OOM Lab Part 1 - SimpleOOM

Locate the SHAREBostonOOM folder on the C drive

WordPad or Notepad will be used to open several of the files.

You will also need to open a Command Prompt window (ie, DOS prompt)

In the SHAREBostonOOM folder, open the file named
Simplejavacore.20100727....txt
and answer the following questions:

1. What is the exception thrown? What field name contains the exception?

2. What is the zOS release in use?

3. What is the java version in use?

4. What java 'command' was executed?

5. What is the Application or Class name in use?

6. Locate the MEMINFO section

7. How much heap is available? How much is allocated?
What can you determine based on these numbers?

8. Find the 'Current Thread Details' and review the java stack

9. What source code line number was last executed when the exception occurred?

10. Is the method JIT (Just in Time) compiled? What might that mean?

NEXT, Open the file named SimpleGCOut using Notepad or WordPad

11. What is this output? And how is it captured?

(Ok, I'll help you here: if you've not seen this before,
this is a Garbage Collection trace, captured by passing -verbosegc
when java is initialized)

The key gc type is the 'Global' entry, this is what performs the actual garbage collection

12. How frequently are Global GC's occurring?
(note the 'intervalms' field)

13. Does this seem excessive?

Lastly, go to the bottom of the file, and note the exception is captured.

Given the simplicity of this example, you may be able to review the GC trace activity easily. But, what if this were data collected over a long period of time? We'll answer this in a moment...

BUT, what if -verbosegc wasn't in use? Let's move to the Snap trc files...

Snap trc files contain raw JVM trace data. It is formatted using the command (using an IBM JVM)

java com.ibm.jvm.format.TraceFormat <Snap....trc>

The default output file will be named with a suffix of '.fmt'

appended to the Snap trace input filename

So, let's look at the the file named SimpleSnap.20100727...trc.fmt using WordPad or Notepad. Scroll to the bottom of the file.
The key entry to look for is J9Allocateobject()

14. How large of an object was requested but failed?

Next, we'll try using PMAT (Pattern Modeling and Analysis Tool)
From a DOS window,
  cd SHAREBostonOOM
and enter the following at the prompt:

 java -Xmx900m -jar ga401.jar SimpleGCOut

Not the GCStats, etc...
Move your cursor to the top of the tool bar, and select the option
  "Graph View All"

15. Notice the vertical bar to the right? What is that displayed?
  (hint:recall the Gc's occured milliseconds apart)

****
OOM Lab Part 2 - StartThreads

try answering the same questions from Lab 1. All the files for Lab 2 will begin with 'Start', ie, Startjavacore...., StartSnap..   etc.

When reviewing the javacore, take note of any differences, especially in the 'Current Thread Details'

16. What does Native Method mean?

Review the ENVINFO

17. is there anything there that warrants further investigation?

Review the gc tracing in StartGCOut

?????????

PMAT won't help.....

Review StartSnap....trc.fmt

?????????

If you didn't notice, return to Startjavacore...txt , and review the exception..

This error is a NATIVE OOM error, requiring a zOS tdump to analyze further.
Capturing a tdump requires use of :

-Xdump:system:events=systhrow,filter=java/lang/OutOfMemoryError