The definitive guide to Java agents

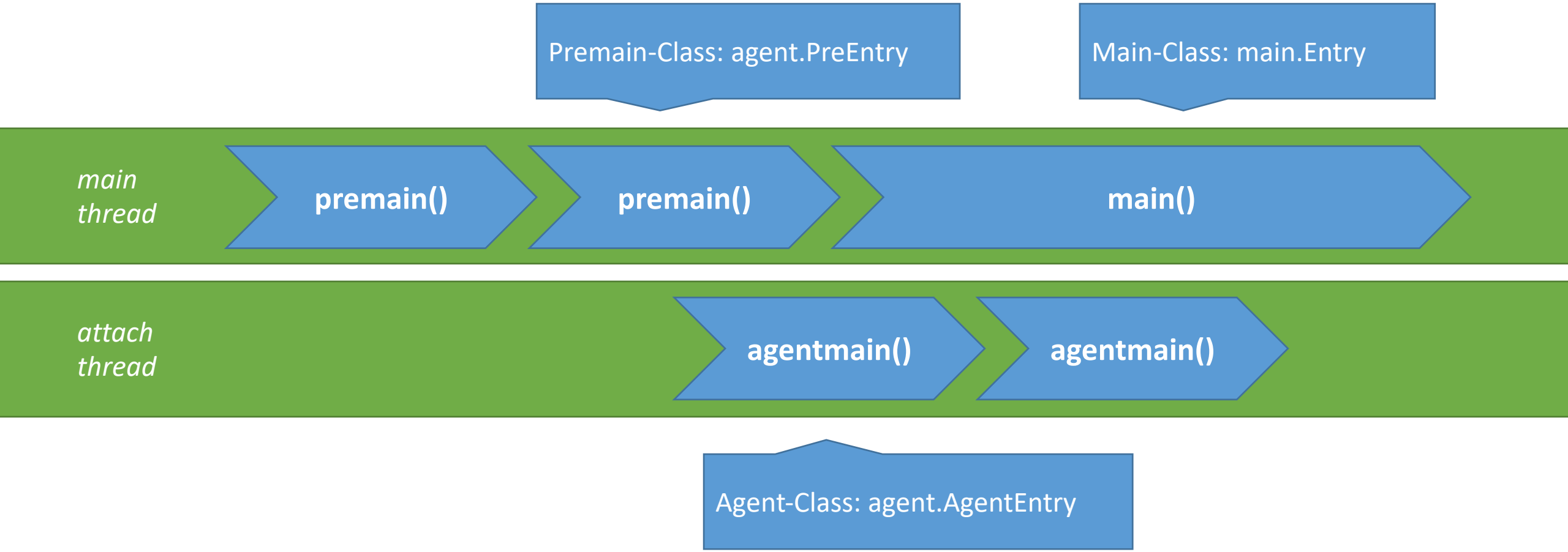
agent fundamentals

code instrumentation

class loading

miscellaneous

What are Java agents?



Running a static Java agent

```
java -cp:. -javaagent:/home/user/myagent.jar=MyAgent main.AppEntry
```

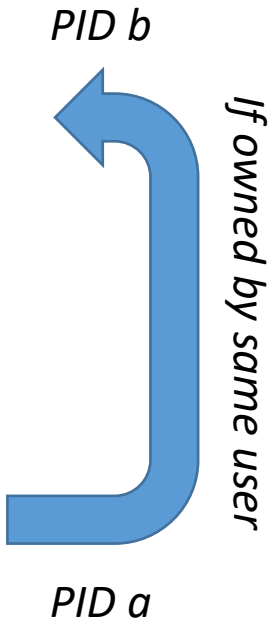
```
class PreEntry {
  public static void premain(String arg) {
    System.out.println("Hello from " + arg);
  }
}
```

Running a dynamic Java agent

```
java -cp:. main.AppEntry
```

```
class AgentEntry {
  public static void agentmain(String arg) {
    System.out.println("Hello from " + arg);
  }
}
```

```
VirtualMachine vm = VirtualMachine.attach(pid);
try {
  vm.loadAgent("/home/user/myagent.jar", "MyAgent");
} finally {
  vm.detach();
}
```



Transforming a class

```
java -cp:. -javaagent:/home/user/myagent.jar=MyAgent main.AppEntry
```

```
class PreEntry {
  public static void premain(String arg, Instrumentation inst) {
    inst.addTransformer(
    (classLoader, className, classIfLoaded, protectionDomain, classFile) -> {
      if (shouldTransform(className)) {
        return transform(classFile);
      } else {
        return null;
      }
    });
  }
}
```

Retransforming a class

```
java -cp:. main.AppEntry


class AgentEntry {
  public static void agentmain(String arg, Instrumentation inst) {
    inst.addTransformer(
    (classLoader, className, classIfLoaded, protectionDomain, classFile) -> {
      if (shouldTransform(classIfLoaded)) {
        return transform(classFile);
      } else {
        return null;
      }
    }, true);
    inst.retransformClasses(getClassesToRetransform());
  }
}
```

Retransforming a class: limitations

```
class Sample {
  String m() {
    return m2();
  }
} private String m2() {
    return "bar";
  }
}
```

Possible future extension: JEP 159 or Dynamic code evolution VM

But currently this does not seem to be a goal.

Running a native agent

```
java -agentlib:/home/user/myagent.so=MyAgent main.AppEntry
```

```
JNIEXPORT jint JNICALL Agent_OnLoad(
        JavaVM *jvm, char *options, void *reserved) {
  jvmtiEnv *jvmti; jint res; jvmtiError error;
  jvmtiCapabilities capabilities; jvmtiEventCallbacks callbacks;

  res = (*vm)->GetEnv(vm, (void **)&jvmti, JVMTI_VERSION_1);

  (void) memset(&capabilities, 0, sizeof(jvmtiCapabilities));
  capabilities.can_retransform_classes = 1;
  error = jvmti->AddCapabilities(&capabilities);

  callbacks.ClassFileLoadHook = &myClassFileLoadHook;
  error = (*jvmti)->SetEventCallbacks(jvmti, &callbacks, (jint) sizeof(callbacks));

  return JNI_OK;
}
```

How can classes be manipulated at runtime?

```
class Foo {
  String bar() {
    return "bar";
}
```

Foo.java

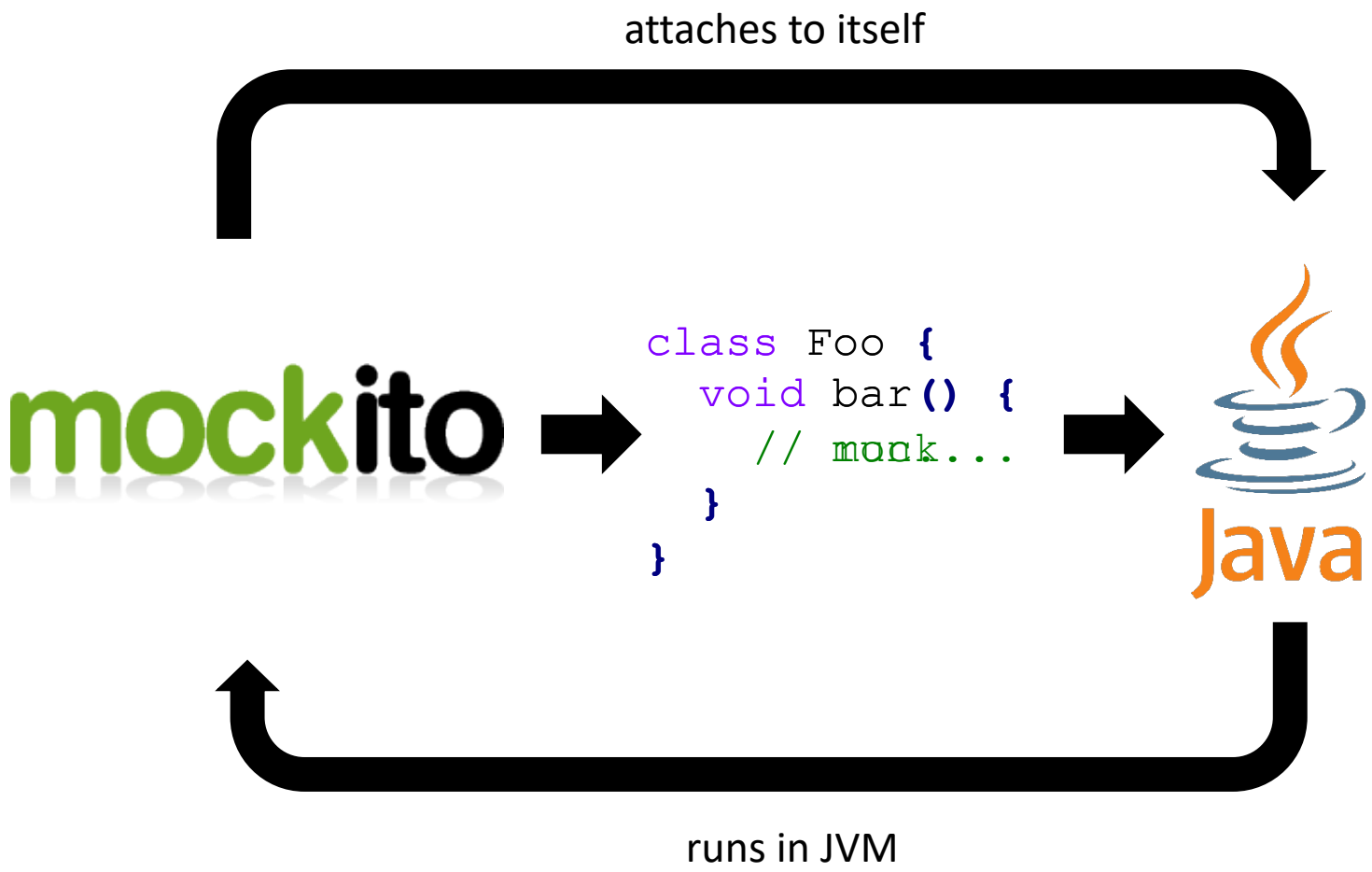
```
class Foo
Constant pool:
  #1 = String #4
  #2 = Class #6
  #3 = Utf8 Code
  #4 = Utf8 bar
  #5 = Utf8 ()Ljava/lang/String;
  #6 = Utf8 Foo
  #7 = Utf8 java/lang/Object
java.lang.String bar();
  descriptor: ()Ljava/lang/String;
  Code:
    0: ldc #2
    2: areturn
```

Foo.class (javap)

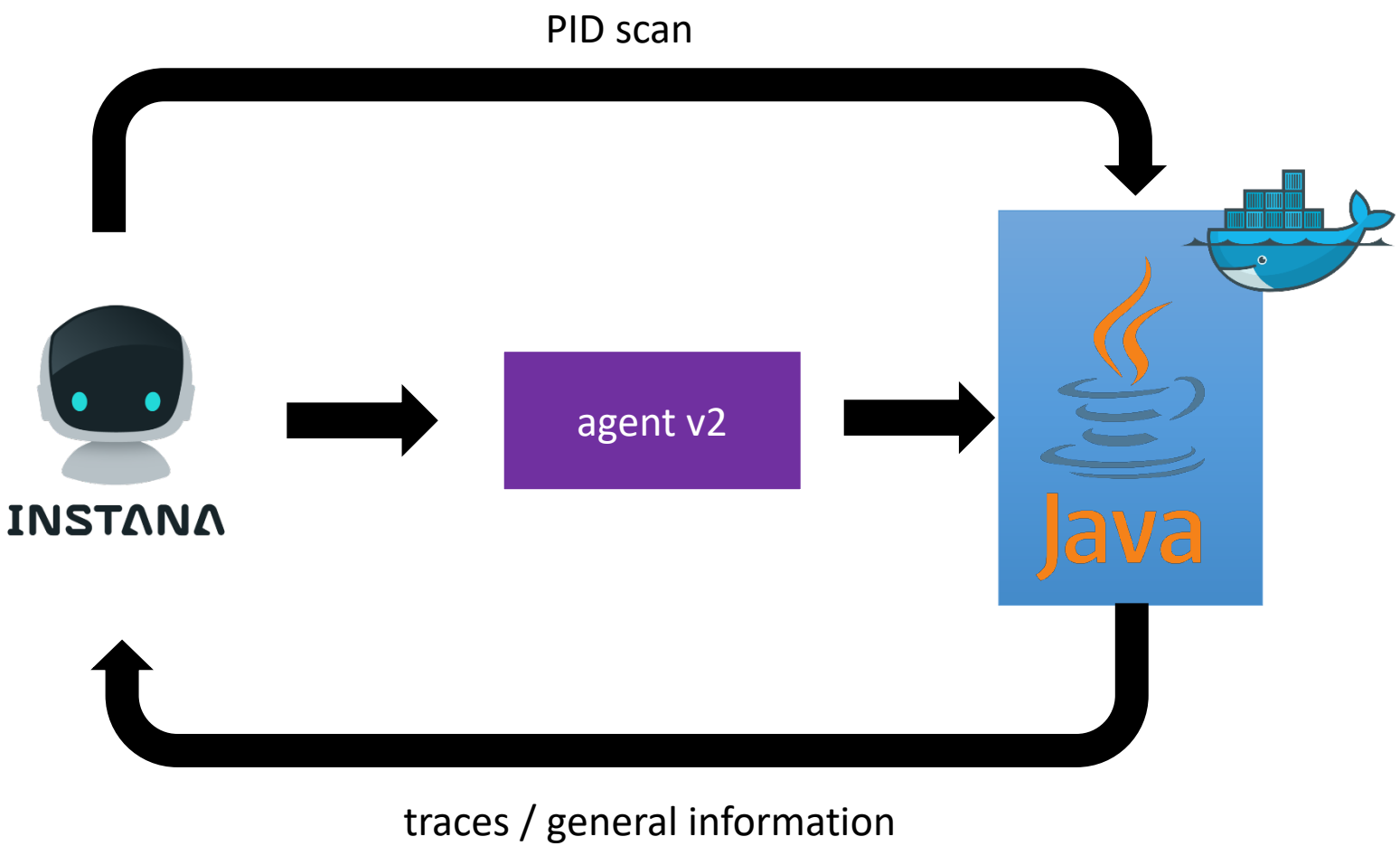
```
ca fe ba be 00 00 00
34 00 10 0a 00 04 00
0d 08 00 09 07 00 0e
07 00 0f 01 00 06 3c
69 6e 69 74 3e 01 00
03 28 29 56 01 00 04
43 6f 64 65 01 00 0f
4c 69 6e 65 4e 75 6d
62 65 72 54 61 62 6c
65 01 00 03 62 61 72
01 00 14 28 29 4c 6a
61 76 61 ...
```

`byte[]`

Example application: Mockito inline-mock maker



Example application: APM



agent fundamentals

code instrumentation

class loading

miscellaneous

Transforming classes using ASM

```
static byte[] transform(byte[] classFile) {
  ClassReader reader = new ClassReader(classFile);
  ClassWriter writer = new ClassWriter(0);
  reader.accept(new ClassVisitor(ASM_7, writer) {
    @Override public MethodVisitor visitMethod(
        int access, String name, String desc,
        String signature, String[] exceptions) {
      MethodVisitor mv = super.visitMethod(access, name, desc, signature, exceptions);
      return new MethodVisitor(ASM7, mv) {
        @Override public void visitCode() {
          super.visitCode();
          visitFieldInsn(GET_STATIC, "java/lang/System",
              "out", "Ljava/lang/PrintStream;");
          visitLdc("Hello World");
          visitMethodInsn(INVOKEINTERFACE, "Ljava/lang/PrintStream;",
              "println", "(Ljava/lang/String;)V", true);
        }
      };
    }
  }, 0);
  return writer.toByteArray();
}
```

Transforming and matching classes using Byte Buddy

```
class PreEntry {
  public static void premain(String arg, Instrumentation inst) {
    new AgentBuilder.Default()
      .type(hasSuperType(named("my.target.UserType")))
      .transform((builder, type, classLoader, module) -> builder
          .method(any())
          .intercept(MethodCall.invoke(
              PrintStream.class.getMethod("println", String.class))
          .onField(System.class.getField("out"))
          .with("Hello world")
          .andThen(SuperMethodCall.INSTANCE)))
      .installOn(inst);
  }
}
```

How is a transformation applied by Byte Buddy?

```
class UserType {

  String foo() {
    return "foo"
  }
}
```

```
class UserType {

  String foo() {
    System.out.println("Hello World!");
    return foo$original();
  }

  String foo$original() {
    return "foo";
  }

  public String toString() {
    System.out.println("Hello World!");
    return super.toString();
  }
}
```

Decorating and retransforming classes using Byte Buddy

```
class AgentEntry {
  public static void agentmain(String arg, Instrumentation inst) {
    new AgentBuilder.Default()
      .disableClassFormatChanges()
      .with(RedefinitionStrategy.RETRANSFORMATION)
      .type(isSubTypeOf(HelloWorldAdvice.class))
      .transform((builder, type, classLoader, module) -> builder
        .visit(Advice.to(HelloWorldAdvice.class).on(isMethod())))
      .installOn(inst);
  }
}
```

```
class HelloWorldAdvice {
  @Advice.OnMethodEnter
  static void onEnter() {
    System.out.println("Hello world!");
  }
}
```

```
class UserType {
  String foo() {
    System.out.println("Hello world!");
    return "foo";
  }
}
```

Advice: exit instrumentation

```
class HelloWorldAdvice {
  @Advice.OnMethodEnter
  static void onEnter() {
    System.out.println("Method enter");
  }
  @Advice.OnMethodExit
  static void onExit() {
    System.out.println("Method exit");
  }
}
```

```
class UserType {
  String foo() {
    System.out.println("Method enter");
    String $return = "foo";
    System.out.println("Method exit");
    return $return;
  }
}
```

Advice: multiple returns

```
class HelloWorldAdvice {
  @Advice.OnMethodEnter
  static void onEnter() {
    System.out.println("Method enter");
  }
  @Advice.OnMethodExit
  static void onExit() {
    System.out.println("Method exit");
  }
}
```

```
class UserType {
  String foo() {
    System.out.println("Method enter");
    String $return;
    if (System.nanoTime() % 2 == 0) {
      $return = "bar";
      return $return;
    }
    doSomething();
    $return = "baz";
    end:
    System.out.println("Method exit");
    return $return;
  }
}
```

Advice: method arguments

```
class HelloWorldAdvice {
  @Advice.OnMethodEnter
  static void onEnter(
      @Advice.Argument(0) String val) {
    System.out.println("Val: " + val);
  }
}
```

```
class UserType {
  String foo(String val) {
    System.out.println("Val: " + val);
  } return "foo" + val;
} }
}
```

Advice: annotation overview

`@Advice.Enter`	Returns value produced by enter advice. Allows for conditional method skipping.
`@Advice.Exit`	Returns value produced by exit advice. Allows for conditional method repeating.
`@Advice.FieldValue`	Allows reading and writing of instance field value.
`@Advice.Argument`	Allows accessing method argument. Argument retention can be set on method level.
`@Advice.AllArguments`	Allows accessing all method arguments.
`@Advice.This`	Allows accessing instrumented instance.
`@Advice.Origin`	Allows accessing method context.
`@Advice.Return`	Allows accessing method return value (if any).
`@Advice.Thrown`	Allows accessing thrown instance by method (if any).
`@Advice.Local`	Allows sharing stack-allocated value between enter and exit advice.

Limitations of advice templates

```
class FaultyAdvice {

  static String value;

  @Advice.OnMethodEnter
  static void onEnter(@MyConstantValue String value) {
    helper(value);
  }
  // #BB-613: since Java 8, a copy would be possible
  private static void helper(String value) {
    // ...
  }
}
```

In-method code substitution

```
class UserType {
  void foo() {
    int random = ThreadLocalRandom.current().nextInt();
    System.out.println(random);
  }
}
```

```
class PreEntry {
  public static void premain(String arg, Instrumentation inst) {
    new AgentBuilder.Default()
      .type(hasSuperType(named("my.target.UserType")))
      .transform((builder, type, classLoader, module) -> builder
      .visit(MemberSubstitution.strict()
        .method(named("println"))
        .replaceWithMethod(named("print").and(takesArguments(String.class))
        .on(named("foo")))
      .installOn(inst);
  }
}
```

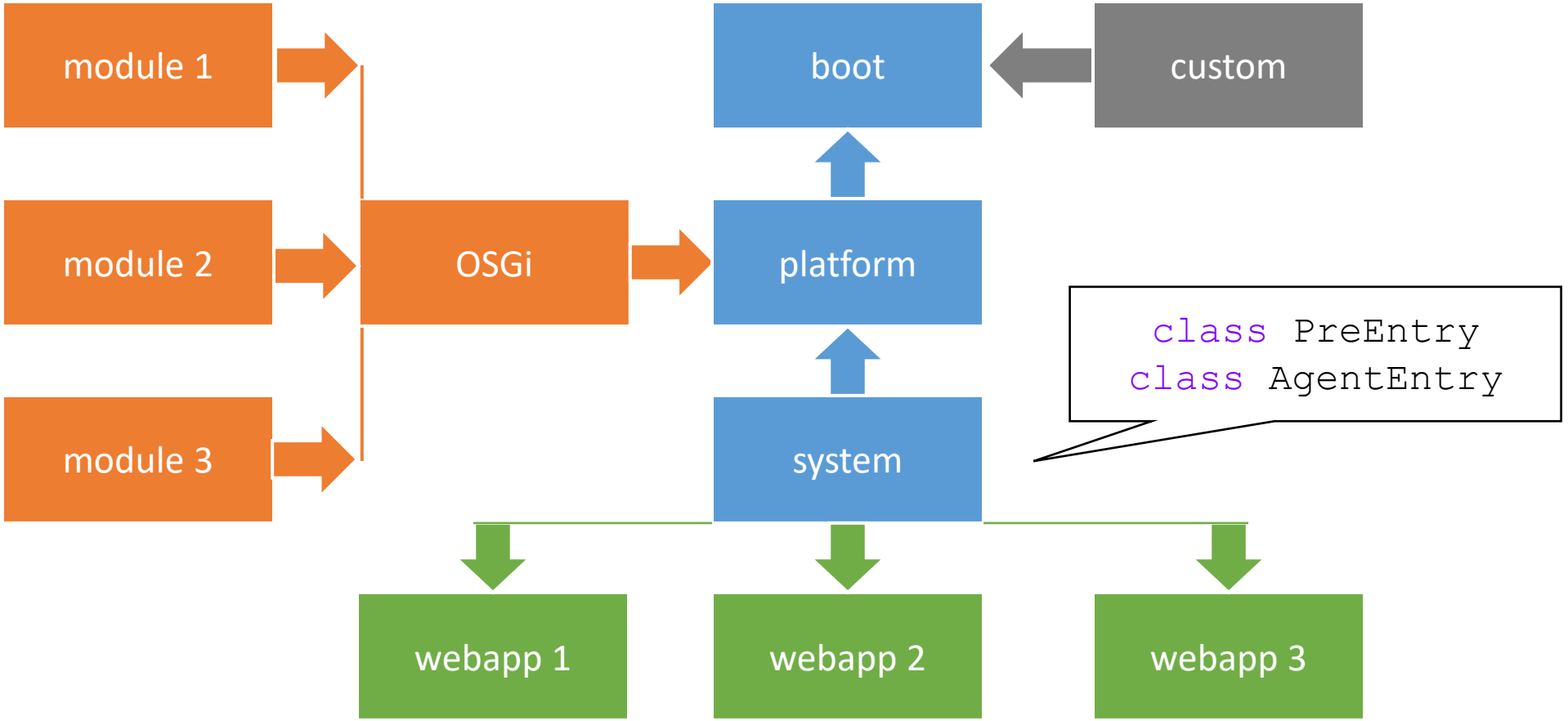
agent fundamentals

code instrumentation

class loading

miscellaneous

Class loader hierarchies



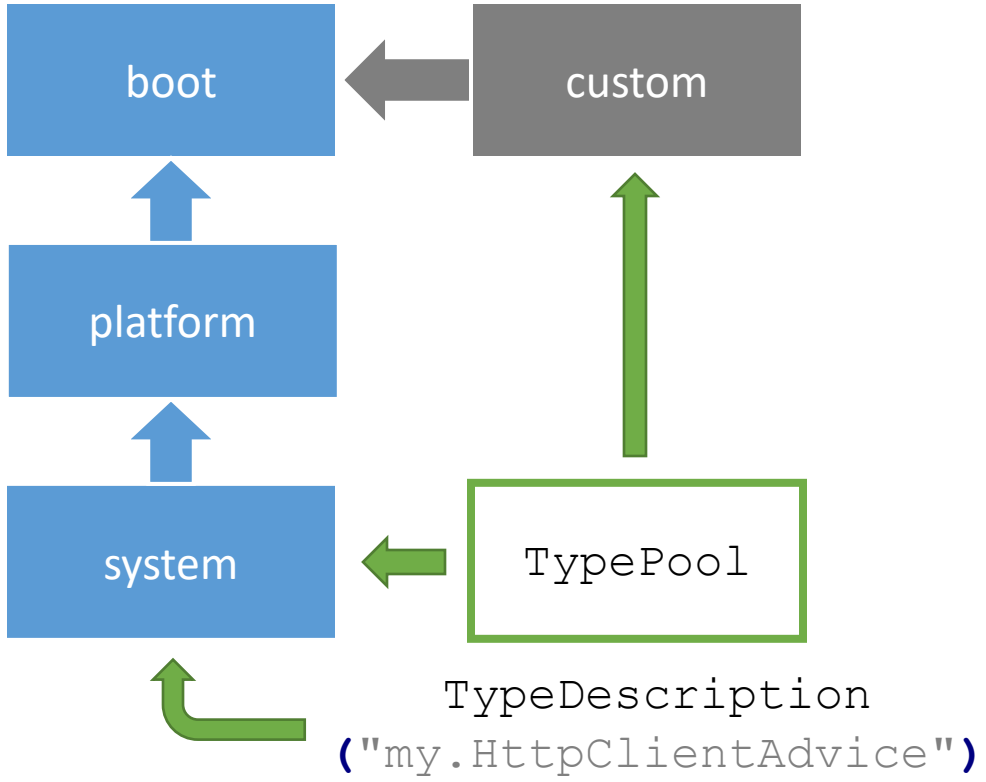
Supporting unknown class loader hierarchies

```
class HttpClientAdvice {
  @Advice.OnMethodEnter
  static long onEnter() {
    return System.currentTimeMillis();
  }
  @Advice.OnMethodExit
  static void onExit(
      @Advice.Argument(0) HttpRequest request,
      @Advice.Return HttpResponse response,
      @Advice.Enter long time) {
    Tracing.record(request.getUrl(),
      response.getStatus(),
      System.currentTimeMillis() - time);
  }
}
```

Supporting unknown class loader hierarchies

```
class PreEntry {
  public static void premain(String arg, Instrumentation inst) {
    new AgentBuilder.Default()
      .type(named("some.HttpClient"))
      .transform((builder, type, classLoader, module) -> builder
        .visit(new AgentBuilder.InstallationListener.ErrorAdvice())
        .visit(Advice.to(HttpClientAdvice.class).on(named("send")))
      .installOn(PreEntry.class.getClassLoader())
  }      .advice(named("send"), "my.HttpClientAdvice")))
}    .installOn(inst);
  }
}
```

Supporting unknown class loader hierarchies

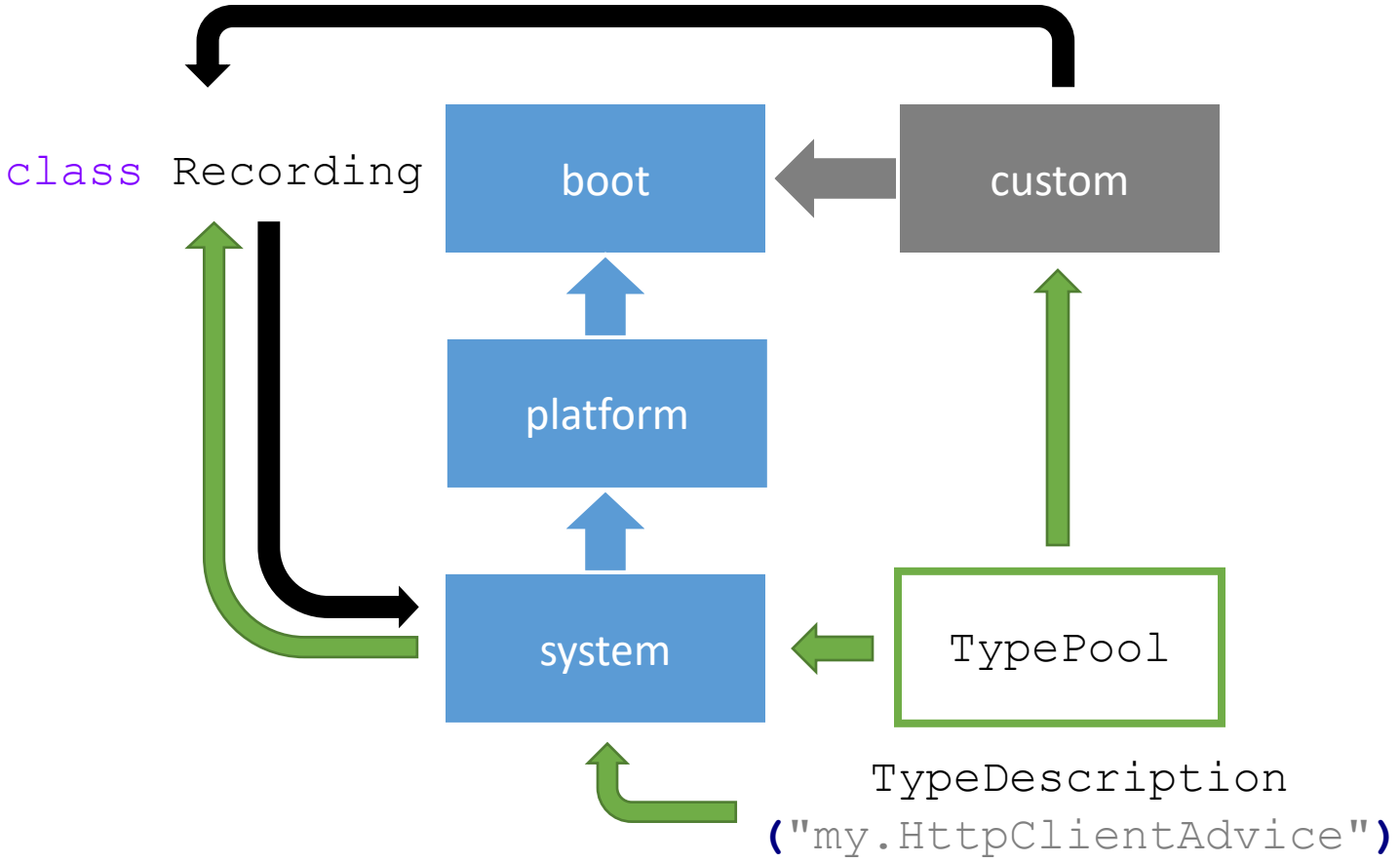


Supporting unknown class loader hierarchies

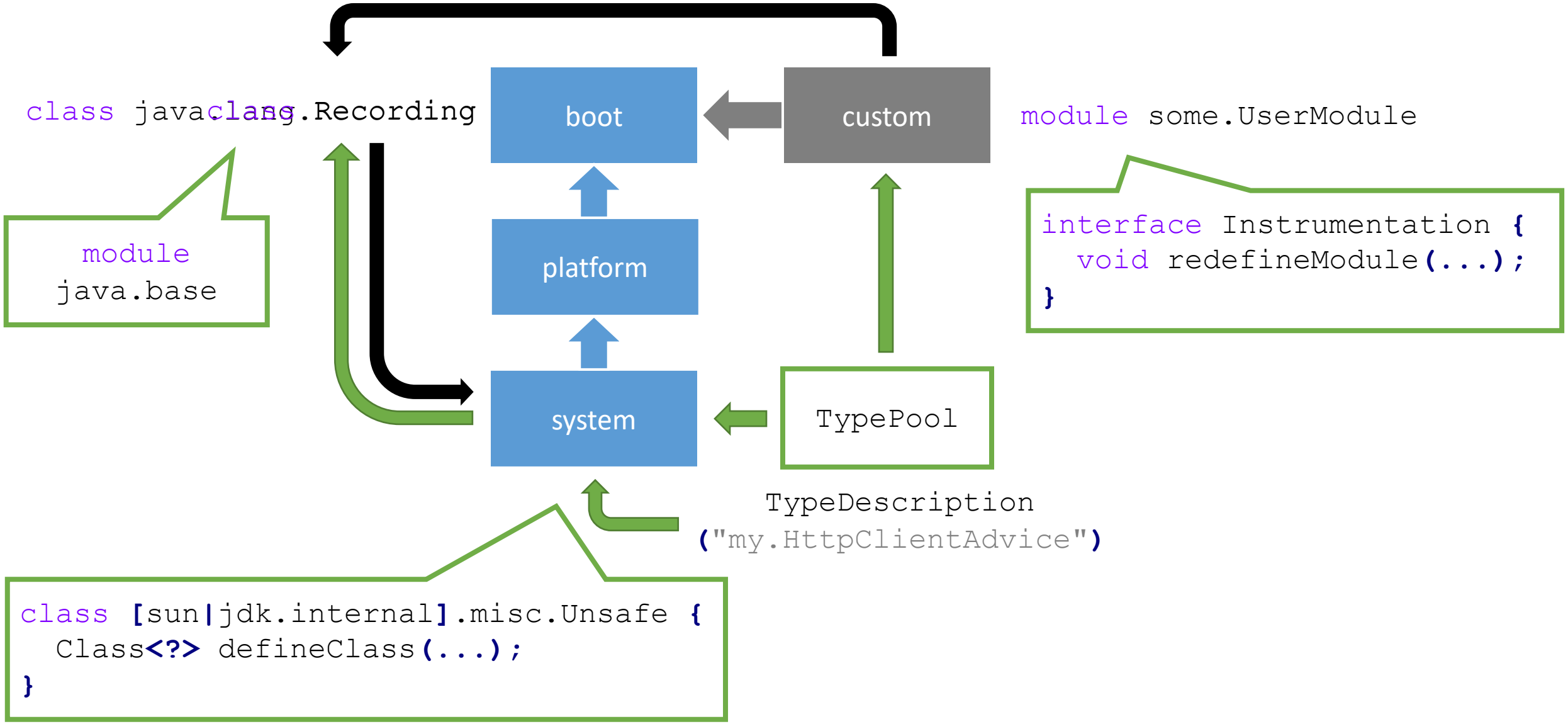
```
class PreEntry {
  public static void premain(String arg, Instrumentation inst) {
    inst.appendToBootstrapClassLoaderSearch(new JarFile("trace.jar"));
    Recording.dispatcher = initializeRecording();
    new AgentBuilder.Default()
      .type(named("some.HttpClient"))
      .transform((builder, type, classLoader, module) -> builder
        .visit(new AgentBuilder.Transformer.ForAdvice()
          .include(PreEntry.class.getClassLoader())
          .advice(named("send"), "my.HttpClientAdvice")))
      .installOn(inst);
  }
}

public abstract class Recording {
  public static Recording dispatcher;
  public static void record(String url, int status, long time) {
    tracing.doRecord(url, status, time);
  }
  protected abstract void doRecord(String url, int status, long time);
}
```

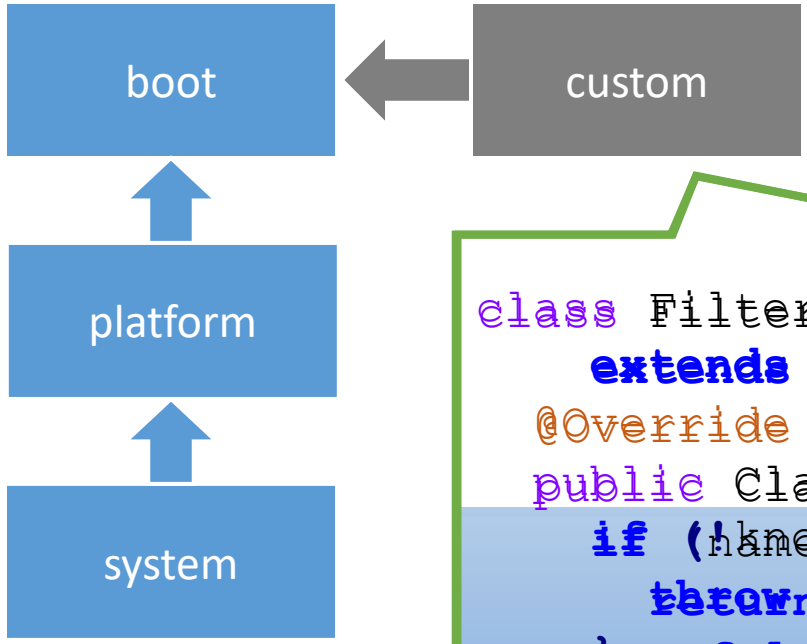
Supporting unknown class loader hierarchies



Supporting unknown module hierarchies



Supporting class loaders with explicit filters

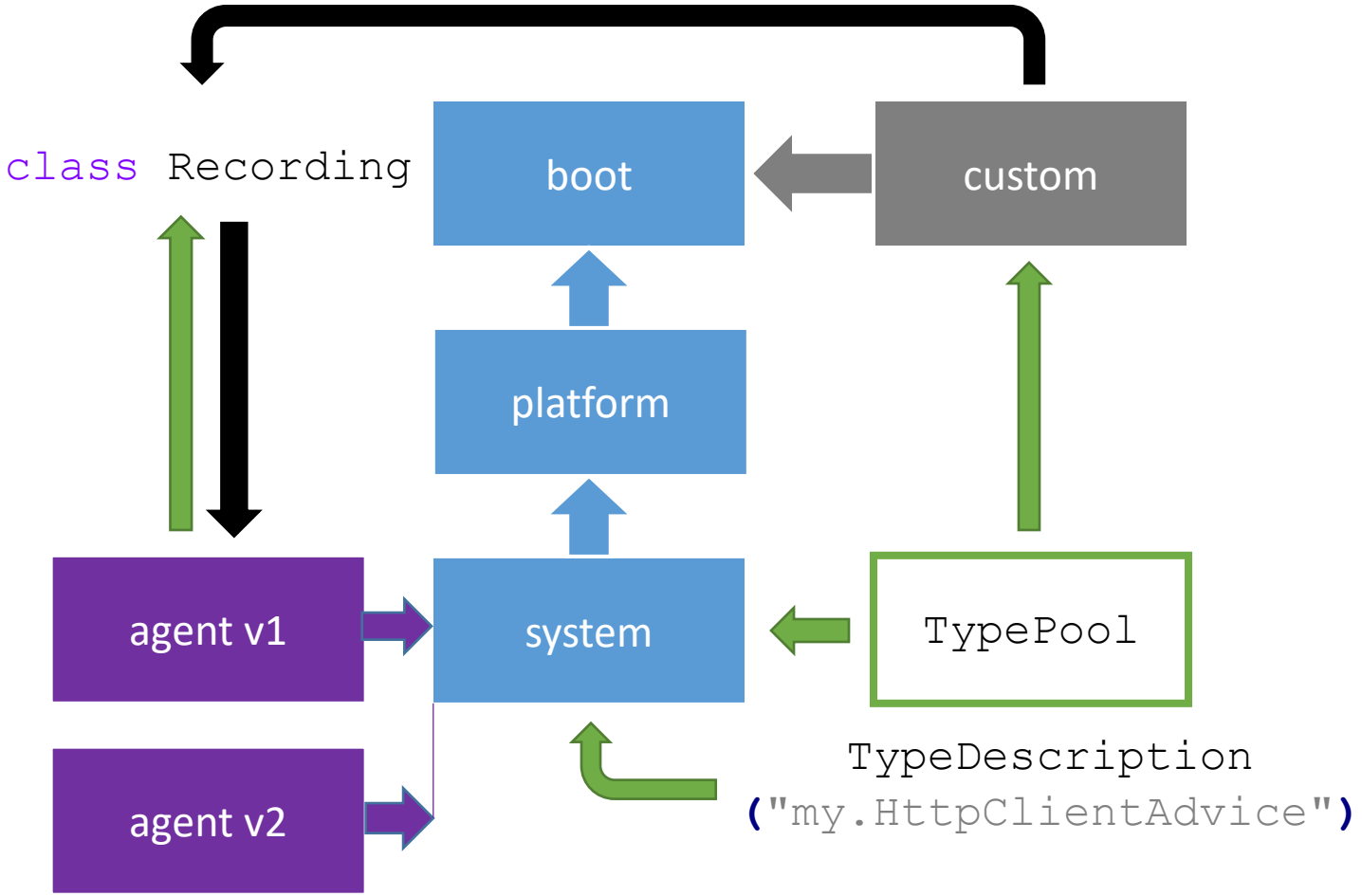



```
class FilteringClassLoader
    extends ClassLoader {
  @Override
  public Class<?> loadClass(String name) {
    if (name.startsWith("java.lang.")) {
      return Class.forName(name,
          false, null);
    }
    if (!knownClasses.contains(named)) {
      throw new IllegalStateException();
    }
    //...
  }
}
```

Making agents updatable

```
class AgentEntry {
  static ClassLoader previous;
  public static void agentmain(String arg, Instrumentation inst)
      throws Exception {
    ClassLoader loader = new URLClassLoader(new URL[] {arg});
    loader.loadClass("agent.EntryPoint")
      .getMethod("init", Instrumentation.class, ClassLoader.class)
      .invoke(inst, previous);
    previous = loader;
  }
}
```

Making agents updatable



agent fundamentals

code instrumentation

class loading

miscellaneous

Self-attaching for testing a Java agent

```
long pid = ProcessHandle.current().pid();
VirtualMachine.attach(String.valueOf(pid));
try {
  vm.loadAgent(MyAttachHelper.class.getProtectionDomain()
    .getCodeSource()
    .getLocation()
    .toURI()
    .getPath(), null);
} finally {
  vm.detach();
}
});
Instrumentation inst = MyAttachHelper.inst;

class MyAttachHelper {
  Instrumentation inst;
  public static void agentmain(String arg, Instrumentation inst) {
    MyAttachHelper.inst = inst;
  }
}
```

Attachment using Byte Buddy

```
Instrumentation inst = ByteBuddyAgent.install();
ByteBuddyAgent.attach(new File("myagent.jar"), "123");
```

```
interface AttachmentProvider
```

1. jdk.attach (JDK only) / IBM-namespace-aware
2. jdk.attach via helper process (JDK only) / IBM-namespace-aware
3. Attachment emulation (via JNA/JNI), no self-attachment constraint (POSIX, Solaris, Windows / HotSpot, OpenJ9)

Instance-bound state without adding fields

```
public class StateHandler { // injected into boot loader
  public static final Map<Object, Object> state =
    new ConcurrentWeakIdentityHashMap<>();
}
```

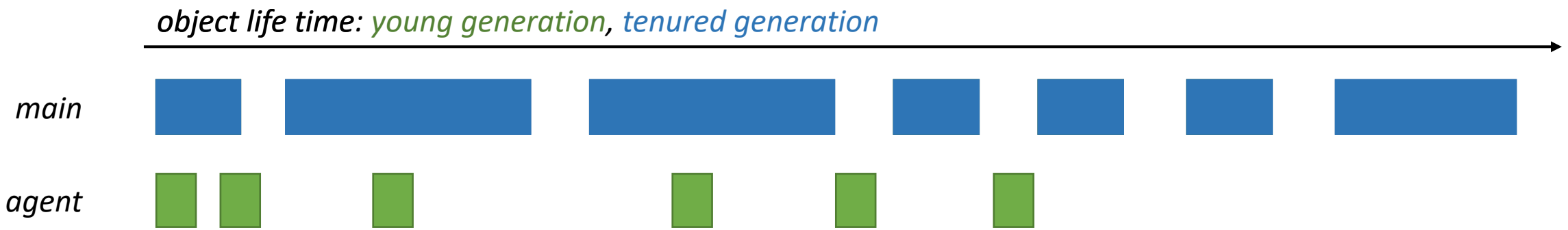
```
class UserClass {
  void foo() {
    // ...
  }
  static {
  } Class.forName(
}       "net.bytebuddy.Nexus", false, ClassLoader.getSystemClassLoader())
      .getMethod("init", Class.class)
      .invoke(null, UserClass.class);
  }
  void foo() {
    dispatcher.run();
    // ...
  }
}
```

Increasing retransformation resilience

```
class AgentEntry {
  public static void agentmain(String arg, Instrumentation inst) {
    new AgentBuilder.Default()
      .disableClassFormatChanges()
      .with(RedefinitionStrategy.RETRANSFORM)
      .with(BatchAllocator.Partitioning.of(4))
      .with(DiscoveryStrategy.Reiterating.INSTANCE)
      .with(new RedefinitionStrategyListener() {
        public void onBatch(int index, List<Class<?>> batch, List<Class<?>> types) {
          System.out.println("Batch " + index + ": " + batch);
        }
        public void onComplete(int amount, List<Class<?>> types, Map<List<Class<?>>, Throwable> failures) {
          System.out.println("Completed with " + failures.size() + " failures");
        }
      })
      .type(ElementMatchers.nameStartsWith("net.bytebuddy.test."))
      .transform((builder, type, classLoader, module) -> builder
        .visit(Advice.to(SomeAdvice.class).on(isMethod())))
      .installOn(inst);
  }
}
```

Balancing performance: JIT and GC inference

```
-XX:MaxInlineLevel // maximum nested calls
-XX:MaxInlineSize // maximum byte code size
-XX:FreqInlineSize // maximum byte code size (frequently called)
-XX:MaxTrivialSize // maximum byte code size (trivial, e.g. getter/setter)
-XX:MinInliningThreshold // minimum amount of calls before inlining
-XX:LiveNodeCountInliningCutoff // max number of live nodes
```



https://rafael.codes
@rafaelcodes

---

https://documents4j.com
https://github.com/documents4j/documents4j

---

https://bytebuddy.net
https://github.com/raphw/byte-buddy