Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Факультет программной инженерии и компьютерной техники

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ 4

ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

ВАРИАНТ 53999

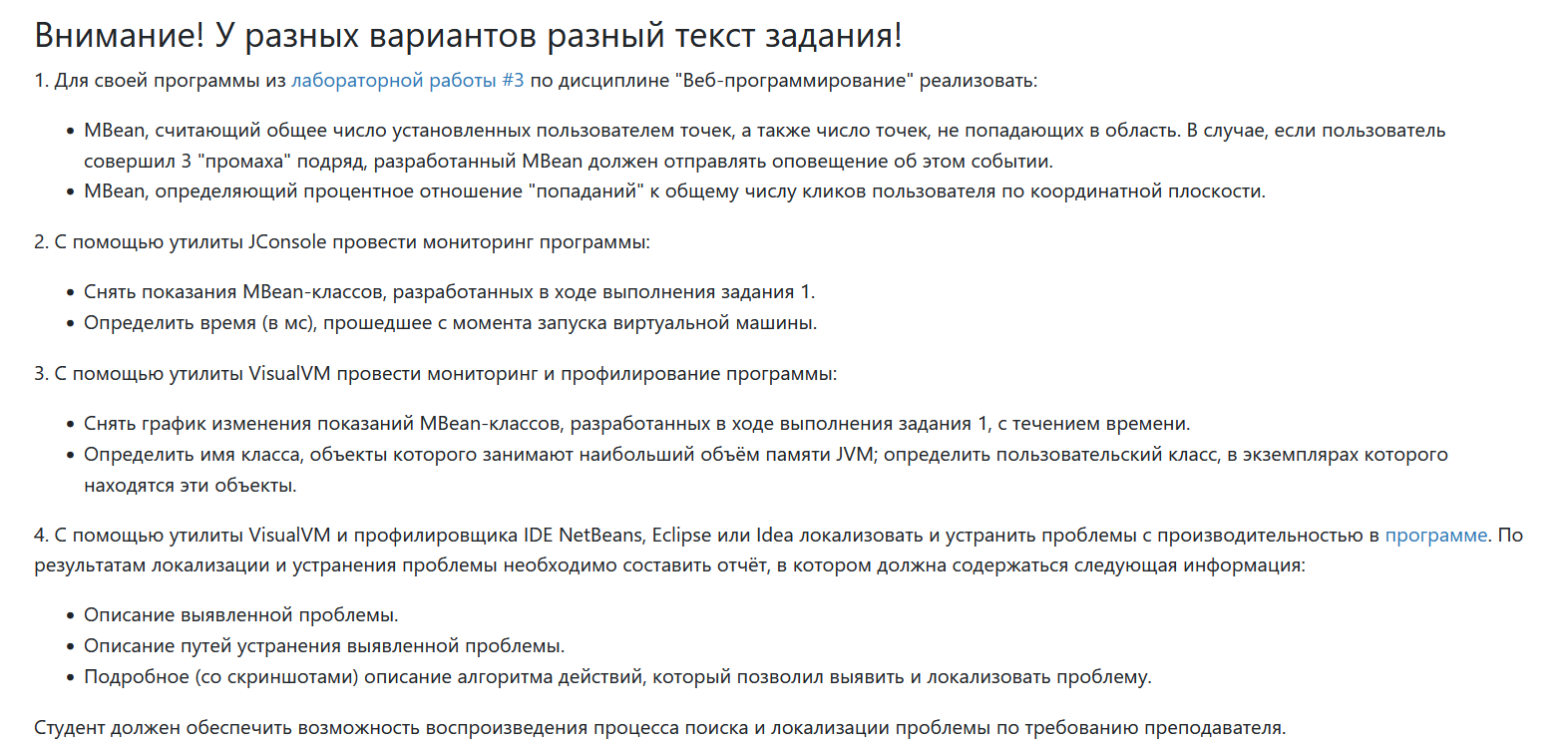
­

Студент: Пышкин Никита Сергеевич, P3213

Преподаватель: Райла Мартин

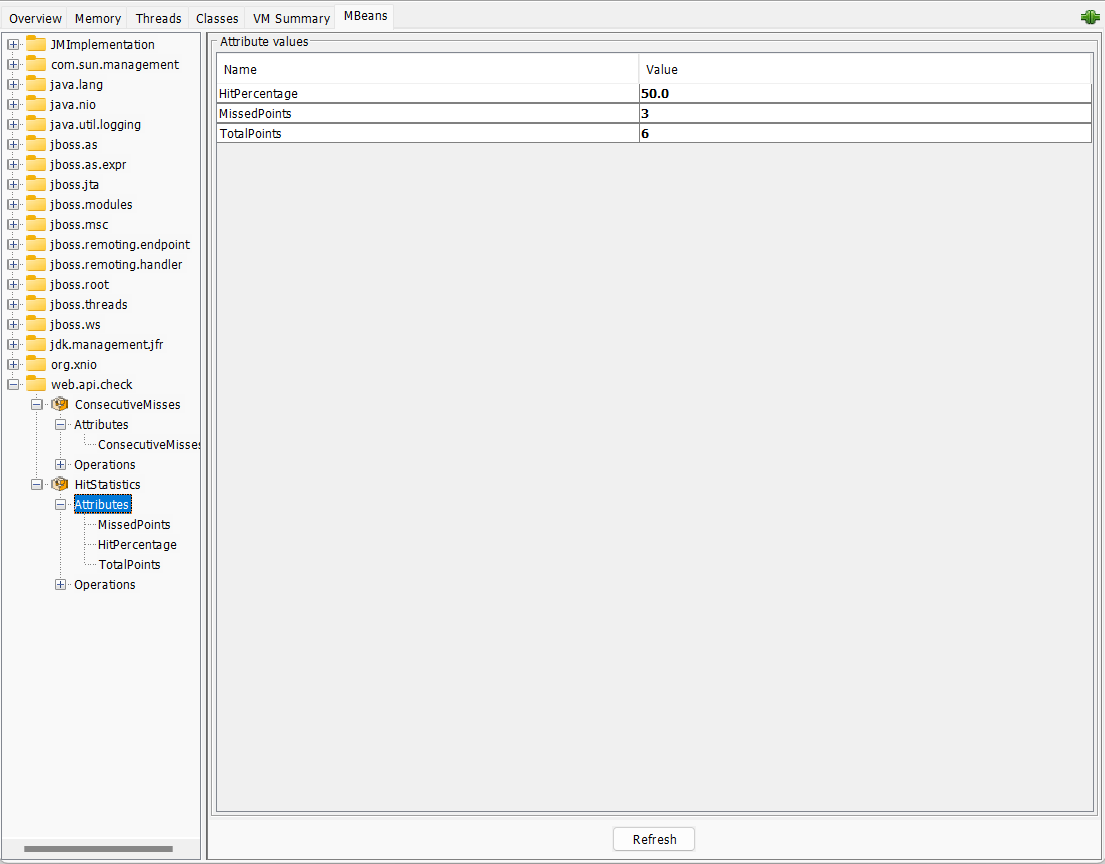
Санкт Петербург 2025

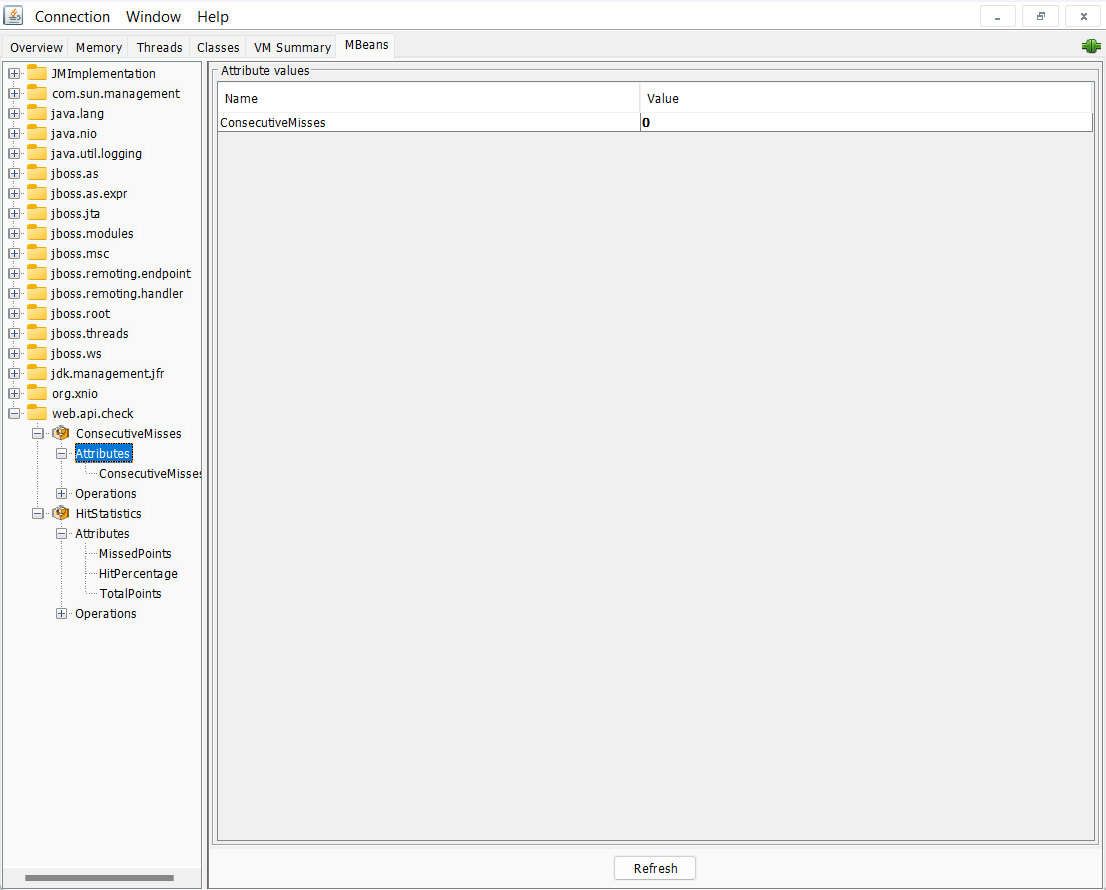
# Текст задания



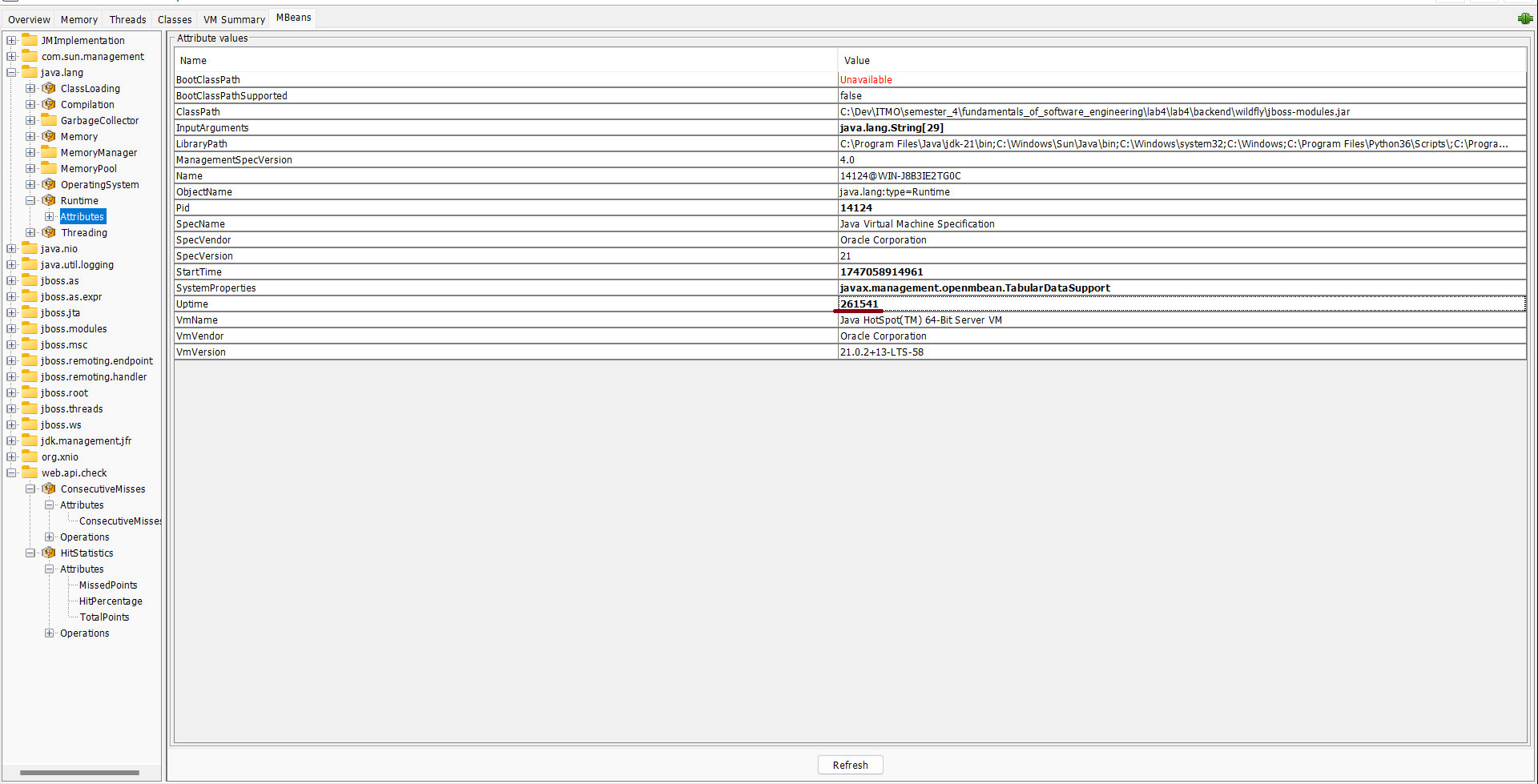
# Мониторинг программы JConsole

Показания Bean-ов:



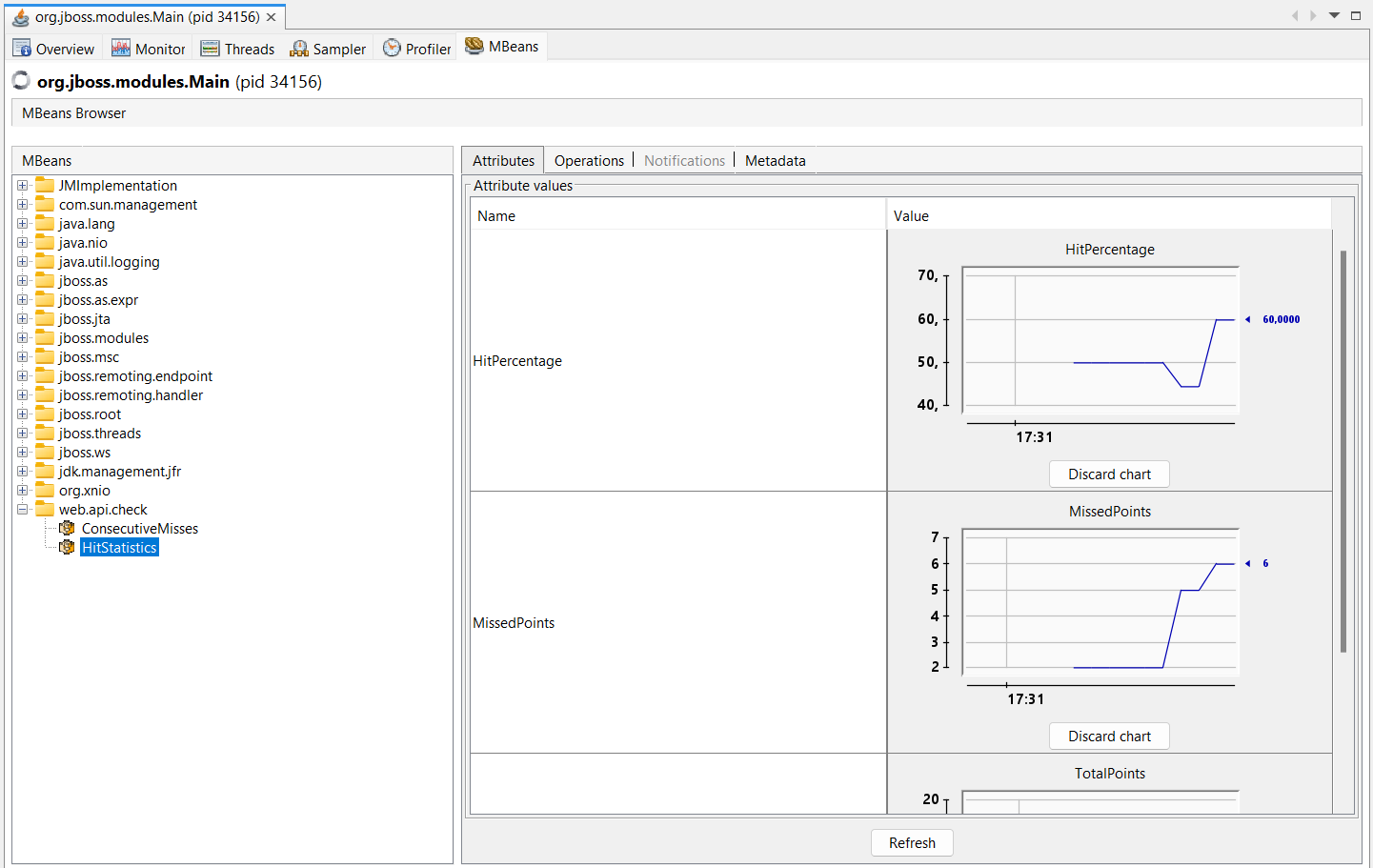
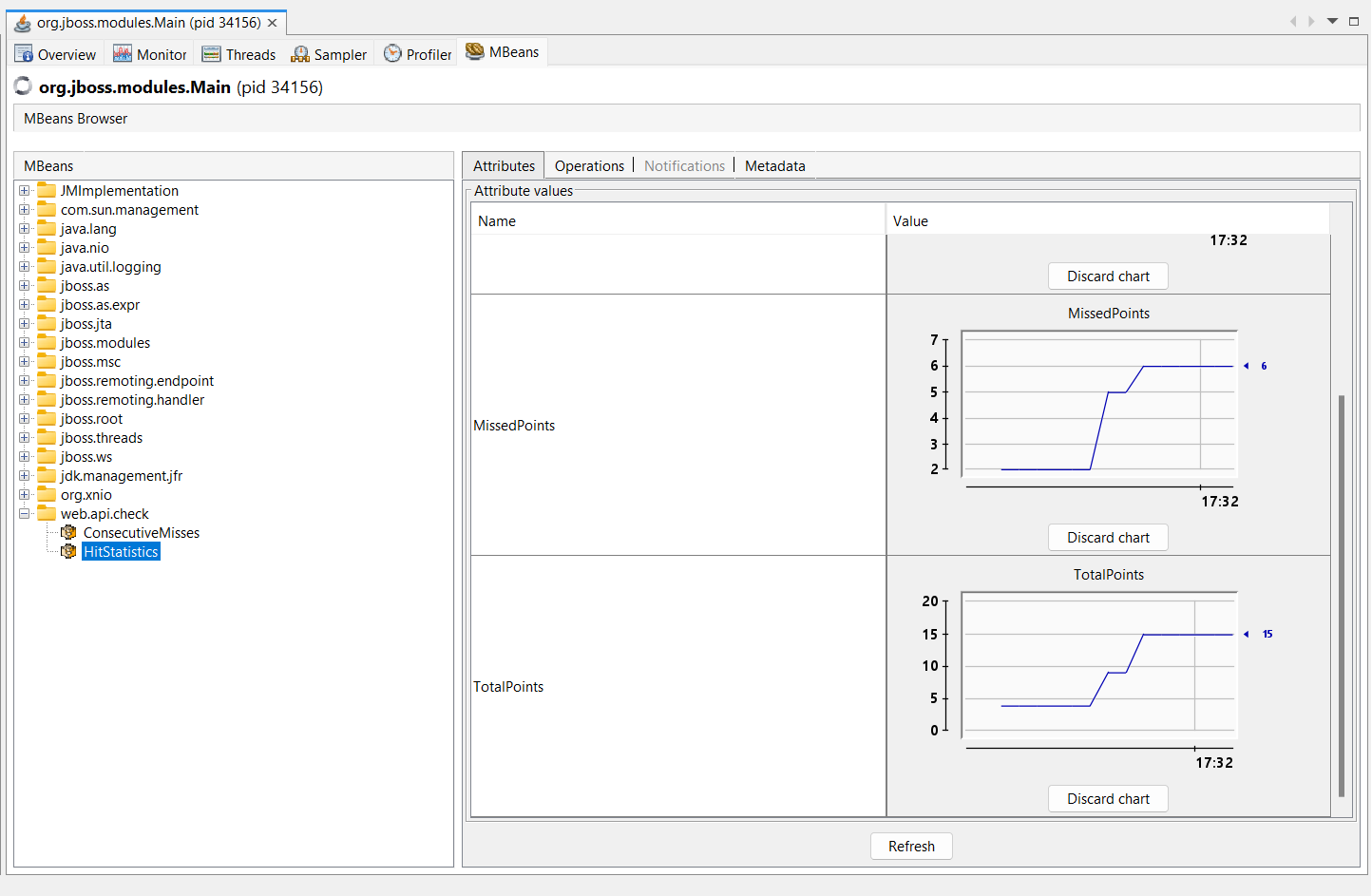


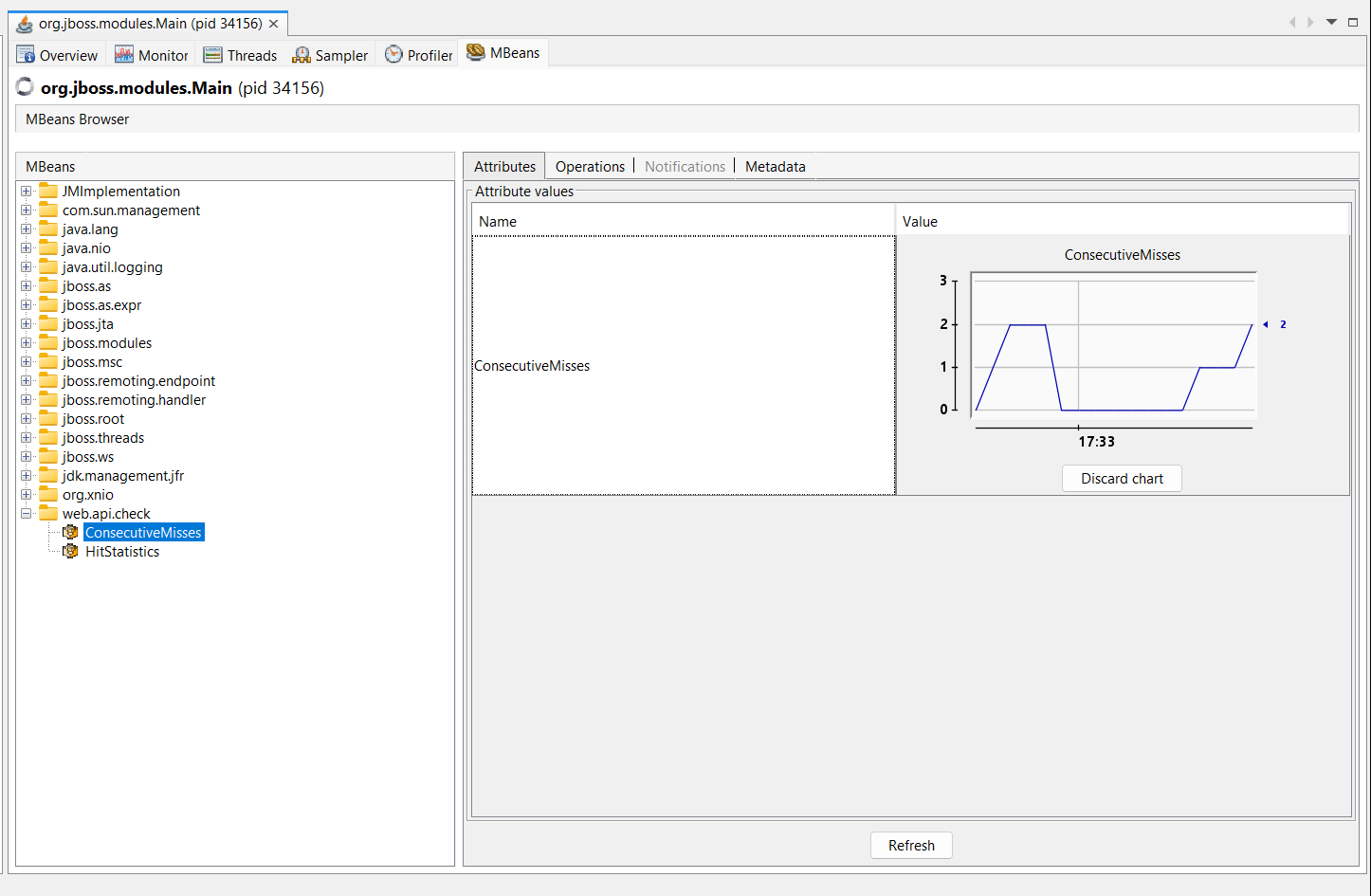
Uptime виртуальной машины:

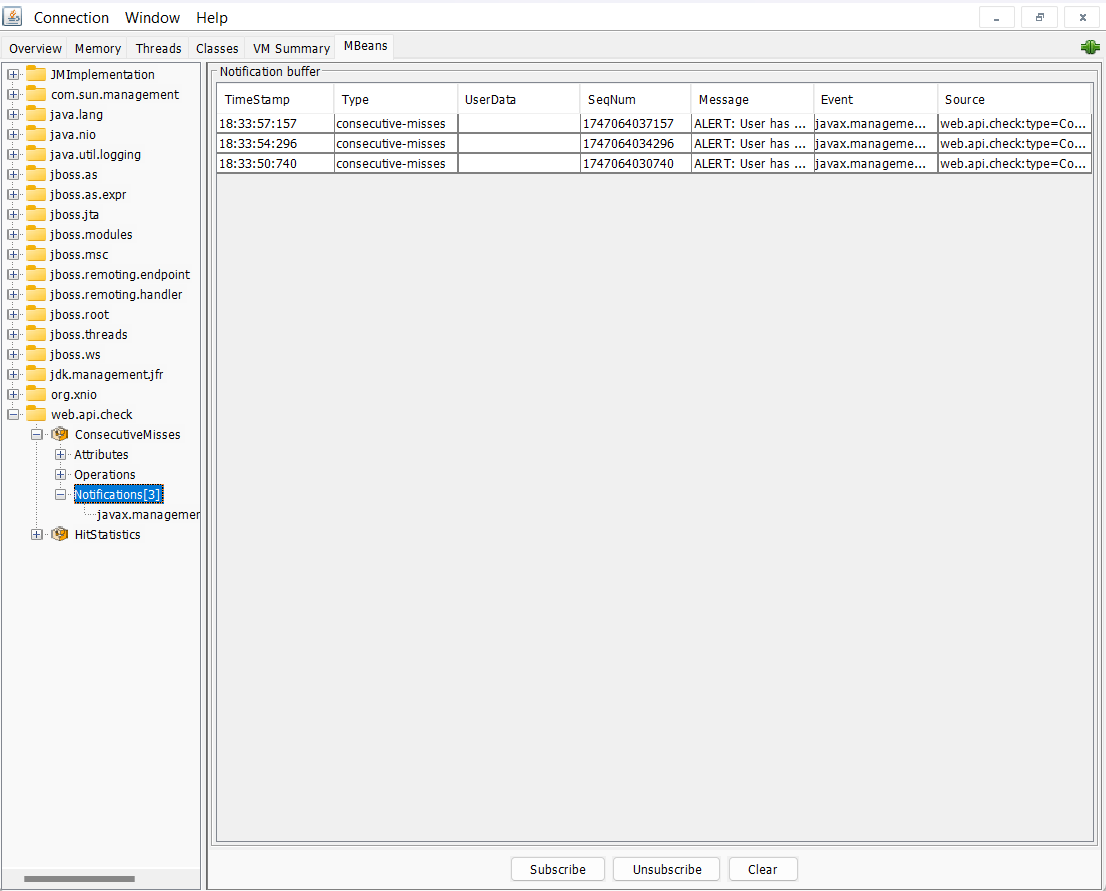


# Мониторинг программы VisualVM

Показания Bean-ов:

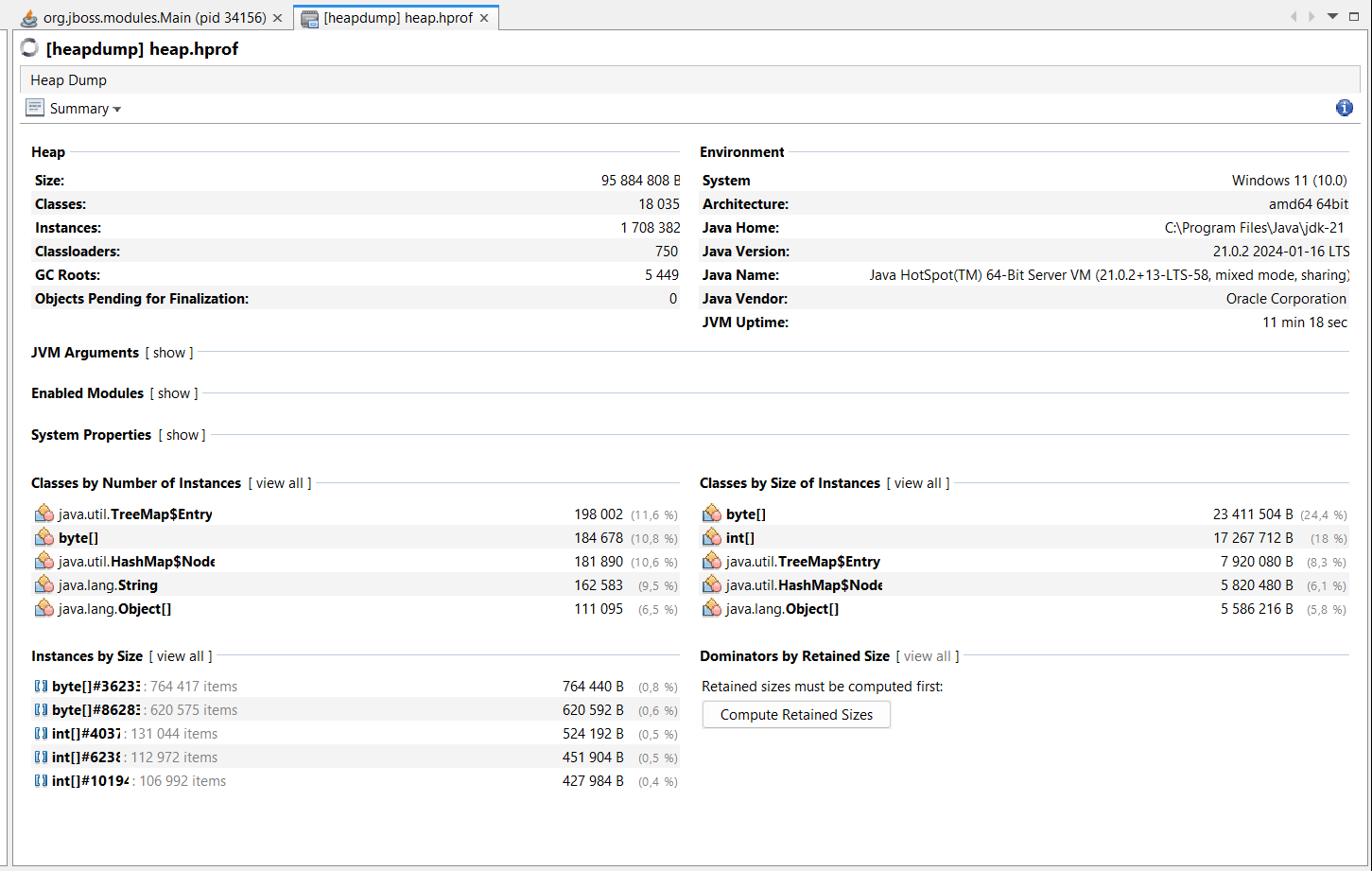




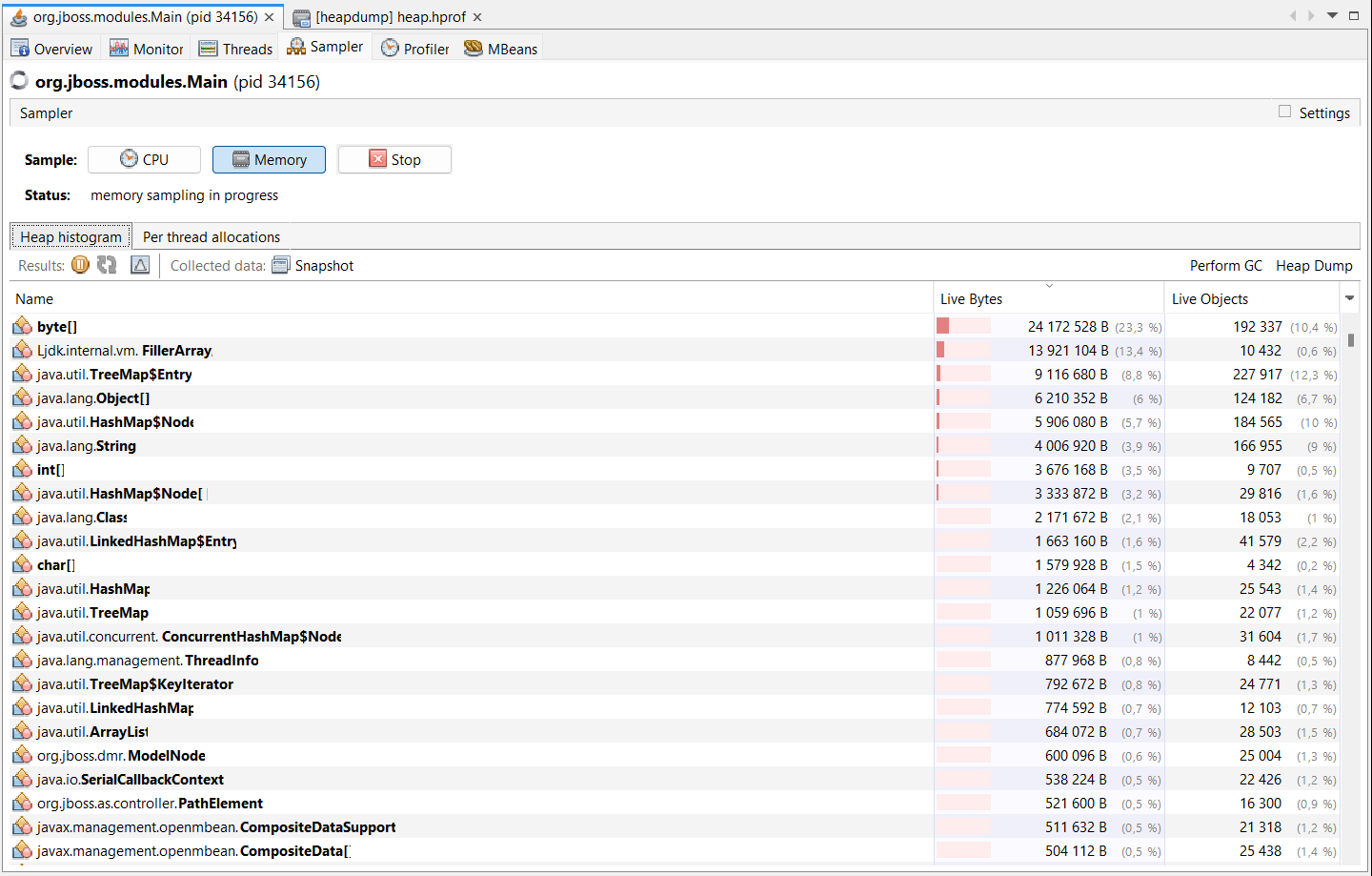
Смотрим дамп:

Либо так

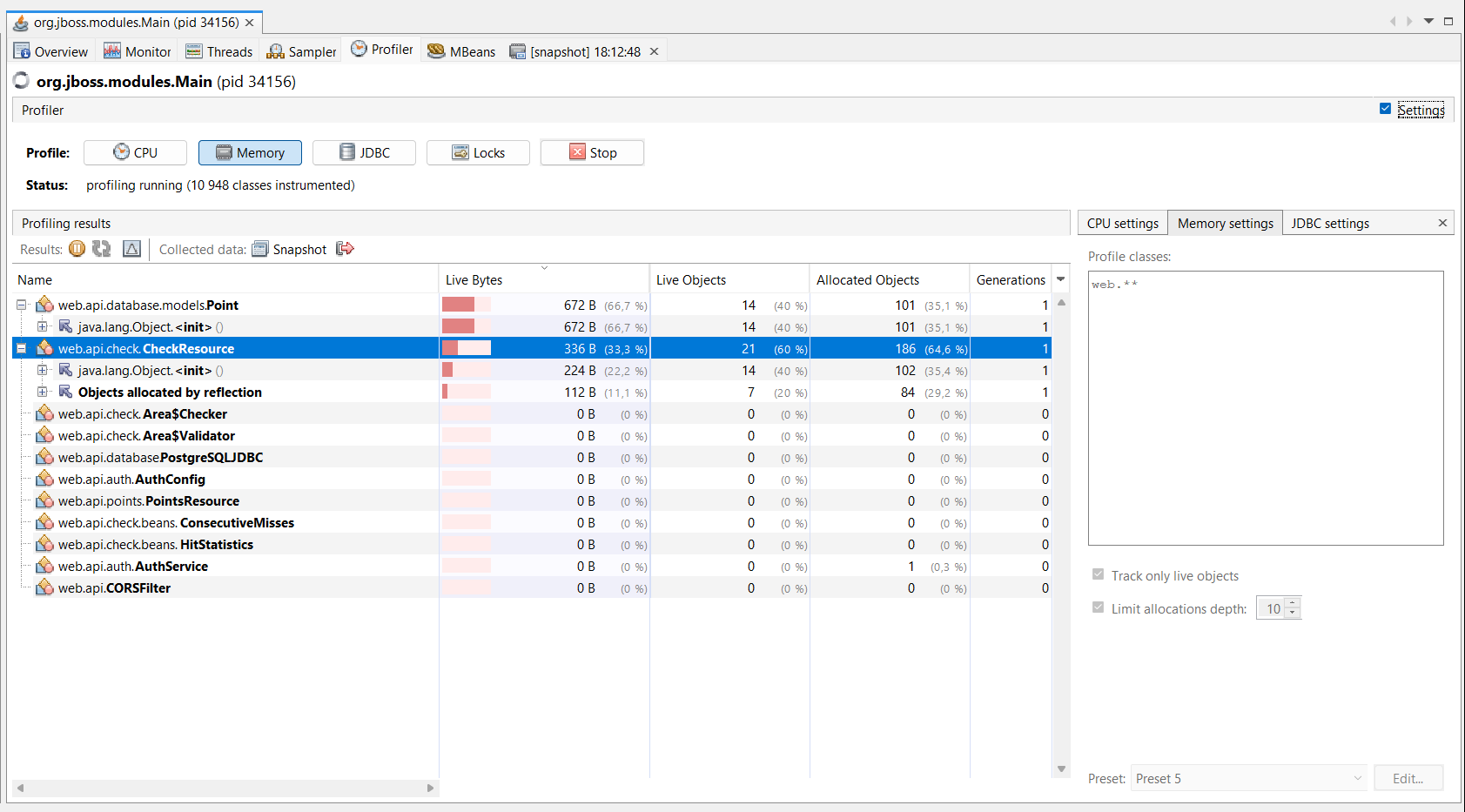
jmap -dump:format=b,file=heap.hprof 34156

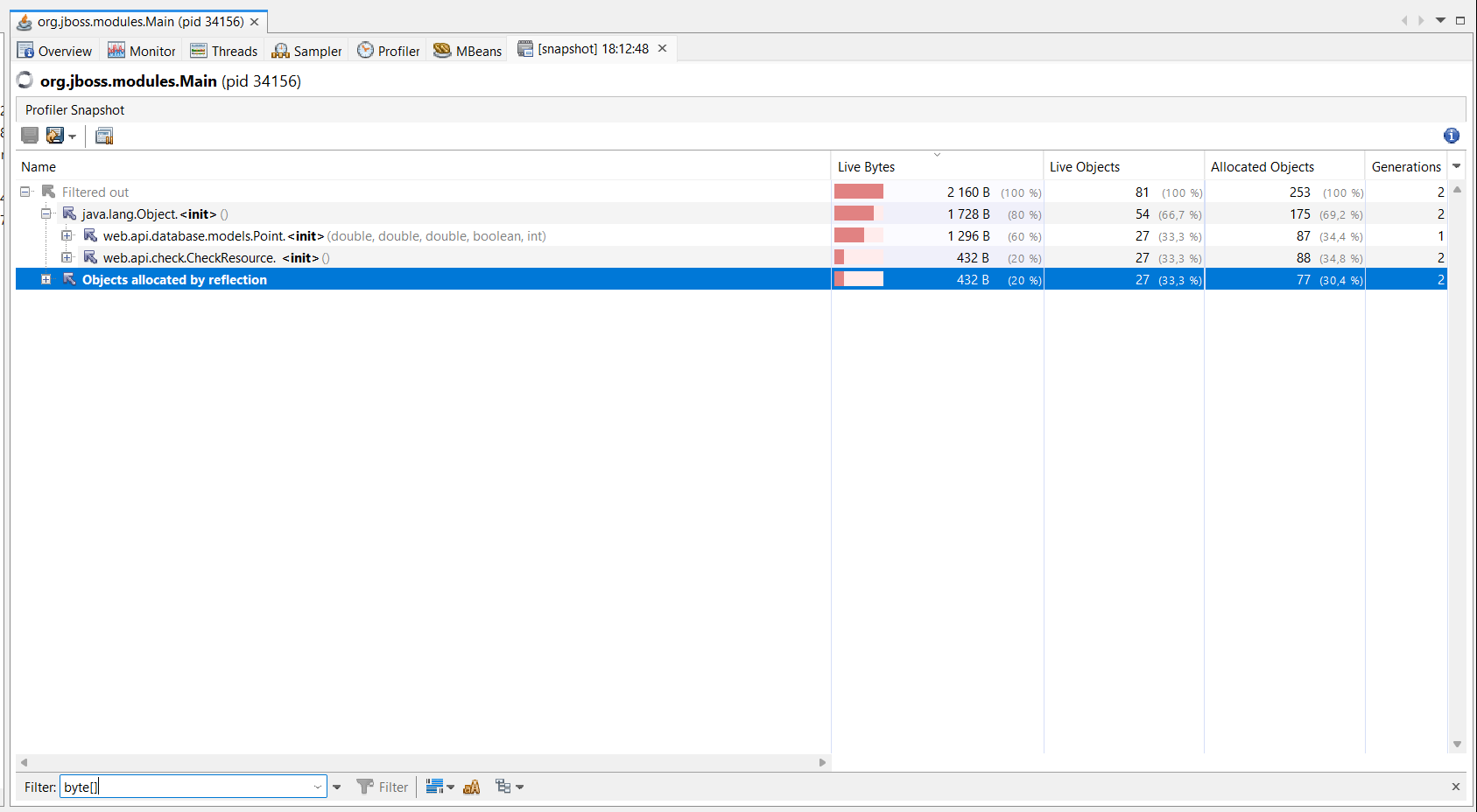


Либо так:



Ищем в каких пользовательских классах есть byte[]





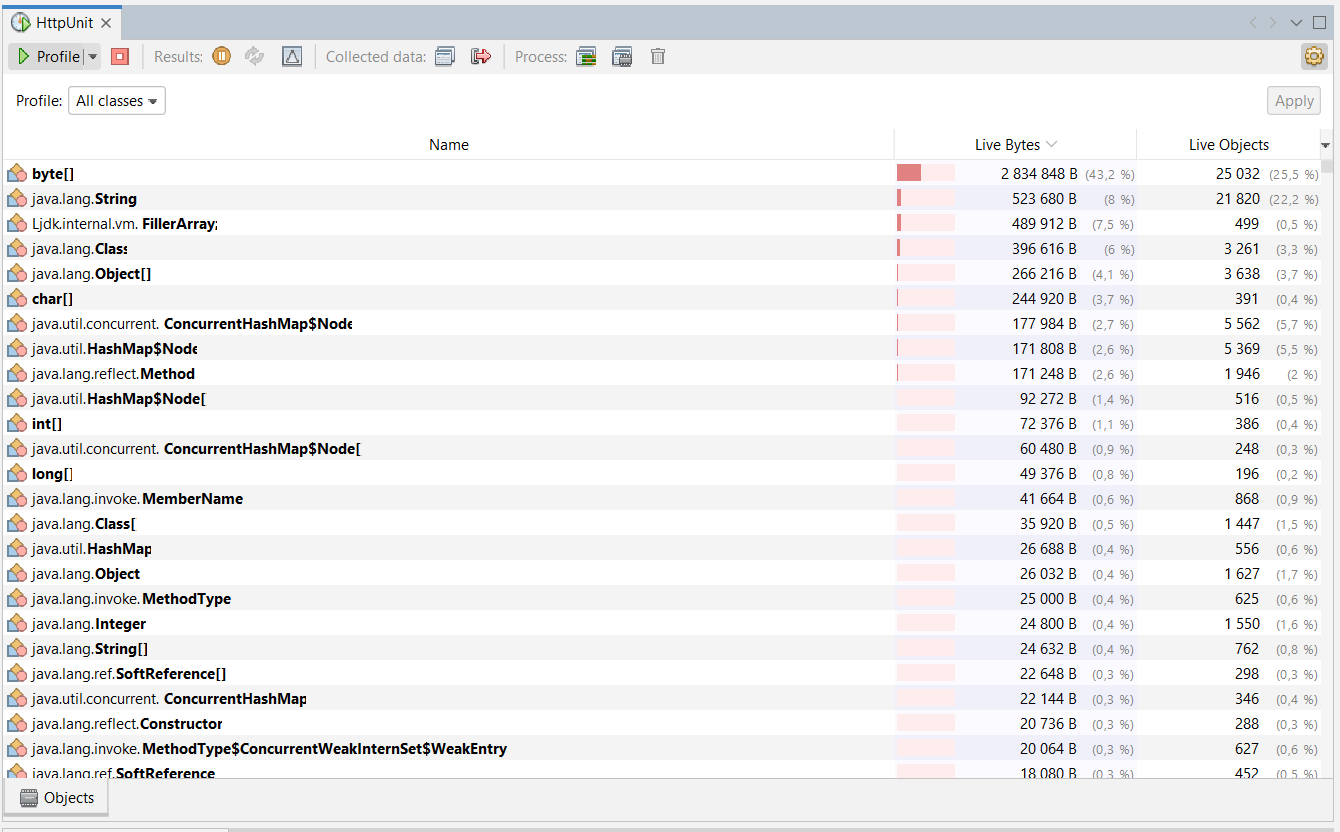
# Нахождение и устранение проблемы

Конфигурация G1:

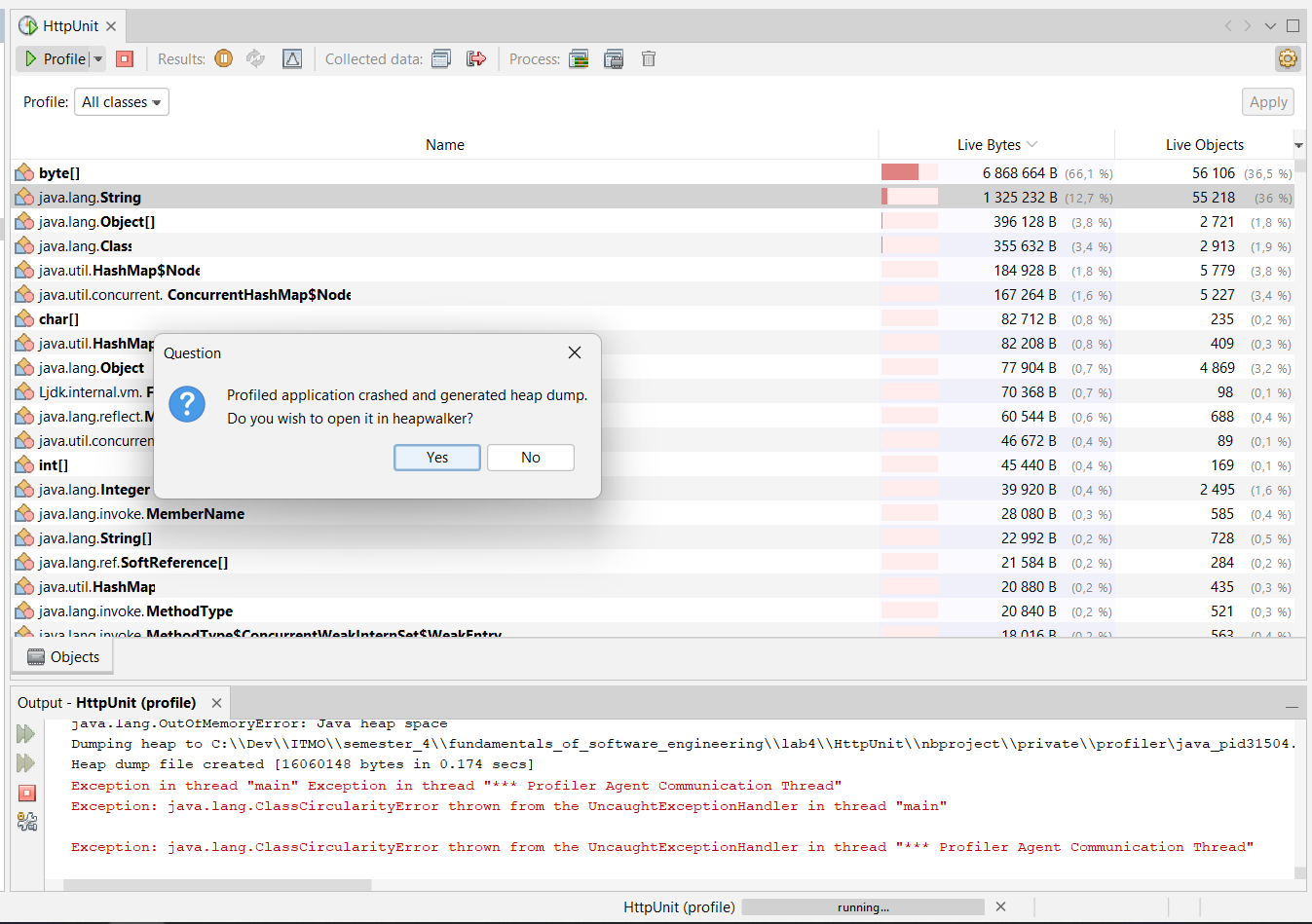
-Xlog:gc\*=debug:file=gc.log:time,uptime,tid:filecount=5,filesize=10m

-XX:+UseG1GC

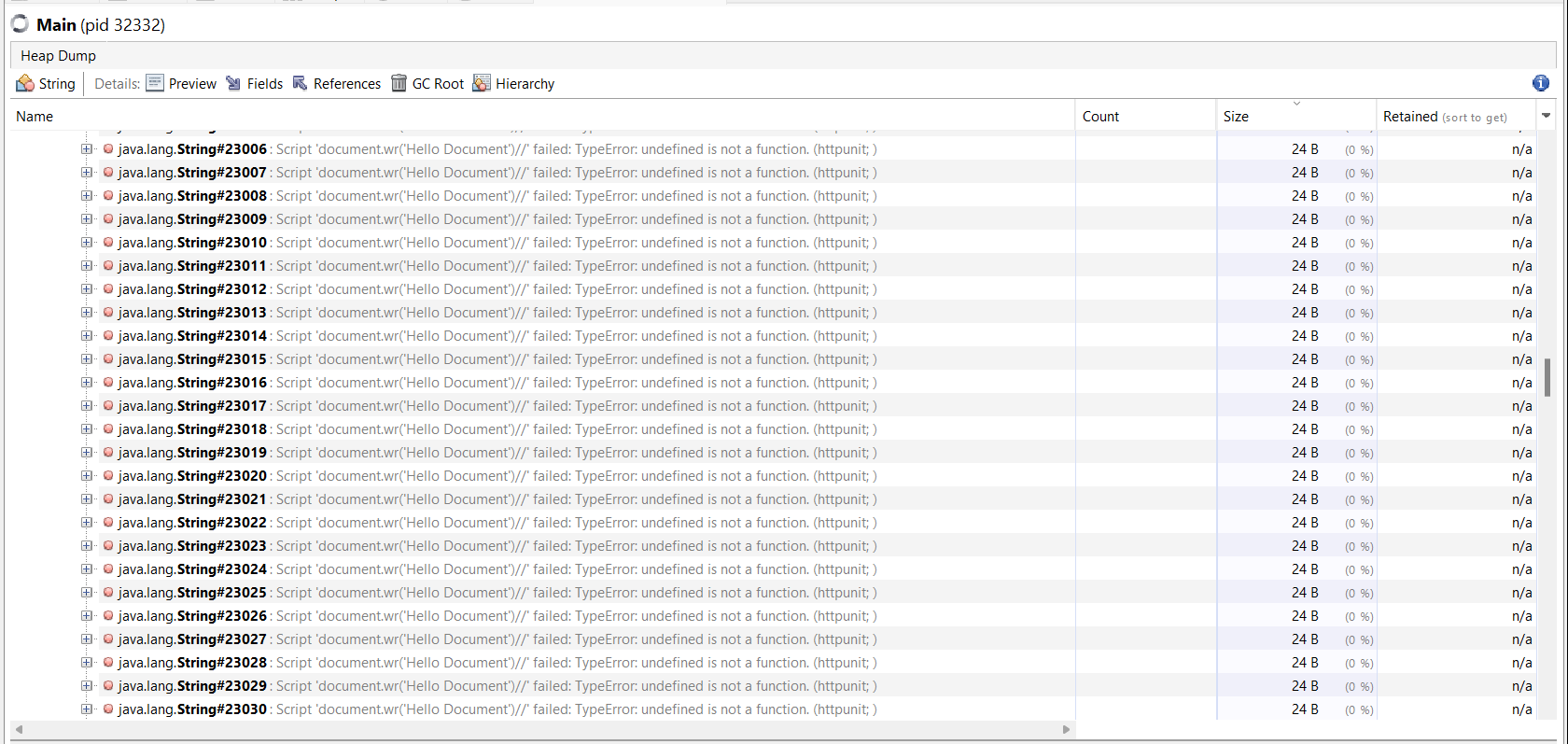
Метрики в начале:

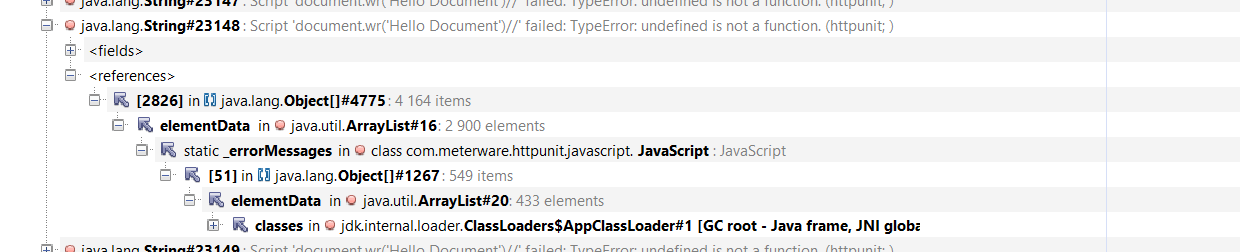


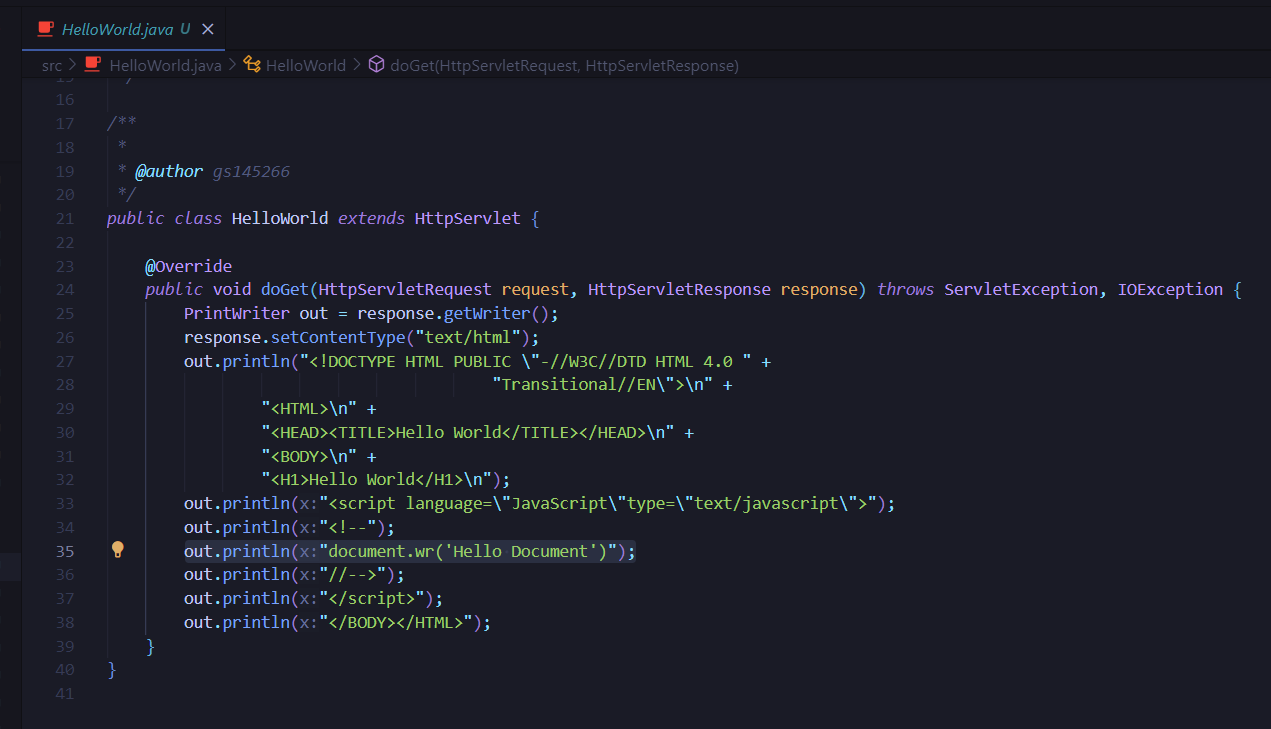
Метрики в конце:

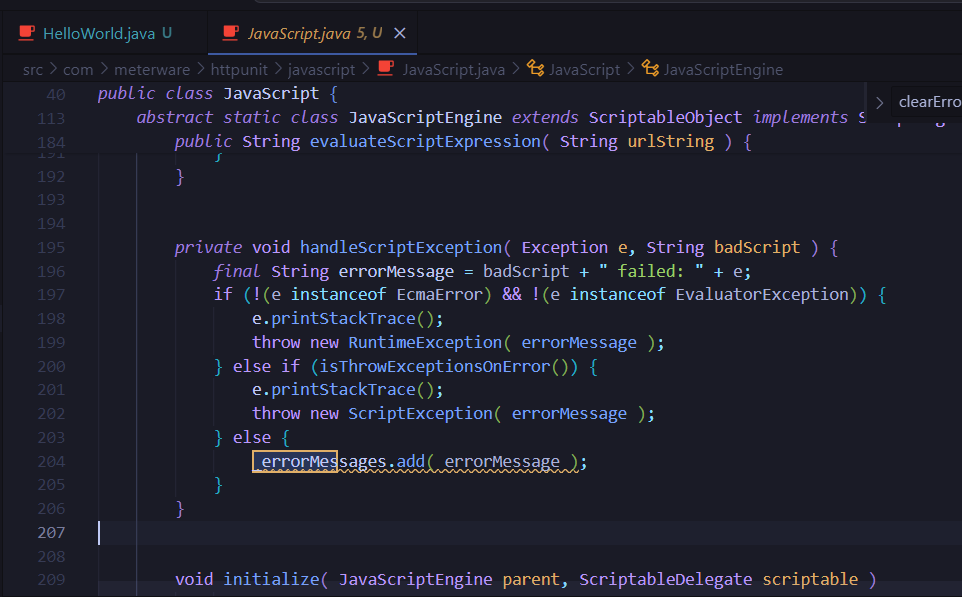


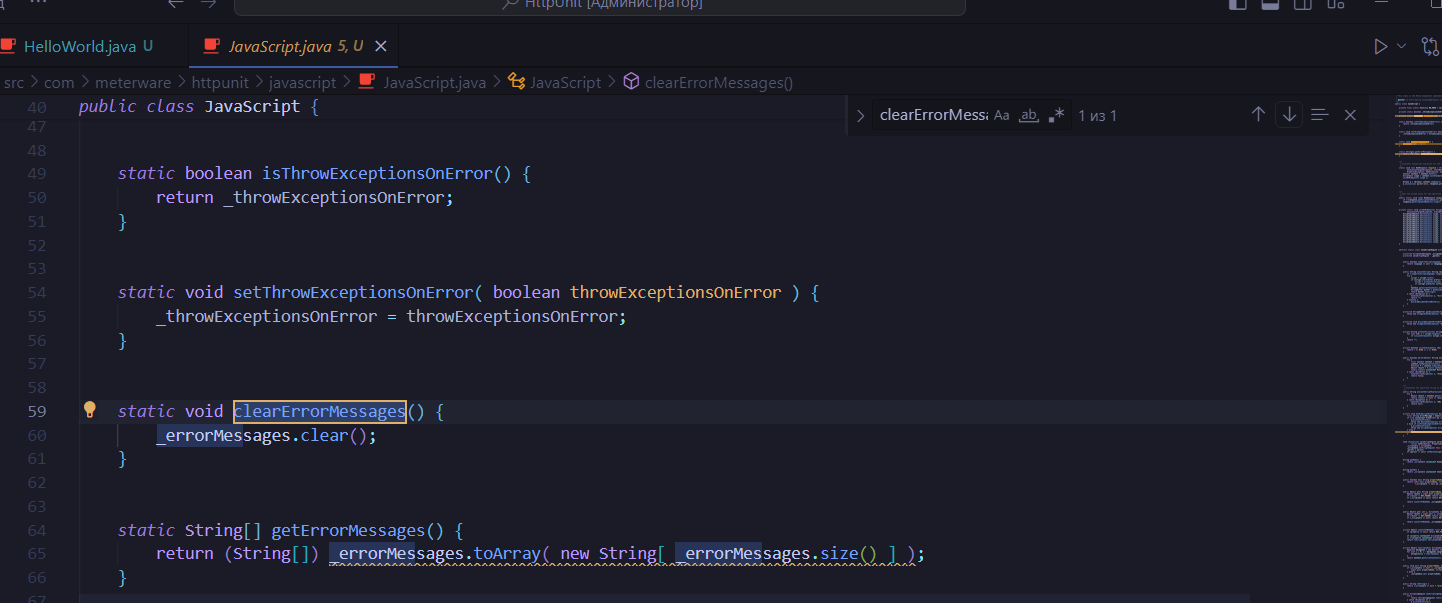
Как можно заметить основной причиной утечки памяти являются строки. Если посмотреть на объекты строк, то можно увидеть кучу следующих ошибок:











В итоге мы получаем, что document.wr('Hello Document') генерируем бесконечное количество ошибок, которые не очищаются.

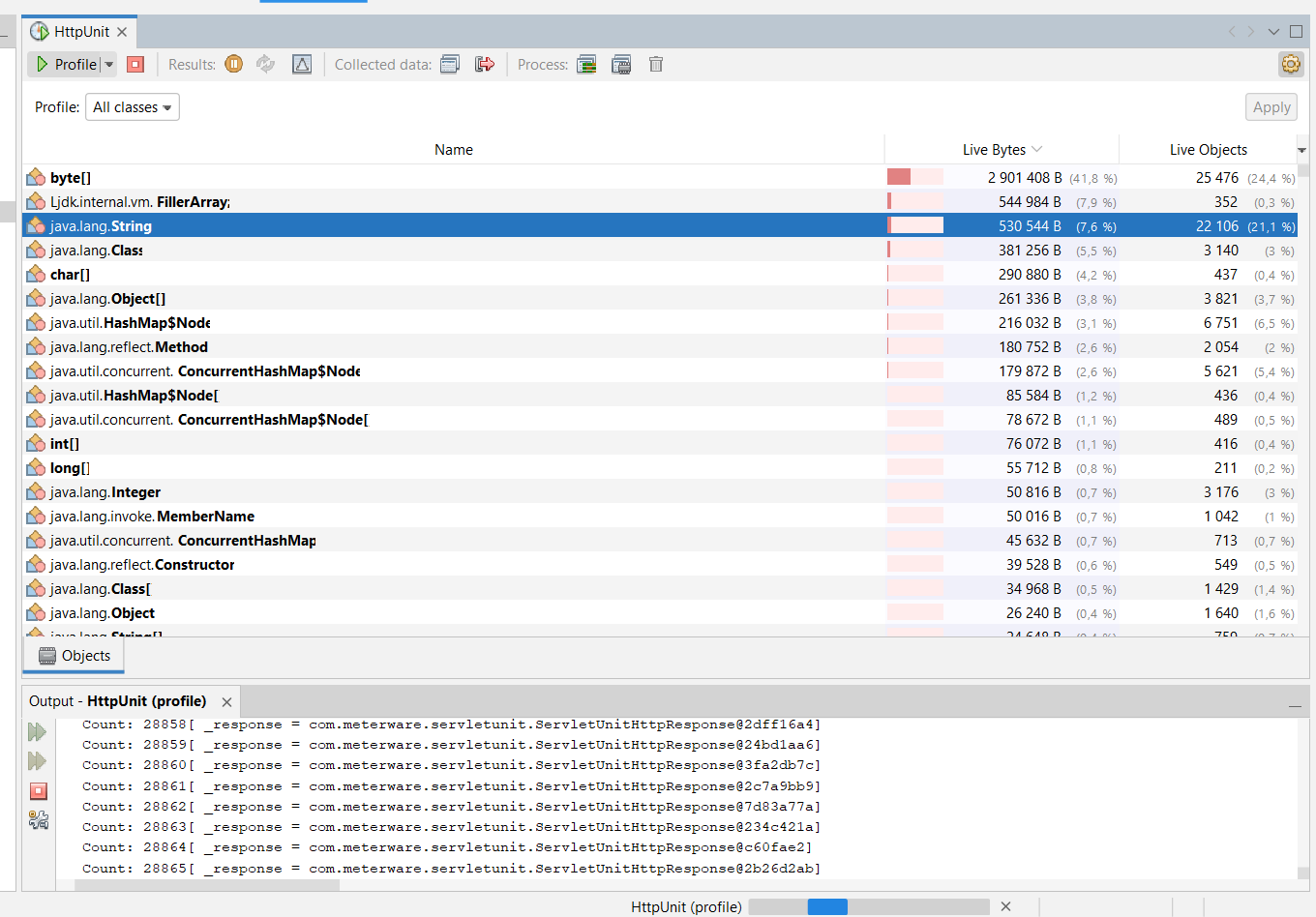
Возможные пути исправления:

1) Исправить JavaScript код, который спамит ошибками (уязвимости не исправит, но память нам сохранит)

2) Очищать массив ошибок, хранить в нем только N ошибок и т.п.

Так как в приоритете, конечно, пофиксить уязвимость, добавляем очистку ошибок: 

Ошибка устранена:



# То же самое, но в JProfiler

