МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

**Факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА №4**

по дисциплине

“Методы и средства программной инженерии”

Вариант № -1

**Студенты:**

Чухно Матвей Романович

Воробьев Кирилл Олегович

            Группа P3230 & P3231

**Преподаватель:**

Письмак Алексей Евгеньевич

Изображение выглядит как текст, знак

Автоматически созданное описание

Санкт-Петербург, 2022

**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 количество установленных пользователем точек стало кратно 5, разработанный MBean должен отправлять оповещение об этом событии.
* MBean, определяющий площадь получившейся фигуры.

1. С помощью утилиты JConsole провести мониторинг программы:

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

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

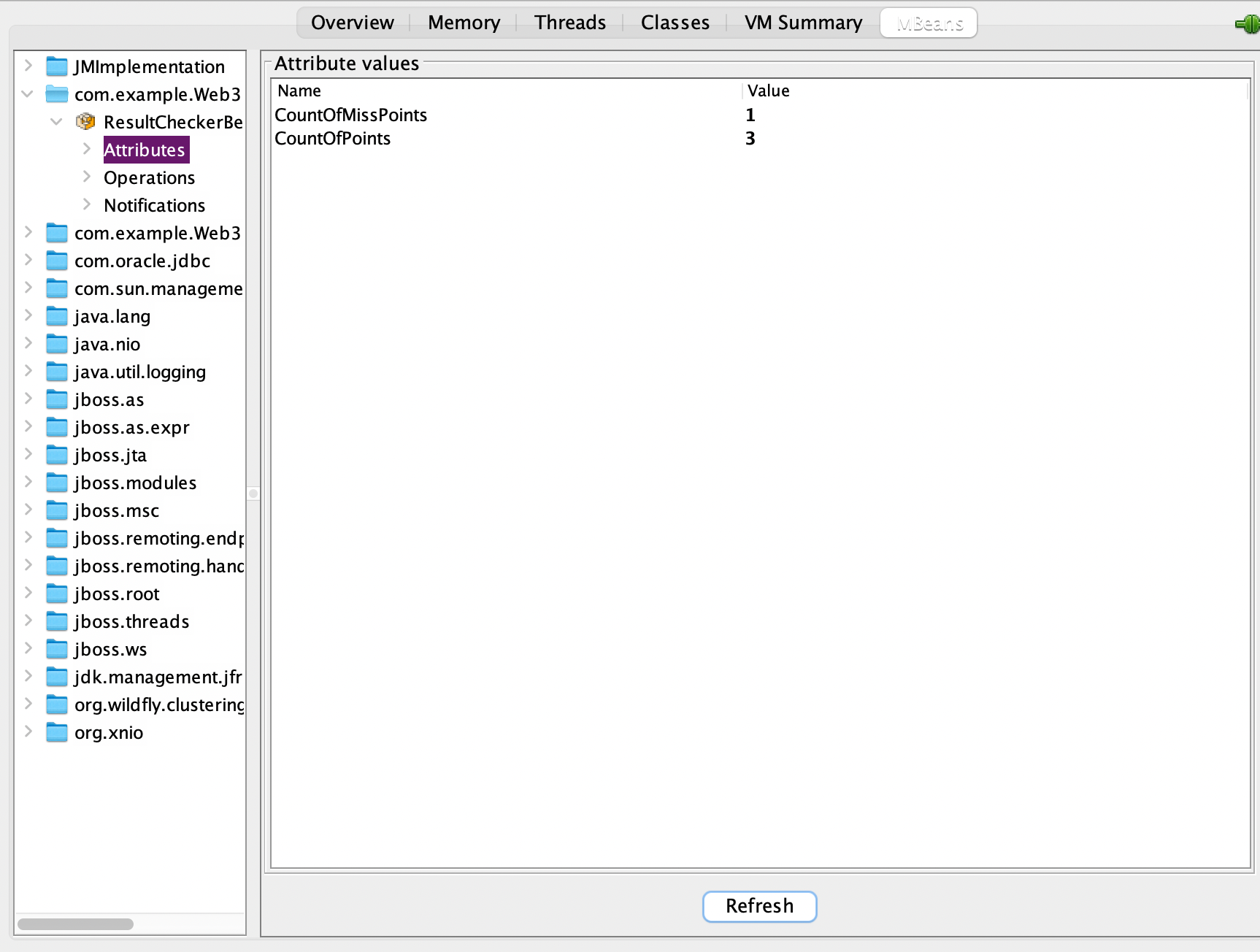
* Снять график изменения показаний MBean-классов, разработанных в ходе выполнения задания 1, с течением времени.
* Определить имя потока, потребляющего наибольший процент времени CPU.

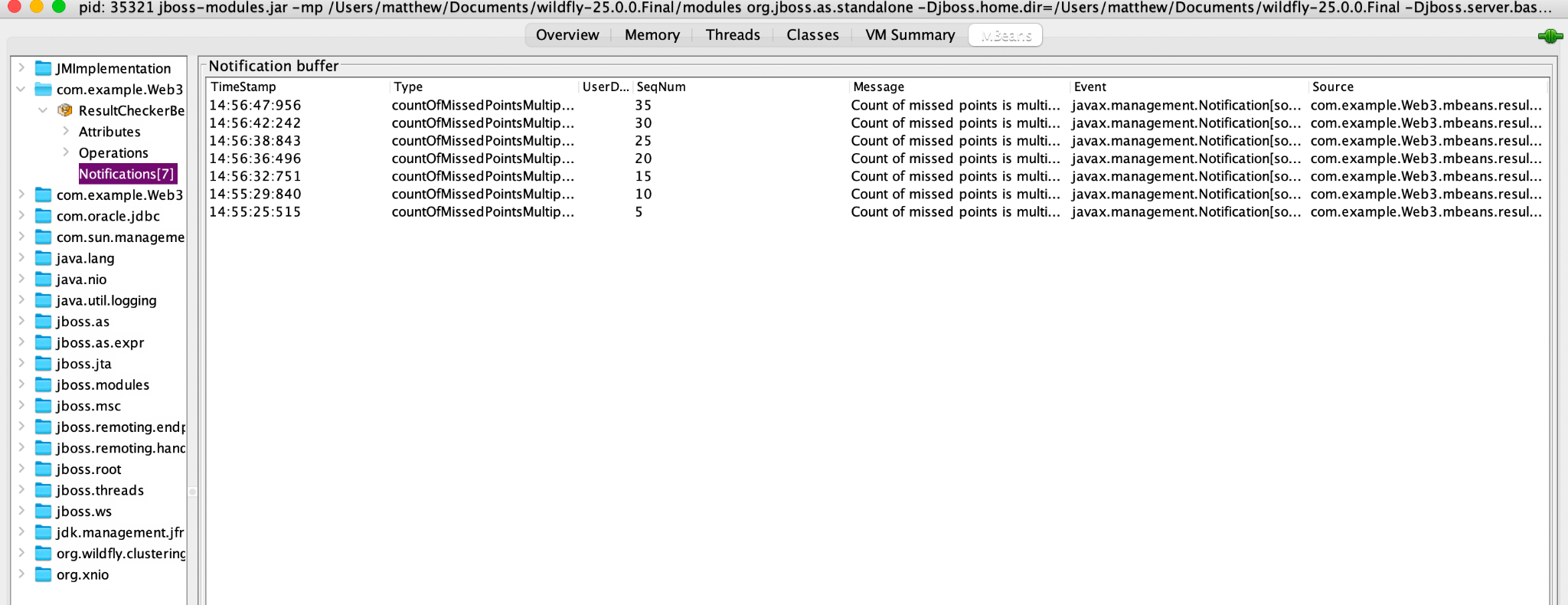
1. С помощью утилиты VisualVM и профилировщика IDE NetBeans, Eclipse или Idea локализовать и устранить проблемы с производительностью в [программе](https://se.ifmo.ru/documents/10180/189115/HttpUnit.tar.gz/7bf1032e-d16e-be85-c71b-dbe73c0178ba?t=1651168887037&download=true). По результатам локализации 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.

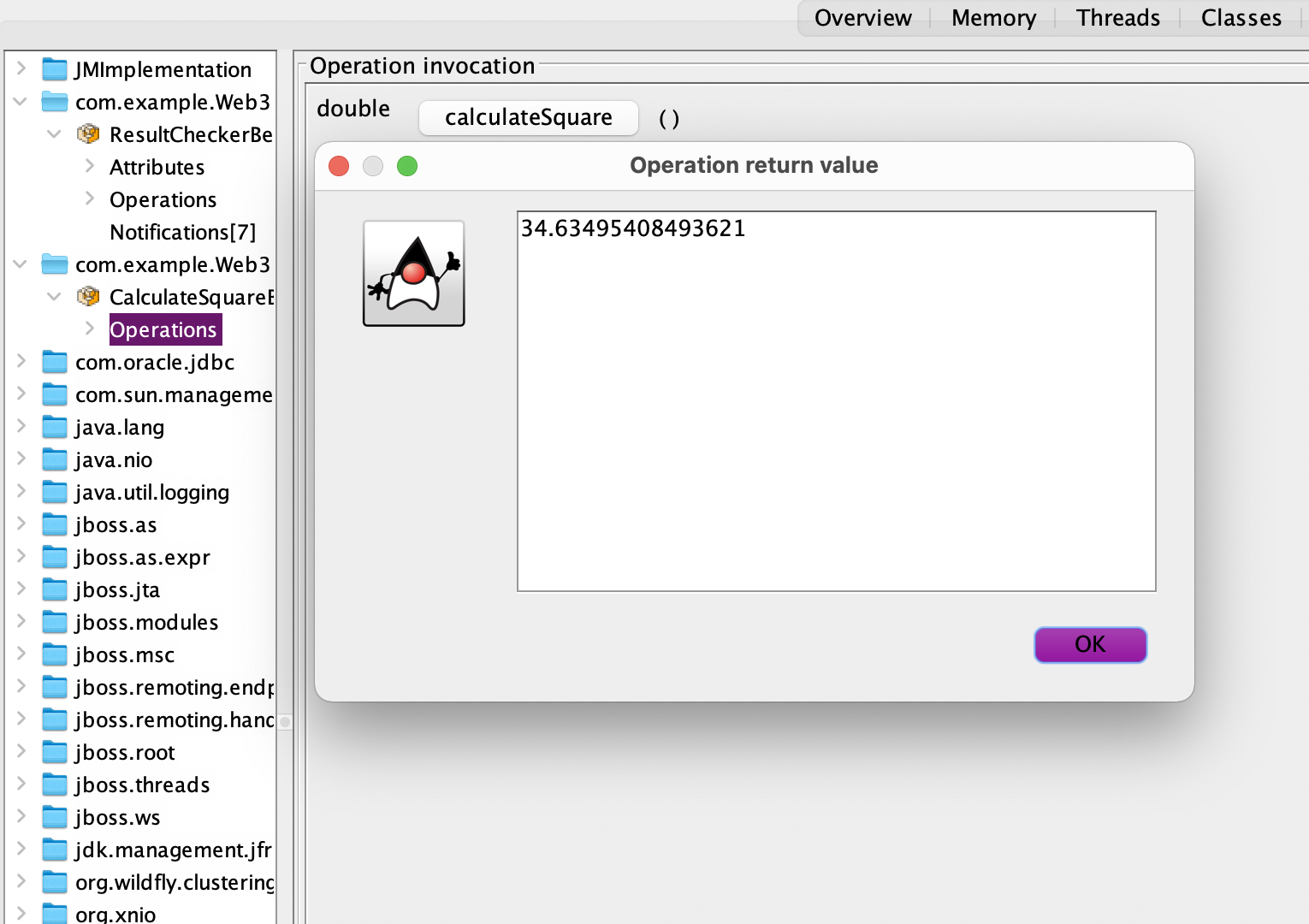
**JConsole:**

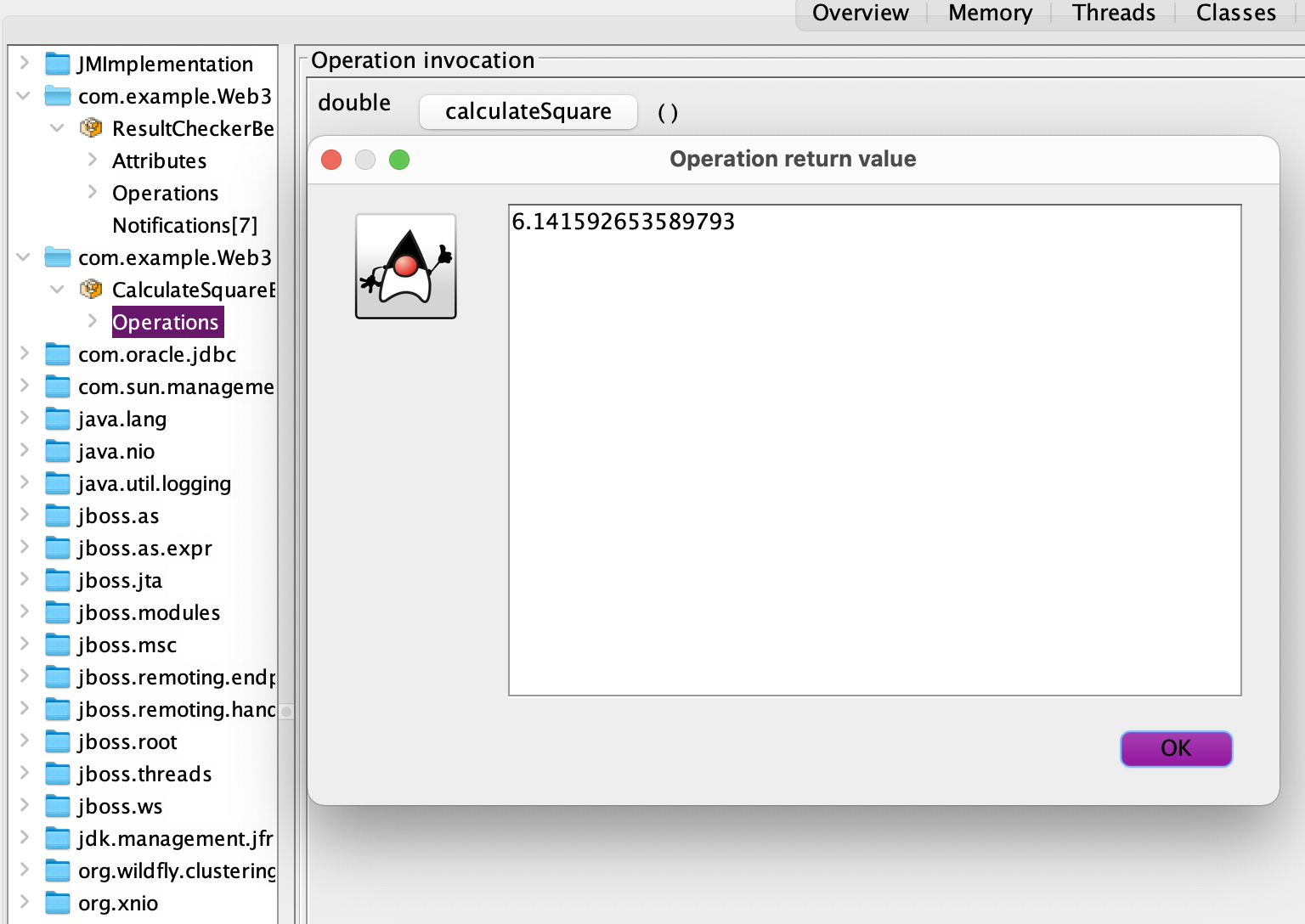
**ResultMBean**

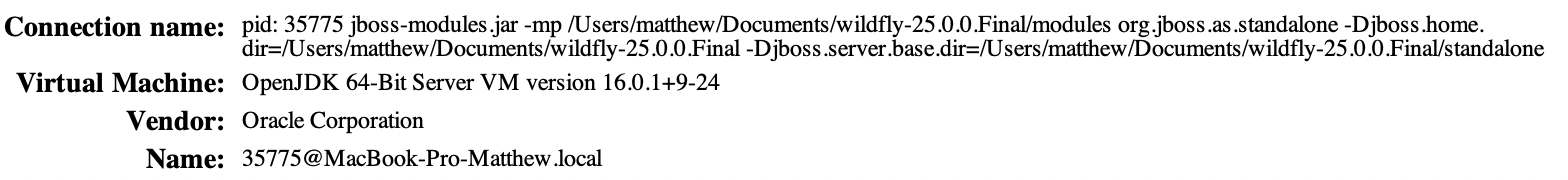
****

****

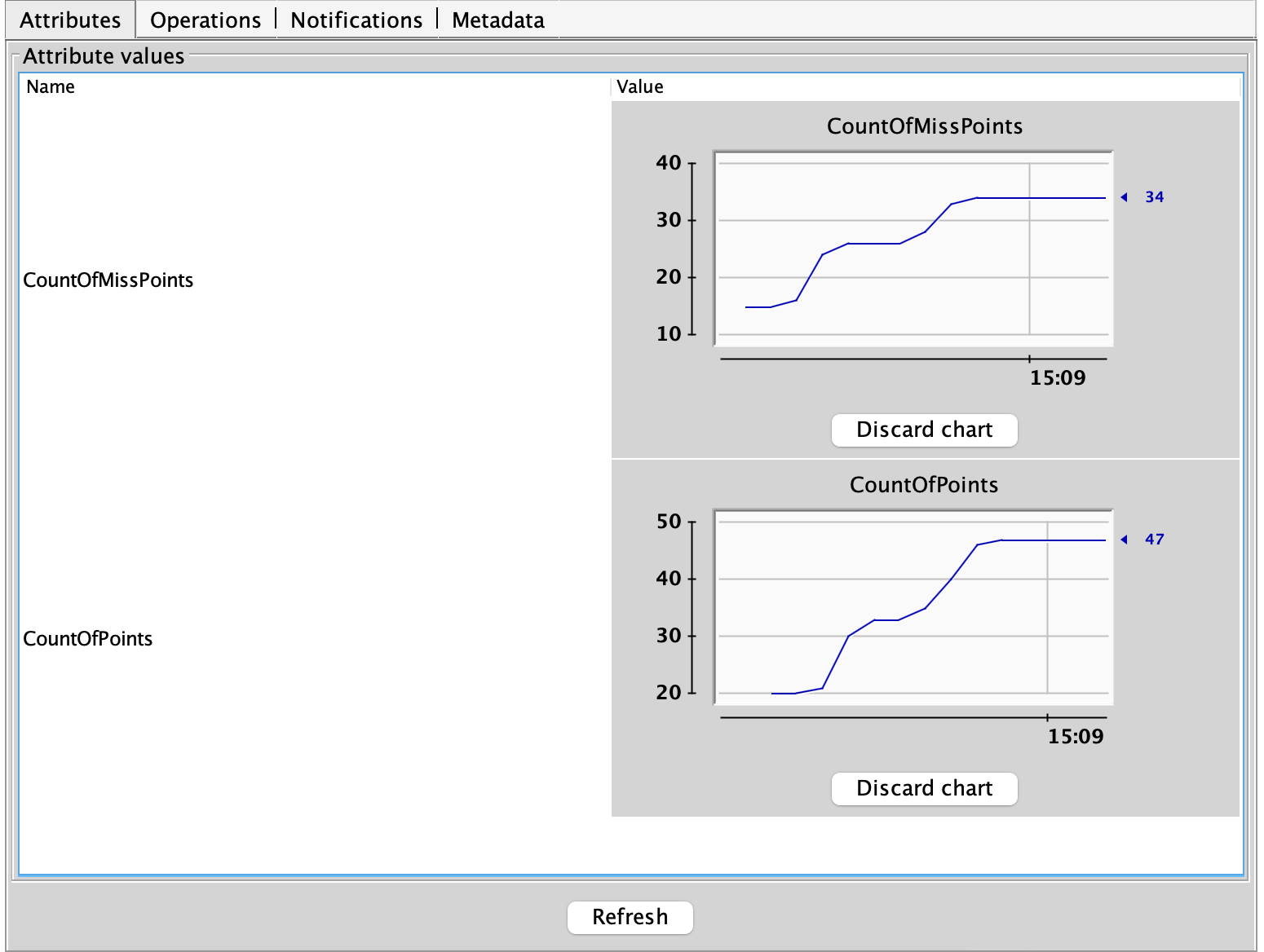
**SquareMBean**

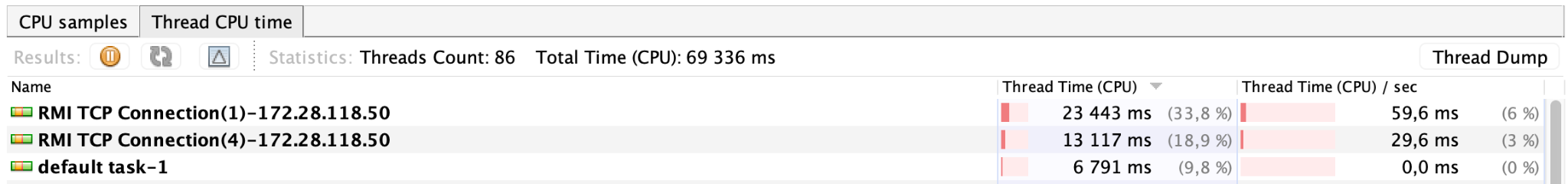
****

****

****

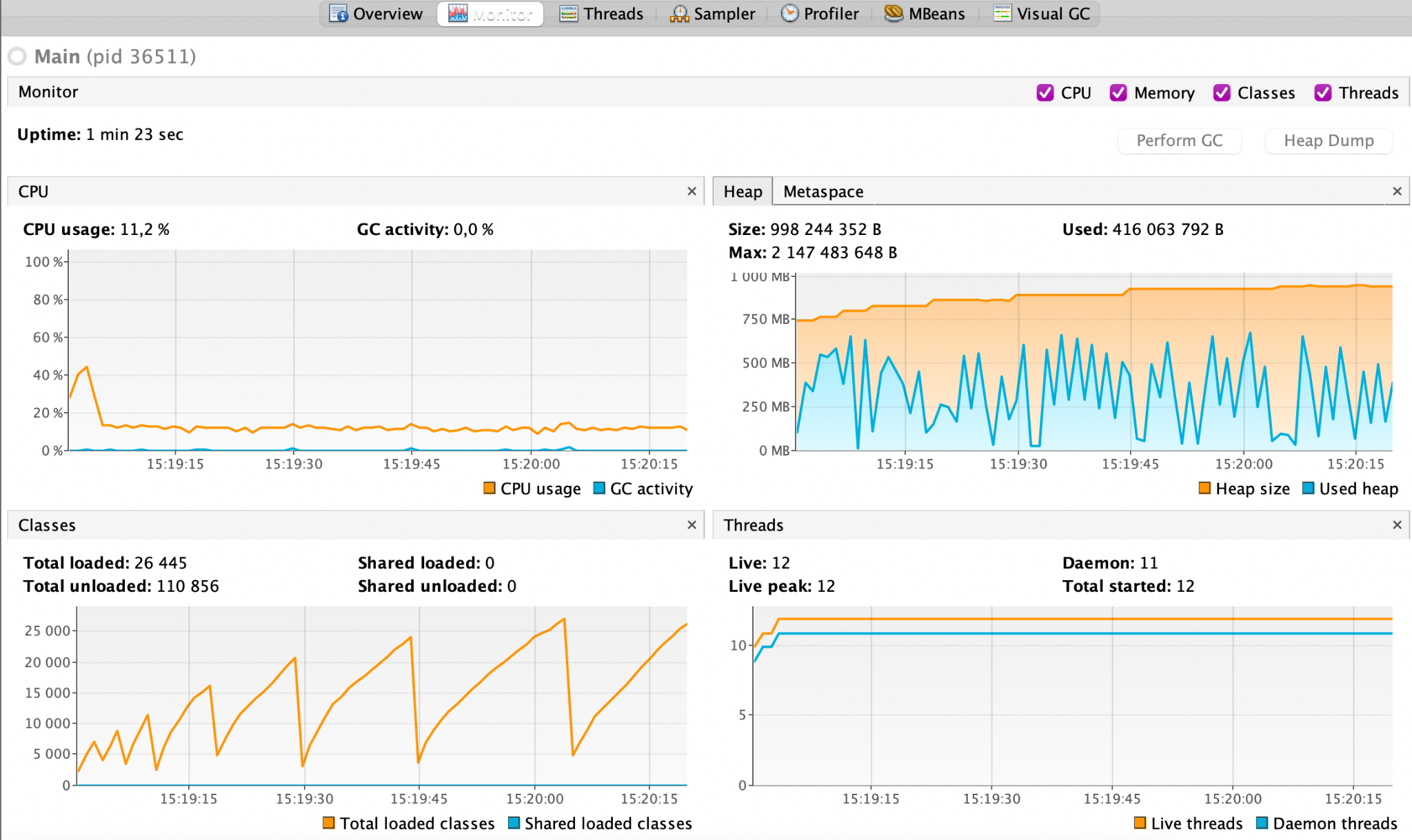
**VisualVM:**

****

****

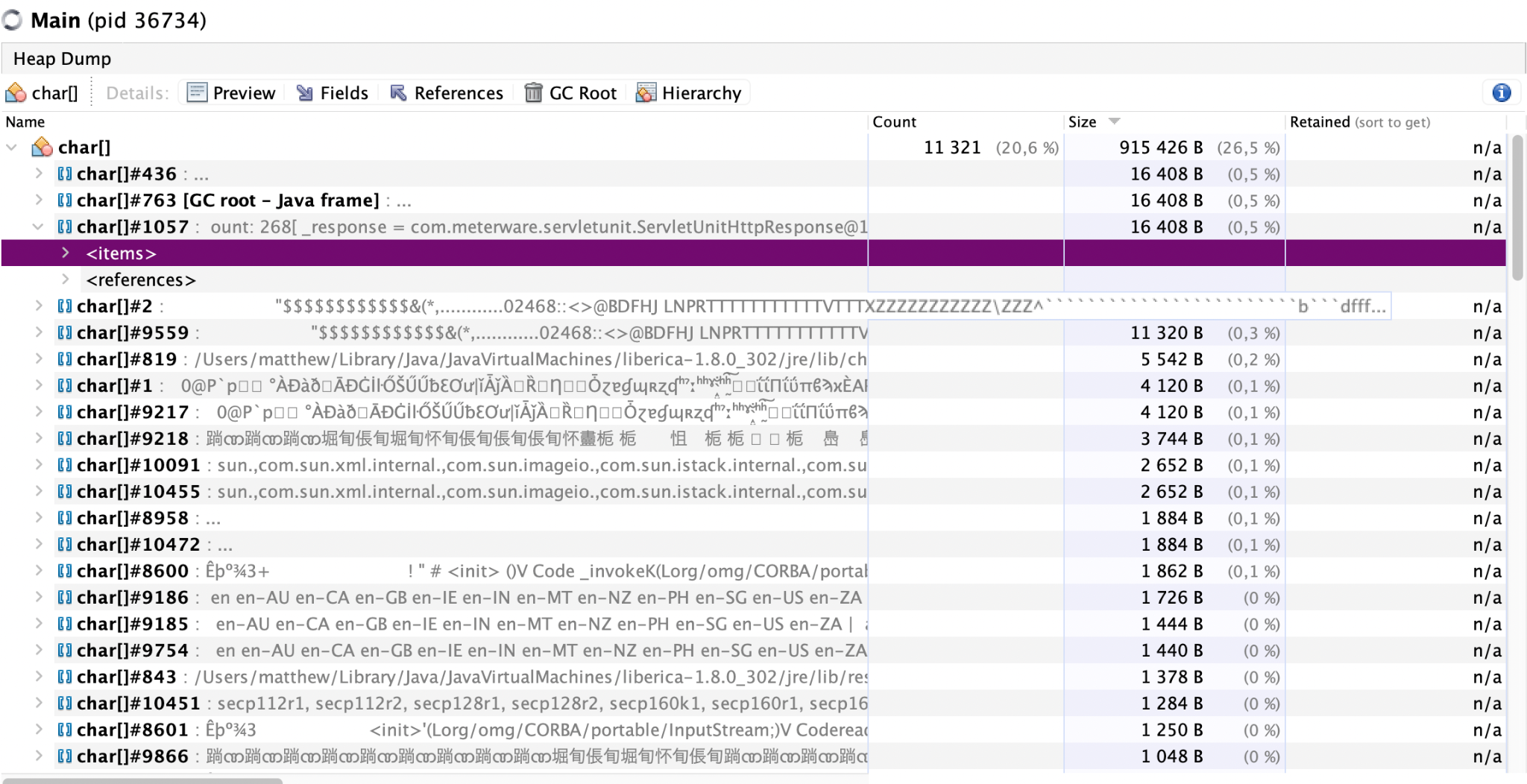
**Обнаружение и устранение проблемы:**

В утилите VisualVM во вкладке Monitor наблюдаем за использованием ресурсов. По графикам видно, что количество классов постоянно растет. Даже после отработки GC тенденция к росту остается. Размер кучи достаточно часто меняется. По графику Used Heap также можно заметить частый вызов GC, это негативно влияет на производительность.

****

Первая странность – постоянная рост количества массивов char, даже после отработки GC их максимальное количество растет.

****

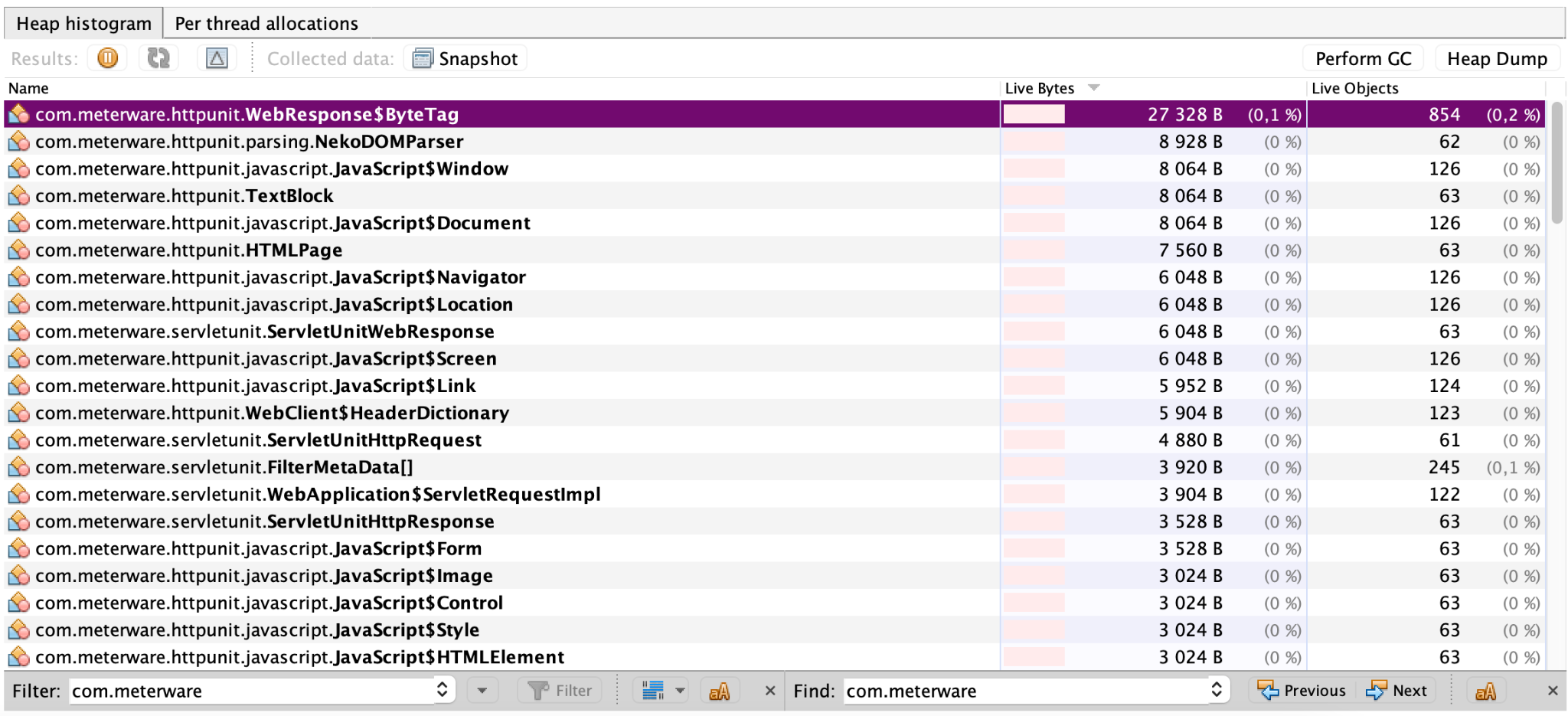
****

Отфильтруем по com.meterware. т.е. по нашему пакету. Видим, что класс

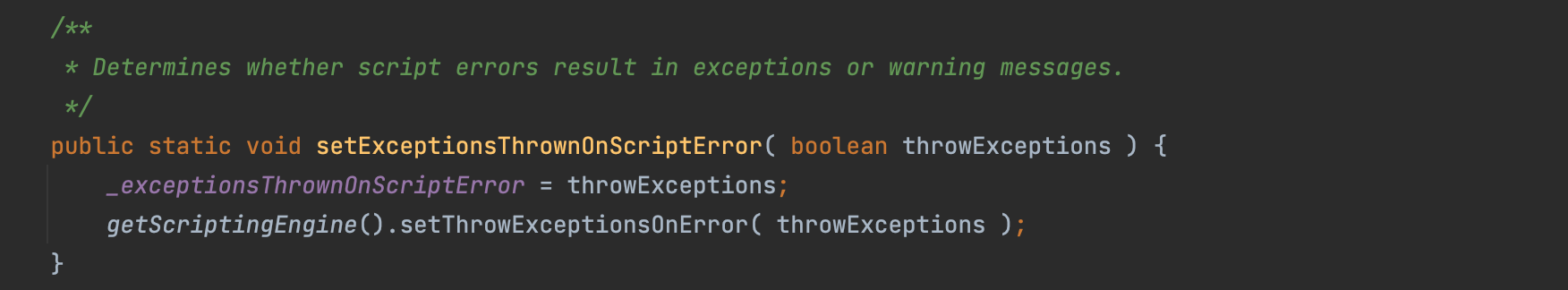
WebResponse (а точнее его часть) обладает таким же поведением. Постоянно

растёт и удаляется GC. Попробуем посмотреть, что происходит у нас в

программе.

****

Основная проблема:

Изначально в коде программы лежит данная строчка:

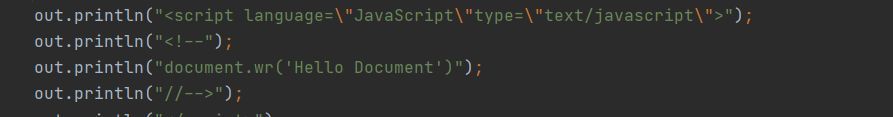


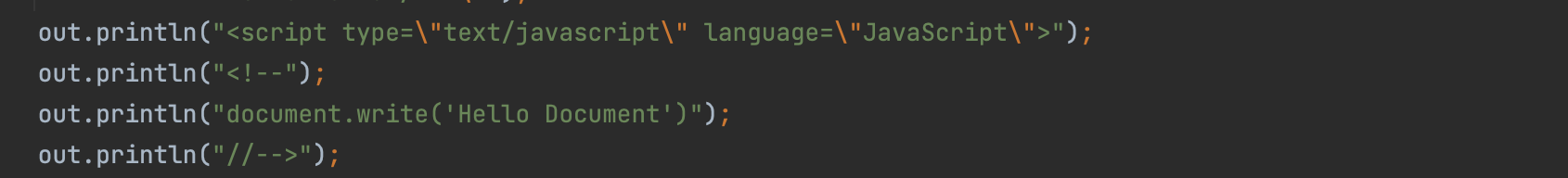
Она определяет, приводят ли ошибки скрипта к исключениям или сообщениям об ошибке и записывает их в ArrayList:



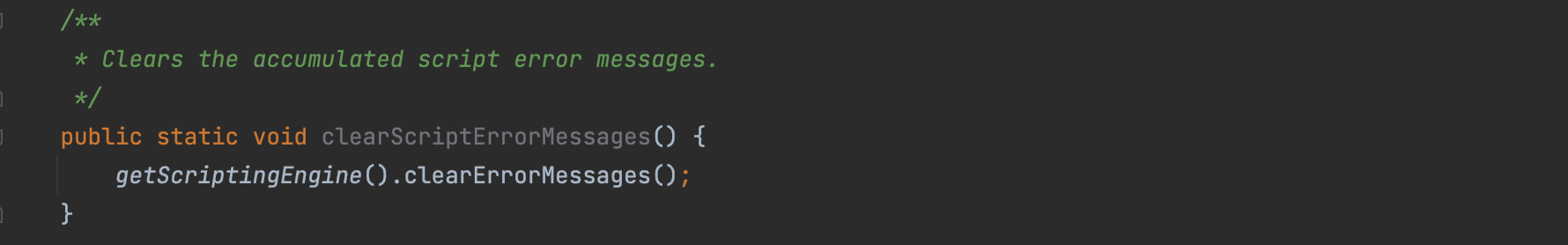
Далее при изменении аргумента на true при запуске программы начали вылетать exception’ы (которые ранее копились в arraylist’е).

В сервлете находится строчка document.wr(‘Hello Document’):



При парсинге вылетает исключение, так как нет функции document.wr(), но есть функция document.write()

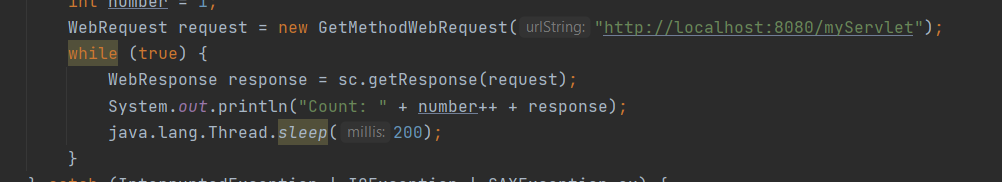
Соответственно при каждом запросе возникала ошибка и добавлялась в список ошибок. Также в классе JavaScript есть метод, который очищает данный лист, но он нигде не вызывается.



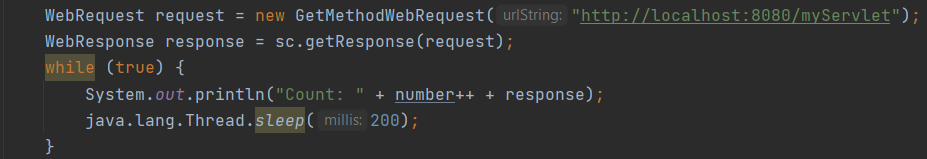
Поэтому есть два решения этой проблемы:

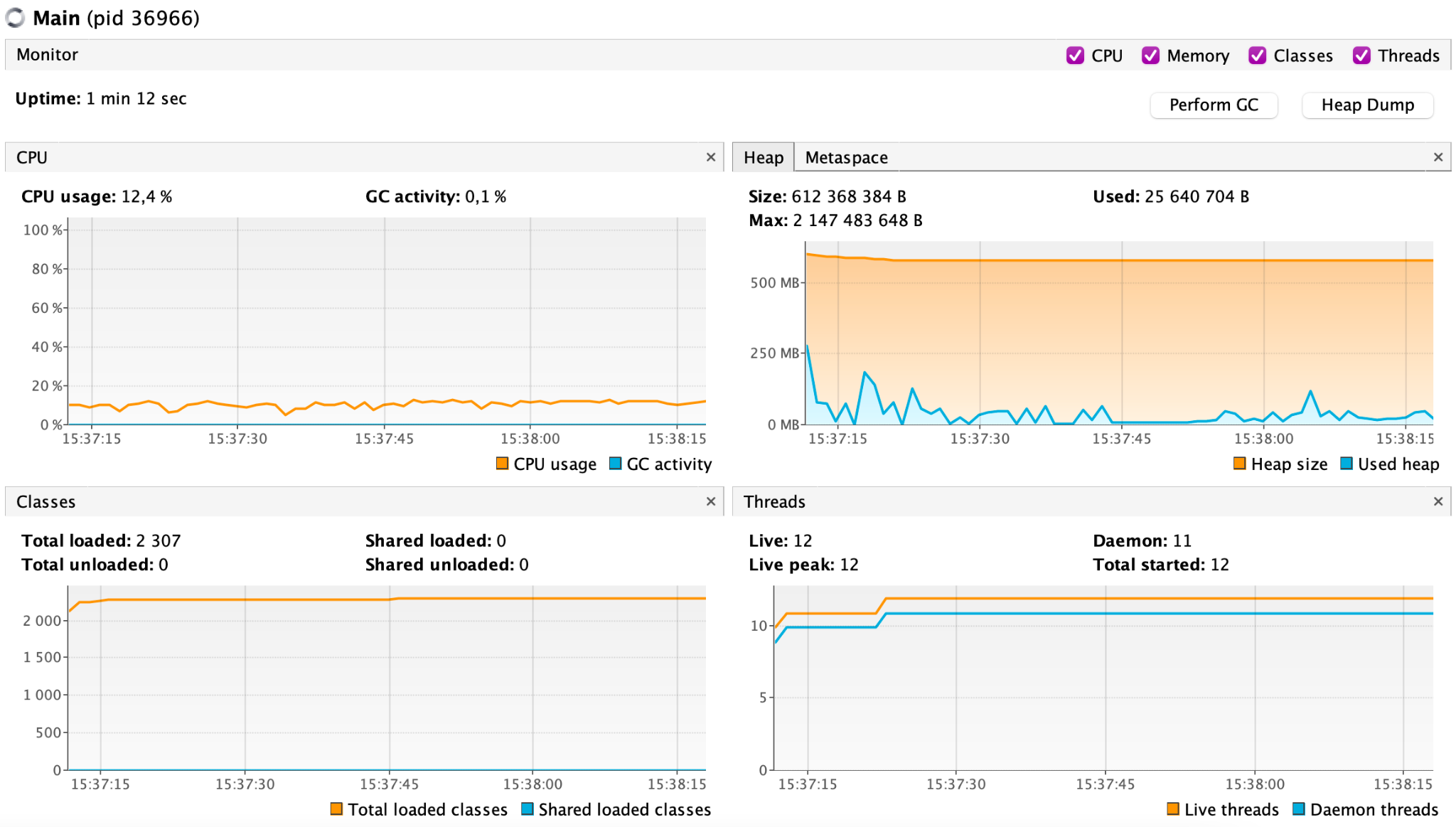
1. Перед добавлением ошибки в ArrayList проверять его размер и при определенном значении вызывать функцию clearScriptErrorMessages
2. Исправить функцию document.wr() на document.write, тогда ошибки не будут возникать и записываться в ArrayList.

Также существует еще такой недочет:



Видно, что в данном куске кода создается WebRequest и в бесконечном цикле постоянно создается WebResponse на один и тот же request. Поскольку ответ всегда будет один и тот же, то не стоит каждый раз создавать объекты WebResponse, достаточно один раз получить ответ и его выводить (соответственно увеличивая переменную number).





<https://youtube.com/clip/Ugkx-3SpxgTBRzDJcq117GnA1KchDwmtbi4S>

**Вывод:** В ходе данной лабораторной работы мы познакомились с профилированием Java приложений , поработали с технологией JMX , а именно создали свои MBean’ы для сбора статистики. В JMX Agent’e (JConsole) сняли показания написанных бинов и посмотрели на нагрузку , создаваемую приложением. С помощью VisualVM были выявлены и устранены проблемы в предоставленной программе.