

**Министерство науки и высшего образования Российской Федерации**  
федеральное государственное автономное образовательное учреждение высшего  
образования  
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

по лабораторной работе №4  
Вариант №367958

по дисциплине **«Основы программной инженерии»**

Автор: Баянов Равиль Динарович

Факультет: ПИиКТ

Группа: Р3234

Преподаватель: Кулинич Ярослав Вадимович



Санкт-Петербург, 2024

# Оглавление

Задание.....	3
Исходный код MBean-классов.....	4
Интерфейс MBean.....	4
Методы реализуемого интерфейса MBean.....	4
Класс-инструмент для регистрации MBean.....	5
JConsole.....	7
Показания MBean-классов.....	7
Время с момента запуска JVM.....	9
VisualVM.....	10
График изменения показаний MBean-классов.....	10
Имя потока, потребляющего наибольший процент времени CPU.....	11
Проблемы с производительностью в программе.....	12
Список используемой литературы.....	15
Вывод.....	16

# 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 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 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 MBean-классов

## Интерфейс MBean

```
1. package ru.ravvcheck.web3lab.model.services;
2.
3. public interface ResultManagerMBean {
4.     long getDotsCount ();
5.     long getMissedDotsCount ();
6.
7.     Double searchAverageClickInterval ();
8. }
```

## Методы реализуемого интерфейса MBean

```
1. @Override
2.     public long getMissedDotsCount () {
3.         return getDotsCount () - this.results.stream()
4.             .filter(Result::isHit)
5.             .count ();
6.     }
7.
8.     @Override
9.     public long getDotsCount () {
10.        return this.results.size ();
11.    }
12.
13.    @Override
14.    public Double searchAverageClickInterval () {
15.        DateTimeFormatter formatter =
16.            DateTimeFormatter.ofPattern("HH:mm:ss dd.MM.yyyy");
17.        List<ZonedDateTime> zonedDateTimes = results.stream()
18.            .map(result ->
19.                ZonedDateTime.of(LocalDateTime.parse(result.getCurrentTime()),
20.                    formatter), ZoneId.systemDefault()))
21.            .collect(Collectors.toList());
22.        List<Long> differences = zonedDateTimes.stream()
23.            .skip(1)
24.            .map(time -> time.toInstant().toEpochMilli() -
25.                zonedDateTimes.get(zonedDateTimes.indexOf(time) -
26.                    1).toInstant().toEpochMilli())
27.            .collect(Collectors.toList());
28.        return differences.stream()
29.            .mapToDouble(Long::doubleValue)
30.            .average ();
31.    }
```

```

26.         .orElse(0);
27.     }
28.
29.     @Override
30.     public MBeanNotificationInfo[] getNotificationInfo() {
31.         String[] types = new String[]
32.         {AttributeChangeNotification.ATTRIBUTE_CHANGE};
33.         String name =
34.         AttributeChangeNotification.class.getName();
35.         String description = "Miss notification";
36.         MBeanNotificationInfo info = new
37.         MBeanNotificationInfo(types, name, description);
38.         return new MBeanNotificationInfo[]{info};
39.     }

```

## Класс-инструмент для регистрации MBean

```

1. package ru.ravvcheck.web3lab.utils;
2.
3. import jakarta.servlet.ServletContextListener;
4.
5. import javax.management.*;
6. import java.lang.management.ManagementFactory;
7. import java.util.HashMap;
8.
9. public class MBeanRegistryUtil implements ServletContextListener {
10.     private static final HashMap<Class<?>, ObjectName> beans =
11.     new HashMap<>();
12.     public static void registerBean(Object bean, String name) {
13.         try {
14.             String type = bean.getClass().getSimpleName();
15.             ObjectName objectName = new
16.             ObjectName(String.format("ru.ravvcheck.web3lab:type=%s,name=%s", type,
17.             name));
18.             MBeanServer server =
19.             ManagementFactory.getPlatformMBeanServer();
20.             server.registerMBean(bean, objectName);
21.         } catch (MalformedObjectNameException |
22.             NotCompliantMBeanException | InstanceAlreadyExistsException |
23.             MBeanRegistrationException e) {
24.             System.out.println(e.getMessage());
25.         }
26.     }
27.
28.     public static void unregisterBean(Object bean) {
29.         if (!beans.containsKey(bean.getClass())) {

```

```
24.             throw new IllegalArgumentException("Bean not
    registered");
25.         }
26.         ObjectName objectName = beans.get(bean.getClass());
27.         MBeanServer server =
    ManagementFactory.getPlatformMBeanServer();
28.         try {
29.             server.unregisterMBean(objectName);
30.         } catch (InstanceNotFoundException |
    MBeanRegistrationException e) {
31.             System.out.println(e.getMessage());
32.         }
33.     }
34. }
35.
```

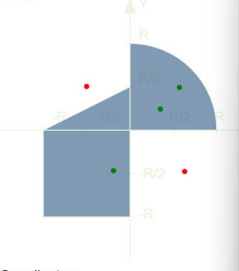
# JConsole

## Показания MBean-классов

The screenshot displays the Java Monitoring & Management Console (JConsole) interface. The background shows a web application titled "Bayanov Ravil \* P... Variant \* 7777" with a graph and coordinate input fields. The JConsole window is open, showing the "Attributes" tab for the "ResultManager" MBean. The "Attribute values" table lists two attributes: "DotsCount" with a value of 4 and "MissedDotsCount" with a value of 2. The "Refresh" button is visible at the bottom of the table.

Name	Value
DotsCount	4
MissedDotsCount	2

Bayanov Ravil \* P  
Variant \* 7777  
Graph



Coordinates  
X Value: -5  
Y Value: Input Y (-3;3)  
R Value: 5.0  
SUBMIT RESET

Java Monitoring & Management Console - pid: 11016 jboss-modules.jar -mp C:\Users\Ravil\OneDrive\Desktop\wildfly-27.0.1.Final\modules\org.jboss...

Connection Window Help

Overview Memory Threads Classes VM Summary MBeans

Attribute value

Name	Value
DotsCount	5

Refresh

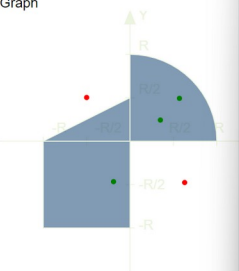
MBeanAttributeInfo

Name	Value
Attribute:	
Name	DotsCount
Description	Attribute exposed for management
Readable	true
Writable	false
Is	false
Type	long

Descriptor

Name	Value
------	-------

Bayanov Ravil \* P  
Variant \* 7777  
Graph



Coordinates  
X Value: -5  
Y Value: Input Y (-3;3)  
R Value: 5.0  
SUBMIT RESET

Java Monitoring & Management Console - pid: 11016 jboss-modules.jar -mp C:\Users\Ravil\OneDrive\Desktop\wildfly-27.0.1.Final\modules\org.jboss...

Connection Window Help

Overview Memory Threads Classes VM Summary MBeans

Attribute value

Name	Value
MissedDotsCount	2

Refresh

MBeanAttributeInfo

Name	Value
Attribute:	
Name	MissedDotsCount
Description	Attribute exposed for management
Readable	true
Writable	false
Is	false
Type	long

Descriptor

Name	Value
------	-------



Bayanov Ravil \* P  
Variant \* 7777  
Graph

Coordinates  
X Value: \* -5  
Y Value: \* Input Y (-3;3)  
R Value: \* 5.0  
SUBMIT RESET

Java Monitoring & Management Console - pid: 11016 jboss-modules.jar -mp C:\Users\Ravil\Check1\wildfly-27.0.1.Final\modules org.jb...

Overview Memory Threads Classes VM

Operation return value

Operation: 14750.0

MBeanOp

Name

searchAverageClickInterval

Description Operation exposed for management

Impact UNKNOWN

ReturnType java.lang.Double

Descriptor

Name	Value
------	-------



Bayanov Ravil \* P  
Variant \* 7777  
Graph

Coordinates  
X Value: \* -5  
Y Value: \* Input Y (-3;3)  
R Value: \* 5.0  
SUBMIT RESET

Java Monitoring & Management Console - pid: 11016 jboss-modules.jar -mp C:\Users\Ravil\Check1\wildfly-27.0.1.Final\modules org.jb...

Overview Memory Threads Classes VM Summary MBeans

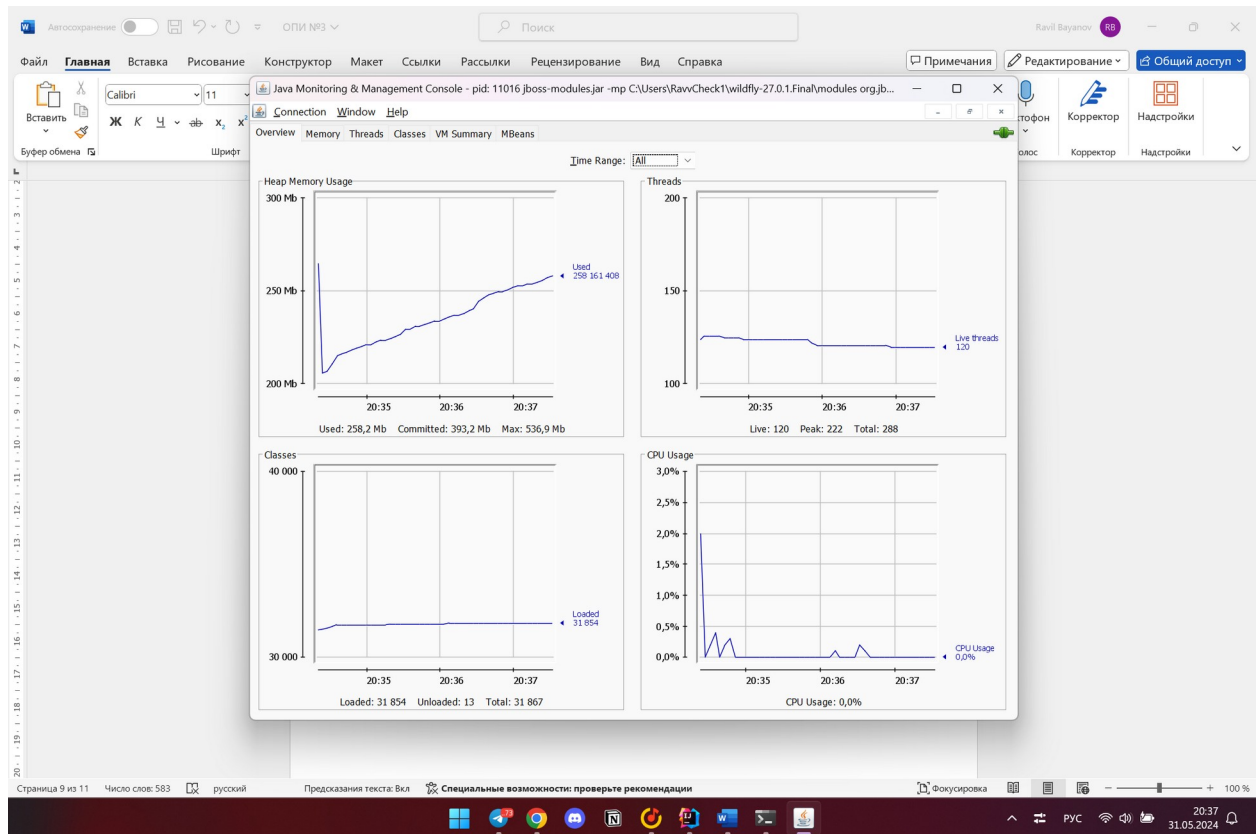
Notification buffer

TimeStamp	Type	UserData	SeqNum	Message	Event	Source
20:36:34.274	Misses		2	Two misses in a ...	javax.manageme...	ResultManager
20:36:29.916	Misses		1	Two misses in a ...	javax.manageme...	ResultManager

Subscribe Unsubscribe Clear

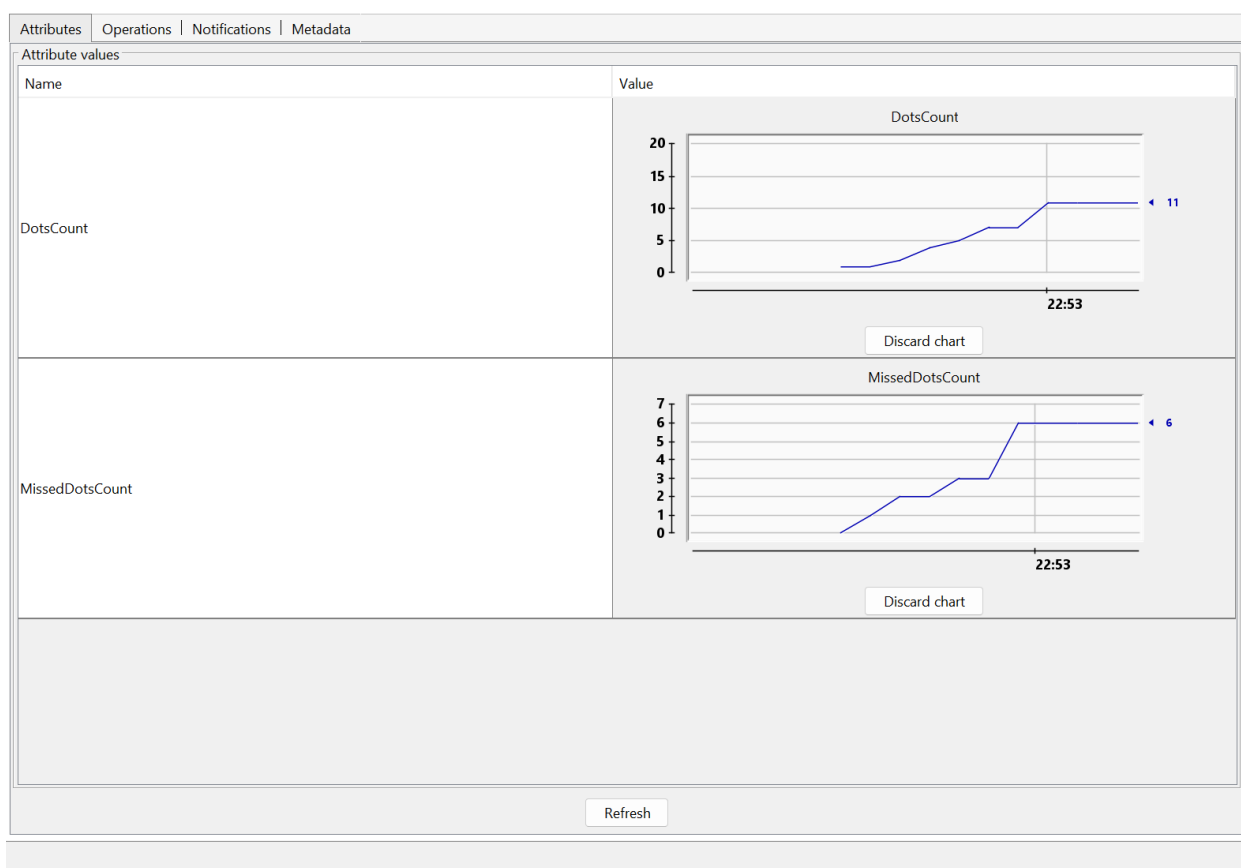


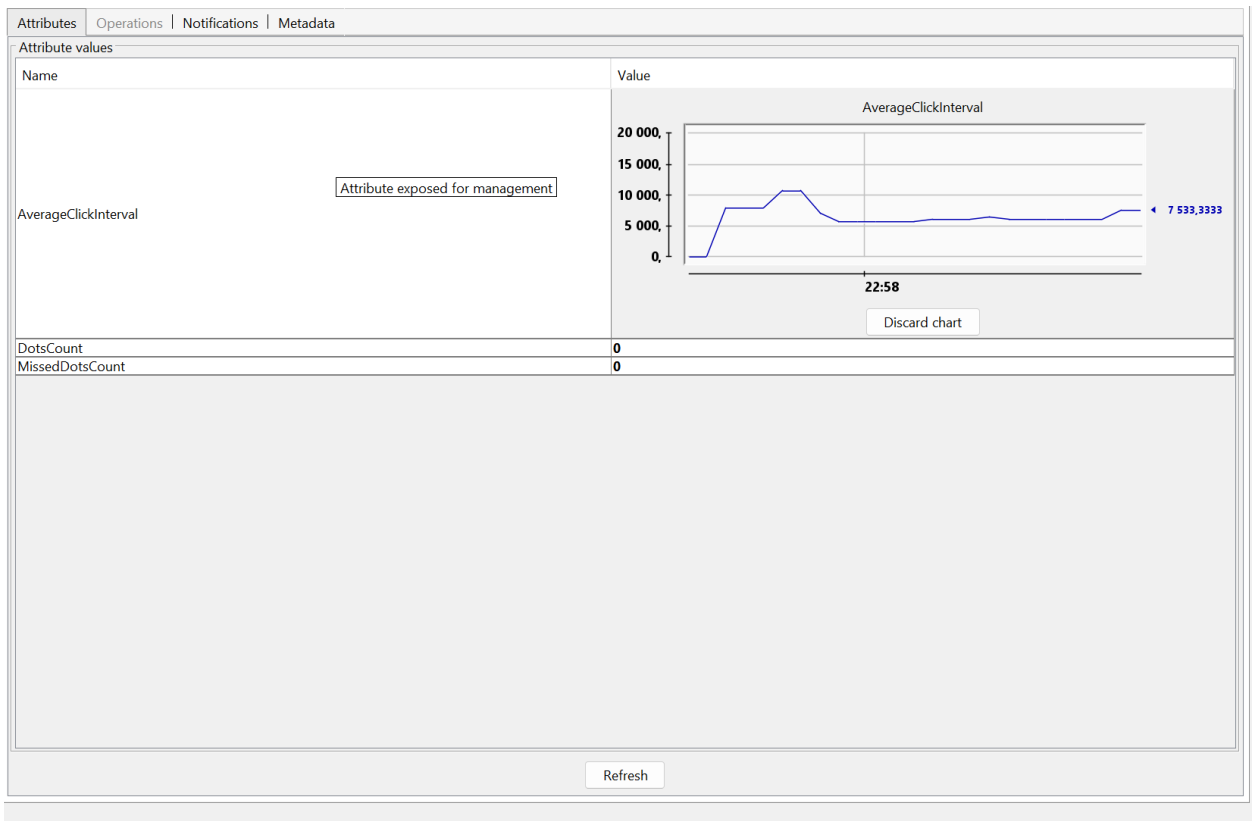
# Время с момента запуска JVM



# VisualVM

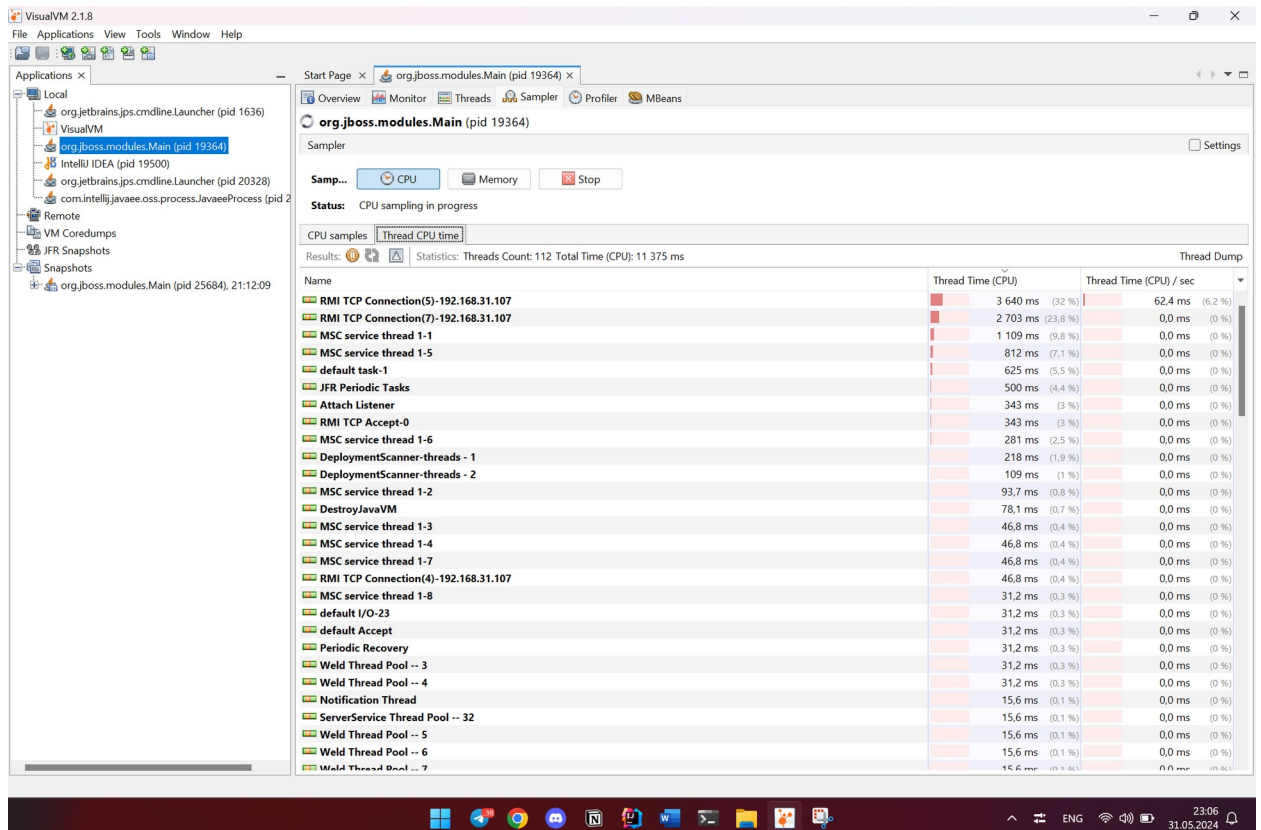
## График изменения показаний MBean-классов





## Имя потока, потребляющего наибольший процент времени CPU

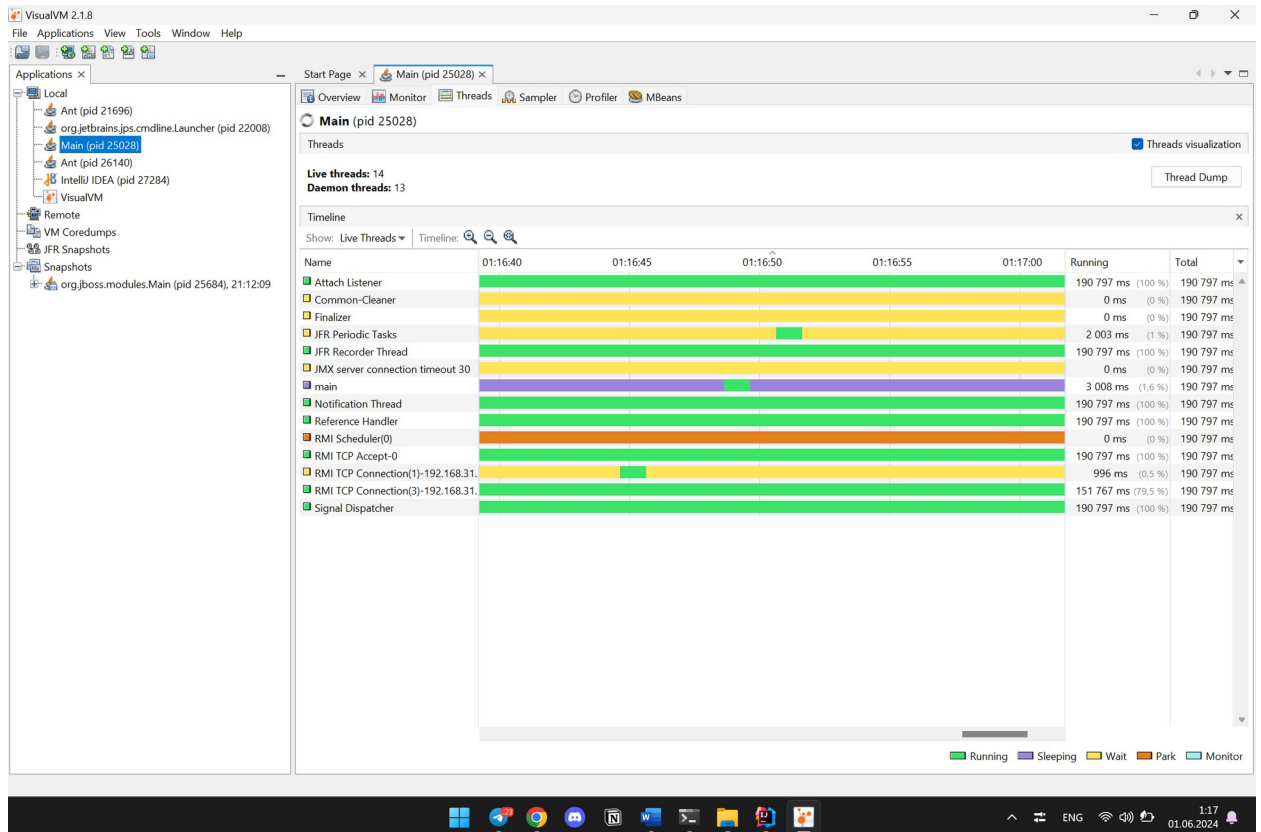
RMI TCP Connection(5)-192.168.31.107



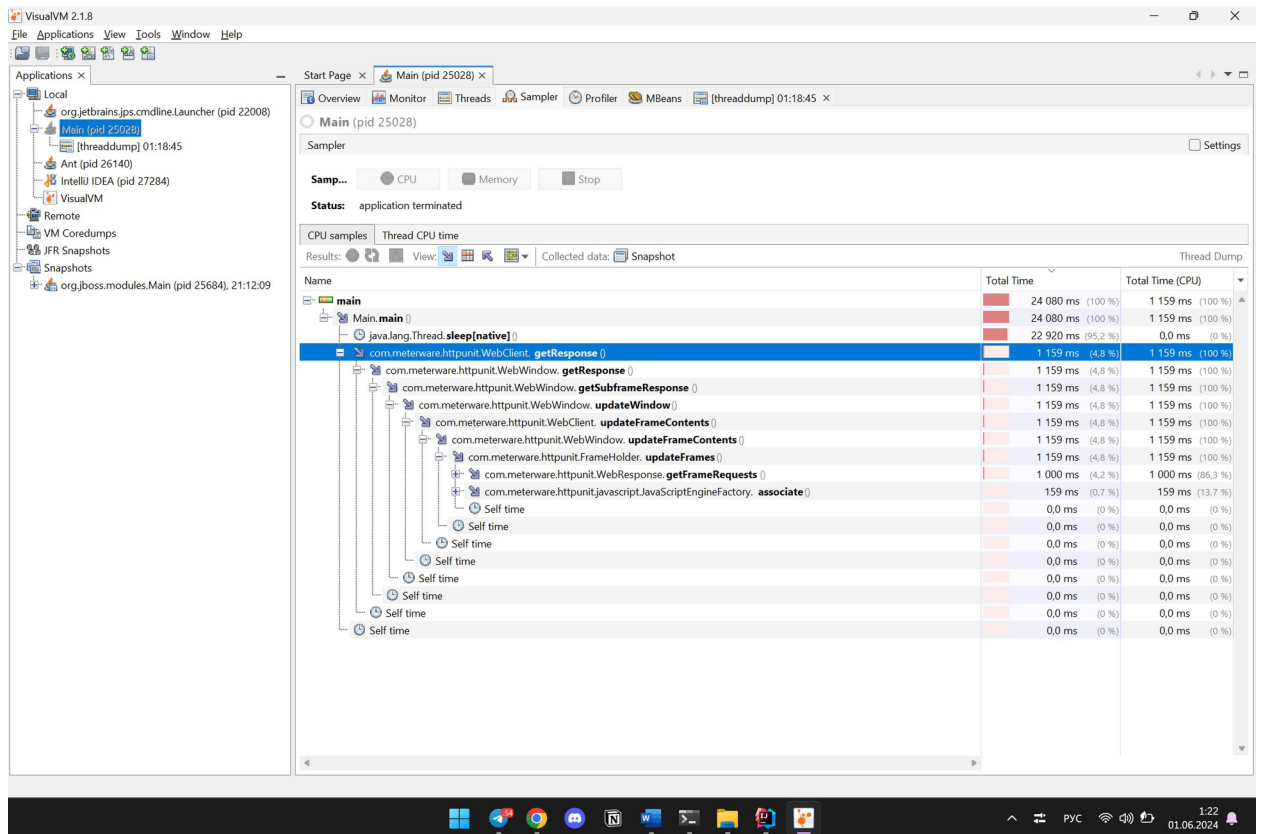


# Проблемы с производительностью в программе

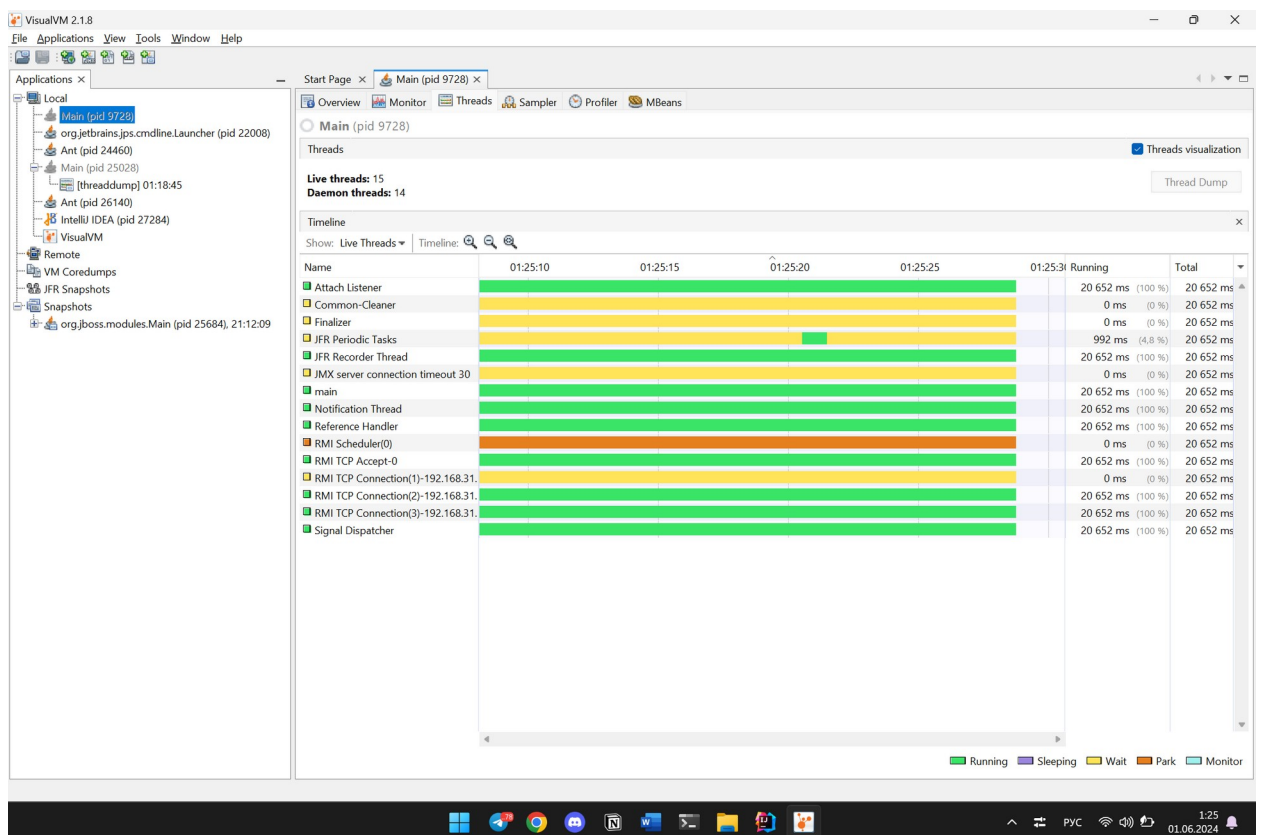
Запустим программу с помощью Ant и будем мониторить программу с помощью VisualVM



Заметим, что наша программа, а именно поток `main` основную часть времени находится в состоянии `Sleeping`. Попытаемся найти место, которое снижает скорость выполнения нашей программы. Заметим, что строчка `java.lang.Thread.sleep(200);` сильно задерживает программу. Следовательно, закомментируем её, тем самым ускорим нашу программу.



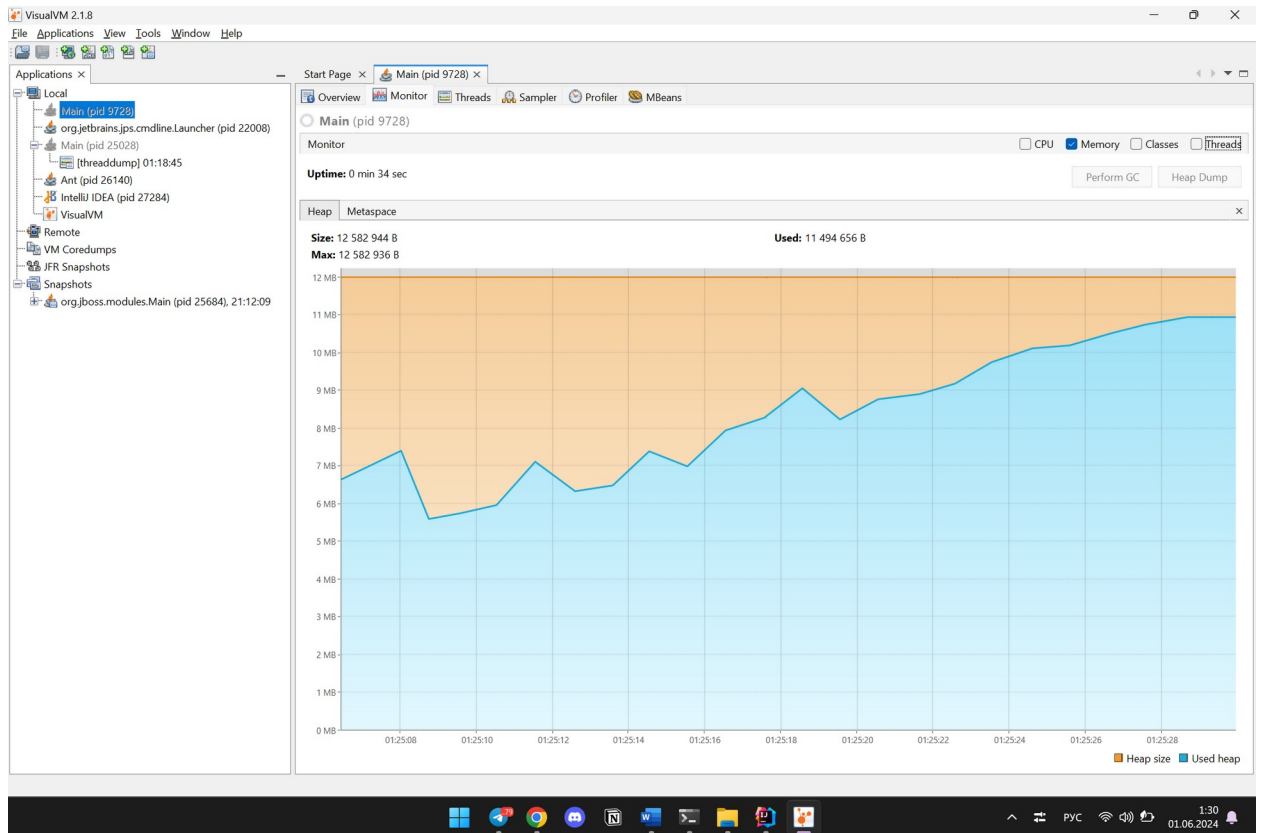
Запустим программу снова. И увидим, что наш поток main всегда running.



Установим в project.properties параметр для размера кучи в 12мб. И запустим программу снова.

```
run.jvmargs=-Xmx12m
```

Заметим, что объём свободной памяти в кучи стремительно уменьшается. Констатируем факт утечки памяти в нашей программе.

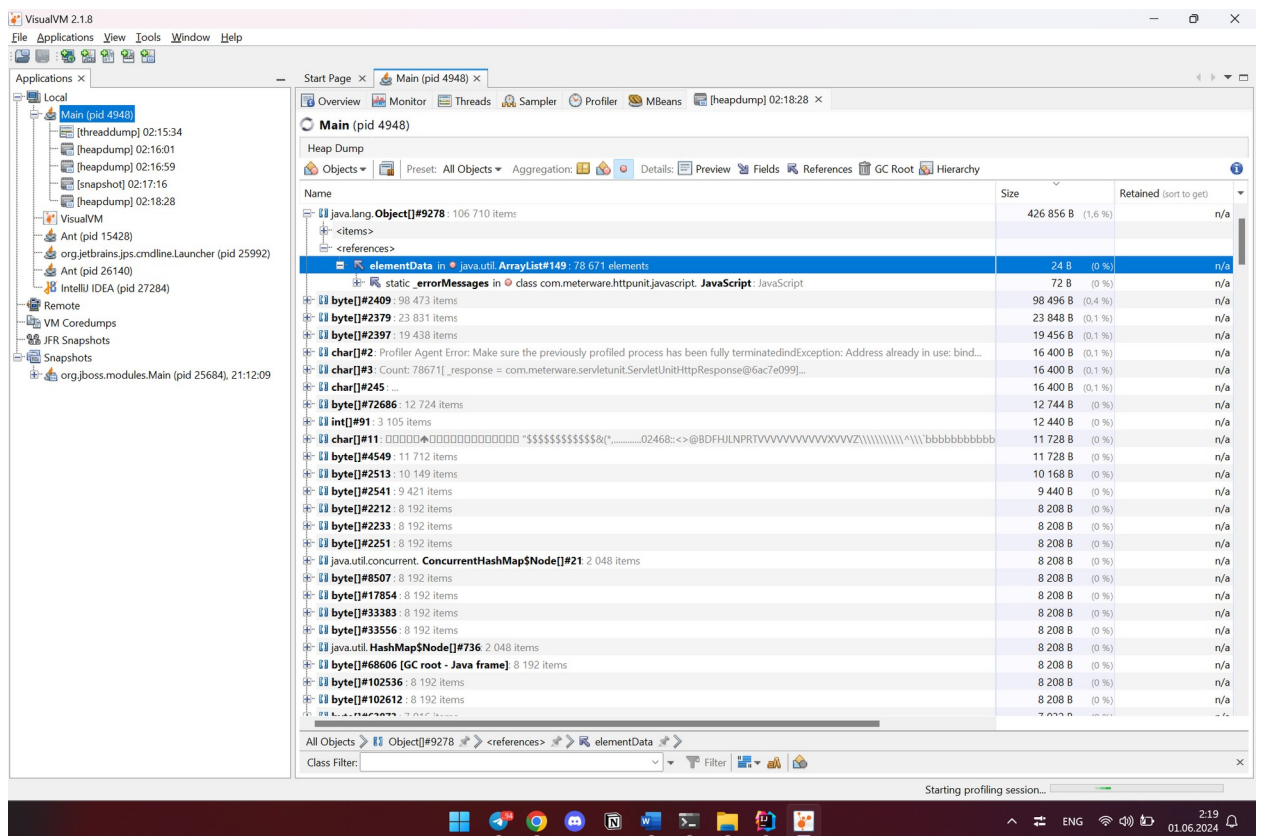


Со временем получим ошибку, связанную с нехваткой памяти.

```
[java] Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
[java]   at org.cyberneko.html.HTMLScanner$CurrentEntity.<init>(Unknown Source)
[java]   at org.cyberneko.html.HTMLScanner.setInputSource(Unknown Source)
[java]   at org.cyberneko.html.HTMLConfiguration.setInputSource(Unknown Source)
[java]   at org.cyberneko.html.HTMLConfiguration.parse(Unknown Source)
[java]   at org.apache.xerces.parsers.XMLParser.parse(Unknown Source)
[java]   at org.apache.xerces.parsers.DOMParser.parse(Unknown Source)
[java]   at com.meterware.httpunit.parsing.NekoHTMLParser.parse(NekoHTMLParser.java:41)
[java]   at com.meterware.httpunit.HTMLPage.parse(HTMLPage.java:255)
[java]   at com.meterware.httpunit.WebResponse.getReceivedPage(WebResponse.java:1109)
[java]   at com.meterware.httpunit.WebResponse.getFrames(WebResponse.java:1098)
[java]   at com.meterware.httpunit.WebResponse.getFrameRequests(WebResponse.java:875)
[java]   at com.meterware.httpunit.FrameHolder.updateFrames(FrameHolder.java:179)
[java]   at com.meterware.httpunit.WebWindow.updateFrameContents(WebWindow.java:252)
[java]   at com.meterware.httpunit.WebClient.updateFrameContents(WebClient.java:485)
```



Посмотрим на Heap Dump и попробуем определить, что в программе занимает больше всего памяти.



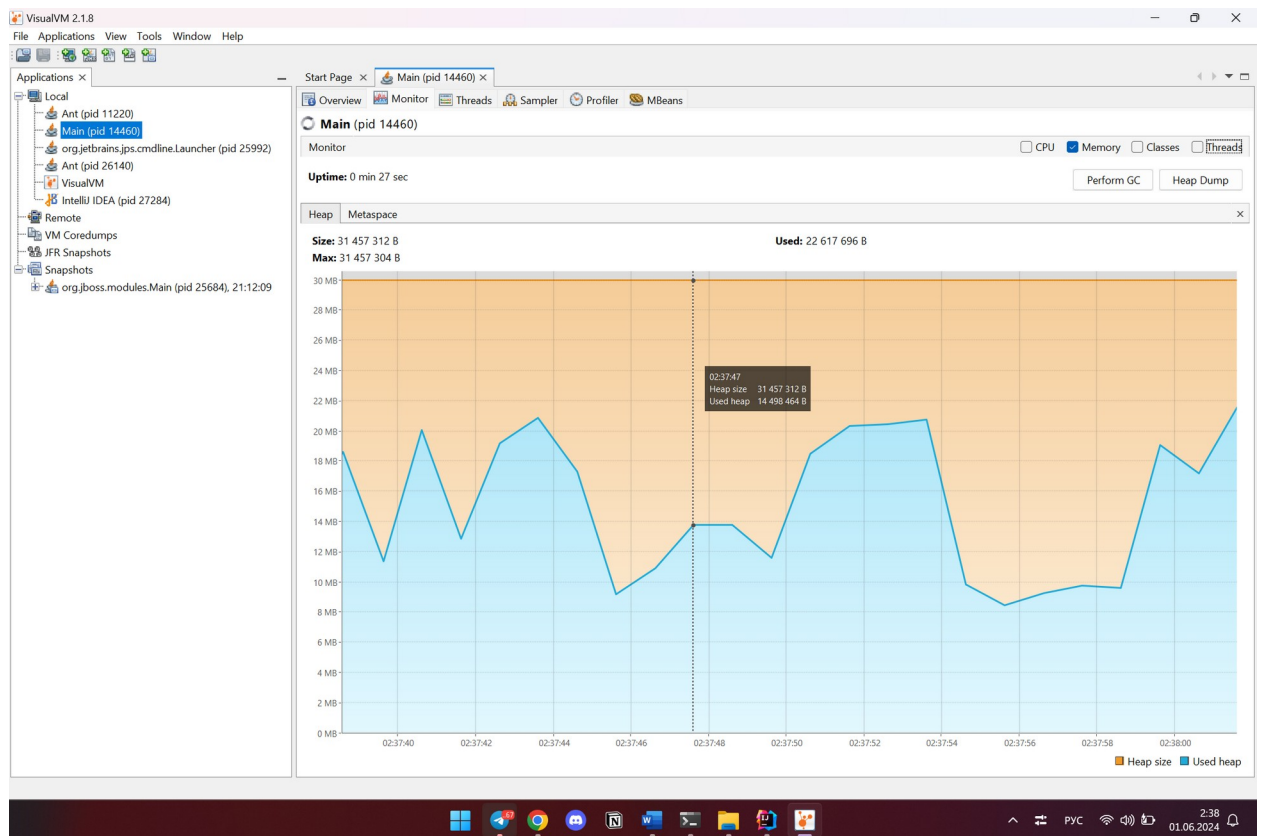
Обнаруживается массив со String (с сообщениями об ошибках).

```
public static void clearScriptErrorMessages() {
    getScriptingEngine().clearErrorMessages();
}
```

С помощью удобного поиска в IDEA через `ctrl` направляемся вот к такому методу, который как мы видим нигде не вызывается (а должен!!!)

Решим проблему с утечкой памяти вызовом этого метода в `main` сразу после выполнения запроса. И снова запустим программу.

```
while (true) {
    WebResponse response = sc.getResponse(request);
    System.out.println("Count: " + number++ + response);
    HttpUnitOptions.clearScriptErrorMessages();
    java.lang.Thread.sleep(200);
}
catch (InterruptedException ex) {
```



Ошибка исправлена!

# Список используемой литературы

- Клименков С. В., Цопа Е. А., Козин И. О. Конспект лекций по дисциплине “Основы программной инженерии”, 2021. Электронный ресурс / Режим доступа:  
<https://se.ifmo.ru/documents/10180/671657/%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D0%B8+%D0%BF%D0%BE+%D0%9E%D0%9F%D0%98+v1.3.3.pdf/50ce1e06-00d9-4900-be9c-a3316a746d6d>

-

## **Вывод**

При выполнении данной лабораторной работы я узнал много про JMX, про мониторинг и профилирование программ. Написал небольшой собственный MBean и отмерил его показания с помощью таких программ как: JConsole и VisualVM. Также научился пользоваться VisualVM для обнаружения утечек памяти и их устранения.