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 №4**

по дисциплине

«ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ»

Вариант № 321801

***Выполнил:***

Студент группы P3218

Рамеев Тимур

Ильгизович

***Преподаватель:***

Егошин Алексей

Васильевич

# Задание

1. Для своей программы из лабораторной работы №3 по дисциплине "Веб-программирование" реализовать:
   1. MBean, считающий общее число установленных пользователем точек, а также число точек, не попадающих в область. В случае, если пользователь совершил 3 "промаха" подряд, разработанный MBean должен отправлять оповещение об этом событии.
   2. MBean, определяющий процентное отношение "попаданий" к общему числу кликов пользователя по координатной плоскости.
2. С помощью утилиты JConsole провести мониторинг программы:
   1. Снять показания MBean-классов, разработанных в ходе выполнения задания 1.
   2. Определить имя и версию ОС, под управлением которой работает JVM.
3. С помощью утилиты VisualVM провести мониторинг и профилирование программы:
   1. Снять график изменения показаний MBean-классов, разработанных в ходе выполнения задания 1, с течением времени.
   2. 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. По результатам локализации и устранения проблемы необходимо составить отчёт, в котором должна содержаться следующая информация:
   1. Описание выявленной проблемы.
   2. Описание путей устранения выявленной проблемы.
   3. Подробное (со скриншотами) описание алгоритма действий, который позволил выявить и локализовать проблему.

# Выполнение

## MBeans

Код созданных Бинов:

Листинг 1: Интерфейс Первого бина:

package mbeans;  
  
import lab3.Attempt;  
  
import java.util.Map;  
  
public interface FirstMXBean {  
 Map<Integer, Integer> getSessionsPoints();  
  
 Map<Integer, Integer> getSessionsPointsBad();  
  
 void checkMissesInRow(Attempt attempt, int sessionId);  
}

Листинг 2: 1-й бин PointsChecker.java:

package mbeans;  
  
import entity.AttemptEntity;  
import jakarta.enterprise.context.ApplicationScoped;  
import jakarta.inject.Named;  
import jakarta.persistence.\*;  
import lab3.Attempt;  
  
import javax.management.\*;  
import java.lang.management.ManagementFactory;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Objects;  
  
@Named("pointsChecker")  
@ApplicationScoped  
public class PointsChecker extends NotificationBroadcasterSupport implements FirstMXBean {  
  
 private final EntityManagerFactory entityManagerFactory = Persistence.*createEntityManagerFactory*("default");  
 private final EntityManager entityManager = entityManagerFactory.createEntityManager();  
 private final EntityTransaction transaction = entityManager.getTransaction();  
  
 private final Map<Integer, Integer> sessionsPoints = new HashMap<>();  
 private final Map<Integer, Integer> sessionsPointsBad = new HashMap<>();  
 private final Map<Integer, Integer> sessionsRowOfMisses = new HashMap<>();  
  
 @Override  
 public Map<Integer, Integer> getSessionsPoints() {  
 setSessionPoints();  
 return sessionsPoints;  
 }  
  
 @Override  
 public Map<Integer, Integer> getSessionsPointsBad() {  
 setSessionPointsBad();  
 return sessionsPointsBad;  
 }  
  
 public PointsChecker() throws NotCompliantMBeanException, InstanceAlreadyExistsException, MBeanRegistrationException, MalformedObjectNameException {  
 ObjectName name = new ObjectName("org.example.lab3:name=PointsChecker");  
 MBeanServer server = ManagementFactory.*getPlatformMBeanServer*();  
 try{  
 server.unregisterMBean(name);  
 } catch (InstanceNotFoundException ignored){}  
 server.registerMBean(this, name);  
 System.*out*.println("Point Checker has been created");  
 }  
  
 private void setSessionPoints () {  
 sessionsPoints.clear();  
 TypedQuery<AttemptEntity> listOfSession = entityManager.createNamedQuery("getAttempts", AttemptEntity.class);  
 for (AttemptEntity attempt : listOfSession.getResultList()) {  
 int currentNumber = sessionsPoints.get(attempt.getSessionId()) == null ? 0 : sessionsPoints.get(attempt.getSessionId());  
 sessionsPoints.put(attempt.getSessionId(), currentNumber + 1);  
 }  
 }  
  
 private void setSessionPointsBad () {  
 sessionsPointsBad.clear();  
 TypedQuery<AttemptEntity> listOfSession = entityManager.createNamedQuery("getAttempts", AttemptEntity.class);  
 for (AttemptEntity attempt : listOfSession.getResultList()) {  
 int currentNumberBad = sessionsPointsBad.get(attempt.getSessionId()) == null ? 0 : sessionsPointsBad.get(attempt.getSessionId());  
 if (Objects.*equals*(attempt.getIshit(), "No")){  
 sessionsPointsBad.put(attempt.getSessionId(), currentNumberBad + 1);  
 }  
 }  
 }  
  
 @Override  
 public void checkMissesInRow (Attempt attempt, int sessionId) {  
 final int NOTIFY\_LIMIT = 3;  
 if (!sessionsRowOfMisses.containsKey(sessionId)){  
 sessionsRowOfMisses.put(sessionId, 0);  
 }  
 if (attempt.getIsHit().equals("No")){  
 sessionsRowOfMisses.put(sessionId, sessionsRowOfMisses.get(sessionId) + 1);  
 } else {  
 sessionsRowOfMisses.put(sessionId, 0);  
 }  
 if (sessionsRowOfMisses.get(sessionId) >= NOTIFY\_LIMIT) {  
 System.*out*.println("Missed in a row");  
 Notification n = new Notification("Bad Accuracy", this, 0,  
 String.*format*("User with Session id%d reached %d miss in a row", sessionId, NOTIFY\_LIMIT));  
 sendNotification(n);  
 sessionsRowOfMisses.put(sessionId, 0);  
 }  
 }  
}

Листинг 3: Интерфейс 2-го бина:

package mbeans;  
  
import java.util.Map;  
  
public interface SecondMXBean {  
 Map<Integer, Double> getPercentOfHits();  
}

Листинг 4: Код 2-го бина PercentCounter.java:

package mbeans;  
  
import jakarta.annotation.PostConstruct;  
import jakarta.enterprise.context.ApplicationScoped;  
import jakarta.inject.Inject;  
import jakarta.inject.Named;  
  
import javax.management.\*;  
import java.lang.management.ManagementFactory;  
import java.util.HashMap;  
import java.util.Map;  
  
@Named("persentCounter")  
@ApplicationScoped  
public class PercentCounter implements SecondMXBean {  
  
 private final Map<Integer, Double> sessionsPercentOfHit = new HashMap<>();  
  
 @Inject  
 private PointsChecker pointsChecker;  
  
 public PercentCounter() throws MalformedObjectNameException, MBeanRegistrationException, NotCompliantMBeanException, InstanceAlreadyExistsException {  
 ObjectName name = new ObjectName("org.example.lab3:name=PercentCounter");  
 MBeanServer server = ManagementFactory.*getPlatformMBeanServer*();  
 try{  
 server.unregisterMBean(name);  
 } catch (InstanceNotFoundException ignored){}  
 server.registerMBean(this, name);  
 System.*out*.println("PersentCounter has been created");  
 }  
  
 public PointsChecker getPointsChecker() {  
 return pointsChecker;  
 }  
  
 public void setPointsChecker(PointsChecker pointsChecker) {  
 this.pointsChecker = pointsChecker;  
 }  
  
 @Override  
 public Map<Integer, Double> getPercentOfHits(){  
 setPercentOfHits();  
 return sessionsPercentOfHit;  
 }  
  
 @PostConstruct  
 private void setPercentOfHits() {  
 for (Map.Entry<Integer, Integer> entry : pointsChecker.getSessionsPoints().entrySet()) {  
 if (pointsChecker.getSessionsPointsBad().containsKey(entry.getKey())) {  
 Double percentNumber = (1 - (double) pointsChecker.getSessionsPointsBad().get(entry.getKey()) / pointsChecker.getSessionsPoints().get(entry.getKey())) \* 100;  
 sessionsPercentOfHit.put(entry.getKey(), percentNumber);  
 }  
 }  
 }  
}

## JConsole

1. PercentOfHits Bean

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

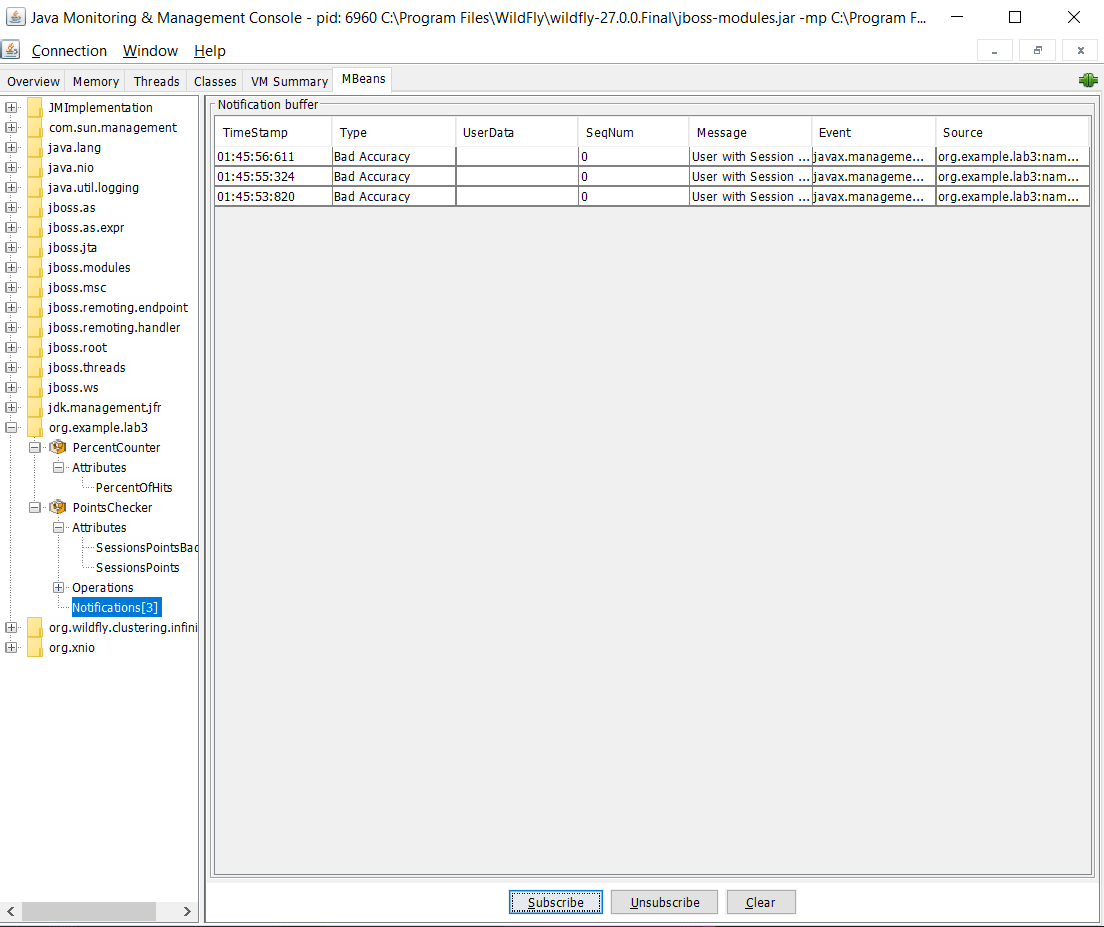
Автоматически созданное описание

1. Изображение выглядит как текст, снимок экрана, программное обеспечение, число

   Автоматически созданное описаниеPointsChecker Bean (Поле SessionsPointsBad)

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

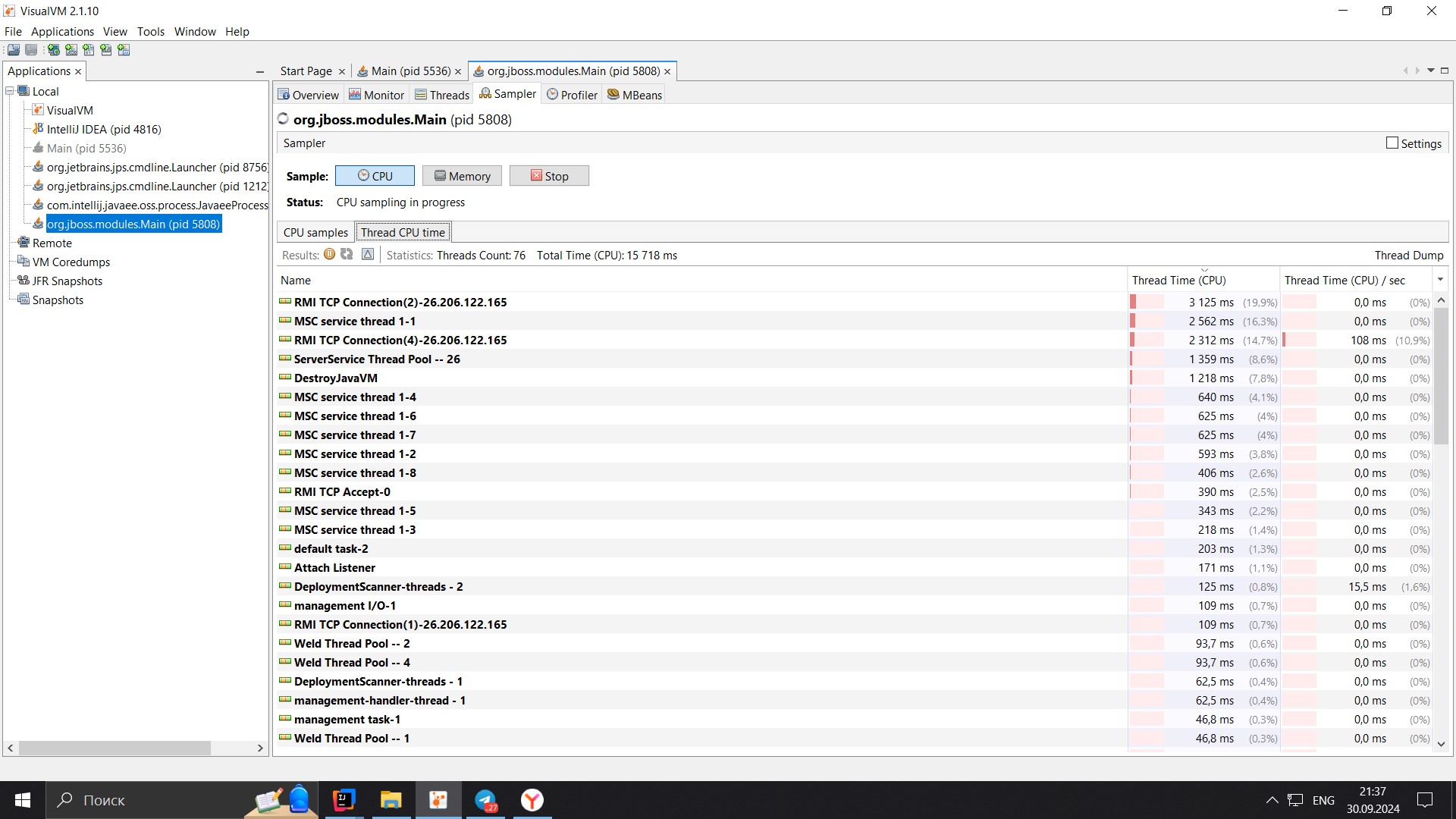
Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание3. Notifications

## VisualVM Мониторинг и профилирование

Отобразить графики не получается, так как графики доступны только для чисел, у нас используются HashMaps.

Произведем Мониторинг потоков и определим поток, потребляющий наибольший процент времени CPU.



## VisualVM для устранения проблем производительности заданного приложения

Рассматриваемая программа представляет из себя бесконечный цикл, отправляющий GET-запросы на имитатор HTTP-сервера. На каждой итерации цикла поток засыпает на некоторое время (для ускорения мониторинга и профилирования задержка была уменьшена)

Для поиска утечки нам необходимо пронаблюдать за работой программы некоторое время, пока по графику загруженности кучи (Heap) не будет видно, что GC неиспользуемые данные, и они постепенно переполняют кучу.

Изображение выглядит как текст, снимок экрана, График, диаграмма

Автоматически созданное описание

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

Автоматически созданное описаниеКогда я увидел, что переполнение неизбежно, я сделал heap dump для определения причины переполнения кучи. Оказалось, что больше всего места занимают массивы типа char[], которые хранятся в классе JavaScript в поле \_errorMessages.

Затем в этом же классе я заметил метод для очистки этого поля, который решил применить на каждой итерации нашего цикла. Таким образом утечку памяти удалось устранить. Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

# Вывод

Мы познакомились с основами профилирования и мониторинга программ на примере утилит JConsole и VisualVM для Java. Нам удалось написать собственные MBean-ы, осуществить профилирование программы, а также обнаружить утечку памяти в предоставленной программе.