Week 4

1. Polymorphism ?

ANS: 多型，我們可以預先定義一個常態操作介面(類別定義的操作方法)，使其處理一特定形式的資料，此時，當我們有其他類別都需要透過此常態操作介面處理資時，我們就可以使其繼承此常態操作介面，接著便可以透過此繼承的子類別同時改寫(Override)繼承自父類別的函式或變數，產生不同的資料處理結果，也就是說，同一常態操作介面可以依處理物件的型式執行其相對應的運算。

2. important lifecycle methods ?

ANS:

onCreate() ：activity被創建時調用，此時會進行初始化

onStart() ：當activity在我的畫面上處於前台運作並可見時調用

onResume() ：當我的activity正在被關注(用戶可與activity互動)使用時調用

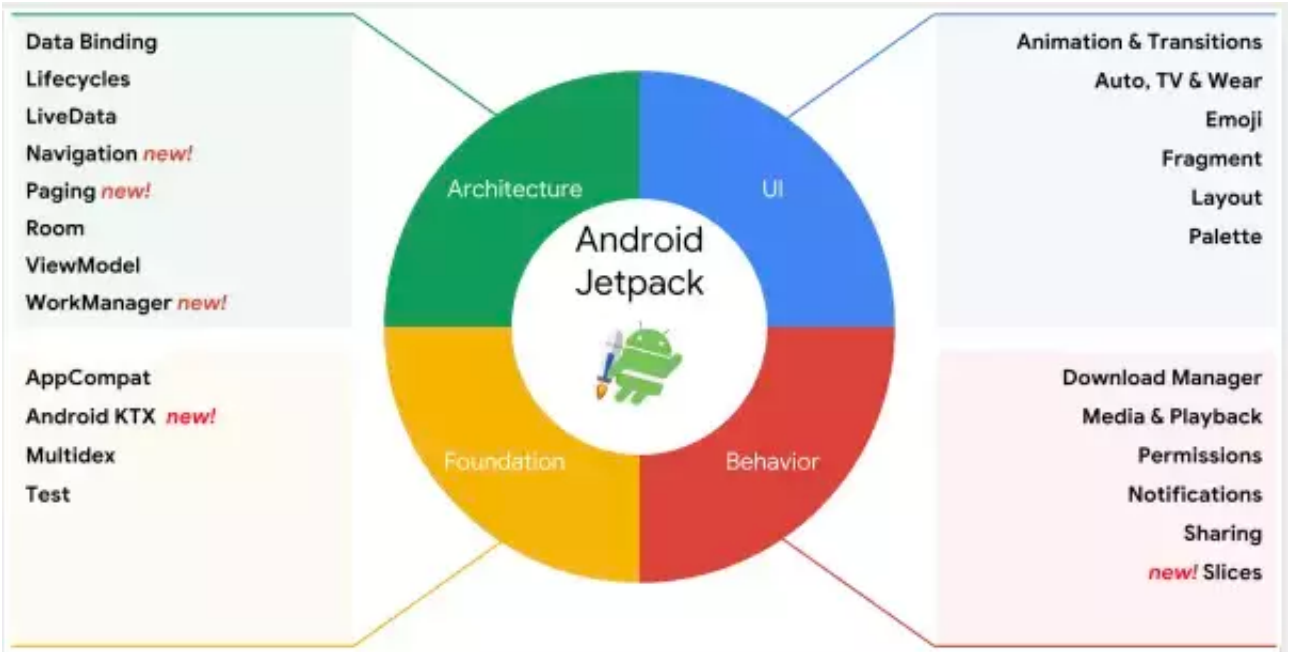
onPause() ：當我的activity目前正失去關注(用戶無法與activity互動，不包含結束程式狀態)時調用

onStop() ：當activity因離開不在我的畫面上運作時調用，此時會永久保留activity數據

onDestroy() ：當activity被破壞，也就是當程式關閉不在記憶體上運行時調用

3. Android Jetpack ?

ANS: 一個強大且可供測試的資料庫，包含data tooling、guidance 以及相關文件，可以讓開發者在程式編寫上更方便；Gradle 內的repository：google()就有來自Jetpack提供的服務。Jetpack兼容kotlin，其組件包括：基礎（Foundation）、架構（Architecture）、行為（Behavior）、界面（UI）：



4. Coroutines ?

ANS: 協程，當我們程式在運作時，呼叫的函式會從頭到尾執行完才換下一個，但是當我們遇到諸如網路速度或是系統運算上不夠快速時，可能會因此造成UI畫面hang住或hiccup，相當不方便，此時，我們便可以引入協程的方式，Coroutines會將要執行的程式拆進不同的thread執行，其允許函式執行到一半就中斷（yield），中斷時內部狀態會被保留下來，呼叫端可以隨時在之後恢復（resume）這個 coroutine，也就是說當程式需要時間處理大量資料時，coroutine可以先中斷這個好時的程序，先去其他thread執行別的程式，等到此好時的程序執行完再回來接續執行其接下來的部分，以降低程式計算時需等待的時間。