Introduction

Crossroads Java edition

by Vishal Shah

Welcome to the Winter 1997 issue of Crossroads. In recent years Java has generated enormous interest as a language, a platform and a philosophy. The numbers speak for themselves. Serious Java programmers: 450,000+; JDK 1.1 downloads: over one million; Universities offering Java courses: 200; books in print about Java: 800+ [1]. Java promises to revolutionize personal computing, make the proprietary-closed-software-running desktop obsolete and shift the center of gravity in the computer world toward the network.

Java has developed from a simple language into a full scale development platform. It can talk to the networks, legacy applications, smart cards, object brokers, databases and (believe it or not) telephones and washing machines too. Java has collected the right combination of buzzwords to describe itself: simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, multi-threaded, and dynamic. All kidding aside, these words reflect features of Java that provide solutions to real software problems. JDK 1.1 has matured and its new APIs provide the features required to build mission-critical Internet/intranet and stand-alone applications.

Java is a pretty cool, relatively easy, quick and fun programming language. But what's cool to me may not necessarily be cool to everybody. Java lags behind in performace compared to C and C++. Security and interoperability are still major issues. Experienced programmers complain about the inability to do quick-and-dirty programming. Platform independence cannot be taken for granted. Moreover, Java is showing the potential of splintering into local dialects. Whether Java has achieved all that it promised is debatable. However, there is no denying that Java has continued to be hot and is taken seriously by everyone.

In this issue, we have gone beyond the hype surrounding Java, and brought you a useful cross-section of articles on which you can build your next generation of Java applications. We begin this issue with an article by Fouzi Husaini on using the Java Native Interface. JNI is a core technology in the JDK 1.1 release, and enables Java programmers to integrate native code into their Java applications. Fouzi, in addition to explaining JNI via examples, provides some precious tips for developers.

Moving to some of the more research oriented articles, Anita J. Van Engen, Michael K. Bradshaw and Nathan Oostendorp have written an excellent paper on extending Java to support shared resource protection and deadlock detection. The need for these functionalities is evident because the current Java specification does not provide adequate support for parallel programs.

Brian T. Kurotsuchi has described the technology of object serialization. This is one of the new features of JDK 1.1 and promises to elevate Java from a language for writing nifty applets into a language in which serious business applications can be written. Brian provides an excellent introduction and tutorial to utilize this technology effectively.

During the fall of 1996, Georgia Tech made a transition from procedural to object-oriented programming at their introductory level courses. Jason I. Hong discusses the choices, perils, pitfalls and experiences of this effort. We hope this article would inspire other educators to consider Java as a first programming language.

To conclude, we present the first installment of the revived Objective Viewpoint column written by George E. Crowford. This column will contain all you ever wanted to know about Java but were afraid to ask. Here's your chance to jumpstart the future.

Let us know if you enjoyed this issue. You are always welcome to send questions, suggestions, or to simply drop us a line at crossroads@acm.org. Also, sign our guestbook at http://www.acm.org/crossroads/gbook/sign.html

References

1

Martin, H. *Day one: Java Internet Business Expo*. http://www.javasoft.com/events/jibe/dayone.html