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

Факультет программной инженерии и компьютерной техники

Лабораторная работа №4
Вариант №645

Выполнил
Путинцев Данил Денисович
Группа P3207
Проверил(а)
Преподаватель: Миху Вадим Дмитриевич

Санкт-Петербург 2025 год

Текст задания

Вариант 645

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, попадающих в область. В случае, если количество установленных пользователем точек стало кратно 15, 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 наименование и версию JVM, поставщика виртуальной машины Java и номер её сборки.

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 потока, потребляющего наибольший процент времени CPU.

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.

Исходный код разработанных MBean-классов и сопутствующих классов.

```java
package org.example;

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

import java.io.Serializable;
import java.time.Duration;
import java.time.Instant;

@Named("Interval")
@ApplicationScoped
public class Interval implements Serializable, IntervalMBean {
    private String averageInterval;
    private Instant lastClickTime;
    private Long duration = 0L;
    private Long totalIntervalMillis = 0L;

    public Interval(){

    }

    @Inject
    public Interval(RegMBeans reg){
```

```
        this.reg = reg;
        reg.registerBean(this);
    }
    RegMBeans reg;

    @Inject
    Count count;

    @Override
    public String getAverageInterval() {
        return String.format("%.2f сек", calcAverageInterval() / 1000);
    }

    public void averageInterval(String averageInterval) {
        this.averageInterval = averageInterval;
    }

    public Instant getLastClickTime() {
        return lastClickTime;
    }
    public void setLastClickTime(Instant lastClickTime) {
        this.lastClickTime = lastClickTime;
    }

    @Override
    public void registerClick() {
        Instant now = Instant.now();
        if (lastClickTime != null) {
            long interval = Duration.between(lastClickTime, now).toMillis();
            totalIntervalMillis += interval;
        }
        lastClickTime = now;
    }

    @Override
    public double calcAverageInterval() {
        if (count.getAllPoints() < 2) return 0;
        return (double) totalIntervalMillis / (count.getAllPoints() - 1);
    }
}
```

```
package org.example;
```

```java
public interface IntervalMBean {
    void registerClick();
    double calcAverageInterval();
    String getAverageInterval();
}
```

```java
package org.example;


import jakarta.enterprise.context.SessionScoped;
import jakarta.inject.Inject;
import jakarta.inject.Named;

import java.io.Serializable;

@Named("Count")
@SessionScoped
public class Count implements Serializable, CountMBean{
    private Integer allPoints = 0;
    private Integer insidePoints = 0;

    public Count(){

    }
    @Inject
    public Count(RegMBeans reg){
        this.reg = reg;
        reg.registerBean(this);
    }
    RegMBeans reg;

    @Override
    public Integer getAllPoints() {
        return allPoints;
    }

    @Override
    public void setAllPoints(Integer allPoints) {
        this.allPoints = allPoints;
    }

    @Override
    public Integer getInsidePoints() {
```

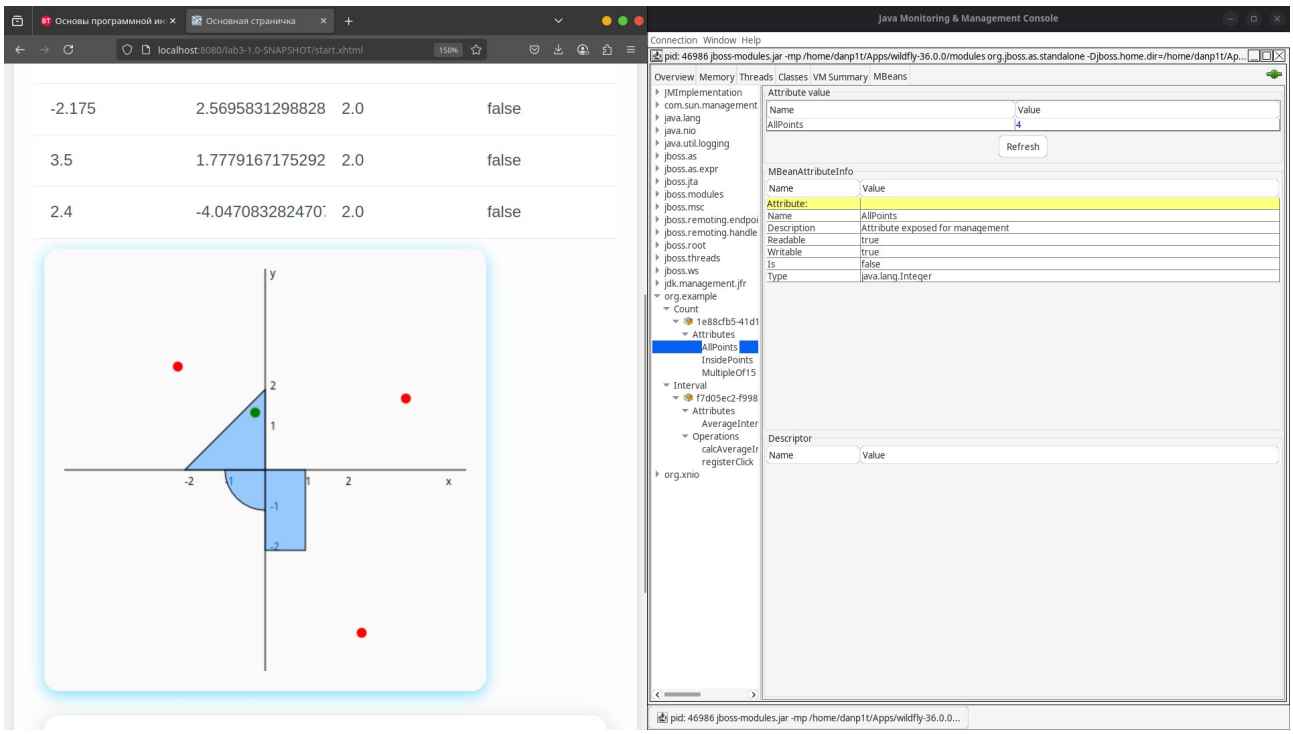
```
        return insidePoints;
    }

    @Override
    public void setInsidePoints(Integer insidePoints) {
        this.insidePoints = insidePoints;
    }

    @Override
    public boolean isMultipleOf15() {
        return allPoints % 15 == 0 && allPoints != 0;
    }


}
```

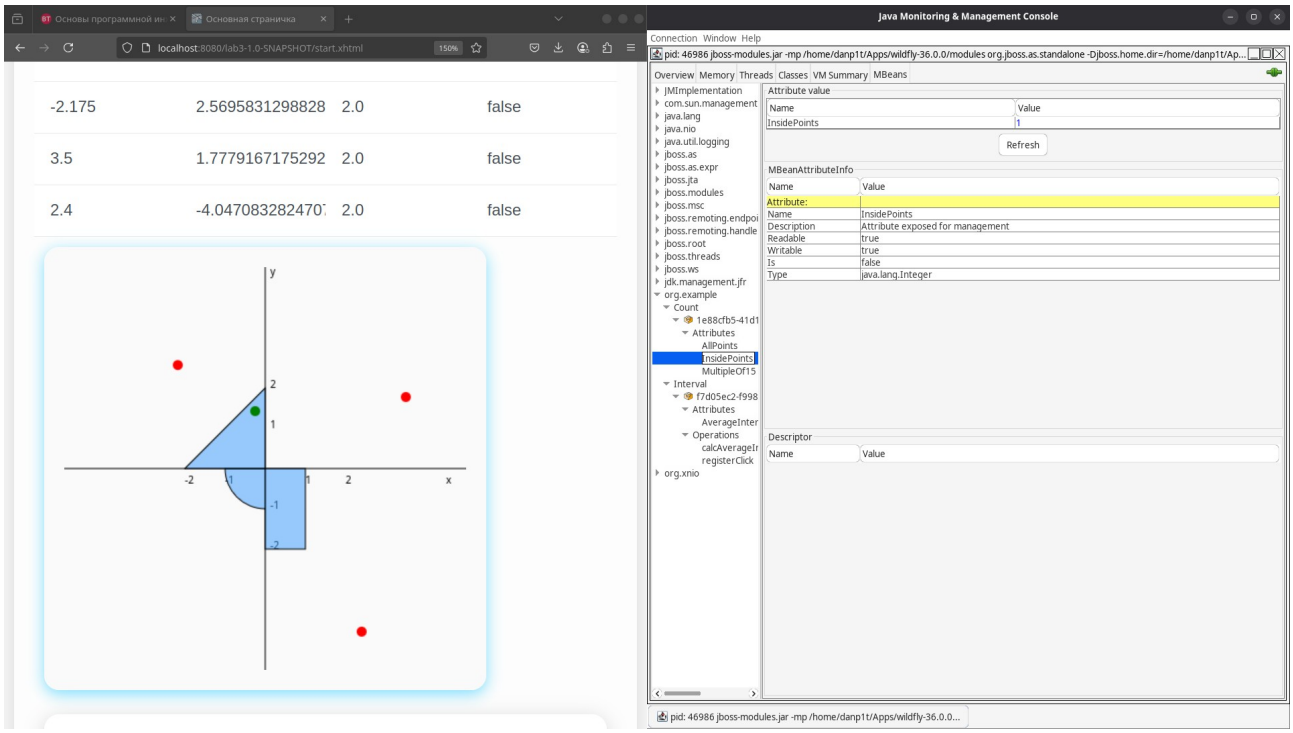
```
package org.example;

public interface CountMBean {
    Integer getAllPoints();
    void setAllPoints(Integer allPoints);
    Integer getInsidePoints();
    void setInsidePoints(Integer insidePoints);
    boolean isMultipleOf15();
}
```

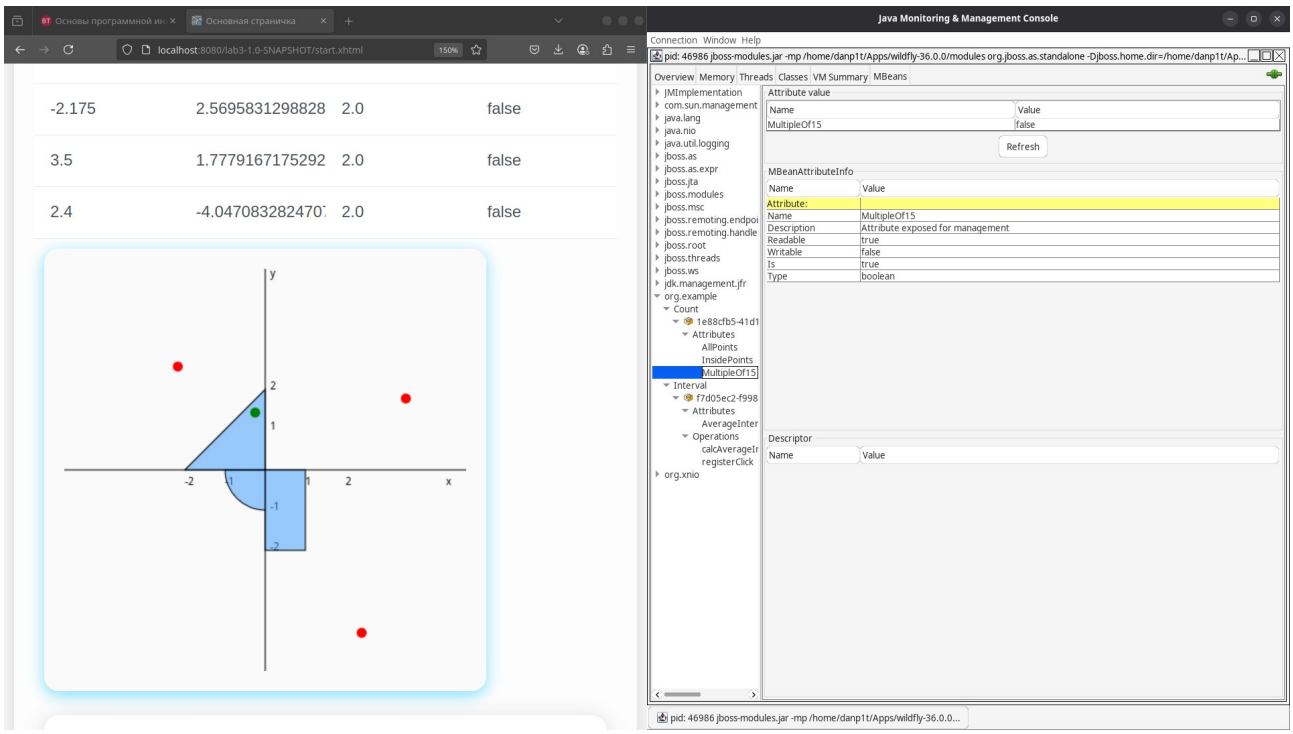
```
package org.example;

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

import javax.management.*;
import java.lang.management.ManagementFactory;
import java.util.HashMap;
import java.util.UUID;
@Named
@ApplicationScoped
public class RegMBeans implements ServletContextListener {
    private final HashMap<Object, ObjectName> bean_names = new HashMap<>();
    public void registerBean(Object bean) {
        try {
```

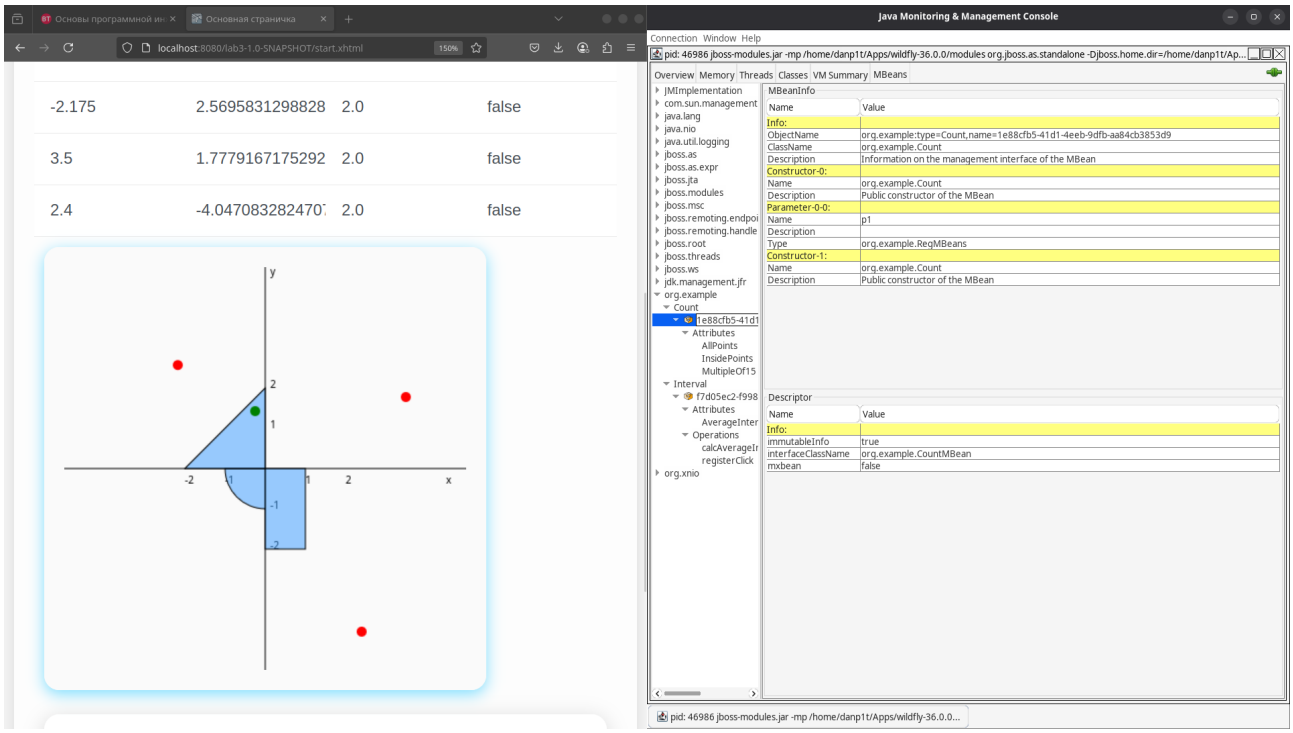
```
            String cur_name = UUID.randomUUID().toString();
            var domain = bean.getClass().getPackageName();
            var type = bean.getClass().getSimpleName();
            var objectName = new ObjectName(String.format("%s:type=%s,name=%s",
domain, type, cur_name));
            bean_names.put(bean, objectName);
            ManagementFactory.getPlatformMBeanServer().registerMBean(bean,
objectName);
        } catch (InstanceAlreadyExistsException | MBeanRegistrationException |
NotCompliantMBeanException | MalformedObjectNameException e) {
            e.printStackTrace();
        }
    }

    public void unregisterBean(Object bean) {
        try {

ManagementFactory.getPlatformMBeanServer().unregisterMBean(bean_names.get(be
an));
        } catch (InstanceNotFoundException | MBeanRegistrationException e) {
            e.printStackTrace();
        }
    }
}
```

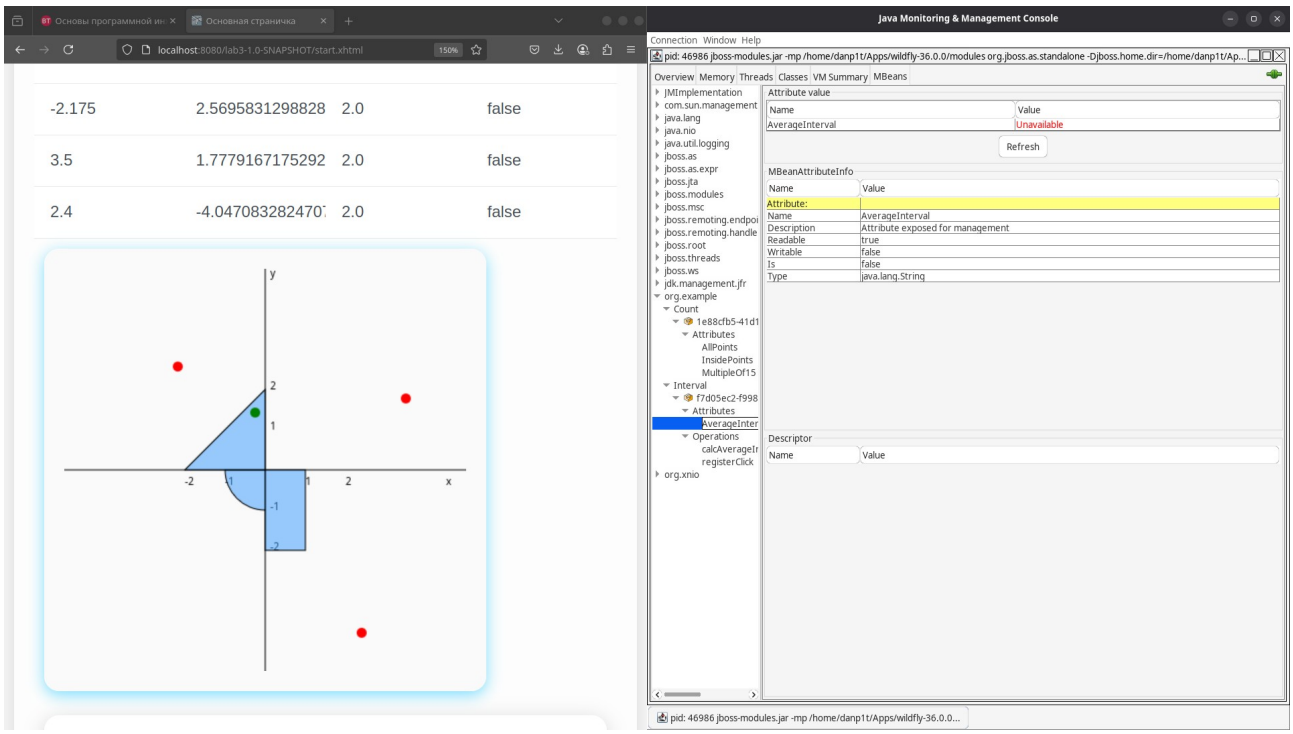
Скриншоты программы JConcole со снятыми показаниями, выводы по результатам мониторинга.

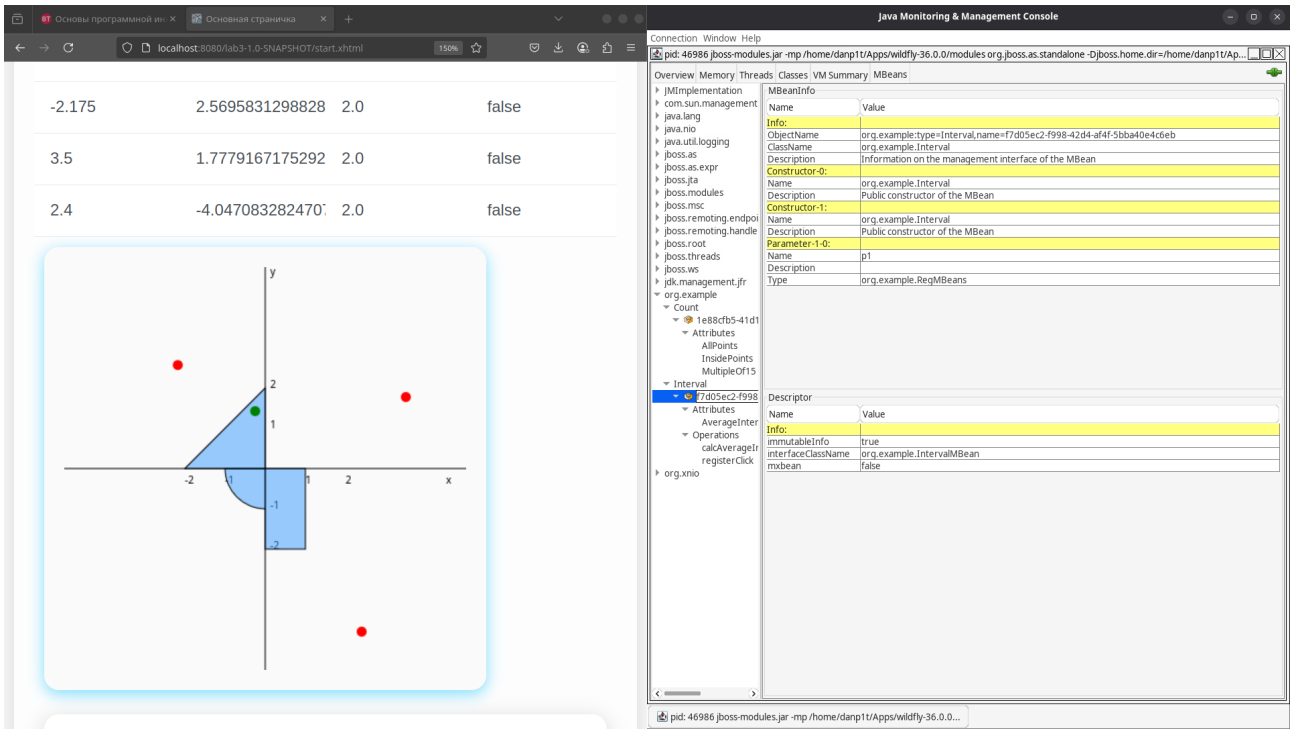

Основы программной ин
Основная страничка
localhost:8080/lab3-1.0-SNAPSHOT/start.xhtml
150%
-2.175 2.5695831298828 2.0 false
3.5 1.7779167175292 2.0 false
2.4 -4.047083282470 2.0 false
y
x
Java Monitoring & Management Console
Connection Window Help
pid: 46986 jboss-modules.jar -mp /home/danp1t/Apps/wildfly-36.0.0/modules org.jboss.as.standalone -Djboss.home.dir=/home/danp1t/Ap...
Overview Memory Threads Classes VM Summary MBeans
Attribute value
Name Value
AllPoints 4
Refresh
MBeanAttributeInfo
Attribute:
Name AllPoints
Description Attribute exposed for management
Readable true
Writable true
Is false
Type java.lang.Integer
Descriptor
pid: 46986 jboss-modules.jar -mp /home/danp1t/Apps/wildfly-36.0.0...

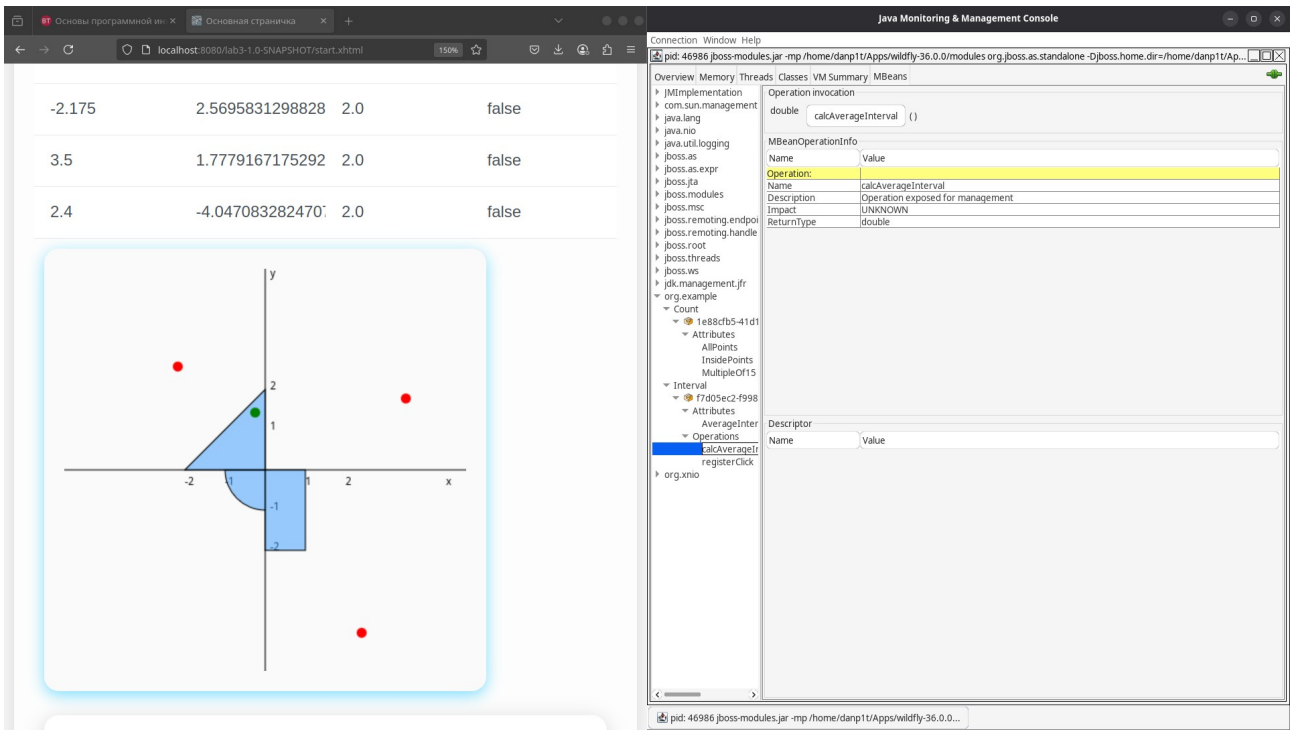

Основы программной ин
Основная страничка
localhost:8080/lab3-1.0-SNAPSHOT/start.xhtml
150%
-2.175 2.5695831298828 2.0 false
3.5 1.7779167175292 2.0 false
2.4 -4.047083282470 2.0 false
y
x
Java Monitoring & Management Console
Connection Window Help
pid: 46986 jboss-modules.jar -mp /home/danp1t/Apps/wildfly-36.0.0/modules org.jboss.as.standalone -Djboss.home.dir=/home/danp1t/Ap...
Overview Memory Threads Classes VM Summary MBeans
Attribute value
Name Value
InsidePoints 1
Refresh
MBeanAttributeInfo
Attribute:
Name InsidePoints
Description Attribute exposed for management
Readable true
Writable true
Is false
Type java.lang.Integer
Descriptor
pid: 46986 jboss-modules.jar -mp /home/danp1t/Apps/wildfly-36.0.0...

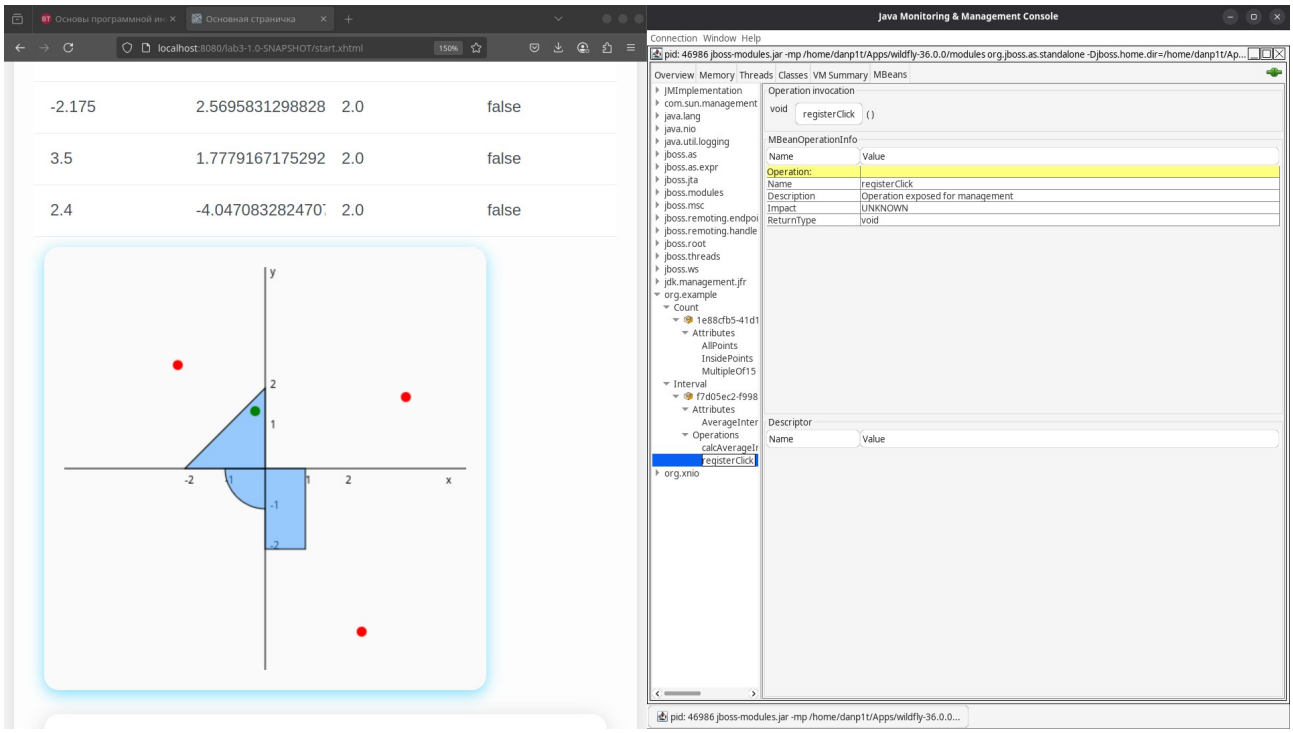

Основы программной ин
Основная страничка
localhost:8080/lab3-1.0-SNAPSHOT/start.xhtml
150%
-2.175 2.5695831298828 2.0 false
3.5 1.7779167175292 2.0 false
2.4 -4.047083282470 2.0 false
y
x
Java Monitoring & Management Console
Connection Window Help
pid: 46986 jboss-modules.jar -mp /home/danp1t/Apps/wildfly-36.0.0/modules org.jboss.as.standalone -Djboss.home.dir=/home/danp1t/Ap...
Overview Memory Threads Classes VM Summary MBeans
Attribute value
Name Value
MultipleOf15 false
Refresh
MBeanAttributeInfo
Name Value
Attribute:
Name MultipleOf15
Description Attribute exposed for management
Readable true
Writable false
Is true
Type boolean
Descriptor
Name Value
pid: 46986 jboss-modules.jar -mp /home/danp1t/Apps/wildfly-36.0.0...

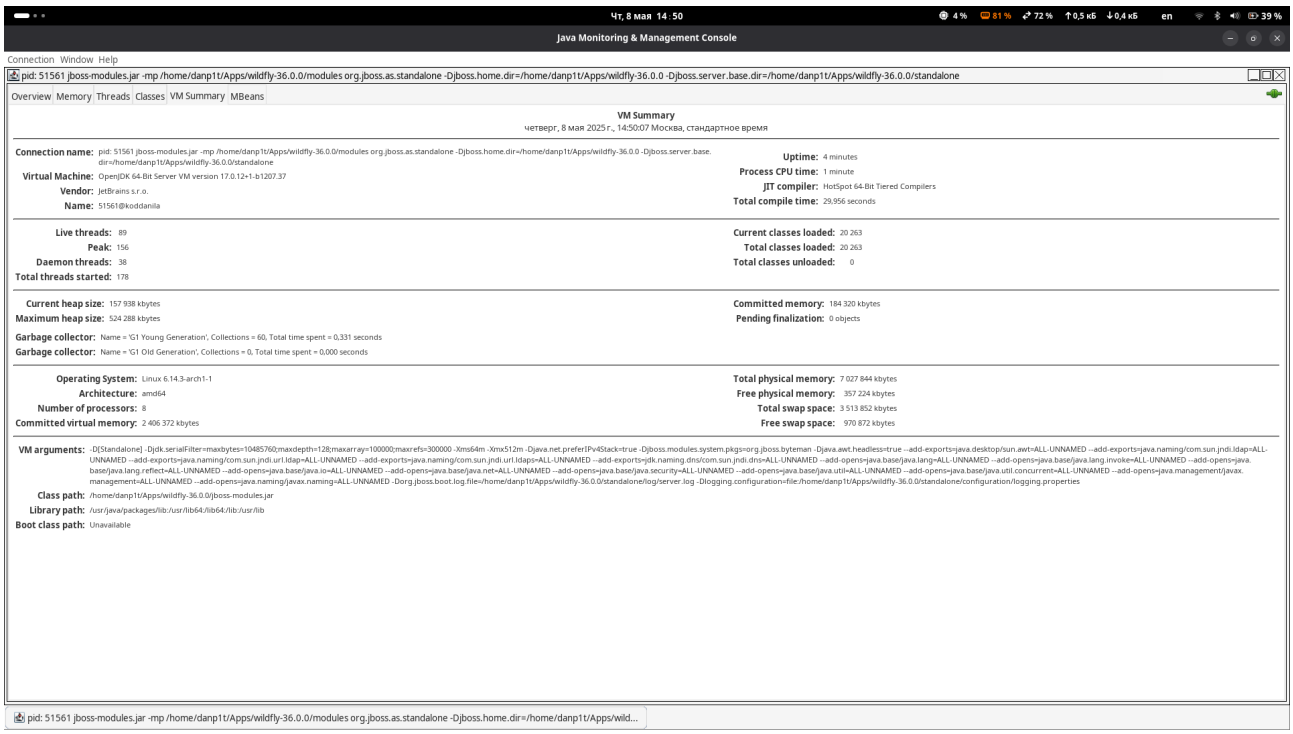

Основы программной ин
Основная страничка
localhost:8080/lab3-1.0-SNAPSHOT/start.xhtml
150%
-2.175 2.5695831298828 2.0 false
3.5 1.7779167175292 2.0 false
2.4 -4.047083282470 2.0 false
y
x
Java Monitoring & Management Console
Connection Window Help
pid: 46986 jboss-modules.jar -mp /home/danp1t/Apps/wildfly-36.0.0/modules org.jboss.as.standalone -Djboss.home.dir=/home/danp1t/Ap...
Overview Memory Threads Classes VM Summary MBeans
MBeanInfo
Name Value
Info:
ObjectName org.example:type=Count,name=1e88cfb5-41d1-4eeb-9dfb-aa84cb3853d9
ClassName org.example.Count
Description Information on the management interface of the MBean
Constructor-0:
Name org.example.Count
Description Public constructor of the MBean
Parameter-0-0:
Name p1
Description
Type org.example.ReqMBeans
Constructor-1:
Name org.example.Count
Description Public constructor of the MBean
Descriptor
Name Value
Info:
immutableInfo true
interfaceClassName org.example.CountMBean
mxbean false
pid: 46986 jboss-modules.jar -mp /home/danp1t/Apps/wildfly-36.0.0...


Основы программной ин
Основная страничка
localhost:8080/lab3-1.0-SNAPSHOT/start.xhtml
150%
-2.175 2.5695831298828 2.0 false
3.5 1.7779167175292 2.0 false
2.4 -4.047083282470 2.0 false
y
x
2
1
-2
-1
1
2
-1
-2
Java Monitoring & Management Console
Connection Window Help
pid: 46986 jboss-modules.jar -mp /home/danp1t/Apps/wildfly-36.0.0/modules org.jboss.as.standalone -Djboss.home.dir=/home/danp1t/Ap...
Overview Memory Threads Classes VM Summary MBeans
JMImplementation
com.sun.management
java.lang
java.nio
java.util.logging
jboss.as
jboss.as.expr
jboss.jta
jboss.modules
jboss.msc
jboss.remoting.endpoi
jboss.remoting.handle
jboss.root
jboss.threads
jboss.ws
jdk.management.jfr
org.example
Count
1e88cfb5-41d1
Attributes
AllPoints
InsidePoints
MultipleOf15
Interval
f7d05ec2-f998
Attributes
AverageInter
Operations
calcAverageI
registerClick
org.xnio
Attribute value
Name
Value
AverageInterval
Unavailable
Refresh
MBeanAttributeInfo
Name
Value
Attribute:
Name AverageInterval
Description Attribute exposed for management
Readable true
Writable false
Is false
Type java.lang.String
Descriptor
Name
Value
pid: 46986 jboss-modules.jar -mp /home/danp1t/Apps/wildfly-36.0.0...


Основы программной ин
Основная страничка
localhost:8080/lab3-1.0-SNAPSHOT/start.xhtml
150%
-2.175 2.5695831298828 2.0 false
3.5 1.7779167175292 2.0 false
2.4 -4.047083282470 2.0 false
y
x
2
1
-2
-1
1
2
-1
-2
Java Monitoring & Management Console
Connection Window Help
pid: 46986 jboss-modules.jar -mp /home/danp1t/Apps/wildfly-36.0.0/modules org.jboss.as.standalone -Djboss.home.dir=/home/danp1t/Ap...
Overview Memory Threads Classes VM Summary MBeans
JMImplementation
com.sun.management
java.lang
java.nio
java.util.logging
jboss.as
jboss.as.expr
jboss.jta
jboss.modules
jboss.msc
jboss.remoting.endpoi
jboss.remoting.handle
jboss.root
jboss.threads
jboss.ws
jdk.management.jfr
org.example
Count
1e88cfb5-41d1
Attributes
AllPoints
InsidePoints
MultipleOf15
Interval
f7d05ec2-f998
Attributes
AverageInter
Operations
calcAverageI
registerClick
org.xnio
MBeanInfo
Name
Value
Info:
ObjectName org.example:type=Interval,name=f7d05ec2-f998-42d4-af4f-5bba40e4c6eb
ClassName org.example.Interval
Description Information on the management interface of the MBean
Constructor-0:
Name org.example.Interval
Description Public constructor of the MBean
Constructor-1:
Name org.example.Interval
Description Public constructor of the MBean
Parameter-1-0:
Name p1
Description
Type org.example.ReqMBeans
Descriptor
Name
Value
Info:
immutableInfo true
interfaceClassName org.example.IntervalMBean
mxbean false
pid: 46986 jboss-modules.jar -mp /home/danp1t/Apps/wildfly-36.0.0...


Основы программной ин
Основная страничка
localhost:8080/lab3-1.0-SNAPSHOT/start.xhtml
150%
-2.175 2.5695831298828 2.0 false
3.5 1.7779167175292 2.0 false
2.4 -4.047083282470 2.0 false
y
x
Java Monitoring & Management Console
Connection Window Help
Overview Memory Threads Classes VM Summary MBeans
Operation invocation
double calcAverageInterval ( )
MBeanOperationInfo
Name Value
Operation:
Name calcAverageInterval
Description Operation exposed for management
Impact UNKNOWN
ReturnType double
Descriptor
Name Value
org.example
Count
Attributes
AllPoints
InsidePoints
MultipleOf15
Interval
Operations
registerClick
org.xnio


Основы программной ин
Основная страничка
localhost:8080/lab3-1.0-SNAPSHOT/start.xhtml
150%
-2.175 2.5695831298828 2.0 false
3.5 1.7779167175292 2.0 false
2.4 -4.047083282470 2.0 false
y
x
Java Monitoring & Management Console
Connection Window Help
Overview Memory Threads Classes VM Summary MBeans
Operation invocation
void registerClick ( )
MBeanOperationInfo
Name Value
Operation:
Name registerClick
Description Operation exposed for management
Impact UNKNOWN
ReturnType void
Descriptor
Name Value
org.example
Count
Attributes
AllPoints
InsidePoints
MultipleOf15
Interval
Operations
registerClick
org.xnio



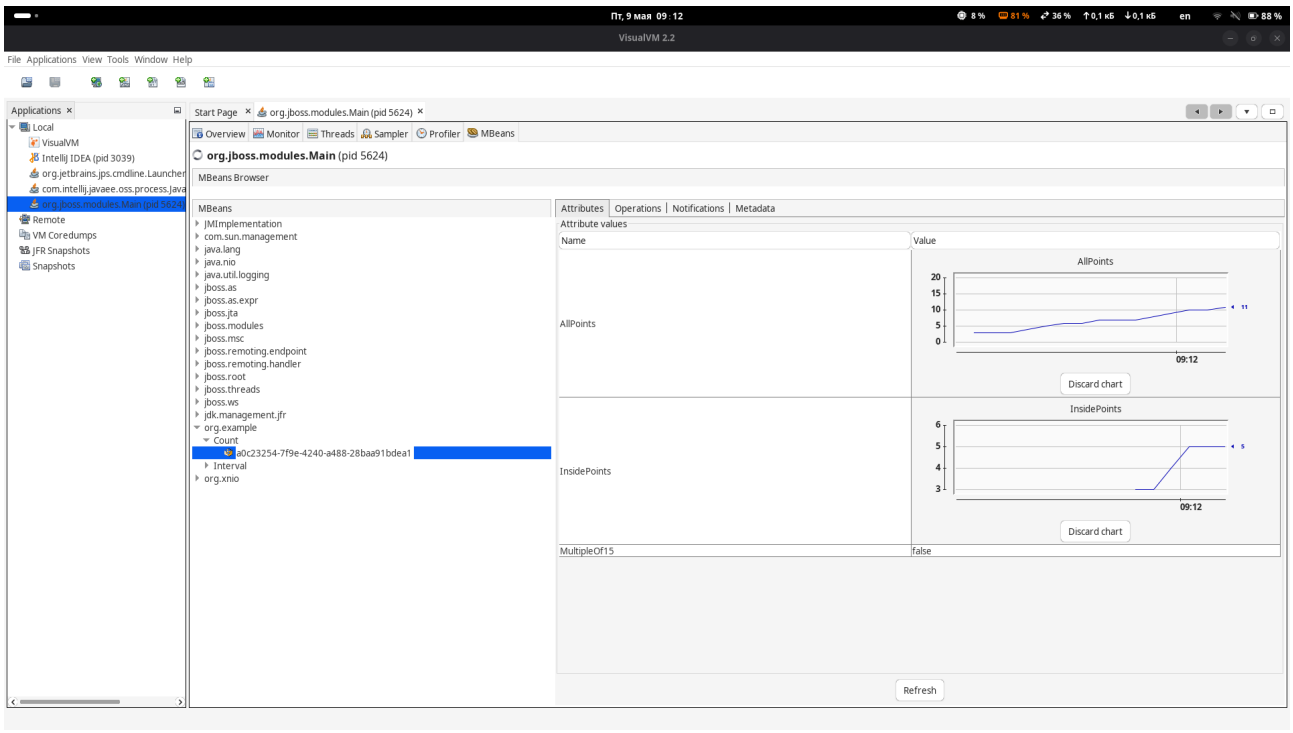
Благодаря утилите Jconsole мы можем увидеть информацию о Mbean, в частности, название класса, конструкторы и параметры конструкторов, также можем посмотреть значение полей Mbean, можем их изменять из утилиты. Также можем увидеть наименование и версию JVM

У меня наименование и версия JVM: OpenJDK-64-Bit Server VM version 17.0.12+1-b1207.37

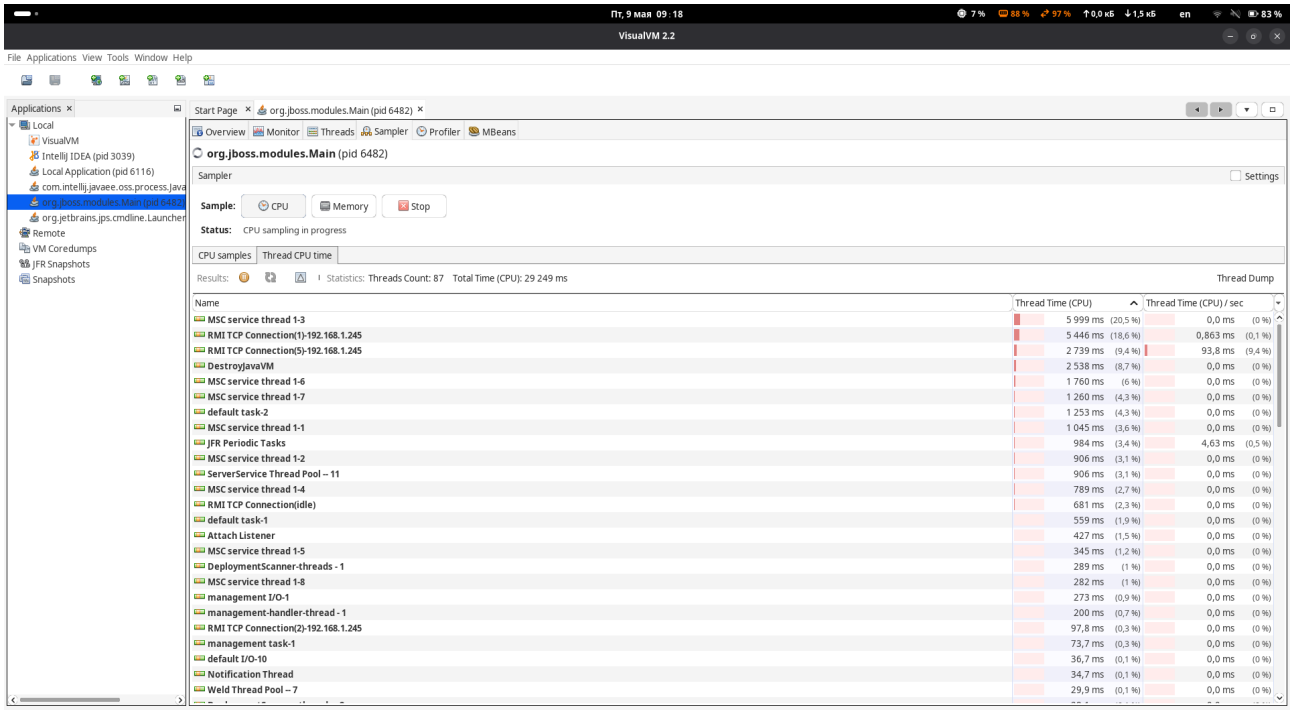
Поставщик виртутальной машины Java: JetBrains s.r.o

Скриншоты программы VisualVM со снятыми показаниями, выводы по результатам профилирования.

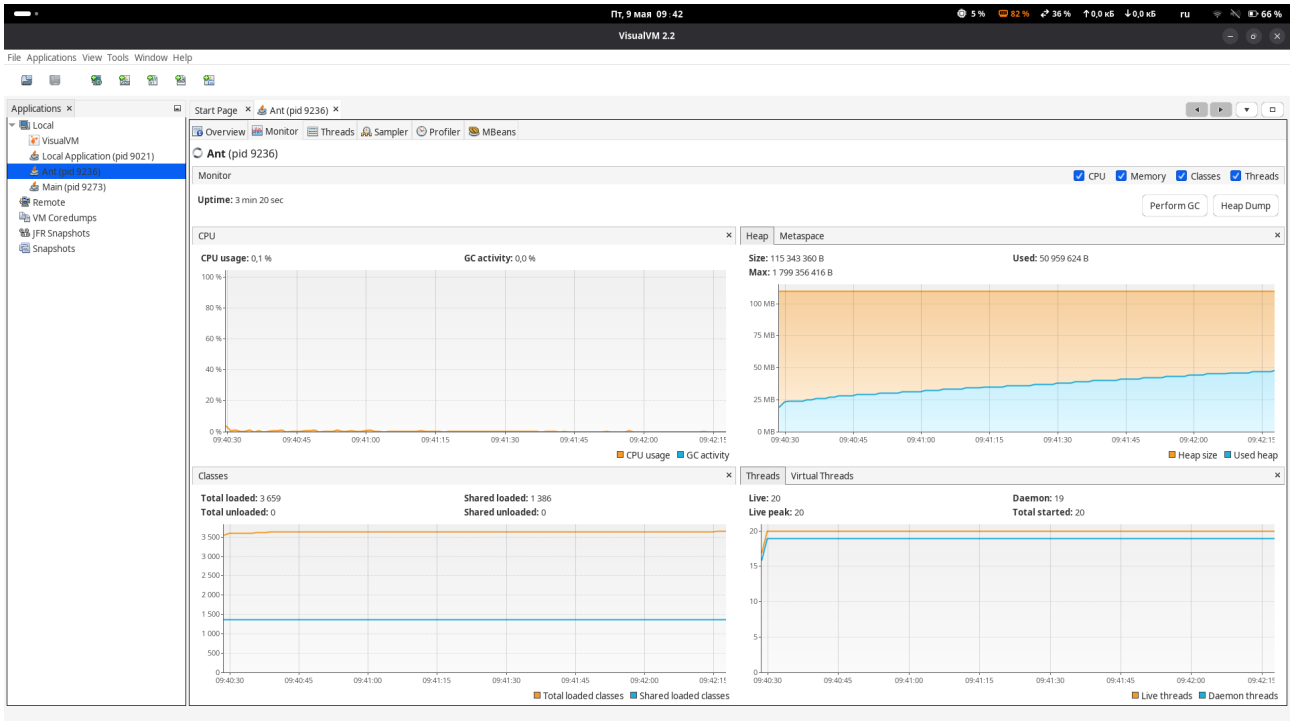
Снять график изменения показаний MBean-классов, разработанных в ходе выполнения задания 1, с течением времени:

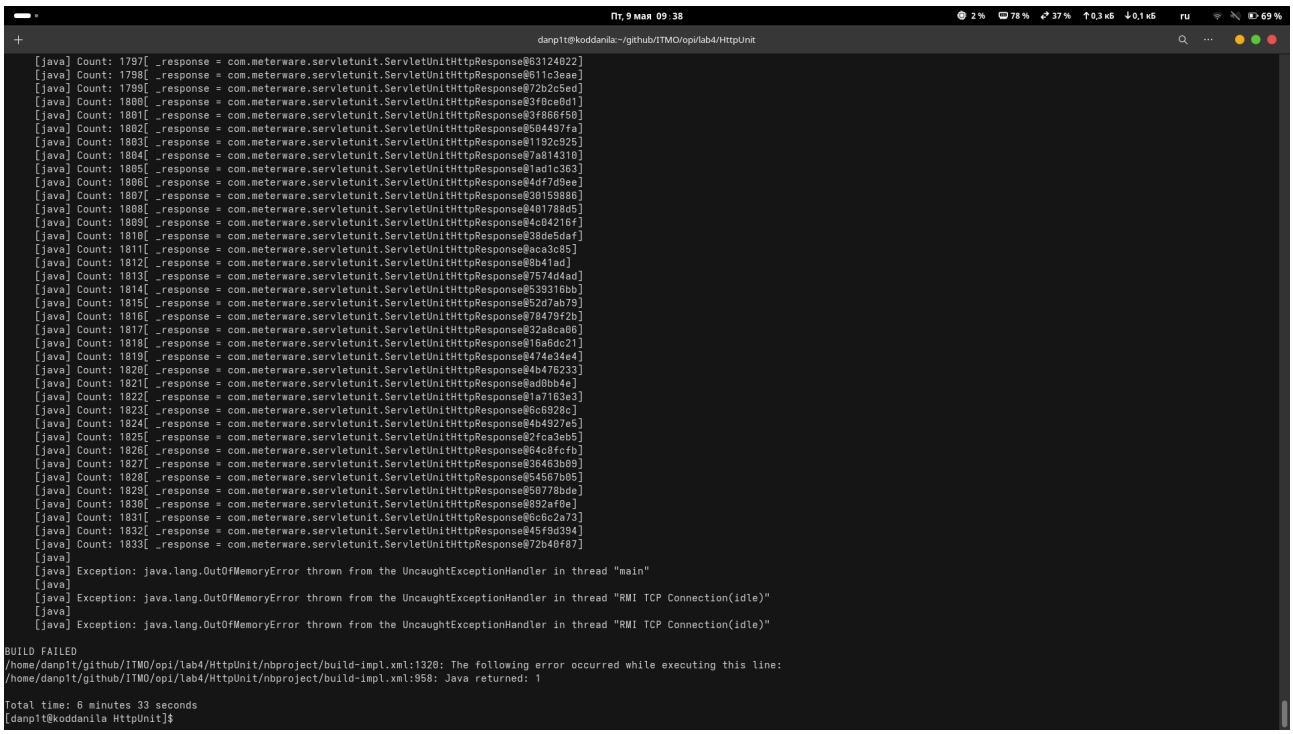


Определить имя потока, потребляющего наибольший процент времени CPU:

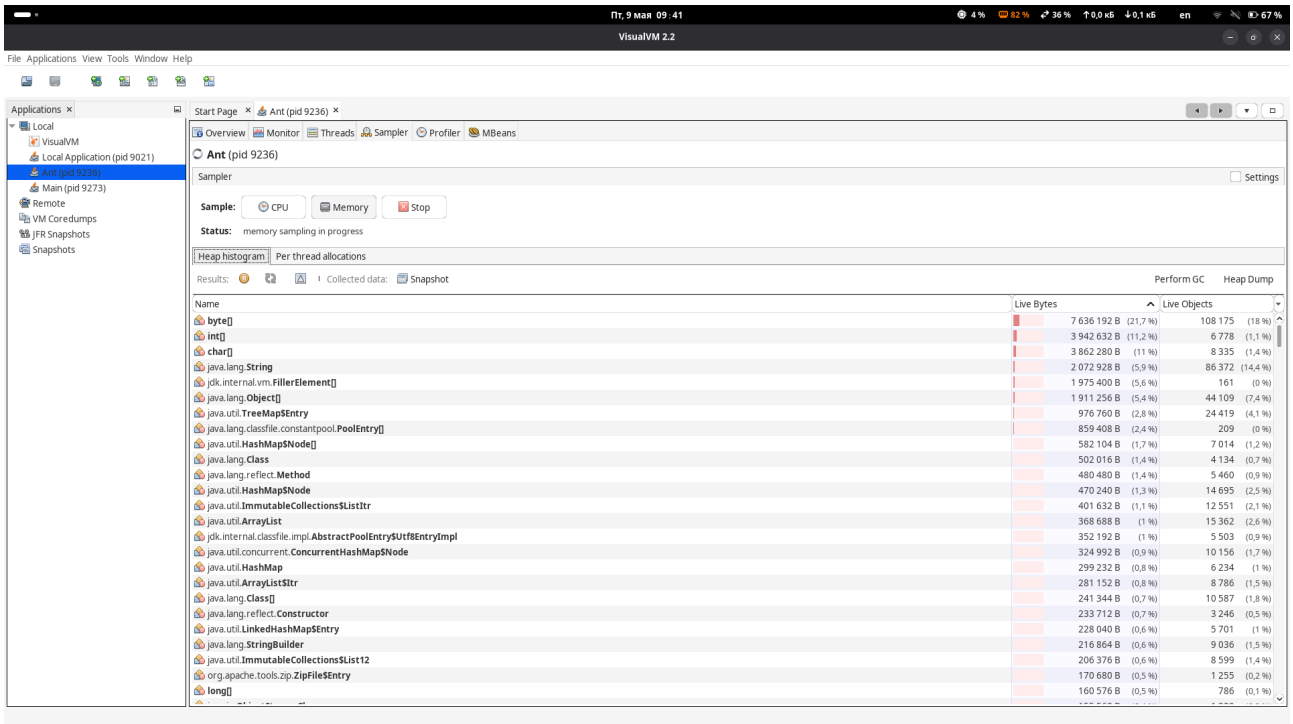


Скриншоты программы VisualVM с комментариями по ходу поиска утечки памяти.

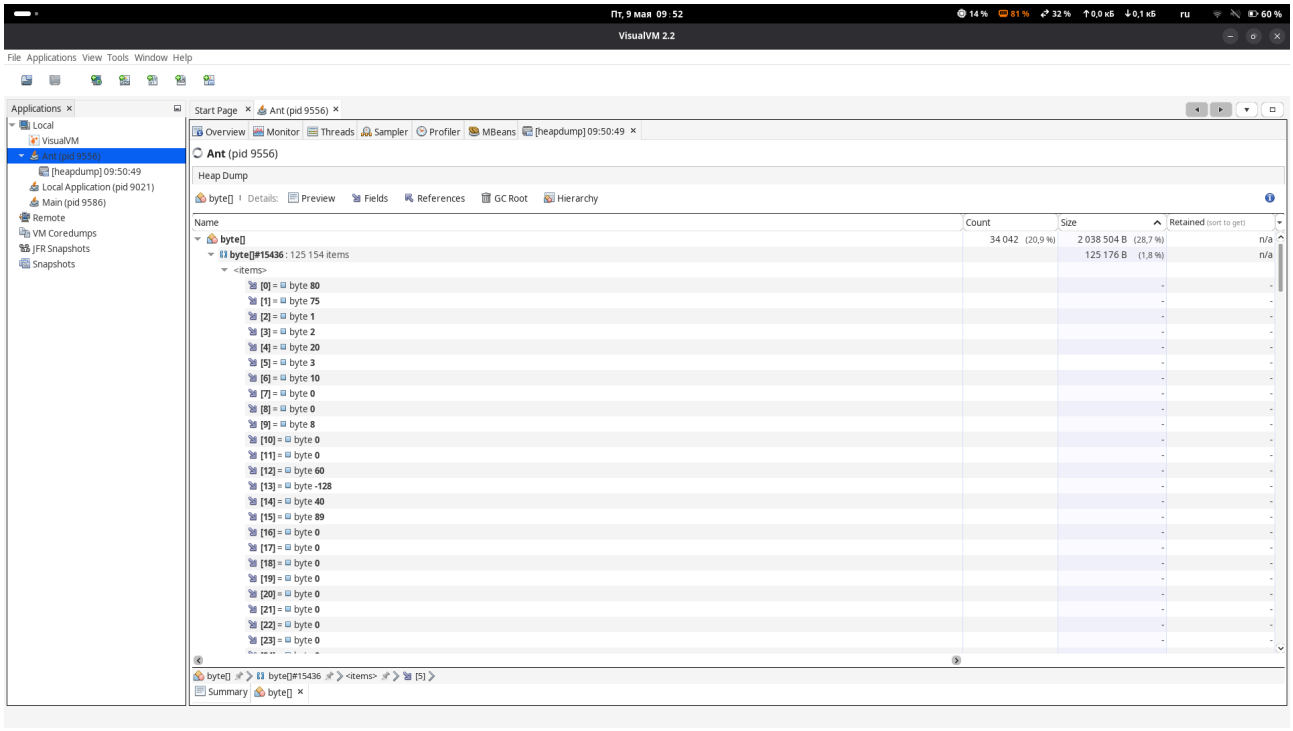


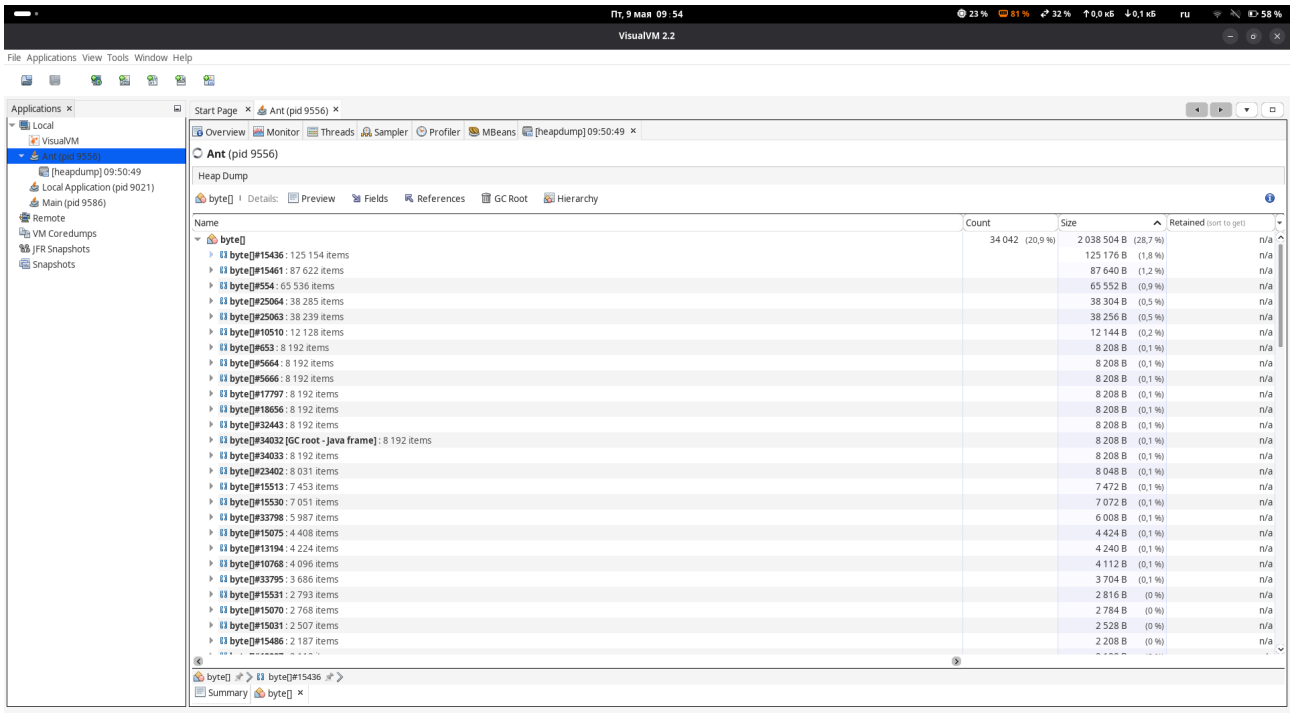


По графику и скриншоту видно, что у нас происходит утечка памяти. Перейдем в вкладку Sampler для просмотра в каком конкретно месте у нас происходит это.

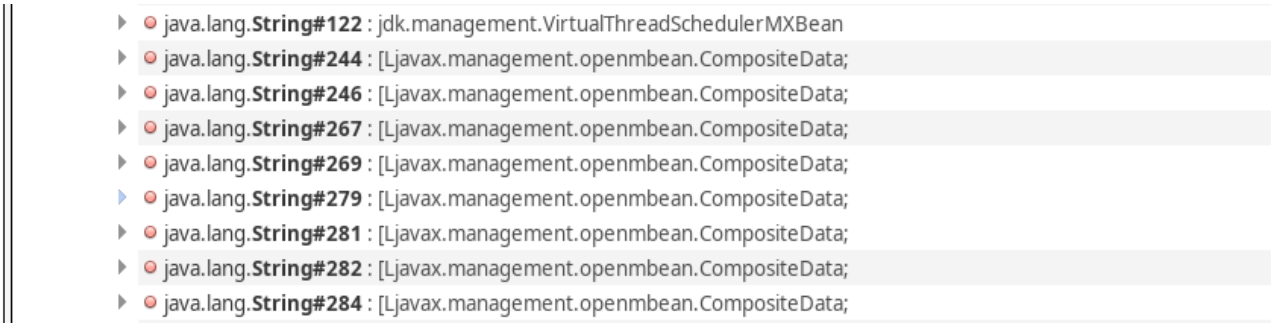


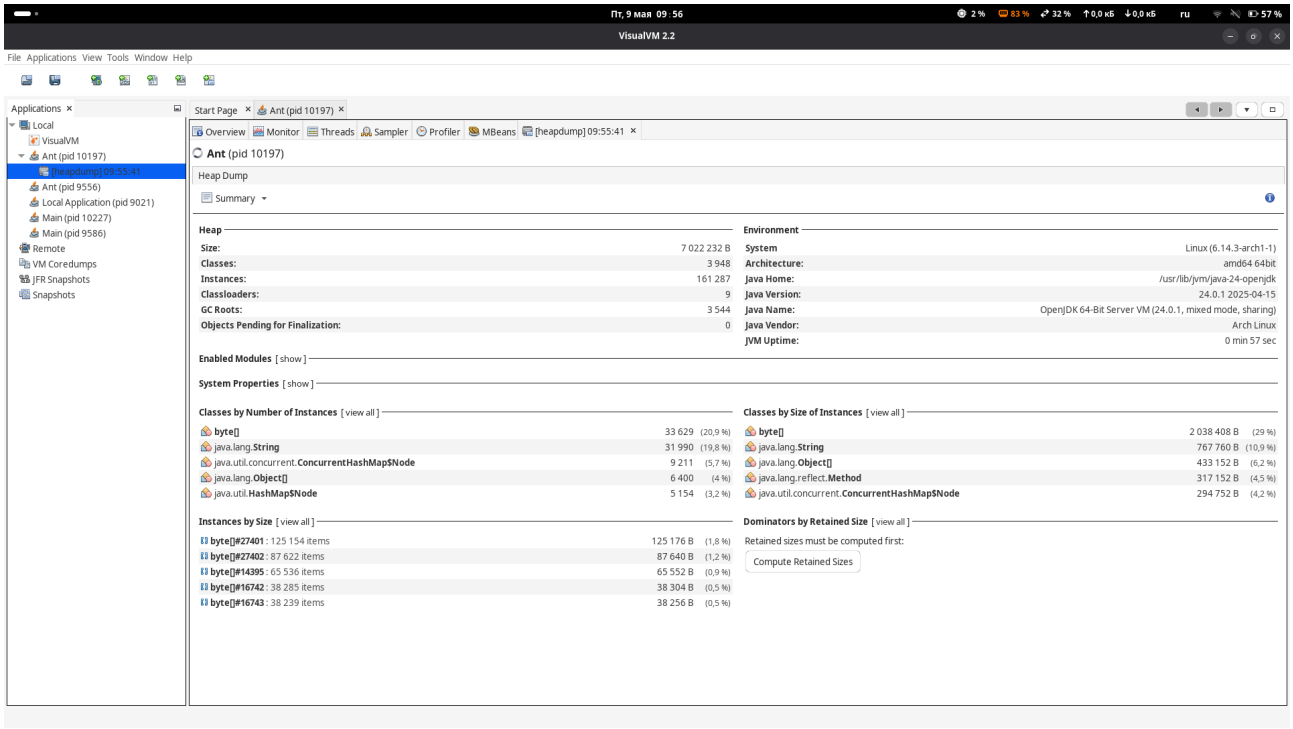
Как видно из скриншота, что-то не так с массивом byte[]. Так как его размер постоянно увеличивается.


Пт, 9 мая 09:52
VisualVM 2.2
File Applications View Tools Window Help
Applications
Local
VisualVM
Ant (pid 9556)
[heapdump] 09:50:49
Local Application (pid 9021)
Main (pid 9586)
Remote
VM Coredumps
JFR Snapshots
Snapshots
Start Page
Ant (pid 9556)
Overview
Monitor
Threads
Sampler
Profiler
MBeans
[heapdump] 09:50:49
Ant (pid 9556)
Heap Dump
byte[]
Details:
Preview
Fields
References
GC Root
Hierarchy
Name
Count
Size
Retained (sort to get)
byte[] 34 042 (20,9 %) 2 038 504 B (28,7 %) n/a
byte[]#15436 : 125 154 items 125 176 B (1,8 %) n/a
<items>
[0] = byte 80
[1] = byte 75
[2] = byte 1
[3] = byte 2
[4] = byte 20
[5] = byte 3
[6] = byte 10
[7] = byte 0
[8] = byte 0
[9] = byte 8
[10] = byte 0
[11] = byte 0
[12] = byte 60
[13] = byte -128
[14] = byte 40
[15] = byte 89
[16] = byte 0
[17] = byte 0
[18] = byte 0
[19] = byte 0
[20] = byte 0
[21] = byte 0
[22] = byte 0
[23] = byte 0
byte[] byte[]#15436 <items> [5]
Summary
byte[]


Пт, 9 мая 09:54
VisualVM 2.2
File Applications View Tools Window Help
Applications
Local
VisualVM
Ant (pid 9556)
[heapdump] 09:50:49
Local Application (pid 9021)
Main (pid 9586)
Remote
VM Coredumps
JFR Snapshots
Snapshots
Start Page
Ant (pid 9556)
Overview
Monitor
Threads
Sampler
Profiler
MBeans
[heapdump] 09:50:49
Ant (pid 9556)
Heap Dump
byte[]
Details:
Preview
Fields
References
GC Root
Hierarchy
Name
Count
Size
Retained (sort to get)
byte[] 34 042 (20,9 %) 2 038 504 B (28,7 %) n/a
byte[]#15436 : 125 154 items 125 176 B (1,8 %) n/a
byte[]#15461 : 87 622 items 87 640 B (1,2 %) n/a
byte[]#554 : 65 536 items 65 552 B (0,9 %) n/a
byte[]#25064 : 38 285 items 38 304 B (0,5 %) n/a
byte[]#25063 : 38 239 items 38 256 B (0,5 %) n/a
byte[]#10510 : 12 128 items 12 144 B (0,2 %) n/a
byte[]#653 : 8 192 items 8 208 B (0,1 %) n/a
byte[]#5664 : 8 192 items 8 208 B (0,1 %) n/a
byte[]#5666 : 8 192 items 8 208 B (0,1 %) n/a
byte[]#17797 : 8 192 items 8 208 B (0,1 %) n/a
byte[]#18656 : 8 192 items 8 208 B (0,1 %) n/a
byte[]#32443 : 8 192 items 8 208 B (0,1 %) n/a
byte[]#34032 [GC root - Java frame] : 8 192 items 8 208 B (0,1 %) n/a
byte[]#34033 : 8 192 items 8 208 B (0,1 %) n/a
byte[]#23402 : 8 031 items 8 048 B (0,1 %) n/a
byte[]#15513 : 7 453 items 7 472 B (0,1 %) n/a
byte[]#15530 : 7 051 items 7 072 B (0,1 %) n/a
byte[]#33798 : 5 987 items 6 008 B (0,1 %) n/a
byte[]#15075 : 4 408 items 4 424 B (0,1 %) n/a
byte[]#13194 : 4 224 items 4 240 B (0,1 %) n/a
byte[]#10768 : 4 096 items 4 112 B (0,1 %) n/a
byte[]#33795 : 3 686 items 3 704 B (0,1 %) n/a
byte[]#15531 : 2 793 items 2 816 B (0 %) n/a
byte[]#15070 : 2 768 items 2 784 B (0 %) n/a
byte[]#15031 : 2 507 items 2 528 B (0 %) n/a
byte[]#15486 : 2 187 items 2 208 B (0 %) n/a
byte[] byte[]#15436
Summary
byte[]

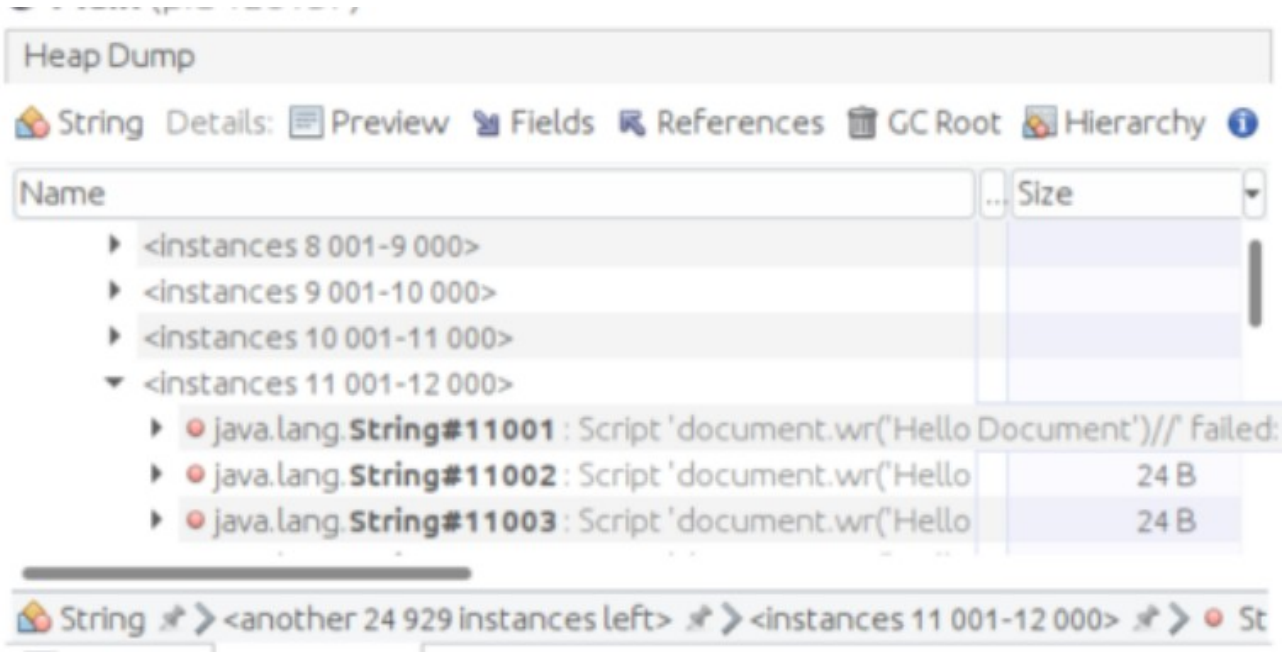
A

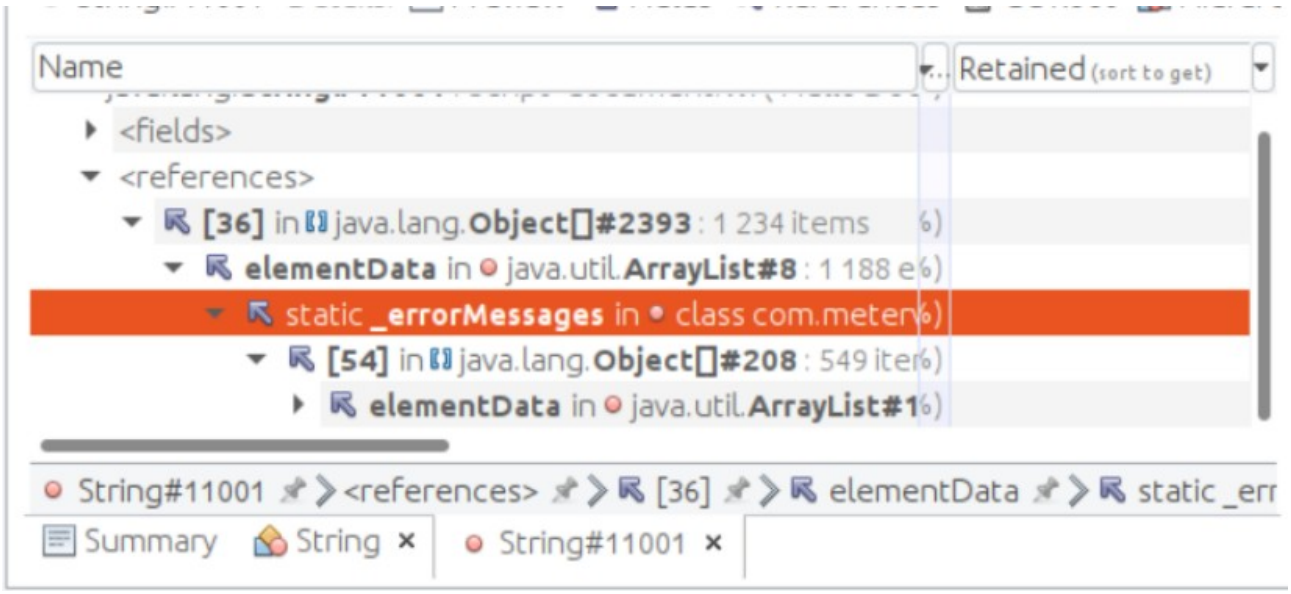

java.lang.String#122 : jdk.management.VirtualThreadSchedulerMXBean
java.lang.String#244 : [Ljavax.management.openmbean.CompositeData;
java.lang.String#246 : [Ljavax.management.openmbean.CompositeData;
java.lang.String#267 : [Ljavax.management.openmbean.CompositeData;
java.lang.String#269 : [Ljavax.management.openmbean.CompositeData;
java.lang.String#279 : [Ljavax.management.openmbean.CompositeData;
java.lang.String#281 : [Ljavax.management.openmbean.CompositeData;
java.lang.String#282 : [Ljavax.management.openmbean.CompositeData;
java.lang.String#284 : [Ljavax.management.openmbean.CompositeData;



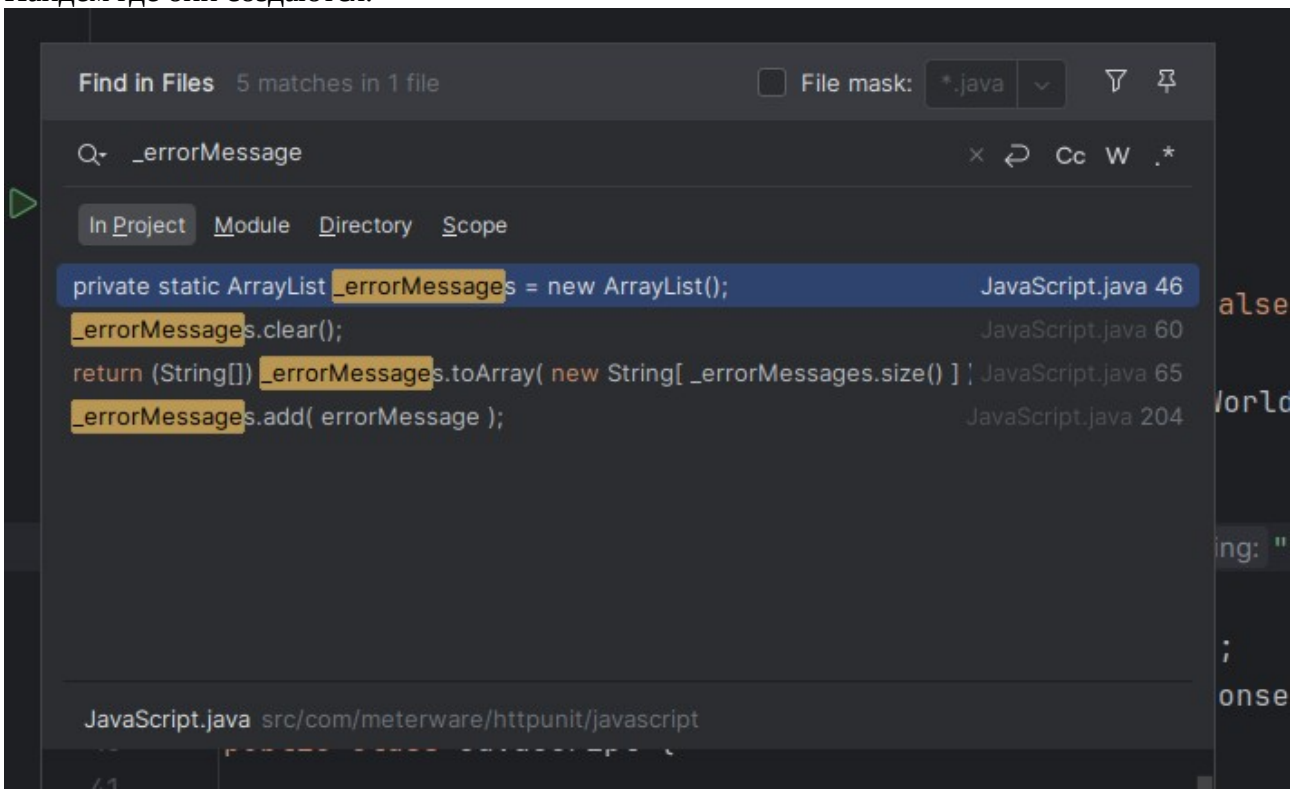
Можем заметить, что создается очень много String и массивов byte и потом их Garbage Collector не убивает, тем самым они остаются в памяти и еще они создаются каждую итерацию. Что приводит к ошибке OutOfMemoryError

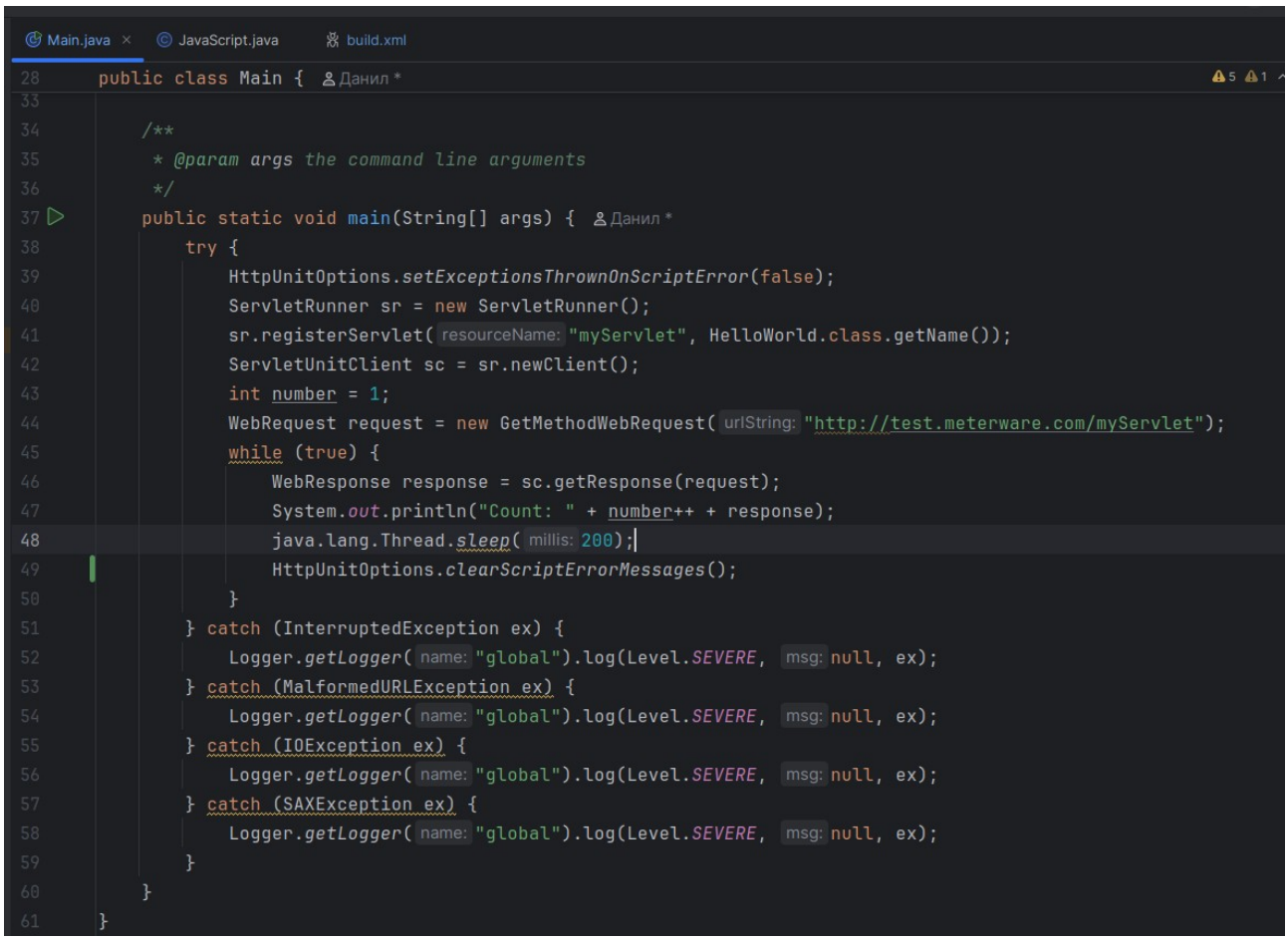
Найдем повторяющиеся строчки:





Найдем где они создаются.





Выводы по работе.

В ходе выполнения этой лабораторной работы я очень устал… Сначала было необходимо вспомнить, что такое JSF, потом написать Bean, потом узнать, что Jconsole поддерживает только ManagedBean. Сделал костыль, который позволяет регистрировать CDI бины как ManagedBean. С Jconsole разобрался, настало время VisualVM, первое что удивило, оно не захотело запускаться на Java 24, потому что некоторые методы стали deprecated и компьютер отказывался запускать эту утилиту. С установкой старой версии Java, утилита заработала. Но тут возникла проблема в том, что не было вкладки для просмотра содержимого ManagedBean, тут моя неокрепшая нервная система запаниковала и начала консультироваться с высшим разумом для прояснения этого вопроса. Выяснилось, что необходимо установить плагин для их отображения. Всё. Бины видны, но тут возникла проблема в том, что компоненты моих бинов отображаются синим цветом, а не черным, чтобы построить для них график. В итоге я 3 часа решал этот вопрос, не решил и лег спать, на следующее утро я проснулся и понял, что у меня в интерфейсе для ManagedBean указаны сеттеры, а чтобы делать график компоненты должны быть read-only. Починил, все заработало. Настало время HttpUnit… Это монстр. Он съел всю мою память на компьютере… И моей задачей было найти место, где протикает у нас память. В ходе поисков я научился делать HeapDump, да что уж там, я нашел OQL скрипты на StackOverFlow для обнаружения строк, которые часто появляются. Одно дело их найти с помощью запроса, а другое дело в HeapDump, чтобы найти какая функция их генерирует. Около 50000 строчек String было просмотрено мной, чтобы найти их! То ли я их пропускал своим глазом, либо что, но поиск с первого раза не сработал, пришлось

пересматривать… И случилось чудо, я их нашел! Далее было дело техники, найти эту функцию, понять что массив у нас статический, а это значит, он сам не умрет, надо ~~убивать~~ отищать.

На самом деле лаба очень устарела, как морально так и физически. Большую часть своего времени я либо делал костыли, чтобы у меня заработали утилиты, либо настраивал рабочее окружение, чтобы эти утилиты корректно работали.

Что же тогда предложить вместо этой лабы. Можно использовать Grafana, Prometheus для сбора метрик и построения дашбордов. Так студенты получат реливантый опыт, а не опыт страдания...