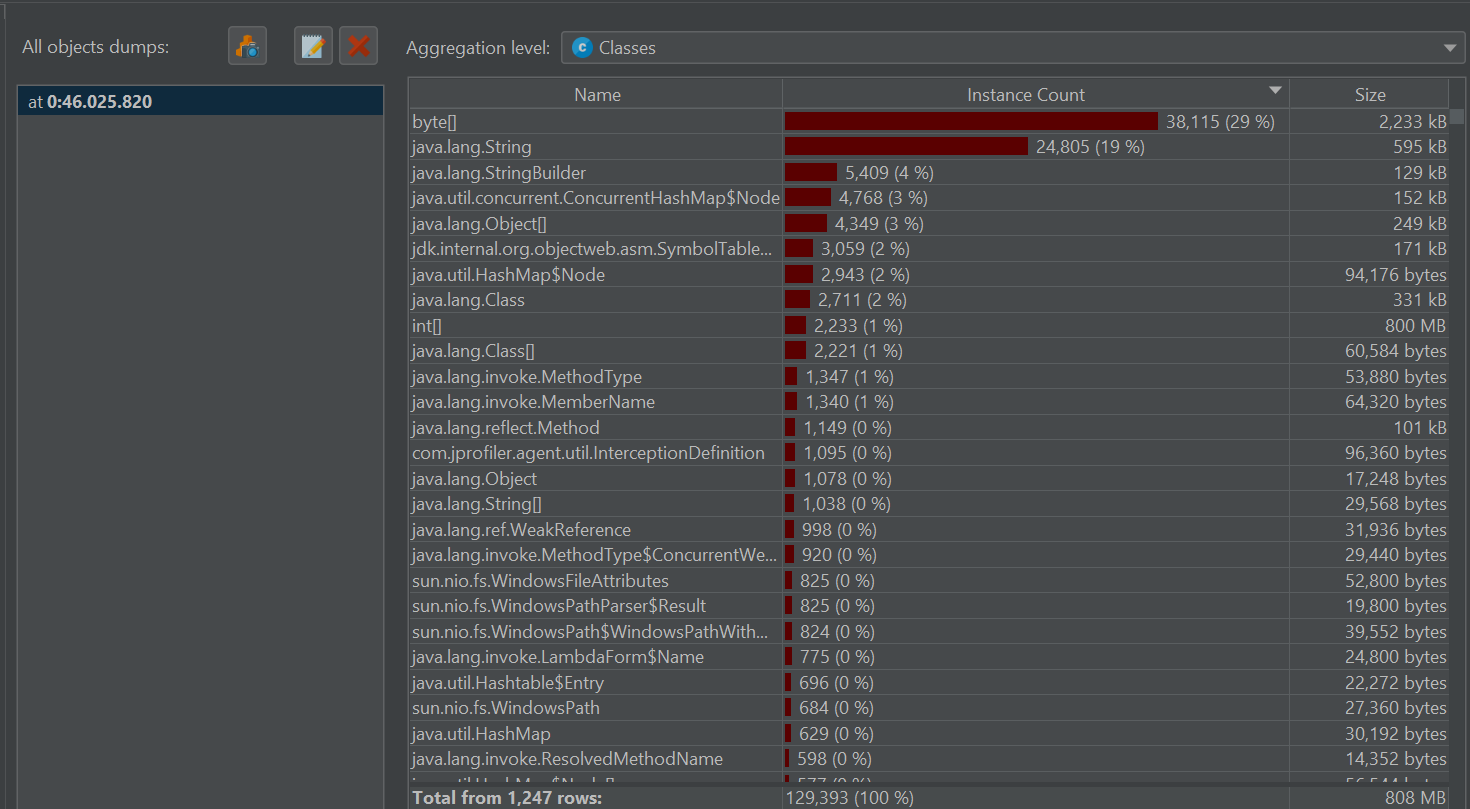
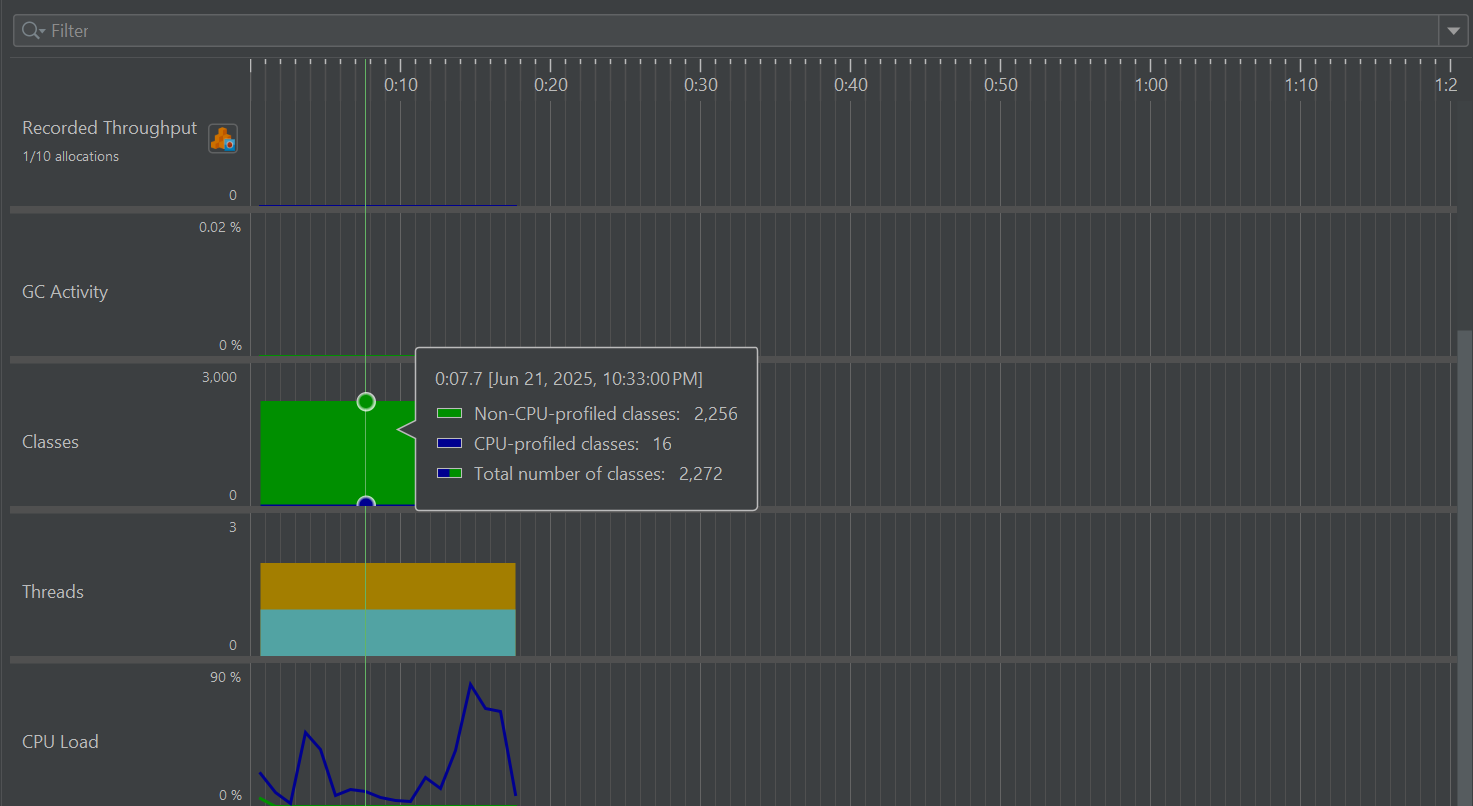
Heap Dump

Generator Code –

package HeapThreadDump;  
  
import com.sun.management.HotSpotDiagnosticMXBean;  
  
import java.io.File;  
import java.lang.management.ManagementFactory;  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
  
public class HeapDumpGenerator {  
  
 // JVM MXBean name for heap dump  
 private static final String *HOTSPOT\_BEAN\_NAME* = "com.sun.management:type=HotSpotDiagnostic";  
  
 private static HotSpotDiagnosticMXBean *hotspotMBean*;  
  
 public static void dumpHeap(String filePath, boolean live) throws Exception {  
 int a[] = new int[200000080];  
 if (*hotspotMBean* == null) {  
 *hotspotMBean* = ManagementFactory.*newPlatformMXBeanProxy*(  
 ManagementFactory.*getPlatformMBeanServer*(),  
 *HOTSPOT\_BEAN\_NAME*,  
 HotSpotDiagnosticMXBean.class  
 );  
 }  
  
 File file = new File(filePath);  
 if (file.exists()) {  
 System.*out*.println("File already exists, deleting: " + filePath);  
 if (!file.delete()) {  
 throw new RuntimeException("Unable to delete existing heap dump file.");  
 }  
 }  
  
 *hotspotMBean*.dumpHeap(filePath, live);  
 System.*out*.println("✅ Heap dump created at: " + filePath);  
 }  
  
 public static void main(String[] args) {  
 try {  
 String timestamp = LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*("yyyy-MM-dd\_HH-mm-ss"));  
 String filePath = "heapdump\_" + timestamp + ".hprof";  
  
 System.*out*.println("🔁 Generating heap dump...");  
 *dumpHeap*(filePath, true);  
  
 // 🔒 Keep the process alive for 2 minutes  
 System.*out*.println("⏳ Waiting so you can inspect the process (PID)... Press Ctrl+C to exit early.");  
 // 2 minutes  
 } catch (Exception e) {  
 System.*err*.println("❌ Failed to generate heap dump:");  
 e.printStackTrace();  
 }  
 }  
}

Outputs –





Thread Dump

Generator Code –

package HeapThreadDump;  
  
import java.io.FileWriter;  
import java.io.IOException;  
import java.lang.management.ManagementFactory;  
import java.lang.management.ThreadInfo;  
import java.lang.management.ThreadMXBean;  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
  
public class ThreadDumpGenerator {  
  
 public static void generateThreadDump(String filePath) throws IOException {  
 ThreadMXBean threadMXBean = ManagementFactory.*getThreadMXBean*();  
  
 // Get all thread info including monitors and synchronizers  
 ThreadInfo[] threadInfos = threadMXBean.dumpAllThreads(true, true);  
  
 try (FileWriter writer = new FileWriter(filePath)) {  
 for (ThreadInfo ti : threadInfos) {  
 writer.write(ti.toString());  
 for (StackTraceElement ste : ti.getStackTrace()) {  
 writer.write("\tat " + ste + System.*lineSeparator*());  
 }  
 writer.write(System.*lineSeparator*());  
 }  
 }  
  
 System.*out*.println("✅ Thread dump written to: " + filePath);  
 }  
  
 public static void main(String[] args) {  
 try {  
 // Generate timestamped filename  
 String timestamp = LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*("yyyy-MM-dd\_HH-mm-ss"));  
 String filePath = "threaddump\_" + timestamp + ".txt";  
  
 System.*out*.println("🔁 Generating thread dump...");  
 *generateThreadDump*(filePath);  
  
 // Optional: wait to keep process alive  
 System.*out*.println("⏳ Sleeping to inspect thread state...");  
 Thread.*sleep*(120000); // Keep alive for 2 minutes  
  
 } catch (Exception e) {  
 System.*err*.println("❌ Failed to generate thread dump:");  
 e.printStackTrace();  
 }  
 }  
}

Output –

