Das BlueJ Tutorial

Version 2.0.1

für BlueJ Version 2.0.x

Michael Kölling
Mærsk Institute
University of Southern Denmark
Deutsche Übersetzung: Matthias Langlotz und Ingo Rhöse

Inhalt

1 Vorwort

1.1 Über BlueJ

Dieses Tutorial ist eine Anleitung zur Benutzung der BlueJ Programmieroberfläche. BlueJ ist eine Java™ Entwicklungsumgebung, die speziell für den Schulunterricht entworfen wurde. Es wurde vom BlueJ Team an der Deakin Universität in Melbourne, Australien, und der Universität von Kent in Canterbury, Großbritannien, entworfen und umgesetzt.

Mehr Informationen über BlueJ sind unter http://www.bluej.org verfügbar.

1.2 Anwendungsbereich und Zielgruppe

Dieses Tutorial ist für Personen gedacht, die sich selbst mit den Fähigkeiten dieser Umgebung vertraut machen wollen. Auf Gestaltungsentscheidungen, die der Konstruktion der Umgebung zu Grunde liegen oder die Hintergründe der Entwicklung wird nicht eingegangen.

Es ist nicht Ziel dieses Tutorials, Java zu lehren. Anfängern in der Javaprogrammierung wird empfohlen, ein Java-Lehrbuch für Anfänger zu lesen oder einen Javakurs besuchen.

Dieses Tutorial ist kein komplettes Handbuch für BlueJ. Viele Details fehlen – der Schwerpunkt liegt auf einer kurzen und präzisen Einleitung an Stelle einer kompletten Aufzählung der Funktionen. Für detailliertere Hinweise schlagen sie in *The BlueJ Environment Reference Manual* nach, das über die BlueJ Homepage (www.bluej.org) erhältlich ist.

Jeder Abschnitt beginnt mit einem Überblick in einem Satz. Dies erlaubt denjenigen Benutzern, die mit diesem Teil des Programms bereits vertraut sind zu entscheiden, ob sie diesen Abschnitt lesen oder überspringen möchten. Kapitel 11 wiederholt alle Überblick-Sätze und dient damit dem schnellen Nachschlagen.

1.3 Copyright, Lizenzierung und Weiterverkauf

Das BlueJ Programm und dieses Tutorial sind “so wie sie sind” erhältlich, kostenlos für den privaten Gebrauch und nicht für den kommerziellen Weiterverkauf. Das Zerlegen des Systems in Einzelbestandteile ist nicht gestattet.

Kein Teil des BlueJ Programms oder seine Unterlagen dürfen für kommerziell vertrieben oder in ein Paket aufgenommen werden, dass ohne schriftliche Erlaubnis der Autoren verkauft wird.

Das Copyright © für BlueJ liegt M. Kölling und J. Rosenberg.

1.4 Feedback

Kommentare, Fragen, Korrekturen, Kritik und jede andere Art von Feedback hinsichtlich des BlueJ Programms oder dieses Tutorials sind ausdrücklich erwünscht. Bitte senden Sie diese an Michael Kölling (mik@mip.sdu.dk).

2 Installation

BlueJ wird in drei verschiedenen Formate vertrieben: für Windows-, MacOS- und andere Systeme. Es zu unkompliziert installieren.

Vorraussetzung

Sie müssen J2SE V1.4 (auch als JDK 1.4 bekannt) oder höher auf ihrem System installiert haben, um BlueJ zu nutzen. Im Allgemeinen wird die Aktualisierung auf die neueste stabile (nicht-beta) Java Version empfohlen. Wenn Sie kein JDK installiert haben, können Sie es auf der Homepage von Sun http://java.sun.com/j2se/ herunterladen. Auf MacOS X ist die neueste J2SE Version vorinstalliert – Sie brauchen diese nicht selbst zu aufzuspielen. Wenn Sie eine Downloadseite finden, auf der sowohl ein "JRE" (Java Runtime Environment) als auch ein "SDK" (Software Development Kit) angeboten werden, sollten Sie das "SDK" laden da das JRE für unsere Zwecke nicht ausreicht.

2.1 Installation unter Windows

Die Installationsdatei für Windows-Systeme heißt *bluejsetup-xxx.exe*, wobei *xxx* die Versionsnummer ist. Beispielsweise heißt die Installationsdatei von BlueJ Version 2.0.0 *bluejsetup-200.exe*. Sie können diese Datei auf einer CD erhalten oder Sie können Sie auf der BlueJ Homepage http://www.bluej.org herunterladen. Führen Sie dann die Installationsdatei aus.

Im Installationsprogramm können Sie ein Laufwerk zum Installieren auswählen. Als Option können Sie eine Verknüpfung im Startmenü und auf dem Desktop anlegen. Nachdem die Installation beendet ist, finden Sie das Programm bluej.exe im Installationsverzeichnis von BlueJ.

Wenn Sie BlueJ das erste Mal starten, wird es nach einem Javasystem (JDK) suchen. Findet es mehr als ein mögliches Javasystem (z.B. falls Sie JDK 1.4.2 und JDK 1.5.0 installiert haben), können Sie in einem Dialog auswählen, welches Sie benutzen wollen. Wird kein JDK gefunden, werden Sie aufgefordert, den Pfad zum JDK selbst einzugeben (das kann passieren, wenn ein JDK System installiert wurde, aber die entsprechenden Registriereintragungen entfernt worden sind).

Der BlueJ Installer installiert auch ein Programm namens *vmselect.exe*. Durch Aufrufen dieses Programms können Sie später die Javaversion ändern, die BlueJ nutzt. Führen Sie *vmselect* aus, um BlueJ mit einer anderen Javaversion zu starten.

Die Wahl des JDK wird für jede BlueJ Version gespeichert. Wenn Sie verschiedene Versionen von BlueJ installiert haben, können Sie eine Version von BlueJ mit JDK 1.4.2 und eine andere BlueJ Version mit JDK 1.5 benutzen. Das Ändern der Javaversion für BlueJ gilt für alle BlueJ Installationen der gleichen Version für den gleichen Benutzer.

2.2 Installation auf Macintosh

Bitte beachten Sie, dass BlueJ nur auf MacOS X läuft.

Die Installationsdatei für MacOS heißt *BlueJ-xxx.zip*, wobei *xxx* die Versionsnummer ist. Beispielsweise heißt die Installationsdatei von BlueJ Version 2.0.0 *BlueJ-200.zip*. Sie können diese Datei auf einer CD erhalten oder Sie können Sie auf der BlueJ Homepage http://www.bluej.org herunterladen.

MacOS wird diese Datei normalerweise automatisch nach dem Download dekomprimieren. Wenn nicht, starten Sie das Entpacken durch Doppelklick.

Nach dem Dekomprimieren, finden Sie einen Ordner namens BlueJ-*xxx*. Verschieben Sie diesen Ordner in ihren Anwendungsordner (oder wohin immer Sie ihn haben möchten). Es ist keine weitere Installation notwendig.

2.3 Installation auf Linux/Unix und anderen Systemen

Die allgemeine Installationsdatei ist eine ausführbare jar Datei. Sie heißt *bluej-xxx.jar*, wobei *xxx* die Versionsnummer ist. Beispielsweise heißt die Installationsdatei von BlueJ Version 2.0.0 *bluejsetup-200.jar*. Sie können diese Datei auf einer CD erhalten oder Sie können Sie auf der BlueJ Homepage http://www.bluej.org herunterladen.

Starten Sie den Installer, indem Sie folgenden Befehl ausführen. Hinweis: Dieses Beispiel benutzt die Installationsdatei *bluej-200.jar* – Sie müssen den Dateinamen ihrer Datei verwenden (mit der korrekten Versionsnummer).

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

Wobei *<j2se-path>* das Verzeichnis ist, in dem das J2SE SDK installiert wurde.

Ein Fenster öffnet sich und Sie können das BlueJ Installationsverzeichnis und die Javaversion wählen, die verwendet werden soll, um BlueJ zu starten.

Klicken Sie *Installieren*. Nach Abschluss des Programms sollte BlueJ installiert sein.

2.4 Probleme bei der Installation

Falls Probleme bei der Installation auftreten sollten, schauen Sie in den *Frequently Asked Questions* (FAQ) auf der BlueJ Homepage (http://www.bluej.org/help/faq.html) nach und lesen Sie die *How To Ask For Help Sektion* (http://www.bluej.org/help/ask-help.html).

2.5 Umstellen der Umgebungssprache

Sie können BlueJ so einstellen, dass die Programmoberfläche in deutsch angezeigt wird. Öffnen Sie dazu die Datei <*BlueJ-Home*>/lib/bluej.defs in einem beliebigen Editor (dabei ist <*BlueJ-Home*> das Installationsverzeichnis von BlueJ). In dieser Datei finden Sie unter anderem diese Zeilen

```
bluej.language=english
#bluej.language=afrikaans


...
# bluej.language=german
...
```

Ein Doppelkreuz am Beginn einer Zeile zeigt den Beginn eines Kommentars an. Die Zeile ohne Doppelkreuz beinhaltet folglich die Spracheinstellung. Sie können die Umgebungssprach auf deutsch ändern, wenn sie vor die Zeile bluej.language=english ein Doppelkreuz setzen und dieses am Anfang der Zeile #bluej.language=german entfernen.

Alle Angaben zu Menübefehlen in diesem Tutorial beziehen sich auf die Umgebungssprache deutsch.

3 Der Start – editieren / kompilieren / ausführen

3.1 Starten von BlueJ

Auf Windows und MacOS ist ein Programm namens *BlueJ* installiert. Starten Sie es.

Auf Unix Systemen schreibt der Installer ein Skript namens *bluej* ins Installationsverzeichnis. Bei einer grafischen Oberfläche klicken Sie doppelt auf die Datei. Von einer Kommandozeile aus können Sie BlueJ mit oder ohne ein Projekt als Argument starten:

```
$ bluej
```

oder

```
$ bluej examples/people
```

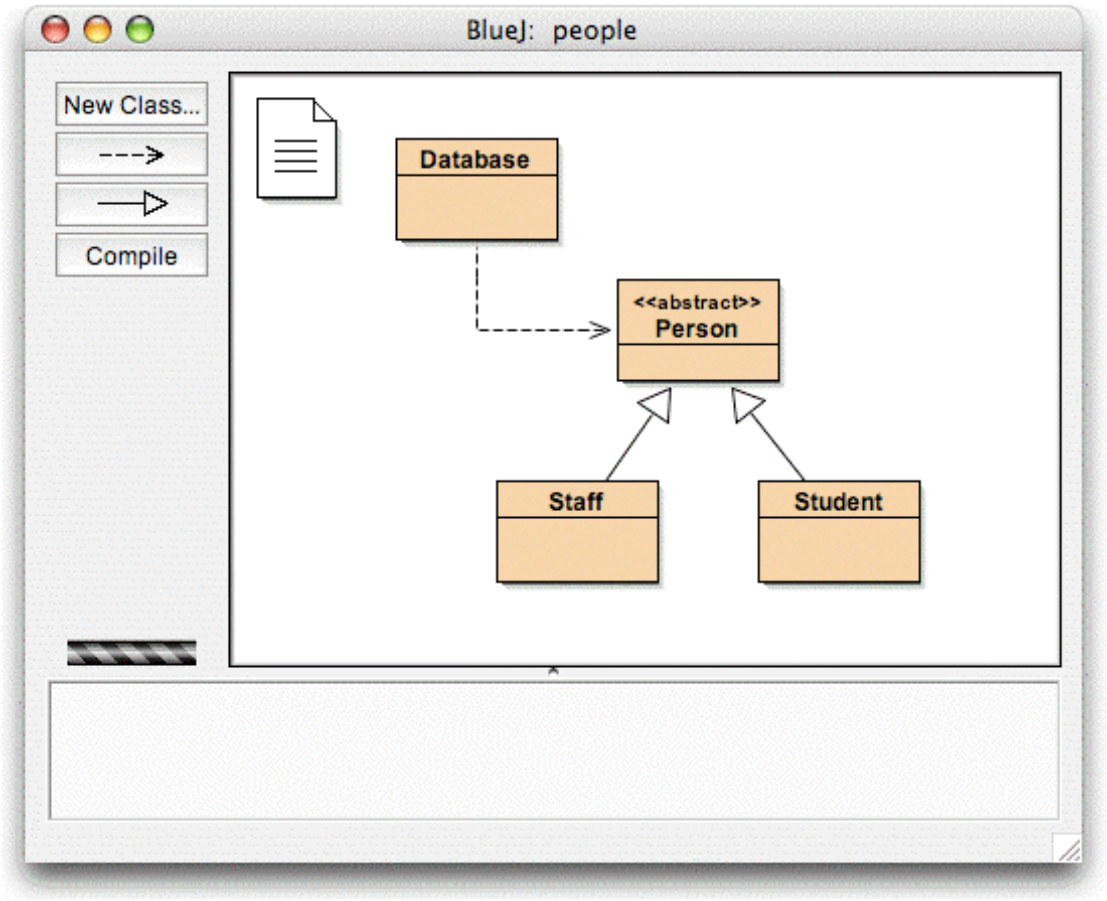


Abbildung 1: Das BlueJ Hauptfenster

3.2 Öffnen eines Projekts

Überblick: Um ein Projekt zu öffnen, wählen Sie den Befehl Öffnen *im Menü* Projekt*.*

BlueJ Projekte sind Verzeichnisse – wie normale Java-Packages auch. Die Dateien, aus denen das Projekt besteht, sind in diesem Verzeichnis enthalten.

Nach dem Start von BlueJ wählen Sie Projekt öffnen... im Menü Projekt um ein Projekt in die Umgebung zu laden.

Die Beispielprojekte finden Sie in der Standardinstallation von BlueJ im *Examples*-Verzeichnis.

Für diesen Abschnitt des Tutorials öffnen Sie das Projekt *people*. Der Ordner *Examples* befindet im Installationsverzeichnis von BlueJ. Nach dem Öffnen des Projekts sehen Sie ein Fenster ähnlich wie in Abbildung 1. Dieses Fenster muss auf ihrem System nicht identisch aussehen, die Unterschiede jedoch sollten gering sein.

3.3 Erstellen von Objekten

Überblick: Um ein Objekt zu erstellen, wählen Sie einen Konstruktor aus dem Popupmenü der Klasse.

Eine der grundlegenden Eigenschaften von BlueJ ist, dass Sie nicht nur eine komplette Anwendung ausführen können, sondern auch auf einzelne Instanzen einer beliebigen Klasse direkt einwirken und ihre öffentlichen Methoden ausführen können. Eine Programmausführung in BlueJ ist normalerweise mit dem Erstellen eines Objekts und dem anschließenden Aufrufen einer seiner Methoden verbunden. Dieses ist während der Entwicklung einer Anwendung sehr nützlich – Sie können Klassen einzeln prüfen, sobald sie geschrieben worden sind. Damit besteht keine Notwendigkeit mehr, gleich die komplette Anwendung zu schreiben.

> Anmerkung*: Statische Methoden können direkt ausgeführt werden ohne ein Objekt zu erzeugen. Eine der statischen Methoden kann „main“ sein, so können wir die gleiche Sache tun, die normalerweise in den Javaanwendungen geschieht – das Starten einer Anwendung nur durch das Ausführen einer statischen Hauptmethode. Wir kommen hierauf später zurück. Zunächst werden wir einige andere interessantere Sachen tun, die normalerweise nicht in der Javaumgebung erledigt werden können.*

Im Mittelteil des Hauptfensters sehen Sie einige Quadrate (beschriftet mit *Datenbank*, *Person*, *Personal* und *Student*). Diese Icons, stellen die Klassen dar, aus denen die Anwendung besteht. Sie können ein Popupmenü mit Befehlen erhalten, die auf eine Klasse anwendbar sind, indem Sie mit der rechten Maustaste auf das Klassenicon klicken (Macintosh: Ctrl + Klick[1]) (Abbildung 2). Die Befehle, die im Menü oben angezeigt werden, sind die *new*-Befehle, die durch jeden der Konstruktoren für diese Klasse bereitgestellt werden, gefolgt von den Operationen, die durch die BlueJ-Umgebung bereitgestellt werden.

[1] Wenn im Folgenden von einem Rechtsklick die Rede ist, lesen Macintosh-Nutzer dies als ctrl-click.

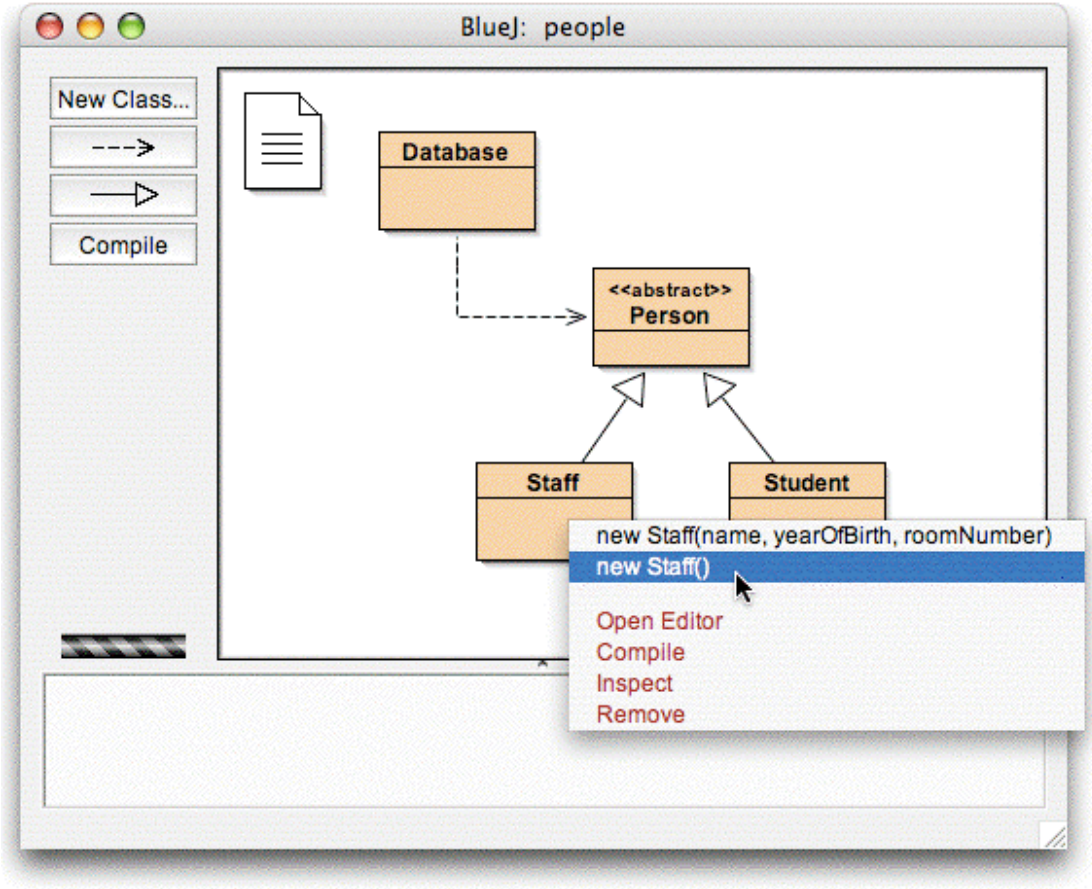


Abbildung 2: Klassenoperationen (Popupmenü)

Wir wollen zunächst ein *Staff*-Objekt erstellen. Klicken Sie dazu mit der rechten Maustaste auf das *Staff*-Icon (dadurch erscheint das oben in Abbildung 2 gezeigte Menü). Im Menü finden Sie zwei Konstruktoren: Einen mit und einen ohne Parameter. Wählen Sie jetzt den Konstruktor ohne Parameter. Ein Dialog, ähnlich wie in Abbildung 3 gezeigt, erscheint.

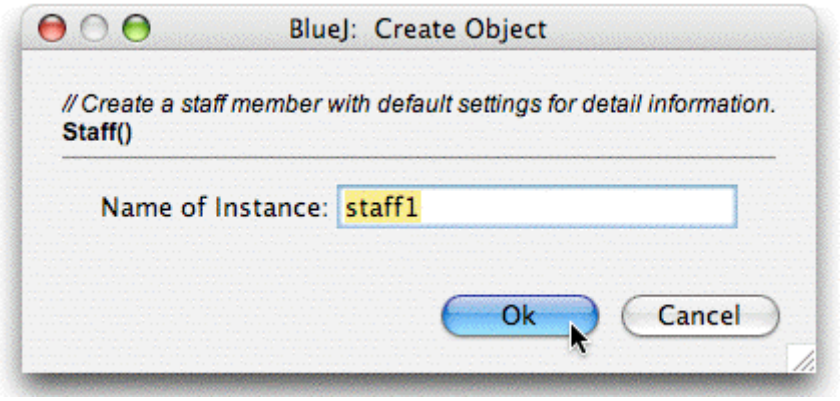


Abbildung 3: Objekterstellung ohne Parameter

Im Dialogfeld können Sie einen Namen für das zu erstellende Objekt eingeben. Gleichzeitig wird ein Name (*staff1*) vorgeschlagen. Dieser ist momentan ausreichend, also klicken Sie jetzt *OK*. Eine Instanz der Klasse *Staff* wird erstellt.

Sobald das Objekt erstellt worden ist, wird es auf die Objektleiste gesetzt (Abbildung 4). Mehr ist zum Erzeugen eines Objekts nicht notwendig: Wählen sie einen Konstruktor aus dem Klassenmenü und führen Sie ihn aus, dann wird das Objekt auf der Objektleiste abgelegt.

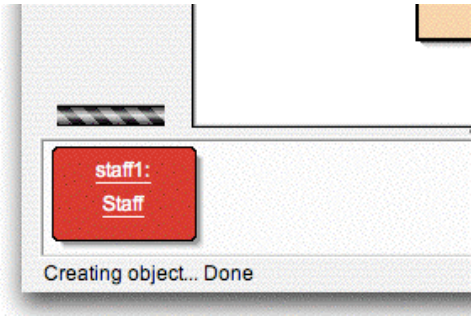


Abbildung 4: Ein Objekt auf der Objektleiste

Vielleicht habe Sie festgestellt, dass die Klasse *Person* mit <<abstract>> gekennzeichnet ist (es ist eine abstrakte Klasse). Sie werden bemerken, dass Sie (wenn Sie es versuchen) keine Objekte der abstrakten Klasse erzeugen können (das ist in der Sprachspezifikation von Java so festgelegt).

3.4 Ausführen von Methoden

Überblick: Um eine Methode auszuführen, wählen Sie sie vom Objekt-Popupmenü.

Nachdem Sie ein Objekt erstellt haben, können Sie seine öffentlichen Operationen ausführen. (Java nennt die Operationen *Methoden*.) Klicken Sie mit der rechten Maustaste auf das Objekt und ein Popupmenü mit den Objektoperationen öffnet sich (Abbildung 5). Das Menü zeigt die Methoden, die für dieses Objekt verfügbar sind und zwei spezielle Operationen, die durch die BlueJ-Umgebung bereitgestellt werden (*Inspizieren* und *Entfernen*). Wir besprechen diese später. Wir wollen uns zunächst auf die Methoden konzentrieren.

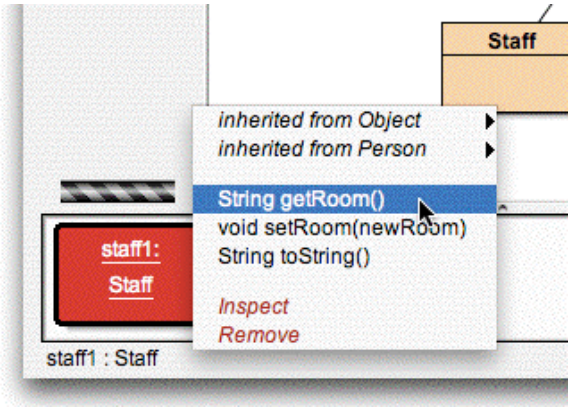


Abbildung 5: Das Objektmenü

Sie sehen, dass dort die Methoden *setRoom* und *getRoom* angezeigt werden, mit denen die Raumnummer für die Personalmitglieder eingestellt und abgefragt werden können. Versuchen Sie *getRoom* aufzurufen. Wählen Sie es einfach vom Objektmenü und es wird ausgeführt. Ein Dialog erscheint und zeigt Ihnen das Ergebnis des Aufrufs (Abbildung 6). In diesem Fall ist der Name "(unknown room)", weil wir für diese Person noch keinen Raum festgelegt haben.

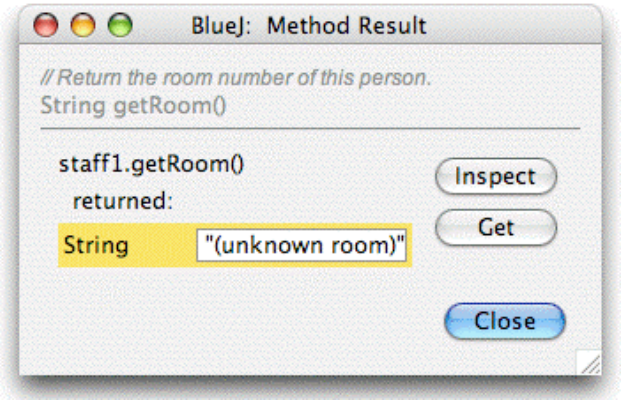


Abbildung 6: Anzeige eines Funktionsresultats

Auf Methoden, die von einer Superklasse übernommen wurden, kann über ein Untermenü zugegriffen werden. Im oberen Bereich des Popupmenüs des Objektes finden Sie zwei Untermenüs, eines für die Methoden, die von der Klasse *Object* übernommen wurden, und eines für die von *Person* (Abbildung 5) geerbten. Sie können diese Methoden von *Person* aufrufen (wie *getName*), indem Sie sie im Untermenü auswählen. Versuchen Sie es. Sie werden bemerken, dass die Antwort die gleiche ist: Es wird "(*unknown name*)" zurückgegeben, weil wir unserer Person keinen Namen gegeben haben.

Versuchen Sie jetzt, eine Zimmernummer zu einzugeben. Damit sehen Sie, wie man eine Methode mit Parametern aufruft. (Die Aurufe *getRoom* und *getName* hatten Rückgabewerte, aber keine Parameter). Rufen Sie die Funktion *setRoom* auf, indem Sie sie vom Objektmenü wählen. Ein Dialogfeld erscheint und fordert Sie auf, einen Parameter einzutragen (Abbildung 7).

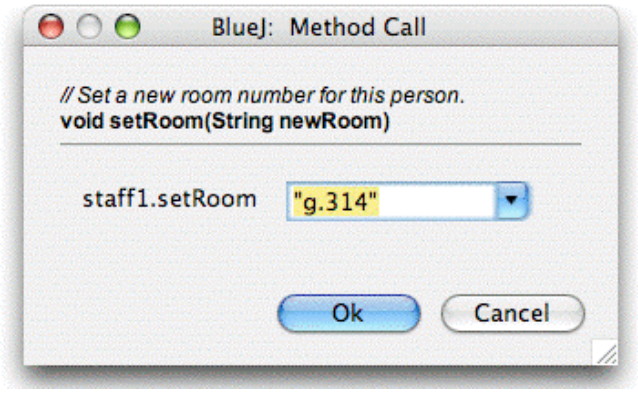


Abbildung 7: Aufrufdialogfeld einer Funktion mit Parametern

Im Dialogfeld wird oben die Schnittstelle der aufgerufenen Methode angezeigt (einschließlich Methodenkommentar und -signatur). Darunter befindet sich ein Textfeld, in das Sie den Parameter eintragen können. Die Methodensignatur zeigt an, dass ein Parameter vom Typ String erwartet wird. Geben Sie die Raumnummer als Zeichenkette in das Textfeld ein (einschließlich Anführungsstriche) und klicken Sie auf *OK*.

Das ist alles – da diese Methode keinen Wert zurückgibt, erscheint kein Ergebnisdialog. Rufen Sie jetzt erneut *getRoom* auf. Damit prüfen Sie, ob sich die Raumnummer wirklich geändert hat. Spielen Sie ruhig etwas mit der Objekterstellung und dem Aufrufen von Methoden. Versuchen Sie einen Konstruktor mit Argumenten aufzurufen und rufen Sie mehrere Methoden auf, bis Sie mit den Befehlen vertraut sind.

3.5 Bearbeiten einer Klasse

Überblick: Um den Quelltext einer Klasse zu bearbeiten, klicken Sie doppelt auf deren Icon.

Bisher haben wir uns nur mit der Schnittstelle eines Objektes beschäftigt. Jetzt ist es Zeit, „nach innen“ zu schauen. Sie können die Implementierung einer Klasse sehen, indem Sie *Bearbeiten* aus dem Popup-Menü einer Klasse wählen (zur Erinnerung: Rechtsklick auf das Klassen-Icon zeigt die Klassenbefehle) Ein Doppelklick auf das Klassenicon hat den gleichen Effekt. Der Editor wird in diesem Tutorial nicht im Einzelnen beschrieben, er sollte unkompliziert zu benutzen sein. Auf bestimmte Details des Editors wird später separat eingegangen. Öffnen Sie jetzt die Implementierung der Klasse *Staff*. Suchen Sie die Implementierung der Methode *getRoom*. Sie gibt, wie schon der Name vermuten lässt, die Raumnummer des Personalmitgliedes zurück. Ändern Sie die Methode *getRoom*, indem Sie das Präfix “room“ dem Funktionsresultat hinzufügen (damit die Methode “room M.3.18“ anstatt nur “M.3.18“ zurückgibt). Sie können dies tun, indem Sie die Zeile

```
return room;
```

ändern zu

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

BlueJ unterstützt Java vollständig, Sie müssen bei der Implementierung ihrer Klassen auf keine Besonderheiten Rücksicht nehmen.

3.6 Kompilieren

Überblick: Um eine Klasse zu kompilieren, klicken Sie den Übersetzen-*Button im Editor an. Um ein Projekt zu kompilieren, klicken Sie den* Übersetzen-*Button im Projektfenster an.*

Nachdem Sie die Änderungen eingegeben haben (noch bevor Sie etwas anderes tun), überprüfen Sie den Projektüberblick (im Hauptfenster). Sie werden bemerken, dass sich das Klassenicon für die Klasse *Staff* geändert hat: es ist jetzt gestreift. Dieses Aussehen kennzeichnet Klassen, die seit der letzten Änderung nicht kompiliert worden sind. Zurück zum Editor.

> Anmerkung*: Vielleicht fragen Sie sich, warum das Klassenicon nicht gestreift war, als Sie dieses Projekt geöffnet haben. Das liegt daran, dass die Klassen im Projekt* People *bereits kompiliert waren. Häufig werden BlueJ Projekte unkompiliert verteilt; gehen Sie also davon aus, dass in den meisten Fällen gestreiften Klassenicons zu sehen sind, wenn Sie erstmals ein Projekt öffnen.*

In der Werkzeugleiste oben im Editor befinden sich einige Buttons mit häufig verwendeten Funktionen. Einer von ihnen ist Übersetzen. Mit dieser Funktion können Sie eine Klasse direkt innerhalb des Editors kompilieren. Klicken jetzt Sie den *Übersetzen*-Button. Wenn Sie keinen Fehler gemacht haben, sollte eine Anzeige im Informationsbereich an der Unterseite des Editors erscheinen und ihnen mitteilen, dass die Klasse kompiliert wurde. Wenn Sie einen Fehler gemacht haben, der zu einem Syntaxerror führt, wird die fehlerhafte Zeile hervorgehoben und eine Fehlermeldung im Informationsbereich angezeigt. (Falls das Kompilieren gleich funktioniert hat versuchen Sie jetzt einen Fehler einzubauen – beispielsweise ein fehlendes Semikolon – und kompilieren Sie, damit Sie wissen wie eine solche Fehlermeldung aussieht).

Nachdem Sie die Klasse erfolgreich kompiliert haben schließen sie den Editor.

> Anmerkung: *Es ist keine zwingende Notwendigkeit, den Quelltext der Klasse explizit zu speichern. Quellcode wird automatisch gespeichert, wann immer es notwendig ist (z.B. wenn der Editor geschlossen wird oder bevor eine Klasse kompiliert wird). Sie können speichern, wann immer Sie wollen (es gibt einen Befehl im Menü* Klasse *des Editors), dies ist aber wirklich nur erforderlich, wenn Ihr System instabil ist, häufig abstürzt und Sie Bedenken haben, dass ihre Arbeit verloren gehen könnte.*

Die Werkzeugleiste des Projektfensters hat auch einen *Übersetzen*-Button. Dieser Befehl kompiliert das vollständige Projekt. (tatsächlich legt es fest, welche Klasse eine Rekompilierung benötigt und rekompiliert dann diese Klassen in der richtigen Reihenfolge). Probieren Sie es aus, indem Sie zwei oder mehr Klassen ändern (damit zwei oder mehr Icons im Klassendiagramm gestreift erscheinen) und klicken Sie dann den *Übersetzen*-Button. Wenn während des Übersetzens ein Fehler in den Klassen auftritt, wird der Editor geöffnet und Fehlerposition und -beschreibung werden angezeigt.

Sie haben vielleicht bemerkt, dass die Objektleiste geleert wurde. Die Objekte werden jedes Mal entfernt, wenn die Implementierung geändert wird.

3.7 Hilfe bei Kompilerfehlern

Überblick: Um Hilfe bei einer Fehlermeldung des Compilers zu erhalten, klicken Sie auf das Fragezeichen neben der Fehlermeldung.

Sehr häufig haben Kursteilnehmer anfangs Schwierigkeiten, die Fehlermeldungen des Compilers zu verstehen. Wir versuchen nun eine Hilfestellung zu geben. Öffnen Sie den Editor wieder, fügen Sie einen Fehler in der Quelldatei ein und kompilieren Sie. Nun sollte eine Fehlermeldung im Informationsbereich des Editors angezeigt werden. Am rechten Rand des Informationsbereichs erscheint ein Fragezeichen, das Sie anklicken können, um genauere Informationen über die Art des Fehlers zu erhalten (Abbildung 8).

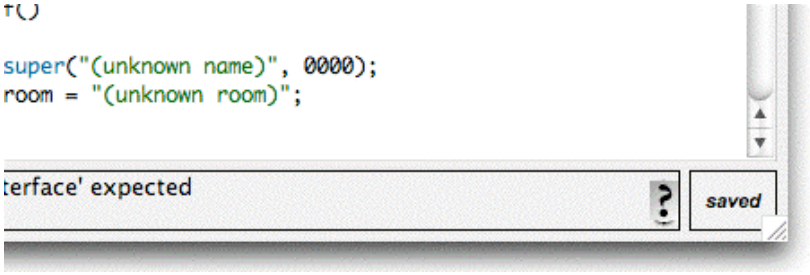


Abbildung 8: Fehlermeldungen des Compilers und Hilfe-Taste

Im Moment sind noch nicht für alle Fehlermeldungen Hilfetexte vorhanden. Einige Hilfetext müssen noch erstellt werden. Aber einen Versuch ist es wert – zu vielen Fehlern gibt es bereits Hinweise. Die verbleibenden werden in einer späteren Version von BlueJ integriert werden.

4 Ein bisschen mehr können ...

In diesem Abschnitt, zeigen wir Ihnen ein paar Dinge, die Sie in BlueJ nutzen können. Diese sind zwar nicht wesentlich, werden aber sehr oft verwendet.

4.1 Inspektion

Überblick: Die Objektinspektion erlaubt einfaches Debbuging, indem sie den internen Zustand eines Objektes zeigt.

Wann immer Sie Methoden eines Objektes ausgeführt haben, ist Ihnen möglicherweise der *Inspizieren*-Befehl aufgefallen, der zusätzlich zu den benutzerdefinierten Methoden eines Objekts verfügbar ist. (Abbildung 5). Dieser Befehl erlaubt das Überprüfen des Zustandes der Instanzvariablen von Objekten (auch „Felder“ oder „Attribute“ genannt). Versuchen Sie, einen Gegenstand mit einigen benutzerdefinierten Werten zu erstellen. (z.B. ein *Staff* mit dem Konstruktor, der Parameter erwartet). Wählen Sie dann *Inspizieren* aus dem Objekt-Menü. Ein Dialog zeigt die Felder des Objekts, ihre Typen und ihre Werte an. (Abbildung 9).

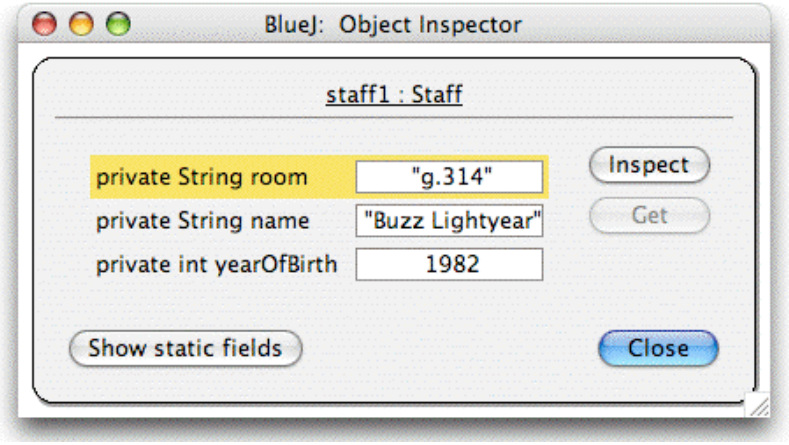


Abbildung 9: Inspizieren-Dialog

Das Inspizieren ist nützlich, um schnell zu überprüfen, ob ein verändernder Vorgang richtig ausgeführt wurde. Damit ist Inspizieren ein einfaches Debbugingwerkzeug.

Im *Staff*-Beispiel sind alle Felder primitive Datentypen (weder Objekte noch Strings). Der Wert dieser Typen kann direkt gezeigt werden. Sie können sofort sehen, ob der Konstruktor die richtigen Anweisungen gegeben hat.

In komplexeren Fällen können die Werte der Felder Referenzen auf benutzerdefinierte Objekte sein. Um ein solches Beispiel zu betrachten verwenden wir ein anderes Projekt. Öffnen Sie nun das Projekt *people2*, das auch Bestandteil der Standard BlueJ Version ist. Das Schaubild *people2* wird in Bild 10 gezeigt.

Wie Sie sehen können, hat dieses zweite Beispiel eine Klasse *Address* zusätzlich zu den Klassen, die im vorherigen Projekt vorhanden waren. Eines der Felder in der Klasse *person* ist von der benutzerdefinierten Art *Address*.

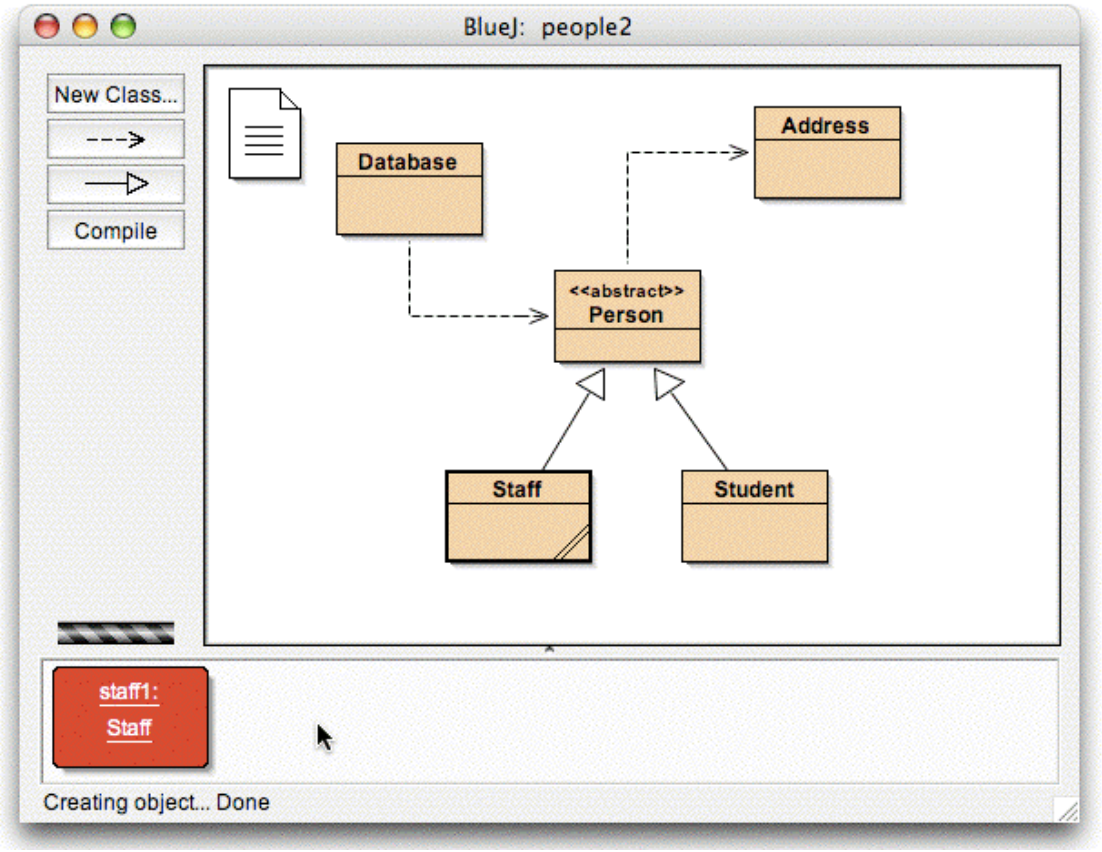


Abbildung 9: Das Projektfenster *people2*

Für das, was wir nun probieren möchten – das Inspizieren von Feldern die Objekte enthalten – erstellen Sie ein *Staff*-Objekt und rufen Sie dann seine *setAddress*-Methode auf (Sie finden sie im Popupmenu *des Objekts*). Geben Sie eine Adresse ein. Der *Staff*-Code erstellt eine Instanz der Klasse *Address* und speichert es in seinem *Address* Feld.

Jetzt inspizieren sie das Objekt *Staff*. Der resultierende Kontrolldialog wird in Abbildung 10 gezeigt. Die Felder innerhalb des *Staff* Objektes umfassen jetzt eines namens *address*. Wie Sie sehen können, wird sein Wert als Pfeil gezeigt, das bedeutet, dass es eine Referenz auf ein anderes Objekt ist. Da dieses ein komplexes, benutzerdefiniertes Objekt ist, kann sein Wert nicht direkt in dieser Liste gezeigt werden. Um die Adresse weiter zu überprüfen, wählen Sie das *address* Feld in der Liste und klicken Sie dann auf die *Inspiziere*-Taste im Dialogfeld. (Sie können auch doppelt auf *address* klicken.) Ein anderes Kontrollfenster wird geöffnet und zeigt die Details des *Address*-Objektes an (Abbildung 11).

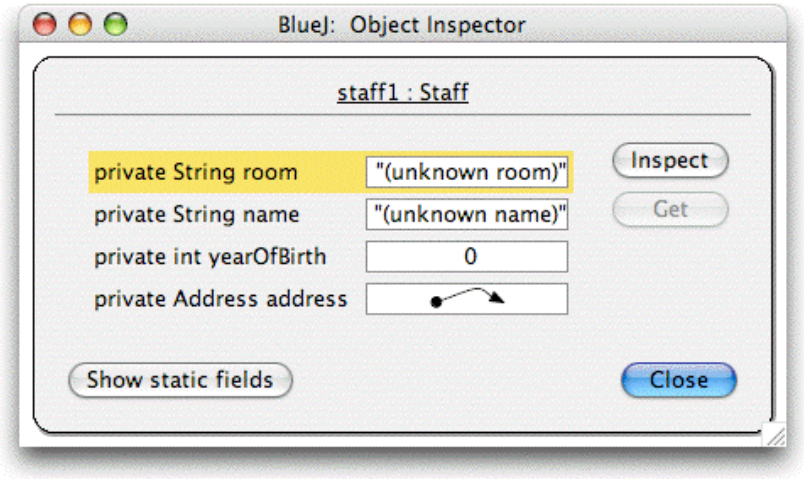


Abbildung 10: Kontrolle mit Objektshinweisen

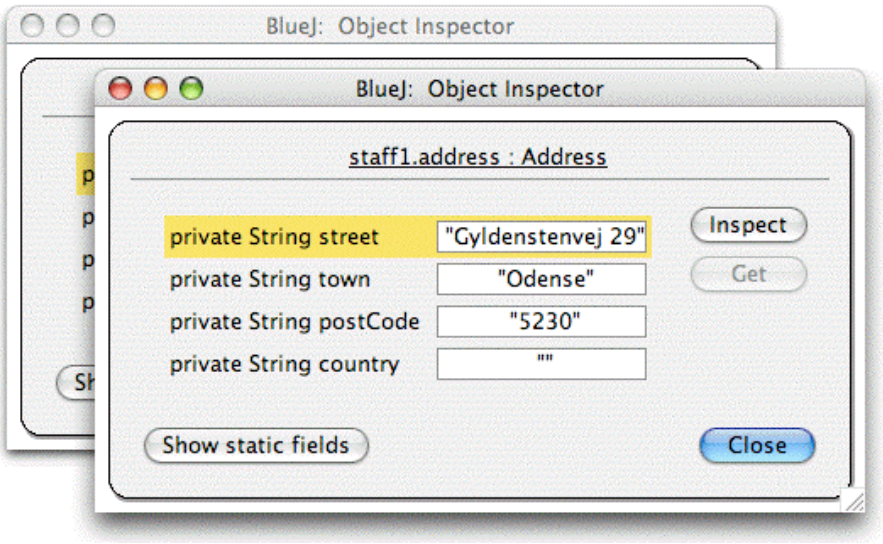


Abbildung 11: Kontrolle des internen Objektes

Wenn das ausgewählte Feld öffentlich ist, können Sie anstatt *Inspiziere* auch das Adressfeld wählen und *Hole* drücken. Dieser Vorgang setzt das gewählte Objekt auf die Objektbank. Dort können Sie es weiter überprüfen, indem sie dessen Methoden aufrufen.

4.2 Übergeben von Objekten als Parameter

Überblick: Ein Objekt kann als Parameter bei einem Methodenaufruf übergeben werden, indem man auf das Objekticon klickt.

Objekte können Methoden anderer Objekte als Parameter übergeben werden. Lassen Sie es uns an einem Beispiel versuchen. Erstellen Sie ein Objekt der Klasse *Database*. (Sie werden bemerken, dass die Klasse *Database* nur einen Konstruktor hat, der keine Parameter erwartet, so ist der Aufbau eines Objektes einfach). Das *Database* Objekt hat die Fähigkeit eine Liste von Personen zu halten. Es hat die Aufgabe, Objekte vom Typ *Person* hinzuzufügen und die

anzuzeigen, die momentan gespeichert sind (es *Database* zu nennen ist wirklich ein wenig übertrieben).

Wenn Sie nicht bereits ein *Staff*- oder *Student*-Objekt auf der Objektleiste haben, dann erstellen Sie eines von beiden. Für das folgende Beispiel benötigen Sie sowohl ein *Database*-Objekt als auch ein *Staff*- oder *Student*-Objekt auf der Objektleiste.

Nun rufen sie die *addPerson* Methode des *Database* Objektes auf. Anhand ihrer Signatur können Sie erkennen, dass ein Parameter vom Typ *Person* erwartet wird. (Bitte beachten Sie: die Klasse *Person* ist abstrakt, also gibt es keine Objekte, die direkt vom Typ *Person* sind. Aber wegen des Subtypings können Objekte vom Typ *Student* und *Staff* für *Person*-Objekte eingesetzt werden. Damit ist es möglich, ein Objekt vom Typ *Student* oder *Staff* zu übergeben, wo eines vom Typ *Person* erwartet wird.) Um das Objekt auf der Objektleiste der aufgerufenen Methode zu übergeben, können sie dessen Namen in das Eingabefeld für den Parameter eintragen oder – kürzer – auf das Objekt klicken (Damit wird ebenfalls der Name des Objekts in das Dialogfeld übertragen). Klicken Sie *OK* und der Aufruf ist beendet. Weil die Methode *addPerson* kein Wert zurückgibt, sehen Sie nicht sofort ein Ergebnis. Sie können Methode *listAll* des Objekts *Database* aufrufen, um zu sehen dass die Operation wirklich durchgeführt wurde. Die Operation *listAll* schreibt die Informationen der Person auf die Standard-Ausgabe (Konsole) Sie werden bemerken, dass sich ein Textfenster mit den Daten automatisch öffnet.

Versuch Sie es ruhig einmal mit mehr als einer in der Datenbank eingetragenen Person.

5 Erstellen eines neuen Projektes

Dieses Kapitel gibt Ihnen eine kurze Anleitung, wie man ein neues Projekt erstellt.

5.1 Erstellen des Projektverzeichnisses

Überblick: Um ein Projekt zu erstellen, wählen Sie Neues Projekt *aus dem* Projekt *Menü.*

Wir wollen nun ein Projekt neu erstellen. Wählen Sie dazu *Neues Projekt* aus dem Menü *Projekt*. Ein Dialogfenster öffnet sich und sie können einen Namen und den Speicherort für das Projekt wählen. Sie können einen beliebigen Namen für ihr Projekt eingeben. Nachdem Sie *OK* geklickt haben wird ein Verzeichnis mit dem von Ihnen gewählten Namen erstellt und das Hauptfenster zeigt das neue, leere Projekt.

5.2 Erstellen von Klassen

Überblick: Um eine neue Klasse zu erstellen, klicken Sie auf den Button Neue Klasse… *und geben einen Namen für die Klasse ein.*

Sie können nun neue Klassen erstellen, indem sie den *Neue Klasse…* Button auf der Werkzeugleiste klicken. Sie werden gebeten, einen Namen für die Klasse anzugeben – dieser Name muss ein gültiger Java Bezeichner sein.

Sie können auch aus vier Klassentypen auswählen: abstract, interface, applet oder "standard". Diese Wahl legt fest, welches Code-Grundgerüst zuerst für ihre Klasse verwendet wird. Später können sie die Art der Klasse mithilfe des Quellcodes ändern (beispielsweise indem man dem Code das Schlüsselwort "abstract" hinzufügt).

Nachdem Sie eine Klasse erstellt haben, wird diese als Icon im Diagramm dargestellt. Wenn es keine "normale" Klasse ist, wird ihr Typ Art (interface, abstract, oder applet) im Klassenicon angezeigt. Wenn Sie die neue Klasse im Editor öffnen werden Sie bemerken, dass bereits ein Grundgerüst für die Klasse erstellt wurde – das soll den Anfang erleichtern. Der Standardcode ist syntaktisch korrekt. Er kann kompiliert werden (tut aber nicht viel). Versuchen Sie, einige Klassen zu erstellen und kompilieren Sie diese.

5.3 Erstellen von Abhängigkeiten

Überblick: Um einen Pfeil zu erstellen, klicken Sie auf die Pfeil-Button und ziehen Sie den Pfeil im Diagramm oder schreiben Sie den Quelltext in den Editor.

Das Klassendiagramm zeigt Abhängigkeiten zwischen Klassen in Form von Pfeilen. Vererbungsbeziehungen („extends" oder „implements") werden als Pfeile mit einer leeren Drei-

ecksform als Spitze gezeigt; Benutzungsbeziehungen („uses“) werden durch gestrichelte Pfeile dargestellt.

Sie können Abhängigkeiten entweder grafisch hinzufügen (direkt im Diagramm) oder im Quelltext. Wenn Sie einen Pfeil grafisch hinzufügen, wird der Quelletext automatisch aktualisiert; wenn Sie die Abhängigkeit in dem Quelltext hinzufügen, wird das Diagramm aktualisiert.

Um einen Pfeil grafisch hinzuzufügen, klicken Sie die passende Pfeiltaste (hohler Pfeil für “extends” oder “implements”, gestrichelter Pfeil für “uses”) und ziehen Sie den Pfeil von einer Klasse zur anderen.

Das Hinzufügen eines Vererbungspfeiles setzt eine „extends”- oder „implements”- Deklaration in den Quelltext der Klasse (abhängig davon, ob das Ziel eine Klasse oder eine Schnittstelle ist).

Das Hinzufügen eines “uses” Pfeil ändert nicht sofort die Quelle (es sei denn, das Ziel ist eine Klasse von einem anderen Paket. In diesem Fall erzeugt es eine “import” Aussage, aber das haben wir nicht in unseren Beispielen vorgesehen). Einen Benutzungspfeil auf eine Klasse, die momentan nicht verwendet wird, erzeugt eine Warnung. Diese weist Sie darauf hin, dass eine Benutzungsbeziehung zu einer Klasse deklariert wurde, aber nicht besteht.

Die Pfeile im Text hinzuzufügen, ist einfach: Schreiben Sie den Code, wie sie es sonst auch tun würden. Sobald Sie die Klasse speichern, wird das Diagramm aktualisiert. (Denken Sie daran: Wenn Sie den Editor schließen, wird automatisch gespeichert.)

5.4 Entfernen von Elementen

Überblick: Um eine Klasse oder einen Pfeil zu entfernen, klicken sie auf die Löschfunktion aus dessen Kontextmenü.

Um eine Klasse aus dem Diagramm zu löschen, wählen Sie die Klasse aus und klicken dann auf *Entfernen* aus dem *Bearbeiten*-Menu. Sie können aber auch *Entfernen* aus dem Kontextmenü der Klasse wählen. Beide Möglichkeiten funktionieren für Pfeile ebenso: Entweder Sie wählen zuerst den Pfeil und wählen dann *Entfernen* aus dem Menü, oder Sie benutzen das Kontextmenü des Pfeils.

6 Benutzen der Direkteingabe

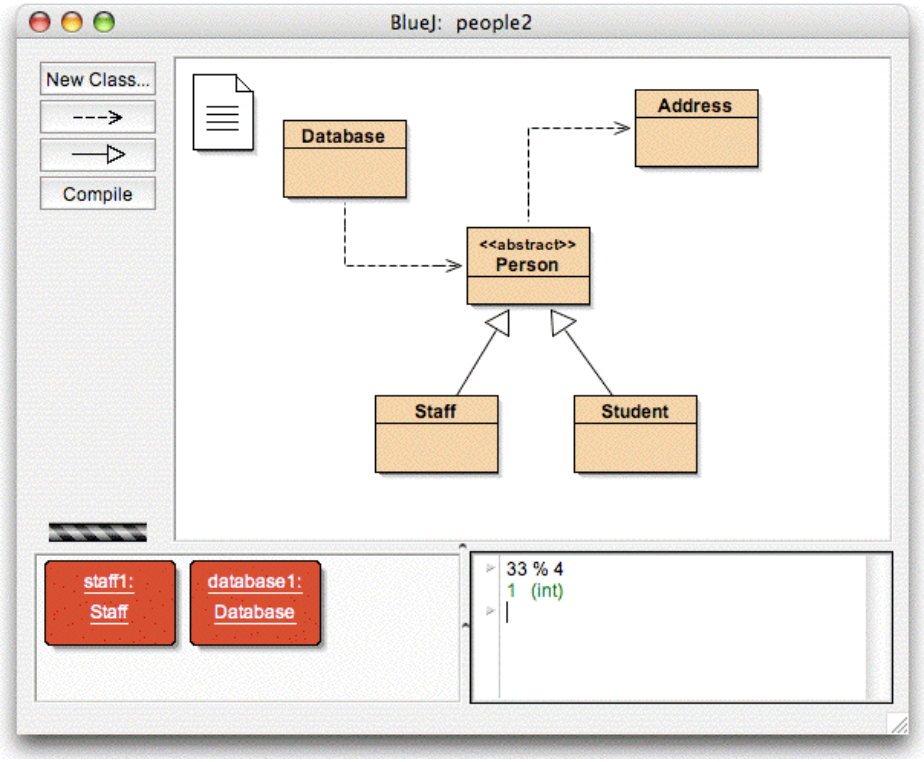
Die BlueJ Direkteingabe erlaubt die schnelle und einfache Auswertung einzelner Java-Code-Fragmente (Ausdrücke und Anweisungen). Damit kann die Direkteingabe benutzt werden, um Details der Semantik von Java zu erforschen und die Syntax zu veranschaulichen und zu experimentieren.

6.1 Aufrufen der Direkteingabe

Überblick: Um die Direkteingabe zu benutzen, wählen Sie Direkteingabe anzeigen *aus dem Menü* Ansicht.

Die Direkteingabe wird nicht standardmäßig angezeigt. Um Sie einzublenden benutzen Sie *Direkteingabe anzeigen* aus dem Menü *Ansicht*. Das Hauptfenster enthält nun die Oberfläche der Direkteingabe auf der unteren rechten Seite, neben der Objektleiste (Abbildung 9). Die horizontalen und vertikalen Ränder der Direkteingabe und der Objektleiste können verschoben werden, um die Größe anzupassen.

Das Direkteingabefenster kann nun benutzt werden, um Ausdrücke oder Anweisungen einzugeben. Beim Drücken der *Enter*-Taste wird jede Zeile ausgewertet und gegebenenfalls ein Resultat angezeigt.



9: Das Hauptfenster wird zusammen mit dem Direkteingabe angezeigt

6.2 Einfache Auswertungen von Ausdrücken

Überblick: Um Java-Ausdrücke auszuwerten, geben Sie sie einfach in die Direkteingabe ein.

Die Direkteingabe kann benutzt werden, um einfache Ausdrücke auszuwerten. Versuchen Sie folgendes Beispiel:

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

Ausdrücke können sich sowohl auf Standardwerte und -objekte von Java beziehen, als auch auf Klassen aus dem aktuellen Projekt. Die Direkteingabe zeigt den Wert des Ergebnisses an, gefolgt von seinem Typ (in den Klammern) oder eine Fehlermeldung, wenn der Ausdruck nicht korrekt ist.

Sie können auch die Objekte benutzen, die auf der Objektleiste abgelegt sind. Versuchen Sie folgendes: Legen Sie ein Objekt der Klasse Student auf der Objektleiste ab (benutzen sie das Kontextmenü der Klasse) und nennen Sie es *student1*.

In der Direkteingabe können Sie jetzt schreiben

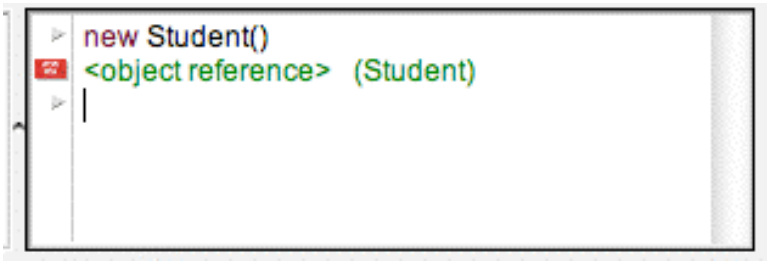
```
student1.getName()
```

Ähnlich können Sie sich auf alle vorhandenen Methoden von Ihren Projektklassen beziehen.

6.3 Empfangen von Objekten

Überblick: Um Objekte von der Direkteingabe auf die Objektleiste zu verschieben, ziehen Sie das kleine Objekticon mit der Maus vom Direkteingabe auf die Objektleiste.

Einige Resultate von Anweisungen sind Objekte, andere hingegen sind einfache Werte. Im ersten Fall wird das Ergebnis als *<object reference>* angezeigt, gefolgt von der Art des Objekts, und einem kleinen Objekticon, welches neben der Ergebnislinie angezeigt wird (Abbildung 10).



10: Ein Objekt als Ergebnis eines Direkteingabe-Ausdrucks

Ist das Resultat eine Zeichenkette, wird der Wert der Zeichenkette als das Ergebnis angezeigt, Sie sehen aber auch das kleine Objekticon (da Zeichenketten Objekte sind).

Einige Ausdrücke, mit denen Sie versuchen können Objekt zu erstellen sind

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

Das kleine Objekticon kann nun dazu benutzt werden, um mit dem Ergebnisobjekt weiterzuarbeiten. Sie können das Icon mit der Maus auf die Objektleiste ziehen (Abbildung Bild 11). Dadurch wird das Objekt auf der Objektleiste platziert, von wo aus weiterhin seine Methoden aufgerufen werden können, entweder über das Kontextmenü oder über die Direkteingabe.

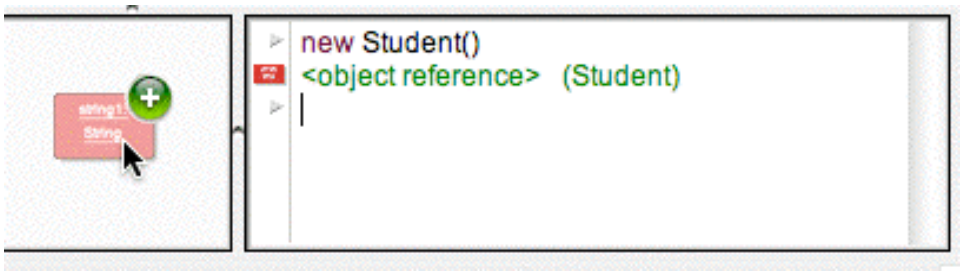


Bild 11: Das Objekt auf die Objektleiste ziehen

6.4 Inspizieren der Objekte

Überblick: Um die Ergebnisobjekte der Direkteingabe zu inspizieren, klicken Sie doppelt auf das kleine Objekticon.

Wenn Sie ein Objekt inspizieren wollen, das als Ergebnis eines Direkteingabe-Ausdrucks zurückgegeben wurde, dann können Sie dieses tun, ohne es auf der Objektleiste abzulegen: klicken Sie doppelt auf das Objekticon um den Objektinspektor zu öffnen.

6.5 Ausführung von Anweisungen

Überblick: Anweisungen, die in die Direkteingabe eingegeben werden, werden ausgeführt.

Sie können die Direkteingabe auch dazu benutzen, um Anweisungen auszuführen. (Das sind Java-Befehle ohne Rückgabewert). Versuchen Sie zum Beispiel:

```
System.out.println("Gurkensalat");

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

Die Anweisungen werden korrekt ausgewertet und ausgeführt; mit oder ohne abschließende Semikolons.

6.6 Mehrzeilige Anweisungen und Anweisungssequenzen

Überblick: Benutzen Sie Shift-Enter *am Ende einer Zeile, um eine mehrzeilige Anweisung einzugeben.*

Sie können Sequenzen von Anweisungen eintragen oder Befehle, die mehrere Zeilen umfassen, indem Sie *Shift-Enter* am Ende der eingegebenen Zeile drücken. Die Benutzung von *Shift-Enter* führt den Cursor zum Anfang der nächsten Zeile, aber das (bereits) Eingegebene wird noch nicht ausgeführt. Am Ende der zuletzt eingegebenen Zeile drücken Sie *Enter* um alle Zeilen gemeinsam auswerten zu lassen. Versuchen Sie z.B. eine *for* Schleife:

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

6.7 Arbeiten mit Variablen

Überblick: Lokale Variablen können in einzelnen und mehrzeiligen Anweisungen benutzt werden. Die Namen der Objekte auf der Objektbank dienen als Instanzfelder.

Variablen – Instanzfelder und lokale Variablen – können in eingeschränkter Weise in der Direkteingabe eingesetzt werden.

Sie können lokale Variablen in der Direkteingabe deklarieren, das ist allerdings nur bei mehrzeiligen Anweisungen sinnvoll, da Variablen zwischen verschiedenen Eingaben gelöscht

werden. Zum Beispiel: Sie können den folgenden einzelnen Absatz mehrzeilig eingeben, und es funktioniert wie erwartet:

```
int summe;
summe = 0;
for (int i=0; i<100; i++) {
 summe += i;
}
System.out.println("Die Summe beträgt: " + summe);
```

Die Eingabe der gleichen Sequenz als einzelne Zeilen wird nicht funktionieren, weil die lokale Variable *summe* nicht zwischen den einzelnen Eingaben gespeichert wird.

Man kann die Eingabe als Code innerhalb eines Methodenrumpfes auffassen. Jeden Code, den man im Rumpf einer Java Methode schreiben kann, kann man auch in das Direkteingabe eingeben. In jedem Fall werden mehrere einzelne Eingaben als Teil *verschiedener* Methoden interpretiert, sodass von einer Zeile ein Zugriff auf eine Variable, die in einer anderen Zeile deklariert wurde, nicht möglich ist.

Sie können Objekte auf der Objektleiste mit Instanzfeldern vergleichen. Sie können keine neuen Objektattribute innerhalb eines Methodenrumpfes (oder in der Direkteingabe) definieren, sehr wohl aber auf Attribute oder Methoden eines Objektes (auf der Objektleiste) zugreifen.

Sie können ein neues Instanzfeld erstellen, indem sie ein Objekt von der Direkteingabe auf die Objektleiste ziehen.

6.8 Eingabe-History

Überblick: Benutzen Sie die Pfeil-Auf- *und* Pfeil-Ab-*Tasten, um die Eingabe-History zu verwenden.*

Die Direkteingabe speichert die Reihenfolge Ihrer Eingaben. Wenn sie die Auf- oder Ab-Pfeile ihrer Tastatur benutzen, können sie ganz einfach vorherige Eingabezeilen aufrufen und können diese ändern, bevor sie die Eingabe abschließen.

7 Debugging

Dieser Abschnitt stellt die wichtigsten Aspekte der Debuggingfunktion von BlueJ dar. In Gesprächen mit Informatiklehrern haben wir oft gehört, dass der Einsatz eines Debuggers sehr gut wäre, aber die Zeit nicht ausreiche, um ihn einzuführen. Schüler haben Probleme im Umgang mit dem Editor, Compiler und dem Ausführen der Programme; das heißt, dass keine Zeit mehr vorhanden ist, ein weiteres komplexes Werkzeug einzuführen

Das ist der Grund, warum wir uns entschlossen haben, den Debugger so einfach wie möglich zu gestalten. Das Ziel war ein Debugger, den man innerhalb von 15 Minuten erklären kann, und den die Schüler dann ohne weitere Instruktionen nutzen können. Lassen Sie uns sehen, ob wir erfolgreich waren.

Als erstes haben wir die Funktionalität von traditionellen Debuggern auf 3 Tätigkeiten beschränkt:

- Setzen von Haltepunkten
- Schrittweises Ausführen des Quelletextes
- Inspizieren von Variablen

Jede der 3 Aufgaben ist sehr einfach. Wir werden uns jetzt jede einzelne ansehen.

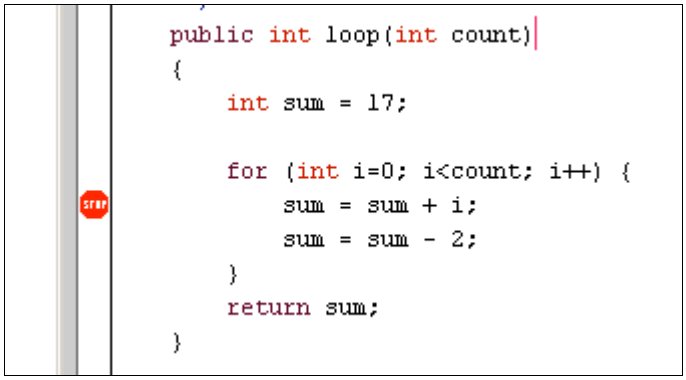
Öffnen Sie zunächst das Projekt *debugdemo*, das sich im *examples*-Verzeichnis befindet. Dieses Projekt enthält einige Klassen die ausschließlich dazu dienen, die Funktionalität des Debuggers zu erklären – ansonsten haben diese Klassen keinen Nutzen.

7.1 Setzen von Haltepunkten

Überblick: Um einen Haltepunkt zu setzen, klicken Sie in die Haltepunktfläche links neben dem Text im Editor.

Durch das Setzen eines Haltepunktes können Sie die Programmausführung an einem bestimmten Punkt im Code anhalten. Wenn die Ausführung unterbrochen ist, können Sie den Zustand Ihrer Objekte untersuchen. Damit werden Sie leichter verstehen können, was in Ihrem Code passiert.

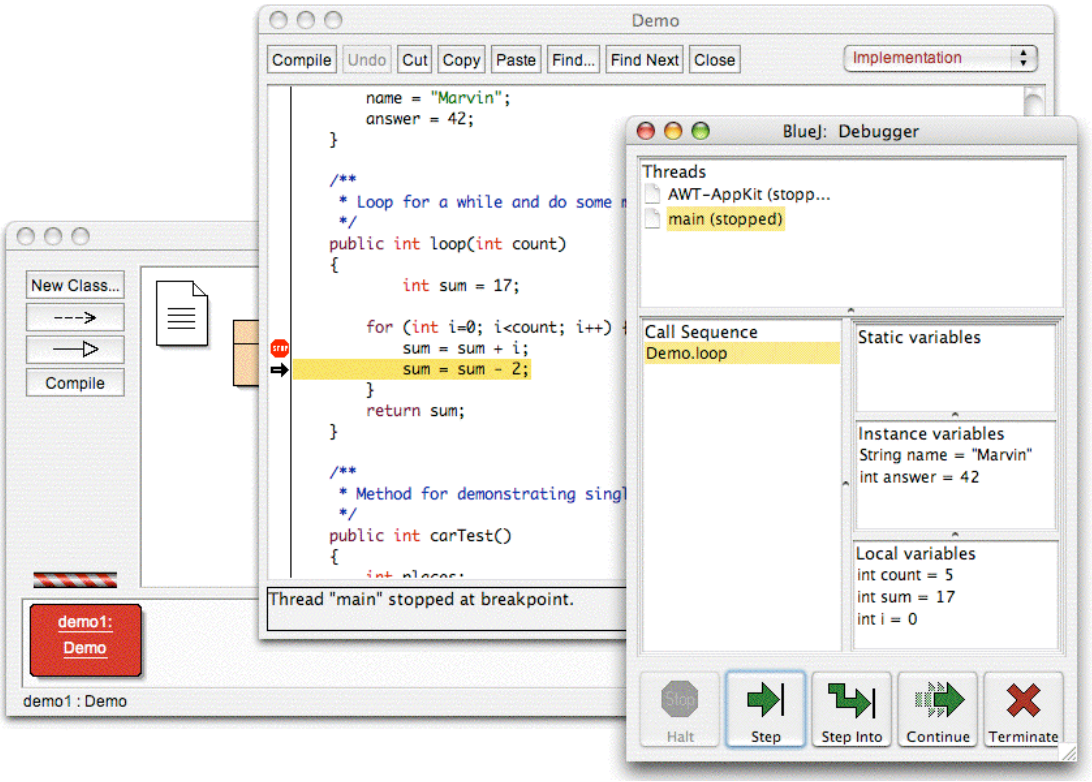
Im Editor befindet sich links vom Programmtext die Haltepunktfläche (Abbildung 12). Sie können einen Haltepunkt setzen, indem Sie in diese Fläche klicken. Ein kleines Stoppzeichen erscheint und markiert den Haltepunkt. Versuchen Sie das nun: Öffnen Sie die Klasse *Demo*, suchen Sie die Methode *loop*, und setzen Sie einen Haltepunkt irgendwo in der *for*-Schleife. Das Stoppzeichen sollte nun im Editor zu sehen sein.



12: Ein Haltepunkt

Wenn die Codezeile erreicht wird, in der der Haltepunkt gesetzt wurde, wird die Ausführung unterbrochen. Das sehen wir jetzt an.

Erstellen Sie ein Objekt der Klasse *Demo* und rufen die *loop* Methode mit einem Parameter, zum Beispiel 10, auf. Sobald der Haltepunkt erreicht ist, öffnen sich das Editorfenster (die aktuelle Zeile des Codes wird angezeigt) und das Debug-Fenster. Es sollte ähnlich wie in Abbildung 13 aussehen.



13: Das Debug-Fenster

Die Zeile, die als nächstes ausgeführt wird, ist farbig hinterlegt. (Die Ausführung wurde gestoppt *bevor* die Zeile verarbeitet wurde.)

7.2 Den Code schrittweise ausführen

*Überblick: Um Schritt für Schritt durch den Code zu gehen, benutzen Sie die „*Schritt“*- und* „Schritt hinein“*-Buttons des Debbuggers.*

Nachdem die Ausführung gestoppt wurde (damit konnten wir uns überzeugen, dass die Methode wirklich ausgeführt und dieser Punkt im Code auch wirklich erreicht wird), können wir Schritt für Schritt durch den Code gehen und sehen, wie die Ausführung abläuft. Dieses erreichen Sie durch einen Klick auf den *Schritt*-Button im Debug-Fenster. Sie sollten sehen, dass sich die Zeile im Editor ändert (die Hervorhebung bewegt sich mit der gerade ausgeführten Zeile mit). Jedes Mal, wenn Sie auf den *Schritt*-Button klicken, wird eine einzelne Zeile ausgeführt, dann wird das Programm wieder angehalten. Achten Sie darauf, dass sich die Werte der Variablen ändern, die im Debug-Fenster angezeigt werden (im Beispiel der Wert von *summe*.) So können Sie sich Schritt für Schritt durcharbeiten und beobachten, was passiert. Sollte Ihnen das zu langweilig werden, so klicken Sie erneut auf den Haltepunkt um diesen zu entfernen und dann den *Fortsetzen*-Button im Debugger um die Ausführung neu zu starten und normal weiterlaufen zu lassen.

Lassen Sie uns das nochmals mit einer anderen Methode versuchen. Setzen Sie einen Haltepunkt in der Klasse *Demo*, Methode *carTest()*, in der Zeile

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

Rufen Sie die Methode auf. Wenn der Haltepunkt erreicht ist, erfolgt in der nächsten auszuführenden Zeile ein Methodenaufruf der Methode *seats()* in der Klasse *Car*. Klicken Sie auf *Schritt*, so wird im nächsten Schritt die komplette Zeile abgearbeitet. Klicken Sie jetzt *Schritt hinein*. Wenn Sie einen Schritt in einen Methodenaufruf hineingehen, dann betreten Sie diese Methode und können selbige ebenfalls Zeile für Zeile durchlaufen (und nicht mit nur einem Schritt wie oben angedeutet). In diesem Fall werden Sie sich in der Methode *seats()* der Klasse *Car* wieder finden. Sie können sich jetzt durch die Methode klicken bis Sie das Ende erreicht haben; dann kehren Sie zu der aufrufenden Methode zurück. Achten Sie darauf, wie sich das Debug-Fenster ändert.

Schritt und *Schritt hinein* verhalten sich identisch, wenn die aktuelle Programmzeile keinen Methodenaufruf enthält.

7.3 Inspizieren von Variablen

Überblick: Das Inspizieren von Variablen ist einfach – sie werden automatisch im Debugger angezeigt.

Wenn Sie ihren Code debuggen ist es wichtig, den Zustand Ihrer Objekte kontrollieren zu können (lokale Variablen und Instanzvariablen).

Diese Variablenkontrolle ist sehr einfach – das Meiste haben Sie bereits gesehen. Sie brauchen keine speziellen Befehle, um Variablen zu inspizieren: statische Variablen, Instanzvariablen des aktuellen Projektes und lokale Variablen der aktuellen Methode werden ständig angezeigt und aktualisiert.

Sie können Methoden in der Aufrufreihenfolge auswählen, um Variablen anderer aktiver Objekten und Methoden zu sehen. Versuchen Sie beispielsweise wieder einen Haltepunkt in der

Methode *carTest()* zu setzen. Auf der linken Seite des Debuggerfensters sehen Sie die Aufrufreihenfolge. Zurzeit erscheinen

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

Dieses zeigt, dass **`Car.seats`** von **`Demo.carTest`** aufgerufen wurde. Sie können **`Demo.carTest`** in der Liste auswählen, um den Quellcode sowie die aktuellen Werte der Variablen der Methode zu kontrollieren.

Wenn Sie die Zeile mit der **`new Car(...)`**-Anweisung ausführen können Sie beobachten, dass der Wert der lokalen Variable als <*object reference*> angezeigt wird. Alle Werte von Objekten (außer Zeichenketten) werden so angezeigt. Sie können diese Variable kontrollieren, indem Sie sie doppelt anklicken. Dadurch wird ein Objektkontrollfenster geöffnet, identisch mit dem schon beschriebenen. Es gibt keinen Unterschied zwischen der Inspektion von Objekten im Debug-Fenster und auf der Objektleiste.

7.4 Anhalten und Beenden

Überblick: Anhalten *und* Beenden *können benutzt werden, um eine Durchführung temporär oder permanent zu stoppen.*

Manchmal läuft ein Programm lange Zeit, und Sie fragen sich, ob das so in Ordnung ist. Möglicherweise ist es in eine Endlosschleife geraten, vielleicht braucht es auch nur sehr lang. Dies können wir herausfinden. Rufen Sie die Methode *longloop()* aus der Klasse *Demo* auf. Diese Methode läuft eine Weile.

Nun wollen wir wissen, was los ist. Rufen Sie das Debug-Fenster auf, falls es nicht schon eingeblendet ist.

Klicken Sie nun den *Anhalten*-Knopf. Die Durchführung wird unterbrochen, so, als hätten wir einen Haltepunkt erreicht. Nun können Sie eine größere Zahl von Einzelanweisung durchführen lassen und die Variablen beobachten um zu sehen, dass alles in Ordnung ist – es braucht nur etwas mehr Zeit. Sie können *Fortsetzen* und *Anhalten* mehrmals drücken, um zu sehen wie schnell gezählt wird. Wenn Sie nicht weitermachen wollen (zum Beispiel, wenn Sie entdeckt haben, dass es sich um eine Endlosschleife handelt) können Sie *Beenden* betätigen um die ganze Durchführung zu beenden. *Beenden* sollten Sie nicht zu häufig verwenden – Sie könnten korrekt geschriebene und erzeugt Objekte in einem inkonsistenten Zustand hinterlassen; es ist also ratsam, diese Maßnahme nur im Notfall zu verwenden.

8 Erstellen von Einzelanwendungen

Überblick: Um eine Einzelanwendung (die sich auch ohne BlueJ starten lässt) zu erstellen, benutzen Sie Projekt – Als jar-Archiv speichern …

BlueJ kann ausführbare jar-Dateien erstellen. Ausführbare jar-Dateien können auf einigen Systemen durch Doppelklick ausgeführt werden (beispielweise Windows und MacOS X) oder bei Eingabe des Befehls **`java -jar <file-name>.jar`** (Unix- oder DOS-Aufforderung).

Wir versuchen dieses mit dem Beispielprojekt *hello*. Öffnen Sie es (Sie finden es im *examples* Verzeichnis). Überprüfen Sie, ob das Projekt kompiliert ist. Wählen Sie die *Als jar-Archiv speichern …* -Funktion aus dem Menü *Projekt*.

Ein Dialog öffnet sich, in dem die Hauptklasse angeben können (Abbildung 14). Die Klasse muss über eine korrekt definierte *main-Methode* verfügen (deren Signatur immer **`public static void main(String[] args)`** ist).

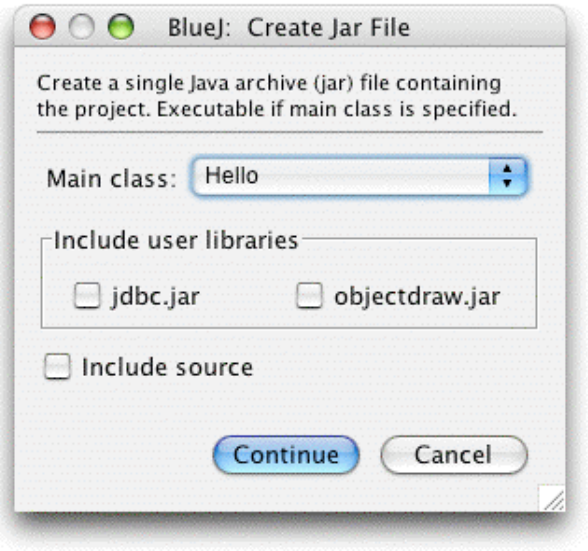


Abbildung 14: Der "Create Jar File" Dialog

In unserem Beispiel ist es einfach die Hauptklasse zu wählen: Es gibt nur eine Klasse. Wählen Sie *Hello* im Popupmenü. Wenn Sie andere Projekte haben, dann wählen Sie die Klasse mit der main-methode, die sie ausführen wollen.

Im Normalfall werden Sie Quellentexte nicht in ausführbare Dateien schreiben. Aber Sie können das tun, um Ihre Quelltexte zu verteilen. (Sie können das jar-Format beispielsweise verwenden, um ihr gesamtes Projekt in einer einzigen Datei via E-Mail zu versenden.)

Wenn Sie BlueJ für die Benutzung von eigenen Bibliotheksklassen eingerichtet haben (entweder über Einstellungen in *Werkzeuge – Einstellungen – Bibliotheken* oder durch Benutzung des *lib/userlib* Verzeichnisses) sehen Sie eine Feld in der Mitte des Dialogs mit der Überschrift *Benutzerbibliotheken mitspeichern*. (Verwenden Sie keine eigenen Bibliotheksklassen,

dann wird diese Fläche ausgeblendet.) Sie sollten jede Klasse auswählen, die von ihrem aktuellen Projekt benutzt wird.

Klicken Sie *Weiter*. Im nächstes Fenster geben Sie einen Namen für Ihre zu erstellende jar-Datei ein. Tippen Sie *hello* und klicken Sie auf *Erstellen*.

Sollten Sie keine Bibliotheken einschließen möchten, wird eine Datei namens *hello.jar* erstellt, im anderen Fall wird ein Verzeichnis namens *hello* erstellt, und darin die jar-Datei *hello.jar*. Dieses Verzeichnis beinhaltet alle notwendigen Bibliotheken. Ihre jar-Datei erwartet, die verwendeten Bibliotheksklassen im gleichen Verzeichnis zu finden wie sich selbst – achten Sie darauf, dass die jar-Dateien zusammenbleiben, wenn Sie sie verschieben.

Sie können nun doppelt auf die jar-Datei klicken wenn die Applikation eine grafische Oberfläche besitzt. Unser Beispiel benutzt die Konsole, deswegen müssen wir es von der Eingabeaufforderung aus starten. Versuchen Sie nun, die jar-Datei aufzurufen.

Öffnen Sie ein Terminal oder ein DOS-Fenster. Gehen Sie dann zu dem Verzeichnis, in dem sie die jar-Datei gespeichert haben (Sie sollten eine Datei namens *hello.jar* sehen). Für den Fall, dass Java richtig auf ihrem Computer installiert ist, sollten die Datei nach der Eingabe von:

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

ausgeführt werden.

9 Applets erstellen

9.1 Ein Applet aufrufen

Überblick: Um eine Applet zu starten, wählen Sie Applet ausführen *vom Popupmenü des Applets.*

BlueJ erlaubt es, Applets ebenso wie Anwendungen zu erstellen und auszuführen. Im *examples*-Verzeichnis finden Sie auch ein Applet. Zuerst möchten wir ein Applet ausführen. Öffnen Sie das Projekt *appletdemo* aus dem *examples*-Verzeichnis.

Wie Sie sehen, besitzt dieses Projekt nur eine Klasse, die *CaseConverter* heißt. Das Icon der Klasse ist als Applet markiert (mit dem Tag <<applet>>). Nach dem Kompilieren wählen Sie den Befehl *Applet ausführen* vom Popupmenü der Klasse

Ein Dialogfenster öffnet sich und Sie können einige Werte einstellen (Abbildung15).

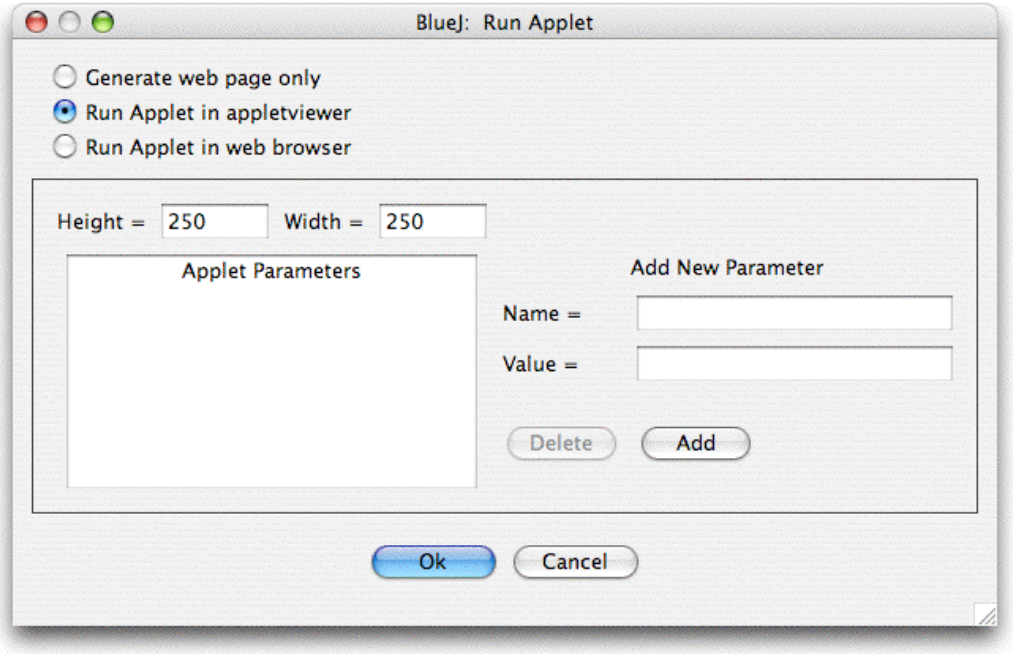


Abbildung15: Der "Run Applet" Dialog

Wie Sie sehen, haben Sie die Möglichkeit eine Applikation in einem Web-Browser oder im „Applet-Viewer“ laufen zu lassen (oder nur um die Webseite zu generieren ohne sie laufen zu lassen). Behalten Sie die Standardeinstellungen bei und klicken Sie *OK*. Nach ein paar Sekunden, sollte sich der „Applet-Viewer“ öffnen, der das CaseConverter Applet enthält.

Der „Applet-Viewer“ wurde zusammen mit dem J2SE SDK (ihrer Java Installation) installiert, so ist immer sichergestellt, dass er die gleiche Version wie der Java Compiler hat. Er verursacht im Allgemeinen wenige Probleme als ein Internetbrowser. Ihr Internetbrowser

kann beispielsweise eine andere Version von Java enthalten. Abhängig davon, welchen Browser Sie benutzen, können Problemen entstehen. Mit den meisten aktuellen Browsern sollte es dennoch gut funktionieren. Unter Microsoft Windows und MacOS-Systemen benutzt BlueJ den Standardbrowser. Auf Unix Systemen wird der Browser in den BlueJ Einstellungen festgelegt.

9.2 Erstellen eines Applets

Überblick: Um ein Applet zu erstellen, klicken Sie den Button Neue Klasse … *und wählen* Applet *als Klassentyp.*

Nach dem Sie gesehen haben, wie man ein Applet ausführt, wollen wir nun ein eigenes erstellen.

Erstellen Sie eine neue Klasse und wählen Sie *Applet* als Klassentyp (dies können Sie im Dialog *Neue Klasse ...* auswählen). Übersetzen Sie das Applet und starten Sie es.

Applets (wie auch andere Klassen) werden mit einem Standardgerüst erstellt, das einen simplen Code beinhaltet. Dieser einfache Code enthält für Applets 2 Zeilen. Sie können nun den Editor öffnen und das Applet ändern, um ihren eigenen Code hineinzuschreiben.

Sie sehen, dass alle allgemeinen Methoden der Applikation bereits vorhanden sind, jede mit einem Kommentar, die deren Zweck erklärt. Den Beispielcode finden Sie in der Methode *paint*.

9.3 Testen eines Applets

In einigen Situationen kann es nützlich sein, eine Instanz eines Applets auf der Objektleiste zu erstellen (wie bei einer normale Klassen). Sie können dies tun – der Konstruktor wird im Popupmenü des Applets angezeigt. Von der Objektleiste aus können Sie nicht das Applet nicht vollständig ausführen lassen, Sie haben jedoch die Möglichkeit, einige Methoden aufzurufen. Das kann nützlich sein, um einzelne Methoden zu prüfen, die Sie als Teil Ihrer Applet-Implementierung geschrieben haben.

Wenn Sie einen Haltepunkt in ihrem Applet gesetzt haben und es in einem „Applet-Viewer“ oder einem Browser laufen lassen, wird das keinen Einfluss auf die Ausführung des Applets haben. Das ist deshalb so, weil die eigenen virtuellen Maschinen von Browser und „Applet-Viewer“ keine Kenntnis von den Haltepunkten in BlueJ haben.

Wenn Sie Haltepunkte und Einzelschritte in einem Applet nutzen wollen, können Sie die Klasse *AppletWindow* nutzen, die von Michael Trigoboff geschrieben wurde. Die Klasse unterstützt einen Frame, der das Applet direkt in BlueJ laufen lässt, so dass die normalen Methoden des Debuggers funktionieren. Sie finden die Klasse und eine Demo in der *Resources*-Sektion der BlueJ Homepage.

10 Andere Anwendungsmöglichkeiten

10.1 Öffnen von Nicht-BlueJ Paketen in BlueJ

Überblick: Nicht-BlueJ Paketen laden Sie mit Fremdprojekt öffnen ... *im Menü* Projekt.

In BlueJ können Sie existierende Pakete öffnen, die außerhalb von BlueJ erstellt wurden. Um dies zu tun, wählen Sie Projekt – Fremdprojekt öffnen... im Menü. Suchen Sie das Verzeichnis, das den Java Quellcode beinhaltet, klicken Sie dann den In BlueJ öffnen-Button. Sie werden gefragt, ob Sie das Verzeichnis öffnen wollen.

10.2 Hinzufügen von bereits existierenden Klassen zu ihrem Projekt

Überblick: Fremde Klassen können in ein Projekt kopiert werden, indem man den Befehl Klasse aus Datei hinzufügen … *benutzt.*

Häufig werden Sie eine Klasse benutzen wollen, die Sie außerhalb ihrer BlueJ-Umgebung erstellt oder erhalten haben. Zum Beispiel gibt ein Lehrer Schülern Java Klassen, die sie in einem Projekt benutzen sollen. Sie können leicht eine existierende Klasse in ihr Projekt einbringen, indem Sie auf Bearbeiten – Klasse aus Datei hinzufügen ... im Menü klicken. Dies lässt Sie eine Java Quelldatei (die mit *.java* endet) importieren.

Wenn die Klasse in das Projekt kopiert wird, dann wird eine Kopie dieser Klasse erstellt und im aktuellen Projektverzeichnis abgelegt. Der Effekt ist der gleiche, als wenn Sie eine neue Klasse erstellt und den gesamten Quelltext (ab)geschrieben hätten.

Eine Alternative ist es, eine Quelltextdatei der neuen Klasse außerhalb von BlueJ in das Projektverzeichnis zu kopieren. Wenn Sie das nächste Mal das Projekt öffnen, wird die Klasse im Projektdiagramm angezeigt.

10.3 Aufrufen von *main* und anderen statischen Methoden

Überblick: Statische Methoden können vom Popupmenü der Klasse aufgerufen werden.

Öffnen Sie das *hello* Projekt aus dem *examples* Verzeichnis. Die einzige Klasse in dem Projekt (Klasse *Hello*) definiert eine main-Methode.

Klicken Sie rechts auf die Klasse. Sie werden bemerken, dass das Klassenmenü nicht nur einen Konstruktor der Klasse enthält, auch die statische Methode *main*. Sie können nun *main* von diesem Menü aus aufrufen (ohne vorher ein Objekt zu erstellen).

Alle statischen Methoden können so aufgerufen werden. Die standard-main-Methode erwartet ein String-Array als Parameter. Sie können eine String-Array übergeben, indem Sie die normale Java-Syntax-benutzen. Zum Beispiel, können Sie

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

(einschließlich der Klammern) der Methode übergeben. Probieren Sie es aus!

> Anmerkung: *Im Standard Java können Konstanten-Arrays nicht als Parameter bei Methodenaufrufen benutzt werden. Sie dürfen nur zur Initialisierung eingesetzt werden. Um in BlueJ einen interaktiven Aufruf von Standard-main-Methoden zu ermöglichen, haben wir die Übergabe von Konstanten-Arrays als Parameter erlaubt.*

10.4 Generieren von Dokumentationen

Überblick: Um eine Dokumentation für ein Projekt zu generieren, wählen Sie Dokumentation erzeugen *aus dem Menü* Werkzeuge.

Sie können Dokumentationen für ihr Projekt im Standard *javadoc* Format in BlueJ erstellen. Um dies zu tun, wählen Sie Documentation erzeugen im Menü Werkzeuge. Diese Funktion erstellt die Dokumentation für alle Klassen im Projekt anhand des Klassenquelltextes und öffnet einen Internetbrowser um sie anzuzeigen.

Sie können auch eine Dokumentation für eine einzelne Klasse im BlueJ Editor erstellen und betrachten. Um dies zu tun, öffnen Sie den Editor und benutzen die Auswahlliste in der Toolbar des Editors (oben rechts). Wechseln sie dort von *Implementation* zu *Interface*. Dies zeigt die *javadoc* Dokumentation (die Schnittstelle der Klasse) im Editor an.

10.5 Arbeiten mit Bibliotheken

Überblick: Die Java Standard API kann angezeigt werden, wenn man die im Menü Hilfe *den Punkt* Java Klassenbibliotheken *auswählt*.

Wenn Sie ein Java-Programm schreiben, werden Sie sich häufig auf die Java Standardbibliothek beziehen müssen. Sie können einen Internetbrowser öffnen um die JDK API Dokumentation aufzurufen, indem Sie auf Java Klassenbibliotheken im Menü Hilfe klicken (falls Sie mit dem Internet verbunden sind).

Die JDK Dokumentation kann auch lokal gespeichert werden (offline). Näheres wird in der Hilferubrik auf der BlueJ Homepage erläutert.

10.6 Objekten von Bibliotheksklassen erstellen

Überblick: Um Objekte durch Bibliotheksklassen zu erstellen, wählen Sie Werkzeuge – Klasse aus Bibliothek verwenden ...

BlueJ beinhaltet außerdem eine Funktion, um Objekte von Klassen zu erstellen, die nicht Teil ihres Projektes, aber in einer Bibliothek definiert sind. Sie können beispielsweise Objekte der

Klasse String oder ArrayList erstellen. Dies kann sehr nützlich sein, um diesen Bibliotheksklassen schnall auszuprobieren.

Sie können ein Bibliotheksobjekt erstellen durch Auswahl von Klasse aus Bibliothek verwenden ... im Menü Werkzeuge. Ein Fenster öffnet sich, das Sie auffordert, einen kompletten Klassennamen einzugeben, etwa *java.lang.String*. (Beachten Sie: Sie müssen eine vollqualifizierten Klassennamen eingeben, dieser muss auch den Namen des Packages enthalten, in dem sich die Klasse befindet.)

Im Eingabedialog finden sie eine Auswahlliste, die zuletzt benutzte Klassen zeigt. Sobald ein Klassenname eingetragen worden ist, und Sie *Enter* drücken, werden alle Konstruktoren und statische Methoden dieser Klasse in einem Dialog aufgelistet. Jeder dieser Konstruktoren oder jede dieser statischen Methoden können jetzt aufgerufen werden, indem man sie aus dieser Liste wählt.

Dieser Aufruf läuft genauso ab wie jeder andere Konstruktor- oder Methodenaufruf.

11 Die Überblicke

Der Start – editieren / kompilieren / ausführen

1. Um ein Projekt zu öffnen, wählen Sie den Befehl *Öffnen* im Menü *Projekt.*
2. Um ein Objekt zu erstellen, wählen Sie einen Konstruktor aus dem Popupmenü der Klasse.
3. Um eine Methode auszuführen, wählen Sie sie vom Objekt-Popupmenü.
4. Um den Quelltext einer Klasse zu bearbeiten, klicken Sie doppelt auf deren Icon.
5. Um eine Klasse zu kompilieren, klicken Sie den *Übersetzen-Button* im Editor an. Um ein Projekt zu kompilieren, klicken Sie den *Übersetzen-Button* im Projektfenster an.
6. Um Hilfe bei einer Fehlermeldung des Compilers zu erhalten, klicken Sie auf das Fragezeichen neben der Fehlermeldung.

Ein bisschen mehr können ...

7. Die Objektinspektion erlaubt einfaches Debbuging, indem sie den internen Zustand eines Objektes zeigt.
8. Ein Objekt kann als Parameter bei einem Methodenaufruf übergeben werden, indem man auf das Objekticon klickt.

Erstellen eines neuen Projektes

9. Um ein Projekt zu erstellen, wählen Sie *Neues Projekt* aus dem *Projekt* Menü.
10. Um eine neue Klasse zu erstellen, klicken Sie auf den Button *Neue Klasse...* und geben einen Namen für die Klasse ein.
11. Um einen Pfeil zu erstellen, klicken Sie auf die Pfeil-Button und ziehen Sie den Pfeil im Diagramm oder schreiben Sie den Quelltext in den Editor.
12. Um eine Klasse oder einen Pfeil zu entfernen, klicken sie auf die Löschfunktion aus dessen Kontextmenü.

Benutzen der Direkteingabe

13. Um die Direkteingabe zu benutzen, wählen Sie *Direkteingabe anzeigen* aus dem Menü *Ansicht*.
14. Um Java-Ausdrücke auszuwerten, geben Sie sie einfach in die Direkteingabe ein.
15. Um Objekte von der Direkteingabe auf die Objektleiste zu verschieben, ziehen Sie das kleine Objekticon mit der Maus vom Direkteingabe auf die Objektleiste.
16. Um die Ergebnisobjekte der Direkteingabe zu inspizieren, klicken Sie doppelt auf das kleine Objekticon.

17. Anweisungen, die in die Direkteingabe eingegeben werden, werden ausgeführt.
18. Benutzen Sie *Shift-Enter* am Ende einer Zeile, um eine mehrzeilige Anweisung einzugeben.
19. Lokale Variablen können in einzelnen und mehrzeiligen Anweisungen benutzt werden. Die Namen der Objekte auf der Objektbank dienen als Instanzfelder.
20. Benutzen Sie *die Pfeil-Auf-* und *Pfeil-Ab*-Tasten, um die Eingabe-History zu verwenden.

Debugging

21. Um einen Haltepunkt zu setzen, klicken Sie in die Haltepunktfläche links neben dem Text im Editor.
22. Um einen Haltepunkt zu setzen, klicken Sie in die Haltepunktfläche links neben dem Text im Editor.
23. Um Schritt für Schritt durch den Code zu gehen, benutzen Sie die „Schritt"- und „Schritt hinein"-Buttons des Debbuggers.
24. Das Inspizieren von Variablen ist einfach – sie werden automatisch im Debugger angezeigt.
25. *Anhalten* und *Beenden* können benutzt werden, um eine Durchführung temporär oder permanent zu stoppen.

Erstellen von Einzelanwendungen

26. Um eine Einzelanwendung (die sich auch ohne BlueJ starten lässt) zu erstellen, benutzen Sie *Projekt – Als jar-Archiv speichern ...*

Applets erstellen

27. Um eine Applet zu starten, wählen Sie *Applet ausführen* vom Popupmenü des Applets.
28. Um ein Applet zu erstellen, klicken Sie den Button *Neue Klasse ...* und wählen *Applet* als Klassentyp.

Andere Anwendungsmöglichkeiten

29. Nicht-BlueJ Paketen laden Sie mit *Fremdprojekt öffnen ...* im Menü *Projekt.*
30. Fremde Klassen können in ein Projekt kopiert werden, indem man den Befehl *Klasse aus Datei hinzufügen ...* benutzt.
31. Statische Methoden können vom Popupmenü der Klasse aufgerufen werden.
32. Um eine Dokumentation für ein Projekt zu generieren, wählen Sie *Dokumentation erzeugen* aus dem Menü *Werkzeuge*.
33. Die Java Standard API kann angezeigt werden, wenn man die im Menü *Hilfe* den Punkt *Java Klassenbibliotheken* auswählt.

