Topic for discussion:

Android Jetpack Compose

Umer Farooq Lead Android Engineer

What's Jetpack Compose?

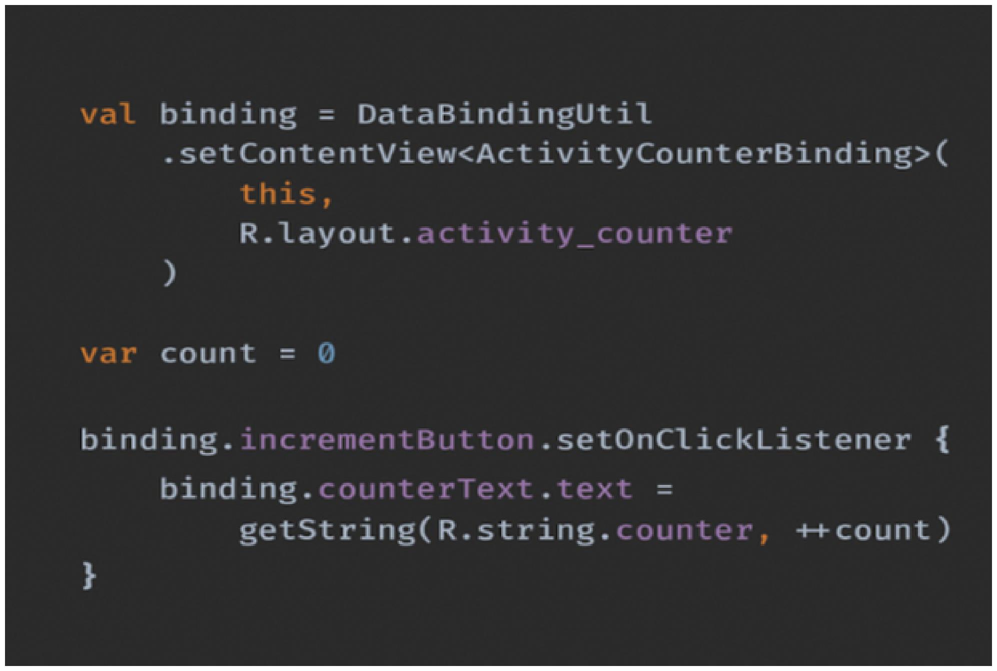
A declarative toolkit for building UI
that combines a reactive programming model with the conciseness and
ease of use of the Kotlin programming language.

Declarative vs Imperative Way

Declarative programming is “the act of programming in languages that conform to the mental model of the developer rather than the operational model of the machine.”

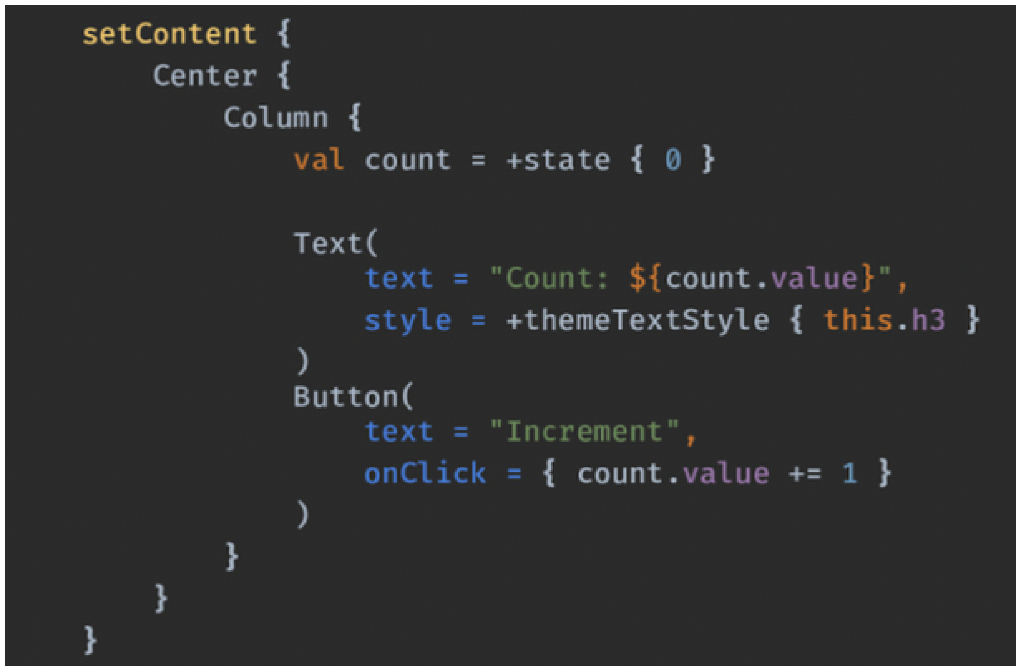
In computer science, declarative programming is a programming paradigm that expresses the logic of a computation without describing its control flow.

HOW



```kotlin
val binding = DataBindingUtil
    .setContentView<ActivityCounterBinding>(
        this,
        R.layout.activity_counter
    )

var count = 0

binding.incrementButton.setOnClickListener {
    binding.counterText.text =
        getString(R.string.counter, ++count)
}
```

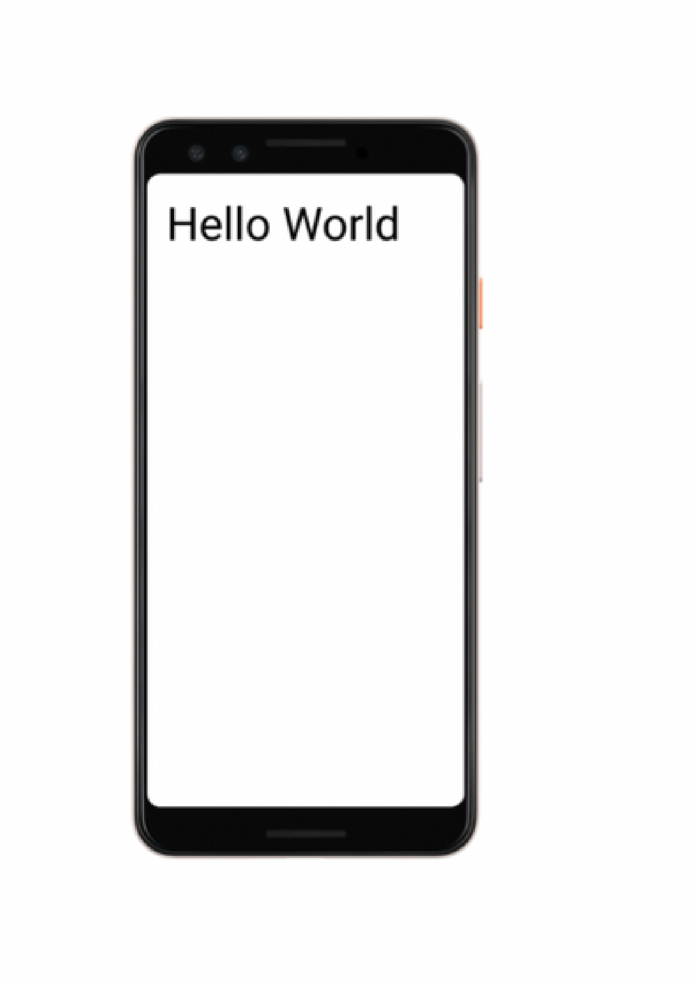
What

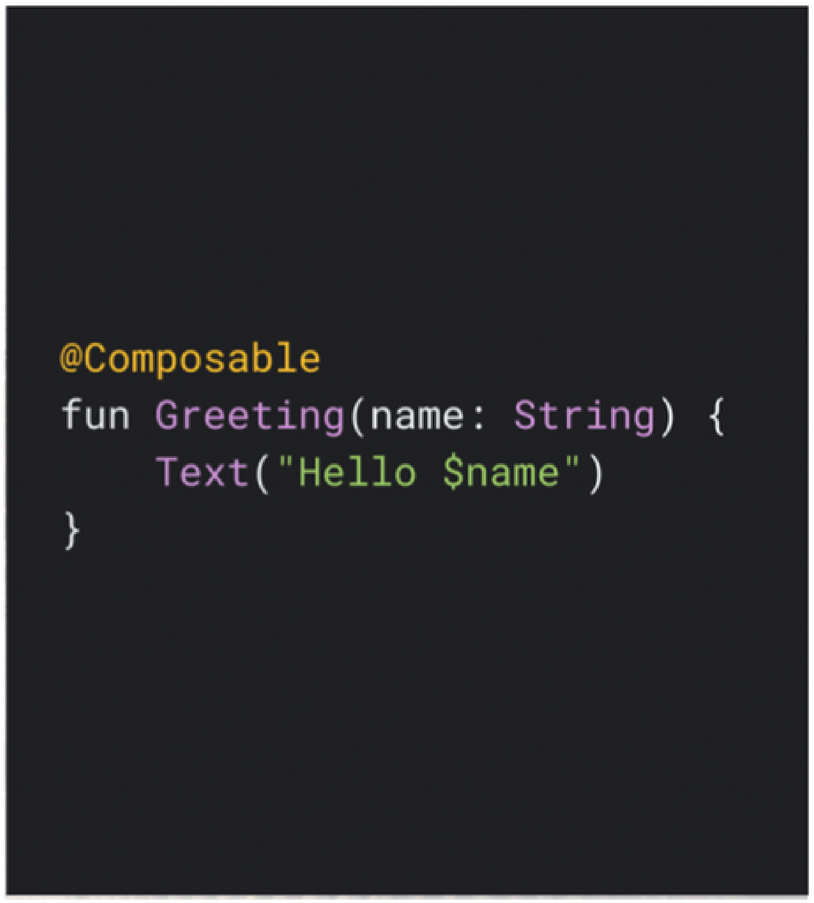


```kotlin
setContent {
    Center {
        Column {
            val count = +state { 0 }

            Text(
                text = "Count: ${count.value}",
                style = +themeTextStyle { this.h3 }
            )
            Button(
                text = "Increment",
                onClick = { count.value += 1 }
            )
        }
    }
}
```

A simple composable function

Using Compose, you can build your user interface by defining a set of composable functions that take in data and emit UI elements. A simple example is a Greeting widget, which takes in a String and emits a Text widget which displays a greeting message.





```kotlin
@Composable
fun Greeting(name: String) {
    Text("Hello $name")
}
```

Dynamic content

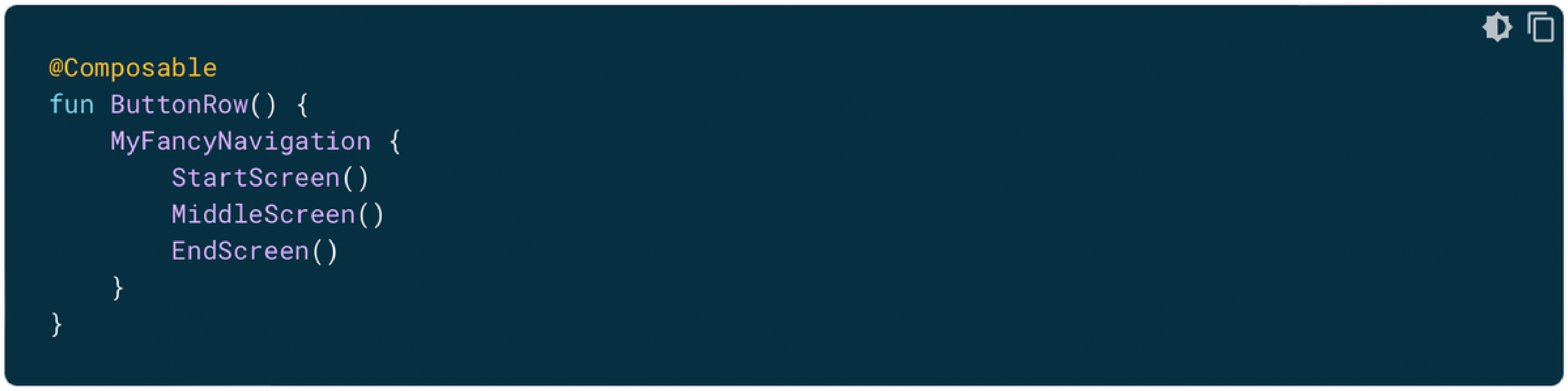
```kotlin
@Composable
fun Greeting(names: List<String>) {
    for (name in names) {
        Text("Hello $name")
    }
}
```

Recomposition

```kotlin
@Composable
fun ClickCounter(clicks: Int, onClick: () -> Unit) {
    Button(onClick = onClick) {
        Text("I've been clicked $clicks times")
    }
}
```

Composable functions can execute in any order

Suppose you have code like this to draw three screens in a tab layout:



```
@Composable
fun ButtonRow() {
    MyFancyNavigation {
        StartScreen()
        MiddleScreen()
        EndScreen()
    }
}
```

Why adopt Compose?

- Less code
- Intuitive
- Accelerate development
- Powerful

Adding Jetpack Compose to your app

STEPS:

- Android Gradle Plugin
- Configure Kotlin
- Configure Gradle
- Add Jetpack Compose toolkit dependencies
- Start creating Composable

Step 1 : Android Gradle Plugin

For the best experience developing with Jetpack Compose, you should download Android Studio, and configure the Android Gradle Plugin that corresponds to the version of Android Studio:

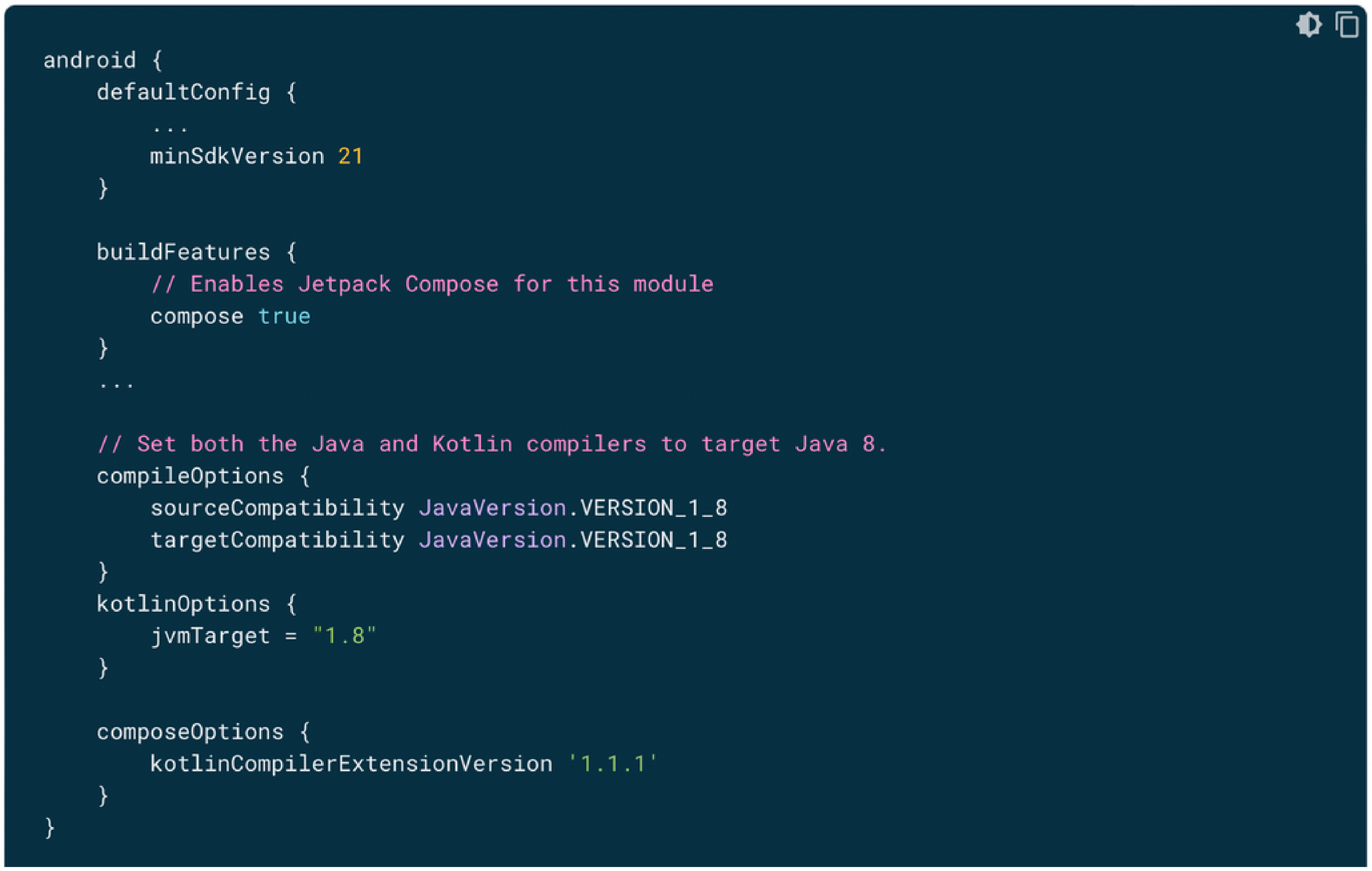
```
buildscript {
    ...
    dependencies {
        classpath "com.android.tools.build:gradle:7.0.0"
        ...
    }
}
```

Step 2 : Configure Kotlin

```
plugins {
    id 'kotlin-android'
}
```

Step 3 : Configure Gradle

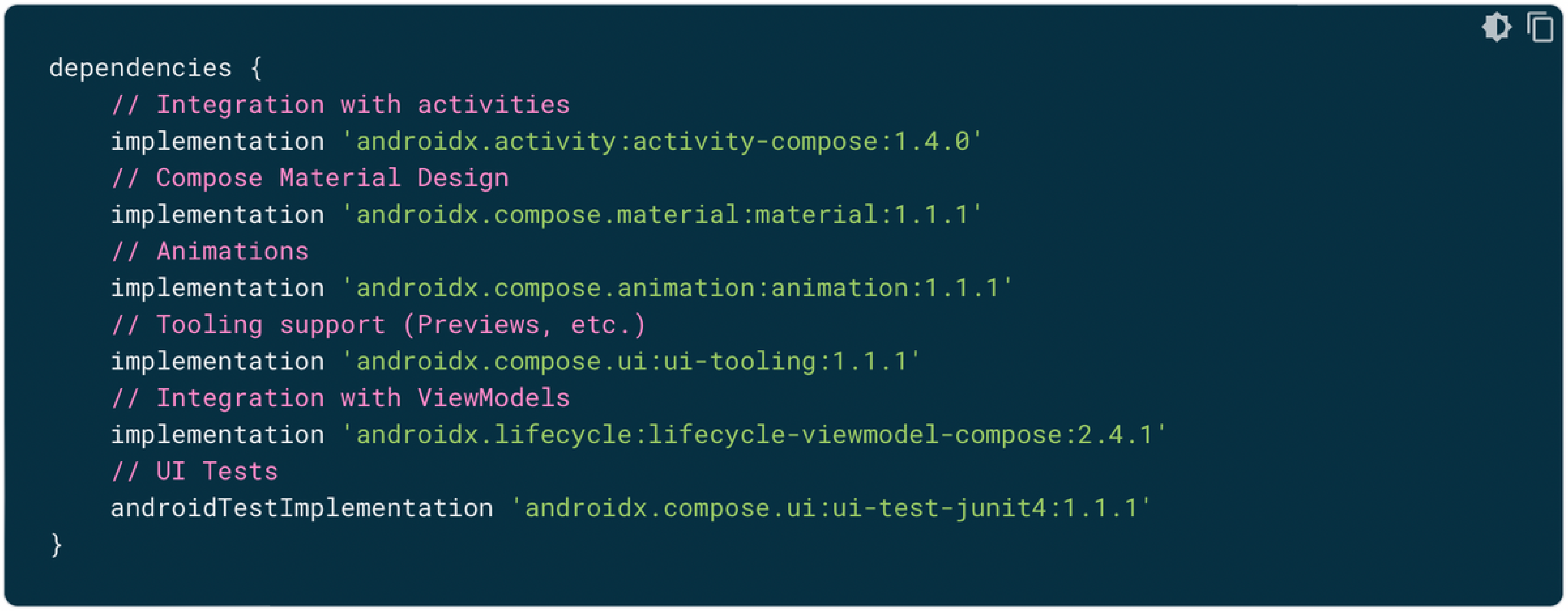
You need to set your app’s minimum API level to 21 or higher and enable Jetpack Compose in your app's build.gradlefile, as shown below. Also set the version for the Kotlin compiler plugin.



```
android {
    defaultConfig {
        ...
        minSdkVersion 21
    }

    buildFeatures {
        // Enables Jetpack Compose for this module
        compose true
    }
    ...

    // Set both the Java and Kotlin compilers to target Java 8.
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = "1.8"
    }

    composeOptions {
        kotlinCompilerExtensionVersion '1.1.1'
    }
}
```

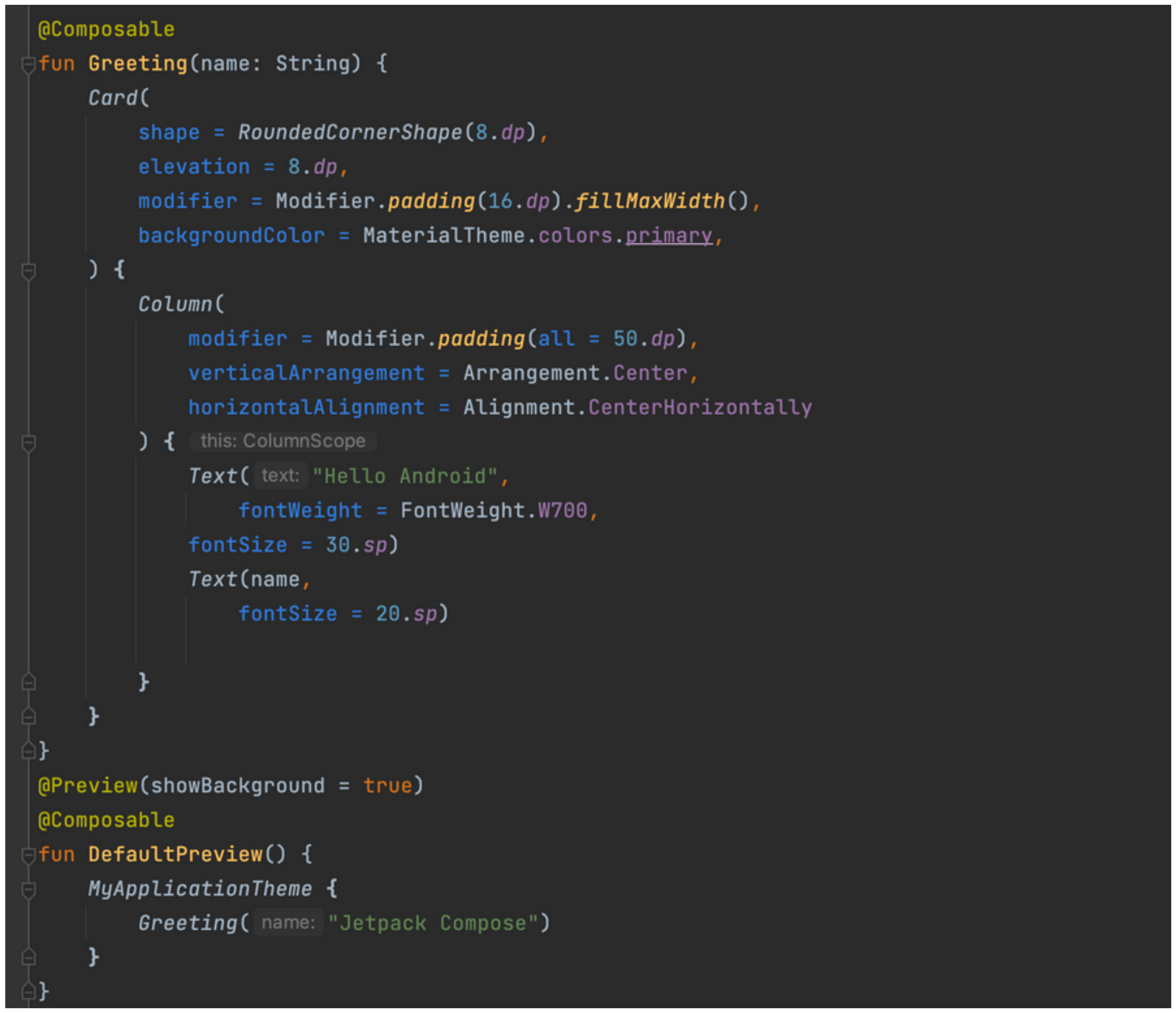
Step 4 : Add Jetpack Compose toolkit dependencies

You need to set your app’s minimum API level to 21 or higher and enable Jetpack Compose in your app's build.gradlefile, as shown below. Also set the version for the Kotlin compiler plugin.

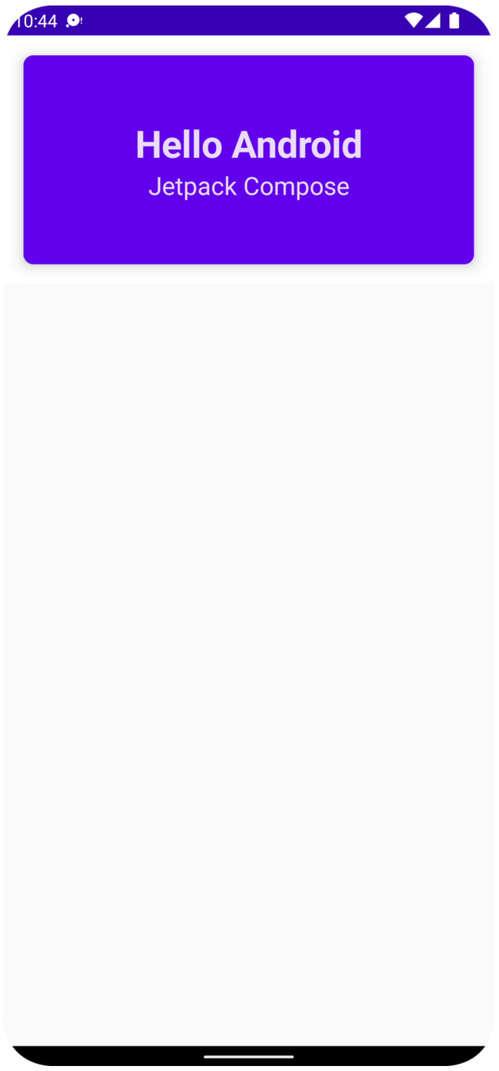


```
dependencies {
    // Integration with activities
    implementation 'androidx.activity:activity-compose:1.4.0'
    // Compose Material Design
    implementation 'androidx.compose.material:material:1.1.1'
    // Animations
    implementation 'androidx.compose.animation:animation:1.1.1'
    // Tooling support (Previews, etc.)
    implementation 'androidx.compose.ui:ui-tooling:1.1.1'
    // Integration with ViewModels
    implementation 'androidx.lifecycle:lifecycle-viewmodel-compose:2.4.1'
    // UI Tests
    androidTestImplementation 'androidx.compose.ui:ui-test-junit4:1.1.1'
}
```

MainActivity.kt



```kotlin
@Composable
fun Greeting(name: String) {
    Card(
        shape = RoundedCornerShape(8.dp),
        elevation = 8.dp,
        modifier = Modifier.padding(16.dp).fillMaxWidth(),
        backgroundColor = MaterialTheme.colors.primary,
    ) {
        Column(
            modifier = Modifier.padding(all = 50.dp),
            verticalArrangement = Arrangement.Center,
            horizontalAlignment = Alignment.CenterHorizontally
        ) { this: ColumnScope
            Text( text: "Hello Android",
                fontWeight = FontWeight.W700,
            fontSize = 30.sp)
            Text(name,
                fontSize = 20.sp)

        }
    }
}
@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    MyApplicationTheme {
        Greeting( name: "Jetpack Compose")
    }
}
```



Thank You