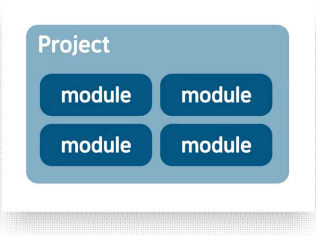
#12 코틀린의 기본 프로젝트 구조

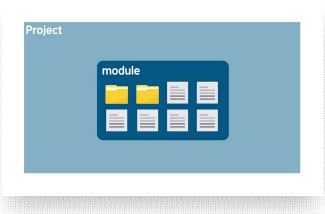
프로젝트

우리가 코틀린으로 어플리케이션을 짤 때 관련한 모든 내용을 담는 '큰 틀'

모듈



하나의 프로젝트는 여러 개의 모듈로 이루어질 수 있다.
이 모듈은 '직접' 만들 수도 있고 필요한 기능을 이미 구현해 둔 '라이브러리 모듈'을 가져와 붙일 수도 있기 때문에 굉장히 편리한 기능 단위이다.



모듈 안에는 다수의 폴더와 파일이 들어갈 수가 있다. 여기에는 코틀린 코드파일(.kt) 뿐만 아니라 모듈과 관련된 설정 및 리소스 파일 등도 포함될 수 있다.

이렇게 프로젝트, 모듈, 폴더 및 파일이 실제 파일 시스템에 기반한 물리적인 구조를 담당한다고 하면

패키지

논리적인 구조로는 **패키지**라는 것이 있는데, 패키지는 개발 시에 소스 코드의 '소속'을 지정하기 위한 논리적 단위이다. 코드를 작성할 때에는 코드내에서 사용하는 이름이 용도에 따라 서로 충돌하지 않도록 유니크한 패키지 이름을 지어주는 것이 좋다.

일반적으로 패키지 이름을 지을 때는 개발한 회사가 가진 도메인을 거꾸로 배열하고 그 뒤에 프로젝트 명을 붙인 후 그 아래에 기능별로 세분화하는 방식으로 짓는다.

서비스 도메인	패키지 명
youtube.com	com.youtube

com.youtube.dimo
com.youtube.dimo.base
com.youtube.dimo.kotlin
com.youtube.dimo.kotlin.android
com.youtube.dimo.talk

코드 파일들을 패키지에 넣는 방법

```
package com.youtube.dimo
fun main(){
}
```

코드 파일 맨 윗줄에 **package**를 적고 패키지 이름을 써준다.
*패키지를 명시하지 않으면 자동으로 **default** 패키지로 묶이게 되는 것도 알아두자

코틀린은 자바와 달리 폴더 구조와 패키지 명을 일치시키지 않아도 된다. 단순히 파일 상단에 패키지만 명시해 주면 컴파일러가 알아서 처리해준다.

그런데 같은 패키지 내에서는 변수, 함수, 클래스를 공유할 수 있지만 패키지가 다르면 그냥 쓸 수는 없다. 바로 '**import**' 작업을 해야 한다.


```
package com.youtube.dimo
import com.youtube.dimo.base
fun main(){
    println("프로젝트의 구조는 재밌어")
}
```

코드 파일 내에서 패키지 선언 바로 아래에 'import'를 쓰고 사용할 외부 패키지 이름을 써주면 다른 패키지의 변수나 함수, 클래스 등을 그대로 사용할 수 있다. 이때, 이름이 중복되는 요소가 있다면 패키지명을 포함한 풀 네임을 명시해야 한다는 점을 주의하자

코틀린은 클래스명과 파일명이 일치하지 않아도 되며 하나의 파일에 '여러 개의 클래스'를 넣어도 알아서 컴파일이 가능하다. 이는 파일이나 폴더로 구분하지 않고 파일 내에 있는 package 키워드를 기준으로 구분하기 때문이다.