



УНИВЕРСИТЕТ ИТМО

Факультет программной
инженерии и компьютерной техники

Лабораторная работа № 4
по дисциплине
«Методы и средства программной инженерии»

Вариант 666333

Преподаватель: Покид Александр Владимирович

Выполнили: Геллер Л. А., Блинова М. А.

Группа: Р3230

Санкт-Петербург
2021 г.

Задание

1. Для своей программы из [лабораторной работы #4](#) по дисциплине "Программирование интернет-приложений" реализовать:

- MBean, считающий общее число установленных пользователем точек, а также число точек, попадающих в область. В случае, если координаты установленной пользователем точки вышли за пределы отображаемой области координатной плоскости, разработанный MBean должен отправлять оповещение об этом событии.
- MBean, определяющий процентное отношение "попаданий" к общему числу кликов пользователя по координатной плоскости.

2. С помощью утилиты JConsole провести мониторинг программы:

- Снять показания MBean-классов, разработанных в ходе выполнения задания 1.
- Определить имена всех потоков, выполняющихся при запуске программы.

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 объем памяти JVM; определить пользовательский класс, в экземплярах которого находятся эти объекты.

4. Получить HeapDump, и с помощью утилиты VisualVM локализовать и устранить "утечку памяти" в программе ниже

1. MBean

- PointsCount

```
@Component
public class PointCount extends NotificationBroadcasterSupport implements PointCountMBean {

    private long sequenceNumber = 1;
    private int shots = 0;
    private int hits = 0;

    @Override
    public int getTotal() { return shots; }

    @Override
    public int getHits() { return hits; }
    public void setTotal(int n) { shots = n; }
    public void setHits(int n) { hits = n; }

    public void takeAShot(Data point) {
        shots++;
        if (point.getResult()) hits++;
        if (Math.abs(point.getX()) > point.getR()*1.15 || Math.abs(point.getY()) > point.getR()*1.15) {
            System.out.println("Out of displayed area!");
            Notification n = new Notification( type: "OutOfDisplayedArea", this.getClass().getName(),
                sequenceNumber,
                message: "Point coordinates (\\" + point.getX() + \";\\" + point.getY() + \") " +
                    "are out of displayed area of coordinate plane");
            sendNotification(n);
            sequenceNumber++;
        }
    }
}

@Override
public MBeanNotificationInfo[] getNotificationInfo() {
    String[] types = new String[]{AttributeChangeNotification.ATTRIBUTE_CHANGE};
    String name = AttributeChangeNotification.class.getName();
    String description = "Point coordinates are out of displayed area of coordinate plane";
    MBeanNotificationInfo info = new MBeanNotificationInfo(types, name, description);
    return new MBeanNotificationInfo[]{info};
}
```

- HitPercent

```
@Component
public class HitPercent extends NotificationBroadcasterSupport implements HitPercentMBean {

    private double percent = 0;

    @Override
    public double getPercent() { return percent; }

    public void setPercent(int total, int hits) { percent = (double)hits/total; }

    public void updatePercent(Data point, int total) {
        if (point.getResult()) percent = (percent * total + 1) / (total + 1);
        else percent *= (double) total / (total + 1);
    }
}
```

2. JConsole

Показания MBean-классов:

PointsCount:

Attribute values	
Name	Value
Hits	8
Total	12

HitPercent:

Attribute values	
Name	Value
Percent	0.6666666666666666

Notifications:

Notification buffer						
TimeStamp	Type	UserData	SeqNum	Message	Event	Source
11:36:57:189	OutOfDisplayedA...		2	Point coordinates...	javax.manageme...	web.lab4.mbeans.Poi...
11:36:49:181	OutOfDisplayedA...		1	Point coordinates...	javax.manageme...	web.lab4.mbeans.Poi...
				Point coordinates (1.5;0.0) are out of displayed area of		

Количество классов, загруженных в JVM в процессе выполнения программы:

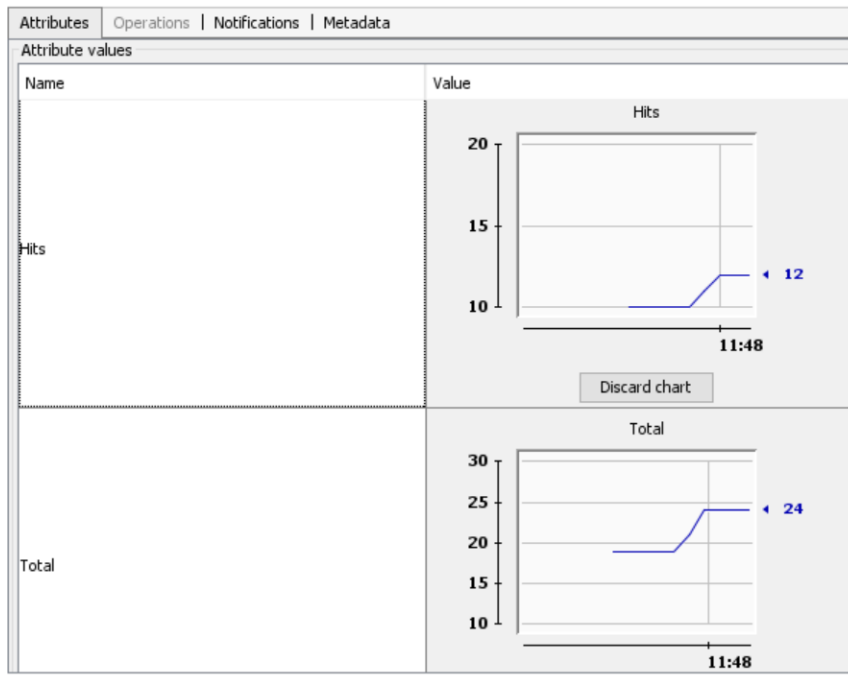
Current classes loaded: 64 064

Total classes loaded: 64 106

Total classes unloaded: 42

3. VisualVM

График изменения показаний MBean-классов:



Поток, потребляющий наибольший процент времени CPU:

The screenshot shows the 'Thread CPU time' window in VisualVM. It displays a list of threads and their CPU usage. The thread 'MSC service thread 1-5' is highlighted as the one consuming the most CPU time.

Name	Thread Time (CPU)	Thread Time (CPU) / sec
MSC service thread 1-5	5 203 ms (12.1%)	0,0 ms (0%)
MSC service thread 1-8	4 843 ms (11.3%)	0,0 ms (0%)
MSC service thread 1-7	4 531 ms (10.6%)	0,0 ms (0%)
MSC service thread 1-1	4 437 ms (10.4%)	0,0 ms (0%)
MSC service thread 1-6	4 312 ms (10.1%)	0,0 ms (0%)
MSC service thread 1-2	3 703 ms (8.6%)	0,0 ms (0%)
MSC service thread 1-4	3 250 ms (7.6%)	0,0 ms (0%)
RMI TCP Connection(idle)	2 953 ms (6.9%)	0,0 ms (0%)
MSC service thread 1-3	2 890 ms (6.7%)	0,0 ms (0%)
ServerService Thread Pool -- 27	1 578 ms (3.7%)	0,0 ms (0%)
DestroyJavaVM	906 ms (2.1%)	0,0 ms (0%)
RMI TCP Accept-0	718 ms (1.7%)	0,0 ms (0%)
default task-1	593 ms (1.4%)	0,0 ms (0%)
Reference Handler	359 ms (0.8%)	0,0 ms (0%)
Finalizer	343 ms (0.8%)	0,0 ms (0%)
DeploymentScanner-threads - 2	234 ms (0.5%)	0,0 ms (0%)
Weld Thread Pool -- 4	203 ms (0.5%)	0,0 ms (0%)
RMI TCP Connection(6)-172.20.10.2	187 ms (0.4%)	0,0 ms (0%)

Больше всего процессорного времени занимает поток MSC service thread 1-5

Утечка памяти

Проблема: предоставленная программа падает с OutOfMemoryException

HeapDump:

leak-jar (pid 4120)

Heap Dump

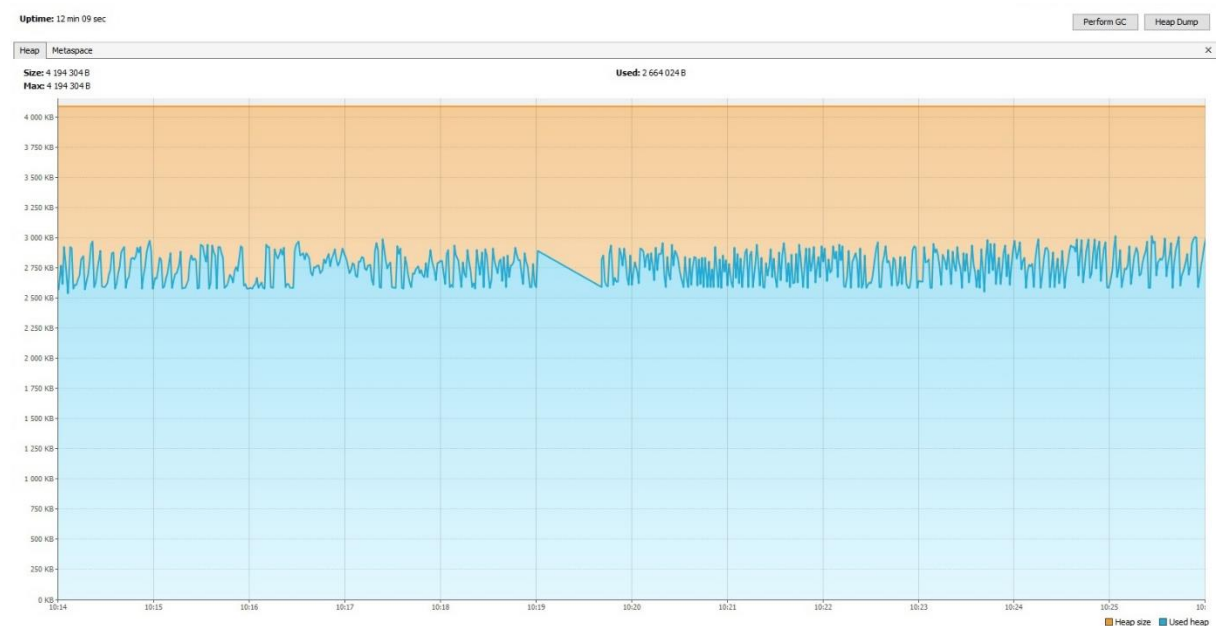
Objects | Preset: All Objects | Aggregation: | Details: | Preview | Fields | References | GC Root | Hierarchy

Name	Count	Size	Retained (sort to get)
byte[]	483 (1.4%)	1 223 094 B (40.2%)	n/a
char[]	6 741 (22.8%)	484 122 B (17.2%)	n/a
java.lang.String	6 684 (22.6%)	187 152 B (6.6%)	n/a
java.lang.Object[]	1 796 (6.1%)	158 984 B (5.6%)	n/a
java.lang.reflect.Method	482 (1.6%)	70 372 B (2.5%)	n/a
int[]	826 (2.8%)	50 256 B (1.8%)	n/a
java.util.HashMapNode	923 (3.1%)	40 612 B (1.4%)	n/a
java.util.HashMapNode[]	205 (0.7%)	33 464 B (1.2%)	n/a
java.lang.ref.SoftReference	496 (1.7%)	27 776 B (1%)	n/a
java.lang.String[]	397 (1.3%)	24 080 B (8.9%)	n/a
java.util.concurrent.ConcurrentHashMapNode	484 (1.6%)	21 296 B (0.8%)	n/a
java.lang.reflect.Constructor	170 (0.6%)	20 740 B (0.7%)	n/a
java.lang.Class[]	513 (1.7%)	18 632 B (0.7%)	n/a
java.util.HashMap	288 (1%)	18 432 B (0.7%)	n/a
java.util.HashMapEntry	331 (1.1%)	14 560 B (0.5%)	n/a
java.util.HashMapEntry[]	83 (0.3%)	12 864 B (0.5%)	n/a
sun.misc.FDBigInteger	341 (1.2%)	11 253 B (0.4%)	n/a
java.util.concurrent.ConcurrentHashMapNode[]	26 (0.1%)	10 608 B (0.4%)	n/a
java.lang.invoke.MemberName	196 (0.7%)	10 192 B (0.4%)	n/a
java.lang.ref.SoftReference[]	94 (0.3%)	10 152 B (0.4%)	n/a
java.lang.invoke.LambdaForm\$Name	200 (0.7%)	10 000 B (0.4%)	n/a
java.lang.invoke.MethodType	140 (0.5%)	8 960 B (0.3%)	n/a
java.lang.invoke.MethodType\$ConcurrentWeakInternSet\$WeakEntry	142 (0.5%)	7 384 B (0.3%)	n/a
java.util.WeakHashMapEntry	108 (0.4%)	7 344 B (0.3%)	n/a
java.lang.Long	303 (1%)	7 272 B (0.3%)	n/a
javax.management.ImmutableDescriptor	189 (0.6%)	6 804 B (0.2%)	n/a
java.util.WeakHashMapEntry[]	34 (0.1%)	6 384 B (0.2%)	n/a
java.lang.Integer	279 (0.9%)	5 980 B (0.2%)	n/a
sun.reflect.DelegatingClassLoader	57 (0.2%)	5 529 B (0.2%)	n/a
java.lang.invoke.LambdaForm\$Name[]	70 (0.2%)	5 464 B (0.2%)	n/a
java.lang.invoke.MethodHandle[]	28 (0.1%)	5 440 B (0.2%)	n/a
java.io.ObjectStreamField	101 (0.3%)	5 353 B (0.2%)	n/a
com.sun.jmx.mbeanserver.ConvertingMethod	128 (0.4%)	5 248 B (0.2%)	n/a
java.lang.ref.Finalizer	81 (0.3%)	5 184 B (0.2%)	n/a
java.util.TreeMapEntry	90 (0.3%)	5 130 B (0.2%)	n/a
java.lang.ref.WeakReference	106 (0.4%)	5 088 B (0.2%)	n/a
java.util.Vector	139 (0.5%)	5 048 B (0.2%)	n/a
java.util.Hashtable	76 (0.3%)	4 864 B (0.2%)	n/a

Решение:

- очищать буферы
- убрать ненужные String.valueOf()

Результат:



Вывод

При выполнении данной лабораторной работы мы написали свой первый MBean и познакомились с JConsole и VisualVM

