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*.

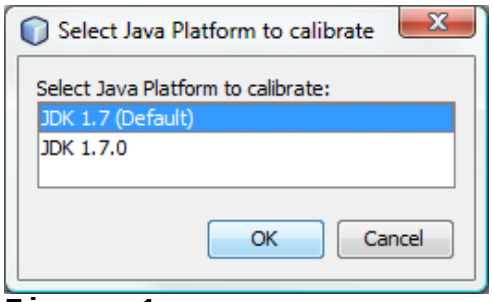

Figure 1

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

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.

2. Create a class named SetListPerformanceTest in the project.

3. Copy and paste the code in Listing 23.6 SetListPerformanceTest.java for the class in NetBeans, as shown in Figure 2.

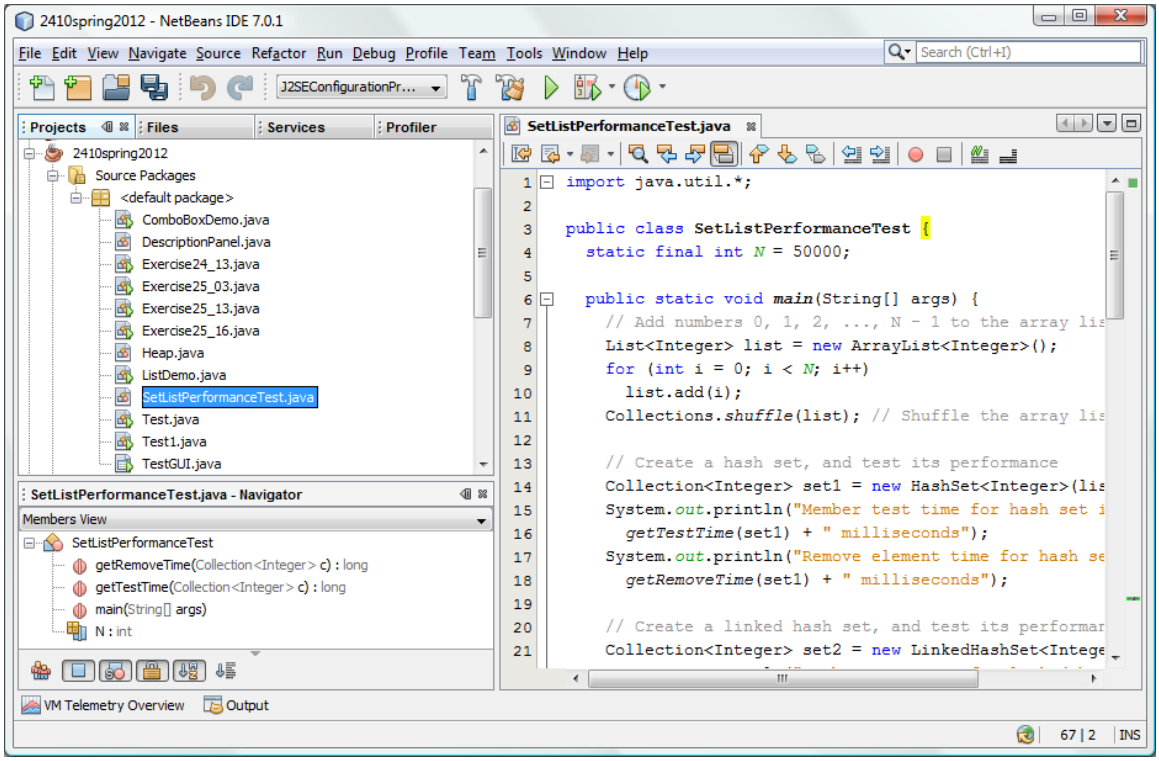

Figure 2

The class SetListPerformanceTest is created in the project.

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

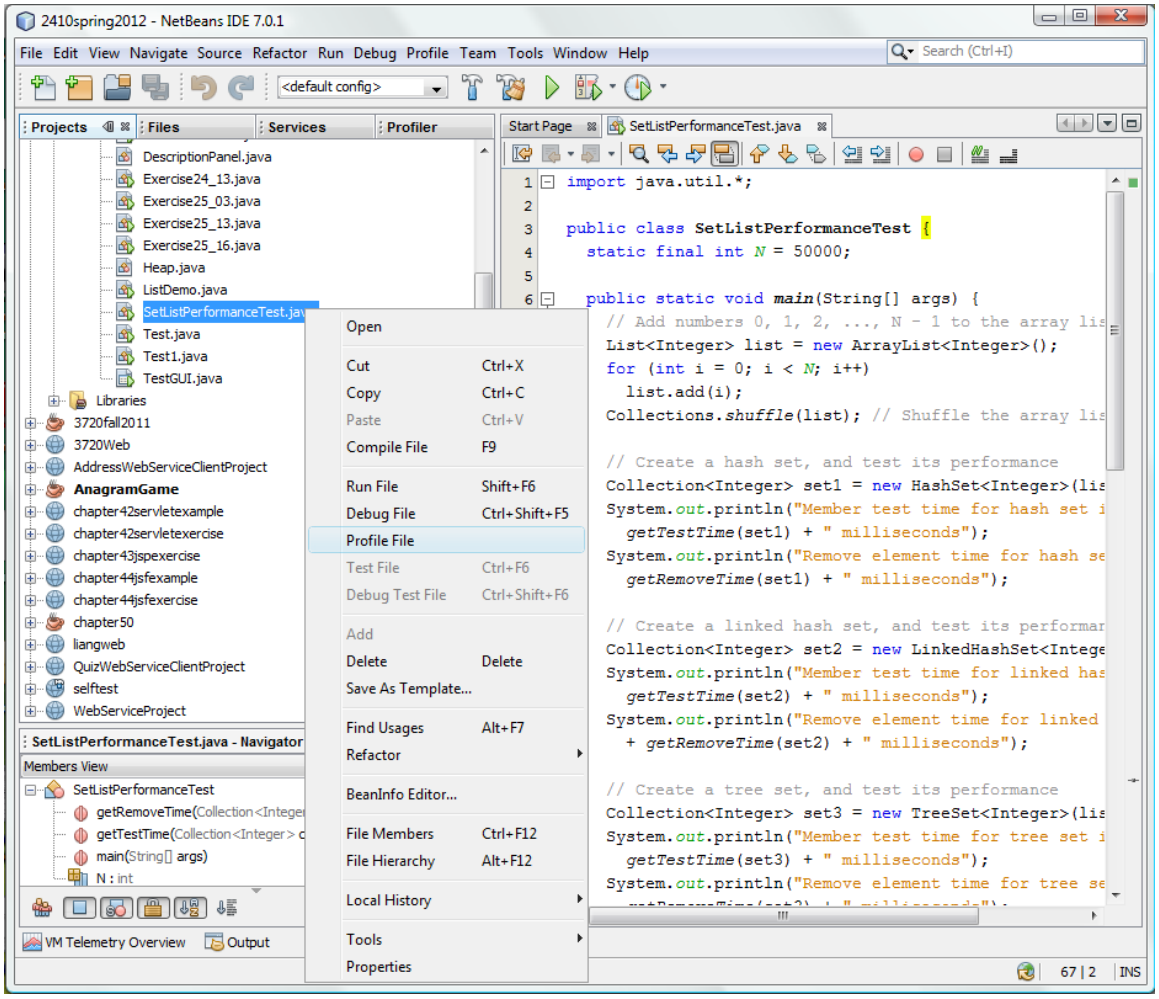

Figure 3

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.

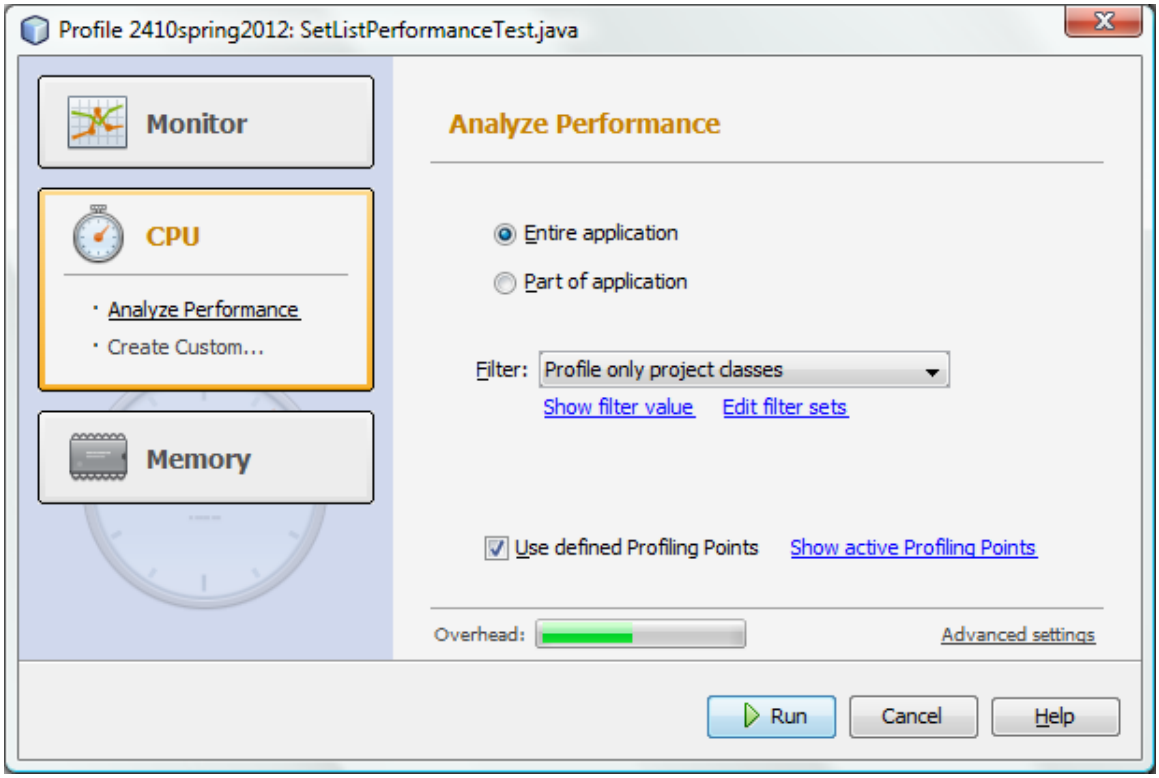

Figure 4

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

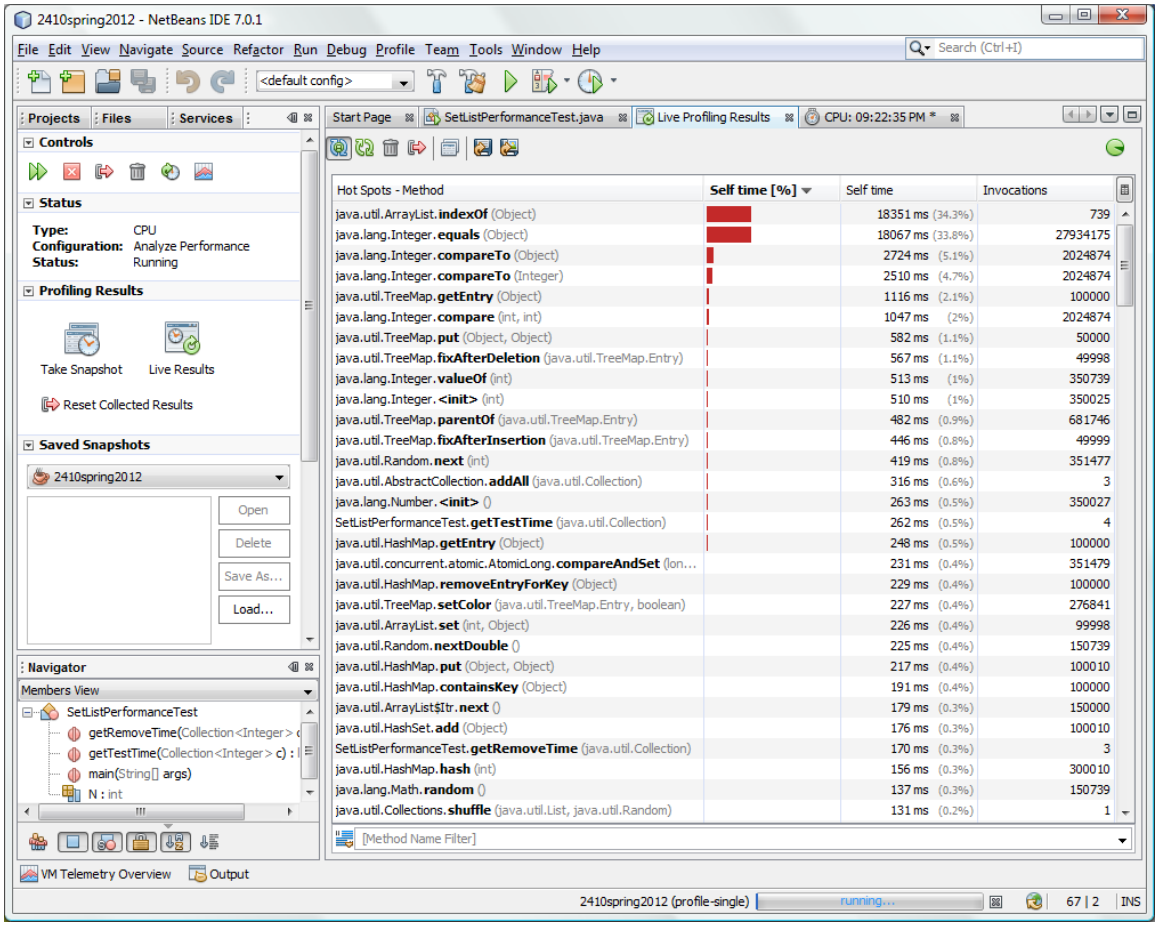

Figure 5

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

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

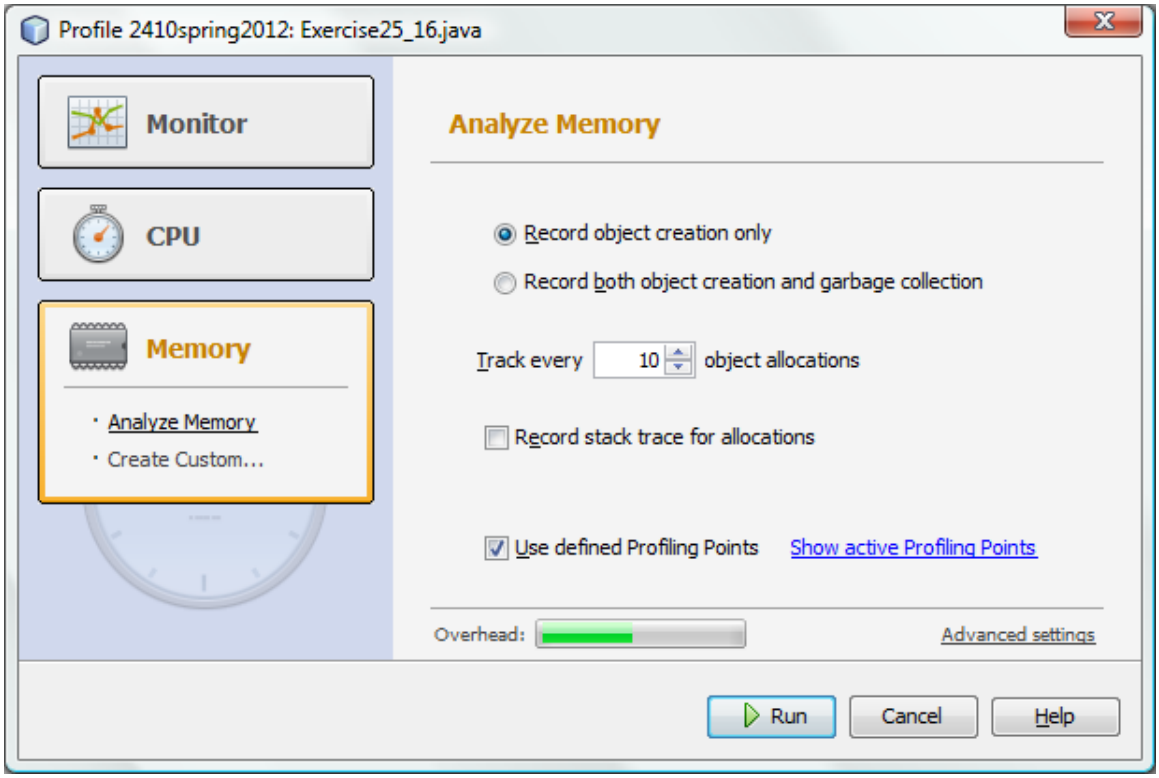

Figure 6

You can choose Memory to profile memory usage.

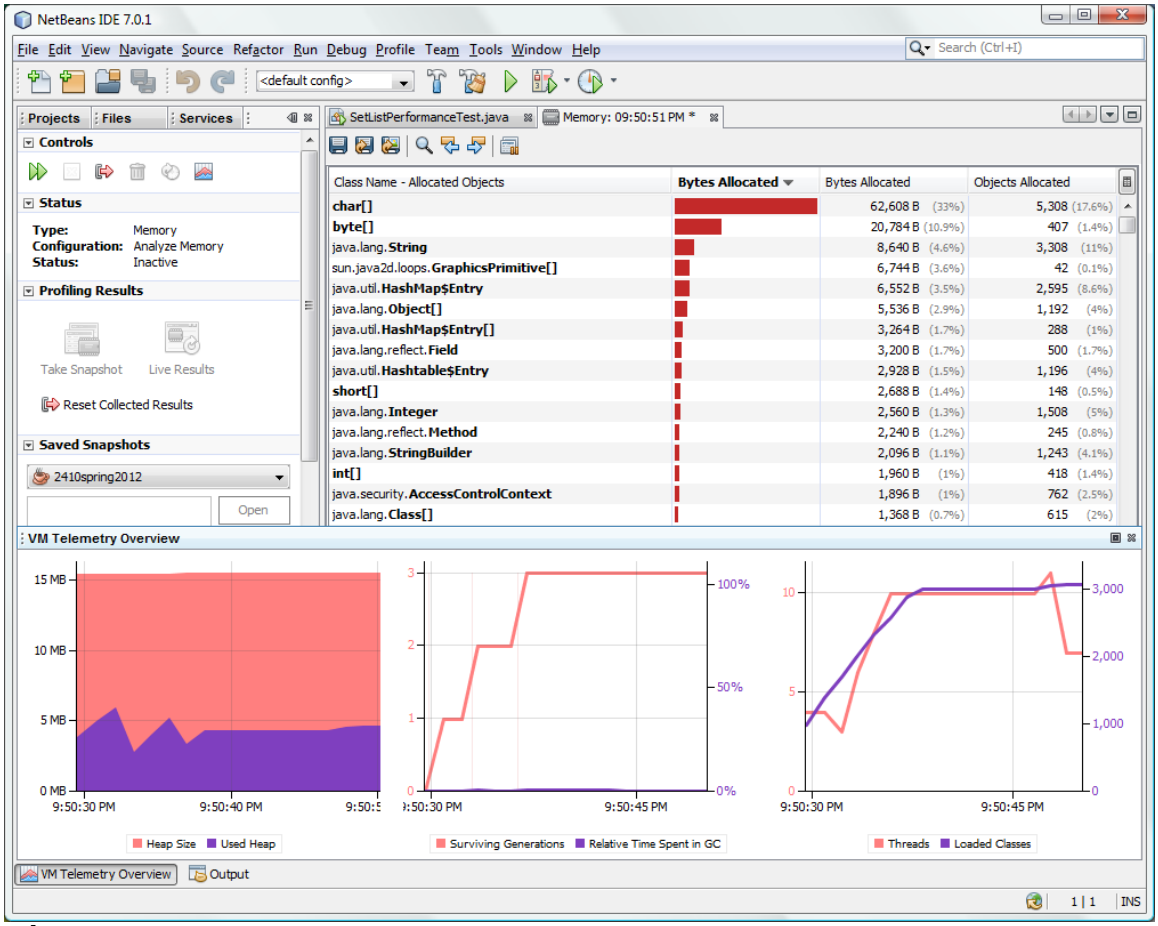

Figure 7

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.

