**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

по лабораторной работе №4

Вариант №367958

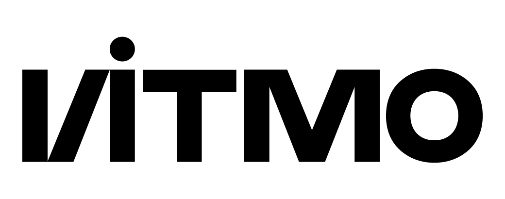
по дисциплине «**Основы программной инженерии**»

Автор: Баянов Равиль Динарович

Факультет: ПИиКТ

Группа: P3234

Преподаватель: Кулинич Ярослав Вадимович



Санкт-Петербург, 2024

Оглавление

[Задание 3](#_Toc168100316)

[Исходный код MBean-классов 4](#_Toc168100317)

[Интерфейс MBean 4](#_Toc168100318)

[Методы реализуемого интерфейса MBean 4](#_Toc168100319)

[Класс-инструмент для регистрации MBean 5](#_Toc168100320)

[JConsole 7](#_Toc168100321)

[Показания MBean-классов 7](#_Toc168100322)

[Время с момента запуска JVM 9](#_Toc168100323)

[VisualVM 10](#_Toc168100324)

[График изменения показаний MBean-классов 10](#_Toc168100325)

[Имя потока, потребляющего наибольший процент времени CPU 11](#_Toc168100326)

[Проблемы с производительностью в программе 12](#_Toc168100327)

[Список используемой литературы 15](#_Toc168100328)

[Вывод 16](#_Toc168100329)

# Задание

1. Для своей программы из [лабораторной работы #3](https://se.ifmo.ru/courses/web#lab3) 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 2 "промаха" подряд, разработанный MBean должен отправлять оповещение об этом событии.
* MBean, определяющий средний интервал между кликами пользователя по координатной плоскости.

2. С помощью утилиты JConsole провести мониторинг программы:

* Снять показания MBean-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 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](https://se.ifmo.ru/documents/10180/189115/HttpUnit.tar.gz/7bf1032e-d16e-be85-c71b-dbe73c0178ba?t=1651168887037&download=true). 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.

# Исходный код MBean-классов

## Интерфейс MBean

|  |
| --- |
| 1. **package** ru.ravvcheck.web3lab.model.services; 3. **public** **interface** ResultManagerMBean { 4. **long** getDotsCount(); 5. **long** getMissedDotsCount(); 7. [Double](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+double) searchAverageClickInterval(); 8. } |

## Методы реализуемого интерфейса MBean

|  |
| --- |
| 1. @Override 2. **public** **long** getMissedDotsCount() { 3. **return** getDotsCount() - **this**.results.stream() 4. .filter(Result::isHit) 5. .count(); 6. } 8. @Override 9. **public** **long** getDotsCount() { 10. **return** **this**.results.size(); 11. } 13. @Override 14. **public** [Double](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+double) searchAverageClickInterval() { 15. DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm:ss dd.MM.yyyy"); 16. List<ZonedDateTime> zonedDateTimes = results.stream() 17. .map(result -> ZonedDateTime.of(LocalDateTime.parse(result.getCurrentTime(), formatter), ZoneId.systemDefault())) 18. .collect(Collectors.toList()); 19. List<Long> differences = zonedDateTimes.stream() 20. .skip(1) 21. .map(time -> time.toInstant().toEpochMilli() - zonedDateTimes.get(zonedDateTimes.indexOf(time) - 1).toInstant().toEpochMilli()) 22. .collect(Collectors.toList()); 23. **return** differences.stream() 24. .mapToDouble([Long](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+long)::doubleValue) 25. .average() 26. .orElse(0); 27. } 29. @Override 30. **public** MBeanNotificationInfo[] getNotificationInfo() { 31. [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string)[] types = **new** [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string)[]{AttributeChangeNotification.ATTRIBUTE\_CHANGE}; 32. [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) name = AttributeChangeNotification.**class**.getName(); 33. [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) description = "Miss notification"; 34. MBeanNotificationInfo info = **new** MBeanNotificationInfo(types, name, description); 35. **return** **new** MBeanNotificationInfo[]{info}; 36. } |

## Класс-инструмент для регистрации MBean

|  |
| --- |
| 1. **package** ru.ravvcheck.web3lab.utils; 3. **import** jakarta.servlet.ServletContextListener; 5. **import** javax.management.\*; 6. **import** java.lang.management.ManagementFactory; 7. **import** java.util.HashMap; 9. **public** **class** MBeanRegistryUtil **implements** ServletContextListener { 10. **private** **static** **final** HashMap<Class<?>, ObjectName> beans = **new** HashMap<>(); 11. **public** **static** **void** registerBean([Object](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+object) bean, [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) name) { 12. **try** { 13. [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) type = bean.getClass().getSimpleName(); 14. ObjectName objectName = **new** ObjectName([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string).format("ru.ravvcheck.web3lab:type=%s,name=%s", type, name)); 15. MBeanServer server = ManagementFactory.getPlatformMBeanServer(); 16. server.registerMBean(bean, objectName); 17. } **catch** (MalformedObjectNameException | NotCompliantMBeanException | InstanceAlreadyExistsException | MBeanRegistrationException e) { 18. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println(e.getMessage()); 19. } 20. } 22. **public** **static** **void** unregisterBean([Object](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+object) bean) { 23. **if** (!beans.containsKey(bean.getClass())) { 24. **throw** **new** [IllegalArgumentException](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+illegalargumentexception)("Bean not registered"); 25. } 26. ObjectName objectName = beans.get(bean.getClass()); 27. MBeanServer server = ManagementFactory.getPlatformMBeanServer(); 28. **try** { 29. server.unregisterMBean(objectName); 30. } **catch** (InstanceNotFoundException | MBeanRegistrationException e) { 31. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println(e.getMessage()); 32. } 33. } 34. } |

# JConsole

## Показания MBean-классов

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание

## Время с момента запуска JVM

Изображение выглядит как текст, снимок экрана, диаграмма, дисплей

Автоматически созданное описание

# VisualVM

## График изменения показаний MBean-классов

Изображение выглядит как текст, снимок экрана, диаграмма, дисплей

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

## Имя потока, потребляющего наибольший процент времени CPU

RMI TCP Connection(5)-192.168.31.107

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

# Проблемы с производительностью в программе

Запустим программу с помощью Ant и будем мониторить программу с помощью VisualVM

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Заметим, что наша программа, а именно поток main основную часть времени находится в состоянии Sleeping. Попытаемся найти место, которое снижает скорость выполнения нашей программы. Заметим, что строчка java.lang.Thread.sleep(200); сильно задерживает программу. Следовательно, закомментируем её, тем самым ускорим нашу программу.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Запустим программу снова. И увидим, что наш поток main всегда running.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Установим в project.properties параметр для размера кучи в 12мб. И запустим программу снова.



Заметим, что объём свободной памяти в кучи стремительно уменьшается. Констатируем факт утечки памяти в нашей программе.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Со временем получим ошибку, связанную с нехваткой памяти.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Посмотрим на Heap Dump и попробуем определить, что в программе занимает больше всего памяти. Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Обнаруживается массив со String (с сообщениями об ошибках).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

С помощью удобного поиска в IDEA через ctrl направляемся вот к такому методу, который как мы видим нигде не вызывается (а должен!!!)

Решим проблему с утечкой памяти вызовом этого метода в main сразу после выполнения запроса. И снова запустим программу.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, диаграмма

Автоматически созданное описание

Ошибка исправлена!

# Список используемой литературы

* Клименков С. В., Цопа Е. А., Козин И. О. Конспект лекций по дисциплине “Основы программной инженерии”, 2021. Электронный ресурс / Режим доступа: <https://se.ifmo.ru/documents/10180/671657/%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D0%B8+%D0%BF%D0%BE+%D0%9E%D0%9F%D0%98+v1.3.3.pdf/50ce1e06-00d9-4900-be9c-a3316a746d6d>

# Вывод

При выполнении данной лабораторной работы я узнал много про JMX, про мониторинг и профилирование программ. Написал небольшой собственный MBean и отмерил его показания с помощью таких программ как: JConsole и VisualVM. Также научился пользоваться VisualVM для обнаружения утечек памяти и их устранения.