

# Fuel for the Fire

Casey Marshall

*Or, yet another ill-informed opinion about free Java*

*This is a rough draft, which I haven't had the time or capacity of mind to finish. Some of what I've said here still makes a good point, however, and some of it is kind of funny, so maybe someone else can build upon it.*

IT IS POSSIBLY not entirely helpful of me to publish these opinions about Java, the programming language, given the noise about the subject that has filled the press lately. The debate, such as it is, has remained juvenile throughout, since Eric Raymond's open letter [1] first started appearing in the news and peoples' weblogs, through the mumbling-under-their-breath of IBM, through Sun's own wishy-washy stance over the whole matter. So this tract of mine may simply be another voice in this whole kindergarten affair — something to yell out into a cacophony only to become embarrassed about on hindsight.

But since I have opinions about this matter, and since I do think they are important enough that not saying them now might be a bigger source of regret, here comes nothing.

## About the Author

I am a brash, naïve young software engineer who lives in California. I write Java code for a living, and have written code for numerous free Java projects, the most notable being GNU Crypto [2], a full-featured cryptographic library, and Jessie, a Java implementation of the secure sockets layer, SSL [3]. I enjoy hacking on free Java runtimes, like Kaffe [4] and GCJ [5], in my spare time, when quite honestly there are somewhat more interesting things my twenty-something body and I could be out doing.

I see a great value in free software. More than anything else, it has allowed me to collaborate with some brilliant people who live all over the world, who, like me, find a unique beauty in computer programming. Software freedom then is, above all, a means to sustain this community, which exists outside the bounds of corporations and countries. It isn't freedom-for-freedom's-sake, but the real benefits of community and collaboration that make free software important.

In Java I've found a great platform for exploring computer science. For me it solved most of the problems with C and C++, and it didn't sacrifice simplicity and elegance to do so. It is worth building and maintaining a community around Java.

## Spaghetti with a Spoon, or My Irrational Fear of Forks

The only real argument Sun has made against making Java free software — aside from the asinine “how could it *be* any more open” [6] — is that an open source version of Java would immediately be forked by everyone, and suddenly all the applications ever written in Java would become incompatible, and the world would collapse.

One wonders when reading such statements whether or not Sun has ever seen how an important free software project is run. How many forks of GCC have been made that destroyed compatibility of C programs? Or how about Perl? Even possibly the most traumatic fork of free software — Emacs and XEmacs — only resulted in two independent implementations that continually improve one another.

In free software, a fork is almost universally a reaction against making the software non-free. See, for example, the immediate fork that was made of the XFree86 project’s implementation of the X window system as soon as they made the license more restrictive. A fork is merely a community preserving itself, routing around damage. If someone makes a fork of your software, it is likely because of something selfish you have done, and probably something you deserve. If there are no problems like this in a software package, there is no reason to fork it, because a fork *would* result in binary incompatibility, which would make such a fork irrelevant given the body of software that works the canonical version. In short, a fork can be a guaranteed failure if it is incompatible, so it is only a last resort.

Of course there are valid reasons for wanting to “fork” a piece of software: adapting it to fit a particular need. What if someone wanted to write an AWT and Java2D implementation that ran, for example, on the Y windowing system? Would that be a fork, and if so, would it be a bad thing? Or a new JIT for a new architecture, or a new garbage collector? All of these would *enrich* the Java platform, not degrade it. If anything, forked versions of Java would only enhance it across all platforms, because being incompatible is not valuable.

Companies and groups all over are writing extension libraries for the Java language, and programs are being written that will only work with the combination of the Java standard libraries and these extension libraries. Such programs *aren’t* using a forked version of Java, however.

One last rhetorical question on this issue: IBM distributes a JDK that implements the Java 2 Standard Edition, version 1.4, which is a fork of Sun’s code base. They distribute with it an implementation of the Mars cipher they designed for the AES process, and Sun does not. Is this not already a binary incompatibility? If so, then why doesn’t Sun have an issue with this, and stop IBM from distributing it?

## The Java Community, Part 1: Choose any Color as Long as it is Blue

Personally, I don’t care that much about Sun’s reference implementation of Java. In the end, any particular implementation of Java is irrelevant itself; the language and the bytecode format will be around longer than the JDK Sun distributes, and it will likely outlive Sun Microsystems as they exist today. GNU Classpath and the runtimes that

use it are maturing fast enough that Sun's sources aren't essential, or even important, for their success. What is essential is ensuring that these independent implementations are compatible with the standard, and this is only possible if the standards are freely available. The terms under which the Technology Compatibility Kit is available are not compatible with free software licenses such as the GPL, making a free implementation of Java that *has been proven compatible with the specification* impossible.

Sun made a decent gesture with bringing Java out into the open with their Java Community Process, but the JCP is hardly ideal. It may invite outside opinions on the direction and shape that the Java language will take, but the agreement contributors need to sign has so many exceptions and holes for Sun that contributors have no teeth when defining new specifications. There is little bargaining room for everyone else. Such a "community" is meaningless, as long as one player has the biggest stick and everyone knows it.

This is an earmark of Sun's approach to free software: that it is a source of free labor that requires no investment other than the privilege of working on their code. Richard Gabriel and Bill Joy's memo on the subject, *Sun Community Source License Principles* [7], is so appallingly cynical that it purports that not only are proprietary and open-source licensing models compatible, but that under such a combination *anyone except Sun sees tangible benefits*.

I strongly believe that Sun's reticence with opening Java up is *not* compatibility; no-one, aside from players as powerful and domineering as Microsoft, would find benefit in making an incompatible version of Java. Sun's problem is control. They have control of a powerful commodity, and don't want to relinquish it.

## Wooden Nickels

All of the recent noise surrounding this issue is due to misunderstanding what the free software community wants out of Java, and I blame [1] in particular for this. We must seem like an ungrateful bunch, Sun investing so much into creating and maintaining Java, and we throw it back in their face because it isn't free enough! Who are we, anyway, to demand Sun's source code?

Like I said above, I do not want Sun's source code. I would rather let it go on its own merry way, and lend my hand to help GNU Classpath and free runtimes prosper. Why, then, would I be dissatisfied with Sun's conduct with Java? Isn't it enough that they created and published a great programming language, and let everyone download the binaries and source?

To paraphrase [8], *I can love only what I value*. Since Sun will not release a free version of Java themselves, and because they have made free, independent versions of Java impossible, I find nothing I value in them. Laying inside a proprietary Java is the death of the community growing around it. If we cannot be certain that Java will continue to be available under the sufficiently non-discriminatory terms it is now, nor if Sun Microsystems will not try to undermine the work of others by enforcing rights over its "intellectual property" [9], we cannot call Java a solid foundation. If it were made free — for now and ever — it would be a different story.

I tend to focus on the notion of *value* quite a lot, and it has informed my outlook on issues such as this. W

## **The Java Community, Part 2: And Some People Still Think the Monarchy is a Good Idea**

A man came wandering up to a cottage in the middle of a rainstorm. He banged on the door, yelling to be let in out of the torrential rain. The occupant of the cottage merely scoffed at this, and told the man outside that it could not be raining, because he was as dry as a bone.<sup>1</sup>

Particularly unhelpful throughout this entire discussion have been the hordes of Java “developers” who swarm about internet discussion boards, parroting Sun’s party line about forks and incompatibility.

Copyright © 2004 Casey Marshall

This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/2.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

---

<sup>1</sup>I’m pretty sure this is based on something, but I can’t recall what. Pls remind me if you know.

## References

- [1] Eric S. Raymond. Open Letter to Sun: Let Java Go.  
<http://www.catb.org/~esr/writings/let-java-go.html>.
- [2] GNU Crypto. <http://www.gnu.org/software/gnu-crypto/>.
- [3] Jessie: A Free Implementation of the JSSE.  
<http://www.nongnu.org/jessie/>.
- [4] The Kaffe OpenVM. <http://www.kaffe.org/>.
- [5] GCJ: The GNU Compiler for Java. <http://gcc.gnu.org/java/>.
- [6] James Gosling. Open sourcing java.  
<http://today.java.net/jag/page7.html#62>.
- [7] Richard P. Gabriel and William N. Joy. Sun Community Source License Principles. <http://www.sun.com/981208/scsl/principles.html>.
- [8] David Foster Wallace. *Girl With Curious Hair: Westward the Course of Empire Takes Its Way*, page 234. W. W. Norton & Company, 1989.
- [9] Dalibor Topic. Pulling a SCO for fun and profit.  
<http://www.advogato.org/person/robilad/diary.html?start=49>,  
also <http://www.javablogs.com/ViewEntry.action?id=143248>.