Nama : M Rezky Raya Kilwouw

NIM : 222313190

Kelas : 3SI1

PRAKTIKUM 10

PEMROGRAMAN PLATFORM KHUSUS

Membangun Aplikasi Dice Roller dengan Jetpack Compose

I. Persiapan Infrastruktur dan Struktur Kode

Langkah pertama adalah membangun struktur dasar aplikasi menggunakan Composable Function dan menyiapkan container UI menggunakan Surface.

Langkah Kerja:

1. Menyiapkan fungsi MainActivity dengan DiceRollerTheme.
2. Menambahkan Surface sebagai wadah utama layout dengan warna latar belakang dari tema.
3. Membuat fungsi DiceRollerApp() dengan anotasi @Preview sebagai entry point.

Kode Implementasi (MainActivity.kt):

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            DiceRollerTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    DiceRollerApp()
                }
            }
```

```
        }
    }
}

@Preview
@Composable
fun DiceRollerApp() {
    DiceWithButtonAndImage(modifier = Modifier
        .fillMaxSize()
        .wrapContentSize(Alignment.Center)
    )
}
```

II. Menambahkan Modifier dan Tata Letak

Aplikasi diatur agar mengisi seluruh layar (fillMaxSize) dan konten diposisikan di tengah layar (Alignment.Center) menggunakan parameter modifier yang diteruskan ke komponen utama.

Kode Tata Letak:

```
@Composable
fun DiceRollerApp() {
    DiceWithButtonAndImage(modifier = Modifier
        .fillMaxSize()
        .wrapContentSize(Alignment.Center)
    )
}
```

III. Menambahkan Aset Gambar dan Tombol

Pada tahap ini, aset gambar dadu diimpor dan ditampilkan menggunakan Image composable. Sebuah tombol (Button) ditambahkan di bawah gambar untuk interaksi pengguna, dengan ukuran teks yang disesuaikan menjadi 24.sp.

Langkah Kerja:

1. Mengunduh dan mengimpor gambar dadu ke folder res/drawable.

2. Menyusun Image dan Button secara vertikal menggunakan Column.
3. Mengatur teks tombol menggunakan stringResource dan memperbesar ukurannya.

Kode Penambahan UI:

```
Column(modifier = modifier, horizontalAlignment = Alignment.CenterHorizontally) {
    Image(
        painter = painterResource(imageResource),
        contentDescription = result.toString()
    )

    Button(
        onClick = { result = (1..6).random() },
    ) {
        Text(text = stringResource(R.string.roll), fontSize = 24.sp)
    }
}
```

IV. Implementasi Logika dan State Management

Langkah terakhir adalah membuat dadu berubah secara acak. Variabel result dibuat menggunakan remember dan mutableStateOf agar perubahan nilai dapat memicu pembaruan UI (Recomposition).

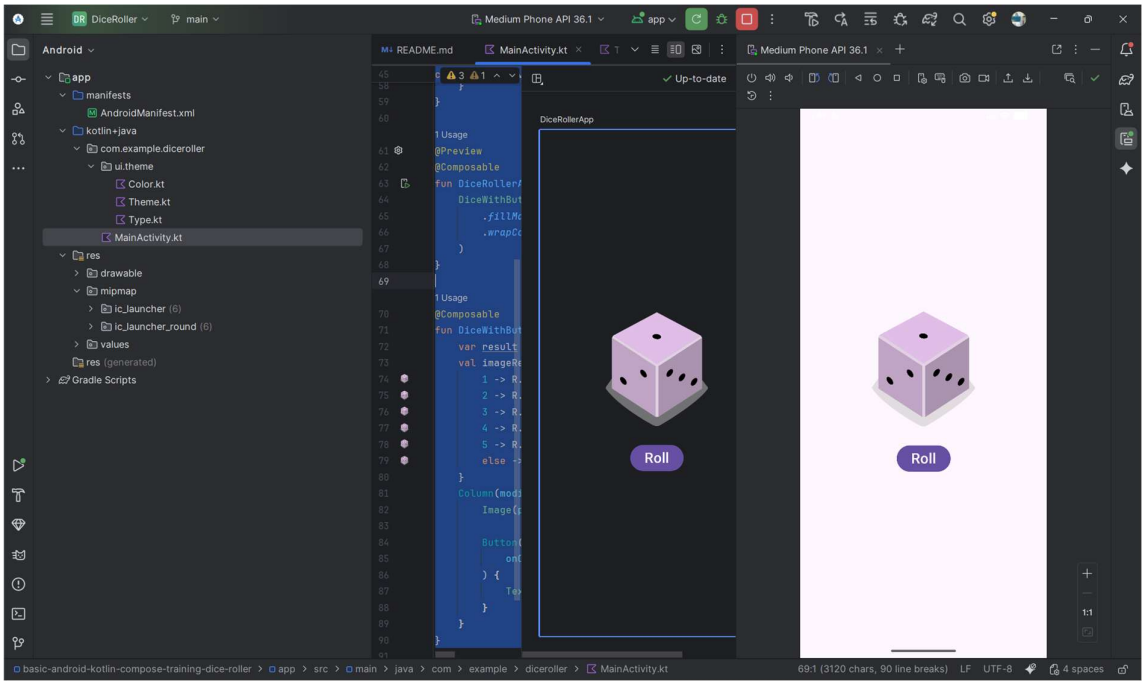
Kode Lengkap Fungsi DiceWithButtonAndImage:

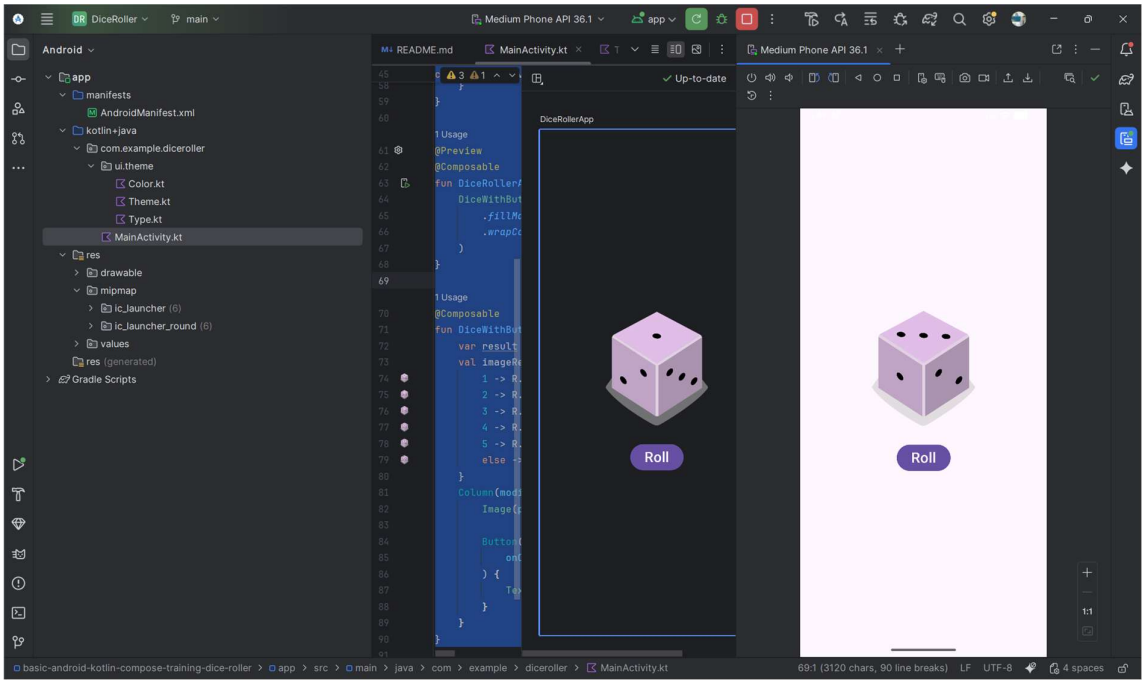
```
@Composable
fun DiceWithButtonAndImage(modifier: Modifier = Modifier) {
    // State untuk menyimpan nilai dadu (1-6)
    var result by remember { mutableStateOf(1) }

    // Logika pemilihan gambar berdasarkan nilai result
    val imageResource = when(result) {
        1 -> R.drawable.dice_1
```

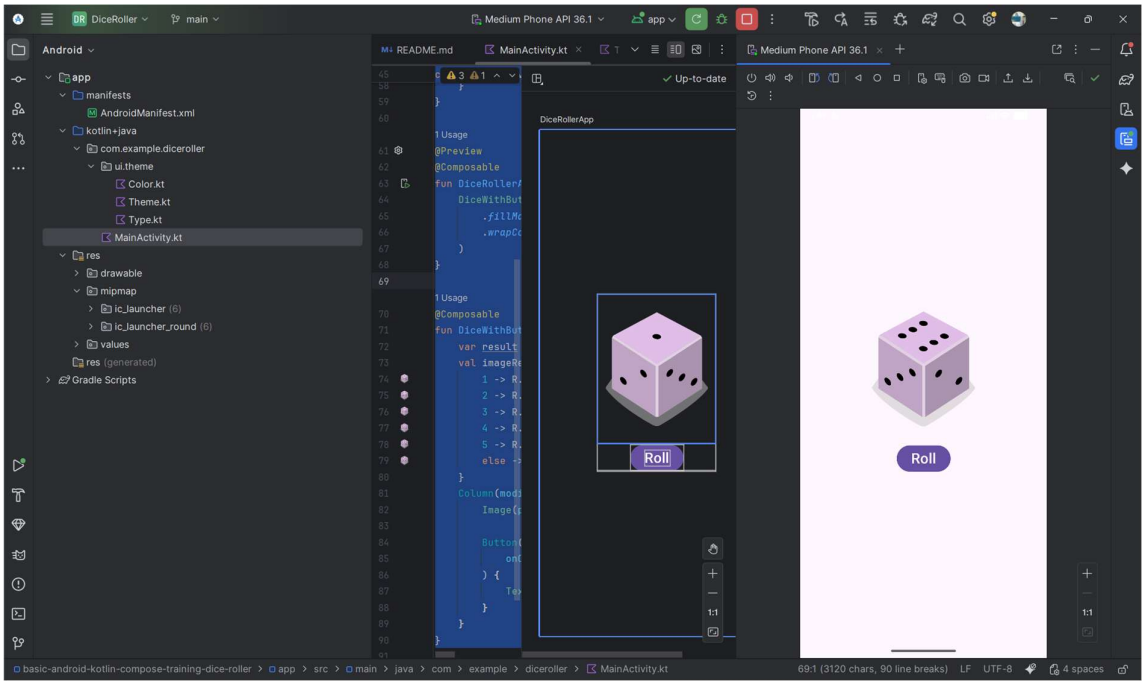
```
        2 -> R.drawable.dice_2
        3 -> R.drawable.dice_3
        4 -> R.drawable.dice_4
        5 -> R.drawable.dice_5
        else -> R.drawable.dice_6
    }

    // Tampilan UI
    Column(modifier = modifier, horizontalAlignment = Alignment.CenterHorizontally) {
        Image(painter = painterResource(imageResource), contentDescription = result.toString())

        Button(
            onClick = { result = (1..6).random() },
        ) {
            Text(text = stringResource(R.string.roll), fontSize = 24.sp)
        }
    }
}
```

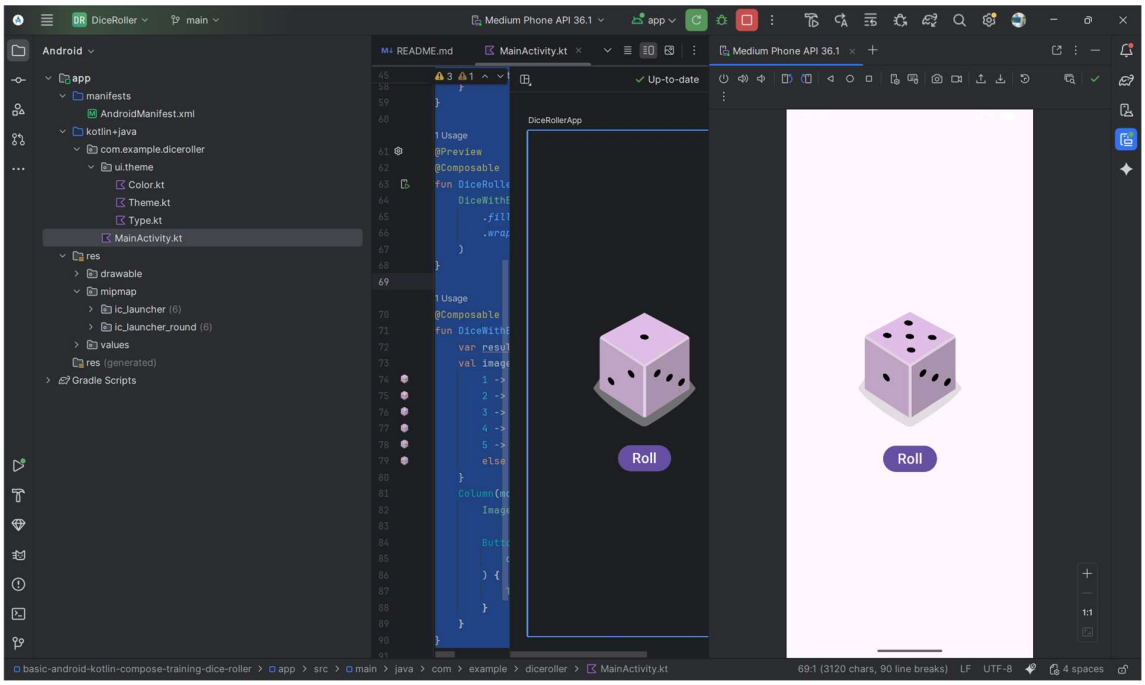
V. Hasil Akhir

Berikut adalah tampilan aplikasi yang sudah berjalan di emulator. Ketika tombol "Roll" ditekan, gambar dadu berubah sesuai angka acak yang dihasilkan.


DiceRoller
main
Medium Phone API 36.1
app
Android
README.md
MainActivity.kt
Up-to-date
DiceRollerApp
Roll
app
manifests
AndroidManifest.xml
kotlin+java
com.example.diceroller
ui.theme
Color.kt
Theme.kt
Type.kt
MainActivity.kt
res
drawable
mipmap
ic_launcher (6)
ic_launcher_round (6)
values
res (generated)
Gradle Scripts
basic-android-kotlin-compose-training-dice-roller
app
src
main
java
com
example
diceroller
MainActivity.kt
69:1 (3120 chars, 90 line breaks)
LF
UTF-8
4 spaces


DiceRoller
main
Medium Phone API 36.1
app
Android
README.md
MainActivity.kt
Up-to-date
DiceRollerApp
Roll
app
manifests
AndroidManifest.xml
kotlin+java
com.example.diceroller
ui.theme
Color.kt
Theme.kt
Type.kt
MainActivity.kt
res
drawable
mipmap
ic_launcher (6)
ic_launcher_round (6)
values
res (generated)
Gradle Scripts
basic-android-kotlin-compose-training-dice-roller
app
src
main
java
com
example
diceroller
MainActivity.kt
69:1 (3120 chars, 90 line breaks)
LF
UTF-8
4 spaces





State Hoisting dan Input Pengguna (Tip Time App)

I. Struktur Utama dan Pengaturan Activity

Pada MainActivity, aplikasi diatur agar mendukung tampilan *Edge-to-Edge* untuk pengalaman visual yang modern. Struktur UI utama dibungkus dalam TipTimeTheme dan Surface.

Kode onCreate:

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        enableEdgeToEdge() // Mengaktifkan tampilan layar penuh
        super.onCreate(savedInstanceState)
        setContent {
            TipTimeTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                ) {
                    TipTimeLayout()
                }
            }
        }
    }
}
```

II. Penerapan State Hoisting pada Layout Utama

Logika utama aplikasi terletak pada TipTimeLayout. Di sini, konsep **State Hoisting** diterapkan: variabel state amountInput disimpan di fungsi induk ini, bukan di dalam komponen input teks anak.

Layout ini juga dilengkapi dengan modifier verticalScroll agar tampilan dapat digulir jika tertutup keyboard, serta statusBarsPadding dan safeDrawingPadding untuk menghindari tabrakan dengan UI sistem.

Kode TipTimeLayout (Parent Composable):

```
@Composable
fun TipTimeLayout() {
    // 1. State disimpan di sini (Hoisting)
```

```
var amountInput by remember { mutableStateOf("") }

// 2. Logika konversi dan perhitungan otomatis saat state berubah
val amount = amountInput.toDoubleOrNull() ?: 0.0
val tip = calculateTip(amount)

Column(
    modifier = Modifier
        .statusBarsPadding()
        .padding(horizontal = 40.dp)
        .verticalScroll(rememberScrollState()) // Agar bisa discroll
        .safeDrawingPadding(),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
) {
    Text(
        text = stringResource(R.string.calculate_tip),
        modifier = Modifier
            .padding(bottom = 16.dp, top = 40.dp)
            .align(alignment = Alignment.Start)
    )

    // 3. Memanggil komponen input dengan meneruskan value dan callback
    EditNumberField(
        value = amountInput,
        onValueChanged = { amountInput = it },
        modifier = Modifier.padding(bottom = 32.dp).fillMaxWidth()
    )

    Text(
```

```
        text = stringResource(R.string.tip_amount, tip),
        style = MaterialTheme.typography.displaySmall
    )
    Spacer(modifier = Modifier.height(150.dp))
  }
}
```

III. Komponen Input yang Stateless

Fungsi EditNumberField dibuat menjadi komponen yang *Stateless* (tidak menyimpan state sendiri). Komponen ini hanya bertugas menerima data (value) dari induk untuk ditampilkan dan mengirimkan event (onValueChanged) ke induk saat pengguna mengetik.

Kode EditNumberField (Child Composable):

```
@Composable
fun EditNumberField(
  value: String,
  onValueChanged: (String) -> Unit, // Callback lambda
  modifier: Modifier
) {
  TextField(
    value = value,
    singleLine = true,
    modifier = modifier,
    onValueChange = onValueChanged, // Memicu callback ke induk
    label = { Text(stringResource(R.string.bill_amount)) },
    keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number)
  )
}
```

IV. Logika Bisnis (Perhitungan Tip)

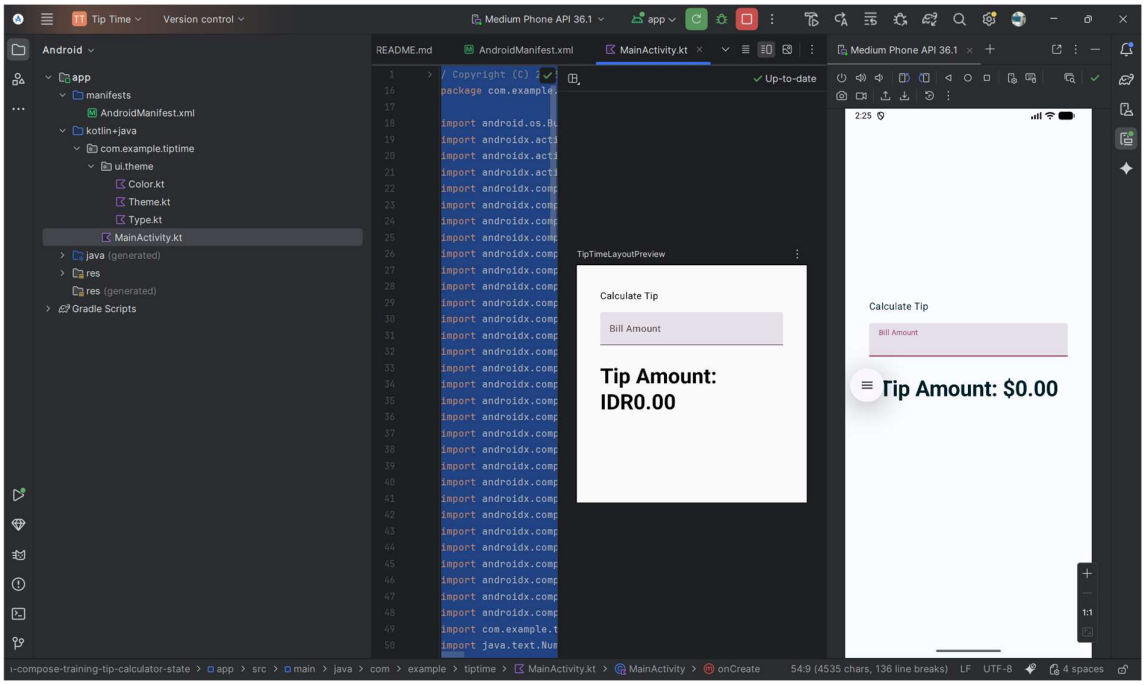
Fungsi calculateTip menangani logika matematika secara terpisah dari UI. Fungsi ini menghitung persentase (default 15%) dan memformat hasilnya menjadi format mata uang lokal (misalnya: $10.00 atau Rp15.000,00 tergantung locale HP).

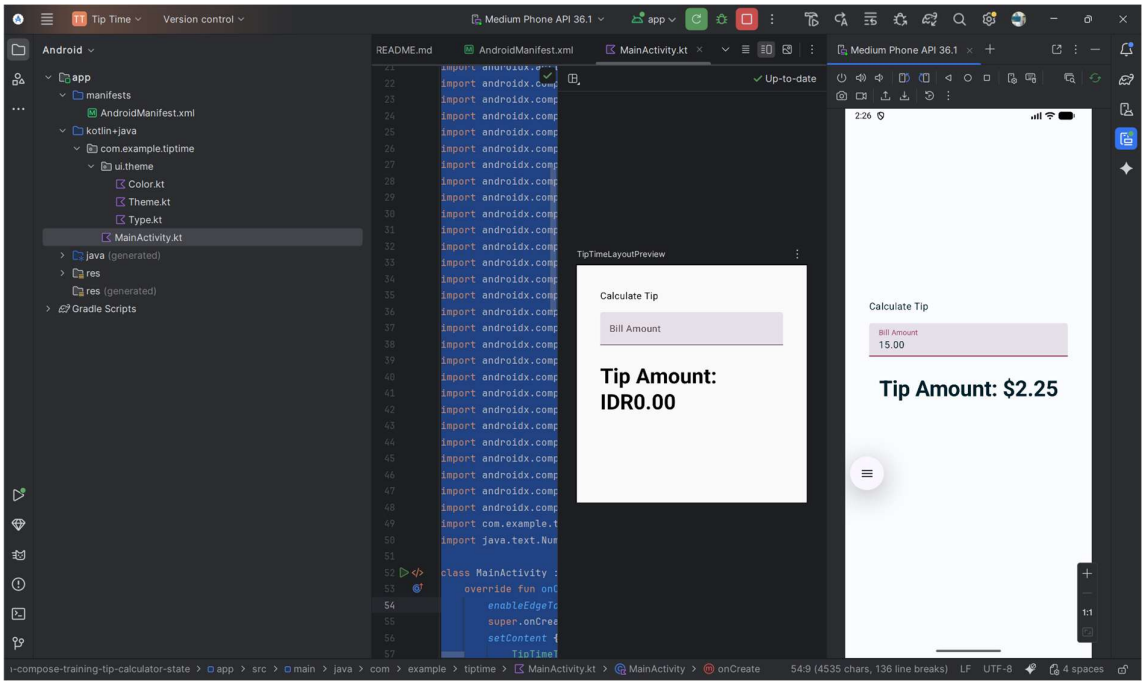
Kode Logika:

```
private fun calculateTip(amount: Double, tipPercent: Double = 15.0): String {
    val tip = tipPercent / 100 * amount
    return NumberFormat.getCurrencyInstance().format(tip)
}
```

V. Hasil Akhir

Berikut adalah tampilan aplikasi saat dijalankan. Pengguna memasukkan nominal tagihan, dan aplikasi secara *real-time* menampilkan jumlah tip yang harus dibayar.




Tip Time
Version control
Medium Phone API 36.1
app
Android
app
manifests
AndroidManifest.xml
kotlin+java
com.example.tiptime
ui.theme
Color.kt
Theme.kt
Type.kt
MainActivity.kt
java (generated)
res
res (generated)
Gradle Scripts
README.md
AndroidManifest.xml
MainActivity.kt
Up-to-date
TipTimeLayoutPreview
Calculate Tip
Bill Amount
Tip Amount: IDR0.00
Medium Phone API 36.1
2:26
Calculate Tip
Bill Amount
15.00
Tip Amount: $2.25
i-compose-training-tip-calculator-state
app
src
main
java
com
example
tiptime
MainActivity.kt
MainActivity
onCreate
54:9 (4535 chars, 136 line breaks)
LF
UTF-8
4 spaces