



The BlueJ Tutorial

Version 1.2
for BlueJ version 1.1

Συγγραφέας: Michael Kölling
School of Network Computing
Monash University

Επιμέλεια: Κλεάνθης Θραμπουλίδης

**Τμήμα Ηλεκτρολόγων Μηχανικών
& Τεχνολογίας Υπολογιστών
Πανεπιστήμιο Πατρών
ΠΑΤΡΑ 2001**

ΠΕΡΙΕΧΟΜΕΝΑ

- 1 ΕΙΣΑΓΩΓΗ
 - 1.1 [Πρόλογος](#)
 - 1.2 [Αντικείμενο και ακροατήριο](#)
- 2 ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ
 - 2.1 [Γενικά](#)
 - 2.2 [Προαπαιτούμενα προγράμματα](#)
 - 2.3 [Πώς θα αποκτήσετε το BlueJ](#)
 - 2.4 [SDK, JDK και JRE](#)
 - 2.5 [Εγκατάσταση](#)
- 3 ΧΡΗΣΗ ΤΟΥ BLUEJ
 - 3.1 [Ξεκινώντας το BlueJ](#)
 - 3.2 [Άνοιγμα ενός BlueJ project](#)
- 4 ΤΑ ΒΑΣΙΚΑ– EDIT / COMPILE / EXECUTE
 - 4.1 [Δημιουργία στιγμιotypών](#)
 - 4.2 [Εκτέλεση \(EXECUTION \)](#)
 - 4.3 [Τροποποίηση \(EDITING\) μιας Κλάσης](#)
 - 4.4 [Compilation](#)
 - 4.5 [Βοήθεια για τα λάθη του COMPILER](#)
- 5 ΠΡΟΧΩΡΩΝΤΑΣ ΛΙΓΟ ΠΑΡΑΠΕΡΑ ...
 - 5.1 [Επιθεώρηση \(Inspection \)](#)
 - 5.2 [Σύνθεση \(Composition \)](#)
- 6 ΔΗΜΙΟΥΡΓΙΑ ΝΕΟΥ PROJECT
 - 6.1 [Δημιουργία καταλόγου για το project](#)
 - 6.2 [Δημιουργία κλάσεων](#)
 - 6.3 [Δημιουργία εξαρτήσεων \(Dependencies\)](#)
 - 6.4 [Διαγραφή στοιχείων](#)
- 7 ΔΙΟΡΘΩΣΗ ΛΑΘΩΝ (DEBUGGING)
 - 7.1 [Τοποθέτηση Breakpoint](#)
 - 7.2 [Ελέγχοντας τον κώδικα βήμα προς βήμα](#)
 - 7.3 [Επιθεωρώντας μεταβλητές \(Inspecting\)](#)
 - 7.4 [Οι λειτουργίες Halt & Terminate](#)
- 8 [ΔΗΜΙΟΥΡΓΙΑ ΑΝΕΞΑΡΤΗΤΩΝ ΕΦΑΡΜΟΓΩΝ](#)
- 9 ΔΗΜΙΟΥΡΓΙΑ APPLETS
 - 9.1 [Εκτέλεση ενός Applet](#)
 - 9.2 [Δημιουργία ενός Applet](#)
 - 9.3 [Δοκιμάζοντας ένα Applet \(Testing\)](#)
- 10 ΑΛΛΕΣ ΛΕΙΤΟΥΡΓΙΕΣ

- 10.1 Άνοιγμα ενός project που δεν ανήκει στο BlueJ με το BlueJ
- 10.2 Προσθήκη κλάσης σε ένα project
- 10.3 Κλήση της main και άλλων static μεθόδων
- 10.4 Δουλεύοντας με βιβλιοθήκες

1. ΕΙΣΑΓΩΓΗ

1.1 Πρόλογος

Το εγχειρίδιο αυτό αποτελεί εισαγωγή στο περιβάλλον προγραμματισμού **BlueJ**. Το **BlueJ** είναι ένα περιβάλλον ανάπτυξης **Java** προγραμμάτων που σχεδιάστηκε ειδικά για να υποστηρίξει την εκπαίδευση νέων προγραμματιστών στην **Java**. Σχεδιάστηκε και υλοποιήθηκε απ' την ομάδα **BlueJ** του πανεπιστημίου *Monash* της Μελβούρνης στην Αυστραλία.

Περισσότερες πληροφορίες για το **BlueJ** είναι διαθέσιμες στην διεύθυνση : <http://bluej.monash.edu>

1.2 Αντικείμενο και ακροατήριο

Το παρόν εγχειρίδιο απευθύνεται σε ανθρώπους που θέλουν να εξοικειωθούν με τις δυνατότητες του περιβάλλοντος. Πρέπει να σημειωθεί ότι δεν επεξηγεί τεχνικές σχεδιασμού του περιβάλλοντος και δεν αναφέρεται στις ερευνητικές διαστάσεις που αυτό εισάγει.

Υποθέτει πως ο αναγνώστης είναι εξοικειωμένος με τη γλώσσα προγραμματισμού **Java**. Πρέπει να τονισθεί ότι στόχος αυτού του φυλλαδίου δεν είναι η εκμάθηση της **Java**.

Στη συνέχεια επιχειρείται μια συνοπτική παρουσίαση της χρήσης του περιβάλλοντος και όχι μια πλήρης και λεπτομερής αναφορά στα χαρακτηριστικά του.

2. ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ

2.1 Γενικά

Το **BlueJ** διατίθεται ως αρχείο **Java** κλάσεων . Γι' αυτό έχει την κατάληξη **".jar"** . Η εγκατάσταση του είναι ιδιαίτερα απλή.

2.2 Προαπαιτούμενα προγράμματα

Για την εγκατάσταση του είναι απαραίτητο να έχετε ήδη εγκαταστήσει το **JDK 1.2.2** ή κάποια νεότερη έκδοσή του. Πιο συγκεκριμένα ορισμένες συναρτήσεις του **BlueJ** δουλεύουν καλύτερα με το **JDK 1.3** . Γι' αυτό συστήνεται η εγκατάστασή του.

Το **JDK** μπορείτε να το κατεβάσετε από την παρακάτω δικτυακή διεύθυνση της Sun <http://java.sun.com/j2se/>.

2.3 Πώς θα αποκτήσετε το BlueJ

Το **BlueJ** διατίθεται σαν αρχείο με όνομα **bluej-xxx.jar** . Τα **xxx** αναφέρονται στην έκδοση του προγράμματος . Το αρχείο **bluej-111.jar** για παράδειγμα αποτελεί την έκδοση 1.1.1 του **BlueJ**.

Μπορείτε να κατεβάσετε το αρχείο από τη διεύθυνση: <http://bluej.monash.edu>

2.4 SDK, JDK και JRE

Μερικές φορές επικρατεί σύγχυση γύρω από τα διάφορα περιβάλλοντα της **Java** τα οποία είναι τα παρακάτω : **SDK** , **JDK** και **JRE** .

Όπως προαναφέραμε απαιτείται η εγκατάσταση κάποιας εκ των νεότερων εκδόσεων του **Java 2 SDK** (Software Development Kit).

Ο όρος **JDK** (Java Development Kit) είναι μια παλιότερη ονομασία για το ίδιο πρόγραμμα καθώς η *Sun* κάποια στιγμή αποφάσισε την αλλαγή αυτή. Παρ' όλα αυτά χρησιμοποιούνται και οι δύο ονομασίες . Για παράδειγμα είτε εγκαταστήσετε το **Java 2 SDK v. 1.3** είτε το **JDK 1.3** θα προκύψει ο κατάλογος **jdk1.3** .

Όσον αφορά στο **JRE** (Java Runtime Environment) εδώ υπάρχει διαφορά . Συγκεκριμένα το **JRE** είναι ένα τμήμα του **SDK**, γι' αυτό και εγκαθίσταται αυτόματα κατά την εγκατάσταση του **SDK**.

Είναι φανερό ότι το **JRE** δεν είναι αρκετό για τη χρήση του **BlueJ**, αφού όπως αναφέραμε αυτό απαιτεί για να τρέξει μια έκδοση του **SDK** το οποίο περιλαμβάνει ορισμένα εργαλεία ανάπτυξης που το **BlueJ** χρησιμοποιεί.

2.5 Εγκατάσταση

Αρχικά κάντε διπλό κλικ στο αρχείο **bluej-111.jar** . Αν το σύστημά σας δεν αναγνωρίζει **jar** αρχεία, τότε το διπλό κλικ δε θα λειτουργήσει .Στην περίπτωση αυτή ανοίξτε ένα παράθυρο σε **DOS prompt** και χρησιμοποιήστε εντολές **γραμμής διαταγής**. Πληκτρολογείτε τα παρακάτω:

```
<jdk-path>/bin/java -jar bluej-111.jar
```

όπου **<jdk-path>** είναι ο κατάλογος (το πλήρες path) που έχετε εγκαταστήσει το **JDK** Στη συνέχεια πατάτε enter .

Ανοίγει ένα παράθυρο που σας επιτρέπει να επιλέξετε τον κατάλογο όπου θα εγκαταστήσετε το **BlueJ** και την έκδοση του **JDK** που θα χρησιμοποιείτε κάθε φορά που θα τρέχει το **BlueJ**.

ΠΡΟΣΟΧΗ : Το path που θα περιέχει τον κατάλογο του **BlueJ** δεν πρέπει να περιέχει καταλόγους με κενά (π.χ "Program Files").

Τέλος πατήστε **Install** και η εγκατάσταση έχει ολοκληρωθεί .

Για οποιοδήποτε πρόβλημα που τυχόν θα αντιμετωπίσετε μπορείτε να ανατρέξετε στο site του BlueJ (ελέγξτε το *FAQ*).

3. ΧΡΗΣΗ ΤΟΥ BLUEJ

3.1 Ξεκινώντας το BlueJ

Κατά τη διαδικασία εγκατάστασης του *BlueJ* έχει δημιουργηθεί στον κατάλογο του προγράμματος ένα *DOS script* με όνομα *BlueJ*.

Για να ενεργοποιήσετε το περιβάλλον *BlueJ* κάντε διπλό κλικ από το γραφικό περιβάλλον. Εναλλακτικά από τη γραμμή διαταγών ενεργοποιήστε το σε περιβάλλον *DOS/UNIX* με τις παρακάτω εντολές που αντιστοιχούν σε ενεργοποίηση χωρίς ή με *project*

\$ bluej

και

\$ bluej examples/people

ΠΡΟΣΟΧΗ: Αν μετά την εκτέλεση των παραπάνω ενεργειών το *BlueJ* δεν ξεκινήσει αλλά το DOS σας βγάλει το μήνυμα :
Η μνήμη του περιβάλλοντος εξαντλήθηκε
 κάντε δεξί κλικ στο *DOS* αρχείο με όνομα *bluej*
 επιλέγετε *ιδιότητες => μνήμη => αρχικό περιβάλλον*
 και αυξάνετε την διαθέσιμη μνήμη στην
 απαιτούμενη τιμή ώστε να τρέχει το *BlueJ*.

3.2 Άνοιγμα ενός BlueJ project

Τα *BlueJ projects* είναι πακέτα *Java* αρχείων και διατίθενται ως κατάλογοι που περιλαμβάνουν αυτά τα αρχεία.

Μπορείτε να ξεκινήσετε το *BlueJ* ανοίγοντας απευθείας ένα *project* (χρησιμοποιώντας μετά το όνομα του *BlueJ* το όνομα του *project*) ή να ανοίξετε το *BlueJ* πρώτα και στη συνέχεια απ' το διαθέσιμο μενού να επιλέξετε

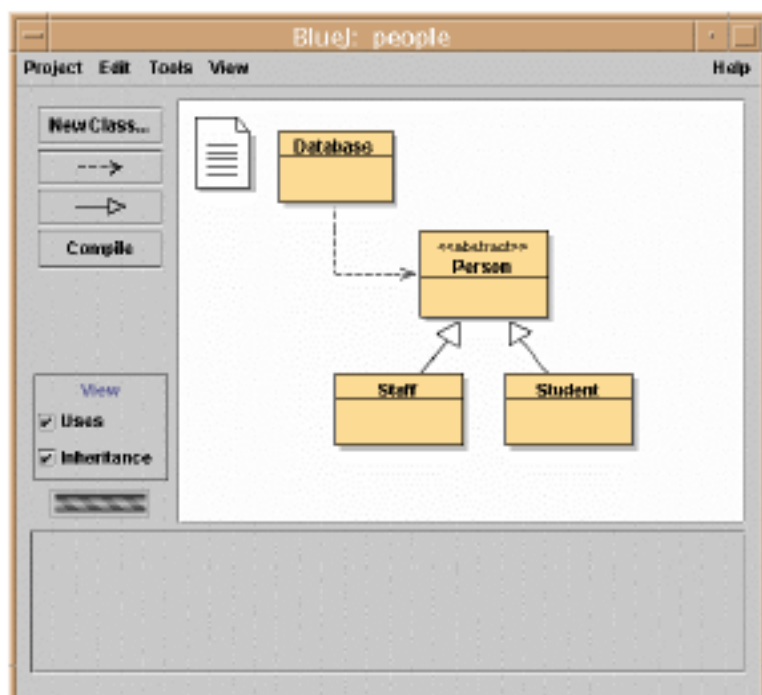
Project => Open

και να ανοίξετε το *project* που θέλετε.

4. ΤΑ ΒΑΣΙΚΑ– EDIT / COMPILE / EXECUTE

Για να προχωρήσετε το κεφάλαιο αυτό ανοίξτε το *project People* το οποίο παρέχεται μαζί με το *BlueJ*. Το *project People* βρίσκεται στον κατάλογο *examples* που είναι υποκατάλογος του αρχικού καταλόγου *BlueJ*).

Στην οθόνη σας θα εμφανιστεί ένα παράθυρο όμοιο με αυτό που φαίνεται στην Εικόνα 1. Πιθανόν να υπάρχουν ορισμένες μικρές διαφορές μεταξύ των δυο παραθύρων, οι διαφορές όμως αυτές είναι ασήμαντες και δεν θα μας απασχολήσουν.



Εικόνα 1 : Το κύριο παράθυρο του BlueJ

4.1 Δημιουργία στιγμιotypών

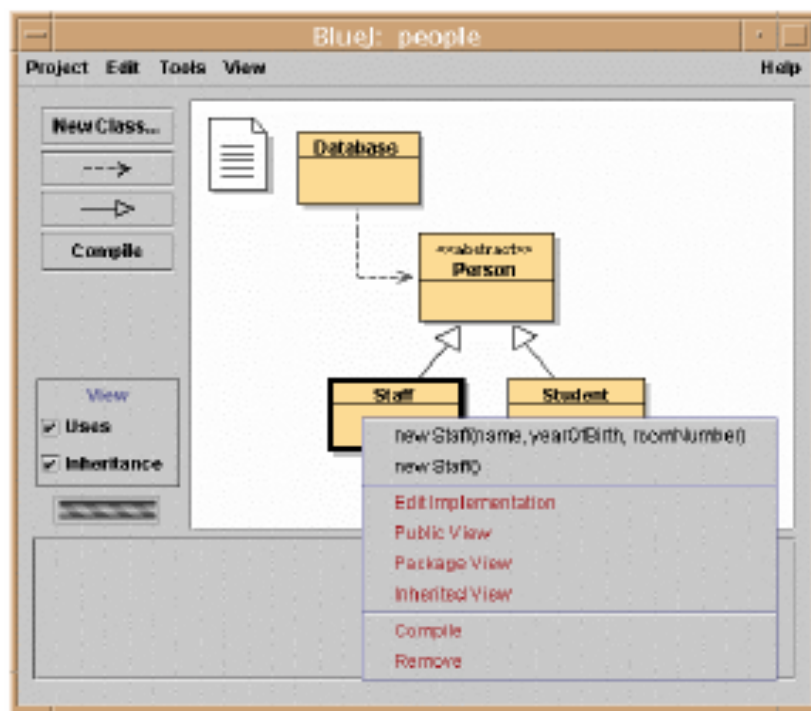
Ένα από τα βασικά χαρακτηριστικά του **BlueJ** είναι το γεγονός ότι μας παρέχει τη δυνατότητα, όχι μόνο να εκτελέσουμε μια ολοκληρωμένη εφαρμογή, αλλά να αλληλεπιδράσουμε απευθείας με στιγμιότυπα οποιασδήποτε κλάσης και να εκτελέσουμε τις **public** μεθόδους της.

Μια εκτέλεση στο **BlueJ** συνήθως περιλαμβάνει τη δημιουργία ενός στιγμιότυπου και στη συνέχεια την ενεργοποίηση μιας απ' τις μεθόδους του. Το γεγονός αυτό είναι πολύ χρήσιμο στην διαδικασία ανάπτυξης μιας εφαρμογής μια και μας επιτρέπει να ελέγχουμε την κάθε κλάση ξεχωριστά αμέσως μετά το γράψιμο της. Δεν είναι απαραίτητο να γράψουμε πρώτα όλη την εφαρμογή μας και μετά να την ελέγξουμε.

ΣΗΜΕΙΩΣΗ : Προφανώς οι **static** μέθοδοι μιας κλάσης μπορούν να εκτελεστούν απευθείας χωρίς να είναι απαραίτητη η δημιουργία στιγμιότυπου. Καθώς μια **static** μέθοδος μπορεί να είναι η **"main"**, μπορούμε να κάνουμε τα ίδια πράγματα που κανονικά συμβαίνουν στις Java εφαρμογές, δηλαδή να ξεκινήσουμε μια εφαρμογή απλά με

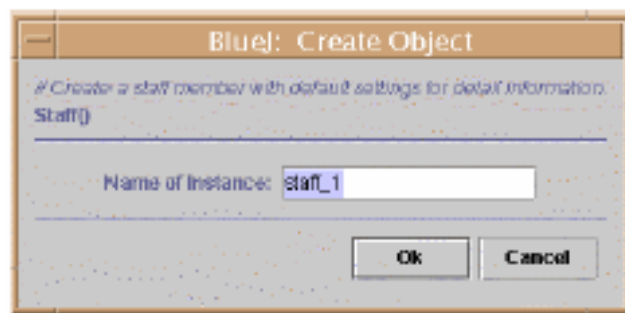
την εκτέλεση μιας *static main* μεθόδου. Στο γεγονός αυτό θα επανέλθουμε αργότερα. Προς το παρόν θα αναφέρουμε κάποια πιο ενδιαφέροντα πράγματα τα οποία δεν μπορούν να γίνουν σε κλασικά Java περιβάλλοντα.

Τα ορθογώνια με ονόματα *Database*, *Person*, *Staff* και *Student* που φαίνονται στο κέντρο του κυρίου παραθύρου της Εικόνας 2 αναπαριστούν τις κλάσεις που περιλαμβάνει η συγκεκριμένη εφαρμογή. Κάνοντας δεξί κλικ πάνω σε οποιαδήποτε από τα παραπάνω εικονίδια ανοίγει ένα μενού με τις λειτουργίες που παρέχει η κάθε κλάση (Εικόνα 2). Στις λειτουργίες που εμφανίζονται περιλαμβάνονται πρώτα οι δημιουργοί της κλάσης με την πρόθεση της λέξης κλειδί *new* και ακολουθούν ορισμένες λειτουργίες που παρέχονται από το περιβάλλον.



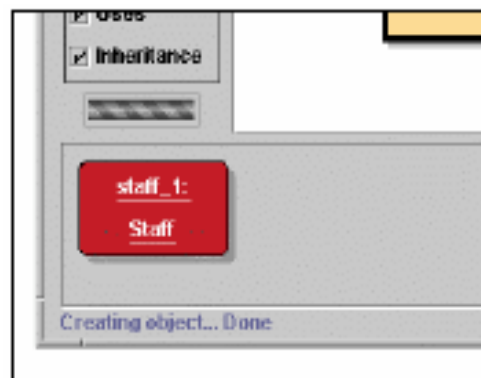
Εικόνα 2 : Κλάση - Μενού – Λειτουργίες

Ας υποθέσουμε πως θέλουμε να δημιουργήσουμε ένα στιγμιότυπο τύπου *Staff*. Κάνουμε δεξί κλικ στο εικονίδιο *Staff* και εμφανίζεται το μενού της εικόνας 2 με τους δύο δημιουργούς, ο ένας με μια παράμετρο, ο άλλος με καμία. Αρχικά επιλέγουμε τον δευτερο δημιουργό, αυτόν χωρίς παραμέτρους. Το αποτέλεσμα είναι η εμφάνιση του παραθύρου της εικόνας 3 .



Εικόνα 3: Δημιουργία στιγμιότυπου χωρίς παραμέτρους

Το παράθυρο της εικόνας 3 σας ζητάει να δώσετε ένα όνομα στο αντικείμενο που θα δημιουργηθεί, ενώ ταυτόχρονα προτείνεται *by default* το όνομα **staff_1**. Για αρχή αυτό το όνομα είναι αρκετά καλό, οπότε πατώντας **OK** δημιουργείται το πρώτο στιγμιότυπο τύπου **Staff**. Το νέο αντικείμενο τοποθετείται στην περιοχή που διατίθεται από το κύριο παράθυρο του *BlueJ* (κάτω μέρος) όπως φαίνεται στην εικόνα 4 για το σκοπό αυτό. Για να εξοικειωθείτε με την κατασκευή στιγμιότυπων δημιουργήστε ορισμένα στιγμιότυπα για τις υπόλοιπες κλάσεις της εφαρμογής.



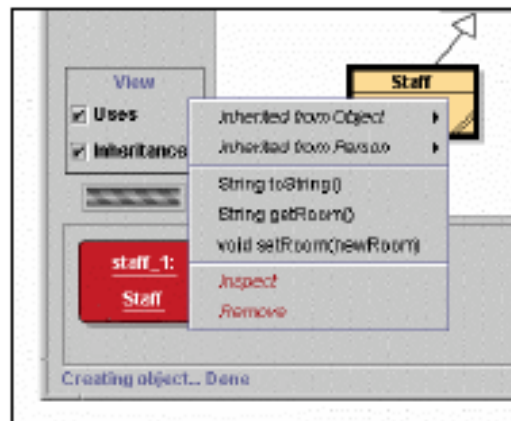
Εικόνα 4 : Τοποθέτηση Αντικειμένου

Θα έχετε παρατηρήσει ήδη ότι στην κλάση **Person** υπάρχει η ετικέτα **<<abstract>>** ενώ αν επιχειρήσετε να δημιουργήσετε αντικείμενο από την κλάση αυτή θα διαπιστώσετε ότι δεν είναι δυνατό (Το γεγονός αυτό είναι άλλωστε χαρακτηριστικό της γλώσσας *Java*).

Περίληψη : Για να κατασκευάσετε ένα στιγμιότυπο επιλέξτε
 ένα δημιουργό από το μενού μιας κλάσης
 αρκεί αυτή να μην είναι **<<abstract>>**.

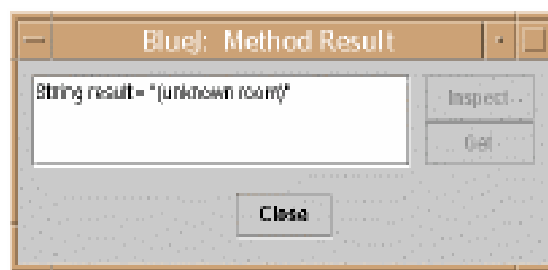
4.2 Εκτέλεση (EXECUTION)

Αφού έχετε ήδη δημιουργήσει ένα νέο στιγμιότυπο, μπορείτε τώρα να εκτελέσετε τις public λειτουργίες που αυτό παρέχει. Έτσι κάνοντας δεξί κλικ πάνω του ανοίγει ένα μενού με τις διαθέσιμες λειτουργίες του αντικειμένου (Εικόνα 5). Συγκεκριμένα το μενού αυτό εμφανίζει τις διαθέσιμες μεθόδους για το αντικείμενο καθώς και δύο ειδικές λειτουργίες που παρέχει το περιβάλλον (**Inspect and Remove**). Σ' αυτές τις τελευταίες θα επανέλθουμε αργότερα. Προς το παρόν θα επικεντρώσουμε το ενδιαφέρον μας στις μεθόδους του αντικειμένου.



Εικόνα 5 : Μενού Αντικειμένου

Όπως μπορείτε να δείτε υπάρχουν οι μέθοδοι **getRoom** και **setRoom** οι οποίες θέτουν και επιστρέφουν αντίστοιχα τον αριθμό του δωματίου για το μέλος του προσωπικού **staff_1**. Αν προσπαθήσετε να καλέσετε τη μέθοδο **getRoom**, επιλέγοντας τη απ' το μενού θα ανοίξει ένα παράθυρο που εμφανίζει το αποτέλεσμα της κλήσης (Εικόνα 6). Σ' αυτήν τη περίπτωση βλέπουμε ότι προκύπτει το όνομα "**(unknown room)**" μια και δεν έχουμε προσδιορίσει κάποιο όνομα δωματίου για αυτό το μέλος (**staff_1**).



Εικόνα 6 : Αποτέλεσμα κλήσης μεθόδου

Οι μέθοδοι που αντιστοιχούν σε μια σούπερκλαση είναι διαθέσιμες μέσω ενός υπομενού. Γυρίζοντας στην εικόνα 5 βλέπουμε ότι στην κορυφή του κεντρικού

μενού υπάρχουν δυο υπομενού , ένα για τις μεθόδους που αντιστοιχούν στο αντικείμενο

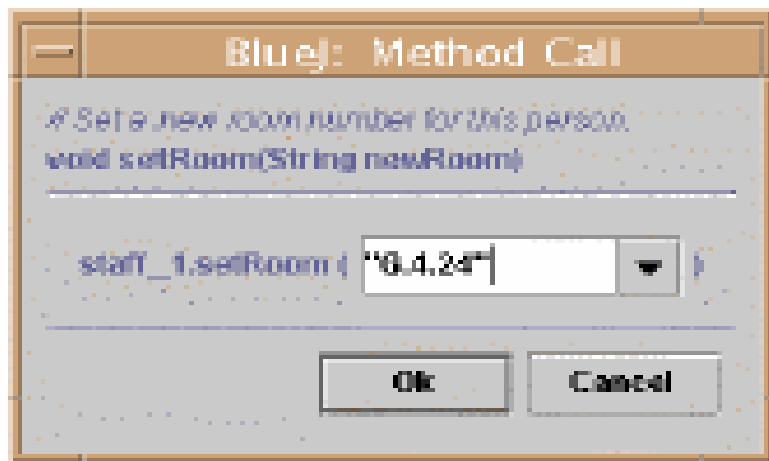
Object(inherited from Object)

και ένα για εκείνες του αντικειμένου

Person(inherited from Person).

Καλέστε μία από τις μεθόδους του αντικειμένου **Person** (έστω τη **getName**) επιλέγοντάς τη απ' το υπομενού. Όπως προαναφέραμε θα προκύψει η αόριστη απάντηση **"(unknown name)"** , αφού δεν έχουμε δώσει στο αντικείμενο μας (**Person**) ένα όνομα.

Ας δούμε τώρα πως μπορούμε να κάνουμε μια κλήση συνάρτησης με παραμέτρους προσδιορίζοντας για παράδειγμα ένα όνομα δωματίου . Μια και οι μέθοδοι **getRoom** και **getName** επιστρέφουν τιμές χωρίς παραμέτρους , καλέστε τη συνάρτηση **setRoom** επιλέγοντας τη απ' το μενού . Θα διαπιστώσετε ότι ανοίγει ένα παράθυρο (Εικόνα 7) ζητώντας σας παραμέτρους.



Εικόνα 7: Αποτέλεσμα κλήσης μεθόδου με παραμέτρους

Στο πάνω μέρος του παραθύρου διαφαίνεται το **interface** της καλούμενης συνάρτησης (περιλαμβάνοντας σχόλια και μία πρόταση). Πιο κάτω υπάρχει μια περιοχή υποδοχής κειμένου στην οποία μπορείτε να δώσετε παραμέτρους. Η πρόταση του πάνω μέρους μας υποδεικνύει ότι η παράμετρος που αναμένεται είναι τύπου **String**. Δώστε λοιπόν ένα όνομα τύπου **String** περικλείοντας το σε διπλά εισαγωγικά (") όπως φαίνεται και στην εικόνα 7. Πατώντας **OK** το όνομα αποθηκεύεται μια και η μέθοδος αυτή δεν επιστρέφει κάποια παράμετρο αλλά μόνο δέχεται. Για να ελέγξετε αν όντως το όνομα που δώσατε έχει αποθηκευτεί καλέστε πάλι τη **getName**. Επαναλάβετε την παραπάνω διαδικασία δημιουργώντας τα δικά σας αντικείμενα καλώντας τους αντίστοιχους δημιουργούς.

Περίληψη : Για να εκτελέσετε μια μέθοδο απλά καλέστε τη απ' το αντίστοιχο μενού.

4.3 Τροποποίηση (EDITING) μιας Κλάσης

Όπως είδαμε παραπάνω μέχρι τώρα ασχοληθήκαμε με τη διασύνδεση ενός αντικειμένου. Παρακάτω θα δούμε την υλοποίηση μιας κλάσης επιλέγοντας **Edit Implementation** από τις λειτουργίες της κλάσης (Να θυμίσουμε ότι με δεξί κλικ πάνω στο εικονίδιο μιας κλάσης προκύπτει το μενού με τις λειτουργίες της – το ίδιο αποτέλεσμα θα έχουμε κάνοντας διπλό κλικ πάνω στο εικονίδιο). Ο κειμενογράφος (*editor*) δεν περιγράφεται λεπτομερώς σ' αυτό το φυλλάδιο μια και η χρήση του είναι ιδιαίτερα απλή. Κάποια στοιχεία για τον κειμενογράφο θα αναφερθούν αργότερα. Προς το παρόν ανοίξτε την υλοποίηση της κλάσης **Staff** και βρείτε την υλοποίηση της μεθόδου **getRoom**. Τροποποιήστε τον κώδικα της μεθόδου προσθέτοντας το πρόθεμα **"room"** στο αποτέλεσμα της συνάρτησης έτσι ώστε να επιστρέφει **"room G.4.24"** αντί για **"G.4.24"**. Συγκεκριμένα αλλάξτε την γραμμή :

return room;

με

return "room " + room;

Να σημειώσουμε ότι το **BlueJ** υποστηρίζει πλήρως **Java** όποτε μπορείτε να κάνετε άφοβα όσες αλλαγές θέλετε σε κάθε κλάση αφού ο compiler θα εντοπίσει πιθανά λάθη .

Περίληψη : Για να δείτε τον πηγαίο κώδικα μιας κλάσης κάντε διπλό κλικ πάνω στο εικονίδιο της .

4.4 Compilation

Αφού κάνετε την παραπάνω αλλαγή στον κώδικα πριν κάνετε οτιδήποτε άλλο ελέγξτε το κύριο παράθυρο του **BlueJ** . Θα παρατηρήσετε ότι το εικονίδιο της κλάσης **Staff** έχει αλλάξει μορφή και είναι σκιαγραφημένο . Το γεγονός αυτό οφείλεται στο ότι δεν έχει γίνει **compilation** στον κώδικα της κλάσης μετά την αλλαγή που κάναμε . Θα αναρωτηθείτε βέβαια γιατί το εικονίδιο της κλάσης αυτής δεν ήταν διαγραμμισμένο όταν πρωτοανοίξατε το **project "people"**.

Απλά οι κλάσεις που παρέχονται στο συγκεκριμένο **project** είναι ήδη ελεγμένες για λάθη (έχουν γίνει **compiled**) . Υπάρχουν βέβαια αρκετές περιπτώσεις **project** του **BlueJ** τα οποία δεν είναι ελεγμένα (**uncompiled**),γι' αυτό ανοίγοντάς τα για πρώτη φορά θα δείτε διαγραμμισμένα τα εικονίδια των επιμέρους κλάσεων .

Ας επιστρέψουμε πίσω στον κειμενογράφο. Στο πάνω μέρος του υπάρχουν κάποια «κουμπιά» που αντιστοιχούν στις πιο συχνά χρησιμοποιούμενες λειτουργίες του κειμενογράφου. Ανάμεσα σ' αυτές είναι και αυτή της συμβολομετάφρασης (**compile**) ,

η οποία σας επιτρέπει να ελέγξετε τον κώδικα της κάθε κλάσης απευθείας απ' τον κειμενογράφο. Πατήστε το «κουμπί» **compile** οπότε, αν δεν υπάρχει λάθος, θα εμφανιστεί ένα μήνυμα στην περιοχή πληροφοριών στο κάτω μέρος του κειμενογράφου που θα το επιβεβαιώνει ("**Class compiled –No syntax errors**").

Αντίθετα αν ο **compiler** εντοπίσει κάποιο λάθος που οδηγεί σε συντακτικό λάθος τότε σκιαγραφείται η γραμμή στην οποία εντοπίστηκε το λάθος ενώ κάτω στη περιοχή πληροφοριών αναφέρονται βοηθητικά σχόλια για την διόρθωση του. Σε περίπτωση που το *project* που ανοίξατε είναι ήδη ελεγμένο (π.χ *people*) εισάγετε κάποια λάθη έτσι ώστε να εξοικειωθείτε με τον **compiler** και τα μηνύματα λάθους του. Αφού ολοκληρώσετε επιτυχώς την διαδικασία της συμβολομετάφρασης (**compiling**) κλειστέ τον κειμενογράφο (**editor**) .

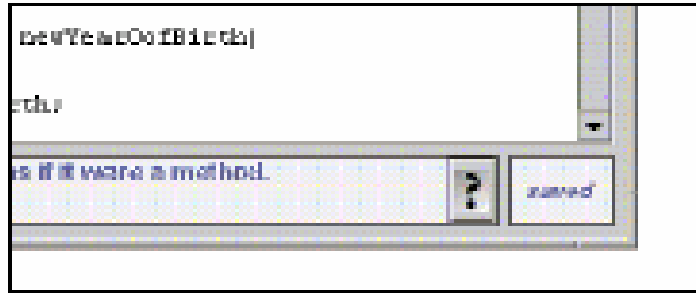
Να σημειώσουμε ότι δεν είναι απαραίτητο να σώζετε τον κώδικα κάθε κλάσης αφού σώζεται αυτόματα (όταν κλείνετε τον **editor** ή πριν κάνετε **compile** μια κλάση). Ο **editor** βέβαια σας παρέχει την δυνατότητα να σώζετε την δουλειά σας κυρίως όμως για περιπτώσεις που το σύστημά σας παρουσιάζει αστάθεια και φοβάστε μήπως χαθούν τα αρχεία σας όταν το σύστημα κολλάει .

Ας γυρίσουμε πίσω στο κεντρικό παράθυρο του *BlueJ* απ' όπου παρέχεται ξανά η δυνατότητα να κάνουμε **compile** (υπάρχει και εδώ όπως μπορούμε να δούμε «κουμπί» **Compile**). Κάνοντας, όμως, **compile** από δω, ελέγχουμε ολόκληρο το *project* για λάθη (συγκεκριμένα από δω εξετάζεται ποιες κλάσεις χρειάζονται **recompilation** οι οποίες γίνονται **compiled** με την σωστή σειρά). Αλλάζτε τον κώδικα μιας ή περισσότερων κλάσεων (με αποτέλεσμα αυτές να γραμμοσκιαστούν) και κάντε τες **compile**. Αν ο **compiler** εντοπίσει λάθος, ο **editor** ανοίγει αυτόματα και μας πηγαίνει στην γραμμή που υπάρχει το λάθος, ενώ στο κάτω μέρος του αναφέρονται πληροφορίες για τη διόρθωση του.

4.5 Βοήθεια για τα λάθη του COMPILER

Πολύ συχνά οι αρχάριοι αντιμετωπίζουν δυσκολίες στην κατανόηση των μηνυμάτων λάθους του **compiler** . Για το λόγο αυτό υπάρχει βοήθεια (**help**).

Ανοίξτε τον **editor** , εισάγετε κάποια λάθη και κάντε **Compile** . Θα παρατηρήσετε ότι στην περιοχή πληροφοριών εμφανίζονται μηνύματα σχετικά με το κάθε λάθος. Θα δείτε επίσης ότι στην κάτω δεξιά γωνία του **editor** υπάρχει ένα ερωτηματικό (?) πατώντας το οποίο σας παρέχονται επιπλέον πληροφορίες για τον τύπο του λάθους (Εικόνα 8) .



Εικόνα 8: Ένα λάθος του *compiler* και το κουμπί *help*

Προς το παρόν και για την τρέχουσα έκδοση του **BlueJ** η βοήθεια δεν διατίθεται για όλα τα μηνύματα λάθους. Παρ' όλα αυτά αρκετά λάθη επεξηγούνται. Τα υπόλοιπα θα συμπεριληφθούν σε νεότερη έκδοση του **BlueJ**.

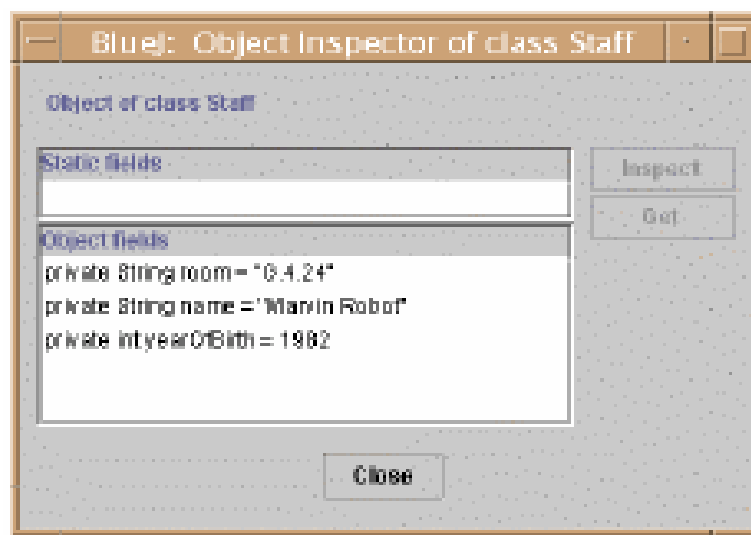
Περίληψη : Για να πάρετε επιπλέον πληροφορίες σχετικά με ένα λάθος πατήστε το ερωτηματικό που βρίσκεται δίπλα στο μήνυμα λάθους.

5. ΠΡΟΧΩΡΩΝΤΑΣ ΛΙΓΟ ΠΑΡΑΠΕΡΑ ...

Σε αυτή την ενότητα θα ασχοληθούμε με κάποιες επιπλέον λειτουργίες που παρέχει το περιβάλλον που δεν είναι μεν απαραίτητες, ωστόσο χρησιμοποιούνται συχνά.

5.1 Επιθεώρηση (Inspection)

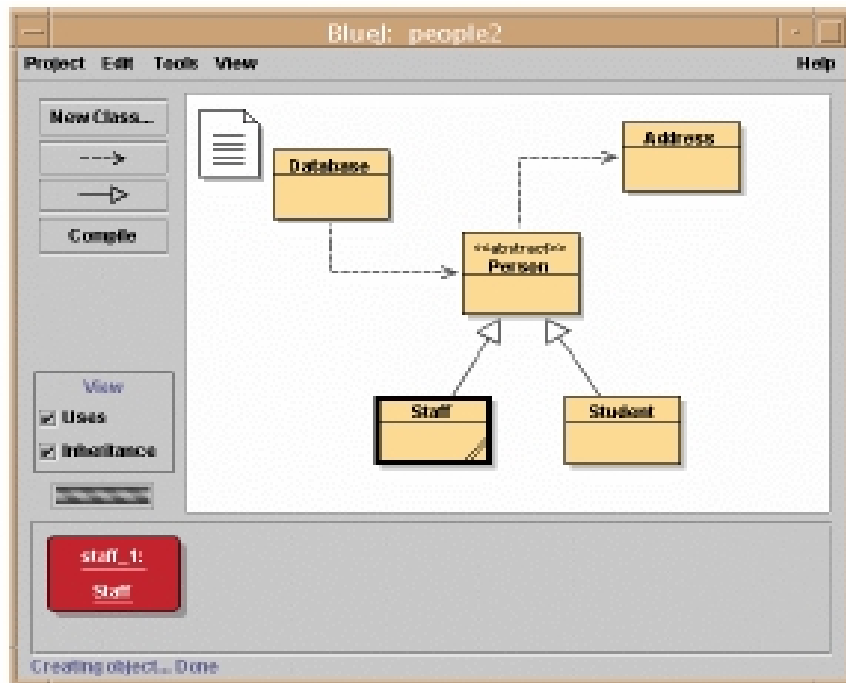
Κατά την εκτέλεση μιας μεθόδου ενός αντικειμένου, θα έχετε παρατηρήσει ότι η μέθοδος αυτή διαθέτει (με δεξί κλικ) την λειτουργία **Inspect** (Εικόνα 5) . Η λειτουργία αυτή μας επιτρέπει να ελέγχουμε την κατάσταση των προσδιδόμενων μεταβλητών σε κάθε αντικείμενο ("**fields**") . Δοκιμάστε να δημιουργήσετε ένα αντικείμενο το οποίο δέχεται παραμέτρους (π.χ αντικείμενο τύπου **Staff** δίνοντας παραμέτρους) . Κατόπιν επιλέξτε τη λειτουργία **Inspect** από το μενού του νέου αντικειμένου που μόλις δημιουργήσατε. Έτσι θα προκύψει ένα παράθυρο το οποίο εμφανίζει τα πεδία (**object fields**) με τον τύπο και την τιμή των παραμέτρων που δώσατε (Εικόνα 9).



Εικόνα 9: Παράθυρο Επιθεώρησης (Inspection)

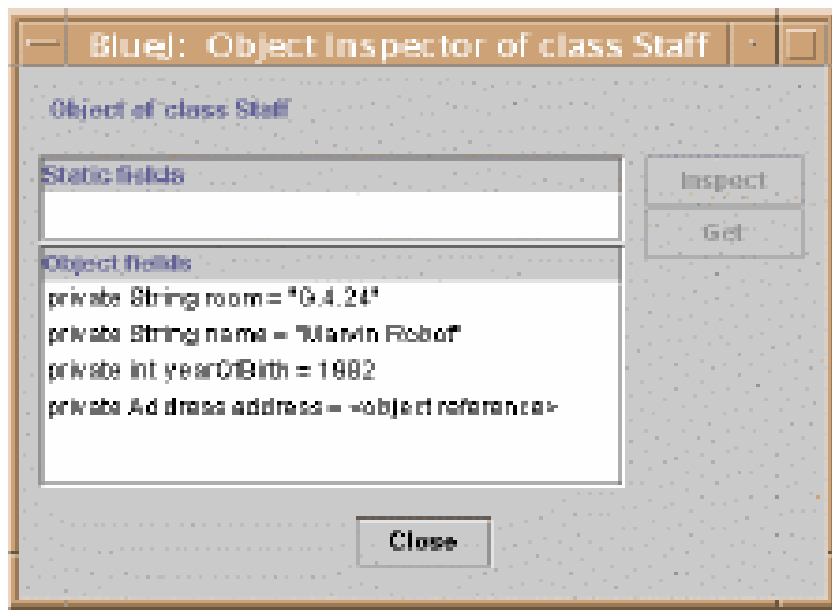
Η επιθεώρηση είναι χρήσιμη για το γρήγορο έλεγχο μιας μεταβαλλόμενης λειτουργίας (μια λειτουργία που έχει τη δυνατότητα να αλλάζει την κατάσταση ενός αντικειμένου) αν εκτελέστηκε σωστά . Στην ουσία η λειτουργία **Inspection** αποτελεί ένα εργαλείο έλεγχου λαθών (**debugging tool**). Στο παράδειγμα **Staff** όλα τα πεδία – παράμετροι (**fields**) είναι απλού τύπου (είτε **strings** είτε τύποι χωρίς αντικείμενο) . Έτσι με την παραπάνω λειτουργία οι τιμές αυτών των τύπων μπορούν να προκύψουν απευθείας, όποτε μπορούμε αμέσως να δούμε αν ο κατασκευαστής έχει κάνει σωστές αναθέσεις . Σε πιο πολύπλοκες περιπτώσεις οι τιμές των παραμέτρων μπορεί να αποτελέσουν αναφορά για αντικείμενα τα οποία είναι ορισμένα από το χρήστη . Για να εξετάσουμε ένα τέτοιο παράδειγμα θα χρησιμοποιήσουμε το *project people2* , το οποίο διατίθεται μαζί με το **BlueJ** .

Ανοίγοντας λοιπόν το *people2* θα προκύψει το παράθυρο της εικόνας 10. Όπως βλέπετε αυτό το δεύτερο παράδειγμα περιέχει μια επιπλέον κλάση σε σχέση με το προηγούμενο *project (people)*, την κλάση *Address*. Μία εκ των παραμέτρων της κλάσης *Person* είναι τύπου *Address* (ορισμένη από το χρήστη).

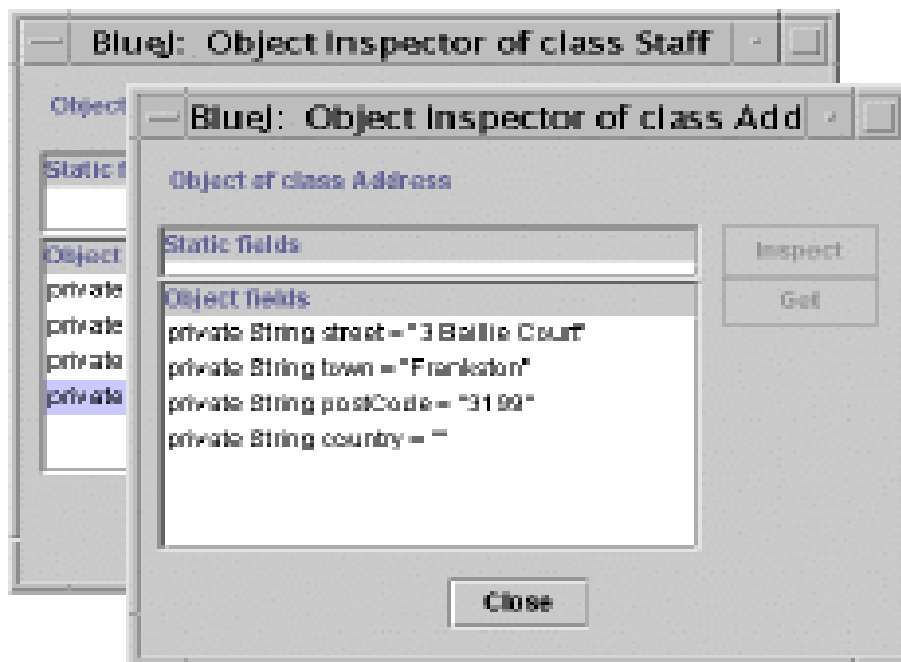


Εικόνα 10: Το παράθυρο του project people2

Στη συνέχεια θα προσπαθήσουμε να κάνουμε επιθεώρηση (*inspection*) σε κλάσεις με παραμέτρους αντικειμένων. Φτιάξτε ένα αντικείμενο τύπου *Staff* και καλέστε την μέθοδο *setAddress* του αντικειμένου (μέσω του υπομενού του *Person*). Δίνοντας μια διεύθυνση, το αντικείμενο *Staff* δημιουργεί εσωτερικά ένα αντικείμενο τύπου *Address* και το αποθηκεύει στο πεδίο διευθύνσεών του. Τώρα κάντε *inspect* στο αντικείμενο *Staff_1* που μόλις δημιουργήσατε. Το αποτέλεσμα της λειτουργίας αυτής (*inspection*) φαίνεται στην εικόνα 11. Όπως βλέπετε στην εικόνα, ένα απ' τα πεδία που προέκυψαν είναι και αυτό της διεύθυνσης (*private Address address*) μόνο που η τιμή της έχει τώρα αντικατασταθεί από την πρόταση *<object reference>*, αφού όπως προαναφέραμε, όταν πρόκειται για πολύπλοκα οριζόμενα από το χρήστη αντικείμενα η τιμή τους δε φαίνεται απ' ευθείας στη λίστα. Για να ελέγξετε τη διεύθυνση επιλέξτε απ' τη λίστα το πεδίο της διεύθυνσης στο παράθυρο που προέκυψε μετά το *inspect* και πατήστε το κουμπί *Inspect* (μπορείτε επίσης να κάνετε διπλό κλικ πάνω στο πεδίο της διεύθυνσης). Έτσι ανοίγει άλλο ένα παράθυρο επιθεώρησης με τις λεπτομέρειες για το αντικείμενο *Address* (Εικόνα 12).



Εικόνα 11: *Inspection με αναφορά αντικειμένου*



Εικόνα 12: *Inspection εσωτερικού αντικειμένου*

Εάν το επιλεγμένο πεδίο είναι **public**, αντί να πατήσετε **Inspect**, μπορείτε επίσης αφού επιλέξετε το πεδίο διεύθυνσης, να πατήσετε το «κουμπί» **Get**. Η λειτουργία αυτή τοποθετεί το επιλεγμένο αντικείμενο στη λίστα αντικειμένων. Προκειμένου να

εξοικειωθείτε με τις παραπάνω λειτουργίες δώστε δικές σας τιμές και ελέγξτε αν έχουν καταχωρηθεί σωστά .

Περίληψη: Η επιθεώρηση αντικειμένων είναι μια απλή μέθοδος εντοπισμού λαθών παρουσιάζοντας την αρχική κατάσταση ενός αντικειμένου .

5.2 Σύνθεση (Composition)

Ο όρος σύνθεση αναφέρεται στη δυνατότητα που έχουμε να περνάμε αντικείμενα σαν παραμέτρους σε άλλα αντικείμενα . Ας δούμε ένα παράδειγμα . Κατασκευάστε ένα αντικείμενο της κλάσης **Database** (θα παρατηρήσετε ότι η κλάση αυτή έχει ένα μόνο κατασκευαστή (**constructor**) , ο οποίος δεν παίρνει παραμέτρους , οπότε η κατασκευή αντικειμένου γίνεται απ' ευθείας). Το αντικείμενο τύπου **Database** έχει την ικανότητα να κρατάει λίστα ατόμων, ενώ διαθέτει τις λειτουργίες προσθήκης ατόμων (αντικείμενα τύπου **person**) και της ταυτόχρονης εμφάνισης όλων των ατόμων που έχουν αποθηκευτεί (αν και η κλήση του **Database** ανεξάρτητα από τα άλλα αντικείμενα της εφαρμογής δε θα έχει αποτέλεσμα). Συγκεκριμένα αν δεν έχετε κατασκευάσει ήδη ένα αντικείμενο τύπου **Staff** ή **Student** , κατασκευάστε τουλάχιστον ένα, μια και για να δεχτεί παραμέτρους το αντικείμενο **Database** πρέπει να υπάρχει στη λίστα αντικειμένων μαζί μ' αυτό ένα τύπου **Staff** ή **Student**. Στη συνέχεια καλέστε τη μέθοδο **addPerson** του αντικειμένου **Database**. Στο παράθυρο που ανοίγει βλέπετε σας υποδεικνύεται να δώσετε παράμετρο τύπου **Person** (θυμηθείτε ότι η κλάση **Person** είναι **abstract** , οπότε δεν μπορούν να προκύψουν αντικείμενα απ' ευθείας από την κλάση αυτή , απλά θεωρούμε τα αντικείμενα τύπου **Staff** ή **Student** ως αντικείμενα τύπου **Person** , έτσι λοιπόν δίνουμε στην **addPerson** αντικείμενο **Staff** ή **Student** παρόλο που μας ζητείται τύπου **Person**) . Για να περάσετε το όνομα του αντικειμένου το οποίο έχετε επιλέξει απ' αυτά που βρίσκονται στη λίστα αντικειμένων (τύπου **Staff** ή **Student**) ως παράμετρο στη μέθοδο **addPerson** που έχετε καλέσει, θα πρέπει είτε να γράψετε το όνομα του αντικειμένου είτε να κάνετε κλικ πάνω στο αντικείμενο αυτό. Τότε το όνομά του περνάει αυτόματα . Τέλος πατήστε **OK** και η κλήση έχει ολοκληρωθεί . Μια και η μέθοδος αυτή δεν επιστρέφει καμία τιμή , για να ελέγξουμε αν η καταχώρηση έγινε σωστά καλούμε τη μέθοδο **listAll** του αντικειμένου **Database** . Η κλήση της μεθόδου αυτής μας εμφανίζει ένα κείμενο με όλες τις τιμές για όλα τα αντικείμενα που έχουμε καταχωρήσει . Επαναλάβετε την παραπάνω διαδικασία για περισσότερα από ένα άτομα για να εξοικειωθείτε με την λειτουργία της σύνθεσης (**Composition**).

Περίληψη: Ένα αντικείμενο μπορεί να περάσει ως παράμετρος στην κλήση μιας μεθόδου με ένα απλό κλικ πάνω στο εικονίδιο του αντικειμένου .

6. ΔΗΜΙΟΥΡΓΙΑ ΝΕΟΥ PROJECT

Η ενότητα αυτή κάνει μια σύντομη αναφορά στη δημιουργία ενός νέου *project*.

6.1 Δημιουργία καταλόγου για το project

Για να δημιουργήσετε ένα νέο *project* επιλέξτε *New...* απ' το κεντρικό μενού. Τότε ανοίγει ένα παράθυρο στο οποίο μπορούμε να καθορίσουμε το όνομα και το φάκελο αποθήκευσης του νέου *project*. Αφού κάνετε τα παραπάνω πατάτε *Create* και ο άδειος φάκελος με το όνομα που δώσατε έχει δημιουργηθεί .

Περίληψη : Για να δημιουργήσουμε ένα νέο *project* επιλέγουμε *New...* απ' το κεντρικό μενού.

6.2 Δημιουργία κλάσεων

Αφού ανοίξετε το *project* στο οποίο θέλετε να δημιουργήσετε τη νέα κλάση πατήστε *New Class* , όποτε θα σας ζητηθεί ένα όνομα για αυτή την κλάση (το όνομα θα πρέπει να βρίσκεται σε μορφή την οποία αναγνωρίζει η *Java*, για παράδειγμα δεν δέχεται ελληνικούς χαρακτήρες). Επίσης σας ζητείται να επιλέξετε τι είδους κλάση θέλετε να δημιουργήσετε, *abstract*, *interface*, *applet* ή *standard* . Η επιλογή αυτή καθορίζει τη δομή του κώδικα που θα κατασκευαστεί αρχικά ως σκελετός της κλάσης σας . Στη συνέχεια βέβαια μπορείτε να αλλάξετε τον τύπο της κλάσης επεμβαίνοντας στον κώδικά της (π.χ προσθέτετε το πρόθεμα *abstract* , αν θέλετε να μετατρέψετε την κλάση σας σε τέτοιου τύπου κ.ο.κ). Αφού κάνετε τα παραπάνω θα παρατηρήσετε ότι κάθε νέα κλάση αναπαριστάται με το αντίστοιχο εικονίδιο - μόνο που έχουν διαφορετικό χρώμα μεταξύ τους ανάλογα με τον τύπο της κλάσης που αντιστοιχούν , έτσι για παράδειγμα οι κανονικές κλάσεις έχουν μπλε χρώμα , οι *abstract* πιο ανοιχτό μπλε , τα *interfaces* ακόμα πιο ανοιχτό μπλε κτλ. Ανοίγοντας τον *editor* κάθε νέας κλάσης θα παρατηρήσετε ότι έχει δημιουργηθεί ήδη *by default* ο σκελετός του κώδικα διευκολύνοντάς σας να ξεκινήσετε .Ο σκελετός αυτός είναι μεν συντακτικά σωστός (άρα μπορεί να γίνει *compile*) , αλλά δεν είναι αρκετός για να κάνει κάτι συγκεκριμένο. Δημιουργήστε τις δικές σας κλάσεις και κάντε τες *compile* για να εξοικειωθείτε με τις παραπάνω λειτουργίες .

Περίληψη : Για να δημιουργήσουμε μια νέα κλάση επιλέγουμε *New Class* από το κεντρικό μενού και της δίνουμε το αντίστοιχο όνομα .

6.3 Δημιουργία εξαρτήσεων (Dependencies)

Στο διάγραμμα κλάσεων φαίνεται η εξάρτηση που υπάρχει μεταξύ τους με τη βοήθεια διάφορων τύπων βελών. Σχέσεις κληρονομικότητας (*Inheritance*), δηλαδή σχέσεις

προέκτασης ή εφαρμογής (**"extends" ή "implements"**) αναπαρίστανται με απλά βέλη, ενώ σχέσεις χρήσης (**"uses"**) με διακεκομμένα. Κατά τη δημιουργία καινούριου πακέτου μπορούμε να προσθέσουμε εξαρτήσεις (βέλη) είτε γραφικά (απ' ευθείας από το διάγραμμα) είτε δηλώνοντάς τα στον κώδικα κάθε κλάσης. Έτσι, με την προσθήκη ενός βέλους γραφικά ανανεώνεται ταυτόχρονα και ο κώδικας, ενώ αντίστοιχα όταν η προσθήκη γίνεται με τη δήλωση της εξάρτησης στον κώδικα ταυτόχρονα εμφανίζεται και στο διάγραμμα. Η προσθήκη γραφικά γίνεται αρχικά με το πάτημα του αντιστοίχου «κουμπιού» που βρίσκονται στο αριστερό μέρος του διαγράμματος (απλά βέλη για σχέσεις κληρονομικότητας και διακεκομμένα για σχέσεις χρήσης) και με τη διασύνδεση των κλάσεων που θέλουμε. Η προσθήκη ενός βέλους κληρονομικότητας προσθέτει στον κώδικα της κλάσης-πηγής τα ορίσματα **"extends" ή "implements"** αναλογα με το αν η κλάση είναι απλή κλάση ή **interface**. Η προσθήκη ενός βέλους χρήσης δεν αλλάζει αμέσως τον κώδικα της κλάσης-πηγής (εκτός αν η κλάση-προορισμού βρίσκεται σε άλλο πακέτο. Σ' αυτή την περίπτωση προστίθεται στον κώδικα το όρισμα **"import"** χωρίς όμως μέχρι τώρα να το έχουμε συναντήσει στα παραδείγματα μας). Η τοποθέτηση ενός βέλους χρήσης στο διάγραμμα προς μια κλάση η οποία στην πραγματικότητα δεν χρησιμοποιείται , θα εμφανίσει αργότερα μήνυμα στον κώδικά της το οποίο θα αναφέρει ότι έχει δηλωθεί μια σχέση χρήσης προς μία κλάση , η οποία δε χρησιμοποιείται ποτέ . Η προσθήκη βέλους μέσω του κώδικα γίνεται εύκολα με την προσθήκη της αντίστοιχης δήλωσης . Όταν σώσετε τον κώδικα της κλάσης , το διάγραμμα ανανεώνεται αυτόματα (θυμηθείτε ότι κλείνοντας τον **editor** , σώζεται οποιαδήποτε αλλαγή έχετε κάνει στον κώδικα) .

Περίληψη: Για να προσθέσετε ένα βέλος ,είτε πατάτε το αντίστοιχο «κουμπί» βέλους και το τοποθετείτε στο διάγραμμα, είτε απλά το δηλώνουμε στον κώδικα της κλάσης-πηγής .

6.4 Διαγραφή στοιχείων

Για να διαγράψουμε μια κλάση από το διάγραμμα , την επιλέγουμε και κατόπιν πατάμε: **Edit => Remove Class** από το κεντρικό μενού του **BlueJ** .Μπορούμε επίσης να επιλέξουμε **Remove** από το μενού που προκύπτει με δεξί κλικ πάνω στην κλάση που θέλουμε να διαγράψουμε.Για να διαγράψουμε ένα βέλος , επιλέγουμε **Edit => Remove Arrow** από το κεντρικό μενού και κάνουμε κλικ πάνω στο βέλος που θέλουμε να διαγράψουμε .

Περίληψη: Για να διαγράψουμε μια κλάση ή ένα βέλος επιλέγουμε τη λειτουργία **Remove** από το αντίστοιχο μενού .

7. ΔΙΟΡΘΩΣΗ ΛΑΘΩΝ (DEBUGGING)

Η ενότητα αυτή παραθέτει τις πιο σημαντικές έννοιες σχετικά με τη λειτουργία της διόρθωσης λαθών (**debugging**) στο **BlueJ** . Βέβαια διάφοροι καθηγητές υπολογιστών αναφέρουν ότι η χρήση της λειτουργίας αυτής στα πρώτα βήματα διδασκαλίας ενός φοιτητή θα ήταν πολύ χρήσιμη , απλά δεν υπάρχει χρόνος για να τη διδάξουν .

Αλλωστε , επισημαίνουν ότι οι φοιτητές δυσκολεύονται ήδη με τη χρήση του **editor**, **compiler** και **execution** , όποτε δεν υπάρχει λόγος και χρόνος να εισάγουν άλλο ένα πολύπλοκο εργαλείο. Γι' αυτό λοιπόν αποφασίσαμε να παρουσιάσουμε τον **debugger** όσο πιο απλά γίνεται . Γίνεται μια προσπάθεια να εξηγήσουμε τον **debugger** τόσο σύντομα ,ώστε οι φοιτητές να μπορούν να τον χρησιμοποιήσουν χωρίς περαιτέρω οδηγίες . Αρχικά λοιπόν, έχουμε χωρίσει τη διαδικασία παρουσίασης σε 3 βήματα :

A . Τοποθέτηση **breakpoints**

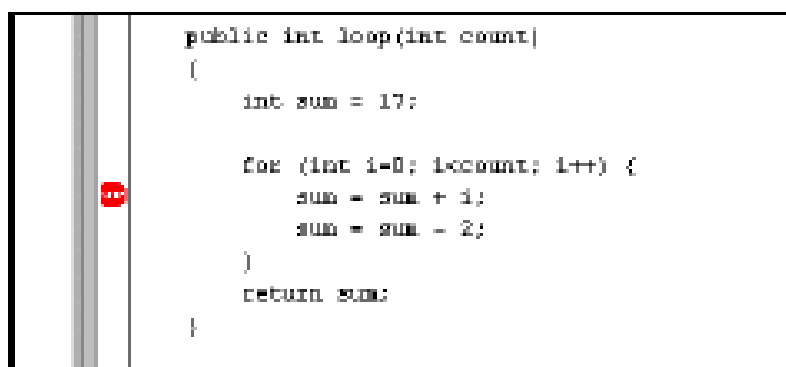
B . Εξέταση βήμα προς βήμα του κώδικα

Γ . Επιθεώρηση (**inspecting**) των μεταβλητών

Στη συνέχεια θα προσπαθήσουμε να εξηγήσουμε καθένα από τα παραπάνω βήματα . Αρχικά ανοίξτε το project **debugdemo** , το οποίο βρίσκεται στον κατάλογο **examples** του **BlueJ** . Το project αυτό περιέχει δύο κλάσεις , οι οποίες δεν αποφέρουν κανένα αποτέλεσμα , απλά είναι ειδικά γραμμένες για την καλύτερη κατανόηση της λειτουργίας του **debugger** .

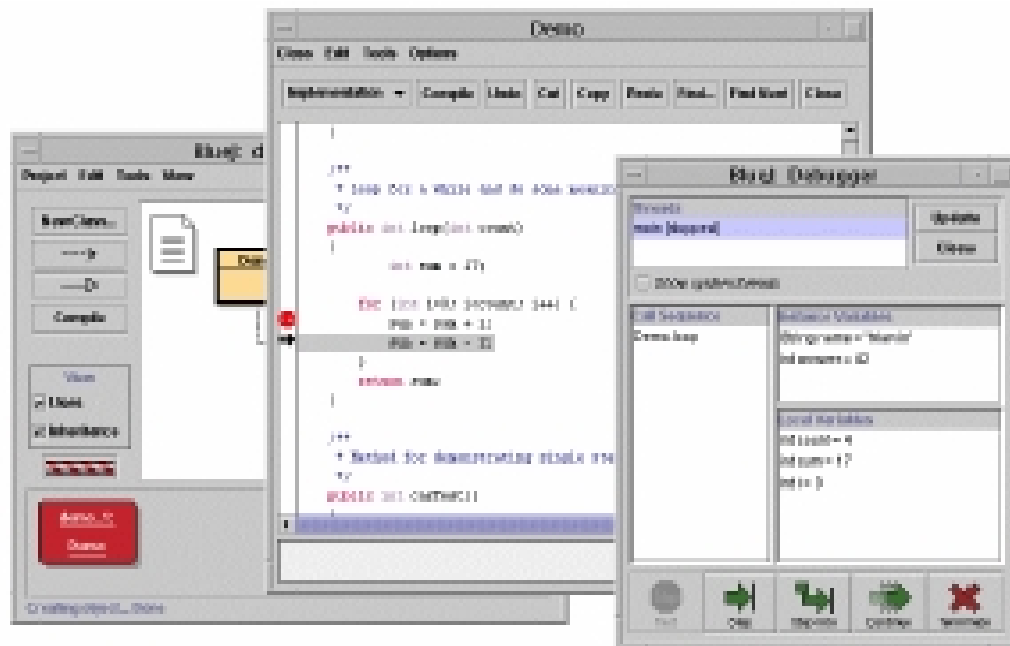
7.1 Τοποθέτηση Breakpoint

Η τοποθέτηση ενός **breakpoint** σε κάποιο σημείο του κώδικα μας επιτρέπει να διακόψουμε την εκτέλεση (**execution**) στο σημείο αυτό. Κατά τη διακοπή της εκτέλεσης μπορείτε να ελέγξετε την κατάσταση των αντικειμένων σας , ενώ σας βοηθάει να καταλάβετε πιο εύκολα τον κώδικα σας βλέποντάς τον κατά τμήματα . Έτσι, ανοίγοντας τον **editor** θα δείτε ότι στο αριστερό του άκρο υπάρχει η περιοχή τοποθέτησης **breakpoints** , στην οποία μπορείτε να θέσετε τα σημεία διακοπής με ένα απλό κλικ δίπλα στη γραμμή του κώδικα που θέλετε να σταματήσει η εκτέλεση του κώδικα (Εικόνα 13).Θα παρατηρήσετε ένα σήμα **'stop'** στο σημείο αυτό μετά το κλικ. Δοκιμάστε λοιπόν, ανοίγοντας την κλάση **Demo** , βρείτε τη μέθοδο **loop** και θέστε **breakpoint** σε κάποιο σημείο του **for loop** .Τότε θα δείτε το σηματάκι **'stop'** που θα εμφανιστεί στο σημείο αυτό .



Εικόνα 13: Τοποθέτηση **breakpoint**

Έτσι όταν κατά την εκτέλεση φτάσουμε στη γραμμή που υπάρχει το **breakpoint** η διαδικασία αυτόματα θα διακοπεί . Κατασκευάστε ένα αντικείμενο της κλάσης **Demo** , καλέστε τη μέθοδο **loop** και δίνοντας μια παράμετρο της αρεσκείας σας (έστω 10) θα δείτε ότι η εκτέλεση θα σταματήσει στο σημείο όπου υπάρχει το **breakpoint**, ενώ έχει προστεθεί και σηματάκι (βέλος) λάθους . Ταυτόχρονα ανοίγει και ένα παράθυρο του **debugger** παρόμοιο με αυτό της Εικόνας 14 .



Εικόνα 14: Παράθυρο του debugger

Το πιο σημαντικό σημείο είναι ότι το βέλος στον **editor** δείχνει επίσης την επόμενη γραμμή που θα εκτελεστεί (δηλ. η εκτέλεση έχει σταματήσει πριν από αυτή τη γραμμή – σημείο **breakpoint**).

Περίληψη : Για να τοποθετήσουμε ένα **breakpoint** απλά κάνουμε κλικ στο αριστερό μέρος του **editor** στη γραμμή που θέλουμε .

7.2 Ελέγχοντας τον κώδικα βήμα προς βήμα

Έχοντας ακολουθήσει την παραπάνω διαδικασία , η εκτέλεση θα έχει σταματήσει στο σημείο που έχουμε καθορίσει . Στη συνέχεια μπορούμε να ανατρέξουμε στον κώδικα και να δούμε σταδιακά την πρόοδο της εκτέλεσης . Αυτό επιτυγχάνεται πατώντας το «κουμπί» **Step** (βρίσκεται στο κάτω μέρος του παραθύρου του **debugger**) .Θα δείτε ότι κάθε φορά που πατάτε το **Step** εκτελείται μία μόνο γραμμή και μετά σταματάει η εκτέλεση μέχρι να το ξαναπατήσετε οπότε η εκτέλεση μεταφέρεται στην επόμενη γραμμή κώδικα (αυτό φαίνεται και από τη μετακίνηση του βέλους, το οποίο υποδεικνύει την επόμενη γραμμή που θα εκτελεστεί).Θα παρατηρήσετε επίσης ότι ανάλογα με τη γραμμή που εκτελείται αλλάζουν και οι τιμές των μεταβλητών που απεικονίζονται στο κεντρικό παράθυρο του **debugger** (π.χ η μεταβλητή **sum**) . Έτσι μπορείτε να εκτελέσετε τον κώδικα βήμα προς βήμα ελέγχοντας τον καλύτερα . Αν κουραστήκατε από τη σταδιακή εκτέλεση μπορείτε να κάνετε κλικ πάνω στο **breakpoint** , οπότε αυτό διαγράφεται και κατόπιν να πατήσετε το «κουμπί» **Continue** για να ξεκινήσει η εκτέλεση από την αρχή , χωρίς διακοπή πλέον .

Ας δοκιμάσουμε τα παραπάνω σε μια άλλη μέθοδο. Θέστε **breakpoint** στη μέθοδο **cartest()** της κλάσης **Demo** , στην γραμμή : **places = myCar.seats();** και καλέστε τη μέθοδο . Όταν η εκτέλεση φθάσει στη γραμμή του **breakpoint** , είστε έτοιμοι να εκτελέσετε μια γραμμή η οποία περιέχει την κλήση της μεθόδου **seats()** της κλάσης

Car . Πατώντας τώρα το **Step** του *debugger* η εκτέλεση θα υπερπηδήσει τη συγκεκριμένη γραμμή . Κατόπιν πατήστε το **Step Into** .Εδώ πρέπει να σημειώσουμε ότι με το **Step Into** μπαίνουμε στη μέθοδο που καλείται στον κώδικα .Η μέθοδος αυτή εκτελείται γραμμή προς γραμμή (πατώντας το **Step**) και επιστρέφουμε στην αρχική κλάση μέσα από την οποία κάναμε την κλήση. Συγκεκριμένα στο παράδειγμα μας η εκτέλεση μεταφέρεται στην μέθοδο *seats()* της κλάσης *Car* και όταν η εκτέλεση της ολοκληρωθεί ο έλεγχος επιστρέφει κανονικά στην κλάση *Demo*. Οι λειτουργίες **Step** και **Step Into** έχουν ακριβώς το ίδιο αποτέλεσμα αν στη γραμμή που πρόκειται να εκτελεστεί δεν περιέχεται κλήση μεθόδου.Δοκιμάστε τα παραπάνω για να εξοικειωθείτε με τη λειτουργία και τα μηνύματα που εμφανίζει ο *debugger* .

Περίληψη : Για την βήμα προς βήμα εκτέλεση του κώδικα πατήστε τα «κουμπιά» **Step** και **Step Into** του *debugger*.

7.3 Επιθεωρώντας μεταβλητές (Inspecting)

Όταν ελέγχουμε τον κώδικά μας για λάθη (*debugging*) , είναι σημαντικό να μπορούμε να επιβλέπουμε την κατάσταση των αντικειμένων μας (τοπικές και καθολικές μεταβλητές) . Παρόλο που τα περισσότερα σχετικά μ' αυτήν την ενότητα τα έχουμε ήδη συναντήσει, θα δούμε ότι δε χρειαζόμαστε επιπλέον εντολές για να επιθεωρούμε μεταβλητές μιας και τόσο οι καθολικές μεταβλητές του ενός αντικειμένου όσο και οι τοπικές μεταβλητές μιας μεθόδου ανανεώνονται και εμφανίζονται αυτόματα στον *debugger* . Ακόμη, μπορούμε , αφού επιλέξουμε την αντίστοιχη μέθοδο από τη λίστα (*call sequence*) του *debugger* , να επιβλέπουμε μεταβλητές άλλων ενεργών αντικειμένων και μεθόδων. Δοκιμάστε για παράδειγμα , θέτοντας ένα *breakpoint* πάλι στην μέθοδο *carTest()* και καλέστε την. Στην αριστερή πλευρά του *debugger* , όπως προαναφέρθηκε, υπάρχει η ακολουθία κλήσεων (*call sequence*), η οποία περιέχει τις παρακάτω μεθόδους :

Car.seats

Demo.carTest.Debugging

Η σειρά αυτή υποδεικνύει ότι η *Car.seats* κλήθηκε από την *Demo.carTest* , με την οποία μπορούμε επιλέγοντάς την να δούμε την κλάση προορισμού της και τις παρούσες τιμές των μεταβλητών της . Εάν τώρα θέσετε *breakpoint* στην γραμμή *Car Mylar = new Car(2, 3);* και καλέσετε πάλι την *carTest()* , θα παρατηρήσετε ότι η τοπική μεταβλητή *myCar* αναφέρεται ως *<object reference>*. Με τον όρο αυτό παρουσιάζονται όλες οι τιμές που αναφέρονται σε τύπους αντικειμένων (εκτός από *Strings*) . Μπορούμε να επιθεωρήσουμε τη μεταβλητή αυτή με διπλό κλικ πάνω της , όποτε ανοίγει ένα παράθυρο με τις διαθέσιμες τιμές γι' αυτή τη μεταβλητή . Στην πραγματικότητα δεν υπάρχει διάφορα μεταξύ της επιθεώρησης (*inspection*) αντικειμένων με αυτό τον τρόπο και της επιθεώρησης απ' ευθείας από τη λίστα αντικειμένων .

Περίληψη : Η επιθεώρηση (*inspection*) μεταβλητών μιας και εμφανίζονται αυτόματα απ' τον *debugger*.

7.4 Οι λειτουργίες Halt & Terminate

Μερικές φορές ένα πρόγραμμα μπορεί να τρέχει για αρκετή ώρα με αποτέλεσμα να αναρωτιόμαστε αν όλα πηγαίνουν καλά . Αυτό μπορεί να οφείλεται σε κάποιον μη καλά ορισμένο βρόχο ή απλά το πρόγραμμα να απαιτεί χρόνο για να εκτελεστεί . Το **BlueJ** μας δίνει τη δυνατότητα να επεμβαίνουμε και σε τέτοιες περιπτώσεις . Καλέστε για παράδειγμα την μέθοδο **longloop()** της κλάσης **Demo**, η οποία θέλει κάποιο χρόνο για να εκτελεστεί . Εάν λοιπόν δεν έχετε ανοίξει ήδη τον **debugger** , κάντε διπλό κλικ στο εικονίδιο του **BlueJ** (βρίσκεται κάτω απ'το πλαίσιο στο οποίο τσεκάρουμε για Uses και Inheritance) που μας δείχνει πόσο διαρκεί η εκτέλεση μιας εφαρμογής - αποκτά ριγέ κόκκινο χρώμα όσο διαρκεί η εκτέλεση . Το εικονίδιο αυτό ουσιαστικά αποτελεί συντόμευση του **debugger** , οπότε με διπλό κλικ τη στιγμή που η εκτέλεση βρίσκεται σε εξέλιξη , ο **debugger** ανοίγει . Έτσι πατώντας το «κουμπί» **Halt** του **debugger** η εκτέλεση διακόπτεται , όπως ακριβώς θα συνέβαινε αν είχαμε θέσει **breakpoint** . Κατόπιν μπορούμε αν θέλουμε να ελέγξουμε τις μεταβλητές μας ή οτιδήποτε άλλο , να συνεχίσουμε την εκτέλεση βήμα προς βήμα κατά τα γνωστά με τα «κουμπιά» **Step** και **Step Into** ή να αφήσουμε την εκτέλεση να συνεχιστεί πατώντας το **Continue** , να την διακόψουμε πάλι με το **Halt** κ.ο.κ. Αν θέλετε να τερματίσετε εντελώς την εκτέλεση (για παράδειγμα μπορεί να διαπιστώσετε ότι η εκτέλεση καθυστερεί εξαιτίας ενός ατέρμονα βρόχου) πατήστε το **Terminate**. Εδώ πρέπει να πούμε ότι η λειτουργία **Terminate** δεν πρέπει να χρησιμοποιείται πολύ συχνά , γιατί μπορεί να προκαλέσει σφάλματα σε σωστά γραμμένο κώδικα , άρα ενδείκνυται η χρήση του μόνο σε περιπτώσεις ανάγκης .

Περίληψη: Οι λειτουργίες **Halt** και **Terminate** χρησιμοποιούνται για προσωρινή και μόνιμη διακοπή της εκτέλεσης ενός προγράμματος αντίστοιχα .

8. ΔΗΜΙΟΥΡΓΙΑ ΑΝΕΞΑΡΤΗΤΩΝ ΕΦΑΡΜΟΓΩΝ

Το **BlueJ** μας δίνει τη δυνατότητα να δημιουργήσουμε εκτελέσιμα **jar** αρχεία . Τα αρχεία αυτά μπορούν να εκτελεστούν είτε με διπλό κλικ πάνω τους σε κάποια λειτουργικά (π.χ **Windows**) ή απλά με την εντολή **java -jar <file-name>.jar** σε περιβάλλοντα **Unix** ή **DOS prompt** . Ανοίξτε λοιπόν το project **hello** (βρίσκεται στον κατάλογο **examples** του **BlueJ**), κάντε το **compile** (αν δεν έχει γίνει ήδη) και επιλέξτε **Project => Export...** από το κεντρικό μενού. Ανοίγει ένα παράθυρο, το οποίο σας ζητάει να καθορίσετε με ποιο **format** θα αποθηκευτεί το αρχείο. Επιλέξτε **"jar file"** για να δημιουργήσετε εκτελέσιμο **jar** αρχείο. Για να γίνει εκτελέσιμο πρέπει επίσης να καθορίσουμε μια κλάση **main** η οποία πρέπει να περιέχει μια έγκυρα ορισμένη μέθοδο **main** (δηλ. με τη δήλωση **public static void main(String[] args)**) . Στο παράδειγμά μας η επιλογή της κλάσης **main** είναι εύκολη μια και υπάρχει μόνο μια κλάση, η κλάση **Hello**. Επιλέγουμε λοιπόν αυτήν, ενώ αν είχαμε άλλο αντικείμενο με περισσότερες επιλογές θα διαλέγαμε την κλάση εκείνη, η οποία περιέχει τη μέθοδο **main** που θέλουμε να εκτελέσουμε . Παρόλο που συνήθως αποφεύγεται να

συμπεριλαμβάνουμε τον πηγαίο κώδικα στα εκτελέσιμα αρχεία, υπάρχει η δυνατότητα να το κάνουμε αν θέλουμε. Για να γυρίσουμε στα προηγούμενα πατάμε **Continue**, όποτε ανοίγει ένα παράθυρο το οποίο μας ζητάει να δώσουμε το όνομα του αρχείου και τον κατάλογο προορισμού. Δίνουμε το όνομα **hello**, πατάμε **Create** και το εκτελέσιμο αρχείο **hello.jar** δημιουργείται. Μπορείτε να τρέξετε το αρχείο με διπλό κλικ μόνο αν η εφαρμογή χρησιμοποιεί γραφικό περιβάλλον (**GUI interface**). Το παράδειγμα μας χρησιμοποιεί κείμενο εισόδου-εξόδου (**I/O**) όποτε πρέπει να το ξεκινήσουμε είτε από ένα τερματικό (**UNIX**) είτε μέσω **MSDOS**. Για να το τρέξουμε (είτε από τερματικό είτε από **DOS**), μπαίνουμε στον κατάλογο που έχουμε σώσει το αρχείο (θα δούμε ότι υπάρχει ένα αρχείο με το όνομα **hello.jar**) και γράφουμε την ακόλουθη εντολή:

java -jar hello.jar

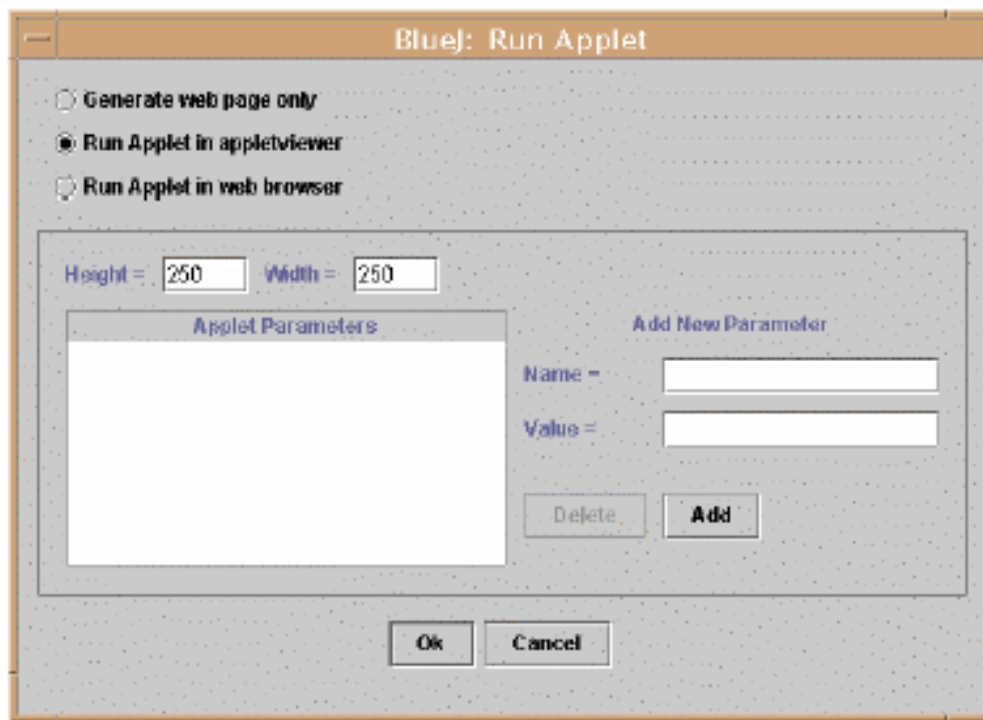
Όποτε αν η **java** έχει εγκατασταθεί σωστά στο σύστημα μας το αρχείο θα εκτελεστεί κανονικά.

Περίληψη: Για να δημιουργήσουμε μια νέα εφαρμογή χρησιμοποιούμε την εντολή **Project => Export...** από το κεντρικό μενού του **BlueJ**.

9. ΔΗΜΙΟΥΡΓΙΑ APPLETS

9.1 Εκτέλεση ενός Applet

Το **BlueJ** μας δίνει τη δυνατότητα να δημιουργήσουμε και να εκτελέσουμε **applets** σαν όλες τις υπόλοιπες εφαρμογές. Για το σκοπό αυτό διατίθενται από το **BlueJ** κάποια έτοιμα **applets**. Ας προσπαθήσουμε να τρέξουμε ένα εξ' αυτών. Ανοίξτε λοιπόν το project **appletClock** από τον κατάλογο **examples**. Θα παρατηρήσετε ότι το project αυτό περιέχει μια μόνο κλάση, την κλάση **Clock**, η οποία πιθανόν να είναι μαρκαρισμένη με τα γράμματα **www** (η συγκεκριμένη δεν είναι) ενδεικτικό του ότι η κλάση είναι **applet**. Επιλέξτε τη λειτουργία **Run Applet** από το μενού που προκύπτει με δεξί κλικ πάνω στην κλάση. Έτσι προκύπτει ένα παράθυρο που μας επιτρέπει να κάνουμε τις επιλογές μας (Εικόνα 15).



Εικόνα 15: Παράθυρο εκτέλεσης Applet

Όπως μπορείτε να δείτε μπορούμε να τρέξουμε το *applet* είτε σε *web browser* είτε σε *applet viewer* ή να δημιουργήσουμε απευθείαν τη *web page* χωρίς να το τρέξουμε. Αφήστε λοιπόν τις ήδη επιλεγμένες ρυθμίσεις και πατήστε **OK**. Μετά από λίγο ο *applet viewer* θα ανοίξει παρουσιάζοντας το *applet* σε λειτουργία. Να σημειώσουμε εδώ ότι ο *applet viewer* έχει ήδη εγκαταθεί στο σύστημά σας κατά την εγκατάσταση του *JDK* που διαθέτεται, ώστε να εξασφαλίζεται ότι είναι ίδια έκδοση με τον *Java compiler* σας. Ο *applet viewer* συνήθως δεν προκαλεί προβλήματα σε σχέση με τους *web browsers* οι οποίοι σε περιπτώσεις που δεν έχουν την ίδια έκδοση με τον *compiler* της *Java* μπορεί να μη λειτουργήσουν σωστά. Αυτά βέβαια έχουν ξεπεραστεί στις νεότερες εκδόσεις *browsers*. Στα συστήματα με *Microsoft Windows*, το *BlueJ* χρησιμοποιεί τον default *browser* του συστήματος, ενώ στα *Unix* συστήματα ο *browser* καθορίζεται από τις ρυθμίσεις του *BlueJ*.

Περίληψη: Για να τρέξουμε ένα *applet* επιλέγουμε *Run Applet* από το μενού του.

9.2 Δημιουργία ενός Applet

Η δημιουργία ενός *Applet* δε διαφέρει σε τίποτα από τη δημιουργία μιας νέας κλάσης. Έτσι πατάμε *New Class* από το μενού του *BlueJ* και στο παράθυρο που ανοίγει επιλέγουμε *Applet* σαν *Class Type*, δίνουμε το όνομα που θέλουμε, πατάμε **OK** και η δημιουργία του νέου *Applet* έχει ολοκληρωθεί. Κατόπιν, αφού το κάνουμε *Compile*

μπορούμε να το τρέξουμε με τον τρόπο που έχουμε ήδη αναφέρει . Όπως είδαμε και κατά τη δημιουργία νέας κλάσης έτσι και εδώ το νέο **Applet** που δημιουργήσαμε περιέχει ήδη ένα σκελετό με κώδικα και σχόλια για κάθε μέθοδο , ώστε να μας καθοδηγήσει και να προσθέσουμε με ευκολία το δικό μας κώδικα (ο συγκεκριμένος κώδικας όταν τρέχει παράγει δυο γραμμές κειμένου) .

Περίληψη: Η δημιουργία ενός **Applet** γίνεται με το πάτημα του «κουμπιού» **New Class** απ'το μενού και με την επιλογή **Applet** σαν **Class Type**.

9.3 Δοκιμάζοντας ένα Applet (Testing)

Μερικές φορές είναι χρήσιμο να κατασκευάζουμε αντικείμενα προερχόμενα από **Applet** όπως ακριβώς κάνουμε και για τις κλάσεις . Έτσι με δεξί κλικ πάνω στην κλάση-applet προκύπτει ένα μενού. Αν επιλέξουμε **new...** δημιουργείται στην περιοχή αντικειμένων ένα αντικείμενο της κλάσης αυτής. Από το αντικείμενο αυτό δεν μπορούμε να εκτελέσουμε το συνολικό **applet** , αλλά μπορούμε να καλέσουμε κάποιες μεθόδους . Έτσι μας βοηθάει να τεστάρουμε τις επιμέρους μεθόδους που μπορεί να έχουμε γράψει σαν τμήμα της εφαρμογής μας .

10. ΑΛΛΕΣ ΛΕΙΤΟΥΡΓΙΕΣ

10.1 Άνοιγμα ενός project που δεν ανήκει στο BlueJ με το BlueJ

Το **BlueJ** μας επιτρέπει να ανοίξουμε project, τα οποία δημιουργήθηκαν και έχουν αποθηκευτεί εκτός **BlueJ**. Για το σκοπό αυτό επιλέγουμε **Project => Open Non BlueJ...** από το μενού και στη συνέχεια στο παράθυρο που ανοίγει πηγαίνουμε στον κατάλογο που έχουμε αποθηκεύσει το αρχείο μας, το επιλέγουμε και πατάμε το «κουμπί» **Open in BlueJ**. Το **BlueJ** πριν ανοίξει το αρχείο θα μας ζητήσει επιβεβαίωση για να συνεχίσει .

Περίληψη: Για να ανοίξουμε **Non-BlueJ project** επιλέγουμε **Project => Open Non BlueJ...** απ' το μενού .

10.2 Προσθήκη κλάσης σε ένα project

Υπάρχουν φορές που θέλουμε να χρησιμοποιήσουμε μια έτοιμη κλάση που έχουμε ήδη αποθηκεύσει κάπου αλλού σε κάποιο **project** μας . Για παράδειγμα μπορεί κάποιος καθηγητής να δώσει στους φοιτητές του μια κλάση να την συμπεριλάβουν στο **project** τους . Αυτό γίνεται εύκολα από το μενού του **BlueJ** με την επιλογή **Edit=>Add Class from File...** Η λειτουργία αυτή μας επιτρέπει να εισάγουμε την κλάση που θέλουμε (αρκεί να έχει την κατάληξη **.java**) στο **project** μας όπου δημιουργείται πλέον ένα

αντίγραφο της κλάσης αυτής και αποθηκεύεται στον κατάλογο του *project* μας . Η κλάση αυτή έχει ακριβώς την ίδια λειτουργία σαν να την είχαμε δημιουργήσει και να είχαμε γράψει τον κώδικά της απ' την αρχή. Να σημειώσουμε επίσης ότι η κλάση που θέλουμε να προσθέσουμε δεν είναι απαραίτητο να προέρχεται από *project* του **BlueJ** , αλλά από οποιοδήποτε άλλο κατάλογο. Έτσι λοιπόν , αφού έχουμε προσθέσει μια κλάση σύμφωνα με τα παραπάνω , την επόμενη φορά που θα ανοίξουμε το *project* μας θα δούμε ότι θα περιλαμβάνει και τη νέα κλάση .

Περίληψη: Η προσθήκη μιας έτοιμης κλάσης σε ένα *project* γίνεται με την επιλογή του μενού **Edit => Add Class from File...**

10.3 Κλήση της *main* και άλλων *static* μεθόδων

Ανοίγουμε το *project hello* από τον κατάλογο *examples* του **BlueJ** και παρατηρούμε ότι η μοναδική κλάση (*hello*) του *project* ορίζει μια στάνταρ *main* μέθοδο. Με δεξί κλικ πάνω στην κλάση, στο μενού που προκύπτει εκτός από την επιλογή για δημιουργία νέου αντικειμένου (*new Hello()*) υπάρχει και αυτή της *static* μεθόδου *main* (*void main(args)*). Καλούμε λοιπόν τη *main* απ' ευθείας από το μενού (χωρίς να χρειάζεται να δημιουργήσουμε πρώτα ένα αντικείμενο της κλάσης αυτής , όπως θα κάναμε για οποιαδήποτε άλλη *static* μέθοδο) . Το παράθυρο που ανοίγει μας ζητάει να καθορίσουμε τα ορίσματα του πίνακα της *main*. Τα ορίσματα αυτά πρέπει να συμφωνούν με τη σύνταξη των εντολών της *Java* και να περιέχονται μεταξύ ("). Έτσι για παράδειγμα μπορείτε να δώσετε τα παρακάτω ορίσματα :

```
{"one", "two", "three"}
```

ΠΡΟΣΟΧΗ: Σύμφωνα με τους κανόνες της *Java* οι σταθερές ενός πίνακα δεν μπορούν να χρησιμοποιηθούν ως πραγματικά ορίσματα κατά την κλήση μεθόδων , αλλά χρησιμοποιούνται μόνο για αρχικοποίηση . Στο **BlueJ** , όμως , για να πραγματοποιούμε *interactive* κλήσεις *main* μεθόδων , επιτρέπεται να περνάμε σταθερές σαν παραμέτρους-ορίσματα.

Περίληψη: Η κλήση μιας *Static* μεθόδου στο **BlueJ** μπορεί να γίνει απ' ευθείας από το μενού της αντίστοιχης κλάσης .

10.4 Δουλεύοντας με βιβλιοθήκες

Συχνά όταν γράφουμε ένα πρόγραμμα σε *Java* χρειάζεται να ορίσουμε τις αντίστοιχες *Java* βιβλιοθήκες (*libraries*) . Αν επιλέξετε από το μενού του **BlueJ** , **Help => Java Standard Classes** θα οδηγηθείτε στο site της *Java* στο διαδίκτυο όπου μπορείτε να κατεβάσετε το **JDK API** , ώστε να το εγκαταστήσετε και να το χρησιμοποιείτε όταν βρίσκεστε offline . Περαιτέρω πληροφορίες μπορείτε να πάρετε από την επιλογή **Help => BlueJ reference manual** .

Περίληψη:Μεσώ του *Help* του *BlueJ* μπορούμε να βρούμε στο διαδίκτυο όποια έκδοση του *JDK API* θέλουμε καθώς και τις αντίστοιχες βιβλιοθήκες .