Java 7 Launch and Apache Lucene / Solr:

Crashes and File Corruption due to Hotspot Bugs

http://s.apache.org/Java7LaunchBugBlog

Uwe Schindler

Apache Lucene Core Committer & PMC Member

uschindler@apache.org

@ThetaPh1

SD DataSolutions GmbH, Wätjenstr. 49, 28213 Bremen, Germany
Tel: +49 421 40889785-0, http://www.sd-datasolutions.de

My Background

- I am committer and PMC member of *Apache Lucene and Solr*. My main focus is on development of Lucene Java.
- Implemented *fast numerical search* and maintaining the *new attribute-based text analysis API*. Well known as *Generics and Sophisticated Backwards Compatibility Policeman*.
- Working as consultant and software architect for *SD DataSolutions GmbH* in Bremen, Germany. The main task is maintaining *PANGAEA (Publishing Network for Geoscientific & Environmental Data)* where I implemented the portal's geo-spatial retrieval functions with *Lucene Java*.
- Talks about Lucene at various international conferences like ApacheCon EU/US, Lucene Revolution, Lucene Eurocon, Berlin Buzzwords and various local meetups.

Agenda

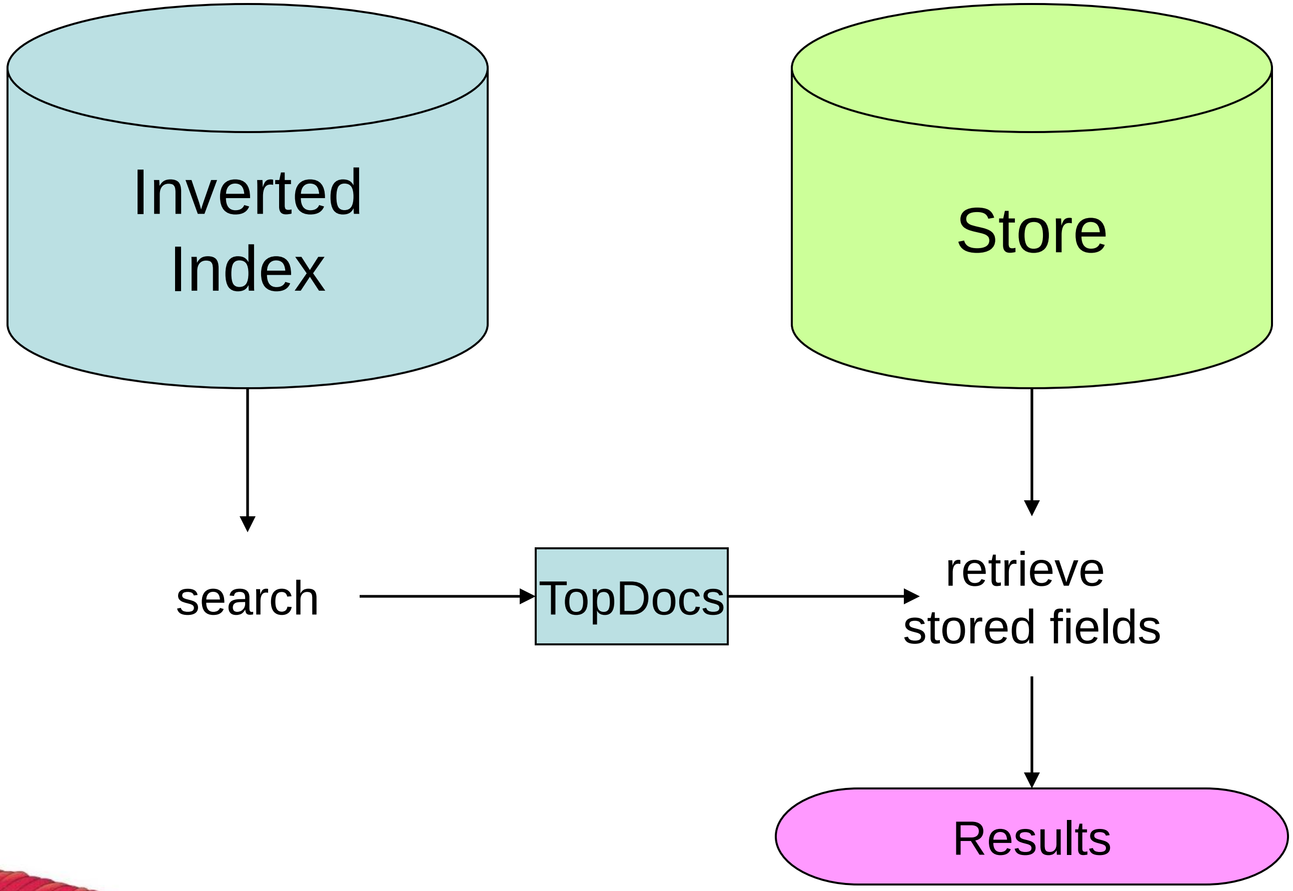
- Short introduction about Apache Lucene / Solr
- What happened? – *Chronology*
- Java 7 Crashes Eclipse – *or "The Porter Stemmer SIGSEGV Bug"*
- Loop Unwinding – *or "The Vint bug"*
- How to debug hotspot problems

Short introduction

About Apache Lucene Core

- **Apache Lucene Core** is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform.
- Supports text tokenization, inverted indexing and retrieval using the vector space model (VSM).
- Recently support for additional ranking models like *Okapi BM25* Model, *Amati and Rijsbergen‘s DFR*, *Clinchant and Gaussier's* Information-based models for IR, *Zhai and Lafferty's* language models.

Lucene’s data structures



Inverted Index

Store

search

TopDocs

retrieve stored fields

Results

c:\docs\einstein.txt:

The important thing is not to stop questioning.

c:\docs\shakespeare.txt:

To be or not to be.

Query: not

c:\docs\einstein.txt:

The important thing is not to stop questioning.

c:\docs\shakespeare.txt:

To be or not to be.

Query: not

c:\docs\einstein.txt:

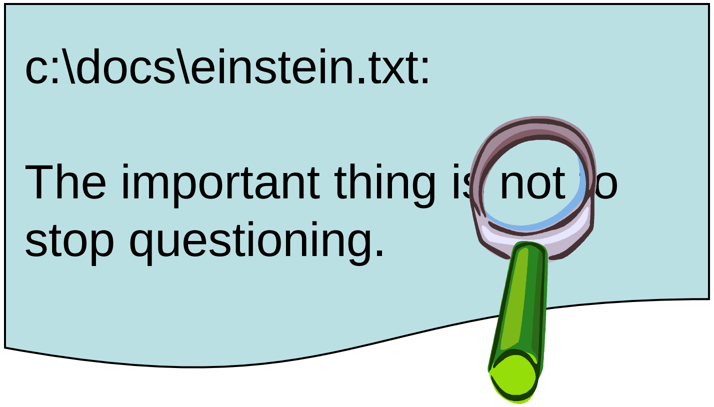
The important thing is not to stop questioning.

c:\docs\shakespeare.txt:

To be or not to be.

Query: not

String comparison slow!



c:\docs\einstein.txt:

The important thing is not to stop questioning.

c:\docs\shakespeare.txt:

To be or not to be.

Query: not

String comparison slow!

Solution: Inverted index

c:\docs\einstein.txt:

The important thing is not to stop questioning.

c:\docs\shakespeare.txt:

To be or not to be.

Inverted index

Query: not

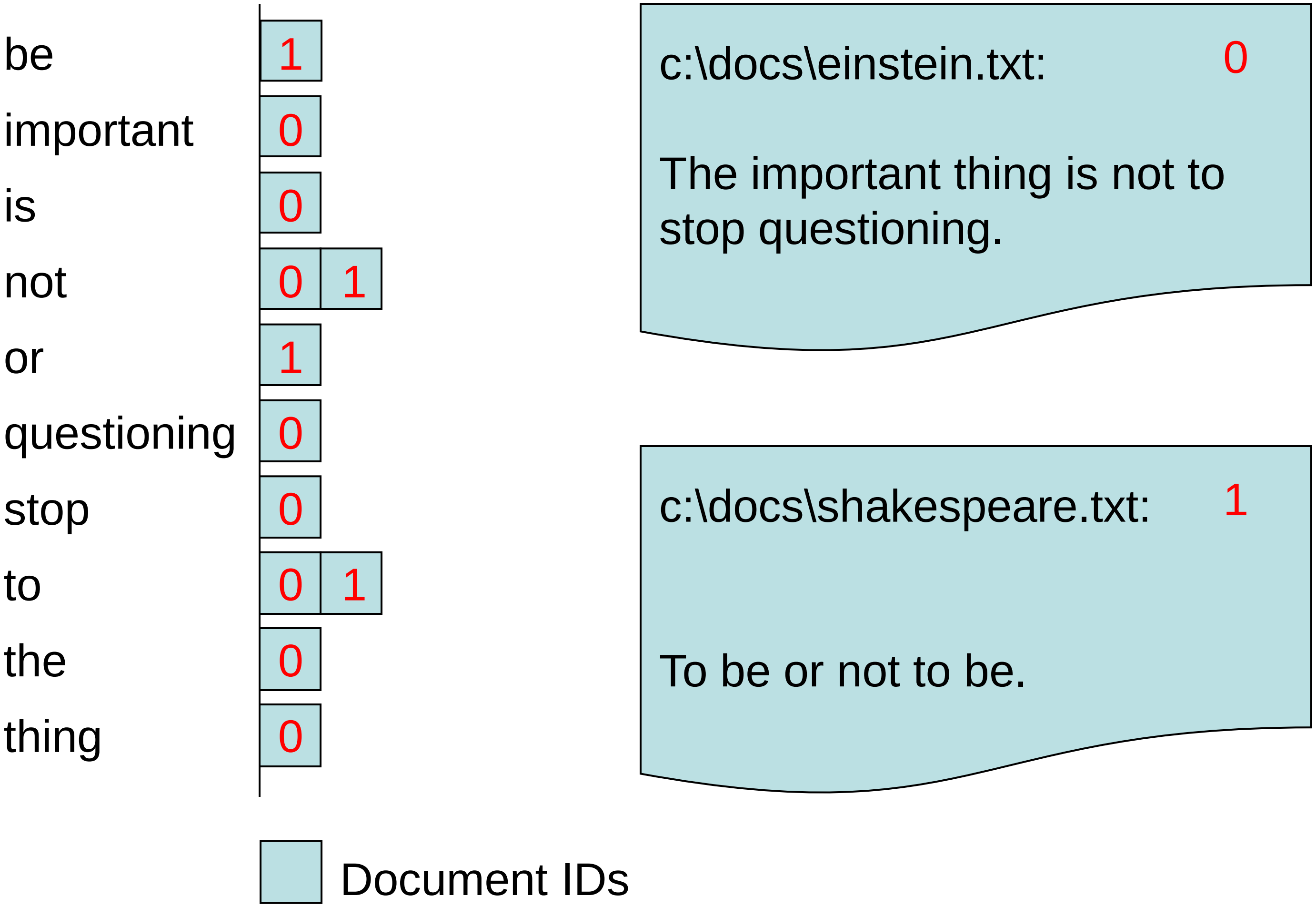
c:\docs\einstein.txt:

The important thing is not to stop questioning.

c:\docs\shakespeare.txt:

To be or not to be.

Inverted index



be	1	
important	0	
is	0	
not	0	1
or	1	
questioning	0	
stop	0	
to	0	1
the	0	
thing	0	

c:\docs\einstein.txt: 0

The important thing is not to stop questioning.

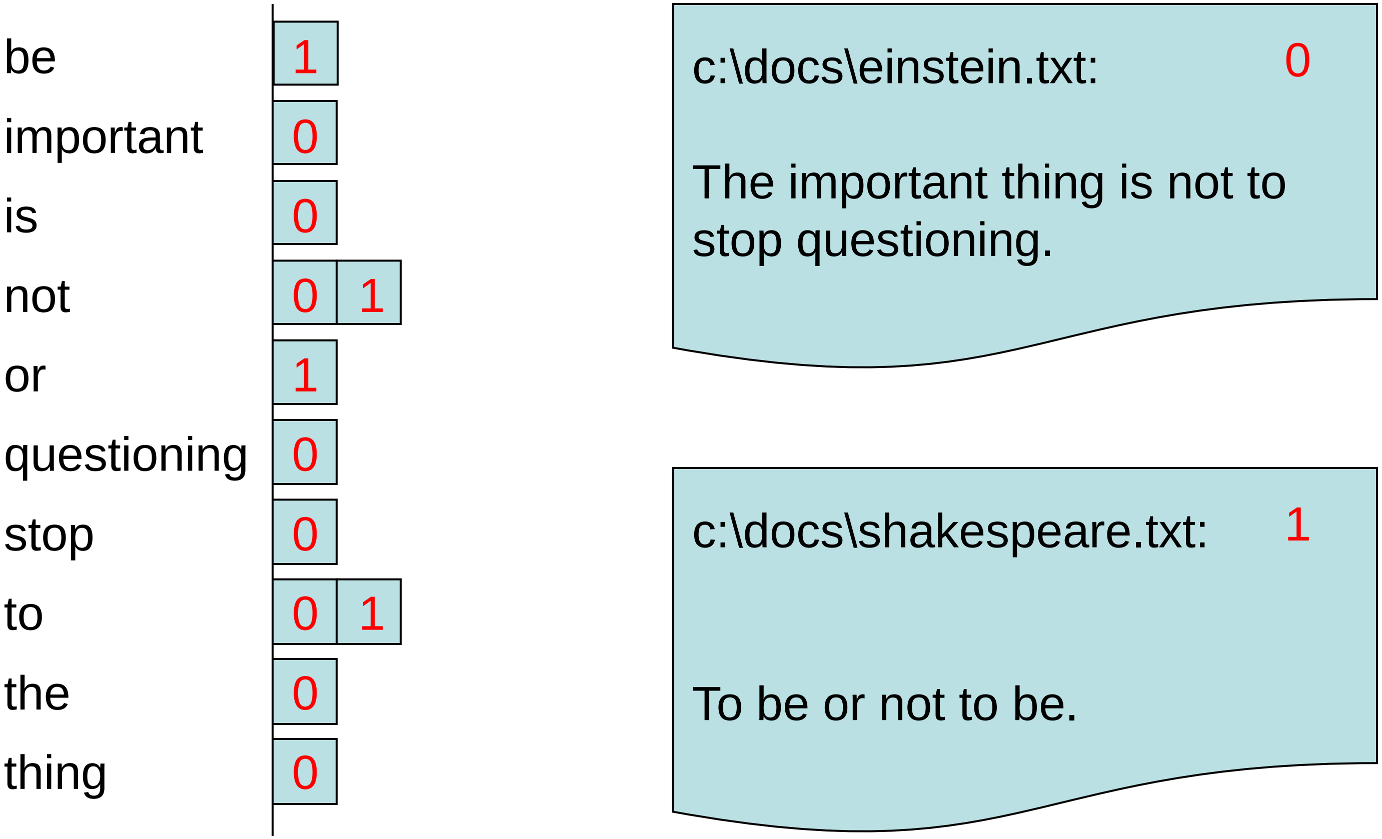
c:\docs\shakespeare.txt: 1

To be or not to be.

□ Document IDs

Inverted index

Query: not



be	1
important	0
is	0
not	0 1
or	1
questioning	0
stop	0
to	0 1
the	0
thing	0

c:\docs\einstein.txt: 0

The important thing is not to stop questioning.

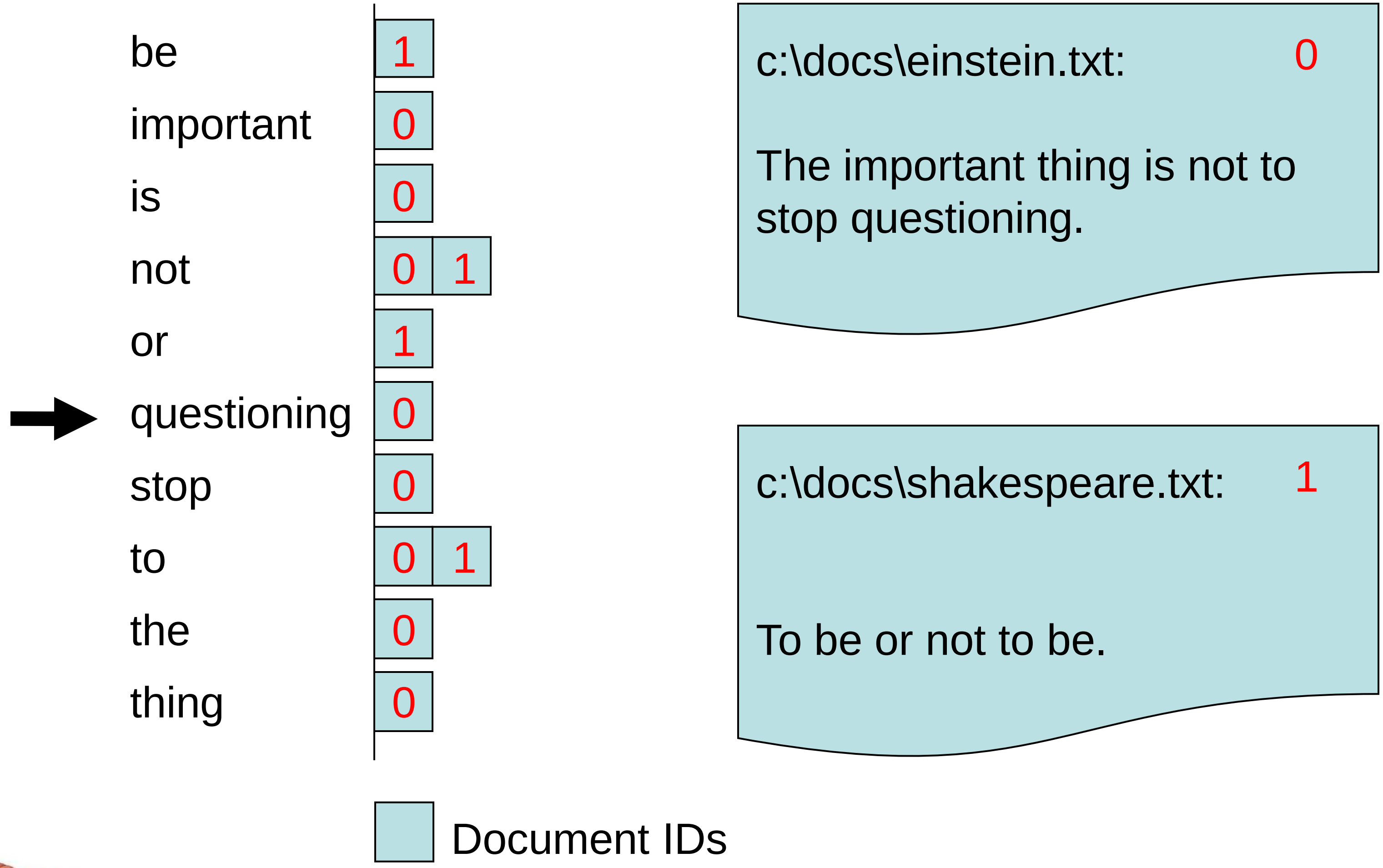
c:\docs\shakespeare.txt: 1

To be or not to be.

□ Document IDs

Inverted index

Query: not



	be	1	
	important	0	
	is	0	
	not	0	1
	or	1	
➡	questioning	0	
	stop	0	
	to	0	1
	the	0	
	thing	0	

c:\docs\einstein.txt: 0

The important thing is not to stop questioning.

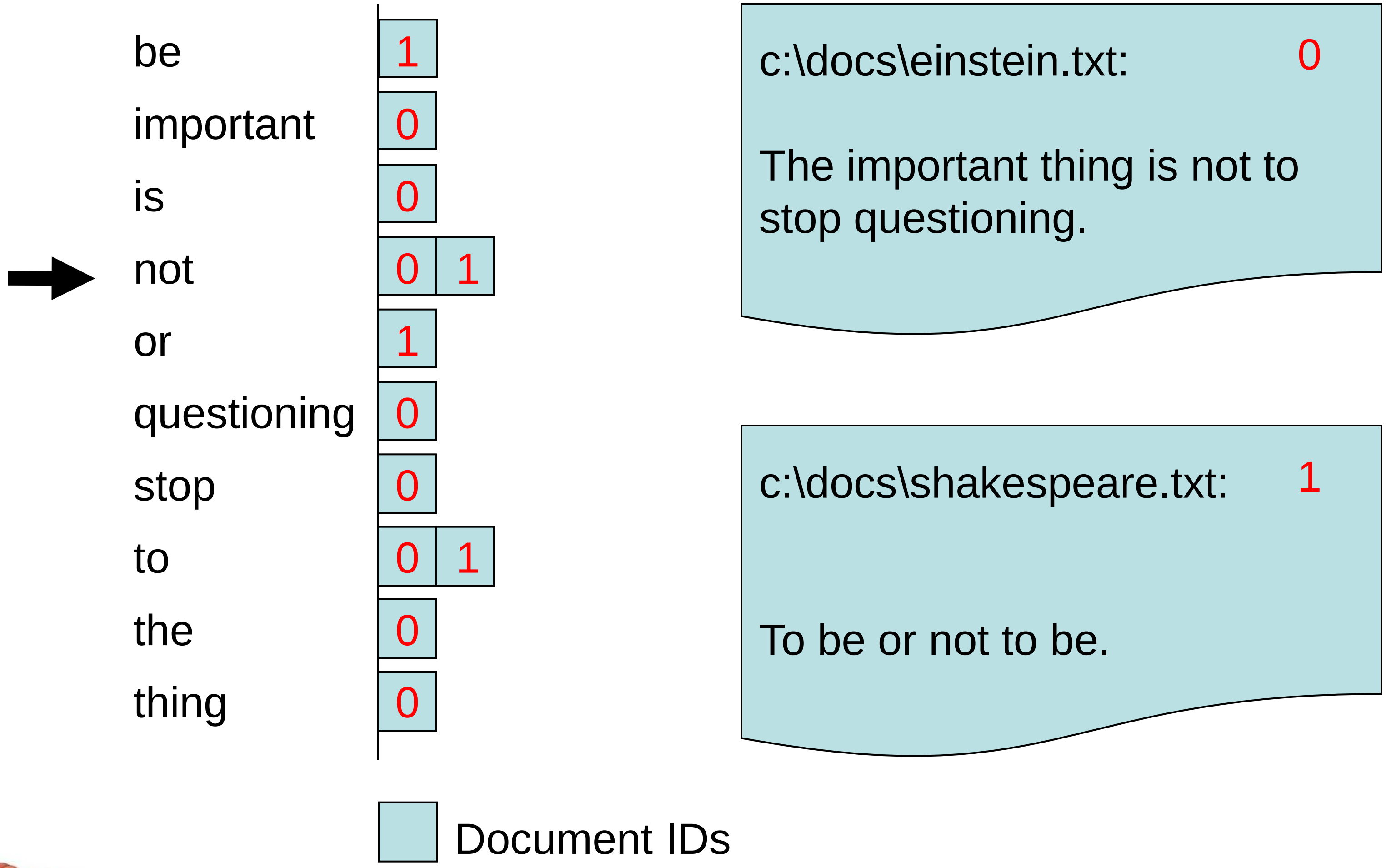
c:\docs\shakespeare.txt: 1

To be or not to be.

🟦 Document IDs

Inverted index

Query: not



be	1	
important	0	
is	0	
not	0	1
or	1	
questioning	0	
stop	0	
to	0	1
the	0	
thing	0	

c:\docs\einstein.txt: 0

The important thing is not to stop questioning.

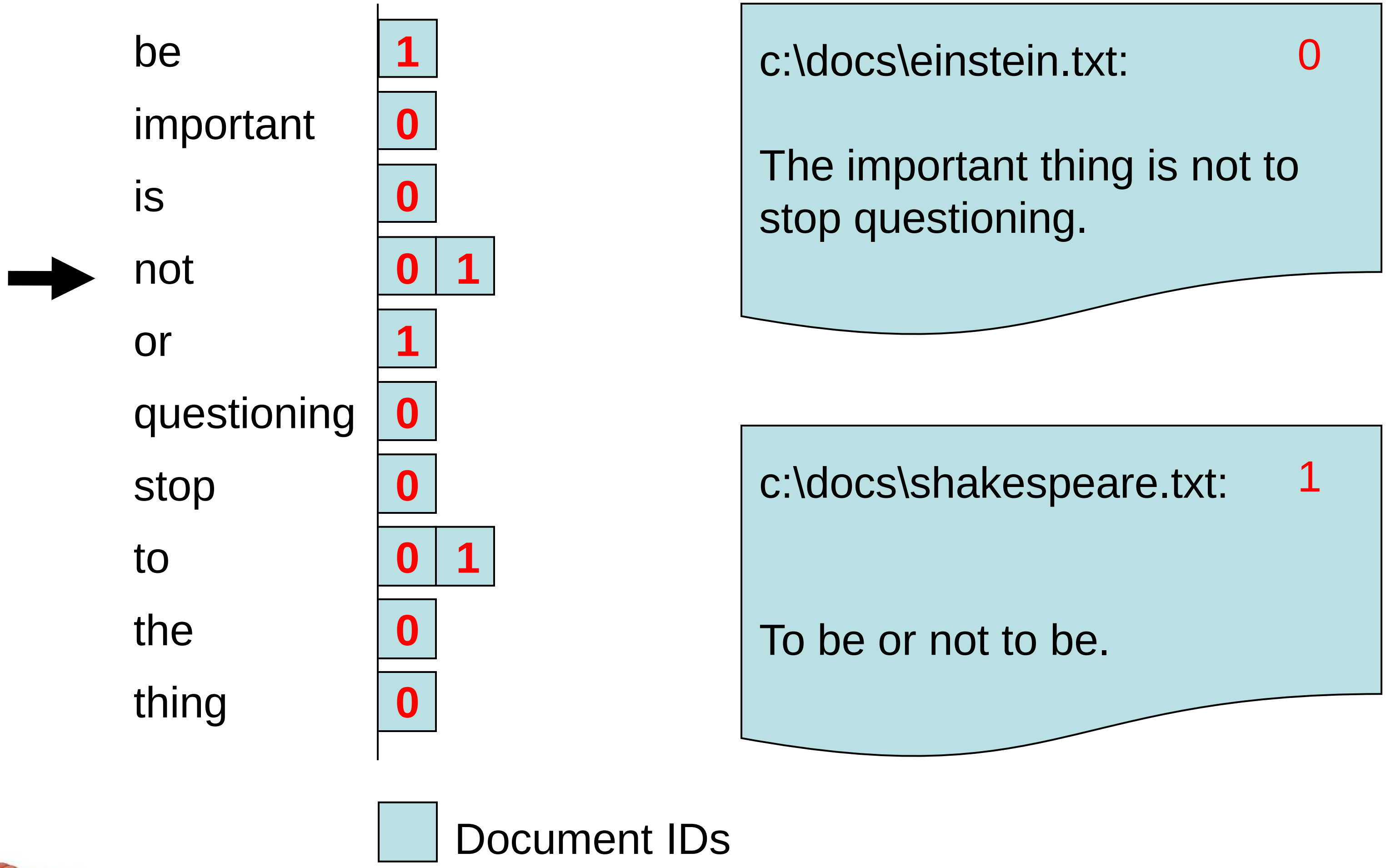
c:\docs\shakespeare.txt: 1

To be or not to be.

□ Document IDs

Inverted index

Query: not



	be	1	
	important	0	
	is	0	
→	not	0	1
	or	1	
	questioning	0	
	stop	0	
	to	0	1
	the	0	
	thing	0	

c:\docs\einstein.txt: 0

The important thing is not to stop questioning.

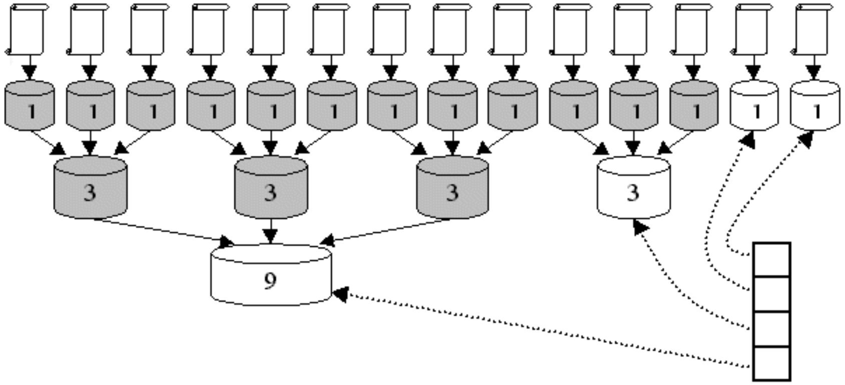
c:\docs\shakespeare.txt: 1

To be or not to be.

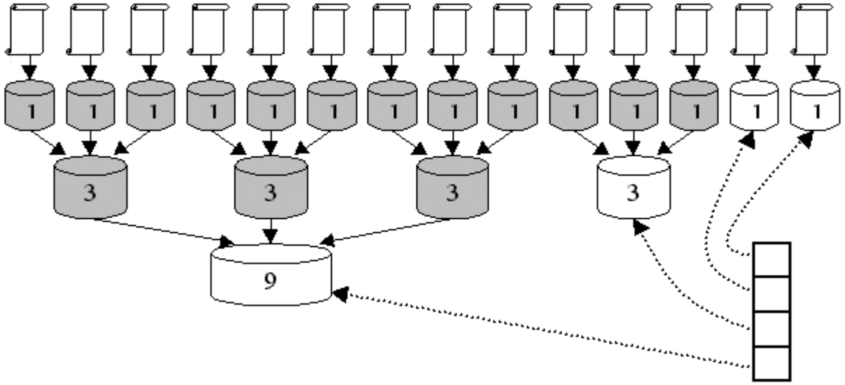
🟦 Document IDs

Segments in Lucene

- Each index consists of various segments placed in the index directory. All documents are added to new in-RAM segment files, merged to on-disk files after flushing.

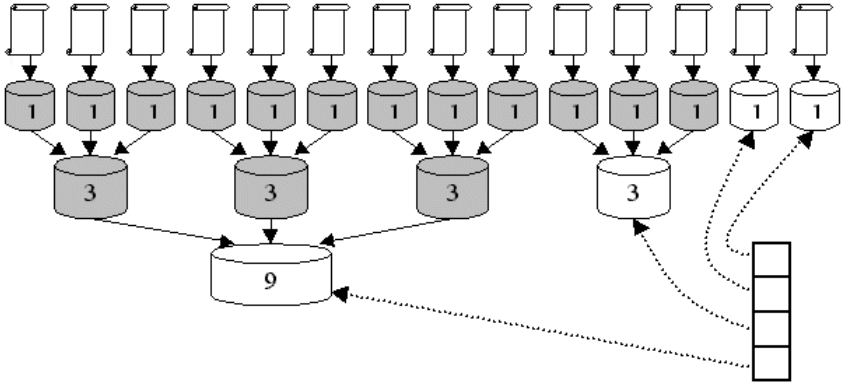


Segments in Lucene



- Each index consists of various segments placed in the index directory. All documents are added to new in-RAM segment files, merged to on-disk files after flushing.
- Lucene writes segments incrementally and then can merge them.

Segments in Lucene



- Each index consists of various segments placed in the index directory. All documents are added to new in-RAM segment files, merged to on-disk files after flushing.
- Lucene writes segments incrementally and then can merge them.
- Optimized index consists of one segment.

Algorithms in Apache Lucene

- Lot’s of performance-critical tight loops
- Heavy disk I/O code

- Most implementation code is hand-optimized, sometimes code duplication because of same code working on different datatypes

Apache Solr

- **Enterprise search server** based on Apache Lucene Core
- **REST API** with support for various input/output formats: **XML, JSON, CSV**
- Since version 3.1 **shares one source tree with Apache Lucene** 3.1 => same version numbers, closer integration of new features

What happened?

CHRONOLOGY

Chronology

- **Java 7 Release Candidate** released July 6, 2011 as build 147 *(compiled and signed on June 27, 2011 – also the release date of OpenJDK 7 b147)*
- **Saturday, July 23, 2011:**
  - downloaded it to do some testing with Lucene trunk, core test ran fine on my Windows 7 x64 box
  - Installation of FreeBSD package on Apache's Jenkins "Lucene" slave **=> heavy testing started: various crashes/failures:**

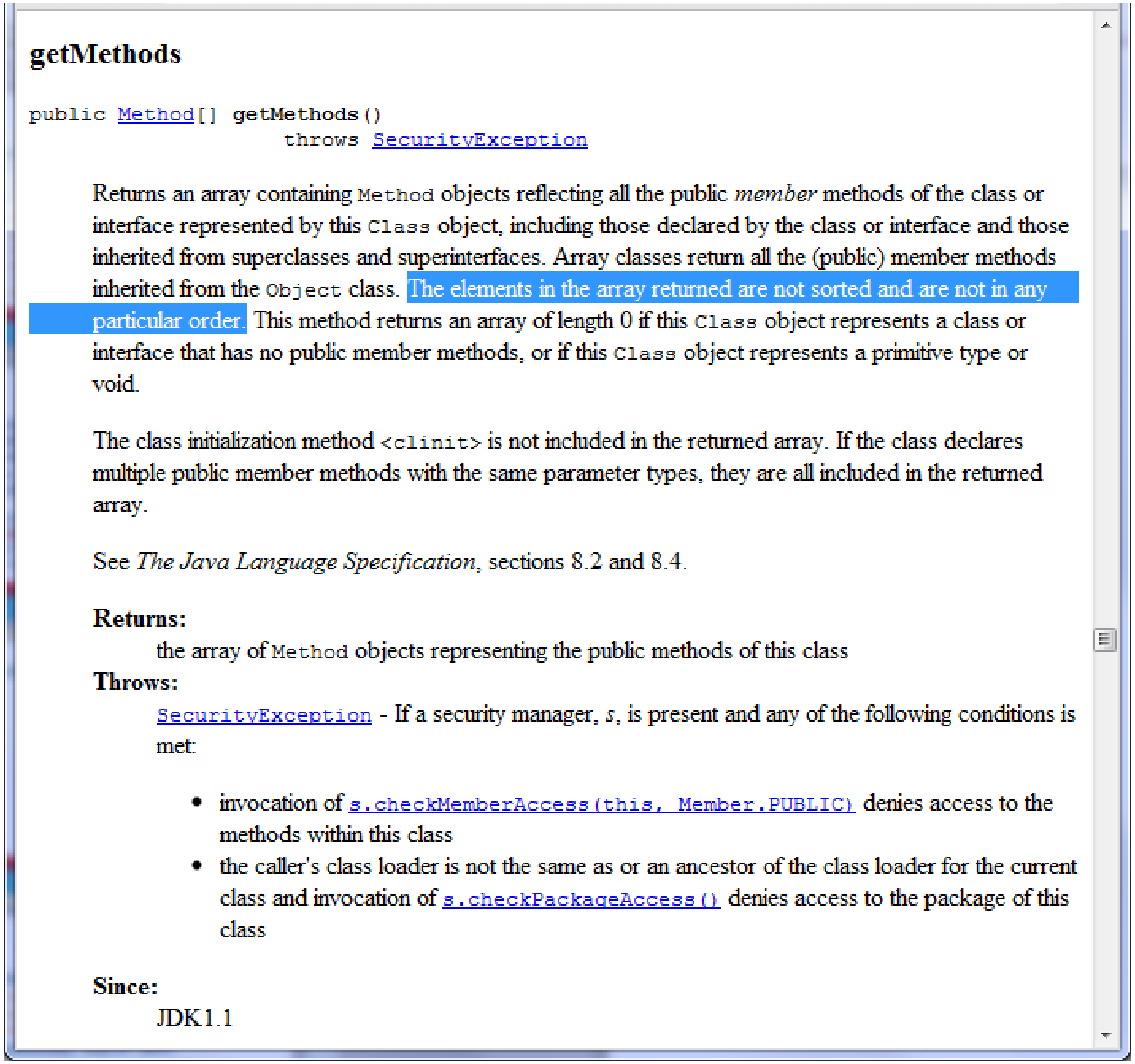
Issues found

- Jenkins reveals **SIGSEGV bug in Porter stemmer** (found when number of iterations were raised) [LUCENE-3335]
- New Lucene 3.4 facetting test sometimes **produces corrupt indexes** [LUCENE-3346]

- Small issue in **ICU** tests [LUCENE-3344, ICU bug #8734]
- Test of **WordDelimiterFilter** fails [simple fix committed]
- Lot’s of Solr tests suddenly fail [SOLR-2673]

Excurse: If your tests randomly fail with Java 7

- JUnit uses `Class.getMethods()` to find all tests in a class
- This list is not explicitly sorted in any order!
- Until Java 6 the methods were returned in same order as declared in source file!



getMethods

```
public Method[] getMethods()
                    throws SecurityException
```

Returns an array containing `Method` objects reflecting all the public *member* methods of the class or interface represented by this `Class` object, including those declared by the class or interface and those inherited from superclasses and superinterfaces. Array classes return all the (public) member methods inherited from the `Object` class. The elements in the array returned are not sorted and are not in any particular order. This method returns an array of length 0 if this `Class` object represents a class or interface that has no public member methods, or if this `Class` object represents a primitive type or void.

The class initialization method `<clinit>` is not included in the returned array. If the class declares multiple public member methods with the same parameter types, they are all included in the returned array.

See *The Java Language Specification*, sections 8.2 and 8.4.

Returns:
the array of `Method` objects representing the public methods of this class

Throws:
`SecurityException` - If a security manager, *s*, is present and any of the following conditions is met:

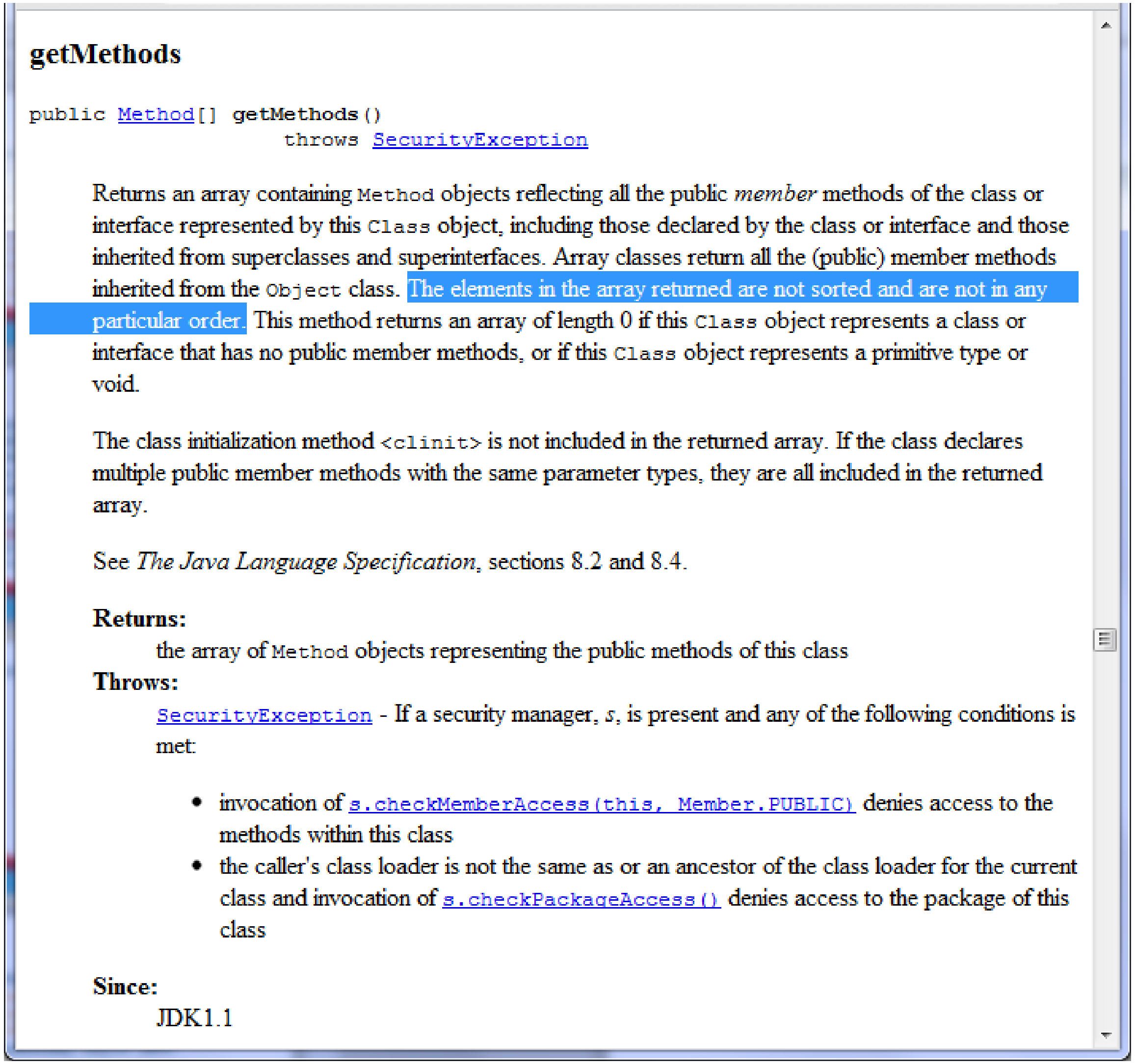
- invocation of `s.checkMemberAccess(this, Member.PUBLIC)` denies access to the methods within this class
- the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of `s.checkPackageAccess()` denies access to the package of this class

Since:
JDK1.1

Excurse: If your tests randomly fail with Java 7

- JUnit uses `Class.getMethods()` to find all tests in a class
- This list is not explicitly sorted in any order!
- Until Java 6 the methods were returned in same order as declared in source file!

Repair your tests to not rely on execution order of `@Test` methods!



getMethods

```
public Method[] getMethods()
                    throws SecurityException
```

Returns an array containing `Method` objects reflecting all the public *member* methods of the class or interface represented by this `Class` object, including those declared by the class or interface and those inherited from superclasses and superinterfaces. Array classes return all the (public) member methods inherited from the `Object` class. The elements in the array returned are not sorted and are not in any particular order. This method returns an array of length 0 if this `Class` object represents a class or interface that has no public member methods, or if this `Class` object represents a primitive type or void.

The class initialization method `<clinit>` is not included in the returned array. If the class declares multiple public member methods with the same parameter types, they are all included in the returned array.

See *The Java Language Specification*, sections 8.2 and 8.4.

Returns:

the array of `Method` objects representing the public methods of this class

Throws:

`SecurityException` - If a security manager, *s*, is present and any of the following conditions is met:

- invocation of `s.checkMemberAccess(this, Member.PUBLIC)` denies access to the methods within this class
- the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of `s.checkPackageAccess()` denies access to the package of this class

Since:

JDK1.1

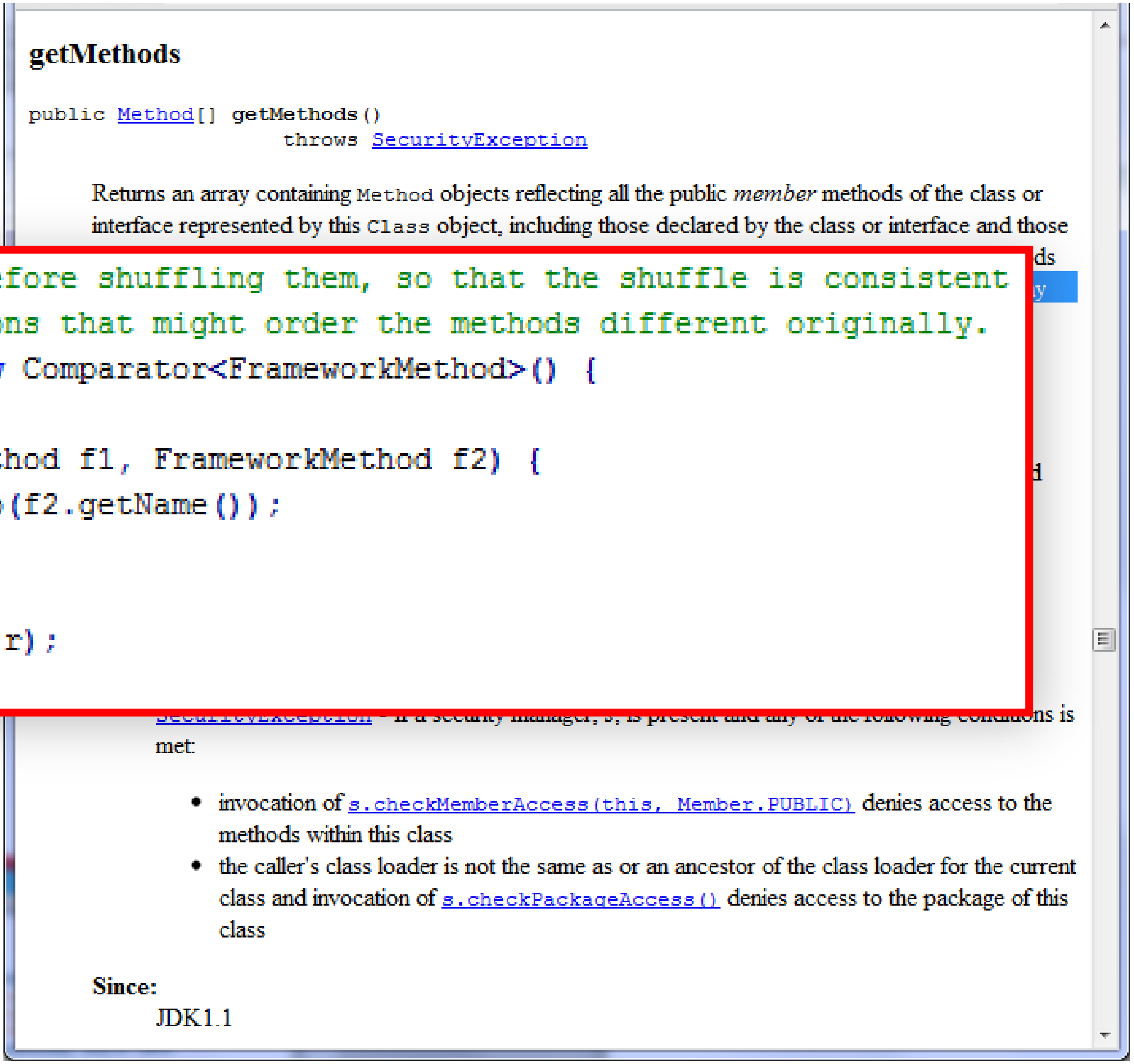
Excurse: If your tests randomly fail with Java 7

- JUnit uses `Class.getMethods()` to find all tests in a class
- This list is not explicitly
-

```
// sort the test methods first before shuffling them, so that the shuffle is consistent
// across different implementations that might order the methods different originally.
Collections.sort(testMethods, new Comparator<FrameworkMethod>() {
  @Override
  public int compare(FrameworkMethod f1, FrameworkMethod f2) {
    return f1.getName().compareTo(f2.getName());
  }
});
Collections.shuffle(testMethods, r);
return testMethods;
```

R

rely on execution order of `@Test` methods!



getMethods

```
public Method[] getMethods()
                    throws SecurityException
```

Returns an array containing `Method` objects reflecting all the public *member* methods of the class or interface represented by this `Class` object, including those declared by the class or interface and those

ds

y

d

`SecurityException` - If a security manager, s, is present and any of the following conditions is met:

- invocation of `s.checkMemberAccess(this, Member.PUBLIC)` denies access to the methods within this class
- the caller's class loader is not the same as or an ancestor of the class loader for the current class and invocation of `s.checkPackageAccess()` denies access to the package of this class

Since:
JDK1.1

Chronology

- **Saturday, July 23, 2011:**
  - Porter Stemmer SIGSEGV failure reported as bug at Oracles bug tracker, bug invisible to public [#7070134]
- **Wednesday, July 27, 2011:**
  - Analyzed index corruption bugs
  - Those were already reported to Oracle before [#7044738, #7068051]

Chronology

- **Monday, July 25, 2011:**
  - Directly **contacted the hotspot developers** on the **OpenJDK** mailing list, they confirmed the bug about Porter Stemmer and supplied a patch *(thanks to Vladimir Kozlov)*
  - Applied patch to OpenJDK installation on Apache Jenkins server *(patched FreeBSD package)*
  - **All bugs fixed!**
  - Patch had fix for 3 bugs, which showed us the related bug numbers => fixed the corruption issues

WARNING !!!

- **Also Java 6 affected!**
  (some time after the only stable version 1.6.0_18)
- Optimizations disabled by default, so:

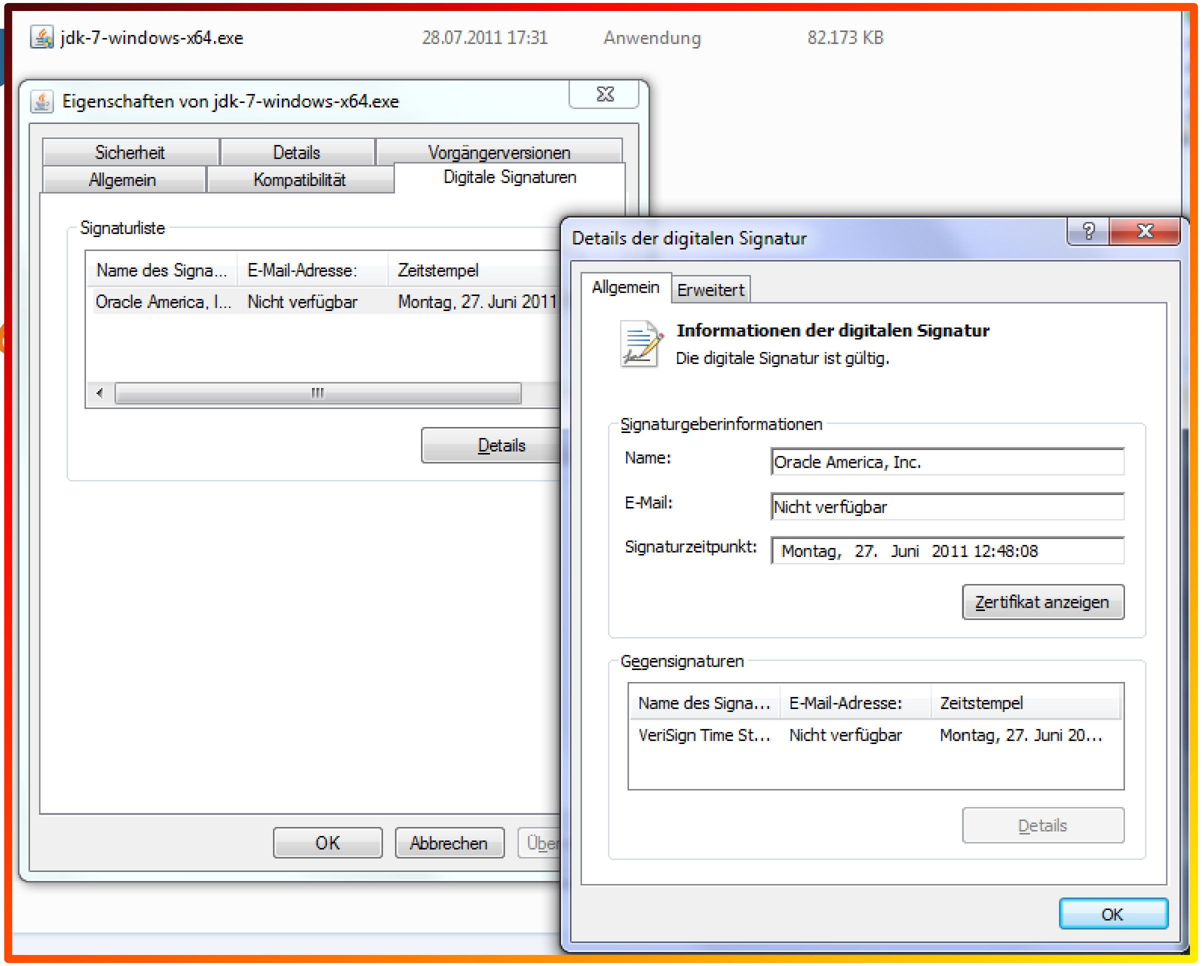
Don’t use `–XX:+AggressiveOpts`
if you want your loops behave correctly!

Chronology

- **Thursday, July 28, 2011:**
  - Oracle released JDK 7 to public
  - Package was identical to release candidate *(Windows EXE signature dated June 27, 2011)*

Chronology

- Thursday, J
  - Oracle releas
  - Package was *EXE signatur*



jdk-7-windows-x64.exe 28.07.2011 17:31 Anwendung 82.173 KB

Eigenschaften von jdk-7-windows-x64.exe

Name des Signa...	E-Mail-Adresse:	Zeitstempel
Oracle America, I...	Nicht verfügbar	Montag, 27. Juni 2011

Details der digitalen Signatur

Informationen der digitalen Signatur

Die digitale Signatur ist gültig.

Signaturgeberinformationen

Name: Oracle America, Inc.

E-Mail: Nicht verfügbar

Signaturzeitpunkt: Montag, 27. Juni 2011 12:48:08

Gegensignaturen

Name des Signa...	E-Mail-Adresse:	Zeitstempel
VeriSign Time St...	Nicht verfügbar	Montag, 27. Juni 20...

Chronology

- **Thursday, July 28, 2011:**
  - Oracle released JDK 7 to public
  - Package was identical to release candidate *(Windows EXE signature dated June 27, 2011)*

- Apache Lucene PMC decided to warn users on web page and announce@apache.org mailing list

The warning

Oracle released Java 7 today. Unfortunately it contains hotspot compiler optimizations, which miscompile some loops. This can affect code of several Apache projects. Sometimes JVMs only crash, but in several cases, results calculated can be incorrect, leading to bugs in applications (see Hotspot bugs 7070134, 7044738, 7068051).

Apache Lucene Core and **Apache Solr** are two Apache projects, which are affected by these bugs, namely all versions released until today. Solr users with the default configuration will have Java crashing with SIGSEGV as soon as they start to index documents, as one affected part is the well-known Porter stemmer (see LUCENE-3335). Other loops in Lucene may be miscompiled, too, leading to index corruption (especially on Lucene trunk with pulsing codec; other loops may be affected, too - LUCENE-3346).

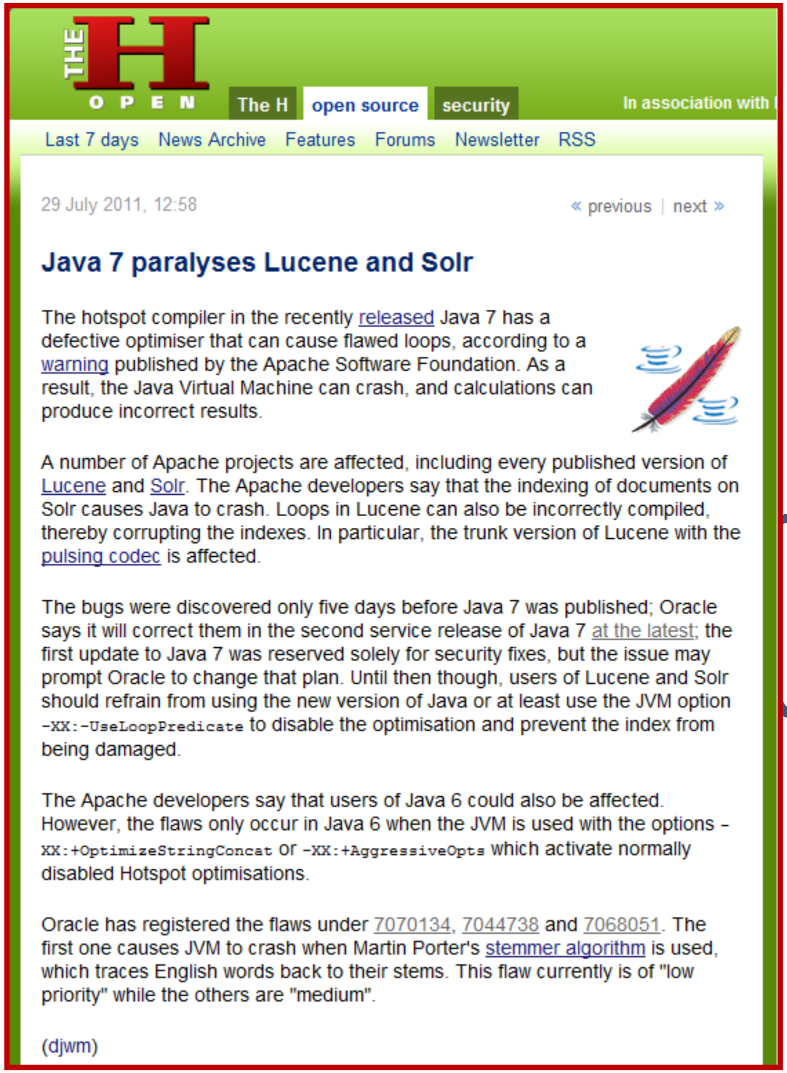
These problems were detected only 5 days before the official Java 7 release, so Oracle had no time to fix those bugs, affecting also many more applications. In response to our questions, they proposed to include the fixes into service release u2 (eventually into service release u1, see this mail). **This means you cannot use Apache Lucene/Solr with Java 7 releases before Update 2!** If you do, please don't open bug reports, it is not the committers' fault! At least disable loop optimizations using the `-XX:-UseLoopPredicate` JVM option to not risk index corruptions.

Please note: Also Java 6 users are affected, if they use one of those JVM options, which are **not** enabled by default: `-XX:+OptimizeStringConcat` or `-XX:+AggressiveOpts`.

It is strongly recommended not to use any hotspot optimization switches in any Java version without extensive testing!

In case you upgrade to Java 7, remember that you may have to reindex, as the unicode version shipped with Java 7 changed and tokenization behaves differently (e.g. lowercasing). For more information, read `JRE_VERSION_MIGRATION.txt` in your distribution package!

Chronology:
Friday, July 29, 2011



29 July 2011, 12:58

Java 7 paralyses Lucene and Solr

The hotspot compiler in the recently released Java 7 has a defective optimiser that can cause flawed loops, according to a warning published by the Apache Software Foundation. As a result, the Java Virtual Machine can crash, and calculations can produce incorrect results.

A number of Apache projects are affected, including every published version of Lucene and Solr. The Apache developers say that the indexing of documents on Solr causes Java to crash. Loops in Lucene can also be incorrectly compiled, thereby corrupting the indexes. In particular, the trunk version of Lucene with the pulsing codec is affected.

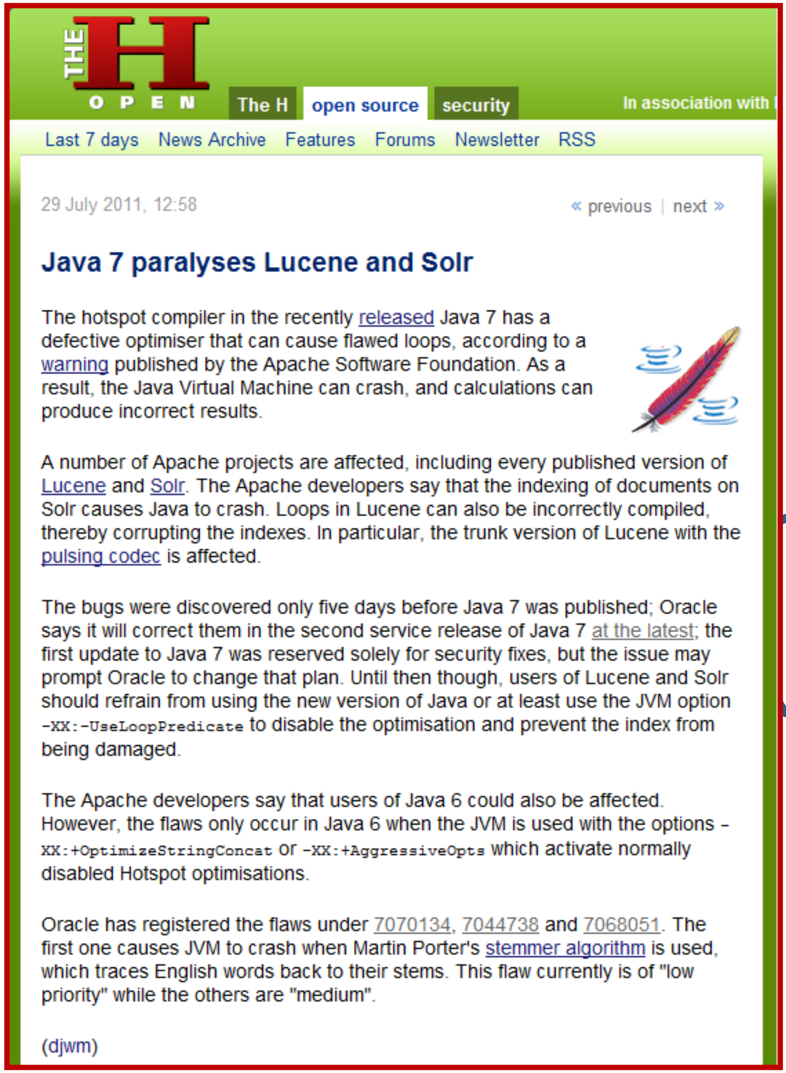
The bugs were discovered only five days before Java 7 was published; Oracle says it will correct them in the second service release of Java 7 at the latest; the first update to Java 7 was reserved solely for security fixes, but the issue may prompt Oracle to change that plan. Until then though, users of Lucene and Solr should refrain from using the new version of Java or at least use the JVM option `-XX:-UseLoopPredicate` to disable the optimisation and prevent the index from being damaged.

The Apache developers say that users of Java 6 could also be affected. However, the flaws only occur in Java 6 when the JVM is used with the options `-XX:+OptimizeStringConcat` or `-XX:+AggressiveOpts` which activate normally disabled Hotspot optimisations.

Oracle has registered the flaws under 7070134, 7044738 and 7068051. The first one causes JVM to crash when Martin Porter's stemmer algorithm is used, which traces English words back to their stems. This flaw currently is of "low priority" while the others are "medium".

(djwm)

ronology:
July 29, 2011



29 July 2011, 12:58

« previous | next »

Java 7 paralyses Lucene and Solr

The hotspot compiler in the recently released Java 7 has a defective optimiser that can cause flawed loops, according to a warning published by the Apache Software Foundation. As a result, the Java Virtual Machine can crash, and calculations can produce incorrect results.

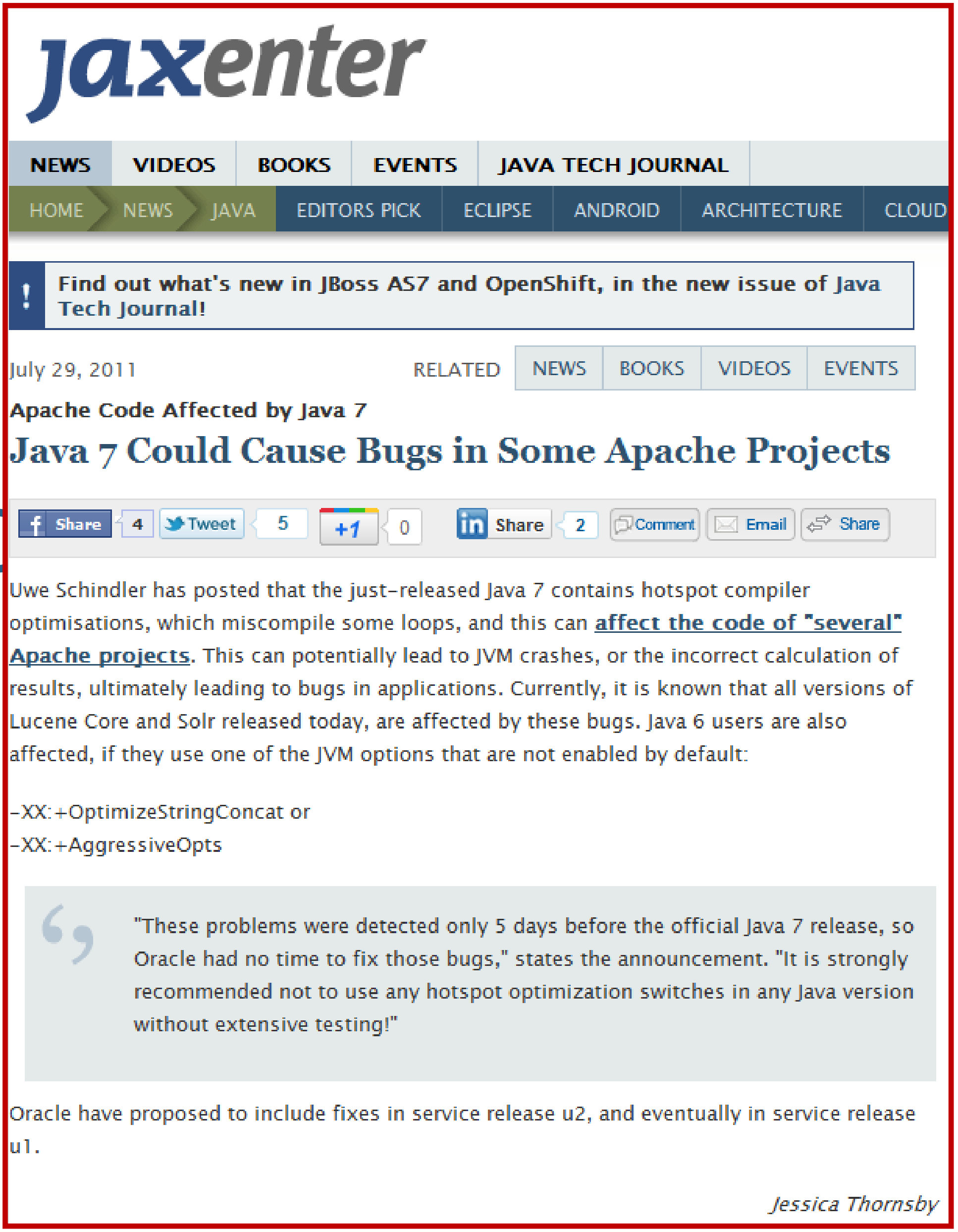
A number of Apache projects are affected, including every published version of Lucene and Solr. The Apache developers say that the indexing of documents on Solr causes Java to crash. Loops in Lucene can also be incorrectly compiled, thereby corrupting the indexes. In particular, the trunk version of Lucene with the pulsing codec is affected.

The bugs were discovered only five days before Java 7 was published; Oracle says it will correct them in the second service release of Java 7 at the latest; the first update to Java 7 was reserved solely for security fixes, but the issue may prompt Oracle to change that plan. Until then though, users of Lucene and Solr should refrain from using the new version of Java or at least use the JVM option `-XX:-UseLoopPredicate` to disable the optimisation and prevent the index from being damaged.

The Apache developers say that users of Java 6 could also be affected. However, the flaws only occur in Java 6 when the JVM is used with the options `-XX:+OptimizeStringConcat` or `-XX:+AggressiveOpts` which activate normally disabled Hotspot optimisations.

Oracle has registered the flaws under 7070134, 7044738 and 7068051. The first one causes JVM to crash when Martin Porter's stemmer algorithm is used, which traces English words back to their stems. This flaw currently is of "low priority" while the others are "medium".

(djwm)



! Find out what's new in JBoss AS7 and OpenShift, in the new issue of Java Tech Journal!

July 29, 2011

Apache Code Affected by Java 7

Java 7 Could Cause Bugs in Some Apache Projects

Uwe Schindler has posted that the just-released Java 7 contains hotspot compiler optimisations, which miscompile some loops, and this can **affect the code of "several" Apache projects**. This can potentially lead to JVM crashes, or the incorrect calculation of results, ultimately leading to bugs in applications. Currently, it is known that all versions of Lucene Core and Solr released today, are affected by these bugs. Java 6 users are also affected, if they use one of the JVM options that are not enabled by default:

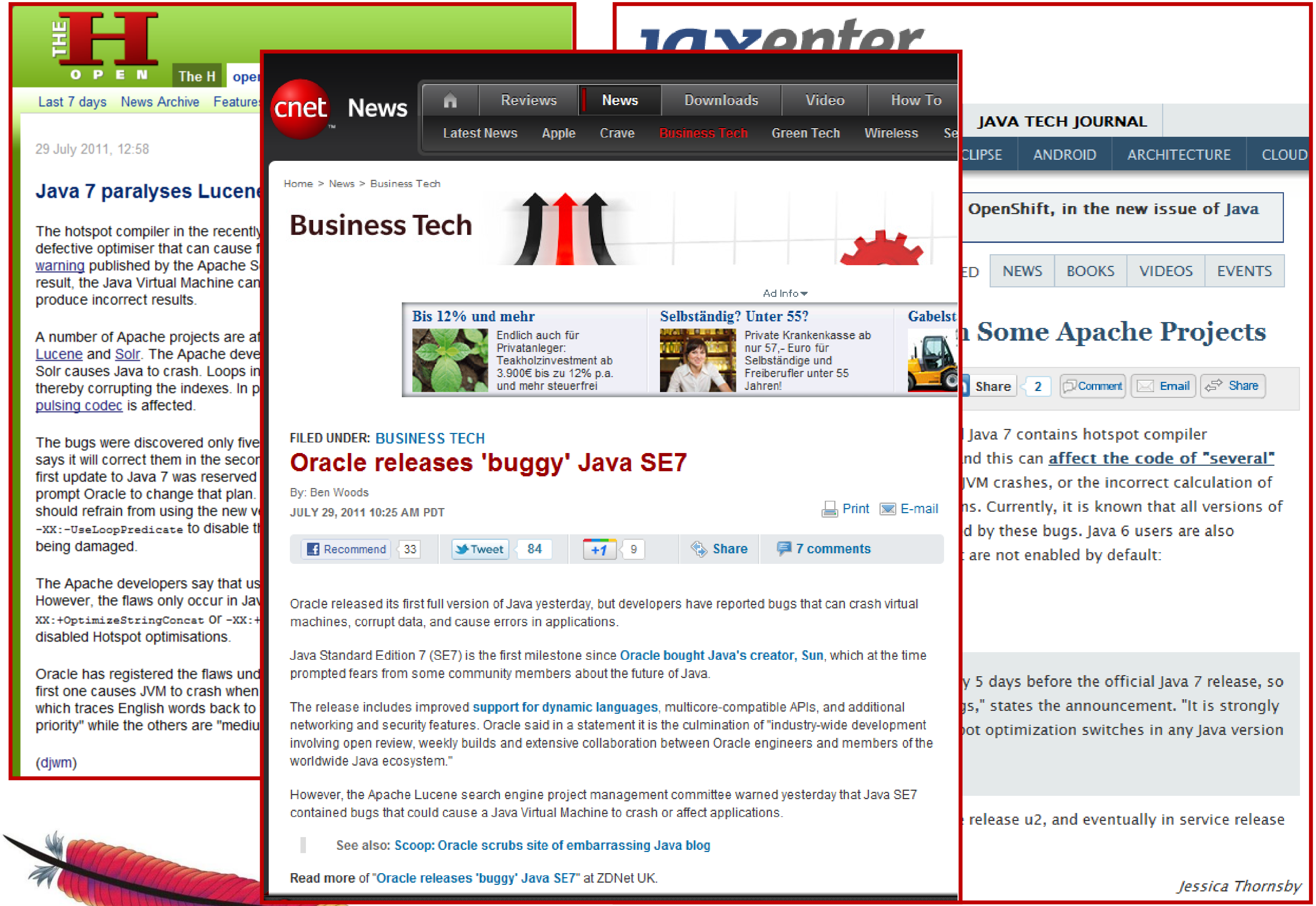
-XX:+OptimizeStringConcat or

-XX:+AggressiveOpts

> "These problems were detected only 5 days before the official Java 7 release, so Oracle had no time to fix those bugs," states the announcement. "It is strongly recommended not to use any hotspot optimization switches in any Java version without extensive testing!"

Oracle have proposed to include fixes in service release u2, and eventually in service release u1.

Jessica Thornsby



29 July 2011, 12:58

Java 7 paralyses Lucene

The hotspot compiler in the recently
defective optimiser that can cause f
warning published by the Apache S
result, the Java Virtual Machine can
produce incorrect results.

A number of Apache projects are af
Lucene and Solr. The Apache deve
Solr causes Java to crash. Loops in
thereby corrupting the indexes. In p
pulsing codec is affected.

The bugs were discovered only five
says it will correct them in the secon
first update to Java 7 was reserved
prompt Oracle to change that plan.
should refrain from using the new v
`-XX:-UseLoopPredicate` to disable th
being damaged.

The Apache developers say that us
However, the flaws only occur in Jav
`XX:+OptimizeStringConcat` or `-XX:+`
disabled Hotspot optimisations.

Oracle has registered the flaws und
first one causes JVM to crash when
which traces English words back to
priority" while the others are "mediu

(djwm)

Business Tech

FILED UNDER: BUSINESS TECH

Oracle releases 'buggy' Java SE7

By: Ben Woods

JULY 29, 2011 10:25 AM PDT

Oracle released its first full version of Java yesterday, but developers have reported bugs that can crash virtual machines, corrupt data, and cause errors in applications.

Java Standard Edition 7 (SE7) is the first milestone since Oracle bought Java's creator, Sun, which at the time prompted fears from some community members about the future of Java.

The release includes improved support for dynamic languages, multicore-compatible APIs, and additional networking and security features. Oracle said in a statement it is the culmination of "industry-wide development involving open review, weekly builds and extensive collaboration between Oracle engineers and members of the worldwide Java ecosystem."

However, the Apache Lucene search engine project management committee warned yesterday that Java SE7 contained bugs that could cause a Java Virtual Machine to crash or affect applications.

See also: Scoop: Oracle scrubs site of embarrassing Java blog

Read more of "Oracle releases 'buggy' Java SE7" at ZDNet UK.

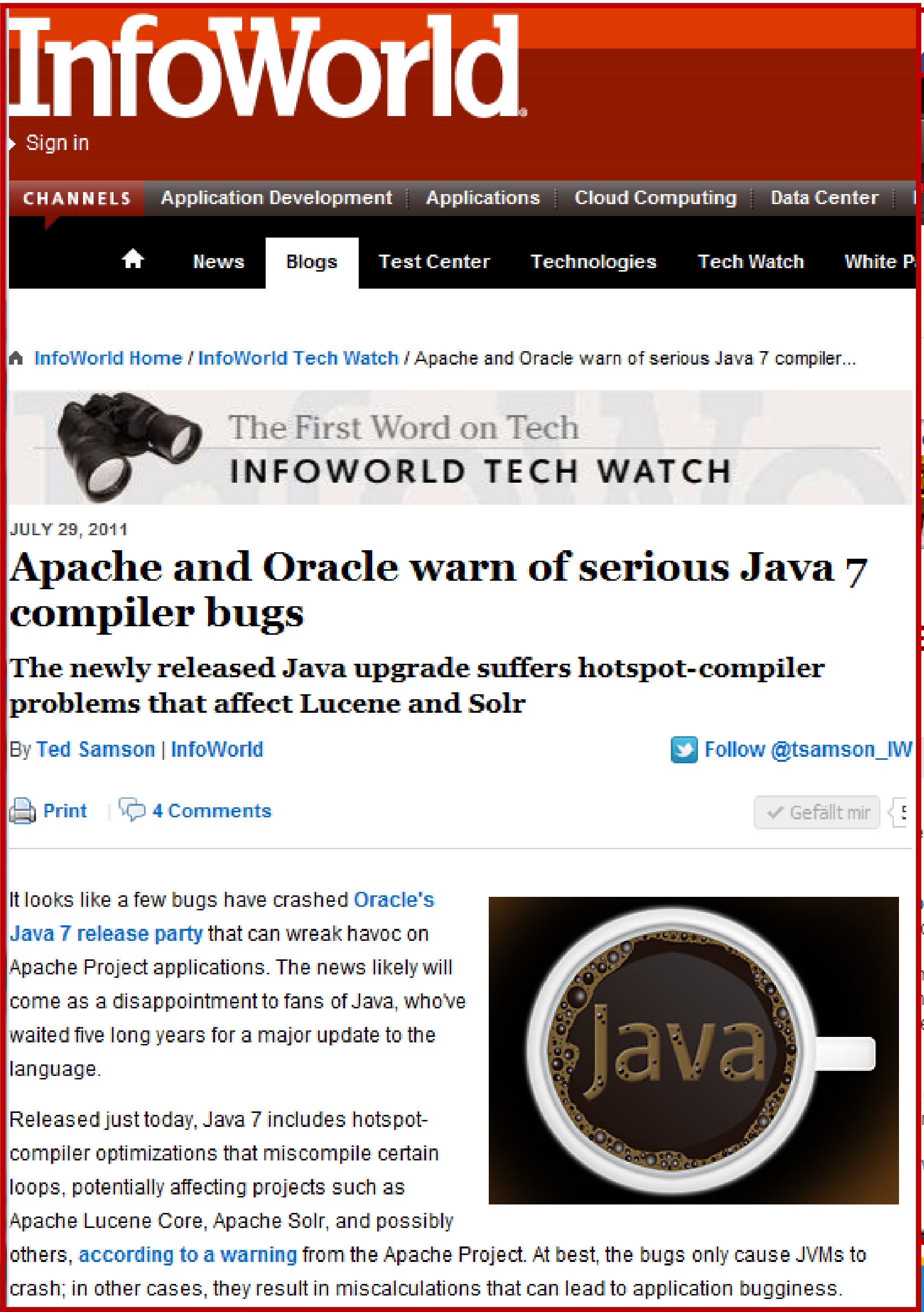
n Some Apache Projects

Java 7 contains hotspot compiler
nd this can affect the code of "several"
JVM crashes, or the incorrect calculation of
ns. Currently, it is known that all versions of
d by these bugs. Java 6 users are also
are not enabled by default:

y 5 days before the official Java 7 release, so
s," states the announcement. "It is strongly
ot optimization switches in any Java version

release u2, and eventually in service release

Jessica Thornsby



InfoWorld Home / InfoWorld Tech Watch / Apache and Oracle warn of serious Java 7 compiler...

JULY 29, 2011

Apache and Oracle warn of serious Java 7 compiler bugs

The newly released Java upgrade suffers hotspot-compiler problems that affect Lucene and Solr

By Ted Samson | InfoWorld

Follow @tsamson_IW

Print | 4 Comments

Gefällt mir

It looks like a few bugs have crashed Oracle's Java 7 release party that can wreak havoc on Apache Project applications. The news likely will come as a disappointment to fans of Java, who've waited five long years for a major update to the language.

Released just today, Java 7 includes hotspot-compiler optimizations that miscompile certain loops, potentially affecting projects such as Apache Lucene Core, Apache Solr, and possibly others, according to a warning from the Apache Project. At best, the bugs only cause JVMs to crash; in other cases, they result in miscalculations that can lead to application bugginess.

E7

Print E-mail

Share 7 comments

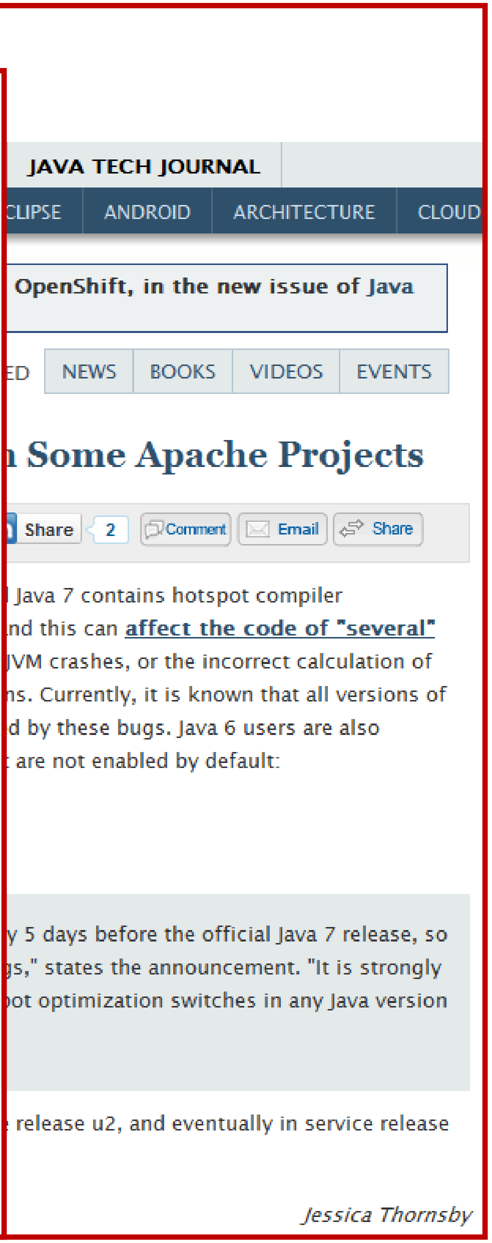
rs have reported bugs that can crash virtual

ought Java's creator, Sun, which at the time of Java.

multicore-compatible APIs, and additional e culmination of "industry-wide development etween Oracle engineers and members of the

committee warned yesterday that Java SE7 r affect applications.

va blog



OpenShift, in the new issue of Java

n Some Apache Projects

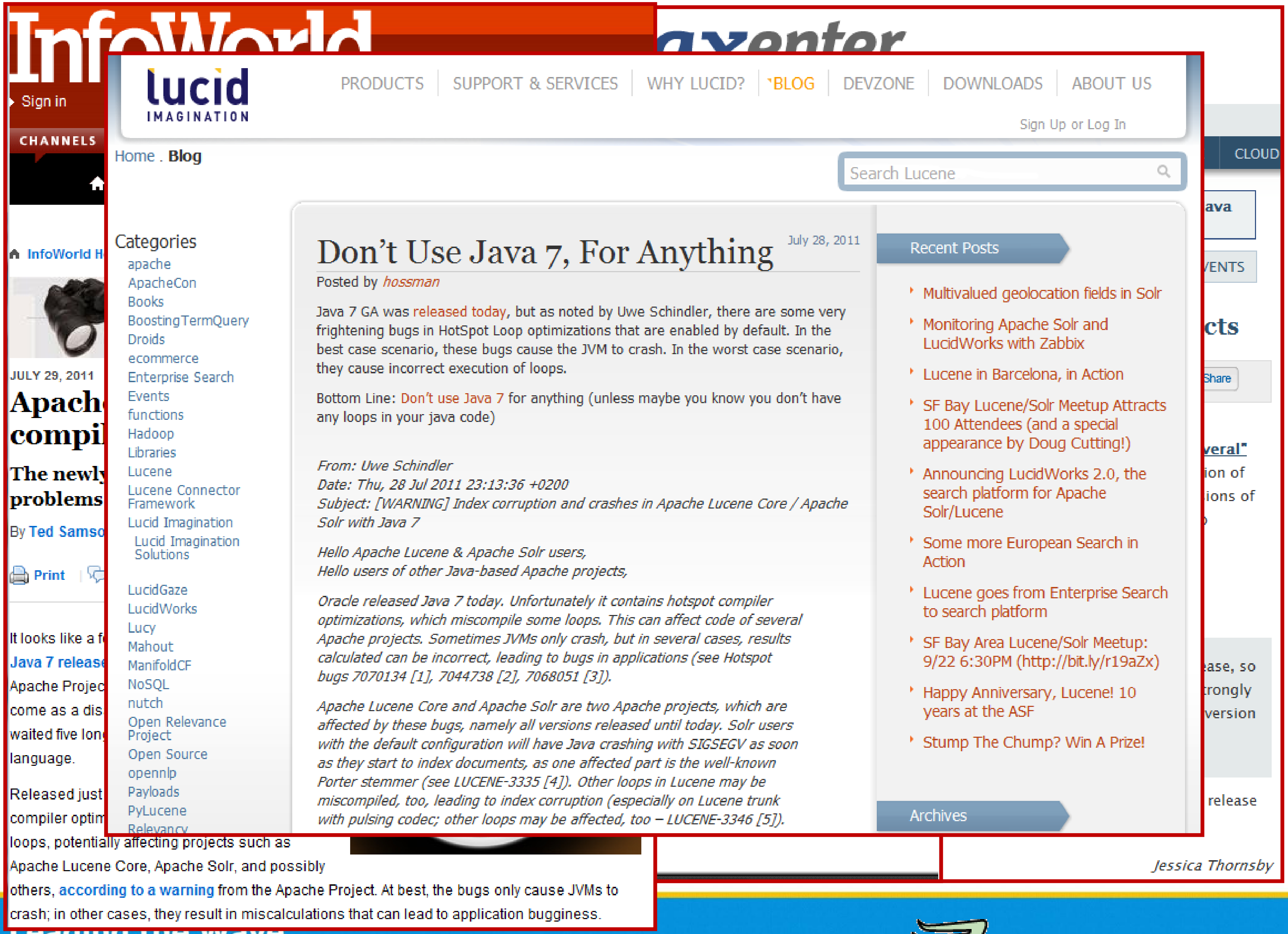
Share 2 Comment Email Share

Java 7 contains hotspot compiler nd this can affect the code of "several" JVM crashes, or the incorrect calculation of ns. Currently, it is known that all versions of d by these bugs. Java 6 users are also are not enabled by default:

y 5 days before the official Java 7 release, so gs," states the announcement. "It is strongly pot optimization switches in any Java version

release u2, and eventually in service release

Jessica Thornsby



Categories

- apache
- ApacheCon
- Books
- BoostingTermQuery
- Droids
- ecommerce
- Enterprise Search
- Events
- functions
- Hadoop
- Libraries
- Lucene
- Lucene Connector Framework
- Lucid Imagination
  - Lucid Imagination Solutions
- LucidGaze
- LucidWorks
- Lucy
- Mahout
- ManifoldCF
- NoSQL
- nutch
- Open Relevance Project
- Open Source
- opennlp
- Payloads
- PyLucene
- Relevancy

Don't Use Java 7, For Anything

July 28, 2011

Posted by *hossman*

Java 7 GA was released today, but as noted by Uwe Schindler, there are some very frightening bugs in HotSpot Loop optimizations that are enabled by default. In the best case scenario, these bugs cause the JVM to crash. In the worst case scenario, they cause incorrect execution of loops.

Bottom Line: Don't use Java 7 for anything (unless maybe you know you don't have any loops in your java code)

From: Uwe Schindler
Date: Thu, 28 Jul 2011 23:13:36 +0200
Subject: [WARNING] Index corruption and crashes in Apache Lucene Core / Apache Solr with Java 7

Hello Apache Lucene & Apache Solr users,
Hello users of other Java-based Apache projects,

Oracle released Java 7 today. Unfortunately it contains hotspot compiler optimizations, which miscompile some loops. This can affect code of several Apache projects. Sometimes JVMs only crash, but in several cases, results calculated can be incorrect, leading to bugs in applications (see Hotspot bugs 7070134 [1], 7044738 [2], 7068051 [3]).

Apache Lucene Core and Apache Solr are two Apache projects, which are affected by these bugs, namely all versions released until today. Solr users with the default configuration will have Java crashing with SIGSEGV as soon as they start to index documents, as one affected part is the well-known Porter stemmer (see LUCENE-3335 [4]). Other loops in Lucene may be miscompiled, too, leading to index corruption (especially on Lucene trunk with pulsing codec; other loops may be affected, too – LUCENE-3346 [5]).

Recent Posts

- Multivalued geolocation fields in Solr
- Monitoring Apache Solr and LucidWorks with Zabbix
- Lucene in Barcelona, in Action
- SF Bay Lucene/Solr Meetup Attracts 100 Attendees (and a special appearance by Doug Cutting!)
- Announcing LucidWorks 2.0, the search platform for Apache Solr/Lucene
- Some more European Search in Action
- Lucene goes from Enterprise Search to search platform
- SF Bay Area Lucene/Solr Meetup: 9/22 6:30PM (http://bit.ly/r19aZx)
- Happy Anniversary, Lucene! 10 years at the ASF
- Stump The Chump? Win A Prize!

Archives

InfoWorld H

JULY 29, 2011

Apach
compi

The newly
problems

By Ted Samso

Print

It looks like a f
Java 7 release
Apache Projec
come as a dis
waited five lon
language.

Released just
compiler optim
loops, potentially affecting projects such as
Apache Lucene Core, Apache Solr, and possibly
others, according to a warning from the Apache Project. At best, the bugs only cause JVMs to
crash; in other cases, they result in miscalculations that can lead to application bugginess.

veral"
ion of
ions of

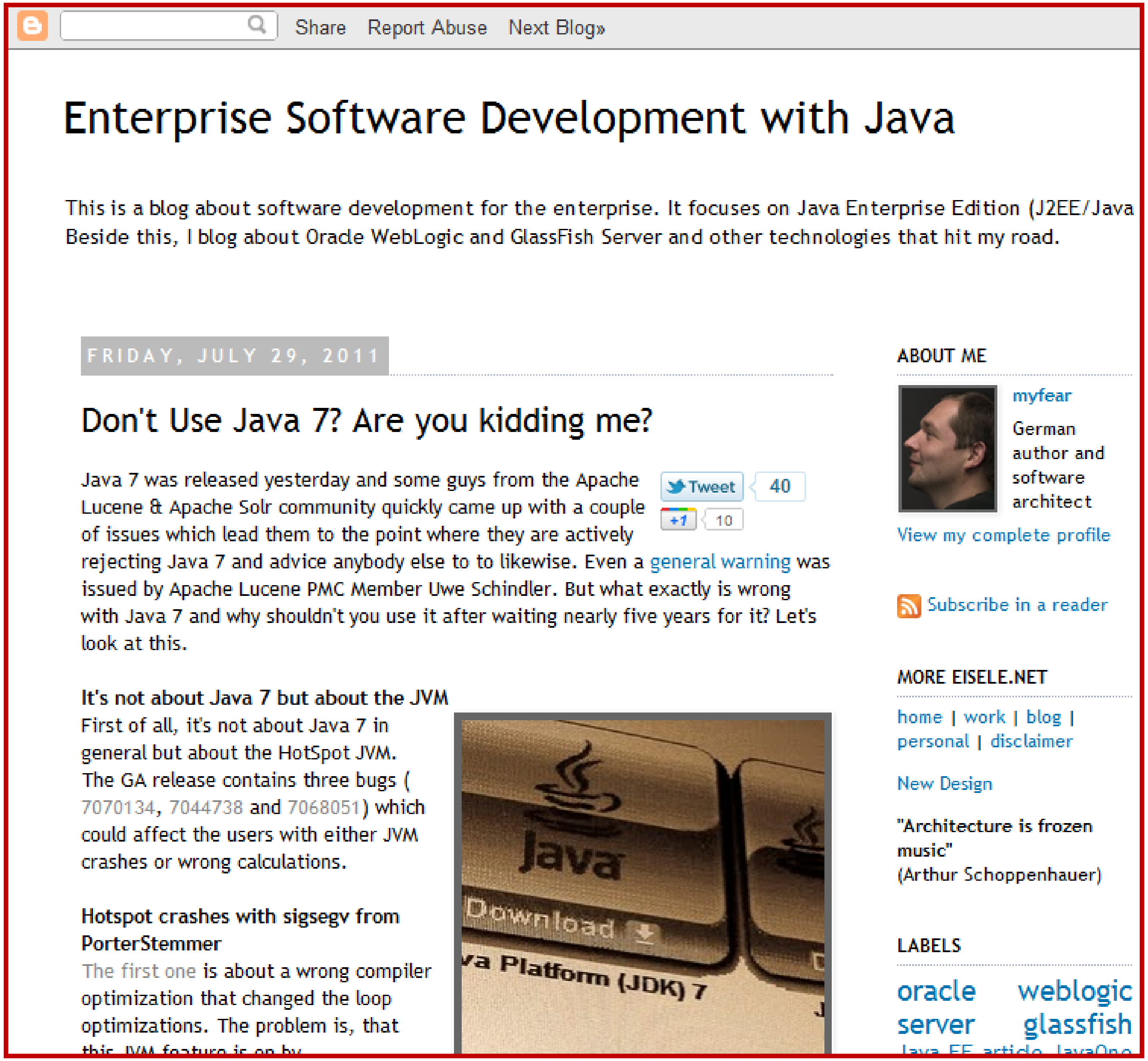
ase, so
rongly
version

release

Jessica Thornsby

Further analysis the week after

Further



Enterprise Software Development with Java

This is a blog about software development for the enterprise. It focuses on Java Enterprise Edition (J2EE/Java
Beside this, I blog about Oracle WebLogic and GlassFish Server and other technologies that hit my road.

FRIDAY, JULY 29, 2011

Don't Use Java 7? Are you kidding me?

Java 7 was released yesterday and some guys from the Apache Lucene & Apache Solr community quickly came up with a couple of issues which lead them to the point where they are actively rejecting Java 7 and advice anybody else to to likewise. Even a general warning was issued by Apache Lucene PMC Member Uwe Schindler. But what exactly is wrong with Java 7 and why shouldn't you use it after waiting nearly five years for it? Let's look at this.

It's not about Java 7 but about the JVM
First of all, it's not about Java 7 in general but about the HotSpot JVM. The GA release contains three bugs (7070134, 7044738 and 7068051) which could affect the users with either JVM crashes or wrong calculations.

Hotspot crashes with sigsegv from PorterStemmer
The first one is about a wrong compiler optimization that changed the loop optimizations. The problem is, that this JVM feature is on by

ABOUT ME

myfear
German author and software architect

View my complete profile

Subscribe in a reader

MORE EISELE.NET

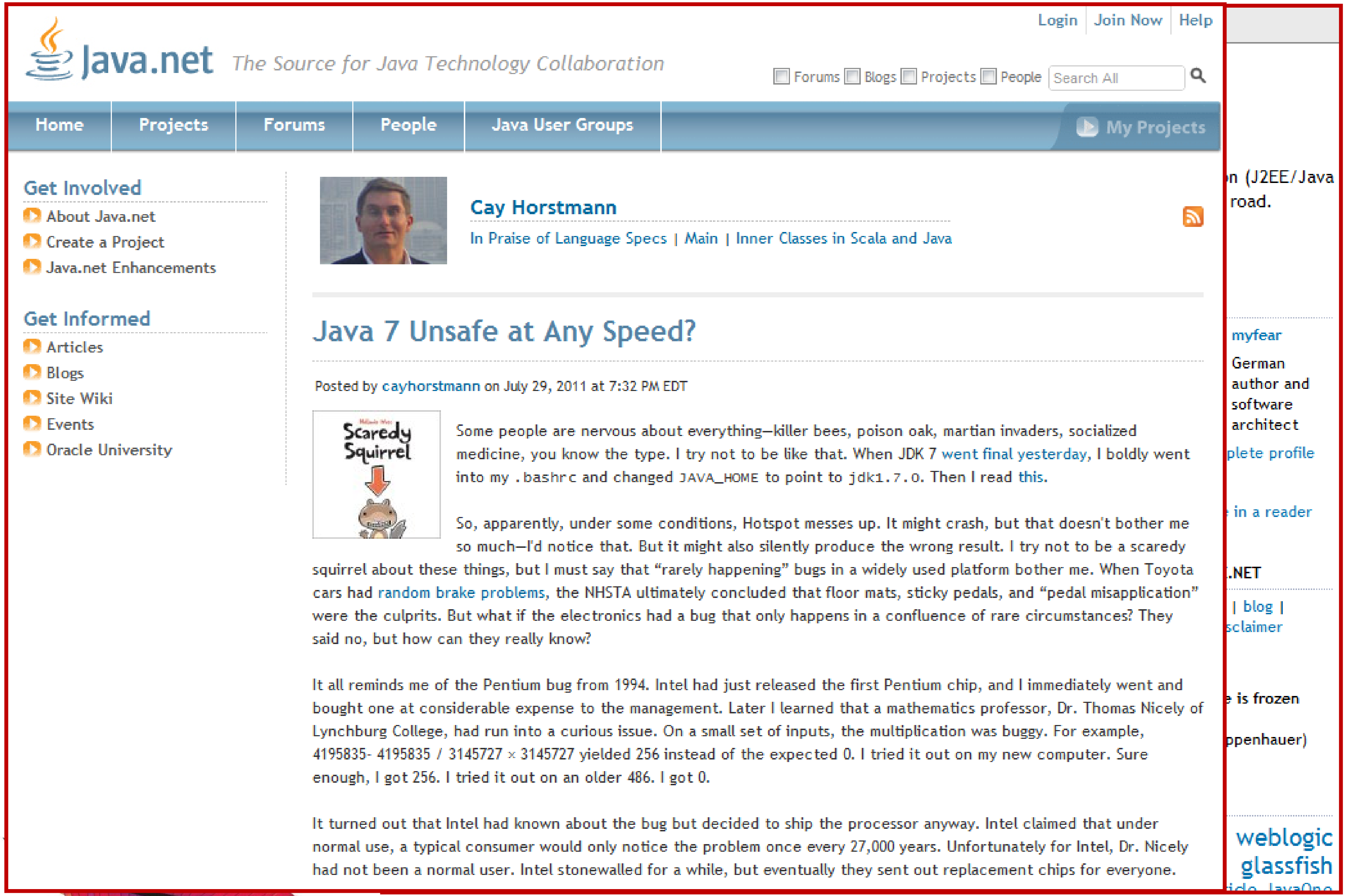
home | work | blog | personal | disclaimer

New Design

"Architecture is frozen music"
(Arthur Schoppenhauer)

LABELS

oracle weblogic server glassfish Java EE article JavaOne


Login | Join Now | Help
Java.net The Source for Java Technology Collaboration
□ Forums □ Blogs □ Projects □ People Search All
Home Projects Forums People Java User Groups My Projects

Get Involved

- About Java.net
- Create a Project
- Java.net Enhancements

Get Informed

- Articles
- Blogs
- Site Wiki
- Events
- Oracle University

Cay Horstmann

In Praise of Language Specs | Main | Inner Classes in Scala and Java

Java 7 Unsafe at Any Speed?

Posted by cayhorstmann on July 29, 2011 at 7:32 PM EDT

Some people are nervous about everything—killer bees, poison oak, martian invaders, socialized medicine, you know the type. I try not to be like that. When JDK 7 went final yesterday, I boldly went into my `.bashrc` and changed `JAVA_HOME` to point to `jdk1.7.0`. Then I read this.

So, apparently, under some conditions, Hotspot messes up. It might crash, but that doesn't bother me so much—I'd notice that. But it might also silently produce the wrong result. I try not to be a scaredy squirrel about these things, but I must say that "rarely happening" bugs in a widely used platform bother me. When Toyota cars had random brake problems, the NHSTA ultimately concluded that floor mats, sticky pedals, and "pedal misapplication" were the culprits. But what if the electronics had a bug that only happens in a confluence of rare circumstances? They said no, but how can they really know?

It all reminds me of the Pentium bug from 1994. Intel had just released the first Pentium chip, and I immediately went and bought one at considerable expense to the management. Later I learned that a mathematics professor, Dr. Thomas Nicely of Lynchburg College, had run into a curious issue. On a small set of inputs, the multiplication was buggy. For example, 4195835- 4195835 / 3145727 × 3145727 yielded 256 instead of the expected 0. I tried it out on my new computer. Sure enough, I got 256. I tried it out on an older 486. I got 0.

It turned out that Intel had known about the bug but decided to ship the processor anyway. Intel claimed that under normal use, a typical consumer would only notice the problem once every 27,000 years. Unfortunately for Intel, Dr. Nicely had not been a normal user. Intel stonewalled for a while, but eventually they sent out replacement chips for everyone.

n (J2EE/Java
road.

myfear
German
author and
software
architect
plete profile

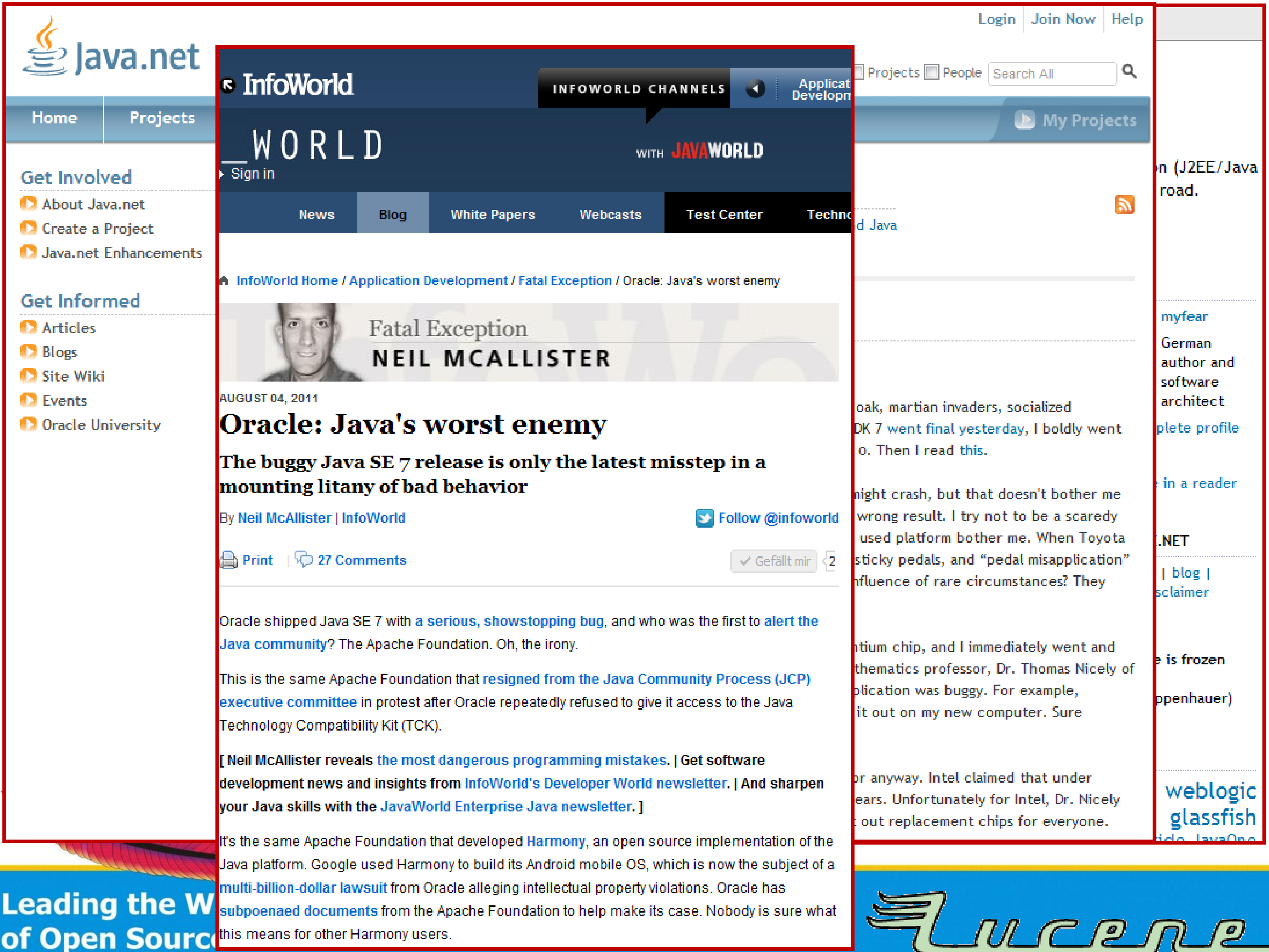
in a reader

.NET
| blog |
sclaimer

is frozen

ppenhauer)

weblogic
glassfish



Java.net

Login | Join Now | Help

□ Projects □ People Search All

Home | Projects | My Projects

Get Involved

- About Java.net
- Create a Project
- Java.net Enhancements

Get Informed

- Articles
- Blogs
- Site Wiki
- Events
- Oracle University

n (J2EE/Java
road.

d Java

oak, martian invaders, socialized
DK 7 went final yesterday, I boldly went
o. Then I read this.

night crash, but that doesn't bother me
wrong result. I try not to be a scaredy
used platform bother me. When Toyota
sticky pedals, and "pedal misapplication"
nfluence of rare circumstances? They

ntium chip, and I immediately went and
thematics professor, Dr. Thomas Nicely of
plication was buggy. For example,
it out on my new computer. Sure

or anyway. Intel claimed that under
ears. Unfortunately for Intel, Dr. Nicely
out replacement chips for everyone.

myfear
German
author and
software
architect
plete profile

in a reader

.NET

| blog |
sclaimer

is frozen

ppenhauer)

weblogic
glassfish

InfoWorld | INFOWORLD CHANNELS | Applicat Developm

_WORLD WITH JAVAWORLD

Sign in

News | Blog | White Papers | Webcasts | Test Center | Techno

InfoWorld Home / Application Development / Fatal Exception / Oracle: Java's worst enemy

Fatal Exception
NEIL MCALLISTER

AUGUST 04, 2011

Oracle: Java's worst enemy

The buggy Java SE 7 release is only the latest misstep in a mounting litany of bad behavior

By Neil McAllister | InfoWorld   Follow @infoworld

Print | 27 Comments   ✓ Gefällt mir 2

Oracle shipped Java SE 7 with a serious, showstopping bug, and who was the first to alert the Java community? The Apache Foundation. Oh, the irony.

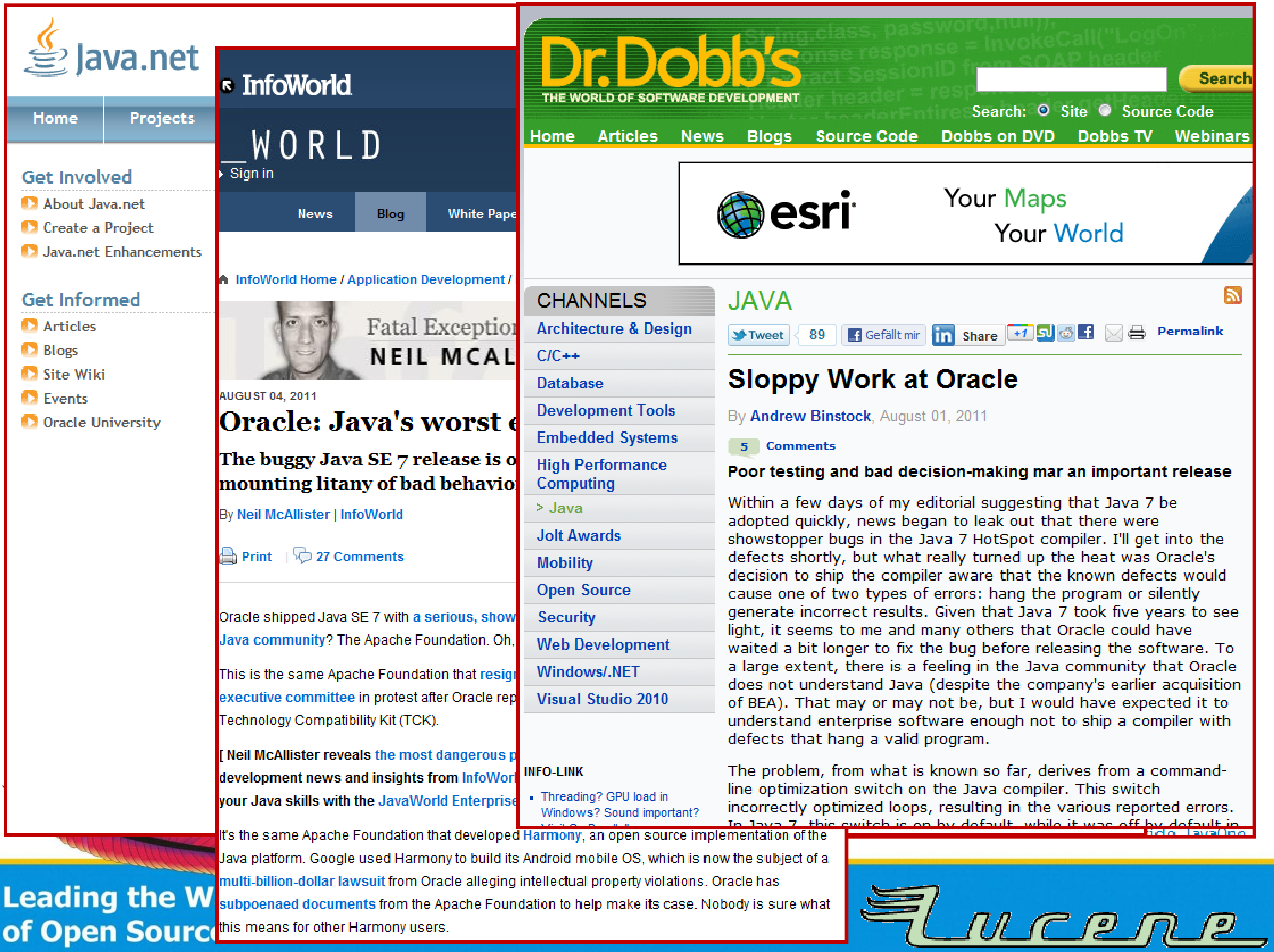
This is the same Apache Foundation that resigned from the Java Community Process (JCP) executive committee in protest after Oracle repeatedly refused to give it access to the Java Technology Compatibility Kit (TCK).

[Neil McAllister reveals the most dangerous programming mistakes. | Get software development news and insights from InfoWorld's Developer World newsletter. | And sharpen your Java skills with the JavaWorld Enterprise Java newsletter.]

It's the same Apache Foundation that developed Harmony, an open source implementation of the Java platform. Google used Harmony to build its Android mobile OS, which is now the subject of a multi-billion-dollar lawsuit from Oracle alleging intellectual property violations. Oracle has subpoenaed documents from the Apache Foundation to help make its case. Nobody is sure what this means for other Harmony users.

Leading the W
of Open Sourc

Lucene



Java.net

Home | Projects

Get Involved

- About Java.net
- Create a Project
- Java.net Enhancements

Get Informed

- Articles
- Blogs
- Site Wiki
- Events
- Oracle University

InfoWorld

_WORLD

Sign in

News | Blog | White Pape

InfoWorld Home / Application Development /

Fatal Exceptio

NEIL MCAL

AUGUST 04, 2011

Oracle: Java's worst e

The buggy Java SE 7 release is o mounting litany of bad behavio

By Neil McAllister | InfoWorld

Print | 27 Comments

Oracle shipped Java SE 7 with a serious, show Java community? The Apache Foundation. Oh,

This is the same Apache Foundation that resig executive committee in protest after Oracle rep Technology Compatibility Kit (TCK).

[Neil McAllister reveals the most dangerous p development news and insights from InfoWorl your Java skills with the JavaWorld Enterprise

It's the same Apache Foundation that developed Harmony, an open source implementation of the Java platform. Google used Harmony to build its Android mobile OS, which is now the subject of a multi-billion-dollar lawsuit from Oracle alleging intellectual property violations. Oracle has subpoenaed documents from the Apache Foundation to help make its case. Nobody is sure what this means for other Harmony users.

Dr.Dobb's

THE WORLD OF SOFTWARE DEVELOPMENT

Search

Search: ● Site ○ Source Code

Home | Articles | News | Blogs | Source Code | Dobbs on DVD | Dobbs TV | Webinars

CHANNELS

- Architecture & Design
- C/C++
- Database
- Development Tools
- Embedded Systems
- High Performance Computing
- > Java
- Jolt Awards
- Mobility
- Open Source
- Security
- Web Development
- Windows/.NET
- Visual Studio 2010

INFO-LINK

- Threading? GPU load in Windows? Sound important?

JAVA

Tweet 89 | Gefällt mir | in Share | +1 | Permalink

Sloppy Work at Oracle

By Andrew Binstock, August 01, 2011

5 Comments

Poor testing and bad decision-making mar an important release

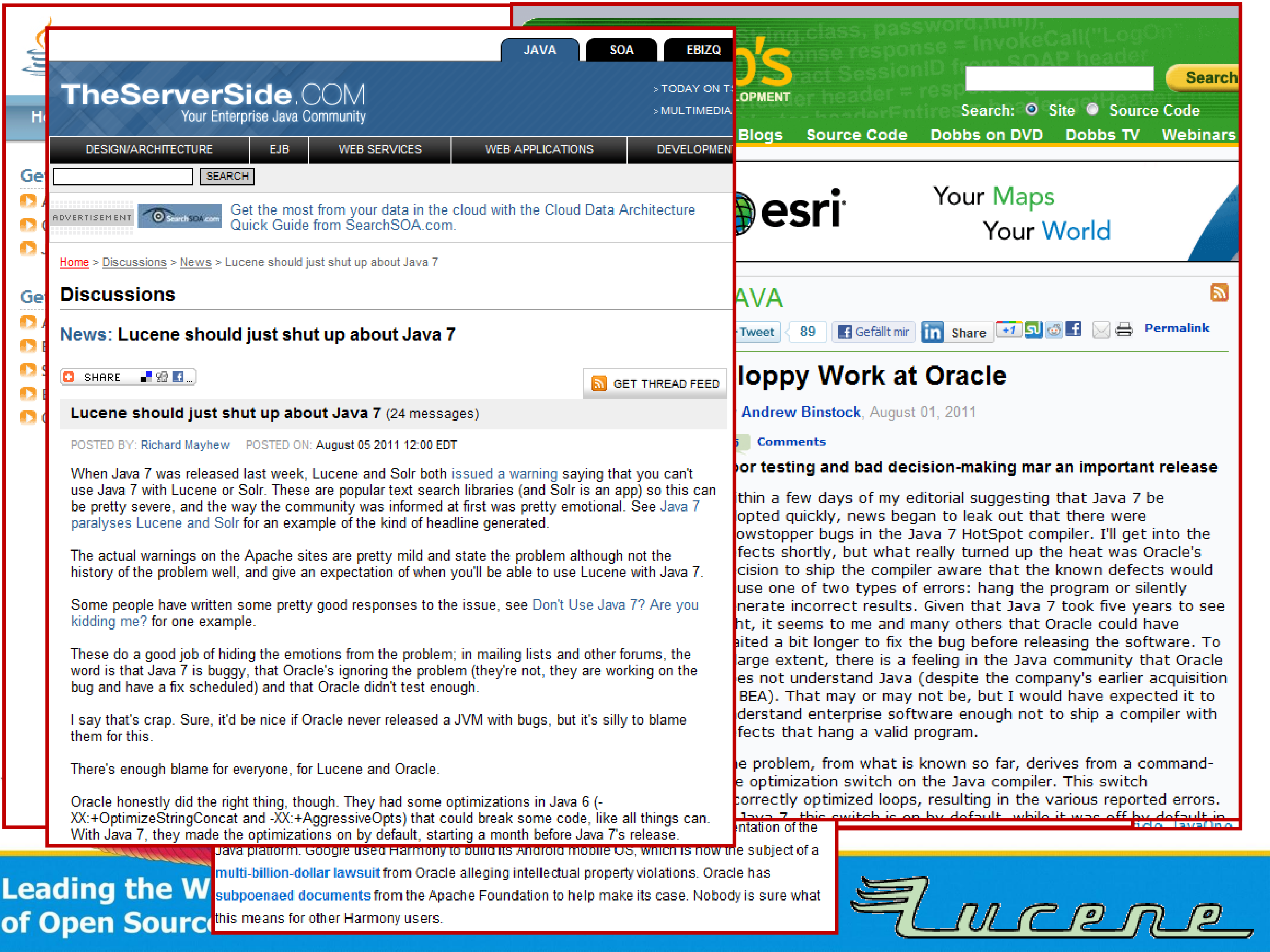
Within a few days of my editorial suggesting that Java 7 be adopted quickly, news began to leak out that there were showstopper bugs in the Java 7 HotSpot compiler. I'll get into the defects shortly, but what really turned up the heat was Oracle's decision to ship the compiler aware that the known defects would cause one of two types of errors: hang the program or silently generate incorrect results. Given that Java 7 took five years to see light, it seems to me and many others that Oracle could have waited a bit longer to fix the bug before releasing the software. To a large extent, there is a feeling in the Java community that Oracle does not understand Java (despite the company's earlier acquisition of BEA). That may or may not be, but I would have expected it to understand enterprise software enough not to ship a compiler with defects that hang a valid program.

The problem, from what is known so far, derives from a command-line optimization switch on the Java compiler. This switch incorrectly optimized loops, resulting in the various reported errors. In Java 7, this switch is on by default, while it was off by default in

Leading the W

of Open Sourc

Lucene



TheServerSide.COM
Your Enterprise Java Community

Home > Discussions > News > Lucene should just shut up about Java 7

Discussions

News: Lucene should just shut up about Java 7

Lucene should just shut up about Java 7 (24 messages)

POSTED BY: Richard Mayhew POSTED ON: August 05 2011 12:00 EDT

When Java 7 was released last week, Lucene and Solr both issued a warning saying that you can't use Java 7 with Lucene or Solr. These are popular text search libraries (and Solr is an app) so this can be pretty severe, and the way the community was informed at first was pretty emotional. See Java 7 paralyses Lucene and Solr for an example of the kind of headline generated.

The actual warnings on the Apache sites are pretty mild and state the problem although not the history of the problem well, and give an expectation of when you'll be able to use Lucene with Java 7.

Some people have written some pretty good responses to the issue, see Don't Use Java 7? Are you kidding me? for one example.

These do a good job of hiding the emotions from the problem; in mailing lists and other forums, the word is that Java 7 is buggy, that Oracle's ignoring the problem (they're not, they are working on the bug and have a fix scheduled) and that Oracle didn't test enough.

I say that's crap. Sure, it'd be nice if Oracle never released a JVM with bugs, but it's silly to blame them for this.

There's enough blame for everyone, for Lucene and Oracle.

Oracle honestly did the right thing, though. They had some optimizations in Java 6 (-XX:+OptimizeStringConcat and -XX:+AggressiveOpts) that could break some code, like all things can. With Java 7, they made the optimizations on by default, starting a month before Java 7's release.

AVA

loppy Work at Oracle

Andrew Binstock, August 01, 2011

Comments

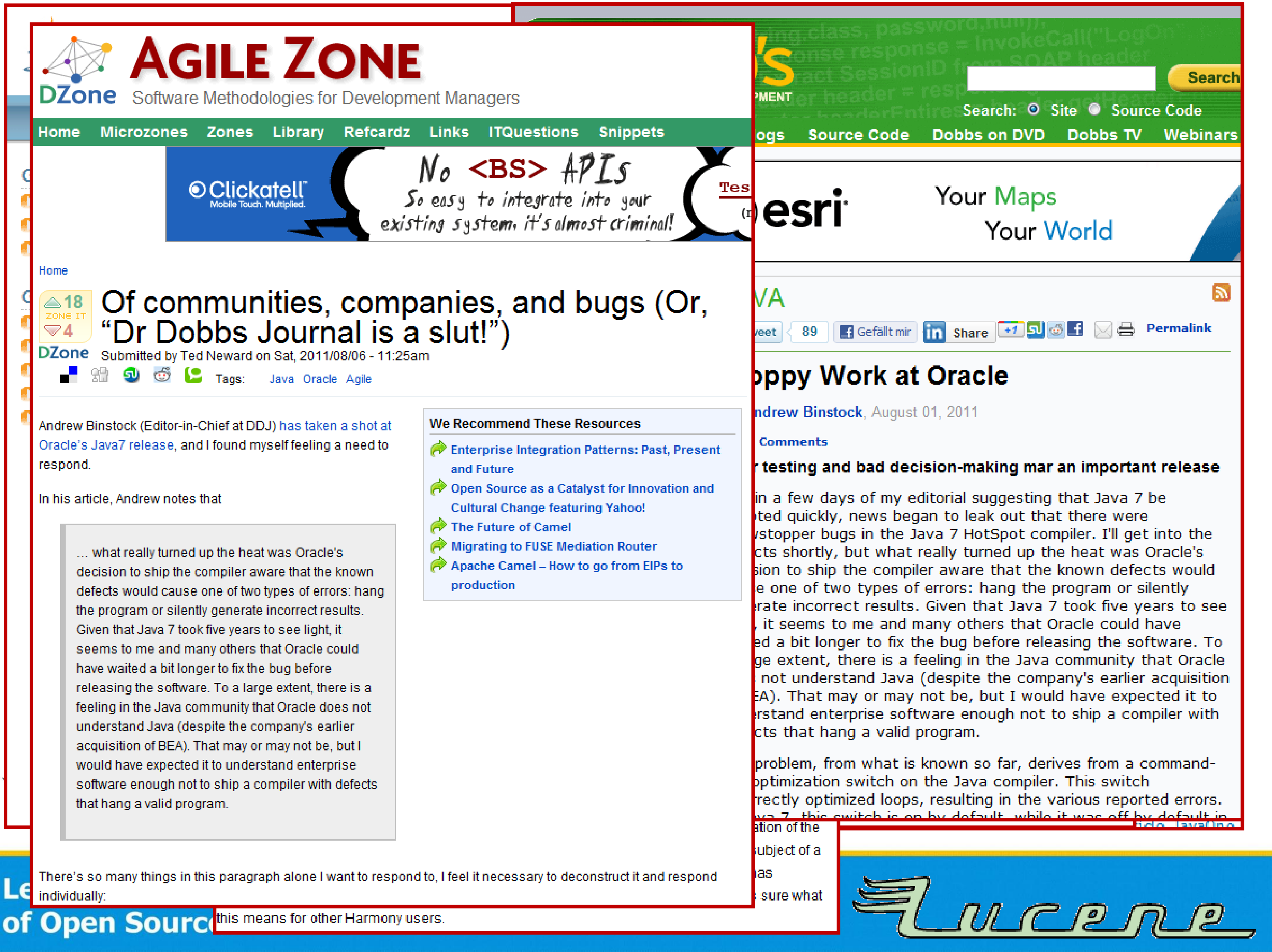
oor testing and bad decision-making mar an important release

thin a few days of my editorial suggesting that Java 7 be opted quickly, news began to leak out that there were owstopper bugs in the Java 7 HotSpot compiler. I'll get into the fects shortly, but what really turned up the heat was Oracle's cision to ship the compiler aware that the known defects would use one of two types of errors: hang the program or silently nerate incorrect results. Given that Java 7 took five years to see ht, it seems to me and many others that Oracle could have ited a bit longer to fix the bug before releasing the software. To arge extent, there is a feeling in the Java community that Oracle es not understand Java (despite the company's earlier acquisition BEA). That may or may not be, but I would have expected it to derstand enterprise software enough not to ship a compiler with fects that hang a valid program.

e problem, from what is known so far, derives from a command-e optimization switch on the Java compiler. This switch correctly optimized loops, resulting in the various reported errors.

ntation of the Java platform. Google used Harmony to build its Android mobile OS, which is now the subject of a multi-billion-dollar lawsuit from Oracle alleging intellectual property violations. Oracle has subpoenaed documents from the Apache Foundation to help make its case. Nobody is sure what this means for other Harmony users.

Leading the W
of Open Sourc



AGILE ZONE
DZone
Software Methodologies for Development Managers
Home Microzones Zones Library Refcardz Links ITQuestions Snippets

Of communities, companies, and bugs (Or, "Dr Dobbs Journal is a slut!")

Submitted by Ted Neward on Sat, 2011/08/06 - 11:25am

Tags: Java Oracle Agile

Andrew Binstock (Editor-in-Chief at DDJ) has taken a shot at Oracle's Java7 release, and I found myself feeling a need to respond.

In his article, Andrew notes that

> ... what really turned up the heat was Oracle's decision to ship the compiler aware that the known defects would cause one of two types of errors: hang the program or silently generate incorrect results. Given that Java 7 took five years to see light, it seems to me and many others that Oracle could have waited a bit longer to fix the bug before releasing the software. To a large extent, there is a feeling in the Java community that Oracle does not understand Java (despite the company's earlier acquisition of BEA). That may or may not be, but I would have expected it to understand enterprise software enough not to ship a compiler with defects that hang a valid program.

There's so many things in this paragraph alone I want to respond to, I feel it necessary to deconstruct it and respond individually:

We Recommend These Resources
Enterprise Integration Patterns: Past, Present and Future
Open Source as a Catalyst for Innovation and Cultural Change featuring Yahoo!
The Future of Camel
Migrating to FUSE Mediation Router
Apache Camel – How to go from EIPs to production

's
PMENT
Search
Search: ● Site ○ Source Code
ogs Source Code Dobbs on DVD Dobbs TV Webinars

oppy Work at Oracle

ndrew Binstock, August 01, 2011

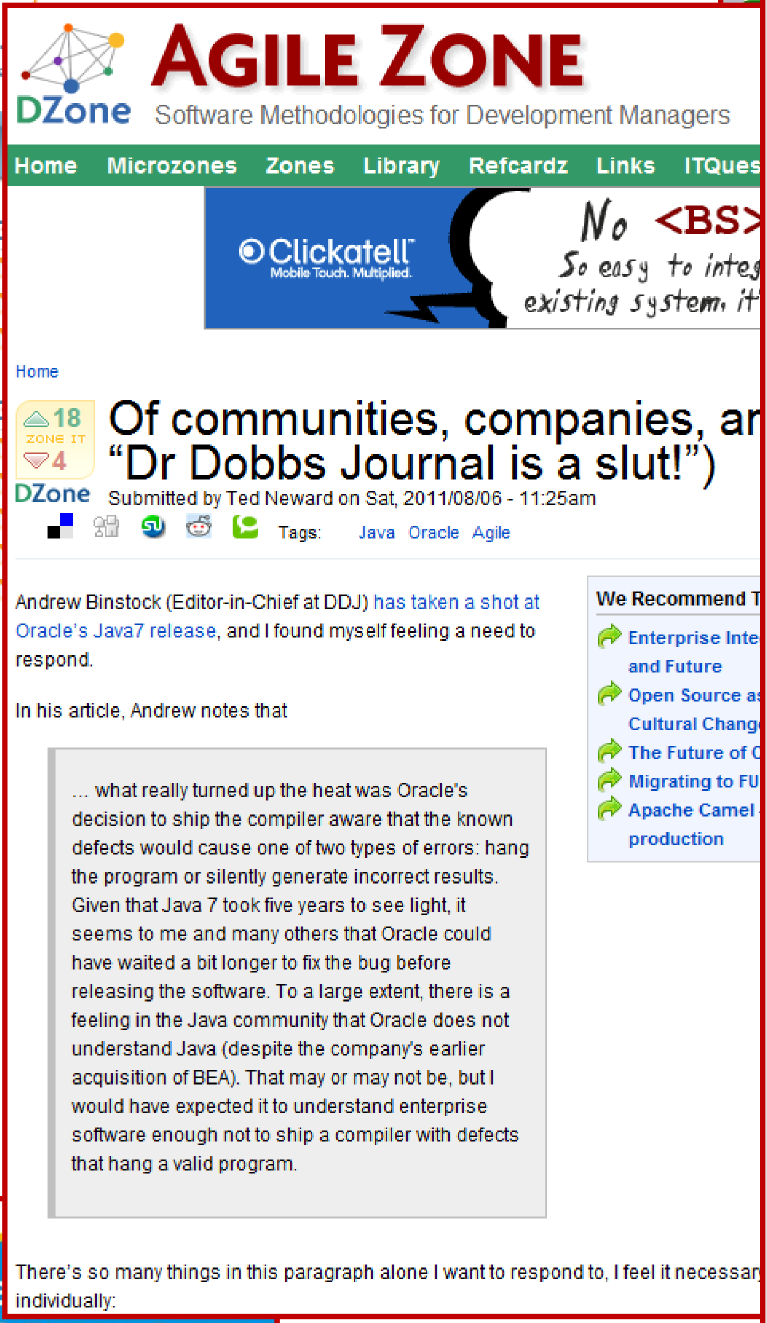
Comments

r testing and bad decision-making mar an important release

in a few days of my editorial suggesting that Java 7 be
ted quickly, news began to leak out that there were
stopper bugs in the Java 7 HotSpot compiler. I'll get into the
cts shortly, but what really turned up the heat was Oracle's
sion to ship the compiler aware that the known defects would
e one of two types of errors: hang the program or silently
rate incorrect results. Given that Java 7 took five years to see
, it seems to me and many others that Oracle could have
ed a bit longer to fix the bug before releasing the software. To
ge extent, there is a feeling in the Java community that Oracle
not understand Java (despite the company's earlier acquisition
EA). That may or may not be, but I would have expected it to
rstand enterprise software enough not to ship a compiler with
cts that hang a valid program.

problem, from what is known so far, derives from a command-
optimization switch on the Java compiler. This switch
rectly optimized loops, resulting in the various reported errors.

ation of the
ubject of a
as
sure what
this means for other Harmony users.



AGILE ZONE
DZone
Software Methodologies for Development Managers
Home Microzones Zones Library Refcardz Links ITQues
Clickatell Mobile Touch. Multiplied.
No <BS>
So easy to integ
existing system, it
Home
18
ZONE IT
4
DZone
Tags: Java Oracle Agile

Of communities, companies, ar
“Dr Dobbs Journal is a slut!”)

Submitted by Ted Neward on Sat, 2011/08/06 - 11:25am

Andrew Binstock (Editor-in-Chief at DDJ) has taken a shot at Oracle's Java7 release, and I found myself feeling a need to respond.

In his article, Andrew notes that

> … what really turned up the heat was Oracle's decision to ship the compiler aware that the known defects would cause one of two types of errors: hang the program or silently generate incorrect results. Given that Java 7 took five years to see light, it seems to me and many others that Oracle could have waited a bit longer to fix the bug before releasing the software. To a large extent, there is a feeling in the Java community that Oracle does not understand Java (despite the company's earlier acquisition of BEA). That may or may not be, but I would have expected it to understand enterprise software enough not to ship a compiler with defects that hang a valid program.

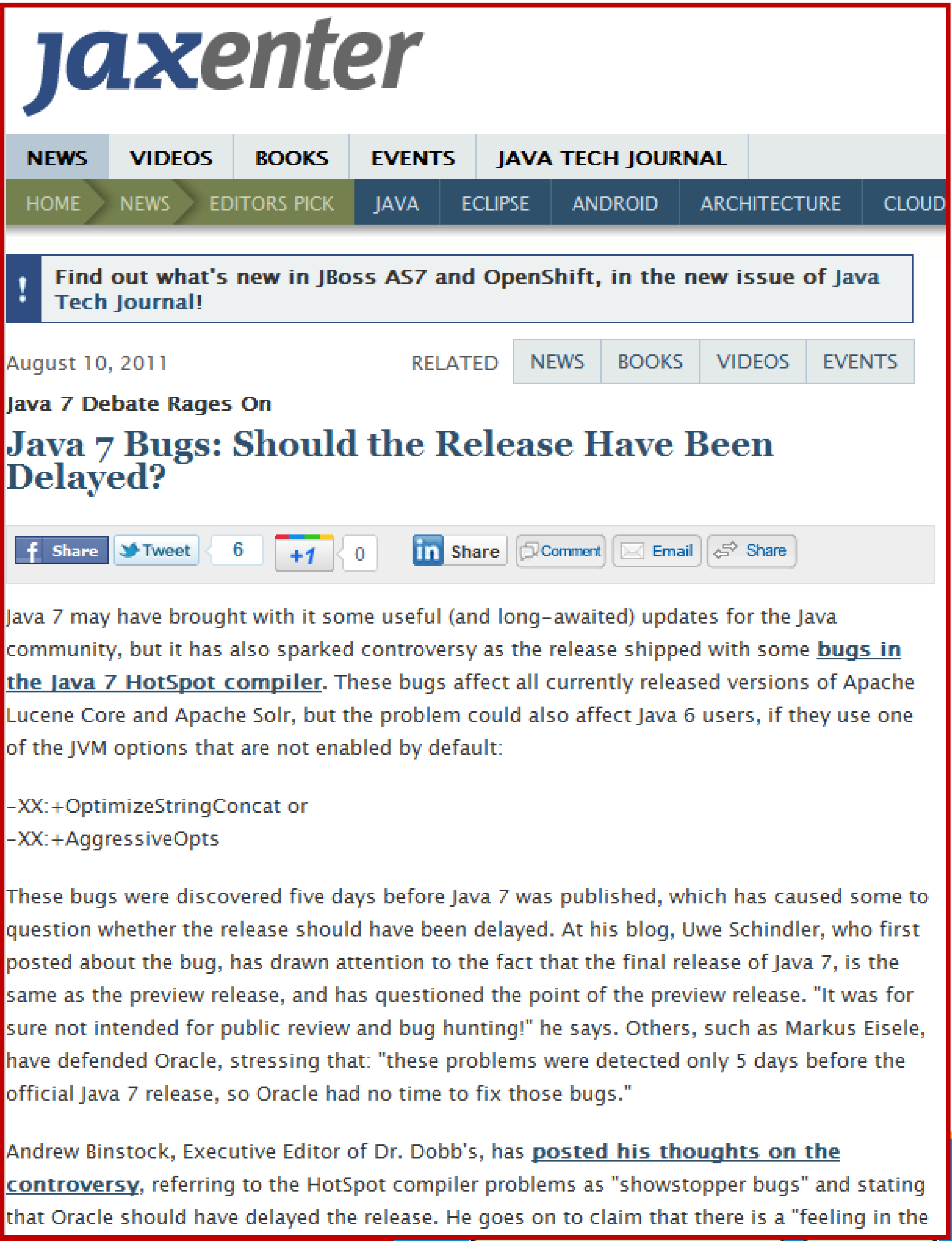
There's so many things in this paragraph alone I want to respond to, I feel it necessar individually:

We Recommend T

- Enterprise Inte and Future
- Open Source as Cultural Chang
- The Future of C
- Migrating to FU
- Apache Camel production

Le
of Open Sourc

this means for other Harmony users.



jaxenter
NEWS VIDEOS BOOKS EVENTS JAVA TECH JOURNAL
HOME NEWS EDITORS PICK JAVA ECLIPSE ANDROID ARCHITECTURE CLOUD
RELATED NEWS BOOKS VIDEOS EVENTS
Share Tweet 6 +1 0 Share Comment Email Share

Find out what's new in JBoss AS7 and OpenShift, in the new issue of Java Tech Journal!

August 10, 2011

Java 7 Debate Rages On

Java 7 Bugs: Should the Release Have Been Delayed?

Java 7 may have brought with it some useful (and long-awaited) updates for the Java community, but it has also sparked controversy as the release shipped with some bugs in the Java 7 HotSpot compiler. These bugs affect all currently released versions of Apache Lucene Core and Apache Solr, but the problem could also affect Java 6 users, if they use one of the JVM options that are not enabled by default:

-XX:+OptimizeStringConcat or
-XX:+AggressiveOpts

These bugs were discovered five days before Java 7 was published, which has caused some to question whether the release should have been delayed. At his blog, Uwe Schindler, who first posted about the bug, has drawn attention to the fact that the final release of Java 7, is the same as the preview release, and has questioned the point of the preview release. "It was for sure not intended for public review and bug hunting!" he says. Others, such as Markus Eisele, have defended Oracle, stressing that: "these problems were detected only 5 days before the official Java 7 release, so Oracle had no time to fix those bugs."

Andrew Binstock, Executive Editor of Dr. Dobb's, has posted his thoughts on the controversy, referring to the HotSpot compiler problems as "showstopper bugs" and stating that Oracle should have delayed the release. He goes on to claim that there is a "feeling in the

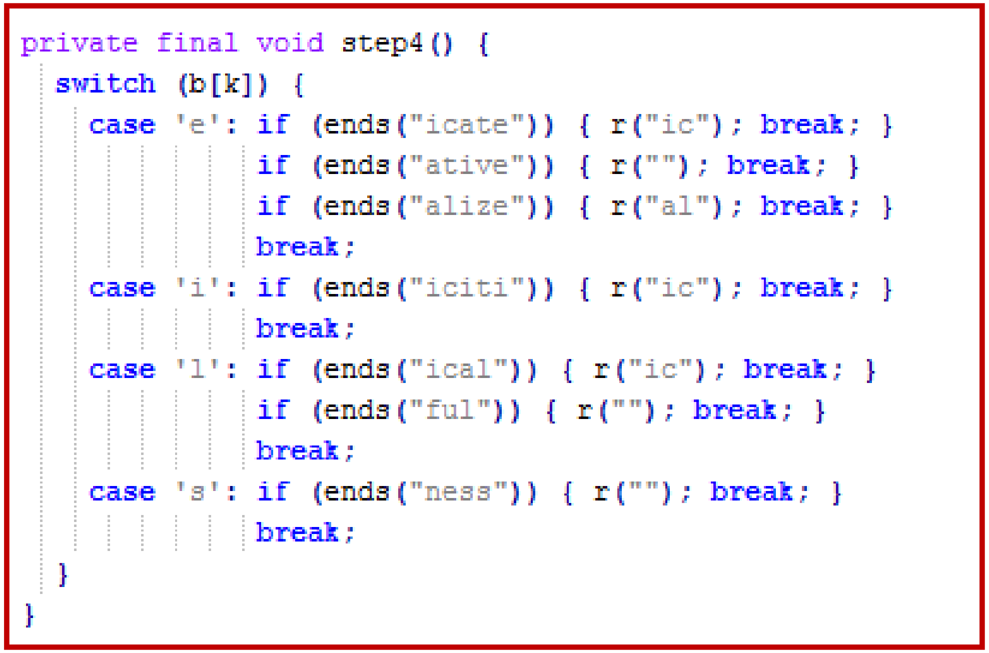
Oracle’s offers

- **Dalibor Topic** *(Oracle)* explained Oracles plans for managing bugs in his blog: “A bugs live” (http://robilad.livejournal.com/87097.html):
  - They are trying to improve the bug reports coming in over the web interface
  - Information how bug fixes are merged from Java 8 to Java 7, special cases for Hotspot
- Oracle offers **Java CAP** (Compatibility and Performance Program):
  - early access to Java builds to check compatibility
  - support technican assigned
- Oracle offers almost weekly **preview builds** of JDK **7u2** and **6u29** on http://jdk(7|6).java.net

Java 7 Crashes Eclipse…

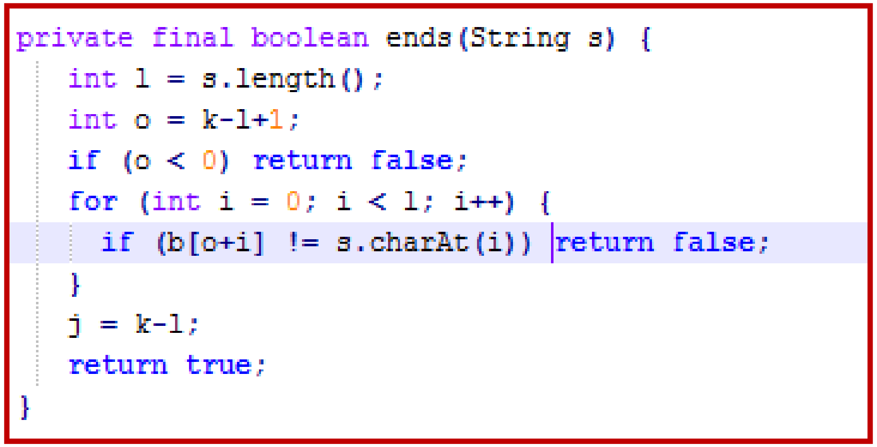
THE PORTER STEMMER SIGSEGV BUG

What’s wrong with these methods?



```
private final void step4() {
  switch (b[k]) {
    case 'e': if (ends("icate")) { r("ic"); break; }
              if (ends("ative")) { r(""); break; }
              if (ends("alize")) { r("al"); break; }
              break;
    case 'i': if (ends("iciti")) { r("ic"); break; }
              break;
    case 'l': if (ends("ical")) { r("ic"); break; }
              if (ends("ful")) { r(""); break; }
              break;
    case 's': if (ends("ness")) { r(""); break; }
              break;
  }
}
```

Let’s try it out!



```
private final boolean ends(String s) {
  int l = s.length();
  int o = k-l+1;
  if (o < 0) return false;
  for (int i = 0; i < l; i++) {
    if (b[o+i] != s.charAt(i)) return false;
  }
  j = k-l;
  return true;
}
```

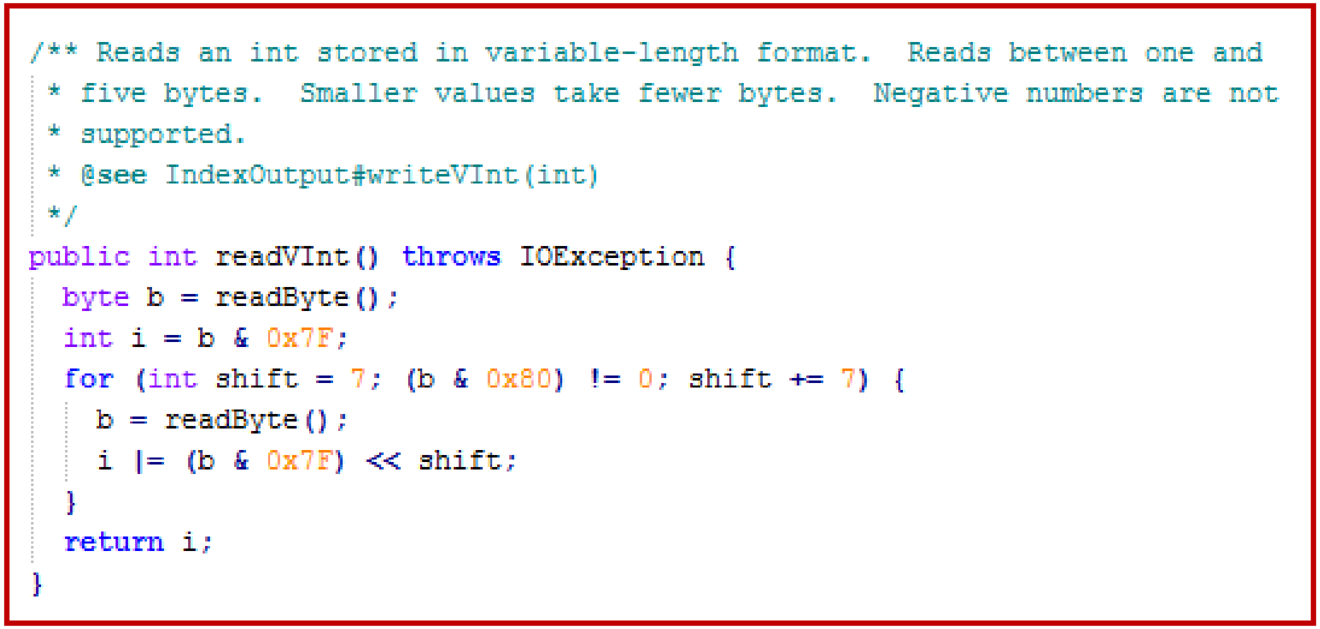
Conclusion: Porter Stemmer Bug

- Less serious bug as your virtual machine simply crashes. You won’t use it!
- Oracle made bug report “serious”, as this affects their software reproducible to everyone.
- Can be prevented by JVM option:
  `-XX:-UseLoopPredicate`

Loop Unwinding

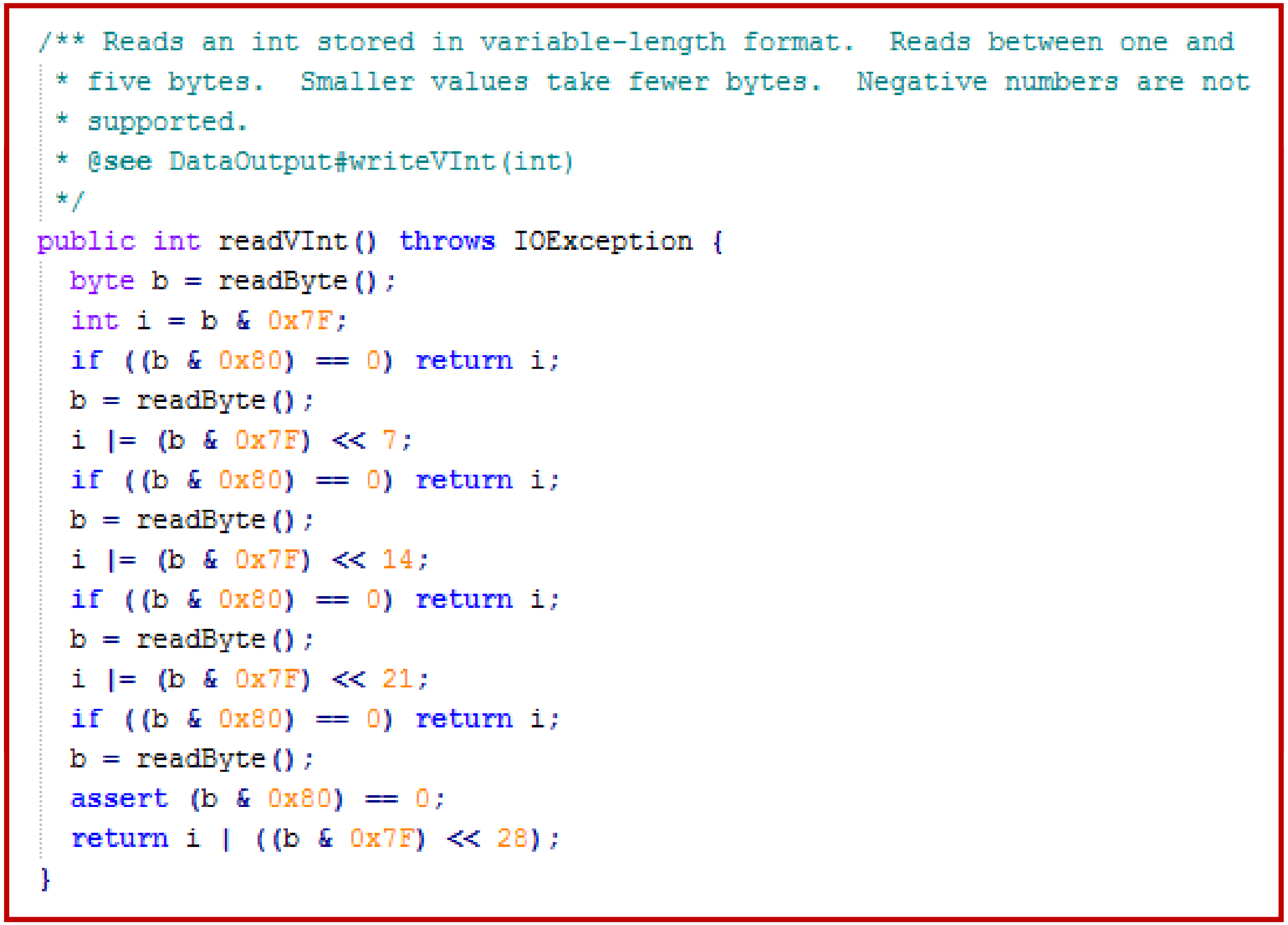
THE VINT BUG

What’s wrong with this method?



```
/** Reads an int stored in variable-length format.  Reads between one and
 * five bytes.  Smaller values take fewer bytes.  Negative numbers are not
 * supported.
 * @see IndexOutput#writeVInt(int)
 */
public int readVInt() throws IOException {
  byte b = readByte();
  int i = b & 0x7F;
  for (int shift = 7; (b & 0x80) != 0; shift += 7) {
    b = readByte();
    i |= (b & 0x7F) << shift;
  }
  return i;
}
```

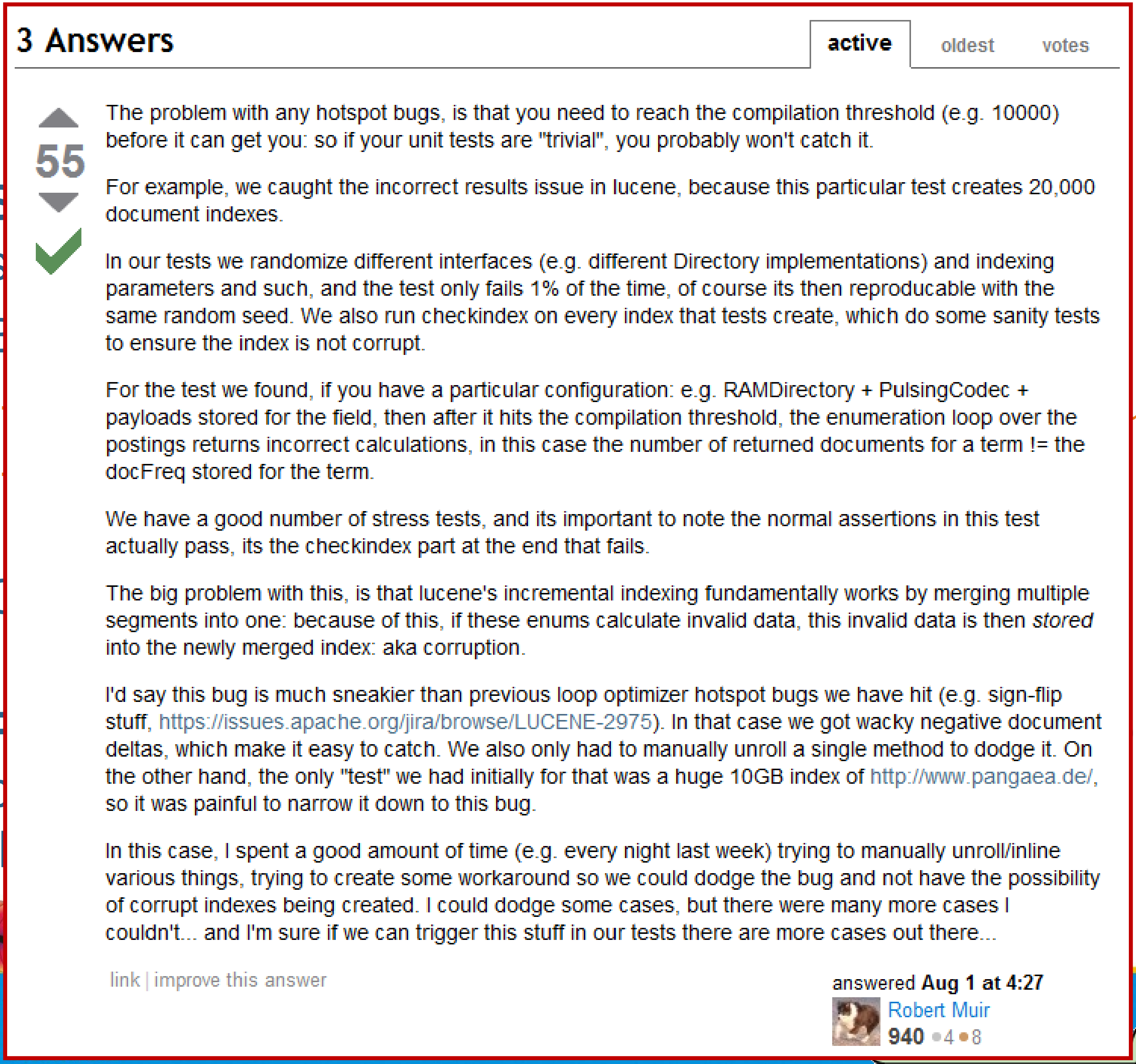
What’s wrong with this method?



```java
/** Reads an int stored in variable-length format.  Reads between one and
 * five bytes.  Smaller values take fewer bytes.  Negative numbers are not
 * supported.
 * @see DataOutput#writeVInt(int)
 */
public int readVInt() throws IOException {
  byte b = readByte();
  int i = b & 0x7F;
  if ((b & 0x80) == 0) return i;
  b = readByte();
  i |= (b & 0x7F) << 7;
  if ((b & 0x80) == 0) return i;
  b = readByte();
  i |= (b & 0x7F) << 14;
  if ((b & 0x80) == 0) return i;
  b = readByte();
  i |= (b & 0x7F) << 21;
  if ((b & 0x80) == 0) return i;
  b = readByte();
  assert (b & 0x80) == 0;
  return i | ((b & 0x7F) << 28);
}
```

Conclusion: Vint Bug

- **Serious data corruption:** Some methods using loops silently return wrong results!
- Bug already existed in Java 6
  - appeared some time after 1.6.0_18, enabled by default
  - is prevented since Lucene 3.1 by manual loop unwinding *(helps only in Java 6)*
- Cannot easily be reproduced, Oracle assigned “medium” bug priority – was never fixed in Java 6.
- **Problems got worse with Java 7**, only safe way to prevent is to disable loop unwinding completely, but that makes Lucene very slow.



The problem with any hotspot bugs, is that you need to reach the compilation threshold (e.g. 10000) before it can get you: so if your unit tests are "trivial", you probably won't catch it.

For example, we caught the incorrect results issue in lucene, because this particular test creates 20,000 document indexes.

In our tests we randomize different interfaces (e.g. different Directory implementations) and indexing parameters and such, and the test only fails 1% of the time, of course its then reproducable with the same random seed. We also run checkindex on every index that tests create, which do some sanity tests to ensure the index is not corrupt.

For the test we found, if you have a particular configuration: e.g. RAMDirectory + PulsingCodec + payloads stored for the field, then after it hits the compilation threshold, the enumeration loop over the postings returns incorrect calculations, in this case the number of returned documents for a term != the docFreq stored for the term.

We have a good number of stress tests, and its important to note the normal assertions in this test actually pass, its the checkindex part at the end that fails.

The big problem with this, is that lucene's incremental indexing fundamentally works by merging multiple segments into one: because of this, if these enums calculate invalid data, this invalid data is then *stored* into the newly merged index: aka corruption.

I'd say this bug is much sneakier than previous loop optimizer hotspot bugs we have hit (e.g. sign-flip stuff, https://issues.apache.org/jira/browse/LUCENE-2975). In that case we got wacky negative document deltas, which make it easy to catch. We also only had to manually unroll a single method to dodge it. On the other hand, the only "test" we had initially for that was a huge 10GB index of http://www.pangaea.de/, so it was painful to narrow it down to this bug.

In this case, I spent a good amount of time (e.g. every night last week) trying to manually unroll/inline various things, trying to create some workaround so we could dodge the bug and not have the possibility of corrupt indexes being created. I could dodge some cases, but there were many more cases I couldn't... and I'm sure if we can trigger this stuff in our tests there are more cases out there...

answered Aug 1 at 4:27
Robert Muir
940 ●4 ●8

Hands-On

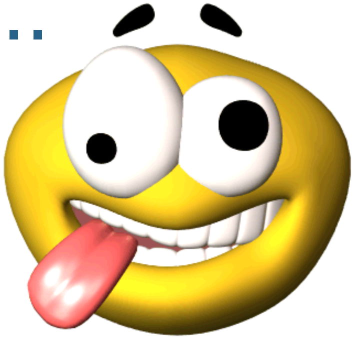
HOW TO DEBUG HOTSPOT PROBLEMS

First…

- Fetch some beer!
- Tell your girlfriend that you will not come to bed!
- Forget about Eclipse & Co! We need a command line and our source code…

Hardcore: **Debugging *without* Debugger**

- Open `hs_err` file and watch for stack trace.
  (if your JVM crashed like in Porter stemmer)
- *Otherwise:* disable Hotspot to verify that it’s not a logic error! (`-Xint` / `-Xbatch`)
- Start to dig around by adding `System.out.println`, assertions,...
  Please note: You cannot use a debugger!!!



Digging…

- If you found a method that works incorrectly, disable Hotspot optimizations for only that one:

```
-XX:CompileCommand=exclude,your/package/Class,method
```

  - If program works now, you found a workaround!
  - But this may not be the root cause - does not help at all!
- Step down the call hierarchy and replace exclusion by methods called from this one.
- **Open a bug report at Oracle!**
- Inform **hotspot-compiler-dev@openjdk.java.net** mailing list.

Let’s try it out!

Recommendation

- Test methods with tight loops in real-world scenarios
  - Large data structures that require lots of iterations
  - Lower the compilation threshold during tests
- Test `-client` and `-server`!
- Use randomness during testing
  - Reproduceable random seeds
  - See Lucene's modified JUnit test framework

Thank You!