BlueJ 教程

版本 1.2
针对 BlueJ 版本 1.1.x

Michael Kölling
澳洲 Monash 大学网络计算学院

清华大学计算机系知识工程组
郝建文、顾志峰 翻译

1 前言

1.1 关于 BlueJ

这份教程介绍了如何使用 BlueJ 这个编程环境。BlueJ 是一个专门为入门级教学设计的 Java™开发环境。它是由澳大利亚墨尔本 Monash 大学 BlueJ 小组设计并开发的。

需要更多信息请访问：http://bluej.monash.edu/

1.2 本书范围及读者

这份教程主要针对那些想让自己熟悉如何使用这个开发环境的读者。而不会讨论这个环境设计过程中的结构组织和那些值得研究的问题。

这份教程不打算介绍如何使用 Java，因此读者应该已经对 Java 比较熟悉。

这份教程不是一份全面的环境参考手册。许多细节问题没有涉及到——通过简单准确的介绍突出重点，而不是对各个专题全面的讨论。

大多数小结有一个一行左右的“本节小结”作为结尾。第 0 小结只是重复了一遍这些小结，作为一个快速的参考。

1.3 版权, 使用许可以及重新发布

1.4 反馈

我们欢迎鼓励任何形式的对 BlueJ 和这份教程的反馈意见，包括声明，问题，更正，批评等。请发信给 Michael Kölling (mik@monash.edu.au)

2 开始

2.1 安装

BlueJ 是作为一些 Java 的 class 文件以 jar 文件格式发布的。安装十分简单

安装前的准备

你必须在你的系统上安装了 JDK1.2.2 或更高的版本。一些函数在 JDK1.3 下工作会更好一些，所以推荐安装或升级到最新的 JDK 版本。如果你还没有安装 JDK，你可以从 sun 的网站下载：http://java.sun.com/j2se.

得到 BlueJ

BlueJ 的发布文件通常叫做 *bluej-xxx.jar*，其中 *xxx* 是版本号。比如，BlueJ 版本 1.1.1 的发布文件名是 *bluej-111.jar*，你可以从光盘上拷贝该文件，或者从 BuleJ 的网站下载：http://bluej.monash.edu/.

关于 SDK，JDK 和 JRE

有的时候存在一些混淆在 Java 的各个不同发布版本之间：SDK，JDK，JRE。你应该安装的是最新版的 Java 2 SDK(软件开发包)。JDK 和 SDK 是一样的，只是 JDK 是一个旧的名字。Sun 公司已经改变了他们的命名规则在某些场合，但是有的时候旧的名字还在使用，比如安装了 Java 2 SDK 版本 1.3，那么缺省的安装目录是 jdk1.3。

JRE(Java 运行环境)却有所不同，它是能运行 Java 程序的 SDK 的一个子集，对 BlueJ 而言，JRE 是不够的。我们需要 SDK 是因为 SDK 包含了一些 BlueJ 要使用的开发工具。JRE 在 SDK 安装的时候是会自动安装的。

安装

Windows：

双击安装文件*(bluej-xxx.jar)*

如果你的系统没有配置执行 jar 文件，那么双击可能会不起作用。在这种情况下，请打开一个 DOS 窗口并参考下面的 UNIX 安装方法

Unix：

使用下面的命令运行安装程序。注意：在这个例子中，我使用发布文件 *bluej-111.jar*—你应该使用你得到的文件的名字（使用正确的版本号）

<jdk-path>/bin/java –jar bluej-111.jar

*<jdk-path>*是 JDK 安装的目录

接着会出现一个窗口，让你选择 Blue 的安装目录和 JDK 的版本用来运行 BlueJ。注意：BlueJ 的安装路径（任何一个父目录）中不能有任何空格（比如：“Program files”）!

单击 *Install*, 完成之后，BlueJ 就装完了。

如果你有任何问题，请访问 BlueJ 网站上的 FAQ

2.2 启动 BlueJ

BlueJ 的安装程序会在安装目录里安装一个名叫 B*lueJ* 的脚本。在图形界面里，只要双

击那个

文件便可启动 BlueJ。在命令行界面（比如：Unix 或者 Dos），你可以带参数或不带参数启动 BlueJ：

$ bluej

或者

$ bluej example/people

2.3 打开一个工程

BlueJ 的工程和标准的 Java 包一样，是一个包含工程文件的目录。如果你是从命令行启动的 BlueJ，并且启动时你给出了一个工程目录作为参数，那么工程会自动被打开。如果启动时没有加参数，那么可以使用 *工程－打开* 菜单来选择并打开一个工程。

3 基础知识-编辑\编译\执行

为了学习在这一节，打开 Blue 的发布版本中的工程 *people*。你能够在 BlueJ 安装目录下的 example 目录下找到这个工程。打开工程后你因该可以看到和图 1 相似的窗口，这个窗口可能看上去和图 1 不完全一样在你的系统上，但区别应该很小。

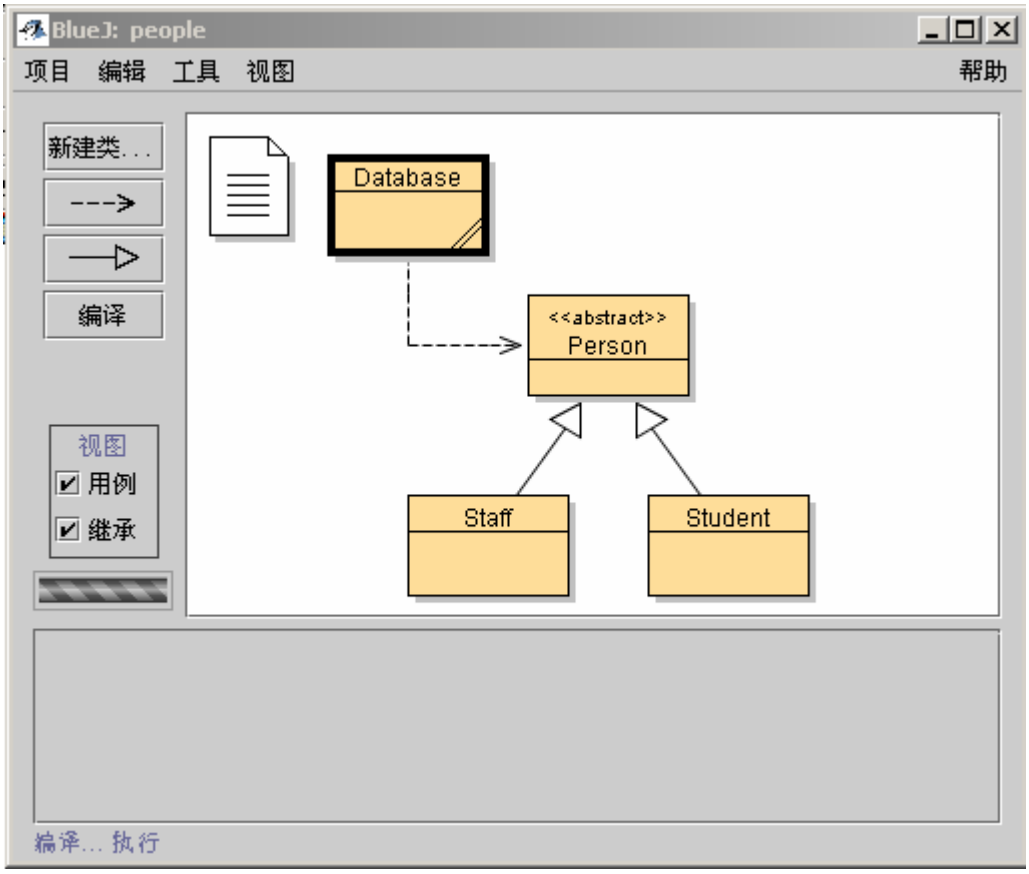

图 1：BlueJ 主窗口

3.1 创建对象

BlueJ 的一个很基本的特点是你不仅仅能够执行一个完整的应用程序，而且能直接和任何类的对象交互并执行其中的公共（public）方法。在 BlueJ 中一个执行过程通常通过创建一个对象，然后调用其中的方法来完成。这种模式在开发一个应用软件的过程中是十分有帮助的－你可以单独测试每个类一旦它被写完。这样就没有必要一开始就把整个应用程序写完。

> *旁注：静态方法能够被直接调用而不需要先创建一个类。main（）是一个静态方法，因此我们可以做和通常在一个 Java 应用程序中发生事件的相同的事情－通过执行静态方法 main（）启动一个应用程序。我们会在后面继续讨论。现在，我们来做一些其他有趣的不能在一般的 Java 环境中做到的事情。*

你看到的主窗口中间的方块（标注了 *数据库，人，教员和学生*）是代表了在这个应用程序中用到的类的图标。右键单击类的图标，你会得到一个包含能够应用到该类的操作的一个菜单（图 2）。图中显示的操作由 new 操作符和类中定义的各个构造函数的组合，以及 BlueJ 环境提供的一些操作组成。

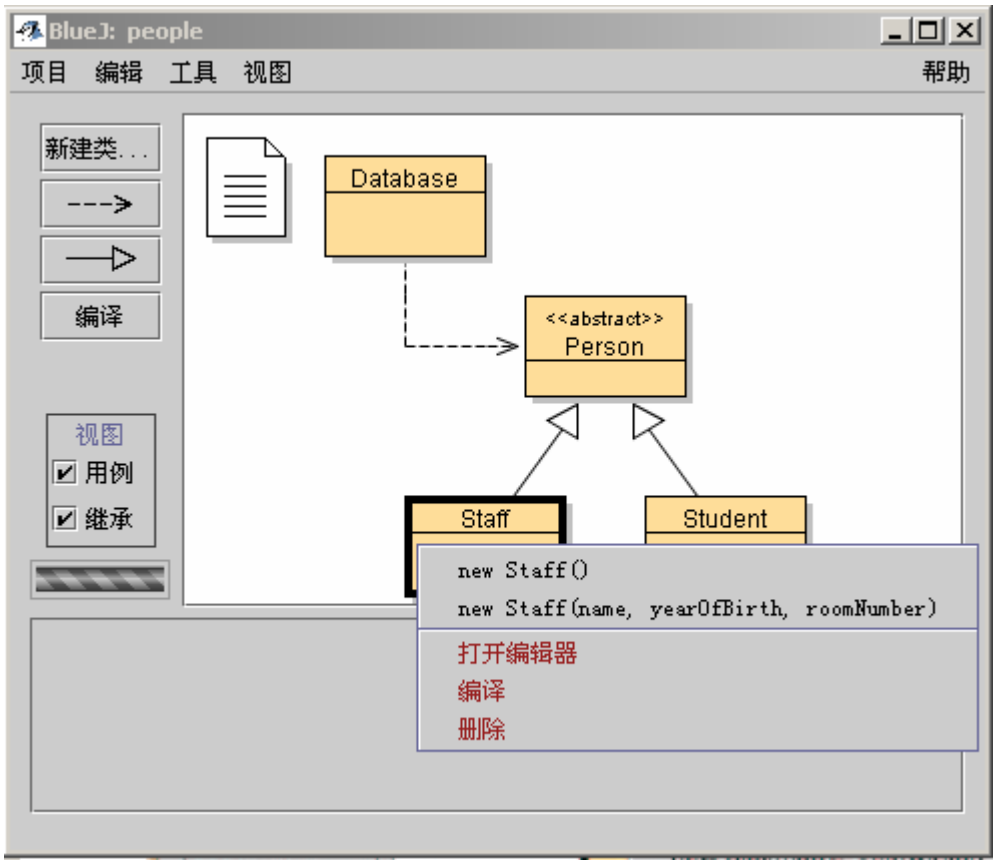

图 2：类操作（弹出菜单）

我们希望创建一个教员对象，所以你应该右键单击教员图标（会弹出图 2 中显示的菜单）。这个菜单显示了两个构造函数可以用来构造教员类。一个有参数一个没有。首先，选择没有参数的构造函数。弹出的对话框应该如图 3 所示.

图 3：创建没有参数的对象

这个对话框要求你输入一个被创建对象的名字。同时提供了一个缺省的名字（*staff_1*）。这个缺省的名字对我们来说足够好了。所以马上点击 *OK*。一个教员对象就会被创建。
一旦一个对象被创建，它会被放在窗口底部的对象槽里（图 4）。

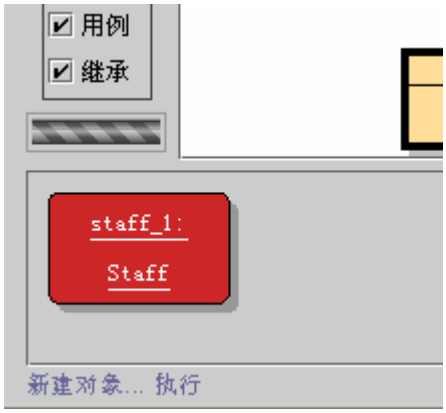

图 4：一个对象在对象槽里

你可能已经注意到 *Person* 类被标注为<<*abstract*>>（它是一个抽象类）。你会发现（如果你试一试）你不能创建抽象类的对象（就跟 Java 语言定义的规则一样）。

小结：*要创建一个对象，从类菜单中选择一个构造函数*

3.2 执行

现在你已经创建了一个对象，你可以执行它的公共（public）方法。用右键单击对象图标，就会弹出一个包含对象操作的菜单（图 5）。这个菜单显示了该对象所有可执行的方法和两个 BlueJ 环境提供的操作（*查看对象和删除对象*）。我们会在后面讨论这些功能。首先，让我们集中精力在方法上。

图 5：对象菜单

你可以看到菜单里有两个方法 *getRoom* 和 *setRoom*，分别设置和返回这个教员的房间号。试着调用 *getRoom*。只要简单的从菜单里选择它，它就会被执行。一个对话框会出现向你显示调用的结果（图 6）。在上面的情况下，结果是“(unknown room)”，因为我们还没有为这个人指定房间号。

图 6：显示一个调用函数的结果

从父类继承而来的方法在一个子菜单中。在对象弹出菜单的顶部有两个子菜单。一个包含从 *Object* 继承而来的方法，另一个包含从 *Person* 继承而来的方法（图 5）。你可以调用的 *Person* 的方法（比如：*getName*），通过选择子菜单中的函数名。试一下。你会发现结果同样的奇怪：它返回“（unknown name）”，因为我们还没有给这个人取一个名字。

现在让我试着来指定一个房间号。这个操作演示了如何调用有参数的方法（调用 *getRoom* 和 *getName* 有返回值，但没有参数）。通过选择菜单中相应的项调用 *setRoom*。一个对话框会出现提示你输入参数（图 7）。

图 7：调用带参数的函数

在这个对话框的最上端显示的是被调用方法的接口声明（包括注释和定义）。下面是一个文本框，你可以在里面输入参数。上面的定义告诉我们需要一个 String 类型的参数。在文本框里输入新的名字作为一个 String（包括引号），然后点击 *OK*。这样就结束了－因为这个方法没有返回值，所以没有结果对话框。再一次调用 *getName* 来检查名字是否真的改变了。

对对象的创建和方法调用练习一段时间。然后试着调用有参数的构造函数和更多的方法直到你对这些操作熟悉为止。

小结：*要执行一个方法，从对象菜单中选择它*

3.3 编辑一个类

到目前为止，我们只涉及到了一个对象的接口。现在是时候窥探一下对象内部了。你能够看到一个类的实现通过在类菜单里选择“*编辑类实现*”（提示：右键单击类图标就会跳出类菜单）。双击类图标可以实现同样的功能。这份教材对 BlueJ 的编辑器没有很深入的描述，但是它的使用应该来说是很简单的。关于编辑器的细节描述在后面会单独列出。现在，打开 *Stuff* 类的实现。找到 *getRoom* 方法的实现。正如函数所说的那样，它会返回该教员的房间号。让我们改变这个方法，在函数返回值的前面加上一个“*room*”前缀（这样这个方法就会返回，“room G..4.24”而不是“G..4.24”）。我们通过修改下面这一行做到这一点。

return room;

改为

return “room　” + room;

BlueJ 完全支持 Java，所以没有什么特殊的要求关于你如何修改你的类实现。只要遵循 Java 规范即可。

小结：*要编辑类的源码，双击类图标*

3.4 编译

改变代码之后（在你做任何其他事情之前），检查工程的全貌（主窗口）。你会发现教员的图标改变了：现在图标出现了条纹。条纹的出现意味着类文件还没有被重新编译过，自从上次修改以来。返回编辑器。

旁注：*你可能正在怀疑为什么那些类图标没有条纹在工程第一次被打开的时候。这是因为那些 people 工程中发布的类都是已经被编译过的。通常情况下随 BlueJ 一起发布的工程是没有被编译过的。所以做好看到大多数类都有条纹的准备，如果你是第一次打开工程的话*

在编辑器顶端的工具条包含一些经常使用的功能按钮，其中一个是 *编译* 。你可以使用这个按钮直接编译当前打开的类文件。现在点击 *编译* 按钮。如果你没有犯任何错误，那么在编辑器最下方的消息区会出现一条消息提示你这个类已经被编译完。如果你犯了一个错误导致语法出错，错误行会高亮显示，并且在消息区会有相应的错误提示。（假设你第一次进行编译是没有出错，现在试着造成一个语法错误－比如漏写分号－然后再次编译，看看会出现什么效果？）

在你成功的编译完之后，关闭编辑器。

旁注：*你没有必要显式的去保存一个类文件。源文件会自动保存在某些合适的时候（比如当编辑器关闭时或者类文件被编译时）。你也可以显式的保存文件如果你愿意（在编辑器的工具栏上有相应的按钮），但是这种操作真的没什么必要，除非你的系统相当不稳定，随时都有崩溃的危险，你又很担心你会丢失你的工作成果。*

工程窗口的工具条上同样也有编译按钮。这个编译操作会编译整个工程（实际上它判断哪些类需要重新编译然后按照正确的顺序重新编译这些类）。你可以试着修改 2 个或多个类（这样会有 2 个或多个类的图标出现条纹），然后点击 *编译* 按钮，如果在被编译的几个类中出现错误，编辑器就会被打开，错误位置和错误信息会显示出来。

你可能注意到了对象槽又被清空了。旧的对象会被删除每次实现改变之后。

小结：*要编译一个类，单击编辑器里 编译 按钮。要编译一个工程，单击工程窗口里的 编译 按钮*

3.5 使用编译器错误帮助

一个很普遍的现象是：初学的学生对理解编译器错误总是有很大的困难。我们试着提供了一些帮助。

再次打开编辑器，在源文件里制造一个错误，然后编译。一个错误信息会出现在编辑器的消息区。在它的右端有一个问号，你可以点击它来得到更多的关于这个类型错误的信息（图 8）。

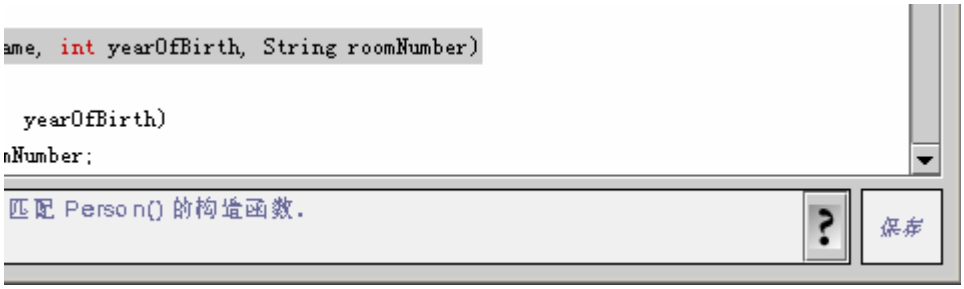

图 8：编译器错误和帮助按钮

在这种情况下，并不是所有的错误信息都有相应的帮助文档。一些帮助文档仍然需要被编写。但是还是值得一试的——许多错误已经有了相应的解释。剩下的那些错误信息将会被编写并包含在 BlueJ 将来的版本中。

小结：要得到编译器错误的帮助，单击错误信息旁边的问号图标。

4 进一步

在这部分中，我们将浏览一下另外一些经常在这个环境中需要做的事情。这些事情虽然不是必需的，但却是经常用到的。

4.1 观察对象

当你执行一个对象的方法的时候，你可能会注意到 *查看对象* 这项操作，它可以应用于除用户定义的方法之外的所有对象（图 5）。此操作可以查看对象的实例变量（“域”）的状态。试着用一些用户自定义的值创建一个对象（例如：以代参数的构造函数构造的一个 Staff 对象的实例）。然后从对象菜单中选择 *查看对象* ，就会弹出一个对话框显示该对象的域，其类型和其值（图 9）。

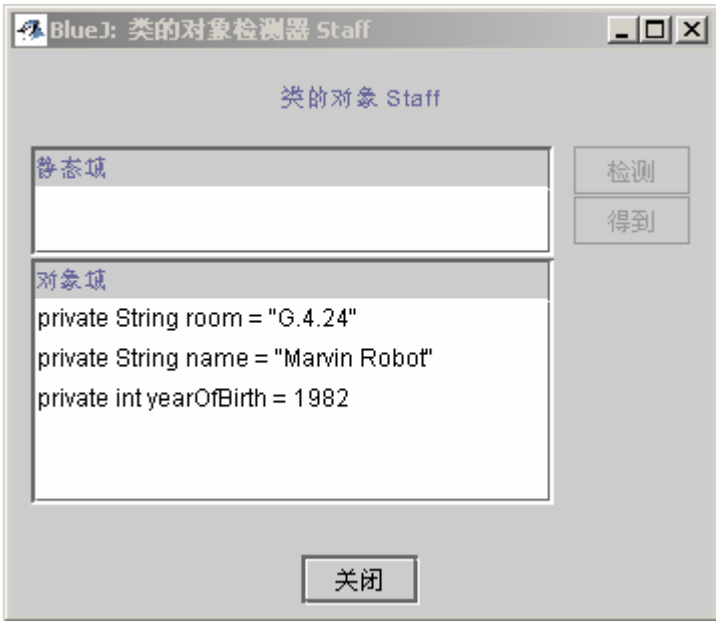

图 9：查看对话框

查看对象对于快速检查那些改变对象状态的操作是否被正确的执行了是很有用的。因此“查看对象”功能是一个简单的调试工具。

在 Staff 这个例子中，所有的域都是简单数据类型（要么是非对象类型，要么是字符串类型）。这些类型的值可以直接显示出来。这样你可以及时查看构造方法是否正确的完成了任务。

在更复杂的情况下，域的值可能是对于用户自定义对象的引用。我们将用另一个工程来演示这样的例子：打开工程 people2，它同样包含在标准的 BlueJ 版本。People2 的界面如下图 10 所示。可以看到，第二个例子除了有前面看到的那些类之外还有一个 Address 类。Person 类的一个域就是用户自定义的类型 Address。

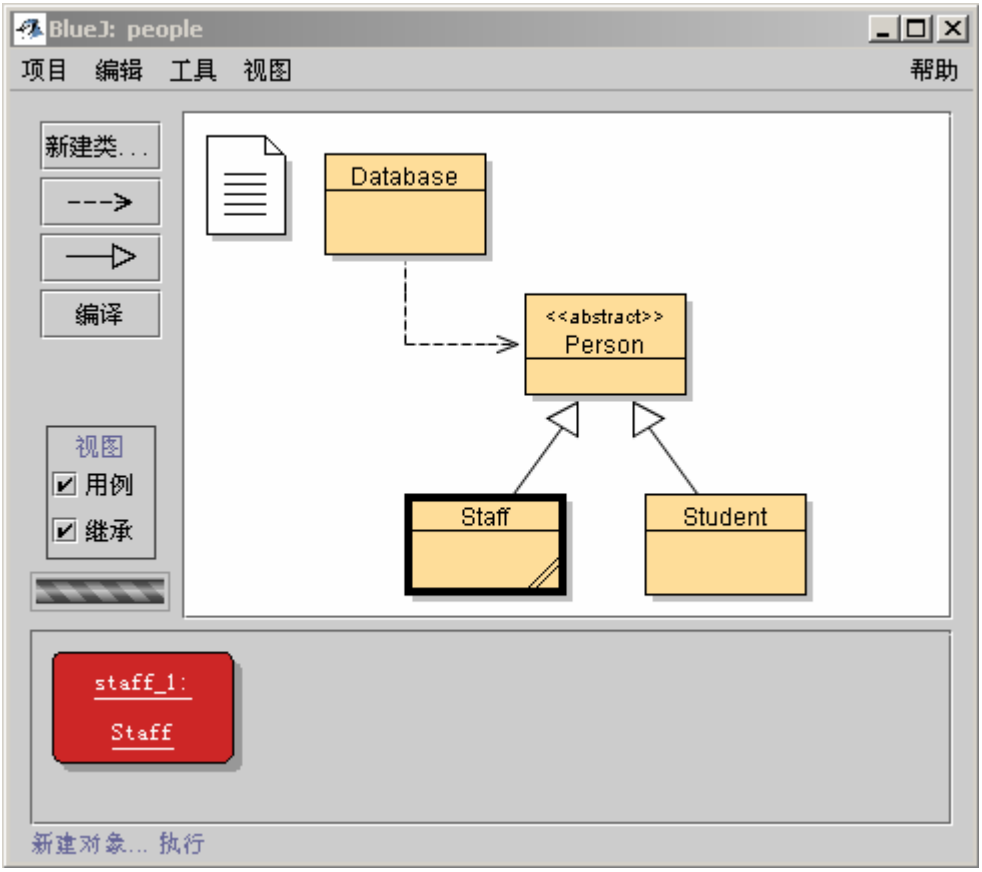

图 10：People2 工程窗口

接下来我们要试验的是——查看对象的域——创建一个 Staff 对象并且调用它的 setAddress 方法（你可以在 Person 子菜单中找到）。输入一个地址，在内部，Staff 的代码创建 Address 类的一个对象并且储存在 address 域里。

现在查看 Staff 对象。查看结果的对话框如图 11 所示。此时，Staff 类内部的域包含 address。如你所见，他的值显示为 <object reference>—因为这是一个复杂的，用户自定义的对象，他的值不能直接被显示在列表中。为了进一步检查 address，在列表中选择 address 域并且点击对话框中的 *查看对象* 按钮（你同样也可以双击 address 域）。这时就会弹出另一个查看窗口，其中显示了 address 对象的详细内容（图 12）。

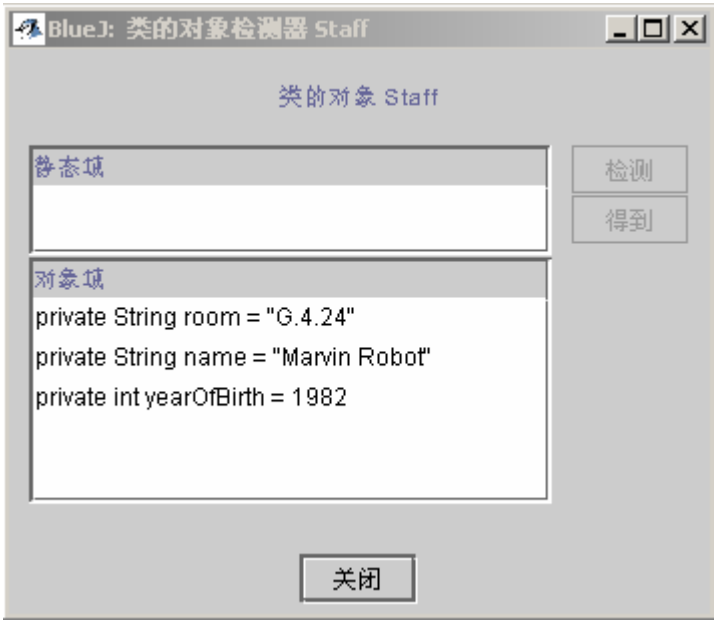

图 11：查看对对象的引用

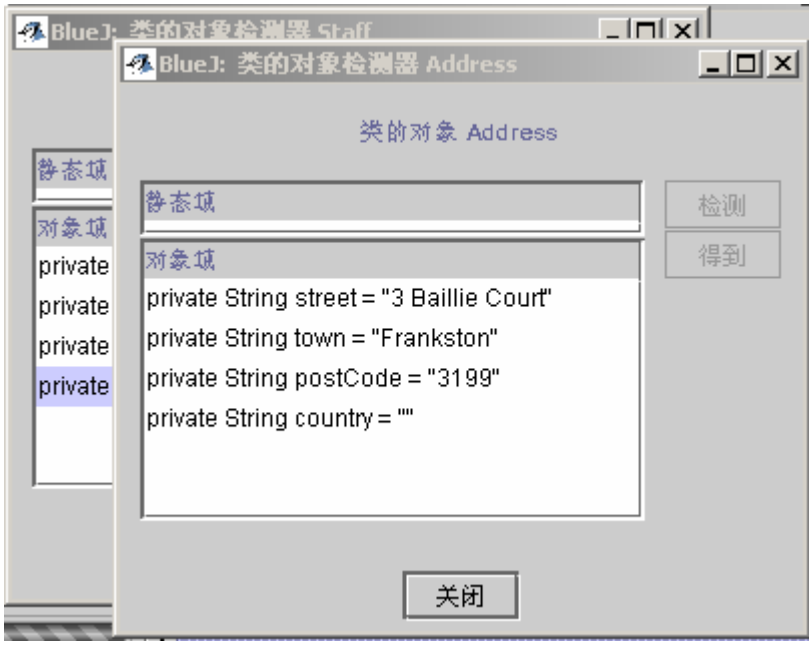

图 12：查看内部对象

如果选中的域是公共的，你同样可以选择 address 域并且点击 *Get* 按钮而不是 *查看对象* 按钮。此操作将选中的对象放入对象槽中。你可以进一步的通过调用它的方法查看它。

小结：*通过显示对象的内部状态，对象查看可以作为一些简单的调试手段。*

4.2 组合

“组合”指的是将对象作为参数传递给其他对象的能力。让我们试验一个例子。创建一

个 Database 类的对象（你会注意到 Database 类只有一个不带参数的构造方法，所以构造一个对象只有这样一种方式）。Database 对象有容纳一个 persons 列表的能力。它有相应的一些操作来添加 person 对象和显示所有当前储存的 person。（用 Database 来调用它实际上有点夸张了！）

如果你还没有在对象槽中建立一个 Staff 或者 Student 对象，首先创建它们中的一个。接下来，你在对象槽中同时需要一个 Database 对象和一个 Staff 或者 Student 对象。

现在调用 Database 对象的 addPerson 方法。提示告诉你需要一个 Person 类型的参数。（记住：Person 类是一个抽象类，因此没有任何对象是直接的 Person 类型。但是，作为自类型，Student 和 Staff 对象可以作为 Person 对象的替代。因此在需要 Person 的时候传送 Staff 或者 Student 是合法的。）为了将你的对象槽中的对象作为一个参数传送给你调用的方法，你可以在参数域中输入它的名字或者作为一种快捷方式，只需要点击你需要的对象，这将把它的名字输入到方法调用对话框中去。点击 *确定* 之后调用就生效了。因为这个方法没有任何的返回值，我们不能立刻看到结果。你可以调用 Database 的 listAll 方法查看此操作确实被执行了。listAll 操作把个人信息写入到标准输出。你会注意到一个文本终端自动被打开来显示这些文本。

可以再试几次，输入更多的 person 到“database”中。

小结：*通过点击一个对象的图标可以把一个对象作为参数传给一个方法调用。*

5 创建一个新工程

这一章将带你快速浏览如何建立一个新的工程。

5.1 创建工程目录

要创建一个新的工程，从菜单中选择 **工程－新建……**。然后自动打开一个文件选择对话框允许你为新工程确定一个名字和位置。现在试着这样做，你可以为你的工程选择任何名字。当你选择 *确定* 之后，将按照你提供的名字创建一个目录，并且主窗口显示当前这个新的，空的工程。

小结： 想创建一个工程，从 **工程** 菜单中选择 **新建……**。

5.2 创建类

现在你可以通过在工程工具条上点击 **新建类** 按钮创建你的类了。你将需要为这个类提供一个名字——这个名字必须是一个合法的 Java 标识符。

你同样可以一下四种不同类中选择：abstract, interface, applet 或者“standard”。这种选择将决定你的类将以何种初始的代码框架创建。你也可以通过编辑源代码修改类的类型（例如，把“abstract”关键字加入你的代码中）。

在创建一个类之后，它在图中以一个图标表示。不同的颜色标识不同类型的类，例如，蓝色表示一般的类，浅蓝色表示抽象类，绿色表示接口。当你为一个新类打开编辑器的时候，你会注意到一个默认的类框架已经搭好了——这将为你写入代码带来方便。默认框架代码是没有语法错误的，可以通过编译（但是它没有什么功能）。试着创建一些类并且编译它们。

小结： 想创建一个类，点击 **新建类** 按钮并且确定一个类名。

5.3 创建依赖关系

类框图以箭头显示框中各个类之间的依赖关系。继承关系（“extends”或者“implements”）被显示为双箭头；“uses”关系被显示为单箭头。

你既可以通过图形方式（直接在框图中）也可以通过在源代码中以文本方式添加依赖关系。如果你以图形方式添加了一个箭头，源文件也同时自动的更新了；如果你在源代码中添加了以来关系，框图也会自动更新。

想以图形方式添加一个箭头，点击想要的箭头按钮（双箭头是“extends”或者“implements”关系，单箭头是“uses”关系）。

添加一个继承箭头将在源文件中加入“extends”或者“implements”定义（依赖于目标是一个类还是一个接口）。

添加一个“uses”箭头不会直接改变源代码（除非目标是在另一个包中的类。那样将会产生一个“import”语句，但是在我们的例子中是看不见的）。如果拥有一个“uses”箭头指向另一个类而在源代码中实际上没有用到这个类，稍后将会产生一个警告，告知程序员声名

了对另一个类的“uses”关系但是这个类却没有用到。

用文本方式添加箭头很容易：只需要像平常一样敲入代码即可。当代码被保存的时候，框图也更新了（记住：关闭文本编辑器将自动保存文本）。

小结：想创建一个箭头，可以点击箭头按钮并且在框图中拖动箭头，或者只在编辑器中修改源代码。

5.4 删除元素

想从框图中删除一个类，选中这个类并且从编辑菜单中选择 **删除类**。你同样可以从这个类的弹出菜单中选择 **删除** 。想删除一个箭头，从菜单中选择 **删除箭头** 并且选中你想删除的箭头。

小结：想删除一个类，从它的弹出菜单中选择删除功能。想删除一个箭头，从编辑菜单中选择删除并且点击想要删除的箭头。

6 调试

本章介绍了BlueJ提供的调试功能的最重要的几个方面。在与计算机老师的交谈过程中，我们经常能听到这样的评论：如果在教学的第一年就能使用一个调试器将是一件很美好的事情，但是往往没有时间去介绍它。学生不得不重复修改，编译，然后执行这样的调试过程；没有剩下多少时间可以用来介绍另一个复杂的工具（调试器）。

所以我们决定把这个调试器做的尽可能的简单。我们的目标是编写一个 15 分钟内就能学会使用的调试器，并且学生们能够在没有更多指导的情况下使用它。现在来看看我们是如何做到这一点的。首先我们简化了传统调试器的在以下三个任务上的功能：

- 设置断点
- 单步执行
- 查看变量

作为回报，这三项任务每一项都变得很简单。我们现在就试着演示每一项任务。开始前，请打开工程 *debugdemo*，在安装目录下的 *example* 目录下可以找到。这个工程包含了一些专门为演示调试器功能的类——他们不关心其他的一些事情。

6.1 设置断点

设置断点允许你在源代码的某一行打断程序的执行。当程序的执行被打断后，你可以查看的你的工程的状态。它通常可以帮助你理解你的源代码到底做了什么。

在编辑器文本区的左边是断点区（图 13）。你可以通过在断点区单击来设置断点。一个小的停止图标会出现来标明断点。现在试一下。打开类 *Demo*，找到方法 *loop*，在 *for* 循环内部设置一个断点，停止图标应该会出现在你的编辑器窗口里。

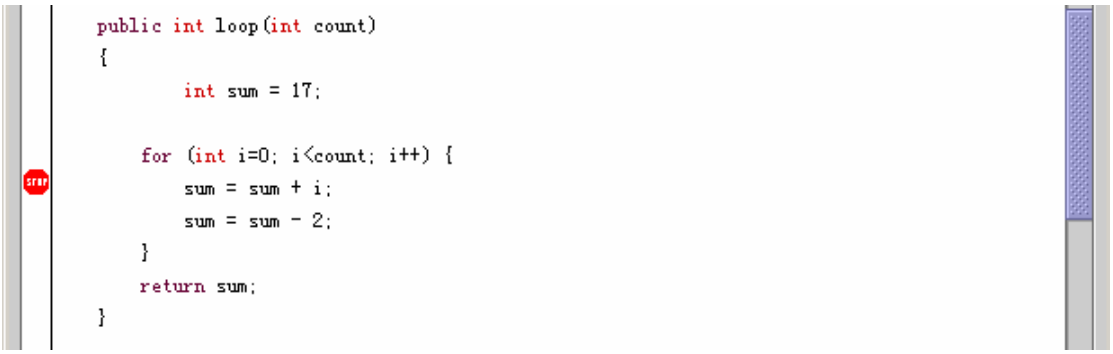

图 13：一个断点

当执行到断点所在的行时，执行过程就会被打断。让我们试一下。

创建一个 *Demo* 对象并且用参数（10）调用 *loop* 方法。只要执行到断点所在行，编辑器窗口就会自动跳出，显示当前行的代码。同时调试器窗口也会出现，它应该看起来如图 14 所示。

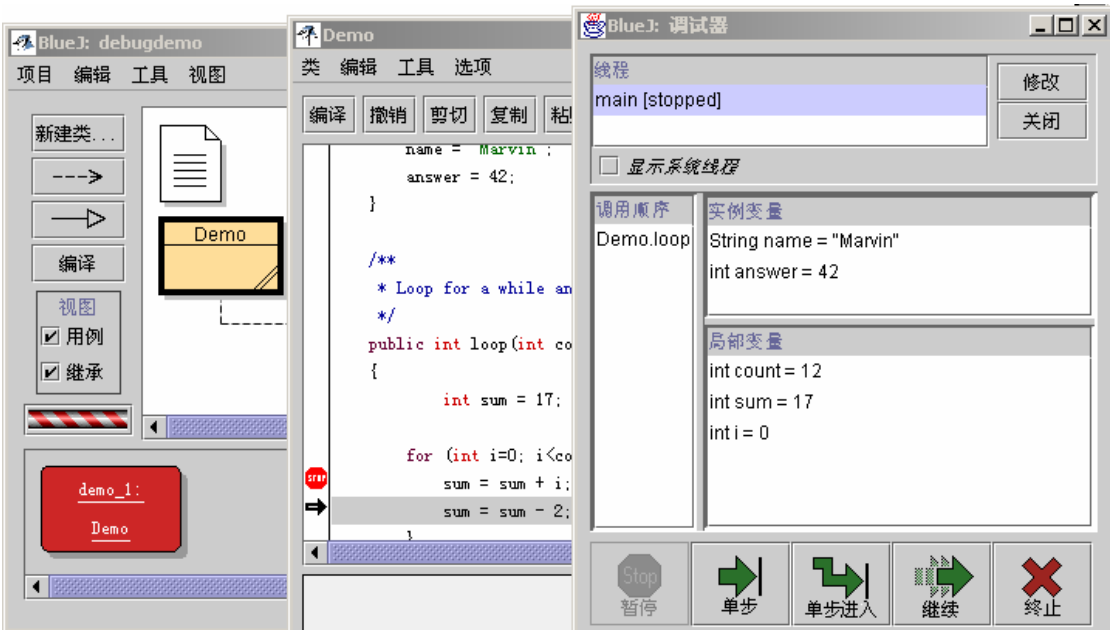

图 14：调试窗口

高亮显示的行是下一步要执行的行。（执行过程在改行还没被执行前被打断）。

小结：*要设置断点，单击编辑器左边的断点区。*

6.2 单步执行

现在我们已经打断了程序的执行（这是我们相信，这个方法确实被执行了，并且断点的代码也确实被执行到了），我们能够单步执行去查看这个程序是如何向下执行的。要做到这一点，不断的点击调试窗口里的 *单步* 按钮。你应该可以看到编辑器里的行号不断的变化（高亮显示的行随着将要被执行的行而移动）。每次你点击 *单步* ，一行会被执行并且又一次停止执行。同时请注意在调试窗口里显示的变量值也在不停的改变（比如 *sum* ）。所以你可以一步一步的执行来观察发生了哪些变化。一旦你厌倦了单步执行，你可以再一次点击断点图标来删除它，然后点击调试器里的 *继续* 图标来重新启动执行过程，按正常的执行顺序执行。

让我们用另一个方法来试一下。在 *Demo* 类 *carTest*（）方法里如下一行设置一个断点：

places = myCar.seats();

调用这个方法。当该行被执行到时，正要执行 *Car* 类的 *seats*（）方法。单击 *单步* 会执行整个行，而不会进入 *seats*（）方法。这次让我们试试 *单步进入* 。如果你使用单步进入，你就会进入到一个方法的内部，单步执行该方法（就跟 *单步* 一样）。你能够很高兴的单步跟踪这个方法知道该方法结束返回调用它的函数，注意调试器显示的变化。

单步 和 *单步进入* 在当前行没有的用方法的情况下是一样的。

小结：*要单步调试，使用调试器的* 单步 *或* 单步进入 *按钮*

6.3 查看变量

当你调试调试你的代码的时候，能够了解你的对象的当前的状态是很重要的（局部变量和成员实例变量）。

要做到这一点是很简单的——大部分你已经看到过了。你不需要特别的命令来查看变量；当前对象的成员变量和当前方法的局部变量都会被自动显示和更新。你可以选择调用栈里的方法来查看其他当前活动的对象或方法的变量。试一下，比如，在 *carTest（）*方法里再一次设置断点。在调试器窗口的左端，你能看见调用栈，当前的显示是

Car.seats

Demo.carTest

这表示 Car.seats 被 Demo.carTest 调用了，你可以选择这个列表中的 Demo.carTest 来查看源代码和当前各个变量的值。

如果你单步运行包含 new Car（…）的行。你可以看到局部变量 myCar 的值显示在 <*Object Reference*>里。所有 Object 子类类型变量的值（除了 String）是通过这个方法显示的。你可以通过双击这些变量来查看它们的值。这样做的话会弹出一个和前面描述一致的对象查看窗口（4.1 节），在这里查看对象状态和在对象槽里查看是一样的。

小结：*查看变量是很方便的——它们被自动显示在调试器窗口里*

6.4 暂停和中止调试

有的时候一个程序运行了很长的时间。你开始怀疑是不是出了问题。也许存在一个无限循环，也许长时间运行是正常的。好，我们可以检查一下。在 *Demo* 类里调用 *longloop()*，这个方法运行很长时间。

现在我们想知道程序运行的怎么样了。打开调试器窗口，如果它还没有被打开的话。（顺便说一下，在执行过程中点击代表虚拟机正在运行的旋转图标是打开调试器的一个快捷方式）

现在点击 *暂停* 按钮。程序的执行被停止了就跟我们设置了一个断点一样。现在你就可以单步执行一段，观察对象，看看是不是一切正常——它只是需要更多的时间来完成运行。你可以 *继续 暂停* 几次来看看它计数有多快。如果你真的不想让它再继续执行下去（比如，你发现你的程序已经陷入了死循环），你可以点击 *中止* 来中止程序的执行。中止操作不应该被频繁使用——这样会使用那些实现的很完美的类进入不一致的状态。所以建议只把这项功能作为一项紧急措施来使用。

小结：*暂停和中止可以用来暂时或永久打断程序的执行*

7 创建独立的应用程序

BlueJ 可以创建可执行的 jar 文件。可执行 jar 文件在某些系统上可以通过双击来执行（比如 windows）。或者通过使用命令 *java –jar <file-name>.jar*（Unix 或者 Dos 命令行）
我们通过例子程序 *hello* 来演示这一点，打开它（在 example 目录下）。确定这个工程被编译完了。从 *工程* 菜单选择 *导出...* 功能。

一个对话框会出现，询问你保存的格式，选择 *jar 文件* 来创建一个可执行的 jar 文件。为了是这个 jar 文件可执行，你还必须指定一个主类。这个类必须有一个有效定义的 main（）方法（使用定义 public static void main（String args[]））。

在我们的例子里选择主类是很简单的：因为只有一个类。从弹出菜单里选择 *Hello*。如果你有其他的工程，选择包含你希望被执行的 main（）方法的类作为主类。
通常情况下，你不需要把源文件包含到可执行文件中。但如果你愿意，你也可以同时发布你的源代码。

点击 *继续* ，接下来你会看见一个文件选择对话框让你指定要生成的 jar 文件的文件名。输入 Hello 然后点击 *确定*。jar 文件的创建就完成了。
你能够双击 jar 文件除非你的应用程序是图形界面的。我们的例子使用文本输入输出，所以我们不得不从一个字符终端启动它。现在让我们试着运行它。
打开一个终端或者 Dos 窗口。然后进入你保存 jar 文件的目录（你应该能够看见一个叫 Hello.jar 的文件）。假设你的系统上已经正确的安装了 Java，这样的话你就可以使用以下的命令

java –jar hello.jar

来执行这个文件

小结：*要创建一个独立的应用程序，使用 工程－导出...*

8 创建 Applet

8.1 运行一个 Applet

BlueJ 允许像应用程序一样创建和执行 Applets。在发布版本的示例目录中有一些 Applets。首先，我们要试着运行其中一个。从示例中打开 appletClock 工程。

你会看到这个项目中只有一个名为 Clock 的类。类的图标已经被（用 WWW 字样）标明为 applet。从类的弹出菜单中选择 *运行 Applet* 命令。

此时会弹出一个对话框，你可以输入一些选项（图 15）。

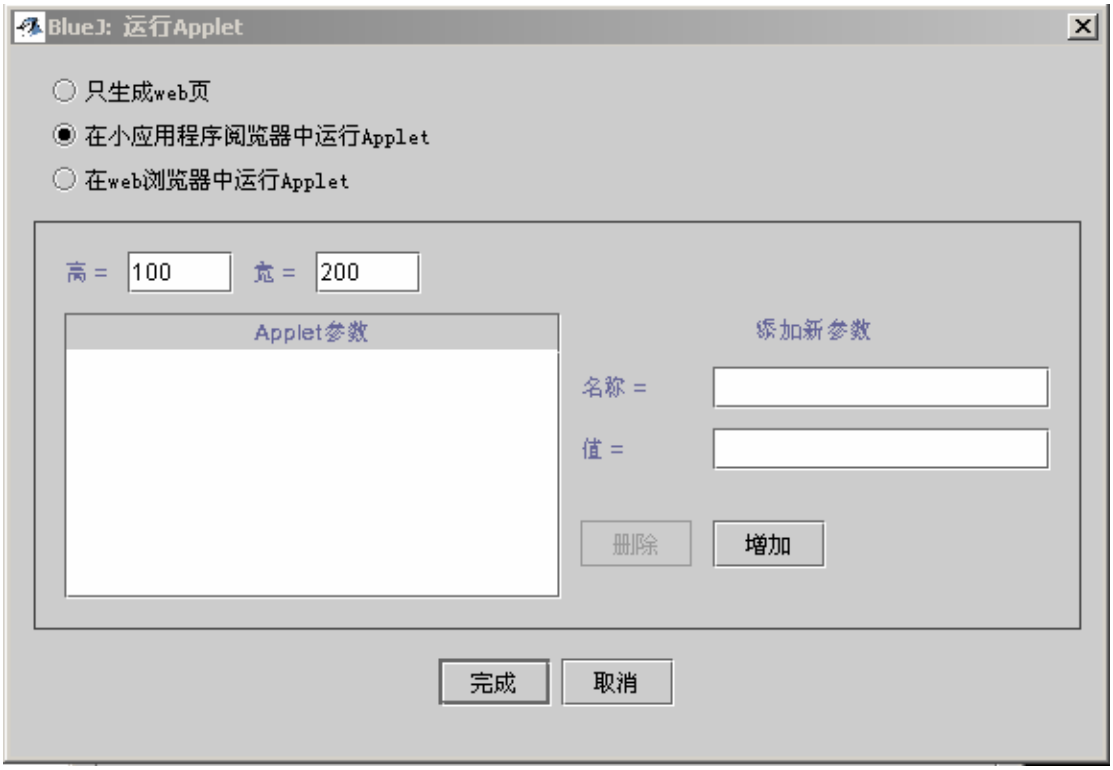

图 15：“运行 Applet “对话框

你会发现你可以选择是在一个浏览器中还是在一个 applet 查看器中运行一个 applet（或者仅仅创建一个 Web 页面而不运行它）。保留默认设置并且点击 *确定* 。过一会儿便会弹出一个 applet 查看器显示出这个时钟 applet。

Applet 查看器与你的 JDK 安装在一起，因此保证了它总是和你的 Java 编译器是同一个版本的。它通常会比浏览器产生更少的错误。你的浏览器可能运行不同版本的 Java 并且可能产生错误，这依赖与你所使用的浏览器版本。但是，现在大部分版本是工作正常的。

在 Microsoft Windows 操作系统中，BlueJ 使用你的默认浏览器。在 Unix 操作系统中，浏览器需要在 BlueJ 的设置中定义。

小结： 想要运行一个 applet，从 applet 的弹出菜单中选择 运行 Applet 。

8.2 创建一个 Applet

知道怎么运行一个 applet 之后，我们想要创建一个自己的 applet。

创建一个新的类，在类类型一栏中选择 Applet（你可以在新建类的对话框中选择类型）。编译，然后运行这个 applet，就这样。这不是很方便的吗？

跟其他类一样，Applet 创建的时候就已经自动构造了一个由一些合法代码组成的框架。这些代码用两行文字构成了一个简单的 applet。现在你可以打开编辑器并且编辑你的 applet，向其中插入你自己的代码。

其中你可以看到所有一般的 applet 方法，每一个方法都有一段说明其目的的注释。所有的样本代码都在 paint 方法中。

小结： 想创建一个 applet，点击 **新建类** 按钮并且在类的类型栏目中选择 Applet。

8.3 测试一个 Applet

某些情况下在对象槽中创建一个 applet 对象是很有用的（就如普通类一样）。你可以很容易的做到这一点——applet 的构造函数就在它的弹出菜单之中。你不能在对象槽中运行整个 applet，但是你可以调用一些方法。当你想要测试你在 applet 中已经实现的某一个方法的时候，这是很有用的。

9 其它操作

9.1 在 BlueJ 中打开非 BlueJ 的软件包

BlueJ 允许你打开那些已经存在的但并非用 BlueJ 生成的软件包。你只需要从菜单中选择 工程—打开非 BlueJ…… 即可。选择包含了 Java 源文件的目录，然后点击 在 BlueJ 中打开 按钮。系统将会要求你确认是否要打开这个目录。

小结： 非 BlueJ 软件包可以用 **工程：打开非 BlueJ** 命令打开。

9.2 在工程中加入已有的类文件

你经常想要在 BlueJ 的工程中使用一些从其他地方得到的类。例如，老师可能给学生一个 Java 类，这个类要在某个工程中用到。你只需要在菜单中选择 编辑—从文件添加一个类…… 就可以轻松的将以一个已经存在的类合并到你的工程中去。这样 BlueJ 允许你导如一个选中的 Java 源文件（文件名以 .java 结尾）。

当某个类成功的被导入一个工程以后，它就被复制并且储存到当前的工程目录中。这跟你创建一个类并且写入所有的代码效果是完全一样的。

另一种可供选择的方法是，直接从外部添加一个类的源文件到你的工程所在目录中。下次启动 BlueJ 的时候，这类就会被包含在工程框图中。

小结： 可以使用 **从文件添加类……** 命令把需要的类从外部拷贝到工程中来。

9.3 调用 main 函数和其他静态成员方法

从 examples 目录中打开 hello 工程。工程中唯一的一个类（类 Hello）定义了一个标准的 main 方法。

鼠标右键点击这个类，你可以看到类的菜单中不仅有这个类的构造方法，而且还有一个静态的 main 方法。你可以直接从菜单中调用 main 方法（并不需要像我们期待的那样为静态方法首先创建一个对象）。

所有的静态方法都可以这样调用。标准的 main 方法需要一个字符串数组参数。你可以使用标准的 Java 数组构造语法传递一个字符串数组。例如，你可以把

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

传递给方法。试一试！

> *旁注：在标准 Java 中，数组构造方法不能作为方法调用的实际参数。它只能用来初始化数组。但在 BlueJ 中，为了支持对标准 main 方法的交互式调用，我们允许把数组的构造方法作为参数传递。*

小结： 静态方法可以从类的弹出菜单中直接调用。

9.4 使用类库

在你写 Java 程序的时候，你经常需要查看 Java 的标准库。你可以通过从菜单中选择 帮助—Java 标准类 在 web 浏览器来打开 JDK API 文档（如果你是在线的）。

JDK 的帮助文档也可以在本地安装和使用（未联机方式）。在 BlueJ 的参考手册中有详细说明。

小结： 可以选择 **帮助—Java 标准库** 来查看 Java 的标准类 API。

10 小结

- 基础知识：
  1. 要创建一个对象，从类菜单中选择一个构造函数
  2. 要执行一个方法，从对象菜单中选择它
  3. 要编辑类的源码，双击类图标
  4. 要编译一个类，单击编辑器里 *编译* 按钮。要编译一个工程，单击工程窗口里的 *编译* 按钮
  5. 要得到编译器错误的帮助，单击错误信息旁边的问号图标
- 进一步：
  6. 通过显示对象的内部状态，对象查看可以作为一些简单的调试手段
  7. 通过点击一个对象的图标可以把一个对象作为参数传给一个方法调用
- 创建一个新工程：
  8. 想创建一个工程，从 工程 菜单中选择 新建……
  9. 想创建一个类，点击 新建类 按钮并且确定一个类名
  10. 想创建一个箭头，可以点击箭头按钮并且在框图中拖动箭头，或者只在编辑器中修改源代码
  11. 想删除一个类，从它的弹出菜单中选择删除功能
  12. 想删除一个箭头，从编辑菜单中选择删除并且点击想要删除的箭头
- 调试：
  13. 断点，单击编辑器左边的断点区
  14. 要单步调试，使用调试器的 *单步* 或 *单步进入* 按钮
  15. 查看变量是很方便的--它们被自动显示在调试器窗口里
  16. 暂停和中止可以用来暂时或永久打断程序的执行
- 创建独立的应用程序：
  17. 要创建一个独立的应用程序，使用 工程－导出…
- 创建小应用程序

  18、运行小应用程序，从小应用程序的弹出菜单中选择“运行 applet” 功能。

  19、创建小应用程序，点击“新建类” 按钮，并指定文件名及类型为 applet。
- 其它操作

  20、打开非 BlueJ 包，用主菜单中的“工程”——“打开非 BlueJ 文件”。

  21、类文件可从工程外被拷贝到工程中，用“编辑”——“从文件添加类”命令。

  22、静态方法可用弹出菜单调用。

  23、可通过菜单中的“帮助”——“Java 类库”，打开浏览器查看 Java API 文档。