

Федеральное государственное автономное  
образовательное учреждение высшего образования  
«Научно-образовательная корпорация ИТМО»

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия

### **Отчёт по лабораторной работе №4**

По дисциплине «Основы Программной Инженерии» (4 семестр)

**Студенты:**

Дениченко Александр Р3212

Беляев Михаил Р3212

**Практик:**

Осипов Святослав Владимирович

Санкт-Петербург  
2024 г.

# 1 Задание

1. Для своей программы из лабораторной работы 3 по дисциплине "Веб-программирование" реализовать:

MBean, считающий общее число установленных пользователем точек, а также число точек, не попадающих в область. В случае, если пользователь совершил 2 "промаха" подряд, разработанный MBean должен отправлять оповещение об этом событии.

MBean, определяющий средний интервал между кликами пользователя по координатной плоскости.

2. С помощью утилиты JConsole провести мониторинг программы:

Снять показания MBean-классов, разработанных в ходе выполнения задания 1. Определить версию Java Language Specification, реализуемую данной средой исполнения.

3. С помощью утилиты VisualVM провести мониторинг и профилирование программы:

Снять график изменения показаний MBean-классов, разработанных в ходе выполнения задания 1, с течением времени. Определить имя класса, объекты которого занимают наибольший объём памяти JVM; определить пользовательский класс, в экземплярах которого находятся эти объекты.

4. С помощью утилиты VisualVM и профилировщика IDE NetBeans, Eclipse или Idea локализовать и устранить проблемы с производительностью в программе. По результатам локализации и устранения проблемы необходимо составить отчёт, в котором должна содержаться следующая информация:

Описание выявленной проблемы.

Описание путей устранения выявленной проблемы.

Подробное (со скриншотами) описание алгоритма действий, который позволил выявить и локализовать проблему.

# 2 Исходный код разработанных MBean-классов

Листинг 1: Класс для подсчёта интервалов

```
1 @Getter
2 @Setter
3 @ManagedBean(name = "interval", eager = true)
4 @SessionScoped
5 public class ClickIntervalBean implements ClickIntervalBeanMBean, MBeanRegistration {
6     private static final Logger logger = Logger.getLogger(ClickIntervalBean.class.getName());
7
8     private List<Long> clickTimestamps = new LinkedList<>();
9     private double averageInterval = 0;
10
11     @Override
12     public void addClickTimestamp() {
13         long currentTime = System.currentTimeMillis();
14         if (!clickTimestamps.isEmpty()) {
15             long lastTime = clickTimestamps.get(clickTimestamps.size() - 1);
16             averageInterval = ((averageInterval * (clickTimestamps.size() - 1)) + (currentTime -
17                 ↪ lastTime)) / clickTimestamps.size();
18         }
19         clickTimestamps.add(currentTime);
20         logger.info("New click timestamp added: " + currentTime + "; Updated average interval: " +
21             ↪ averageInterval);
22     }
23
24     @Override
25     public void postRegister(Boolean registrationDone) {
26         logger.info("ClickIntervalBean registered to the MBean server: " + registrationDone);
27     }
28
29     @Override
30     public void postDeregister() {
31         logger.info("ClickIntervalBean deregistered from the MBean server");
32     }
33 }
```

```

30     }
31
32     @Override
33     public void preDeregister() {
34         logger.info("ClickIntervalBean is about to be deregistered from the MBean server");
35     }
36
37     @Override
38     public ObjectName preRegister(MBeanServer server, ObjectName name) throws Exception {
39         // Optionally modify the name
40         return name;
41     }
42 }

```

Листинг 2: Класс для подсчёта количества кликов

```

1  @Getter
2  @Setter
3  @ManagedBean(name = "counter", eager = true)
4  public class CounterBean implements CounterBeanMBean {
5      private static final Logger logger = Logger.getLogger(CounterBean.class.getName());
6
7      private int countHits;
8      private int loseHits;
9      private boolean lastHit;
10
11     public CounterBean() {
12         countHits = 0;
13         loseHits = 0;
14         lastHit = true;
15     }
16
17     @Override
18     public void addHit(boolean res) {
19         countHits++;
20         if (!res) {
21             loseHits++;
22             if (!lastHit) {
23                 logger.info("2");
24             }
25         }
26         lastHit = res;
27     }
28
29     @Override
30     public int getCountHits() {
31         return countHits;
32     }
33
34     @Override
35     public int getLoseHits() {
36         return loseHits;
37     }
38
39     @Override
40     public boolean isLastHit() {
41         return lastHit;
42     }
43 }

```

Листинг 3: Класс выступающий главным бином

```

1 @Getter
2 @Setter
3 @Named(value = "user")
4 @ApplicationScoped
5 public class UserDataBean {
6     private static final Logger logger = Logger.getLogger(UserDataBean.class.getName());
7
8     @Inject
9     private CounterBean counterBean;
10    @Inject
11    private ClickIntervalBean clickIntervalBean;
12    private Double x = (double) 0;
13    private Double y = (double) 0;
14    private Double r = (double) 0;
15    private Boolean hit;
16    private List<UserData> resultList;
17
18    @PostConstruct
19    public void init() {
20        MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
21
22        try {
23            ObjectName nameCounter = new ObjectName("UserDataBean:name=CounterBean");
24            mbs.registerMBean(counterBean, nameCounter);
25
26            ObjectName nameInterval = new ObjectName("UserDataBean:name=ClickIntervalBean");
27            mbs.registerMBean(clickIntervalBean, nameInterval);
28        } catch (Exception e) {
29            e.printStackTrace();
30        }
31        logger.info("JConsole for manage");
32    }
33
34
35    private void updateLocal() {
36        SessionFactory sessionFactory = HibernateUtil.getSessionFactory();
37        UserDataDao userDataDao = new UserDataDao(sessionFactory);
38        resultList = userDataDao.getUserData();
39    }
40    public void clearTable(){
41        SessionFactory sessionFactory = HibernateUtil.getSessionFactory();
42        UserDataDao userDataDao = new UserDataDao(sessionFactory);
43        userDataDao.clearResultTable();
44        updateLocal();
45    }
46
47    public void saveData() {
48        if ((x >= 0 && x <= r / 2) && (y >= 0 && y <= r)){
49            hit = true;}
50        else if ((x <= 0 && x >= -r) && (y <= 0 && y >= -r/2)){
51            hit = (x>=-r-2*y);}
52        else if ((x >= 0 && x <= r) && (y <= 0 && y >= -r)){
53            hit = (Math.pow(x, 2) + Math.pow(y, 2) <= Math.pow(r, 2));}
54        else{
55            hit = false;}
56        if(r==0){
57            hit = false;
58        }
59
60        if (x == null) {
61            x = (double) 0;

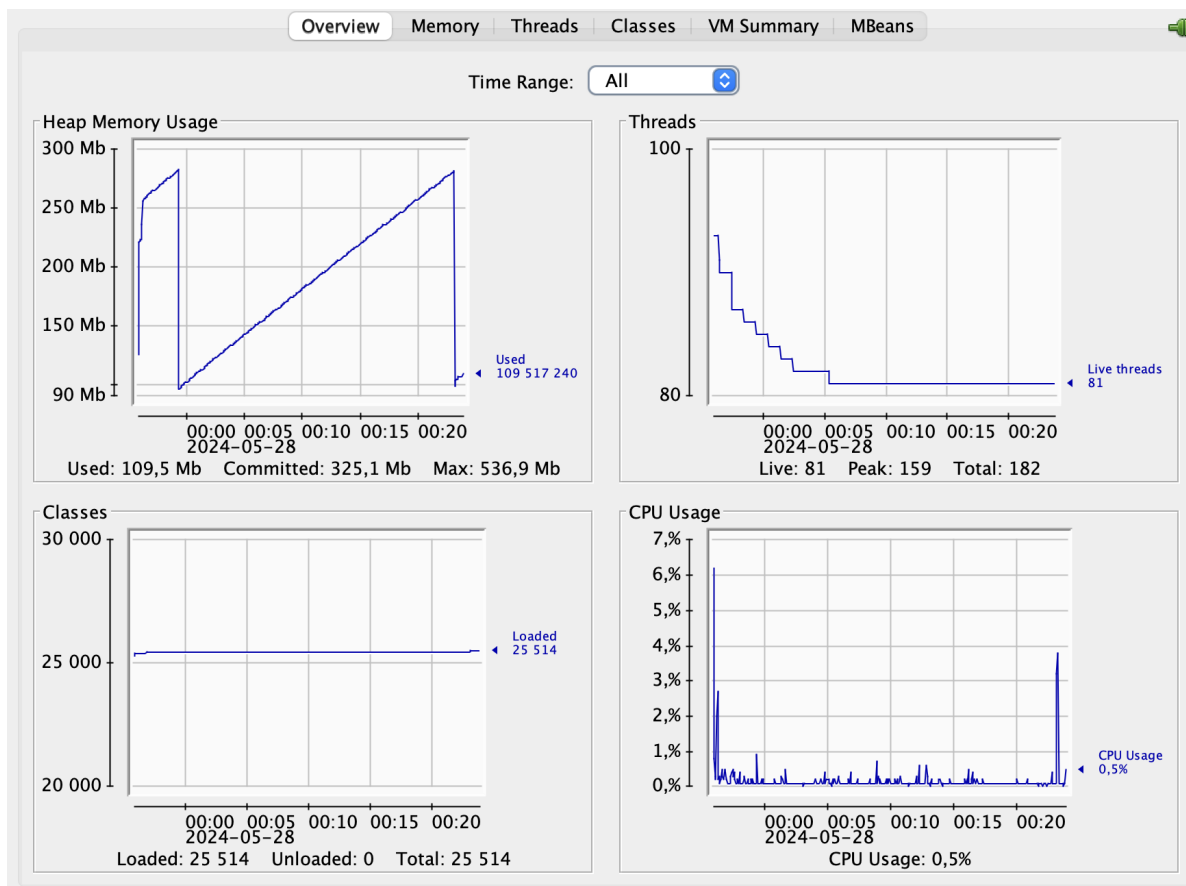
```

```

62     } else if (!(x instanceof Number)) {
63         x = (double) 0;
64     }
65     if (y == null) {
66         y = (double) 0;
67     } else if (!(y instanceof Number)) {
68         y = (double) 0;
69     }
70     if (r == null) {
71         r = (double) 0;
72     } else if (!(r instanceof Number)) {
73         r = (double) 0;
74     }
75
76     counterBean.addHit(hit);
77     clickIntervalBean.addClickTimestamp();
78     logger.info(String.valueOf(clickIntervalBean.getAverageInterval()));
79
80     SessionFactory sessionFactory = HibernateUtil.getSessionFactory();
81     UserDataDao userDataDao = new UserDataDao(sessionFactory);
82     userDataDao.saveUserData(x, y, r, hit);
83     updateLocal();
84 }
85 }

```

### 3 Снятые показания в ходе мониторинга



Скриншот 1. Общий вид

The screenshot shows the JBoss MBeans console with the 'MBeans' tab selected. In the left-hand tree, 'ClickIntervalBean' under 'UserDataBean' is highlighted. The right-hand pane displays the MBeanInfo and Descriptor for this bean.

Name	Value
<b>Info:</b>	
ObjectName	UserDataBean:name=ClickIntervalBean
ClassName	beans.ClickIntervalBean
Description	Information on the management interface of the MBean
<b>Constructor-0:</b>	
Name	beans.ClickIntervalBean
Description	Public constructor of the MBean

Name	Value
<b>Info:</b>	
immutableInfo	true
interfaceClassName	beans.ClickIntervalBeanMBean
mxbean	false

Скриншот 2. Бин интервал

The screenshot shows the JBoss MBeans console with the 'MBeans' tab selected. In the left-hand tree, 'CounterBean' under 'ClickIntervalBean' is highlighted. The right-hand pane displays the MBeanInfo and Descriptor for this bean.

Name	Value
<b>Info:</b>	
ObjectName	UserDataBean:name=CounterBean
ClassName	beans.CounterBean
Description	Information on the management interface of the MBean
<b>Constructor-0:</b>	
Name	beans.CounterBean
Description	Public constructor of the MBean

Name	Value
<b>Info:</b>	
immutableInfo	true
interfaceClassName	beans.CounterBeanMBean
mxbean	false

Скриншот 3. Клик бин

OverviewMemoryThreadsClassesVM SummaryMBeans

VM Summary

вторник, 28 мая 2024 г., 00:29:16 Москва, стандартное время

Connection name:

pid: 88609 jboss-modules.jar -mp /Users/alexalex/Desktop/wildfly-21.0.0.Final/modules org.jboss.as.standalone -Djboss.home.dir=/Users/alexalex/Desktop/wildfly-21.0.0.Final -Djboss.server.base.dir=/Users/alexalex/Desktop/wildfly-21.0.0.Final/standalone

Virtual Machine:

OpenJDK 64-Bit Server VM version 17.0.4.1+7-b469.62

Vendor:

JetBrains s.r.o.

Name:

88609@MacBook-Air-Alex-2.local

Uptime:

34 minutes

Process CPU time:

36,855 seconds

JIT compiler:

HotSpot 64-Bit Tiered Compilers

Total compile time:

17,503 seconds

Live threads:

81

Peak:

159

Daemon threads:

30

Total threads started:

182

Current classes loaded:

25 520

Total classes loaded:

25 520

Total classes unloaded:

0

Current heap size:

146 886 kbytes

Maximum heap size:

524 288 kbytes

Garbage collector:

Name = 'G1 Young Generation', Collections = 43, Total time spent = 0,191 seconds

Garbage collector:

Name = 'G1 Old Generation', Collections = 0, Total time spent = 0,000 seconds

Committed memory:

317 440 kbytes

Pending finalization:

0 objects

Operating System:

Mac OS X 14.1.2

Architecture:

aarch64

Number of processors:

8

Committed virtual memory:

411 155 392 kbytes

Total physical memory:

8 388 608 kbytes

Free physical memory:

66 528 kbytes

Total swap space:

4 194 304 kbytes

Free swap space:

637 888 kbytes

VM arguments:

-D[Standalone] -d32 -Xms64m -Xmx512m -XX:MetaspaceSize=96m -XX:MaxMetaspaceSize=256m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true --add-exports=java.base/sun.nio.ch=ALL-UNNAMED --add-exports=jdk.unsupported/sun.misc=ALL-UNNAMED --add-exports=jdk.unsupported/sun.reflect=ALL-UNNAMED -Dorg.jboss.boot.log.file=/Users/alexalex/Desktop/wildfly-21.0.0.Final/standalone/log/server.log -Dlogging.configuration=file:/Users/alexalex/Desktop/wildfly-21.0.0.Final/standalone/configuration/logging.properties

Class path:

/Users/alexalex/Desktop/wildfly-21.0.0.Final/jboss-modules.jar

Library path:

/Users/alexalex/Library/Java/Extensions/Library/Java/Extensions:/Network/Library/Java/Extensions:/System/Library/Java/Extensions:/usr/lib/java.

Boot class path:

Unavailable

Скриншот 4. Версия

org.jboss.modules.Main (pid 6571)

Sampler Settings

Sample: CPU Memory Stop

Status: CPU sampling in progress

CPU samples Thread CPU time

Results: Snapshot Thread Dump

Name	Total Time	Total Time (CPU)
> Reference Reaper	247 109 ms (100 %)	0,0 ms (- %)
> MSC service thread 1-1	247 109 ms (100 %)	0,0 ms (- %)
> MSC service thread 1-2	247 109 ms (100 %)	0,0 ms (- %)
> MSC service thread 1-3	247 109 ms (100 %)	0,0 ms (- %)
> MSC service thread 1-4	247 109 ms (100 %)	0,0 ms (- %)
> MSC service thread 1-5	247 109 ms (100 %)	0,0 ms (- %)
> MSC service thread 1-6	247 109 ms (100 %)	0,0 ms (- %)
> MSC service thread 1-7	247 109 ms (100 %)	0,0 ms (- %)
> MSC service thread 1-8	247 109 ms (100 %)	0,0 ms (- %)
> ServerDeploymentRepository-temp-threads - 1	247 109 ms (100 %)	0,0 ms (- %)
> ServerService Thread Pool -- 1	247 109 ms (100 %)	0,0 ms (- %)
> ServerService Thread Pool -- 26	247 109 ms (100 %)	0,0 ms (- %)
> DeploymentScanner-threads - 1	247 109 ms (100 %)	379 ms (100 %)
> ServerService Thread Pool -- 41	247 109 ms (100 %)	0,0 ms (- %)
> ConnectionValidator	247 109 ms (100 %)	0,0 ms (- %)
> IdleRemover	247 109 ms (100 %)	0,0 ms (- %)
> management I/O-1	247 109 ms (100 %)	0,0 ms (- %)
> management I/O-2	247 109 ms (100 %)	0,0 ms (- %)
> management Accept	247 109 ms (100 %)	0,0 ms (- %)
> default I/O-1	247 109 ms (100 %)	0,0 ms (- %)
> default I/O-2	247 109 ms (100 %)	0,0 ms (- %)
> default I/O-3	247 109 ms (100 %)	0,0 ms (- %)
> default I/O-4	247 109 ms (100 %)	0,0 ms (- %)
> default I/O-5	247 109 ms (100 %)	106 ms (100 %)
> default I/O-6	247 109 ms (100 %)	0,0 ms (- %)
> default I/O-7	247 109 ms (100 %)	0,0 ms (- %)
> default I/O-8	247 109 ms (100 %)	0,0 ms (- %)
> default I/O-9	247 109 ms (100 %)	0,0 ms (- %)
> default I/O-10	247 109 ms (100 %)	0,0 ms (- %)

Скриншот 5. Показания VisualVM

Attribute values

Name	Value
ClickIntervalBean	1716894034602
ClickIntervalBean	1716894039759
ClickIntervalBean	1716894049445
ClickIntervalBean	1716894272230
ClickIntervalBean	1716894272530
ClickIntervalBean	1716894272822

ClickTimestamps

AverageInterval

20 000, 15 000, 9 000, 14:05

9 771,8182

Discard chart

Скриншот 6. Показания VisualVM для интервального бина





Скриншот 7. Показания VisualVM для количественного бина

## 4 Проверка кода на утечки памяти

Нашли цикл, где поменяли время сна на 0, а так же выставили максимаильный размер кучи 50MB.

```
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    try {
        HttpUnitOptions.setExceptionsThrownOnScriptError(false);
        ServletRunner sr = new ServletRunner();
        sr.registerServlet( resourceName: "myServlet", HelloWorld.class.getName());
        ServletUnitClient sc = sr.newClient();
        int number = 1;
        WebRequest request = new GetMethodWebRequest( urlString: "http://test.meterware.com/myServlet");
        while (true) {
            WebResponse response = sc.getResponse(request);
            System.out.println("Count: " + number++ + response);
            java.lang.Thread.sleep( millis: 0);
        }
    }
}
```

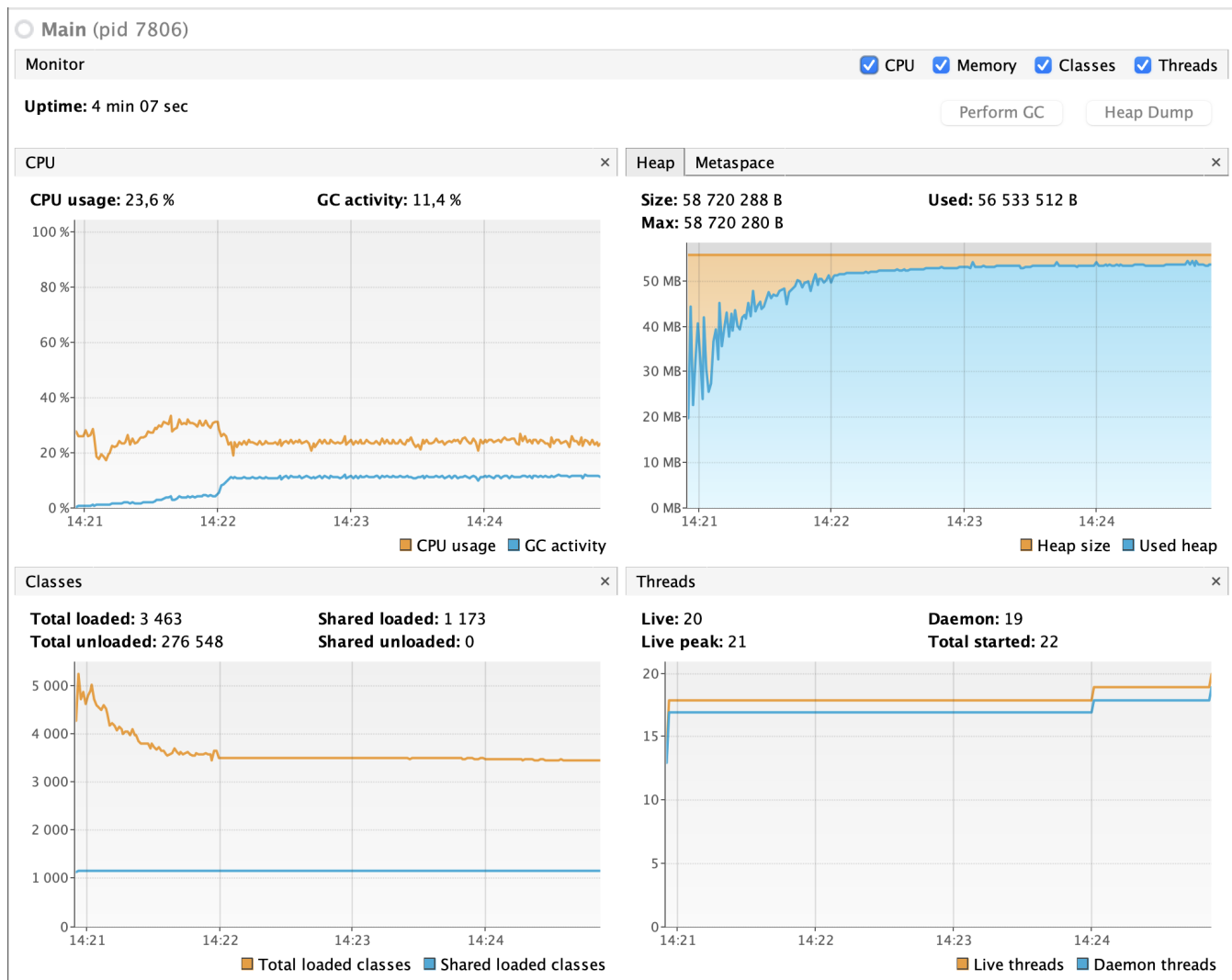
Скриншот 8. Проблемное место

Подождал и вылетела ошибка

```
Count: 262975[ _response = com.meterware.servletunit.ServletUnitHttpResponse@17c0decb]
Count: 262976[ _response = com.meterware.servletunit.ServletUnitHttpResponse@36ef14b0]
Count: 262977[ _response = com.meterware.servletunit.ServletUnitHttpResponse@eb8ed73]
Count: 262978[ _response = com.meterware.servletunit.ServletUnitHttpResponse@2feddaf0]
Count: 262979[ _response = com.meterware.servletunit.ServletUnitHttpResponse@4af8dac0]
Exception in thread "main" java.lang.OutOfMemoryError Create breakpoint : Java heap space
```

Скриншот 9. Ошибка кучи

При профилировании уже сразу была видна проблема



Скриншот 10. Ошибка кучи

Сделал 2 раза HeapDump с небольшим промежутком по времени, чтобы увидеть где подтикает память:

◀ MBeans Buffer Pools JConsole Plugins Visual GC Tracer [heapdump] 14:31:28 x [heapdump] 14:32:01 x

○ Main (pid 8060)

Heap Dump

Summary

Heap		Environment	
Size:	29 759 144 B	System	Mac OS X (14.1.2)
Classes:	3 670	Architecture:	aarch64 64bit
Instances:	640 025	Java Home:	γ/Java/JavaVirtualMachines/openjdk-20.0.1/Contents/Home
Classloaders:	24	Java Version:	20.0.1 2023-04-18
GC Roots:	3 247	Java Name:	penJDK 64-Bit Server VM (20.0.1+9-29, mixed mode, sharing)
Objects Pending for Finalization:	0	Java Vendor:	Oracle Corporation
		JVM Uptime:	0 min 25 sec

Enabled Modules [ show ]

System Properties [ show ]

Classes by Number of Instances [ view all ]

byte[]	119 122	(18,6 %)
java.lang.String	118 133	(18,5 %)
java.util.TreeMap\$Entry	107 219	(16,8 %)
java.lang.Long	54 789	(8,6 %)
java.util.TreeMap	34 042	(5,3 %)

Instances by Size [ view all ]

java.lang.Object[]#12082 : 106 710 items	426 856 B	(1,4 %)
int[]#643 : 30 352 items	121 424 B	(0,4 %)
byte[]#111 : 98 473 items	98 496 B	(0,3 %)
byte[]#42724 [GC root - JNI global] : 97 975 ite	97 992 B	(0,3 %)
byte[]#39 : 70 821 items	70 840 B	(0,2 %)

Classes by Size of Instances [ view all ]

byte[]	12 829 312 B	(43,1 %)
java.util.TreeMap\$Entry	4 288 760 B	(14,4 %)
java.lang.String	2 835 192 B	(9,5 %)
java.util.TreeMap	1 634 016 B	(5,5 %)
java.lang.Long	1 314 936 B	(4,4 %)

Dominators by Retained Size [ view all ]

Retained sizes must be computed first:

Compute Retained Sizes

Скриншот 11. Heap Dump

Main (pid 8060) x

◀ MBeans Buffer Pools JConsole Plugins Visual GC Tracer [heapdump] 14:31:28 x [heapdump] 14:32:01 x

○ Main (pid 8060)

Heap Dump

Summary

Heap		Environment	
Size:	47 915 304 B	System	Mac OS X (14.1.2)
Classes:	3 665	Architecture:	aarch64 64bit
Instances:	882 348	Java Home:	γ/Java/JavaVirtualMachines/openjdk-20.0.1/Contents/Home
Classloaders:	17	Java Version:	20.0.1 2023-04-18
GC Roots:	3 216	Java Name:	penJDK 64-Bit Server VM (20.0.1+9-29, mixed mode, sharing)
Objects Pending for Finalization:	0	Java Vendor:	Oracle Corporation
		JVM Uptime:	0 min 58 sec

Enabled Modules [ show ]

System Properties [ show ]

Classes by Number of Instances [ view all ]

byte[]	240 631	(27,3 %)
java.lang.String	239 649	(27,2 %)
java.util.TreeMap\$Entry	107 241	(12,2 %)
java.lang.Long	54 286	(6,2 %)
java.util.TreeMap	34 046	(3,9 %)

Instances by Size [ view all ]

java.lang.Object[]#4602 : 240 097 items	960 408 B	(2 %)
int[]#853 : 30 352 items	121 424 B	(0,3 %)
byte[]#111 : 98 473 items	98 496 B	(0,2 %)
byte[]#49580 [GC root - JNI global] : 97 975 ite	97 992 B	(0,2 %)
int[]#248 : 22 038 items	88 168 B	(0,2 %)

Classes by Size of Instances [ view all ]

byte[]	27 392 280 B	(57,2 %)
java.lang.String	5 751 576 B	(12 %)
java.util.TreeMap\$Entry	4 289 640 B	(9 %)
java.lang.Object[]	1 638 544 B	(3,4 %)
java.util.TreeMap	1 634 208 B	(3,4 %)

Dominators by Retained Size [ view all ]

Retained sizes must be computed first:

Compute Retained Sizes

Скриншот 12. Heap Dump

Нашёл класс из-за которого все проблемы:

○ Main (pid 8060)

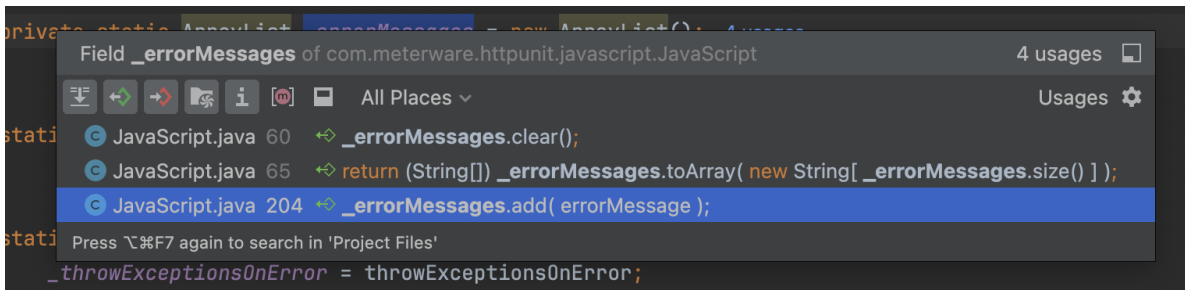
Heap Dump

Object[]#4602 Details: Preview Fields References GC Root Hierarchy

Name	Size	Retained (sort to get)
java.lang.Object[]#4602 : 240 097 items	960 408 B (2 %)	n/a
> <items>		
> <references>		
elementData in java.util.ArrayList#39 : 211 448 elements	24 B (0 %)	n/a
static _errorMessages in class com.meterware.httpunit.javascript.JavaScript : JavaScript	72 B (0 %)	n/a

Скриншот 13. err msg

В данный список добавление только в одном месте



Скриншот 14. add

Есть смысл очищать данные при помощи следующего метода:

```
static void clearErrorMessages() { _errorMessages.clear(); }
```

Скриншот 15. clear

```
public static void main(String[] args) {
    try {
        HttpUnitOptions.setExceptionsThrownOnScriptError(false);
        ServletRunner sr = new ServletRunner();
        sr.registerServlet( resourceName: "myServlet", HelloWorld.class.getName());
        ServletUnitClient sc = sr.newClient();
        int number = 1;
        WebRequest request = new GetMethodWebRequest( urlString: "http://test.meterware.com/myServlet");
        while (true) {
            WebResponse response = sc.getResponse(request);
            System.out.println("Count: " + number++ + response);
            java.lang.Thread.sleep( millis: 0);
            HttpUnitOptions.clearScriptErrorMessages();
        }
    }
}
```

Скриншот 16. clear

После локализации вышел следующий результат



Скриншот 17. res

## 5 Вывод

В ходе данной лабораторной работы я ознакомился с утилитами JConsole и VisualVM для мониторинга и профилирования Java приложений. Также я научился локализовывать и устранять проблемы, связанные с производительностью приложения.