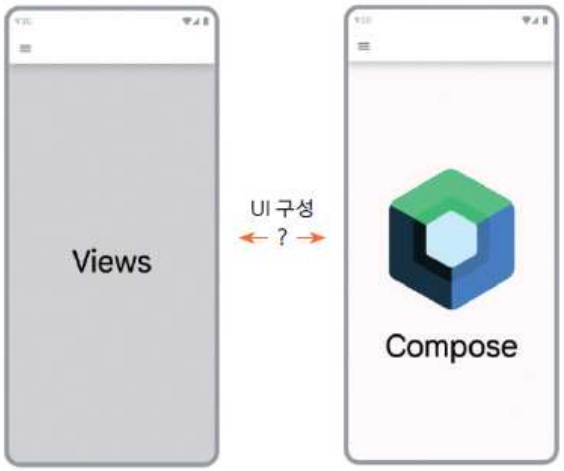
04-1. Overview

Compose

22-1 컴포즈 이해하기

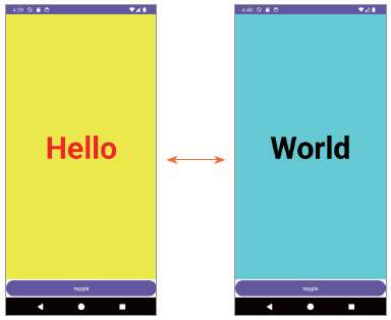
컴포즈란?

- 컴포즈compose란 제트팩에서 제공하는 기술로 앱의 화면을 구성하는 방법
- 뷰와 컴포즈는 상호 연동은 가능
- 컴포즈의 사용 목적은 **선언형 UI 프로그래밍**과 **상태 관리**



22-1 컴포즈 이해하기

- **선언형 UI 프로그래밍이란?**
  - 선언형 UI 프로그래밍Declarative UI Programming은 명령형 UI 프로그래밍Imperative UI Programming과 반대되는 개념
  - 명령형 UI 프로그래밍은 개발자 코드에서 UI 구성과 관련된 모든 코드를 작성하는 방식
  - 명령형 UI로 화면을 구성하게 되면 화면 출력을 위한 개발자 코드가 길어질 수 밖에 없으며 개발자는 화면과 관련된 많은 API를 알고 있어야 합니다.



• 레이아웃 XML 파일(명령형 UI 프로그래밍으로 작성한 예)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Hello"
        android:textSize="80sp"
        android:gravity="center"
        android:textColor="#FF0000"
        android:background="#FFFF00"
        android:textStyle="bold"/>
```

• 버튼 클릭 시 화면 변경 처리(명령형 UI 프로그래밍으로 작성한 예)

```
binding.button.setOnClickListener {
    binding.textView.run {
        if(text === "Hello"){
            text = "World"
            setTextColor(Color.BLACK)
            setBackgroundColor(Color.CYAN)
        }else {
            text = "Hello"
            setTextColor(Color.RED)
            setBackgroundColor(Color.YELLOW)
        }
    }
}
```

22-1 컴포즈 이해하기

- **선언형 UI 프로그래밍이란?**
  - 선언형 UI 프로그래밍은 화면에 어떻게 출력되어야 한다는 정보만 선언하는 방식

• 버튼 클릭 시 화면 변경 처리(선언형 UI 프로그래밍으로 작성한 예)

```
var textState by remember { mutableStateOf("Hello") }
var textColorState by remember { mutableStateOf(Color.Red) }
var textBackgroundColorState by remember { mutableStateOf(Color.Yellow) }
Column {
    Text(
        textState,
        fontSize = 40.sp,
        fontWeight = FontWeight.Bold,
        color = textColorState,
        textAlign = TextAlign.Center,
        modifier = Modifier
            .fillMaxWidth()
            .weight(1f)
            .background(textBackgroundColorState)
            .wrapContentHeight(align = Alignment.CenterVertically)
    )
    Button(modifier = Modifier.fillMaxWidth(), onClick = {
        if (textState === "Hello") {
            textState = "World"
            textColorState = Color.Black
            textBackgroundColorState = Color.Cyan
        } else {
            textState = "Hello"
            textColorState = Color.Red
            textBackgroundColorState = Color.Yellow
        }
    }) {
        Text("toggle")
    }
}
```

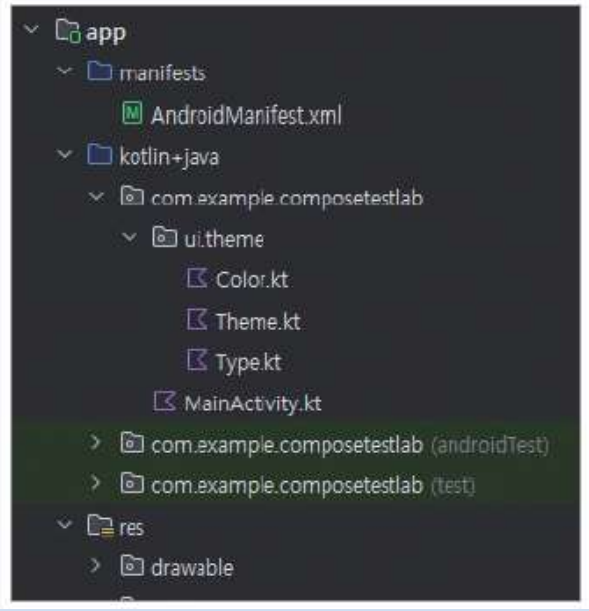
22-1 컴포즈 이해하기

- **상태 관리란?**
  - 상태는 데이터입니다
  - 상태는 화면에 출력되면서 다양한 이유(사용자 이벤트, 서버 네트워킹 등)에 의해 변경되고, 변경이 이뤄지면 화면 갱신이 되어야 하는 데이터를 의미
  - 정보가 변경되었다는 것을 컴포즈 에 인식시켜 컴포즈가 변경된 값으로 화면을 갱신

22-1 컴포즈 이해하기

컴포즈 프로젝트 구조

- 뷰를 사용하는 방식과 비교해 보면 전체적으로 디렉터리 파일 구성은 비슷하지만 컴포즈는 화면을 레이아웃 XML로 구성하지 않아 res 폴더 안에 layout 폴더가 만들어지지 않습니다.
- ui.theme 패키지에 앱의 테마를 선언하기 위한 코틀린 파일들이 자동으로 만들어집니다.



22-1 컴포즈 이해하기

build.gradle.kts(프로젝트 수준)

- 컴포즈는 코틀린으로 개발해야 하므로 자동으로 코틀린과 관련된 플러그인이 추가

• 자동으로 추가된 코틀린 관련 플러그인 확인

```
plugins {
    alias(libs.plugins.android.application) apply false
    alias(libs.plugins.kotlin.android) apply false
    alias(libs.plugins.kotlin.compose) apply false
}
```

22-1 컴포즈 이해하기

build.gradle.kts(모듈 수준)

- 컴포즈를 사용한다는 선언이 android 영역에 자동으로 추가
- dependencies 부분에 컴포즈를 사용하기 위한 라이브러리가 자동으로 추가

• 컴포즈 사용 선언과 컴포즈 관련 라이브러리 추가

```
android {

    buildFeatures {
        compose = true
    }
}

dependencies {

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.activity.compose)
    implementation(platform(libs.androidx.compose.bom))
    implementation(libs.androidx.ui)
    implementation(libs.androidx.ui.graphics)
    implementation(libs.androidx.ui.tooling.preview)
    implementation(libs.androidx.material3)
    (... 생략 ...)
}
```

22-1 컴포즈 이해하기

build.gradle.kts(모듈 수준)

- libs.androidx.lifecycle.runtime.ktx: 제트팩의 Lifecycle Aware Component 부분을 지원하기 위한 라이브러리입니다. lifecycle로 시작하는 많은 라이브러리들이 있는데 그중 lifecycle-runtime은 ViewModel, LiveData가 포함되지 않은 라이브러리이며 코어 Lifecycle Aware Component 부분을 지원하기 위한 라이브러리입니다.
- libs.androidx.activity.compose: activity로 시작하는 라이브러리들은 액티비티를 위한 다양한 기능을 지원합니다. 그중 activity.compose는 컴포즈를 이용하는 액티비티를 지원하기 위한 라이브러리입니다.
- libs.androidx.ui: 사이즈, 배치 등 기본적으로 화면을 구성할 수 있는 요소들을 제공하는 라이브러 리입니다.
- libs.androidx.ui.graphics: 다양한 그래픽 기능을 제공하는 라이브러리입니다.
- libs.androidx.ui.tooling.preview: 컴포즈로 개발한 코드 미리보기 기능을 제공하는 라이브 러리입니다.
- libs.androidx.material3: 머티리얼 디자인이 적용된 화면 구성을 지원하는 라이브러리입니다.

22-1 컴포즈 이해하기

MainActivity.kt

- 컴포즈를 이용하는 액티비티는 ComponentActivity를 상속
- 화면 출력을 setContent() 함수를 이용
- setContent()의 매개변수에 지정된 AndroidLabTheme(), Surface(), Greeting() 등은 컴포저블composable 함수라고 부르며 화면을 구성하는 역할
- 컴포즈로 프로그램을 작성한다는 것은 컴포저블 함수를 만드는 것을 의미
- 컴포저블 함수는 @Composable 어노테이션으로 선언되는 함수

• 자동으로 생성되는 메인 액티비티 확인

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            AndroidLabTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    Greeting(
                        name = "Android",
                        modifier = Modifier.padding(innerPadding)
```

• Greeting 컴포저블 함수 선언

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",
        modifier = modifier
    )
}
```

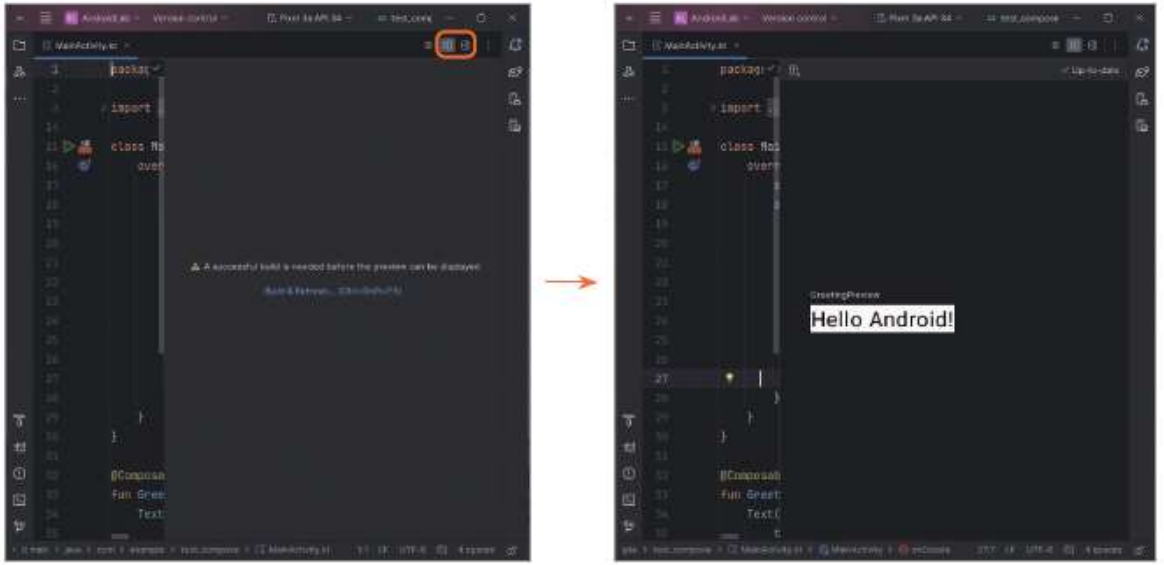
22-1 컴포즈 이해하기

MainActivity.kt

- GreetingPreview()는 @Preview 어노테이션으로 선언되며 실제 앱이 빌드되어 실행될 때의 화면 출력을 목적으로 하지 않고 안드로이드 스튜디오의 preview 화면에 출력될 내용을 표현하기 위한 컴포저블 함수

• GreetingPreview 컴포저블 함수 선언

```
@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    AndroidLabTheme {
        Greeting("Android~")
    }
}
```



22-1 컴포즈 이해하기

컴포저블 함수 좀 더 알기

- 컴포즈에서 가장 중요한 부분이 컴포저블 함수
- 이미 제공되는 Text(),Button() 같은 컴포저블 함수를 이용해 개발자가 원하는 화면을 컴포저블 함수로 선 언
- @Composable 어노테이션으로 선언
- 컴포저블 함수의 정보를 참조하고 이를 통해 화면을 구성
- 컴포저블 함수는 다른 컴포저블 함수 내에서만 호출
- 컴포저블 함수에서 일반 함 수 호출이 가능
- 컴포저블 함수는 리턴값을 가질 수 있기는 하지만 리턴값이 화 면 구성 정보로 사용되지 않았기 때문에 리턴값 자체가 의미가 없습니다.

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare