Free but Shackled: The Java Trap

GNU philosophy

Since this article was first published, on 12 April 2004, Sun has relicensed most of its Java platform reference implementation under the GNU General Public License, and there is now a free development environment for Java. Thus, the Java language as such is no longer a trap.

You must be careful, however, because not every Java platform is free. Sun continues distributing an executable Java platform which is nonfree, and other companies do so too.

The free environment for Java is called IcedTea; the source code Sun freed is included in that. So that is the one you should use. Many GNU/Linux distributions come with IcedTea, but some include nonfree Java platforms.

To reliably ensure your Java programs run fine in a free environment, you need to develop them using IcedTea. Theoretically the Java platforms should be compatible, but they are not compatible 100 percent.

In addition, there are nonfree programs with "Java" in their name, such as JavaFX, and there are nonfree Java packages you might find tempting but need to reject. So check the licenses of whatever packages you plan to use. If you use Swing, make sure to use the free version, which comes with IcedTea.

Aside from those Java specifics, the general issue described here remains important, because any nonfree library or programming platform can cause a similar problem. We must learn a lesson from the history of Java, so we can avoid other traps in the future.

This essay was first published on http://gnu.org, in 2004.

Free but Shackled: The Java Trap

If your program is free software, it is basically ethical—but there is a trap you must be on guard for. Your program, though in itself free, may be restricted by nonfree software that it depends on. Since the problem is most prominent today for Java programs, we call it the Java Trap.

A program is free software if its users have certain crucial freedoms. Roughly speaking, they are: the freedom to run the program, the freedom to study and change the source, the freedom to redistribute the source and binaries, and the freedom to publish improved versions. (See *The Free Software Definition*.) Whether any given program in source form is free software depends solely on the meaning of its license.

Whether the program can be used in the Free World, used by people who mean to live in freedom, is a more complex question. This is not determined by the program's own license alone, because no program works in isolation. Every program depends on other programs. For instance, a program needs to be compiled or interpreted, so it depends on a compiler or interpreter. If compiled into byte code, it depends on a byte-code interpreter. Moreover, it needs libraries in order to run, and it may also invoke other separate programs that run in other processes. All of these programs are dependencies. Dependencies may be necessary for the program to run at all, or they may be necessary only for certain features. Either way, all or part of the program cannot operate without the dependencies.

If some of a program's dependencies are nonfree, this means that all or part of the program is unable to run in an entirely free system—it is unusable in the Free World. Sure, we could redistribute the program and have copies on our machines, but that's not much good if it won't run. That program is free software, but it is effectively shackled by its nonfree dependencies.

This problem can occur in any kind of software, in any language. For instance, a free program that only runs on Microsoft Windows is clearly useless in the Free World. But software that runs on GNU/Linux can also be useless if it depends on other nonfree software. In the past, Motif (before we had LessTif) and Qt (before its developers made it free software) were major causes of this problem. Most 3D video cards work fully only with nonfree drivers, which also cause this problem. But the major source of this problem today is Java, because people who write free software often feel Java is sexy. Blinded by their attraction to the language, they overlook the issue of dependencies and fall into the Java Trap.

Sun's implementation of Java is nonfree. The standard Java libraries are nonfree also. We do have free implementations of Java, such as the GNU Compiler for Java (GCJ) and GNU Classpath, but they don't support all the features yet. We are still catching up.

If you develop a Java program on Sun's Java platform, you are liable to use Sun-only features without even noticing. By the time you find this out, you may have been using them for months, and redoing the work could take more months. You might say, "It's too much work to start over." Then your program will have fallen into the Java Trap; it will be unusable in the Free World.

The reliable way to avoid the Java Trap is to have only a free implementation of Java on your system. Then if you use a Java feature or library that free software does not yet support, you will find out straightaway, and you can rewrite that code immediately.

Sun continues to develop additional "standard" Java libraries, and nearly all of them are nonfree; in many cases, even a library's specification is a trade secret, and Sun's latest license for these specifications prohibits release of anything less than a full implementation of the specification. (See `http://jcp.org/aboutJava/communityprocess/JSPA2.pdf` and `http://jcp.org/aboutJava/communityprocess/final/jsr129/j2me_pb-1_0-fr-spec-license.html` for examples.)

Fortunately, that specification license does permit releasing an implementation as free software; others who receive the library can be allowed to change it and are not required to adhere to the specification. But the requirement has the effect of prohibiting the use of a collaborative development model to produce the free implementation. Use of that model would entail publishing incomplete versions, something those who have read the spec are not allowed to do.

In the early days of the free software movement, it was impossible to avoid depending on nonfree programs. Before we had the GNU C compiler, every C program (free or not) depended on a nonfree C compiler. Before we had the GNU C library, every program depended on a nonfree C library. Before we had Linux, the first free kernel, every program depended on a nonfree kernel. Before we had BASH, every shell script had to be interpreted by a nonfree shell. It was inevitable that our first programs would initially be hampered by these dependencies, but we accepted this because our plan included rescuing them subsequently. Our overall goal, a self-hosting GNU operating system, included free replacements for all those dependencies; if we reached the goal, all our programs would be rescued. Thus it happened: with the GNU/Linux system, we can now run these programs on free platforms.

The situation is different today. We now have powerful free operating systems and many free programming tools. Whatever job you want to do, you can do it on a free platform; there is no need to accept a nonfree dependency even temporarily. The main reason people fall into the trap today is because they are not thinking about it. The easiest solution to the problem is to teach people to recognize it and not fall into it.

To keep your Java code safe from the Java Trap, install a free Java development environment and use it. More generally, whatever language you use, keep your eyes open, and check the free status of programs your code depends on. The easiest way to verify that a program is free is by looking for it in the Free Software Directory (`http://fsf.org/directory`). If a program is not in the directory, you can check its license(s) against the list of free software licenses (`http://gnu.org/licenses/license-list.html`).

We are trying to rescue the trapped Java programs, so if you like the Java language, we invite you to help in developing GNU Classpath. Trying your programs with the GCJ Compiler and GNU Classpath, and reporting any problems you encounter in classes already implemented, is also useful. However, finishing GNU Classpath will take time; if more nonfree libraries continue to be added, we may never have all the latest ones. So please don't put your free software in shackles. When you write an application program today, write it to run on free facilities from the start.