

Kotlin : 러시아 Baltic Sea 인근에 있는 섬

Kotlin : Introduction

Mobile Software
2021 Fall

All rights reserved, 2021, Copyright by Youn-Sik Hong (편집, 배포 불허)

What to do next?





- Kotlin 등장 배경 및 개요
- Kotlin 기본 프로그래밍
- Kotlin 강의 노트에서는
 - 소스 코드를 별도로 제공하지 않음
 - 실행 결과를 포함하지 않음.
 - 직접 코드를 입력하고, 강의 노트 설명을 확인해야 함.
 - **Required assignments**
 - 강의 노트에 해당하는 강의를 종료되면, 해당 주 일요일 23:59:59까지 kotlin 실습 파일 제출 → 평소 점수에 반영

Why Kotlin?(1/2)

- **Google announces Kotlin for Android**
 - <https://www.youtube.com/watch?v=d8ALcQiuPWs>
 - 2017. 5. 18 google I/O 에서
 - Google이 Android 공식 언어로 kotlin을 추가
 - Android Studio 3.0부터 kotlin 기본 지원
 - 이전 버전에서도 plug-in만 설치하면 됨

Why Kotlin?(2/2)

- <https://kotlinlang.org> – official web site
- JetBrains (<http://www.jetbrains.com>)에서 개발
 - 2011년 최초 공개 → 2016년 정식 버전(1.0) 출시
 - Java와 100% 호환 : Java ↔ Kotlin

 Multiplatform Mobile The natural way to share code between mobile platforms	 Server-side Modern development experience with familiar JVM technology	 Web Frontend Extend your projects to web	 Android Recommended by Google for building Android apps
--	---	---	--

Google은 왜 java에서 Kotlin으로 바꿨을까?



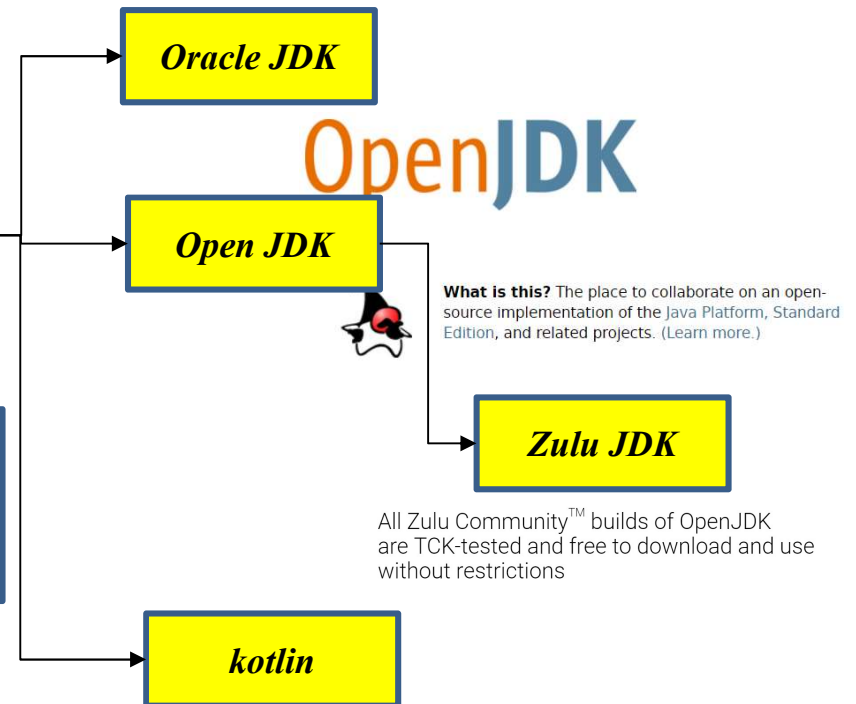
James Gosling
at Sun Microsystems
Father of Java



Oracle America, Inc. v. Google, Inc.



a current legal case within the USA related to the **nature of computer code and copyright law.**



April 13, 2021

U.S. Supreme Court Rules for Google in Landmark Decision Recognizing Fair Use of Computer Code

Court finds that Google's re-use of code from Oracle's Java API constitutes fair use under the Copyright Act

On April 5, 2021, the U.S. Supreme Court released its decision in *Google LLC v. Oracle America, Inc.*,¹ a long-awaited case addressing the boundaries of the copyright fair use defense as it applies to computer code. This is the first time the Court has addressed fair use in 28 years. It issued a resounding opinion in favor of innovation that dramatically changes how courts will analyze fair use with respect to computer software going forward.

By a 6-2 vote, the Court sided with Google, finding that its use of 11,500 lines of declaring code from Oracle's Java Application Programming Interface (API) to enable coding in the Java programming language for the Android operating system constituted fair use and did not violate the copyright law. The Court concluded that "where Google reimplemented a user interface, taking only what was needed to allow users to put their accrued talents to work in a new and transformative program, Google's copying ... was fair use."

Kotlin 특징

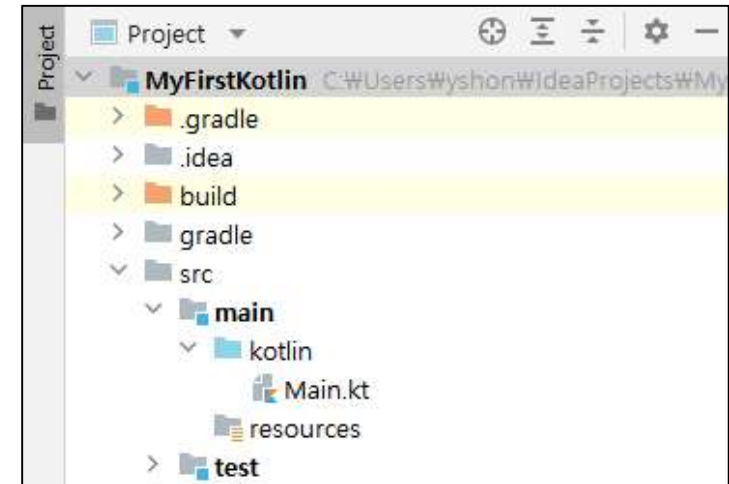
- **Static type**
 - 모든 프로그램 구성 요소의 타입을 compile time에 알 수 있음.
 - 타입 추론 (type inference)
- **Concise and safe**
 - Python, swift 등 최근 언어 추세
- **Multi-paradigm language**
 - 함수형 프로그래밍과 객체지향 프로그래밍이 모두 가능
 - 함수형 프로그래밍
 - **순수 함수** (pure function) : 같은 인자에 대해 항상 같은 결과를 반환
 - **람다 식** (lambda expression) : 이름이 없는 함수를 식으로 표현
 - **고차 함수** (high order function) : 다른 함수를 인자로 사용하거나 함수를 결과로 반환

What to do next?

- Kotlin 등장 배경 및 개요
- Kotlin 기본 프로그래밍

Kotlin 프로그램 구조 : 함수 (1)

- **main() 함수만 있는 경우**
 - 함수(fun) 및 클래스(class)
 - 파일 확장자: **kt**



```
1 fun main() {  
2     println("Hello World!")  
3 }  
4
```

main() 함수는 반드시 포함해야 함
→ 프로그램 실행을 위한 entry point 역할

Decompile : Kotlin → bytecode → java

Main.kt

```
1 fun main() {  
2     println("Hello World!")  
3 }  
4
```



실행 : Run > Run



Decompile 탭 클릭



Tools > Kotlin
> Show Kotlin bytecode

Java는 클래스로만 구성해야 하며,
클래스 중 하나는 main() 메소드를
갖고 있어야 함.

```
Main.kt x Main.decompiled.java x  
1 import kotlin.Metadata;  
2  
3 @Metadata  
9 public final class MainKt {  
10     public static final void main() {  
11         String var0 = "Hello World!";  
12         boolean var1 = false;  
13         System.out.println(var0);  
14     }
```

Kotlin 프로그램 구조 : 함수 (2)

- import 문을 포함

```
import kotlin.math.PI
import kotlin.math.abs

fun main() {
    println(PI)
    println(abs(x: -12.6))
    println("Hello World!")
}
```

Kotlin 프로그램 구조 : 함수 + 클래스

- 클래스와 main() 함수가 있는 경우
 - Java 처럼 파일 이름으로 클래스 이름을 사용하지 않아도 됨.

```
class Person(var name:String, var age:Int)

fun main() {
    var hong = Person( name: "Hong", age: 24)
    println(hong.age)
}
```

회색 단어(name, age)는 입력하면 안됩니다.
에러가 발생합니다.

IntelliJ에서 어떤 parameter를 입력해야 하는지
알려주기 위해 보여주는 것입니다.

Kotlin functions (1/2)

```
fun main() {  
    println(sum( a: 23, b: 47))  
    print_sum( a: 12, b: 23)  
}
```

return type은 유추(infer)
할 수 있으므로 생략

```
fun sum(a:Int, b:Int): Int {  
    return a + b  
}
```

return 값(타입: **Int**)이 있음

```
fun print_sum(a:Int, b:Int): Unit {  
    println("sum of $a and $b is ${a+b}")  
}
```

return 값이 없음(**Unit**)
Unit 은 생략 할 수 있음.



```
fun sum(a:Int, b:Int) = a + b  
  
fun print_sum(a:Int, b:Int) {  
    println("sum of $a and $b is ${a+b}")  
}
```

return type은 유추(inter)할 수
있기 때문에 생략

Unit 생략

Kotlin functions (2/2)

```
fun main() {  
    println(max( a: 17, b: 23))  
}
```

```
fun max(a:Int, b:Int): Int {  
    return if (a > b) a else b  
}
```

본문이 식(expression)인 함수

```
fun max(a: Int, b: Int): Int = if ( a > b ) a else b
```

함수 본문이 블록(block)

→ block : 중괄호({, })로 둘러싸인 문장

return 타입 생략 – 본문이 식인 함수에서만 가능

```
fun max(a: Int, b: Int) = if ( a > b ) a else b
```

val, var : 2 종류의 변수

- **val** (value): **Immutable** variables. read-only variables
 - 값을 변경할 수 없는 변수. → java의 **final**과 같음.

```
val a: Int = 1 // immediate assignment
val b = 2      // `Int` type is inferred
val c: Int     // Type required when no initializer is provided
c = 3          // deferred assignment
```

val 형 변수는 선언할 때 값을 할당하거나, 선언하고 나서 1회 값을 할당할 수 있음.
값이 할당된 이후에는 값을 변경할 수 없음.

- **var** (variable) : **Mutable** variables. writeable variables
 - 언제든지 값을 바꿀 수 있는 변수.

```
var x = 5 // `Int` type is inferred
x += 1
```