

เอกสารประกอบการปฏิบัติการ

รายวิชา SC362007 Mobile Device Programming

Lab 11 การสร้างแอปพลิเคชันจัดการฐานข้อมูล SQLite (Sec2)

วัตถุประสงค์

1. เพื่อให้ให้นักศึกษาพัฒนาแอปพลิเคชันในการ เพิ่ม ลบ แก้ไข และดึงข้อมูลจากฐานข้อมูล SQLite ได้

คำชี้แจง

1. ให้นักศึกษาแคปผลลัพธ์ลงใน Word และบันทึกไฟล์เป็น PDF โดยให้ตั้งชื่อว่า Lab11_รหัสนักศึกษา(มีขีด).pdf ไฟล์หน้าจอการทำงาน ไฟล์คลาส และไฟล์ MainActivity.kt แล้ว zip รวมกันทั้ง 8 ไฟล์ โดยให้ตั้งชื่อว่า Lab11_รหัสนักศึกษา(มีขีด).zip
2. เมื่อนักศึกษาทำเสร็จเรียบร้อยแล้วให้แจ้งผู้ช่วยสอน (TA) สำหรับตรวจ และตอบคำถาม เพื่อรับคะแนนปฏิบัติการ
(คะแนนจะมาจากการตรวจภายในห้องเรียนเท่านั้น)
3. เมื่อตรวจกับ TA เรียบร้อยแล้วให้นักศึกษาส่งใน Classroom เพื่อสำรองข้อมูลงานของนักศึกษา
4. ไม่อนุญาตให้ส่งงานย้อนหลังได้ ยกเว้นกรณีลาป่วย อุบัติเหตุ หรือเหตุจำเป็น (ต้องมีใบรับรองแพทย์ หรือหลักฐานอื่น ๆ)

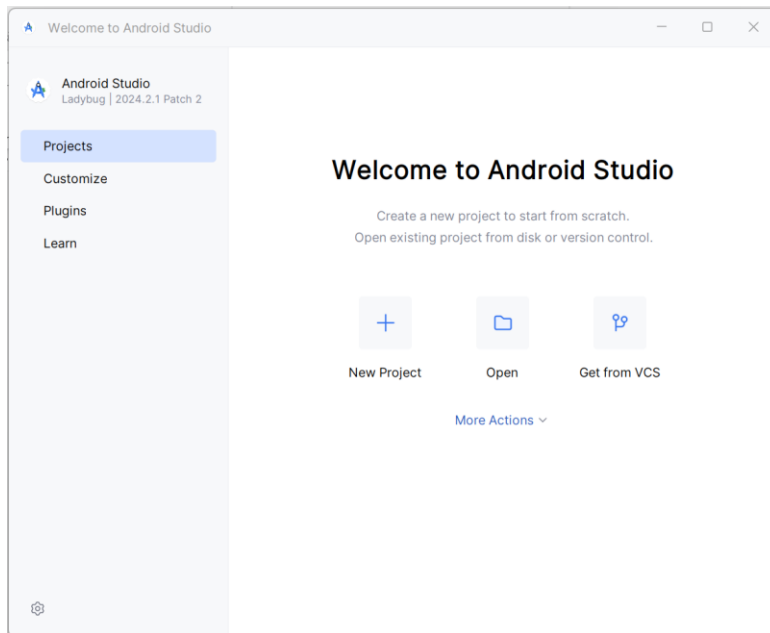
โจทย์การทำงานของแอปพลิเคชัน

การทำงานหน้าจอแรกจะเป็นการดึงข้อมูลทั้งหมดจากฐานข้อมูล SQLite มาแสดง ที่ Lazy Column ดังภาพหน้าจอดังภาพด้านล่าง และเมื่อกดปุ่ม Add Student เพื่อเพิ่มข้อมูลลงในฐานข้อมูล SQLite ได้ และในหน้าแรกส่วนของแต่ละ Item มีข้อความ Edit/Delete และเมื่อกดปุ่ม Edit/Delete เพื่อแก้ไขและลบข้อมูลในอีกหน้าจอ ดังภาพด้านล่าง เมื่อแก้ไขข้อมูลเสร็จแล้วกดที่ปุ่ม Update จะ Update ข้อมูลที่ตารางแล้วกลับไปหน้าแรก

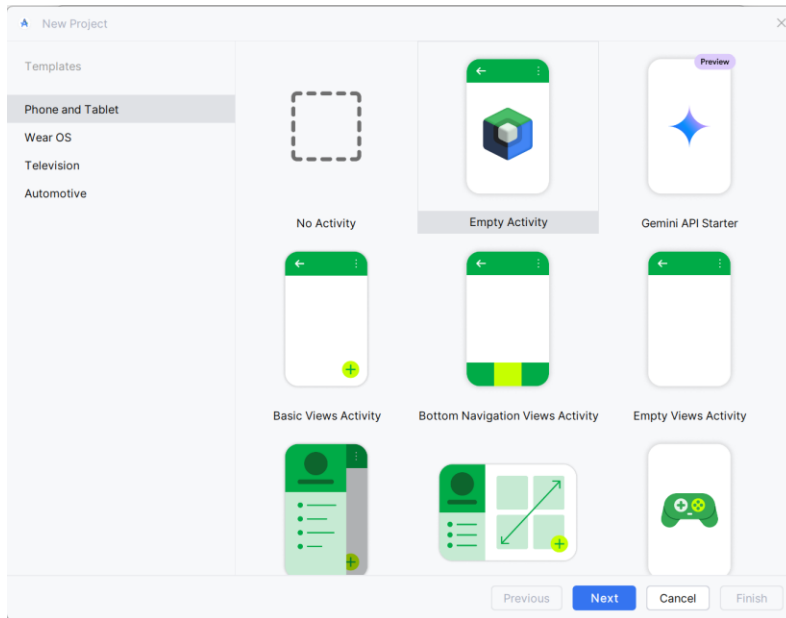
ขั้นตอนการสร้างแอปพลิเคชันส่วนของ Android Studio

การสร้างโปรเจกใหม่

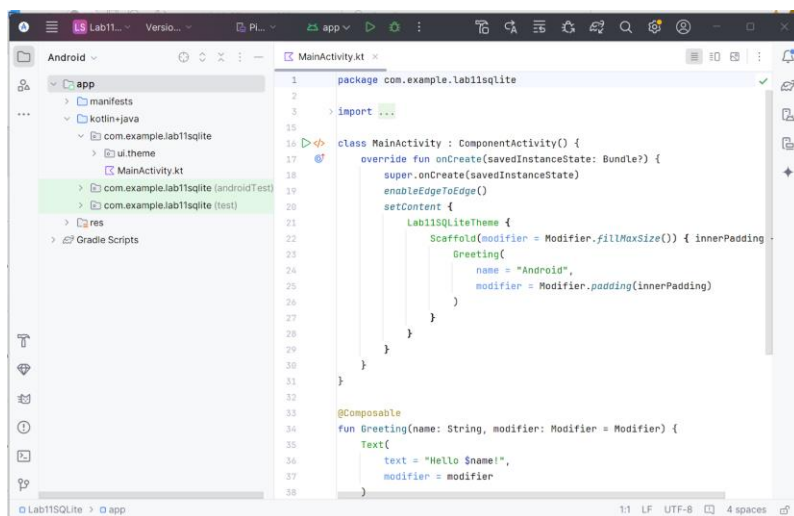
1. ให้เปิดโปรแกรม Android Studio จะแสดงหน้าจอดังนี้ ให้เลือก Create New Project



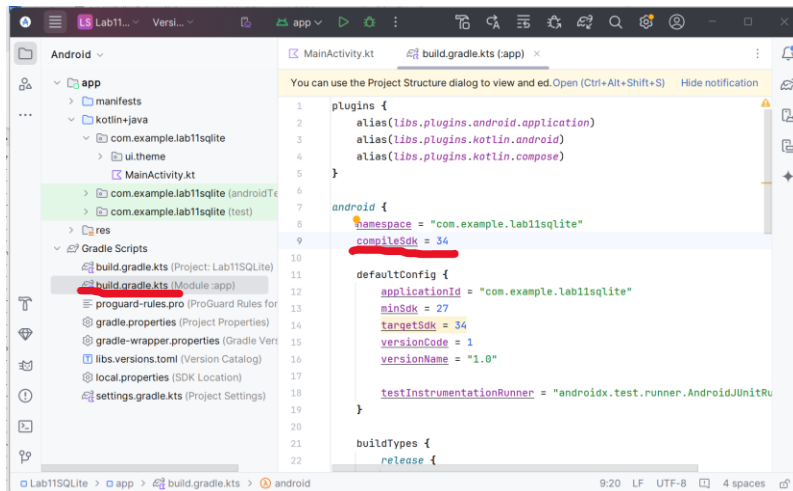
2. จากนั้นเลือก Empty Activity แล้วคลิกที่ปุ่ม Next



3. ถัดมาให้ตั้งชื่อ Application name คือ Lab11SQLite และ Language เป็น Kotlin จากนั้นกด Finish



จากนั้นไปคลิกที่ build.gradle.kts(Modules:app)



ถัดมาแก้ค่าของ compileSdk = 35

```
android {
    namespace = "com.example.lab11sqlite"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.lab11sqlite"
```

ถัดมาให้เพิ่มคำสั่งในส่วนของ plugins ดังนี้

id ("kotlin-parcelize")

```
plugins { this: PluginDependenciesSpecScope
    id("com.android.application")
    id("org.jetbrains.kotlin.android")

    id ("kotlin-parcelize")
}
```

จากนั้นให้เพิ่มคำสั่งในส่วนของ dependencies ดังนี้

```
implementation("androidx.navigation:navigation-compose:2.8.5")
```

```
implementation("androidx.lifecycle:lifecycle-runtime-compose:2.8.7")
```

แล้วให้คลิกที่ Sync New ด้านบนขวา

iles have changed since last project sync. A project sync [Sync Now](#) [Ignore these changes](#)

```
dependencies {  
  
    implementation(libs.androidx.core.ktx)  
    implementation(libs.androidx.lifecycle.runtime.ktx)  
    implementation(libs.androidx.activity.compose)  
    implementation(platform(libs.androidx.compose.bom))  
    implementation(libs.androidx.ui)  
    implementation(libs.androidx.ui.graphics)  
    implementation(libs.androidx.ui.tooling.preview)  
    implementation(libs.androidx.material3)  
  
    implementation("androidx.navigation:navigation-compose:2.8.5")  
    implementation("androidx.lifecycle:lifecycle-runtime-compose:2.8.7")  
  
    testImplementation(libs.junit)  
    androidTestImplementation(libs.androidx.junit)
```

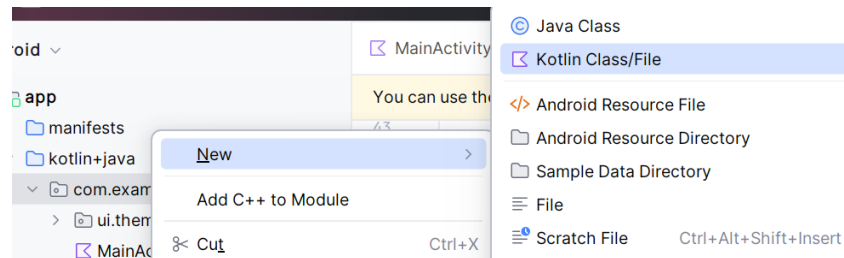
โครงสร้างของแอปพลิเคชัน ประกอบด้วยไฟล์ ดังนี้

1. ไฟล์คลาส Employee.kt คือ Data Class ของนักศึกษา
2. ไฟล์คลาส Screen.kt คือ sealed class
3. ไฟล์คลาส DatabaseHelper คือ คำสั่งในการสร้างและจัดการฐานข้อมูล SQLite
4. ไฟล์ HomeScreen.kt คือ หน้าแสดงข้อมูลของพนักงาน
5. ไฟล์ InsertScreen.kt คือ การเพิ่มข้อมูลของพนักงาน
6. ไฟล์ EditScreen.kt คือ การแก้ไขและลบข้อมูลของพนักงาน
7. ไฟล์ NavGraph.kt
8. ไฟล์ MainActivity.kt คือ หน้าแรกในการนำหน้าทุกหน้ามาแสดง

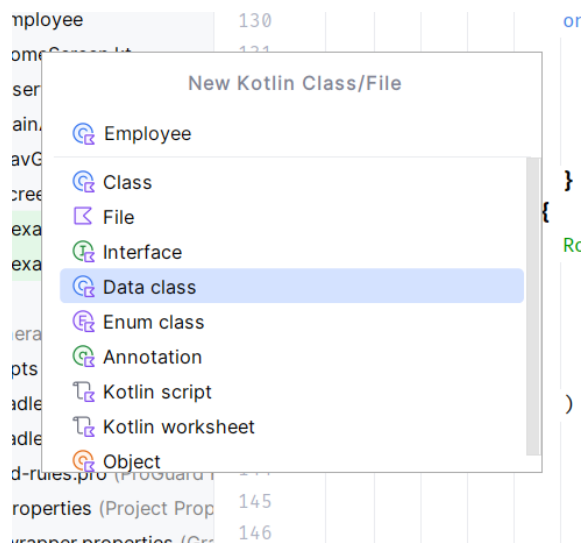
การสร้างไฟล์และเขียนคำสั่งแต่ละไฟล์

1. ไฟล์ Student.kt เป็นการสร้าง Data Class มีขั้นตอน ดังนี้

- การสร้างไฟล์ ไฟล์ Student.kt ให้ไปที่ packet แล้วคลิกขวา New >> Kotlin File/Class




- จากนั้นจะแสดงหน้าจอให้กรอกชื่อคลาสชื่อ Student และกำหนดชนิดไฟล์เป็น Data Class



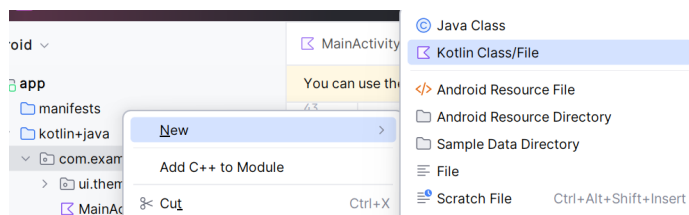
- ให้พิมพ์โค้ดคำสั่งดังภาพข้างล่าง โดยกำหนดให้มี emp_id(รหัสพนักงาน), emp_name(ชื่อพนักงาน), emp_position(ตำแหน่งของพนักงาน) เป็นประเภทสตริง และ emp_salary(เงินเดือนพนักงาน) เป็นประเภท Integer ซึ่งจะมีการเรียกใช้งาน Parcelable ด้วย

```

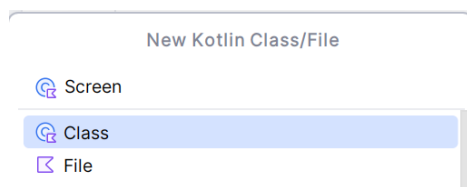
6      @Parcelize
7      data class Employee(
8          val emp_id: String,
9          val emp_name: String,
10          val emp_salary: Int,
11         val emp_position: String,|
12     ):Parcelable
13

```

2.การสร้างไฟล์คลาส Screen.kt



- จากนั้นแสดงหน้าจอให้กรอกชื่อคลาสชื่อ Screen และกำหนดชนิดไฟล์เป็น Class แล้วกดปุ่ม Enter



- จากนั้นให้พิมพ์โค้ดคำสั่งของคลาส Screen ดังภาพด้านล่าง

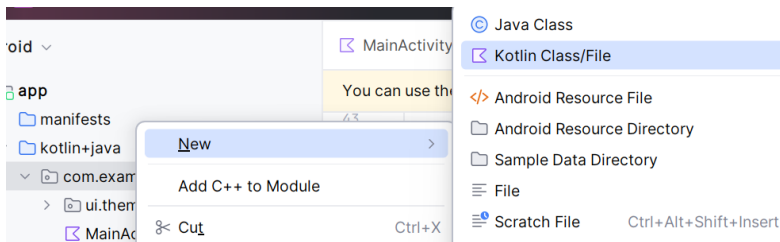
```

sealed class Screen(val route: String, val name: String) {
    data object Home: Screen(route = "home_screen", name = "Home")
    data object Insert: Screen(route = "insert_screen", name = "Insert")
    data object Edit: Screen(route = "edit_screen", name = "Edit")
}

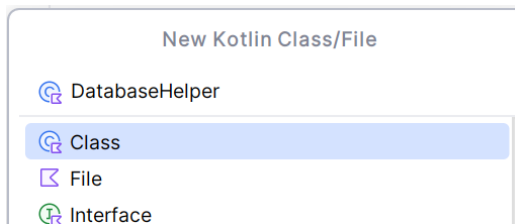
```

3. การสร้างไฟล์คลาส DatabaseHelper ใช้สร้างและจัดการฐานข้อมูลใน SQLite มีขั้นตอน ดังนี้

- การสร้างไฟล์ ไฟล์ DatabaseHelper.kt ให้ไปที่ packet แล้วคลิกขวา New >> Kotlin File/Class



- จากนั้นจะแสดงหน้าจอให้กรอกชื่อคลาสชื่อ DatabaseHelper และกำหนดชนิดไฟล์เป็น Class




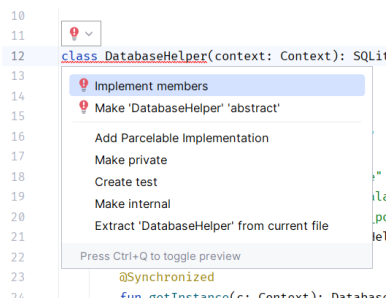
- สำหรับ Class ชื่อ DatabaseHelper สืบทอดมาจากคลาส SQLiteOpenHelper และให้เพิ่มคำสั่งและสร้าง companion object ก่อน ดังนี้

```

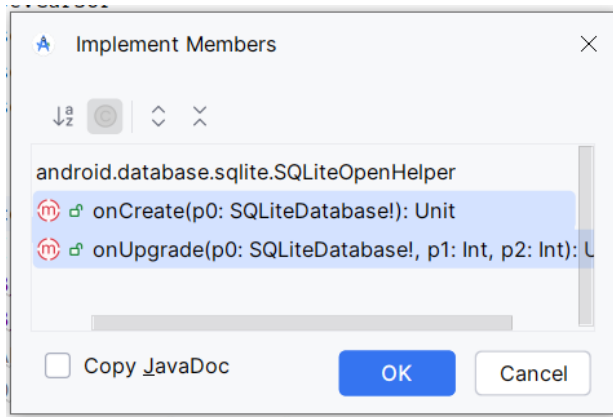
12 class DatabaseHelper(context: Context): SQLiteOpenHelper(context,DB_NAME, factory null,DB_VERSION) {
13     companion object {
14         private val DB_NAME = "employeeDB"
15         private val DB_VERSION = 1
16         private val TABLE_NAME = "employee"
17         private val COLUMN_ID = "emp_id"
18         private val COLUMN_NAME = "emp_name"
19         private val COLUMN_SALARY = "emp_salary"
20         private val COLUMN_POSITION = "emp_position"
21         private val sqliteHelper: DatabaseHelper? = null
22
23         @Synchronized
24         fun getInstance(c: Context): DatabaseHelper {
25             if (sqliteHelper == null) {
26                 return DatabaseHelper(c.applicationContext)
27             } else {
28                 return sqliteHelper
29             }
30         }
31     }
32 }

```

หลังจากนั้นให้สร้างเมธอด สามารถสร้างได้โดยเมื่อแสดงข้อความ Error ให้คลิกที่  แล้วเลือก Implement members(ถ้าไม่ขึ้น icon ให้เอาเมาส์ไปชี้ครับ) แล้วเลือก Implement members



- จากนั้นจะแสดงหน้าจอ Implement members ให้เลือกทุกเมธอดและกดที่ปุ่ม OK



โปรแกรมจะสร้างคำสั่งให้ ดังนี้

```
33 override fun onCreate(p0: SQLiteDatabase?) {  
34     TODO(reason: "Not yet implemented")  
35 }  
36  
37 override fun onUpgrade(p0: SQLiteDatabase?, p1: Int, p2: Int) {  
38     TODO(reason: "Not yet implemented")  
39 }  
40 }  
41
```

จากนั้นให้พิมพ์คำสั่งเพิ่มในส่วน **TODO** ใหม่ ดังนี้

```
33 override fun onCreate(db: SQLiteDatabase?) {  
34     val CREATE_TABLE = "CREATE TABLE $TABLE_NAME ($COLUMN_ID TEXT PRIMARY KEY, " +  
35         "$COLUMN_NAME TEXT, " + "$COLUMN_SALARY INTEGER, " + "$COLUMN_POSITION TEXT)"  
36     db?.execSQL(CREATE_TABLE)  
37  
38     val sqlInsert = "INSERT INTO $TABLE_NAME VALUES('EMP-001','David Lee',160000, 'Front end developer')"  
39     db?.execSQL(sqlInsert)  
40 }  
41  
42 override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {  
43     db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")  
44     onCreate(db)  
45 }  
46
```

จากนั้นให้เพิ่มคำสั่งเพื่อสร้างฟังก์ชันของการ Query, Insert, Update และ Delete ในคลาส DatabaseHelper ดังต่อไปนี้

- คำสั่งในการ Query โดยสร้างฟังก์ชัน getAllEmployee() ข้อมูล

```
47     @SuppressWarnings("Range")
48     fun getAllEmployee(): ArrayList<Employee>{
49         val employee = ArrayList<Employee>()
50         val db = readableDatabase
51         var cursor: Cursor? = null
52         try{ cursor = db.rawQuery( sql: "SELECT * FROM $TABLE_NAME", selectionArgs: null)
53         }catch (e: SQLiteException){
54             onCreate(db)
55             return ArrayList()
56         }
57         var emp_id :String
58         var emp_name :String
59         var emp_salary :Int
60         var emp_position: String
61         if(cursor.moveToFirst()){
62             while (!cursor.isAfterLast){
63                 emp_id = cursor.getString(cursor.getColumnIndex(COLUMN_ID))
64                 emp_name = cursor.getString(cursor.getColumnIndex(COLUMN_NAME))
65                 emp_salary = cursor.getInt(cursor.getColumnIndex(COLUMN_SALARY))
66                 emp_position = cursor.getString(cursor.getColumnIndex(COLUMN_POSITION))
67                 employee.add(Employee(emp_id,emp_name,emp_salary, emp_position))
68                 cursor.moveToNext()
69             }
70         }
71         db.close()
72         return employee
73     }
```

- คำสั่งในการ Insert โดยสร้างฟังก์ชัน insertEmployee() ข้อมูล

```
74     fun insertEmployee(emp:Employee):Long{
75         val db = writableDatabase
76         val value = ContentValues()
77         value.put(COLUMN_ID,emp.emp_id)
78         value.put(COLUMN_NAME,emp.emp_name)
79         value.put(COLUMN_SALARY,emp.emp_salary)
80         value.put(COLUMN_POSITION,emp.emp_position)
81         val success = db.insert(TABLE_NAME, nullColumnHack: null,value)
82         db.close()
83         return success
84     }
```

- คำสั่งในการ Update โดยสร้างฟังก์ชัน updateStudent() ข้อมูล

```

86 fun updateEmployee(emp:Employee):Int{
87     val db = writableDatabase
88     val value = ContentValues()
89     value.put(COLUMN_NAME, emp.emp_name)
90     value.put(COLUMN_SALARY, emp.emp_salary)
91     value.put(COLUMN_POSITION, emp.emp_position)
92     val success =db.update(TABLE_NAME,value, whereClause: "$COLUMN_ID = ?", arrayOf(emp.emp_id))
93     db.close()
94     return success
95 }
96

```

- คำสั่งในการ Delete โดยสร้างฟังก์ชัน deleteEmployee() ข้อมูล

```

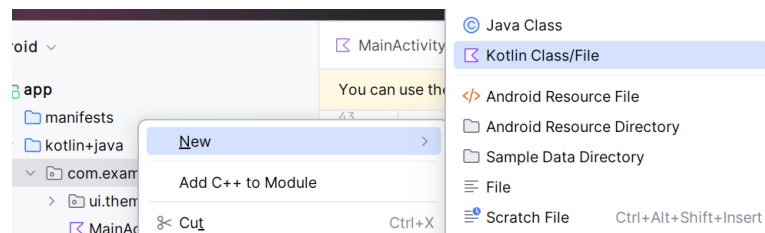
98 fun deleteEmployee(emp_id:String):Int{
99     val db = writableDatabase
100     val success = db.delete(TABLE_NAME, whereClause: "$COLUMN_ID = ?", arrayOf(emp_id))
101     db.close()
102     return success
103 }

```

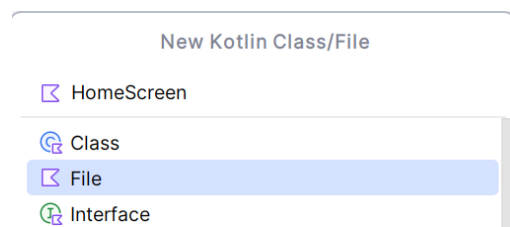
4. การสร้างไฟล์ HomeScreen.kt

- สำหรับไฟล์ HomeScreen เป็นการสร้างหน้าจอสร้างข้อมูลของพนักงานด้วย Lazy List

ซึ่งสามารถสร้างไฟล์โดยคลิกขวา New >> Kotlin File/Class



- จากนั้นแสดงหน้าจอให้กรอกชื่อไฟล์ HomeScreen และกำหนดชนิดไฟล์เป็น File แล้วกดปุ่ม Enter



- ถัดมาให้พิมพ์โค้ดคำสั่งของไฟล์ HomeScreen ดังภาพด้านล่าง

```
43 @Composable
44 fun HomeScreen(navController: NavHostController) {
45     var employeeItemList = remember { mutableStateListOf<Employee>() }
46     val contextForToast = LocalContext.current.applicationContext
47     var textFieldID by remember { mutableStateOf( value: "" ) }
48
49     ///// Check Lifecycle State
50     val lifecycleOwner = androidx.lifecycle.compose.LocalLifecycleOwner.current
51     val lifecycleState by lifecycleOwner.lifecycle.currentStateFlow.collectAsState()
52     LaunchedEffect(lifecycleState) {
53         when (lifecycleState) {
54             Lifecycle.State.DESTROYED -> {}
55             Lifecycle.State.INITIALIZED -> {}
56             Lifecycle.State.CREATED -> {}
57             Lifecycle.State.STARTED -> {}
58             Lifecycle.State.RESUMED -> {
59                 showAllData(employeeItemList, contextForToast)
60             } /// End RESUMED
61         }
62     }
63     /////
64     Column {
65         Row (
66             Modifier
67                 .fillMaxWidth()
68                 .padding(5.dp),
69                 verticalAlignment = Alignment.CenterVertically,
70                 horizontalArrangement = Arrangement.Center
71         ) {
72             //Row
73             Text(text = "Search:",
74                 fontSize = 20.sp
75             )
76             OutlinedTextField(
77                 modifier = Modifier
78                     .width(230.dp)
79                     .padding(10.dp),
80                 value = textFieldID,
81                 onValueChange = { textFieldID = it },
82                 label = { Text( text: "Employee ID" ) }
83             )
84             Button(onClick = {}) {
85                 Icon(imageVector = Icons.Default.Search, contentDescription = "Search")
86             }
87             //Row
88         }
```

```

89     Row(
90         Modifier
91             .fillMaxWidth()
92             .padding(16.dp),
93         verticalAlignment = Alignment.CenterVertically
94     ) {
95         //Row
96         Column(modifier = Modifier.weight(0.85f))
97         {
98             Text(
99                 text = "Employee Lists: ${employeeItemList.size} ",
100                 fontSize = 25.sp
101             )
102         }
103         Button(onClick = {
104             navController.navigate(Screen.Insert.route)
105         }) {
106             Text(text: "Add Employee")
107         }
108         //Row
109     }
110

```

```

111 LazyColumn(
112     verticalArrangement = Arrangement.spacedBy(6.dp)
113 ) {
114     var itemClick = Employee( emp_id: "", emp_name: "", emp_salary: 0, emp_position: "")
115     itemsIndexed(
116         items = employeeItemList,
117     ) { index, item ->
118         Card(
119             modifier = Modifier.fillMaxWidth()
120                 .padding(horizontal = 8.dp, vertical = 8.dp)
121                 .height(130.dp),
122             colors = CardDefaults.cardColors(
123                 containerColor = Color.White,
124             ),
125             elevation = CardDefaults.cardElevation(
126                 defaultElevation = 2.dp
127             ),
128             shape = RoundedCornerShape(corner = CornerSize(16.dp)),
129             onClick = {
130                 Toast.makeText(
131                     contextForToast, text: "Click on ${item.emp_name}.",
132                     Toast.LENGTH_SHORT
133                 ).show()
134             }
135         )

```

```

135     ) {
136         Row(
137             Modifier.fillMaxWidth()
138                 .height(Dp( value: 130f))
139                 .padding(16.dp),
140             verticalAlignment = Alignment.CenterVertically
141         ) {
142             Text(text = "ID: ${item.emp_id}\n" +
143                 "Name: ${item.emp_name}\n" +
144                 "Salary: ${item.emp_salary}\n" +
145                 "Position: ${item.emp_position}",
146                 Modifier.weight(0.85f)
147             )
148             TextButton(onClick = {
149                 itemClick = item
150                 navController.currentBackStackEntry?.savedStateHandle?.set( "data",
151                     Employee(item.emp_id, item.emp_name,item.emp_salary,item.emp_position)
152                 )
153                 navController.navigate(Screen.Edit.route)
154             } )
155             { Text( text: "Edit/Delete") }
156         }
157     }
158 }
159 }
160 }
161 }

```

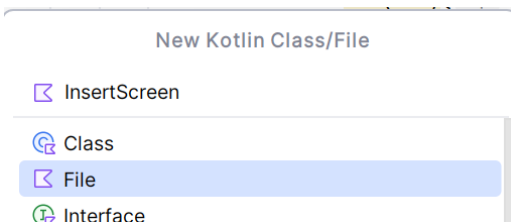
```

163 fun showAllData(employeeItemLists:MutableList<Employee>, context: Context){
164     val dbHandler= DatabaseHelper.getInstance(context)
165     dbHandler.writableDatabase
166     employeeItemLists.clear()
167     employeeItemLists.addAll(dbHandler.getAllEmployee())
168 }

```

5. การสร้างไฟล์ InsertScreen.kt

- สำหรับไฟล์ InsertScreen เป็นการสร้างหน้าจอการเพิ่มข้อมูลของนักศึกษา ซึ่งสามารถสร้างไฟล์โดยมีขั้นตอนคล้ายกับการสร้างไฟล์ HomeScreen แต่เปลี่ยนชื่อไฟล์เป็น InsertScreen และกำหนดชนิดไฟล์เป็น File แล้วกดปุ่ม Enter



- ถัดมาให้พิมพ์โค้ดคำสั่งของไฟล์ InsertScreen ดังภาพด้านล่าง

```
38  @Composable
39  fun InsertScreen(navController: NavHostController) {
40      var textFieldID by remember { mutableStateOf( value: "") }
41      var textFieldName by remember { mutableStateOf( value: "") }
42      var textFieldSalary by remember { mutableStateOf( value: "") }
43      val contextForToast = LocalContext.current
44      var selectedPosition by remember { mutableStateOf( value: "") }
45      var isEnabled by remember { mutableStateOf( value: false) }
46
47      val dbHelper= DatabaseHelper.getInstance(contextForToast)
48      dbHelper.writableDatabase
49  }
```

```

50 Column(
51     modifier = Modifier.fillMaxSize(),
52     horizontalAlignment = Alignment.CenterHorizontally,
53 ) {
54     Spacer(modifier = Modifier.height(25.dp))
55     Text(
56         text = "Insert New Employee",
57         fontSize = 25.sp
58     )
59     OutlinedTextField(
60         modifier = Modifier.fillMaxWidth()
61             .padding(16.dp),
62         value = textFieldID,
63         onChange = { textFieldID = it
64             , isEnabled = validateInput(textFieldID, textFieldName,
65                 selectedPosition, textFieldSalary)},
66         label = { Text( text: "Employee ID" ) },
67         keyboardOptions = KeyboardOptions(
68             keyboardType = KeyboardType.Text,
69             imeAction = ImeAction.Next
70         )
71     )

72     OutlinedTextField(
73         modifier = Modifier.fillMaxWidth()
74             .padding(16.dp),
75         value = textFieldName,
76         onChange = { textFieldName = it },
77         label = { Text( text: "Employee Name" ) },
78         keyboardOptions = KeyboardOptions(
79             keyboardType = KeyboardType.Text,
80             imeAction = ImeAction.Next
81         )
82     )
83     selectedPosition = MyDropdown()
84     OutlinedTextField(
85         modifier = Modifier
86             .fillMaxWidth()
87             .padding(16.dp),
88         value = textFieldSalary,
89         onChange = { textFieldSalary = it
90             , isEnabled = validateInput(textFieldID, textFieldName,
91                 selectedPosition, textFieldSalary)},
92         label = { Text( text: "Employee Salary" ) },
93         keyboardOptions = KeyboardOptions(
94             keyboardType = KeyboardType.Number,
95             imeAction = ImeAction.Done)
96     )
97
98

```



```

99     Row(
100         Modifier
101             .fillMaxWidth()
102             .padding(16.dp),
103         verticalAlignment = Alignment.CenterVertically,
104         horizontalArrangement = Arrangement.Center
105     ) {
106         Button(modifier = Modifier
107             .width(130.dp),
108             onClick = {
109                 val result = dbHandler.insertEmployee(
110                     Employee(textFieldID, textFieldName,
111                         textFieldSalary.toInt(), selectedPosition)
112                 )
113                 if(result > -1){
114                     Toast.makeText(contextForToast, text: "The student is inserted successfully",
115                         Toast.LENGTH_LONG).show()
116                 }else{
117                     Toast.makeText(contextForToast, text: "Insert Failure",
118                         Toast.LENGTH_LONG).show()
119                 }
120                 navController.navigateUp()
121             },
122             enabled = isButtonEnabled,
123         ) {
124             Text( text: "Save")
125         }

```

```

126         Spacer(modifier = Modifier.width(10.dp))
127         Button(modifier = Modifier
128             .width(130.dp),
129             onClick = {
130                 textFieldID=""
131                 textFieldName = ""
132                 navController.navigate(Screen.Home.route)
133             },
134         ) {
135             Text( text: "Cancel")
136         }
137     }
138 }
139 }
140 }

```

-ฟังก์ชัน MyDropdown()

```

142     @OptIn(ExperimentalMaterial3Api::class)
143     @Composable
144     fun MyDropdown():String {
145         val positions = listOf("Front end developer", "Back end developer","Full Stack developer" ,"DevOps")
146         var expanded by remember { mutableStateOf( value: false) }
147         var selectedPosition by remember { mutableStateOf(positions[0]) }
148         val contextForToast = LocalContext.current.applicationContext
149         Text(
150             text = "Employee Position:",
151             textAlign = TextAlign.Start,
152             modifier = Modifier.fillMaxWidth()
153                 .padding(start = 16.dp, top = 10.dp),
154         )
155         ExposedDropdownMenuBox(
156             expanded = true,
157             modifier = Modifier
158                 .padding(top = 8.dp),
159             onExpandedChange = {
160                 expanded = !expanded
161             }
162         ) {
163
164             OutlinedTextField(
165                 modifier = Modifier
166                     .menuAnchor(),
167                 readOnly = true,
168                 value = selectedPosition,
169                 onValueChange = {},
170                 label = { Text( text: "Position")},
171                 trailingIcon = { ExposedDropdownMenuDefaults.TrailingIcon(expanded = expanded)},
172                 colors = ExposedDropdownMenuDefaults.textFieldColors(),
173             )
174             //Menu
175             ExposedDropdownMenu(
176                 expanded = expanded,
177                 onDismissRequest = {
178                     expanded = false
179                 },
180                 modifier = Modifier.fillMaxWidth()
181             ){

```

```

180     ){
181         positions.forEach{ selectedOption ->
182             DropdownMenuItem(
183                 text = {Text(selectedOption)},
184                 onClick = {
185                     selectedPosition = selectedOption
186                     expanded = false
187                     Toast.makeText(contextForToast, text: "Item is $selectedPosition", Toast.LENGTH_LONG).show()
188                 },
189                 contentPadding = ExposedDropdownMenuDefaults.ItemContentPadding,
190             )
191         }
192     }
193 }
194 }
195 return selectedPosition
196 }
197
198 fun validateInput(emp_id: String, emp_name: String, emp_salary: String, emp_position:String)
199 : Boolean {
200     return emp_id.isNotEmpty() && emp_name.isNotEmpty()
201         && emp_salary.isNotEmpty() && emp_position.isNotEmpty()
202 }

```

6. การสร้างไฟล์ EditScreen.kt

- สำหรับไฟล์ EditScreen

เป็นการสร้างหน้าจอการแก้ไขและลบข้อมูลของพนักงานโดยมีขั้นตอนคล้ายกับการสร้างไฟล์ HomeScreen

แต่เปลี่ยนชื่อไฟล์เป็น EditScreen

- ถัดมาให้พิมพ์โค้ดคำสั่งของไฟล์ EditScreen ดังภาพด้านล่าง และส่วนที่เหลือเป็นงาน Lab ให้นักศึกษาเขียนเอง

```

41     @Composable
42     fun EditScreen(navController: NavHostController) {
43         val data =
44             navController.previousBackStackEntry?.savedStateHandle?.get<Employee>("data")
45             ?: Employee( emp_id: "", emp_name: "", emp_salary: 0, emp_position: "" )
46         //ให้นักศึกษาเขียนเอง
47     }

```

7. การสร้างไฟล์ NavGraph.kt

- สำหรับไฟล์ NavGraph เป็นการสร้างเส้นทางของการนำทางไปยังหน้าหน้าที่ได้สร้างไว้ ซึ่งสามารถสร้างไฟล์โดยมีขั้นตอนคล้ายกับการสร้างไฟล์ HomeScreen แต่เปลี่ยนชื่อไฟล์เป็น NavGraph และกำหนดชนิดไฟล์เป็น File แล้วกดปุ่ม Enter



- ถัดมาให้พิมพ์โค้ดคำสั่งของไฟล์ NavGraph ดังภาพด้านล่าง

```

8     @Composable
9     fun NavGraph(navController: NavHostController) {
10         NavHost(
11             navController = navController,
12             startDestination = Screen.Home.route
13         ) {
14             composable(route = Screen.Home.route) {
15                 HomeScreen(navController)
16             }
17             composable(route = Screen.Insert.route) {
18                 InsertScreen(navController)
19             }
20             composable(route = Screen.Edit.route) {
21                 EditScreen(navController)
22             }
23         }
24     }

```

8. การเพิ่มคำสั่งในไฟล์ MainActivity.kt

ในไฟล์นี้เป็นหน้าจอให้การแสดงหน้าจอแต่ละหน้าที่ได้สร้างไว้ โดยเขียนฟังก์ชัน MyScreen() เพิ่ม ดังนี้

```

31     @Composable
32     fun MyScreen() {
33         val navController = rememberNavController()
34         Column(
35             modifier = Modifier
36                 .fillMaxSize(),
37             horizontalAlignment = Alignment.CenterHorizontally
38         ) {
39         }
40         NavGraph(navController = navController)
41     }

```

จากนั้นให้เพิ่มคำสั่งในส่วน setContent เพื่อเรียกฟังก์ชัน MyScreen()

```

19  class MainActivity : ComponentActivity() {
20      override fun onCreate(savedInstanceState: Bundle?) {
21          super.onCreate(savedInstanceState)
22          enableEdgeToEdge()
23          setContent {
24              Lab_11_Sec2Version2Theme {
25                  MyScreen()
26              }
27          }
28      }
29  }

```

จากนั้นให้ทดลอง Run โปรแกรม

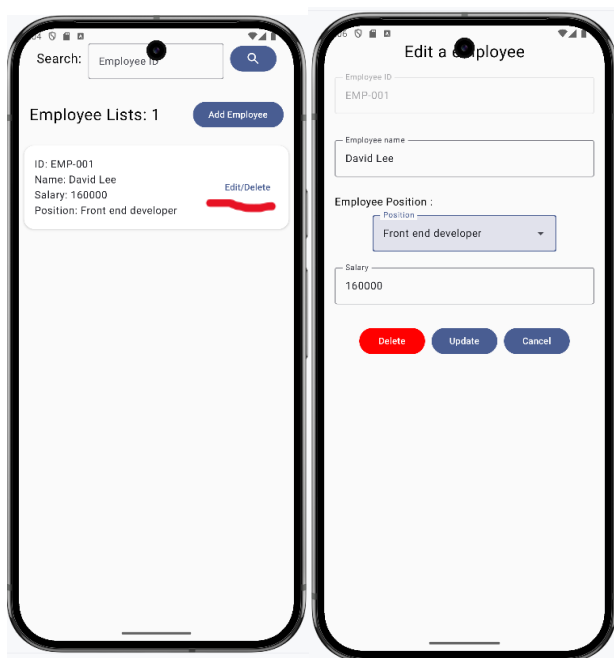
+++++

งาน Lab ข้อที่ 1 ให้เพิ่มคำสั่งเพื่อสร้างหน้าจอการทำงานของหน้า EditScreen

เพื่อรับข้อมูลจากหน้า HomeScreen มาแสดงดังภาพด้านล่าง

โดยข้อมูลที่กำหนดในช่องกรอกแต่ละช่องและ Dropdown จะต้องเป็นข้อมูลเดียวกันกับ ข้อมูลที่รับมาจากหน้า HomeScreen และ Keyboard type ในช่องกรอกข้อมูล Salary จะต้องแสดงให้กดแค่ตัวเลขเท่านั้น และ หน้า UI ปุ่ม Delete จะต้องเป็นสีแดง และ กลุ่มของปุ่มจะต้องอยู่ตรงกลาง และ ช่องกรอก emp_id จะต้องกรอกไม่ได้

ดังรูปภาพด้านล่าง



งาน Lab ข้อที่ 2 ให้เขียนคำสั่งในหน้า EditScreen ให้เมื่อกดปุ่ม Update แล้วข้อมูลในฐานข้อมูลถูกแก้ไข และให้กลับไปยังหน้า HomeScreen แสดงข้อมูลที่แก้ไขแล้ว ดังภาพด้านล่าง

