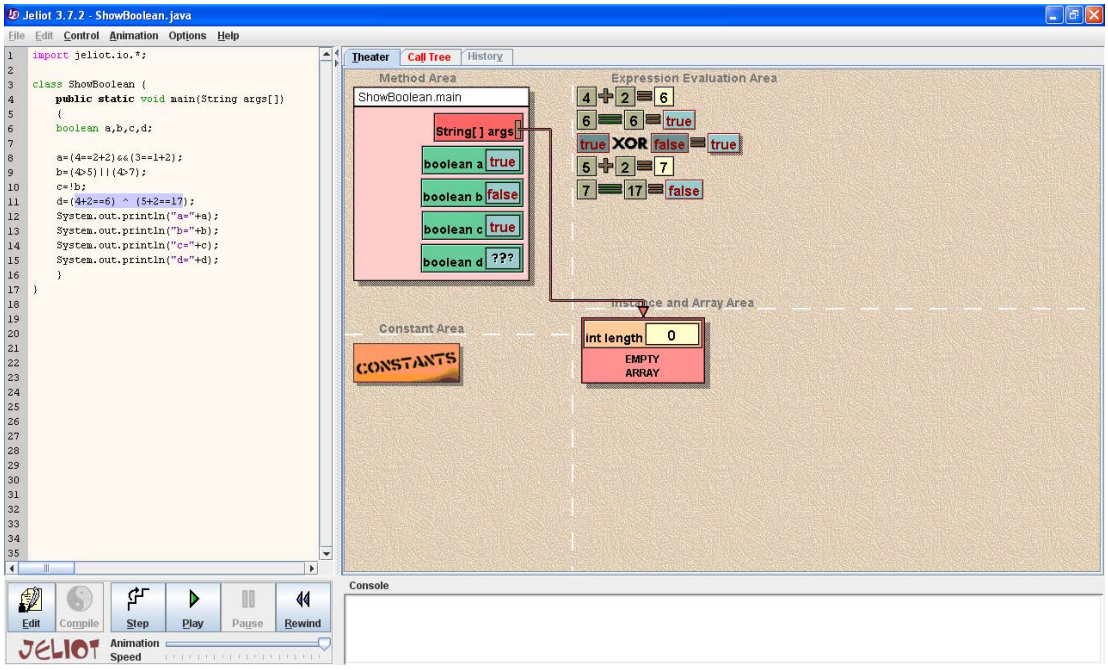
Σχήμα 5.3γ: Παράδειγμα χρήσης λογικών τελεστών

Στο σχήμα 5.3γ γίνεται έλεγχος της εντολής b=(4>5)||(4>7); δηλαδή αν το (4>5 = false) η το (4>7=false). Το αποτέλεσμα είναι (false OR false) = false



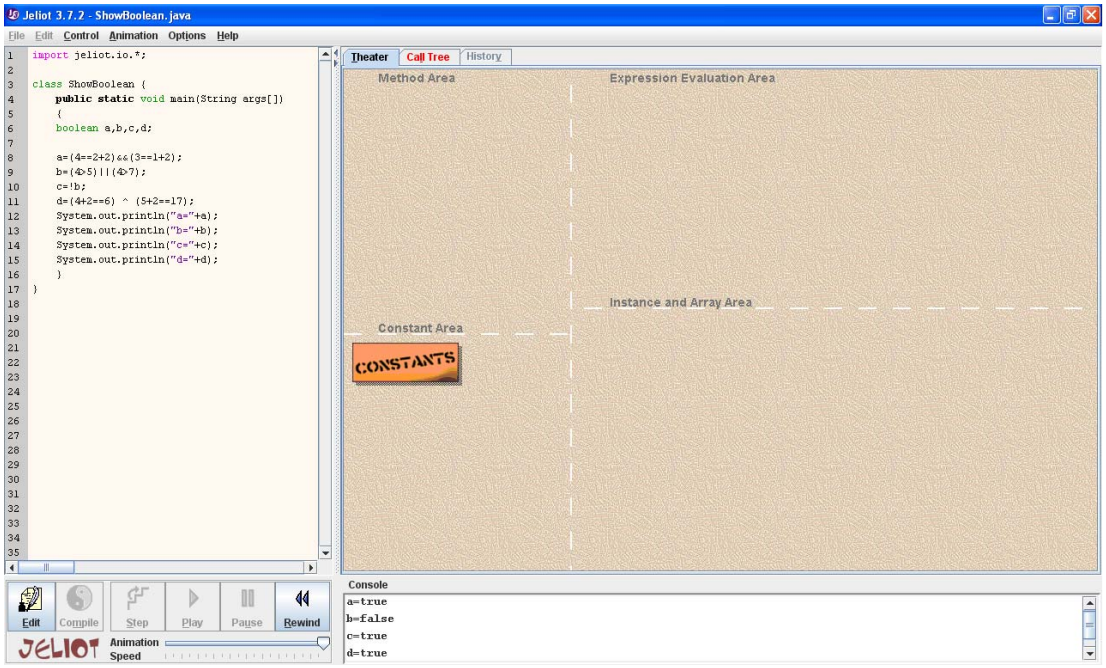
Σχήμα 5.3δ: Παράδειγμα χρήσης λογικών τελεστών

Στο σχήμα 5.3δ γίνεται έλεγχος της εντολής c=!b; ο τελεστής ! αντιστρέφει μια λογική πρόταση. Στην περίπτωση μας θα πάρουμε το αποτέλεσμα της b=false ανεστραμμένο δηλαδή c=true.



Σχήμα 5.3ε: Παράδειγμα χρήσης λογικών τελεστών

Στο σχήμα 5.3ε γίνεται έλεγχος της εντολής d=(4+2==6) ^ (5+2==17). Ο τελεστής XOR συνδέει 2 λογικές προτάσεις και χρησιμοποιείται για να εξακριβώσει εάν η μία και μόνον η μία από της δύο προτάσεις είναι αληθής. Δηλαδή αν το (6=6 = true) και (7=17 = false). Το αποτέλεσμα είναι (true XOR false) = true.



Σχήμα 5.3ζ: Παράδειγμα χρήσης λογικών τελεστών

Στο σχήμα 5.3ζ έχουμε το αποτέλεσμα του προγράμματος στην console της εφαρμογής Jeliot με τα ακόλουθα : a=true, b=false, c=true και d=true

Κώδικας προγράμματος :

```
class ShowBoolean {
   public static void main(String args[])
   {
   boolean a,b,c,d;
   a=(4==2+2)&&(3==1+2);
   b=(4>5)||(4>7);
   c=!b;
   d=(4+2==6) ^ (5+2==17);
   System.out.println("a="+a);
   System.out.println("b="+b);
   System.out.println("c="+c);
   System.out.println("d="+d);
   }
}
```

5.3.5 Δυαδικοί τελεστές

Με τους τελεστές αυτούς μπορούμε και κάνουμε πράξεις σε επίπεδο bit.

Τελεστής	Χρήση	Περιγραφή
>>	n1 >> n2	Κάνει ολίσθηση n2 bits δεξιά στο n1
<<	n1<<n2	Κάνει ολίσθηση n2 bits αριστερά στο n1
>>>	n1 >>> n2	Το ίδιο με τον >> αλλά χωρίς πρόσημο
&	n1&n2	Δυαδικό ΚΑΙ των n1 και n2
|	n1 | n2	Δυαδικό Ή των n1 και n2
^	n1 ^ n2	Αποκλειστικό Ή (XOR) των n1 και n2
~	~n	Δυαδικό συμπλήρωμα του n

Πίνακας 5.5: Δυαδικοί τελεστές

Οι τελεστές δυαδικών πράξεων χρησιμοποιούνται όταν θέλουμε να επέμβουμε στην τιμή μιας μεταβλητής σε επίπεδο bit.

Για παράδειγμα έστω ότι έχουμε δηλώσει τις ακόλουθες σταθερές:

final int TURBO = 1;

final int TWO_DOORS = 2;

final int CHEAP = 4;

Που έστω ότι αφορούν τα χαρακτηριστικά ενός αυτοκινήτου. Θα μπορούσαμε στη συνέχεια να δηλώσουμε μία μεταβλητή int έστω την int characteristics = 0 και στη συνέχεια να αποδώσουμε σε αυτή τη μεταβλητή ταυτόχρονα δύο ή και τρεις ιδιότητες χρησιμοποιώντας τον τελεστή | όπως παρακάτω:

characteristics = TURBO | TWO_DOORS;

5.3.6 Τελεστές καταχώρισης

Ο τελεστής καταχώρισης τιμής σε μεταβλητή είναι ο =. Ο = αποδίδει την τιμή της έκφρασης στα δεξιά του στη μεταβλητή που βρίσκεται στα αριστερά του, όπως στην εντολή:

x = y+10;

Όπου θα υπολογιστεί η έκφραση y+10 και στη συνέχεια θα καταχωρηθεί η τιμή αυτή στη μεταβλητή x.

Στο Jeliot χρησιμοποιούνται και άλλοι τελεστές απόδοσης τιμής όπου είναι ουσιαστικά συντομεύσεις για απόδοση τιμής σε μεταβλητή με ταυτόχρονη αύξηση, μείωση κτλ αυτής της μεταβλητής.

Για παράδειγμα η έκφραση με τον τελεστή +=:

x += 10;

είναι ταυτόσημη με την έκφραση: x = x+10;

Αντίστοιχη σημασία έχουν και οι τελεστές: -= *= /= %= &= |= ^= <<= >>= >>>=

5.4 Δομές ελέγχου ροής προγράμματος

Σ' αυτό το τμήμα θα ασχοληθούμε με τις δομές ροής ελέγχου προγράμματος Jeliot, όπως η if και η while.

Υπάρχουν γενικά δύο ήδη δομών ελέγχου ροής (control flow):

- Οι δομές επιλογής και
- Οι δομές επανάληψης

Ο ακόλουθος πίνακας 5.6 συνοψίζει αυτές τις δομές

Είδος δομής ελέγχου ροής	**Δομή ελέγχου ροής**
Δομές Επιλογής	if-else
	switch-case
Δομές Επανάληψης	for
	while
	do-while

Πίνακας 5.6: Δομές ελέγχου ροής στο Jeliot

Επίσης κάποιες άλλες εντολές πέρα από τις ίδιες τις δομές ελέγχου ροής που είδαμε προηγουμένως είναι οι: break, continue και return που μεταφέρουν το έλεγχο ροής σε άλλα σημεία του προγράμματος. Την χρήση αυτών των εντολών θα τη δούμε στη συνέχεια.

Το Jeliot επιτρέπει και τη χρήση ετικετών αλλά δεν υποστηρίζει την εντολή goto. Αντί αυτής μπορούν και με τις ετικέτες να χρησιμοποιηθούν οι εντολές break και continue.

5.4.1 Η δομή επιλογής if-else

Η εντολή if μας βοηθάει στον έλεγχο μίας λογικής έκφρασης (της συνθήκης) και αν είναι αληθής εκτελούνται μία ή περισσότερες εντολές ενώ αν είναι ψευδής δεν εκτελούνται. Αν οι εντολές είναι περισσότερες από μία τότε θα πρέπει οι εντολές να περικλεισθούν ανάμεσα από άγκιστρα (τα οποία ομαδοποιούν εντολές).

Το συντακτικό της εντολής if είναι το ακόλουθο

if (συνθήκη)

εντολές

π.χ.

```
if (x!=0) {
        System.out.println("Το x δεν είναι 0");
        y = 1/x;
}
```

Η εντολή if έχει επίσης και τη φράση else με την οποία επιτρέπεται η εκτέλεση μίας ομάδας εντολών εναλλακτικά αν δεν ισχύει η συνθήκη. Φυσικά μπορούμε να έχουμε και if μέσα σε άλλα if όσες φορές θέλουμε.

Το συντακτικό της if επαυξημένο με τη φράση else είναι το ακόλουθο:

if (συνθήκη)

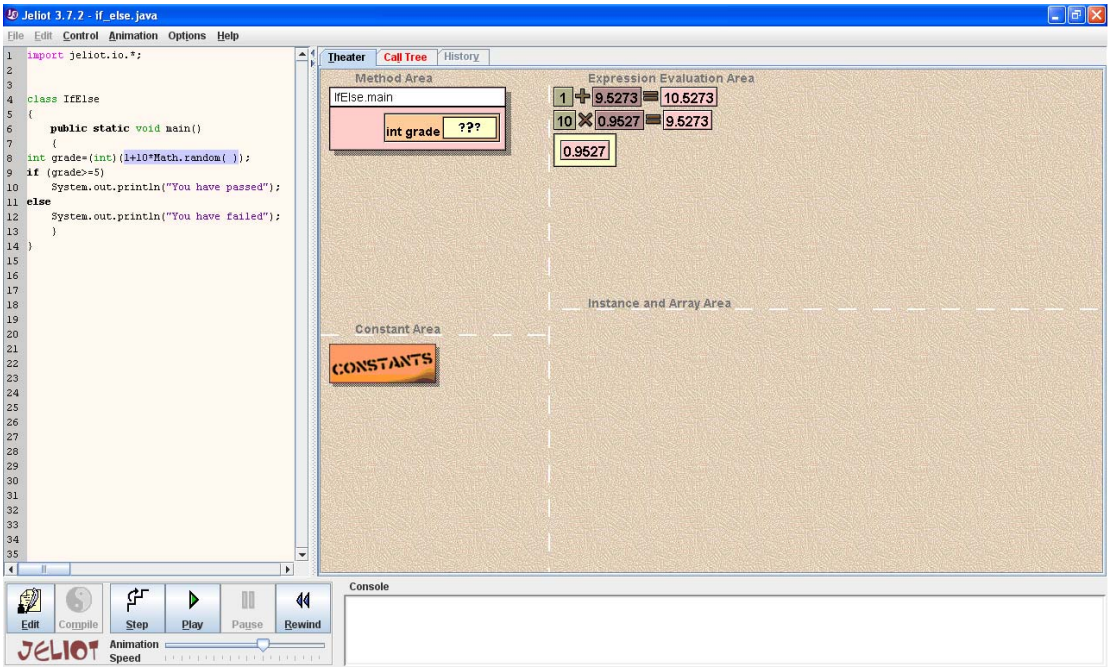
ομάδα-εντολών-1

else

ομάδα-εντολών-2

Ένα παράδειγμα στην εφαρμογή Jeliot

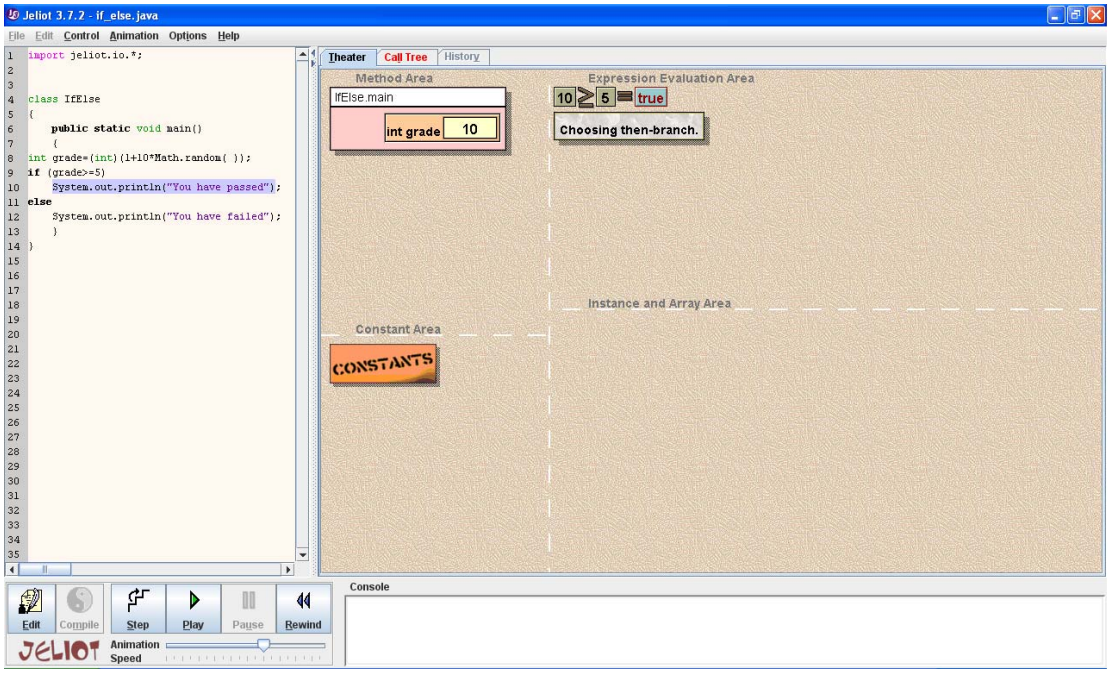
Έστω ότι θέλουμε μια τυχαία εμφάνιση ενός αριθμού από το 1 έως το 10. Για τους αριθμούς που θα είναι μεγαλύτεροι ή ίσοι με 5 να εμφανίζεται το μήνυμα 'You have passed' ενώ για τους αριθμούς που θα είναι μικρότεροι από το 5 το μήνυμα 'You have failled'.



Σχήμα 5.4α: Παράδειγμα χρήσης If - Else

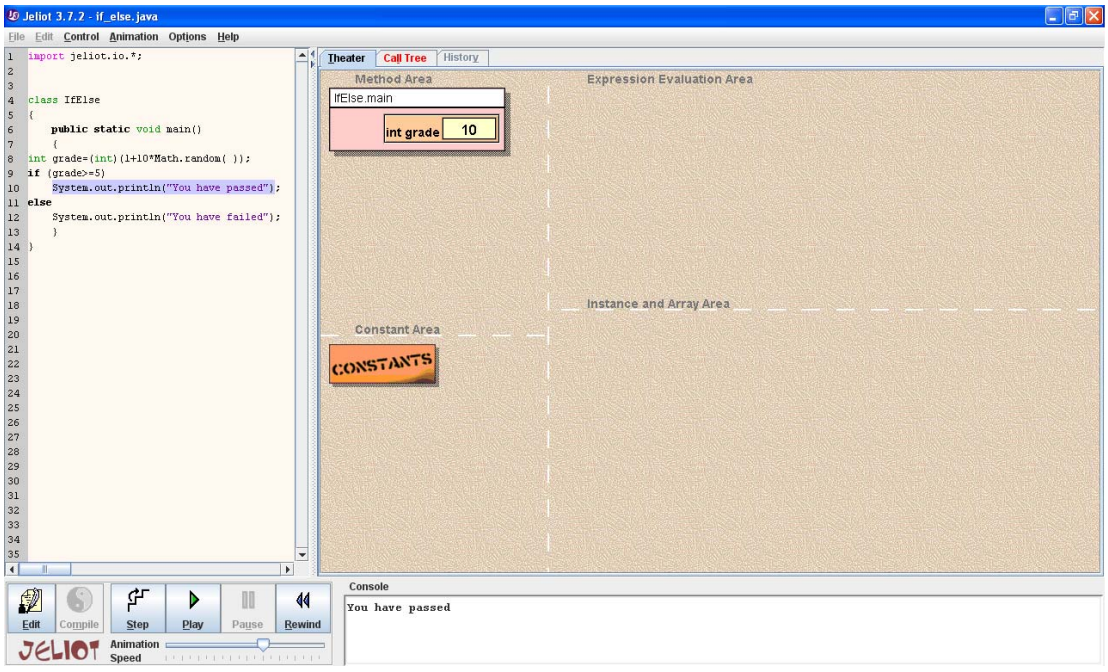
Στο σχήμα 5.4α δηλώνεται η μεταβλητή grade.

int grade=(int)(1+10*Math.random( )). Πολλαπλασιάζετε το (10*(τον τυχαίο αριθμό) στην περίπτωση μας είναι ο αριθμός 0.9527) και μας δίνει το αποτέλεσμα 9.5273. Στην συνέχεια έχουμε την πρόσθεση του 1+9.5273=10.5273. Το δεκαδικό μέρος του grade είναι 10.



Σχήμα 5.4β: Παράδειγμα χρήσης If - Else

Στο σχήμα 5.4β στην εντολή if (grade>=5) ο έλεγχος 10>=5 βγάζει αποτέλεσμα true επομένως θα εκτελεστούν οι εντολές της if. Αν η τιμή του grade ήταν μικρότεροι η ίσοι με 4 τότε θα είχαμε την εκτέλεση της εντολής else.



Σχήμα 5.4γ: Παράδειγμα χρήσης If - Else

Στο σχήμα 5.4γ έχουμε το αποτέλεσμα του προγράμματος στην console της εφαρμογής Jeliot με τα ακόλουθα : You have passed

Κώδικας προγράμματος :

```
class IfElse
{
   public static void main()
   {
int grade=(int)(1+10*Math.random( ));
if (grade>=5)
   System.out.println("You have passed");
else
   System.out.println("You have failed");
   }
}
```

5.4.2 Η δομή επιλογής switch-case

Όταν οι εναλλακτικές περιπτώσεις που πρέπει να ελέγξουμε με την if είναι πάρα πολλές και αφορούν τον έλεγχο για ισότητα της τιμής μιας μεταβλητής ή μιας έκφρασης με κάποιες τιμές προτιμάται η switch-case η οποία έχει την ακόλουθη γενική μορφή:

```
switch (έκφραση)
{
    case τιμή-1:
εντολές-1; [break;]
          …
    case τιμή-ν:
εντολές-ν; [break;]
    [default:
εντολές; [break;]]
}
```

Η switch-case αποτιμά την τιμή της έκφρασης που ελέγχεται και στη συνέχεια διατρέχει με τη σειρά όλες τις περιπτώσεις που δίνονται. Αν κάποια περίπτωση βρεθεί αληθής τότε εκτελούνται οι εντολές που δίνονται μετά την άνω-κάτω τελεία

γι’ αυτή τη περίπτωση. Αν καμία περίπτωση δεν βρεθεί αληθής τότε εκτελείται η default περίπτωση αν υπάρχει.

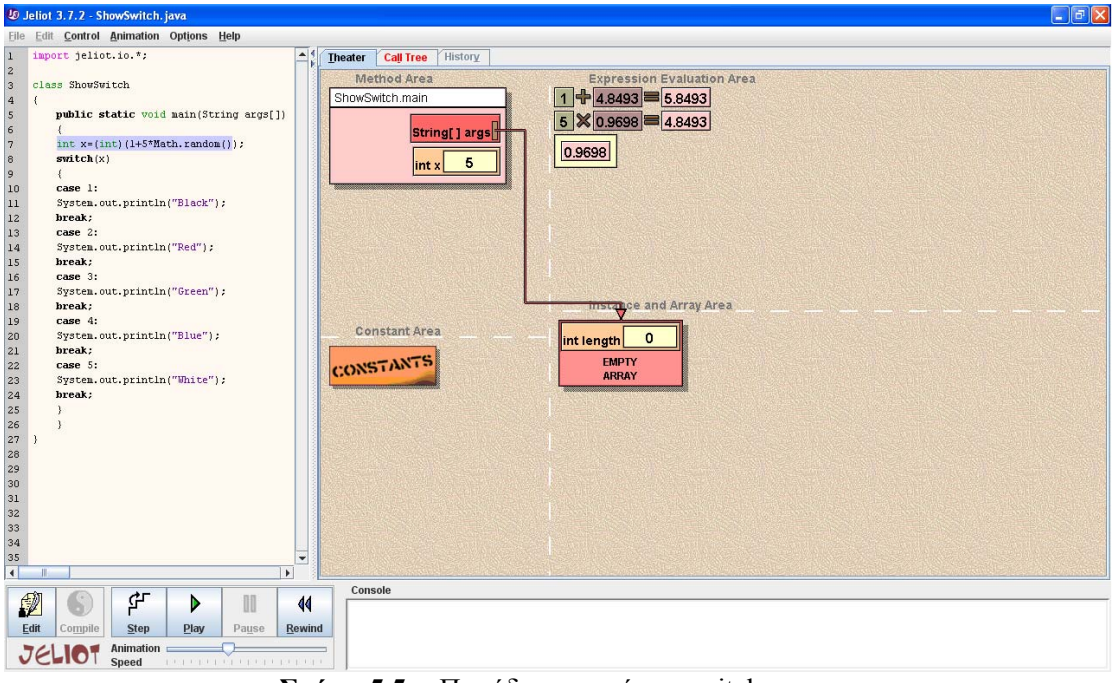
Προσοχή χρειάζεται στη χρήση του break που είναι μεν προαιρετική αλλά απαιτείται τις περισσότερες φορές, μια και αν δεν υπάρχει τότε θα εκτελεστούν και οι εντολές τις επόμενης περίπτωσης (χωρίς να ελεγχθεί η τιμή της) και πιθανώς και άλλων, μέχρι να βρεθεί το επόμενο break ή να τελειώσει η switch-case. Παρόλα αυτά ενδέχεται να υπάρχουν κάποιες περιπτώσεις που αυτό θα ήταν βολικό, όπως δείχνει το ακόλουθο τμήμα κώδικα που υπολογίζει τις μέρες ενός μήνα ανάλογα με το ποιος μήνας είναι (η τιμή της μεταβλητής month) και το αν το έτος (year) είναι δίσεκτο:

```
switch (month) {
   case 1:
   case 3:
   case 5:
   case 7:
   case 8:
   case 10:
   case 12:
      numDays = 31;
      break;
   case 4:
   case 6:
   case 9:
   case 11:
      numDays = 30;
      break;
   case 2:
```

```
        if ( ((year % 4 == 0) && !(year % 100 == 0))
                || (year % 400 == 0) )
            numDays = 29;
        else
            numDays = 28;
        break;
}
```

Ένα παράδειγμα στην εφαρμογή Jeliot

Έστω ότι θέλουμε μια τυχαία εμφάνιση ενός αριθμού από το 1 έως το 5. Για τις τιμές 1 και 5 θέλουμε να τυπώνονται στην οθόνη τα χρώματα μαύρο και άσπρο αντίστοιχα, ενώ για τις τιμές 2, 3, 4, να τυπώνονται τα τρία βασικά χρώματα (κόκκινο, πράσινο, μπλε).

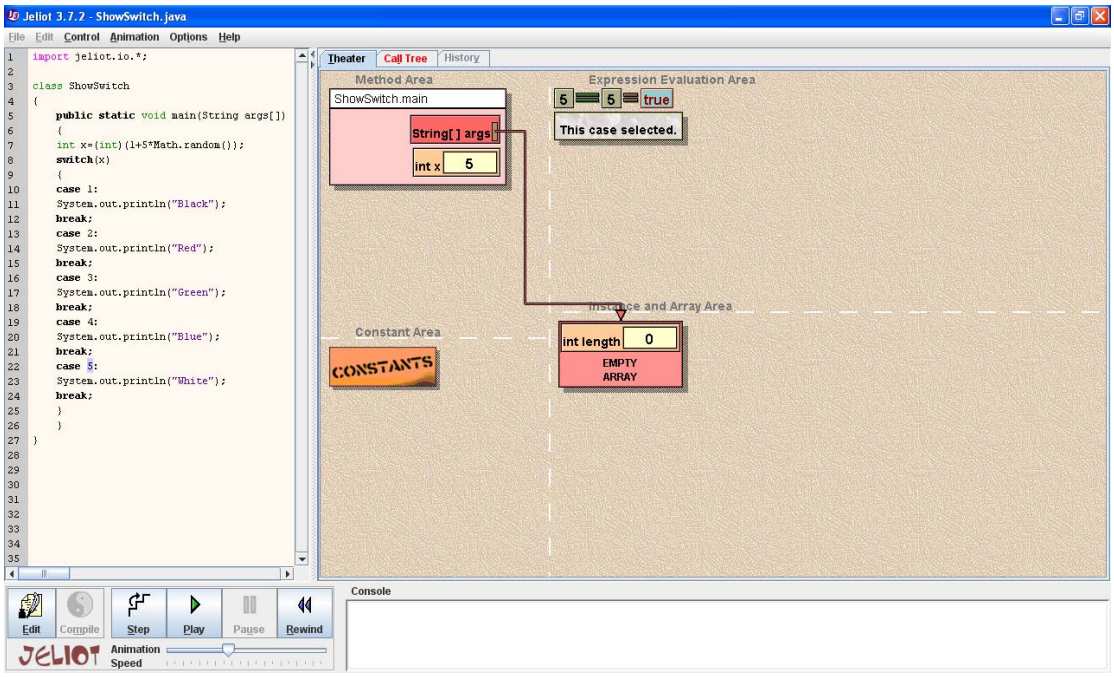


Σχήμα 5.5α: Παράδειγμα χρήσης switch - case

Στο σχήμα 5.5α δηλώνεται η μεταβλητή x.

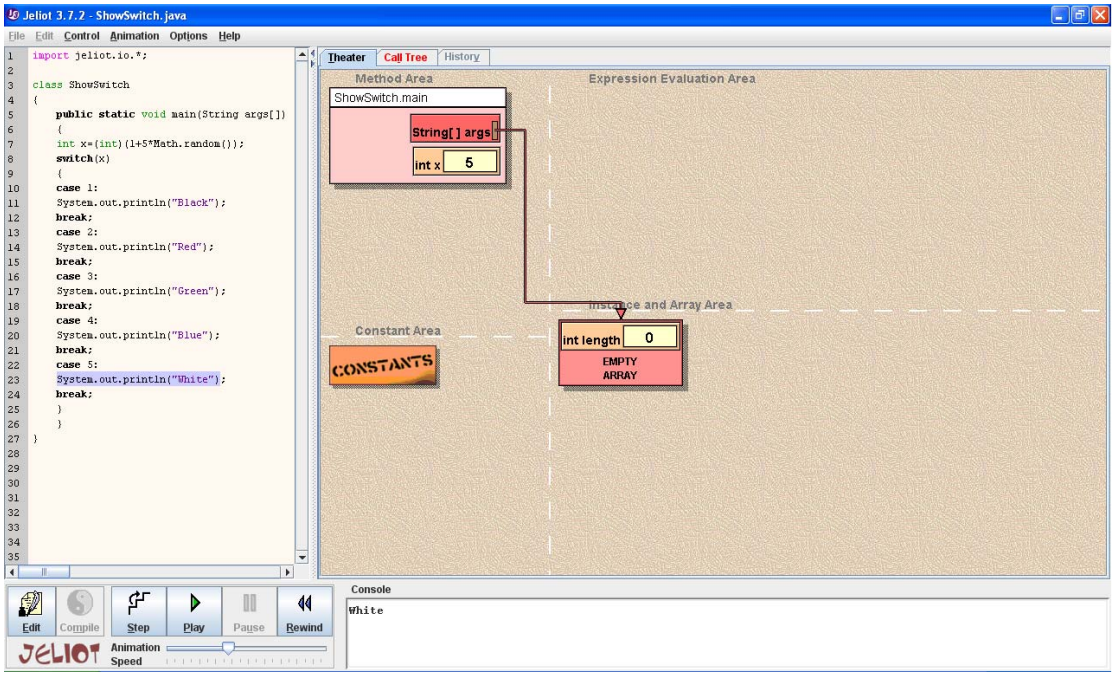
int x=(int)(1+5*Math.random( )). Πολλαπλασιάζετε το (5*(τον τυχαίο αριθμό) στην περίπτωση μας είναι ο αριθμός 0.9698) και μας δίνει το αποτέλεσμα 4.8493. Στην

συνέχεια έχουμε την πρόσθεση του 1+4.8493=5.8493. Το δεκαδικό μέρος του x είναι 5.



Σχήμα 5.5β: Παράδειγμα χρήσης switch - case

Στο σχήμα 5.5β έχουμε είσοδο στην εντολή switch όπου ο έλεγχος 5=5 βγάζει αποτέλεσμα true.



Σχήμα 5.5γ: Παράδειγμα χρήσης switch - case

Στο σχήμα 5.5γ έχουμε το αποτέλεσμα του προγράμματος στην console της εφαρμογής Jeliot με τα ακόλουθα : White

Κώδικας προγράμματος :

```
class ShowSwitch
{
   public static void main(String args[])
   {
   int x=(int)(1+5*Math.random());
   switch(x)
   {
   case 1:
   System.out.println("Black");
   break;
   case 2:
   System.out.println("Red");
   break;
   case 3:
   System.out.println("Green");
   break;
   case 4:
   System.out.println("Blue");
   break;
   case 5:
   System.out.println("White");
   break;
   }
   }
}
```

5.4.3 Η δομή επανάληψης for

Η εντολή for είναι χρήσιμη για την επανάληψη μιας σειράς εντολών όταν είναι γνωστό εκ των προτέρων πόσες φορές θέλουμε να επαναληφθούν. Έτσι συνήθως το for ελέγχεται από ένα μετρητή ο οποίος μεταβάλλεται από μία αρχική τιμή μέχρι να ξεπεράσει μία τελική τιμή. Στη for επίσης καθορίζεται το πόσο θα μεταβάλλεται αυτός ο μετρητής σε κάθε βήμα.

Η γενική μορφή της for είναι η ακόλουθη:

for (αρχικοποίηση; τερματισμός; αύξηση)

εντολές;

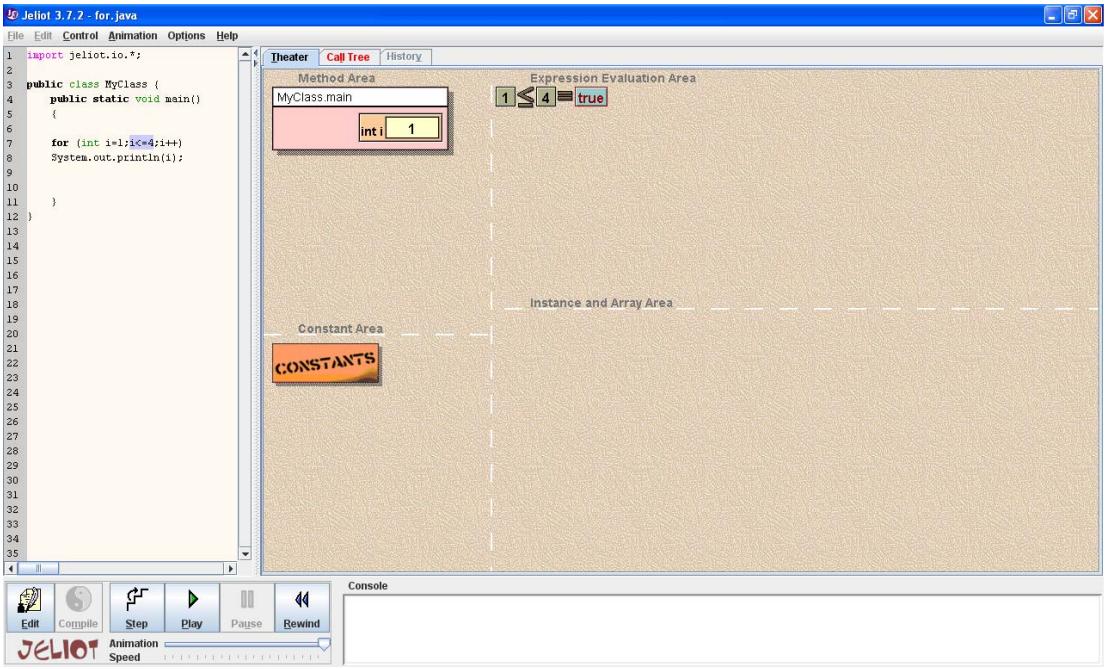
Για παράδειγμα το ακόλουθο τμήμα κώδικα εμφανίζει στην οθόνη τους αριθμούς από το 1 μέχρι το 10:

```
for (int i=1; i<=10; i++)
        System.out.println(i);
```

Στη φράση της αρχικοποίησης το i δηλώνεται (η εμβέλειά του είναι το for loop) και αρχικοποιείται στο1. Στη φράση του τερματισμού το i ελέγχεται για το αν έχει ξεπεράσει το 10. Άρα ο συγκεκριμένος βρόχος θα επαναληφθεί μέχρι το i να ξεπεράσει το 10. Στη φράση της μεταβολής το i αυξάνεται κατά 1. Αυτό σημαίνει ότι το i θα πάρει διαδοχικά τις τιμές 1, 2, …, 10

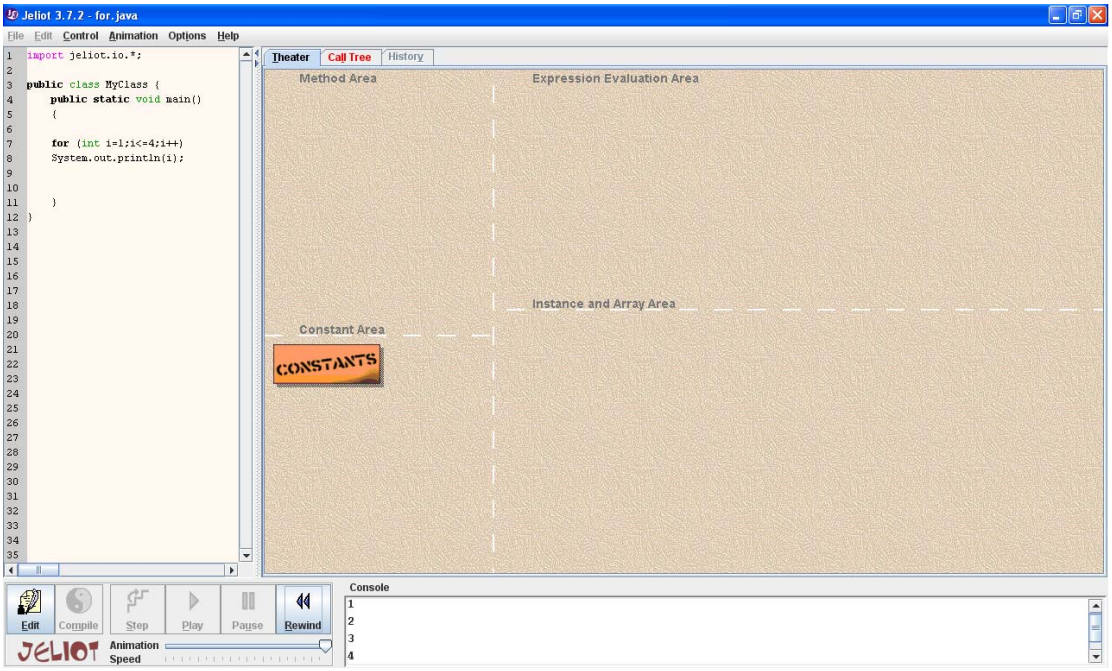
Σε κάθε βήμα της επανάληψης ελέγχεται η τιμή του i. Αν το i ξεπεράσει το 10 ο βρόχος τερματίζεται, αν όχι αυξάνεται κατά 1 και εκτελείται ξανά η μία και μοναδική εντολή αυτού του βρόχου.

Ένα παράδειγμα στην εφαρμογή Jeliot



Σχήμα 5.6α: Παράδειγμα χρήσης for

Στο σχήμα 5.6α έχουμε τον έλεγχο της συνθήκης i<=4 και για όσες φορές ισχύει αυτό θα έχουμε την εκτύπωση του αποτελέσματος του i στην console του Jeliot.



Σχήμα 5.6β: Παράδειγμα χρήσης for

Στο σχήμα 5.6β έχουμε το αποτέλεσμα του προγράμματος στην console της εφαρμογής Jeliot με τα ακόλουθα : 1, 2, 3, 4

Κώδικας προγράμματος :

```
public class MyClass {
   public static void main()
   {
   for (int i=1;i<=4;i++)
   System.out.println(i);
   }
}
```

5.4.4 Η δομή επανάληψης while

Η εντολή for που είδαμε προηγουμένως είναι κατάλληλη όταν γνωρίζουμε πόσες επαναλήψεις θα γίνουν. Αν δεν γνωρίζουμε πόσες επαναλήψεις θα γίνουν μία καλύτερη εντολή είναι η while.

Η while έχει την ακόλουθη γενική μορφή:

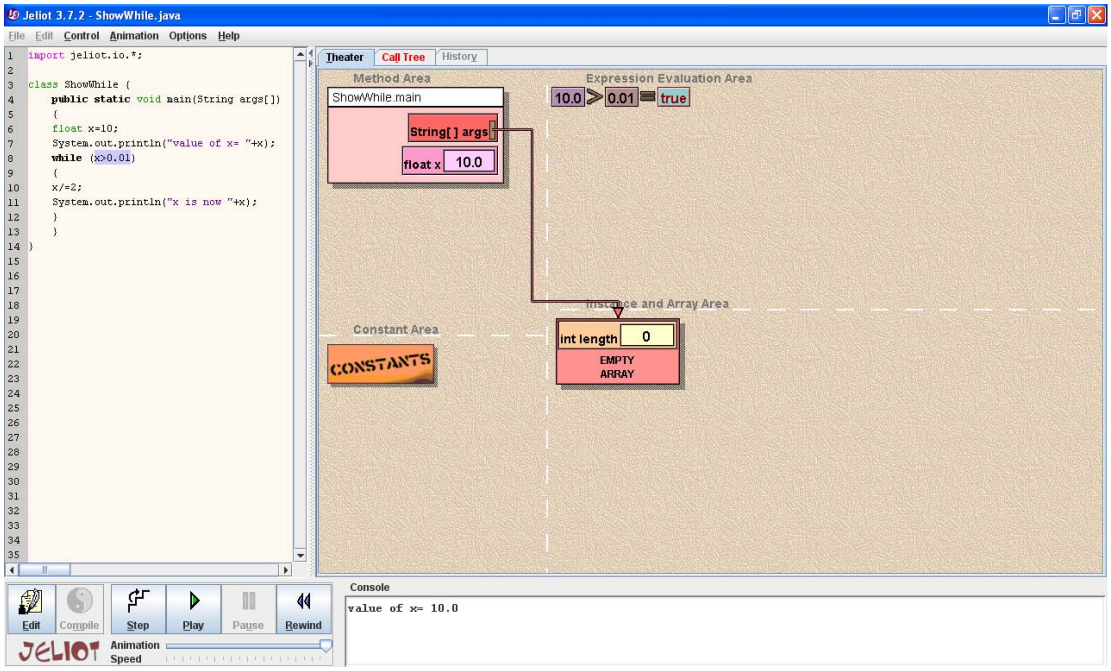
while (συνθήκη)

εντολές;

Η εντολή ή εντολές που ακολουθούν το while θα εκτελεστούν όσο η συνθήκη είναι αληθής. Ο βρόχος δηλαδή θα τερματιστεί όταν η συνθήκη – που είναι μία boolean έκφραση – γίνει ψευδής

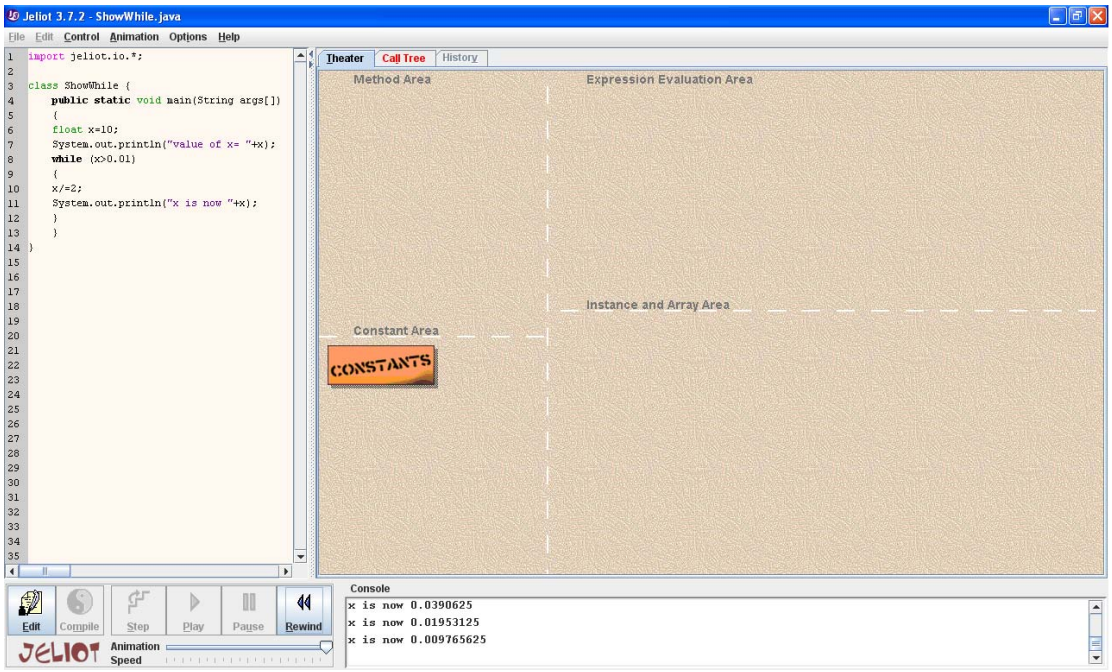
Ένα παράδειγμα στην εφαρμογή Jeliot

Να δημιουργηθεί ένα πρόγραμμα το οποίο να δίνει σε μία μεταβλητή x τύπου float την τιμή 10.0. Στη συνέχεια, η μεταβλητή αυτή να υποδιπλασιάζεται μέχρι να γίνει μικρότερη από την τιμή 0.01.



Σχήμα 5.7α: Παράδειγμα χρήσης while

Στο σχήμα 5.7α οι εντολές που ακολουθούν την while (x>=10) η οποία θα εκτελεστούν για όσες φόρες βγει η συνθήκη αληθής.



Σχήμα 5.7β: Παράδειγμα χρήσης while

Στο σχήμα 5.7β έχουμε το αποτέλεσμα του προγράμματος στην console της εφαρμογής Jeliot με τα ακόλουθα :

value of x= 10.0
x is now 5.0
x is now 2.5
x is now 1.25
x is now 0.625
x is now 0.3125
x is now 0.15625
x is now 0.078125
x is now 0.0390625
x is now 0.01953125
x is now 0.009765625

Κώδικας προγράμματος :

```
class ShowWhile {
   public static void main(String args[])
   {
   float x=10;
   System.out.println("value of x= "+x);
   while (x>0.01)
   {
   x/=2;
   System.out.println("x is now "+x);
   }
   }
}
```

5.4.5 Η δομή επανάληψης do-while

Υπάρχουν κάποιες περιπτώσεις στις οποίες θα θέλαμε οι εντολές μέσα στο βρόχο να εκτελεστούν τουλάχιστον μία φορά και στη συνέχεια να ελεγχθεί η συνθήκη εξόδου. Σ' αυτές τις περιπτώσεις προτιμάται η χρήση της do-while αντί της while.

Η do-while έχει την ακόλουθη γενική μορφή:

do

εντολές;

while (συνθήκη);

Στην do-while πρώτα εκτελούνται οι εντολές και στη συνέχεια ελέγχεται η συνθήκη. Ο βρόχος τερματίζεται αν η συνθήκη βρεθεί ψευδής. Η do-while δεν χρησιμοποιείται πολύ συχνά αλλά έχει κι αυτή τις χρήσεις της. Για παράδειγμα όταν διαβάζουμε χαρακτήρες από ένα αρχείο μέχρι να διαπιστώσουμε το τέλος του αρχείου θα πρέπει να διαβάσουμε τουλάχιστον ένα χαρακτήρα, όπως δείχνει και το ακόλουθο τμήμα κώδικα:

```
int c;

Reader in;

...

do {

   c = in.read();

   ...

} while (c != 1);

...
```

5.5 Εντολές διακλάδωσης

Το Jeliot έχει τρεις εντολές διακλάδωσης οι οποίες είναι βολικές σε αρκετές περιπτώσεις και ιδιαίτερα με τις επαναληπτικές δομές που έχουμε συζητήσει.

Οι εντολές διακλάδωσης είναι οι: break, continue και return

Την εντολή break ήδη την είδαμε σε μία χρήση της με την switch-case. Μία ακόμα αρκετά συνήθη χρήση της break είναι η χρήση της για άμεση έξοδο από κάποιο βρόχο.

Η χρήση της continue μέσα σε ένα βρόχο προκαλεί την άμεση αποτίμηση και πάλι της συνθήκης εξόδου. Για να γίνει αυτό στις εντολές for και while ο έλεγχος μεταφέρεται στη πρώτη γραμμή του βρόχου ενώ στο do-while στη τελευταία. Στη συνέχεια αποτιμάται και πάλι η συνθήκη τερματισμού και η επαναληπτική διαδικασία τερματίζεται ή συνεχίζεται ανάλογα με την τιμή της συνθήκης (true ή false) ως συνήθως.

Οι εντολές break και continue έχουν και μία μορφή για ετικέτες (labeled break και labeled continue) που δεν θα συζητήσουμε εδώ μια και στη πράξη οι ετικέτες δεν χρησιμοποιούνται αφού οι υπόλοιπες δομές που είδαμε αρκούν για όλες τις πιθανές χρήσεις, χωρίς να εμφανίζουν πολλά από τα προβλήματα που πιθανώς να εμφανιστούν με την χρήση των ετικετών.

5.6 Πίνακες

Σ' αυτό το μάθημα θα ασχοληθούμε με τους πίνακες ομοειδών στοιχείων (π.χ. ακεραίων) στο Jeliot.

5.6.1 Δήλωση και δημιουργία πινάκων

Οι πίνακες, όπως και στις άλλες γλώσσες προγραμματισμού, είναι δομές δεδομένων για την συλλογή και διαχείριση δεδομένων του ιδίου τύπου. Η διαφορά με τις περισσότερες από τις άλλες γλώσσες είναι ότι οι πίνακες στο Jeliot είναι αντικείμενα και έχουν και ιδιότητες όπως η length που μας δείχνει ποιο είναι το μέγεθος ενός πίνακα.

Η δήλωση ενός πίνακα γίνεται ως εξής:

τύπος[] όνομα;

όπως για παράδειγμα στην ακόλουθη δήλωση:

int[] arrayOfInts;

που δηλώνει ένα πίνακα ακεραίων με το όνομα arrayOfInts.

Η δήλωση του πίνακα δεν αρκεί για τη χρησιμοποίησή του. Γι' αυτό θα πρέπει να δεσμευτεί ο χώρος μνήμης που απαιτείται για το πίνακα. Επειδή ο πίνακας είναι αντικείμενο απαιτείται να δημιουργήσουμε ένα αντικείμενο (instantiation) για τον πίνακα με την εντολή new.

Έτσι συνήθως η δήλωση ενός πίνακα συνδυάζεται με την δημιουργία του αντικειμένου με την new, στην οποία προσδιορίζεται και το μέγεθος του πίνακα.

Αυτό γίνεται ως εξής:

τύπος[] όνομα = new τύπος[μέγεθος];

όπως για παράδειγμα στην ακόλουθη δήλωση:

int[] arrayOfInts = new int[10];

που δημιουργεί ένα πίνακα 10 ακεραίων με το όνομα arrayOfInts.

Η δημιουργία του πίνακα δεν είναι απαραίτητο να γίνει ταυτόχρονα με την δήλωσή του μπορεί να γίνει και αργότερα, όπως δείχνει το ακόλουθο τμήμα κώδικα:

```
int[] arrayOfInts;

...

arrayOfInts = new int[10];

...
```

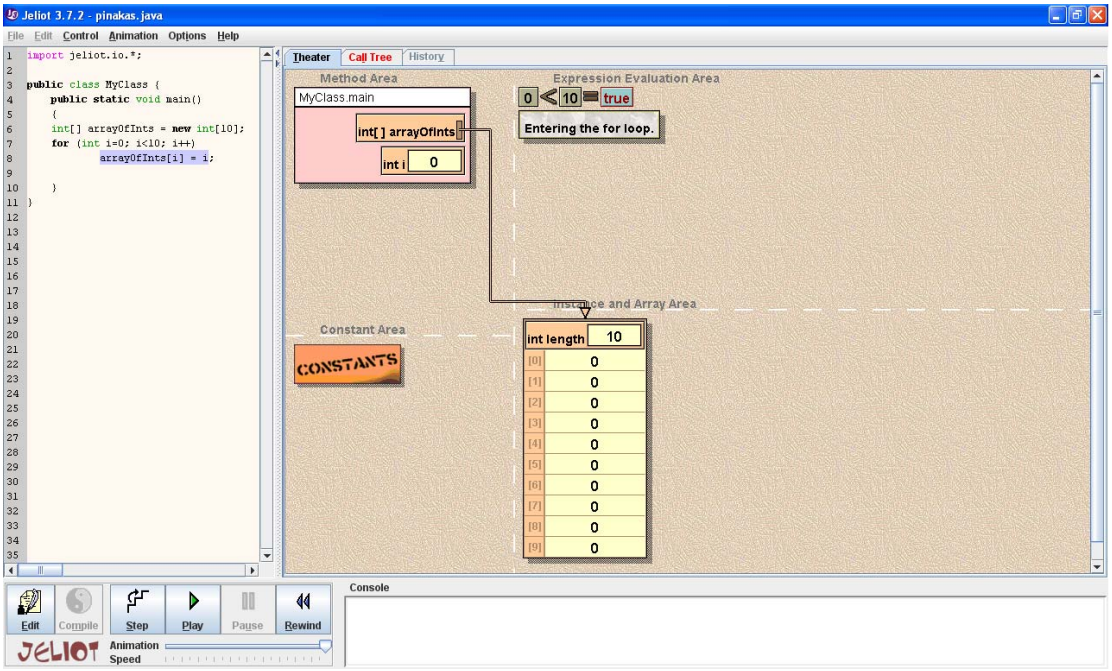
Σε κάθε περίπτωση πάντως δεν επιτρέπεται η αναφορά σε στοιχεία του πίνακα πριν να δημιουργηθεί ο πίνακας με την εντολή new.

5.6.2 Χρήση πινάκων

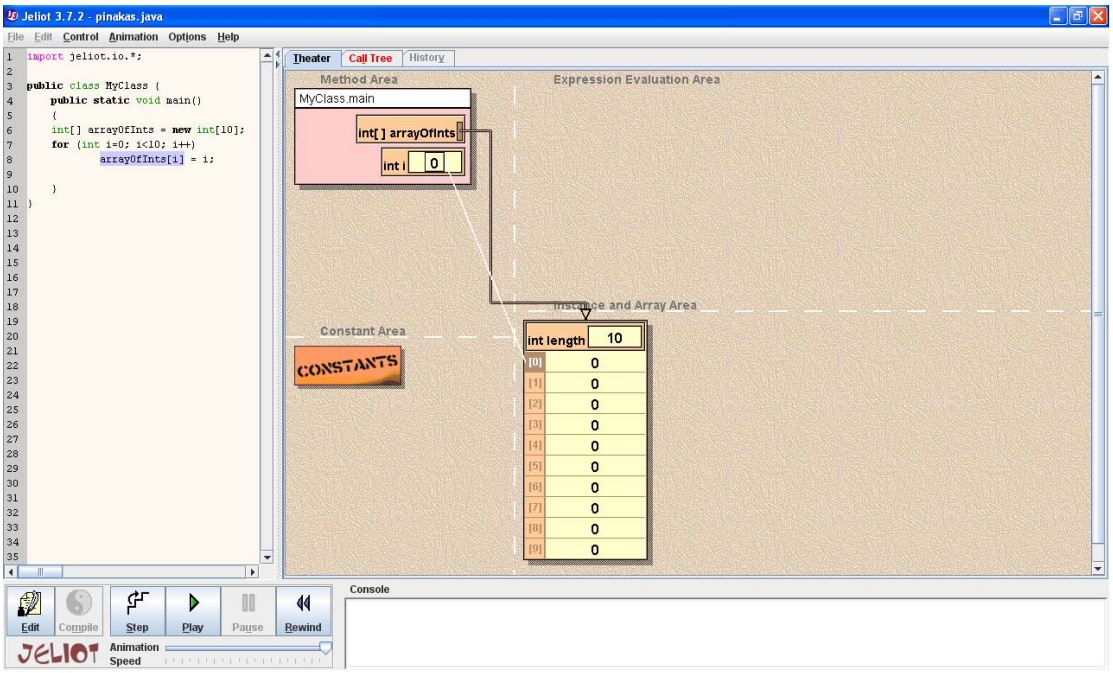
Το ακόλουθο τμήμα κώδικα δημιουργεί ένα πίνακα 10 ακεραίων και στη συνέχεια τοποθετεί μέσα σ' αυτόν τους ακεραίους από το 0 μέχρι το 9.

```
int[] arrayOfInts = new int[10];

for (int i=0; i<10; i++)

arrayOfInts[i] = i;
```

Το παράδειγμα στην εφαρμογή Jeliot

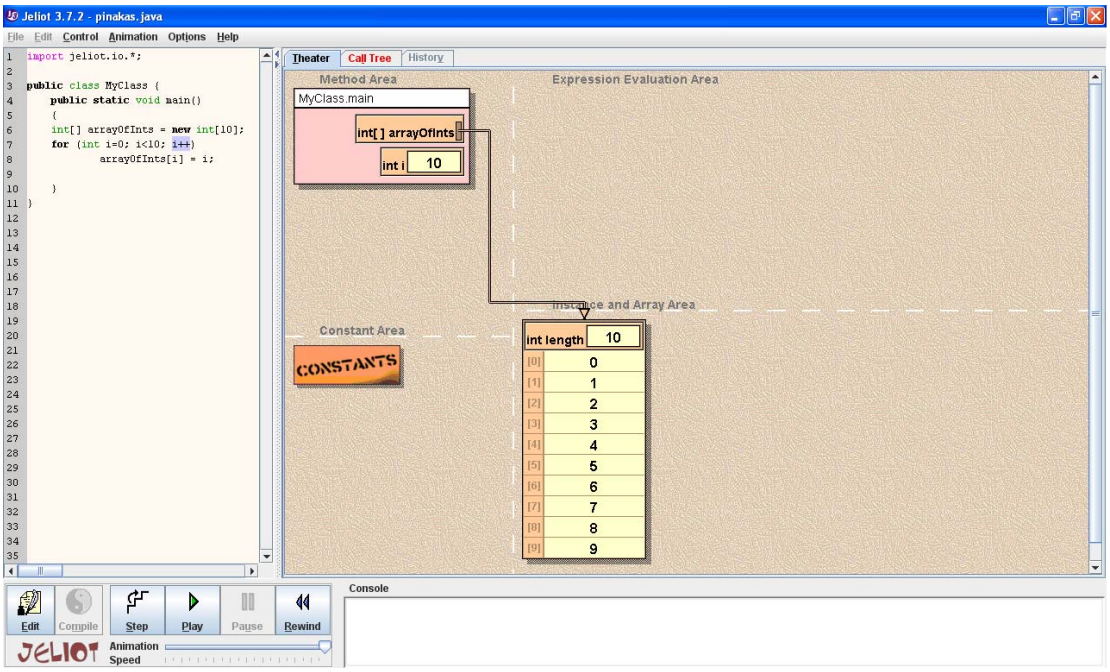


Σχήμα 5.8α: Παράδειγμα χρήσης πινάκων



Σχήμα 5.8β: Παράδειγμα χρήσης πινάκων

Στα σχήματα 5.8α, 5.8β βλέπουμε την λειτουργία των πινάκων στο Jeliot.



Σχήμα 5.8γ: Παράδειγμα χρήσης πινάκων

Στο σχήμα 5.8γ βλέπουμε το αποτέλεσμα της τοποθέτησης 10 ακεραίων αριθμών σε ένα πίνακα 10 θέσεων.

Το ακόλουθο τμήμα κώδικα εμφανίζει τα στοιχεία που βρίσκονται στο πίνακα arrayOfInts.

Παρατηρείστε τη χρήση της ιδιότητας length για να αναφερθούμε στο μήκος του πίνακα:

```
for (int i=0; i<arrayOfInts.length; i++)

        System.out.println(arrayOfInts[i]);
```

5.6.3 Πίνακες περισσοτέρων διαστάσεων

Στη Java μπορείτε να έχετε πίνακες δύο, τριών, τεσσάρων κλπ διαστάσεων. Στη πράξη όμως σπάνια συναντούμε περιπτώσεις στις οποίες η χρήση πινάκων που έχουν περισσότερες από δύο διαστάσεις.

Ο ακόλουθος πίνακας 5.7 δίνει κάποια παραδείγματα για την δημιουργία, αναφορά σε στοιχεία και εύρεση μήκους πινάκων μέχρι και τριών διαστάσεων.

Διάσταση	Δήλωση και Δημιουργία ενός πίνακα ακεραίων	Αναφορά	Εύρεση Μήκους
1	int a[] = new int[10]	a[i]	a.length
2	int a[][]=new int[3][10]	a[i][j]	a.length a[i].length
3	int a[][][]= new int[3][5][10]	a[i][j][k]	a.length a[i].length a[i][j].length

Πίνακας 5.7: Πολυδιάστατοι πίνακες

Το ακόλουθο είναι ένα απλό παράδειγμα χρήσης δύο διαστάσεων. Ανάλογη είναι και η χρήση πινάκων τριών διαστάσεων κλπ.

```
 class Array2 {
  public static void main(String[] args) {
    int marks[][] = new int[3][10];
    for (int i=0; i<marks.length; i++) {
      for (int j=0; j<marks[i].length; j++) {
        marks[i][j] = i*j;
        System.out.print(marks[i][j]+" ");
      }
      System.out.println();
    }
  }
}
```

5.7 Συμβολοσειρές (Strings)

Σ' αυτό το μάθημα θα αναφερθούμε αναλυτικά στην τάξη java.lang.String.

5.7.1 Δήλωση και δημιουργία String

Μία συμβολοσειρά είναι μία σειρά από χαρακτήρες. Στο Jeliot μία συμβολοσειρά υλοποιείται με τη τάξη String του πακέτου java.lang.

Ένα String είναι λοιπόν ένα αντικείμενο και όπως όλα τα αντικείμενα πρέπει να δηλωθεί και να δημιουργηθεί με την εντολή new, όπως στο παράδειγμα:

String s = new String();

Εναλλακτικά μπορούμε να δημιουργήσουμε ένα String αν του αποδώσουμε μία τιμή που θα είναι μία σταθερά συμβολοσειρά, όπως στο παράδειγμα:

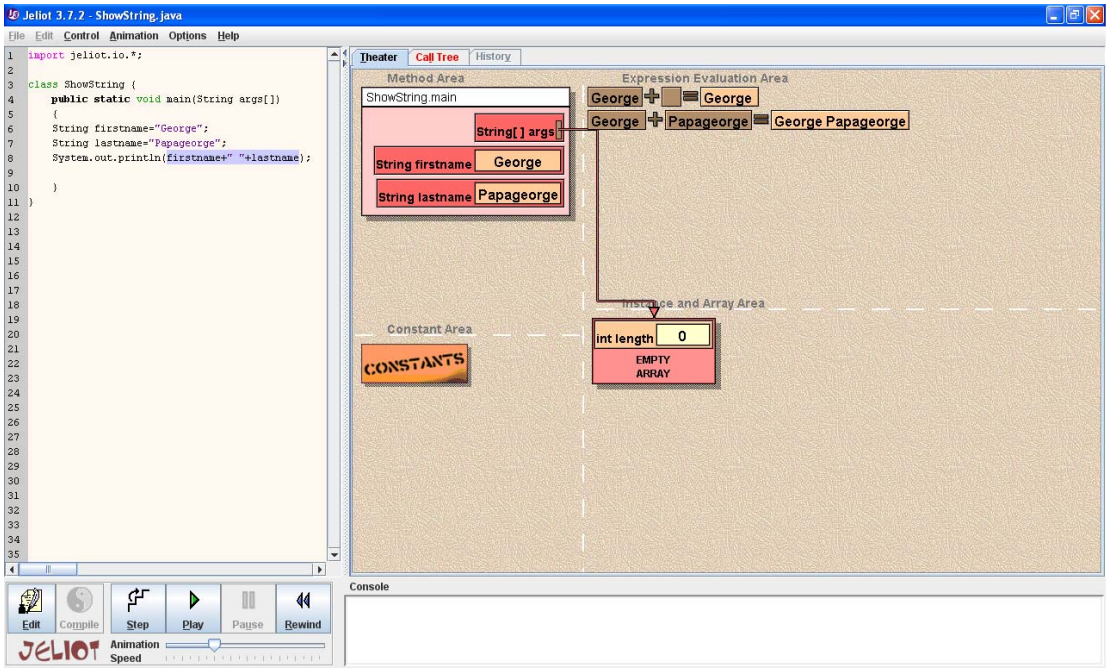
String s = "George";

Αφού δημιουργήσουμε το String μπορούμε στη συνέχεια να το χρησιμοποιήσουμε. Η τάξη String παρέχει πολλές χρήσιμες μεθόδους γι' αυτό.

ΠΡΟΣΟΧΗ: Μια συνήθη πηγή λαθών για τους νέους στο Jeliot είναι η σύγκριση για ισότητα των String. Επειδή τα String είναι αντικείμενα ο τελεστής == δεν συγκρίνει τη τιμή τους, αλλά τη διεύθυνσή τους στη μνήμη. Αν θέλουμε να δούμε αν δύο String έχουν την ίδια τιμή θα πρέπει να χρησιμοποιήσουμε τη συνάρτηση equals. Για παράδειγμα για να δούμε αν το String s1 έχει την ίδια τιμή με το String s2, θα πρέπει να αποτιμήσουμε την έκφραση s1.equals(s2) ή s2.equals(s1).

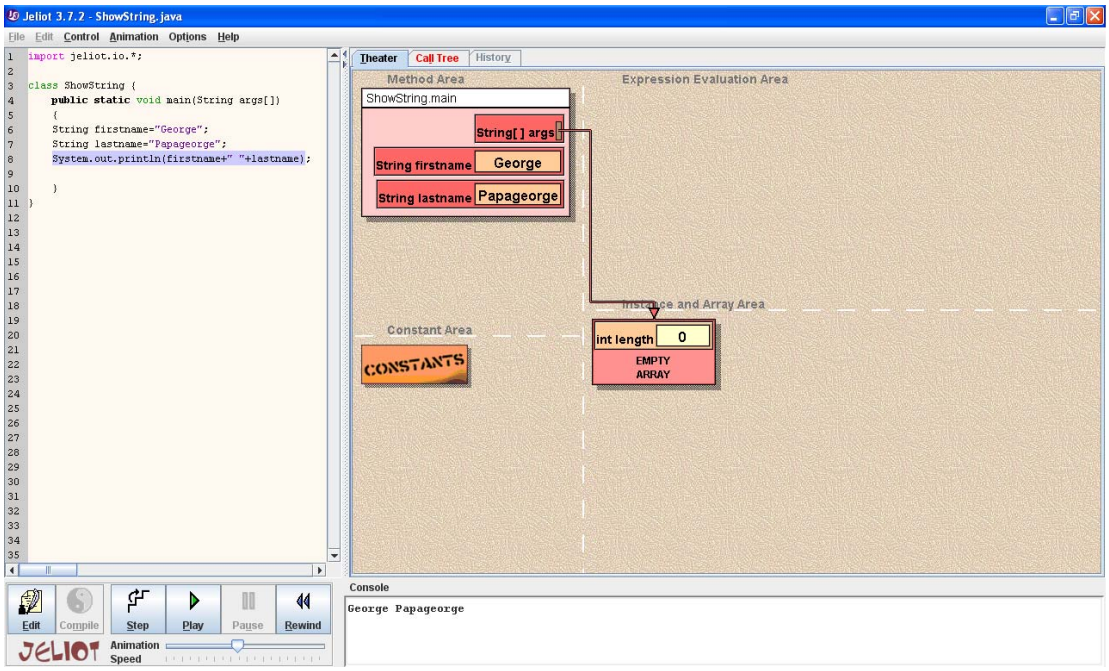
Ένα παράδειγμα στην εφαρμογή Jeliot

Να γίνει ένα πρόγραμμα το όποιο να συνδέει 2 String και να τα εκτυπώνει στην Console.



Σχήμα 5.9α: Παράδειγμα χρήσης String

Στο σχήμα 5.9α έχουμε την δήλωση των μεταβλητών firstname και lastname ως String με αρχικές τιμές 'George', 'Papageorge' αντίστοιχα.



Σχήμα 5.9β: Παράδειγμα χρήσης String

Στο σχήμα 5.9β έχουμε το αποτέλεσμα του προγράμματος στην console της εφαρμογής Jeliot με τα ακόλουθα : George Papageorge

Κώδικας προγράμματος :

```
class ShowString {
   public static void main(String args[])
   {
   String firstname="George";
   String lastname="Papageorge";
   System.out.println(firstname+" "+lastname);

   }
}
```

Κεφάλαιο 6

Συμπεράσματα

Το γεγονός ότι οι σημειώσεις παρουσιάζουν μια στατική απεικόνιση ενός δυναμικού γεγονότος έχει ως αποτέλεσμα να γίνονται συχνά πηγή απογοήτευσης και παρανόησης από τους μαθητές. Οι εφαρμογές τις οπτικοποίησης έχουν ένα πολύ μεγάλο πλεονέκτημα αυτό της επανάληψης αφού επαναλαμβάνουν τα παραδείγματα της τάξης ακριβώς με τον ίδιο τρόπο που παρουσιάστηκαν στο μάθημα.

Η εργασία αυτή αποδεικνύει ότι το Jeliot είναι μια εφαρμογή οπτικοποίησης που βοηθάει αρκετά τη διαδικασία κατανόησης της γλώσσας προγραμματισμού Java. Η δυναμική παρουσίαση του προγράμματος επιτρέπει την παρακολούθηση της εκτέλεσης του πηγαίου κώδικα εμφανίζοντας την επίδραση κάθε εντολής στην απεικόνιση του προγράμματος. Παρέχει μια άποψη της διαδικασίας εκτέλεσης του προγράμματος και εμφανίζει τις αλλαγές των μεταβλητών κατά την διάρκεια της εκτέλεσης του.

Βιβλιογραφία

Bederson, B. (2002). Interfaces for Staying in the Flow. Keynote at IEEE Human-Centric Computing Languages and Environments. Available at **http://www.cs.umd.edu/~bederson/talks/HCCKeynote-Sept2002.ppt**.

Ben-Ari, M. (1998). *Constructivism in computer science education*. In Proceedings of the 29th SIGCSE Technical Symposium, Atlanta, 257–261.

Ben-Ari, M. (2001). *Program Visualization in Theory and Practice*. Informatik / Informatique, Special Issue on Visualization of Software, 8–11.

Ben-Ari, M., Myller, N., Sutinen, E. and Tarhio, J. (2002). *Perspectives on Program Animation with Jeliot*. In Diehl, S. (Ed). Software Visualization, Springer-Verlang, LNCS Vol. 2269, 31-45.

Ben-Bassat Levy, R., Ben-Ari, M. and Uronen, P.A. (2003). *The Jeliot 2000 program animation system*. Computers & Education, 40(1), 15-21.

Birch, M., Boroni, C., Goosey, F., Patton, S., Poole, D., Pratt, C. and Ross, R.J. (1995). *DYNALAB: A Dynamic Computer Science Laboratory Infrastructure Featuring Program Animation*. ACM, SIGSCE Bulletin, 27(1), 29-33.

Braune, B., and Wilhelm, R. (2000). *Focusing in Algorithm Explanation*. IEEE Transactions on Visualization and Computer Graphics, 6(1), 1–7.

Brown, G., Carling, R., Herot, C., Kramlich, D. and Souza, P. (1985). *Program visualization: graphical support for software development*, Computer, 18(8), 27–35.

Brown, M.H., and Hershberger, J. (1991). *Color and sound in algorithm animation*. Tech. Rep. 76a, Systems Research Center, Digital Equipment Corporation.

Brown, M.H., and Najork, M.A. (2001). *Three-dimensional web-based algorithm animations*. Tech. Rep. 170, Compaq Systems Research Center.

Brown, M.H., Najork, M.A., and Raisamo, R. (1997). *A java-based implementation of collaborative active textbooks*. In IEEE Symposium on Visual Languages, 372–379.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A, and Miller, P. (1997). *Mini-languages: A Way to Learn Programming Principles*. Education and Information Technologies, 2(1), 65-83.

Byrne, M.D., Catrambone, R., and Stasko J.T. (1996). *Do Algorithm Animations Aid Learning?* Technical Report GIT-GVU-96-18, Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA.

Cordell, B. (1991). *A Study of Learning Styles and Computer-Assisted Instruction*. Computers and Education, 16(2), 175-83.

Cordova, J.L. (2003). *The use of Web-based visualization techniques and its effect on student comprehension.* Journal of Computing Science in Colleges, 18(6), 67-71.

Cunningham, S., and Hubbold, R. (Eds.). (1992). *Interactive Learning through Visualization*. Springer-Verlag, Berlin, Germany.

Dierker, R. (1995). *The Future of Electronic Education*. In The Electronic Classroom: A Handbook for Education in the Electronic Environment, Erwin Boschmann (Ed.), Learned Information Inc., Medford, New Jersey, 228-235.

Dix, A.J., Finlay, J.E., Abowd, G.D., and Beale, G. (2004). *Επικοινωνία Ανθρώπου-Υπολογιστή*. Δεύτερη έκδοση. Εκδόσεις ΓΚΙΟΥΡΔΑΣ, Αθήνα.

Frechtling, J., and Sharp, L. (1997). User-Friendly Handbook for Project Evaluation. National Science Foundation.

Gallagher, R., (Eds.) (1995). *Computer Visualization: Graphics Techniques for Scientific and Engineering Analysis*. CRC Press.

George, C. (2002). *Using visualization to aid* program *construction tasks*. In Proceedings of the 33rd SIGCSE Technical Symposium, Northern Kentucky, 191–195.

Gloor, P.A. (1992). *AACE - Algorithm Animation for Computer Science Education*. In Proceedings of the 1992 IEEE Workshop on Visual Languages, 25–31.

Green, T.R.G., and Petre, M. (1996). *Usability analysis of visual programming environments: a 'cognitive dimensions' framework*. Journal of Visual Languages and Computing, 7, 131–174.

Hübscher-Younger, T., and Narayanan, N.H. (2001). *How Undergraduate Students' Learning Strategy and Culture Effects Algorithm Animation Use and Interpretation*. In Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT'01), 113-118.

Hwnag, W-Y., Shiu, R-L., Wu, S-L., and Li, C-C. (2002). *An exploration of learning ability transition and material information*. Journal of Educational Computing Research, 26(3), 301-324.

Jenkins, J., and Visser, G. (2003). Making Learning Fun. Available at **http://www.imaginal.nl/learningFun.htm**.

Jerding, D.F. (1994). *Using visualisation to foster Object-oriented Program understanding.* Technical Report GIT-GVU-94-33, Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA.

Kann, C., Lindeman, R.W., and Heller, R. (1997). *Integrating algorithm animation into a learning environment*. Computers & Education, 28(4), 223-228.

Khalili, A., and Shashaani, L. (1994). *The Effectiveness of Computer Applications*, Journal of Research on Computing in Education, 27(1), 48-61.

Korhonen, A., Malmi, L., Mård, P., Salonen, H., and Silvasti, P. (2002). Electronic course material on Data structures and Algorithms. In Proceedings of the Second Annual Finnish / Baltic Sea Conference on Computer Science Education, 16-20.

Lattu, M., Meisalo, V., and Tarhio, J. (2001). *On Using a Visualization Tool as a Demonstration Aid*. First International Program Visualization Workshop, Porvoo, Finland. University of Joensuu Press, 141–162.
McCrickard, D.S., Stasko, J.T., and Zhao, A.Q. (1999). *Exploring animation as a presentation technique for dynamic information sources*. Technical Report GIT-GVU-99-47, Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA.

Meyers, C., and Jones, T. (1993). *Promoting Active Learning*. Jossey-Bass Inc., San Francisco, California.

Moreno, A., Myller, N., Sutinen, E., and Ben-Ari, M. (2004). *Visualizing Program with Jeliot 3*. In Proceedings of the International Working Conference on Advanced Visual Interfaces, AVI 2004, 373-380.

Morrison, M. (1996). *Animation Techniques*. In Vanderburg, G.L. et al. Tricks of the Java Programming Gurus. Sams.net Publishing, 263-291.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodgers, S., and Velázquez-Iturbide, J. Á. (2003A). *Exploring the role of visualization and engagement in computer science education*. ACM SIGCSE Bulletin, 35(2), 131-152.

Naps, T.L, and Stenglein, J. (1996). *Tools for visual exploration of scope and parameter passing in a programming language course*. In Proceedings of the 27th ACM SIGCSE Technical Symposium, Philadelphia, 305–309.

Naps, T.L. (1997). *Algorithm Visualization in the World Wide Web – The Difference Java Makes.* In Proceedings of the 2nd Conference on Integrating Technology into Computer Science Education, 59–61.

Naps, T.L., and Swander, B. (1994). *An Object-Oriented Approach to Program Visualization – Easy, Extensible, and Dynamic*. In Proceedings of the 25th SIGCSE Symposium on Computer Science Education, 46–50.

Naps, T.L., Eagan, J.R, and Norton, L.L. (2000). *JHAVÉ—An Environment to Actively Engage Students in Web-based Algorithm Visualizations*. In Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education, 109–113.

Pierson, W., and Rodger, S.H. (1998). *Web-based Animation of Data Structures Using JAWAA*. In Proceedings of the 29th ACM SIGCSE Technical Symposium on Computer Science Education, 267–271.

Rößling, G., and Freisleben, B. (2000). *Experiences in Using Animations in Introductory Computer Science Lectures*. In Proceedings of the 31st ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2000), Austin, Texas, 134–138.

Shneiderman, B. (1998). Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley, Reading, MA.

Stasko, J.T., Domingue, J., Brown, M.H., and Price, B.A. (Eds.) (1998). *Software Visualization – Programming as a Multimedia Experience*, Cambridge, MA, MIT Press.

Velázquez-Iturbide, J.Á., and Presa-Vázquez, A. (1999). *Customization of Visualizations in a Functional Programming Environment*. In Proceeding of the 29th ASEE/IEEE Frontiers in Education Conference, San Juan, Puerto Rico, 22–28.

Watson, A.B. (1995). *Towards supporting software maintenance with visualisation techniques.* University of Strathclyde.

Yildez, R., and Atkins, M. (1993). *Evaluating Multimedia Applications*, Computers Education, 21(1-2), 133-139.