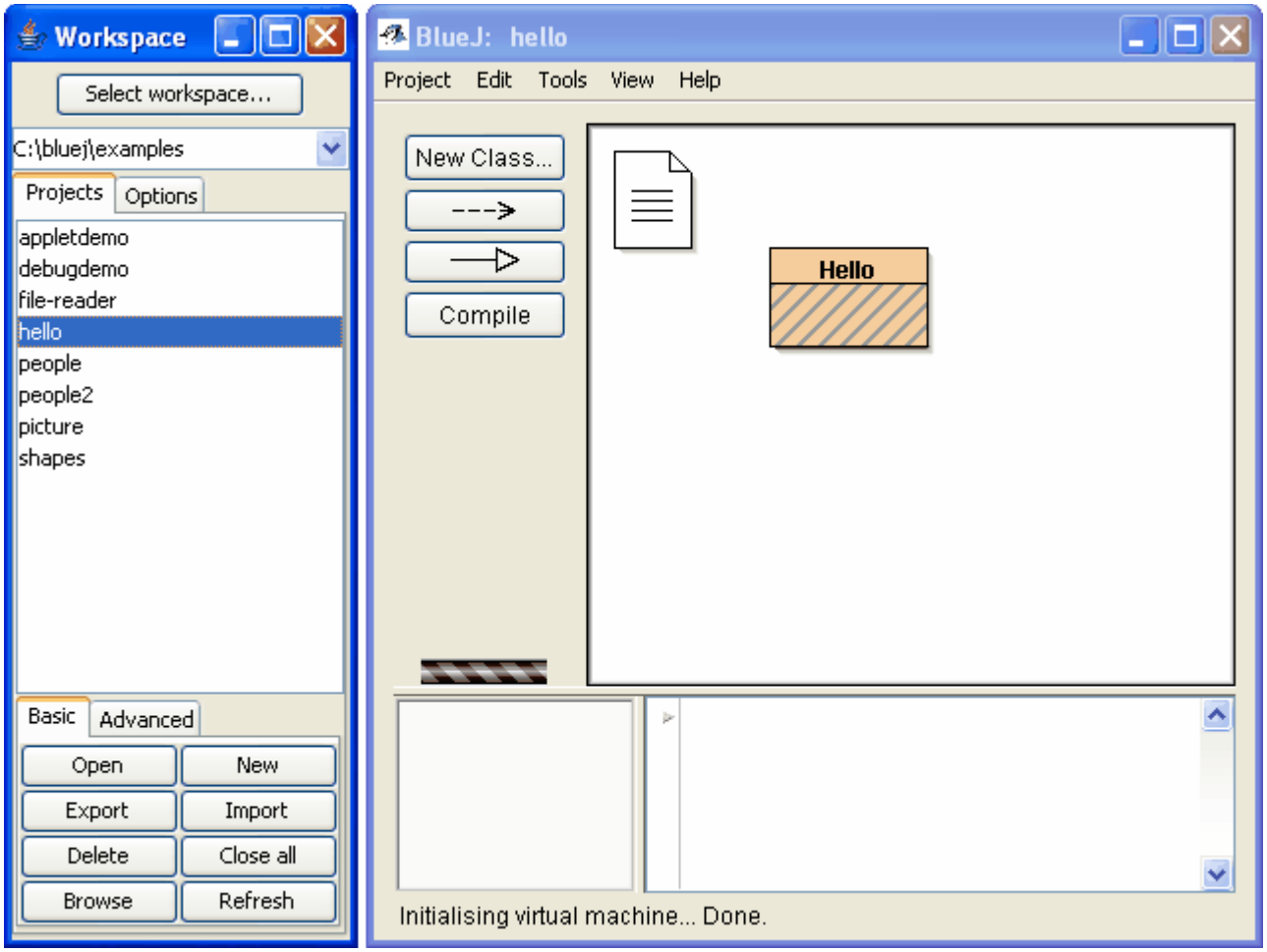
Multi-Project Workspace

Manage multiple projects within BlueJ



Version 2008.06.25

by Manuel Haim

Philipps Universität Marburg

C o n t e n t s

Preface

BlueJ is a famous integrated Java environment for learning object-oriented programming. However, when dealing with multiple projects, there is no intended way to quickly switch between them. Keeping several projects open at the same time is a resource intensive approach and will even result in cascading windows scattered all over the desktop. Teachers and teaching assistants thus spend much time in project switching (when correcting exercises), and students tend to lose track of all their projects.

Furthermore, some students are encountering problems when submitting their weekly homework. The *Submitter* extension is difficult to use and configure, it cannot be used offline, and there is no control or feedback of what exactly has been sent. Nevertheless, exporting a project as Java Archive (jar file) is complicated as well, which sometimes leads to missing source files. Adding additional files (such as images or documents) fails due to issues locating the project's directory. As the students like to be on the safe side, they send in their files and projects manually and even use proprietary archive tools to pack them.

This is where the idea of a multi-project workspace extension began to take shape. The available projects should be accessible quickly and easily and be stored within a single directory (the *workspace*). Import and export functions should help users both backup and exchange complete projects as single jar files (while still being compatible with any default BlueJ installation). Some further functionality might allow teachers to handle bunches of jar files economically.

The Multi-Project Workspace was developed during my time being a teaching assistant to Prof. Dr. H. P. Gumm for his lectures in computer science at the Philipps-University of Marburg (Germany). It finally emerged to a fully-functional and user-friendly BlueJ extension and was acknowledged as an advanced practical work.

Manuel Haim, April 2008

Software requirements for running the extension

BlueJ ≥ 2.1.3
Java Development Kit ≥ 1.4.2

Software used for developing / testing the extension

– BlueJ 2.1.3 and 2.2.1 –
BlueJ is a Java coding environment for beginners. It is being developed at the University of Kent (UK) and the Deakin University (Australia). Classes and their dependencies are visualized in a UML-like manner, their methods and constructors can be called easily via mouseclick. Generated objects are displayed on the *object test bench* for further interaction. The functionality of BlueJ can be extended by plugin-like extensions.

– bluejext.jar 2.4 (comes with BlueJ 2.1.3) –
This jar file contains all Java classes necessary for writing BlueJ extensions.

– Eclipse 3.2.1 –
The Eclipse IDE is one of the most powerful environments for developing Java applications. It has been initially developed by IBM and is an open source project. There are lots of available plugins for nearly any purpose.

– Eclipse Visual Editor 1.2.0 –
The Visual Editor is an Eclipse plugin which enables users of the Eclipse IDE to visually develop graphical user interfaces for Java applications rapidly. However, some knowledge of Swing (javax.swing) is required to get started.

– Java Development Kit 1.4.2 and 1.5.0 –
All default Java applications run on top of a virtual machine which is part of the Java Runtime Environment (JRE) by Sun Microsystems. The Java Development Kit (JDK) 1.4.2 is a set of tools and libraries needed to develop applications suitable for the JRE 1.4.2.

License

The Multi-Project Workspace extension is released under the MIT License which reads as follows:

Installation

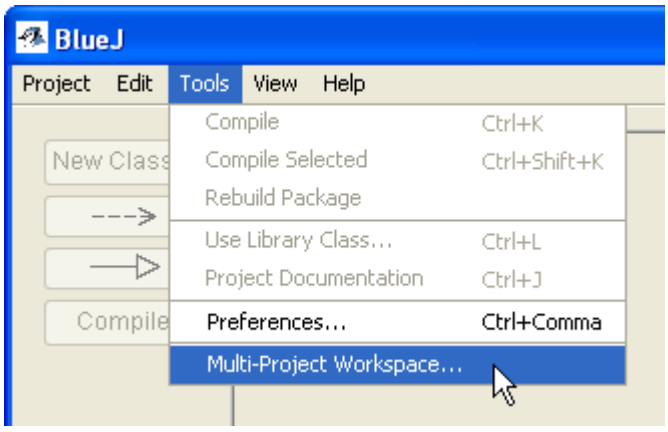
The Multi-Project Workspace extension comes as a single file (*multiproject.jar*), ready for download at *http://multiproject.sf.net* or via the *http://www.bluej.org* extension website. Copying it to the BlueJ extension directory will make it available for all users of the computer. Assuming that `<BLUEJ_HOME>` is the directory of your BlueJ installation, the extension directory is located at one of the following locations:

Unix, Linux:	`<BLUEJ_HOME>/lib/extensions`
Windows:	`<BLUEJ_HOME>\lib\extensions`
Mac:	`<BLUEJ_HOME>/BlueJ.app/Contents/Resources/Java/extensions` (Control-click *BlueJ.app* and choose *Show Package Contents*)

Alternatively, you may choose the user-specific extension directory (within the user's home directory `<USER_HOME>`) to have the extension installed for a single user only:

Unix, Linux:	`<USER_HOME>/.bluej/extensions`
Windows:	`<USER_HOME>\bluej\extensions`
Mac:	`<USER_HOME>/Library/Preferences/org.bluej/extensions`

Finally, run BlueJ and select the *Multi-Project Workspace...* entry from the Tools menu. This will pop up the extension window (or bring it to front when it is already opened).



Please note: If you quit BlueJ with the extension window open, the Multi-Project Workspace will be opened automatically on next BlueJ startup. Closing the extension window will unload the extension at any time.

Usage

– Getting started –

This quick tour will guide you through the most basic features of the Multi-Project Workspace.

1. Select a workspace

When you first use the *Multi-Project Workspace*, you must choose a directory to contain your BlueJ projects. You may create a new directory or even select an existing one.

Click the *Select workspace...* button to open a file requester and try to locate the BlueJ examples directory. Alternatively, you may simply enter *examples* in the field below. If you did it all right, the available projects will appear in the project list.

The last five workspaces will be stored in a drop-down list, so you may quickly go back to a previous location.

2. Open a project

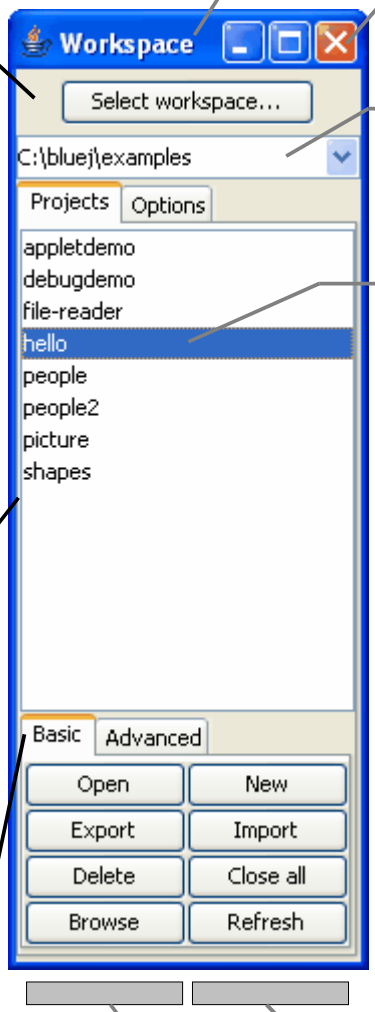
On the *Projects* panel, you will find the project list and a number of action buttons. Double-click a list item to open the corresponding project. You may instead select a list item by single-click and then click the *Open* button below.

Whenever you open a project this way, all currently open projects will be closed before. Remember that BlueJ saves all their changes automatically. This allows for easy switching between several projects.

3. Explore the basic functionality

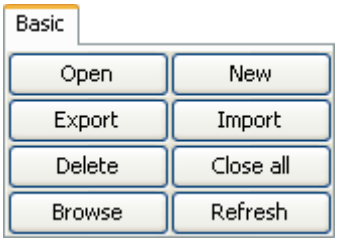
The *Basic* panel shows all functionality which you need to get started. The buttons to the left always relate to the single selected project from the project list, while the remaining buttons perform tasks relating to the project list as such.

Leave the mouse cursor over a button for some seconds to get a short description of its action. A more detailed description can be found on the next page of this document.



– Basic actions –

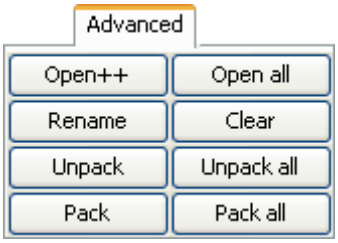
The action buttons on the *Basic* panel implement all features needed to create and handle single projects. Students will like the ability to export and import projects easily. Using the *Browse* button, they finally have quick access to a project's directory.



Open an existing project The *Open* button opens the selected project in BlueJ. All currently open projects will be closed (BlueJ saves their changes automatically). If missing, bluej.pkg files are created recursively. To quickly open a project, you may simply double-click it on the project list (this raises exactly the same action as the *Open* button).	**Create a new project** When you start off with a clear workspace (i.e. an empty workspace directory), you may want to create a new BlueJ project. To do this, simply click the *New* button. You will be prompted for a project name, then the project is created within the workspace, and finally BlueJ pops up with the newly created project (after closing the open ones).
Export a project as jar file To share projects between different computers, workspaces or BlueJ installations, you may export a complete project (with all files of the project's directory) as a single jar file (Java Archive). Just select a project, click *Export* and choose a location and filename. See the *Options* panel for further settings.	**Import a jar file as a new project** Any jar file may be imported as a new project, even Non-BlueJ ones. After clicking *Import* and selecting the jar file, you are prompted for the new project name (which defaults to the filename of the jar file). If the project already exists, you may overwrite it or cancel the action. Finally the new project will be opened in BlueJ.
Delete a project To get rid of a project, select it on the project list and click the *Delete* button. You have to confirm that you really want to delete it. Right after that, the project is closed and its directory will be deleted completely.	**Close all opened projects** The *Close all* button closes all opened projects immediately. Again, BlueJ will save all changes. This leaves you off with a tidy desktop and allows you to exit BlueJ with a single click on BlueJ's close button.
Browse a project's directory If you need to put additional files into the project's directory, try the *Browse* button. It will locate and open the project's directory by using the default directory browser. This works under MS Windows, Linux (Gnome or KDE needed) and Apple Mac. If currently no project is selected (e.g. after clicking *Refresh*), the workspace directory will be opened.	**Refresh the project list** Whenever you need to reload the workspace directory (maybe after switching disks or creating projects manually), click the *Refresh* button. Any subdirectory will be treated as a valid project.

– Advanced actions –

The action buttons on the *Advanced* panel implement some actions which may be helpful for teachers and teaching assistants. They allow them to open multiple projects simultaneously. Projects can be quickly (re-) imported or exported (this is called *Unpack* and *Pack* here) and even be renamed. The *Clear* button will eventually flush the whole workspace.



Open a project without closing the others Using the *Open++* button, you can open a second, third, fourth... project additionally to the already opened projects. Just select a project and click the button.	**Open all projects** If you are looking for a way to quickly open all available projects simultaneously, the *Open all* button is the ultimate solution. Note that you will be prompted for confirmation first.
Rename a project If you do not like a project's name anymore, there is no need to worry. The *Rename* button will allow you to enter a new name for the selected project. If the project is currently open, it will be closed and reopened (without closing other projects).	**Clear the workspace** To completely delete all projects and files in the current workspace, click the *Clear* button. After confirming the action, all projects from this workspace will be closed and any file or subdirectory will be deleted.
Unpack a project The *Unpack* button will import (but not open) the selected project from the corresponding jar file within the import directory. The import directory and further settings can be set on the *Options* panel. If the project's directory already exists, you may confirm to overwrite it completely or cancel the action. Note that non-existing importable projects are marked by an asterisk (*) in the project list.	**Unpack all projects** To quickly import all projects from the import directory, use the *Unpack all* button. You will be asked only once to overwrite all existing projects, to just import the non-existing projects, or to cancel the whole action before anything has happened. See the description of the *Unpack* button for a clear definition of the import process.
Pack a project The *Pack* button will create a jar file from the selected project and store it within the workspace. You can use this to quickly backup your project. If the jar file already exists, you will be asked to overwrite it or not. See the *Options* panel for further settings.	**Pack all projects** The *Pack all* button will create jar files from all projects in the workspace and store them right there. At first, you will be asked if existing files should be overwritten or not, or if you would like to cancel the action. See the *Options* panel for further settings.

– Optional settings –

The functionality of the action buttons is influenced by the settings on the *Options* panel. These settings are saved on a per-workspace basis.

Unpack options

The *Unpack* and *Unpack all* action buttons only work after an import directory has been set. You can do this here by clicking the *Set import dir...* button, or just enter the path directly into the field below.

The import directory should contain some jar files which will be interpreted as packed projects. For example, a file called *test.jar* will be imported as project *test*. Again, even non-BlueJ jar files are allowed.

To rename projects automatically, you may put the corresponding jar file into a project-specific subdirectory of the import dir and choose to *unzip *.jar* from there. Assuming your subdirectory is called *test*, then all jar files contained therein will be imported to a single project *test*.

Furthermore, to import non-packed BlueJ projects or to simply add additional files, you may choose to *copy files* from project-specific subdirectories. This will copy the subdirs' remaining files and directories recursively.

Export / Pack options

Whenever a project is exported or packed to a single jar file, this will bundle just everything from the project's directory. To leave out some files, you may choose to do so here.

Check *exclude javadoc* to ignore the javadoc files (i.e. the *doc* directory).

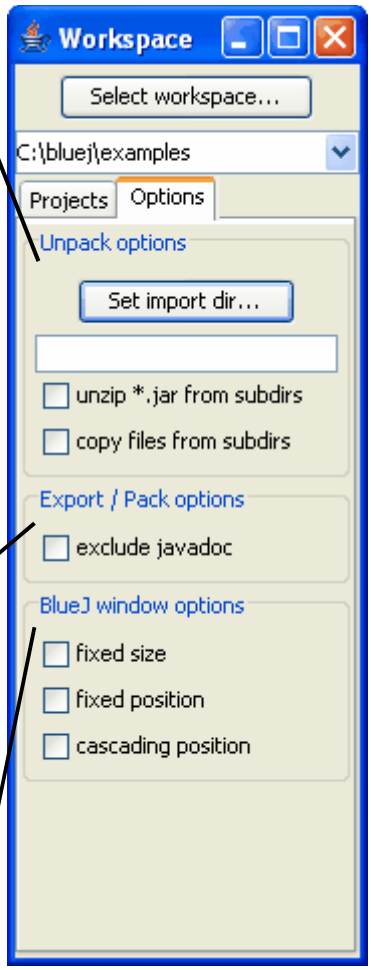
BlueJ window options

When you open unknown projects or create new ones, BlueJ's package windows seem to pop up just anywhere. These options allow you to control this behaviour.

Check *fixed size* to make all newly opened package windows have the same size as the recently opened or closed package window.

The *fixed position* checkbox will make all newly opened package windows appear in the same position.

If *cascading position* is checked, the position of newly opened package windows is set according to their hierarchy level (i.e. subpackage windows are displayed indented relatively to their parent package).



Usage scenarios

– Simply manage multiple projects –

Assume a user simply wants to keep track of his projects. He installs the Multi-Project Workspace extension and starts off with an empty workspace. Clicking the *Select workspace...* button, he navigates to his home directory, creates a *bluejworkspace* directory right there and chooses this to be his personal workspace.

Instead of using BlueJ's *Project* menu, the user clicks the *New* button each time he wants to create a new project. This will guarantee that all projects are stored in the same place and can be easily found again. When he has finished his work, he quits BlueJ as usual, and the Multi-Project Workspace will pop up automatically on his next BlueJ session.

– Doing homework –

When a student has to do and submit his weekly homework, the Multi-Project Workspace can be very helpful. The student creates a new project for each exercise sheet, e.g. *sheet3*. If he wants to add a jpeg image file, for example, the *Browse* button will help him to quickly locate the project's directory. Using the *Export* button, he finally creates a single Java Archive (jar file). Then he may use his favourite e-mail client (or even a floppy disk or USB memory stick) to submit this file.

After receiving the revised version of his homework, the *Import* button will allow him to easily integrate it into his workspace.

– Revising the students' homework –

A teacher may use the advanced actions to quickly import jar files when revising all the students' homework. He creates a workspace for the current exercise sheet (e.g. *sheet3*) and stores the received jar files in another directory (e.g. *sheet3import*) under slightly different names (e.g. *group2sheet3.jar* or simply *02.jar*). When setting this directory as *import directory* within the *Options* panel, the jar files will become available within the project list, marked by an asterisk (*). These projects are imported automatically when opened for the first time, or using the *Unpack* and *Unpack all* buttons on the *Advanced* panel.

The idea of the *import directory* is to have a separate directory which will not be altered or deleted accidentally by the Multi-Project Workspace. Of course, the workspace directory may be used for these imports, either.

To annotate pieces of code in an outstanding way, BlueJ already offers its *stand-out comments*. Rather than using default Java comments, teachers may enter annotations beginning with `/*#` and ending with `*/` which will usually be displayed in a pink color.

After finishing the revision of all homework, the teacher can use the *Pack all* button to create jar files of all projects. These will be stored within the workspace directory. He then may use his e-mail client or some type of PHP script to deliver them back to the students.

– Revising the students' homework (part 2) –

To open and compare multiple projects at the same time, a teacher may use the *Open++* and *Open all* buttons on the *Advanced* panel. This opens one or all projects additionally to the already opened projects. The *Close all* button will finally close all projects.

– Revising the students' homework (part 3) –

Sometimes students submit several single files, e.g. when having trouble with the jar file export or when some work is done separately (like documents or drawn images). To import these along with the jar files of the *import directory*, the teacher may create subdirectories right there and put the single files into them. He just needs to set the *copy files from subdirs* option on the *Options* panel. Like the jar files, the subdirectories must be named after the projects they stand for (e.g. *group2sheet3*). Please note that the *import directory* must be different from the workspace directory now.

When checking the *unzip *.jar from subdirs* option on the *Options* panel, the students' jar files may even be put into the subdirectories, so there is no need to rename them.

If there are multiple sources for a single file (e.g. if there is a file *Readme.txt* within the jar file as well as within the subdirectory), the target file will be written only once. Depending on the way Java handles your file system, the sources may happen to be in alphabetical order or not.

Troubleshooting

– When selecting a workspace on MS Windows, I cannot create a new directory –

The file requester on MS Windows XP hides the *Create folder* icon while navigating through special folders like the *My Documents* folder. This seems to be a Java Runtime Environment issue. Try to create a directory from within Windows Explorer or using your favourite file manager.

– My project cannot be renamed –

When renaming a project, please ensure that no other program is accessing files or directories from within the project's directory. If you saved a document from e.g. a word processor right there, try to exit that piece of software. Close any directory browser window pointing to the project's directory. When programming I/O routines, make sure to close all files after reading/writing. Finally, if none of these works, you may try to quit and restart BlueJ.

– The "fixed size" and "fixed / cascading position" options do not work correctly –

BlueJ sends an event whenever it opens or closes a package. However, this event is sent *before* BlueJ creates and positions the package frame. Thus, a thread is needed to set the frame constraints up to ten times and verify them soon after. This proved to work on all platforms, but might lead to errors under certain circumstances like heavy system load.

The sole exception was Gnome / KDE on Linux when using *Compiz Fusion* as window manager. Under this setup, the Java Runtime Environment cannot determine the window position correctly.

– Within a package window, some packages keep to be selected –

If you switch to another package by double-clicking it in the BlueJ window, the package contents will pop up. However, the package is then selected in the previous window and will remain selected after closing and re-opening that window. This is a BlueJ error. Try to deselect the package using Ctrl + mouseclick.