Tutorialul pentru BlueJ

Versiunea 2.0.1

pentru BlueJ Versiunea 2.0.x

Michael Kölling
Mærsk Institute
University of Southern Denmark

Versiune în limba românã - Marius Joldos - Universitatea Tehnicã din Cluj-Napoca

Cuprins

1 Cuvânt înainte

1.1 Despre BlueJ

Acest tutorial reprezintã o introducere în folosirea IDE BlueJ. BlueJ reprezintã un mediu de dezvoltare Java™ anume proiectat pentru învatare de nivel introductiv. BlueJ a fost proiectat si implementat de catre echipa BlueJ de la Deakin University, Melbourne, Australia, si de la University of Kent at Canterbury, UK.

Informatii suplimentare despre BlueJ sunt disponibile la **http://www.bluej.org.**

1.2 Scop si audientã

Acest tutorial este destinat celor care doresc sã se familiarizeze cu capabilitãtile mediului. El nu explicã deciziile care stau la baza constructiei mediului nici considerentele legate de cercetare care stau în spatele acestuia.

Acest tutorial nu este menit a învãta pe cineva Java. Începatorii in programarea în Java sunt sfãtuiti sã studieze si o carte introductivã in Java sau sã urmeze un curs de Java.

De asemenea, acesta nu este un manual de referintã cuprinzator. Multe detalii sunt omise – accentul este pe introducerea scurtã si concisã si nu pe acoperirea completã a caracteristicilor. Pentru referinte detaliate, vezi *The BlueJ Environment ReferenceManual,* disponibil pe situl de web al BlueJ (**www.bluej.org**).

Fiecare sectiune începe cu o propozitie rezumativã de o linie. Aceasta permite utilizatorilor deja familiarizati cu pãrtile sistemului sã decida dacã doresc sã citeascã sau sã sarã o anumitã sectiune. Sectiunea 11 repetã doar liniile de rezumat, spre referintã.

1.3 Copyright, licentiere si redistribuire

1.4 Feedback

Comentarii, întrebari, corectii, critici si orice ale reactii privind sistemul BlueJ sau acest tutorial sunt binevenite si încurajate activ. Vã rugãm sã le expediati lui Michael Kölling (mik@mip.sdu.dk).

2 Instalarea

BlueJ este distribuit în trei formate diferite: unul pentru sisteme Window, un altul pentru MacOS si un altul pentru toate celelalte systeme. Instalarea este directã.

Cerinte

Trebuie sã aveti J2SE v1.4 (JDK 1.4) sau mai nouã instalatã pe sistem ca sã folositi BlueJ. În general, actualizarea la ultima versiune stabilã (non-beta) de Java este recomandatã. Dacã nu aveti JDK instalat îl puteti descãrca de pe situl web al Sun Corp. http://java.sun.com/j2se / . Pe MacOS X, este preinstalatã o versiune recentã de J2SE - nu trebuie sa o instalati singur(a). Dacã gãsiti pagini care oferã "JRE" (Java Runtime Environment) si "SDK" (Software Development Kit), trebuie sã descãrcati "SDK" – JRE nu este suficient.

2.1 Instalarea sub Windows

Fisierul cu distributia pentru sistemele Windows se numeste *bluejsetup-xxx.exe*, unde *xxx* este numãrul versiunii. Spre exemplu, distributia BlueJ versiunea 2.0.0 se numeste *bluejsetup-200.exe*. Puteti primi acest fisier pe disc sau îl puteti descãrca de pe situl BlueJ la adresa http://www.bluej.org . Executati acest program de instalare.

Programul de instalare vã permite sã alegeti un director destinatie. De asemenea vã va oferi optiunea de instalare a unei scurtãturi in meniul Start si pe desktop.

Dupã terminarea instalarii, veti gãsi programul *bluej.exe* în directorul de instalare al BlueJ.

La prima lansare a BlueJ, programul va cãuta un sistem de dezvoltare Java (JDK). Dacã gãseste mai mult de un sistem Java corespunzãtor (d.e. aveti JDK 1.4.2 si JDK 1.5.0 instalate), vi se va permite sã alegeti unul printr-un dialog. Dacã nu gaseste nici unul vi se va cere sã-l localizati singuri (acest lucru se poate întâmpla atunci când a fost instalat un JDK, dar intrãrile corespunzatoare din registry au fost înlãturate).

Instalatorul BlueJ instaleazã si un program numit *vmselect.exe*. Folosind acest program puteti schimba ulterior versiunea de Java pe care o foloseste BlueJ. Executati *vmselect* pentru a lansa BlueJ cu o versiune diferitã de Java.

Alegerea fãcutã pentru JDK este stocatã pentru fiecare versiune de BlueJ. Dacã aveti instalate versiuni diferite de BlueJ, atunci puteti folosi o versiune de BlueJ cu JDK 1.4.2 si o alta cu JDK 1.5. Schimbarea versiunii de Java pentru BlueJ va efectua aceasta schimbare pentru toate instalãrile de BlueJ de aceeasi versiune, pentru acelasi utilizator.

2.2 Instalarea pe Macintosh

BlueJ ruleaza numai pe MacOS X.

Distributia pentru MacOS este in fisierul numit *BlueJ-xxx.zip*, unde *xxx* este un numãr de versiune. Spre exemplu, distributia BlueJ versiunea 2.0.0 se numeste *BlueJ-200.zip*. Acest fisier poate fi distribuit pe disc sau îl puteti descãrca de pe web, la http://www.bluej.org.

MacOS va decomprima (de obicei) automat acest fisier dupã descãrcare. Dacã nu, faceti dublu-clic pe el pentru decomprimare.

Dupã decomprimare, veti avea un director numit BlueJ-*xxx*. Mutati acest director în directorul numit Applications (sau unde doriti sa-l pastrati). Nu mai sunt necesari alti pasi în instalare.

2.3 Instalarea pe Linux/Unix si alte sisteme

Fisierul de distributie general pentru acestea este un fisier jar executabil. Fisierul e numit *bluej-xxx.jar,* unde *xxx* este un numãr de versiune. De exemplu, distributia BlueJ versiunea 2.0.0 este numita *bluej-200.jar*. Fisierul îl puteti primi pe disc sau îl puteti descãrca de pe situl BlueJ la adresa http://www.bluej.org

Rulati programul de instalare executând comanda urmãtoare. Obs.: Pentru acest exemplu am folosit distributia *bluej-200.jar* – Dvs. folositi numele de fisier pe care îl aveti (cu numãrul de versiune corect).

```
<j2se-path>/bin/java -jar bluej-200.jar
```

<*j2se-path*> este directorul în care a fost instalat J2SE SDK.

Va apãrea o fereastrã care vã permite sã alegeti directorul de instalare pentru BlueJ si versiunea de Java pe care o folositi pentru a rula BlueJ.

Dati clic pe *Install*. Dupã terminare, BlueJ are trebui sã fie instalat.

2.4 Probleme legate de instalare

Dacã aveti probleme verificati lista de întrebãri frecvente - *Frequently Asked Questions* (FAQ) de pe situl de web BlueJ (http://www.bluej.org/help/faq.html) si cititi sectiunea "Cum sã ceri ajutor" (*How To Ask For Help* - http://www.bluej.org/help/ask-help.html).

3 Cum se începe – editare / compilare / executie

3.1 Pornirea BlueJ

Pe Windows si MacOS, este instalat un program numit *BlueJ*. Lansati-l.

Pe sistemele Unix instalatorul instalează un scenariu numit *bluej* în directorul de instalare. Pentru interfete GUI, faceti doar dublu-clic pe iconita *BlueJ*. În linie de comandă, pentru lansarea BlueJ cu sau fără un proiect ca argument:

```
$ bluej
```

sau

```
$ bluej examples/people
```

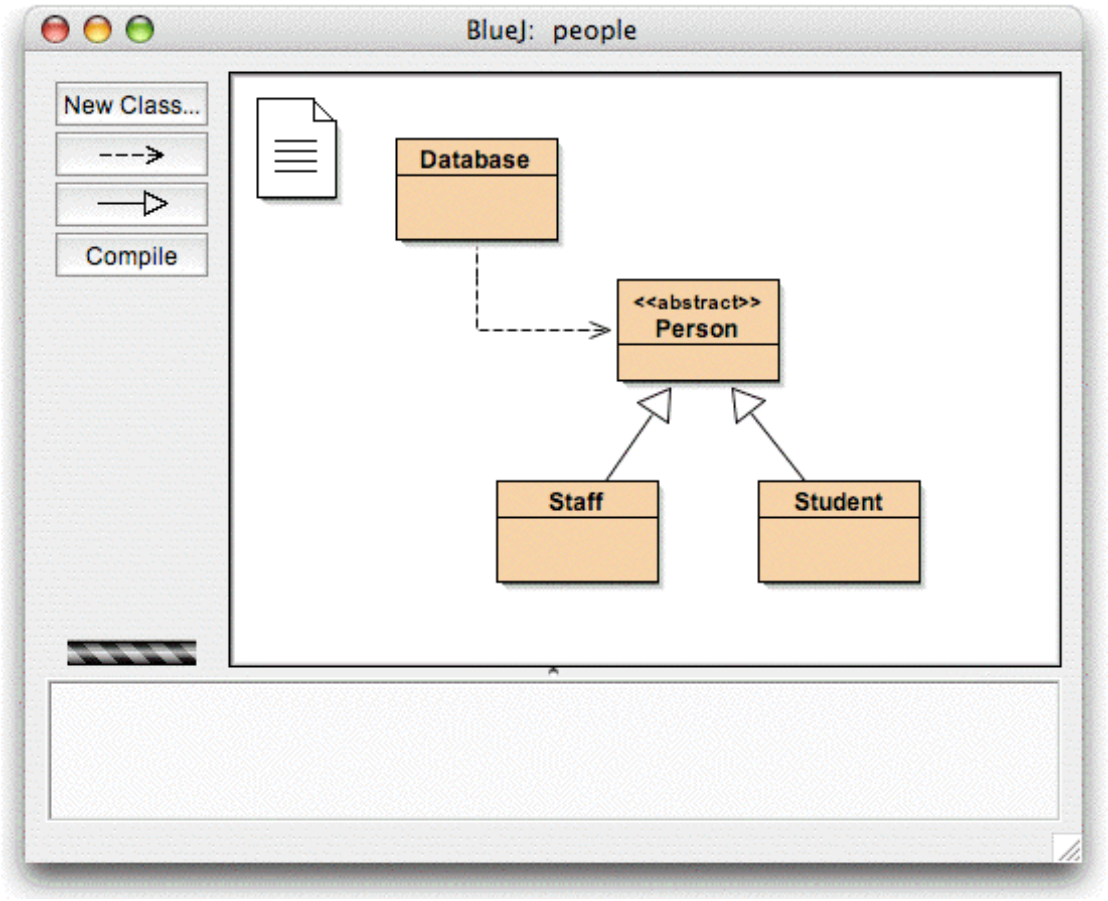

Figura 1: Fereastra principală a BlueJ

3.2 Deschiderea unui proiect

Rezumat: Pentru a deschide un proiect, alegeti Open *din meniul* Project*.*

Proiectele BlueJ, la fel ca pachetele Java standard, sunt directoare care contin fisierele incluse în proiect.

Dupã lansarea BlueJ, folositi comanda Project – Open... (din meniul Project) pentru a alege si a deschide un proiect.

Câteva exemple de proiecte sunt incluse în distributia standard BlueJ în directorul *examples*.

Pentru aceastã sectiune a tutorialului, deschideti projectul *people*, inclus în acest director. Directorul *examples* îl puteti gãsi în directorul 'home' al BlueJ. Dupã deschiderea proiectului, trebuie sã aparã o fereastra similarã celei din figura 1. Fereastra poate sã nu arate identic pe sistemul Dvs., dar diferentele ar trebui sã fie minore.

3.3 Crearea de obiecte

Rezumat: Pentru a crea un obiect, alegeti un constructor din meniul vertical al clasei.

Una dintre caracteristicile fundamentale ale BlueJ este nu doar cã puteti executa o aplicatie completã, dar si cã puteti interactiona direct cu obiecte singure ale oricarei clase si puteti executa metode publice ale clasei respective. O executie in BlueJ este de obicei efectuatã prin crearea unui obiect si apoi invocarea uneia dintre metodele obiectului. Acest lucru este foarte util în dezvoltarea unei aplicatii – vã puteti testa clasele individual, de îndatã ce au fost scrise. Nu e nevoie sã scrieti mai întâi întreaga aplicatie.

> Notã*: Metodele statice pot fi executate direct fãrã a crea mai întâi un obiect. Una dintre metodele statice poate fi "main", astfel cã putem face acelasi lucru care se întâmplã în mod normal în aplicatiile Java – pornirea unei aplicatii prin executarea unei metode main statice. Vom reveni asupra acestui lucru. Mai întâi vom face alte lucruri mai interesante care nu pot fi facute în mod normal în mediile Java.*

Pãtratele pe care le vedeti în partea centralã a ferestrei principale (etichetatã *Database*, *Person*, *Staff* si *Student*) sunt icoane care reprezintã clasele implicate în aceastã aplicatie. Puteti obtine meniul cu operatiile aplicabile unei clase prin clic pe icoana clasei cu butonul din dreapta al mouse (Macintosh: ctrl-clic[1]) (Figura 2). Operatiile prezentate sunt operatiile *new* cu fiecare dintre constructorii definiti pentru aceastã clasã (mai întâi) urmate de câteva operatii furnizate de mediu.

[1] Ori de cate ori mentionam un clic-dreapta in acest tutorial, utilizatorii de Macintosh vor citi *ctrl-clic*.

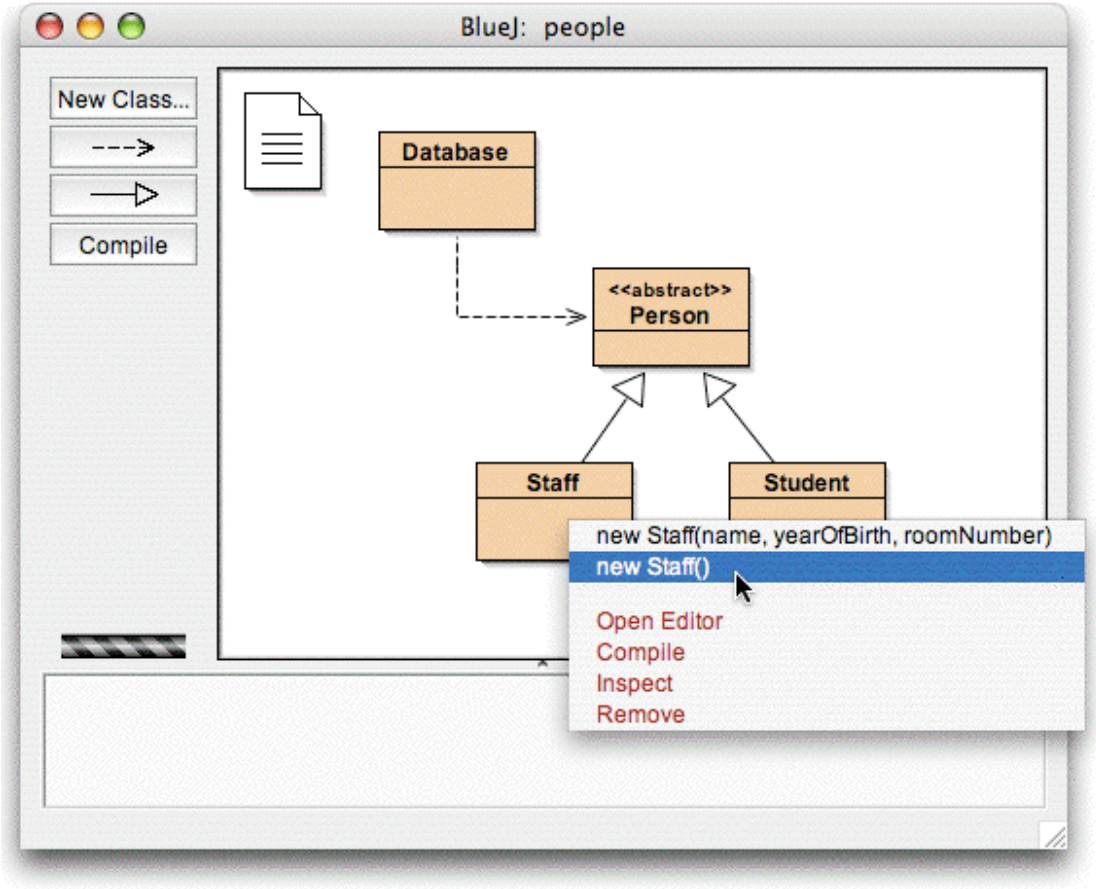

Figura 2: Operatii asupra unei clase (meniu vertical)

Dorim sã creãm un obiect *Staff*, asa cã trebuie clic-dreapta pe icoana *Staff* (care face sã aparã meniul prezentat în figura 2). Meniul prezintã doi constructori pentru crearea unui obiect *Staff*, unul cu parametri si altul fãrã. Mai întâi, alegeti constructorul fãrã parametri. Apare dialogul din figura 3.

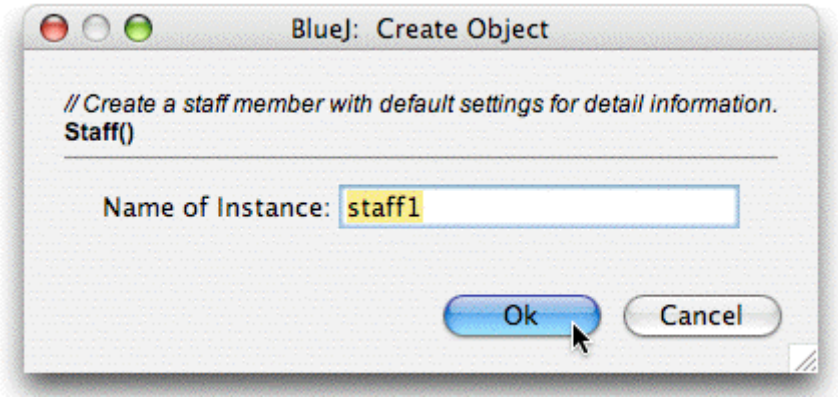

Figura 3: Crearea unui obiect fãrã parametri

Acest dialog vã solicitã sã dati un nume obiectului de creat. În acelasi timp, vã sugereazã un nume implicit (*staff1*). Acest nume este suficient de bun deocamdatã, asa cã clic pe *OK*. Va fi creat un obiect *Staff*.

O datã ce obiectul a fost creat, el este plasat pe bancul de obiecte (Figura 4). Asta e tot ce existã la crearea obiectelor: se alege un constructor din meniul clasei, se executã si aveti obiectul pe bancul de obiecte.

Figura 4: Un obiect pe bancul de obiecte

Probabil cã ati observat cã clasa *Person* este etichetatã <<abstract>> (este o clasã abstractã) Veti observa (dacã încercati) cã nu puteti crea obiecte din clase abstracte (asa si defineste specificatia limbajului Java).

3.4 Executia

Rezumat: Pentru a executa o metodã, alegeti-o din meniul vertical al obiectului.

Acum cã ati creat un obiect, îi puteti executa operatiile publice. (Java numeste operatiile *metode*.) Clic cu butonul din dreapta pe obiect si apare un meniu cu operatiile obiectului (Figura 5). Meniul aratã metodele disponibile pentru acest obiect si încã douã operatii speciale furnizate de mediu (*Inspect* si *Remove* pe care le vom trata mai tarziu).
Acum sã ne concentrãm asupra metodelor.

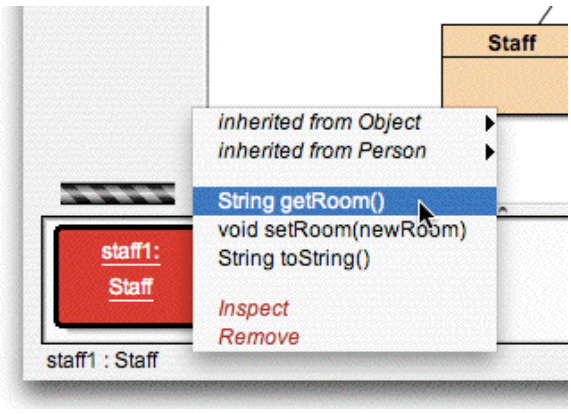

Figura 5: Meniul vertical al obiectului

Vedeti cã existã metodele *setRoom* si *getRoom* care stabilesc si returneaza numãrul biroului (camerei) pentru acest membru al personalului. Incercati sa apelati *getRoom*. Pur si simplu alegeti metoda din meniu si ea se va executa. Va apãrea un dialog ca-

re aratã rezultatul executiei metodei (Figura 6). În acest caz, numele spune "(unknown room)" , adicã birou necunoscut pentru cã nu am precizat un birou pentru aceastã persoanã.

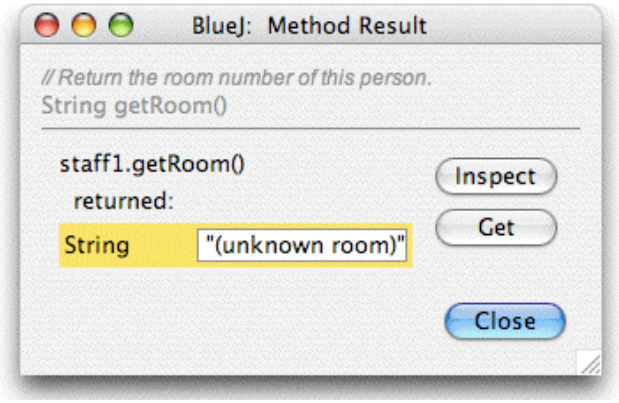

Figura 6: Afisarea rezultatului unei functii

Metodele mostenite dintr-o superclasã sunt disponibile prin intermediul unui sub-meniu. În partea cea mai de sus a meniului vertical al obiectului existã doua sub-meniuri, unul pentru metodele mostenite de la *Object* si un altul pentru cele de la *Person* (Figura 5). Puteti apela metode ale *Person*, cum este *getName*) alegându-le din sub-menu. Încercati. Veti observa cã rãspunsul este iarasi vag: "(unknown name)", deoarece nu am dat un nume persoanei noastre.

Acum sã încercãm sã specificãm un numãr de birou. Aceasta ne va arãta cum sã apelãm o metodã cu parametri. (Apelurile la *getRoom* si *getName* au valori de retur, dar nu au parametri). Apelati functia *setRoom* alegând-o din meniu. Apare un dialog care vã solicitã sã introduceti un parametru (Figura 7).

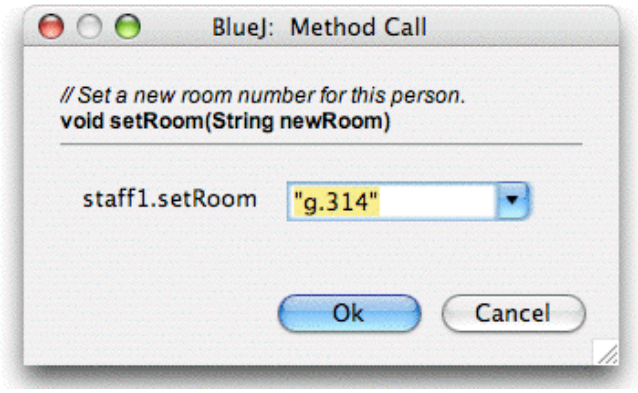

Figura 7: Dialog pentru apelul unei functii cu parametri

În partea de sus, acest dialog aratã interfata metodei de apelat (inclusiv comentariile si semnãtura). Sub acestea se afla o cãsutã de text unde puteti introduce parametrul. Semnãtura de sus ne spune cã se asteapta un parametru de tipul String. Introduceti noul numãr de birou ca sir (inclusiv ghilimelele) în campul de text si apãsati butonul *OK* al formei.

Asta e tot – de vreme ce aceasta metodã nu returneazã vreun parametru, nu existã dialog rezultat. Apelati *getRoom* din nou pentru a verifica faptul cã identificarea biroului s-a schimbat.

Exersati un timp crearea de obiecte si apelul metodelor. Încercati sã apelati un construc- cu argumente si apelati încã câteva metode pânã vã familiarizati cu aceste operatii.

3.5 Editarea unei clase

Rezumat: Pentru a edita sursa unei clase, dati dublu-clic pe icoana clasei.

Pâna acum am tratat doar interfata unui obiect. Acum este momentul sã privim înãuntru. Puteti vedea implementarea unei clase alegând *Open Editor* din operatiile clasei (Aducere-aminte: prin clic-dreapta pe icoana unei clase se relevã operatiile ei.) Dublu-click pe icoana clasei este o scurtatura spre aceeasi functie. Editorul nu este descris în detaliu în acest tutorial, dar este foarte usor de folosit.
Detaliile editorului vor fi descrise separat mai tarziu. Deocamdatã, deschideti implementarea clasei *Staff*. Localizati implementarea metodei *getRoom*. Aceasta returneazã, dupã cum îi sugereaza numele, numãrul biroului unui membru al personalului. Sã schimbãm metoda adãugând prefixul "`room`" la rezultatul functiei, (astfel ca metoda sã returneze, sã zicem, "room M.3.18" în loc de doar "M.3.18"). Putem face acest lucru schimbând linia

```
return room;
```

în

```
return "room " + room;
```

BlueJ suportã Java fãrã nici o modificare, asa cã nu exista nimic special modul în care sã vã implementati clasele.

3.6 Compilarea

Rezumat: Pentru a compila o clasã, apãsati butonul Compile *al editorului. Pentru a compila un proiect, apãsati butonul* Compile *din fereastra proiectului.*

Dupã inserarea textului (înainte de a face orice altceva), verificati rezumatul proiectului în fereastra principala). Veti observa cã icoana de clasã pentru clasa *Staff* s-a schimbat: acum este în dungi. Marcajul în dungi al claselor aratã cã acestea nu au fost compilate de la ultima schimbare. Înapoi la editor.

> Notã*: Vã puteti întreba de ce icoanele claselor nu au fost dungate la prima deschidere a acestui proiect. Acest lucru nu s-a întâmplat fiindcã clasele din proiect erau deja compilate. Adesea proiectele BlueJ sunt distribuite necompilate, asa cã sã vã asteptati sã vedeti dungate majoritatea icoanelor claselor la prima deschidere a unui proiect de acum incolo.*

În bara de unelte din partea de sus a editorului existã câteva butoane cu functii folosite frecvent. Unul este *Compile*. Aceastã functie permite compilarea acestei clase direct din editor. Acum dati clic pe butonul *Compile*. Dacã nu ati gresit, ar trebui sã aparã

un mesaj în zona de informatii de la baza editorului care sã vã notifice compilarea clasei. Dacã ati comis o gresealã care duce la o eroare de sintaxã, atunci linia cu eroare va fi evidentiatã si se va afisa un mesaj de eroare în zona de informatii. (În cazul în care compilarea a reusit de la început, încercati sã introduceti o eroare de sintaxã acum – d.e. un punct si virgula lipsã – si compilati din nou, doar pentru a vedea cum aratã).

După compilarea cu succes a clasei, închideti editorul.

> Notã*: Nu este nevoie sã salvati explicit sursa clasei. Sursele sunt salvate automat ori de cate ori este nevoie (d.e. la închiderea editorului sau înainte de compilarea clasei respective). Puteti cere explicit salvarea dacã doriti (existã o functie în meniul* Class *al editorului), dar acest lucru este într-adevãr necesar doar dacã sistemul Dvs. este instabil si crapã frecvent si vã este teamã sã nu vã pierdeti munca.*

Bara de unelte din fereastra peoiectului are si ea un buton *Compile*. Acest buton cauzeazã compilarea întregului proiect. (De fapt, el determinã care clase necesitã recompilarea si le recompileazã în ordinea corespunzãtoare.) Încercati acest lucru prin schimbarea uneia sau mai multor clase (astfel ca douã sau mai multe clase apar dungate în diagrama de clase) si apoi apãsati butonul *Compile*. Dacã se detecteaza o eroare într-una dintre clasele compilate, atunci se va deschide automat editorul si se va afisa locul erorii si un mesaj. Puteti observa cã bancul de obiecte este din nou gol. Obiectele sunt înlãturate de fiecare datã când se schimba implementarea.

3.7 Ajutor la erorile de compilare

Rezumat: Pentru a obtine ajutor pentru un mesaj de eroare al compilatorului dati clic pe semnul de întrebare de lângã mesajul de eroare.

Foarte frecvent, studentii începãtori au dificultãti în a întelege mesajele de eroare date de compilator. Vom încerca sa oferim ceva ajutor.

Deschideti din nou editorul, introduceti o eroare în fisierul sursã si compilati. Se va afisa un mesaj de eroare în zona de informatii a editorului. La capãtul din dreapta al zonei de informatii apare un semn de întrebare pe care puteti da clic pentru a obtine ceva mai multe informatii despre acest gen de eroare (Figura 8).

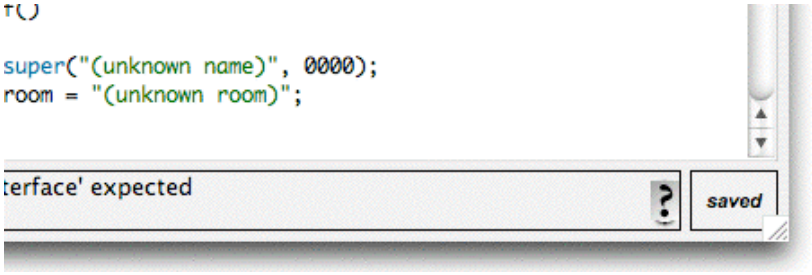

Figura 8: O eroare de compilare si butonul *Help*

În aceastã etapã nu sunt disponibile texte de ajutor pentru toate mesajele de eroare. Mai sunt texte ajutatoare care trebuie scrise. Dar merita încercat – multe erori sunt deja explicate. Cele care mai rãmân vor fi scrise si incluse in versiuni viitoare ale BlueJ.

4 Un pic mai mult...

În această sectiune vom parcurge alte câteva lucruri care le puteti face în mediu. Lucruri care nu sunt esentiale, dar sunt foarte comun folosite.

4.1 Inspectia

Rezumat: Inspectia obiectelor permite un anumit nivel de depanare prin afisarea stării interne a unui obiect.

Atunci când ati executat metodele unui obiect, poate ca ati remarcat operatia *Inspect* care este disponibilă pentru obiecte, pe lângă metodele definite de utilizator (Figura 5). Această operatie permite verificarea stării variabilelor de instantă ("câmpuri") ale obiectelor. Încercati să creati un obiect cu câteva valori definite de utilizator (d.e. un obiect *Staff* folosind constructorul cu parametri). Apoi selectati *Inspect* din meniul objectului. Apare un dialog care afiseazã câmpurile obiectului, tipurile si valorile acestora (Figura 9).

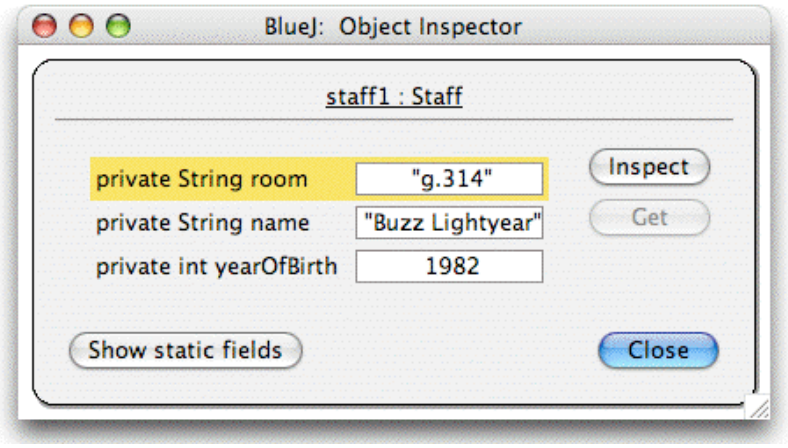

Figure 9: Dialogul pentru inspectie

Inspectia este utilă pentru a verifica rapid dacă o operatie 'mutator' (o operatie care schimba starea obiectului) a fost executată corect. Astfel, inspectia este o unealtă de depanare simplă.

În exemplul *Staff*, toate câmpurile sunt tipuri simple (fie tipuri non-obiect fie siruri). Valorile acestor tipuri pot fi afisate direct. Puteti vedea imediat dacă constructorul a efectuat atribuirile corespunzătoare.

În cazuri mai complexe, valorile câmpurilor ar putea fi referinte la obiecte definite de utilizator. Pentru a ne uita la un asemenea exemplu, vom folosi un alt proiect. Deschideti proiectul *people2*, care este si el inclus in distributia standard a BlueJ. Desktop-ul pentru *people2* este prezentat în Figura 10. Cum se vede, acest al doilea

examplu are o clasã *Address* pe lângã clasele vãzute anterior. Unul dintre câmpuri din clasa *Person* este de tipul *Address*, definit de utilizator.

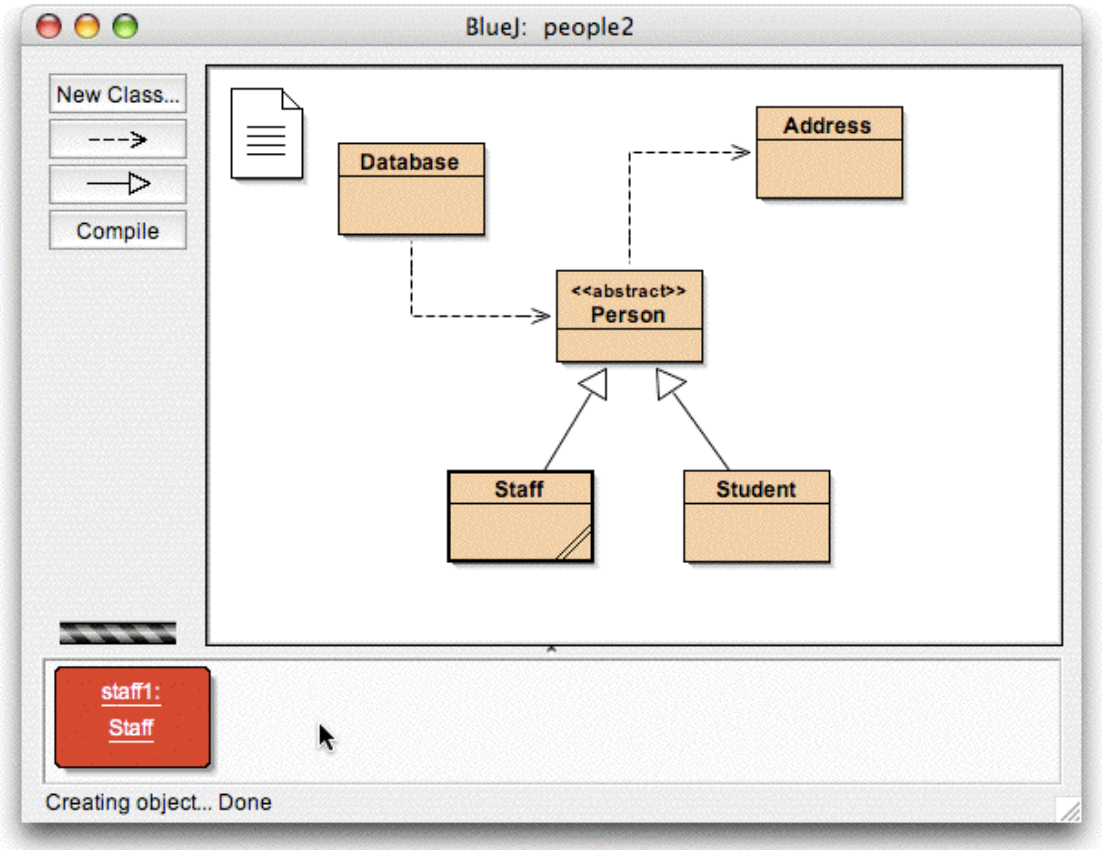

Figura 10: Fereastra proiectului *people2*

Pentru urmatorul lucru pe care dorim sã-l încercãm – inspectia cu câmpuri obiect – creati un obiect *Staff* si apoi apelati metoda *setAddress* pentru acest obiect (o veti gãsi în submeniul *Person*). Introduceti adresa. Intern, codul lui *Staff* creazã un obiect de clasã *Address* si-l memoreazã în câmpul sãu *address*.

Inspectati acum obiectul *Staff*. Dialogul de inspectie rezultat este arãtat în Figura 11. Câmpurile din obiectul *Staff* includ acum *address*. Dupã cum se vede, valoarea sa este arãtatã sub forma unei sãgeti, lucru care semnificã o referintã la alt obiect. Cum acesta este un obiect complex, definit de utilizator, valoarea sa nu poate fi arãtatã direct în aceastã listã. Pentru a examina mai departe adresa, selectati câmpul *address* field din lista si dati clic pe butonul *Inspect* din dialog. (Sau puteti da dublu-click pe câmpul *address*.) Se deschide o altã fereastrã de inspectie care aratã detaliile obiectului *Address* (Figura 12).

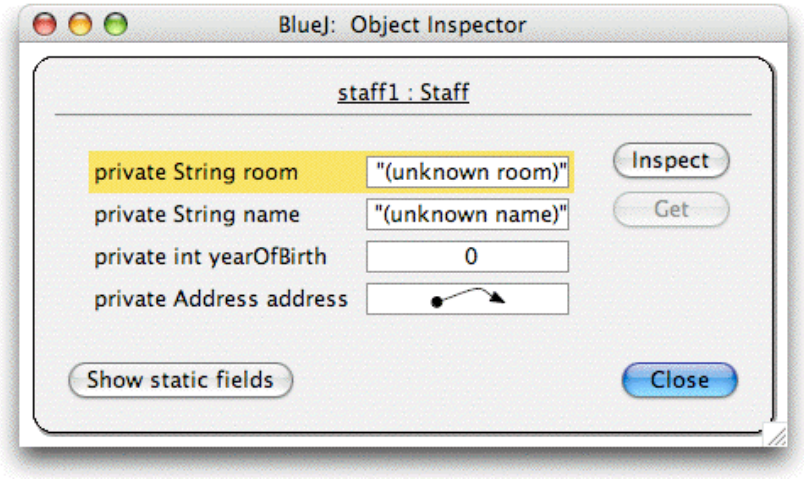

Figura 11: Inspectia cu referintã la un obiect

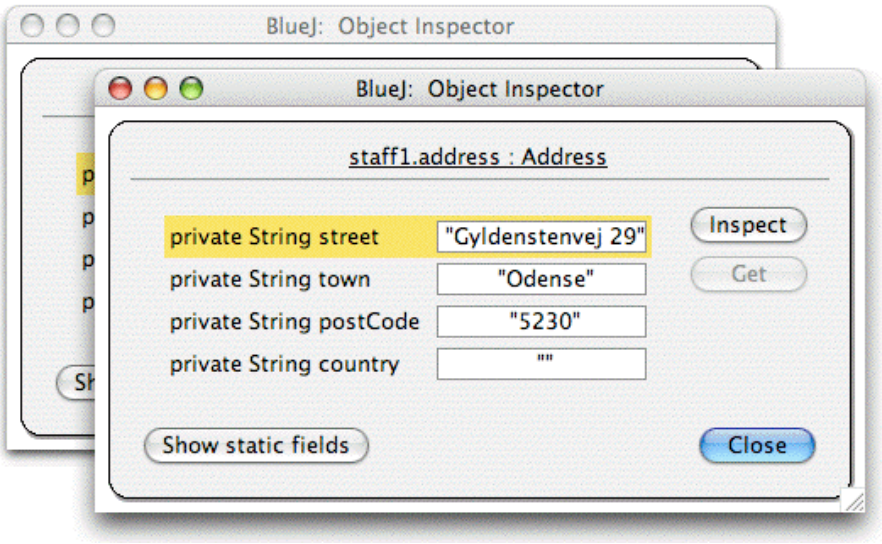

Figure 12: Inspectia obiectului intern

Dacã ati selectat un câmp public, atunci, în loc sã dati clic pe *Inspect,* puteti alege câmpul address si sã apasati butonul *Get*. Aceaatã operatie plaseazã obiectul ales pe bancul de obiecte. Acolo îl puteti examina mai departe prin apelarea metodelor sale.

4.2 Transmiterea obiectelor ca parametri

Rezumat: Un obiect poate fi transmis ca parametru unei metode prin click pe icoana obiectului.

Obiectele pot fi transmise ca parametri spre metodele altor obiecte. Sã încercãm un exemplu. Creati un obiect de clasa *Database*. (Veti observa cã clasa *Database* are doar

un constructor care nu are nici un parametru, asa cã obiectul se construieste direct.) Obiectul *Database* are abilitatea de a pãstra o listã de persoane. El posedã operatii pentru adãugarea de obiecte persoanã si pentru a afisa toate persoanele stocate la un moment dat. Faptul ca este numit *Database* este un pic exagerat!)

Dacã nu aveti deja un obiect *Staff* sau *Student* pe bancul de obiecte, creati si unul de acel tip. Pentru cele ce urmeazã aveti nevoie de un obiect *Database* si de un obiect *Staff* sau ***Student*** pe bancul de obiecte în acelasi timp.

Apelati acum metoda *addPerson* a obiectului *Database*. Semnãtura vã spune cã se asteaptã un parametru de tipul *Person*. (Amintiti-vã: clasa *Person* este abstractã, asa cã nu existã obiecte care sã fie direct de tipul *Person*. Dar, datoritã subtipãrii, obiectele de tipul *Student* si *Staff* pot substitui obiecte de tipul ***Person***. Asa cã este legal sã se transmitã un *Student* sau un *Staff* acolo unde se asteaptã un obiect *Person*.) Pentru a transmite obiectul de pe bancul de lucru în apelul în curs, ati putea sã-i introduceti numele în câmpul de parametru, sau, ca scurtãturã, doar sã dati clic pe obiect. Acest lucru introduce numele sãu în dialogul de apel al metodei. Clic *OK* si apelul este efectuat. De vreme ce nu existã valoare returnatã pentru aceastã metodã, nu vedem imediat un rezultat. Puteti invoca metoda ***listAll*** pe obiectul *Database* pentru a vedea dacã operatia a fost într-adevãr efectuatã. Operatia *listAll* scrie informatiille persoanei la iesirea standard. Vet observa cã se deschide automat un terminal în modul text pentru a afisa textul respectiv.

Încercati asta din nou cu mai mult de o persoanã în "baza de date".

5 Crearea unui proiect nou

Acest capitol vã face un scurt tur al modului de setare a unui proiect nou.

5.1 Crearea directorului proiectului

Rezumat: Pentru a crea un proiect nou, alegeti New... *din meniul* Project.

Pentru a crea un proiect nou, alegeti Project – New... din meniu. Se deschide un dialog pentru alegerea unui fisier care vã permite sã specificati un nume si o locatie pentru noul proiect. Puteti alege orice nume pentru proiect. Dupã ce ati dat clic pe OK, se va crea un director cu numele specificat, iar fereastra principalã va arãta proiectul cel nou, gol.

5.2 Crearea claselor

Rezumat: Pentru a crea o clasã, faceti clic pe butonul New Class *si specificati numele clasei.*

Puteti crea clase prin clic pe butonul *New Class* de pe bara de unelte Project. Vi se va cere sã furnizati un nume pentru clasã - acest nume trebuie sã fie un identificator Java valid.

De asemenea puteti alege dintre patru feluri de clase: abstractã, interfatã, applet sau "standard". Aceastã alegere determinã ce schelet de cod este creat initial pentru clasa Dvs. Puteti schimba ulterior tipul clasei modificând codul sursã (spre exemplu, prin adaugarea cuvântului cheie "abstract" în cod).

Dupa crearea unei clase, aceasta este reprezentatã printr-o icoanã în diagramã. Dacã ea este o clasã standard, tipul (interfatã, abstract, sau applet) este indicat în icoana clasei. Atunci când deschideti editorul pentru o clasã nouã, veti observa cã s-a creat un schelet de clasã implicit - aceasta ar trebui sã usureze începutul. Codul implicit este corectlly sintactic. El poate fi compilat (dar nu face prea multe). Încercati sã creati câteva clase si sã le compilati.

5.3 Crearea dependentelor

Rezumat: Pentru a crea o sãgeatã, faceti clic pe butonul sãgeatã si târâti sãgeata în diagramã, sau pur si simplu scrieti codul în editor.

Diagrama claselor aratã dependentele între clase sub forma unor sãgeti. Relatiile de mostenire ("extinde" [extends] sau "implementeazã" [implements]) sunt vizualizate ca sãgeti cu vârful gol; relatiile "foloseste" (uses) sunt vizualizate ca linii întrerupte cu sãgeti cu vârful deschis. Puteti adauga dependente fie grafic (direct în diagramã) sau textual în codul sursã. Dacã adaugati grafic o sãgeatã, atunci sursa este automat actualizatã. Dacã adãugati o dependentã în sursã, se actualizeaza diagrama.

Pentru a adãuga o sãgeata grafic, dati clic pe butonul-sãgeatã corespunzãtor (sãgeatã cu vârful gol pentru "extends" sau "implements", sãgeatã cu linie întreruptã pentru "uses") si târâti sãgeata de la o clasa la alta.

Adãugarea unei sãgeti de mostenire inserează definitia "extends" sau "implements" în codul sursã al clasei (depinde dacã tinta este o clasa sau o interfatã).

Adãugarea unei sãgeti "uses" nu schimbã imediat sursa (dacã nu cumva tinta este o clasã din alt pachet. În acel caz, ea generează o instructiune "import", dar noi nu am vãzut asta încã în exemplele noastre). Dacã existã o sãgeatã "uses" în diagramã care indicã o clasã care nu este folositã în sursa clasei va genera ulterior un avertisment care va preciza cã s-a declarat o relatie "uses" spre o clasã, dar clasa nu este folositã niciodatã.

Adãugarea sãgetilor textual este facilã: doar tastati codul în mod normal. De îndatã ce clasa a fost salvatã, diagrama este actualizatã. (Si reamintiti-vã: închiderea editorului provoacã salvarea automatã.)

5.4 Înlãturarea de elemente

Rezumat: Pentru a înlãtura o clasã sau o sãgeatã, selectati functia Remove din meniul vertical

Pentru a elimina o clasã din diagramã, selectati clasa si apoi selectati *Remove* din meniul *Edit*. Puteti, de asemenea sã selectati *Remove* din meniul vertical al clasei. Ambele optiuni functionează si pentru sãgeti: puteti fie alege mai întâi sãgeata si apoi selecta *Remove* din meniu fie puteti folosi meniul vertical al sãgetii.

6 Folosirea tãblitei de cod

Tãblita de cod a BlueJ permite evaluarea rapidã si facilã a portiunilor arbitrare de cod Java (expresii si instructiuni). Astfel, tãblita de cod poate fi folositã pentru a investiga detalii ale semanticii limbajului Java si pentru a ilustra si experimenta sintaxa Java.

6.1 Afisarea tablitei de cod

Rezumat: Pentru a începe sã folositi tãblita de cod, alegeti Show Code Pad *din meniul* View.

Tãblita de cod nu este implicit vizibilã. Pentru a o face vizibilã, folositi ***Show Code Pad*** din meniul *View*. Acum fereastra principalã va include interfata tãblitei de cod în dreapta jos, în vecinãtatea bancului de obiecte (Figura 13). Atât limita orizontalã cât si cea verticalã a tãblitei de codprecum si cea a bancului de obiecte pot fi ajustate.

Zona tãblitei de cod poate fi acum folositã penru a introduce expresii sau instructiuni. La **apãsarea tastei** *Enter*, fiecare linie va fi evaluatã, iar un eventual rezultat poate fi afisat.

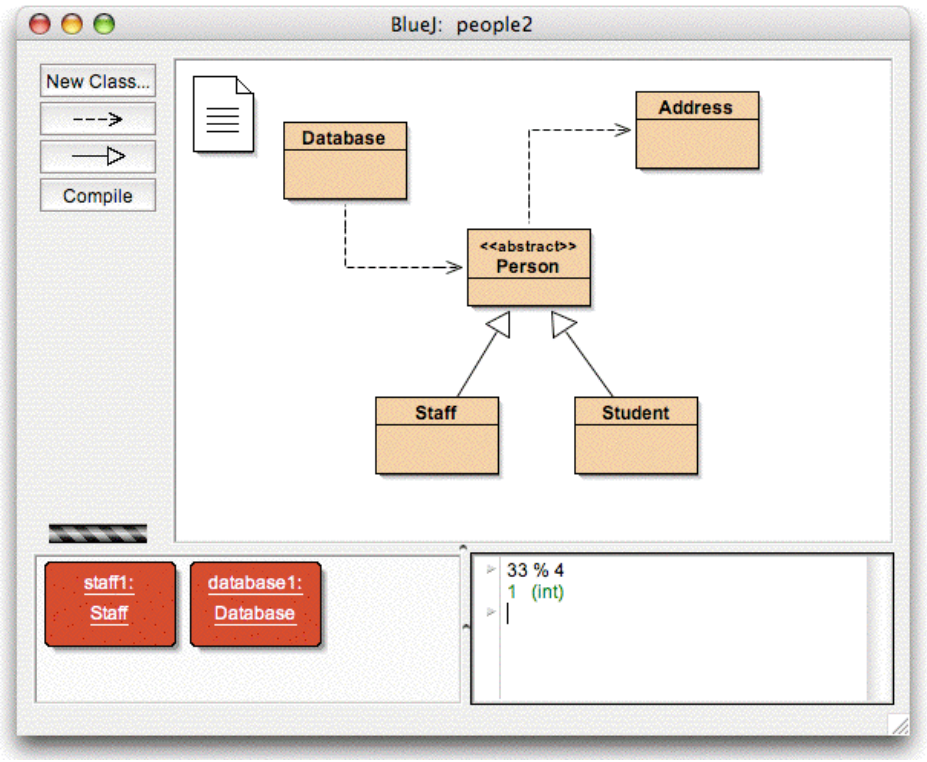

Figura 13: Fereastra principalã cu tãblita de cod afisatã

6.2 Evaluarea expresiilor simple

Rezumat: Penru a evalua expresii Java, tastati-le pur si simplu în tãblita de cod.

Încercati sã introduceti, de exemplu:

```
4 + 45
"hello".length()
Math.max(33, 4)
(int) 33.7
javax.swing.JOptionPane.showInputDialog(null, "Name:")
```

Expresiile se pot referi la valori si obiecte Java standard, precum si la clase din proiectul curent. Tãblita de cod va afisa valoarea rezultatului, urmatã de tipul sãu (între paranteze) sau un mesaj de eroare dacã expresia este incorectã.

De asemenea puteti folosi obiectele pe care le aveti pe bancul de obiecte. Încercati urmãtoarele: plasati un obiect de clasa ***Student*** pe bancul de obiecte (folosind meniul vertical al clasei cum am descris mai înainte). Numiti-l *student1*.

În tãblita de cod puteti acum scrie

```
student1.getName()
```

Similar, puteti referi toate metodele disponibile din clasele proiectului Dvs.

6.3 Transferul de obiecte

Rezumat: Pentru a transfera obiecte din tãblita de cod pe bancul de obiecte târâti mica icoanã a obiectului.

Unele rezultate ale expresiilor sunt obiecte, nu valori simple. În acest caz, rezultatul este afisat ca <*object reference*>, urmat de tipul obiectului si se deseneazã o micã icoanã lângã linia rezultat (Figura 14).

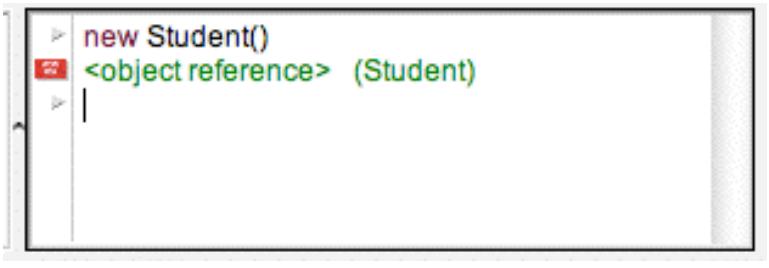

Figura 14: Un obiect rezultat dintr-o expresie din tãblita de cod

Dacã rezultatul este un sir (string), valoarea sa va fi afisata ca rezultat, dar veti vedea si mica icoanã de obiect (deoarece sirurile sunt obiecte).

Câteva expresii pe care le puteti încerca pentru a crea obiecte sunt:

```
new Student()
"marmelade".substring(3,8)
new java.util.Random()
"hello" + "world"
```

Mica icoanã de obiect poate fi acum folositã pentru a continua lucrul cu obiectul rezultat. Puteti selecta icoana si o puteti târî pe bancul de obiecte (Figure 15). Acest lucru va plasa obiectul pe banc, unde va fi disponibil pentru apeluri ulterioare ale metodelor sale, fie prin intermediul meniului sãu vertical, fie via tãblita de cod.

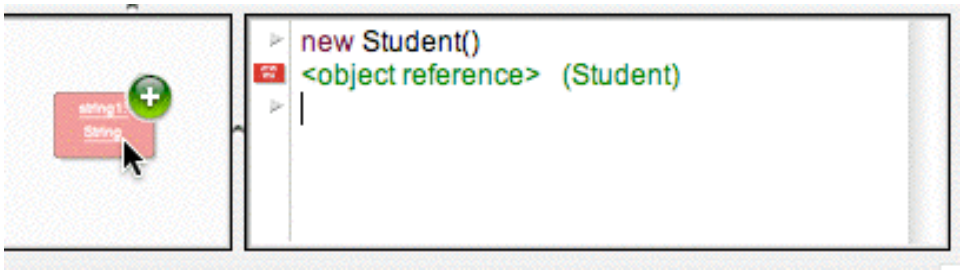

Figure 15: Târârea unui obiect în bancl de obiecte

6.4 Inspectarea obiectelor

Rezumat: Pentru a inspecta obiecte rezultat în tãblita de cod, dati dublu-clic pe mica icoana de obiect.

Dacã doriti sa inspectati un obiect returnat ca rezultat al unei expresii din tãblita de cod puteti face asta fãrã a plasa obiectul pe bancul de obiecte: puteti da pur si simplu dublu-clic pe icoana obiectului pentru a deschide inspectorul de obiecte uzual.

6.5 Executia instructiunilor

Rezumat: Instructiunile tastate în tãblita de cod sunt executate.

Puteti folosi tãblita de cod si pentru a executa instructiuni (adicã: instructiuni Java care nu returneaza o valoare). Încercati acestea, spre exemplu:

```
System.out.println("Gurkensalat");
System.out.println(new java.util.Random().nextInt(10));
```

Instructiunile sunt evaluate corect si executate cu sau fãrã punct si virgulã la sfârsit.

6.6 Instructiuni multi-linie si secvente de instructiuni

Rezumat: Folositi shift-Enter *la sfârsitul unei linii pentru a introduce instructiuni multi-linie.*

Puteti introduce secvente de instructiuni sau instructiuni care se întind pe mai multe linii folosind *shift-Enter* la sfârsitul liniei (în loc de ***Enter***). Folosirea *shift-enter* va deplasa cursorul la începutul liniei urmãtoare, dar nu va provoca (deocamdatã) executia intrãrii. La sfârsitul ultimei linii, tastati *Enter* pentru a evalua toate liniile împreunã. Încercati, de exemplu, o buclã *for*:

```
for (int i=0; i<5; i++) {
   System.out.println("number: " + i);
}
```

6.7 Lucrul cu variabile

Rezumat: Variabilele locale pot fi folosite într-o singurã instructiune multi-linie. Numele obiectelor de pe bancul de obiecte servesc pe post de câmpuri de instantã.

Variabilele – câmpuri de instantã si variabilele locale – pot fi folosite în tãblita de cod în moduri restrictionate.

Pueti declara variabile locale în tablita de cod, dar acest lucru este util doar ca parte a secventelor de instructiuni multi-linie, deoarece variabilele sunt pierdute de la o intrare la alta. Spre exemplu: puteti introduce urmãtorul bloc sub forma unei singure intrãri multi-linie si va merge asa cum s-a asteptat:

```
int sum;
sum = 0;
for (int i=0; i<100; i++) {
   sum += i;
}
System.out.println("The sum is: " + sum);
```

Introducerea aceleiasi secvente ca instructiuni separate va esua, deoarece variabila localã *sum* nu este pãstratã între intrãri.

Puteti concepe intrarea ca fiind textul din corpul unei metode. Orice se poate scrie legal în corpul unei metode Java este legal si în tâblita de cod. Totusi, fiecare intrare de text pe care o tastati formeazã o parte a unei metode *diferite*, asa cã nu vã puteti referi dintr-o linie de intrare la o variabilã dintr-o alta linie de intrare.

Vã puteti gândi la obiectele de pe bancul de obiecte ca la câmpurile de instantã. Nu puteti defini noi câmpuri ale unei instante din interiorul corpului unei metode (sau din tãblita de cod), dar puteti referi câmpuri de instanta si puteti apela obiectele continute în ele.

Puteti crea un nou câmp de instantã târând un obiect din tablita de cod pe bancul de obiecte.

6.8 Istoricul comenzilor

Rezumat: Folositi tastele sãgeatã-sus (up-arrow) *si sãgeatã-jos and* (down-arrow) *pentru istoric.*

Tãblita de cod retine o istorie a intrãrilor anterioare. Folosind tastele sãgeti *sus* sau *jos* puteti accesa linii introduse anterior si le puteti modifica înainte de a le refolosi.

7 Depanarea

Acesatã sectiune prezintã cele mai importante aspecte ale functionalitãtii de depanare din BlueJ. În discutiile cu profesorii de calculatoare am auzit adesea comentariul ca folosirea unei depanator la studentii din primul an ar fi bunã, dar pur si simplu nu este timp pentru a o prezenta. Studentii se luptã cu editorul, compilatorul si executia; nu este timp ramas pentru a prezenta o alta unealta complicatã.

De aceea am decis sã facem depanatorul cât mai simplu cu putintã. Scopul este sa avem un depanator pe care sã-l puteti explica în 15 minute si pe care studentii sã-l poata folosi din acel moment fãra alte instructiuni. Sã vedem dacã am reusit.

Mai intâi, am redus functionalitatea depanatoarelor traditionale la trei sarcini:

- setarea punctelor de întrerupere (breakpoints)
- executia în mod pas cu pas
- inspectarea variabilelor

În schimb, fiecare dintre cele trei sarcini este foarte simplã. Le vom încerca acum pe fiecare.

Pentru a începe, deschideti proiectul *debugdemo*, care este inclus în directorul *examples* din distributie. Acest proiect contine câteva clase pentru singurul scop al demonstrãrii functionalitãtii depanatorului – altfel nu prea au sens.

7.1 Setarea punctelor de întrerupere

Rezumat: Pentru a seta un punct de întrerupere, dati click în zona de breakpoint din stânga editorului.

Setarea unui punct de întrerupere vã permite sã întrerupeti executia într-un anumit punct din cod. Atunci când executia este întreruptã, puteti investiga starea obiectelor. Adesea este de ajutor ca sa întelegeti ce se întâmplã în codul Dvs.

În editor editor, la stânga textului, se aflã zona punctelor de întrerupere (Figura 16). Puteti seta un punct de întrerupere dând click în locul dorit. Apare un mic semn stop care marcheazã tpunctul de întrerupere. Încercati acum asta. Deschideti clasa *Demo*, gãsiti metoda *loop* si setati un punct de întrerupere undeva în bucla *for*. Ar trebui sa aparã un semn stop în editor.

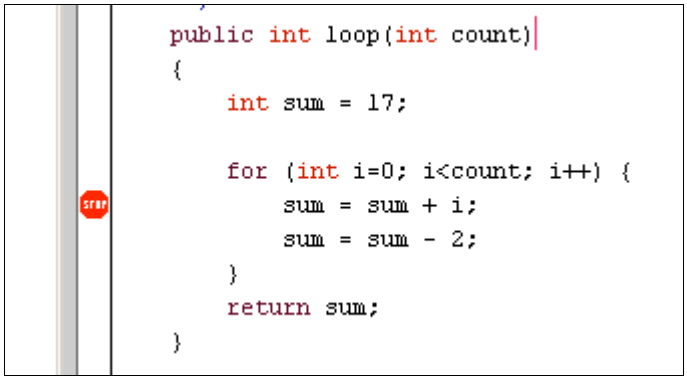

Figura 16: Un punct de intrerupere

Atunci când se ajunge la linia de cod care are punctul de întrerupere atasat, executia va fi întreruptã. Sã încercãm acum asta.

Creati un obiect de clasa *Demo*si apelati-i metoda *loop* cu un parametru de, sa zicem, 10. De îndatã ce s-a ajuns la punctul de întrerupere, va fi afisatã în prim plan fereastra editorului care va sublinia linia de cod respectivã; de asemernea va apãrea în prim plan o fereastrã de depanator precum cea din figura 17.

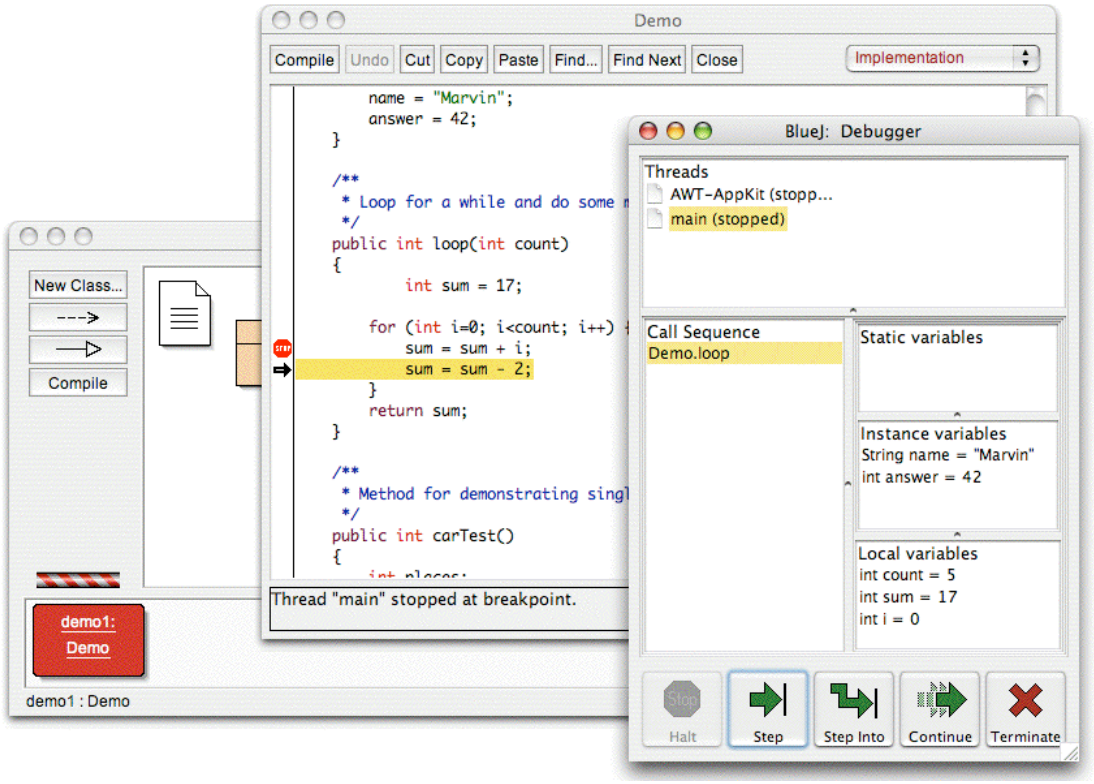

Figura 17: Fereastra depanatorului

În editor este evidentiatã linia urmãtoare de executat. (Executia este stopatã ***înainte*** ca aceastã linie sã fi fost executatã.)

7.2 Executia în modul pas cu pas

Rezumat: Pentru a executa codul pas cu pas, folositi butoanele Step *si* Step Into *din depanator.*

Acum ca am stopat executia (ceea ce ne convinge cã metoda este într-adevãr executatã si cã acest punct din cod este într-adevãr atins), putem trasa pas cu pas codul si putem vedea cum progresează executia. Pentru a face acest lucru, dati clic în mod repetat pe butonul *Step* din fereastra depanatorului. Ar trebui sã vedeti cã se schimbã linia sursã din editor (marcajul în culoare se mutã cu linia executatã). De fiecare datã când dati clic pe butonul *Step* se executã o singurã linie de cod si executia se opreste din nou. Observati si cã valorile variabilelor afisate în depanator se schimbã (de exemplu valoarea variabilei *sum*.) Astfel puteti executa codul pas cu pas si puteti observa ce se întâmplã. Dupã ce v-ati plictisit de asta, putetida clic din nou pe punctul de întrerupere pentru a-l elimina si apoi da clic pe butonul *Continue* din depanator pentru a reporni executia si a continua în mod normal.

Sã încercãm din nou o alta metodã. Setati un punct de întrerupere în clasa *Demo*, metoda *carTest()*, în linia care contine

```
places = myCar.seats();
```

Apelati metoda. Atunci când se atinge acest punct, urmeazã sã se execute o linie care contine un apel al metodei *seats()* din clasa *Car*. Dacã dati clic pe *Step* atunci se executã intreaga linie într-un singur pas. Sa încercãm acum *Step Into*. Dacã *pãsiti* ***intr-un*** apel de metodã, atunci intrati în metoda curentã si executati metoda însãsi linie cu linie (nu într-un singur pas). În acest caz, executia ajunge în metoda *seats()* din clasa *Car*. Acum puteti pãsi prin metodã pânã la sfârsit (cu ***Step***) si sã ajungeti înapoi în metoda apelantã. Observati cum se schimbã afisajul depanatorului.

Step si *Step Into* se comporta identic daca linia curenta nu contine un apel de metodã.

7.3 Inspectarea variabilelor

Rezumat: Inspectarea variabilelor este usoarã – ele sunt automat afisate în fereastra depanatorului.

Atunci când depanati codul, este important sã fiti capabil(ã) sã inspectati starea obiectelor (variabile locale si variabile de instantã).

Acesta este un lucru simplu – majoritatea celor necesare le-ati vãzut deja. Nu aveti nevoie de comenzi speciale pentru a inspecta variabilele; variabilele statice, variabilele de instantã ale obiectului curent si variabilele locale ale metodei curente sunt întotdeauna afisate si actualizate automat.

Puteti alege metode din secventa de apeluri pentru a vizualiza variabilele altor obiecte si metode active în mod curent. Încercati, de exemplu, un punct de întrerupere în metoda

carTest() din nou. Pe partea stânga a ferestrei depanatorului vedeti secventa de apeluri. Acum aratã

```
Car.seats
Demo.carTest
```

Aceasta indicã faptul cã metoda `Car.seats` a fst apelatã de `Demo.carTest`. Puteti selecta `Demo.carTest` din aceastã listã pentru a inspecta sursa si valorile curente ale variabilelor din aceastã metodã.

Dacã pãsiti peste linia care contine instrcutiunea `new Car(...)`, puteti observa ca valoarea variabilei locale `myCar` este afisatã ca <*object reference*>(referintã la obiect). Toate valorile de tipuri obiect (cu exceptia sirurilor [String]) sunt afisate în acest fel. Puteti inspecta variabila dând dublu-clic pe ea. Daca facet asa, se va deschide o fereastrã de inspectare a obiectului identicã cu aceea descrisã anterior (sectiunea 4.1). Nu exista o diferenta realã între inspectarea obiectelor aici si inspectarea lor pe bancul de obiecte.

7.4 Oprirea si terminarea

Rezumat: Halt *si* Terminate *pot fi folosite pentru a opri executia temporar sau permanent.*

Câteodata un program ruleaza de mult timp si vã întrebati dacã totul este în ordine. Poate e în buclã infinitã, poate doar necesitã prea mult timp. Ei bine, putem verifica. Apelati metoda *longloop()* din clasa *Demo*. Aceastã metodã rulează ceva timp.

Acum dorim sã stim ce se întâmplã. Afisati fereastra depanatorului dacã nu e deja pe ecran.

Acum dati clic pe butonul *Halt*. Executia este întreruptã la fel ca în cazul în care am fi dat peste un punct de întrerupere. Puteti merge pas cu pas câtiva pasi, puteti observa variabilele si sa vedeti ca totul e în regulã – pur si simplu e nevoie de mai mult timp. Opriti cu *Halt* de câteva ori pentru a vedea cât de repede numãrã. Daca nu doriti sã continuati (de exemplu, ati descoperit ca vã aflati întradevãr într-o buclã infinitã) puteti da clic pe *Terminate* pentru a termina întreaga executie. *Terminate* nu ar trebui folosit prea frecvent – pentru ca puteti lãsa obiecte scrise perfect de bine într-o stare inconsistentã prin terminarea executiei masinii, asa cã este recomandat sã o folositi doar ca mecanism de urgentã.

8 Crearea aplicatiilor de sine stãtãtoare

Rezumat: Pentru a crea o aplicatie de sine stãtãtoare, folositi Project - Create Jar File...

BlueJ poate crea fisiere jar executabile. Fieisrele jar executabile pot fi executate pe unele sisteme dând dublu clic pe fisier (spre exemplu sub Windows si MacOS X), sau prin comnda text `java -jar <file-name>.jar` (Unix sau DOS prompt).

Vom încerca aceasta cu proiectul exemplu *hello*. Deschideti-l (se aflã în directorul *examples*). Asigurati-vã ca proiectul este compilat. Selectati functia *Create Jar File...* din meniul *Project*.

Se deschide un dialog care vã permite sã specificati clasa main (Figure 18). Aceastã clasã trebuie sa aibã definitã o ***metodã*** *main* (cu semnãtura `public static void main(String[] args)`).

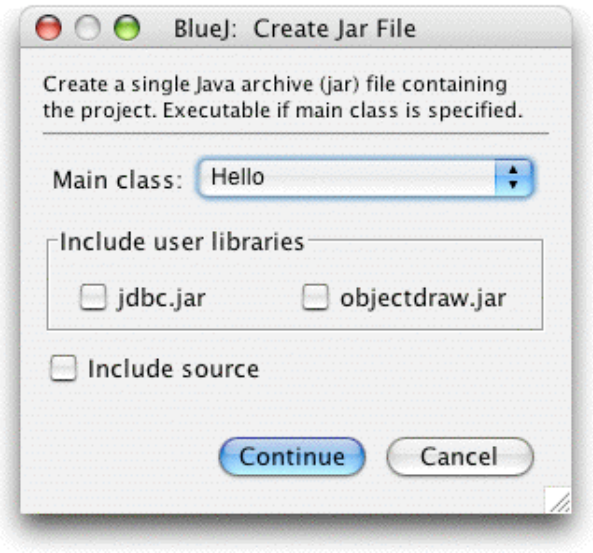

Figure 18: Dialogul "Create Jar File"

În exemplul nostru, alegerea calsei main este usoarã: existã doar una. Alegeti *Hello* din meniu. Daca aveti alte proiecte, alegeti clasa care contine metoda "main" pe care doriti sã o executati.

De obicei nu se includ surse în fisierele executabile. Dar puteti, dacã doriti sa distribuiti si sursele. (Puteti folosi formatul jar pentru a trimite întregul proiect altcuiva prin email într-un singur fisier, de exemplu.)

Dacã ati configurat BlueJ sã folosească biblioteci ale utilizatorului (fie via setarea ***Preferences/Libraries***, fie folosind directorul *lib/userlib*) veti vedea o zonã intitulatã *Include user libraries* în mijlocul dialogului. (Dacã nu folositi nici un fel de biblioteci aceasta zona va lipsi.) Trebuie sã verificati fiecare bibliotecã pe care o foloseste proiectul Dvs. curent.

Clic pe *Continue*. Apoi veti vedea un dialog de alegere a fisierului care vã permite sã dati un nume fisierului jar care se va crea. Tastati *hello* si dati clic pe ***Create***.

Dacã nu aveti biblioteci de inclus, atunci se va crea acum un fisier numit *hello.jar* Daca aveti biblioteci, se va crea un director numit *hello* si în el fisierul jar numit *hello.jar*. Directorul va contine si toate bibliotecile necesare. Fisierul Dvs. jar se asteapta sa gãseascã bibliotecile referite în acelasi director în care se afla si el – asa cã asigurati-va cã pãstrati aceste fisiere jar împreunã atunci când le mutati.

Puteti face dublu-clic pe fisierul jar doar dacã aplicatia foloseste o interfatã GUI. Exemplul nostru foloseste I/E în mod text, asa cã trebuie sã-l lansãm dintr-un terminal în modul text. Sa încercãm acum sã lansãm fisierul jar.

Deschideti o fereastra de terminal sau o fereastra DOS. Apoi navigati în directorul unde ati salvat fisierul file (ar trebui sa vedeti un fisier numit *hello.jar*). Presupunand ca Java este instalat corect pe sistemul Dvs., ar trebui sa puteti tasta

```
java -jar hello.jar
```

pentru a executa fisierul.

9 Crearea applet-urilor

9.1 Rularea unui applet

Rezumat: Pentru a rula un applet, selectati Run Applet *din meniul vertical al applet-ului.*

BlueJ permite atât crearea si executarea applet-urilor cât si a aplicatiilor. Am inclus un applet în directorul ***examples*** din distributie. Mai intâi, dorim sã încercãm executarea unui applet. Deschideti proiectul *appletdemo* din ***examples***.

Veti vedea ca acest proiect are o singura clasa; ea este numitã *CaseConverter*. Icoana clasei este marcatã (cu eticheta <<applet>>) ca applet. Dupa compilare, alegeti comanda *Run Applet* din meniul vertical al clasei.

Apare un dialog care va lasã sa faceti câteva selectii (Figura 19).

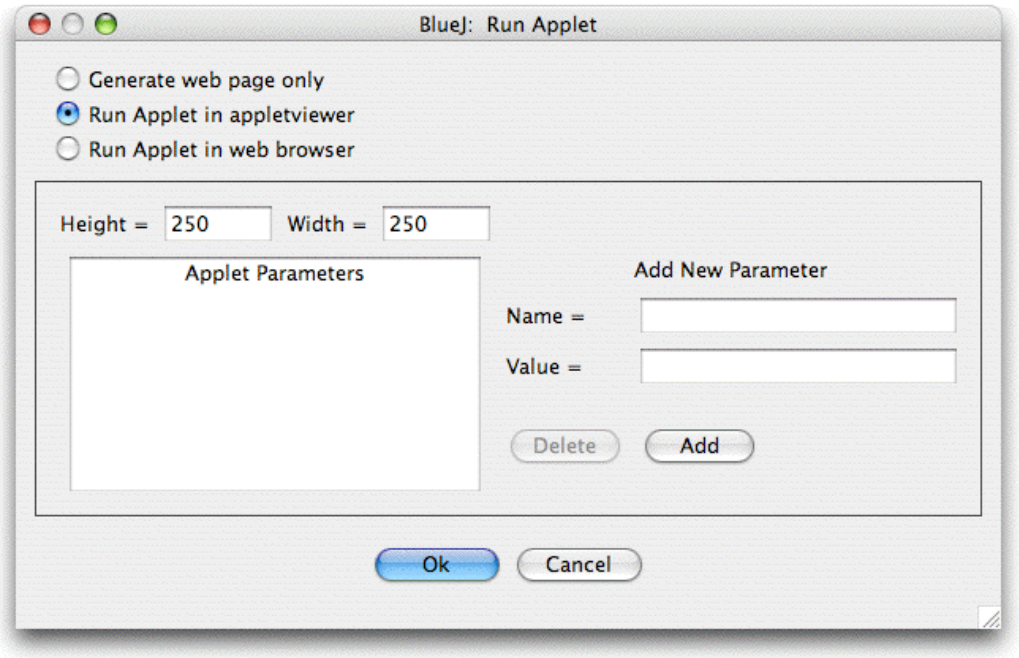

Figura 19: Dialogul "Run Applet"

Vedeti cã aveti optinuea de a rula applet-ul într-un browser sau într-un vizualizator de applet-uri (sau doar sã generati pagina de web farã a-l rula). Lãsati setãrile implicite si dati clic pe *OK*. Dupa câteva secunde, ar trebui sã aparã un vizualizator de applet-uri care sa afiseze applet-ul de conversie majuscule/minuscule.

Vizualizatorul de applet-uri este instalat împreuna cu J2SE SDK (instalarea de Java), asa ca este întotdeauna garantat cã se afla în aceiasi versiune ca si compilatorul Java.

În general, cauzează mai putin probleme decât browser-ele. Browser-ul Dvs. de web poate sa ruleze o versiune de Java diferită si, în functie de care versiune a cărui browser o folositi poate cauza probleme. Cu cele mai noi browsere ar trebui sa meargă, totusi, bine.

Pe Microsoft Windows si MacOS, BlueJ foloseste browser-ul implicit. Pe sistemele Unix browser-ul este definit în setările BlueJ.

9.2 Crearea unui applet

Rezumat: Pentru a crea un applet, dati clic pe butonul New Class *si alegeti* Applet *ca tip de clasă.*

Dupa ce am văzut cum se ruleaza applet-ul, dorim să ne creăm unul.

Creati o clasa nouă cu *Applet* ca tip de clasă (puteti alege tipul î dialogul *New Class*). Compilati si apoi executati applet-ul. Asta e tot! N-a fost prea rau, nu?

Applet-urile (ca si alte clase) sunt generate cu un schelet de clasă implicit, care contine ceva cod valid. Pentru applet-uri, acest cod prezintă un applet simplu cu două linii de text. Puteti acum deschide editorul si edita applet-ul pentru a va insera codul propriu.

Veti vedea ca toate metodele comune ale applet-urilor sunt acolo, fiecare însotita de un comentariu care îi explică scopul. Codul exemplu este cuprins tot în metoda *paint*.

9.3 Testarea applet-ului

În unele situatii poate fi util să cream un obiect applet pe bancul de obiecte (ca pentru clasele normale). Puteti face asta – constructorul este prezent în meniul vertical al applet-ului. Din bancul de obiecte nu puteti executa întregul applet, dar puteti apela unele metode. Asta poate fi util pentru testarea metodelor solitare pe care le veti fi scris ca parte a implementării applet-ului.

Dacă setati puncte de întrerupere într-un applet, ele nu vor avea nici un efect la rularea applet-ului într-un vizualizator sau într-un browser de web. Acesata este fiindcă atât browser-ele cât si vizualizatoarele de applet-uri folosesc propriile masini virtuale pentru executia applet-ului, masini care nu înteleg nimic despre punctele de întrerupere BlueJ.

Dacă doriti sa folositi puncte de întrerupere si executia pas cu pas într-un applet, puteti folosi *AppletWindow*, scris de Michael Trigoboff. Clasa oferă un cadru (frame) care vă permite să rulati applet-ul direct în BlueJ, asa că depanarea normala functionează. Puteti găsi această clasă si un demo în sectiunea *Resources* pe situl de web al BlueJ.

10 Alte operatii

10.1 Deschiderea pachetelor non-BlueJ în BlueJ

Rezumat: Pachetele non-BlueJ pot fi deschise cu comanda Project: Open Non BlueJ...

BlueJ vã permite sã deschideti pachete existente care au fost create în afara BlueJ. Pentru a face aceasta selectati Project – Open Non BlueJ... din meniu. Alegeti in director care contine fisierele sursã Java, apoi dati clic pe butonul Open in BlueJ. Sistemul vã va solicita sã confirmati cã doriti sã deschideti acest director.

10.2 Adãugarea de clase la un proiect

Rezumat: Clasele pot fi copiate într-un proiect din afarã folosind comanda Add Class from File...

Adesea, doriti sã folositi o clasã pe care ati obtinut-o de altundeva în proiectul BlueJ al Dvs. Spre exemplu, un profesor poate da o clasa Java studentilor ca sa o foloseascã în proiect. Se poate incorpora cu usurintã o clasã existentã în proiectul Dvs. selectãnd – Add Class from File... din meniu. Aceasta vã va permite sã alegeti un fisier sursã Java (cu un nume care se terminã în *.java*) ca sã fie importat.

La importul unei clase în proiect, se face o copie si se stochează în directorul proiectului curent. Efectul este exact acelasi ca si când ati fi creat clasa respectivã si i-ati fi scris codul sursã.

O alternativa este sã adaugati fisierul sursã al noii clase în directorul proiectului din afara BlueJ. La urmatoarea deschidere a acelui proiect, clasa va fi inclusã în diagrama proiectului.

10.3 Apelul lui *main* si a altor metode statice

Rezumat: Metodele statice pot fi apelate din meniul vertical al clasei.

Deschideti proiectul *hello* din directorul *examples*. Singura clasa din proiect (clasa ***Hello***) defineste o metoda main standard.

Clic-dreapta pe clasã, si veti vedea ca meniul clasei nu include doar constructorul classei, ci si metoda staticã *main*. Puteti acum invoca *main* direct din acest meniu (fãrã a crea mai întâi un obiect).

Toate metodele statice pot fi invocate în acest fel. Metoda main standard asteaptã ca argument un tablou de siruri (Strings). Puteti transfera un tablou de siruri folosind sintaxa Java standard pentru constante de tip tablou. Spre exemplu, puteti transfera

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

(inclusiv acoladele) metodei. Încercati!

> Notã: *În Java standard, constantele tablou nu pot fi folosite ca argumente efective la apelul metodelor. Ele pot fi folosite doar ca initializatori. În BlueJ, pentru a permite apeluri interactive ale metodelor main standard, permitem transferul constantelor tablou*

10.4 Generarea documentatiei

Rezumat: Pentru a genera documentatia unui proiect, alegeti Project Documentation *din meniul* Tools.

Puteti genera docmentatie pentru proiectul Dvs. în formatul standard *javadoc* din interul BlueJ. Pentru a face asta, selectati Tools - Project Documentation din meniu. Aceastã functie va genera documentatia pentru toate clasele din proiect folosind codul sursã al clasei si va deschide un browser de web penru a o afisa.

De asemenea puteti genera si vizualiza documentatia pentru o singurã clasã direct în editorul BlueJ. Pentru a face asta, deschideti editorul si folositi meniul vertical din bara de unelte a editorului. Schimbati selectia din *Implementation* în *Interface*. Aceasta va arãta documentatia în stil *javadoc* (interfata clasei) în editor.

10.5 Lucrul cu biblioteci

Rezumat: API standard Java a clasei poate fi vizualizatã selectând Help - Java Class Libraries.

Frecvent, atunci când scrieti un program Java, trebuie sã referiti biblioteci Java standard. Puteti deschide un browser de web care sã prezinte documentatia JDK API alegând Help - Java Standard Classes din meniu (daca sunteti online).

Documentatia JDK poate fi instalatã si folositã local (offline). Detailiile sunt explicate în sectiunea de help pe situl de web BlueJ.

10.6 Crearea de obiecte din clase de bibliotecã

Rezumat: Pentru a crea obiecte din clase de bibliotecã, folositi Tools – Use Library Class.

BlueJ oferã si o functie pentru a crea obiecte din clase care nu sunt parte a proiectului Dvs. ci sunt definite într-o bibliotecã. Puteti, de exemplu, crea obiecte de clasa String sau ArrayList. Aceasta poate fi foarte util pentru experimentarea rapidã cu aceste obiecte de bibliotecã.

Puteti crea un obiect de bibliotecã selectând Tools – Use Library Class... din meniu. Se va afisa un dialog care va cere sã introduceti un nume de clasã complet calificat cum este *java.lang.String*. (Observati ca trebuie sa dati numele complet calificat, adicã numele inclusiv al pachetelor care contin clasa.)

Câmpul de intrare text are un meniu vertical asociat care aratã clasele folosite recent. O datã ce s-a introdus un nume de clasã apãsarea tastei *Enter* va provoca afisarea tuturor constructorilor si a metodelor statice ale acelei clase într- listã în dialog. Oricare dintre acesti constructori sau metode statice poate fi acum invocat prin selectarea din aceasta listã.

Invocarea continuã ca orice alt apel de constructor sau de metodã.

11 Rezumate

Cum se incepe

1. Pentru a deschide un proiect alegeti *Open* din meniul *Project*.
2. Pentru a crea un obiect, alegeti un constructor din meniul vertical.
3. Pentru a executa o metodã, alegeti-o din meniu vertical al obiectului.
4. Rezumat: Pentru a edita sursa unei clase, dati dublu-clic pe icoana clasei.
5. Pentru a compila o clasã, dati clic pe butonul *Compile* din editor. Pentru a compila un proiect dati clic pe butonul *Compile* din fereastra proiectului.
6. Pentru a obtine ajutor pentru un mesaj de eroare al compilatorului dati clic pe semnul de întrebare de lângã mesajul de eroare.

U n pic mai mult...

7. Inspectia obiectului permite ceva depanare simplã prin verificarea stãrii interne a obiectului.
8. Un obiect poate fi transferat ca parametru la o apelul unei metode prin clicl pe icoana obiectului.

Crearea unui proiect nou

9. Pentru a crea un proiect, alegeti *New...* din meniul *Project*.
10. Pentru a crea o clasã, faceti clic pe butonul ***New Class*** si specificati numele clasei.
11. Pentru a crea o sãgeatã, faceti clic pe butonul sãgeatã si târâti sãgeata în diagramã, sau pur si simplu scrieti codul în editor.
12. Pentru a înlãtura o clasã sau o sãgeatã, selectati functia ***Remove*** din meniul vertical

Folosirea tãblitei de cod

13. Pentru a începe sã folositi tãblita de cod, alegeti ***Show Code Pad*** din meniul ***View***
14. Penru a evalua expresii Java, tastati-le pur si simplu în tãblita de cod.
15. Pentru a transfera obiecte din tãblita de cod pe bancul de obiecte târâti mica icoana de obiect.
16. Pentru a inspecta obiecte rezultat în tãblita de cod, dati dublu-clic pe mica icoanã de obiect.
17. Instructiunile tastate în tãblita de cod sunt executate.
18. Folositi ***shift-Enter*** la sfârsitul unei linii pentru a introduce instructiuni multi-linie.
19. Variabilele locale pot fi folosite într-o singura instructiune multi-linie statements. Numele obiectelor de pe bancul de obiecte serevesc drept câmpuri de instantã.
20. Folositi tastele sãgeatã-sus (***up-arrow***) si sãgeatã-jos and (***down-arrow***) pentru istoricul intrarilor.

Depanarea

21. Pentru a seta un punct de întrerupere, dati click în zona de breakpoint din stânga editorului.
22. Pentru a executa codul pas cu pas, folositi butoanele ***Step*** si ***Step Into*** din depanator
23. Inspectarea variabilelor este usoarã – ele sunt automat afisate în fereastra depanatorului.
24. *Halt* si *Terminate* pot fi folosite pentru a opri executia temporar sau permanent.

Crearea aplicatiilor de sine stãtãtoare

25. Pentru a crea o aplicatie de sine stãtãtoare, folositi *Project - Create Jar File...*

Crearea applet-urilor

26. Pentru a rula un applet, selectati *Run Applet* din meniul vertical al applet-ului.
27. Pentru a crea un applet, dati clic pe butonul New Class si alegeti Applet ca tip de clasã.

Alte operatii

28. Pachetele non-BlueJ pot fi deschise cu comanda ***Project: Open Non BlueJ***... .
29. Clasele pot fi copiate într-un proiect din afarã folosind comanda *Add Class from file...*
30. Metodele statice pot fi apelate din meniul vertical al clasei.
31. Pentru a genera documentatia unui proiect, alegeti *Project Documentation* din meniul *Tools*
32. API standard Java a clasei poate fi vizualizatã selectând *Help Java Standard Libraries*
33. Pentru a crea obiecte din clase de bibliotecã, folositi *Tools - Use Library Class.*