Kotlin Code Generation

Alec Strong & Jake Wharton

```java
public final class User {
  private final String firstName;
  private final String lastName;
  private final int age;

  public User(String firstName, String lastName, int age) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.age = age;
  }

  public String getFirstName() {
    return firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public int getAge() {
    return age;
  }

  @Override public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    User user = (User) o;
    return age == user.age
        && Objects.equals(firstName, user.firstName)
        && Objects.equals(lastName, user.lastName);
  }

  @Override public int hashCode() {
    return Objects.hash(firstName, lastName, age);
  }

  @Override public String toString() {
    return "User{"
        + "firstName='"
        + firstName
        + '\''
        + ", lastName='"
        + lastName
        + '\''
        + ", age="
        + age
        + '}';
  }
}
```

```kotlin
data class User(
    val firstName: String,
    val lastName: String,
    val age: Int
)
```

Java™

Bytecode

Protocol Buffers

Protocol Buffers

SQL

Protocol Buffers

SQL

Swagger

Protocol Buffers
SQL
Swagger
Android XML

Protocol Buffers
SQL
Swagger
Android XML
YAML

SQL
Swagger
Android XML
YAML
Mirrors/Elements

Swagger
Android XML
YAML
Mirrors/Elements
PSI

Android XML

YAML

Mirrors/Elements

PSI

UAST

YAML

Mirrors/Elements

PSI

UAST

???

Protocol Buffers
SQL
Swagger
Android XML
YAML
Mirrors/Elements
PSI
UAST
???

Protocol Buffers

```
syntax = "proto2";

package com.sample;

message Ogre {
  required string name = 1;
  repeated int32 layers = 2;
  optional string swamp = 3;
}
```

```java
public final String name;
public final List<Integer> layers;
public final String swamp;

public Ogre(String name, List<Integer> layers, String swamp) {
  this(name, layers, swamp, ByteString.EMPTY);
}

public Ogre(String name, List<Integer> layers, String swamp,
    ByteString unknownFields) {
  super(ADAPTER, unknownFields);
  this.name = name;
  this.layers = Internal.immutableCopyOf("layers", layers);
  this.swamp = swamp;
}
```

```java
public final String name;
public final List<Integer> layers;
public final String swamp;

public Ogre(String name, List<Integer> layers, String swamp) {
  this(name, layers, swamp, ByteString.EMPTY);
}

public Ogre(String name, List<Integer> layers, String swamp,
    ByteString unknownFields) {
  super(ADAPTER, unknownFields);
  this.name = name;
  this.layers = Internal.immutableCopyOf("layers", layers);
  this.swamp = swamp;
}
```

```java
public final String name;
public final List<Integer> layers;
public final String swamp;

public Ogre(String name, List<Integer> layers, String swamp) {
  this(name, layers, swamp, ByteString.EMPTY);
}

public Ogre(String name, List<Integer> layers, String swamp,
    ByteString unknownFields) {
  super(ADAPTER, unknownFields);
  this.name = name;
  this.layers = Internal.immutableCopyOf("layers", layers);
  this.swamp = swamp;
}
```

```
public final String name;
public final List<Integer> layers;
public final String swamp;

public Ogre(String name, List<Integer> layers, String swamp) {
  this(name, layers, swamp, ByteString.EMPTY);
}

public Ogre(String name, List<Integer> layers, String swamp,
    ByteString unknownFields) {
  super(ADAPTER, unknownFields);
  this.name = name;
  this.layers = Internal.immutableCopyOf("layers", layers);
  this.swamp = swamp;
}
```

```java
public final String name;
public final List<Integer> layers;
public final String swamp;

public Ogre(String name, List<Integer> layers, String swamp) {
  this(name, layers, swamp, ByteString.EMPTY);
}

public Ogre(String name, List<Integer> layers, String swamp,
    ByteString unknownFields) {
  super(ADAPTER, unknownFields);
  this.name = name;
  this.layers = Internal.immutableCopyOf("layers", layers);
  this.swamp = swamp;
}
```

```java
public final String name;
public final List<Integer> layers;
public final String swamp;

public Ogre(String name, List<Integer> layers, String swamp) {
  this(name, layers, swamp, ByteString.EMPTY);
}

public Ogre(String name, List<Integer> layers, String swamp,
    ByteString unknownFields) {
  super(ADAPTER, unknownFields);
  this.name = name;
  this.layers = Internal.immutableCopyOf("layers", layers);
  this.swamp = swamp;
}
```

```
class Ogre @JvmOverloads constructor(
    val name: String,
    val layers: List<Int>,
    val swamp: String?,
    unknownFields: ByteString = ByteString.EMPTY
) : Message<Ogre, Ogre.Builder>(ADAPTER, unknownFields)
```

```
class Ogre @JvmOverloads constructor(
    val name: String,
    val layers: List<Int>,
    val swamp: String?,
    unknownFields: ByteString = ByteString.EMPTY
) : Message<Ogre, Ogre.Builder>(ADAPTER, unknownFields)
```

```
Ogre shrek = new Ogre("Shrek", Arrays.asList(1, 2), null);
System.out.println(shrek.getSwamp());
```

```kotlin
class Ogre @JvmOverloads constructor(
    @JvmField val name: String,
    @JvmField val layers: List<Int>,
    @JvmField val swamp: String?,
    unknownFields: ByteString = ByteString.EMPTY
) : Message<Ogre, Ogre.Builder>(ADAPTER, unknownFields)
```

```java
Ogre shrek = new Ogre("Shrek", Arrays.asList(1, 2), null);
System.out.println(shrek.swamp);
```

```
override fun equals(other: Any?): Boolean {
  if (other === this) return true
  if (other !is OgreKt) return false
  val o = other as OgreKt?
  return (unknownFields() == o!!.unknownFields()
      && name == o.name
      && layers == o.layers
      && Internal.equals(swamp, o.swamp))
}

override fun hashCode(): Int {
  var result = super.hashCode
  if (result == 0) {
    result = unknownFields().hashCode()
    result = result * 37 + name.hashCode()
    result = result * 37 + layers.hashCode()
    result = result * 37 + (swamp?.hashCode() ?: 0)
    super.hashCode = result
  }
  return result
}

override fun toString(): String {
  val builder = StringBuilder()
  builder.append(", name=").append(name)
  if (!layers.isEmpty()) builder.append(", layers=").append(layers)
  if (swamp != null) builder.append(", swamp=").append(swamp)
  return builder.replace(0, 2, "Ogre{").append('}').toString()
}
```

```
class Ogre @JvmOverloads constructor(
    @JvmField val name: String,
    @JvmField val layers: List<Int>,
    @JvmField val swamp: String?,
    unknownFields: ByteString = ByteString.EMPTY
) : Message<Ogre, Ogre.Builder>(ADAPTER, unknownFields)
```

```
data class Ogre @JvmOverloads constructor(
    @JvmField val name: String,
    @JvmField val layers: List<Int>,
    @JvmField val swamp: String?,
    unknownFields: ByteString = ByteString.EMPTY
) : Message<Ogre, Ogre.Builder>(ADAPTER, unknownFields)
```

```
data class Ogre @JvmOverloads constructor(
    @JvmField val name: String,
    @JvmField val layers: List<Int>,
    @JvmField val swamp: String?,
    unknownFields: ByteString = ByteString.EMPTY
) : Message<Ogre, Ogre.Builder>(ADAPTER, unknownFields)
```

```
data class Ogre @JvmOverloads constructor(
    @JvmField val name: String,
    @JvmField val layers: List<Int>,
    @JvmField val swamp: String?,
    private val unknownFields: ByteString = ByteString.EMPTY
) : Message<Ogre, Ogre.Builder>(ADAPTER, unknownFields)
```

```
class Ogre @JvmOverloads constructor(
    @JvmField val name: String,
    @JvmField val layers: List<Int>,
    @JvmField val swamp: String?,
    unknownFields: ByteString = ByteString.EMPTY
) : Message<Ogre, Ogre.Builder>(ADAPTER, unknownFields)
```

```java
public static final class Builder extends Message.Builder<Ogre, Builder> {
  public String name;
  public List<Integer> layers;
  public String swamp;

  public Builder() {
    layers = Internal.newMutableList();
  }

  public Builder name(String name) {
    this.name = name;
    return this;
  }

  public Builder layers(List<Integer> layers) {
    Internal.checkElementsNotNull(layers);
    this.layers = layers;
    return this;
  }

  public Builder swamp(String swamp) {
    this.swamp = swamp;
    return this;
  }

  @Override public Ogre build() {
    if (name == null) {
      throw Internal.missingRequiredFields(name, "name");
    }
    return new Ogre(name, layers, swamp, super.buildUnknownFields());
  }
}
```

```java
public static final class Builder extends Message.Builder<Ogre, Builder> {
  public String name;
  public List<Integer> layers;
  public String swamp;

  public Builder() {
    layers = Internal.newMutableList();
  }

  public Builder name(String name) {
    this.name = name;
    return this;
  }

  public Builder layers(List<Integer> layers) {
    Internal.checkElementsNotNull(layers);
    this.layers = layers;
    return this;
  }

  public Builder swamp(String swamp) {
    this.swamp = swamp;
    return this;
  }

  @Override public Ogre build() {
    if (name == null) {
      throw Internal.missingRequiredFields(name, "name");
    }
    return new Ogre(name, layers, swamp, super.buildUnknownFields());
  }
}
```

```java
public static final class Builder extends Message.Builder<Ogre, Builder> {
  public String name;
  public List<Integer> layers;
  public String swamp;

  public Builder() {
    layers = Internal.newMutableList();
  }

  public Builder name(String name) {
    this.name = name;
    return this;
  }

  public Builder layers(List<Integer> layers) {
    Internal.checkElementsNotNull(layers);
    this.layers = layers;
    return this;
  }

  public Builder swamp(String swamp) {
    this.swamp = swamp;
    return this;
  }

  @Override public Ogre build() {
    if (name == null) {
      throw Internal.missingRequiredFields(name, "name");
    }
    return new Ogre(name, layers, swamp, super.buildUnknownFields());
  }
}
```

```
class Ogre @JvmOverloads constructor(
    @JvmField val name: String,
    @JvmField val layers: List<Int> = emptyList(),
    @JvmField val swamp: String? = null,
    unknownFields: ByteString = ByteString.EMPTY
) : Message<Ogre, Ogre.Builder>(ADAPTER, unknownFields)
```

```
val shrek = Ogre(
    name = "shrek",
    layers = listOf(1, 2),
    swamp = null
)
```

```
Ogre shrek = new Ogre();
```

@NotNull String name, @NotNull List<Integer> layers, @Nullable String swamp, @NotNull ByteString unknownFields
@NotNull String name, @NotNull List<Integer> layers, @Nullable String swamp
@NotNull String name, @NotNull List<Integer> layers
@NotNull String name

```java
Ogre shrek = new Ogre("Shrek", Arrays.asList(1, 2), "Shrek");
```

Java Interop

```
class Ogre(
    @JvmField val name: String,
    @JvmField val layers: List<Int> = emptyList(),
    @JvmField val swamp: String? = null
)
```

```
interface Ogre {
    val name: String
    val layers: List<Int>
    val swamp: String?
}
```

```kotlin
interface Ogre {
    val name: String
    val layers: List<Int>
    val swamp: String?
}
```

```java
shrek.getLayers();
```

```
inline fun <reified T> Ogre.findFriendOfType(): T? {
  for (friend in friends()) {
    if (friend is T) return friend
  }
  return null
}
```

```
inline fun <reified T> Ogre.findFriendOfType(): T? {
  for (friend in friends()) {
    if (friend is T) return friend
  }
  return null
}


shrek.findFriendOfType<Donkey>()
```

```kotlin
inline fun <reified T> Ogre.findFriendOfType(): T? {
  for (friend in friends()) {
    if (friend is T) return friend
  }
  return null
}


shrek.findFriendOfType<Donkey>()
```

```java
shrek.<Donkey>findFriendOfType();
```

```
inline fun <reified T> Ogre.findFriendOfType(): T? {
  for (friend in friends()) {
    if (friend is T) return friend
  }
  return null
}
```

```
fun <T> Ogre.findFriendOfType(type: Class<T>): T? {
  for (friend in friends()) {
    if (type.isInstance(friend)) return friend as T?
  }
  return null
}

inline fun <reified T> Ogre.findFriendOfType(): T? {
  for (friend in friends()) {
    if (friend is T) return friend
  }
  return null
}
```

```
fun <T> Ogre.findFriendOfType(type: Class<T>): T? {
  for (friend in friends()) {
    if (type.isInstance(friend)) return friend as T?
  }
  return null
}

inline fun <reified T> Ogre.findFriendOfType(): T? {
  return findFriendOfType(T::class.java)
}
```

```
interface SwampChangedListener {
  fun swampChanged(newSwamp: String)
}

fun addSwampChangedListener(listener: SwampChangedListener) = …
```

```kotlin
interface SwampChangedListener {
  fun swampChanged(newSwamp: String)
}

fun addSwampChangedListener(listener: SwampChangedListener) = …
```

```java
shrek.addSwampChangedListener(new SwampChangedListener() {
  @Override public void swampChanged(@NotNull String newSwamp) {
    System.out.println("What are you doing in " + newSwamp);
  }
});
```

```kotlin
interface SwampChangedListener {
  fun swampChanged(newSwamp: String)
}

fun addSwampChangedListener(listener: SwampChangedListener) = …
```

```java
shrek.addSwampChangedListener(newSwamp -> {
  System.out.println("What are you doing in " + newSwamp);
});
```

```
interface SwampChangedListener {
  fun swampChanged(newSwamp: String)
}

fun addSwampChangedListener(listener: SwampChangedListener) = …
```

```kotlin
interface SwampChangedListener {
  fun swampChanged(newSwamp: String)
}

fun addSwampChangedListener(listener: SwampChangedListener) = …
```

```kotlin
shrek.addSwampChangedListener(object : SwampChangedListener {
  override fun swampChanged(newSwamp: String) {
    System.out.println("What are you doing in $newSwamp")
  }
})
```

KT-7770

```
fun addSwampChangedListener(listener: (String) -> Unit) = …
```

```
fun addSwampChangedListener(listener: (String) -> Unit) = …

shrek.addSwampChangedListener { newSwamp ->
  System.out.println("What are you doing in $newSwamp")
}
```

```
fun addSwampChangedListener(listener: (String) -> Unit) = …

shrek.addSwampChangedListener { newSwamp ->
  System.out.println("What are you doing in $newSwamp")
}
```

```
shrek.addSwampChangedListener(newSwamp -> {
  System.out.println("What are you doing in " + newSwamp);
  return Unit.INSTANCE;
});
```

KT-21018

```kotlin
fun addSwampChangedListener(listener: SwampChangedListener) = …
```

```java
public interface SwampChangedListener {
  void swampChanged(String newSwamp);
}
```

```kotlin
fun addSwampChangedListener(listener: SwampChangedListener) = …


shrek.addSwampChangedListener(SwampChangedListener { newSwamp ->
  System.out.println("What are you doing in $newSwamp")
})
```

```java
public interface SwampChangedListener {
  void swampChanged(String newSwamp);
}
```

```kotlin
fun addSwampChangedListener(listener: SwampChangedListener) = …
```

```java
public interface SwampChangedListener {
  void swampChanged(String newSwamp);
}

shrek.addSwampChangedListener(newSwamp -> {
  System.out.println("What are you doing in " + newSwamp);
});
```

```
sealed class Optional<T : Any>
data class Some<T : Any>(val value: T): Optional<T>()
object None : Optional<Nothing>()
```

```
sealed class Optional<T : Any>
data class Some<T : Any>(val value: T): Optional<T>()
object None : Optional<Nothing>()

fun <T : Any> T?.asOptional() = if (this == null) None else Some(this)
```

```kotlin
@file:JvmName("Optionals")

sealed class Optional<T : Any>
data class Some<T : Any>(val value: T): Optional<T>()
object None : Optional<Nothing>()

@JvmName("ofNullable")
fun <T : Any> T?.asOptional() = if (this == null) None else Some(this)
```

```kotlin
sealed class Optional<T : Any> {
  companion object {
    @JvmName("ofNullable") @JvmStatic
    fun <T : Any> T?.asOptional() = if (this == null) None else Some(this)
  }
}
data class Some<T : Any>(val value: T): Optional<T>()
object None : Optional<Nothing>()
```

```
sealed class Optional<T : Any> {
  companion object {
    @JvmStatic
    fun <T : Any> ofNullable(value: T?) = value.asOptional()
  }
}
data class Some<T : Any>(val value: T): Optional<T>()
object None : Optional<Nothing>()

fun <T : Any> T?.asOptional() = if (this == null) None else Some(this)
```

KT-15286

```kotlin
@file:JvmName("-Optionals")

sealed class Optional<T : Any> {
  companion object {
    @JvmStatic
    fun <T : Any> ofNullable(value: T?) = value.asOptional()
  }
}
data class Some<T : Any>(val value: T): Optional<T>()
object None : Optional<Nothing>()

fun <T : Any> T?.asOptional() = if (this == null) None else Some(this)
```

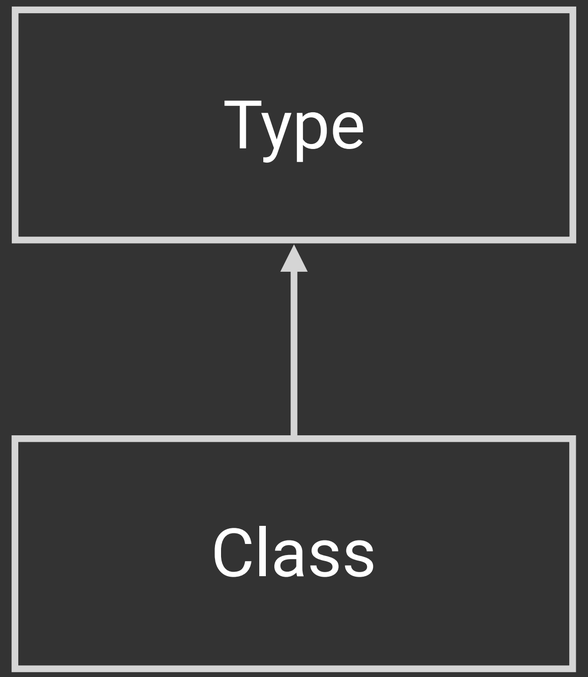
```
class Foo (
    val bar: Nothing,
    val listBar: List<Nothing>
)
```

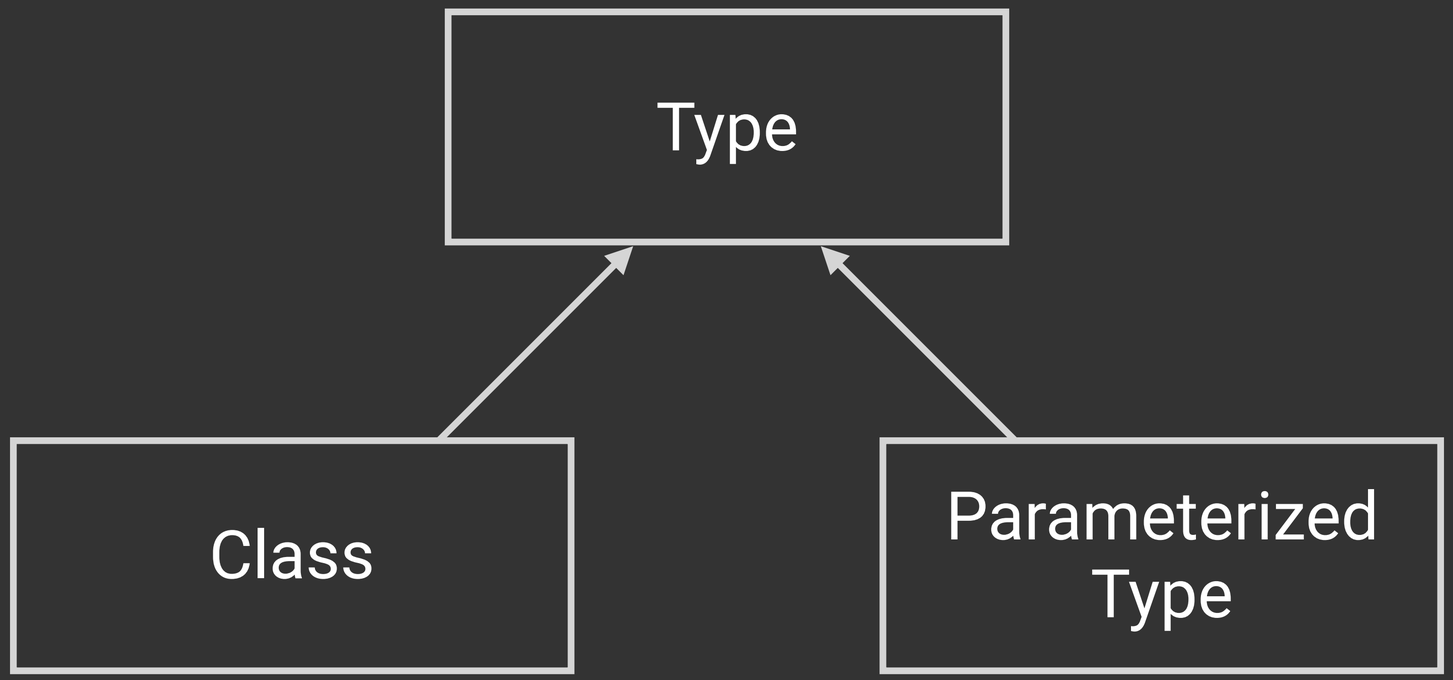
```kotlin
class Foo (
    val bar: Nothing,
    val listBar: List<Nothing>
)
```

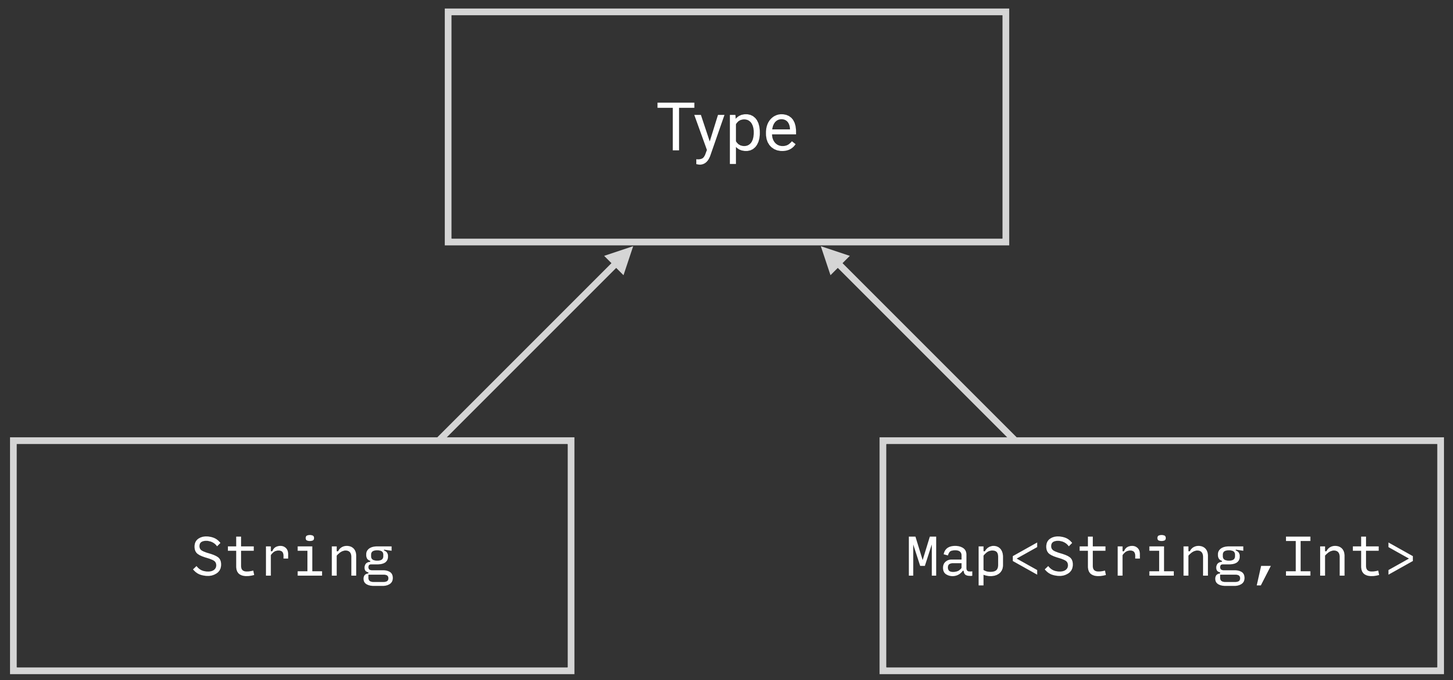
```java
Void bar = foo.getBar();
List listBar = foo.getListBar();
```

KotlinPoet

Type


Type
Class


Type
Class
Parameterized Type


Type
String
Map<String,Int>

```
val stringClass: KClass<String> = String::class
```

```
val stringClass: KClass<String> = String::class
val stringName: ClassName = stringClass.asClassName()
```

```kotlin
val stringClass: KClass<String> = String::class
val stringName: ClassName = stringClass.asClassName()
```

```kotlin
import kotlin.String

val foo: String = ""
```

```kotlin
val stringClass: KClass<String> = String::class
val stringName: ClassName = stringClass.asClassName()
```

```kotlin
import kotlin.String

val foo: String = ""

class String
```

```kotlin
val stringClass: KClass<String> = String::class
val stringName: ClassName = stringClass.asClassName()
```

```kotlin
class String {
  val foo: kotlin.String = ""
}
```

```
val stringClass: KClass<String> = String::class
val stringName: ClassName = stringClass.asClassName()
```

```
val stringClass: KClass<String> = String::class
val stringName: ClassName = stringClass.asClassName()

val mapStringToInt = ParameterizedTypeName.get(
    Map::class.asTypeName(), stringName, Int::class.asTypeName())
```

```
val stringClass: KClass<String> = String::class
val stringName: ClassName = stringClass.asClassName()

val mapStringToInt = ParameterizedTypeName.get(
    Map::class.asTypeName(), stringName, Int::class.asTypeName())

val nullableMap = mapStringToInt.asNullable()
```

```
val foo = PropertySpec.builder("foo", String::class)
    .initializer("%S", "foo")
    .build()
```

```
val foo = PropertySpec.builder("foo", String::class)
    .initializer("%S", "foo")
    .build()
```

```
val foo = PropertySpec.builder("foo", String::class)
    .initializer("%S", "foo")
    .build()
```

```
val foo = PropertySpec.builder("foo", String::class)
    .initializer("%S", "foo")
    .build()
```

```
val foo = PropertySpec.builder("foo", String::class)
    .initializer("%S", "foo")
    .build()
println(foo)
```

```
val foo: kotlin.String = "foo"
```

```
val foo = PropertySpec.builder("foo", String::class)
    .initializer("%S", "foo\tbar")
    .build()
println(foo)
```

```
val foo: kotlin.String = "foo\tbar"
```

```
val foo = PropertySpec.builder("foo", String::class)
    .initializer("%S", "foo\tbar")
    .build()
println(foo)
```

```
val foo: kotlin.String = "foo\tbar"
```

```
val foo = PropertySpec.builder("foo", String::class)
    .initializer("%S", "foo\tbar")
    .build()

val file = FileSpec.builder("com.example", "KotlinConf.kt")
    .addProperty(foo)
    .build()
println(file)
```

```
package com.example

import kotlin.String

val foo: String = "foo\tbar"
```

```
PropertySpec.builder("foo", String::class)
    .initializer("%S", "foo\tbar")
    .build()
```

```
package com.example

import kotlin.String

val foo: String = "foo\tbar"
```

```
FunSpec.builder("main")
    .addParameter("args", String::class, VARARG)
    .build()
```

```
package com.example

import kotlin.String

fun main(vararg args: String) {
}
```

```
TypeSpec.classBuilder("User")
    .addModifiers(DATA)
    .build()
```

```
package com.example

data class User
```

```
TypeSpec.classBuilder("User")
    .addModifiers(DATA)
    .build()
```

```
package com.example

data class User
```

```
TypeSpec.classBuilder("User")
    .addModifiers(DATA)
    .addProperty(PropertySpec.builder("name", String::class).build())
    .build()
```

```
package com.example

data class User {
  val name: String
}
```

```
TypeSpec.classBuilder("User")
    .addModifiers(DATA)
    .addProperty(PropertySpec.builder("name", String::class).build())
    .primaryConstructor(FunSpec.constructorBuilder()
        .addParameter("name", String::class)
        .build())
    .build()
```

```
package com.example

data class User(name: String) {
  val name: String
}
```

```
TypeSpec.classBuilder("User")
    .addModifiers(DATA)
    .addProperty(PropertySpec.builder("name", String::class)
        .initializer("name")
        .build())
    .primaryConstructor(FunSpec.constructorBuilder()
        .addParameter("name", String::class)
        .build())
    .build()
```

```
package com.example

data class User(val name: String)
```

```
TypeSpec.classBuilder("User")
    .addModifiers(DATA)
    .addProperty(PropertySpec.builder("name", String::class)
        .initializer("name.toLowerCase()")
        .build())
    .primaryConstructor(FunSpec.constructorBuilder()
        .addParameter("name", String::class)
        .build())
    .build()
```

```
package com.example

data class User(name: String) {
  val name: String = name.toLowerCase()
}
```

```
FunSpec.builder("seconds")
    .receiver(Long::class)
    .returns(Duration::class)
    .addStatement("return %T.ofSeconds(this)", Duration::class)
    .build()
```

```
package com.example

import java.time.Duration
import kotlin.Long

fun Long.seconds(): Duration = Duration.ofSeconds(this)
```

```
FunSpec.builder("seconds")
    .receiver(Long::class)
    .returns(Duration::class)
    .addStatement("require(this >= 0L)")
    .addStatement("return %T.ofSeconds(this)", Duration::class)
    .build()
```

```
package com.example

import java.time.Duration
import kotlin.Long

fun Long.seconds(): Duration {
  require(this >= 0L)
  return Duration.ofSeconds(this)
}
```

```
FunSpec.builder("seconds")
    .receiver(Long::class)
    .returns(Duration::class)
    .addStatement("require(this >= 0L)")
    .addStatement("return %T.ofSeconds(this)", Duration::class)
    .build()
```

```
val longSeconds = FunSpec.builder("seconds")
    .receiver(Long::class)
    .returns(Duration::class)
    .addStatement("require(this >= 0L)")
    .addStatement("return %T.ofSeconds(this)", Duration::class)
    .build()
```

```
val longSeconds = FunSpec.builder("seconds")
    .receiver(Long::class)
    .returns(Duration::class)
    .addStatement("require(this >= 0L)")
    .addStatement("return %T.ofSeconds(this)", Duration::class)
    .build()

FunSpec.builder("main")
    .addParameter("args", String::class, VARARG)
    .addStatement("println(%L.%N())", 2L, longSeconds)
    .build()
```

```
val longSeconds = FunSpec.builder("seconds")
    .receiver(Long::class)
    .returns(Duration::class)
    .addStatement("require(this >= 0L)")
    .addStatement("return %T.ofSeconds(this)", Duration::class)
    .build()

FunSpec.builder("main")
    .addParameter("args", String::class, VARARG)
    .addStatement("println(%L.%N())", 2L, longSeconds)
    .build()
```

```
val longSeconds = FunSpec.builder("seconds")
    .receiver(Long::class)
    .returns(Duration::class)
    .addStatement("require(this >= 0L)")
    .addStatement("return %T.ofSeconds(this)", Duration::class)
    .build()

FunSpec.builder("main")
    .addParameter("args", String::class, VARARG)
    .addStatement("println(%L.%N())", 2L, longSeconds)
    .build()
```

```
fun main(vararg args: String) {
  println(2L.seconds())
}
```

```
val code = CodeBlock.builder()
```

```
val code = CodeBlock.builder()

code.addStatement("val foo = %T.MIN_VALUE", Int::class)
```

```
val code = CodeBlock.builder()

code.addStatement("val foo = %T.MIN_VALUE", Int::class)

code.add("val bar = ")
when (answer) {
  YES -> code.add("0L")
  NO -> code.add("%T.MIN_VALUE", Int::class)
}
code.add(".toString()\n")
```

```
val code = CodeBlock.builder()

code.addStatement("val foo = %T.MIN_VALUE", Int::class)

code.add("val bar = ")
when (answer) {
  YES -> code.add("0L")
  NO -> code.add("%T.MIN_VALUE", Int::class)
}
code.add(".toString()\n")

code.beginControlFlow("if (bar.isEmpty())")
    .addStatement("println(%S)", "Empty!")
    .nextControlFlow("else")
    .addStatement("println(bar)")
    .endControlFlow()
```

```
val intMin = CodeBlock.of("%T.MIN_VALUE", Int::class)
val intMax = CodeBlock.of("%T.MAX_VALUE", Int::class)
val longMin = CodeBlock.of("%T.MIN_VALUE", Long::class)
val longMax = CodeBlock.of("%T.MAX_VALUE", Long::class)
val values = listOf(intMin, intMax, longMin, longMax)
```

```
val intMin = CodeBlock.of("%T.MIN_VALUE", Int::class)
val intMax = CodeBlock.of("%T.MAX_VALUE", Int::class)
val longMin = CodeBlock.of("%T.MIN_VALUE", Long::class)
val longMax = CodeBlock.of("%T.MAX_VALUE", Long::class)
val values = listOf(intMin, intMax, longMin, longMax)

// elsewhere
val list = values.joinToCode(prefix = "listOf(", suffix = ")")
```

```
val intMin = CodeBlock.of("%T.MIN_VALUE", Int::class)
val intMax = CodeBlock.of("%T.MAX_VALUE", Int::class)
val longMin = CodeBlock.of("%T.MIN_VALUE", Long::class)
val longMax = CodeBlock.of("%T.MAX_VALUE", Long::class)
val values = listOf(intMin, intMax, longMin, longMax)

// elsewhere
val list = values.joinToCode(prefix = "listOf(", suffix = ")")
```

```
listOf(Int.MIN_VALUE, Int.MAX_VALUE, Long.MIN_VALUE, Long.MAX_VALUE)
```

```
inline fun <reified T, R : CharSequence>
    (String.() -> R).crazy(noinline foo: T.(R?) -> Unit)
    where T : Runnable, T : Closeable
    = 42L
```

```
val typeT = TypeVariableName("T", Runnable::class, Closeable::class)
val typeR = TypeVariableName("R", CharSequence::class)
val stringExtToR = LambdaTypeName.get(String::class.asClassName(), returnType = typeR)
val tRtoUnit = LambdaTypeName.get(typeT,
        parameters = typeR.asNullable(), returnType = UNIT)
val crazy = FunSpec.builder("crazy")
    .addModifiers(INLINE)
    .receiver(stringExtToR)
    .addTypeVariable(typeT.reified())
    .addTypeVariable(typeR)
    .addParameter("foo", tRtoUnit, NOINLINE)
    .addStatement("return 42L")
    .build()
```

```
inline fun <reified T, R : CharSequence>
    (String.() -> R).crazy(noinline foo: T.(R?) -> Unit)
    where T : Runnable, T : Closeable
    = 42L
```

Kotlin Compatible Code

```java
@Nullable public String getSwamp() {
  return swamp;
}
```

```java
@Nullable public String getSwamp() {
  return swamp;
}
```

```kotlin
val swamp: String? = shrek.swamp
```

Build tool

Java™

Bytecode

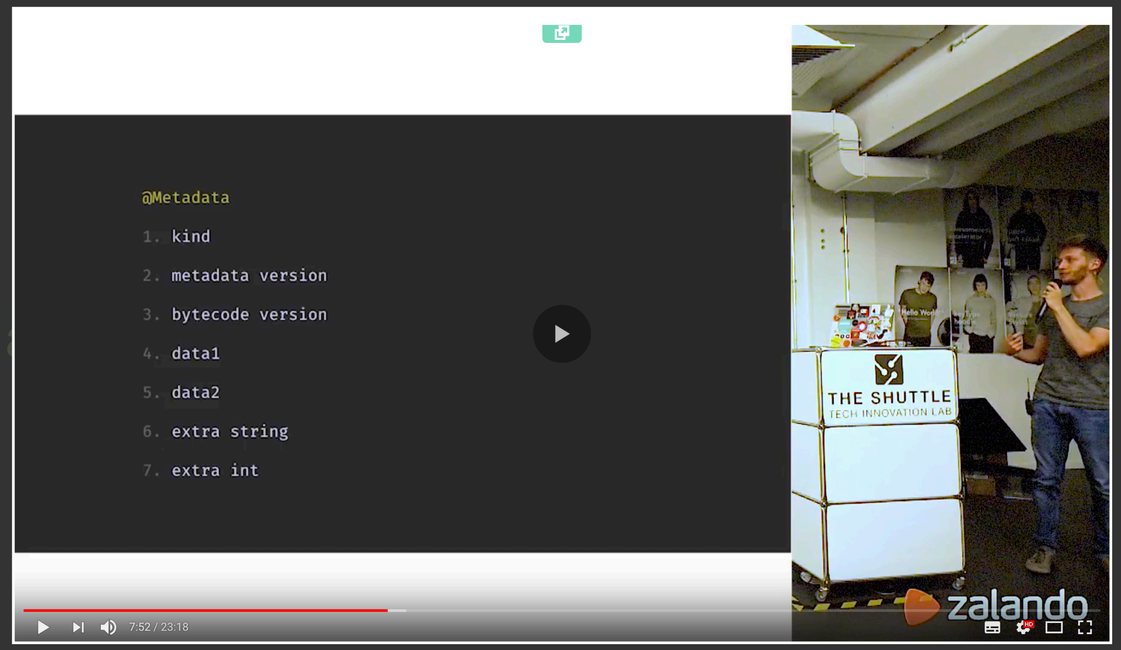
Build tool

```
interface Foo {
  val bar: String
}
```

```
public abstract interface com/sample/Foo {


  // access flags 0x401
  public abstract getBar()Ljava/lang/String;
  @Lorg/jetbrains/annotations/NotNull;() // invisible
    LOCALVARIABLE this Lcom/sample/Foo; L0 L1 0

  @Lkotlin/Metadata;(
     mv={1, 1, 7},
     bv={1, 0, 2},
     k=1,
     d1={“…”},
     d2={"Lcom/sample/Foo;", "", "bar", "", “getBar" }
  )
  // compiled from: Foo.kt
}
```

```
public abstract interface com/sample/Foo {

  // access flags 0x401
  public abstract getBar()Ljava/lang/String
  @Lorg/jetbrains/annotations/NotNull;() /
    LOCALVARIABLE this Lcom/sample/Foo; L0

  @Lkotlin/Metadata;(
     mv={1, 1, 7},
     bv={1, 0, 2},
     k=1,
     d1={"…"},
     d2={"Lcom/sample/Foo;", "", "b
  )
  // compiled from: Foo.kt
}
```

goo.gl/GcT9gz

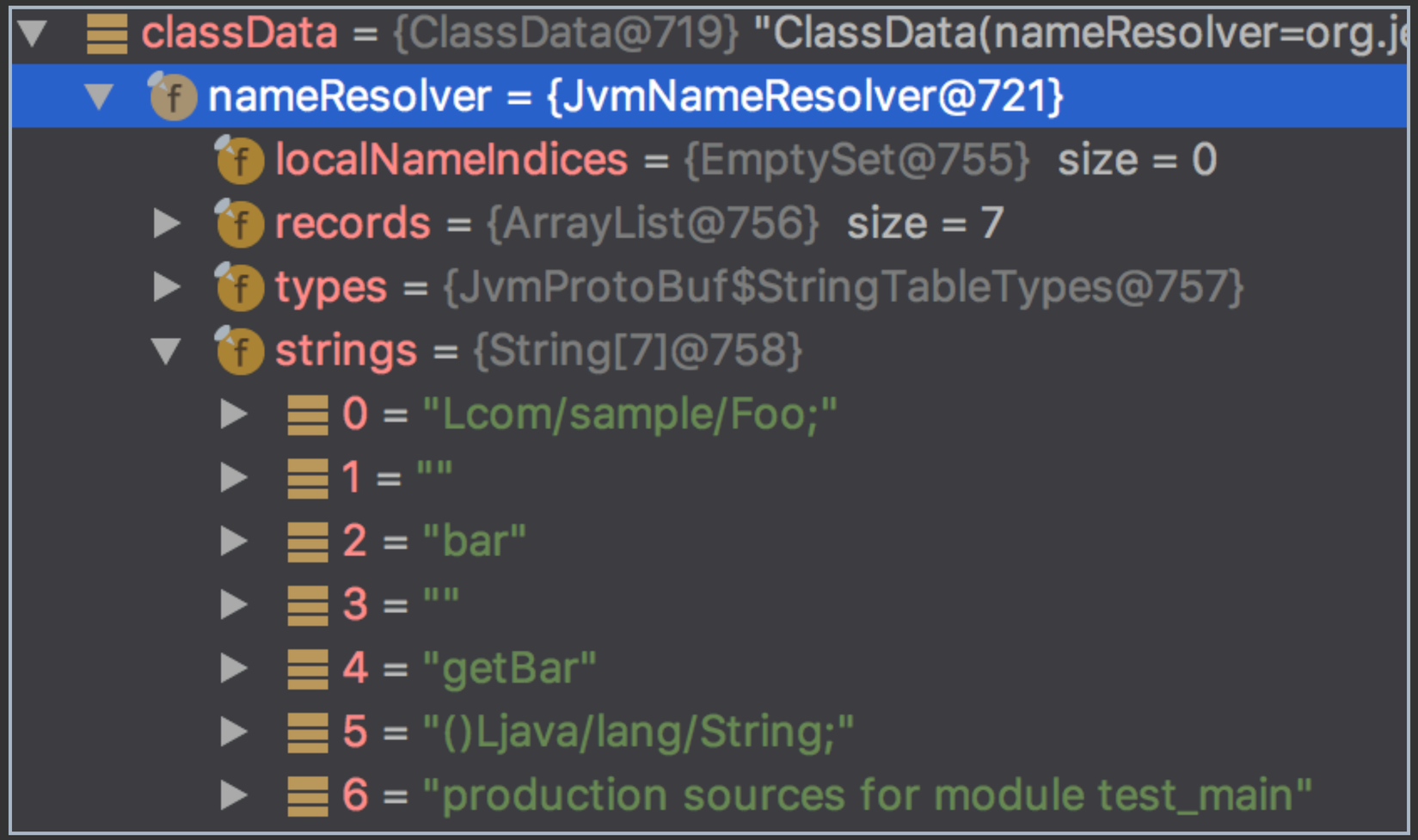


github.com/Takhion/kotlin-metadata

org.jetbrains.kotlin.serialization.ProtoBuf.Class

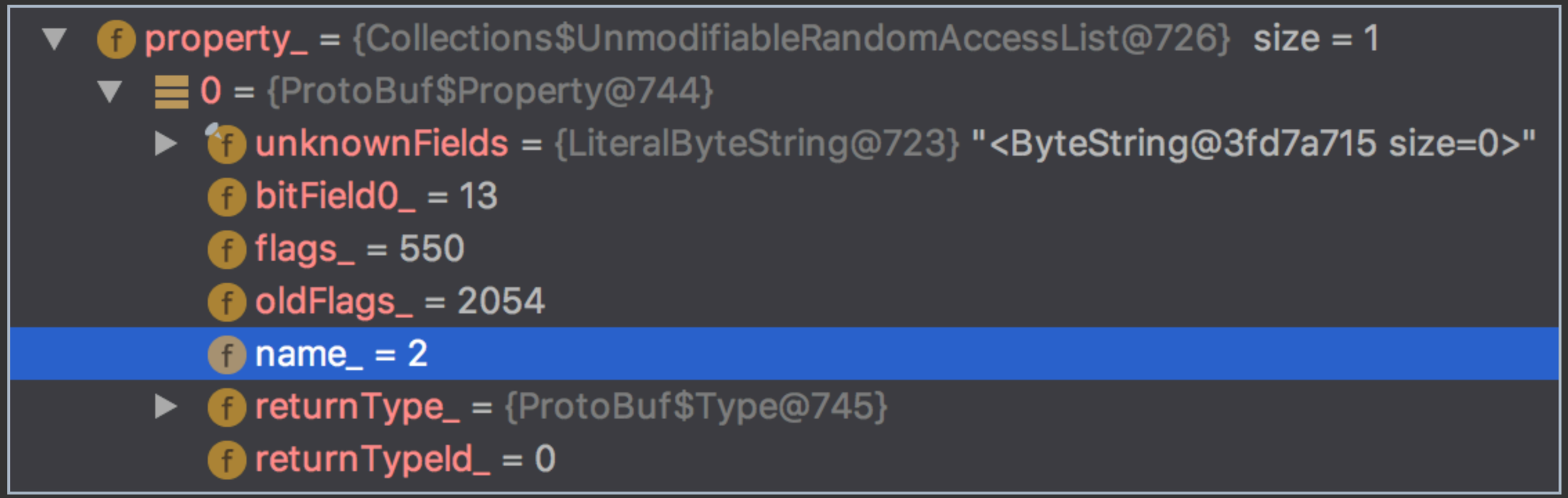
github.com/Takhion/kotlin-metadata

org.jetbrains.kotlin.serialization.ProtoBuf.Class



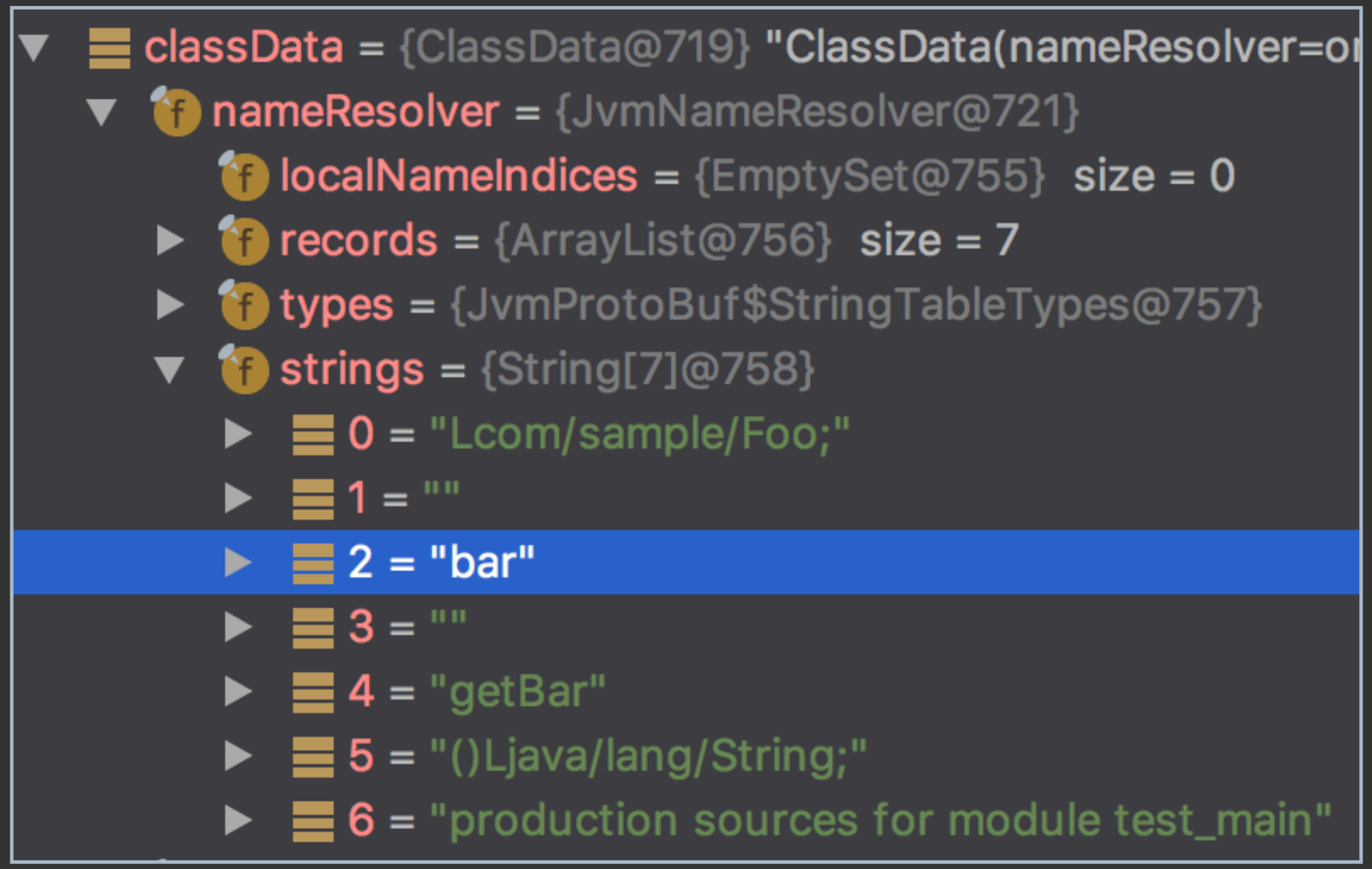
github.com/Takhion/kotlin-metadata

org.jetbrains.kotlin.serialization.ProtoBuf.Class



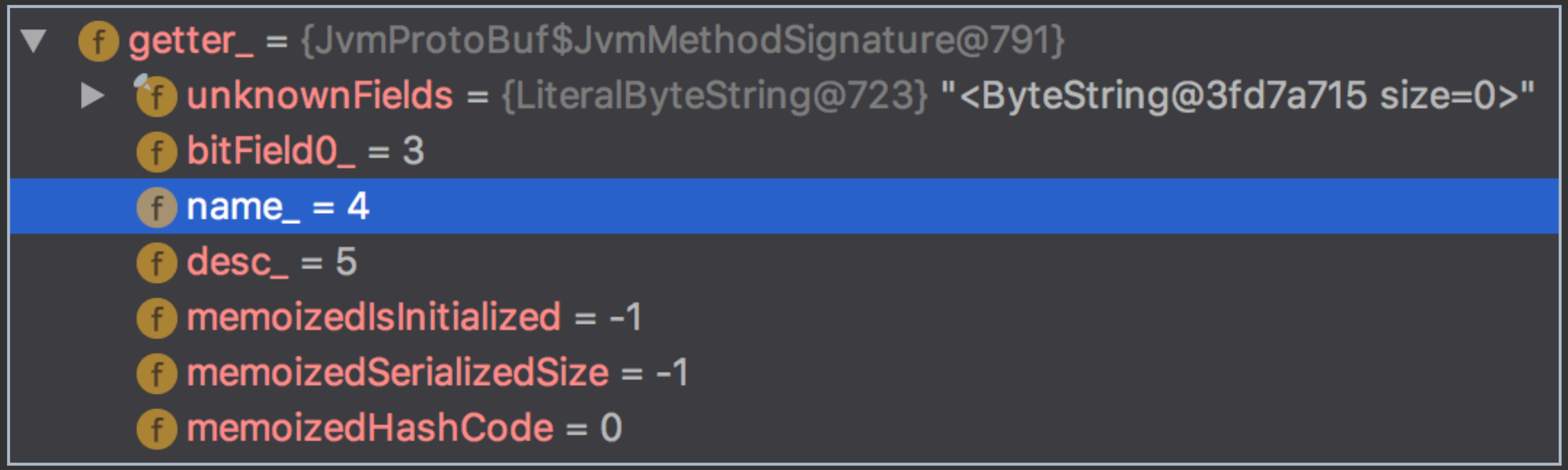
github.com/Takhion/kotlin-metadata

org.jetbrains.kotlin.serialization.ProtoBuf.Class



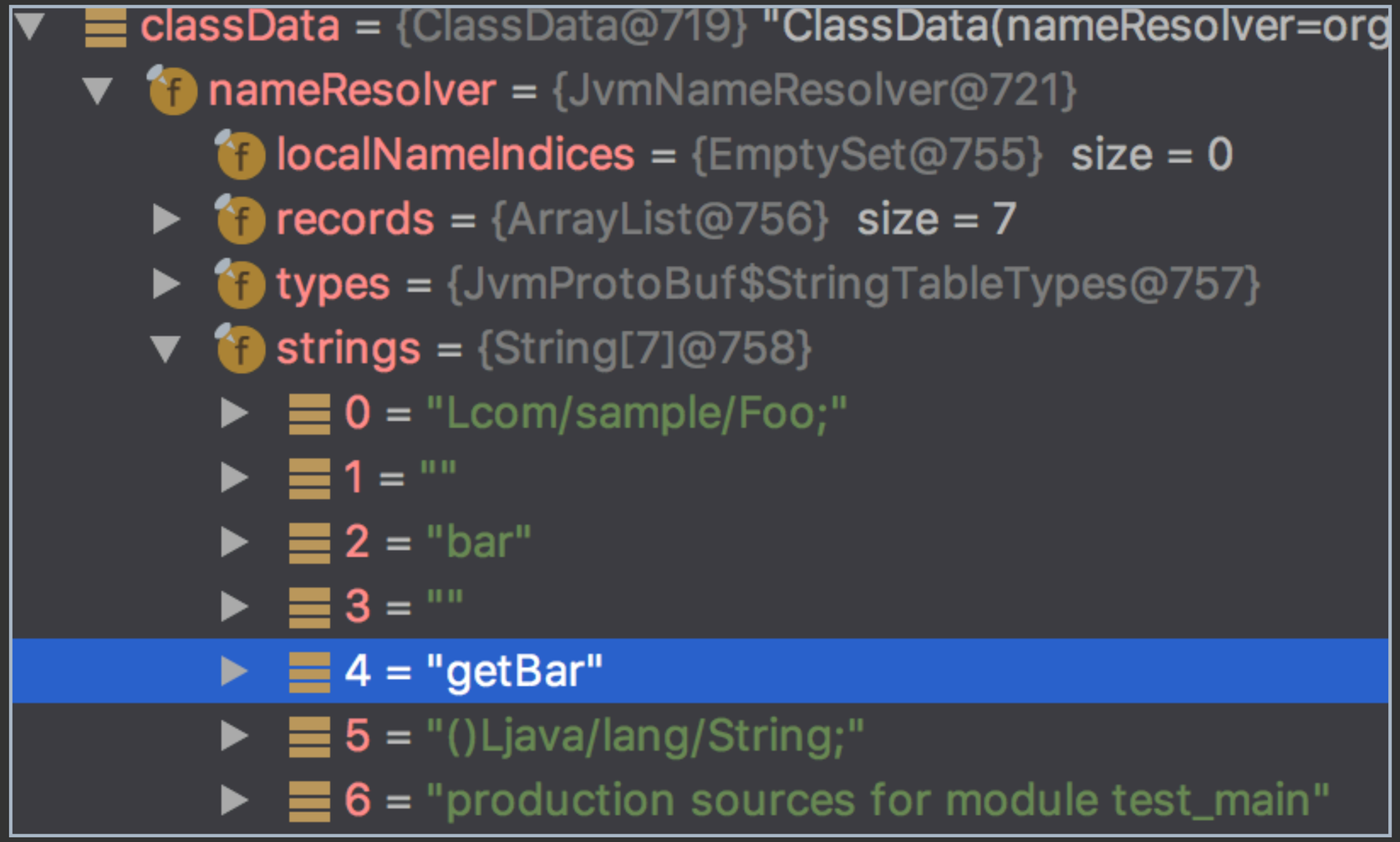
github.com/Takhion/kotlin-metadata

org.jetbrains.kotlin.serialization.ProtoBuf.Class



github.com/Takhion/kotlin-metadata

org.jetbrains.kotlin.serialization.ProtoBuf.Class



```
public abstract interface com/sample/Foo {


  // access flags 0x401
  public abstract getBar()Ljava/lang/String;
  @Lorg/jetbrains/annotations/NotNull;() // invisible
    LOCALVARIABLE this Lcom/sample/Foo; L0 L1 0

  @Lkotlin/Metadata;(
     mv={1, 1, 7},
     bv={1, 0, 2},
     k=1,
     d1={“…”},
     d2={"Lcom/sample/Foo;", "", "bar", "", “getBar" }
  )
  // compiled from: Foo.kt
}
```

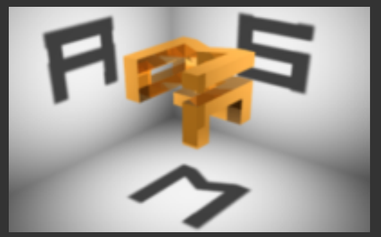

Java
Bytecode
ASM
Java


Java
Bytecode
Java

```
public abstract interface com/sample/Foo {


  // access flags 0x401
  public abstract getBar()Ljava/lang/String;
  @Lorg/jetbrains/annotations/NotNull;() // invisible
    LOCALVARIABLE this Lcom/sample/Foo; L0 L1 0

  @Lkotlin/Metadata;(
     mv={1, 1, 7},
     bv={1, 0, 2},
     k=1,
     d1={“…”},
     d2={"Lcom/sample/Foo;", "", "bar", "", “getBar" }
  )
  // compiled from: Foo.kt
}
```

```
public abstract interface com/sample/Foo {


  // access flags 0x401
  public abstract getBar()Ljava/lang/String;
  @Lorg/jetbrains/annotations/NotNull;() // invisible
    LOCALVARIABLE this Lcom/sample/Foo; L0 L1 0

  @Lkotlin/Metadata;(
     mv={1, 1, 7},
     bv={1, 0, 2},
     k=1,
     d1={"…"},
     d2={"Lcom/sample/Foo;", "", "bar", "", "getBar" }
  )
  // compiled from: Foo.kt
}
```

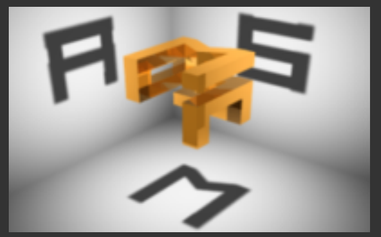
```
public abstract interface com/sample/Foo {


  // access flags 0x401
  public abstract getBar()Ljava/lang/String;
  @Lorg/jetbrains/annotations/NotNull;() // invisible
    LOCALVARIABLE this Lcom/sample/Foo; L0 L1 0

  @Lkotlin/Metadata;(
     mv={1, 1, 7},
     bv={1, 0, 2},
     k=1,
     d1={"…"},
     d2={"Lcom/sample/Foo;", "", "bar", "", "getBar" }
  )
  // compiled from: Foo.kt
}
```



```
{
  AnnotationVisitor av1 = av0.visitArray("d2");
  av1.visit(null, "Lcom/sample/Foo;");
  av1.visit(null, "");
  av1.visit(null, "bar");
  av1.visit(null, "");
  av1.visit(null, "getBar");
  av1.visit(null, "()I");
  av1.visitEnd();
}

{
  mv = cw.visitMethod(ACC_PUBLIC + ACC_ABSTRACT,
      "getBar", "()I", null, null);
  mv.visitEnd();
}
```

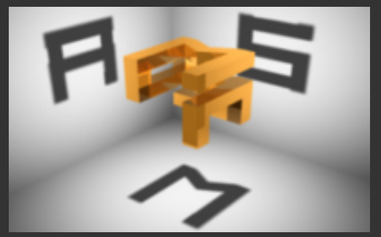
```
{
  AnnotationVisitor av1 = av0.visitArray("d2");
  av1.visit(null, "Lcom/sample/Foo;");
  av1.visit(null, "");
  av1.visit(null, "bar");
  av1.visit(null, "");
  av1.visit(null, "getBar");
  av1.visit(null, "()I");
  av1.visitEnd();
}

{
  mv = cw.visitMethod(ACC_PUBLIC + ACC_ABSTRACT,
      "getBar", "()I", null, null);
  mv.visitEnd();
}
```

```
{
  AnnotationVisitor av1 = av0.visitArray("d2");
  av1.visit(null, "Lcom/sample/Foo;");
  av1.visit(null, "");
  av1.visit(null, "bar");
  av1.visit(null, "");
  av1.visit(null, "bar");
  av1.visit(null, "()I");
  av1.visitEnd();
}

{
  mv = cw.visitMethod(ACC_PUBLIC + ACC_ABSTRACT,
      "bar", "()I", null, null);
  mv.visitEnd();
}
```



```
{
  AnnotationVisitor av1 = av0.visitArray("d2");
  av1.visit(null, "Lcom/sample/Foo;");
  av1.visit(null, "");
  av1.visit(null, "bar");
  av1.visit(null, "");
  av1.visit(null, "bar");
  av1.visit(null, "()I");
  av1.visitEnd();
}

{
  mv = cw.visitMethod(ACC_PUBLIC + ACC_ABSTRACT,
      "bar", "()I", null, null);
  mv.visitEnd();
}
```

```
public abstract interface com/sample/Foo {


  // access flags 0x401
  public abstract bar()Ljava/lang/String;
  @Lorg/jetbrains/annotations/NotNull;() // invisible
    LOCALVARIABLE this Lcom/sample/Foo; L0 L1 0

  @Lkotlin/Metadata;(
     mv={1, 1, 7},
     bv={1, 0, 2},
     k=1,
     d1={"…"},
     d2={"Lcom/sample/Foo;", "", "bar", "", "bar" }
  )
  // compiled from: Foo.kt
}
```



```
{
  AnnotationVisitor av1 = av0.visitArray("d2");
  av1.visit(null, "Lcom/sample/Foo;");
  av1.visit(null, "");
  av1.visit(null, "bar");
  av1.visit(null, "");
  av1.visit(null, "bar");
  av1.visit(null, "()I");
  av1.visitEnd();
}

{
  mv = cw.visitMethod(ACC_PUBLIC + ACC_ABSTRACT,
      "bar", "()I", null, null);
  mv.visitEnd();
}
```

```kotlin
class KotlinFoo(override val bar: Int): Foo
```

```java
class JavaFoo implements Foo {
  @Override
  public int bar() {
    return 0;
  }
}
```

```
void setPadding(int l, int t, int r, int b) {
  // …
}
```

```
void setPadding(int l, int t, int r, int b) {
  // …
}
```

```
view.setPadding(10, 0, 0, 10)
```

```java
void setPadding(int l, int t, int r, int b) {
  // …
}
```

```kotlin
view.setPadding(left = 10, bottom = 10)
```

Java™

Bytecode

Multiplatform

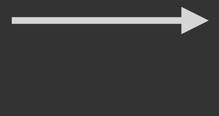
Java™

Bytecode

Java

Bytecode

JS



Java

Bytecode

LLVM

```
enum class Direction { NORTH, SOUTH, EAST, WEST }

class GamePeice(
    movement: Set<Direction> = EnumSet.of(Direction.NORTH)
)
```

```
enum class Direction { NORTH, SOUTH, EAST, WEST }

class GamePeice(
    movement: Set<Direction> = EnumSet.of(Direction.NORTH)
)
```

```
import java.util.EnumSet

enum class Direction { NORTH, SOUTH, EAST, WEST }

class GamePeice(
    movement: Set<Direction> = EnumSet.of(Direction.NORTH)
)
```

```
enum class Direction { NORTH, SOUTH, EAST, WEST }

class GamePeice(
    movement: Set<Direction> = setOf(Direction.NORTH)
)
```

```
import com.google.common.collect

@AutoImplement
interface Party {
  val people: ImmutableList<Person>
}
```

```
import com.google.common.collect

@AutoImplement
interface Party {
  val people: ImmutableList<Person>
}
```

```
import com.google.common.collect

@Generated
class AutoParty(override val people: ImmutableList<Person>) : Party
```

```
CREATE TABLE user(
  name TEXT NOT NULL,
  location TEXT AS android.location.Location
)
```

```
CREATE TABLE user(
  name TEXT NOT NULL,
  location TEXT AS android.location.Location
)
```

```
import android.location.Location

data class User(
  val name: String,
  val location: Location
)
```

```
@file:JvmName("ByteStrings")

@JvmName("from")
fun ByteArray.asByteString(): ByteString = …
```

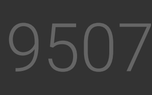
```kotlin
@file:JvmName("ByteStrings")

@JvmName("from")
fun ByteArray.asByteString(): ByteString = …
```

```java
byte[] bytes = …
ByteString b = ByteStrings.from(bytes);
```

```kotlin
@file:JvmName("ByteStrings")

@JvmName("from")
fun ByteArray.asByteString(): ByteString = …
```

```java
byte[] bytes = …
ByteString b = ByteStrings.from(bytes);
```

```
Cannot access 'JvmName': it is internal in 'kotlin.jvm'
```



KT-19507

```
@JsName("from")
fun ByteArray.asByteString(): ByteString = …
```

```kotlin
@JsName("from")
fun ByteArray.asByteString(): ByteString = …
```

KT-19508

Verbosity is okay. API is the #1 priority.

Verbosity is okay. API is the #1 priority.

If you do any kind of Java codegen, keep Kotlin in mind.

Verbosity is okay. API is the #1 priority.

If you do any kind of Java codegen, keep Kotlin in mind.

Don’t depend on platform types unless the user does.

KotlinPoet 0.6.0 released!

Kotlin Code Generation

@Strongolopolis & @JakeWharton