

# Customising BlueJ for your students

Ian Utting



# Customising BlueJ

Obviously, you can now customise to your heart's content, but there are easier ways.

Simple stuff (adding libraries, controlling editor font size) for single users is done through the Preferences Panel.

Less common, or global, stuff is done either through template files in the BlueJ installation, or BlueJ system & user properties stored in the files `bluej.defs` and `moe.defs` (system), and `bluej.properties` and `moe.properties`. Each contains lists of properties (`key: value`). User properties override system properties.

# Languages and error messages

The BlueJ user interface is currently available in 17 languages. See `lib/language`. Change which one is used by setting the `bluej.language` property.

You can change how BlueJ describes compile-time errors, or runtime exceptions by modifying the “glosses” in: `lib/language/javac.help` and `exception.help`

In these files, the first line is the error message which is matched (a `*` at the start and/or the end allows partial matches). The rest (until the next blank line) is the associated gloss.

# Code templates

You can control what appears in the New Class dialog box by creating templates in

`language/templates/newclass/name.tmpl`

- If *name* starts with “abstract”, “interface” or “applet”, any class created will have the appropriate UML stereotype.
- Create a property `pkgmgr.newClass.name`: Menu item in the language files.
- Add *name* to the `bluej.classTemplates` property to control order-of-appearance. Otherwise, it will still appear, but in a random order.

# JVM and compiler arguments

BlueJ runs two separate JVMs. One for itself (including the compiler and any extensions) and one for user-code, including objects invocation and the codepad. They are governed by separate properties:

- BlueJ itself: via `bluej.windows.vm.args` on Windows (picked-up by the launcher), but via the launcher batch-script on other platforms
- User code: via `bluej.vm.args`

The Java compiler is governed by a separate property: `bluej.compiler.options`

# Adding your own libraries and help menu items

Libraries can either be coupled explicitly with BlueJ:

- per-project in a folder `+libs` in the project folder
- per-installation in `lib/userlib` (you can override that location with the `bluej.userlibLocation` property)

Or referenced externally (watch out, your students might not have them installed) by setting `bluej.userlibraryn.location`

Add a tag to the property `bluej.help.items`, then create `bluej.help.label.<tag>`: Menu entry and `bluej.help.url.<tag>`: URL

# Repackaging your mods for distribution

If you have the source, you can always `ant dist` and ship the resulting `package/bluej-dist-new.jar`, or even re-build the Windows installable package.

Alternatively, crack open the standard distribution JAR file (e.g. `bluej-2.5.0.jar`), then crack the `bluej-dist.jar` file it contains. Make your changes and re-pack in reverse order.