Memory Analysis in a Nutshell

Elena Nayashkova
SAP AG

Agenda

1. **Concepts**
2. Automated Memory Leak Report
3. Developer’s Use Case - Finding the Needle…
4. Summary
5. Q & A

Introduction

- The number of memory-related problems is very high
- They are extremely difficult to analyze
- Analysis requires expertise in the analyzed coding

Eclipse Memory Analyzer:

- Simplifies memory analysis
- Extensible
- Free for download

Causes of Memory Leaks in Java

- Unwanted object references
- Long-living (static) objects, e.g. static Collections
- Unregistered Listeners
- Huge Sessions
- Forgotten Threads
- Blocking Finalizers
- etc.

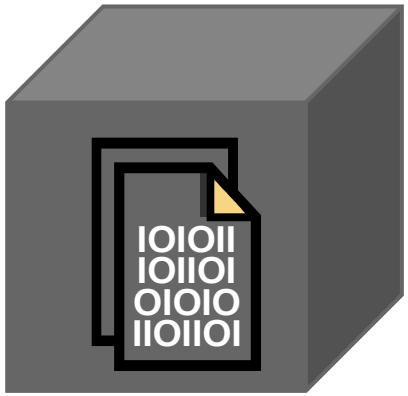
HPROF Binary Heap Dump

A heap dump is a **snapshot of objects that are alive** at one point in time.
It contains:

- Objects: fields, references, primitive values, …
- Classes: class loader, super class, static fields, …

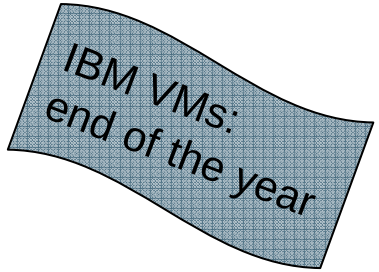
A heap dump does **not** contain

- where the objects have been created
- which objects have been garbage collected



How to Get a Heap Dump

- Non-Interactive
  -XX:+HeapDumpOnOutOfMemoryError

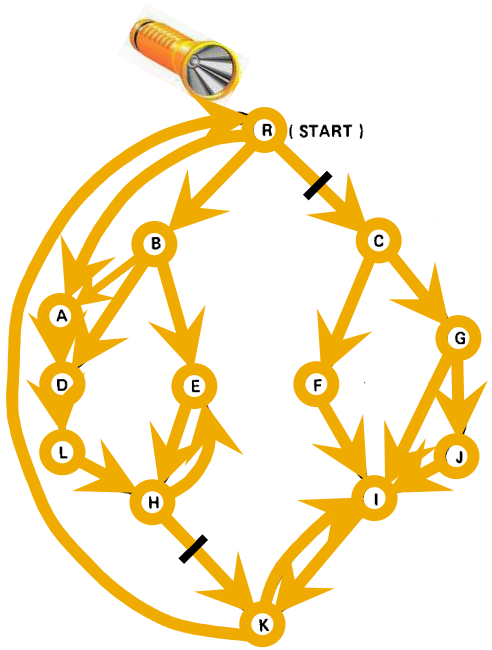


- On Demand
  JDK1.4.2_12 and -XX:+HeapDumpOnCtrlBreak
  JDK6 and Jconsole

- More…
  http://wiki.eclipse.org/index.php/MemoryAnalyzer#Getting_a_Heap_Dump

Definition of Retained Set and Retained Size

- Shallow heap is the memory consumed by one object
- Retained set of **X** is the set of objects that will be garbage collected if **X** is garbage collected
- Retained heap of **X** is the sum of shallow sizes of all objects in the retained set of **X**, i.e. memory kept alive by **X**

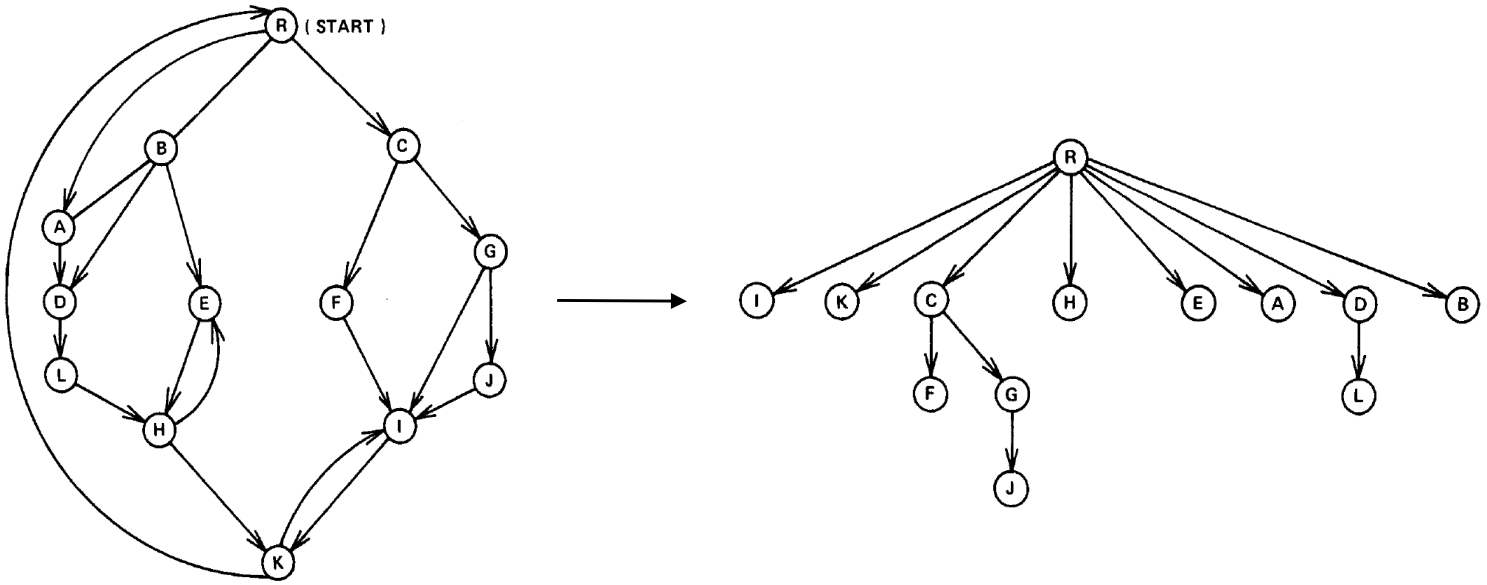


Set of elements	Retained Set
C	C, F, G, J
K	K
C, K	C, F, G, J, K, I

Dominator Tree

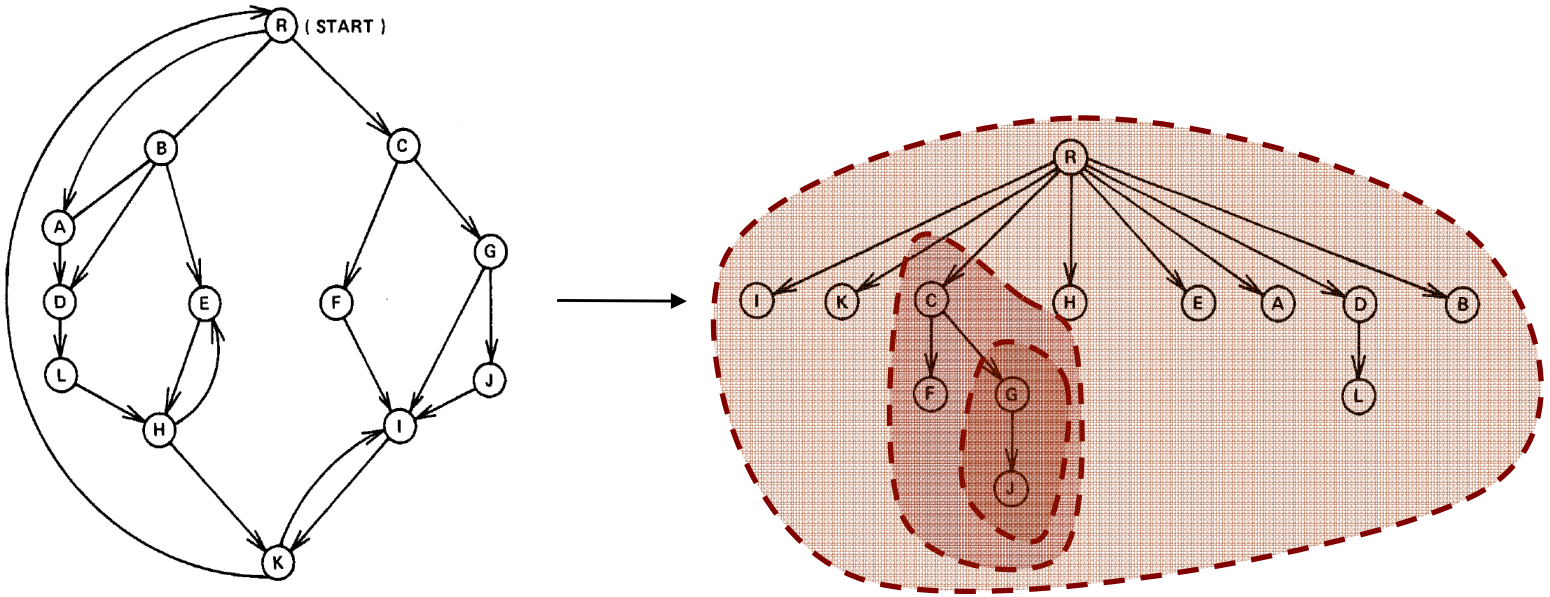
The Dominator Tree is a Transformation of the Cyclic Object Graph into a „Keep-Alive“ Tree:

- Every node in the Tree is directly responsible for keeping alive its children
- Object **X** dominates object **Y** if all paths from the roots to **Y** run through **X**



Dominator Tree: Benefits

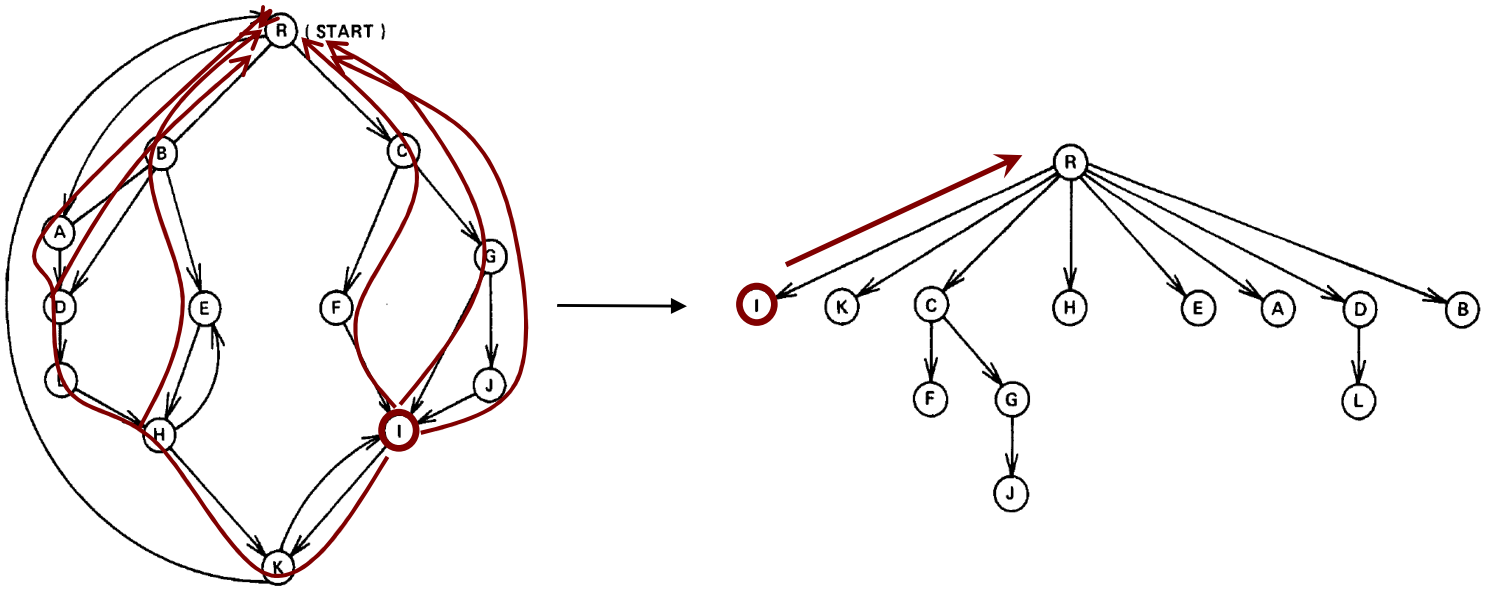
Fast Calculation of the Retained Size (sum all children)



Retained Set is a subtree

Dominator Tree: Benefits

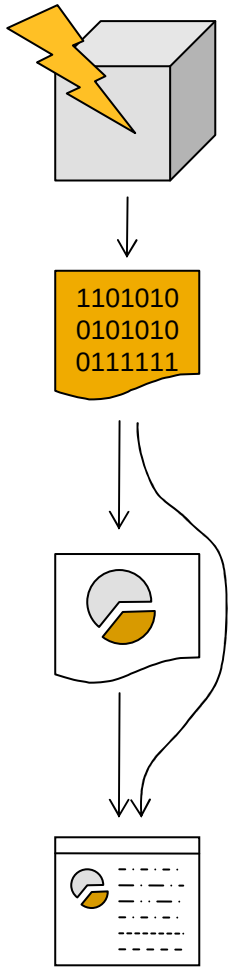
Fast Identification of Responsible Objects (just go up the tree)



Agenda

1. Concepts
2. **Automated Memory Leak Report**
3. Developer's Use Case - Finding the Needle…
4. Summary
5. Q & A

Behind the Scenes



VM throws an Out Of Memory Error
-XX:+HeapDumpOnOutOfMemoryError

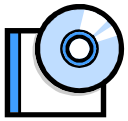
A heap dump is written to the file system

Parsing of the heap dump is triggered
Index files are generated for a fast access to the data
A Dominator Tree is computed out of the object graph

Analysis is performed and a report is generated

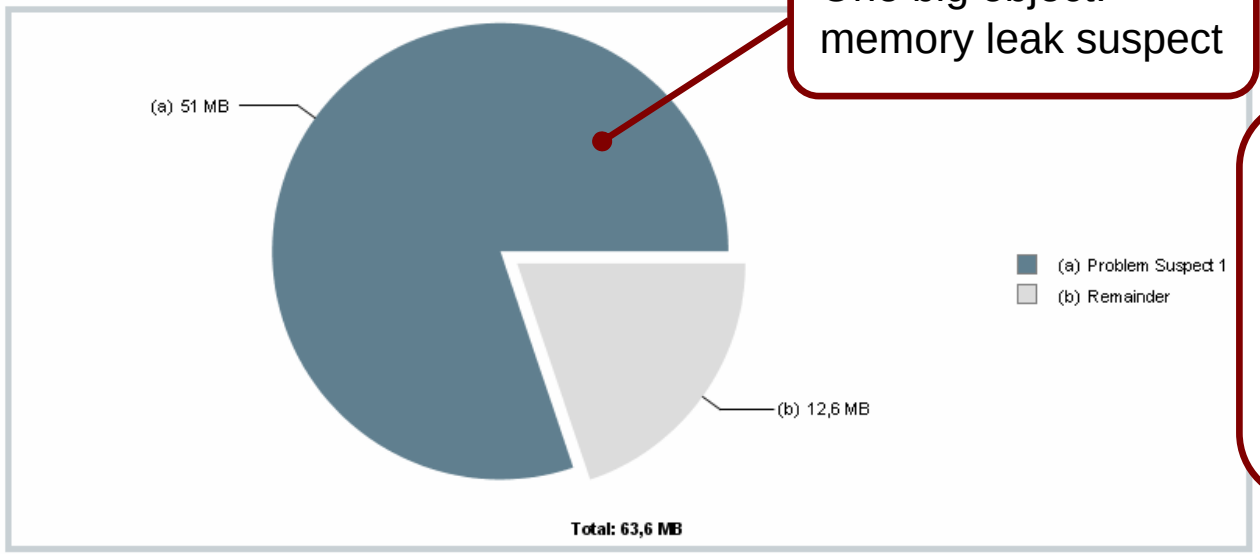
Demo

Memory Leak Hunter



- Automatically detect memory leak suspects
- Discover if the issue is known (and a fix available)
- Collect details for in depth analysis by the code experts

Report Overview



One big object: memory leak suspect

Any up-to-date architecture loads components with separate class loaders, be it OSGi or JEE application servers.
Extensible to display meaningful names.

Problem Suspect 1

One instance of **"org.eclipse.mat.demo.leak.LeakingQueue"** loaded by **"org.eclipse.mat.demo.leak"** occupies **53.487.288 (80,18%)** bytes. The memory is accumulated in one instance of **"java.lang.Object[]"** loaded by **"<system class loader>"**.

Keywords
java.lang.Object[]
org.eclipse.mat.demo.leak.LeakingQueue
org.eclipse.mat.demo.leak

CSN Components
SOME-COMPONENT for "org.eclipse.mat.demo.leak"

Details »

Search by keywords: identify if problem is known

Classification for trouble ticket system: less ping-pong of trouble tickets.

Report Details

Shortest Paths To the Accumulation Point

Class Name	Shallow Heap	Retained Heap
java.lang.Object[768] @ 0x49645a0	3.088	53.487.248
queue java.util.PriorityQueue @ 0x3a703b8	24	53.487.272
events org.eclipse.mat.demo.leak.LeakingQueue @ 0x3a703a8	16	53.487.288
eventQueue org.eclipse.mat.demo.leak.LeakQueueProcessor @ 0x3a703d0 LeakQueue Processor Thread Thread	**96**	**148.560**
lq class org.eclipse.mat.demo.leak.AnotherClassReferenciqTheQueue @ 0x7a216d0 »	8	8
∑ **Total: 2 entries**		

The chain of objects and references which keep the suspect alive

Accumulated Objects

Class name	Shallow Heap	Retained Heap	Percentage
org.eclipse.mat.demo.leak.LeakingQueue @ 0x3a703a8	16	53.487.288	80,18%
java.util.PriorityQueue @ 0x3a703b8	24	53.487.272	80,18%
java.lang.Object[768] @ 0x49645a0	3.088	53.487.248	80,18%
org.eclipse.mat.demo.leak.LeakEventImpl @ 0x2ada618	16	74.080	0,11%
org.eclipse.mat.demo.leak.AnotherLeakEventImpl @ 0x2ada640	16	74.080	0,11%
org.eclipse.mat.demo.leak.LeakEventImpl @ 0x2ada668	16	74.080	0,11%
org.eclipse.mat.demo.leak.AnotherLeakEventImpl @ 0x2ada690	16	74.080	0,11%
org.eclipse.mat.demo.leak.LeakEventImpl @ 0x2ada6b8	16	74.080	0,11%
org.eclipse.mat.demo.leak.AnotherLeakEventImpl @ 0x2ada6e0	16	74.080	0,11%
org.eclipse.mat.demo.leak.LeakEventImpl @ 0x2ada708	16	74.080	0,11%
org.eclipse.mat.demo.leak.AnotherLeakEventImpl @ 0x2ada730	16	74.080	0,11%

A significant drop in the retained sizes shows the accumulation point

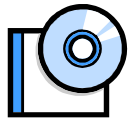
Accumulated objects

Agenda

1. Concepts
2. Finding Memory Leaks
3. **Developer's Use Case - Finding the Needle…**
4. Summary
5. Q & A

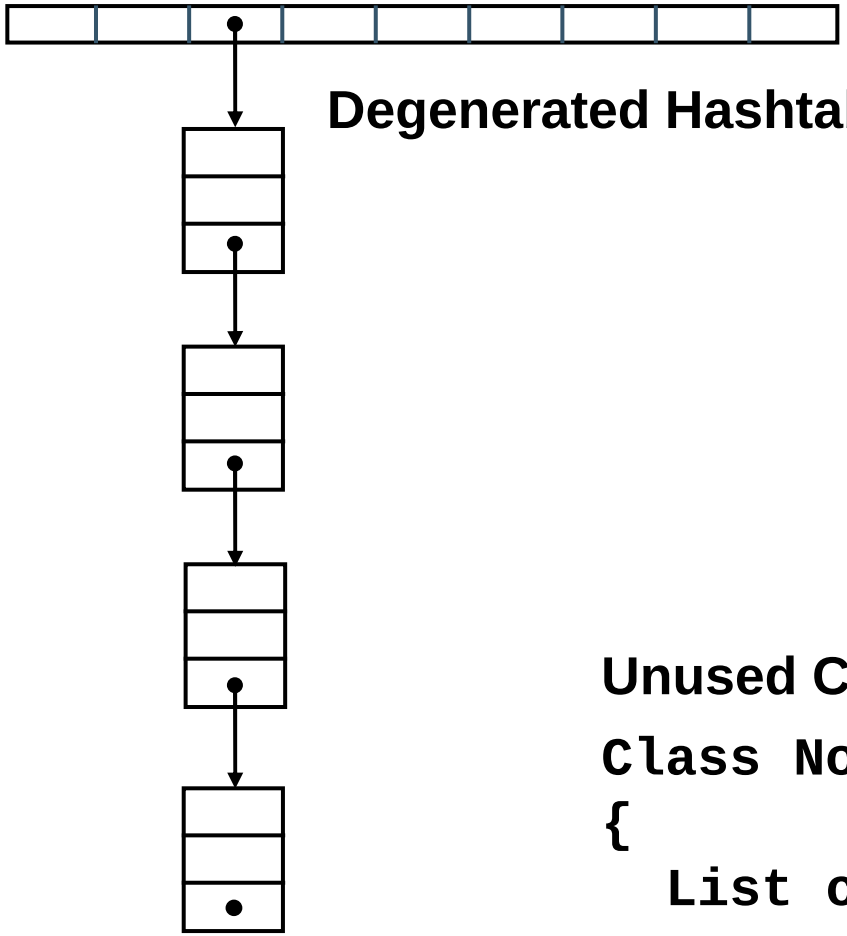
Demo

Developer Use Case



- Limit analysis to developer‘s components
- Give hints where memory footprint can be optimized
- Check for known antipatterns

Inefficient Data Structures



Degenerated Hashtable

Duplicate Strings

Unused Collections

```
Class Node
{
  List children =
        new ArrayList();
}
```

Agenda

1. Concepts
2. Finding Memory Leaks
3. Developer's Use Case - Finding the Needle…
4. **Summary**
5. Q & A

Summary

- Memory Analyzer reduces complexity of handling memory issues
- Automated analysis of memory leaks and footprint
- SAP has contributed Memory Analyzer to the open source

Agenda

1. Concepts
2. Finding Memory Leaks
3. Developer’s Use Case - Finding the Needle…
4. Summary
5. **Q & A**

Thank you!

Elena Nayashkova
SAP AG

Memory Analyzer @ Eclipse:
www.eclipse.org/mat

Memory Analyzer Wiki @ SAP:
www.sdn.sap.com/irj/sdn/wiki?path=/display/Java/Java+Memory+Analysis

Blogs:
dev.eclipse.org/blogs/memoryanalyzer