Java Flight Recorder

What is it?

Java flight recorder is a profiling tool packaged with the Oracle jvm for free since java 7. Profiling in general is used in large applications to identify and rectify performance bottlenecks. If necessary, you can then make changes to the slower parts of your code as you see fit. Java flight recorder can be used to profile code to recognize hot methods, that is methods that run for the most amount of time during the run time of an application and memory allocation among other statistics. In this assignment you are expected to use this tool to analyze the **"hot methods"** in the sample program that you write for analyzing the performance of the Tree Map and your custom hash table. The most efficient implementation should ideally run for a smaller amount of time or rather less hot.

How to use it?

By default the flight recorder is not enabled. So, certain flags must be set to enable it. There are two kinds of recordings that are performed usually - timed and continuous. Timed recording is used when your application is long running and you want to profile it for a fixed amount of time. We will be using continuous recording to basically record events for the entire duration of our sample application. Here is a step by step guide to use the flight recorder and analyze the results after.

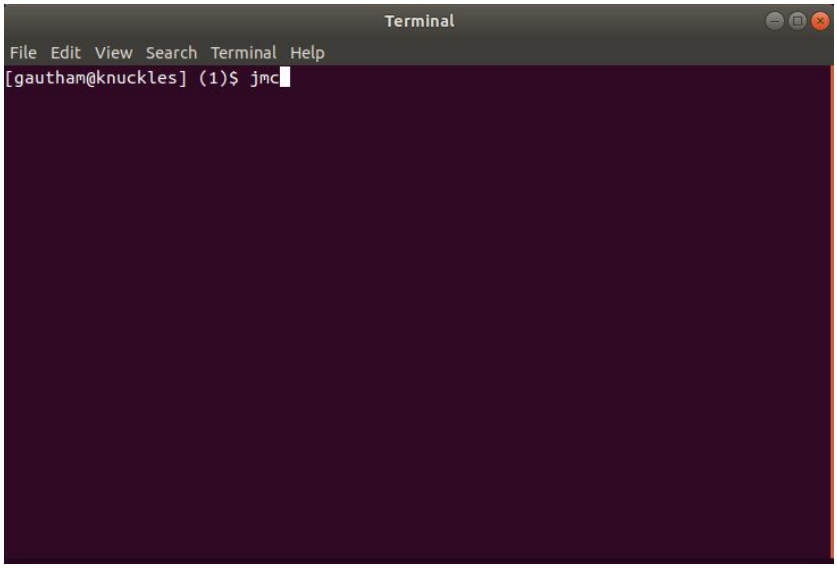
1. **Compilation of the profiling code:** Just regular compilation using *javac*

2. **Running the code:** As mentioned earlier you need to specify a couple of flags in order to activate the flight recorder. They are : **`-XX:+UnlockCommercialFeatures`** and **`-XX:+FlightRecorder`**
   To run a continuous recording, you will need to execute your profiling application as specified below. (when copying the command, please ensure it is in one line on the terminal)
   ```
   java -XX:+UnlockCommercialFeatures -XX:+FlightRecorder
   -XX:FlightRecorderOptions=defaultrecording=true,dumponexit=true,d
   umponexitpath=./recordings/profile.jfr,settings=profile
   ProfileSample 10000000
   ```

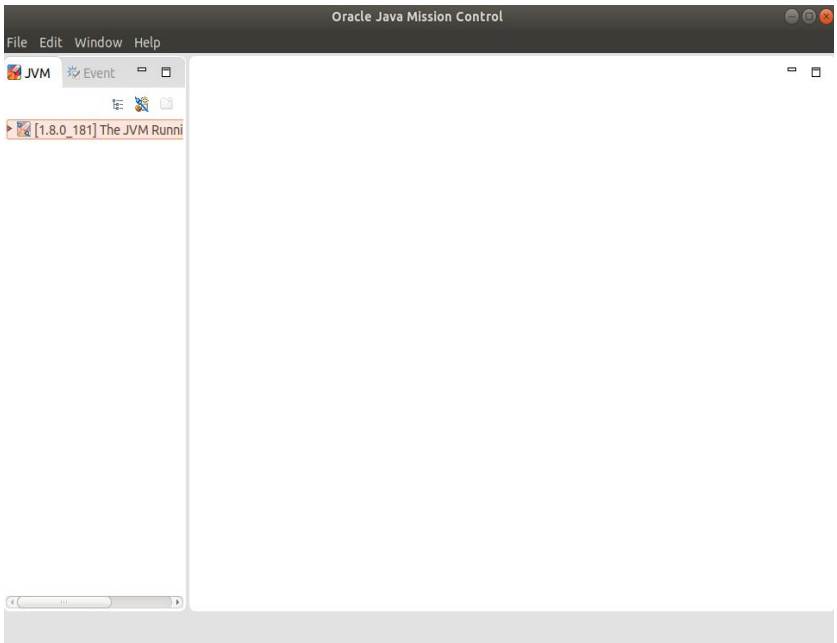
   This command will execute ProfileSample and simultaneously start recording events to identify hot methods, memory allocation etc. Continuous recording by default doesn't generate a recording file, the option to do so is specified using **`dumponexit=true`**. The recorded file is written to the path specified using **`dumponexitpath.`** The **`profile`**

settings used is available by default in flight recorder. It records all events except object statistics as they tend to start affecting the performance of the application.
Note: The ProfileSample code inserts n elements(argument passed from the command line, viz 10000000 in this case) in to a hash set. You need to make your profiling code slightly long running to give the flight recorder enough time to capture events. It records them once every 10ms.

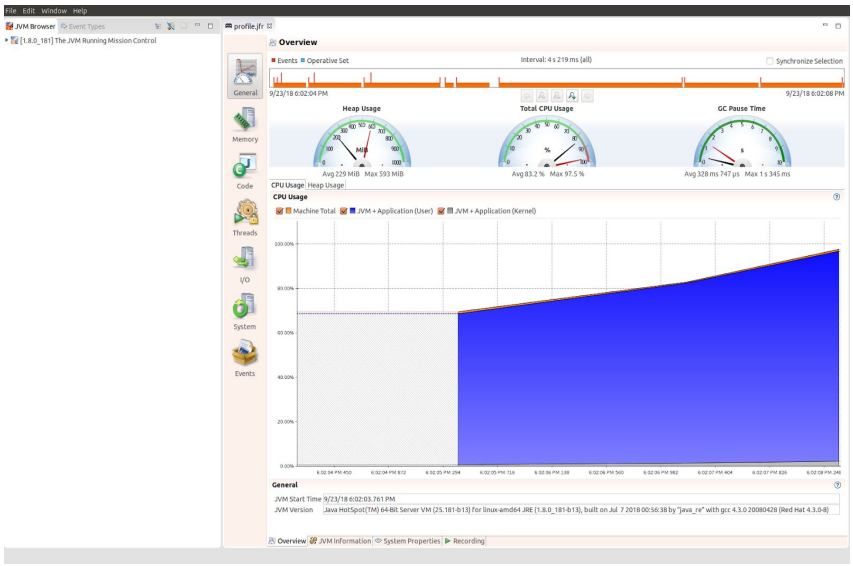
3. **Analysis of the recording:** The recorded file with the extension jfr can be analyzed using the java mission control (JMC) that is already installed in all of the lab machines. You can start the application by typing **`jmc`** and hitting enter on the command line**.**



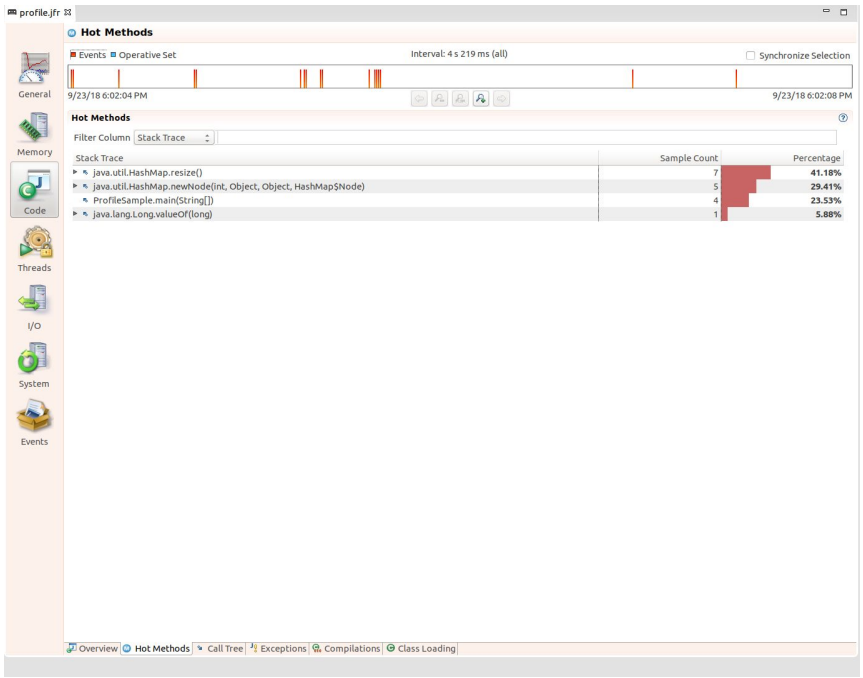
On hitting enter, the JMC ui will show up.



JMC shows all the JVM's currently running on the system. As can be seen above, there is some java application currently running, which in this case is the JMC itself.
The recording we captured previously was saved in the recordings folder in the current directory. Using the file option in the UI you can open it. You should see something like this:



If you click on code, you will be able to see the hot methods.



The resize method of the HashMap which is used internally in a HashSet was running for the most amount of time in the ProfileSample code.

Note:

1. To analyze the heap statistics, that is dynamic storage allocated during the execution of the program, you will be provided with a **heap_stats.jfc** file which has the heap statistics configuration on. So, instead of providing **settings=profile** in the java command in step 2, you should provide the path to the **heap_stats.jfc** file. The heap stats can be analyzed under the memory tab. The rest of the steps remain the same.

2. For Windows and Mac users, you need to ensure you are using the Oracle JDK and not the OpenJDK. If using the Oracle JDK, the process described above should work on both Windows and Macintosh. I have tried it on my Mac and it works!

3. As mentioned in the earlier write up, you need to run Java 7+ to use the flight recorder as mentioned above.

4. For more information on java flight recorder you can refer to https://www.oracle.com/technetwork/java/javaseproducts/mission-control/java-mission-control-1998576.html