Tutorial BlueJ

versiunea 2.0.1
pentru BlueJ versiunea 2.0.x

Michael Kolling
Maersk Institute
University of Southern Denmark

Traducere de **Dumitrascu Irina** si **Popa Adrian**

Cuprins

1 Cuvant de inceput

1.1 Despre BlueJ

Acest tutorial este o introducere in mediul de programare BlueJ. BlueJ este un mediu de dezvoltare Java™ realizat special pentru a preda programare la un nivel introductiv. El a fost proiectat si implementat de echipa BlueJ de la Universitatea Deakin din Melbourne, Australia si de Universitatea Kent din Cantebury, UK.

Mai multe informatii despre BlueJ puteti gasi la http://www.bluej.org

1.2 Audienta tinta

Acest tutorial este destinat persoanelor care vor sa se familiarizeze cu capacitatiile acestui mediu de dezvoltare. El nu explica deciziile luate in dezvoltarea mediului sau problemele de care s-a lovit in stadiul de cercetare.

Acest tutorial nu este menit sa fie un ghid de programare in Java. Programatorii incepatori in Java sunt sfatuiti sa citeasca si un tutorial de Java sau sa ia parte la un curs Java.

Acesta nu e un tutorial atotcuprinzator. Multe detalii au fost omise pentru a se accentua asupra unei introduceri scurte si concise in locul unei prezentari amanuntite. Pentru detalii ulterioare consultati *The BlueJ Environment Reference Manual* disponibil pe site-ul BlueJ.

Fiecare sectiune incepe cu o fraza care rezuma continutul sectiunii. Acest lucru permite cititorilor care cunosc parti din sistem sa decida daca vor sa citeasca acea sectiune sau daca trec mai departe. Sectiunea 11 repeta aceste fraze, pentru a fi mai usor de urmarit structura tutorialului.

1.3 Copyright, licenta si redistributie

1.4 Feedback

Comentarii, corectii, intrebari si critici legate de sistemul BlueJ sau de tutorial sunt bine venite si incurajate. Va rugam sa le trimiteti lui Michael Kolling (mik@mip.sdu.dk)

2. Instalare

BlueJ e distribuit in 3 formate distincte: unul pentru Windows, unul pentru MacOS, si unul pentru restul sistemelor. Instalarea este destul de simpla.

Elemente necesare

Trebuie sa aveti instalat inainte Java2SE v 1.4 (adica JDK 1.4) sau o versiune mai noua pentru a folosi BlueJ. In general este recomandata trecerea la cea mai noua versiune stabila (non-beta). Daca nu aveti un JDK instalat, puteti sa-l downloadati de pe site-ul Sun: http://java.sun.com/j2se/ . Pe MacOS X o versiune recenta de J2SE e preinstalata – asa ca nu mai e nevoie sa o instalati manual. Daca gasiti o pagina care va ofera sa downloadati JRE (Java Runtime Environment) si SDK (Software Development Kit), trebuie sa downloadati SDK deoarece JRE nu e suficient.

2.1 Instalare pe Windows

Fisierul prin care e distribuit BlueJ pe sisteme Windows se numeste *bluejsetup-xxx.exe*, unde *xxx* e numarul versiunii. De exemplu, BlueJ versiunea 2.0.0 are fisierul *bluejsetup-200.exe*. Puteti obtine fisierul de pe un disc, sau il puteti incarca de la site-ul BlueJ – http://www.bluej.org. Lansati in executie installer-ul.

Installer-ul va permite sa alegeti directorul in care vreti sa instalati. Va oferi si posibilitatea de a creea o scurtatura pe Desktop sau in Start Menu.

Dupa ce se incheie instalarea, puteti gasi fisierul *bluej.exe* in directorul de instalare a BlueJ-ului.

Prima data cand lansati BlueJ, el va cauta un sistem JDK. Daca descopera mai mult de un sistem JDK potrivit (de exemplu aveti instalat JDK 1.4.2 si JDK 1.5.0) o fereastra de dialog va va intreba pe care doriti sa-l folositi. Daca nu gaseste nici unul va va ruga sa ii specificati manual calea catre JDK (acest lucru se poate intampla cand un JDK a fost instalat, dar intrarile acestuia din registry au fost sterse).

Installer-ul BlueJ instaleaza si un program numit *vmselect.exe*. Folosind acest program puteti hotara mai tarziu ce masina virtuala sa foloseasca BlueJ-ul.

Alegerea JDK-ului este salvata pentru fiecare versiune de BlueJ. Daca aveti versiuni diferite de BlueJ instalate, puteti folosi o versiune cu JDK 1.4.2 si alta cu JDK 1.5.0. Schimbarea versiunii Java pentru BlueJ va modifica versiunea pentru toate instalarile BlueJ de aceeasi versiune pentru un user.

2.2 Instalare pe Macintosh

Aveti in vedere ca BlueJ ruleaza doar pe MacOS X.

Fisierul prin care e distribuit BlueJ pentru MacOS se numeste *BlueJ-xxx.zip*, unde *xxx* este versiunea. De exemplu, BlueJ versiunea 2.0.0 are fisierul *BlueJ-200.zip*. Puteti obtine

fisierul de pe un disc, sau il puteti incarca de la site-ul BlueJ – http://www.bluej.org.

MacOS de obicei decomprima fisierul imediat dupa download. Daca nu a facut asta, efectuati un dublu-click pe fisier pentru a-l decomprima.

Dupa decompresie veti obtine un director numit *BlueJ-xxx*. Mutati acest director in directorul Apllications (sau oriunde doriti sa il tineti). Nu mai este nevoie de alte actiuni pentru instalare.

2.3 Instalare pe Linux/Unix si alte sisteme

Formatul general in care e distribuit BlueJ pentru astfel de sisteme este un fisier jar executabil. El e numit *bluej-xxx.jar*, unde *xxx* e versiunea. De exemplu, BlueJ versiunea 2.0.0 are fisierul *bluej-200.jar*. Puteti obtine fisierul de pe un disc, sau il puteti incarca de la site-ul BlueJ – http://www.bluej.org.

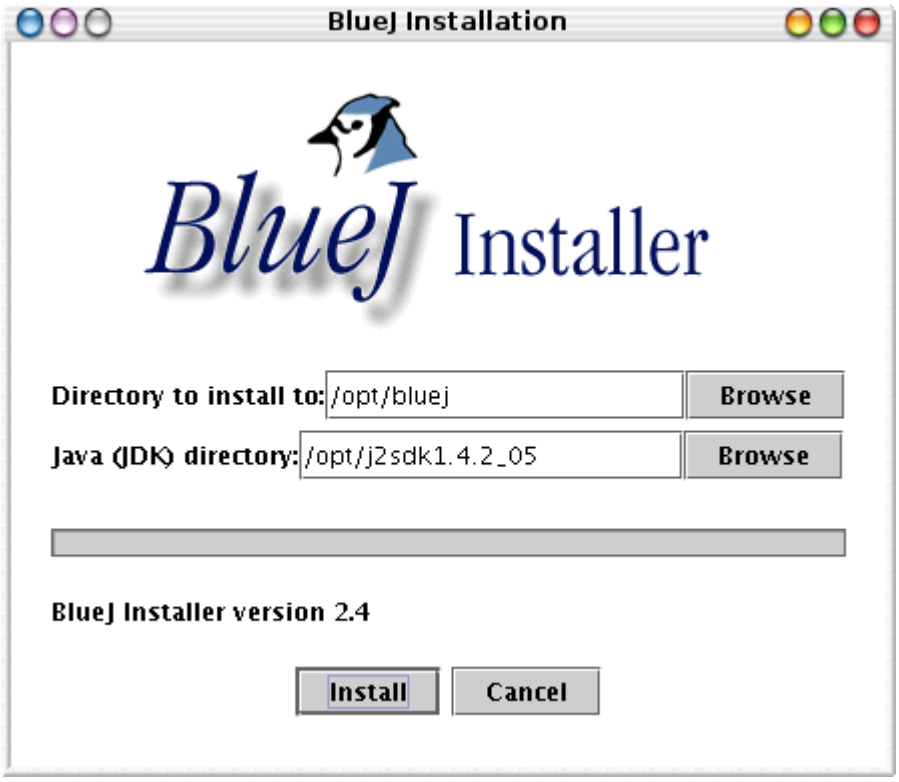
Rulati installer-ul executand urmatoarea comanda. NOTA: Pentru exemplu am folosit fisierul *bluej-200.jar* – trebuie sa folositi numele fisierului pe care l-ati obtinut (cu versiunea corecta).

```
<calea_catre_j2se>/bin/java -jar bluej-200.jar
```

<calea_catre_j2se> e calea catre directorul in care a fost instalat J2SE SDK.

Va aparea o fereastra in care puteti alege locul in care se va face instalarea si versiunea de Java ce va fi folosita de BlueJ.

Apasati *Install*. Dupa ce termina, BlueJ ar trebui sa fie instalat.



2.4 Probleme la instalare

Daca aveti orice fel de probleme, vizitati sectiunea *Frequently Asked Questions (FAQ)* de pe site-ul BlueJ (http://www.bluej.org/help/faq.html) si cititi sectiunea *How to ask for help* (http://www.bluej.org/help/ask-help.html).

3. Inceputul – editarea, compilarea si executia

3.1 Pornirea BlueJ

Pe sistemele Windows si MacOS se instaleaza un program numit BlueJ. Rulati-l.

Pe sisteme UNIX installer-ul creeaza un script numit *bluej* in directorul instalarii. Dintr-o interfata grafica dati dublu click pe fisier. Din linia de comanda puteti lansa BlueJ cu sau fara un proiect ca argument:

```
$ bluej
```

sau

```
$ bluej examples/people
```

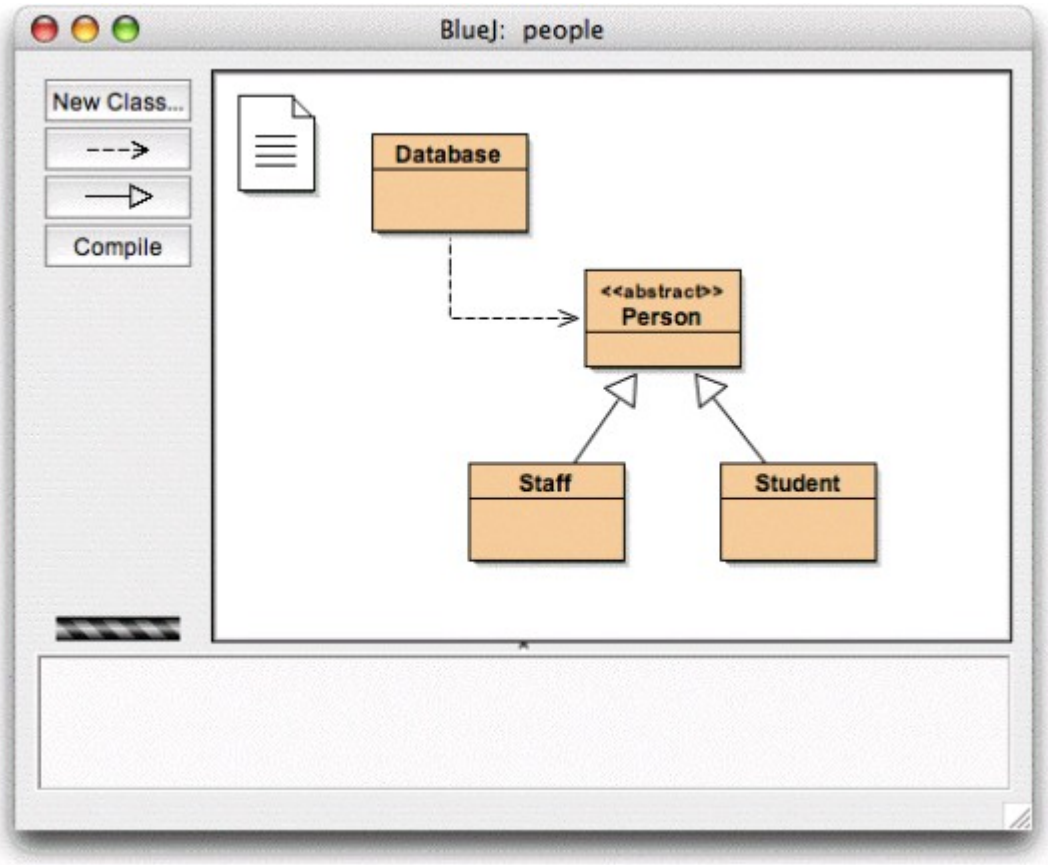


Fig 1. Fereastra centrala BlueJ

3.2 Deschiderea unui proiect

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

Proiectele BlueJ, ca si pachetele standard Java, sunt directoare ce contin fisierele incluse in proiect.

Dupa pornirea BlueJ, folositi meniul *Project – Open...* pentru a selecta si deschide un proiect.

Cateva proiecte sunt incluse pe post de exemple in distributia BlueJ standard in directorul *examples*.

In aceasta sectiune de tutorial, deschideti proiectul *people*, care este inclus in acest director. Puteti gasi directorul *examples* in directorul de instalare al BlueJ. Dupa ce deschideti proiectul puteti vedea ceva similar cu Fig 1. Fereastra nu va arata exact la fel pe sistemul dumneavoastra, dar diferentele sunt minore.

3.3 Crearea obiectelor

Rezumat: Pentru a creea un obiect, selectati un constructor din meniul pop-up al clasei.

O caracteristica fundamentala a BlueJ-ului este ca puteti executa o aplicatie intreaga, dar puteti si interactiona direct cu obiecte ale oricarei clase si le puteti executa metodele publice. Executia in BlueJ este facuta de obicei prin crearea unui obiect si apoi invocarea unei metode a obiectului. Acest lucru este foarte util in timpul dezvoltarii aplicatiei – puteti testa clasele imediat ce au fost scrise. Nu este nevoie sa fie scrisa toata aplicatia de la inceput.

NOTA: *Metodele statice pot fi executate direct, fara a necesita crearea unui obiect. Una din metodele statice poate fi main(), asa ca putem face aceleasi lucruri care se petrec intr-o aplicatie Java normala – pornirea aplicatiei prin rularea unei metode statice. Vom reveni la acest aspect mai tarziu. In primul rand vom face anumite lucruri, mai interesante, ce nu pot fi facute in mod normal intr-un mediu Java.*

Dreptunghiurile ce pot fi observate in fereastra centrala (numite *Database*, *Person, Staff* si *Student*) sunt pictograme ce reprezinta clasele implicate in aplicatie. Puteti obtine un meniu cu actiuni referitoare la o clasa apasand pe pictograma ei cu butonul drept (Macintosh: ctrl+click[1]) (Fig 2). Operatiunile aratate apeleaza *new* pentru fiecare din constructorii definiti pentru aceasta clasa (prima) urmate de diferite operatii puse la dispozitie de mediu.

1 Cand vom mai mentiona click dreapta in acest tutorial, utilizatorii de Macintosh trebuie sa foloseasca *ctrl+click*

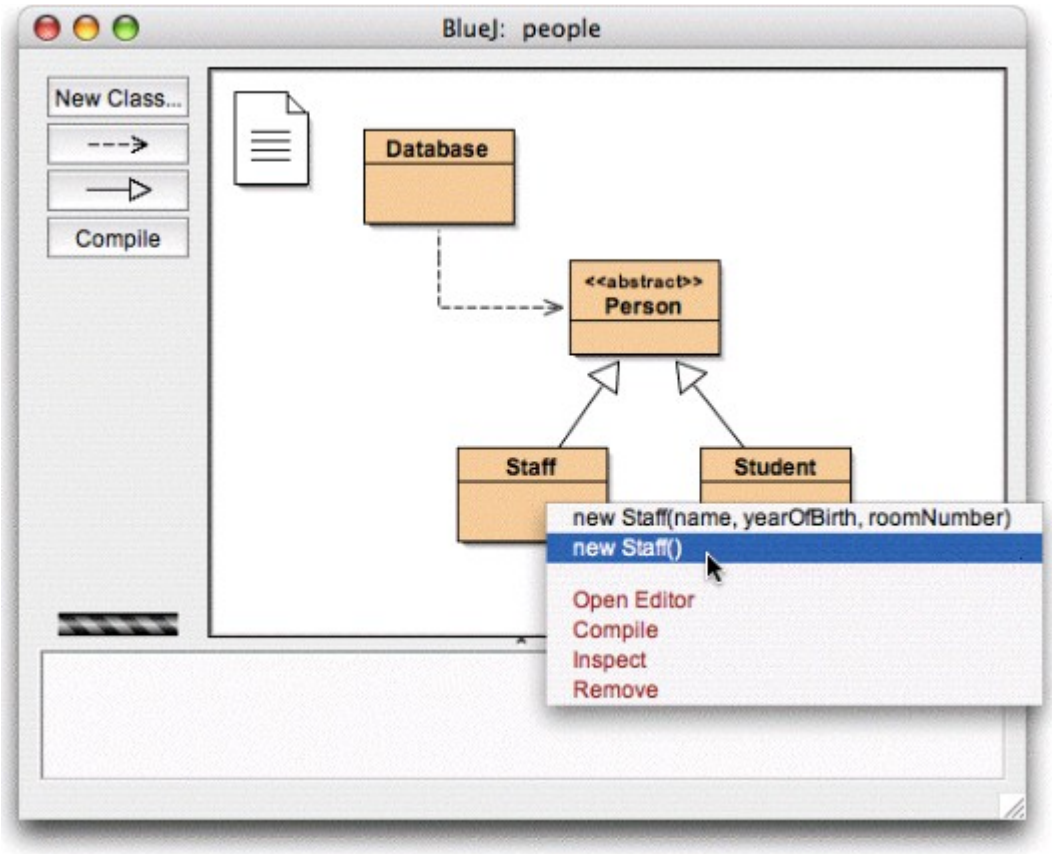


Fig 2. Operatiuni cu clasa (meniu pop-up)

Dorim sa creem un obiect *Staff* asa ca trebuie sa dam click dreapta pe pictograma *Staff* (ceea ce duce la afisarea meniului din figura 2). Meniul indica doi constructori pentru a creea un obiect *Staff*, unul cu parametrii si altul fara. Mai intai alegeti constructorul fara parametrii. Va aparea fereastra din figura 3.

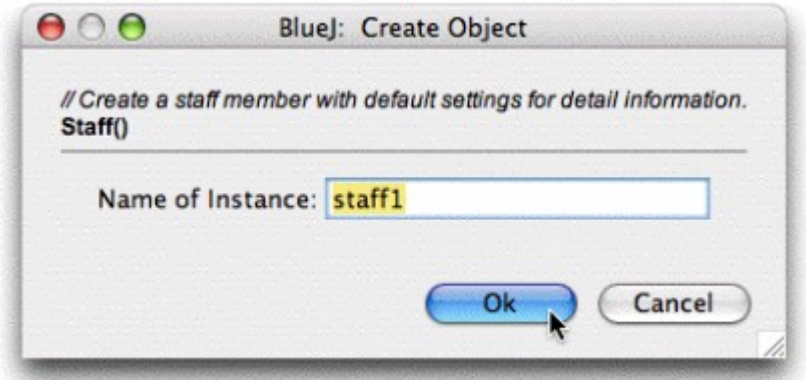


Fig 3. Crearea obiectelor fara parametrii

Fereastra de dialog va cere un nume pentru obiectul creat. In acelasi timp un nume implicit (*staff1*) este sugerat. Numele implicit este suficient de bun pentru acum, asa ca apasati Ok. Un obiect *Staff* va fi creeat.

Imediat ce obiectul este creeat, el va fi plasat in object bench (Fig 4). La asta se rezuma crearea obiectelor: selectati un constructor din meniul clasei, executati-l si va alegeti

cu un obiect in object bench.

Fig 4. Un obiect in object bench

Probabil ca ati observat ca clasa *Person* e etichetata <<abstract>> (e o clasa abstracta). Veti observa (daca incercati) ca nu puteti creea obiecte din clase abstracte (specificatia Java interzice acest lucru).

3.4 Executia

Rezumat: Pentru a executa o metoda, selectati-o din meniul pop-up al obiectului.

Acum ca ati creeat un obiect, puteti sa-i executati metodele publice. Dati un click cu butonul drept al mouse-ului pe obiect si un meniu cu metodele obiectului va aparea (Fig 5). Meniul indica metodele disponibile pentru acest obiect si doua operatii speciale specificate de mediu (*Inspect* si *Remove*). Acestea vor fi discutate mai tarziu. Mai intai sa ne concentram asupra metodelor.

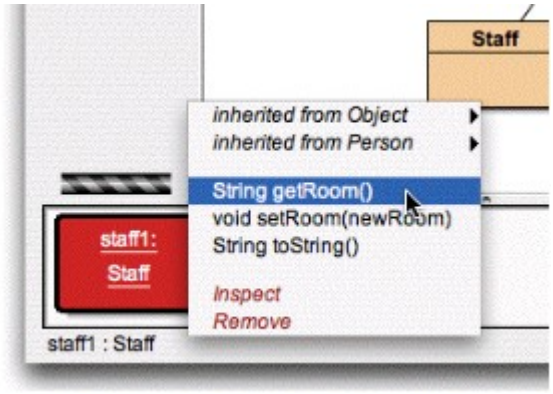


Fig 5. Meniul obiectului

Se vede ca exista metodele *setRoom()* si *getRoom()* care seteaza si returneaza numarul camerei pentru acest membru *Staff*. Incercati sa apelati *getRoom()*. Selectati-o din meniul obiectului si ea va fi executata. O fereastra de dialog va aparea si va afisa rezultatele apelului (Fig 6). In acest caz este returnat *"unknown room"* pentru ca nu am specificat o camera pentru aceasta persoana.

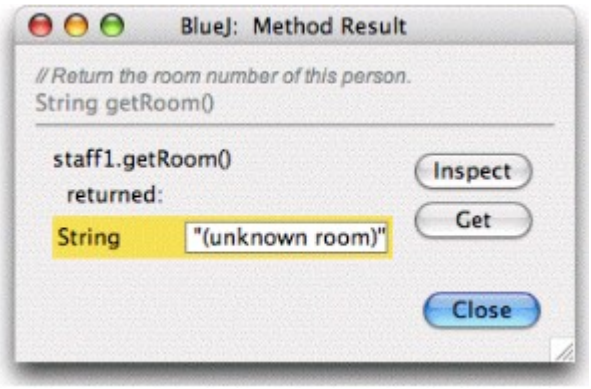


Fig 6. Afisarea rezultatului unei metode

Metodele mostenite de la o superclasa sunt disponibile printr-un submeniu. Deasupra meniului pop-up pentru obiect sunt doua submeniuri, unul pentru metode mostenite de la *Object* si altul pentru metode mostenite de la *Person* (fig 5). Puteti apela metode ale clasei *Person* (cum ar fi *getName()*) selectandu-le din submeniu. Incercati asta. Veti observa ca raspunsul este la fel de vag: raspunsul este *"unknown name"* deoarece nu am dat persoanei noastre un nume.

Acum sa incercam sa specificam un numar al camerei. Acest lucru va indica cum sa apelam o metoda cu parametrii. (Apelurile catre *getRoom()* si *getName()* aveau tip de date de intoarcere, dar nu aveau parametrii). Apelati metoda *setRoom()* selectand-o din meniu. Va aparea o casuta de dialog care va cere un parametru (fig 7).

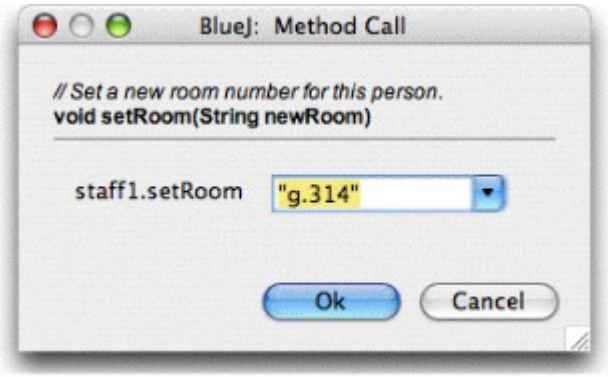


Fig 7: Fereastra pentru apelul metodei cu parametrii

Deasupra, fereastra indica interfata metodei ce va fi chemata (incluzand comentariul si semnatura). Mai jos exista un camp de text in care poate fi introdus parametrul. Semnatura de deasupra ne informeaza ca se asteapta un parametru de tip *String*. Introduceti valoarea pentru o camera noua inclusiv ghilimelele si apasati Ok.

Gata. Deoarece metoda nu returneaza nimic, nu mai apare fereastra cu rezultatele. Apelati *getRoom()* din nou pentru a vedea ca s-a facut o schimbare.

Experimentati o vreme cu crearea obiectelor si apelul metodelor. Incercati sa apelati un constructor cu parametrii si apelati mai multe metode pana va familiarizati cu aceste operatii.

3.5 Editarea unei clase

Rezumat: Pentru a edita sursa unei clase, dati dublu click pe pictograma clasei.

Pana acum am lucrat doar cu interfata unui obiect. Acum e momentul sa privim inauntru. Puteti sa aveti acces la codul clasei selectand *Open Editor* din meniul clasei (dand click dreapta pe pictograma clasei se obtine meniul clasei). Daca dati dublu click pe pictograma clasei veti apela automat *Open Editor*. Editorul nu este prezentat in mare detaliu in acest tutorial, dar se presupune ca e usor de folosit. Detalii legate de editor vor fi prezentate mai tarziu. Pentru moment deschideti implementarea clasei *Staff*. Gasiti implementarea metodei *getRoom()*. Ea returneaza, asa cum sugereaza numele, numarul camerei in care este obiectul. Sa modificam metoda adaugand prefixul "*room"* rezultatului metodei (pentru ca metoda sa returneze, sa zicem "room M.3.18" in loc de "M.3.18"). Putem face acest lucru modificand linia

```
return room;
```

la

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

BlueJ suporta Java nativ, asa ca nu exista nici o particularitate asupra cum va implementati clasele.

3.6 Compilarea

Rezumat: Pentru a compila o clasa, apasati butonul Compile *din editor. Pentru a compila un proiect, apasati* Compile *din fereastra proiectului.*

Dupa ce ati inserat textul, si inainte de a face altceva verificati proiectul principal (fereastra principala). Puteti observa ca pictograma pentru clasa *Staff* s-a modificat: acum este dungata. Acest mod de reprezentare indica clasele care nu au fost recompilate de la ultima modificare. Intorceti-va la editor.

NOTA: *Probabil ca va intrebati de ce pictogramele claselor nu erau dungate cand ati incarcat proiectul. Acest lucru se datoreaza faptului ca clasele din proiectul people au fost distribuite deja compilate. De obicei proiectele BlueJ sunt distribuite necompilate, asa ca obisnuiti-va sa vedeti pictogramele claselor dungate de acum inainte*

In bara de butoane a editorului sunt reprezentate functiile cele mai des folosite. Una dintre ele este *Compile*. Acest buton va permite sa compilati clasa direct din editor. Acum apasati pe *Compile.* Daca nu ati facut nici o greseala trebuie sa apara un mesaj in zona informationala a editorului, instiintandu-va ca nu este nici o eroare. (In cazul in care compilarea a mers cu succes, introduceti o eroare in codul sursa – cum ar fi un ';' lipsa doar ca sa vedeti cum arata).

Dupa ce ati compilat clasa cu succes, inchideti editorul.

NOTA: *Nu e nevoie sa salvati explicit sursa java. Sursele sunt salvate automat cand mediul considera ca e necesar (de ex cand se inchide editorul, sau cand e compilata clasa). Puteti salva explicit daca doriti (este o functie in meniul* Class *al editorului), dar nu este necesar decat daca sistemul dvs este foarte nesigur si se blocheaza frecvent, iar dvs doriti sa nu pierdeti modificarile.*

Bara de butoane a ferestrei proiectului are un buton *Compile*. Aceasta operatiune compileaza tot proiectul. (De fapt afla care clase trebuiesc recompilate si le compileaza in ordinea impusa). Incercati acest lucru modificand doua-trei clase (astfel incat sa apara hasurate in diagrama de clase) si apoi apasati butonul *Compile*. Daca se detecteaza o eroare intr-una din clasele compilate, editorul va fi deschis si se va afisa linia eronata si va fi afisat un mesaj de eroare.

Puteti observa ca zona 'object bench' este goala din nou. Obiectele sunt eliminate de fiecare data cand se schimba implementarea.

3.7 Ajutor la erorile de compilare

Rezumat: Pentru a obtine ajutor in cazul unei erori de compilare, apasati pe semnul intrebarii de langa mesajul de eroare.

Destul de des studentii incepatori au problemele in a intelege mesajele de eroare furnizate de compilator. Noi incercam sa oferim putin ajutor suplimentar.

Deschideti editorul din nou, introduceti o eroare in fisierul sursa si compilati. Un mesaj de eroare va fi afisat in bara de informatii a editorului. In partea dreapta a zonei informationale apare un semn de intrebare ce poate fi apasat pentru a afla mai multe informatii despre eroare (Fig 8).

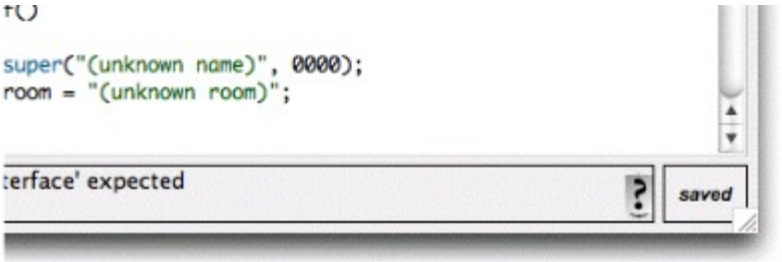


Fig 8: O eroare de compilare si butonul *Help*

In acest moment nu au fost scrise texte pentru toate mesajele de eroare. Unele texte explicative trebuie sa fie scrise. Dar merita sa incercati – erorile uzuale au texte explicative. Restul textelor vor fi adaugate pe parcurs, in versiunile urmatoare de BlueJ.

4 Facand mai multe...

In aceasta sectiune vom prezenta o serie de lucruri ce pot fi facute in mediul de dezvoltare. Aceste lucruri nu sunt esentiale, dar sunt folosite adesea.

4.1 Inspectie

Rezumat: Inspectia permite efectuarea unui debugging simplu, afisand starea interna a obiectelor.

Cand executati metode ale unui obiect, puteti observa operatiunea *Inspect* care este disponibila pentru obiecte, in afara metodelor definite de utilizator (Fig 5). Aceasta operatie permite verificarea starii interne a variabilelor de instanta (“campuri”) ale obiectelor. Incercati sa creeati un obiect cu niste valori definite de utilizator (de ex un obiect *Staff* instantiat cu constructorul care ia parametrii). Apoi alegeti *Inspect* din meniul obiectului. Apare o casuta de dialog care prezinta campurile obiectului, tipurile lor si valorile lor (Fig 9).

Fig 9: Fereastra de dialog *Object Inspector*

Inspectarea e folosita pentru a afla rapid daca o operatie mutator (o operatie care schimba starea interna a obiectului) a fost executata corect. Astfel, inspectia e o unealta simpla pentru debugging.

In exemplul *Staff* toate campurile sunt tipuri simple (ori nu sunt obiecte, ori sunt string-uri). Valoarea acestora poate fi afisata direct. Se poate observa cu ochiul liber daca constructorul a facut asocierile necesare.

In cazuri mai complexe, valorile campurilor pot fi referinte catre obiecte create de utilizator. Pentru a vedea un asemenea exemplu, vom folosi un alt proiect. Deschideti proiectul *people2*, care este inclus de asemenea in distributia standard BlueJ. Clasele proiectului *people2* sunt prezentate in Fig 10. Cum puteti observa, acest exemplu are in plus o clasa *Address* pe langa cele prezentate inainte. Unul din campurile din *Person* e de tipul *Address*.

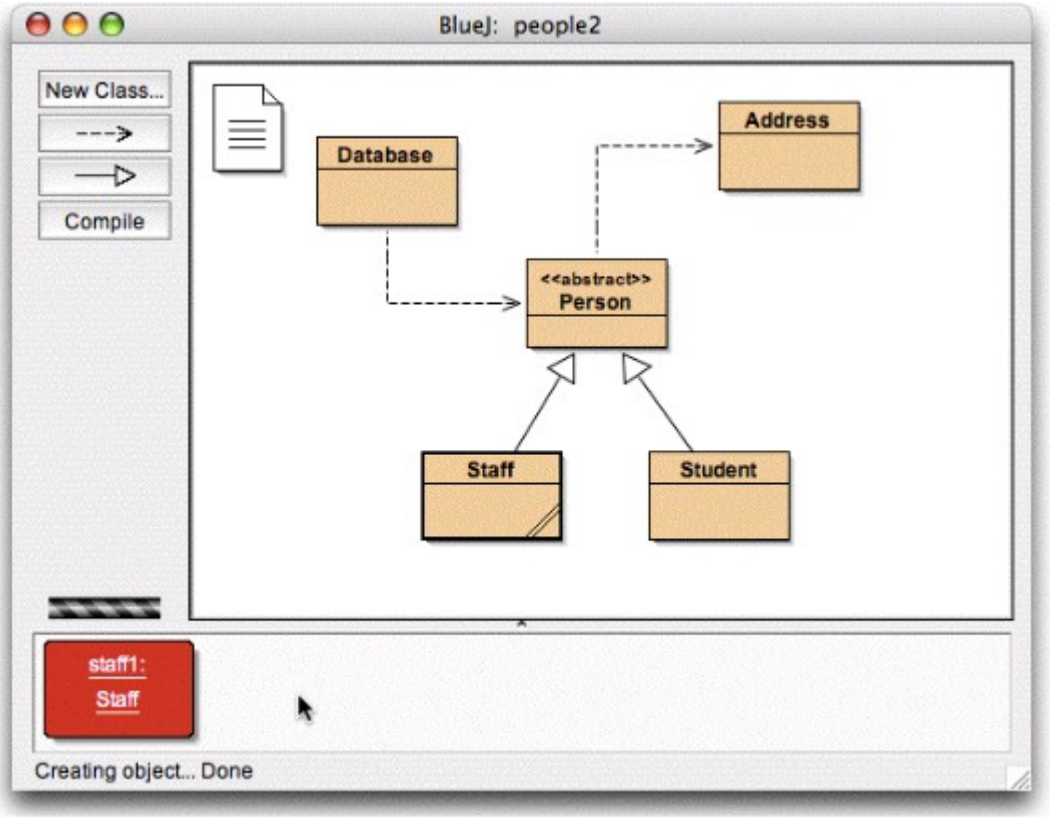


Figura 10: Fereastra proiectului *people2*

Urmatorul lucru pe care vrem sa-l incercam – inspectarea obiectelor cu alte obiecte in campuri – necesita crearea unui obiect de tip *Staff* si apelarea metodei *setAddress()* a acestui obiect (o veti gasi in submeniul *Person*). Introduceti o adresa. In interior codul clasei *Staff* creeaza un obiect de clasa *Address* si il asociaza variabilei *address*.

Acum inspectati obiectul *Staff*. Fereastra de dialog afisata e prezentata in Fig 11. Campurile din obiectul *Staff* includ acum si campul *address.* Precum vedeti, valoarea sa e reprezentata de o sageata, ceea ce reprezinta o referinta catre un alt obiect. Deoarece reprezinta un obiect complex, definit de utilizator, valoarea sa nu poate fi prezentata direct in lista. Pentru a examina in continuare adresa, apasati pe campul *address* si apoi pe butonul *Inspect.* (De asemenea, puteti da dublu click pe campul *address*). Va fi deschisa alta fereastra de inspectie, aratand detaliile obiectului *Address* (Fig 12).

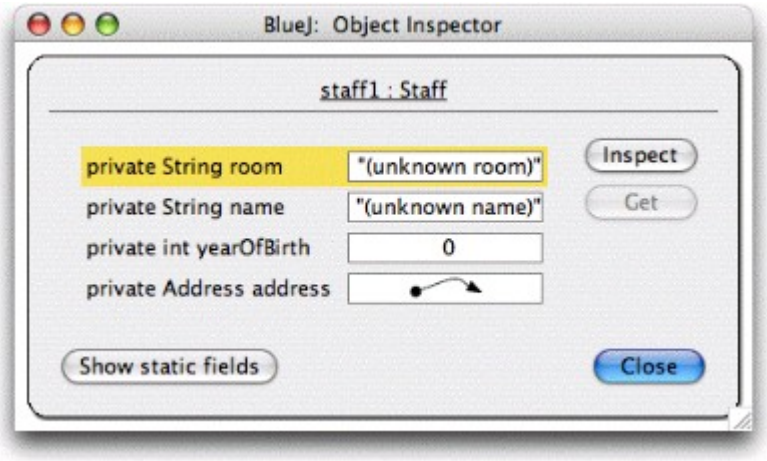


Fig 11: Inspectarea unui obiect cu referinte

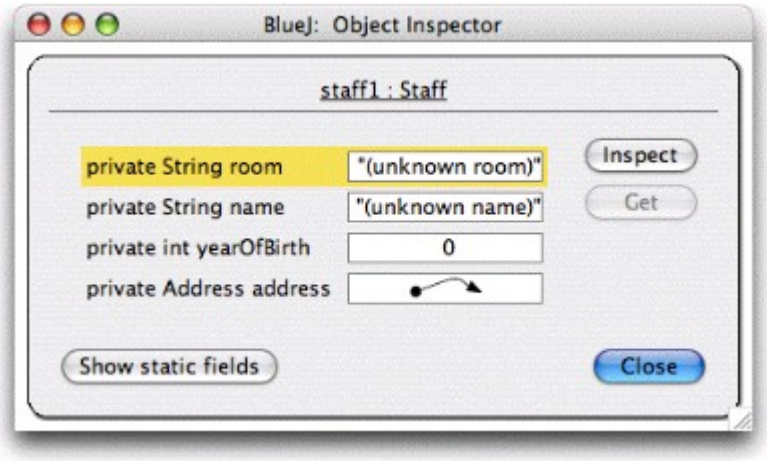


Fig 12: Inspectarea unui obiect intern

Daca campul selectat este public, in loc de a da click pe *Inspect* puteti selecta campul de adresa si sa apasati butonul *Get*. Aceasta operatie aduce obiectul selectat in 'object bench'. De acolo il puteti examina mai in detaliu, apelandu-i metodele.

4.2 Trimiterea obiectelor ca parametrii

Rezumat: Un obiect poate fi trimis ca parametru al unei metode dand click pe pictograma obiectului

Obiectele pot fi trimise ca parametrii metodelor altor obiecte. Sa incercam urmatorul lucru. Creati un obiect al clasei *Database*. (Veti observa ca clasa *Database* nu are decat un singur constructor fara parametrii, deci crearea obiectului nu ar trebui sa puna probleme). Obiectul *Database* are abilitatea de a stoca o lista de persoane. Are operatii de adaugare de persoane si de listare a tuturor persoanelor memorate. (Numirea aceste clase *Database* este totusi o exagerare!).

Daca nu aveti deja un obiect *Staff* sau *Student* in object bench, creati unul. In continuare aveti nevoie de un obiect *Database* si de un obiect *Staff* sau *Student*.

Acum apelati metoda *addPerson()* al obiectului *Database.* Semnatura va spune ca se asteapta un obiect de tip *Person*. (Retineti: clasa *Person* e abstracta, asa nu exista obiecte directe de tip *Person*. Dar, din cauza mostenirii obiectele *Student* si *Staff* pot tine locul unui obiect *Person*. Asadar este corect sa trimitem ca parametru un obiect *Student* sau *Staff* acolo unde se asteapta un obiect *Person*). Pentru a trimite un obiect deja existent in object bench unei metode, ii puteti tasta numele, sau mai scurt, puteti da click pe obiectul respectiv. Acest lucru ii introduce numele in fereastra de dialog. Apasati *Ok* si se face apelul metodei. Deoarece aceasta metoda nu are tip de date de intoarcere, nu vedem imediat un rezultat. Puteti apela metoda *listAll()* a obiectului *Database* pentru a verifica ca actiunea a fost indeplinita. Metoda *listAll()* afiseaza continutul la iesirea standard. Veti observa ca un terminal de text se va deschide automat.

Incercati acest lucru din nou cu mai mult de o persoana introdusa in “baza de date”.

5 Crearea unui proiect nou

Acest capitol va face un tur rapid in crearea proiectelor noi.

5.1 Crearea directorului proiectului

Rezumat: Pentru a creea un proiect selectati New... *din meniul* Project.

Pentru a creea un proiect nou selectati *Project – New...* din meniu. O fereastra de dialog se va deschide si va lasa sa specificati numele si locatia unde va fi salvat proiectul. Dupa ce dati Ok, un director va fi creeat cu numele specificat, iar fereastra principala va afisa un proiect nou gol.

5.2 Crearea claselor

Rezumat: Pentru a creea o clasa, apasati butonul New Class *si specificati un nume de clasa.*

Puteti creea clase noi apasand butonul *New Class* din bara de butoane a proiectului. Veti fi intrebat care este numele acestei clase – acest nume trebuie sa fie un identificator valid Java.

Puteti de asemenea sa alegeti intre patru tipuri de clase: abstracte, interfete, applet sau “standard”. Acest lucru impune ce cod de baza va fi scris initial in clasa dvs. Puteti modifica tipul clasei mai tarziu editand codul sursa (de exemplu adaugand cuvantul “abstract” in definitia clasei).

Dupa crearea unei clase, ea e reprezentata de o pictograma in proiect. Daca nu e o clasa standard, tipul (abstract, interfata sau applet) e indicat in pictograma. Daca vizualizati codul sursa al unei clase nou create, puteti observa ca contine un cod schelet care va permite sa va apucati rapid de munca. Codul implicit e corect din punct de vedere sintactic (dar nu face mai nimic). Incercati sa faceti mai multe clase si sa le compilati.

5.3 Crearea dependintelor

Rezumat: Pentru a creea o sageata, apasati simbolul sageata si trasati sageata in diagrama, sau doar scrieti codul sursa in editor.

Diagrama de clase arata dependenta intre clase prin sageti. Relatii de tip mostenire (“extends” sau “implements”) sunt indicate ca sageti cu capat gol; relatii de tip “foloseste” sunt indicate de sageti punctate cu capat gol.

Puteti adauga dependinte noi in mod grafic (direct in diagrama) sau textual in codul sursa. Daca adaugati o sageata in mod grafic, codul este actualizat; daca modificati codul,

diagrama este actualizata.

Pentru a adauga grafic o sageata, apasati butonul cu sageata (sageata goala pentru “extends” sau “implements” si sageata punctata pentru “foloseste”) si trageti sageata de la o clasa la cealalta.

Adaugarea unei sageti de mostenire adauga codul “extends” sau “implements” in codul sursa al clasei (in functie de tipul tintei: clasa sau interfata).

Adaugarea unei sageti de folosire nu modifica imediat codul sursa (decat daca tinta e o clasa din alt pachet. In acest caz se genereaza o comanda *import* dar nu am intalnit inca acest lucru in exemplele noastre). Existenta unei sageti de folosire in diagrama indicand o clasa care nu e folosita, va genera un avertisment mai tarziu spunand ca o relatie de tip “foloseste” a fost definita catre o clasa, dar acea clasa nu a fost folosita.

Adaugarea sagetilor textual e usor: tastati codul in mod normal. Imediat ce clasa e salvata, diagrama este actualizata. (Retineti ca inchiderea editorului duce la salvarea automata).

5.4 Eliminarea elementelor

Rezumat: Pentru a elimina o clasa sau o sageata, alegeti optiunea Remove *din meniul ei popup.*

Pentru a elimina o clasa din diagrama, selectati clasa si alegeti *Remove* din meniul *Edit.* Puteti selecta *Remove* si din meniul popup al clasei. Ambele optiuni functioneaza si pentru sageti: Puteti inlatura o sageata selectand-o iar apoi apeland *Remove* din meniul *Edit*, sau apeland *Remove* din meniul popup.

6 Utilizand 'code pad'

Unealta 'code pad' din BlueJ permite analizarea rapida si usoara a unor diferite bucatele de cod. Astfel code pad poate fi folosit pentru a investiga detaliile semanticii Java si pentru a ilustra si experimenta sintaxa Java.

6.1 Afisarea 'code pad'

Rezumat: Pentru a incepe sa folositi code pad, selectati Show code pad *din meniul* View.

Code pad nu e afisat implicit. Pentru a-l afisa folositi *Show code pad* din meniul *View*. Fereastra principala va include interfata code pad in coltul din dreapta jos, langa object bench (Fig 13). Limitele orizontale si verticale pot fi modificate pentru a schimba dimensiunea code pad sau a object bench.

Zona code pad poate fi folosita acum pentru a insera expresii sau blocuri de prelucrare. Cand se apasa *Enter* fiecare linie e evaluata si e posibil sa apara un rezultat.

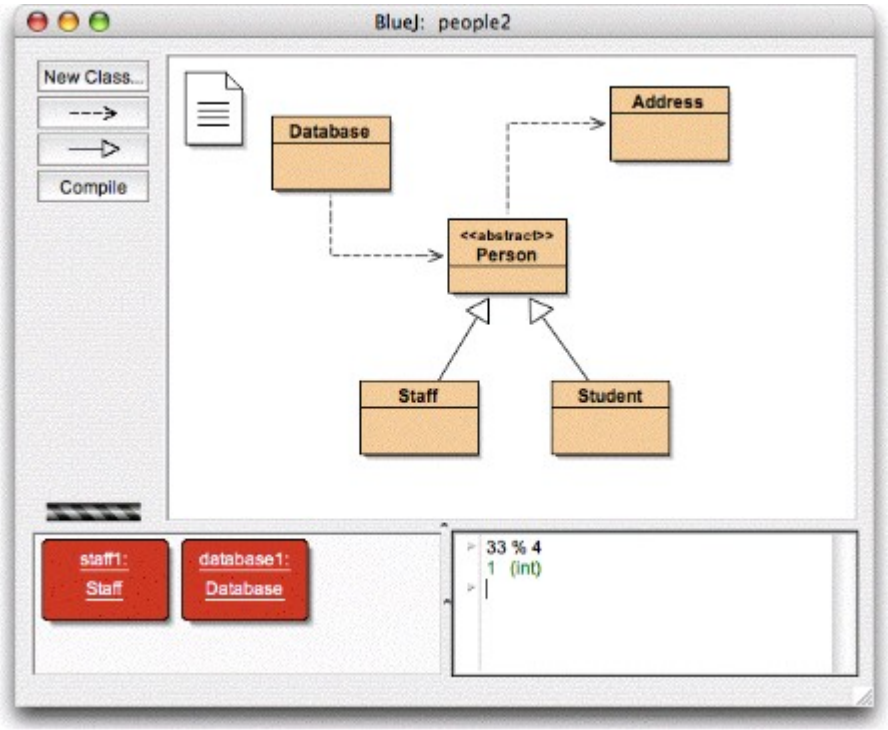


Fig 13: Fereastra principala impreuna cu zona code pad

6.2 Evaluarea expresiilor simple

Rezumat: Pentru a evalua expresii Java, introduceti-le in code pad.

Zona code pad poate fi folosita pentru a evalua expresii simple. Incercati sa 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 Java standard si obiecte, dar si la clase din proiectul curent. Zona code pad va afisa valoarea de intoarcere, dar si tipu acesteia (in paranteza), sau un mesaj de eroare daca expresia e incorecta.

Puteti folosi si obiectele pe care le aveti in object bench. Incercati urmatoarele: pozitionati un obiect de tip *Student* in object bench (folosind meniul popup al clasei, asa cum am descris anterior). Numiti-l *student1*.

In code pad puteti tasta:

```
student1.getName()
```

In mod similar puteti apela toate metodele disponibile din clasele proiectului dvs.

6.3 Primirea obiectelor

Rezumat: Pentru a transfera obiecte din code pad in object bench, trageti de pictograma obiectului.

Unele rezultate ale expresiilor sunt obiecte, si nu simple primitive. In acest caz, rezultatul e afisat ca <*object reference*>, urmat de tipul obiectului si de o mica pictograma desenata langa linia cu rezultat (Fig 14).

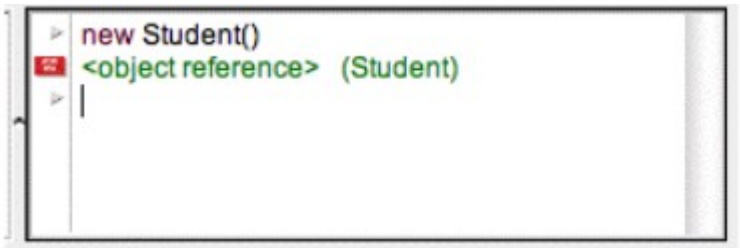


Fig 14: Un obiect rezultat in urma unei operatii in code pad

Daca valoarea returnata e un string, atunci valoarea ei va fi afisata ca rezultat, dar veti vedea si o pictograma mica, deoarece string-urile sunt obiecte.

Puteti incerca urmatoarele expresii pentru a creea obiecte:

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

Mica pictograma rezultata poate fi folosita pentru a lucra in continuare cu obiectul. Puteti trage pictograma peste object bench (Fig 15). Aceasta actiune va pozitiona obiectul in object bench, de unde i se vor putea apela metodele prin meniul popup sau din code pad.

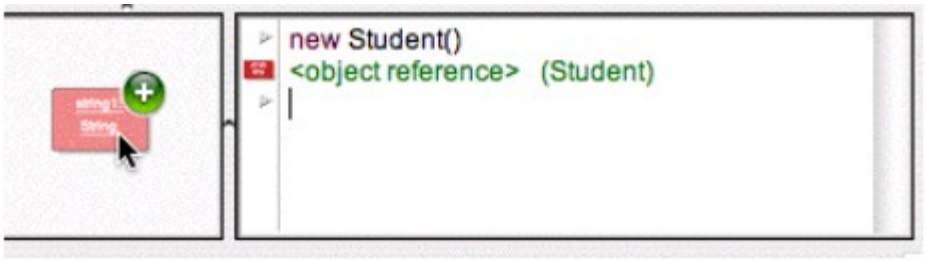


Fig 15: Pozitionarea obiectului peste object bench

6.4 Inspectarea obiectelor

Rezumat: Pentru a inspecta obiecte in code pad, dati dublu click pe mica pictograma

Daca doriti sa inspectati un obiect care a fost returnat in urma unei expresii din code pad, puteti face asta fara sa-l plasati in object bench: puteti da dublu click pe pictograma obiectului pentru a deschide fereastra de inspectie.

6.5 Executia comenzilor

Rezumat: Comenzile tastate in code pad sunt executate.

Puteti folosi code pad pentru a executa comenzi (adica instructiuni Java care nu returneaza o valoare). Incercati aceste comenzi:

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

Comenzile sunt evaluate corect fie ca au sau nu punct si virgula la sfarsit.

6.6 Comenzi multilinie si secvente de comenzi

Rezumat: Folositi Shift + Enter *la sfarsitul unei linii pentru a introduce comenzi pe mai multe linii.*

Puteti introduce secvente de comenzi, sau comenzi scrise pe mai multe linii folosind *Shift + Enter* la sfarsitul unei linii. Folosind *Shift + Enter* veti pozitiona cursorul la inceputul unei linii, dar nu va executa (inca) comanda. La sfarsitul ultimei linii, apasati *Enter* pentru a evalua toate liniile impreuna. Incercati, de exemplu, o bucla *for*:

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

6.7 Lucrul cu variabilele

Rezumat: Variabilele locale pot fi folosite intr-o singura comanda multilinie. Numele obiectelor din object bench e folosit drept campuri de instanta.

Variabilele – campuri de instanta sau variabile locale – pot fi folosite in code pad intr-o maniera restransa.

Puteti declara variabile locale in code pad, dar nu sunt utile decat in comenzi multilinie, deoarece variabilele se pierd intre linii succesive. De exemplu, puteti introduce acest bloc ca o comanda multilinie si el va functiona asa cum va asteptati:

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

Daca introduceti acelasi cod ca si linii individuale, nu va produce un rezultat, deoarece variabila *sum* nu este retinuta dupa ce ati introdus prima linie.

Puteti considera ce tastati pe o linie ca fiind corpul unei metode. Tot codul ce poate fi scris in corpul unei metode, poate fi scris si in code pad. Totusi, fiecare linie noua (separata de *Enter*) formeaza o metoda noua, asa ca nu puteti face referiri la o variabila declarata intr-o linie anterioara.

Puteti considera obiectele din object bench ca si campuri de instanta. Nu puteti defini noi campuri de instanta din interiorul corpului unei metode (sau din code pad), dar puteti apela obiectele si campurile acestora.

Puteti creea o instanta noua creeand un obiect in code pad si tragandu-l in object bench.

6.8 Istoricul liniei de comanda

Rezumat: Folositi sageata-sus *si* sageata-jos *pentru a cicla prin instoricul liniei de comanda.*

Code pad retine comenzile introduse. Folositi sageata *sus* si *jos* pentru a rescrie rapid o linie anterioara, ce poate fi modificata inainte de a fi relansata in executie.

7 Debugging

Aceasta sectiune introduce cele mai importante aspecte ale debugging-ului in BlueJ. In discutiile noastre cu profesorii care predau programarea am constatat ca este de dorit sa se predea un instrument de debugging in primul an, dar de multe ori nu mai este timp pentru asta. Studentii se chinuie cu editorul, compilatorul si cu executia – nu mai au timp si pentru un instrument complicat.

De aceea am hotarat sa facem debuger-ul cat mai simplu cu putinta. Telul e sa ai un debugger pe care il poti explica in 15 min, si care sa poata fi folosit de studenti de acum inainte fara ajutor suplimentar. Sa vedem daca am reusit.

In primul rand, am redus functionalitatea debuger-ilor traditionali la trei elemente:

- setarea de breakpoint-uri
- executia pas cu pas
- inspectarea variabilelor

In schimb, fiecare din acest element este foarte simplu. Acum vom verifica fiecare element.

Pentru a incepe, deschideti proiectul *debugdemo*, care este inclus in directorul *examples* din distributie. Acest proiect include cateva clase menite sa demonstreze functionalitatea debuger-ului – altfel nu fac nimic util.

7.1 Setarea de breakpoint-uri

Rezumat: Pentru a seta un breakpoint, dati click in zona de breakpoint din stanga textului, din editor.

Setarea unui breakpoint va permite sa opriti executia intr-un anumit punct din cod. Cand se intrerupe executia, puteti investiga starea obiectelor dvs. De regula va ajuta sa intelegeti ce se intampla in codul dvs.

In editor, in stanga textului este zona pentru breakpoint-uri (Fig 16). Puteti stabili un breakpoint dand un click acolo. Un simbol mic de stop va aparea sa marcheze acest lucru. Acum deschideti clasa *Demo*, gasiti metoda *loop* si puneti un breakpoint undeva in bucla *for*. Simbolul de stop ar trebui sa apara in editorul dvs.

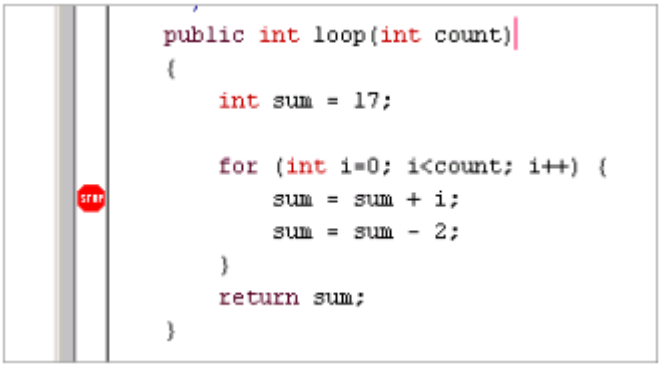


Fig 16: Un breakpoint

Cand se atinge linia care are breakpoint-ul atasat, exeutia va fi intrerupta. Sa incercam acest lucru.

Creati un obiect din clasa *Demo* si apelati-i metoda *loop()* cu un parametru de, sa zicem 10. Imediat ce se atinge breakpoint-ul, se va deschide editorul indicand linia curenta de cod si se va deschide si fereastra debugger-ului. Veti vedea ceva ca in fig 17.

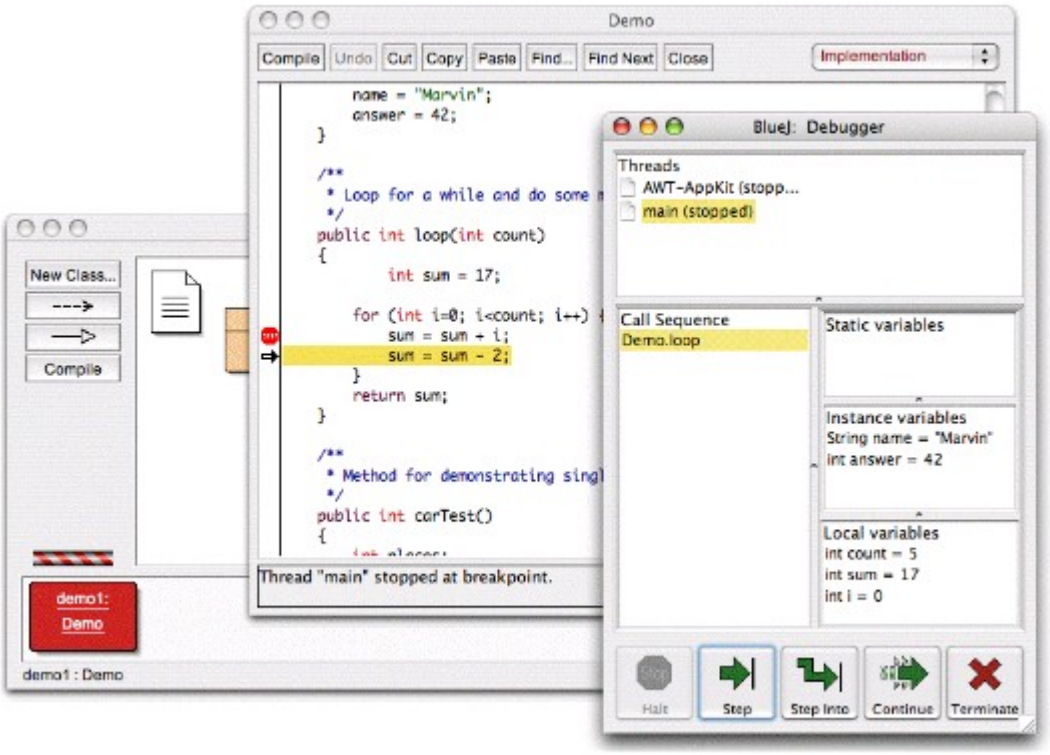


Fig 17: Fereastra debugger-ului

Linia subliniata din editor indica linia ce se va executa in continuare. (Executia s-a oprit inainte ca aceasta linie sa fie executata).

7.2 Parcurgerea codului pas cu pas

Rezumat: Pentru a parcurge codul pas cu pas, folositi butoanele Step *sau* Step Into *din debugger.*

Acum ca s-a oprit executia (ceea ce ne convinge ca metoda se executa si se poate ajunge la acest punct din cod), putem executa pas cu pas sa vedem ce se intampla. Pentru a face asta apasati in mod repetat pe butonul *Step* din fereastra debugger-ului. Ar trebui sa vedeti ca se modifica linia sursa din editor (linia subliniata se modifica in functie de linia ce se va executa). De fiecare data cand apasati *Step*, o singura linie de cod se va executa si executia se va opri. Observati si ca variabilele din debugger se modifica (de exemplu valoarea pentru *sum*). Asa ca puteti executa pas cu pas si veti observa ce se intampla. Dupa ce va plictisiti de asta, puteti apasa pe breakpoint din nou pentru a-l sterge, si apoi apasati butonul *Continue* din debugger pentru a continua executia normal.

Sa incercam acelasi lucru cu alta metoda. Stabiliti un breakpoint in clasa *Demo*, in metoda *carTest()*, la linia:

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

Apelati metoda. Cand se atinge breakpoint-ul, sunteti pe cale sa apelati o linie ce contine apleul catre metoda *seats()* din clasa *Car*. Daca apasati *Step* va trece peste toata linia. Sa incercam sa apasam *Step Into* de data asta. Daca apasati acest buton pentru o metoda, veti intra in metoda si veti executa linie cu linie metoda respectiva (nu intr-un singur pas). In acest caz sunteti dus in metoda *seats()* din clasa *Car*. Puteti parcurge intreaga metoda pana cand ajungeti la sfarsitu ei si va intoarceti la metoda apelanta. Observati ca debugger-ul afiseaza lucruri modificate.

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

7.3 Inspectarea variabilelor

Rezumat: Inspectarea variabilelor e usoara: ele sunt afisata automat in debugger.

Cand faceti debugging, este important sa puteti inspecta starea obiectelor (variabile locale sau de instanta).

Sa faceti acest lucru e trivial – deja ati vazut mare parte. Nu aveti nevoie de comenzi speciale pentru a inspecta variabilele; variabilele statice, de instanta ale obiectului curent si cele locale ale metodei curente sunt afisate automatic.

Puteti selecta metode din secventa de apeluri pentru a vedea variabilele altor obiecte si metode active in mod curent. Stabiliti de exemplu un breakpoint in metoda *carTest()* din nou. In partea stanga a debugger-ului puteti vedea secventa de metode. Momentan arata:

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

Acest lucru indica faptul ca *Car.seats* a fost apelat de *Demo.carTest.* Puteti selecta *Demo.carTest* din aceasta lista pentru a inspecta sursa si variabilele curente ale acestei

metode.

Daca treceti de linia cu *new Car(...)*, puteti observa ca valoarea variabilei *myCar* e indicata ca <*object reference*>. Toate variabilele de tip obiect (cu exceptia string-urilor) sunt afisate in acest mod. Puteti inspecta aceasta variabila, dand dublu click pe ea. Acest lucru va deschide o fereastra de inspectare a obiectelor identica cu cele descrise anterior (sectiunea 4.1). Nu e nici o diferenta in a inspecta obiecte aici sau a le inspecta in object bench.

7.4 Oprirea programului

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

Cateodata un program ruleaza de mult timp si va intrebati daca este ok. Poate ca a intrat intr-o bucla infinita sau poate ca e normal sa dureze atat. Pai putem verifica. Apelati metoda *longloop()* din clasa *Demo*. Executia codului va dura ceva vreme.

Acum vrem sa aflam ce se intampla. Afisati debugger-ul daca cumva nu e afisat. Acum apasati butonul *Halt*. Executia se opreste ca si cum ar fi ajuns la un breakpoint. Puteti parcurge cativa pasi, sa observati variabilele si sa va convingeti ca totul e in ordine si metoda are nevoie de ceva timp sa termine. Puteti apasa *Continue* si *Halt* de cateva ori pentru a vedea cat de repede numara. Daca nu vreti sa continuati (de exemplu, ati descoperit ca sunteti intr-o bucla infinita) puteti apasa *Terminate* pentru a opri intreaga executie. *Terminate* nu trebuie folosit prea des – pot ramane obiecte care au fost bine gandite intr-o stare inconsistenta, asa ca e recomandabil sa nu folositi aceasta optiune decat ca un mecanism de urgenta.

8 Crearea de aplicatii de sine statatoare

Rezumat: Pentru a creea aplicatii de sine statatoare folositi Project – Create Jar File...

BlueJ poate creea fisiere executabile jar. Aceste fisiere pot fi executate pe unele sisteme doar prin dublu click (de exemplu pe Windows si MacOS), sau prin apelarea comenzii *java -jar <numefisier.jar>* (in UNIX sau in DOS).

Vom incerca acest lucru cu proiectul *hello*. Deschideti-l (e in directorul *examples*). Asigurati-va ca ati compilat proiectul. Selectati *Create Jar File...* din meniul *Project.*

Va aparea o fereastra de dialog care va cere sa specificati clasa principala (Fig 18). Aceasta clasa trebuie sa aiba o metoda main valida (cu semnatura *public static void main(String[] args)*).

Fig 18: Fereastra "Create Jar File"

In exemplul nostru, alegerea unei clase principale e usor – avem o singura clasa. Alegeti *Hello* din meniul pop-up. Daca aveti alte proiecte, alegeti metoda main pe care doriti sa o executati.

In mod normal nu veti include sursele in proiectele executabile. Dar puteti, daca vreti, sa va distribuiti si sursele. (Puteti folosi proiectul jar pentru a trimite tot proiectul altcuiva prin e-mail, intr-un singur fisier).

Daca ati configurat BlueJ sa foloseasca bibliotecile user-ului (ori prin setarea *Preferences/Libraries* sau folosind directorul *lib/userlib*) veti vedea o zona numita *Include user libraries* in mijlocul ferestrei. (Daca nu folositi alte biblioteci, aceasta zona va lipsi). Trebuie sa bifati toate bibliotecile pe care le foloseste proiectul dvs.

Apasati *Continue*. In continuare veti vedea o fereastra de dialog in care puteti stabili numele fisierului jar pe care doriti sa-l creeati. Tastati *hello* si apoi *Create*.

Daca nu aveti alte biblioteci ce trebuiesc incluse, va fi creeat un fisier numit *hello.jar*. Daca aveti biblioteci de inclus, va fi creeat un director *hello* si in interiorul lui un fisier *hello.jar*. Directorul va contine si bibliotecile necesare. Fisierul jar se asteapta sa gaseasca bibliotecile

in aceeasi structura, asa ca aveti grija sa le distribuiti impreuna.

Puteti lansa in executie fisierul jar doar daca aplicatia foloseste o interfata GUI. Exemplul nostru foloseste o interfata text asa ca va trebui pornit din linia de comanda. Sa incercam sa rulam fisierul jar acum.

Deschideti un terminal si navigati pana in directorul unde e salvat fisierul jar (ar trebui sa vedeti un fisier *hello.jar*). Presupunand ca Java e instalat corect in 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 context al applet-ului.*

BlueJ permite crearea si rularea appleturilor si a aplicatiilor. Am inclus un applet in directorul *examples* al distributiei. Mai intai dorim sa incercam sa executam un applet. Deschideti proiectul *appletdemo* din exemple.

Veti observa ca acest proiect are o singura clasa numita *CaseConverter.* Puteti observa ca pictograma clasei e marcata cu <<*applet*>>. Dupa compilare selectati *Run applet* din meniul context al clasei.

Va aparea o fereastra de dialog ce va permite sa faceti niste selectii (Fig 19).

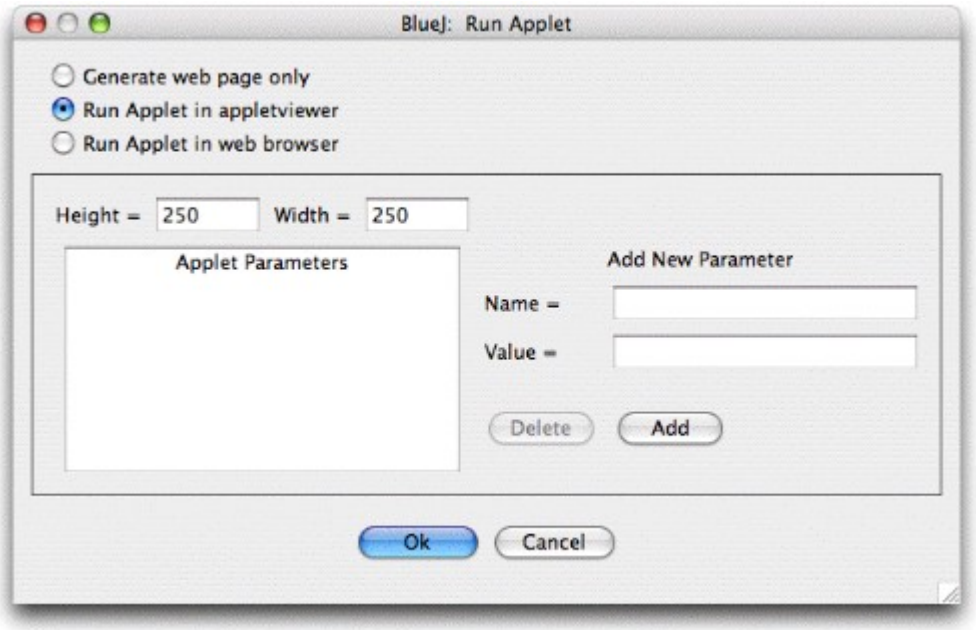


Fig 19: Fereastra de dialog “Run Applet”

Aici vedeti ca aveti de ales intre rularea applet-ului intr-un browser sau intr-un appletviewer (sau doar sa generati codul html fara sa rulati applet-ul). Lasati setarile implicite si apasati Ok. In cateva secunde trebuie sa porneasca un applet viewer cu applet-ul incarcat.

Appletviewer-ul e instalat odata cu J2SE SDK, asa ca e garantat sa aiba aceeasi versiune cu compilatorul dvs Java. Acest lucru genereaza de obiecei mai putine probleme decat browserele. Browser-ul dvs web poate rula o versiune diferita de java, si acest lucru poate duce la probleme. Totusi, ar trebui sa mearga bine cu browserele curente.

Pe Microsoft Windows si in MacOs, BlueJ foloseste browser-ul implicit. Pe sistemele UNIX browser-ul e definit in setari.

9.2 Crearea unui applet

Rezumat: Pentru a creea un applet, selectati butonul New Class *si apoi selectati* Applet *ca tip de clasa*

Dupa ce am vazut cum ruleaza un applet, vrem sa creeam unul.

Creati o clasa noua de tipul *Applet* (puteti stabili tipul in dialogul *New Class)*. Compilati si rulati applet-ul. Nu a fost asa de greu, nu-i asa?

Applet-urile (ca si alte clase) sunt generate cu un cod schelet valid. Pentru applet-uri, acest cod va afisa un applet simplu cu doua linii de text. Puteti deschide editorul si puteti insera propriul cod.

Veti vedea ca toate apelurile uzuale pentru applet-uri sunt aici, fiecare cu cate un comentariu care ii explica scopul. Codul care se executa e in metoda *paint()*.

9.3 Testarea applet-ului

In anumite situatii e util sa se creeze un obiect *Applet* in object bench (ca si obiectele normale). Puteti face acest lucru – constructorul e afisat in meniul context al clasei respective. Din object bench nu puteti rula intreg appletul, dar puteti rula unele metode. Acest lucru e util daca vreti sa testati metode scrise in implementarea applet-ului dvs.

Daca stabiliti breakpoint-uri in applet-ul dvs, ele nu vor avea efect cand rulati applet-ul intr-un browser sau appletviewer. Acest lucru se intampla deoarece si web browser-ul si appletviewer-ul folosesc masini java diferite pentru a rula applet-ul si acestea nu inteleg breakpoint-urile stabilite de BlueJ.

Daca doriti sa folositi breakpoint-uri si executie pas cu pas intr-un applet, puteti folosi clasa *AppletWindow*, scrisa de Michael Trigoboff. Clasa va pune la dispozitie un cadru in care sa rulati applet-ul direct din BlueJ, astfel incat sa poata fi aplicate metodele clasice de debugging. Puteti gasi aceasta clasa si un demo in sectiunea *Resurse* de pe site-ul BlueJ.

10 Alte operatii

10.1 Deschiderea pachetelor non-BlueJ in BlueJ

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

BlueJ va permite sa deschideti pachete existente creeate in afara mediului BlueJ. Pentru a face asta, selectati *Project – Open non BlueJ...* din meniu. Selectati directorul care contine fisierele sursa Java si apoi apasati butonul *Open in BlueJ.* Sistemul va cere confirmarea ca doriti sa deschida acel director.

10.2 Adaugarea claselor existente in proiect

Rezumat: Clase pot fi incarcate din exteriorul unui proiect folosind comanda Add Class from File...

Adeseori doriti sa folositi o clasa intr-un proiect BlueJ pe care ati obtinut-o din alta sursa. De exemplu, un profesor da o clasa Java studentilor sa o foloseasca intr-un proiect. Puteti introduce usor o clasa existenta intr-un proiect selectand *Edit – Add Class from File...* Astfel puteti selecta si importa o sursa java (cu extensia *.java*).

Cand se importa o clasa externa, i se face o copie si se pastreaza in directorul curent. Are acelasi efect ca si cand ati fi creat clasa si ati fi scris tot codul ei sursa.

Alternativa e sa adaugati codul sursa al clasei din afara BlueJ. Data viitoare cand deschideti proiectul, noua clasa va aparea in diagrama.

10.3 Apelul main() si a altor metode statice

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

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

Daca dati click dreapta pe clasa veti observa ca meniul nu contine doar constructorul dar si metoda statica main(). Puteti apela acum metoda main din acest meniu (fara sa creeati mai intai un obiect).

Toate metodele statice pot fi apelate astfel. Metoda main standard asteapta un vector de elemente *String* ca argument. Puteti trimite ca parametru un vector de elemente *String* folosind sintaxa Java:

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

Incercati acest lucru!

NOTA: *In Java nu e permisa trimiterea ca parametru a unui vector pentru metode. Nu pot fi folositi decat ca initializatori. In BlueJ, pentru a putea apela metodele statice in mod interactiv, permitem pasarea ca argumente a vectorilor constanti.*

10.4 Generarea documentatiei

Rezumat: Pentru a genera documentatie pentru un proiect, selectati Project Documentation *din meniul* Tools.

Puteti genera documentatie in formatul standard *javadoc* din interiorul BlueJ. Pentru a face asta selectati *Tools – Project Documentation* din meniu. Aceasta functie va genera documentatia pentru toate clasele dintr-un proiect, bazandu-se pe codul lor sursa si va deschide un browser cu rezultatul final.

Puteti genera si vizualiza documentatia pentru o singura clasa direct in editorul BlueJ. Pentru a face acest lucru, deschideti editorul si folositi meniul lui. Schimbati selectia din *Implementation* in *Interface*. Acest lucru va arata documentatia in format *javadoc* (interfata clasei) in editor.

10.5 Lucrul cu biblioteci

Rezumat: Biblioteciile standard Java (Java API) pot fi vizualizate selectand Help – Java Class Libraries

In mod uzual, cand scrieti aplicatii Java, aveti nevoie de informatii din biblioteca standard. Puteti deschide un browser web cu API-ul Java selectand *Help – Java Class Libraries* din meniu (daca sunteti online).

Documentatia JDK poate fi instalata si folosita si offline. Detaliile sunt explicate in sectiunea Help de pe site-ul BlueJ.

10.6 Crearea obiectelor din clasele de biblioteca

Rezumat: Pentru a creea obiecte din clasele de biblioteca folositi Tools – Use Library Class

BlueJ ofera de asemenea facilitatea de a creea obiecte din clase care nu fac parte din proiect, dar tin de o biblioteca. De exemplu, puteti creea obiecte de tip *String* sau *ArrayList*. Acest lucru e util pentru o experimentare rapida cu aceste obiecte de biblioteca.

Puteti creea un obiect din clasele de biblioteca selectand *Tools – Use Library Class*. Va aparea o fereastra de dialog care va va cere numele complet al clasei, de exemplu *java.lang.String.* (Aveti grija pentru ca trebuie sa tastati numele complet al clasei, adica

numele clasei plus al pachetelor)

Casuta de text are un meniu asociat al claselor folosite anterior. Dupa ce ati tastat numele clasei si ati dat *Enter*, va fi afisata lista constructorilor disponibili si a metodelor statice ale clasei. Orice constructor sau metoda statica a clasei poate fi apelata selectand-o din lista.

Invocarea se petrece la fel cu orice apel de constructor.

11 Rezumat

Inceputul

1. *Pentru a deschide un proiect, selectati* Open *din meniul* Project.
2. *Pentru a creea un obiect, selectati un constructor din meniul pop-up al clasei.*
3. *Pentru a executa o metoda, selectati-o din meniul pop-up al obiectului.*
4. *Pentru a edita sursa unei clase, dati dublu click pe pictograma clasei.*
5. *Pentru a compila o clasa, apasati butonul* Compile *din editor. Pentru a compila un proiect, apasati* Compile *din fereastra proiectului.*
6. *Pentru a obtine ajutor in cazul unei erori de compilare, apasati pe semnul intrebarii de langa mesajul de eroare.*

Facand mai multe...

7. *Inspectia permite efectuarea unui debugging simplu, afisand starea interna a obiectelor.*
8. *Un obiect poate fi trimis ca parametru al unei metode dand click pe pictograma obiectului*

Crearea unui proiect nou

9. *Pentru a creea un proiect selectati* New... *din meniul* Project.
10. *Pentru a creea o clasa, apasati butonul* New Class *si specificati un nume de clasa.*
11. *Pentru a creea o sageata, apasati simbolul sageata si trasati sageata in diagrama, sau doar scrieti codul sursa in editor.*
12. *Pentru a elimina o clasa sau o sageata, alegeti optiunea* Remove *din meniul ei popup.*

Utilizand 'code pad'

13. *Pentru a incepe sa folositi code pad, selectati* Show code pad *din meniul* View.
14. *Pentru a evalua expresii Java, introduceti-le in code pad.*
15. *Pentru a transfera obiecte din code pad in object bench, trageti de pictograma obiectului.*
16. *Pentru a inspecta obiecte in code pad, dati dublu click pe mica pictograma*
17. *Comenzile tastate in code pad sunt executate.*
18. *Folositi* Shift + Enter *la sfarsitul unei linii pentru a introduce comenzi pe mai*

multe linii.

19. *Variabilele locale pot fi folosite intr-o singura comanda multilinie. Numele obiectelor din object bench e folosit drept campuri de instanta.*

20. *Folositi* sageata-sus *si* sageata-jos *pentru a cicla prin instoricul liniei de comanda.*

Debugging

21. *Pentru a seta un breakpoint, dati click in zona de breakpoint din stanga textului, din editor.*

22. *Pentru a parcurge codul pas cu pas, folositi butoanele* Step *sau* Step Into *din debugger.*

23. *Inspectarea variabilelor e usoara: ele sunt afisata automat in debugger.*

24. *Comenzile* Halt *si* Terminate *pot fi folosite pentru a opri executia programului temporar sau permanent.*

Crearea de aplicatii de sine statatoare

25. *Pentru a creea aplicatii de sine statatoare folositi* Project – Create Jar File...

Crearea applet-urilor

26. *Pentru a rula un applet, selectati* Run applet *din meniul context al applet-ului.*

27. *Pentru a creea un applet, selectati butonul* New Class *si apoi selectati* Applet *ca tip de clasa*

Alte operatii

28. *Pachetele non-BlueJ pot fi deschise cu comanda* Project – Open non BlueJ...

29. *Clase pot fi incarcate din exteriorul unui proiect folosind comanda* Add Class from File...

30. *Metodele statice pot fi apelate din meniul context al clasei.*

31. *Pentru a genera documentatie pentru un proiect, selectati* Project Documentation *din meniul* Tools.

32. *Biblioteciile standard Java (Java API) pot fi vizualizate selectand* Help – Java Class Libraries

33. *Pentru a creea obiecte din clasele de biblioteca folositi* Tools – Use Library Class