Instrument to Remove

Using Java agents for fun and profit

Johannes Bechberger


BOSCH
L-BOX
BOSCH
BOSCH

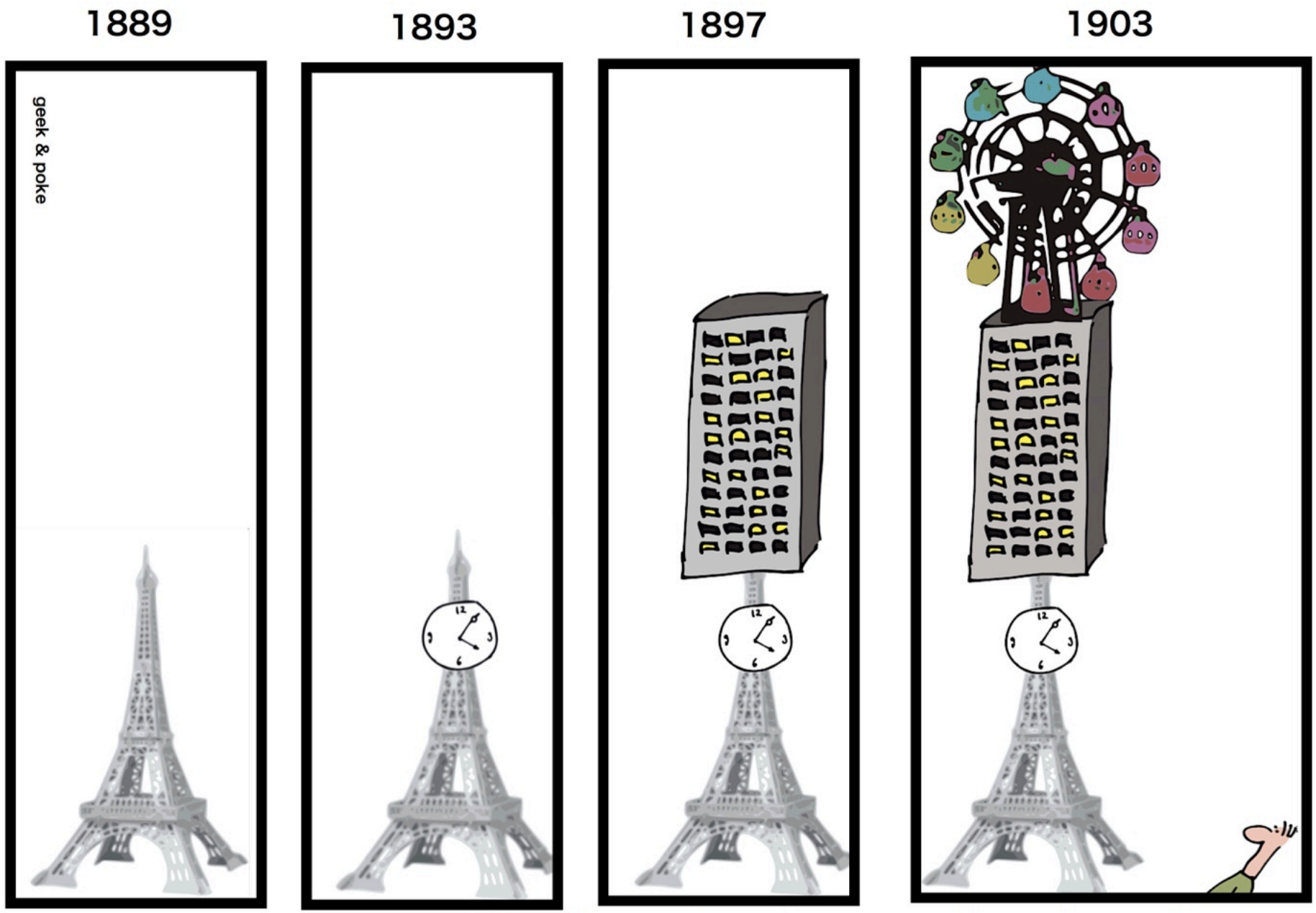

O Rings
SPCings
Key Stock & Swivel ends
Wood & Sheet Rock Screws
Lag Screws
Lag Screws
E
Line & Hose Clamps
1/8" Pipe
1/8" Pipe
1/4" Pipe
3/8" Pipe
1/2" Pipe
1/2" Pipe
3/4" Pipe
E
JIC & Pipe
Lugs & Wheel Studs GMC Lug Caps


maximo
JUMBO
DO IT · DECO · GARDEN
Herzlich willkommen
in Ihrem
JUMBO maximo Arbon.
Montag – Donnerstag
8.30 – 19.00 Uhr
Freitag
8.30 – 20.00 Uhr
Samstag
8.00 – 16.00 Uhr
ZH 874458
SG 172298
www.schluessel-schneider.ch

```
web</artifactId>
</dependency>
<dependency>

<groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-
validation</artifactId>
</dependency>
<dependency>

<groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-
thymeleaf</artifactId>
</dependency>
<dependency>

<groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-
test</artifactId>
  <scope>test</scope>
</dependency>

<!-- Databases - Uses H2 by default -->
<dependency>
  <groupId>com.h2database</groupId>
```

Do you need all?

Probably not



Thank god not everything is software

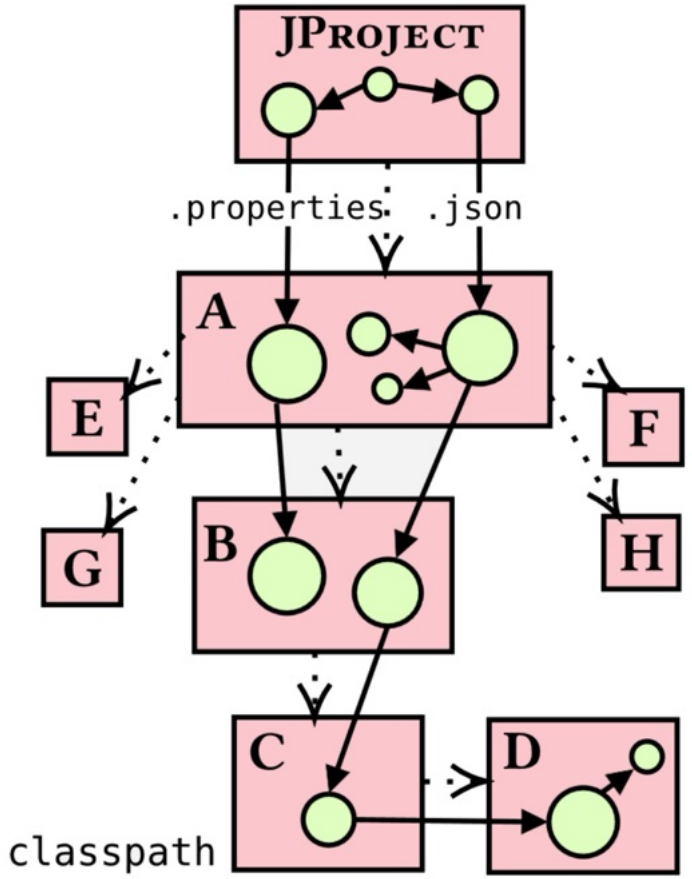
“

Software systems have a natural tendency to grow in size and complexity over time.

A part of this growth comes with new features or bug fixes, while another part is due to useless code that accumulates over time.

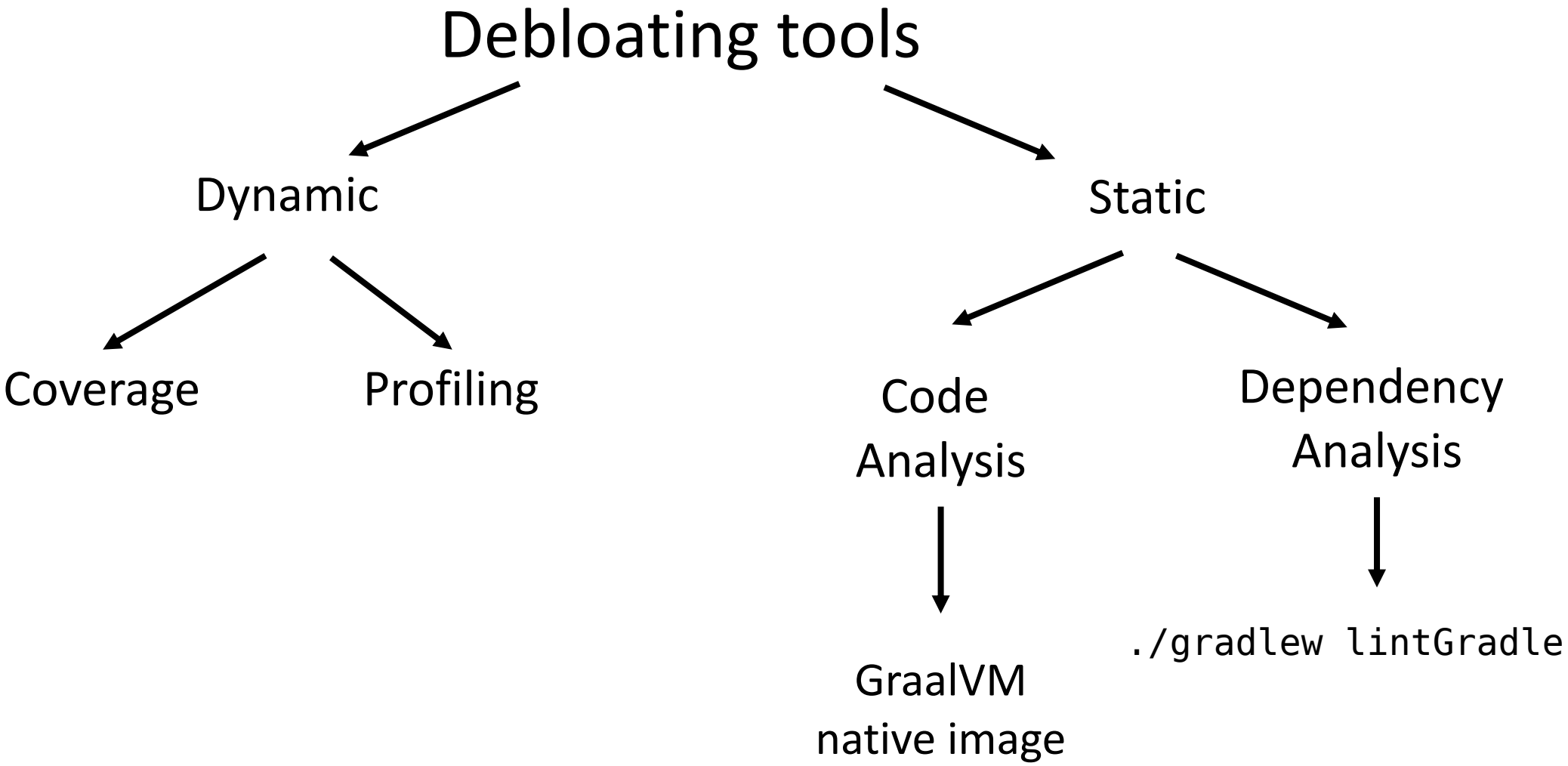
The problem of safely debloating real-world applications remains a long-standing software engineering endeavor today.

— Soto-Valero et. al



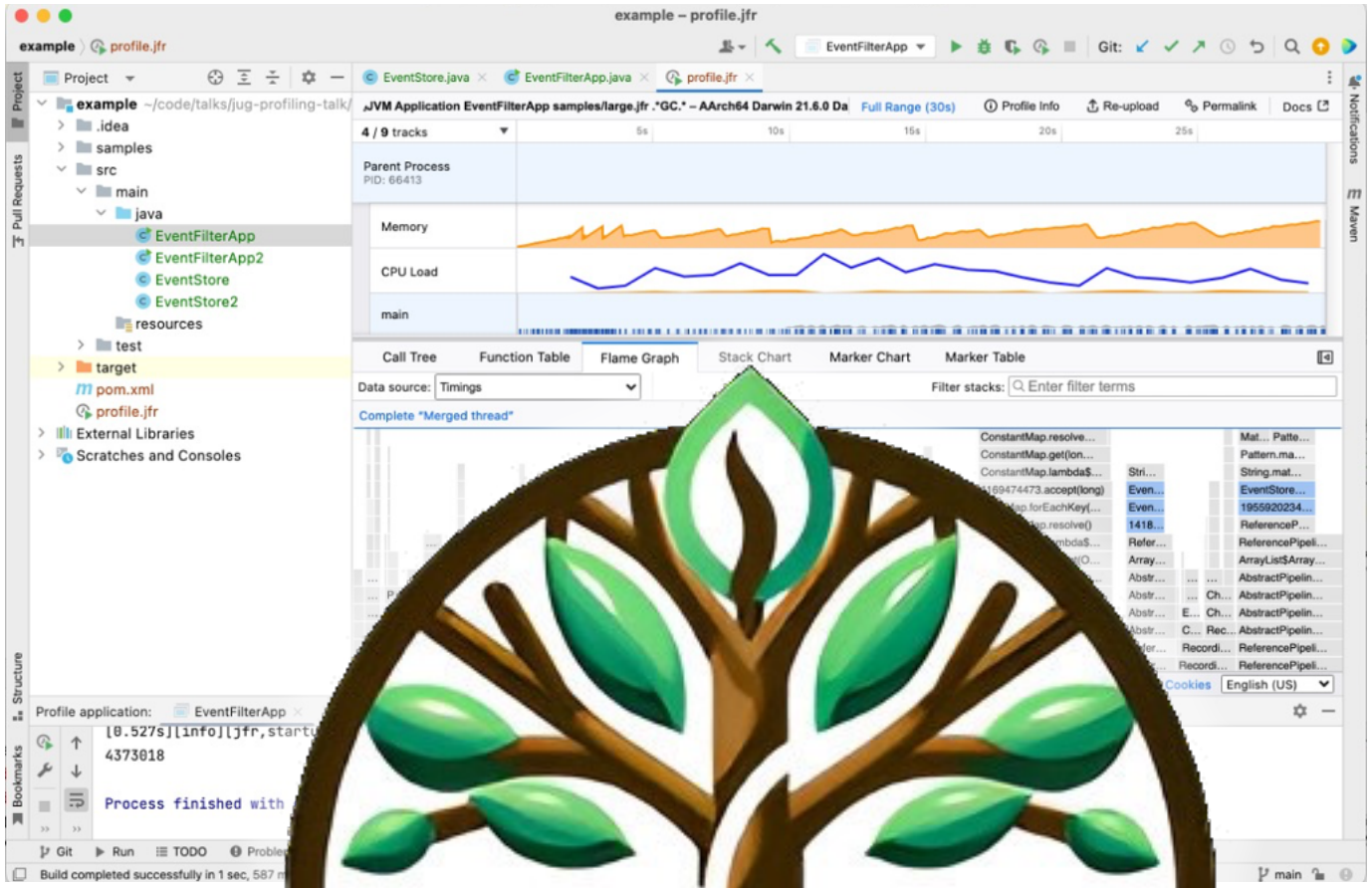
Source: https://arxiv.org/pdf/2008.08401.pdf

How do we debloat?


Debloating tools
Dynamic
Static
Coverage
Profiling
Code Analysis
Dependency Analysis
GraalVM native image
./gradlew lintGradle


jar profiler


Sap
Machine



async-profiler

This project is a low overhead sampling profiler for Java that does not suffe features HotSpot-specific APIs to collect stack traces and to track memory with OpenJDK, Oracle JDK and other Java runtimes based on the HotSpot J

async-profiler can trace the following kinds of events:

- CPU cycles
- Hardware and Software performance counters like cache misses, branc switches etc.
- Allocations in Java Heap
- Contented lock attempts, including both Java object

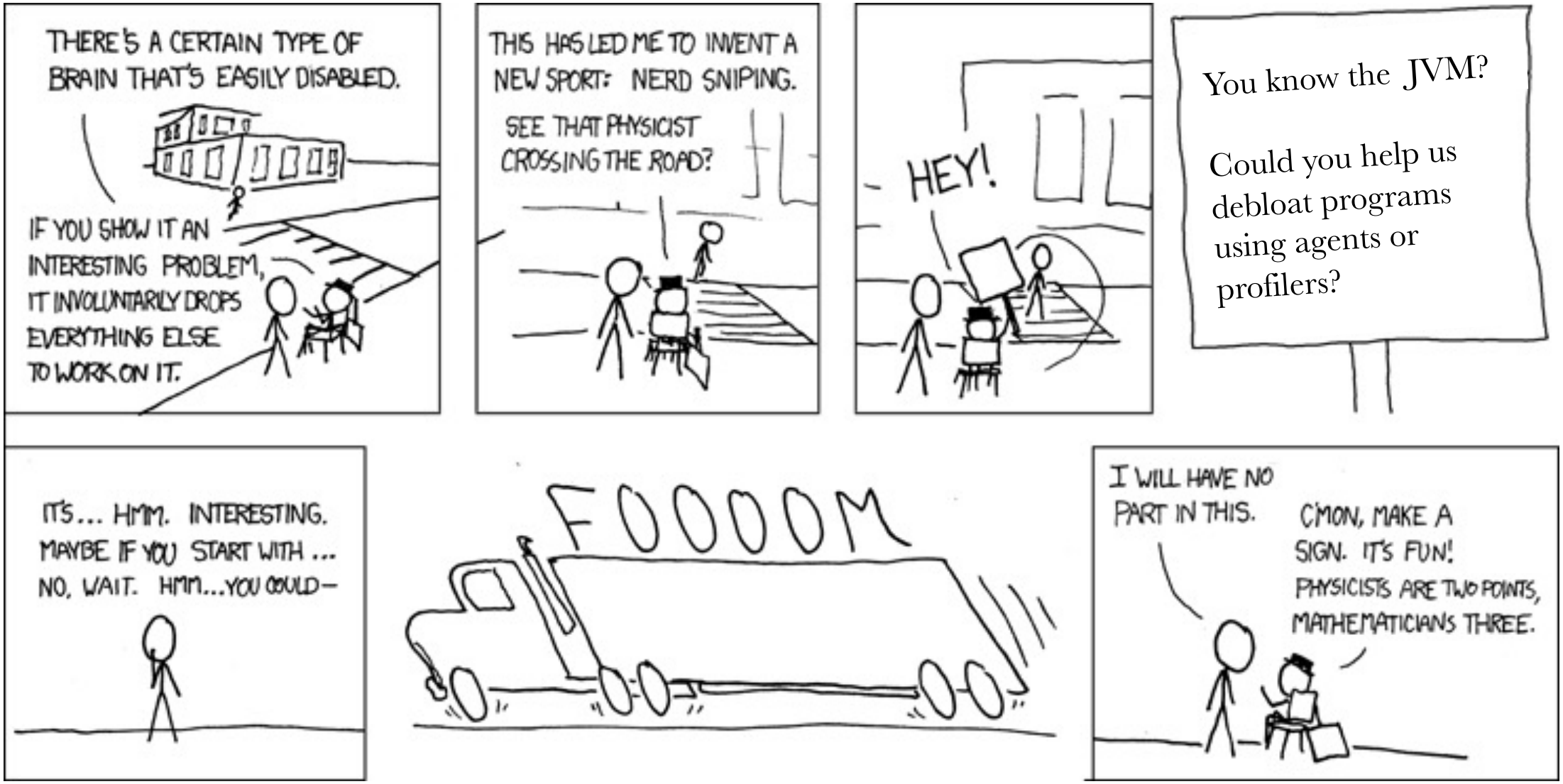
hello eBPF

RESEARCH-ARTICLE OPEN ACCESS

Coverage-Based Debloating for Java Bytecode

Authors: César Soto-Valero, Thomas Durieux, Nicolas Harrand, Benoit Baudry Authors Info & Claims

ACM Transactions on Software Engineering and Methodology, Volume 32, Issue 2 • Article No.: 38, pp 1–34 • https://doi.org/10.1145/3546948


THERE'S A CERTAIN TYPE OF BRAIN THAT'S EASILY DISABLED.
IF YOU SHOW IT AN INTERESTING PROBLEM, IT INVOLUNTARILY DROPS EVERYTHING ELSE TO WORK ON IT.
THIS HAS LED ME TO INVENT A NEW SPORT: NERD SNIPING.
SEE THAT PHYSICIST CROSSING THE ROAD?
HEY!
You know the JVM?
Could you help us debloat programs using agents or profilers?
IT'S... HMM. INTERESTING. MAYBE IF YOU START WITH ... NO, WAIT. HMM...YOU COULD—
FOOOOM
I WILL HAVE NO PART IN THIS.
C'MON, MAKE A SIGN. IT'S FUN! PHYSICISTS ARE TWO POINTS, MATHEMATICIANS THREE.

Existing tools have problems...

So let’s write our own*

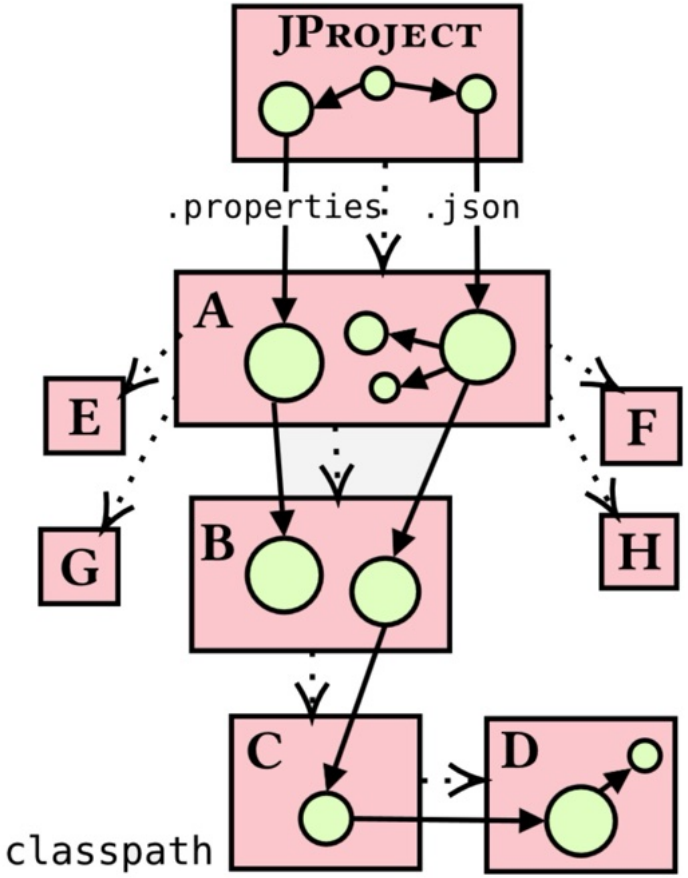
* as a proof of concept for a blog post

Instrumenting Java Code to Find and Handle Unused Classes

Posted on **April 6, 2023**

Reduce the Problem

Find unused classes



Source: https://arxiv.org/pdf/2008.08401.pdf

Find unused classes

```
class A {


    private int field;
    public void method() {...}
}
```

When are static initializers called

“

- T is a class and an instance of A is created.
- A static method declared by A is invoked.
- A static field declared by A is assigned.
- A static field declared by A is used and the field is not a constant variable

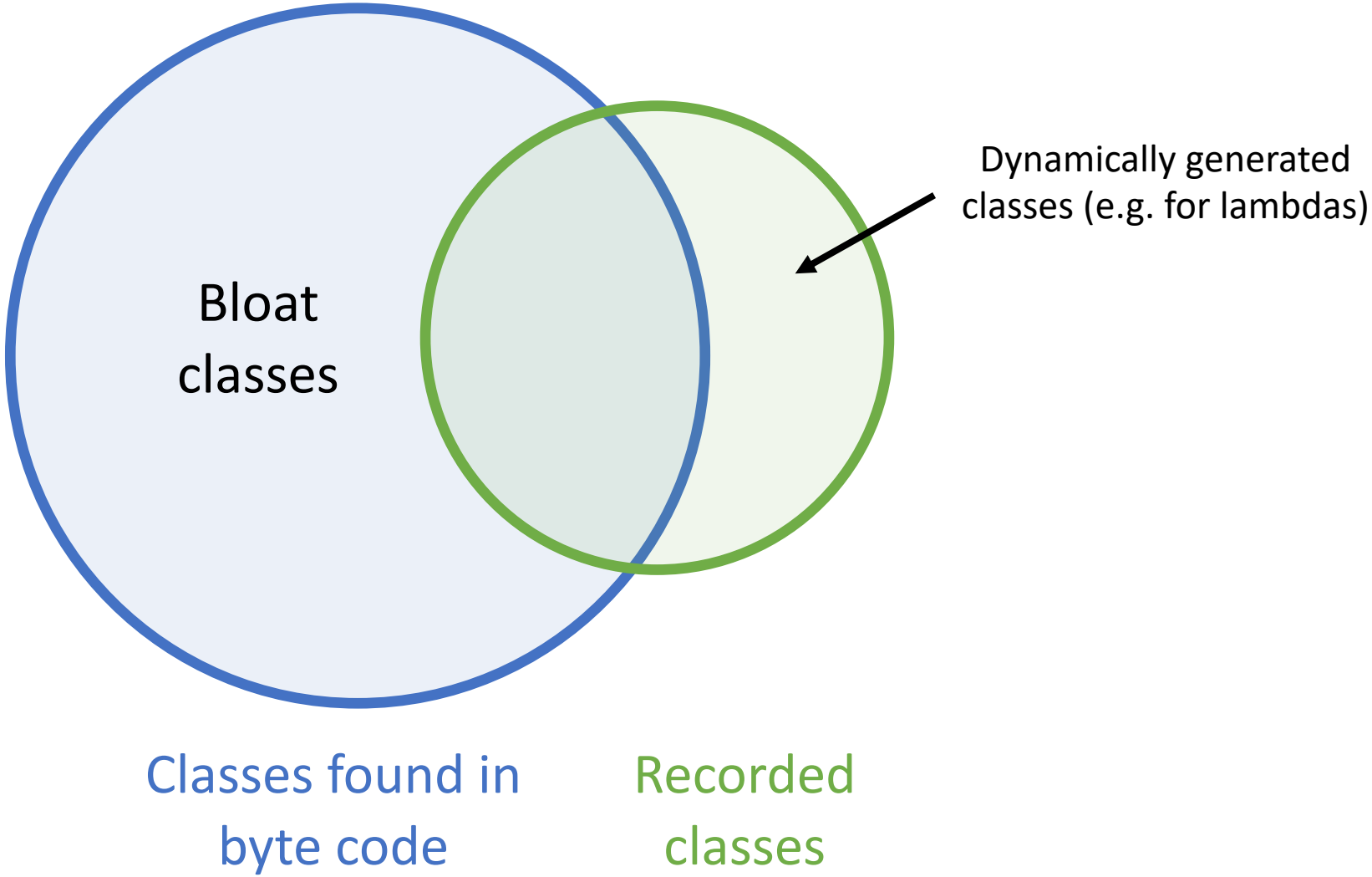
– JLS SE17 §12.4.2. Detailed Initialization Procedure

Static initializers in interfaces

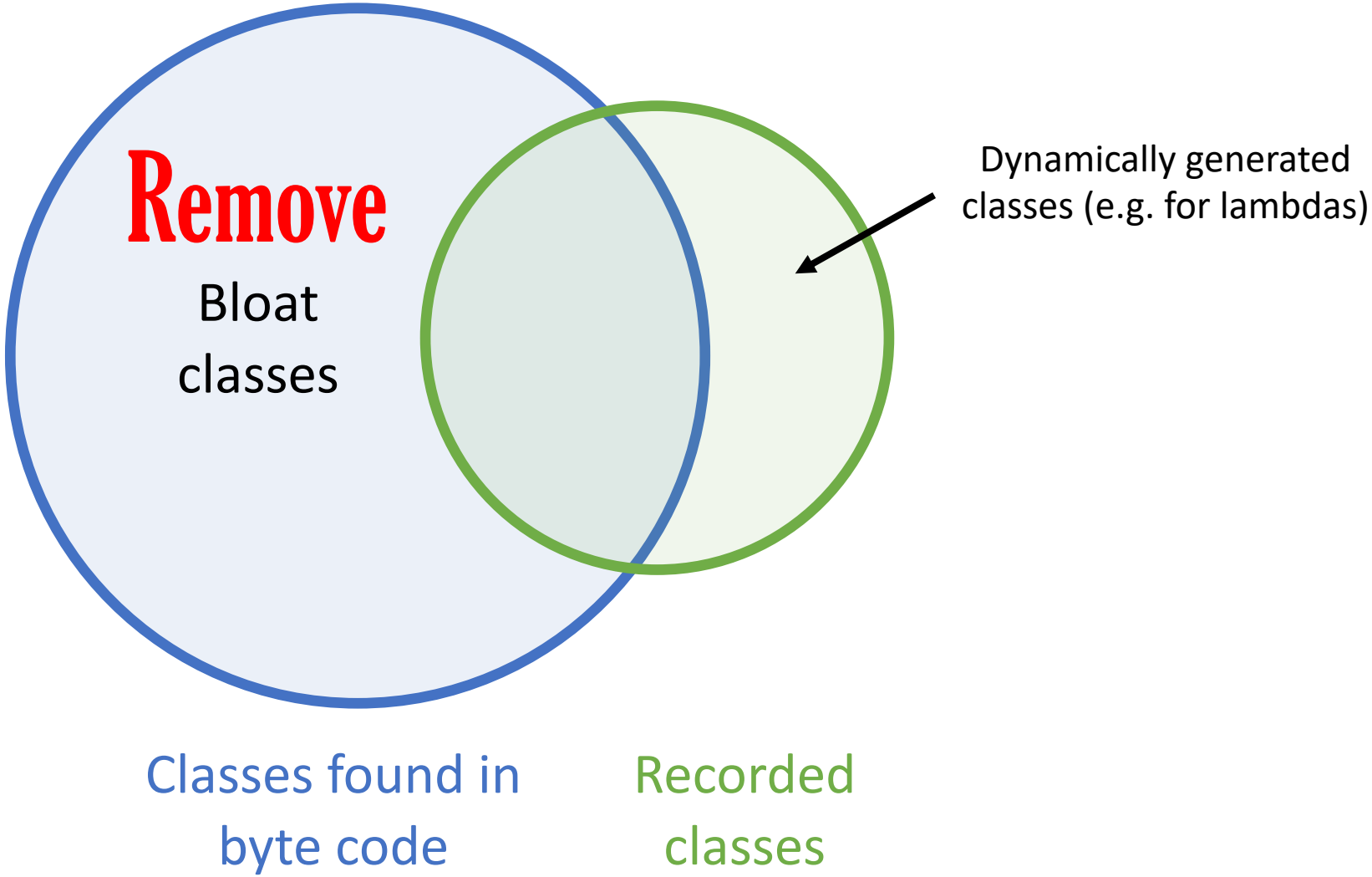
```
interface I {
    static {
        Store.getInstance(I.class).recordUsage("I");
    }
    void method();
}
```

Allowed

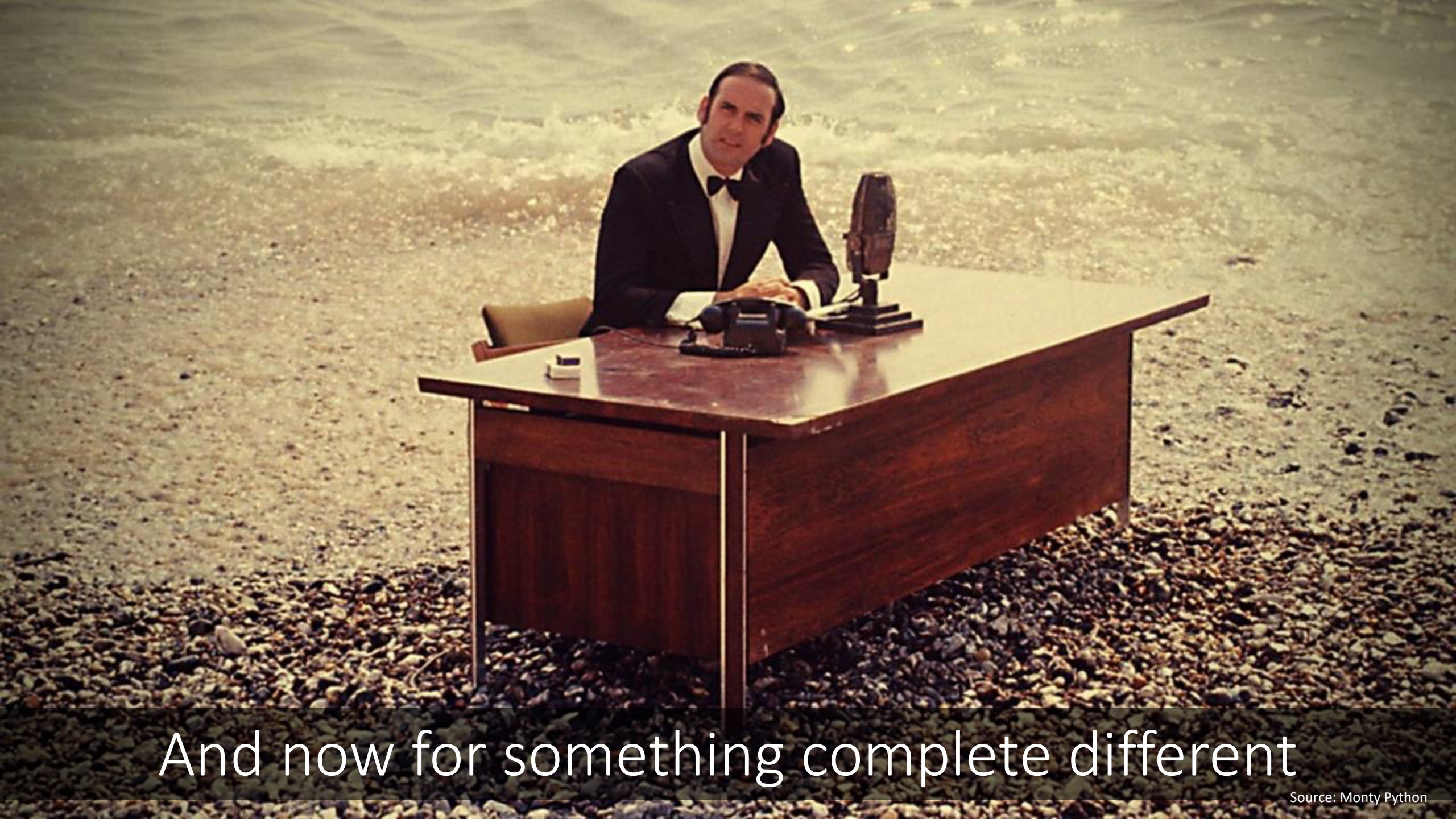
Forbidden Bytecode


Bloat
classes
Dynamically generated
classes (e.g. for lambdas)
Classes found in
byte code
Recorded
classes

What to do with this information?


Remove
Bloat
classes
Dynamically generated
classes (e.g. for lambdas)
Classes found in
byte code
Recorded
classes

```
class UnusedClass {
    static {
        LOG.error("Class UnusedClass is used " +
                  "which is not allowed");
    }
    private int field;
    public void method() {...}
}
```



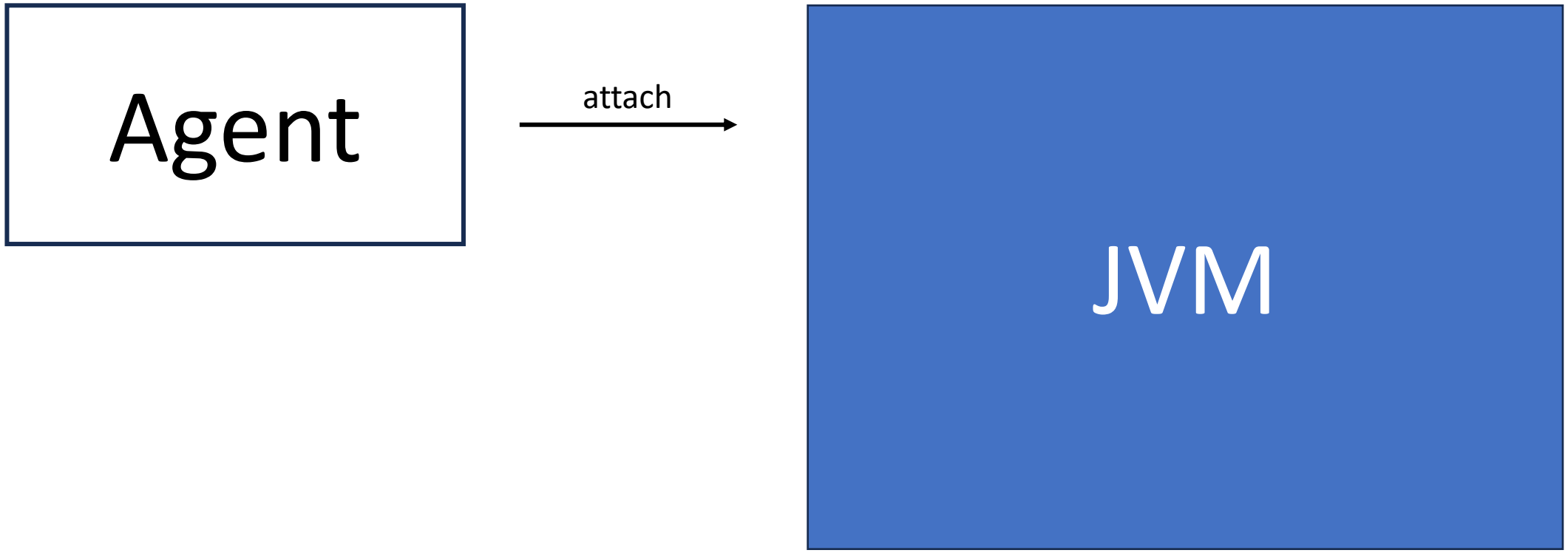
Source: Monty Python

Implementation

In come Java Agents

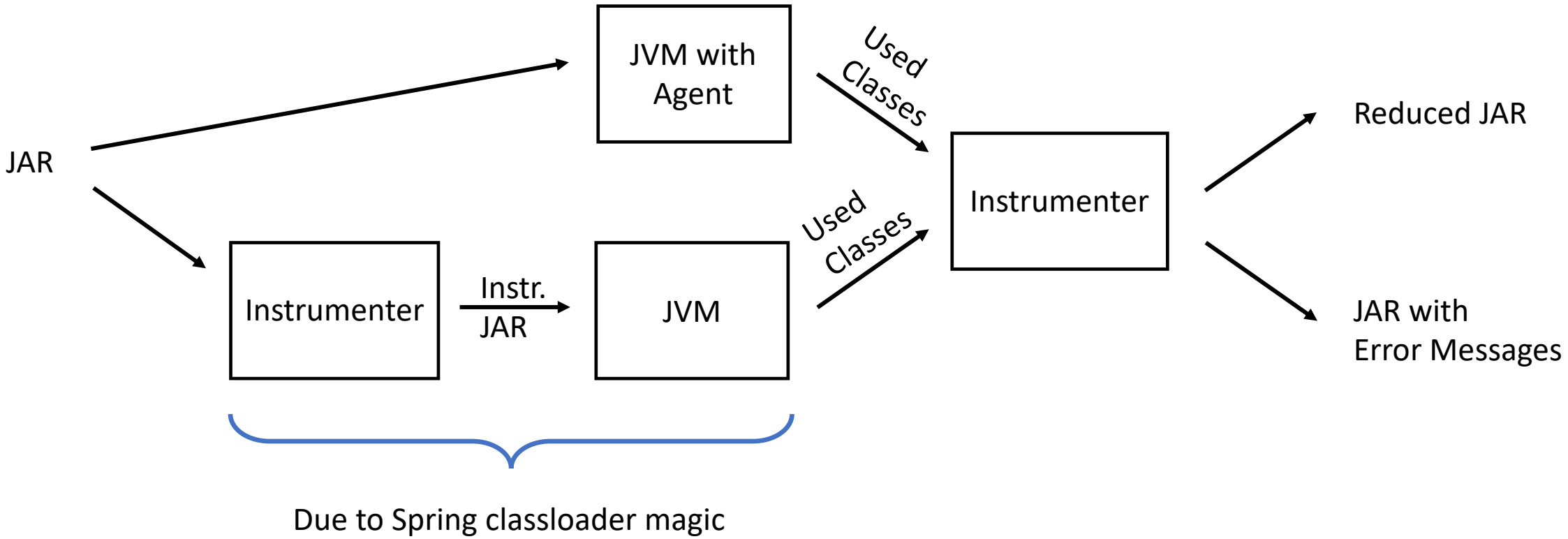
They are essentially like bike computers

https://commons.wikimedia.org/wiki/File:Cyclocomputer.jpg


Agent
attach
JVM

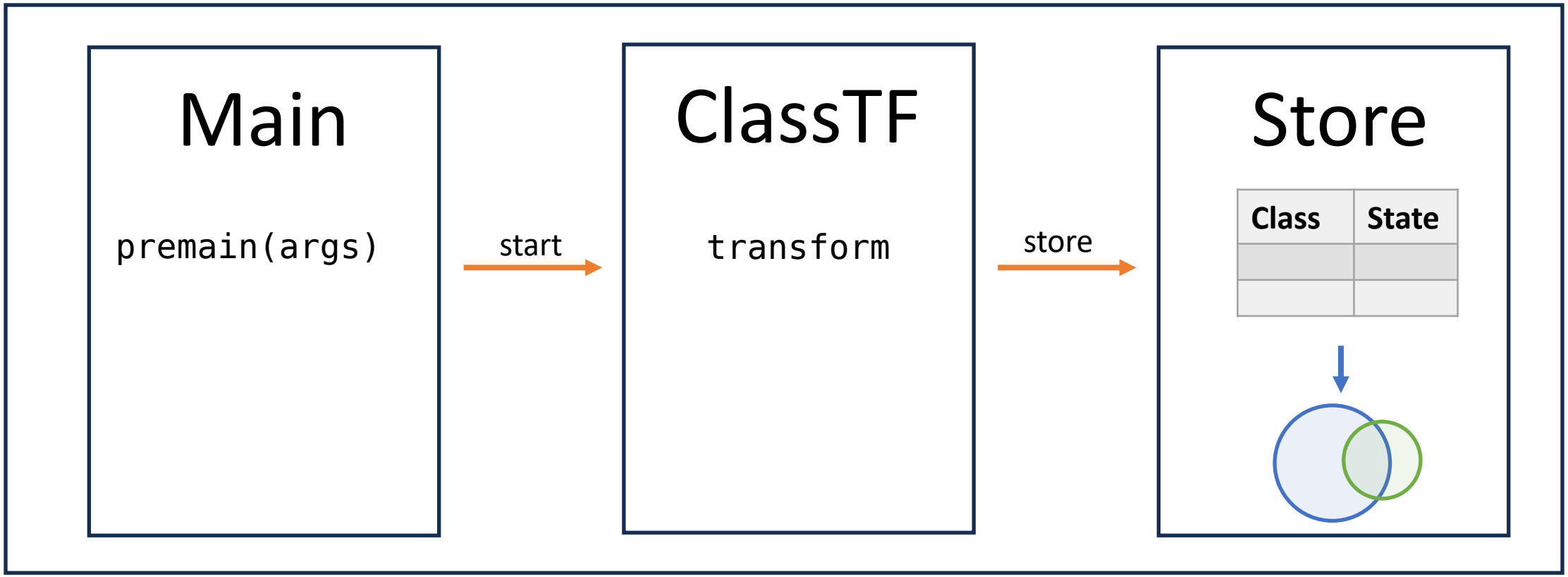

Agent
JVM

Structure


JAR
JVM with Agent
Used Classes
Instrumenter
Reduced JAR
Instrumenter
Instr. JAR
JVM
Used Classes
JAR with Error Messages
Due to Spring classloader magic

https://github.com/parttimenerd/dead-code-agent

Agent Structure

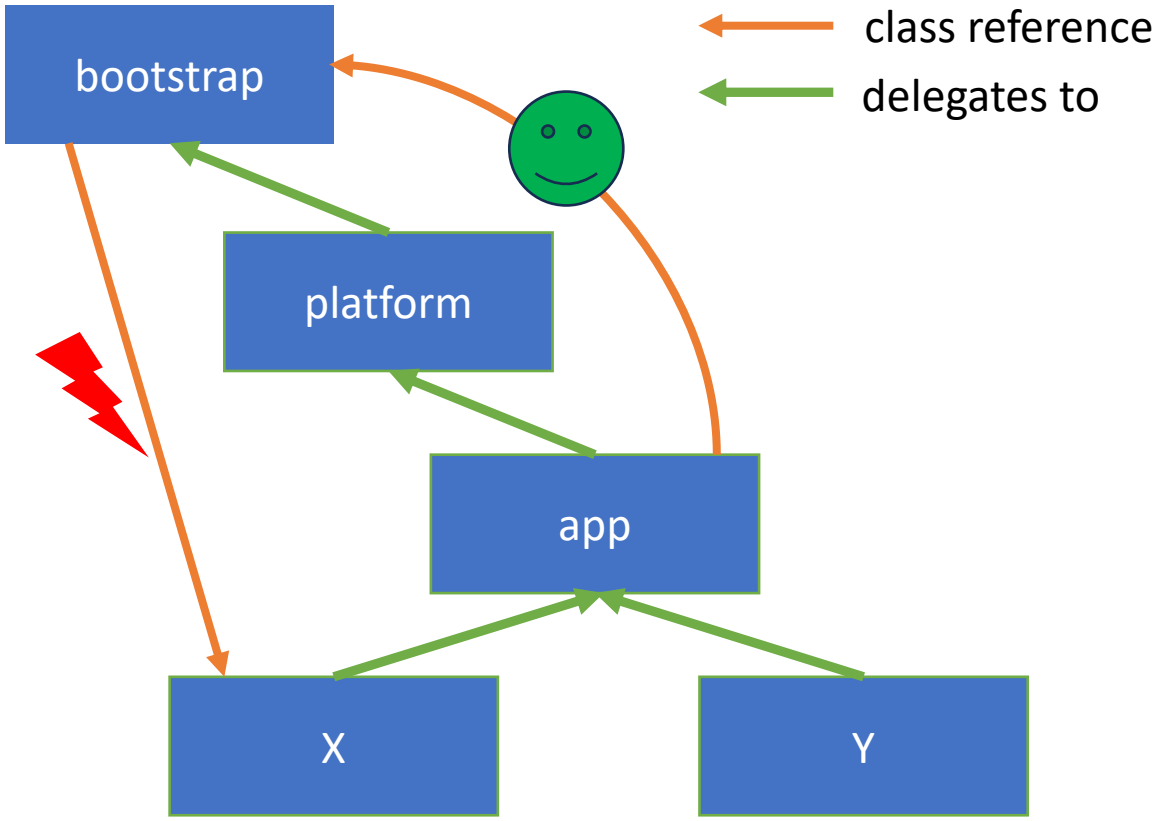


Only one problem...

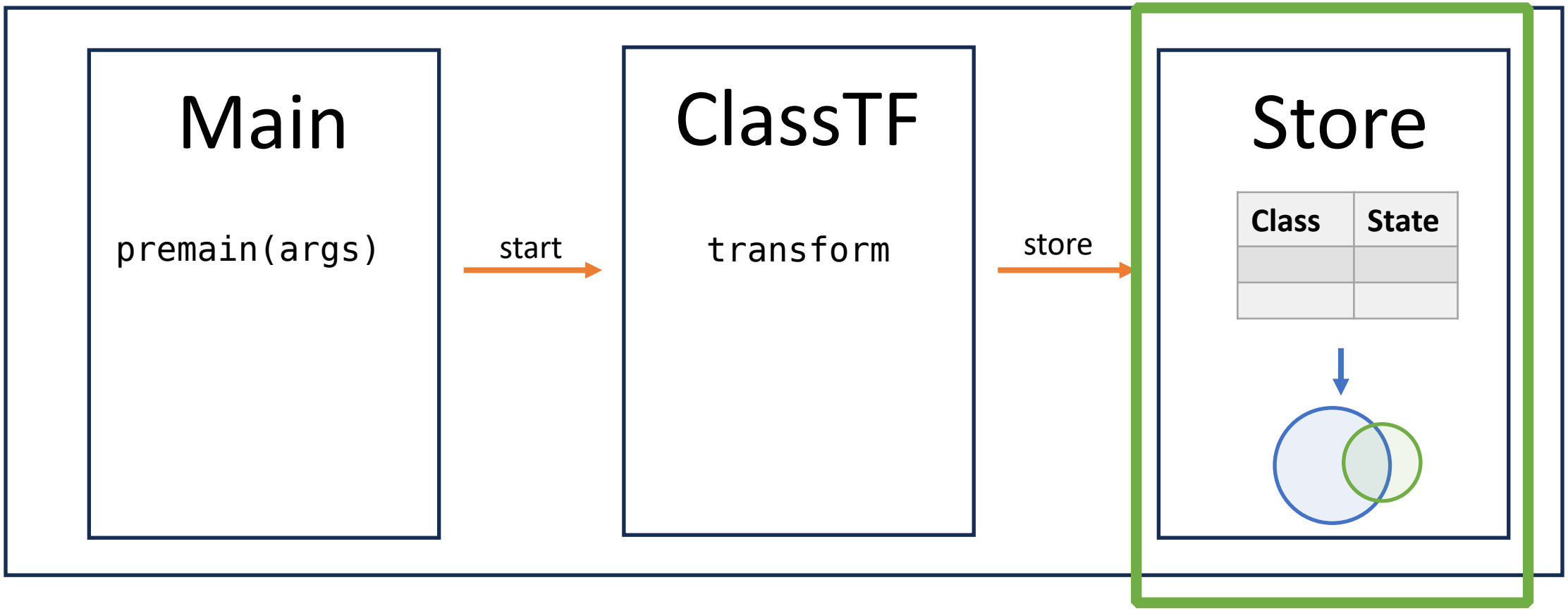
```
package my.app;

class A {
    static {
        Store.getInstance().processClassUsage("my.app.A");
    }
    private int field;
    public void method() {...}
}
```

```
package java.lang;

public final class String implements ... {
    static {
        Store.getInstance().processClassUsage("java.lang.String");
    }
    @Stable
    private final byte[] value;
    ...
    public String() {...}
    ...
}
```

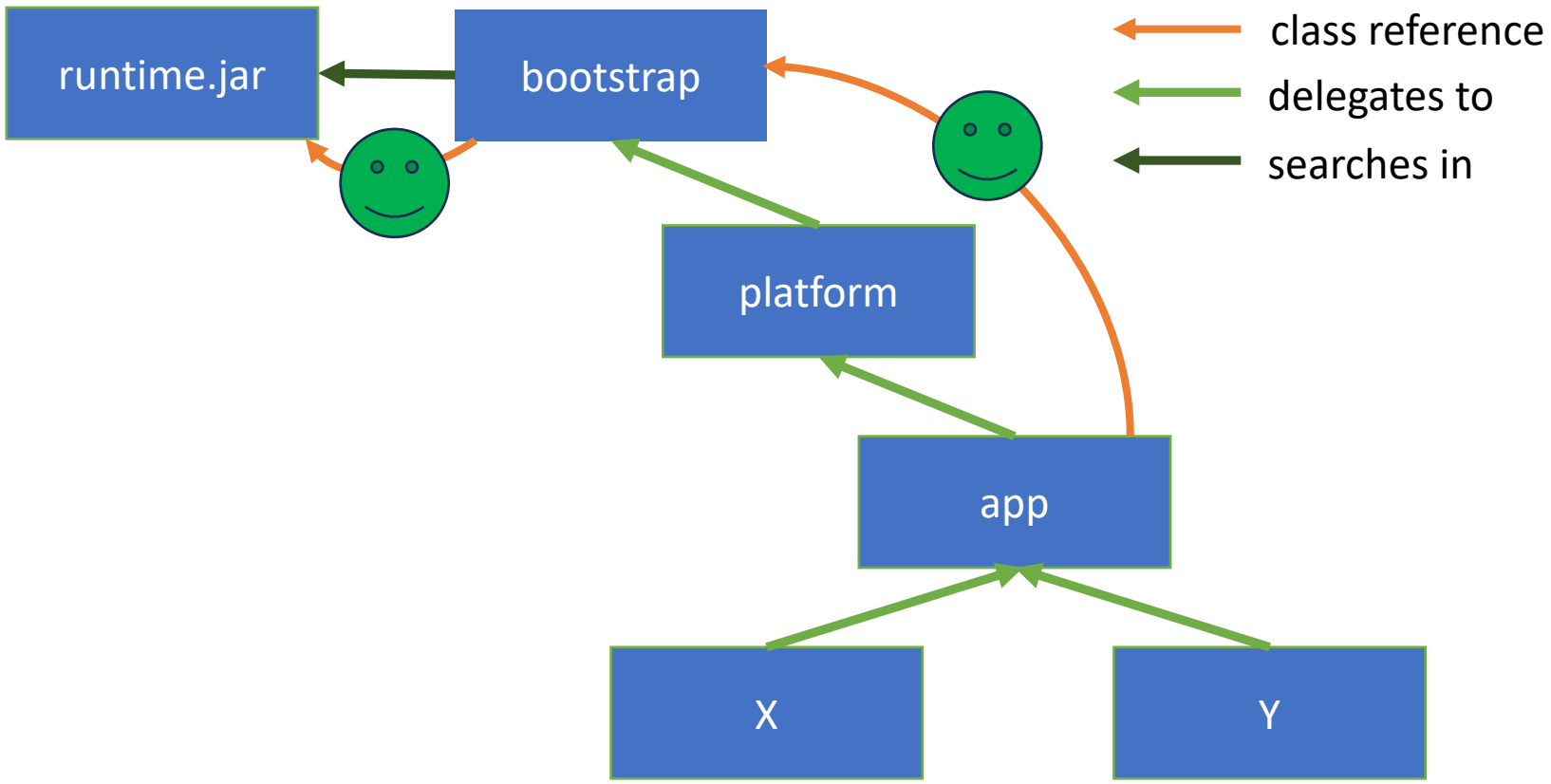
Class Loader Hierarchies



Agent Structure



Class Loader Hierarchies



https://mostlynerdless.de/blog/2023/06/02/class-loader-hierarchies/

Implemented via

```
instrumentation Instrumentation {
        void appendToBootstrapClassLoaderSearch(JarFile jarfile);
}
```

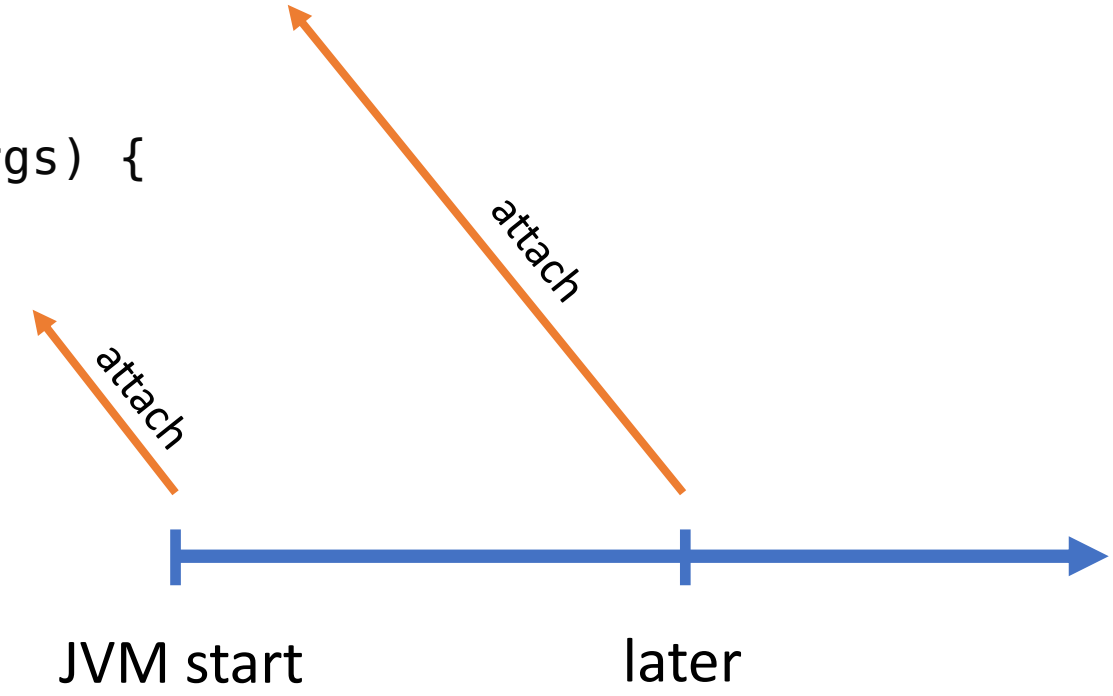
```
inst.appendToBootstrapClassLoaderSearch(new JarFile(
        getExtractedJARPath().toFile()));
```

Normally

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

public static void main(String args[]) {
    ...
}

void main() {
    ...
}
```

Main Class

```
public static void agentmain(String agentArgs) {
    ...
}

public static void premain(String agentArgs) {
    ...
}
```



Main Class

```
public static void agentmain(String agentArgs, Instrumentation inst) {
    ...
}

public static void premain(String agentArgs, Instrumentation inst) {
    ...
}
```

Main Class

```
public class Main {

    public static void premain(String agentArgs, Instrumentation inst) {
        AgentOptions options = new AgentOptions(agentArgs);

        ...
        inst.appendToBootstrapClassLoaderSearch(
                    new JarFile(getExtractedJARPath().toFile()));

        ...
        inst.addTransformer(new ClassTransformer(options), true);
    }
...
}
```

Instrumentation.retransformClasses

ClassTransformer `extends ClassFileTransformer`

> An agent registers an implementation of this interface using the `addTransformer` method so that the transformer's `transform` method is invoked when classes are loaded, redefined, or retransformed
>
> – *ClassFileTransformer Documentation*

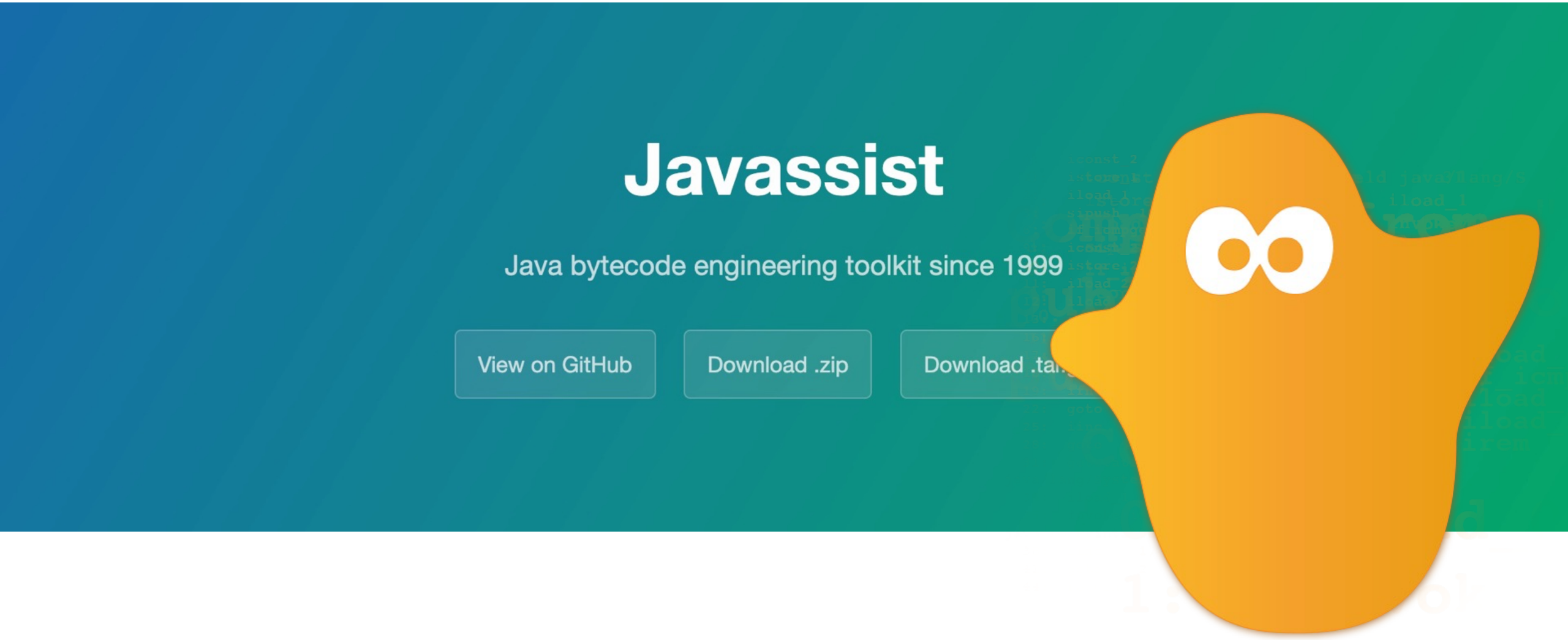
ClassTransformer#transform

```
byte[]
transform(  ClassLoader                 loader,
            String                      className,
            Class<?>                    klass,
            ProtectionDomain            domain,
            byte[]                      classfileBuffer)
    throws IllegalClassFormatException {
    ...
}
```

ClassTransformer#transform

```
if (className.startsWith("me/bechberger/runtime/Store") ||
    className.startsWith("me/bechberger/ClassTransformer") ||
    className.startsWith("java/") ||
    className.startsWith("jdk/internal") ||
    className.startsWith("sun/")) {
    return classfileBuffer;
}
```

How to actually transform the bytecode?



https://www.javassist.org/

ClassTransformer#transform

?

```
private void transform(String className, CtClass cc) {
    String cn = formatClassName(className);
    Store.getInstance()
       .processClassLoad(cn,cc.getClassFile().getInterfaces());

    cc.makeClassInitializer()
       .insertBefore(
             String.format("me.bechberger.runtime.Store.getInstance()" +
                           ".processClassUsage(\"%s\");", cn));
}
```

With Spring-PetClinic

```
java -javaagent:./target/dead-code.jar=output=classes.txt \
    -jar petclinic.jar
```

```
u ch.qos.logback.classic.encoder.PatternLayoutEncoder
l ch.qos.logback.classic.joran.JoranConfigurator
u ch.qos.logback.classic.jul.JULHelper
u ch.qos.logback.classic.jul.LevelChangePropagator
...
```

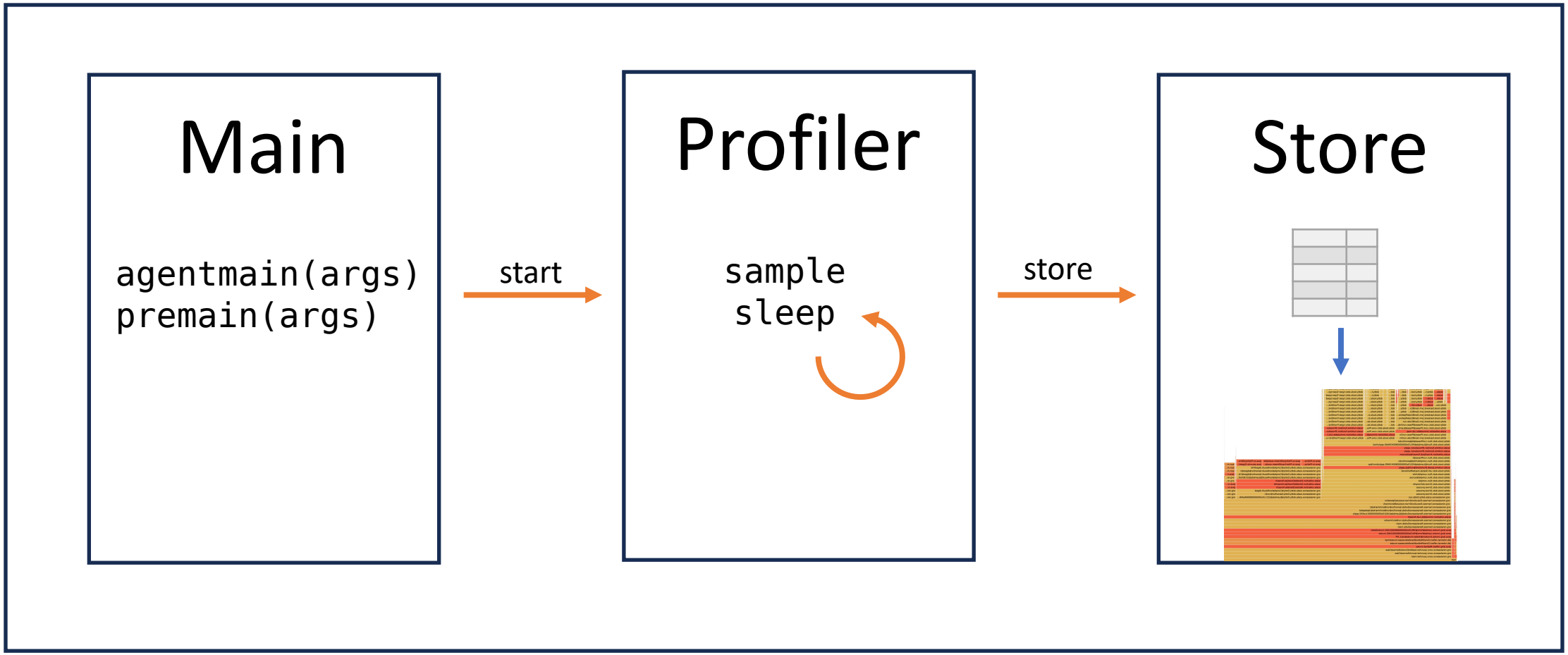
```
Class org.apache.tomcat.util.net.SocketBufferHandler is used which is not allowed
Class org.apache.tomcat.util.net.SocketBufferHandler$1 is used which is not allowed
Class org.apache.tomcat.util.net.NioChannel is used which is not allowed
Class org.apache.tomcat.util.net.NioChannel$1 is used which is not allowed
...
```

Who has wrote an agent before?

Who has used an agent before?

Other applications of agents

Write your own profiler



Who instruments the instrumenter?

```java
@ExtendWith(MockitoExtension.class)
public class MockitoTest {

  @Mock
  List<String> mockedList;

  @Test
  public void whenNotUseMockAnnotation_thenCorrect()
   throws InterruptedException {
    mockedList.add("one");
    Mockito.verify(mockedList).add("one");
    assertEquals(0, mockedList.size());

    Mockito.when(mockedList.size()).thenReturn(100);
    assertEquals(100, mockedList.size());

    Thread.sleep(10000000L);
  }
}
```

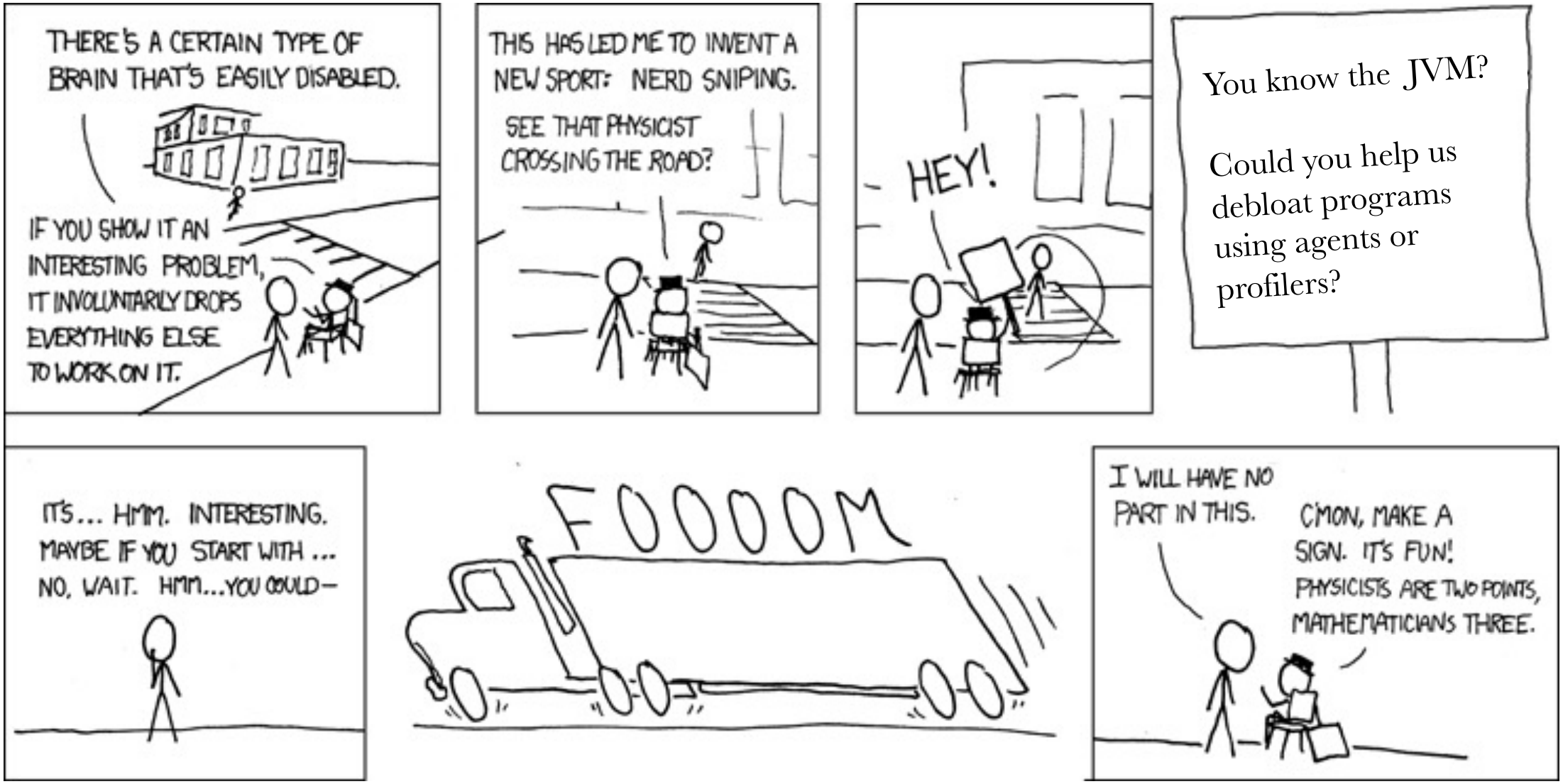
This uses ByteBuddy

org.mockito.internal.creation.bytebuddy.InlineBytecodeGenerator

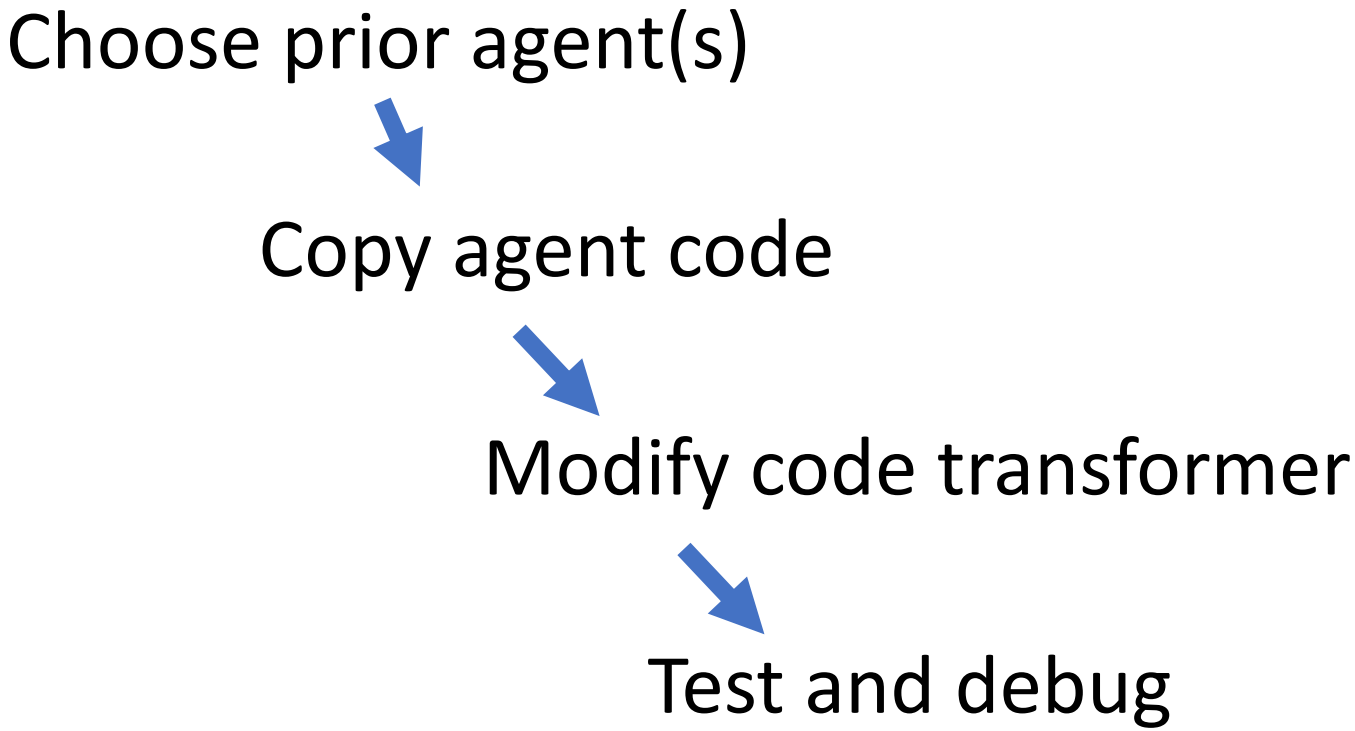
Output format: Line by Line

Files changed (4)

{old → new}/java.lang.Iterable.java	+27 -5
{old → new}/java.util.Collection.java	+68 -13
{old → new}/java.util.List.java	+354 -64
{old → new}/java.util.SequencedCollection.java	+87 -14


THERE'S A CERTAIN TYPE OF BRAIN THAT'S EASILY DISABLED.
IF YOU SHOW IT AN INTERESTING PROBLEM, IT INVOLUNTARILY DROPS EVERYTHING ELSE TO WORK ON IT.
THIS HAS LED ME TO INVENT A NEW SPORT: NERD SNIPING.
SEE THAT PHYSICIST CROSSING THE ROAD?
HEY!
You know the JVM?
Could you help us debloat programs using agents or profilers?
IT'S... HMM. INTERESTING. MAYBE IF YOU START WITH ... NO, WAIT. HMM...YOU COULD—
FOOOOM
I WILL HAVE NO PART IN THIS.
C'MON, MAKE A SIGN. IT'S FUN! PHYSICISTS ARE TWO POINTS, MATHEMATICIANS THREE.

Agent design process



```
public interface Collection<E> extends Iterable<E> {
  // ...
  default Stream<E> stream() {



    return StreamSupport.stream(this.spliterator(), false);









  }
  // ...
}
```

```
public interface Collection<E> extends Iterable<E> {
  // ...
  default Stream<E> stream() {
    MockMethodDispatcher var1 =
            MockMethodDispatcher.get("APErU6j7", this);
    Callable var4 = var1 ≠ null
            && var1.isMocked(this)
            && !var1.isOverridden(this,
              Collection.class.getMethod("stream"))
            ? var1.handle(this,
              Collection.class.getMethod("stream"),
            new Object[0])
            : null;
    Stream var2 = var4 ≠ null ? null :
            StreamSupport.stream(this.spliterator(), false);
    if (var4 ≠ null) {
      var2 = (Stream)var4.call();
    }

    return var2;
  }
  // ...
}
```

Instrumenting agent for instrumenting instrumenting agents

```
var exprEditor =
    new ExprEditor() {
      @Override
      public void edit(MethodCall m) throws CannotCompileException {
        if (!isAddTransformerMethod(m)) {
          return;
        }
        m.replace(
              "me.bechberger.meta.runtime.InstrumentationHandler" +
              ".addTransformer($0, $1);");
      }
    };
```

```
private void transform(String className, CtClass cc)
    throws CannotCompileException {
  var exprEditor = 1/* ..  */;
  for (CtConstructor constructor : cc.getDeclaredConstructors()) {
    constructor.instrument(exprEditor);
  }
  for (CtMethod method : cc.getDeclaredMethods()) {
    method.instrument(exprEditor);
  }
}
```

Decompiled bytecode using vineflower, obtained when ever this page is loaded.

org.mockito.internal.creation.bytebuddy.InlineBytecodeGenerator

Output format: Side by Side

https://github.com/parttimenerd/meta-agent

Files changed (4)

- {old → new}/java.lang.Iterable.java +27 -
- {old → new}/java.util.Collection.java +68 -1
- {old → new}/java.util.List.java +354 -6
- {old → new}/java.util.SequencedCollection.java +87 -1

{old → new}/java.lang.Iterable.java RENAMED

```
@@ -1,23 +1,45 @@
 1   package java.lang;                                          1   package java.lang;
 2                                                              2
 3   import java.util.Iterator;                                 3   import java.util.Iterator;
 4   import java.util.Objects;                                  4   import java.util.Objects;
 5   import java.util.Spliterator;                              5   import java.util.Spliterator;
 6   import java.util.Spliterators;                             6   import java.util.Spliterators;
                                                            +   7   import java.util.concurrent.Callable;
 7   import java.util.function.Consumer;                        8   import java.util.function.Consumer;
                                                            +   9   import org.mockito.internal.creation.bytebuddy.inject.MockMethodDispatcher;
 8                                                             10
 9   public interface Iterable<T> {                            11  public interface Iterable<T> {
10      Iterator<T> iterator();                                12     Iterator<T> iterator();
11                                                             13
12      default Spliterator<T> spliterator() {                 14     default Spliterator<T> spliterator() {
                                                            +  15        MockMethodDispatcher var1 = MockMethodDispatcher.get("APErU6j7", this
                                                            +  16        Callable var4 = var1 != null && var1.isMocked(this) && !var1.isOverri
                                                            +  17           ? var1.handle(this, Iterable.class.getMethod("spliterator"), new O
                                                            +  18           : null;
13  -       return Spliterators.spliteratorUnknownSize(this.iterator(), 0);  19  +  Spliterator var2 = var4 != null ? null : Spliterators.spliteratorUnkn
                                                            +  20        if (var4 != null) {
                                                            +  21           var2 = (Spliterator)var4.call();
```


@parttimen3rd on Twitter
parttimenerd on GitHub
mostlynerdless.de
@SweetSapMachine
sapmachine.io