BlueJのチュートリアル

バージョン 2.0.1

BlueJ バージョン 2.0.x 用

Michael Kölling
Mærsk Institute
University of Southern Denmark

桐蔭横浜大学・工学部・電子情報工学科
Toin University of Yokohama
アルベルト・パラシオス・パウロブスキ
Alberto Palacios Pawlovsky　教授　訳

平成 17 年 4 月 4 日

目 次

第1章　序文

1.1　BlueJ について

このチュートリアルは、BlueJ プログラミング環境を使うための入門書です。BlueJ は、入門レベルで教育を行うために設計された $Java^{TM}$ 開発環境です。オーストラリア・メルボルンの Deakin 大学と英国・カンタベリーの Kent 大学の BlueJ チームによって設計され、開発されました。BlueJ に関する情報は http://www.bluej.org/にあります。

1.2　範囲と対象読者

このチュートリアルは、この開発環境の使用に慣れたい人を対象にしています。環境の設計に関する決定または関連の研究についての説明ではありません。

また、このチュートリアルは Java を教育するためのものでもありません。Java の初心者は Java の入門用の教科書を勉強して、あるいは入門のコースを受講するのをお勧めします。

さらに、数多くの詳細が省略されているので、環境の詳しリファレンス・マニュアルでもありません。簡単で分かりやすい入門に重点が置かれています。より詳細な参考書が必要な場合は、BlueJ のウエブ・サイト（www.bluej.org）にある The BlueJ Environment Reference Manual を参照して下さい。

ほとんどの節は、一行の要約文で始まります。これを参考に、既にシステムを知っている人が該当の節を読むか否かを判断するために使えます。第 11 章は、クイック・リファレンス形式でこれらの要約文を再掲しています。

1.3　著作権、ライセンシング、再配布

BlueJ システムおよびこのチュートリアルはすべての人に、使用および再配布の何れの用途に対してもフリー（無料）です。システムの分割は禁じられています。

著者らの許可無く BlueJ およびその資料、もしくはその一部分を営利目的で販売したり、商用のパッケージに添付したりすることができません。

M. Kölling および J. Rosenberg が BlueJ の著作権を所有します。

1.4　フィードバック

BlueJ のシステムに関するコメント、質問、訂正、批判および他のいかなるフィドバックを歓迎致します。どうぞ Michel Kölling（mik@mip.sdu.dk）までメールを下さい。

第2章　インストール

BlueJ は三つの異なる形式で配布されています。一つは Windows 用のもです。もう一つは MacOS 用のものです。最後のはその他のシステム用のものです。インストールは非常に簡単です。

インストールの前提条件：
BlueJ を使うには、自分のシステムに J2SE バージョン 1.4（通称：JDK1.4）以降がインストールされていなければなりません。通常、最新（ベータ版以外の）版へのアップデートをお勧めします。もし、JDK がインストールされていない場合は、http://java.sun.com/j2se/の Sun の Web サイトからダウンロードできます。MacOS ではいつも最新版の JDK が既にインストールされているので自分でインストールする必要はありません。もし、ダウンロード時に JRE（Java Runtime Environment）および SDK（Software Development Kit）を提供するダウンロード・ページに至った場合は、SDK をダウンロードして下さい（JRE だけは十分ではありません）。

2.1　Windows 上のインストール

Windows 用のインストール・ファイルは bluejsetup-xxx.exe となっています（xxx はバージョンの番号を表しています）。たとえば、BlueJ の 2.0.0 のバージョンは bluejsetup-200.exe です。このファイルをディスクで取得することか、あるいは BlueJ の http://www.blue.org/の Web サイトからダウンロードすることができます。そのインストーラーを実行して下さい。
インストーラーがインストール先のディレクトリの指定を要求します。また、デスクトップもしくはスタート・メニューに本ソフトへのショート・カットの作成も提供します。
インストール後、インストールで指定したディレクトリ内に bluej.exe が現れます。
BlueJ が最初の起動では Java システム（JDK）を探します。一つ以上の使用可能な Java システム（例：JDK1.4.2 と JDK1.5.0）を見つけた場合は、一つを選択するためのダイアログを表示します。Java システムを一つも見つけない場合は、手動でそのシステムを指定しなければなりません（これは JDK があるがその登録用のエントリーが削除された場合に起こります）。
BlueJ のインストーラが vmselect.exe のプログラムもインストールします。このプログラムを使えば、BlueJ の使用する Java システムが変更できます。現行のと異なる Java バージョンを使いたいときはこのプログラムを実行して下さい。
選択された Java バージョンの情報が BlueJ に格納されます。異なるバージョンの BlueJ をインストールされていれば、一つで JDK1.4.2 を使って、もう一つで JDK1.5 を使用することができます。一つの BlueJ のバージョンの Java システムを変えると同じユーザの同バージョンのすべての BlueJ の Java システムが変更されます。

2.2　Macintosh 上のインストール

注意：BlueJ は MacOS X でしか使用できません。

MacOS の配布ファイルは BlueJ-xxx.zip です。xxx は BlueJ のバージョン番号です。たとえば、BlueJ の 2..0.0 のバージョンは BlueJ-200.zip です。このファイルをディスクで取得することか、あるいは BlueJ の http://www.blue.org/の Web サイトからダウンロードすることができます。通常、MacOS がこのファイルを自動的に解凍します。解凍されない場合は、このファイルをダブル・クリックで解凍します。

解凍後、解凍した位置に新たな BlueJ.xxx というフォルダが現れます。このフォルダをアプリケーション・フォルダー（あるいは好きなところ）に移動して下さい。これでインストールが終わりです。

2.3　Linux/Unix、他のシステム上でのインストール

汎用配布ファイルは実行可能な jar ファイルです。このファイルは bluej-xxx.jar となっています。xxx はバージョンの番号です。たとえば、BlueJ の 2.0.0 のバージョンは bluej-200.jar です。このファイルをディスクで取得することか、あるいは BlueJ の http://www.blue.org/の Web サイトからダウンロードすることができます。

インストーラーを以下のコマンドで実行して下さい。注意：この例では、配布ファイル bluej-200.jar を使用します。このコマンドのファイル名を取得したファイル名（正しいバージョン番号のファイル名）に置き換える必要があります。

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

ここでは、$< j2se - path >$ は、J2SE JDK がインストールされているディレクトリです。上記のコマンドでウィンドウが現れ、そこで BlueJ のインストール先のディレクトリと、BlueJ の実行に使用する JDK バージョンを選択して、Install ボタンをクリックします。インストールの終了後、BlueJ がインストールされているはずです。

2.4　インストールでの問題

何か問題がありましたら、BlueJ の Web サイト（http://www.bluej.org/help/faq.html）で FAQ（よくある質問）をチェックして、および How To Ask For Help（助けの求め方）の項目（http://www.bluej.org/help/ask-help.html）を読んで下さい。

第3章　基本的な操作：編集、コンパイル、実行

3.1　BlueJ の起動

Windows および MacOS では BlueJ というプログラムがインストールされるので、これを実行して下さい。Unix ではインストーラーが bluej というスクリプトをインストール・ディレクトリにインストールします。GUI インタフェースでそのファイルをダブル・クリックして下さい。コマンド・ラインでは、次のように対象プロジェクトがなくても、これを指定しても BlueJ が起動できます。

```
$ bluej
```

または

```
$bluej examples/people
```

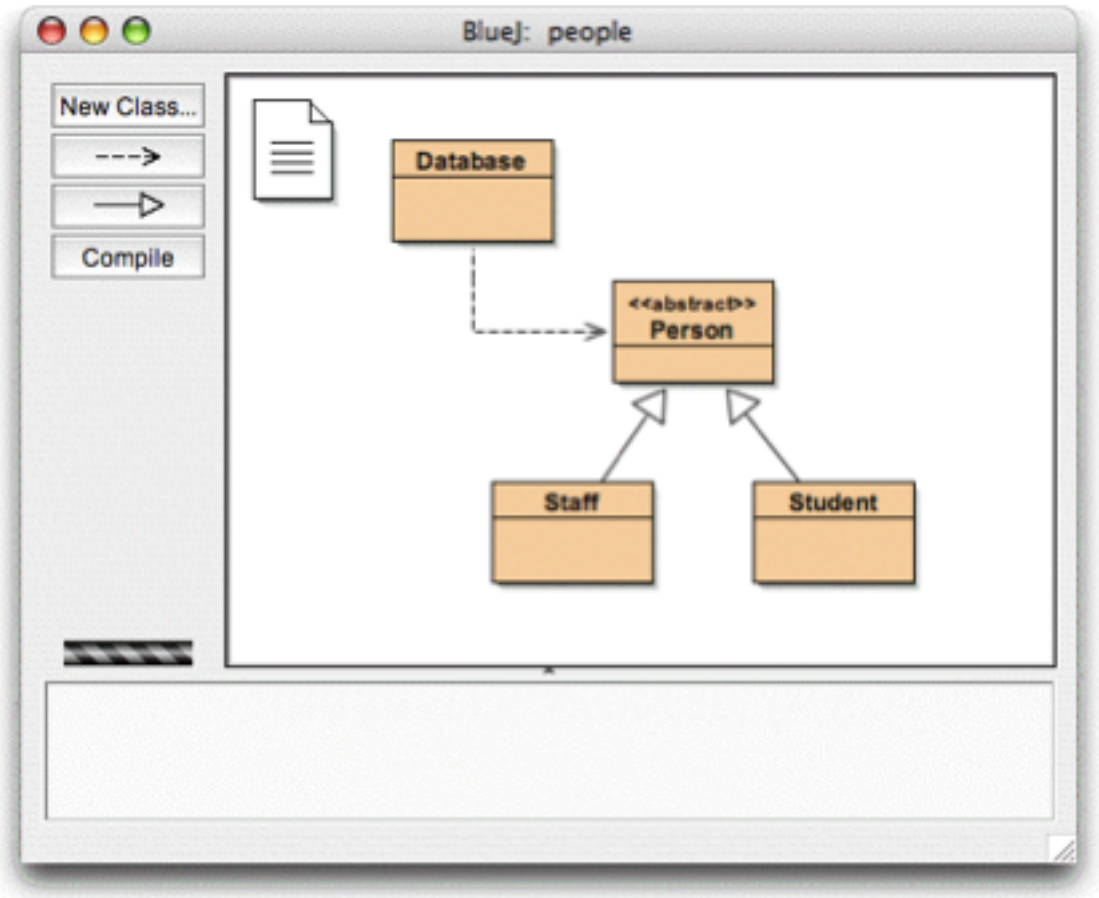

図 3.1: BlueJ のメイン・ウインド

3.2　プロジェクトの開き方

要約： プロジェクトを開くためには、Project メニューから Open を選択する。

BlueJ のプロジェクトは、標準の Java パッケージと同様に、プロジェクトのファイルすべてを含んでいます。

BlueJ が起動しているとき、 Project メニューの Open... の項目でプロジェクトを選択して開きます。

標準の BlueJ 配布では、examples ディレクトリに幾つかの例題プロジェクトがあります。

このチュートリアルの節のために、そのディレクトリにある people のプロジェクトを開いて下さい。examples ディレクトリは BlueJ のホーム・ディレクトリにあります。プロジェクトを開いた後図 3.1 に示すような画面が表示されます。システムによってウインドウ表示が多少異なります。

3.3　オブジェクトを生成する方法

要約： オブジェクトを生成するためには、クラスのポップ・アップ・メニューからコンストラクターを一つ選択する。

完全なアプリケーションが実行できるだけではなく BlueJ の基本的な特長の一つは、何れのクラスのオブジェクトとやりとりができ、それらのメソッドの実行もできます。BlueJ での実行は、通常、オブジェクトを生成して、オブジェクトのメソッドの呼び出しで行われます。この特徴は、アプリケーションの開発過程にとても役に立ちます。クラスのできあがった時点で一つずつテストができます。アプリケーションすべてを書く必要はありません。

サイド・ノート：オブジェクトを生成しなくても静的なメソッドが実行できます。main はその一つのメソッドであるので、通常の Java アプリケーションでできる main からの実行が可能です。これについては 後述で再びふれます。ここではまず、通常の Java 環境でできないことについて見ていきます。

メイン・ウインドウの中央部に見える四角形（Database、Person、Staff、および Student）のものは、このアプリケーションのクラスのアイコンです。クラス・アイコンを右クリック（Macintosh では control+クリック[1]）すると、そのクラスに適用できる操作のメニューが表示されます（図 3.2）。表示される操作項目は、一部分はクラスの記述にあるコンストラクタに関連の new 操作です。同メニューの下部には、開発環境で提供されている操作項目も幾つかあります。

Staff クラスのオブジェクトを一つ生成したいので、Staff クラスのアイコンを右クリックします（これは図 3.2 のメニューをポップ・アップさせます）。このメニューは、Staff オブジェクトを生成するために、パラメーターのあるものとないものとの二つのコンストラクタを表示します。ここではまず、パラメーターのないコンストラクタを選んで下さい。図 3.3 に示すダイアログが現われます。

このダイアログは、生成されるオブジェクトの名前を要求します。これと同時に、デフォルトの名前（staff1）も提示されます。そのデフォルト名はよいので、そのまま Ok ボタンをクリックして下さい。これで Staff オブジェクトが一つ生成されます。

生成されたオブジェクトがオブジェクト・ベンチ（図 3.4）に置かれます。オブジェクトの生成に関してはこれは全部です。つまり、オブジェクト生成を行うにはクラス・メニューからコンス

[1] MacOS ユーザー：右クリックの記述を control+クリックに読み直して下さい。

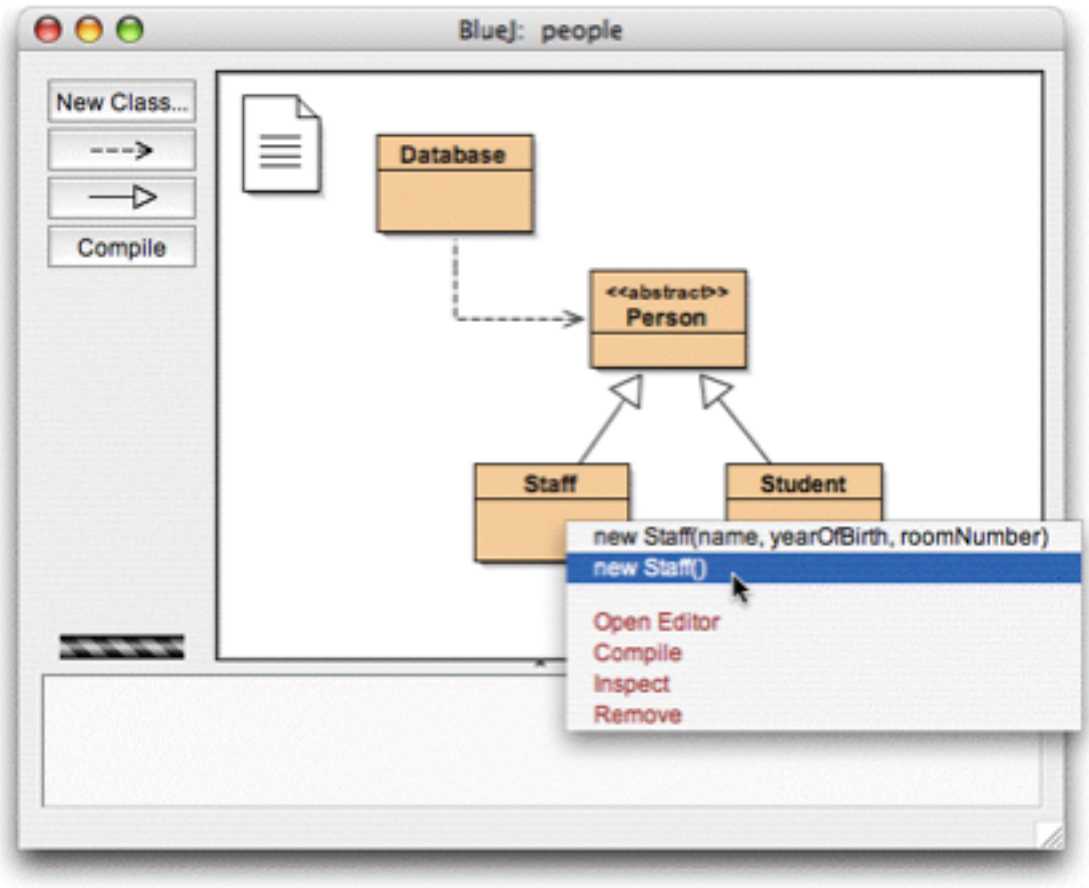

図 3.2: クラスの操作（用ポップ・アップ・メニュー）

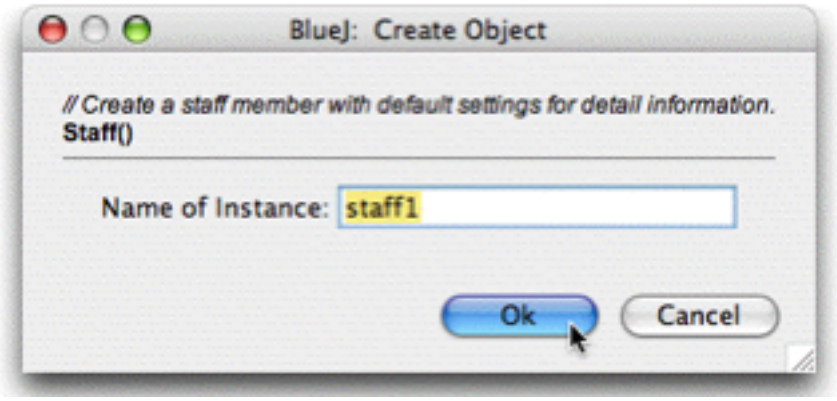

図 3.3: パラメーターのないオブジェクトの生成

トラクターを選択し、実行します。これで該当のオブジェクトがオブジェクト・ベンチに置かれます。

Person クラスのラベルは << *abstract* >> であることに気づいたでしょう（抽象クラスです）。抽象クラスのオブジェクトが生成できないことを試してもらえば確認できます（これは Java 言語仕様の定義による制限です）。

図 3.4: オブジェクト・ベンチ内のオブジェクト

3.4　実行

要約： メソッドを実行するためには、そのメソッドをオブジェクトのポップ・アップ・メニューから選択する。

オブジェクトを生成したので、その公衆（public の）の操作を実行することができます（Java ではこれらの操作をメソッドという）。オブジェクトを右クリックすると、オブジェクト用の操作メニューがポップ・アップします（図 3.5）。このメニューは、オブジェクト用使用可能なメソッドと、開発環境が提供する特別な二つの操作 （Inspect と Remove） を示しています。後者についてはあとで述べます。ここではまず、メソッドに集中しましょう。

図 3.5: オブジェクト・メニュー

この Staff メンバーの部屋番号を設定するメソッド setRoom と部屋番号を返すメソッド getRoom がメニューにあるとわかります。getRoom を呼び出してみましょう。このメソッドをオブジェクト・メニューから選ぶと、実行されます。そして呼び出しの結果を示すダイアログが表示されます（図 3.6）。この場合は、この人の部屋を指定しなかったので、「”(unknown room)”」が表示されます。

スーパー・クラスから継承されたメソッドは、上記メニューのサブメニューから使用できます。本オブジェクトのポップ・アップメニューの先頭に、二つのサブメニューがあります。一つは Object から継承したもの用で、もう一つは Person から継承したもの用のサブメニューです（図 3.5）。サブメニューから Person のメソッド（getName 等）が呼べます。試してみて下さい。この場合も、人の名前を設定しなかったので、「”(unknown name)”」の不明な結果しか得られません。

今度、部屋番号の指定を試みましょう。これはパラメターのある呼び出しを例示します（getRoom と getName の呼び出しは、戻り値はあるが、パラメーターはありません）。メニューから setRoom を呼び出して下さい。パラメーターの入力を要求するダイアログが表示されます（図 3.7）。

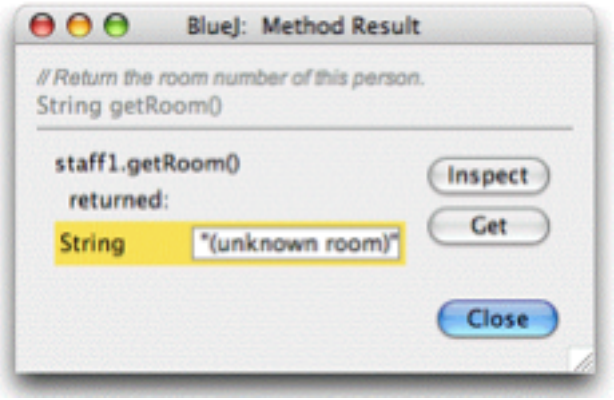

図 3.6: メソッドの実行結果

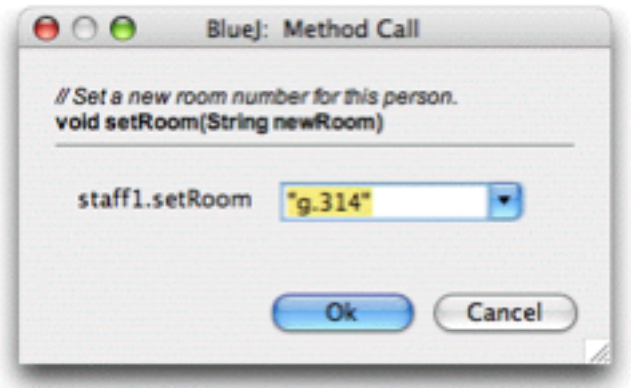

図 3.7: パラメターのあるメソッドの呼び出し用ダイアログ

ダイアログの上部に呼び出されているメソッド（コメントおよびシグネチャを含めて）のインターフェースが表示されます。その下にパラメーターを入力するための欄があります。上部のシグネチャは、String 型のパラメーターの記入が期待されていることを表しています。部屋番号（ダブル・クオテーション「 ”」を含めて）をテキスト欄に入力し、Ok をクリックして下さい。

これで全部です。このメソッドはパラメーターを返さないので、結果用のダイアログはありません。再び getRoom を呼び出して、部屋番号が変わったかをチェックして下さい。

オブジェクトの生成やメソッドの呼び出し等を色々試してみて下さい。このような操作に慣れるためにたとえば、引数のあるコンストラクタを呼び出したり、その他の幾つかのメソッドを呼び出したりして下さい。

3.5 クラスの編集方法

要約： クラスのソースを編集するためには、クラスのアイコンをダブル・クリックする

これまでは、オブジェクトのインタフェースだけをみてきました。これからはオブジェクトの中を見ていきます。クラスの操作メニューから Open Editor を選択すると、クラスのインプリメンテーション（ソース・ファイル）を見ることができます（催促：クラスのアイコンを右クリックするとクラスの操作メニューが現れる）。クラスのアイコンをダブル・クリックすることで、同じ操作ができます。エディターは、このチュートリアルで詳細に説明されていないが、使い方は簡単です。エディターの詳細は、別の資料で説明する予定です。今は、Staff クラスのインプリメンテー

ションを開いて下さい。そして、getRoom メソッドのインプリメンテーション（記述）を見つけて下さい。これは、その名前通り、スタッフのメンバーの部屋番号を返します。現行の”M.3.18”だけではなく、”room M.3.18”を返すようにメソッドの return 文の先頭に”room ”を加えて変更しましょう。これは

```
return room;
```

の行を、

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

に変更することで実現できます。BlueJ は変更無しの全 Java をサポートしているので、クラスをインプリメント（作成）するときは、特別な作業・配慮は一切不要です。

3.6　コンパイルの仕方

要約： クラスをコンパイルするためには、エディターの Compile ボタンをクリックする。プロジェクトをコンパイルするためには、プロジェクト・ウインドの Compile ボタンをクリックする。

テキストを記入したあと（しかもその他のことをやる前に)、プロジェクト概略（メイン・ウインド）をチェックして下さい。Staff クラスのアイコンが変わったでしょう。斜線がかかっています。これは、クラスの最後の変更後、クラスがコンパイルされていないことを表しています。エディターに戻りましょう。

サイド・ノート：プロジェクトを開いたときに、なぜクラスのアイコンは斜線がかかっていなかったことを疑問に思うでしょう。これは、配布された例題はコンパイル済みであるからです。ほとんどの BlueJ プロジェクトは未コンパイル状態で配布されるので、それらのプロジェクトを開くときはクラスの数多くのアイコンが斜線のかかっているものになるでしょう。

エディターの上部によく使用する機能用のボタンのツール・バーがあります。その一つは Compile のボタンです。この機能はエディターから直接、クラスをコンパイルすることを可能にします。今は、Compile ボタンをクリックして下さい。ミスがなければ、クラスがコンパイルされたことを通知するメッセージが下部のインフォメーション領域に現われます。構文エラーを起こすミスがあると、エラーの行が強調され、エラー・メッセージがインフォメーション領域に表示されます (一回目のコンパイルで成功した場合は、この機能を確かめるために、例えば一つのセミコロンを削除して、エラーを起こさせましょう)。

クラスのコンパイルが成功したら、エディターを閉じて下さい。

サイド・ノート：クラスのソースを自分で保存する必要はない。ソースは適切なときに自動的に保存されます（例えば、エディターを終了させる時に、およびコンパイルする前に)。もちろん、自分でも保存ができます（保存用の機能はエディターのクラス・メニューにあります)。しかし、これはシステ ムが非常に不安定でクラッシュが頻繁に起こるとき、および作業内容が失われる心配があるときのみ必要です。

プロジェクト・ウィンドウのツール・バーにも Compile ボタンがあります。このボタンでプロジェクト全体がコンパイルされます（実は、どのクラスの再コンパイルが必要かが判断され、クラ

スが正しい順序で再コンパイルされます)。これを試すために、二つ以上のクラスを変更して（該当のクラス・アイコンは斜線付きのものになります)、そして Compile ボタンをクリックして下さい。コンパイルしたクラスにエラーが発見されれば、エディターが開かれ、エラーの位置とエラー・メッセージが表示されます。

オブジェクト・ベンチが空けられたことに気が付いたでしょう。インプリメンテーションが変更される度に、該当のオブジェクトは削除されます。

3.7 コンパイル・エラー用のヘルプ

要約： コンパイル・エラー・メッセージ用のヘルプを得るためには、エラー・メッセージの右にある「？」のボタンをクリックする。

初心者の学生は、しばしばコンパイル・エラーのメッセージの意味が理解できなくて悩みます。このため、手助けとなる機能を提供しています。

再びエディターを開いて、ソース・ファイルにエラーを入れて再びコンパイルして下さい。エディターのインフォメーション領域にエラー・メッセージが表示されるはずです。インフォメーション領域の右にある「？」をクリックすると、そのエラーのタイプに関する情報をもう少し取得できます（図 3.8)。

図 3.8: コンパイラー・エラーとヘルプ・ボタン

現在は、すべてのエラー・メッセージに対するヘルプ文書ははありません。まだ幾つかのヘルプ文書が書かれなければなりません。しかし、多くのエラーが既に説明されているので、ヘルプを試みる価値があります。いつか残りのヘルプ文書は執筆され、今後の BlueJ のリリースに含まれるでしょう。

第4章　もう少しやってみよう

この章では、BlueJ の環境でできることについてもう少し述べます。説明することは必要不可欠ではないが、よく使われています。

4.1　検閲

要約： オブジェクトの検閲はオブジェクトの内部状態を示すため、簡単なデバッグを可能にする。

オブジェクトのメソッドを実行したとき、ユーザーが定義したメソッド（図 3.5）の他に検閲（Inspect）機能が利用できると気が付いたでしょう。この操作は、オブジェクトのインスタンス変数（フィールド）の状態チェックを可能にします。ユーザーの定義した幾つかの値を持つオブジェクトを生成してみて下さい（例：パラメーターを取るコンストラクタの Staff オブジェクト）。そして、そのオブジェクトのメニューから Inspect を選んで下さい。オブジェクトのフィールド、そのタイプ、およびその値を表示するダイアログが現われます（図 4.1）。

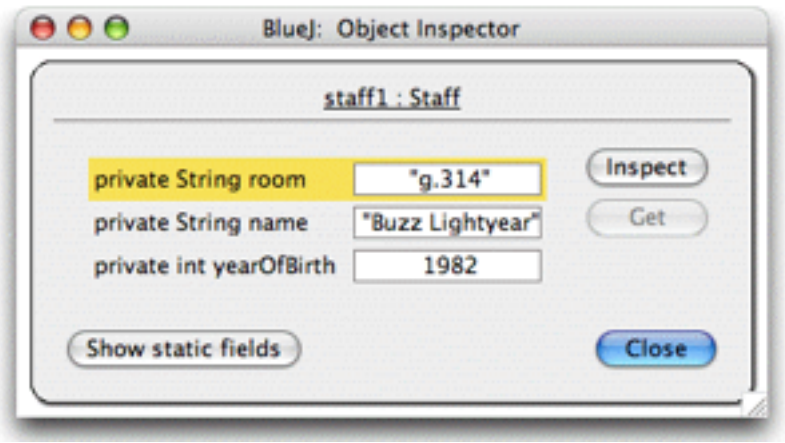

図 4.1: 検閲のダイアログ

ミュテーター（変換）操作（オブジェクトの状態を変更する操作）が正しく実行されたかどうかを迅速にチェックするときには、検閲機能が有用です。このため、検閲は簡単なデバッグ・ツールです。

Staff の例では、そのすべてのフィールドがシンプルなタイプです（非オブジェクトか String 型です）。そのようなタイプの値は直接に表示できます。この機能の使用でコンストラクタが値を正しく割り当てたかどうかをすぐに見ることができます。

複雑なケースでは、フィールドの値はユーザーの定義したオブジェクトへの参照になっているかもしれません。そのような例を見るために、もう一つのプロジェクトを使います。標準 BlueJ 配布に含まれている people2 のプロジェクトを開いて下さい。 people2 のメイン・ウインドが図 4.2 に示されています。ご覧のように、この 2 番目の例には今まで見ていたクラスの他に Address クラスがあります。また、Person クラスの一つのフィールドはユーザーが定義した Address のタイプです。

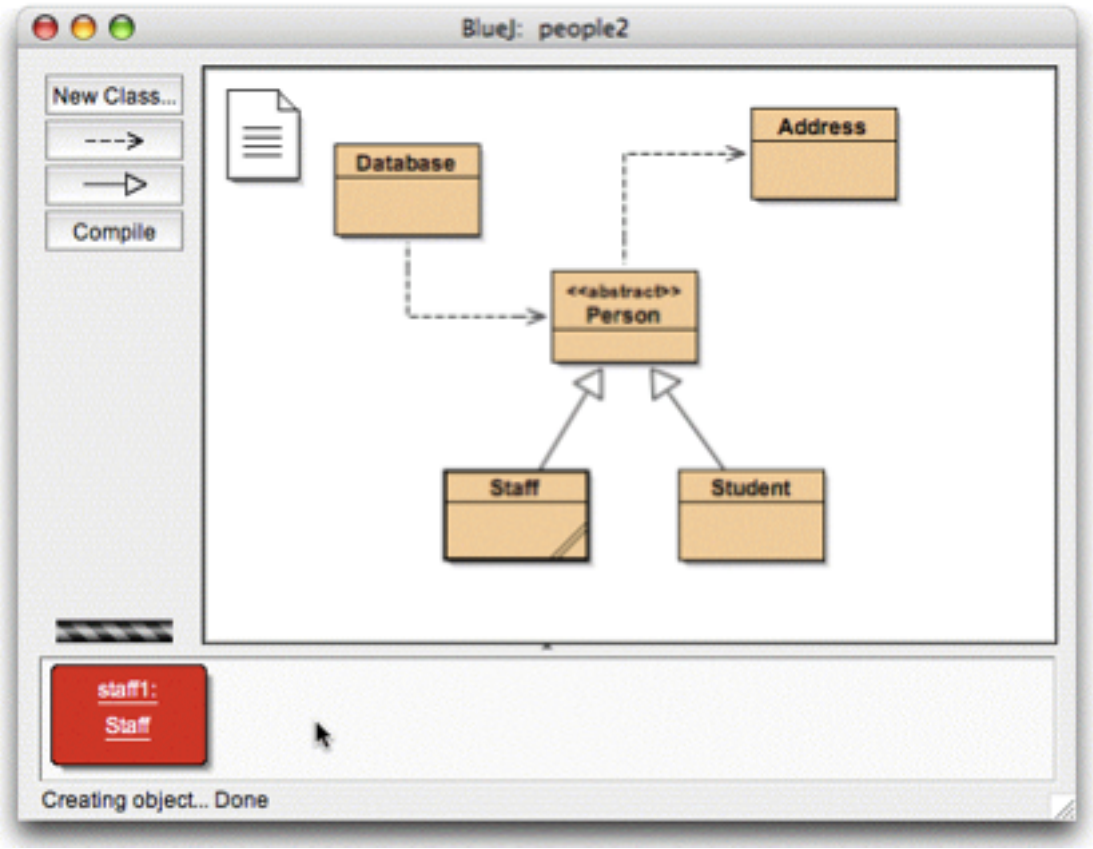

図 4.2: people2 のプロジェクト・ウインド

次に試したいこと（オブジェクト・フィールドの検閲）のために、 Staff オブジェクトを作成して、そして setAddress を呼び出して下さい（これが Person のサブメニューにあります)。 住所を入力して下さい。内部では、Staff のコードは Address クラスのオブジェクトを生成し、入力された住所を address フィールドに格納します。

今度は、Staff オブジェクトを検閲して下さい。得られる検閲用ダイアログは図 4.3 に示されています。今度の Staff オブジェクトのフィールドは address を含んでいます。ご覧のように、address

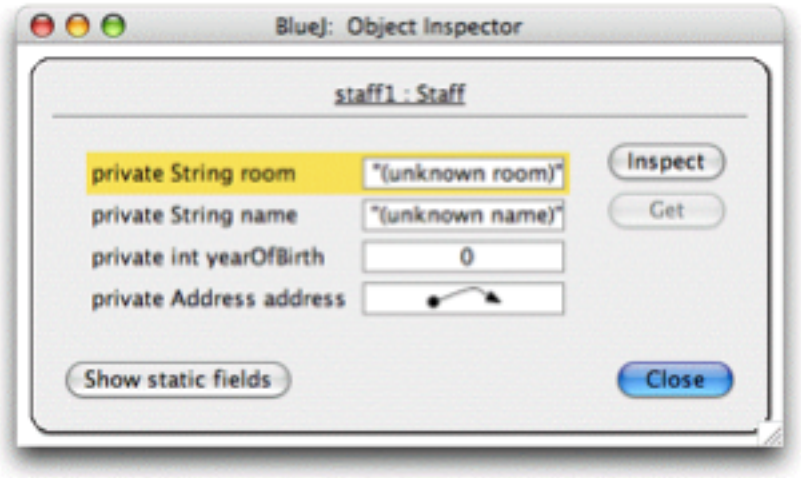

図 4.3: オブジェクト参照の検閲

の値は矢印で示されています。これは他オブジェクトへの参照を意味します。address は複雑で

ユーザー定義型のオブジェクトのため、その値は直接リストに表示できません。address をさらに調べるために、リストの address フィールドを選択して、ダイアログの Inspect ボタンをクリックして下さい（address フィールドのダブル・クリックでもこの操作が行えます）。これに対して、もう一つの検閲ウインドウが開かれ、そのウインドが Address オブジェクトの詳細を表示します（図 4.4）。

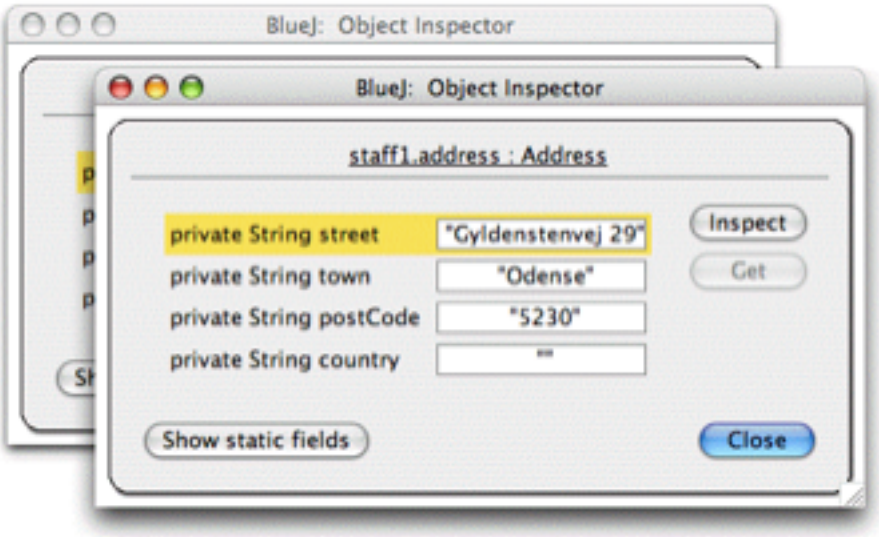

図 4.4: 内部オブジェクトの検閲

選択したフィールドが公衆（public）であれば、Inspect ボタンをクリックしないで、address を選択して、Get ボタンをクリックします。この操作は、選択したオブジェクトの一つをオブジェクト・ベンチに置きます。そのオブジェクトのメソッドの呼び出しで、さらに詳細な検査ができます。

4.2 オブジェクトをパラメーターとしての渡し方

要約： オブジェクトのアイコンをクリックすることによって、そのオブジェクトをメソッドにパラメーターとして渡すことができる。

オブジェクトを、他のオブジェクトのメソッドにパラメーターとして引き渡すことができます。一つの例を試してみましょう。Database クラスのオブジェクトを一つ生成して下さい（Database クラスは、パラメーターのないコンストラクタが一つしかもっていないので、オブジェクトの生成は簡単です）。Database オブジェクトは、Person オブジェクトのリストを保持することができます。また、Person オブジェクトの追加、および現在保持しているすべての Person オブジェクトを表示するメソッドを持っています（実際はデータ・ベースの名前は少々誇張的です）。

Staff オブジェクト、または Student オブジェクトをまだオブジェクト・ベンチになければ、その何れの一つを生成して下さい。以降の説明には、オブジェクト・ベンチに Database オブジェクトの一個と、Staff オブジェクト、または Student オブジェクトの一個が必要です。

今度は、Database オブジェクトの addPerson メソッドを呼び出して下さい。そのシグネチャは、Person 型のパラメーターが必要であることを指定します（覚えておいて下さい。Person クラスが抽象型なので、Person タイプのオブジェクトはありません。しかし、サブタイピングで Student および Staff オブジェクトは Person オブジェクトの代わりに使えます。したがって、Person の期待されているとき、代わりに Student か Staff を引き渡すことが合法的です）。 オブジェクト・ベンチにあるオブジェクトを、メソッド呼び出しにパラメーターとして引き渡すには、パラメーター

の記入欄にオブジェクト名を記入するか、近道として該当オブジェクトをクリックします。これは、メソッド呼び出しダイアログの該当欄にオブジェクト名を挿入します。Ok をクリックすれば呼び出しが実行されます。 このメソッドに戻り値がないため、すぐには結果が見れません。操作が本当に実行されたかどうかをチェックするために、Database オブジェクトの listAll メソッドを呼び出すことができます。listAll 操作は、Person に関する情報を標準出力に書き出します。テキストを表示するために、自動的にターミナル画面が開いてきます。

一人以上のデータベースでこれらの操作を自分で再び試してみて下さい。

第5章　新プロジェクトの作成方法

この章では、新しいプロジェクトを設定するためのクイック・ツアーです。

5.1　プロジェクト・ディレクトリの作成方法

要約： 新しいプロジェクトを作成するためには、Project メニューから New... を選択する。

新しいプロジェクトを作成するには、Project メニューから New... を選択します。新規プロジェクトの場所と名前の指定を可能にするファイル選択ダイアログが開かれます。今、これを試してみて下さい。プロジェクト名の選択は自由です。Ok をクリックしたあと、指定した名前でディレクトリが作られ、メイン・ウインドウは、新規の空プロジェクトを表示します。

5.2　クラスの作成方法

要約： クラスを作成するためには、New Class... ボタンをクリックして、クラス名を指定する。

今度は、プロジェクトのツール・バーにある New Class... ボタンをクリックすれば、自分のクラスを作ることができます。そのとき、新規クラスの名前は要求されます（この名前は Java における有効な名前でなければなりません）。

abstract、interface、applet、および standard の四種類のクラスの中から選択することができます。この選択により、クラスに生成される初期骨組みコードが決定されます。クラスのソース・コードを変えれば、クラスのタイプを変えることができます （例：コードに abstract キーワードを記入する）。

クラスを作成すると、メイン・ウインドウにアイコンとして現れます。標準クラスではない場合は、そのタイプ（abstract、interface、または applet）がクラスのアイコンに表示されます。新規作成したクラスをエディターで開くと、デフォルトのクラスの骨組みが生成されたことが分かります。これは作業開始を容易にするためです。デフォルトのコードは文法的に正しいであり、コンパイルもできます（しかし、大したことをしません）。幾つかのクラスを作成して、コンパイルしてみて下さい。

5.3　依存関係の作成方法

要約： 矢印を生成するためには、矢印のボタンをクリックしてプロジェクト・ウインドウ上に矢印をドラッグするか、またはソース・コードをエディターで書く。

クラスの図表はクラス間の依存関係を矢印で表示します。継承関係（extends、または implements）は点線の矢印で、使用関係（uses）は実線の矢印で表されます。

（直接図表上に）図形を用いるか、あるいはソース・コードの入力で依存関係を追加することができます。図形で矢印を付けると、ソースは自動的にアップデートされます。また、ソースで依存関係を追加すると、図表は更新されます。

図形で矢印を追加するために、該当の矢印ボタンをクリックして（extends、または implements の場合は点線の矢印、uses の場合は実線の矢印)、一つのクラスからもう一方のクラスへ矢印をドラッグします。

継承の矢印を追加すると、extends か、または implements の定義はクラスのソース・コードに追加されます（それぞれ、クラスか、またはインタフェースに追加されます)。

使用関係矢印の追加は、ソースをすぐには変えません（対象が別のパッケージのクラスである場合は、直ぐに import 文が生成されます。ここまでの例題プロジェクトではそのようなケースをまだ見ていません)。実際に使わないクラスを uses 矢印で指すと、そのクラスが使用されていないという警告を発生させます。

コーディングで矢印を付けるのは簡単です。コードを普通に記入するだけです。クラスが保存されると、クラス図表が更新されます（エディターを閉じると自動的に保存が行われることを覚えておきましょう)。

5.4　要素の削除方法

要約： クラスか矢印を削除するためには、そのポップ・アップ・メニューから Remove を選択する。

図表からクラスを削除するには、そのクラスを選択してから Edit メニューから Remove を選択します。また、そのクラスのポップ・アップ・メニューから Remove も選べます。これらの方法は矢印の削除にも使えます。つまり、矢印を選択してからメニューで Remove が選択できるし、矢印のポップ・アップ・メニューが使用できます。

第6章　コード・パッドの使用方法

BlueJ のコード・パッドは迅速にしかも簡単に短い Java コード（式や文）の試しを可能にします。このため、コード・パッドは Java 構文の詳細の調べか構文の例示と試しに使用できます。

6.1　コード・パッドの表示方法

要約： コード・パッドの使用を開始するためには、View メニューから Show Code Pad を選択する。

コード・パッドはデフォルトで表示されません。表示させるために View メニューから Show Code Pad を選択して下さい。すると、メイン・ウインドはその右下にオブジェクト・ベンチの隣にコード・パッドのインタフェースを示します（図 6.1）。該当領域の幅と高さの変更でコード・パッドとオブジェクト・ベンチのサイズは変えられます。

今度からは、コード・パッドが式や文の記入に使用できます。Enter キーの押すことによって該当の行は評価され、その結果が表示されます。

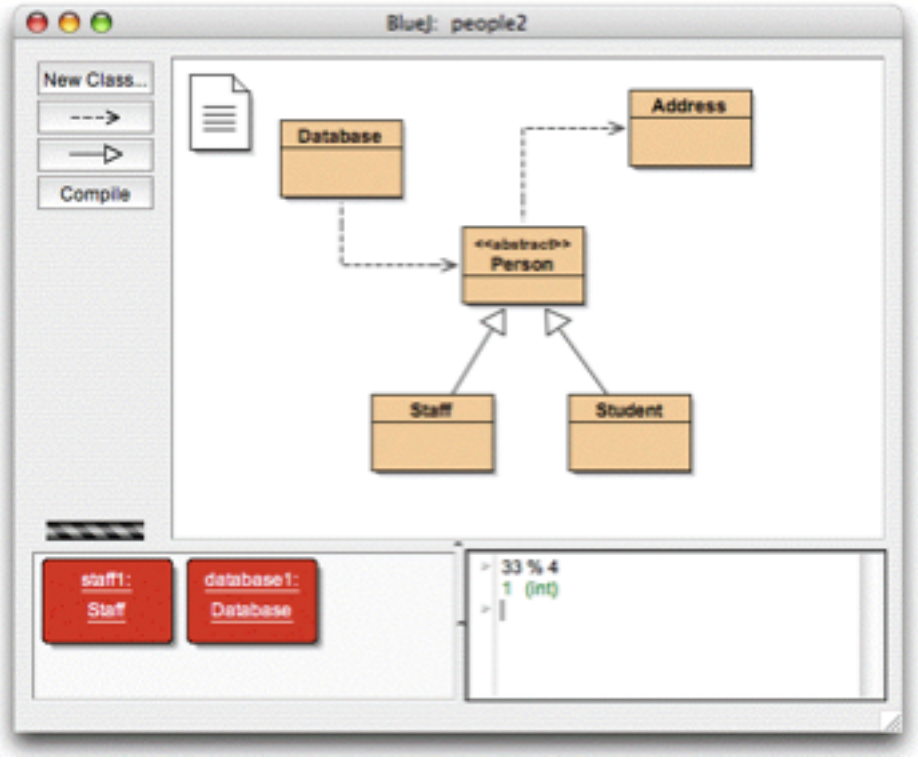

図 6.1: コード・パッドを示すメイン・ウインド

6.2　簡単な式の評価方法

要約： Java の式を評価するためには、これらをコード・パッドに記入する。

コード・パッドは簡単な式の評価には使用できます。例えば、次の式を記入してみて下さい。

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

式は Java 標準の値やオブジェクト、および現行プロジェクトのクラスに参照できます。コード・パッドは結果の値とそのタイプ（括弧で囲まれたもの)、あるいは式が不正のときエラー・メッセージを表示します。

また、オブジェクト・ベンチにあるオブジェクトも使用できます。次のことを試して下さい。オブジェクト・ベンチに student クラスのオブジェクトを置いて下さい。その名を student1 にして下さい。

そして、今度コード・パッドに次の文が記入できます。

```
student1.getName()
```

同様に、自分のプロジェクトのクラスの全メソッドを参照することができます。

6.3　オブジェクトの受け取り方

要約： コード・パッドのオブジェクトをオブジェクト・ベンチに転送するためには、オブジェクトの小さなアイコンをドラッグする。

時々、式の結果は単純な値ではなくオブジェクトとなります。この場合は、結果は< object reference >、並びにオブジェクトのタイプで表示され、結果行の隣に小さな赤いアイコンが表示されます（図 6.2)。

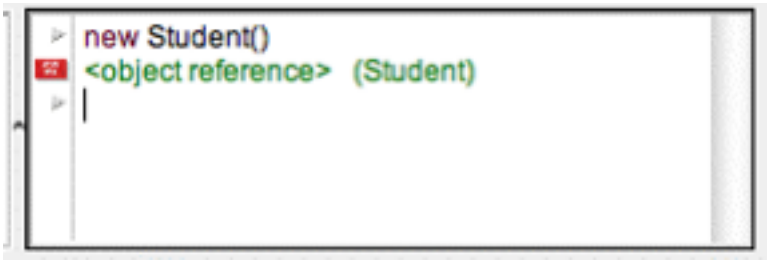

図 6.2: コード・パッドの式で得られたオブジェクト

結果は文字列の場合は、文字列の値は表示されるが、小さなアイコンも表示されます（文字列はオブジェクトであるためです)。

オブジェクトを生成する式の中から次のようなものが試せます。

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

小さなオブジェクト・アイコンは、結果オブジェクトの扱いに使用できます。アイコンをオブジェクト・ベンチにドラッグすることができます（図 6.3)。これは、オブジェクトをベンチに置いて、オブジェクトのポップ・アップ・メニューかコード・パッドでそのメソッドへの参照を可能にします。

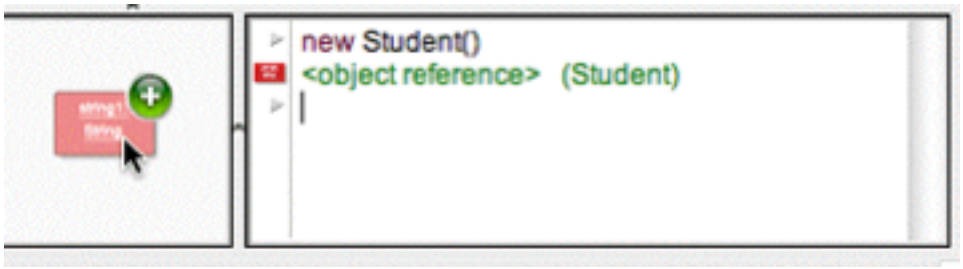

図 6.3: オブジェクトをオブジェクト・ベンチにドラッグするときの図

6.4 オブジェクトの検閲方法

要約： 結果のオブジェクトをコード・パッドで検閲するためには、小さなアイコンをダブル・クリックする。

コード・パッドの式で得られたオブジェクトを検閲したいときは、オブジェクトをオブジェクト・ベンチに置かなくても、オブジェクト・アイコンのダブル・クリックで通常のオブジェクト検閲を開くことができます。

6.5 文の実行方法

要約： コード・パッドに記入される文が実行されます。

コード・パッドは文（つまり、結果の返さない Java の命令）の実行に使用できます。次の文を試して下さい。

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

文は、末尾に「；」がなくても正しく評価され、実行されます。

6.6 複数行の文および文のシーケンス

要約： 数行にわたる文を記入するためには、行末に shift-Enter を使用する。

入力行の末尾に shift-Enter を使用すると、文のシーケンスもしくは数行にわたる文の記入ができます。shift-Enter の使用は記入位置を次の行の先頭に移動して文を実行しません。最後の記入行で Enter キーを押すと、全ての行は評価されます。例えば、次の for ループを試して下さい。

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

6.7 変数の扱い方

要約： 局所的な変数は単文および複数行の文に使用できます。オブジェクト・ベンチにあるオブジェクトの名前はインスタンス・フィールドとして扱います。

変数（インスタンス・フィールド、および局所的な変数）は、コード・パッドには限られた形で使用できます。

コード・パッドには局所的な変数が宣言できるが、異なる入力の間で変数が削除されるため、複数行の文を使用したときのみ有用です。例えば、次のブロックを一つの複数行の文として入力すると、期待通りで機能します。

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

しかし、一行ずつを異なる入力で記入すると、sum の変数は入力間に記憶されないのでこのシーケンスは失敗します。

記入する文はメソッドの中に書くテキストとして考えることができます。Java のメソッドで記入可能な全ての文はコード・パッドでも記入可能です。しかし、各の文は異なるメソッドのテキストとして見なされるので、一つの入力から別の入力で宣言された変数を参照することができません。

オブジェクト・ベンチにあるオブジェクトはインスタンス・フィールドとして考えられます。メソッド内（あるいはコード・パッド）で新インスタンス・フィールドを定義することができないが、インスタンス・フィールドを参照したり、そこにあるオブジェクトを呼び出したりすることができます。

コード・パッドからオブジェクトをオブジェクト・ベンチにドラッグすると、インスタンス・フィールドの生成が可能になります。

6.8　コマンドのヒストリ

要約： ヒストリを使用するためには、上向きおよび下向きの矢印キーを使用する。

コード・パッドは過去の入力のヒストリを保持します。上向きおよび下向きの矢印キーで過去の入力を呼び出しして、使用する前に編集することができます。

第7章　デバッグ

この章は、BlueJ のデバッグ機能の最も重要な要素を紹介します。コンピュータ関係の教員との話しから、初心者用の講義でもデバッガーの使用が好ましいが、その時間がなかなかないと聞きました。学生は、エディター、コンパイラ、およびプログラムの実行で精一杯で、もう一つの複雑なツールを学ぶ余裕がほとんどないそうです。

このため、本環境のデバッガーをできるかぎりシンプルにしました。目標は、15 分で説明できるデバッガーで、学生がそれ以上の説明なしですぐに使えるデバッガーです。では、成功したかどうかを見ていきましょう。

まず、伝統的なデバッガの機能を次の 3 つに絞りました。

- ブレイク・ポイントの設定
- コードのステップごとの実行
- 変数の検閲

このため、三つの何れの作業がとても簡単です。これからは、その一つ一つを試していきます。

始めるためにまず、BlueJ 配布の examples ディレクトリ内の debugdemo のプロジェクトを開いて下さい。このプロジェクトは、デバッガー機能を例示するために数クラスを含んでいます。これらのクラスはそんなに意味がありません。

7.1　ブレイク・ポイントの設定方法

要約： ブレイク・ポイントを設定するためには、エディターのテキストの左にあるブレイク・ポイント領域をクリックする。

ブレイク・ポイントを設定すると、コードの特定のポイントで実行を一時停止することができます。実行が停止している間に、オブジェクトの状態を調べることができます。これは、しばしばコードで行われていることを理解するための助けになります。

エディターで、テキストの左側にブレイク・ポイント領域があります（図 7.1）。そこにクリックすると、ブレイク・ポイントが一つ設定できます。ブレイク・ポイントを示す小さな STOP 標識が現われます。これを試してみましょう。Demo クラスを開いて、loop メソッドを見つけて、ループのどこかにブレイク・ポイントを一つ設定して下さい。エディターに STOP 標識が現われるはずです。

ブレイク・ポイントのあるコード行に辿り着いたとき、実行は停止します。これを試してみましょう。

Demo クラスのオブジェクトを生成して、その loop メソッドをたとえば 10 のパラメーターで呼び出して下さい。ブレイク・ポイントに達すると、すぐにエディターのウインドウがポップ・アップし、現在のコード行を強調します。デバッガーのウインドウもポップ・アップします。図 7.2 のようになります。

エディターに強調されている行は、次に実行される行です（実行がこの行の前に停止されました）。

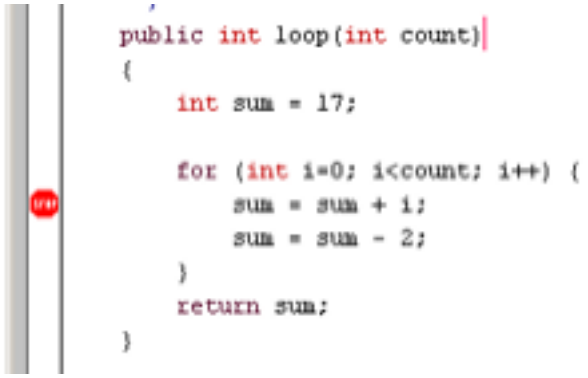

図 7.1: ブレイク・ポイントの一つ

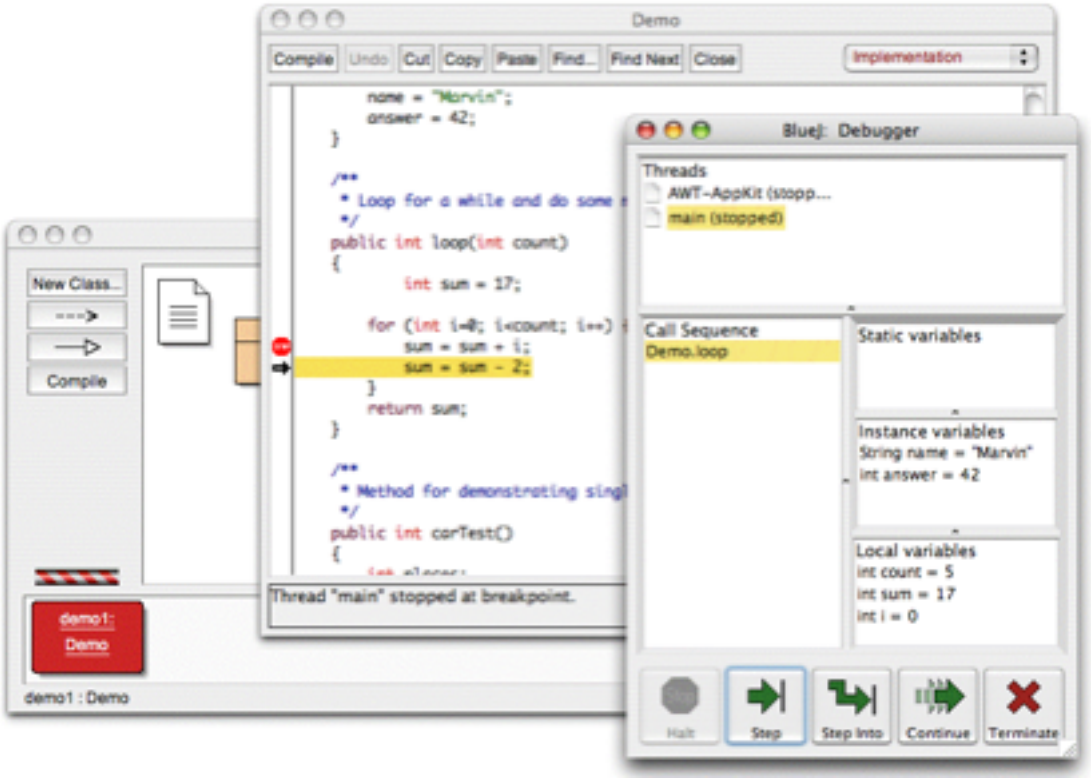

図 7.2: ブレイク・ポイントの一つ

7.2　ステップ毎の実行

要約： コードをステップ毎に実行するためには、デバッガーの Step もしくは Step Into のボタンを使用する。

実行の停止した状態で（これでメソッドが本当に実行され、そして現地点に辿り着くことを納得します)、コードを一行ずつ実行し、処理の進行を見ることができます。このため、デバッガー・ウインドウで Step ボタンのクリックを繰り返して下さい。この操作で、エディターのソース行が変わっていきます（ハイライト表示が、実行された行に合わせて移動します)。Step ボタンをクリックする度に、コードの一行だけが実行され、再び実行が停止します。デバッガー・ウインドウに表示される変数の値も変わります（たとえば、sum の値)。このように一行ずつ実行して、起こることが観察できます。これに疲れたら、再度ブレイク・ポイントをクリックすれば、これが削除できます。そうしてから、デバッガーの Continue をクリックすると、実行を再開し、通常通り続

けることができます。

これが別のメソッドで再び試してみましょう。Demo クラスの carTest() メソッドの次の行にブレイク・ポイントを設定して下さい。

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

そして、carTest() メソッドを呼び出して下さい。ブレイク・ポイントまでに辿ると、ちょうど、Car クラスの seats() メソッドを呼び出す直前に実行が停止します。Step をクリックすれば、その行が実行され、実行がその次の行に移動されます。このため、今回は、Step Into を試してみましょう。Step Into でメソッドを呼び出すと、そのメソッドに入って、メソッド自身を（単一のステップ実行でではなく）一行ずつ実行します。この場合は、Car クラスの seats() メソッドの内部に誘導されます。そこからコードの終端に辿って、呼び出しのメソッドに戻るまでに、快適にステップ実行ができます。この場合もデバッガーが変化するものを表示します。

対象の行がメソッドの呼び出しを含んでいない場合は、Step と Step Into は同様に動作します。

7.3 変数の検閲方法

要約： 変数の検閲は簡単です。変数の値がデバッガーで自動的に表示される。

自分のコードをデバッグするとき、オブジェクトの状態（ローカル変数やインスタンスの変数）が検閲できることは重要です。

BlueJ でこれをやるのは当たり前のことです。ここまでもう既にそのほとんどを見てきました。変数を検閲するために特別なコマンドは要りません。現行のオブジェクトの静的変数とインスタンスの変数、および実行中のメソッドのローカル変数はいつも自動的に表示され、更新されます。

その他の稼働中のオブジェクトとメソッドの変数を見るために、呼び出しシーケンスからメソッドを選択することができます。たとえば、carTest() メソッドにブレイク・ポイントを一つ再び試してみて下さい。デバッガー・ウインドウの左側に呼び出しシーケンスがあります。現在これは、

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

を表示しています。

これは Car.seats が、Demo.carTest によって呼ばれたことを示しています。このリストから Demo.carTest を選択することで、そのソースとメソッドの変数の現在値を検閲するこたができます。

new Car(...) の行を実行した後、ローカル変数 myCar の値が $< object\ reference >$ として示されていることが観察できます。String を除くオブジェクト・タイプのすべての値がこのように表示されます。この変数をダブル・クリックすれば、その検閲が可能になります。これをしますと、前に説明した（第 4 章の第 1 節）検閲用のウインドウが開いてきます。これでオブジェクトを検閲することと、オブジェクト・ベンチでオブジェクトを検閲することとは、ほとんど同じことです。

7.4 Halt と Terminate

要約： Halt は実行を一時停止する。Terminate は実行を終了させる。

時々、プログラムの実行が長引くと、すべてが順調かを疑問になってきます。無限ループがあるかも知れないし、ただ単に長時間処理を要するかも知れません。このことをチェックすることが

できます。Demo クラスの longloop() メソッドを呼び出してみて下さい。これはしばらく実行されます。

何がどうなっているのか知りたいです。画面の上に既にデバッガがなければ、デバッガー・ウインドウを開いて下さい。

次に、Halt ボタンをクリックして下さい。ブレイク・ポイントが設定されたのように、実行が中断されます。ここから、Step を使用して数ステップで変数を観察し、全部が順調であるかを見て下さい。完了するまでに多少の時間が更に必要でしょう。数回の Continue と Halt の使用で、どれほど速く数えられているかが観察できます。続けたくない場合には（たとえば、本当に無限ループに入っていると分かったとき)、全体の実行を終わらせるために Terminate をクリックすることができます。Terminate は頻繁に使うものではありません。マシンの終了で、正確に書かれているオブジェクトが不安な状態にして置いてしまうから、緊急事態のときのみには Terminate が使用されるべきです。

第8章　スタンド・アロン・アプリケーションの作成方法

要約： スタンド・アロン・アプリケーションを作成するためには、Project メニューの Create Jar File... を使う。

BlueJ は実行可能な jar ファイルを作成することができます。幾つかのシステムでは、実行可能な jar ファイルのダブル・クリックで実行ができます（例：Windows、および MacOS ）。また、次のコマンド、

```
java -jar <file-name>.jar
```

の入力でも実行できます（Unix や DOS プロンプト等）。

これはプロジェクト例の Hello で試してみましょう。Hello を開いて下さい（examples ディレクトリにあります）。プロジェクトがコンパイルされていることを確認しましょう。そして、Project メニューから Create Jar File... を選択して下さい。

主なクラスの指定を可能にするダイアログが開いてきます（図 8.1）。このクラスには有効な main メソッドが定義されていなければなりません（そのシグネチャは public static void main(String[] args) です）。

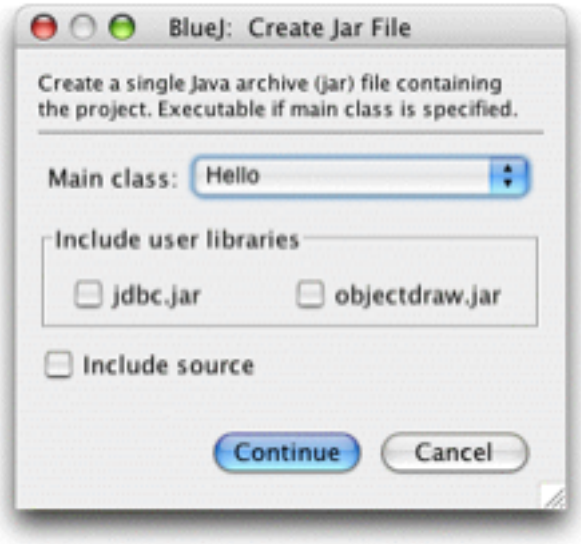

図 8.1: Create Jar File のダイアログ

この例では、主なクラスを選ぶのは簡単です。クラスは一つしかないからです。ポップ・アップメニューから Hello を選んで下さい。他のプロジェクトがあれば、実行したい main メソッドの持つクラスを選択します。

通常、実行可能なファイルにソースを含めないが、ソースを配布したいなら、それは可能です（例えば、他の人に電子メールでプロジェクト全体を送りたいときはこの jar フォーマットを使用することができます）。

ユーザのライブラリを使用するために BlueJ が（Preferences/Libraries か lib/userlib かで）設定されていれば、ダイアログの真ん中に Include user Libraries の領域が表示されます（何れのライブラリは使用されていなければ、これは表示されません)。自分のプロジェクトに使用されているあらゆるライブラリをチェックすべきです。

Continue をクリックして下さい。次に、作成する jar ファイルの名前を指定するファイル選択ダイアログが表示されます。hello を記入して、Create ボタンをクリックして下さい。

組み込まれるライブラリがなければ、hello.jar のファイルが作成されます。ライブラリがある場合は、hello のディレクトリが作成され、その中に hello.jar の jar ファイルが作成されます。このディレクトリは全必要なライブラリを含めます。あなたの jar ファイルは、自分の含まれているディレクトリ内に参照するライブラリがあると期待します。このため、ファイルを移動するときはこれらのファイルを一緒にして置いて下さい。

アプリケーションが GUI を利用する場合のみ、jar ファイルをダブル・クリックすることができます。我々の例はテキスト入出力を使用するので、テキスト・ターミナルから起動させなければなりません。今は、jar ファイルを実行してみましょう。

ターミナル・ウインドウか DOS ウインドウを開いて下さい。そして、jar ファイルのディレクトリに移動して下さい（そこに hello.jar ファイルがあるはずです)。システムに Java が正しくインストールされていれば、次のコマンドの入力で

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

ファイルが実行できるはずです。

第9章　アプレットの作成方法

9.1　アプレットの実行

要約： アプレットを実行するためには、アプレットのポップ・アップ・メニューから Run Applet を選択する。

BlueJ は、アプリケーションと同様にアプレットの作成も、実行もできます。BlueJ 配布ファイルの examples ディレクトリにアプレットが一つあります。まず、そのアプレットを実行してみたいです。examples ディレクトリの appletdemo プロジェクトを開いて下さい。

このプロジェクトには CaseConverter の名前の一つのクラスしかありません。クラスのアイコンは（<< *applet* >> で）アプレットとして表示されています。コンパイルの後、このクラスのポップ・アップ・メニューから Run Applet コマンドを選んで下さい。

幾つかの選択を可能にするダイアログがポップ・アップします（図 9.1）。

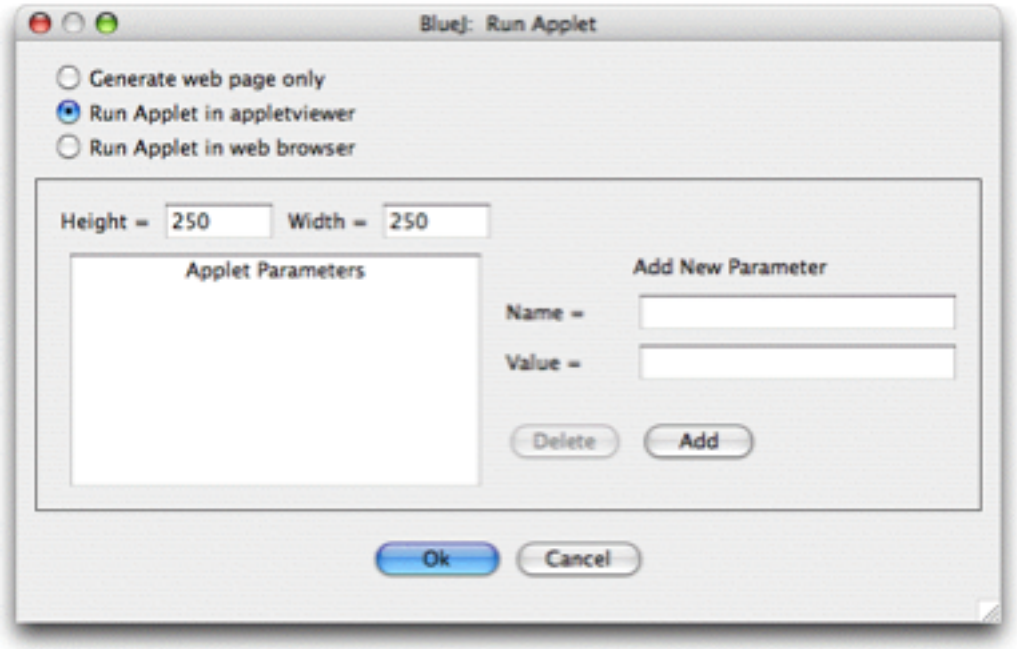

図 9.1: Run Applet のダイアログ

アプレット実行を、ブラウザ上で行なうか、あるいはアプレット・ビューアーで行なうかの選択があると分かるでしょう（また、実行しないで、該当ウエブ・ページだけを生成することもできます）。デフォルトの設定をそのままにして、Ok をクリックして下さい。数秒後、アプレット・ビューアーがポップ・アップして、CaseConverter アプレットを表示します。

アプレット・ビューアーが J2SE JDK（あなたの Java 環境）とともに常にインストールされているので、Java コンパイラと同じバージョンであることが保証されています。通常、アプレット・ビューアーの起こす問題数は、ブラウザの引き起こす問題数より少ないです。ブラウザが使用す

る Java バージョンが異なっているかもしれないし、ブラウザのバーションによって問題が起こるかもしれません。現在のブラウザでも問題が起こらないでしょう。

Microsoft Windows、および MacOS のシステムでは、BlueJ はデフォルトのブラウザを使用します。UNIX システムでは、ブラウザは BlueJ の設定により定義されます。

9.2　アプレットの作成方法

要約： アプレットを作成するためには、New Class ボタンをクリックして、クラス・タイプを Applet にする。

アプレットの実行方法を見てしまうと、今度は自分でアプレットを作成してみたくなります。

新しいクラスを作成して、そのタイプを Applet にして下さい（New Class のダイアログでタイプが選択できます)。コンパイルして、アプレットを実行して下さい。これだけです。別に悪くはないでしょう。

（他のクラスと同様に）アプレットにも幾つかの有効なコードを含むデフォルト・クラスの骨組みが作られます。アプレットの場合は、そのコードは二行のテキストです。エディターを開いて自分のコードを記入すれば、アプレットが編集できます。

デフォルトのコードに、通常アプレットに使用されるメソッドがコメント付き記入されます。また、サンプル・コードすべては paint メソッドにあります。

9.3　アプレットのテスト方法

場合によっては、(標準クラスと同様に）オブジェクト・ベンチにアプレットのオブジェクトを生成して置くことが便利です。これは可能です。アプレット・クラスのポップ・アップ・メニューには該当のコンストラクタがあります。オブジェクト・ベンチでアプレット全体を実行することはできないが、幾つかのメソッドを呼び出すことができます。これは、アプレットの一部分のために自分で書いたメソッドをテストするのに役立つでしょう。

アプレットにブレイク・ポイントを設けると、これらはウェイブ・ブラウザ、およびアプレット・ビューアーでの実行では無効です。これは、ウェブ・ブラウザおよびアプレット・ビューアーが自分のバーチャル・マシンをアプレットの実行に使用するからです。これらのマシンは BlueJ のブレイク・ポイントが全く解釈できません。

アプレットにはブレイク・ポイント、およびステップごとの実行を使用したいときは、Michael Trigoboff 氏が書いた AppletWindow のクラスを使用することができます。このクラスはフレームの提供で、BlueJ 内でアプレットの実行を可能にするためデバッグができます。このクラスおよび一つのデモは BlueJ のサイトの resources にあります。

第10章　その他の操作

10.1　非BlueJパッケージの開き方

要約： 非BlueJパッケージがProjectメニューのOpen Non BlueJ... コマンドで開ける。

BlueJでは、BlueJで作成されていないパッケージを開くことができます。これを行うには、Projectメニューから Open Non BlueJ... を選択します。Javaソース・ファイルのディレクトリを選択して、Open in BlueJボタンをクリックします。システムがそのディレクトリを開くかどうかを問います。

10.2　現行プロジェクトにクラスの追加方法

要約： 外部からプロジェクトにクラスをコピーするためには、Add Class from File... コマンドを使用する。

しばしば、他のところから取得したクラスをBlueJのプロジェクトに使いたいことがあります。たとえば、プロジェクトに使わせるために教員が生徒にJavaのクラスを与えたりすることがあります。Editメニューから Add Class from File... を選べば、自分のプロジェクトに既存のクラスを容易に組み入れることができます。これでは、組み入れるJavaソースファイル（.java拡張子のもの）を選択することができます。

クラスがプロジェクトにインポートされるとき、そのコピーが作成され、現行プロジェクトのディレクトリに保存されます。そのクラスを作成し、すべてのソース・コードを自分で書いたかのような結果になります。

もう一つの方法は、BlueJの外部から、対象となるプロジェクト・ディレクトリに新しいクラスのソース・ファイルを追加することです。次回該当プロジェクトを開くと、追加したクラスもプロジェクト・ウインドウに現れます。

10.3　mainおよびその他の静的なメソッドの呼び出し方法

要約： クラスのポップ・アップ・メニューから静的なメソッドが呼び出せる。

examplesディレクトリにあるhelloプロジェクトを開いて下さい。このプロジェクトの唯一のクラス（Helloクラス）は、標準的なmainメソッドを定義しています。

クラス・アイコンを右クリックすると、メニューにはコンストラクタだけでなく、静的なmainメソッドも含まれていることが分かります。このメニューから直接にmainを呼ぶことができます(オブジェクトが一つ生成しなくても可能です)。

すべての静的なメソッドは、このようにして呼び出すことができます。標準的なmainメソッドがパラメーターとしてString型の配列の入力を期待します。配列定数用の標準的なJavaの文法を使って、mainメソッドにString型配列を渡すことができます。たとえば、

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

のように（大括弧を含む）配列定数が渡せます。これを試してみて下さい。

サイド・ノート：標準 Java では、配列の定数がメソッドの呼び出しに使えません。初期化にしか使えません。BlueJ では対話的な呼び出しを可能にするために、パラメーターの受け渡しに配列の定数の使用が許されています。

10.4　資料の作成方法

要約： プロジェクト用の資料を生成するためには、Tools メニューから Project Documentation を選択する。

BlueJ はプロジェクト用の標準の javadoc 資料を生成することができます。これを行うには、Tools メニューから Project Documentation を選択します。この機能は、プロジェクトの全てのクラスのソース・コードから資料を生成して、その検閲のためにウエブ・ブラウザを開きます。

一つのクラスの資料は BlueJ のエディターでも生成し、検閲することができます。これを行うには、エディターを開いて、ツール・バーのポップ・アップ・メニューを使用します。Implementation の選択を Interface に変えて下さい。これは、javadoc の資料（クラスのインターフェース）をエディターで表示します。

10.5　ライブラリの使用方法

要約： Java の標準 API は Help メニューの Java Standard Libraries で見ることができる。

Java プログラムを書くときは、頻繁に Java の標準ライブラリを参照しなければなりません。オンライン（インターネットに接続している）状態であれば、Help メニューの Java Standard Classes の選択で、JDK API の資料を示すウエブ・ブラウザが開けます。

JDK 資料のインストールで、ローカル（オフライン）で見ることもできます。これについての詳細は BlueJ のウエブ・サイトのヘルプの節で説明されています。

10.6　ライブラリ・クラスからのオブジェクトの作成方法

要約： クラスのライブラリからオブジェクトを生成するためには、Tools の Use Library Class を使う。

BlueJ は、プロジェクト外部のライブラリで定義されているクラスからオブジェクトの生成を可能にしています。たとえば、String クラスか ArrayList クラスのオブジェクトを生成することができます。これらのライブラリのオブジェクトを試すためにはこの機能がとても便利です。

ライブラリのオブジェクトを生成するには、Tools の Use Library Class を使います。これを選択すると、java.lang.String のような有効なクラス名を要求するダイアログがポップ・アップします（注意：クラスを含む完全な正規のパッケージ名も必要です）。

テキスト入力用のフィールドは、最近使用したクラスのリストの示すポップ・アップ・メニューを持っています。クラス名を記入した後、Enter（Return）キーを押すと、ダイアログのリストにそのクラスのすべての静的なメソッドおよびコンストラクターが表示されます。リストからコンストラクターかメソッドの選択で、これらを呼び出すことができます。

その呼び出しはその他のコンストラクターかメソッドの呼び出しと同様です。

第11章　要約集

基本的な操作

1. プロジェクトを開くためには、Project メニューから Open を選択する。
2. オブジェクトを生成するためには、クラスのポップ・アップ・メニューからコンストラクタを一つ選択する。
3. メソッドを実行するためには、そのメソッドをオブジェクトのポップ・アップ・メニューから選択する。
4. クラスのソースを編集するためには、クラスのアイコンをダブル・クリックする。
5. クラスをコンパイルするためには、エディターの Compile ボタンをクリックする。プロジェクトをコンパイルするためには、プロジェクト・ウィンドウの Compile ボタンをクリックする。
6. コンパイル・エラー・メッセージ用のヘルプを得るためには、エラー・メッセージの右にある「?」のボタンをクリックする。

   もう少しやってみよう

7. オブジェクトの検閲はオブジェクトの内部状態を示すため、簡単なデバッグを可能にする。
8. オブジェクトのアイコンをクリックすることによって、そのオブジェクトをメソッドにパラメーターとして渡すことができる。

   新プロジェクトの作成方法

9. 新しいプロジェクトを作成するためには、Project メニューから New... を選択する。
10. クラスを作成するためには、New Class... ボタンをクリックして、クラス名を指定する。
11. 矢印を作成するためには、矢印のボタンをクリックしてプロジェクト・ウインドウ上に矢印をドラッグするか、またはソース・コードをエディターで書く。
12. クラスか矢印を削除するためには、そのポップ・アップ・メニューから Remove を選択する。

    コード・パッドの使用方法

13. コード・パッドの使用を開始するためには、View メニューから Show Code Pad を選択する。
14. Java の式を評価するためには、これらをコード・パッドに記入する。
15. コード・パッドのオブジェクトをオブジェクト・ベンチに転送するためには、オブジェクトの小さなアイコンをドラッグする。

16. 結果のオブジェクトをコード・パッドで検閲するためには、小さなアイコンをダブル・クリックする。
17. コード・パッドに記入される文が実行されます。
18. 数行にわたる文を記入するためには、行末に shift-Enter を使用する。
19. 局所的な変数は単文および複数行の文に使用できます。オブジェクト・ベンチにあるオブジェクトの名前はインスタンス・フィールドとして扱います。
20. ヒストリを使用するためには、上向きおよび下向きの矢印キーを使用する。

    デバッグ

21. ブレイク・ポイントを設定するためには、エディターのテキストの左にあるブレイク・ポイント領域をクリックする。
22. コードをステップ毎に実行するためには、デバッガーの Step ボタンもしくは Step Into のボタンを使用する。
23. 変数の検閲は簡単です。変数の値がデバッガーで自動的に表示される。
24. Halt は実行を一時停止する。Terminate は、実行を終了させる。

    スタンド・アローン・アプリケーションの作成方法

25. スタンド・アローン・アプリケーションを作成するためには、Project メニューの Create Jar File... を使う。

    アプレットの作成方法

26. アプレットを実行するためには、アプレットのポップ・アップ・メニューから Run Applet を選択する。
27. アプレットを作成するためには、New Class ボタンをクリックして、クラス・タイプを Applet にする。

    その他の操作

28. 非 BlueJ のパッケージが Project メニューの Open Non BlueJ... コマンドで開ける。
29. 外部からプロジェクトにクラスをコピーするためには、Add Class from File... コマンドを使用する。
30. クラスのポップ・アップ・メニューから静的なメソッドが呼び出せる。
31. プロジェクト用の資料を作成するためには、Tools メニューから Project Documentation を選択する。
32. Java の標準 API は Help メニューの Java Standard Libraries で見ることができる。
33. クラスのライブラリからオブジェクトを生成するためには、Tools の Use Library Class を使う。