A Builder's Intro to Kotlin

Jake Wharton

Kotlin?


JET
BRAINS


JET
BRAINS


AC


PS


TC


CL


PC


UP


DG


R#


WS


IJ


RM


YT


JET
BRAINS


AC


PS


TC


CL


PC


UP


DG

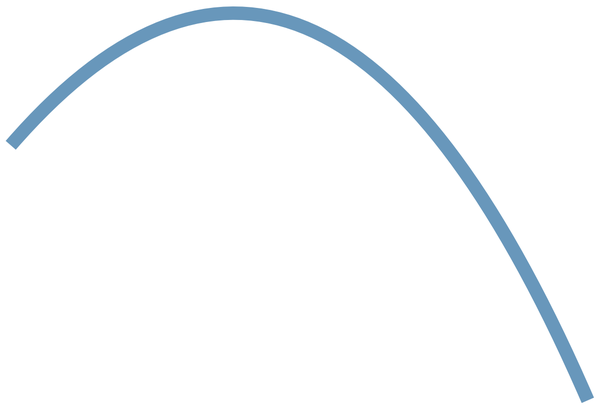

R#

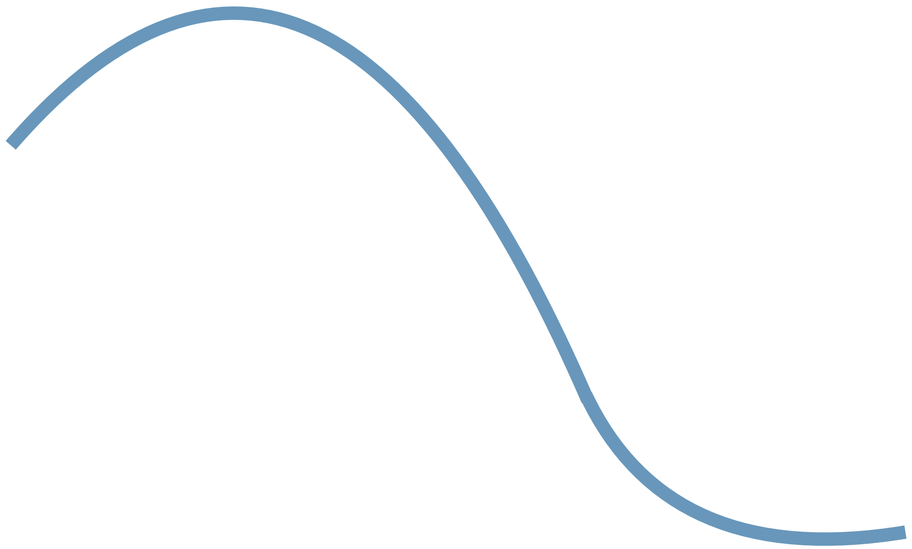

WS

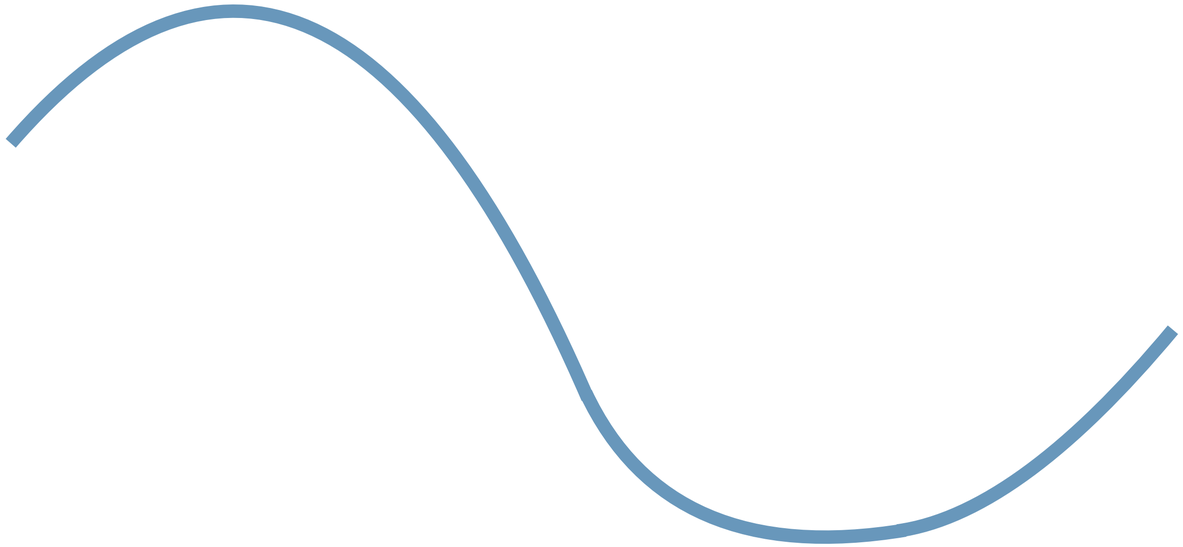

IJ

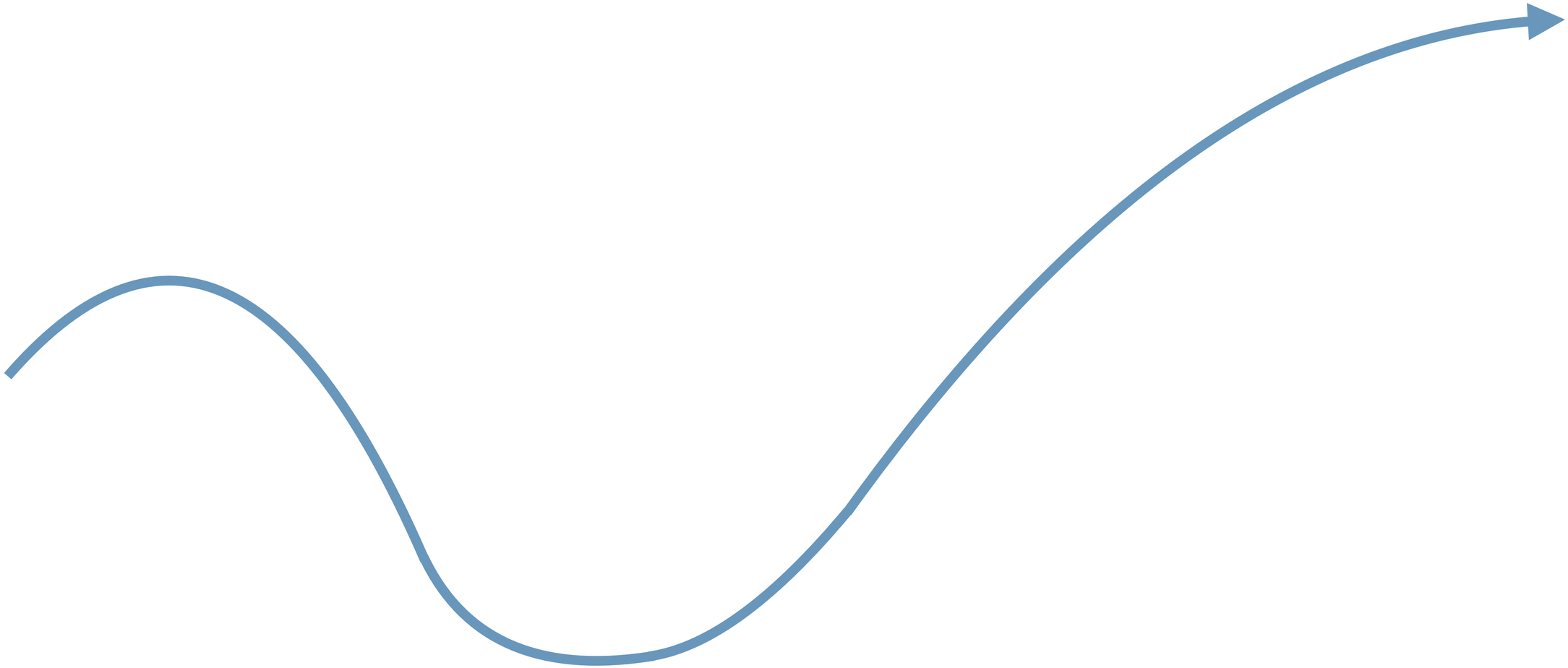

RM

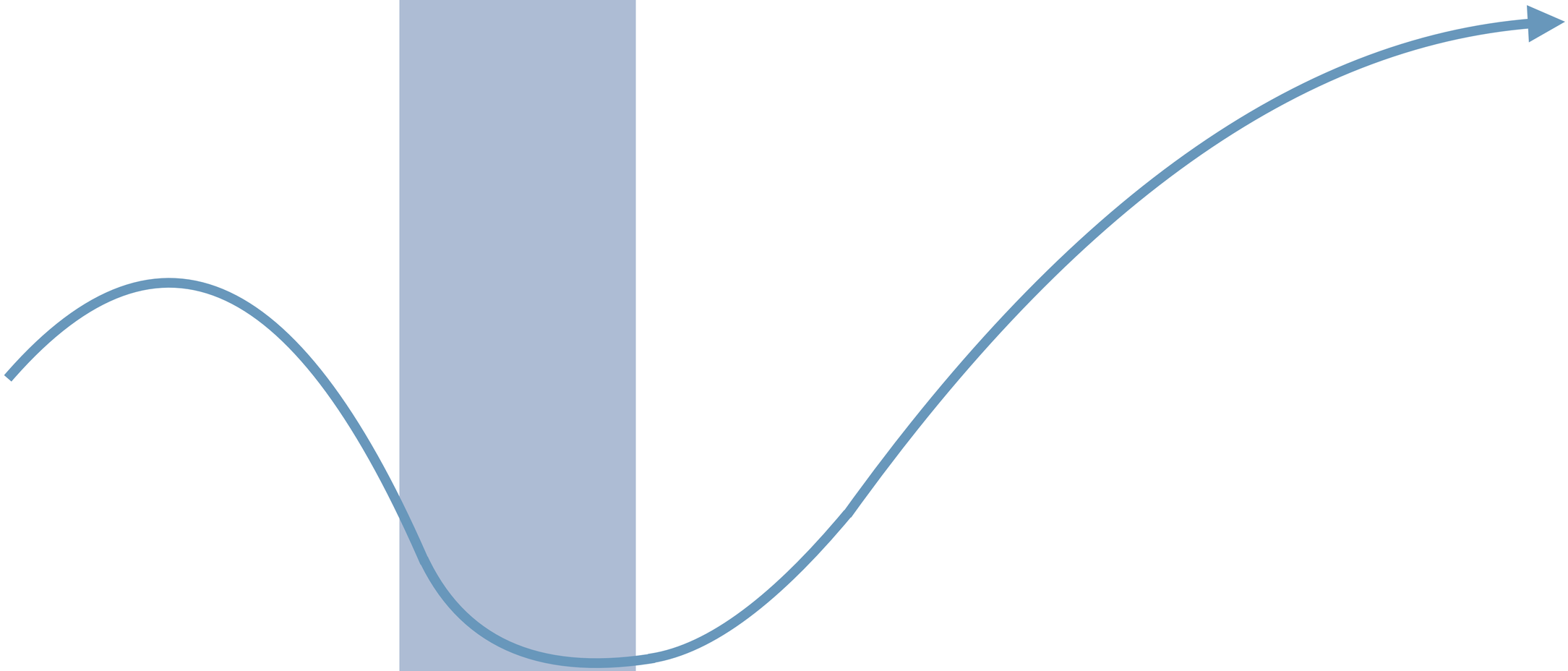

YT

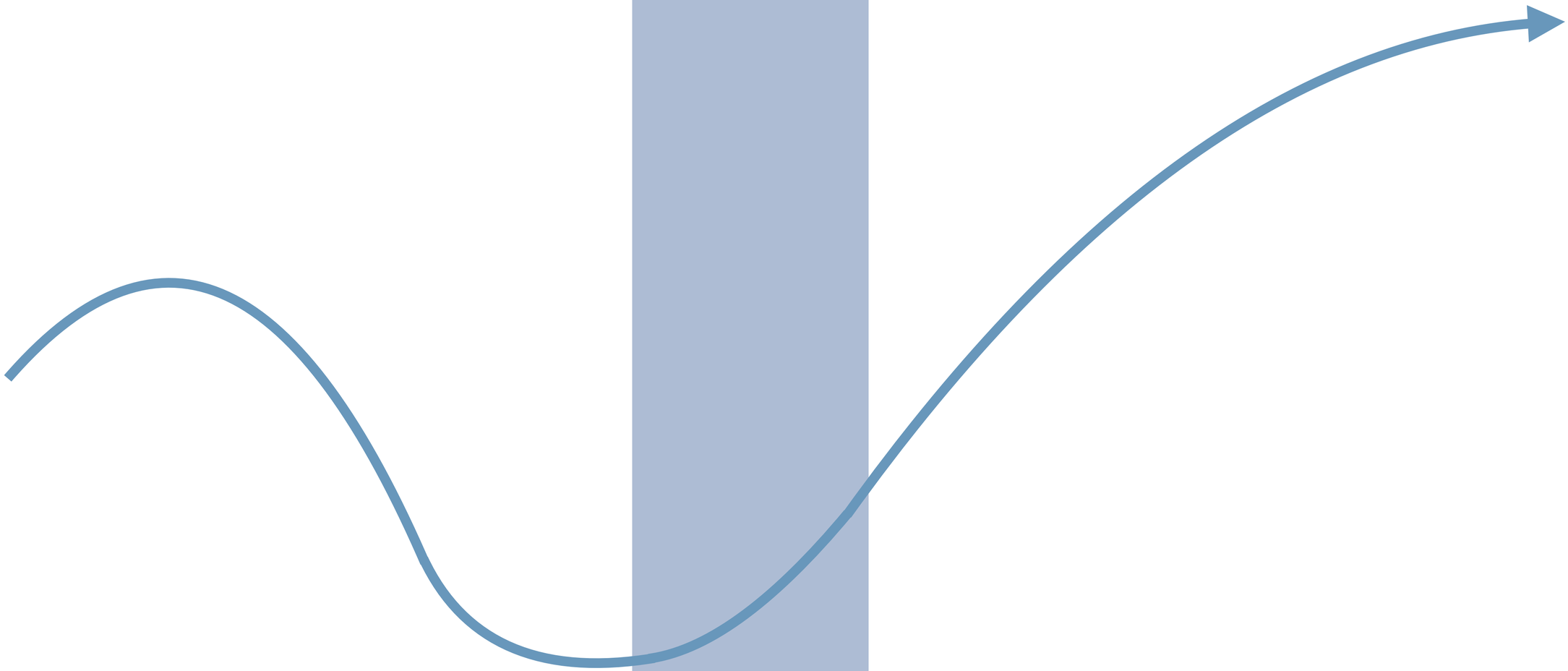













JET
BRAINS


JET
BRAINS


AC


PS


TC


CL


PC


UP


DG


R#

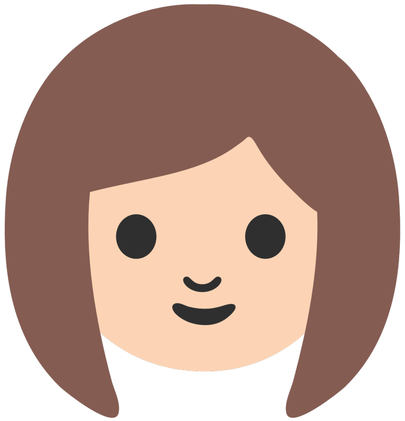

WS


IJ

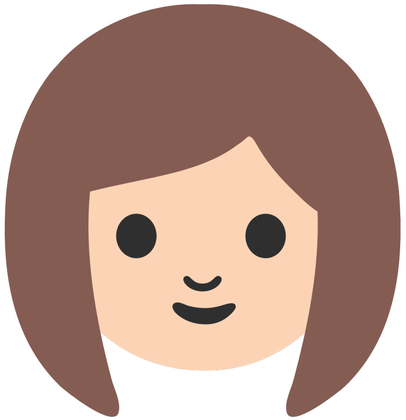

RM


YT

Why Kotlin?

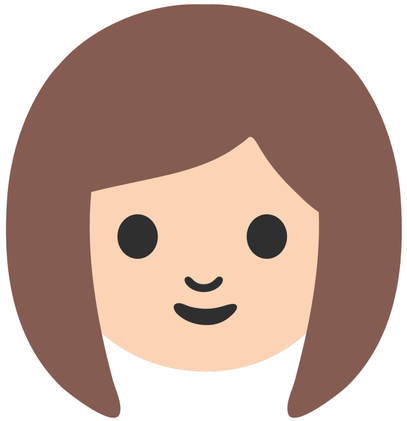


Plugin Author



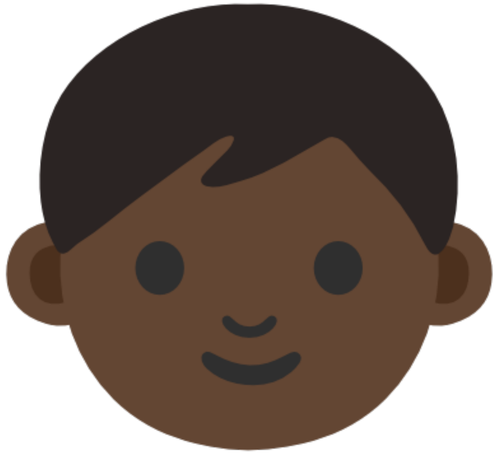
Plugin Author

Buildscript Author



Plugin Author

Buildscript Author



App Developer

```
val firstName: String = "Jake"
val lastName: String? = null
```

```kotlin
val firstName: String = "Jake"
val lastName: String? = null
```

```
val firstName: String = "Jake"
val lastName: String? = null
```

```
val firstName: String = "Jake"
val lastName: String? = null
```

```
val firstName: String = "Jake"
val lastName: String? = null
```

```
val firstName = "Jake"
val lastName: String? = null
```

```java
class User {
  public String getName() {
    // ...
  }
  public void setName(String name) {
    // ...
  }
}

// ^^^ Java
```

```
class User {
  public String getName() {
    // ...
  }
  public void setName(String name) {
    // ...
  }
}

// ^^^ Java   vvv Kotlin

val user = User()
println("Name is " + user.name)
```

```
class User {
  public String getName() {
    // ...
  }
  public void setName(String name) {
    // ...
  }
}

// ^^^ Java   vvv Kotlin

val user = User()
println("Name is " + user.name)
```

```
class User {
  public String getName() {
    // ...
  }
  public void setName(String name) {
    // ...
  }
}

// ^^^ Java   vvv Kotlin

val user = User()
println("Name is ${user.name}")
```

```
class User {
  public String getName() {
    // ...
  }
  public void setName(String name) {
    // ...
  }
}

// ^^^ Java   vvv Kotlin

val user = User()
println("Name is $user")
```

```
class User {
  public String getName() {
    // ...
  }
  public void setName(String name) {
    // ...
  }
}

// ^^^ Java   vvv Kotlin

val user = User()
println("Name is $user")
```

```
class User {
  var name = "Jake"
}

// ^^^ Kotlin
```

```
class User {
  var name = "Jake"
}

// ^^^ Kotlin   vvv Java

User user = new User();
System.out.println("Name is " + user.getName());
```

```
class User {
  var name = "Jake"
}

// ^^^ Kotlin   vvv Java

User user = new User();
System.out.println("Name is " + user.getName());
```

```
class User {
  var name = "Jake"
}

// ^^^ Kotlin   vvv Java

User user = new User();
System.out.println("Name is " + user.getName());
user.setName("Jane");
```

```
class User {
  var name = "Jake"
}

// ^^^ Kotlin   vvv Java

User user = new User();
System.out.println("Name is " + user.getName());
user.setName("Jane");
```

```
val user = User()
```

```
val user = User()
user = User()
```

```
val user = User()
user = User()

var currentUser = User()
currentUser = User()
```

```
fun Date.isTuesday(): Boolean {
  return day == 2
}
```

```
fun Date.isTuesday(): Boolean {
  return day == 2
}

val epoch = Date(1970, 0, 0)
if (epoch.isTuesday()) {
  println("The epoch was a Tuesday.")
} else {
  println("The epoch was not a Tuesday.")
}
```

```
fun Date.isTuesday(): Boolean {
  return day == 2
}

val epoch = Date(1970, 0, 0)
if (epoch.isTuesday()) {
  println("The epoch was a Tuesday.")
} else {
  println("The epoch was not a Tuesday.")
}
```

```kotlin
fun Date.isTuesday(): Boolean {
  return day == 2
}

val epoch = Date(1970, 0, 0)
if (epoch.isTuesday()) {
  println("The epoch was a Tuesday.")
} else {
  println("The epoch was not a Tuesday.")
}

// ^^^ Kotlin   vvv Java

DateKt.isTuesday(date)
```

```
val executor = Executors.newSingleThreadExecutor();
executor.execute { println("Background thread!") }
```

```
val executor = Executors.newSingleThreadExecutor();
val foo = Foo()
executor.execute(foo::printIt)

class Foo {
  fun printIt() {
    println("Background thread!")
  }
}
```

```
val executor = Executors.newSingleThreadExecutor();
val foo = Foo()
executor.execute(foo::printIt)

class Foo {
  fun printIt() {
    println("Background thread!")
  }
}
```

```
fun <T> List<T>.filter(predicate: (T) -> Boolean): List<T> {
  // ...
}
```

```
fun <T> List<T>.filter(predicate: (T) -> Boolean): List<T> {
  // ...
}
```

```
fun <T> List<T>.filter(predicate: (T) -> Boolean): List<T> {
  // ...
}

val items = listOf(1, 2, 3)
val odds = items.filter({ item -> item % 2 != 0 })
```

```
fun <T> List<T>.filter(predicate: (T) -> Boolean): List<T> {
  // ...
}

val items = listOf(1, 2, 3)
val odds = items.filter({ item -> item % 2 != 0 })
```

```
fun <T> List<T>.filter(predicate: (T) -> Boolean): List<T> {
  // ...
}

val items = listOf(1, 2, 3)
val odds = items.filter({ it % 2 != 0 })
```

```
fun <T> List<T>.filter(predicate: (T) -> Boolean): List<T> {
  // ...
}

val items = listOf(1, 2, 3)
val odds = items.filter() { it % 2 != 0 }
```

```
fun <T> List<T>.filter(predicate: (T) -> Boolean): List<T> {
  // ...
}

val items = listOf(1, 2, 3)
val odds = items.filter { it % 2 != 0 }
```

```
fun <T> List<T>.filter(predicate: (T) -> Boolean): List<T> {
  // ...
}

val items = listOf(1, 2, 3)
val oddList = items.filter { it % 2 != 0 }
val oddSet = items.filterTo(mutableListOf()) { it % 2 != 0 }
```

```
fun <T> List<T>.filter(predicate: (T) -> Boolean): List<T> {
  // ...
}

val items = listOf(1, 2, 3)
val odds = items.filter { it % 2 != 0 }
```

```
inline fun <T> List<T>.filter(predicate: (T) -> Boolean): List<T> {
  // ...
}

val items = listOf(1, 2, 3)
val odds = items.filter { it % 2 != 0 }
```

```
inline fun <T> List<T>.filter(predicate: (T) -> Boolean): List<T> {
  val destination = mutableListOf<T>()
  for (item in this) {
    if (predicate(item)) destination.add(item)
  }
  return destination
}

val items = listOf(1, 2, 3)
val odds = items.filter { it % 2 != 0 }
```

```
inline fun <T> List<T>.filter(predicate: (T) -> Boolean): List<T> {
  val destination = mutableListOf<T>()
  for (item in this) {
    if (predicate(item)) destination.add(item)
  }
  return destination
}

val items = listOf(1, 2, 3)
val destination = mutableListOf<Int>()
for (item in items) {
  if (item % 2 != 0) destination.add(item)
}
val odds = destination
```

```
fun <T> List<T>.filterIsInstance(c: Class<T>): List<T> {
  val destination = mutableListOf<T>()
  for (item in this) {
    if (c.isInstance(item)) destination.add(item)
  }
  return destination
}
```

```kotlin
fun <T> List<T>.filterIsInstance(c: Class<T>): List<T> {
  val destination = mutableListOf<T>()
  for (item in this) {
    if (c.isInstance(item)) destination.add(item)
  }
  return destination
}
```

```
fun <T> List<T>.filterIsInstance(): List<T> {
  val destination = mutableListOf<T>()
  for (item in this) {
    if (item in T) destination.add(item)
  }
  return destination
}
```

```
fun <T> List<T>.filterIsInstance(): List<T> {
  val destination = mutableListOf<T>()
  for (item in this) {
    if (item in T) destination.add(item)
  }
  return destination
}
```

```
inline fun <T> List<T>.filterIsInstance(): List<T> {
  val destination = mutableListOf<T>()
  for (item in this) {
    if (item in T) destination.add(item)
  }
  return destination
}
```

```
inline fun <reified T> List<T>.filterIsInstance(): List<T> {
  val destination = mutableListOf<T>()
  for (item in this) {
    if (item in T) destination.add(item)
  }
  return destination
}
```

```
inline fun <reified T> List<T>.filterIsInstance(): List<T> {
  val destination = mutableListOf<T>()
  for (item in this) {
    if (item in T) destination.add(item)
  }
  return destination
}
```

```kotlin
inline fun <reified T> List<T>.filterIsInstance(): List<T> {
  val destination = mutableListOf<T>()
  for (item in this) {
    if (item in T) destination.add(item)
  }
  return destination
}

val list = listOf(1, 2f, 3, 4f)
val ints = list.filterIsInstance<Int>()
```

```kotlin
inline fun <reified T> List<T>.filterIsInstance(): List<T> {
  val destination = mutableListOf<T>()
  for (item in this) {
    if (item in T) destination.add(item)
  }
  return destination
}

val list = listOf(1, 2f, 3, 4f)
val ints = list.filterIsInstance<Int>()
```

```
ALOAD 7 # item
INSTANCEOF java/lang/Integer
IFEQ L8
ALOAD 5 # destination
ALOAD 7 # item
INVOKEINTERFACE java/util/Collection.add (Ljava/lang/Object;)Z
```

```
class User {
  val name = "Jake"
}
```

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

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

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

```
class User(val name: String)

val jake = User("Jake")
println("Hello, $jake!")
```

```
class User(val name: String)

val jake = User("Jake")
println("Hello, $jake!")
```

```
Hello, User@3a71f4dd!
```

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

val jake = User("Jake")
println("Hello, $jake!")
```

```
Hello, User@3a71f4dd!
```

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

val jake = User("Jake")
println("Hello, $jake!")
```

```
Hello, User(name=Jake)!
```

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

val jake = User("Jake")
println("Hello, $jake!")
```

```
Hello, User(name=Jake)!
```

```
class UserPersisence(db: SqlDatabase) {
  private val deleteByName
      = db.createStatement("DELETE FROM user WHERE name = ?")

  fun delete(name: String) {
    deleteByName.bind(1, name)
    deleteByName.execute()
  }
}
```

```
class UserPersisence(db: SqlDatabase) {
  private val deleteByName
      = db.createStatement("DELETE FROM user WHERE name = ?")

  fun delete(name: String) {
    deleteByName.bind(1, name)
    deleteByName.execute()
  }
}
```

```
class UserPersisence(db: SqlDatabase) {
  private val deleteByName by lazy {
    db.createStatement("DELETE FROM user WHERE name = ?")
  }

  fun delete(name: String) {
    deleteByName.bind(1, name)
    deleteByName.execute()
  }
}
```

```
val deleteByName by lazy {
  db.createStatement("DELETE FROM user WHERE name = ?")
}
```

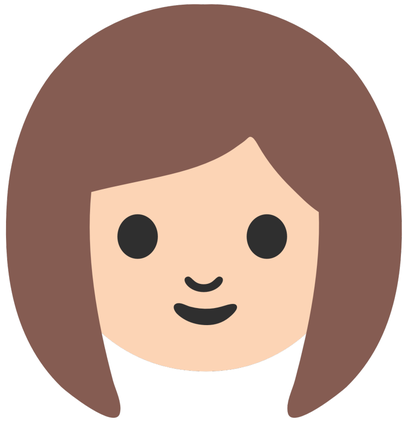
```
val deleteByName by lazy {
  db.createStatement("DELETE FROM user WHERE name = ?")
}

var name by Delegates.observable("Jane") { prop, old, new ->
  println("Name changed from $old to $new")
}
```

```
val deleteByName by lazy {
  db.createStatement("DELETE FROM user WHERE name = ?")
}

var name by Delegates.observable("Jane") { prop, old, new ->
  println("Name changed from $old to $new")
}

var address by Delegates.notNull<String>()
```

```
val deleteByName by lazy {
  db.createStatement("DELETE FROM user WHERE name = ?")
}

var name by Delegates.observable("Jane") { prop, old, new ->
  println("Name changed from $old to $new")
}

var address by Delegates.notNull<String>()

val nameView by bindView<TextView>(R.id.name)
```

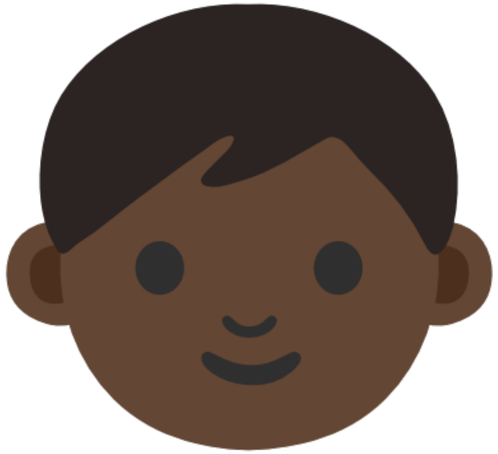
```
val deleteByName by lazy {
  db.createStatement("DELETE FROM user WHERE name = ?")
}

var name by Delegates.observable("Jane") { prop, old, new ->
  println("Name changed from $old to $new")
}

var address by Delegates.notNull<String>()

val nameView by bindView<TextView>(R.id.name)
```

```
fun main(vararg args: String) = runBlocking<Unit> {
  val jobs = List(100_000) {
    launch(CommonPool) {
      delay(1000L)
      print(".")
    }
  }
  jobs.forEach { it.join() }
}
```

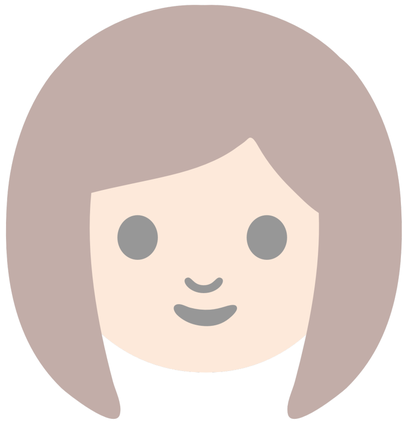


Plugin Author

Buildscript Author

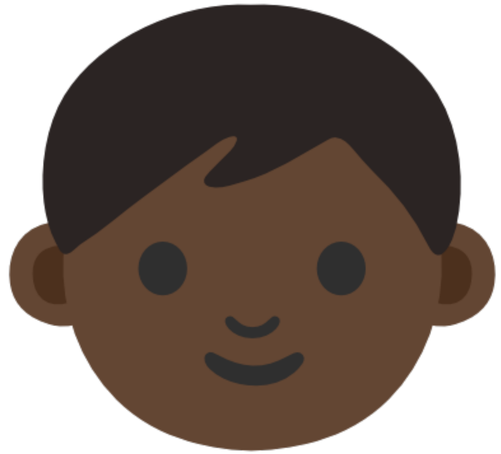


App Developer



Plugin Author

Buildscript Author



App Developer

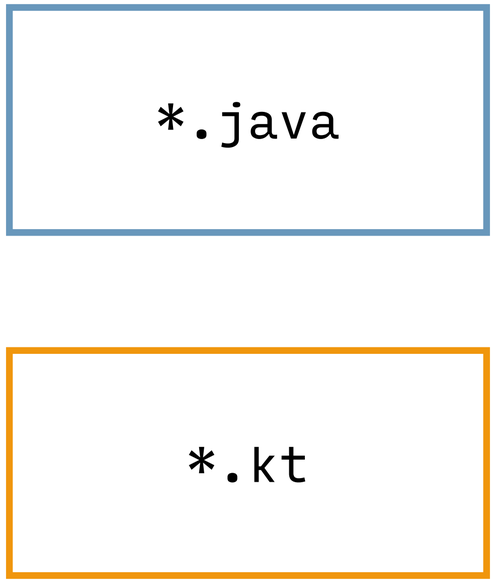
```
apply plugin: 'org.jetbrains.kotlin.jvm'
```

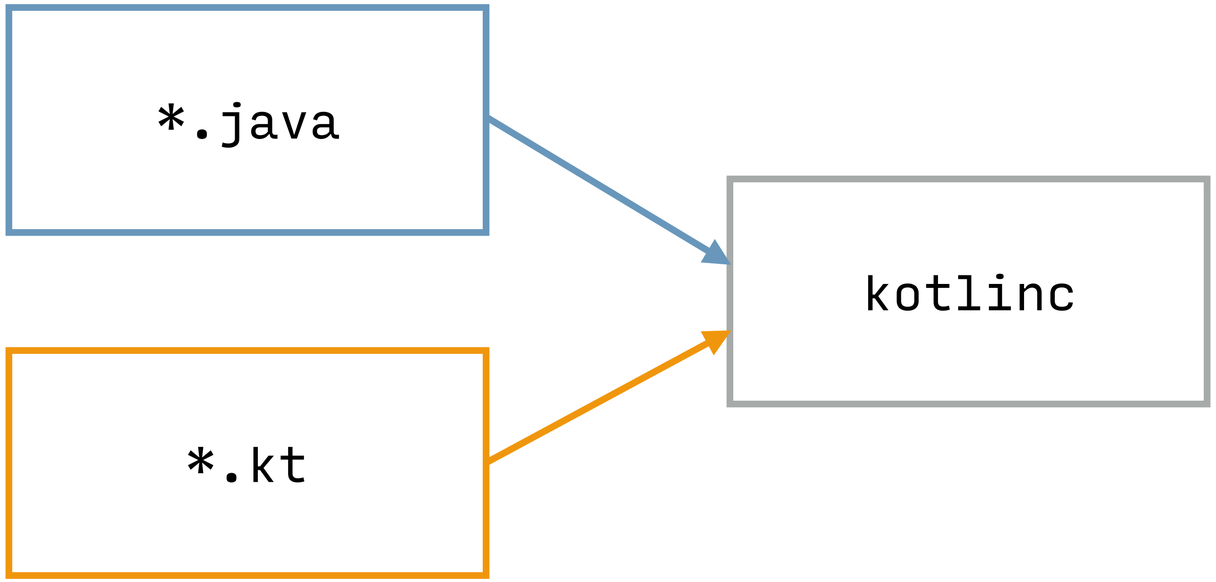
```
apply plugin: 'org.jetbrains.kotlin.jvm'
```

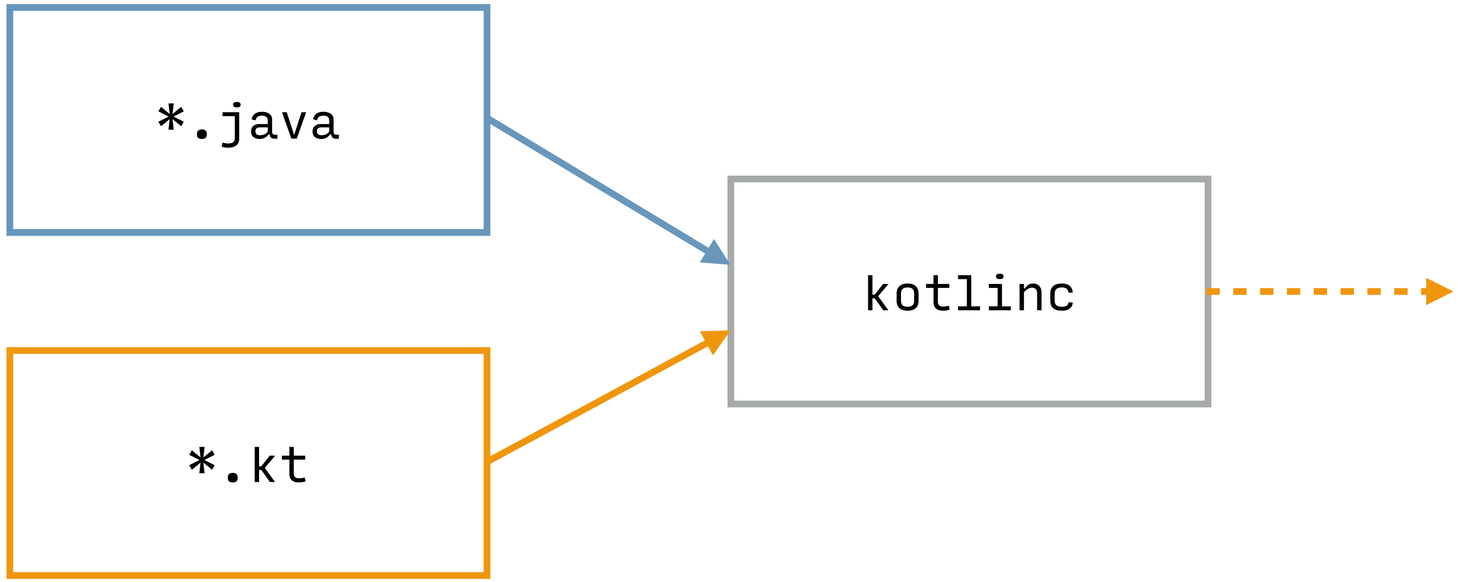
```
src/
    main/
        java/
            Fizz.java
            Buzz.java
```

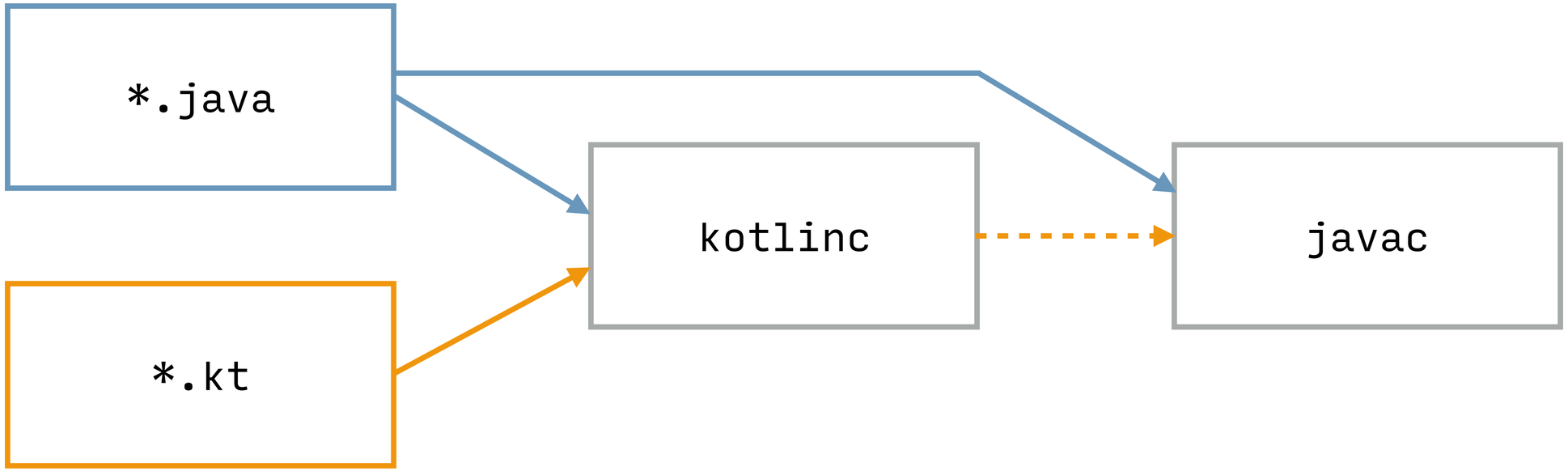
```
apply plugin: 'org.jetbrains.kotlin.jvm'
```

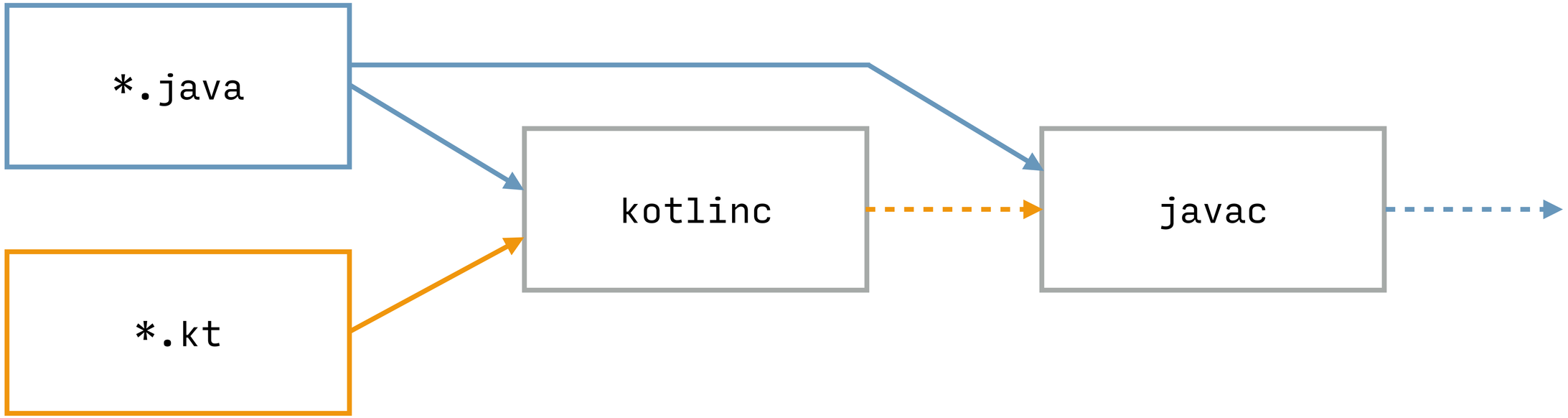
```
src/
    main/
        java/
            Fizz.java
            Buzz.kt
```

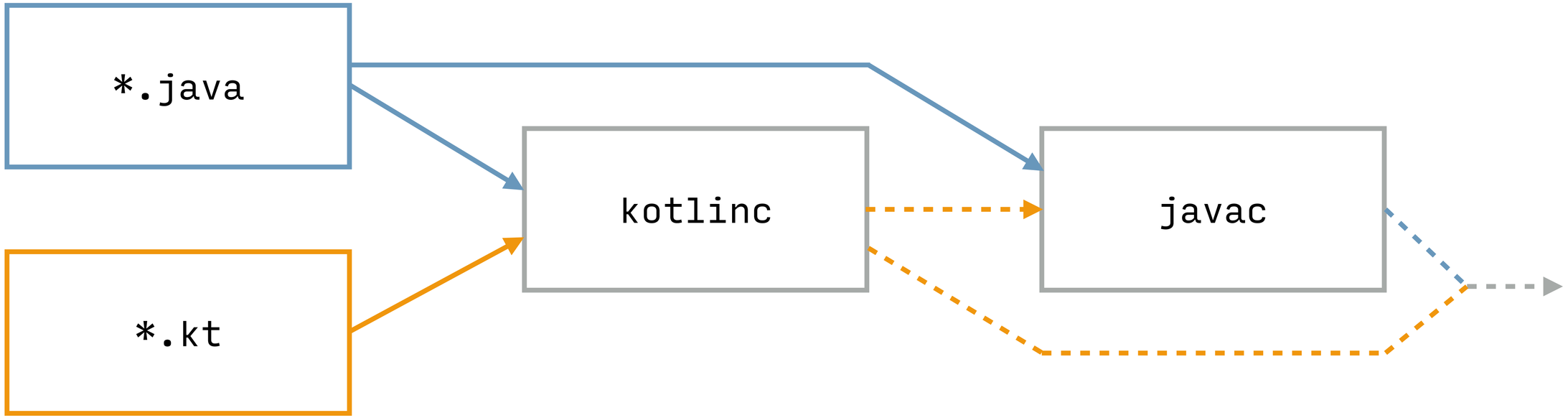

*.java
*.kt

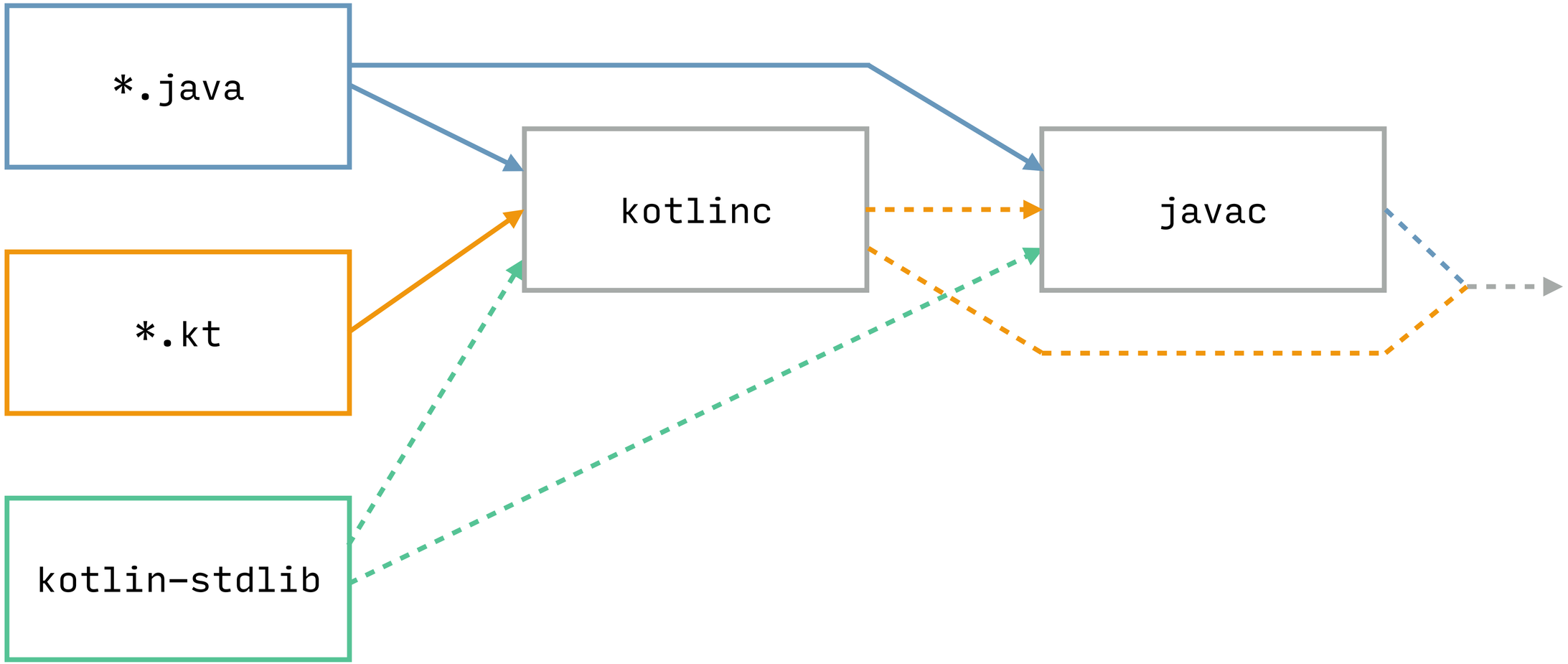

*.java
*.kt
kotlinc

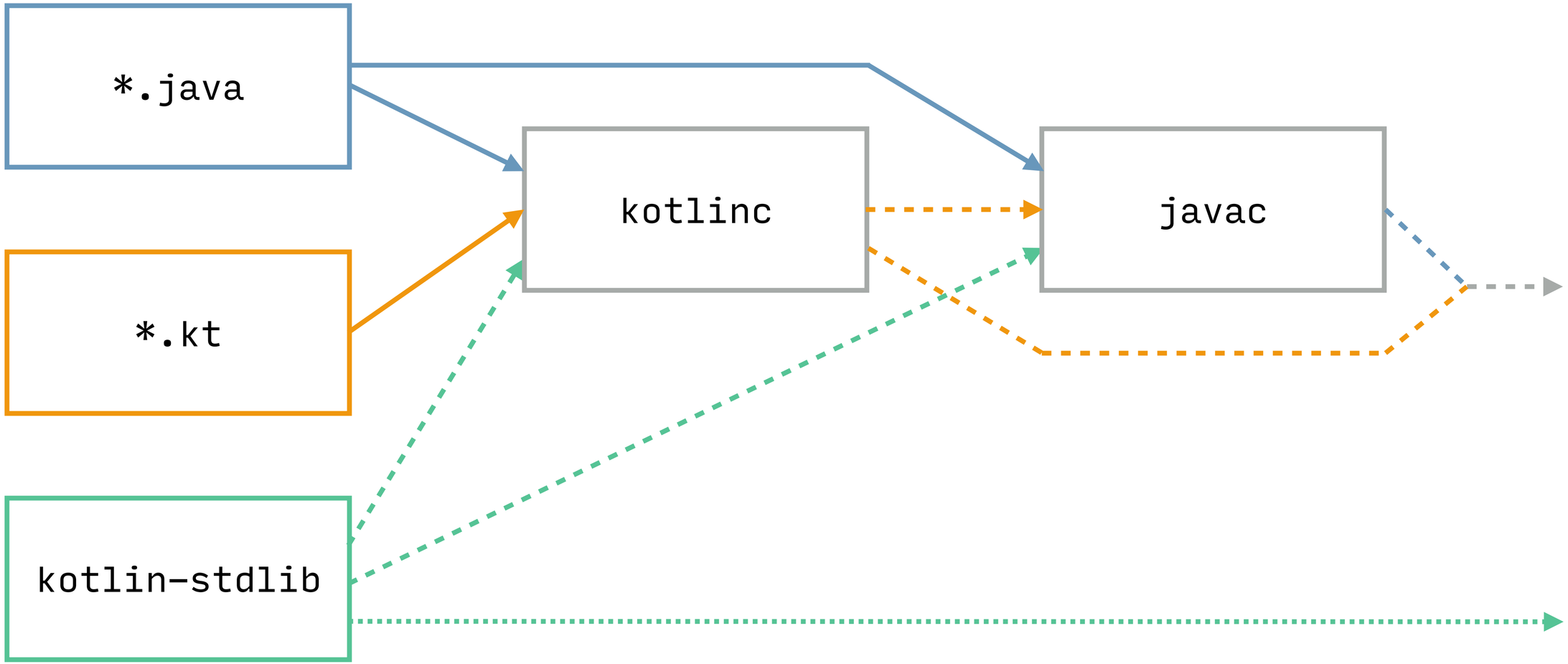

*.java
*.kt
kotlinc

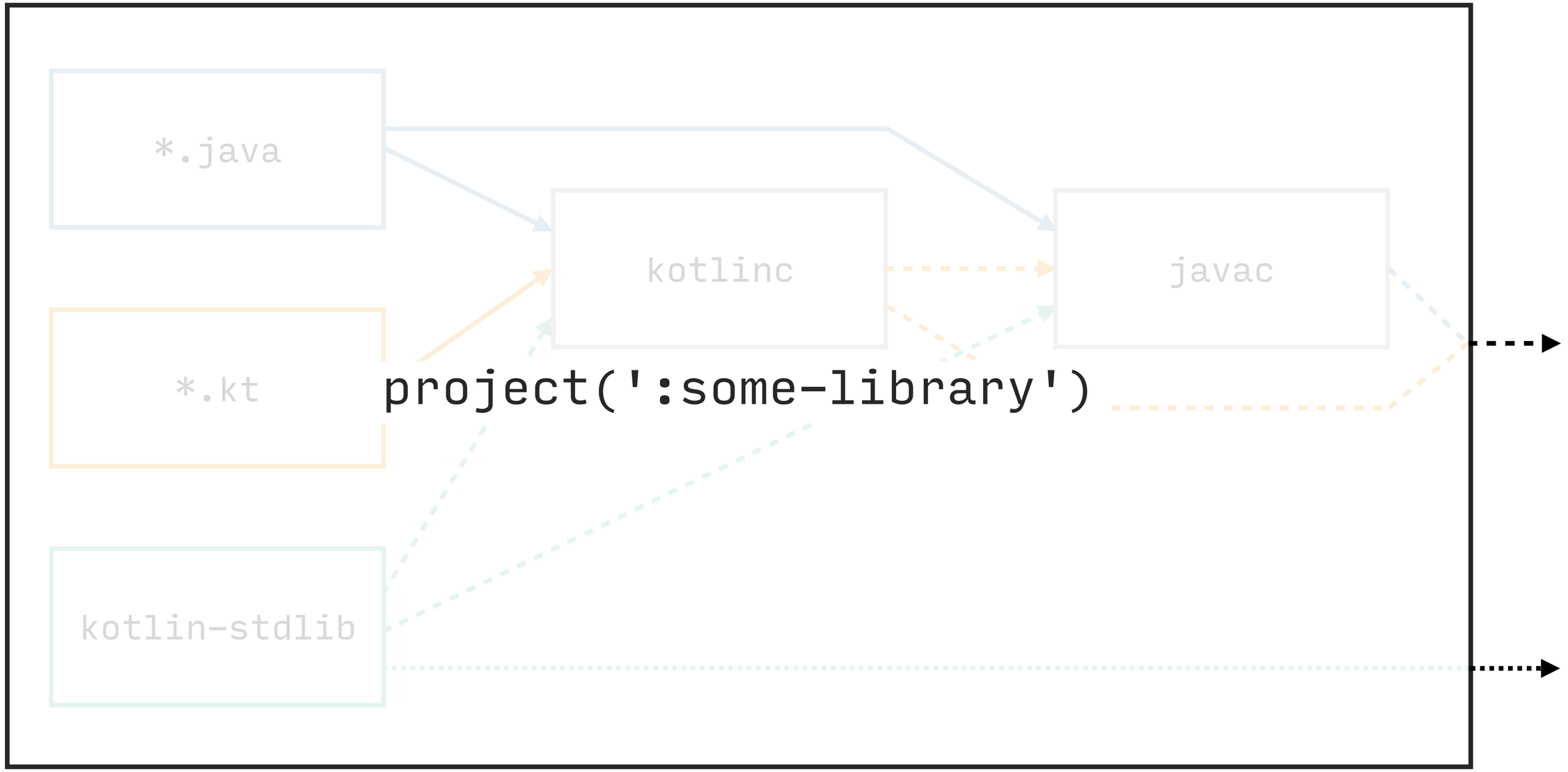

*.java
*.kt
kotlinc
javac


*.java
*.kt
kotlinc
javac


*.java
*.kt
kotlinc
javac


*.java
*.kt
kotlin-stdlib
kotlinc
javac


*.java
*.kt
kotlin-stdlib
kotlinc
javac


*.java
kotlinc
javac
*.kt
project(':some-library')
kotlin-stdlib

```
apply plugin: 'org.jetbrains.kotlin.jvm'
```

```
apply plugin: 'org.jetbrains.kotlin.jvm'

apply plugin: 'org.jetbrains.kotlin.android'
```

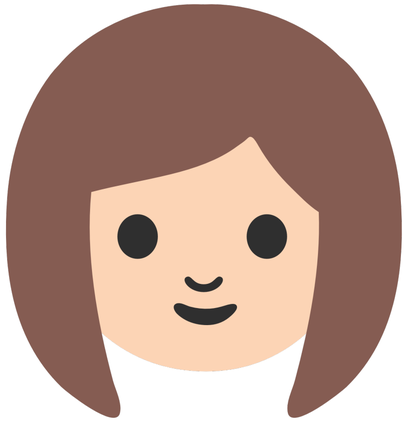
```
apply plugin: 'org.jetbrains.kotlin.jvm'

apply plugin: 'org.jetbrains.kotlin.android'

apply plugin: 'kotlin2js'
```

```
apply plugin: 'org.jetbrains.kotlin.jvm'

apply plugin: 'org.jetbrains.kotlin.android'

apply plugin: 'kotlin2js'

apply plugin: 'konan' # Different classpath dependency
```

```
apply plugin: 'org.jetbrains.kotlin.jvm'
apply plugin: 'org.jetbrains.kotlin.android'
apply plugin: 'kotlin2js'
apply plugin: 'konan' # Different classpath dependency
```

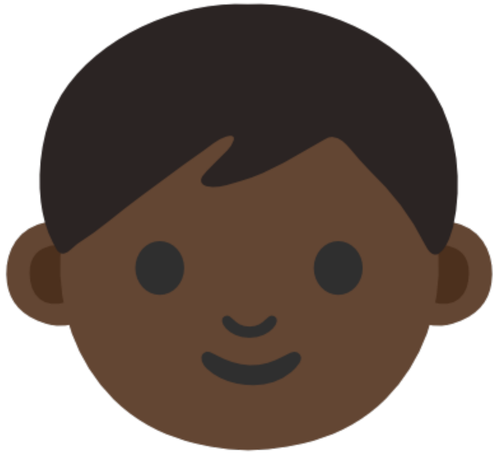
or

```
apply plugin: 'org.jetbrains.kotlin.platform.common'
apply plugin: 'org.jetbrains.kotlin.platform.jvm'
apply plugin: 'org.jetbrains.kotlin.platform.js'
apply plugin: 'konan' # Different classpath dependency
```

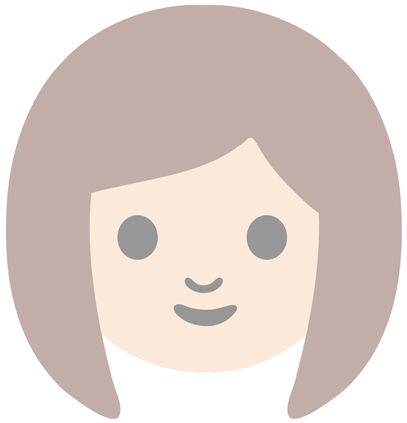


Plugin Author

Buildscript Author



App Developer



Plugin Author

Buildscript Author

App Developer

```
apply plugin: 'org.jetbrains.kotlin.jvm'
apply plugin: 'org.jetbrains.kotlin.kapt'

sourceCompatibility = JavaVersion.VERSION_1_8
targetCompatibility = JavaVersion.VERSION_1_8

dependencies {
  compile deps.kotlin.stdLibJre8
  compile deps.javaPoet
  compile deps.autoCommon

  compileOnly deps.autoService
  kapt deps.autoService

  testCompile deps.junit
  testCompile deps.truth
  testCompile deps.compileTesting
}
```

```
apply plugin: 'org.jetbrains.kotlin.jvm'
apply plugin: 'org.jetbrains.kotlin.kapt'

sourceCompatibility = JavaVersion.VERSION_1_8
targetCompatibility = JavaVersion.VERSION_1_8

dependencies {
  compile deps.kotlin.stdLibJre8
  compile deps.javaPoet
  compile deps.autoCommon

  compileOnly deps.autoService
  kapt deps.autoService

  testCompile deps.junit
  testCompile deps.truth
  testCompile deps.compileTesting
}
```

```
apply(plugin: 'org.jetbrains.kotlin.jvm')
apply plugin: 'org.jetbrains.kotlin.kapt'

sourceCompatibility = JavaVersion.VERSION_1_8
targetCompatibility = JavaVersion.VERSION_1_8

dependencies {
  compile deps.kotlin.stdLibJre8
  compile deps.javaPoet
  compile deps.autoCommon

  compileOnly deps.autoService
  kapt deps.autoService

  testCompile deps.junit
  testCompile deps.truth
  testCompile deps.compileTesting
}
```

```
apply(singletonMap('plugin', 'org.jetbrains.kotlin.jvm'))
apply plugin: 'org.jetbrains.kotlin.kapt'

sourceCompatibility = JavaVersion.VERSION_1_8
targetCompatibility = JavaVersion.VERSION_1_8

dependencies {
  compile deps.kotlin.stdLibJre8
  compile deps.javaPoet
  compile deps.autoCommon

  compileOnly deps.autoService
  kapt deps.autoService

  testCompile deps.junit
  testCompile deps.truth
  testCompile deps.compileTesting
}
```

```
apply(singletonMap('plugin', 'org.jetbrains.kotlin.jvm'))
apply plugin: 'org.jetbrains.kotlin.kapt'

sourceCompatibility = JavaVersion.VERSION_1_8
targetCompatibility = JavaVersion.VERSION_1_8

dependencies {
  compile deps.kotlin.stdLibJre8
  compile deps.javaPoet
  compile deps.autoCommon

  compileOnly deps.autoService
  kapt deps.autoService

  testCompile deps.junit
  testCompile deps.truth
  testCompile deps.compileTesting
}
```

```
apply(singletonMap('plugin', 'org.jetbrains.kotlin.jvm'))
apply plugin: 'org.jetbrains.kotlin.kapt'

sourceCompatibility = JavaVersion.VERSION_1_8
targetCompatibility = JavaVersion.VERSION_1_8

dependencies {
  compile deps.kotlin.stdLibJre8
  compile deps.javaPoet
  compile deps.autoCommon

  compileOnly deps.autoService
  kapt deps.autoService

  testCompile deps.junit
  testCompile deps.truth
  testCompile deps.compileTesting
}
```

```
apply(singletonMap('plugin', 'org.jetbrains.kotlin.jvm'))
apply plugin: 'org.jetbrains.kotlin.kapt'

sourceCompatibility = JavaVersion.VERSION_1_8
targetCompatibility = JavaVersion.VERSION_1_8

dependencies {
  compile deps.kotlin.stdLibJre8
  compile(deps.javaPoet)
  compile deps.autoCommon

  compileOnly deps.autoService
  kapt deps.autoService

  testCompile deps.junit
  testCompile deps.truth
  testCompile deps.compileTesting
}
```

```
apply(singletonMap('plugin', 'org.jetbrains.kotlin.jvm'))
apply plugin: 'org.jetbrains.kotlin.kapt'

sourceCompatibility = JavaVersion.VERSION_1_8
targetCompatibility = JavaVersion.VERSION_1_8

dependencies {
  compile deps.kotlin.stdLibJre8
  compile(deps.javaPoet)
  compile deps.autoCommon

  compileOnly deps.autoService
  kapt deps.autoService

  testCompile deps.junit
  testCompile deps.truth
  testCompile deps.compileTesting
}
```

```
android {
  // ...

  testOptions {
    unitTests.all {
      systemProperty('robolectric.dependency.repo.id', 'example-nexus')
      systemProperty('robolectric.dependency.repo.url',
          'https://nexus.example.com/content/groups/public')
    }
  }
}
```

```
android {
  // ...

  testOptions {
    unitTests.all {
      
```

assembleInternalDebug	task	DefaultTask
assembleProductionRelease	task	DefaultTask
assembleProductionReleaseUnitTest	task	DefaultTask
compileInternalDebugSources	task	DefaultTask
compileInternalReleaseShaders	task	ShaderCompile
extensions		ExtensionContainer
plugins		PluginContainer
name		String
projectDir		File
project		Project
project(String path)		Project

```
android {
    // ...

    testOptions {
        unitTests.all {
            system
        }
    }

}
```

No suggestions

build.gradle

```
apply plugin: 'org.jetbrains.kotlin.jvm'
apply plugin: 'org.jetbrains.kotlin.kapt'

sourceCompatibility = JavaVersion.VERSION_1_8
targetCompatibility = JavaVersion.VERSION_1_8

dependencies {
  compile deps.kotlin.stdLibJre8
  compile deps.javaPoet
  compile deps.autoCommon

  compileOnly deps.autoService
  kapt deps.autoService

  testCompile deps.junit
  testCompile deps.truth
  testCompile deps.compileTesting
}
```

build.gradle.kts

```
apply {
  plugin('org.jetbrains.kotlin.jvm')
  plugin('org.jetbrains.kotlin.kapt')
}

java {
  sourceCompatibility = JavaVersion.VERSION_1_8
  targetCompatibility = JavaVersion.VERSION_1_8
}

dependencies {
  compile(deps.kotlin.stdLibJre8)
  compile(deps.javaPoet)
  compile(deps.autoCommon)

  compileOnly(deps.autoService)
  kapt(deps.autoService)

  testCompile(deps.junit)
  testCompile(deps.truth)
  testCompile(deps.compileTesting)
}
```

build.gradle.kts

```
apply {
  plugin('org.jetbrains.kotlin.jvm')
  plugin('org.jetbrains.kotlin.kapt')
}

java {
  sourceCompatibility = JavaVersion.VERSION_1_8
  targetCompatibility = JavaVersion.VERSION_1_8
}

dependencies {
  compile(deps.kotlin.stdLibJre8)
  compile(deps.javaPoet)
  compile(deps.autoCommon)

  compileOnly(deps.autoService)
  kapt(deps.autoService)

  testCompile(deps.junit)
  testCompile(deps.truth)
  testCompile(deps.compileTesting)
}
```

build.gradle.kts

```
apply {
  plugin('org.jetbrains.kotlin.jvm')
  plugin('org.jetbrains.kotlin.kapt')
}

java {
  sourceCompatibility = JavaVersion.VERSION_1_8
  targetCompatibility = JavaVersion.VERSION_1_8
}

dependencies {
  compile(deps.kotlin.stdLibJre8)
  compile(deps.javaPoet)
  compile(deps.autoCommon)

  compileOnly(deps.autoService)
  kapt(deps.autoService)

  testCompile(deps.junit)
  testCompile(deps.truth)
  testCompile(deps.compileTesting)
}
```

build.gradle.kts

```
apply {
  plugin('org.jetbrains.kotlin.jvm')
  plugin('org.jetbrains.kotlin.kapt')
}

java {
  sourceCompatibility = JavaVersion.VERSION_1_8
  targetCompatibility = JavaVersion.VERSION_1_8
}

dependencies {
  compile(deps.kotlin.stdLibJre8)
  compile(deps.javaPoet)
  compile(deps.autoCommon)

  compileOnly(deps.autoService)
  kapt(deps.autoService)

  testCompile(deps.junit)
  testCompile(deps.truth)
  testCompile(deps.compileTesting)
}
```

build.gradle.kts

```
apply {
  plugin('org.jetbrains.kotlin.jvm')
  plugin('org.jetbrains.kotlin.kapt')
}

java {
  sourceCompatibility = JavaVersion.VERSION_1_8
  targetCompatibility = JavaVersion.VERSION_1_8
}

dependencies {
  compile(deps.kotlin.stdLibJre8)
  compile(deps.javaPoet)
  compile(deps.autoCommon)

  compileOnly(deps.autoService)
  kapt(deps.autoService)

  testCompile(deps.junit)
  testCompile(deps.truth)
  testCompile(deps.compileTesting)
}
```

build.gradle.kts

```
apply {
  plugin('org.jetbrains.kotlin.jvm')
  plugin('org.jetbrains.kotlin.kapt')
}

java {
  sourceCompatibility = JavaVersion.VERSION_1_8
  targetCompatibility = JavaVersion.VERSION_1_8
}

dependencies {
  compile(deps.kotlin.stdLibJre8)
  compile(deps.javaPoet)
  compile(deps.autoCommon)

  compileOnly(deps.autoService)
  kapt(deps.autoService)

  testCompile(deps.junit)
  testCompile(deps.truth)
  testCompile(deps.compileTesting)
}
```

Pleasant Build Script Authoring with Kotlin

Rodrigo B. de Oliveira

Principal Engineer at Gradle Inc.

Friday, June 23 - 2:50 PM

Room: MEDITERRANEAN II

Since Gradle 3.0, Kotlin - the expressive, performant and strongly typed language from JetBrains - can be used to author Gradle build scripts.

Gradle build scripts authored in Kotlin provide a richer IDE experience, with more precise content assist, quicker access to documentation, better source code navigation, refactoring and more.

In this talk you will learn how you and your team can start taking advantage of the improved IDE support and faster configuration times.

Powering-up your builds with Kotlin

Nadav Cohen

Developer Productivity at Netflix

Rodrigo B. de Oliveira

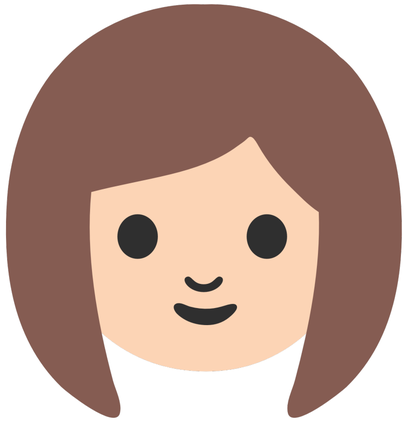
Principal Engineer at Gradle Inc.

Friday, June 23 - 3:45 PM

Room: MEDITERRANEAN I

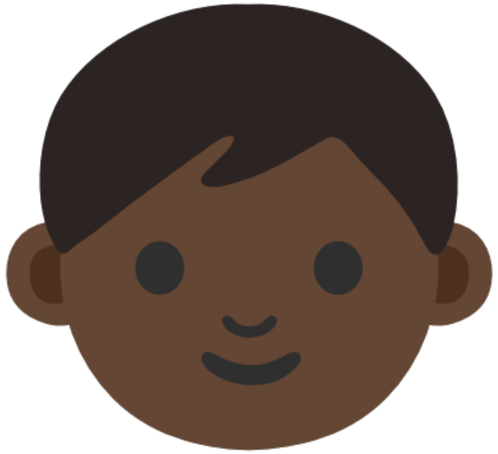
Gradle now supports a Kotlin-based DSL that allows for better tooling, type-safety, and unprecedented expressiveness. What problems does Gradle Script Kotlin solve? How would you convert existing code to Kotlin? And how would it interact with Groovy in the process?

In this live coding session, we will walk through several Groovy-based build examples and convert them to Kotlin. In this process, we'll uncover a few interesting language features and demonstrate how your builds can leverage them to take your builds to the next level.

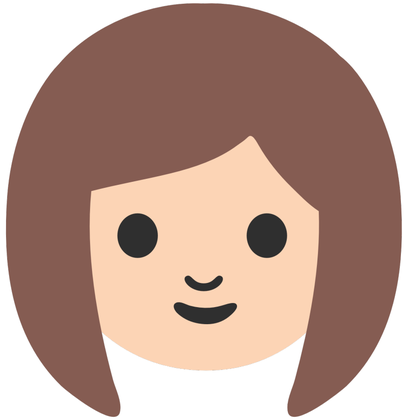


Plugin Author

Buildscript Author



App Developer



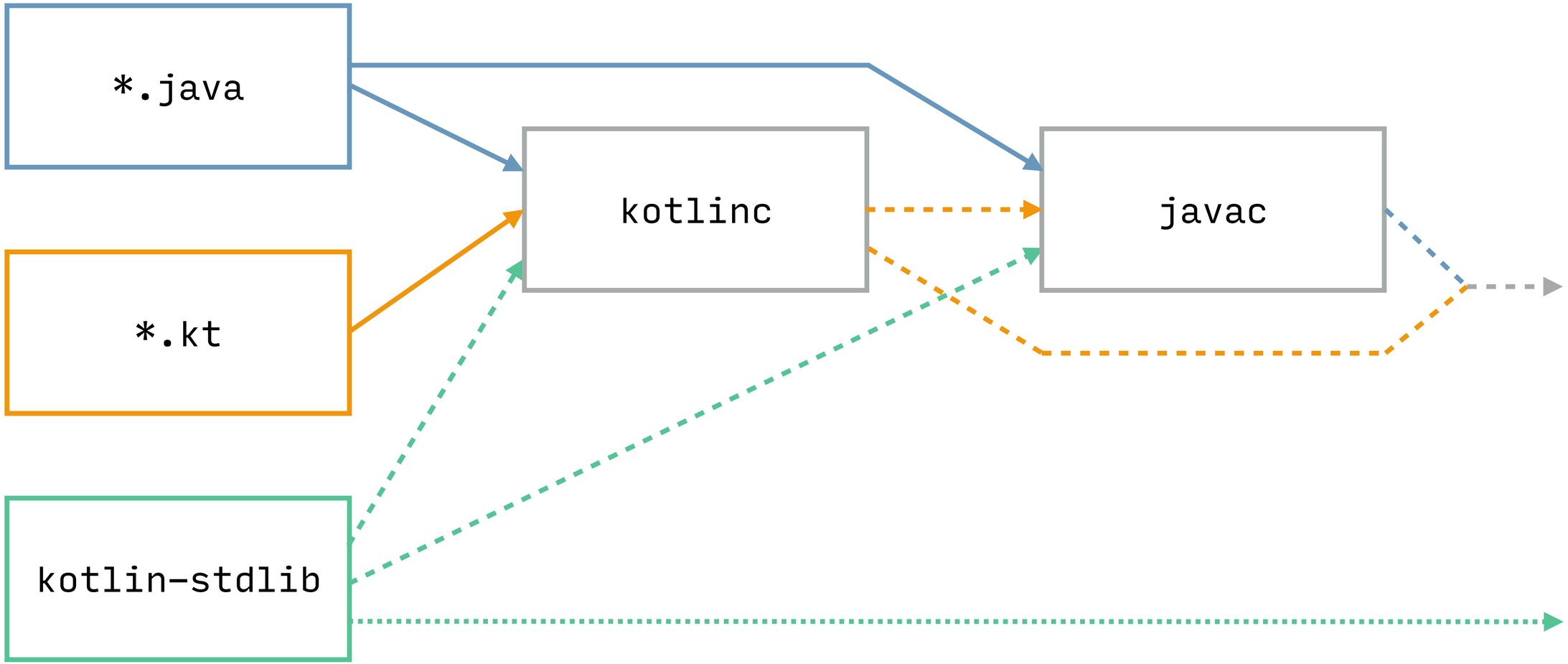
Plugin Author

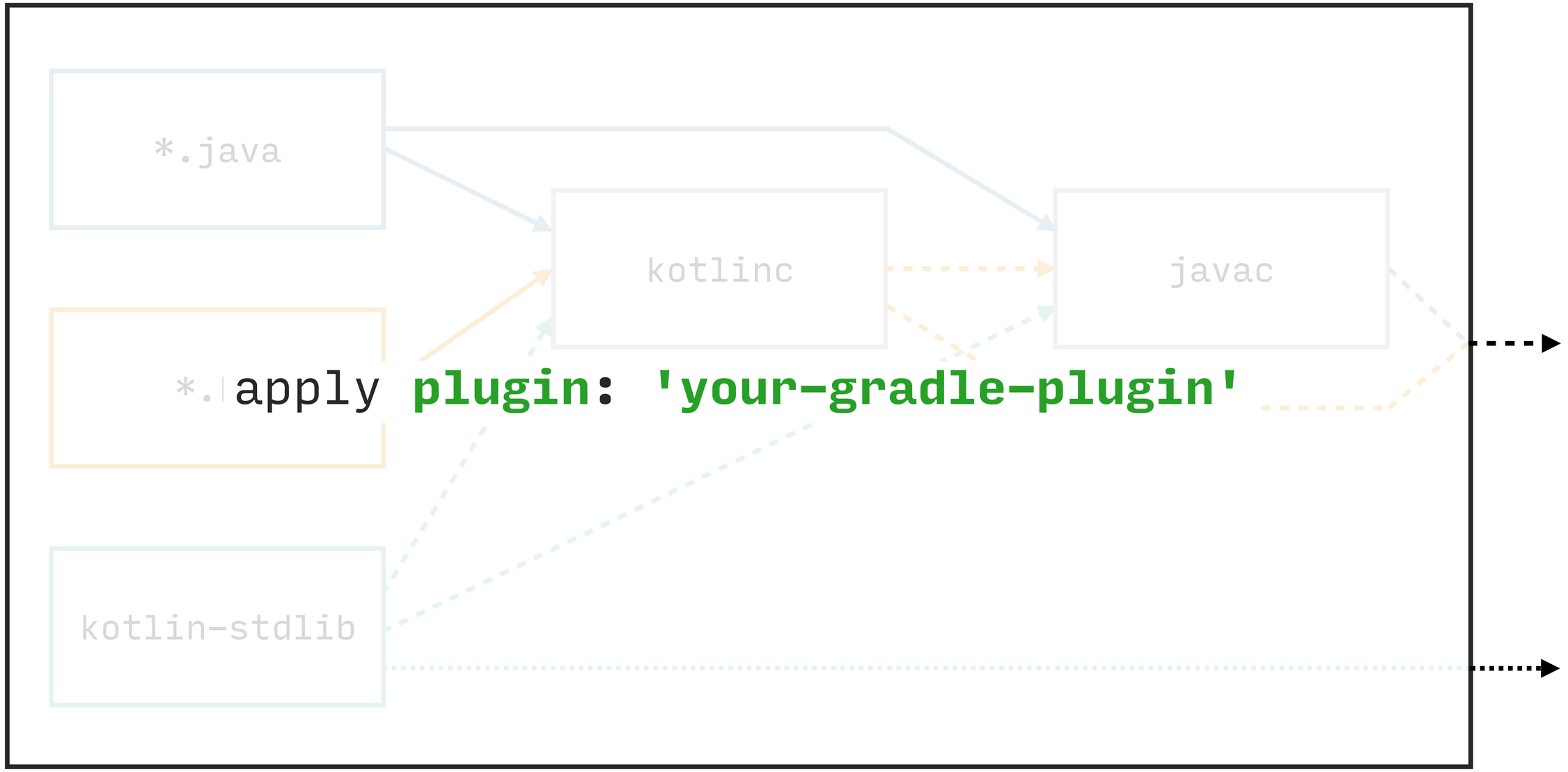
Buildscript Author

App Developer

```
class MyPlugin : Plugin<Project> {
  override fun apply(project: Project) {
    // ...
  }
}
```

```kotlin
import org.gradle.kotlin.dsl.*

class MyPlugin : Plugin<Project> {
  override fun apply(project: Project) {
    // ...
  }
}
```


*.java
*.kt
kotlin-stdlib
kotlinc
javac


*.java
kotlinc
javac
*.
apply plugin: 'your-gradle-plugin'
kotlin-stdlib

Pleasant Build Script Authoring with Kotlin

Rodrigo B. de Oliveira

Principal Engineer at Gradle Inc.

Friday, June 23 - 2:50 PM

Room: MEDITERRANEAN II

Since Gradle 3.0, Kotlin - the expressive, performant and strongly typed language from JetBrains - can be used to author Gradle build scripts.

Gradle build scripts authored in Kotlin provide a richer IDE experience, with more precise content assist, quicker access to documentation, better source code navigation, refactoring and more.

In this talk you will learn how you and your team can start taking advantage of the improved IDE support and faster configuration times.

Powering-up your builds with Kotlin

Nadav Cohen

Developer Productivity at Netflix

Rodrigo B. de Oliveira

Principal Engineer at Gradle Inc.

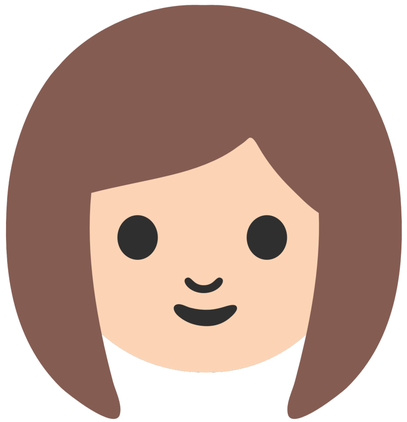
Friday, June 23 - 3:45 PM

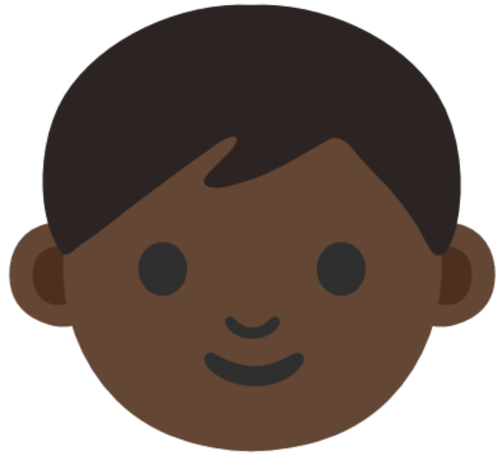
Room: MEDITERRANEAN I

Gradle now supports a Kotlin-based DSL that allows for better tooling, type-safety, and unprecedented expressiveness. What problems does Gradle Script Kotlin solve? How would you convert existing code to Kotlin? And how would it interact with Groovy in the process?

In this live coding session, we will walk through several Groovy-based build examples and convert them to Kotlin. In this process, we'll uncover a few interesting language features and demonstrate how your builds can leverage them to take your builds to the next level.

+





A Builder's Intro to Kotlin

twitter.com/ jakewharton

github.com/ jakewharton

.com