eclipseCON™ 2008

Memory Analysis Simplified
Automated Heap Dump Analysis

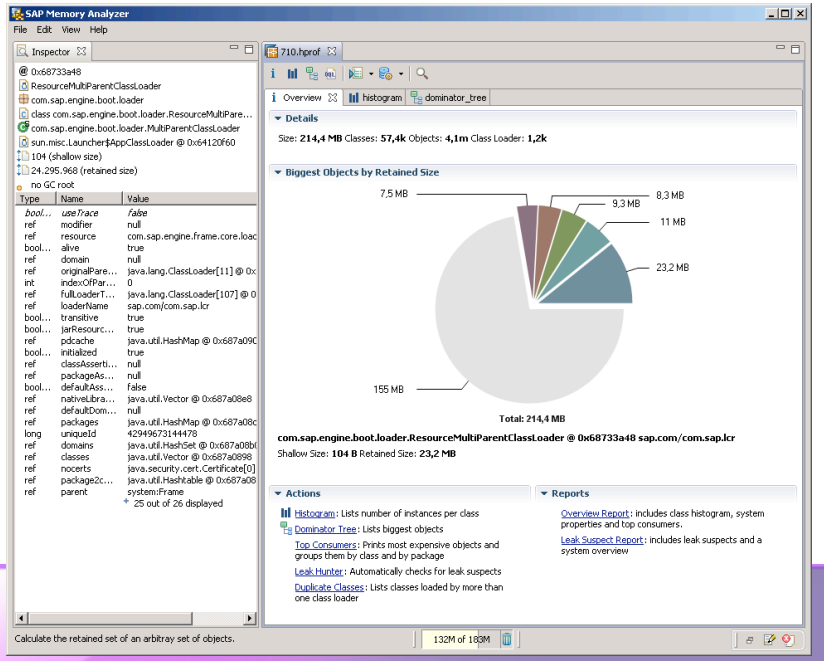
for Developers, Testers and Technical Support Employees

Andreas Buchen, SAP

Eclipse Memory Analyzer

- Provide General Purpose Tooling to analyze Java Heap Dumps.
- Report automatically detected Leak Suspects.
- Stable Code Basis donated by SAP.

- Pluggable
  - Heap Formats
  - Application Knowledge
  - Heap Inspections



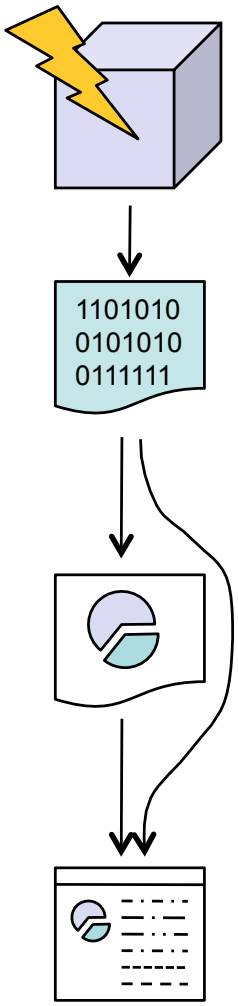
Agenda

- Automation for Technical **Support** Staff
  - Leak Suspects
  - Memory Intensive Threads
- Automation for **Developers**
- Questions & Answers

Support Use Case – Goals

- Find Leaks without immediate Familiarity and Expertise with the Java Code at hand
- Search if the Issue is known (and a Fix available)
- Forward to the responsible Development Support

Support Use Case – Scenario



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

The Heap Dump is a Snapshot of all Objects alive at one Point in Time: objects, classes, class loaders.

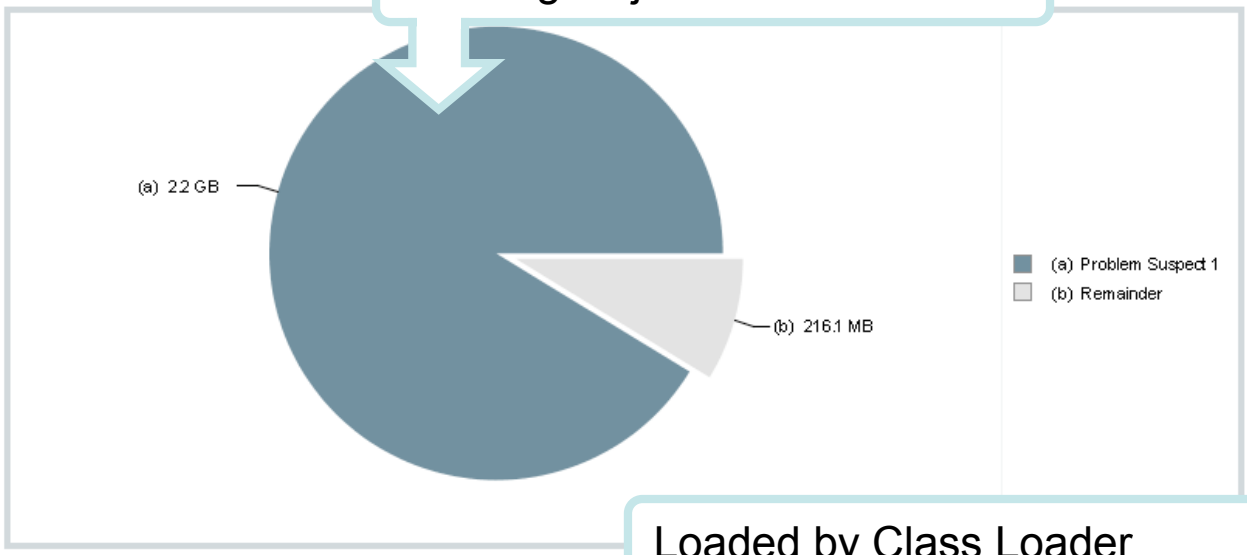
Parse & Report Leak Suspects

Developer continues Analysis Interactively

Demo

- Parse Heap Dump
- Investigate Leak Report

One Big Object



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

Loaded by Class Loader

Problem Suspect 1

Keywords

One instar…epository" loaded by "library:bi~mmr~mmrlib" occupies 2,366,237,448 (91.26%) b… …memory is accumulated in one instance of "java.lang.Object[]" loaded by "<system class loader>".

Keywords
com.sap.ip.mmr.Repository
library:bi~mmr~mmrlib
java.lang.Object[]

CSN Components
BW-BEX-MMR for "library:bi~mmr~mmrlib"

Det…

Search tuple: Identify if problem is known.

Less ping-pong of trouble tickets.

Classification for trouble ticket system

Shortest Paths to the Accumulation Point

Short…

Class Na…	Shallow Heap	Retained Heap
java.lang.Object[58837] @ 0xb66e7d50	470,720	2,366,169,488
elementData java.util.ArrayList @ 0xc41cb690	40	2,366,169,528
messages com.sap.ip.mmr.Repository @ 0xc400f928	120	2,366,237,448
<Java Local> com.sap.engine.core.thread.impl3.SingleThread @ 0xb818c280 HTTP Worker [2] [user: Guest, app: sap.com/is~cmp.bop.bcl.crmws~ear] Native Stack, Thread	224	7,380,160
repository com.sap.ip.mmr.xmi20.XmiReader @ 0xb7f95290 »	24	24
repository com.sap.ip.mmr.xmi20.Xmi2Mof14 @ 0xb7f952c8 »	120	2,328
repository com.sap.ip.mmr.Connection @ 0xc40c3c08 »	56	56
rcl com.sap.ip.mmr.foundation.ClassLoaderMMR @ 0xc40c3a60 com.sap.ip.mmr.foundation.ClassLoaderMMR @ 0xc40c3a60 »	168	1,328
∑ Total: 5 e…		

Who is keeping the leak suspect alive?

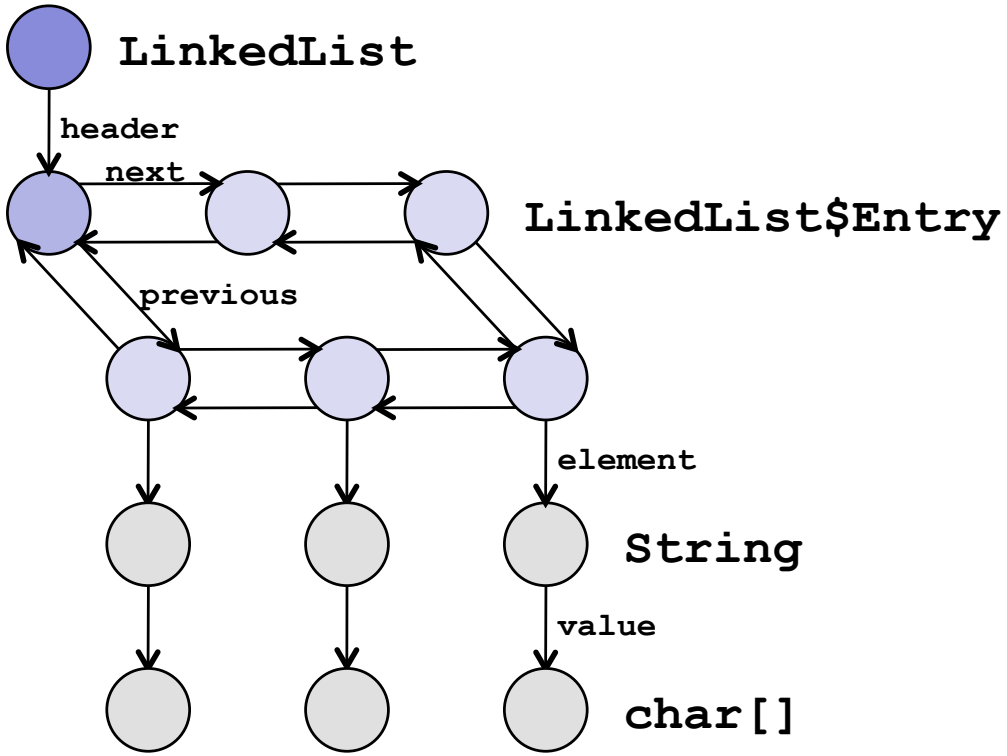
Accumulation Point in the Dominator Tree

Accumulation p…

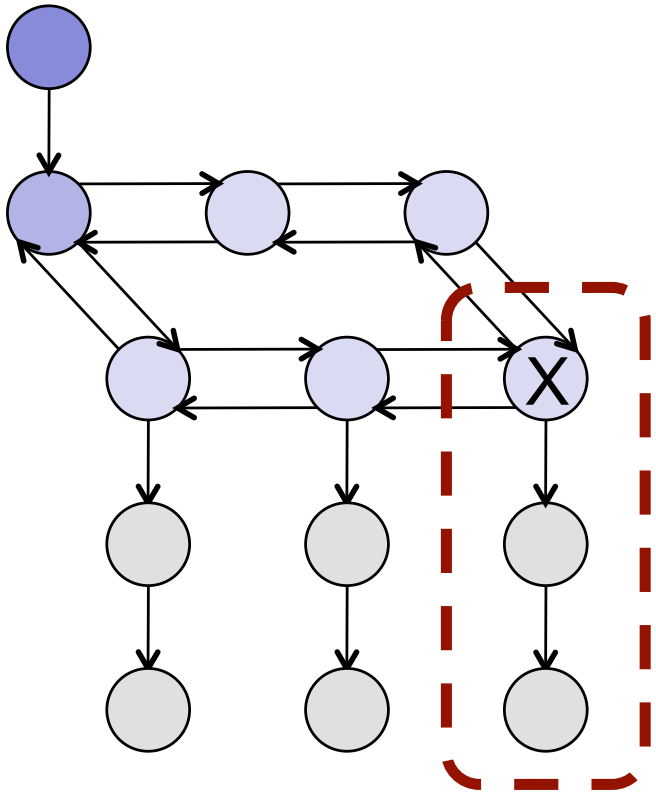
Class name	Shallow Heap	Retained Heap	Percentage
com.sap.ip.mmr.Repository @ 0xc400f928	120	2,366,237,448	91.26%
java.util.ArrayList @ 0xc41cb690	40	2,366,169,528	91.26%
java.lang.Object[58837] @ 0xb66e7d50	470,720	2,366,169,488	91.26%
java.lang.String @ 0xb5ad83f8 com.sap.ip.mmr.foundation.JmiException: Element name cannot be resolved; unknown package com.sap.caf.metamodel.BusinessEntityInterface, Value is: com.sap.caf.metamodel.BusinessEntityInterface\ud\ua\u9at com.sap.ip.mmr.xmi20.Xmi2Mof14.resolveElementName(Xm…	40	59,744	0.00%
java.lang.String @ 0xf3b7c348 com.sap.ip.mmr.foundation.JmiException: Element name cannot be resolved; unknown package com.sap.caf.metamodel.BusinessEntityInterface, Value is: com.sap.caf.metamodel.BusinessEntityInterface\ud\ua\u9at com.sap.ip.mmr.xmi20.Xmi2Mof14.resolveElementName(Xm…	40	59,744	0.00%
java.lang.String @ 0xfbe851c0 com.sap.ip.mmr.foundation.JmiException: Element			

What is kept alive by the leak suspect?

A Simple Object Graph



How to Calculate the Retained Size



The Retained Size of an Object X is the Memory that would be freed by the Garbage Collector if no References to the Object X would exist.

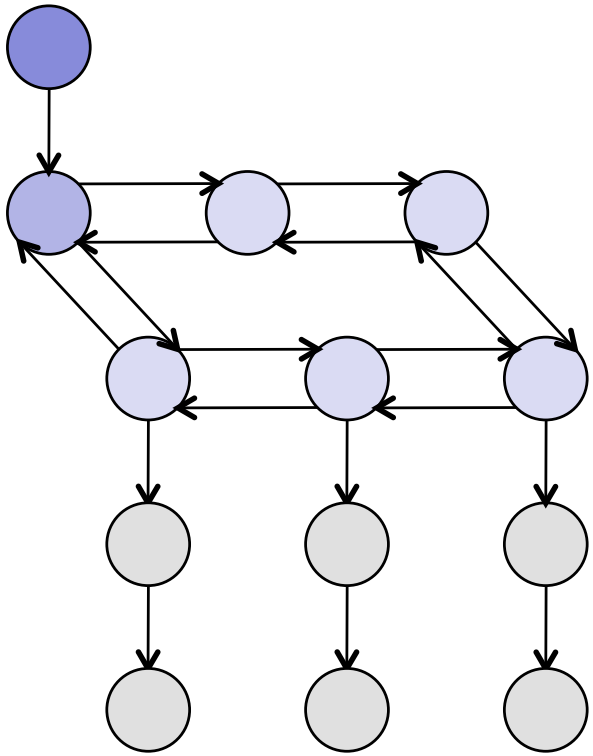
How to Calculate the Retained Size (2)

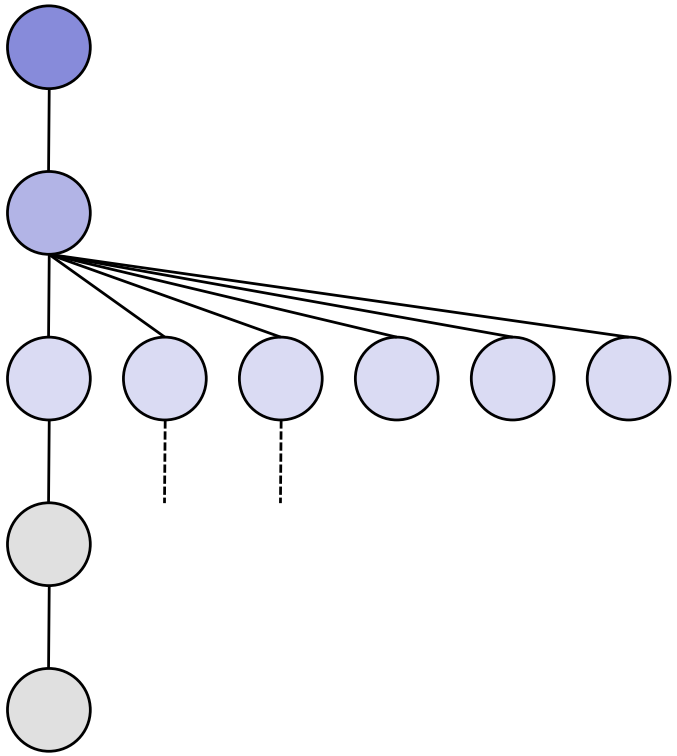
- To calculate the Retained Size, one removes any References to the object and marks all Objects reachable from the GC Roots. The sum of the Shallow Sizes of the non-marked Objects is the Retained Size.

Garbage Collection (GC) Roots

- GC Roots are objects that are assumed to be reachable. Typically, these include all objects referenced from the current call stacks and classes loaded by the system class loader.

Dominator Tree: Transformation into a “Keep-Alive” Tree





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.
  - “X dominates Y if all paths from the roots to Y run through X”

http://en.wikipedia.org/wiki/Dominator_%28graph_theory%29

Dominator Tree – Benefits

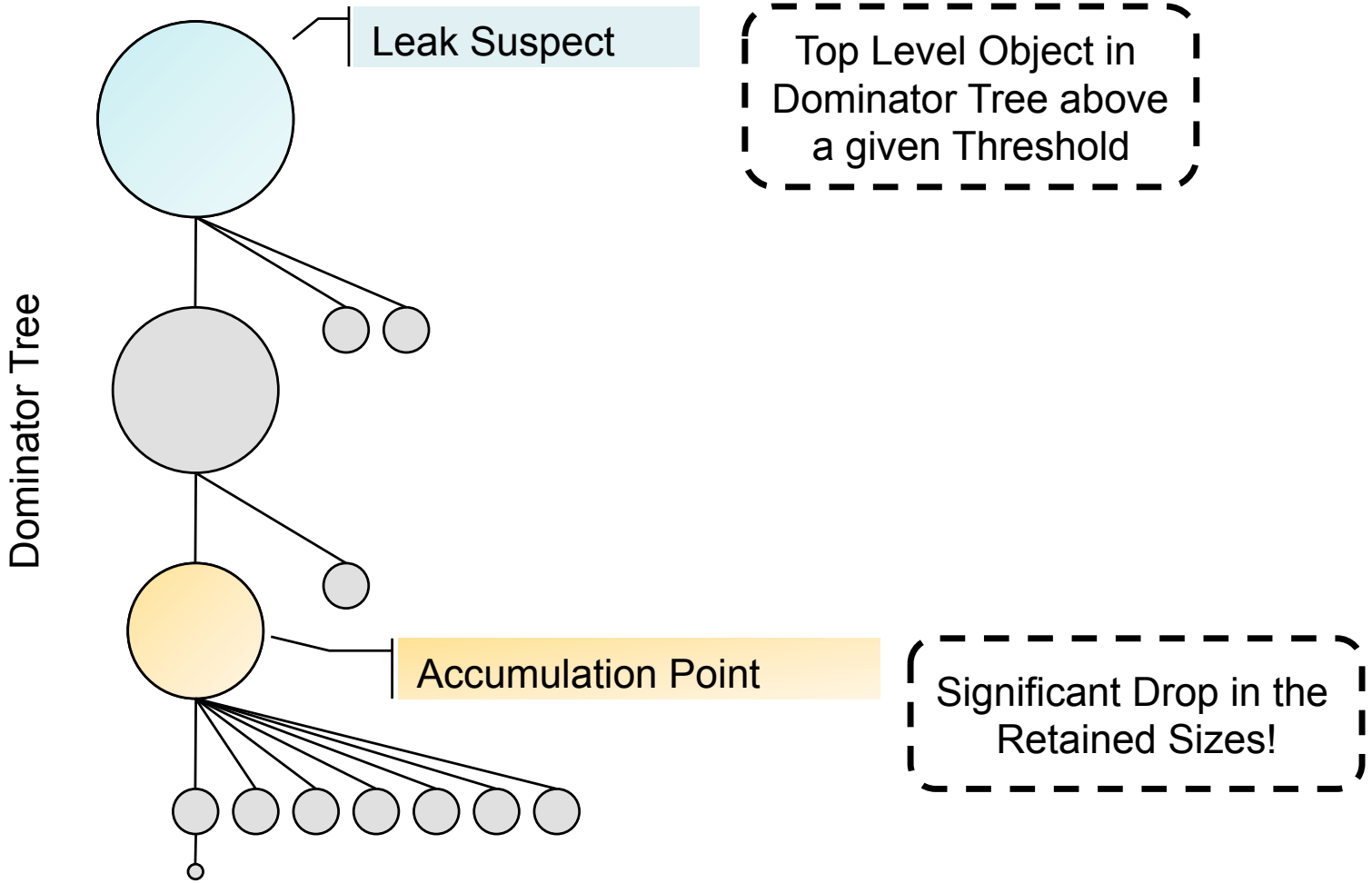
→Fast Calculation of the Retained Size (sum all children)

→List of Distinct Big Objects (first Level of the Tree)

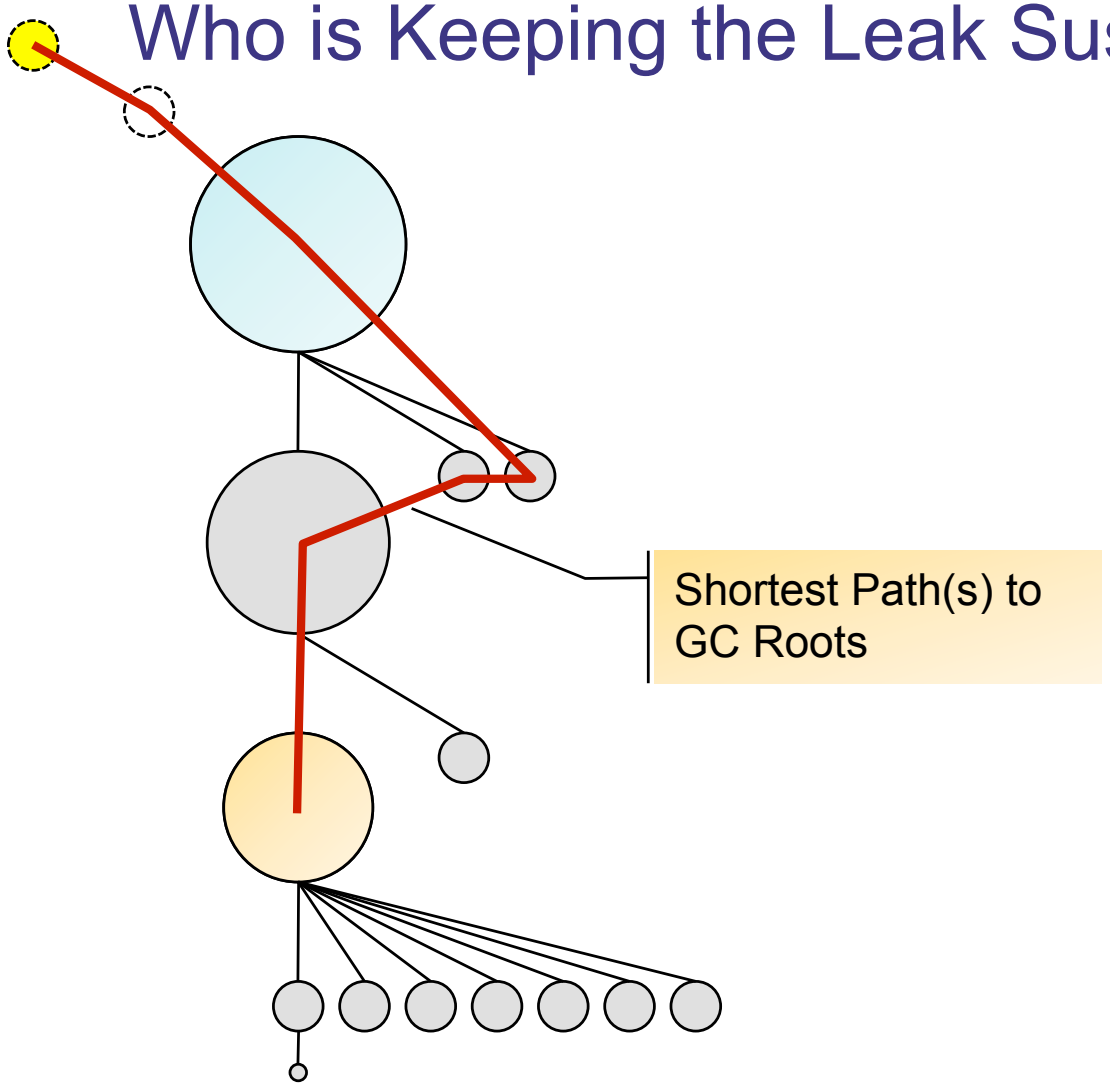
→Fast Identification of responsible Objects (just go up the tree)

http://en.wikipedia.org/wiki/Dominator_%28graph_theory%29

What is a Leak Suspect?



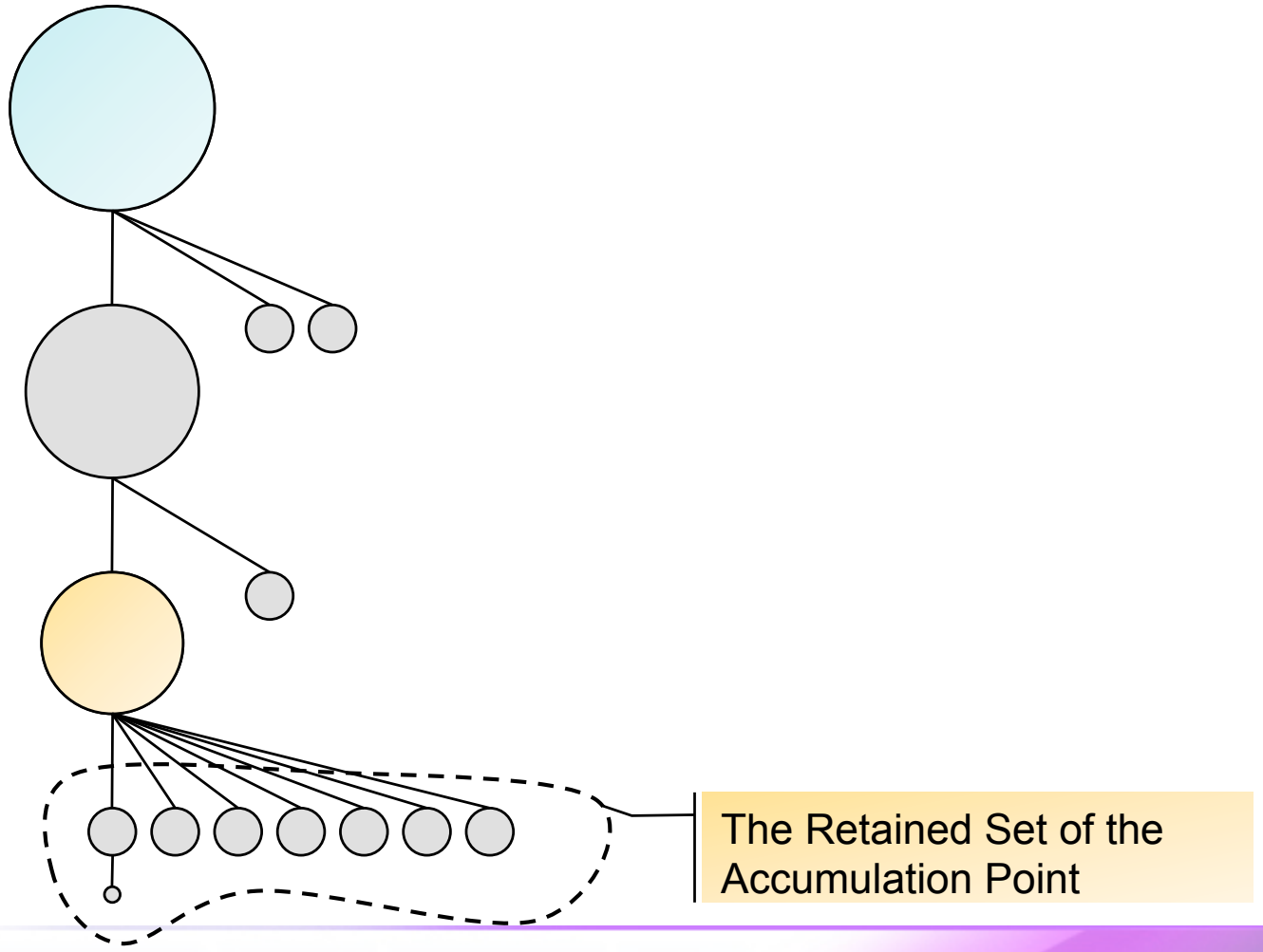
Who is Keeping the Leak Suspect Alive?



Shortest Path(s) to GC Roots

Note: All Paths must go through the Parent Nodes, but not necessarily directly (see Linked List Example).

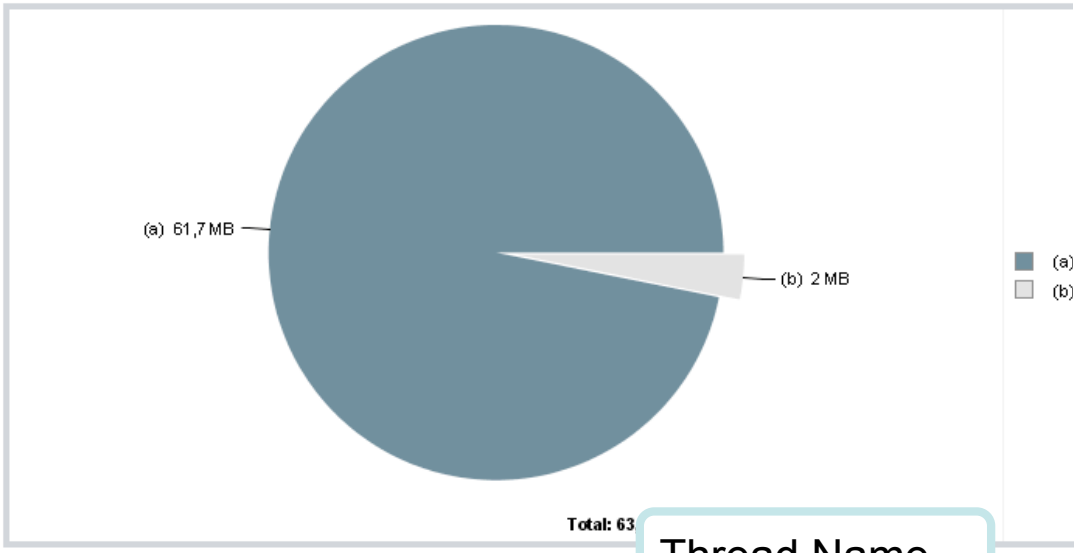
What Is Kept Alive by the Leak Suspect?



Agenda

- Automation for Technical **Support** Staff
  - Leak Suspects
  - **Memory Intensive Threads**
- Automation for **Developers**
- Questions & Answers

Demo



Problem Suspect 1

The thread **org.mortbay.thread.BoundedThreadPool$PoolThread @ 0x2eee408 btpool0-9** keeps variables with total size **64.746.336 (96,85%)** bytes.

The memory is accumulated in one instance of **"java.util.LinkedList$Entry"** loaded by **"<system cl loader>"**.

The thread is executing an HTTP Request to **/demo/index.jsp**.

Keywords
java.util.LinkedList$Entry

Details »

Thread Name

Request URL

Thread Details

Thread btpool0-9

Thread Properties

Name	btpool0-9
Instance	org.mortbay.thread.BoundedThreadPool$PoolThread @ 0x2eee408
Shallow Heap	112
Retained Heap	64.746.336
Status	5
Context Class Loader	/demo
Pool Name	
Task Name	
Subtask Name	
User Name	
Task Stack Depth	
∑ Total: 11 entries	

Summary

The thread is executing an HTTP Request to **/demo/index.jsp**.

URI

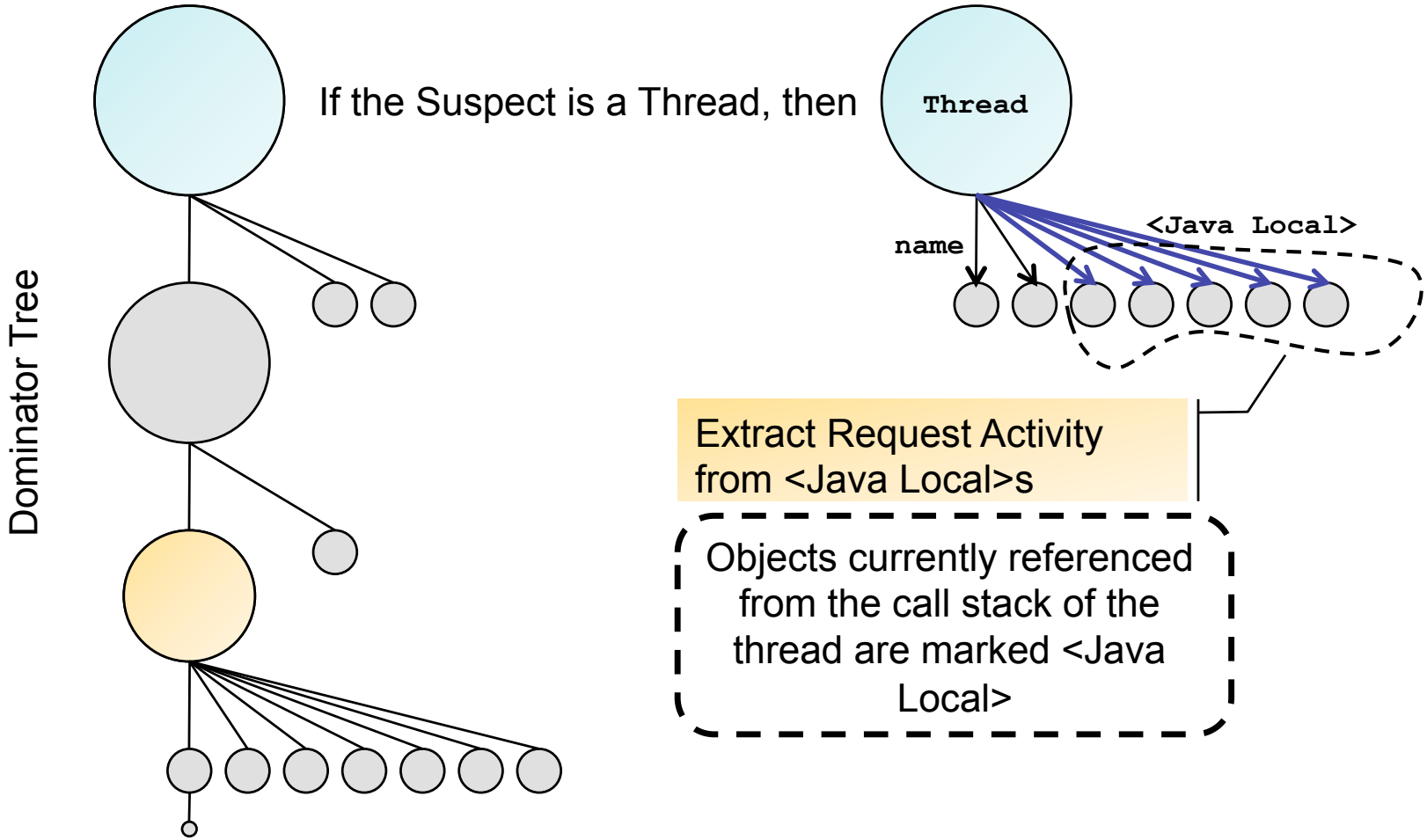
```
GET /demo/?flag=true&mbPerSecond=20 HTTP/1.1..Host: localhost:8080..Us
```

Parameters

Key	Value
flag	true
mbPerSecond	20

More Request Details (Parameters etc.) provided by Pluggable Request Resolvers

What are we doing? (No Magic)



Agenda

- Automation for Technical **Support** Staff
  - Leak Suspects
  - Memory Intensive Threads
- **Automation for Developers**
- Questions & Answers

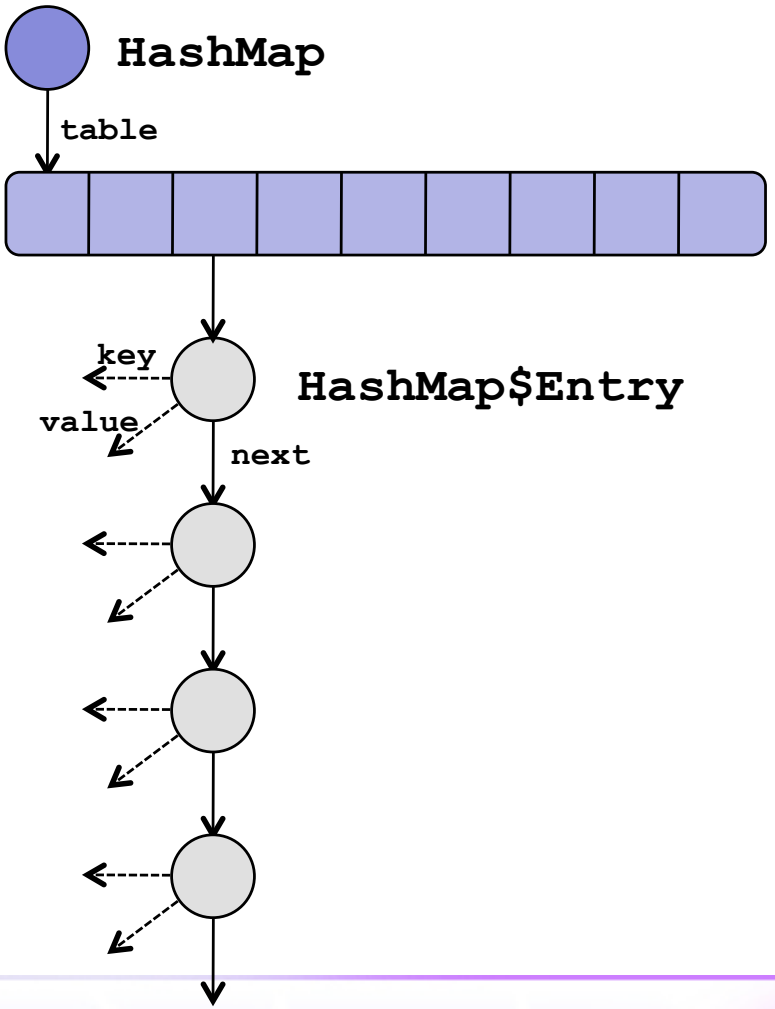
Developer Use Case - Goals

- Limit Analysis to Components, not the whole Heap
- Report on Memory Consumption & Leaks
  - Excessive Use of Empty Collections
  - Duplicate Strings
  - Leaking OSGi Bundles
  - …
- Drill-Down to the last Bits of the Object Graph (if necessary)

Demo

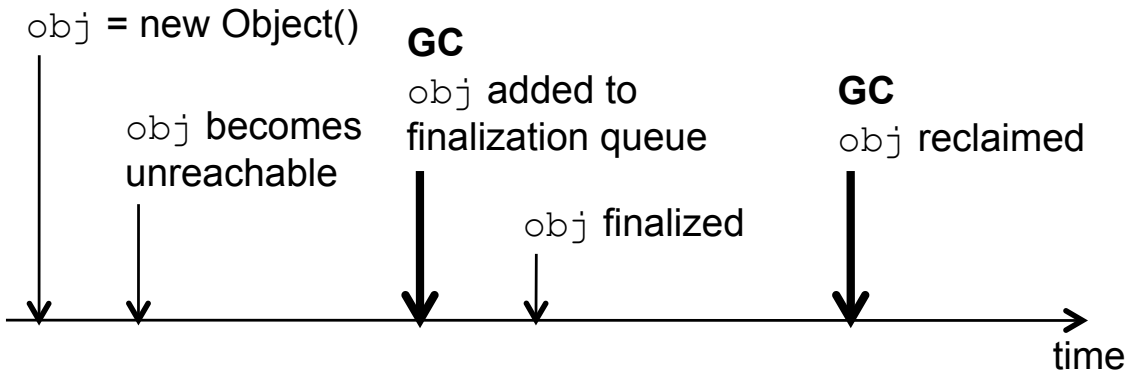
- Investigate Memory Consumption per Component
- “Leaking” Eclipse Bundles
- Finalizer

Degenerated Maps



Finalizer

- A Finalizer is a method that is executed when an object is garbage collected.



How to Handle Java Finalization's Memory-Retention Issues: http://java.sun.com/developer/technicalArticles/javase/finalization/

In Memory Analyzer run
Java Basics → Finalizer Overview

Finalizer in the Heap

Finalizer In Processing

Class Name	Shallow Heap	Retained Heap
java.io.FileInputStream @ 0xa8d73f8 »	16	40

Object currently processed

Finalizer Queue

Class Name	Shallow Heap	Retained Heap
java.io.FileInputStream @ 0xa8d7db0 »	16	40
java.io.FileInputStream @ 0xa8d7800 »	16	40
java.io.FileInputStream @ 0xa8d73f8 »	16	40
java.io.FileInputStream @ 0xa8d73b0 »	16	40
java.io.FileInputStream @ 0xa8d7308 »	16	40
java.io.FileInputStream @ 0xa8d7260 »	16	40
java.io.FileInputStream @ 0xa8d7198 »	16	40
java.io.FileInputStream @ 0xa8d7088 »	16	40
java.io.FileInputStream @ 0xa8d6fa8 »	16	40
java.io.FileInputStream @ 0xa8d6dc8 »	16	40
∑ Total: 10 entries		

Objects ready for Finalization in Processing Order

Finalizer Thread

Class Name	Shallow Heap	Retained Heap
java.lang.ref.Finalizer$FinalizerThread @ 0xa8c3328 Finalizer **Native Stack, Thread** »	88	168

The Finalizer Thread Object

Finalizer Thread Locals

Class Name	Shallow Heap	Retained Heap
java.io.FileInputStream @ 0xa8d73f8 »	16	40

Object currently processed

Agenda

- Automation for Technical **Support** Staff
  - Leak Suspects
  - Memory Intensive Threads
- Automation for **Developers**
- **Questions & Answers**

Infos & Download

Eclipse

http://www.eclipse.org/mat/

Forum

eclipse.technology.memory-analyzer

(Old WIKI @ SAP)

- Download, Screen Cam, FAQ

https://www.sdn.sap.com/irj/sdn/wiki?path=/display/Java/Java+Memory+Analysis

Legal Notices

- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. In the United States, other countries, or both.
- Other company, product, or service names may be trademarks or service marks of others.

