1. Heap Statistics

Learn more about Heap Statistics

Total Size : 388.83mb

Class Count : 13,168

Object Count : 7,976,573

Thread Count : 43

Memory wasted: 108.8mb (28%)

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

Learn more about What's in Memory

Class	Percentage	Size	Count
String	38.2%	148.72mb	1,573,401

byte[]	18.5%	71.76mb	2,760,561
j.l.StringBuilder	12.0%	46.73mb	87,765
com.google.protobuf.ByteString$LiteralByteString	3.7%	14.5mb	633,467
j.u.HashSet	3.2%	12.28mb	1,311
bisq.core.trade.statistics.TradeStatistics2	2.7%	10.57mb	106,574

Show all records >>

3. Large objects

Learn more about Large Objects

Name	Percentage	Size
Java Static bisq.desktop.main.MainView.rootContainer <<Object might be causing memory leak.>>	47.1%	182.99mb
Unreachable (garbage) objects	41.7%	162.09mb
Java Static bisq.core.provider.fee.FeeService.daoStateService	3.2%	12.28mb
Java Static bisq.desktop.components.BalanceWithConfirmationTextField.walletService	1.9%	7.58mb
Java Static [org.apache.commons.logging.LogFactory.thisClassLoader], Java Local(jdk.internal.loader.ClassLoaders$AppClassLoader) [@a3c498f0] (1 thread(s))	1.0%	3.78mb
... and 18238 more objects retaining 8.56mb (2.2%)		

Show all records >>

4. Object Headers

Learn more about Object Headers

Object Header Size	Total size of all headers
12b	91.28mb (23.5%)

Top Object Headers

Class	Percentage	Total header size	Avg obj size	Count
byte[]	8.1%	31.59mb	87	2,760,561
String	4.6%	18.01mb	24	1,573,401
com.google.protobuf.ByteString$LiteralByteString	1.9%	7.25mb	24	633,467
j.u.HashMap$Node	1.1%	4.36mb	32	380,614
java.util.concurrent.ConcurrentHashMap$Node	0.6%	2.16mb	32	189,123

Show all records >>

How to fix excessive Object headers?

To see our recommendations, please purchase Enterprise Edition.

5. Duplicate Strings

Learn more about Duplicate Strings

Not Detected

6. Inefficient collections

Learn more about Inefficient Collections

Total Collections	Inefficient collections	Wasted Memory
78,540	74,861	11.69mb (3.0%)

Top inefficient collections

Problem	Percentage	Wasted
99% of j.u.LinkedHashMap contains 1 element only	2.4%	9.36mb
5% of java.util.concurrent.ConcurrentHashMap contains half empty elements	0.2%	692.35kb

Who is holding Inefficient Collections?

Object Tree	Percentage	size
com.google.protobuf.MapField$MutatabilityAwareMap.delegate	2.4%	9,508K
bisq.core.trade.statistics.TradeStatistics2Store.map	0.2%	607K
j.u.WeakHashMap$EntryIterator.this$0	<0.1%	168K

How to fix Inefficient Collections?

To see our recommendations, please purchase Enterprise Edition.

7. Inefficient Object Arrays

Learn more about Inefficient Object Arrays

Total Object Arrays	Inefficient Object Arrays	Wasted Memory

201,896

38,772

1.03mb (0.3%)

Who is holding Inefficient Object Arrays?

Object Tree	Percentage	size
Unreachable (garbage) objects	<0.1%	130K
org.spongycastle.math.ec.custom.sec.SecP256K1Point.zs	<0.1%	95K
java.lang.reflect.Method.parameterTypes	<0.1%	77K

How to fix Inefficient Object Arrays?

To see our recommendations, please purchase Enterprise Edition.

8. Inefficient Primitive Arrays

Learn more about Inefficient Primitive Arrays

Total Primitive Arrays	Inefficient Primi Arrays	Wasted Memory

2,794,263	14,269	4.42mb (1.1%)

Top inefficient Primitive Arrays

Problem	Percentage	Wasted
< 0.1% of byte[] contains lot of 0s	0.4%	1.68mb
< 0.1% of byte[] contains no elements	0.3%	1.36mb
2% of float[] contains lot of 0s	0.1%	473.55kb

Who is holding Inefficient Primitive Arrays?

Object Tree	Percentage	size
Unreachable (garbage) objects	0.3%	1,311K
java.nio.HeapByteBuffer.hb	0.2%	982K
byte[]	0.1%	502K
com.sun.prism.impl.VertexBuffer.coordArray	0.1%	441K

How to fix Inefficient Primitive Arrays?

To see our recommendations, please purchase Enterprise Edition.

9. Boxed Numbers

Learn more about Boxed Numbers

Not Detected

10. Duplicate Objects

Learn more about Duplicate Objects

Not Detected

11. Duplicate Primitive Arrays

Learn more about Duplicate Primitive Arrays

Total Duplicate Arrays	Wasted Memory
1,776,639	91.6mb (23.6%)

Types of Duplicate Arrays

Array Type	Percentage	Wasted	Duplicate Count
byte[]	23.3%	90.53mb	1,759,175
int[]	0.2%	818.3kb	13,299
long[]	<0.1%	170.73kb	3,243
char[]	<0.1%	79.34kb	430
short[]	<0.1%	12.12kb	155

Show all records >>

Top Duplicate Arrays

Duplicate Array	Percentage	Wasted	Count
byte[9437182](109, 101, 115, 115, 97, 103, 101, 95, 118, 101, 114, 115, 105, 111, 110, 58, 32, 49, 48, 10, ...)	2.3%	9mb	2
byte[3]('B', 'T', 'C')	0.6%	2.44mb	106,643
byte[52998](0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...)	0.3%	1.21mb	25
byte[12]('B', 'L', 'O', 'C', 'K', '_', 'C', 'H', 'A', 'I', 'N', 'S')	0.3%	1.03mb	33,802
byte[7]('a', 'r', 'b', 'A', 'd', 'd', 'r')	0.2%	796.5kb	33,985

Show all records >>

Who is holding Duplicate Arrays?

Object Tree	Percentage	size

Unreachable (garbage) objects	3.6%	14,300K
com.google.protobuf.ByteString$LiteralByteString.bytes	1.8%	7,157K
com.google.protobuf.ByteString$LiteralByteString.bytes	1.7%	6,673K
bisq.network.p2p.storage.P2PDataStorage$ByteArray.bytes	0.8%	3,330K
com.google.protobuf.ByteString$LiteralByteString.bytes	0.8%	3,193K

Show all records >>

How to fix Duplicate Arrays?

To see our recommendations, please purchase Enterprise Edition.

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?

To see our recommendations, please purchase Enterprise Edition.

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.