Kotlin Study

week 1

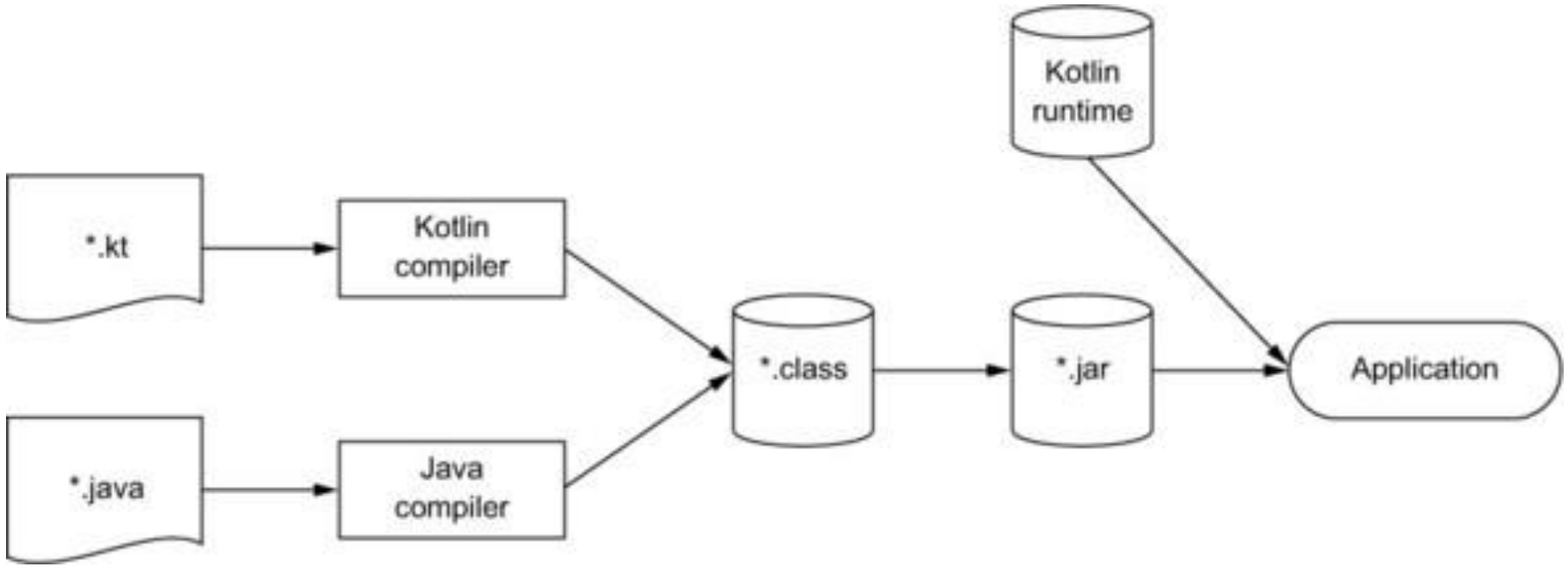
목차

- **Kotlin Compile To Runtime**
  - **어떻게 컴파일 되는가**
  - **어떻게 실행 되는가**
- **코틀린 타입(from. Kotlin Language Spec 1.6+RFC)**

Kotlin Compile & Runtime

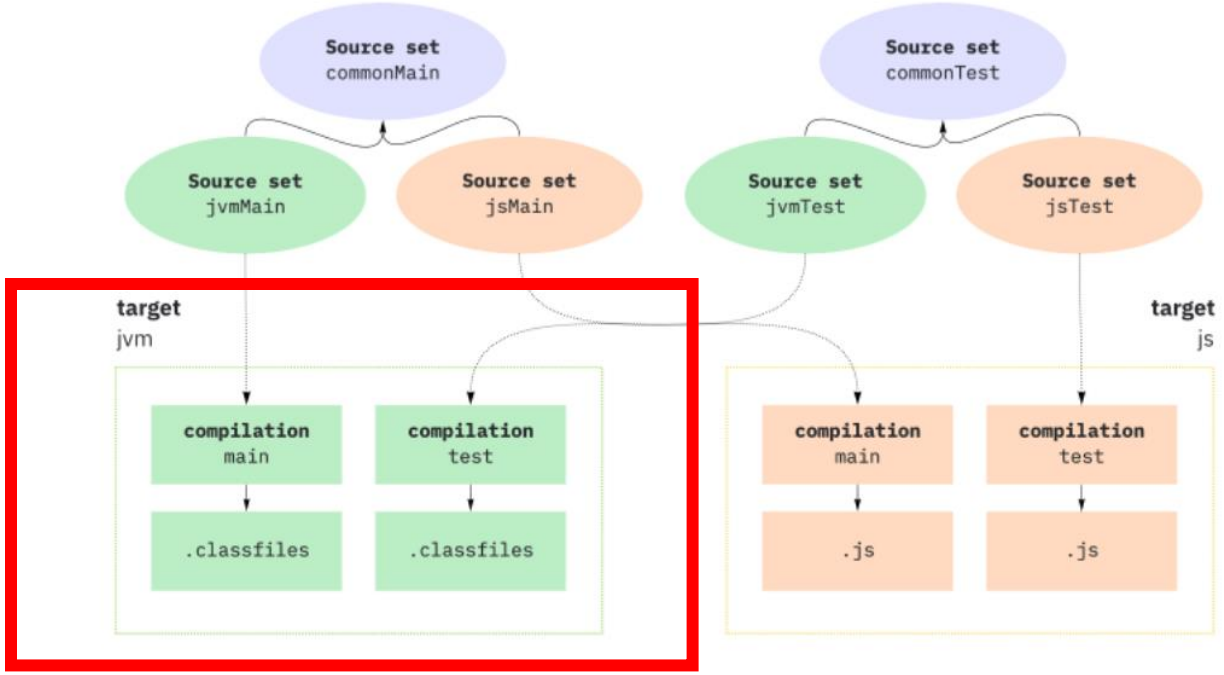
Kotlin은 어떻게 컴파일 되는가.

- 익숙한 그림.. (feat. Kotlin in Action)



Kotlin은 어떻게 컴파일 되는가.

- https://kotlinlang.org/docs/multiplatform-configure-compilations.html
- Kotlin은 멀티플랫폼(Multi Platform)을 지향하는 언어이지만,
- 우리는 JVM 플랫폼 target에 대한 부분에만 집중해봅니다.



Kotlin은 어떻게 컴파일 되는가.

https://kotlinlang.org/docs/faq.html#which-versions-of-jvm-does-kotlin-target

What does Kotlin compile down to?

When targeting the JVM, Kotlin produces Java compatible bytecode.

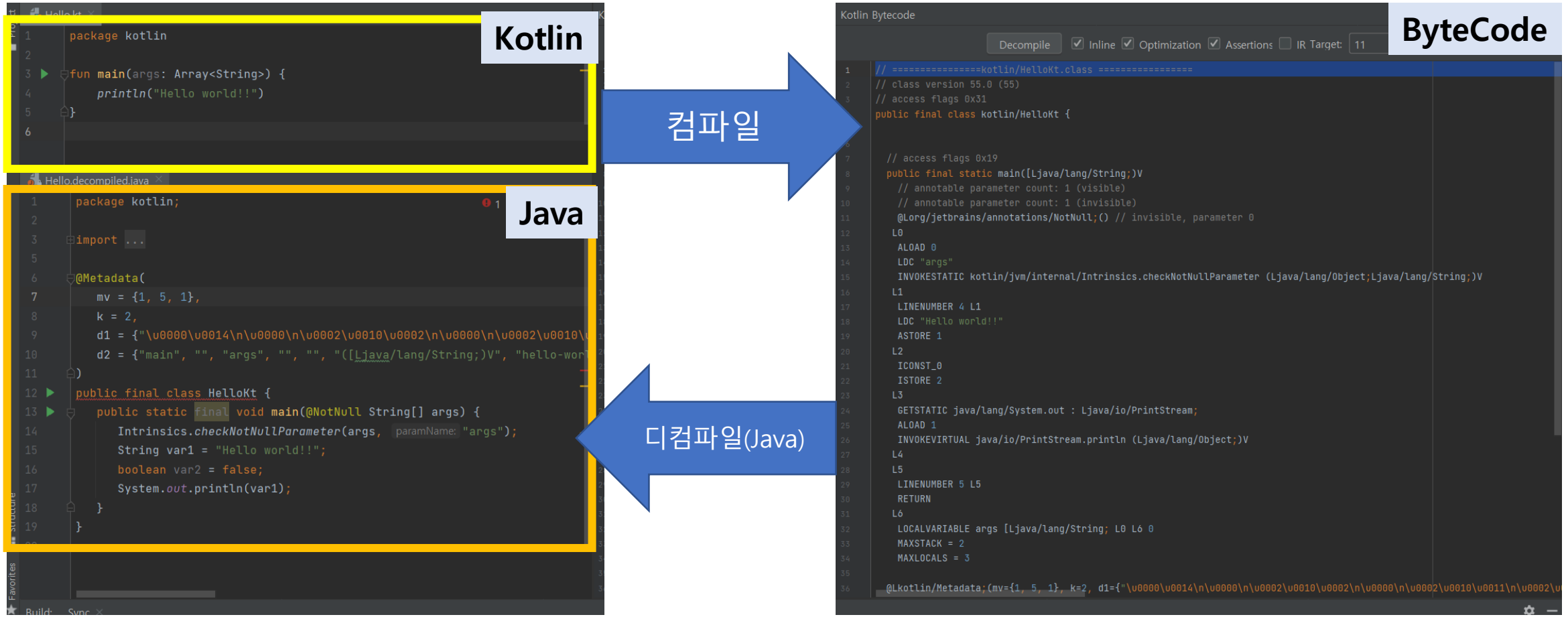
When targeting JavaScript, Kotlin transpiles to ES5.1 and generates code which is compatible with module systems including AMD and CommonJS.

When targeting native, Kotlin will produce platform-specific code (via LLVM).

Which versions of JVM does Kotlin target?

Kotlin lets you choose the version of JVM for execution. By default, the Kotlin/JVM compiler produces Java 8 compatible bytecode. If you want to make use of optimizations available in newer versions of Java, you can explicitly specify the target Java version from 9 to 17. Note that in this case the resulting bytecode might not run on lower versions.

Kotlin은 어떻게 컴파일 되는가.



Kotlin은 어떻게 컴파일 되는가.

https://blog.jetbrains.com/kotlin/2019/04/kotlin-1-3-30-released/

Specifying JVM bytecode targets 9 – 12

If you run the code under JVM versions 9, 10, 11, or 12, you can now set the corresponding `jvmTarget`. This setting will affect the version of the generated classfiles, and the resulting bytecode won't run on any lower version of JVM. Note that so far newer versions don't add any bytecode optimizations or features beyond the ones that exist in lower versions, but that is going to change in the future.

Kotlin은 어떻게 컴파일 되는가.

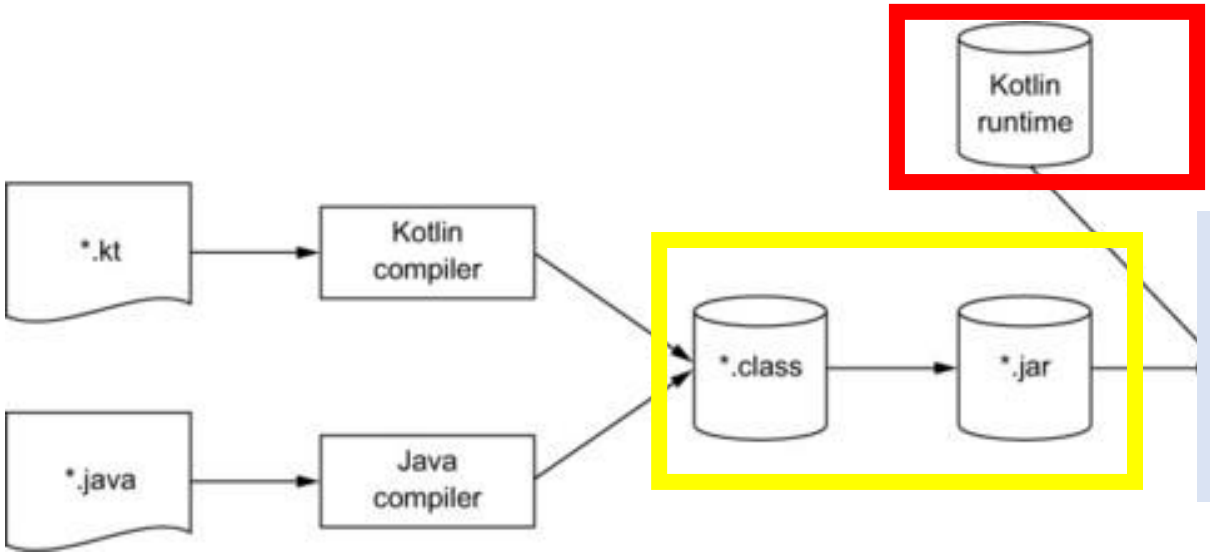
- 사실 대부분 Application을 개발할 때, 직접 빌드 환경 구성을 하기보다는 다른 도구를 사용합니다..
- 대표적으로 Maven / Gradle 을 사용하고 있구요, 각자 실무에서 사용하고있는 환경에 따라 Kotlin Application의 빌드 구성에 대한 메커니즘의 학습을 구체적으로 해나가면 될 듯 합니다.
- Gradle Kotlin에 대한 참고 링크만 걸어놓겠습니다.
  - https://kotlinlang.org/docs/gradle.html#targeting-multiple-platforms

Kotlin은 어떻게 실행되는가

- Target JVM에 맞는 바이트코드가 생성이 되면, JVM위에서 동작하는 메커니즘은 Java와 동일 한 것으로 보임
- JVM과 자바 바이트코드 관련된 내용으로 많은 정보가 있으므로 따로 정리 생략.

Kotlin은 어떻게 실행 되는가.

- 여기서 한걸음 더 들어가서 드는 의문은... 코틀린 인 액션 책에서는 아래 예제 그림이 소개가 되고 있음.
- 동일한 기능을 하는 소스코드를 Kotlin Compiler로 컴파일한 바이트코드와 Java 컴파일러로 컴파일한 바이트 코드도 동일하다면... Kotlin 런타임이 필요한 이유가 무엇일까?
- 확실한 근거와 메커니즘을 확인하지는 못했지만, 아래 StackOverFlow 글을 참고하여 Kotlin Runtime의 관계에 대해 추측해볼 수 있었습니다.
  - https://stackoverflow.com/questions/30565036/why-kotlin-needs-to-bundle-its-runtime-after-compiled



Kotlin 컴파일 이후
java ByteCode와 완벽하게 호환되는 Kotlin Byte Code가 생성이 된다면,
실행 환경에서 Kotlin Runtime Library가 필요한 이유가 뭐가 있을까?

JRE외에 Kotlin 특화 패키지들???

Kotlin Type

Kotlin Type : Reference

- Kotlin은 JetBrains라는 회사의 주도로 개발되고 유지보수 되는 언어이므로 비교적 스펙과 공식 문서가 디테일하고 체계적으로 관리되고 있는 오픈소스 라고 생각이 듭니다.(본인 피셜)
- 다만, 어떤 언어를 학습하는데 있어서 처음부터 이런 백서(White Paper)같은 문서를 A-Z까지 보는 것은 상당히 가성비가 떨어지는 학습 방법이라 저는 별로 권장하고 싶지 않은 학습 방법이라고 생각합니다.
- Kotlin Language Spec의 모든 컨텐츠를 다루기에는 여럭이 안되므로, “Type System” 챕터만 부분 발췌하여 정리하는 차원에서 스터디를 마무리 해보고자 합니다. 관심 있으신 분들을 위해 링크를 아래 남겨놓습니다.
  - https://kotlinlang.org/spec/introduction.html


Kotlin language specification

Version 1.6-rfc+0.1

Marat Akhin Mikhail Belyaev

Kotlin type

Kotlin In Action이나 다른 Kotlin 개념서에서
일반적으로 분류하는 개념과는 다소 차이가 있습니다.

2.1 Type kinds

For the purposes of this section, we establish the following type kinds — different flavours of types which exist in the Kotlin type system.

- Built-in types
- Classifier types
- Type parameters
- Function types
- Array types
- Flexible types
- Nullable types
- Intersection types
- Union types

We distinguish between *concrete* and *abstract* types. Concrete types are types which are assignable to values. Abstract types need to be instantiated as concrete types before they can be used as types for values.

Kotlin type – Built In Type

타입 시그니처	설명
Any	• Supertype -> 모든 타입의 슈퍼 타입 • Java의 Object 타입과 대응
Nothing	• Uninhabited type -> 모든 타입의 서브타입 • 용례 : 종료가 되지 않은 제어 흐름, 예외 리턴, 제어 흐름을 전환 • Spec문서에서는 상세 설명이 부족하여 다른 기술 블로그를 통해 관련 내용 보완합니다.
Function	• Function Types 에서 설명
Int / Short / Byte / Long	• Built-in Integer types
Array	• 코틀린 배열 • 기본 자료형에 따라 Element 타입에 따라 특화된 Array 타입이 정의되어 있음(DoubleArray, FloatArray, LongArray, IntArray….)

Kotlin Type – Classifier Type

- class, interface, object
- 다형성을 지원하는 Kotlin 특성상 두 가지의 Classifier Type을 지원
  - Simple classifier types
    - 예시
      class BaseClass
      class DerivedClass : BaseClass
      class InvalidClass : BaseClass? // nullable Type은 Supertype이 될 수 없으므로 부적절한 선언
  - Parameterized classifier types
    - **제너릭(Gerneric)**과 유사한 개념
    - 예시
      interface Generic<A, B>
      interface Out<out A>

Kotlin Type – Type Parameters

- **제너릭(Generic)**

Actual support for multiple class type bounds would be needed only in very rare cases, such as the following example.

```
interface Foo
interface Bar

open class A<T>
class B<T> : A<T>

class C<T> where T : A<out Foo>, T : B<out Bar>
// A convoluted way of saying T <: B<out Foo & Bar>,
// which contains a non-denotable intersection type
```

Kotlin Type – Function type

- Kotlin은 함수가 일급 객체(First-Class)로 정의 될 수 있음
- 함수 시그니처 자체가 Data Type처럼 간주되고 활용이 가능.
- 함수 타입에 대한 변성(Variance)도 가능

Example:

```
// A function of type Function1<Number, Number>
//   or (Number) -> Number
fun foo(i: Number): Number = ...

// A valid assignment w.r.t. function type variance
// Function1<in Int, out Any> :> Function1<in Number, out Number>
val fooRef: (Int) -> Any = ::foo

// A function with receiver of type Function1<Number, Number>
//   or Number.() -> Number
fun Number.bar(): Number = ...

// A valid assignment w.r.t. function type variance
// Receiver is just yet another function argument
// Function1<in Int, out Any> :> Function1<in Number, out Number>
val barRef: (Int) -> Any = Number::bar
```

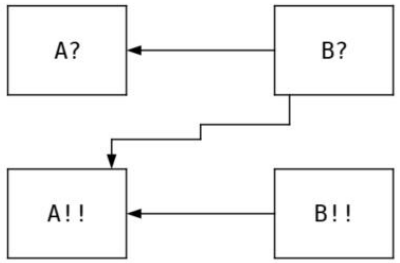
Kotlin Type – Flexible types

- Null Safety를 지원하지 않는 플랫폼에 대응하기 위한 코틀린의 특별한 타입 시스템
- **플랫폼 타입(Platform Type)** : 다른 플랫폼 언어(Java)에서 전달되어서 Nullable 여부를 알 수 없는 타입(Effective Kotlin Item.3)

Kotlin Type – Nullable types

- 코틀린은 NullSafty를 지원하기 위해 평행 성격의 타입 시스템을 채택(nullable / non-nullable)
- 기본은 Non-Nullable
- Nullable이라는 개념이 형(Type)에 가깝게 관리가 되고 있으므로

Nullability lozenge



Nullability lozenge represents valid possible subtyping relations between two nullable or non-nullable types in different combinations of their *versions*. For type T, we call T!! its non-nullable version, T? its nullable version.

Kotlin Type – Intersection types

?

Kotlin Type – Union types

- ?