LESSON 03

AARHUS
UNIVERSITY
DEPARTMENT OF ELECTRICAL AND COMPUTER
ENGINEERING

28 JANUARY 2024

MICHEL VEDEL HOWARD
ASSISTANT PROFESSOR

CONTENT

1. Several Activities
    1. A drawer menu with several screens (Activities)

FIRST ACTIVITY

```
class HomeActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            BasicExamplesTheme(dynamicColor = false) {
                val scope = rememberCoroutineScope()
                val drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)
                val models = listOf(
                    MenuModel(Icons.Default.Home, text: "Home") {
                    },
                    MenuModel(Icons.Default.Search, text: "Search") {
                    }
                )
                Drawer(drawerState, menuModels = models) {
                    AppScaffold({ scope.launch { drawerState.open() } }) {
                        Home()
                    }
                }
            }
        }
    }
}
```

SECOND ACTIVITY

```
class ProfileActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            BasicExamplesTheme(dynamicColor = false) {
                val scope = rememberCoroutineScope()
                val drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)
                val models = listOf(
                    MenuModel(Icons.Default.Home, text: "Home") {
                    },
                    MenuModel(Icons.Default.Person, text: "Profile") {
                    }
                )
                Drawer(drawerState, menuModels = models) {
                    AppScaffold({ scope.launch { drawerState.open() } }) {
                        Profile()
                    }
                }
            }
        }
    }
}
```

MAKING NEW ACTIVITIES

In this approach we need to get introduced to how to create Activities. We will be getting back to this again in more detail.

To start a new activity, one must create a so-called Intent object

```
val intent = Intent(this, HomeActivity::class.java)
```

The intent in this case contains the activity we are coming from(this) the activity we want to start is the next parameter.

To start the new activity, we invoke:

```
startActivity(intent)
```

which is a method on the Activity object that ComponentActivity inherits from

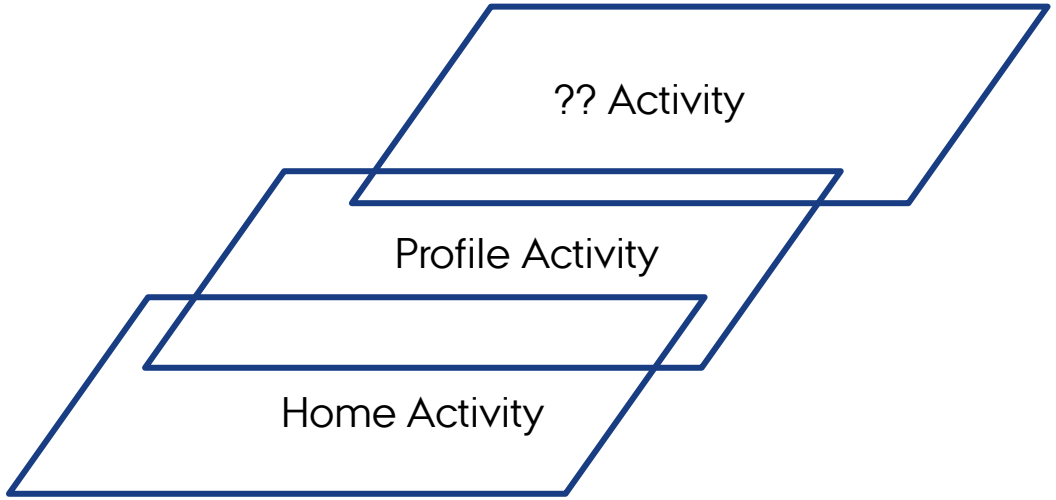
INTENT FLAGS

One should at this point recognize that activities are maintained on a **stack** called the **Back Stack.**

This means that newly created activities are pushed onto this stack.

So, is the same activity is pushed on multiple times???

Yes, if we do not take care of this it is, and this might not be our intention. So, control is needed



INTENT FLAGS

```
val intent = Intent(this, HomeActivity::class.java)
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK)
```

This flag will clear the stack, and you simply create a new activity when starting the activity

```
val intent = Intent(this, HomeActivity::class.java)
intent.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP)
```

This will reuse an activity if is at the top already

INTENT FLAGS

```
val intent = Intent(this, HomeActivity::class.java)
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
```

This flag implies:

If the activity created is on the stack already:

1. A new activity will not be created
2. All activities on the stack on top of the activity in question are removed from the stack
3. This often implies that the activities removed are destroyed.
4. The activity in question is now on top and is rendered

To start the new activity, we invoke: startActivity(intent)

INTENT FLAGS

```
val intent = Intent(this, HomeActivity::class.java)
intent.addFlags(Intent.REORDER_TO_FRONT)
```

If the activity you are launching already exists in the current task's back stack, this flag will move it to the front of the stack. All the activities above it in the stack will be kept intact.

This seems to be a good choice in the current case

Notice that you can ADD flags(uses bitwise or) And it may create conflicts that is handled by some rule.

MAKING NEW ACTIVITIES

If we want to use several activities these should be

- Created in separate classes inheriting from ComponentActivity
- And stated in the **manifest.xml**

NEW ACTIVITIES

```
class SettingsActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            BasicExamplesTheme(dynamicColor = false) {
                val scope = rememberCoroutineScope()
                val drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)
                val models = listOf(
                    MenuModel(Icons.Default.Home, text: "Home") {
                    },
                    MenuModel(Icons.Default.Search, text: "Search") {
                    }
                )
                Drawer(drawerState, menuModels = models) {
                    AppScaffold({ scope.launch { drawerState.open() } }) {
                        Settings()
                    }
                }
            }
```

NEW ACTIVITIES

Notice the duplication that is necessary

```
class ProfileActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            BasicExamplesTheme(dynamicColor = false) {
                val scope = rememberCoroutineScope()
                val drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)
                val models = listOf(
                    MenuModel(Icons.Default.Home, text: "Home") {
                    },
                    MenuModel(Icons.Default.Person, text: "Profile") {
                    }
                )
                Drawer(drawerState, menuModels = models) {
                    AppScaffold({ scope.launch { drawerState.open() } }) {
                        Profile()
                    }
                }
            }
        }
    }
```

MANIFEST

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    <application
        android:icon="@mipmap/ic_launcher"
        android:label="BasicExamples"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BasicExamples"
        tools:targetApi="31">
        <activity
            android:name=".HomeActivity"
            android:exported="true"
            android:theme="@style/Theme.BasicExamples">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ProfileActivity"
            android:exported="true"
            android:theme="@style/Theme.BasicExamples" />
        <activity
            android:name=".SettingsActivity"
            android:exported="true"
            android:theme="@style/Theme.BasicExamples" />
    </application>
```

STARTING NEW ACTIVITIES

```
class SettingsActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            BasicExamplesTheme(dynamicColor = false) {
                val scope = rememberCoroutineScope()
                val drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)
                val models = listOf(MenuModel(Icons.Default.Home, text: "Home") {
                    scope.launch {
                        drawerState.close()
                        val intent = Intent( packageContext: this@SettingsActivity, HomeActivity::class.java)
                        intent.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT)
                        startActivity(intent)
                    }
                }, MenuModel(Icons.Default.Person, text: "Profile") {
                    scope.launch {
                        drawerState.close()
                        val intent = Intent( packageContext: this@SettingsActivity, ProfileActivity::class.java)
                        intent.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT)
                        startActivity(intent)
                    }
                }, MenuModel(Icons.Default.Settings, text: "Settings") {
                    // do nothing
                }
```

PRO AND CONS

The activity approach seems to be elaborate. But it could be simplified somewhat reusing some of the code.

The main problem is that we need to manage activities on the stack. But we do not have to manage a state and a when statement in the same way.

The state approach does not make new activities, so it corresponds to a single page(screen) app. But we need to manage a state and a potentially elaborate when statement if a lot of screens are involved.

The problems of the state-management are addressed in the navigation framework

But when using activities to represent almost the same screen some duplication is to be expected.

A SIMPLE APPROACH

A third approach is much easier

To abandon the menu entirely when a menu button is activated

Then of course you would have to go back from the newly created activity

A SIMPLE APPROACH

Now there is no need for home navigation
Because the menu will be absent in the other activities.

A SIMPLE APPROACH

```
class AppActivity : ComponentActivity() {                                    10
    override fun onCreate(savedInstanceState: Bundle?) {
        setContent {
            BasicExamplesTheme(dynamicColor = false) {
                val scope = rememberCoroutineScope()
                val drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)
                val models = listOf(
                    MenuModel(Icons.Default.Person, text: "Profile") {
                        scope.launch {
                            drawerState.close()
                            val intent = Intent( packageContext: this@AppActivity, SimpleProfileActivity::class.java)
                            intent.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT)
                            startActivity(intent)
                        }
                    },
                    MenuModel(Icons.Default.Settings, text: "Settings") {
                        scope.launch {
                            drawerState.close()
                            val intent =
                                Intent( packageContext: this@AppActivity, SimpleSettingsActivity::class.java)
                            intent.addFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT)
                            startActivity(intent)
                        }
                    }
                )
                Drawer(drawerState, menuModels = models) {
                    AppScaffold({ scope.launch { drawerState.open() } }) {
                        Home()
                    }
```

A SIMPLE APPROACH

```
class SimpleProfileActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        enableEdgeToEdge()
        setContent {
            BasicExamplesTheme(dynamicColor = false) {
                Scaffold(topBar = {
                    TopAppBar(title = { }, navigationIcon = {
                        IconButton(onClick = { finish() }) {
                            Icon(
                                imageVector = Icons.AutoMirrored.Filled.ArrowBack,
                                contentDescription = "back"
                            )
                        }
                    })
                }) { innerPadding ->
                    Box(modifier = Modifier.padding(innerPadding)) {
                        Profile()
                    }
                }
            }
        }
    }
}
```

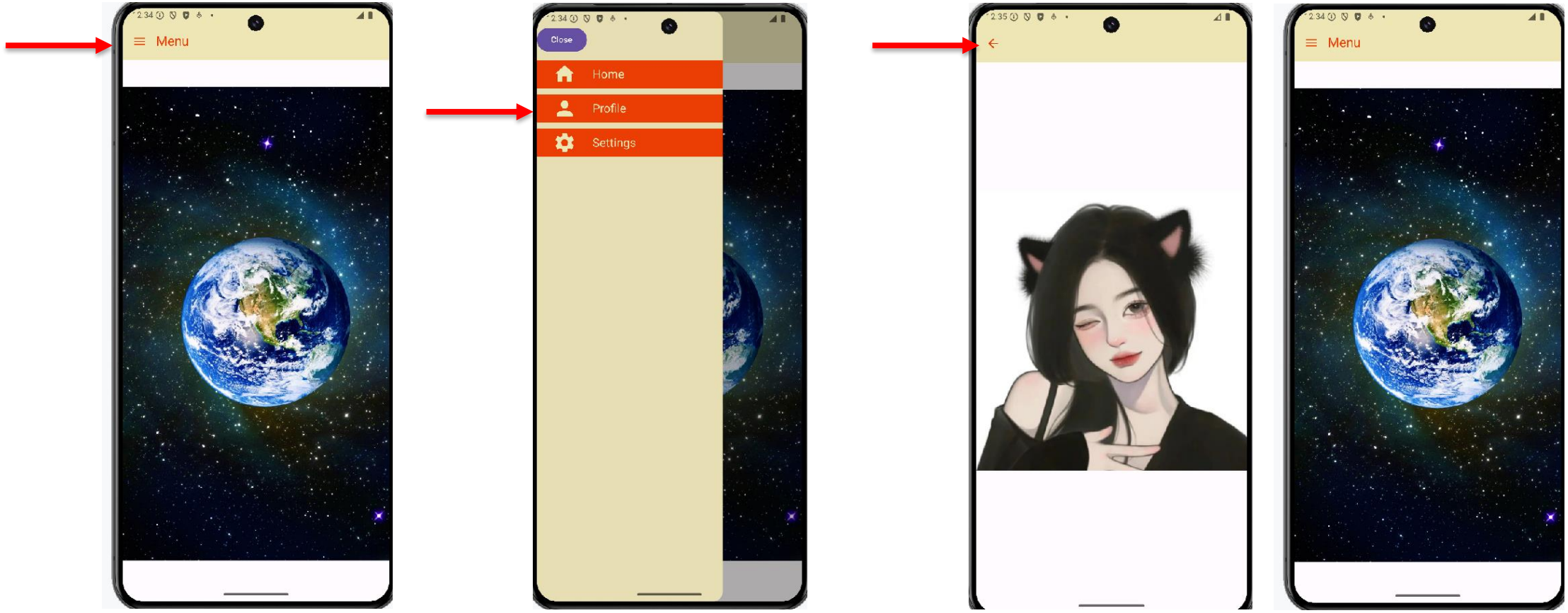
A SIMPLE APPROACH

```
class SimpleSettingsActivity : ComponentActivity() {
    @OptIn(ExperimentalMaterial3Api::class)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            BasicExamplesTheme(dynamicColor = false) {
                Scaffold(
                    topBar = {
                        TopAppBar(
                            title = { },
                            navigationIcon = {
                                IconButton(onClick = { finish() }) {
                                    Icon(
                                        imageVector = Icons.AutoMirrored.Filled.ArrowBack,
                                        contentDescription = "back"
                                    )
                                }
                            }
                        )
                    }
                ) { innerPadding ->
                    Box(modifier = Modifier.padding(innerPadding)) {
                        Settings()
                    }
                }
            }
        }
```

REMEMBER MANIFEST

```
<activity
    android:name=".AppActivity"
    android:exported="true"
    android:theme="@style/Theme.BasicExamples">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".ProfileActivity"
    android:exported="true"
    android:theme="@style/Theme.BasicExamples" />
<activity
    android:name=".SettingsActivity"
    android:exported="true"
    android:theme="@style/Theme.BasicExamples" />
<activity
    android:name=".SimpleProfileActivity"
    android:exported="true"
    android:theme="@style/Theme.BasicExamples" />
<activity
    android:name=".SimpleSettingsActivity"
    android:exported="true"
```

A THIRD APPROACH



AARHUS
UNIVERSITY