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: **811**

**Преподаватель**:   
Карташев Владимир Сергеевич

**Выполнил**:

Кирячек Тимофей

**Группа**:

Р3209

Санкт-Петербург, 2025

Оглавление

[1. Задание 3](#_Toc197885054)

[2. Выполнение 4](#_Toc197885055)

[1. Исходный код разработанных MBean 4](#_Toc197885056)

[2. JConsole 7](#_Toc197885057)

[3. VisualVM 11](#_Toc197885058)

[4. Исследование программы на утечки памяти 16](#_Toc197885059)

[3. Вывод 22](#_Toc197885060)

# Задание

Вариант: **811**

|  |
| --- |
|  |

# Выполнение

## 1. Исходный код разработанных MBean

* web.beans.AreaMBean.java

*package* web.beans;  
  
*public interface AreaMBean* {  
 *double* getArea();  
 *void* updateStats(*double* r);  
}

* web.beans.ResultsMBean.java

*package* web.beans;  
  
*public interface ResultsMBean* {  
 *int* getTotalAttempts();  
 *int* getTotalMisses();  
 *void* updateStats(*boolean* hit);  
}

* web.beans.Area.java

*package* web.beans;  
  
*import* jakarta.enterprise.context.ApplicationScoped;  
*import* jakarta.inject.Named;  
  
*import* java.io.*Serializable*;  
  
@Named("area")  
@ApplicationScoped  
*public class* Area *implements Serializable*, *AreaMBean* {  
 *private double* area = 0;  
  
 @Override  
 *public double* getArea() {  
 *return* area;  
 }  
  
 @Override  
 *public void* updateStats(*double* r) {  
 area = ((Math.***PI*** \* Math.*pow*(r / 2, 2) / 4) + ((r \* r/2) / 2) + (r \* r/2));  
 }  
  
}

* web.beans.Results.java

*package* web.beans;  
  
*import* jakarta.enterprise.context.ApplicationScoped;  
*import* jakarta.inject.Named;  
  
*import* javax.management.\*;  
*import* java.io.*Serializable*;  
*import* java.util.concurrent.atomic.AtomicInteger;  
  
@Named("results")  
@ApplicationScoped  
*public class* Results *implements Serializable*, *NotificationBroadcaster*, *ResultsMBean* {  
 *private int* sequenceNumber = 0;  
  
 *private final* AtomicInteger totalAttempts = *new* AtomicInteger();  
 *private final* AtomicInteger totalMisses = *new* AtomicInteger();  
  
 *private final* NotificationBroadcasterSupport broadcaster = *new* NotificationBroadcasterSupport();  
  
 @Override  
 *public int* getTotalAttempts() {  
 *return* totalAttempts.get();  
 }  
  
 @Override  
 *public int* getTotalMisses() {  
 *return* totalMisses.get();  
 }  
  
 @Override  
 *public void* updateStats(*boolean* hit) {  
 totalAttempts.incrementAndGet();  
 *if* (!hit) {  
 totalMisses.incrementAndGet();  
 }  
  
 *if* (totalAttempts.get() % 5 == 0) {  
 broadcaster.sendNotification(*new* Notification(  
 "Total dots count is multiple of 5",  
 getClass().getSimpleName(),  
 sequenceNumber++,  
 "The total count of user-set dots is now multiple of 5!"  
 ));  
 }  
 }  
  
 @Override  
 *public void* addNotificationListener(*NotificationListener* listener, *NotificationFilter* filter, Object handback) *throws* IllegalArgumentException {  
 broadcaster.addNotificationListener(listener, filter, handback);  
 }  
  
 @Override  
 *public void* removeNotificationListener(*NotificationListener* listener) *throws* ListenerNotFoundException {  
 broadcaster.removeNotificationListener(listener);  
 }  
  
 @Override  
 *public* MBeanNotificationInfo[] getNotificationInfo() {  
 String[] types = *new* String[] { AttributeChangeNotification.***ATTRIBUTE\_CHANGE*** };  
 String name = AttributeChangeNotification.*class*.getName();  
 String description = "Miss notification";  
 MBeanNotificationInfo info = *new* MBeanNotificationInfo(types, name, description);  
 *return new* MBeanNotificationInfo[] { info };  
 }  
}

* web.utils.MBeanRegistry.java

*package* web.utils;  
  
*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* MBeanRegistry *implements ServletContextListener* {  
 *private final Map*<Class<?>, ObjectName> ***beans*** = *new* HashMap<>();  
  
 *public void* registerBean(Object bean, String name) {  
 *try* {  
 *var* domain = bean.getClass().getPackageName();  
 *var* type = bean.getClass().getSimpleName();  
 *var* 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* {  
 ManagementFactory.*getPlatformMBeanServer*().unregisterMBean(***beans***.get(bean.getClass()));  
 } *catch* (InstanceNotFoundException | MBeanRegistrationException ex) {  
 ex.printStackTrace();  
 }  
 }  
}

## 2. JConsole

Показания MBean-классов, разработанных в ходе выполнения задания 1:

* **Метаданные** Mbean’ов:

|  |
| --- |
|  |
|  |

* **Показания** Mbean’ов:

|  |
| --- |
|  |
|  |

* **VM Summary** и имя и версия ОС, под управлением которой работает JVM.

|  |
| --- |
| Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение  Контент, сгенерированный ИИ, может содержать ошибки. |

Это FreeBSD 14.1-STABLE

* **Графики**:

|  |
| --- |
|  |
|  |

* **Уведомления**:

|  |
| --- |
|  |
|  |

* **Выводы по результатам мониторинга**:

В ходе мониторинга с использованием утилиты JConsole было

определено, что:

* Узнать имя и версию ОС, под которой работает JVM можно в разделе VM Summary/Operation System.
* MBean Area и Results были успешно разработаны и

зарегистрированы. Когда общее количество точек становится кратно 5,

MBean Results отправляет уведомления, которые были получены с помощью JConsole. Таким образом, было зарегистрировано и

протестировано получение уведомлений от MBean, что позволяет

оперативно реагировать на события, что является важной частью

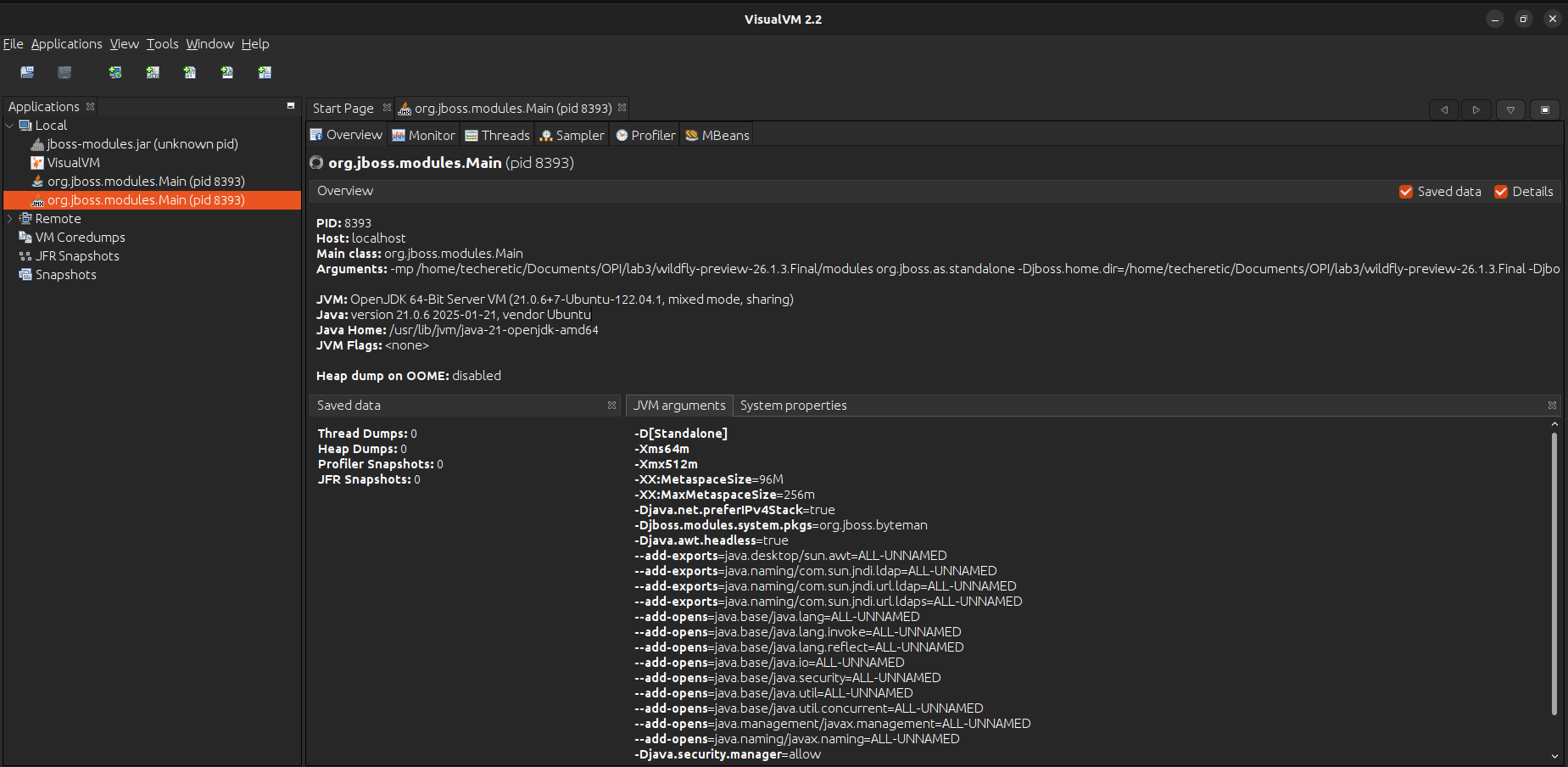
мониторинга.

## 3. VisualVM

* **Снять график изменения показаний MBean-классов**:

|  |
| --- |
|  |
|  |

* **JVM Overview**:



* **Мониторинг**:

|  |
| --- |
|  |

* Определить **имя класса, объекты которого занимают наибольший объём** памяти JVM; **определить пользовательский класс**, в экземплярах которого находятся эти объекты.

|  |
| --- |
|  |
|  |

* **Выводы по результатам мониторинга и профилирования**:

**Изменения показаний** MBean-классов с течением времени:

● Results MBean: Графики AttemptStats показывают общее число установленных пользователем точек, а также число точек, не попадающих в область, и также меняются со временем, что отражает взаимодействие пользователя с системой (создание точек на графике).

● Area MBean: График Area показывает динамику изменений области в зависимости от выбора радиус

Определение **имени класса**, объекты которого занимают **наибольший объём** памяти JVM; определение пользовательского класса, в экземплярах которого находятся эти объекты.

● На основе профилирования памяти видно, что больше всего памяти занимают значения коллекции TreeMap, на втором месте итераторы – всего объекты класса TreeMap занимают ~59%.

● Больше всего памяти из пользовательских классов занимают объекты Attempt – 79.5% от пользовательских классов. Однако в масштабе всего приложения эти объекты занимают примерно 0.2%.

## 4. Исследование программы на утечки памяти

**Искусственная задержка программы**

В процессе анализа производительности приложения была обнаружена проблема, связанная с избыточным вызовом метода java.lang.Thread.sleep(200). Данный метод вызывает приостановку выполнения потока на 200 миллисекунд, что приводит к ненужным задержкам и снижению общей производительности системы.

|  |
| --- |
|  |

Метод не несёт никакой функциональной нагрузки и является избыточным – поэтому его удаление не повлияет на корректность работы приложения, но повысит производительность за счет устранения задержек.

|  |
| --- |
|  |

**Утечка памяти**

Установим максимальный размер кучи в 25Мб с помощью -Xmx25m и запустим программу и посмотрим, как она себя ведет, в VisualVM.

1. При мониторинге видна очевидная переработка GC:

|  |
| --- |
|  |

2. Программа падает через ~3-4 минуты, независимо от выделенной памяти для кучи:

|  |
| --- |
|  |

3. С помощью Heap Dump найдем объекты, занимающие большую часть памяти.

|  |
| --- |
|  |
|  |
| Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере  Контент, сгенерированный ИИ, может содержать ошибки. |

Из графиков использования памяти видно, что

* При работе приложения на каждый запрос создаются экземпляры строк и массивы byte[], затем используются по всему приложению и остаются использованными, но не очищенными в памяти. Хоть GC и отрабатывает корректно, он тратит слишком много ресурсов на очищение памяти, поэтому чтобы убрать излишнюю нагрузку надо устранить утечку;
* Размер кучи постоянно увеличивается, что свидетельствует о проблемах с использованием памяти в программе;
* Через некоторое количество времени получаем ошибку OutOfMemoryError.

Немного исследовав кучу, находим объект, занимающий много памяти (свыше 4мб), при этом содержащий повторяющиеся строки:

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Объекты \_errorMessages хранятся в ArrayList

Найдем строчку с добавлением объектов в этот список:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

В результате получается накопление \_errorMessages в списке, за счет чего и получается переполнение памяти. В программе есть функция для очистки этого массива, однако можно заметить, что на самом деле она не используется нигде

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, рукописный текст, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Решением будет очистка списка \_errorMessage после выполнения очередного запроса.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Запустим программу. Теперь изменения памяти во времени стали более стабильными, и ее значение не стремится к максимальному, так что можно сделать вывод, что GC работает более оптимально.

Изображение выглядит как текст, снимок экрана, число, линия

Контент, сгенерированный ИИ, может содержать ошибки.Программа работает стабильно и не падает с OutOfMemoryException.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

# Вывод

Во время выполнения лабораторной работы я познакомился с практикой написания MBeans в веб-приложениях, были изучены утилиты для мониторинга и профилирования работы программы JConsole и VisualVM, а также был получен опыт по полученным данным определять утечки памяти и устранять их.