บทที่ 6

ความรู้เบื้องต้นเกี่ยวกับการพัฒนา Android ด้วย Kotlin

อ.ดร.พงษ์ดนัย จิตตวิสุทธิกุล

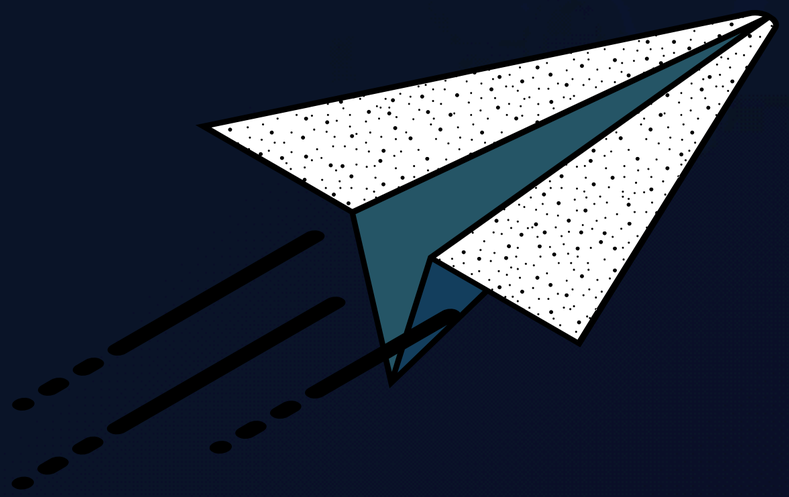
Topic

3 หัวข้อหลัก:

- Android Studio
- ความรู้เบื้องต้นเกี่ยวกับภาษา Kotlin
- ปฏิบัติการในห้องเรียน

Objective behavior

3 วัตถุประสงค์เชิงพฤติกรรม:



ระบุหน้าที่ของเครื่องมือพัฒนาหลัก (Android Studio) และสามารถสร้างโปรเจกต์ Android พื้นฐานด้วยเทมเพลตที่ถูกต้องได้

อธิบายข้อดีของภาษา Kotlin และสามารถประยุกต์ใช้ไวยากรณ์พื้นฐานในการเขียนโปรแกรมได้

แก้ไขและรันแอปพลิเคชัน “Hello, World” ด้วย Jetpack Compose และสามารถใช้เครื่องมือพื้นฐานในการดีบักได้

จากทฤษฎีสู่การปฏิบัติ

สัปดาห์นี้เราจะเปลี่ยนบทบาทจาก
"ผู้วิเคราะห์" มาเป็น "ผู้สร้าง"
เริ่มต้นการเดินทางสู่การลงมือปฏิบัติจริง
ด้วยเครื่องมือระดับมืออาชีพ

Android Studio: ศูนย์บัญชาการนักพัฒนา

- **Official IDE:** สภาพแวดล้อมการพัฒนาแบบเบ็ดเสร็จอย่างเป็นทางการจาก Google สร้างบนพื้นฐานของ IntelliJ IDEA
- **Intelligent Code Editor:** ระบบช่วยเขียนโค้ด เติมโค้ดอัตโนมัติ และตรวจจับข้อผิดพลาดสำหรับ Kotlin และ Java
- **Flexible Build System:** ใช้ Gradle ในการจัดการไลบรารีและสร้างไฟล์แอปพลิเคชัน (APK/App Bundle)
- **Fast Emulator:** โปรแกรมจำลองการทำงานของมือถือเพื่อทดสอบแอปได้ทันที
- **Profiler and Debugger:** สำหรับตรวจสอบประสิทธิภาพของแอป การใช้หน่วยความจำ และค้นหาข้อผิดพลาด

หน้าต่างการทำงานหลัก (Workspace)

Project View

มุมมองโครงสร้างไฟล์ จัดกลุ่มไฟล์ตามหน้าที่ ทำให้ง่ายต่อการค้นหาไฟล์ Source Code (java/kotlin) และไฟล์ Resources (res)

Code Editor

พื้นที่ทำงานหลักสำหรับไฟล์ใช้เขียนทั้งตรรกะการทำงาน (Logic) และออกแบบหน้าจอ (UI) ด้วย Jetpack Compose

Logcat & Debugger

หน้าต่างสำคัญที่อยู่ด้านล่าง ใช้แสดงข้อความ Log และตรวจสอบค้นหาข้อผิดพลาดที่เกิดขึ้นขณะแอปทำงาน

ทำไมต้อง Kotlin?

Google แนะนำให้ใช้ Kotlin เป็นอันดับแรกสำหรับการพัฒนา Android ตั้งแต่ปี 2019 ด้วยเหตุผลหลัก 4 ประการ:

1. กระชับกว่า (Concise): ลดโค้ดที่ต้องเขียนซ้ำซาก (Boilerplate) เช่น Data Class ในบรรทัดเดียว
2. ปลอดภัยกว่า (Safe): มีระบบ Null Safety ป้องกันแอปแครช
3. ทันสมัย (Modern): รองรับ Functional Programming และ Coroutines
4. เข้ากันได้ 100%: ทำงานร่วมกับโค้ดและไลบรารี Java เดิมได้ทันที

ไวยากรณ์พื้นฐาน: ตัวแปรและชนิดข้อมูล

การประกาศตัวแปร

เปรียบเสมือน "กล่อง" เก็บข้อมูล

`val` (Value): เปลี่ยนแปลงค่าไม่ได้
เหมือนเขียนป้ายชื่อกล่องด้วย "ปากกา"

```
val myName = "Somchai"
```

`var` (Variable): เปลี่ยนแปลงค่าได้
เหมือนเขียนป้ายชื่อกล่องด้วย "ดินสอ"

```
var score = 100; score = 120
```

ชนิดข้อมูล (Data Types)

ประเภทของที่เก็บในกล่อง

`String` : ข้อความ (เช่น "Hello")

`Int` : จำนวนเต็ม (เช่น 10, -99)

`Double` : ทศนิยม (เช่น 3.14)

`Boolean` : ค่าความจริง

(`true` / `false`)

ความปลอดภัย (Null Safety) และ ฟังก์ชัน

ระบบ Null Safety

แก้ปัญหาแอปเด้งจาก `NullPointerException`
โดยบังคับให้เราแยกระหว่างตัวแปรที่
"ห้ามว่าง" กับ "ว่างได้"

✔ Non-Nullable (ค่าห้ามเป็น Null)
var a: String = "abc"

? Nullable (ต้องเติม `?` ต่อท้าย)
var b: String? = null

*คอมไพเลอร์จะบังคับให้เราเช็คค่า null ก่อนใช้งานเสมอ

การสร้างฟังก์ชัน (Functions)

เปรียบเสมือน "สูตรอาหาร" ที่ตั้งชื่อไว้
เรียกใช้ซ้ำได้ โดยใช้คีย์เวิร์ด `fun`

```
fun greet(name: String) {
    println("Hello, $name!")
}
```

ประกอบด้วย: คำสั่ง fun, ชื่อฟังก์ชัน,
พารามิเตอร์ (วัตถุดิบ)
และ Body (ขั้นตอนในปีกกา)

การควบคุมการทำงาน (Control Flow)

คำสั่ง if / else (การตัดสินใจ 2 ทาง) ใช้เมื่อโปรแกรมต้องเลือกทำอย่างใดอย่างหนึ่ง "ถ้าเงื่อนไขเป็นจริง ให้ทำแบบแรก มิฉะนั้นให้ทำแบบที่สอง"

คำสั่ง when (การตัดสินใจหลายทาง) ทำหน้าที่คล้าย switch ในภาษาอื่น แต่มีความยืดหยุ่นกว่ามาก เหมาะสำหรับการตรวจสอบตัวแปรที่มีค่าเป็นไปได้หลากหลาย (เช่น การตัดเกรด A, B, C)

ตัวอย่างการใช้ when when (grade) { "A" -> println("Excellent"), else -> println("Keep trying") } ช่วยให้โค้ดอ่านง่ายและสะอาดตา

ปฏิบัติการที่ 1: สร้างแอป "Hello, World"

ขั้นตอนการปฏิบัติ:

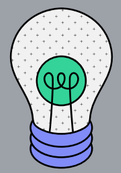
1. สร้างโปรเจกต์ใหม่: เลือกเทมเพลต Empty Activity (รองรับ Jetpack Compose)
   ตั้งชื่อแอปว่า MyFirstApp
2. แก้ไขโค้ด Kotlin: เปิด `MainActivity.kt` ค้นหาฟังก์ชัน `@Composable fun Greeting`
3. ปรับแต่ง UI: เปลี่ยนข้อความจาก `"Hello $name!"` เป็น `"Hello, [ชื่อของคุณ]!"`
4. ดูผลลัพธ์ (Preview): คลิกแท็บ Split ด้านขวาเพื่อดูหน้าตาแอป หรือกด Run บน
   Emulator

ปฏิบัติการที่ 2: พื้นฐานภาษา Kotlin (Playground)

สนามเด็กเล่นสำหรับทดสอบโค้ด:

1. สร้างไฟล์ใหม่: คลิกขวาที่ Package Name เลือก New > Kotlin Class/File สร้างไฟล์ชื่อ Practice.kt
2. สร้าง Main Function: พิมพ์ `fun main() { }` เพื่อเป็นจุดเริ่มต้นรันโปรแกรมแยก
3. ทดสอบตัวแปร: ลองประกาศ val และ var แล้วใช้คำสั่ง `println()` เพื่อแสดงผลลัพธ์ผ่านหน้าต่าง Run ด้านล่าง
4. ทดสอบ Logcat: กลับไปที่ MainActivity.kt เพิ่มคำสั่ง `Log.d("Tag", "Message")` ใน `onCreate()` เพื่อทดสอบระบบ Debugger

Questions & Discussion



วันนี้เราได้เรียนรู้การใช้งาน Android Studio เบื้องต้น และไวยากรณ์พื้นฐานของภาษา Kotlin
มีคำถามหรือข้อสงสัยเพิ่มเติมหรือไม่?