

|  |  |
| --- | --- |
| Employee Name: | MANOJ KUMAR YEKOLLU |
| Training Supervised by: | **KRISHNA BODDULURI** |
| Training Task #: | **ASSIGNMENT08012019** |
| Training Resource Materials: | **Dzone, Github, javacodegeeks, baledung** |
| Training Task Date: | **08/01/2019** |
| Task Due Date: | **08/15/2019** |
| Task Submitted Date: | **08/15/2019** |
| Github link: | **https://github.com/NightFury546/JavaHeapLoader.git** |
| Technologies used for Training | **Java, Memory Management, Eclipse, Eclipse Memory Analyzer.** |

**Task Description/Requirement:**

1.        Create Heap Dump using java program

2.        Using Heap Dump analyzer, find out what exactly cause the heap dump

**High Level Synopsis:**

**Overview**

Identifying reason for the out of memory error of java applications with larger heap size may be one of the nightmares of a developer, because most of the out of memory situations may not be identified during the testing phase. It may occur only in production after running for a long time. The purpose of this article is to explain the use of Heap analyser tool to identify the memory leakage of larger enterprise java applications, which uses larger size of heaps. Before going to the details we will discuss about following points.

* Heap
* Garbage Collection
* Out of memory

**What is heap?**

The space used by the Java Runtime to allocate memory to Objects and JRE Classes is called Heap. The heap space can be configured using the following JVM arguments.

o   -Xmx<size> - Setting maximum Java heap size

o   -Xms<size> - Setting initial Java heap size

The maximum heap size can be configured in a 32 bit JVM is 2GB. If any application requires more than 2 GB, it should run on 64 bit JVM. 64MB is the maximum heap size by default.

**Garbage Collection**

One of the advantages of Java over C++ is automatic memory management. In C++ memory can be released manually, but it will happen automatically in Java using a process call Garbage collection. Garbage collection will free up the memory of the object that doesn’t have any reference. That is will destroy the unused objects. Garbage collection process can be tune for different applications based on the object creation characteristics of the application. This can be achieved through number of JVM arguments. Following are the few JVM arguments which can be used to tune the garbage collection process.

|  |  |
| --- | --- |
| **Option** | **Description** |
| -XX:-UseParallelGC | Use parallel garbage collection for scavenges. |
| -XX:-UseParallelOldGC | Use parallel garbage collection for the full collections. Enabling this option automatically sets -XX:+UseParallelGC. |
| -XX:NewRatio | Ratio of old/new generation sizes. The default value is 2. |
| -XX:SurvivorRatio | Ratio of eden/survivor space size. The default value is 8. |
| -XX:ParallelGCThreads | Sets the number of threads used during parallel phases of the garbage collectors. The default value varies with the platform on which the JVM is running |

GC Execution has direct relationship with the size of the Heap.

* Larger Heap size will increases the GC execution time but decreases the number of GC executions.
* Smaller Heap Size will increases the number of GC executions but decreases the GC execution time

**Out of memory**

The **java.lang.OutOfMemoryError** will occur when the application tries to add more objects into the heap and there is no space left. This will happen when the maximum heap size set in the start of application is filled with the objects and garbage collector is not able free up the memory because the all objects in heap still have some references. This may happen because of two reasons.

* The application may need more memory to run; because of the currently allocated heap size is not enough to accommodate the objects generated during the run time.
* The other reason may be due to the coding error the application is keeping the references of unwanted objects

The solution for the first reason is, increase the heap size. But the solution for the second is analyzing the code flow and heap dump to identify the unwanted objects in heap. To analyse the application heap we need to take the heap dump and open the same in memory analyzing tool.

In this article we will discuss on how to take heap dump of an application running on **Oracle Java** and analyse the same **in Eclipse Memory Analyzer**.

**How to take Heap Dump**

Heap dump can be take in two ways,

* JVM argument can be added to generate heap dump whenever an OutOfMemoryError occurs

-XX:+HeapDumpOnOutOfMemoryError option can be added to generate a heap dump on OutOfMemoryError. By default the heap dump is created in a file called java\_pid pid .hprof in the working directory of the VM. But we can set alternative path using the JVM option -XX:HeapDumpPath=path

* Using jmap tool available with JDK. Following command can be executed from the command line.

             jmap -dump:format=b,file=heap.bin <pid>

                   <pid> can be replaced with process id of the application

**Eclipse Memory Analyzer**

Eclipse Memory Analyzer can download from the following location.

<https://www.eclipse.org/mat/downloads.php>

Unzip the downloaded file and double click ‘MemoryAnalyzer’ to start the application.

Execute a java program which is continuously running.

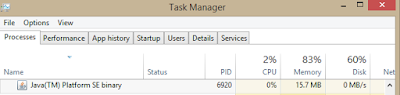
[https://2.bp.blogspot.com/-ctFTJ5OoBC8/W108humx1JI/AAAAAAAACA8/nIZ_HNexmREtZH3dU43BA56oF2ZbhPrCwCLcBGAs/s400/image1.png](https://2.bp.blogspot.com/-ctFTJ5OoBC8/W108humx1JI/AAAAAAAACA8/nIZ_HNexmREtZH3dU43BA56oF2ZbhPrCwCLcBGAs/s1600/image1.png)

Take heap dump using jmap

***jmap -dump:format=b,file=heap.bin 6920***

[https://1.bp.blogspot.com/-7uikQPymX1s/W108ltP5P3I/AAAAAAAACBE/NP252yPoeP4x4HWpe11qSJmwAQhbwniawCEwYBhgL/s400/image2.png](https://1.bp.blogspot.com/-7uikQPymX1s/W108ltP5P3I/AAAAAAAACBE/NP252yPoeP4x4HWpe11qSJmwAQhbwniawCEwYBhgL/s1600/image2.png)

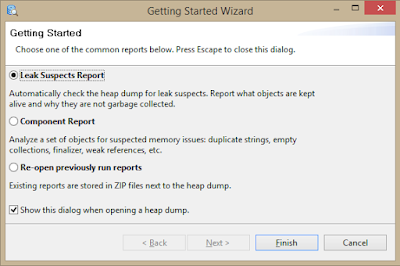
Identify the process id of running application in Windows from Task Manager

[](https://1.bp.blogspot.com/-3ibVqE7yvSw/W108mGLnQEI/AAAAAAAACBo/rM-NQ7vcmQAzFZ7f1jwCYBUXLqmeZwehwCEwYBhgL/s1600/image3.png)

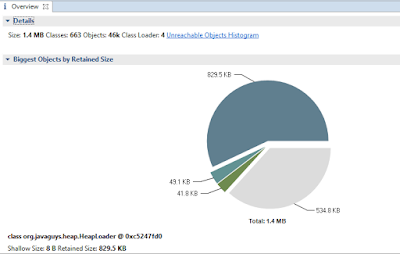
In Linux to identify the process id use ***‘ps –ef | grep java’***

Open the heap dump in Eclipse Memory Analyzer using the option **File Open Heap Dump**

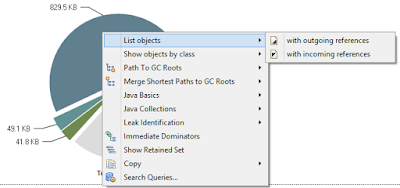
First it will prompt to create a leak suspect report. User can create or skip the same.

[](https://3.bp.blogspot.com/-e3c1qvqBw3I/W108m9DOqiI/AAAAAAAACBw/Edvqt7XYHXciXm2dtAvwnG5KKq-xbXs2ACEwYBhgL/s1600/image4.png)

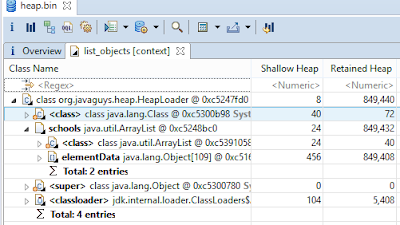
In overview tab of the memory analyser will show the total size of the heap and it will show a pie chart of object wise size.

[](https://1.bp.blogspot.com/-nn_-n_TxLVg/W108nW6Qt0I/AAAAAAAACBw/njM388Q0ML0aLrZMSx412aI5IsjD2U61ACEwYBhgL/s1600/image5.png)

Click on the highest value in the pie chart and select **List Objects  with outgoing references**

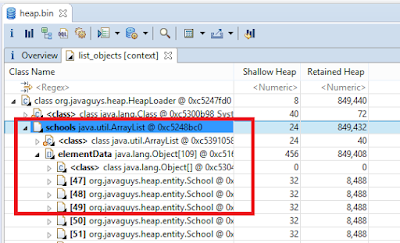
[](https://3.bp.blogspot.com/-rLMJoaI7Az0/W108nl_MMyI/AAAAAAAACBs/3nX6FVPv_Tsi-YELZplc4iqoA7d2oLLVQCEwYBhgL/s1600/image6.png)

It will open a new tab **‘List Objects’**and expand the tree structure of the object.

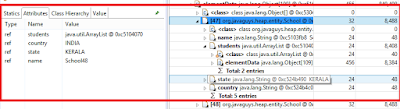
[](https://3.bp.blogspot.com/-klZV0dylK8A/W108nug3mgI/AAAAAAAACBw/NVN6XGKX-GMMzjJqsFYLgGUBBKyA40tWwCEwYBhgL/s1600/image7.png)

User can expand and find the objects available in heap. In this current example the HeapLoader object consist of a list of School object and school object consist of a list of students.

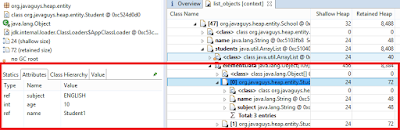
***School List in HeapLoader***

[](https://4.bp.blogspot.com/-YmCyhtBkd8s/W108n-olw4I/AAAAAAAACB0/oupjcKXNE-k9Fn4Y3GBp1TFmMwyOHEORwCEwYBhgL/s1600/image8.png)

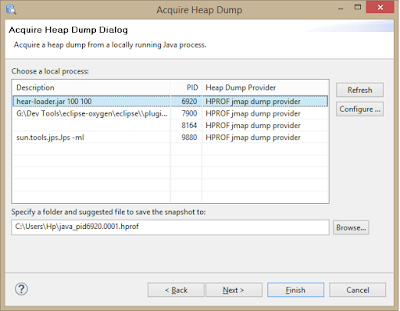
***Each school object in List and its attributes***

[](https://4.bp.blogspot.com/-OxOejX0CQac/W108oA1ZgqI/AAAAAAAACB4/E9UiolDVd_s3oMLZ5MXNilKOc60xKqH5QCEwYBhgL/s1600/image9.png)

***Student object in school and its attribute***

[](https://1.bp.blogspot.com/-hFPuYNBT6bs/W108la6mz_I/AAAAAAAACB0/a4cbFgr7q-AMGLzeuvEgh0-zZOxDa6UsgCEwYBhgL/s1600/image10.png)

There is an option available in Eclipse Memory Analyser to Acquire Heap Dump, if the application is running on same machine. ***Select File  Acquire Heap Dump*** option will show the all Java applications available in machine. Select the process, browse the path where the heap want to save and click finish button.

[](https://1.bp.blogspot.com/-cEYJhBd5gu0/W108lmHEUxI/AAAAAAAACBo/9Wd_-3cGKdMgf764K69_9V0ySW1IQvP4QCEwYBhgL/s1600/image11.png)

These are the basics steps to analyse a heap dump. There are more options available in Memory Analyser, which will give more insights into the heap dump. And Eclipse Heap Analyser can be used for the dumps generated from the application running on Oracle Java. **To analyse the heap dump running on IBM java can be downloaded from**

<ftp://public.dhe.ibm.com/software/websphere/appserv/support/tools/HeapAnalyzer/ha456.jar>

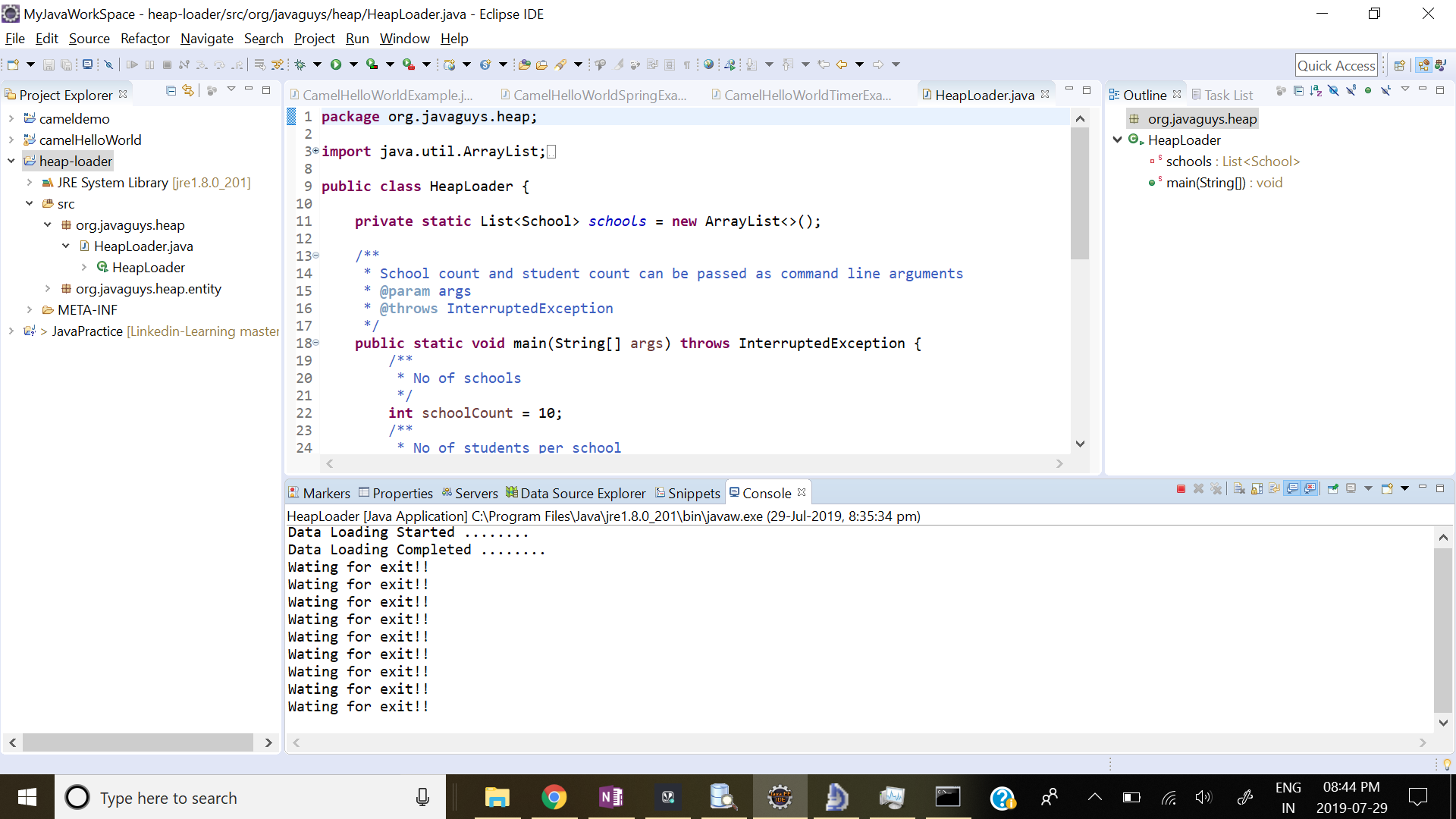
**Sample application**

The sample application used in the above example consists of two objects School and Student.

School will have a list of Students. The application has a list in the main class which will load schools with students. And the application will continue in an infinite loop. The number of schools and students can be given as input from command line

**I acknowledge that this document can be supplied to USCIS in compliance with CPT/OPT/STEM OPT audit:**

**Output:**



Heap Dump analysis using Eclipse Memory Analyzer

