



코틀린 변수 / 자료형

App:ple 코틀린 1차시 | 강이자 최재혁

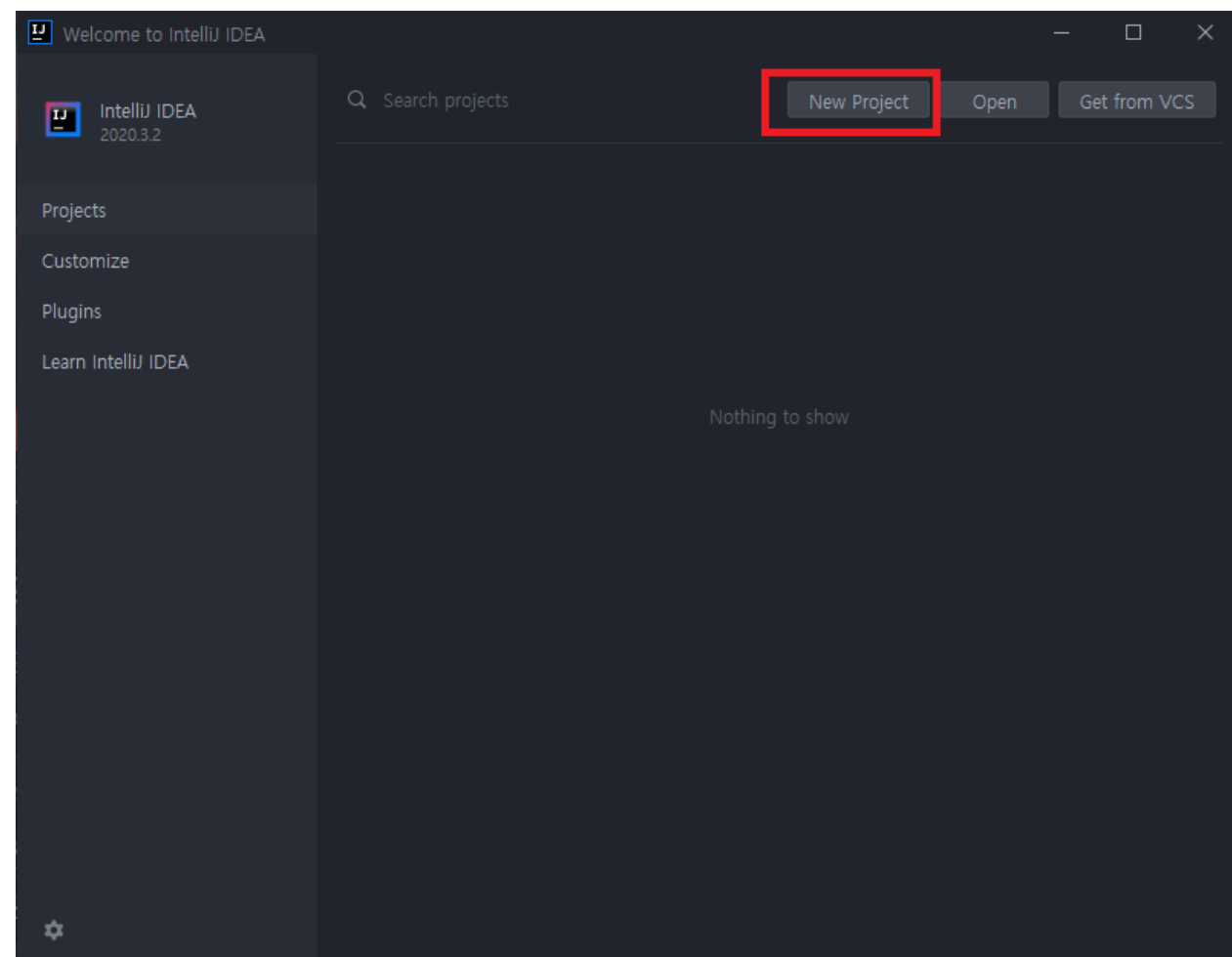
프로젝 생성 & 단축키

인텔리제이 프로젝트를 생성하는 방법과
알아 두면 간편하고 쓸 곳이 있을 단축키들을 알아봅시다

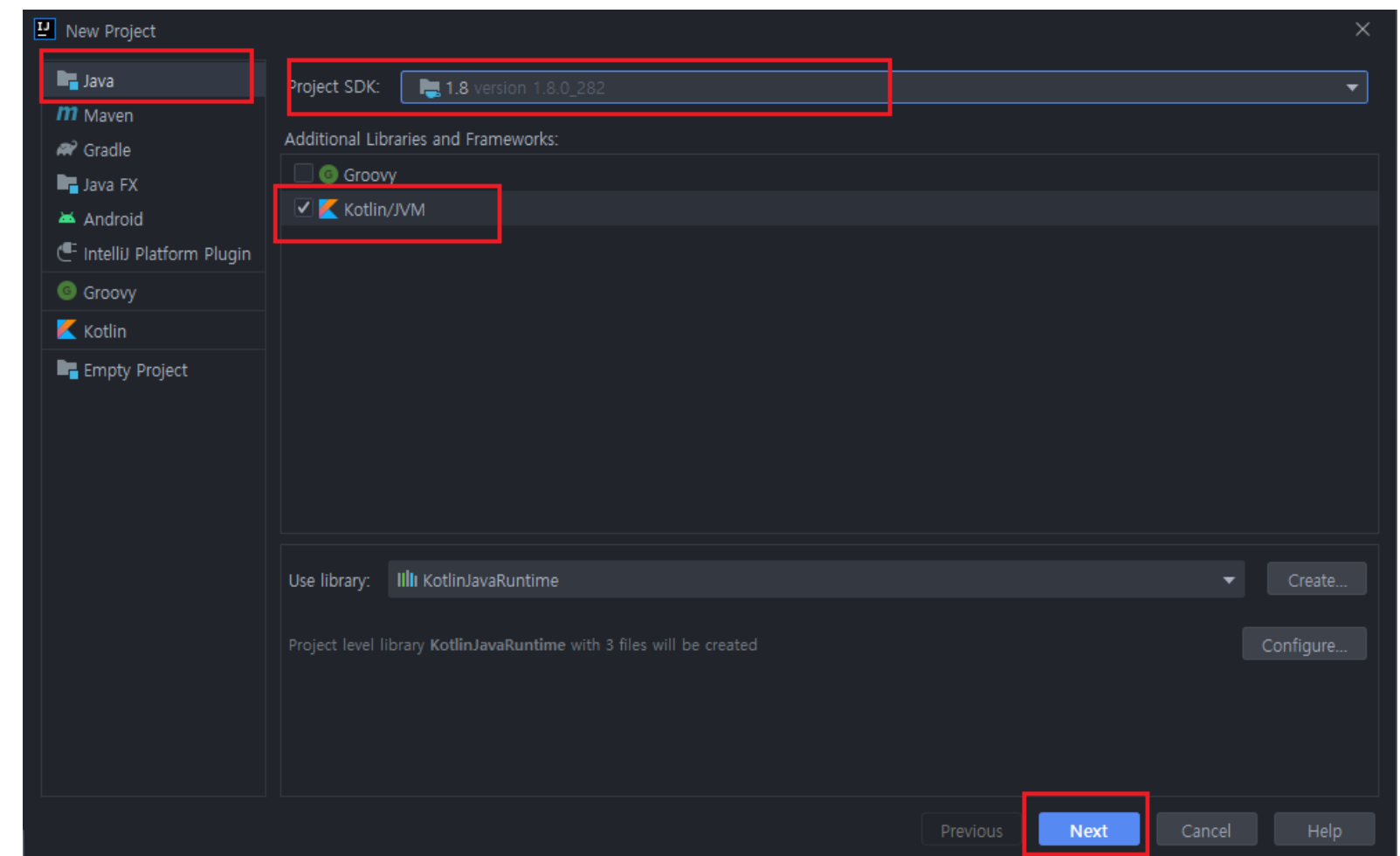
자료 출처 <https://gogigood.tistory.com/9>
설치 시 참고 <https://goddaehee.tistory.com/195>

<https://www.jetbrains.com/idea/download/#section=windows>

위 사이트에서, 인텔리제이를 다운로드를 받아주세요.



인텔리제이 실행 후, New Project 클릭

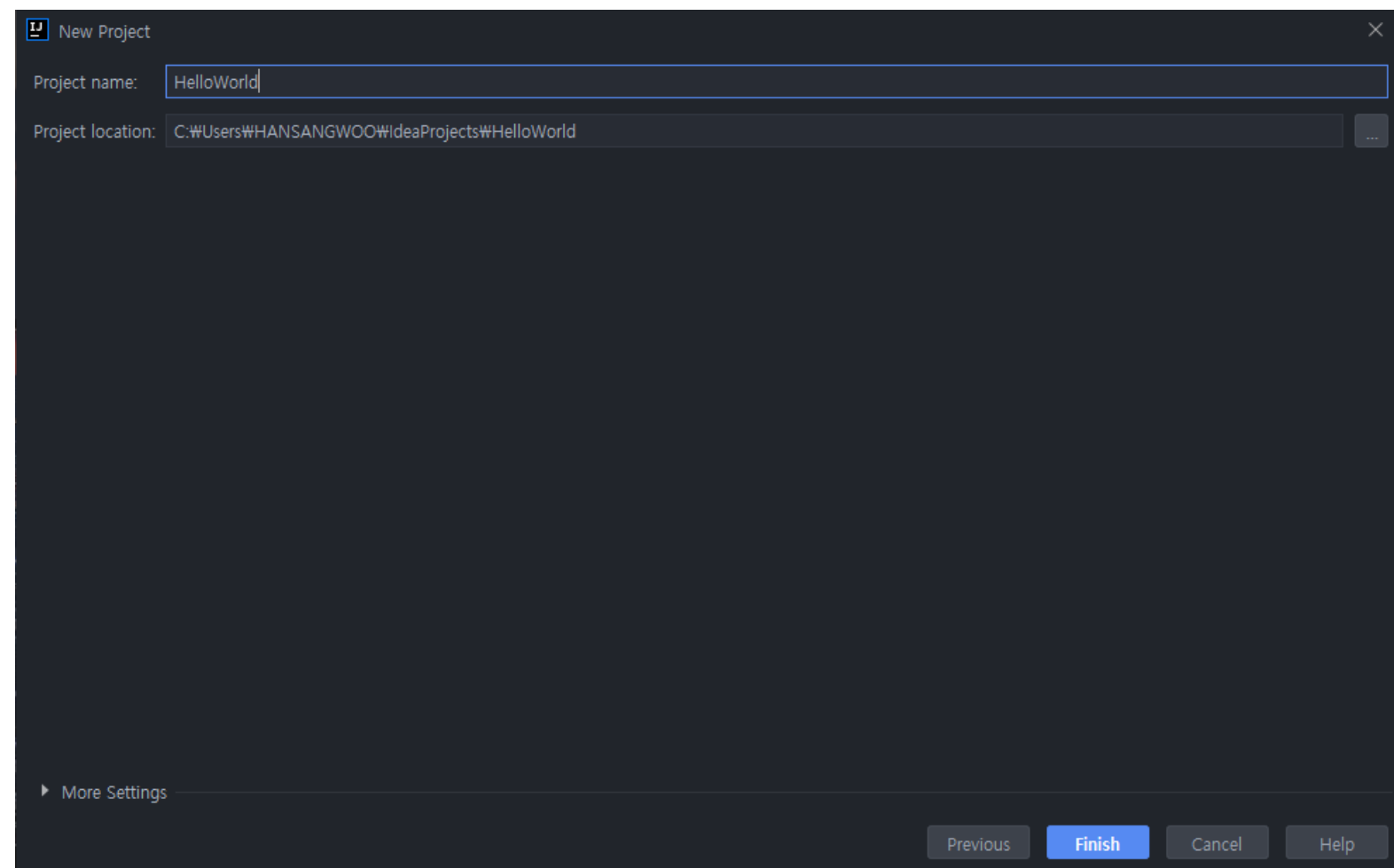


JAVA -> Kotlin/JVM -> Next 클릭

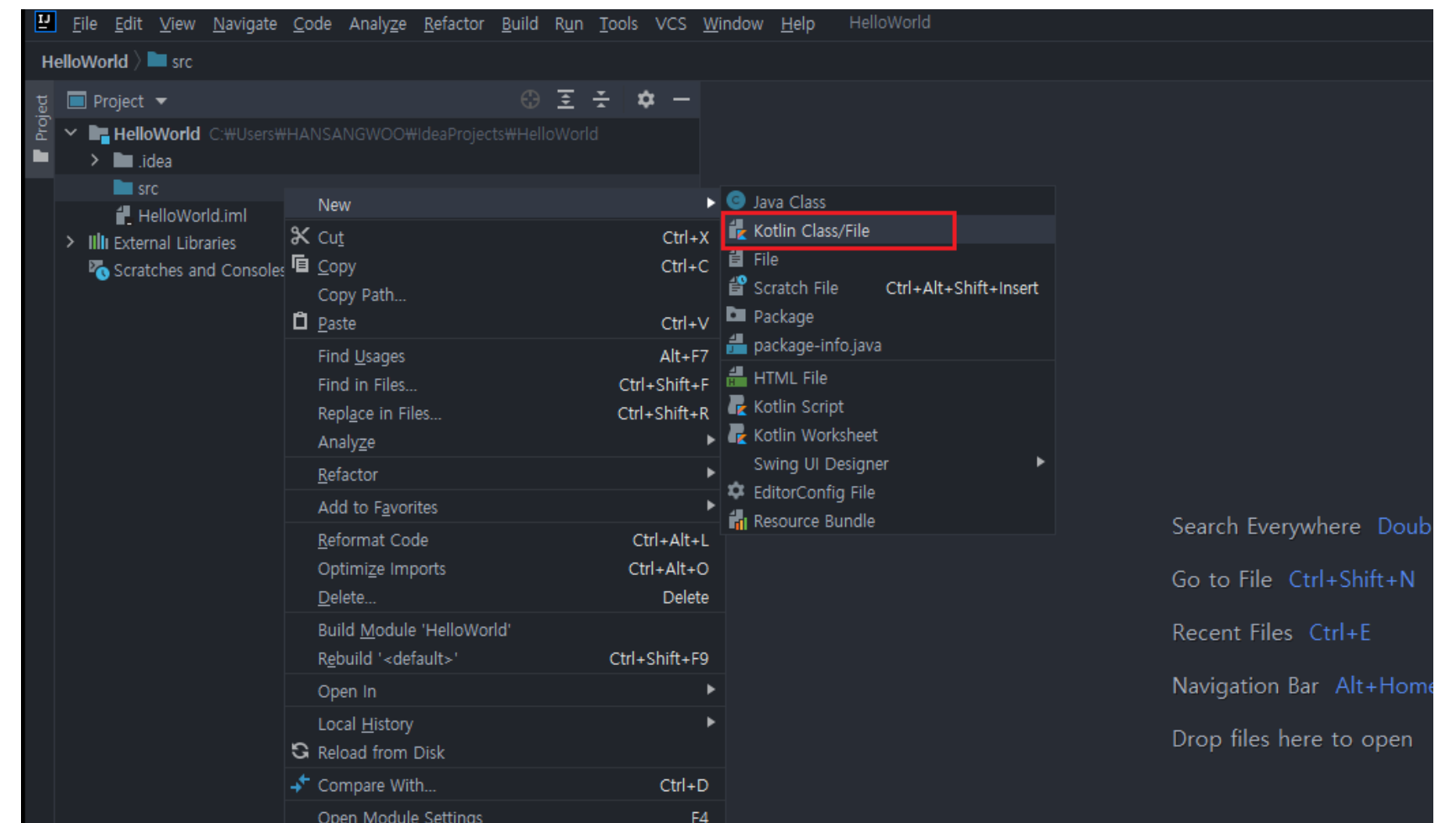
자료 출처 <https://gogigood.tistory.com/9>
설치 시 참고 <https://goddaehee.tistory.com/195>

<https://www.jetbrains.com/idea/download/#section=windows>

위 사이트에서, 인텔리제이를 다운로드를 받아주세요.



프로젝트 이름 설정 후 Finish 버튼 클릭

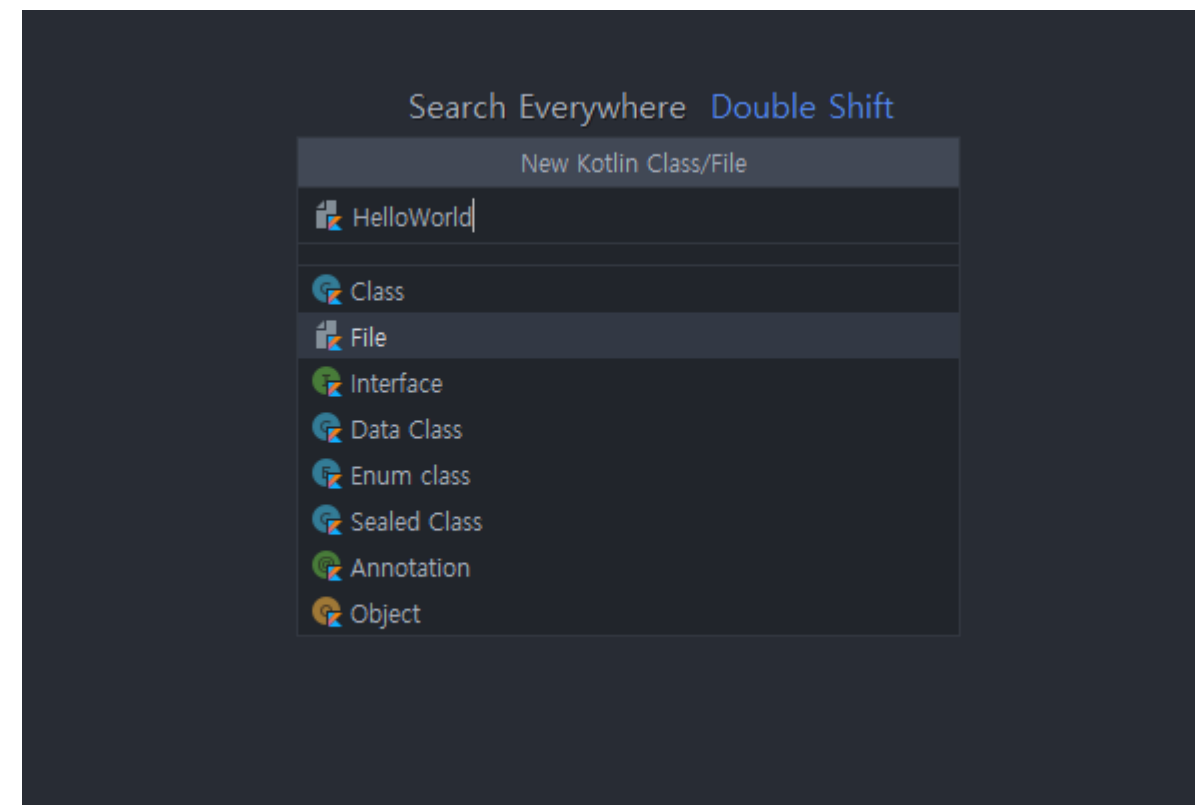


Src에 오른쪽 마우스 -> New -> Kotlin File

자료 출처 <https://gogigood.tistory.com/9>
설치 시 참고 <https://goddaehee.tistory.com/195>

<https://www.jetbrains.com/idea/download/#section=windows>

위 사이트에서, 인텔리제이를 다운로드를 받아주세요.



파일 이름을 설정한 뒤, File 클릭 후 엔터

만약, 설치가 되지 않을 경우에는 아래 사이트를 이용하면 됩니다
<https://play.kotlinlang.org/>

단축키	설명
Ctrl + Shift + Space	적합한 코드 자동완성을 추천함
Shift + Shift	파일, 클래스, 설정 등 키워드에 관련된 모든 것을 검색함
Alt + Enter	퀵픽스 제안 (오류가 발견됐을 시 문제 수정 제안 / 발견이 안되었을 때도 사용 가능함)
F2	오류와 경고 사이를 이동할 때 다음 오류나 경고로 이동시켜줌
Shift + F6	변수명, 함수명이 같은 것들을 한 번에 바꿀 수 있음
Ctrl + E	최근 연 파일 목록을 검색함
Ctrl + /	줄 위에서 단축키를 누르면 주석이 처리됨 (이미 처리된 경우 해제됨)
Shift + F10 / F9	프로그램 실행(F10) / 디버깅(F9)

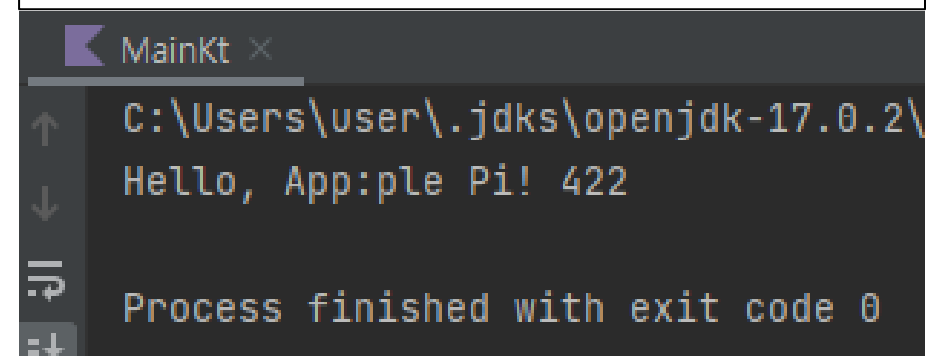
입력 / 출력

모든 언어의 기초라고 불리는
Hello World를 찍고 시작하도록 합시다

01. 코틀린에서의 출력문

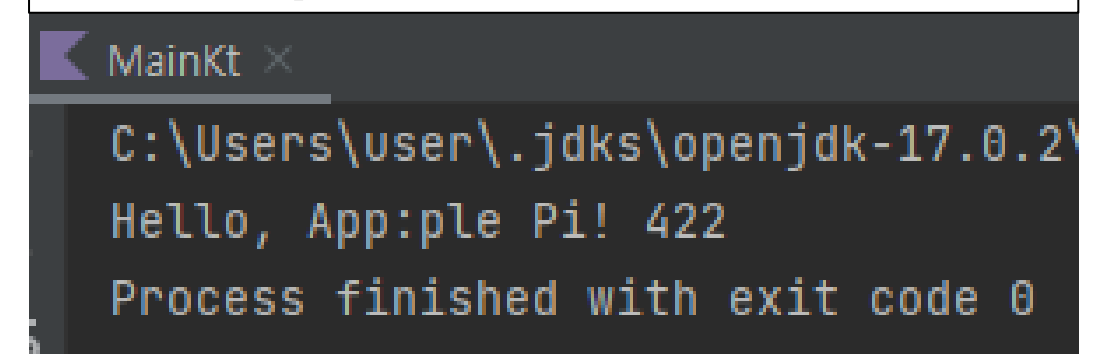
```
fun main() {  
    var a = "Hello"  
    var b = 400  
    var c = 22  
    println("$a, App:ple Pi! ${b+c}")  
}
```

[println을 사용할 때]



```
MainKt x  
C:\Users\user\.jdk\openjdk-17.0.2\  
Hello, App:ple Pi! 422  
Process finished with exit code 0
```

[print를 사용할 때]



```
MainKt x  
C:\Users\user\.jdk\openjdk-17.0.2\  
Hello, App:ple Pi! 422  
Process finished with exit code 0
```

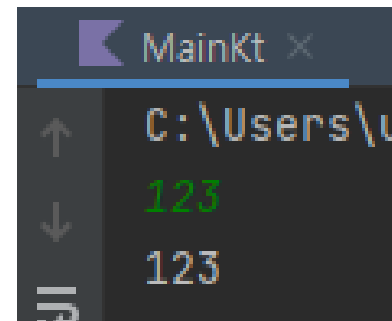
println("\$a, App:ple Pi! \${b+c}")

Print를 사용할 경우 문장 끝에 \n 미포함,
Println을 사용할 경우 문장 끝에 \n 포함

Print의 문자열 안에서 변수를 출력하기 위해서는 '문자열 템플릿' 이 필요한데,
사용하는 방법으로는 \${변수명}을 문자열 안에 넣어주면 된다

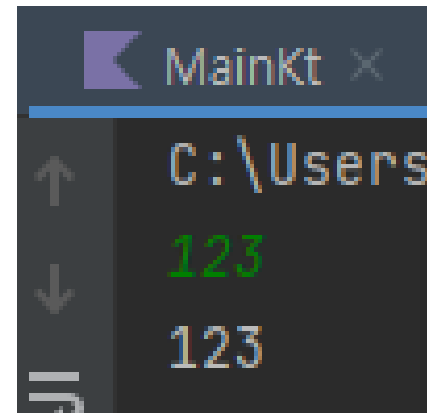
02. 코틀린에서의 입력문

```
fun main() {  
    var a = readLine()  
    println(a)  
}
```



[01] readLine() 을 이용하여 입력하는 방법
모든 문자열을 String(문자열)로 받기에,
숫자로 입력을 받아주기 위해서는, 숫자로 변환을 해주어야 한다

```
import java.util.Scanner  
fun main() {  
    val input = Scanner(System.`in`)  
    var a = input.nextInt()  
    println(a)  
}
```



[02] Scanner를 이용하는 방법
자바의 스캐너를 이용하는 형식으로, 자바와 동일 형식의
Import java.util.* 를 추가 후 스캐너를 사용한다
왼쪽의 예시는, 자바처럼 변수를 만들어서 사용하는 것이다

변수 & 자료형

코틀린에서의 변수는 무엇이고,
다른 언어와 무엇이 다른 지 살펴봅시다

01. C언어에서의 변수는

자료형 변수명 = 값 할당;

자료의 형태에 따라, 사용했던 자료형이 달랐어

형태	이름	크기 (sizeof)	출력 가능한 값
정수형	char	1 byte	-128 이상 +127 이하
	short	2 byte	-32,768 이상 +32,767 이하
	Int (기본)	4 byte	-2,147,483,648 이상 2,147,483,647 이하
	long	4 byte	-2,147,483,648 이상 2,147,483,647 이하
	long long	8 byte	-9,223,372,036,854,775,808 이상 +9,223,372,036,854,775,807 이하
실수형	float	4 byte	$\pm 3.4 \times 10^{-37}$ 이상 $\pm 3.4 \times 10^{37}$ 이하
	double (기본)	8 byte	$\pm 1.7 \times 10^{-307}$ 이상 $\pm 1.7 \times 10^{307}$ 이하
	long double	8 byte 이상	double 이상의 표현범위

변수 명에서는, 명명 규칙을 따라줘야 했었지

- 1) 변수의 이름은 알파벳, 숫자, 언더바로 구성한다
- 2) C언어는 대소문자를 구분한다 (x와 X는 다름)
- 3) 변수의 이름은 숫자로 시작할 수 없다
- 4) 변수의 이름으로 키워드를 사용할 수 없다
- 5) 변수의 이름 사이에 공백을 넣을 수 없다

변수에 값을 바로 할당 해주어도 되고,
바로 할당하지 않고 나중에 할당 해도 상관 없어!

02. 구성 요소 차이 알아보기



변경여부
변경여부

변수명 : 자료형 = 값 할당
변수명 : 자료형

값 할당을 할 경우
자료형을 적을 필요가 없음

변수 선언만 할 경우
자료형을 반드시 적어야 함

C에서는 존재 하지 않는
"변경여부" 존재

가변 - var (Changeable)	값을 계속 할당이 가능함
불변 - val (Unchangeable)	값을 한 번만 할당이 가능함 (값을 재할당하려고 했을 시, 에러)

02. 구성 요소 차이 알아보기

- 1) 변수의 이름은 알파벳, 숫자, 언더바로 구성한다
- 2) C언어는 대소문자를 구분한다 (x와 X는 다름)
- 3) 변수의 이름은 숫자로 시작할 수 없다
- 4) 변수의 이름으로 키워드를 사용할 수 없다
- 5) 변수의 이름 사이에 공백을 넣을 수 없다

코틀린의 경우, 변수 선언 시 카멜 표기법을 주로 이용함

1. 카멜 표기법 (ex. camelCase)

각 단어의 첫 문자를 대문자로 적는다
단, 가장 첫 번째 글자는 소문자로 적는다

2. 스네이크 표기법 (ex. snake_case)

모든 단어를 소문자로 사용하고,
각 단어 사이에 _ (언더바)를 넣어서 적는다

3. 파스칼 표기법 (ex. PascalCase)

카멜 표기법과 같지만,
첫 번째 글자까지 대문자를 사용해서 적는다

03. 변경 여부에 따른 변수 선언



변경여부 변수명 : 자료형 = 값 할당

추후 변수를 바꿀 것인지?

가변 - var Changeable	값을 계속 할당이 가능함
불변 - val Unchangeable	값을 한 번만 할당이 가능함 (값을 재할당하려고 했을 시, 에러)

```
fun main() {  
    var a = 5  
    a = 3  
}
```

a 값은 정상적으로 3으로 변경됨
(에러 없이 정상적으로 변경)

```
fun main() {  
    val b = 5  
    b = 3  
}
```

Val cannot be reassigned
Val은 재할당이 불가하다는 오류 발생

04. 자료형에 따른 변수 선언



변경여부 변수명 : 자료형 = 값 할당

형태	이름	크기 (sizeof)	사용 예시
정수형	Byte	1 byte	var num1 : Byte = 123
	short	2 byte	var num2 : Short = 123
	Int (기본)	4 byte	var num3 = 123 (10진수를 표기하려면 0x를, 2진수를 표기하려면 0b를 앞에 붙이기)
	long	8 byte	var num4 : Long = 123L (L을 통해 더 큰 메모리를 가진 변수임을 알려줘야 함)
실수형	float	4 byte	var num5 : Float = 123.5f
	double (기본)	8 byte	var num6 = 123.4
문자형	Char (한 개의 문자)		var str1 = 'a'
문자열	String (한 개 이상의 문자)		var str2 = "String"
논리형	Boolean		var bool = true

05. 형변환



변경여부 변수명 : 자료형 = 값 할당

```
fun main() {  
    var num1 : Int = 123  
    var num2 : Double = num1  
}
```

Type mismatch: inferred type is Int but Double was expected
(유형이 일치하지 않음 : 추론된 형식은 Int이지만, 더블 형식이 예상됨)

```
fun main() {  
    var num1 : Int = 123  
    var num2 : Double = num1.toDouble()  
}
```

.to자료형 을 이용하여, 자료형을 변환할 수 있음
(.toInt(), .toDouble(), .toByte(), .toFloat(), toChar() 등)

연산자

코틀린에서의 연산자는 무엇이고,
다른 언어와 무엇이 다른 지 살펴봅시다

01. 코틀린에서의 연산자

코틀린에서의 연산자는, C언어의 연산자와 매우 유사합니다

++, -- 증감 연산자	값을 하나씩 증가 혹은 감소하는 단항 연산자
-, *, /, %, +, - 산술 연산자	사칙 연산을 할 수 있는 연산자
<, <=, >, >=, ==, != 관계(비교) 연산자	피연산자의 대소 관계를 나타낼 수 있는 연산자 값은 Boolean으로 나오며, true 혹은 false로 이야기함
<<, >>, >>> 쉬프트 연산자	비트를 좌측 혹은 우측으로 이동할 수 있는 연산자 (>>> 는 부호비트를 포함하여 오른쪽으로 이동함)
~, &, ^, 비트 연산자	비트의 관계를 나타낼 수 있는 연산자
&&, 논리 연산자	피연산자의 논리적 관계를 나타낼 수 있는 연산자
=, +=, -=, *=, /=, %= 대입 연산자	대입 연산자는 우변에 있는 값/결과를 좌측 변수에 저장시키는 연산자 복합 대입 연산자는 대입 연산자와 산술 연산자를 합쳐 놓은 연산자
변수 is 자료형, 변수 !is 자료형	is) 변수의 자료형이 맞다면 True, 틀리다면 False !is) is의 반대 연산자로, 반대로 작동하는 연산자임

Null Safety

코틀린에서 빈 값이 들어가게 하려면
어떻게 해야 하는 지 알아보시다

Null이란 무엇일까?

코틀린에서는 변수 사용 시, 반드시 값이 할당되어 있어야 한다
-> 기본적으로 null값이 될 수 없음

Null값을 허용하려면, Safe-call 연산자를 사용하여
Null값을 할당할 수 있음



0



null



undefined

01. safe-call 연산자

```
fun main() {  
    var a : String = null  
    println(a)  
}
```

이후 출력문으로 왼쪽의 선언을 출력 할 시
Null can not be a value of a non-null type String
(널은 널 타입이 아닌 문자열의 값이 될 수 없음) 오류가 발생하게 됩니다.



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

자료형 뒤에 ? (safe-call) 연산자는 null값을 허용함

02. Null 값인지 판별하기

```
fun main() {  
    var a : String? = null  
  
    if (a != null){  
        println("$a")  
    }  
}
```

```
MainKt x  
C:\Users\user\...  
Process finished
```

```
fun main() {  
    var a = 30  
    if(a != null){  
        print("$a")  
    }  
}
```

```
MainKt x  
C:\Users\use...  
30  
Process fini
```

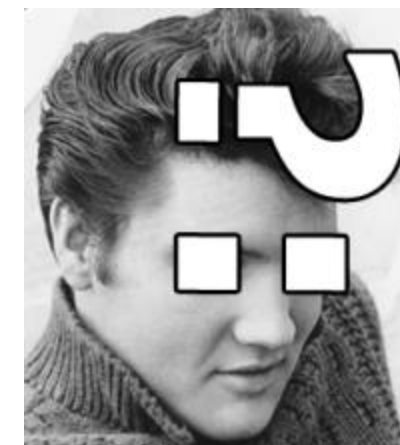
A라는 변수가 null이 아니라면,
a의 값을 출력함

```
fun main() {  
    nullcheck("문자열")  
    nullcheck(null)  
}
```

```
fun nullcheck(str : String?){  
    val checkResult : String = str ?: "널값이 들어왔음"  
    println("체크 결과 : $checkResult")  
}
```

```
MainKt x  
C:\Users\user\jdk\openjdk...  
체크 결과 : 문자열  
체크 결과 : 널값이 들어왔음
```

?: 를 기준으로 null이 들어왔다면 오른쪽을 실행함



?: 는 elvis operator라고 부름

과제

배운 내용을 활용하여
과제를 수행 해봅시다

01. 과제 제출 방법 안내

오늘의 과제는, 계산기를 만들어서 제출하는 것입니다!
조건은 다음페이지에 남겨져 있으며,
제출 기한은 04/05 자정까지

구글 폼 링크

<https://forms.gle/GKPpugT6acGZ1s4o7>

과제에 대한 질문이 있을 경우
디스코드 : 재혁 #2293, 인스타 : @jaehyeok3017,
애플파이 12,13기 단체 카톡을 이용하여 질문을 하시면 됩니다.

02. 과제 요구사항

계산기 만들기
(과제 제출 기한 : 04/05 자정까지)

기능이 일부 구현 되지 않아도 되니
자신이 할 수 있는 **최대한으로 시도 해서 제출해주세요**



과제 요구 사항

01. 사용자가 프로그램을 실행시켰을 때,

- ★ +(더하기), -(빼기), *(곱하기), /(나누기)를 입력하여 연산의 종류를 입력 받음
- ★ 2개의 수를 입력 받음 (따로 입력 받아도 상관 X)

02. 사용자가 입력했을 때,

- ★ 연산 종류에 따라 연산을 수행한 뒤, 연산 결과를 나타낸다
- ★ 연산 결과를 나타낼 때는, [수1] [연산자] [수2] = [결과값] 으로 한다. (ex $1 * 2 = 2$)



코틀린 1차시 끝!

위 페이지에서 제시하였던
제출 기한까지 반드시 제출해주세요!