

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

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

**Факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА №4**  
по дисциплине  
“Методы и средства программной инженерии”

Вариант № -1

**Студенты:**

Чухно Матвей  
Романович

Воробьев Кирилл Олегович

Группа Р3230 & Р3231

**Преподаватель:**

Письмак Алексей Евгеньевич



Санкт-Петербург, 2022

## **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 количество установленных пользователем точек стало кратно 5, разработанный MBean должен отправлять оповещение об этом событии.
  - MBean, определяющий площадь получившейся фигуры.
2. С помощью утилиты JConsole провести мониторинг программы:
  - Снять показания MBean-классов, разработанных в ходе выполнения задания 1.
  - Определить версию Java Language Specification, реализуемую данной средой исполнения.
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.

## JConsole:

**ResultMBean**

Overview | Memory | Threads | Classes | VM Summary | MBeans

**Attribute values**

Name	Value
CountOfMissPoints	1
CountOfPoints	3

[Refresh](#)

pid: 35321 jboss-modules.jar -mp /Users/matthew/Documents/wildfly-25.0.0.Final/modules org.jboss.as.standalone -Dboss.home.dir=/Users/matthew/Documents/wildfly-25.0.0.Final -Dboss.server.base...

Overview | Memory | Threads | Classes | VM Summary | MBeans

**Notification buffer**

TimeStamp	Type	UserD...	SeqNum	Message	Event	Source
14:56:47.956	countOfMissedPointsMultip...		35	Count of missed points is multi...	javax.management.Notification[so...	com.example.Web3.mbeans.resul...
14:56:42.242	countOfMissedPointsMultip...		30	Count of missed points is multi...	javax.management.Notification[so...	com.example.Web3.mbeans.resul...
14:56:38.843	countOfMissedPointsMultip...		25	Count of missed points is multi...	javax.management.Notification[so...	com.example.Web3.mbeans.resul...
14:56:36.496	countOfMissedPointsMultip...		20	Count of missed points is multi...	javax.management.Notification[so...	com.example.Web3.mbeans.resul...
14:56:32.751	countOfMissedPointsMultip...		15	Count of missed points is multi...	javax.management.Notification[so...	com.example.Web3.mbeans.resul...
14:55:29.840	countOfMissedPointsMultip...		10	Count of missed points is multi...	javax.management.Notification[so...	com.example.Web3.mbeans.resul...
14:55:25.515	countOfMissedPointsMultip...		5	Count of missed points is multi...	javax.management.Notification[so...	com.example.Web3.mbeans.resul...

## SquareMBean

Overview | Memory | Threads | Classes |

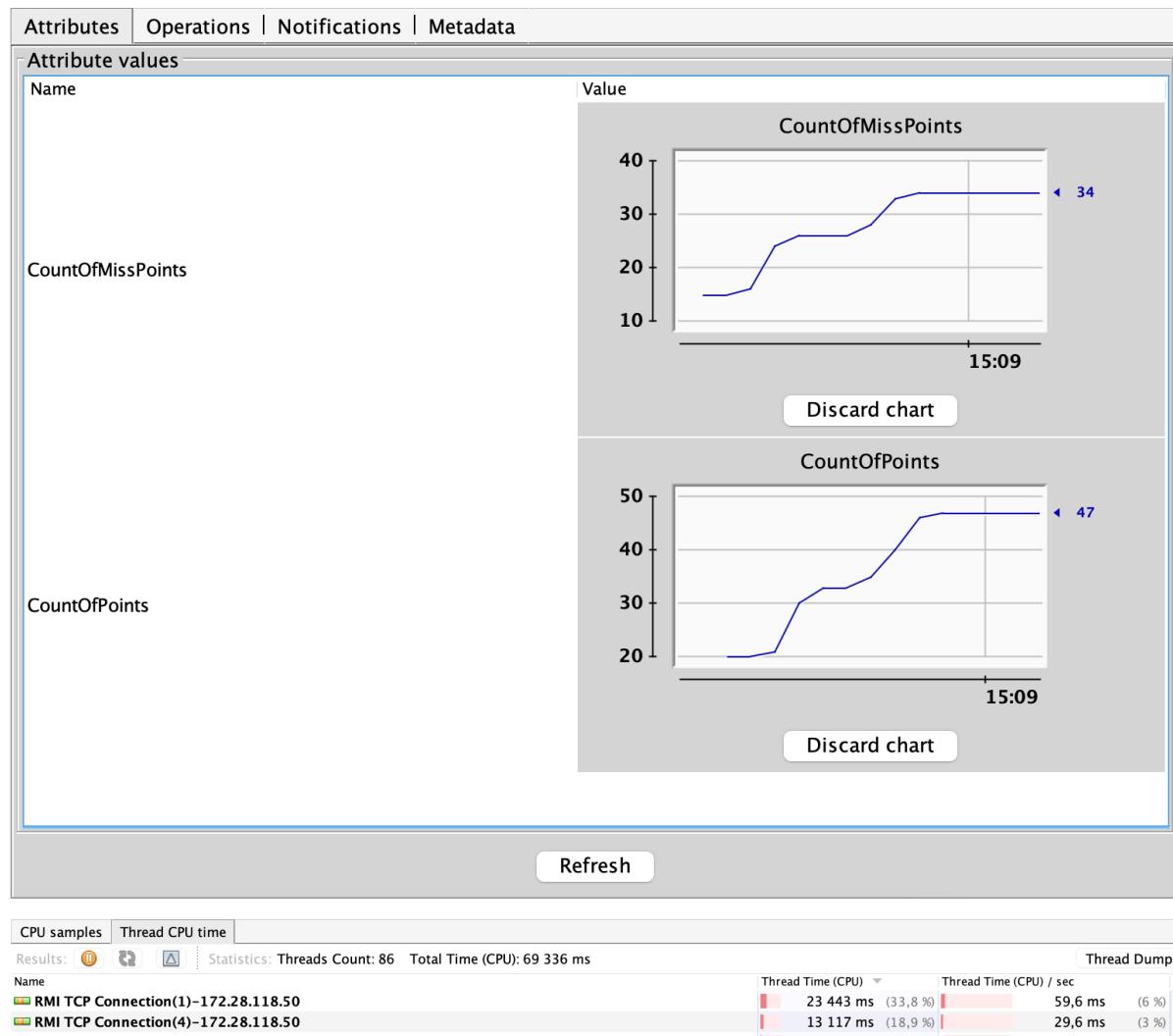
The screenshot shows the JBoss Tools interface for a 'SquareMBean'. On the left, a tree view lists various Java packages and components, including 'JMImplementation', 'com.example.Web3' (which contains 'ResultCheckerBE' and 'CalculateSquareE'), and several JBoss subsystems like 'com.oracle.jdbc', 'com.sun.management', etc. In the center, a 'Operation invocation' dialog is open for the 'calculateSquare' method of the 'CalculateSquareE' component. The method signature is 'double calculateSquare ()'. A modal window titled 'Operation return value' displays the result '34.63495408493621'. The 'OK' button at the bottom right of the modal is highlighted with a purple border.

Overview | Memory | Threads | Classes |

This screenshot is nearly identical to the one above, showing the same JBoss Tools interface and the 'SquareMBean' component. However, the 'Operation return value' modal window now displays the error message '6.141592653589793', which is the value of the square root of 36, indicating a failure or an unexpected result for the square calculation.

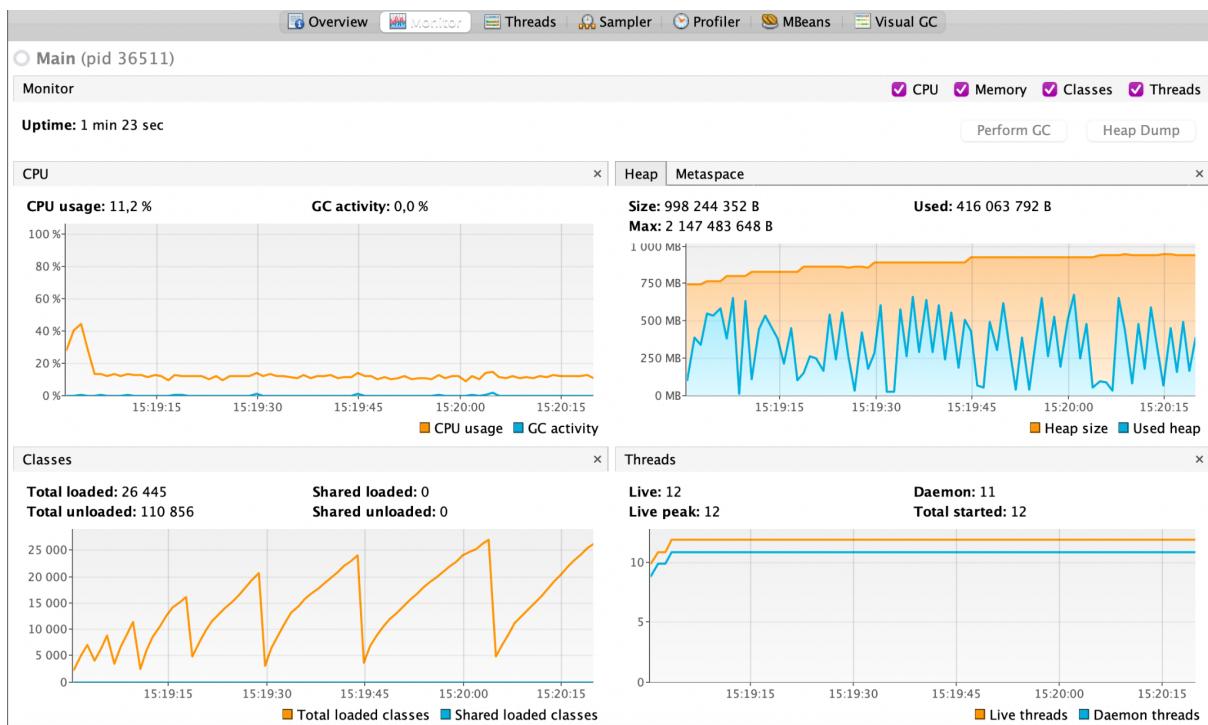
**Connection name:** pid: 35775 jboss-modules.jar -mp /Users/matthew/Documents/wildfly-25.0.0.Final/modules org.jboss.as.standalone -Djboss.home. dir=/Users/matthew/Documents/wildfly-25.0.0.Final -Djboss.server.base.dir=/Users/matthew/Documents/wildfly-25.0.0.Final/standalone  
**Virtual Machine:** OpenJDK 64-Bit Server VM version 16.0.1+9-24  
**Vendor:** Oracle Corporation  
**Name:** 35775@MacBook-Pro-Matthew.local

## VisualVM:

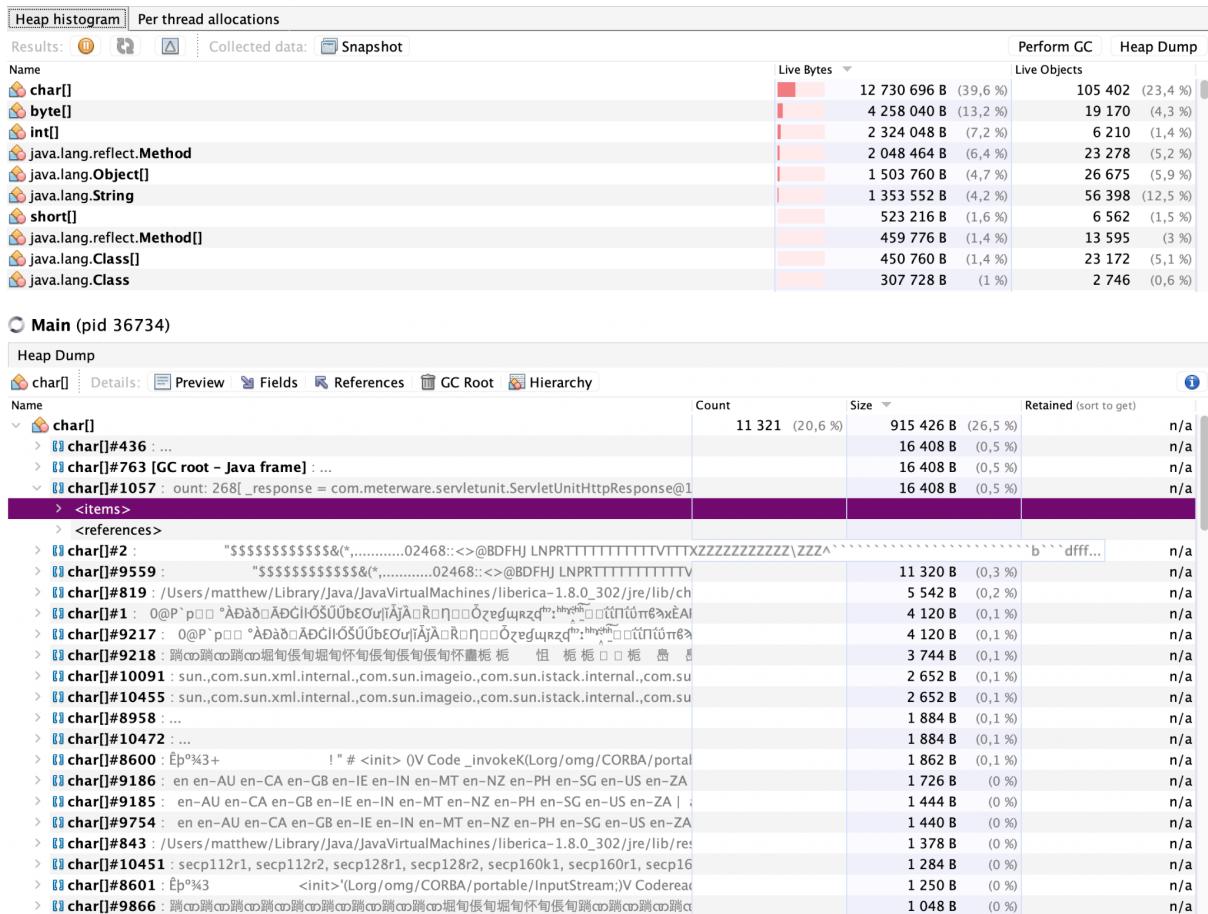


## Обнаружение и устранение проблемы:

В утилите VisualVM во вкладке Monitor наблюдаем за использованием ресурсов. По графикам видно, что количество классов постоянно растет. Даже после отработки GC тенденция к росту остается. Размер кучи достаточно часто меняется. По графику Used Heap также можно заметить частый вызов GC, это негативно влияет на производительность.



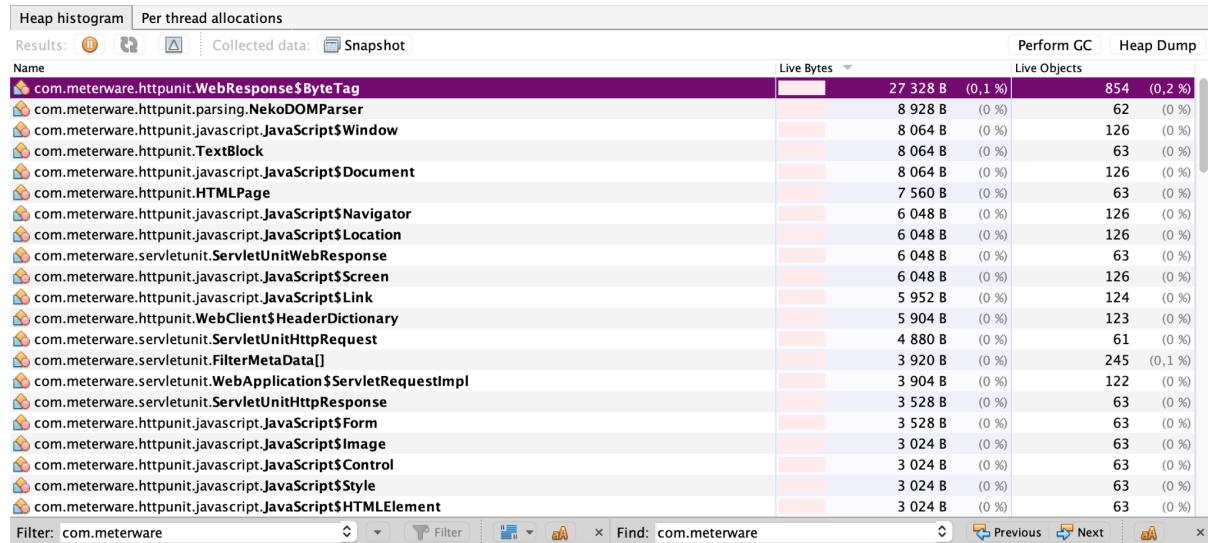
Первая странность – постоянная рост количества массивов char, даже после отработки GC их максимальное количество растет.



Отфильтруем по `com.meterware`. т.е. по нашему пакету. Видим, что класс `WebResponse` (а точнее его часть) обладает таким же поведением.

Постоянно

растёт и удаляется GC. Попробуем посмотреть, что происходит у нас в программе.



Основная проблема:

Изначально в коде программы лежит данная строчка:

```
HttpUnitOptions.setExceptionsThrownOnScriptError(false);

/**
 * Determines whether script errors result in exceptions or warning messages.
 */
public static void setExceptionsThrownOnScriptError( boolean throwExceptions ) {
    _exceptionsThrownOnScriptError = throwExceptions;
    getScriptingEngine().setThrowExceptionsOnError( throwExceptions );
}

private void handleScriptException( Exception e, String badScript ) {
    final String errorMessage = badScript + " failed: " + e;
    if (!(e instanceof EcmaError) && !(e instanceof EvaluatorException)) {
        e.printStackTrace();
        throw new RuntimeException( errorMessage );
    } else if (isThrowExceptionsOnError()) {
        e.printStackTrace();
        throw new ScriptException( errorMessage );
    } else {
        _errorMessages.add( errorMessage );
    }
}
```

Она определяет, приводят ли ошибки скрипта к исключениям или сообщениям об ошибке и записывает их в ArrayList:

```
private static ArrayList _errorMessages = new ArrayList();
```

Далее при изменении аргумента на true при запуске программы начали вылетать exception'ы (которые ранее копились в arrayList'e).

В сервлете находится строчка document.wr('Hello Document'):

```
out.println("<script language=\"JavaScript\" type=\"text/javascript\">");  
out.println("<!--");  
out.println("document.wr('Hello Document')");  
out.println("//-->");  
out.println("</script>");
```

При парсинге вылетает исключение, так как нет функции document.wr(), но есть функция document.write()

```
out.println("<script type=\"text/javascript\" language=\"JavaScript\">");  
out.println("<!--");  
out.println("document.write('Hello Document')");  
out.println("//-->");
```

Соответственно при каждом запросе возникала ошибка и добавлялась в список ошибок. Также в классе JavaScript есть метод, который очищает данный лист, но он нигде не вызывается.

```
/**  
 * Clears the accumulated script error messages.  
 */  
public static void clearScriptErrorMessages() {  
    getScriptingEngine().clearErrorMessages();  
}
```

Поэтому есть два решения этой проблемы:

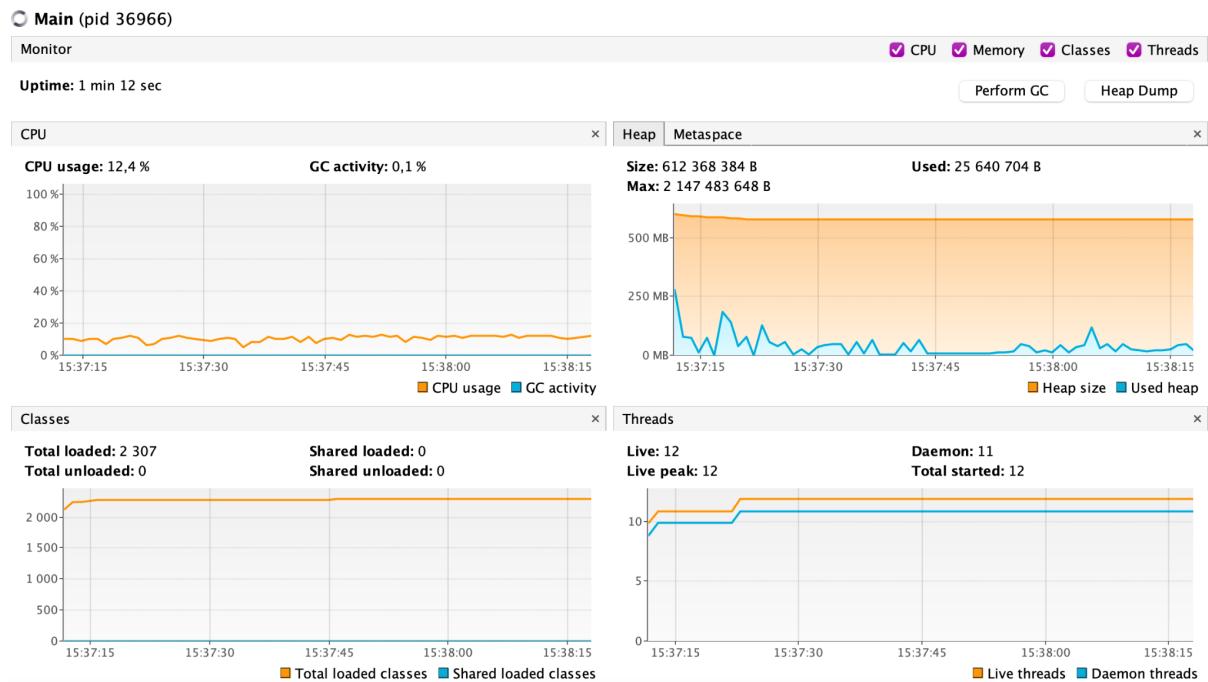
1. Перед добавлением ошибки в ArrayList проверять его размер и при определенном значении вызывать функцию clearScriptErrorMessages
2. Исправить функцию document.wr() на document.write, тогда ошибки не будут возникать и записываться в ArrayList.

Также существует еще такой недочет:

```
int number = 1;  
WebRequest request = new GetMethodWebRequest( urlString: "http://localhost:8080/myServlet");  
while (true) {  
    WebResponse response = sc.getResponse(request);  
    System.out.println("Count: " + number++ + response);  
    java.lang.Thread.sleep( millis: 200);  
}
```

Видно, что в данном куске кода создается WebRequest и в бесконечном цикле постоянно создается WebResponse на один и тот же request. Поскольку ответ всегда будет один и тот же, то не стоит каждый раз создавать объекты WebResponse, достаточно один раз получить ответ и его выводить (соответственно увеличивая переменную number).

```
WebRequest request = new GetMethodWebRequest( urlString: "http://localhost:8080/myServlet");
WebResponse response = sc.getResponse(request);
while (true) {
    System.out.println("Count: " + number++ + response);
    java.lang.Thread.sleep( millis: 200);
}
```



<https://youtube.com/clip/Ugkx-3SpwgTBRzDJcq117GnA1KchDwmtbi4S>

**Вывод:** В ходе данной лабораторной работы мы познакомились с профилированием Java приложений , поработали с технологией JMX , а именно создали свои MBean'ы для сбора статистики. В JMX Agent'е (JConsole) сняли показания написанных бинов и посмотрели на нагрузку ,

создаваемую приложением. С помощью VisualVM были выявлены и устранены проблемы в предоставленной программе.