**Профилирование MemTracker**

Для анализа работы приложения MemTracker использовалась программа **jconsole**.

В работе приложения MemTracker применён сборщик мусора типа **Parallel** (флаг -XX:+UseParallelGC) по следующим причинам:

- лучшая производительность перед типом сборщика мусора Serial;

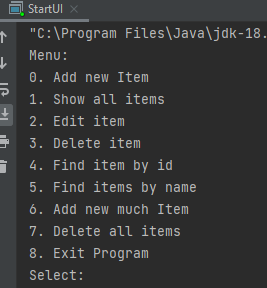
- в работе приложения MemTracker паузы допустимы;

- работа приложения MemTracker производится на многопроцессорном оборудовании;

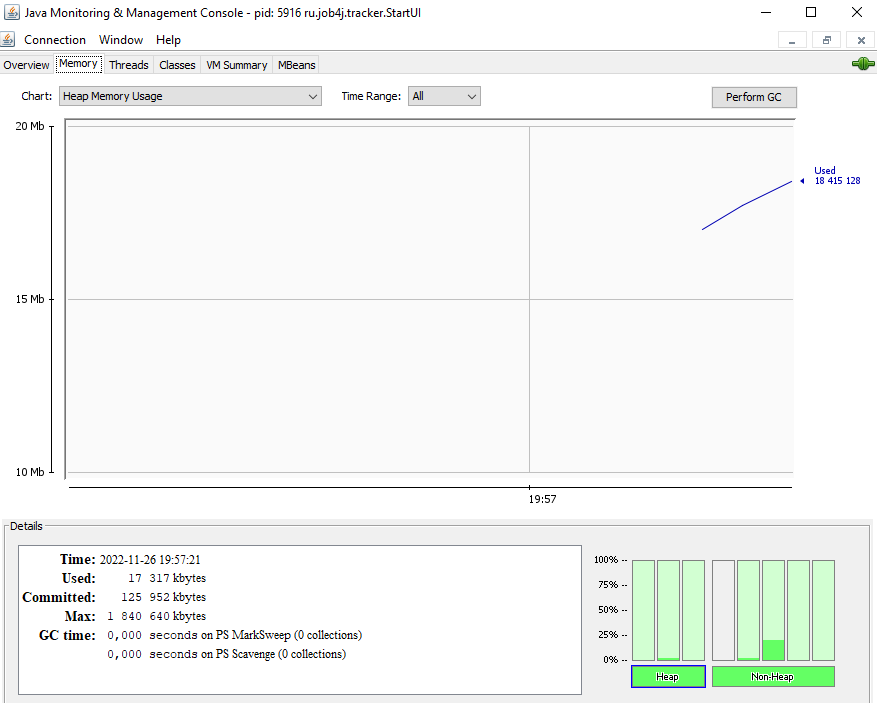
- важнее пропускная способность, чем время отклика приложения MemTracker;

- требует меньше ресурсов для рабочего оборудования (компьютера), чем G1, ZGC.

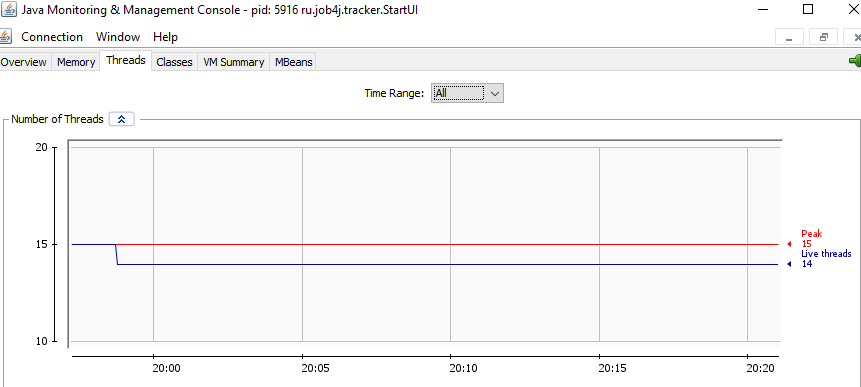
**Запуск приложения:**



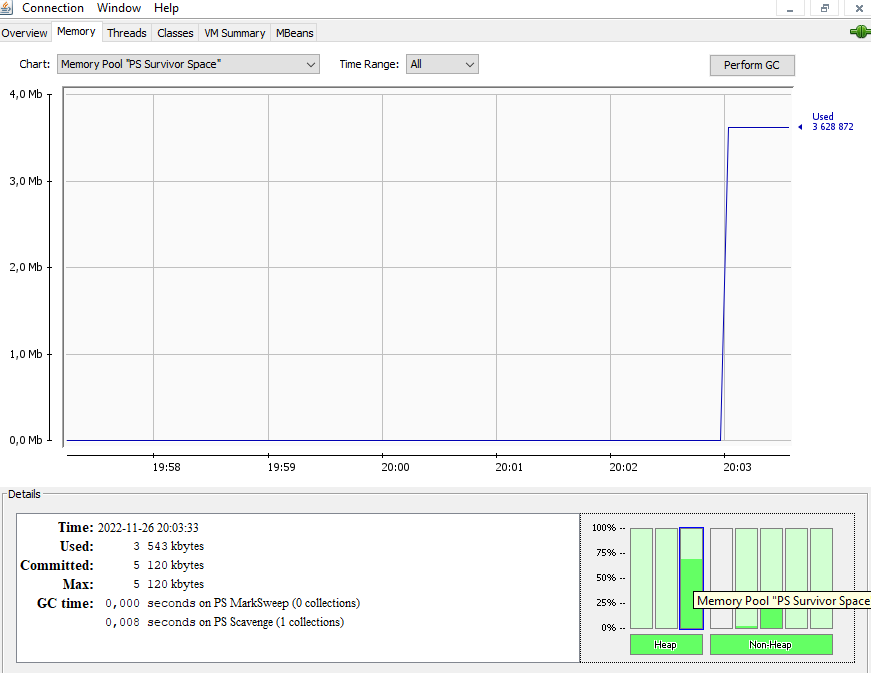
Наблюдаем минимальное заполнение объектами пула памяти в Эдене в размере от 20 Mb до 30 Mb. А также выделение памяти на Stack в Non-heap:



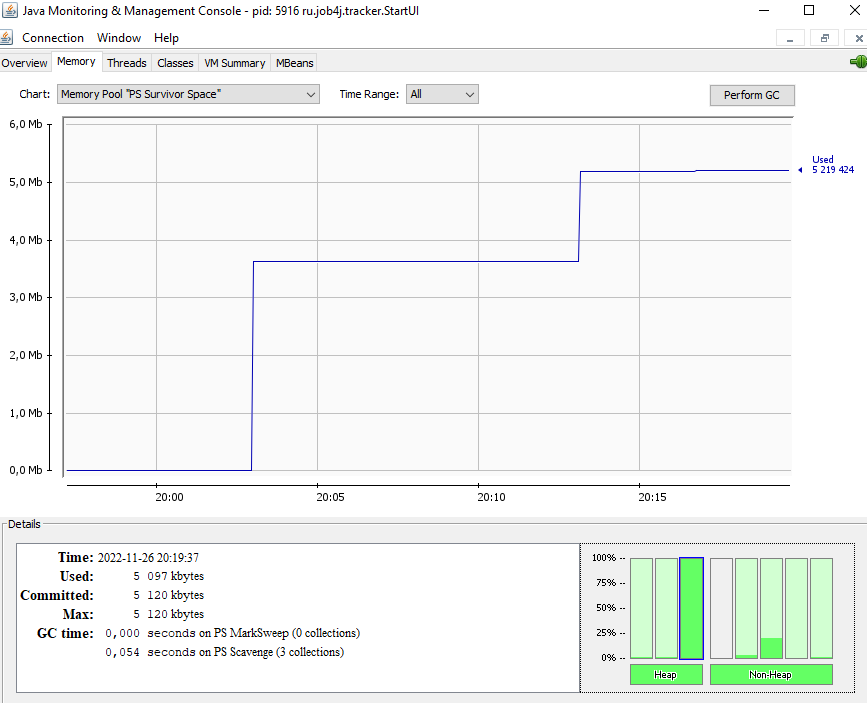
Также наблюдаем 14 активных потоков:



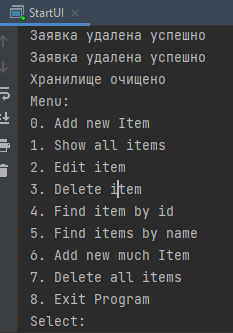
При достижении выделенной памяти в Эдене до уровня ~33 Mb объекты из Эдена постарели и перешли в пул памяти Survivol space:



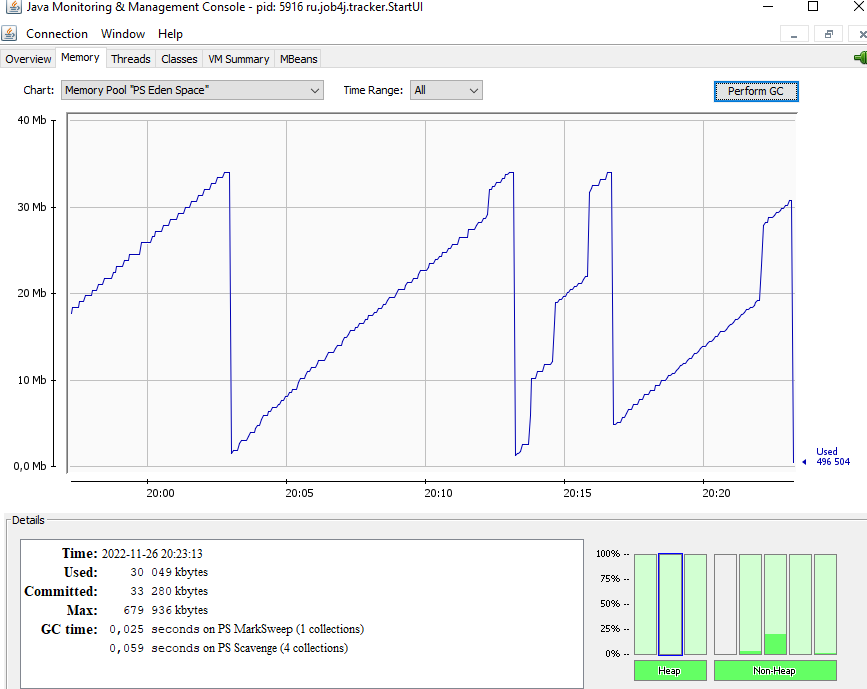
При переполнении пула памяти Survivol space происходит автоматическое расширение количества выделяемой памяти:



Произвел удаление всех объектов:

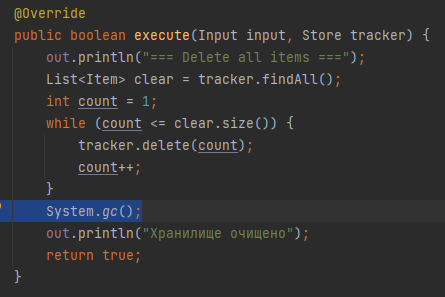


И вызвал очистку мусора, нажатием на Perform GC:



Как результат – пустой Heap.

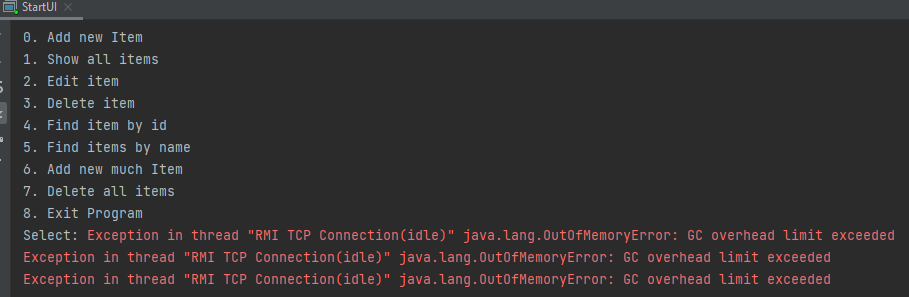
Логично добавить вызов сборки мусора из кода:



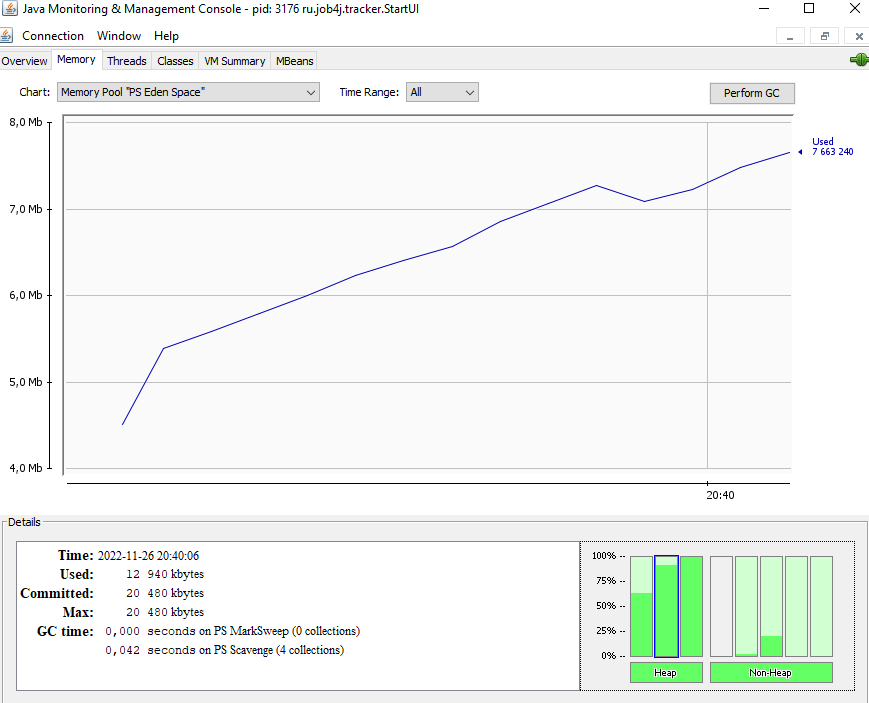
В этом случае происходит заметное «Stop the World» в IDEA.

**Переполнение памяти.**

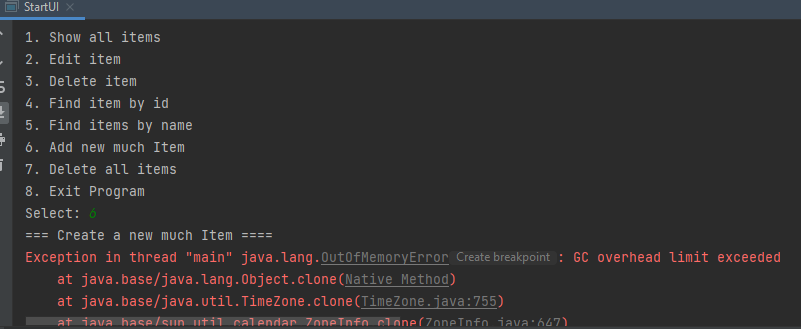
Задал максимальный размер хипа с помощью ключа **–Xmx5m** и получил:



Задал максимальный размер хипа с помощью ключа **–Xmx30m** и получил распределение объектов в OLD Gen для более быстрой работы с Эдемом.:



При очередной попытке добавить множество объектов получил нехватку памяти для приложения:



Для справки:

