Week 4

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

   Polymorphism 翻譯為多型，意思大概是：發出一個訊息，但訊息實際運行的方式由收到訊息的一方去作解釋。

   隨便舉一個翻成白話文的例子來說，有一個假設有一個父類別叫煮菜，有基本功能開火→關火→盛盤，但依據買到不同的菜應該要有不同的作法。因此我可以 override 子類別為不同的菜色，並使用不同的作法，在基本功能上加入切菜、洗菜、油炸等任意功能，因此呈現出來的結果會不一樣。同樣都是煮菜，但依據買到不同的菜去使用煮菜功能，呈現出來的東西就不一樣。這樣的表現方式就是 Polymorphism。

   Polymorphism 的好處應該是方便 code 的擴展和編寫，同時可以讓 code 更簡潔，看起來其實就是繼承概念的延伸。

2. Here are the 6 important lifecycle methods of an Activity:
   i. onCreate()
   ii. onStart()
   iii. onResume()
   iv. onPause()
   v. onStop()
   vi. onDestroy()
   Try to explain when they are called during the lifecycle of Activity.

   onCreat()：

   第一次創建活動時觸發此功能，用以開始執行應用程式，通常在整個應用程式生命週期中指觸發一次。接收到此功能觸發時，相關參數都會收到此消息，並將應用程式維持 Created 狀態。

onStart()：

Created 狀態後，通常會直接進入 Started 狀態，系統會用 onStart() 讓應用程式進到前台讓使用者看到互動模式，此功能通常是應用程式初始化維護 UI 的 code 位置。接收到此功能觸發時，相關參數都會收到此消息。

onResume()：

通常在 Started 狀態完成後，也會直接進入 Resumed 狀態，並保持在這個狀態。此狀態是使用中與應用程式互動中的狀態，當此功能觸發時，相關參數都會收到此消息，並且在此生命週期中可以啟用運行所需要的功能，例如啟動相機、麥克風。當 Resumed 狀態受到其他事件如電話 call in 而中斷時，就會進入 Paused 狀態，從 Paused 恢復到 Resumed 時，可以初始化 onResume() 所觸發的功能參數。

初始化功能參數的位置，取決於是否想要這些功能在後台持續運行，若初始化位置放在 onStart() 後，功能在 Paused 時就會持續在後台運行，但同時損耗資源。但如果初始化是放在 onResume() 後，則功能會暫停運行，停止耗損資源，並於恢復 Resumed 時重新啟動。

onPause()：

當應用程式進到 Paused 狀態時，onPause() 將暫停或適度繼續應用程式於後台運行，也可以運用此功能釋放系統資源、停止任何不需要的功能如 GPS。接收到此功能觸發時，相關參數都會收到此消息。但在多視窗模式下，仍然可以看見暫停的應用程式，如果想要使其完全不可見，則應該使用 onStop()。

onPause() 的執行快速簡短，因此不適合用於執行保存的功能，也應該使用 onStop() 作為儲存。

onStop()：

當進入 Stoped 狀態時，使用者將完全不可見到應用程式。接收到此功能觸發時，相關參數都會收到此消息，生命週期中的相關不需要再運行的元件都可以停止，並不被使用者看見，同時釋放系統資源。保存動作也可以在此功能中執行。

onDestroy()：

此功能會使應用程式進入 Destroyed 狀態，並銷毀不需要的數據。通常用在兩種狀況下：1. 活動正在關閉或完成，此步驟為應用程式生命週期的最後一步。2. 設定更改，系統破壞了活動，onDestroy() 會釋放 onStop() 未釋放的資源，並銷毀資料。兩者再次啟動應用程式時則會回到 onCreat() 步驟。

3. What is the Android Jetpack?

Android Jetpack 是一系列的開發者輔助工具和資料庫，類似於我們常用音樂、影像等後製軟體中的 Library 庫，Jetpack 可以完整接受 Kotlin 語言，更新頻率較 Android 快，且可以向後兼容。

就我的理解是：由於 Android 相關開發工具和函式資料庫等太過於雜亂，許多常用的工具散落於第三方或為測試版，因此 Google 整合出一個官方認證版的集合庫，稱之為 Android Jetpack，支援基礎 SDK 以外的部分，其中包含 Foundation, Architecture 架構, 常用 UI 介面, Behavior 等四大面向，並會積極維護和更新，目的是幫開發者更順利穩定的創造。

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

Coroutine 為 Cooperation + Routine 兩字組合而成，表示合作例行作業，也翻為協同式多工、協作程序。相較於傳統模式 Thread，Coroutine 可以讓需要花時間的程序進行同時，同時執行別的工作。

舉例而言，假設執行功能：

fun A()

fun B()

fun C()

main Tread 必須要依照執行順序 A → B → C，但若 fun B() 需要等待 fun A() 完成，main Tread 會為了接續準備運行 fun B() 而進行單純的等待，此時畫面就會靜止。但

在 Coroutine 方式運行下，等待 fun A() 完成的同時，可以先停止 fun B()，並讓 main Tread 先去做別的事，等到 fun A() 運算完成，再繼續 fun B()，這樣畫面可以不需要等待運行過程，仍可以寫入其他使用者顯示程序，好處應該不言而喻。