# 文档

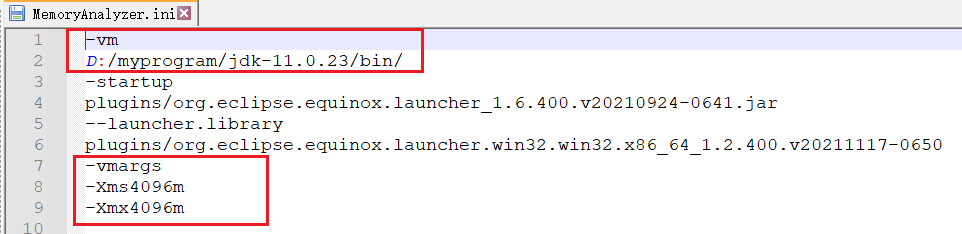
[Help - Eclipse Platform](https://help.eclipse.org/latest/index.jsp?topic=/org.eclipse.mat.ui.help/welcome.html)

# 安装

jdk-17 Memory Analyzer 1.14 及更高版本需要

jdk-11 Memory Analyzer 1.12 及更高版本需要

jdk-1.8 Memory Analyzer 1.8 至 1.11



-vm

D:/myprogram/jdk-11.0.23/bin/javaw.exe

-vmargs

-Xms4096m

-Xmx4096m

<https://eclipse.dev/mat/downloads.php>

# 生成dumap设置

-Xms21m -Xmx21m -XX:+HeapDumpOnOutOfMemoryError

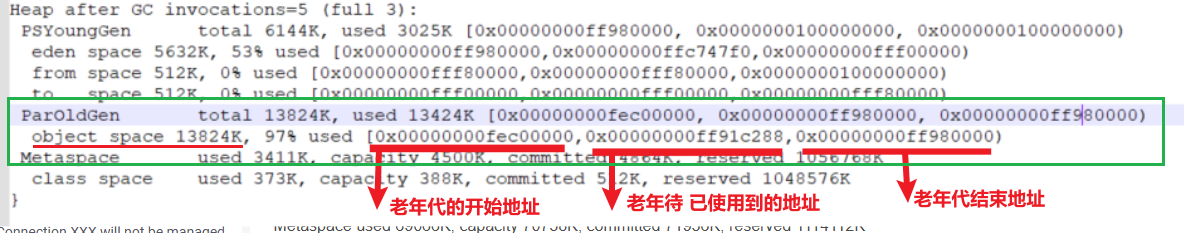


# 查看老年代对象

## 1. Java程序参数配置

-Xms20m -Xmx20m -XX:+HeapDumpBeforeFullGC -XX:+PrintGCDetails -XX:+PrintHeapAtGC -Xloggc:./logs/gc.log

## 2. 查询GC日志中的老年代内存地址



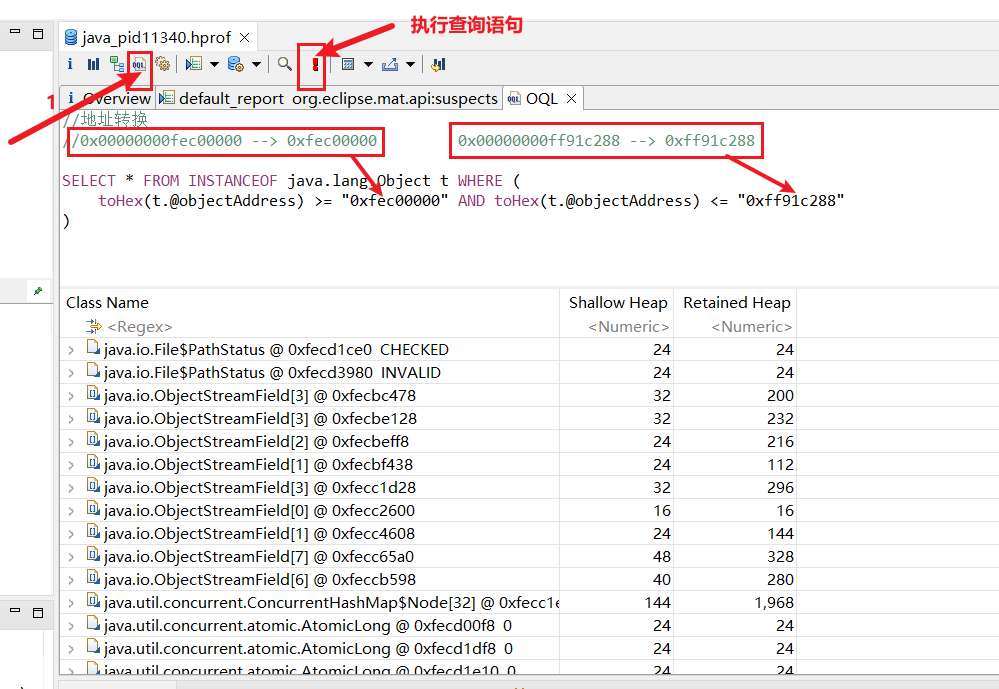
## 3. 导入堆文件查询老年代对象

**OQL:**

SELECT \* FROM INSTANCEOF java.lang.Object t WHERE (

toHex(t.@objectAddress) >= "0xfec00000" AND toHex(t.@objectAddress) <= "0xff91c288"

)



## 参考

[java old gen\_Java查看Old代对象具体信息\_hprof](https://blog.csdn.net/Dongguabai/article/details/123425128)

# 基础概念

## Incomming Reference

指的是**引用当前对象**的对象，例如：当前char被string对象引用

谁引用了我

## Outgoing Reference

指的是**当前对象**引用的对象。 例如：当前char引用的 char[] 对象

我引用了谁



## Shallow heap

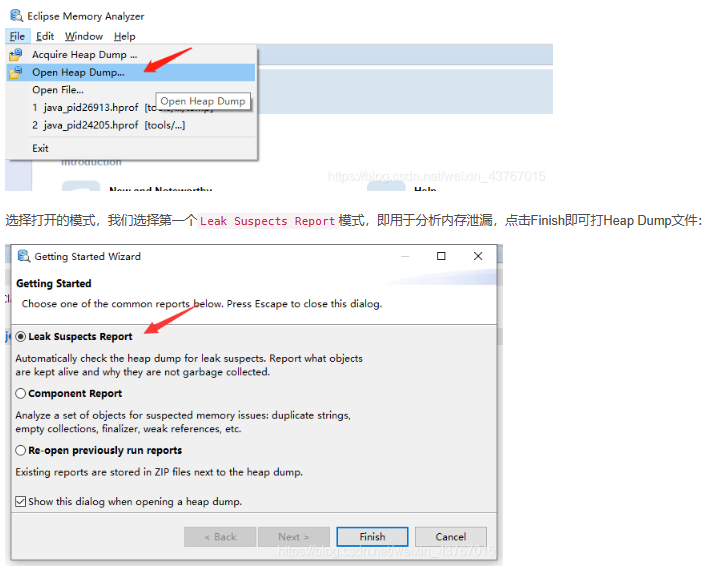
指的是对象自身所占据的内存。

## Retained heap

指的是当对象不再被引用时，垃圾回收器所能回收的总内存，包括对象自身所占据的内存，以及仅能够通过该对象引用到的其他对象所占据的内存。

# 介绍

## 导入



泄露怀疑报告

**Leak Suspects Report**

自动检查堆转储程序中是否存在怀疑泄漏者。报告什么对象是活的，为什么它们不是垃圾

Automatically check the heap dump for leak suspects.Report what objects are kept alive

and why they are not garbage collected.

收集组件报告

**Component Report**

分析是否存在可疑内存问题的一组对象：重复的字符串，空的集合，终结器，弱的参考文献等。

Analyze a set of objects for suspected memory issues:

duplicate strings,empty collections,finalizer,weak references,etc.

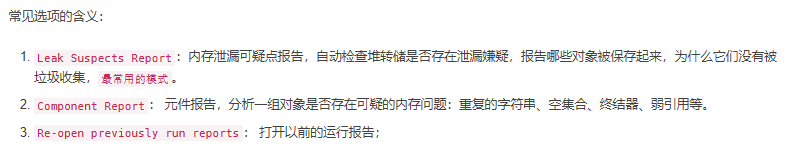
打开以前运行的报告

**Re-open previously run reports**

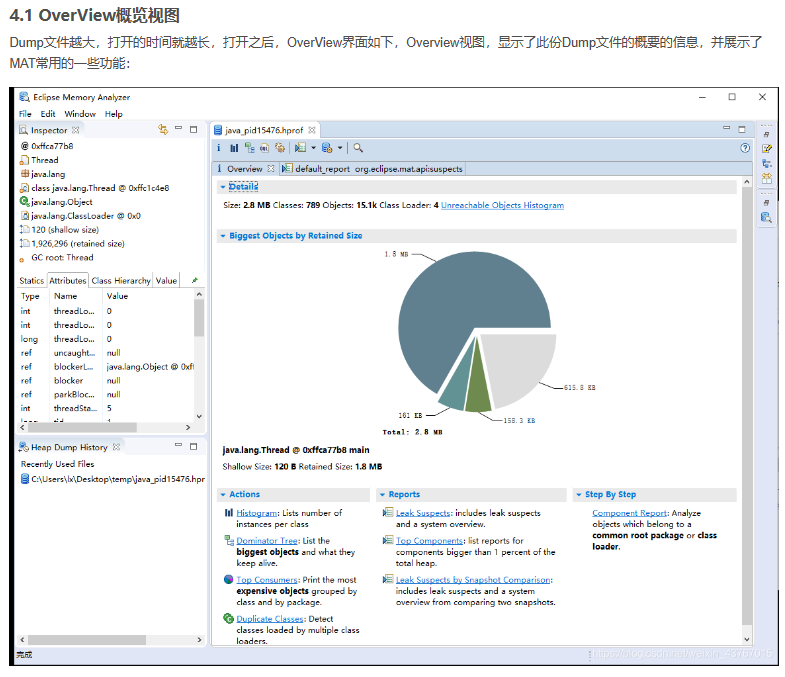
现有的报告存储在堆转储文件旁边的ZIP文件中。在打开堆转储时，将显示此对话框。

Existing reports are stored in ZIP files next to the heap dump.

Show this dialog when opening a heap dump.

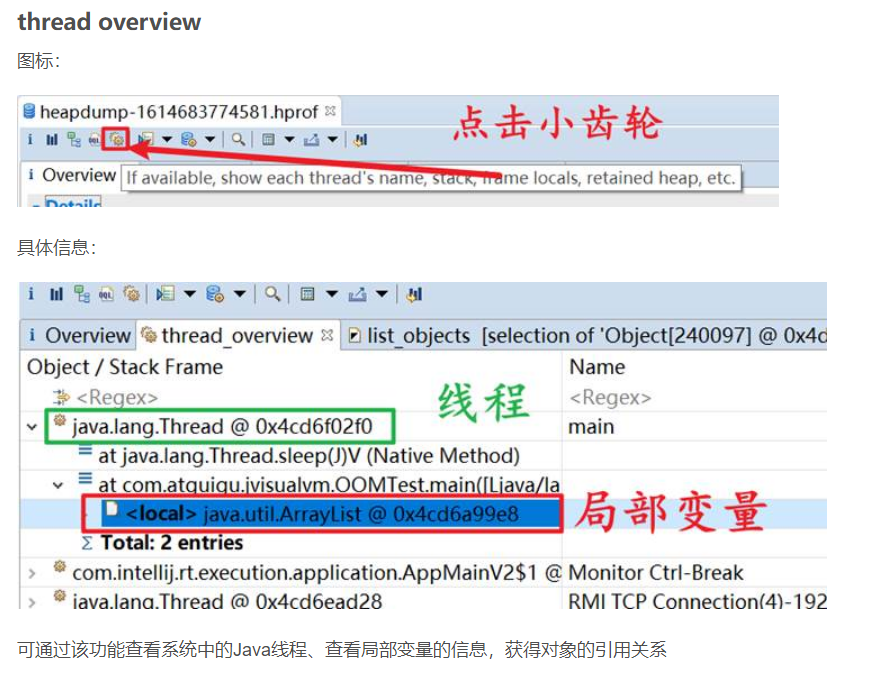


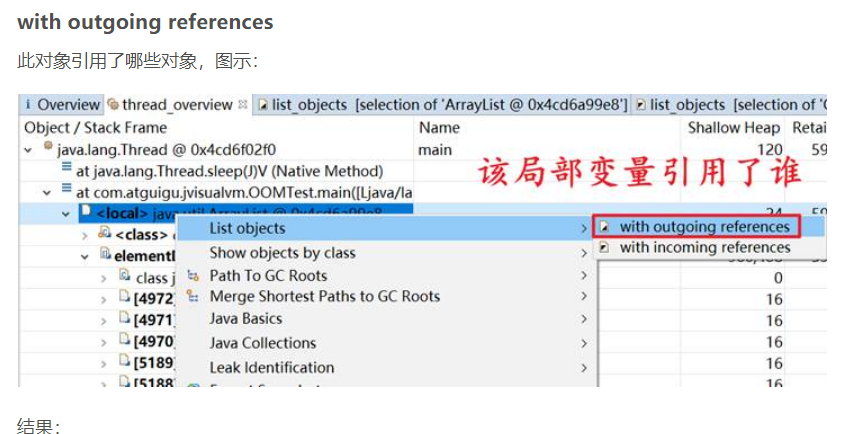
## 首页

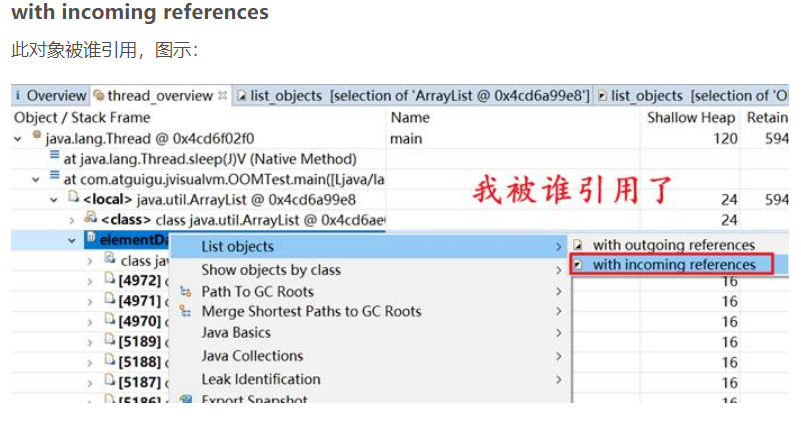




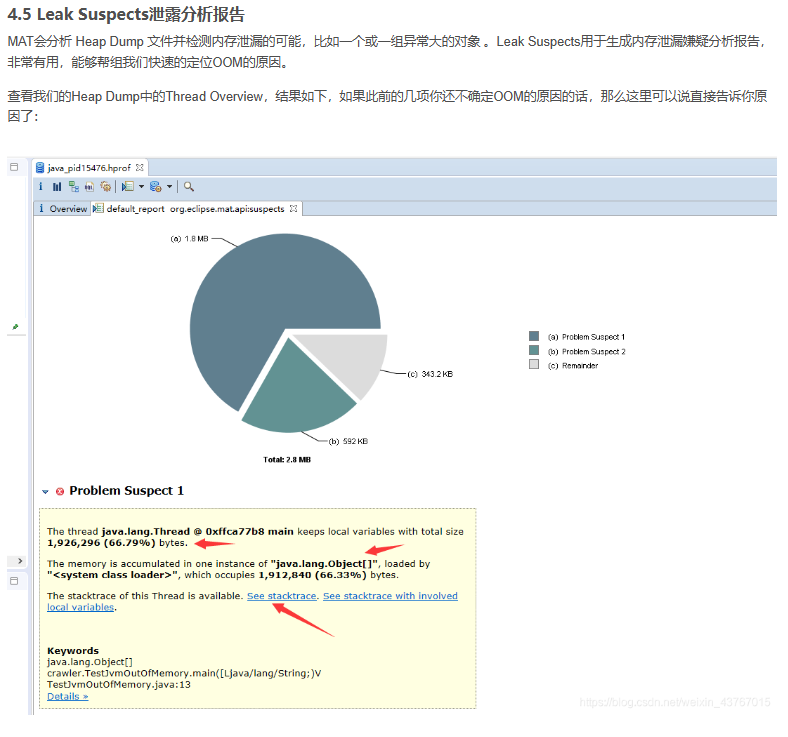
## 线程





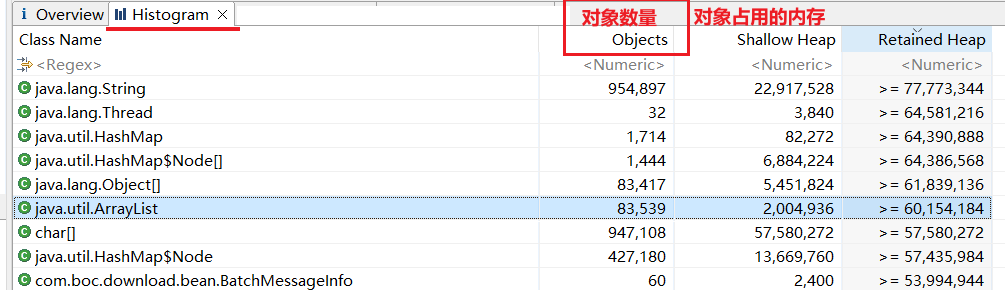


## 原因

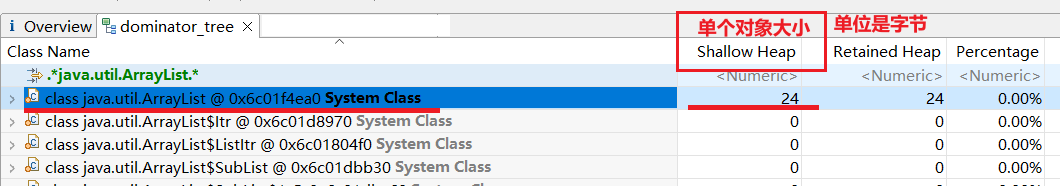


# 使用细节

## 直方图与支配树的作用

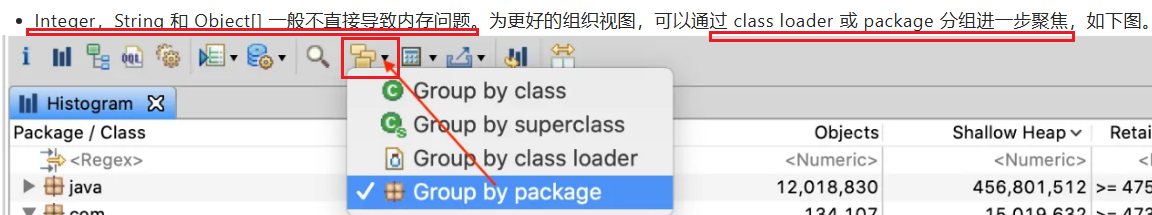


**查看 类的 对象数量**



**查看线程 对应的 对象数量**

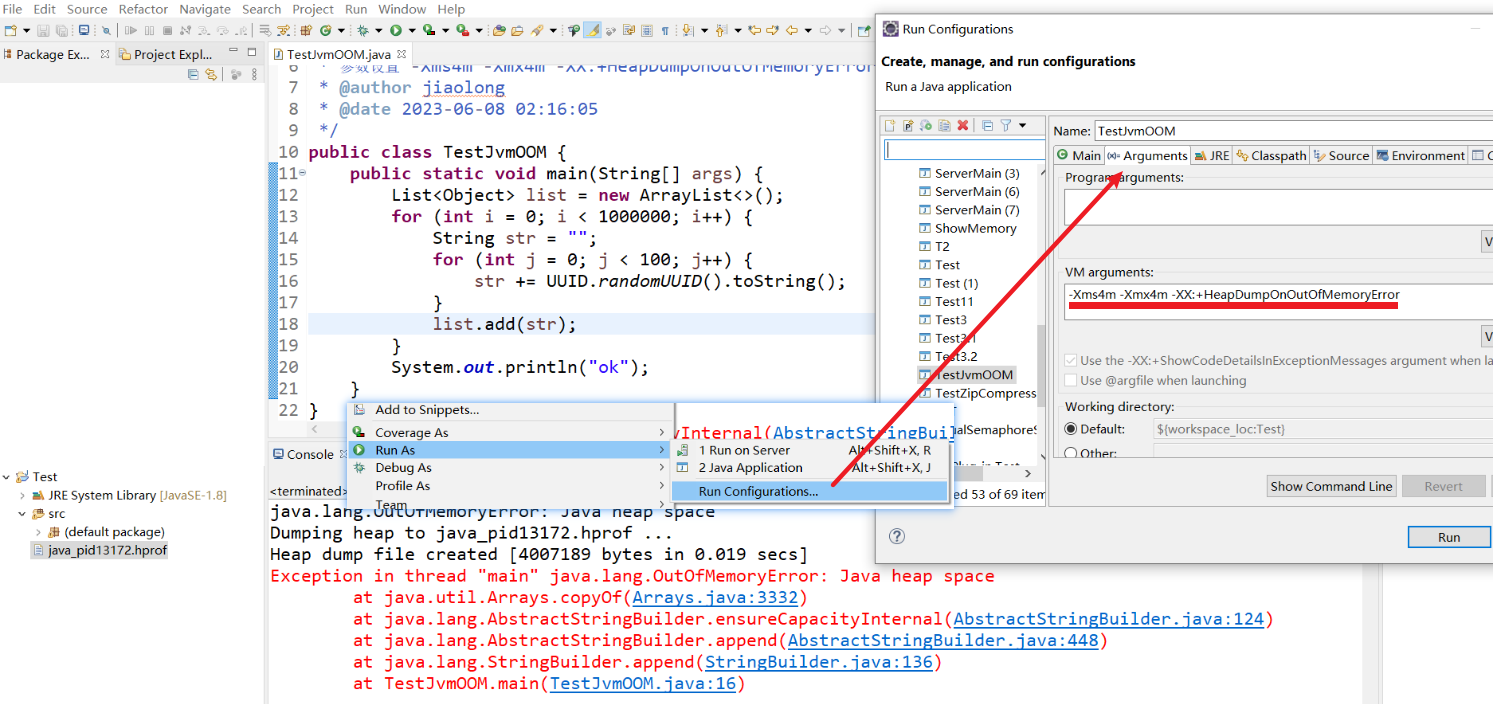
## 过滤技巧



# 案例

1）[Java内存分析工具MAT(Memory Analyzer Tool)的介绍与使用](https://blog.csdn.net/weixin_43767015/article/details/118208237)

2）[Java内存分析工具MAT(Memory Analyzer Tool)安装使用实例](https://blog.csdn.net/Jin_Kwok/article/details/80326088)



# 参考

案例1. <https://www.cnblogs.com/zhihongShee/p/17461264.html>

案例2：<https://blog.csdn.net/forBurnInG/article/details/103530737>

[Mat使用详解-非常详细](https://www.cnblogs.com/zh94/p/14051852.html)

[MAT（Memory Analyzer Tool）工具使用超详细版](https://blog.csdn.net/lyd135364/article/details/121449969)

[性能优化工具-MAT -总体介绍](https://www.jianshu.com/p/97251691af88)

[MAT工具的概念讲解与使用-重点看](https://zhuanlan.zhihu.com/p/585668729)