

Федеральное государственное автономное образовательное учреждение высшего
образования Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Основы программной инженерии

Лабораторная работа №4

Вариант № 1583

Выполнили: студенты группы Р3208,
Васильев Н. А., Петров В. М.

Преподаватель: Воронина Д. С.

Санкт-Петербург 2025

Текст задания

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 10, 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 количество классов, загруженных в JVM в процессе выполнения программы.

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.

Код программы

<https://github.com/kihort-si/opi-lab4>

Показания JConsole

Показания MBean-классов, разработанных в ходе выполнения задания 1:

Overview Memory Threads Classes VM Summary MBeans

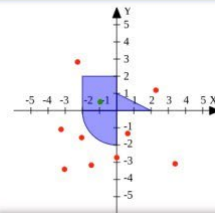
MBeanInfo

Name	Value
Info:	
ObjectName	org.viacheslav.beans.type=PointCounter,name=pointCounter
ClassName	org.viacheslav.beans.PointCounter
Description	Information on the management interface of the MBean
Constructor-0:	
Name	org.viacheslav.beans.PointCounter
Description	Public constructor of the MBean

Descriptor

Name	Value
Info:	
immutableInfo	false
interfaceClassName	org.viacheslav.beans.PointCounterMBean
mbean	false

Petrov and Vasiliev P3208 lab4 var.1583



Введите X:

0

Введите Y (-3 ... 5):

0.0

Введите R:

2

Площадь фигуры для последней точки: 8.14

Отправить Очистить

На главную
страничку

ID	X	Y	R	Попал?	Дата	Точку поставил я?
1054	3.4	-3.09	2.0	Нет	05.05.2025 06:45	Да
1055	0.64	1.33	2.0	Нет	05.05.2025	Да

Overview Memory Threads Classes VM Summary MBeans

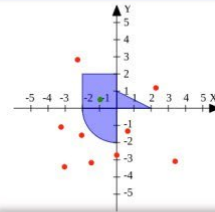
MBeanInfo

Name	Value
Info:	
ObjectName	org.viacheslav.beans.type=ShapeArea,name=shapeArea
ClassName	org.viacheslav.beans.ShapeArea
Description	Information on the management interface of the MBean
Constructor-0:	
Name	org.viacheslav.beans.ShapeArea
Description	Public constructor of the MBean

Descriptor

Name	Value
Info:	
immutableInfo	true
interfaceClassName	org.viacheslav.beans.ShapeAreaMBean
mbean	false

Petrov and Vasiliev P3208 lab4 var.1583



Введите X:

0

Введите Y (-3 ... 5):

0.0

Введите R:

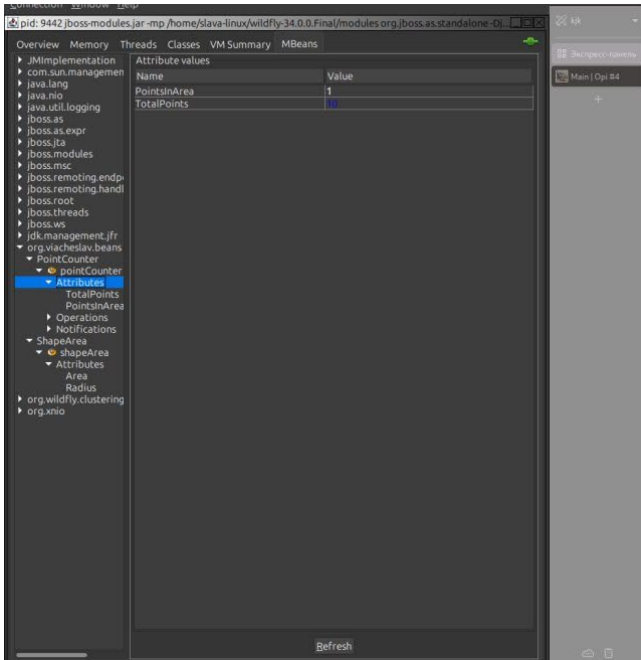
2

Площадь фигуры для последней точки: 8.14

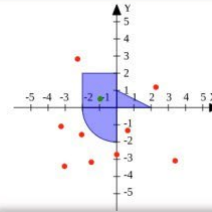
Отправить Очистить

На главную
страничку

ID	X	Y	R	Попал?	Дата	Точку поставил я?
1054	3.4	-3.09	2.0	Нет	05.05.2025 06:45	Да
1055	0.64	1.33	2.0	Нет	05.05.2025	Да



Petrov and Vasiliev P3208 lab4 var.1583



Введите X:

0

Введите Y (-3 ... 5):

0.0

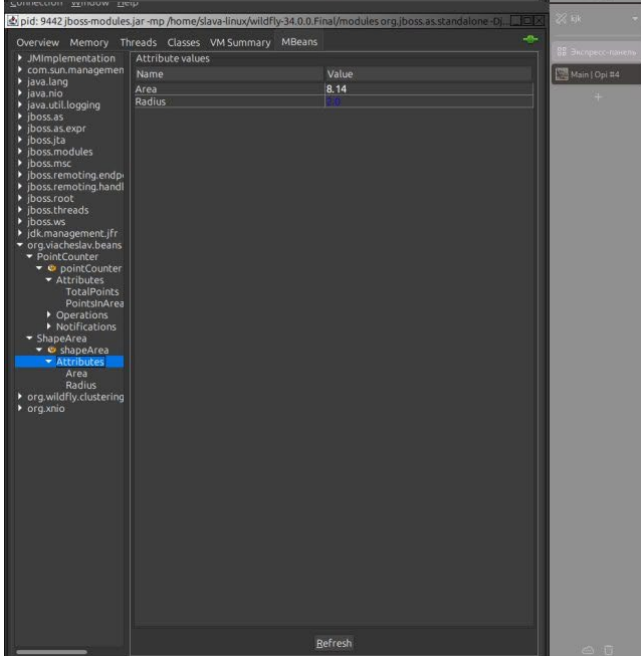
Введите R:

На главную
страничку

Площадь фигуры для последней точки: 8.14

Отправить Очистить

ID	X	Y	R	Попал?	Дата	Точку поставил я?
1054	3.4	-3.09	2.0	Нет	05.05.2025 06:45	Да
1055	0.64	1.33	2.0	Нет	05.05.2025	Да



Petrov and Vasiliev P3208 lab4 var.1583



На главную
страничку

Введите X:

0

Введите Y (-3 ... 5):

0.0

Введите R:

2

Площадь фигуры для последней точки: 8.14

Отправить Очистить

ID	X	Y	R	Попал?	Дата	Точку поставил я?
1054	3.4	-3.09	2.0	Нет	05.05.2025 06:45	Да
1055	0.64	1.33	2.0	Нет	05.05.2025	Да

Petrov and Vasiliev
P3208 lab4 var.1583

Введите X:

0

Введите Y (-3 ... 5):

0.0

Введите R:

2

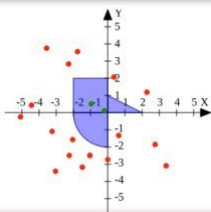
Площадь фигуры для последней точки: 8.14

Отправить Очистить

ID	X	Y	R	Попал?	Дата	Точку поставил?
На главную страницу	3.4	-3.09	2.0	Yes	05.05.2025 06:45	Yes

Petrov and Vasiliev

P3208 lab4 var.1583



На главную
страничку

Введите X:

0

Введите Y (-3 ... 5):

0.0

Введите R:

2

Площадь фигуры для последней точки: 8.14

Отправить

Очистить

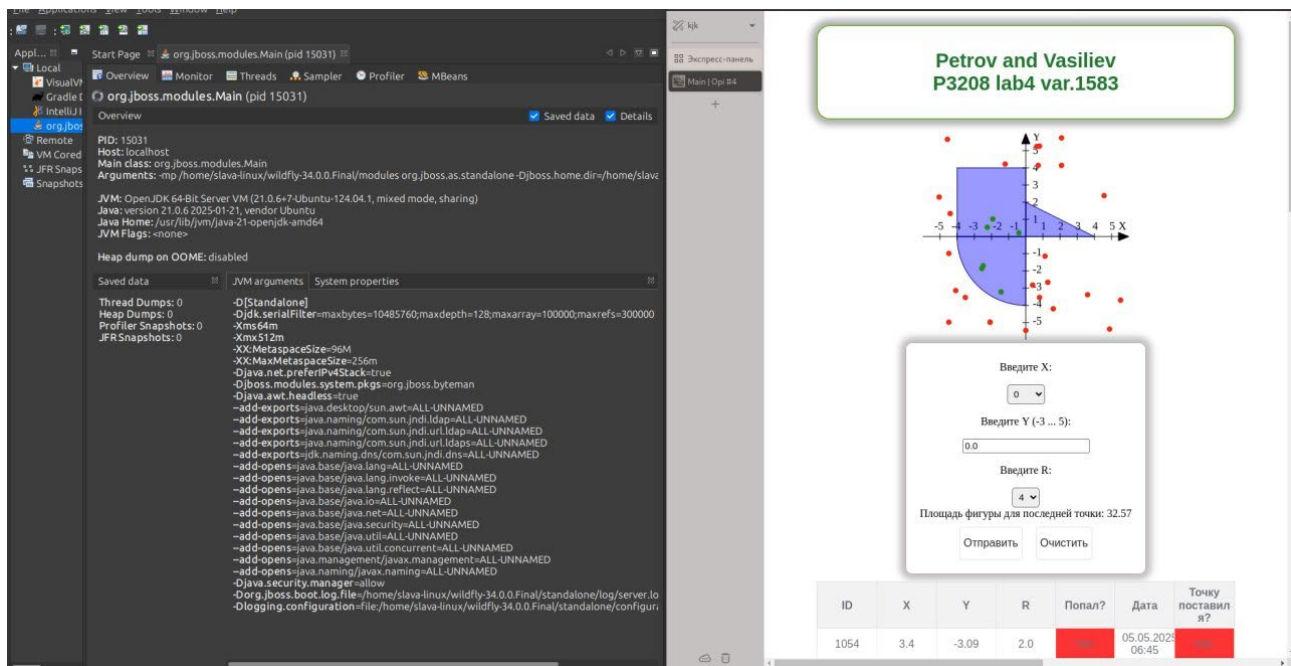
ID	X	Y	R	Попал?	Дата	Точку поставил?
1054	3.4	-3.09	2.0	Нет	05.05.2025 06:45	Да
					05.05.2025	

- В процессе мониторинга с применением утилиты JConsole было установлено следующее:
- Для определения времени работы JVM можно воспользоваться показателем Uptime в разделе VM Stats.
- MBeans с именами PointCounter и ShapeArea были успешно созданы и зарегистрированы в системе.
- MBean PointCounter настроен на отправку уведомлений при десяти попаданиях. Эти уведомления корректно отображаются в JConsole.

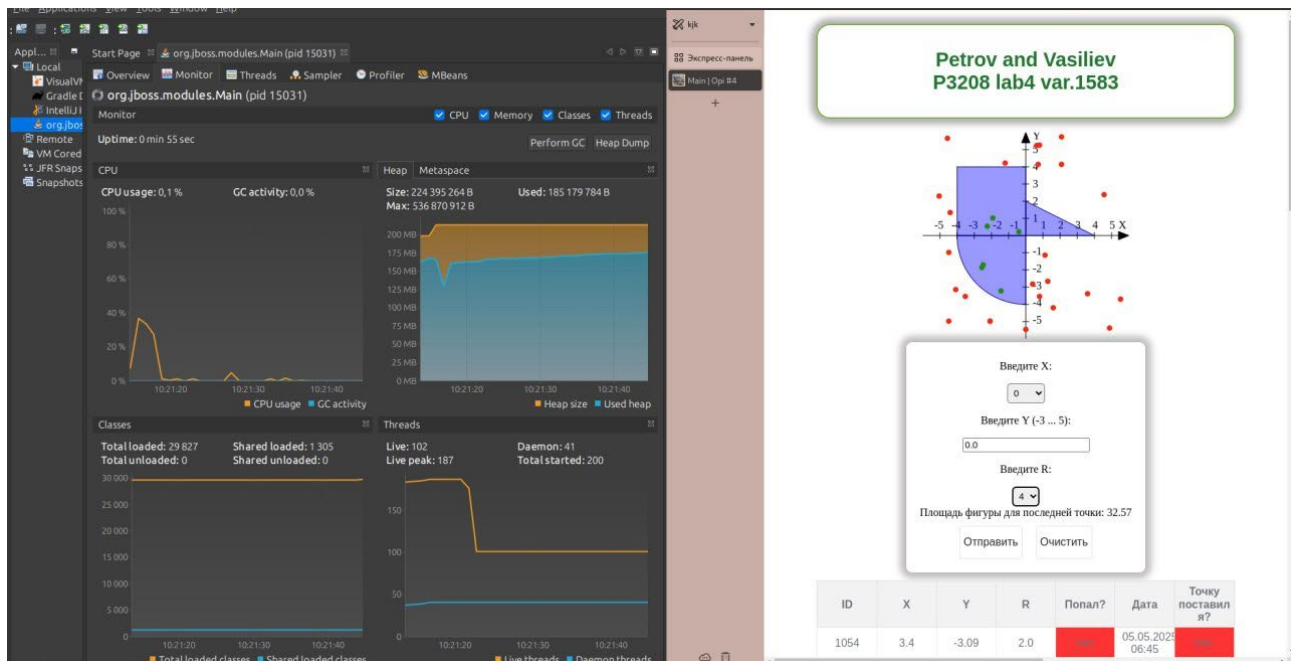
- Таким образом, механизм отправки и получения уведомлений от MBeans был успешно реализован и протестирован, что обеспечивает возможность своевременного реагирования на события — ключевой аспект эффективного мониторинга.

Показания VisualVM

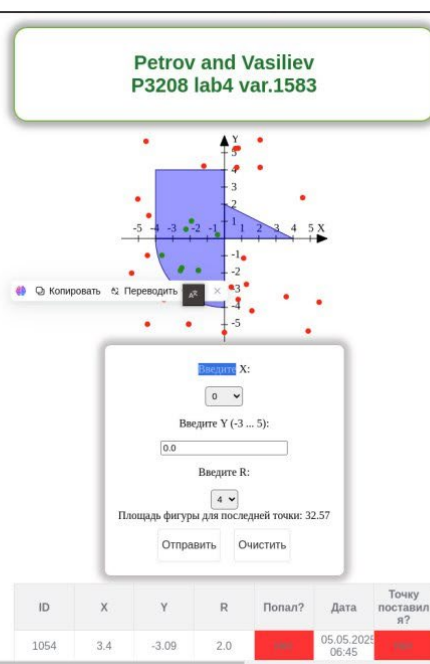
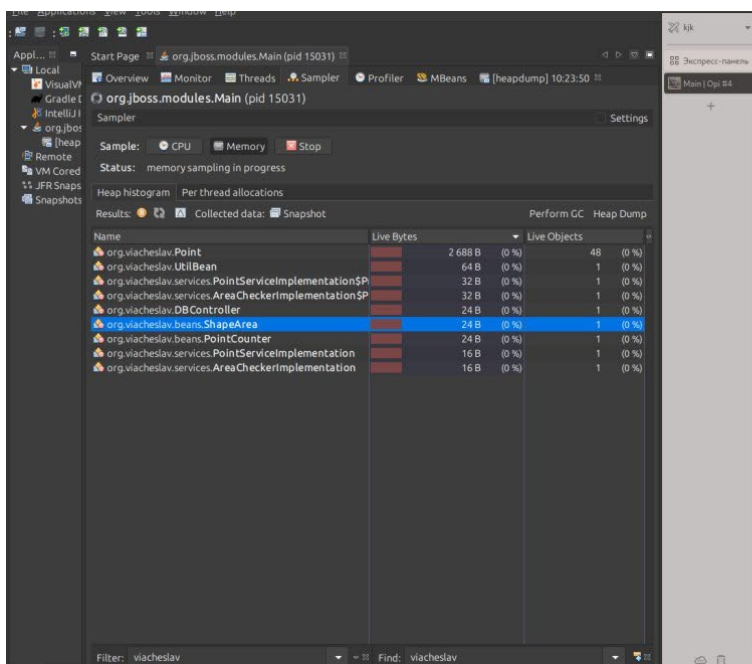
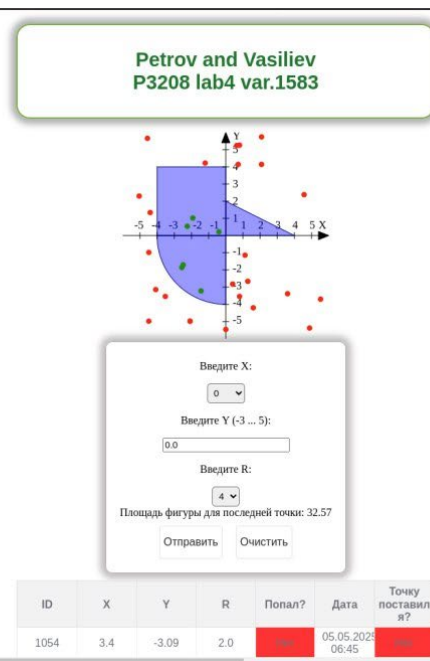
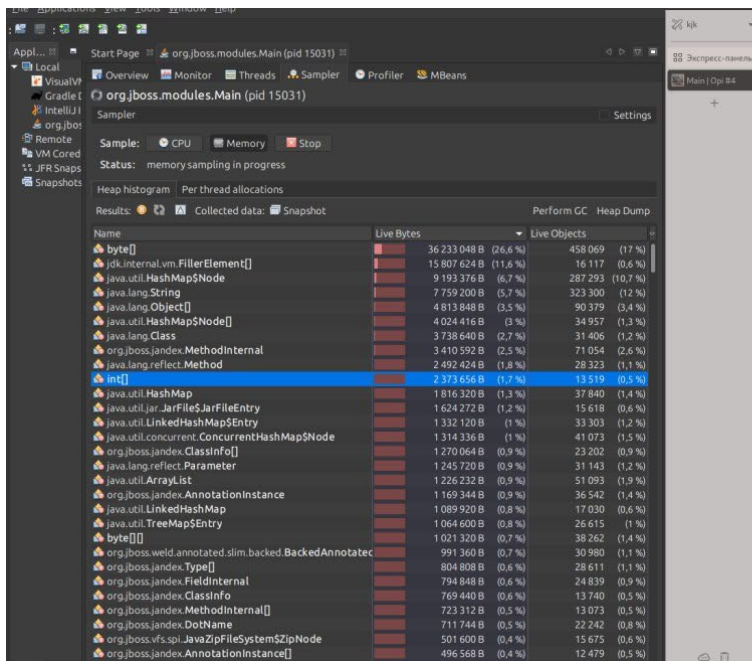
JVM Overview:



Мониторинг:



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



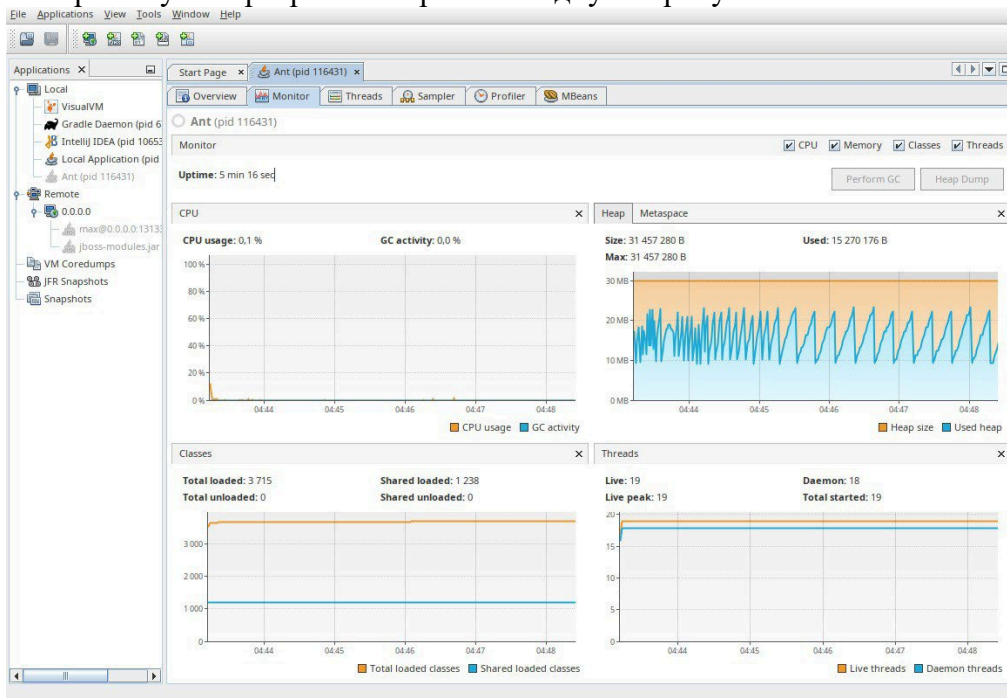
Исследования проблем с производительностью в программе HttpUnit

```
44      WebRequest request = new GetMethodWebRequest( urlString: "http://test.meterw
45      while (true) {
46          WebResponse response = sc.getResponse(request);
47          System.out.println("Count: " + number++ + response);
48          java.lang.Thread.sleep( millis: 200);
```

В ходе анализа производительности приложения была выявлена проблема, связанная с частыми вызовами метода `java.lang.Thread.sleep(200)`. Этот метод приостанавливает выполнение потока на 200 миллисекунд, что влечёт за собой необоснованные задержки и снижает общую эффективность работы системы.

Поскольку данный вызов не выполняет полезных действий и не влияет на логику работы приложения, его можно безопасно удалить. Это позволит устранить лишние паузы и повысить производительность.

Зададим максимальный размер кучи равным 30 МБ с помощью параметра `-Xmx30m`, указав его при запуске программы через командную строку.



```
[java] Count: 66234 [ _response = com.meterware.servletunit.ServletUnitHttpResponse@550059c0]
[java] Count: 66235 [ _response = com.meterware.servletunit.ServletUnitHttpResponse@1221b1d6]
[java] Count: 66236 [ _response = com.meterware.servletunit.ServletUnitHttpResponse@66f169dd]
[java] Count: 66237 [ _response = com.meterware.servletunit.ServletUnitHttpResponse@12b7b04d]
[java] Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
[java] at java.base/java.util.Arrays.copyOf(Arrays.java:3745)
[java] at java.base/java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractStringBuilder.java:172)
[java] at java.base/java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:538)
[java] at java.base/java.lang.StringBuilder.append(StringBuilder.java:174)
[java] at java.base/java.lang.StringBuilder.append(StringBuilder.java:168)
[java] at java.base/java.lang.Throwable.printStackTrace(Throwable.java:662)
```


С помощью Heap Dump найдем объекты, занимающие большую часть памяти.

Name	Count	Size	Retained (sort to get)
byte[]	28 785 (19,9 %)	3 240 768 B (38,3 %)	n/a
java.lang.String	27 622 (19,1 %)	662 928 B (7,8 %)	n/a
java.lang.String#1 : ug-Cyrl	1	24 B (0 %)	n/a
java.lang.String#2 : tg-Arab	1	24 B (0 %)	n/a
java.lang.String#3 : sw-Arab	1	24 B (0 %)	n/a
java.lang.String#4 : so-Arab	1	24 B (0 %)	n/a
java.lang.String#5 : shi-Latn	1	24 B (0 %)	n/a
java.lang.String#6 : sd-Sind	1	24 B (0 %)	n/a
java.lang.String#7 : sd-Khoj	1	24 B (0 %)	n/a
java.lang.String#8 : sd-Deva	1	24 B (0 %)	n/a
java.lang.String#9 : sat-Deva	1	24 B (0 %)	n/a
java.lang.String#10 : pa-Arab	1	24 B (0 %)	n/a
java.lang.String#11 : ms-Arab	1	24 B (0 %)	n/a
java.lang.String#12 : mni-Mtei	1	24 B (0 %)	n/a
java.lang.String#13 : mn-Mong	1	24 B (0 %)	n/a
java.lang.String#14 : ml-Arab	1	24 B (0 %)	n/a
java.lang.String#15 : ky-Latn	1	24 B (0 %)	n/a
java.lang.String#16 : ky-Arab	1	24 B (0 %)	n/a
java.lang.String#17 : ku-Arab	1	24 B (0 %)	n/a
java.lang.String#18 : ks-Deva	1	24 B (0 %)	n/a
java.lang.String#19 : kk-Arab	1	24 B (0 %)	n/a
java.lang.String#20 : ku-Latn	1	24 B (0 %)	n/a
java.lang.String#21 : ha-Arab	1	24 B (0 %)	n/a
java.lang.String#22 : ff-Arab	1	24 B (0 %)	n/a
java.lang.String#23 : ff-Adlm	1	24 B (0 %)	n/a
java.lang.String#24 : en-Shaw	1	24 B (0 %)	n/a
java.lang.String#25 : dyo-Arab	1	24 B (0 %)	n/a
java.lang.String#26 : bm-Nkoo	1	24 B (0 %)	n/a
java.lang.String#27 : az-Cyrl	1	24 B (0 %)	n/a
java.lang.String#28 : en-001	1	24 B (0 %)	n/a

Name	Count	Size	Retained (sort to get)
byte[]	28 785 (19,9 %)	3 240 768 B (38,3 %)	n/a
java.lang.String	27 622 (19,1 %)	662 928 B (7,8 %)	n/a
java.util.concurrent.ConcurrentHashMap\$Node	5 849 (4 %)	187 168 B (2,2 %)	n/a
java.lang.ref.WeakReference	5 397 (3,7 %)	172 704 B (2 %)	n/a
java.lang.Class[]	5 054 (3,5 %)	115 080 B (1,4 %)	n/a
java.lang.Object[]	5 039 (3,5 %)	287 952 B (3,4 %)	n/a
java.util.HashMap\$Node	4 584 (3,2 %)	146 688 B (1,7 %)	n/a
java.util.LinkedHashMap\$Entry	3 853 (2,7 %)	154 120 B (1,8 %)	n/a
java.lang.ref.SoftReference	2 843 (2 %)	113 720 B (1,3 %)	n/a
java.lang.reflect.Method	2 763 (1,9 %)	243 144 B (2,9 %)	n/a
java.lang.Class\$ReflectionData	2 596 (1,8 %)	166 144 B (2 %)	n/a

Исследовав кучу, находим повторяющиеся строки:

```
java.lang.Object[]#3431 : 106 710 items
<items>
[0] = java.lang.String#4633 : Script 'document.wr("Hello Document")' failed: TypeError: undefined is not a function. (httpunit;)
[1] = java.lang.String#4632 : Script 'document.wr("Hello Document")' failed: TypeError: undefined is not a function. (httpunit;)
[2] = java.lang.String#4631 : Script 'document.wr("Hello Document")' failed: TypeError: undefined is not a function. (httpunit;)
[3] = java.lang.String#4630 : Script 'document.wr("Hello Document")' failed: TypeError: undefined is not a function. (httpunit;)
[4] = java.lang.String#4629 : Script 'document.wr("Hello Document")' failed: TypeError: undefined is not a function. (httpunit;)
[5] = java.lang.String#4628 : Script 'document.wr("Hello Document")' failed: TypeError: undefined is not a function. (httpunit;)

java.lang.Object[]#3431 : 106 710 items
<items>
<references>
  elementData in java.util.ArrayList#24 : 82 384 elements
  static _errorMessages in class com.meterware.httpunit.javascript.JavaScript : JavaScript
```

Объекты _errorMessages хранятся в ArrayList

```

40     public class JavaScript {
46         private static ArrayList _errorMessages = new ArrayList();
47
48

```

Найдем строчку с добавлением объектов в этот список:

```

        throw new ScriptException( errorMessage );
    } else {
        _errorMessages.add( errorMessage );
    }

```

В результате получается накопление _errorMessages в списке, за счет чего и получается переполнение памяти. В программе есть функция для очистки _errorMessage, однако мы можем увидеть, что на самом деле она не используется

```

static void clearErrorMessages() {
    _errorMessages.clear();
}

public void clearErrorMessages() {
    JavaScript.clearErrorMessages();
}

```

```

/**
 * Clears the accumulated script error messages.
 */
no usages
public static void clearScriptErrorMessages() {
    getScriptingEngine().clearErrorMessages();
}

```

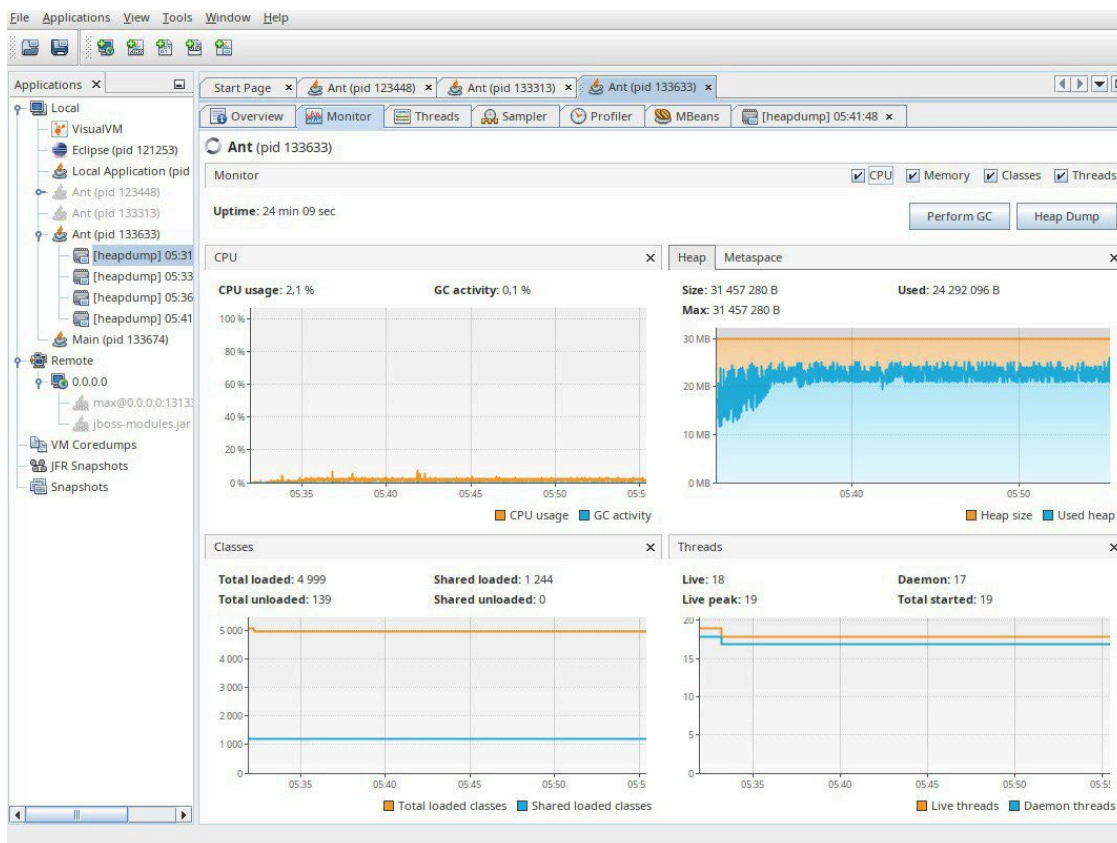
Решением будет очистка списка с _errorMessage после выполнения очередного запроса.

```

while (true) {
    WebResponse response = sc.getResponse(request);
    System.out.println("Count: " + number++ + response);
    HttpUnitOptions.clearScriptErrorMessages();
}

```

Запустим программу. Теперь изменения памяти во времени менее пилообразные, GC работает более оптимально.



Программа работает стабильно и не падает с OutOfMemoryException.

```
[java] Count: 2005303[ _response = com.meterware.servletunit.ServletUnitHttpResponse@46069bbd]
[java] Count: 2005304[ _response = com.meterware.servletunit.ServletUnitHttpResponse@32dd0028]
[java] Count: 2005305[ _response = com.meterware.servletunit.ServletUnitHttpResponse@75b8eb36]
[java] Count: 2005306[ _response = com.meterware.servletunit.ServletUnitHttpResponse@64a8bc94]
[java] Count: 2005307[ _response = com.meterware.servletunit.ServletUnitHttpResponse@4f95f8cb]
[java] Count: 2005308[ _response = com.meterware.servletunit.ServletUnitHttpResponse@16de43aa]
[java] Count: 2005309[ _response = com.meterware.servletunit.ServletUnitHttpResponse@3bd5b9b4]
[java] Count: 2005310[ _response = com.meterware.servletunit.ServletUnitHttpResponse@1858b57]
[java] Count: 2005311[ _response = com.meterware.servletunit.ServletUnitHttpResponse@676e7d4c]
[java] Count: 2005312[ _response = com.meterware.servletunit.ServletUnitHttpResponse@75275ed6]
[java] Count: 2005313[ _response = com.meterware.servletunit.ServletUnitHttpResponse@262f5cb1]
[java] Count: 2005314[ _response = com.meterware.servletunit.ServletUnitHttpResponse@5b40a9c5]
[java] Count: 2005315[ _response = com.meterware.servletunit.ServletUnitHttpResponse@11bd03de]
[java] Count: 2005316[ _response = com.meterware.servletunit.ServletUnitHttpResponse@7958beba]
[java] Count: 2005317[ _response = com.meterware.servletunit.ServletUnitHttpResponse@4f01dffcc]
[java] Count: 2005318[ _response = com.meterware.servletunit.ServletUnitHttpResponse@28f0ac3e]
[java] Count: 2005319[ _response = com.meterware.servletunit.ServletUnitHttpResponse@3ed2ae5f]
[java] Count: 2005320[ _response = com.meterware.servletunit.ServletUnitHttpResponse@398504c8]
[java] Count: 2005321[ _response = com.meterware.servletunit.ServletUnitHttpResponse@191d03aa]
[java] Count: 2005322[ _response = com.meterware.servletunit.ServletUnitHttpResponse@c9c3b58]
[java] Count: 2005323[ _response = com.meterware.servletunit.ServletUnitHttpResponse@788895bf]
[java] Count: 2005324[ _response = com.meterware.servletunit.ServletUnitHttpResponse@4360d585]
[java] Count: 2005325[ _response = com.meterware.servletunit.ServletUnitHttpResponse@7f1a46e8]
[java] Count: 2005326[ _response = com.meterware.servletunit.ServletUnitHttpResponse@61700b14]
[java] Count: 2005327[ _response = com.meterware.servletunit.ServletUnitHttpResponse@1167d8d]
[java] Count: 2005328[ _response = com.meterware.servletunit.ServletUnitHttpResponse@35ba37dd]
[java] Count: 2005329[ _response = com.meterware.servletunit.ServletUnitHttpResponse@4c817ff8]
[java] Count: 2005330[ _response = com.meterware.servletunit.ServletUnitHttpResponse@1cd7ed48]
[java] Count: 2005331[ _response = com.meterware.servletunit.ServletUnitHttpResponse@9fa89e6]
[java] Count: 2005332[ _response = com.meterware.servletunit.ServletUnitHttpResponse@5b5d2574]
[java] Count: 2005333[ _response = com.meterware.servletunit.ServletUnitHttpResponse@33043228]
[java] Count: 2005334[ _response = com.meterware.servletunit.ServletUnitHttpResponse@27d2a548]
[java] Count: 2005335[ _response = com.meterware.servletunit.ServletUnitHttpResponse@1344b5f5]
[java] Count: 2005336[ _response = com.meterware.servletunit.ServletUnitHttpResponse@3d23abd6]
[java] Count: 2005337[ _response = com.meterware.servletunit.ServletUnitHttpResponse@1526f7de]
[java] Count: 2005338[ _response = com.meterware.servletunit.ServletUnitHttpResponse@74c32dcc]
[java] Count: 2005339[ _response = com.meterware.servletunit.ServletUnitHttpResponse@263aa202]
[java] Count: 2005340[ _response = com.meterware.servletunit.ServletUnitHttpResponse@40c4416b]
```

Вывод

Во время выполнения лабораторной работы я познакомился с практикой написания MBeans в веб-приложениях, были изучены утилиты для мониторинга и профилирования работы программы JConsole и VisualVM, а также был получен опыт по полученным данным определять утечки памяти и устранять их.