

How to Run Java Programs Using BlueJ



Please read Chapter 2 before this appendix.

BlueJ is unique among enhanced editors (some may prefer to call it a simple IDE) for its automatic display of UML-like program diagrams and its environment that lets users create objects and interact with them through visual interface. However, we will describe only the process of creating and running the same program used in the minimalist approach so we can provide a clear comparison between the two approaches. We won't be describing the unique features of BlueJ in any detail here. If you choose to use BlueJ, then we recommend you to study the tutorial available at the BlueJ website.

1. Installation

Installation must proceed in two steps:

1. Download and install the most recent version J2SE SDK ver 1.4 for MS Windows. We assume the SDK is installed in the directory

`c:\j2sdk1.4.0_01`

Warning: J2SE SDK must be installed before installing BlueJ. BlueJ website recommends SDK ver 1.3 at the time of this writing. Please check their website and install the appropriate SDK.

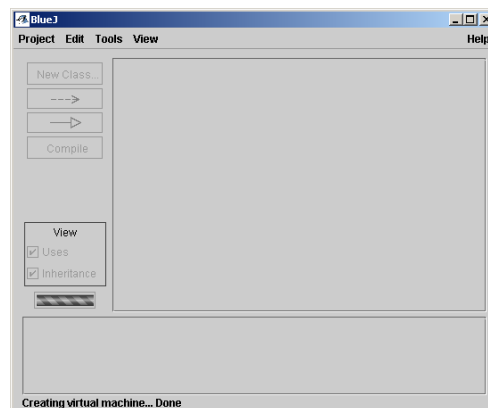
2. Download and install the most recent version of BlueJ (as of this writing, it is version 1.2.2) from www.bluej.org. Select any directory for installing BlueJ. During the installation, verify the option to create a desktop icon is selected (this is default). When the installation is successful, you should see the shortcut



on the desktop.

3. Start BlueJ

Double-click the Blue shortcut on the desktop. Initial BlueJ window appears on the screen.

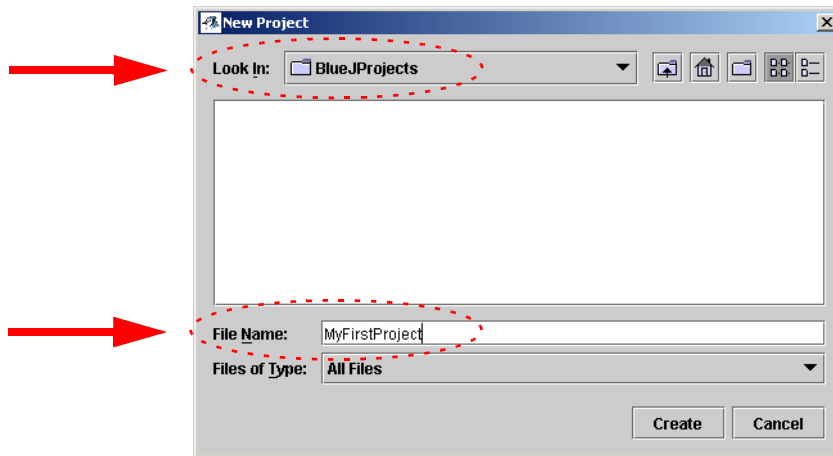


4. Create a Project

To create and run Java programs with BlueJ, we must create project files. BlueJ uses a project file to keep track of various information about a program. Before we create a new project, we will create a folder to keep all of the BlueJ projects. We assume the folder

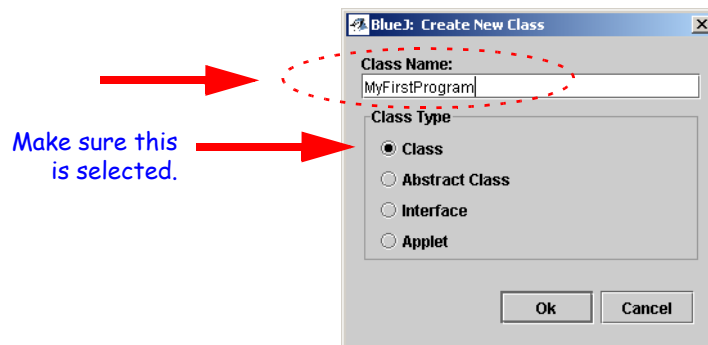
```
C:\BlueJProjects
```

in this example. After this folder is created, select the menu choice Project/ New Project... The New Project dialog appears. Change the folder to C:\BlueJProjects and enter MyFirstProject as the name of the project.

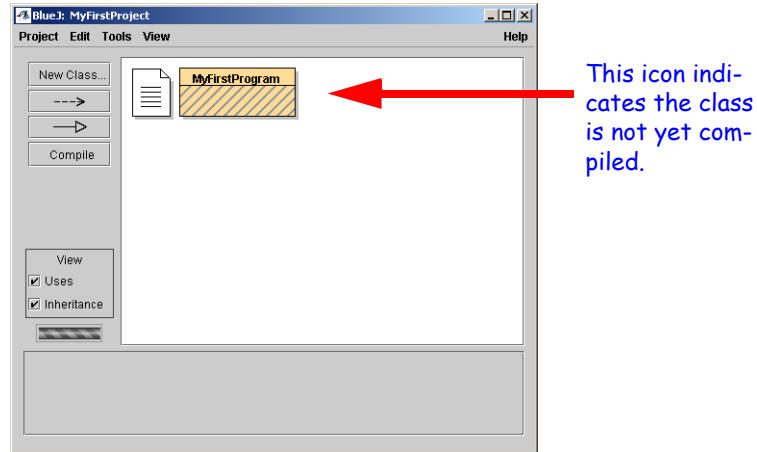


5. Create a Program

We are now ready to create a program. This program contains one (main) class. Select the menu choice Edit/New Class... The Create New Class dialog appears on the screen. Enter MyFirstProgram as the class name:



After you click the OK button, BlueJ displays a class icon:



Double-click the class icon, and an editor window appears with a template class definition. We won't be using this template, so erase everything in the window and enter the following program:

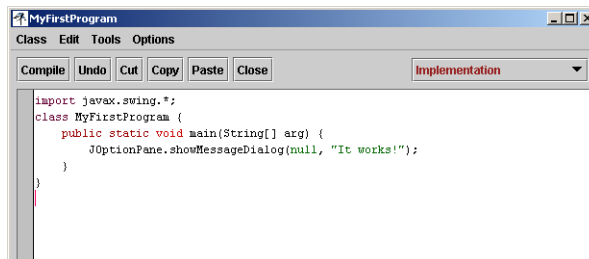
```
import javax.swing.*;
class MyFirstProgram {
    public static void main(String[] arg) {
        JOptionPane.showMessageDialog(null, "It works!");
    }
}
```

Type in the program exactly as shown, making sure the upper and lower-case letters are entered correctly. Notice that the statement

```
System.exit(0)
```

is not included in the program. The `exit` statement is not necessary in the BlueJ environment so we omitted it. It will cause no problem if you includ-

ed it, however. After the program is entered, the editor window should look as follows:



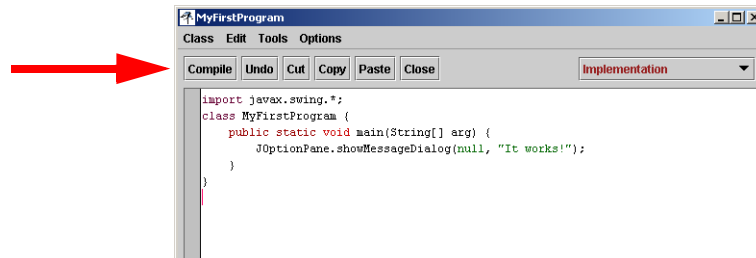
```
import javax.swing.*;
class MyFirstProgram {
    public static void main(String[] arg) {
        JOptionPane.showMessageDialog(null, "It works!");
    }
}
```

6. Save the Program

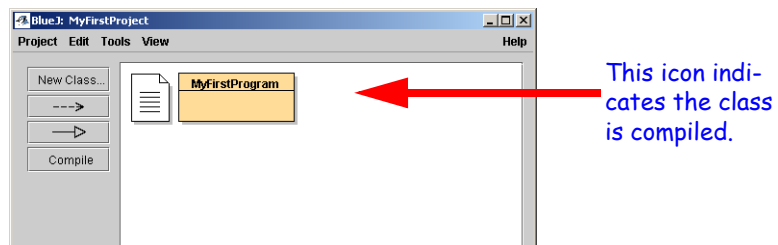
From the editor window, select the menu choice **Class/Save** to save the class. This step is actually optional because when you create a new class, its source file is automatically saved and updated by BlueJ as you edit the class.

7. Compile the Program

Click the **Compile** button on the editor window:

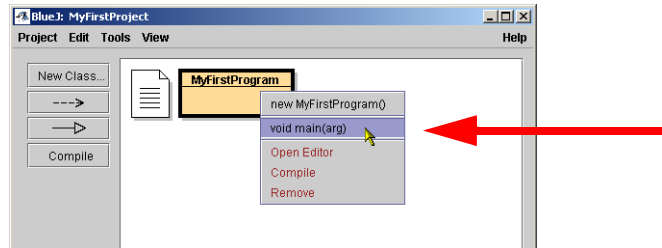


You can also compile a class by right-clicking (click with the right button) the class icon and selecting the menu choice **Compile** in the popup menu. When the compilation is successful, the visual appearance of the class icon changes to a plain rectangle:

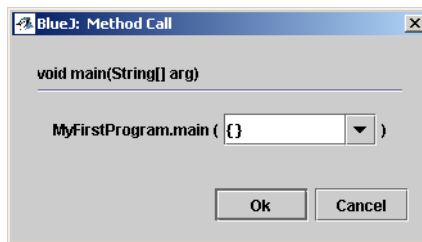


8. Run the Program

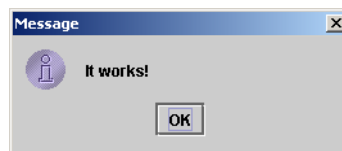
To run the program, right-click (click with the right button) the class icon and select the void main(arg) choice in the popup menu.:



With this action, you are calling the main method of the (main) class. The Method Call dialog appears on the screen:



This dialog allows the programmer to pass different values as arguments to the main method. We won't be using such features when calling the main method of the sample programs in this book. Just click the OK button. The program starts, and the dialog with the message It Works! appears on the screen:



(Note: If you don't see the dialog, it may be hidden behind the BlueJ window.)

Using Programmer-Defined Packages

When using classes from the system packages such as `javax.swing`, `java.util`, and others, all we do is to include appropriate import statements in the program. This is not enough when using classes from programmer-defined packages. We must

also set the environment correctly. How we set the environment is dependent on the development tools we use. We will describe how to set the environment to use programmer-defined packages with the minimalist approach and BlueJ.

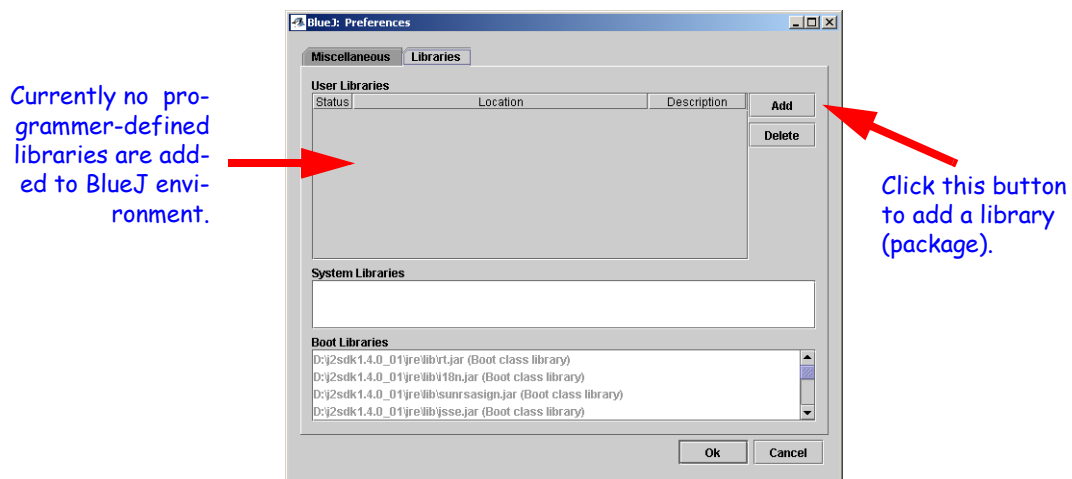
We will use the author-provided javabook package as an example to illustrate the procedure. You can download this package and its documentation from the textbook website. We assume you have downloaded and install the javabook classes under the C:\JavaPrograms\javabook directory. Notice that the name of the package is javabook, and the directory that contains the classes in this package is also named javabook. This is the requirement. The classes in the package xyz must be placed in the directory named xyz. The directory xyz, however, can be placed anywhere you want. In this example, we put the javabook directory under C:\JavaPrograms. When the installation is done correctly, you should see the source and bytecode files of the javabook classes such as MainWindow.java, MainWindow.class, OutputBox.java, OutputBox.class, and others in the C:\JavaPrograms\javabook directory.



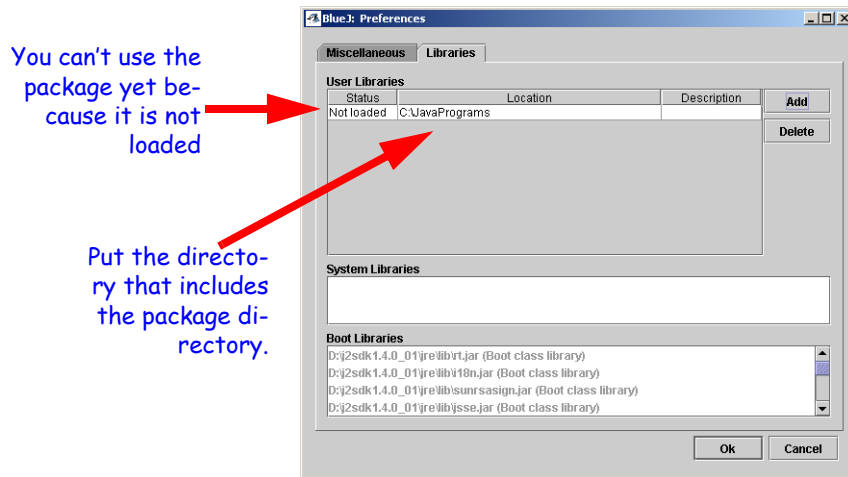
For this example, we assume the classes from author-provided javabook package are placed in the directory

C:\JavaPrograms\javabook

Select the menu choice Tools/Preferences... The Preferences dialog appears on the screen. Click on the Libraries tab. No programmer-defined libraries (i.e., packages) are listed at this point:

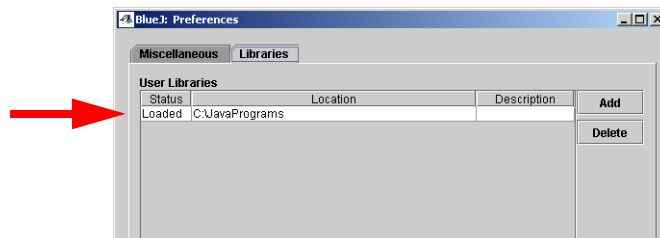


Click the Add button. The File Chooser dialog appears. Select the C:\JavaPrograms directory and click the OK button. The specified directory is added to the User Libraries list.



Notice that we are not listing the package directory itself, but the directory that includes the package directory.

The Status field says “Not Loaded,” which means you cannot import the classes from the package in your program yet. To load the package and be able to use the classes in your program, you must first close BlueJ and then restart it. After you restart BlueJ, open the Preferences dialog and click the Libraries tab. The Status field should now say “Loaded.”



You are ready to import javabook classes and use them in your program.