

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


Learn more about [What's in Memory](#)






Class	Percentage	Size	Count
<a href="#">String</a>	50.8%	104.93mb	1,354,264
<a href="#">byte[]</a>	9.0%	18.65mb	1,633,106
<a href="#">bisq.core.trade.statistics.TradeStatistics2</a>	5.1%	10.51mb	105,934
<a href="#">j.u.LinkedHashMap</a>	4.6%	9.51mb	55,557
<a href="#">java.util.concurrent.ConcurrentHashMap</a>	3.5%	7.3mb	439
<a href="#">j.u.HashSet</a>	3.1%	6.39mb	1,230

[Show all records >>](#)

## 3. Large objects

Learn more about [Large Objects](#)

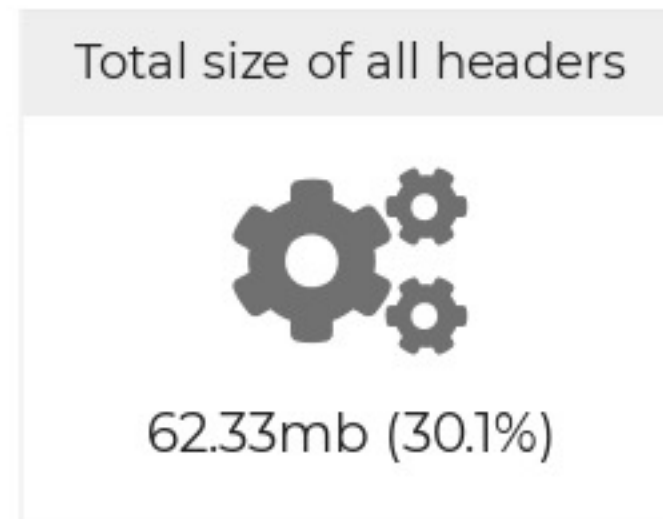
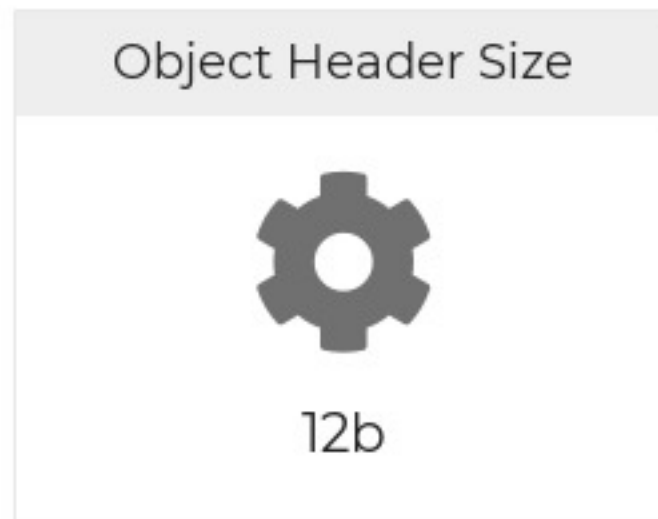
Name	Percentage	Size
 <a href="#">Java Static_bisq.desktop.main.MainView.rootContainer</a>	44.7%	92.46mb

<<Object might be causing memory leak.>>		
 <a href="#">Java Static java.lang.ApplicationShutdownHooks.hooks</a>  <<Object might be causing memory leak.>>		88.68mb
<a href="#">Java Static org.apache.commons.logging.LogFactory.thisClassLoader</a> 		3.7mb
<a href="#">Unreachable (garbage) objects</a> 		2.6mb
<a href="#">Java Static com.sun.javafx.css.StyleManager\$InstanceHolder.INSTANCE</a> 		1.93mb
... and 17734 more objects retaining 7.35mb (3.6%)		

[Show all records >>](#)

## 4. Object Headers

Learn more about [Object Headers](#)



### Top Object Headers

Class	Percentage	Total header size	Avg obj size	Count
-------	------------	-------------------	--------------	-------

byte[]	9.0%	18.69mb	59	1,633,106
String	7.5%	15.5mb	24	1,354,264
j.u.HashMap\$Node	1.2%	2.46mb	32	214,845
java.util.concurrent.ConcurrentHashMap\$Node	1.0%	2.15mb	32	188,188
bisq.network.p2p.storage.P2PDataStorage\$ByteArray	0.9%	1.82mb	16	158,821

[Show all records >>](#)

## How to fix excessive Object headers?

To see our recommendations, please purchase [Enterprise Edition](#).

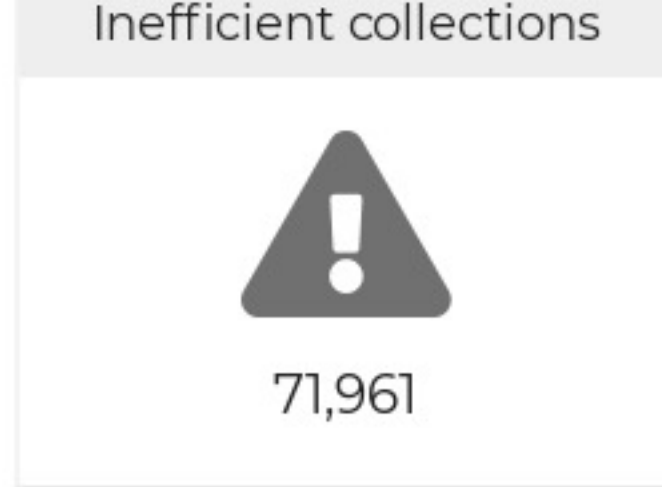
## 5. Duplicate Strings

Learn more about [Duplicate Strings](#)

Not Detected

## 6. Inefficient collections

Learn more about [Inefficient Collections](#)



## 💡 Top inefficient collections

Problem	Percentage	Wasted
99% of j.u.LinkedHashMap contains 1 element only	4.5%	9.25mb
7% of java.util.concurrent.ConcurrentHashMap contains half empty elements	0.3%	693.56kb
39% of j.u.ArrayList contains 1 element only	0.2%	324.82kb

## ❓ Who is holding Inefficient Collections?

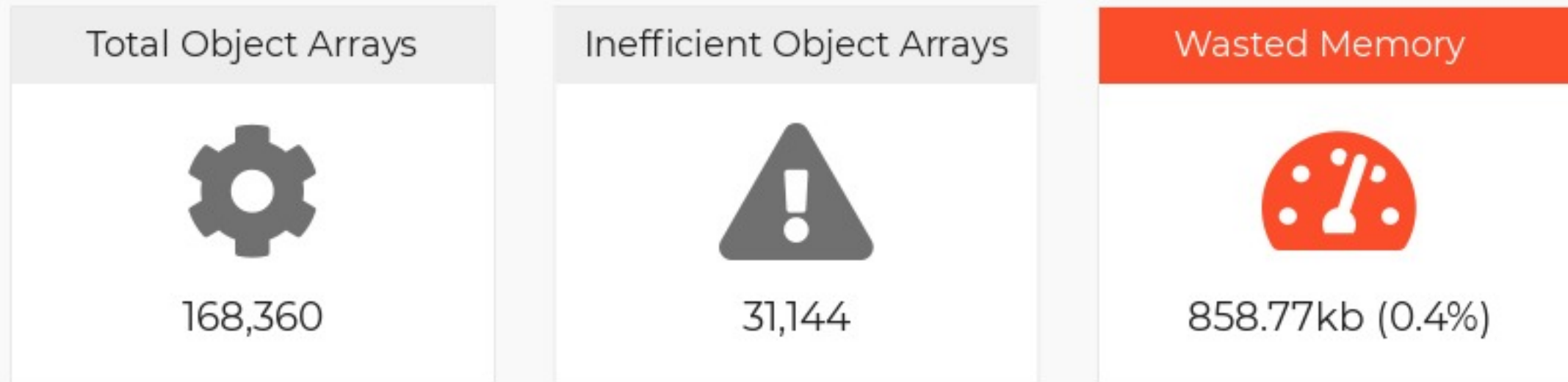
Object Tree	Percentage	size
<a href="#">com.google.protobuf.MapField\$MutatabilityAwareMap.delegate</a>	4.4%	9,398K
<a href="#">bisq.core.trade.statistics.TradeStatistics2Store.map</a>	0.3%	610K
<a href="#">jdk.internal.loader.ClassLoaders\$PlatformClassLoader.parallelLockMap</a>	<0.1%	36K

## 🔧 How to fix Inefficient Collections?

To see our recommendations, please purchase [Enterprise Edition](#).

# 7. Inefficient Object Arrays

Learn more about [Inefficient Object Arrays](#)



## ? Who is holding Inefficient Object Arrays?

Object Tree	Percentage	size
<a href="#">org.spongeycastle.math.ec.custom.sec.SecP256K1Point.zs</a>	<0.1%	95K
<a href="#">java.lang.reflect.Method.parameterTypes</a>	<0.1%	68K
<a href="#">java.lang.reflect.Method.parameterTypes</a>	<0.1%	60K

## How to fix Inefficient Object Arrays?

To see our recommendations, please purchase [Enterprise Edition](#).

# 8. Inefficient Primitive Arrays

Learn more about [Inefficient Primitive Arrays](#)



## 💡 Top inefficient Primitive Arrays

Problem	Percentage	Wasted
< 0.1% of byte[] contains lot of 0s	0.7%	1.44mb
2% of float[] contains lot of 0s	0.2%	472.68kb

## ❓ Who is holding Inefficient Primitive Arrays?

Object Tree	Percentage	size
<a href="#">java.nio.HeapByteBuffer.hb</a>	0.5%	982K
<a href="#">com.sun.prism.impl.VertexBuffer.coordArray</a>	0.2%	440K
<a href="#">byte[]</a>	0.2%	381K

## 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



925,022

Wasted Memory



42.49mb (20.5%)

## ☰ Types of Duplicate Arrays

Array Type	Percentage	Wasted	Duplicate Count
byte[]	20.2%	41.8mb	912,104
int[]	0.2%	525.18kb	9,305
long[]	<0.1%	141.19kb	3,108
char[]	<0.1%	9.7kb	145
double[]	<0.1%	9.38kb	107

[Show all records >>](#)

## 💡 Top Duplicate Arrays

Duplicate Array	Percentage	Wasted	Count
byte[3]('B', 'T', 'C')	1.2%	2.43mb	106,003
byte[12]('B', 'L', 'O', 'C', 'K', '_', 'C', 'H', 'A', 'I', 'N', 'S')	0.5%	1.03mb	33,727
byte[7]('d', 'r', 'b', 'A', 'd', 'd', 'r')	0.4%	796.5kb	33,985
byte[5]('t', 't', 't', 't', 't')	0.3%	715.5kb	33,333



byte[3]('E', 'U', 'R')	0.3%	725.6kb	30,960
byte[3]('X', 'M', 'R')	0.3%	699.7kb	29,855

[Show all records >>](#)

## ? Who is holding Duplicate Arrays?

Object Tree	Percentage	size
<a href="#">bisq.core.dao.monitoring.model.DaoStateHash.hash</a>	0.4%	881K
<a href="#">bisq.core.dao.monitoring.model.DaoStateHash.prevHash</a>	0.4%	881K
<a href="#">bisq.core.account.sign.SignedWitness.signerPubKey</a>	0.3%	532K
<a href="#">bisq.core.account.sign.SignedWitness.witnessOwnerPubKey</a>	0.2%	418K


## 🔧 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](#)





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.

