03: jvisualvm to debug deadlocks in Java applications

Posted on June 23, 2015 by Arulkumaran Kumaraswamipillai

This extends JConsole for debugging deadlocks in Java applications, using jvisualvm that gets shipped with your JDK.

Step 1: Java code that creates a **dead lock** situation by

a) thread-0 holding on to lock1 and waiting for the lock2,

b) and thread-1 holding on to lock2 and waiting for the lock1.

```
package com.debug.multithread;

public class DeadLockTest extends Thread {

    public static Object lock1 = new Object();
    public static Object lock2 = new Object();

    public void method1() {
        synchronized (lock1) {
            delay(500);  //some operation
            System.out.println("method1: " + Thr
            synchronized (lock2) {
                System.out.println("method1 is e
            }
        }
    }

    public void method2() {
        synchronized (lock2) {
            delay(500);   //some operation
            System.out.println("method1: " + Thr
            synchronized (lock1) {
                System.out.println("method2 is e
            }
        }
    }

    @Override
    public void run() {
        method1();
        method2();
    }

    /**
     * main metghod runs on a main thread
     * @param args
     */
    public static void main(String[] args) {
        DeadLockTest thread1 = new DeadLockTest(
        DeadLockTest thread2 = new DeadLockTest(

        thread1.start();
        thread2.start();
    }

    /**
     * The delay is to simulate some real operat
     * @param timeInMillis
     */
    private void delay(long timeInMillis) {
        try {
            Thread.sleep(timeInMillis);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

}
```

Step 2: From a DOS command prompt use "**jps**" to list the java process ids. This command is in %JAVA_HOME%/bin

```
C:\>jps

9460 DeadLockTest
1952 Jps
8148
```

Step 3: Process id "9460" is the "DeadLockTest". So, let's open **jvisualvm** that is shipped with Java in %JAVA_HOME%/bin.

16 Technical Key Areas

open all | close all

- ⊞ Best Practice (6)
- ⊞ Coding (26)
- ⊞ Concurrency (6)
- ⊞ Design Concepts (7)
- ⊞ Design Patterns (11)
- ⊞ Exception Handling (
- ⊞ Java Debugging (21)
- ⊞ Judging Experience I
- ⊞ Low Latency (7)
- ⊞ Memory Managemen
- ⊞ Performance (13)
- ⊞ QoS (8)
- ⊞ Scalability (4)
- ⊞ SDLC (6)
- ⊞ Security (13)
- ⊞ Transaction Managen

80+ step by step Java Tutorials

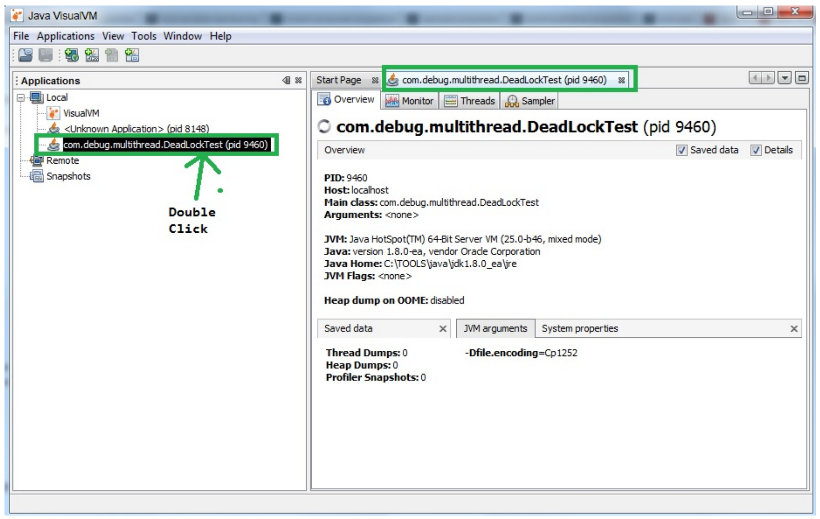
open all | close all

- ⊞ Setting up Tutorial (6)
- ⊞ Tutorial - Diagnosis (
- ⊞ Akka Tutorial (9)
- ⊞ Core Java Tutorials (
- ⊞ Hadoop & Spark Tuto
- ⊞ JEE Tutorials (19)
- ⊞ Scala Tutorials (1)
- ⊞ Spring & HIbernate T
- ⊞ Tools Tutorials (19)
- ⊞ Other Tutorials (45)

```
C:\>jvisualvm 9460
```

Step 4: jvisualvm shows where the problem is:

Double click on **DeadLockTest**.



com.debug.multithread.DeadLockTest (pid 9460)

PID: 9460
Host: localhost
Main class: com.debug.multithread.DeadLockTest
Arguments: <none>

JVM: Java HotSpot(TM) 64-Bit Server VM (25.0-b46, mixed mode)
Java: version 1.8.0-ea, vendor Oracle Corporation
Java Home: C:\TOOLS\java\jdk1.8.0_ea\jre
JVM Flags: <none>

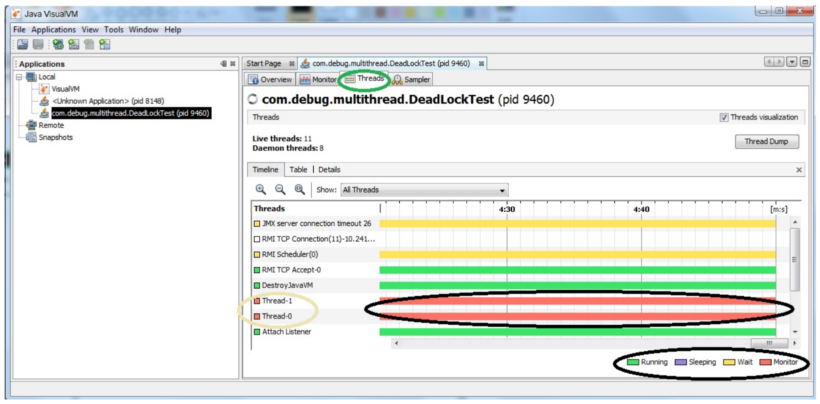
Heap dump on OOME: disabled

Thread Dumps: 0
Heap Dumps: 0
Profiler Snapshots: 0

-Dfile.encoding=Cp1252

jvisualvm

Thread states are color coded, and we are interested in the "Blocked" states, which are red in color.



com.debug.multithread.DeadLockTest (pid 9460)

Live threads: 11
Daemon threads: 8

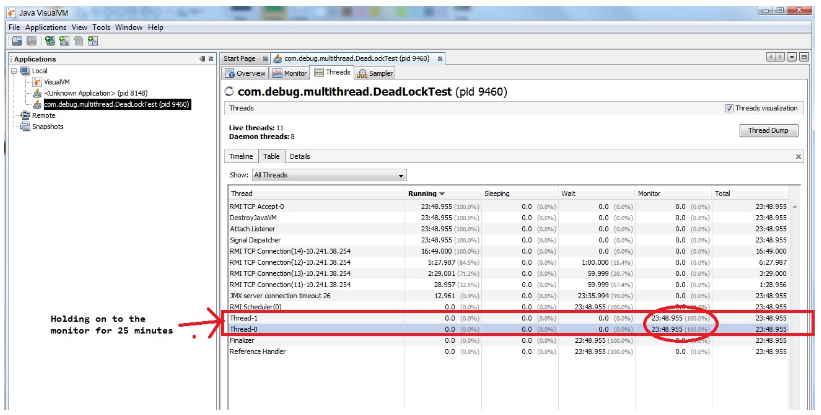
Threads
- 🟨 JMX server connection timeout 26
- ☐ RMI TCP Connection(11)-10.241...
- 🟨 RMI Scheduler(0)
- 🟩 RMI TCP Accept-0
- 🟩 DestroyJavaVM
- 🟥 Thread-1
- 🟥 Thread-0
- 🟩 Attach Listener

jvisualvm color codes the thread states

How long is the monitor being held for?

100+ Java pre-interview coding tests

How good are your .....?


Java VisualVM
File Applications View Tools Window Help
Applications
Local
VisualVM
<Unknown Application> (pid 8148)
com.debug.multithread.DeadLockTest (pid 9460)
Remote
Snapshots
Start Page
Overview
Monitor
Threads
Sampler
Threads visualization
Thread Dump
Timeline
Table
Details
Holding on to the monitor for 25 minutes

com.debug.multithread.DeadLockTest (pid 9460)

Threads

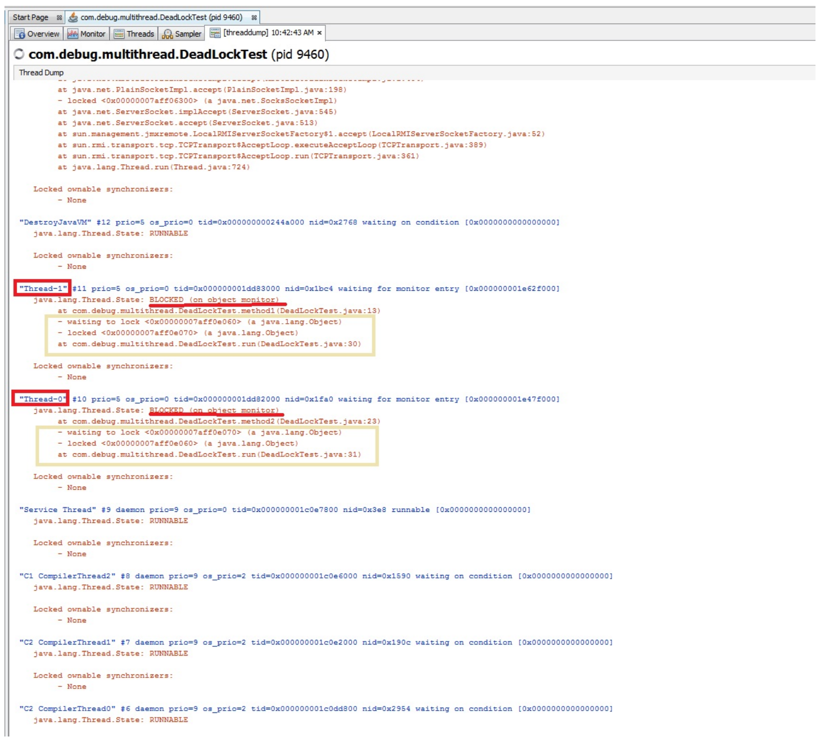
Live threads: 11
Daemon threads: 8

Show: All Threads

Thread	Running	Sleeping	Wait	Monitor	Total
RMI TCP Accept-0	23:48.955 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	23:48.955
DestroyJavaVM	23:48.955 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	23:48.955
Attach Listener	23:48.955 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	23:48.955
Signal Dispatcher	23:48.955 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	23:48.955
RMI TCP Connection(14)-10.241.38.254	16:49.000 (100.0%)	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	16:49.000
RMI TCP Connection(12)-10.241.38.254	5:27.987 (84.5%)	0.0 (0.0%)	1:00.000 (15.4%)	0.0 (0.0%)	6:27.987
RMI TCP Connection(13)-10.241.38.254	2:29.001 (71.2%)	0.0 (0.0%)	59.999 (28.7%)	0.0 (0.0%)	3:29.000
RMI TCP Connection(11)-10.241.38.254	28.957 (32.5%)	0.0 (0.0%)	59.999 (67.4%)	0.0 (0.0%)	1:28.956
JMX server connection timeout 26	12.961 (0.9%)	0.0 (0.0%)	23:35.994 (99.0%)	0.0 (0.0%)	23:48.955
RMI Scheduler(0)	0.0 (0.0%)	0.0 (0.0%)	23:48.955 (100.0%)	0.0 (0.0%)	23:48.955
Thread-1	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	23:48.955 (100.0%)	23:48.955
Thread-0	0.0 (0.0%)	0.0 (0.0%)	0.0 (0.0%)	23:48.955 (100.0%)	23:48.955
Finalizer	0.0 (0.0%)	0.0 (0.0%)	23:48.955 (100.0%)	0.0 (0.0%)	23:48.955
Reference Handler	0.0 (0.0%)	0.0 (0.0%)	23:48.955 (100.0%)	0.0 (0.0%)	23:48.955

jvisualvm the monitor (i.e. lock) has been held for a long time

Click the button “**Thread Dump**” to get more details



com.debug.multithread.DeadLockTest (pid 9460)

Thread Dump

```
        at java.net.PlainSocketImpl.accept(PlainSocketImpl.java:198)
        - locked <0x00000007aff06300> (a java.net.SocksSocketImpl)
        at java.net.ServerSocket.implAccept(ServerSocket.java:545)
        at java.net.ServerSocket.accept(ServerSocket.java:513)
        at sun.management.jmxremote.LocalRMIServerSocketFactory$1.accept(LocalRMIServerSocketFactory.java:52)
        at sun.rmi.transport.tcp.TCPTransport$AcceptLoop.executeAcceptLoop(TCPTransport.java:389)
        at sun.rmi.transport.tcp.TCPTransport$AcceptLoop.run(TCPTransport.java:361)
        at java.lang.Thread.run(Thread.java:724)

   Locked ownable synchronizers:
        - None

"DestroyJavaVM" #12 prio=5 os_prio=0 tid=0x000000000244a000 nid=0x2768 waiting on condition [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"Thread-1" #11 prio=5 os_prio=0 tid=0x000000001dd83000 nid=0x1bc4 waiting for monitor entry [0x000000001e62f000]
   java.lang.Thread.State: BLOCKED (on object monitor)
        at com.debug.multithread.DeadLockTest.method1(DeadLockTest.java:13)
        - waiting to lock <0x00000007aff0e060> (a java.lang.Object)
        - locked <0x00000007aff0e070> (a java.lang.Object)
        at com.debug.multithread.DeadLockTest.run(DeadLockTest.java:30)

   Locked ownable synchronizers:
        - None

"Thread-0" #10 prio=5 os_prio=0 tid=0x000000001dd82000 nid=0x1fa0 waiting for monitor entry [0x000000001e47f000]
   java.lang.Thread.State: BLOCKED (on object monitor)
        at com.debug.multithread.DeadLockTest.method2(DeadLockTest.java:23)
        - waiting to lock <0x00000007aff0e070> (a java.lang.Object)
        - locked <0x00000007aff0e060> (a java.lang.Object)
        at com.debug.multithread.DeadLockTest.run(DeadLockTest.java:31)

   Locked ownable synchronizers:
        - None

"Service Thread" #9 daemon prio=9 os_prio=0 tid=0x000000001c0e7800 nid=0x3e8 runnable [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"C1 CompilerThread2" #8 daemon prio=9 os_prio=2 tid=0x000000001c0e6000 nid=0x1590 waiting on condition [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"C2 CompilerThread1" #7 daemon prio=9 os_prio=2 tid=0x000000001c0e2000 nid=0x190c waiting on condition [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

   Locked ownable synchronizers:
        - None

"C2 CompilerThread0" #6 daemon prio=9 os_prio=2 tid=0x000000001c0dd800 nid=0x2954 waiting on condition [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE
```

Thread dump from jvisualvm

Popular Member Posts

♦ 11 Spring boot interview questions & answers

905 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

816 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

427 views

18 Java scenarios based interview Questions and Answers

409 views

♦ 7 Java debugging interview questions & answers

324 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

311 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

304 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

301 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

251 views

♦ Object equals Vs == and pass by reference Vs value

234 views

0 Like Share reddit 0 G+1 Share

Bio **Latest Posts**

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ ♥ Creating a Maven profile to compile using a different JDK version tutorial

♥ javap for debugging and better understanding some Java concepts with 3 practical examples ›

Posted in Concurrency, Java Debugging, jvisualvm profiling, member-paid

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from "Blah blah" to "Wow"? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer