

- ① Ты умрешь таким же, каким жил:  
жалким и невежественным.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное  
образовательное учреждение высшего образования  
“Национальный исследовательский университет ИТМО”

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ  
И КОМПЬЮТЕРНОЙ ТЕХНИКИ

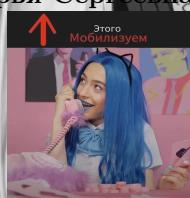
### ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине  
“МЕТОДЫ И СРЕДСТВА ПРОГРАММНОЙ ИНЖЕНЕРИИ”

Вариант: ~~1000-7?~~

выполнили:

Студенты группы Р32311  
Птицын Максим Евгеньевич  
Воронина Дарья Сергеевна



Преподаватель  
Бострикова Дарья Константиновна

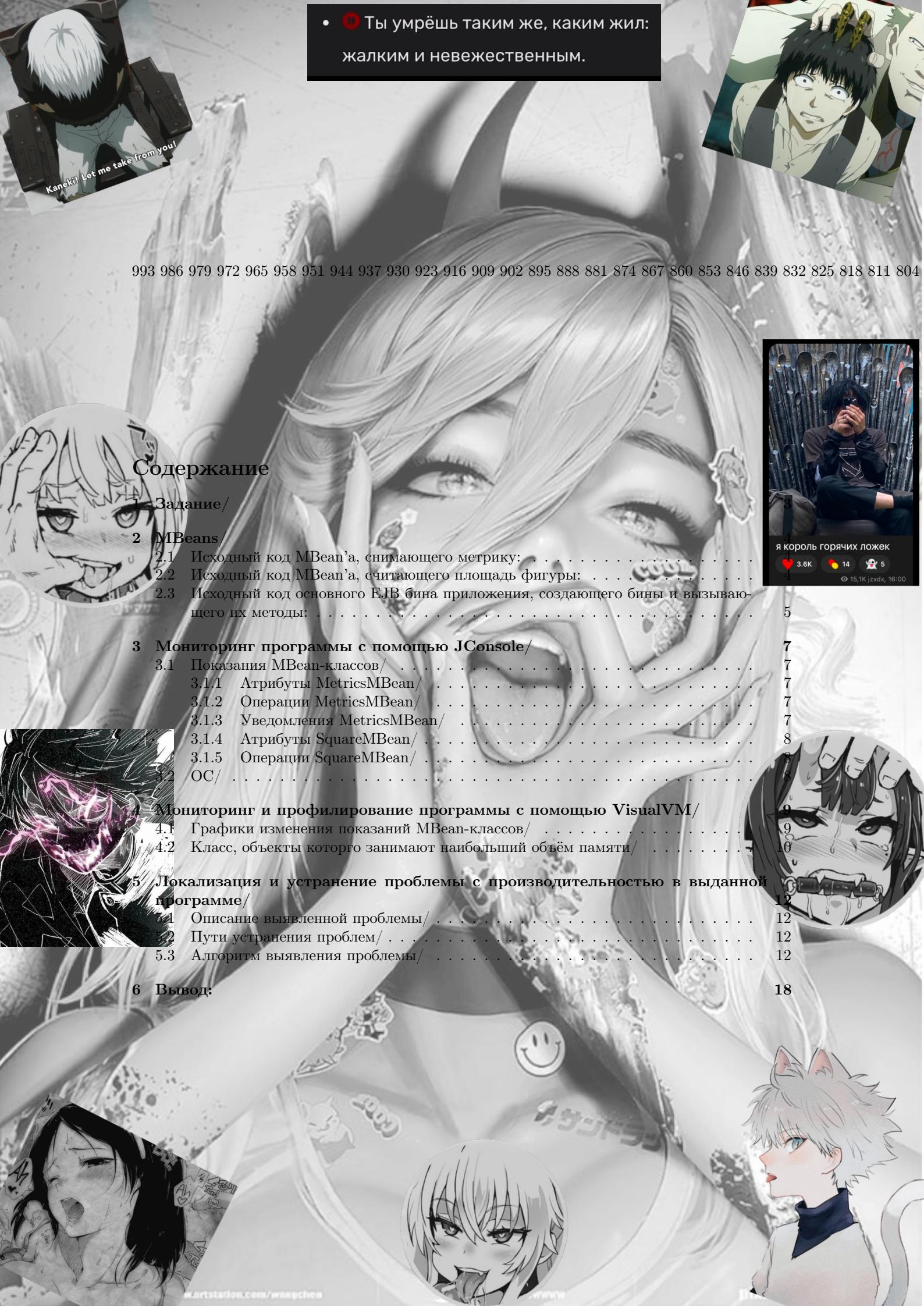


- Ты умрешь таким же, каким жил:  
жалким и невежественным.

993 986 979 972 965 958 951 944 937 930 923 916 909 902 895 888 881 874 867 860 853 846 839 832 825 818 811 804

## Содержание

<b>1 Задание/</b>	5
<b>2 MBeans/</b>	5
2.1 Исходный код MBean'a, снимающего метрику: . . . . .	5
2.2 Исходный код MBean'a, считающего площадь фигуры: . . . . .	5
2.3 Исходный код основного ЕJB бина приложения, создающего бины и вызывающего их методы: . . . . .	5
<b>3 Мониторинг программы с помощью JConsole/</b>	7
3.1 Показания MBean-классов/ . . . . .	7
3.1.1 Атрибуты MetricsMBean/ . . . . .	7
3.1.2 Операции MetricsMBean/ . . . . .	7
3.1.3 Уведомления MetricsMBean/ . . . . .	7
3.1.4 Атрибуты SquareMBean/ . . . . .	8
3.1.5 Операции SquareMBean/ . . . . .	8
OC/ . . . . .	8
<b>4 Мониторинг и профилирование программы с помощью VisualVM/</b>	8
4.1 Графики изменения показаний MBean-классов/ . . . . .	9
4.2 Класс, объекты которого занимают наибольший объём памяти/ . . . . .	9
<b>5 Локализация и устранение проблемы с производительностью в выданной программе/</b>	10
5.1 Описание выявленной проблемы/ . . . . .	12
5.2 Пути устранения проблем/ . . . . .	12
5.3 Алгоритм выявления проблемы/ . . . . .	12
<b>6 Вывод:</b>	18



- Ты умрёшь таким же, каким жил:  
жалким и невежественным.



## 1 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, не попадающих в область/ В случае, если пользователь совершил 4 "ромаха" подряд, 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 имя и версию ОС, под управлением которой работает 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/

Студент должен обеспечить возможность воспроизведения процесса поиска и локализации проблемы по требованию преподавателя/



- Ты умрешь таким же, каким жил:  
жалким и невежественным.

## 2 MBeans /

### 2.1 Исходный код MBean'a, снимающего метрику:

```
public class Metrics extends NotificationBroadcasterSupport implements MetricsMXBean, Serializable {
    private int hitsCount = 0;
    private int missedHitsCount = 0;
    private int missedHitsStreakCount = 0;
    int sequenceNumber = 1;

    public Metrics() {
    }

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

    @Override
    public int missedAndStreakInc() {
        missedHitsCount++;
        missedHitsStreakCount++;
        if (missedHitsStreakCount % 4 == 0) {
            final Notification notification = new Notification("4 miss streak", this, sequenceNumber++, "4 miss streak.");
            sendNotification(notification);
        }
        return missedHitsStreakCount;
    }

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

    public int getHitsCount() {
        return hitsCount;
    }

    public int getMissedHitsCount() {
        return missedHitsCount;
    }

    public int getMissedHitsStreakCount() {
        return missedHitsStreakCount;
    }
}
```



### 2.2 Исходный код MBean'a, считающего площадь фигуры:

```
public class Square implements SquareMXBean, Serializable {
    double lastSquare = 0;

    public Square() {
    }

    @Override
    public double calculateSquare(double r) {
        double squareSquare = r * 0.5 * r * 1;
        double triangleSquare = (r * (r / 2)) / 2;
        double circleSquare = (Math.PI * r * r) / 4;
        double totalSquare = squareSquare + triangleSquare + circleSquare;
        lastSquare = totalSquare;
        return totalSquare;
    }

    @Override
    public double getLastSquare() {
        return lastSquare;
    }
}
```

- ❶ Ты умрешь таким же, каким жил:  
жалким и невежественным.

## 2.3 Исходный код основного ЕJB бина приложения, создающего бины и вызывающего их методы:

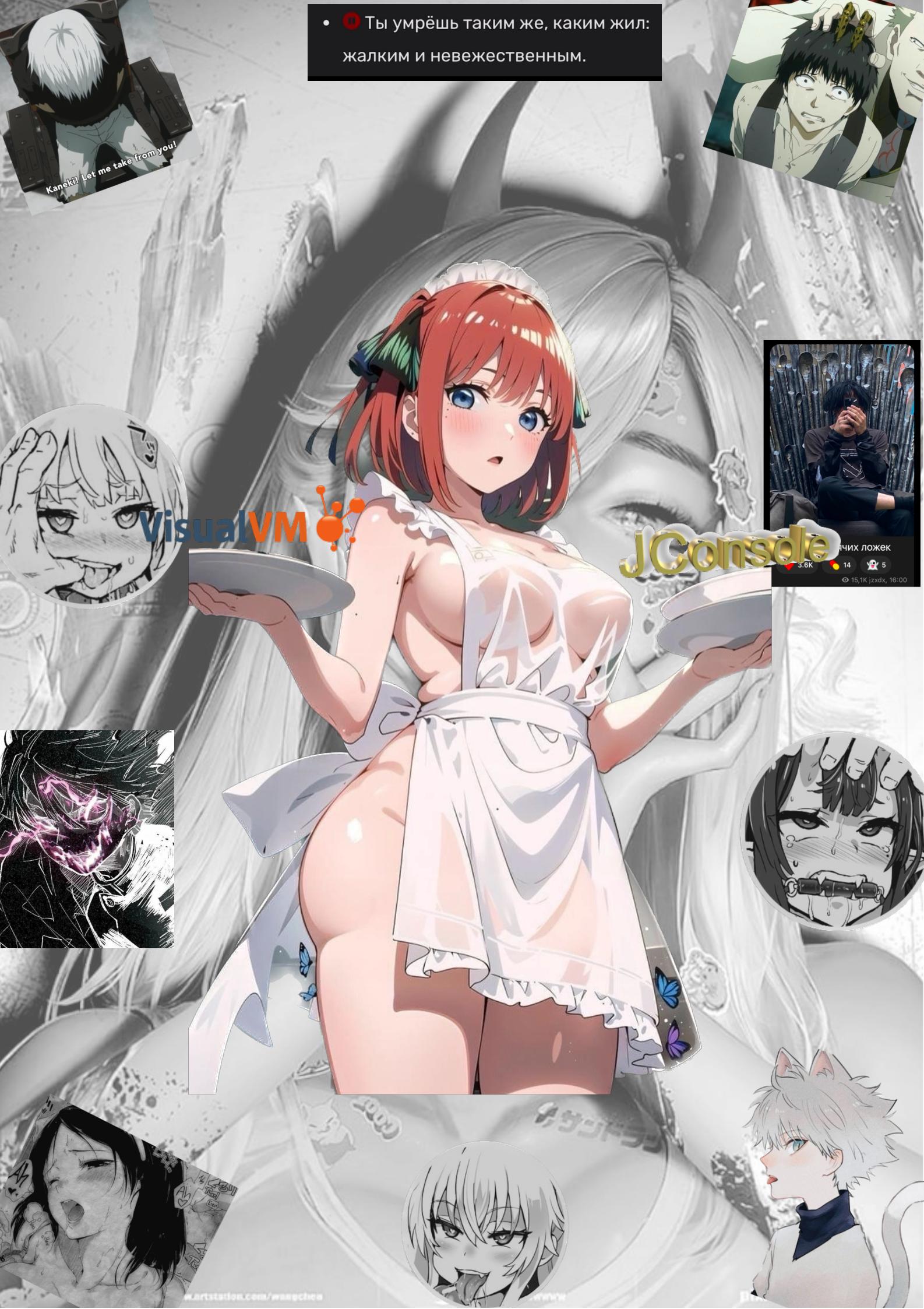
```
public class AppBean {  
  
    private MBeanServer mbs;  
    MetricsMXBean metricsMXBean;  
    SquareMXBean squareMXBean;  
  
    public AppBean(List<Result> results) {  
        this.mbs = ManagementFactory.getPlatformMBeanServer();  
        this.metricsMXBean = new Metrics();  
        this.squareMXBean = new Square();  
        ObjectName metricsName = null;  
        ObjectName squareName = null;  
        try {  
            metricsName = new ObjectName("AppBean:name=metricsMXBean");  
            squareName = new ObjectName("AppBean:name=squareMXBean");  
            mbs.registerMBean(metricsMXBean, metricsName);  
            mbs.registerMBean(squareMXBean, squareName);  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
  
    private void resultMBeansHandle(Result result) {  
        metricsMXBean.hitsInc();  
        if (result.isMatch()) {  
            metricsMXBean.clearMissedStreak();  
        } else {  
            metricsMXBean.missedAndStreakInc();  
        }  
        double currSquare = squareMXBean.calculateSquare(result.getR());  
    }  
}
```



трахать сук  
пить сок



- ❶ Ты умрёшь таким же, каким жил:  
жалким и невежественным.



- Ты умрешь таким же, каким жил:  
жалким и невежественным.

### 3 Мониторинг программы с помощью JConsole/

#### 3.1 Показания MBean-классов/

##### 3.1.1 Атрибуты MetricsMBean/

Screenshot of the JConsole Attribute values table for the metricsMXBean MBean:

Name	Value
HitsCount	11
MissedHitsCount	10
MissedHitsStreakCount	8

##### 3.1.2 Операции MetricsMBean/

Screenshot of the JConsole Operation invocation dialog for the hitsInc operation:

int hitsInc ()

Operation return value dialog showing the result: 12

OK

##### 3.1.3 Уведомления MetricsMBean/

Screenshot of the JConsole Notification buffer table for the metricsMXBean MBean:

TimeStamp	Type	UserData	SeqNum	Message	Event	Source
16:30:22:849	4 miss streak		2	4 miss streak.	jax.mana...	AppBean:name=...

- Ты умрёшь таким же, каким жил:  
жалким и невежественным.

### 3.1.4 Атрибуты SquareMBean/

Attribute values

Name	Value
LastSquare	6.141592653589793

### 3.1.5 Операции SquareMBean/

Operation invocation

double calculateSquare ( p0 12 )

MBeanOperationInfo

Name	Value
i	221.09733552923257

OK

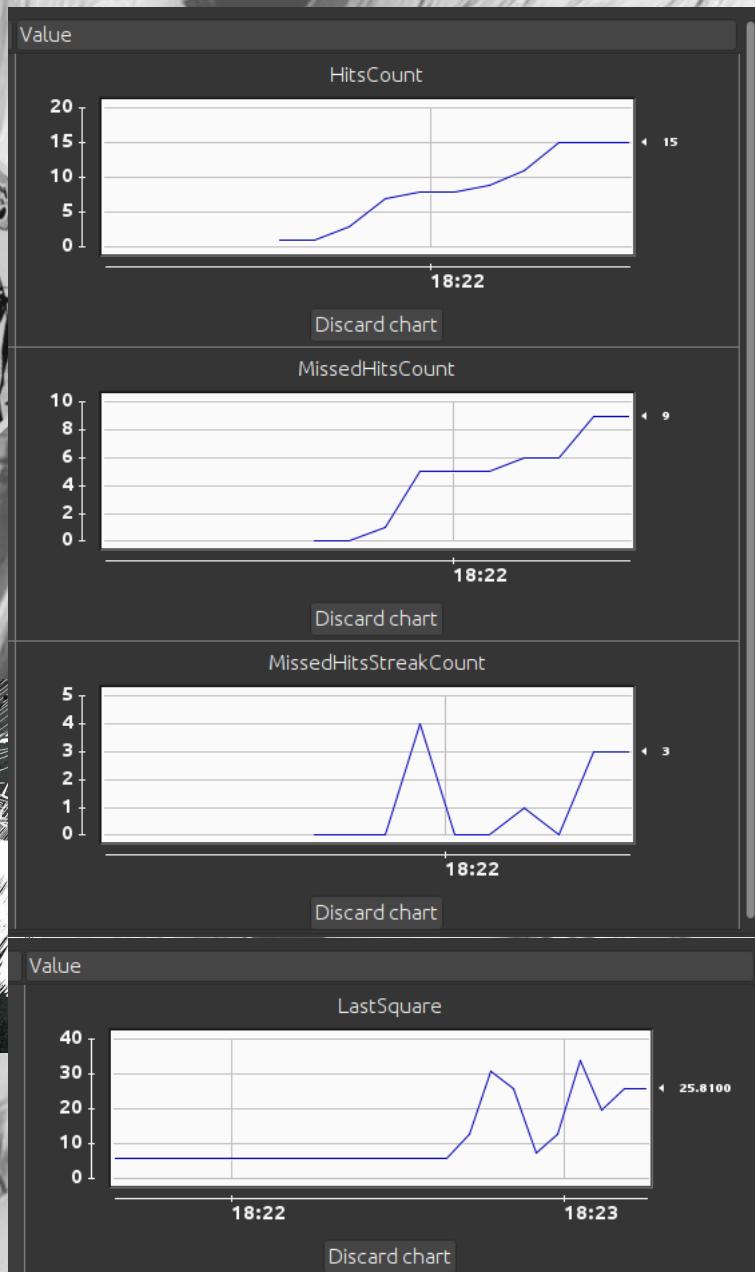
## 3.2 OC/

Operating System: Linux 5.19.0-41-generic

- Ты умрёшь таким же, каким жил:  
жалким и невежественным.

## 4 Мониторинг и профилирование программы с помощью VisualVM /

### 4.1 Графики изменения показаний MBean-классов



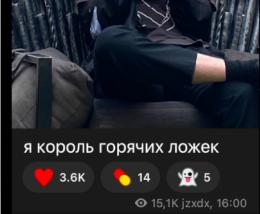
-  Ты умрёшь таким же, каким жил:  
жалким и невежественным.

#### 4.2 Класс, объекты которого занимают наибольший объём памяти/

Name	Live Bytes	Live Objects	Allocated Objects	Genera...
proto.alaba3.util.Result	704 B (65.2%)	11 (45.8%)	11 (45.8%)	1 ^
proto.alaba3.util.AppBean	144 B (13.3%)	2 (8.3%)	2 (8.3%)	1
proto.alaba3.util.ResultDao\$\$view2	48 B (4.4%)	2 (8.3%)	2 (8.3%)	1
proto.alaba3.util.ResultServiceRealiz	48 B (4.4%)	2 (8.3%)	2 (8.3%)	1
proto.alaba3.util.ResultServiceRealiz	48 B (4.4%)	3 (12.5%)	3 (12.5%)	1
proto.alaba3.mbeans.Metrics	40 B (3.7%)	1 (4.2%)	1 (4.2%)	1
proto.alaba3.util.ResultDao	32 B (3%)	2 (8.3%)	2 (8.3%)	1
proto.alaba3.mbeans.Square	16 B (1.5%)	1 (4.2%)	1 (4.2%)	1
proto.alaba3.util.Result\$HibernatePr	0 B (0%)	0 (0%)	0 (0%)	0

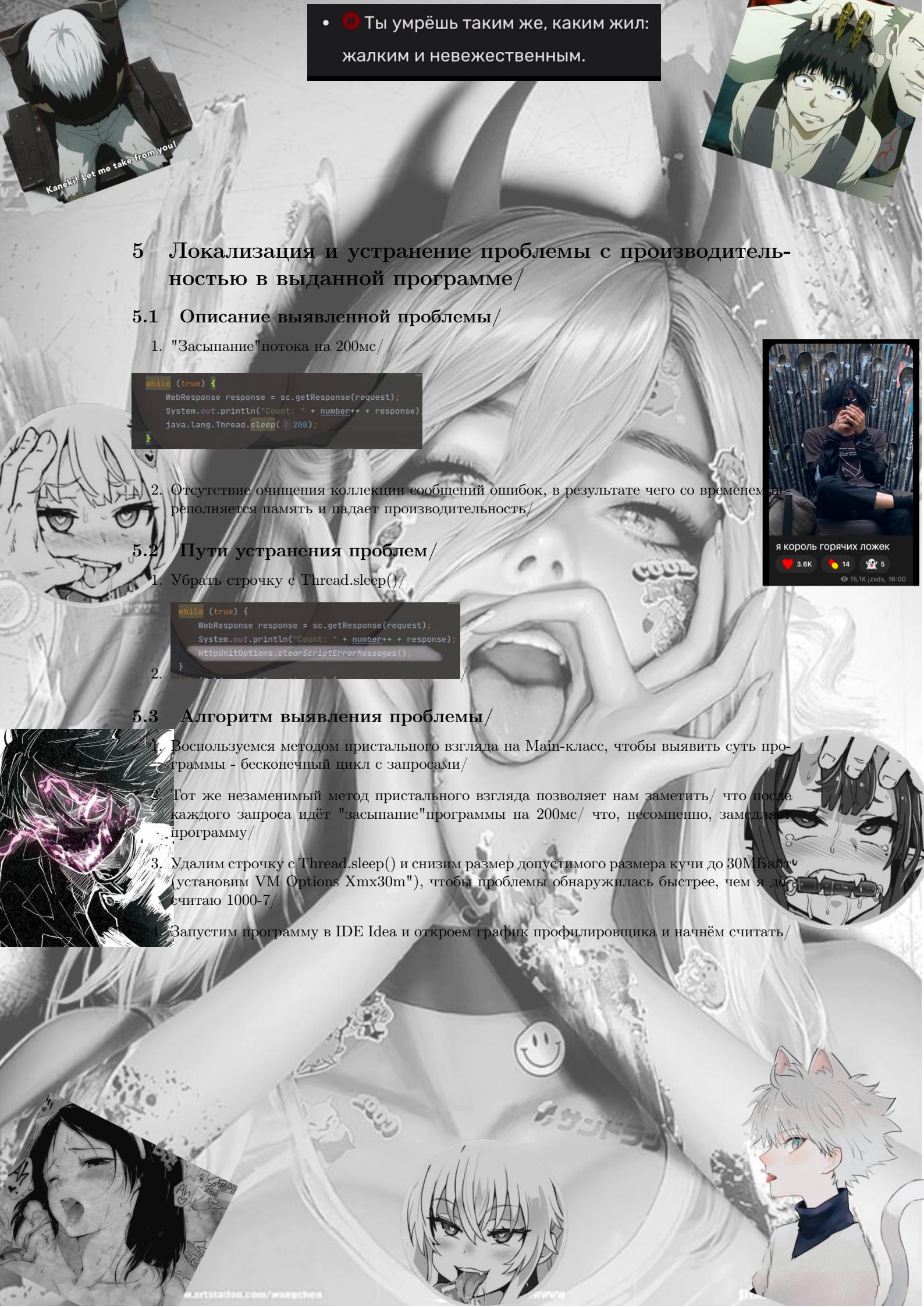
Объекты этого класса хранятся в пользовательском классе AppBean/

```
@ManagedBean(name = "appbean")
public class AppBean {
    3 usages
    private List<Result> results;
```



• ❶ Ты умрёшь таким же, каким жил:  
жалким и невежественным.



- 
- Ты умрешь таким же, каким жил:  
жалким и невежественным.

## 5 Локализация и устранение проблемы с производительностью в выданной программе/

### 5.1 Описание выявленной проблемы/

1. "Засыпание" потока на 200мс/

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

2. Отсутствие очищения коллекции сообщений ошибок, в результате чего со временем не заполняется память и падает производительность/

### 5.2 Пути устранения проблем/

1. Убрать строчку с Thread.sleep()//

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

2. //

### 5.3 Алгоритм выявления проблемы/

1. Воспользуемся методом пристального взгляда на Main-класс, чтобы выявить суть программы - бесконечный цикл с запросами/
2. Тот же незаменимый метод пристального взгляда позволяет нам заметить/ что после каждого запроса идёт "засыпание" программы на 200мс/ что, несомненно, замедляет программу/
3. Удалим строчку с Thread.sleep() и снизим размер допустимого размера кучи до 30МБайт (установим VM Options Xmx30m"), чтобы проблема обнаружилась быстрее, чем я додумался считать 1000-7/
4. Запустим программу в IDE Idea и откроем график профилировщика и начнём считать/



• Ты умрешь таким же, каким жил:  
жалким и невежественным.

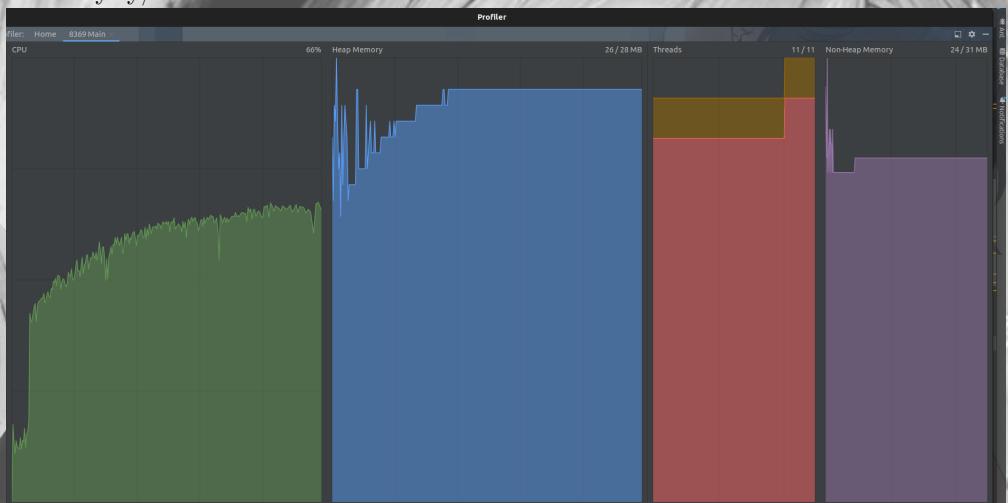


993	825	657	489	321	153
986	818	650	482	314	146
979	811	643	475	307	139
972	804	636	468	300	132
965	797	629	461	293	125
958	790	622	454	286	118
951	783	615	447	279	111
944	776	608	440	272	104
937	769	601	433	265	97
930	762	594	426	258	90
923	755	587	419	251	83
916	748	580	412	244	76
909	741	573	405	237	69
902	734	566	398	230	62
895	727	559	391	223	55
888	720	552	384	216	48
881	713	545	377	209	41
874	706	538	370	202	34
867	699	531	363	195	27
860	692	524	356	188	20
853	685	517	349	181	13
846	678	510	342	174	6
839	671	503	335	167	-1
832	664	496	328	160	

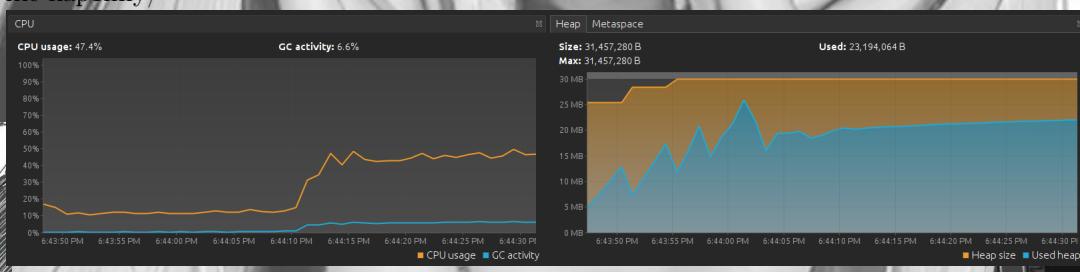


•  Ты умрешь таким же, каким жил:  
жалким и невежественным.

5. Воспользуемся тем же замечательным методом пристального взгляда для обнаружения недопустимого увеличения нагрузки на процессор и сильном замедлении программы (примерно 1 запрос в 30 секунд) при приближении размера кучи к установленному максимуму /



6. Запустим программу заново и проанализируем её с помощью VisualVM. Видим там ту же картину /



7. Дополнительно, при мониторинге программы через VisualVM, программа вскоре завершится, выдав OutOfMemoryError /



-  Ты умрёшь таким же, каким жил:  
жалким и невежественным.



- Ты умрешь таким же, каким жил:  
жалким и невежественным.

8. Создадим Heap Dump и проанализируем объекты в памяти/

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

9. Нетрудно заметить, какие объекты занимают больше всего места в памяти/



ль горячих ложек

10. Определим, в каком классе они хранятся/

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 их в коде и проанализируем/

4 usages

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

All Places

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

12. Видим, что на данное поле написан метод, добавляющий ошибки в него, очищающий коллекцию и геттер/



- Ты умрешь таким же, каким жил:  
жалким и невежественным.

13. При этом, можно проследить вызовы методов геттера и очищающего/



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

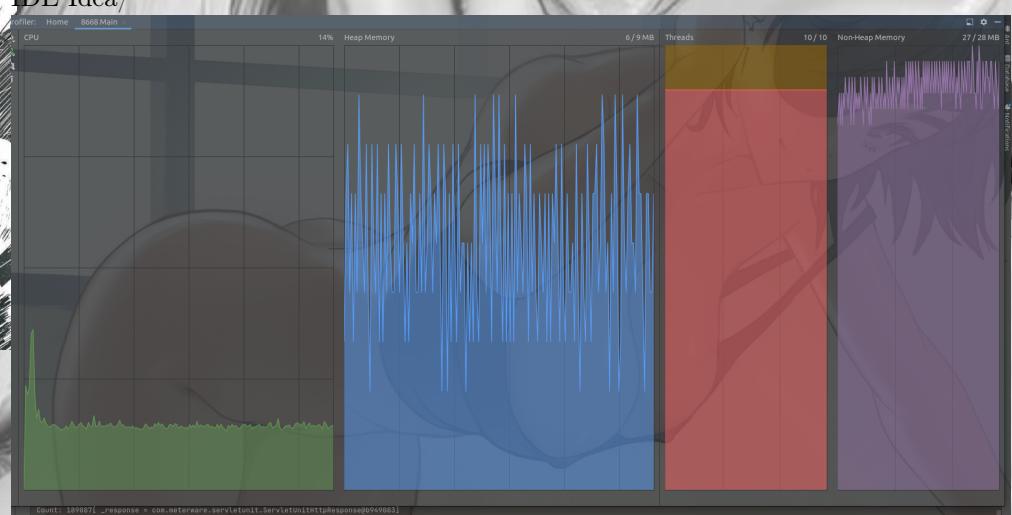
no usages
public static String[] getScriptErrorMessages() {
    return getScriptingEngine().getErrorMessages();
}

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

14. Нетрудно заметить, что очищающий метод нигде не вызывается/ а так же содержимое данной коллекции нигде в программе не используется/

15. Будем вызывать метод очистки коллекции после каждого запроса для решения проблемы с переполнением памяти/

16. Уменьшим размер кучи до 10МБайт промониторим программу через профилировщик IDE Idea/



- ① Ты умрешь таким же, каким жил:  
жалким и невежественным.

17. В очередной раз воспользуемся великолепным методом пристального взгляда и увидим, что проблема с производительностью устранена: выделенной памяти хватает для корректной работы программы / а нагрузка на процессор в пределах нормы/

## 6 Вывод:

В ходе программы мы познакомились с MBeans / утилитами JConsole / VisualVM и встречающими в IDE профилировщиками / Можно отметить, что MBeans является удобной альтернативой дебаггингу для мониторинга работы сложных систем / а изученные утилиты позволят подробно проанализировать работу программы и локализовать возможные ошибки в работе программы и её быстродействии/

