Fungsi Callback

Praktikum Pemrograman Mobile - 04

Wanda Gusdya

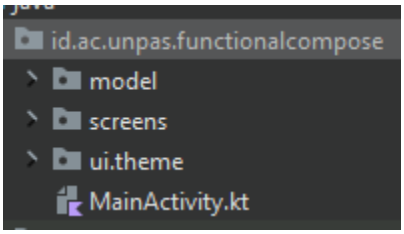
2023

Pendahuluan

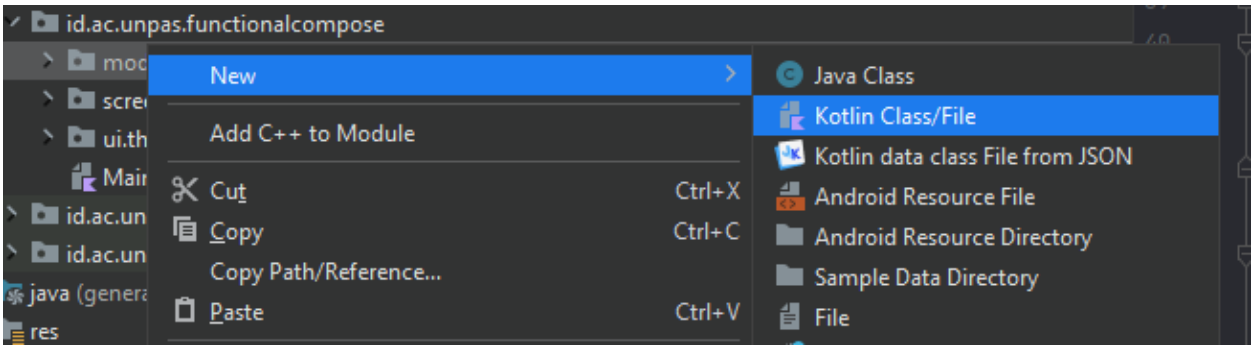
Dalam pemrograman komputer, callback yang juga dikenal sebagai fungsi call-after, adalah referensi apa pun untuk kode yang dapat dieksekusi sebagai argumen untuk kode (fungsi) lain; kode (fungsi) lain tersebut diharapkan untuk memanggil kembali (callback, dengan cara menjalankan) kode pada waktu tertentu.

Latihan

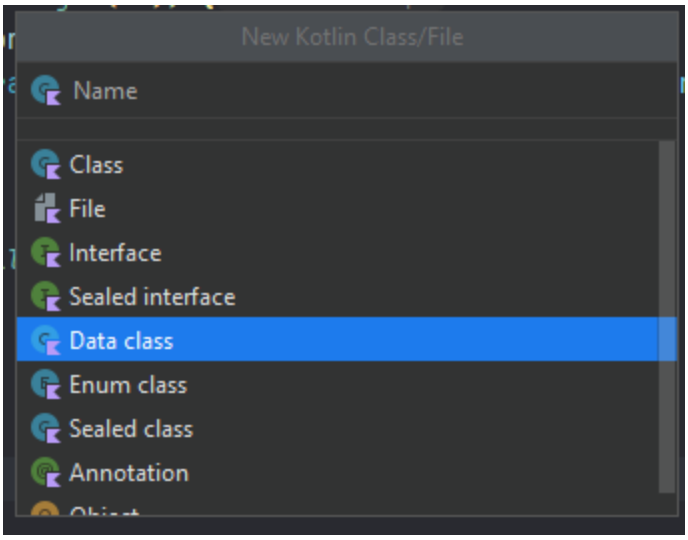
Untuk lebih memahami fungsi callback, mari kita coba membuat sebuah project android baru dengan nama "Functional Compose" dengan cara yang sudah diajarkan di modul kedua. Kita akan membuat sebuah aplikasi dengan form inputan untuk mencatat transaksi penyetoran sampah di sebuah bank sampah. Setelah project siap, buatlah package model dan screens sehingga struktur project akan terlihat seperti berikut.



Selanjutnya, buatlah kelas Data baru di package model dengan nama SetoranSampah dengan cara klik kanan di package model, kemudian pilih New, lalu Kotlin Class/File.



Lalu pilih Data class pada pilihan jenis kelas/file yang akan dibuat.



Beri nama kelas tersebut SetoranSampah kemudian tambahkan kode sehingga kelas SetoranSampah terlihat seperti berikut.

```
package id.ac.unpas.functionalcompose.model

data class SetoranSampah(
    val tanggal: String,
    val nama: String,
    val berat: String
)
```

Kelas data adalah jawaban kotlin untuk kelas POJO dari Java yang memiliki terlalu banyak kode boilerplate (kode yang mubazir).

Selanjutnya, buatlah file kotlin baru di package screens dengan cara yang sama, namun pilih File pada pilihan kelas/file. Beri nama FormPencatatanSampah. Tambahkan kode sebagai berikut.

```
package id.ac.unpas.functionalcompose.screens

import androidx.compose.foundation.layout.*
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.*
import androidx.compose.runtime.Composable
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.input.KeyboardCapitalization
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.TextFieldValue
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import id.ac.unpas.functionalcompose.model.SetoranSampah
import id.ac.unpas.functionalcompose.ui.theme.Purple700
import id.ac.unpas.functionalcompose.ui.theme.Teal200
```

```
@Composable
fun FormPencatatanSampah(onSimpan: (SetoranSampah) -> Unit) {
    val tanggal = remember { mutableStateOf(TextFieldValue("")) }
    val nama = remember { mutableStateOf(TextFieldValue("")) }
    val berat = remember { mutableStateOf(TextFieldValue("")) }

    Column(modifier = Modifier
        .padding(10.dp)
        .fillMaxWidth()) {

        OutlinedTextField(
            label = { Text(text = "Tanggal") },
            value = tanggal.value,
            onValueChange = {
                tanggal.value = it
            },
            modifier = Modifier.padding(4.dp).fillMaxWidth(),
            placeholder = { Text(text = "yyyy-mm-dd") }
        )

        OutlinedTextField(
            label = { Text(text = "Nama") },
            value = nama.value,
            onValueChange = {
                nama.value = it
            },
            modifier = Modifier.padding(4.dp).fillMaxWidth(),
            keyboardOptions = KeyboardOptions(capitalization =
KeyboardCapitalization.Characters, keyboardType = KeyboardType.Text),
            placeholder = { Text(text = "XXXXX") }
        )

        OutlinedTextField(
            label = { Text(text = "Berat") },
            value = berat.value,
            onValueChange = {
                berat.value = it
            },
            modifier = Modifier.padding(4.dp).fillMaxWidth(),
            keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Decimal),
            placeholder = { Text(text = "5") }
        )

        val loginButtonColors = ButtonDefaults.buttonColors(
            backgroundColor = Purple700,
            contentColor = Teal200
        )

        val resetButtonColors = ButtonDefaults.buttonColors(
            backgroundColor = Teal200,
            contentColor = Purple700
        )

        Row (modifier = Modifier.padding(4.dp).fillMaxWidth()) {
            Button(modifier = Modifier.weight(5f), onClick = {
```

```
                val item = SetoranSampah(tanggal.value.text, nama.value.text,
berat.value.text)
                onSimpan(item)
                tanggal.value = TextFieldValue("")
                nama.value = TextFieldValue("")
                berat.value = TextFieldValue("")
            }, colors = loginButtonColors) {
                Text(
                    text = "Simpan",
                    style = TextStyle(
                        color = Color.White,
                        fontSize = 18.sp
                    ), modifier = Modifier.padding(8.dp)
                )
            }

            Button(modifier = Modifier.weight(5f), onClick = {
                tanggal.value = TextFieldValue("")
                nama.value = TextFieldValue("")
                berat.value = TextFieldValue("")
            }, colors = resetButtonColors) {
                Text(
                    text = "Reset",
                    style = TextStyle(
                        color = Color.White,
                        fontSize = 18.sp
                    ), modifier = Modifier.padding(8.dp)
                )
            }
        }
    }
}
```

Beberapa kode yang perlu diperhatikan, diantaranya adalah:

1. Header fungsi FormPencatatanSampah memiliki parameter bertipe Unit yang akan menjadi fungsi callback pada saat form tersebut menyimpan data. Fungsi callback akan menerima 1 parameter bertipe SetoranSampah, yang dapat diterima oleh fungsi composable yang memanggil fungsi FormPencatatanSampah ini.

```
fun FormPencatatanSampah(onSimpan: (SetoranSampah) -> Unit) {
```

2. Di event handler onClick untuk button Simpan, kita panggil fungsi onSimpan yang merupakan parameter callback dari FormPencatatanSampah. Parameter onSimpan dapat diperlakukan/dipanggil seperti fungsi pada umumnya.

```
Button(modifier = Modifier.weight(5f), onClick = {
    val item = SetoranSampah(tanggal.value.text, nama.value.text,
berat.value.text)
    onSimpan(item)
    tanggal.value = TextFieldValue("")
    nama.value = TextFieldValue("")
    berat.value = TextFieldValue("")
}, colors = loginButtonColors) {
    Text(
```

```
            text = "Simpan",
            style = TextStyle(
                color = Color.White,
                fontSize = 18.sp
            ), modifier = Modifier.padding(8.dp)
        )
    }
```

Kemudian, tambahkan sebuah file lain di package screens dengan nama PengelolaanSampahScreen dan kode sebagai berikut.

```
package id.ac.unpas.functionalcompose.screens

import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material.Divider
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.runtime.remember
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import id.ac.unpas.functionalcompose.model.SetoranSampah
import kotlinx.coroutines.flow.MutableStateFlow

@Composable
fun PengelolaanSampahScreen() {
    val _list = remember { MutableStateFlow(listOf<SetoranSampah>()) }
    val list by remember { _list }.collectAsState()

    Column(modifier = Modifier.fillMaxWidth()) {
        FormPencatatanSampah { item ->
            val newList = ArrayList(list)
            newList.add(item)
            _list.value = newList
        }

        LazyColumn(modifier = Modifier.fillMaxWidth()) {
            items(items = list, itemContent = { item ->

                Row(modifier = Modifier
                    .padding(15.dp)
                    .fillMaxWidth()) {
                    Column(modifier = Modifier.weight(3f)) {
                        Text(text = "Tanggal", fontSize = 14.sp)
                        Text(text = item.tanggal, fontSize = 16.sp,
fontWeight = FontWeight.Bold)
                    }
```

```
                    Column(modifier = Modifier.weight(3f)) {
                        Text(text = "Nama", fontSize = 14.sp)
                        Text(text = item.nama, fontSize = 16.sp, fontWeight =
FontWeight.Bold)
                    }

                    Column(modifier = Modifier.weight(3f)) {
                        Text(text = "Berat", fontSize = 14.sp)
                        Text(text = "${item.berat} Kg", fontSize = 16.sp,
fontWeight = FontWeight.Bold)
                    }
                }

                Divider(modifier = Modifier.fillMaxWidth())

            })
        }
    }

}
```

Beberapa kode yang perlu diperhatikan diantaranya:

1. Kita mempunyai 2 variabel yang akan kita gunakan untuk mengelola data yang akan kita tampulkan, yaitu _list dan list. Variabel _list adalah variabel internal yang disimpan dalam flow, dimana data akan berubah, sedangkan variabel list adalah variabel observer yang memantau perubahan data _list. Komponen GUI akan mengikatkan dirinya dengan variabel list. Hal ini dilakukan untuk menghindari memory leak.

```
val _list = remember { MutableStateFlow(listOf<SetoranSampah>()) }
val list by remember { _list }.collectAsState()
```

2. Fungsi composable FormPencatatanSampah dipanggil di dalam Column, dengan parameter callback onSimpan diwakili oleh lambda. Kita harus membuat implementasi untuk fungsi onSimpan, dalam latihan ini kita masukkan item baru dari FormPencatatanSampah ke dalam _list.

```
FormPencatatanSampah { item ->
    val newList = ArrayList(list)
    newList.add(item)
    _list.value = newList
}
```

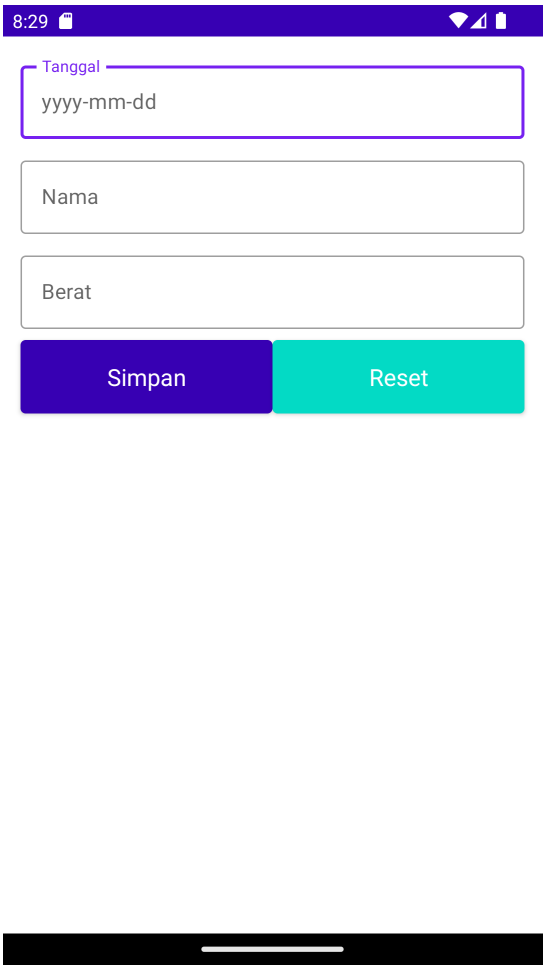
3. Kita gunakan LazyColumn untuk menampilkan list, dengan mengaitkannya di dalam fungsi items di parameter content LazyColumn tersebut. Dengan demikian, LazyColumn akan berubah ketika list berubah.

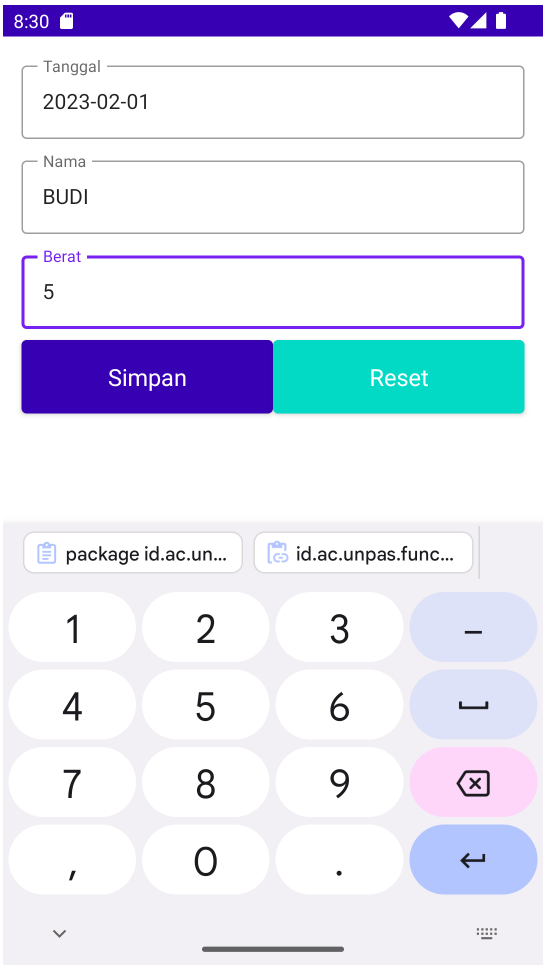
```
LazyColumn(modifier = Modifier.fillMaxWidth()) {
    items(items = list, itemContent = {
```

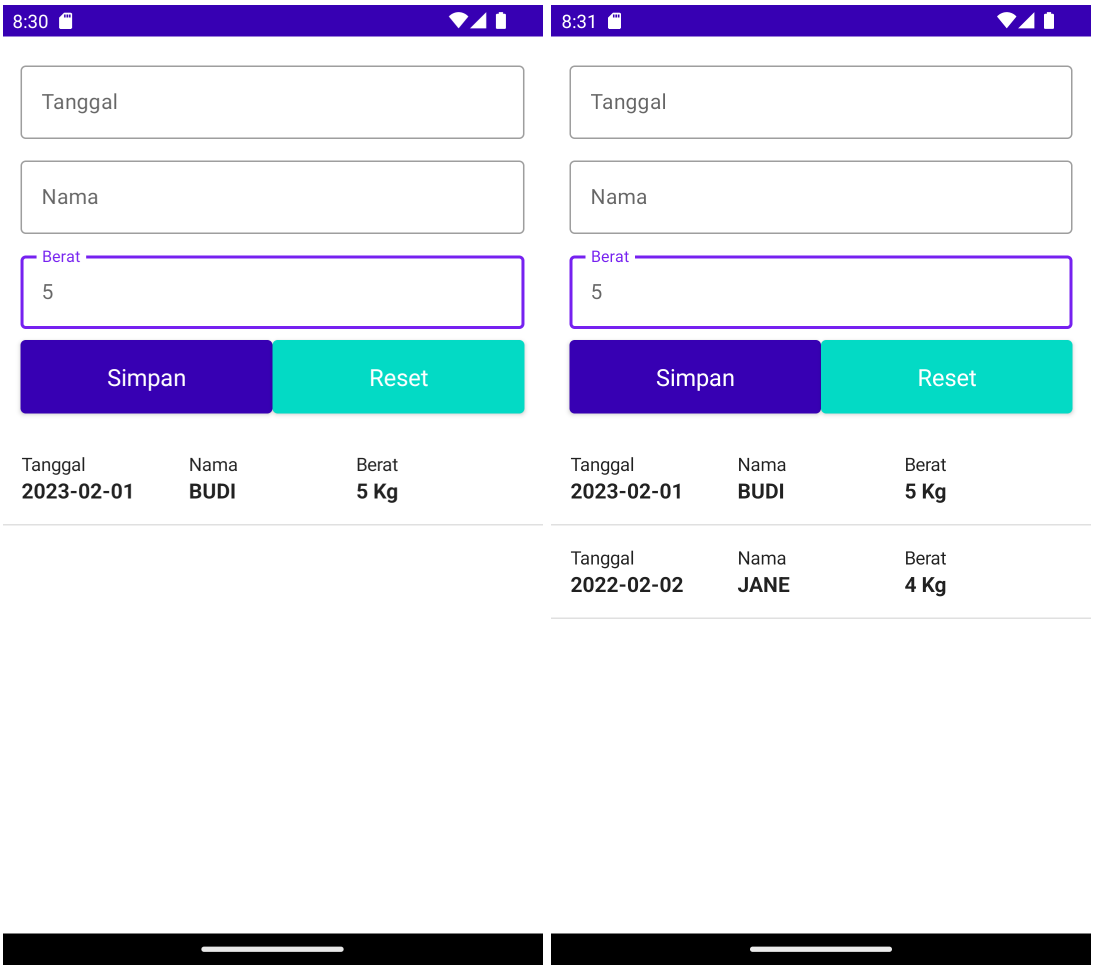
Kemudian, ubahlah file MainActivity sehingga memanggil fungsi PengelolaanSampahScreen dan bukan fungsi Greeting seperti kode berikut.

```
package id.ac.unpas.functionalcompose

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import id.ac.unpas.functionalcompose.screens.PengelolaanSampahScreen
import id.ac.unpas.functionalcompose.ui.theme.FunctionalComposeTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            FunctionalComposeTheme {
                // A surface container using the 'background' color from the
theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {
                    PengelolaanSampahScreen()
                }
            }
        }
    }
}

@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    FunctionalComposeTheme {
        PengelolaanSampahScreen()
    }
}
```

Jalankan aplikasi, coba gunakan, dan perhatikan cara kerjanya.


8:29
Tanggal
yyyy-mm-dd
Nama
Berat
Simpan
Reset


8:30
Tanggal
2023-02-01
Nama
BUDI
Berat
5
Simpan
Reset
package id.ac.un...
id.ac.unpas.func...
1
2
3
4
5
6
7
8
9
,
0
.



Tugas

Buatlah aplikasi untuk menangani inputan pencatatan keuangan harian dengan menggunakan pendekatan seperti yang ditunjukkan dalam latihan di atas. Data yang dicatat adalah tanggal, keterangan, pemasukan, dan pengeluaran. Tambahkan validasi untuk inputan tersebut, tanggal dan keterangan harus diisi, sedangkan pemasukan dan pengeluaran hanya boleh (dan wajib) diisi salah satunya.