Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Основы программной инженерии

Лабораторная работа №4

Вариант № 453164

Выполнили:

Кобик Никита Алексеевич

Маликов Глеб Игоревич

Группа № P3224

Преподаватель:

Коновалов Арсений Антонович

г. Санкт-Петербург
2024

1 Задание

1. Для своей программы из лабораторной работы №3 по дисциплине "Веб программирование"реализовать:

   - MBean, считающий общее число установленных пользователем точек, а также число точек, не попадающих в область. В случае, если координаты установленной пользователем точки вышли за пределы отображаемой области координатной плоскости, разработанный MBean должен отправлять оповещение об этом событии.
   - MBean, определяющий процентное отношение "попаданий"к общему числу кликов пользователя по координатной плоскости.

2. С помощью утилиты JConsole провести мониторинг программы:

   - Снять показания MBean-классов, разработанных в ходе выполнения задания 1.
   - Определить время (в мс), прошедшее с момента запуска виртуальной машины.

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 потока, потребляющего наибольший процент времени CPU.

4. С помощью утилиты VisualVM и профилировщика IDE NetBeans, Eclipse или Idea локализовать и устранить проблемы с производительностью в программе. 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.

2 Отчет

2.1 MBeans

Интерфейс MBean

```
1  package com.example.weblab3.management;
2
3  public interface RegisterMBean {
4
5      int get_total();
6      int get_missed();
7      double get_ratio();
8      void increment_total_points();
9      void increment_missed_points();
10
11     void sendNotification();
12
13     void updateRatio();
14
15 }
```

MBean класс отслеживания точек

```
1  package com.example.weblab3.management;
2
3
4  import jakarta.enterprise.context.ApplicationScoped;
5  import jakarta.inject.Named;
6
7  @Named
8  @ApplicationScoped
9  public class Register implements RegisterMBean{
10     private int total_points = 0;
11     private int missed_points = 0;
12     private double ratio = 0.0;
13
14     @Override
15     public int get_total() {
16         return this.total_points;
17     }
18
19     @Override
20     public int get_missed() {
21         return this.missed_points;
22     }
23
24     @Override
25     public double get_ratio() {
26         return this.ratio;
27     }
28
29     @Override
```

```
30      public void increment_total_points() {
31          this.total_points += 1;
32          this.updateRatio();
33          System.out.println("Total points : " + this.total_points +
        " Ratio : " + this.ratio);
34      }
35
36      @Override
37      public void increment_missed_points() {
38          this.missed_points += 1;
39          this.updateRatio();
40          this.sendNotification();
41          System.out.println("Total missed : " + this.missed_points)
        ;
42      }
43
44      @Override
45      public void updateRatio() {
46          this.ratio = (double) (this.total_points - this.
        missed_points) / this.total_points * 100;
47      }
48
49      @Override
50      public void sendNotification() {
51          System.out.println("Point is out of bounds");
52      }
53  }
```

Agent

```
1  package com.example.weblab3.management;
2
3  import jakarta.annotation.PostConstruct;
4  import jakarta.enterprise.context.ApplicationScoped;
5  import jakarta.enterprise.context.Initialized;
6  import jakarta.enterprise.event.Observes;
7  import jakarta.inject.Inject;
8  import jakarta.inject.Named;
9
10 import javax.management.MBeanServer;
11 import javax.management.ObjectName;
12 import java.lang.management.ManagementFactory;
13
14 @Named
15 @ApplicationScoped
16 public class Agent {
17
18     @Inject
19     private Register register;
20
21     @PostConstruct
22     public void initAgent() {
```

```
23          MBeanServer mbs = ManagementFactory.getPlatformMBeanServer
    ();
24
25          ObjectName mBean;
26
27          try {
28              mBean = new ObjectName("com.example.weblab3.management
    :type=Register");
29              if (!mbs.isRegistered(mBean)) {
30                  mbs.registerMBean(register, mBean);
31              }
32          } catch(Exception e) {
33              e.printStackTrace();
34          }
35      }
36
37      public void logSimpleAgentStarted() {
38          System.out.println("SimpleAgent.logSimpleAgentStarted");
39      }
40
41      public void startup(@Observes @Initialized(ApplicationScoped.
    class) Object context) {
42          Agent a = new Agent();
43          a.logSimpleAgentStarted();
44      }
45 }
```

2.2 JConsole

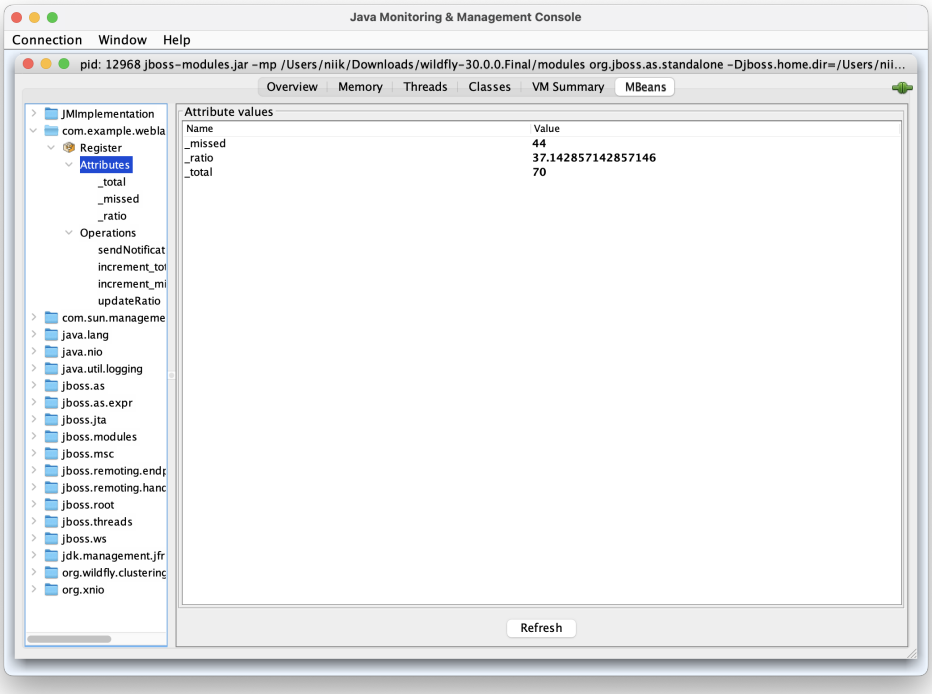


Рис. 1: Показания разработанного класса

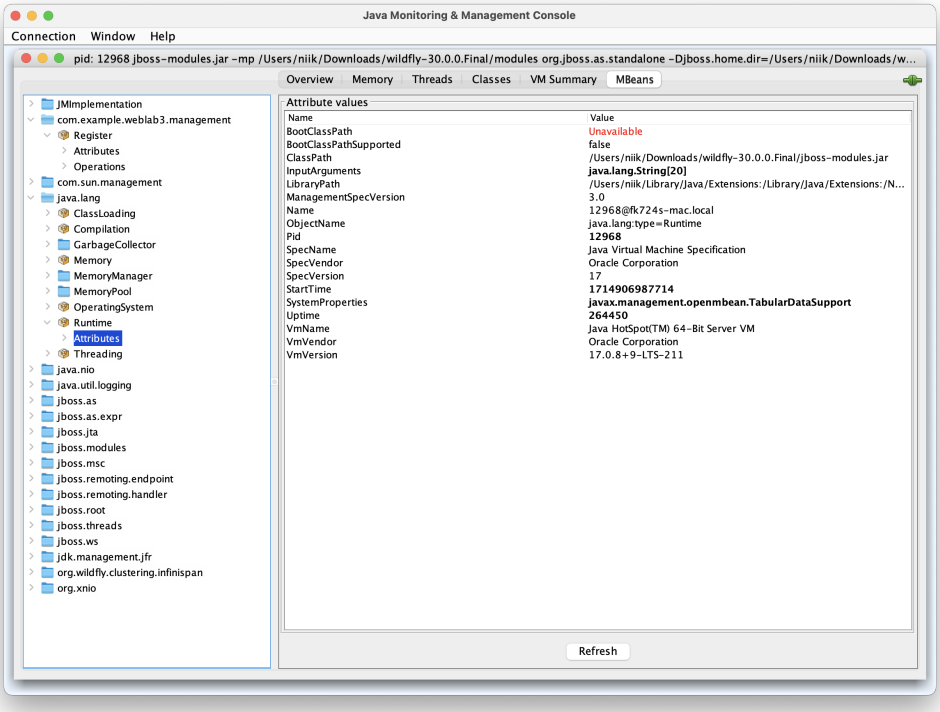


Рис. 2: Runtime информация

2.3 VisualVM

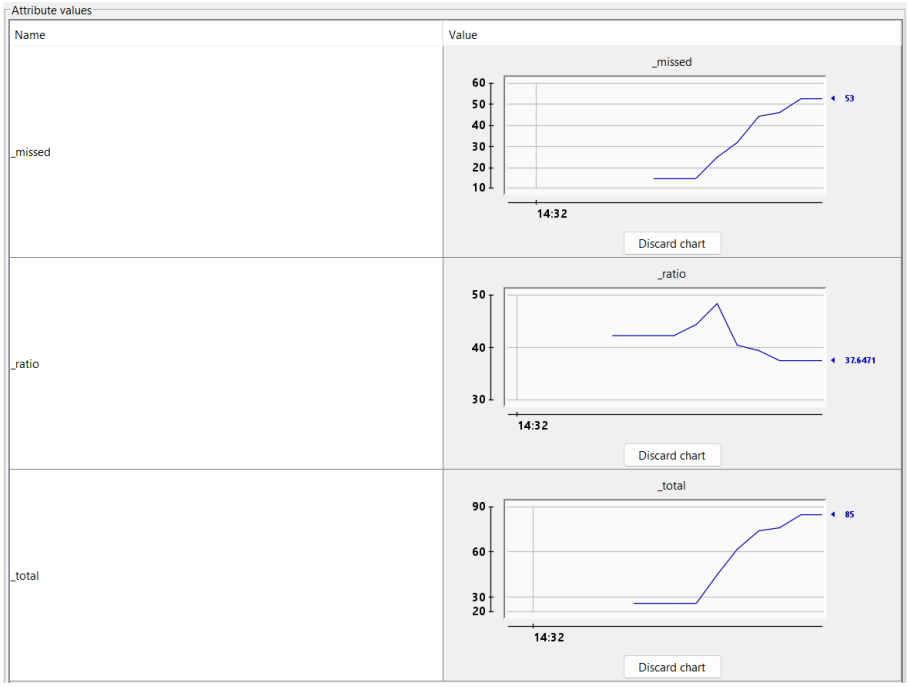


Рис. 3: Графики показаний MBean классов

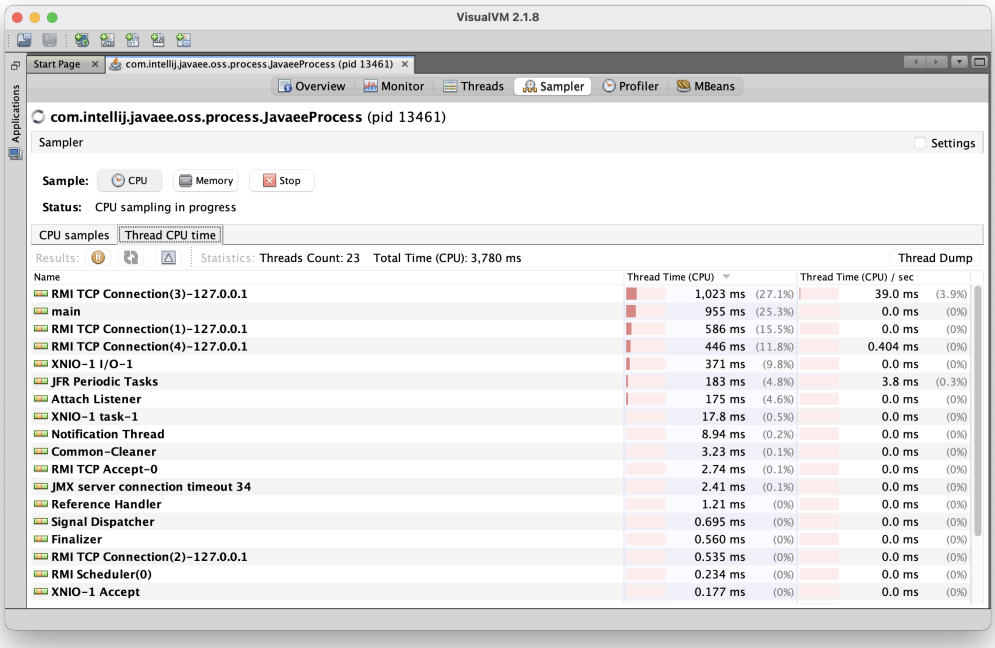
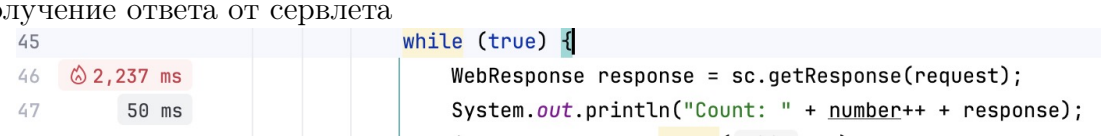


Рис. 4: Потоки с наибольшим процентом времени CPU

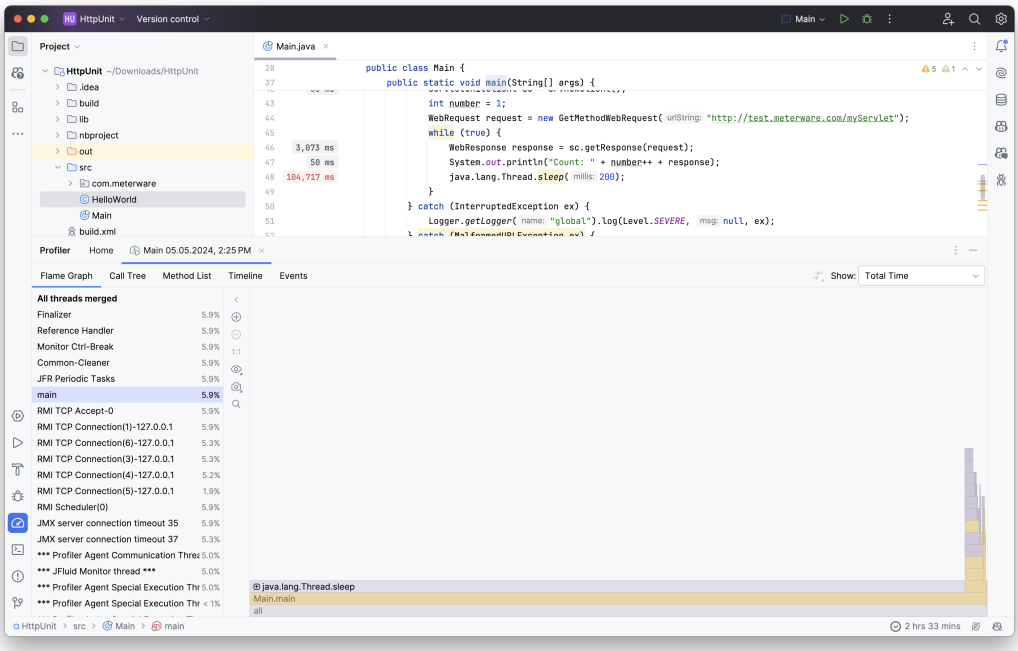
2.4 Устранение проблем производительности

2.4.1 Проблема

Код создает запросы к сервлету с помощью HTTPUnit и выводит ответы в консоль. По данным из IntelliJ IDEA Profiler наибольшее время процессора занимает получение ответа от сервлета



Хотя, наибольшее общее время занимает пауза потока Thread.sleep(200)



Также на производительность влияют:

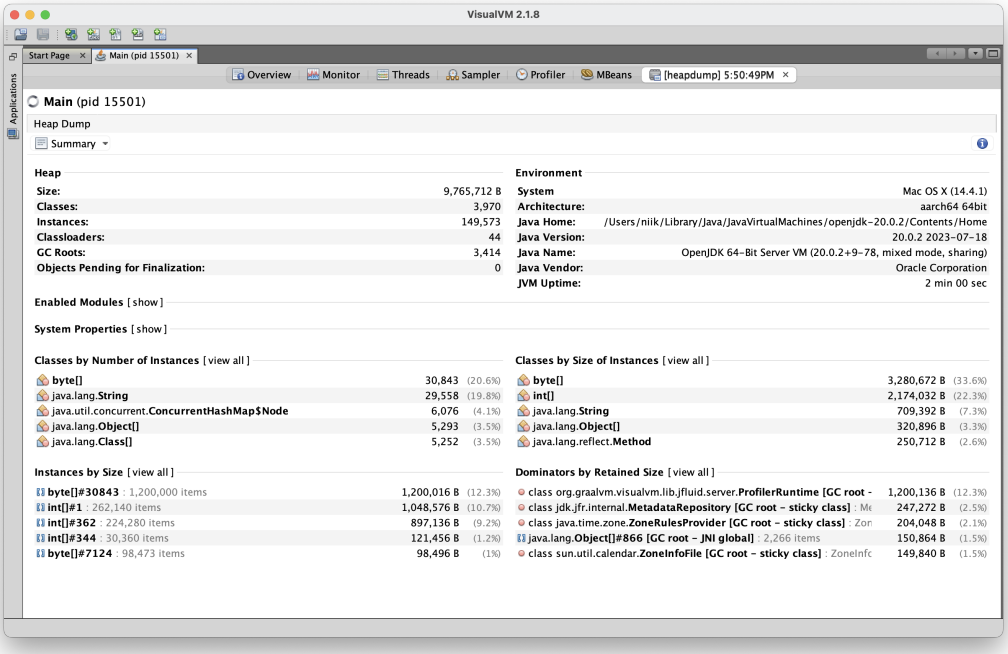
- Бесконечный цикл
- Отсутствие закрытия соединения
- Отсутствие проверки на null в ответе от сервлета
- Большая пауза между итерациями

2.4.2 Устранение

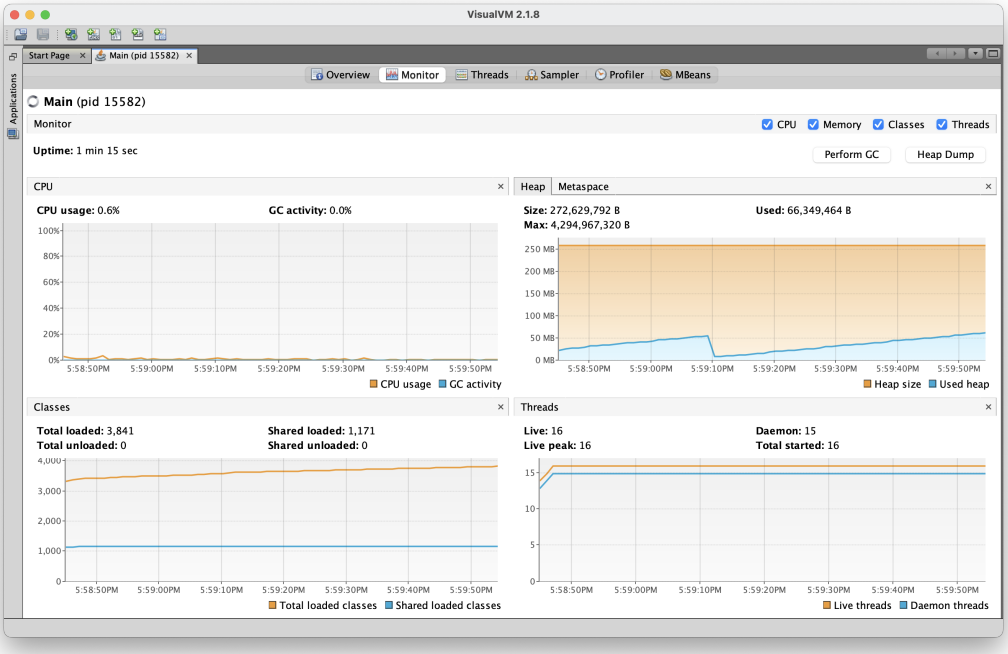
- Задать ограничение на количество итераций
- Закрыть соединение перед завершением
- Добавить проверку ответа на null

```
1 public static void main(String[] args) {
2     try {
3         HttpUnitOptions.setExceptionsThrownOnScriptError(false);
4
```

```
5           ServletRunner sr = new ServletRunner();
6           sr.registerServlet("myServlet", HelloWorld.class.getName()
    );
7           ServletUnitClient sc = sr.newClient();
8
9           int number = 1;
10          int maxRequests = 1000;
11
12          WebRequest request = new GetMethodWebRequest("http://test.
    meterware.com/myServlet");
13          WebResponse response;
14
15          while (number <= maxRequests) {
16              response = sc.getResponse(request);
17
18              if (response != null) {
19                  System.out.println("Count: " + number++ + " " +
    response);
20              } else {
21                  System.out.println("Response is null");
22              }
23
24              Thread.sleep(100);
25          }
26
27          sr.shutDown();
28
29      } catch (InterruptedException ex) {
30          Logger.getLogger("global").log(Level.SEVERE, null, ex);
31      } catch (MalformedURLException ex) {
32          Logger.getLogger("global").log(Level.SEVERE, null, ex);
33      } catch (IOException ex) {
34          Logger.getLogger("global").log(Level.SEVERE, null, ex);
35      } catch (SAXException ex) {
36          Logger.getLogger("global").log(Level.SEVERE, null, ex);
37      }
38  }
```



Видно, что наибольший размер в памяти имеет сам garbage collector, к тому же, очистки производятся не так часто что означает корректную работу системы



3 Заключение

В ходе выполнения лабораторной работы мы познакомились со средствами мониторинга и оценки производительности Jconsole и VirtualVM