### Supplement: Java Profiler

### For Introduction to Java Programming By Y. Daniel Liang

#### 1 Introduction

Java profiler is a tool for analyzing the performance of Java programs. Using this tool, you can obtain the CPU time used by the methods in the program and memory used by objects. This tool is integrated with NetBeans. This supplement introduces how to use Java profiler from NetBeans.

NOTE: Assume that you know how to use NetBeans. For information on creating projects and programs and running programs on NetBeans, see Supplement II.B.

#### 2 Calibrating Profiler

Before using profiler for the first time, you need to calibrate the profiler to achieve accurate profiling results. Calibration needs to be performed only once for the Java platform used to run your program. Here are the steps to perform calibration:

- 1. Choose Profile, Advanced Commands, Run Profiler Calibration.
- 2. Select a Java platform, as shown in Figure 1, and click *OK*.

Select Java Pla	atform to calibrate			
Select Java Platform to calibrate:				
JDK 1.7 (Default) JDK 1.7.0				
	OK Cancel			

### Figure 1

You need to select a Java platform to perform calibration.

If you have multiple platforms, calibrate each at a time. Calibration data for each Java platform will be used for profiling programs running on the Java platform.

#### 3 Profiling an Application

© Copyright Y. Daniel Liang, 2005

Listing 23.6 SetListPerformanceTest.java in the text gives a program that shows the execution time of (1) testing whether an element is in a hash set, linked hash set, tree set, array list, and linked list, and (2) removing elements from a hash set, linked hash set, tree set, array list, and linked list. We will use this program as an example to demonstrate how to use the profiler. Here are the steps to profile this program.

- 1. Create a project for Java application.
- Create a class named SetListPerformanceTest in the project.
- Copy and paste the code in Listing 23.6 SetListPerformanceTest.java for the class in NetBeans, as shown in Figure 2.



#### Figure 2

The class SetListPerformanceTest is created in the project.

 Right-click on SetListPerformanceTest.java in the project to display a context menu and choose Profile File, as shown in Figure 3.

© Copyright Y. Daniel Liang, 2005

© 2410spring2012 - NetBeans IDE 7.0.1			
File Edit View Navigate Source Refactor R	un Debug Profile	Team Tools Wind	low Help Q Search (Ctrl+I)
🗄 🔁 🔛 🤚 📁 🍊 🛛 <defaul< th=""><th>t config&gt; 👻</th><th>T 💓 🕨 🗄</th><th><b>B</b> • <b>(b</b> •</th></defaul<>	t config> 👻	T 💓 🕨 🗄	<b>B</b> • <b>(b</b> •
Projects @ 28 Files Services	Profiler	Start Page	🗱 🚳 SetListPerformanceTest.java 🕺 🔹 👘
DescriptionPanel.java		^ 🛛 🐼 🖉 🗸	
Exercise24_13.java		1 - im	port java.util.*;
Exercise 25_03.java		2	blie elses Catlistrations and I
Exercise25_16.java	3 public class SetListPerformanceTest (		
Heap.java		5	
ListDemo.java		6 🖵	<pre>public static void main(String[] args) {</pre>
Test java	Open		// Add numbers 0, 1, 2,, N - 1 to the array lis
Test1.java	C-4	CHL Y	<pre>List<integer> list = new ArrayList<integer>();</integer></integer></pre>
TestGUI.java	Cut	Ctrl+X	for (int i = 0; i < N; i++)
🗄 🔂 Libraries	Copy	Ctrl+C	Collections.shuffle(list): // Shuffle the array lis
Burner 3720tall2011	Commile File	Ctri+V	
AddressWebServiceClientProject	Complie File	F9	// Create a hash set, and test its performance
🕀 🍯 AnagramGame	Run File	Shift+F6	Collection <integer> set1 = new HashSet<integer>(lis</integer></integer>
the chapter 42 servlet example	Debug File	Ctrl+Shift+F5	System.out.println("Member test time for hash set i
the start and th	Profile File		<pre>getTestTime(set1) + " milliseconds"); Supton sup spintle ("Denome alegent time for back as</pre>
that the second	Test File	Ctrl+F6	getRemoveTime(set1) + " milliseconds"):
chapter 44jsfexercise	Debug Test File	Ctrl+Shift+F6	
🕀 🎂 chapter 50	Add		// Create a linked hash set, and test its performar
I iangweb	Delete	Delete	Collection <integer> set2 = new LinkedHashSet<intege< th=""></intege<></integer>
	Save As Template	, Denete	System.out.println("Member test time for linked has
Burger WebServiceProject	Sure As remplate		<pre>getTestTime(set2) + " milliseconds"); Support and an analysis of the set of the set</pre>
SetListPerformanceTest.iava - Navigator	Find Usages	Alt+F7	+ getRemoveTime(set2) + " milliseconds"):
Members View	Refactor	•	······································
SetListPerformanceTest	BeanInfo Editor		// Create a tree set, and test its performance
getRemoveTime(Collection <integer< th=""><th></th><th>C. 1. 510</th><th>Collection<integer> set3 = new TreeSet<integer>(lis</integer></integer></th></integer<>		C. 1. 510	Collection <integer> set3 = new TreeSet<integer>(lis</integer></integer>
<pre>getTestTime(Collection <integer> c main(String[] args)</integer></pre>	File Members	Ctrl+F12	System.out.println("Member test time for tree set i
N:int	File Hierarchy	Alt+F12	<pre>gettestime(set3) + " milliseconds"); System.out.println("Remove element time for tree se</pre>
♣ <b>□</b> 50 <b>₽</b> 4 <u>5</u> 4 <u>5</u>	Local History	÷	
😹 VM Telemetry Overview 🛛 🔁 Output	Tools	•	
	Properties		3 67   2   INS

Choose Profile File to start profiling the application.

5. A dialog box is displayed as shown in Figure 4. You can choose the tasks for Monitor, CPU, or Memory. The Monitor task monitors the application. The CPU task tracks the CPU time used for each method in the application. The Memory task tracks the memory usage for each object in the application. Choose CPU and Profile all classes in the Filter. Click *Run* to start profiling. Click Live Results in the Profiling Results pane to see the time spent on executing each method, as shown in Figure 5.

Profile 2410spring2012: SetListPerformanceTest.java		
Monitor	Analyze Performance	
CPU  Analyze Performance Create Custom  Memory	<ul> <li>Entire application</li> <li>Part of application</li> <li>Filter: Profile only project classes</li> <li>Show filter value Edit filter sets</li> </ul>	
	Use defined Profiling Points       Show active Profiling Points         Overhead:       Advanced settings	
	Run Cancel Help	

The task dialog box enables you to choose a task for profiling.

2410spring2012 - NetBeans IDE 7.0.1						
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help						
👚 🞦 🞴 🍓 🦻 🍊 🛛 <default.co< td=""><td>nfig&gt; 🕞 🍞 🍞 🕨 🎼 - 🕀 -</td><td></td><td></td><td></td></default.co<>	nfig> 🕞 🍞 🍞 🕨 🎼 - 🕀 -					
Projects Files Services :	Start Page 🛛 🚳 SetListPerformanceTest.java 🖇 🐻 Live Pr	ofiling Results 🛛 🔞 🤇	CPU: 09:22:35 PM * 8			
Controls						
D 🗈 🖬 🕙 🔤		C-16 time [0/]	Calf Fran	Taurantina		
💌 Status	Hot spots - Method	Seir unie [‰] ♥	Sell une			
Type: CPU	java.utii.ArrayList.indexof (Object)		18351 ms (34.3%)	27024175		
Configuration: Analyze Performance	java.lang.integer.equals (Object)		2724 ms (5.1%)	2/9341/3		
Status: Running	java.lang.Integer.compareTo (Object)		2724 ms (5.1%) 2510 ms (4.7%)	2024874		
Profiling Results	iava util TreeMan getEntry (Object)	1	1116 ms (3.1%)	100000		
E	iava lang Integer compare (int. int)		1047 ms (2.176)	2024874		
	iava.util.TreeMan.nut (Object, Object)		582 ms (1.1%)	50000		
	iava.util.TreeMap.fixAfterDeletion (iava.util.TreeMap.Entry)	1	567 ms (1.1%)	49998		
Take Snapshot Live Results	java.lang.Integer.valueOf (int)		513 ms (1%)	350739		
Prove Collected Devices	iava.lang.Integer. <init> (int)</init>		510 ms (1%)	350025		
IL-V Reset Collected Results	iava.util.TreeMap.parentOf (iava.util.TreeMap.Entry)		482 ms (0.9%)	681746		
Caved Spanshots	iava.util.TreeMap.fixAfterInsertion (iava.util.TreeMap.Entry)		446 ms (0.8%)	49999		
	java.util.Random.next (int)		419 ms (0.8%)	351477		
🍉 2410spring2012 👻	java.util.AbstractCollection.addAll (java.util.Collection)		316 ms (0.6%)	3		
	java.lang.Number. <init> ()</init>		263 ms (0.5%)	350027		
Open	SetListPerformanceTest.getTestTime (java.util.Collection)		262 ms (0.5%)	4		
Delete	java.util.HashMap.getEntry (Object)		248 ms (0.5%)	100000		
	java.util.concurrent.atomic.AtomicLong.compareAndSet (lon		231 ms (0.4%)	351479		
Save As	java.util.HashMap.removeEntryForKey (Object)		229 ms (0.4%)	100000		
Load	java.util.TreeMap.setColor (java.util.TreeMap.Entry, boolean)		227 ms (0.4%)	276841		
	java.util.ArrayList.set (int, Object)		226 ms (0.4%)	99998		
▼	java.util.Random.nextDouble ()		225 ms (0.4%)	150739		
Navigator	java.util.HashMap.put (Object, Object)		217 ms (0.4%)	100010		
Members View	java.util.HashMap.containsKey (Object)		191 ms (0.4%)	100000		
SetListPerformanceTest	java.util.ArrayList\$Itr. <b>next</b> ()		179 ms (0.3%)	150000		
getRemoveTime(Collection <integer></integer>	java.util.HashSet.add (Object)		176 ms (0.3%)	100010		
<pre>@ getTestTime(Collection <integer> c) :  </integer></pre>	SetListPerformanceTest.getRemoveTime (java.util.Collection)		170 ms (0.3%)	3		
main(String] args)	java.util.HashMap.hash (int)		156 ms (0.3%)	300010		
N:int 🔻	java.lang.Math. <b>random</b> ()		137 ms (0.3%)	150739		
< •	java.util.Collections.shuffle (java.util.List, java.util.Random)		131 ms (0.2%)	1 👻		
	[Method Name Filter]			•		
WM Telemetry Overview 🐻 Output						
2410spring2012 (profile-single) running 🕅 👩 67   2   INS						

The execution time for each method in the application is displayed in the result pane.

 After profiling CPU task is completed, restart profiling by choosing the Memory task, as shown in Figure 6.

Profile 2410spring2012: Exercise25_16.java			
Monitor	Analyze Memory		
СРИ	<ul> <li><u>Record object creation only</u></li> <li>Record <u>b</u>oth object creation and garbage collection</li> </ul>		
Memory     Analyze Memory	Track every 10 - object allocations		
Create Custom	☑ Use defined Profiling Points Show active Profiling Points		
	Overhead: Advanced se	ttings	
	Run Cancel	elp	

You can choose Memory to profile memory usage.



The Memory profiler tracks the memory usage.

7. Click Live Results in the Profiling Results window to see the memory usage in the result pane. You can also see the heap size and garbage collection in the VM Telemetry Overview window, as shown in Figure 7.