Apache Lucene and Java 9+ Opportunities and Challenges for Apache Solr and Elasticsearch

Uwe Schindler

SD DataSolutions GmbH / Apache Software Foundation

thetaph1 – https://www.thetaphi.de

My Background

- **Committer** and **PMC member** of **Apache Lucene and Solr** - main focus is on development of Lucene Core.
- Implemented fast numerical search and maintaining the new attribute-based text analysis API. Well known as *Generics and Sophisticated Backwards Compatibility* 👮.
- **Elasticsearch** lover.
- Working as consultant and software architect at **SD DataSolutions GmbH** in Bremen, Germany.
- Maintaining **PANGAEA** (Data Publisher for Earth & Environmental Science) where I implemented the portal's geo-spatial retrieval functions with Apache Lucene Core and Elasticsearch.

What is this talk about?

- **History** of **Java 9** and **Apache Lucene/Solr**
- **Migration** and **testing** your with **Java 9's module system (Jigsaw)**
- Common **pitfalls** with Java 7 / Java 8 code, that just used to work
- *Performance?*

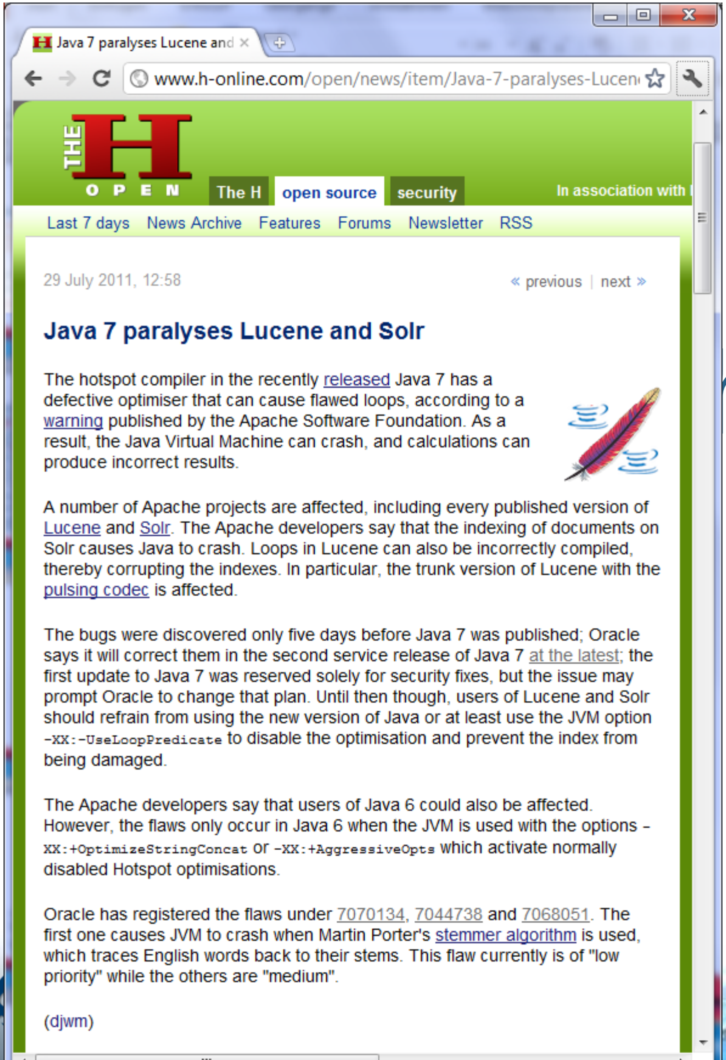
Oracle & Apache Lucene

History

Remember 2011?

Chronology:
Friday, July 29, 2011

nronology:
, July 29, 2011



Java 7 paralyses Lucene and

www.h-online.com/open/news/item/Java-7-paralyses-Lucen

The H | open source | security

Last 7 days | News Archive | Features | Forums | Newsletter | RSS

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.

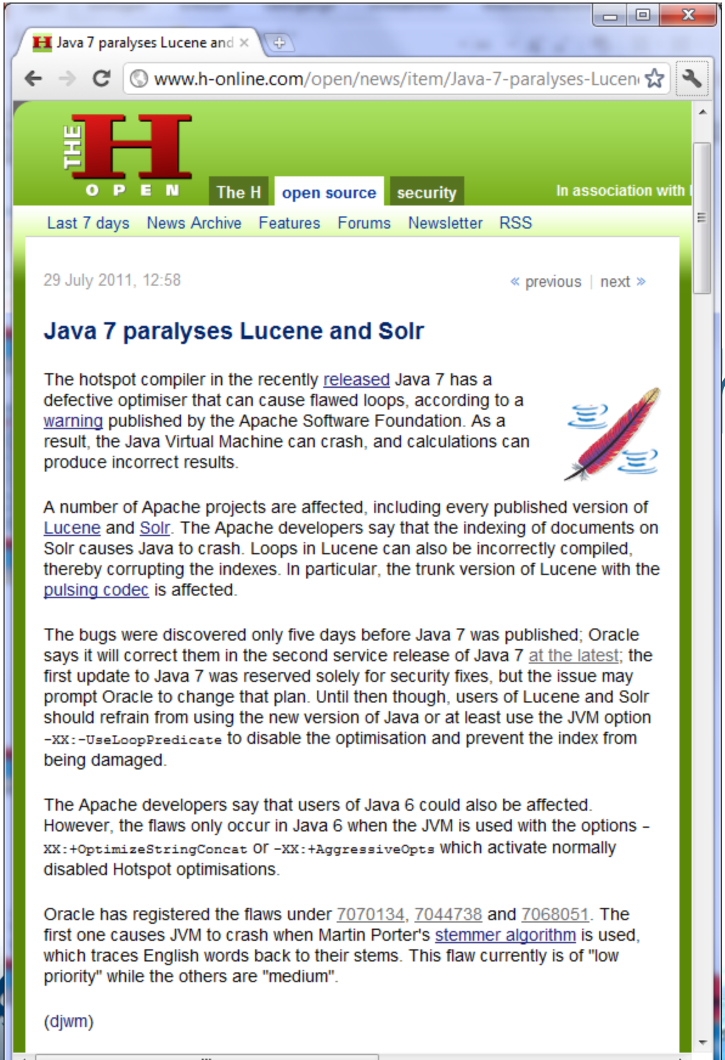
The Apache developers say that users of Java 6 could also be affected. However, the flaws only occur in Java 6 when the JVM is used with the options `-XX:+OptimizeStringConcat` or `-XX:+AggressiveOpts` which activate normally disabled Hotspot optimisations.

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)

nronc

, July



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.

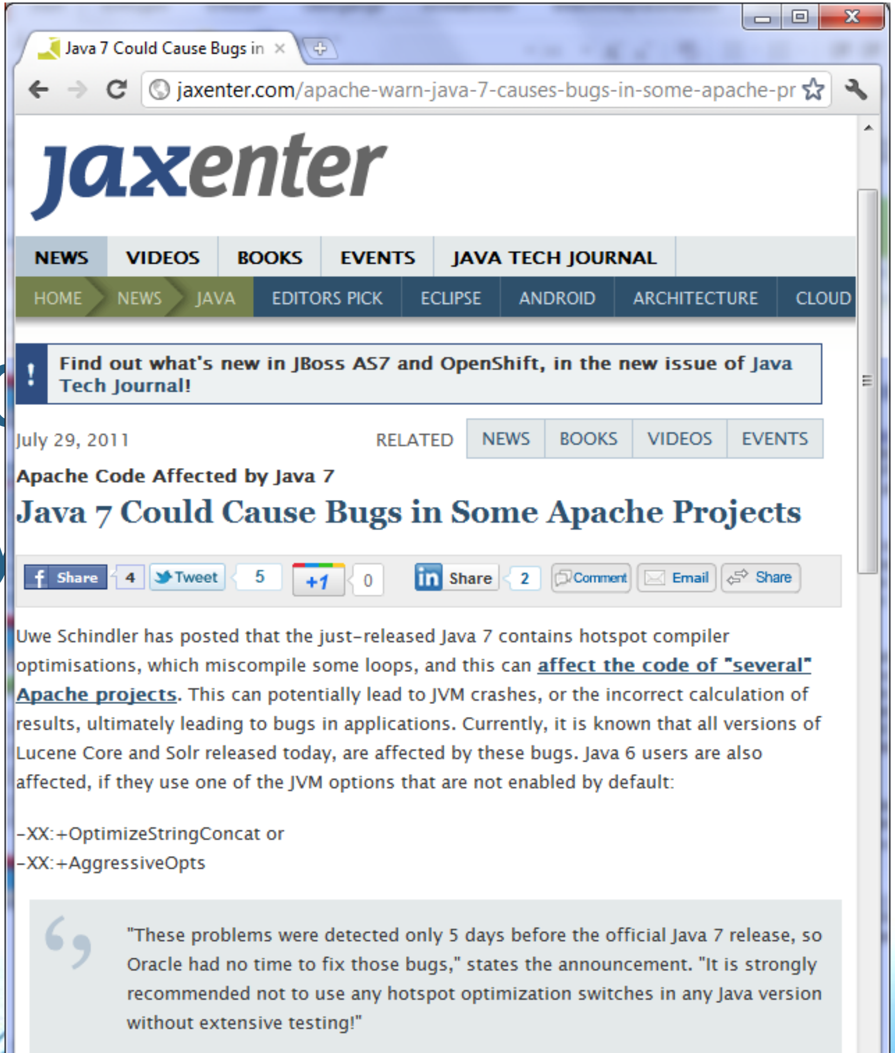
A number of Apache projects are affected, including every published version of Lucene and Solr. The Apache developers say that the indexing of documents on Solr causes Java to crash. Loops in Lucene can also be incorrectly compiled, thereby corrupting the indexes. In particular, the trunk version of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

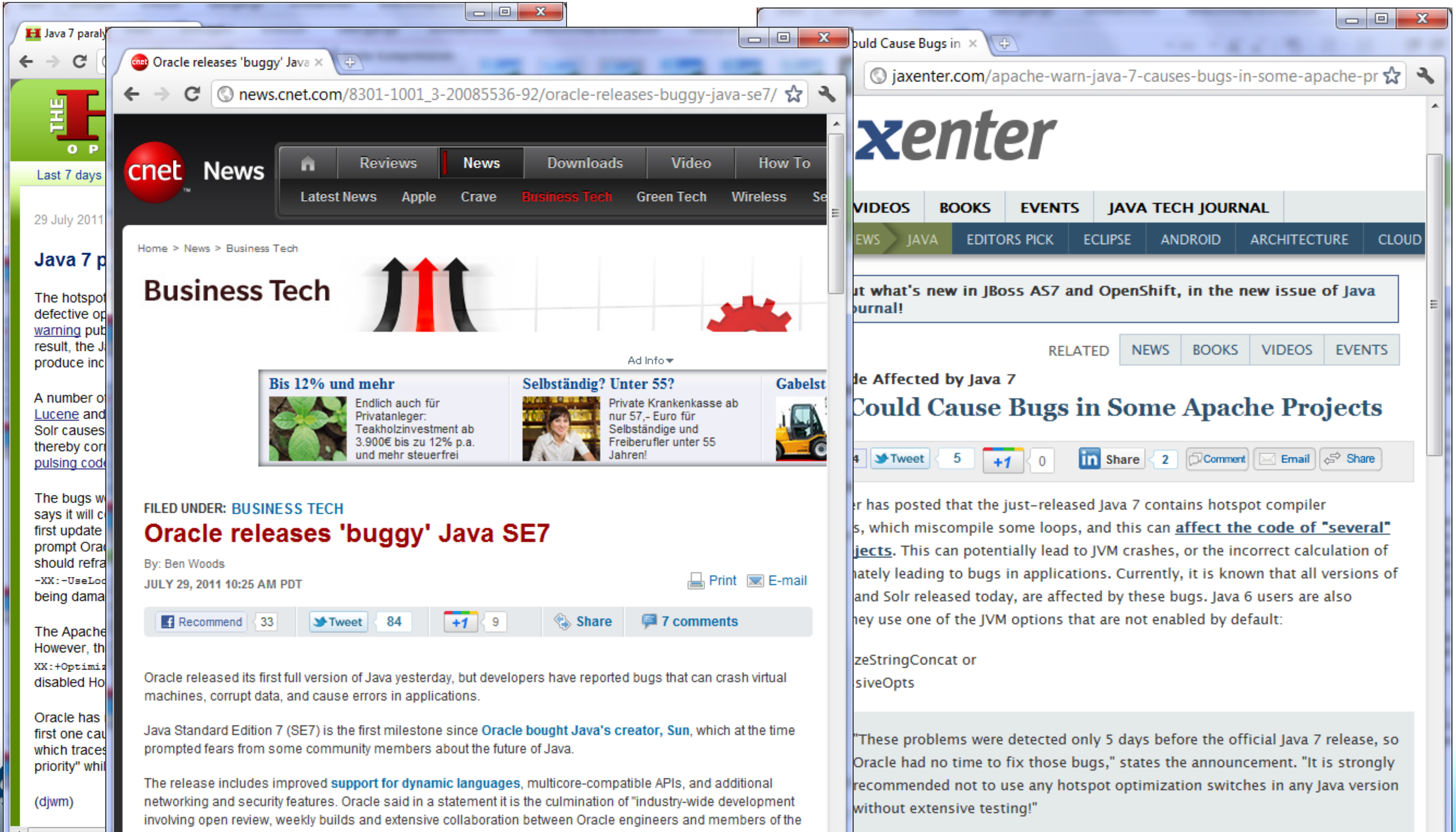
Java 7 Could Cause Bugs in Some Apache Projects

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!"



29 July 2011

Java 7 p

The hotspot
defective op
warning pub
result, the J
produce inc

A number of
Lucene and
Solr causes
thereby cor
pulsing code

The bugs w
says it will c
first update
prompt Ora
should refra
-XX:-UseLoo
being dama

The Apache
However, th
XX:+Optimiz
disabled Ho

Oracle has
first one cau
which traces
priority" whi

(djwm)

Oracle releases 'buggy' Java
news.cnet.com/8301-1001_3-20085536-92/oracle-releases-buggy-java-se7/
cnet News
Reviews
News
Downloads
Video
How To
Latest News
Apple
Crave
Business Tech
Green Tech
Wireless
Se
Home > News > Business Tech
Business Tech
Ad Info

Bis 12% und mehr
Endlich auch für Privatanleger: Teakholzinvestment ab 3.900€ bis zu 12% p.a. und mehr steuerfrei

Selbständig? Unter 55?
Private Krankenkasse ab nur 57,- Euro für Selbständige und Freiberufler unter 55 Jahren!

Gabelst

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

ut what's new in JBoss AS7 and OpenShift, in the new issue of Java
ournal!

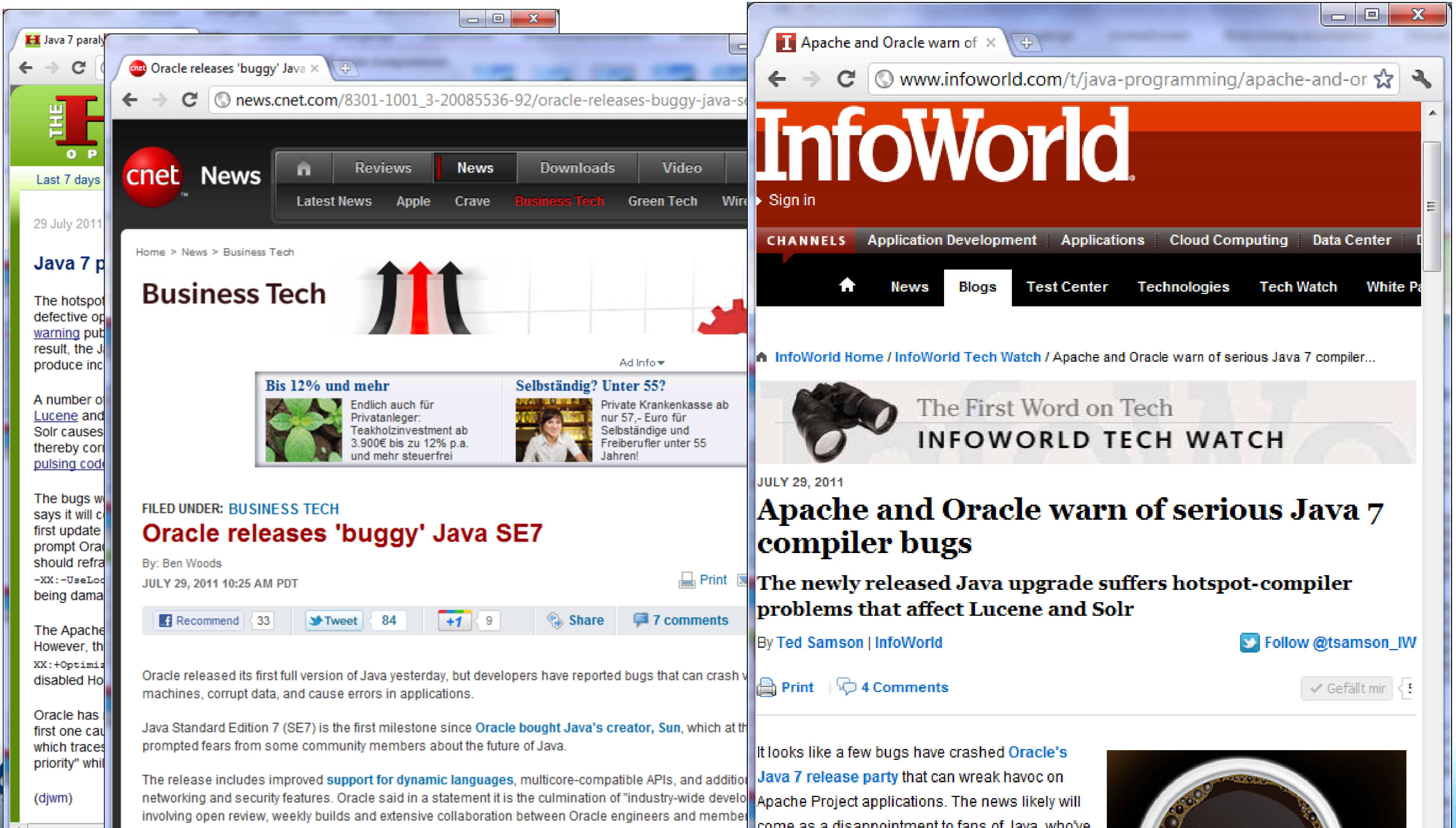
de Affected by Java 7

Could Cause Bugs in Some Apache Projects

r has posted that the just-released Java 7 contains hotspot compiler
s, which miscompile some loops, and this can affect the code of "several"
jects. This can potentially lead to JVM crashes, or the incorrect calculation of
ately leading to bugs in applications. Currently, it is known that all versions of
and Solr released today, are affected by these bugs. Java 6 users are also
hey use one of the JVM options that are not enabled by default:

zeStringConcat or
siveOpts

"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!"



Java 7 paraly

Last 7 days

29 July 2011

Java 7 p

The hotspot
defective op
warning pub
result, the J
produce inc

A number o
Lucene and
Solr causes
thereby cor
pulsing cod

The bugs w
says it will c
first update
prompt Ora
should refra
-XX:-UseLo
being dama

The Apache
However, th
XX:+Optimi
disabled Ho

Oracle has
first one ca
which trace
priority" whi

(djwm)

Oracle releases 'buggy' Java

news.cnet.com/8301-1001_3-20085536-92/oracle-releases-buggy-java-s

Home > News > Business Tech

Business Tech

Ad Info

Bis 12% und mehr
Endlich auch für Privatanleger: Teakholzinvestment ab 3.900€ bis zu 12% p.a. und mehr steuerfrei

Selbständig? Unter 55?
Private Krankenkasse ab nur 57,- Euro für Selbständige und Freiberufler unter 55 Jahren!

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
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 th
prompted fears from some community members about the future of Java.

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

Apache and Oracle warn of

www.infoworld.com/t/java-programming/apache-and-or

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

The First Word on Tech
INFOWORLD TECH WATCH

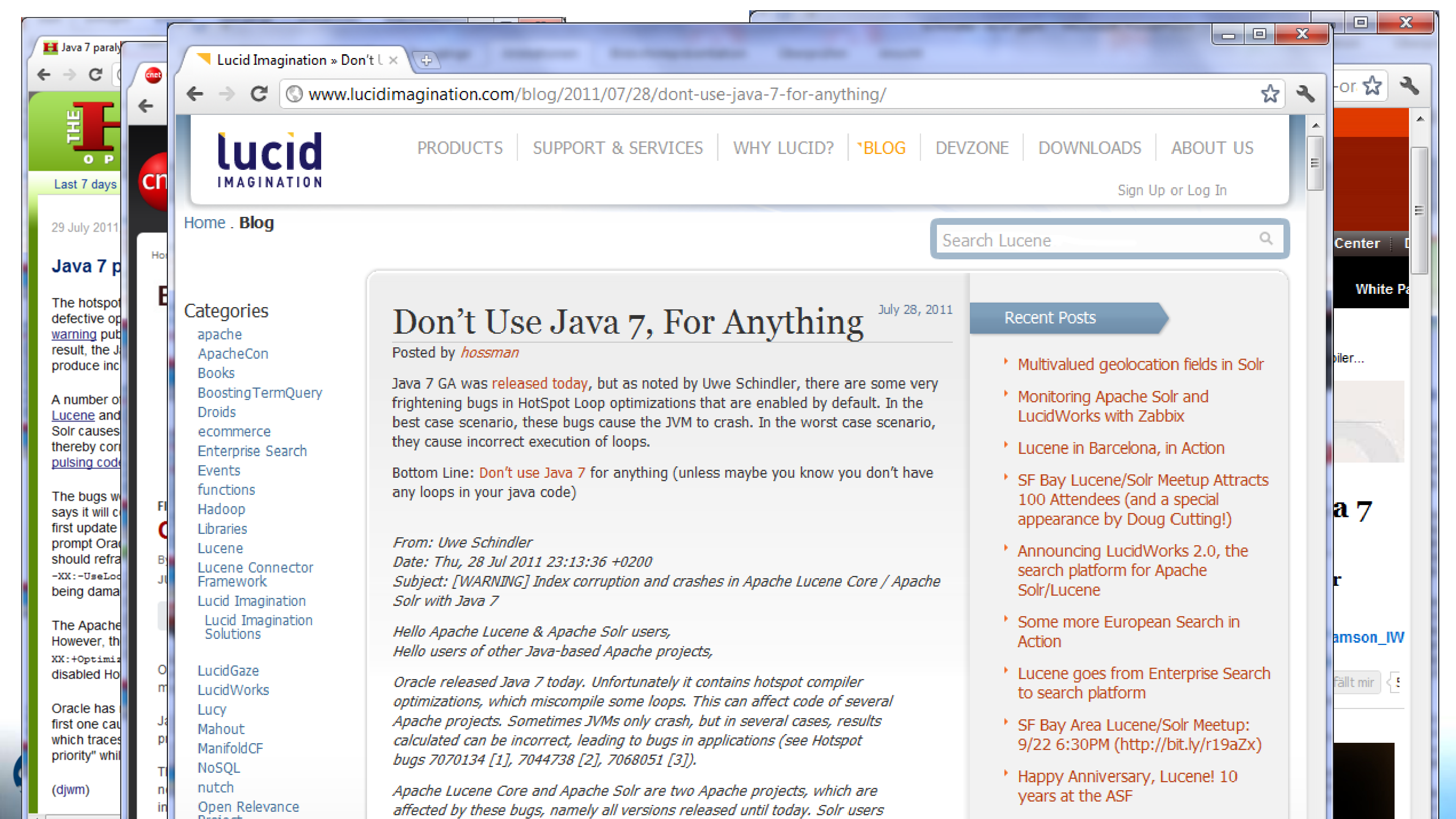
JULY 29, 2011

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

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



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

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

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

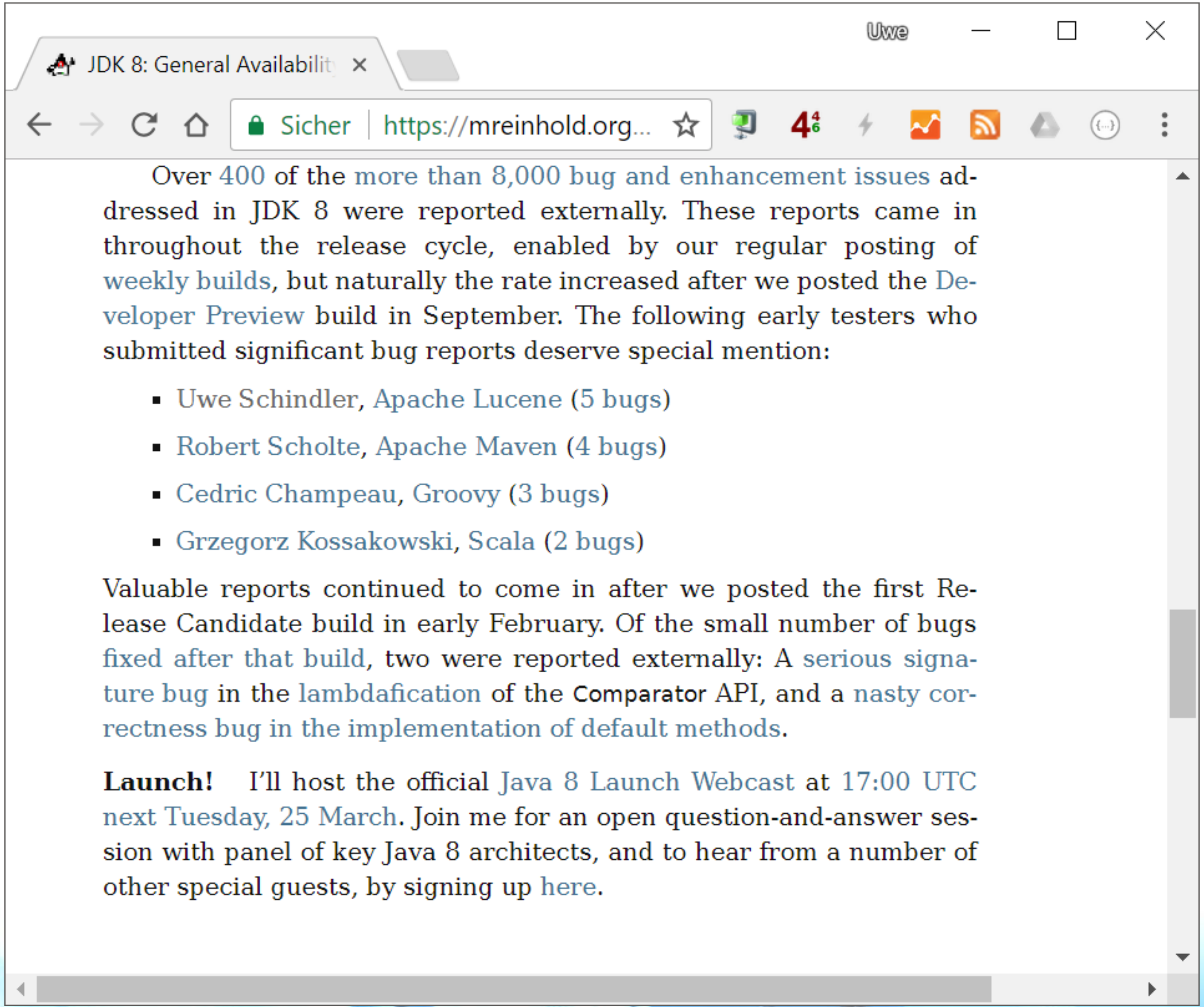
Reaction

Oracle *(Rory O'Donnell)* contacted Lucene PMC.

Weekly preview builds.

Other Open Source projects started to test with preview builds of Java 8 – and later **Java 9.**

Easy and fast bug reporting!



Over 400 of the more than 8,000 bug and enhancement issues addressed in JDK 8 were reported externally. These reports came in throughout the release cycle, enabled by our regular posting of weekly builds, but naturally the rate increased after we posted the Developer Preview build in September. The following early testers who submitted significant bug reports deserve special mention:

- Uwe Schindler, Apache Lucene (5 bugs)
- Robert Scholte, Apache Maven (4 bugs)
- Cedric Champeau, Groovy (3 bugs)
- Grzegorz Kossakowski, Scala (2 bugs)

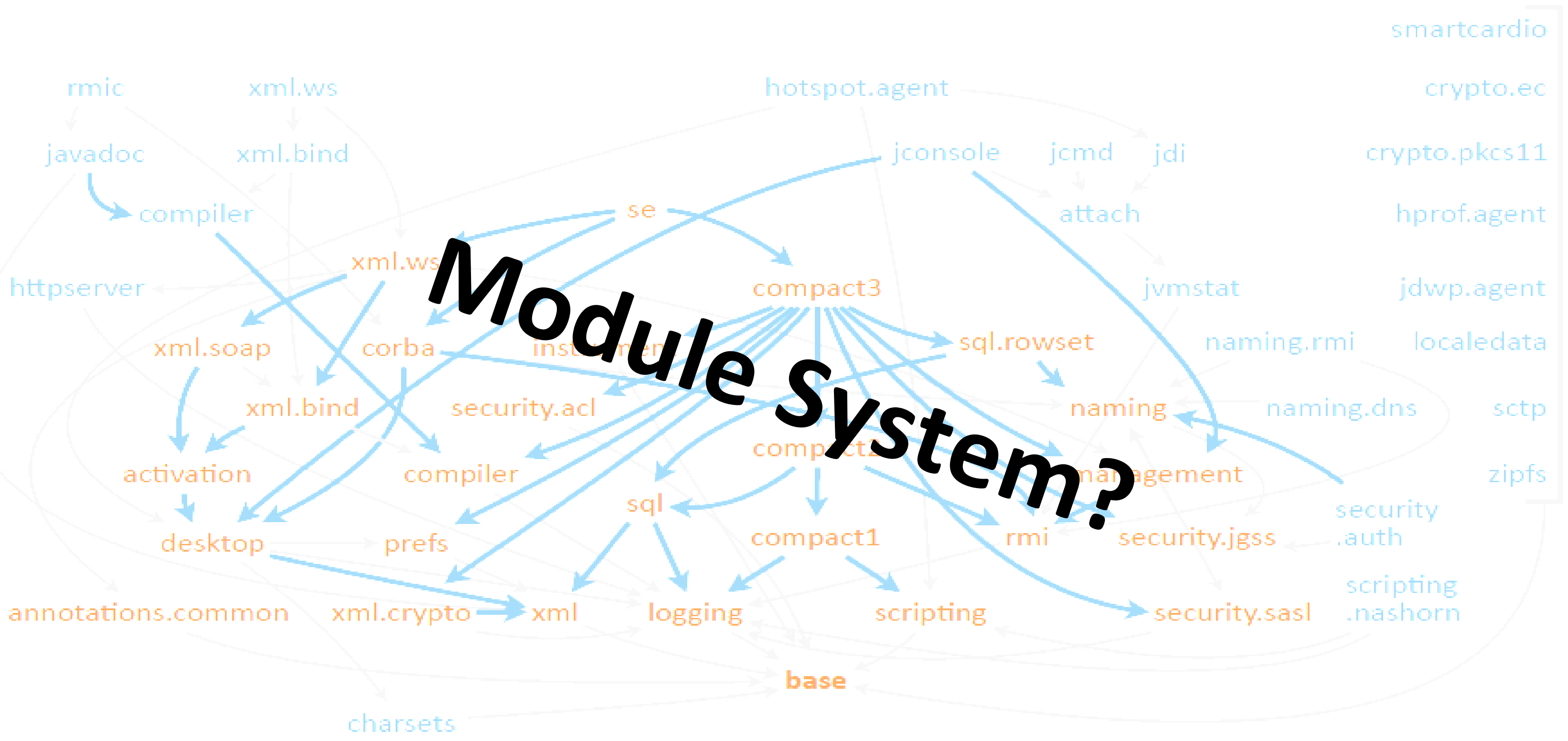
Valuable reports continued to come in after we posted the first Release Candidate build in early February. Of the small number of bugs fixed after that build, two were reported externally: A serious signature bug in the lambdafication of the `Comparator` API, and a nasty correctness bug in the implementation of default methods.

Launch! I'll host the official Java 8 Launch Webcast at 17:00 UTC next Tuesday, 25 March. Join me for an open question-and-answer session with panel of key Java 8 architects, and to hear from a number of other special guests, by signing up here.

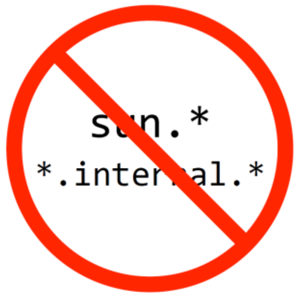
Going forward...

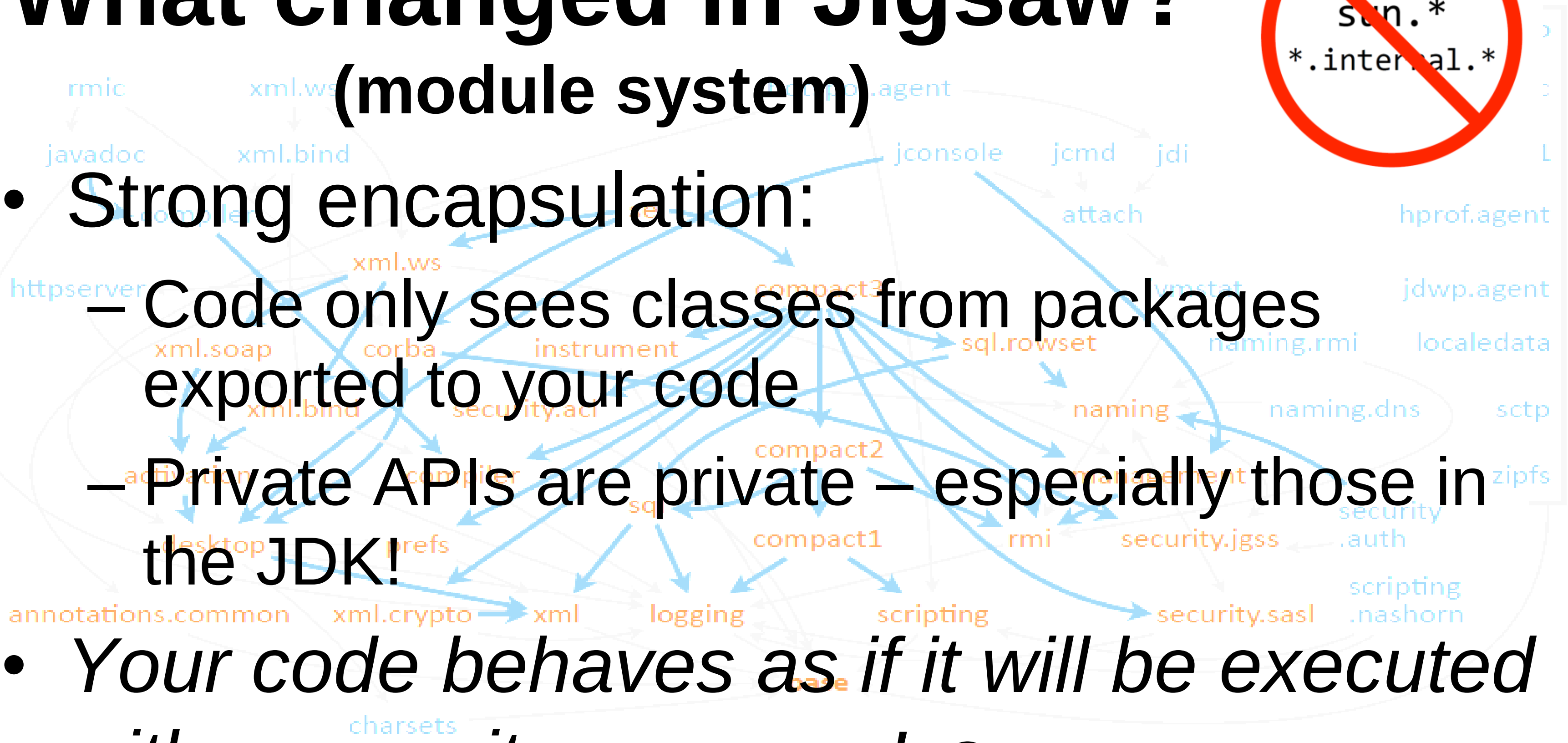
Java 9 and Apache Lucene

Module System?



What changed in Jigsaw?
(module system)





- Strong encapsulation:
  - Code only sees classes from packages exported to your code
  - Private APIs are private – especially those in the JDK!
- *Your code behaves as if it will be executed with a security manager!* ☺

What else is wrong with Jigsaw?

#ReflectiveAccessToNonExportedTypes

#AwkwardStrongEncapsulation

- New since build 148 of Java 9
- Prevents reflective access to any class from Java runtime

What else is wrong with Jigsaw?

#AwkwardStrongEncapsulation: A non-public element of an exported package can still be accessed via the `AccessibleObject::setAccessible` method of the core reflection API. The only way to strongly encapsulate such an element is to move it to a non-exported package. This makes it awkward, at best, to encapsulate the internals of a package that defines a public API.

from Java runtime

What else is wrong with Jigsaw?

#ReflectiveAccessToNonExportedTypes

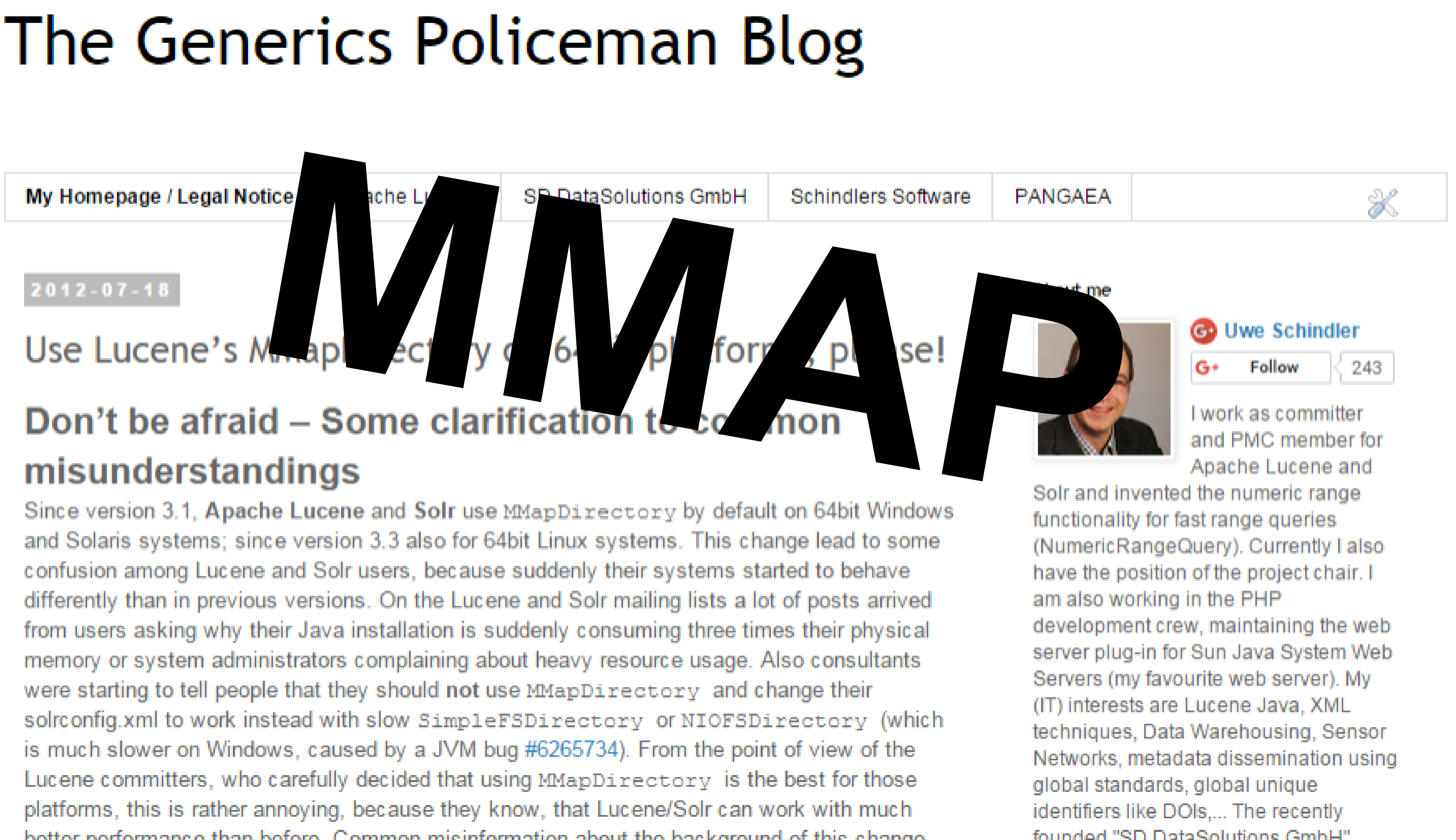
#AwkwardStrongEncapsulation

- New since build 148 of Java 9
- Prevents reflective access to any class from Java runtime

Unsafe, Byte Buffers & Co.

Undocumented APIs?

MMAP



The Generics Policeman Blog

My Homepage / Legal Notice | ache L | SD DataSolutions GmbH | Schindlers Software | PANGAEA

2012-07-18

Use Lucene's M ap ect y o 6 pl for , p se!

Don't be afraid – Some clarification to co mon misunderstandings

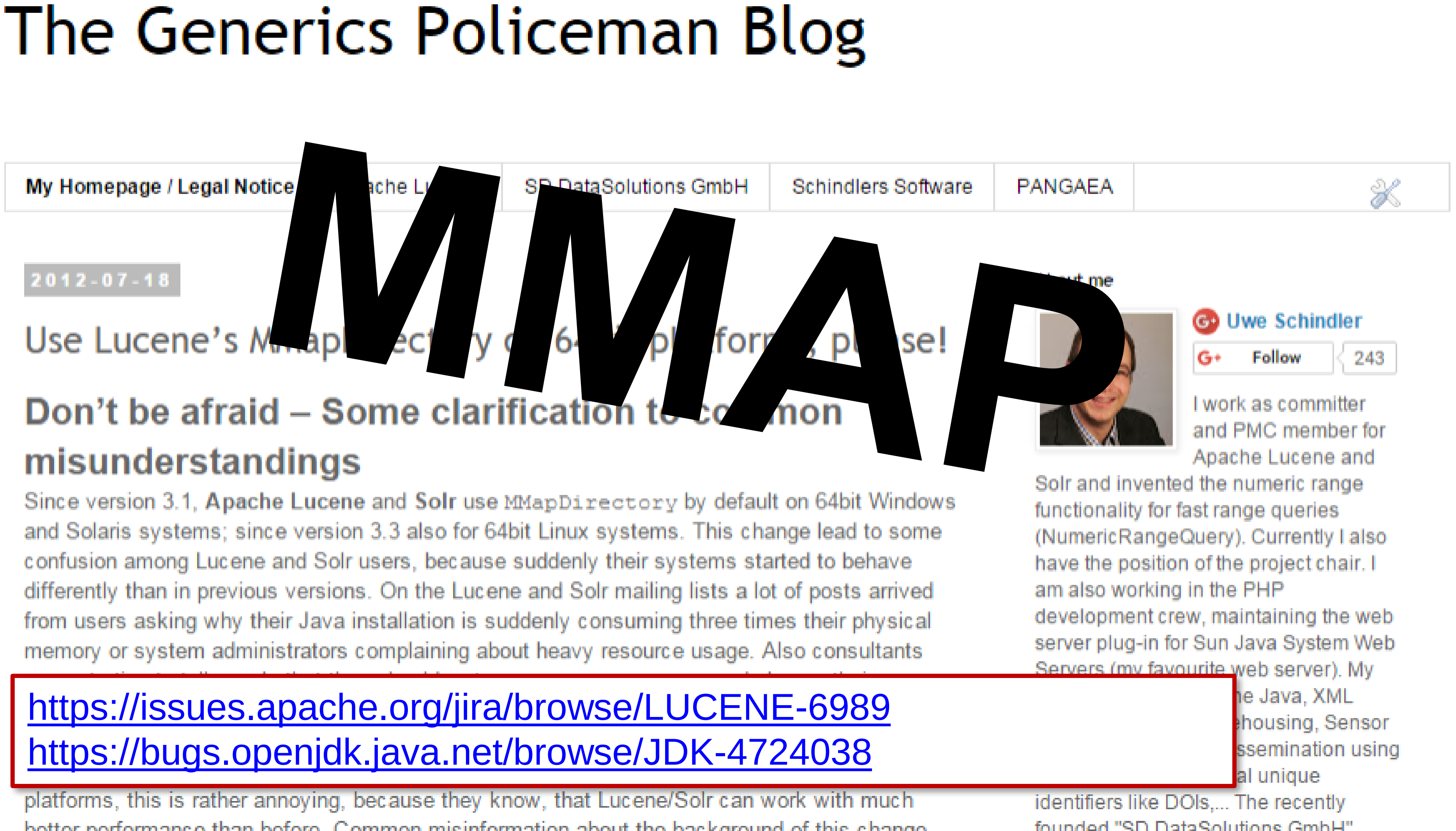
Since version 3.1, **Apache Lucene** and **Solr** use `MMapDirectory` by default on 64bit Windows and Solaris systems; since version 3.3 also for 64bit Linux systems. This change lead to some confusion among Lucene and Solr users, because suddenly their systems started to behave differently than in previous versions. On the Lucene and Solr mailing lists a lot of posts arrived from users asking why their Java installation is suddenly consuming three times their physical memory or system administrators complaining about heavy resource usage. Also consultants were starting to tell people that they should **not** use `MMapDirectory` and change their solrconfig.xml to work instead with slow `SimpleFSDirectory` or `NIOFSDirectory` (which is much slower on Windows, caused by a JVM bug #6265734). From the point of view of the Lucene committers, who carefully decided that using `MMapDirectory` is the best for those platforms, this is rather annoying, because they know, that Lucene/Solr can work with much better performance than before. Common misinformation about the background of this change

ut me

Uwe Schindler

Follow 243

I work as committer and PMC member for Apache Lucene and Solr and invented the numeric range functionality for fast range queries (NumericRangeQuery). Currently I also have the position of the project chair. I am also working in the PHP development crew, maintaining the web server plug-in for Sun Java System Web Servers (my favourite web server). My (IT) interests are Lucene Java, XML techniques, Data Warehousing, Sensor Networks, metadata dissemination using global standards, global unique identifiers like DOIs,... The recently founded "SD DataSolutions GmbH"



The Generics Policeman Blog

My Homepage / Legal Notice | ache L | SD DataSolutions GmbH | Schindlers Software | PANGAEA

MMAP

2012-07-18

Use Lucene's M ap ect y o 64 pl for , p se!

Don't be afraid – Some clarification to co mon misunderstandings

Since version 3.1, **Apache Lucene** and **Solr** use `MMapDirectory` by default on 64bit Windows and Solaris systems; since version 3.3 also for 64bit Linux systems. This change lead to some confusion among Lucene and Solr users, because suddenly their systems started to behave differently than in previous versions. On the Lucene and Solr mailing lists a lot of posts arrived from users asking why their Java installation is suddenly consuming three times their physical memory or system administrators complaining about heavy resource usage. Also consultants

platforms, this is rather annoying, because they know, that Lucene/Solr can work with much better performance than before. Common misinformation about the background of this change

ut me

Uwe Schindler

Follow 243

I work as committer and PMC member for Apache Lucene and Solr and invented the numeric range functionality for fast range queries (NumericRangeQuery). Currently I also have the position of the project chair. I am also working in the PHP development crew, maintaining the web server plug-in for Sun Java System Web Servers (my favourite web server). My

e Java, XML

ehousing, Sensor

ssemination using

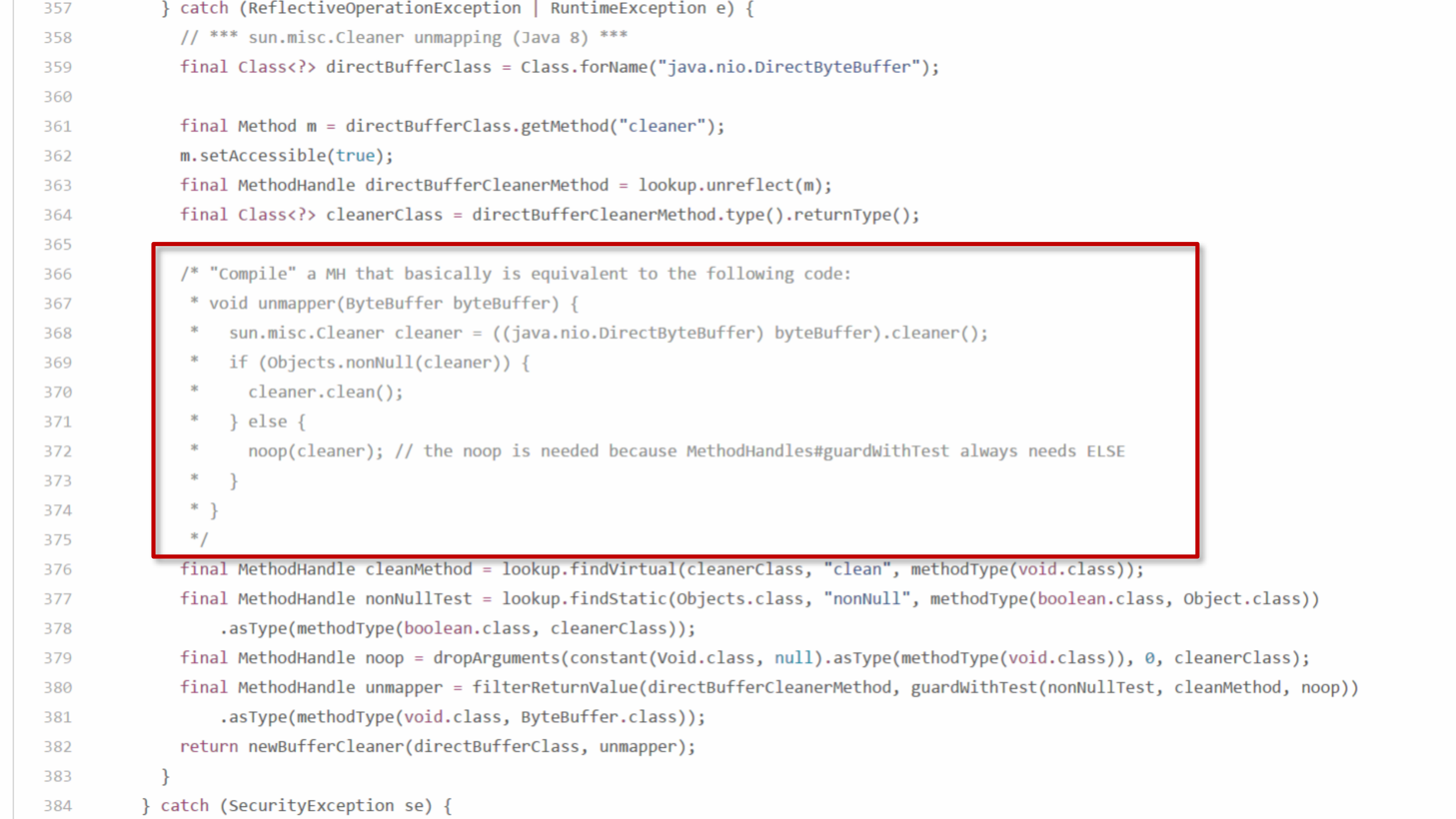
al unique

identifiers like DOIs,... The recently founded "SD DataSolutions GmbH"

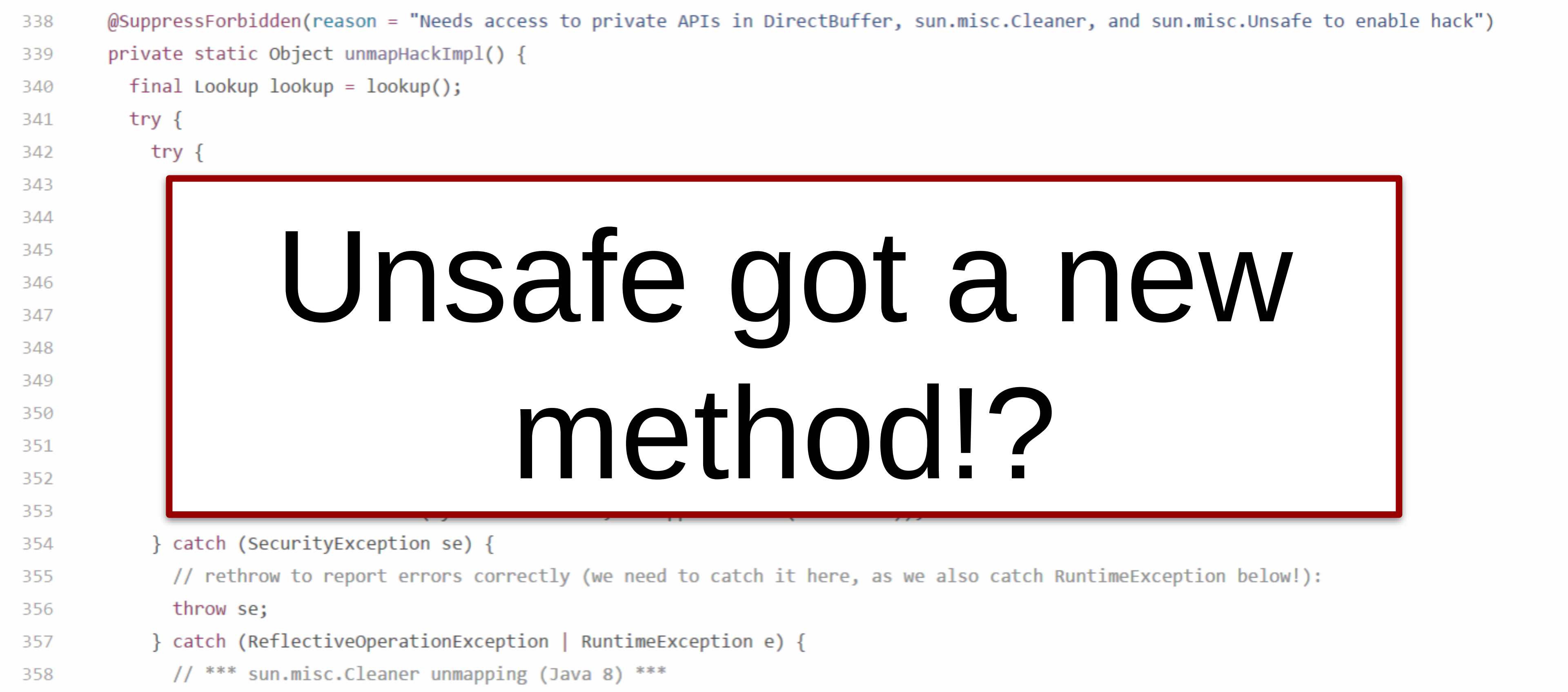
https://issues.apache.org/jira/browse/LUCENE-6989

https://bugs.openjdk.java.net/browse/JDK-4724038

```java
          } catch (ReflectiveOperationException | RuntimeException e) {
            // *** sun.misc.Cleaner unmapping (Java 8) ***
            final Class<?> directBufferClass = Class.forName("java.nio.DirectByteBuffer");

            final Method m = directBufferClass.getMethod("cleaner");
            m.setAccessible(true);
            final MethodHandle directBufferCleanerMethod = lookup.unreflect(m);
            final Class<?> cleanerClass = directBufferCleanerMethod.type().returnType();

            /* "Compile" a MH that basically is equivalent to the following code:
             * void unmapper(ByteBuffer byteBuffer) {
             *   sun.misc.Cleaner cleaner = ((java.nio.DirectByteBuffer) byteBuffer).cleaner();
             *   if (Objects.nonNull(cleaner)) {
             *     cleaner.clean();
             *   } else {
             *     noop(cleaner); // the noop is needed because MethodHandles#guardWithTest always needs ELSE
             *   }
             * }
             */
            final MethodHandle cleanMethod = lookup.findVirtual(cleanerClass, "clean", methodType(void.class));
            final MethodHandle nonNullTest = lookup.findStatic(Objects.class, "nonNull", methodType(boolean.class, Object.class))
                .asType(methodType(boolean.class, cleanerClass));
            final MethodHandle noop = dropArguments(constant(Void.class, null).asType(methodType(void.class)), 0, cleanerClass);
            final MethodHandle unmapper = filterReturnValue(directBufferCleanerMethod, guardWithTest(nonNullTest, cleanMethod, noop))
                .asType(methodType(void.class, ByteBuffer.class));
            return newBufferCleaner(directBufferClass, unmapper);
          }
        } catch (SecurityException se) {
```


```java
  } catch (ReflectiveOperationException | RuntimeException e) {
    // *** sun.misc.Cleaner unmapping (Java 8) ***
    final Class<?> directBufferClass = Class.forName("java.nio.DirectByteBuffer");

    final Method m = directBufferClass.getMethod("cleaner");
    m.setAccessible(true);
    final MethodHandle directBufferCleanerMethod = lookup.unreflect(m);
    final Class<?> cleanerClass = directBufferCleanerMethod.type().returnType();

    /* "Compile" a MH that basically is equivalent to the following code:
     * void unmapper(ByteBuffer byteBuffer) {
     *   sun.misc.Cleaner cleaner = ((java.nio.DirectByteBuffer) byteBuffer).cleaner();
     *   if (Objects.nonNull(cleaner)) {
     *     cleaner.clean();
     *   } else {
     *     noop(cleaner); // the noop is needed because MethodHandles#guardWithTest always needs ELSE
     *   }
     * }
     */
    final MethodHandle cleanMethod = lookup.findVirtual(cleanerClass, "clean", methodType(void.class));
    final MethodHandle nonNullTest = lookup.findStatic(Objects.class, "nonNull", methodType(boolean.class, Object.class))
        .asType(methodType(boolean.class, cleanerClass));
    final MethodHandle noop = dropArguments(constant(Void.class, null).asType(methodType(void.class)), 0, cleanerClass);
    final MethodHandle unmapper = filterReturnValue(directBufferCleanerMethod, guardWithTest(nonNullTest, cleanMethod, noop))
        .asType(methodType(void.class, ByteBuffer.class));
    return newBufferCleaner(directBufferClass, unmapper);
  }
} catch (SecurityException se) {
      } catch (ReflectiveOperationException | RuntimeException e) {
        // *** sun.misc.Cleaner unmapping (Java 8) ***
        final Class<?> directBufferClass = Class.forName("java.nio.DirectByteBuffer");

        final Method m = directBufferClass.getMethod("cleaner");
        m.setAccessible(true);
        final MethodHandle directBufferCleanerMethod = lookup.unreflect(m);
        final Class<?> cleanerClass = directBufferCleanerMethod.type().returnType();

        /* "Compile" a MH that basically is equivalent to the following code:
         * void unmapper(ByteBuffer byteBuffer) {
         *   sun.misc.Cleaner cleaner = ((java.nio.DirectByteBuffer) byteBuffer).cleaner();
         *   if (Objects.nonNull(cleaner)) {
         *     cleaner.clean();
         *   } else {
         *     noop(cleaner); // the noop is needed because MethodHandles#guardWithTest always needs ELSE
         *   }
         * }
         */
        final MethodHandle cleanMethod = lookup.findVirtual(cleanerClass, "clean", methodType(void.class));
        final MethodHandle nonNullTest = lookup.findStatic(Objects.class, "nonNull", methodType(boolean.class, Object.class))
            .asType(methodType(boolean.class, cleanerClass));
        final MethodHandle noop = dropArguments(constant(Void.class, null).asType(methodType(void.class)), 0, cleanerClass);
        final MethodHandle unmapper = filterReturnValue(directBufferCleanerMethod, guardWithTest(nonNullTest, cleanMethod, noop))
            .asType(methodType(void.class, ByteBuffer.class));
        return newBufferCleaner(directBufferClass, unmapper);
      }
    } catch (SecurityException se) {
```

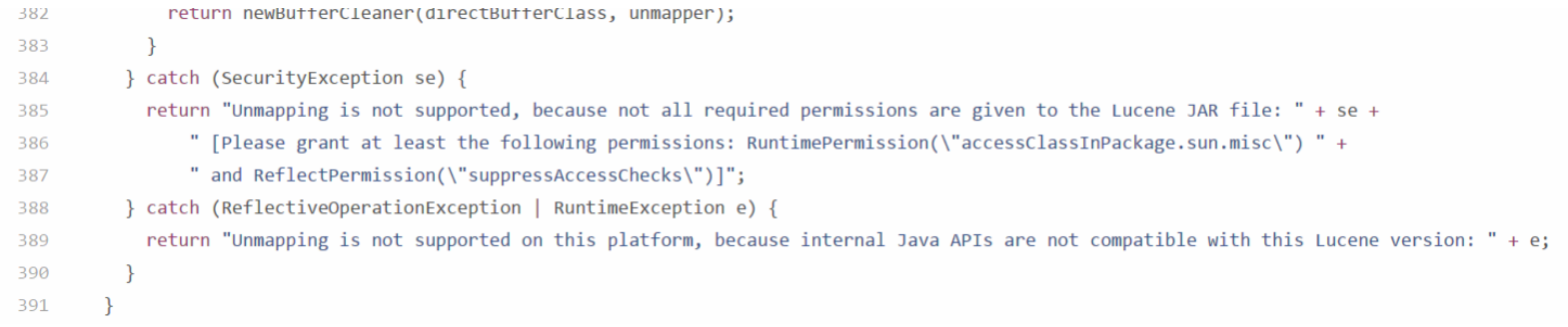
```java
  @SuppressForbidden(reason = "Needs access to private APIs in DirectBuffer, sun.misc.Cleaner, and sun.misc.Unsafe to enable hack")
  private static Object unmapHackImpl() {
    final Lookup lookup = lookup();
    try {
      try {
        // *** sun.misc.Unsafe unmapping (Java 9+) ***
        final Class<?> unsafeClass = Class.forName("sun.misc.Unsafe");
        // first check if Unsafe has the right method, otherwise we can give up
        // without doing any security critical stuff:
        final MethodHandle unmapper = lookup.findVirtual(unsafeClass, "invokeCleaner",
            methodType(void.class, ByteBuffer.class));
        // fetch the unsafe instance and bind it to the virtual MH:
        final Field f = unsafeClass.getDeclaredField("theUnsafe");
        f.setAccessible(true);
        final Object theUnsafe = f.get(null);
        return newBufferCleaner(ByteBuffer.class, unmapper.bindTo(theUnsafe));
      } catch (SecurityException se) {
        // rethrow to report errors correctly (we need to catch it here, as we also catch RuntimeException below!):
        throw se;
      } catch (ReflectiveOperationException | RuntimeException e) {
        // *** sun.misc.Cleaner unmapping (Java 8) ***
```



```
  @SuppressForbidden(reason = "Needs access to private APIs in DirectBuffer, sun.misc.Cleaner, and sun.misc.Unsafe to enable hack")
  private static Object unmapHackImpl() {
    final Lookup lookup = lookup();
    try {
      try {

      } catch (SecurityException se) {
        // rethrow to report errors correctly (we need to catch it here, as we also catch RuntimeException below!):
        throw se;
      } catch (ReflectiveOperationException | RuntimeException e) {
        // *** sun.misc.Cleaner unmapping (Java 8) ***
```

Unsafe got a new method!?

```java
  @SuppressForbidden(reason = "Needs access to private APIs in DirectBuffer, sun.misc.Cleaner, and sun.misc.Unsafe to enable hack")
  private static Object unmapHackImpl() {
    final Lookup lookup = lookup();
    try {
      try {
        // *** sun.misc.Unsafe unmapping (Java 9+) ***
        final Class<?> unsafeClass = Class.forName("sun.misc.Unsafe");
        // first check if Unsafe has the right method, otherwise we can give up
        // without doing any security critical stuff:
        final MethodHandle unmapper = lookup.findVirtual(unsafeClass, "invokeCleaner",
            methodType(void.class, ByteBuffer.class));
        // fetch the unsafe instance and bind it to the virtual MH:
        final Field f = unsafeClass.getDeclaredField("theUnsafe");
        f.setAccessible(true);
        final Object theUnsafe = f.get(null);
        return newBufferCleaner(ByteBuffer.class, unmapper.bindTo(theUnsafe));
      } catch (SecurityException se) {
        // rethrow to report errors correctly (we need to catch it here, as we also catch RuntimeException below!):
        throw se;
      } catch (ReflectiveOperationException | RuntimeException e) {
        // *** sun.misc.Cleaner unmapping (Java 8) ***
```



```
      return newBufferCleaner(directBufferClass, unmapper);
    }
  } catch (SecurityException se) {
    return "Unmapping is not supported, because not all required permissions are given to the Lucene JAR file: " + se +
        " [Please grant at least the following permissions: RuntimePermission(\"accessClassInPackage.sun.misc\") " +
        " and ReflectPermission(\"suppressAccessChecks\")]";
  } catch (ReflectiveOperationException | RuntimeException e) {
    return "Unmapping is not supported on this platform, because internal Java APIs are not compatible with this Lucene version: " + e;
  }
}
```

Compact Strings & Co.

Other Changes

Compact Strings

Java 9 internally stores strings in compact form, if they only contain ISO-8859-1 characters

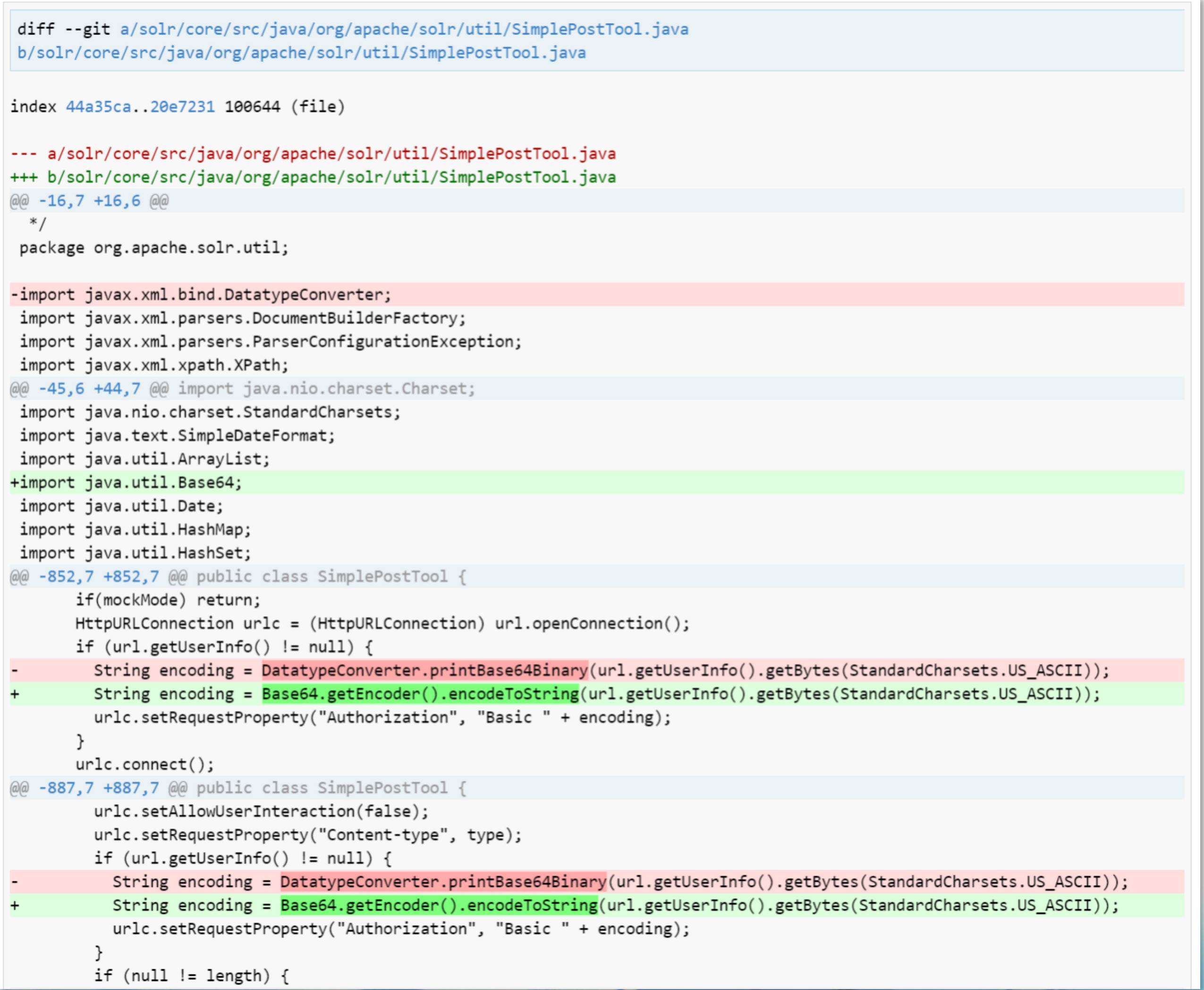
Indyfied String Concat

```
"Hallo " + 123 + ' ' + object +
" is a concatted string";
```

- Java 1.0 to 1.8: a chain of `StringBuilder.appends()`
- Java 9: Invokedynamic with `StringConcatFactory`:

```
    String concat(String,int,char,Object,String)
```

Just a funny detail...



```diff
diff --git a/solr/core/src/java/org/apache/solr/util/SimplePostTool.java
b/solr/core/src/java/org/apache/solr/util/SimplePostTool.java

index 44a35ca..20e7231 100644 (file)

--- a/solr/core/src/java/org/apache/solr/util/SimplePostTool.java
+++ b/solr/core/src/java/org/apache/solr/util/SimplePostTool.java
@@ -16,7 +16,6 @@
   */
 package org.apache.solr.util;
 
-import javax.xml.bind.DatatypeConverter;
 import javax.xml.parsers.DocumentBuilderFactory;
 import javax.xml.parsers.ParserConfigurationException;
 import javax.xml.xpath.XPath;
@@ -45,6 +44,7 @@ import java.nio.charset.Charset;
 import java.nio.charset.StandardCharsets;
 import java.text.SimpleDateFormat;
 import java.util.ArrayList;
+import java.util.Base64;
 import java.util.Date;
 import java.util.HashMap;
 import java.util.HashSet;
@@ -852,7 +852,7 @@ public class SimplePostTool {
       if(mockMode) return;
       HttpURLConnection urlc = (HttpURLConnection) url.openConnection();
       if (url.getUserInfo() != null) {
-        String encoding = DatatypeConverter.printBase64Binary(url.getUserInfo().getBytes(StandardCharsets.US_ASCII));
+        String encoding = Base64.getEncoder().encodeToString(url.getUserInfo().getBytes(StandardCharsets.US_ASCII));
         urlc.setRequestProperty("Authorization", "Basic " + encoding);
       }
       urlc.connect();
@@ -887,7 +887,7 @@ public class SimplePostTool {
         urlc.setAllowUserInteraction(false);
         urlc.setRequestProperty("Content-type", type);
         if (url.getUserInfo() != null) {
-          String encoding = DatatypeConverter.printBase64Binary(url.getUserInfo().getBytes(StandardCharsets.US_ASCII));
+          String encoding = Base64.getEncoder().encodeToString(url.getUserInfo().getBytes(StandardCharsets.US_ASCII));
           urlc.setRequestProperty("Authorization", "Basic " + encoding);
         }
         if (null != length) {
```

Performance

Hotspot Changes

ByteBuffers

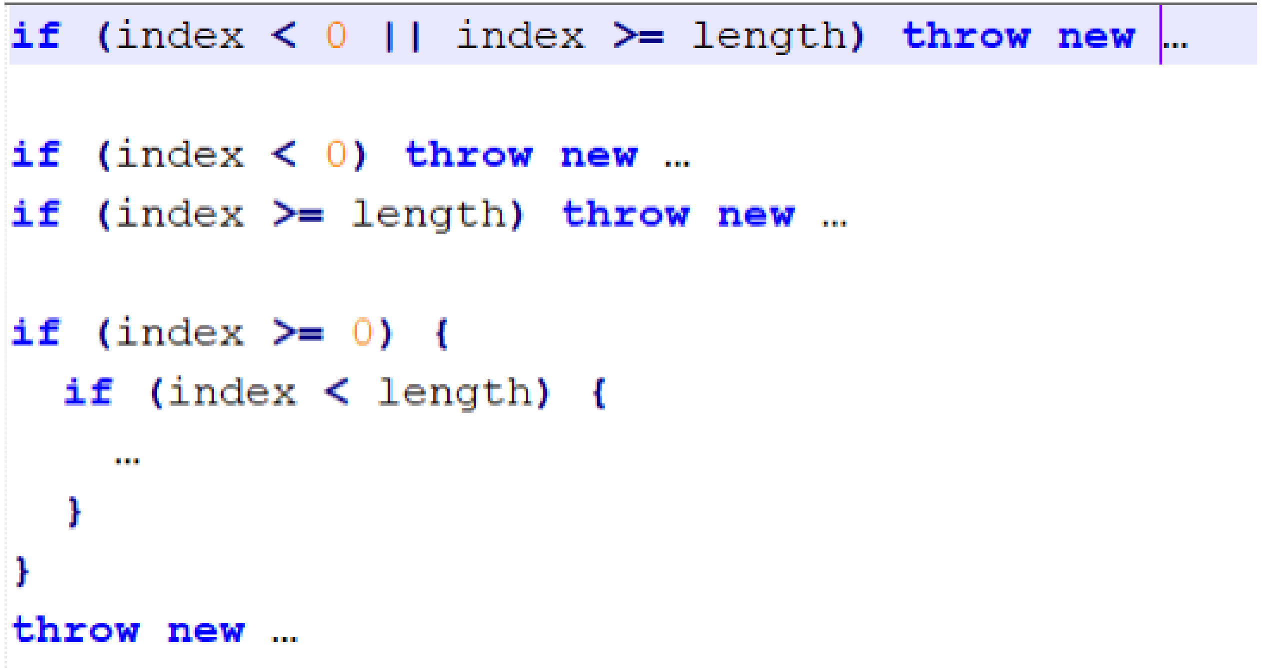
Same speed as arrays!

Intrinsics

`java.util.Objects` / `java.util.Arrays`
classes:

- Bounds checks
- Array comparisons (signed / unsigned)
- Array differences

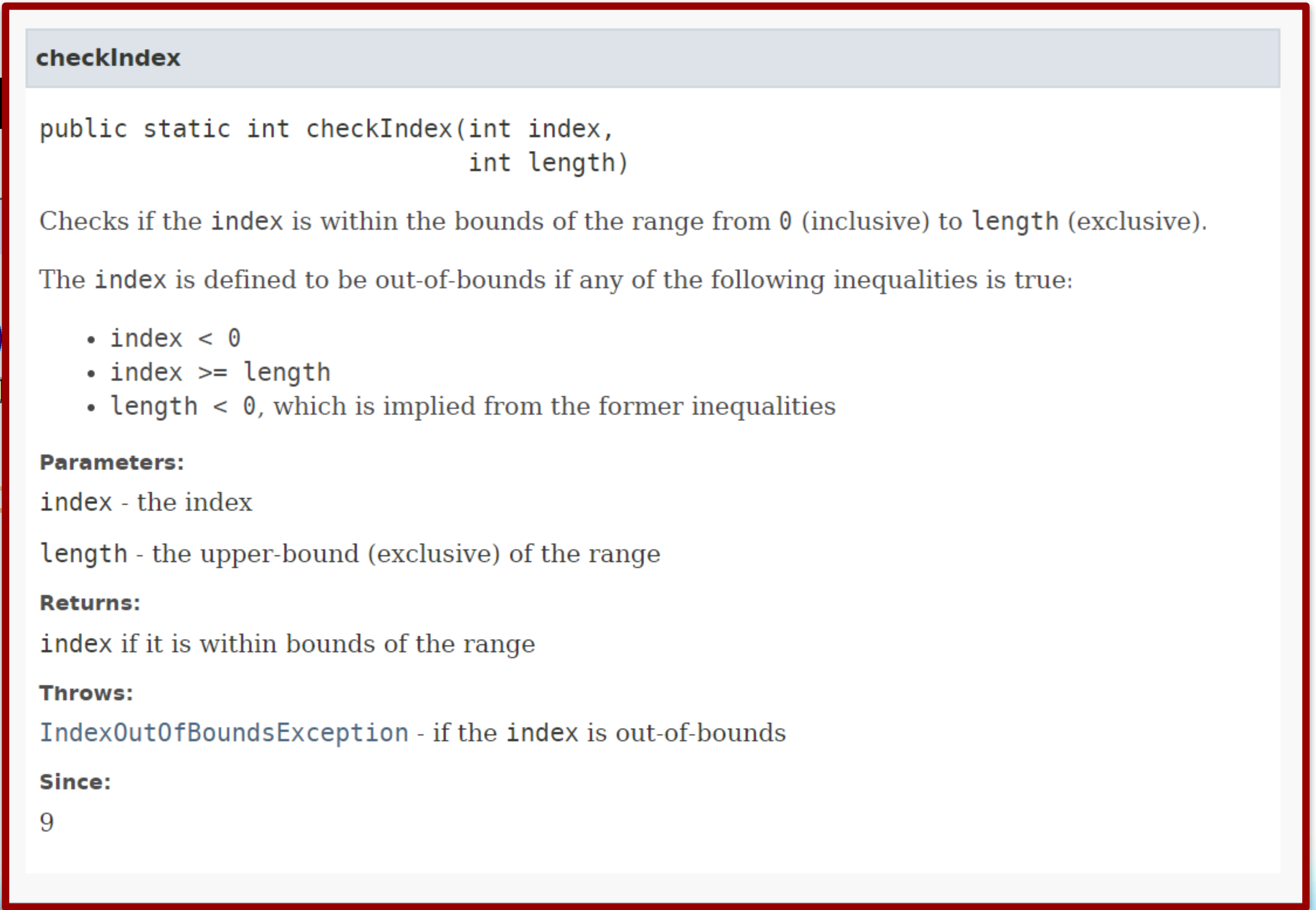
Know this type of code?



```
if (index < 0 || index >= length) throw new …

if (index < 0) throw new …
if (index >= length) throw new …

if (index >= 0) {
  if (index < length) {
    …
  }
}
throw new …
```

K

```
if (index < 0

if (index < 0)
if (index >= l

if (index >= (
  if (index <
    ...
  }
}
throw new ...
```



checkIndex

```
public static int checkIndex(int index,
                             int length)
```

Checks if the `index` is within the bounds of the range from `0` (inclusive) to `length` (exclusive).

The `index` is defined to be out-of-bounds if any of the following inequalities is true:

- `index < 0`
- `index >= length`
- `length < 0`, which is implied from the former inequalities

Parameters:

`index` - the index

`length` - the upper-bound (exclusive) of the range

Returns:

`index` if it is within bounds of the range

Throws:

`IndexOutOfBoundsException` - if the `index` is out-of-bounds

Since:

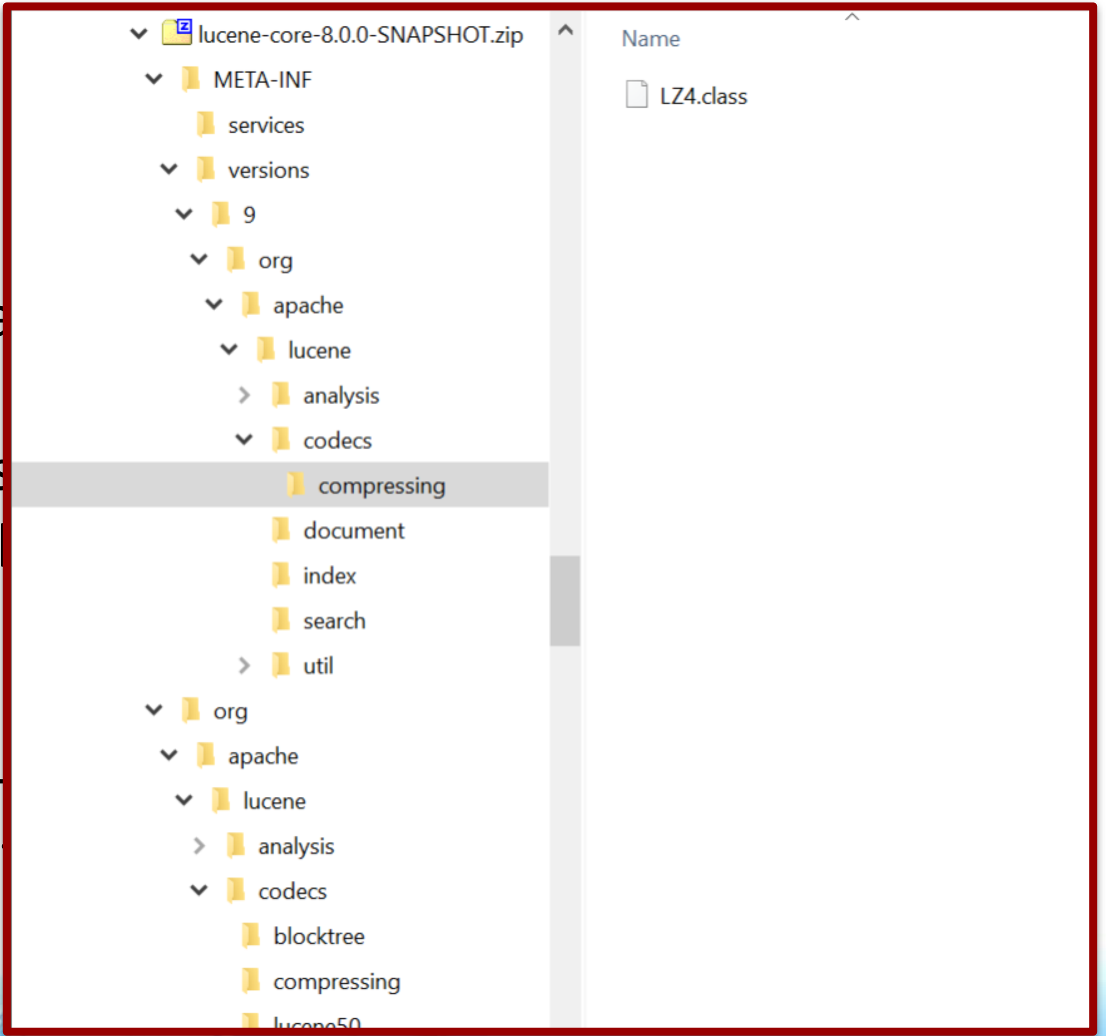
9

Solution: Multi-Release JAR (JEP 238)

- Lucene adds plain Java implementations of `java.util.Objects` and `java.util.Arrays` to own codebase (with exact same signatures)
- After compilation all class files are “patched” to use Java 9 signatures and stored in separate folder
- Builds **MR-JAR** with:
  - unmodified Java 8-compatible classes
  - Patched classes with Java 9 signatures in extra folder

Solution: Mu

- Lucene adds plain Java `java.util.Objec` codebase (with exact s
- After compilation all cl signatures and stored
- Builds **MR-JAR** with:
  - unmodified Java 8-
  - Patched classes wi



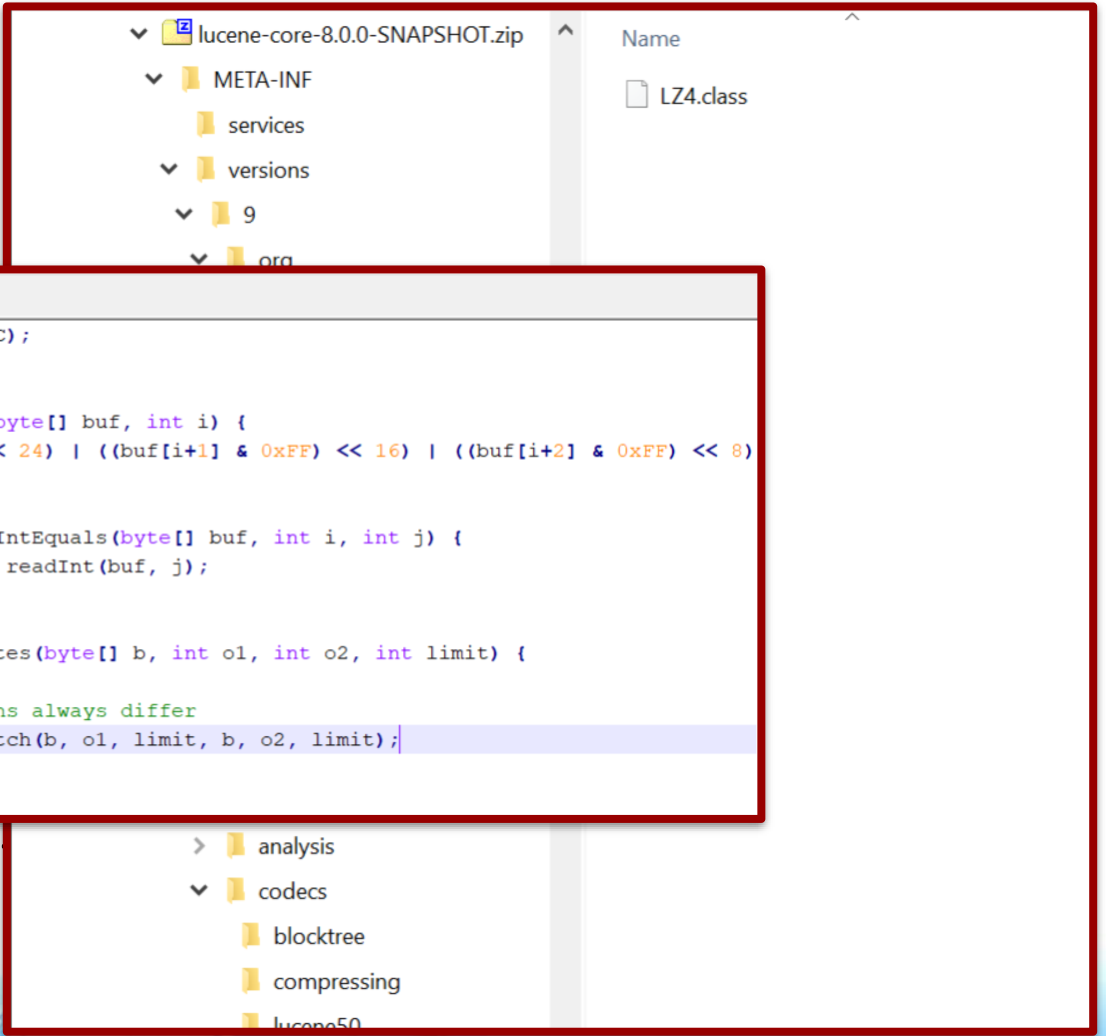
```
lucene-core-8.0.0-SNAPSHOT.zip
  META-INF
    services
    versions
      9
        org
          apache
            lucene
              analysis
              codecs
                compressing
              document
              index
              search
              util
  org
    apache
      lucene
        analysis
        codecs
          blocktree
          compressing
          lucene50
```

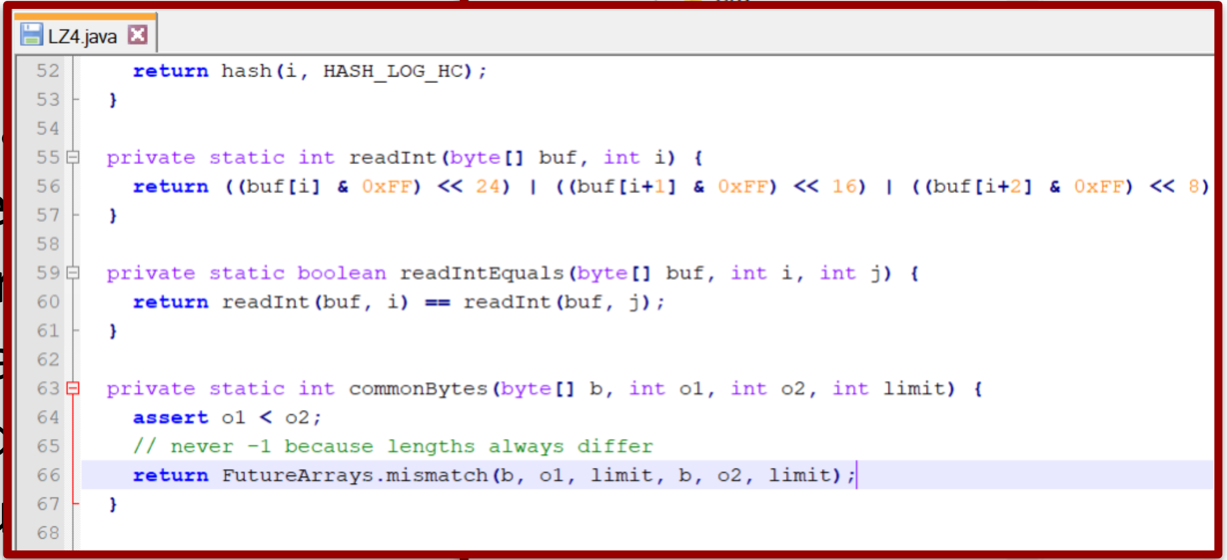
Name

LZ4.class

Solution: Mu

- Luce
  jav
  code
- After
  signa
- Build
  - u
  - Patched classes wi

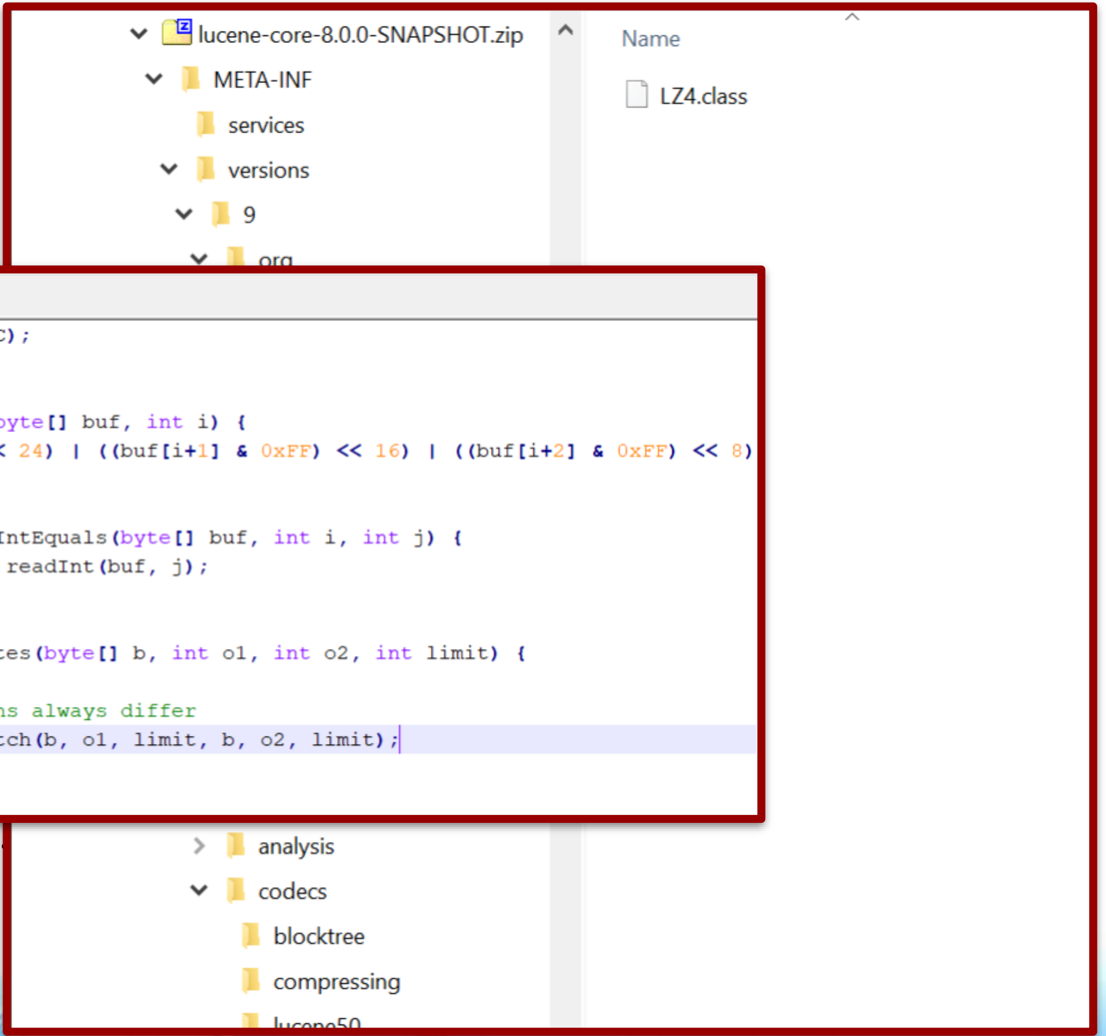


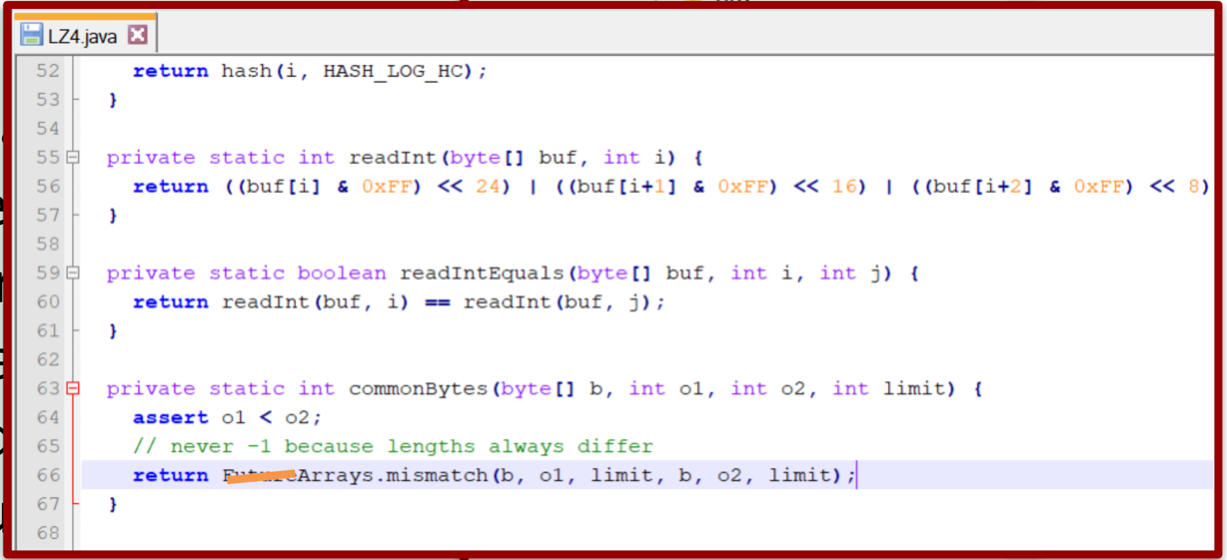


```
    return hash(i, HASH_LOG_HC);
  }

  private static int readInt(byte[] buf, int i) {
    return ((buf[i] & 0xFF) << 24) | ((buf[i+1] & 0xFF) << 16) | ((buf[i+2] & 0xFF) << 8)
  }

  private static boolean readIntEquals(byte[] buf, int i, int j) {
    return readInt(buf, i) == readInt(buf, j);
  }

  private static int commonBytes(byte[] b, int o1, int o2, int limit) {
    assert o1 < o2;
    // never -1 because lengths always differ
    return FutureArrays.mismatch(b, o1, limit, b, o2, limit);
  }

```

Solution: Mu

- Luce
  `jav`
  code
- After
  signa
- Build
  - u
  - Patched classes wi





```
    return hash(i, HASH_LOG_HC);
  }

  private static int readInt(byte[] buf, int i) {
    return ((buf[i] & 0xFF) << 24) | ((buf[i+1] & 0xFF) << 16) | ((buf[i+2] & 0xFF) << 8)
  }

  private static boolean readIntEquals(byte[] buf, int i, int j) {
    return readInt(buf, i) == readInt(buf, j);
  }

  private static int commonBytes(byte[] b, int o1, int o2, int limit) {
    assert o1 < o2;
    // never -1 because lengths always differ
    return ~~Future~~Arrays.mismatch(b, o1, limit, b, o2, limit);
  }

```

G1GC

Garbage Collector!

New Default Garbage Collector

- G1GC is now the default
  - Previously it was ParallelGC
  - No need to care in most cases, as Solr / Elasticsearch use a hardcoded default
- CMS collector deprecated!
  - Warning on start of process!
  - Migrate to G1GC?

New Default Garbage Collector

```
$ java -XX:+UseConcMarkSweepGC
Java HotSpot(TM) 64-Bit Server VM
warning: Option UseConcMarkSweepGC was
deprecated in version 9.0 and will likely
be removed in a future release.
```

- Warning on start of process!
- Migrate to G1GC?

New Default Garbage Collector

- G1GC is now the default
  - Previously it was ParallelGC
  - No need to care in most cases, as Solr / Elasticsearch use a hardcoded default
- CMS collector deprecated!
  - Warning on start of process!
  - Migrate to G1GC?

Improvements?

Why update your cluster to Java 9 or 10 ?

Lower GC pause times with G1GC

Security

More security also without `SecurityManager`:

No risk of bad plugins hacking Java internals!

Performance

- Slightly improved performance for some queries!
- With Lucene/Solr 7.3+ (LUCENE-7966):
  - Compression of large blobs during indexing (Elasticsearch JSON “`_source`”)
  - Sorting against docvalues with `MMapDirectory`

The future

Support?

Java 11

- Release will be in September 2018
- **Long Term Support** (LTS) by Oracle
- Most people will use this version
  - Java 9 and Java 10 are short-living
  - **Ubuntu 18.04** will use Java 10 as default, but switch to Java 11 in September (including LTS support)

Java 8 / 9 / 10 / 11

- After September 2018, no more (Oracle) Java 9 or 10 releases
- Java 8 has still LTS support till January 2019 (by Oracle)
- Ubuntu has LTS support for Java 8 and 10/11
- Redhat may package tar.gz files of Oracle 7, 8, 9, 10, 11 for much longer time!

Summary: Lucene / Solr

- **Minimum version** stays at **Java 8**
- **Full runtime support** for **Java 9** starting with Lucene/Solr 7.0
- Speed improvements by **MR-JAR** usage after Lucene/Solr 7.3
- **Solr:** Support for Java 10+ since Solr 7.3 *(startup scripts were broken)*

Thank you!

Questions?