

# EDCAN 수업 자료

Kotlin 1

이 수업에서 배우는 것들

**Kotlin이란, 출력, 변수, 자료형, Null,  
입력, 연산자, List, Array, Map,  
if문, for문, while문, when문**

# Kotlin 이란

이란 억 ㅋㅋ



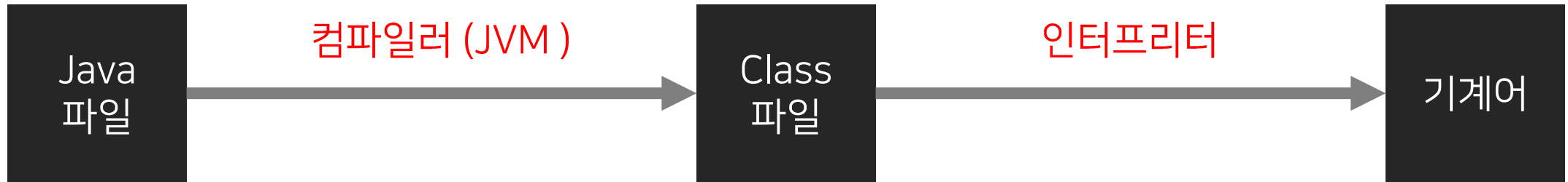
# Kotlin 이란

- 2011년 젓브레인에서 제작한 언어
- Android 개발을 위해서 사용하는 언어 (구글이 공식적으로 권장하는 언어)
- Java와 100% 호환된다.
- Java의 특징을 전부 가져온다.

# Java 란

- 1991년 오라클에서 제작한 언어
- 안드로이드 개발에 사용된다. (권장 되지 않음)
- 객체 지향 언어

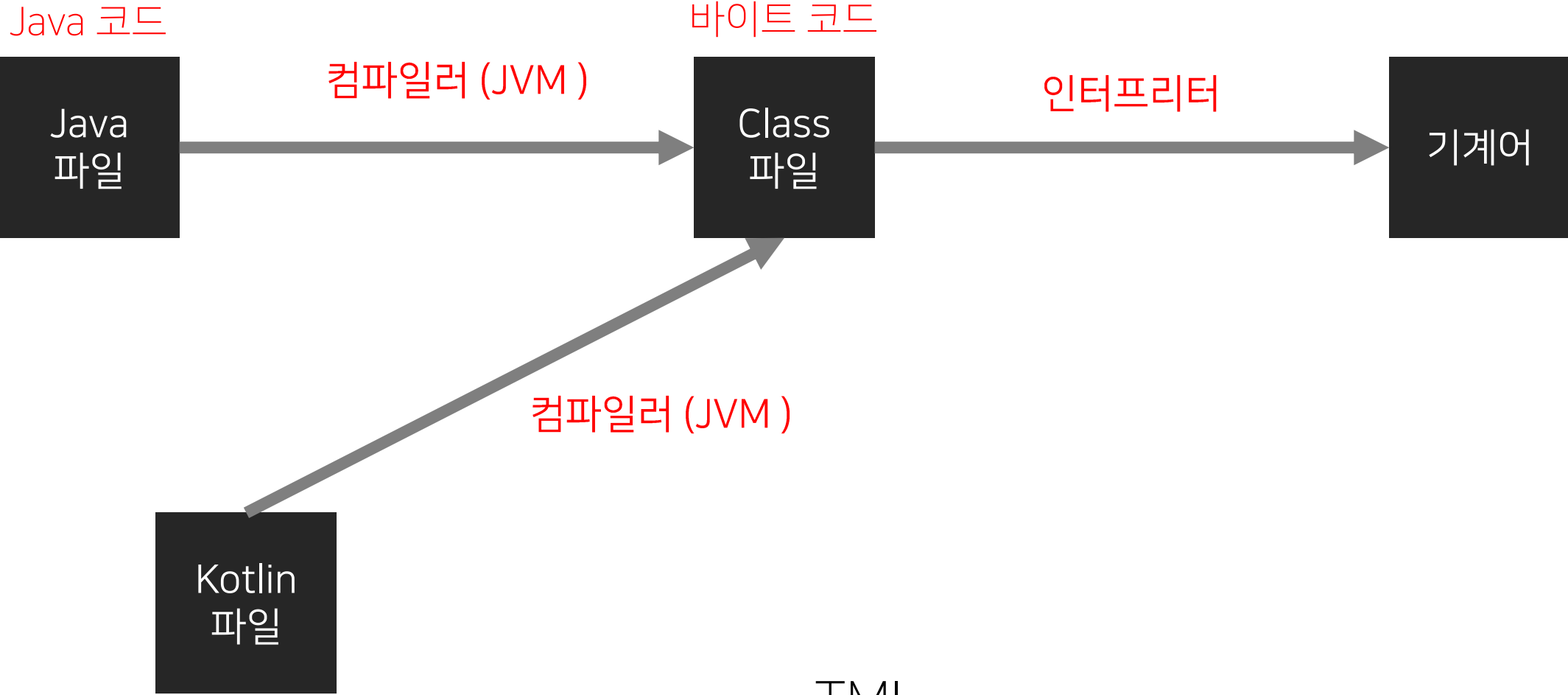
## Java의 빌드 과정



TMI

JVM : Java Virtual Machine, 자바 가상 머신

# Kotlin과 Java가 100% 호환되는 이유



TMI  
JVM : Java Virtual Machine, 자바 가상 머신

# Kotlin과 Java가 100% 호환되는 이유

더 자세한 내용은 이 영상을 참고하자



<https://youtu.be/VvVruEDCSSY>

JVM, JRE, JDK가 뭔가요? - 알박한 코딩사전

# Kotlin이 안드로이드 개발 언어가 된 이유 (TMI)

- 기존의 안드로이드 개발 언어는 Java였다.
- Java를 제작한 오라클이 자신들의 언어로 안드로이드를 만들었다는 이유로 구글에 소송을 걸었다
- 구글은 Java와 100% 호환되고 문법이 더 간단한 Kotlin을 공식 언어로 지정하고 대부분의 회사에서도 Kotlin을 사용해서 개발하는 추세이다.



**출력**

# 출력

- print() 함수 사용
- println() : 문자열을 출력하면서 줄바꿈을

```
#include <stdio.h>


int main() {
    printf("Hello World\n");
}
```

C언어 출력 :printf() 함수 사용

```
fun main() {
    print("Hello World");
}
```

Kotlin 출력 :print() 함수 사용

# 변수



```
fun main() {  
    val 변수 : Int = 3  
    println(변수)  
}
```

# 변수

```
fun main() {  
    val 변수 : Int = 3  
    println(변수)  
}
```

변수를 선언할 때 사용하는 키워드

# 변수

```
fun main() {  
    val 변수 : Int = 3  
    println(변수)  
}
```

## 변수의 자료형

변수를 선언하면서 값을 넣어주는 경우에는 자동으로 Int형으로 지정한다.  
이때는 자료형을 적어주지 않아도 된다.

# 변수

```
fun main() {  
    val 변수 : Int = 3  
    println(변수)  
}
```

변수에 넣어주는 값

# 변수

val을 사용해서 만든 변수는  
값을 수정하지 못하는 변수이다



```
fun main() {  
    val 변수 : Int = 3  
    변수 = 10 에러 발생  
}
```

값이 수정 가능한  
변수를 만들기 위해서는  
var을 사용한다.



```
fun main() {  
    var 변수 : Int = 3 var 사용  
    변수 = 10  
}
```

# 변수



```
fun main() {  
    var a : Int  
  
    a = 100  
}
```

변수를 선언한 뒤  
나중에 값을 할당하기 위해서는  
var을 사용해야한다.

이때 변수를 선언할 때  
자료형을 지정해야 한다



# Null



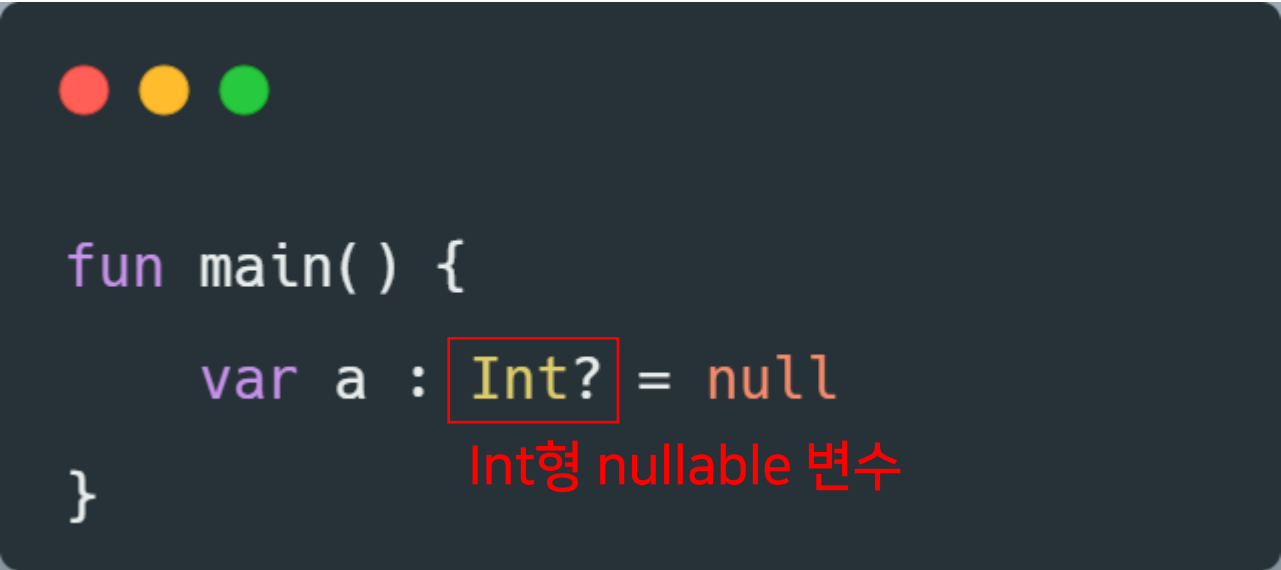
null은 값이 없는 것을 의미한다.

# Null

- 변수에 null이 들어왔다면 버그가 생길 확률이 높아진다.
- Kotlin에서는 변수에 null이 들어온 경우를 엄격하게 처리한다.

# Null

- 변수에 null이 들어 올 수 있는 변수를 nullable 변수라고 한다.
- nullable 변수를 만들 때에는 자료형 옆에 ? 를 붙여준다.



```
fun main() {  
    var a : Int? = null  
}
```

Int형 nullable 변수

# Null

- nullable 변수는 함부로 더하거나 빼거나 연산 할 수 없다.
- 이때 null safty가 작동한다.

```
fun main() {  
    var a : Int? = null  
    val b : Int = a + 100;  
}
```

에러 발생

아무것도 없는 곳에 100을 더하는게  
상식적으로 불가능 하다.

# Null safety

```
fun main() {  
    var a : Int? = null  
    a?.plus(100);  
}
```

변수 뒤에 ?를 붙여주면

만약 변수의 값이 null이면 뒤의 코드를  
실행하지 않고 종료한다

```
fun main() {  
    var a : Int? = null  
    a!! + 100;  
}
```

변수 뒤에 !!를 붙여주면

변수가 **null이 아님을 보장한다.**  
실행하는 도중에 만약 null이면 에러 발생  
-> 함부로 사용하지 말자

**연산자**

# 연산자

$+$ ,  $-$ ,  $*$ ,  $/$ ,  $\%$ ,

$+=$ ,  $-=$ ,  $*=$ ,  $/=$ ,  $\%=$ ,

$<$ ,  $>$ ,  $<=$ ,  $>=$ ,  $==$ ,  $!=$



## 이것들 다 됨

연산자 끝

**List, Array, Map**



# Array (배열)

- 배열 : 변수들을 선형으로 연결해서 만든 것
- 배열안의 각 요소들은 **인덱스**라고 하는 순서로 아이টে을 구분한다.
- 배열을 한번 만들면 아이টে을 수정할 수는 있지만 아이টে을의 크기를 늘릴 수는 없다

숫자_배열	1	2	2	3	4	5
	인덱스 0번	1번	3번	4번	5번	6번

TMI : 모든 인덱스는 0번 부터 시작 된다.

# Array (배열) – 배열 만들기

숫자_배열	1	2	2	3	4	5
인덱스	0번	1번	3번	4번	5번	6번



```
fun main() {  
    val 숫자_배열 = arrayOf(1, 2, 3, 4, 5)  
  
    //0번 인덱스 요소 가져오기  
    숫자_배열[0] //1  
}
```

# Array (배열) – 배열 만들기

- 기본적으로 Kotlin의 배열은 여러 자료형을 넣을 수 있다.




```
val 여러_자료형_배열 = arrayOf(1, "문자", 1.2, "EDCAN 정상영업 합니다.")
```

여러_자료형_배열	1	"문자"	1.2	"EDCAN 정상영업 합니다."
인덱스	0번	1번	3번	4번


# Array (배열) – 배열 만들기

- 하나의 자료형만 들어올 수 있는 배열을 만들기 위해선 앞에 자료형을 붙여준다.



```
val Int형_배열 = intArrayOf(1, 2, 3, 4)
val Float형_배열 = floatArrayOf(1.1f, 1.2f, 3.14f)
```

# Array (배열) – 배열 수정하기



```
val 숫자_배열 = arrayOf(1, 2, 3, 4, 5)
```

```
//0번 인덱스 요소 가져오기
```

```
숫자_배열[0] //1
```

```
//0번 인덱스 요소 100으로 수정
```

```
숫자_배열[0] = 100
```

# Array – 배열의 다양한 기능

설명	사용법
아이템 접근하기	array[index]
배열 크기	array.size
배열이 비어있는지 확인	array.isEmpty()
배열에 해당 요소가 있는지 확인	array.contains( item )
배열 정렬	array.sort()

# List

- 배열과 비슷하지만, 개수 제한이 없다.
- 무제한으로 아이템을 추가 할 수 있다.
- 하나의 리스트는 하나의 자료형만 들어올 수 있다.

# List - 리스트 만들기



```
val Int형_리스트 = listOf<Int>(1, 2, 3, 4)
```

제네릭 문법

//2번째 인덱스 아이템 가져오기

```
Int형_리스트[2]
```

제네릭 문법은 자료형을 인수로 넘겨주는 것과 비슷하다.



# List - 리스트 만들기

기본 list는 값을  
수정할 수 없다.



값이 수정 가능한  
리스트를 만들기 위해서는  
**mutableList**를 사용한다.



```
val Int형_리스트 = listOf<Int>(1, 2, 3, 4)
```

```
Int형_리스트[1] = 10
```

 에러 발생

```
val Int형_리스트 = mutableListOf<Int>(1, 2, 3, 4)
```

```
Int형_리스트[1] = 10
```

 mutableList 사용

# List – 리스트에 요소 추가하기

mutableList에 요소를 추가할 때는 add() 함수를 사용한다.



```
val 수정_가능한_Int_리스트 = mutableListOf<Int>()
```

```
수정_가능한_Int_리스트.add(10)
```

```
수정_가능한_Int_리스트.add(20)
```

```
수정_가능한_Int_리스트.add(30)
```

```
수정_가능한_Int_리스트.add(40)
```

```
수정_가능한_Int_리스트.add(50)
```

# List – 리스트의 다양한 기능

설명	사용법
아이템 접근하기	list[index]
아이템 추가	list.add( item )
아이템 삭제	list.remove( item )
아이템 인덱스로 삭제	list.removeAt({index})
리스트 크기	list.size
리스트가 비어있는지 확인	list.isEmpty()
리스트에 해당 요소가 있는지 확인	list.contains( item )
리스트 정렬	list.sort()
리스트 비우기	list.clear()

# Map

- Key와 value를 한쌍으로 갖는 자료형

**Baseball** = 야구


**Apple** = 사과

**Developer** = 개발자

key

value

# Map – map 만들기




Key의 자료형, value의 자료형

```
val 영어사전 = mapOf<String, String>(  
    "Baseball" to "야구",  
    "Apple" to "사과",  
    "Developer" to "개발자",  
)
```

```
영어사전["Baseball"] //야구
```

# Map – map 만들기

- 수정 가능한 map : mutableMap



```
val 영어사전 = mutableMapOf<String, String>(
    "Baseball" to "야구",
    "Apple" to "사과",
    "Developer" to "개발자",
)
```

```
영어사전["Baseball"] = "베이스볼"
```

**If, for, while, when 문**

# if-else if-else문

- C언어의 if과 동일
- 다음과 같은 문법도 가능하다.

```
val 조건 = 1

val 변수 = if(조건 == 1){
    3
} else {
    5
}
```

If문의 조건이 참이면 변수에 3을 대입하고 거짓이면 5를 대입한다.



# for

- Kotlin의 for문은 범위를 만들어서 변수에 하나씩 대입해서 반복 한다.



1~10까지의 범위 만들기

```
for(i in 1..10){  
    println("${i}번째 반복입니다.")  
}
```

## 실행 결과

1번째 반복입니다.  
2번째 반복입니다.  
3번째 반복입니다.  
4번째 반복입니다.  
5번째 반복입니다.  
6번째 반복입니다.  
7번째 반복입니다.  
8번째 반복입니다.  
9번째 반복입니다.  
10번째 반복입니다.

# 범위

- 기본 범위 만들기



```
val 범위 = 1..10
```

- 마지막 요소는 포함하지 않는 범위 만들기



```
val 범위 = 1 until 10
```

- 범위를 만들 때 수들의 간격 주기



```
val 범위 = 1 .. 10 step 2
```

- 내림차순 범위 만들기



```
val 범위 = 10 downTo 1
```

# for

- 범위가 아니라 리스트 또는 배열을 사용할 수 있다.



```
val 동아리_리스트 = listOf<String>(
    "EDCAN", "App:ple Pi", "IWOP",
    "RG", "Zer0pen", "AnA"
)

for (i in 동아리_리스트)
    println(i)
```

## 실행 결과

EDCAN

App:ple Pi

IWOP

RG

Zer0pen

AnA

# while문

- C언어의 while문과 동일

