



DECLARATION

**A PROFILER FOR TESTING JAVA PROGRAMMES
(Socket Server and the Analyser)**

A PROJECT REPORT PRESENTED BY

NISHANTHA SRIPALI WEERAKOON

ACC NO	29876
CALL NO.	005.133 WEE

to the Board of Study in Statistics and Computer Science of the

POSTGRADUATE INSTITUTE OF SCIENCE



*in partial fulfilment of the requirement
for the award of the degree of*

MASTER OF SCIENCE IN COMPUTER SCIENCE

of the

UNIVERSITY OF PERADENIYA

SRI LANKA

2003

ABSTRACT

Java programming language was first designed to be used as a controlling language for consumer electronic devices but later this was redesigned to be used as a platform independent programming language, which uses an interpreter that can be embedded in other applications. With this redesign, Java became very popular among, especially, web based application programmers. Java programmes began to grow larger and larger. In the meantime, most of larger Java programmes were not satisfactory in performance and tools were required to analyse and measure the performance of larger Java programmes.

Many tools were built for testing Java programmes and most of these tools used profiling as their approach. Almost all of the early Java profilers used an instrumented Java VM to obtain the necessary information. There were many drawbacks of those profiling tools so that the Sun Microsystems Inc. introduced a general-purpose profiling instrumentation, JVMPI, with their standard Java release. At the moment this is used as a standard for the development of profiling tools. The most interested areas of profiling are the memory usage, CPU usage and monitor contention. In this project a general-purpose memory profiler, which is useful in analysing memory leakage problems in a single Java VM is designed.

In this profiling process, JVMPI is programmed to provide information on class loading, object allocation, object release and object movement events. These data are then sent to the profiler front-end and stored there. Since memory problems can be identified by visualising the obtained information graphically, all information regarding the classes and objects are presented to the user through a graphical user interface. The system uses advanced Java features such as JVMPI to obtain the information from the Java VM, JNI to incorporate native codes written to receive information from the profiler agent and Swing to build a decent graphical user interface.

A makefile is provided to automate the compilation and installation of all the source files using Unix make utility.

TABLE OF CONTENTS

CHAPTER 01: Introduction	1
01.1. The Java Language: From Browsers to Servers	1
01.2. Performance Tuning: A Major Problem.....	2
01.3. Profiling: Java's Approach to Performance Tuning.....	4
01.4. Objectives of the Project: Building a Memory Profiler.....	5
CHAPTER 02: literature review.....	7
02.1. Existing Profiling Tools: A Review of Commercial Profilers	7
02.2. Typically obtained Information	9
02.3. Critical Comparison: Existing profilers and the developed profiler.....	10
CHAPTER 03: The Design of the profiling tool	12
03.1. Prerequisites: The Environment.....	12
03.2. General Design: An Overview.....	12
03.3. Obtaining the Information: The Profiler Agent.....	13
03.4. Analysing and Visualising Data: The Front End.....	16
03.5. Compiling and Installing the Profiler: Using the Make Utility.....	21
CHAPTER 04: results of the design process	23
04.1. The Profiler: An Overview	23
04.2. Configuration: How to Compile and Install the files	26
04.3. Interpretation: Analysis of the Results	27
CHAPTER 05: Discussion and conclusions.....	30
05.1. Strong Points: Advantages of The Profiler.....	30
05.2. Weak Points: Drawbacks of the Profiler.....	30
05.3. Further Developments: The Way Forward.....	31
APPENDIX 1: Programme listings.....	33
01.1. Profiler Programme	33
01.2. The Graphical User Interface.....	60
01.3. The Scale.....	71
01.4. The Profiler Agent.....	74
01.5. The Native Socket Implementation	79
01.6. The Building and Installation Script.....	82
APPENDIX 2: Bibliography.....	84