1. Heap Statistics

Memory wasted: 7.86mb (31%)

Duplicate Primitive Arrays 1.36mb (5.4%)

Inefficient Object Arrays 1.11mb (4.4%)

Inefficient Collections 1.96mb (7.8%)

Duplicate Strings 684.18kb (2.6%)

Duplicate Objects 498.69kb (1.9%)

Inefficient Primitive Arrays 2.25mb (8.9%)

Objects waiting finalization 48b (<0.1%)

Total Size	Object Count
25.28mb	514,429

Class Count	Thread Count
14,119	38

2. What's in your Memory (by class)?

Class	Percentage	Size	Count
String	29.5%	7.47mb	73,593

char[] 🔗	10.1%	2.56mb	52,681
java.lang.reflect.Method 🔗	7.8%	1.98mb	23,564
java.util.concurrent.ConcurrentHashMap 🔗	7.7%	1.94mb	698
j.u.LinkedHashMap 🔗	4.0%	1.01mb	9,652
byte[] 🔗	3.0%	782.72kb	3,890

Show all records »
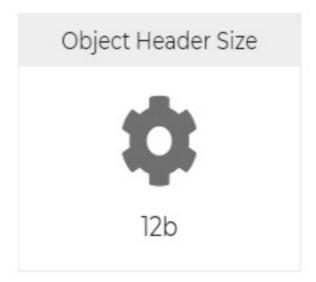
3. Large objects
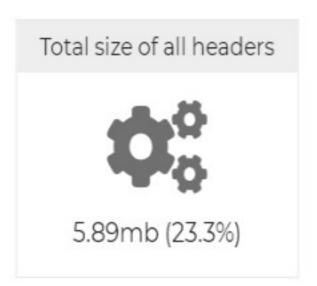
Learn more about Large Objects

| Name	Percentage	Size
Java Static java.lang.ApplicationShutdownHooks.hooks 🔗	21.6%	5.45mb
Unreachable (garbage) objects 🔗	17.8%	4.51mb
Java Static org.apache.catalina.core.StandardHostValve.MY_CLASSLOADER 🔗	13.6%	3.44mb
Java Static org.springframework.data.jpa.util.JpaMetamodel.CACHE 🔗	2.6%	677.8kb
Java Static org.apache.tomcat.util.modeler.Registry.registry 🔗	2.4%	625.47kb
… and 11304 more objects retaining 3.42mb (13.5%)		

4. Object Headers

Learn more about Object Headers

Object Header Size	Total size of all headers
12b	5.89mb (23.3%)

Top Object Headers
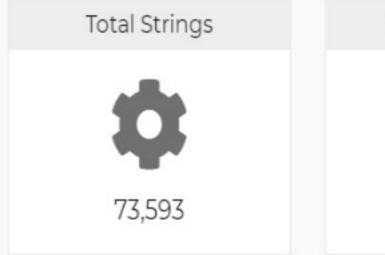
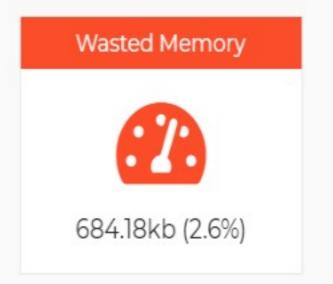| Class	Percentage	Total header size	Avg obj size	Count
String	3.3%	862.42kb	24	73,593
char[]	2.4%	617.36kb	166	52,681
java.util.concurrent.ConcurrentHashMap$Node	2.2%	576.26kb	32	49,174
Object	1.4%	374.03kb	16	31,917
j.l.Class[]	1.1%	287.93kb	22	24,570

🔧 How to fix excessive Object headers?

Please refer to our recommendations

5. Duplicate Strings

Learn more about Duplicate Strings

Total Strings	Unique Strings	Wasted Memory
⚙️	📄	⏱️
73,593	49,756	684.18kb (2.6%)

💡 Top Duplicate Strings

Duplicate String	Percentage	Wasted	Count
""	<0.1%	16.41kb	669
"java.lang.String"	<0.1%	15.61kb	611
"java.lang.Object"	<0.1%	12.91kb	518
"value"	<0.1%	9.71kb	386
"org.springframework.context.annotation.Bean"	<0.1%	8.27kb	211

⑦ Who is holding Duplicate Strings?

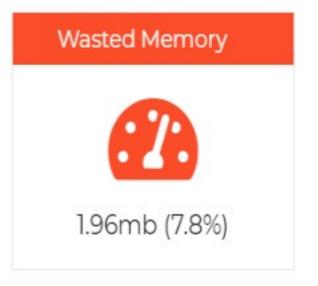| Object Tree	Percentage	size
java.lang.reflect.Method.signature ⬈	<0.1%	17K

🔧 How to fix Duplicate Strings?

Please refer to our recommendations

6. Inefficient collections

Learn more about Inefficient Collections

Total Collections	Inefficient collections	Wasted Memory
⚙	⚠	🎛
25,964	20,402	1.96mb (7.8%)

💡 Top inefficient collections

Problem	Percentage	Wasted

22% of j.u.LinkedHashMap contains 1 element only	1.3%	345.82kb
11% of j.u.LinkedHashMap contains 2 - 4 elements only	0.8%	207.8kb
28% of j.u.LinkedHashSet contains 1 element only	0.6%	167.2kb
45% of j.u.ArrayList contains no elements	0.6%	160.41kb
44% of j.u.ArrayList contains 1 element only	0.6%	159.71kb

Show all records »

⑦ Who is holding Inefficient Collections?

Object Tree	Percentage	size
org.springframework.boot.autoconfigure.condition.ConditionEvaluationReport$ConditionAndOutcomes.outcomes ⬈	0.3%	75K
j.l.Class$AnnotationData.annotations ⬈	0.3%	68K
java.lang.reflect.Method.declaredAnnotations ⬈	0.2%	47K
java.lang.reflect.Field.declaredAnnotations ⬈	0.1%	33K
Unreachable (garbage) objects ⬈	0.1%	31K

Show all records »

🔧 How to fix Inefficient Collections?
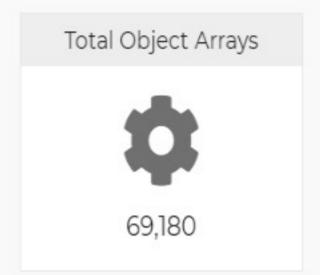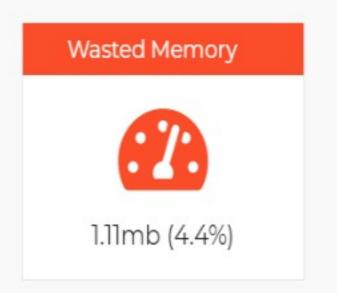
Please refer to our recommendations.

7. Inefficient Object Arrays

Learn more about Inefficient Object Arrays

Total Object Arrays	Inefficient Object Arrays	Wasted Memory
69,180	42,000	1.11mb (4.4%)

⚟ Top inefficient Object Arrays

Problem	Percentage	Wasted
47% of j.l.Class[] declared with 1 length	1.1%	273.05kb
33% of j.l.Class[] declared with 0 length	0.5%	128.91kb
13% of Object[] contains no elements	0.4%	97.7kb
9% of j.u.HashMap$Node[] contains 1 element only	0.2%	60.77kb
10% of Object[] contains 1 element only	0.2%	52.79kb

Show all records >>
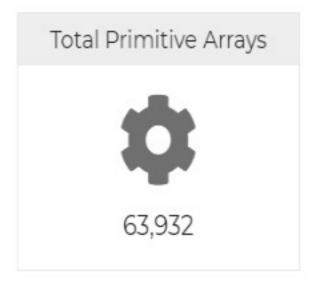
⦿ Who is holding Inefficient Object Arrays?

Object Tree	Percentage	size
Unreachable (garbage) objects ⬈	0.7%	189K
java.lang.reflect.Method.parameterTypes ⬈	0.3%	80K
java.lang.reflect.Method.parameterTypes ⬈	0.2%	50K
java.lang.reflect.Method.parameterTypes ⬈	0.2%	47K
j.l.Class$ReflectionData.declaredConstructors ⬈	0.1%	31K

Show all records »

🔧 How to fix Inefficient Object Arrays?
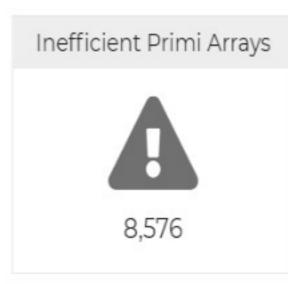
Please refer to our recommendations.

8. Inefficient Primitive Arrays

Learn more about Inefficient Primitive Arrays

Total Primitive Arrays	Inefficient Primi Arrays	Wasted Memory
⚙	⚠	🕑
63,932	8,576	2.25mb (8.9%)

📍 Top inefficient Primitive Arrays

🔧 Top Inefficient Primitive Arrays

Problem	Percentage	Wasted
< 0.1% of char[] contains no elements	5.9%	1.5mb
46% of byte[] contains lot of 0s	1.4%	353.4kb
6% of int[] contains no elements	0.4%	101.8kb
16% of int[] contains lot of 0s	0.3%	73.01kb
59% of int[] declared with 0 length	0.2%	63.58kb

Show all records >>
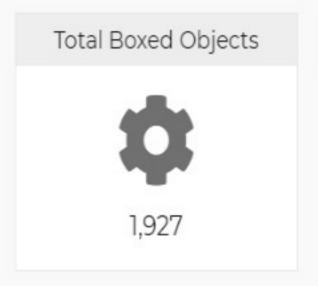
❓ Who is holding Inefficient Primitive Arrays?

Object Tree	Percentage	size
static.sun.nio.cs.ext.IBM1364.c2b 🔗	0.3%	78K
static.sun.nio.cs.ext.IBM933.c2b 🔗	0.3%	77K
java.io.BufferedOutputStream.buf 🔗	0.3%	75K
java.io.BufferedOutputStream.buf 🔗	0.3%	75K
com.mysql.cj.protocol.ReadAheadInputStream.buf 🔗	0.3%	74K

Show all records >>

🔧 How to fix Inefficient Primitive Arrays?
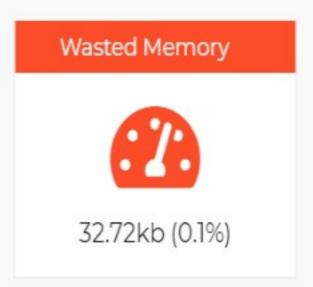
Please refer to our recommendations

9. Boxed Numbers

Learn more about [Boxed Numbers](#)

Total Boxed Objects	Wasted Memory
⚙	🎛
1,927	32.72kb (0.1%)
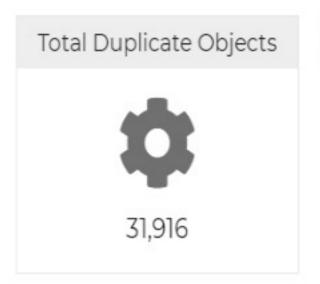
⍰ Who is holding Boxed Numbers?
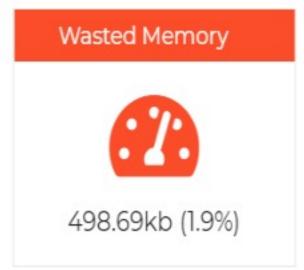
Object Tree	Percentage	size
j.l.Byte[] ↗	<0.1%	4K
j.l.Long[] ↗	<0.1%	4K
j.l.Short[] ↗	<0.1%	4K

🔧 How to fix Boxed Numbers?

Please refer to our [recommendations](#)

10. Duplicate Objects

Learn more about Duplicate Objects

Total Duplicate Objects	Wasted Memory
⚙	🕐
31,916	498.69kb (1.9%)

Types of Duplicate Objects

Object	Percentage	Wasted	Duplicate Count
Object	1.9%	498.69kb	31,916

💡 Top Duplicate Objects

Duplicate Object	Percentage	Wasted	Count
Objec)	1.9%	498.69kb	31,917

❓ Who is holding Duplicate Objects?

Object Tree	Percentage	size
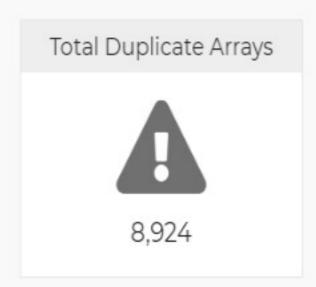
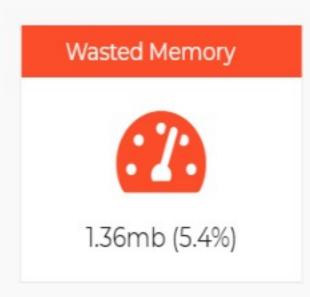| Object Tree	Percentage	size
{java.util.concurrent.ConcurrentHashMap} values ⬚	0.6%	156K
{java.util.concurrent.ConcurrentHashMap} values ⬚	0.6%	156K
{java.util.concurrent.ConcurrentHashMap} values ⬚	0.6%	155K

🔧 How to fix Duplicate Objects?

Please refer to our recommendations

---

11. Duplicate Primitive Arrays

Learn more about Duplicate Primitive Arrays

Total Duplicate Arrays	Wasted Memory
⚠️	🕐
8,924	1.36mb (5.4%)

☰ Types of Duplicate Arrays

Array Type	Percentage	Wasted	Duplicate Count
char[]		674.11kb	232

char[]	2.6%	971.11kb	252
byte[]	1.7%	446.87kb	2,545
int[]	0.9%	242.92kb	5,761
boolean[]	<0.1%	20.05kb	353
long[]	<0.1%	3.22kb	14

Show all records >>

💡 Top Duplicate Arrays

Duplicate Array	Percentage	Wasted	Count
byte[16384](1, 44, 0, 0, 2, 3, 100, 101, 102, 0, 0, 0, 22, 64, 64, 115, 101, 115, 115, 105, ...)	0.5%	128.12kb	9
byte[16384](30, 0, 0, 0, 3, 83, 69, 76, 69, 67, 84, 32, 64, 64, 115, 101, 115, 115, 105, 111, ...)	0.5%	128.12kb	9
char[32512](,.......................)	0.5%	127.03kb	3
char[29696](,.......................)	0.4%	116.03kb	3
char[27392](,.......................)	0.4%	107.03kb	3

Show all records >>

❓ Who is holding Duplicate Arrays?

Object Tree	Percentage	size
com.mysql.cj.protocol.ReadAheadInputStream.buf 🔗	0.3%	71K
java.io.BufferedOutputStream.buf 🔗	0.3%	71K
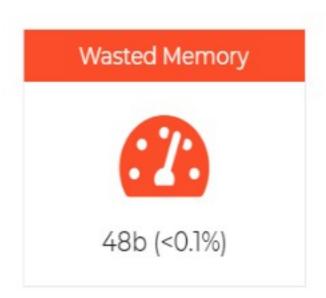
Unreachable (garbage) objects ☑	0.2%	63K
com.mysql.cj.protocol.ReadAheadInputStream.buf ☑	0.2%	56K
java.io.BufferedOutputStream.buf ☑	0.2%	56K

Show all records »

🔧 How to fix Duplicate Arrays?

Please refer to our recommendations.

12. Objects waiting for Finalization

Learn more about Objects waiting for Finalization

Wasted Memory

48b (<0.1%)

⍰ What are the objects waiting for finalization?

To see objects waiting for finalization, click here ☑

🔧 How to fix objects waiting for finalization?

🔧 How to fix objects waiting for finalization?

Please refer to our recommendations

---

13. Threads

Learn more about Threads

To view all threads stacktrace click here ☑

---

14. Heap settings

Learn more about Heap Settings

No major recommendations.

---

15. System Properties

Learn more about System Properties

Key	Value

PID	39600
awt.toolkit	sun.awt.windows.WToolkit
catalina.base	C:\Users\Caree2\AppData\Local\Temp\tomcat.5002208786504499207.9095
catalina.home	C:\Users\Caree2\AppData\Local\Temp\tomcat.15809530l0250039306.6094
catalina.useNaming	false

Show all records ››