

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: **811**

Преподаватель:

Карташев Владимир Сергеевич

Выполнил:

Кирячек Тимофей

Группа:

P3209

Оглавление

1. Задание	3
2. Выполнение	4
1. Исходный код разработанных MBean	4
2. JConsole	7
3. VisualVM	11
4. Исследование программы на утечки памяти	15
3. Вывод	21

1. Задание

Вариант: 811

Лабораторная работа #4

Вариант 811

Внимание! У разных вариантов разный текст задания!

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

- MBeap, считающий общее число установленных пользователем точек, а также число точек, не попадающих в область. В случае, если количество установленных пользователем точек стало кратно 5, разработанный MBeap должен отправлять оповещение об этом событии.
- MBeap, определяющий площадь получившейся фигуры.

2. С помощью утилиты JConsole провести мониторинг программы:

- Снять показания MBeap-классов, разработанных в ходе выполнения задания 1.
- Определить имя и версию ОС, под управлением которой работает JVM.

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

- Снять график изменения показаний MBeap-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.

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

2. Выполнение

1. Исходный код разработанных MBean

- web.beans.AreaMBean.java

```
package web.beans;

public interface AreaMBean {
    double getArea();
    void updateStats(double r);
}
```

- web.beans.ResultsMBean.java

```
package web.beans;

public interface ResultsMBean {
    int getTotalAttempts();
    int getTotalMisses();
    void updateStats(boolean hit);
}
```

- web.beans.Area.java

```
package web.beans;

import jakarta.enterprise.context.ApplicationScoped;
import jakarta.inject.Named;

import java.io.Serializable;

@Named("area")
@ApplicationScoped
public class Area implements Serializable, AreaMBean {
    private double area = 0;

    @Override
    public double getArea() {
        return area;
    }

    @Override
    public void updateStats(double r) {
        area = ((Math.PI * Math.pow(r / 2, 2) / 4) + ((r * r/2) / 2) + (r * r/2));
    }
}
```

- web.beans.Results.java

```
package web.beans;

import jakarta.enterprise.context.ApplicationScoped;
import jakarta.inject.Named;

import javax.management.*;
import java.io.Serializable;
import java.util.concurrent.atomic.AtomicInteger;
```

```

@Named("results")
@ApplicationScoped
public class Results implements Serializable, NotificationBroadcaster, ResultsMBean {
    private int sequenceNumber = 0;

    private final AtomicInteger totalAttempts = new AtomicInteger();
    private final AtomicInteger totalMisses = new AtomicInteger();

    private final NotificationBroadcasterSupport broadcaster = new
NotificationBroadcasterSupport();

    @Override
    public int getTotalAttempts() {
        return totalAttempts.get();
    }

    @Override
    public int getTotalMisses() {
        return totalMisses.get();
    }

    @Override
    public void updateStats(boolean hit) {
        totalAttempts.incrementAndGet();
        if (!hit) {
            totalMisses.incrementAndGet();
        }

        if (totalAttempts.get() % 5 == 0) {
            broadcaster.sendNotification(new Notification(
                "Total dots count is multiple of 5",
                getClass().getSimpleName(),
                sequenceNumber++,
                "The total count of user-set dots is now multiple of 5!"
            ));
        }
    }

    @Override
    public void addNotificationListener(NotificationListener listener,
NotificationFilter filter, Object handback) throws IllegalArgumentException {
        broadcaster.addNotificationListener(listener, filter, handback);
    }

    @Override
    public void removeNotificationListener(NotificationListener listener) throws
ListenerNotFoundException {
        broadcaster.removeNotificationListener(listener);
    }

    @Override
    public MBeanNotificationInfo[] getNotificationInfo() {
        String[] types = new String[] { AttributeChangeNotification.ATTRIBUTE_CHANGE
};

        String name = AttributeChangeNotification.class.getName();
        String description = "Miss notification";
        MBeanNotificationInfo info = new MBeanNotificationInfo(types, name,
description);
        return new MBeanNotificationInfo[] { info };
    }
}

```

- web.utils.MBeanRegistry.java

```
package web.utils;

import jakarta.servlet.ServletContextListener;
import lombok.experimental.UtilityClass;

import javax.management.*;
import java.lang.management.ManagementFactory;
import java.util.HashMap;
import java.util.Map;

@UtilityClass
public class MBeanRegistry implements ServletContextListener {
    private final Map<Class<?>, ObjectName> beans = new HashMap<>();

    public void registerBean(Object bean, String name) {
        try {
            var domain = bean.getClass().getPackageName();
            var type = bean.getClass().getSimpleName();
            var objectName = new ObjectName(String.format("%s:type=%s,name=%s",
domain, type, name));

            ManagementFactory.getPlatformMBeanServer().registerMBean(bean,
objectName);
            beans.put(bean.getClass(), objectName);
        } catch (InstanceAlreadyExistsException | MBeanRegistrationException |
NotCompliantMBeanException |
            MalformedObjectNameException ex) {
            ex.printStackTrace();
        }
    }

    public void unregisterBean(Object bean) {
        if (!beans.containsKey(bean.getClass())) {
            throw new IllegalArgumentException("Specified bean is not registered.");
        }

        try {
            ManagementFactory.getPlatformMBeanServer().unregisterMBean(beans.get(bean.getClass())
);
        } catch (InstanceNotFoundException | MBeanRegistrationException ex) {
            ex.printStackTrace();
        }
    }
}
```

2. JConsole

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

- **Метаданные Mbean'ов:**

The image displays two screenshots of the JConsole application, specifically the MBeans tab, showing the metadata for two custom MBeans: 'area' and 'results'.

Top Screenshot: 'area' MBean

The left pane shows the tree structure with 'web.beans.area' selected. The right pane displays the MBeanInfo for 'area'.

Name	Value
Info:	
ObjectName	web.beans:type=Area,name=area
ClassName	web.beans.Area
Description	Information on the management interface of the MBean
Constructor-0:	
Name	web.beans.Area
Description	Public constructor of the MBean

The bottom pane shows the Descriptor for 'area'.

Name	Value
Info:	
immutableInfo	true
interfaceClassNa...	web.beans.AreaMBean
mxbean	false

Bottom Screenshot: 'results' MBean

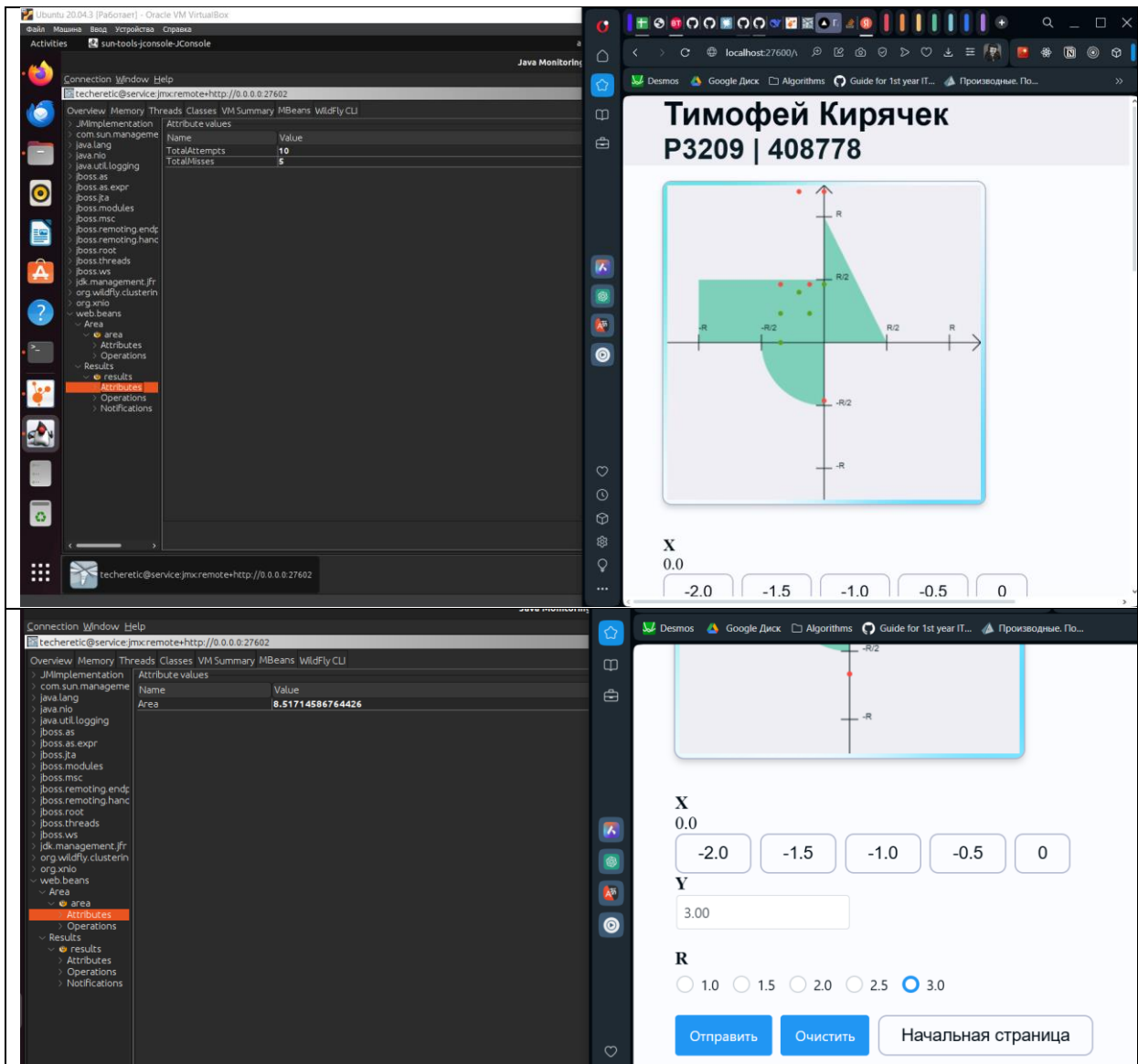
The left pane shows the tree structure with 'web.beans.results' selected. The right pane displays the MBeanInfo for 'results'.

Name	Value
Info:	
ObjectName	web.beans:type=Results,name=results
ClassName	web.beans.Results
Description	Information on the management interface of the MBean
Constructor-0:	
Name	web.beans.Results
Description	Public constructor of the MBean

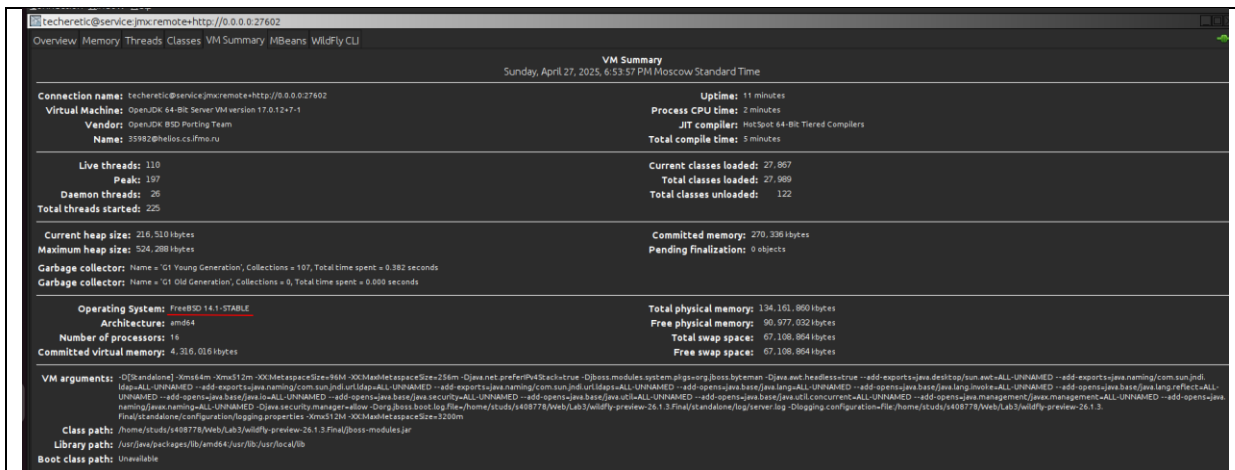
The bottom pane shows the Descriptor for 'results'.

Name	Value
Info:	
immutableInfo	false
interfaceClassNa...	web.beans.ResultsMBean
mxbean	false

- **Показания Мbean'ов:**

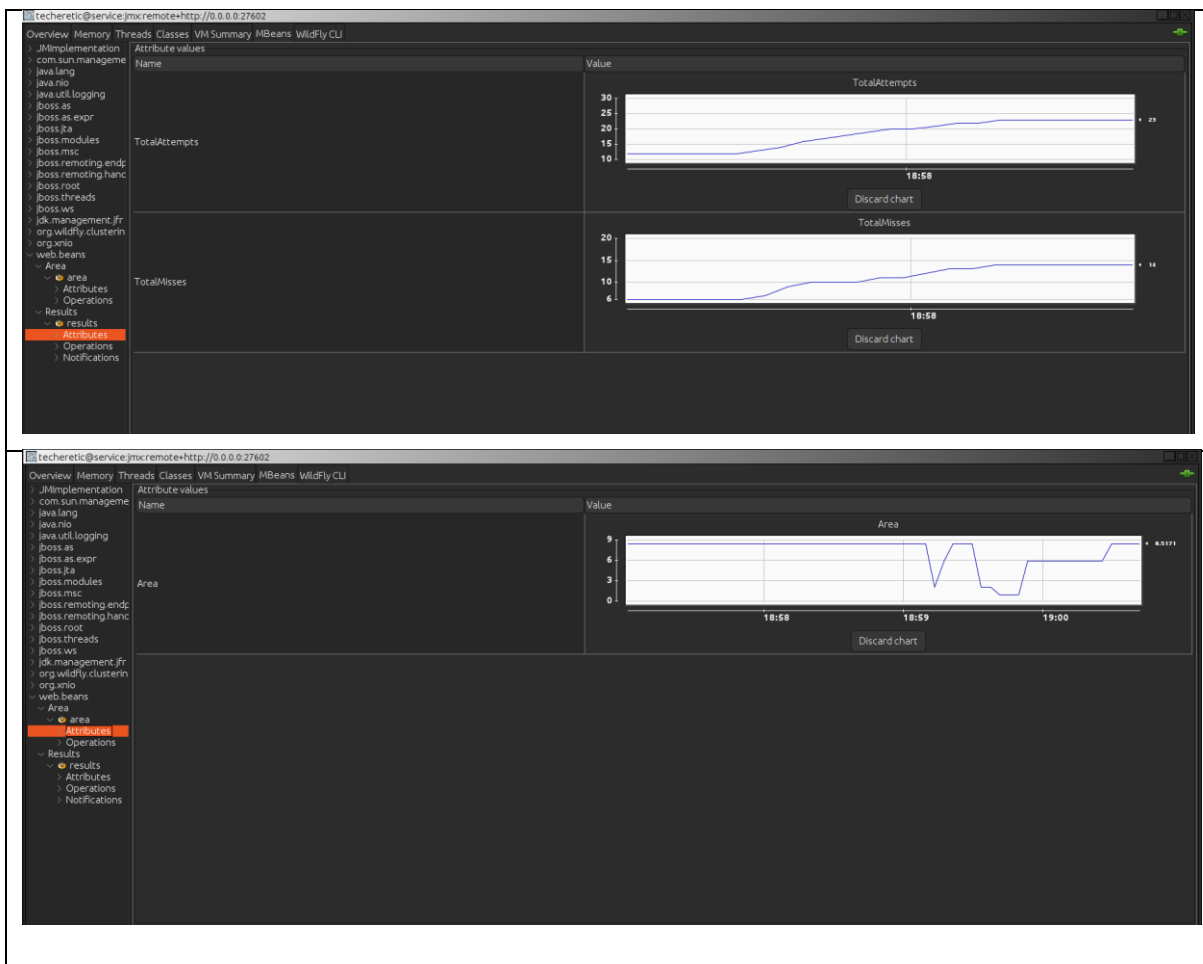


- **VM Summary** и имя и версия ОС, под управлением которой работает JVM.

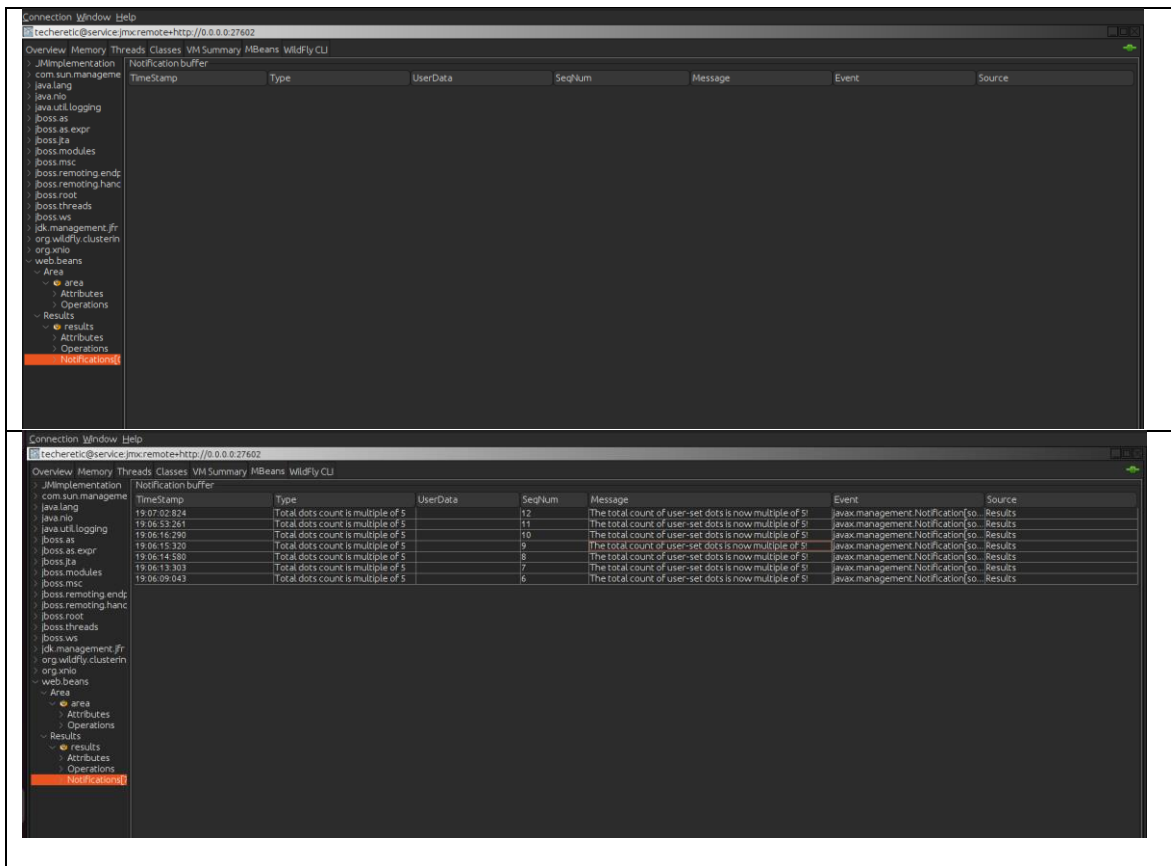


ЭТО FreeBSD 14.1-STABLE

- **Графики:**



- **Уведомления:**



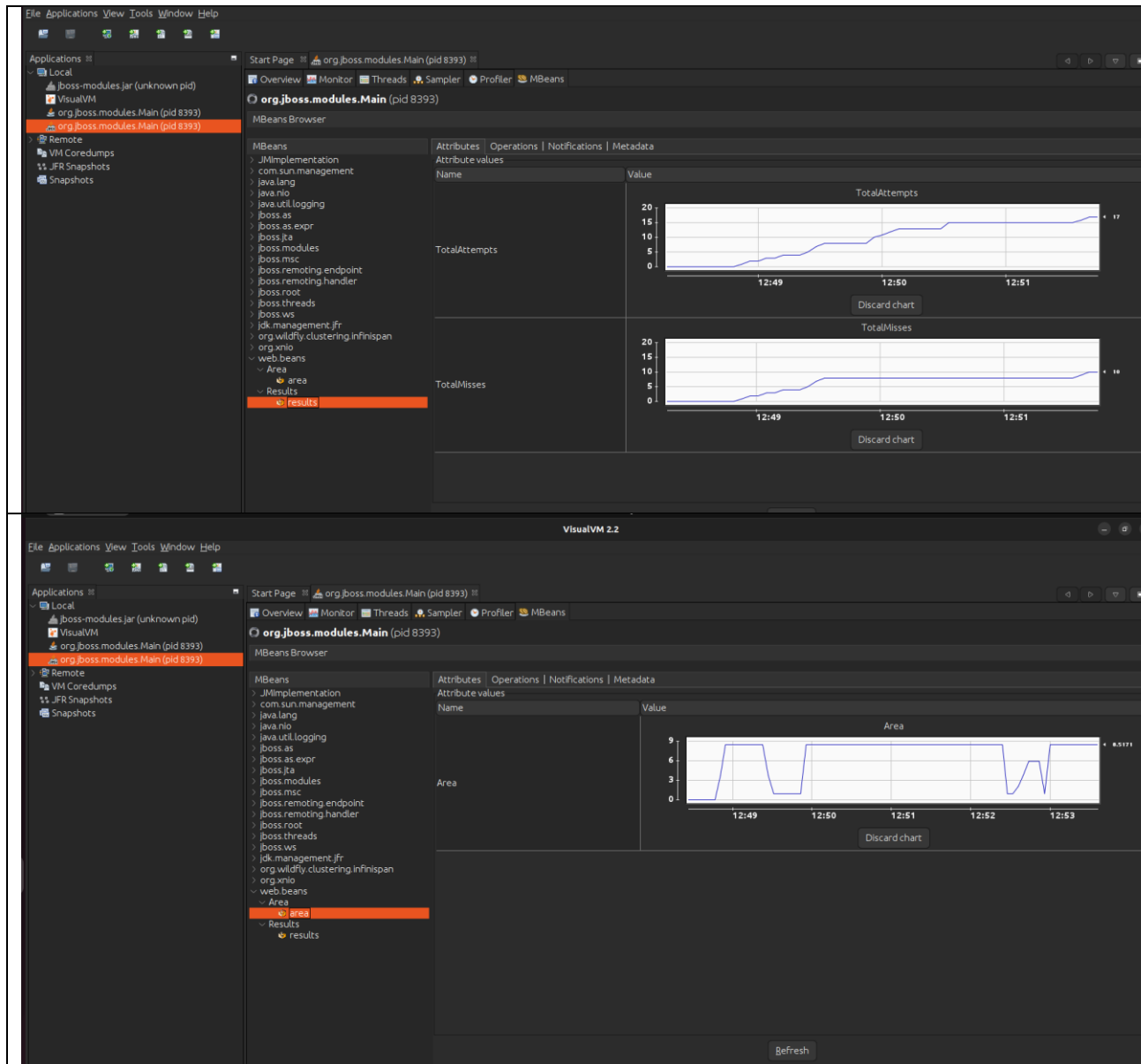
- **Выводы по результатам мониторинга:**

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

- Узнать имя и версию ОС, под которой работает JVM можно в разделе VM Summary/Operation System.
- MBean Area и Results были успешно разработаны и зарегистрированы. Когда общее количество точек становится кратно 5, MBean Results 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.

3. VisualVM

- Снять график изменения показаний MBean-классов:



- **JVM Overview:**

VisualVM 2.2

File Applications View Tools Window Help

Applications ▾

- Local
 - jboss-modules.jar (unknown pid)
 - VisualVM
 - org.jboss.modules.Main (pid 8393)
 - org.jboss.modules.Main (pid 8393)
- Remote
 - VM CoreDumps
 - JFR Snapshots
 - Snapshots

Start Page ▾ org.jboss.modules.Main (pid 8393) ▾

Overview Monitor Threads Sampler Profiler MBeans

org.jboss.modules.Main (pid 8393)

Overview

PID: 8393
Host: localhost
Main class: org.jboss.modules.Main
Arguments: -mp /home/techeretic/Documents/OP/lab3/wildfly-preview-26.1.3.Final/modules/org.jboss.as.standalone -Djboss.home.dir=/home/techeretic/Documents/OP/lab3/wildfly-preview-26.1.3.Final-Djboss.home.dir
JVM: OpenJDK 64-Bit Server VM (21.0.6+7-Ubuntu-122.04.1, mixed mode, sharing)
Java: version 21.0.6 2025-01-21, vendor Ubuntu
Java Home: /usr/lib/jvm/java-21-openjdk-amd64
JVM Flags: <none>

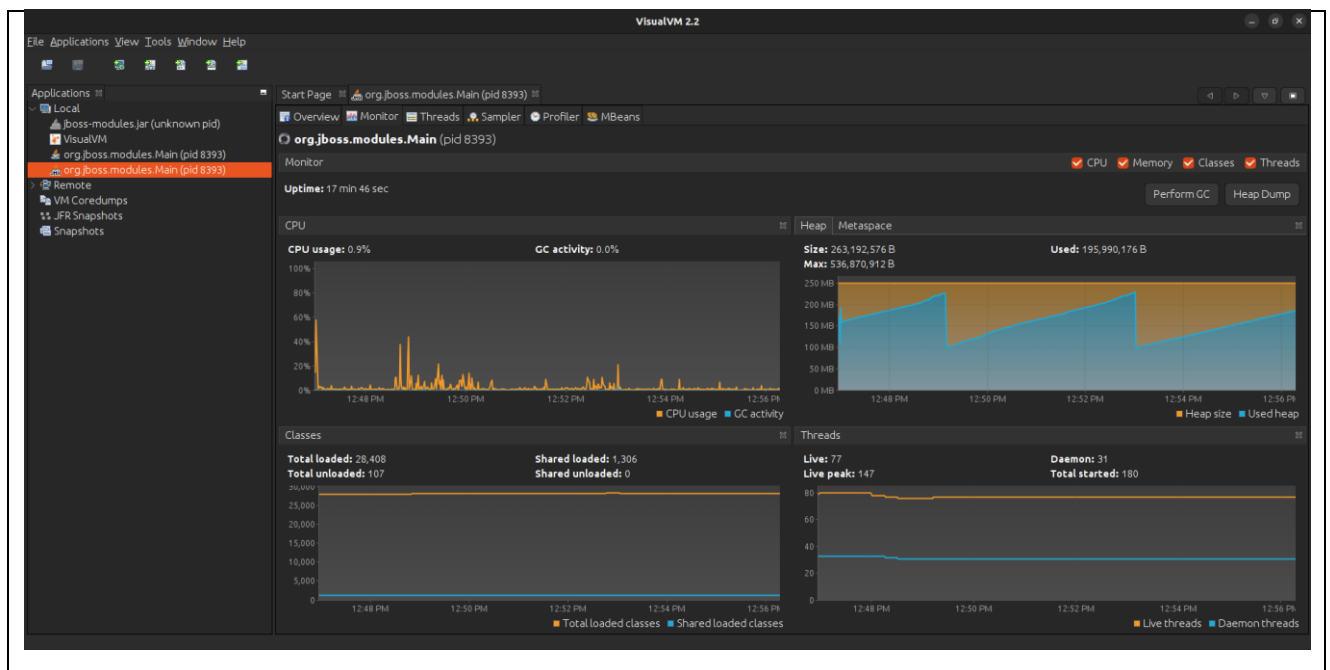
Heap dump on OOME: disabled

Thread Dumps: 0
Heap Dumps: 0
Profiler Snapshots: 0
JFR Snapshots: 0

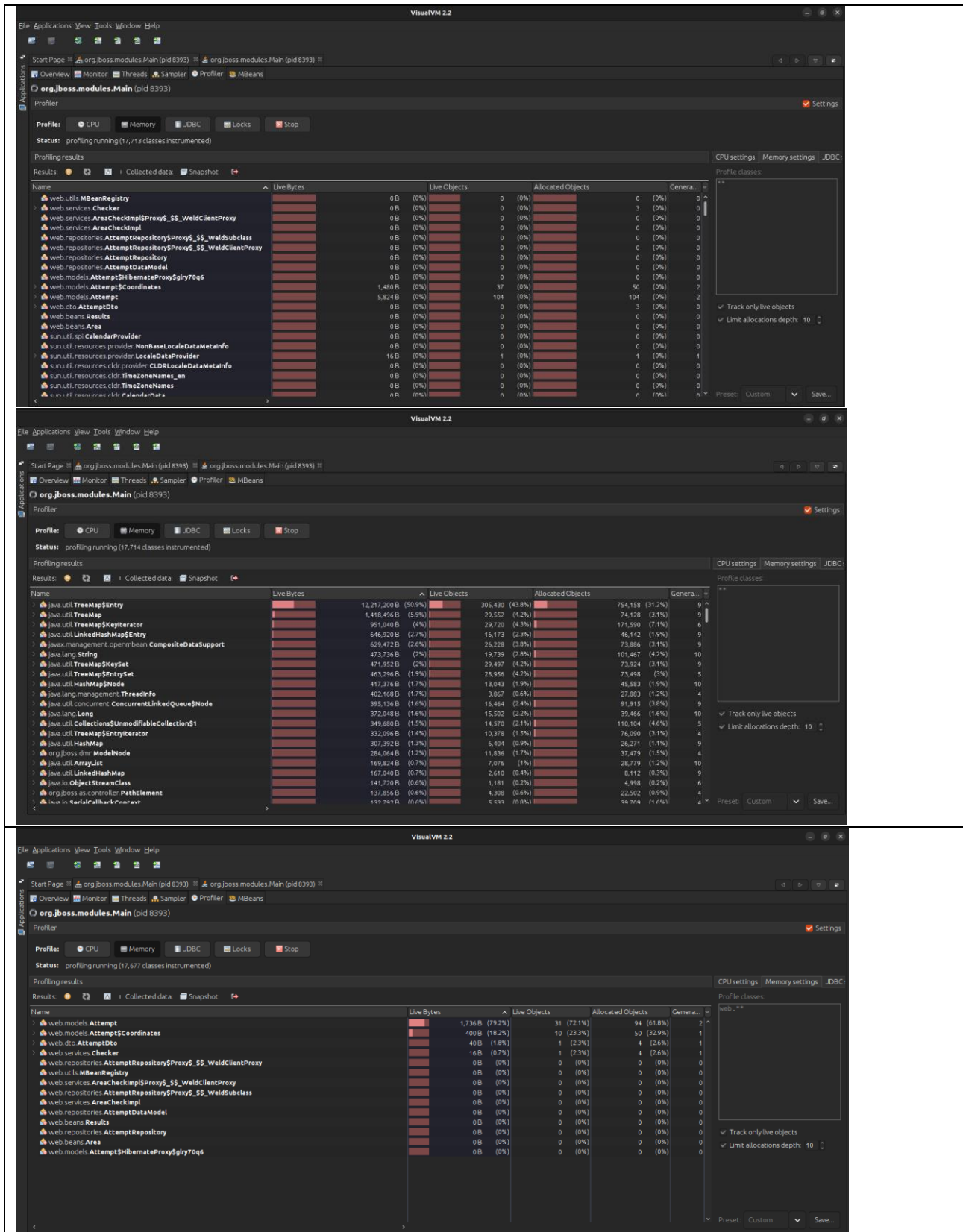
Saved data

JVM arguments	System properties
-D[Standalone]	
-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.desktop/sun.awt=ALL-UNNAMED	
--add-exports=java.naming/com.sun.jndi.ldap=ALL-UNNAMED	
--add-exports=java.naming/com.sun.jndi.ldap=ALL-UNNAMED	
--add-opens=java.base/java.lang=ALL-UNNAMED	
--add-opens=java.base/java.lang.invoke=ALL-UNNAMED	
--add-opens=java.base/java.lang.reflect=ALL-UNNAMED	
--add-opens=java.base/java.io=ALL-UNNAMED	
--add-opens=java.base/java.security=ALL-UNNAMED	
--add-opens=java.base/java.util=ALL-UNNAMED	
--add-opens=java.base/java.util.concurrent=ALL-UNNAMED	
--add-opens=java.management/javax.management=ALL-UNNAMED	
--add-opens=java.naming/javax.naming=ALL-UNNAMED	
-Djava.security.manager=allow	

- **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:

- Results MBean: Графики AttemptStats показывают общее число установленных пользователем точек, а также число точек, не попадающих в область, и также меняются со временем, что отражает взаимодействие пользователя с системой (создание точек на графике).
- Area MBean: График Area показывает динамику изменений области в зависимости от выбора радиус

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 ~59%.
- Больше всего памяти из пользовательских классов занимают объекты Attempt – 79.5% от пользовательских классов. Однако в масштабе всего приложения эти объекты занимают примерно 0.2%.

4. Исследование программы на утечки памяти

Искусственная задержка программы

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(false);
        ServletRunner sr = new ServletRunner();
        sr.registerServlet("myServlet", HelloWorld.class.getName());
        ServletUnitClient sc = sr.newClient();
        int number = 1;
        WebRequest request = new GetMethodWebRequest("http://test.meterware.com/myServlet");
        while (true) {
            WebResponse response = sc.getResponse(request);
            System.out.println("Count: " + number++ + response);
            java.lang.Thread.sleep(200);
        }
    }
}
```

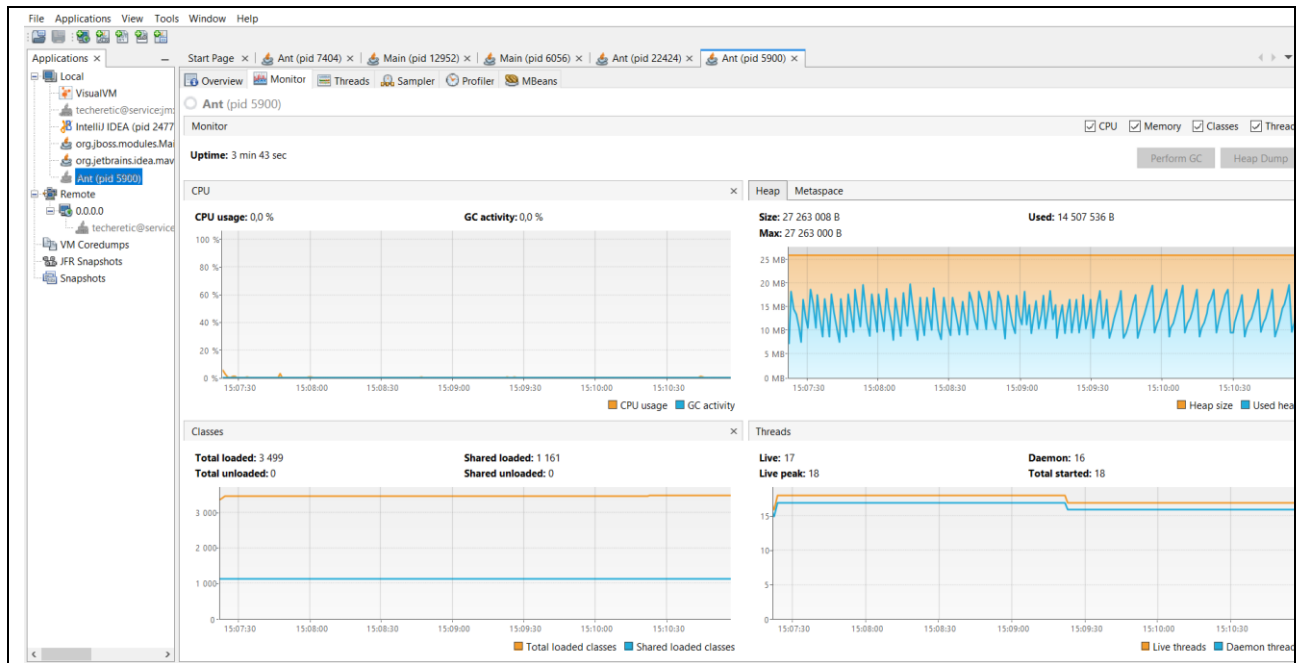
Метод не несёт никакой функциональной нагрузки и является избыточным — поэтому его удаление не повлияет на корректность работы приложения, но повысит производительность за счет устранения задержек.

```
public static void main(String[] args) {
    try {
        HttpUnitOptions.setExceptionsThrownOnScriptError(false);
        ServletRunner sr = new ServletRunner();
        sr.registerServlet("myServlet", HelloWorld.class.getName());
        ServletUnitClient sc = sr.newClient();
        int number = 1;
        WebRequest request = new GetMethodWebRequest("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("global").log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        //
    }
}
```

Утечка памяти

Установим максимальный размер кучи в 25Мб с помощью `-Xmx25m` и запустим программу и посмотрим, как она себя ведет, в VisualVM.

1. При мониторинге видна очевидная переработка GC:



2. Программа падает через ~3-4 минуты, независимо от выделенной памяти для кучи:

```
[java] Count: 66515[ _response = com.meterware.servletunit.ServletUnitHttpResponse@3d430218]
[java] Count: 66516[ _response = com.meterware.servletunit.ServletUnitHttpResponse@722bfee0]
[java] Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
[java]   at java.base/java.util.Arrays.copyOf(Arrays.java:3537)
[java]   at java.base/java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractStringBuilder.java:228)
[java]   at java.base/java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:582)
[java]   at java.base/java.lang.StringBuilder.append(StringBuilder.java:179)
[java]   at java.base/java.lang.StringBuilder.append(StringBuilder.java:173)
[java]   at java.base/java.lang.Throwable.printStackTrace(Throwable.java:673)
```


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

The screenshot displays the IntelliJ IDEA interface with the Heap Dump tool open for the Ant process (pid 18564). The tool provides a comprehensive overview of the JVM's memory state, including heap size, class and instance counts, and a detailed breakdown of objects by size and type.

Summary Statistics:

- Heap Size: 6 997 216 B
- Classes: 3 682
- Instances: 177 111
- Classloaders: 103
- GC Roots: 3 185
- Objects Pending for Finalization: 0

Environment:

- System: Windows 10 (10...)
- Architecture: amd64 64b...
- Java Home: C:\Program Files\Java\jdk-1...
- Java Version: 17.0.8 2023-07-18 L...
- Java Name: Java HotSpot(TM) 64-Bit Server VM (17.0.8+9-LTS-211, mixed mode, sharin...
- Java Vendor: Oracle Corporation
- JVM Uptime: 3 min 20 s

Classes by Number of Instances:

Class	Count	Percentage
byte[]	29 163	16.5 %
java.lang.String	27 844	15.7 %
java.util.TreeMap\$Entry	11 415	6.4 %
java.util.concurrent.ConcurrentHashMap\$Node	8 309	4.7 %
java.util.HashMap\$Node	7 386	4.2 %

Instances by Size:

Class	Count	Percentage
byte[]#2056: 125 176 items	125 192 B	1.8 %
byte[]#281: 65 536 items	65 552 B	0.9 %
char[]#4: <!--*** GENERATED FROM project.xml - DO NOT EDIT ****> EDIT _/build.xml IN	16 400 B	0.2 %
char[]#12: [java] Count: 647381 _response => com.metenware.servletunit.ServletUnitHttpRespc	16 400 B	0.2 %
char[]#5: Buildfile: C:\Users\Admin\IdeaProjects\opli-lab3\httpUnit\build.xml...	16 400 B	0.2 %

Classes by Size of Instances:

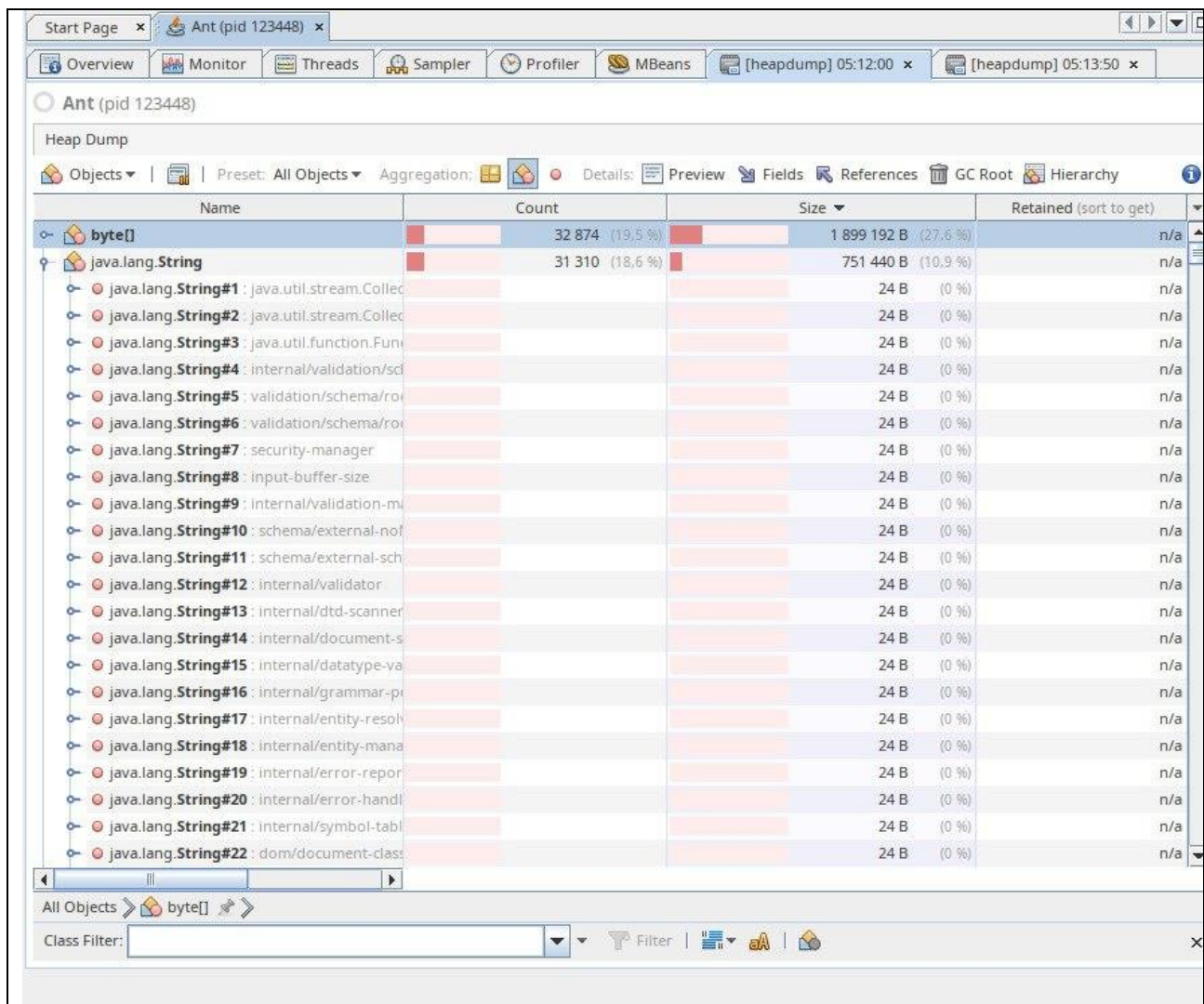
Class	Count	Percentage
byte[]	1 699 064 B	24.3 %
java.lang.String	668 256 B	9.6 %
java.util.TreeMap\$Entry	456 600 B	6.5 %
java.lang.Object[]	350 152 B	5.0 %
java.util.concurrent.ConcurrentHashMap\$Node	265 888 B	3.8 %

Dominators by Retained Size:

Class	Count	Percentage
byte[]	1 699 064 B	24.3 %
java.lang.String	668 256 B	9.6 %
java.util.TreeMap\$Entry	456 600 B	6.5 %
java.lang.Object[]	350 152 B	5.0 %
java.util.concurrent.ConcurrentHashMap\$Node	265 888 B	3.8 %

Detailed Object List:

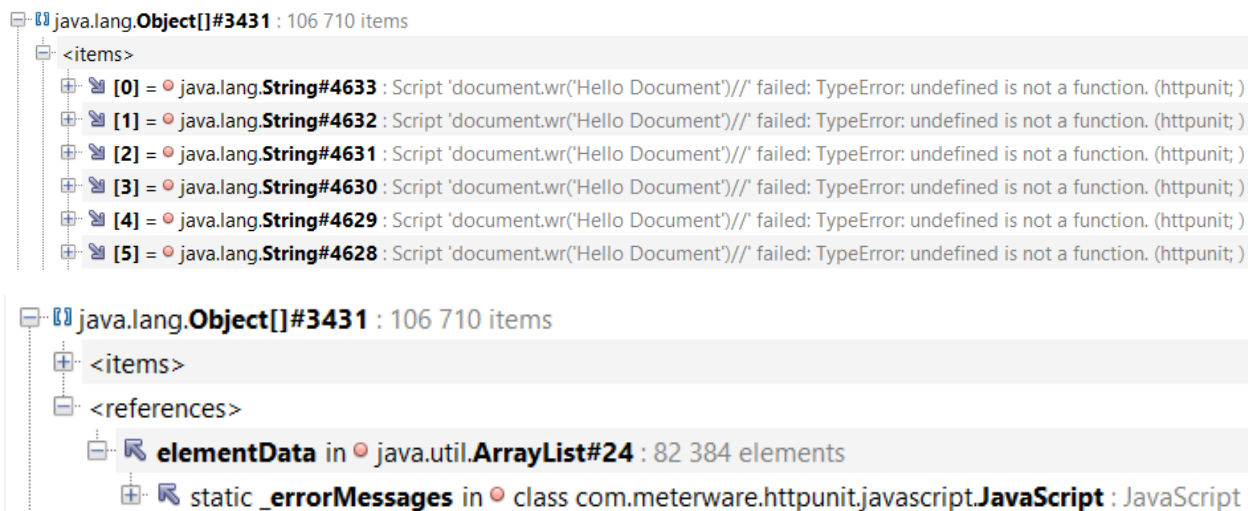
Name	Count	Size	Retained (sort to get)
byte[]	29 163 (16.5 %)	1 699 064 B (24.3 %)	n/a
java.lang.String	27 844 (15.7 %)	668 256 B (9.6 %)	n/a
java.util.TreeMap\$Entry	11 415 (6.4 %)	456 600 B (6.5 %)	n/a
java.util.concurrent.ConcurrentHashMap\$Node	8 309 (4.7 %)	265 888 B (3.8 %)	n/a
java.util.HashMap\$Node	7 386 (4.2 %)	236 352 B (3.4 %)	n/a
java.lang.Object[]	6 874 (3.9 %)	350 152 B (5.0 %)	n/a
java.lang.Long	5 952 (3.4 %)	142 848 B (2.1 %)	n/a
java.util.LinkedHashMap\$Entry	5 337 (3.0 %)	213 480 B (3.1 %)	n/a
java.util.TreeMap	3 577 (2.0 %)	171 696 B (2.5 %)	n/a
java.util.TreeMap\$KeySet	3 545 (2.0 %)	56 720 B (0.8 %)	n/a
javax.management.openmbean.CompositeDataSupport	3 536 (2.0 %)	84 864 B (1.2 %)	n/a
java.util.ArrayList	2 238 (1.3 %)	53 712 B (0.8 %)	n/a
java.util.ImmutableCollections\$List12	2 167 (1.2 %)	52 008 B (0.7 %)	n/a
java.util.HashMap\$Entry	2 131 (1.2 %)	68 192 B (1.0 %)	n/a
java.util.HashMap\$Node[]	2 109 (1.2 %)	221 376 B (3.2 %)	n/a
sun.util.locale.LocaleObjectCache\$CacheEntry	1 985 (1.1 %)	79 400 B (1.1 %)	n/a
java.lang.Object	1 943 (1.1 %)	31 088 B (0.4 %)	n/a
java.util.Collections\$UnmodifiableRandomAccessList	1 923 (1.1 %)	46 152 B (0.7 %)	n/a
java.lang.reflect.Method	1 913 (1.1 %)	168 344 B (2.4 %)	n/a
java.lang.ref.SoftReference	1 857 (1.0 %)	74 280 B (1.1 %)	n/a
java.lang.String[]	1 784 (1.0 %)	61 520 B (0.9 %)	n/a
java.lang.Class[]	1 762 (1.0 %)	41 848 B (0.6 %)	n/a
java.util.Arrays\$ArrayList	1 714 (1.0 %)	41 136 B (0.6 %)	n/a
java.util.LinkedHashMap	1 435 (0.8 %)	80 360 B (1.1 %)	n/a
jdk.jfr.AnnotationElement	1 433 (0.8 %)	34 392 B (0.5 %)	n/a



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 отработывает 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 объект, занимающий много памяти (свыше 4мб), при этом содержащий повторяющиеся строки:



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 этого массива, однако можно заметить, что на самом деле она не используется нигде

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

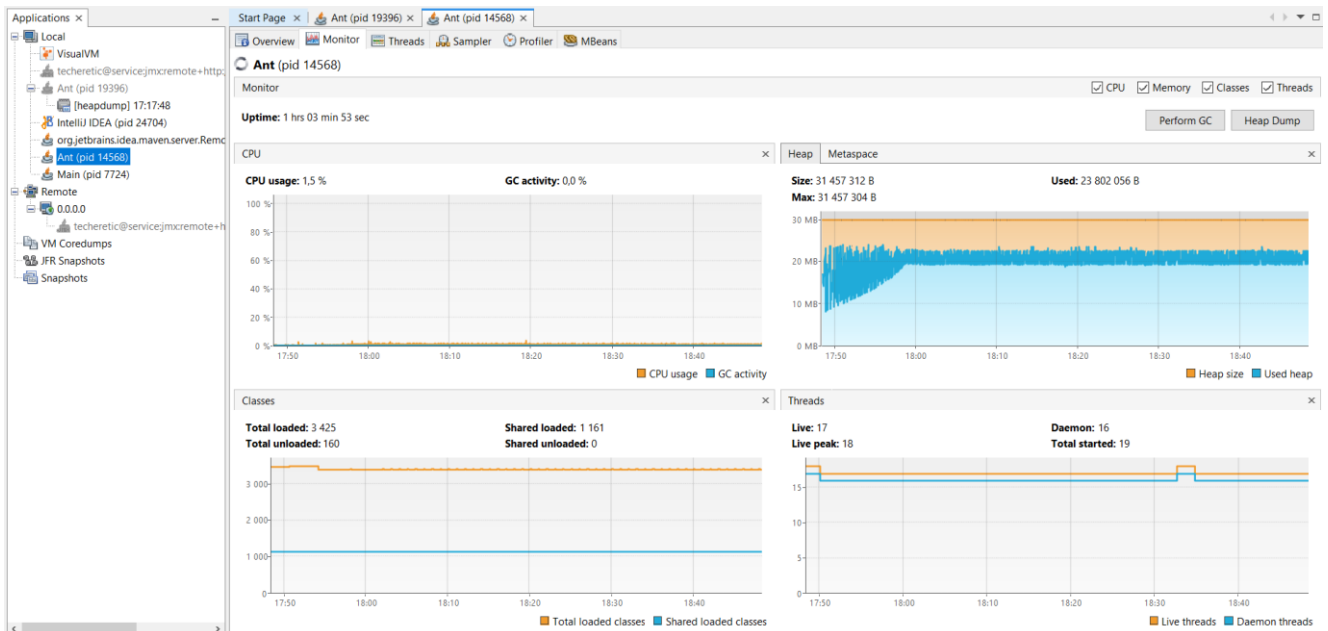
```
public void clearErrorMessages() {  
    JavaScript.clearErrorMessages();  
}
```

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

Решением будет очистка списка `_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 стали более стабильными, и ее значение не стремится к максимальному, так что можно сделать вывод, что GC работает более оптимально.



Программа работает стабильно и не падает с `OutOfMemoryException`.

```
[java] Count: 1773673[ _response = com.meterware.servletunit.ServletUnitHttpResponse@68f48597]
[java] Errors count1
[java] Count: 1773674[ _response = com.meterware.servletunit.ServletUnitHttpResponse@31ebbb3]
[java] Errors count1
[java] Count: 1773675[ _response = com.meterware.servletunit.ServletUnitHttpResponse@1834269b]
[java] Errors count1
[java] Count: 1773676[ _response = com.meterware.servletunit.ServletUnitHttpResponse@309c898d]
[java] Errors count1
[java] Count: 1773677[ _response = com.meterware.servletunit.ServletUnitHttpResponse@c2e4516]
[java] Errors count1
[java] Count: 1773678[ _response = com.meterware.servletunit.ServletUnitHttpResponse@5c4a9717]
[java] Errors count1
[java] Count: 1773679[ _response = com.meterware.servletunit.ServletUnitHttpResponse@c89bcd6]
[java] Errors count1
[java] Count: 1773680[ _response = com.meterware.servletunit.ServletUnitHttpResponse@4702f48a]
[java] Errors count1
[java] Count: 1773681[ _response = com.meterware.servletunit.ServletUnitHttpResponse@795bb855]
```

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.