บทที่ 12

การทดสอบและแก้ไขข้อบกพร่องซอฟต์แวร์

อ.ดร.พงษ์ดนัย จิตตวิสุทธิกุล

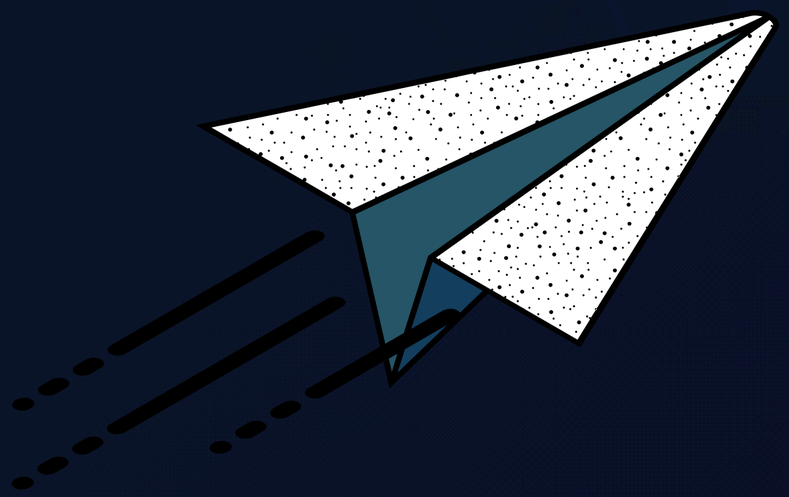
Topic

4 หัวข้อหลัก:

- กลยุทธ์การทดสอบซอฟต์แวร์
- การเจาะลึกกลยุทธ์การทดสอบ
- การแก้ไขข้อบกพร่อง
- ปฏิบัติการในห้องเรียน

Objective behavior

3 วัตถุประสงค์เชิงพฤติกรรม:



เขียน Unit Tests ด้วย JUnit และ Assertions ครอบคลุมกรณีปกติและกรณีขอบเขต (Boundary Cases) ได้

แยกแยะ Unit, Integration และ UI Test รวมถึงเลือกใช้โฟลเดอร์ test และ androidTest ได้ถูกต้อง

ใช้เครื่องมือ Debugger และ Logcat เพื่อแกะรอยปัญหา ตรวจสอบตัวแปร และหาต้นตอการแครชได้

ทำไมต้องทดสอบซอฟต์แวร์?

จาก "ผู้สร้าง" สู่ "ผู้ตรวจสอบ"

ฟีเจอร์ที่ทำงานผิดพลาด (บันทึกข้อมูลผิด) สร้างความเสียหายได้มากกว่าการไม่มีฟีเจอร์นั้นเสียอีก

- **Validation:** ทำงานตรงตามที่ลูกค้าต้องการหรือไม่?
- **Defect Testing:** หาจุดบกพร่องที่ทำให้พฤติกรรมโปรแกรมผิดเพี้ยน
- **Automated Testing:** ช่วยให้เจอ Bug เร็ว, Refactor โค้ดได้อย่างปลอดภัย และลด Technical Debt

กรอบแนวคิด: The Test Pyramid

สัดส่วนของการทดสอบที่โปรเจกต์คุณภาพควรมี หลีกเลี่ยงรูปแบบ "กรวยไอศกรีม" (Ice Cream Cone)

UI Tests (ยอดพีระมิด)

มีจำนวนน้อยที่สุด (ช้าที่สุด, พังง่าย, บำรุงรักษายาก)

Integration Tests (กลางพีระมิด)

ทดสอบการทำงานร่วมกันของโมดูล (เช่น DAO คุยกับ Room)

Unit Tests (ฐานพีระมิด)

มีจำนวนมากที่สุด (เร็วระดับมิลลิวินาที, แยกส่วนปัญหาได้ง่าย, ราคาถูก)

โครงสร้างโฟลเดอร์ Test ใน Android

การตัดสินใจที่สำคัญ: เราจะรันการทดสอบไว้ที่ไหน?

Local Tests (โฟลเดอร์ test)

⚡ **รันบน:** JVM (คอมพิวเตอร์ของเรา)

✅ **ข้อดี:** รวดเร็วมาก (มิลลิวินาที) เหมาะสำหรับฐานของ Test Pyramid

❌ **ข้อจำกัด:** เข้าถึง Android API (เช่น Context, Activity) ไม่ได้ ถ้าเรียกใช้จะพังทันที

**วิธีแก้: ต้องใช้ "Mockito" เพื่อจำลอง (Mock) ข้อมูล*

Instrumented (โฟลเดอร์ androidTest)

📱 **รันบน:** Emulator หรืออุปกรณ์จริง

✅ **ข้อดี:** เข้าถึง Android Framework APIs ได้อย่างสมบูรณ์แบบ 100%

❌ **ข้อจำกัด:** ช้ามาก! เพราะต้อง Build, ติดตั้ง (Deploy) และรันแอปบนอุปกรณ์จริง

เครื่องมือ Unit Test: JUnit & Mockito

การตัดสินใจที่สำคัญ: เราจะรันการทดสอบไว้ที่ไหน?

JUnit (Assertions)

ตรวจสอบว่าผลลัพธ์จริง (Actual) ตรงกับที่คาดหวัง (Expected) หรือไม่

- **assertEquals(A, B) :** เช็คว่าเท่ากัน
- **assertTrue(cond) :** เช็คเงื่อนไข

Mockito

สร้าง "Object จำลอง" เพื่อไม่ให้โค้ดพังเวลาเรียก Android API ในโฟลเดอร์ test

- **@Mock:** สร้างเปลือกจำลอง (Empty Shell)
- **@Spy:** หุ้ม Object ที่มีอยู่จริง
- **when(..).thenReturn(..):** สั่งพฤติกรรมจำลอง

Integration & UI Testing

Integration Test: Room Database มักทำใน androidTest โดยใช้ In-memory Database เพื่อให้ทดสอบได้เร็วและไม่ทิ้งขยะไว้ในระบบจริง

Integration Test: Migration เมื่อเปลี่ยนโครงสร้างฐานข้อมูล (เพิ่มคอลัมน์) ต้องทดสอบด้วย MigrationTestHelper เพื่อให้แน่ใจว่าข้อมูลเก่าไม่หาย

UI Test: Espresso Trinity การเขียน Espresso Test มี 3 องค์ประกอบ: 1. หาปุ่ม (ViewMatchers) -> 2. สั่งกด/พิมพ์ (ViewActions) -> 3. ตรวจสอบผลลัพธ์บนจอ (ViewAssertions)

เทคนิคการแก้ไขข้อบกพร่อง (Debugging Demonstration)

เปลี่ยนจากผู้สังเกตการณ์ สู่การเป็น "นักสืบ"

Logcat vs. Debugger

Mechanics ต่างกันนิดเดียว สร้าง Aesthetics ทางการศึกษาที่ต่างกันสิ้นเชิง

Logcat: เล่าเรื่องราว (Story)

เหมาะสำหรับการวิเคราะห์เชิงเวลา (Temporal Analysis) ดูลำดับเหตุการณ์

- Log.d() (Debug), Log.e() (Error)
- การกรอง (Filter) สำคัญมาก:
- package:mine (แสดงเฉพาะแอปเรา)
- level:ERROR (หาจุดพัง)

Debugger: ถ่ายภาพนิ่ง (Snapshot)

เหมาะสำหรับการวิเคราะห์สถานะ (State Analysis) ณ บรรทัดที่แอปพัง

- **Breakpoints:** จุดสีแดงสั่งหยุดโปรแกรม
- **Stepping:** F8 (ข้ามบรรทัด), F7 (เจาะลึกเข้าไป), Shift+F8 (ออก)
- **Evaluate Expression:** อาวุธลับ! ทดลองรันโค้ดสดๆ ระหว่างแอปหยุดเพื่อหาสาเหตุ NullPointer

ปฏิบัติการ: Unit Test & Debugging

Lab 1: เขียน Unit Test (JUnit)

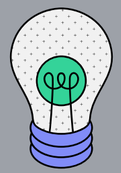
ทดสอบคลาส GradeCalculator

- สร้างเทสเคสกรณีปกติ (ได้ 95 คืนค่า 'A')
- สร้างเทส **Boundary Cases (เงื่อนไขขอบเขต)** เช่น ได้ 79 ต้องเป็น C (ไม่ใช่ B)
- ทดสอบ Error (คะแนน -10 ต้องโยน Exception)

Lab 2: ล่า Bug (NullPointerException)

สร้างแอปที่ตั้งใจส่ง Intent ด้วย Key ผิด "WRONG_KEY" แล้วใช้ Debugger ไล่ดูว่าตัวแปร message กลายเป็น null ที่บรรทัดไหน

Questions & Discussion



โค้ดที่ดี ไม่ใช่แค่โค้ดที่ทำงานได้
แต่คือโค้ดที่ถูกพิสูจน์แล้วว่า "ทดสอบได้
(Testable)" และบำรุงรักษาได้ง่าย