Batch #14 / Android Class Assignment - Week 4

1. What is Polymorphism? Try to explain in Mandarin.

Polymorphism 多型是 object-oriented language 建立在 Inheritance 和 Encapsulation 的原則，運用 class 的繼承關係，superclass 可以作為 subclass 的通用型態，展現物件類別的多樣性。因此若有繼承關係，變數宣告時，可以將物件提升成為 superclass 位階；使用上若是符合子類別型態，也可透過 Cast 強制將物件降為 subclass 類別。通常相同類型的物件類別，可以提取出共同的 method，建立一個不宣告內容的抽象類別，讓繼承物件可以依照不同狀況變換資料型態。因為 abstract class 和 interface 的建立，可以降低單一物件對程式的相依性，並且利用 abstract class 建立類別規格，再由 subclass overriding 分別定義和實作，各自發展特色。雖然 abstract class 無法建立實體，但可以透過 Polymorphism 以 superclass 為基底，傳入各 subclass 對應型別建立物件，再由 superclass 做統一操作。

3. What is the Android Jetpack?

Android Jetpack 是 Google 發布的 android 開發者輔助工具，將一系列的 Android Library 建立標準化，對現有程式庫進行分類，讓程式碼編寫更為便捷，並為不同的版本提供相容。Jetpack 的組件主要有 4 種面向，Foundation、Architecture、Behavior、UI，由各組件提供特定的功能服務，可以提供多個組件協同使用，減少系統崩潰和內存洩漏發生，建立穩固的應用架構，常使用的組件服務包含 data-binding、Lifecycles、LiveData、ViewModel、WorkManager，其中也包含根據 Kotlin 基礎優化的 Android KTX 套件。

4. What is Coroutines? Why do we use it? Try to explain in Mandarin

Coroutines 是 Kotlin 針對同步問題處理開發出的一個框架，以 Cooperative 協同運作處理程序，可以將運行在不同執行緒的程序寫在同一個 scope 中，解決複雜的資源切換問題。以往 Preemptive 形式的 Thread 執行緒，由作業系統根據程式的 Priority 安排當下執行資源，而過多的 Thread 會增加整體效能負擔。Coroutine 的執行切換為禮讓式，且協助管理 Thread 數量，資源交替切換不影響作業系統，最終由一個 Thread 來真正執行處裡。因為不隨意佔用資源，也可以保護主 Thread 的運作順暢。在 Coroutine 中以 job 為工作單位，搭配 Suspend 和 Resume 的函式程式，可以讓每個 Coroutine 暫停動作和再次執行，也有 cancel 方法可以取消 Coroutine 排程。

2. Here are the 7 important lifecycle methods of an Activity. Try to explain when they are called during the lifecycle of Activity.

i. onCreate()

這是開啟應用程式的最先流程，在 Activity 第一次被創建時做初始化的動作，分配資源給這個 Activity，函式內的程式只會執行一次，通常所有常規靜態設置和物件都在此作宣告，這些內容是只要 Activity 啟用就必須可以被響應的。

ii. onStart()

此流程作用在程式是用戶可見卻未能直接操作時，函式內的 view group、adapter、UI 物件初始化，將 Activity 內容顯示到螢幕上，以準備讓操作者使用，這些變數必須是程式前臺運行的時候才能夠被響應。

iii. onResume()

當 Activity 獲得 focus 時取得螢幕的控制權，轉換為在運行中的 Activity 狀態，在這個階段使用者才能與程式互動，大部分的物件屬性都在這個流程被決定，Activity 會不斷重新整理 UI 元件的狀態，直到頁面切換或被其他應用程式佔據。

iv. onPause()

當 Activity 失去 focus 時釋出螢幕的控制權，變為 Paused 狀態，凍結原本的 Activity，撤消在 onResume()中所執行的動作，使用者失去與 Activity 的互動，但應用程式在前台上仍部分可見，UI 物件暫停更新。當 Activity 重新獲得 focus 可以再回到 onResume()階段；或是準備離開程式，進入 onStop()階段。

v. onStop()

當跳離程式畫面用戶已不可見，進入 Stopped 狀態，所有活動停滯，UI 物件消滅，撤消在 onStart()中所執行的動作，Activity 的運作完全停止，但仍存在於後臺，暫存資料也會保存，通常會在函式中釋放手機資源給其他應用程式使用。

vi. *onRestart()

當用戶重新回到程式畫面，Activity 停滯後再次啟用，準備進入 onStart()階段。在此之前，因為 Activity 已被創建， onRestart()會尋找丟失的物件重新建立，並將凍結的 Activity 資源和暫存資料再次來拿使用，緊接著進入 onStart 階段。

vii. onDestroy()

使用者終止應用程式時，這是程式結束前的最後一個動作，函式內的程式也只會執行一次，撤銷 onCreate 函式中宣告的物件，Activity 被完全銷毀，釋放出所有暫用資源。