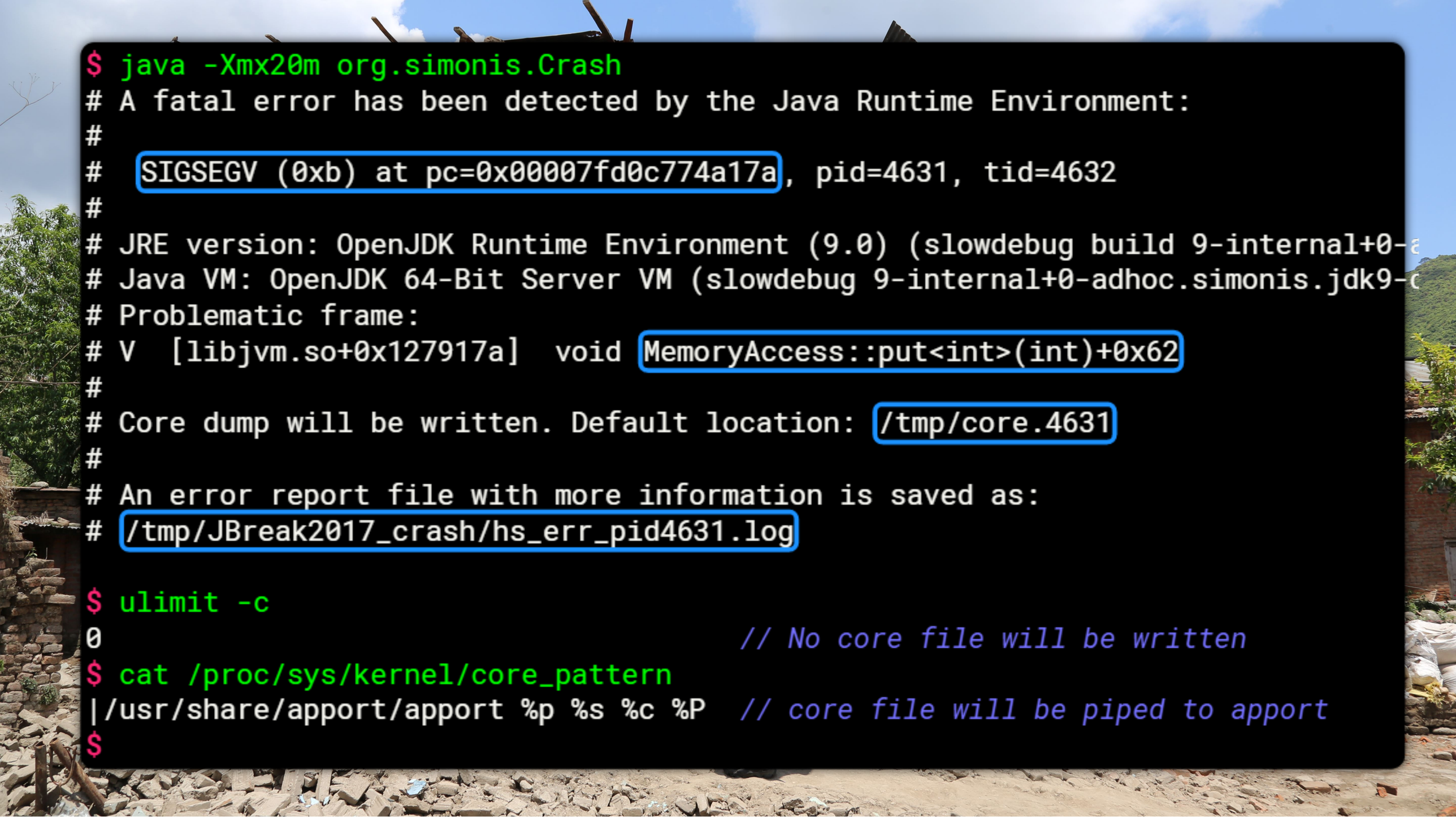
Analyzing HotSpot Crashes

Volker Simonis [Фолкер Симонис], SAP / volker.simonis@gmail.com


$ java -Xmx20m org.simonis.Crash
# A fatal error has been detected by the Java Runtime Environment:
#
SIGSEGV (0xb) at pc=0x00007fd0c774a17a, pid=4631, tid=4632
#
JRE version: OpenJDK Runtime Environment (9.0) (slowdebug build 9-internal+0-a
Java VM: OpenJDK 64-Bit Server VM (slowdebug 9-internal+0-adhoc.simonis.jdk9-c
Problematic frame:
V [libjvm.so+0x127917a] void MemoryAccess::put<int>(int)+0x62
#
Core dump will be written. Default location: /tmp/core.4631
#
# An error report file with more information is saved as:
# /tmp/JBreak2017_crash/hs_err_pid4631.log
$ ulimit -c
0   // No core file will be written
$ cat /proc/sys/kernel/core_pattern
|/usr/share/apport/apport %p %s %c %P // core file will be piped to apport
$

The Crash Example

```
public class Crash {

  final static Unsafe UNSAFE = getUnsafe();

  public static void crash(int x) {
    UNSAFE.putInt(0x99, x);
  }

  public static void main(String[] args) {
    crash(0x42);
  }
}
```

hs_err_pid4631.log - Summary

```
---------------  S U M M A R Y ------------

Command Line: -Xmx20m org.simonis.Crash

Host: simonis, Intel(R) Core(TM) i5-4300U CPU @ 1.90GHz, 4 cores, 7G, \
                                                 Ubuntu 14.04.1 LTS
Time: Tue Mar 14 16:48:36 2017 CET elapsed time: 1 seconds (0d 0h 0m 1s)

---------------  T H R E A D  ---------------

Current thread (0x00007fd0c0019800):  JavaThread "main" \
  [_thread_in_vm, id=4632, stack(0x00007fd0c8fa0000,0x00007fd0c90a0000)]
```

hs_err_pid4631.log - Native/Mixed Stack Trace

```
Stack: [0x00007fd0c8fa0000,0x00007fd0c90a0000],  sp=0x00007fd0c909e500,  free sp
Native frames: (J=compiled Java code, A=aot compiled Java code, j=interpreted, \
V  [libjvm.so+0x127917a]  void MemoryAccess::put<int>(int)+0x62
V  [libjvm.so+0x1270574]  Unsafe_PutInt+0x174
j  jdk.internal.misc.Unsafe.putInt(Ljava/lang/Object;JI)V+0 java.base@9-internal
j  jdk.internal.misc.Unsafe.putInt(JI)V+4 java.base@9-internal
j  sun.misc.Unsafe.putInt(JI)V+5 jdk.unsupported@9-internal
j  org.simonis.Crash.crash(I)V+7
j  org.simonis.Crash.main([Ljava/lang/String;)V+2
v  ~StubRoutines::call_stub
V  [libjvm.so+0xc6abd5]  JavaCalls::call_helper(JavaValue*, methodHandle const&,
V  [libjvm.so+0x10524b1]  os::os_exception_wrapper(void (*)(JavaValue*, methodHa
V  [libjvm.so+0xc6a512]  JavaCalls::call(JavaValue*, methodHandle const&, JavaCa
V  [libjvm.so+0xc887dc]  jni_invoke_static(JNIEnv_*, JavaValue*, _jobject*, JNIC
V  [libjvm.so+0xc9f1f5]  jni_CallStaticVoidMethod+0x33b
C  [libjli.so+0x49f8]  JavaMain+0xac0
C  [libpthread.so.0+0x8182]  start_thread+0xc2
```

hs_err_pid4631.log - Java Stack Trace

```
Java frames: (J=compiled Java code, j=interpreted, Vv=VM code)
j  jdk.internal.misc.Unsafe.putInt(Ljava/lang/Object;JI)V+0 java.base@9-internal
j  jdk.internal.misc.Unsafe.putInt(JI)V+4 java.base@9-internal
j  sun.misc.Unsafe.putInt(JI)V+5 jdk.unsupported@9-internal
j  org.simonis.Crash.crash(I)V+7
j  org.simonis.Crash.main([Ljava/lang/String;)V+2
v  ~StubRoutines::call_stub
```

hs_err_pid4631.log - Register Mapping

```
siginfo: si_signo: 11 (SIGSEGV), si_code: 1 (SEGV_MAPERR), si_addr: 0x000000000

Register to memory mapping:

RAX=0x0000000000000042 is an unknown value
RBX={method} {0x00007fd0798abc98} jdk/internal/misc/Unsafe.putInt
RCX=0x00007fd0c0019800 is a thread
RDX=0x0000000000000099 is an unknown value
RSP=0x00007fd0c909e500 is pointing into the stack for thread: 0x00007fd0c001980
RBP=0x00007fd0c909e540 is pointing into the stack for thread: 0x00007fd0c001980
RSI=0x0000000000000042 is an unknown value
RDI=0x00007fd0c909e5c0 is pointing into the stack for thread: 0x00007fd0c001980
R8 =0x0000000000000042 is an unknown value
R9 =
[error occurred during error reporting (printing register info), id 0xb]
```

hs_err_pid4631.log - Registers

```
Registers:
RAX=0x0000000000000042, RBX=0x00007fd0798abc98, RCX=0x00007fd0c0019800,
RDX=0x0000000000000099, RSP=0x00007fd0c909e500, RBP=0x00007fd0c909e540,
RSI=0x0000000000000042, RDI=0x00007fd0c909e5c0, R8 =0x0000000000000042,
R9 =0x00000000ffffffe0, R10=0x00007fd0a872da3c, R11=0x0000000000000008
R12=0x0000000000000000, R13=0x00007fd0798abc90, R14=0x00007fd0c909e6c8,
R15=0x00007fd0c0019800, RIP=0x00007fd0c774a17a, EFLAGS=0x0000000000010246,
CSGSFS=0x8cd7000000000033, ERR=0x0000000000000006, TRAPNO=0x000000000000000e
```

hs_err_pid4631.log - Stack/Instructions

```
Top of Stack: (sp=0x00007fd0c909e500)
0x00007fd0c909e500:   00000042c909e540 00007fd0c909e5c0
0x00007fd0c909e510:   0000000000000099 0000000000000099
0x00007fd0c909e520:   00007fd0c0019800 00007fd0c909e501
0x00007fd0c909e530:   00007fd0c909e580 8cd73cc3b5998700

Instructions: (pc=0x00007fd0c774a17a)
0x00007fd0c774a15a:   89 c7 e8 ef f5 ff ff 48 89 45 d8 8b 55 c4 48 8b
0x00007fd0c774a16a:   45 c8 89 d6 48 89 c7 e8 60 13 00 00 48 8b 55 d8
0x00007fd0c774a17a:   89 02 48 8d 45 e0 48 89 c7 e8 1e f7 ff ff 48 8b
0x00007fd0c774a18a:   45 f8 64 48 33 04 25 28 00 00 00 74 05 e8 14 01
```

Using `gdb`, `objdump` or `https://onlinedisassembler.com`

```
0x00007fd0c774a17a:   mov %eax,(%rdx)   // %eax=0x42, %rdx=0x99
...
```

Generating `hs_err.log` Files

```
<signal handler>
  signalHandler(sig=11, info=0x7ffff7fda070, uc=0x7ffff7fd9f40)
    JVM_handle_linux_signal(sig=11, info=0x7ffff7fda070,
    abort_if_unrecognized=1)
      VMError::report_and_die(sig=11, pc=0x7ffff668a17a <MemoryAccess::put>)
        VMError::report(stream=0x7ffff71aa9e0 <VMError::log>, _verbose=true)
          print_native_stack(...)
          print_stack_trace(...)
          os::print_register_info(...)
          ...
```

Whenever the HotSpot VM encounters an irrevocable error like:

- an unexpected signal
- a guarantee or assertion
- a native "out of memory" situation

it calls `VMError::report()` which does the job!

Generating `hs_err` Files

Creating the `hs_err` file is not trivial:

- the VM is in an inconsistent state
- the VM may run out of native memory / stack space
- the internal data structures may be corrupted
- have to gracefully handle subsequent signals/assertions/guarantees and timeouts!

```
Register to memory mapping:
RAX=0x0000000000000042 is an unknown value
RBX={method} {0x00007fd0798abc98} jdk/internal/misc/Unsafe.putInt()
RCX=0x00007fd0c0019800 is a thread
RDX=0x0000000000000099 is an unknown value
R9 =
[error occurred during error reporting (printing register info), id 0xb]
```

- use `SafeFetch` (see "HotSpot Internals" from Joker 2016)

Triggering `hs_err.log` from `jcmd`

Prints VM information without crash and thread details:

```
$ jcmd 27211 VM.info
# JRE version: OpenJDK Runtime Environment (9.0) (slowdebug build 9.jdk9-dev)
# Java VM: OpenJDK 64-Bit Server VM (slowdebug 9-internal.simonis.jdk9-dev,     \
                        mixed mode, tiered, compressed oops, g1 gc, linux-amd64)
---------------  S U M M A R Y ------------

Command Line: -Xmx128m HelloAWT

Host: simonis, Intel(R) Core i5-4300U CPU @ 1.90GHz, 4 cores, 7G, Ubuntu 14.04
Time: Fri Mar 24 14:17:50 2017 CET elapsed time: 38 seconds (0d 0h 0m 38s)
---------------  P R O C E S S  ---------------

Heap address: 0x00000000f8000000, size: 128 MB, Compressed Oops mode: 32-bit
Narrow klass base: 0x0000000000000000, Narrow klass shift: 3
Compressed class space size: 1073741824 Address: 0x0000000100000000
```

Where did we crash?

```
# SIGSEGV (0xb) at pc=0x00007f755074ec61, pid=10985, tid=10986
# ...
# Problematic frame:
# j  org.simonis.CrashInt.crash(Lorg/simonis/CrashInt;)I+0

si_signo: 11 (SIGSEGV), si_code: 1 (SEGV_MAPERR), si_addr: 0x000000000badbaca
...
RAX=0x000000000badbabe is an unknown value
...
arraylength  190 [0x00007f755074ec60, 0x00007f755074ec80]  32 bytes
[Disassembling for mach='i386:x86-64']
  0x00007f755074ec60: pop    %rax                    // pop array from stack
  0x00007f755074ec61: mov    0xc(%rax),%eax          // load length field
  0x00007f755074ec64: movzbl 0x1(%r13),%ebx          // load next bytecode
  0x00007f755074ec69: inc    %r13                    // increment bytecode pointer
  0x00007f755074ec6c: movabs $0x7f756fd00980,%r10    // load template table
  0x00007f755074ec76: jmpq   *(%r10,%rbx,8)          // jump to next bytecode
```

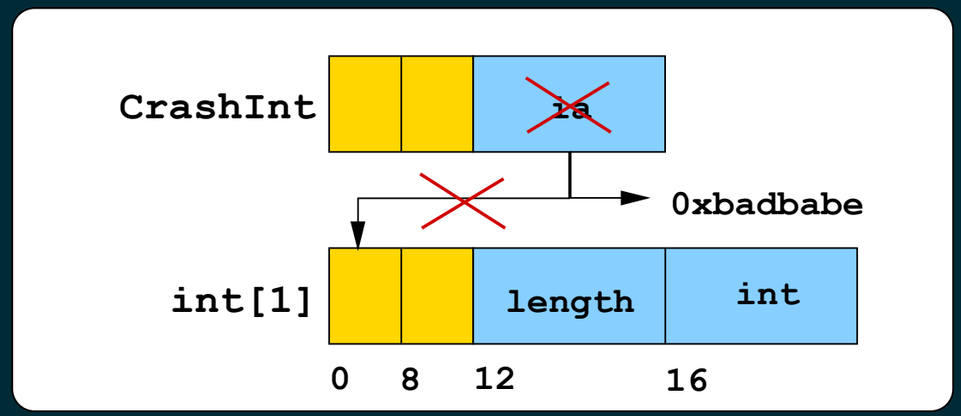
Crash in the Interpreter

```
public class CrashInt {
  int[] ia = new int[1];

  final static Unsafe UNSAFE = getUnsafe();

  public int crash() {
    return ia.length;
  }

  public static void main(String[] args) {
```




```
$ javap -c org.simonis.CrashInt
...
int crash()
```

```
    CrashInt ci = new CrashInt();
    UNSAFE.putLong(ci, 12L, 0xbadbabe);
    ci.crash();
  }
}
```

```
Code:
0: aload_0          // load 'this'
1: getfield #23 // int[] this.ia
4: arraylength
5: ireturn
```

Crash Location and Crash Reason

- Crash location and crash reason can different
- E.g. an error in the JIT can trigger an error in GC
- E.g. an error in native code can trigger an error in the interpreter
- Carefully analyzing the `hs_err` file can give hints to the root cause
- Look for well known byte patterns and recent events like:
  GC, deoptimization, class loading/redefinition, JIT compilation, exceptions, etc.

Out of Memory

```
$ java -Xmx20m -XX:MaxMetaspaceSize=6m org.simonis.CrashOOM
Exception in thread "main" java.lang.OutOfMemoryError: Metaspace
        at java.base/java.lang.ClassLoader.defineClass1(Native Method)
        at java.base/java.lang.ClassLoader.defineClass(ClassLoader.java:986)
        at org.simonis.CrashOOM$MyClassLoader.myDefineClass(CrashOOM.java:10)
        at org.simonis.CrashOOM.crash(CrashOOM.java:30)
        at org.simonis.CrashOOM.main(CrashOOM.java:40)
```

Out of Memory - Let it Crash!

```
$ java -Xmx20m -XX:MaxMetaspaceSize=6m -XX:+CrashOnOutOfMemoryError \
                                       org.simonis.CrashOOM
Aborting due to java.lang.OutOfMemoryError: Metaspace
# To suppress the following error report, specify this argument
# after -XX: or in .hotspotrc:  SuppressErrorAt=/debug.cpp:340
#
# A fatal error has been detected by the Java Runtime Environment:
#
#  fatal error: OutOfMemory encountered: Metaspace
#
# Core dump will be written. Default location: /tmp/JBreak2017_crash_oom/core
#
# An error report file with more information is saved as:
# /tmp/JBreak2017_crash_oom/hs_err_pid23919.log
```

Out of Memory - hs_err_pid23919.log

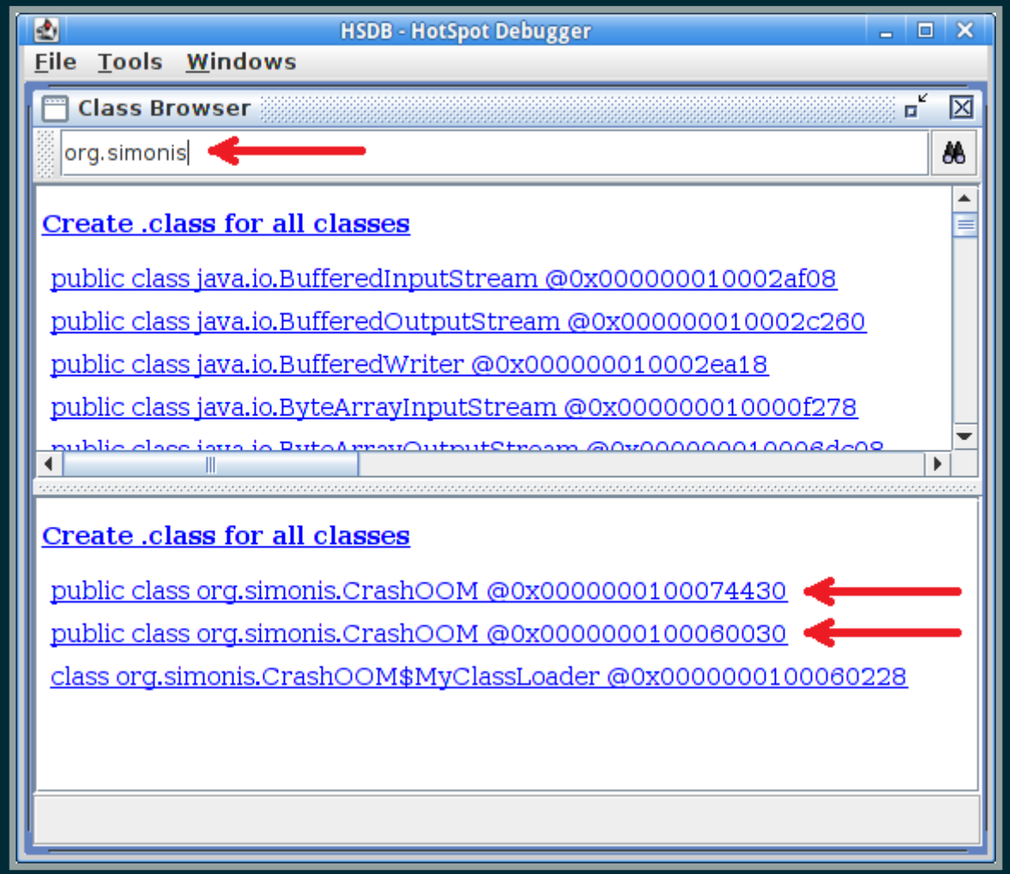
```
Native frames: (J=compiled Java code, j=interpreted, Vv=VM code, C=native code)
V  [libjvm.so+0xfa6346]  Metaspace::report_metadata_oome()+0x1aa
V  [libjvm.so+0xfa60a8]  Metaspace::allocate()+0x308
V  [libjvm.so+0x5bee05]  MetaspaceObj::operator new()+0x4f
V  [libjvm.so+0x97bc93]  ConstantPool::allocate()+0x8f
V  [libjvm.so+0x851b21]  ClassFileParser::parse_stream()+0x42f
V  [libjvm.so+0x85128c]  ClassFileParser::ClassFileParser()+0x61c
V  [libjvm.so+0xe2ee3f]  KlassFactory::create_from_stream()+0x259
V  [libjvm.so+0x11fc770]  SystemDictionary::resolve_from_stream()+0x238
V  [libjvm.so+0xcec45c]  jvm_define_class_common()+0x37f
V  [libjvm.so+0xcec983]  JVM_DefineClassWithSource+0x207
C  [libjava.so+0xe6b1]  Java_java_lang_ClassLoader_defineClass1+0x231
j  java.lang.ClassLoader.defineClass1()Ljava/lang/Class;+0 java.base@9-internal
j  java.lang.ClassLoader.defineClass()Ljava/lang/Class;+27 java.base@9-internal
j  org.simonis.CrashOOM$MyClassLoader.myDefineClass()Ljava/lang/Class;+7
j  org.simonis.CrashOOM.crash(I)V+27
```

Out of Memory - hs_err_pid23919.log

```
Events (10 events):
Event: 1,269 loading class org/simonis/CrashOOM$MyClassLoader
Event: 1,269 loading class org/simonis/CrashOOM$MyClassLoader done
Event: 1,464 Executing VM operation: CollectForMetadataAllocation
Event: 1,714 Executing VM operation: CollectForMetadataAllocation done
GC Heap History (6 events):
Event: 1,465 GC heap before
 Metaspace       used 6056K, capacity 6124K, committed 6144K, reserved 1056768K
Event: 1,507 GC heap after
 Metaspace       used 6056K, capacity 6124K, committed 6144K, reserved 1056768K
Event: 1,507 GC heap before
 Metaspace       used 6056K, capacity 6124K, committed 6144K, reserved 1056768K
Event: 1,614 GC heap after
 Metaspace       used 6056K, capacity 6124K, committed 6144K, reserved 1056768K
Event: 1,614 GC heap before
 Metaspace       used 6056K, capacity 6124K, committed 6144K, reserved 1056768K
Event: 1,713 GC heap after
 Metaspace       used 6053K, capacity 6120K, committed 6144K, reserved 1056768K
```

Serviceability Agent - `core.23919`

```
$ jhsdb hsdb --exe /output-jdk9-dev-dbg/images/jdk/bin/java --core core.23919
```



Out of Memory - hs_err_pid23919.log

```
Internal exceptions (10 events):
Event: 1,459 Thread 0x00007f4150019800 Exception a 'java/lang/LinkageError':
 loader (instance of  org/simonis/CrashOOM$MyClassLoader):
 attempted duplicate class definition for name: "org/simonis/CrashOOM" thrown
Event: 1,460 Thread 0x00007f4150019800 Exception a 'java/lang/LinkageError':
 loader (instance of  org/simonis/CrashOOM$MyClassLoader):
 attempted duplicate class definition for name: "org/simonis/CrashOOM" thrown
Event: 1,461 Thread 0x00007f4150019800 Exception a 'java/lang/LinkageError':
 loader (instance of  org/simonis/CrashOOM$MyClassLoader):
 attempted duplicate class definition for name: "org/simonis/CrashOOM" thrown
Event: 1,462 Thread 0x00007f4150019800 Exception a 'java/lang/LinkageError':
 loader (instance of  org/simonis/CrashOOM$MyClassLoader):
 attempted duplicate class definition for name: "org/simonis/CrashOOM" thrown
...
Event: 1,465 Thread 0x00007f4150019800 Exception a 'java/lang/LinkageError':
 loader (instance of  org/simonis/CrashOOM$MyClassLoader):
 attempted duplicate class definition for name: "org/simonis/CrashOOM" thrown
```

Out of Memory - Serviceability Agent

```
public class ClassStatistics  extends sun.jvm.hotspot.tools.Tool {
  public void run() {
    ClassLoaderDataGraph cldg = VM.getVM().getClassLoaderDataGraph();
    for (ClassLoaderData cld =
            cldg.getClassLoaderGraphHead(); cld != null; cld = cld.next()) {
      for (Klass k = cld.getKlasses(); k != null; k = k.getNextLinkKlass()) {
        if (k instanceof InstanceKlass) {
          InstanceKlass ik = (InstanceKlass)k;
          Oop cl = ik.getClassLoader();
          System.out.println(k.getName().asString() + " (" +
              (cl == null ? "null" : cl.getKlass().getName().asString()) + ")");
        }
      }
    }
  }
  public static void main(String[] args) {
    ClassStatistics cs = new ClassStatistics();
    cs.execute(args);
```

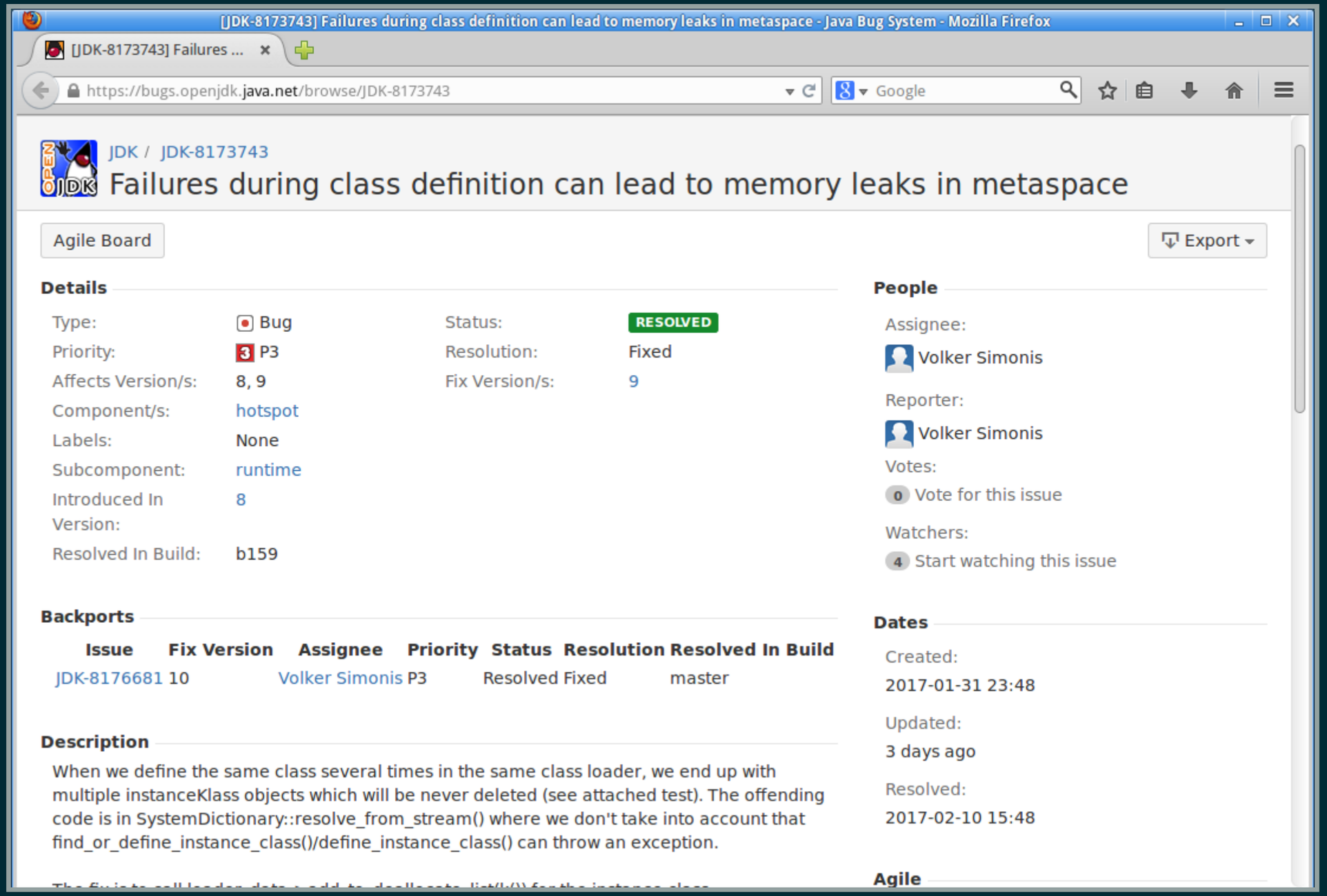
Out of Memory - hs_err_pid23919.log

```
$ java --add-modules jdk.hotspot.agent --add-exports .. \
    org.simonis.ClassStatistics /output-jdk9-dev-dbg/images/jdk/bin/java core
Debugger attached successfully.
Server compiler detected.
JVM version is 9-internal+0-adhoc.simonis.jdk9-dev
org/simonis/CrashOOM (org/simonis/CrashOOM$MyClassLoader)
org/simonis/CrashOOM (org/simonis/CrashOOM$MyClassLoader)
org/simonis/CrashOOM (org/simonis/CrashOOM$MyClassLoader)
org/simonis/CrashOOM (org/simonis/CrashOOM$MyClassLoader)
org/simonis/CrashOOM (org/simonis/CrashOOM$MyClassLoader)
org/simonis/CrashOOM (org/simonis/CrashOOM$MyClassLoader)
org/simonis/CrashOOM (org/simonis/CrashOOM$MyClassLoader)
...
$ java --add-modules jdk.hotspot.agent --add-exports .. \
    org.simonis.ClassStatistics /output-jdk9-dev-dbg/images/jdk/bin/java core \
    | grep CrashOOM | wc
    320     640   18600
```

Out of Memory

```
public class CrashOOM {
  public static void crash(int count) throws Exception {
    byte[] buf = getBytecodes("CrashOOM");
    // exposes protected ClassLoader.defineClass() as myDefineClass()
    MyClassLoader cl = new MyClassLoader();
    for (int i = 0; i < count; i++) {
      try {
        cl.myDefineClass("org.simonis.CrashOOM", buf, 0, buf.length);
      }
      catch (LinkageError jle) {
        if (i == 0) throw new Exception("Should succeed the first time.");
      }
    }
  }
  public static void main(String[] args) throws Exception {
    crash(args.length > 0 ? Integer.parseInt(args[0]) : 1000);
  }
```

https://bugs.openjdk.java.net/browse/JDK-8173743


[JDK-8173743] Failures during class definition can lead to memory leaks in metaspace - Java Bug System - Mozilla Firefox
[JDK-8173743] Failures ...
https://bugs.openjdk.java.net/browse/JDK-8173743
Google
JDK / JDK-8173743
Failures during class definition can lead to memory leaks in metaspace
Agile Board
Export
Details
Type: Bug
Priority: P3
Affects Version/s: 8, 9
Component/s: hotspot
Labels: None
Subcomponent: runtime
Introduced In Version: 8
Resolved In Build: b159
Status: RESOLVED
Resolution: Fixed
Fix Version/s: 9
People
Assignee: Volker Simonis
Reporter: Volker Simonis
Votes: 0 Vote for this issue
Watchers: 4 Start watching this issue
Backports
Issue Fix Version Assignee Priority Status Resolution Resolved In Build
JDK-8176681 10 Volker Simonis P3 Resolved Fixed master
Dates
Created: 2017-01-31 23:48
Updated: 3 days ago
Resolved: 2017-02-10 15:48
Description
When we define the same class several times in the same class loader, we end up with multiple instanceKlass objects which will be never deleted (see attached test). The offending code is in SystemDictionary::resolve_from_stream() where we don't take into account that find_or_define_instance_class()/define_instance_class() can throw an exception.
Agile

org.simonis.Crash5

```
$ java -Xbatch -XX:-DoEscapeAnalysis org.simonis.Crash5
# A fatal error has been detected by the Java Runtime Environment:
#
#  SIGSEGV (0xb) at pc=0x00007f8f33fab35d, pid=31451, tid=31469
#
# JRE version: OpenJDK Runtime Environment (9.0) (slowdebug build 9-internal...)
# Java VM: OpenJDK 64-Bit Server VM (slowdebug 9-internal+0-adhoc.simonis...)
# Problematic frame:
# V  [libjvm.so+0xeca35d]  PhaseMacroExpand::generate_arraycopy(...)+0x65b
#
# Core dump will be written. Default location: /tmp/core.31451
#
# An error report file with more information is saved as:
# /tmp/hs_err_pid31451.log
#
# Compiler replay data is saved as:
# /tmp/replay_pid31451.log
```

hs_err_pid31451.log

```
Command Line: -Xbatch -XX:-DoEscapeAnalysis org.simonis.Crash5
...
Current thread (0x00007f8f2c256000): JavaThread "C2 CompilerThread1" daemon ..
...
Current CompileTask:
C2:   1436  116     b  4       org.simonis.Crash5$A::crash (14 bytes)
...
Stack: [0x00007f8ee4457000,0x00007f8ee4558000], sp=0x007f8ee4552630, free=1005k
Native frames: (J=compiled, A=aot compiled, j=interpreted, Vv=VM code, C=native
V  [libjvm.so+0xeca35d]  PhaseMacroExpand::generate_arraycopy(...)+0x65b
V  [libjvm.so+0xece34e]  PhaseMacroExpand::expand_arraycopy_node(...)+0x8a8
V  [libjvm.so+0xec6f59]  PhaseMacroExpand::expand_macro_nodes()+0x733
V  [libjvm.so+0x90c023]  Compile::Optimize()+0xe37
V  [libjvm.so+0x9048b2]  Compile::Compile(...)+0x11d2
V  [libjvm.so+0x7a15d9]  C2Compiler::compile_method(...)+0x14d
V  [libjvm.so+0x923173]  CompileBroker::invoke_compiler_on_method(...)+0x715
V  [libjvm.so+0x922153]  CompileBroker::compiler_thread_loop()+0x2d3
V  [libjvm.so+0x123d762]  compiler_thread_entry(JavaThread*, Thread*)+0x85
```

-XX:+ReplayCompiles

```
$ java -XX:+ReplayCompiles -XX:ReplayDataFile=/tmp/replay_pid31451.log
Resolving klass java/lang/PublicMethods$MethodList at 721
Resolving klass java/lang/reflect/Method at 23
Resolving klass java/lang/reflect/Executable at 95
Resolving klass java/lang/Object at 3
Resolving klass org/simonis/Crash5$X at 16
$ java -XX:+ReplayCompiles -XX:-DoEscapeAnalysis \
                           -XX:ReplayDataFile=/tmp/replay_pid31451.log
Resolving klass java/lang/PublicMethods$MethodList at 721
Resolving klass java/lang/reflect/Method at 23
Resolving klass java/lang/reflect/Executable at 95
Resolving klass java/lang/Object at 3
Resolving klass org/simonis/Crash5$X at 16
#  SIGSEGV (0xb) at pc=0x00007f3f93dff35d, pid=31648, tid=31665
...
# Problematic frame:
# V  [libjvm.so+0xeca35d]  PhaseMacroExpand::generate_arraycopy(...)+0x65b
```

Debugging the Crash

```
$ gdb java
(gdb) b *(&'PhaseMacroExpand::generate_arraycopy' + 0x65b)
Breakpoint 1 at 0x7ffff62db35d: file .../macroArrayCopy.cpp, line 360.
(gdb) run -XX:+ReplayCompiles -XX:-DoEscapeAnalysis \
                              -XX:ReplayDataFile=/tmp/replay_pid31451.log
Breakpoint 1, 0x00007ffff62db35d in PhaseMacroExpand::generate_arraycopy (...) \
           at /OpenJDK/jdk9-dev/hotspot/src/share/vm/opto/macroArrayCopy.cpp:360
360                *(int*)0 = 0xcafebabe;
(gdb) list
354    // Crash demo for JBreak/Jpoint
355    ciKlass* k = _igvn.type(ac->in(ArrayCopyNode::DestKlass))->...->klass();
356    if (k->is_array_klass()) {
357      ciType* t = k->as_array_klass()->base_element_type();
358      if (strcmp(t->name(), "org/simonis/Crash5$X") == 0 ||
359          strcmp(t->name(), "org/simonis/Crash6$X") == 0) {
360        *(int*)0 = 0xcafebabe;
```

org.simonis.Crash5

```
public class Crash5 {

  public static class X {}
  public static class Y extends X {}

  public static class A {
    public static void crash(Object src) {
      System.arraycopy(src, 0, new X[1], 0, 1);
    }
  }

  public static void main(String[] args) {
    for (int i = 0; i < 20_000; i++) {
      A.crash(new Y[1]);
    }
  }
}
```

org.simonis.Crash6

```
$ java -Xbatch -XX:-DoEscapeAnalysis org.simonis.Crash6
# A fatal error has been detected by the Java Runtime Environment:
#
#  SIGSEGV (0xb) at pc=0x00007fe0254c735d, pid=480, tid=499
#
# JRE version: OpenJDK Runtime Environment (9.0) (slowdebug build 9-internal...)
# Java VM: OpenJDK 64-Bit Server VM (slowdebug 9-internal+0-adhoc.simonis...)
# Problematic frame:
# V  [libjvm.so+0xeca35d]  PhaseMacroExpand::generate_arraycopy(...)+0x65b
#
# Core dump will be written. Default location: /tmp/core.480
#
# An error report file with more information is saved as:
# /tmp/hs_err_pid480.log
#
# Compiler replay data is saved as:
# /tmp/replay_pid480.log
```

-XX:+ReplayCompiles

```
$ java -XX:+ReplayCompiles -XX:-DoEscapeAnalysis \
                           -XX:ReplayDataFile=/tmp/replay_pid480.log
java.lang.NoClassDefFoundError: java/lang/invoke/LambdaForm$BMH
Error while parsing line 37: java/lang/invoke/LambdaForm$BMH

java.lang.NoClassDefFoundError: java/lang/invoke/LambdaForm$BMH
Caused by: java.lang.ClassNotFoundException: java.lang.invoke.LambdaForm$BMH
        at jdk.internal.loader.BuiltinClassLoader.loadClass(java.base@9-internal
        at jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(java.base@9
        at java.lang.ClassLoader.loadClass(java.base@9-internal/ClassLoader.java

Failed on java/lang/invoke/LambdaForm$BMH
```

-XX:+ReplayCompiles

```
$ java -XX:+ReplayCompiles -XX:-DoEscapeAnalysis -XX:+ReplayIgnoreInitErrors \
                           -XX:ReplayDataFile=/tmp/replay_pid480.log
java.lang.NoClassDefFoundError: java/lang/invoke/LambdaForm$BMH
Error while parsing line 37: java/lang/invoke/LambdaForm$BMH
java.lang.NoClassDefFoundError: java/lang/invoke/LambdaForm$DMH
Error while parsing line 38: java/lang/invoke/LambdaForm$DMH
...
Resolving klass java/lang/Integer at 41
Resolving klass java/lang/NumberFormatException at 50
Error while parsing line 1231: constant pool length mismatch: wrong class files?

$ java -XX:+ReplayCompiles -XX:-DoEscapeAnalysis -XX:+ReplayIgnoreInitErrors \
       -XX:+PrintCompilation -XX:ReplayDataFile=/tmp/replay_pid480.log
...
Resolving klass java/lang/NumberFormatException at 50
Error while parsing line 1231: constant pool length mismatch: wrong class files?

   1490   17     b  4       org.simonis.Crash6$A::crash (12 bytes)
```

hs_err_pid480.log

```
Classes redefined (1 events):
Event: 10,631 Thread 0x7fe0202128 redefined class org.simonis.Crash6$A, count=1

Events (10 events):
Event: 10,607 loading class java/lang/instrument/ClassDefinition
Event: 10,608 loading class java/lang/instrument/ClassDefinition done
Event: 10,609 loading class org/simonis/Crash6$A
Event: 10,609 loading class org/simonis/Crash6$A done
Event: 10,621 Executing VM operation: RedefineClasses
Event: 10,643 Executing VM operation: RedefineClasses done
Event: 10,661 loading class org/simonis/Crash6$Y
Event: 10,661 loading class org/simonis/Crash6$Y done
Event: 10,755 loading class org/simonis/Crash6$X
Event: 10,755 loading class org/simonis/Crash6$X done
```

org.simonis.Crash6

```
public class Crash6 {
  ...
  public static class A {
    public static void crash(Object src) {
      System.arraycopy(src, 0, new Y[1], 0, 1);
    }
  }
  public static void main(String[] args) throws Exception {
    ...
    byte[] buf = getBytecodes("Crash6$A");
    for (int i = 0; i < buf.length; i++) {
      // Change 'new Y[1]' to 'new X[1]' in A.crash()
      if (buf[i] == 'Y') buf[i] = 'X';
    }
    inst.redefineClasses(new ClassDefinition(A.class, buf));

    for (int i = 0; i < 20_000; i++) {
      A.crash(new Y[1]);
```

Serviceability Agent - Dump Classes from Core

```
$ jhsdb clhsdb
hsdb> attach java core
Opening core file, please wait...
hsdb> buildreplayjars
hsdb> quit
$ ll *.jar
-rw-rw-r-- 1 simonis simonis   90487 Mär 26 21:20 app.jar
-rw-rw-r-- 1 simonis simonis 2396555 Mär 26 21:20 boot.jar
$ java -XX:+ReplayCompiles -XX:-DoEscapeAnalysis -XX:+ReplayIgnoreInitErrors \
       -cp app.jar -XX:ReplayDataFile=/tmp/replay_pid480.log
java.lang.NoClassDefFoundError: java/lang/invoke/LambdaForm$BMH
Error while parsing line 37: java/lang/invoke/LambdaForm$BMH
...
Resolving klass java/lang/System at 19
Resolving klass org/simonis/Crash6$X at 33
#  SIGSEGV (0xb) at pc=0x00007fb2150de35d, pid=1898, tid=1916
#
# V  [libjvm.so+0xeca35d]  PhaseMacroExpand::generate_arraycopy(...)+0x65b
```

Further reading

- Andrei Pangin, JVM crash dump analysis, RigaDevDay 2015
- Андрей Паньгин — Аварийный дамп — «черный ящик» JVM, JPoint 2014
- Андрей Паньгин — Лучший отладчик — сделанный своими руками, Joker 2014
- Java SE 8 Troubleshooting Guide
- Hunt, Beckwith, Parhar, Rutisson - "Java Performance Companion", Addison-Wesley, 2016
- Poonam Bajaj, "HotSpot's Hidden Treasure", Java Magazine, Jul/Aug 2012
- Andrei Pangin, "Build yout own JVM debugging tools", Java Magazine, Jan/Feb 2017
- Kenneth Russell, Lars Bak, "The HotSpot™ Serviceability Agent",
  JVM Research and Technology Symposium, 2001, Monterey

https://github.com/simonis/AnalyzingHotSpotCrashes

https://rawgit.com/simonis/AnalyzingHotSpotCrashes/master/analyzingHotSpotCrashes.xhtml