Федеральное государственное автономное
образовательное учреждение высшего образования
«Научно-образовательная корпорация ИТМО»

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Отчёт по лабораторной работе №4

По дисциплине «Основы Программной Инженерии» (4 семестр)

Студенты:
Дениченко Александр P3212
Беляев Михаил P3212
Практик:
Осипов Святослав Владимирович

Санкт-Петербург
2024 г.

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 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 версию Java Language Specification, реализуемую данной средой исполнения.

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.

2 Исходный код разработанных MBean-классов

Листинг 1: Класс для подсчёта интервалов

```
@Getter
@Setter
@ManagedBean(name = "interval", eager = true)
@SessionScoped
public class ClickIntervalBean implements ClickIntervalBeanMBean, MBeanRegistration {
    private static final Logger logger = Logger.getLogger(ClickIntervalBean.class.getName());

    private List<Long> clickTimestamps = new LinkedList<>();
    private double averageInterval = 0;

    @Override
    public void addClickTimestamp() {
        long currentTime = System.currentTimeMillis();
        if (!clickTimestamps.isEmpty()) {
            long lastTime = clickTimestamps.get(clickTimestamps.size() - 1);
            averageInterval = ((averageInterval * (clickTimestamps.size() - 1)) + (currentTime -
                ↪ lastTime)) / clickTimestamps.size();
        }
        clickTimestamps.add(currentTime);
        logger.info("New click timestamp added: " + currentTime + "; Updated average interval: " +
            ↪  averageInterval);
    }

    @Override
    public void postRegister(Boolean registrationDone) {
        logger.info("ClickIntervalBean registered to the MBean server: " + registrationDone);
    }

    @Override
    public void postDeregister() {
        logger.info("ClickIntervalBean deregistered from the MBean server");
```

```
    }

    @Override
    public void preDeregister() {
        logger.info("ClickIntervalBean is about to be deregistered from the MBean server");
    }

    @Override
    public ObjectName preRegister(MBeanServer server, ObjectName name) throws Exception {
        // Optionally modify the name
        return name;
    }
}
```

Листинг 2: Класс для подсчёта количества кликов

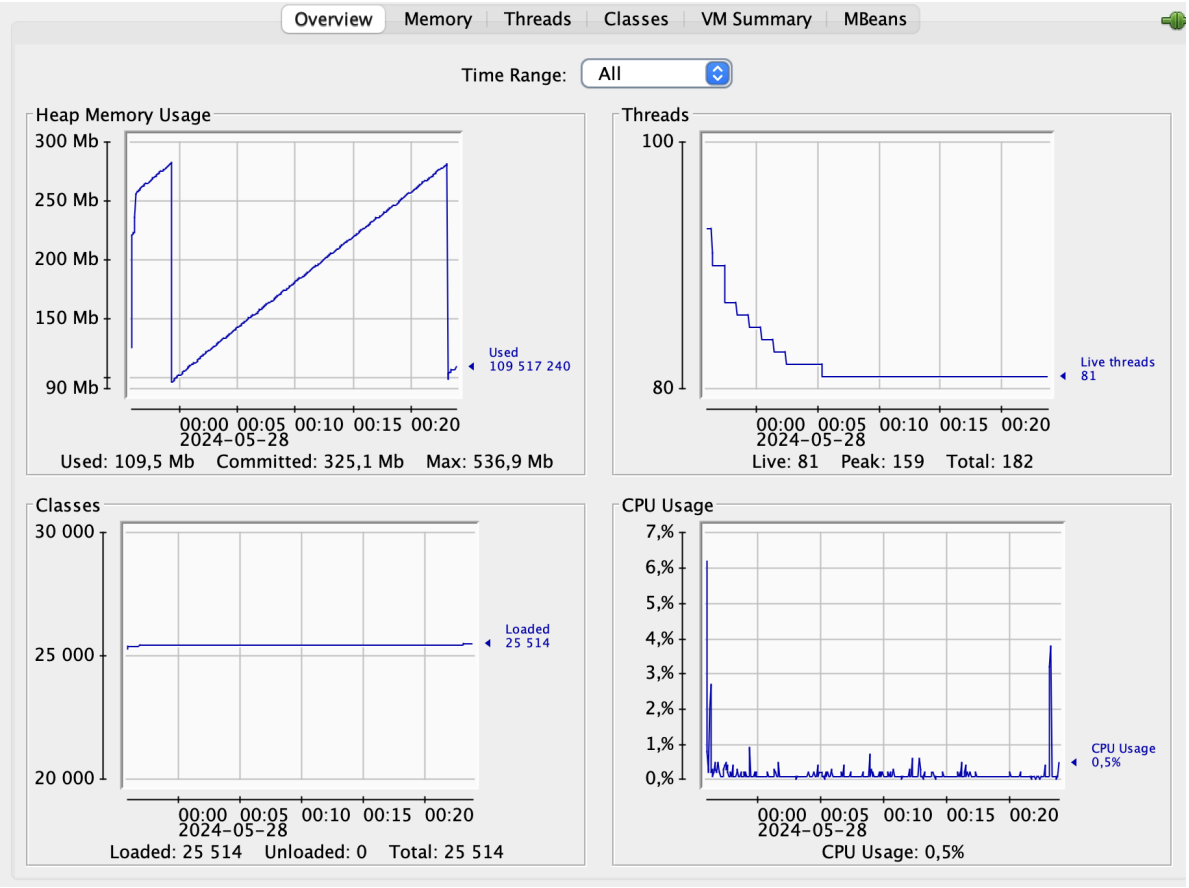
```
@Getter
@Setter
@ManagedBean(name = "counter", eager = true)
public class CounterBean implements CounterBeanMBean {
    private static final Logger logger = Logger.getLogger(CounterBean.class.getName());

    private int countHits;
    private int loseHits;
    private boolean lastHit;

    public CounterBean() {
        countHits = 0;
        loseHits = 0;
        lastHit = true;
    }

    @Override
    public void addHit(boolean res) {
        countHits++;
        if (!res) {
            loseHits++;
            if (!lastHit) {
                logger.info("2");
            }
        }
        lastHit = res;
    }

    @Override
    public int getCountHits() {
        return countHits;
    }

    @Override
    public int getLoseHits() {
        return loseHits;
    }

    @Override
    public boolean isLastHit() {
        return lastHit;
    }
}
```

Листинг 3: Класс выступающий главным бином

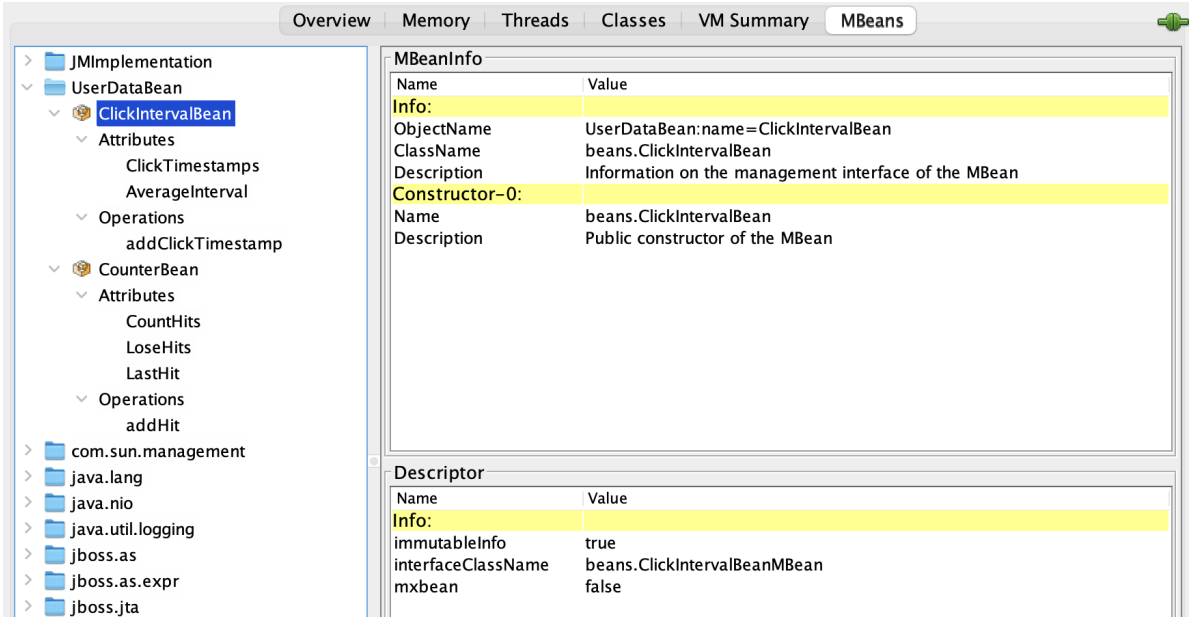
```java
@Getter
@Setter
@Named(value = "user")
@ApplicationScoped
public class UserDataBean {
    private static final Logger logger = Logger.getLogger(UserDataBean.class.getName());

    @Inject
    private CounterBean counterBean;
    @Inject
    private ClickIntervalBean clickIntervalBean;
    private Double x = (double) 0;
    private Double y = (double) 0;
    private Double r = (double) 0;
    private Boolean hit;
    private List<UserData> resultList;

    @PostConstruct
    public void init() {
        MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();

        try {
            ObjectName nameCounter = new ObjectName("UserDataBean:name=CounterBean");
            mbs.registerMBean(counterBean, nameCounter);

            ObjectName nameInterval = new ObjectName("UserDataBean:name=ClickIntervalBean");
            mbs.registerMBean(clickIntervalBean, nameInterval);
        } catch (Exception e) {
            e.printStackTrace();
        }
        logger.info("JConsole for manage");
    }


    private void updateLocal() {
        SessionFactory sessionFactory = HibernateUtil.getSessionFactory();
        UserDataDao userDataDao = new UserDataDao(sessionFactory);
        resultList = userDataDao.getUserData();
    }
    public void clearTable(){
        SessionFactory sessionFactory = HibernateUtil.getSessionFactory();
        UserDataDao userDataDao = new UserDataDao(sessionFactory);
        userDataDao.clearResultTable();
        updateLocal();
    }

    public void saveData() {
        if ((x >= 0 && x <= r / 2) && (y >= 0 && y <= r)){
            hit = true;}
        else if ((x <= 0 && x >= -r) && (y <= 0 && y >=  -r/2)){
            hit = (x>=-r-2*y);}
        else if ((x >= 0 && x <= r) && (y <= 0 && y >= -r)){
            hit = (Math.pow(x, 2) + Math.pow(y, 2) <= Math.pow(r, 2));}
        else{
            hit = false;}
        if(r==0){
            hit = false;
        }

        if (x == null) {
            x = (double) 0;
```

```
62          } else if (!(x instanceof Number)) {
63              x = (double) 0;
64          }
65          if (y == null) {
66              y = (double) 0;
67          } else if (!(y instanceof Number)) {
68              y = (double) 0;
69          }
70          if (r == null) {
71              r = (double) 0;
72          } else if (!(r instanceof Number)) {
73              r = (double) 0;
74          }
75
76          counterBean.addHit(hit);
77          clickIntervalBean.addClickTimestamp();
78          logger.info(String.valueOf(clickIntervalBean.getAverageInterval()));
79
80          SessionFactory sessionFactory = HibernateUtil.getSessionFactory();
81          UserDataDao userDataDao = new UserDataDao(sessionFactory);
82          userDataDao.saveUserData(x, y, r, hit);
83          updateLocal();
84      }
85  }
```

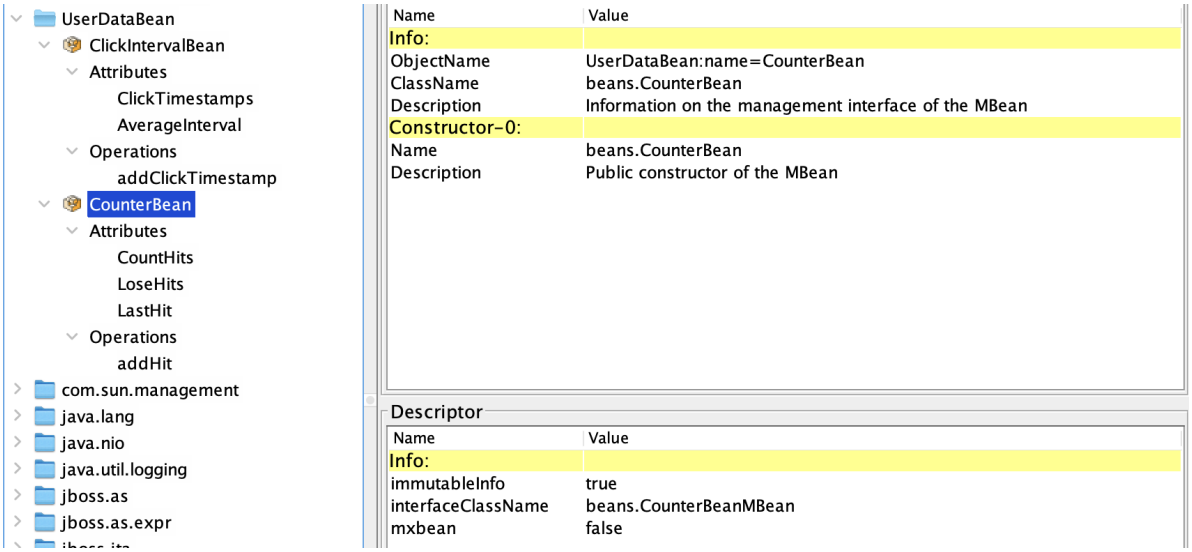
3 Снятые показания в ходе мониторинга



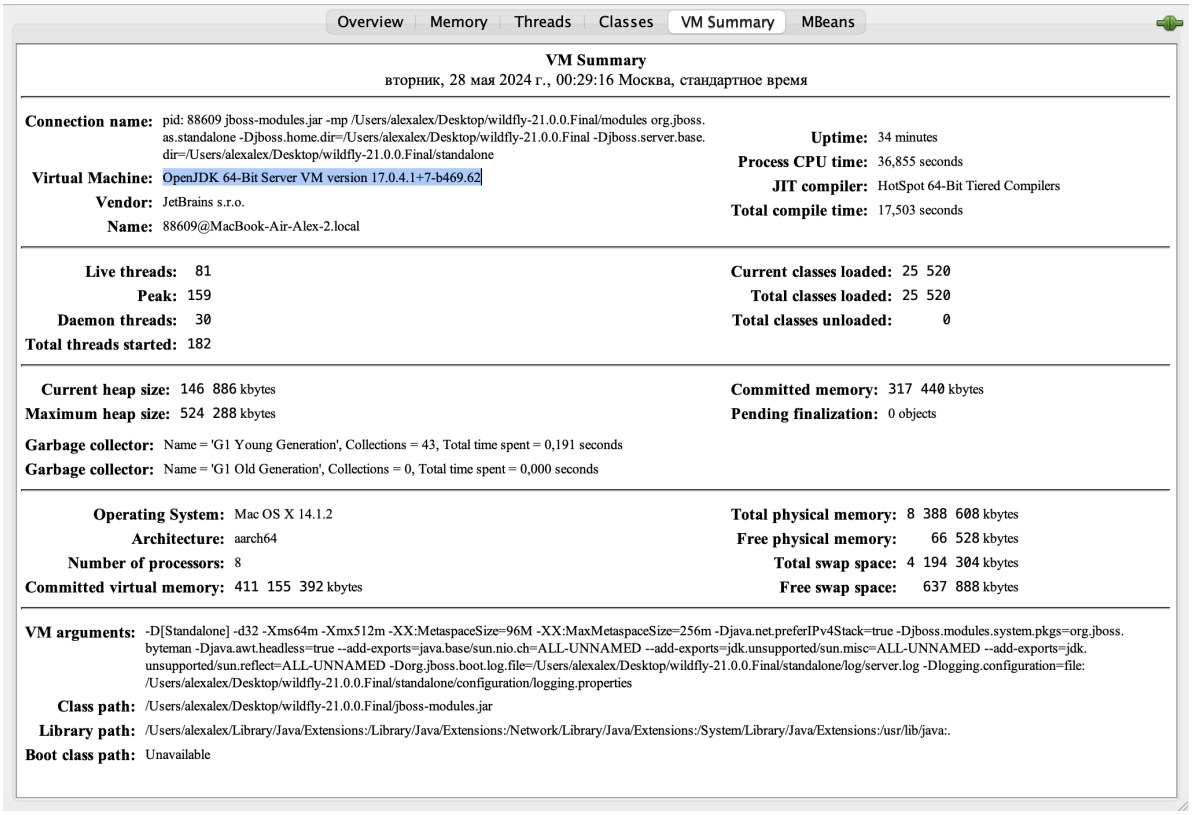
Скриншот 1. Общий вид



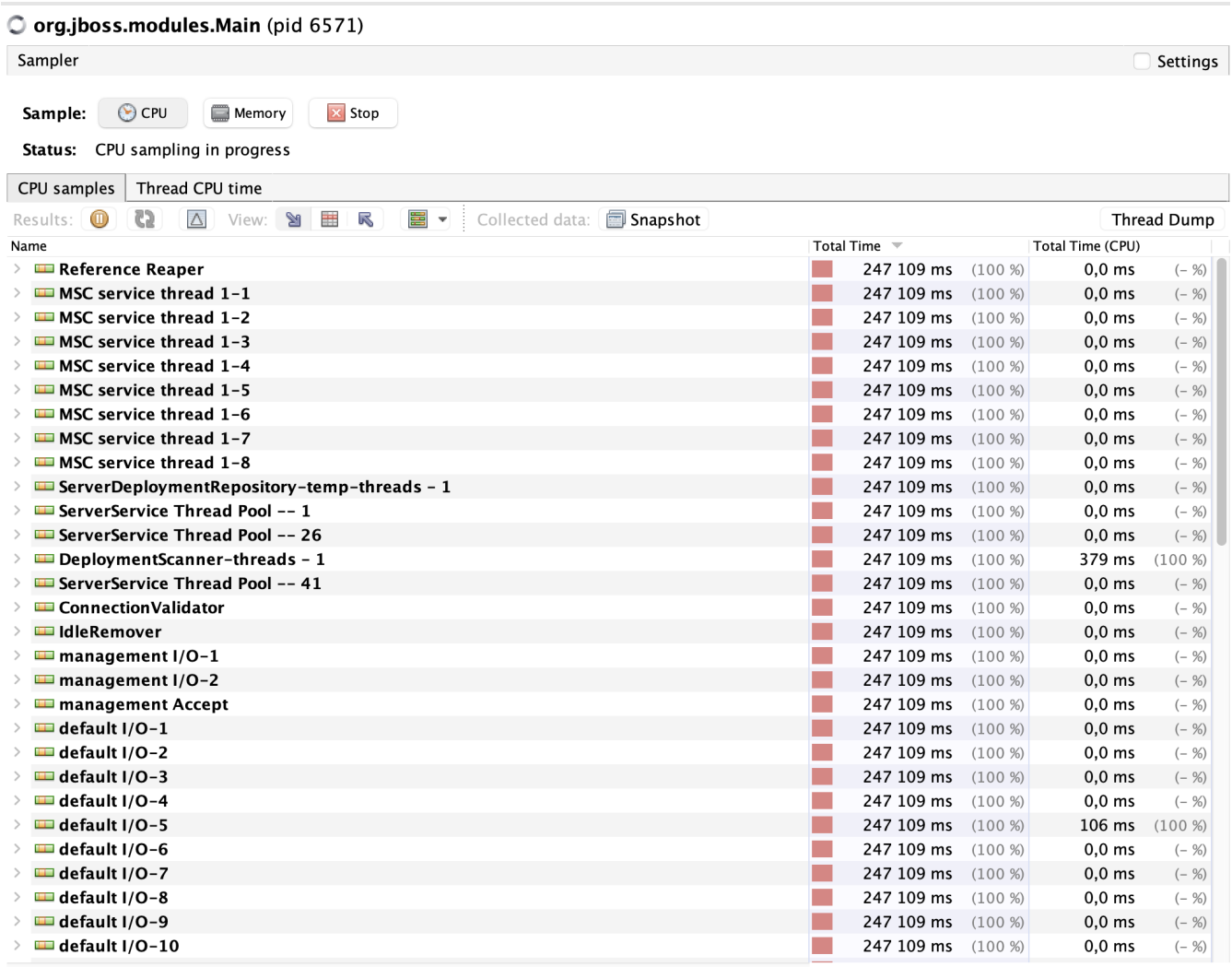
Скриншот 2. Бин интервал



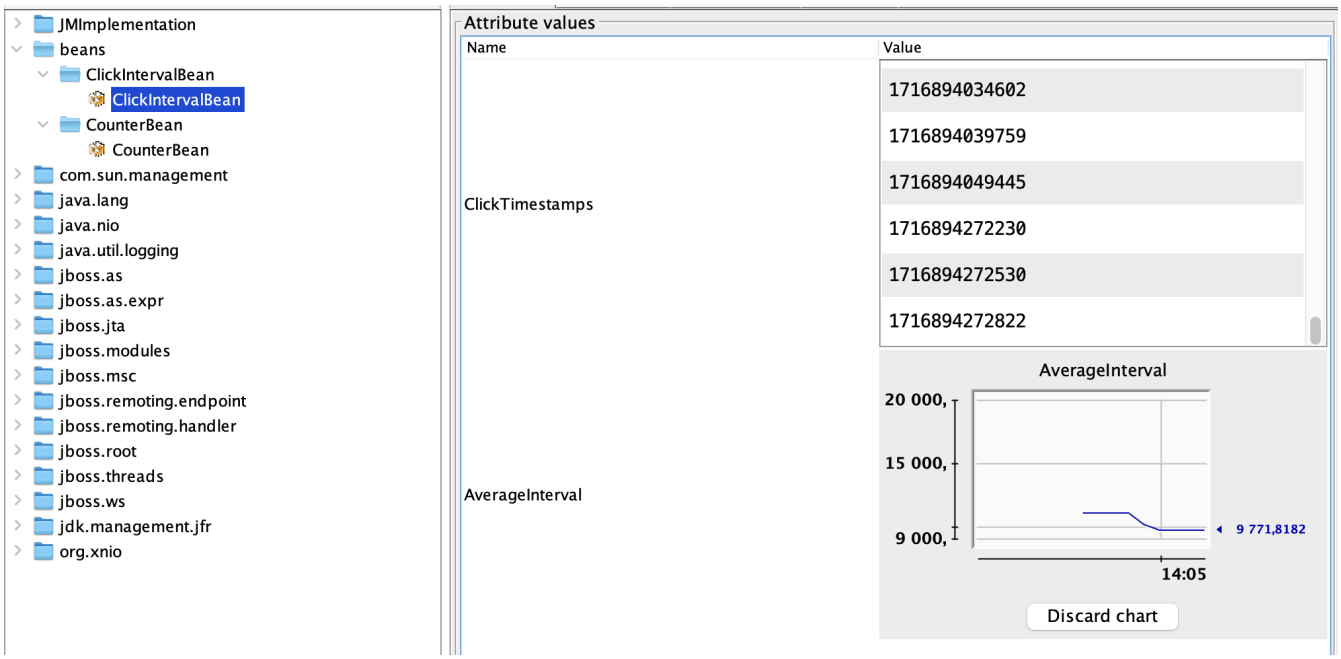
Скриншот 3. Клик бин



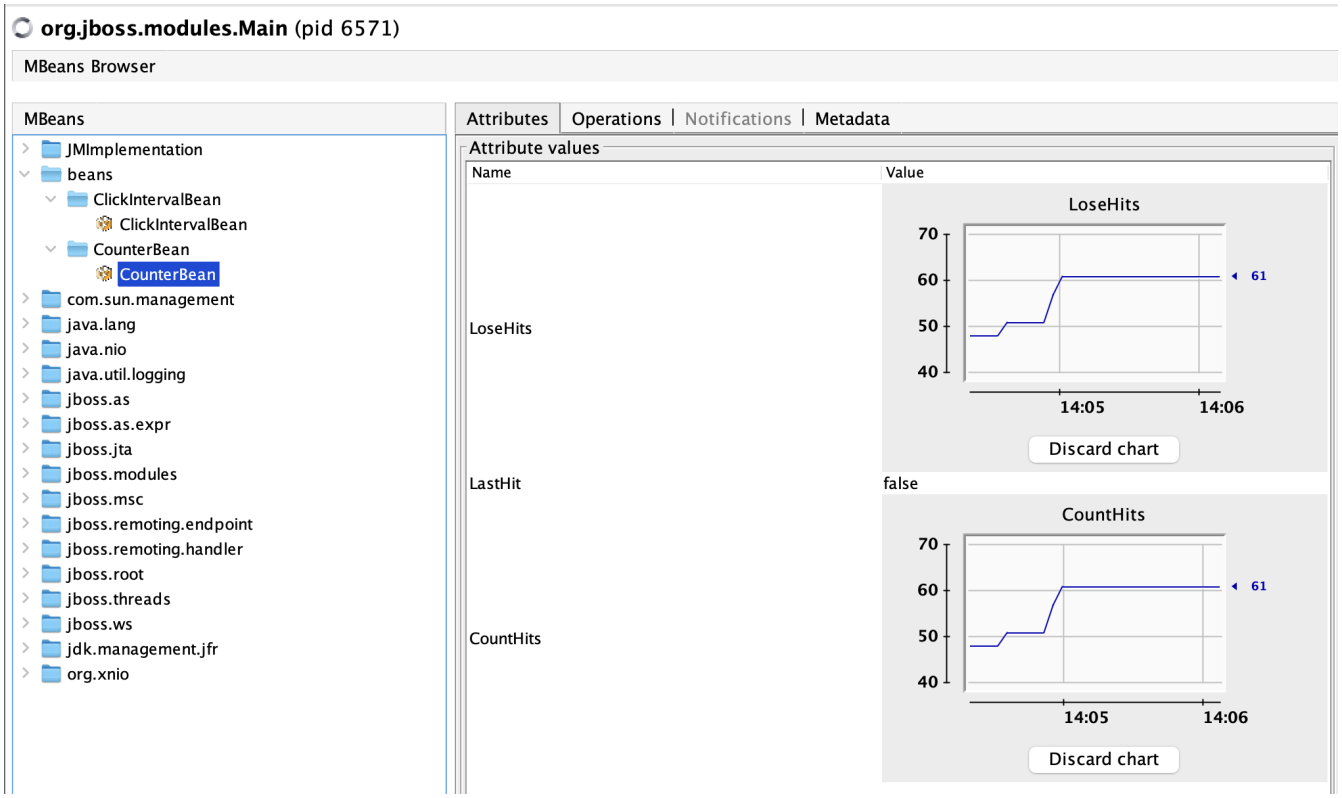
Скриншот 4. Версия



Скриншот 5. Показания VisualVM



Скриншот 6. Показания VisualVM для интервального бина



Скриншот 7. Показания VisualVM для количественного бина

4 Проверка кода на утечки памяти

Нашли цикл, где поменяли время сна на 0, а так же выставили максмаильный размер кучи 50MB.

```
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    try {
        HttpUnitOptions.setExceptionsThrownOnScriptError(false);
        ServletRunner sr = new ServletRunner();
        sr.registerServlet( resourceName: "myServlet", HelloWorld.class.getName());
        ServletUnitClient sc = sr.newClient();
        int number = 1;
        WebRequest request = new GetMethodWebRequest( urlString: "http://test.meterware.com/myServlet");
        while (true) {
            WebResponse response = sc.getResponse(request);
            System.out.println("Count: " + number++ + response);
            java.lang.Thread.sleep( millis: 0);

        }
```

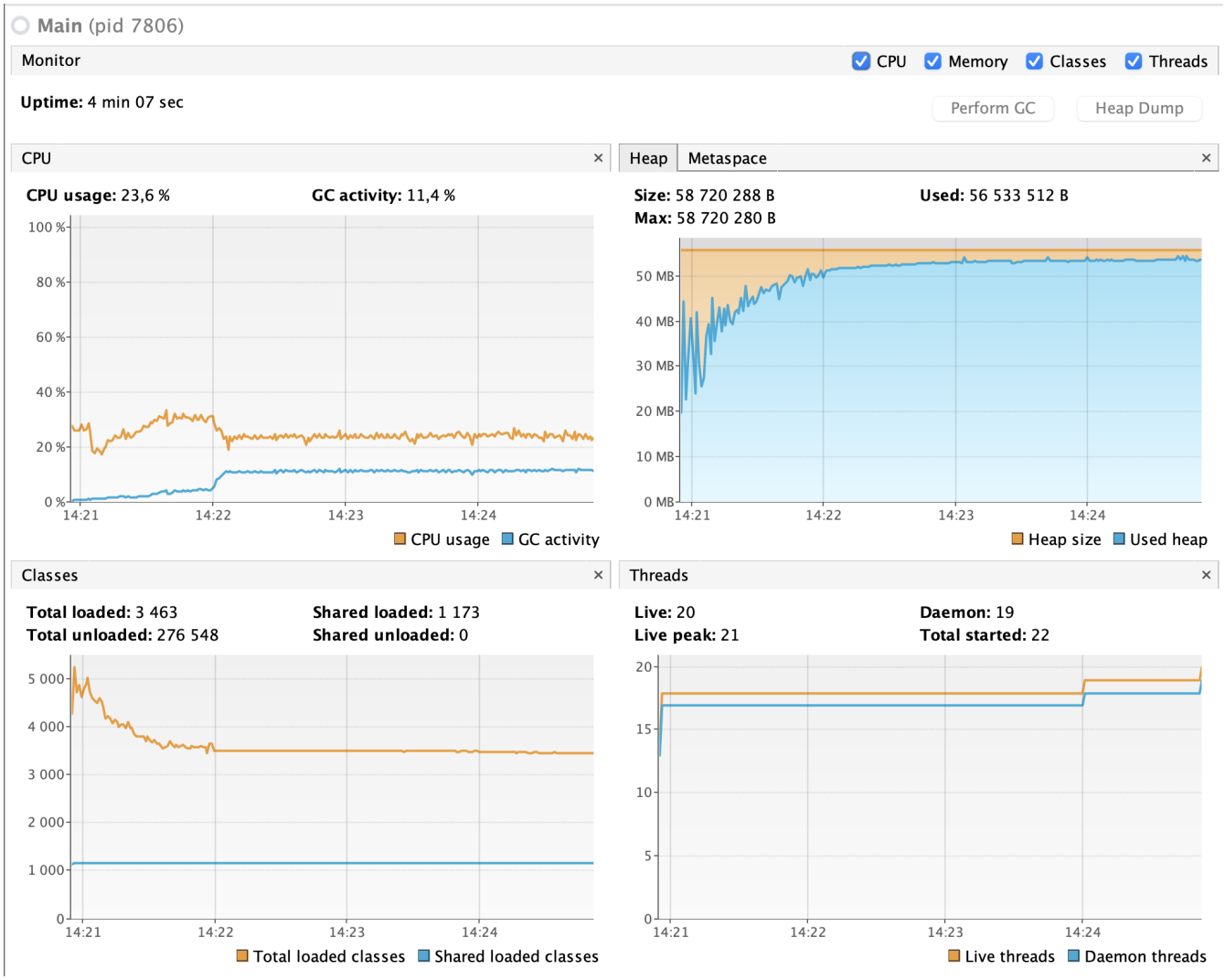
Скриншот 8. Проблемное место

Подождал и вылетела ошибка

```
Count: 262975[ _response = com.meterware.servletunit.ServletUnitHttpResponse@17c0decb]
Count: 262976[ _response = com.meterware.servletunit.ServletUnitHttpResponse@36ef14b0]
Count: 262977[ _response = com.meterware.servletunit.ServletUnitHttpResponse@eb8ed73]
Count: 262978[ _response = com.meterware.servletunit.ServletUnitHttpResponse@2feddaf0]
Count: 262979[ _response = com.meterware.servletunit.ServletUnitHttpResponse@4af8dac0]
Exception in thread "main" java.lang.OutOfMemoryError Create breakpoint : Java heap space
```

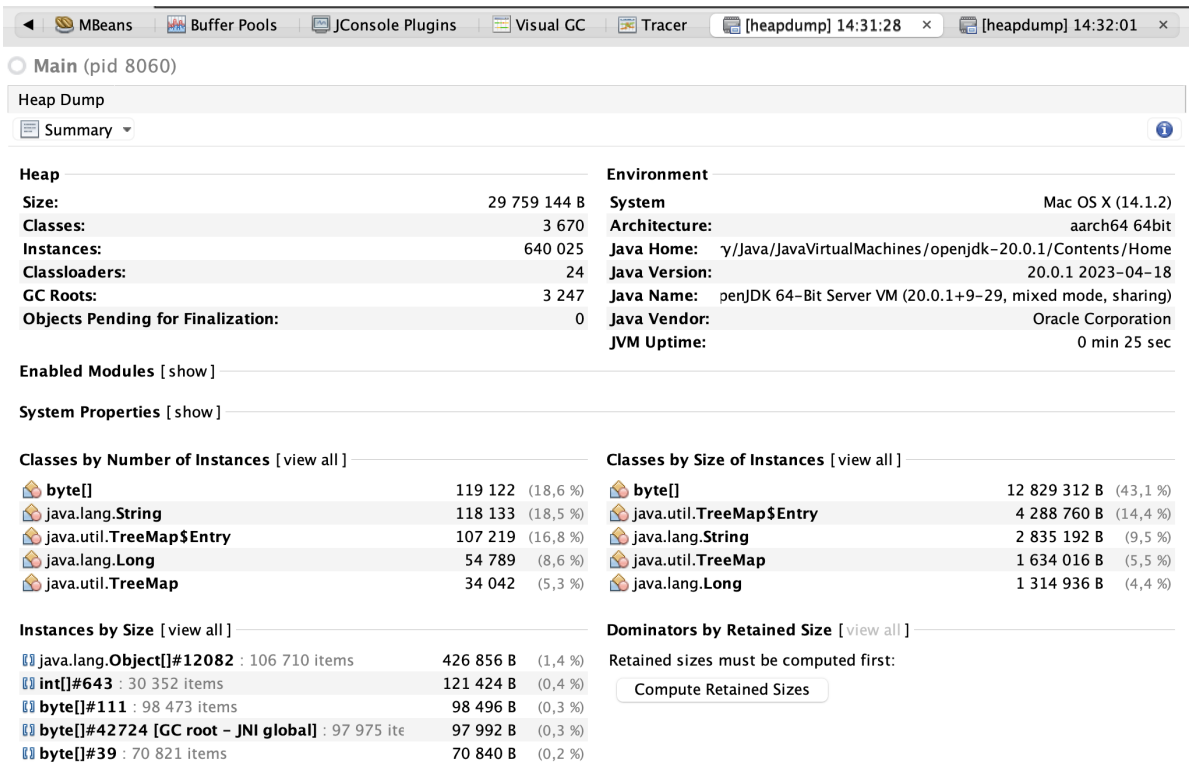
Скриншот 9. Ошибка кучи

При профилировании уже сразу была видна проблема

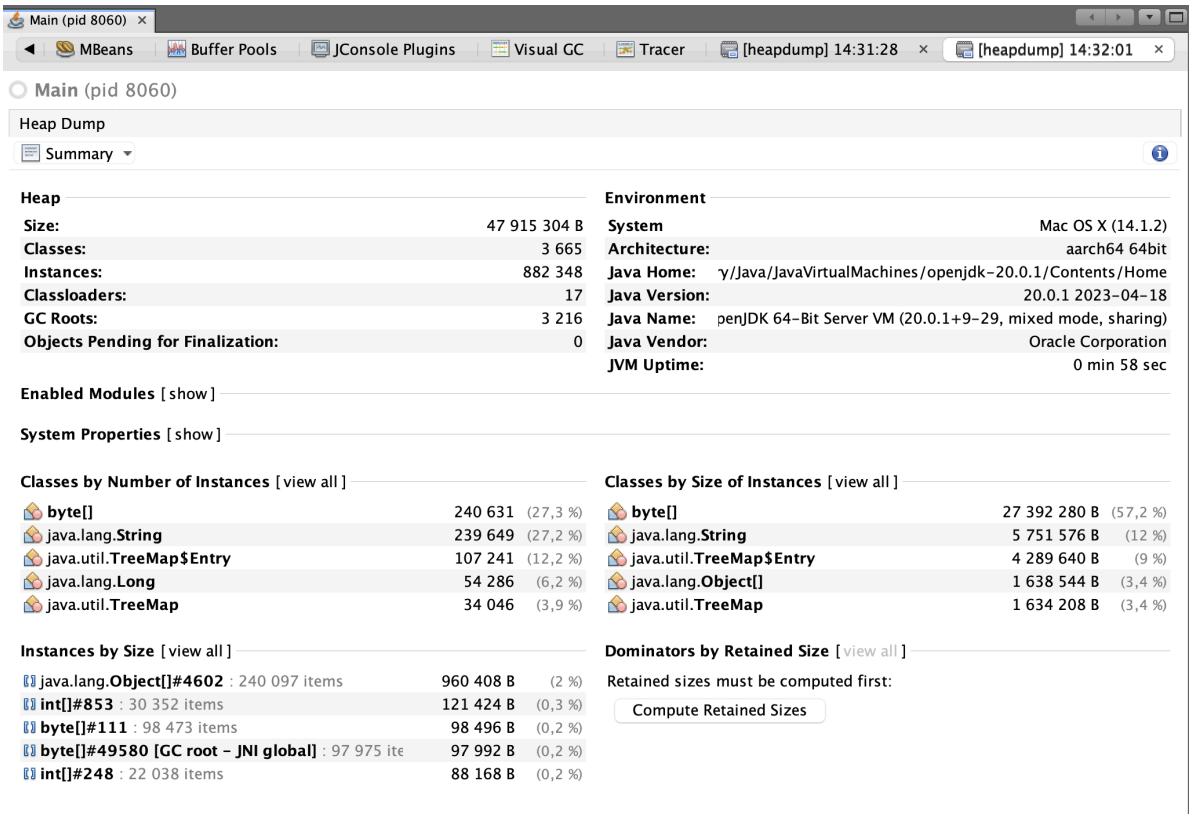


Скриншот 10. Ошибка кучи

Сделал 2 раза HeapDump с небольшим промежутком по времени, чтобы увидеть где подтикает память:

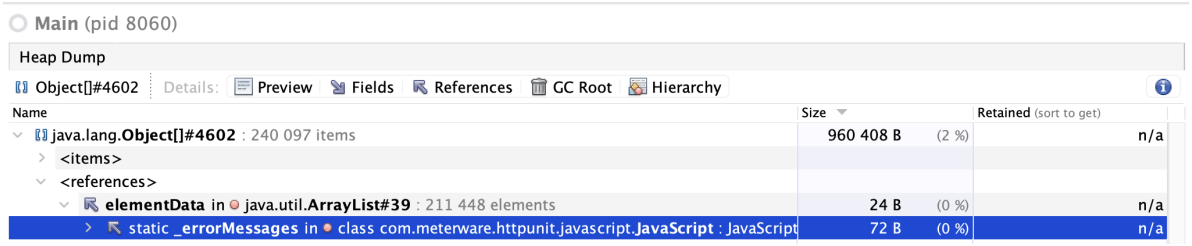


Скриншот 11. Heap Dump



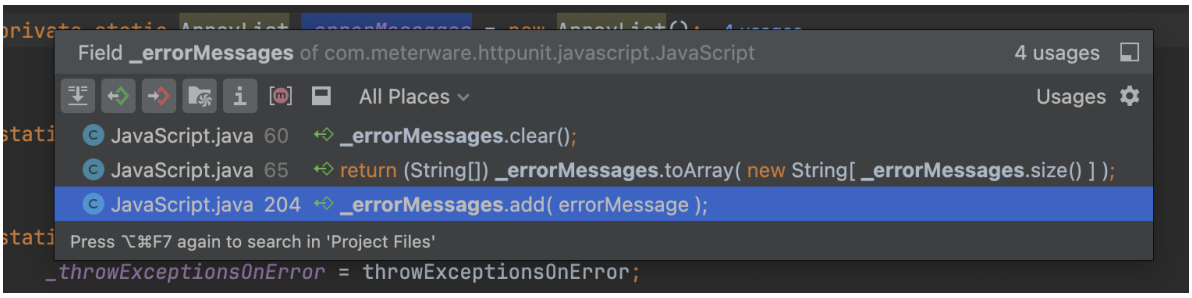
Скриншот 12. Heap Dump

Нашёл класс из-за которого все проблемы:



Скриншот 13. err msg

В данный список добавление только в одном месте



Скриншот 14. add

Есть смысл очищать данные при помощи следующего метода:

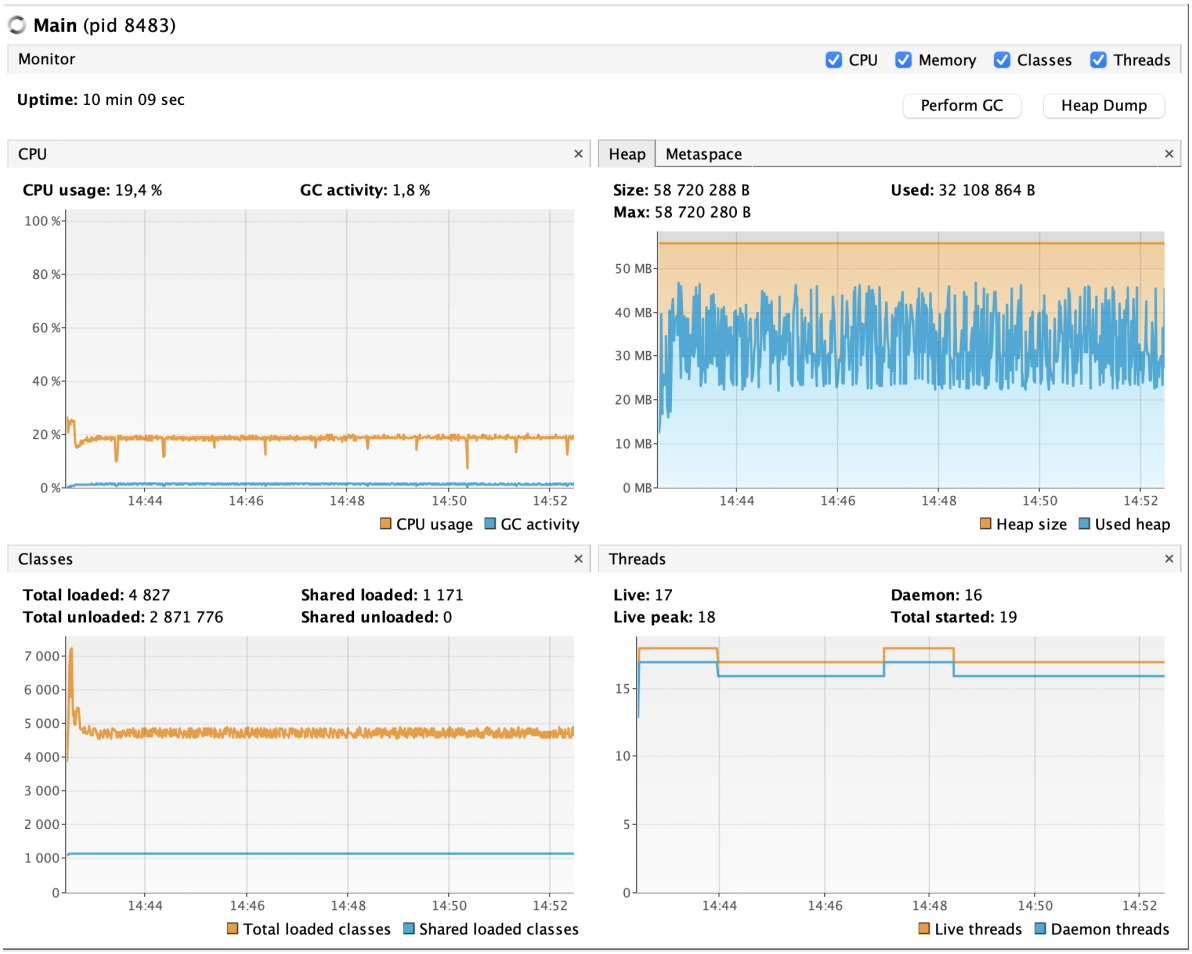
```
static void clearErrorMessages() { _errorMessages.clear(); }
```

Скриншот 15. clear

```
public static void main(String[] args) {
    try {
        HttpUnitOptions.setExceptionsThrownOnScriptError(false);
        ServletRunner sr = new ServletRunner();
        sr.registerServlet( resourceName: "myServlet", HelloWorld.class.getName());
        ServletUnitClient sc = sr.newClient();
        int number = 1;
        WebRequest request = new GetMethodWebRequest( urlString: "http://test.meterware.com/myServlet");
        while (true) {
            WebResponse response = sc.getResponse(request);
            System.out.println("Count: " + number++ + response);
            java.lang.Thread.sleep( millis: 0);
            HttpUnitOptions.clearScriptErrorMessages();
        }
```

Скриншот 16. clear

После локализации вышел следующий результат



Скриншот 17. res

5 Вывод

В ходе данной лабораторной работы я ознакомился с утилитами JConsole и VisualVM для мониторинга и профилирования Java приложений. Также я научился локализовывать и устранять проблемы, связанные с производительностью приложения.