Java Keybinding Chooser and Editor

Author: Damian Johnson (atagar1@gmail.com)
Last Modified: 9/28/07
Compiled with Java version 1.6.0_02

This project consists of two tools for the use of customizable key bindings in Java applications. The first is an editor for creating or changing lists of bindings to be loaded from applications. The second is a chooser that can be used within applications to provide Swing dialogs for customizing the bindings. Keeping to the Unix philosophy I'm hoping for this to do one thing, and do it well. Please email me any bug reports, feature requests, or even notes saying if it's helpful or not.

Java Keybindings

First of all a quick explanation of Java key bindings may be in order. If you are already aware of keyboard listeners and input maps you can safely skip this section.

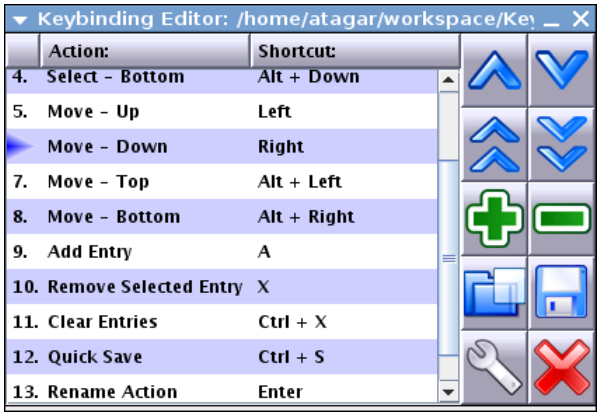
Bindings are generally defined as mappings of keystrokes to strings representing the actions they perform. Since this is a mapping the editor and chooser prevent duplicates among the keyboard shortcuts.

Java provides two methods for detecting keyboard events:
1. Key Listeners (java.awt.event.KeyAdaptor)
Listeners can be attached to components to receive keyboard events. Unfortunately this can be rather tricky since it requires that all focusable components have a copy of the listener. This is the only way of capturing arbitrary keystrokes.

2. Input Maps (javax.swing.InputMap)
This is the preferred method, avoiding much of the focus complications. The JComponent's InputMap provides mappings between keystrokes and objects (usually strings), then an ActionMap maps between those objects and code to be executed.

For more information see Sun's explanation at:
http://java.sun.com/docs/books/tutorial/uiswing/misc/keybinding.html

Keybinding Editor

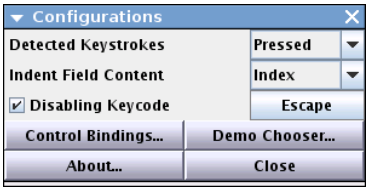


The editor is a tool for making or changing lists of key bindings that can be used within applications. Shortcuts can be clicked to edit their bindings and actions can be changed by double clicking on them. The rest of the controls are obvious with a little experimentation.

The folder and disk buttons are for loading and saving respectively. This editor supports several types of standard Java persistence including:

- Serialized linked or normal hash maps
- Serialized input maps
- Properties mapping (java.util.Properties)- either as plain text or XML

If your application is only using a small list of bindings then this can also output the source code that generates the bindings (skipping the fuss of dealing with persistence). Please note that the only format that preserves ordering is serialization with a linked hash map.



The wrench button provides the configuration controls for the editor, which gives the following options:

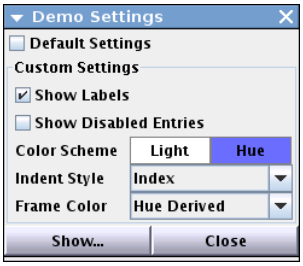
Detected Keystrokes- Type of keyboard event detected when setting shortcuts (see java.awt.event.KeyEvent for what this means).

Indent Field Content- Type of information provided in the left column.

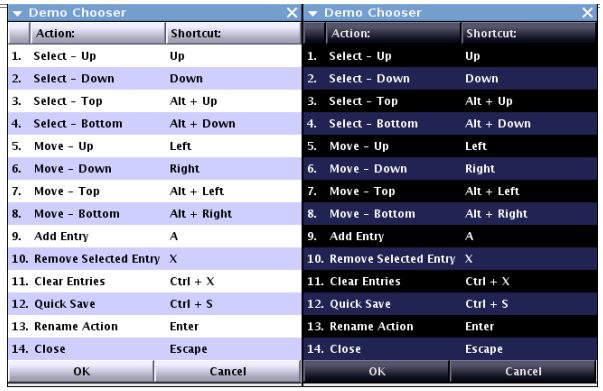
Disabling Keycode- Defined if and what key sets an undefined mapping when entered for the shortcut. Currently none of the persistence formats support undefined mappings.

Control Bindings- Keyboard shortcuts used by the editor itself.

Demo Chooser- Shows an example of how the current bindings would appear in a chooser dialog. This first gives the dialog to the right, providing options for how the chooser is displayed.



Keybinding Chooser



The chooser is a UI utility that can be included in applications to provide to provide dialogs for changing the bindings (much as the JColorChooser allows the selection of colors). The dialog's appearance is fairly customizable, the image to the right showing choosers with a light and dark blue color scheme. The javadocs for these classes are rather extensive I'll just give an overview of the classes involved:

chooser.BindingEntry- Single row of the display, acting both as a display element and definition of a key binding.

chooser.BindingPanel- Abstract panel providing most utility functions for a list of bindings (adding, clearing, getting the map, etc). Its abstract components are methods to handle the functionality when the contents change or clicks are detected.

chooser.BindingAdaptor- KeyAdaptor used in the BindingChooser to detect keyboard events for changing the shortcuts.

chooser.BindingChooser- This is the implementation of the BindingPanel providing the ability to customize the shortcuts, display dialogs, and make sweeping changes to the appearance.

chooser.Persistence- Convenience methods for saving and loading key bindings.