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 № 2

Студенты:

Степанов Михаил
Андреевич

Агеев Дмитрий Сергеевич

Группа P3231

Преподаватель:

Письмак Алексей Евгеньевич

Санкт-Петербург, 2022

1. Для своей программы из лабораторной работы #3 по дисциплине "Веб-программирование" реализовать:

- MBean, считающий общее число установленных пользователем точек, а также число точек, 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 значение переменной classpath для данной JVM.

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 класса, объекты которого занимают наибольший объём памяти JVM; определить пользовательский класс, в экземплярах которого находятся эти объекты.

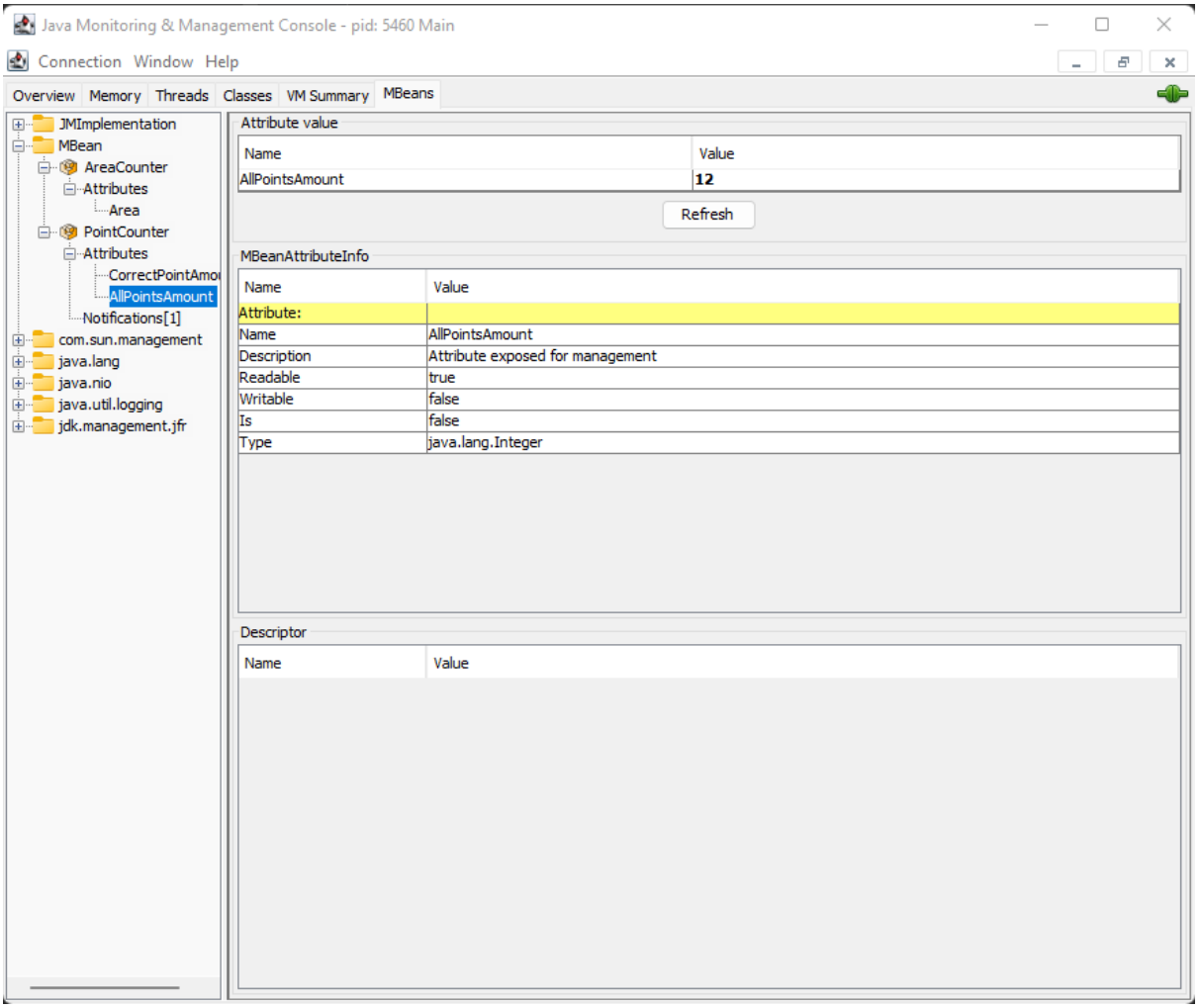
4. С помощью утилиты VisualVM и профилировщика IDE NetBeans, Eclipse или Idea локализовать и устранить проблемы с производительностью в программе. По результатам локализации и устранения проблемы необходимо составить отчёт, в котором должна содержаться следующая информация:

- Описание выявленной проблемы.
- Описание путей устранения выявленной проблемы.
- Подробное (со скриншотами) описание алгоритма действий, который позволил выявить и локализовать проблему.

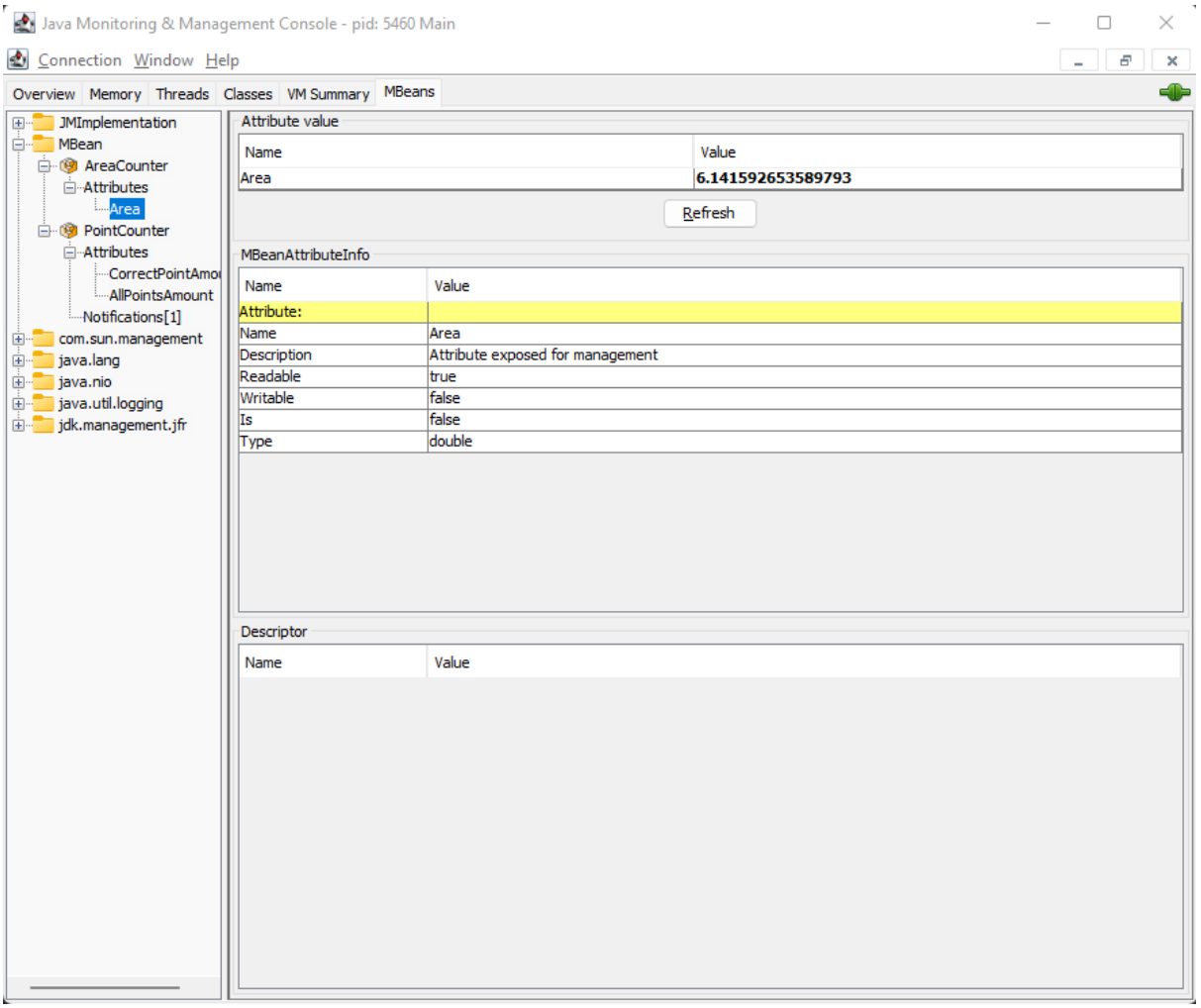
Задание 2

Смотреть следующую страницу

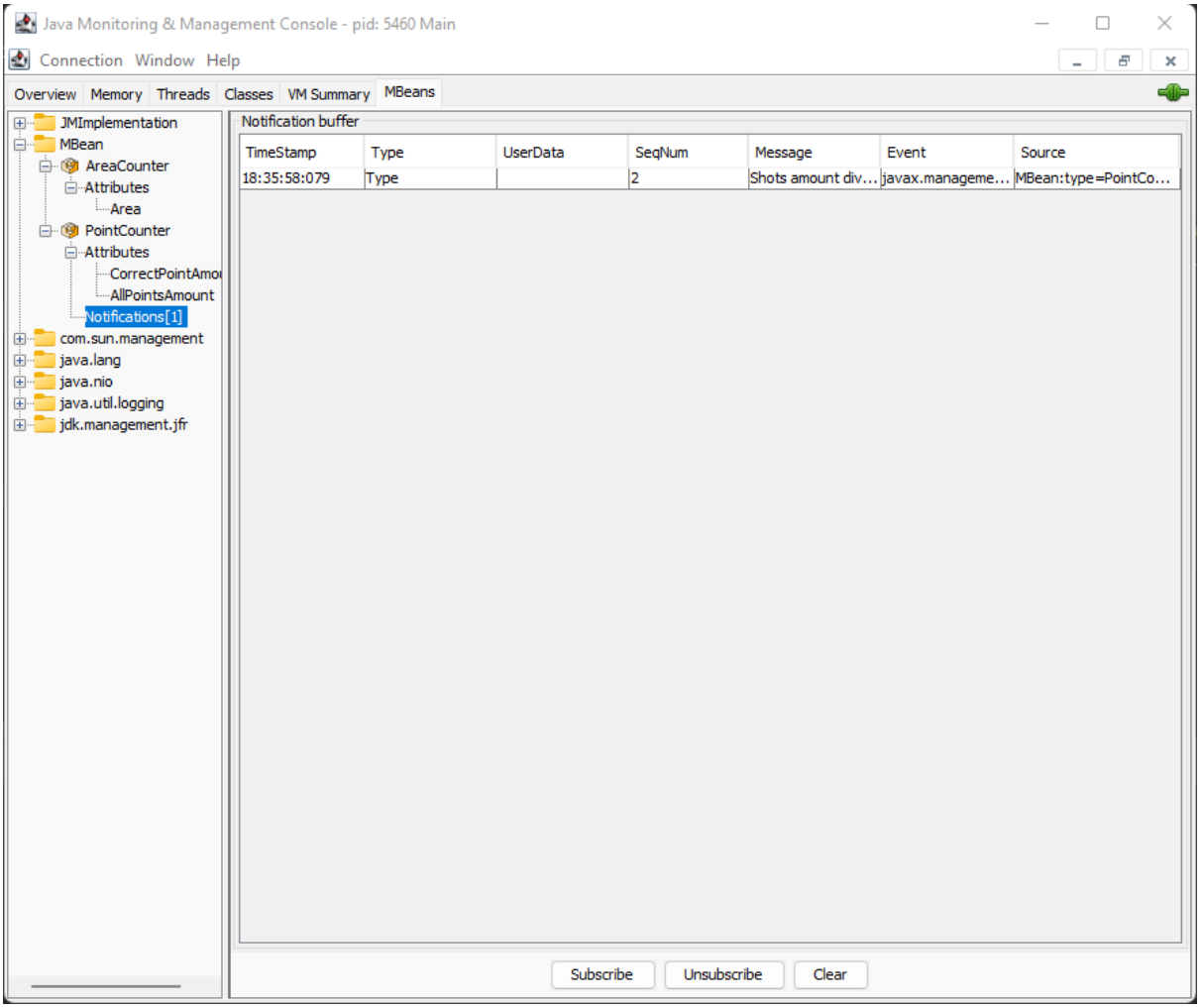
PointCounterBean



AreaCounterBean



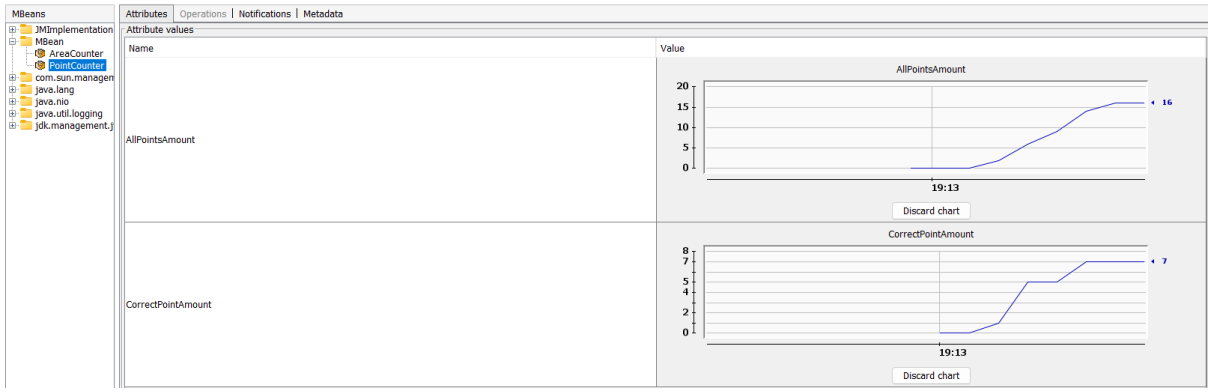
BeanMessage

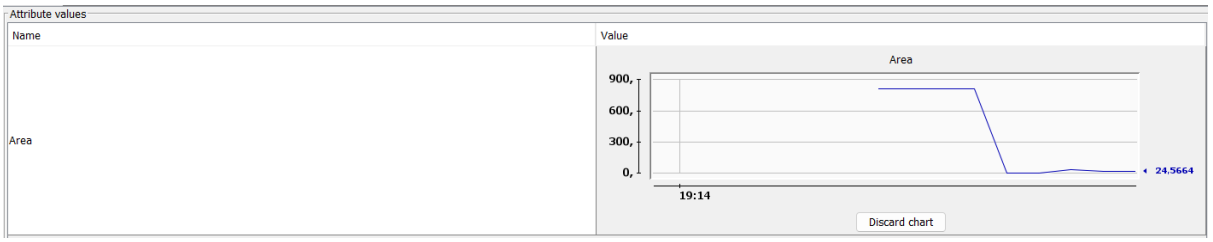


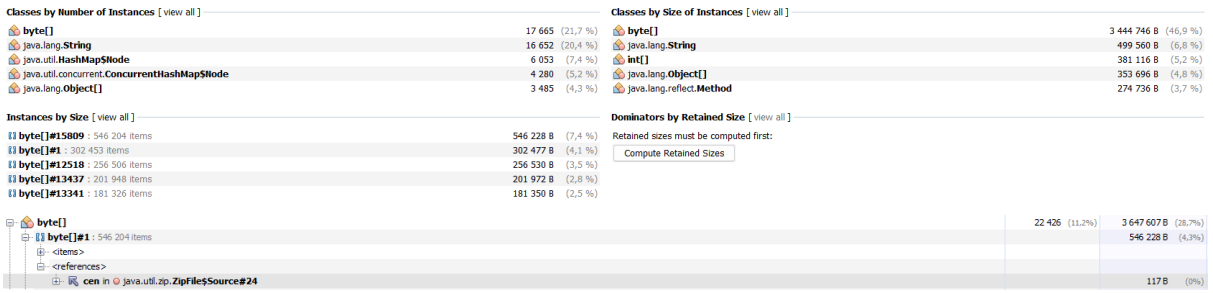
Classpath for JVM

Class path: C:\Users\Dmitriy\Desktop\lab3\target\classes;C:\Users\Dmitriy\.m2\repository\org\hibernate\hibernate-core\5.4.30.Final\hibernate-core-5.4.30.Final.jar;C:\Users\Dmitriy\.m2\repository\org\jboss\logging\jboss-logging\3.4.1.Final\jboss-logging-3.4.1.Final.jar;C:\Users\Dmitriy\.m2\repository\javax\persistence\javax.persistence-api\2.2\javax.persistence-api-2.2.jar;C:\Users\Dmitriy\.m2\repository\org\javassist\javassist\3.27.0-GA\javassist-3.27.0-GA.jar;C:\Users\Dmitriy\.m2\repository\net\bytebuddy\byte-buddy\1.10.21\byte-buddy-1.10.21.jar;C:\Users\Dmitriy\.m2\repository\antlr\antlr\2.7.7\antlr-2.7.7.jar;C:\Users\Dmitriy\.m2\repository\org\jboss\spec\javax\transaction\jboss-transaction-api_1.2_spec\1.1.1.Final\jboss-transaction-api_1.2_spec-1.1.1.Final.jar;C:\Users\Dmitriy\.m2\repository\org\jboss\jandex\2.2.3.Final\jandex-2.2.3.Final.jar;C:\Users\Dmitriy\.m2\repository\com\fasterxml\classmate\1.5.1\classmate-1.5.1.jar;C:\Users\Dmitriy\.m2\repository\javax\activation\javax.activation-api\1.2.0\javax.activation-api-1.2.0.jar;C:\Users\Dmitriy\.m2\repository\org\dom4j\dom4j\2.1.3\dom4j-2.1.3.jar;C:\Users\Dmitriy\.m2\repository\org\hibernate\common\hibernate-commons-annotations\5.1.2.Final\hibernate-commons-annotations-5.1.2.Final.jar;C:\Users\Dmitriy\.m2\repository\javax\xml\bind\jaxb-api\2.3.1\jaxb-api-2.3.1.jar;C:\Users\Dmitriy\.m2\repository\org\glassfish\jaxb\jaxb-runtime\2.3.1\jaxb-runtime-2.3.1.jar;C:\Users\Dmitriy\.m2\repository\org\glassfish\jaxb\txw2\2.3.1\txw2-2.3.1.jar;C:\Users\Dmitriy\.m2\repository\com\sun\istack\istack-commons-runtime\3.0.7\istack-commons-runtime-3.0.7.jar;C:\Users\Dmitriy\.m2\repository\org\jvnet\staxex\stax-ex\1.8\stax-ex-1.8.jar;C:\Users\Dmitriy\.m2\repository\com\sun\xml\fastinfoset\FastInfoset\1.2.15\FastInfoset-1.2.15.jar;C:\Users\Dmitriy\.m2\repository\org\hibernate\validator\hibernate-validator\6.2.0.Final\hibernate-validator-6.2.0.Final.jar;C:\Users\Dmitriy\.m2\repository\jakarta\validation\jakarta.validation-api\2.0.2\jakarta.validation-api-2.0.2.jar;C:\Users\Dmitriy\.m2\repository\org\thymeleaf\thymeleaf\3.0.12.RELEASE\thymeleaf-3.0.12.RELEASE.jar;C:\Users\Dmitriy\.m2\repository\ognl\ognl\3.1.26\ognl-3.1.26.jar;C:\Users\Dmitriy\.m2\repository\org\attoparser\attoparser\2.0.5.RELEASE\attoparser-2.0.5.RELEASE.jar;C:\Users\Dmitriy\.m2\repository\org\unbescape\unbescape\1.1.6.RELEASE\unbescape-1.1.6.RELEASE.jar;C:\Users\Dmitriy\.m2\repository\org\slf4j\slf4j-api\1.7.25\slf4j-api-1.7.25.jar;C:\Users\Dmitriy\.m2\repository\org\primefaces\primefaces\10.0.0\primefaces-10.0.0.jar;C:\Users\Dmitriy\.m2\repository\org\icefaces\icefaces\4.3.0\icefaces-4.3.0.jar;C:\Users\Dmitriy\.m2\repository\org\glassfish\javax.faces\2.3.5\javax.faces-2.3.5.jar;C:\Users\Dmitriy\.m2\repository\org\icefaces\icefaces-ace\4.3.0\icefaces-ace-4.3.0.jar;C:\Users\Dmitriy\.m2\repository\javax\faces\jsf-facelets\1.1.11a\jsf-facelets-1.1.11a.jar;C:\Users\Dmitriy\.m2\repository\com\sun\facelets\jsf-facelets\1.1.14\jsf-facelets-1.1.14.jar;C:\Users\Dmitriy\.m2\repository\com\oracle\ojdbc\ojdbc8\19.3.0.0\ojdbc8-19.3.0.0.jar;C:\Users\Dmitriy\.m2\repository\com\oracle\ojdbc\ucp\19.3.0.0\ucp-19.3.0.0.jar;C:\Users\Dmitriy\.m2\repository\com\oracle\ojdbc\oraclepki\19.3.0.0\oraclepki-19.3.0.0.jar;C:\Users\Dmitriy\.m2\repository\com\oracle\ojdbc\osdt_cert\19.3.0.0\osdt_cert-19.3.0.0.jar;C:\Users\Dmitriy\.m2\repository\com\oracle\ojdbc\osdt_core\19.3.0.0\osdt_core-19.3.0.0.jar;C:\Users\Dmitriy\.m2\repository\com\oracle\ojdbc\simplefan\19.3.0.0\simplefan-19.3.0.0.jar;C:\Users\Dmitriy\.m2\repository\com\oracle\ojdbc\ons\19.3.0.0\ons-19.3.0.0.jar;C:\Users\Dmitriy\.m2\repository\com\oracle\ojdbc\orai18n\19.3.0.0\orai18n-19.3.0.0.jar

Задание 3

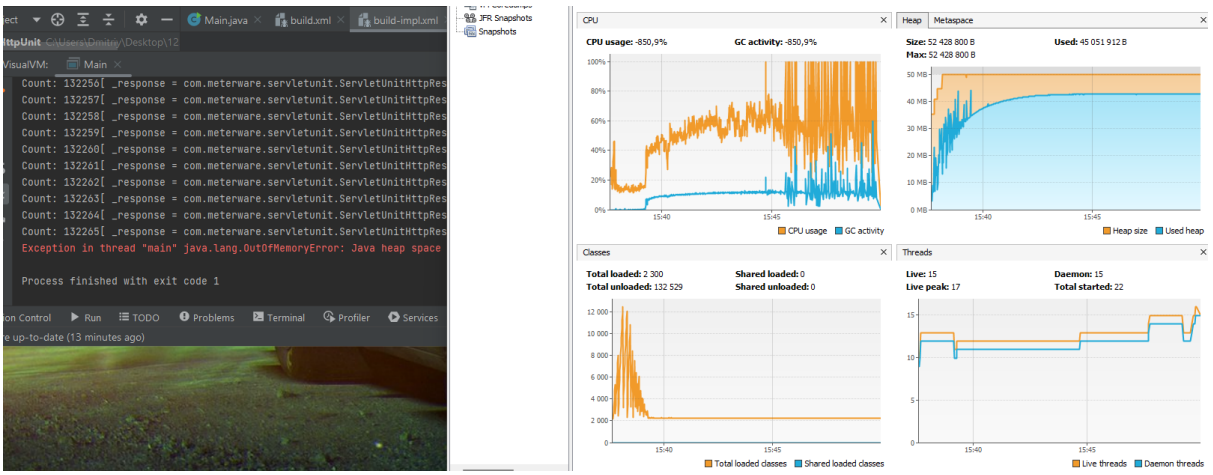






Обнаружение и устранение проблемы:

Попробуем поймать OutOfMemoryError, ограничив размер кучи до 50 Мб с запуском программы. Как видим, exception пойман.



Затем открываем heap dump на error и смотрим какие объекты больше всего занимали память. Эти объекты лежали в ArrayList в классе com.meterware.httpunit.javascript.Javascript.


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

```
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 );
    }
}
```

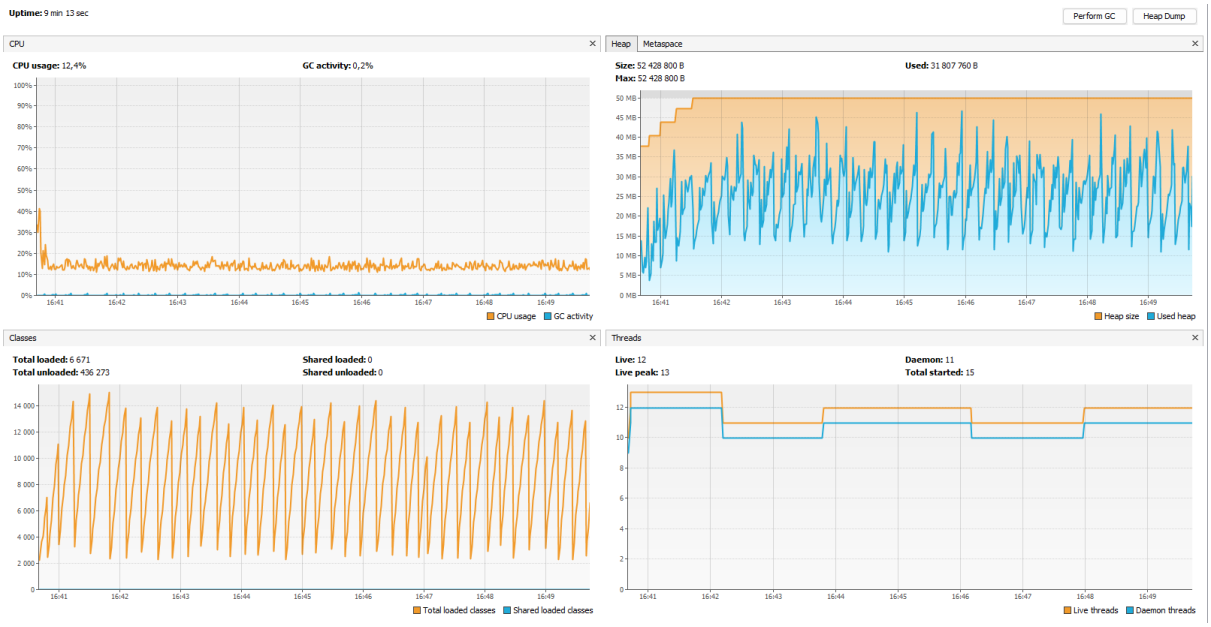
Получается, что лист ошибок не очищался и приводил к OutOfMemoryError. Попробуем найти метод для очистки листа и, вау, он есть.

```
1 usage
static void clearErrorMessages() {
    _errorMessages.clear();
}
```

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

Применяем его и смотрим, что программа не расходует больше 50 Мб кучи и GC работает в нормальном режиме.

```java
public static void main(String[] args) {
    try {
        HttpUnitOptions.setExceptionsThrownOnScriptError(false);
        ServletRunner sr = new ServletRunner();
        sr.registerServlet( resourceName: "myServlet", HelloWorld.class.getName());
        ServletUnitClient sc = sr.newClient();
        int number = 1;
        WebRequest request = new GetMethodWebRequest( urlString: "http://test.meterware.com/myServlet");
        while (true) {
            WebResponse response = sc.getResponse(request);
            System.out.println("Count: " + number++ + response);
            HttpUnitOptions.clearScriptErrorMessages();
              java.lang.Thread.sleep(200);
        }
    } catch (MalformedURLException ex) {
        Logger.getLogger( name: "global").log(Level.SEVERE,  msg: null, ex);
    } catch (IOException ex) {
        Logger.getLogger( name: "global").log(Level.SEVERE,  msg: null, ex);
    }
    catch (SAXException ex) {
        Logger.getLogger( name: "global").log(Level.SEVERE,  msg: null, ex);
    }
}
```



Вывод:

При выполнении данной лабораторной работы мы ознакомились с профилированием Java приложений. На примере создания MBean’ов для сбора статистики была изучена технология JMX. Также при помощи JMX Instrumentation (в данном случае JConsole) сняли показания статистики и изучили нагрузку, создаваемую приложением. С помощью VisualVM были выявлены проблемы в предложенной программе, а после этого в ней были устранены недочеты.