PERFORMANCE ANALYSIS TOOL

Project Report   p-1203

Submitted in partial fulfillment of the
Requirements for the award of the degree of the

Bachelor of Computer Science and Engineering
Of
Bharathiar University, Coimbatore.

Submitted by

S.V.Madhana
0027K0178

D.Madhubhashini
0027K0179

Under the guidance of

Ms.S.Rajini
Senior Lecturer

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE – 641006.

MARCH 2004.

CERTIFICATE

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University)

COIMBATORE – 641 006, TAMIL NADU, INDIA

Approved by AICTE, New Delhi - Accredited by NBA

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project entitled

PERFORMANCE ANALYSIS TOOL

has been submitted by

D.Madhubhashini and S.V.Madhana

in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering of the Bharathiar University, Coimbatore – 641 046 during the academic year 2003-2004

[illegible]

Head of the Department
(Dr.S.Thangasamy Ph.D.,)

[illegible]

Project Guide
(Ms.S.Rajini B.E., M.S.,)

Certified that the candidates were examined by us in the project

Viva Voce examination held on 23.3.2004

[illegible] 23/3/'04

Internal Examiner

[illegible] 23/3

External Examiner

DECLARATION

DECLARATION

We,

S.V.Madhana	**0027K0178**
D.Madhubhashini	**0027K0179**

declare that the project entitled "Performance Analysis Tool" is done by us and to the best of our knowledge, a similar work has not been submitted earlier to the Bharathiar University or any other institution, for fulfillment of the requirement of the course study.

This project report is submitted on the partial fulfillment of the requirement for all awards of the degree of Bachelor Of Computer Science and Engineering of Bharathiar University.

[illegible]

D.Madhubhashini

Place: Coimbatore.

Date: 23.3.2004

S.V. Madhana

S.V.Madhana

Counter Signed by the guide: Rajini

(Ms.S.Rajini B.E., M.S.,)

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

The exhilaration achieved upon the successful completion of any task should be definitely shared with the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken shape.

At the onset, we take this opportunity to thank the management of our college for having provided us excellent facilities to work with. We express our deep gratitude to our Principal Dr.K.K.Padmanabhan for ushering us in the path of triumph.

We are always thankful to our beloved Professor and the Head of the Department, Dr.S.Thangasamy whose consistent support and enthusiastic involvement helped us a great deal.

We are greatly indebted to our beloved guide Ms.S.Rajini B.E, Senior Lecturer and project coordinator Mrs.D.Chandrakala M.E, Senior Lecturer, Department of Computer Science and Engineering for her excellent guidance and timely support during the course of this project. As a token of our esteem and gratitude, we honor her for her assistance towards this cause.

We also thank our beloved class advisor Mrs.M.S.Hema B.E., for her invaluable assistance.

We also feel elated in manifesting our deep sense of gratitude to all the staff and lab technicians in the Department of Computer Science and Engineering.

We feel proud to pay our respectful thanks to our Parents for their enthusiasm and encouragement and also we thank our friends who have associated themselves to bring out this project successfully.

SYNOPSIS

SYNOPSIS

This project entitled "***Performance Analysis Tool***" is to develop a toolkit that analyzes the performance of any JAVA application. It is developed using JAVA and Swing. This helps the programmers to optimize their source code there by enhancing the performance. This involves three main modules.

Generally, JAVA uses byte code and hence its performance is slow. In order to avoid further delay and enhance the performance rate, this toolkit can be used. Any JAVA application (JAVA core) is taken as input.

Performance analysis is the first module. This constitutes many sub-modules. The unused variables in the source code, which delay the compilation and execution time, are identified and brought to our notice. All the loops are scanned and their time of execution and compilation are also got. This also gives the overall time taken. Memory consumption during execution and compilation is also found.

Besides, this facilitates the user to access the system details. It includes the memory status of all the drives present, RAM size, version number of the current operating system and user details such as user directory and so on. This will be the second module.

The testing module performs testing on the Java application with the intent of detecting the simple error, which may cause complexity. Two tests, which are carried out using this tool, are white box and regression testing.

CONTENTS

CONTENTS

INTRODUCTION

INTRODUCTION

1.1) Existing system:

In the existing system, the Java compiler after compilation displays only the presence of errors. It does not give the presence of any unused variable. It does not output the time consumption of the application with respect to compilation and execution. One cannot view the time consumption of any loops present in a Java application.

1.2) Proposed System:

The aim of the proposed system is to give a proper analysis of the performance of any Java core application. The main functions include,

- Compilation with the display of the used and unused variables.
- Identification of the loops and the time taken for their compilation and execution.
- Display of the methods in all the classes in the application.
- It also helps the user get the memory details, OS and JVM details.

SOFTWARE REQUIREMENTS ANALYSIS

SOFTWARE REQUIREMENTS ANALYSIS

2.1) Product Definition:

Performance analysis tool is very simple. The user has to just run this tool and he gets to compile and execute any Java application and on error, the text editor provided will allow you to easily modify the application to suit your own needs. This tool is best suited to get the various system details required.

2.2) Project Plan:

The project entitled ' Performance Analysis Tool' is to develop a Java toolkit to analyze the performance of any Java application. A Java writing pad, which will act as a text editor, will be provided to make any modification required. Another important feature added is System Testing. This is made possible by two main testing methods:

- White box testing
- Regression testing

White box testing:

The tool performs the white-box testing automatically. The user has to give the Java 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 suggestions to the programmer specifying the changes that are to be made in the application.

Regression testing:

Using this the user can execute the Java application that he has chosen already any number of times as desired and makes sure he gets the same final output all the time. This will identify and confirm the consistency of the particular application being tested.

Both these testing methods will ensure the programmer that his application is in good working condition.

SOFTWARE REQUIREMENTS SPECIFICATIONS

SOFTWARE REQUIREMENTS SPECIFICATION

1. Introduction:

Performance Analysis Tool 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 APIs.

SRS 1:

Purpose of the document:

The purpose of this document is to provide the requirements for the Performance Analysis Tool. They are primarily intended for customers of the application, but will also be of interest for Java programmers.

SRS 2:

Scope:

The document is limited to the description of Performance Analysis Tool, a code analyzer used for analyzing the performance of any Java application and generate report on them.

Overall description:

The main function associated with this product is described in this section. The characteristics of a user of this product are indicated.

SRS 2.1

Product perspective:

This product was intended to fulfill the needs of users who do Java programming and for those who have a great interest in analyzing the performance and improving the Java program. The design and documentation for this will make it convenient to expand and modify the tools. It is interactive with users.

SRS 2.2

Product function:

The product is in the form of a text editor that can open Java applications, make required modifications and then save them. Then one can access the performance analysis part by just going into the "Tool" menu provided. Also, he can get the system details along the OS and JVM details.

SRS 2.3

User characteristics:

Users of this product are mostly Java programmers, who are able to understand every aspect. The user must know where exactly he should go about the optimization of the loops used and the whole program.

SRS 2.4:

Constraints:

- The product operates on Personal Computer's running Windows and is not tested under other platforms.

SRS 3:

Specific requirements:

In this section we will see about the specific requirements like interface, functional requirements and performance requirement of the product.

SRS 3.1:
Interface requirements:

SRS 3.1.1:
User interface:

The user interface of this product is a text area for source code modification of the Java application being worked upon.

SRS 3.2:
Functional requirements:

This will define the fundamental actions, which take place in the product.

SRS 3.2.1

Information flow:

The information flow is in the form of graphics and text .The user will first run the Java text editor pad or the Java Swing pad and then gets the access of the performance analysis section through the options provided.

SRS 3.2.2

Process description:

In order to use this analyzer product, the user first runs the Java Swing pad and then gets on to the code analyzer option provided as one of the menu items. There after he can compile or execute any Java application and find out various flaws that can take place. If he wants to know the system details, he can switch over to the system details option and get the clear data of the details that corresponds to OS configuration, JVM details and so on.

SRS 3.2.3

Data construct specification:

Data construct are prepared for each data referenced in the dataflow diagram and process description. The IEEE STD 830-1993,a data construct describes a data item name and its type. If the type of a data item is a record its constituent fields are given. If the data construct describes a state, it is expressed in terms of its partition into the parts of the system state.

SRS 3.3

Performance requirements:

The static as well as dynamic numerical requirements placed on the software or on human interaction with the product are described in this section. Examples of the performance requirements are:

Dynamic: All click and drop operation is accomplished within some milliseconds.

Static: Each controller involved should be static.

SYSTEM DESIGN

SYSTEM DESIGN

1. Introduction:

This document will specify the logical structure of the program. This will show the detailed description of the product.

1.1 Purpose:

The purpose of the software design document (SDD) is to provide an overall description of "Performance Analysis Tool" and to specify the system's detailed design.

1.2 Scope:

The scope of the software design document is constrained by the requirements stated in the "Performance Analysis Tool" software requirement specification.

1.3 Definitions, Acronyms and Abbreviations:

Java Swing Pad: It is an area for opening Java source code and for performing some modifications to it based on the analysis made by the Java tool or to modify otherwise also.

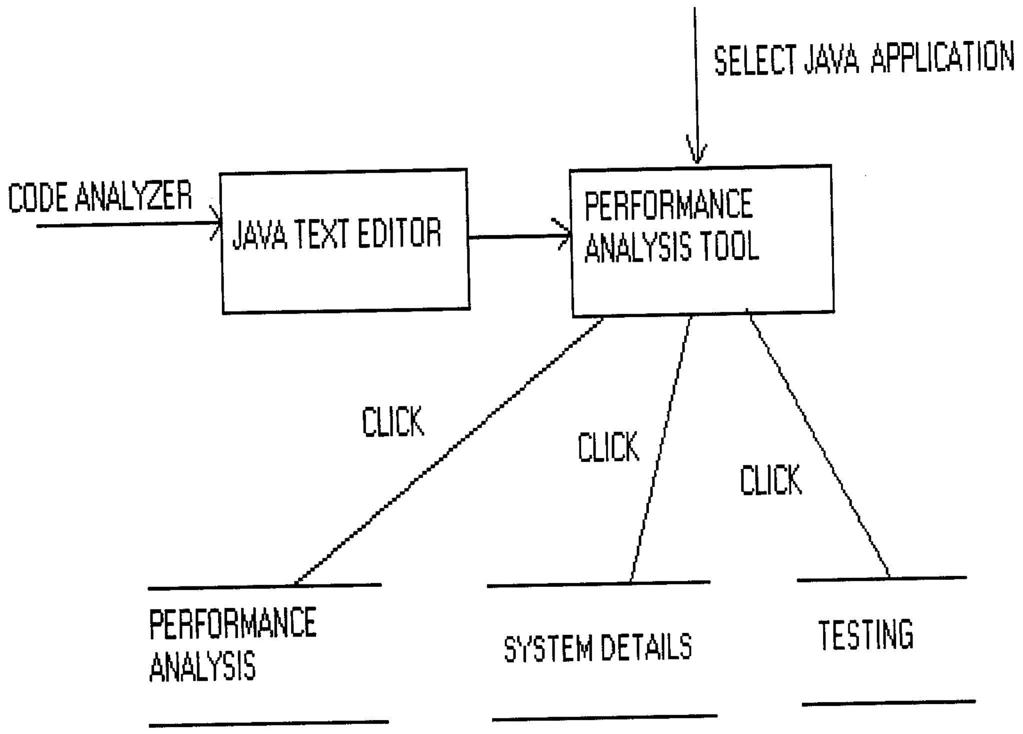
1.4 References

Software Engineering -an engineering approach
By- JAMES F.PETERS & PEDRYCZ

IEEE Design document 1016-1998
www.dur.ac.uk/`dcs8s08/design/design.html

2. **Product Design**:

HIGH LEVEL DATA FLOW DIAGRAM:



SYSTEM TESTING

SYSTEM TESTING

1. Scope and reference:

1.1 Inside the scope:

The scope of this document is to give a test case for testing the product's functionality & performance. The test case should be designed in such a way that it has a high probability of finding as-yet undiscovered error.

1.2 Reference:

IEEE standard for software unit testing – std.1008-1987.

Software Engineering a Practitioner's Approach – Roger S. Pressman

http://cs.anu.sdu.au/student/comp3100/delivs/type2/pr-ut.html

2. Unit testing activities:

3.1 Plan the general approach, resource and schedule.

3.2 Determine features to be tested.

3.3 Refine the general plan

3.4 Design the set of test

3.5 Implement the refined plan and design.

3.6 Execute the test procedure

3.7 Check for termination

3.8 Evaluate the test efforts and unit

3. Product Testing:

Software resting is a process of executing a program with the intent of finding an error. Software testing is a critical element of software quality assurance and represents the ultimate review of system specification, design coding. Testing is the last chance to uncover the errors and defects in the software and facilitate the delivery of quality system.

Testing Principles

The basic principles for effective software testing are follows:

- A good test case is one that has a high probability of finding.
- A successful test is one that Uncovers an as-yet undiscovered error.
- All tests should be traceable to the customer requirements.
- Tests should be planned long before testing begins.
- Testing should begin in the small" and progress towards testing " in the large"
- Exhaustive testing is not possible.

System Testing Requirements

Software testing is not an activity to take up when the product is ready. An effective testing with a proper plan begins from the user requirements stage Itself. Software testability is the case with which a computer program is tested. Metrics can be used to measure the testability of a product.

Software Testing Phases

Several testing strategies lead to the following generic Characteristic:

- Testing begins at the unit level and works "outward" towards the integration of the entire system.
- Different testing techniques are appropriate at different points of software development cycle.

White Box Testing

White box testing is used to test control structures of a procedural Design. We can derive test cases to ensure

- All independent paths are exercised at least once.
- All logical decisions are exercised for both true and false paths.
- All loops are executed at their boundaries and within operational Bounds.
- All internal data structures are exercised to ensure validity.

Black Box Testing

Black Box Testing is testing without knowledge of the internal workings being tested. Black Box Testing is also known as functional testing.
It is a software testing technique where by internal workings of the its being tested are not known by the tester. Black box testing implies that the selection of test data as well as interpretation of test results are performed on the basis of the functional properties of a piece of the software. Black Box testing should not be performed by the author of the program who knows too much

about the program internals. For example, when black box testing is applied to Software engineering, the tester would only know the “legal” inputs and what the excepted outputs should be, but not how the program actually arrives at those outputs. It is because of this that black box testing can be considered testing with respect to the specifications, with no other knowledge and independent of one another, avoiding programmer bias toward his own work. For the tests planning can begin as soon as the specifications are written.

Advantages of this type of testing include

- The test is unbiased because the designer and the tester are Independent of each other.
- The tester does not need knowledge of any specific programming language.
- The test is done from the point of view of the user, not the designer.
- Test cases can be designed as soon as the specifications are complete.

Testing is divided into four phases as follows

- Integration Testing
- System Testing
- Acceptance Testing

Testing is the major control measure employed for software development. After the coding phase, computer programs are available that can be executed for testing purpose.

4. Integration Testing:

During integration of modules, integration testing is performed. The goal of this is to detect design errors, while focusing on testing the interconnections between modules.

5. System Testing:

After the system is put together, system testing is performed. System testing is the stage, that aims at ensuring that the system works accurately and efficiently before live operations commence.

6. Acceptance Testing:

Finally, Acceptance testing is performed to demonstrate the operation of the system to the user, on the real time environment with the user's live data.

Testing is an extremely critical and time-consuming activity. It requires proper planning of the overall testing process.

FUTURE ENHANCEMENTS

FUTURE ENHANCEMENTS

- In future, “black box” testing may be included.

- Also, this toolkit can be further developed to analyze the performance of other languages besides JAVA.

CONCLUSION

CONCLUSION

This project will prove to be user friendly helping the user in obtaining the system details like the OS name, OS version, OS architecture, user directory, user home, its memory details and that of the drives used. It also gives the JVM details like class path, Java vendor etc. The programmer gets a clear idea about the time consumed for compilation and execution of the Java application. So the programmer can think of the ways to improve the performance of his applications. The project helped us understand about the importance of teamwork and hard work.

REFERENCES

REFERENCES

• David M. Geary, Graphic JAVA 2 Mastering the JFC, the sun Microsystems

Press –a prentice-hall, 1999.

• Patrick Naughton and Herbert Schildt, 'The Complete Reference' 'Java 2 '3rd edition – TATA McGRAW HILL.

• Vartan Piroumian, 'Java GUI Development', 1st edition – Techmedia, 1999.

APPENDIX

APPENDIX

1. sample source code:

```
import java.io.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class mem
{


            String s,s1,s2,tex="",fex="",size="",bl="";
            String v[]=new String[3];
            String v1[]=new String[3];
            String v2[]=new String[3];
            String v3[]=new String[3];
            String v4[]=new String[3];
            String v5[]=new String[3];
            public mem()throws Exception
            {
                 try
                 {

                        Runtime r=Runtime.getRuntime();
```

```
                    Process p=r.exec(msg+ " /c mem >>tt1");
                    BufferedReader b = new BufferedReader(new
FileReader("tt1"));
                    s=b.readLine();
                    s=b.readLine();
                    while((s=b.readLine())!=null)
                    {
                          if(s.startsWith("Total Expanded (EMS)"))
                          {
                                int start = s.lastIndexOf("(EMS)") + 21;
                                int end = s.indexOf("(", start);
                                tex = s.substring(start, end).trim();
                                System.out.println("tex:"+tex);

                           }

                           if(s.startsWith("Free Expanded (EMS)"))
                                {
                                      int start = s.lastIndexOf("(EMS)") + 20;
                                      int end = s.indexOf("(", start);
                                      fex = s.substring(start, end).trim();
                                      System.out.println("fex:"+fex);

                                 }

                           StringTokenizer t= new StringTokenizer(s," ");

                           while(t.hasMoreTokens())
                           {
                             s1 = t.nextToken();
                             System.out.println("s1:"+s1);
```

```
if(s1.trim().equals("Conventional"))
{
v[0]= t.nextToken();
v[1]= t.nextToken();
v[2] = t.nextToken();
      }
      if(s1.trim().equals("Upper"))

{
               v1[0]= t.nextToken();
               v1[1]= t.nextToken();
               v1[2] = t.nextToken();
      }
      if(s1.trim().equals("Reserved"))
{
               v2[0]= t.nextToken();
               v2[1]= t.nextToken();
               v2[2] = t.nextToken();
      }
      if(s1.trim().equals("Extended"))
      {
              t.nextToken();
              v3[0]= t.nextToken();
              v3[1]= t.nextToken();
              v3[2] = t.nextToken();
      }
      if(s1.trim().equals("Total"))
      {
              s2=t.nextToken();
              System.out.println("mems2:"+s2);
              if(s2.trim().equals("memory"))
```

```
                    {
                            v4[0]= t.nextToken();
                            v4[1]= t.nextToken();
                            v4[2] = t.nextToken();
                    }
                    if(s2.trim().equals("under"))
                    {
                            t.nextToken();
                            t.nextToken();
                            v5[0]= t.nextToken();
                            v5[1]= t.nextToken();
                            v5[2] = t.nextToken();
                    }
            }

            if(s1.trim().equals("size"))
                    size=t.nextToken();
            if(s1.trim().equals("block"))
                    bl=t.nextToken();
  }
}

}catch(Exception e){System.out.println("from e:"+e);}
            for(int i=0;i<3;i++)
            {
            System.out.println("val:"+v[i]);
            System.out.println("val1:"+v1[i]);
            System.out.println("val2:"+v2[i]);
            System.out.println("val3:"+v3[i]);
            System.out.println("val4:"+v4[i]);
            System.out.println("val5:"+v5[i]);
```

```
                        }
                        System.out.println("tex:"+tex);
                        System.out.println("fex:"+fex);
                        System.out.println("size:"+size);
                        System.out.println("block:"+bl);
                        /*for(int i=0;i<3;i++)
                        System.out.println("val1:"+v1[i]);
                        for(int i=0;i<3;i++)
                        System.out.println("val2:"+v2[i]);*/

                }

}

//package pack;
import java.io.*;
import java.util.*;
 public class loops2
 {

    String s,s1,s2,s3,s4,s5,s6;
    int x=0,y=0,l,i=0,z=0;
    boolean f=false,b1=false;
    int n,c=0,c1=0;
    long st=0,end=0;
    char a;
    String lp;
    public Vector fv=new Vector();
    public Vector wv=new Vector();
    public Vector dwv=new Vector();
public loops2(String pt) throws IOException
{
  RandomAccessFile b=new RandomAccessFile(pt,"r");
      while(true)
      {
        x=y=z=0;
        st=0;end=0;
   while((s=b.readLine())!=null)
   {
                  c++;
                  f=false;
                    StringTokenizer t = new StringTokenizer(s," ()");

        while(t.hasMoreTokens())
        {
        s1 = t.nextToken();

if((s1.trim().equals("for"))||(s1.trim().equals("while"))||(s1.trim().equals("do")))
                        {
                              if(s1.trim().equals("while"))
                              {
                                if(s.endsWith(";"))
                                break;
                              }
                              st=System.currentTimeMillis();
                              if(s1.trim().equals("for"))
                              {
                                      lp=s1;
                                      fv.add("Main for loop");
                                      fv.add("Line No: "+c);
                              }
                    if(s1.trim().equals("while"))
                    {
                            lp=s1;
                            wv.add("Main while loop");
                            wv.add("Line No: "+c);
                    }
                    if(s1.trim().equals("do"))
                    {
                            lp=s1;
                            dwv.add("Main do loop");
                            dwv.add("Line No: "+c);
                    }



                System.out.println("st:"+st);
                System.out.println("s:"+s);
                     System.out.println("Main For loop:"+c);
            while(t.hasMoreTokens())
                     {
                       s2 = t.nextToken();
                       if(s2.startsWith("{"))
        f=true;
    }
  if (f==false)
  {
  while((s3=b.readLine())!=null)
  {
                              c++;
              if(s3.trim().startsWith("{"))
   f=true;
  if (f==true)
```

```
            break;
                          }
           }
                    }
       }

           if (f==true)
            break;
       }



//inner

while((s4=b.readLine())!=null)
    {
                   c++;
       //System.out.println("s4:"+s4);
       if((s4.trim().startsWith("for"))||((s4.trim().startsWith("while"))&&
(!s4.trim().endsWith(";")))||(s4.trim().startsWith("do")))
       {

                          b1=true;
                          if(s1.trim().equals("for"))
                           {
                                fv.add("Inner for loop");
                                fv.add("Line No: "+c);
                           }
                          if(s1.trim().equals("while"))
                          {
                                wv.add("Inner while loop");
                                wv.add("Line No: "+c);
```

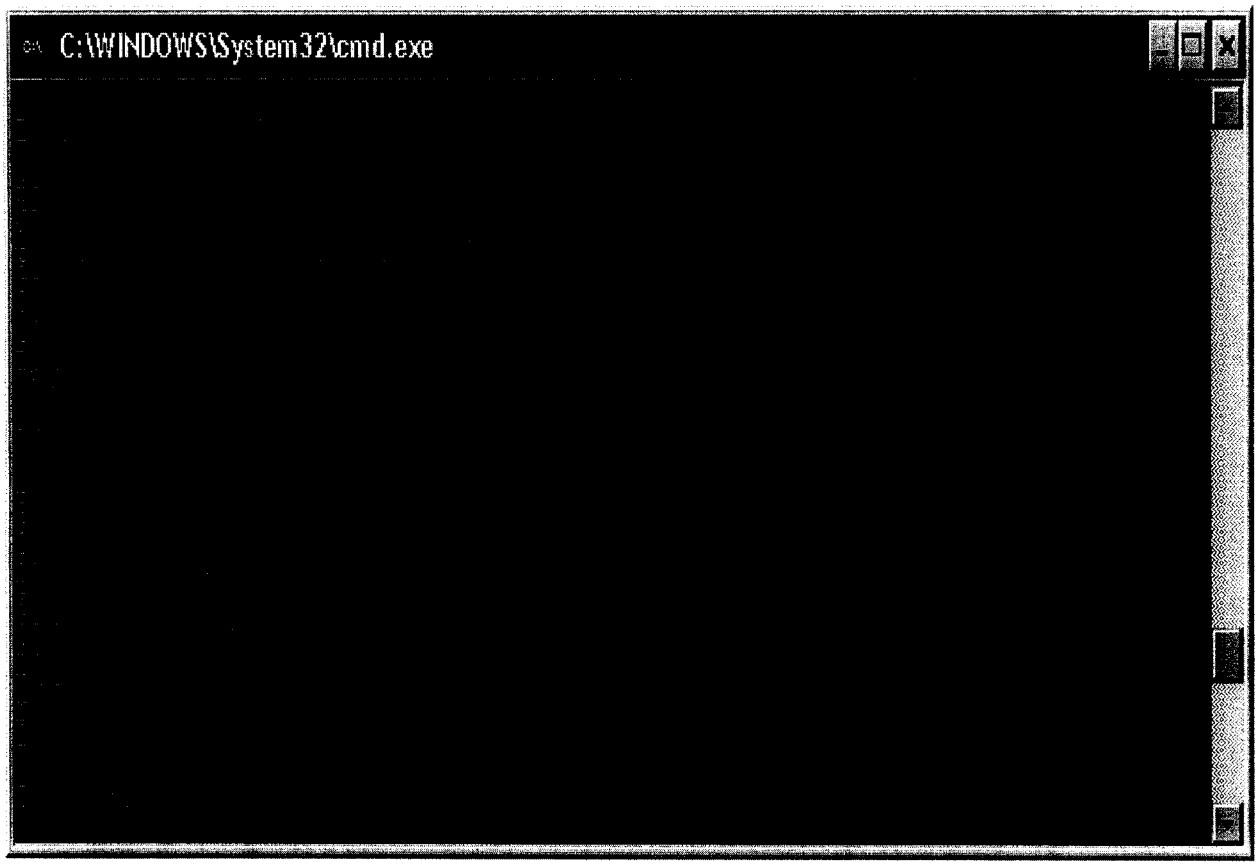
```
                }
                if(s1.trim().equals("do"))
                {
                        dwv.add("Inner do loop");
                  dwv.add("Line No: "+c);
                }
System.out.println("Inner Loops:"+c);
System.out.println("Inner Loop:"+s4);
            StringTokenizer t = new StringTokenizer(s4," ();");

while(t.hasMoreTokens())
{
                  s1 = t.nextToken();
                 // System.out.println("s1:"+s1);
             if(s1.startsWith("{"))
                         { x++;
                          // System.out.println("x:"+x);
                           b1=false;
                           }
                        if(s1.startsWith("}"))
                          {    y++;
                                z=x-y;
                                //System.out.println("y:"+y+"  z1:"+z);
                                    if(z==-1)

                                          break;



                             }
          }
          if(b1==true)
```

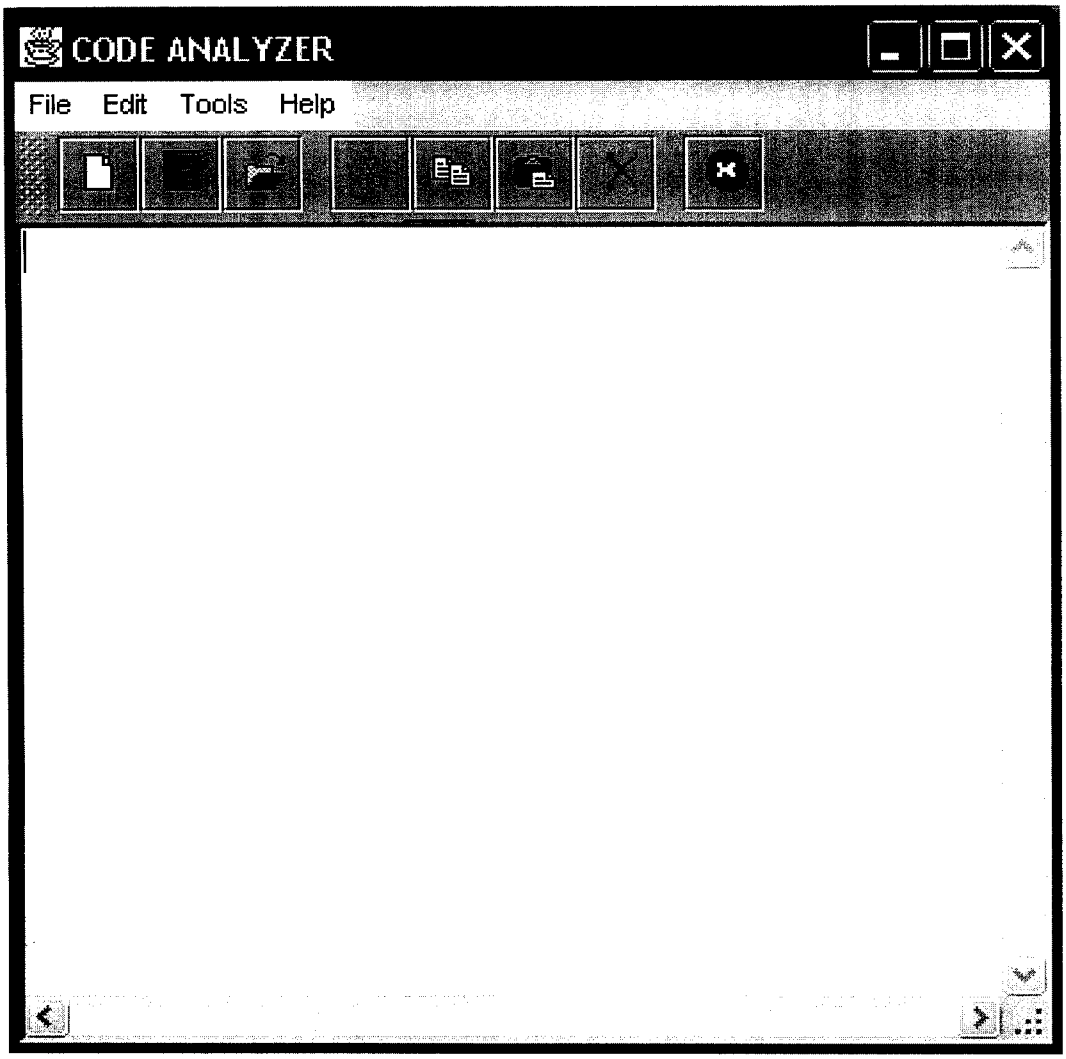
```
    {
  s2=b.readLine();
  c++;
   StringTokenizer tt = new StringTokenizer(s2," ();");

                      while(tt.hasMoreTokens())
                     {
                                   s3 = tt.nextToken();


                        if(s3.trim().startsWith("{"))
                             {
                                  x++;
                                  //System.out.println("x:"+x);
                                  b1=false;
                          }
                          if(s3.startsWith("}"))
                             {
                                  y++;
                                  z=x-y;
                                  //System.out.println("y:"+y+"  z1:"+z);
                                  if(z==-1)
                                        break;
                          }
                      }

      }
    }

 else
 {
```

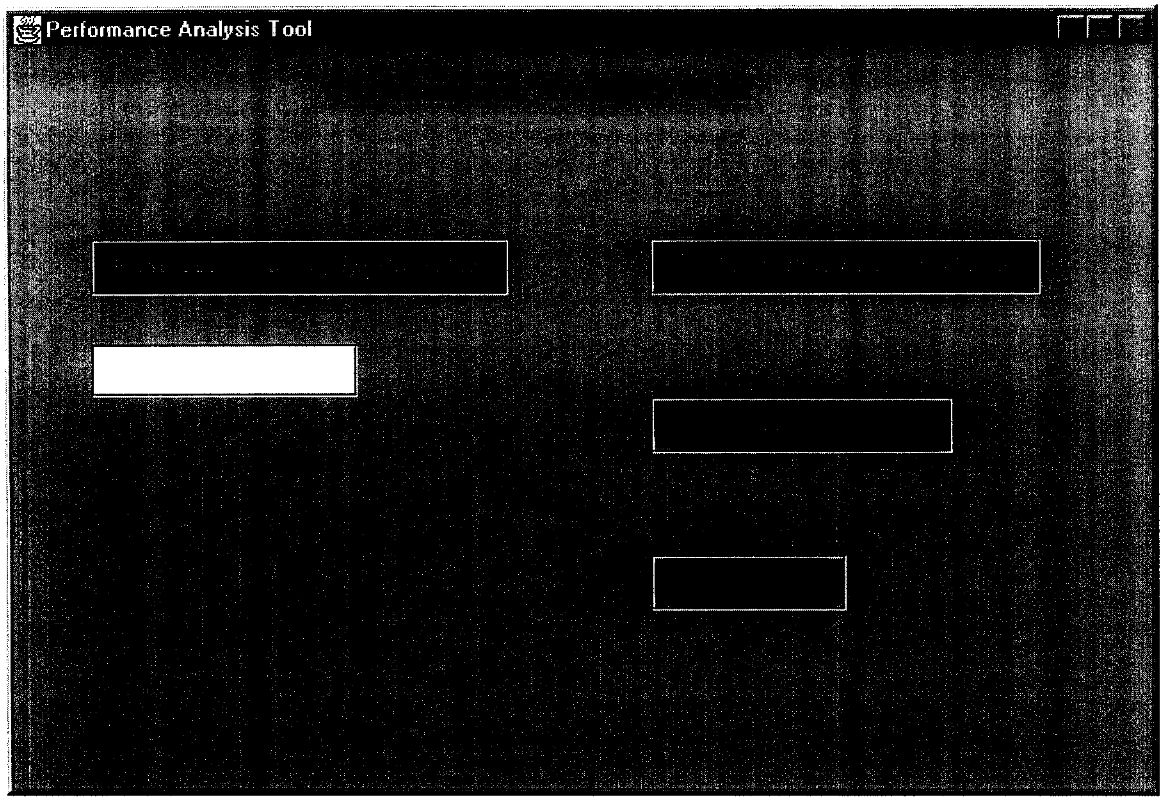
```
                    StringTokenizer t1 = new StringTokenizer(s4," ();");
     while(t1.hasMoreTokens())
     {
                              s2=t1.nextToken();
       if(s2.trim().startsWith("{"))
                    {
                          x++;
                          //System.out.println("x:"+x);
                    // b1=false;


                    }

                    else
                    {

                              if(s2.trim().startsWith("}"))
              {
                                   y++;
                                   z=x-y;
                                   //System.out.println("y:"+y+"  z1:"+z);
                                   if(z==-1)
                                         break;
                      }
                     }
             }
            }
if(z==-1)
{
         System.out.println("x:"+x);
         System.out.println("y:"+y);
         end=System.currentTimeMillis();
```

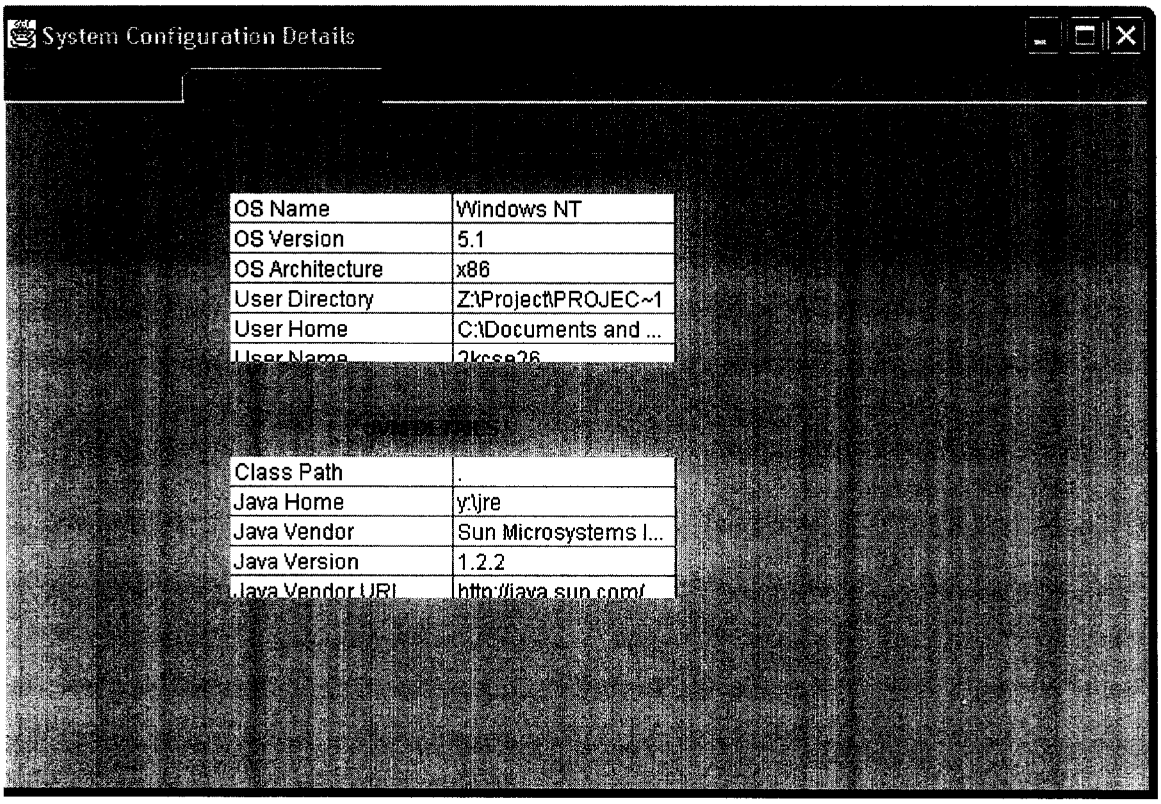
```
                System.out.println("end:"+end);
                System.out.println("Execution Time :"+(end-st));
                if (lp.trim().equals("for"))
                fv.add("Execution Time :"+(end-st));
                if (lp.trim().equals("while"))
                wv.add("Execution Time :"+(end-st));
                if (lp.trim().equals("do"))
                dwv.add("Execution Time :"+(end-st));
                break;
            }

        }


if((s==null)||(s4==null))
     break;
}
for(int i=0;i<fv.size();i++)
{
        System.out.println("f:"+fv.elementAt(i));
}
for(int i=0;i<wv.size();i++)
{
        if(wv.isEmpty())
        System.out.println("No Loop");
        else
        System.out.println("w:"+wv.elementAt(i));
}
for(int i=0;i<dwv.size();i++)
{
        System.out.println("d:"+dwv.elementAt(i));}}}
```

2. Sample Screens:



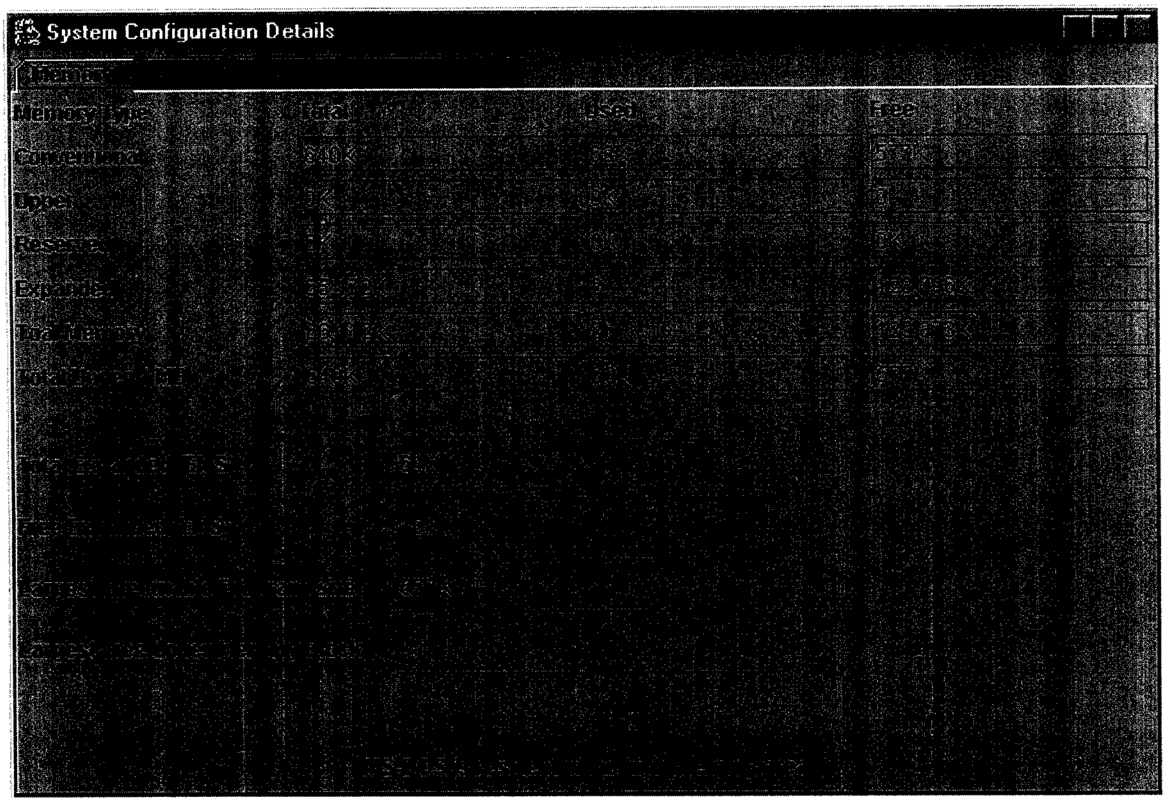

CODE ANALYZER
File
Edit
Tools
Help





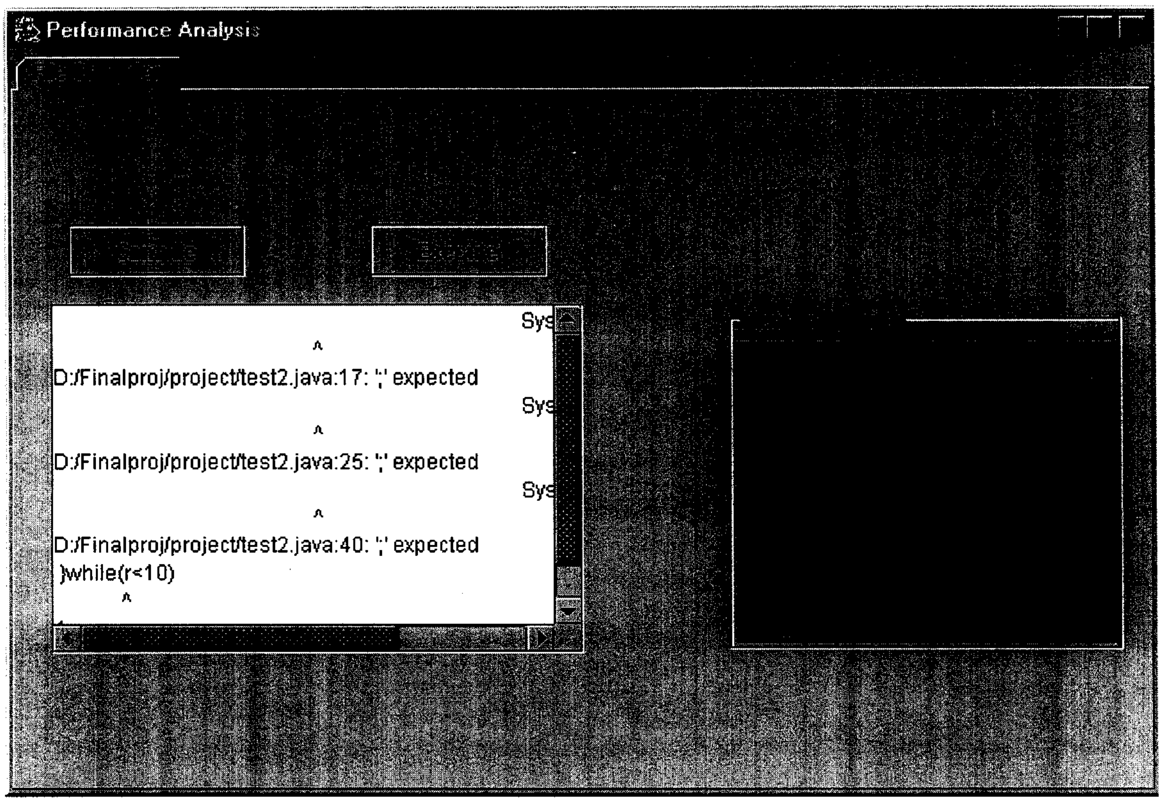
OS Name	Windows NT
OS Version	5.1
OS Architecture	x86
User Directory	Z:\Project\PROJEC~1
User Home	C:\Documents and ...
User Name	2kcse26

Class Path	.
Java Home	y:\jre
Java Vendor	Sun Microsystems I...
Java Version	1.2.2
Java Vendor URL	http://java.sun.com/



System Configuration Details

Memory Type	Total	Used	Free
Conventional	640K	[illegible]	[illegible]
Upper	[illegible]	[illegible]	[illegible]
Reserved	[illegible]	[illegible]	[illegible]
Expanded	[illegible]	[illegible]	[illegible]
Total memory	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]



Performance Analysis

```
                                         Sys
                         ^
D:/Finalproj/project/test2.java:17: ';' expected
                                         Sys
                         ^
D:/Finalproj/project/test2.java:25: ';' expected
                                         Sys
                         ^
D:/Finalproj/project/test2.java:40: ';' expected
 }while(r<10)
          ^
```