Advanced Production Debugging

About Me

Co-founder – Takipi, JVM Production Debugging.

Director, AutoCAD Web & Mobile.

Software Architect at IAI Aerospace.

Coding for the past 16 years - C++, Delphi, .NET, Java.

Focus on real-time, scalable systems.

Blogs at blog.takipi.com

Overview

Dev-stage debugging is forward-tracing.

Production debugging is focused on backtracing.

Modern production debugging poses two challenges:

- **State isolation.**
- **Data distribution.**

Agenda

1. Logging at scale.

2. Preemptive jstacks

1. Extracting state with Btrace

1. Extracting state with custom Java agents.

Best Logging Practices

A primary new consumer is a log analyzer. Context trumps content.

1. Code context.
2. Time + duration.
3. **Transactional data** (for async & distributed debugging).

Transactional IDs

- Modern logging is done over a multi-threads / processes.
- Generate a UUID at every thread entry point into your app – the transaction ID.
- Append the ID into each log entry.
- Try to maintain it across machines – critical for debugging **Reactive and microservice apps.**

```
[20-07 07:32:51][BRT   -1473 -S4247] ERROR - Unable to retrieve data for
Job J141531. {CodeAnalysisUtil TID: Uu7XoelHfCTUUlvol6d2a9pU}
[SQS-prod_taskforce1_BRT-Executor-1-thread-2]
```

Logging Performance

1. Don’t catch exceptions within loops and log them (*implicit* and explicit).

For long running loops this will flood the log, impede performance and bring a server down.

```
void readData {

  while (hasNext()) {

  try {
    readData();
  }
  catch {Exception  e) {
    logger.errror("error reading " X + " from " Y, e);
  }
}
```

2. Do not log Object.toString(), especially collections.
Can create an implicit loop. If needed - make sure length is limited.

Thread Names

- Thread *name* is a mutable property.
- Can be set to hold transaction specific state.
- Some frameworks (e.g. EJB) don’t like that.
- Can be super helpful when debugging in tandem with **jstack**.

Thread Names (2)

For example:

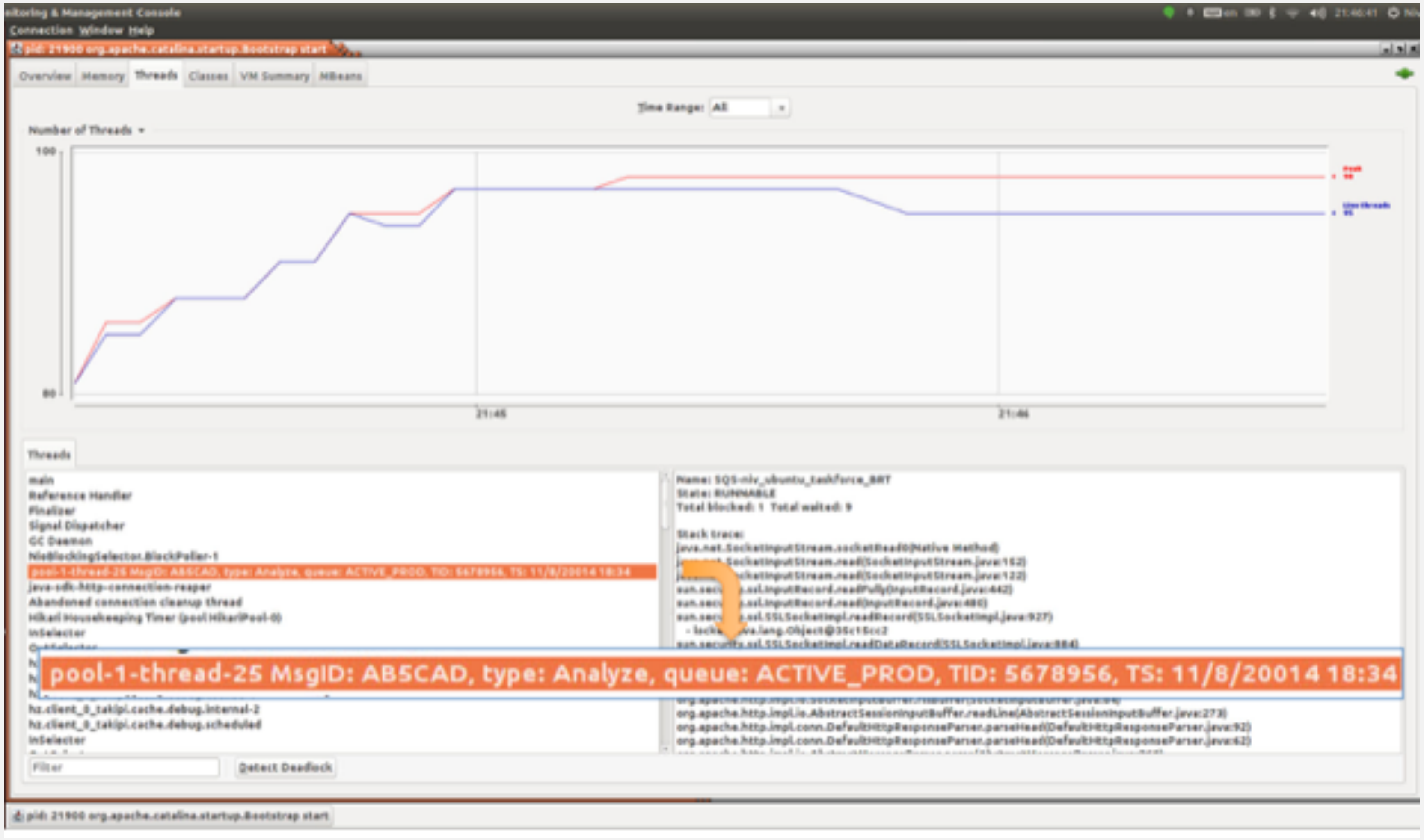
```
Thread.currentThread().setName(
    Context + TID + Params + current Time, ...);
```

Before:

```
"pool-1-thread-1" #17 prio=5 os_prio=31
tid=0x00007f9d620c9800 nid=0x6d03 in Object.wait()
[0x000000013ebcc000
```

After:

```
"Queue Processing Thread, MessageID: AB5CAD, type:
AnalyzeGraph, queue: ACTIVE_PROD, Transaction_ID: 5678956,
Start Time: 10/8/2014 18:34" #17 prio=5 os_prio=31
tid=0x00007f9d620c9800 nid=0x6d03 in Object.wait()
[0x000000013ebcc000]
```

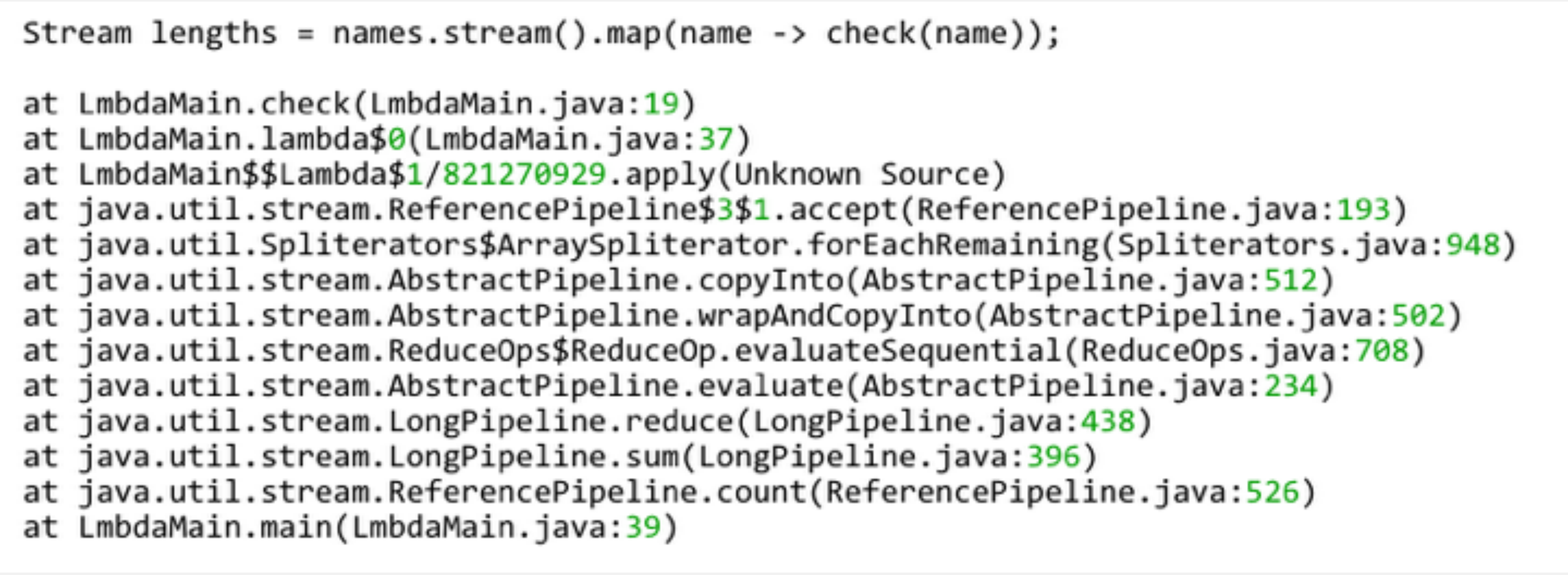


```
main
Reference Handler
Finalizer
Signal Dispatcher
GC Daemon
NioBlockingSelector.BlockPoller-1
pool-1-thread-25 MsgID: AB5CAD, type: Analyze, queue: ACTIVE_PROD, TID: 5678956, TS: 11/8/20014 18:34
java-sdk-http-connection-reaper
Abandoned connection cleanup thread
Hikari Housekeeping Timer (pool HikariPool-0)
InSelector
hz.client_0_takipi.cache.debug.internal-2
hz.client_0_takipi.cache.debug.scheduled
InSelector
```

```
Name: SQS-nlv_ubuntu_taskforce_BRT
State: RUNNABLE
Total blocked: 1  Total waited: 9

Stack trace:
java.net.SocketInputStream.socketRead0(Native Method)
j        SocketInputStream.read(SocketInputStream.java:152)
j        ocketInputStream.read(SocketInputStream.java:122)
sun.sec      .ssl.InputRecord.readFully(InputRecord.java:442)
sun.sec      .ssl.InputRecord.read(InputRecord.java:480)
sun.sec      .ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:927)
  - lock      va.lang.Object@35c15cc2
sun.security.ssl.SSLSocketImpl.readDataRecord(SSLSocketImpl.java:884)
org.apache.http.impl.io.AbstractSessionInputBuffer.readLine(AbstractSessionInputBuffer.java:273)
org.apache.http.impl.conn.DefaultHttpResponseParser.parseHead(DefaultHttpResponseParser.java:92)
org.apache.http.impl.conn.DefaultHttpResponseParser.parseHead(DefaultHttpResponseParser.java:62)
```

pool-1-thread-25 MsgID: AB5CAD, type: Analyze, queue: ACTIVE_PROD, TID: 5678956, TS: 11/8/20014 18:34

Modern Stacks - Java 8

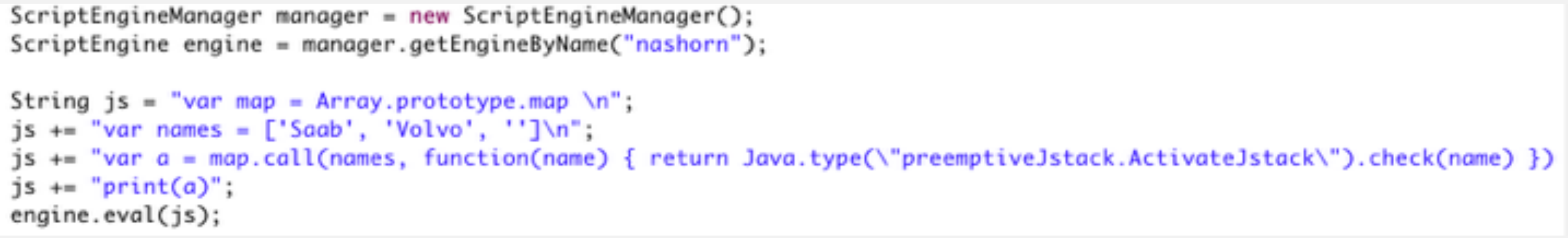


```
Stream lengths = names.stream().map(name -> check(name));

at LmbdaMain.check(LmbdaMain.java:19)
at LmbdaMain.lambda$0(LmbdaMain.java:37)
at LmbdaMain$$Lambda$1/821270929.apply(Unknown Source)
at java.util.stream.ReferencePipeline$3$1.accept(ReferencePipeline.java:193)
at java.util.Spliterators$ArraySpliterator.forEachRemaining(Spliterators.java:948)
at java.util.stream.AbstractPipeline.copyInto(AbstractPipeline.java:512)
at java.util.stream.AbstractPipeline.wrapAndCopyInto(AbstractPipeline.java:502)
at java.util.stream.ReduceOps$ReduceOp.evaluateSequential(ReduceOps.java:708)
at java.util.stream.AbstractPipeline.evaluate(AbstractPipeline.java:234)
at java.util.stream.LongPipeline.reduce(LongPipeline.java:438)
at java.util.stream.LongPipeline.sum(LongPipeline.java:396)
at java.util.stream.ReferencePipeline.count(ReferencePipeline.java:526)
at LmbdaMain.main(LmbdaMain.java:39)
```

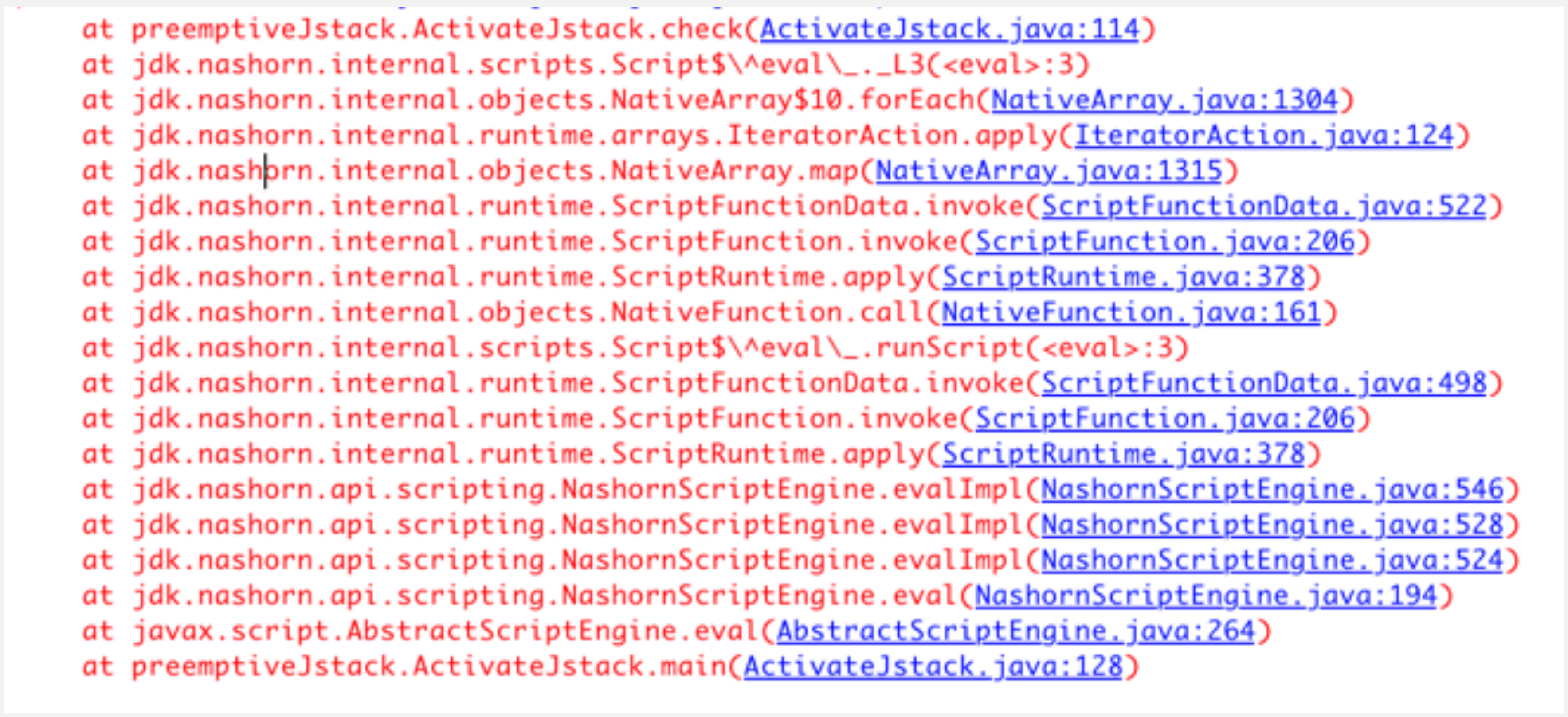
Modern Stacks - Scala

```scala
val lengths = names.map(name => check(name.length))
```

```
at Main$.check(Main.scala:6)
at Main$$anonfun$1.apply(Main.scala:12)
at Main$$anonfun$1.apply(Main.scala:12)
at scala.collection.TraversableLike$$anonfun$map$1.apply(TraversableLike.scala:244)
at scala.collection.TraversableLike$$anonfun$map$1.apply(TraversableLike.scala:244)
at scala.collection.immutable.List.foreach(List.scala:318)
at scala.collection.TraversableLike$class.map(TraversableLike.scala:244)
at scala.collection.AbstractTraversable.map(Traversable.scala:105)
at Main$delayedInit$body.apply(Main.scala:12)
at scala.Function0$class.apply$mcV$sp(Function0.scala:40)
at scala.runtime.AbstractFunction0.apply$mcV$sp(AbstractFunction0.scala:12)
at scala.App$$anonfun$main$1.apply(App.scala:71)
at scala.App$$anonfun$main$1.apply(App.scala:71)
at scala.collection.immutable.List.foreach(List.scala:318)
at scala.collection.generic.TraversableForwarder$class.foreach(TraversableForwarder.scala:3
at scala.App$class.main(App.scala:71)
at Main$.main(Main.scala:1)
at Main.main(Main.scala)
```



```java
ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine engine = manager.getEngineByName("nashorn");

String js = "var map = Array.prototype.map \n";
js += "var names = ['Saab', 'Volvo', '']\n";
js += "var a = map.call(names, function(name) { return Java.type(\"preemptiveJstack.ActivateJstack\").check(name) })
js += "print(a)";
engine.eval(js);
```



```
    at preemptiveJstack.ActivateJstack.check(ActivateJstack.java:114)
    at jdk.nashorn.internal.scripts.Script$\^eval\_._L3(<eval>:3)
    at jdk.nashorn.internal.objects.NativeArray$10.forEach(NativeArray.java:1304)
    at jdk.nashorn.internal.runtime.arrays.IteratorAction.apply(IteratorAction.java:124)
    at jdk.nashorn.internal.objects.NativeArray.map(NativeArray.java:1315)
    at jdk.nashorn.internal.runtime.ScriptFunctionData.invoke(ScriptFunctionData.java:522)
    at jdk.nashorn.internal.runtime.ScriptFunction.invoke(ScriptFunction.java:206)
    at jdk.nashorn.internal.runtime.ScriptRuntime.apply(ScriptRuntime.java:378)
    at jdk.nashorn.internal.objects.NativeFunction.call(NativeFunction.java:161)
    at jdk.nashorn.internal.scripts.Script$\^eval\_.runScript(<eval>:3)
    at jdk.nashorn.internal.runtime.ScriptFunctionData.invoke(ScriptFunctionData.java:498)
    at jdk.nashorn.internal.runtime.ScriptFunction.invoke(ScriptFunction.java:206)
    at jdk.nashorn.internal.runtime.ScriptRuntime.apply(ScriptRuntime.java:378)
    at jdk.nashorn.api.scripting.NashornScriptEngine.evalImpl(NashornScriptEngine.java:546)
    at jdk.nashorn.api.scripting.NashornScriptEngine.evalImpl(NashornScriptEngine.java:528)
    at jdk.nashorn.api.scripting.NashornScriptEngine.evalImpl(NashornScriptEngine.java:524)
    at jdk.nashorn.api.scripting.NashornScriptEngine.eval(NashornScriptEngine.java:194)
    at javax.script.AbstractScriptEngine.eval(AbstractScriptEngine.java:264)
    at preemptiveJstack.ActivateJstack.main(ActivateJstack.java:128)
```

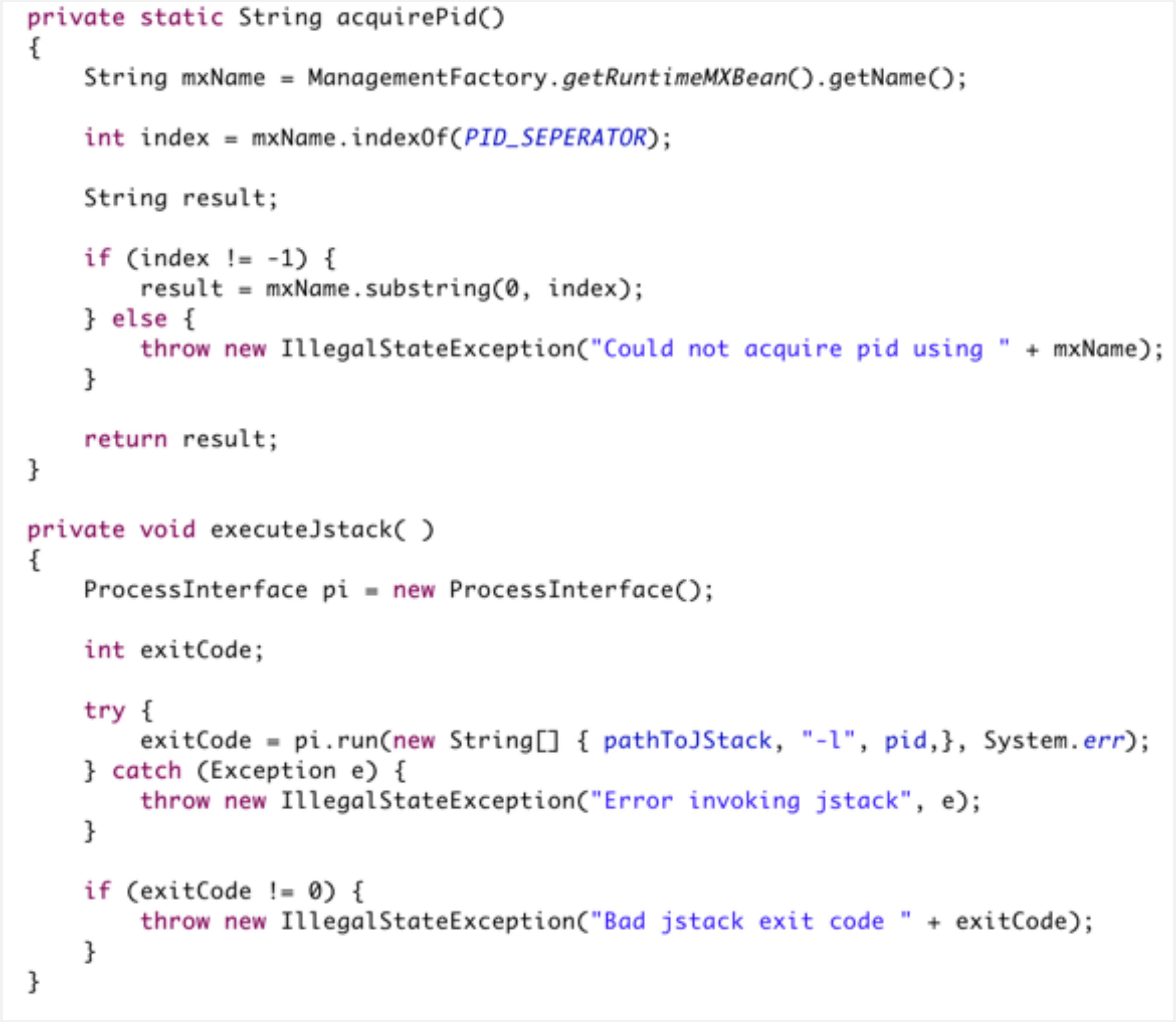
Preemptive jstack

github.com/takipi/jstack

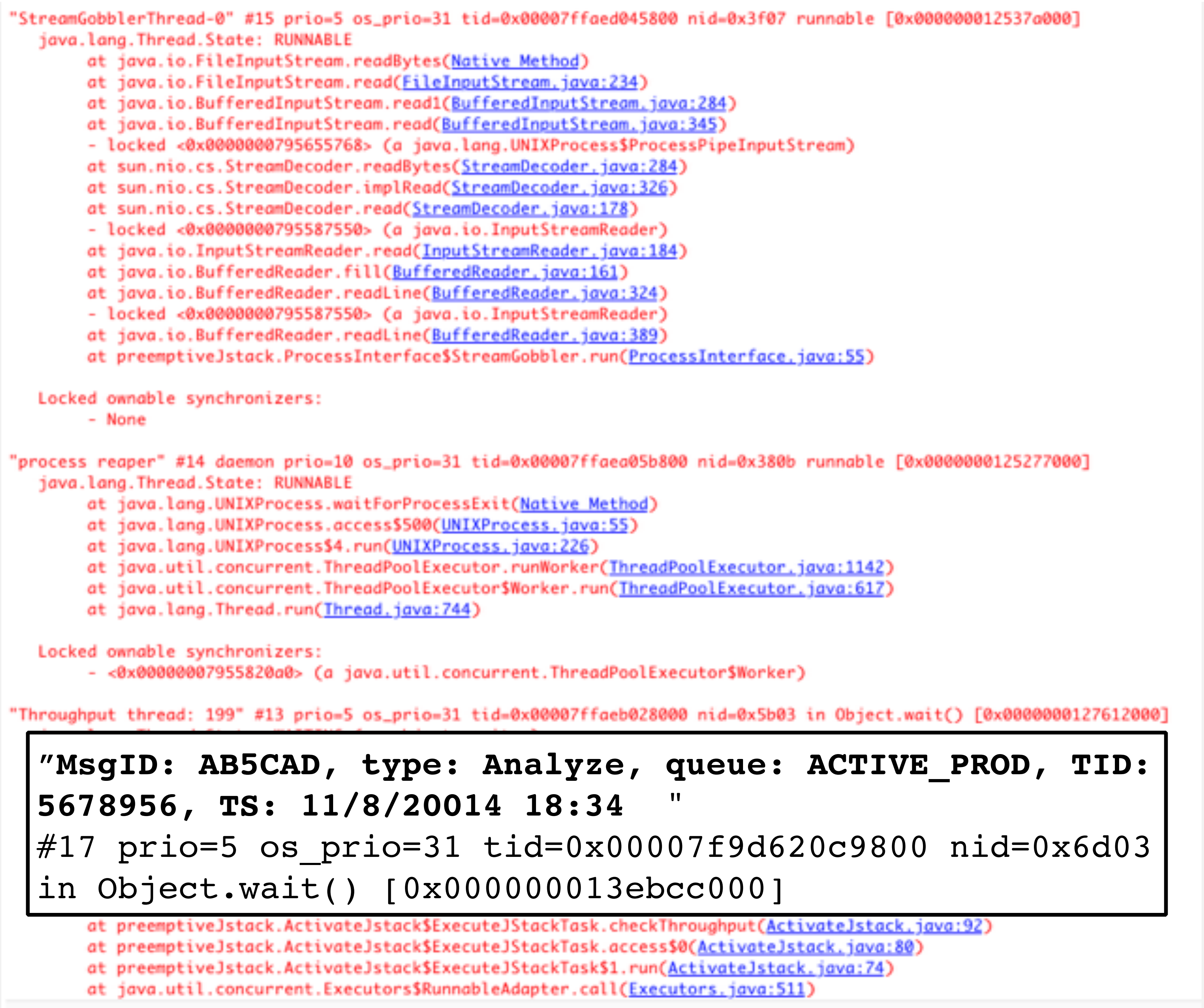
Preemptive jstack

- A production debugging foundation.
- Presents two issues –
  - Activated only in retrospect.
  - **No state:** does not provide any variable state.
- Let’s see how we can overcome these with preemptive jstacks.

```java
public void startScheduleTask() {

    scheduler.scheduleAtFixedRate(new Runnable() {
        public void run() {

            checkThroughput();

            }
        }, APP_WARMUP, POLLING_CYCLE, TimeUnit.SECONDS);
}

private void checkThroughput()
{
    if (adder.intValue() == -1)
    {
        return;
    }

    int value = adder.intValue();

    if (value < MIN_THROUGHPUT) {
        Thread.currentThread().setName("Throughput thread: " + value);
        System.err.println("Minimal throughput failed: exexuting jstack");
        executeJstack();
    }

    adder.reset();
}

public void incThrughput(int val) {
    adder.add(val);
}

public int throughput()
{
    return adder.intValue();
}
```



```java
private static String acquirePid()
{
    String mxName = ManagementFactory.getRuntimeMXBean().getName();

    int index = mxName.indexOf(PID_SEPERATOR);

    String result;

    if (index != -1) {
        result = mxName.substring(0, index);
    } else {
        throw new IllegalStateException("Could not acquire pid using " + mxName);
    }

    return result;
}

private void executeJstack( )
{
    ProcessInterface pi = new ProcessInterface();

    int exitCode;

    try {
        exitCode = pi.run(new String[] { pathToJStack, "-l", pid,}, System.err);
    } catch (Exception e) {
        throw new IllegalStateException("Error invoking jstack", e);
    }

    if (exitCode != 0) {
        throw new IllegalStateException("Bad jstack exit code " + exitCode);
    }
}
```



```
"StreamGobblerThread-0" #15 prio=5 os_prio=31 tid=0x00007ffaed045800 nid=0x3f07 runnable [0x000000012537a000]
   java.lang.Thread.State: RUNNABLE
        at java.io.FileInputStream.readBytes(Native Method)
        at java.io.FileInputStream.read(FileInputStream.java:234)
        at java.io.BufferedInputStream.read1(BufferedInputStream.java:284)
        at java.io.BufferedInputStream.read(BufferedInputStream.java:345)
        - locked <0x0000000795655768> (a java.lang.UNIXProcess$ProcessPipeInputStream)
        at sun.nio.cs.StreamDecoder.readBytes(StreamDecoder.java:284)
        at sun.nio.cs.StreamDecoder.implRead(StreamDecoder.java:326)
        at sun.nio.cs.StreamDecoder.read(StreamDecoder.java:178)
        - locked <0x0000000795587550> (a java.io.InputStreamReader)
        at java.io.InputStreamReader.read(InputStreamReader.java:184)
        at java.io.BufferedReader.fill(BufferedReader.java:161)
        at java.io.BufferedReader.readLine(BufferedReader.java:324)
        - locked <0x0000000795587550> (a java.io.InputStreamReader)
        at java.io.BufferedReader.readLine(BufferedReader.java:389)
        at preemptiveJstack.ProcessInterface$StreamGobbler.run(ProcessInterface.java:55)

   Locked ownable synchronizers:
        - None

"process reaper" #14 daemon prio=10 os_prio=31 tid=0x00007ffaea05b800 nid=0x380b runnable [0x0000000125277000]
   java.lang.Thread.State: RUNNABLE
        at java.lang.UNIXProcess.waitForProcessExit(Native Method)
        at java.lang.UNIXProcess.access$500(UNIXProcess.java:55)
        at java.lang.UNIXProcess$4.run(UNIXProcess.java:226)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
        at java.lang.Thread.run(Thread.java:744)

   Locked ownable synchronizers:
        - <0x00000007955820a0> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"Throughput thread: 199" #13 prio=5 os_prio=31 tid=0x00007ffaeb028000 nid=0x5b03 in Object.wait() [0x0000000127612000]
```

```
"MsgID: AB5CAD, type: Analyze, queue: ACTIVE_PROD, TID:
5678956, TS: 11/8/20014 18:34  "
#17 prio=5 os_prio=31 tid=0x00007f9d620c9800 nid=0x6d03
in Object.wait() [0x000000013ebcc000]
```

```
        at preemptiveJstack.ActivateJstack$ExecuteJStackTask.checkThroughput(ActivateJstack.java:92)
        at preemptiveJstack.ActivateJstack$ExecuteJStackTask.access$0(ActivateJstack.java:80)
        at preemptiveJstack.ActivateJstack$ExecuteJStackTask$1.run(ActivateJstack.java:74)
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
```

Jstack Triggers

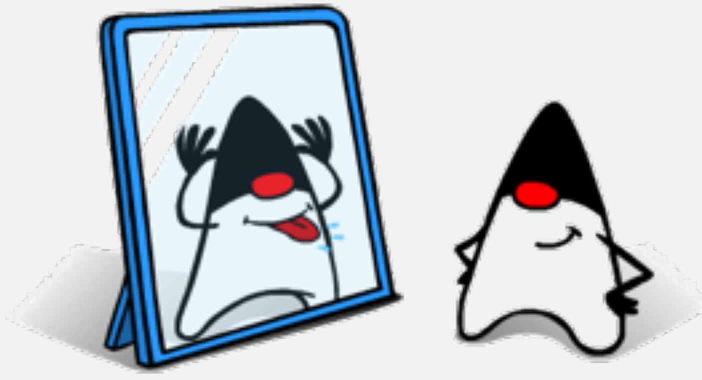
- A queue exceeds capacity.
- Throughput exceeds or drops below a threshold.
- CPU usage passes a threshold.
- Locking failures / Deadlock.

Integrate as a first class citizen with your logging infrastructure.

```java
public static class DeadLockDetector extends Thread {
    @Override
    public void run() {
        while (true) {

            ThreadMXBean threadMXBean = ManagementFactory.getThreadMXBean();
            long[] ids = threadMXBean.findDeadlockedThreads();

            if (ids != null) {
                System.out.println("Deadlocked threads = " + ids);
                ThreadInfo[] threadInfos = threadMXBean.getThreadInfo(ids);

                for (ThreadInfo threadInfo : threadInfos)
                {
                    System.out.println("Deadlocked threads info = "
                            + threadInfo.getBlockedTime() + " " + threadInfo.getLockName());
                }

                activateJStack();
            }
            try {
                Thread.sleep(1000);
            }
            catch (InterruptedException e) { //LOG
            }
        }
    }
}
```

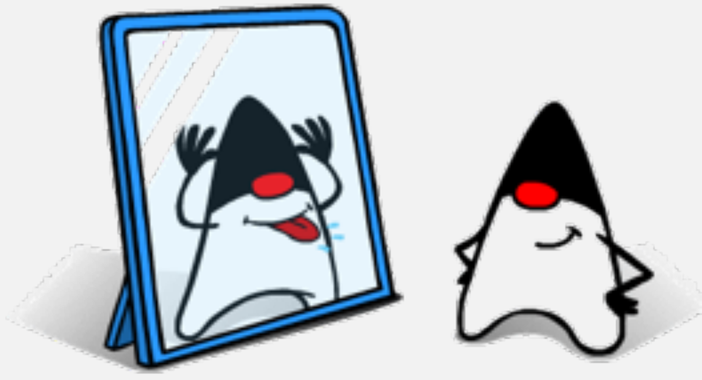
BTrace

- An advanced open-source tool for extracting state from a live JVM.
- Uses a Java agent and a meta-scripting language to capture state.
- **Pros**: Lets you probe variable state without modifying / restarting the JVM.
- **Cons:** read-only querying using a custom syntax and libraries.



Usage

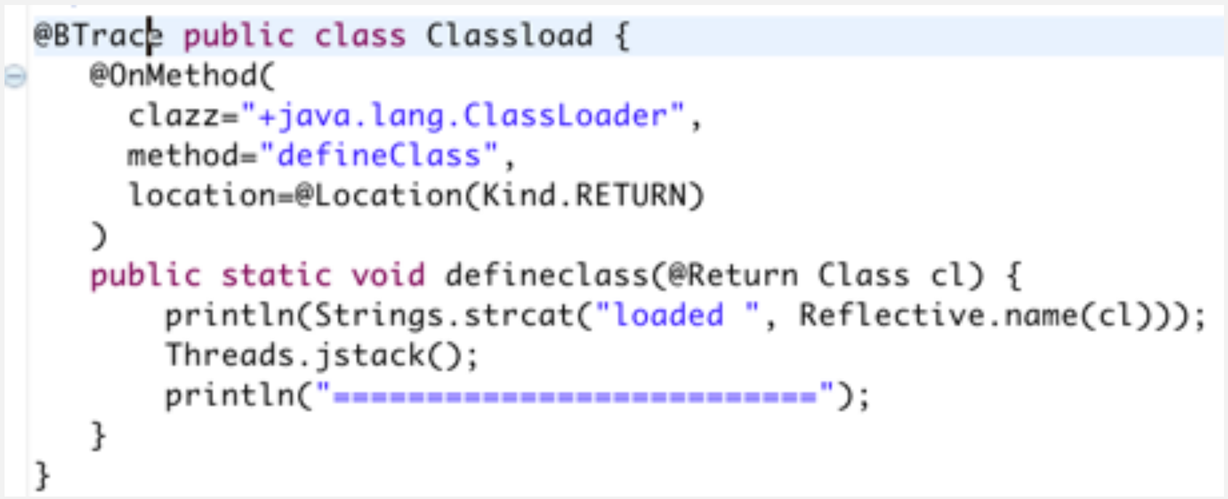
- No JVM restart needed. Works remotely.
- *btrace [-I <include-path>] [-p <port>] [-cp <classpath>] <pid> <btrace-script> [<args>]*
- Example: Btrace 9550 myScript.java
- Available at: kenai.com/projects/btrace



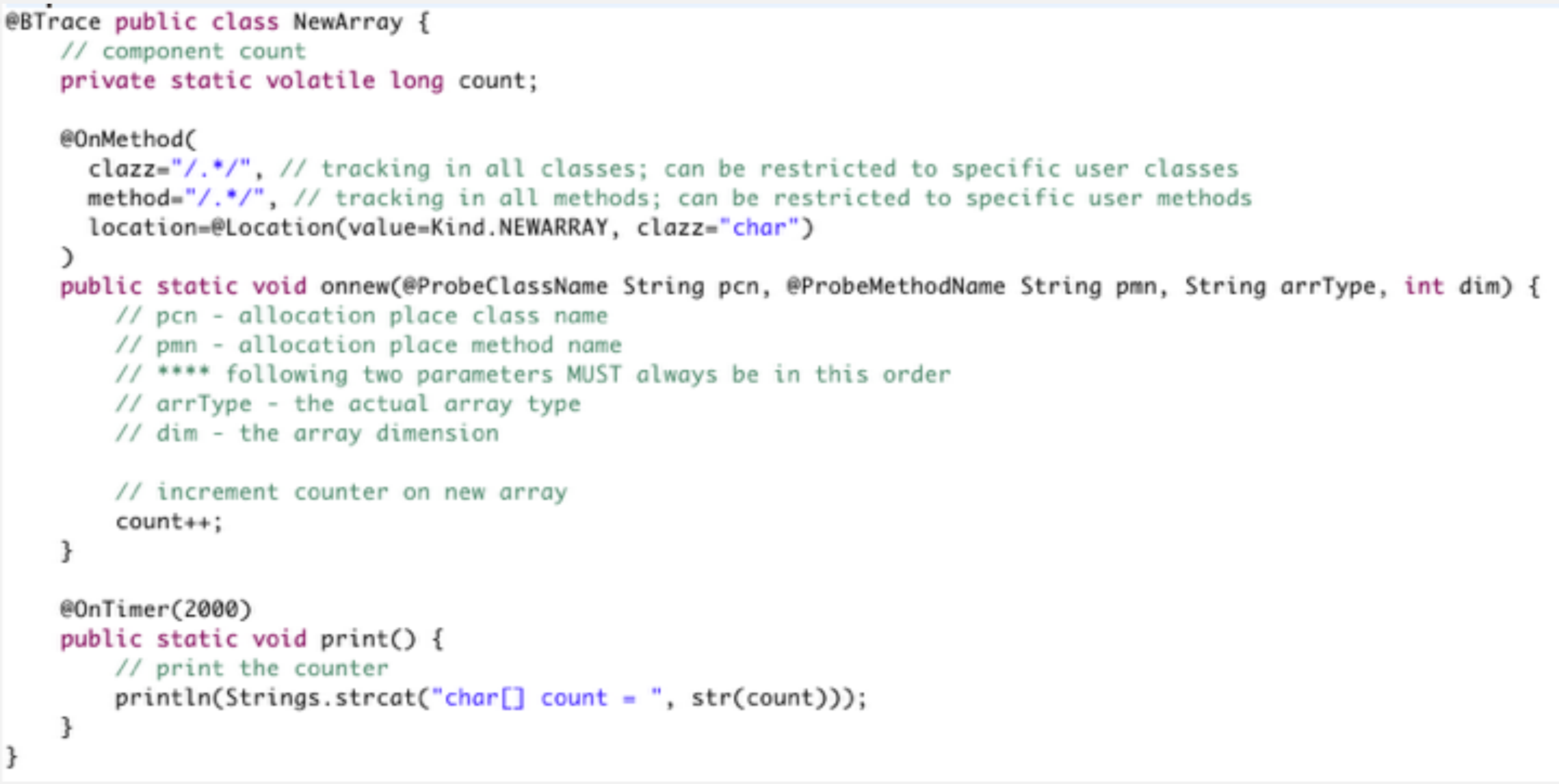
BTrace - Restrictions

- Can not create new objects.
- Can not create new arrays.
- Can not throw exceptions.
- Can not catch exceptions.
- Can not make arbitrary instance or static method calls - only the public static methods of com.sun.btrace.BTraceUtils class may be called from a BTrace program.
- Can not assign to static or instance fields of target program's classes and objects. But, BTrace class can assign to it's own static fields ("trace state" can be mutated).
- Can not have instance fields and methods. Only static public void returning methods are allowed for a BTrace class. And all fields have to be static.
- Can not have outer, inner, nested or local classes.
- Can not have synchronized blocks or synchronized methods.
- can not have loops (for, while, do..while)
- Can not extend arbitrary class (super class has to be java.lang.Object)
- Can not implement interfaces.
- Can not contains assert statements.
- Can not use class literals

```
@BTrace public class FileTracker {
    @TLS private static String name;

    @OnMethod(
        clazz="java.io.FileInputStream",
        method="<init>"
    )
    public static void onNewFileInputStream(@Self FileInputStream self, File f) {
        name = Strings.str(f);
    }

    @OnMethod(
        clazz="java.io.FileInputStream",
        method="<init>",
        type="void (java.io.File)",
        location=@Location(Kind.RETURN)
    )
    public static void onNewFileInputStreamReturn() {
        if (name != null) {
            println(Strings.strcat("opened for read ", name));
            name = null;
        }
    }

    @OnMethod(
        clazz="java.io.FileOutputStream",
        method="<init>"
    )
    public static void onNewFileOutputStream(@Self FileOutputStream self, File f, boolean b) {
        name = str(f);
    }

    @OnMethod(
        clazz="java.io.FileOutputStream",
        method="<init>",
        type="void (java.io.File, boolean)",
        location=@Location(Kind.RETURN)
    )
    public static void OnNewFileOutputStreamReturn() {
        if (name != null) {
            println(Strings.strcat("opened for write ", name));
            name = null;
        }
    }
```



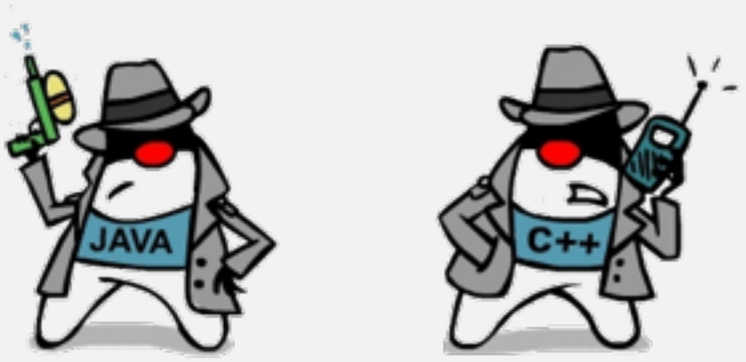
```java
@BTrace public class Classload {
    @OnMethod(
      clazz="+java.lang.ClassLoader",
      method="defineClass",
      location=@Location(Kind.RETURN)
    )
    public static void defineclass(@Return Class cl) {
        println(Strings.strcat("loaded ", Reflective.name(cl)));
        Threads.jstack();
        println("==========================");
    }
}
```



```
@BTrace public class NewArray {
    // component count
    private static volatile long count;

    @OnMethod(
      clazz="/.*/", // tracking in all classes; can be restricted to specific user classes
      method="/.*/", // tracking in all methods; can be restricted to specific user methods
      location=@Location(value=Kind.NEWARRAY, clazz="char")
    )
    public static void onnew(@ProbeClassName String pcn, @ProbeMethodName String pmn, String arrType, int dim) {
        // pcn - allocation place class name
        // pmn - allocation place method name
        // **** following two parameters MUST always be in this order
        // arrType - the actual array type
        // dim - the array dimension

        // increment counter on new array
        count++;
    }

    @OnTimer(2000)
    public static void print() {
        // print the counter
        println(Strings.strcat("char[] count = ", str(count)));
    }
}
```

Java Agents

- An advanced technique for instrumenting code dynamically.
- The foundation of modern profiling / debugging tools.
- Two types of agents: Java and Native.
- **Pros**: extremely powerful technique to collect state from a live app.
- **Cons**: requires knowledge of creating *verifiable* bytecode.



Agent Types

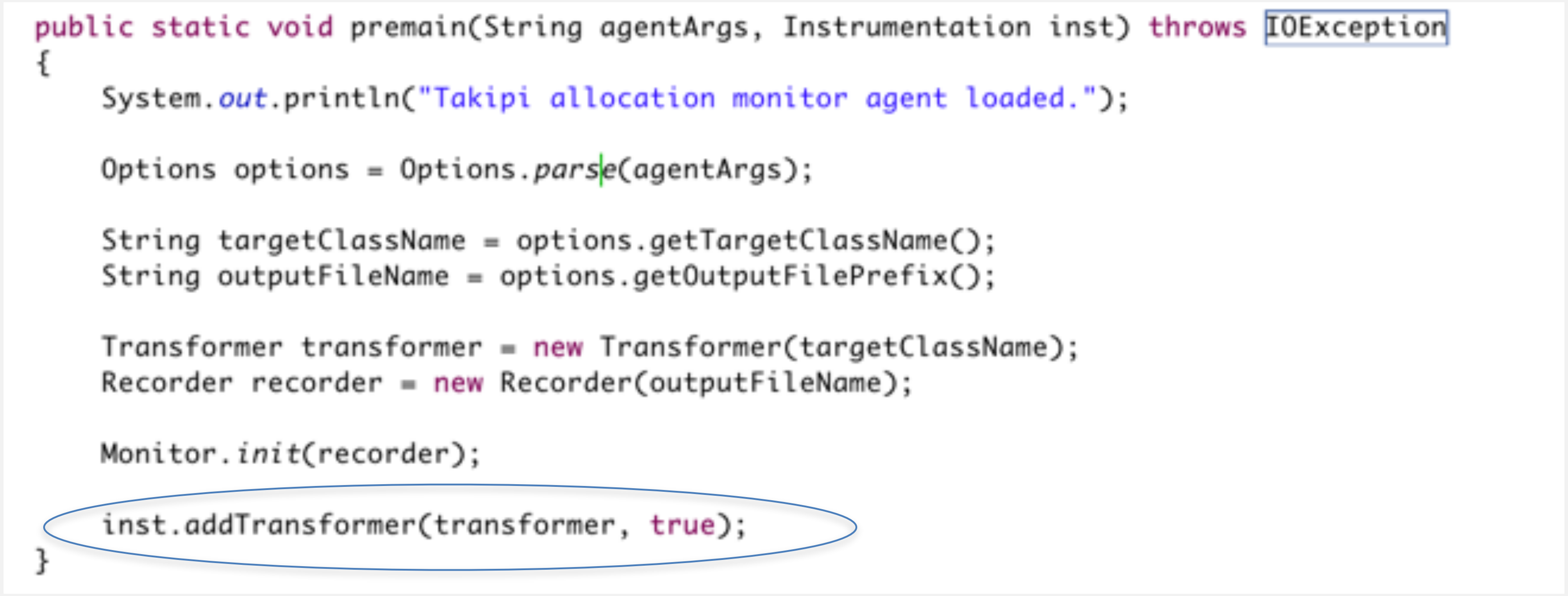
- Java agents are written in Java. Have access to the *Instrumentation* BCI API.
- Native agents – written in C++.
- Have access to JVMTI – the JVM's low-level set of APIs and capabilities.
  - JIT compilation, Garbage Collection, Monitor acquisition, Exception callbacks, ..
- More complex to write.
- Platform dependent.

Java Agents

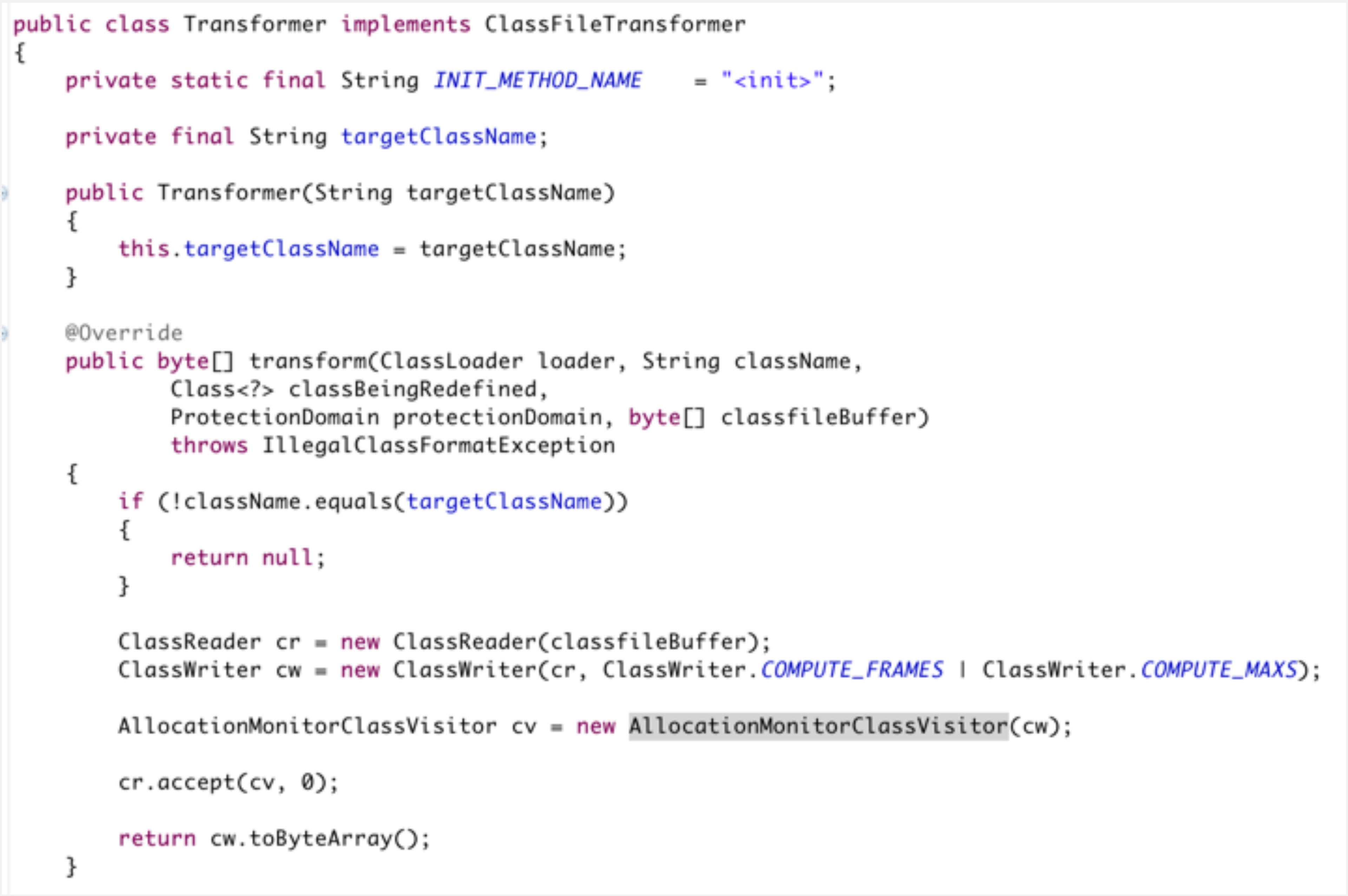
github.com/takipi/debugAgent

Attach at startup: *`java -Xmx2G -agentlib:myAgent -jar myapp.jar start`*

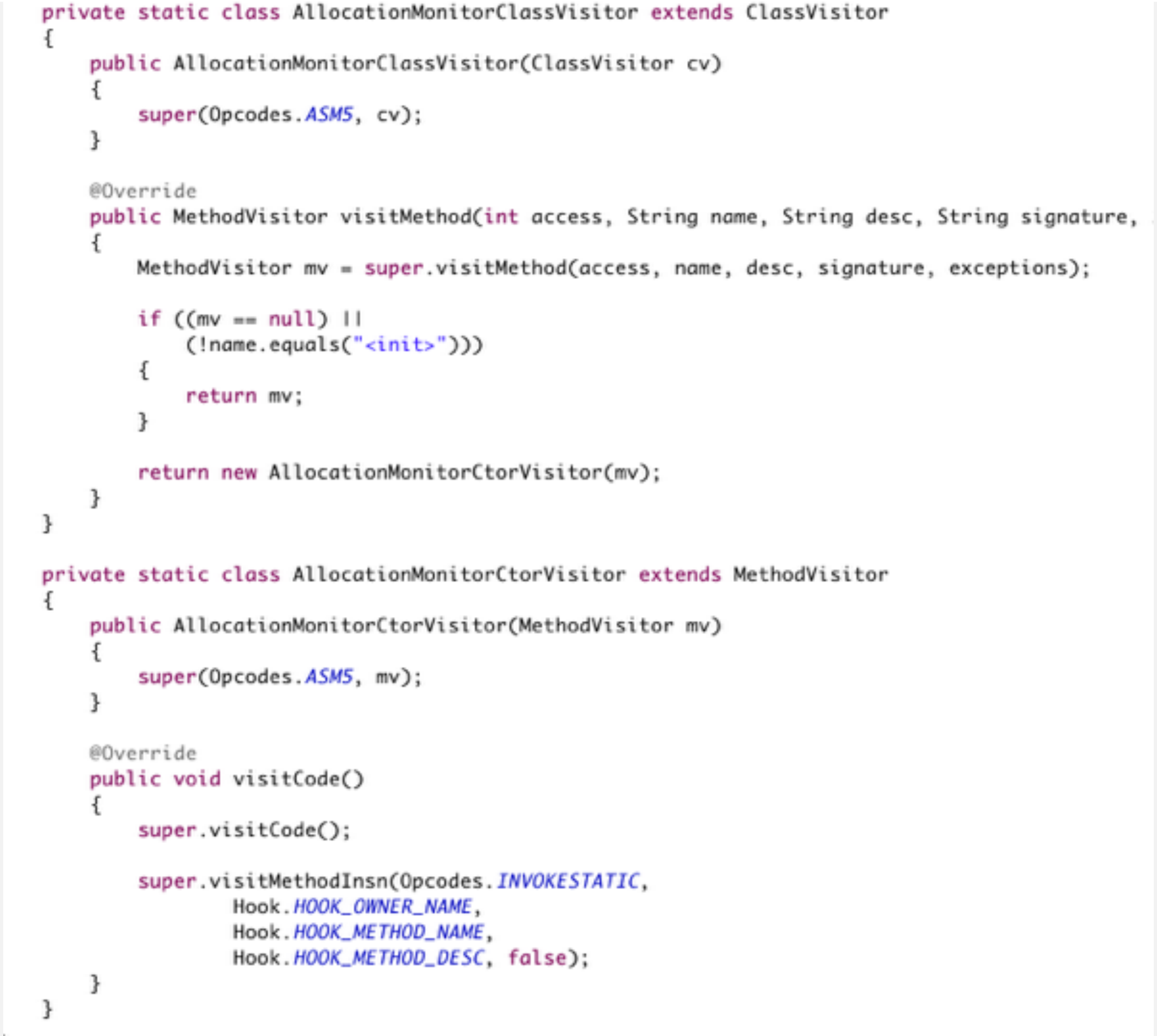
To a live JVM using: *com.sun.tools.attach.VirtualMachine* Attach API.



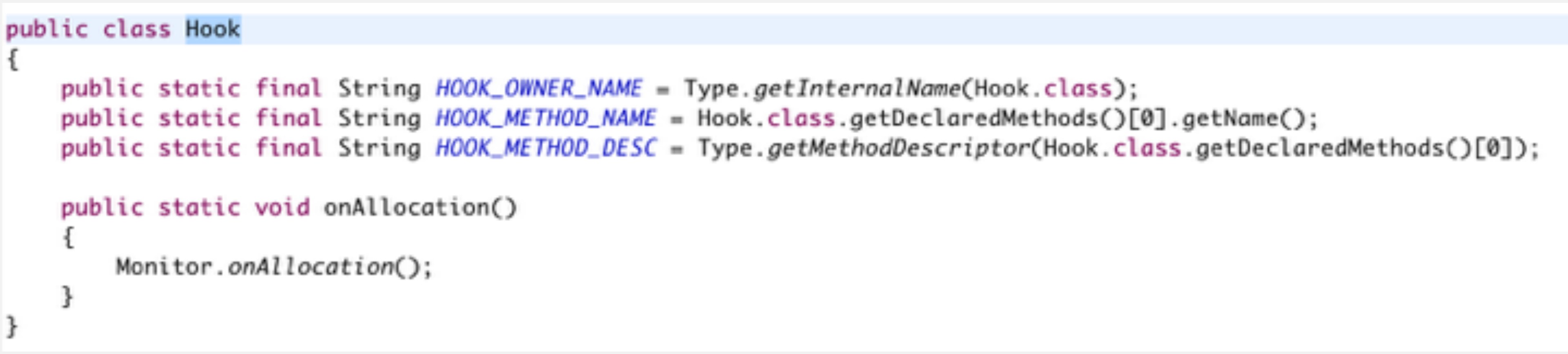
```
public static void premain(String agentArgs, Instrumentation inst) throws IOException
{
    System.out.println("Takipi allocation monitor agent loaded.");

    Options options = Options.parse(agentArgs);

    String targetClassName = options.getTargetClassName();
    String outputFileName = options.getOutputFilePrefix();

    Transformer transformer = new Transformer(targetClassName);
    Recorder recorder = new Recorder(outputFileName);

    Monitor.init(recorder);

    inst.addTransformer(transformer, true);
}
```



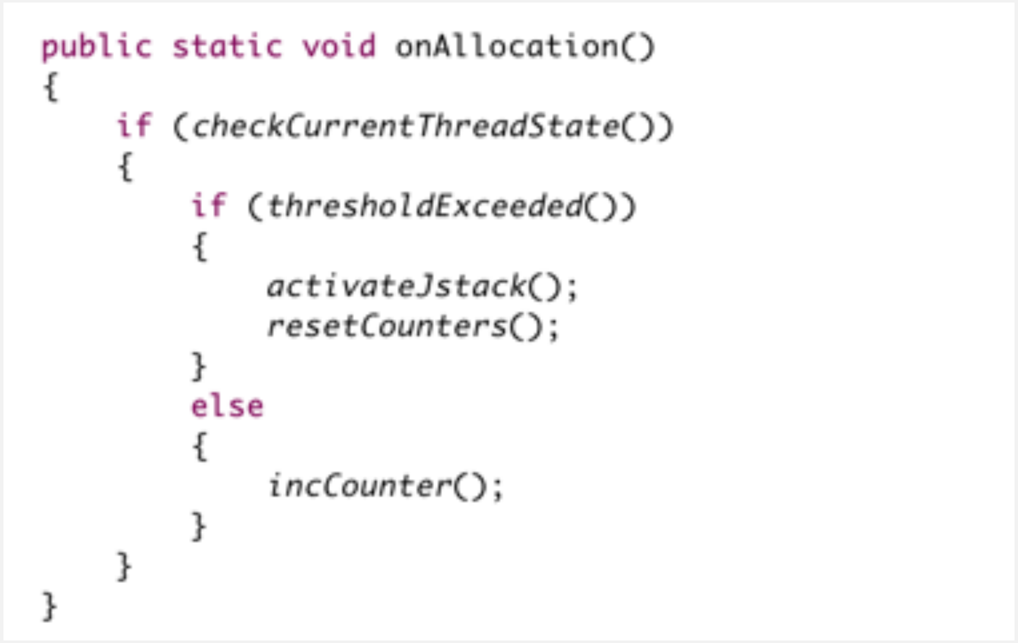
```java
public class Transformer implements ClassFileTransformer
{
    private static final String INIT_METHOD_NAME    = "<init>";

    private final String targetClassName;

    public Transformer(String targetClassName)
    {
        this.targetClassName = targetClassName;
    }

    @Override
    public byte[] transform(ClassLoader loader, String className,
            Class<?> classBeingRedefined,
            ProtectionDomain protectionDomain, byte[] classfileBuffer)
            throws IllegalClassFormatException
    {
        if (!className.equals(targetClassName))
        {
            return null;
        }

        ClassReader cr = new ClassReader(classfileBuffer);
        ClassWriter cw = new ClassWriter(cr, ClassWriter.COMPUTE_FRAMES | ClassWriter.COMPUTE_MAXS);

        AllocationMonitorClassVisitor cv = new AllocationMonitorClassVisitor(cw);

        cr.accept(cv, 0);

        return cw.toByteArray();
    }
```



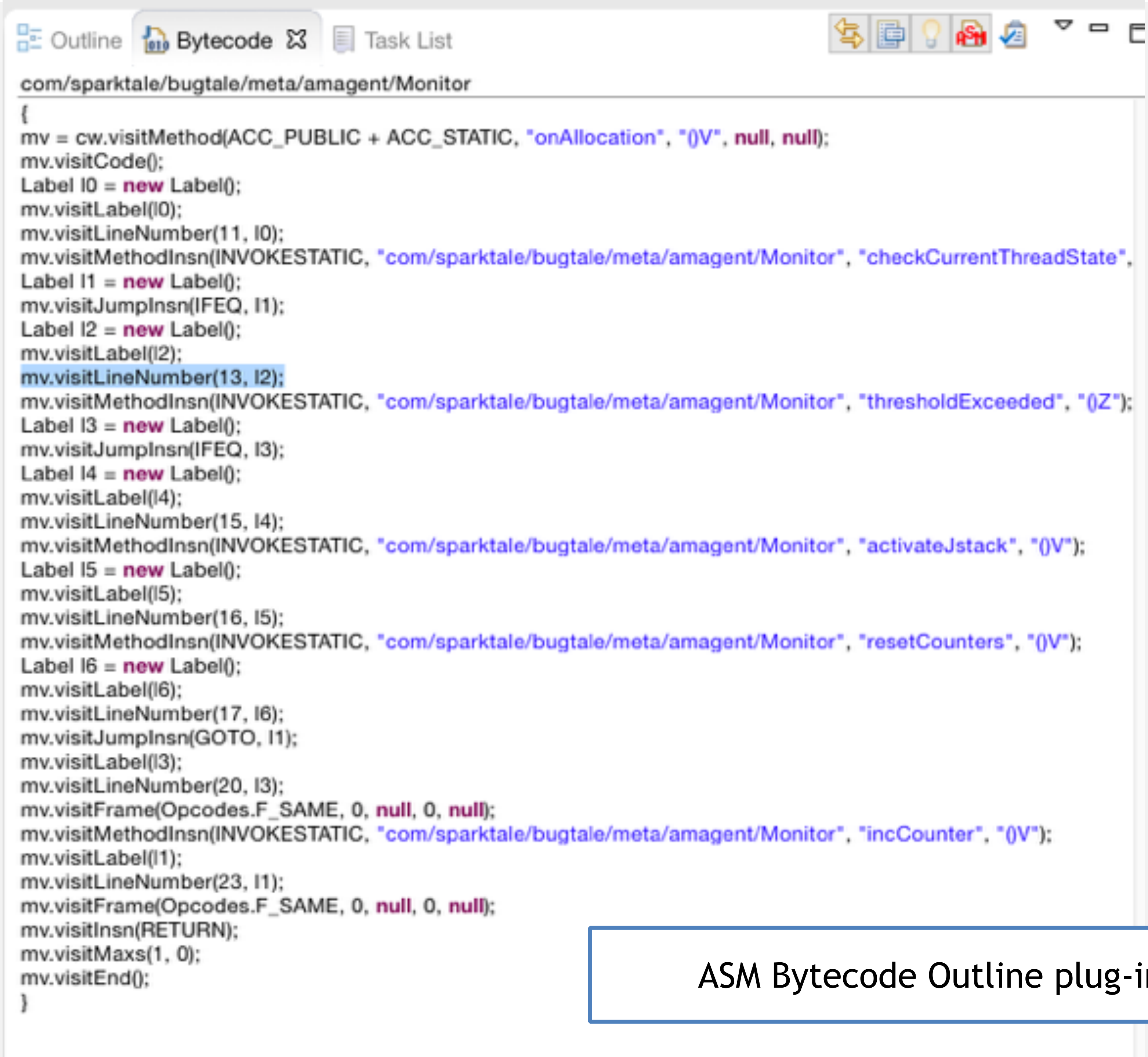
```
private static class AllocationMonitorClassVisitor extends ClassVisitor
{
    public AllocationMonitorClassVisitor(ClassVisitor cv)
    {
        super(Opcodes.ASM5, cv);
    }

    @Override
    public MethodVisitor visitMethod(int access, String name, String desc, String signature,
    {
        MethodVisitor mv = super.visitMethod(access, name, desc, signature, exceptions);

        if ((mv == null) ||
            (!name.equals("<init>")))
        {
            return mv;
        }

        return new AllocationMonitorCtorVisitor(mv);
    }
}

private static class AllocationMonitorCtorVisitor extends MethodVisitor
{
    public AllocationMonitorCtorVisitor(MethodVisitor mv)
    {
        super(Opcodes.ASM5, mv);
    }

    @Override
    public void visitCode()
    {
        super.visitCode();

        super.visitMethodInsn(Opcodes.INVOKESTATIC,
                Hook.HOOK_OWNER_NAME,
                Hook.HOOK_METHOD_NAME,
                Hook.HOOK_METHOD_DESC, false);
    }
}
```



```java
public class Hook
{
    public static final String HOOK_OWNER_NAME = Type.getInternalName(Hook.class);
    public static final String HOOK_METHOD_NAME = Hook.class.getDeclaredMethods()[0].getName();
    public static final String HOOK_METHOD_DESC = Type.getMethodDescriptor(Hook.class.getDeclaredMethods()[0]);

    public static void onAllocation()
    {
        Monitor.onAllocation();
    }
}
```



```java
public static void onAllocation()
{
    if (checkCurrentThreadState())
    {
        if (thresholdExceeded())
        {
            activateJstack();
            resetCounters();
        }
        else
        {
            incCounter();
        }
    }
}
```



com/sparktale/bugtale/meta/amagent/Monitor

```
{
mv = cw.visitMethod(ACC_PUBLIC + ACC_STATIC, "onAllocation", "()V", null, null);
mv.visitCode();
Label l0 = new Label();
mv.visitLabel(l0);
mv.visitLineNumber(11, l0);
mv.visitMethodInsn(INVOKESTATIC, "com/sparktale/bugtale/meta/amagent/Monitor", "checkCurrentThreadState",
Label l1 = new Label();
mv.visitJumpInsn(IFEQ, l1);
Label l2 = new Label();
mv.visitLabel(l2);
mv.visitLineNumber(13, l2);
mv.visitMethodInsn(INVOKESTATIC, "com/sparktale/bugtale/meta/amagent/Monitor", "thresholdExceeded", "()Z");
Label l3 = new Label();
mv.visitJumpInsn(IFEQ, l3);
Label l4 = new Label();
mv.visitLabel(l4);
mv.visitLineNumber(15, l4);
mv.visitMethodInsn(INVOKESTATIC, "com/sparktale/bugtale/meta/amagent/Monitor", "activateJstack", "()V");
Label l5 = new Label();
mv.visitLabel(l5);
mv.visitLineNumber(16, l5);
mv.visitMethodInsn(INVOKESTATIC, "com/sparktale/bugtale/meta/amagent/Monitor", "resetCounters", "()V");
Label l6 = new Label();
mv.visitLabel(l6);
mv.visitLineNumber(17, l6);
mv.visitJumpInsn(GOTO, l1);
mv.visitLabel(l3);
mv.visitLineNumber(20, l3);
mv.visitFrame(Opcodes.F_SAME, 0, null, 0, null);
mv.visitMethodInsn(INVOKESTATIC, "com/sparktale/bugtale/meta/amagent/Monitor", "incCounter", "()V");
mv.visitLabel(l1);
mv.visitLineNumber(23, l1);
mv.visitFrame(Opcodes.F_SAME, 0, null, 0, null);
mv.visitInsn(RETURN);
mv.visitMaxs(1, 0);
mv.visitEnd();
}
```

ASM Bytecode Outline plug-in

Questions?

takipi.com

blog.takipi.com