

Eclipse Memory Analyzer

... and other Java stuff

Jan Rehwaltdt

1. Juli 2013

JVM Tool Interface

PROFILING IN THE JVM

JVM Tool Interface

- Comprehensive interface for programming and monitoring tools ($\geq$ JDK 4; before: JVM PI + DI)
- Allows inspecting state and control execution of applications run in the JVM
- Support for profiling, debugging, monitoring, thread analysis, and coverage analysis tools, ...

Profiling in the JVM

- JDK tools
 - jmap, jinfo, jps, jstack, jconsole, jvisualvm, ...
- Comprehensive Profiling Tools
 - VisualVM (free; included with $\geq$ JDK 6u7; lightweight)
 - NetBeans profiler (free; embedded in IDE)
 - YourKit (free for open-source projects)
 - JProfiler (free for open-source projects)
 - JProbe, Spring Insight, ... <http://java-source.net/open-source/profilers>

Profiling in the JVM: VisualVM

The screenshot shows the VisualVM Profiler interface. At the top, there are tabs for Overview, Monitor, Threads, Sampler, Profiler, Visual GC, and Tracer. The Profiler tab is active, showing a 'Profile:' section with buttons for CPU, Memory, and Stop. The status indicates 'profiling running (1.817 methods instrumented)'. Below this, there are 'Profiling results' and 'CPU settings' tabs. The 'CPU settings' tab is active, showing a list of 'Hot Spots - Method' with columns for 'Self time ...', 'Self time', and 'I...'. The list includes methods like 'org.eclipse.mat.ui.snapshot.OpenSnapshot\$Visitor.go' and 'org.eclipse.mat.impl.chart.ChartBuilder.create'. On the right, there is a 'Start profiling from classes:' section with a text area containing 'org.eclipse.mat.**' and a checkbox for 'Profile new Runtimes'.

Profile: CPU Memory Stop

Status: profiling running (1.817 methods instrumented)

Profiling results

CPU settings Memory settings

Start profiling from classes:

```
org.eclipse.mat.**
```

☒ Profile new Runtimes

☒ Profile only classes: ☐ Do not

```
org.eclipse.mat.*
```

Preset: Custom

Hot Spots - Method	Self time ...	Self time	I...
org.eclipse.mat.ui.snapshot.OpenSnapshot\$Visitor.go (org.eclipse.swt.wi...	392...	(37,9%)	1
org.eclipse.mat.impl.chart.ChartBuilder.create (org.eclipse.mat.query.IR...	857...	(8,3%)	1
org.eclipse.mat.ui.internal.chart.ChartCanvas.generateChartState ()	658...	(6,4%)	1
org.eclipse.ui.progress.UIJob\$1.run ()	539...	(5,2%)	27
org.eclipse.mat.ui.snapshot.editor.HeapEditor.createInitialPanels ()	398...	(3,9%)	1
org.eclipse.mat.query.registry.QueryRegistry.createDelegate (org.edi...	318...	(3,1%)	59
org.eclipse.mat.ui.snapshot.actions.OpenSnapshotAction\$1.visit (org.edi...	267...	(2,6%)	1
org.eclipse.mat.ui.internal.chart.ChartCanvas.<init> (org.eclipse.swt.wi...	202...	(2%)	1
org.eclipse.mat.collect.HashMapIntObject.readObject (java.io.ObjectIn...	197...	(1,9%)	10
org.eclipse.mat.ui.editor.MultiPaneEditor.updateToolBar ()	178...	(1,7%)	2
org.eclipse.ui.internal.ViewIntroAdapterPart\$2.run ()	140...	(1,4%)	1
org.eclipse.mat.ui.internal.GettingStartedWizard\$ChoicePage.createContr...	133...	(1,3%)	1
org.eclipse.mat.ui.snapshot.views.inspector.InspectorView\$8.run ()	122...	(1,2%)	2
org.eclipse.mat.query.registry.QueryRegistry.registerQuery (org.edips...	115...	(1,1%)	59
org.eclipse.mat.ui.internal.chart.PieChartPane.initWithArgument (Object)	111...	(1,1%)	1
org.eclipse.mat.snapshot.registry.ClassSpecificNameResolverRegistry\$Registr...	110...	(1,1%)	21
org.eclipse.mat.ui.SnapshotHistoryService.initializeDocument ()	105...	(1%)	1

Profiling in the JVM: VisualVM

The screenshot displays the VisualVM Profiler interface. At the top, a toolbar includes buttons for Overview, Monitor, Threads, Sampler, Profiler, Visual GC, and Tracer. Below this, a red banner reads "Profiling using instrumentation". The main window is titled "Profiling results" and has tabs for "CPU settings" and "Memory settings". On the left, a "Hot Spots - Method" table lists methods with their self-time and count. On the right, a text area shows "Start profiling from classes:" with the package `org.eclipse.mat.**` entered. A checkbox "Profile new Runnables" is checked. At the bottom, a red banner reads "Package-level restriction".

Hot Spots - Method	Self time ... ▼	Self time	I...
org.eclipse.mat.ui.snapshot.OpenSnapshot\$Visitor. go (org.eclipse.swt.wi...		392... (37,9%)	1
org.eclipse.mat.impl.chart.ChartBuilder. create (org.eclipse.mat.query.IR...		857... (8,3%)	1
org.eclipse.mat.ui.internal.chart.ChartCanvas. generateChartState ()		658... (6,4%)	1
org.eclipse.ui.progress.UIJob\$1. run ()		539... (5,2%)	27
org.eclipse.mat.ui.snapshot.editor.HeapEditor. createInitialPanels ()		398... (3,9%)	1
org.eclipse.mat.query.registry.QueryRegistry. createDelegate (org.edi...		318... (3,1%)	59
org.eclipse.mat.ui.snapshot.actions.OpenSnapshotAction\$1. visit (org.edi...		267... (2,6%)	1
org.eclipse.mat.ui.internal.chart.ChartCanvas. <init> (org.eclipse.swt.wi...		202... (2%)	1
org.eclipse.mat.collect.HashMapIntObject. readObject (java.io.ObjectIn...		197... (1,9%)	10
org.eclipse.mat.ui.editor.MultiPaneEditor. updateToolBar ()		178... (1,7%)	2
org.eclipse.ui.internal.ViewIntroAdapterPart\$2. run ()		140... (1,4%)	1
org.eclipse.mat.ui.internal.GettingStartedWizard\$ChoicePage. createContr		132... (1,3%)	1
org.eclipse.mat.ui.snapshot.views.inspector.InspectorView\$8. run ()			
org.eclipse.mat.query.registry.QueryRegistry. registerQuery (org.edips...			
org.eclipse.mat.ui.internal.chart.PieChartPane. initWithArgument (Object)		111... (1,1%)	1
org.eclipse.mat.snapshot.registry.ClassSpecificNameResolverRegistry\$Registr		110... (1,1%)	21
org.eclipse.mat.ui.SnapshotHistoryService. initializeDocument ()		105... (1%)	1

Start profiling from classes:

```
org.eclipse.mat.**
```

☒ Profile new Runnables

Preset: Custom

Profiling in the JVM: VisualVM

The screenshot shows the VisualVM Overview tab with the CPU profile selected. The status indicates 'profiling running (1.817 methods instrumented)'. The 'Hot Spots - Method' table lists the most active methods, and the 'Memory settings' panel on the right shows 'Profile new Runnables' checked and 'Profile only classes' set to 'org.eclipse.mat.*'.

Profile: CPU Memory

Status: profiling running (1.817 methods instrumented)

Profiling results

Hot Spots - Method

Method	Count	Percentage	Self Time
org.eclipse.mat.ui.snapshot.OpenSnapshot\$Visitor.go (org.e...	267...	(2,6%)	1
org.eclipse.mat.impl.chart.ChartBuilder.create (org.eclipse...	202...	(2%)	1
org.eclipse.mat.ui.internal.chart.ChartCanvas.generateCh...	197...	(1,9%)	10
org.eclipse.ui.progress.UIJob\$1.run ()	178...	(1,7%)	2
org.eclipse.mat.ui.snapshot.editor.HeapEditor.createInitia...	140...	(1,4%)	1
org.eclipse.mat.query.registry.QueryRegistry.createDeleg...	133...	(1,3%)	1
org.eclipse.mat.ui.snapshot.actions.OpenSnapshotAction\$1.vi...	122...	(1,2%)	2
org.eclipse.mat.ui.internal.chart.ChartCanvas.<init> (org.eclipse.swt.wi...	115...	(1,1%)	59
org.eclipse.mat.collect.HashMapIntObject.readObject (java.io.ObjectIn...	111...	(1,1%)	1
org.eclipse.mat.ui.editor.MultiPaneEditor.updateToolBar ()	110...	(1,1%)	21
org.eclipse.ui.internal.ViewIntroAdapterPart\$2.run ()	105...	(1%)	1
org.eclipse.mat.ui.internal.GettingStartedWizard\$ChoicePage.createContr...			
org.eclipse.mat.ui.snapshot.views.inspector.InspectorView\$8.run ()			
org.eclipse.mat.query.registry.QueryRegistry.registerQuery (org.edips...			
org.eclipse.mat.ui.internal.chart.PieChartPane.initWithArgument (Object)			
org.eclipse.mat.snapshot.registry.ClassSpecificNameResolverRegistry\$Registr...			
org.eclipse.mat.ui.snapshot.registry.ClassSpecificNameResolverRegistry\$Registr...			
org.eclipse.mat.ui.SnapshotHistoryService.initializeDocument ()			

memory settings

n classes:

.mat.**

☒ Profile new Runnables

☒ Profile only classes: ☐ Do n

org.eclipse.mat.*

Preset: Custom

9

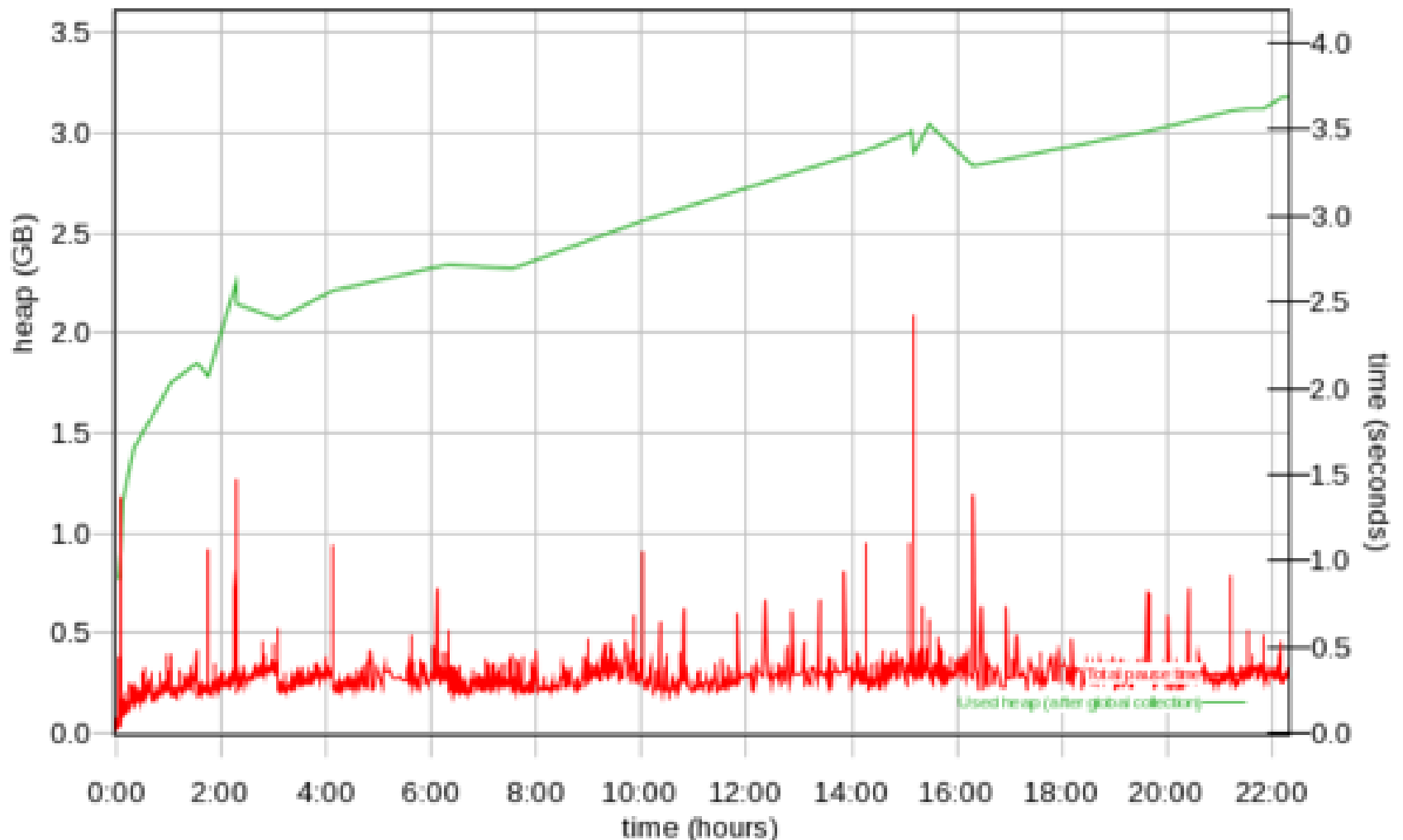
- ✓ Statistics (JVM, GC, ...)
- ✓ Thread Monitoring
- ✓ Sampling
- ✓ Profiling
- ✓ Tracer (BTrace)
- ✓ Heap Dump Analysis

ECLIPSE MEMORY ANALYZER

Eclipse Memory Analyzer

- Show memory structure of Java programs
- Not while runtime, but generally based on heap dumps
 - often post-mortem
 - hprof file generated by Oracle JVM (experimental format)
 - phd file generated by IBM JVM
- Donated by SAP; Contributed to by IBM

Eclipse Memory Analyzer – when?



Heap dump

- **jmap** (included in JDK)
 - `jmap -dump:format=b <pid>`
- **Acquire heap dump** (inside Eclipse Memory Analyzer; uses jmap)
- **On OutOfMemory** (flag: `-XX:+HeapDumpOnOutOfMemoryError`)
- **On [Ctrl]+[Break]** (flag: `-XX:+HeapDumpOnCtrlBreak`; $\geq$ JDK 5u7)

Eclipse Memory Analyzer – when?

- Helps:
 - Find memory wasting datastructures
 - Indicate misuse of Java datatypes
 - Find memory leaks
 - Improve code by checking impacts of refactorings by comparing heap dumps
 - Understand memory consumption of code

Eclipse Memory Analyzer – when not?

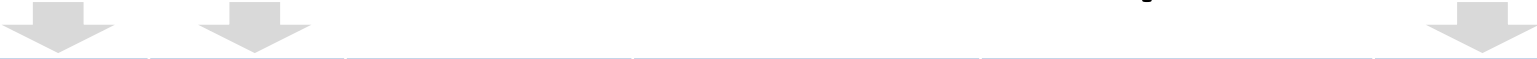
- Does not help:
 - Threading issues, wrong / missing variable assignments and other debuggable problems: No debugging or stepping
 - Where has an object been created?
 - Analyze JVM's memory footprint.

Demo: MAT Roundtrip

#ID	Sex	Culture	Forename	Surname	Address	...
1	M	Unk	Jan	Rehwaldt	Pusemukkel 4	...
2	M	Eng	James	Albright		...
3	F	Rus	Svetlana	Dyakonova	Newski Prospekt 106	...
4	M	Est	Epp	Sibul	Uus 20	...
5	F	Rus	Sveta	D.	Newskoi P. 106	...
6		Usa	Mr.	Nobel Prize	NSA Headquarter	...
7	J	Usa	Mr.	Obama	The White House	...
8		Eng	Jamie	Albritt	Unknown	...
9	M	Por	Sebastian	Serano	Rosa Carreras	...
10	F	Por	Ana	Abílio	Heaven 9	...
11	F	Ger	Jahn	Rehfeld	Pusem. 4a	...
12	F	Eng	Pita	Brown	Trafalgar Square 13	...
13	M	Unk	M			...
...	...	...	...	...	...	...
1Mio.	F	Ukr	Susanne	Mirkowitsch	Dom Spasibo ul.	

Address Deduplication

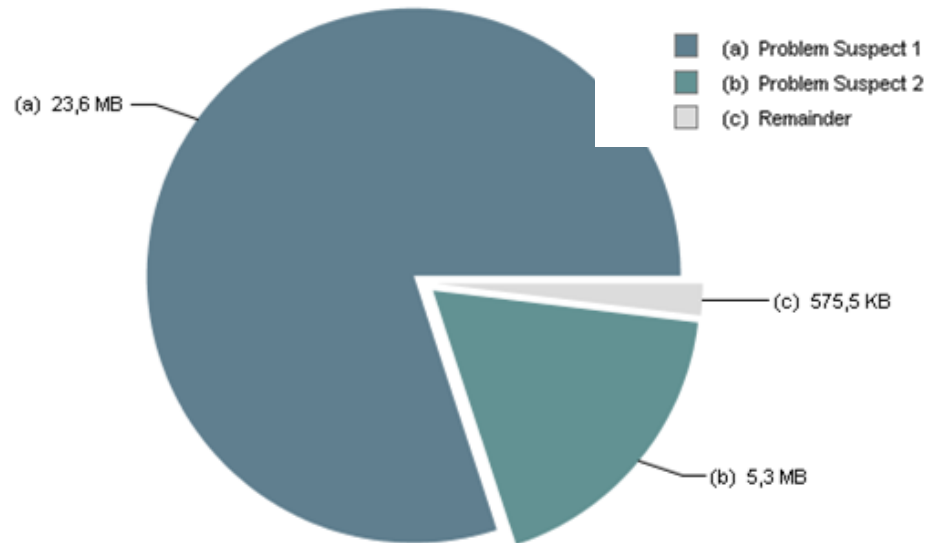
Demo: MAT Roundtrip



#ID	Sex	Culture	Forename	Surname	Address	...
1	M	Unk	Jan	Rehwaldt	Pusemukkel 4	...
2	M	Eng	James	Albright		...
3	F	Rus	Svetlana	Dyakonova	Newski Prospekt 106	...
4	M	Est	Epp	Sibul	Uus 20	...
5	F	Rus	Sveta	D.	Newskoi P. 106	...
6		Usa	Mr.	Nobel Prize	NSA Headquarter	...
7	J	Usa	Mr.	Obama	The White House	...
8		Eng	Jamie	Albritt	Unknown	...
9	M	Por	Sebastian	Serano	Rosa Carreras	...
10	F	Por	Ana	Abílio	Heaven 9	...
11	F	Ger	Jahn	Rehfeld	Pusem. 4a	...
12	F	Eng	Pita	Brown	Trafalgar Square 13	...
13	M	Unk	M			...
...	...	...	...	...	...	...
1Mio.	F	Ukr	Susanne	Mirkowitsch	Dom Spasibo ul.	

Grouping Approach

Leak Suspect



▼ ✖ Problem Su

Total: 29,4 MB

The thread **java.lang.Thread @ 0x26f105b8 main** keeps local variables with total size **24.705.288 (80,04%)** bytes.

The memory is accumulated in one instance of "**java.lang.Thread**" loaded by "**<system class loader>**".

The stacktrace of this Thread is available. [See stacktrace.](#)

Keywords

java.lang.Thread

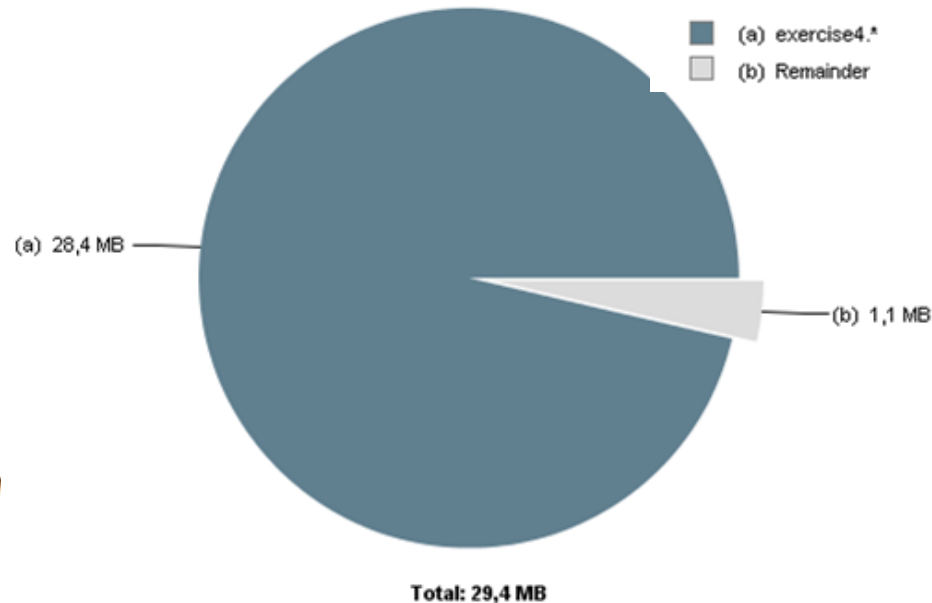
[Details »](#)

▼ ✖ Problem Suspect 2

Dominator Tree

Class Name	Shallow Heap	Retained Heap	Percentage
<Regex>	<Numeric>	<Numeric>	<Numeric>
java.lang.Thread @ 0x26f105b8 main Thread	104	24.705.288	80,04 %
java.util.HashMap\$KeyIterator @ 0x28bb0028	32	9.849.968	31,91 %
java.util.ArrayList @ 0x28ba7870	24	563.560	1,83 %
exercise4.partitions.WindowPartition @ 0x27a9b728	24	40.192	0,13 %
exercise4.partitions.GroupingPartition @ 0x279d4098	32	3.168	0,01 %
java.util.ArrayList @ 0x27a7a7b8	24	3.040	0,01 %
java.util.ArrayList\$SubList\$1 @ 0x28c51320	32	64	0,00 %
java.util.ArrayList\$ListItr @ 0x28c77fd0	32	32	0,00 %
Σ Total: 3 entries			
exercise4.partitions.GroupingPartition @ 0x279d4138	32	3.168	0,01 %
exercise4.partitions.GroupingPartition @ 0x27a7af68	32	3.168	0,01 %
exercise4.partitions.GroupingPartition @ 0x279d3d98	32	2.168	0,01 %
exercise4.partitions.GroupingPartition @ 0x279d3db8	32	2.168	0,01 %
exercise4.partitions.GroupingPartition @ 0x279d3dd8	32	2.168	0,01 %
exercise4.partitions.GroupingPartition @ 0x279d4038	32	2.168	0,01 %
Σ Total: 25 of 13.279 entries; 13.254 more			
class java.lang.System @ 0x38b068b8 System Class	32	60.192	0,20 %
sun.misc.Launcher\$AppClassLoader @ 0x26f100e8	80	53.576	0,17 %
java.util.jar.JarFile @ 0x26f55010	56	38.864	0,13 %
class java.lang.CharacterData00 @ 0x28c0ccf8 System Class	32	30.992	0,10 %
class java.nio.charset.Charset @ 0x38b0b498 System Class	40	14.712	0,05 %
class java.io.File @ 0x28b0d8e8 System Class	32	11.548	0,04 %

Component Report



▼ Possible Memory

▼ Duplicate Strings

Found 583 occurrences of `char[]` with at least 10 instances having identical content. Total size is 1.072.464 bytes.

Top elements include:

- 13.628 × (16 bytes)
- 4.453 × `f` (16 bytes)
- 2.091 × `nsw` (24 bytes)
- 1.546 × `vic` (24 bytes)
- 1.363 × `usa` (24 bytes)

[Details »](#)

▼ Empty Collections

No excessive usage of empty collections found.

1. Juli 2013

[Details »](#)

Possible Memory Waste

▼ Possible Memory Waste

▼ Duplicate Strings

Found 583 occurrences of `char[]` with at least 10 instances having identical content. Total size is 1.072.464 bytes.

Top elements include:

- 13.628 × (16 bytes)
- 4.453 × **f** (16 bytes)
- 2.091 × **nsw** (24 bytes)
- 1.546 × **vic** (24 bytes)
- 1.363 × **usa** (24 bytes)

[Details »](#)

▼ Empty Collections

No excessive usage of empty collections found.

[Details »](#)

▼ Collection Fill Ratios

Detected the following collections with fill ratios below 20%:

- 8.265 instances of **java.util.ArrayList** retain **>= 16.860.600** bytes.

[Details »](#)

▼ Miscellaneous

1. Juli 2013

▼ Soft Reference Statistics

Object Query Language (OQL)

```
SELECT *  
FROM exercise4.partitions.GroupingPartition p  
WHERE (p.partition.size <= p.partition.elementData.@length * 0.20)
```

**All GroupingPartitions
with a partition filled max. 20%**

Class Name			
 exercise4.pa			
 exercise4.partitions.GroupingPartition @ 0x28b8fe00	32	2.072	
 exercise4.partitions.GroupingPartition @ 0x28b8f5e8	32	2.168	
 exercise4.partitions.GroupingPartition @ 0x28b8edd0	32	2.072	
 exercise4.partitions.GroupingPartition @ 0x28b8e5b8	32	2.072	
 exercise4.partitions.GroupingPartition @ 0x28b8dda0	32	2.072	
 exercise4.partitions.GroupingPartition @ 0x28b8d588	32	2.072	
 exercise4.partitions.GroupingPartition @ 0x28b8cd70	32	2.072	
 exercise4.partitions.GroupingPartition @ 0x28b8c558	32	2.072	
 exercise4.partitions.GroupingPartition @ 0x28b8bd40	32	2.072	
 exercise4.partitions.GroupingPartition @ 0x28b8b528	32	2.072	
 exercise4.partitions.GroupingPartition @ 0x28b8ad10	32	2.072	
 Total: 31 of 8.265 entries; 8.234 more			

Pros and Cons

- Pros
 - Very (most?!) powerful and comprehensive
 - Provides hints for memory wastes
- Cons
 - Very complicated handling
 - A lot of memory required

Sources

- <http://eclipsesource.com/blogs/2013/01/21/10-tips-for-using-the-eclipse-memory-analyzer/>
- <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>
- MAT Tutorials & Cheat Sheet
- Trail & Error