Il Tutorial per BlueJ

Versione 1.4
per BlueJ Versione 1.2.x

Michael Kölling
Mærsk Institute
University of Southern Denmark

Tradotto in italiano da Vito Di Viggiano, Orlando Demauro, Gianluca Leo, Gianpaolo Saracino
I.T.I.S. - L.S.T. "Oreste del Prete"
Sava - Taranto
Maggio 2002

INDICE

1 Introduzione

1.1 Su BlueJ

Questo tutorial è un'introduzione all'uso dell'ambiente di programmazione BlueJ.
BlueJ è un ambiente di sviluppo Java™ specificatamente progettato per l'insegnamento del linguaggio a livello introduttivo. E' stato progettato ed implementato dal team di BlueJ alla Monash University di Melbourne, Australia ed alla University of Southern Denmark di Odense.

Ulteriori informazioni su BlueJ sono disponibili all'indirizzo http://www.bluej.org.

1.2 Obiettivi e destinatari

Questo tutorial è indirizzato a coloro che vogliono familiarizzare da soli con le potenzialità dell'ambiente. Il tutorial non spiega lo scopo delle scelte fondamentali, alla base della costruzione dell'ambiente di sviluppo ovvero la ricerca dei problemi ad esso legati.

Questo tutorial non è stato realizzato per insegnare Java. I principianti della programmazione in Java sono invitati a studiare un testo introduttivo a Java o a seguire un corso su Java.

Non è un ampio manuale di riferimento dell'ambiente di sviluppo. Molti dettagli sono stati omessi – si è data importanza ad una breve e concisa introduzione, piuttosto che ad una completa descrizione delle caratteristiche.

Ogni sezione inizia con un breve sommario di una riga. Questo permette agli utenti già familiari con parti dell'ambiente di decidere se leggere o saltare ognuna delle singole parti. Il capitolo 10 ripete le righe dei vari sommari come riferimento rapido.

1.3 Diritti d'autore, licenze e ridistribuzione

1.4 Feedback

Commenti, domande, correzioni, critiche e qualunque altro tipo di osservazione riguardanti il sistema BlueJ o relativamente a questo tutorial sono ben accolte e attivamente incoraggiate.
Scrivete a Michael Kölling (mik@mip.sdu.dk).

2 Installazione

BlueJ è distribuito in tre differenti formati: uno per i sistemi Windows, uno per MacOS, ed uno per tutti gli altri sistemi. L'installazione è abbastanza semplice.

Prerequisiti

Per usare BlueJ devi aver installato sul tuo sistema il J2SE v1.3 (conosciuto come JDK 1.3) o versioni successive. Se non hai il JDK installato, puoi scaricarlo dal sito della Sun all'indirizzo http://java.sun.com/j2se/. Sul sistema operativo MacOS X, una versione recente del JDK è preinstallata - non hai bisogno di installarla. Se trovi una pagina web che ti permette di effettuare il download del "JRE" (Java Runtime Environment) e "SDK" (Software Development Kit), devi scaricare "SDK" - perché il JRE non è sufficiente.

2.1 Installazione su Windows

Il file di distribuzione per i sistemi Windows è chiamato *bluejsetup-xxx.exe*, dove *xxx* è il numero della versione. Per esempio, la versione 1.2.0 di BlueJ è chiamata *bluejsetup-120.exe*. Puoi prendere questo file da un disco o puoi scaricarlo dal sito di BlueJ all'indirizzo http://www.bluej.org. Esegui il programma di installazione.

Il programma di installazione ti permette di selezionare una cartella in cui installarlo. Esso, inoltre, ti offre la possibilità di creare sul desktop un collegamento al programma di avvio.

Dopo che l'installazione è terminata, troverai il programma *bluej.exe* nella cartella di installazione di BlueJ.

La prima volta che lanci BlueJ, esso cercherà un Java system (JDK). Se trova più Java system disponibili (es. se hai installato JDK 1.3.1 e il JDK 1.4), una finestra ti lascerà scegliere quale usare. Se non ne trova nessuno, ti verrà chiesto di cercarlo (questo può accadere quando un JDK system è stato installato, ma le voci corrispondenti nel registry sono state rimosse).

Il programma di installazione di BlueJ, installa anche un programma chiamato *vmselect.exe*. Utilizzando questo programma, puoi successivamente cambiare la versione di Java che userà BlueJ. Esegui *vmselect* per avviare BlueJ con una diversa versione di Java.

La scelta del JDK è memorizzata per ogni versione di BlueJ. Se hai installato versioni differenti di BlueJ, puoi usare una versione di BlueJ con il JDK 1.3.1 e un'altra versione di BlueJ con il JDK 1.4. Cambiando la versione di Java per BlueJ, si produrrà una variazione per tutte le installazioni di BlueJ della stessa versione per lo stesso utente.

2.2 Installazione su Macintosh

Fai attenzione, BlueJ verrà eseguito solo sul sistema MacOS X.

Il file di distribuzione per i sistemi MacOS è chiamato B*lueJ-xxx.sit*, dove *xxx* è il numero della versione. Per esempio, la versione 1.2.0 della distribuzione di BlueJ è chiamata B*lueJ-120.sit*. Puoi prendere questo file da un disco o puoi scaricarlo dal sito di BlueJ all'indirizzo http://www.bluej.org. Questo file può essere decompresso con *StuffIt Expander*. Molti browser decomprimeranno questo file in automatico. Altrimenti, un doppio click sull'icona dovrebbe decomprimerlo.

Dopo la decompressione, dovresti avere una cartella chiamata BlueJ-xxx. Sposta questa cartella nella tua cartella Applications (oppure in qualsiasi altro posto che ti piace). Non è necessaria nessuna ulteriore operazione di installazione.

2.3 Installazione su Linux/Unix e altri sistemi

Il file generale di distribuzione per questi sistemi è un file jar eseguibile. E' chiamato *bluej-xxx.jar dove xxx* è il numero della versione. Per esempio, la versione 1.2.0 della distribuzione di BlueJ è chiamata b*luej-120.jar*. Puoi prendere questo file da un disco o puoi scaricarlo dal sito di BlueJ all'indirizzo http://www.bluej.org.

Fai partire l'installazione eseguendo il seguente comando. NOTA: Per questo esempio, io uso il file della distribuzione *bluej-120.jar* – tu, invece, hai bisogno di usare il nome del file che possiedi (con il corretto nome della versione).

```
<jdk-path>/bin/java -jar bluej-120.jar
```

`<jdk-path>` è la cartella in cui è installato il JDK.

Viene visualizzata una finestra che ti permette di scegliere la cartella in cui installare BlueJ e la versione del JDK da utilizzare per l'esecuzione di BlueJ. Importante: il path di BlueJ (cioè, il nome della sua cartella e di quelle che la contengono) non deve contenere spazi.

Clicca su *Install*. Al termine dell'esecuzione, BlueJ dovrebbe essere installato.

2.4 Problemi d'installazione

Se incontri qualche problema, consulta le *Frequently Asked Questions* (FAQ) sul sito web di BlueJ (http://www.bluej.org/help/faq.html) e leggi la sezione *How To Ask For Help* (http://www.bluej.org/help/ask-help.html).

3 Per iniziare - scrivi / compila / esegui

3.1 Iniziare con BlueJ

Sui sistemi Windows e MacOS è installato un programma chiamato *BlueJ*. Eseguilo.

Sui sistemi Unix il programma di installazione ha installato uno script chiamato *bluej* nella cartella di installazione. Da un'interfaccia GUI, basta fare doppio click sul file. Dalla linea di comando, puoi lanciare BlueJ con o senza un progetto come argomento:

```
$ bluej
```

oppure

```
$ bluej examples/people
```

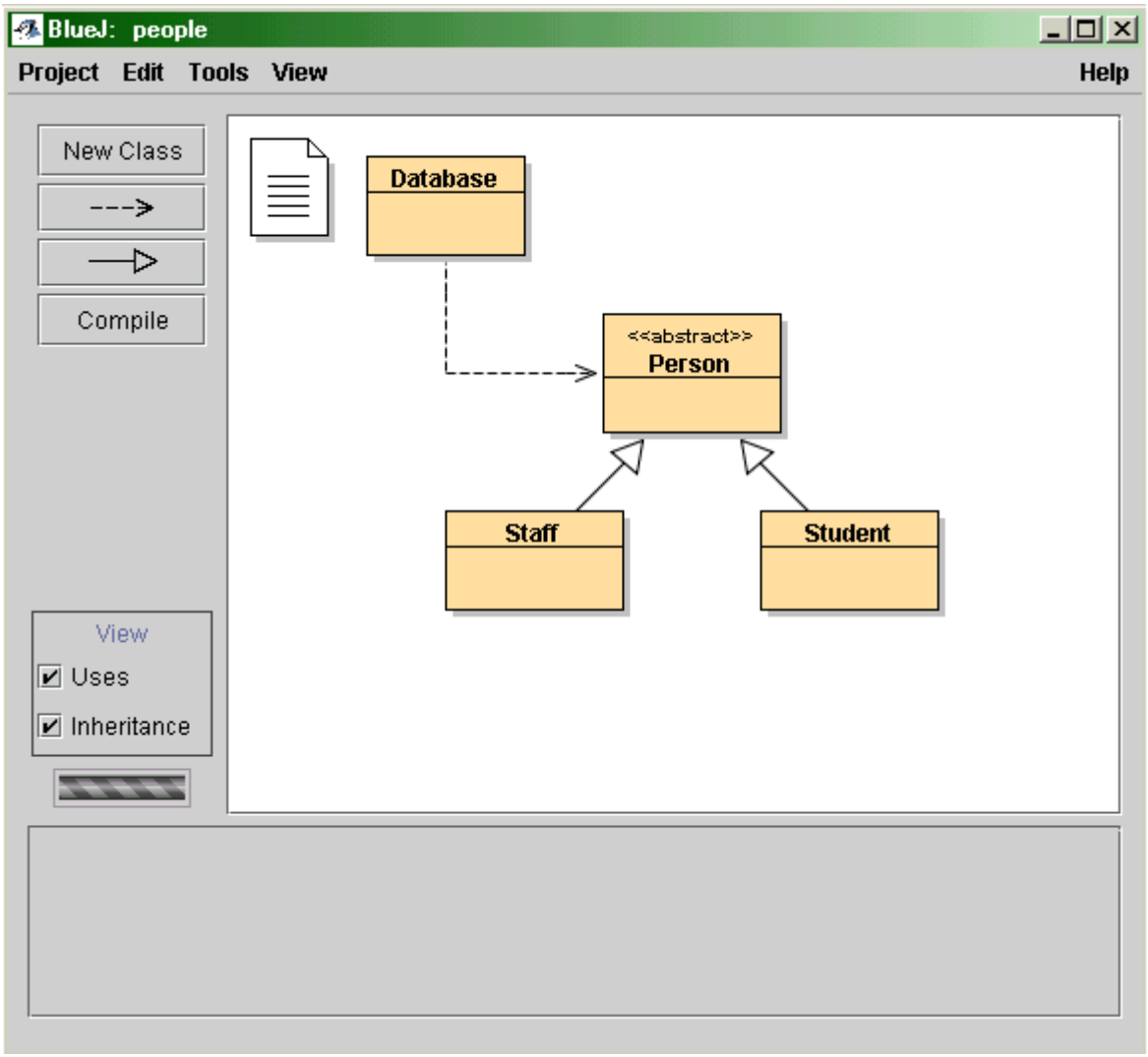

Figura 1: La finestra principale di BlueJ

3.2 Aprire un progetto

Sommario: per aprire un progetto, seleziona Open *dal menu* Project

I progetti BlueJ, assomigliano ai package standard di Java, sono cartelle contenenti i file inclusi nel progetto.

Dopo aver avviato BlueJ, usa il comando Project – Open…, dal menu, per selezionare e aprire un progetto.

Alcuni progetti d'esempio sono inclusi con la distribuzione standard di BlueJ nella cartella *examples*.

Per questa sezione del tutorial, apri il progetto *people*, che è incluso in questa cartella. Puoi trovarlo nella cartella *examples* inserita all'interno della cartella principale di BlueJ. Dopo aver aperto il progetto dovresti vedere qualcosa di simile alla finestra mostrata in Figura 1. La finestra potrebbe non essere esattamente uguale a quella del tuo sistema, ma ci potrebbero essere delle piccole differenze.

3.3 Creare oggetti

Sommario: Per creare un oggetto, seleziona un metodo costruttore dal menu popup della classe.

Una delle caratteristiche fondamentali di BlueJ è che, oltre ad eseguire un'applicazione completa, puoi anche interagire con i singoli oggetti di qualunque classe ed eseguire i loro metodi pubblici. Una esecuzione in BlueJ si svolge, di solito, creando un oggetto e poi invocando uno dei suoi metodi. Questo è molto utile durante la fase di sviluppo dell'applicazione – puoi testare le classi individualmente non appena vengono scritte. Non c'è bisogno di scrivere l'intera applicazione prima di fare questo.

> **Nota a margine** : *I metodi statici possono essere eseguiti direttamente, senza necessariamente creare un oggetto. Uno dei metodi statici potrebbe essere "main", così possiamo fare la stessa cosa che normalmente avviene in un'applicazione Java – iniziare un'applicazione attraverso l'esecuzione di un metodo statico principale ("main"). Torneremo a parlare di questo argomento più tardi. Prima, faremo altre cose, più interessanti che normalmente non possono essere fatte negli ambienti tradizionali di sviluppo di Java.*

I rettangoli che vedi al centro della parte principale della finestra (etichettate *Database*, *Person*, *Staff* e *Student*) sono delle icone che rappresentano le classi coinvolte in questa applicazione. Puoi far apparire un menu con tutte le operazioni applicabili ad una classe, cliccando sull'icona della classe con il pulsante destro del mouse (Macintosh: ctrl-click[1]) (Figura 2). Le operazioni mostrate sono le operazioni *new* associate a ciascuno dei metodi costruttori definiti per questa classe (posizionate in testa), seguite da alcune operazioni fornite dall'ambiente di sviluppo.

[1] Ogni volta che in questo tutorial, scriviamo right-click (click col tasto destro), gli utenti Macintosh dovrebbero leggerlo come ctrl-click

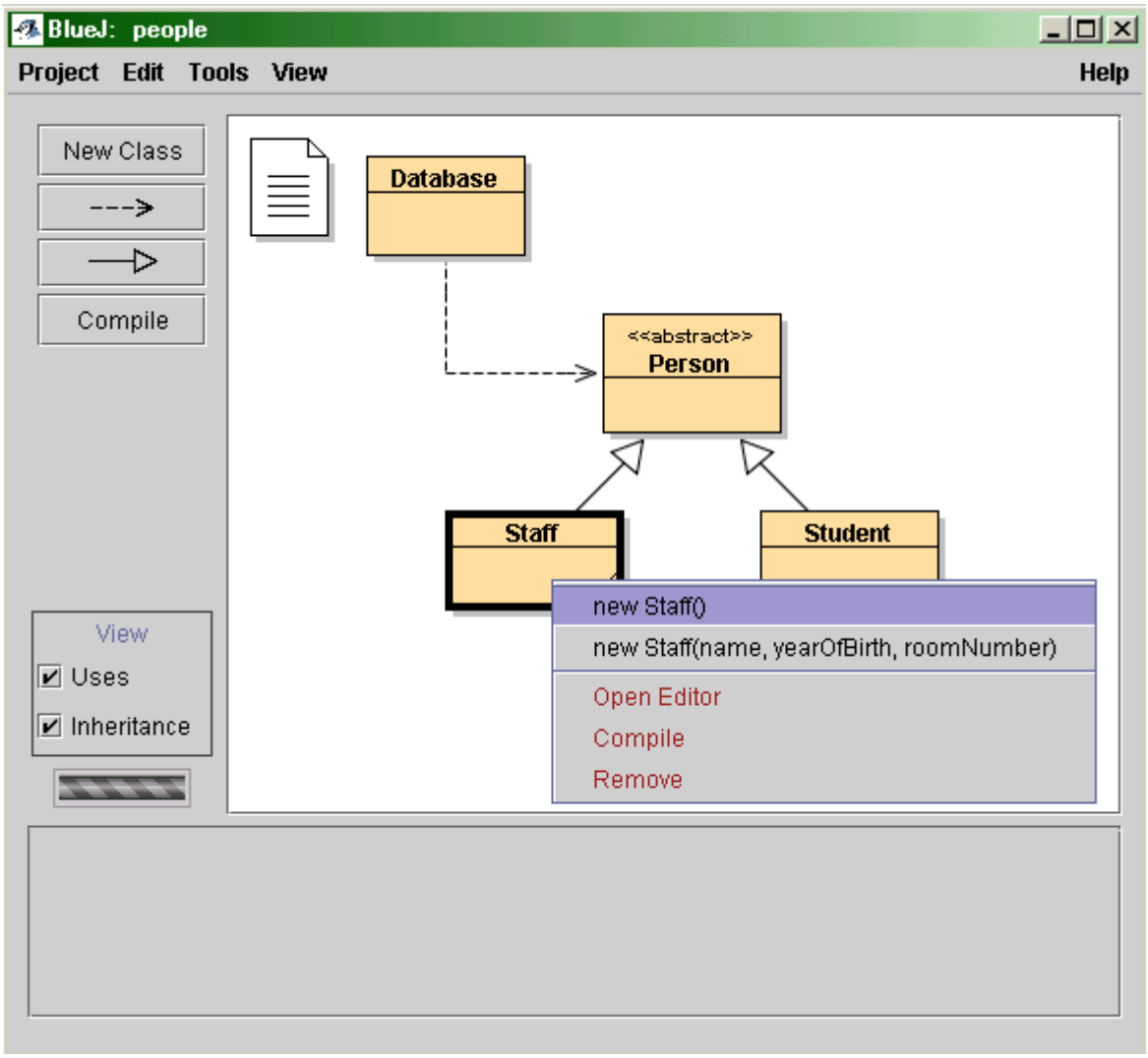

Figura 2: Operazioni sulla Classe (menu popup)

Per creare un oggetto di tipo *Staff*, dovresti fare click col pulsante destro sull'icona della classe *Staff* (che farà apparire il menu mostrato in Figura 2). Il menu mostra due metodi costruttori, uno con parametri e uno senza, per creare un oggetto *Staff*. Inizialmente, seleziona il metodo costruttore senza parametri. Apparirà la finestra di dialogo mostrata in Figura 3.

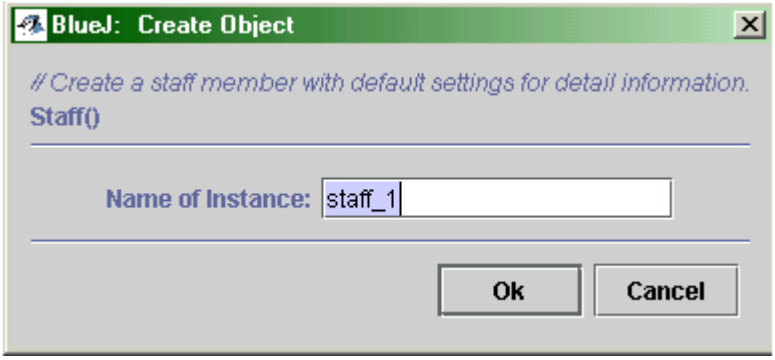

Figura 3: Creazione di un oggetto senza parametri

Questa finestra ti chiede un nome per l'oggetto che sta per essere creato. Allo stesso tempo, viene suggerito un nome di default (*staff_1*). Questo nome è abbastanza buono per ora, basta solo cliccare su *OK*. Verrà creato un oggetto di tipo *Staff*.

Una volta creato l'oggetto, esso viene posizionato nell'area degli oggetti (Figura 4). Questo è tutto quanto bisogna sapere per la creazione di un oggetto: seleziona un metodo costruttore dal menu della classe, eseguilo e avrai l'oggetto nell'area degli oggetti.

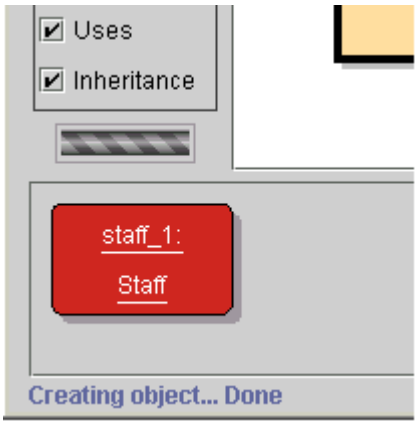

Figura 4: Un oggetto nell'area degli oggetti

Avrai notato che la classe *Person* è etichettata <<abstract>> (si tratta di una classe astratta). Ti accorgerai (se ci provi) che non puoi creare oggetti dalle classi astratte (come stabiliscono le specifiche del linguaggio Java).

3.4 Esecuzione

Sommario: Per eseguire un metodo, selezionalo dal menu popup dell'oggetto

Ora che hai creato un oggetto, puoi eseguire le sue operazioni pubbliche. Clicca con il pulsante destro del mouse sull'oggetto e verrà visualizzato un menu popup contenente le operazioni dell'oggetto (Figura 5). Il menu mostra i metodi utilizzabili da questo oggetto e due speciali operazioni previste dall'ambiente di sviluppo (*Inspect* e *Remove*). Ma queste saranno discusse più tardi. Prima, concentriamoci sui metodi.

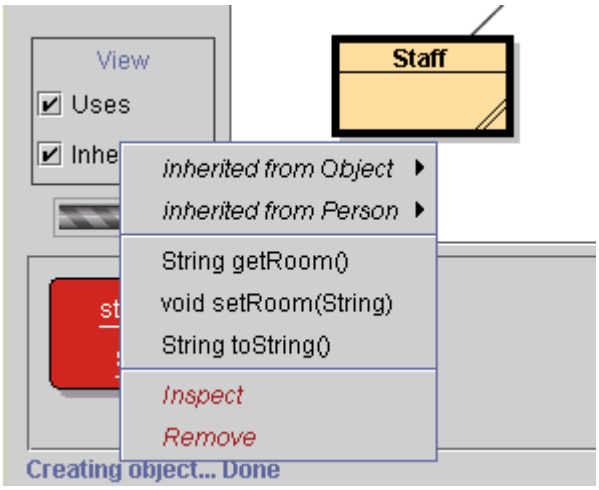

Figura 5: Il menu degli oggetti

Osserva la presenza dei metodi *setRoom* e *getRoom* che provvedono a modificare e ritornare i numeri della stanza per questi membri dello staff. Prova a chiamare *getRoom*. Selezionalo semplicemente dal menu dell'oggetto ed esso sarà eseguito. Appare una finestra di dialogo che visualizza i risultati della chiamata (Figura 6). In questo caso il risultato ottenuto è "(unknown room)" poiché non abbiamo assegnato alcuna stanza per questa persona.

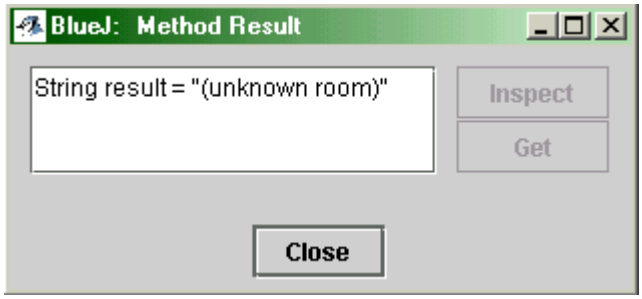

Figura 6: Visualizzazione del risultato di una funzione

I metodi ereditati dalla superclasse sono disponibili attraverso un sottomenu. All'inizio del menu popup dell'oggetto, ci sono due sottomenu, uno per i metodi ereditati dalla classe *Object* e uno per quelli ereditati della classe *Person* (Figura 5). Puoi chiamare i metodi della classe *Person* (come *getName*) selezionandoli dal sottomenu. Prova ad effettuare l'operazione. Avrai notato che la risposta è ugualmente vaga: risponde "(unknown name)", perché non abbiamo dato un nome alla nostra persona.

Ora proviamo a specificare un nome per la stanza. Questo operazione ti mostrerà come effettuare una chiamata con passaggio di parametri. (Le chiamate *getRoom* e *getName* hanno valori di ritorno, ma non hanno parametri). Chiama la funzione *setRoom* selezionandola dal menu. Apparirà una finestra di dialogo che ti chiederà di inserire i parametri (Figura 7).

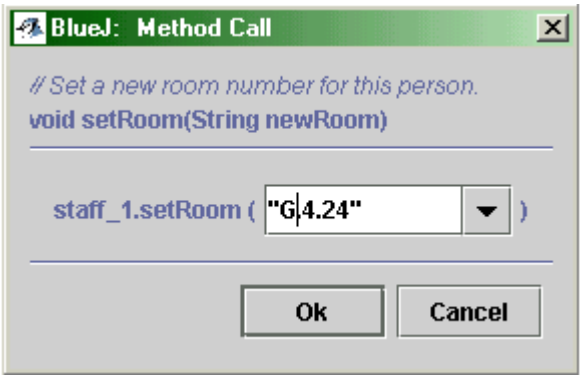

Figura 7: Finestra per la chiamata di una funzione con passaggio di parametri

Le prime righe in questa finestra, mostrano l'interfaccia del metodo che è stato chiamato (incluso il commento e la signature). Subito sotto c'è un campo per l'immissione del testo dove puoi inserire i parametri. La signature mostrata nella finestra di dialogo ci dice che bisogna inserire un parametro di tipo String. Inserisci un nuovo nome come stringa (scrivendo anche le virgolette) nel campo di testo e clicca *OK*.
Questo è tutto – dato che questo metodo non ritorna alcun valore, non verrà visualizzata la finestra di dialogo con il risultato.
Chiama nuovamente *getName* per controllare che il nome sia stato realmente cambiato.

Divertiti un po' creando oggetti e chiamando metodi. Prova a chiamare un costruttore con argomenti e chiama qualche altro metodo fino a quando non avrai familiarizzato con queste operazioni.

3.5 Modificare una classe

Sommario: Per modificare il sorgente di una classe, fare doppio click sull'icona della classe.

Finora, abbiamo lavorato solo con l'interfaccia di un oggetto. Ora è il momento di guardare cosa c'è all'interno. Puoi vedere l'implementazione di una classe selezionando *Edit Implementation* dalle operazioni della classe. (Promemoria: un click col tasto destro sull'icona della classe mostra le operazioni della classe). Fare doppio click sull'icona della classe è una scorciatoia della stessa funzione. L'editor non è descritto dettagliatamente in questo tutorial, ma dovrebbe essere molto semplice da usare. I dettagli sull'editor saranno descritti separatamente più tardi. Per ora, apri l'implementazione della classe *Staff*. Trova l'implementazione del metodo *getRoom*. Esso ritorna, come il nome suggerisce, il numero della stanza del componente dello staff. Cambiamo il metodo aggiungendo il prefisso "room" al risultato della funzione (cosicché il metodo ritornerà, "room G.4.24" anziché solo "G.4.24"). Possiamo fare ciò modificando la riga

```
return room;
```

in

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

BlueJ supporta pienamente le specifiche di Java, quindi non è richiesto nessun accorgimento particolare per implementare le tue classi.

3.6 Compilazione

Sommario: Per compilare una classe, clicca sul pulsante Compile *nell'editor. Per compilare un progetto, clicca sul pulsante* Compile *della finestra del progetto.*

Dopo aver inserito il testo (se non devi fare nient'altro), fai un controllo generale (la finestra principale). Noterai che l'icona della classe *Staff* è cambiata: ora è rappresentata a "strisce". Questa rappresentazione a strisce individua le classi che non sono state compilate dall'ultimo cambiamento. Torna all'editor.

> **Nota a margine**: *potresti essere meravigliato perché le icone delle classi non erano a strisce quando hai aperto inizialmente questo progetto. Questo perché le classi del progetto people erano state distribuite già compilate. Spesso i progetti BlueJ sono distribuiti senza essere compilati, perciò d'ora in avanti aspettati di vedere la maggior parte delle icone delle classi rappresentate a strisce quando apri per la prima volta un progetto.*

Nella barra degli strumenti in cima all'editor ci sono alcuni pulsanti relativi ad alcune funzioni utilizzate normalmente. Una di queste è *Compile*. Questa funzione ti permette di compilare la classe direttamente dall'interno dell'editor. Clicca ora sul pulsante *Compile*. Se non ci sono errori, appare un messaggio, nell'area informazioni in fondo all'editor, che ti informa che la classe è stata compilata. Se ci sono errori di sintassi (*syntax error*), la riga in cui si è verificato l'errore verrà evidenziata e un messaggio di errore sarà visualizzato nell'area informazioni. (Se la compilazione non produce errori, prova ad introdurre, in questo momento, un errore di sintassi – per esempio un punto e virgola omesso – e compila di nuovo, giusto per vedere cosa succede).

Dopo aver compilato la classe con successo, chiudi l'editor.

Nota a margine: *Non c'è bisogno di salvare esplicitamente il sorgente della classe. I sorgenti vengono salvati automaticamente non appena è possibile farlo (es. quando si chiude l'editor o dopo che la classe è stata compilata). Puoi salvare esplicitamente se preferisci (c'è una funzione nell'editor del menu Class), ma ne avrai bisogno solo quando il tuo sistema diverrà veramente instabile, si bloccherà frequentemente e ti preoccuperai per la perdita del lavoro.*

Nella barra degli strumenti della finestra del progetto, è presente il pulsante *Compile* per la compilazione. *Compile* viene utilizzato per compilare l'intero progetto. (Infatti, indica quali classi hanno bisogno di una ricompilazione e poi le ricompila nel giusto ordine). Prova a farlo, modificando due o più classi (cosicché due o più classi appaiono disegnate a strisce nel diagramma delle classi) e poi clicca sul pulsante *Compile*. Se il compilatore troverà un errore nel codice delle classi, si aprirà la finestra dell'editor, sarà evidenziata la posizione dell'errore e verrà visualizzato il messaggio di errore.

Puoi notare che l'area di visualizzazione degli oggetti è di nuovo vuota. Gli oggetti vengono cancellati ogni volta che viene modificata l'implementazione (il sorgente) della classe.

3.7 Messaggi di aiuto per gli errori di compilazione

Sommario: Per ricevere la spiegazione del messaggio d'errore prodotto dal compilatore, clicca sul punto interrogativo a lato del messaggio.

Molto spesso, all'inizio, gli studenti trovano difficoltà a capire i messaggi di errore del compilatore. Proviamo a fornire qualche aiuto.

Apri di nuovo l'editor, introduci un errore nel file sorgente, e compila. Sarà visualizzato un messaggio d'errore nell'area delle informazioni dell'editor. Sulla destra, alla fine dell'area delle informazioni, è presente un punto interrogativo che puoi cliccare per avere maggiori informazioni sul tipo di errore (Figura 8).

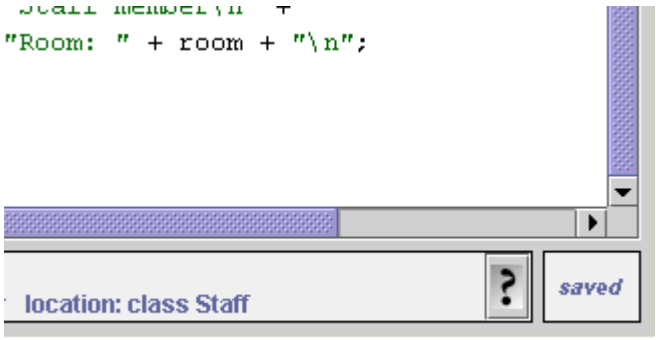

Figura 8: Un errore di compilazione ed il pulsante di *Help*

Attualmente, i testi di spiegazione non sono disponibili per tutti i messaggi di errore. Alcuni di essi sono tuttavia già stati scritti. E' facile verificare – che molti errori hanno già un loro testo di spiegazione. I restanti messaggi di spiegazione saranno scritti e inclusi nella futura versione di BlueJ.

4 Facendo un po' di più…

In questo paragrafo, analizzeremo altre cose che puoi fare nell'ambiente di sviluppo.
Cose che non sono fondamentali, ma che vengono usate abitualmente.

4.1 Ispezione

Sommario: l'ispezione dell'oggetto permette alcune semplici operazioni di debug mostrando lo stato interno dell'oggetto.

Quando esegui i metodi di un oggetto, avrai notato che si può anche utilizzare l'operazione *Inspect* sugli oggetti, in aggiunta ai metodi definiti (Figura 5).
Questa operazione permette di verificare lo stato delle variabili di istanza (dette anche "campi" o attributi) degli oggetti. Prova a creare un oggetto con alcuni valori passati dall'utente (es. un oggetto *Staff* con il costruttore che accetta parametri). Seleziona poi l'operazione *Inspect* dal menu dell'oggetto.
Viene visualizzata una finestra di dialogo che fa vedere i campi dell'oggetto, il loro tipo ed il loro valore.(Figura 9)

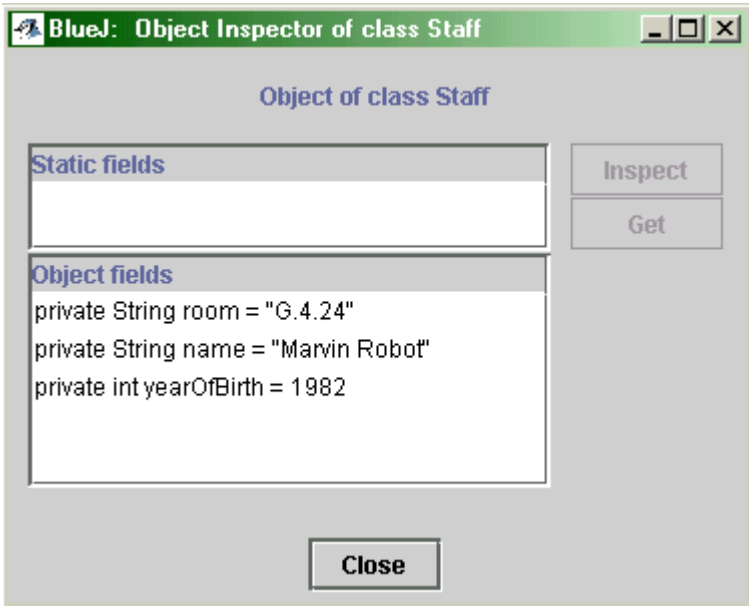

Figura 9 : Finestra di ispezione

L'ispezione è utile per verificare velocemente se un'operazione di modifica (un'operazione che cambia lo stato degli oggetti) è stata eseguita correttamente. Per questo motivo, l'ispezione è un semplice strumento per effettuare il debug.

Nell'esempio *Staff*, tutti i campi sono di tipo primitivo (cioè non di tipo oggetto oppure di tipo stringhe). Il valore di queste variabili, di tipi primitivo, può essere visto direttamente. Puoi vedere direttamente se il costruttore ha fatto le giuste assegnazioni.

In casi più complessi, i valori dei campi possono contenere reference ad oggetti definiti dagli utenti. Per fare un esempio useremo un altro progetto. Apri il progetto *people2* che è incluso nella distribuzione standard di BlueJ. Il diagramma delle classi del progetto *people2* è mostrato in Figura 10. Come puoi

vedere, questo secondo esempio ha una classe *Address* in aggiunta alle classi viste precedentemente. Uno dei campi nella classe *Person* è definito di tipo *Address*

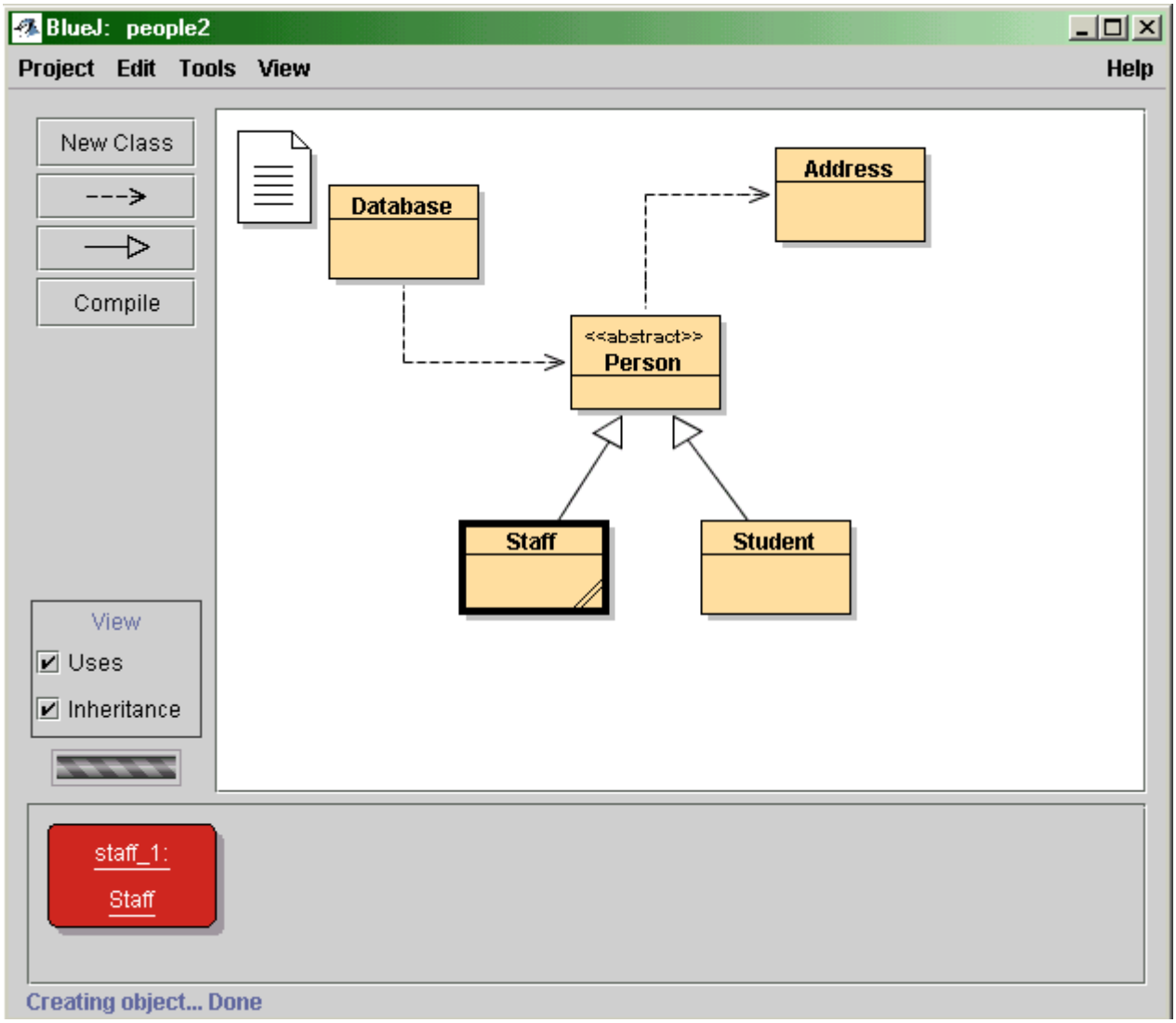

Figura 10: La finestra del progetto *people2*

Per la prossima funzionalità che vogliano esaminare – ispezionare i campi degli oggetti – crea un oggetto dalla classe *Staff* e poi chiama il metodo *setAddress* di questo oggetto (lo troverai nel sottomenu *Person*). Inserisci un indirizzo. Internamente, il codice della classe *Staff* crea un oggetto di classe *Address* e lo memorizza nel suo campo *address*.

Adesso, ispeziona l'oggetto *Staff*. La finestra di dialogo per l'ispezione è mostrata in Figura 11. Nei campi dell'oggetto *Staff* adesso viene mostrato anche *address*. Come puoi vedere, il suo valore è rappresentato come <*object reference*> - dal momento che è un oggetto complesso definito dall'utente, ed il suo valore non può essere mostrato direttamente in questo elenco. Inoltre, per esaminare l'indirizzo, seleziona il campo *address* nell'elenco e clicca il pulsante *Inspect* nella finestra di dialogo (puoi anche cliccare due volte sul campo *address*). Un'altra finestra di ispezione si apre mostrando i dettagli dell'oggetto *Address* (Figura 12).

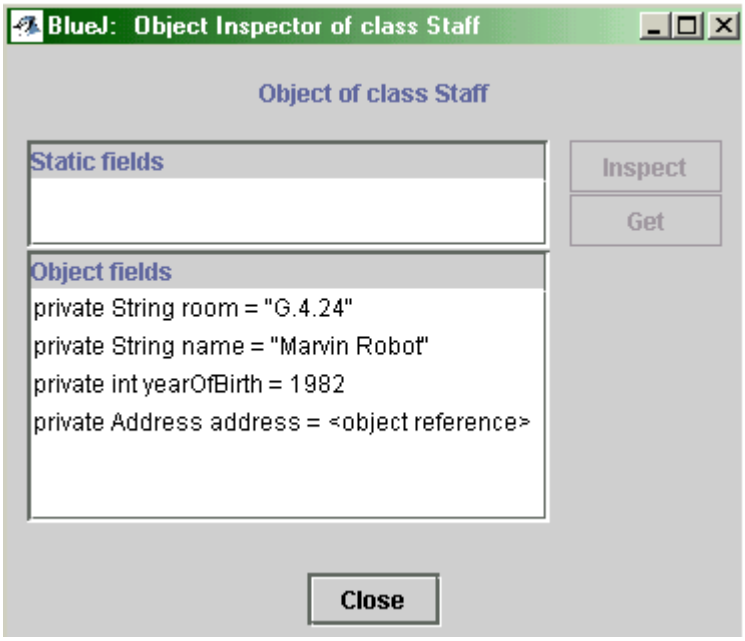

Figura 11: Finestra di ispezione con un reference ad un oggetto

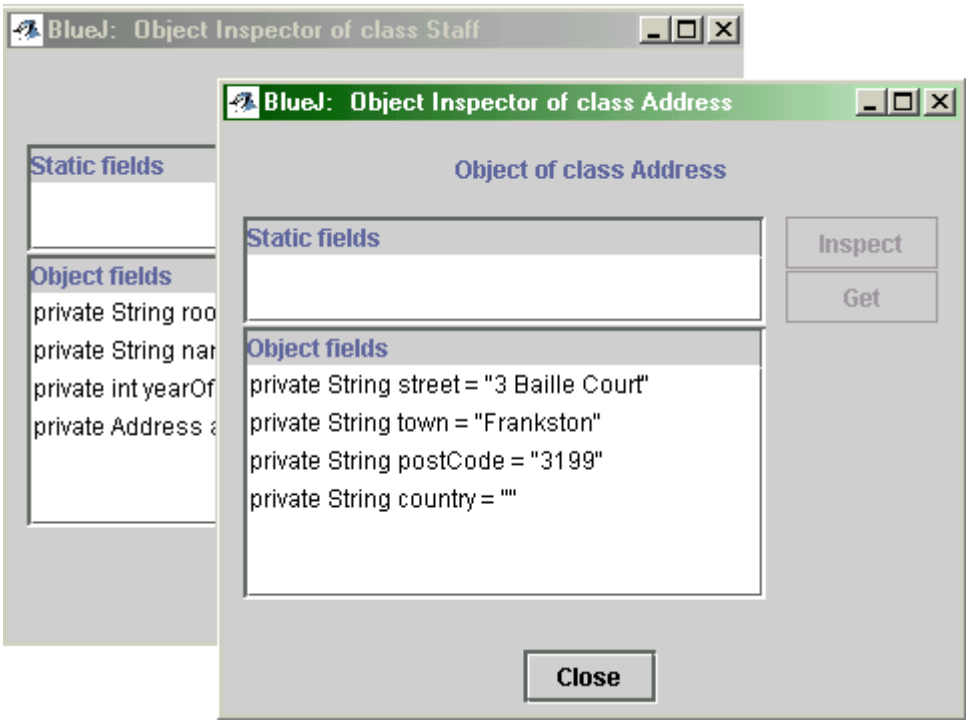

Figura 12: Finestra di ispezione dello stato interno dell'oggetto

Se il campo selezionato è pubblico, invece di cliccare *Inspect*, puoi anche selezionare il campo *address* e premere il pulsante *Get*. Questa operazione colloca l'oggetto selezionato nell'area degli oggetti. Qui puoi esaminarlo ulteriormente facendo delle chiamate ai suoi metodi.

4.2 Composizione

Sommario: un oggetto può essere passato come parametro, durante la chiamata di un metodo, cliccando sull'icona dell'oggetto.

Il termine "composizione" si riferisce all'abilità di passare gli oggetti come parametri ad altri oggetti. Adesso prova con un esempio. Crea un oggetto di classe *DataBase* (osserva che la classe *DataBase* ha solo un costruttore che non accetta parametri, cosi la costruzione di un oggetto è più semplice). L'oggetto *DataBase* ha la capacità di contenere una lista di persone. Possiede operazioni per aggiungere oggetti *Person* e mostrare tutte le persone attualmente memorizzate (chiamarlo *DataBase* è in effetti una esagerazione!).

Se non hai già un oggetto *Staff* o un oggetto *Student* nell'area degli oggetti, creane uno. Per il seguito hai bisogno di un oggetto *DataBase* e di un oggetto *Staff* oppure di un oggetto *Student* nell'area degli oggetti, contemporaneamente.

Adesso chiama il metodo *addPerson* dell'oggetto *DataBase*. La signature ti dice che è atteso un parametro di tipo *Person*. (ricorda: la classe *Person* è astratta, quindi non si possono creare oggetti direttamente da questa classe. Tuttavia per il subtyping, gli oggetti Student e Staff possono essere sostituiti con oggetti *Person*. Cosi è possibile passare un oggetto *Student* o *Staff* dove è atteso un oggetto *Person*). Per passare l'oggetto, che hai nell'area degli oggetti, come parametro durante la chiamata di un metodo, puoi inserire il suo nome nel campo del parametro, oppure come scorciatoia, basta cliccare sull'oggetto. Questo inserisce il nome dell'oggetto nella finestra di dialogo della chiamata del metodo. Clicca *OK* e la chiamata del metodo è effettuata. Poiché non c'è nessun valore di ritorno per questo metodo, non vedremo immediatamente il risultato. Puoi chiamare il metodo *listAll* dell'oggetto *DataBase* per verificare che l'operazione è stata realmente eseguita. L'operazione *listAll* scrive le informazione della persona nella finestra di output standard. Puoi osservare che una finestra testuale si apre automaticamente per visualizzare il testo.

Prova più volte queste operazioni con più persone inserite nel "database".

5 Creare un nuovo progetto

Questo capitolo ti mostra come creare rapidamente un nuovo progetto.

5.1 Creare la cartella del progetto

Sommario: per creare un progetto seleziona New… dal menu Project.

Per creare un nuovo progetto, seleziona Project - New … dalla barra del menu. Una finestra di dialogo si apre e ti permette di specificare un nome ed una cartella per il nuovo progetto. Prova a farlo adesso. Puoi scegliere qualsiasi nome per il tuo progetto. Dopo aver cliccato OK, verrà creata una cartella con il nome che hai specificato, e la finestra principale mostrerà il nuovo progetto vuoto.

5.2 Creare le classi

Sommario: Per creare una classe, clicca il pulsante New Class *e scrivi il nome della classe.*

Adesso puoi creare le tue classi cliccando il pulsante *New Class* sulla barra degli strumenti del progetto. Ti verrà chiesto di fornire un nome per la classe - questo nome deve essere un identificatore Java valido.

Puoi quindi scegliere tra quattro tipi di classi: astratta(abstract), interfaccia(interface), applet o “standard”. Questa scelta determina quale struttura iniziale (skeleton) in linguaggio Java verrà inizialmente creata per la tua classe. Puoi cambiare il tipo di classe successivamente, modificando il codice sorgente (per esempio, aggiungendo la parola chiave "abstract" nel codice).

Dopo aver creato una classe, essa viene rappresentata da un’icona nel diagramma. Se non è una classe standard, il tipo (interface, abstract o applet) è indicato nell'icona della classe. Quando apri l'editor su una nuova classe, puoi osservare che una struttura iniziale di default della classe è già stata creata - questo dovrebbe rendere più semplice implementare il codice per la classe. Il codice di default è sintatticamente corretto. Questo può essere compilato ed eseguito (ma non fa molto). Prova a creare qualche nuova classe e a compilarla.

5.3 Creare le dipendenze

Sommario: Per creare una freccia, clicca il pulsante della freccia e trascina la freccia nel diagramma, oppure scrivi solo il codice sorgente nell'editor.

Il diagramma delle classi mostra le dipendenze tra le classi sotto forma di frecce. Le relazioni di ereditarietà (“extends" o "implements”) sono mostrate con freccia a linea continua; le relazioni “uses” sono mostrate come freccia a linea tratteggiata.

Puoi aggiungere le dipendenze o graficamente (direttamente nel diagramma) o testualmente, nel codice sorgente. Se aggiungi una freccia graficamente, il sorgente è automaticamente aggiornato; se aggiungi la dipendenza nel sorgente il diagramma è aggiornato.

Per aggiungere una freccia graficamente, clicca il pulsante appropriato (freccia a linea continua per “extends” o “implements”, freccia a linea tratteggiata per "uses") e trascina la freccia da una classe all’altra.

Aggiungendo una freccia che rappresenta l'ereditarietà, nel codice sorgente della classe verrà inserita la definizione “extends" o "implements" (dipende se il destinatario è una classe o una interfaccia).

Aggiungendo una freccia “uses” il sorgente non si modifica immediatamente (a meno che il destinatario è una classe di un altro package. In questo caso viene generata una istruzione "import", ma ciò non lo abbiamo ancora visto nei nostri esempi). Se una freccia uses, nel diagramma, punta ad una classe che non è attualmente usata nel tuo sorgente, verrà generato un messaggio di avvertimento che ti comunica come una relazione "uses" ad una classe è stata impostata ma in realtà non viene mai usata.

Aggiungere le frecce testualmente è facile: basta solo aggiungere il codice come faresti normalmente. Non appena la classe viene salvata, il diagramma viene aggiornato (Ricorda: il salvataggio avviene automaticamente chiudendo l'editor).

5.4 Eliminare gli elementi

Sommario: Per rimuovere una classe, seleziona la funzione di rimozione dal suo menu popup. Per rimuovere una freccia, seleziona Remove *dal menu* Edit *e clicca sulla freccia.*

Per rimuovere una classe dal diagramma, seleziona la classe ed infine scegli *Remove Class* dal menu *Edit*. Puoi inoltre selezionare direttamente *Remove* dal menu popup della classe. Per rimuovere una freccia, seleziona *Remove Arrow* dal menu e quindi scegli la freccia che vuoi cancellare.

6 Debugging

Questa sezione introduce gli aspetti più importanti delle funzionalità di debugging in BlueJ. Parlando con i docenti di informatica, emerge spesso l'osservazione che è auspicabile usare il debugger nel primo anno d'insegnamento, ma purtroppo non c'è il tempo per introdurlo. Gli studenti familiarizzano con l'editor, il compilatore e l'esecuzione dei programmi; non c'è il tempo utile per introdurre un altro strumento complicato.

E' per questo che abbiamo deciso di rendere il debugger il più semplice possibile. L'obiettivo è di avere un debugger, che può essere spiegato in 15 minuti, e che gli studenti possono usare senza ulteriori chiarimenti. Vediamo un po' se ci siamo riusciti.

Prima di tutto, abbiamo ridotto le funzionalità, rispetto ai debugger tradizionali, a soli tre compiti:

- Impostare dei breakpoints
- Eseguire il codice istruzione dopo istruzione (step)
- Ispezionare le variabili

Ognuno dei tre compiti è molto semplice. Vogliamo adesso verificare ognuno di essi.
Per iniziare, apri il progetto *debugdemo*, che è incluso nella cartella degli esempi nella distribuzione. Questo progetto contiene poche classi con il solo scopo di mostrare la funzionalità del debugger – altrimenti non avrebbe molto senso.

6.1 Impostare i breakpoints

Sommario: Per impostare un breakpoint, clicca nell' area breakpoint, alla sinistra del testo nell'editor.

Impostare un breakpoint significa interrompere l'esecuzione del programma ad un certo punto del codice. Quando l'esecuzione è interrotta, puoi ispezionare lo stato dei tuoi oggetti. Spesso questo ti aiuta a capire cosa sta succedendo nel tuo codice.

Nell'editor, alla sinistra del testo vi è l'area dei breakpoints (Figura 13). Puoi impostare un breakpoint cliccandoci sopra. Un piccolo segnale di stop appare per indicare il breakpoint. Provalo adesso. Apri la classe *Demo*, trova il metodo *loop* e imposta il breakpoint in qualsiasi punto del ciclo *for*. Il segnale stop appare nel tuo editor.

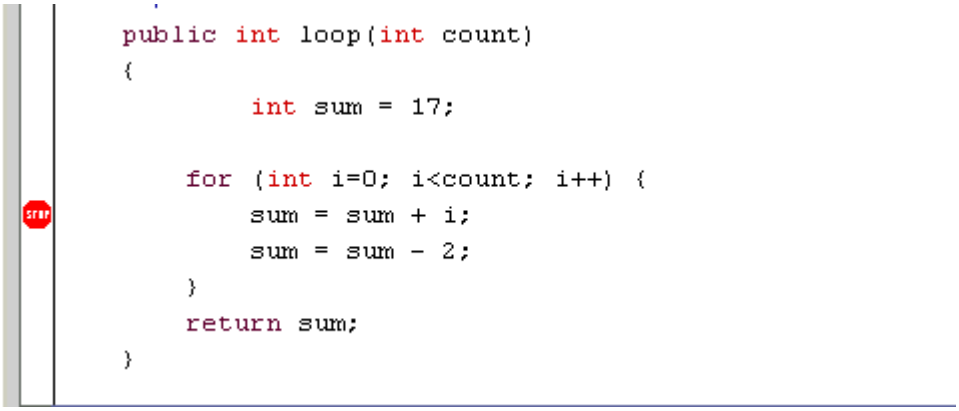

Figura 13: Un breakpoint

Quando la riga del codice, che ha il breakpoint impostato, viene raggiunta, l'esecuzione viene bloccata. Prova a sperimentarlo ora.

Crea un oggetto di classe *Demo* e chiama il metodo *loop* passando un parametro, ad esempio 10. Non appena il breakpoint è raggiunto, si apre la finestra dell'editor, mostrando la riga corrente del codice, insieme alla finestra del debugger. Viene visualizzato qualcosa di analogo alla Figura 14.

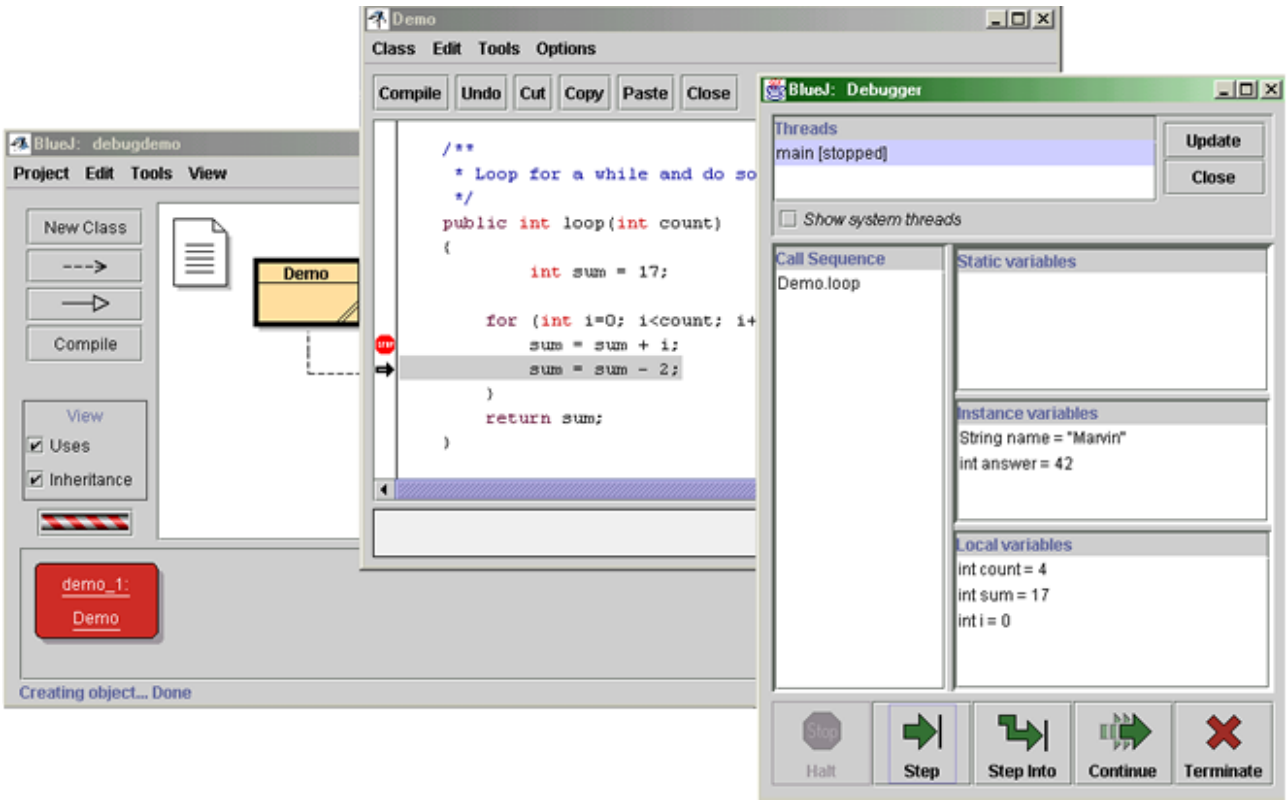

Figura 14 : La finestra del debugger

L'evidenziazione nell'editor mostra la riga che verrà eseguita successivamente. (L'esecuzione è fermata *prima* che la riga sia eseguita).

6.2 Eseguire il codice istruzione dopo istruzione

Sommario: Per eseguire il programma riga per riga, usa i tasti Step *e* Step Into *nel debugger*.

Adesso che abbiamo fermato l'esecuzione (questo ci convince del fatto che il metodo viene realmente eseguito e questo punto del codice viene certamente raggiunto), possiamo eseguire una istruzione per volta e vedere come si evolve il processo di esecuzione. Per fare questo clicca ripetutamente sul pulsante *Step* nella finestra del debugger. Dovresti veder cambiare la riga del codice sorgente nell'editor (l'evidenziatore si sposta sulla riga che deve essere eseguita). Ogni volta che clicchi il pulsante *Step*, ogni singola riga del codice viene eseguita e l'esecuzione viene fermata di nuovo. Nota inoltre che i valori delle variabili, mostrati nelle finestra del debugger, cambiano (per esempio il valore di *sum*). Cosi puoi eseguire passo dopo passo (istruzione dopo istruzione) e osservare cosa succede. Una volta che hai

terminato la verifica, puoi cliccare ancora sul breakpoint per rimuoverlo, e poi clicca il pulsante *Continue* nella finestra del debugger per riavviare l'esecuzione e continuare normalmente.

Riprova ancora con un altro metodo. Imposta un breakpoint nella classe *Demo*, nel metodo *carTest()*, nella riga

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

Chiama il metodo. Quando il breakpoint viene raggiunto, stai appunto per eseguire una riga che contiene una chiamata al metodo *seats()* nella classe *Car*. Cliccando *Step* eseguirai totalmente la riga dell'istruzione. Questa volta, invece, prova *Step Into*. Se entri (S*tep Into*) nel metodo chiamato, puoi accedere al codice sorgente ed eseguirlo riga per riga (non come una singola istruzione). In questo caso stai entrando nel metodo *seats()* della classe *Car*. Puoi ora facilmente eseguire le istruzioni di questo metodo fino a raggiungere la fine e ritornare al metodo chiamante. Osserva come il debugger visualizza continuamente i cambiamenti.

Step e *Step Into* si comportano allo stesso modo se la riga di codice non contiene una chiamata ad un metodo.

6.3 Ispezionare le variabili

Sommario: Ispezionare le variabili è facile - sono automaticamente visualizzate nella finestra del debugger.

Quando effettui il debug sul tuo codice, è importante saper ispezionare lo stato degli oggetti (variabili locali e variabili di istanza).
Fare questo è banale - molte di queste cose le abbiamo già viste. Non hai bisogno di comandi speciali per ispezionare le variabili; le variabili d'istanza (attributi) dell'oggetto corrente e le variabili locali del metodo corrente sono sempre automaticamente visualizzate e aggiornate.
Puoi selezionare i metodi nella sequenza delle chiamate per esaminare le variabili di altri oggetti e metodi ancora attivi. Prova, per esempio, ad inserire nuovamente un breakpoint nel metodo *carTest()*. Nella parte sinistra della finestra del debugger, osserverai la sequenza delle chiamate dei metodi. Al momento visualizza

```
Car.seats

Demo.carTest
```

Ciò indica che `Car.seats` è stata chiamata da `Demo.carTest`. Puoi selezionare `Demo.carTest` in questa lista per ispezionare il codice sorgente e i valori delle variabili correnti in questo metodo.

Se vai avanti, dopo la riga che contiene l'istruzione `new Car(…)`, puoi osservare che il valore della variabile locale `myCar` è visualizzata come <*object reference*>. Tutti i valori di tipo oggetto (eccetto per le String) sono mostrati in questo modo. Puoi ispezionare questa variabile cliccando doppiamente su di essa. Così facendo aprirai una finestra di ispezione dell'oggetto uguale a quella descritta in precedenza (Paragrafo 4.1). Non c'è nessuna reale differenza tra ispezionare gli oggetti qui e ispezionare gli oggetti nell'area degli oggetti.

6.4 Interrompere e terminare l'esecuzione

Sommario : Halt *e* Terminate *possono essere usati per fermare l'esecuzione temporaneamente o permanentemente.*

A volte un programma rimane in esecuzione per molto tempo, e ti chiedi se ogni cosa è a posto. Probabilmente c'è un ciclo infinito che rende l'esecuzione troppo lunga. Bene, possiamo controllare. Chiama il metodo *longLoop()* della classe *Demo*. Questo verrà eseguito per un pò.

Adesso vogliamo sapere cosa succede. Apri la finestra del debugger, se non è già sullo schermo (a proposito, una scorciatoia per aprire la finestra del debugger è quella di cliccare il simbolo rotante, che segnala la fase di esecuzione della macchina virtuale di Java durante l'elaborazione).

Adesso clicca il pulsante *Halt*. L'esecuzione è interrotta come se avessimo trovato un breakpoint.
Adesso esegui un paio di operazioni, osserva le variabili, e vedi se tutto è a posto - per questo hai bisogno di un po' più di tempo per terminare. Ti basta usare i pulsanti *Continue e Halt* molte volte per vedere la velocità con cui si incrementa il contatore. Se non vuoi proseguire (per esempio, hai scoperto che sei in un ciclo infinito) è sufficiente premere il pulsante *Terminate* per interrompere l'esecuzione.
Il pulsante *Terminate* deve essere usato con moderazione – poiché alcuni oggetti funzionanti, possono assumere uno stato inconsistente interrompendo così l'esecuzione della macchina virtuale di Java (JVM) - per cui si raccomanda di usarlo solo in caso di emergenza.

7 Creare applicazioni autonome

Sommario: Per creare un' applicazione autonoma, usa Project – Export…

BlueJ è in grado di creare file jar eseguibili. Questi file possono essere eseguiti su alcuni sistemi operativi tramite un doppio click sul file stesso (per esempio sotto Windows e MacOS X), o utilizzando il comando `java –jar <nome del file>.jar` (Unix o Dos).

Possiamo provare tutto questo con il progetto d'esempio *hello.* Aprilo (si trova nella cartella *examples*). Assicurati che il progetto sia compilato. Seleziona la funzione *Export…* dal menu *Project*.

La finestra di dialogo che si aprirà ti permetterà di specificare il tipo di formato di memorizzazione (Figura 15). Scegli "jar file" per creare un file jar eseguibile. Per rendere un file jar eseguibile, bisogna specificare anche una classe principale. Questa deve avere un metodo *main* definito correttamente (con la signature `public static void main (String[] args)`).

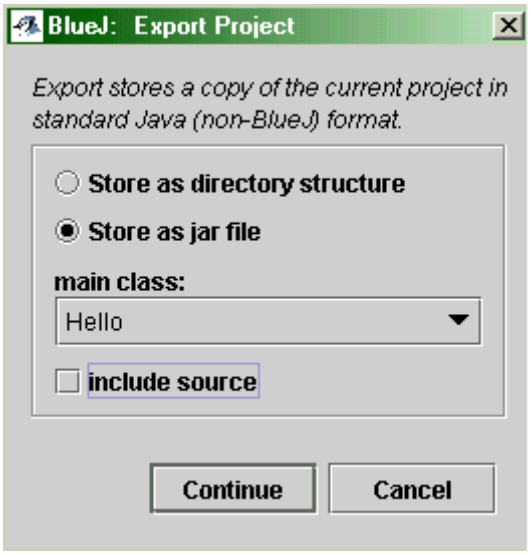

Figura 15: La finestra di dialogo "Export"

Nel nostro esempio, è facile scegliere la classe principale: c'è né solo una. Seleziona *Hello* dal menu popup. Se hai altri progetti, seleziona la classe che contiene il metodo "main" che vuoi eseguire.

Di solito, non c'è la necessità di includere il codice sorgente nei file eseguibili. Ma tu puoi, se vuoi, distribuire anche il tuo codice sorgente.

Clicca su *Continue*. Apparirà una finestra di dialogo per la scelta del file, in cui specificare il nome del file jar da creare. Scrivi *Hello* e premi OK. La creazione del file jar eseguibile è completata.

Puoi fare un doppio click sul file jar solo se l'applicazione usa un'interfaccia grafica(GUI). I nostri esempi utilizzano operazioni di I/O da ambiente testo, così possiamo solo eseguirlo da una finestra in emulazione terminale di testo. Proviamo ad eseguire adesso il file jar.

Apri una finestra in emulazione terminale o DOS. Vai quindi nella cartella in cui hai salvato il tuo file jar (puoi vedere un file *hello.jar*). Supponendo che Java sia installato correttamente sul tuo computer, dovresti poter scrivere

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

per eseguire il file.

8 Creare le applet

8.1 Eseguire un'applet

Sommario: Per eseguire un'applet, scegli Run applet *dal menu popup dell'applet.*

BlueJ permette di creare ed eseguire applet così come le applicazioni. Noi abbiamo incluso alcune applet nella cartella examples della distribuzione. Prima proveremo ad eseguirne una. Apri il progetto *appletClock* nella cartella examples.

Osserva che questo progetto è composto da una sola classe chiamata *Clock*. L'icona della classe è etichettata (con le lettere WWW) come un'applet. Seleziona il comando *Run Applet* dal menu popup della classe.

Sulla finestra di dialogo che viene visualizzata, puoi fare alcune scelte (Figura 16).

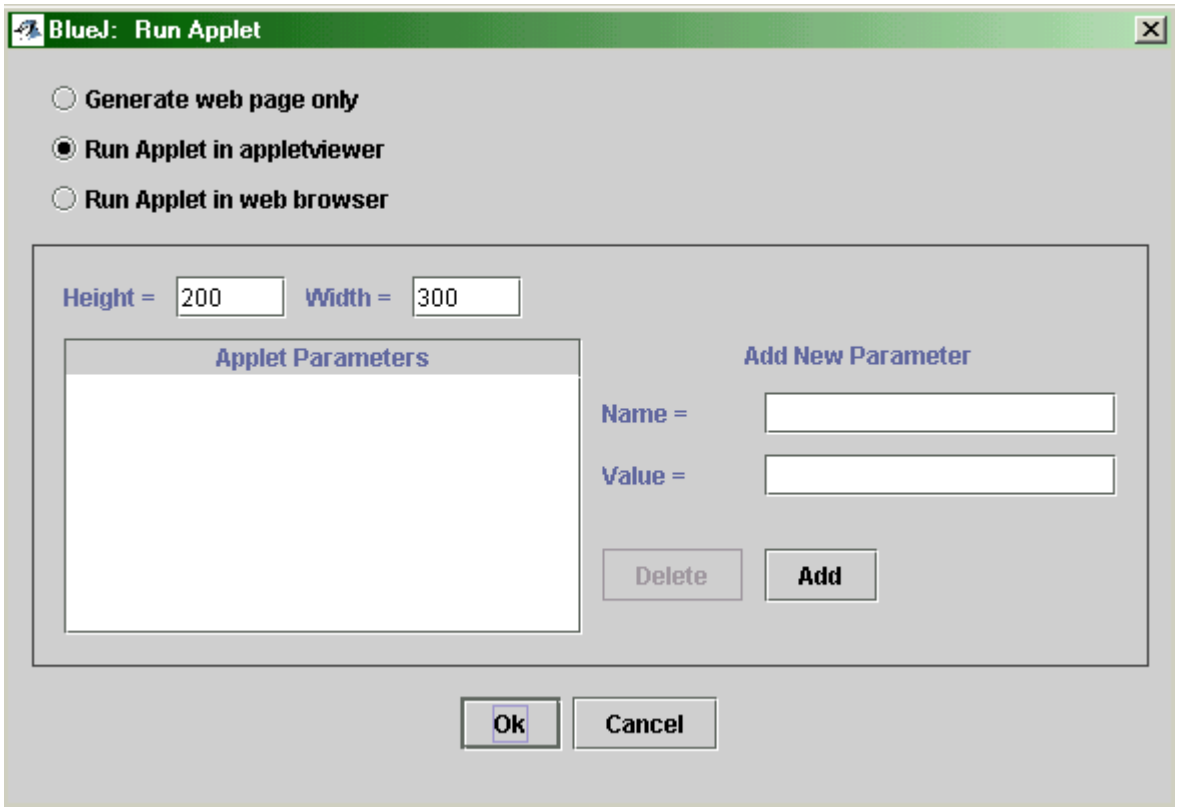

Figura 16: La finestra di dialogo "Run Applet"

Osserva che puoi scegliere se eseguire l'applet in un browser o in un visualizzatore delle applet (o generare subito la pagina web senza lanciare l'applet). Lascia le scelte di default e premi OK. Dopo pochi secondi, un visualizzatore di applet dovrebbe mostrare l'applet orologio.

Il visualizzatore delle applet è installato insieme al JDK, in questo caso è sempre garantito che esso abbia la stessa versione del compilatore Java. Generalmente provoca meno problemi rispetto ai browser. Il tuo browser può eseguire differenti versioni di Java, e considerando la versione utilizzata dal tuo browser, ci possono essere dei problemi. Però con le attuali versioni dei browser dovrebbe funzionare tutto bene.

Nei sistemi operativi Microsoft Windows e MacOS, BlueJ usa il browser di default. Sui sistemi Unix, il browser è definito dalle impostazioni di BlueJ.

8.2 Creare un'applet

Sommario: Per creare un'applet, premi il pulsante New Class *e seleziona* Applet *come tipo della classe.*

Dopo aver visto come eseguire un'applet, cercheremo di crearne una.

Crea una nuova classe di tipo *Applet* (puoi selezionare il tipo nella finestra *New Class*). Compila, poi lancia l'applet. Tutto qui. Non è difficile, vero?

Le applet (come le altre classi) sono generate con una struttura di default, contenente codice valido. Questo codice visualizza una semplice applet con due righe di testo. Ora puoi aprire l'editor e modificare l'applet per inserire il tuo codice.

Potrai vedere che tutti i metodi più comuni delle applet sono presenti, ognuno con un commento che ne spiega lo scopo. Il codice d'esempio è presente solamente nel metodo *paint.*

8.3 Testare un'applet

In alcune situazioni è utile creare un oggetto applet nell'area degli oggetti (come per le normali classi). Puoi fare questo – il metodo costruttore è presente nel menu popup dell'applet. Dall'area degli oggetti non puoi eseguire l'applet in maniera completa, ma puoi richiamare alcuni suoi metodi. Questo può essere utile per testare i singoli metodi che hai implementato per l'applet.

9 Altre operazioni

9.1 Aprire dei package non-BlueJ in BlueJ

Sommario: I package non-BlueJ possono essere aperti con il comando Project: Open Non BlueJ…

BlueJ ti permette di aprire dei package esistenti, creati fuori da BlueJ. Per fare questo seleziona Project - Open Non BlueJ… dal menu. Seleziona la cartella che contiene i file Java sorgenti, dopo premi il pulsante Open in BlueJ. Il sistema chiederà conferma per aprire la cartella.

9.2 Aggiungere classi esistenti al tuo progetto

Sommario: Classi esterne possono essere copiate in un progetto usando il comando Add Class from File…

Spesso vuoi usare nel tuo progetto in BlueJ, una classe che si trova in qualche altro posto. Per esempio, il professore fornisce agli studenti una classe Java, memorizzata su un disco, che deve essere usata in un progetto. Puoi facilmente incorporare una classe esistente nel tuo progetto selezionando Edit – Add Class from File… dal menu. La finestra che appare, ti lascerà selezionare un file Java sorgente (con estensione .java) da importare.

Quando la classe è importata nel progetto, una copia viene memorizzata nella cartella corrente del progetto. Il risultato è esattamente lo stesso a quello che avresti ottenuto se avessi creato la classe e scritto tutto il suo codice sorgente.

Un'alternativa è quella di aggiungere il file sorgente della nuova classe nella cartella del progetto trovandoti fuori da BlueJ. La prossima volta che aprirai il progetto, la classe sarà inclusa nel diagramma delle classi del progetto.

9.3 Chiamare il metodo main ed altri metodi statici

Sommario: Metodi statici possono essere chiamati dal menu popup delle classi.

Apri il progetto *hello* dalla cartella *examples*. L'unica classe del progetto (classe *Hello*) possiede un metodo *main* standard.

Fai un click-destro sulla classe, e vedrai che il menu della classe comprende oltre al metodo costruttore, anche il metodo statico *main*. Ora puoi chiamare il *main* direttamente da questo menu (senza creare prima un oggetto, come ci aspettiamo da un metodo statico).

Tutti i metodi statici possono essere chiamati in questo modo. Il metodo principale standard (main) accetta come parametro un array di String. Puoi passargli un array di String utilizzando la sintassi standard di Java per gli array costanti. Per esempio, puoi passare

```
{“one”,   “two”,  “three”}
```

(parentesi graffe comprese) al metodo. Prova!

Nota a margine: In Java standard, gli array costanti non possono essere usati come parametri attuali nelle chiamate dei metodi. Possono essere usati solo nelle inizializzazioni. In BlueJ, per abilitare la chiamata al metodo principale standard (main), permettiamo il passaggio di array costanti come parametri.

9.4 Generare la documentazione

Sommario: Per generare la documentazione di un progetto, seleziona Project Documentation *dal menu* Tools.

Puoi generare la documentazione per il tuo progetto, nel formato standard *javadoc*, dall'interno di BlueJ. Per fare questo, seleziona Tools - Project Documentation dal menu. Questa funzione genererà la documentazione per tutte le classi del progetto dal codice sorgente delle classi e aprirà un browser web per visualizzarla.

9.5 Lavorare con le librerie

Sommario: Le classi API standard di Java possono essere visionate selezionando Help – Java Standard Libraries.

Frequentemente, quando scrivi un programma Java, devi poter fare riferimento alla libreria standard di Java. Puoi aprire un browser web e visualizzare la documentazione API JDK selezionando Help – Java Standard Classes dal menu (se tu sei online).

La documentazione JDK può anche essere installata e usata in modalità locale (offline). I dettagli sono spiegati nel manuale di riferimento di BlueJ.

9.6 Creare oggetti dalla libreria delle classi.

Sommario: Per creare oggetti dalla libreria delle classi, usa Tools - Use Library Class.

BlueJ offre inoltre una funzione per creare oggetti dalle classi che non sono parti del tuo progetto, ma definite nella libreria. Puoi, per esempio, creare oggetti di classe String o ArrayList. Questo può essere molto utile per veloci esperienze con questi oggetti della libreria.
Puoi creare un oggetto dalla libreria, selezionando Tools - Use Library Class… dal menu. Una finestra apparirà e ti consentirà di inserire il nome completo della classe, come *java.lang.String*. (Osserva che devi scrivere il nome completo, cioè il nome comprendente i nomi dei package che contengono la classe.)
Il campo testo ha associato un menu popup che ti mostra le classi usate di recente. Una volta che il nome di una classe è stato inserito, premendo *Enter* verranno visualizzati tutti i costruttori e i metodi statici di

quella classe, in una lista all'interno della finestra. Ognuno di questi costruttori o metodi statici può ora essere eseguito selezionandolo dalla lista.

L'esecuzione procederà come se fosse una qualsiasi chiamata di costruttori o chiamata di metodi.

10 Solo i sommari

Per iniziare

1. Per aprire un progetto, seleziona *Open* dal menu *Project*
2. Per creare un oggetto, seleziona un metodo costruttore dal menu popup della classe.
3. Per eseguire un metodo, selezionalo dal menu popup dell'oggetto
4. Per modificare il sorgente di una classe, fare doppio click sull'icona della classe.
5. Per compilare una classe, clicca sul pulsante *Compile* nell'editor. Per compilare un progetto, clicca sul pulsante *Compile* della finestra del progetto.
6. Per ricevere la spiegazione del messaggio d'errore prodotto dal compilatore, clicca sul punto interrogativo a lato del messaggio.

Facendo un po' di più…

7. L'ispezione dell'oggetto permette alcune semplici operazioni di debug, mostrando lo stato interno dell'oggetto
8. Un oggetto può essere passato come parametro, durante la chiamata di un metodo, cliccando sull'icona dell'oggetto.

Creare un nuovo progetto

9. Per creare un progetto seleziona *New…* dal menu *Project.*
10. Per creare una classe, clicca il pulsante *New Class* e scrivi il nome della classe.
11. Per creare una freccia, clicca il pulsante della freccia e trascina la freccia nel diagramma, oppure scrivi solo il codice sorgente nell'editor.
12. Per rimuovere una classe, seleziona la funzione *Remove* dal suo menu popup.
13. Per rimuovere una freccia, seleziona *Remove* dal menu *Edit* e clicca sulla freccia.

Debugging

14. Per impostare un breakpoint, clicca nell'area breakpoint a sinistra del testo nell'editor.
15. Per eseguire il programma riga per riga, usa i tasti *Step* e *Step Into* nel debugger.
16. Ispezionare le variabili è facile - sono automaticamente visualizzate nella finestra del debugger.
17. *Halt* e *Terminate* possono essere usati per fermare l'esecuzione temporaneamente o permanentemente.

Creare applicazioni autonome

18. Per creare un'applicazione autonoma, usa *Project – Export…*

Creare delle applets

19. Per eseguire un'applet, scegli *Run applet* dal menu popup dell'applet.
20. Per creare un'applet, premi il pulsante *New Class* e seleziona *Applet* come tipo della classe.

Altre operazioni

21. I Package non-BlueJ possono essere aperti con il comando *Project: Open Non BlueJ…*
22. Classi esterne possono essere copiate in un progetto usando il comando *Add Class from File…*
23. Metodi statici possono essere chiamati dal menu popup delle classi.
24. Per generare la documentazione di un progetto, seleziona *Project Documentation* dal menu *Tools*.
25. Le classi API standard di Java possono essere visionate selezionando *Help – Java Standard Libraries*.
26. Per creare oggetti dalla libreria delle classi, usa *Tools - Use Library Class.*