

Федеральное государственное автономное образовательное учреждение высшего  
образования «Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

**Лабораторная работа №4**  
по дисциплине «Основы программной инженерии»

Вариант: **1580**

**Преподаватель:**  
Цопа Евгений Алексеевич

**Выполнил:** Барсуков Максим  
**Группа:** Р3215

Санкт-Петербург, 2024

# **Оглавление**

1.	Задание.....	3
2.	Выполнение.....	4
1.	Исходный код разработанных MBean .....	4
2.	JConsole .....	8
3.	VisualVM .....	12
4.	Исследование программы на утечки памяти.....	17
3.	Вывод.....	23

# 1. Задание

## Вариант: 1580

### Лабораторная работа #4

Вариант 1580

#### **Внимание! У разных вариантов разный текст задания!**

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 3 "промаха" подряд, разработанный MBean должен отправлять оповещение об этом событии.
- MBean, определяющий процентное отношение "попаданий" к общему числу кликов пользователя по координатной плоскости.

2. С помощью утилиты JConsole провести мониторинг программы:

- Снять показания MBean-классов, разработанных в ходе выполнения задания 1.
- Определить время (в мс), прошедшее с момента запуска виртуальной машины.

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 проблем 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.

Студент должен обеспечить возможность воспроизведения процесса поиска и локализации проблемы по требованию преподавателя.

## 2. Выполнение

<https://github.com/maxbarsukov-itmo/mispi-4>



### 1. Исходный код разработанных MBean

- web.beans. HitRatioMBean.java

```
1 package web.beans;  
2  
3 public interface HitRatioMBean {  
4     int getTotalAttempts();  
5     int getTotalHits();  
6     double getHitRatio();  
7 }
```

- web.beans. AttemptStatsMBean.java

```
1 package web.beans;  
2  
3 public interface AttemptStatsMBean {  
4     int getTotalAttempts();  
5     int getTotalMisses();  
6     void checkForConsecutiveMisses();  
7 }
```

- `web.beans.HitRatio.java`

```
1 package web.beans;
2
3 import jakarta.enterprise.context.SessionScoped;
4 import jakarta.inject.Named;
5
6 import java.io.Serializable;
7 import java.util.concurrent.atomic.AtomicInteger;
8
9 @Named("hitRatio")
10 @SessionScoped
11 public class HitRatio implements HitRatioMBean, Serializable {
12     private final AtomicInteger totalAttempts = new AtomicInteger();
13     private final AtomicInteger totalHits = new AtomicInteger();
14
15     @Override
16     public int getTotalAttempts() {
17         return totalAttempts.get();
18     }
19
20     @Override
21     public int getTotalHits() {
22         return totalHits.get();
23     }
24
25     @Override
26     public double getHitRatio() {
27         if (totalAttempts.get() == 0) {
28             return 0.0;
29         }
30         return (double) totalHits.get() / totalAttempts.get() * 100.0;
31     }
32
33     public void updateStats(boolean hit) {
34         totalAttempts.incrementAndGet();
35         if (hit) {
36             totalHits.incrementAndGet();
37         }
38     }
39 }
```

- web.beans.HitRatio.java

```
● web.beans.HitRatio.java

1 package web.beans;
2
3 import jakarta.enterprise.context.SessionScoped;
4 import jakarta.inject.Named;
5
6 import javax.management.*;
7 import java.io.Serializable;
8 import java.util.concurrent.atomic.AtomicInteger;
9
10 @Named("attemptStats")
11 @SessionScoped
12 public class AttemptStats implements AttemptStatsMBean, NotificationBroadcaster,
13     Serializable {
14     private final AtomicInteger totalAttempts = new AtomicInteger();
15     private final AtomicInteger totalMisses = new AtomicInteger();
16     private final AtomicInteger consecutiveMisses = new AtomicInteger();
17
18     private final NotificationBroadcasterSupport broadcaster = new
19         NotificationBroadcasterSupport();
20
21     @Override
22     public int getTotalAttempts() {
23         return totalAttempts.get();
24     }
25
26     @Override
27     public int getTotalMisses() {
28         return totalMisses.get();
29     }
30
31     @Override
32     public void checkForConsecutiveMisses() {
33         if (consecutiveMisses.get() == 3) {
34             broadcaster.sendNotification(new Notification(
35                 "consecutive.misses",
36                 this,
37                 System.currentTimeMillis(),
38                 "3 consecutive misses detected."
39             ));
40             consecutiveMisses.set(0);
41         }
42     }
43
44     public void updateAttempt(boolean hit) {
45         totalAttempts.incrementAndGet();
46         if (!hit) {
47             totalMisses.incrementAndGet();
48             consecutiveMisses.incrementAndGet();
49         } else {
50             consecutiveMisses.set(0);
51         }
52         checkForConsecutiveMisses();
53     }
54
55     @Override
56     public void addNotificationListener(NotificationListener listener, NotificationFilter
57         filter, Object handback) throws IllegalArgumentException {
58         broadcaster.addNotificationListener(listener, filter, handback);
59     }
60
61     @Override
62     public void removeNotificationListener(NotificationListener listener) throws
63         ListenerNotFoundException {
64         broadcaster.removeNotificationListener(listener);
65     }
66
67     @Override
68     public MBeanNotificationInfo[] getNotificationInfo() {
69         String[] types = new String[] { "consecutive.misses" };
70         String name = Notification.class.getName();
71         String description = "Notification sent when 3 consecutive misses are recorded";
72         return new MBeanNotificationInfo[] { new MBeanNotificationInfo(types, name,
73             description) };
74     }
75 }
```

- `web.utils.MBeanRegistry.java`

```
1 package web.utils;
2
3 import jakarta.servlet.ServletContextListener;
4 import lombok.experimental.UtilityClass;
5
6 import javax.management.*;
7 import java.lang.management.ManagementFactory;
8 import java.util.HashMap;
9 import java.util.Map;
10
11 @UtilityClass
12 public class MBeanRegistry implements ServletContextListener {
13     private final Map<Class<?>, ObjectName> beans = new HashMap<>();
14
15     public void registerBean(Object bean, String name) {
16         try {
17             var domain = bean.getClass().getPackageName();
18             var type = bean.getClass().getSimpleName();
19             var objectName = new ObjectName(String.format("%s:type=%s,name=%s", domain, type,
name));
20
21             ManagementFactory.getPlatformMBeanServer().registerMBean(bean, objectName);
22             beans.put(bean.getClass(), objectName);
23         } catch (InstanceAlreadyExistsException | MBeanRegistrationException |
24 NotCompliantMBeanException |
25 MalformedObjectNameException ex) {
26             ex.printStackTrace();
27         }
28     }
29
30     public void unregisterBean(Object bean) {
31         if (!beans.containsKey(bean.getClass())) {
32             throw new IllegalArgumentException("Specified bean is not registered.");
33         }
34         try {
35             ManagementFactory.getPlatformMBeanServer().unregisterMBean(beans.get(bean.getClass()));
36         } catch (InstanceNotFoundException | MBeanRegistrationException ex) {
37             ex.printStackTrace();
38         }
39     }
40 }
```

## 2. JConsole

Показания MBean-классов, разработанных в ходе выполнения задания 1:

- Метаданные Mbean'ов:

The screenshot displays two side-by-side Java Monitoring & Management Console (JConsole) windows and their corresponding web-based interfaces.

**Left Window (MBean 1):**

- MBeanInfo:**
  - Name: web.beans.type=AttemptStats.name=attemptStats
  - Description: Information on the management interface of the MBean
  - Constructor:
    - Name: web.beans.AttemptStats
    - Description: Public constructor of the MBean
- Descriptor:**
  - Name: Value
  - Info:
    - immutableInfo: false
    - interfaceClassName: web.beans.AttemptStatsMBean
    - method: false

**Right Window (MBean 1):**

**Максим Барсуков  
Р3215, Вариант 18881**

A plot in the first quadrant showing a green shaded region bounded by a quarter-circle of radius R and a line segment from the origin to (R, R). The region is labeled "S". The plot includes axes labeled X and Y, and tick marks at -2.0, -1.5, -1.0, -0.5, and 0. Below the plot are input fields for X (0.0), Y (0.00), and R (1.0, 1.5, 2.0, 2.5, 3.0). Buttons for "Отправить" (Send), "Очистить" (Clear), and "Начальная страница" (Home page) are present.

**Left Window (MBean 2):**

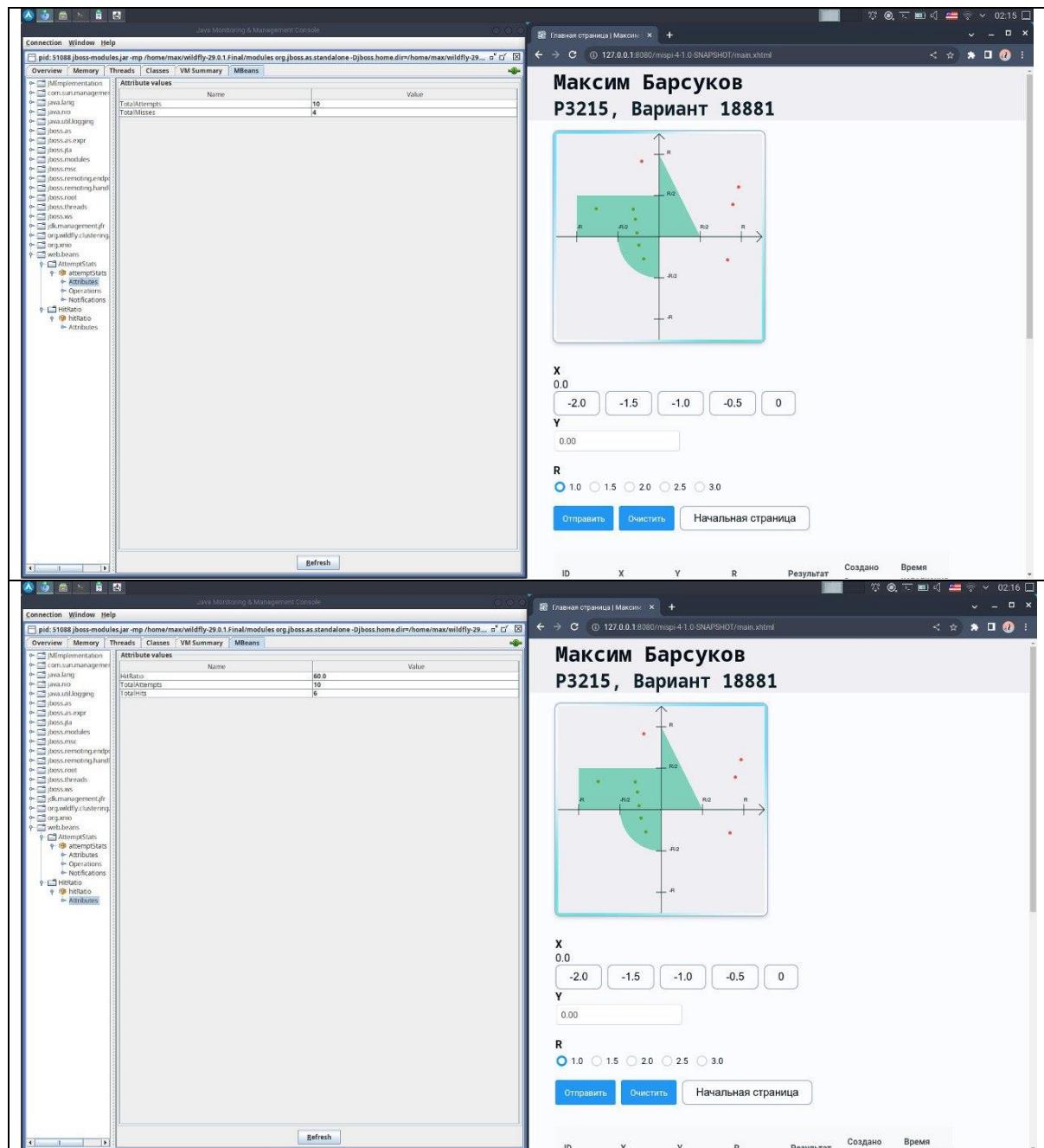
- MBeanInfo:**
  - Name: web.beans.type=HitRatio.name=hitRatio
  - Description: Information on the management interface of the MBean
  - Constructor:
    - Name: web.beans.HitRatio
    - Description: Public constructor of the MBean
- Descriptor:**
  - Name: Value
  - Info:
    - immutableInfo: true
    - interfaceClassName: web.beans.HitRatioMBean
    - method: false

**Right Window (MBean 2):**

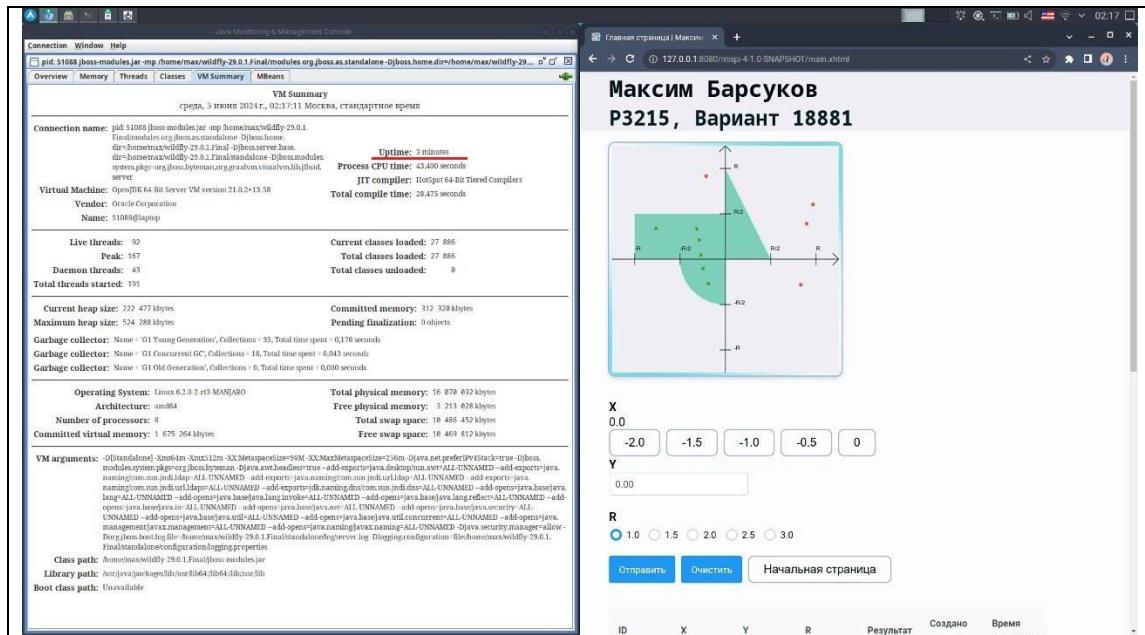
**Максим Барсуков  
Р3215, Вариант 18881**

A plot in the first quadrant showing a green shaded region bounded by a quarter-circle of radius R and a line segment from the origin to (R, R). The region is labeled "S". The plot includes axes labeled X and Y, and tick marks at -2.0, -1.5, -1.0, -0.5, and 0. Below the plot are input fields for X (0.0), Y (0.00), and R (1.0, 1.5, 2.0, 2.5, 3.0). Buttons for "Отправить" (Send), "Очистить" (Clear), and "Начальная страница" (Home page) are present.

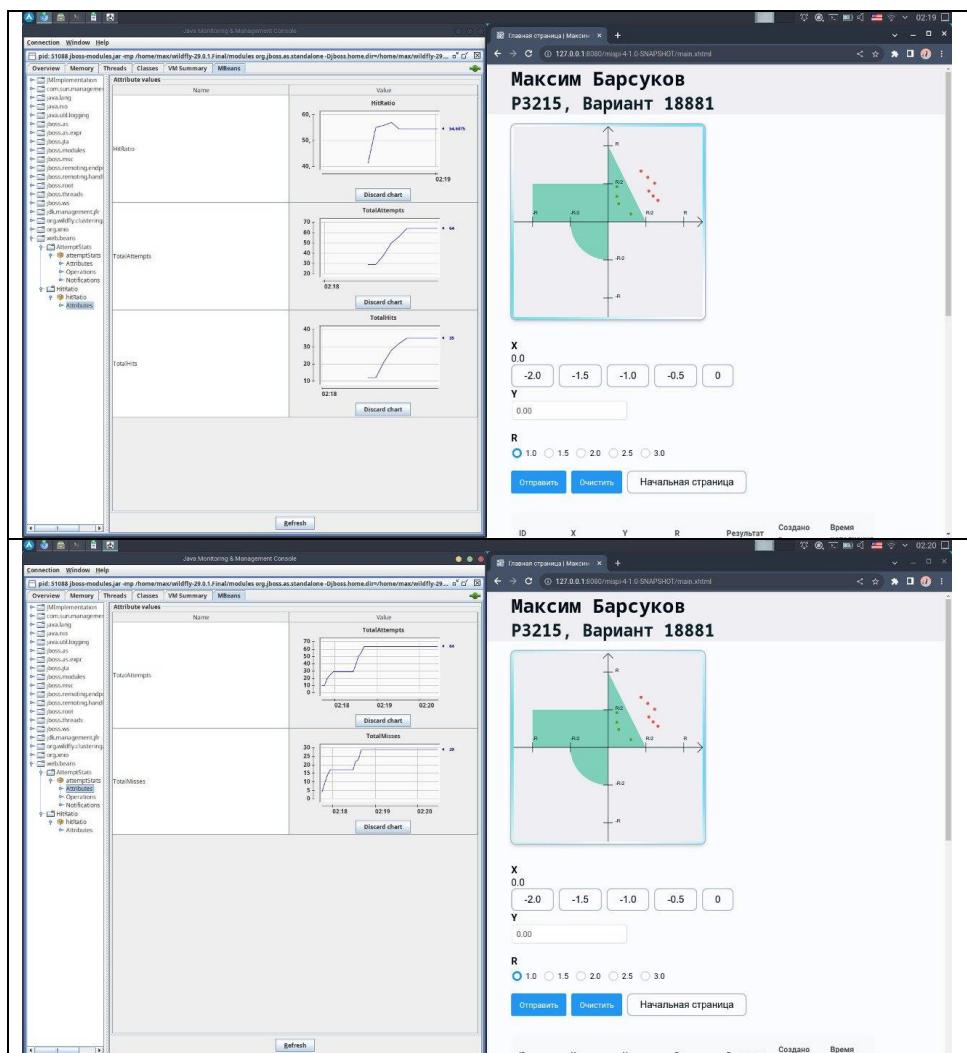
- Показания Mbean'ов:



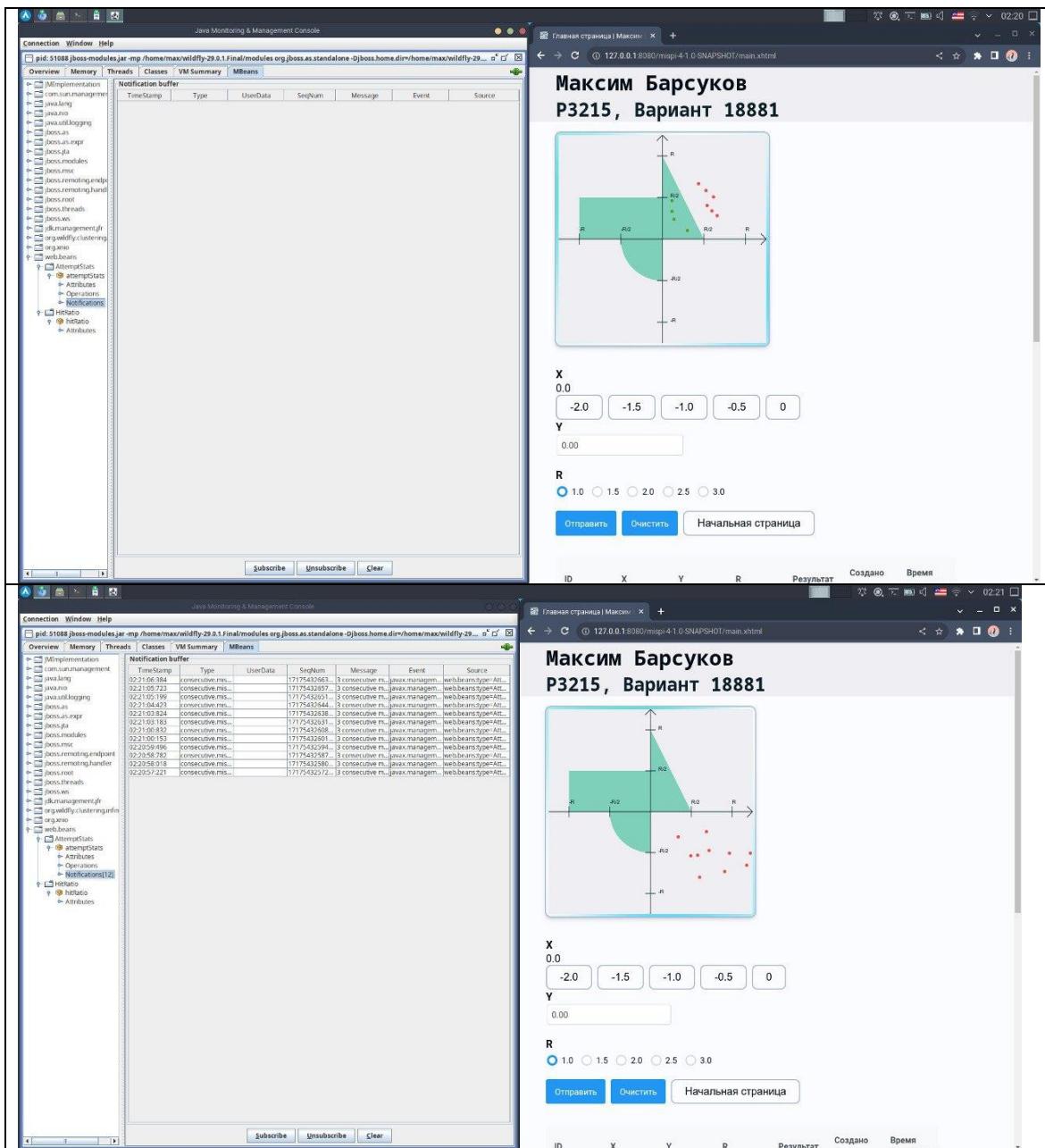
- **VM Summary** и время, прошедшее с момента запуска виртуальной машины:



- Графики:



- Уведомления:



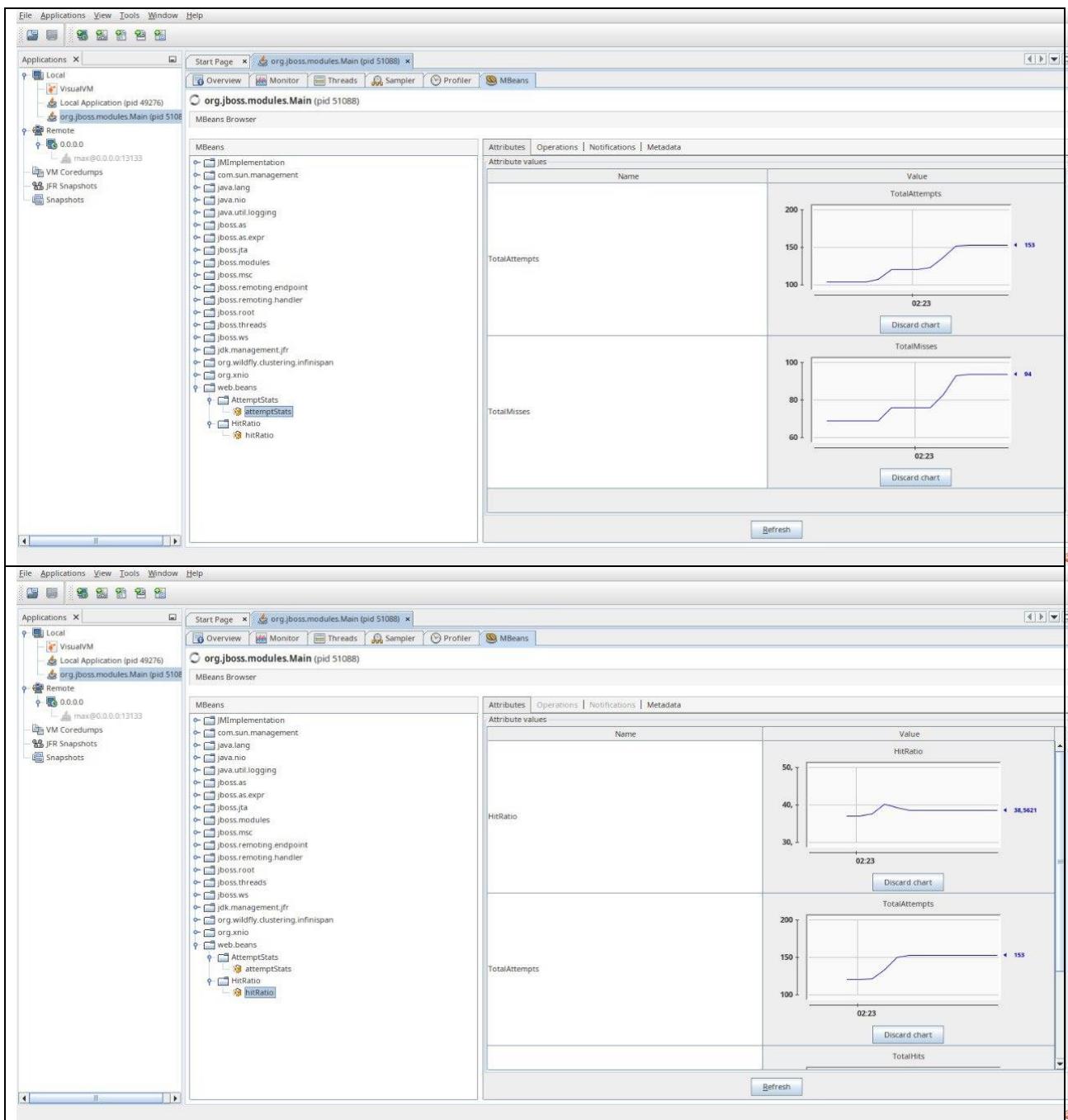
- Выводы по результатам мониторинга:

В ходе мониторинга с использованием утилиты JConsole было определено, что:

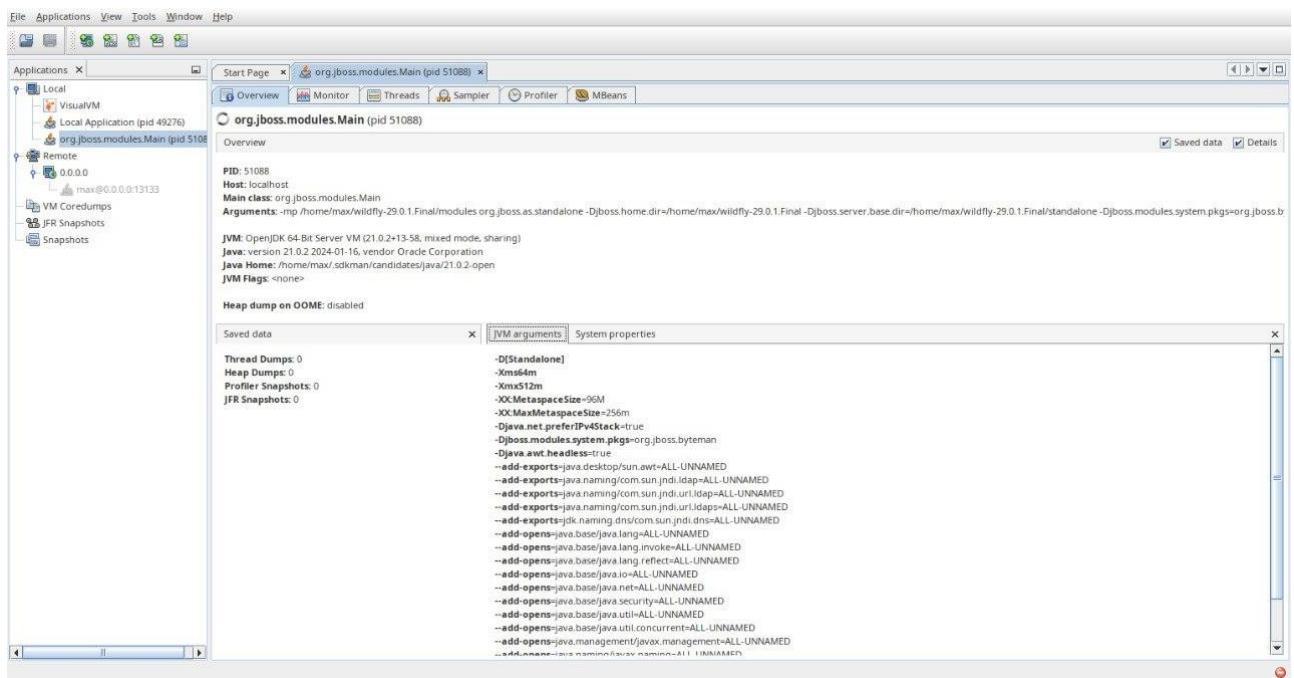
- Узнать время работы JVM можно с помощью Uptime в VM Stats.
- MBean HitRatio и AttemptStats были успешно разработаны и зарегистрированы. В случае трех последовательных промахов MBean AttemptStats отправляет уведомления, которые были получены с помощью JConsole. Таким образом, было зарегистрировано и протестировано получение уведомлений от MBean, что позволяет оперативно реагировать на события, что является важной частью мониторинга. Аналогично с HitRatio MBean.

### 3. VisualVM

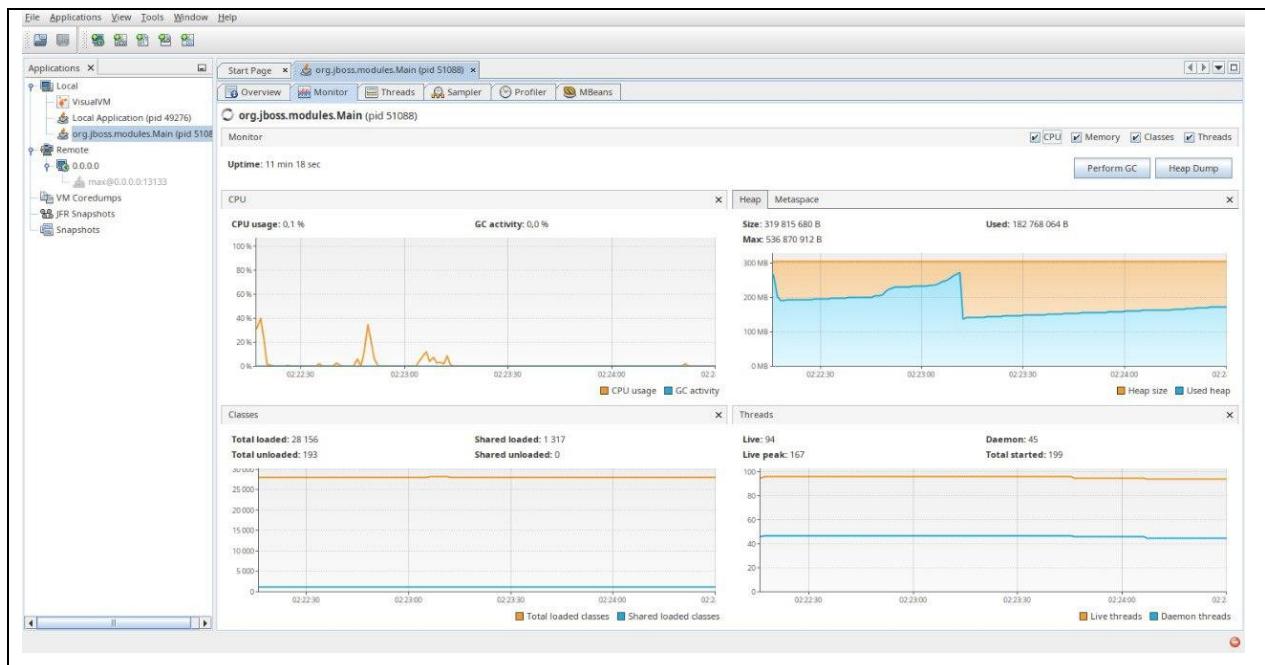
- Снять график изменения показаний MBean-классов:



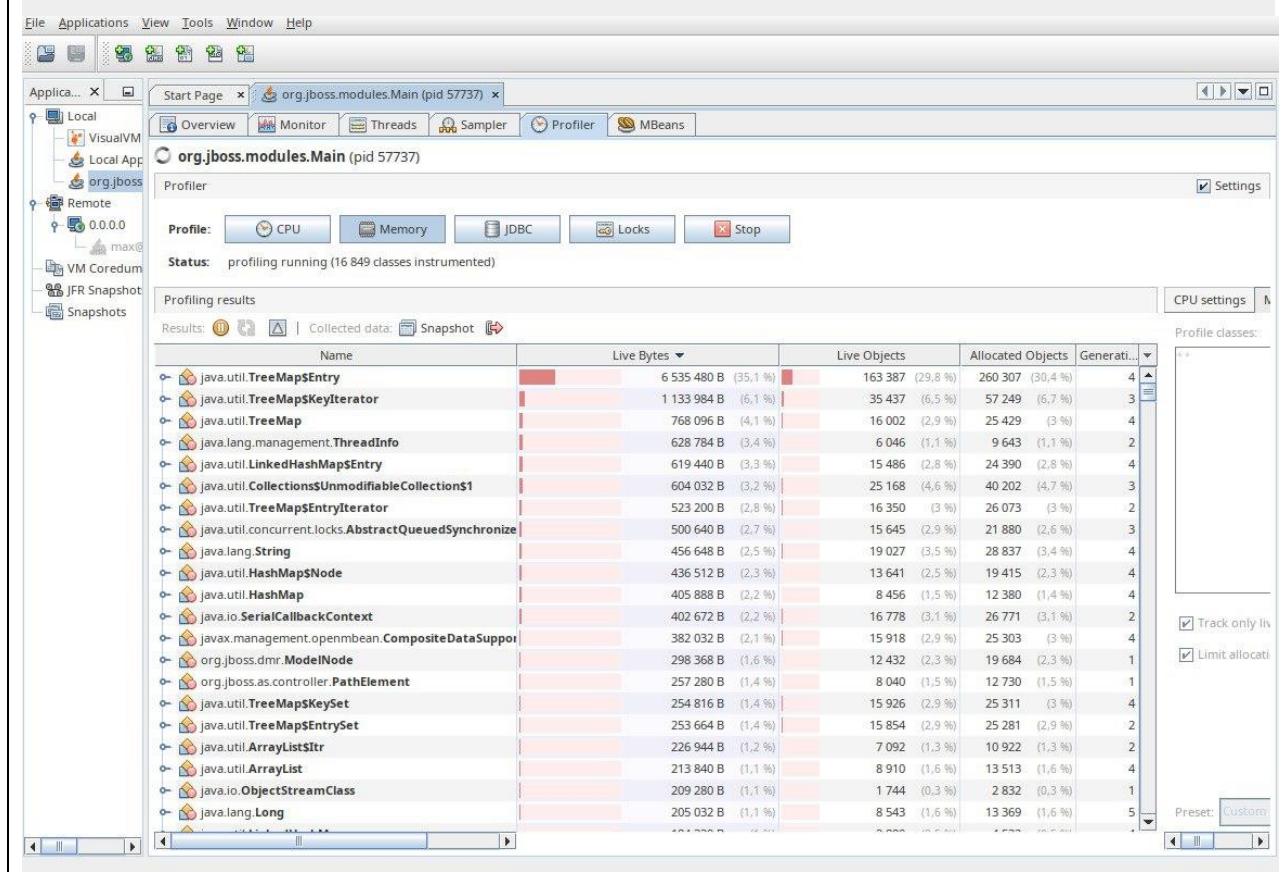
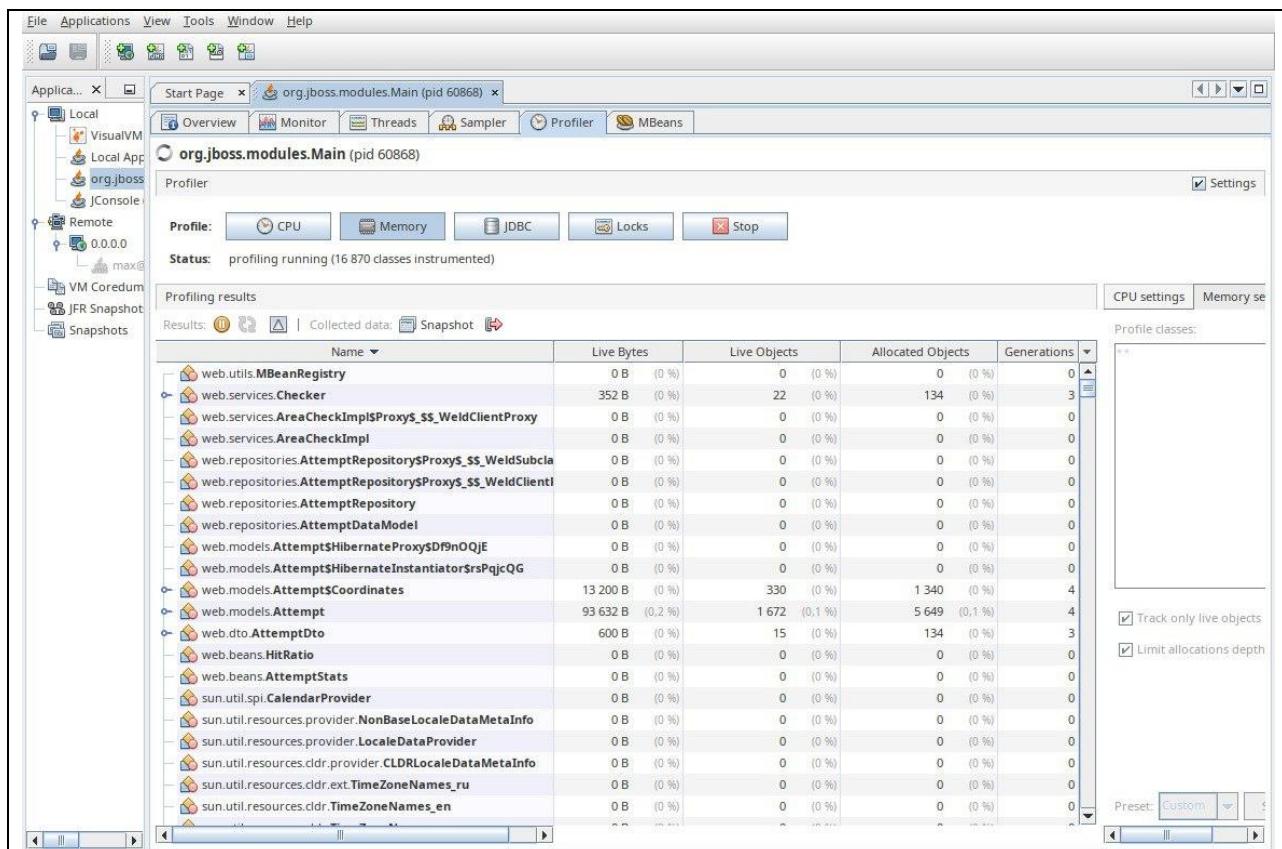
## • JVM Overview:

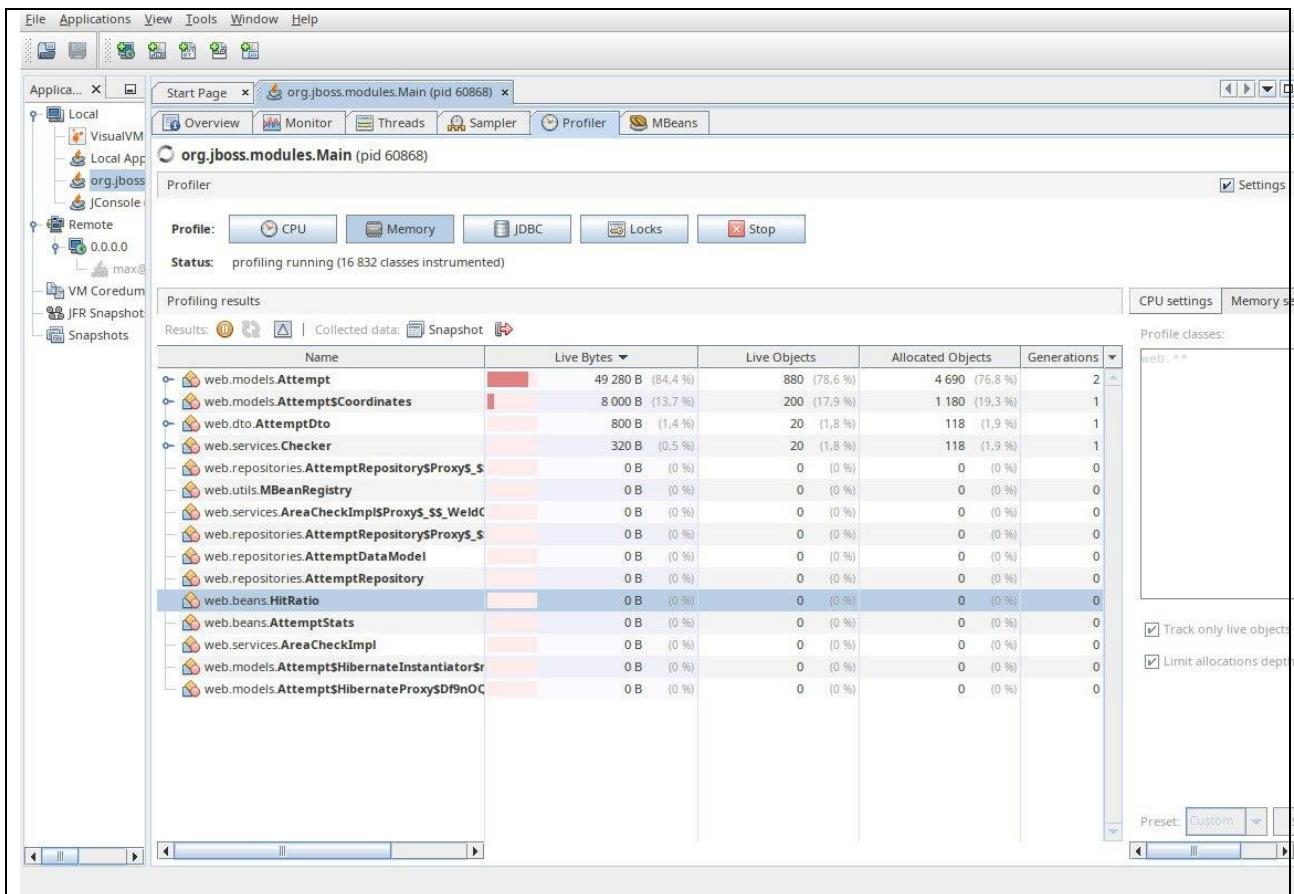


## • Мониторинг:



- Определить имя класса, объекты которого занимают наибольший объём памяти JVM; определить пользовательский класс, в экземплярах которого находятся эти объекты.





- Выводы по результатам мониторинга и профилирования:

Изменения показаний MBean-классов с течением времени:

- HitRatio MBean: На графике HitRatio видны изменения количества точек в базе данных приложения, количество попаданий и процент попаданий. График показывает динамику добавления попавших и не попавших точек. Видно, что процент "попаданий" варьируется с течением времени, что отражает взаимодействие пользователя с системой (создание точек на графике).
- AttemptStats MBean: График AttemptStats показывает общее число установленных пользователем точек, а также число точек, не попадающих в область и также меняется со временем.

Определение **имени класса**, объекты которого занимают **наибольший объём** памяти JVM; определение пользовательского класса, в экземплярах которого находятся эти объекты.

- На основе профилирования памяти видно, что больше всего памяти занимают значения коллекции TreeMap, на втором месте итераторы – всего объекты класса TreeMap занимают ~45%.
- Больше всего памяти из пользовательских классов занимают объекты Attempt – 84% от пользовательских классов. Однако в масштабе всего приложения эти объекты занимают примерно 0.2%.

## 4. Исследование программы на утечки памяти

### Избыточный вызов метода

В процессе анализа производительности приложения была обнаружена проблема, связанная с избыточным вызовом метода `java.lang.Thread.sleep(200)`. Данный метод вызывает приостановку выполнения потока на 200 миллисекунд, что приводит к ненужным задержкам и снижению общей производительности системы.

```
public static void main(String[] args) {
    try {
        HttpUnitOptions.setExceptionsThrownOnScriptError(throwExceptions: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:200);
        }
    } catch (InterruptedException ex) {
        Logger.getLogger(name:"global").log(Level.SEVERE, msg:null, ex);
    } catch (MalformedURLException ex) {
        Logger.getLogger(name:"global").log(Level.SEVERE, msg:null, ex);
    }
}
```

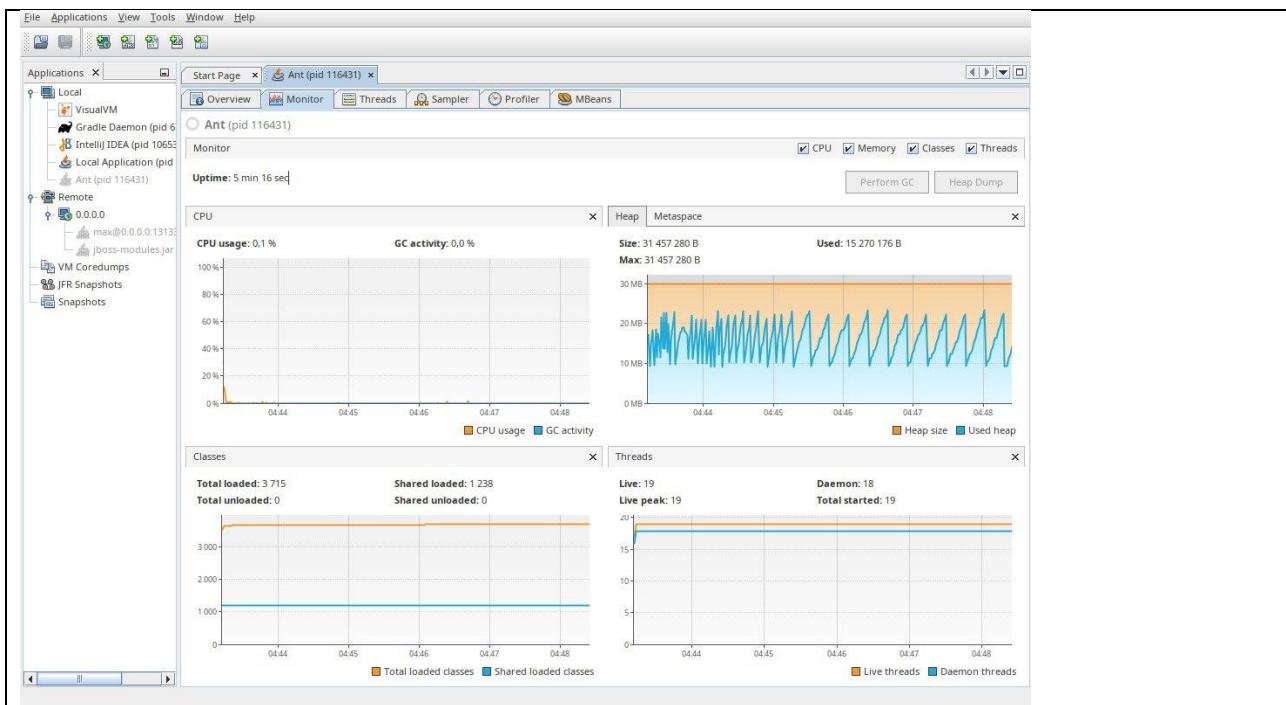
Метод не несёт никакой функциональной нагрузки и является избыточным – поэтому его удаление не повлияет на корректность работы приложения, но повысит производительность за счет устранения задержек.

```
public static void main(String[] args) {
    try {
        HttpUnitOptions.setExceptionsThrownOnScriptError(throwExceptions: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(200);
        }
    } catch (InterruptedException ex) {
        // Logger.getLogger("global").log(Level.SEVERE, null, ex);
    } catch (MalformedURLException ex) {
        Logger.getLogger(name:"global").log(Level.SEVERE, msg:null, ex);
    }
}
```

## Утечка

Установим максимальный размер кучи в 30Мб с помощью `-Xmx30m` и запустим программу и посмотрим, как она себя ведет, в VisualVM.

### 1. Мониторинг, видна очевидная переработка GC:



### 2. Программа падает через ~5 минут, независимо от выделенной памяти для кучи:

```
[java] Count: 66234[ _response = com.meterware.servletunit.ServletUnitHttpResponse@30039c0]
[java] Count: 66235[ _response = com.meterware.servletunit.ServletUnitHttpServletResponse@1221b1d6]
[java] Count: 66236[ _response = com.meterware.servletunit.ServletUnitHttpServletResponse@66f169dd]
[java] Count: 66237[ _response = com.meterware.servletunit.ServletUnitHttpServletResponse@12b7b04d]
[java] Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
[java]     at java.base/java.util.Arrays.copyOf((Arrays.java:3745)
[java]     at java.base/java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractStringBuilder.java:172)
[java]     at java.base/java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:538)
[java]     at java.base/java.lang.StringBuilder.append(StringBuilder.java:174)
[java]     at java.base/java.lang.StringBuilder.append(StringBuilder.java:168)
[java]     at java.base/java.lang.Throwable.printStackTrace(Throwable.java:662)
```

### 3. С помощью Heap Dump найдем объекты, занимающие большую часть памяти.

**Top-level statistics:**

Category	Value	Description
Size:	6 892 120 B	Total heap size
Classes:	3 849	Number of distinct classes
Instances:	168 516	Total number of objects
Classloaders:	25	Number of classloaders
GC Roots:	3 450	Number of GC roots
Objects Pending for Finalization:	0	Number of objects pending finalization

**Environment:**

System	Linux (6.2.0-2-rt3-MANJARO)
Architecture	amd64 64bit
Java Home:	/home/max/.sdkman/candidates/java/21.0.2-open
Java Version:	21.0.2 2024-01-16
Java Name:	OpenJDK 64-Bit Server VM (21.0.2+13-58, mixed mode, sharing)
Java Vendor:	Oracle Corporation
JVM Uptime:	1 min 50 sec

**Enabled Modules:** [ show ]

**System Properties:** [ show ]

**Classes by Number of Instances:** [ view all ]

Class	Count	Percentage
byte[]	32 874	(19.5 %)
java.lang.String	31 310	(18.6 %)
java.util.concurrent.ConcurrentHashMap\$Node	9 008	(5.3 %)
java.lang.Object[]	6 752	(4 %)
java.util.TreeMap\$Entry	6 402	(3.8 %)

**Classes by Size of Instances:** [ view all ]

Class	Size	Percentage
byte[]	1 899 192 B	(27,6 %)
java.lang.String	751 440 B	(10,9 %)
java.lang.Object	428 936 B	(6,2 %)
java.util.concurrent.ConcurrentHashMap\$Node	288 256 B	(4,2 %)
java.util.TreeMap\$Entry	256 080 B	(3,7 %)

**Instances by Size:** [ view all ]

Object	Count	Size	Percentage
byte[]#10833	125 331 items	125 352 B	(1,8 %)
byte[]#10235	65 536 items	65 552 B	(1 %)
byte[]#19725	37 398 items	37 416 B	(0,5 %)
byte[]#19823	37 353 items	37 376 B	(0,5 %)
java.lang.Object@#6554 [GC root - JNI global]	7 938 items	31 768 B	(0,5 %)

**Dominator by Retained Size:** [ view all ]

Retained sizes must be computed first:

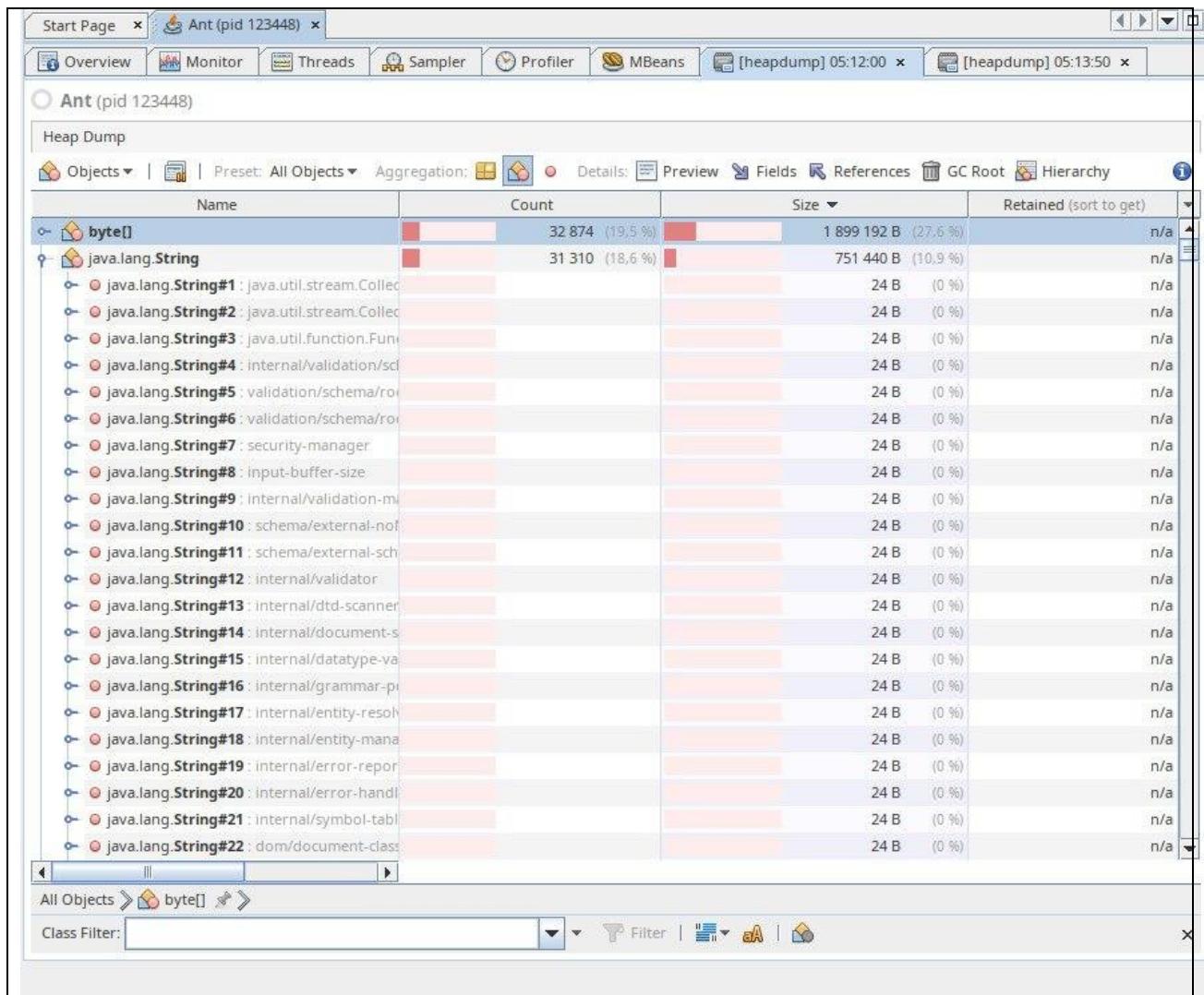
[Compute Retained Sizes](#)

**Heap Dump Details:**

Name	Count	Size	Retained (sort to get)
byte[]	32 874 (19.5 %)	1 899 192 B (27.6 %)	n/a
java.lang.String	31 310 (18.6 %)	751 440 B (10.9 %)	n/a
java.lang.Object[]	6 752 (4 %)	428 936 B (6.2 %)	n/a
java.util.concurrent.ConcurrentHashMap\$Node	9 008 (5.3 %)	288 256 B (4.2 %)	n/a
java.util.TreeMap\$Entry	6 402 (3.8 %)	256 080 B (3.7 %)	n/a
java.lang.reflect.Method	1 907 (1,1 %)	167 816 B (2,4 %)	n/a
java.util.HashMap\$Node[]	1 842 (1,1 %)	167 280 B (2,4 %)	n/a
char[]	343 (0,2 %)	147 136 B (2,1 %)	n/a
java.util.LinkedHashMap\$Entry	3 658 (2,2 %)	146 320 B (2,1 %)	n/a
java.util.HashMap\$Node	4 156 (2,5 %)	132 992 B (1,9 %)	n/a
java.util.TreeMap	2 009 (1,2 %)	96 432 B (1,4 %)	n/a
java.util.LinkedHashMap	1 374 (0,8 %)	87 936 B (1,3 %)	n/a
java.lang.Long	3 541 (2,1 %)	84 984 B (1,2 %)	n/a
int[]	625 (0,4 %)	84 392 B (1,2 %)	n/a
sun.util.locale.LocaleObjectCache\$CacheEntry	2 091 (1,2 %)	83 640 B (1,2 %)	n/a
java.util.concurrent.ConcurrentHashMap\$Node	116 (0,1 %)	79 776 B (1,2 %)	n/a
org.apache.tools.ant.UnknownElement	1 057 (0,6 %)	76 104 B (1,1 %)	n/a
java.lang.ref.SoftReference	1 730 (1 %)	69 200 B (1 %)	n/a
java.util.Hashtable\$Entry	2 130 (1,3 %)	68 160 B (1 %)	n/a
java.lang.invoke.MemberName	1 446 (0,9 %)	57 840 B (0,8 %)	n/a
java.util.ImmutableCollections\$List12	2 246 (1,3 %)	53 904 B (0,8 %)	n/a
java.util.ArrayList	2 245 (1,3 %)	53 880 B (0,8 %)	n/a
org.apache.tools.ant.RuntimeConfigurable	1 057 (0,6 %)	50 736 B (0,7 %)	n/a
javax.management.openmbean.CompositeEL	1 972 (1,2 %)	47 328 B (0,7 %)	n/a

All Objects > byte[] >

Class Filter:



Из графиков использования памяти видно, что

- При работе приложения на каждый запрос создаются экземпляры строк и массивы byte[], затем используются по всему приложению и остаются использованными, но не очищенными в памяти. Хотя GC и отрабатывает корректно, он тратит слишком много ресурсов на очищение памяти, поэтому чтобы убрать излишнюю нагрузку надо устранить утечку;
- Размер кучи постоянно увеличивается, что свидетельствует о проблемах с использованием памяти в программе;
- Через некоторое количество времени получаем ошибку OutOfMemoryError.

Немного исследовав кучу, находим повторяющиеся строки:

```
java.lang.Object[]#3431 : 106 710 items
  <items>
    [0] = java.lang.String#4633 : Script 'document.wr('Hello Document')// failed: TypeError: undefined is not a function. (httpunit; )
    [1] = java.lang.String#4632 : Script 'document.wr('Hello Document')// failed: TypeError: undefined is not a function. (httpunit; )
    [2] = java.lang.String#4631 : Script 'document.wr('Hello Document')// failed: TypeError: undefined is not a function. (httpunit; )
    [3] = java.lang.String#4630 : Script 'document.wr('Hello Document')// failed: TypeError: undefined is not a function. (httpunit; )
    [4] = java.lang.String#4629 : Script 'document.wr('Hello Document')// failed: TypeError: undefined is not a function. (httpunit; )
    [5] = java.lang.String#4628 : Script 'document.wr('Hello Document')// failed: TypeError: undefined is not a function. (httpunit; )

java.lang.Object[]#3431 : 106 710 items
  <items>
  <references>
    elementData in java.util.ArrayList#24 : 82 384 elements
      static _errorMessages in class com.meterware.httpunit.javascript.JavaScript : JavaScript
```

Объекты `_errorMessages` хранятся в `ArrayList`

```
private static ArrayList _errorMessages = new ArrayList();
```

Найдем строчку с добавлением объектов в этот список:

```
} else {
    _errorMessages.add( errorMessage );
}
```

В результате получается накопление `_errorMessages` в списке, за счет чего и получается переполнение памяти. В программе есть функция для очистки `_errorMessage`, однако мы можем увидеть, что на самом деле она не используется

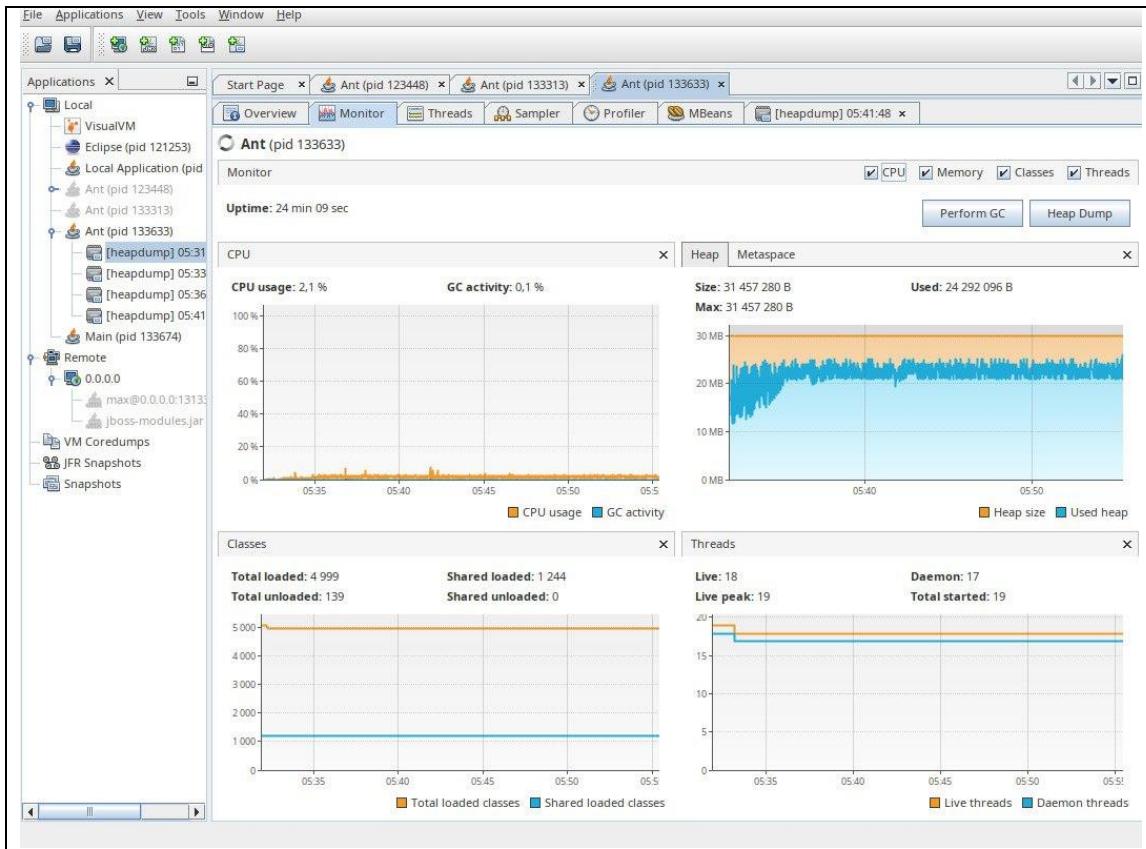
```
static void clearErrorMessages() {
    _errorMessages.clear();
}
public void clearErrorMessages() {
    JavaScript.clearErrorMessages();
}
```

/\*\*  
 \* Clears the accumulated script error messages.  
 \*/  
no usages  
public static void clearScriptErrorMessages() {  
 getScriptingEngine().clearErrorMessages();
}

Решением будет очистка списка с `_errorMessage` после выполнения очередного запроса.

```
while (true) {
    WebResponse response = sc.getResponse(request);
    System.out.println("Count: " + number++ + response);
    HttpUnitOptions.clearScriptErrorMessages();
}
```

Запустим программу. Теперь изменения памяти во времени менее пилообразные, GC работает более оптимально.



Программа работает стабильно и не падает с OutOfMemoryException.

```
[java] Count: 2005303[ _response = com.meterware.servletunit.ServletUnitHttpResponse@46069bbd]
[java] Count: 2005304[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@32dd0028]
[java] Count: 2005305[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@75b8eb36]
[java] Count: 2005306[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@6aa8bc94]
[java] Count: 2005307[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@4f95f8cb]
[java] Count: 2005308[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@16de43aa]
[java] Count: 2005309[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@3bd5b9b4]
[java] Count: 2005310[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@1858b57]
[java] Count: 2005311[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@676e7d4c]
[java] Count: 2005312[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@75275ed6]
[java] Count: 2005313[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@262f5cb1]
[java] Count: 2005314[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@5b40a9c5]
[java] Count: 2005315[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@11bd03de]
[java] Count: 2005316[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@7958beba]
[java] Count: 2005317[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@4f01dff]
[java] Count: 2005318[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@28f0ac3e]
[java] Count: 2005319[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@3ed2ae5f]
[java] Count: 2005320[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@398504c8]
[java] Count: 2005321[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@191d03aa]
[java] Count: 2005322[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@c9c3b58]
[java] Count: 2005323[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@788895bf]
[java] Count: 2005324[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@4360d585]
[java] Count: 2005325[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@7f1a46e8]
[java] Count: 2005326[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@61700b14]
[java] Count: 2005327[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@11678d8]
[java] Count: 2005328[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@35ba37dd]
[java] Count: 2005329[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@4c817ff8]
[java] Count: 2005330[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@1cd7ed48]
[java] Count: 2005331[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@9fa89e6]
[java] Count: 2005332[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@5b5d2574]
[java] Count: 2005333[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@33043228]
[java] Count: 2005334[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@27d2a548]
[java] Count: 2005335[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@1344b5f5]
[java] Count: 2005336[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@3d23abd6]
[java] Count: 2005337[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@1526f7de]
[java] Count: 2005338[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@74c32dcc]
[java] Count: 2005339[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@263aa202]
[java] Count: 2005340[ _response = com.meterware.servletunit.HttpServletUnitHttpResponse@40c4416b]
```

### **3. Вывод**

Во время выполнения лабораторной работы я познакомился с практикой написания MBeans в веб-приложениях, были изучены утилиты для мониторинга и профилирования работы программы JConsole и VisualVM, а также был получен опыт по полученным данным определять утечки памяти и устранять их.