Lab 8 - Lazy List and Navigation

`COMP4107 - SDDT - HKBU - Spring2023`

Getting Started

Native Android applications can be developed using **Android Studio** (https://developer.android.com/studio). Launch it, upgrade it to the latest version. Restart the IDE after upgrade.

Clone the `InfoDay` project from https://classroom.github.com/a/t86LHCCf.

Department Screen

Create a new Kotlin file `DeptScreen.kt` and develop the department data:

```kotlin
data class Dept(val name: String, val id: String) {
    companion object {
        val data = listOf(
            Dept("Computer Science", "COMP"),
            Dept("Communication Studies", "COMS")
        )
    }
}
```

Lazy List

If you need to display **a large number of items** (or an unknown list length), using layouts like `Column` can cause **performance issues** since all items will be composed and laid out **whether visible or not**.

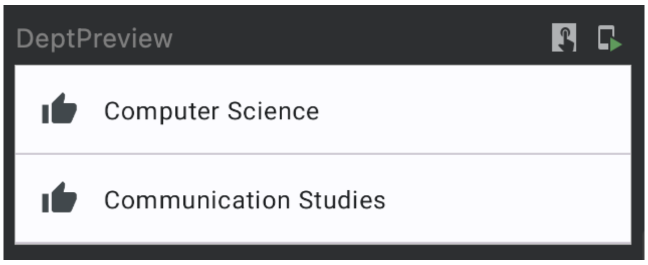
Compose offers components that **only compose and lay out visible items** in the component viewport. These include `LazyColumn` and `LazyRow`.

Here's how our `DeptScreen` composable shows using `LazyColumn`:

```kotlin
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DeptScreen() {
    LazyColumn {
        items(Dept.data) { dept ->
```

```
            ListItem(
                headlineText = { Text(dept.name) },
                leadingContent = {
                    Icon(
                        Icons.Filled.ThumbUp,
                        contentDescription = null
                    )
                }
            )
            Divider()
        }
    }
}
```

For the `items()` function, ensure you have selected **a list as the input argument**.



Computer Science

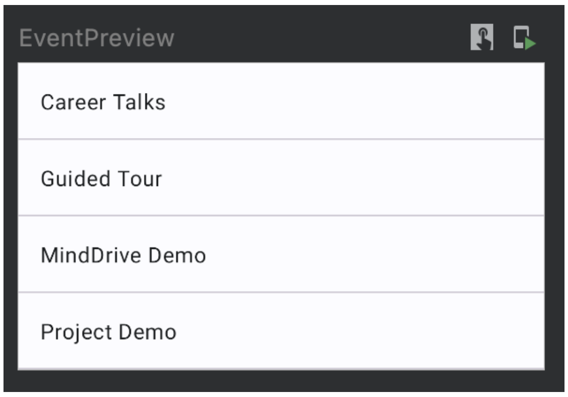
Communication Studies

Now, let's develop the preview composable.
Tapping each department should navigate us to the `Event` screen. So, let's create a new Kotlin file `EventScreen`.

Event Screen

Prepare the data as usual:

```
data class Event(val title: String, val deptId: String, var saved: Boolean) {
    companion object {
        val data = listOf(
            Event(title = "Career Talks", deptId = "COMS", saved = false),
            Event(title = "Guided Tour", deptId = "COMS", saved = true),
            Event(title = "MindDrive Demo", deptId = "COMP", saved = false),
            Event(title = "Project Demo", deptId = "COMP", saved = false)
        )
    }
}
```



Career Talks

Guided Tour

MindDrive Demo

Project Demo

Develop a **lazy column** to efficiently display all events:

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun EventScreen() {
    LazyColumn {
        items(Event.data) { event ->
            ListItem(
                headlineText = { Text(event.title) }
            )
            Divider()
        }
    }
}

@Preview(showBackground = true)
@Composable
fun EventPreview() {
    InfoDayTheme {
        EventScreen()
    }
}
```

The lazy column now displays four events.

Navigation

Now, we want to navigate from `DeptScreen` to `EventScreen`. This can be achieved using a `NavHost`. Let's return to `DeptScreen.kt` and develop this navigation component:

```
@Composable
fun DeptNav(navController: NavHostController) {

    NavHost(
        navController = navController,
        startDestination = "dept",
    ) {
        composable("dept") { DeptScreen(navController) }
        composable("event") { EventScreen() }
    }
}
```

Here, we developed a **navigation graph mapping strings to corresponding composables**. Adding a `NavHost` requires an extra dependency; include this in the module-level Gradle file:

```
dependencies {
    implementation("androidx.navigation:navigation-compose:2.5.3")
}
```

Navigate to a composable

To navigate to a destination in the **navigation graph**, call `navigate` with the route of the destination. `navigate` accepts a single `String` parameter representing the route.

Update the `DeptScreen` composable:

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DeptScreen(navController: NavHostController) {
    LazyColumn {
        items(Dept.data) { dept ->
            ListItem(
                headlineText = { Text(dept.name) },
                modifier = Modifier.clickable {
                    navController.navigate("event")
                },
                leadingContent = {
                    Icon(
                        Icons.Filled.ThumbUp,
                        contentDescription = null
                    )
                }
            )
            Divider()
        }
    }
}
```

Here, `navController.navigate("event")` will navigate to the `EventScreen` composable.

Revise the preview composable as follows:

```
@Preview(showBackground = true)
@Composable
fun DeptPreview() {
    InfoDayTheme {
        DeptNav(rememberNavController())
    }
}
```

Preview this screen and **enable interactive mode**. Tap a department to navigate to the `EventScreen`.

Navigate with arguments

To pass the `dept_id` from `DeptScreen` to `EventScreen`, we'll modify the `NavHost` slightly.

We can extract the **arguments** from the `NavBackStackEntry` in the `composable()` function's lambda.

```
composable("event/{deptId}") { backStackEntry ->
    EventScreen(backStackEntry.arguments?.getString("deptId"))
}
```

The `EventScreen` and `EventPreview()` composables have now been modified as follows:

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun EventScreen(deptId: String?) {
    LazyColumn {
        items(Event.data) { event ->
            ListItem(
                headlineText = { Text(event.title) }
            )
            Divider()
        }
    }
}

@Preview(showBackground = true)
@Composable
fun EventPreview() {
    InfoDayTheme {
        EventScreen("COMP")
    }
}
```

To **navigate with an argument** in the `DeptScreen` composable, we must append it to the route during the navigate call:

```
navController.navigate("event/${dept.id}")
```

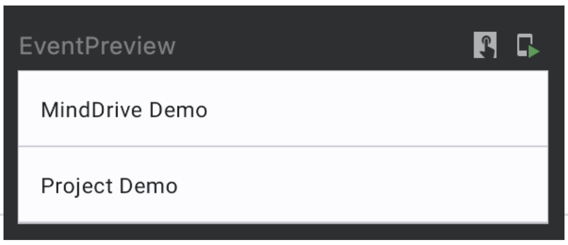
Filtering

Kotlin provides a `filter()` function which accepts a **lambda expression**. **A lambda expression takes a list element and returns a boolean**. This could be implemented as follows:

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun EventScreen(deptId: String?) {
    LazyColumn {
        items(Event.data.filter { it.deptId == deptId }) { event ->
            ListItem(
                headlineText = { Text(event.title) }
            )
            Divider()
        }
    }
}

@Preview(showBackground = true)
@Composable
fun EventPreview() {
    InfoDayTheme {
        EventScreen("COMP")
    }
}
```

Here, `it` refers to the **implicit name** of a **single parameter**.



MindDrive Demo

Project Demo

The Back Button

Finally, head back to `ScaffoldScreen` composable. We have to create a NavHost controller via the `rememberNavController()` method and pass it to our `DeptNav` composable.

```
val navController = rememberNavController()
```

The `ScaffoldScreen` component is recomposed from time to time. To retain all navigation information, we need `rememberNavController`. Its content will not be removed during recomposition.

In the `when()` statement, pass the `DeptNav` component this navigation controller.

```
when (selectedItem) {
    0 -> DeptNav(navController)
    1 -> DeptNav(navController)
    2 -> InfoScreen()
    3 -> InfoScreen()
    4 -> InfoScreen()
}
```

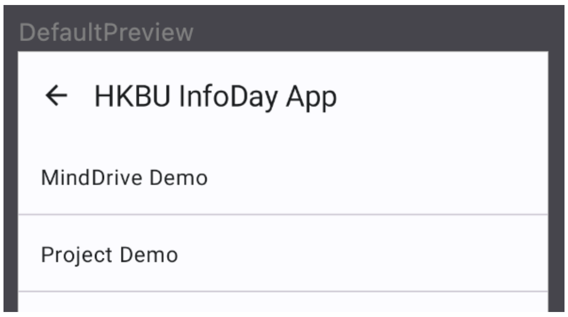
Finally, develop a **back button** when navigation isn't at the top level:

```
topBar = {
    TopAppBar(
        title = { Text("HKBU InfoDay App") },
        navigationIcon = {
            val navBackStackEntry by navController.currentBackStackEntryAsState()

            if (navBackStackEntry?.arguments?.getBoolean("topLevel") == false) {
                IconButton(
                    onClick = { navController.navigateUp() }
                ) {
                    Icon(
                        imageVector = Icons.Filled.ArrowBack,
                        contentDescription = "Back"
                    )
                }
            } else {
                null
            }
        }
    )
},
```

Commit and push your project to the private repo.

02/03/2023 12:33


DefaultPreview
←
HKBU InfoDay App
MindDrive Demo
Project Demo