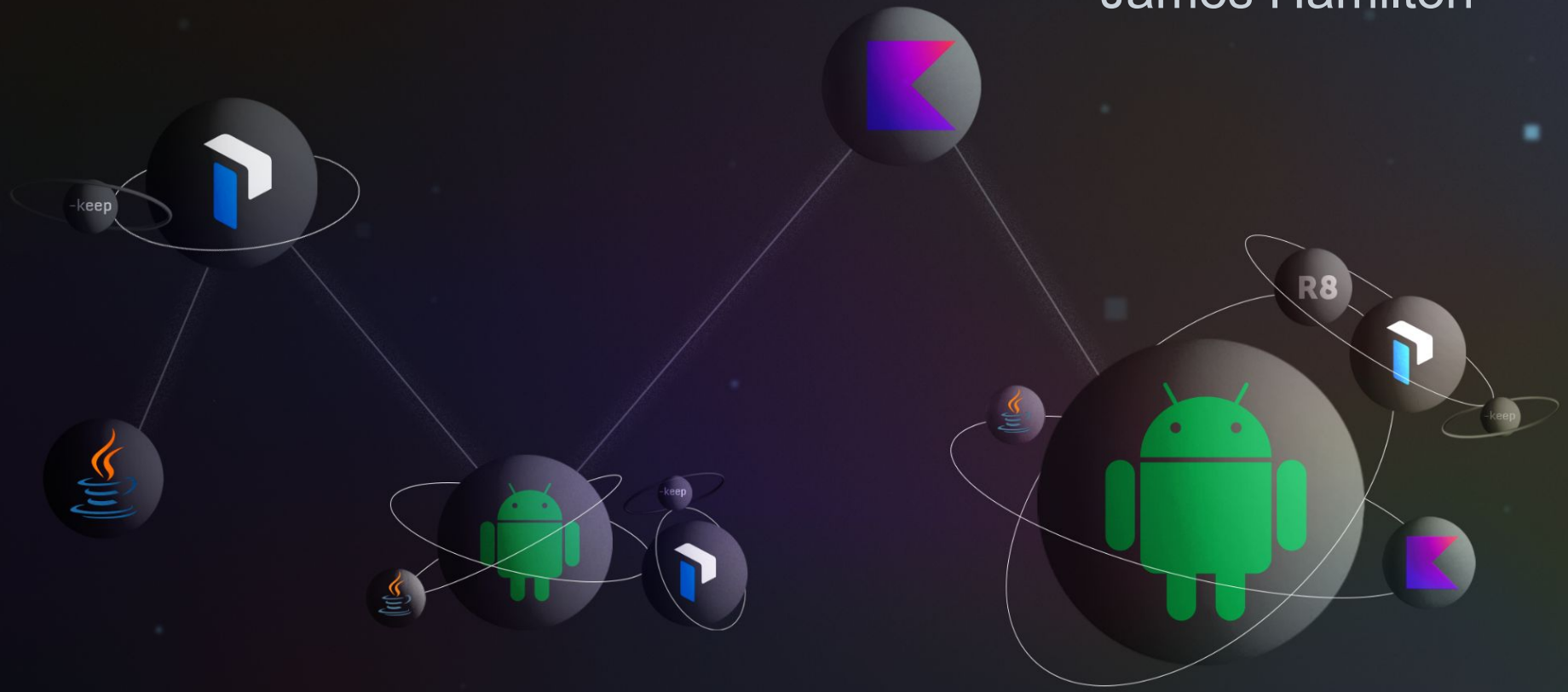
Shrinking in the Age of Kotlin

James Hamilton



Who am I?

Software Engineer @ Guardsquare

- Mobile security
- Java bytecode
- Dalvik bytecode
- Code analysis
- Obfuscation

Previously:

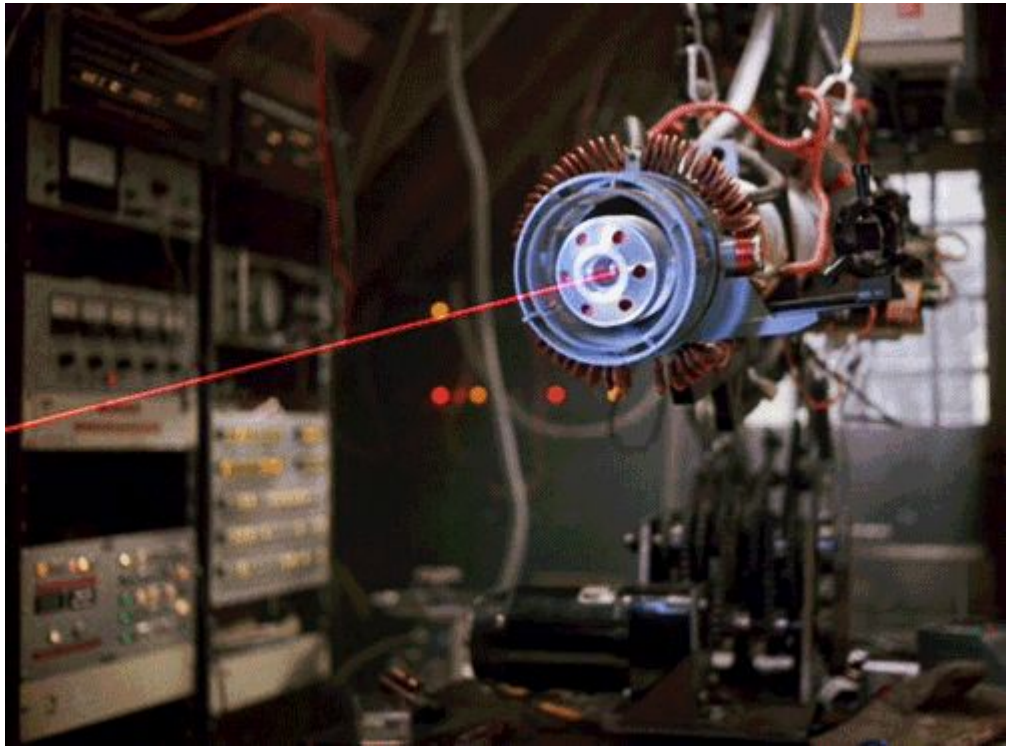
Control Systems @ CERN

PhD Computer Science in code analysis and metrics

What is shrinking?


APK


JAR
JAR




ProGuard


R8


yGuard


Redex

Not A {{Shrinker}} Tutorial

Not A Sales Pitch for {{Shrinker}}

1. How does a shrinker process Kotlin generated code?

2. What’s the difference between Java classes and Kotlin classes?

3. How you can build tools to analyse & modify Kotlin classes

How does a shrinker work?

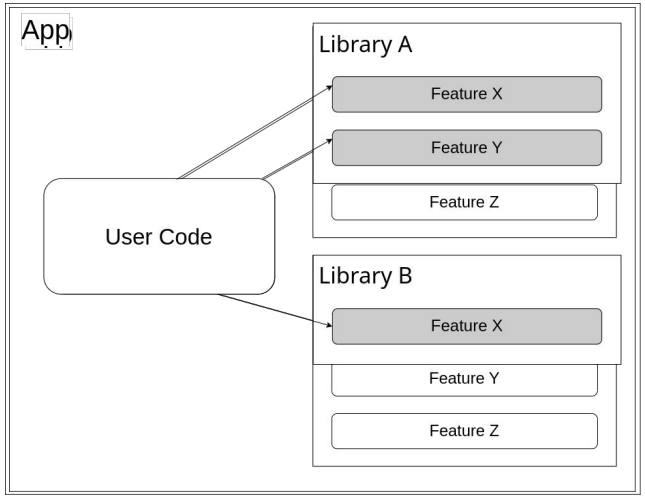
Tree shaking

Code optimization

Name obfuscation

Tree Shaking

Remove unused classes, methods and fields


App
Library A
Feature X
Feature Y
Feature Z
User Code
Library B
Feature X
Feature Y
Feature Z

Code optimization

Rewrite code to reduce size and/or increase performance

```
if (true) {
    println(“OK”);
} else {
    println(“ERR”);
}
```



```
println(“OK”);
```

Name obfuscation

Reduce size by using shorter names


ServiceDefinitionConfigurationBeanPrinterSetterTaskWrapper
a

Name obfuscation alone is not a security solution

In the Age of Kotlin?

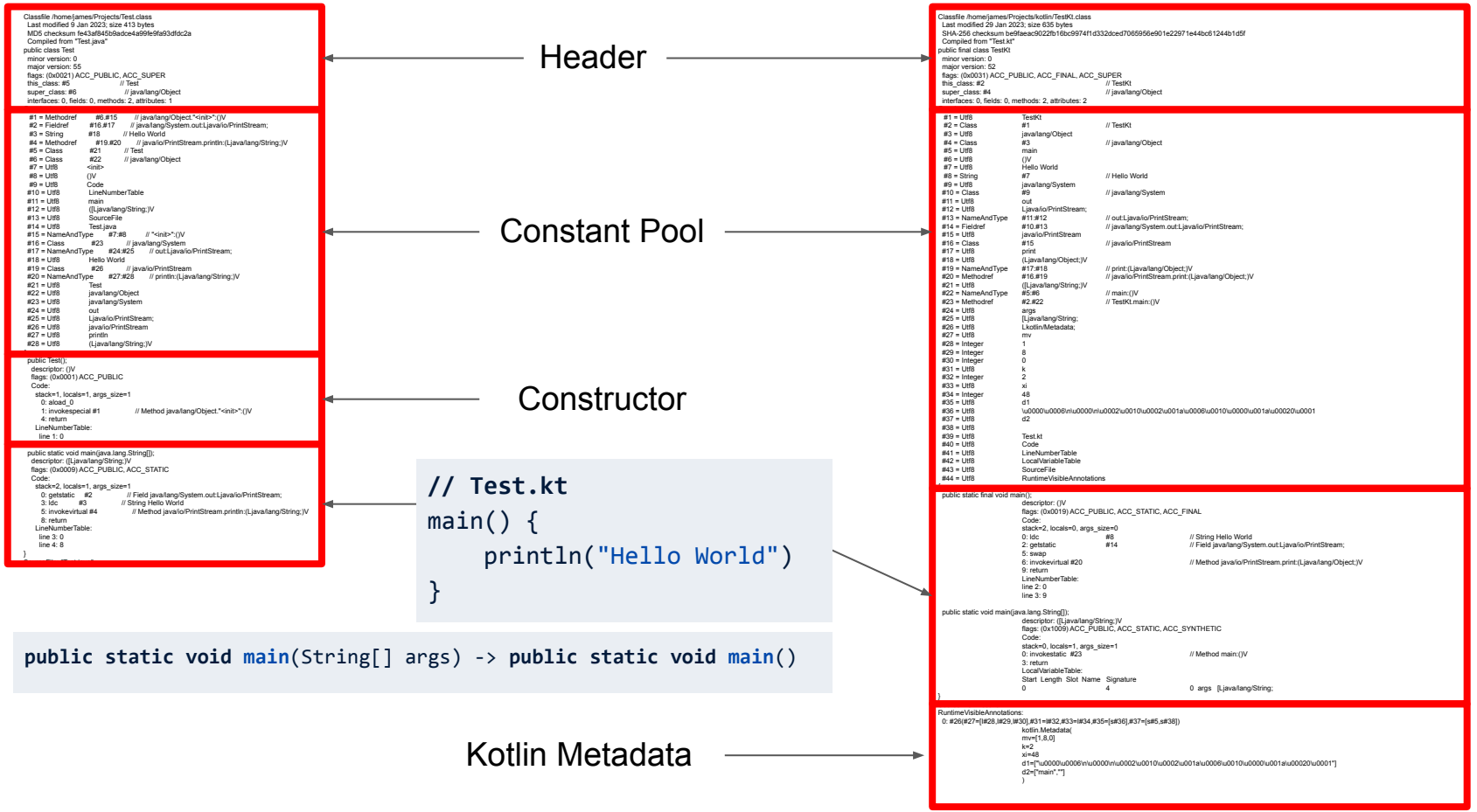
Isn’t it just all Java bytecode?

```java
// Test.java
public class Test {
    public static void main(String[] args) {
      System.out.println("Hello World");
    }
}
```

```
$ javap -c -v -p Test.class
```

```kotlin
// Test.kt
main() {
    println("Hello World")
}
```

```
$ javap -c -v -p TestKt.class
```


Classfile /home/james/Projects/Test.class
Last modified 9 Jan 2023; size 413 bytes
MD5 checksum fe43af845b9adce4a99fe9fa93dfdc2a
Compiled from "Test.java"
public class Test
minor version: 0
major version: 55
flags: (0x0021) ACC_PUBLIC, ACC_SUPER
this_class: #5 // Test
super_class: #6 // java/lang/Object
interfaces: 0, fields: 0, methods: 2, attributes: 1
#1 = Methodref #6.#15 // java/lang/Object."<init>":()V
#2 = Fieldref #16.#17 // java/lang/System.out:Ljava/io/PrintStream;
#3 = String #18 // Hello World
#4 = Methodref #19.#20 // java/io/PrintStream.println:(Ljava/lang/String;)V
#5 = Class #21 // Test
#6 = Class #22 // java/lang/Object
#7 = Utf8 <init>
#8 = Utf8 ()V
#9 = Utf8 Code
#10 = Utf8 LineNumberTable
#11 = Utf8 main
#12 = Utf8 ([Ljava/lang/String;)V
#13 = Utf8 SourceFile
#14 = Utf8 Test.java
#15 = NameAndType #7:#8 // "<init>":()V
#16 = Class #23 // java/lang/System
#17 = NameAndType #24:#25 // out:Ljava/io/PrintStream;
#18 = Utf8 Hello World
#19 = Class #26 // java/io/PrintStream
#20 = NameAndType #27:#28 // println:(Ljava/lang/String;)V
#21 = Utf8 Test
#22 = Utf8 java/lang/Object
#23 = Utf8 java/lang/System
#24 = Utf8 out
#25 = Utf8 Ljava/io/PrintStream;
#26 = Utf8 java/io/PrintStream
#27 = Utf8 println
#28 = Utf8 (Ljava/lang/String;)V
public Test();
descriptor: ()V
flags: (0x0001) ACC_PUBLIC
Code:
stack=1, locals=1, args_size=1
0: aload_0
1: invokespecial #1 // Method java/lang/Object."<init>":()V
4: return
LineNumberTable:
line 1: 0
public static void main(java.lang.String[]);
descriptor: ([Ljava/lang/String;)V
flags: (0x0009) ACC_PUBLIC, ACC_STATIC
Code:
stack=2, locals=1, args_size=1
0: getstatic #2 // Field java/lang/System.out:Ljava/io/PrintStream;
3: ldc #3 // String Hello World
5: invokevirtual #4 // Method java/io/PrintStream.println:(Ljava/lang/String;)V
8: return
LineNumberTable:
line 3: 0
line 4: 8
}
Header
Constant Pool
Constructor
// Test.kt
main() {
println("Hello World")
}
public static void main(String[] args) -> public static void main()
Kotlin Metadata
Classfile /home/james/Projects/kotlin/TestKt.class
Last modified 29 Jan 2023; size 635 bytes
SHA-256 checksum be9faeac9022fb16bc9974f1d3332dced7065956e901e22971e44bc61244b1d5f
Compiled from "Test.kt"
public final class TestKt
minor version: 0
major version: 52
flags: (0x0031) ACC_PUBLIC, ACC_FINAL, ACC_SUPER
this_class: #2 // TestKt
super_class: #4 // java/lang/Object
interfaces: 0, fields: 0, methods: 2, attributes: 2
#1 = Utf8 TestKt
#2 = Class #1 // TestKt
#3 = Utf8 java/lang/Object
#4 = Class #3 // java/lang/Object
#5 = Utf8 main
#6 = Utf8 ()V
#7 = Utf8 Hello World
#8 = String #7 // Hello World
#9 = Utf8 java/lang/System
#10 = Class #9 // java/lang/System
#11 = Utf8 out
#12 = Utf8 Ljava/io/PrintStream;
#13 = NameAndType #11:#12 // out:Ljava/io/PrintStream;
#14 = Fieldref #10.#13 // java/lang/System.out:Ljava/io/PrintStream;
#15 = Utf8 java/io/PrintStream
#16 = Class #15 // java/io/PrintStream
#17 = Utf8 print
#18 = Utf8 (Ljava/lang/Object;)V
#19 = NameAndType #17:#18 // print:(Ljava/lang/Object;)V
#20 = Methodref #16.#19 // java/io/PrintStream.print:(Ljava/lang/Object;)V
#21 = Utf8 ([Ljava/lang/String;)V
#22 = NameAndType #5:#6 // main:()V
#23 = Methodref #2.#22 // TestKt.main:()V
#24 = Utf8 args
#25 = Utf8 [Ljava/lang/String;
#26 = Utf8 Lkotlin/Metadata;
#27 = Utf8 mv
#28 = Integer 1
#29 = Integer 8
#30 = Integer 0
#31 = Utf8 k
#32 = Integer 2
#33 = Utf8 xi
#34 = Integer 48
#35 = Utf8 d1
#36 = Utf8 \u0000\u0006\n\u0000\n\u0002\u0010\u0002\u001a\u0006\u0010\u0000\u001a\u00020\u0001
#37 = Utf8 d2
#38 = Utf8
#39 = Utf8 Test.kt
#40 = Utf8 Code
#41 = Utf8 LineNumberTable
#42 = Utf8 LocalVariableTable
#43 = Utf8 SourceFile
#44 = Utf8 RuntimeVisibleAnnotations
public static final void main();
descriptor: ()V
flags: (0x0019) ACC_PUBLIC, ACC_STATIC, ACC_FINAL
Code:
stack=2, locals=0, args_size=0
0: ldc #8 // String Hello World
2: getstatic #14 // Field java/lang/System.out:Ljava/io/PrintStream;
5: swap
6: invokevirtual #20 // Method java/io/PrintStream.print:(Ljava/lang/Object;)V
9: return
LineNumberTable:
line 2: 0
line 3: 9
public static void main(java.lang.String[]);
descriptor: ([Ljava/lang/String;)V
flags: (0x1009) ACC_PUBLIC, ACC_STATIC, ACC_SYNTHETIC
Code:
stack=0, locals=1, args_size=1
0: invokestatic #23 // Method main:()V
3: return
LocalVariableTable:
Start Length Slot Name Signature
0 4 0 args [Ljava/lang/String;
}
RuntimeVisibleAnnotations:
0: #26(#27=[#28,#29,#30],#31=#32,#33=#34,#35=[s#36],#37=[s#5,s#38])
kotlin.Metadata(
mv=[1,8,0]
k=2
xi=48
d1=["\u0000\u0006\n\u0000\n\u0002\u0010\u0002\u001a\u0006\u0010\u0000\u001a\u00020\u0001"]
d2=["main",""]
)

Why metadata?

```
data class User(val name: String, val age: Int)
```

```
data class User(val name: String, val age: Int)
```


class User { … }

```
context(LoggingContext)
fun foo(withParams: Params) {
    log.info("Operation has started")
}
```



```
public kotlin.Unit foo(LoggingContext context, withParams: Params) {
    context.getLog().info("Operation has started");
}
```

Nullability

Type aliases

```
typealias MyAlias = String
```

Much more…

Missing / invalid metadata is a problem
for code that inspects Kotlin code
e.g. reflection / compiler / IDE

How is the metadata encoded?

```
Classfile /home/james/Projects/Test.class
  Last modified 9 Jan 2023; size 413 bytes
  MD5 checksum fe43af845b9adce4a99fe9fa93dfdc2a
  Compiled from "Test.java"
public class Test
  minor version: 0
  major version: 55
  flags: (0x0021) ACC_PUBLIC, ACC_SUPER
  this_class: #5                          // Test
  super_class: #6                         // java/lang/Object
  interfaces: 0, fields: 0, methods: 2, attributes: 1
Constant pool:
   #1 = Methodref          #6.#15         // java/lang/Object."<init>":()V
   #2 = Fieldref           #16.#17        // java/lang/System.out:Ljava/io/PrintStream;
   #3 = String             #18            // Hello World
   #4 = Methodref          #19.#20        // java/io/PrintStream.println:(Ljava/lang/String;)V
   #5 = Class              #21            // Test
   #6 = Class              #22            // java/lang/Object
   #7 = Utf8               <init>
   #8 = Utf8               ()V
   #9 = Utf8               Code
  #10 = Utf8               LineNumberTable
  #11 = Utf8               main
  #12 = Utf8               ([Ljava/lang/String;)V
  #13 = Utf8               SourceFile
  #14 = Utf8               Test.java
  #15 = NameAndType        #7:#8          // "<init>":()V
  #16 = Class              #23            // java/lang/System
  #17 = NameAndType        #24:#25        // out:Ljava/io/PrintStream;
  #18 = Utf8               Hello World
  #19 = Class              #26            // java/io/PrintStream
  #20 = NameAndType        #27:#28        // println:(Ljava/lang/String;)V
  #21 = Utf8               Test
  #22 = Utf8               java/lang/Object
  #23 = Utf8               java/lang/System
  #24 = Utf8               out
  #25 = Utf8               Ljava/io/PrintStream;
  #26 = Utf8               java/io/PrintStream
  #27 = Utf8               println
  #28 = Utf8               (Ljava/lang/String;)V
{
  public Test();
    descriptor: ()V
    flags: (0x0001) ACC_PUBLIC
    Code:
      stack=1, locals=1, args_size=1
         0: aload_0
         1: invokespecial #1                  // Method java/lang/Object."<init>":()V
         4: return
      LineNumberTable:
        line 1: 0

  public static void main(java.lang.String[]);
    descriptor: ([Ljava/lang/String;)V
    flags: (0x0009) ACC_PUBLIC, ACC_STATIC
    Code:
      stack=2, locals=1, args_size=1
         0: getstatic     #2                  // Field java/lang/System.out:Ljava/io/PrintStream;
         3: ldc           #3                  // String Hello World
         5: invokevirtual #4                  // Method java/io/PrintStream.println:(Ljava/lang/String;)V
         8: return
      LineNumberTable:
        line 3: 0
        line 4: 8
}
SourceFile: "Test.java"
```

Kotlin Metadata

```
Classfile /home/james/Projects/kotlin/TestKt.class
  Last modified 29 Jan 2023; size 635 bytes
  SHA-256 checksum be9faeac9022fb16bc99744f1d332dced7065956e901e22971e44bc61244b1d5f
  Compiled from "Test.kt"
public final class TestKt
  minor version: 0
  major version: 52
  flags: (0x0031) ACC_PUBLIC, ACC_FINAL, ACC_SUPER
  this_class: #2                          // TestKt
  super_class: #4                         // java/lang/Object
  interfaces: 0, fields: 0, methods: 2, attributes: 2
Constant pool:
   #1 = Utf8               TestKt
   #2 = Class              #1             // TestKt
   #3 = Utf8               java/lang/Object
   #4 = Class              #3             // java/lang/Object
   #5 = Utf8               main
   #6 = Utf8               ()V
   #7 = Utf8               Hello World
   #8 = String             #7             // Hello World
   #9 = Utf8               java/lang/System
  #10 = Class              #9             // java/lang/System
  #11 = Utf8               out
  #12 = Utf8               Ljava/io/PrintStream;
  #13 = NameAndType        #11:#12        // out:Ljava/io/PrintStream;
  #14 = Fieldref           #10.#13        // java/lang/System.out:Ljava/io/PrintStream;
  #15 = Utf8               java/io/PrintStream
  #16 = Class              #15            // java/io/PrintStream
  #17 = Utf8               print
  #18 = Utf8               (Ljava/lang/Object;)V
  #19 = NameAndType        #17:#18        // print:(Ljava/lang/Object;)V
  #20 = Methodref          #16.#19        // java/io/PrintStream.print:(Ljava/lang/Object;)V
  #21 = Utf8               ([Ljava/lang/String;)V
  #22 = NameAndType        #5:#6          // main:()V
  #23 = Methodref          #2.#22         // TestKt.main:()V
  #24 = Utf8               args
  #25 = Utf8               [Ljava/lang/String;
  #26 = Utf8               Lkotlin/Metadata;
  #27 = Utf8               mv
  #28 = Integer            1
  #29 = Integer            8
  #30 = Integer            0
  #31 = Utf8               k
  #32 = Integer            2
  #33 = Utf8               xi
  #34 = Integer            48
  #35 = Utf8               d1
  #36 = Utf8               \u0000\u0006\n\u0000\n\u0002\u0010\u0002\u001a\u0006\u0010\u0000\u001a\u00020\u0001
  #37 = Utf8               d2
  #38 = Utf8
  #39 = Utf8               Test.kt
  #40 = Utf8               Code
  #41 = Utf8               LineNumberTable
  #42 = Utf8               LocalVariableTable
  #43 = Utf8               SourceFile
  #44 = Utf8               RuntimeVisibleAnnotations
{
  public static final void main();
                           descriptor: ()V
                           flags: (0x0019) ACC_PUBLIC, ACC_STATIC, ACC_FINAL
                           Code:
                             stack=2, locals=0, args_size=0
                                0: ldc           #8                  // String Hello World
                                2: getstatic     #14                 // Field java/lang/System.out:Ljava/io/PrintStream;
                                5: swap
                                6: invokevirtual #20                 // Method java/io/PrintStream.print:(Ljava/lang/Object;)V
                                9: return
                             LineNumberTable:
                               line 2: 0
                               line 3: 9

  public static void main(java.lang.String[]);
                           descriptor: ([Ljava/lang/String;)V
                           flags: (0x1009) ACC_PUBLIC, ACC_STATIC, ACC_SYNTHETIC
                           Code:
                             stack=0, locals=1, args_size=1
                                0: invokestatic  #23                 // Method main:()V
                                3: return
                             LocalVariableTable:
                               Start  Length  Slot  Name   Signature
                                   0       4     0  args   [Ljava/lang/String;
}
RuntimeVisibleAnnotations:
  0: #26(#27=[#28,#29,#30],#31=#32,#33=#34,#35=[s#36],#37=[s#5,s#38])
                           kotlin.Metadata(
                           mv=[1,8,0]
                           k=2
                           xi=48
                           d1=["\u0000\u0006\n\u0000\n\u0002\u0010\u0002\u001a\u0006\u0010\u0000\u001a\u00020\u0001"]
                           d2=["main",""]
                           )
```

F

```
public annotation class Metadata(
    @get:JvmName("k")
    val kind: Int = 1,
    @get:JvmName("mv")
    val metadataVersion: IntArray = [],
    @get:JvmName("bv")
    val bytecodeVersion: IntArray = [],
    @get:JvmName("d1")
    val data1: Array<String> = [],
    @get:JvmName("d2")
    val data2: Array<String> = [],
    @get:JvmName("xs")
    val extraString: String = "",
    @get:JvmName("pn")
    val packageName: String = "",
    @get:JvmName("xi")
    val extraInt: Int = 0
)
```

..."]

It’s “just” a runtime visible annotation

https://github.com/JetBrains/kotlin/blob/master/libraries/stdlib/jvm/runtime/kotlin/Metadata.kt

```
public annotation class Metadata(
    @get:JvmName("k")
    val kind: Int = 1,
    @get:JvmName("mv")
    val metadataVersion: IntArray = [],
    @get:JvmName("bv")
    val bytecodeVersion: IntArray = [],
    @get:JvmName("d1")
    val data1: Array<String> = [],
    @get:JvmName("d2")
    val data2: Array<String> = [],
    @get:JvmName("xs")
    val extraString: String = "",
    @get:JvmName("pn")
    val packageName: String = "",
    @get:JvmName("xi")
    val extraInt: Int = 0
)
```

kind

1. Class
2. File
3. Synthetic class
4. Multi-file class facade (@JvmMultifileClass)
5. Multi-file class part (@JvmMultifileClass)

```
public annotation class Metadata(
    @get:JvmName("k")
    val kind: Int = 1,
    @get:JvmName("mv")
    val metadataVersion: IntArray = [],
    @get:JvmName("bv")
    val bytecodeVersion: IntArray = [],
    @get:JvmName("d1")
    val data1: Array<String> = [],
    @get:JvmName("d2")
    val data2: Array<String> = [],
    @get:JvmName("xs")
    val extraString: String = "",
    @get:JvmName("pn")
    val packageName: String = "",
    @get:JvmName("xi")
    val extraInt: Int = 0
)
```

metadata **v**ersion

e.g. 1.8.0

bytecode **v**ersion

deprecated

```
public annotation class Metadata(
    @get:JvmName("k")
    val kind: Int = 1,
    @get:JvmName("mv")
    val metadataVersion: IntArray = [],
    @get:JvmName("bv")
    val bytecodeVersion: IntArray = [],
    @get:JvmName("d1")
    val data1: Array<String> = [],
    @get:JvmName("d2")
    val data2: Array<String> = [],
    @get:JvmName("xs")
    val extraString: String = "",
    @get:JvmName("pn")
    val packageName: String = "",
    @get:JvmName("xi")
    val extraInt: Int = 0
)
```

data **1**

Metadata in binary protobuf format

```
public annotation class Metadata(
    @get:JvmName("k")
    val kind: Int = 1,
    @get:JvmName("mv")
    val metadataVersion: IntArray = [],
    @get:JvmName("bv")
    val bytecodeVersion: IntArray = [],
    @get:JvmName("d1")
    val data1: Array<String> = [],
    @get:JvmName("d2")
    val data2: Array<String> = [],
    @get:JvmName("xs")
    val extraString: String = "",
    @get:JvmName("pn")
    val packageName: String = "",
    @get:JvmName("xi")
    val extraInt: Int = 0
)
```

data **2**

Strings referenced by the metadata

```
public annotation class Metadata(
    @get:JvmName("k")
    val kind: Int = 1,
    @get:JvmName("mv")
    val metadataVersion: IntArray = [],
    @get:JvmName("bv")
    val bytecodeVersion: IntArray = [],
    @get:JvmName("d1")
    val data1: Array<String> = [],
    @get:JvmName("d2")
    val data2: Array<String> = [],
    @get:JvmName("xs")
    val extraString: String = "",
    @get:JvmName("pn")
    val packageName: String = "",
    @get:JvmName("xi")
    val extraInt: Int = 0
)
```

extra **s**tring

For a multi-file part class, use for the internal name of the facade class

package **n**ame

Use for the package name if it differs from the Java package due to @JvmPackageName

extra **i**nteger

Bits mean various things

Shrinking?

```
@Metadata(mv = {1, 8, 0}, k = 1, xi = 48, d1 = {...}, d2 = {...})
public final class User {
...
}
```

```
@Metadata(mv = {1, 8, 0}, k = 1, xi = 48, d1 = {...}, d2 = {...})
public final class User {
...
}
```

```
@Metadata(mv = {1, 8, 0}, k = 1, xi = 48, d1 = {..}, d2 =
{"LUser;", "", "name", "", "age", "", "(Ljava/lang/String;I)V",
"getAge", "()I", "getName", "()Ljava/lang/String;", "component1",
"component2", "copy", "equals", "", "other", "hashCode",
"toString"})
public final class A {
    x() { }
    y() { }
    …
}
```

```
@Metadata(mv = {1, 8, 0}, k = 1, xi = 48, d1 = {..}, d2 =
{"LUser;", "", "name", "", "age", "", "(Ljava/lang/String;I)V",
"getAge", "()I", "getName", "()Ljava/lang/String;", "component1",
"component2", "copy", "equals", "", "other", "hashCode",
"toString"})
public final class User {
    getName() { }
    ~~getAge() { }~~
    ...
}
```

How does ProGuard process Kotlin metadata?

class metadata

```
@Metadata {
    kind: 2
    name: MyDataClass
    flags: public data
    functions: {
        sum: {
            flags: public infix
            jvmSignature: sum(II)
        }
        //…
    }
}
public class MyDataClass {
    public int sum(int a, int b) { }
    //…
}
```

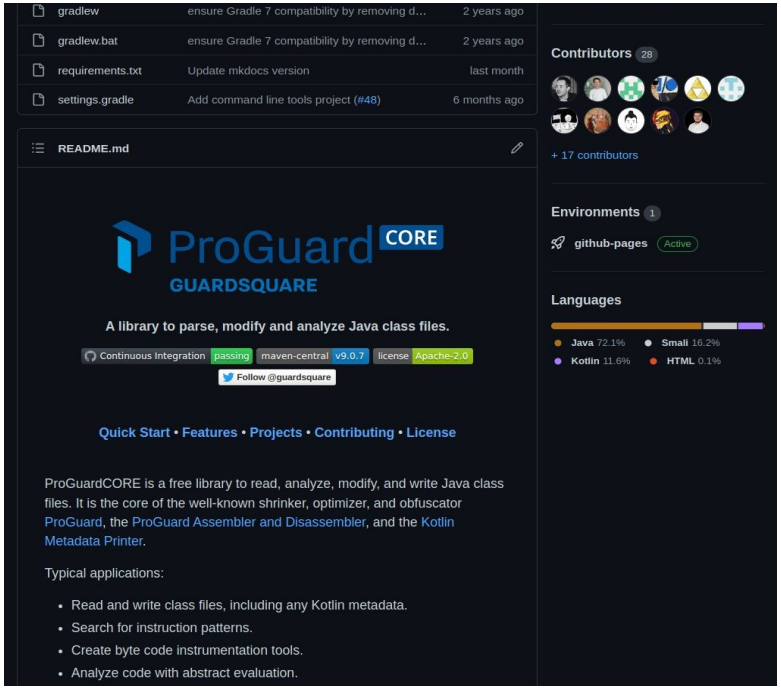
function metadata

- If Java part is renamed: rename Kotlin part
- If Java part is unused : remove Kotlin part

ProGuardCORE

https://github.com/Guardsquare/proguard-core

- Born out of the ProGuard project
- Read and write class + jar files
- Modify, generate and analyse code
- Inspect & modify Kotlin metadata powered by **kotlinx.metadata-jvm**



https://github.com/JetBrains/kotlin/tree/master/libraries/kotlinx-metadata

Reading and modifying Kotlin metadata

```
plugins {
    kotlin("jvm") version "1.8.0"
    application
}

group = "org.example"
version = "1.0-SNAPSHOT"

repositories {
    mavenCentral()
}

dependencies {
    implementation("com.guardsquare:proguard-core:9.0.8")
}
```

Add a dependency on ProGuardCORE

```
fun main() {

}
```

```
fun main() {
    val programClassPool =
        IOUtil.read(
            "build/classes/kotlin/main/MainKt.class",
            /* isLibrary = */ false,
            /* initializeKotlinMetadata = */ true
        )
}
```

```
fun main() {
    val programClassPool =
        IOUtil.read(
            "build/classes/kotlin/main/MainKt.class",
            /* isLibrary = */ false,
            /* initializeKotlinMetadata = */ true
        )
    programClassPool.classesAccept(
        ClassReferenceInitializer(programClassPool,
                                  ClassPool()))
}
```

```
fun main() {
    ...
    programClassPool.classesAccept(
        ReferencedKotlinMetadataVisitor(
            AllFunctionVisitor(
                KotlinFunctionVisitor {  , _, funMetadata ->
                    println(funMetadata.name)
                }
            )
        )
    )
}
```

```
fun main() {
    ...
    programClassPool.classesAccept(
        ReferencedKotlinMetadataVisitor(
            AllFunctionVisitor(
                KotlinFunctionVisitor {  , _, funMetadata ->
                    println(funMetadata.name)
                }
            )
        )
    )
}

fun foo() { }
```

```
fun main() {
    ...
    programClassPool.classesAccept(
        AllMemberVisitor(
            MemberRenamer { clazz, member ->
                when (member.getName(clazz)) {
                    "foo" -> "newFoo"
                    else -> member.getName(clazz)
                }
            }
        )
    )
}

fun foo() { }
```

Metadata is now out of sync!

```
fun main() {
   ...
  programClassPool.classesAccept(
   ReferencedKotlinMetadataVisitor(
    KotlinFileFacadeVisitor { clazz, declContainer ->
     declContainer.functionsAccept(clazz) { _, _, funMetadata ->
      funMetadata.name =
                    funMetadata.referencedMethod.getName(clazz)
     }
    }
   )
  )
}

fun foo() { }
```

```
fun main() {
    ...
  programClassPool.classesAccept(
    ReferencedKotlinMetadataVisitor(
      KotlinFileFacadeVisitor { clazz, declContainer ->
        declContainer.functionsAccept(clazz) { _, _, funMetadata ->
          funMetadata.name =
                        funMetadata.referencedMethod.getName(clazz)
        }
      }
    )
  )
  programClassPool.classesAccept(ClassReferenceFixer(false))
}

fun foo() { }
```

```
fun main() {
    ...
    programClassPool.classesAccept(
        KotlinMetadataWriter { _, error -> println(error) },
    )
}

fun foo() { }
```

```
fun main() {
  ...
  val dataOutputStream =
              DataOutputStream(
              FileOutputStream(
                "build/classes/kotlin/main/MainKt.class"))
  programClassPool.classesAccept(
    ProgramClassWriter(dataOutputStream)
  )
  dataOutputStream.close()
}

fun foo() { }
```


Main.kt
MainKt.class
build.gradle.kts (kotlin-presentation)
Decompiled .class file, bytecode version: 52.0 (Java 8)
1 // IntelliJ API Decompiler stub source generated from a class file
2 // Implementation of methods is not available
3
4 public fun newFoo(): kotlin.Unit { /* compiled code */ }
5
6 public fun main(): kotlin.Unit { /* compiled code */ }
7
8

Next steps

ProGuardCORE manual
https://guardsquare.github.io/proguard-core/

Kotlin Metadata Printer
https://github.com/Guardsquare/kotlin-metadata-printer

kotlinx-metadata-jvm
https://github.com/JetBrains/kotlin/tree/master/libraries/kotlinx-metadata/jvm

ProGuard Playground
https://playground.proguard.com

Twitter: @jag_hamilton