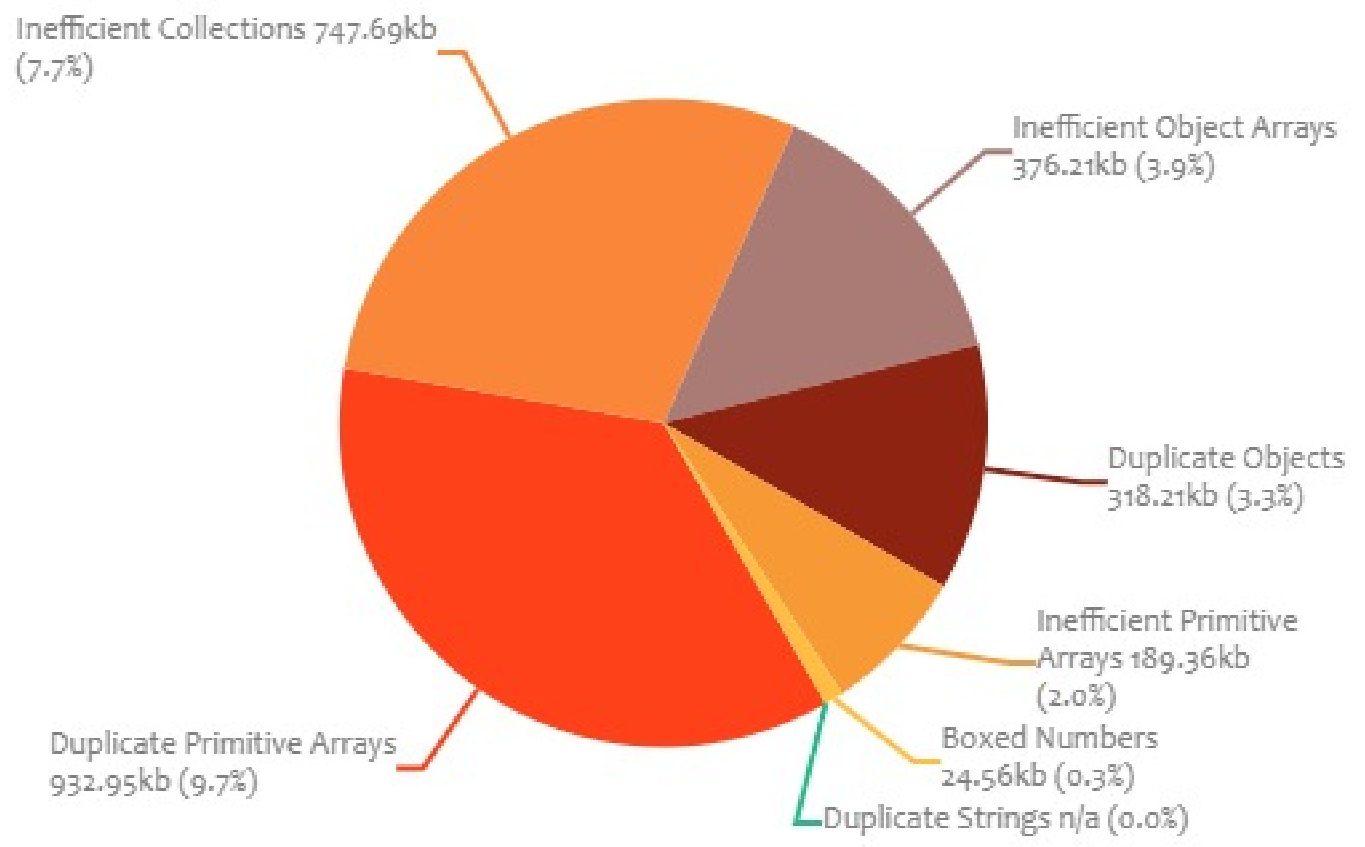
1. Heap Statistics

Learn more about Heap Statistics

Memory wasted: 2.53mb (27%)



Total Size	Object Count
9.43mb	244,037

Class Count	Thread Count
7,726	24

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

Learn more about What's in Memory

Class	Percentage	Size	Count
String	32.9%	3.11mb	40,886

java.util.concurrent.ConcurrentHashMap	10.7%	1.01mb	510
byte[]	3.9%	372.22kb	42,297
Object[]	3.6%	351.36kb	7,730
j.u.HashMap	3.5%	340.24kb	1,227
j.u.LinkedHashMap	3.2%	307.16kb	2,675

Show all records >>

3. Large objects

Learn more about Large Objects

Name	Percentage	Size
Java Static java.lang.ApplicationShutdownHooks.hooks	12.1%	1.14mb
Unreachable (garbage) objects	10.3%	997.52kb
Java Static org.apache.catalina.core.StandardHostValve.MY_CLASSLOADER	8.5%	821.25kb
Java Static org.apache.tomcat.util.modeler.Registry.registry	5.2%	502.78kb
Java Static jdk.internal.loader.ClassLoaders.APP_LOADER	5.0%	485.22kb
... and 10619 more objects retaining 3.91mb (41.4%)		

Show all records >>

4. Object Headers

Learn more about Object Headers

Object Header Size	Total size of all headers
12b	2.79mb (29.6%)

Top Object Headers

Class	Percentage	Total header size	Avg obj size	Count
byte[]	5.1%	495.67kb	62	42,297
String	5.0%	479.13kb	24	40,886
java.util.concurrent.ConcurrentHashMap$Node	3.0%	289.52kb	32	24,706
Object	1.7%	166.92kb	16	14,244
j.u.HashMap$Node	1.4%	132.45kb	32	11,302

Show all records >>

How to fix excessive Object headers?

Please refer to our recommendations

5. Duplicate Strings

Learn more about Duplicate Strings

Not Detected

6. Inefficient collections

Learn more about Inefficient Collections

Total Collections	Inefficient collections	Wasted Memory
9,289	7,366	747.69kb (7.7%)

Top inefficient collections

Problem	Percentage	Wasted

20% of j.u.LinkedHashMap contains 1 element only	0.9%	86.05kb
59% of j.u.ArrayList contains 1 element only	0.9%	84.24kb
13% of j.u.LinkedHashMap contains 2 - 4 elements only	0.7%	70.02kb
35% of j.u.LinkedHashSet contains 1 element only	0.6%	61.32kb
9% of java.util.concurrent.ConcurrentHashMap contains half empty elements	0.4%	42.27kb

Show all records >>

Who is holding Inefficient Collections?

Object Tree	Percentage	size
org.springframework.boot.autoconfigure.condition.ConditionEvaluationReport$ConditionAndOutcomes.outcomes	0.4%	36K
{j.u.HashMap}.values	0.3%	25K
j.l.Class$AnnotationData.annotations	0.3%	25K
sun.reflect.generics.tree.ClassTypeSignature.path	0.2%	22K
sun.reflect.generics.tree.ClassTypeSignature.path	0.2%	19K

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
22,811	11,979	376.21kb (3.9%)

Top inefficient Object Arrays

Problem	Percentage	Wasted
17% of Object[] contains no elements	0.6%	59.24kb
2% of Object[] contains half empty elements	0.4%	34.2kb
34% of j.l.Class[] declared with 1 length	0.3%	28.41kb
13% of Object[] declared with 1 length	0.3%	25.2kb
7% of Object[] contains 1 element only	0.2%	23.77kb

Show all records >>

Who is holding Inefficient Object Arrays?

Object Tree	Percentage	size

org.springframework.util.ConcurrentReferenceHashMap$Segment.references	0.2%	18K
Unreachable (garbage) objects	0.2%	18K
org.springframework.core.annotation.AnnotationAttributes.table	0.1%	13K

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
45,759	2,683	189.36kb (2.0%)

Top inefficient Primitive Arrays

Problem	Percentage	Wasted
< 0.1% of byte[] contains no elements	0.4%	40.55kb

1% of int[] contains no elements	0.4%	34.45kb
70% of int[] declared with 0 length	0.3%	32kb
12% of char[] contains no elements	0.2%	23.79kb
< 0.1% of byte[] contains lot of 0s	0.2%	23.31kb

Show all records >>

Who is holding Inefficient Primitive Arrays?

Object Tree	Percentage	size
java.io.BufferedWriter.cb	0.3%	31K
java.nio.HeapByteBuffer.hb	0.2%	15K
byte[]	0.2%	14K

How to fix Inefficient Primitive Arrays?

Please refer to our recommendations.

9. Boxed Numbers

Learn more about Boxed Numbers

Total Boxed Object

Wasted M

Total Boxed Objects

1,399

Wasted Memory

24.56kb (0.3%)

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

16,692

318.21kb (3.3%)

Types of Duplicate Objects

Object	Percentage	Wasted	Duplicate Count
Object	2.3%	222.55kb	14,243
j.l.r.SoftReference	1.0%	95.66kb	2,449

Top Duplicate Objects

Duplicate Object	Percentage	Wasted	Count
Objec)	2.3%	222.55kb	14,244
j.l.r.SoftReference(referent : null, queue : j.l.r.ReferenceQueue$Null@fec35ed8, next : null, discovered : null, timestamp : 92689085)	0.7%	63.05kb	1,615
j.l.r.SoftReference(referent : null, queue : j.l.r.ReferenceQueue$Null@fec35ed8, next : null, discovered : null, timestamp : 92689085)	<0.1%	4.1kb	106
j.l.r.SoftReference(referent : null, queue : j.l.r.ReferenceQueue$Null@fec35ed8, next : null, discovered : null, timestamp : 92689085)	<0.1%	1.8kb	47
j.l.r.SoftReference(referent : null, queue : j.l.r.ReferenceQueue$Null@fec35ed8, next : null, discovered : null, timestamp : 92689085)	<0.1%	1.09kb	29

Show all records >>

Who is holding Duplicate Objects?

Object Tree	Percentage	size
{java.util.concurrent.ConcurrentHashMap}.values	0.7%	70K
{java.util.concurrent.ConcurrentHashMap}.values	0.7%	70K
{java.util.concurrent.ConcurrentHashMap}.values	0.7%	64K
sun.util.locale.BaseLocale$Key.vart	0.3%	28K
sun.util.locale.BaseLocale$Key.scrt	0.3%	28K

Show all records >>

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
16,274	932.95kb (9.7%)

Types of Duplicate Arrays

Array Type	Percentage	Wasted	Duplicate Count
byte[]	8.3%	799.17kb	13,793
int[]	0.7%	67.16kb	2,210
char[]	0.6%	55.49kb	190
boolean[]	<0.1%	9.05kb	45
long[]	<0.1%	1.53kb	22

Show all records >>

Top Duplicate Arrays

Duplicate Array	Percentage	Wasted	Count
byte[8192](0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...)	0.3%	32.06kb	5
int[0]()	0.3%	31.98kb	2,048
int[1025](0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...)	0.2%	20.12kb	6
byte[16]('j', 'a', 'v', 'a', '.', 'l', 'a', 'n', 'g', '.', 'O', 'b', 'j', 'e', 'c', 't')	0.1%	9.94kb	319
int[512](0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...)	<0.1%	6.05kb	4

Show all records >>

Who is holding Duplicate Arrays?

Object Tree	Percentage	size
byte[][]	0.5%	48K
{java.util.concurrent.ConcurrentHashMap}.values	0.5%	48K
char[][]	0.3%	24K
char[][]	0.3%	24K

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?

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

Not Report in the Heap dump.