Unit II: Android User Interface Development

CTE308- AS2025

Royal University of Bhutan

Tutor: Pema Galey

#17682761

Outline

- User Interface
- UI using XML
- Jetpack Compose for UI Designing

What is a user interface (UI)?

- The user interface (UI) of an app is what you see on the screen: text, images, buttons, and many other types of elements, and how it's laid out on the screen.
- It's how the app shows things to the user and how the user interacts with the app.

Setting Up Android Studio and Creating a Basic App

- Steps as follows:
  - Installing Android Studio
  - Creating a New Project, consider the configurations:
    - Name: MyFirstAppPackage
    - name: com.example.myfirstappSave
    - location: Choose a directory on your computer
    - Language: Java/Kotlin
    - Minimum API level: API 21: Android 5.0 (Lollipop)
  - Project Structure
  - Running the App: Set up an Android Virtual Device (AVD) to emulate a mobile device

Basic User Interface using XML

- For example: To add user interaction to the app by using buttons and handling click events.
- Open activity_main.xml and **add a Button** element below the TextView:

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click Me"
    android:layout_gravity="center" />
```

Handling Button Click

- Open MainActivity.java and add the following code to handle the button click event:

```
Button button = findViewById(R.id.button);
 button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Log.d("MainActivity", "Button clicked!");
    }
 });
```

Changing Text on Button Click

- Modify the button click event to change the text of the TextView when the button is clicked:

```
Button button = findViewById(R.id.button);
// Assuming you have a TextView with id 'text_view'
final TextView textView = findViewById(R.id.text_view);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        textView.setText("Button Clicked!");
    }
});
```

Jetpack Compose

What is Jetpack Compose?

- **Jetpack Compose** is a modern toolkit for building Android UIs. Compose simplifies and accelerates UI development on Android with less code, powerful tools, and intuitive Kotlin capabilities.
- With Compose, you can build your UI by defining a set of functions, called composable functions, that take in data and describe UI elements.

Composable Functions

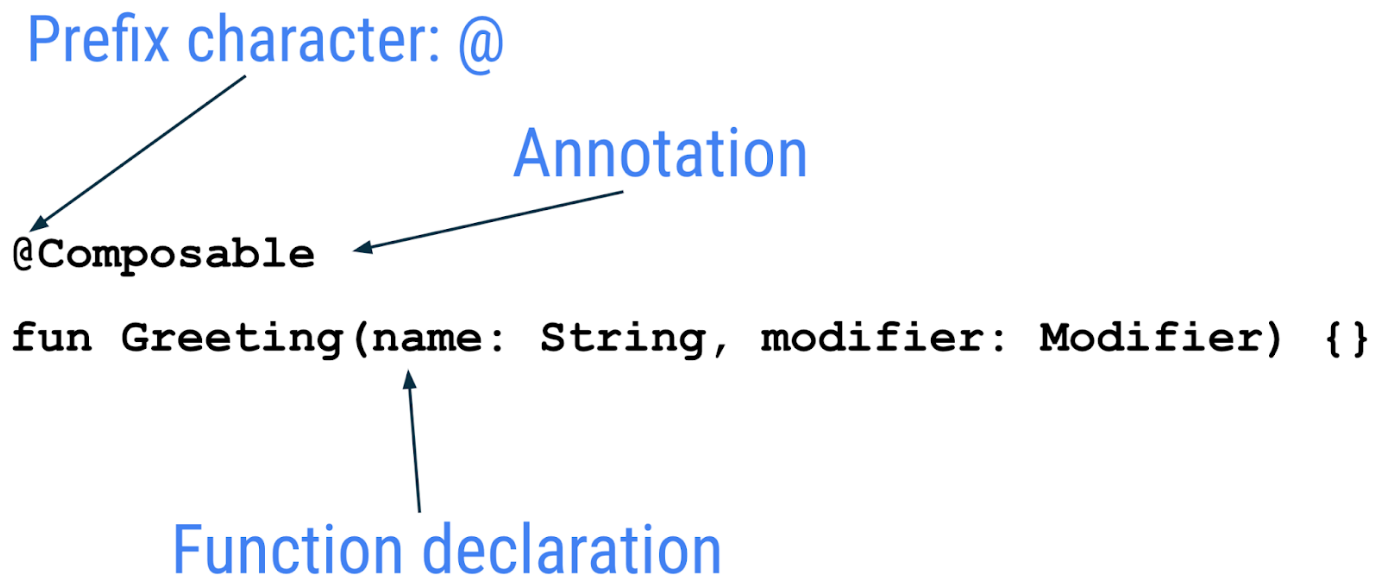
- Composable functions are the basic building block of a UI in Compose.
- A composable function:
  - Describes some part of your UI.
  - Doesn't return anything.
  - Takes some input and generates what's shown on the screen.

Annotations

- Annotations are means of attaching extra information to code.
- This information helps tools like the Jetpack Compose compiler, and other developers understand the app's code.
- An annotation is applied by prefixing its name (the annotation) with the @ character at the beginning of the declaration you are annotating.
- Different code elements, including properties, functions, and classes, can be annotated.

Annotations

- An example of annotated function:

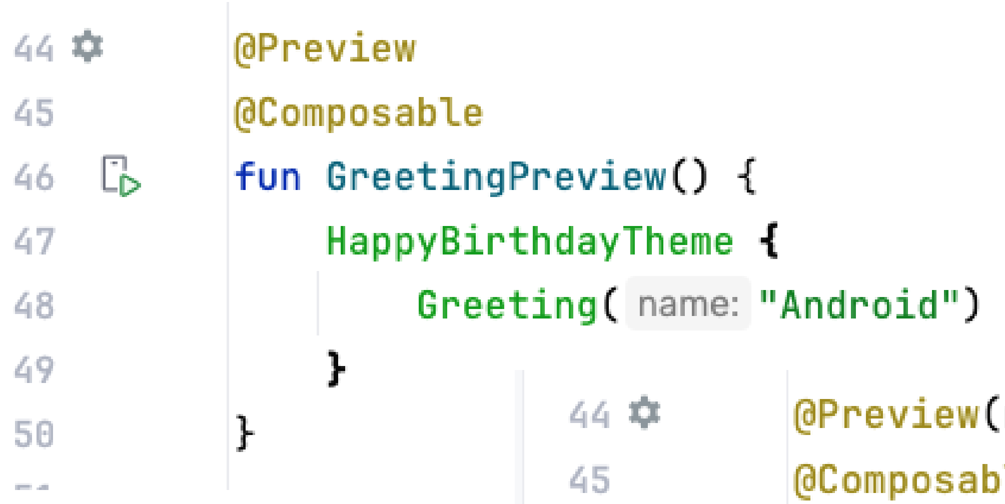


```
@Composable
fun Greeting(name: String, modifier: Modifier) {}
```

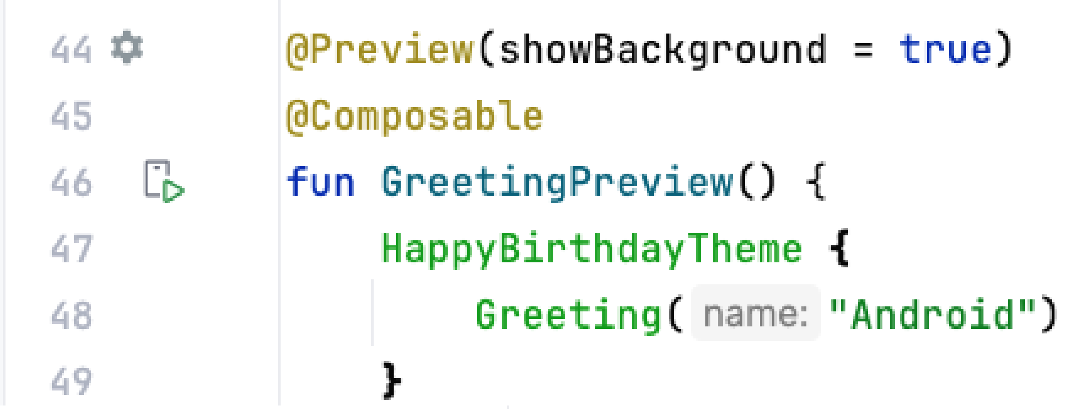
```
@Volatile
private var INSTANCE: AppDatabase? = null
```

Annotations with parameters

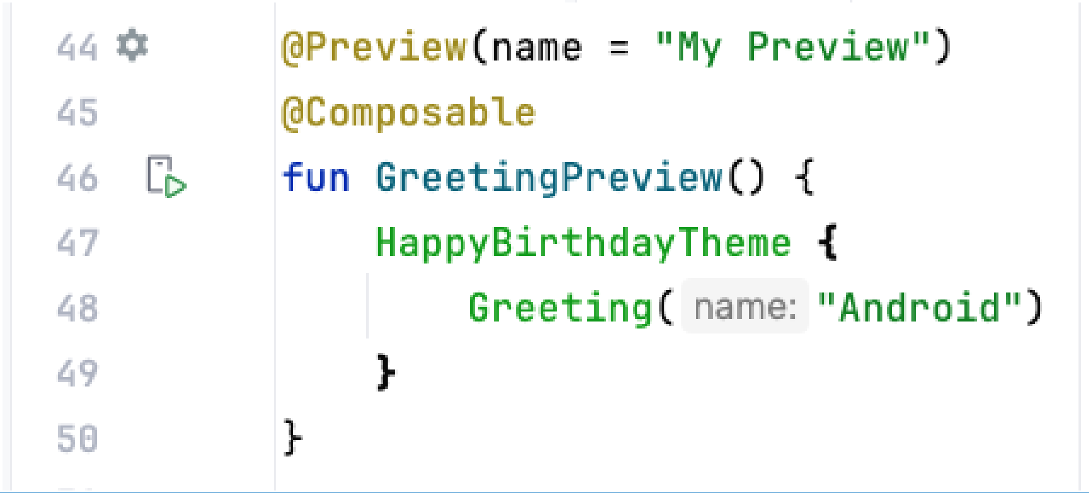
- Annotation without parameters, background and preview title



```
@Preview
@Composable
fun GreetingPreview() {
    HappyBirthdayTheme {
        Greeting( name: "Android")
    }
}
```



```
@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    HappyBirthdayTheme {
        Greeting( name: "Android")
    }
```



```
@Preview(name = "My Preview")
@Composable
fun GreetingPreview() {
    HappyBirthdayTheme {
        Greeting( name: "Android")
    }
}
```

Composable Function

- Jetpack Compose is built around composable functions. These functions let you define your app's UI programmatically by describing how it should look, rather than focusing on the process of the UI's construction.
- To create a composable function, just add the **@Composable** annotation to the function name.
- Composable functions can accept arguments, which let the app logic describe or modify the UI. In this case, your UI element accepts a String so that it can greet the user by name.

Composable Functions in code

1. In Android Studio, open the MainActivity.kt file.
2. Scroll to the GreetingPreview() function. This composable function helps preview the Greeting() function. As a good practice, functions should always be named or renamed to describe their functionality.
3. Change the name of this function to BirthdayCardPreview().

```
@Preview(showBackground = true)
@Composable
fun BirthdayCardPreview() {
   HappyBirthdayTheme {
      Greeting("Android")
   }
}
```

Complete this tutorial…

- Visit Link:

https://developer.android.com/codelabs/basic-android-kotlin-compose-text-composables?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-1-pathway-3%23codelab-https%3A%2F%2Fdeveloper.android.com

Class Work

1. Complete the tutorial provided in the link using Jetpack Compose
2. Replicate the design using XML (UI) and Kotlin/Java (Logic)

Thank you!