MOBILE PROGRAMMING

LAYOUTING WITH JETPACK COMPOSE

ROBBY TAN

Container

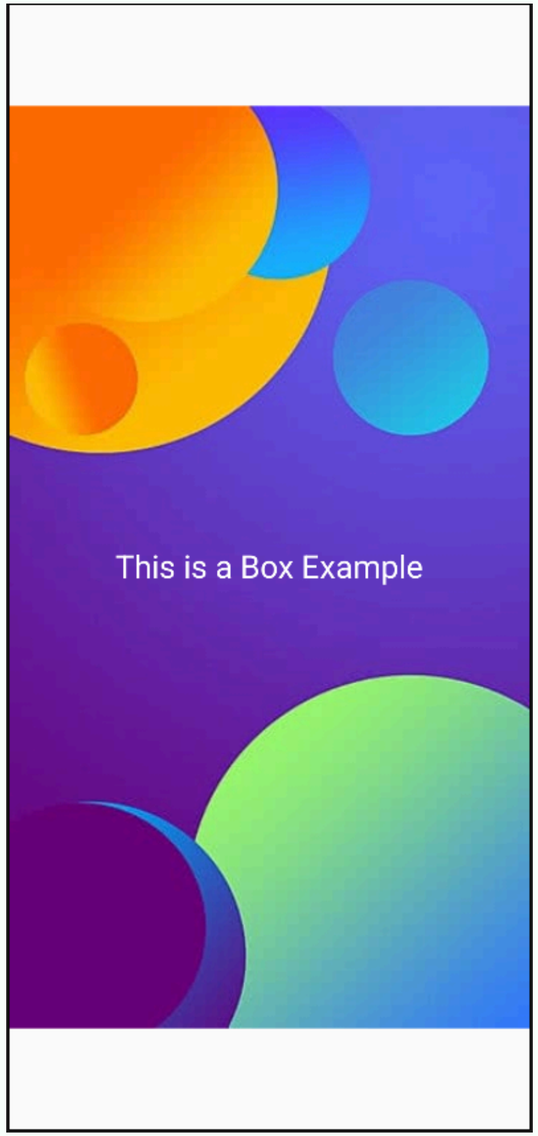
Container in Jetpack Compose

- Sifatnya wajib jika terdapat dua atau lebih komponen yang akan disusun.
- Jenis-jenis container (basic)
  - Box
  - Column
  - Row

Box

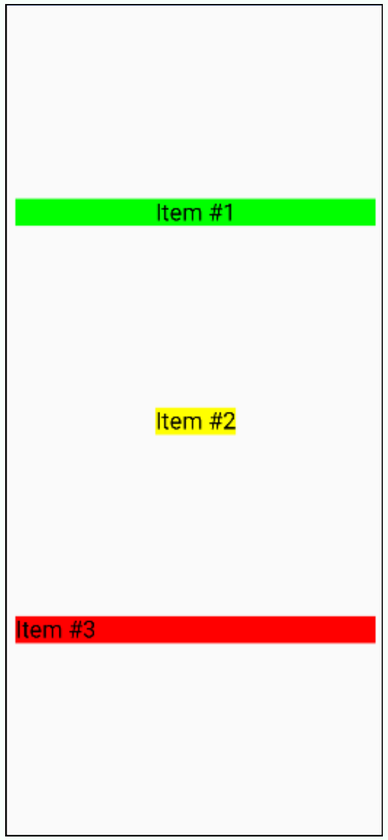
- *Container* di mana sebuah komponen dapat berada di atas/ di bawah komponen lain.
- Padanan *legacy*: **FrameLayout**.

```
1.  @Composable
2.  fun BoxExample() {
3.      Box(
4.          modifier = Modifier.fillMaxSize(),
5.          contentAlignment = Alignment.Center
6.      ) {
7.          Image(
8.              painter = painterResource(id = R.drawable.sample_background),
9.              contentDescription = null,
10.             modifier = Modifier.fillMaxSize()
11.         )
12.         Text(
13.             text = "This is a Box Example",
14.             color = Color.White,
15.             fontSize = 24.sp
16.         )
17.     }
18. }
```



Column

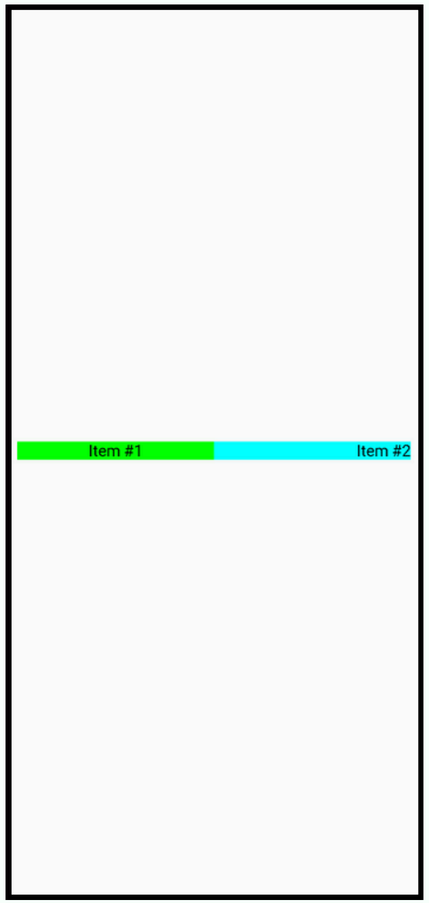
- *Container* di mana komponen diletakkan secara vertikal.
- Padanan *legacy*: **LinearLayout** (orientation = vertical).




```
@Composable
fun ColumnExample() {
    Column(
        modifier = Modifier.fillMaxSize()
            .padding(12.dp),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally,
    ) {
        Surface(
            color = Color.Green,
            modifier = Modifier.fillMaxWidth()
        ) {
            Text(
                text = "Item #1",
                fontSize = 24.sp,
                textAlign = TextAlign.Center
            )
        }
        Surface(color = Color.Yellow) {
            Text(
                text = "Item #2",
                fontSize = 24.sp
            )
        }
        Surface(
            color = Color.Red,
            modifier = Modifier.fillMaxWidth()
        ) {
            Text(
                text = "Item #3",
                fontSize = 24.sp
            )
        }
    }
}
```

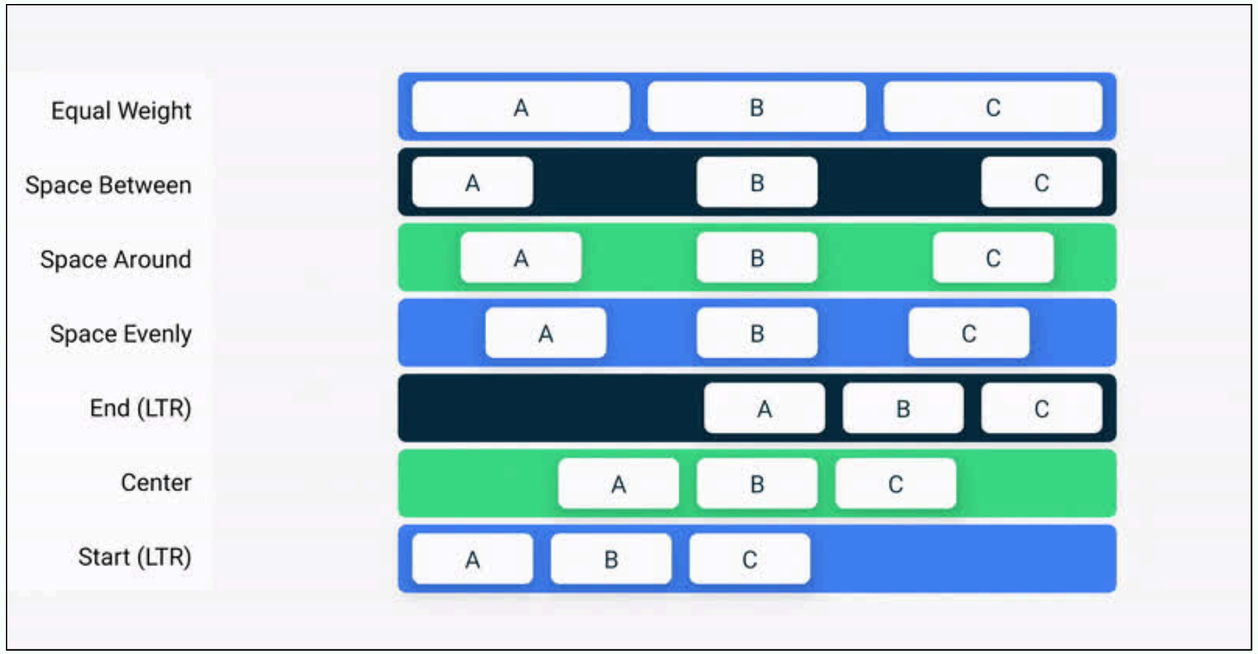
Row

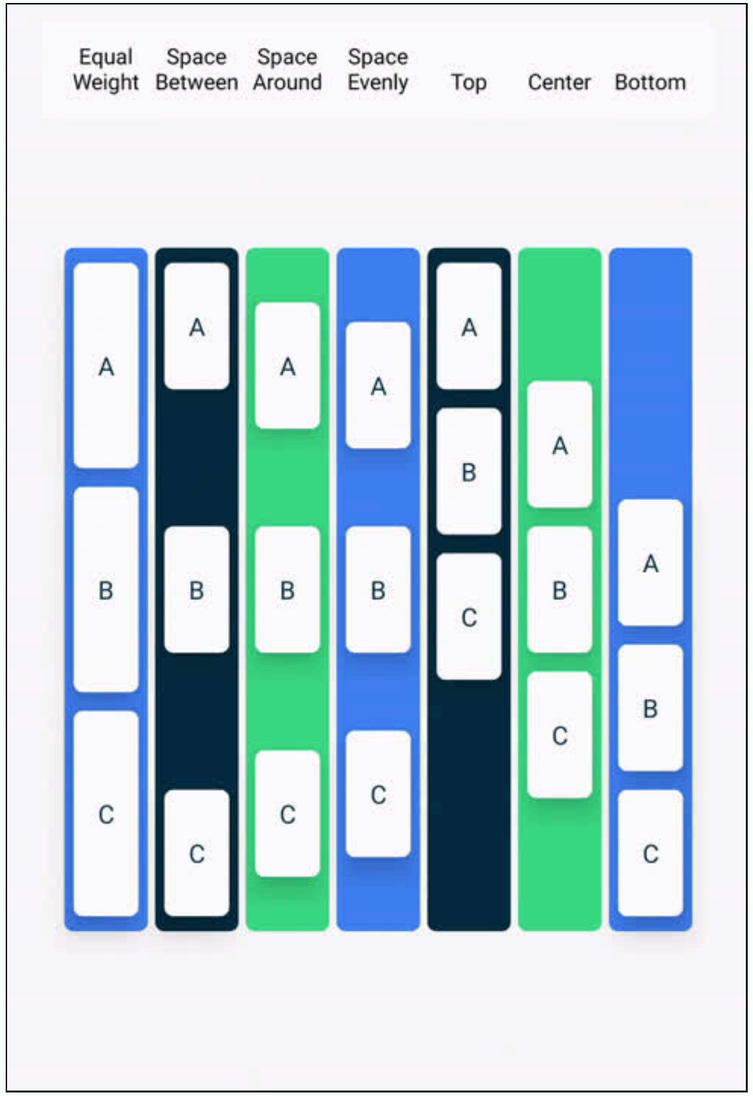
- *Container* di mana komponen diletakkan secara horizontal.
- Padanan *legacy*: **LinearLayout** (orientation = horizontal).




```
1.  @Composable
2.  fun RowExample() {
3.      Row(
4.          modifier = Modifier
5.              .fillMaxSize()
6.              .padding(10.dp),
7.          horizontalArrangement = Arrangement.SpaceEvenly,
8.          verticalAlignment = Alignment.CenterVertically
9.      ) {
10.         Surface(
11.             color = Color.Green,
12.             modifier = Modifier.weight(1f)
13.         ) {
14.             Text(
15.                 text = "Item #1",
16.                 fontSize = 15.sp,
17.                 textAlign = TextAlign.Center
18.             )
19.         }
20.         Surface(
21.             color = Color.Cyan,
22.             modifier = Modifier.weight(1f)
23.         ) {
24.             Text(
25.                 text = "Item #2",
26.                 fontSize = 15.sp,
27.                 textAlign = TextAlign.End
28.             )
29.         }
30.     }
31. }
```

Modifier Arrangement

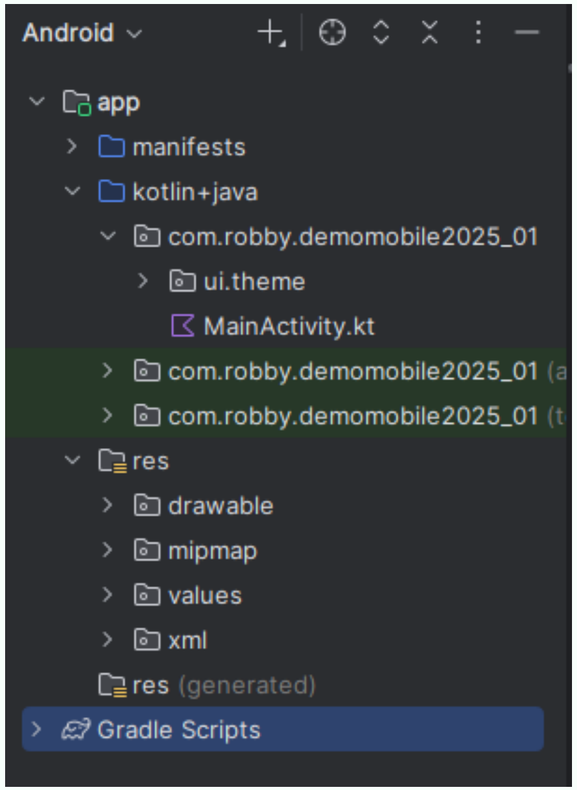




Resource

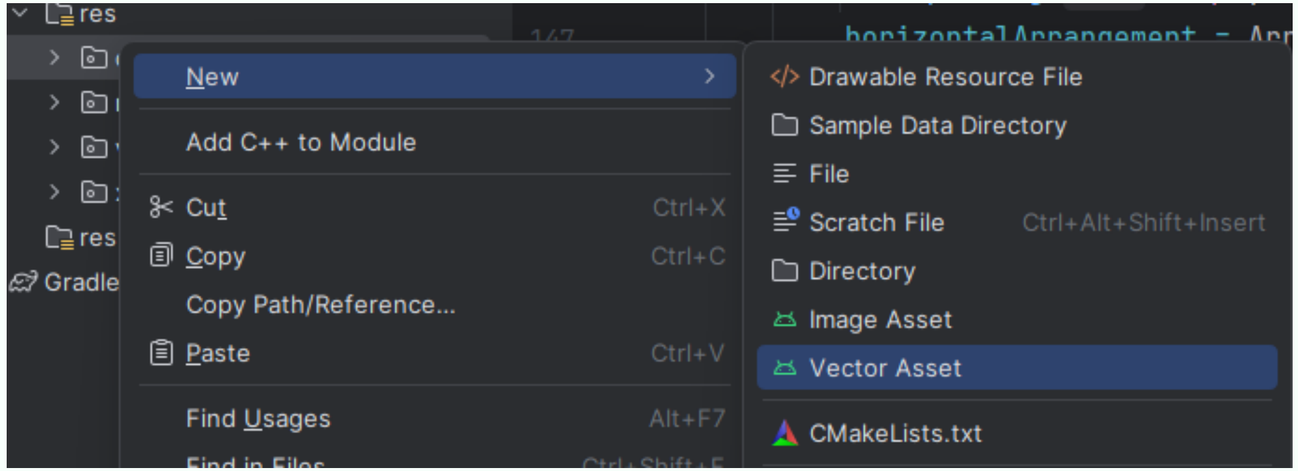
Resource in Android Project

- **src** (Windows Explorer) >> **main**
  - **java** >> kotlin + java
  - **res** >> *File* lain selain Kotlin (Gambar, teks, icon, dll.)
    - **drawable** >> *Image resource* yang digunakan dalam aplikasi.
    - **mipmap** >> Icon untuk aplikasi (*Launcher icon*).
    - **values** >> value resource seperti strings, dimens, colors, dll.



Image/ Vector Resource

- Diletakkan pada folder drawable
- Image
  - *copy-paste file* gambar ke *folder* **drawable**.
  - Spasi pada nama *file* dapat dihilangkan atau diganti menjadi _.
- Vector
  - Klik kanan >> New >> Vector Asset.
  - Pilih *icon*, ukuran, serta warna asset.



String Resource

- Terletak pada *folder* **drawable**.
- Berfungsi menggantikan ***hardcoded*** teks pada aplikasi.

```
01. <resources>
02.     <string name="app_name">Demo Mobile 2025</string>
03.     <string name="title_activity_main">MainActivity</string>
04.     <string name="title">This is title</string>
05.     <string name="description">This is description</string>
06. </resources>
```

Image Composable

- Pastikan file gambar sudah tersedia pada folder *drawable*.

```
1.  @Composable
2.  fun ImageExample(modifier: Modifier = Modifier) {
3.      Image(
4.        painter = painterResource(id = R.drawable.sample_image2),
5.        contentDescription = "Source Code Sample",
6.        modifier = modifier.fillMaxWidth()
7.      )
8.  }
```

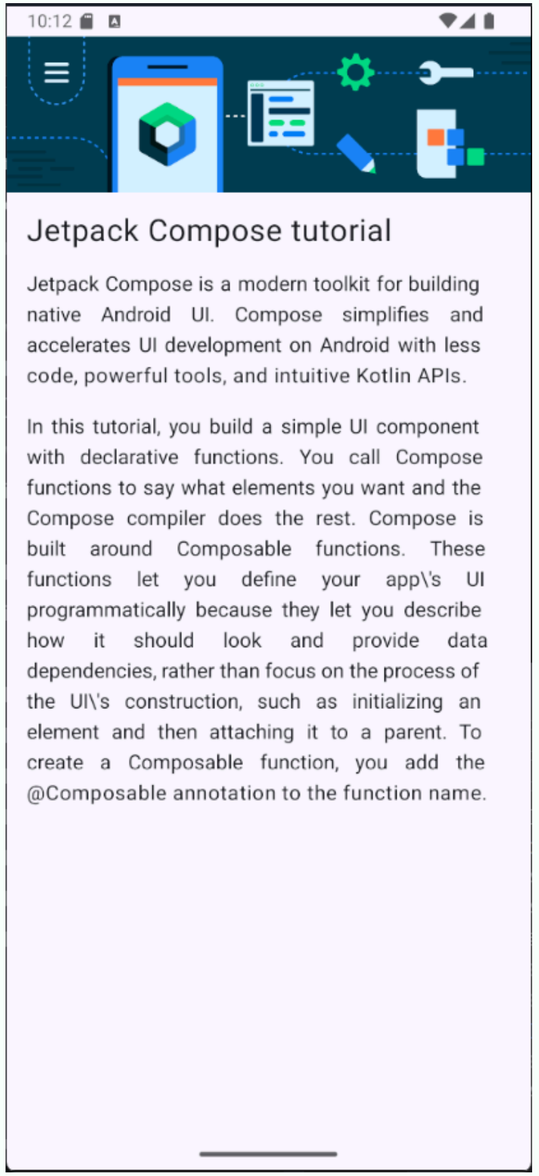
Exercise

Exercise 01

Buatlah sebuah aplikasi Android dengan tampilan hasil akhir seperti gambar di samping. Ketentuan:

1. Gambar memenuhi keseluruhan lebar layar.
2. Text pertama memiiki *font* 24sp dengan *padding* 16dp (*all*).
3. Text kedua memiliki *font default* dengan *padding* 16dp (*start* dan *end*) dan rata kanan-kiri (*justify*).
4. Text ketiga memiliki *font default*, *padding* 16dp (*all*) dan rata kanan-kiri (*justify*).

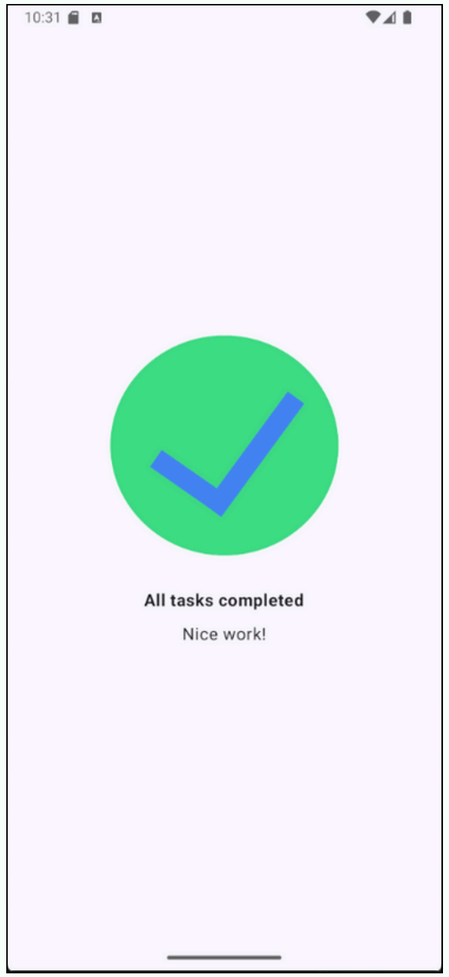
- Jetpack Compose tutorial
- Jetpack Compose is a modern toolkit for building native Android UI. Compose simplifies and accelerates UI development on Android with less code, powerful tools, and intuitive Kotlin APIs.
- In this tutorial, you build a simple UI component with declarative functions. You call Compose functions to say what elements you want and the Compose compiler does the rest. Compose is built around Composable functions. These functions let you define your app\'s UI programmatically because they let you describe how it should look and provide data dependencies, rather than focus on the process of the UI\'s construction, such as initializing an element and then attaching it to a parent. To create a Composable function, you add the @Composable annotation to the function name.



Exercise 02

Buatlah sebuah aplikasi Android dengan tampilan hasil akhir seperti gambar di samping. Ketentuan:

1. Seluruh isi (*content*) dibuat rata tengah secara horisontal dan vertikal.
2. Text pertama memiliki *padding* 24dp (*top*) dan 8dp (*bottom*), serta dicetak tebal.
3. Text kedua memiliki ukuran 16sp.



Exercise 03

Buatlah sebuah aplikasi Android dengan tampilan hasil akhir seperti gambar di samping. Ketentuan:

1. Layar aplikasi dibagi 4 bagian.
2. Masing-masing bagian memiliki *padding* 16dp (*all*).
3. Isi dari masing-masing bagian berada di tengah baik secara horisontal atau vertikal.
4. Text pertama dicetak tebal dan memiliki *padding* 16dp (*bottom*).
5. Text kedua memiliki *font default* dan rata kanan-kiri (*justify*).
6. Gunakan string resource untuk teks yang akan ditampilkan.

Thank You

ROBBY.TAN@IT.MARANATHA.EDU