



MOBILE PROGRAMMING

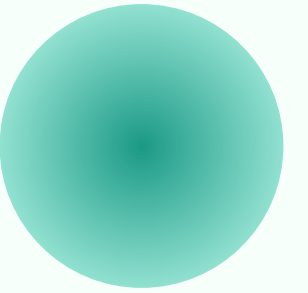
LEGACY ACTIVITY

JETPACK COMPOSE: STATE, TEXTFIELD, SWITCH

ROBBY TAN



Legacy Activity

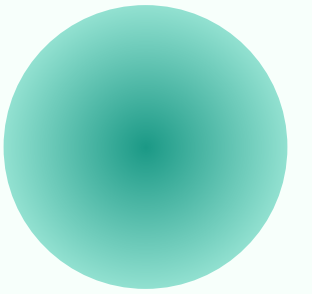


- Activity akan terhubung dengan satu buah view (file berekstensi .xml).
 - LoginActivity.kt >> activity_login.xml
 - MainActivity.kt >> activity_main.xml

```
1. class MainActivity : AppCompatActivity() {
2.
3.     override fun onCreate(savedInstanceState: Bundle?) {
4.         super.onCreate(savedInstanceState)
5.         setContentView(R.layout.activity_main)
6.
7.         btn_web.setOnClickListener {
8.             val uri = Uri.parse("http://it.maranatha.edu")
9.             val intent = Intent(Intent.ACTION_VIEW, uri)
10.            if (intent.resolveActivity(packageManager) != null) {
11.                startActivity(intent)
12.            }
13.        }
14.
15.        btn_call_second.setOnClickListener {
16.            if (et_first_name.text.isEmpty() || et_last_name.text.toString().isEmpty()) {
17.                Toast.makeText(this, R.string.error_empty, Toast.LENGTH_SHORT).show()
18.                Snackbar.make(ll_root, R.string.error_empty, Snackbar.LENGTH_SHORT).show()
19.            } else {
20.                val bundle = Bundle()
21.                bundle.putString(MyViewUtils.KEY_FIRST, et_first_name.text.toString())
22.                bundle.putString(MyViewUtils.KEY_LAST, et_last_name.text.toString())
23.
24.                val intent = Intent(this, SecondActivity::class.java)
25.                intent.putExtras(bundle)
26.                intent.putExtra(Intent.EXTRA_TEXT, et_another_extra.getText().toString())
27.                startActivity(intent)
28.            }
29.        }
30.    }
31. }
32. }
```

```
1. class SecondActivity : AppCompatActivity() {
2.
3.     override fun onCreate(savedInstanceState: Bundle?) {
4.         super.onCreate(savedInstanceState)
5.         setContentView(R.layout.activity_second)
6.
7.         val bundle = intent.extras
8.         if (bundle != null && !bundle.isEmpty) {
9.             val firstName = bundle.getString(MyViewUtils.KEY_FIRST)
10.            val lastName = bundle.getString(MyViewUtils.KEY_LAST)
11.            tv_output.text = String.format("%s %s", firstName, lastName)
12.        }
13.    }
14. }
```

https://github.com/robbytan8/Mobile_02_20192_kt




Legacy vs Jetpack Compose Activity

- Jetpack Compose dapat membungkus seluruh fungsi dalam sebuah MainActivity.
- Navigasi dapat menggunakan NavController dan Composable yang berbeda.
 - Mengganti Composable Greeting dengan Composable lain

```
1. class MainActivity : ComponentActivity() {  
2.     override fun onCreate(savedInstanceState: Bundle?) {  
3.         super.onCreate(savedInstanceState)  
4.         enableEdgeToEdge()  
5.         setContent {  
6.             MobileAppTheme {  
7.                 Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->  
8.                     Greeting(  
9.                         name = "Robby Tan",  
10.                        modifier = Modifier.padding(innerPadding)  
11.                    )  
12.                }  
13.            }  
14.        }  
15.    }  
16. }
```

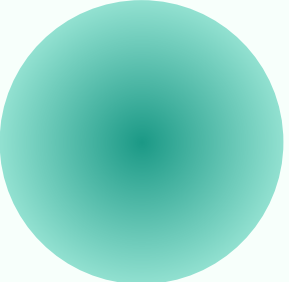
State & Data Class



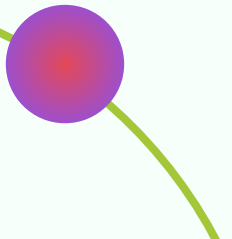


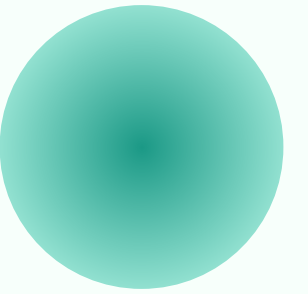
State

- Data yang menyimpan kondisi terkini dari sebuah tampilan.
- variabel count adalah state yang akan diingat dan dimulai dari 0 (Int).
- Setiap eksekusi Button, maka variabel count akan berubah.
- Compose akan melakukan observasi terhadap data dan melakukan perubahan pada UI.



```
1. @Composable
2. fun CounterApp(modifier: Modifier = Modifier) {
3.     var count by remember { mutableIntStateOf(0) }
4.
5.     Column(
6.         modifier = modifier
7.         .padding(16.dp)
8.         .fillMaxSize(),
9.         verticalArrangement = Arrangement.Center,
10.        horizontalAlignment = Alignment.CenterHorizontally
11.    ) {
12.        Text(text = "$count", fontSize = 40.sp)
13.        Button(onClick = { count++ }) {
14.            Text("Add")
15.        }
16.    }
17. }
```



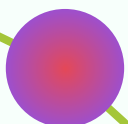


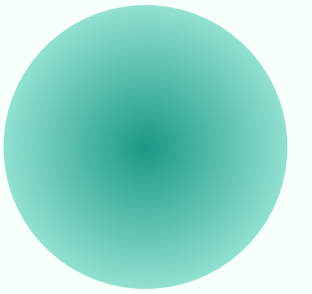
Kotlin data class

- *Class* yang khusus digunakan untuk pemodelan data atau objek.
- Biasa digunakan sebagai pemodelan data dari API.

```
1. class Student(var id: String, var firstNamme: String, var lastName: String)
```

```
1. data class StudentData(var id: String, var firstName: String, var lastName: String)
```





Kotlin data class **vs** Normal class

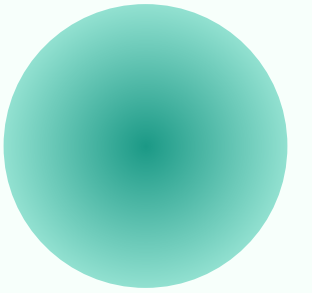
- *data class* digunakan untuk merepresentasikan data tanpa ada *method* kompleks lain, sedangkan *class* biasa digunakan untuk merepresentasikan entitas dengan *method* yang beragam.
 - Person sebagai *data class* hanya berisi nama depan (*firstName*) dan nama belakang (*lastName*)
 - Person sebagai *class* biasa berisi nama depan, nama belakang, serta *method* lain yang dapat dilakukan oleh objek (*instance*) dari *class* Person.
- *Class* biasa hanya menyediakan constructor serta akses atribut dasar (**set** dan **get**). Kotlin *data class* meliputi kedua hal tersebut beserta fungsi data seperti **equals**, **copy**, **toString**, dll.

```
1. fun main() {
2.     val student1 = Student("001", "Robby", "Tan")
3.     val student2 = Student("001", "Robby", "Tan")
4.     println("Check equals: ${student1 == student2}")
5.     println("toString student1: $student1")
6.
7.     val studentData1 = StudentData("002", "John", "Doe")
8.     val studentData2 = StudentData("002", "John", "Doe")
9.     println("Check equals: ${studentData1 == studentData2}")
10.    println("toString student1: $studentData1")
11. }
```

```
Check equals: false
toString student1: com.robby.demo04.Student@6e8cf4c6
Check equals: true
toString student1: StudentData(id=002, firstName=John, lastName=Doe)
```

TextField dan Switch

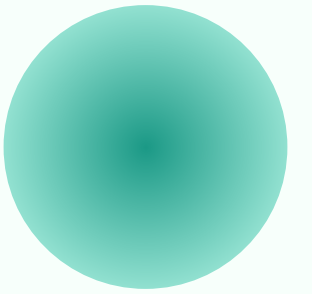




TextField on Jetpack Compose

- Digunakan untuk mengambil input dari pengguna dalam bentuk teks.
- Terdapat pengaturan jenis keyboard (Text, Email, Number, Password, NumberPassword, Uri, etc.)

```
1. @Composable
2. private fun TextFieldExample(modifier: Modifier = Modifier) {
3.     var nameValue by remember { mutableStateOf("") }
4.     val context = LocalContext.current
5.     Column(
6.         modifier = modifier
7.         .fillMaxSize()
8.         .background(Color.LightGray)
9.         .padding(16.dp)
10.    ) {
11.        TextField(
12.            value = nameValue, // required
13.            onChange = { nameValue = it }, // required
14.            singleLine = true,
15.            label = { Text(text = "Name") },
16.            placeholder = { Text(text = "John Doe") },
17.            leadingIcon = {
18.                Image(
19.                    imageVector = Icons.Default.Person,
20.                    contentDescription = ""
21.                )
22.            },
23.            keyboardOptions = KeyboardOptions(
24.                autoCorrectEnabled = false,
25.                keyboardType = KeyboardType.Text,
26.            ),
27.            modifier = Modifier.fillMaxWidth()
28.        )
29.        Spacer(modifier = Modifier.padding(5.dp))
30.        Button(onClick = {
31.            Toast.makeText(context, "$nameValue", Toast.LENGTH_LONG).show()
32.        }, modifier = Modifier.fillMaxWidth()) {
33.            Text(
34.                text = "Submit",
35.                textAlign = TextAlign.Center,
36.                modifier = Modifier.fillMaxWidth()
37.            )
38.        }
39.    }
40. }
```



Switch on Jetpack Compose

- Digunakan untuk mengambil input dari pengguna dalam bentuk teks.
- Terdapat pengaturan jenis keyboard (Text, Email, Number, Password, NumberPassword, Uri, etc.)

```
1. @Composable
2. private fun SwitchExample(modifier: Modifier = Modifier) {
3.     var textFieldValue by remember { mutableStateOf("") }
4.     var isNotificationsEnabled by remember { mutableStateOf(false) }
5.     val context = LocalContext.current
6.
7.     Column(
8.         modifier = modifier
9.         .fillMaxSize()
10.        .padding(16.dp)
11.    ) {
12.        TextField(
13.            value = textFieldValue,
14.            onValueChange = { newText ->
15.                if (newText.all { it.isDigit() }) {
16.                    textFieldValue = newText
17.                }
18.            },
19.            label = { Text("Input phone") },
20.            keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Number),
21.            modifier = Modifier.fillMaxWidth()
22.        )
23.        Row(
24.            modifier = Modifier.fillMaxWidth(),
25.            verticalAlignment = Alignment.CenterVertically
26.        ) {
27.            Text(
28.                text = "Enable Notification",
29.                modifier = Modifier
30.                    .weight(1f)
31.            )
32.            Switch(
33.                checked = isNotificationsEnabled,
34.                onCheckedChange = { newValue ->
35.                    isNotificationsEnabled = newValue
36.                    if (isNotificationsEnabled) {
37.                        Toast.makeText(context, "Notification send to $textFieldValue", Toast.LENGTH_SHORT)
38.                            .show()
39.                    } else {
40.                        Toast.makeText(context, "Notification disabled", Toast.LENGTH_SHORT).show()
41.                    }
42.                }
43.            )
44.        }
45.    }
46. }
47. }
```

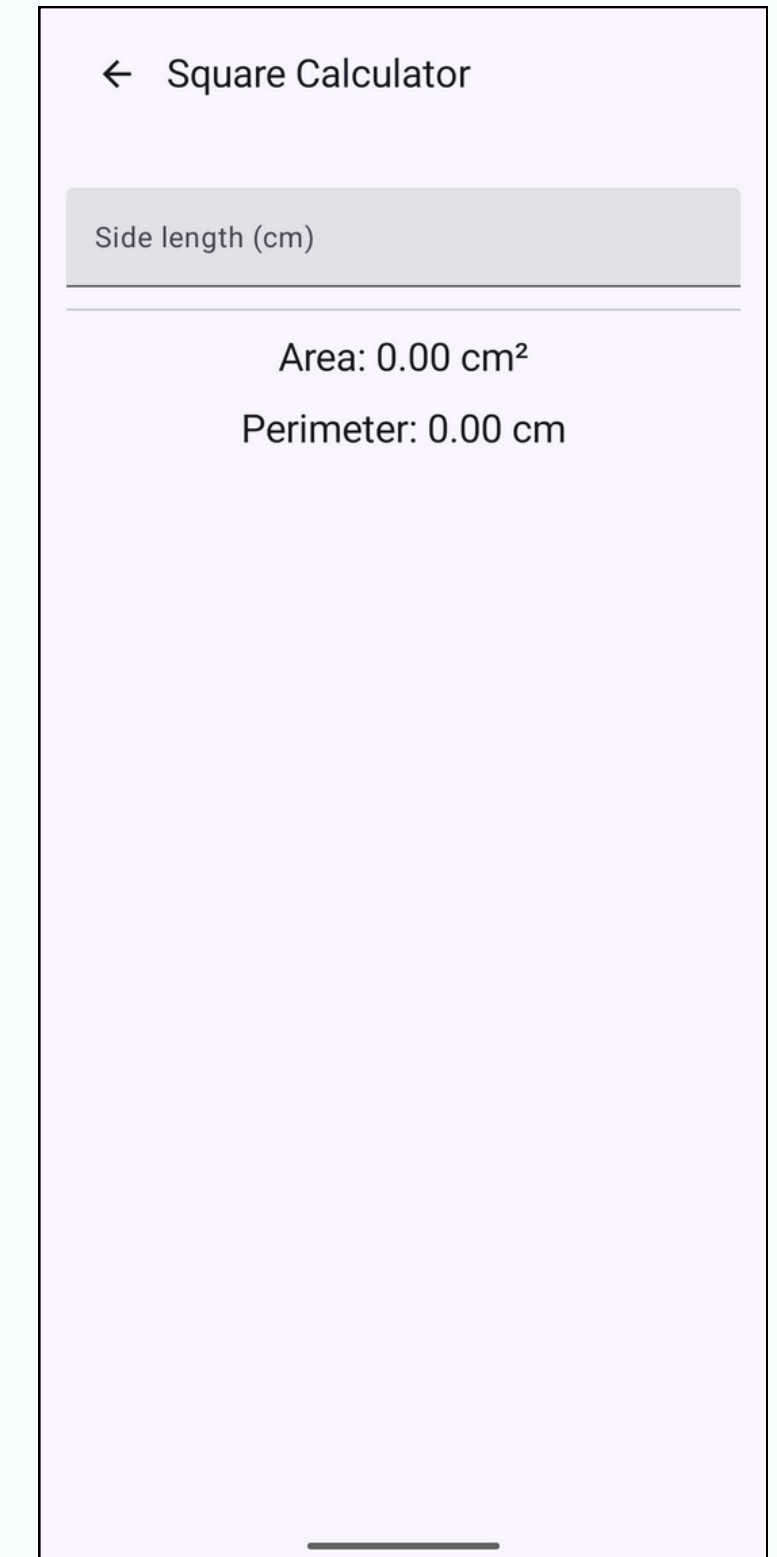
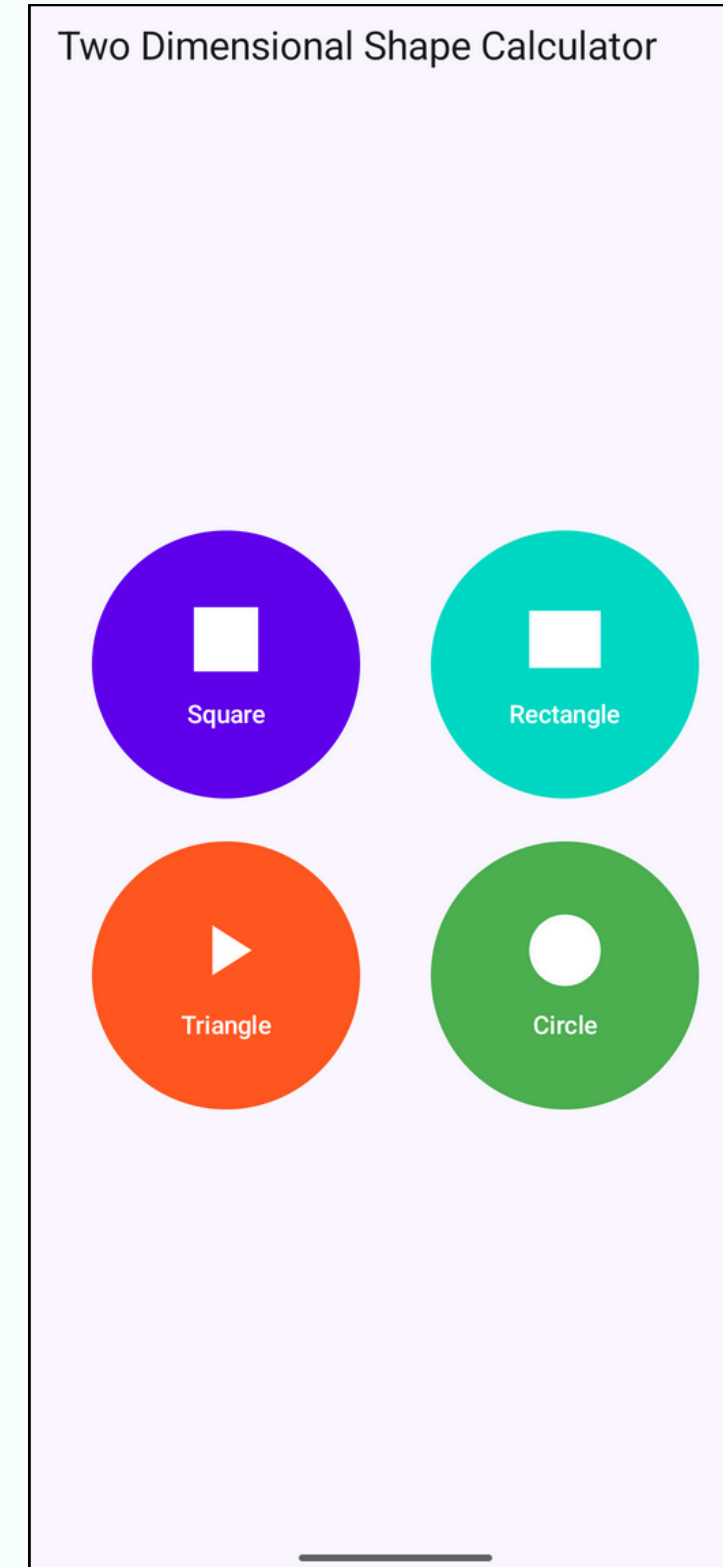
Exercise





2D Shape Calculator

- Buatlah sebuah aplikasi Android untuk perhitungan luas dan keliling bangun datar.
- Halaman Utama (Composable) memiliki 4 buah Menu Button (Composable)
- Menu Button terdiri atas: Button, Column, Icon, Spacer, Text
- Setiap menu button jika diklik akan menuju ke Composable yang lain di mana menampilkan data input dan perhitungan luas serta keliling





Thank You

ROBBY.TAN@IT.MARANATHA.EDU