The Practice of Informatics JAMIA

Brief Review ■

Java and Its Future in Biomedical Computing

R. P. C. RODGERS, MD

Abstract Java, a new object-oriented computing language related to C++, is receiving considerable attention due to its use in creating network-sharable, platform-independent software modules (known as "applets") that can be used with the World Wide Web. The Web has rapidly become the most commonly used information-retrieval tool associated with the global computer network known as the Internet, and Java has the potential to further accelerate the Web's application to medical problems. Java's potentially wide acceptance due to its Web association and its own technical merits also suggests that it may become a popular language for non-Web-based, object-oriented computing.

■ **JAMIA.** 1996;3:303–307.

Revolutions are exciting, messy, and dangerous. However, they also present an opportunity to people who are prepared for them. Some would deny that the current frenetic activity surrounding network-based computing is a revolution, but very few would deny that it is accompanied by the requisite amounts of excitement, mess, danger, and opportunity. At the center of the current tangle of technical virtuosity, dashed hopes, commercial hyperbole, and new, dying, or reanimated technologies sits a new computer language known as Java.[1] Java's current prominence and fate are strongly tied to those of the global computer network known as the Internet and to the Internet's most widely used application to date, the World Wide Web (WWW).[2] It is impossible to consider Java properly without discussing its context.

Affiliation of the author: U.S. National Library of Medicine, Bethesda, MD.

Correspondence and reprints: R. P. C. Rodgers, MD, Computer Science Branch, Lister Hill National Center for Biomedical Communications, U.S. National Library of Medicine, Bethesda, MD 20894. e-mail: rodgers@nlm.nih.gov

Received for publication: 4/17/96; accepted for publication 5/17/96.

Developed by James Gosling and colleagues at Sun Microsystems in about 1990, Java was conceived as an object-oriented C++-like programming language that would be embedded within such consumer electronics products as "personal digital assistants" (PDAs). With the commercial failure of the Apple Newton PDA and its kin, attention turned to interactive television in 1992. When video-on-demand services failed to generate commercial excitement, the developers shifted direction once more. The new computer language made a quiet public appearance in 1994 under the provisional name Object Application Kernel (OAK), along with a World Web Web client known as WebRunner (later renamed HotJava). To the outside observer, WebRunner looks suspiciously like an attempt to revive an endangered project by tying it to the spectacular rise of the WWW. In this, Java is in good company; interest in the Internet and the WWW has revitalized many worthy but underutilized technologies, including the Integrated Services Digital Network (ISDN), which is currently (but probably not for long) the only practical way to connect many homes and offices to the Internet at a speed suitable for multimedia, and Standard Generalized Markup Language (SGML, discussed below).

To understand the potential impact of Java, it is necessary to understand the changes that the WWW has

already brought about. The WWW had its inception in the work of Tim Berners-Lee at the European Laboratory for Particle Physics (CERN) in the early 1990s and was further developed by programmers at the University of Illinois's National Center for Supercomputing Applications (NCSA), whose Mosaic Web client was the Internet's first "killer application." More recently, the WWW has been further developed by numerous commercial software houses, Netscape Communications prominent among them. Today, the WWW binds all of the Internet's most important user-level communications protocols, including those for electronic mail, file transfer, and remote log-in, as well as other network information systems such as Gopher, into one simple graphical interface.

The Web augmented existing Internet protocols with two of its own: HyperText Markup Language (HTML), the SGML-based markup used for creating Web hypertext documents, and HyperText Transport Protocol (HTTP), the underlying communications protocol. Based on the ubiquitous server-client model of network computing, a Web-based interaction typically consists of a text- or graphics-based Web client allowing the user to select a highlighted item, in response to which the client sends a request for a corresponding file to a Web server by means of an electronic addressing scheme known as a Uniform Resource Locator (URL). The server responds by returning the requested item or a message explaining why it can not do so. The Web supports an extensible list of multimedia formats; current servers can return text, hypertext, static and moving images, and sound.

The Web has rapidly become the most heavily used information retrieval system on the Internet. Two of the most important reasons for the Web's wide and rapid acceptance are its platform independence (Web servers and clients are available for most existing computing systems) and its support for rapid graphical user interface (GUI) development through extensions to HTML that allow documents to contain forms with various types of text areas, pick-lists, and checkboxes. Together with NCSA's Common Gateway Interface (CGI), which provides a mechanism for hiding virtually any pre-existing information system behind a Web server, these features have allowed information providers to put legacy database systems to work on the Internet rapidly and at a fraction of the cost that would have been entailed by writing native code to create interfaces for the three major software platforms: the PC, the Apple Macintosh, and UNIX machines. The main effort in providing access to many legacy systems is not in developing the interactive forms required for the user interface (though the restricted capability of HTML forms poses interesting challenges) but rather in developing a state engine within the CGI application to overcome the limitations of the single-transaction communication model of HTTP (which is why HTTP is often referred to as a "stateless" or "connectionless" protocol). Stateful CGI applications let a user engage in dialogue with an application that builds on a series of consecutive interactions with the user, as opposed to the single interaction allowed by a classic HTTP request.

The reliance of the Internet and the WWW on *open standards* has made possible the existence of a variety of free or inexpensive Web servers and clients that can operate despite running on quite different computing platforms. Internet specifications (including those for HTML and HTTP) are freely available and may be implemented without a license fee. This has led to an impressive outpouring of creative activity from both commercial and non-commercial software developers.

Where Does Java Fit In?

Java's future as a traditional application language is discussed below. In the context of the Web, it extends the list of items returned from a Web server to include bits of executable software (*applets*). It preserves platform independence by using *bytecodes*. On the server side, a programmer compiles Java source code into the equivalent of machine code for a *virtual* computer. These bytecodes, rather than source, are sent in response to an HTTP request, and the Web client must contain an interpreter that executes the bytecode commands on the local platform. Java bytecode interpreters already exist for MicroSoft platforms (Windows 95, Windows NT), the Apple Macintosh, and several variations of the UNIX operating system.

The first interpreter was embedded into the HotJava Web client, which runs on Sun workstations; Java interpreters have since appeared in various versions of the Netscape Navigator Web client. Java and its compiled bytecodes should not be confused with JavaScript, a client-side interpreted scripting language developed by Netscape (and originally known as "LiveScript"), which currently shares little with Java beyond a name.

Interpretation of bytecodes achieves platform independence at the cost of a performance penalty. Sun is currently developing a "just-in-time" compiler that would receive Java source code from a network and compile it into machine code for the local platform on-the-fly. Sun claims that this will eventually make the speed of Java competitive with, and possibly better than, the use of compiled C. This claim is not preposterous; at current network speeds, the limiting fac-

tor in execution speed will often be network transmission time.

As a language, Java strongly resembles C/C++, but with important simplifications and additions; it has jokingly been referred to as C++−−, or C++ without the knives, guns, and clubs. The type definitions, preprocessor commands, structures, unions, explicit pointers, and functions familiar to the C programmer have all been swept away in favor of the variables and methods that together constitute the "classes" of object-oriented programming. The explicit memory management commands of C have been replaced by an implicit system for the automatic scavenging of unused memory ("garbage collection"). Although the *goto* command has been deprecated for its disastrous influence on programming style since Dijkstra's famous paper,[3] it was still present in C but has been banned altogether from Java. The multiple inheritance and operator overloading features of C++ have disappeared, making work slightly more difficult for competitors in future "obfuscated Java code" contests. Java is a strongly typed language, imposing strict discipline on the programmer.

The six "packages" of predefined classes that accompany Java support the intrinsic language manipulations, utilities, mathematical operations, and input/output capabilities that would be expected from any general-purpose programming language. They also include support for network communications, GUI creation, and the manipulation of image and sound data. Java is multithreaded, a quality that is useful both for programming clarity and in multiprocessing environments. Java employs 16-bit characters, which should facilitate international applications through the use of character sets such as Unicode. A novel "documenting comment" allows the author to embed an extractable document about the program within the source code.

When Java is compiled for use as an applet, some of its capabilities are suppressed for security purposes: in particular, file input/output and calls to the native operating system. This suppression increases security at the cost of ruling out potentially useful operations, such as reading a client-side configuration file to customize the behavior of an applet. In spite of the attention that Java's creators have directed toward questions of applet security, it will be impossible to assess just how successful they have been until the system has been in widespread use for some time. In the first widely reported security incident involving Java, a group at Princeton University devised an applet that was successfully used to compromise system security through exploitation of a bug in the Internet Domain Name Service (DNS) system. Although the problem was actually a DNS defect and vendors responded quickly, this incident illustrates the problems that can arise from the interaction between distinct software systems in a networked environment. More recently, the same group discovered a more serious flaw in Java itself that allowed arbitrary commands to be executed on the client.[4] The question is not whether Java has security bugs but rather how quickly vendors will fix security bugs that are discovered.

Java was not the first technology to support network-downloadable applets for the Web, and it has worthy competition in this regard from Python and Tcl/Tk. Python is an object-oriented interpreted language developed by van Rossum, who has used it to create an experimental Web client known as Grail.[5] The Tool Command Language (Tcl) and its associated graphical windowing interface system, Tk, was developed at the University of California at Berkeley by John Ousterhout, who has since moved the project to Sun Laboratories. His colleague Stephen Uhler has written a Web client in Tcl/Tk, SurfIt!.[6] Both environments support multiple platforms, and their adherents can make cogent arguments for each one's technical advantages over the other system and over Java. However, neither of these alternative environments has attracted the attention that Java has. Nevertheless, even if Java displaces them as applet languages, they are likely to remain in wide use for other applications.

Java's Future Outside the Web

For several years, a number of carefully crafted, new object-oriented programming languages have been freely available from reputable academic centers, but none has developed a decisively large following. The most frequently encountered object-oriented language, C++, is widely perceived as undesirably complex, which may have hindered the broad acceptance of object-oriented programming. Objective C, a simpler commercial system that adds a few object-oriented commands to C, has enthusiastic adherents (particularly among users of the NextStep and OpenStep software development environments) but has not been widely used. Java is attractive both for its kinship with a widely used language, C, and for its elegant parsimony, but its Web tie-in is what helps it stand out from the pack. As more programmers employ Java for Web applets, they are likely to use it for free-standing applications as well. This could transform Java into one of the most widely used object-based languages, in turn giving a boost to object-oriented programming itself. The Common Object Request Broker Architecture (CORBA) is one of sev-

eral competing standards for sharing objects over networks. Workers at the Xerox Palo Alto Research Center (PARC) have created a freely available CORBA-compliant environment called Inter Language Unification (ILU) that can create systems that use any combination of the following languages: ANSI C, C++, Common LISP, Modula-3, and Java. There are also commercial CORBA environments that can be used with Java.[7]

Java's Effect on Biomedical Computing

Assuming that Java is an accelerator for integrating means and resources already brought about by the WWW, we can extrapolate from the recent past. Web-based clinical information systems have already been demonstrated at a number of institutions.[8,9] Most major academic medical centers now have a presence on the Web, and some are offering computer-aided instruction programs, some of which are eligible for continuing Medical Education credits. The National Library of Medicine is actively developing new database services, such as Internet Grateful Med, Online Images, and Sourcerer;[10] it and other institutions are collaborating with publishers to deliver the full text of biomedical journals electronically. Tools and databases for computational biology have appeared, as have interesting experiments in near–real-time robotics control and remote monitoring. The Web has already touched each leg of the tripod of academic medicine: research, teaching, and clinical care.

Java supplies the wherewithal to free GUI design from the restrictions of HTML forms and to integrate additional network protocols directly or indirectly into the Web environment. The facilitating power of the language is demonstrated by Habañero, a recently released application framework developed by NCSA that allows single-user software tools to become collaborative tools employed concurrently by multiple users on a network, independently of the WWW. NCSA chose a Java applet written by a student at Syracuse University as one of the first demonstrations of Habanero. This applet provides visual access to the multigigabyte dataset for the National Library of Medicine's Visible Human Project. In the coming year, we can expect to see new tools for one-on-one (point-to-point), one-to-many, and many-to-many (multicasting) audio and video teleconferencing. An information appliance that merges Web-like capability with the functions of the telephone, television, and remote control devices is within sight. The ability to integrate information retrieval from disparate sources in real-time could have a profound impact in a clinical setting.

Java is also triggering hardware developments that may influence clinical care. Sun has announced plans to transform the Java virtual machine into an actual machine, using a series of microprocessors that will employ Java bytecode as their native machine code. Oracle has announced production of an "Internet Toaster," an inexpensive terminal that derives all of its capabilities by dint of its network connection. Given the considerable expense of maintaining fully-programmable computers as clinical workstations, the appearance of cheap, interchangeable, networked information appliances that obtain their software on demand from a shared repository promises to preserve many of the advantages of distributed computing while restoring some of the benefits of centralized computing that disappeared with the mainframe. Assuming continued progress with the construction of wireless networks, Java may get an opportunity to revisit its original purpose as the application language for portable personal information appliances.

If Java is to fully realize its promise, three overarching concerns must be addressed. First, the security of Java and the Web must be acceptable. The use of public-key cryptography and digital signatures to establish secure channels of communication and to validate the source of Java classes is near at hand. The alacrity of Sun and Netscape in responding to reported security holes in Java itself is also reassuring. Second, network access must be cheap, reliable, and easy to establish. The rush of commercial vendors to get personal computing devices onto the Internet via analog and digital telephones and the new and faster means of connection that are anticipated in the wake of the recent deregulation of the U.S. telecommunications industry suggest that this problem will also be solved.

The third concern is more subtle and less clearly under control: the need to maintain shared open (nonproprietary) standards governing both Java and the WWW. There is constant temptation for a software vendor to add new nonstandard features to a Web server or client to create a perceived advantage of their product over its rivals. This works against the interoperability that has contributed importantly to the Web's success. The primary center for Web standards-making appears to have shifted from the Internet Engineering Task Force (IETF) to the recently formed World Wide Web Consortium (W3C) headquartered jointly at MIT and INRIA (Paris). The W3C's use of the term "de facto standards" in apparent preference to "open standards" is worrisome. With respect to Java, Sun claims to have learned from the factionalization that the UNIX operating system underwent when it entered the commercial market. To avoid the appearance of multiple noninteroperable

Java dialects, Sun is licensing Java's source code to developers while retaining rights to pull back changes that are made, in order to fold them into the generic version. The goal for both the WWW and Java should be to achieve the right balance between adhering to defined open standards and leaving room for continued innovation.

Finally, we must not forget that revolutions stop for no one and that early leaders sometimes become victims. As of this writing, rumors are leaking out of AT&T Bell Laboratories about a crash program, codenamed Inferno,[11] being led by Dennis Ritchie, one of the inventors of UNIX. Very little is known about this project, but its developers have hinted that they think that Java does not go far enough. Hunker down behind your barricades with a good cup of coffee, and watch this space for future developments.

The author thanks Henry McGilton for checking this editorial for factual correctness and sharing useful remarks.

References ■

1. Gosling J, McGilton H. *The Java Language Environment*. Sun Microsystems Computer Company, White Paper. October 1995.
2. Lowe HJ, Lomax EC, Polonsky SE. The World Wide Web: a review of an emerging Internet-based technology for the distribution of biomedical information. J Am Med Inform Assoc. 1996;3:1–14.
3. Dijkstra EW. Go to statement considered harmful. *Communications of the ACM*. 1968;11:147–8.
4. Dean D, Felten E, Wallach D. Java security: from HotJava to Netscape and beyond. *Proceedings, IEEE Symposium on Security and Privacy*. 6–8 May 1996. Refer to: http://www.cs.princeton.edu/~ddean/java/nativecode.html.
5. For further information about Python and Grail, refer to: http://www.python.org/.
6. Ousterhout JK. *Tcl and the Tk Toolkit*. Reading, MA: Addison-Wesley, 1994. For further information about tcl/tk, refer to: http://www.sunlabs.com/research/tcl/; for information about SurfIt!, refer to: http://pastime.anu.edu.au/SurfIt!/.
7. Information about the ILU CORBA package and its use with Java can be found at http://www-db.stanford.edu/~hassan/Java/Jylu/; commercial CORBA environments for use with Java include Sun's JOE (http://www.sun.com/sunsoft/neo/external/neo-joe.html) and PostModern Computing's BlackWidow (http://www.pomoco.com/).
8. Willard KE, Hallgren, JH, Connelly DP. W3 based medical information systems vs. custom client server applications. *Proceedings, Second International World Wide Web Conference. Chicago*. 17–20 October 1994;641–51. Refer to: http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/MedTrack/.
9. Cimino JJ, Socratous SA, Clayton PD. Internet as clinical information system: application development using the World Wide Web. *J Am Med Inform Assoc* . 1995;2:273–84.
10. HyperDOC, NLM's WWW server (http://www.nlm.nih.gov/), provides examples of stateful search services that exploit CGI/WWW (including *Perez on Medicine, Images from the History of Medicine*, and *Internet Grateful Med*) as well as access to the Syracuse Visible Human Java applet.
11. Information about Inferno can be found at: http://inferno.bell-labs.com/inferno/ and at http://www.suck.com/dynasuck/96/04/17/