Университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Методы и средства программной инженерии»

**Отчет**

По лабораторной работе №4

Вариант 123

Выполнили:

*Барабанщиков А.Д.*

*Кочнев Р.Д.*

Преподаватель:

*Абузов Я.А.*

Санкт-Петербург, 2023 г.

Задание:

1. Для своей программы из [лабораторной работы #3](https://se.ifmo.ru/courses/web#lab3) по дисциплине "Веб-программирование" реализовать:

* MBean, считающий общее число установленных пользователем точек, а также число точек, 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 наименование и версию JVM, поставщика виртуальной машины Java и номер её сборки.

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

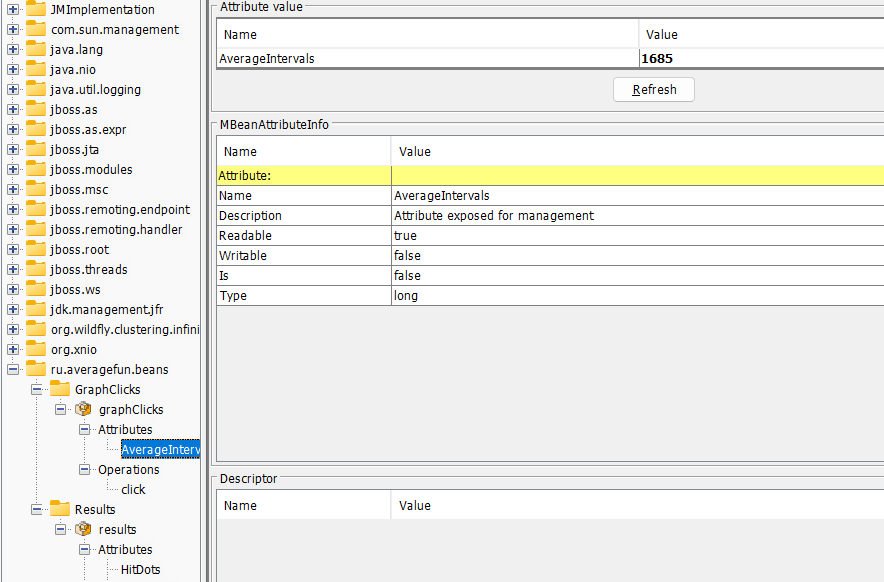
* Снять график изменения показаний MBean-классов, разработанных в ходе выполнения задания 1, с течением времени.
* Определить имя класса, объекты которого занимают наибольший объём памяти JVM; определить пользовательский класс, в экземплярах которого находятся эти объекты.

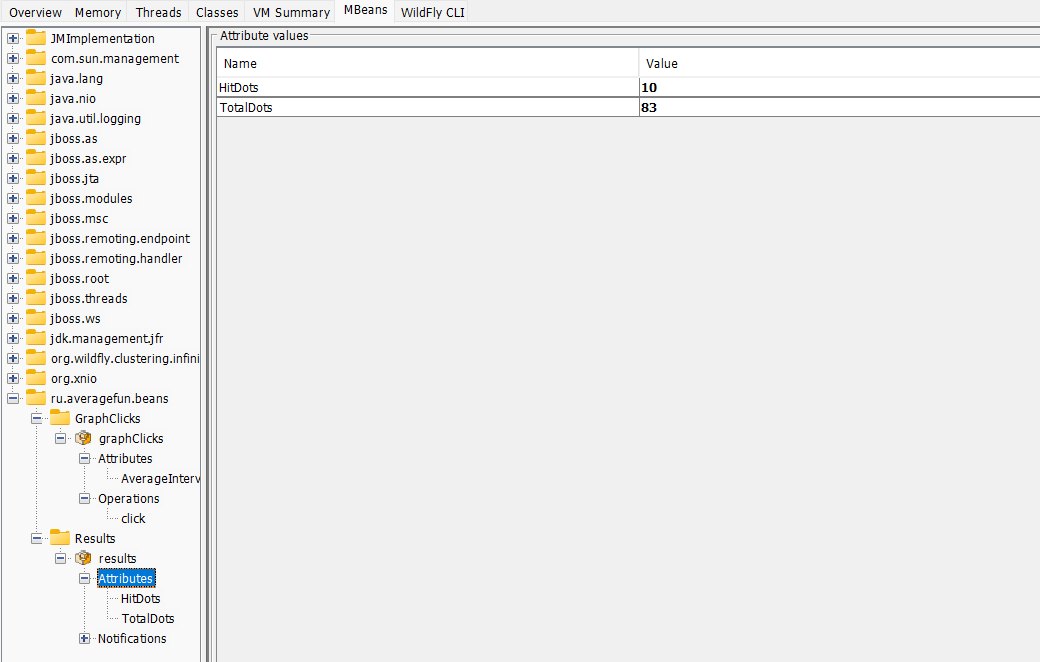
4. С помощью утилиты VisualVM и профилировщика IDE NetBeans, Eclipse или Idea локализовать и устранить проблемы с производительностью в [программе](https://se.ifmo.ru/documents/10180/189115/HttpUnit.tar.gz/7bf1032e-d16e-be85-c71b-dbe73c0178ba?t=1651168887037&download=true). По результатам локализации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

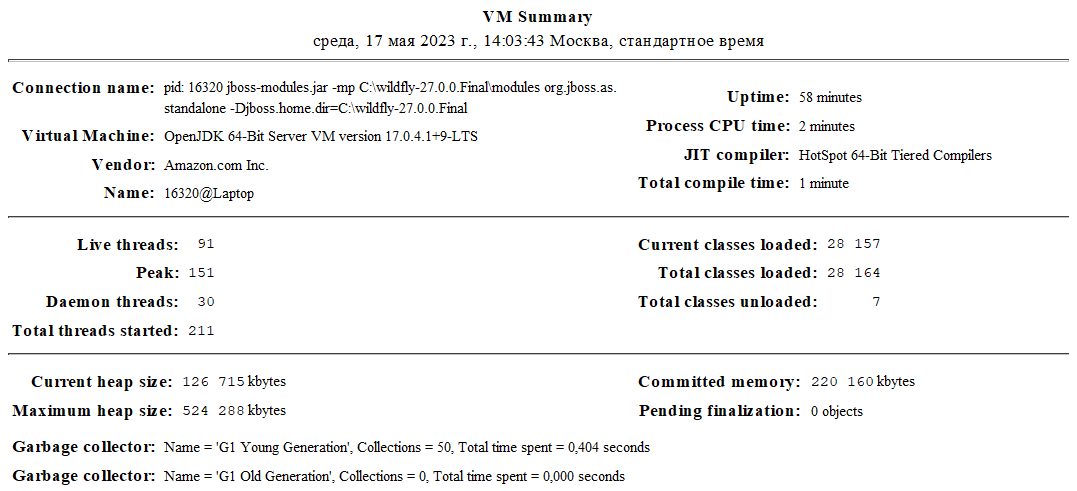
* Описание выявленной проблемы.
* Описание путей устранения выявленной проблемы.
* Подробное (со скриншотами) описание алгоритма действий, который позволил выявить и локализовать проблему.

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

Показания MBean классов из JConsole







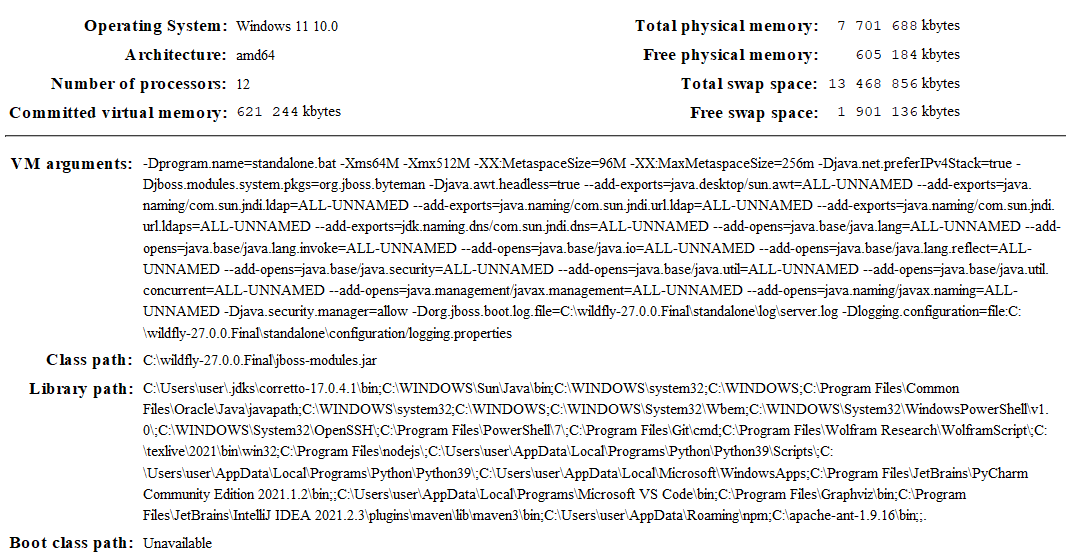
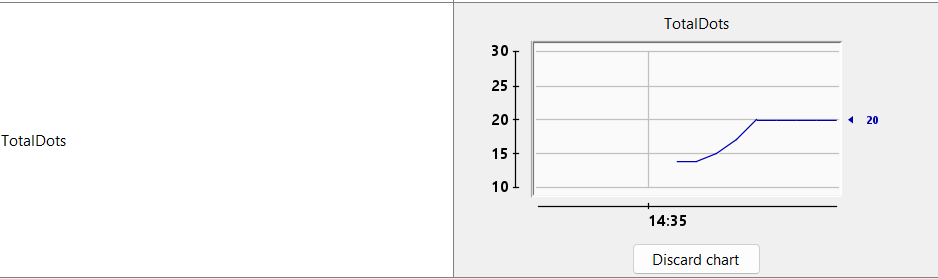
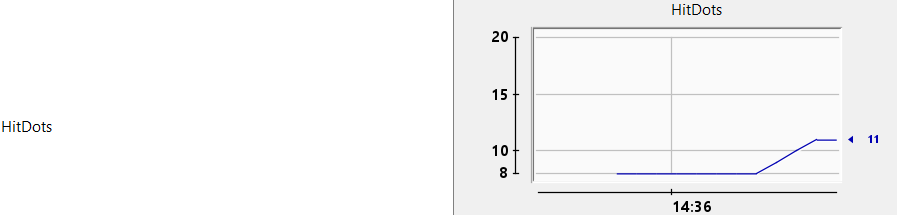


График MBean и наибольший класс об занимаемом объему памяти



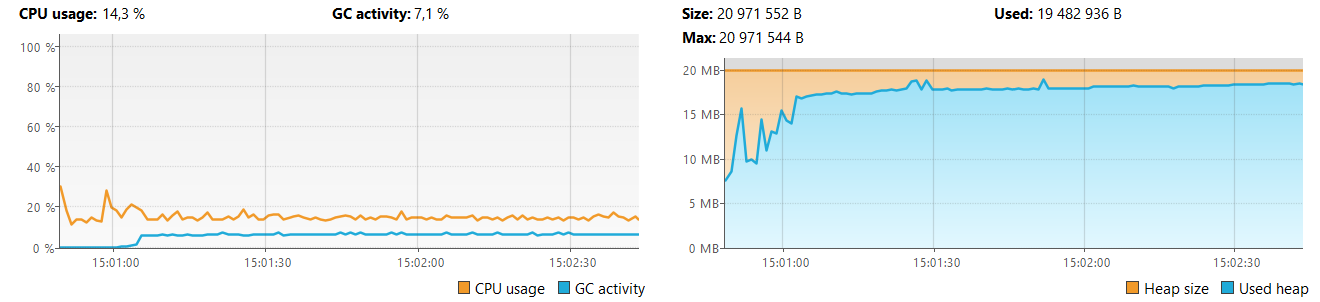




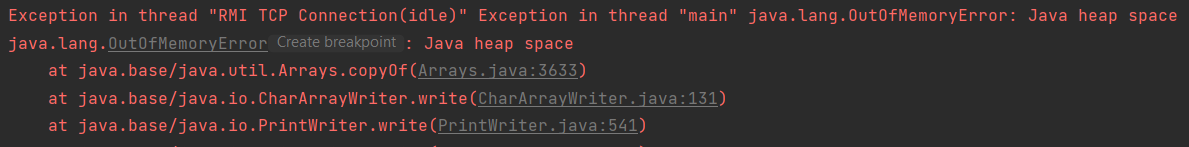
Наибольший процент CPU занимает поток MSC service thread 1-6.

Этот поток является частью механизма управления сервисами, используемого контейнером приложений для запуска, остановки и перезапуска компонентов приложения, таких как EJB-бины или сервлеты.

Исследование программы на утечки памяти

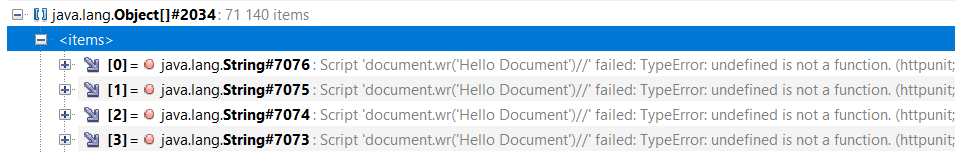


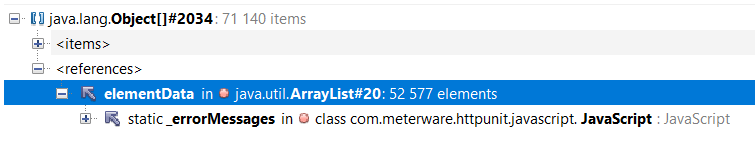
Установим максимальный размер кучи в 20Мб с помощью -Xmx20m и запустим программу. Из графика использования памяти видно, что размер кучи постоянно увеличивается, при этом GC не очищает объекты, что свидетельствует о проблемах с использованием памяти в программе. Через некоторое количество времени получаем ошибку OutOfMemoryError.



С помощью HexDump найдем объекты, занимающие большую часть памяти.

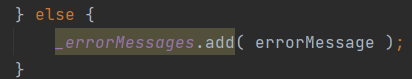






Объекты \_errorMessages хранятся в ArrayList

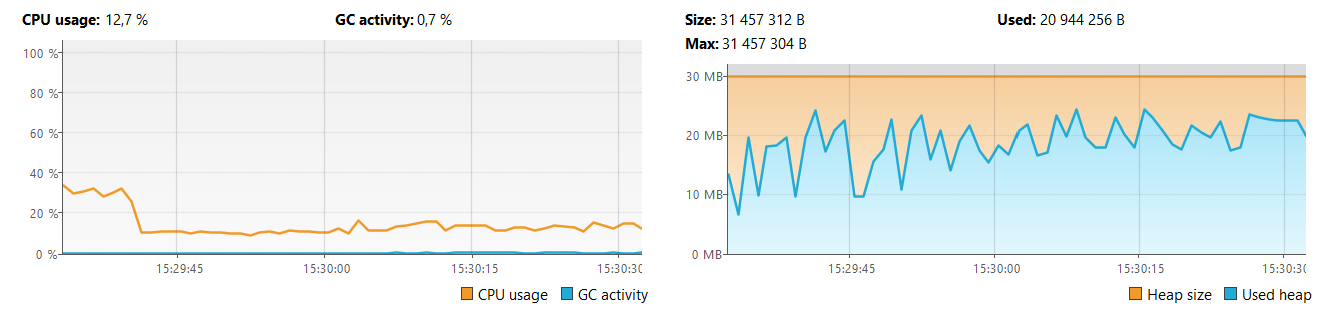
Найдем строчку с добавлением объектов в этот список (она единственна):



В результате получается накопление \_errorMessages в списке, за счет чего и получается переполнение памяти.

Решением будет очистка списка с \_errorMessage после выполнения очередного запроса.

Запустим программу. Видно, что GC теперь загружен не более чем на 1%, также память представляет из себя некую пилу и не стремится к максимальному значению, как было раньше. Теперь программа работает нормально и не выкидывает OutOfMemoryException.



Выводы

Во время выполнения лабораторной работы мы изучили утилиты для мониторинга работы программы JConsole и VisualVM. Научились по полученным данным определять утечки памяти и устранять их.