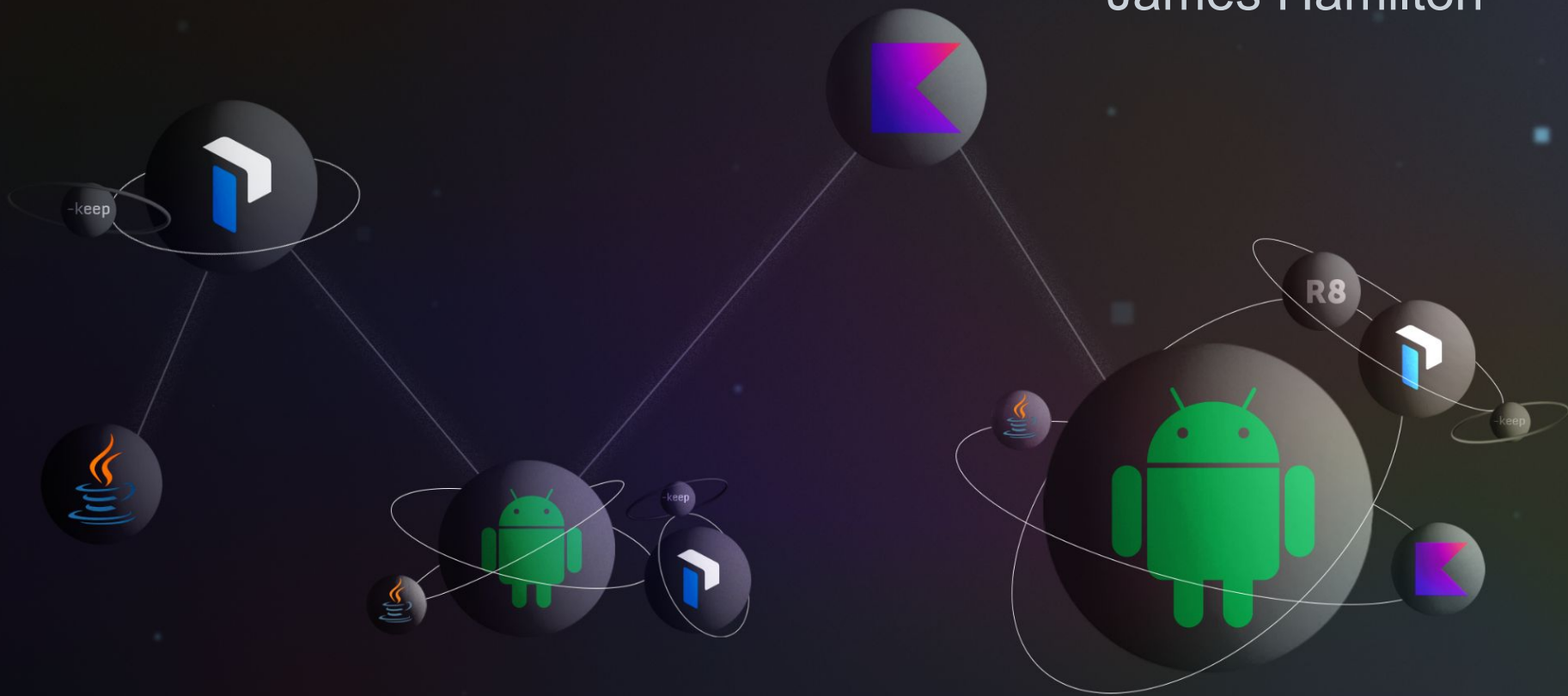


Shrinking in the Age of Kotlin

James Hamilton



Who am I?

Software Engineer @ Guardsquare



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

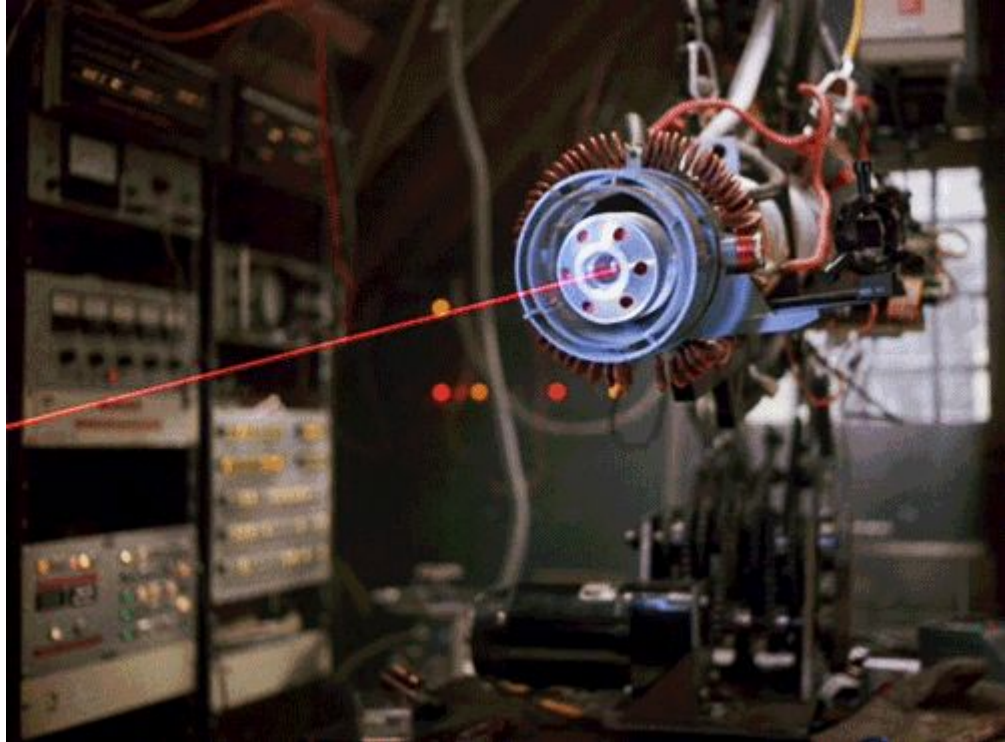
Previously:

Control Systems @ CERN



PhD Computer Science in code analysis and metrics

What is shrinking?





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

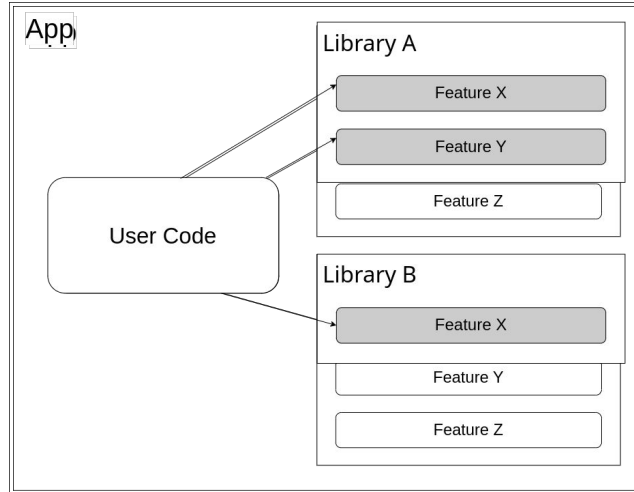
Code optimization

Name obfuscation

Tree Shaking

Remove unused classes, methods and fields





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?


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

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

```
// 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 fe43af845b04b0e4e99f9e9d03d0c2a
Compiled from "Test.java"
public class Test
  minor version: 0
  major version: 55
  flags: (0x0021) ACC_PUBLIC, ACC_SUPER
  this_class: #9 // 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 = URB <init>
#8 = URB (V //
#9 = URB Code
#10 = URB LineNumberTable
#11 = URB main
#12 = URB (Ljava/lang/String;V
#13 = URB SourceFile
#14 = URB Test.java
#15 = NameAndType #17:#8 // <init>()V
#16 = Class #23 // java/lang/Object
#17 = NameAndType #24:#25 // out:Ljava/io/PrintStream;
#18 = URB Hello World
#19 = Class #26 // java/io/PrintStream
#20 = NameAndType #27:#28 // println(Ljava/lang/String;V)
#21 = URB Test
#22 = URB java/lang/Object
#23 = URB java/lang/System
#24 = URB out
#25 = URB Ljava/io/PrintStream;
#26 = URB java/io/PrintStream
#27 = URB println
#28 = URB (Ljava/lang/String;V

public Test();
  descriptor: ()V
  flags: (0x0001) ACC_PUBLIC
  Code:
    stack=1, locals=1, args_size=1
    0: aload_0
    1: invokevirtual #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: 20 Jan 2023; size 535 bytes
SHA-256 checksum bf0faec9022b16bc9974f1d332ced7065956e901e2271e446bc1244b1d5f
Compiled from "TestKt.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 = URB TestKt
#2 = Class #3 // TestKt
#3 = URB java/lang/Object
#4 = Class #5 // java/lang/Object
#5 = URB main
#6 = URB (V //
#7 = URB Hello World
#8 = String #7 // Hello World
#9 = URB java/lang/System
#10 = Class #9 // java/lang/System
#11 = URB out
#12 = URB Ljava/io/PrintStream;
#13 = NameAndType #11:#12 // out:Ljava/io/PrintStream;
#14 = Fieldref #10:#13 // java/lang/System.out:Ljava/io/PrintStream;
#15 = URB java/io/PrintStream
#16 = Class #15 // java/io/PrintStream
#17 = URB print
#18 = URB (Ljava/lang/Object;V // print(Ljava/lang/Object;V)
#19 = NameAndType #17:#18 // java/io/PrintStream.println(Ljava/lang/Object;V)
#20 = Methodref #21:#19 // java/io/PrintStream.println(Ljava/lang/Object;V)
#21 = URB (Ljava/lang/String;V)
#22 = URB #5:#6 // main(V)
#23 = Methodref #22:#2 // TestKt.main(V)
#24 = URB args
#25 = URB (Ljava/lang/String;
#26 = URB Ljdk/metadata;
#27 = URB mv
#28 = Integer 1
#29 = Integer 8
#30 = Integer 0
#31 = Integer k
#32 = Integer 2
#33 = URB v1
#34 = Integer 48
#35 = URB d1
#36 = URB u0000u0006mu0000mu0002u0010u0002u001a0u0006u0010u0000u001a0u0002u0001
#37 = URB c2
#38 = URB
#39 = URB Test.kt
#40 = URB Code
#41 = URB LineNumberTable
#42 = URB LocalVariableTable
#43 = URB SourceFile
#44 = URB 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 #2 // String Hello World
    2: getstatic #3 // Field java/lang/System.out:Ljava/io/PrintStream;
    5: invokevirtual #4 // Method java/io/PrintStream.println(Ljava/lang/String;V)
    8: 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 0 0 4 0 args [Ljava/lang/String;

RuntimeVisibleAnnotations:
  0: #26(#27)=[#28,#29,#30] #31=#32,#33=#34,#35=#36] #37=[#8,#38]
    k=j
    mv=[1,8,0]
    v1=C
    v2=48
    d1=["u0000u0006mu0000mu0002u0010u0002u001a0u0006u0010u0000u001a0u0002u0001"]
    c2=["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 fe43af845b40e4e99fe9a03d8d2a
Compiled from "Test.java"
public class Test
  minor version: 0
  major version: 55
  flags: (0x0021) ACC_PUBLIC, ACC_SUPER
  this_class: #6 // 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 = URB <int>
    #8 = URB (V
    #9 = URB Code
    #10 = URB LineNumberTable
    #11 = URB main
    #12 = URB (Ljava/lang/String;)V
    #13 = URB SourceFile
    #14 = URB Test.java
    #15 = NameAndType #7.#8 // <init>()V
    #16 = Class #23 // java/lang/System
    #17 = NameAndType #24.#25 // out:Ljava/io/PrintStream;
    #18 = URB Hello World
    #19 = Class #26 // java/io/PrintStream
    #20 = NameAndType #27.#28 // println(Ljava/lang/String;)V
    #21 = URB Test
    #22 = URB java/lang/Object
    #23 = URB java/lang/System
    #24 = URB out
    #25 = URB Ljava/io/PrintStream;
    #26 = URB java/io/PrintStream
    #27 = URB println
    #28 = URB (Ljava/lang/String;)V
  {
    public Test();
    descriptor: (V
    flags: (0x0001) ACC_PUBLIC
    Code:
      stack=1, locals=1, args_size=1
      0: aload_0
      1: invokestatic #1 // Method java/lang/Object.<init>()V
      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"

```

```

Classfile home\james\Projects\Kotlin\Test\K.class
Last modified: 20 Jan 2023; size 535 bytes
SHA-256 checksum b6f9eac9022b16bc99741d332dced7065956e901e22071e44bc61244d15f
Compiled from "Test.k"
public final class TestK
  minor version: 0
  major version: 52
  flags: (0x0031) ACC_PUBLIC, ACC_FINAL, ACC_SUPER
  this_class: #2 // TestK
  super_class: #4 // java/lang/Object
  interfaces: 0, fields: 0, methods: 2, attributes: 2
  Constant pool:
    #1 = URB TestK
    #2 = Class #2 // Class java/lang/Object // TestK
    #3 = URB #3 // java/lang/Object
    #4 = Class #3 // java/lang/Object
    #5 = URB main
    #6 = URB (V
    #7 = URB Hello World
    #8 = String #7 // Hello World
    #9 = URB java/lang/System // java/lang/System
    #10 = Class #9 // java/lang/System
    #11 = URB out
    #12 = URB Ljava/io/PrintStream;
    #13 = NameAndType #11.#12 // out:Ljava/io/PrintStream;
    #14 = Fieldref #10.#13 // java/lang/System.out:Ljava/io/PrintStream;
    #15 = URB java/io/PrintStream
    #16 = Class #15 // java/lang/System.out:Ljava/io/PrintStream;
    #17 = URB print
    #18 = URB (Ljava/lang/Object;)V
    #19 = NameAndType #17.#18 // print(Ljava/lang/Object;)V
    #20 = Methodref #16.#19 // java/io/PrintStream.println(Ljava/lang/Object;)V
    #21 = URB (Ljava/lang/String;)V
    #22 = NameAndType #21.#22 // main(V
    #23 = Methodref #21.#22 // TestK.main(V)
    #24 = URB args
    #25 = URB (Ljava/lang/String;
    #26 = URB Ljdk/internal/ClassFileMetadata;
    #27 = URB mv
    #28 = Integer 1
    #29 = Integer 8
    #30 = Integer 0
    #31 = URB k
    #32 = Integer 2
    #33 = URB xi
    #34 = Integer 48
    #35 = URB d1
    #36 = URB u\0000u0006vu0000vu0002u0010u0002u0010u0006u0010u0000u0010u0000u0010u0000u0002u0001
    #37 = URB c2
    #38 = URB
    #39 = URB Test.kt
    #40 = URB Code
    #41 = URB LineNumberTable
    #42 = URB LocalVariableTable
    #43 = URB SourceFile
    #44 = URB 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 #6 // String Hello World
      2: getstatic #2 // Field java/lang/System.out:Ljava/io/PrintStream;
      5: invokevirtual #4 // Method java/io/PrintStream.println(Ljava/lang/String;)V
      8: 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 0 4 0 args [Ljava/lang/String;
  }
  RuntimeVisibleAnnotations:
    0: #26{#7={#28,#29,#30},#31={#32,#33=#34,#35={#36},#37={#38},#39}}
      kotlin.Metadata
      mv={1,8,0}
      k=2
      xi=48
      d1=["\u0000u0006vu0000vu0002u0010u0002u0010u0006u0010u0000u0010u0000u0010u0000u0010u0000u0002u0001"]
      c2=["main"]
  }

```

Kotlin Metadata



```

RuntimeVisibleAnnotations:
  0: #26{#7={#28,#29,#30},#31={#32,#33=#34,#35={#36},#37={#38},#39}}
    kotlin.Metadata
    mv={1,8,0}
    k=2
    xi=48
    d1=["\u0000u0006vu0000vu0002u0010u0002u0010u0006u0010u0000u0010u0000u0010u0000u0010u0000u0002u0001"]
    c2=["main"]
  }

```

```

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

```
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 version

e.g. 1.8.0

bytecode version

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 string

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

package name

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

extra integer

Bits mean various things

Shrinking?

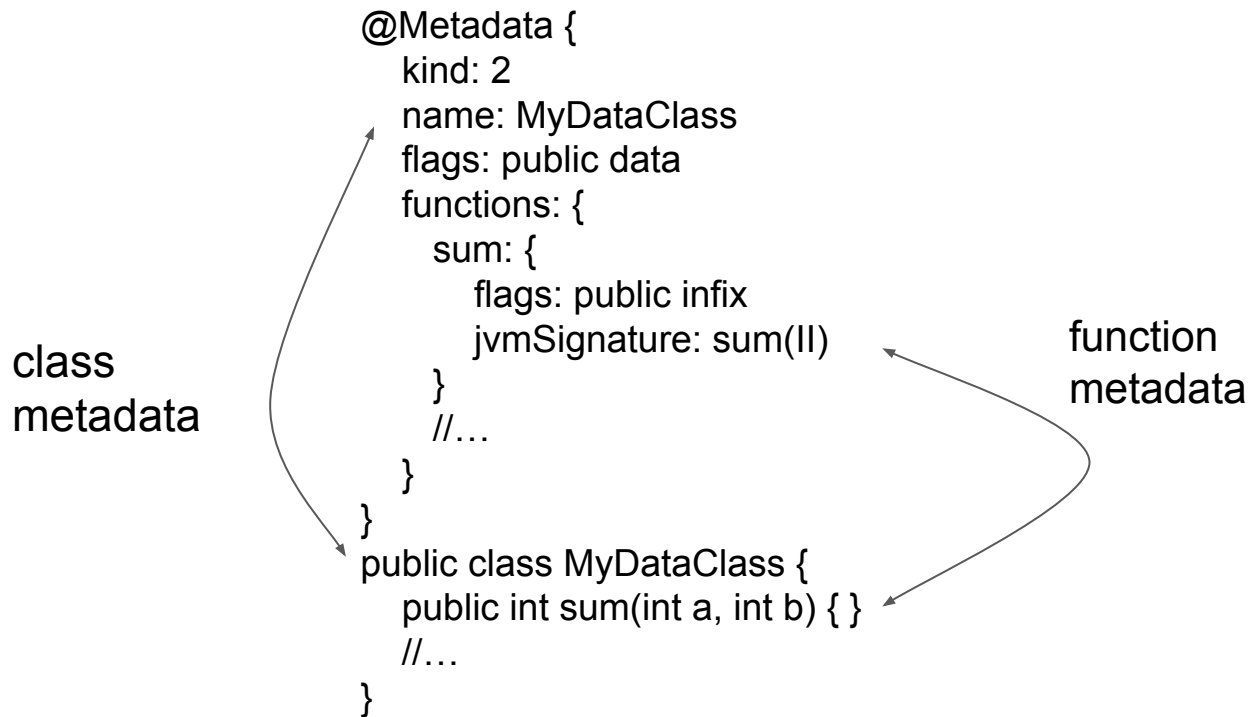
```
@Metadata(mv = {1, 8, 0}, k = 1, xi = 48, d1 = {...}, d2 = {...})  
public final class User {  
    ...  
}
```

```
@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?



- 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**



The screenshot shows the GitHub repository page for ProGuardCORE. The repository name is "ProGuardCORE" by "GUARDSQUARE". The description is "A library to parse, modify and analyze Java class files." The repository has 28 contributors and 17 environments. The languages section shows a bar chart with the following data:

| Language | Percentage |
|----------|------------|
| Java | 72.1% |
| Small | 16.2% |
| Kotlin | 11.6% |
| HTML | 0.1% |

The repository also features a "Quick Start" section with links to "Features", "Projects", "Contributing", and "License". The main content area contains a description of the library and its applications, including reading and writing class files, searching for instruction patterns, creating byte code instrumentation tools, and analyzing code with abstract evaluation.

<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)  
            })  
        )  
    )  
}
```



The screenshot shows the Run console of an IDE. At the top, there is a 'Run' button and several status icons. Below that, the command path is displayed: `/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java ...`. The output shows the word `main`. At the bottom, a message states: `Process finished with exit code 0`. The console also features navigation icons on the left side, such as up/down arrows, a refresh icon, and a trash icon.

```
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)  
                }  
            }  
        )  
    )  
}
```



Metadata is now out of sync!

```
fun foo() { }
```



```
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)

i 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>

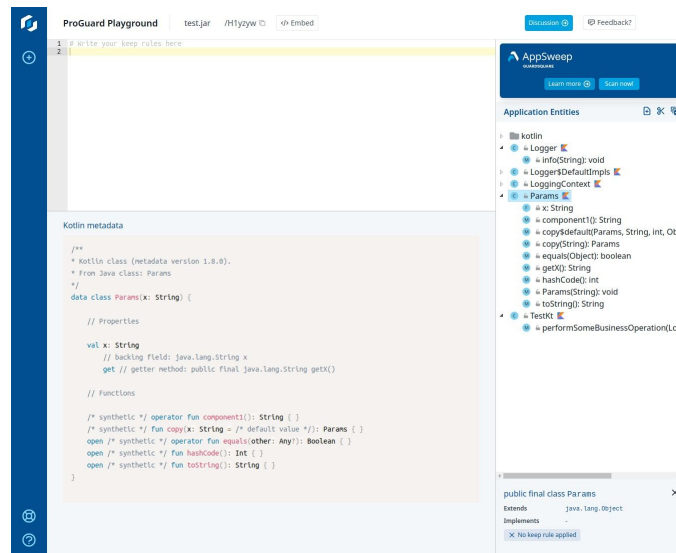
kotlinx-metadata-jvm

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

ProGuard Playground

<https://playground.proguard.com>

Twitter: @jag_hamilton



The screenshot displays the ProGuard Playground interface. The top bar shows the URL 'test.jar /Hjzyw' and an 'Embed' button. Below the URL bar is a text input field with the placeholder '# Write your keep rules here'. The main content area is divided into two sections. The left section, titled 'Kotlin metadata', shows the following code:

```
Kotlin metadata
/**
 * Kotlin class (metadata version 1.0.0).
 * From Java class: Params
 */
data class Params(x: String) {

    // Properties

    val x: String
    // backing field: java.lang.String x
    get // getter method: public final java.lang.String getX()

    // Functions

    /* synthetic */ operator fun component1(): String ()
    /* synthetic */ fun copy(x: String = /* default value */): Params ()
    open /* synthetic */ operator fun equals(other: Any?): Boolean ()
    open /* synthetic */ fun hashCode(): Int ()
    open /* synthetic */ fun toString(): String ()
}

public final class Params
Extends java.lang.Object
Implements
No keep rule applied
```

The right section, titled 'Application Entities', shows a tree view of the application's entities. The tree is expanded to show the 'Params' class, which has the following members:

- x: String
- component(): String
- copyDefault(Params, String, Int, Obj): Params
- copy(String): Params
- equals(Object): Boolean
- getX(): String
- hashCode(): Int
- Params(String): void
- toString(): String
- TestKit
- performSomeBusinessOperationLoc

