

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: 7r4h47.

выполнили:

Иванов Матвей Сергеевич

Сосновцев Григорий Алексеевич

Преподаватель

Бострикова Дарья Константиновна

г. Санкт-Петербург, 2023

Содержание

Задание.....	3
1. Создание MBeans.....	4
1. MBean, считающий общее число установленных пользователем точек, а также число точек, не попадающих в область.....	4
2. MBean, определяющий площадь получившейся фигуры.....	6
3. Основной Bean приложения, управляющий созданными MBeans.....	7
2. Мониторинг программы с помощью утилиты JConsole.....	9
3. Мониторинг и профилирование программы с помощью VisualVM. 10	
4. С помощью утилиты VisualVM и профилировщика IDE Idea локализовать и устранить проблемы с производительностью в выданной программе.....	12
1. Описание выявленной проблемы.....	12
2. Описание путей устранения выявленной проблемы.....	12
3. Описание алгоритма действий, который позволил выявить и локализовать проблему.....	13
Репозиторий с кодом лабораторной работы:.....	15
Выводы.....	15

Задание

1. Для своей программы из лабораторной работы #3 по дисциплине "Веб-программирование" реализовать:
 - a. MBean, считающий общее число установленных пользователем точек, а также число точек, не попадающих в область. В случае, если пользователь совершил 4 "промаха" подряд, разработанный MBean должен отправлять оповещение об этом событии.
 - b. MBean, определяющий площадь получившейся фигуры.
2. С помощью утилиты JConsole провести мониторинг программы:
 - a. Снять показания MBean-классов, разработанных в ходе выполнения задания 1.
 - b. Определить имя и версию ОС, под управлением которой работает JVM.
3. С помощью утилиты VisualVM провести мониторинг и профилирование программы:
 - a. Снять график изменения показаний MBean-классов, разработанных в ходе выполнения задания 1, с течением времени.
 - b. 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 выданной программе. По результатам локализации и устранения проблемы необходимо составить отчёт, в котором должна содержаться следующая информация:
 - a. Описание выявленной проблемы.
 - b. Описание путей устранения выявленной проблемы.
 - c. 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.

1. Создание MBeans

Для выполнения работы была взят код проекта Лабораторной работы №3 по дисциплине “Веб-программирование”:

https://github.com/Mrjoulin/ITMOLabs/tree/master/semester_3/WebLabs/Lab3

1. MBean, считающий общее число установленных пользователем точек, а также число точек, не попадающих в область.

CounterMXBean.java

```
public interface CounterMXBean {
    int addHit(boolean hitResult);

    int addCorrectHit();

    int addMissedHit();

    void clearMissedStreak();

    int getHitsCount();

    int getMissedHitsCount();

    int getMissedHitsStreakCount();
}
```

Counter.java

```
public class Counter extends NotificationBroadcasterSupport
    implements CounterMXBean, Serializable {
    private final int NUM_MISSES_TO_STREAK = 4;
    private int hitsCount = 0;
    private int missedHitsCount = 0;
    private int missedHitsStreakCount = 0;
    private int sequenceNumber = 1;

    @Override
    public int addHit(boolean hitResult) {
        return hitResult ? addCorrectHit() : addMissedHit();
    }
}
```

```

    }

    @Override
    public int addCorrectHit() {
        clearMissedStreak();
        return ++hitsCount;
    }

    @Override
    public int addMissedHit() {
        hitsCount++;
        missedHitsCount++;
        if (++missedHitsStreakCount % NUM_MISSES_TO_STREAK == 0) {
            String message = NUM_MISSES_TO_STREAK + " misses streak";
            sendNotification(
                new Notification(message, this, sequenceNumber++, message)
            );
        }
        return missedHitsStreakCount;
    }

    @Override
    public void clearMissedStreak() {
        missedHitsStreakCount = 0;
    }

    @Override
    public int getHitsCount() {
        return hitsCount;
    }

    @Override
    public int getMissedHitsCount() {
        return missedHitsCount;
    }

    @Override
    public int getMissedHitsStreakCount() {
        return missedHitsStreakCount;
    }
}

```

2. MBean, определяющий площадь получившейся фигуры.

SquareMXBean.java

```
public interface SquareMXBean {  
    double calculateSquare(double r);  
  
    double getLastSquare();  
}
```

Square.java

```
public class Square implements SquareMXBean, Serializable {  
    private double lastSquare = 0;  
    private final double triangleCoefficient = 0.25;  
    private final double rectangleCoefficient = 1.0;  
    private final double circleCoefficient = Math.PI / 4;  
  
    @Override  
    public double calculateSquare(double r) {  
        lastSquare = (  
            triangleCoefficient + rectangleCoefficient + circleCoefficient  
        ) * Math.pow(r, 2);  
  
        return lastSquare;  
    }  
  
    @Override  
    public double getLastSquare() { return lastSquare; }  
}
```

3. Основной Bean приложения, управляющий созданными MBeans.

ClientBean.java

```
@Getter
@Setter
@ToString
@ManagedBean(name = "client")
@SessionScoped
public class ClientBean implements Serializable {
    private final String sessionId;
    private final LinkedList<HitResult> currentHits;

    @ManagedProperty(value = "#{coordinates}")
    private Coordinates coordinates = new Coordinates();
    @ManagedProperty(value = "#{service}")
    private Service service = new Service();

    private MBeanServer mbs;
    private CounterMXBean counterMXBean;
    private SquareMXBean squareMXBean;

    public ClientBean() {
        this.sessionId =
FacesContext.getCurrentInstance().getExternalContext().getSessionId(true);
        this.currentHits = service.getUserHits(sessionId);

        initMBeans();
    }

    private void initMBeans() {
        this.mbs = ManagementFactory.getPlatformMBeanServer();
        this.counterMXBean = new Counter();
        this.squareMXBean = new Square();

        try {
            mbs.registerMBean(counterMXBean,
                new ObjectName("ClientBean:name=counterMXBean"));
            mbs.registerMBean(squareMXBean,
                new ObjectName("ClientBean:name=squareMXBean"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

    public void makeUserRequest() {
        makeRequest(this.coordinates);
    }

    public void makeRemoteRequest() {
        Function<String, Double> getParam = (name) -> {
            return
Double.parseDouble(FacesContext.getCurrentInstance().getExternalContext().get
etRequestParameterMap().get(name));
        };

        try {
            Coordinates coordinates = new Coordinates(getParam.apply("x"),
getParam.apply("y"), getParam.apply("r"));
            makeRequest(coordinates);
        } catch (NullPointerException | NumberFormatException exception) {
            System.out.println("Can't parse values from request params");
        }
    }

    public void makeRequest(Coordinates coordinates) {
        System.out.println("Make request: " + coordinates.toString());
        HitResult result = service.processRequest(this.sessionId,
coordinates);

        if (result != null) {
            this.currentHits.addFirst(result);
            processHandlers(result);
        }
    }

    public void clearHits() {
        currentHits.clear();
        service.clearUserHits(this.sessionId);

        System.out.println("Current hits: " + currentHits);
    }

    private void processHandlers(HitResult result) {
        this.counterMXBean.addHit(result.isResult());
        double square = this.squareMXBean.calculateSquare(result.getR());

        System.out.println("Area square: " + square);
    }
}

```


2. Мониторинг программы с помощью утилиты JConsole.

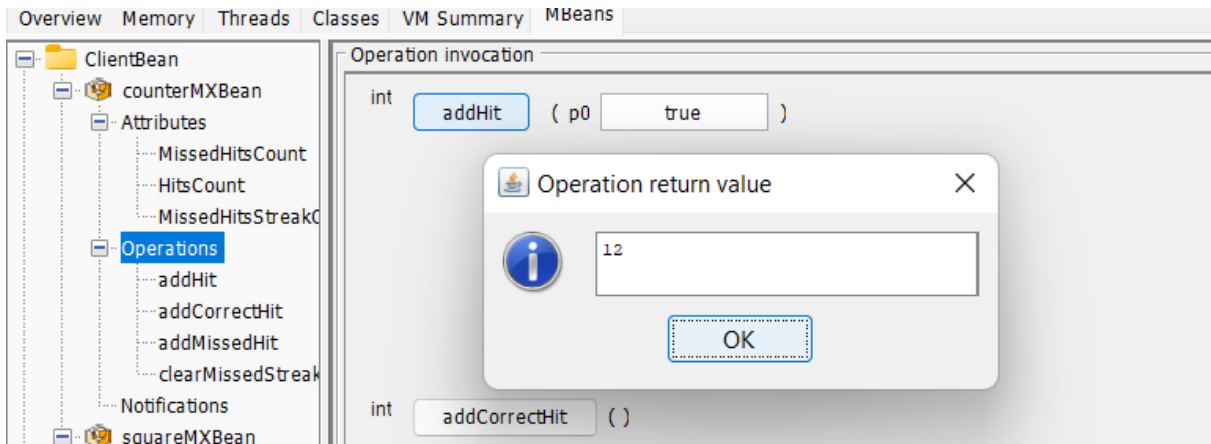
2.1 Показания MBean-классов

2.1.1 Атрибуты CounterMXBean



Name	Value
HitsCount	11
MissedHitsCount	5
MissedHitsStreakCount	4

2.1.2 Операции CounterMXBean



Operation invocation

int addHit (p0 true)

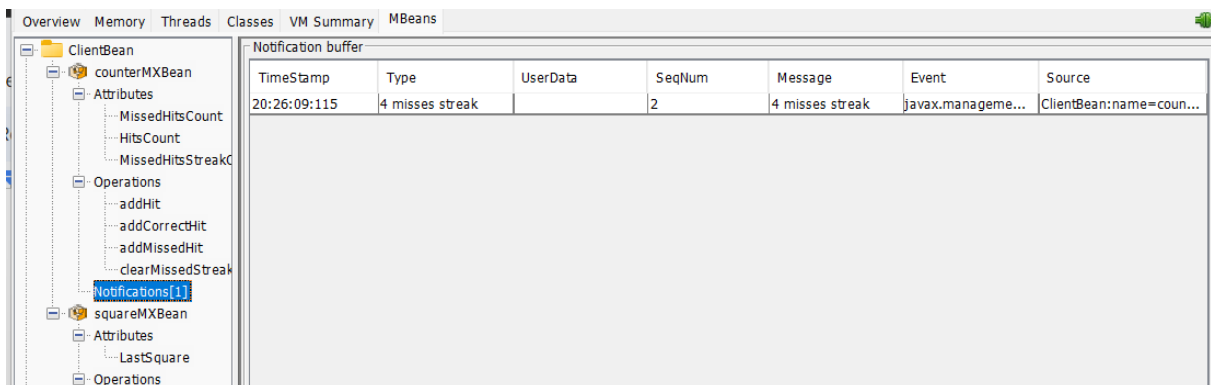
Operation return value

12

OK

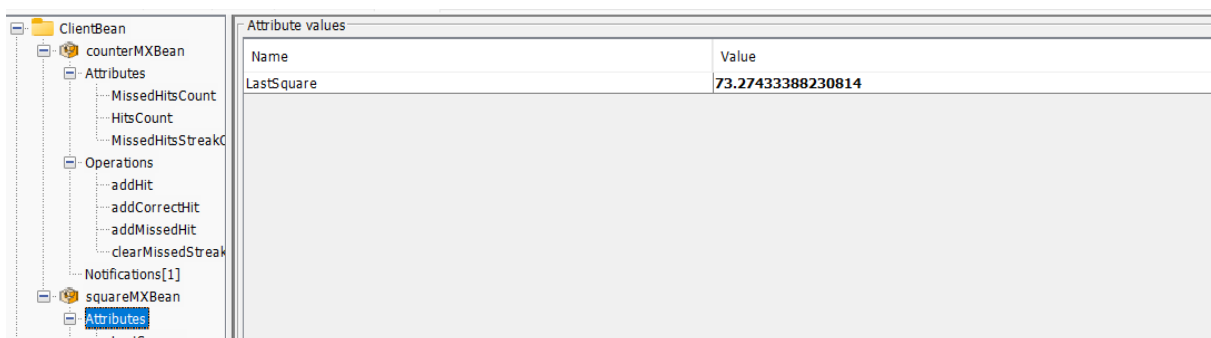
int addCorrectHit ()

2.1.3 Уведомления CounterMXBean



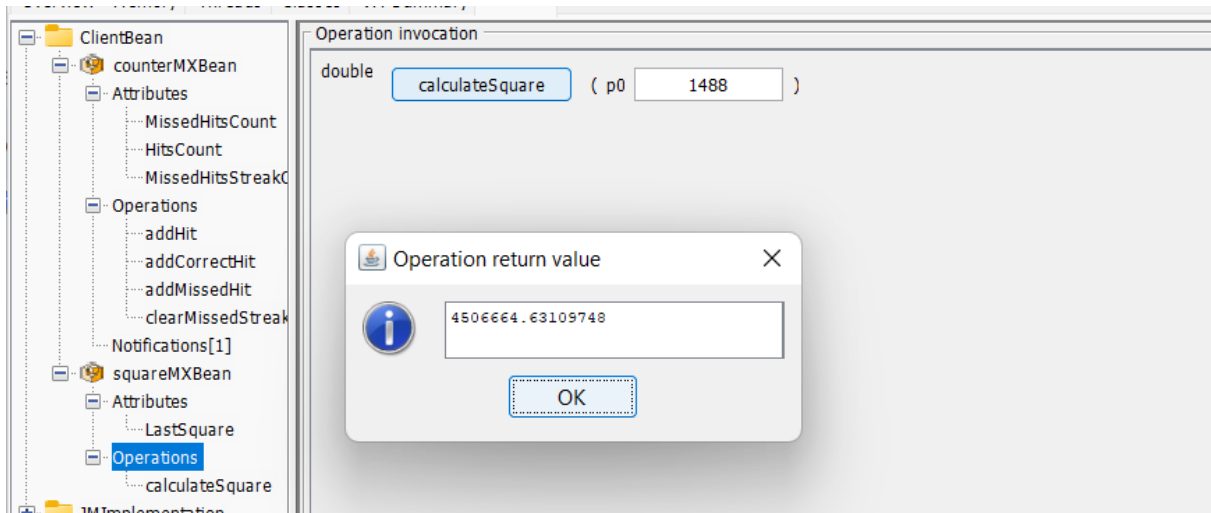
TimeStamp	Type	UserData	SeqNum	Message	Event	Source
20:26:09:115	4 misses streak		2	4 misses streak	javax.management...	ClientBean:name=coun...

2.1.4 Атрибуты SquareMXBean



Name	Value
LastSquare	73.27433388230814

2.1.5 Операции SquareMXBean

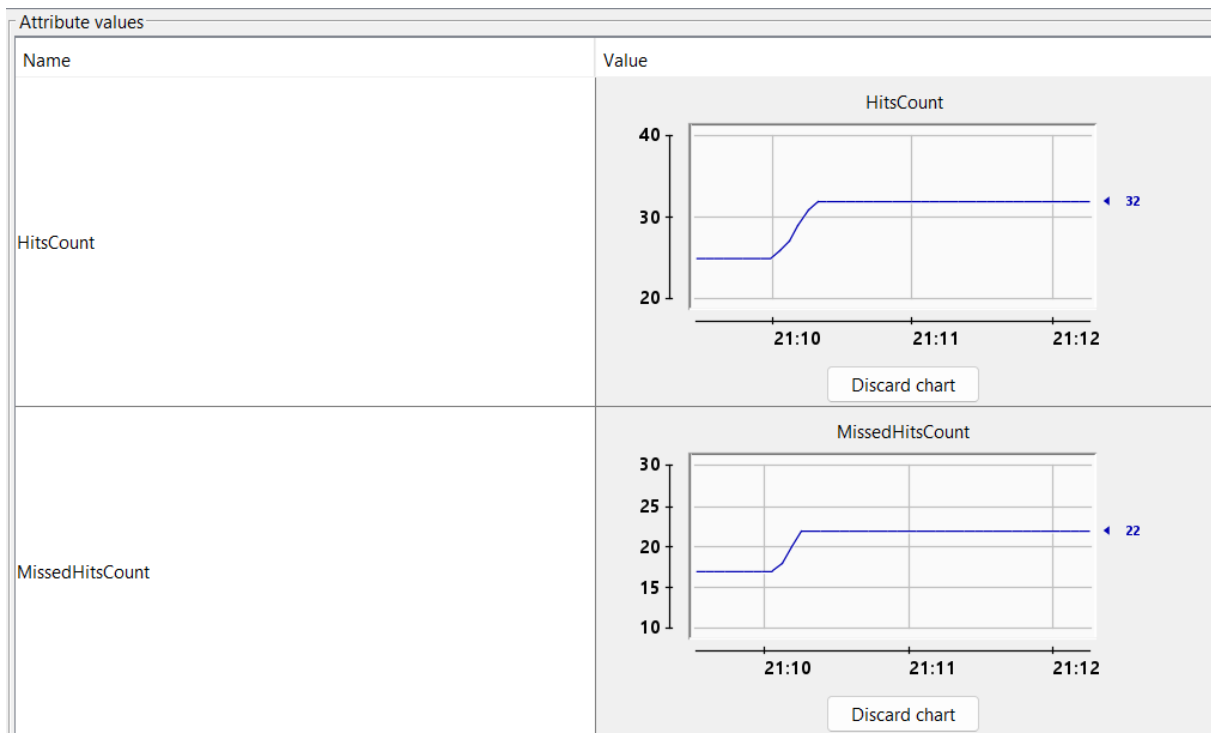


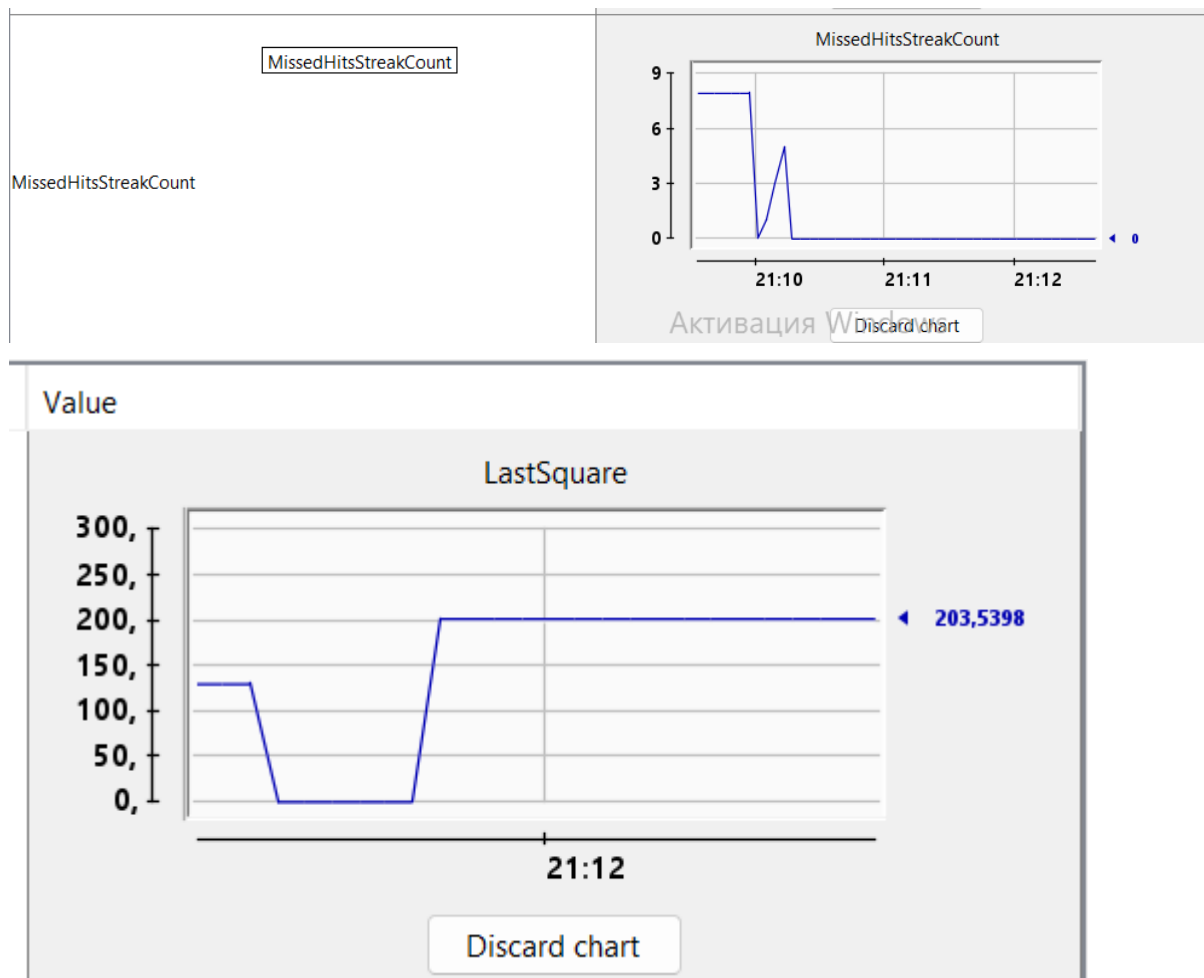
2.2 OC

Operating System: Windows 10 10.0

3. Мониторинг и профилирование программы с помощью VisualVM.

3.1 Графики изменения показаний MBeans





4.2 Класс, объекты которого занимают больше всего памяти

Name	Live Bytes	Live Objects	Allocated Objects	Generations
com.joulin.lab3.db.HitResult	100 %	4 (100 %)	29 (50 %)	1
	224 B (100 %)	4 (100 %)	13 (22,4 %)	1
	224 B (100 %)	4 (100 %)	13 (22,4 %)	1
	224 B (100 %)	4 (100 %)	13 (22,4 %)	1
	224 B (100 %)	4 (100 %)	13 (22,4 %)	1
	224 B (100 %)	4 (100 %)	13 (22,4 %)	1
	224 B (100 %)	4 (100 %)	13 (22,4 %)	1
	224 B (100 %)	4 (100 %)	13 (22,4 %)	1
	224 B (100 %)	4 (100 %)	13 (22,4 %)	1
	224 B (100 %)	4 (100 %)	13 (22,4 %)	1
	0 B (0 %)	0 (0 %)	0 (0 %)	0
	0 B (0 %)	0 (0 %)	0 (0 %)	0
	0 B (0 %)	0 (0 %)	0 (0 %)	0
	0 B (0 %)	0 (0 %)	0 (0 %)	0
	0 B (0 %)	0 (0 %)	0 (0 %)	0
	0 B (0 %)	0 (0 %)	0 (0 %)	0
	0 B (0 %)	0 (0 %)	0 (0 %)	0
	0 B (0 %)	0 (0 %)	0 (0 %)	0
	0 B (0 %)	0 (0 %)	0 (0 %)	0
	0 B (0 %)	0 (0 %)	29 (50 %)	0

4. С помощью утилиты VisualVM и профилировщика IDE Idea локализовать и устранить проблемы с производительностью в выданной программе.

1. Описание выявленной проблемы.

1. Зависание потока на 200 мс
2. Отсутствие очищения коллекции сообщений ошибок, что ведёт к переполнению памяти и падению производительности

2. Описание путей устранения выявленной проблемы.

1. Убрать строчку с Thread.sleep()
2. Очищать коллекцию ошибок после каждого запроса.

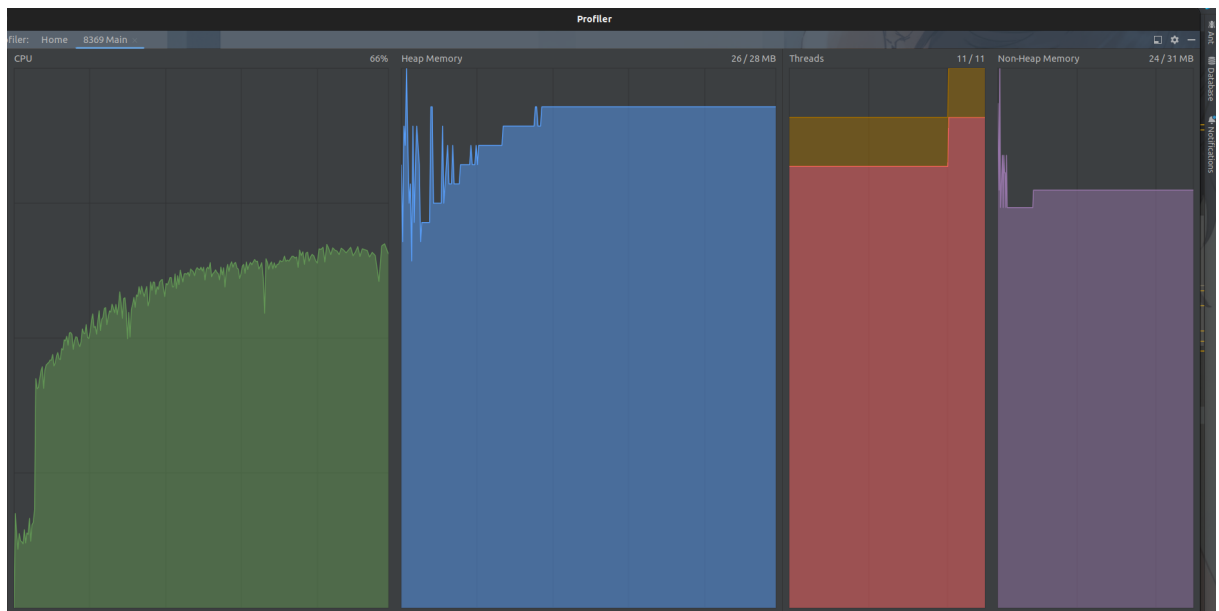
```
while (true) {
    WebResponse response = sc.getResponse(request);
    System.out.println("Count: " + number++ + response);
    HttpUnitOptions.clearScriptErrorMessages();
}
```

3. Описание алгоритма действий, который позволил выявить и локализовать проблему.

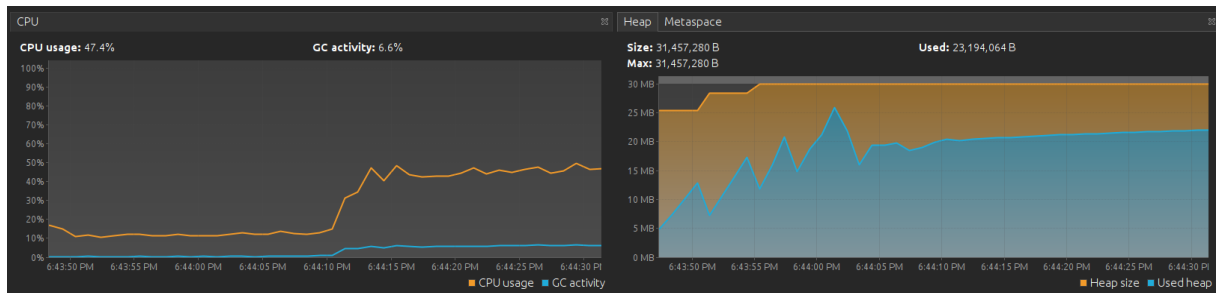
1. Возьмем на основной файл (Main.java) и заметим, что данная программа просто обрабатывает входящие запросы в бесконечном цикле.

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

2. Также в этом цикле заметим “засыпание” - Thread.sleep(200) после каждого обработанного запроса. Это выглядит как один из параметров, явно замедляющих нашу программу, так что удалим эту строчку.
3. Запустим программу в IDE Idea предварительно установив ограничение на максимальный размер кучи 12МБ (Xmx12m).
4. Откроем профилировщик и внимательно проанализируем его графики.



5. Заметим по графикам, что нагрузка на процессор сильно увеличивается и программа ощутимо замедляется при приближении размера кучи к установленному максимуму.
6. Перезапускаем программу и анализируем её с помощью VisualVM. На графиках VisualVM мы наблюдаем такое же поведение.



7. Также через какое-то время программа падает с ошибкой `OutOfMemoryError`.
8. Создадим Heap Dump и проанализируем объекты в памяти

Instances by Size [view all]

<code>java.lang.Object[]</code> #8502 : 31,618 items	126,488 B	(0.7%)
<code>char[]</code> #4746 : count: 27670 [_response = com.meterware.servletunit	16,400 B	(0.1%)
<code>char[]</code> #1866 : ...	16,400 B	(0.1%)
<code>char[]</code> #4584 : ...	16,400 B	(0.1%)
<code>char[]</code> #5553 [GC root - Java frame] : ...	16,400 B	(0.1%)

9. Исходя из этого мы видим, какие объекты требуют наибольшее количество памяти.

10. Определим, в каком классе хранятся эти объекты

```

java.lang.Object[] #8502 : 31,618 items
  <items>
  <references>
    elementData in java.util.ArrayList#19 : 27,670 elements
      static _errorMessages in class com.meterware.httpunit.javascript.Javascript : JavaScript

```

11. Найдём объекты в исходном коде и проанализируем

```

private static ArrayList _errorMessages = new ArrayList();

static boolean ...
JavaScript.java 60 _errorMessages.clear();
JavaScript.java 65 return (String[]) _errorMessages.toArray( new String[ _errorMessages.size() ] );
JavaScript.java 204 _errorMessages.add( errorMessage );

```

12. Как видно, данный объект используется в трёх местах: для добавления новых сообщений об ошибках, для очистки сообщения об ошибках и получения всех сообщений (геттер).
13. При этом, можно проследить, где вызываются методы для получения всех сообщений об ошибке и для очистки их.

```

public String[] getErrorMessages() {
    return JavaScript.getErrorMessages();
}

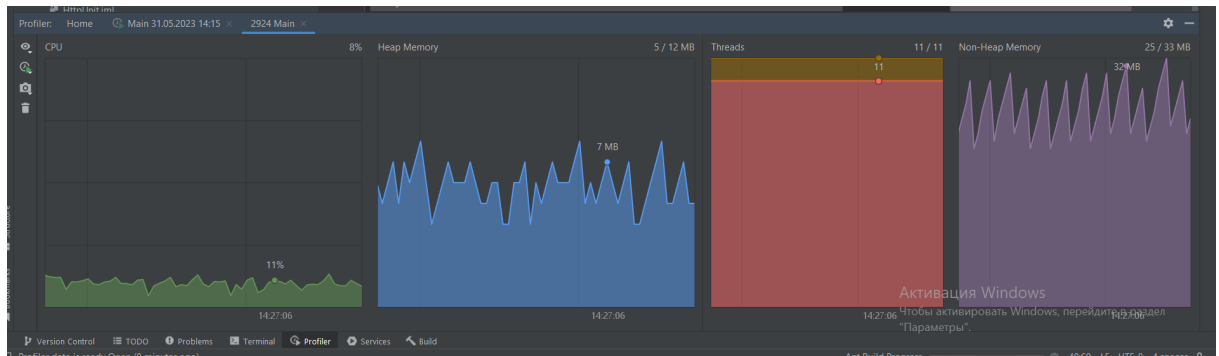
public static String[] getScriptErrorMessages() {
    return getScriptingEngine().getErrorMessages();
}

```

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

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

14. Как видно, очищающий метод нигде не вызывается, а содержимое данной коллекции нигде в программе не используется.
15. Тогда можно вызывать метод для очистки коллекции после каждого запроса, чтобы не хранить ненужные данные.
16. Установим размер кучи на 12 МБайт и снова проанализируем программу через профилировщик IDE IDEA



17. Как видно, проблема с утечкой памяти и падением производительности была устранена и памяти хватает для корректной работы программы.

Репозиторий с кодом лабораторной работы:

https://github.com/Hyperb0rean/itmo_mamose

Выводы

В результате выполнения этой лабораторной работы мы научились мониторить и профилировать. А ещё мы были очень рады, потому что это последняя лабораторная. В общем, теперь мы знаем всё про методы и средства программной инженерии и можем устраиваться в Yandex на позицию Senior Java Developer.