Java-Success.com

Register | Login | Logout | Contact Us

Industrial strength Java/JEE Career Companion to open more doors

search here … Go

Home | Java FAQs | 600+ Java Q&As | Career | Tutorials | Member | Why?

Can u Debug? | Java 8 ready? | Top X | Productivity Tools | Judging Experience?

Home › Tech Key Areas › 13 Technical Key Areas Interview Q&A › Concurrency ›
♦ 5 Ways to debug Java thread-safety issues

♦ 5 Ways to debug Java thread-safety issues

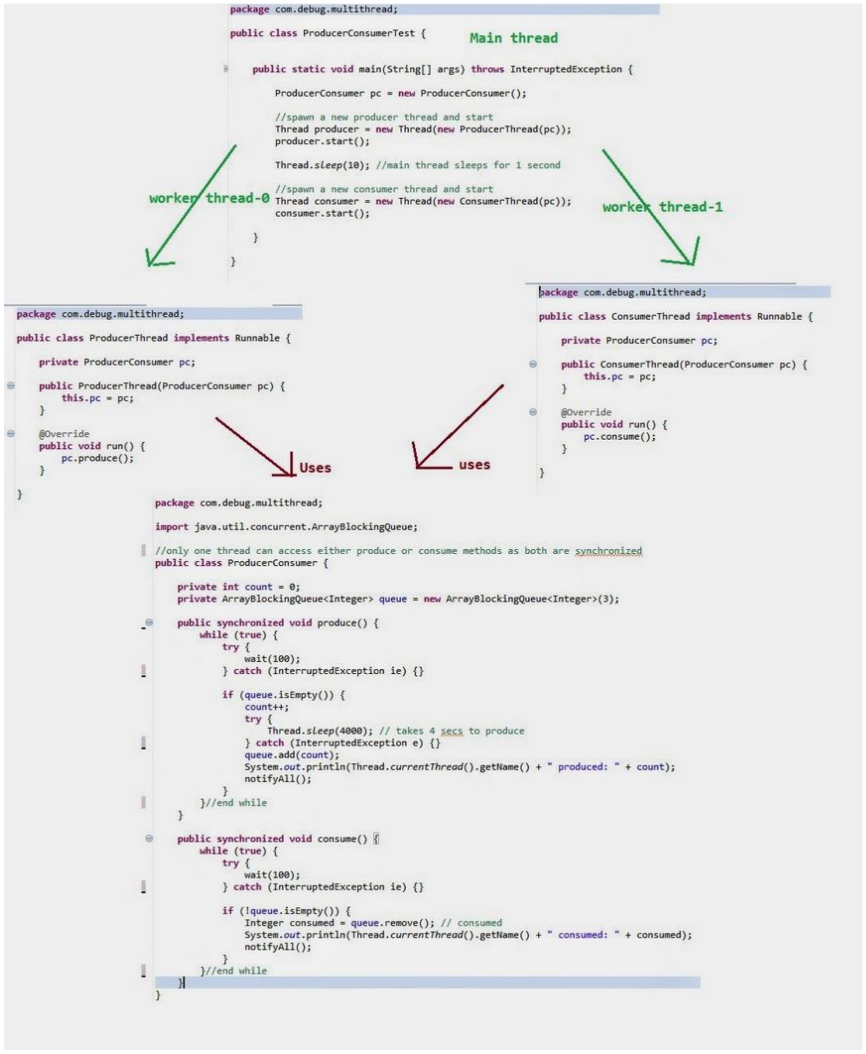
Posted on October 31, 2014 by Arulkumaran Kumaraswamipillai — 4 Comments
↓

Tutorial style debugging of Java thread safety issues. Debugging concurrency issues are not easy.

#1 List all possible causes and add extensive log statements and write test cases to prove or disprove your theories. The log statements will have something like

```
log.info(Thread.currentThread().getName() + " pro
System.out.println(Thread.currentThread().getName

```

#2 Using your IDE debugging capability by setting a conditional break point. *Thread.currentThread().getName().equals("Thread-0")*. For example, stopping for particular thread as demonstrated below step by step in eclipse IDE.



```java
package com.debug.multithread;

public class ProducerConsumerTest {

    public static void main(String[] args) throws InterruptedException {

        ProducerConsumer pc = new ProducerConsumer();

        //spawn a new producer thread and start
        Thread producer = new Thread(new ProducerThread(pc));
        producer.start();

        Thread.sleep(10); //main thread sleeps for 1 second

        //spawn a new consumer thread and start
        Thread consumer = new Thread(new ConsumerThread(pc));
        consumer.start();

    }

}
```

```java
package com.debug.multithread;

public class ProducerThread implements Runnable {

    private ProducerConsumer pc;

    public ProducerThread(ProducerConsumer pc) {
        this.pc = pc;
    }

    @Override
    public void run() {
        pc.produce();
    }

}
```

```java
package com.debug.multithread;

public class ConsumerThread implements Runnable {

    private ProducerConsumer pc;

    public ConsumerThread(ProducerConsumer pc) {
        this.pc = pc;
    }

    @Override
    public void run() {
        pc.consume();
    }

}
```

```java
package com.debug.multithread;

import java.util.concurrent.ArrayBlockingQueue;

//only one thread can access either produce or consume methods as both are synchronized
public class ProducerConsumer {

    private int count = 0;
    private ArrayBlockingQueue<Integer> queue = new ArrayBlockingQueue<Integer>(3);

    public synchronized void produce() {
        while (true) {
            try {
                wait(100);
            } catch (InterruptedException ie) {}

            if (queue.isEmpty()) {
                count++;
                try {
                    Thread.sleep(4000); // takes 4 secs to produce
                } catch (InterruptedException e) {}
                queue.add(count);
                System.out.println(Thread.currentThread().getName() + " produced: " + count);
                notifyAll();
            }
        }//end while
    }

    public synchronized void consume() {
        while (true) {
            try {
                wait(100);
            } catch (InterruptedException ie) {}

            if (!queue.isEmpty()) {
                Integer consumed = queue.remove(); // consumed
                System.out.println(Thread.currentThread().getName() + " consumed: " + consumed);
                notifyAll();
            }
        }//end while
    }
}
```

Get the code used above:

Java Producer-Consumer multi threading code.

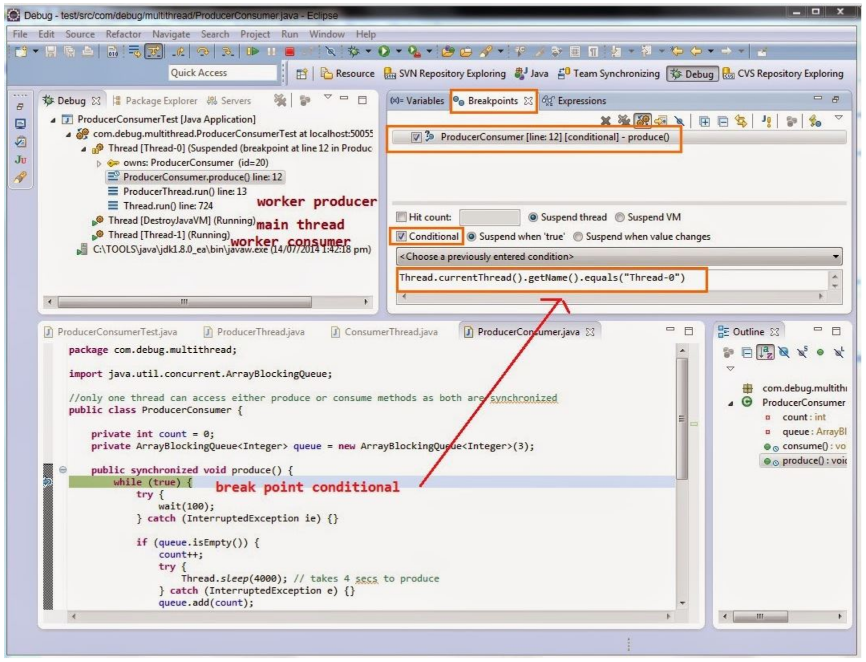
The above code continuously produces output like:

```
Thread-0 produced: 1
Thread-1 consumed: 1
Thread-0 produced: 2
Thread-1 consumed: 2
Thread-0 produced: 3
Thread-1 consumed: 3
Thread-0 produced: 4
Thread-1 consumed: 4
```

9

You can add a break point to the ***ProducerConsumer*** that is used by both worker threads ***ProducerThread*** and ***ConsumerThread***. Both these worker threads are spawned by the default main thread.

Step 1: Create a conditional debug point as shown below in the first line of the produce method.



```
ProducerConsumerTest [Java Application]
  com.debug.multithread.ProducerConsumerTest at localhost:50055
    Thread [Thread-0] (Suspended (breakpoint at line 12 in Produc
      owns: ProducerConsumer  (id=20)
      ProducerConsumer.produce() line: 12
      ProducerThread.run() line: 13
      Thread.run() line: 724
    Thread [DestroyJavaVM] (Running)
    Thread [Thread-1] (Running)
  C:\TOOLS\java\jdk1.8.0_ea\bin\javaw.exe (14/07/2014 1:42:18 pm)
```

```
[✓] ProducerConsumer [line: 12] [conditional] - produce()

[ ] Hit count:        (●) Suspend thread   ( ) Suspend VM
[✓] Conditional   (●) Suspend when 'true'   ( ) Suspend when value changes
<Choose a previously entered condition>
Thread.currentThread().getName().equals("Thread-0")
```

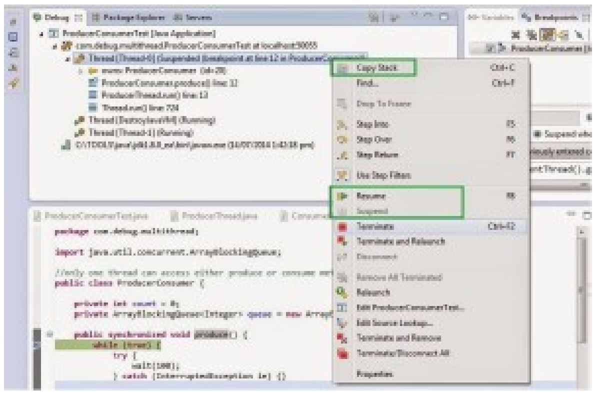
```java
package com.debug.multithread;

import java.util.concurrent.ArrayBlockingQueue;

//only one thread can access either produce or consume methods as both are synchronized
public class ProducerConsumer {

    private int count = 0;
    private ArrayBlockingQueue<Integer> queue = new ArrayBlockingQueue<Integer>(3);

    public synchronized void produce() {
        while (true) {
            try {
                wait(100);
            } catch (InterruptedException ie) {}

            if (queue.isEmpty()) {
                count++;
                try {
                    Thread.sleep(4000); // takes 4 secs to produce
                } catch (InterruptedException e) {}
                queue.add(count);
```

Step 2: Run the *ProducerConsumerTest* in debug mode. The execution stops on the break point when worker "**Thread-0**" enters the break point. In the above example only one thread enters *produce( )* method. But in industrial applications you can have many threads.

You also have options to **suspend** and **resume** the threads you want as shown below with right-click context menu in the debug window. When you are suspended, you can also copy the stack at that suspended point in time.

16 Technical Key Areas

80+ step by step Java Tutorials

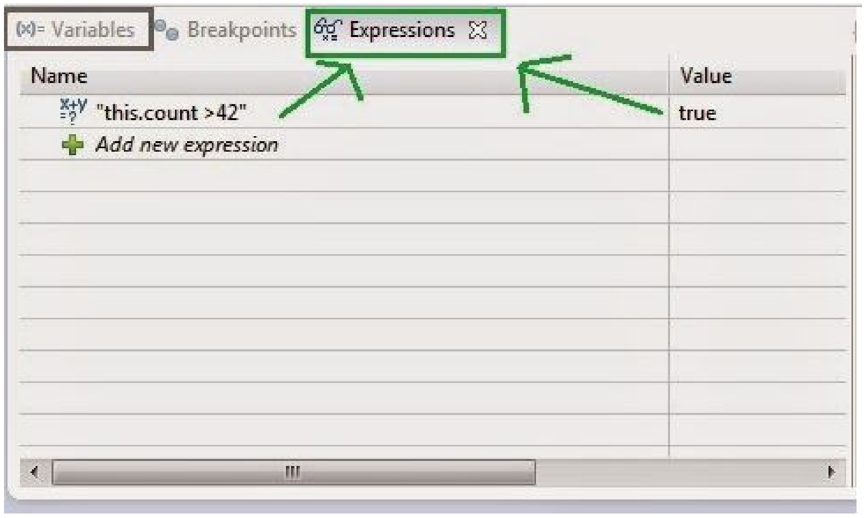


```
ProducerConsumerTest [Java Application]
  com.debug.multithread.ProducerConsumerTest at localhost:50055
    Thread [Thread-0] (Suspended (breakpoint at line 12 in ProducerConsumer))
      owns: ProducerConsumer (id=28)
      ProducerConsumer.produce() line: 12
      ProducerThread.run() line: 12
      Thread.run() line: 724
    Thread [DestroyJavaVM] (Running)
    Thread [Thread-1] (Running)
  C:\TOOLS\java\jdk1.8.0_xx\bin\javaw.exe (14/07/2014 1:42:18 pm)
```

```
package com.debug.multithread;

import java.util.concurrent.ArrayBlockingQueue;

//only one thread can access either produce or consume met
public class ProducerConsumer {

    private int count = 0;
    private ArrayBlockingQueue<Integer> queue = new Array

    public synchronized void produce() {
        while (true) {
            try {
                wait(100);
            } catch (InterruptedException ie) {}
```

Debugging multi-threaded app

You can also **inspect and watch** shared variables to ascertain any thread-safety issues.



Name	Value
"this.count >42"	true

The above diagram adds a **watch expression** on a shared variable.

#3 Thread dumps are very useful for diagnosing synchronization problems such as deadlocks. The trick is to take 5 or 6 sets of thread dumps at an interval of 5 seconds between each to have a log file that has 25 to 30 seconds worth of run-time action. For thread dumps, use kill -3 in Unix and CTRL+BREAK in Windows. There are tools like Thread Dump Analyzer (TDA), Samurai, etc. to derive useful information from the thread dumps to find where the problem is. For example, Samurai colors idle threads in grey, blocked

♥ javap for debu
♥ Top 10 Eclipse
♦ 5 Ways to debu
Debugging like a
Remote debuggi
⊞ Java 8 Tutorial (4)
⊞ JSON and Java Tu
⊞ XML and Java Tuto
⊞ CSV and Java Tuto
⊞ Hadoop & Spark Tuto
⊞ JEE Tutorials (19)
⊞ Scala Tutorials (1)
⊞ Spring & HIbernate Tu
⊞ Tools Tutorials (19)
⊞ Other Tutorials (45)

100+ Java pre-interview coding tests

open all | close all
⊞ Can you write code? (
⊞ ♦ Complete the given
⊞ Converting from A to I
⊞ Designing your classe
⊞ Java Data Structures
⊞ Passing the unit tests
⊞ What is wrong with th
⊞ Writing Code Home A
⊞ Written Test Core Jav
⊞ Written Test JEE (1)

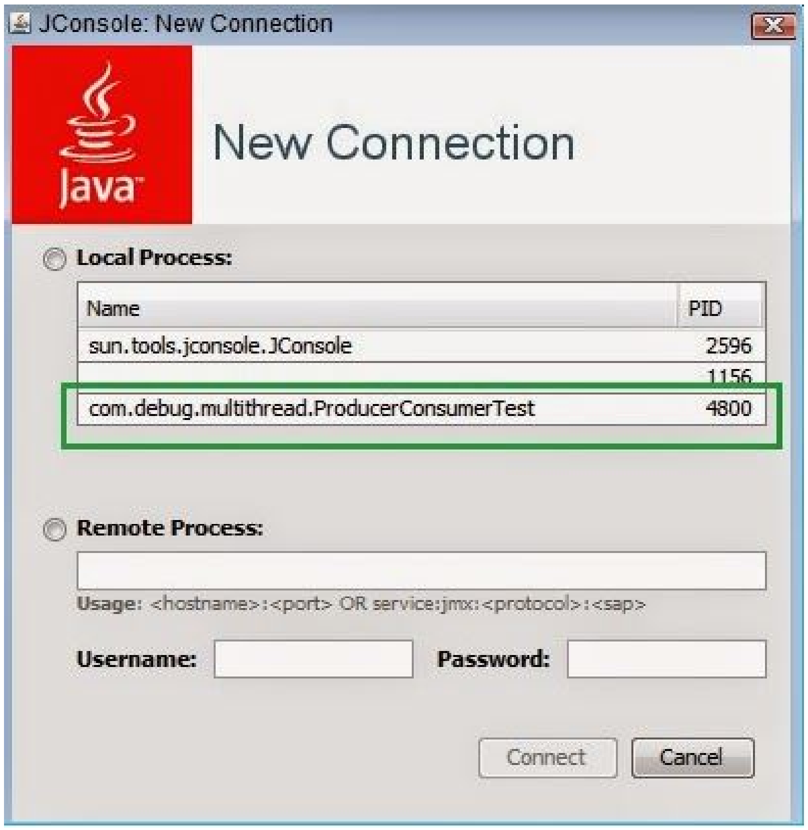
How good are your .....?

open all | close all
⊞ Career Making Know-
⊞ Job Hunting & Resum

threads in red, and running threads in green. You must pay more attention to those red threads.

Creating a thread dump in windows

Step 1: While the *ProducerConsumerTest* is running, open a DOS command prompt at type ***jconsole***.



Name	PID
sun.tools.jconsole.JConsole	2596
	1156
com.debug.multithread.ProducerConsumerTest	4800

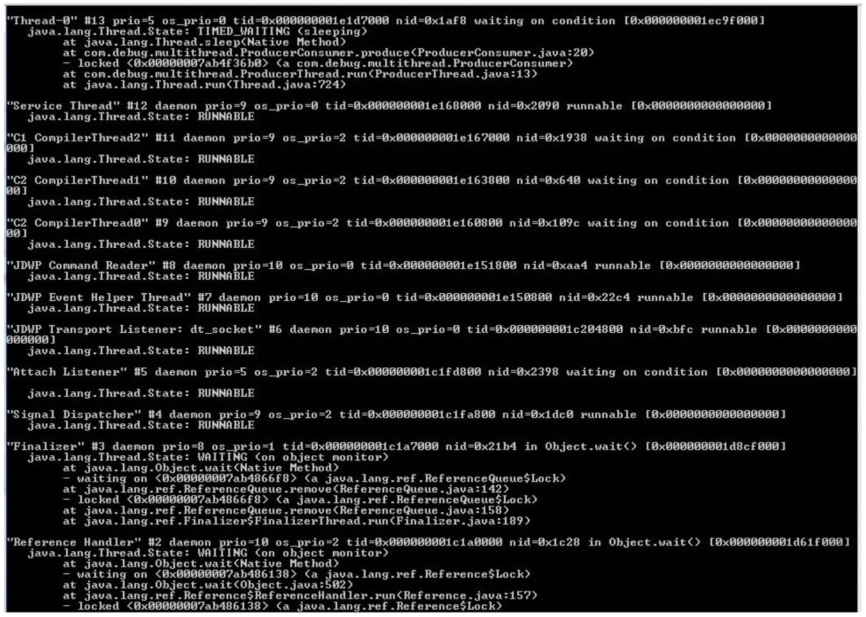
Note down the **process id**: **4800**. Connect to 4800, and

Step 2: You can detect any deadlocks by clicking on the "**Detect Deadlock**" button in the threads tab.

Step 3: To get a thread dump, open a DOS command prompt and type **jstat [pid]**

```
jstat 4800
```

This will produce a stack trace. The stack trace looks something like



```
"Thread-0" #13 prio=5 os_prio=0 tid=0x000000001e1d7000 nid=0x1af8 waiting on condition [0x000000001ec9f000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
        at java.lang.Thread.sleep(Native Method)
        at com.debug.multithread.ProducerConsumer.produce(ProducerConsumer.java:20)
        - locked <0x00000007ab4f36b0> (a com.debug.multithread.ProducerConsumer)
        at com.debug.multithread.ProducerThread.run(ProducerThread.java:13)
        at java.lang.Thread.run(Thread.java:724)

"Service Thread" #12 daemon prio=9 os_prio=0 tid=0x000000001e168000 nid=0x2090 runnable [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"C1 CompilerThread2" #11 daemon prio=9 os_prio=2 tid=0x000000001e167000 nid=0x1938 waiting on condition [0x0000000000000
000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread1" #10 daemon prio=9 os_prio=2 tid=0x000000001e163800 nid=0x640 waiting on condition [0x00000000000000
00]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread0" #9 daemon prio=9 os_prio=2 tid=0x000000001e160800 nid=0x109c waiting on condition [0x00000000000000
00]
   java.lang.Thread.State: RUNNABLE

"JDWP Command Reader" #8 daemon prio=10 os_prio=0 tid=0x000000001e151800 nid=0xaa4 runnable [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"JDWP Event Helper Thread" #7 daemon prio=10 os_prio=0 tid=0x000000001e150800 nid=0x22c4 runnable [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"JDWP Transport Listener: dt_socket" #6 daemon prio=10 os_prio=0 tid=0x000000001c204800 nid=0xbfc runnable [0x0000000000
000000]
   java.lang.Thread.State: RUNNABLE

"Attach Listener" #5 daemon prio=5 os_prio=2 tid=0x000000001c1fd800 nid=0x2398 waiting on condition [0x0000000000000000]

   java.lang.Thread.State: RUNNABLE

"Signal Dispatcher" #4 daemon prio=9 os_prio=2 tid=0x000000001c1fa800 nid=0x1dc0 runnable [0x0000000000000000]
   java.lang.Thread.State: RUNNABLE

"Finalizer" #3 daemon prio=8 os_prio=1 tid=0x000000001c1a7000 nid=0x21b4 in Object.wait() [0x000000001d8cf000]
   java.lang.Thread.State: WAITING (on object monitor)
        at java.lang.Object.wait(Native Method)
        - waiting on <0x00000007ab4866f8> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:142)
        - locked <0x00000007ab4866f8> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:158)
        at java.lang.ref.Finalizer$FinalizerThread.run(Finalizer.java:189)

"Reference Handler" #2 daemon prio=10 os_prio=2 tid=0x000000001c1a0000 nid=0x1c28 in Object.wait() [0x000000001d61f000]
   java.lang.Thread.State: WAITING (on object monitor)
        at java.lang.Object.wait(Native Method)
        - waiting on <0x00000007ab486138> (a java.lang.ref.Reference$Lock)
        at java.lang.Object.wait(Object.java:502)
        at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:157)
        - locked <0x00000007ab486138> (a java.lang.ref.Reference$Lock)
```

You need to pay attention to blocked threads, and there are tools like Thread Dump Analyzer (TDA), Samurai, etc to analyze thread dumps. These tools color code waiting threads, running threads, etc.

jstack and **jconsole** are provided with your JDK installation under jdk[version]/bin.

#4 There are static analysis tools like ***Sonar***, ***ThreadCheck***, etc for catching concurrency bugs at compile-time by analyzing the byte code. Sonar produces reports with recommendations.

#5 Manually reviewing the code for any obvious thread-safety issues. Good knowledge of multi-threading is required. Here are a few scenarios based examples & tutorials to improve your understanding.

1. Review the multi-threaded code shown below and then answer the following 5 questions relating to thread-safety?

2. Understanding Spring bean scopes. A "prototype" scoped bean will be created each time it is "injected", whereas a "singleton" scoped bean will be created once and shared within the application context. Any Bean without a STATE can be singleton. For example beans that easily qualify to be singleton are DAO, Service, and Controller. Choosing a "Singleton" type for a bean that needs to maintain it's state will cause "thread-safety" issues because each thread would try to impose it's own state on that bean there by corrupting the data. 9 Spring Bean scopes interview Q&A

3. More Java Multithreading scenarios interview questions answered covers writing thread-safe lazily initialized singletons, writing thread-safe Counters or custom database sequence number generators, writing thread-safe producer & consumer communication, thread starvation & deadlock issues, & production issues debugging.

Concurrency issues debugging:

1. JConsole for debugging deadlocks in Java applications

2. jvisualvm to debug deadlocks in Java applications

37 Java debugging interview questions & answers**.**

Popular Posts

♦ 11 Spring boot interview questions & answers

827 views

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

767 views

18 Java scenarios based interview Questions and Answers

400 views

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

389 views

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

296 views

♦ 7 Java debugging interview questions & answers

293 views

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

286 views

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

279 views

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

240 views

001B: ♦ Java architecture & design concepts interview questions & answers

202 views

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**

‹ Debugging like a pro with eclipse IDE tutorial for Java developers

♥ Top 10 Eclipse short-cut keys every Java developer using eclipse IDE must know ›

Posted in Concurrency, Debugging Tutorial, Java Debugging, Java Key Area Essentials

4 comments on "♦ 5 Ways to debug Java thread-safety issues"

gurdeep singh says:
November 3, 2015 at 1:40 pm

Hi Arulkumaran ,

i am preparing for exp java interview. i have a question .. got confsed.. should i buy your book OR take subscription for website..

will it be diffrent content in book & website.

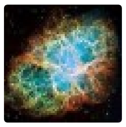
Reply

Arulkumaran Kumaraswamipillai says:
November 3, 2015 at 1:45 pm

The books were last updated in 2009. The online content are updated recently, has a lot more content, and categorized with FAQs, Key Areas, Spring, Core Java topics, etc. So, I would definitely go for the subscription.

Reply

prash2819 says:
June 15, 2015 at 12:20 pm



Hi Arul,
Where can I find the source for Debugging Java thread-safety, multi-threading, or concurrency issues tutorial code uyou are using?
Thanks
Prashanth

Reply

akumaras says:
June 15, 2015 at 3:21 pm

I have provided the link now at the bottom of the post.
http://www.java-success.com/producer-and-consumer-java-multi-threading-code/

Reply

Leave a Reply

Logged in as geethika. Log out?

Comment

Post Comment

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