

LAB 03

MỤC TIÊU

Kết thúc bài thực hành sinh viên có khả năng:

- ✓ Hiểu cơ bản về cách thiết lập một dự án Jetpack Compose và cách chạy ứng dụng.
- ✓ Làm quen với cú pháp và cách sử dụng composable functions.
- ✓ Học cách sử dụng Text composable để hiển thị văn bản trên màn hình.

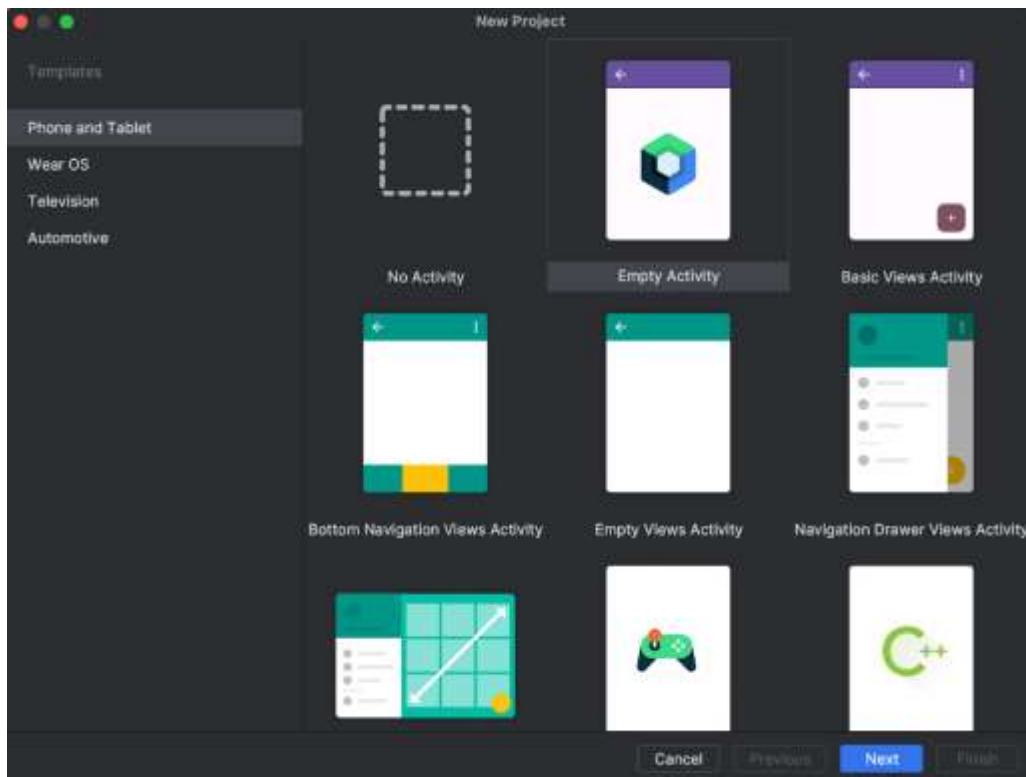
NỘI DUNG

Bài 1: Tạo một ứng dụng Jetpack Compose đơn giản với một Text composable hiển thị "Hello, Họ tên - MSSV!".

Hướng dẫn:

Để bắt đầu một dự án Compose mới, hãy mở Android Studio.

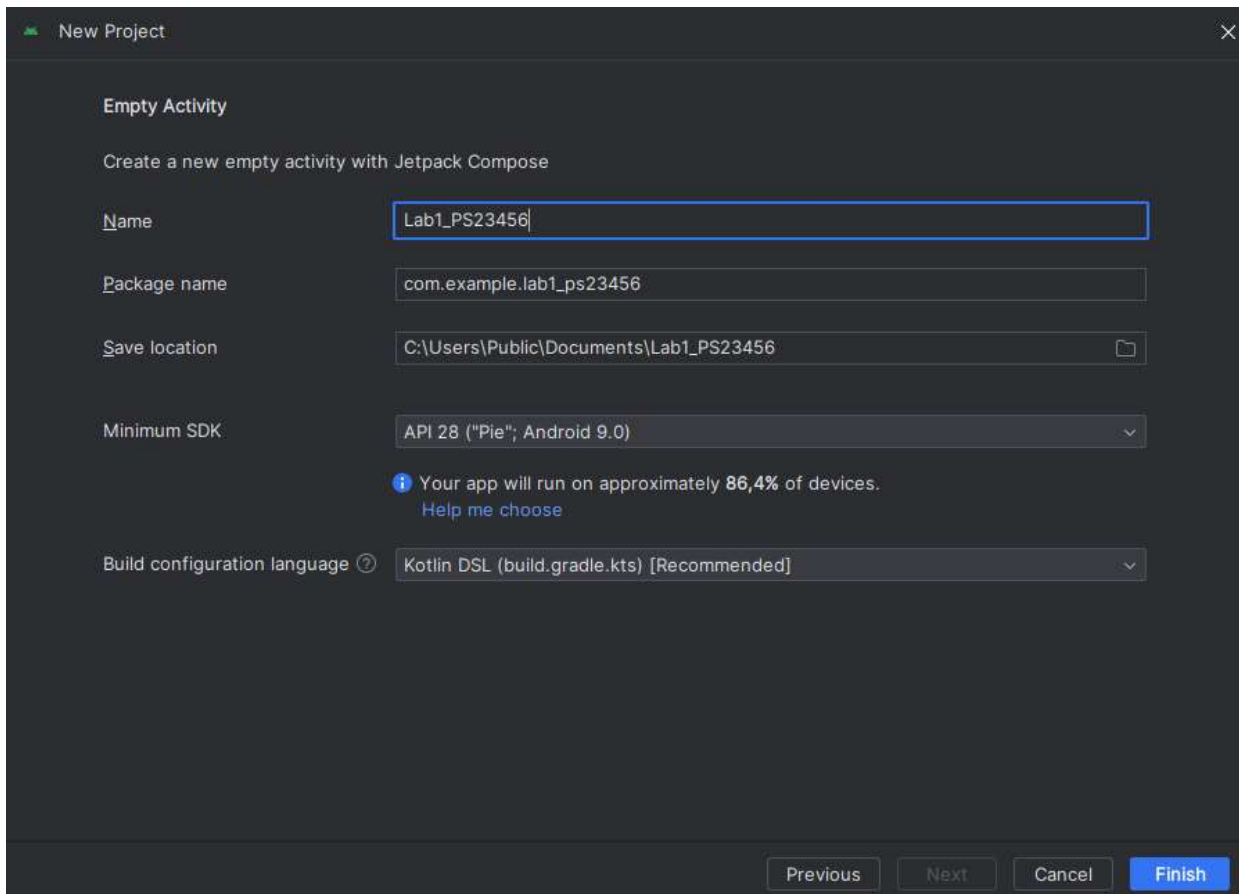
Nếu bạn đang ở cửa sổ Welcome to Android Studio nhấp vào nút New Project.



Đối với dự án mới, hãy chọn **Empty Activity** và nhấn **Next** để tiếp tục.

Ở dialog New Project hãy:

- Đặt tên project theo yêu cầu
- Đặt tên package (nếu cần)
- Chọn nơi lưu trữ project
- Chọn ngôn ngữ cho project là Kotlin (nếu có).
- Chọn minimum SDK



Nhấn **Finish** để tiếp tục.

Sau khi đồng bộ hoá dự án, hãy mở MainActivity.kt và loại bỏ các đoạn code mẫu chỉ để lại class MainActivity.

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView {  
        }  
    }  
}
```

```
}
}
```

Trong Compose, Activity vẫn là điểm bắt đầu của ứng dụng Android. Trong dự án của chúng ta, MainActivity được khởi chạy khi người dùng mở ứng dụng (như được chỉ định trong tệp AndroidManifest.xml). Bạn sẽ sử dụng setContent để xác định bố cục, nhưng thay vì sử dụng tệp XML như thường làm trong hệ thống View truyền thống, hãy gọi các **composable function** trong đó.

❖ Để hiển thị thông tin Họ tên và MSSV lên ứng dụng.

❖ Điều chỉnh file MainActivity.kt như sau:

- Tạo **composable function Greeting** có annotation là @Composable. Thao tác này cho phép hàm của bạn gọi các hàm @Composable khác trong đó.

```
@Composable
fun Greeting(name: String) {
    Text(
        text = "Hello, $name!",
        modifier = Modifier
            .padding(0.dp, 16.dp)
            .fillMaxWidth(),
        color = Color.DarkGray,
        fontSize = 20.sp,
        fontWeight = FontWeight.Bold,
        textAlign = TextAlign.Center,
    )
}
```

- Gọi hàm **Greeting** trong **setContent** trong onCreate của Activity.

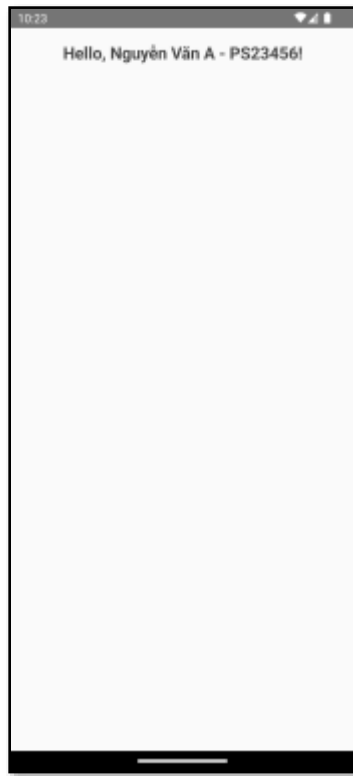
```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Greeting("Nguyễn Văn A - PS23456")
        }
    }
}
```

Lưu ý: Khi nhập các lớp liên quan đến **Jetpack Compose** trong dự án, hãy sử dụng các lớp đó trong các thư viện sau:

androidx.compose.* cho các lớp trình biên dịch và thời gian chạy.

androidx.compose.ui.* cho bộ công cụ và thư viện giao diện người dùng.

- Chạy ứng dụng và xem kết quả.



Trong **Jetpack Compose** ta có thể xem trước giao diện của các **composable function** bằng annotation `@Preview`, annotation này sẽ cập nhật lại giao diện ngay lập tức khi có sự thay đổi.

Để sử dụng `@Preview`, bạn cần thêm annotation này trước hàm **composable function** mà bạn muốn xem trước như sau:

Tạo hàm **PreviewGreeting** và thêm annotation **@Preview**

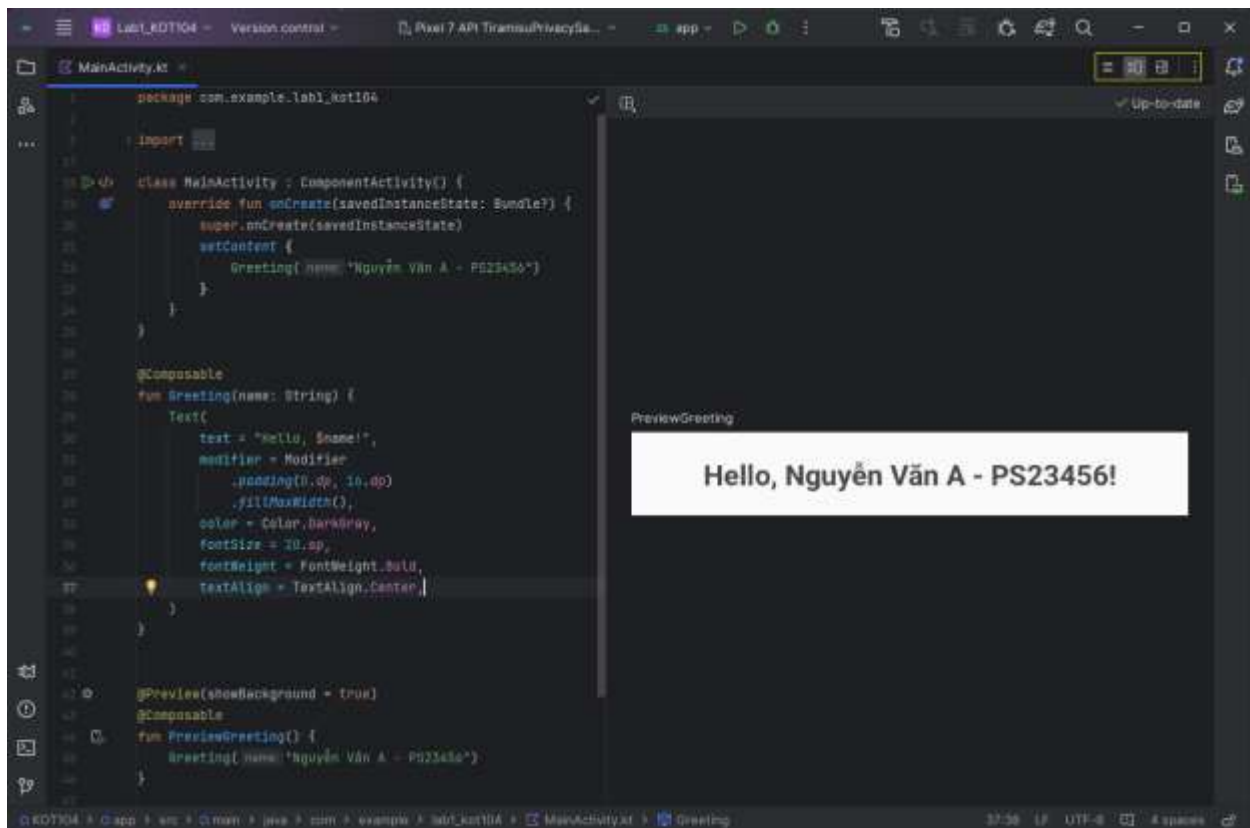
```
@Preview(showBackground = true)
@Composable
fun PreviewGreeting() {
    Greeting("Nguyễn Văn A - PS23456")
}
```

Các thuộc tính của `@Preview`:

- **showBackground**: Kích hoạt nền mặc định cho bản xem trước, giúp nhận diện không gian thành phần.

- **widthDp** và **heightDp**: Đặt kích thước cho bản xem trước với chiều rộng và chiều cao cụ thể trong đơn vị 'dp'.
- **uiMode**: Thiết lập chế độ giao diện (ví dụ: ban đêm hoặc ban ngày) để kiểm tra giao diện trong các trường hợp sử dụng khác nhau.
- **fontScale**: Điều chỉnh tỷ lệ phông chữ, hữu ích cho việc kiểm tra UI với các cài đặt phông chữ khác nhau.
- **locale**: Xác định ngôn ngữ và khu vực, giúp kiểm tra UI trong các ngữ cảnh ngôn ngữ cụ thể.

Trong **Android Studio** chuyển sang chế độ **Split** ở góc trên bên phải của trình soạn thảo mã, bạn sẽ thấy các biểu tượng cho các chế độ xem khác nhau: **Code**, **Split**, và **Design**. Chọn "**Split**" để xem cả mã và bản xem trước cùng một lúc, hoặc chọn "**Design**" để chỉ xem bản xem trước.



Bài 2: Thêm một Button composable vào ứng dụng. Khi nút được nhấn, thay đổi nội dung của Text composable thành "Hi!".



Trước khi click



Sau khi click

Hướng dẫn:

- ❖ Trong MainActivity.kt thực hiện các bước sau:
 - Tạo **composable function GreetingCard**

```
@Composable
fun GreetingCard(msg: String) {
    var text by remember { mutableStateOf(msg) }

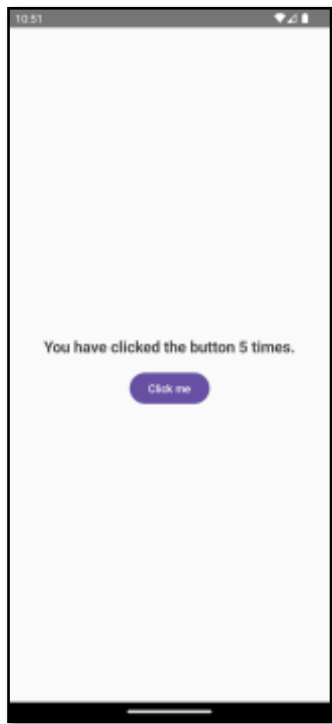
    Column(
        modifier = Modifier
            .fillMaxWidth()
            .padding(24.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        MessageCard(msg = text)
        Button(onClick = { text = "Hi!" }) {
            Text("Say Hi!")
        }
    }
}
```

- Sử dụng `remember` để khai báo một biến trạng thái `text`. Khi giá trị này thay đổi, UI sẽ tự động cập nhật. `mutableStateOf` được sử dụng để tạo một đối tượng trạng thái có thể thay đổi.
- Đổi tên hàm **Greeting** thành **MessageCard** sau đó gọi **MessageCard** trong hàm **GreetingCard**.

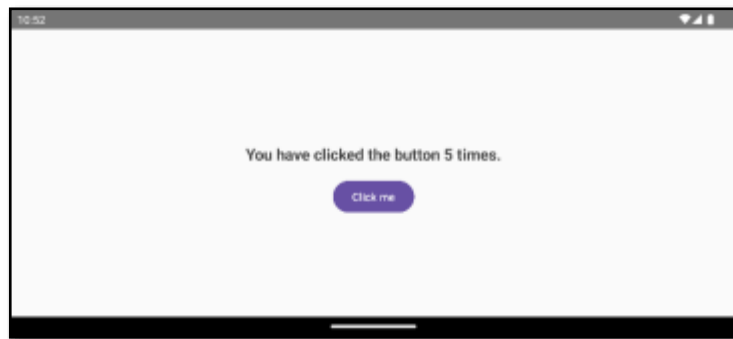
```
@Composable
fun MessageCard(msg: String) {
    Text(
        text = msg,
        modifier = Modifier
            .padding(0.dp, 16.dp)
            .fillMaxWidth(),
        color = Color.DarkGray,
        fontSize = 20.sp,
        fontWeight = FontWeight.Bold,
        textAlign = TextAlign.Center,
    )
}
```

- ❖ Chạy ứng dụng và quan sát kết quả.

Bài 3: Dựa vào bài 1 và bài 2 tạo một counter composable function. Mỗi khi nút được nhấn, giá trị counter tăng lên.



Ở màn hình dọc



Ở màn hình ngang

- ❖ Yêu cầu giá trị vẫn được giữ lại khi xoay màn hình.
- ❖ Trong MainActivity.kt
 - Tạo 1 **composable function CounterCard**.

```
@Composable
fun CounterCard() {
    var count by rememberSaveable { mutableIntStateOf(0) }

    Column(
        modifier = Modifier.fillMaxWidth().padding(24.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        MessageCard("You have clicked the button $count times.")
        Button(onClick = { count++ }) {
            Text("Click me")
        }
    }
}
```

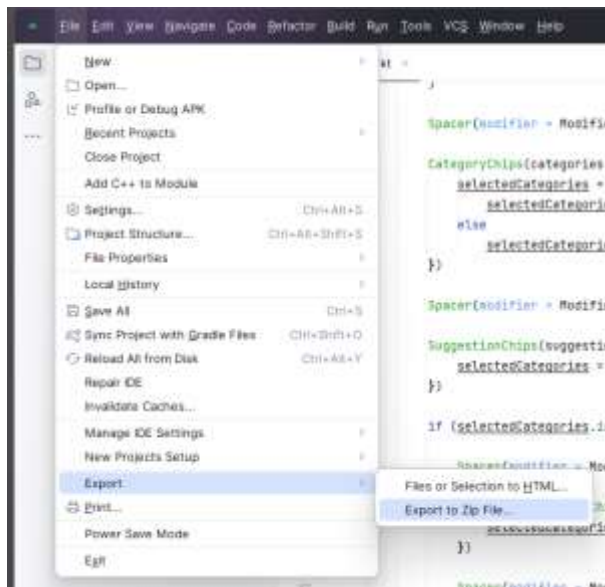

- ❖ `rememberSaveable` được sử dụng để lưu giá trị của biến `count` qua các sự kiện cấu hình. Trong khi `remember` chỉ giữ giá trị trong suốt vòng đời của một `recomposition` và sẽ bị reset khi có thay đổi cấu hình (như xoay màn hình), `rememberSaveable` giữ giá trị qua các thay đổi đó.
- ❖ Chạy ứng dụng và quan sát kết quả khi xoay màn hình, đảm bảo giá trị của `count` không bị reset về 0.

BÀI 4: GV CHO THÊM

*** YÊU CẦU NỘP BÀI:

Sv nén file bao gồm các yêu cầu đã thực hiện trên, nộp lms đúng thời gian quy định của giảng viên. Không nộp bài coi như không có điểm.

Hướng dẫn nén project: **File > Export > Export to Zip File.**



--- Hết ---