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 по

Основам программной инженерии №4

Вариант 32341111

Работу выполнил:

Подольский В.И.

Группа: P3220

Санкт-Петербург,

2024

Оглавление

[Текст задания 3](#_Toc168236365)

[Код реализации MBean 4](#_Toc168236366)

[Показания MBean классов из JConsole 7](#_Toc168236367)

[График изменения показаний MBean-классов с течением времени 10](#_Toc168236368)

[Исследование программы на утечки памяти 12](#_Toc168236369)

[Вывод 15](#_Toc168236370)

## Текст задания

Изображение выглядит как текст, снимок экрана, документ, Шрифт

Автоматически созданное описание

## Код реализации MBean

public interface AreaMBean {  
 void calculateArea(double r);  
 double getArea();  
}

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

public interface PointCountMBean {  
 void updatePointCount();  
 void updateCorrectPoint();  
  
 void MissPlace();  
  
 long getCorrectPoint();  
 long getPointCount();  
 MBeanNotificationInfo[] getNotificationInfo();  
}

import javax.enterprise.context.ApplicationScoped;  
import javax.inject.Named;  
import javax.management.AttributeChangeNotification;  
import javax.management.MBeanNotificationInfo;  
import javax.management.Notification;  
import javax.management.NotificationBroadcasterSupport;  
import java.io.Serializable;  
  
@Named  
@ApplicationScoped  
public class PointCount extends NotificationBroadcasterSupport implements Serializable, PointCountMBean {  
  
 private long pointCount = 0;  
 private long correctPoint = 0;  
 private int sequenceNumber = 0;  
  
 public PointCount(){  
  
 }  
  
  
 @Override  
 public void updatePointCount() {  
 pointCount++;  
 }  
  
 @Override  
 public void updateCorrectPoint() {  
 correctPoint++;  
 }  
  
 @Override  
 public void MissPlace() {  
 Notification notification = new Notification  
 ( "Point miss place", getClass().getSimpleName(), sequenceNumber++, "Point miss correct place" );  
 sendNotification(notification);  
 }  
  
 @Override  
 public long getCorrectPoint() {  
 return correctPoint;  
 }  
  
 @Override  
 public long getPointCount() {  
 return pointCount;  
 }  
 @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 };  
 }  
}

public class SimpleAgent {  
 @Inject  
 private PointCount pointCount;  
 @Inject  
 private Area area;  
  
 public SimpleAgent(){  
  
 }  
  
 @PostConstruct  
 public void startAgent(){  
 MBeanServer server = ManagementFactory.*getPlatformMBeanServer*();  
 ObjectName objectName;  
 try {  
 objectName = new ObjectName("SimpleAgent:name=PointCount");  
 if (!server.isRegistered(objectName)) {  
 server.registerMBean(pointCount, objectName);  
 }  
 objectName = new ObjectName("SimpleAgent:name=Area");  
 if (!server.isRegistered(objectName)){  
 server.registerMBean(area, objectName);  
 }  
  
 } catch (MalformedObjectNameException | InstanceAlreadyExistsException |  
 MBeanRegistrationException | NotCompliantMBeanException e) {  
 e.printStackTrace();  
 }  
 }  
 public void logSimpleAgentStarted() {  
 System.*out*.println("SimpleAgent.logSimpleAgentStarted");  
 }  
  
 public void startup(@Observes @Initialized(ApplicationScoped.class) Object context) {  
 SimpleAgent a = new SimpleAgent();  
 a.logSimpleAgentStarted();  
 }  
}

## Показания MBean классов из JConsole

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

Автоматически созданное описание

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

Автоматически созданное описание

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

Автоматически созданное описание

Изображение выглядит как линия, текст, График, снимок экрана

Автоматически созданное описание

Количество классов, загруженных в JVM в процессе выполнения программы - 28070

## График изменения показаний MBean-классов с течением времени

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

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, диаграмма, дисплей

Автоматически созданное описание

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

Автоматически созданное описание

Имя потока, потребляющего наибольший процент времени CPU – RMI TCP Connection (51)

## Исследование программы на утечки памяти

Изображение выглядит как текст, снимок экрана, диаграмма, число

Автоматически созданное описание

Установим максимальный размер кучи в 30Мб с помощью -Xmx30m и запустим программу. Из графика использования памяти видно, что размер кучи постоянно увеличивается, что свидетельствует о проблемах с использованием памяти в программе. Через некоторое количество времени получаем ошибку OutOfMemoryError.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

С помощью HeapDump найдем объекты, занимающие большую часть памяти.

Изображение выглядит как текст, линия, Шрифт, снимок экрана

Автоматически созданное описание

Объекты \_errorMessages хранятся в ArrayList Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Найдем строчку с добавлением объектов в этот список:

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Статическая переменная \_errorMessages бесконечно добавляет errorMessage при каждом запросе

В программе есть функция для очистки \_errorMessage, однако она не используется

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Решением будет очистка списка с \_errorMessage после выполнения очередного запроса.

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

Автоматически созданное описание

Запустим программу. Теперь память не стремится к максимальному значению, как было раньше. Программа работает нормально и не пробрасывает OutOfMemoryException.

Изображение выглядит как текст, снимок экрана, линия, число

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

## Вывод

Во время выполнения лабораторной работы были изучены утилиты для мониторинга работы программы JConsole и VisualVM. А также был получен опыт по полученным данным определять утечки памяти и устранять их.