Application Note

Quick Start

This documentation explains getting started with TCAT for Java / Windows using a demonstration test case. It then describes the main features of the product.

1.1 Getting Acquainted with TCAT for Java/Windows

This section will familiarize you with the main activities involved in using **TCAT for Java/Windows**, including instrumenting, compiling, and running the target program, and finally, looking at the resulting coverage reports, calltree graphs and digraphs.

The applet used to illustrate the operation of **TCAT for Java/Windows** in Windows is *TicTacToe*, which you will prepare and instrument as a test application. You can then exercise various logical branches or segments of *TicTacToe*, creating trace files from which the coverage reports are generated. It is recommended that you complete the *TicTacToe* example before continuing.

If you are using **TCAT for Java/Windows** for the first time, you will benefit most if you refer to chapters 4 through 7 for in-depth operational instructions and detailed explanation of functionality. If you are an intermediate user, you'll only have to refer to those menu definitions which need further explanation.

1.1.1 Step 1 - Preparing and Instrumenting TicTacToe

1.1.1.1 Setup Environment Variables

For the first time user, check your Java manual to see how the environment variables are set. Add $*TCAT_Java_DIR\Program\* to CLASSPATH.

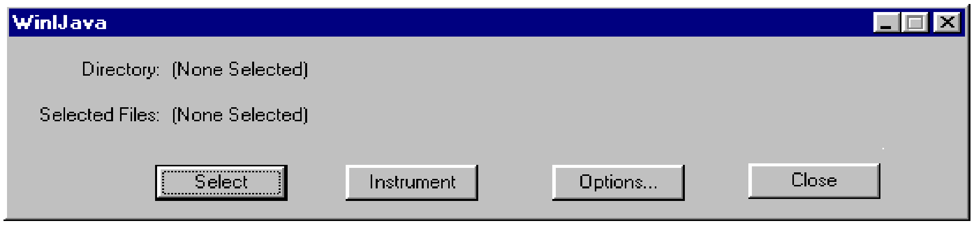
(e.g. set CLASSPATH=.;C:\jdk1.1.4\lib\classes.zip; C:\Program Files\Software Research\Coverage\TCAT-Java\Program))

1.1.1.2 Instrument Using WinIJava

WinIJava instruments the application under test so that any tests can produce trace files.

To instrument the example application:

1. Start up **WinIJava.**



Directory: (None Selected)

Selected Files: (None Selected)

FIGURE 1 WinIJava Window

2. Select *TicTacToe.java* using the **Select** button.

Note: More than one file can be selected and instrumented, and instrumenting multiple files results in more thorough coverage.

3. Select **Instrument**.

   A copyright box pops up before the instrumentation of each file if the license is invalid. During instrumentation, a command-line window displays messages and warnings. The instrumentor parses the applet's source code, looking for logical branches or segments and inserting markers (function calls).

   Instrumenting a program does not change its functionality. When compiled, and executed, the instrumented application behaves as it normally does, except that it writes coverage data to a trace file. For more information on **TCAT for Java/Windows**' instrumentor, refer to *Chapter 4* on the *User's Guide*.

4. When instrumentation is complete, select **Close** from the **WinIJava** window.

Instrumenting *TicTacToe.java* produces the following files in the *TicTacToe* directory:

- *TicTacToe.i* — the instrumented version of the source file

  This file is updated during the instrumentation process.

- *TicTacToe.dg* — a Directed Graph Listing file

  Each instrumented file should have its own *.dg* file.

- *TicTacToe.cg* — a Calltree Graph Listing file

  Each instrumented file should have its own *.cg* file.

- *Prj_Name.mdf* — a Module Definition file

  This file contains information about segments and callpairs in all the processed files.

- *mdf.pro* — a profile of the applet for useing on your Web server.

1.1.2 Step 2 - Executing the Instrumented Application

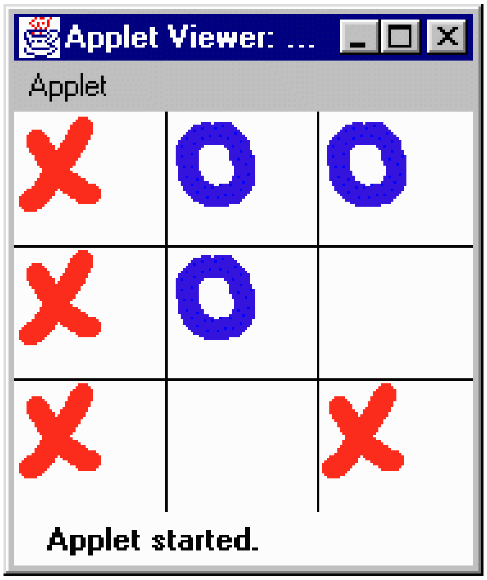
During instrumentation, **TCAT for Java** inserted function calls at each logical branch it found. In order to later determine the C1 coverage, you must run the applet.

By running *TicTacToe* and playing the game, you are exercising segments of the *TicTacToe* program. Because you have instrumented the applet, the exercise will create a trace file and allow you to view coverage information on the exercise.

To run the instrumented applet:

1. Open a **DOS** window. From DOS prompt, CD to *TicTacToe* directory.
2. Type **appletviewer** *TicTacToe*.html.
3. The appletviewer and *TicTacToe* applet will appear. Play the game.
4. When you are finished playing, select the **Applet** and choose “Quit”.

When the *TicTacToe* is running, your display should look like this:



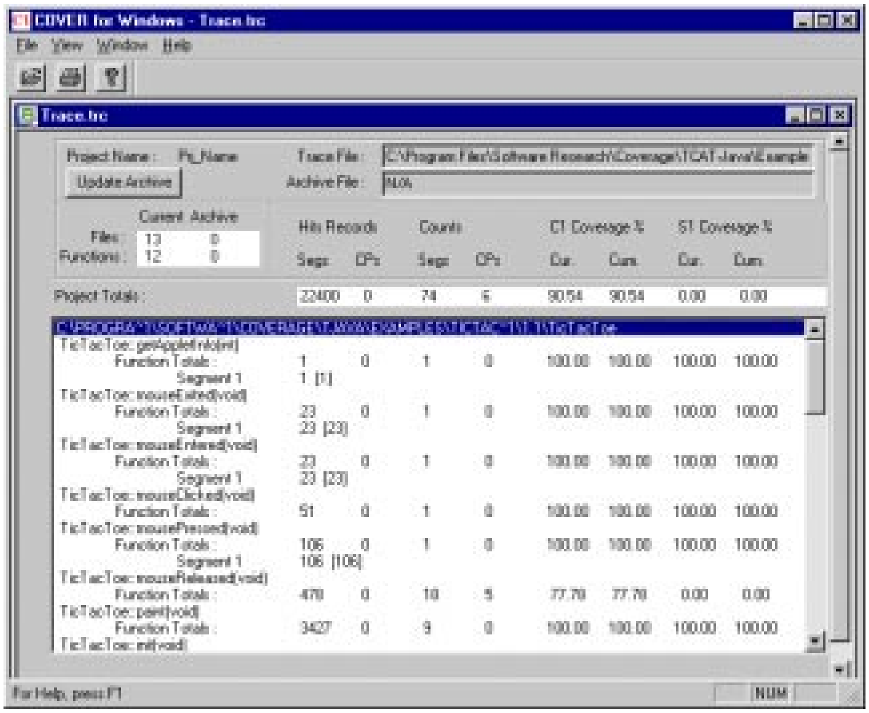
Applet started.

FIGURE 2 Testing TicTacToe

1.1.3 Step 3 - Viewing Coverage Reports Using Cover

1. From the **Program menu**, select **TCAT for Java 1.2** folder.
2. From the resulting window, select **Cover** icon.
3. From the **File** menu, select **Open**.
4. In the **Open** dialogue**,** click on the filename *Trace.trc* from the *tcat_db* \ Prj_Name directory.

A coverage report of the test you ran on the example program appears.



```
Project Name :   Prj_Name          Trace File :   C:\Program Files\Software Research\Coverage\TCAT-Java\Example
                                   Archive File : N/A

            Current  Archive
Files:      13       0              Hits Records    Counts       C1 Coverage %     S1 Coverage %
Functions : 12       0              Segs   CPs      Segs  CPs    Cur.    Cum.      Cur.    Cum.

Project Totals :                    22400  0        74    6      90.54   90.54     0.00    0.00

C:\PROGRA~1\SOFTWA~1\COVERAGE\T-JAVA\EXAMPLES\TICTAC~1\1.1\TicTacToe
TicTacToe::getAppletInfo(int)
        Function Totals :           1      0        1     0      100.00  100.00    100.00  100.00
                  Segment 1         1  [1]
TicTacToe::mouseExited(void)
        Function Totals :           23     0        1     0      100.00  100.00    100.00  100.00
                  Segment 1         23  [23]
TicTacToe::mouseEntered(void)
        Function Totals :           23     0        1     0      100.00  100.00    100.00  100.00
                  Segment 1         23  [23]
TicTacToe::mouseClicked(void)
        Function Totals :           51     0        1     0      100.00  100.00    100.00  100.00
TicTacToe::mousePressed(void)
        Function Totals :           106    0        1     0      100.00  100.00    100.00  100.00
                  Segment 1         106  [106]
TicTacToe::mouseReleased(void)
        Function Totals :           478    0        10    5      77.78   77.78     0.00    0.00
TicTacToe::paint(void)
        Function Totals :           3427   0        9     0      100.00  100.00    100.00  100.00
TicTacToe::init(void)

For Help, press F1
```

FIGURE 3 Cover Main Window Displaying Coverage Report on TicTacToe

Cover displays trace and coverage information on your development project in a treelike list. Clicking on a branch of the list expands the branch and shows its contents, and also contracts it. The several fields in the report have the following meanings:

Hits The number of times the segment and call pair were executed during the test

Count The number of segments and call pairs within the function

C1 The percentage of branch coverage for each function

S1 The percentage of call pair coverage for the function

For detailed information about **Cover**, see *Chapter 5* on the *User's Guide*.

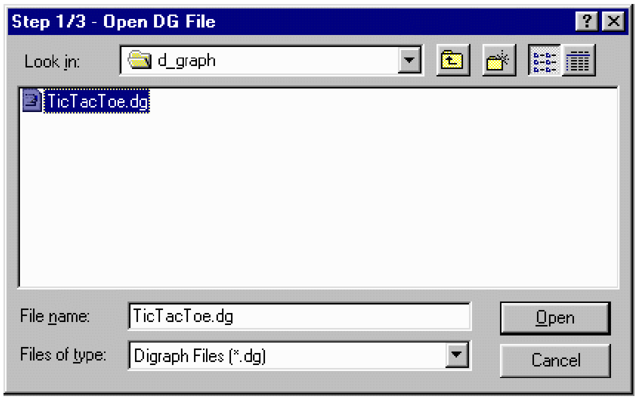
1.1.4 Step 4 - Viewing Directed Graphs with DiGraph

To view a directed graph (digraph) of possible program flows of a function:

1. From the **TCAT Program Group**, select **DiGraph**.
2. Using the **File** menu, select **Open**.
3. A selection box asks for the name of the directed graph to view. For this example, find the TicTacToe.dg file under the *tcat_db \ Prj_Name \ d_graph* directory.

   A selection box asks for the name of the module definition file.

4. Find the *TicTacToe.mdf* file under the *tcat_db \ Prj_Name* directory (one level up from the *TicTacToe.dg* file).



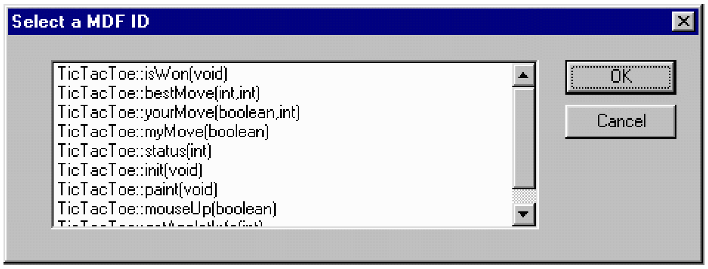
File name: TicTacToe.dg

Files of type: Digraph Files (*.dg)

FIGURE 4 DiGraph Open Dialog Box

A selection box asks which function to display.

5. For this example, select ***TicTacToe::status(int)***.



TicTacToe::isWon(void)
TicTacToe::bestMove(int,int)
TicTacToe::yourMove(boolean,int)
TicTacToe::myMove(boolean)
TicTacToe::status(int)
TicTacToe::init(void)
TicTacToe::paint(void)
TicTacToe::mouseUp(boolean)

FIGURE 5 Select MDF ID Box

A directed graph depicting possible program flows of the function ***TicTacToe::status(int)*** appears.

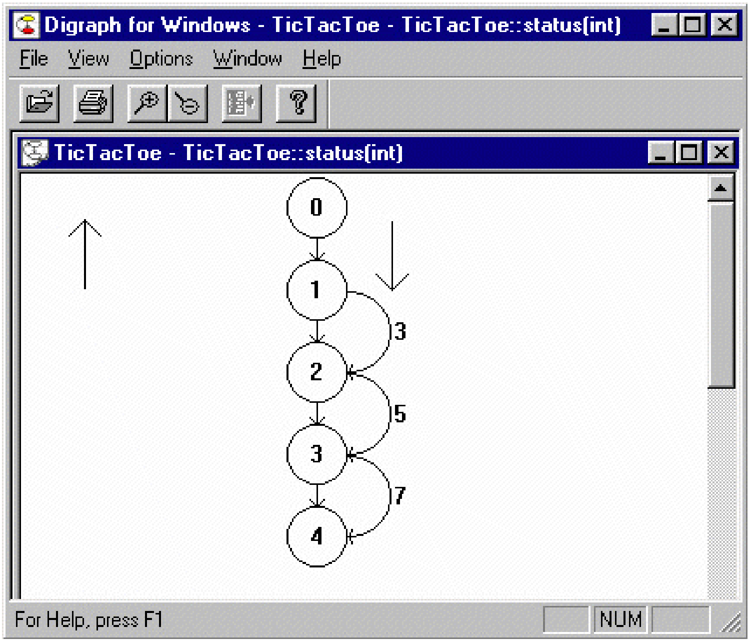

Digraph for Windows - TicTacToe - TicTacToe::status(int)
File View Options Window Help
TicTacToe - TicTacToe::status(int)
0
1
2
3
4
3
5
7
For Help, press F1
NUM

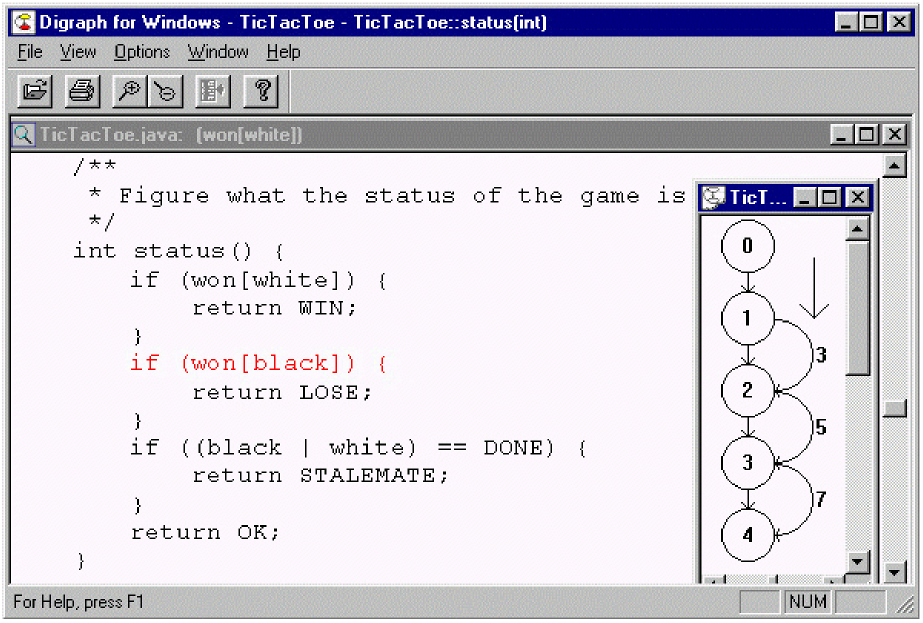
FIGURE 6 Directed Graph of TicTacToe

The digraph shows the set of conditions and paths that make up a function. The next step shows how to look at the code that the digraph displays as numbered segments.

1.1.5 Step 5 - Viewing Source Code from a Digraph

To view the source code represented by a particular segment of the function ***TicTacToe::status(int)***:

1. Click near the number of the segment.
2. From the tool bar, select the **View Source Code** button.


Digraph for Windows - TicTacToe - TicTacToe::status(int)
File
View
Options
Window
Help
TicTacToe.java: [won[white]]
TicT...
0
1
2
3
4
3
5
7
For Help, press F1
NUM

```
/**
 * Figure what the status of the game is
 */
int status() {
    if (won[white]) {
        return WIN;
    }
    if (won[black]) {
        return LOSE;
    }
    if ((black | white) == DONE) {
        return STALEMATE;
    }
    return OK;
}
```

FIGURE 7 Source Code Associated with Segment 3 of Digraph of ***TicTacToe::status(int)***

1.1.6 Step 6 - Viewing a Calltree

To view a calltree of ***TicTacToe***:

1. From the **TCAT Program Group**, select **Calltree**.
2. In the **File** menu, select **Open**.

   You are prompted for the name of the calltree to view.

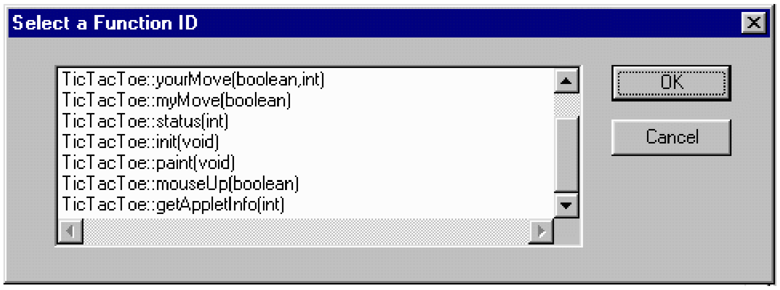
3. Find *TicTacToe.cg* file under the *tcat_db\Prj_Name\ c_graph* directory.

   You are prompted for the name of the database file.

4. Find the *TicTacToe.mdf* file under the *tcat_db\Prj_Name* directory.

   A window appears asking you which function to display.

5. For this example, select ***TicTacToe::mouseup(boolean)***.



TicTacToe::yourMove(boolean,int)
TicTacToe::myMove(boolean)
TicTacToe::status(int)
TicTacToe::init(void)
TicTacToe::paint(void)
TicTacToe::mouseUp(boolean)
TicTacToe::getAppletInfo(int)

FIGURE 8 Select Function ID Box

A calltree depicting the selected function appears.

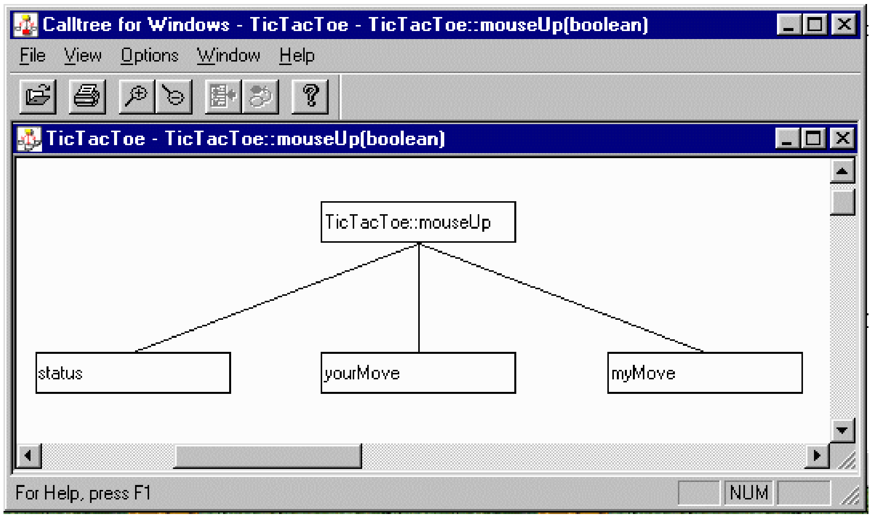

Calltree for Windows - TicTacToe - TicTacToe::mouseUp(boolean)
File View Options Window Help
TicTacToe - TicTacToe::mouseUp(boolean)
TicTacToe::mouseUp
status
yourMove
myMove
For Help, press F1
NUM

FIGURE 9 Displaying a Calltree

The calltree shows all of the callpairs associated with the function ***TicTacToe::mouseup(boolean)***.

The next step shows how to look at digraphs of the possible program flows belonging to this function.

1.1.7 Step 7 - Viewing the Directed Graph Associated With a Calltree Node

To display a directed graph of any callpair shown in the calltree:

1. Select a node by clicking on it.

   Notice that the **View Digraph** button on the toolbar now has a red arrow, indicating that it is available.

2. To display a directed graph of the selected function, click the View DiGraph button.

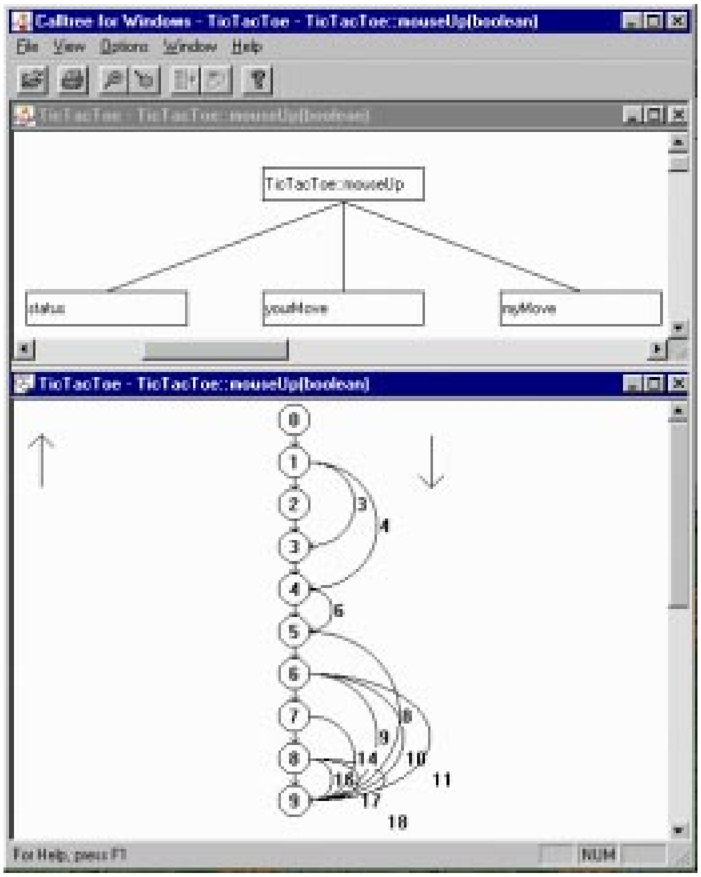

Calltree for Windows - TicTacToe - TicTacToe::mouseUp(boolean)
File View Options Window Help
TicTacToe - TicTacToe::mouseUp(boolean)
TicTacToe::mouseUp
status
yourMove
myMove
TicTacToe - TicTacToe::mouseUp(boolean)
0
1
2
3
4
5
6
7
8
9
3
4
6
8
9
14
10
11
16
17
18
For Help, press F1
NUM

FIGURE 10 Calltree of ***TicTacToe::mouseup(boolean)*** and Digraph of Its Possible Program Flows

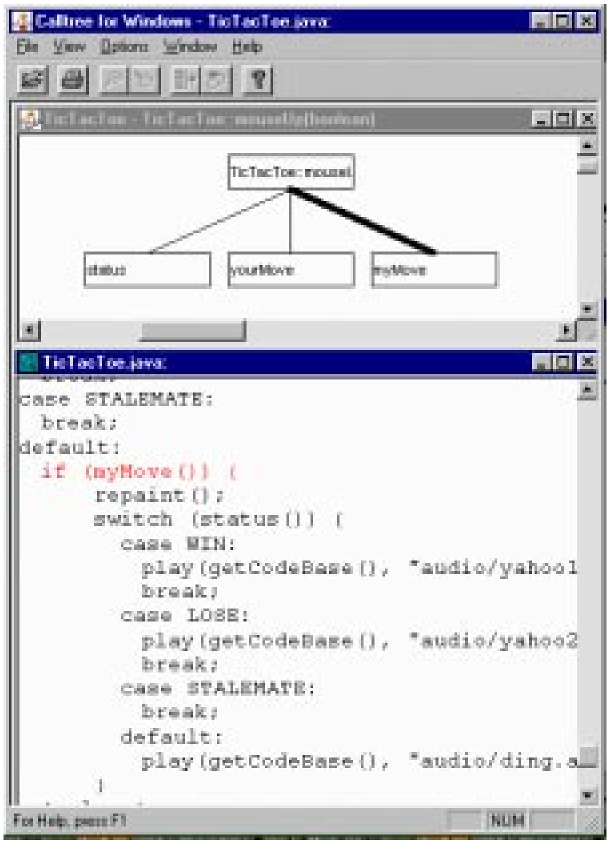
1.1.8 Step 8 - Viewing the Source Code Associated With a Calltree

You can view the source code associated with any node in a calltree by clicking on the corresponding edge.

Notice that the **Source Code** button on the Tool Bar has a red arrow.

1. To display the associated source code, click the Source Code button.

   The code is displayed in a separate window with the calling statement highlighted in red.


Calltree for Windows - TicTacToe.java:
File View Options Window Help
TicTacToe - TicTacToe::mouseUp(boolean)
TicTacToe::mouseU
status
yourMove
myMove
TicTacToe.java:
For Help, press F1
NUM

```
case STALEMATE:
  break;
default:
  if (myMove()) {
      repaint();
      switch (status()) {
        case WIN:
          play(getCodeBase(), "audio/yahoo1
          break;
        case LOSE:
          play(getCodeBase(), "audio/yahoo2
          break;
        case STALEMATE:
          break;
        default:
          play(getCodeBase(), "audio/ding.a
      }
```

FIGURE 11 Source Code Window Displayed from Calltree

1.1.9 Step 9 - Closing TCAT for Java/Windows

After looking at the source code, select one of the following options to complete the session.

To close **TCAT for Java/Windows:**

- Select **File|Exit** from the menu bar of each open program.
- **In Windows NT:** double-click on the frame window **Close Box** of each program.
- **In Windows 95 and 98:** click on the frame window **Close Box** of each program.

You have now seen all the main features of **TCAT for Java/Windows**.

1.2 Summary

If you have completed the proceeding steps successfully, you have seen and practised the basic skills you need to use TCAT for Java / Windows productively. You should have learned how to invoke TCAT for Java / Windows, how to instrument, compile, and run a program, and how to look at the coverage reports.

For best learning you may want to:

- Repeat STEPS 1 - 9 without the manuall and experiment by running the applet several times and looking at the amount of coverage your test input receives.
- Repeat STEPS 1 - 9 with you applet.
- Review the chapters on system operation where you had difficulties. The table of contents can help you locate the topic you want.