Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики»

**факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 4**

по дисциплине

‘Методы и средства программной инженерии’

Вариант № 346712

*Выполнили:*

Студент

Р32111 Павлов Александр Сергеевич

*Преподаватель:*

Бострикова Дарья Константиновна



Санкт-Петербург, 2023

Задание

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Исходный код программы

<https://github.com/ExcaliBBur/mispi/tree/main/lab4>

Показания утилиты JConsole

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Демонстрация полей:

Изображение выглядит как текст, снимок экрана, число, линия

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, линия

Автоматически созданное описание

Наличие метода:

Изображение выглядит как текст, снимок экрана, число, линия

Автоматически созданное описание

Выполнение метода:

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Оповещение о 4 промахах подряд:

Изображение выглядит как текст, линия, Шрифт, снимок экрана

Автоматически созданное описание

**Время, прошедшее с момента запуска виртуальной машины:**

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

16 минут = 16 \* 60 \* 60 = 57600 мс.

Показания утилиты VisualVM

График изменения показаний первого бина (счётчика):

Изображение выглядит как текст, снимок экрана, диаграмма, Параллельный

Автоматически созданное описание

График изменения показаний второго бина (площадь):

Изображение выглядит как линия, График, диаграмма, снимок экрана

Автоматически созданное описание

**Имя класса, объекты которого занимают наибольший объем памяти JVM:**

****

Локализация и устранение проблем с производительностью

- Проверка программы на утечку памяти

Основной цикл программы: (измененный Thread.sleep)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

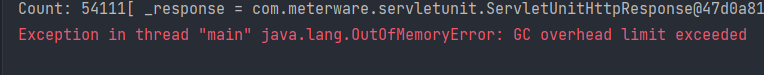
В настройках VM установим максимальный размер кучи -Xmx30m, чтобы быстрее выявить ошибку.

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание

Память постоянно растёт. GC не работает в штатном режиме.

Через 4 минуты Java-программа выбрасывает исключение, которое говорит о нехватке памяти.



**Таким образом, можно сделать вывод о том, что в данной программе существует утечка памяти.**

**-** Локализация проблемы

Анализируя Heap Dump находим объекты, которые занимают много памяти.

**Изображение выглядит как текст, снимок экрана, линия, программное обеспечение

Автоматически созданное описание**

Видим, что char[] занимает подавляющий процент памяти.

- Устранение проблемы

Найдём 5 самых больших объектов по размеру

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Перейдём в java.lang.Object[]#556 и увидим там, что класс JavaScript занимает наибольшее место. Проблема в нём.



Увидим, что добавление элемента в список происходит только в одном методе:

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Элементы накапливаются, но не очищаются. Решением проблемы будет очистка элементов после каждого запроса.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

После этого можно проверять программу на утечку памяти заново.

- Проверка устранения утечки памяти.



После запуска программы получаем следующий график:

Изображение выглядит как снимок экрана, текст, График, линия

Автоматически созданное описание

**Видим, что память не растёт и GC работает в штатном режиме. Проблема устранена.**

Вывод

Во время выполнения лабораторной работы мною были изучены такие утилиты, как JConsole и VisualVM для мониторинга и профилирования Java-приложений.