Randomize Your Tests

…and it will blow your socks off

Dawid WEISS

Carrot Search s.c.

Berlin Buzzwords. May 2014

Introduction

⇓

Motivation

(why are unit tests insufficient?)

⇓

Randomized Testing: Core Concepts

⇓

RT in Practice (Example)

⇓

Summary and Conclusions

Introduction

Historical note

Industry and academia

Duran, Ntafos: An evaluation of random testing, 1984! Haskell: QuickCheck.

Hacking.

Fuzzifiers, vulnerability discovery.

Lucene Test Framework.

Lots of great ideas. Driven by real needs and bugs. Many contributors.

Carrot Search.

Handcrafted pseudo-randomness in Carrot2, Lingo3G, HPPC, …

Motivation

Software Tests

Confidence in implementation.

Reliability.

Insurance against future changes (regressions).

Assumptions $\neq$ The Real World

Can this ever fail?

```
String [] words = {"Berlin", "Spring", "Sun" };
Random r = new Random();
while (true) {
  String buzzword = words[Math.abs(r.nextInt()) % words.length];
  System.out.println(buzzword);
}
```

Can this ever fail?

```
String [] words = {"Berlin", "Spring", "Sun" };
Random r = new Random();
while (true) {
  String buzzword = words[Math.abs(r.nextInt()) % words.length];
  System.out.println(buzzword);
}
```

ArrayIndexOutOfBoundsException

`Math.abs(Integer.MIN_VALUE) < 0` **(!)**

“**Anything that can go wrong will go wrong.**”

(An unlikely event will eventually occur.)

Randomized Testing

Randomized testing is not a paradigm shift!

(Does not replace common-sense unit tests.)

Randomized testing is not mutation-based!

(To increase coverage, like PiTest.)

Randomized Testing

(mini-manifesto)

1. **Each run covers a different execution path.**
   By substituting components or diversifying data.

2. **Each run is deterministic.**
   Can be repeated from the same initial randomization seed.

3. **Tests are repeated lots of times.**
   Not just on changes, continuous build server, developers.

Why not just…

test everything?

What can be randomized?

1. Input data, iteration counts, arguments.

Random, constraint-bound, shuffled.

```
// Lucene/Solr source code (LTC)
final int iters = atLeast(200);
for (int iter = 0; iter < iters; iter++) {
  // ...
}
```

```
// Lucene/Solr source code (LTC)
if (random.nextBoolean()) {
  term = _TestUtil.randomRealisticUnicodeString(random);
} else {
  // we want to mix in limited-alphabet symbols so
  // we get more sharing of the nodes given how few
  // terms we are testing...
  term = simpleRandomString(random);
}
```

What can be randomized?

1. **Input data, iteration counts, arguments.**
   Random, constraint-bound, shuffled.

2. **Software components.**
   If multiple implementations exist. LTC: Field, Directory, IndexSearcher…

```
public static String randomDirectory(Random random) {
  if (rarely(random)) {
    return CORE_DIRECTORIES[random.nextInt(CORE_DIRECTORIES.length)];
  } else {
    return "RAMDirectory";
  }
}
```

What can be randomized?

1. **Input data, iteration counts, arguments.**
   Random, constraint-bound, shuffled.

2. **Software components.**
   If multiple implementations exist. LTC: Field, Directory, IndexSearcher…

3. **Environment.**
   Locale, TimeZone, JVM (!), operating system.

```
locale = TEST_LOCALE.equals("random") ? randomLocale(random) : ...
Locale.setDefault(locale);
timeZone = TEST_TIMEZONE.equals("random") ? randomTimeZone(random) : ...
TimeZone.setDefault(timeZone);
```

What can be randomized?

1. **Input data, iteration counts, arguments.**
   Random, constraint-bound, shuffled.

2. **Software components.**
   If multiple implementations exist. LTC: Field, Directory, IndexSearcher…

3. **Environment.**
   Locale, TimeZone, JVM (!), operating system.

4. **Exceptional triggers.**
   I/O problems, network problems (using mocks or runtime engineering).

Assertions in randomized test code?

Compare against reference.

Naïve, previous or alternative implementations.

Sanity checks.

Crude output checks (boundary conditions). Sanity assertions inside code.

Nothing.

Unchecked exceptions. Or a jvm core dump. Surprisingly effective :)

Randomized Testing in Practice

Lucene Q/A Components

Uwe's forbidden API checker.

https://code.google.com/p/forbidden-apis/

smoketester.py

Pre-release automated sanity checks.

Runtime checks (rules) in LTC.

Static memory leaks, stray threads, temporary file cleanups, assertions…

Security manager.

For enforcing filesystem (and other) policies.

RandomizedRunner.

https://github.com/carrotsearch/randomizedtesting/

RandomizedRunner's goals

Compatibility

with JUnit (and tools). At 99%, relax contracts when useful.

Built-in randomization

including reporting/ stack augmentations.

Test isolation

by tracking spawned threads. Timeouts. Terminations.

Utilities

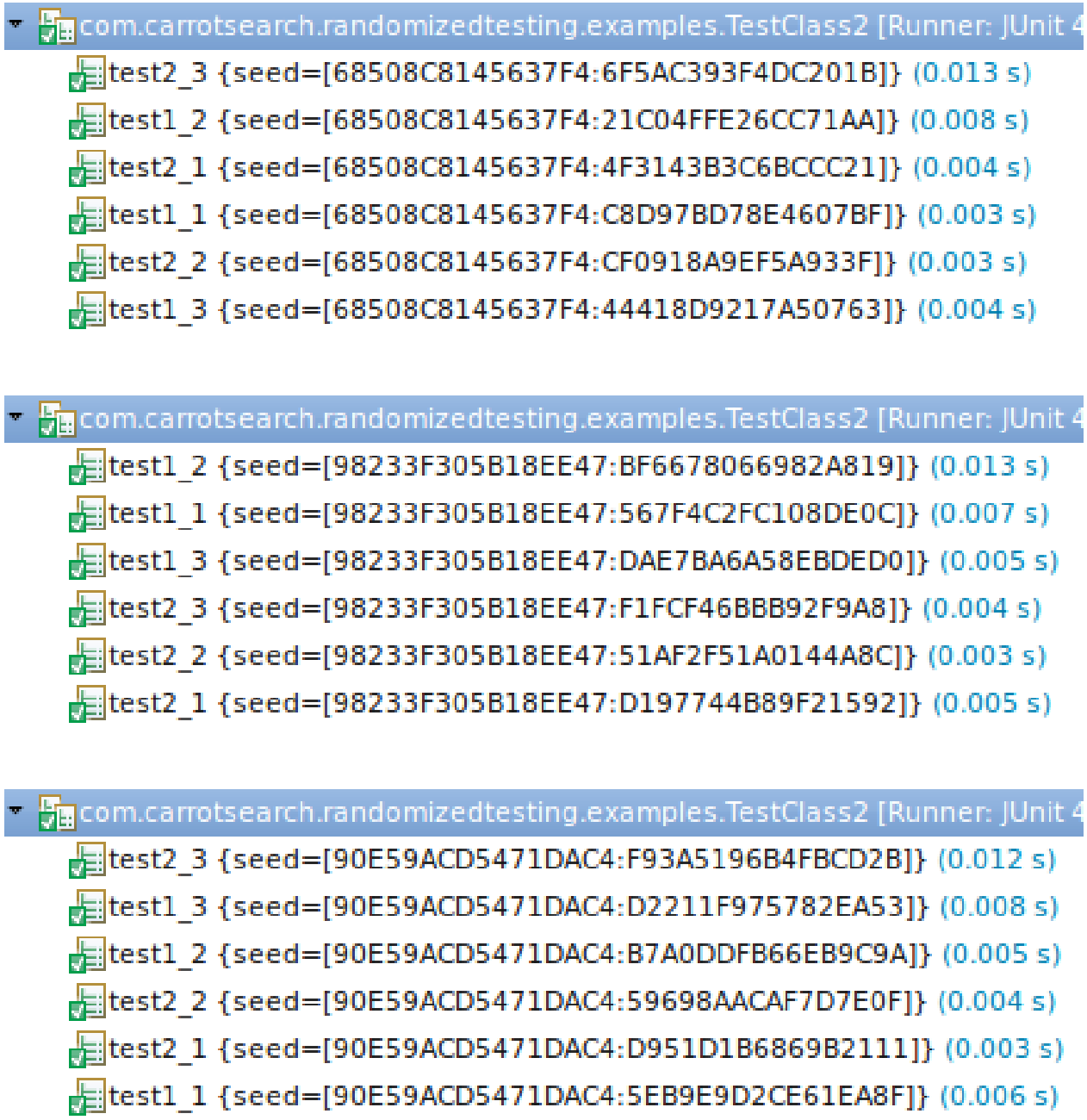
@Repeat, @Seed, @Nightly, @TestGroup, @TestFactories…

EXAMPLE

```
@RunWith(RandomizedRunner.class}  // <--- USE RR!
public class TestClass1 {
  @Test public void test1_1() { }
  @Test public void test1_2() { }
  @Test public void test1_3() { }
}

public class TestClass2 extends TestClass1 {
  @Test public void test2_1() { }
  @Test public void test2_2() { }
  @Test public void test2_3() { }
}
```

Using RR



Execution order shuffling

```java
@RunWith(RandomizedRunner.class)
public class TestMe {
  @Test
  public void theTruthIsOutThere() {
    RandomizedContext ctx = RandomizedContext.current();
    Random rnd = ctx.getRandom(); // => no setSeed()!
    // Your pseudo-randomized code here.
    assertTrue(rnd.nextBoolean());
  }
}
```

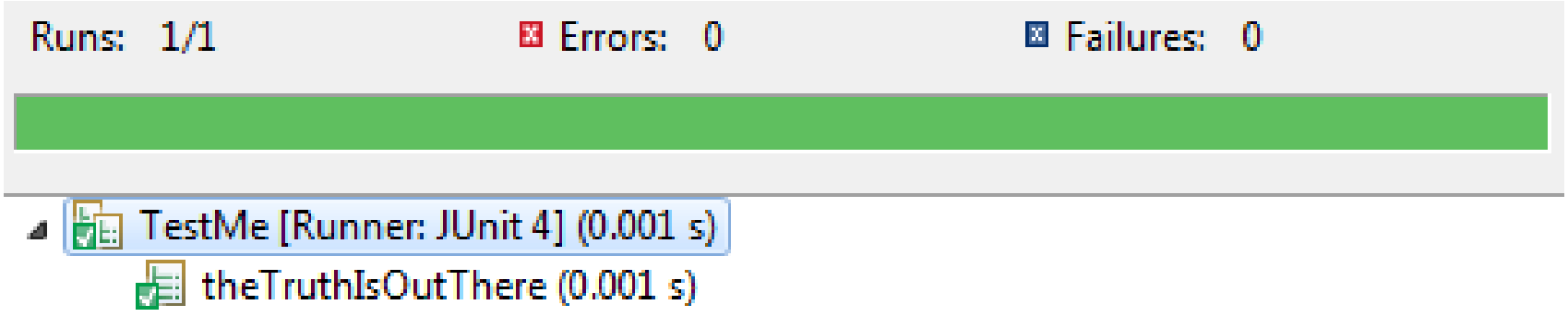
```java
@RunWith(RandomizedRunner.class)
public class TestMe {
  @Test
  public void theTruthIsOutThere() {
    RandomizedContext ctx = RandomizedContext.current();
    Random rnd = ctx.getRandom(); // => no setSeed()!
    // Your pseudo-randomized code here.
    assertTrue(rnd.nextBoolean());
  }
}
```

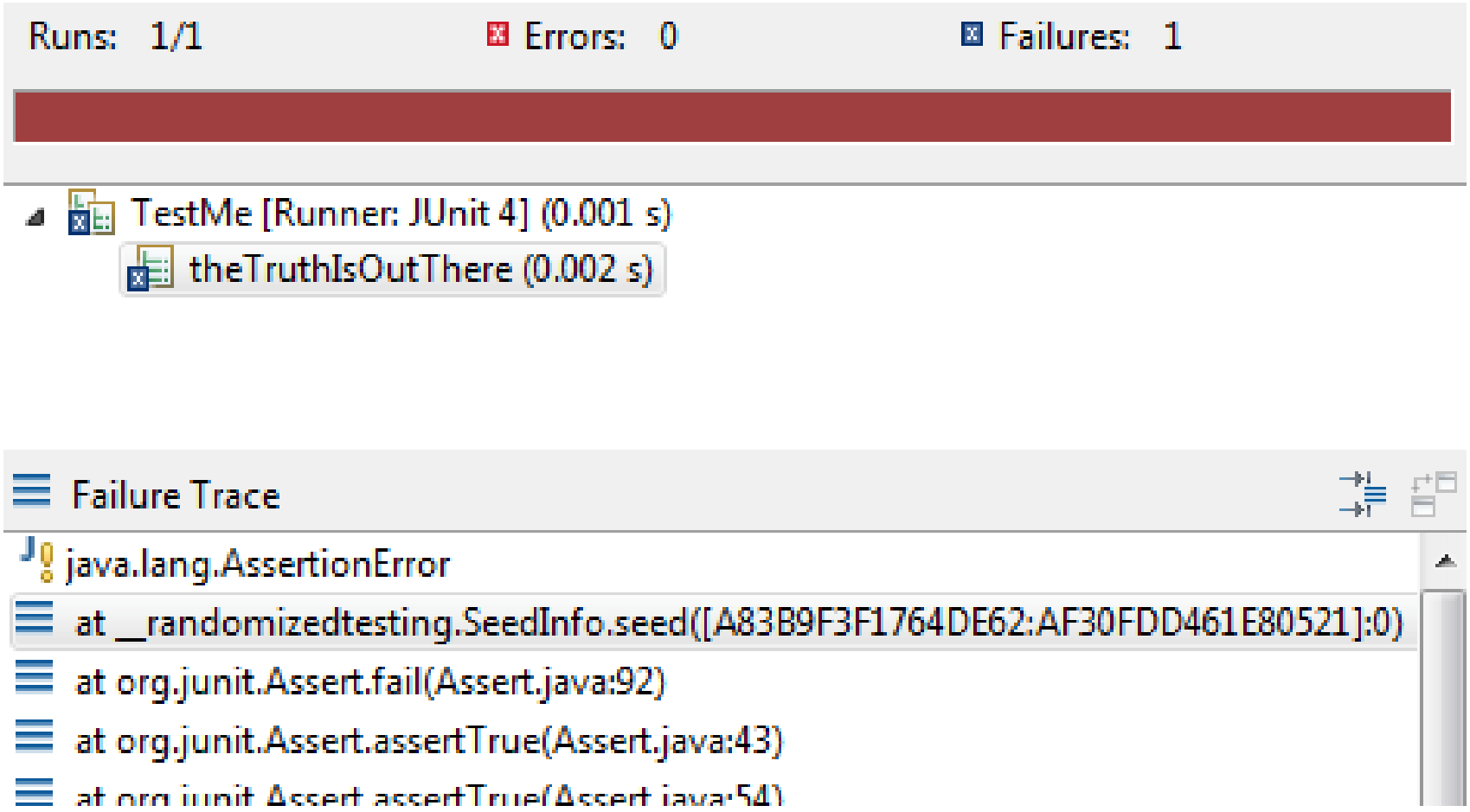
```java
public class TestMe extends RandomizedTest {
  @Test
  public void theTruthIsOutThere() {
    // Your pseudo-randomized code here.
    assertTrue(randomBoolean());
  }
}
```

```java
@RunWith(RandomizedRunner.class)
public class TestMe {
  @Test
  public void theTruthIsOutThere() {
    RandomizedContext ctx = RandomizedContext.current();
    Random rnd = ctx.getRandom(); // => no setSeed()!
    // Your pseudo-randomized code here.
    assertTrue(rnd.nextBoolean());
  }
}
```

```java
public class TestMe extends RandomizedTest {
  @Test
  public void theTruthIsOutThere() {
    // Your pseudo-randomized code here.
    assertTrue(randomBoolean());
  }
}
```

- **The Random is assigned per-thread (increases reproducibility).**
- **The context is invalidated after each scope (suite, test).**
- **Randomness is derived from the master seed, method name, repetition.**


Runs: 1/1
Errors: 0
Failures: 0
TestMe [Runner: JUnit 4] (0.001 s)
theTruthIsOutThere (0.001 s)



Synthetic stack trace frame with the master:method seed.

In case the JVM freezes, use a signal or jstack to get the master seed.

```
> jstack 6502
```

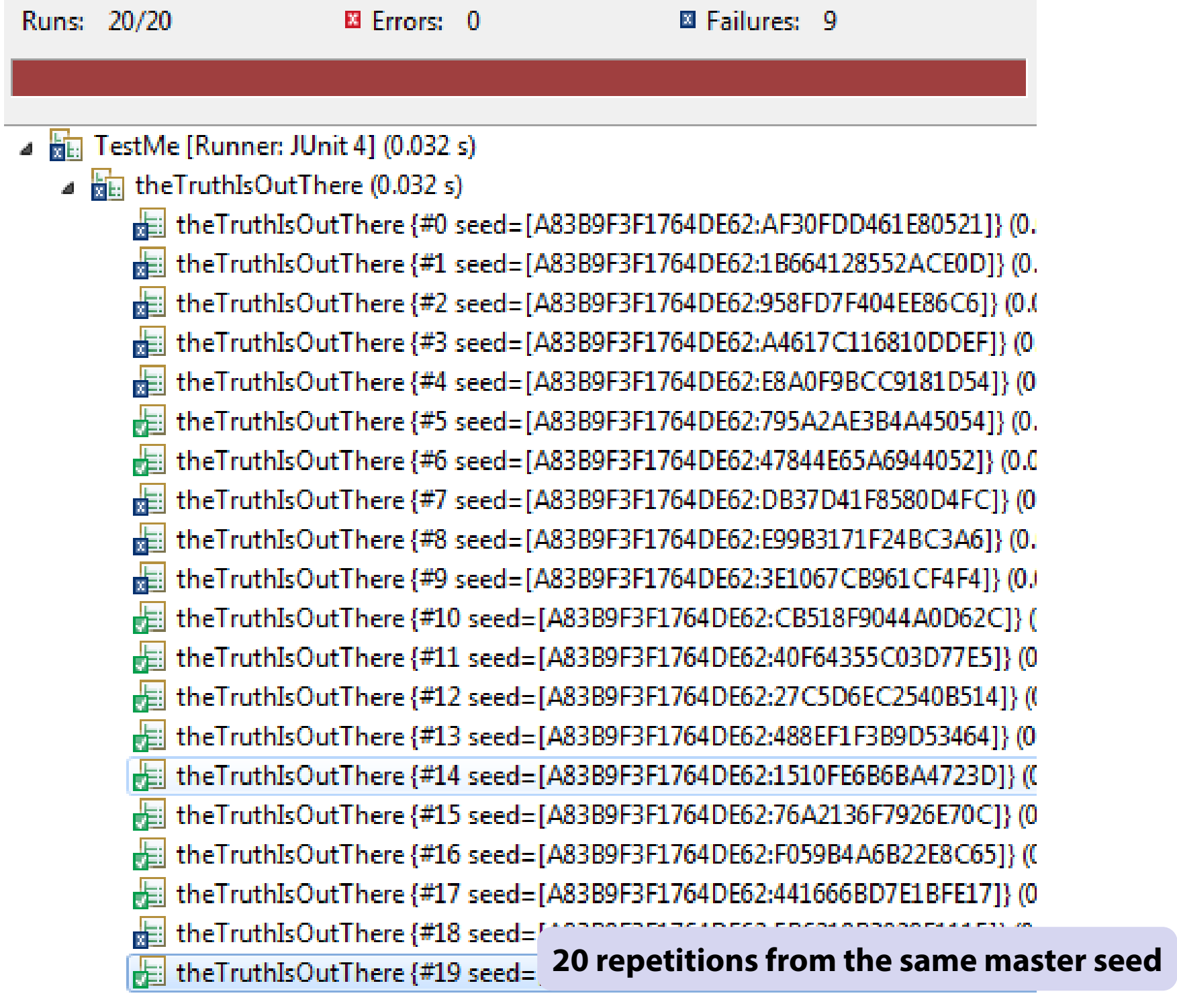
```
...
"TEST-TestScope-TestHanging.myTest-seed#[96F549A92265CBF8]"
waiting on condition [1169ef000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
at java.lang.Thread.sleep(Native Method)
at TestHanging.myTest(TestHanging.java:26)
```

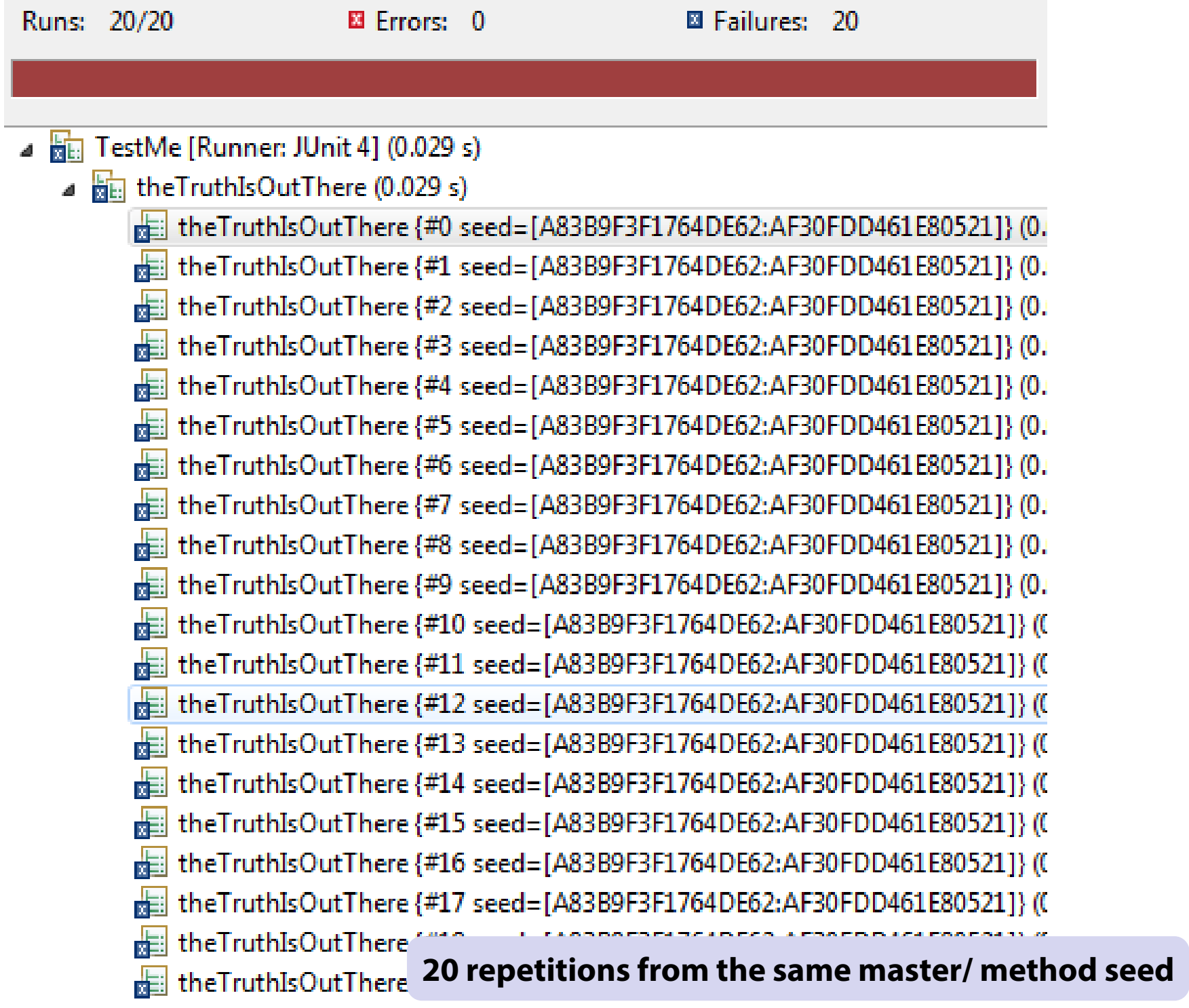
Once you've hit a failure…

```
@Seed("A83B9F3F1764DE62")
public class TestMe extends RandomizedTest {
  @Test
  @Repeat(iterations = 20)
  public void theTruthIsOutThere() {
    // Your pseudo-randomized code here.
    assertTrue(randomBoolean());
  }
}
```

…or set the seed externally via properties…

```
-Dtests.seed=A83B9F3F1764DE62
-Dtests.iter=20
```


Runs: 20/20
Errors: 0
Failures: 9
TestMe [Runner: JUnit 4] (0.032 s)
theTruthIsOutThere (0.032 s)
theTruthIsOutThere {#0 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
theTruthIsOutThere {#1 seed=[A83B9F3F1764DE62:1B664128552ACE0D]} (0.
theTruthIsOutThere {#2 seed=[A83B9F3F1764DE62:958FD7F404EE86C6]} (0.
theTruthIsOutThere {#3 seed=[A83B9F3F1764DE62:A4617C116810DDEF]} (0
theTruthIsOutThere {#4 seed=[A83B9F3F1764DE62:E8A0F9BCC9181D54]} (0
theTruthIsOutThere {#5 seed=[A83B9F3F1764DE62:795A2AE3B4A45054]} (0.
theTruthIsOutThere {#6 seed=[A83B9F3F1764DE62:47844E65A6944052]} (0.
theTruthIsOutThere {#7 seed=[A83B9F3F1764DE62:DB37D41F8580D4FC]} (0
theTruthIsOutThere {#8 seed=[A83B9F3F1764DE62:E99B3171F24BC3A6]} (0.
theTruthIsOutThere {#9 seed=[A83B9F3F1764DE62:3E1067CB961CF4F4]} (0.
theTruthIsOutThere {#10 seed=[A83B9F3F1764DE62:CB518F9044A0D62C]} (
theTruthIsOutThere {#11 seed=[A83B9F3F1764DE62:40F64355C03D77E5]} (0
theTruthIsOutThere {#12 seed=[A83B9F3F1764DE62:27C5D6EC2540B514]} (
theTruthIsOutThere {#13 seed=[A83B9F3F1764DE62:488EF1F3B9D53464]} (0
theTruthIsOutThere {#14 seed=[A83B9F3F1764DE62:1510FE6B6BA4723D]} (
theTruthIsOutThere {#15 seed=[A83B9F3F1764DE62:76A2136F7926E70C]} (0
theTruthIsOutThere {#16 seed=[A83B9F3F1764DE62:F059B4A6B22E8C65]} (
theTruthIsOutThere {#17 seed=[A83B9F3F1764DE62:441666BD7E1BFE17]} (0
theTruthIsOutThere {#18 seed=
theTruthIsOutThere {#19 seed=
20 repetitions from the same master seed


Runs: 20/20
Errors: 0
Failures: 20
TestMe [Runner: JUnit 4] (0.029 s)
theTruthIsOutThere (0.029 s)
theTruthIsOutThere {#0 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
theTruthIsOutThere {#1 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
theTruthIsOutThere {#2 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
theTruthIsOutThere {#3 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
theTruthIsOutThere {#4 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
theTruthIsOutThere {#5 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
theTruthIsOutThere {#6 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
theTruthIsOutThere {#7 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
theTruthIsOutThere {#8 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
theTruthIsOutThere {#9 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (0.
theTruthIsOutThere {#10 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (
theTruthIsOutThere {#11 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (
theTruthIsOutThere {#12 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (
theTruthIsOutThere {#13 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (
theTruthIsOutThere {#14 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (
theTruthIsOutThere {#15 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (
theTruthIsOutThere {#16 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (
theTruthIsOutThere {#17 seed=[A83B9F3F1764DE62:AF30FDD461E80521]} (
theTruthIsOutThere
theTruthIsOutThere
20 repetitions from the same master/ method seed

The downsides of test randomization

Failure reproducibility.

Hard to achieve for concurrent, time-based or otherwise uncoordinated code.

Incompatible random component configurations.

May require ignoring certain tests at runtime.

Randomized tests are typically pretty hairy.

More difficult to debug. Can cause JVM indigestion…

```
public void testCuriousMassiveString() throws Exception {
  String text = "yqt \u0728\u0707\u0712\u0720\u0734 \u204c\u201d hyipy \u2667\u2619" +
      "\u26ec\u267b\u26da uboyjwfbv \u2012\u205d\u2042\u200a\u2047\u2040 gyxmmz yvv %" +
      "\ufb86 \n<script> hupvobbv jvsztd x rww ct{1,5} brteyi dfgyzqbm hdykd ahgeizyhv" +
      " kLn c#\ud8f8\udd74 fPezd ktedq \ufcea=\ud997\uddc9\u876bJ\u0231\ud98a\udce0\uf872" +
      " zquqah \ub9d6\u144e\uc686 3\ud93d\udfca\u1215\ud614 tnorask \u0348\u0334\u0334" +
      "\u0300\u033d geqdeoghh foojebut \ufb52\u227ag\ud9bd\udc3a\u07efK nyantr lksxw fs" +
      "zies ubzqfolksjpgk \u6fa3\ud859\udc36\u0501\ucca0\u0306\u001e\ua756\u052f \ucaf7" +
      "\u0247\u0009\ufdddg \ud83c\udd02\ud83c\uddaf \u05628\u2b49\u02e3\u0718\u0769\u4f1b" +
      "\u0496\u0766\uecaa\ufb44 \u001d \u0006hr\u00f0\ue649\u041a\uda6f\udfa5\uf31b\ue274" +
      " ptgjf  \ud8cc\udf83M\u0013\u04c6i \u205f\u2004\u2032\u2001\u2057\u2066 \u07d0\uacdb" +
      "\u06a5z  pqfxwgbwe \ud1bc\u2eba\u2d45\u02ee\u56df xnujtfs \u1b19\u1b17\u1b39\u1b20" +
      "\u1b69\u1b58\u1b03\u1b6e\u1b73\u1b20 afsl zxlzziqh ahrhckhktf \ud801\udc5b\ud801\udc61" +
      " bkpmeyyqobwi qnkunmpjpihezll plhhws \u37f7\u41d6\u3dca\u3e80\u4923\u36b8 \u195a\u1959" +
      "\u196f\u1968\u1958\u197f </p \u0006s\u019f\uda82\udc90H_\u079d\ufd6f: idpp \u217c" +
      "\u2168\u2185\u2179\u2156\u2159\u217c\u2154\u2158 ({1,5}( jkieylqzmb bfirnaj \uea71" +
      "\uf17f\u0749\u054c \ud9ea\udf83\udbea\ude91j x\u3366\u09c2\ud828\udd13~\u6fda\ueeee" +
      " \ud834\udcd3\ud834\udc2b\ud834\udc8b\ud834\udcd8 dl \ud802\ude3a\ud802\ude36\ud802" +
      "\ude23\ud802\ude56 \u20ad\u20a0\u20a0\u20a0\u20b8\u20b4\u20ad lcql \u0547\u0156]" +
      "\ue344V] \ud90c\udd7d\u000eS\uf965\uf15e\u008f qdn \udac3\ude3c buy m qni \u31a4\u31a3" +
      "\u31b2\u31a9\u31a7 \u20df\u20ee\u20d3\u20d4\u20f1\u20de \u4dc1\u4dc4\u4dde\u4dec\u4df9" +
      "\u4dee\u4dc5 \udb40\udc36 gstfqnfWY \u24ab\u8d69\u0301 n?]v||]i )- \udb0b\ude77\uf634" +
      "\u0762 tpkkjlbcntsk eebtzirw xo hktxy \n</     vxro  xpr mtlp p|tjf|i?.- lxpfo \udbd7" +
      "\udf78\udbf5\udf57 u..b jjj]p?e jtckvhqb \u20eb\u20ea\u20fa\u20ef\u20e1\u20ed\u20eb vvm" +
      "uhbsvyi jx mqwxckf \u00d7 qqzs \u05ae\u022d\udb7c\udfb1\u070b vnhkz egnutyuat \udaa2" +
      "\udf20\ufa45#O\u2b61\u0d0e \u09a2\u0996\u09cd\u09b4 v \udbdb\ude9bqp owsln \ua837\ua833" +
      "\ua83f\ua83f\ua83f\ua831\ua83c\ua839 \u1a15\u1a1c\u1a12 \ud83c\ude20 >&pt&#x>129 \ud9f1" +
      "\udf8c\uecdd \ud809\udc48\ud809\udc72 wswskop \ua70
```

TestICUNormalizer2CharFilter.java

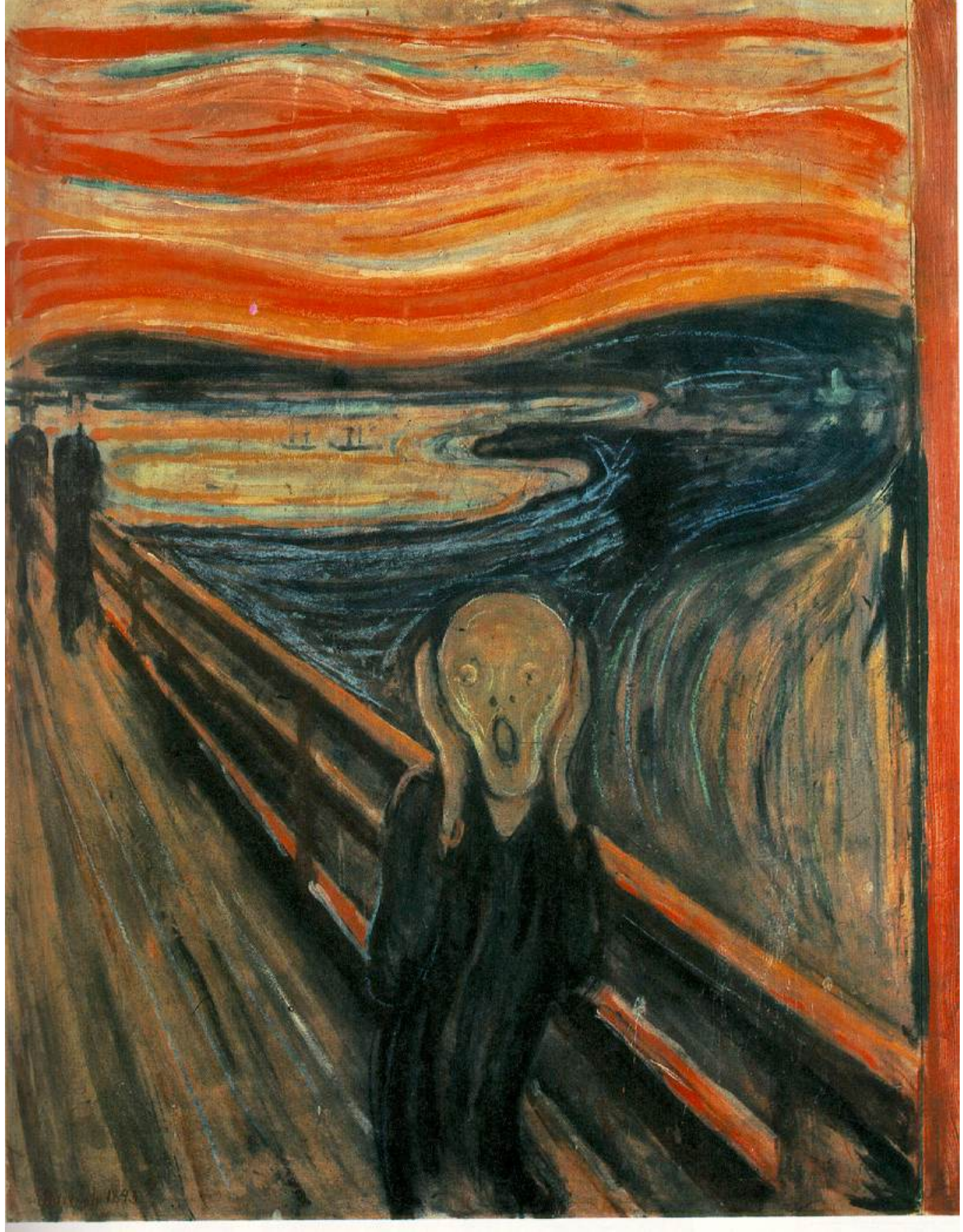
hotspot escape analysis bug in FreqProxTermsWriterPerField::flush

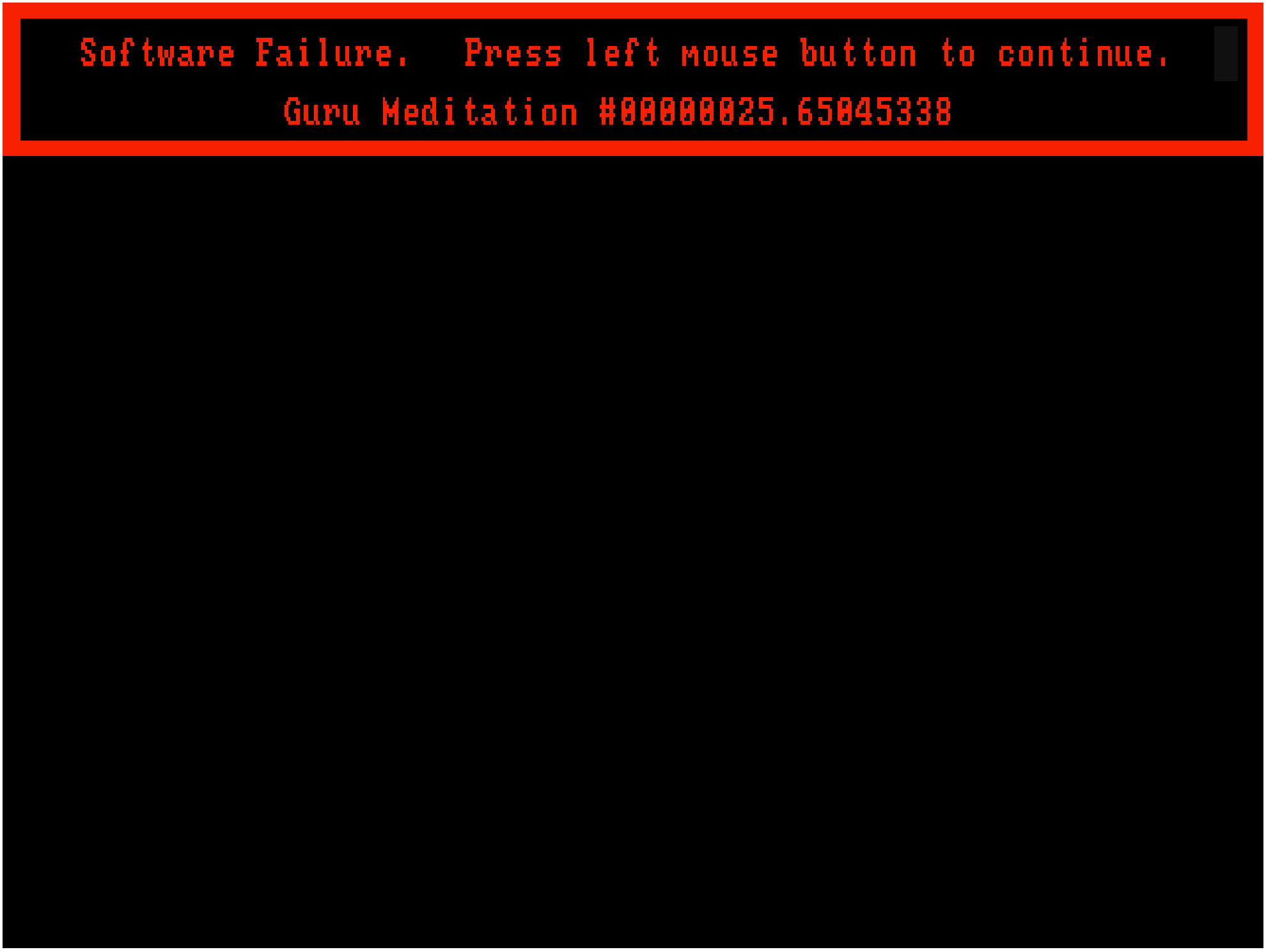
```
org.apache.lucene.index.FreqProxTermsWriterPerField::flush (1207 bytes)
  @ 4  org.apache.lucene.index.FieldInfo::isIndexed (5 bytes) accessor
  @ 16 org.apache.lucene.codecs.perfield.PerFieldPostingsFormat$FieldsWriter::addField (327 b
too big
  @ 23 org.apache.lucene.codecs.BlockTreeTermsWriter$TermsWriter::getComparator (4 bytes)
inline (hot)
    @ 0  org.apache.lucene.util.BytesRef::getUTF8SortedAsUnicodeComparator (4 bytes) inline (h
  @ 32 org.apache.lucene.index.FieldInfo::getIndexOptions (5 bytes) accessor
  @ 61 java.lang.Enum::compareTo (44 bytes) too big
  @ 79 java.lang.Enum::compareTo (44 bytes) too big
  @ 97 java.lang.Enum::compareTo (44 bytes) too big
  @ 238 java.util.HashMap::size (5 bytes) accessor
  @ 267 org.apache.lucene.index.TermsHashPerField::sortPostings (9 bytes) executed < MinInlir
  @ 279 org.apache.lucene.util.BytesRefHash::size (5 bytes) accessor
  @ 288 org.apache.lucene.util.BytesRef::<init> (8 bytes) inline (hot)
    @ 4  org.apache.lucene.util.BytesRef::<init> (9 bytes) inline (hot)
      @ 5  org.apache.lucene.util.BytesRef::<init> (41 bytes) inline (hot)
        @ 1  java.lang.Object::<init> (1 bytes) inline (hot)
        @ 26 org.apache.lucene.util.BytesRef::isValid (329 bytes) hot method too big
  @ 309 org.apache.lucene.index.ByteSliceReader::<init> (5 bytes) inline (hot)
    @ 1  org.apache.lucene.store.DataInput::<init> (5 bytes) inline (hot)
      @ 1  java.lang.Object::<init> (1 bytes) inline (hot)
  @ 318 org.apache.lucene.index.ByteSliceReader::<init> (5 bytes) inline (hot)
    @ 1  org.apache.lucene.store.DataInput::<init> (5 bytes) inline (hot)
      @ 1  java.lang.Object::<init> (1 bytes) inline (hot)
  @ 331 org.apache.lucene.index.SegmentInfo::getDocCount (23 bytes) inline (hot)
  @ 334 org.apache.lucene.util.FixedBitSet::<init> (29 bytes) inline (hot)
    @ 1  org.apache.lucene.search.DocIdSet::<init> (5 bytes) inline (hot)
      @ 1  java.lang.Object::<init> (1 bytes) inline (hot)
    @ 11 org.apache.lucene.util.FixedBitSet::bits2words (17 bytes) inline (hot)
  @ 350 org.apache.lucene.index.Term::<init> (13 bytes) inline (hot)
    @ 6  org.apache.lucene.util.BytesRef::<init> (8 bytes) inline (hot)
```

```
        @ 1  java.lang.Object::<init> (1 bytes) inline (hot)
        @ 26 org.apache.lucene.util.BytesRef::isValid (329 bytes) hot method too big
    @ 9  org.apache.lucene.index.Term::<init> (15 bytes) inline (hot)
      @ 1  java.lang.Object::<init> (1 bytes) inline (hot)
  @ 393 org.apache.lucene.util.ByteBlockPool::setBytesRef (107 bytes) inline (hot)
  @ 405 org.apache.lucene.index.TermsHashPerField::initReader (87 bytes) inline (hot)
    @ 83 org.apache.lucene.index.ByteSliceReader::init (153 bytes) inline (hot)
  @ 427 org.apache.lucene.index.TermsHashPerField::initReader (87 bytes) inline (hot)
    @ 83 org.apache.lucene.index.ByteSliceReader::init (153 bytes) inline (hot)
  @ 434 org.apache.lucene.codecs.BlockTreeTermsWriter$TermsWriter::startTerm (18 bytes)
inline (hot)
  @ 500 org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
  @ 1154 org.apache.lucene.codecs.TermStats::<init> (15 bytes) inline (hot)
    @ 1  java.lang.Object::<init> (1 bytes) inline (hot)
  @ 1157 org.apache.lucene.codecs.BlockTreeTermsWriter$TermsWriter::finishTerm (87 bytes)
inline (hot)
    @ 30 org.apache.lucene.util.fst.Util::toIntsRef (55 bytes) already compiled into a medium
    @ 37 org.apache.lucene.util.fst.NoOutputs::getNoOutput (4 bytes) inline (hot)
    @ 40 org.apache.lucene.util.fst.Builder::add (767 bytes) too big
    @ 52 org.apache.lucene.util.BytesRef::deepCopyOf (15 bytes) inline (hot)
      @ 4  org.apache.lucene.util.BytesRef::<init> (8 bytes) inline (hot)
        @ 4  org.apache.lucene.util.BytesRef::<init> (9 bytes) inline (hot)
          @ 5  org.apache.lucene.util.BytesRef::<init> (41 bytes) inline (hot)
            @ 1  java.lang.Object::<init> (1 bytes) inline (hot)
            @ 26 org.apache.lucene.util.BytesRef::isValid (329 bytes) hot method too big
      @ 10 org.apache.lucene.util.BytesRef::copyBytes (64 bytes) inline (hot)
        @ 52 java.lang.System::arraycopy (0 bytes) (intrinsic)
    @ 56 org.apache.lucene.codecs.BlockTreeTermsWriter$PendingTerm::<init> (16 bytes)
inline (hot)
      @ 2  org.apache.lucene.codecs.BlockTreeTermsWriter$PendingEntry::<init> (10 bytes)
inline (hot)
        @ 1  java.lang.Object::<init> (1 bytes) inline (hot)
  @ 563 org.apache.lucene.store.DataInput::readVInt (114 bytes) inline (hot)
    @ 1  org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
      @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
```

```
    @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
  @ 17 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
    @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
    @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
  @ 38 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
    @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
    @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
  @ 59 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
    @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
    @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
  @ 80 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
    @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
    @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 610 org.apache.lucene.store.DataInput::readVInt (114 bytes) inline (hot)
  @ 1  org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
    @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
    @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
  @ 17 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
    @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
    @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
  @ 38 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
    @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
    @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
  @ 59 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
    @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
    @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
  @ 80 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
    @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
    @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
@ 657 org.apache.lucene.index.SegmentInfo::getDocCount (23 bytes) inline (hot)
@ 710 org.apache.lucene.util.FixedBitSet::set (84 bytes) inline (hot)
@ 763 org.apache.lucene.index.SegmentInfo::getDocCount (23 bytes) call site not reached
@ 848 org.apache.lucene.store.DataInput::readVInt (114 bytes) inline (hot)
  @ 1  org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
    @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
```

```
     @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
    @ 17 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
     @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
     @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
    @ 38 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
     @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
     @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
    @ 59 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
     @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
     @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
    @ 80 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
     @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
     @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
  @ 871 org.apache.lucene.store.DataInput::readVInt (114 bytes) too big
  @ 888 org.apache.lucene.util.BytesRef::<init> (8 bytes) call site not reached
  @ 927 org.apache.lucene.util.BytesRef::grow (34 bytes) call site not reached
  @ 942 org.apache.lucene.index.ByteSliceReader::readBytes (84 bytes) too big
  @ 975 org.apache.lucene.store.DataInput::readVInt (114 bytes) inline (hot)
    @ 1  org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
     @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
     @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
    @ 17 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) inline (hot)
     @ 7  org.apache.lucene.index.ByteSliceReader::eof (52 bytes) inline (hot)
     @ 58 org.apache.lucene.index.ByteSliceReader::nextSlice (198 bytes) inline (hot)
    @ 38 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) NodeCountInliningCutoff
    @ 59 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) NodeCountInliningCutoff
    @ 80 org.apache.lucene.index.ByteSliceReader::readByte (78 bytes) NodeCountInliningCutoff
  @ 985 org.apache.lucene.store.DataInput::readVInt (114 bytes) size > DesiredMethodLimit
  @ 1077 org.apache.lucene.codecs.lucene40.Lucene40PostingsWriter::addPosition (385 bytes)
size > DesiredMethodLimit
  @ 1114 org.apache.lucene.codecs.lucene41.Lucene41PostingsWriter::addPosition (367 bytes)
size > DesiredMethodLimit
  @ 1200 org.apache.lucene.util.FixedBitSet::cardinality (15 bytes) size > DesiredMethodLimit
  @ 1203 org.apache.lucene.codecs.BlockTreeTermsWriter.TermsWriter::finish (354 bytes)
size > DesiredMethodLimit
```




Software Failure. Press left mouse button to continue.
Guru Meditation #00000025.65045338

Summary and Conclusions

The Good

Explore complex boundary conditions.

May or may not hit them, but there is a chance!

Unexpected component-component interactions.

Pairwise component compatibility.

Unexpected environment interactions.

JVM, operating system differences.

Production-grade tool support.

RandomizedTesting used in Lucene, Solr, ES…

The Bad

Failures may prove difficult to debug.

Race conditions. Complex inputs. Multithreaded code.

The Ugly

Looping with different master seed.

Currently requires hacks.

Tight testing sandbox.

WTF moments may be frustrating.

dawid.weiss@carrotsearch.com

Randomized testing package is @labs:
http://labs.carrotsearch.com/randomizedtesting.html