



P-1116

JAVACC COMPILER AND JEDITOR

PROJECT WORK DONE AT
PENTASOFT TECHNOLOGIES
CHENNAI

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THE AWARD OF THE DEGREE OF
M.Sc [APPLIED SCIENCE] SOFTWARE ENGINEERING
OF BHARATHIAR UNIVERSITY, COIMBATORE.

SUBMITTED BY
VISHAL KUMAR SINGHAL
Reg No. 9937S0099

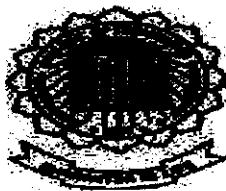
UNDER THE GUIDANCE OF

External Guide

Ms RAJSHREE SHIVA, M.C.A
Pentasoft Technologies
Chennai

Internal guide

Mr. RAJASEHAR, M.C.A
Lecturer – Dept of CSE - KCT
Coimbatore



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE – 641 006

OCT 2003 – MARCH 2004

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University)
COIMBATORE – 641 006
OCT 2003 – MARCH 2004

CERTIFICATE

This is to certify that the project entitled
JavaCC Compiler and JEditor

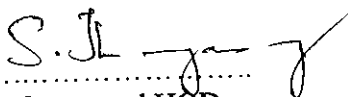
DONE BY

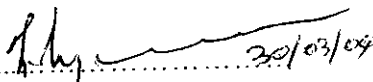
VISHAL KUMAR SINGHAL

Reg No. 9937S0099


SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF

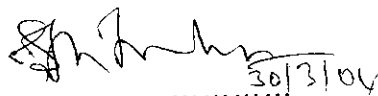
M.Sc [Applied Science] SOFTWARE ENGINEERING
OF BHARATHIAR UNIVERSITY


.....
Professor and HOD


..... 30/03/04
Internal Guide

Submitted to University Examination held on ..30/4/2004.....


.....
Internal Examiner


..... 30/3/04
External Examiner



09 March, 2004

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **Mr. Vishal Kumar Singhal**, pursuing fifth year M.Sc. Software Engineering at **Kumaraguru College of Technology, Coimbatore**, has successfully completed the project titled "**JavaCC Compiler and Editor**" in the area of Java. The duration of the project was from October 2003 till March 2004.

During this period, we found him to be sincere and hardworking.

For PENTASOFT TECHNOLOGIES LTD.,

A handwritten signature in black ink, appearing to read 'M.A. Farzana', written over a horizontal line.

M.A. Farzana

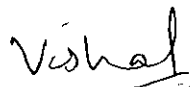
Centre Manager

DECLARATION

I hereby declare that the project entitled "**JavaCC Compiler and JEditor**" submitted to **Bharathiar University, Coimbatore** as the project work of **Master of Science Degree in Software Engineering**, is a record of original work done by me under the supervision and guidance to **Ms. Rajshree Shiva [Project Manager]**, PentaSoft, **Prof.K.R.Baskaran– Asst.Professor & Course Coordinator [Software Engineering]** , **Kumaraguru College of Technology, Coimbatore** and this project work has not found the basis of the award of any Degree/Diploma/Associate ship /Fellowship or similar title to any candidate of any university.

Place : COIMBATORE

Date : 30/4/2004



Vishal Kumar Singhal
Registration No: 9937S0099
M.Sc [Software Engineering]
Kumaraguru College of Technology

ACKNOWLEDGEMENT

At the outset I would like to thank my principal Dr.K.K.Padmanaban, Kumaraguru College of Technology, Coimbatore, and Dr.S.Thangasamy, Head of Department, Computer Science and Engineering for giving me the opportunity to do this project as a part of my course.

I express my sincere gratitude to professor K.R.Baskaran – Asst. Professor and course co-ordinator [Software Engineering] and Mr.R.Rajasehar , M.C.A, Lecturer who personally been my mentor and guide for the successful completion of the project.

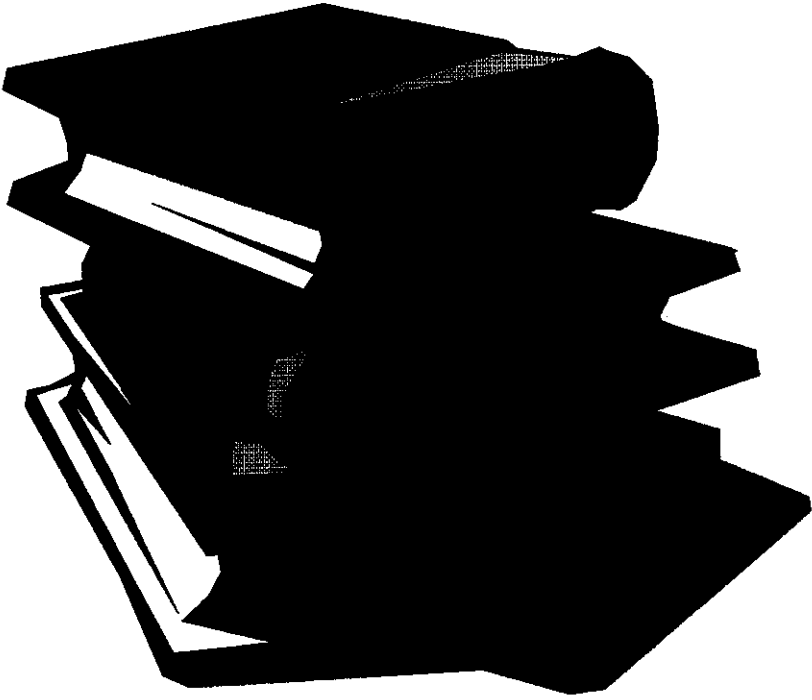
A project of this size requires a guide who takes pain to spend time and hands out invaluable advice from time to time. Here I am greatly indebted to Ms. Rajshree Shiva, M.C.A, Project leader, for his invaluable guidance from start to finish.

I thank my parents who have extended endless support to my all my endeavors.

I thereby thank all the people who in one way or the other have helped me in my completion of the project.

CONTENTS

	PAGE NO.
1. INTRODUCTION	1
1.1. PROJECT OVERVIEW	2
1.2. ORGANIZATION PROFILE	3
2. SYSTEM STUDY & ANALYSIS	4
2.1. EXISTING SYSTEM	5
2.2. PROPOSED SYSTEM	5
3. PROGRAMMING ENVIRONMENT	7
3.1. HARDWARE CONFIGURATION	8
3.2. DESCRIPTION OF SOFTWARES & TOOLS USED	8
4. SYSTEM DESIGN	21
4.1. INPUT DESIGN	22
4.2. PROCESS DESIGN	23
5. SYSTEM IMPLEMENTATION AND TESTING	24
5.1. SYSTEM IMPLEMENTATION	25
5.2. SYSTEM TESTING	27
6. CONCLUSION	31
7. SCOPE FOR FUTURE DEVELOPMENT	32
BIBLIOGRAPHY	33
APPENDIX	
SAMPLE SCREENS	35



INTRODUCTION

1.1 PROJECT OVERVIEW

JavaCC Compiler

The JavaCC Project is a Java software project from DMS which provides a framework for developing compiler applications using Java.

JavaCC contains a set of tools which allow you to edit and generate classfiles: dis (Java disassembler), KSM (Java assembler) and KJC. KJC compiles Java source code to bytecode, with all the same plus even more features as commercial compilers.

- JavaCC can be executed from existing Makefiles and IDE's.
- JavaCC is coded in java, so it is *extremely* fast.
- The resulting byte code is small -- starting at about 11 Kbytes.
- No additional runtime libraries are required. The generated source code is the *entire* parser.

Editor

It is used to tidy up your java code, making it easier to read. It supports features such as Color Syntaxing, bracket matching, and a line number gutter.

Most of its features are obvious, it supports cut, copy, paste, undo, redo as expected,

New features currently being developed include :

- Code Completer feature
- Customizable Toolbar & Menu bar, Popup Menus & Key maps.
- Improvements made to the Project Manager & Jar File Manager Tools.
- File saving facility to back-up edited files before compiling.
- Color printing.

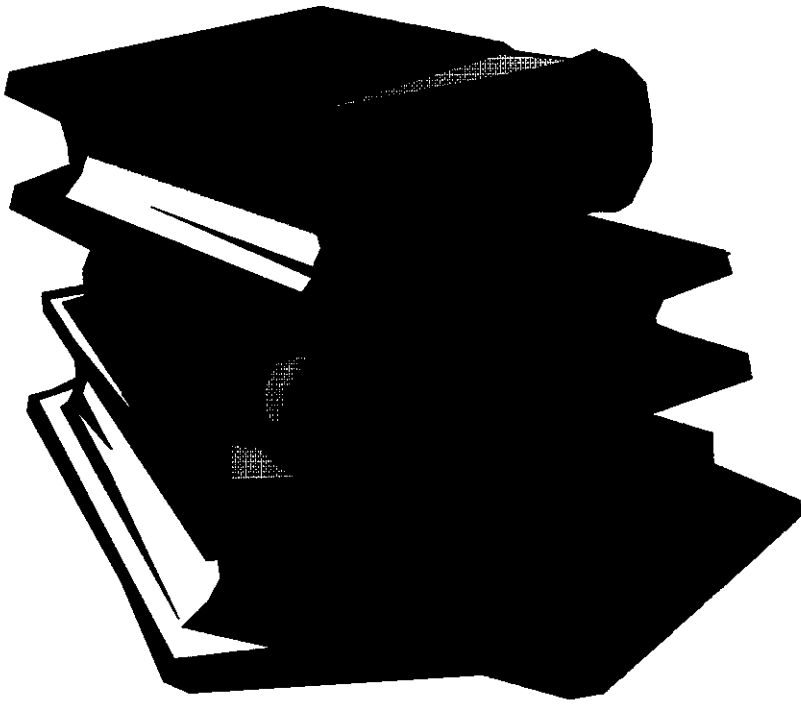
1.2 ORGANIZATION PROFILE

Pentsoft Technologies Limited (formerly Pentfour communication Limited) with the acquisition of the Software Division from Pentfour Software and Exports Limited (PSTL) has recorded acquired for USD 205 million, has added to the revenue growth of the company. The company has registered a turnover Rs.100.01 Crores for the quarter ended December 31, 2000. The turnover for the nine months ended December 31,2000, is Rs.286.70 crores.

The company focuses on 5 main SBU's namely Business Software Services; Education and Training; System Integration; Engineering services and Internet & Communications Services. The Company is awarded ISO 9001 Certificate in the areas of Engineering Services (CAD/CAM).

The company has tied up with IBM as business partner, Sybase, Compaq, Oracle, Microsoft, Novell, silicon Graphics, apple, synon, obsidian, SUN, SSA, Eastman Kodak,. DELL (Asia) and Purdue University, UAS, Siemens for Exchange and Video Conferencing, Comsat Max for VSAT, AT & T for Networking to give a total solution through integration of Hardware, software and Networking.

Pentsoft technologies Ltd., has a wide network of offices spread over North America, Europe, Australia, Mauritius, Singapore, Malaysia, Thailand, Bangalore, Bombay, Calcutta, Coimbatore, Delhi, Hyderabad, Madurai, Mangalore, and Trivandrum.



SYSTEM STUDY & ANALYSIS

2.1 EXISTING SYSTEM

Most proposals which introduce contract programming in Java use preprocessing techniques and translate the source code into an instrumented source code. This approach violates abstraction and modularity. Source to source translators make debugging of the byte code very uncomfortable and prevent separate compilation, since the source code or the body of the methods of the inherited type must be available to the compiler.

2.2 PROPOSED SYSTEM

This proposal needs no extensions to the classloader or other changes to the JVM. The source code of the parent class is not necessary, it is not a source to source translation. Abstraction, modularity and separate compilation is fully supported. The code can be translated to Java byte code in one step. This is very useful because it makes the debugging of the code much easier than debugging of code produced by a precompiler.

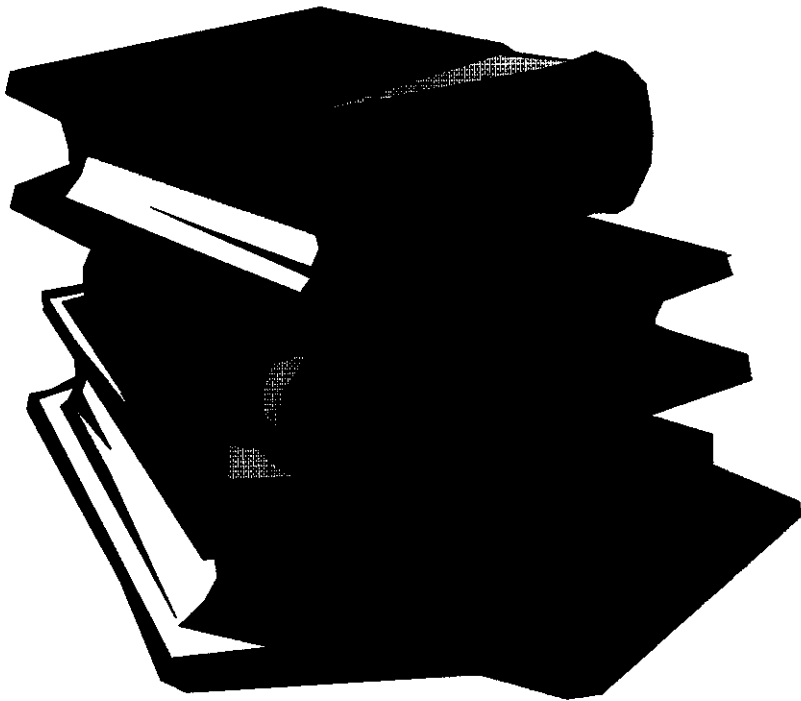
- The development process is safer, quicker, and in some ways simpler since optimizations can be done on the generated Java Code
- It is possible to reuse existing, aggressive optimizers.

Small software components integrated in Web services - These components can undergo frequent changes from one load to another by the same client. As a result, in this context, a JIT compiler is the most appropriate solution.

Platform-independent large software - Such programs may or may not be related to Web services. Java technology is used because of its machine independence. The Java tools themselves are examples of such programs (e.g., compiler, disassembler, ...). These programs change

infrequently and are often used by many users. Therefore, keeping a local, optimized version of the compiled code is advantageous. By comparison to a JIT, that always get the latest version of the software, this approaches requires the management of local optimized versions. This can be implemented by a revision control system that compiles and installs new software versions as they are released, in a automatic and transparent way.

Platform-dedicated software - Examples are operating system components [3] and embedded applications. For these applications, the Java technology provides safety. These applications are characterized by very infrequent changes. Hence, it is advantageous to optimize the final code for the target system.



*PROGRAMMING
ENVIRONMENT*

3.1 HARDWARE CONFIGURATION

SERVER:

Processor	: Pentium III
Speed	: 433 MHZ
Ram	: 128MB
Hard Disk	: 20 GB
Operating System	: Windows 2000
Disk Drives	: 1.44 Floppy Disk Drive, 40 x Compact Disk
Monitor	: 14" Color Monitor

CLIENT:

Processor	: Pentium
Speed	: 400 MHZ
Ram	: 64 MB
Hard Disk	: 10 GB

3.2 DESCRIPTION OF SOFTWARES USED

JAVA DEVELOPMENT TOOLKIT VERSION 1.2.1

Architecture of a JavaCC Compiler

Architectural Overview

A modern optimizing compiler can be logically divided into four parts:

- **The compiler front end**

The front end includes the scanner and parser which read the Java source and build an abstract syntax tree (AST) representation of the source code. The front end must also be able to read the symbol information in the Java ".class" files that are referenced by import statements. After converting the source into an AST, the front end resolves symbol declarations, does semantic analysis and builds the symbol table and other supporting data structures. The output of the front end is an AST where each node in the AST is annotated with either type or symbol information.

- **Java symbol table design issues**

The symbol table is one of the core data structures in a compiler. Unlike the AST, which can be deleted after the flow graph is built, the symbol table "lives" as long as the Java source is being compiled. Java's scoping and lack of unique names within a scope complicate symbol table construction.

- **The middle pass**

The middle pass performs tree to tree transformations and builds the control flow graph of basic blocks that the optimizer works on. An example of a tree to tree transformations is method in-lining.

- **The optimizer**

The optimizer builds data structures that describe the variable usage throughout the control flow graph for the method (this is usually called global data flow). This information is used to optimize data references globally within a method.

- **The code generator**

The code generator generates instructions for the target processor. The code generation phase also does machine dependent optimization, including peep-hole optimization and load/store scheduling.

Features of JEditor include:

- All the work now takes place from one window.
- Colour syntaxing for Java, JSP, C & HTML files.
- Running the Compiler, Appletviewer, Browser, or Application Viewer from within Jeditor.
- Compile error to line finder.
- Editing features including cut, copy, paste, find, replace, goto line, print, undo, redo.
- Customise your own environment, by specifying your own compiler, Interpreter, and browser, and changing fonts, tab spaces, syntax colours & window placements.
- Options to customise menus, toolbars, popup windows & key accelerators.
- Save warnings before closing windows.
- The facility to load Java, project and HTML files from the command line, using drag and drop or double clicking on the Java or project file.
- Built in HTML Help browser. (Improved from 0.89)
- MethodFinder - A quick method finder.
- Facility to decompile a class straight into an editor window.
- Error line to source finder.

Jeditor - Running/Compiling Documents

Jeditor can now open, edit, compile and run documents from any directory. The default setting are designed to enable a user to edit, save, compile and test files which are all stored within the same directory, and using javac as the default compiler tool.

All the options in the execute menu act upon the current document. The only exception to this rule is when a project is open, and all the relevant project details have been entered to override it.

You can set different classpaths and output directories using Options - Java Options and then click on the paths tab. Again if you set any of these, then they override the defaults.

This feature is useful for people who are working on documents that might be stored in different directories, or where external packages might need to be accessed.

Finally, for even greater flexibility, users can run their own scripts from within Jeditor to perform more awkward tasks that Jeditor might be finding a bit to difficult to handle.

All input and output from running, compiling documents etc will appear in the console window which appears underneath the editor window.

Jeditor - Console Window

The console window is at the bottom right of the main Jeditor window, and shares space with the error window. You can switch between the two windows by clicking on the required tab.

From here you can type commands as you would a DOS box or Unix console window.

The console window also displays input and output from the execute menu.

So far the only builtin internal command that the window recognises is `cls` and this will clear the console screen.

The console window is a works in progress, so bear with me.

Jeditor - Methods Finder

The methods finder is displayed in the bottom left part of the main Jeditor window.

It works by reading the current java document, and displaying the documents methods in its own window. If you wanted to find a particular method in the source code, all you would need to do is click on the relevant method, and it automatically hi-lights the line within your source code in the editor window.

Sometimes the methods window gets a bit lost (This shouldn't happen any more, so let me know if it does) In this case, click on the update methods button - located at the end of the bottom toolbar. This will refresh the methods window.

Another part of the Jeditor project that still needs a lot more development.

The Project Manager window is displayed in the top left part of the Jeditor main window.

In order to use the project manager, options are made available through the buttons on the bottom toolbar of Jeditor or the Project option on Jeditors main menu bar.

To use the project manager, firstly you need to create a new project.

When starting a new project the first thing you will be asked for is a project name. This project name will then be displayed in the project window. Now you are ready to add/remove files you your project and also set-up the project properties (Properties such as classpath, buildpath etc). Remember to save your project when ever you make changes & and always use the .prj extension when doing so. This will make it easier for you to find your projects in future. Once you save your project, you can load it up, and not have to bother setting paths again. You can also double click on the file names to open them up in the editor window.

At the moment, there are not many features built into the project window, this is being worked on.

Jeditor - Code Completer

This is the Code complete dialog box that is opened when you press Ctrl-Space whilst editing text.

To find a word, just type it as if you were typing a word in the editor, when the required word is displayed, press enter, and that word is then inserted into your main edit window.

You can also find words by using the up and down arrows to scroll through the list or use the drop down menu using the mouse.

To close the window without selecting a word either use the mouse to close the window, or press escape.

You can also select a word by double clicking on it with the mouse.

Jeditor - Java Options

The java Options dialog box allows you to set java specific settings. It includes three tabs as shown above.

The tabs allow you to alter the following settings:

Directories Tab

Allows you to set the type of compiler, browser, interpreter, debugger (not fully functional) and decompiler of your choice.

User Paths Tab

Allows you to define a common classpath and output directory for your work.

Options Tab

Allows you to set javac options that you can use to compile your code with.

All the above options are saved as soon as you press the OK button, and will take effect from then on. The options are stored in a file called Jeditor.properties and this file can be found in your home directory.

Jeditor - Jeditor Options

The Jeditor Options dialog box allows you alter settings to the main IDE.

It currently has five tabs as shown above.

More details below:

The Editor tab

This allows you to change the font size and font type of the main Jeditor window (includes the project window, the methods finder window and the editor window. The tab spaces allows you to set the tab to spaces ratio in the main editor. The look and feel allows you to change the look and feel of Jeditor.

Console Tab

Allows you to alter the font type and size of the console window. It also allows you to select a foreground and background colour of the console window, and displays these colour settings within the Console Tab window.

Preferences Tab

Not yet implemented

Syntax Colours Tab

Allows you to customise the colour syntaxing for each of the different types of supported languages, and displays there settings within this window. Current languages supported so far are: Java, JSP, JavaScript HTML and C/C++

ABOUT JAVA

Java has generated more interest in the computer industry, particularly in regard to the Internet and multimedia, than any other product during the summer and early fall of 1995. Its impact on the World Wide Web could be as dramatic as the spreadsheet was for PCs. Some have described it as the great enabling technology for electronic commerce. Others predict it will change the software distribution industry and how software providers or ISVs go to market. Still others say it will help level the playing field relative to that small company based in Redmond. So what exactly is it?

Java is a software technology -- actually a computer language -- available for the Internet (albeit useful on plain vanilla networks or as a standalone language). Developed by Sun, it has some very special properties. With Java, *applications* can be requested by a user over the Internet and run on a local machine. The sender need not know what the user environment looks like in terms of either hardware or software. And Java makes virus transmission almost impossible.

As a language Java looks like a simpler version of C++, so the developer community has a minimal learning curve. What's more, Java is a secure language, the most secure language available on the Internet.

Benefits

Because Java is platform independent, software developers need to create and test only one version of their application -- they don't need to write or maintain separate versions for Macintosh, Unix, NT, Windows 95, etc. In the future, those considering investments in hardware or operating software won't have to worry whether a specific application runs in their particular environment, since Java will run virtually anywhere.

For system administrators and information technology executives, Java simplifies revision control and access control because it requires only one copy of the software in one controlled location. This single application just gets loaded to a user at run time. Process and administration is left to those who do it best, without imposing on the flexibility and individuality that PC and workstation users have grown to love.

A new paradigm is born

For software providers, the Internet represents a free distribution medium. More significantly, it gives them equal "shelf" space to the mega companies, and simplifies release and update distribution.

Some experts see the software sales model changing from a fixed purchase price to a potentially more lucrative per-usage scheme. For example, a user might buy a 50-use package for \$9.99, as opposed to a one-time purchase price

of \$199 that provides unlimited use. Not only will software suppliers likely make more money in the long run, but at the smaller price, bootlegging becomes less attractive.

Structurally the model changes, too. Module packaging becomes different -- the word processor gets separated from the spelling checker and the grammar checker and the graphics piece. If you use the word processor and not the graphics or grammar checker, why pay for the suite?

Software providers also could better control who has their software and gain better access to users without having to pray that the warranty card or fax or dial-up registration is completed. Shipping an updated release would be a thing of the past. The next time the user downloads, they get the upgrade.

This new paradigm has significant implications for electronic commerce, from plotting your portfolios with live data and what-if options to securing a hotel reservation while viewing a walk-through of the various room options. Not surprisingly, the game industry is looking at Java very seriously. Advertising, retail, and transaction-oriented applications will realize significant benefits. To send a new home banking package out to fix a bug or add a feature, companies such as financial institutions used to have to contact all their users. Now such improvements can be implemented automatically.

Finally, for the computer industry, Java technology is open and essentially free. Like the NFS approach that Sun pioneered in the 1980s, choice of hardware and software is left to the user, as it should be.

How does this work?

Instead of being written and compiled in a traditional way (that is, for a particular platform), programs written in Java are platform independent. The resulting code (called an applet) is what gets shipped across the network. One of the special characteristics of Java is that it does not allow pointers outside of its own code. External pointers, which Java prohibits, are a prerequisite for viruses as we know them today. To do damage, code must get out of its own memory space, so a

legitimate Java applet cannot transmit a virus. On the user side sits a program (a HotJava browser, for example) that performs two tasks. First it checks the applet syntax to ensure it's a legitimate Java program. Then it executes the applet, supplying the required information relative to the user's environment. Users need the runtime Java interpreter that checks the code, then executes the applet. The interpreter is installed on the users machine either as a part of a browser that plays the applet or as part of a larger OS.

Key Benefits of Java

Why use Java at all? Is it worth learning a new language and a new platform? This section explores some of the key benefits of Java.

Write Once, Run Anywhere

Sun identifies "Write once, run anywhere" as the core value proposition of the Java platform. Translated from business jargon, this means that the most important promise of Java technology is that you only have to write your application once--for the Java platform--and then you'll be able to run it *anywhere*.

Anywhere, that is, that supports the Java platform. Fortunately, Java support is becoming ubiquitous. It is integrated, or being integrated, into practically all major operating systems. It is built into the popular web browsers, which places it on virtually every Internet-connected PC in the world. It is even being built into consumer electronic devices, such as television set-top boxes, PDAs, and cell phones.

Security

Another key benefit of Java is its security features. Both the language and the platform were designed from the ground up with security in mind. The Java platform allows users to download untrusted code over a network and run it in a secure environment in which it cannot do any harm: it cannot infect the host

system with a virus, cannot read or write files from the hard drive, and so forth. This capability alone makes the Java platform unique.

The Java 2 Platform takes the security model a step further. It makes security levels and restrictions highly configurable and extends them beyond applets. As of Java 1.2, any Java code, whether it is an applet, a servlet, a JavaBeans component, or a complete Java application, can be run with restricted permissions that prevent it from doing harm to the host system.

The security features of the Java language and platform have been subjected to intense scrutiny by security experts around the world. Security-related bugs, some of them potentially serious, have been found and promptly fixed. Because of the security promises Java makes, it is big news when a new security bug is found. Remember, however, that no other mainstream platform can make security guarantees nearly as strong as those Java makes. If Java's security is not yet perfect, it has been proven strong enough for practical day-to-day use and is certainly better than any of the alternatives.

Network-centric Programming

Sun's corporate motto has always been "The network is the computer." The designers of the Java platform believed in the importance of networking and designed the Java platform to be network-centric. From a programmer's point of view, Java makes it unbelievably easy to work with resources across a network and to create network-based applications using client/server or multitier architectures. This means that Java programmers have a serious head start in the emerging network economy.

Dynamic, Extensible Programs

Java is both dynamic and extensible. Java code is organized in modular object-oriented units called *classes*. Classes are stored in separate files and are loaded into the Java interpreter only when needed. This means that an application can decide as it is running what classes it needs and can load them when it needs

them. It also means that a program can dynamically extend itself by loading the classes it needs to expand its functionality.

The network-centric design of the Java platform means that a Java application can dynamically extend itself by loading new classes over a network. An application that takes advantage of these features ceases to be a monolithic block of code. Instead, it becomes an interacting collection of independent software components. Thus, Java enables a powerful new metaphor of application design and development.

Internationalization

The Java language and the Java platform were designed from the start with the rest of the world in mind. Java is the only commonly used programming language that has internationalization features at its very core, rather than tacked on as an afterthought. While most programming languages use 8-bit characters that represent only the alphabets of English and Western European languages, Java uses 16-bit Unicode characters that represent the phonetic alphabets and ideographic character sets of the entire world. Java's internationalization features are not restricted to just low-level character representation, however. The features permeate the Java platform, making it easier to write internationalized programs with Java than it is with any other environment.

Performance

As I described earlier, Java programs are compiled to a portable intermediate form known as byte codes, rather than to native machine-language instructions. The Java Virtual Machine runs a Java program by interpreting these portable byte-code instructions. This architecture means that Java programs are faster than programs or scripts written in purely interpreted languages, but they are typically slower than C and C++ programs compiled to native machine language. Keep in mind, however, that although Java programs are compiled to byte code, not all of the Java platform is implemented with interpreted byte codes. For

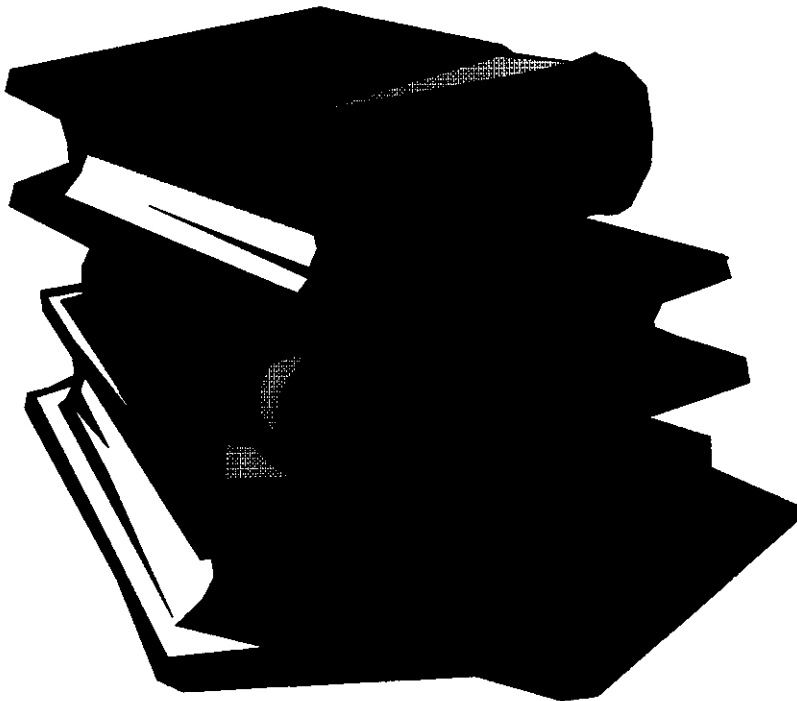
efficiency, computationally intensive portions of the Java platform--such as the string-manipulation methods--are implemented using native machine code.

Although early releases of Java suffered from performance problems, the speed of the Java VM has improved dramatically with each new release. The VM has been highly tuned and optimized in many significant ways. Furthermore, many implementations include a just-in-time compiler, which converts Java byte codes to native machine instructions on the fly. Using sophisticated JIT compilers, Java programs can execute at speeds comparable to the speeds of native C and C++ applications.

Java is a portable, interpreted language; Java programs run almost as fast as native, non-portable C and C++ programs. Performance used to be an issue that made some programmers avoid using Java. Now, with the improvements made in Java 1.2, performance issues should no longer keep anyone away. In fact, the winning combination of performance plus portability is a unique feature no other language can offer.

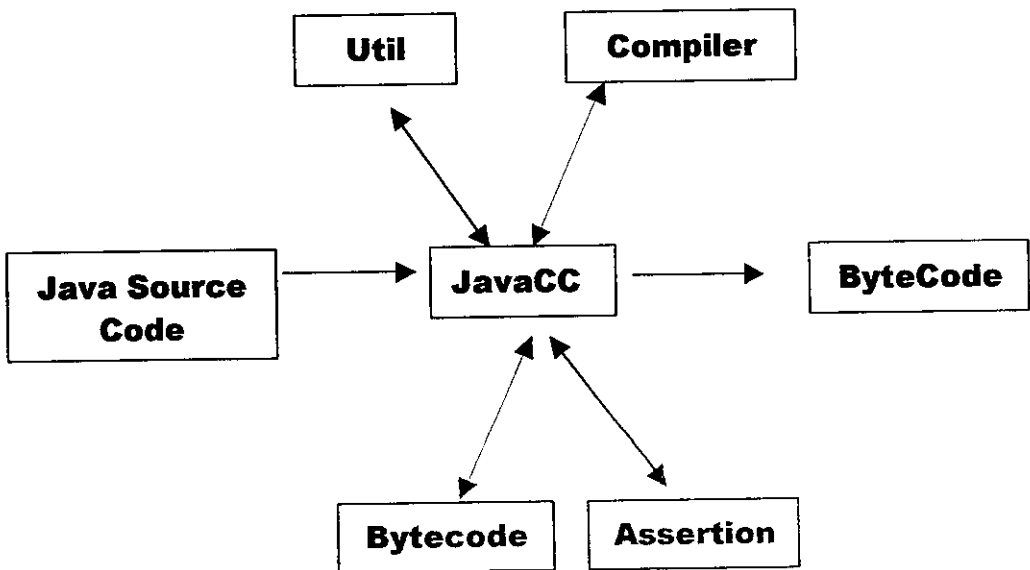
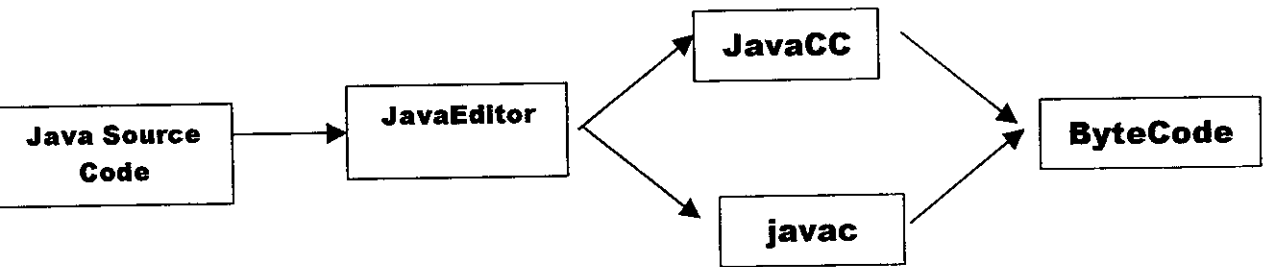
Programmer Efficiency and Time-to-Market

The final, and perhaps most important, reason to use Java is that programmers like it. Java is an elegant language combined with a powerful and well-designed set of APIs. Programmers enjoy programming in Java and are usually amazed at how quickly they can get results with it. Studies have consistently shown that switching to Java increases programmer efficiency. Because Java is a simple and elegant language with a well-designed, intuitive set of APIs, programmers write better code with fewer bugs than for other platforms, again reducing development time.



SYSTEM DESIGN

4.2 PROCESS DESIGN



4.1 INPUT DESIGN

Input design is a part of overall system design, which requires very careful attention. If the data going into the system is incorrect then the processing and output will magnify these errors.

About project data's

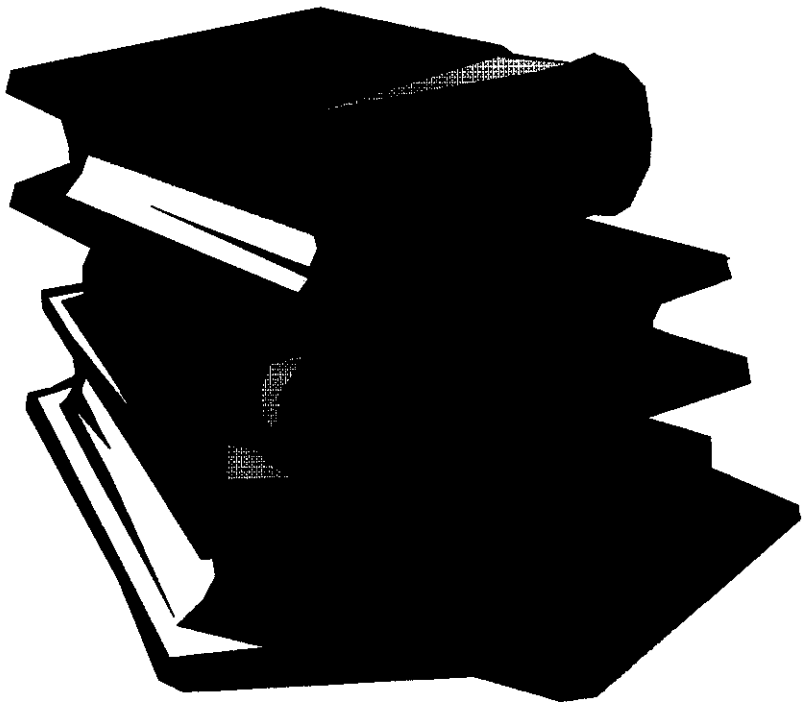
The inputs in the system are of three types:

- 1.External:** which are prime inputs for the system
- 2.Internal:** which are user communications with the system
- 3.Interactive:** which are inputs entered during a dialog with the computer

The above input types enrich the proposed system with numerous facilities that make it more advantageous in comparison with the existing normal system .all the input entered are completely raw, initially, before being entered into a database, each of them availing processing. The input format in this system has been designed with the following objectives in mind.

Intelligent output design will improve systems relationships with the user and help in decision making. Outputs are also used to provide a permanent hardcopy of the results for latter consultations. The most important reason, which tempts the user to go for a new system is the output. The output generated by the system is often regarded as the criterion for evaluating the usefulness for the system. Here the output requirements use to be predetermined before going to the actual system design. The output design is based on the following

- Determining the various outputs to be presented to the user.
- Differentiating between inputs to be displayed and those to be printed.
- The format for the presentation for the outputs.



*SYSTEM IMPLEMENTATION
& TESTING*

5.1 SYSTEM IMPLEMENTATION

Implementation is that stage of the project when the theoretical design is turned into a working system. At this stage of the main workload, the greatest upheaved and the major impact on the existing practices shifts to the under department. A lot of planning has to go in for the successful implementation of the system.

Bearing in mind that implementation is a project in itself; care was taken to develop an effective methodology for implementing the system. The major steps that were carried out in these stages are summarized below:

- Training was given to the user of the system both theoretically as well as practically. They were briefed on the lines on the objectives of the system, how to operate if and the benefits that would be reaped from the system.
- The system as tested in the user's environment and the user was prompted to give his suggestions.
- Existing data's was converted into file structures compatible to the system.
- The strategy used changeover of the system was parallel changeover. The manual system was run parallel along with the automated system to test the validity of the system.

Maintenance issues

Maintenance is the ease with which a program can be corrected if any error is encountered, adapted if its environment changes or enhanced if the customer desires a change in requirements.

The software is characterized by the following activities. In this project considerable amount of time is spent in maintenance and monitoring.

- 1. Corrective maintenance.**
- 2. Adaptive maintenance.**
- 3. Perfective maintenance.**
- 4. Preventive maintenance.**

Corrective maintenance

Corrective maintenance is to uncover the error still exist after testing. During this maintenance work the user is asked to work on the system and if any error is reported.

Adaptive maintenance

The adaptive maintenance is needed if the platform or the environment of the project is to be change. For the project the language takes care of all these things.

Perfective maintenance

The third maintenance activity is perfective maintenance. The recommendation of new capabilities and modification of existing function and

general enhancement are received from the user and proposed for future enhancement.

Preventive maintenance

The preventive maintenance is to improve the future maintainability and reliability and to provide better basis for future enhancement.

5.2 SYSTEM TESTING

Testing is a predominant technique to validate the system developed .the process begins from preparing test plan. The phases in the testing process are that done during implementation to verify the software and one after it to validate the system and to access the reliability of it. We have done both. The test data were provided manually or simulated by writing code for it .we mainly followed a bottom-up approach for testing.

The testing phase, an unavoidable part of software development promotes error detection, a complete verification determining whether the objectives and the user requirements are fulfilled. The system test is based on the given below following

Program Testing

Program testing promotes an error-free program by correcting the syntax and logical error. When a program is tested the actual output is compared with the excepted output. When there is a discrepancy the sequence of instruction must be traced to determine the problem.

Breaking the program down into self-contained portions, each of which can be checked at certain points, facilitates the process. The idea is to compare program values against desk calculated values to isolate the program.

Unit testing

Unit testing is done to check the correctness and validity of modules. Errors are rectified per module and program clarity is increased.

Sequential or series testing

Sequential or series testing is checking the logic of one or more programs in the candidate system, where the out put of one program will effect the processing done by other program

Integration testing

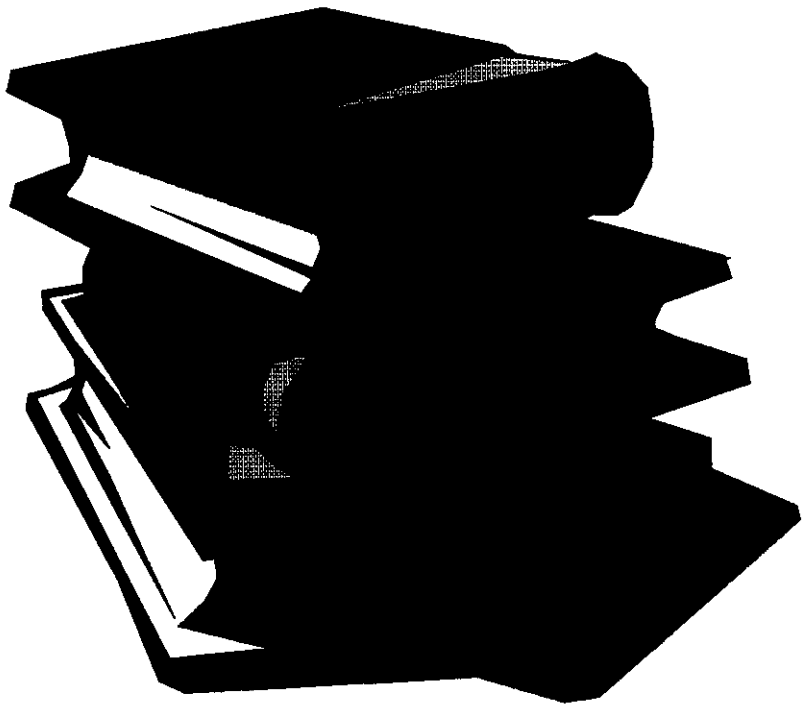
In integration testing all modules are clipped under the major module and tested again to verify the results. A module can have inadvertent, adverse affect on any other or on the global data structures, causing serious problem. A problem arising due to the poor interfacing such as data loss age is corrected in this phase.

System testing

System testing, the final step uncovers the weakness not found in early stages. This involves validation and testing which determines whether the

software functions as the user expects it. Modifications are made so that at the completion phase it satisfied the end-user.

There should be careful planning of how the system will be provoked and the test data designed. The system analyst should be quite clear about the test objectives. System test data can rarely be compressive enough to test the system fully. Some aspects of the system will have to be tested using the live operation.



CONCLUSION

CONCLUSION

The software entitle "JavaCC compiler and Editor" is implemented to replace in Java use preprocessing techniques and translate the source code into an instrumented source code. This approach violates abstraction and modularity. the manual system efficiently .Tthis package designed for the all users to work in GUI,friendly mode in efficient manner.

The developed system is highly interactive one and is user friendly as it is enabled by menu. The system provides accurate updating data validation and integrity is observed in theSystem further extension in the system can be made to submit more reports to the management. This will give the management a clear picture of the proceedings in the company valuables suggestion can be incorporated in the system.

SCOPE FOR FUTURE DEVELOPMENT

The future development of the package is to implement Preferences tab and embedding of jdbc concepts, Bean concepts similar to Jdeveloper.

BIBLIOGRAPHY

Shigeru Chiba ; Javassist – A Reflection-based Programming Wizard for Java ; In Proceedings of OOPSLA'98 Workshop on Reflective Programming in C++ and Java, October 1998.[3]Matt Greenwood ; CFParse ; On-line at

www.alphaWorks.com (search for CFParse).[4]API on-line at

java.sun.com/j2se/1.3/docs/api/java/lang/reflect/Proxy.html[5]From John Rose's work ; Solution at

java.sun.com/products/jfc/tsc/articles/generic-listener2/[6]From John Rose's work ; Solution at

java.sun.com/products/jfc/tsc/articles/generic-listener/[9]Michiaki Tatsubori, Shigeru Chiba ; OpenJava ; On-line at
<http://www.hlla.is.tsukuba.ac.jp/~mich/openjava/4>

The Java Handbook

by Patrick Naughton, Michael Morrison

- Publisher: Osborne/McGraw-Hill
- ISBN: 0-078-82199-1
- Pages: 424
- Price: \$27.95
- Publication Date: April, 1996
- Bottom Line: In Print

Java in a Nutshell: A Desktop Quick Reference for Java Programmers

by David Flanagan

- Publisher: O'Reilly & Associates, Inc.
- ISBN: 1-565-92183-6
- Pages: 460
- Price: \$19.95
- Publication Date: May, 1996
- Bottom Line: Buy It

Thinking in Java

by Bruce Eckel

- Publisher: Prentice Hall
- ISBN: 0-136-59723-8
- Pages: 1152
- Price: \$39.95
- Publication Date: December, 1998
- Bottom Line: Buy It

JavaBeans for Dummies

by Emily Vanderveer

- Publisher: IDG Books Worldwide
- ISBN: 0-764-50153-4
- Price: \$24.99
- Bottom Line: ????

Web Developer's Guide to JavaBeans

by Jalal Fegghi

- Publisher: Coriolis Group
- ISBN: 1-576-10121-5
- Price: \$39.99
- Bottom Line: ????

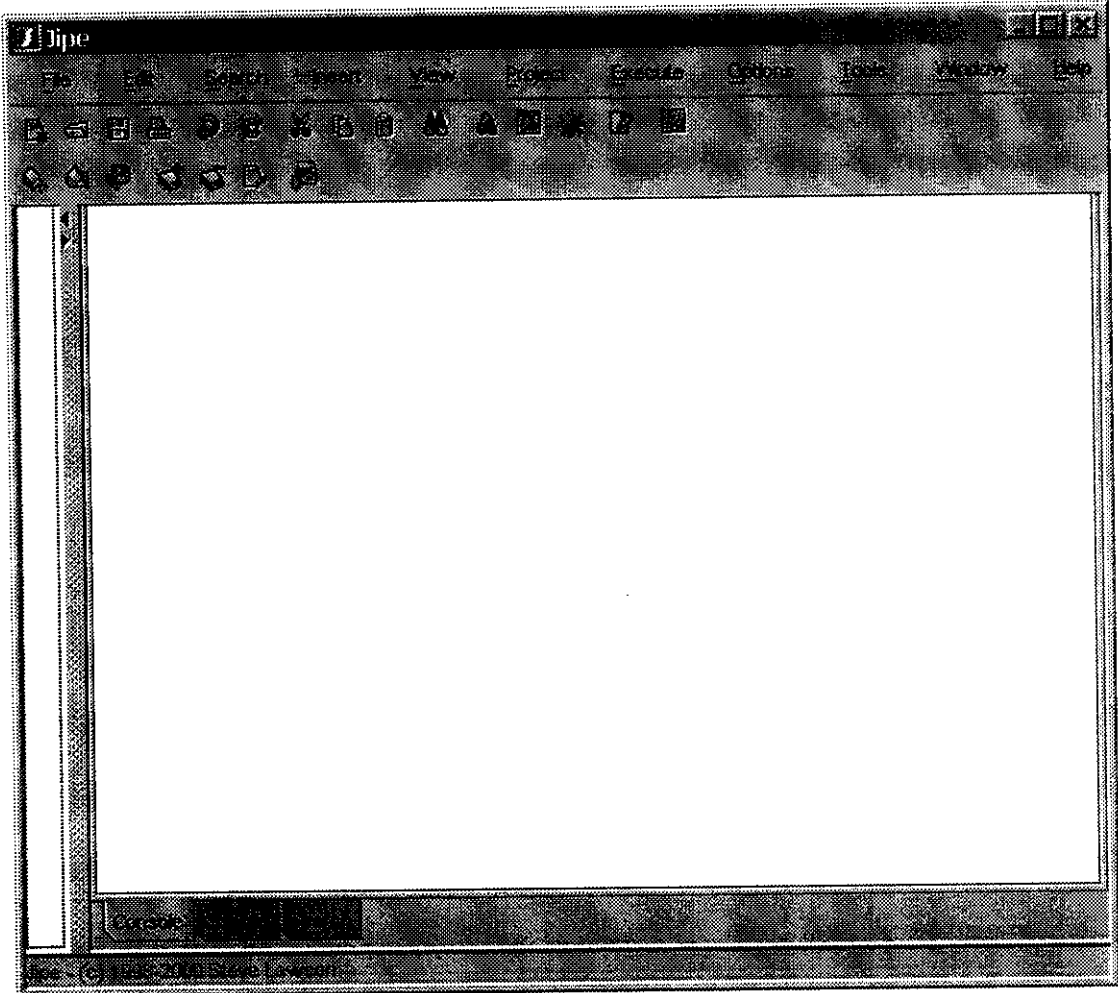
Advanced Java Networking

by Prashant Sridharan, Laraine Peterson, Bill Reiken

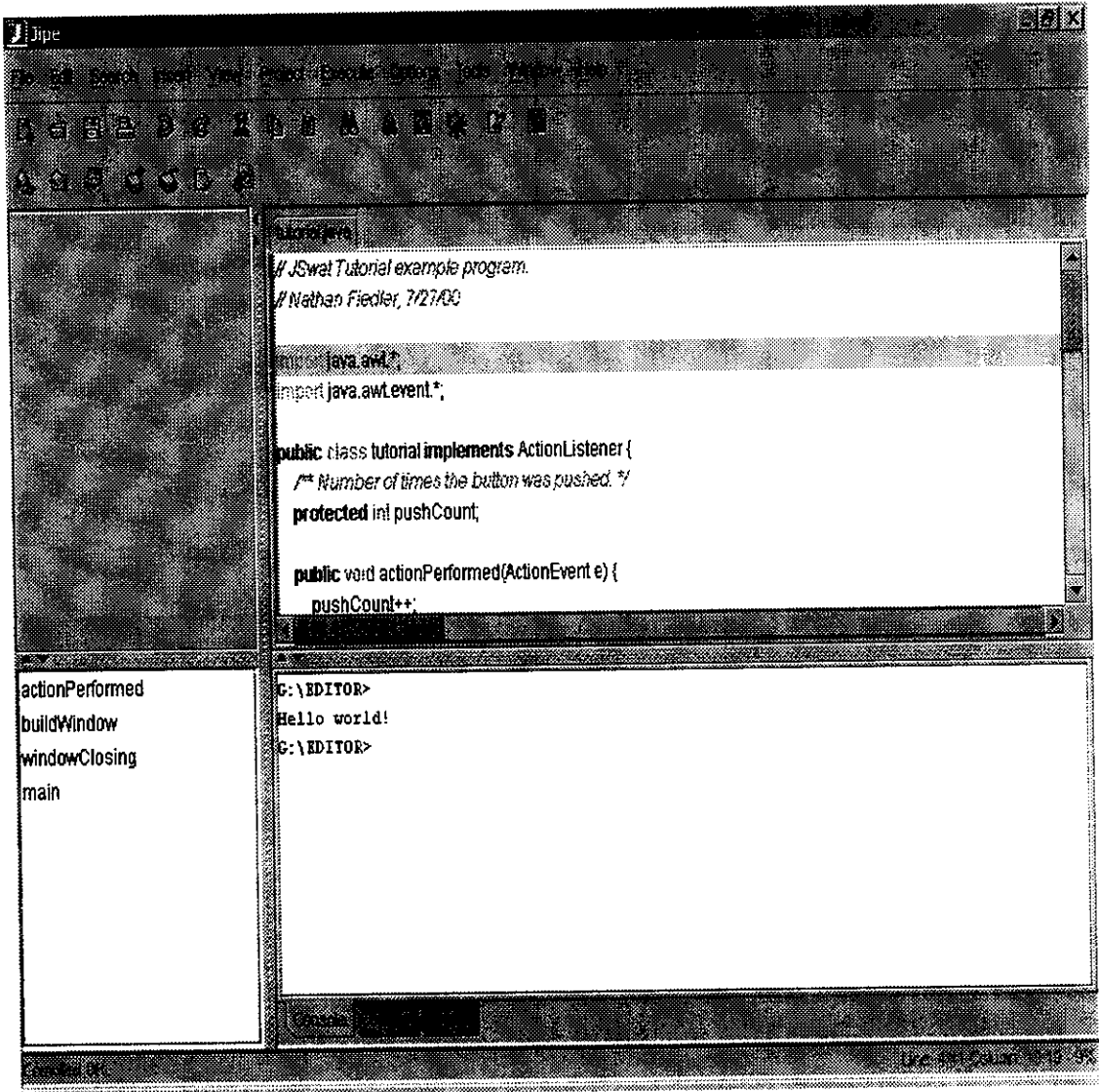
- Publisher: Sunsoft Press/Prentice Hall
- ISBN: 0-137-49136-0
- Pages: 500
- Price: \$49.95
- Publication Date: May, 1997
- Bottom Line: Browse It

SAMPLE SCREENS

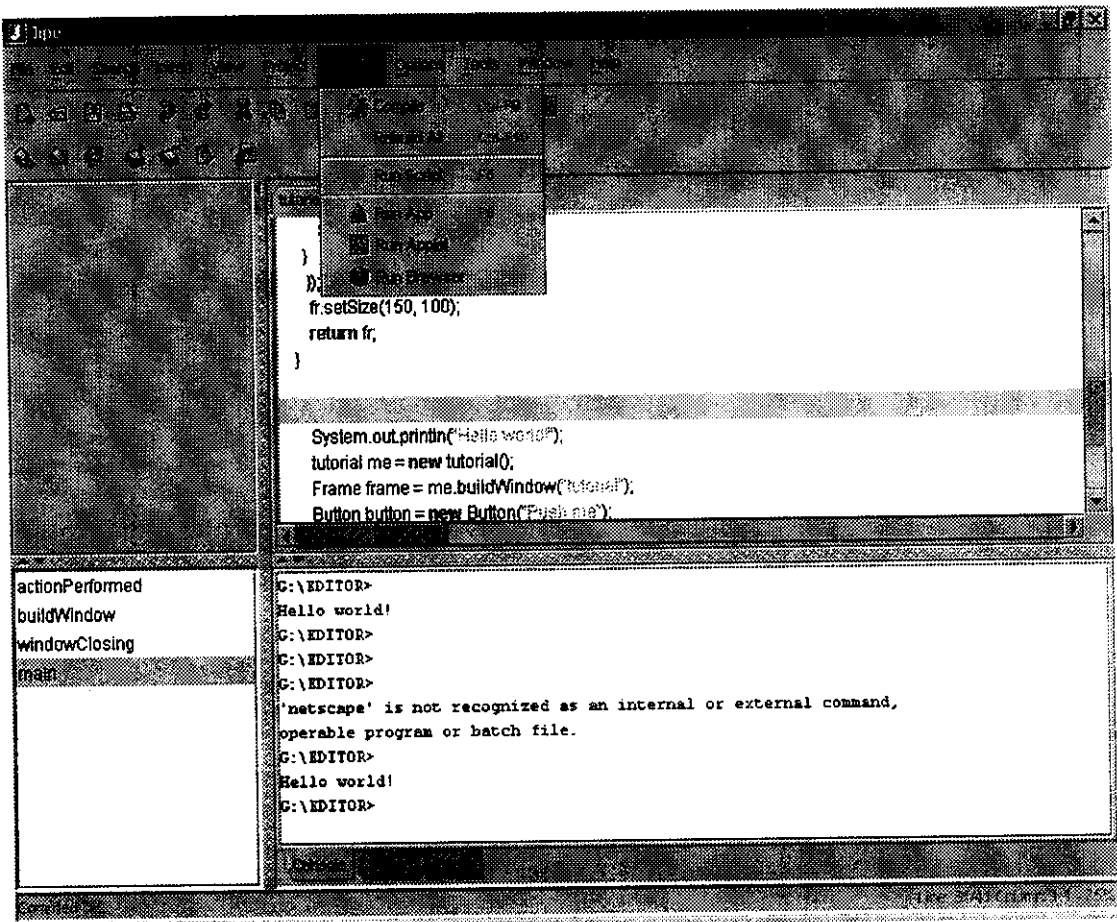
Jeditor Window

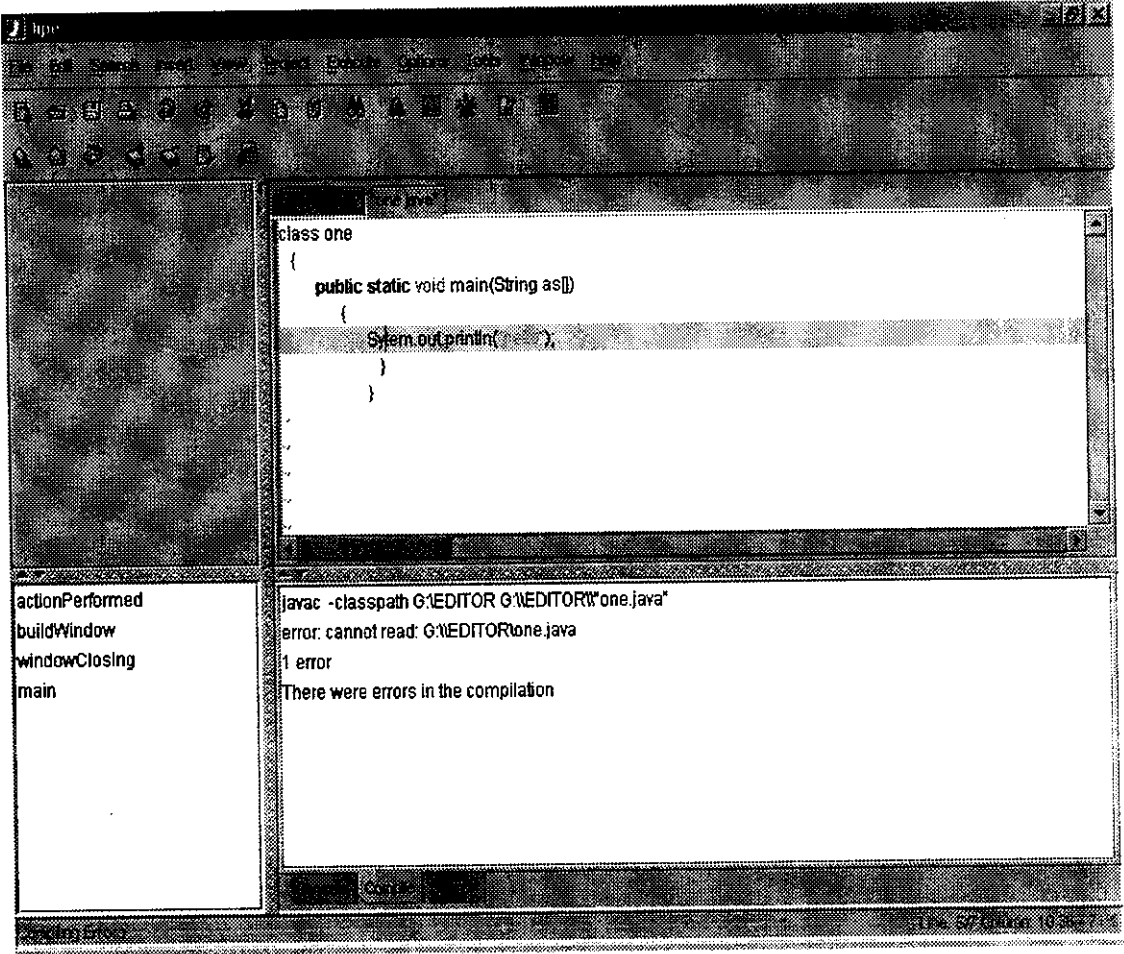


JEDITOR Sample Program

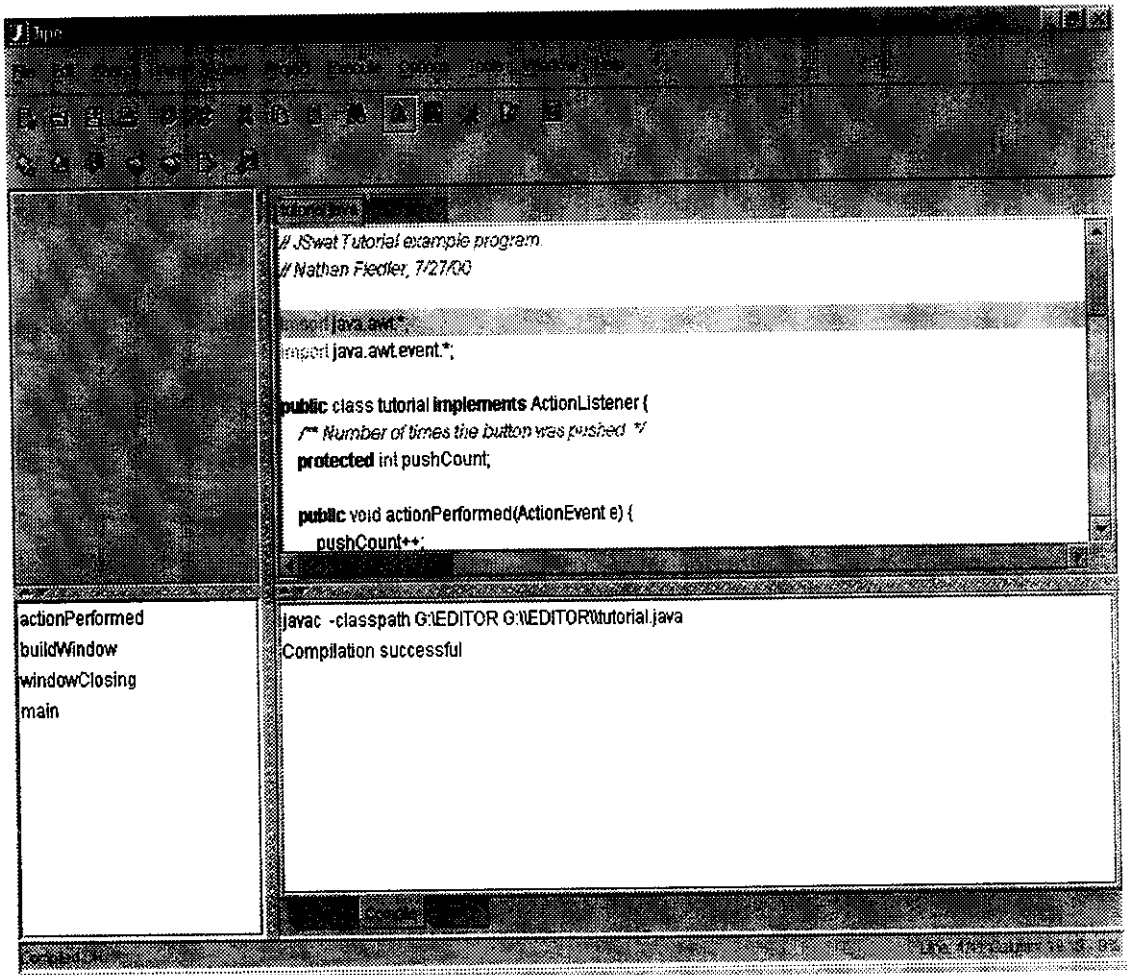


Compiling the java Program and Error Dedcuting

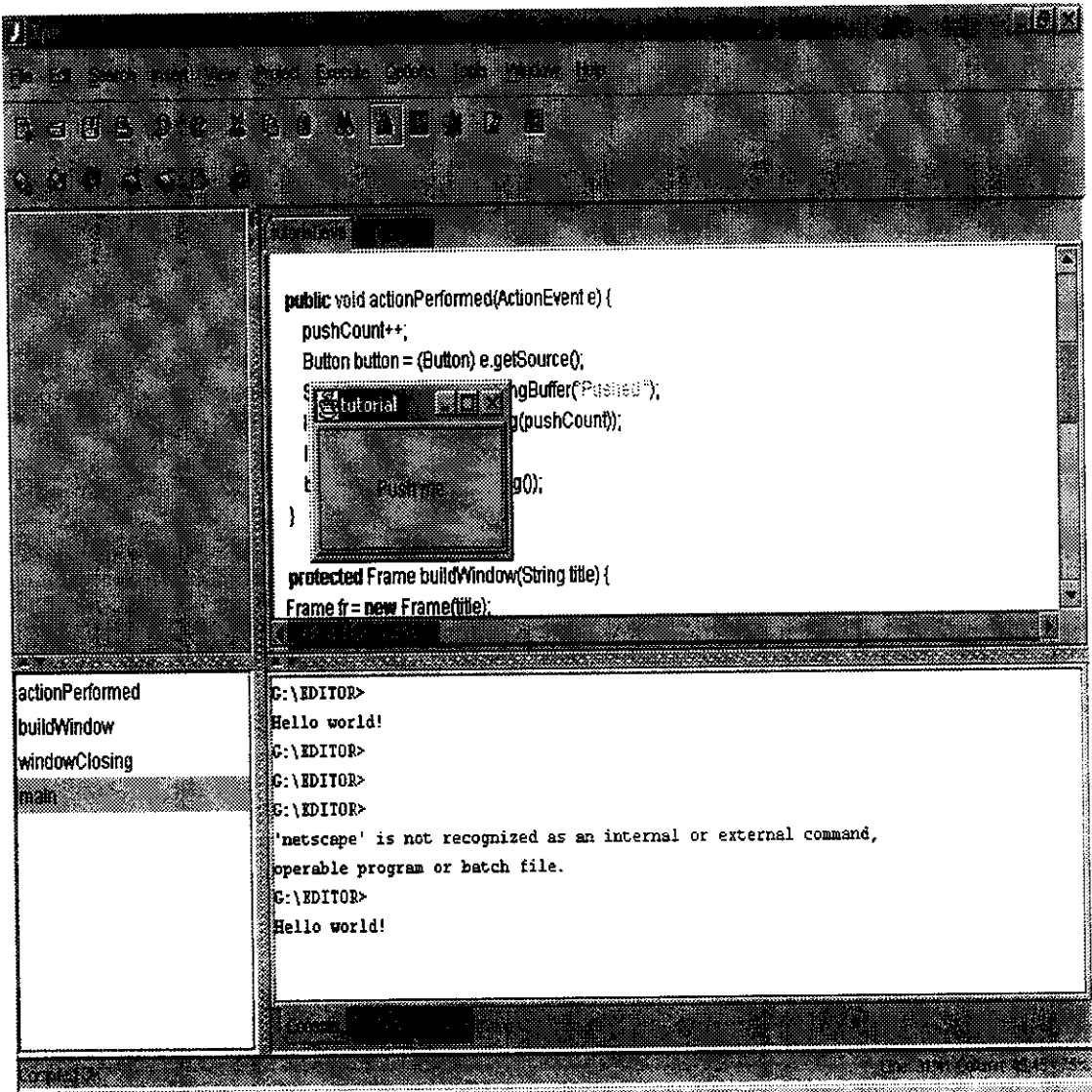




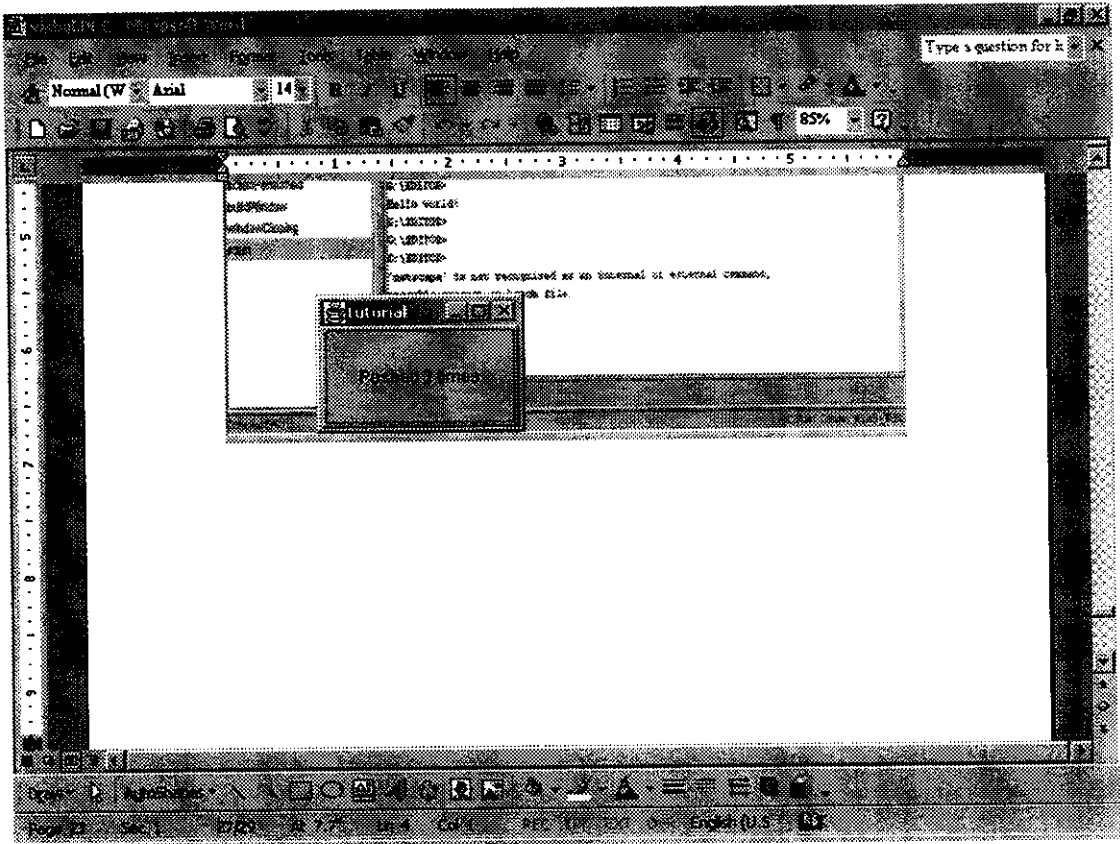
A successful Compiled Program



Executing the program



Output



Opening File Dialog Box

