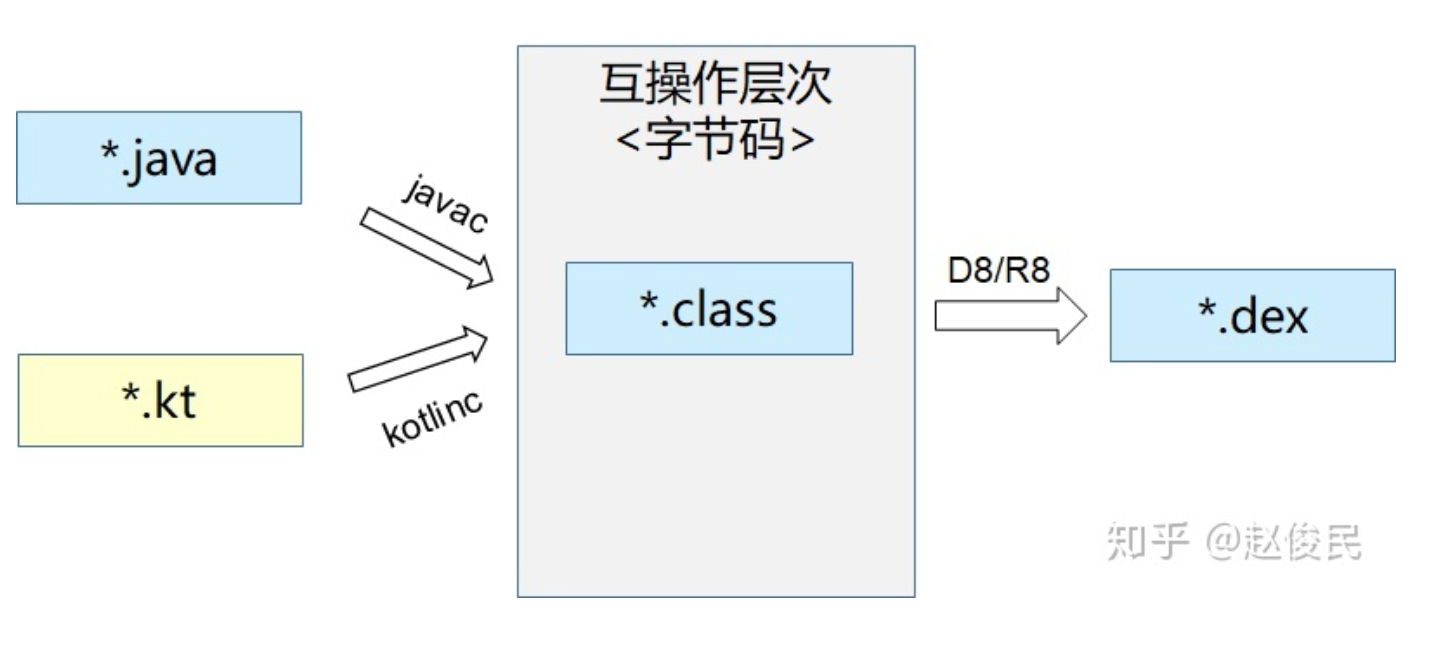
Java/Kotlin在Android平台上都是被转化为字节码文件(dex文件)，在虚拟机执行层面并没用任何区别。kotlinc/javac编译器基本在字节码层面就可以处理好互操作问题，但java/kotlin毕竟是两个不同的语言，因此在语言互操作的时候，还是需要满足一定的规范和要求。

1. 影响Java/Kotlin语言互操作各纬度对比



Java/Kotlin在JVM上产生相同的字节码，在一个虚拟机上运行，因此互操作主要受制于对象模型及其语言Spec。Java/Kotlin在编译器变换产生互相兼容的\*.class文件，互操作关键技术点就是Kotlin变化到class之后，真正的形式是什么。



1. Java/Kotlin语言互操作方式
2. Java调用Kotlin

Kotlin文件在被编译成.class文件过程中会做一些变化，因此Java在调用Kotlin编写的代码的 时候需要注意一些kotlin代码变形后的要求。

Properties类

Package-level functions

@file:JvmName("DemoUtils")标记的产生的类名

...

下面互操作方式按照转化规则都是支持的：

Kotlin类作为Java父类场景

Kotlin接口类作为Java父类场景

java overide父类定义的接口

Kotlin类作为成员（静态对象，实例对象）场景

Java通过对象访问kotlin类的成员变量，函数

Kotlin类作为参数，返回值场景

1. Kotlin调用Java

注意事项：

Methods returning void，Kotlin被看作为Unit类型

Escaping for Java identifiers that are keywords in Kotlin

Null-Safety and Platform Types

Type qualifier nicknames

Type qualifier defaults

Getters and Setter

下面互操作方式按照转化规则都是支持的：

Java类作为父类场景，访问父类的方法和field

Java接口和抽象类作为父类场景，实现父类定义的接口

Java类作为成员（静态对象，实例对象）场景

Kotlin通过对象访问Java类的成员变量，函数

Java类作为参数，返回值场景

泛型类型，需要一些特殊注意

1. Kotlin如何使用JNI

external fun foo(x: Int): Double

1. 实现原理分析

Java/Kotlin互操作完全依赖kotlinc编译器产生javac可以识别的统一的字节码\*.class文件。尽管Kotlin语言引入了大量简化编程的高级语法（语法糖），最终都会被转为字节码文件，javac可以识别对应的字节码文件，实现编译层面的互相调用。理解互操作的关键，需要从字节码层面着手。

1. kotlinc产生字节码文件



