

# Re: Great SWT Program

---

*Source:* <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2007-12/msg00326.html>

---

- *From:* [nebulous99@xxxxxxxxxx](mailto:nebulous99@xxxxxxxxxx)
  - *Date:* Mon, 3 Dec 2007 00:09:16 -0800 (PST)
- 

On Nov 30, 6:39 am, Andreas Leitgeb <a...@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

When in "normal" editing-mode, emacs also removes the char left to the cursor. It's just when in incremental-search-mode, that it interprets the "find-next" as a virtual "keypress", which can be backspaced, as well.

Which is, of course, the violation that I was complaining about originally. Invisible extra things to backspace away are going to trip the user up horribly. They have no way to know what backspace will do at any given moment, unless they've kept exact count of both next-match and backspace presses. More mental arithmetic forced on the user! On top of that, there's no way to go to the third match for "fob" and then jump to the next match for "foo"; if you try to backspace the "b" and type an "o" it behaves wonkily and violates the Principle of Least Astonishment instead. You could actually cancel the search and then do a fresh search for "foo" from there, but of course that's awkward and clunky.

From my own moderate experience with windows, and from what I see others do in windows, the in-place renaming must be quite a nuisance: Whenever you double-click a file too slowly, you end up editing it's name rather than it's content (in the otherwise started editor) If the renaming-feature was invoked by F2 only, then indeed it's a convenient way to correct typos in filenames easily.

I think this is actually a bug. I've seen this too, where double-clicking a file does nothing, and the file subsequently behaves as if F2 had been pressed, but the double-click may have been "fast enough" and it may still happen, and the F2-behavior does not occur the instant the second click does; it occurs after a fraction of a second has passed. This indicates that Windows is not seeing the same input that the user is actually generating, from time to time. Some of the drag and drop oddities in Explorer indicate a similar problem; files moved to or created in a directory "behind Explorer's back" are

## Re: Great SWT Program

normally placed at the end of the file list, but ones dropped into the directory window are normally inserted where dropped. Those few odd instances of a dropped file jumping to the end of the list are easily explained if one part of Explorer "saw" the drop operation and told the underlying filesystem to move the file, but another part didn't and fails to update the file list due to the drop. Explorer then notices "on its own" that the file got moved and appends it to the end of the list. In fact I think it's a race condition: if the file-moved notification from the OS reaches Explorer before the drop operation list update is done, the file-concurrently-moved list update executes and puts the file at the end; then the drop operation list update silently does nothing because the file's already in the list. I've seen other Explorer oddities explained only by input going unnoticed however: deleted files not disappearing for a while, and then Explorer "suddenly notices" that "another process" concurrently deleted the file; ditto with moved files; and renamed files instantly reverting to their old name with no error message, especially when the name change was only to the capitalization, so the filesystem regards the file as unchanged.

These bugs in Explorer are, of course, evidence of Explorer having problems, rather than evidence of graphical file managers being inherently evil or inherently broken. I don't believe the latter to be true at all.

By the way, file-managers don't need to be graphical. If you happen to know "Norton commander" back from the DOS-ages, you know what I mean, and it has a very powerful successor/counterpart in unix-world called "mc" (midnight commander).

DOSShell comes to mind. It was basically a text-mode forerunner to Windows 3.1's File Manager, which was a forerunner to Explorer (turn on the tree pane at left in an Explorer window to get a full 3.1-File-Manager-analogue in terms of functionality).

I expect Norton Commander and Midnight Commander to have text-mode interfaces that are natural and sensible to use, the way MS-DOS Edit and DOSShell did and the way vi and emacs don't. :) The latter's name suggests that it's specifically a clone of the former, and the former, as a late MS-DOS era app, probably has the same sort of "TUI" found in Edit, QBasic (which actually shared Edit's UI engine code) Borland's IDE from that time period, its open source clone RHIDE, and many other MS-DOS applications. Edit's interface is almost identical to Notepad's, to the extent that it possibly can be when the one is text mode and the other graphical. The only things basically missing from Edit are move, resize, minimize, maximize, and close controls, all for the obvious reasons aside from the lack of a close control.

## Re: Great SWT Program

Outside of "mc", my procedure for renaming such files is to enter them once (with help of Tabbing) after the "mv", then press [unobvious keystrokes that are probably specific to a particular shell] to duplicate it on the commandline, then go back with cursorkeys and edit the second argument. Not \*that\* bad.

Still clunky, and dependent on rather esoteric knowledge of obscure shortcut keys of your specific shell. The Windows equivalent being to type something, then hit shift-home shift-del shift-ins shift-ins, which will work in any text input area in any Windows app just about. (C-c or C-x followed by C-v C-v also works, or mixing them, but if you just hit shift-home shift-del and shift-ins are probably more convenient.) No app-specific bindings or other wonkiness, but equally clunky and usually avoidable. Downside: does not work at an MS-DOS Prompt in a console window, which can't in good conscience eat any of the clipboard bindings after all. A smarter command prompt app would fix this -- it could safely intercept the clipboard keys when at a command prompt instead of a different task; it just doesn't. They made it smart enough to exit the command prompt when close-box closed instead of complaining about ending the process mid-stream around the time Windows XP replaced Windows 98/ME, while still complaining if you click the close box with some console-mode process running in it other than the command interpreter, after all. It's actually strange that they didn't add decent clipboard support to the command interpreter at the same time.

.