

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Лабораторная работа по
Основам программной инженерии №4
Вариант 3087

Работу выполнил:

Алферов Г.А.

Группа:

P3207

Санкт-Петербург,

2025

Оглавление

Текст задания	3
Реализация	4
Показания MBean классов из JConsole.....	8
График изменения показаний MBean-классов с течением времени.....	10
Исследование программы на утечки памяти	12
Вывод.....	14

Текст задания

Вариант 3087

Внимание! У разных вариантов разный текст задания!

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

- MBeap, 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 MBeap должен отправлять оповещение об этом событии.
- MBeap, определяющий средний интервал между кликами пользователя по координатной плоскости.

2. С помощью утилиты JConsole провести мониторинг программы:

- Снять показания MBeap-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 MBeap-классов, разработанных в ходе выполнения задания 1, с течением времени.
- Определить имя класса, объекты которого занимают наибольший объем 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 отчет, 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.

Реализация

interface PointTrackerMBean

```
package org.GleBlassUSA.beans;

public interface PointTrackerMBean {
    long getTotalPoints();
    long getMissedPoints();
}
```

interface ClickIntervalMBean

```
package org.GleBlassUSA.beans;

public interface ClickIntervalMBean {
    double getClickInterval();
    long getClickCount();
    void resetStat();
}
```

PointTracker

```
package org.GleBlassUSA.beans;

import jakarta.enterprise.context.ApplicationScoped;
import jakarta.enterprise.context.Destroyed;
import jakarta.enterprise.context.Initialized;
import jakarta.enterprise.event.Observes;
import jakarta.inject.Inject;
import jakarta.inject.Named;
import org.GleBlassUSA.dao.PointDao;
import org.GleBlassUSA.models.Point;

import javax.management.*;
import java.io.Serializable;

@Named("pointTracker")
@ApplicationScoped
public class PointTracker extends NotificationBroadcasterSupport implements
Serializable, PointTrackerMBean {

    private int sequenceNumber = 0;

    @Inject
    private PointDao pointDao;

    public void init(@Observes @Initialized(ApplicationScoped.class) Object
unused) {
        org.GleBlassUSA.beans.MBeanRegistryUtil.registerBean(this,
"pointTracker");
    }

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

    @Override
    public long getTotalPoints() {
```

```

        return pointDao.getPoints().size();
    }

    @Override
    public long getMissedPoints() {
        return pointDao.getPoints().stream()
            .filter(p -> !inArea(p))
            .count();
    }

    public void checkPoint(Point point) {
        if (inArea(point)) {
            sendOut(point);
        }
    }

    private boolean inArea(Point point) {
        return point.calculate();
    }

    private void sendOut(Point point) {
        Notification notification = new Notification(
            "Point out of bounds",
            this.getClass().getName(),
            sequenceNumber++,
            System.currentTimeMillis(),
            String.format("Point (%.2f, %.2f out of bounds:",
point.getX(), point.getY())
        );
        sendNotification(notification);
    }

    @Override
    public MBeanNotificationInfo[] getNotificationInfo() {
        String[] types = new String[]{
            AttributeChangeNotification.ATTRIBUTE_CHANGE
        };
        String name = AttributeChangeNotification.class.getName();
        String description = "Point out of bounds notification";
        return new MBeanNotificationInfo[]{
            new MBeanNotificationInfo(types, name, description)
        };
    }
}

```

ClickInterval

```

package org.GleBlassUSA.beans;

import jakarta.enterprise.context.ApplicationScoped;
import jakarta.enterprise.context.Destroyed;
import jakarta.enterprise.context.Initialized;
import jakarta.enterprise.event.Observes;
import jakarta.inject.Named;

import java.io.Serializable;
import java.util.concurrent.atomic.AtomicLong;
import java.util.concurrent.atomic.AtomicReference;

@Named("ClickInterval")
@ApplicationScoped
public class ClickInterval implements Serializable, ClickIntervalMBean {

```

```

        private final AtomicLong lastClickTime = new AtomicLong();
        private final AtomicLong clickCount = new AtomicLong();
        private final AtomicReference<Double> averageInterval = new
AtomicReference<>(0.0);
        private boolean isFirstClick = true;

        public void init(@Observes @Initialized(ApplicationScoped.class) Object
unused) {
            org.GleBlassUSA.beans.MBeanRegistryUtil.registerBean(this,
"ClickInterval");
        }

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

        public void recordClick() {
            long now = System.currentTimeMillis();
            if (isFirstClick) {
                lastClickTime.set(now);
                isFirstClick = false;
                return;
            }
            long past = lastClickTime.getAndSet(now);
            long interval = now - past;
            long count = clickCount.incrementAndGet();
            averageInterval.updateAndGet(cur -> (cur * (count - 1) + interval) /
count);
        }

        @Override
        public double getClickInterval() {
            return averageInterval.get() / 1000;
        }

        @Override
        public long getClickCount() {
            return clickCount.get();
        }

        @Override
        public void resetStat() {
            lastClickTime.set(0);
            clickCount.set(0);
            averageInterval.set(0.0);
        }
    }
}

```

MBeanRegistryUtil

```

package org.GleBlassUSA.beans;

import jakarta.servlet.ServletContextListener;
import lombok.experimental.UtilityClass;

import javax.management.*;
import java.lang.management.ManagementFactory;
import java.util.HashMap;
import java.util.Map;

```

```

@UtilityClass
public class MBeanRegistryUtil implements ServletContextListener {
    private final Map<Class<?>, ObjectName> beans = new HashMap<>();

    public void registerBean(Object bean, String name) {
        try {
            String domain = bean.getClass().getPackageName();
            String type = bean.getClass().getSimpleName();
            ObjectName objectName = new
ObjectName(String.format("%s:type=%s,name=%s", domain, type, name));
            ManagementFactory.getPlatformMBeanServer().registerMBean(bean,
objectName);
            beans.put(bean.getClass(), objectName);
        } catch (InstanceAlreadyExistsException | MBeanRegistrationException
| NotCompliantMBeanException |
            MalformedObjectNameException ex) {
            ex.printStackTrace();
        }
    }

    public void unregisterBean(Object bean) {
        if (!beans.containsKey(bean.getClass())) {
            throw new IllegalArgumentException("Specified bean is not
registered.");
        }
        try {
            ObjectName objectName = beans.get(bean.getClass());
            ManagementFactory.getPlatformMBeanServer().unregisterMBean(objectName);
        } catch (InstanceNotFoundException | MBeanRegistrationException ex) {
            ex.printStackTrace();
        }
    }
}

```

PointService

```

package org.GleBlassUSA.beans;

import jakarta.enterprise.context.ApplicationScoped;
import jakarta.inject.Inject;
import org.GleBlassUSA.dao.PointDao;
import org.GleBlassUSA.models.Point;

@ApplicationScoped
public class PointService {
    private final PointDao pointDao = PointDao.getInstance();
    @Inject
    private PointTracker pointTracker;

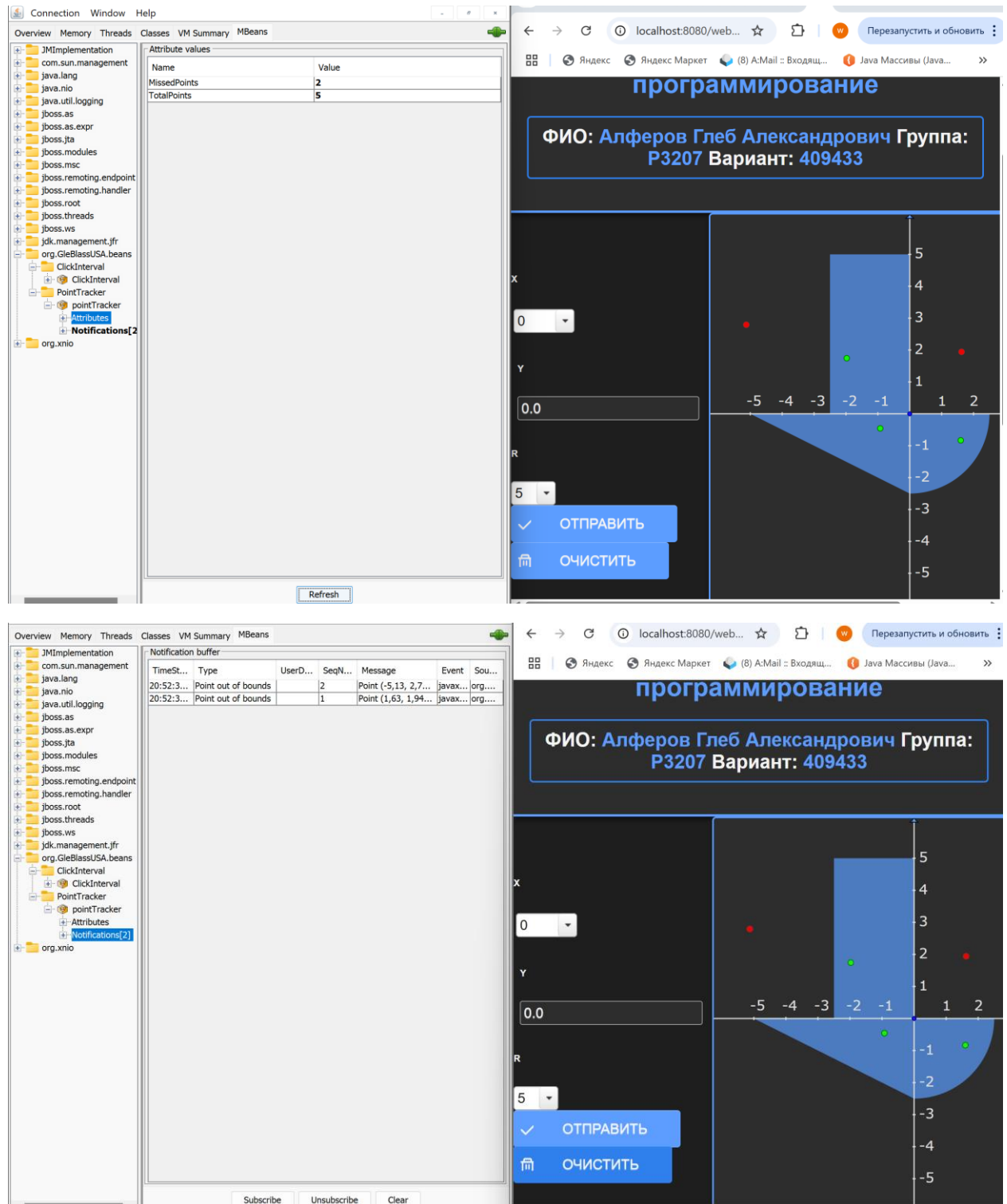
    public void addPoint(Point point) {
        pointDao.addPoint(point);
        pointTracker.checkPoint(point);
    }
}

```

Весь проект

https://github.com/wrakelft/OPI_ITMO/tree/main/lab4/src

Показания MBean классов из JConsole



OverviewMemoryThreadsClassesVM SummaryMBeans

Attribute values

Name	Value
ClickCount	4
ClickInterval	4.369841178995E8

org.jboss.as

org.jboss.as.expr

org.jboss.as.jta

org.jboss.as.modules

org.jboss.as.msc

org.jboss.remoting.endpoint

org.jboss.remoting.handler

org.jboss.root

org.jboss.threads

org.jboss.ws

org.jdk.management.jfr

org.glassfish.usa.beans

ClickInterval

ClickInterval

Attributes

Operations

PointTracker

pointTracker

Attributes

Notifications[3]

org.xnio

Refresh

localhost:8080/web...

Перезапустить и обновить

ЯндексЯндекс Маркет(8) A:Mail :: Входящ...Java Массивы (Java...

программирование

ФИО: Алферов Глеб Александрович Группа: Р3207 Вариант: 409433

0

0.0

5

ОТПРАВИТЬ

ОЧИСТИТЬ

В ходе использования утилиты Jconsole можно сделать вывод что MBeans были успешно разработаны. Уведомления отправляются штатно и позволяют отслеживать события.

Java Monitoring & Management Console - pid: 8608 jboss-modules.jar -mp C:\wildfly\wildfly-33.0.0.Final\modules org.jboss.as.standalone -Djboss.home.dir=C:\wildfly\wildfly-33.0.0.Final

ConnectionWindowHelp

OverviewMemoryThreadsClassesVM SummaryMBeans

VM Summary

четверг, 22 мая 2025 г., 20:56:46 Москва, стандартное время

Connection name: pid: 8608 jboss-modules.jar -mp C:\wildfly\wildfly-33.0.0.Final\modules org.jboss.as.standalone -Djboss.home.dir=C:\wildfly\wildfly-33.0.0.Final

Virtual Machine: Java HotSpot(TM) 64-Bit Server VM version 17.0.8+9-LTS-211

Vendor: Oracle Corporation

Name: 8608@gleb_nt

Uptime: 7 minutes

Process CPU time: 31,875 seconds

JIT compiler: HotSpot 64-Bit Tiered Compilers

Total compile time: 2 minutes

Live threads: 120

Peak: 240

Daemon threads: 32

Total threads started: 286

Current classes loaded: 32 476

Total classes loaded: 32 496

Total classes unloaded: 20

Current heap size: 249 873 kbytes

Maximum heap size: 524 288 kbytes

Garbage collector: Name = 'G1 Young Generation', Collections = 67, Total time spent = 0,474 seconds

Garbage collector: Name = 'G1 Old Generation', Collections = 0, Total time spent = 0,000 seconds

Committed memory: 302 080 kbytes

Pending finalization: 0 objects

Operating System: Windows 11 10.0

Architecture: amd64

Number of processors: 20

Committed virtual memory: 771 400 kbytes

Total physical memory: 16 494 840 kbytes

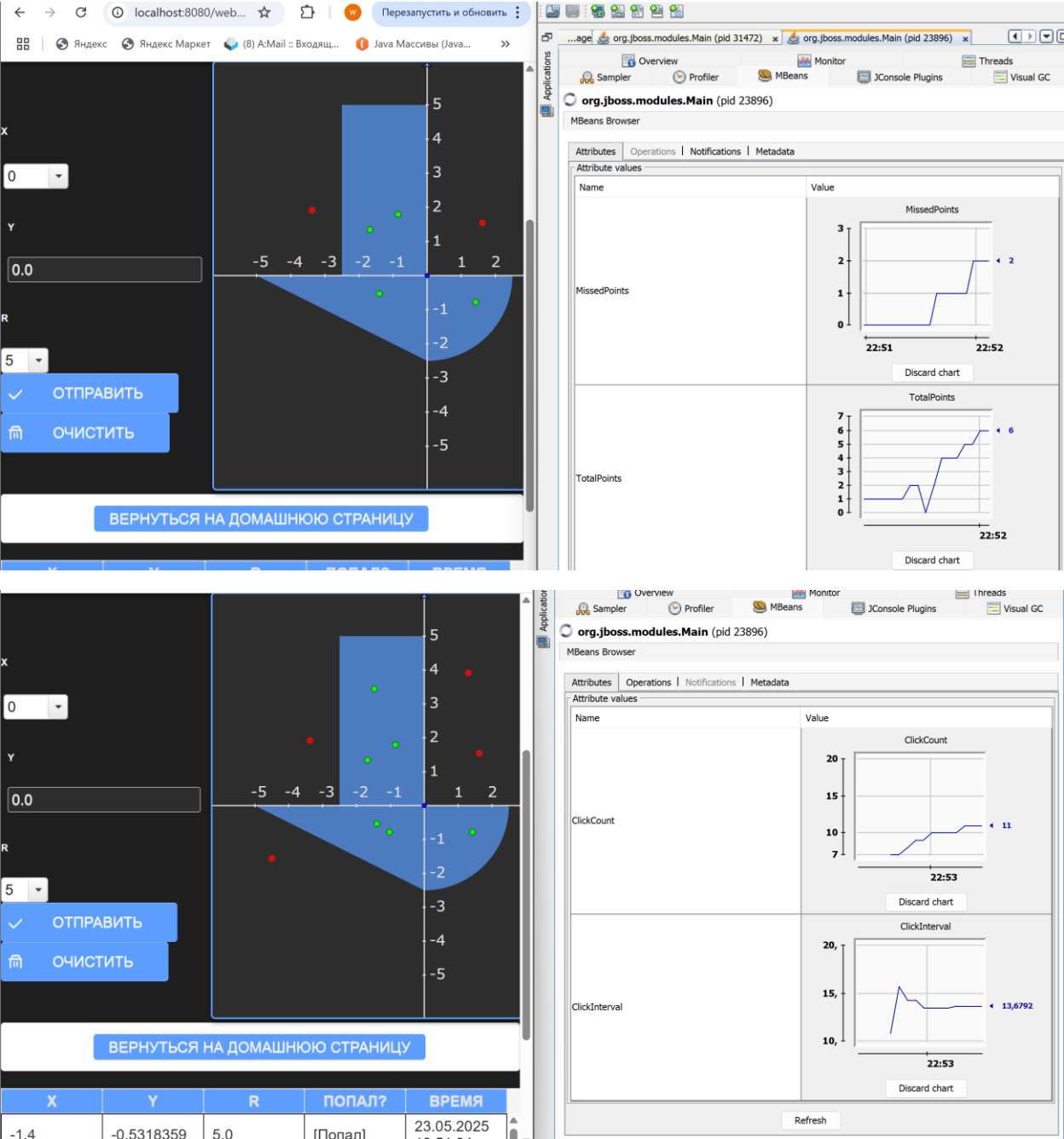
Free physical memory: 1 927 692 kbytes

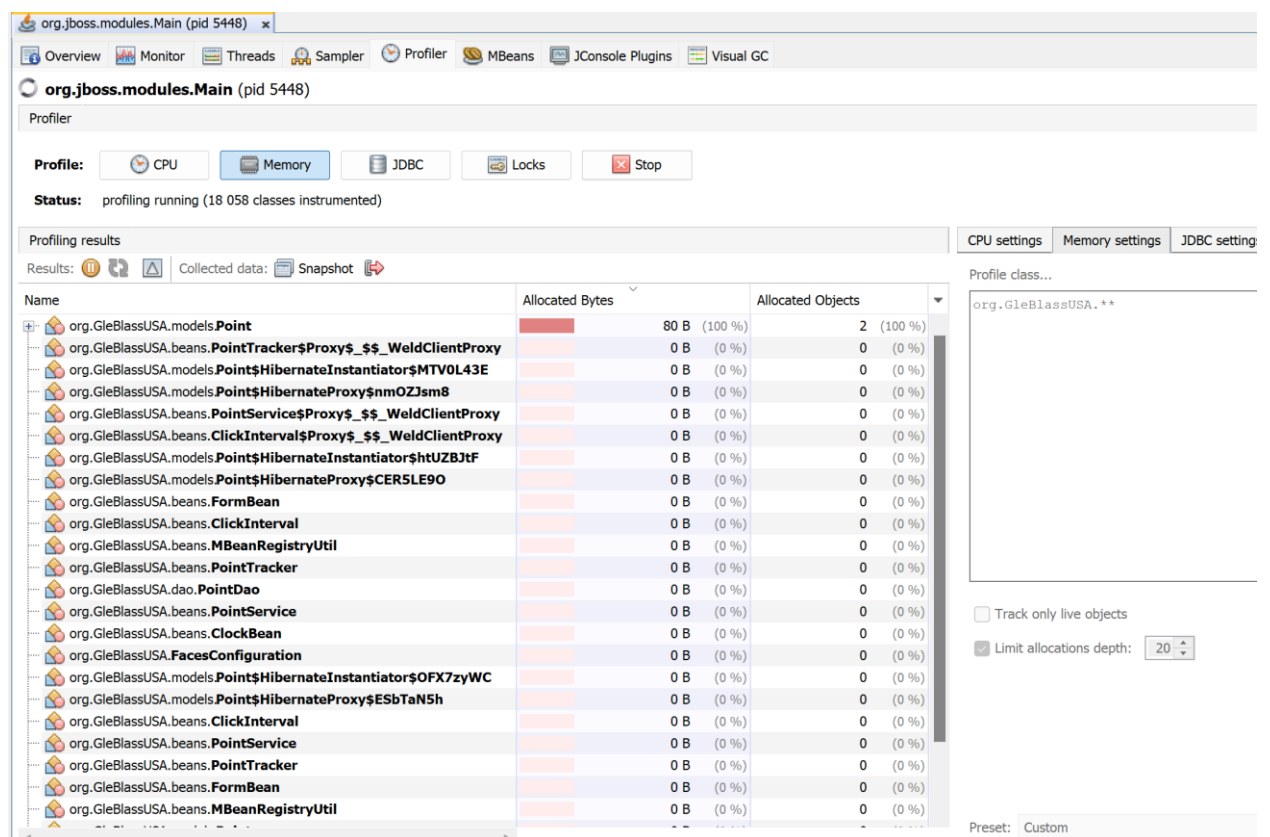
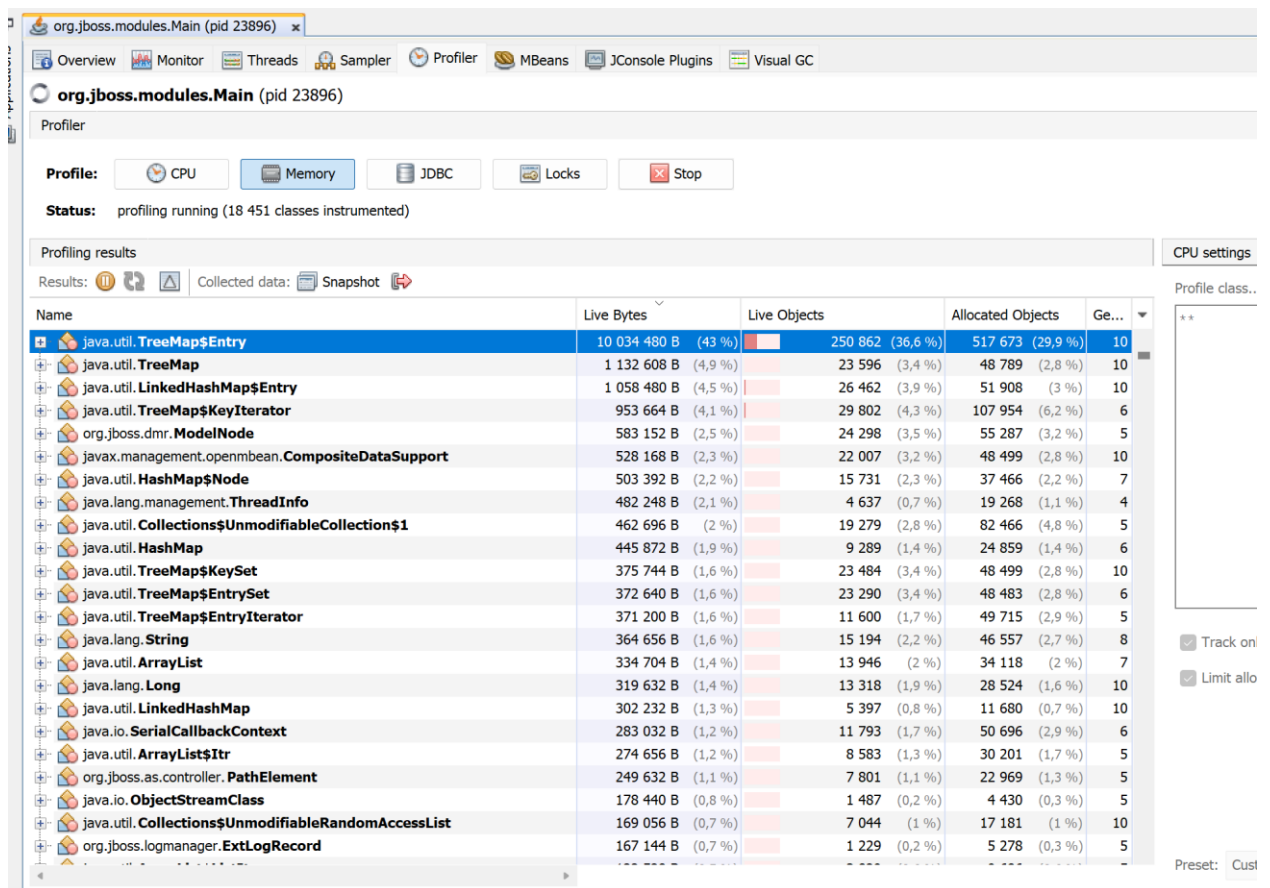
Total swap space: 34 290 408 kbytes

Free swap space: 5 112 616 kbytes

420 000 мс

График изменения показаний MBean-классов с течением времени



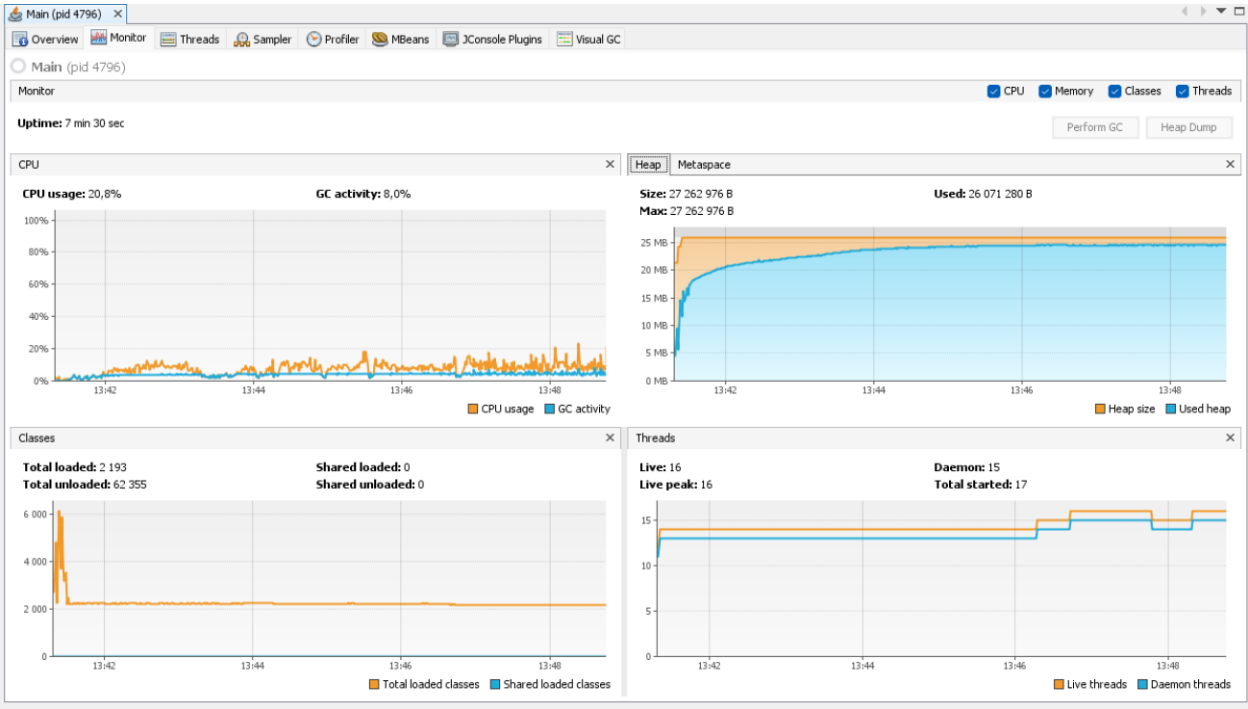


Видно, что больше всего памяти занимают значения TreeMap\$Entry

Больше всего памяти из пользовательских классов занимают объекты Point

Эта не прямая связь обусловлена тем, что во время работы приложения, развернутого на WildFly он автоматически создает TreeMap для своих внутренних процессов (хранение конфигов, кэширования, обработка запросов), а так как при работе приложения WildFly вызывает наши сервлеты и бины он и создает временные TreeMap для инъекций зависимостей или других нужд приложения.

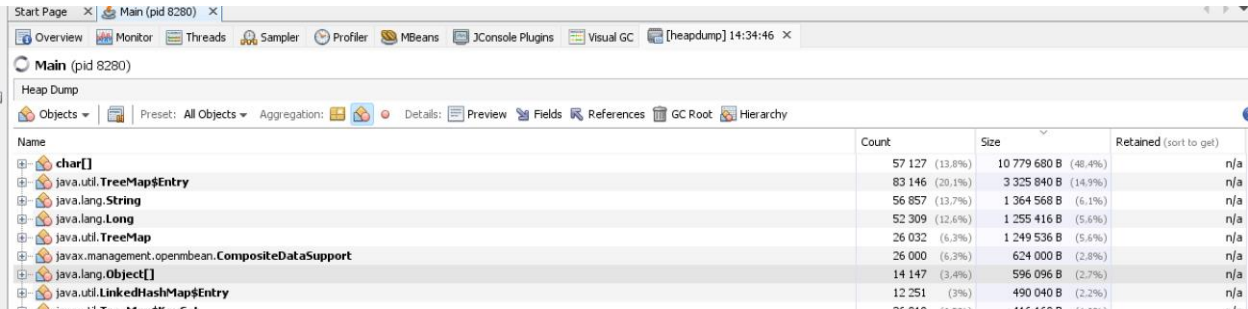
Исследование программы на утечки памяти



Установим максимальный размер кучи в 25Мб с помощью -Xmx25m и запустим программу. Из графика памяти видно, что размер кучи постоянно увеличивается, что говорит о проблемах с использованием памяти в программе.

```
Count: 58009[ _response = com.meterware.servletunit.ServletUnitHttpResponse@188a062b]
Count: 58010[ _response = com.meterware.servletunit.ServletUnitHttpResponse@ed81d88]
Exception in thread "main" java.lang.OutOfMemoryError: Create breakpoint : GC overhead limit exceeded
    at java.util.HashMap.newNode(HashMap.java:1750)
    at java.util.HashMap.putVal(HashMap.java:631)
    at java.util.HashMap.putMapEntries(HashMap.java:515)
    at java.util.HashMap.<init>(HashMap.java:490)
    at java.util.PropertyResourceBundle.<init>(PropertyResourceBundle.java:139)
```

С помощью HeapDump найдем объекты, занимающие большую часть памяти.



Видно, что создаются экземпляры строк на каждый запрос этим можно объяснить и рост `char[]` и `TreeMap$Entry`, так как `char[]` хранит символы строк, `String` его обертка, а `TreeMap$Entry` связывает ключи `String` и другие наши объекты.

Object	Count	Size	Percentage	Memory	Percentage	Class
java.lang.Object[]	14 147	3.4%	596 096 B	(2.7%)	n/a	
java.lang.Object[]#2726	47 427 items		189 728 B	(0.9%)	n/a	
<Items>						
[0] = java.lang.String#8903	Script 'document.wr('Hello Document')' failed: TypeError: undefined is not a function. (httpunit;)		24 B	(0%)	n/a	
[1] = java.lang.String#8902	Script 'document.wr('Hello Document')' failed: TypeError: undefined is not a function. (httpunit;)		24 B	(0%)	n/a	
[2] = java.lang.String#8901	Script 'document.wr('Hello Document')' failed: TypeError: undefined is not a function. (httpunit;)		24 B	(0%)	n/a	
[3] = java.lang.String#8900	Script 'document.wr('Hello Document')' failed: TypeError: undefined is not a function. (httpunit;)		24 B	(0%)	n/a	
[4] = java.lang.String#8899	Script 'document.wr('Hello Document')' failed: TypeError: undefined is not a function. (httpunit;)		24 B	(0%)	n/a	
[5] = java.lang.String#8898	Script 'document.wr('Hello Document')' failed: TypeError: undefined is not a function. (httpunit;)		24 B	(0%)	n/a	
[6] = java.lang.String#8897	Script 'document.wr('Hello Document')' failed: TypeError: undefined is not a function. (httpunit;)		24 B	(0%)	n/a	
[7] = java.lang.String#8896	Script 'document.wr('Hello Document')' failed: TypeError: undefined is not a function. (httpunit;)		24 B	(0%)	n/a	
[8] = java.lang.String#8895	Script 'document.wr('Hello Document')' failed: TypeError: undefined is not a function. (httpunit;)		24 B	(0%)	n/a	
[9] = java.lang.String#8894	Script 'document.wr('Hello Document')' failed: TypeError: undefined is not a function. (httpunit;)		24 B	(0%)	n/a	
[10] = java.lang.String#8893	Script 'document.wr('Hello Document')' failed: TypeError: undefined is not a function. (httpunit;)		24 B	(0%)	n/a	
[11] = java.lang.String#8892	Script 'document.wr('Hello Document')' failed: TypeError: undefined is not a function. (httpunit;)		24 B	(0%)	n/a	

Исследовав, видим повторяющиеся строки

Object	Count	Size	Percentage	Memory	Percentage	Class
java.lang.Object[]	14 147	3.4%	596 096 B	(2.7%)	n/a	
java.lang.Object[]#2726	47 427 items		189 728 B	(0.9%)	n/a	
<Items>						
<References>						
elementData in java.util.ArrayList#92	44 214 elements		24 B	(0%)	n/a	
static _errorMessages in class com.meterware.httpunit.javascript.JavaScript	JavaScript		64 B	(0%)	n/a	

Все они связаны с `__errorMessages`

Объекты `_errorMessages` хранятся в `ArrayList`

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

Добавление объектов в этот список:

```
} else {
    _errorMessages.add( errorMessage );
}
}
```

Накопление `_errorMessages` в списке, за счет чего и получается переполнение памяти.

В программе есть функция для очистки:

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

```
public void clearErrorMessages() { 1 usage
    JavaScript.clearErrorMessages();
}
```

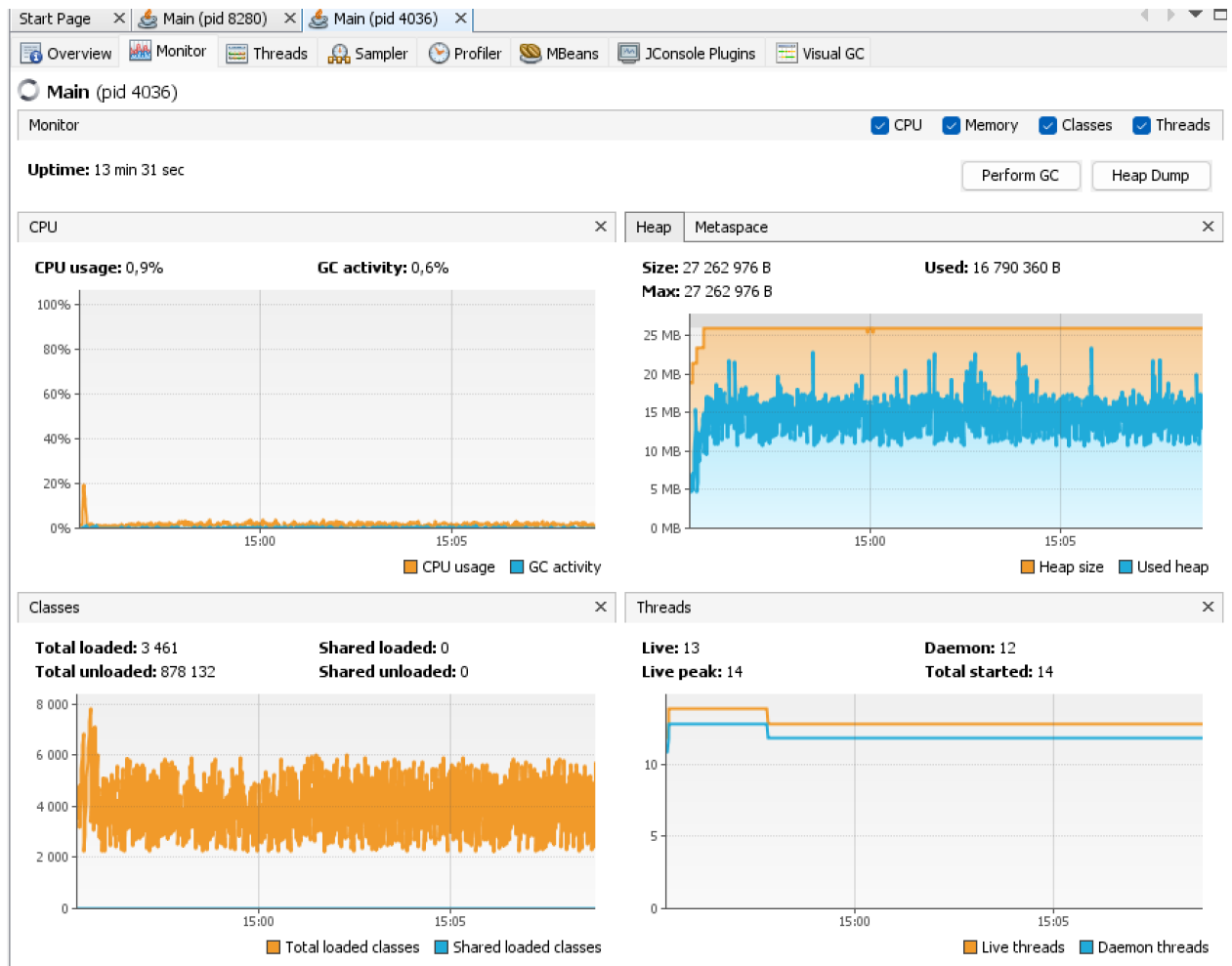
```
public static void clearScriptErrorMessages() { no usages
    getScriptingEngine().clearErrorMessages();
}
```

Просмотрев классы видим, что она не используется.

Решением будет очистка после выполнения запроса.

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

Теперь память не стремится к максимальному значению. Программа не выкидывает OutOfMemoryError.



Вывод

Во время выполнения лабораторной работы я использовал утилиты для мониторинга работы программы JConsole и VisualVM. А также была произведена работа по поиску проблемы, ее локализации и устранению.