Для анализа работы GC в программе Трекер был написан класс `ru.job4j.gcprofile.TrackerGCProfileDemo`. В нем есть метод main, который запускает наш Трекер с некоторыми нужными для исследования параметрами:

1. Создан массив answers с ответами для автоматического создания и удаления большого количества заявок. Для упрощения его заполнения было создано несколько вспомогательных методов.

2. Чтобы добиться вызова GC используются параметры JVM -Xmx30m -Xms30m

3. В качестве сборщика мусора используется Serial GC

4. Создан класс `ru.job4j.gcprofile.CustomStartUI` для замедления процесса обработки заявок (задержка составляет 500 мс). Замедление установлено чтобы не дать программе быстро выполниться и успеть посмотреть работу GC во время выполнения программы.

5. Количество answer-ов задается из консоли. Измерения проводились для значения n = 1000.

Далее с помощью jconsole были исследованы состояние JVM и процесс работы GC. Режим работы был следующим: 1000 запросов - сначала создается 250 заявок, потом большая часть созданных заявок удаляется, потом опять начинается процесс создания заявок. Последней командой является exit.

Сначала идет увеличение количество потребляемой памяти в heap (пока создаются 250 заявок), далее процесс создания заявок заканчивается и начинается их удаление. При достижении ~11-12 МБ начинает работу GC и высвобождается часть места в heap. Процесс удаления продолжается и через некоторое время завершается. Потом начинается процесс создания новых заявок. Опять при значениях примерно ~11-12 МБ начинает работу GC и через некоторое время программа завершает свою работу. Все эти процессы хорошо видны на приложенном графике (Рис. 1)

При уменьшении объема памяти heap, JVM чаще вызывает GC, что видно на графиках на Рис. 2.

При установке значения -Xmx3m объем памяти heap все равно колеблется в пределах от 3 до 4 МБ, что видно на последнем графике Рис. 2.

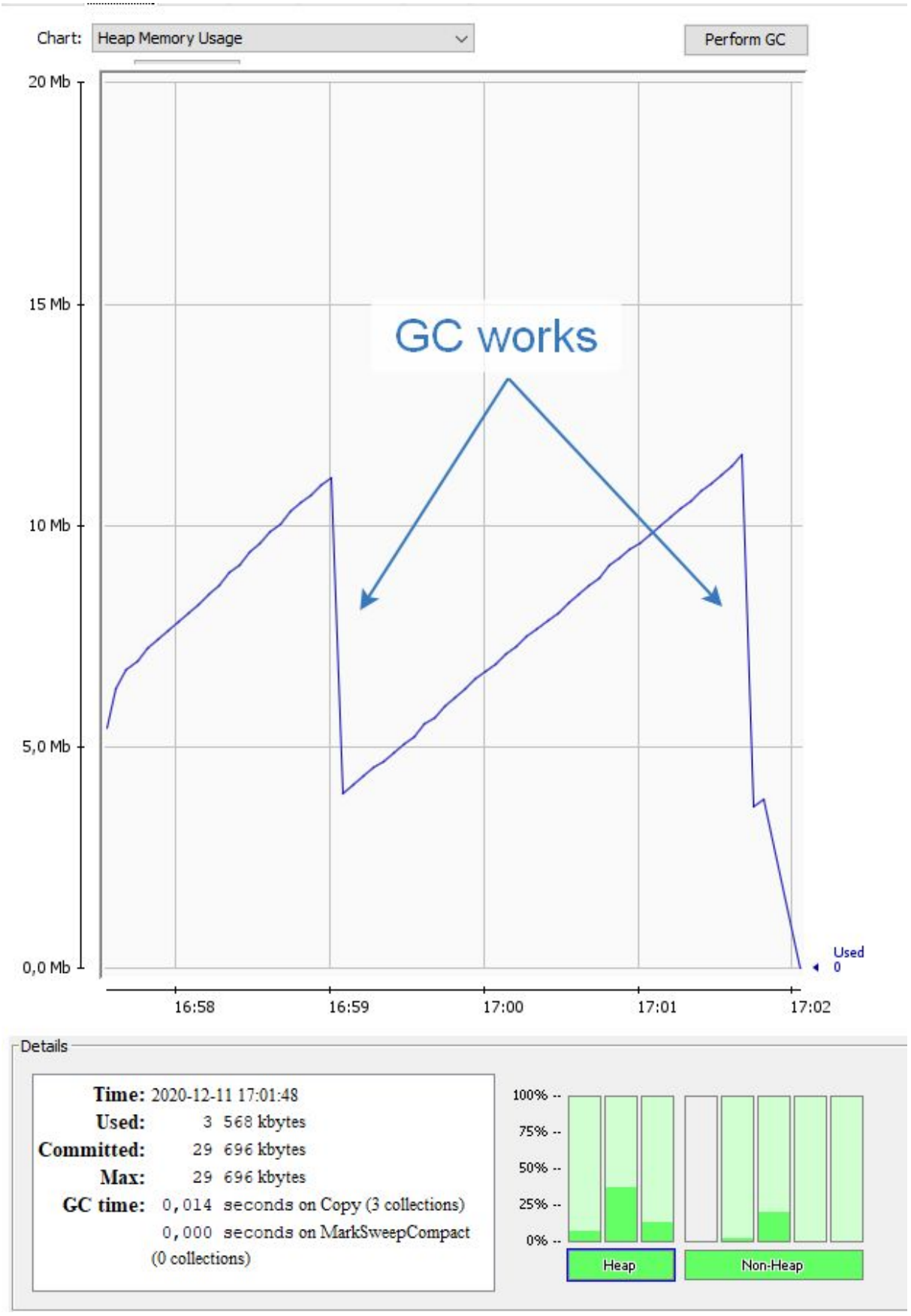

Рис. 1 График jconsole

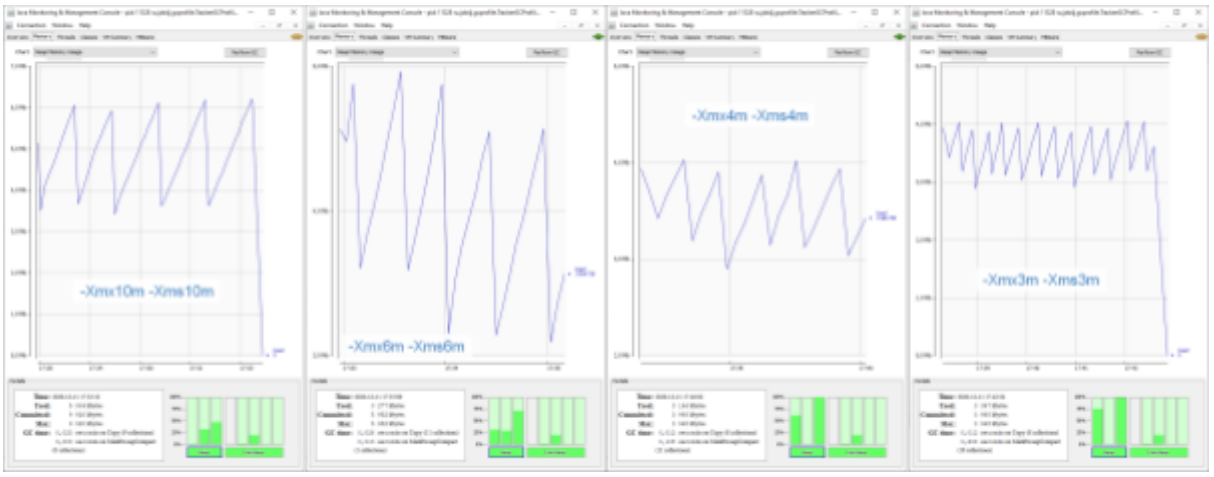

Рис.2 Графики для разных объемов памяти