GUI Based Shell Commander in Java

RAJKUMAR and **LATHA R**
Operating Systems Group
Centre for Development of Advanced Computing
2/1, Ramanashree Plaza, Brunton Road
Bangalore - 560 025, Karnataka, India
Email: raj@cdacb.ernet.in / kumar@engineer.com URL: http://cdacb.ernet.in/~raj

Abstract

Organizing and managing files and directories are very frequently used operations in any operating system. These operations can be performed efficiently and conveniently by using tools, which increase user productivity. However, on most of the operating systems these operations are performed by using command line interface, which are operatinig system specific and difficult to learn and use. Hence, there is need of a tool, which allows to specify the operations on files using point-and-shot method and shortcuts. The tool must be portable across all platforms providing consistent interface i.e., having the same look and feel, on all hardware and software platforms. The Shell Commander (SC) developed in Java is one such tool which effectively leverages features of Java—imaging and multithreading.

Now-a-days much importance is given to smooth user friendly Graphical User Interface (GUI). GUI acts as an interface between the user and the system. The Java's Abstract Windowing Toolkit (AWT) provides a single windowing user interface on systems with widely different native window systems. It does this in a clever way by supporting only the functions that are common to all window systems. The AWT continues the portability goals of the language. The shell commader user interface is built using this facility. In this paper, we discuss a tool, shell commander, which effectively uses Java features, and its object-oriented architecture, design, and development.

1. Introduction

Organizing and managing files and directories are among the fundamental tasks performed in any operating system. Knowing how to organize files into directories and how to manipulate those files will increase the user productivity. Command line programs are much faster. However, the advantage of using File Manager is that it provides some valuable graphical shortcuts, mainly while performing selective file manipulation with mouse interface.

The shell acts as an interpreter or in other words, an interface between the user and the system. When logged into the operating system, we are in the shell. The main function of shell is to interpret and execute the commands presented by the user at the command line. Each operating system has its own shell command language which covers in detail the way of using the shell as a command interpreter. The shell interprets the first word of the command line as a command name. The shells in most operating systems, especially those based on Unix, support two types of commands. First, internal command, a command implemented as a part of shell. Second, external command, which can be an executable file available on the disk. The Shell Commander (SC) can be included in the second type.

The main objective of the project is the development of a platform independent file and process management tool. Though Java was designed to create network-based applications, the architecture neutral approach is useful beyond the scope of this. As an application developer in today's software market we probably want to develop versions of our application that can run on PCs, Macs and UNIX workstations. With multiple flavors of UNIX, Windows 95 and Windows NT on the PC and the new Power PC Macintosh, it is becoming increasingly difficult to produce software for all the possible platforms. If we write our application in Java, it can run on all platforms.

2. Java Language

Java is a programming language developed by a team headed by James Gosling at Sun Microsystems Inc. In *The Java Language: A White Paper* [1], Sun describes Java as follows:

> *Java is simple, object oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multithreaded, and dynamic language. Though this is quite a string*

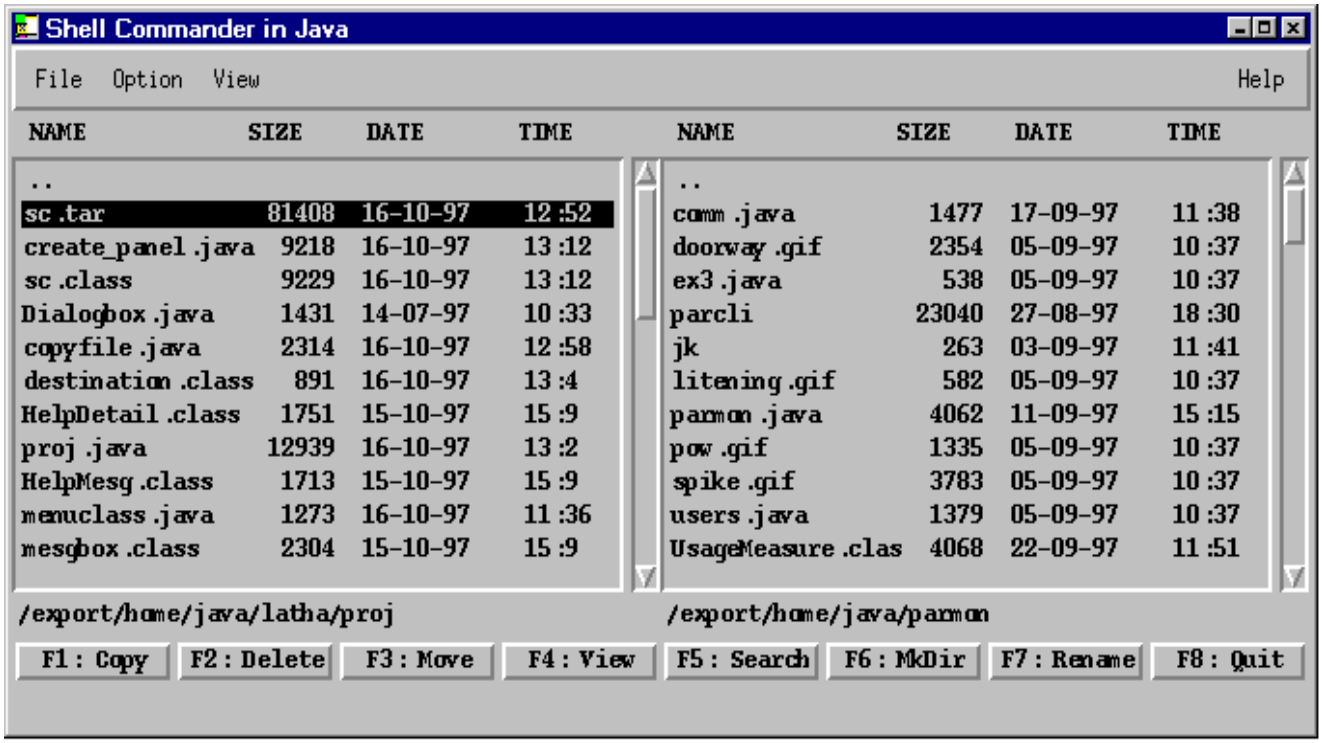

Figure 1: Shell Commander User Interface

of buzzwords, the fact is that they all aptly describe the language.

The Java environment [2] is a threefold set of specifications and tools that allow developers to produce dynamic, portable and high performance programs as easily as possible, while giving the end user and administrator a secure robust runtime environment in which these applications can be executed.

Java is an object oriented, familiar, and easy-to-use language with which applications can be developed. The Java architecture incorporates modern features and is tuned to distributed, heterogeneous platforms. The Java tools provide programmers and end users the flexibility to produce and use its features which will shape the Internet.

3. Shell Commander Specification

The project is aimed at providing a set of GUI based utilities for file, process and device management. (The scope of this project has appeared in [7]—*The Java Solutions Guide*, Sun Microsystems). The programming language shall be Java so that the tool developed will be platform independent. The following are proposed specifications of the shell commander:

1. The tool will be capable of working in any operating system platforms such as Solaris, Windows 95/NT, and Macintosh.
2. The tool should be having a user friendly graphical interface (see Figure 1). All operations can be performed with mouse events as well as with keyboard inputs. Feed back of the operations should be provided to the user.
3. Multiple operations or processes can be performed at the same time (using the multithreading feature of Java).
4. The tool should be capable of saving the current status. When restarted by the same user, it has to display subwindows as per the previous settings.
5. The result of an operation should be visible to the user as soon as the operation completes. If an operation cannot be performed/continued due to some errors, the user should be made aware of the same.

User Interface Specification

The main window of the tool should be having two similar subwindows (see Figure 1—executed in MS Windows environment), both displaying the details of the files and subdirectories in the current directory from where the tool is invoked. By mouse click the user should be capable of moving to any directories. i.e., the tool should be capable of navigating through the entire file system. The current directory name details should be displayed at the bottom. The file details include the files name, size of the files, date and time of last modification. A menu bar is set in the main window. A name panel should be provided at top which acts as the title for the sub windows. A comment panel should be provided

at the bottom where the operation status messages are displayed during execution.

4. Design and Implementation

The shell commander tool is designed to work as a standalone application. The design methodology used is object oriented approach, which is based on the concept of data abstraction. In object oriented methodology an object is defined as a package of information together with the operations that can be performed on that information. In other words, an object supports data abstraction and can be considered as an entity which encapsulates the object data and provides the user with a set of predefined operations to manipulate and access the object's data. The object's data can only be accessed by the methods defined on the object. An object has an internal view and an external view. The external view of an object consists of the interface provided by the object, namely, the operations that the object supports, how the operations are implemented and the structure of the object's data. However, other internals of an object are not visible outside of the object. Internally, the object has to organize and manipulate data in such a fashion that the external behavior of the operations are maintained. From outside, an object can only be allowed to perform an operation.

The shell commander's class hierarchy diagram is shown in Figure 2. In the following section, we discuss the class, `copyfile`, which creates three objects (producer—an instance of the class `source`, consumer—an instance of the class `destination`, and buffer—an instance of the class `buffer`) illustrating producer-consumer model for concurrent processing. Other classes are self explanatory.

The class `CopyFile`

The class `CopyFile` is designed to perform the file copying operation using the multithreading features of Java. By this multiple files can be copied at a time by initiating (starting) the read and write threads. A read thread reads from the source file into the buffer and a write thread reads out from buffer to destination file. Buffer operations are synchronized so that only one thread can access the buffer at a time. As soon as the copying operation terminates, a method in the parent class is invoked from this class to inform that the copy operation is completed. If copy operation fails, that message is passed back to the parent to take necessary corrective actions.

Main Objects: `buffer`, `sou`, `dest`

The objects `buffer`, `sou`, and `dest` are instances of the classes `buffer`, `Source`, and `Destination` respectively. The methods of the class `CopyFile` include:

1. `CopyFile(String source, String Destin, SC parent)`
2. `public synchronized void terminate( boolean finished)`

The method `terminate()` invokes the parent method to inform the completion of copy operation.

The class `Source`

The class `Source` is designed to copy the content of source file to buffer. The class `Source` extends the class `Thread`. The methods of the class `Source` include:

1. `Source(CopyFile parent, buffer b, String source)`
2. `public void run()`

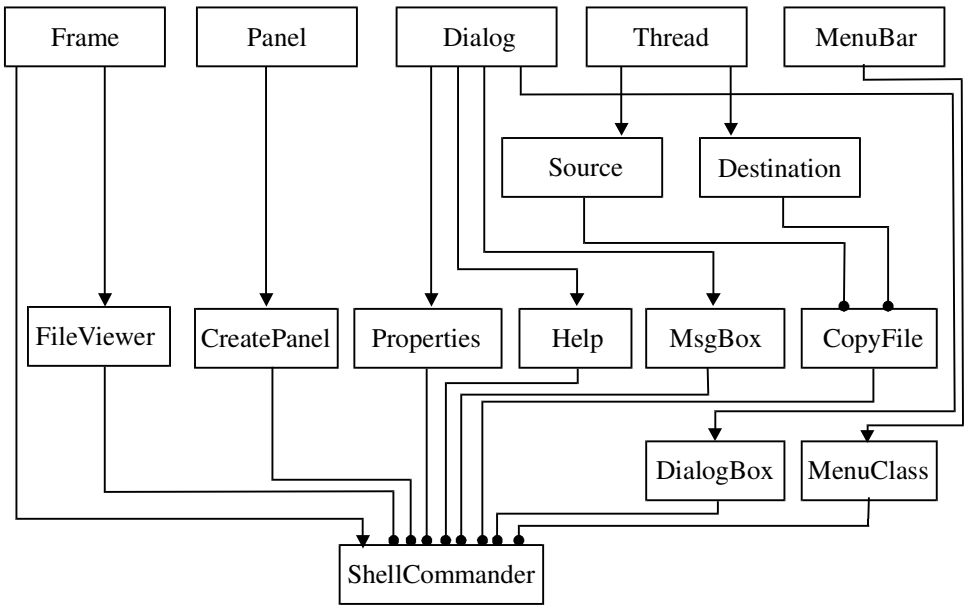

Figure 2: Hierarchy of Classes in Shell Commander

The class `Destination`

The class `Destination` is designed to copy the content of buffer to destination file. The class `Destination` extends the class `Thread`. The methods of the class `Destination` include:

1. `Destination(CopyFile parent,buffer b, String destfile)`
2. `public void run()`

Multithreaded File Copy Algorithm

The data is read from the source file to a buffer and then from the buffer to the destination. It uses multiple buffers which are connected by using a linked list. Multithreading feature of Java is used here. Two threads (read and write) are created as soon as this operation is invoked. *Read thread* reads from the source to the buffer. Once one buffer is full, it reads to the next buffer. *Write thread* takes data from the buffer and writes to the destination. Read thread creates new buffer each time to load the read data which is linked to the previous one. Write thread reads out data from the buffer in this order. By this, the read and write operations progress simultaneously until file copy operation completes. Multithreading reduces the communication and computation latency [6]. Communication time is reduced as these read and write threads are not communicating. Computation is reduced as multiple buffers are used and these buffers are created dynamically. Multithreaded programming allows to overlap computation and communication operations and process them simultaneously, which improves execution speedup.

5. Testing and Sample Operations

In a software development project errors get injected at any stage during the development. The shell commander is rigorously tested so as to make sure that it meets all proposed requirements. All the features of shell commander have been tested and of course, many bugs were found and they have been corrected. The following section illustrates testing procedure adopted in navigating through the file system using shell commander with suitable clippings.

Navigating through the File system

The files and directories in the current directory (from which the tool is invoked) are displayed in the tool subwindows We will often need to change the current directory to look at the contents of another directory. One method is to double click. To move to a new directory double click the mouse at the chosen item or press enter. All the files and subdirectories contained in that directory are displayed in the work panel. To move to the parent directory select the first item in the ListBox which is double dots. All the files and directories in the parent will be displayed on the screen.

Sample Screen Displays

All commands supported by SC can be invoked and processed in point-and-shot manner. The menu bar contains the options File, Option, View, and Help. When File option is selected, the SC user interface appears as shown in Figure 3.

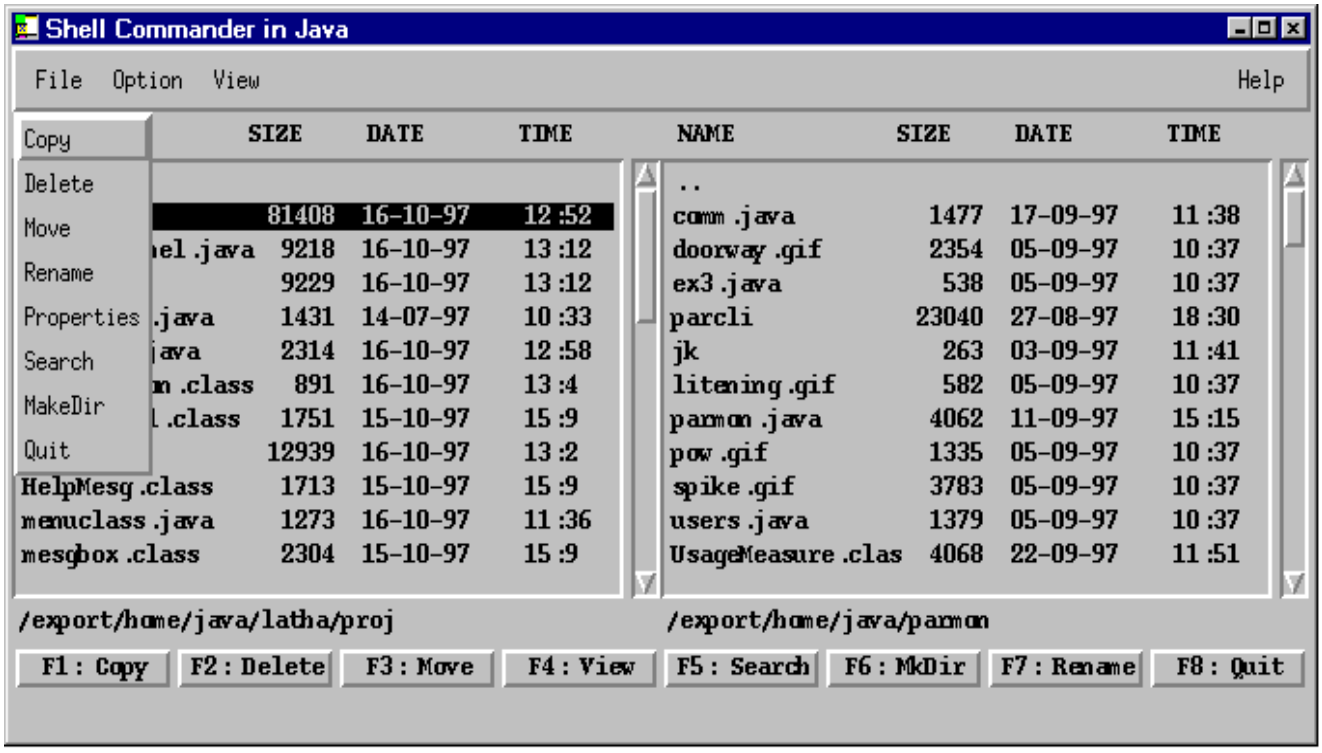

Figure 3: GUI status on Clicking File Option

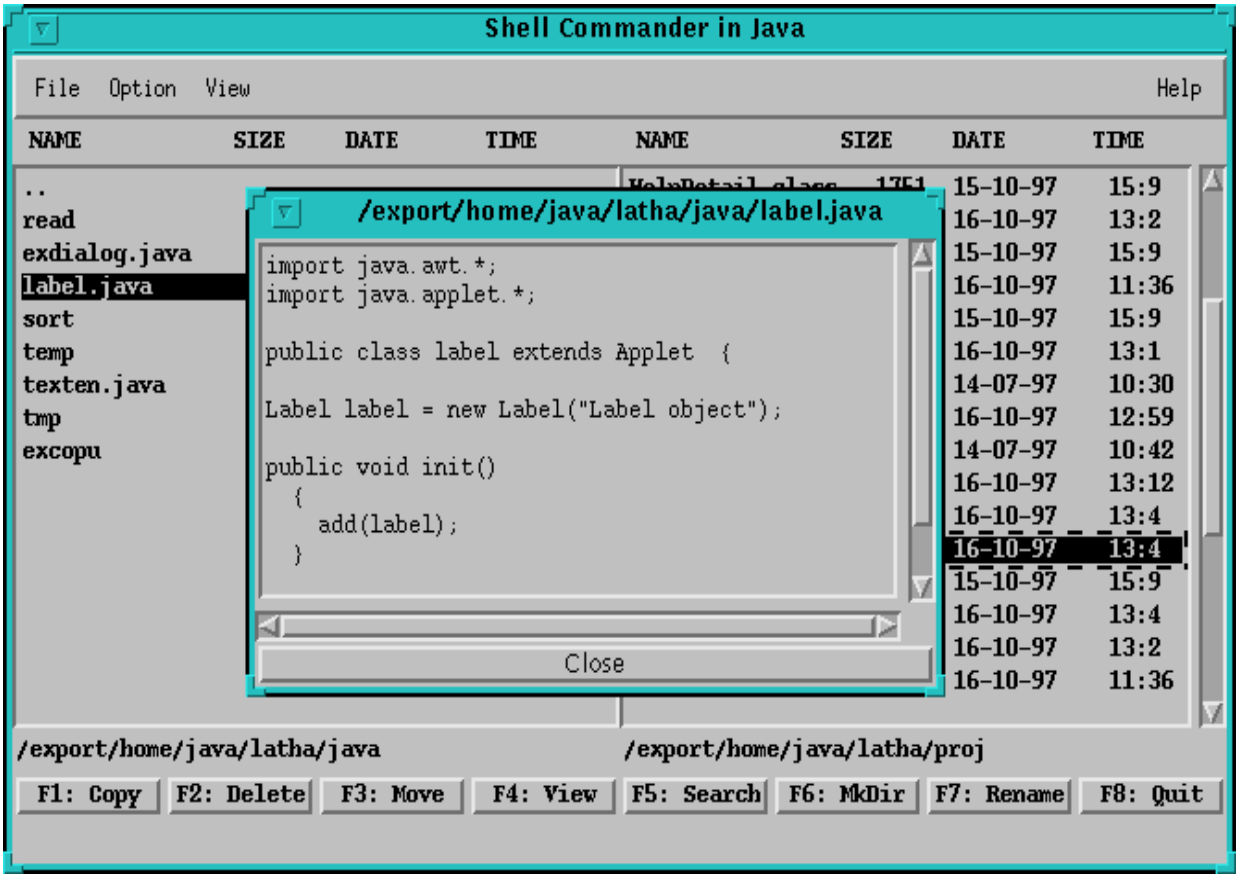

Figure 4: GUI status on Clicking File View option in the View Menu

The selection of `Delete` command from `File`'s submenu deletes an active file. The file must be marked as active by clicking on that before invoking `File` menu from the menu bar. It should be noted that all commands operate on the active file, by default.

The commands supported in `File` menu are: `Delete`, `Move`, `Rename`, `Properties`, `Search`, `MakeDir`, and `Quit`. Note that, these operations work both on files and directories. For instance, when `Delete` is selected and the active file is a directory, then the entire directory will be deleted even if that directory is not empty.

The `Option` submenu contains commands for changing the font size and character width. The `View` submenu contains commands for viewing files and ordering items to be displayed in subwindows. The files in the current directory can be sorted on the basis of name, size, and date of last modification. In addition to this, various commands for changing color and configuration are also provided.

The shell commander creates subwindows when commands such as `FileView` is selected as shown in Figure 4. This sample output is generated by executing shell commander under Unix operating system, Solaris. Note that, the user interface has the same look and feel except that, the window frame is the same as that of the native (Solaris Openwin) windowing system. The execution of the same binary of the shell commander on Java runtime system in Microsoft's Windowing system and Sun's Solaris OpenWin system, clearly demonstrates the portability of the shell commander across multiple platforms.

6. Conclusions and Future Work

A tool developed is said to have flexibility only when it is user friendly. With the help of Java's Abstract Windowing Toolkit this objective is fully achieved in shell commander. The entire package is developed as menu-driven system. The greatest advantage of the shell commander tool is its platform independent nature. The tool is completely developed in Java. It is successfully tested and used in Windows NT/ 95, Unix (Solaris) and Macintosh. The architecture neutral nature of Java allows the tool to be fully compatible with any platform.

The current system does not support drag and drop feature, since, such facilities are not yet available as Java features. Sun has promised to provide these features in future versions of Java, which will also be incorporated in later versions of this package.

7. Availability

The portable shell commander developed in Java is available in the public domain by remote FTP access. At the site, `http://cdacb.ernet.in/~raj` goto the section *Research Publications*, and click on this paper title, which displays the abstract of this paper. Click on Download Shell Commander Package, to download the package. The package is stored in the file `sc.zip`, which contains all source files compressed and zipped using `pkzip` utility. Suggestions for further improvement are welcome and they can be mailed to raj@cdacb.ernet.in / kumar@engineer.com.

References

[1] Sun Microsystems, *Java Language - A White Paper*, Sun Microsystems Computer Company, 1996.

[2] James Gosling,Frank Yellin & The Java Team, *The Java Application Programming Interface,* Volume I and II,Sun Microsystem, Addison Wesley, 1996.

[3] Tim Ritchey, *Java!*, New Riders Publishing, Indianapolis, Indiana, 1996.

[4] Patrick Naughton, *The Java Handbook*, McGraw-Hill Publications,1996.

[5] David Flangan, *Java in a Nutshell*, O'Reilly & Associates Inc,1996.

[6] Rajkumar *et. al*., *PEACE Threads Interface on PARAS Microkernel*, The Third International Conference on Advanced Computing, India, 1995.

[7] Sun Microsystems, *The Java Solutions Guide*, Javastation Launch Edition, Sun Microsystem, 1996.

About the Authors

Rajkumar is a Member of the Technical Staff, Operating Systems Group, Centre for Development of Advanced Computing, Bangalore, India. He received the Bachelor of Engineering degree in Computer Science from the University of Mysore (JMIT) in the year 1992 and the Master of Engineering degree in Computer Science from the University of Bangalore (UVCE) in the year 1995. He was awarded the Dharma Ratnakara Memorial Trust *Gold Medal* for securing the I Rank in the year 1992. He is a visiting faculty member for the Bangalore University. He has coauthored the books *Microprocessor x86 Programming* and *The C++ Handbook*. He is an associate editor of Proceedings of the Fourth International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97), USA. He also serves as a reporter for Asian Technology Information Program, Japan/USA.

Rajkumar lectured extensively on advanced technologies such as Parallel, Distributed and Multithreaded Computing, Client/Server Computing, and Internet and Java all over India as a part of access'96 conference organized by C-DAC and Sun Microsystems. In *High Performance Computing ASIA '97* Conference and Exhibition held at Seoul, Korea, he delivered the tutorial on Multithreaded and Distributed Computing. In *High Performance Computing HiPC '97* Conference to be held in India, he delivers the tutorial on Internet, Java, and Java Computing in Practice. He served as technical session chair on *Unified System View by Software Means*, PDPTA'97, USA. He was on the Program Committee, International Conference on Imaging Science, Systems, and Technology (CISST'97), USA. He is on the Program Committee, Fifth International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), USA. His research papers have appeared in National and International Conferences. His research interests include Programming Paradigms and Operating Environments for Parallel and Distributed Computing. He can be reached by the e-mail at `raj@cdacb.ernet.in` and by the URL `http://cdacb.ernet.in/~raj`.

Latha R is a Project Trainee, Operating Systems Group, Centre for Development of Advanced Computing, Bangalore, India. She has completed Master of Technology in Computer Science from the Cochin University of Science and Technology, Kerala in the year 1997. Her research interests include Operating Systems and Java Computing.