Федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский государственный университет

информационных технологий, механики и оптики

Факультет прикладной информатики и компьютерных технологий

ITMO UNIVERSITY

# Основы Программной Инженерии

## Лабораторная работа №4

## Вариант 550

Группа: P3218

Студент: Петкевич Константин

Преподаватель: Харитонова А.Е.

г. Санкт-Петербург

## Текст Задания

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 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 время (в мс), прошедшее с момента запуска виртуальной машины.

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. Получить HeapDump, и с помощью утилиты VisualVM локализовать и устранить "утечку памяти".

# Mbean

## *Класс GraphAgent*

```
public class GraphAgent implements NotificationListener {
   private MBeanServer mbs = null;
   public GraphAgent() throws Exception {
      this.mbs = ManagementFactory.getPlatformMBeanServer();
      ObjectName intervalCounterName = new ObjectName("management:type=Interval,name=ChiefInterval");
      Interval chiefInterval = new Interval();
      mbs.registerMBean(chiefInterval, intervalCounterName);
      ObjectName pointCounterName = new ObjectName("management:type=PointCounter,name=ChiefPointCounter");
      PointCounter chiefPointCounter = new PointCounter();
      mbs.registerMBean(chiefPointCounter, pointCounterName);
      chiefPointCounter.addNotificationListener(this, null, null);
   }
   @Override
   public void handleNotification(Notification notification, Object handback) {
      System.out.println(notification.toString());
   }
   public void newPoint(GraphPoint2D point2D, Shape shape) throws Exception {
      String[] signature = new String[]{GraphPoint2D.class.getName(), Shape.class.getName()};
      Object[] params = new Object[]{ point2D, shape };
      ObjectName intervalCounterName = new ObjectName("management:type=Interval,name=ChiefInterval");
      this.mbs.invoke(intervalCounterName, "newPoint", params, signature);
      ObjectName pointCounterName = new ObjectName("management:type=PointCounter,name=ChiefPointCounter");
      this.mbs.invoke(pointCounterName, "newPoint", params, signature);
   }
   public void coordClick() throws Exception {
      String[] signature = new String[]{ };
      Object[] params = new Object[]{ };
      ObjectName intervalCounterName = new ObjectName("management:type=Interval,name=ChiefInterval");
      this.mbs.invoke(intervalCounterName, "coordClick", params, signature);
   }
}
```

## *Класс Interval*

```
public class Interval implements IntervalMBean{
    private int clicks = 0;
    private int fitPoints = 0;
    private long startTime;
    public Interval() {
        startTime = System.currentTimeMillis();
    }
    public void newPoint(GraphPoint2D point, Shape shape) {
        if (shape.contains(point)) {
            fitPoints++;
        }
        clicks++;
    }
    @Override
    public double getInterval() {
        if (clicks == 0) {
            return 0;
        }
        long stopTime = System.currentTimeMillis();
        double timePassedFromProgramLaunch = (stopTime - startTime) / 1000;
        return timePassedFromProgramLaunch / clicks;
    }
    public void coordClick() {
        clicks++;
    }
}
```

## *Класс PointCounter*

```
public class PointCounter extends NotificationBroadcasterSupport implements PointCounterMBean{
    private int points = 0;
    private int fitPoints = 0;
    @Override
    public void newPoint(GraphPoint2D point, Shape shape) {
        if (shape.contains(point)) {
            fitPoints++;
        }
        points++;
        if (point.Y() > 61   || point.Y() < -61 ) {
            sendNotification(
                    new Notification(
                            "management.abroad",
                            this,
                            System.currentTimeMillis(),
                            "The point extends beyond the boundaries of coordinates"
                    ));
        }
    }
    @Override
    public int getFitPointsAmount() {
        return fitPoints;
    }
    @Override
    public int getPointsAmount() {
        return points;
    }
}
```

# Jconsole

Показание Mbean классов

| Name | Value |
|---|---|
| FitPointsAmount | 11 |
| PointsAmount | 23 |

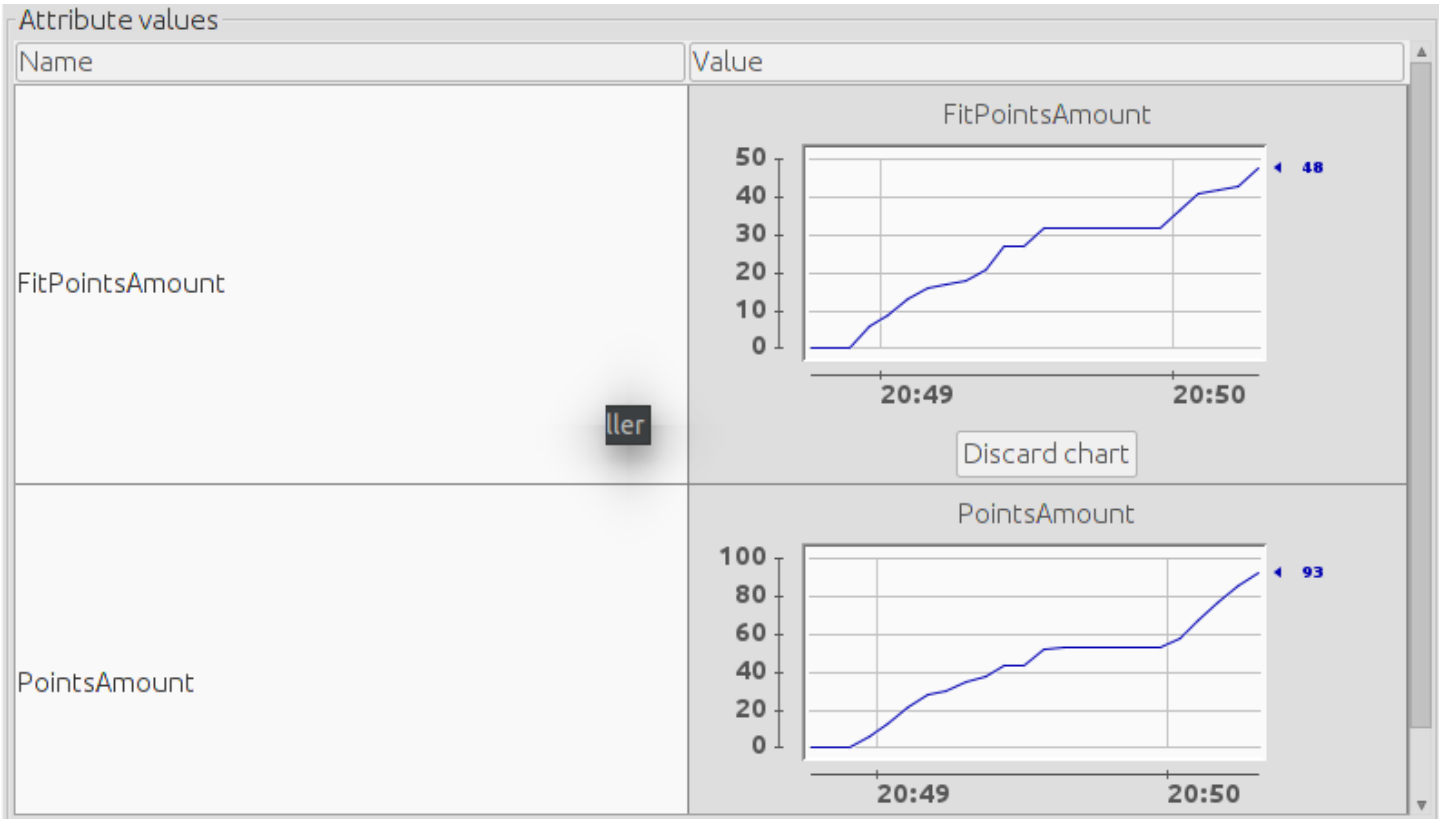
| Name | Value |
|---|---|
| Interval | 3.9166666666666665 |

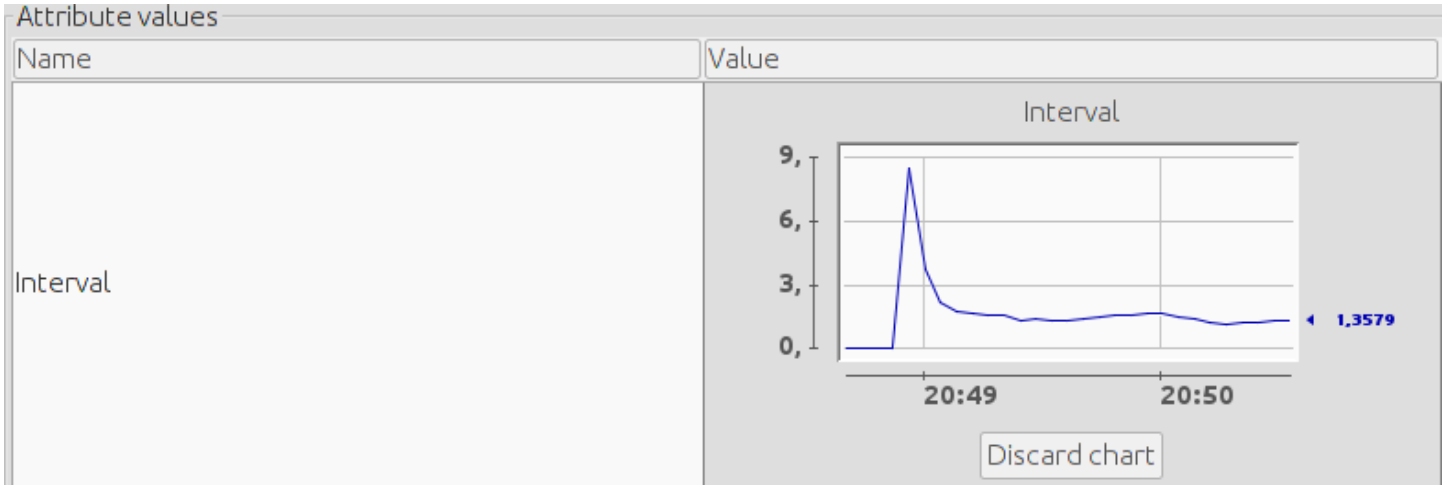
| TimeStamp | Type | UserData | SeqNum | Message | Event | Source |
|---|---|---|---|---|---|---|
| 20:42:18:931 | management.abroad | | 1491414138931 | The point extends be... | javax.management.N... | management:type=Point... |
| 20:42:18:759 | management.abroad | | 1491414138759 | The point extends be... | javax.management.N... | management:type=Point... |

Время, прошедшее с момента запуска виртуальной машины

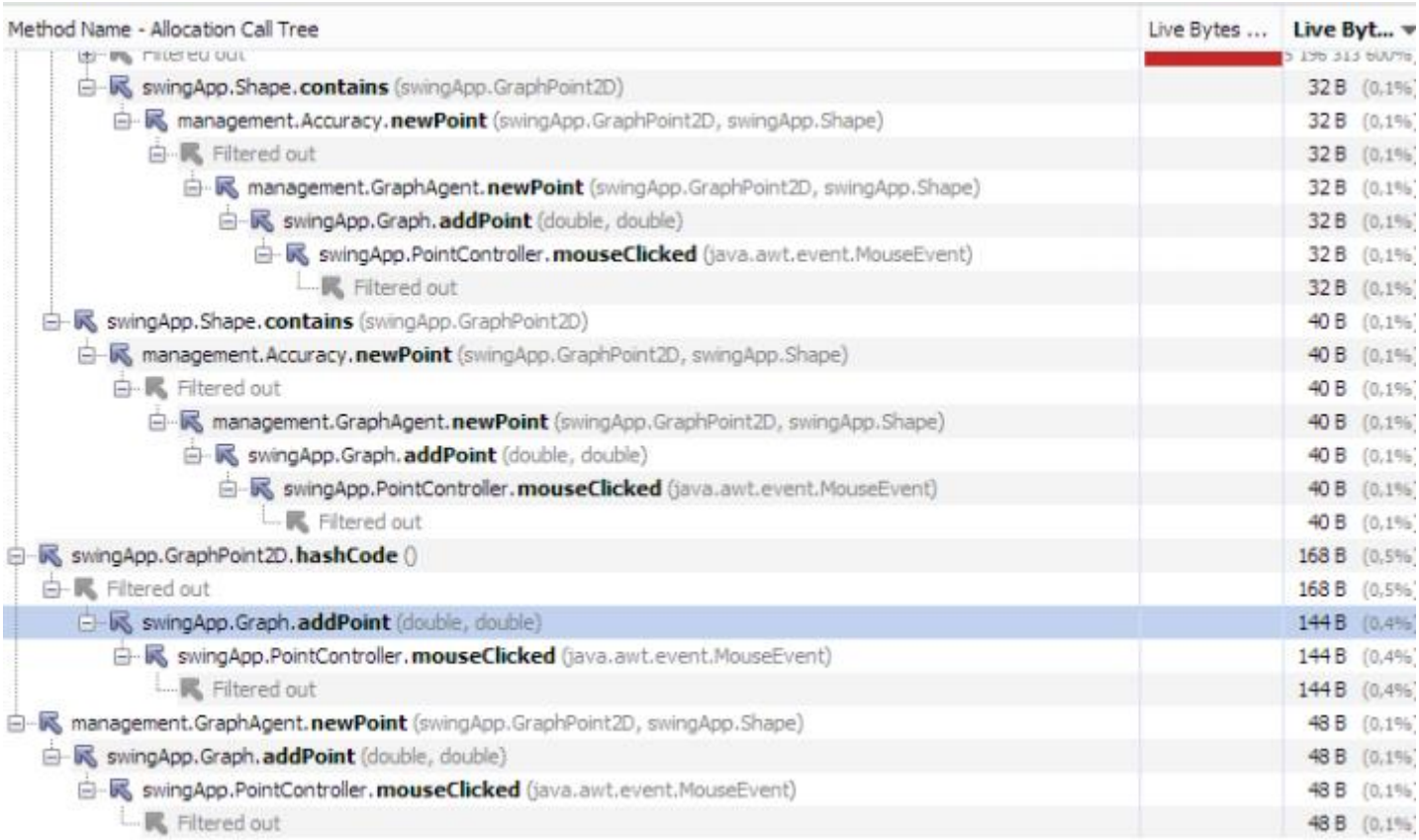
**Uptime:** 4 minutes
**Process CPU time:** 10,480 seconds
**JIT compiler:** HotSpot 64-Bit Tiered Compilers
**Total compile time:** 6,025 seconds

# VisualVM



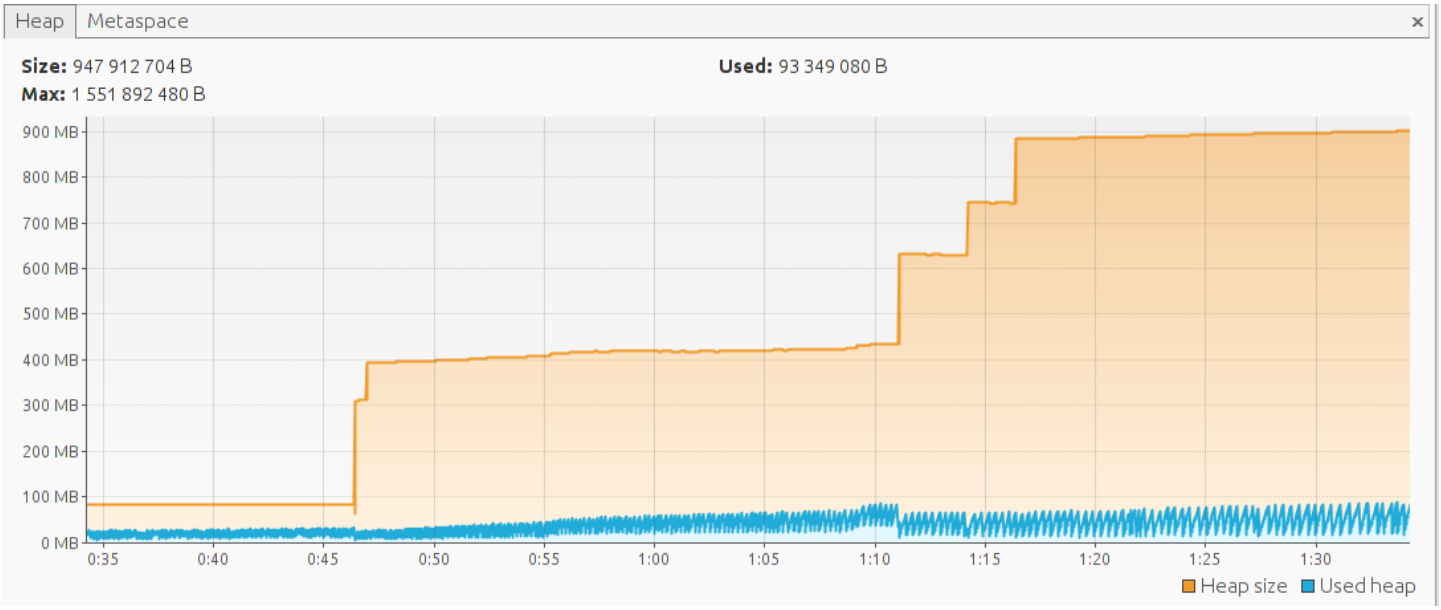



Нахождение класса, объекты которого занимают больше всего места




Большее количество памяти занимают массивы объектов — Object[].
А именно пользовательский класс Graph, который хранит в себе HashSet всех точек.

# Нахождение и устранение утечки памяти




После сравнения snapshot`ов была замеченная утечка в Char[]

| Method Name - Allocation Call Tree | Live Bytes [... | Live Bytes | Live Objects | Allocated Obj... | Avg. Age | Generations |
|---|---|---|---|---|---|---|
| char[] | | 611 912... (100%) | 8 145 (100%) | 2 517 106 | 0,0 | 4 |
| java.util.Arrays.**copyOfRange** (char[], int, int) | | 168 608...(27,6%) | 2 837 (34,8%) | 847 787 | 0,0 | 3 |
| java.lang.String.**<init>** (char[], int, int) | | 168 608...(27,6%) | 2 837 (34,8%) | 847 787 | 0,0 | 3 |
| java.lang.StringBuilder.**toString** () | | 134 728... (22%) | 1 929 (23,7%) | 674 323 | 0,0 | 2 |
| java.lang.StringIndexOutOfBoundsEx | | 62 560 B (10,2%) | 782 (9,6%) | 419 078 | 0,0 | 1 |
| B$4.**run** () | | 13 600 B (2,2%) | 170 (2,1%) | 37 032 | 0,0 | 1 |
| B$3.**run** () | | 13 200 B (2,2%) | 165 (2%) | 37 737 | 0,0 | 1 |
| B$1.**run** () | | 7 440 B (1,2%) | 93 (1,1%) | 24 005 | 0,0 | 1 |
| H$3.**run** () | | 6 400 B (1%) | 80 (1%) | 16 059 | 0,0 | 1 |
| H$2.**run** () | | 5 440 B (0,9%) | 68 (0,8%) | 14 525 | 0,0 | 2 |
| B$5.**run** () | | 5 120 B (0,8%) | 64 (0,8%) | 14 411 | 0,0 | 1 |
| B$6.**run** () | | 4 800 B (0,8%) | 60 (0,7%) | 17 016 | 0,0 | 1 |
| B$11.**run** () | | 4 648 B (0,8%) | 149 (1,8%) | 30 737 | 0,0 | 1 |
| B$8.**run** () | | 4 152 B (0,7%) | 130 (1,6%) | 30 194 | 0,0 | 1 |
| B$10.**run** () | | 3 360 B (0,5%) | 110 (1,4%) | 26 254 | 0,0 | 1 |
| H$1.**run** () | | 1 840 B (0,3%) | 23 (0,3%) | 6 497 | 0,0 | 1 |
| java.io.ObjectInputStream$BlockData | | 1 264 B (0,2%) | 24 (0,3%) | 531 | 0,0 | 1 |

| Method Name - Allocation Call Tree | Live Bytes [... | Live Bytes | Live Objects | Allocated Obj... | Avg. Age | Generations |
|---|---|---|---|---|---|---|
| java.lang.String.**substring** (int, int) | | 32 944 B (5,4%) | 888 (10,9%) | 172 921 | 0,0 | 2 |
| java.lang.String.**substring** (int) | | 624 B (0,1%) | 17 (0,2%) | 451 | 0,0 | 1 |
| javax.management.ObjectName.**setCar** | | 312 B (0,1%) | 3 (0%) | 92 | 0,0 | 1 |
| java.io.ObjectOutputStream$BlockDataOut | | 145 200...(23,7%) | 275 (3,4%) | 62 209 | 0,0 | 1 |
| java.io.ObjectOutputStream.**<init>** (java.io | | 145 200...(23,7%) | 275 (3,4%) | 62 209 | 0,0 | 1 |
| B.**<init>** () | | 143 616...(23,5%) | 272 (3,3%) | 62 033 | 0,0 | 1 |
| H.**<init>** () | | 143 616...(23,5%) | 272 (3,3%) | 62 033 | 0,0 | 1 |
| Lab4$1.**run** () | | 143 616...(23,5%) | 272 (3,3%) | 62 033 | 0,0 | 1 |
| java.lang.Thread.**run** () | | 143 616...(23,5%) | 272 (3,3%) | 62 033 | 0,0 | 1 |
| sun.rmi.server.MarshalOutputStream.< | | 1 584 B (0,3%) | 3 (0%) | 176 | 0,0 | 1 |
| java.util.Arrays.**copyOf** (char[], int) | | 139 592...(22,8%) | 1 580 (19,4%) | 581 227 | 0,0 | 1 |
| java.lang.AbstractStringBuilder.**ensureCa** | | 139 568...(22,8%) | 1 579 (19,4%) | 581 204 | 0,0 | 1 |
| java.lang.AbstractStringBuilder.**append** | | 137 864...(22,5%) | 1 565 (19,2%) | 580 823 | 0,0 | 1 |
| java.lang.StringBuilder.**append** (String | | 137 864...(22,5%) | 1 565 (19,2%) | 580 823 | 0,0 | 1 |
| java.lang.StringIndexOutOfBoundsI | | 71 280 B (11,6%) | 810 (9,9%) | 421 643 | 0,0 | 1 |
| java.lang.String.**substring** (int, int) | | 71 280 B (11,6%) | 810 (9,9%) | 421 643 | 0,0 | 1 |
| B$7.**run** () | | 41 624 B (6,8%) | 473 (5,8%) | 145 754 | 0,0 | 1 |

В большинстве методов «H» и «B» , действия, выполняемые в потоке, итерировались без выхода из цикла. Так же определялись методы, неиспользуемые в программе.

**Вывод:** в процессе выполнения данной лабораторной работы я познакомился с базовыми инструментами мониторинга и профилирования программ в среде Java. Данный инструменты являются мощными анализаторами программ , позволяющими устранять и дислоцировать утечки памяти. Процесс выявления проблем в программе становится во многом значительно проще при их использовании.