Become A Guru

How To Solve A Memory Leak In Under 10 Minutes

Confidential - do not distribute

Hotels.com™

What You Will Learn

- A methodology for approaching memory leaks
- Understanding the generational heap
- Understanding generational aging to find leaks
- Using various tools to identify and analyze leaks
- A step-by-step approach so you don’t need to remember techniques
- Great places to go on holiday

Picture in Hoi An from lwebber28 travelling in Vietnam
https://www.instagram.com/p/BnbyXRVA7Jz/?taken-by=hotelsdotcom

Methodology

A methodology for approaching memory leaks

1. Do I have a leak (that needs fixing) ?
2. What is leaking (which classes) ?
3. What is keeping objects alive (an instance in the app) ?
4. Where is it leaking from (code where the objects are created and/or assigned) ?

Picture of Juanillo Beach from Carla travelling in the Dominican Republic
https://www.instagram.com/p/Bng782cAgNQ/?taken-by=hotelsdotcom

OOME

A methodology for approaching memory leaks

1. **Do I have a leak (that needs fixing) ?**
2. What is leaking (which classes) ?
3. What is keeping objects alive (an instance in the app) ?
4. Where is it leaking from (code where the objects are created and/or assigned) ?

You *might* have a leak if you get an OOME

- IMPORTANT! Read the OOME Message, it tells you specifically which space caused the leak
- You probably have a leak, BUT
- Maybe your heap is just too small for your application, so check if a larger heap works
- The next section on GCViewer will help you work out if it’s a leak

Picture in Plaza Espana from someone travelling in Seville, Spain
https://www.instagram.com/p/BkV5a1kDxRB/?taken-by=hotelsdotcom

Two Generation Heap

A methodology for approaching memory leaks

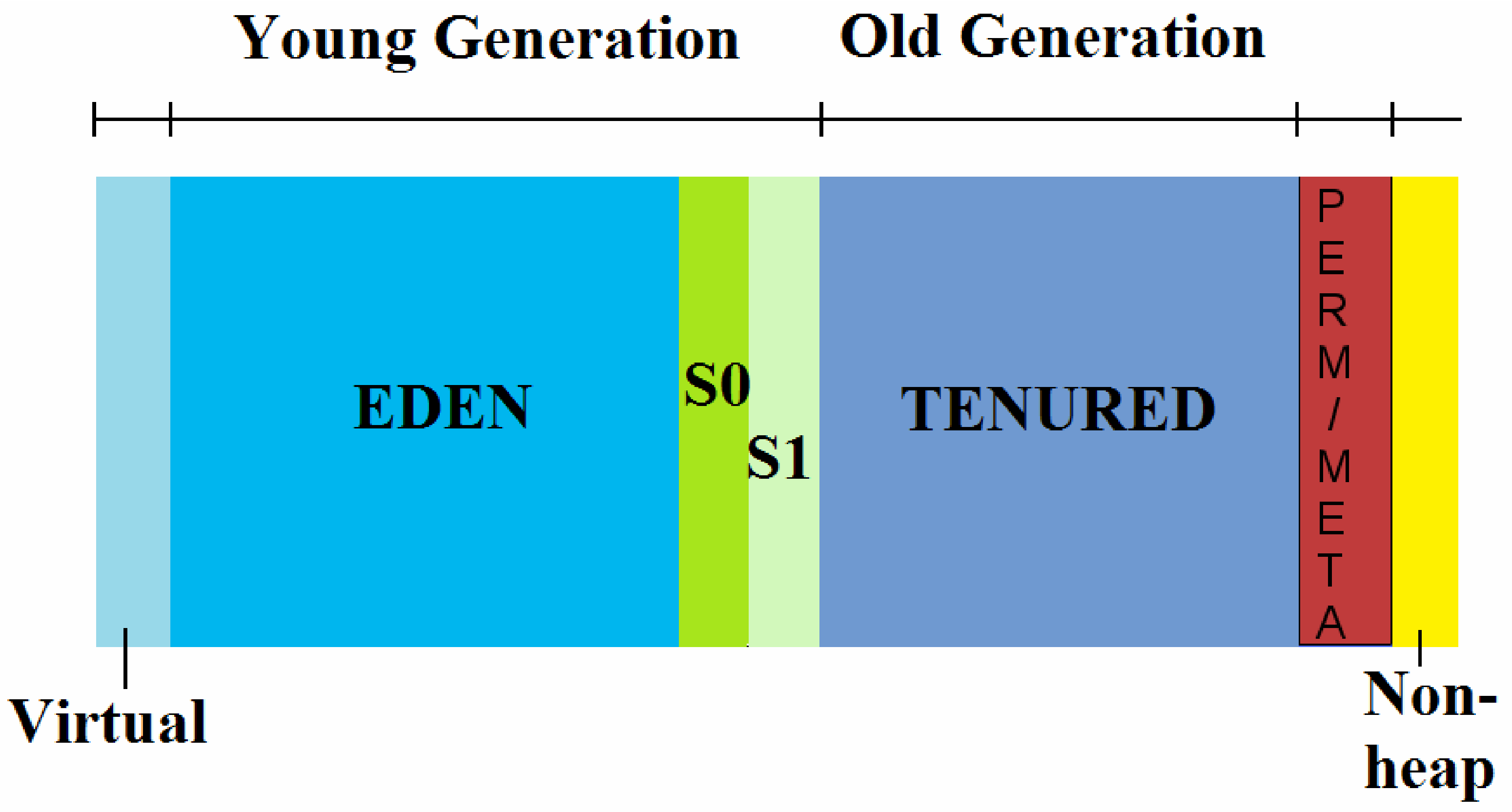
1. **Do I have a leak (that needs fixing) ?**
2. What is leaking (which classes) ?
3. What is keeping objects alive (an instance in the app) ?
4. Where is it leaking from (code where the objects are created and/or assigned) ?

Young And Old Generation Heaps

- You need to know this so that you can analyse GC
- But it’s pretty straightforward for memory leak analysis
- Objects are created in the Young generation and last a while there
- Then if they stay alive long enough, they move to the old generation
  - Old generation GCs take a long time
  - Young generation GCs are quick

- That’s it!

Young And Old Generation Heaps



GC logging

- Turn on GC logging
  - Before Java 9
    - -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -Xloggc:[file] -XX:+PrintReferenceGC -XX:+PrintTenuringDistribution -XX:+PrintGCApplicationStoppedTime -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=10 -XX:GCLogFileSize=10M
  - Java 9+
    - -Xlog:gc*,gc+ref=debug,gc+age=trace,gc+heap=debug:file=gc%p%t.log:tags,uptime,time:filecount=10,filesize=10m

GCViewer & Memory leaks

DEMO

GC Log Memory Leak Identification

- Really simple
- Look at the heap used AFTER each Old Generation GC (Full GC)
- If that heap size is continually increasing, you have a leak

- Can also get sudden spike causing OOME – GCViewer will show that too

- GCViewer only shows the heap, not other spaces, so this doesn’t help identify native memory exhaustion leaks
  - Sorry, that’s another talk

Picture from Michael Long travelling in Jamaica
https://www.instagram.com/p/BeWLc-yFUMX/?taken-by=hotelsdotcom

Class Histogram

A methodology for approaching memory leaks

1. Do I have a leak (that needs fixing) ?
2. **What is leaking (which classes) ?**
3. What is keeping objects alive (an instance in the app) ?
4. Where is it leaking from (code where the objects are created and/or assigned) ?

Class Histogram

- `jmap -histo:live <pid>`
- Most profilers memory analysis histogram
- Heap dump histogram

Memory profiling & analysis

DEMO

Picture from Jonny travelling in Puglia, Italy
https://www.instagram.com/p/BneXJuCD2nG/?taken-by=hotelsdotcom

Heap Dump

A methodology for approaching memory leaks

1. Do I have a leak (that needs fixing) ?
2. What is leaking (which classes) ?
3. **What is keeping objects alive (an instance in the app) ?**
4. Where is it leaking from (code where the objects are created and/or assigned) ?

Heap Dump

- `-XX:+HeapDumpOnOutOfMemoryError`
- `jmap -dump:live,file=<file-path> <pid>`
  - Or without “live,” if you want to see dead objects that have not yet been GCed, “live,” forces a GC before the dump
- JMX: com.sun.management.HotSpotDiagnostic.dumpHeap()
  - Eg from jconsole, visualvm, even programmatically
- `jcmd <pid> GC.heap_dump <file-path>`

Heap Dump Viewers

- Lots of profilers and some utilities
- I’m going to use the most popular: Eclipse MAT

Heap dump analysis

DEMO

Picture of Corona Arch from Michael travelling in Utah, USA
https://www.instagram.com/p/BneXJuCD2nG/?taken-by=hotelsdotcom

Generational Profiling

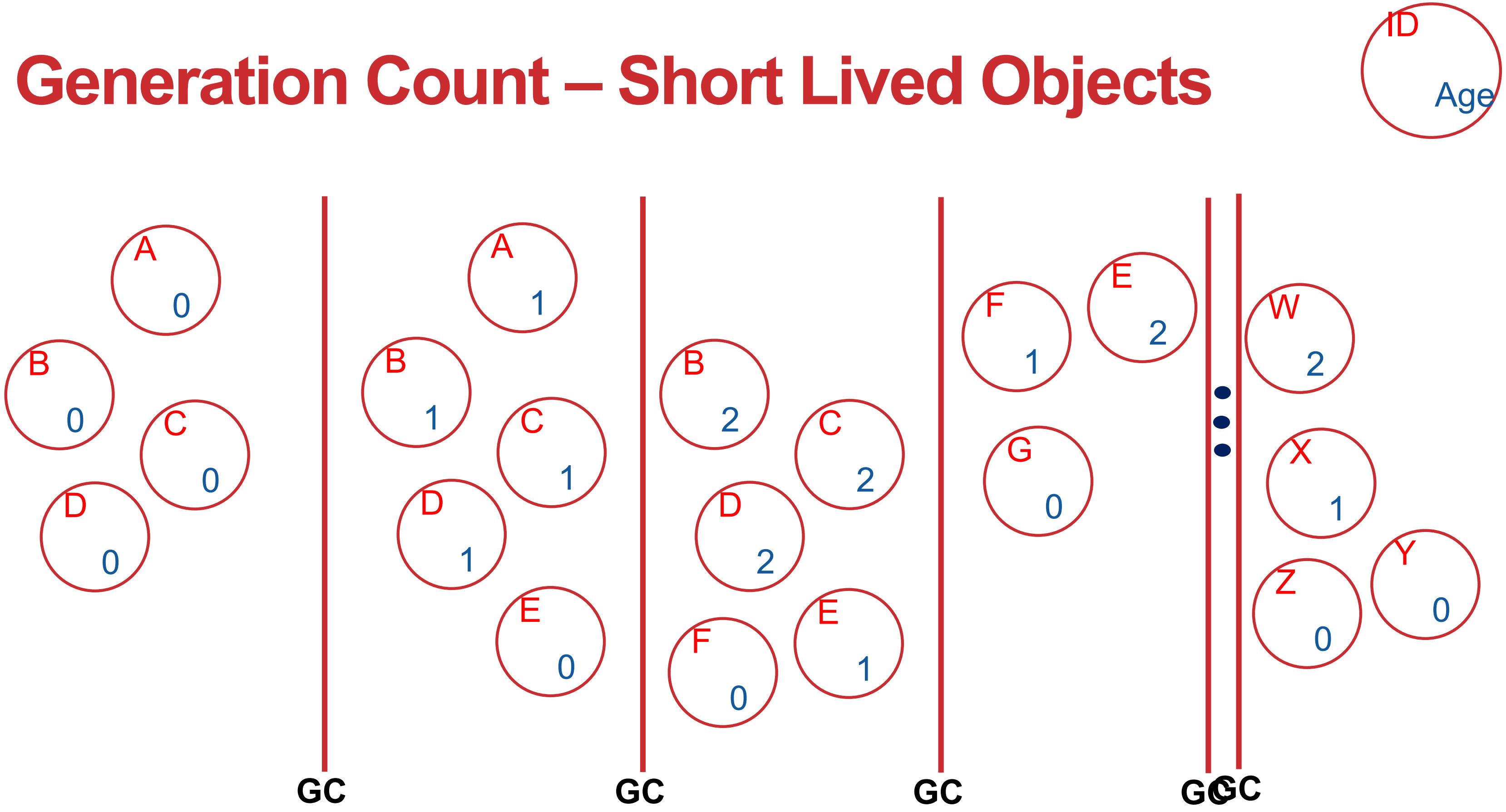
A methodology for approaching memory leaks

1. Do I have a leak (that needs fixing) ?
2. What is leaking (which classes) ?
3. What is keeping objects alive (an instance in the app) ?
4. **Where is it leaking from (code where the objects are created and/or assigned)?**

Memory profiling & analysis

DEMO SETUP

Generation Count – Short Lived Objects



1 generation
(all aged 0)

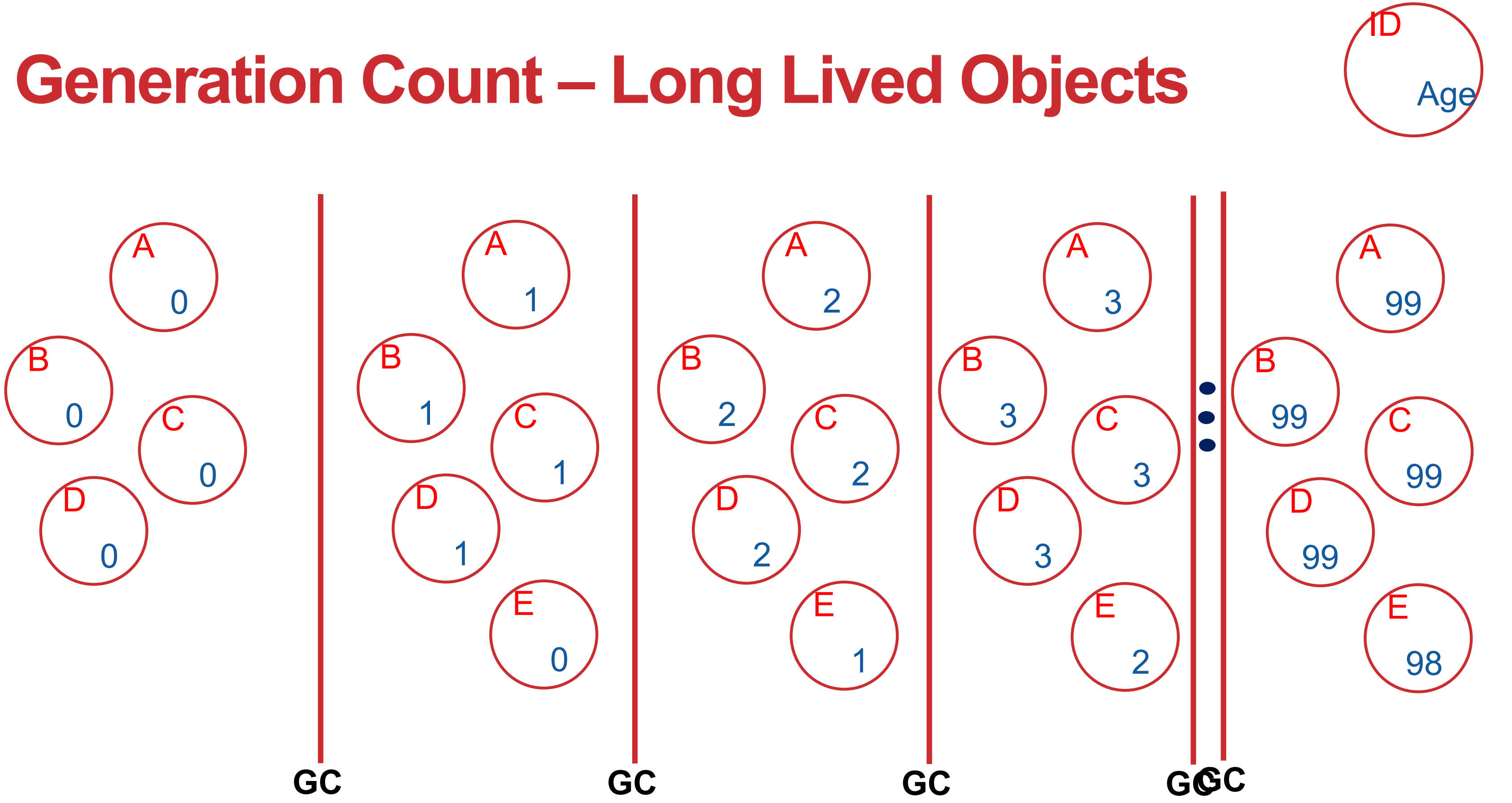
2 generations
(aged 0 and 1)

3 generations
(aged 0, 1 and 2)

3 generations
(aged 0, 1 and 2)

3 generations
(aged 0, 1 and 2)

Generation Count – Long Lived Objects



1 generation (all aged 0)

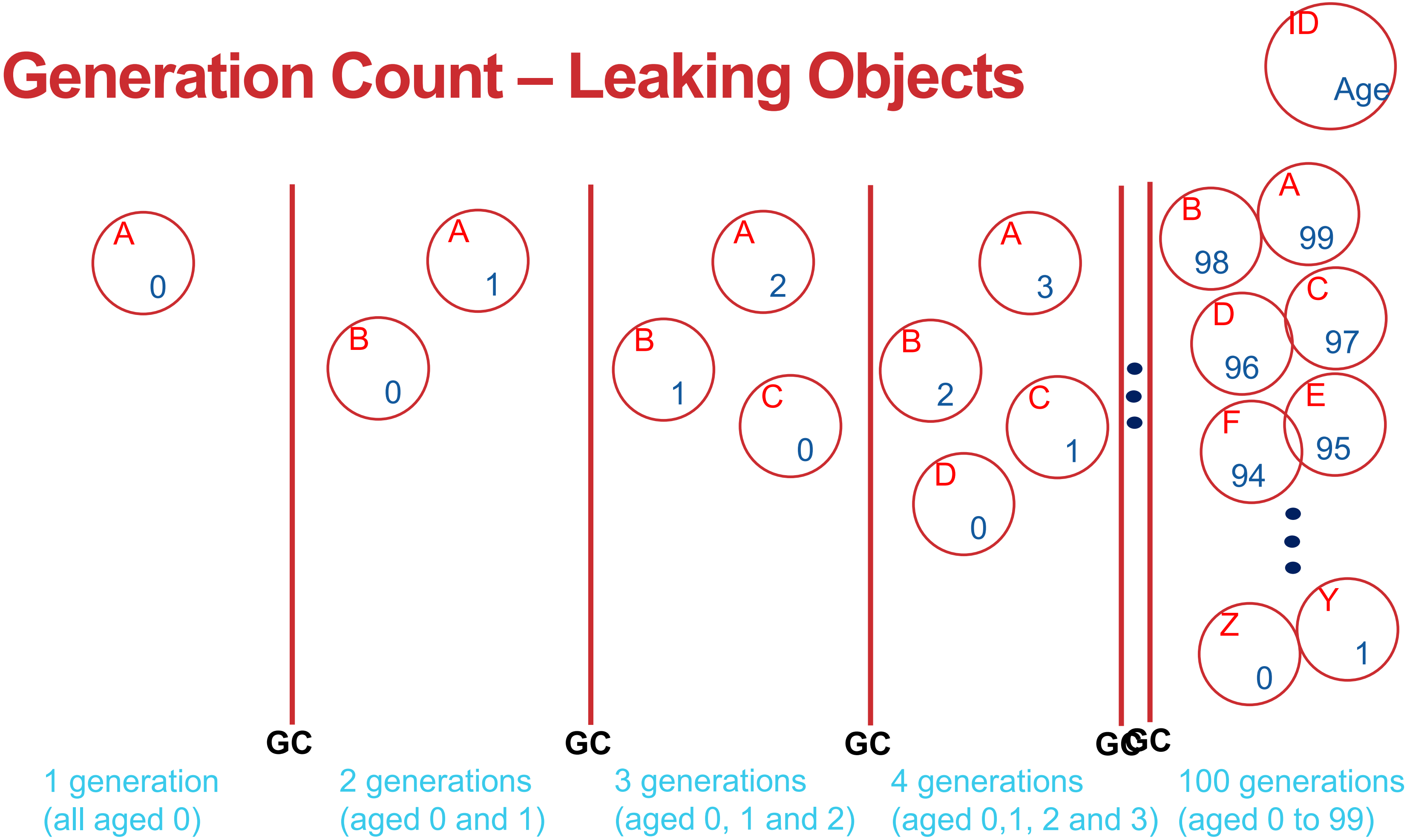
2 generations (aged 0 and 1)

2 generations (aged 1 and 2)

2 generations (aged 2 and 3)

2 generations (aged 98 and 99)

Generation Count – Leaking Objects



Memory profiling & analysis

DEMO

Tools

- GC Logging
  - Suitable for production – GC logs remain after JVM terminates
- GCViewer
  - Suitable for production – views GC logs
- Heap Dumping
  - Suitable for production: **but** freezes JVM so only when necessary – log remains
- Eclipse MAT
  - Suitable for production – views Heap Dumps
- VisualVM (use ‘profiler’ with allocation stack traces recording on)
  - **NOT** Suitable for production – needs a live JVM and can crash it (all too often)

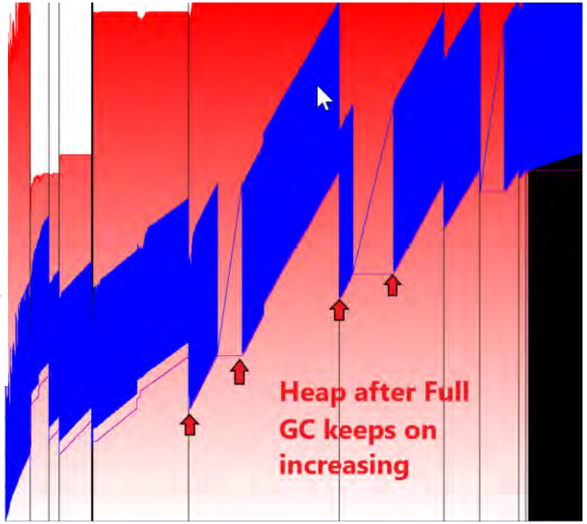
Picture from Alessia travelling in Naples, Italy
https://www.instagram.com/p/BkS_yvDlbxd/?taken-by=hotelsdotcom

Who Am I? Jack Shirazi

- Working in Performance and Reliability Engineering Team at Hotels.com
  - Part of Expedia Group, handling over $100billion in bookings annually
  - World’s largest travel agency
- Founder of JavaPerformanceTuning.com
- Author of Java Performance Tuning (O'Reilly)
- Published over 60 articles on Java Performance Tuning & a monthly newsletter for 15 years & around 10 000 tuning tips

A methodology for approaching memory leaks

1. Do I have a leak (that needs fixing) ?
2. What is leaking (which classes) ?
3. What is keeping objects alive (an instance in the app) ?
4. Where is it leaking from (code where the objects are created and/or assigned) ?


Heap after Full GC keeps on increasing

Tools

- GC Logging
  - Suitable for production – GC logs remain after JVM terminates
- GCViewer
  - Suitable for production – views GC logs
- Heap Dumping
  - Suitable for production – but! freezes the JVM so only when necessary – log remains
- Eclipse MAT
  - Suitable for production – views Heap Dumps
- VisualVM (use 'profiler' with allocation stack traces recording on)
  - **NOT** Suitable for production – needs a live JVM and can crash it (all too often)

Class histogram

- `jmap -histo:live <pid>`
- Most profilers memory analysis histogram
- Heap dump histogram

Heap Dump

- `-XX:+HeapDumpOnOutOfMemoryError`
- `jmap -dump:live,file=<file-path> <pid>`
  - Or without "live," if you want to see dead objects that have not yet been GCed, "live," forces a GC before the dump
- JMX: com.sun.management.HotSpotDiagnostic.dumpHeap()
  - Eg from jconsole, visualvm, even programmatically
- `jcmd <pid> GC.heap_dump <file-path>`

Who am I? Jack Shirazi

- Working in Performance and Reliability Engineering Team at Hotels.com
  - Part of Expedia Group, handling $88billion in bookings 2017
- Founder of JavaPerformanceTuning.com
- Author of Java Performance Tuning (O'Reilly)
- Published over 60 articles on Java Performance Tuning & a monthly newsletter for 15 years & around 10 000 tuning tips
- Also researched Black Hole Thermodynamics & published papers on Protein Structure Prediction with the UKs largest Cancer Research organisation