

**2018年春季学期  
计算机学院大二软件构造课程**

**Lab 5实验报告**

|  |  |
| --- | --- |
| 姓名 | 肖一诺 |
| 学号 | 1160300222 |
| 班号 | 1603002 |
| 电子邮件 | m18846183092@163.com |
| 手机号码 | 18846183092 |

**目录**

[1 实验目标概述 1](#_Toc513573923)

[2 实验环境配置 1](#_Toc513573924)

[3 实验过程 1](#_Toc513573925)

[3.1 Static Program Analysis 1](#_Toc513573926)

[3.1.1 人工代码走查（walkthrough） 1](#_Toc513573927)

[3.1.2 使用CheckStyle和FindBugs进行静态代码分析 1](#_Toc513573928)

[3.2 Java I/O 1](#_Toc513573929)

[3.2.1 多种I/O实现方式 1](#_Toc513573930)

[3.2.2 多种I/O实现方式的效率对比分析 2](#_Toc513573931)

[3.3 Java Memory Management and Garbage Collection (GC) 3](#_Toc513573932)

[3.3.1 使用-verbose:gc参数 3](#_Toc513573933)

[3.3.2 用jstat命令行工具的-gc和-gcutil参数 3](#_Toc513573934)

[3.3.3 使用jmap -heap命令行工具 3](#_Toc513573935)

[3.3.4 使用jmap -histo命令行工具（可选） 3](#_Toc513573936)

[3.3.5 使用jmap -permstat命令行工具（可选） 3](#_Toc513573937)

[3.3.6 使用jconsole或VisualVM工具 3](#_Toc513573938)

[3.3.7 分析垃圾回收过程是否正常、异常 3](#_Toc513573939)

[3.3.8 配置JVM参数并发现最优参数配置 3](#_Toc513573940)

[3.4 Dynamic Program Profiling 3](#_Toc513573941)

[3.4.1 使用Visual VM进行CPU Profiling 3](#_Toc513573942)

[3.4.2 使用Visual VM进行Memory profiling 3](#_Toc513573943)

[3.5 Memory Dump Analysis and Performance Optimization 3](#_Toc513573944)

[3.5.1 内存导出(memory dump) 3](#_Toc513573945)

[3.5.2 使用MAT分析内存导出文件 3](#_Toc513573946)

[3.5.3 发现热点/瓶颈并改进、改进前后的性能对比分析 3](#_Toc513573947)

[3.5.4 jhat和OQL查询内存导出（可选） 4](#_Toc513573948)

[3.5.5 jstack导出java程序运行时的调用栈（可选） 4](#_Toc513573949)

[4 实验进度记录 4](#_Toc513573950)

[5 实验过程中遇到的困难与解决途径 4](#_Toc513573951)

[6 实验过程中收获的经验、教训、感想 4](#_Toc513573952)

# 实验目标概述

本次实验通过对 Lab4 的代码进行静态和动态分析，发现代码中存在的不符 合代码规范的地方、具有潜在 bug 的地方、性能存在缺陷的地方（执行时间热点、 内存消耗大的语句、函数、类），进而使用第 4、7、8 章所学的知识对这些问题 加以改进，掌握代码持续优化的方法，让代码既“看起来很美”，又“运行起来很美”。

具体训练内容：

1. 静态代码分析
2. 动态代码分析
3. JVM内存管理与垃圾回收的优化配置
4. 运行时内存导出及其分析
5. 运行时调用栈及其分析
6. 高性能I/O
7. Jiyu设计模式的代码调优
8. 代码重构

# 实验环境配置

简要陈述你配置本次实验所需环境的过程，必要时可以给出屏幕截图。

特别是要记录配置过程中遇到的问题和困难，以及如何解决的。

在这里给出你的GitHub Lab5仓库的URL地址（Lab5-学号）。

https://github.com/ComputerScienceHIT/Lab5-1160300222.git

# 实验过程

本次实验代码保存在test和src\src\application\Lab5App.java中。

## Static Program Analysis

### 人工代码走查（walkthrough）

列出你所发现的问题和所做的修改。每种类型的问题只需列出一个示例即可。

1. 类型不匹配：

进行强制类型转换或者检查代码中的错误，进行修改

1. 正则表达式部分读取发生错误：

例：String sample3 = "Vertex=<\"movie1\",\"Movie\",<\"2014\",\"China\",\"8.5\">";

最后那里少了一个“>”，导致读取错误。进行修改，改为String sample3 = "Vertex=<\"movie1\",\"Movie\",<\"2014\",\"China\",\"8.5\">>";

1. 写入文件发生错误，进行文件路径的修改，比如：
2. PrintStream fileOutputStream=new PrintStream("src\\file\\ScannerOutPut");

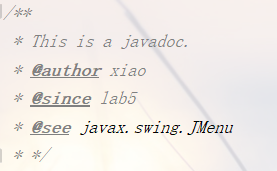
### 使用CheckStyle和FindBugs进行静态代码分析

## 

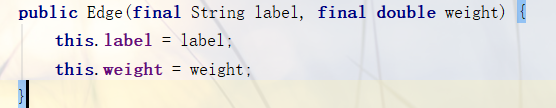
如图所示为用checkstyle检查代码的结果

其中发现的集中比较典型的问题有：

1. 导入不同的库的顺序存在问题，只需要调换一下顺序即可。
2. “=”两端需要有空格，加上即可。
3. 缺少javadoc。Java的文档是十分重要的，java文档为它从程序[源代码](https://baike.baidu.com/item/%E6%BA%90%E4%BB%A3%E7%A0%81/3969)中抽取类、方法、成员等注释形成一个和源代码配套的API帮助文档。显示一个加上javadoc之后的截图。



1. 缺少package-info文件。新建一个对应文件即可。
2. 如果是对私有变量进行赋值操作，那么参数前最好加上final。如图所示：



利用findbugs进行性能分析：

## Java I/O

### 多种I/O实现方式

实现了哪些I/O方式来读写文件，具体如何实现的。

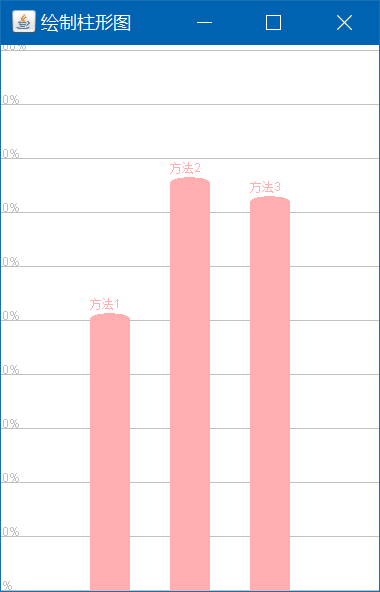
如何用strategy设计模式实现在多种I/O策略之间的切换。

### 多种I/O实现方式的效率对比分析

如何收集你的程序I/O语法文件的时间。

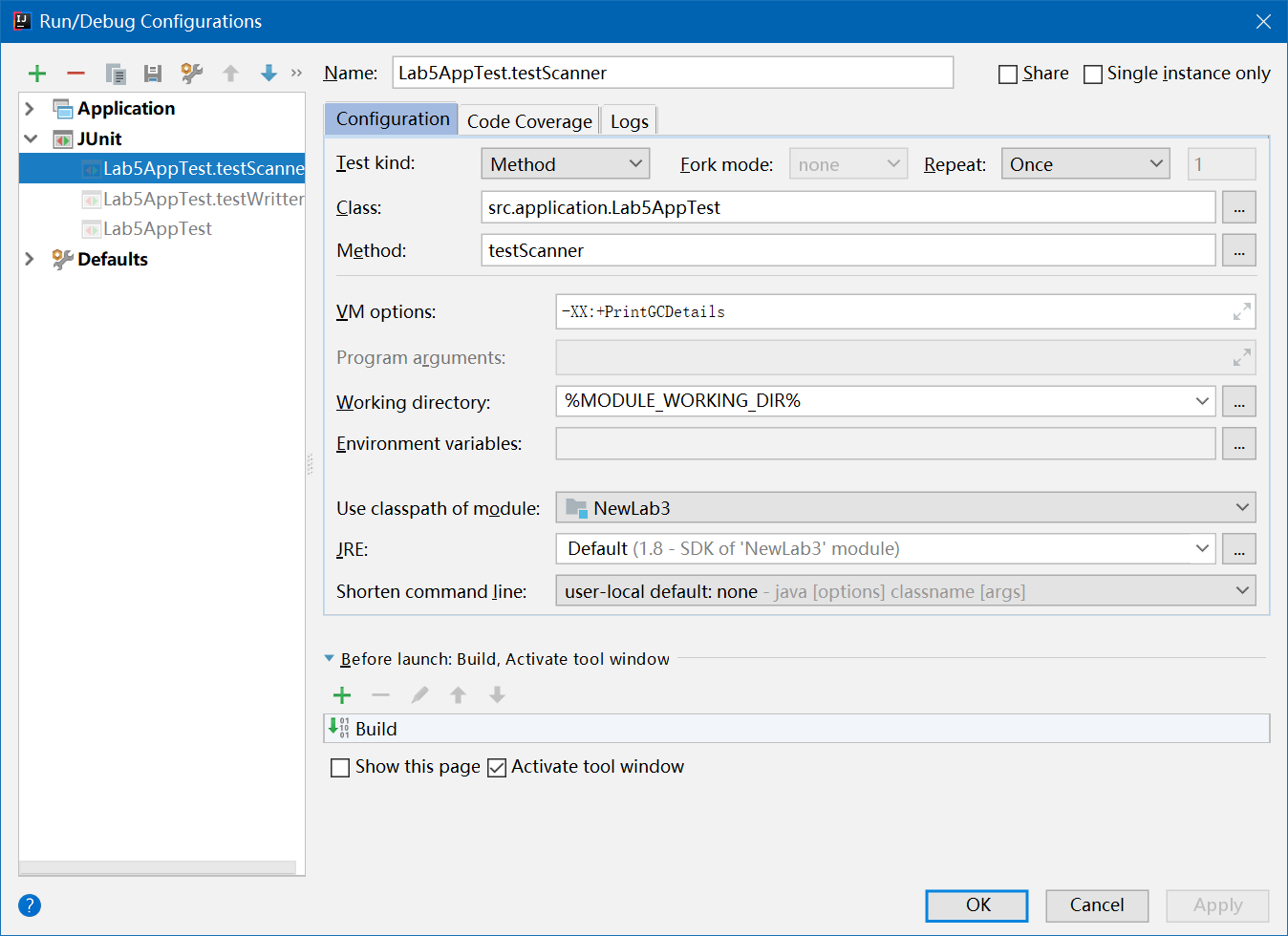
表格方式对比不同I/O的性能。

图形对比不同I/O的性能（可选）。



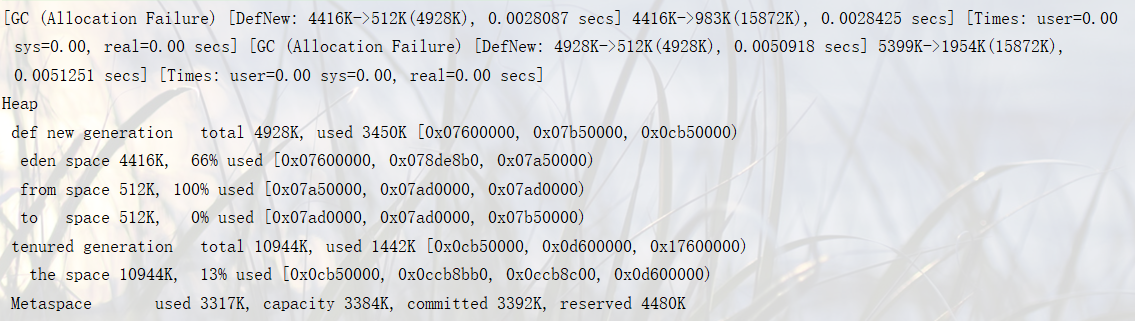
## Java Memory Management and Garbage Collection (GC)

1. 在继承开发环境idea中，修改命令行参数：



通过修改VM options实现打印垃圾回收事件。

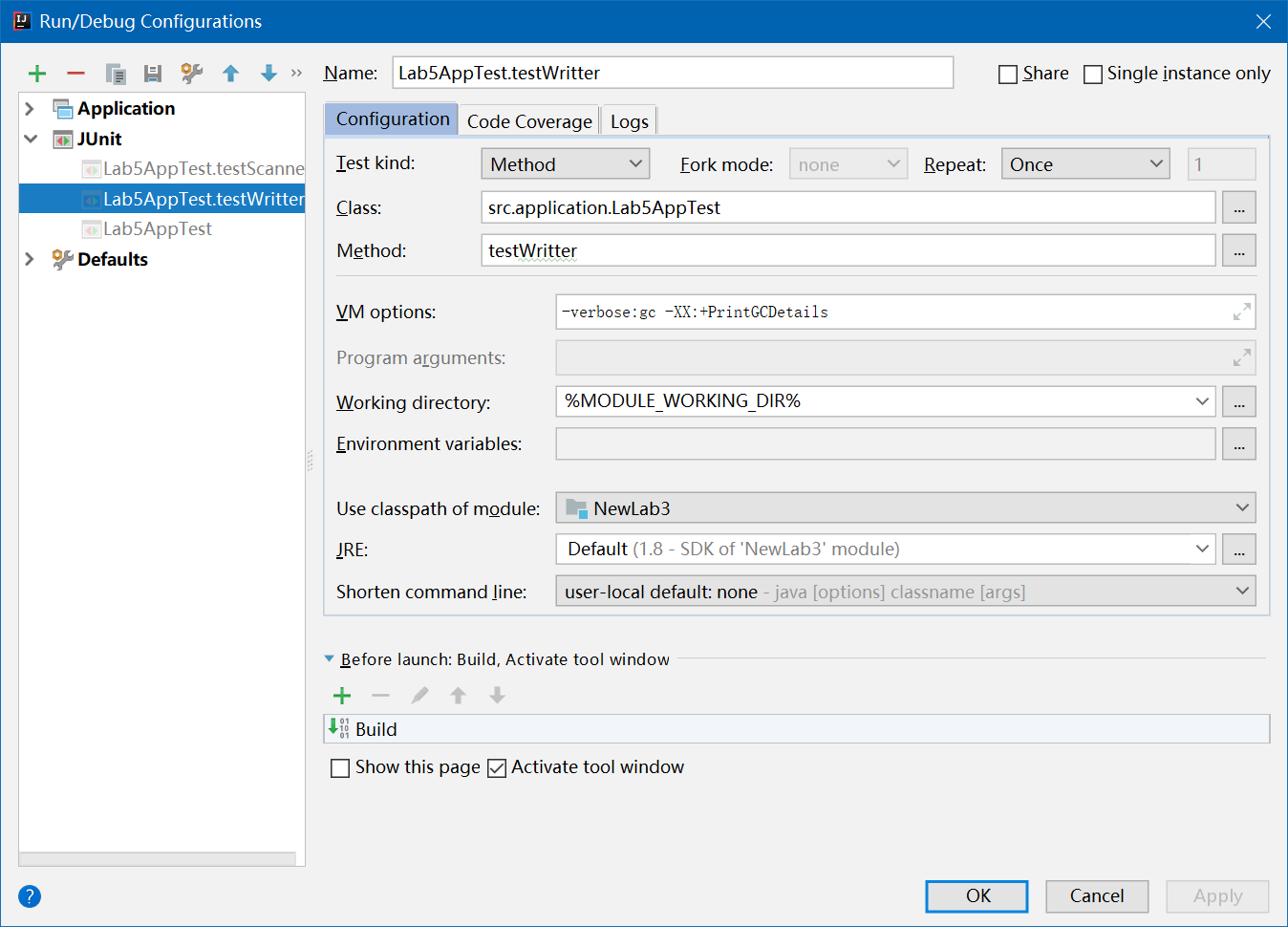
程序运行截图：



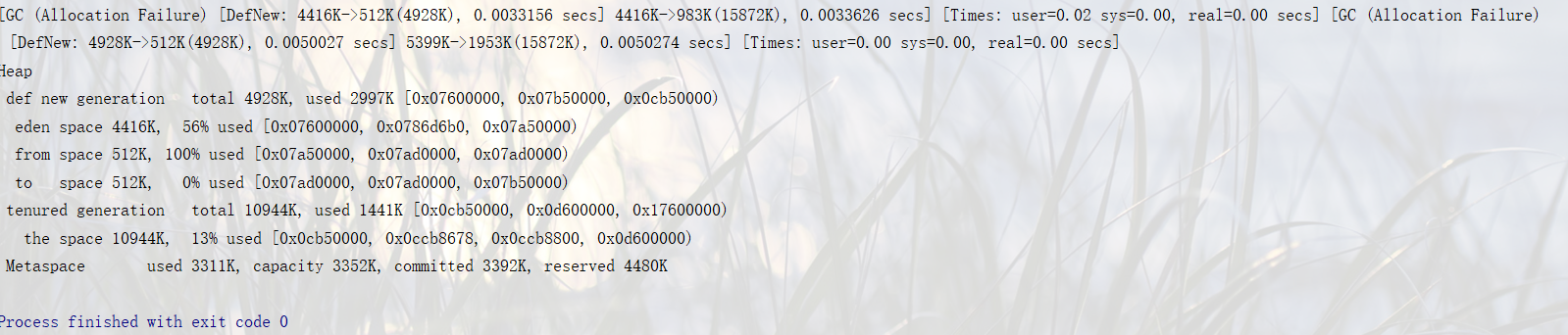
如图所示，为java虚拟机添加两个参数，分别为：

-verbose：gc

-XX：PrintGCDetails

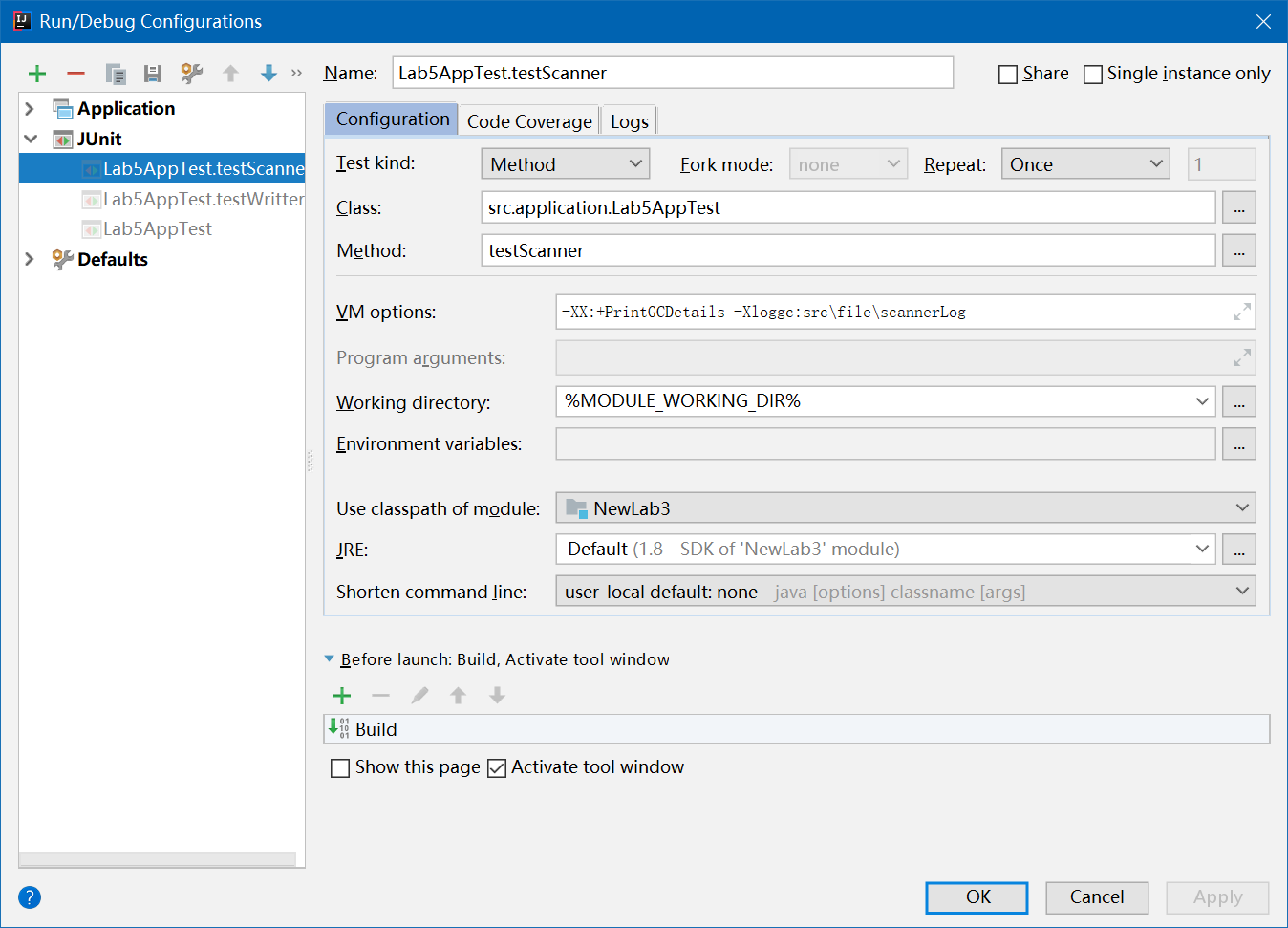


添加完毕后执行程序，得到结果为：

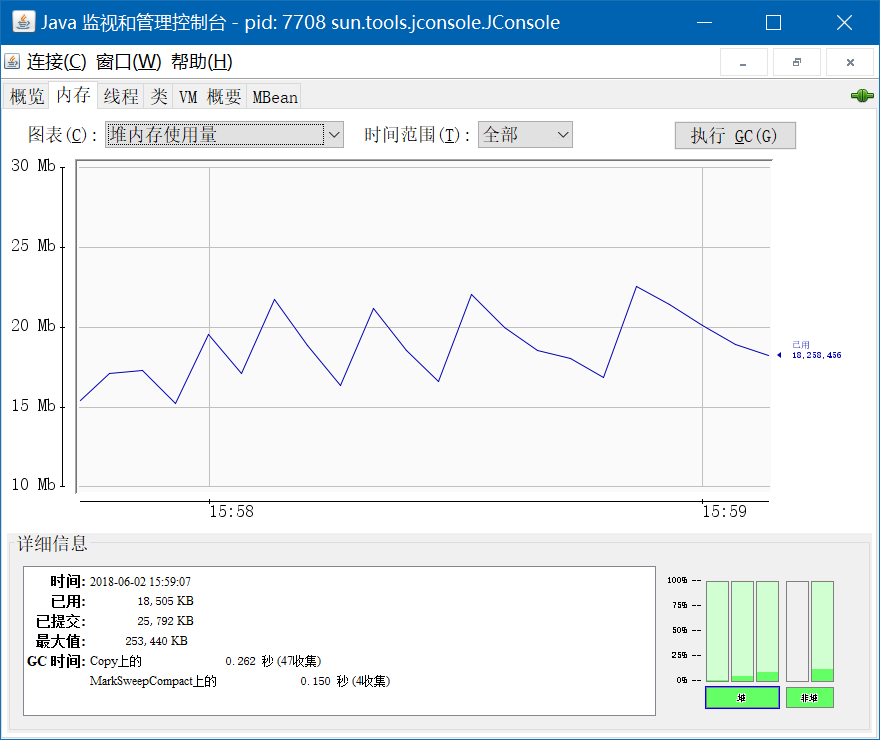


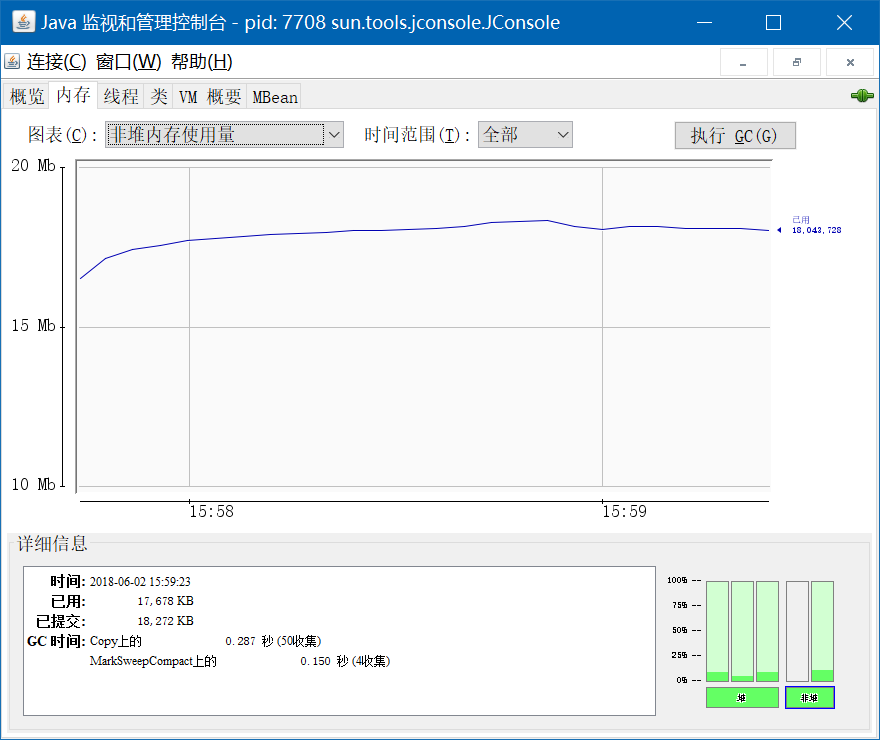
通过加入一行命令使得打印到日志文件中去：

-Xloggc:src\file\scannerLog



1. 使用jconsole：

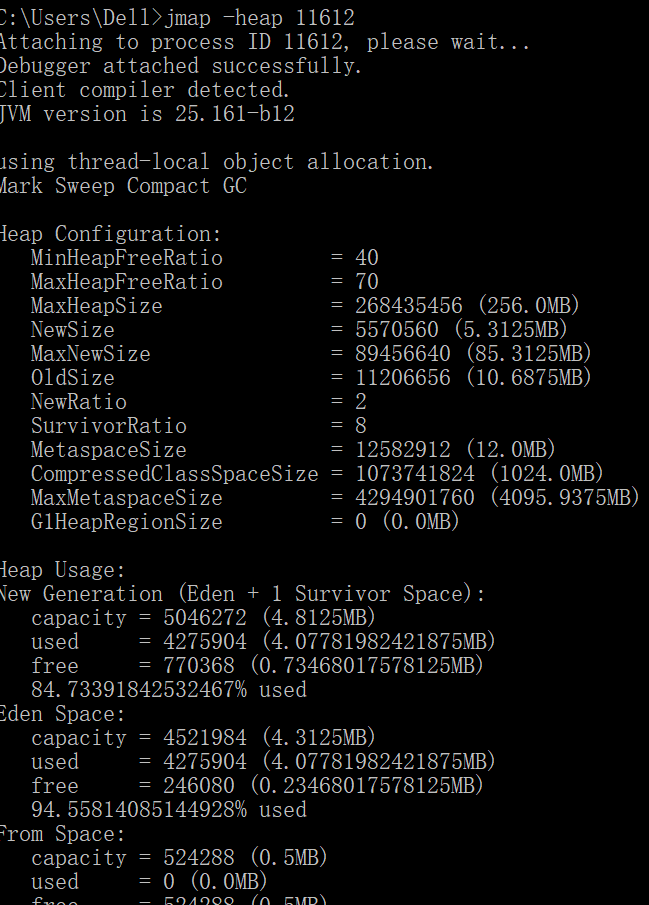


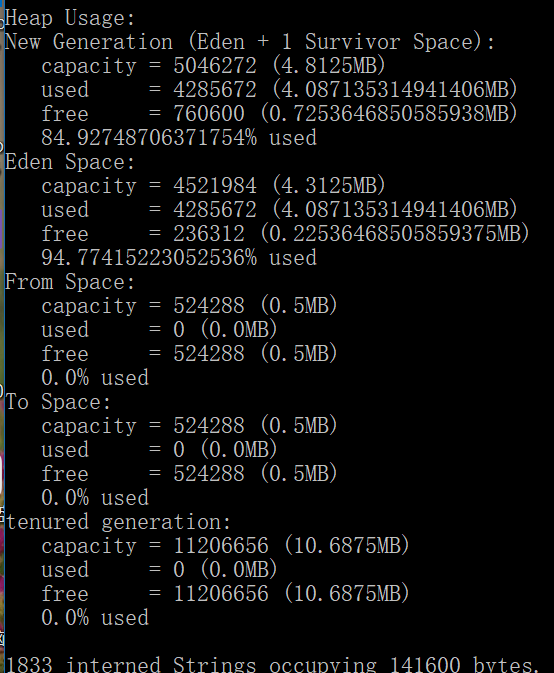


1. 使用JVM参数设定功能，在idea中配置这些参数：

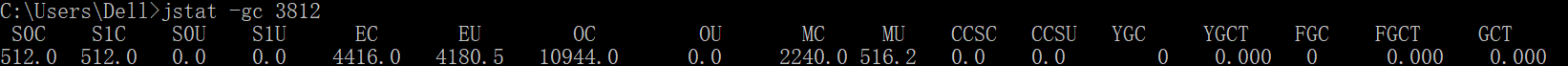


利用命令行使用jmap：

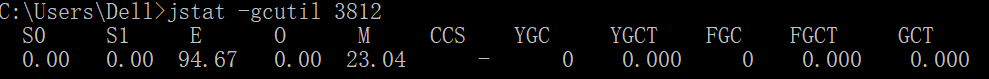




使用jstat，参数为-gc：



使用jstat，参数为-gcutil：

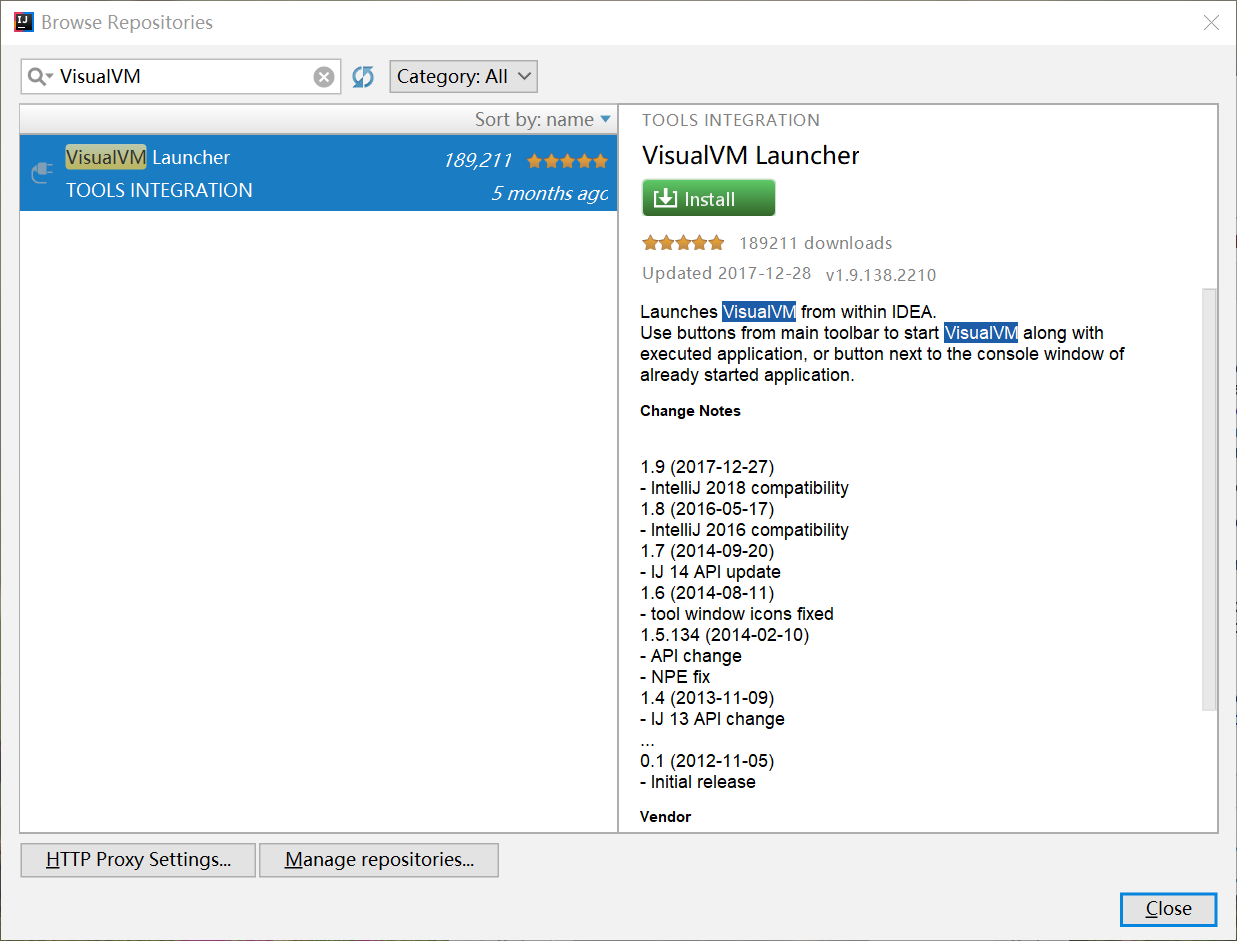


## Dynamic Program Profiling

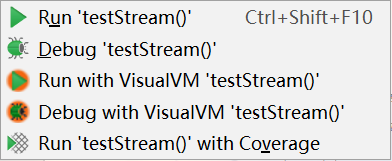
1.利用VisualVM启动程序，动态监测时间性能：

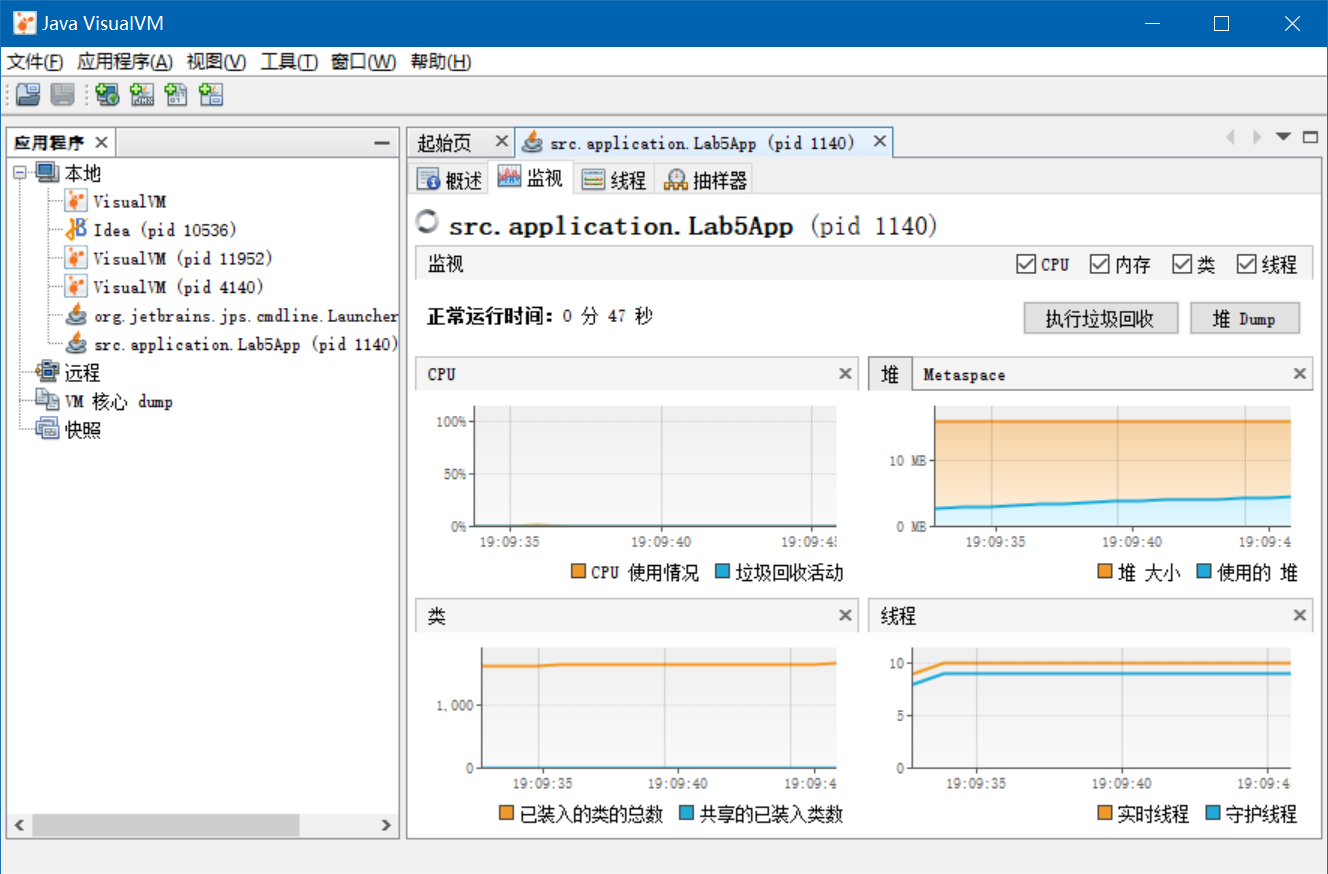
2.利用VisualVM动态监测内存空间性能：

在idea中，可以安装相应的插件来完成任务：

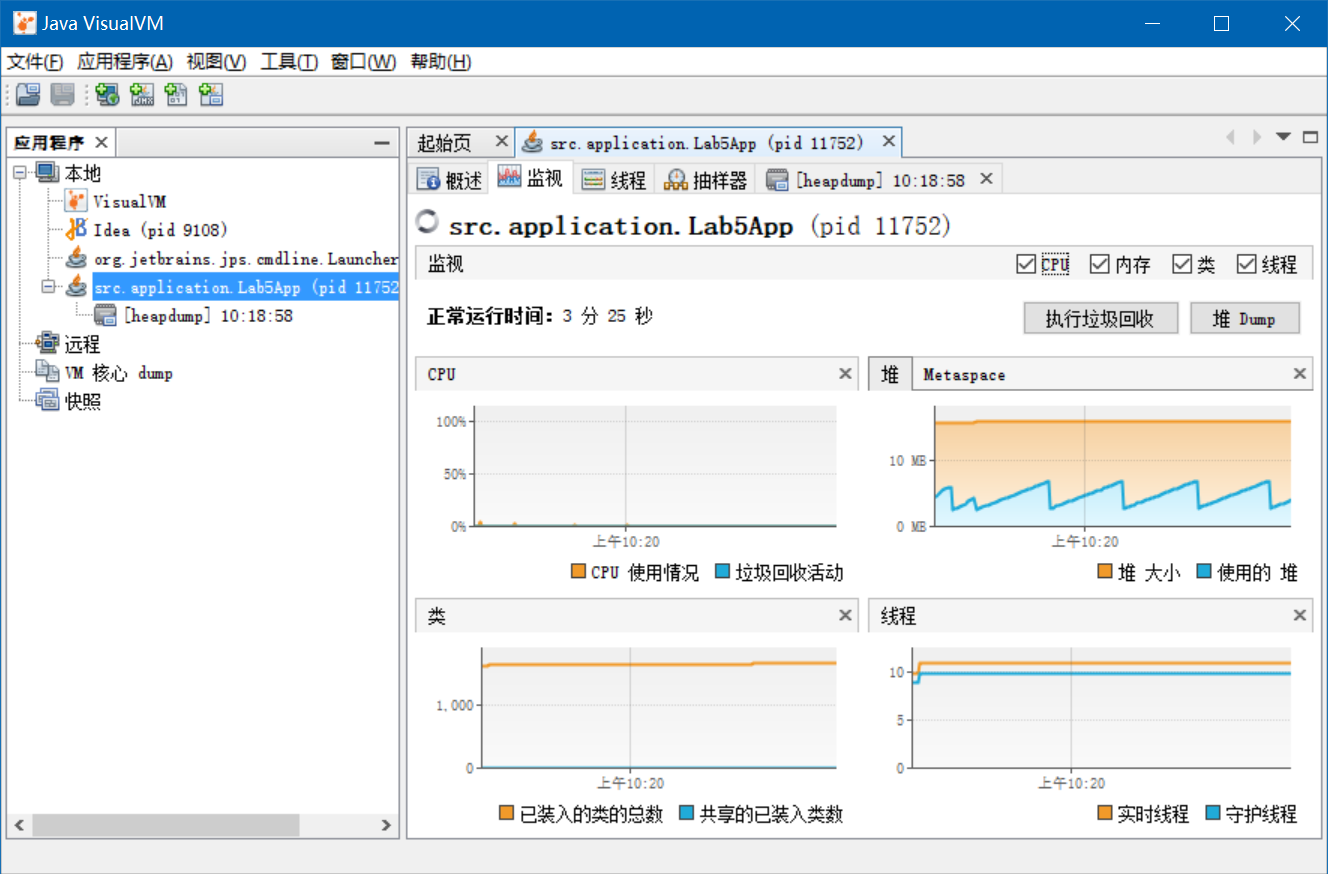


从此以后就可以直接运行程序，并且通过VisualVM进行分析。





## Memory Dump Analysis and Performance Optimization



# VisualVM profiler实验进度记录

请尽可能详细的记录你的进度情况。

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 时间段 | 计划任务 | 实际完成情况 |
|  | 第一周 | 完成静态程序测试和I/O测试 | 完成 |
|  | 第二周 | 完成java内存管理、垃圾回收和动态测试 | 完成 |
|  |  |  |  |

# 实验过程中遇到的困难与解决途径

1. java命令行不会，一开始java的jdk没有添加到环境变量中，添加之后就可以了
2. 发现idea开发环境中有很多很好用的插件，还有一部分功能可以替代命令行，比如添加-verbose:gc参数时，直接在java工程的configurations中添加就行
3. VisualVM一开始不会使用，通过查找相关资料解决问题。
4. 不知如何获取进程ID，上网查找方法，解决问题