**Task 5**

**Problem:** *Implement java application that reads the attached text data file line by line. The application should take 3 first characters (using substring() method) of every line and put all the 3-character lines to ArrayList.*

*Analyze memory consumption.  
Collect heap dump (when the file has been read and ArrayList is fully populated but the application is still running).  
Identify memory leak if any.  
Fix memory leak if any.*

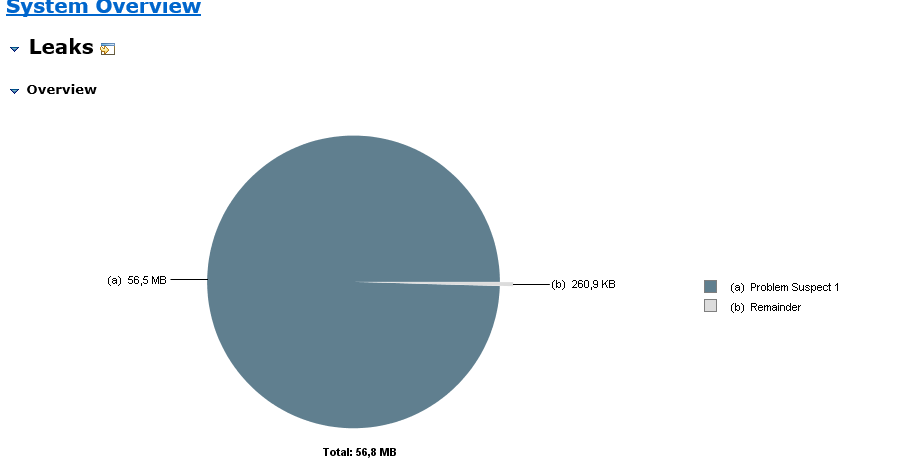
*Results of the task:  
Java source code (with/without memory leak)  
Leak suspect report (from Eclipse MAT)  
Your findings about the memory problem*

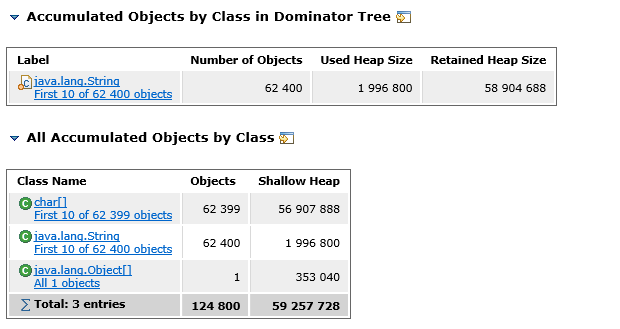
Based on the article:

<https://stackoverflow.com/questions/15612157/substring-method-in-string-class-causes-memory-leak>

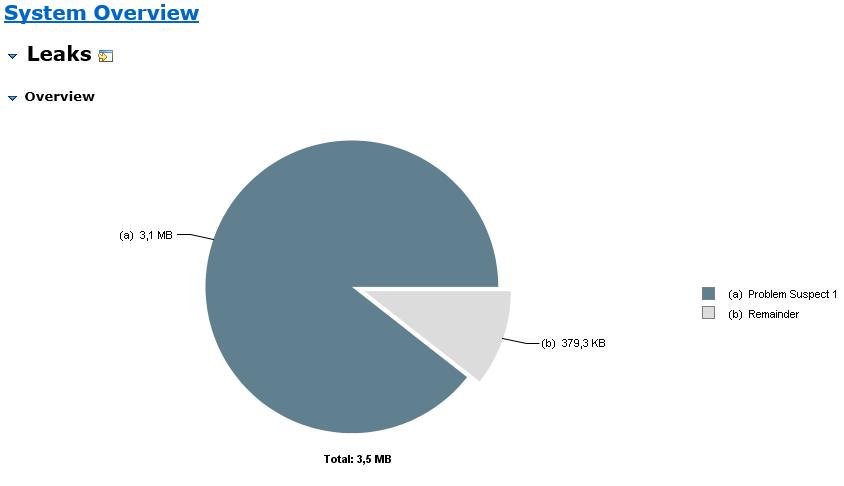
The memory leak problem with substring occurs only in Java 6 and is solved in java 8. So, basically the problem is reproducible if application is compiled using jdk 6 and is gone when compiled using jdk 8. The problem in jdk 6 was that substring(0,3) took and copied the whole string not just the first two characters, so the memory was leaked. The problem could is solved in later versions of java, only the first three characters are copied, thus using much less memory.

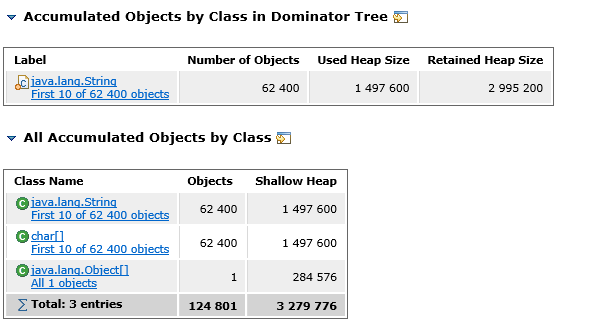
**JDK6**





**JDK8**





**Summary:** As you can see same code was used and text file was parsed but the memory used in jdk 6 is ~**59 mb** but in jdk 8 it was **~3 mb**

**Task 1 (Live locks, deadlocks, bottlenecks)**

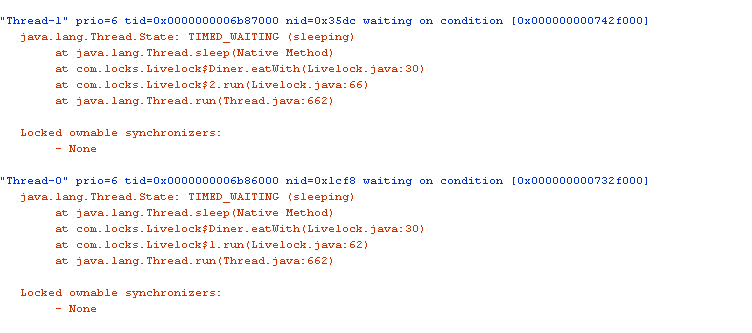
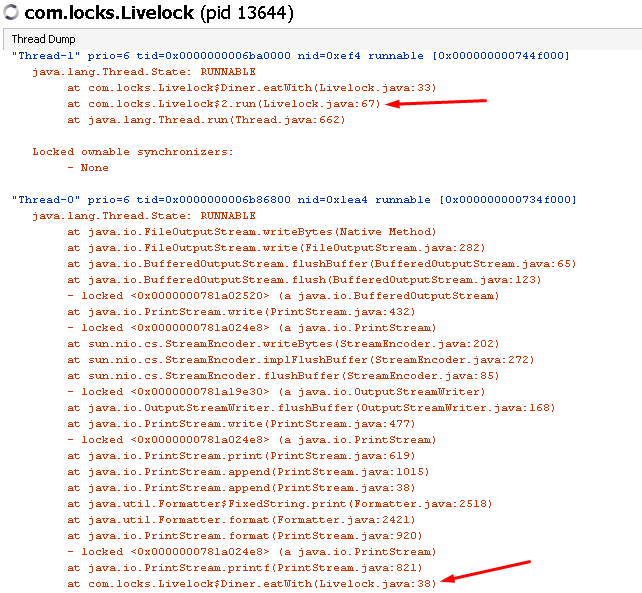
**Problem 1:** Implement java applications to reproduce deadlock, live lock and bottleneck cases. Collect thread dumps and analyze them. Results of the task: Java source code + notes with analysis details.

**Deadlock**

The full threaddump is in ‘deadlock\_threaddump’. As expected analysis showed two threads in status blocked. Each thread is waiting to lock object locked by another thread.

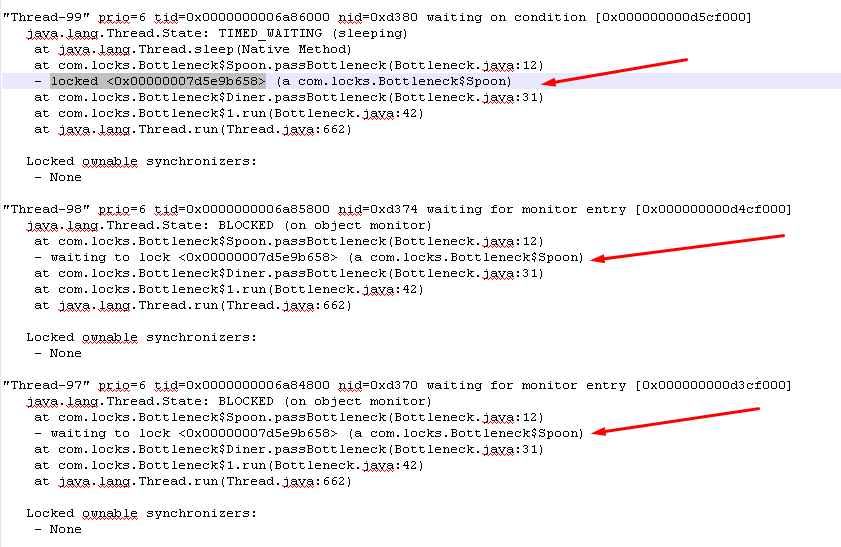


**Livelock**

The full threaddump is in ‘livelock\_threaddump’. As expected analysis showed that both threads that cause livelock are is status ‘running’ not ‘blocked’. Even though both threads are running no work is done and the program is stuck

**Bottleneck**

I created 100 threads and made them all pass through synchronized method ‘passBottleneck’ where the thread is stuck for 10 seconds. The threaddump ‘bottleneck\_threaddump’ demonstrates the problem, while one thread owns the lock (<0x00000007d5e9b658>) and is asleep other threads are ‘blocked’ and waiting for this lock (there are 98 blocked threads in the dump)



**Problem 2:** Further discussion item. Customer says that the app fails with OutOfMemory error on production server. What will you do to investigate and solve the issue?

1.I will add -XX:+HeapDumpOnOutOfMemoryError flag to server startup and get heap dump on next failure

2. Will analyze memory dump with MAT and find out which object take the most memory

3. Knowing the object that uses up memory, will find all places in code that could have caused the memory leak and will try to fix the problem. If there is no memory leak I will request more memory for the production server.