P- 1446

SOURCE CODE PERFORMANCE ANALYSIS TOOL

By

B.ARUN

Reg. No 71202621006

Of

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements

for the award of the degree

Of

MASTER OF COMPUTER APPLICATION

June, 2005

BONAFIDE CERTIFICATE

Certified that this project report titled **Source Code Performance Analysis Tool** is the bonafide work of **Mr. B.Arun** who carried out the research work under my supervision. Certified further, that to the best part of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

[illegible]
Faculty Guide

[illegible]
Head of the Department

Submitted for the viva-voce examination held on 24-06-2005

[illegible]
Internal Examiner

[illegible]
External Examiner
24/6/05

ABSTRACT

"Source Code Performance Analysis Tool is a GUI-based automized tool to scrutinize the performance of Java applications, test the application for code construction and code integrity, and to view the current system specification. The objective of the tool is the faster execution of application and the efficient usage of memory. Performance analysis is looking at program execution to pinpoint the performance problems and locate the code that needs tuning. The core objective of the tool is to make the application to execute faster and to efficiently utilize the memory. The tool is designed in the environment JAVA 2.0 and JAVA SWINGS, wherein it enables the tool to be machine and platform independent.

The tool analyzes the performance of the application and identifies the flaws in the application that leads to performance degradation. The tool is designed to measure the performance of Java based applications because when compared to other languages the Java Application consumes more time for execution as it is an interpreted language. The performance is measured based on factors like execution time and memory usage.

The tool also provides the system specification information, as the performance of the application also depends on the hardware configuration. This helps in comparing the performance of the application executed in different existing system configuration.

The tool automatically performs White-box and Regression testing. It tests the Java Application for code construction and code integrity. Thus the tool reduces testing, debugging and maintenance time.

ACKNOWLEDGEMENT

I am extremely indebted to our chairman, **Dr. N. Mahalingam** for providing excellent environment and infrastructure at Kumaraguru College of Technology, Coimbatore, towards the degree Master of Computer Application. I thank our correspondent, **Dr. K.Arumugam** and our principal **Dr. K.K. Padmanabhan** for providing the motivation and encouragement in making the project a truly successful one.

I convey my sincere regards to our HOD, **Dr. S. Thangasamy**, course coordinator **Mr. A. Muthukumar**, faculty members for their constant guidance and support. My humble regards to my project guide **Mr. R.Rajasehar** for his valuable guidance.

I owe a debt of gratitude to **Mr. V.K.Senthil**, CEO (Logic Version Technologies Ltd.) and **Mr. R.Senthil Kumar**, Project Leader (Logic Version Technologies Ltd.) and all other staff members of **LVT**, Coimbatore for allowing me to use the facilities of the organization and providing me all the information, suggestion and improvements at every stage of the project, which enabled me to complete it in time. I extend my sincere thanks to all my **Parents & Friends** who helped me in my project by sharing their views and ideas.

TABLE OF CONTENTS

S. NO.	TABLE OF CONTENTS	PAGE NO.
	ABSTRACT	iii
	LIST OF FIGURES	viii
	LIST OF ABBREVIATION	ix
1	**INTRODUCTION**	**1**
	1.1 ORGANIZATION PROFILE	1
	1.2 PROBLEM DEFINITION	2
2	**SYSTEM ANALYSIS**	**4**
	2.1 EXISTING SYSTEM	4
	2.2 PROPOSED SYSTEM	5
	2.3 DESIGN ISSUES	6
	2.4 USER INTERFACE REQUIREMENT	6
	2.5 CONTEXT LEVEL DIAGRAM	8
	2.6 DATA FLOW DIAGRAMS	9
3	**SYSTEM DESIGN**	**20**
	3.1 SYSTEM DESIGN	20
	3.1.1 Performance Analysis	21
	3.1.1.1 Used & Unused Variables	22
	3.1.1.1.1 Used Variables	22
	3.1.1.1.2 Unused Variables	23
	3.1.1.2 Loop Optimization	23
	3.1.1.3 Classes & Methods	24

	3.1.1.5 File Compiler	24
	3.1.2 System Configuration Details	24
	3.1.2.1 Memory Capacity & RAM Details	25
	3.1.2.2 Operating System & JVM Details	26
	3.1.2.3 Drive Details	26
	3.1.2.4 Help	26
	3.1.3 Testing	26
	3.1.3.1 White Box Testing	27
	3.1.3.2 Regression Testing	28
4	**SYSTEM CONFIGURATION**	**29**
	4.1 HARDWARE CONFIGURATION	29
	4.2 SOFTWARE CONFIGURATION	29
5	**TESTING**	**30**
	5.1 UNIT TESTING	30
	5.2 INTEGRATION TESTING	31
	5.3 SYSTEM TESTING	31
	5.3.1 Stress Testing	32
6	**CONCLUSION**	33
	6.1 FUTURE ENHANCEMENTS	34
	APPENDICES	**35**
	A.1 SCREEN SHOTS	35
	REFERENCES	**47**

LIST OF FIGURES

Figure Number	**Figure Name**	**Page Number**
2.1	Context Level Diagram	8
2.2	Data Flow Diagram Level 1	9
2.3	Data Flow Diagram Level 2 (Performance Analysis)	10
2.4	Data Flow Diagram Level 2 (System Details)	11
2.5	Data Flow Diagram Level 2 (Testing)	12
2.6	Data Flow Diagram Level 3 (Used & Unused Variables)	13
2.7	Data Flow Diagram Level 3 (Classes & Methods)	14
2.8	Data Flow Diagram Level 3 (Execution Time)	14
2.9	Data Flow Diagram Level 3 (Memory Details)	15
2.10	Data Flow Diagram Level 3 (Loops)	16
2.11	Data Flow Diagram Level 3 (Drive Details)	17
2.12	Data Flow Diagram Level 3 (OS Details)	17
2.13	Data Flow Diagram Level 3 (Testing)	18
2.14	Data Flow Diagram Level 3 (Testing)	18
2.15	Structure Chart (SPAT)	19

LIST OF ABBREVIATION

1.	SPAT	- Source code Performance Analysis Tool
2.	GB	- Giga Bytes
3.	MB	- Mega Bytes
4.	RAM	- Random Access Memory
5.	JVM	- Java Virtual Machine
6.	URL	- Uniform Resource Locator
7.	TEM	- Total Expanded Memory
8.	FEM	- Free Expanded Memory

CHAPTER 1

INTRODUCTION

1.1 ORGANIZATION PROFILE

Logic Version Technologies (LVT) was established in 1989 to provide a full range of services in Information Technology (IT) to clients around the world. Logic Version Technologies has earned a reputation for providing high quality solution and cost effective services to clients who need accurate detailed and timely IT information. LVT is emerging as a full-fledged dynamic and reliable technological solution provider.

'Entrust yourself to Logic Version and Enhance your growth process !' a phrase which appropriately identifies Logic Version Technologies – a multifaceted, IT training and solutions company. They believe in providing clients the competitive edge in a rapidly growing IT scenario through development of value laden, customized software products and services to suit their specific requirements and in the process add value to their client's business.

Services

Logic Version Technologies believe that extensive product knowledge as well as engineering expertise are essential for tailoring the solution to the customer's needs. Thus we provide strong project management, excellent support and effective guidance that always leave our customers in complete satisfaction. Services include specialization in height availability systems and business continuance solutions for medium to large clients that are running huge data base and other mission critical applications.

1.2 PROBLEM DEFINITION

In the current system the performance of Java programming is analyzed or tested manually. Hence it exhibits a list of limitations that has keyed for the development of a new alternative that is our proposed system. Some of the limitations that are visible in the current system and manual processing systems are listed below.

- Time overhead and time consumption is more due to manual processing
- Higher chances of missing some conditions in the application
- Complete analysis of the program is not sometimes possible
- Manual testing is a complex process
- The performance and testing tool exist individually
- The cost factor is very high for the existing system
- Manual operation may leave certain simple flaws undetermined which may later rise as a complex problem
- The tools are not user friendly

- The analysts or programmers alone check for the efficiency which may not be flexible for clients or end users

CHAPTER 2

SYSTEM ANALYSIS

System analysis is the detailed investigation of the component parts of a whole system and their relations in making up the whole. System analysis was conducted to identify the various components of the proposed system and the manner in which these components would communicate with each other.

The lists of documents that need to be generated during the analysis phase are:

- Functional Specification
- Detailed Design Documentation
- Unit Test Plan

2.1 EXISTING SYSTEM:

Performance of the java application is usually done manually by the programmer or the analyst .The manual processing takes more time and it is not possible to analyze accurately. The testing process is also done manually. The debugger performs debugging only about 30% of the testing. Time consumption is more and also the application is not completely tested. Automated testing tools are also available. The existing system is designed such that they either perform for the analysis of Java applications or perform testing as a whole. The existing tool cost is very high and could not be affordable to very small java applications. The available systems are useful and applicable to large sized programming and very well standardized applications and hence could not fit through all small programs.

In the existing system any Java application is analyzed or tested manually. Here the performance of any Java application is analyzed manually. As noted earlier only about 30% of the performance analysis is often completed when carried out manually. The testing performed manually consumes more time and paves way to leave certain flaws undetermined. These undetermined flaws could result in a serious limitation when implemented. Also in the current system the system configuration details of system are to be known only by individual programming that the user or programmer has to code for every single detail. A tool which provides the system configuration details with performance analysis and testing of Java application does not exist.

2.2 PROPOSED SYSTEM:

Source Code Performance Analysis Tool (SPAT) is a GUI-based tool for analyzing the performance of Java applications. Performance analysis is looking at program execution to pinpoint the performance problems and locate the code that needs tuning. The objective of the tool is the faster execution of application and the efficient usage of memory.

The tool primarily looks at how to make Java run faster rather than making it fit into a smaller space. The tool mainly focuses on the core language rather than on the Java Application Program Interfaces - APIs.

The tool generates the reports on the following,

- Performance of the Application
- System Configuration Details
- Testing

2.3 DESIGN ISSUES:

- The Tool must provide the used and unused variable details
- The Tool must provide execution time details
- The Tool must provide class and method details
- The Tool must provide loop details
- The Tool must provide memory and drive details
- The Tool must provide help for new users

2.4 USER-INTERFACE REQUIREMENTS:

- The tool provides a very simple user interface to navigate for end-users
- It is designed to suit all sort of user from experts to beginners

- The user interface strikes the right balance between simplicity and sophistication
- New users to the tool can access the help feature to learn about the tool and the use of the system

2.5 Context Level Diagram:

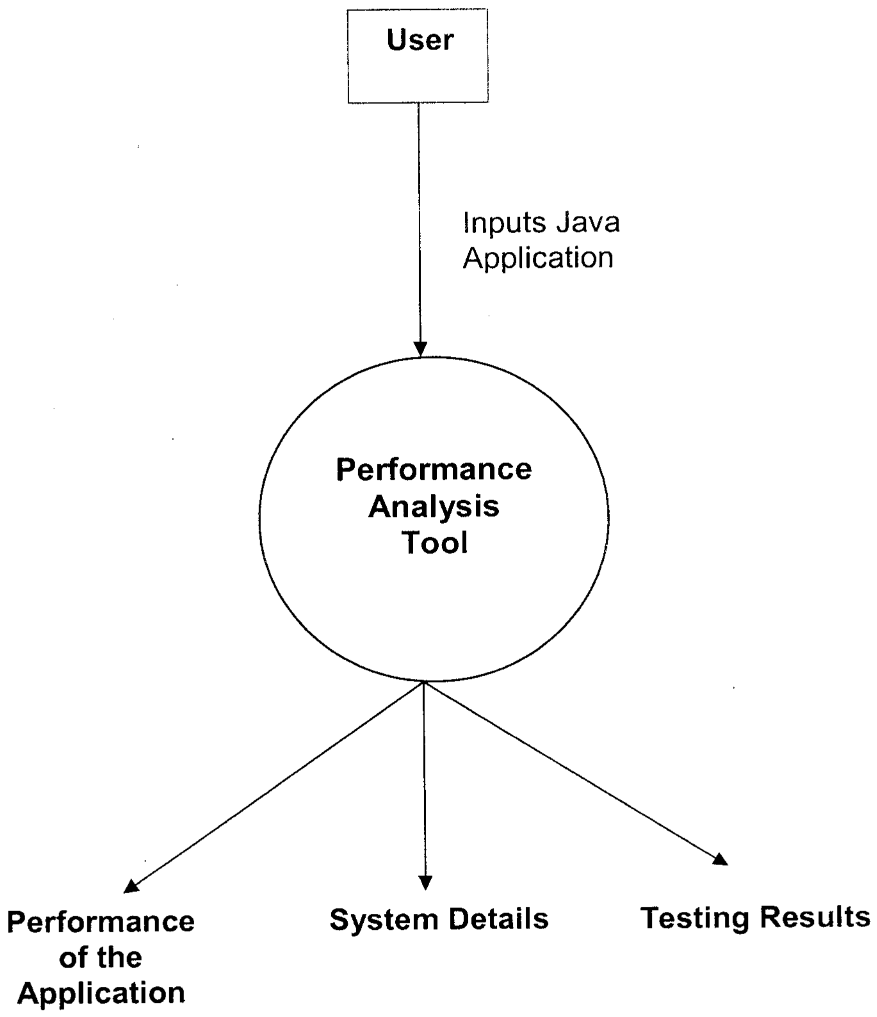


Figure 2.1 – Context Level Diagram

2.6 DATA FLOW DIAGRAMS:

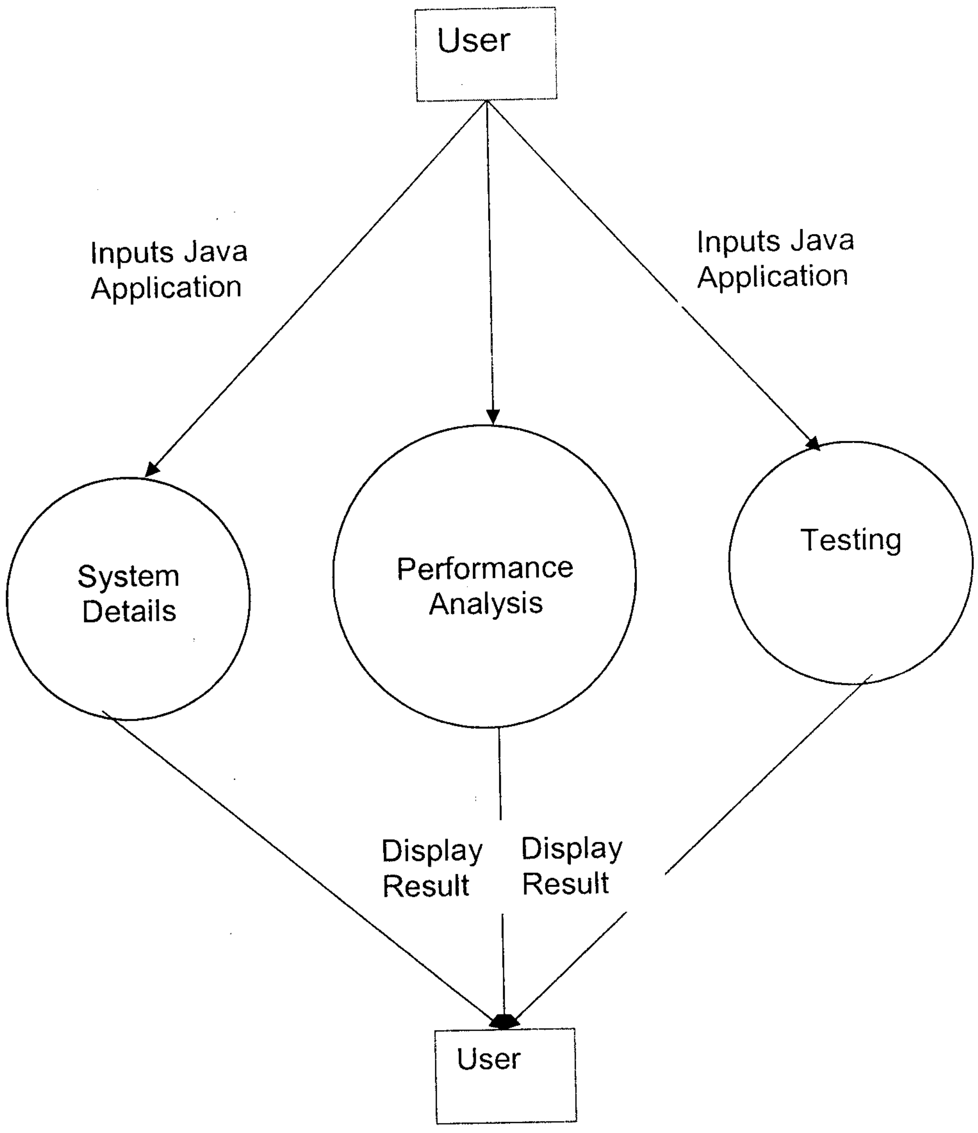


Figure 2.2 Data Flow Diagram Level 1

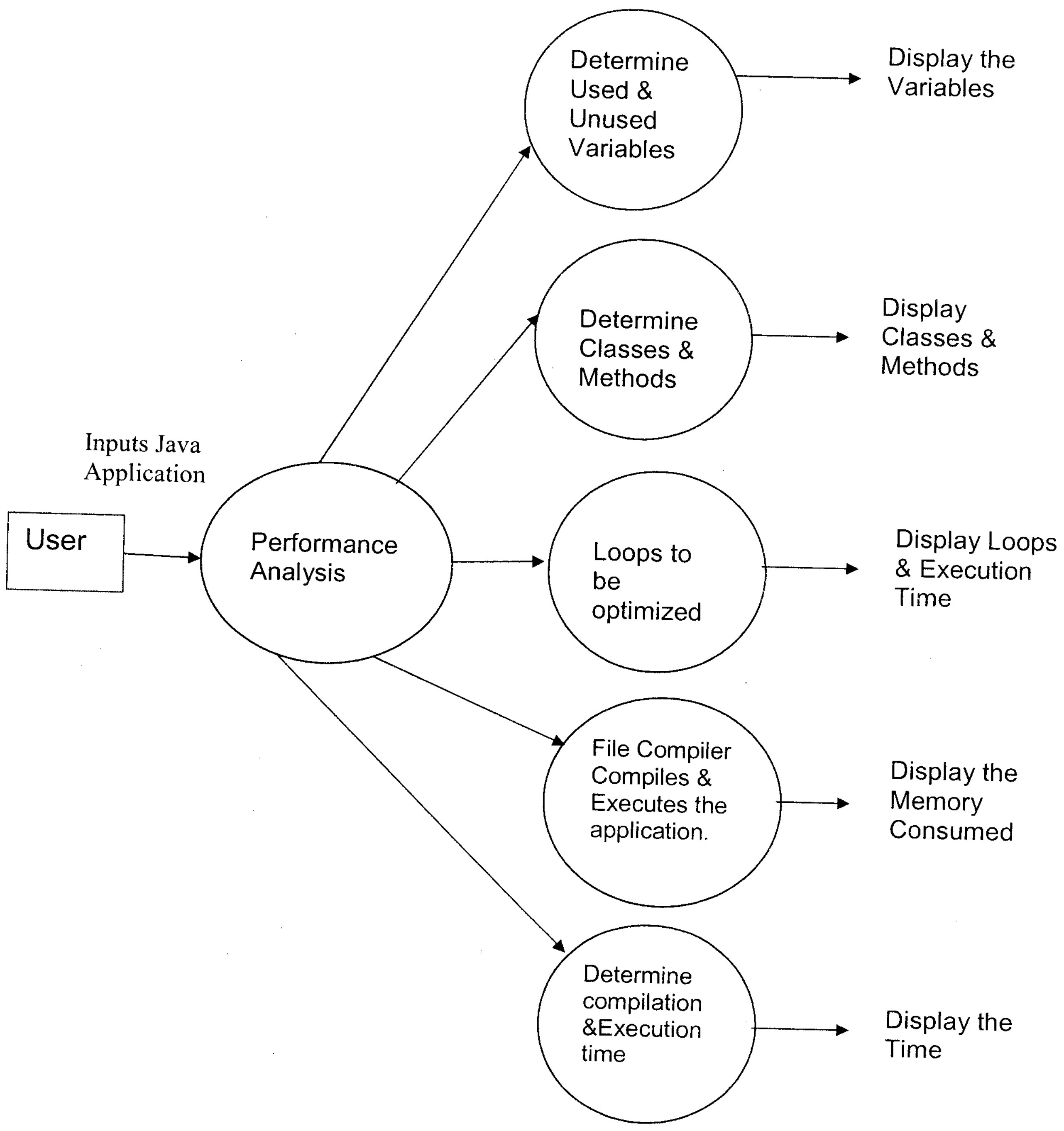

Inputs Java Application
User
Performance Analysis
Determine Used & Unused Variables
Display the Variables
Determine Classes & Methods
Display Classes & Methods
Loops to be optimized
Display Loops & Execution Time
File Compiler Compiles & Executes the application.
Display the Memory Consumed
Determine compilation &Execution time
Display the Time

Figure 2.3 Data Flow Diagram Level 2(Performance Analysis)

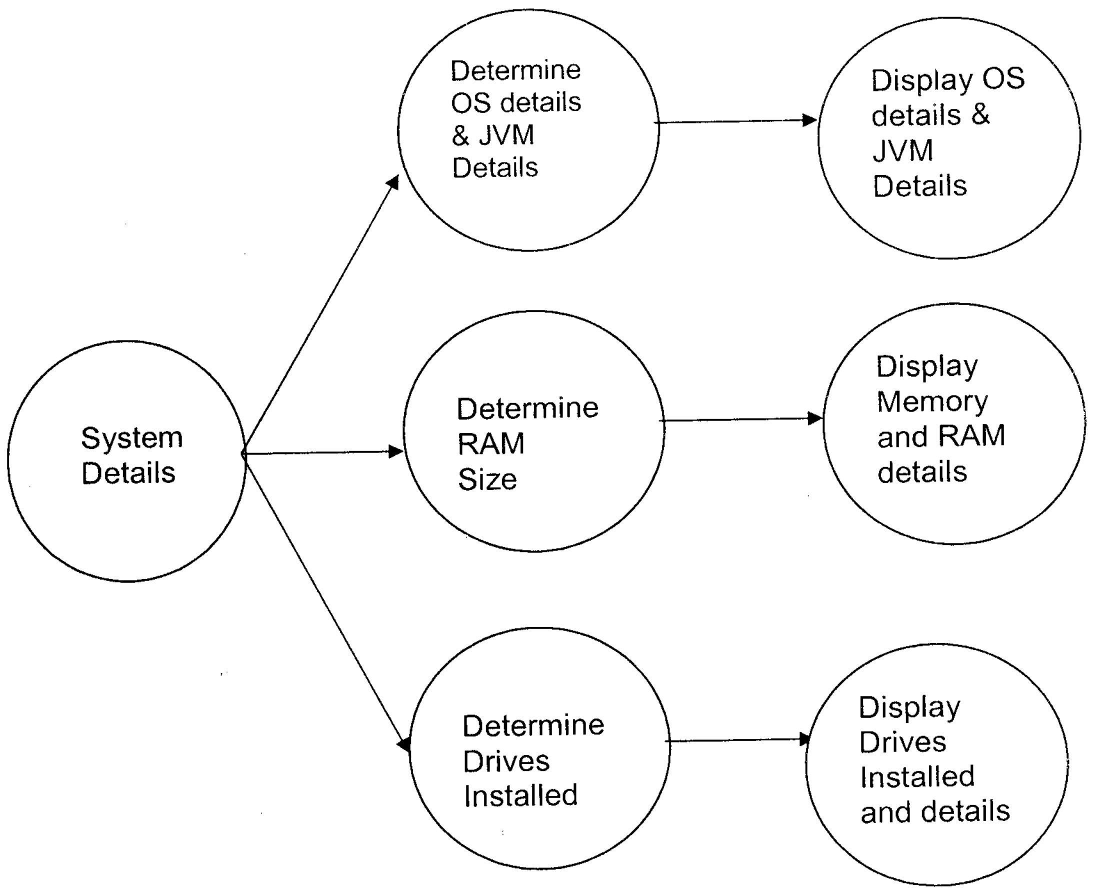

System Details
Determine OS details & JVM Details
Determine RAM Size
Determine Drives Installed
Display OS details & JVM Details
Display Memory and RAM details
Display Drives Installed and details

Figure 2.4 Data Flow Diagram Level 2(System Details)

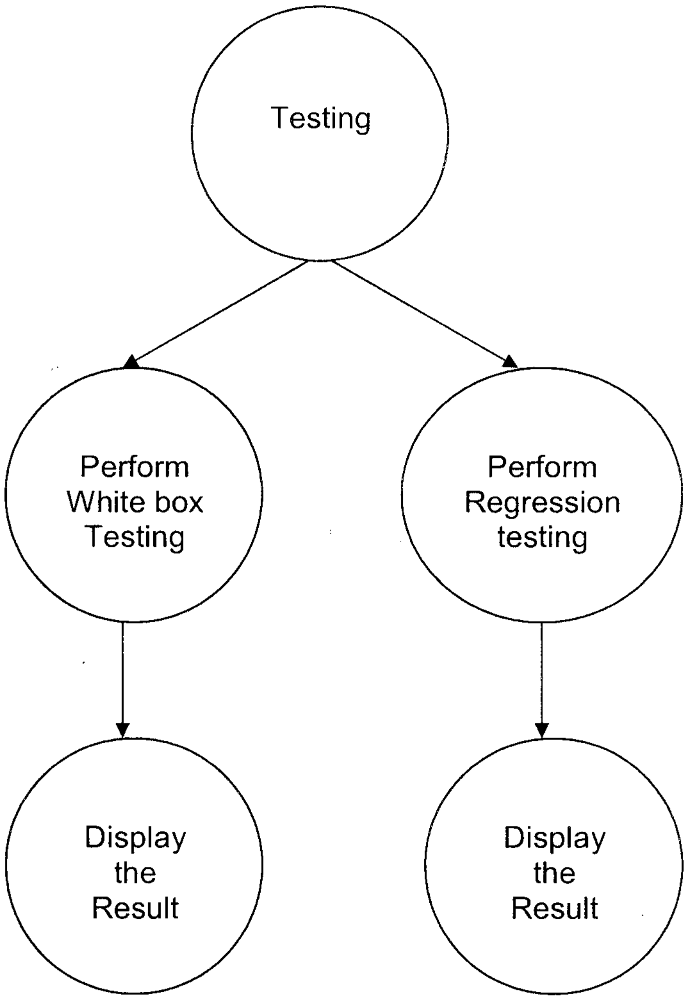


Figure 2.5 Data Flow Diagram Level 2(Testing)

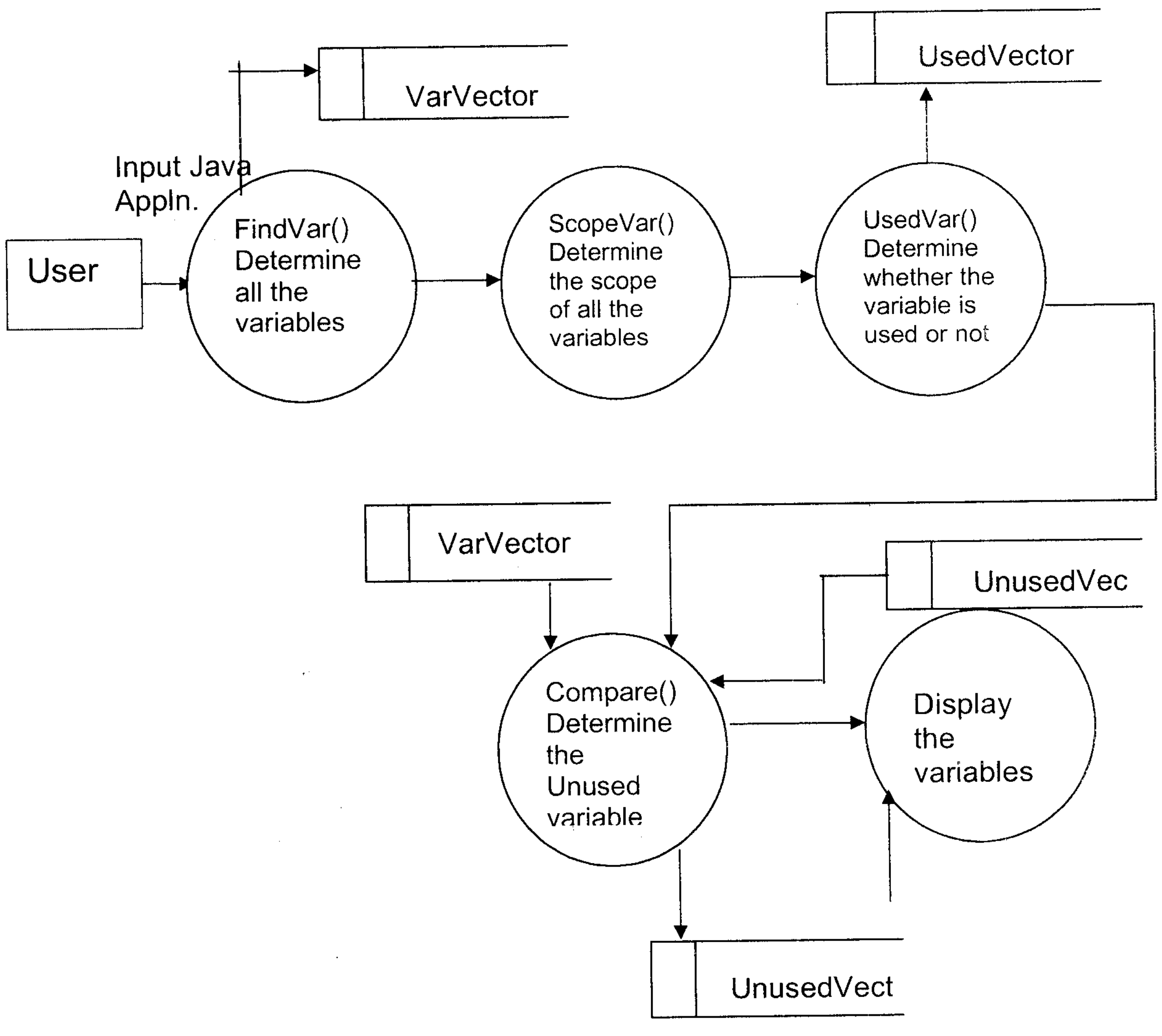

VarVector
UsedVector
Input Java Appln.
User
FindVar() Determine all the variables
ScopeVar() Determine the scope of all the variables
UsedVar() Determine whether the variable is used or not
VarVector
UnusedVec
Compare() Determine the Unused variable
Display the variables
UnusedVect

Figure 2.6 Data Flow Diagram Level 3

(Used & Unused Variables)

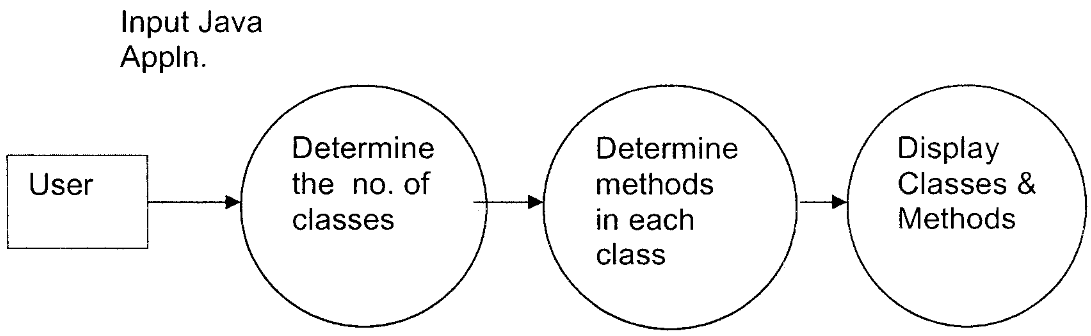

Input Java
Appln.
User
Determine the no. of classes
Determine methods in each class
Display Classes & Methods

Figure 2.7 Data Flow Diagram Level 3(Classes & Methods)

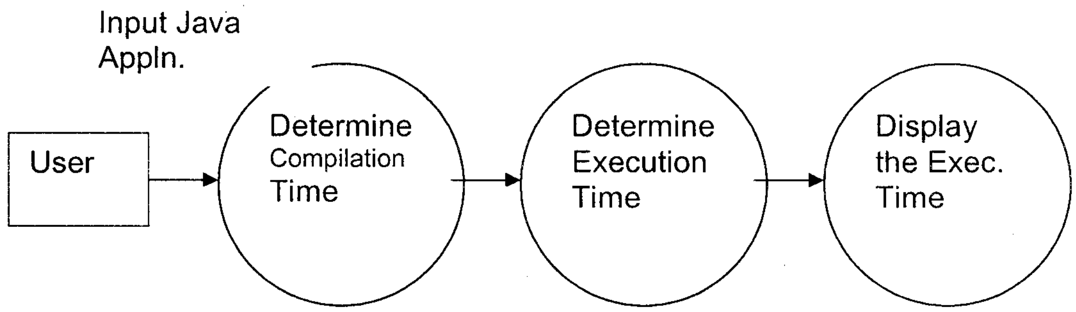


Figure 2.8 Data Flow Diagram Level 3(Execution Time)

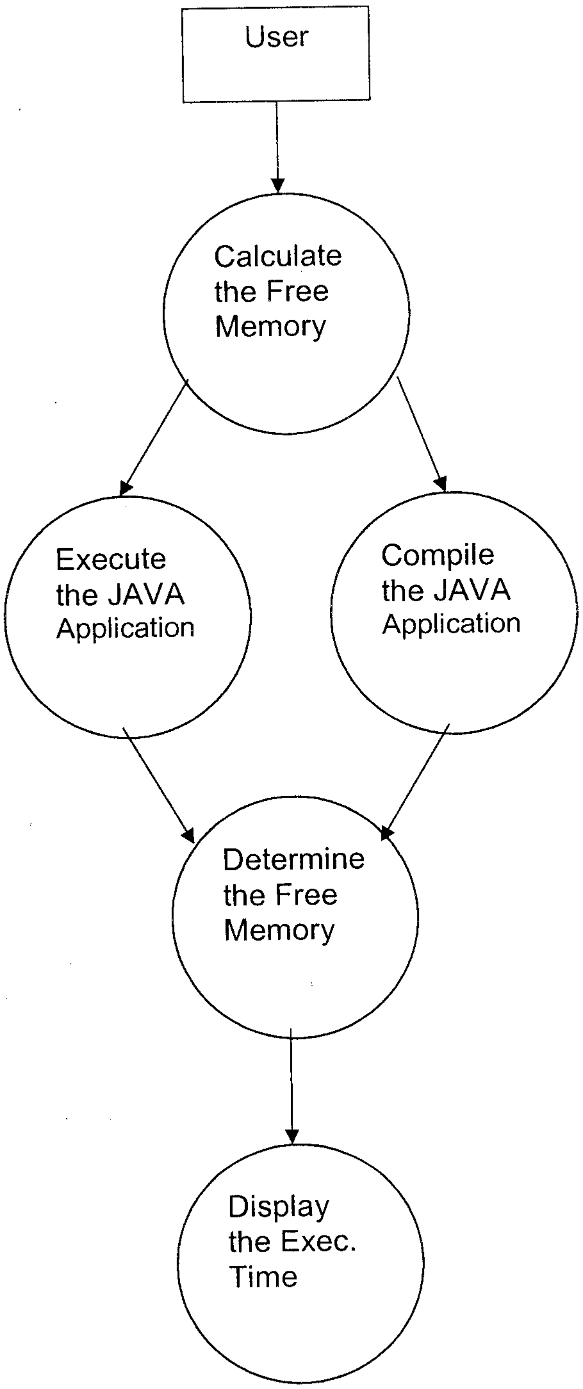

User
Calculate the Free Memory
Execute the JAVA Application
Compile the JAVA Application
Determine the Free Memory
Display the Exec. Time

Figure 2.9 Data Flow Diagram Level 3(Memory Details)

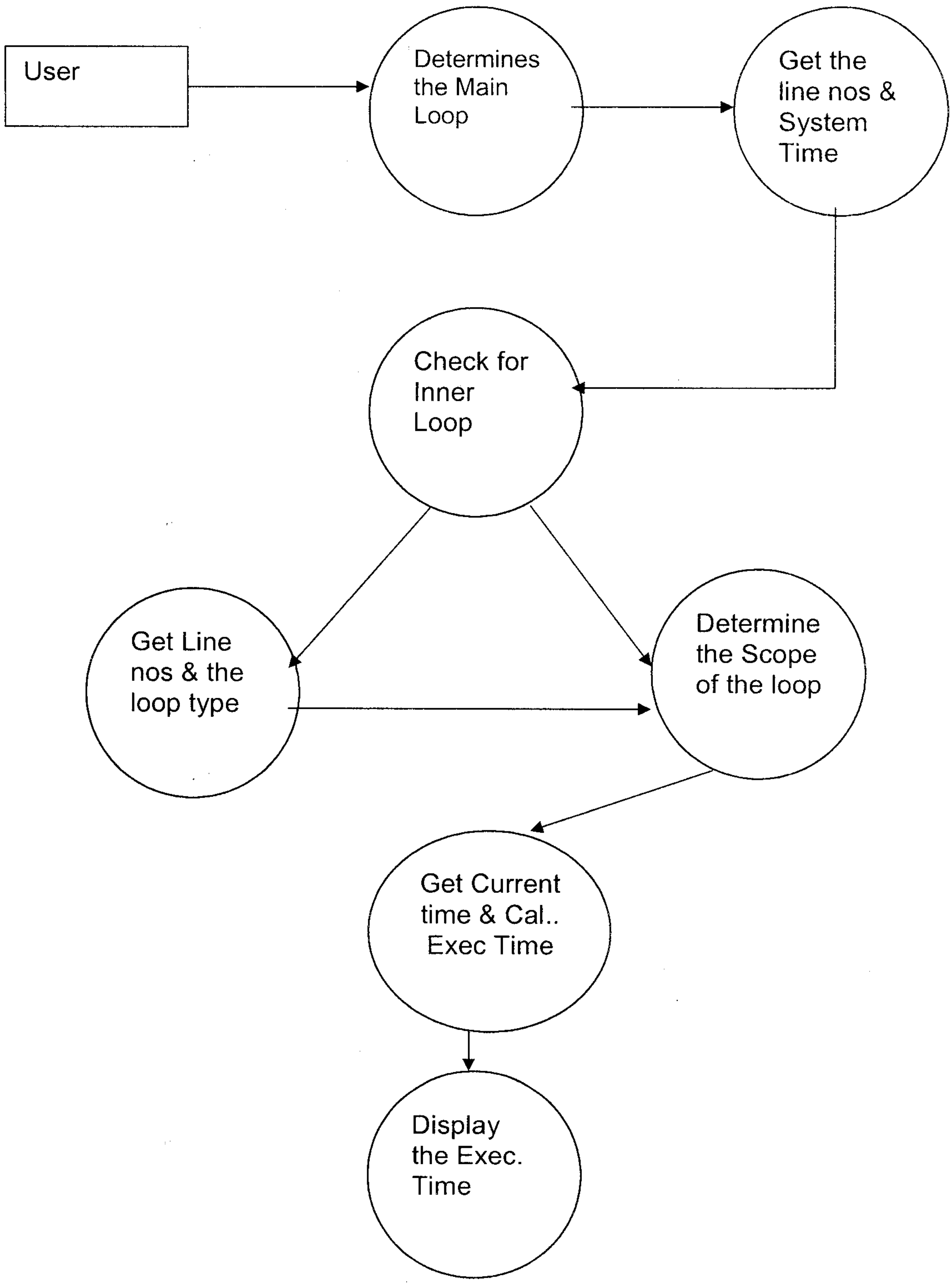

User
Determines the Main Loop
Get the line nos & System Time
Check for Inner Loop
Get Line nos & the loop type
Determine the Scope of the loop
Get Current time & Cal.. Exec Time
Display the Exec. Time

Figure 2.10 Data Flow Diagram Level 3(Loops)

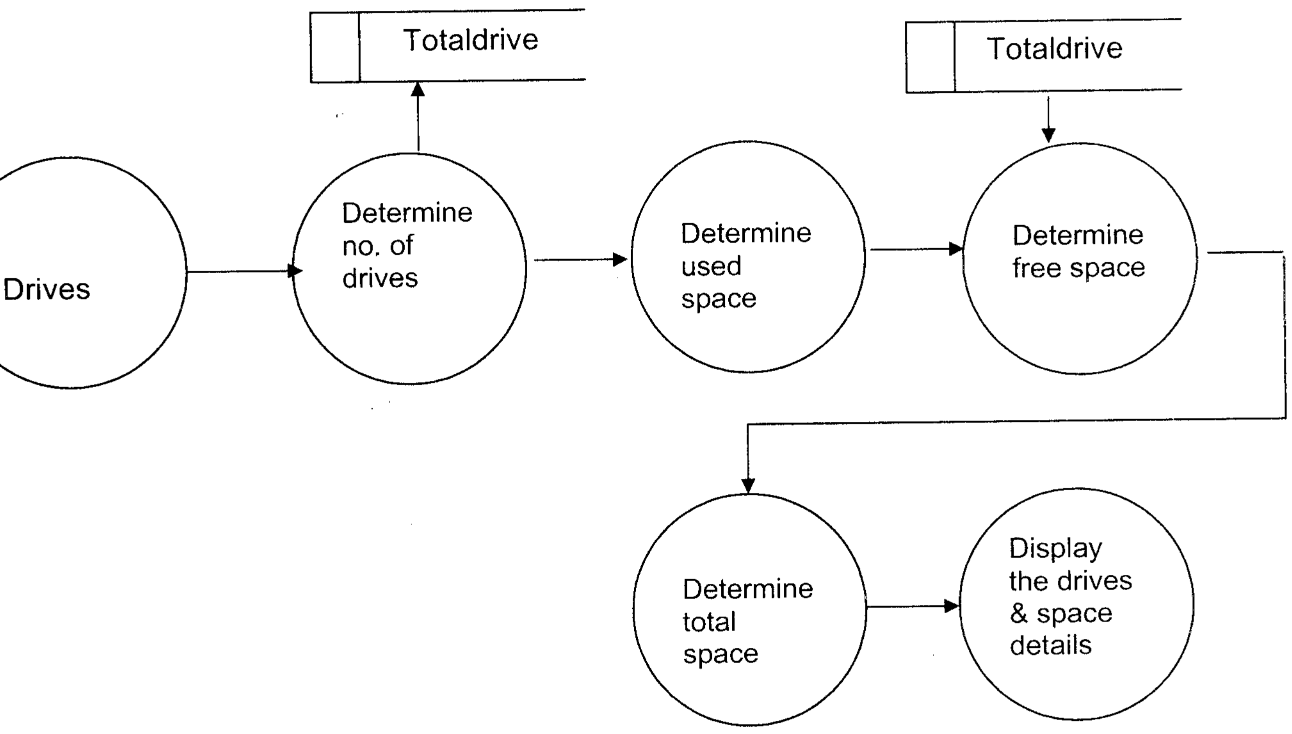


Figure 2.11 Data Flow Diagram Level 3(Drive Details)

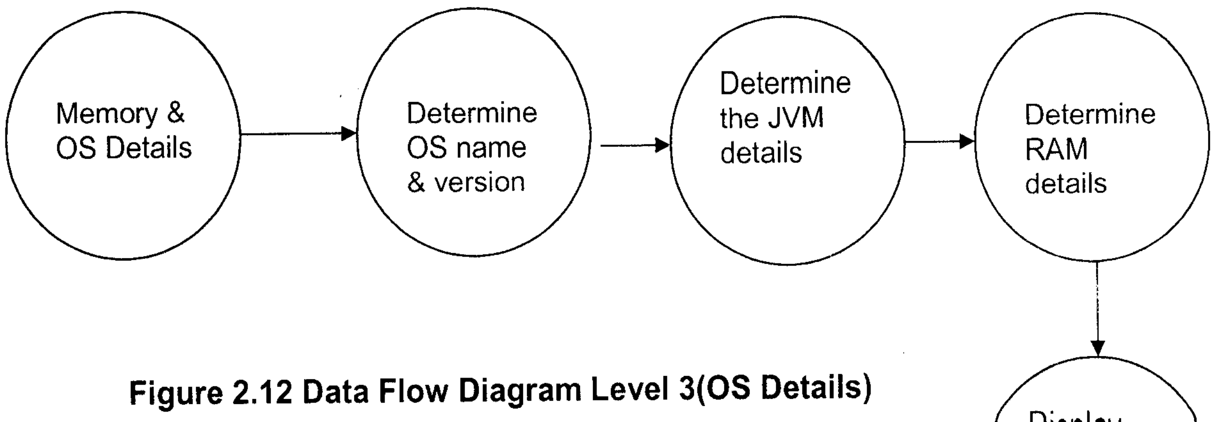


Figure 2.12 Data Flow Diagram Level 3(OS Details)

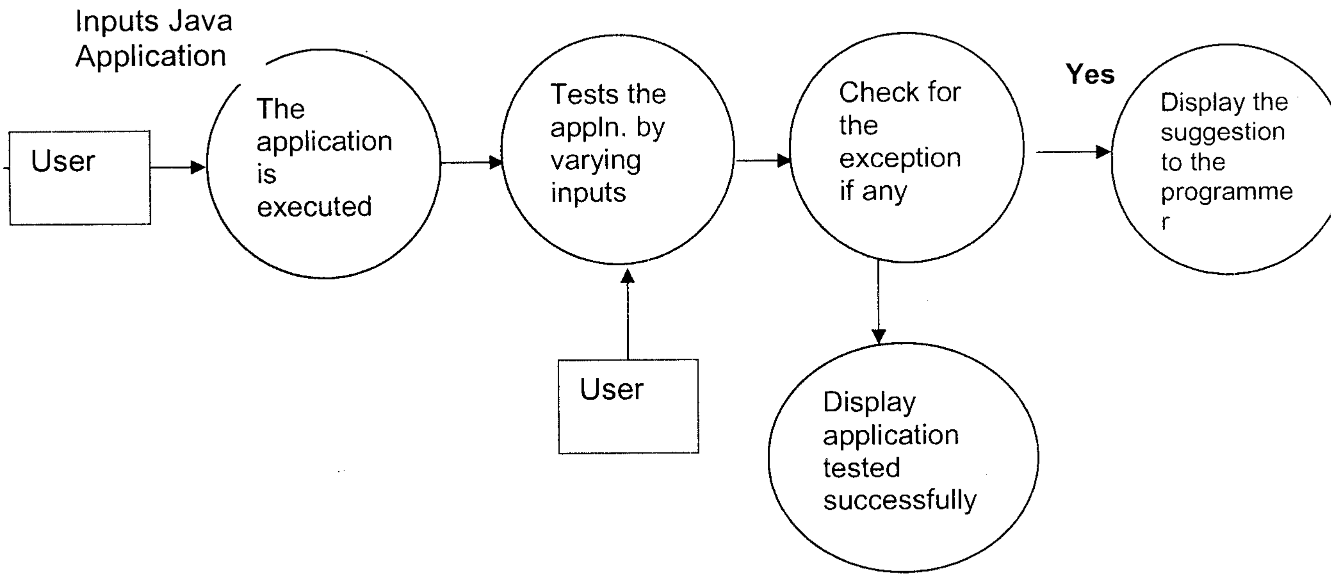
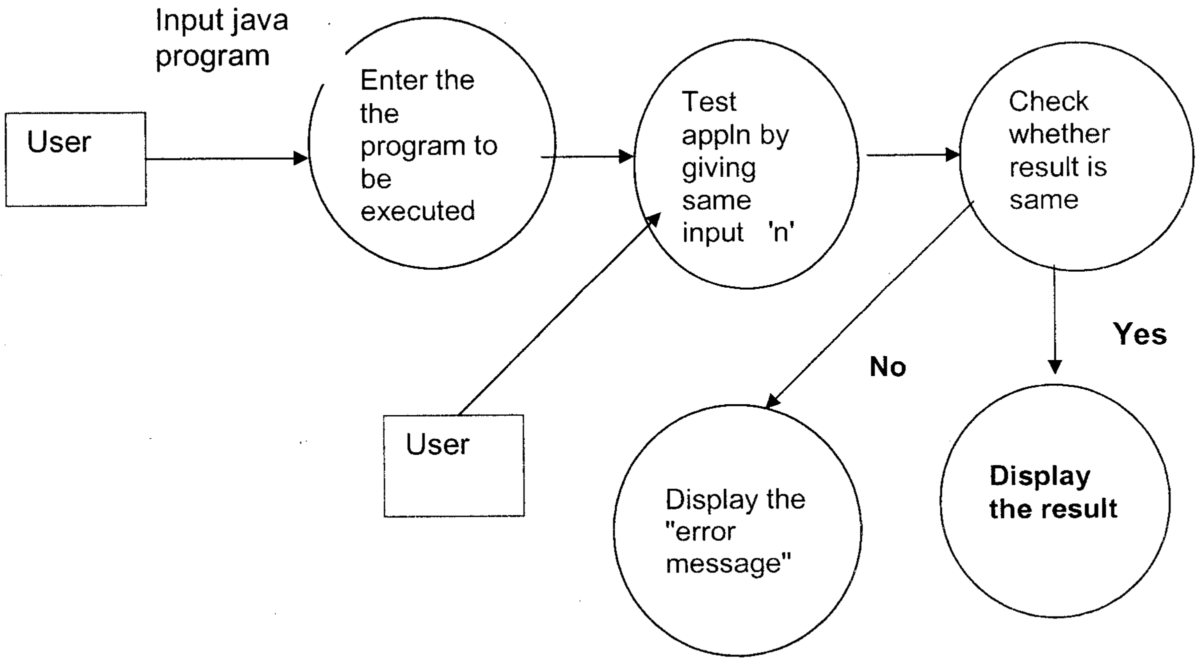
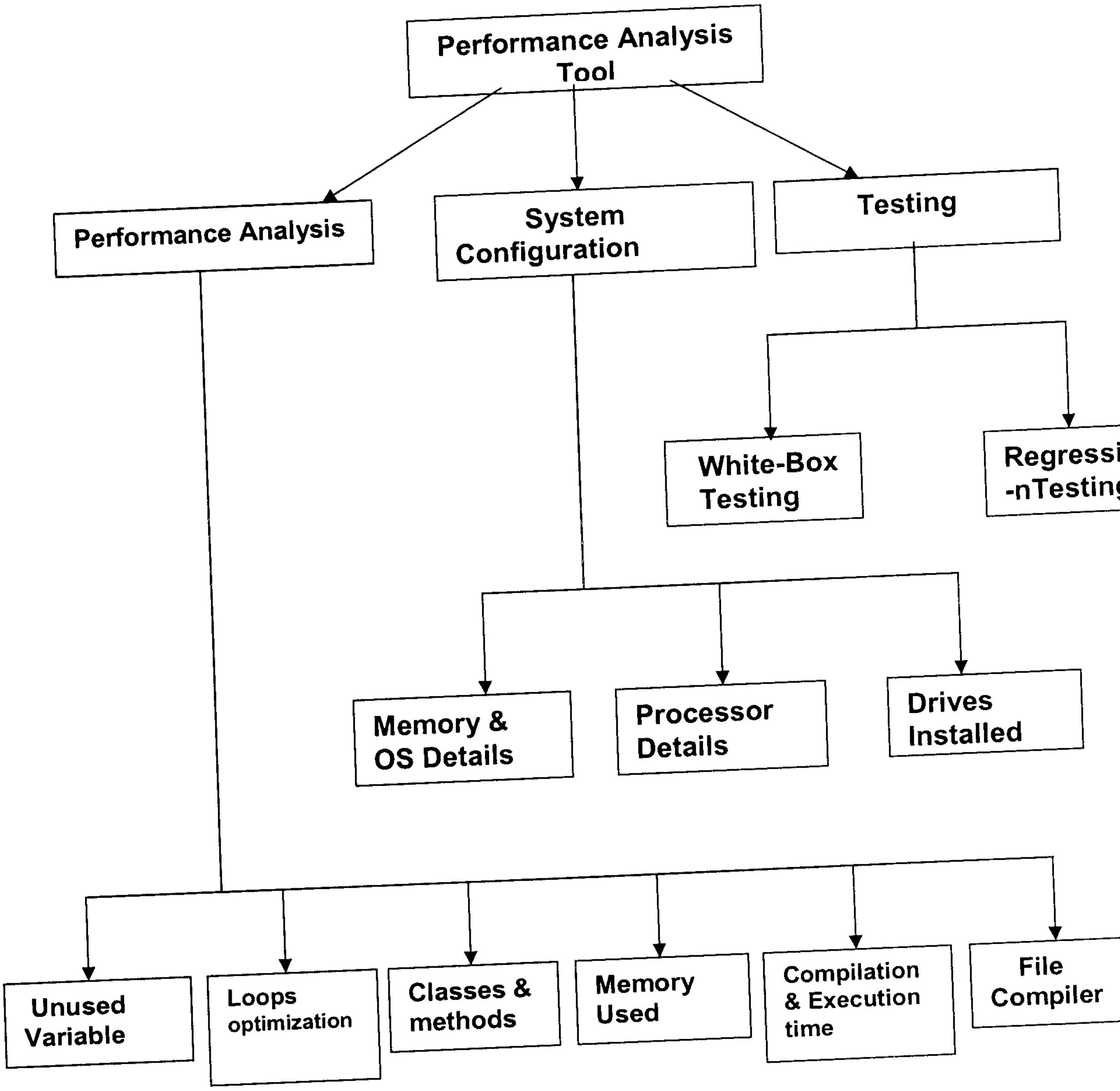

Inputs Java Application
User
The application is executed
Tests the appln. by varying inputs
User
Check for the exception if any
Yes
Display the suggestion to the programme r
Display application tested successfully

Figure 2.13 Data Flow Diagram Level 3(Testing)


Input java program
User
Enter the the program to be executed
Test appln by giving same input 'n'
User
Check whether result is same
No
Yes
Display the "error message"
Display the result

Structure chart:



3.15 Structure Chart (SPAT)

CHAPTER 3

SYSTEM DESIGN

3.1 System Design:

Performance Analysis Tool is a user friendly and fully automaized tool for analyzing performance of Java application, testing the application and to list the system specification. The Tool is designed in the environment JAVA 2.0 and JAVA SWINGS where in it enables the tool to be machine and platform independent.

The various sub systems that were identified in design phase are

- Performance Analysis of the application
- System specification
- Testing of the Application

The efficient coding reduces the memory usage and execution time of the application. The performance analysis module checks for the efficiency of the JAVA application.

The System specification module lists the information about the system where the tool is running. It generates the details of RAM capacity, operating system and Drives in the system.

The Testing module tests the java application with the intent of detecting the errors which may cause complexity.

3.1.1 Performance Analysis of the Application

Performance Analysis Tool is looking at program execution to pinpoint where bottlenecks or other performance problems might occur and locate the code that needs tuning. The performance analysis module checks for the efficiency of the java application. The performance of the application is measured mainly by the two factors

- Memory Usage
- Execution Time

Software performance can be degraded by many factors. They are

- By a particular hardware configuration
- By the wat the hardware is used by the software
- By unexpected interactions between software modules
- Inappropriate use of system resources by the application or middleware software.

- By poor programming or data-structuring technique in the application.

The various sub modules in the performance analysis of the Application are

- validate Used & Unused Code
- Loops to be Optimized
- Classes & Methods
- Time
- Memory

3.1.1.1 Used and Unused Variables

This sub module detects the unused variables in the application. The module determines the used and unused variables in the application based on the following assumptions.

3.1.1.1.1 Used Variables

- A variable is used in the right-hand side of the expression
- a variable passed as a parameter to the function or returns by the function
- Used in the decision making loops or used in conditional loops
- a variable used in methods
- Used in print statements

3.1.1.1.2 Unused Variables

- A variable is declared but not used anywhere in the application
- A variable is declared and initialized but not used anywhere in the application
- A variable is assigned to another variable and is not used further

3.1.1.2 Loop Optimization

This sub module calculates the execution time for each loop. Execution time is calculated for 'While' , 'for' and 'do-while' loops. From the execution time of each loop, the loop to be optimized can be easily determined.

The module identifies the main loops and the inner loop in the application. The execution time is calculated for the main loop. The module displays the main loop, line numbers, inner loops and the execution time.

3.1.1.3 Classes and Methods

In this module the number of classes in the application is determined. By selecting the classes user can view the methods in that class. The tools displays the name of that class.

3.1.1.4 Time

The tool determines the compilation time and execution time of application. By comparing the overall execution time with loop execution time it is possible to identify the loops to be optimized.

3.1.1.5 File Compiler

The tool also provides the additional provision for compiling and executing a application. If there is any error, the user can view the errors and make correction to the application. This File Compiler also has an additional option of viewing the Memory Monitor which displays a graph showing the memory usage of the system.

3.1.2 System Configuration Details

A system configuration detail gives system details on which the tool is running. This is an automated tool and hence reduces the time for the client to gain details regarding the system. This helps in comparing the performance of the application by executing in systems with different configurations.

This module provides detail regarding memory that is memory used and free memory. The operating system installed is also known. The drives in the system are known and the memory usage is listed in this module. System configuration details are generated by the tool.

3.1.2.1 Memory capacity and RAM details

The Memory capacity including the RAM details are listed in the System Configuration Details module. This sub module named Memory provides the user with various details about the memory of the system. This sub module provides the user with details such as Conventional memory, Reserved, Upper, Expanded, Total memory and also the details about the total memory under 1 Mega Byte. All these memory are split under Total, Used and Free memory on all types listed above.

The module also provides details about Total Expanded memory (TEM), Free Expanded memory (FEM), Largest executable program size, Largest free upper memory block. The module shows the Operating System that resides in the high memory area.

3.1.2.2 Operating System and JVM Details

The Operating System Details shows the user about the Operating System used and also the Java Virtual Machine (JVM) details such as OS name, version, architecture and JVM details such as Class path, class home, vendor, version and URL details.

This module also displays the User directory, home and user name.

3.1.2.3 Drive details

The module Drives lists out all the drives used in the system and all details about the drives such as Total space, used space and free space in each drive.

3.1.2.4 Help

The Help option is displayed to the user for his reference about the other sub modules and this page consists of a short note on all the other sub modules showed to the user in that form.

3.1.3 Testing

Testing is a process of executing a program with the intent of finding an error. Testing is the analysis of source/executable code and the controlled execution of executable code to reveal defects that compromise a Java program's executable integrity. Defects often lead to erratic behavior or the premature termination of an executing program. A good test is one that has a high probability of finding an error. The objective is to design tests that systematically uncover different classes of errors and to do with a minimum amount of time and effort. This module performs

Two types of testing is carried out in this module

- White-Box Testing
- Regression Testing

3.1.3.1 White-Box Testing

White box testing is also known as structural testing, clear box testing and glass box testing. It is a test case design method that uses the control structure of the procedural design to derive test cases. White box testing is often first performed at the individual module/method level –Unit Testing. Using white-box testing, the test cases derived should guarantee that all independent paths within a module have been exercised at least once.

This tool performs white box testing automatically. The tool performs the independent path testing. The user has to give the application as input. The tool executes the application and the user tests the application by giving various inputs. If any exception occurs the tool gives the suggestion to the programmer specifying the changes that are to be made in the application.

3.1.3.2 Regression Testing

Regression testing is the re-execution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects. In regression testing the application to be tested is got as input from the user. The user has to enter the number of times the application to executed. The user tests the application n times by giving the same input. The tool checks whether the output is same for n times.

This testing module also consists of a help that provides a short description about the other menus provided in this module.

CHAPTER 4

SYSTEM CONFIGURATION

Source Code Performance Analysis Tool have been developed with the following hardware and software configurations:

4.1 HARDWARE CONFIGURATION:

Development Perspective

Processor	: Pentium 1.4 Giga Hertz
RAM	: 256 MB
Hard Drive	: 20 GB

4.2 SOFTWARE CONFIGRATION:

Tools	: Java, Swing
IDE	: JCreator LE
OS	: Windows ME

CHAPTER 5

TESTING

The testing process focuses on the logical internals of the software assuring that all the statements have been tested and also on the functional externals by conducting tests to uncover errors. This process also ensures that defined input will produce actual results that agree with required results. Testing is an important part of the Development Life Cycle. The amount of testing required is related to the size and complexity of the application. Following are the tests that were carried out to test the working of SPAT:

- Unit Testing
- Integration Testing
- System Testing

5.1 UNIT TESTING:

Unit testing is performed on a single stand-alone module or unit of code. Unit testing is the phase of testing that tests the basic functionality and structure of the code. The module interface is tested to ensure that information properly flows into and out of the program unit under test. Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing. All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once. And finally, all error-handling paths are tested.

5.2 INTEGARTION TESTING:

Integration testing is a systematic technique for constructing the program structure, while at the same time conducting tests to uncover errors associated with interfacing. That is, the program is constructed and tested in small segments, which makes it easier to isolate and correct. This testing focuses on the design and construction of the software architecture. Integration testing is black box oriented. The objective is to take unit-tested modules and build a program structure that has been dictated by the design. Both integration and testing takes place in this phase. Units are combined one at a time and tested, till entire software gets integrated. Top-Down integration approach was followed for testing SPAT since functionality could be tested much early.

5.3 SYSTEM TESTING:

System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system. At this stage the production environment is ready. The purpose of system testing is to validate the system against the functional specification by executing the test cases prepared during the analysis phase. Stress tests and Security tests, which are types of system tests, were carried out successfully for SPAT.

5.3.1 Stress Testing:

Stress testing is designed to confront programs with abnormal situations. Stressing the system may often cause functional failures. SPAT was put into stress testing to check the availability of resources at abnormal conditions. This helps in the fine-tuning of the system.

CHAPTER 6

CONCLUSION

The "Source Code Performance Analysis Tool" is designed efficiently. The Tool is used to analyze the performance of the Java application. The tool generates the system details where it is made to run. The application is also tested automatically for errors .

In performance analysis the tool efficiently determines the used and unused variables, execution time of the application, execution time of each loops and the memory utilized by the application, from which the performance of the application is determined.

In System configuration the tool is involved in finding out the system resource, memory space , drive capacity which is used to compare the performance with respect to configuration and reports are generated. The tools carry out testing process like white-box testing and Regression testing to point out errors automatically which affects application's successful completion.

All these modules are efficiently developed and executed using different java applications and the tool is also tested with different set of users and implanted successfully.(next) this system has been designed to automate the tasks that are routinely done as part of the life cycle. It is an easy to use tool that saves time, often executing the task in seconds that take min or hrs if done manually and there by supports the analyzer and tester in their process. The highlight of this project is that all the operations are handled by user friendly and extensive graphical support.

This automated tool is very flexible and also extensible. The sys can be enhanced to provide solution to the identified flaws in the application. In addition to the available capabilities the tool would also support other testing techniques. The sys can be enhanced in the future to validate the code, correct the errors and to draw a model for the validated source.

6.1 FURTHER ENHANCEMENT

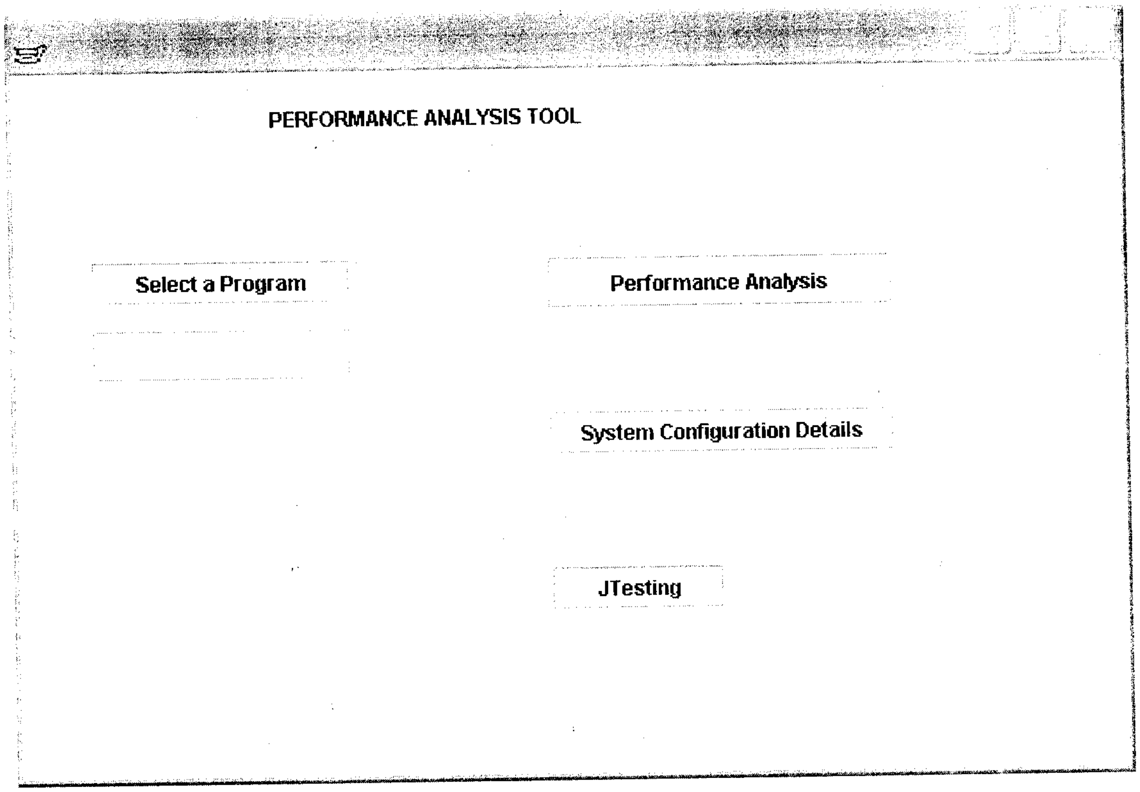
Performance Analysis Tool can be extended to check for the performance and test C, C++. It can be used to check the RMI. (Remote Method Invocation).

Now the tool analyzes and tests the java application using source code, in future the java applications are analyzed and tested using byte code.

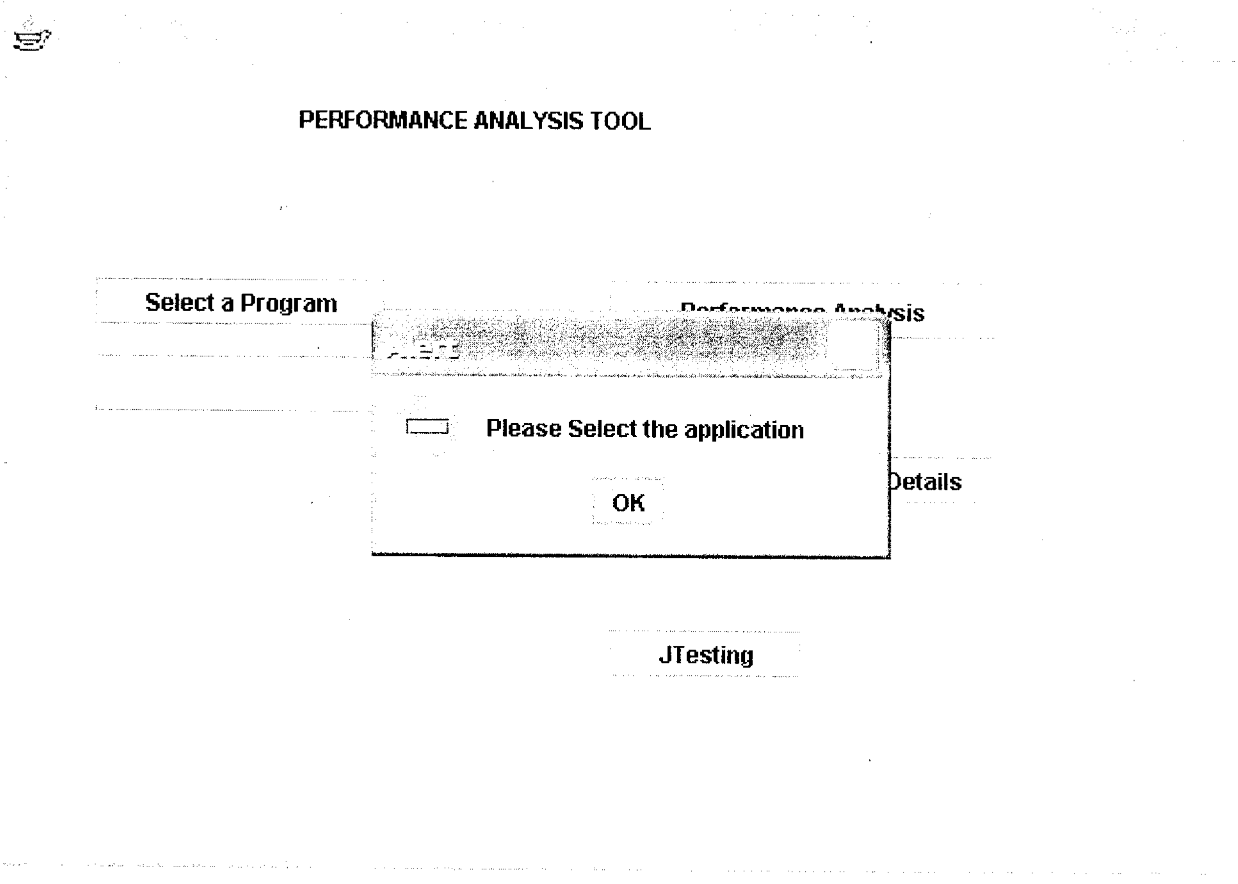
All testing features are to be carried out as a future enhancement. It may be developed to check for the performance on the byte capacity of the system. Graphics can be used to display the performance of the application.

APPENDIX 1

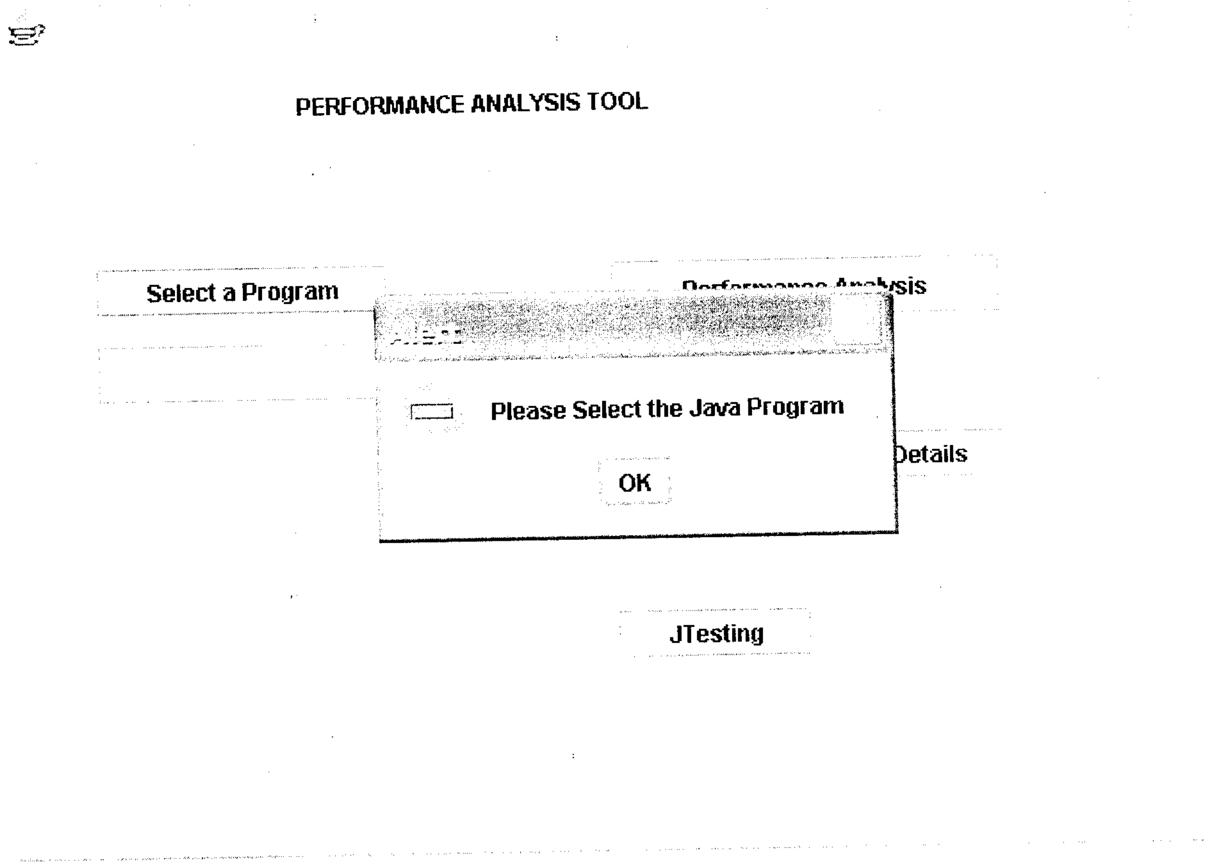
SCREEN SHOTS



Performance Analysis Tool Main Screen

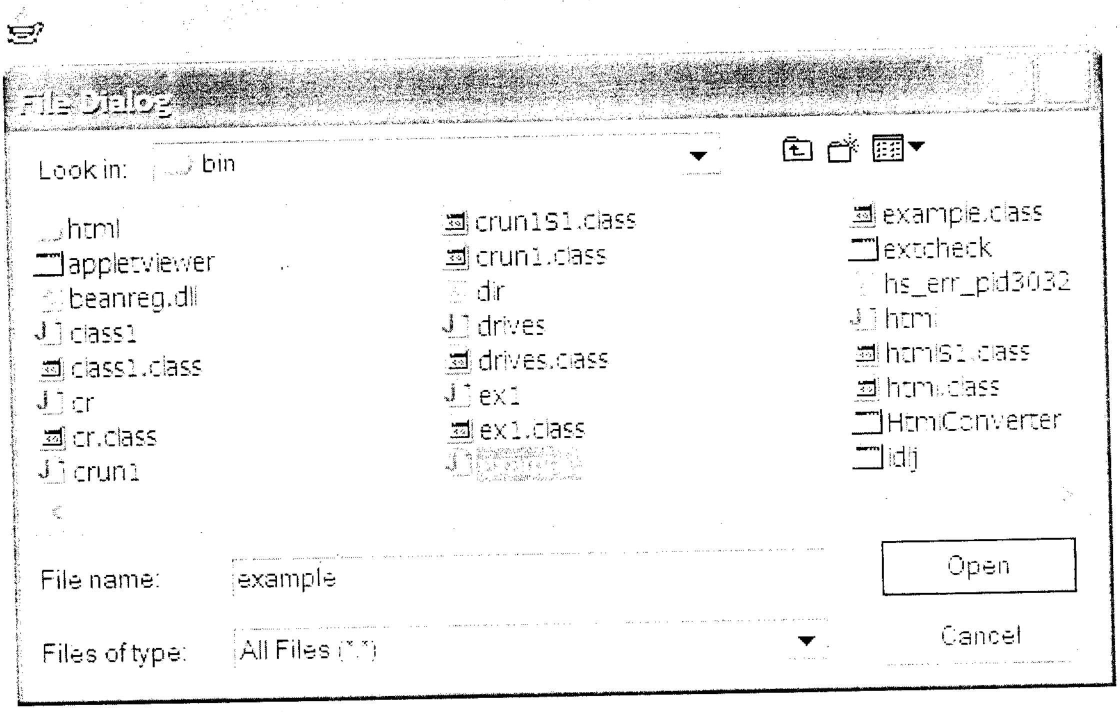


Select A JAVA Application

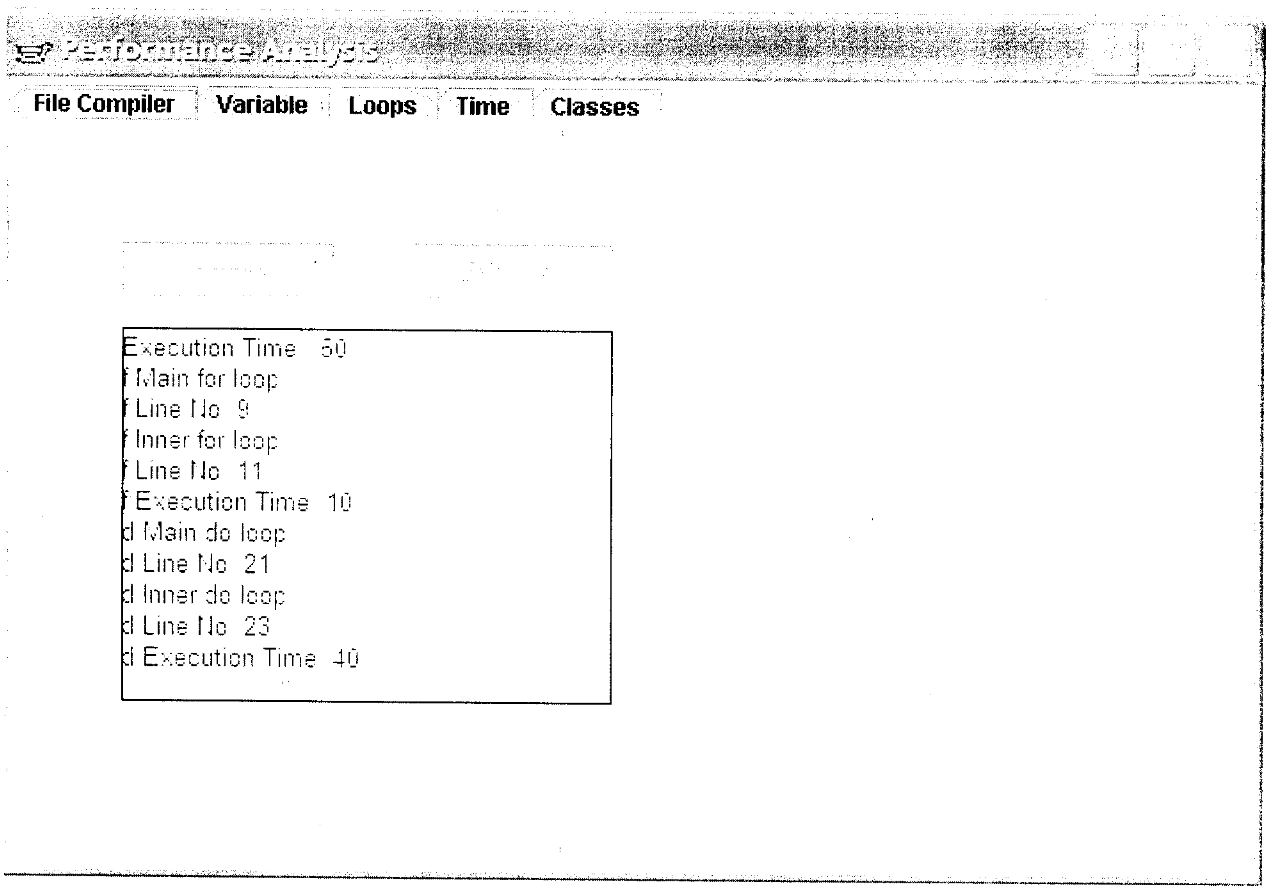


Please Select the Java Program

Select A JAVA Application

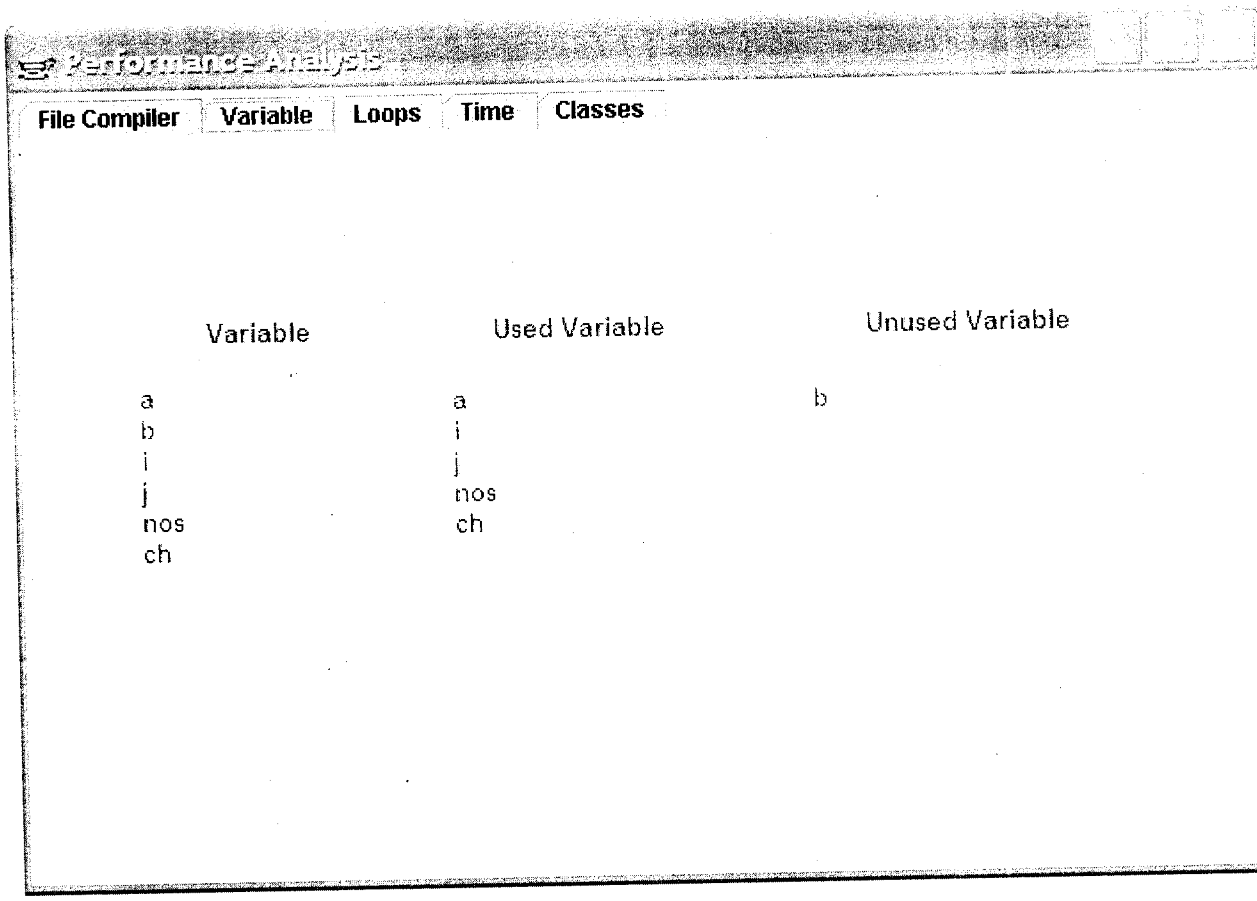


Choose a JAVA File



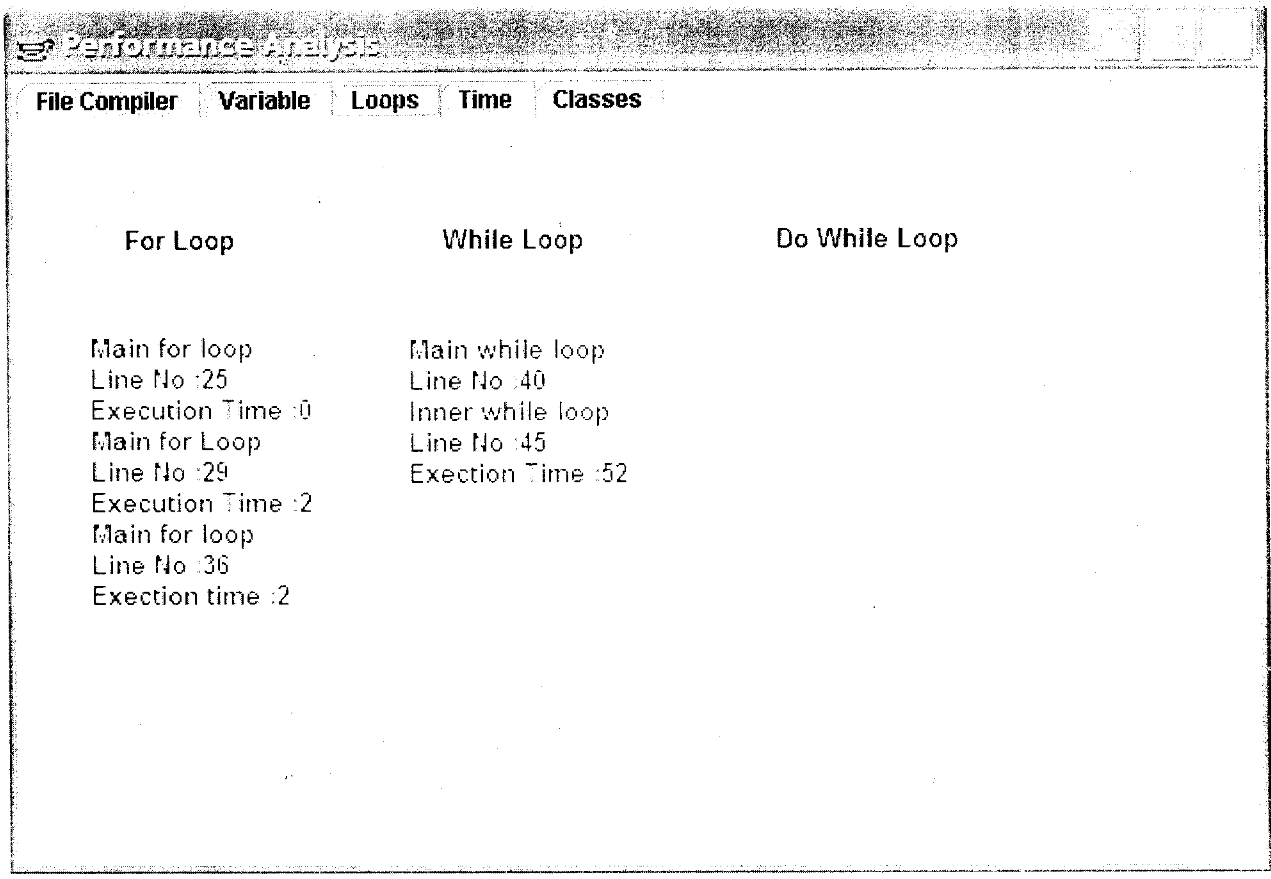
```
Execution Time  50
f Main for loop
f Line No  9
f Inner for loop
f Line No  11
f Execution Time  10
d Main do loop
d Line No  21
d Inner do loop
d Line No  23
d Execution Time  40
```

File Compiler



Variable	Used Variable	Unused Variable
a	a	b
b	i	
i	j	
j	nos	
nos	ch	
ch		

Used & Unused Variables



For Loop

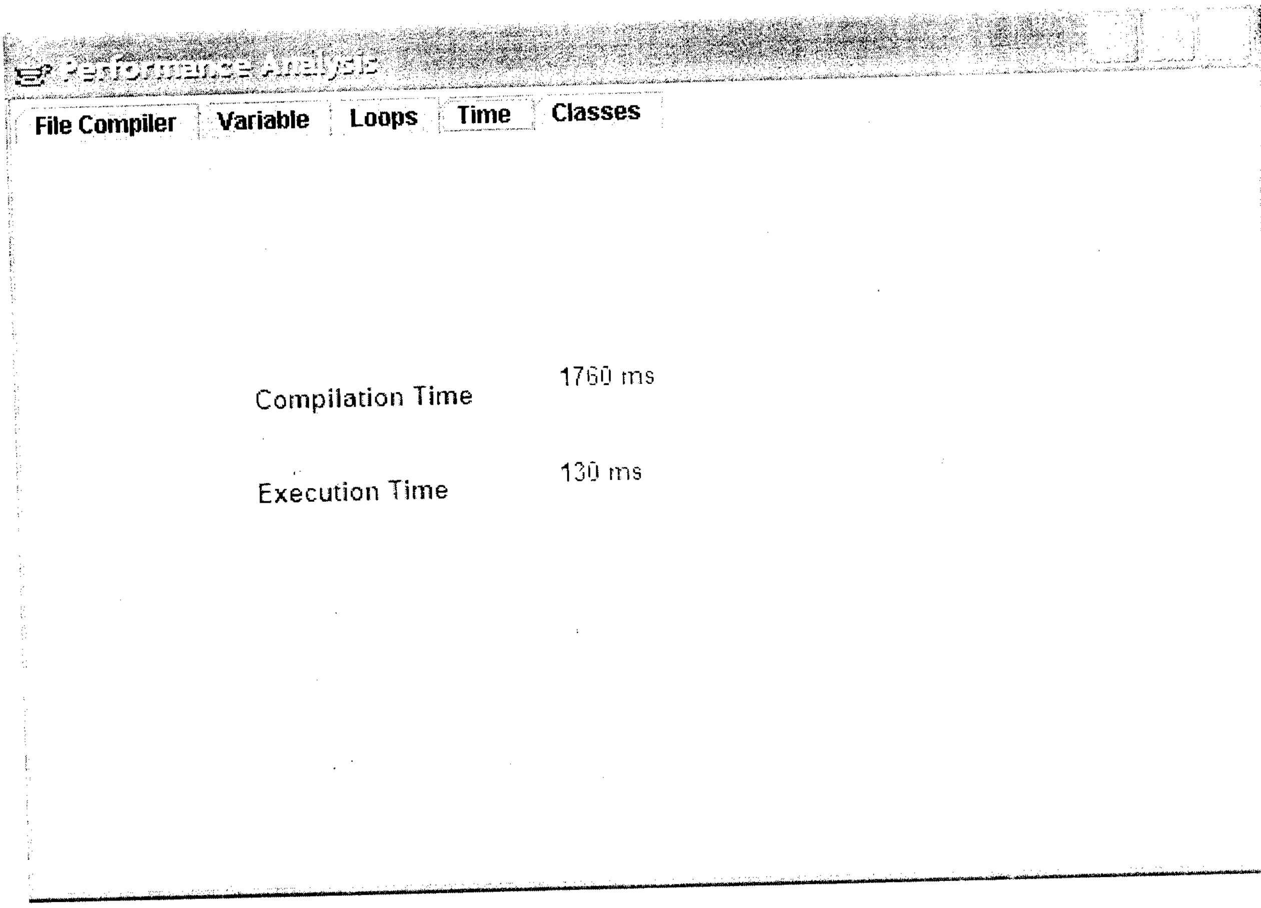
Main for loop
Line No :25
Execution Time :0
Main for Loop
Line No :29
Execution Time :2
Main for loop
Line No :36
Exection time :2

While Loop

Main while loop
Line No :40
Inner while loop
Line No :45
Exection Time :52

Do While Loop

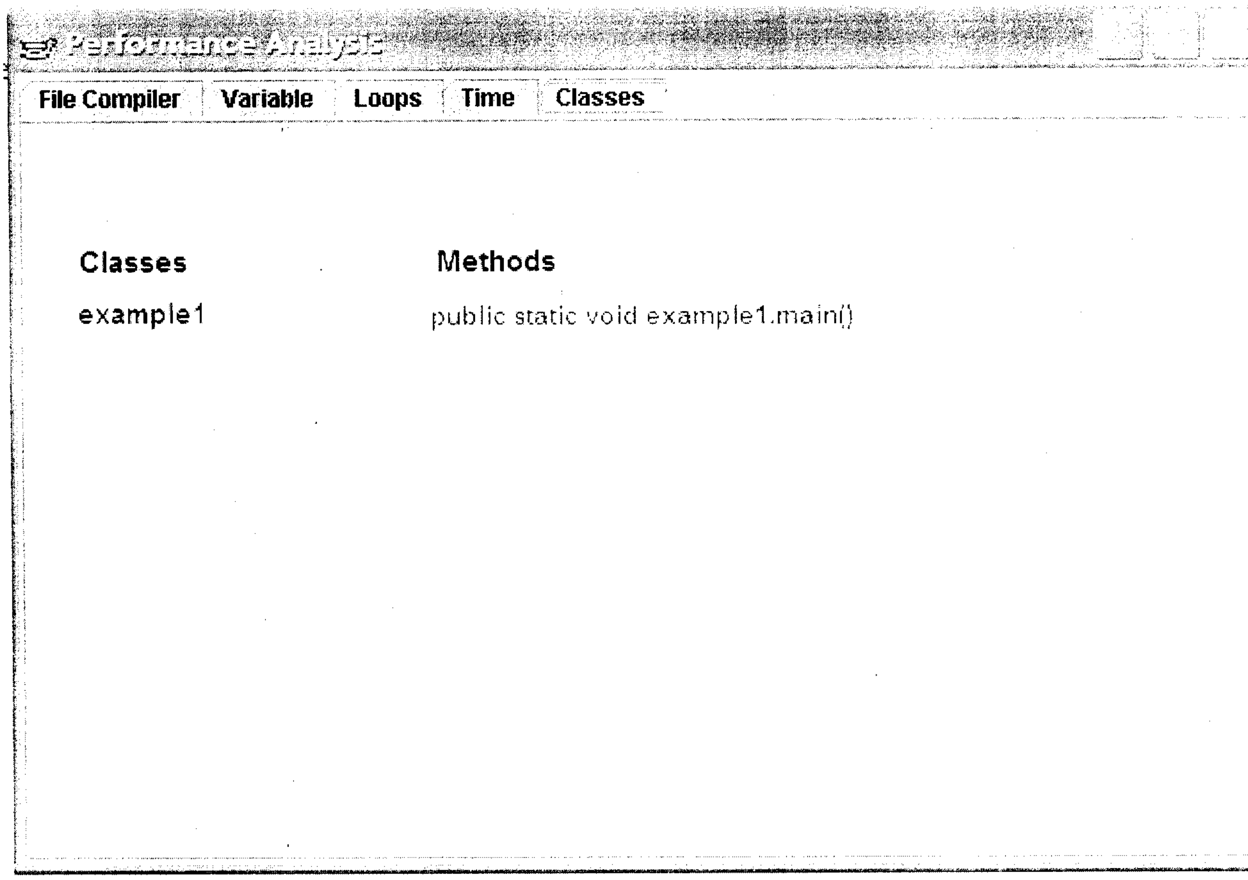
Loops to be optimized



Compilation Time 1760 ms

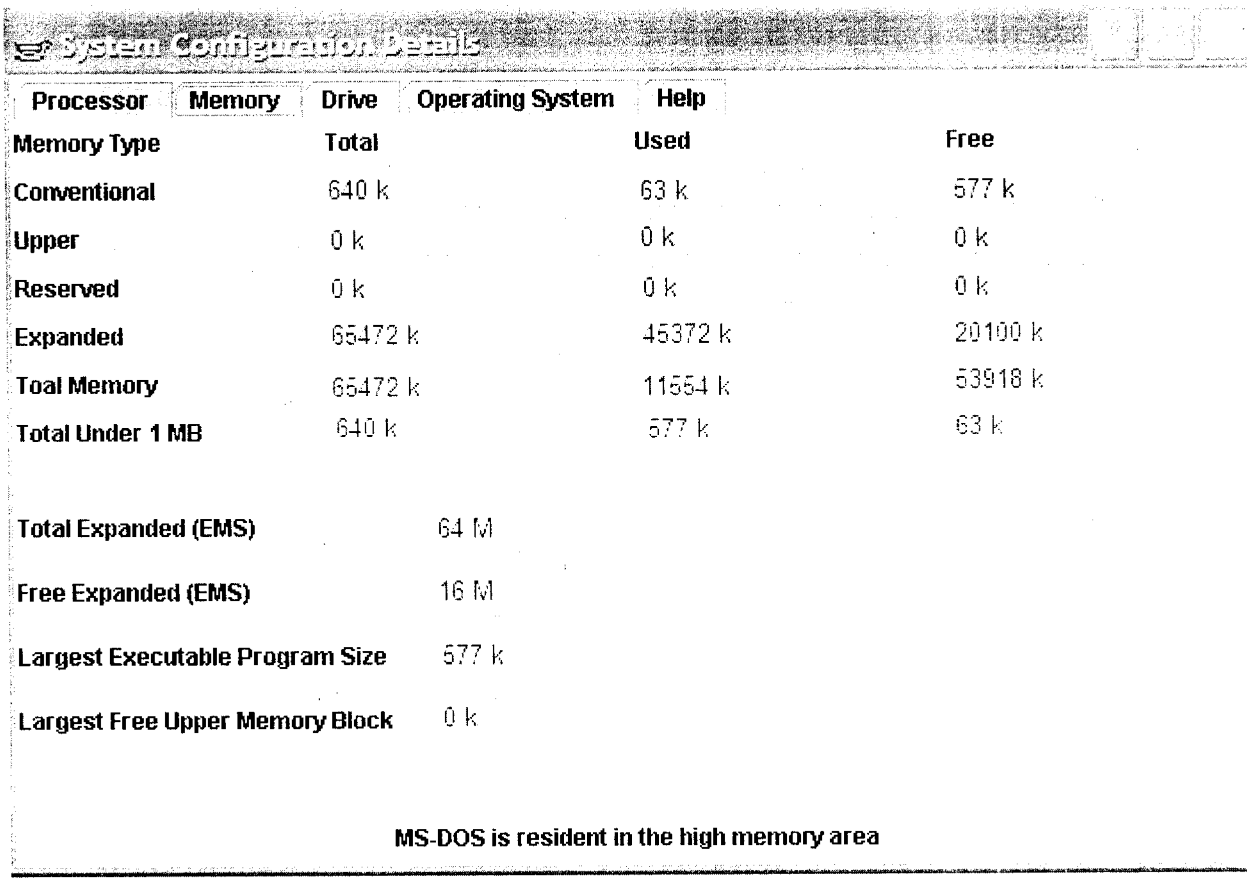
Execution Time 130 ms

Compilation & Execution Time



Classes | Methods
example1 | public static void example1.main()

Classes & its Methods

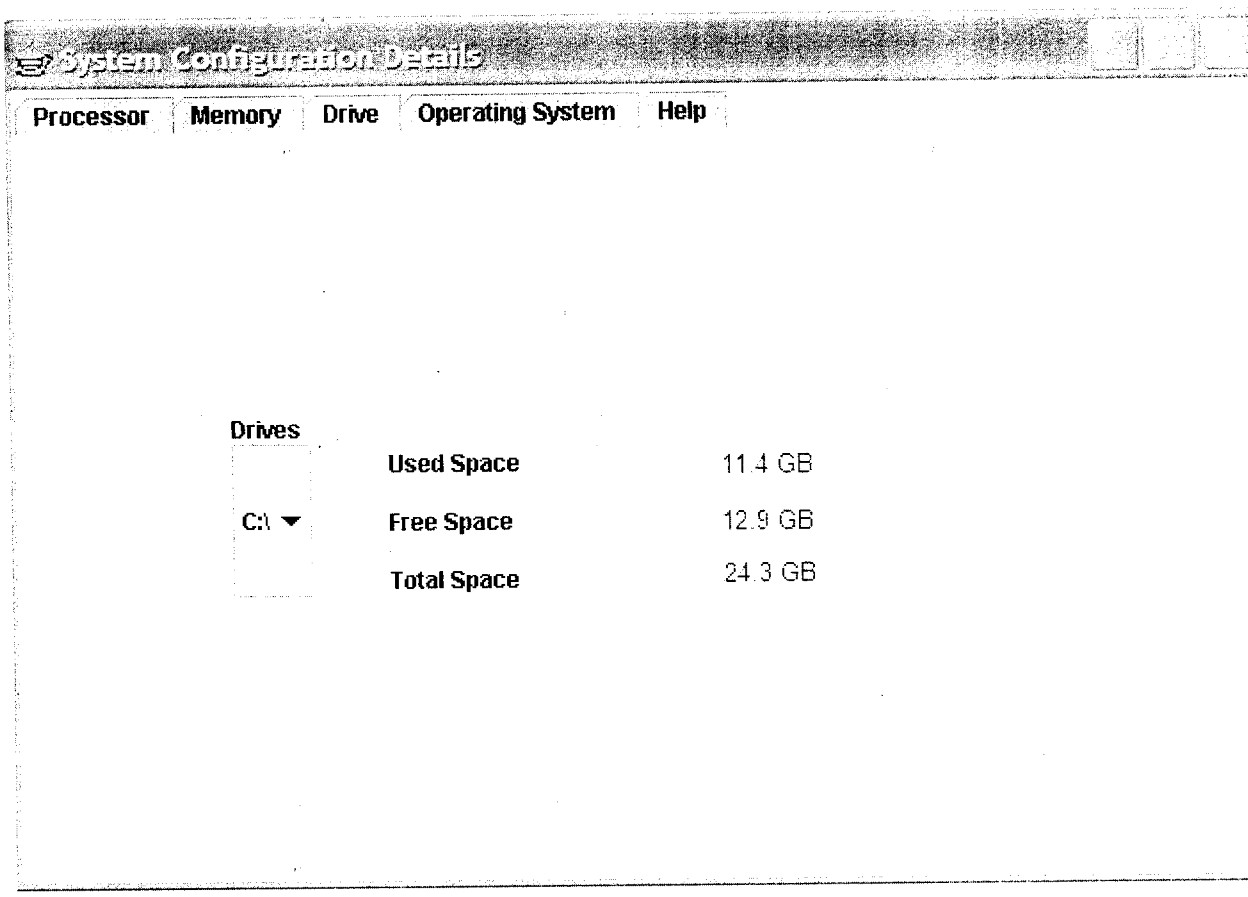


Memory Type	Total	Used	Free
Conventional	640 k	63 k	577 k
Upper	0 k	0 k	0 k
Reserved	0 k	0 k	0 k
Expanded	65472 k	45372 k	20100 k
Toal Memory	65472 k	11554 k	53918 k
Total Under 1 MB	640 k	577 k	63 k

Total Expanded (EMS)	64 M
Free Expanded (EMS)	16 M
Largest Executable Program Size	577 k
Largest Free Upper Memory Block	0 k

MS-DOS is resident in the high memory area

System Configuration details – Memory

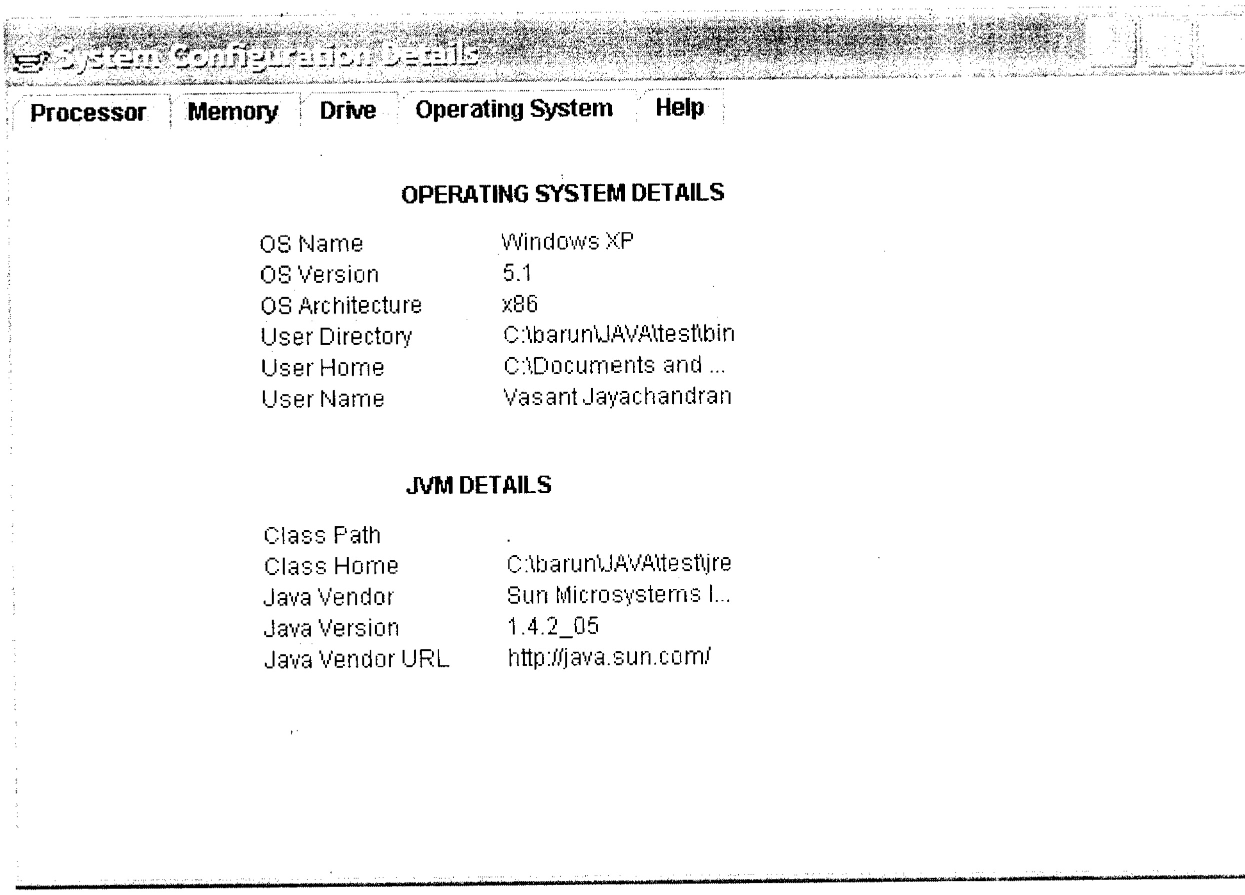


Drives

C:\

Used Space	11.4 GB
Free Space	12.9 GB
Total Space	24.3 GB

Drive Details



OPERATING SYSTEM DETAILS

OS Name	Windows XP
OS Version	5.1
OS Architecture	x86
User Directory	C:\barun\JAVA\test\bin
User Home	C:\Documents and ...
User Name	Vasant Jayachandran

JVM DETAILS

Class Path	.
Class Home	C:\barun\JAVA\test\jre
Java Vendor	Sun Microsystems I...
Java Version	1.4.2_05
Java Vendor URL	http://java.sun.com/

Operating System Details

REFERENCES

1. Robert Eckstein, Marc Loy And Dave Wood , "JAVA Swing" Shroff Publishers & Distributers Pvt. Ltd.,2000

2. http://www.jguru.com

3. http://java.sun.com/developer/Books/javaserverpages/

4. Roger S Pressman, "Software Engineering A Practitioner Approach", Mc Graw Hill,2001