

Федеральное государственное автономное образовательное
учреждение высшего образования
«Национальный исследовательский университет ИТМО»
Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №4 по дисциплине
«Основы программной инженерии»
Вариант № 248311

Студенты:
Сиразетдинов А.Н.,
Шпинева У. С.
Группа: Р3216
Преподаватель:
Письмак А.Е.

г. Санкт-Петербург
2023/24

Задание	3
MBeans	4
JConsole	6
VisualVM:	8
Устранение проблем с производительностью в программе	10
Вывод	14

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, 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 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 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.

MBeans

```
1 implementation  👤 Azat222
public interface PointHandlerMBean {
    1 implementation  👤 Azat222
    long getDotsCount();
    1 implementation  👤 Azat222
    long getSuccessDotsCount();
}
|
```

```
@Named
@ApplicationScoped
public class PointHandler extends NotificationBroadcasterSupport implements Serializable, PointHandlerMBean {

    private Point point = new Point();
    private LinkedList<Point> points = new LinkedList<>();
    private long sequenceNumber = 1;

    👤 Azat222
    public LinkedList<Point> getPoints() { return points; }

    👤 Azat222
    public void init(@Observes @Initialized(ApplicationScoped.class) Object unused) {
        this.points = DatabaseHandler.getDatabaseManager().loadCollection();
        MBeanRegistryUtil.registerBean(bean: this, name: "main");
    }

    👤 Azat222
    public void destroy(@Observes @Destroyed(ApplicationScoped.class) Object unused) {
        MBeanRegistryUtil.unregisterBean(this);
    }
}
```

```
👤 Azat222
@Override
public long getDotsCount() { return this.points.size(); }

👤 Azat222
@Override
public long getSuccessDotsCount() {
    return this.points.stream()
        .filter(Point::getStatus)
        .count();
}
```

```

1 implementation  Azat222
public interface AreaSquareMBean {
    1 implementation  Azat222
    double getSquare();
}

```

```

Azat222
@Named
@ApplicationScoped
public class AreaSquare implements AreaSquareMBean {
    @Inject
    private PointHandler pointHandler;

    Azat222
    public void init(@Observes @Initialized(ApplicationScoped.class) Object unused) {
        MBeanRegistryUtil.registerBean(bean: this, name: "areasquare");
    }

    Azat222
    public void destroy(@Observes @Destroyed(ApplicationScoped.class) Object unused) {
        MBeanRegistryUtil.unregisterBean(this);
    }

    Azat222
    @Override
    public double getSquare() {
        double r = pointHandler.getPoint().getR();
        return r * (r / 2) + 0.5f * (r / 2) * (r / 2) + 0.25f * Math.PI * r * r;
    }
}

```

JConsole

Overview Memory Threads Classes VM Summary MBeans

JMImplementation
com.sun.management
java.lang
java.nio
java.util.logging
jboss.as
jboss.as.expr
jboss.jta
jboss.modules
jboss.msc
jboss.remoting.endpoint
jboss.remoting.handler
jboss.root
jboss.threads
jboss.ws
jdk.management.jfr
models
 SquareBean
 square
 Attributes
 UserBean
 main
 Attributes
 Notifications
org.xnio

Attribute values

Name	Value
RequestsNumber	4
SuccessRequestsNumber	2

Refresh

Overview Memory Threads Classes VM Summary MBeans

JMImplementation
com.sun.management
java.lang
java.nio
java.util.logging
jboss.as
jboss.as.expr
jboss.jta
jboss.modules
jboss.msc
jboss.remoting.endpoint
jboss.remoting.handler
jboss.root
jboss.threads
jboss.ws
jdk.management.jfr
models
 SquareBean
 square
 Attributes
 UserBean
 main
 Attributes
 Notifications
org.xnio

Attribute values

Name	Value
Square	13.015552611167399

Refresh

OverviewMemoryThreadsClassesVM SummaryMBeans

[-] JMImplementation

[-] com.sun.management

[-] java.lang

[-] java.nio

[-] java.util.logging

[-] jboss.as

[-] jboss.as.expr

[-] jboss.jta

[-] jboss.modules

[-] jboss.msc

[-] jboss.remoting.endpoint

[-] jboss.remoting.handler

[-] jboss.root

[-] jboss.threads

[-] jboss.ws

[-] jdk.management.jfr

[-] models

[-] SquareBean

[-] square

[-] Attributes

[-] UserBean

[-] main

[-] Attributes

[-] Notifications[2]

[-] org.xmlio

Notification buffer

TimeStamp	Type	UserData	SeqNum	Message	Event	Source
23:55:09:288	Out of area		2	The point (2.441, -3.794) is out of area.	javax.management.Notification(sourc...	UserBean
23:55:08:265	Out of area		1	The point (-3.0, 3.206) is out of area.	javax.management.Notification(sourc...	UserBean

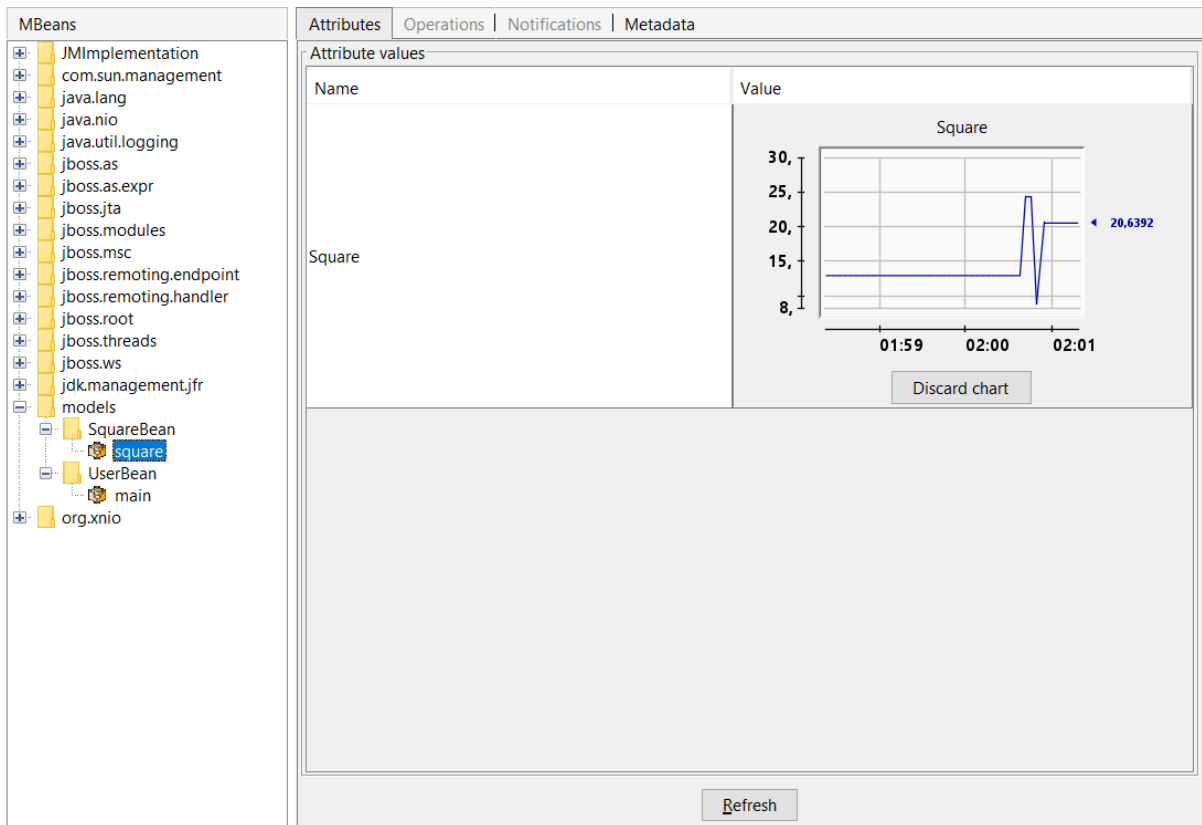
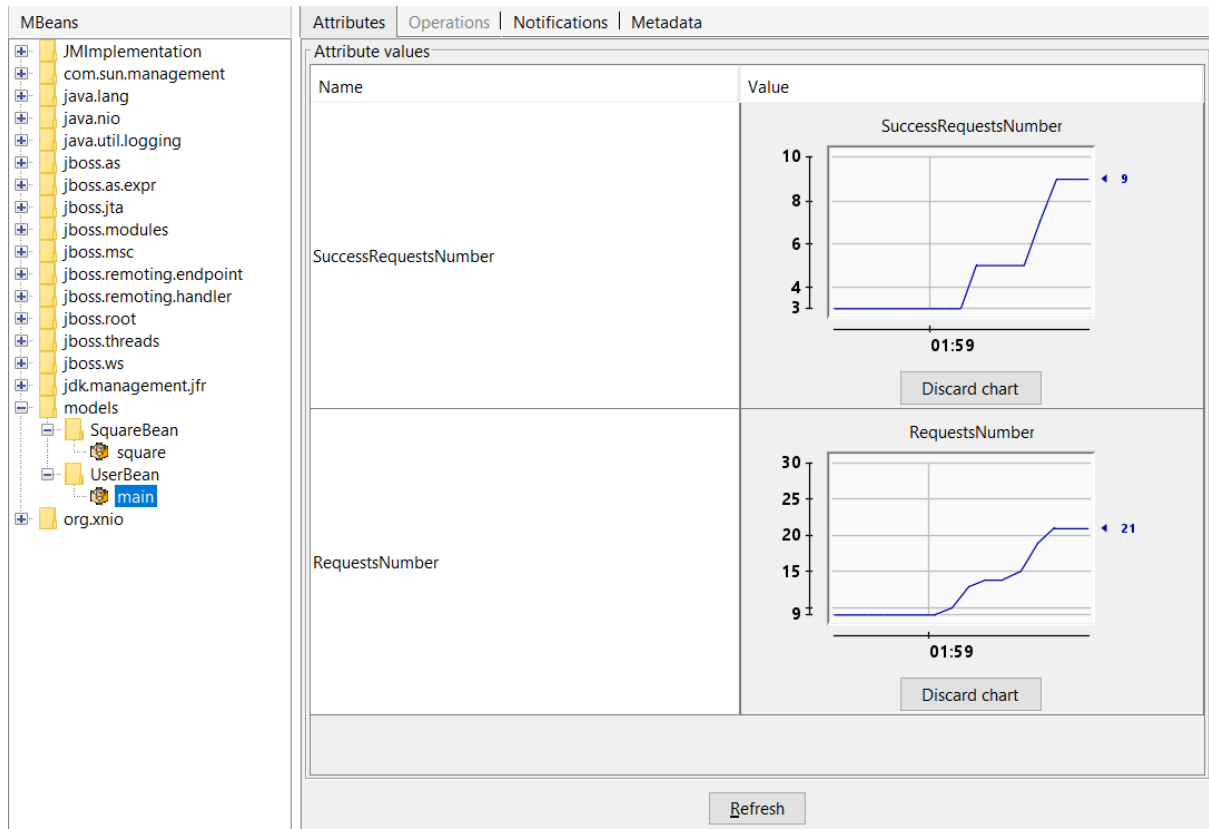
Subscribe






































































Unsubscribe

Clear

Class path: C:\programs\wildfly-30.0.0.Final\jboss-modules.jar

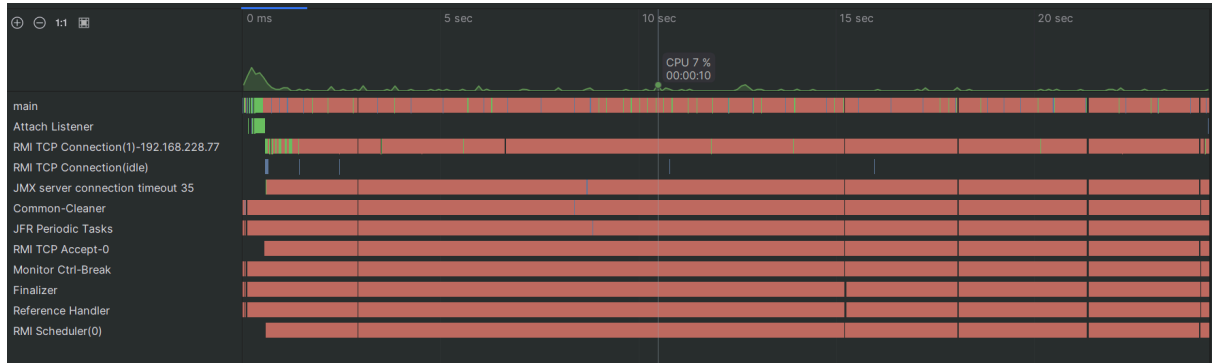
VisualVM:



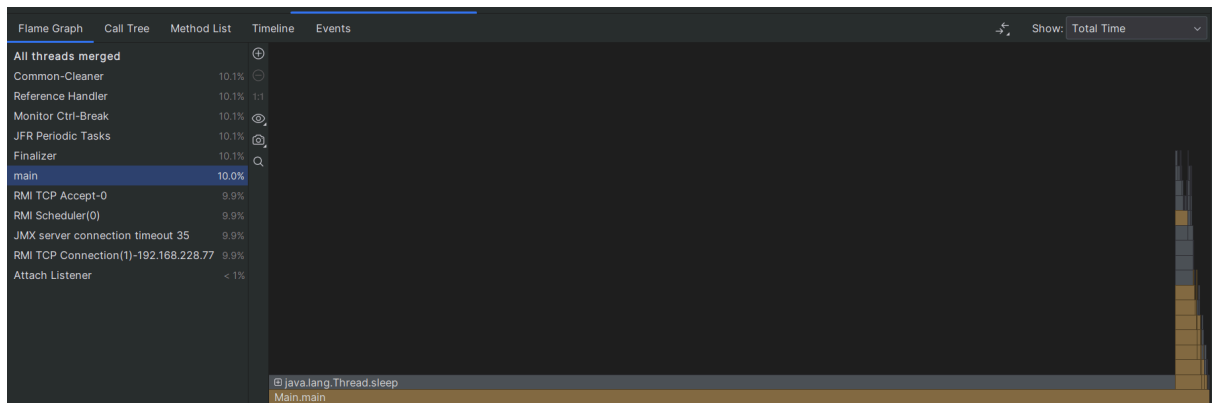
CPU samples		Thread CPU time	
Results:   		Statistics: Threads Count: 82 Total Time (CPU): 24 921 ms	
		Thread Dump	
Name	Thread Time (CPU)	Thread Time (CPU) / sec	
 RMI TCP Connection(15)-192.168.1.7	 4 203 ms (16,9 %)	 0,0 ms (0 %)	^
 RMI TCP Connection(16)-192.168.1.7	 3 875 ms (15,5 %)	 62,5 ms (6,3 %)	
 RMI TCP Accept-0	 1 531 ms (6,1 %)	 0,0 ms (0 %)	
 MSC service thread 1-1	 1 203 ms (4,8 %)	 0,0 ms (0 %)	
 MSC service thread 1-2	 1 171 ms (4,7 %)	 0,0 ms (0 %)	
 JFR Periodic Tasks	 1 171 ms (4,7 %)	 0,0 ms (0 %)	
 DestroyJavaVM	 1 031 ms (4,1 %)	 0,0 ms (0 %)	
 RMI TCP Connection(17)-192.168.1.7	 1 031 ms (4,1 %)	 0,0 ms (0 %)	
 MSC service thread 1-3	 984 ms (3,9 %)	 0,0 ms (0 %)	
 default task-2	 984 ms (3,9 %)	 0,0 ms (0 %)	
 MSC service thread 1-6	 968 ms (3,9 %)	 0,0 ms (0 %)	
 MSC service thread 1-5	 937 ms (3,8 %)	 0,0 ms (0 %)	
 MSC service thread 1-8	 843 ms (3,4 %)	 0,0 ms (0 %)	
 MSC service thread 1-7	 812 ms (3,3 %)	 0,0 ms (0 %)	
 MSC service thread 1-4	 656 ms (2,6 %)	 0,0 ms (0 %)	
 DeploymentScanner-threads - 2	 593 ms (2,4 %)	 0,0 ms (0 %)	
 Attach Listener	 562 ms (2,3 %)	 0,0 ms (0 %)	
 DeploymentScanner-threads - 1	 562 ms (2,3 %)	 0,0 ms (0 %)	
 ServerService Thread Pool -- 26	 281 ms (1,1 %)	 0,0 ms (0 %)	
 default task-1	 250 ms (1 %)	 0,0 ms (0 %)	
 management I/O-1	 171 ms (0,7 %)	 0,0 ms (0 %)	
 RMI TCP Connection(idle)	 171 ms (0,7 %)	 0,0 ms (0 %)	✓

Устранение проблем с производительностью в программе

1) Для начала запустим программу используя профилировщик IntelliJ Profiler



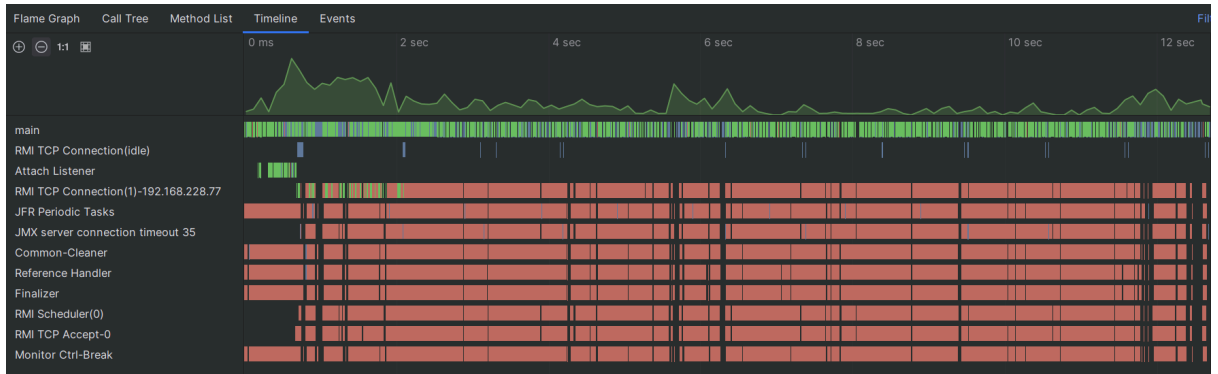
Обратим внимание на то, что процессор большую часть времени занят на 0-1%, а main процесс находится в состоянии WAITING



С помощью диаграммы времени работы процессора найдем участок в коде, в котором вызывается Thread.sleep

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

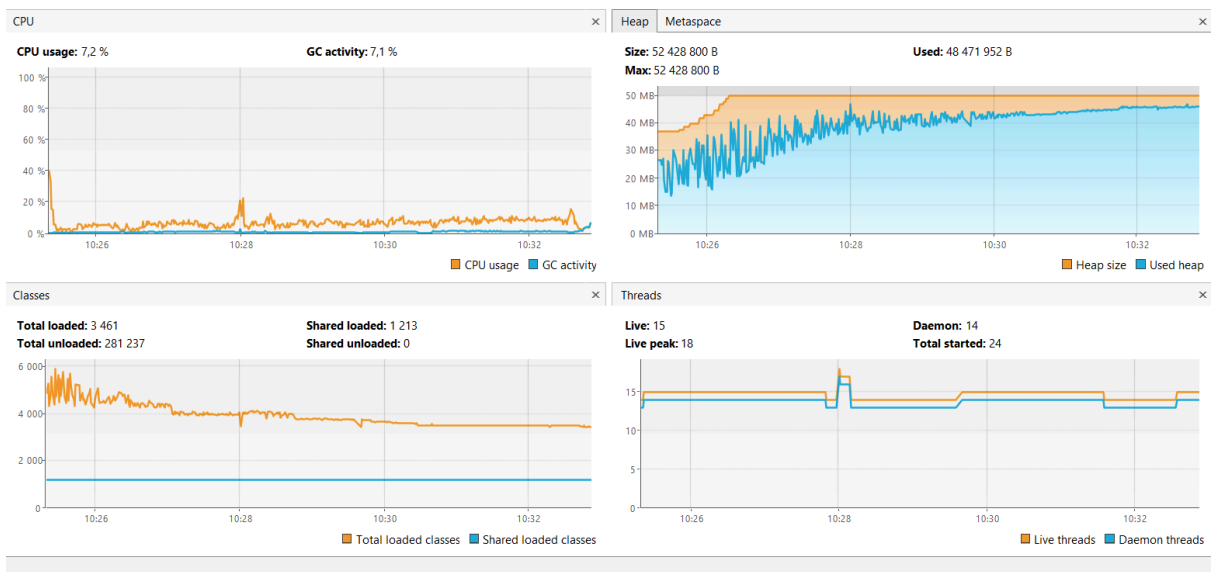
Закомментируем Thread.sleep который сильно замедляет работу программы



Теперь main процесс занят большую часть времени

2) Проверки на утечки памяти

Для начала установим ограничение в размере максимального размера кучи на 50 МБ с помощью флага `-Xmx50m`

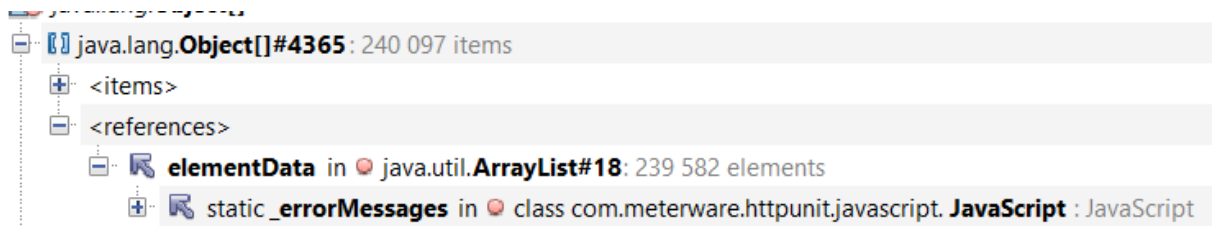


Обратим внимание на график Heap Size. Он неумолимо растет вверх, что говорит об утечке памяти.

Чтобы найти место утечки памяти сделаем Heap Dump

Name	Count	Size	Retained (sort to get)
byte[]	263 660 (43,8 %)	30 351 440 B (74,3 %)	n/a
java.lang.String	262 724 (43,6 %)	6 305 376 B (15,4 %)	n/a
java.lang.Object[]	4 477 (0,7 %)	1 236 792 B (3 %)	n/a
java.lang.Object[]#4365: 240 097 items		960 408 B (2,4 %)	n/a
java.lang.Object[]#4368: 5 120 items		20 496 B (0,1 %)	n/a
java.lang.Object[]#4008: 2 126 items		8 520 B (0 %)	n/a
java.lang.Object[]#3328: 1 267 items		5 088 B (0 %)	n/a
java.lang.Object[]#1518: 1 178 items		4 728 B (0 %)	n/a
java.lang.Object[]#3327: 1 056 items		4 240 B (0 %)	n/a
java.lang.Object[]#568: 1 024 items		4 112 B (0 %)	n/a
java.lang.Object[]#3338: 810 items		3 256 B (0 %)	n/a
java.lang.Object[]#2909: 668 items		2 688 B (0 %)	n/a
java.lang.Object[]#3211: 574 items		2 312 B (0 %)	n/a
java.lang.Object[]#1212: 549 items		2 216 B (0 %)	n/a
java.lang.Object[]#32: 388 items		1 568 B (0 %)	n/a
java.lang.Object[]#2013: 301 items		1 224 B (0 %)	n/a
java.lang.Object[]#1357: 293 items		1 192 B (0 %)	n/a
java.lang.Object[]#2005: 272 items		1 104 B (0 %)	n/a
java.lang.Object[]#270: 252 items		1 024 B (0 %)	n/a
java.lang.Object[]#77: 250 items		1 016 B (0 %)	n/a
java.lang.Object[]#7: 242 items		984 B (0 %)	n/a
java.lang.Object[]#2021: 225 items		920 B (0 %)	n/a
java.lang.Object[]#2022: 211 items		864 B (0 %)	n/a
java.lang.Object[]#3006: 207 items		848 B (0 %)	n/a
java.lang.Object[]#84: 206 items		840 B (0 %)	n/a

Во вкладке Objects найдем массив объектов содержащий 241K элементов



Откроем этот класс

```
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);
    }
}
```

В методе handleScriptException мы добавляем сообщение об ошибке

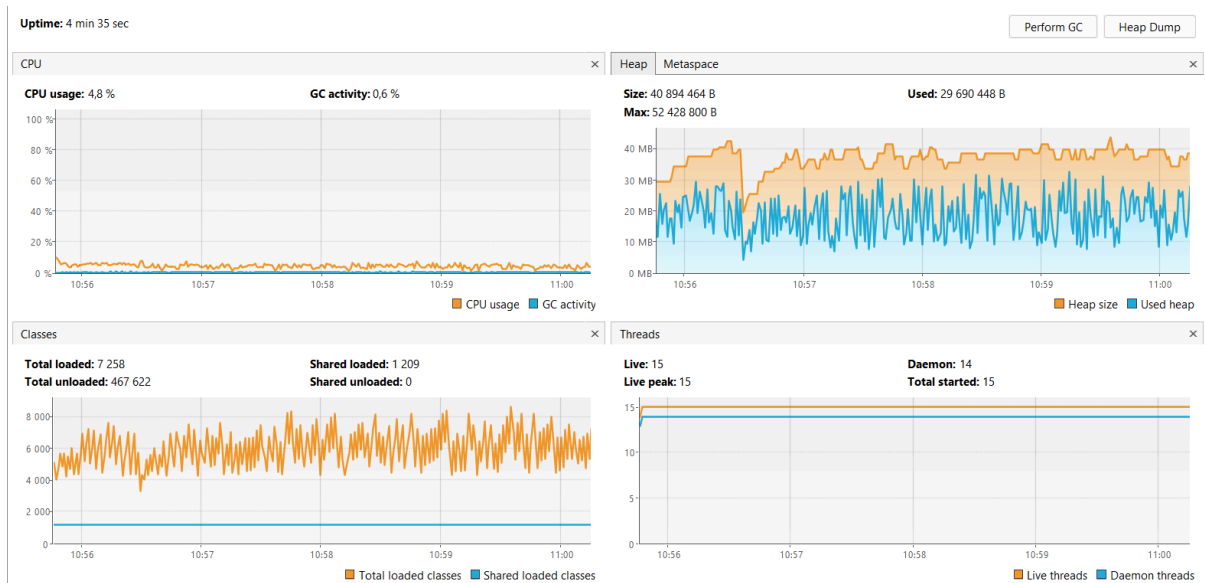
```
static void clearErrorMessages() {
    _errorMessages.clear();
}
```

В методе HTTPUnitOptions есть метод clearScriptErrorMessages

```
public static void clearScriptErrorMessages() {
    getScriptingEngine().clearErrorMessages();
}
```

Будем вызывать его в main методе

```
while (true) {
    WebResponse response = sc.getResponse(request);
    System.out.println("Count: " + number++ + response);
    HttpUnitOptions.clearScriptErrorMessages();
    java.lang.Thread.sleep(200);
}
```



В VisualVM увидим что хип больше не растёт

Вывод

В процессе работы мы узнали про методы наблюдения за выполнением программы, разработали MBeans и пофиксили утечку памяти в программе