Menambahkan Button ke Aplikasi

Buat project baru, beri nama: `Dice Roller`

Dari file MainActivity.kt:

1. Hapus fungsi `Greeting(name: String, modifier: Modifier = Modifier)`.
2. Hapus fungsi `GreetingPreview()`.
3. Buat fungsi `DiceWithButtonAndImage()` dengan anotasi `@Composable`.
4. Buat fungsi `DiceRollerApp()` dengan anotasi `@Preview` dan `@Composable`.

`MainActivity.kt`

```
@Preview
@Composable
fun DiceRollerApp() {

}

@Composable
fun DiceWithButtonAndImage() {

}
```

5. Hapus semua kode di dalam lambda `setContent{}` yang ditemukan dalam metode `onCreate()`.
6. Di bagian lambda `setContent{}`, panggil lambda `DiceRollerTheme{}`, lalu di dalam lambda `DiceRollerTheme{}`, panggil fungsi `DiceRollerApp()`.

`MainActivity.kt`

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContent {
        DiceRollerTheme {
            DiceRollerApp()
        }
    }
}
```

7. Di fungsi `DiceRollerApp()`, panggil fungsi `DiceWithButtonAndImage()`.

`MainActivity.kt`

```
@Preview
@Composable
fun DiceRollerApp() {
```

```
    DiceWithButtonAndImage()
}
```

Menambahkan modifier

Compose menggunakan objek `Modifier` yang merupakan kumpulan elemen yang mendekorasi atau mengubah perilaku elemen UI Compose.

1. Ubah fungsi `DiceWithButtonAndImage()` untuk menerima argumen `modifier` dari tipe `Modifier` dan tetapkan nilai default `Modifier`.

```
@Composable
fun DiceWithButtonAndImage(modifier: Modifier = Modifier) {
}
```

2. Setelah composable `DiceWithButtonAndImage()` memiliki parameter `modifier`, teruskan `modifier` saat composable dipanggil.

```
DiceWithButtonAndImage(modifier = Modifier)
```

3. Buat rantai metode `fillMaxSize()` ke objek `Modifier` sehingga tata letak mengisi seluruh layar.

```
DiceWithButtonAndImage(modifier = Modifier
    .fillMaxSize()
)
```

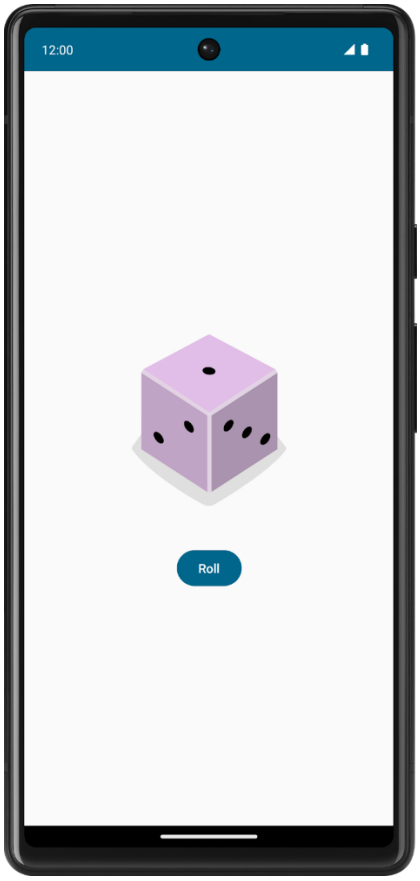
4. Buat rantai metode `wrapContentSize()` ke objek `Modifier`, lalu teruskan `Alignment.Center` sebagai argumen untuk memusatkan komponen. `Alignment.Center` menentukan bahwa komponen dipusatkan secara vertikal dan horizontal.

```
DiceWithButtonAndImage(modifier = Modifier
    .fillMaxSize()
    .wrapContentSize(Alignment.Center)
)
```

Membuat tata letak vertikal

Di Compose, tata letak vertikal dibuat dengan fungsi `Column()`.

Fungsi `Column()` adalah tata letak composable yang menempatkan turunannya dalam urutan vertikal.



1. Dalam fungsi `DiceWithButtonAndImage()`, tambahkan fungsi `Column()`.
2. Teruskan argumen modifier dari deklarasi metode `DiceWithButtonAndImage()` ke argumen modifier `Column()`.
3. Teruskan argumen `horizontalAlignment` ke fungsi `Column()`, lalu tetapkan ke nilai `Alignment.CenterHorizontally`.

```
fun DiceWithButtonAndImage(modifier: Modifier = Modifier) {
    Column (
        modifier = modifier,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {}
}
```

Menambahkan tombol

1. Di file `strings.xml`, tambahkan string dan tetapkan ke nilai `Roll`.

`res/values/strings.xml`

```
<string name="roll">Roll</string>
```

2. Dalam isi lambda `Column()`, tambahkan fungsi `Button()`.
3. Di file `MainActivity.kt`, tambahkan fungsi `Text()` ke `Button()` dalam isi lambda fungsi.

4. Teruskan ID resource string dari string `roll` ke fungsi `stringResource()` dan teruskan hasilnya ke composable `Text`.

`MainActivity.kt`

```
Column(
    modifier = modifier,
    horizontalAlignment = Alignment.CenterHorizontally
) {
    Button(onClick = { /*TODO*/ }) {
        Text(stringResource(R.string.roll))
    }
}
```

Menambahkan gambar

Komponen penting lain dari aplikasi ini adalah gambar dadu yang menampilkan hasil saat pengguna mengetuk tombol **Roll**. Anda menambahkan gambar dengan composable `Image`.

1. Buka **URL ini** untuk mendownload file zip gambar dadu ke komputer, lalu tunggu download selesai.
2. Ekstrak file zip untuk membuat folder `dice_images` baru yang berisi enam file gambar dadu dengan nilai dadu dari 1 sampai 6.

Menambahkan gambar dadu ke aplikasi

1. Di Android Studio, klik **View > Tool Windows > Resource Manager**.
2. Klik **+ > Import Drawables** untuk membuka file browser.
3. Temukan dan pilih enam folder gambar dadu, lalu lanjutkan untuk menguploadnya.
4. Klik **Next**.
5. Klik **Import** untuk mengonfirmasi bahwa Anda ingin mengimpor enam gambar.

Penting! Anda dapat melihat gambar tersebut dalam kode Kotlin dengan ID resource:

```
R.drawable.dice_1
R.drawable.dice_2
R.drawable.dice_3
R.drawable.dice_4
R.drawable.dice_5
R.drawable.dice_6
```

Menambahkan composable Image

1. Pada isi fungsi `Column()`, buat fungsi `Image()` sebelum fungsi `Button()`.

```
Column(
    modifier = modifier,
    horizontalAlignment = Alignment.CenterHorizontally
) {
    Image()
    Button(onClick = { /*TODO*/ }) {
```

```
        Text(stringResource(R.string.roll))
    }
}
```

2. Teruskan argumen `painter` ke fungsi `Image()`, lalu tetapkan nilai `painterResource` yang menerima argumen ID resource drawable. Untuk saat ini, teruskan ID resource berikut: argumen `R.drawable.dice_1`.

```
Image(
    painter = painterResource(R.drawable.dice_1)
)
```

3. Setiap kali membuat Gambar di aplikasi, Anda harus memberikan apa yang disebut dengan "deskripsi konten". Untuk informasi deskripsi konten selengkapnya, lihat Menjelaskan setiap elemen UI. Anda dapat meneruskan deskripsi konten ke gambar sebagai parameter.

```
Image(
    painter = painterResource(R.drawable.dice_1),
    contentDescription = "1"
)
```

4. Tambahkan composable `Spacer` di antara composable `Image` dan `Button`. `Spacer` menggunakan `Modifier` sebagai parameter.

```
Spacer(modifier = Modifier.height(16.dp))
```

Membuat logika pelemparan dadu

Setelah semua composable yang diperlukan tersedia, Anda akan memodifikasi aplikasi agar tombol melempar dadu jika diketuk.

Membuat tombol menjadi interaktif

1. Pada fungsi `DiceWithButtonAndImage()` sebelum fungsi `Column()`, buat variabel `result` dan tetapkan agar sama dengan nilai `1`.

```
fun DiceWithButtonAndImage(modifier: Modifier = Modifier) {
    var result = 1
. . .
```

2. Lihat composable `Button`. Anda akan melihat parameter `onClick` yang ditetapkan ke sepasang tanda kurung kurawal dengan komentar `/*TODO*/` di dalam kurung kurawal. Dalam hal ini kurung kurawal mewakili lambda, area di dalam kurung kurawal yang menjadi badan lambda. Jika diteruskan sebagai argumen, fungsi juga dapat disebut sebagai "`callback`".

```
Button(onClick = { /*TODO*/ })
```

   Lambda adalah literal fungsi yang merupakan fungsi seperti lainnya, tetapi bukannya dideklarasikan secara terpisah dengan kata kunci `fun`, lambda justru ditulis sebagai inline dan diteruskan sebagai ekspresi.

3. Dalam fungsi Button(), hapus komentar `/*TODO*/` dari nilai isi lambda parameter `onClick`.
4. Lemparan dadu bersifat acak. Untuk menunjukkan hal itu dalam kode, Anda perlu menggunakan sintaksis yang benar untuk menghasilkan angka acak. Di Kotlin, Anda dapat menggunakan metode `random()` pada rentang nomor. Dalam isi lambda `onClick`, tetapkan variabel `result` ke rentang antara 1 hingga 6, lalu panggil metode `random()` pada rentang tersebut. Ingat bahwa dalam Kotlin, rentang ditentukan oleh dua titik antara angka pertama dalam rentang dan angka terakhir dalam rentang.

```
fun DiceWithButtonAndImage(modifier: Modifier = Modifier) {
    var result = 1
    Column(
        modifier = modifier,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Image(
            painter = painterResource(R.drawable.dice_1),
            contentDescription = "1"
        )
        Spacer(modifier = Modifier.height(16.dp))
        Button(onClick = { result = (1..6).random() }) {
            Text(stringResource(R.string.roll))
        }
```

```
    }
}
```

Sekarang tombol tersebut dapat diketuk, tetapi ketukan pada tombol belum akan menyebabkan perubahan visual apa pun karena Anda masih harus membangun fungsi tersebut.

Menambahkan kondisional ke aplikasi dice roller

1. Ubah variabel `result` menjadi composable `remember`.
2. Dalam isi composable `remember`, teruskan fungsi `mutableStateOf()`, lalu teruskan argumen `1` ke fungsi tersebut.

Fungsi `mutableStateOf()` akan menampilkan objek yang dapat diamati. Pada dasarnya ini berarti bahwa saat nilai variabel `result` berubah, rekomposisi akan dipicu, nilai hasilnya akan tercermin, dan UI akan di-refresh.

```
var result by remember { mutableStateOf(1) }
```

Sekarang, saat tombol diketuk, variabel `result` akan diupdate dengan nilai angka acak.

Sekarang, variabel `result` dapat digunakan untuk menentukan gambar yang akan ditampilkan.

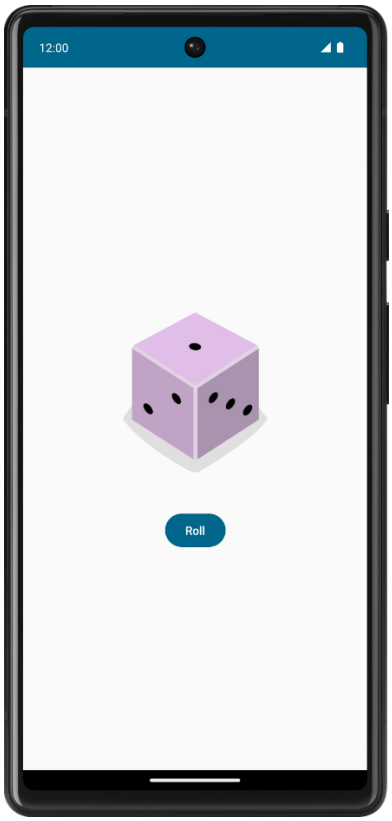
3. Di bawah pembuatan instance variabel `result`, buat variabel `imageResource` tetap yang ditetapkan ke ekspresi `when` yang menerima variabel `result`, lalu tetapkan setiap kemungkinan hasil ke drawable-nya.

```
val imageResource = when (result) {
    1 -> R.drawable.dice_1
    2 -> R.drawable.dice_2
    3 -> R.drawable.dice_3
    4 -> R.drawable.dice_4
    5 -> R.drawable.dice_5
    else -> R.drawable.dice_6
}
```

4. Ubah ID yang diteruskan ke parameter `painterResource` composable Image dari drawable `R.drawable.dice_1` ke variabel `imageResource`.
5. Ubah parameter `contentDescription` composable Image untuk mencerminkan nilai variabel `result` dengan mengonversi variabel `result` menjadi string dengan `toString()` dan meneruskannya sebagai `contentDescription`.

```
Image(
   painter = painterResource(imageResource),
   contentDescription = result.toString()
)
```

6. Jalankan aplikasi Anda. Sekarang seharusnya aplikasi Dice Roller Anda sudah dapat berfungsi sepenuhnya.



Sumber:

https://developer.android.com/codelabs/basic-android-kotlin-compose-build-a-dice-roller-app?hl=id&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-2-pathway-2%3Fhl%3Did%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-build-a-dice-roller-app#0