Q1. Polymorphism(多形)

Polymorphism means that the sender of a stimulus does not need to know the receiving instance’s class. The receiving instance can belogin to an arbitrary class.

If an instance sends a stimulus to another instance, but does not have to be aware of which class the receiving instance belongs to, we say that we have polymorphism.

白話文:

一個訊息（message or event or stimulus）的意義是由接收者（接收到這個訊息的物件）來解釋，而不是由訊息發出者（sender）來解釋。所以，在runtime時只要接受者換成不同的物件或是instance，系統的行為就會改變。具有這樣的特性就稱之為polymorphism。

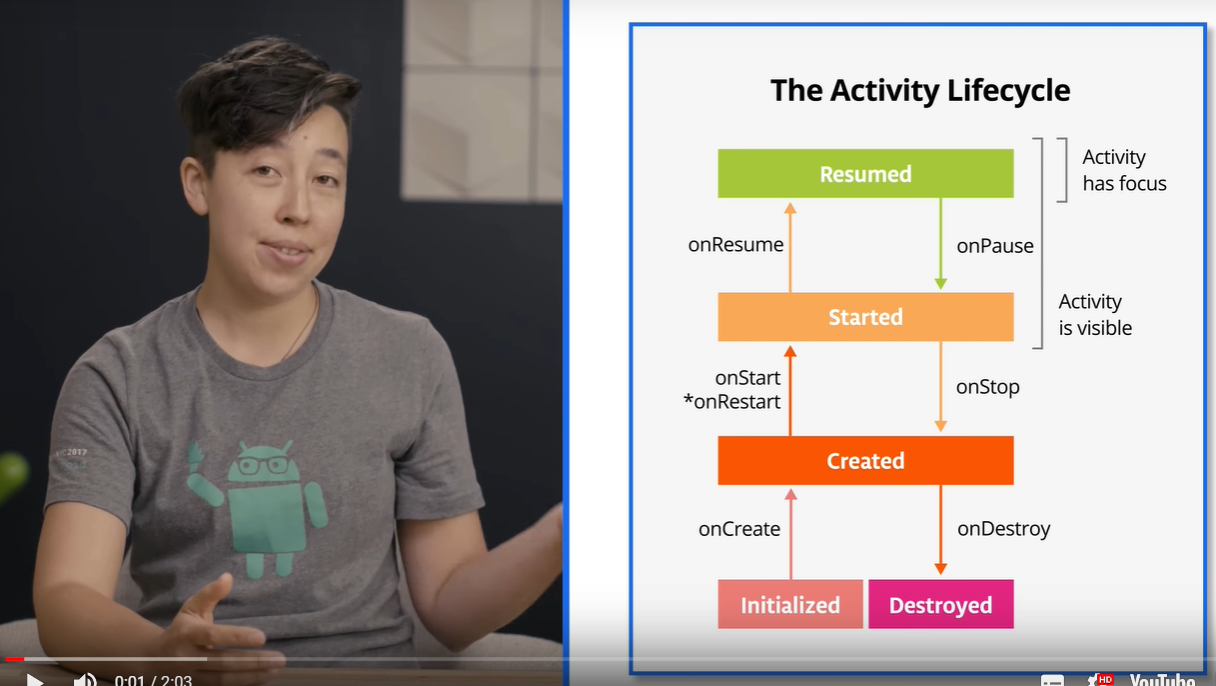
舉個程式的例子：

List<String> list = new LinkedList<String>();

list.add(“This is a book”);

從上面這一行程式來看，我們可以說寫程式的人送了一個add這個訊息給list這個instance。從靜態程式碼來看，鄉民們一定知道list.add(“This is a book”)的行為，但是如果list這個instance不是像上面程式一樣寫死new LinkedList<String>();，而是runtime時傳入的任意符合List介面的某個instance，那麼程式的行為就無法只從靜態的程式碼得知了，而且add(“This is a book”);訊息的解釋也只能看runtime時list instance指到哪一個instance而定。

以上取自<http://teddy-chen-tw.blogspot.com/2012/01/3polymorphism.html>

Q2. 

1. onCreate() : 在程式剛開始執行時，這一個區塊的程式只會執行一次，接下來會接續執行onStart()裡的程式。
2. onStart() : 接續onCreate()的程式行為，繼續執行，在正常的activity循環中也只會執行一次(如是跳出程式或是反轉螢幕後面討論)，並且接續執行onResumed()裡的程式。
3. onResumed() : 個人理解中，主要程式運行的區塊，在程式完全沒有跳出及正常執行下，會停留在這個階段。
4. onPause() : 在程式不再前景，但是畫面還留在螢幕上時，會進入pause狀態，如果馬上點出來就會執行對應的onResume()。
5. onStop() : 程式如果不在螢幕中，則就會執行此段程序，如長時間沒有再喚醒程式或是直接在系統管理面結束程式，會接續onDestroyed()程序讓占用的空間釋出，如果喚醒則會啟用onRestart()方法，而不是onStart()。
6. onDestroyed() : 週期的最後一站，調用此方法之後程式完全結束運行。

Q3. Android Jetpack



為了使開發更加順利，Android Jetpack誕生了。

組成為，架構、界面、基礎和行為

Android Jetpack目前有 4個穩定版本的架構組件他們分別是：Lifecycles、LifeData、Room 以及 ViewModel。就像下圖中展示的，界面組件負責顯示 UI 界面，ViewModel 負責處理 UI 數據，你可以使用 LiveData 來更新數據進而更新 UI 界面。 Respository 是一個唯一的數據層結構，你可以使用 Room 來進行數據持久化。

而其他的功能都可以在下面的資料來源看到<https://juejin.im/post/5baa047c6fb9a05d330abf06>

Q4.

1. Activity: 程式的核心，為程式的生命週期，程式的行為都可以在這裡被呼叫或是規範。
2. Service : 當有些動作不需要或是運行在Activity上太過浪費資源，就會到背景執行，service就可以滿足這個需求。
3. Content Provider : 是Android SDK提供的一個抽象類別，可用以保存及管理資料，更是應用程式之間共享資料的唯一方法。Content Provider就如同應用程式資料之間的橋樑，讓應用程式可以透過這座橋樑共享資料，不過橋樑的通行權卻仍在我們手中控制，使應用程式之間有共享資料的彈性，但也仍然為資料提供了適當的保護。
4. Broadcast Receiver : Android系統有一種特別的「廣播事件」，它可以在系統或其它應用程式發生一些事件的時候，通知需要的應用程式執行一些指定的工作。例如裝置在接到來電的時候，系統會發出一個來電的廣播事件，如果應用程式需要在裝置來電的時候執行一些工作，可以設計一個接收來電廣播事件的「廣播接收元件」。

在一些特別的情況下，應用程式需要發送自己定義廣播事件，裝置中的其它應用程式可以接收與處理這個廣播事件。系統或應用程式自己定義的廣播事件，都是使用Action名稱來識別它是哪一種廣播事件，所以要為自己定義的廣播事件取一個Action名稱，再使用這個名稱發送廣播事件。呼叫Activity元件提供的「sendBroadcast」方法可以發送廣播事件，它需要一個設定好Action名稱的Intent物件，你也可以在Intent物件中設定一些資料，這些資料可以傳送給處理的廣播接收元件使用。

1. Notification : 有時候當我們在使用手機時，常常在上面的狀態列中，突然跳出一個通知，讓使用者可以將狀態列拉出來點選，這樣的功能即稱作通知訊息(Notifications)，它允許在使用不同的 APP 時觸發，也就是說，就算發出該通知的 APP 目前並不是正在使用，使用者還是可以透過通知訊息，取得該 APP 想要告知我的資訊，再進一步透過點選該通知，執行該 APP 希望我們做的動作。
2. Runtime Permission : 當你在Android6.0以上的版本，加強了隱私權的重視，因此在操作APP上，就必須讓使用者自行勾選是否開放這些權限。常見的危險權限有: 
3. ANR（Application Not Responding）:

KeyDispatchTimeout –按鍵或觸摸事件在特定時間內無響應；

BroadcastTimeout –BroadcastReceiver在特定時間內無法處理完成；

ServiceTimeout –Service在特定的時間內無法處理完成；

ANR的發生原因：

1. 應用自身引起，例如：

主線程阻塞、IOWait等；

1. 其他進程間接引起，例如：

當前應用進程進行進程間通信請求其他進程，其他進程的操作長時間沒有反饋；

其他進程的CPU佔用率高，使得當前應用進程無法搶占到CPU時間片；

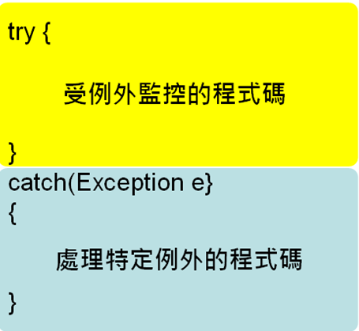
1. OOM(Out Of Memory ) :

Android開發上的限制,在Android中記憶體有分兩種Dalvik Heap 和 Native Heap  
而OOM造成的錯誤就是Native Heap超出了"限制"  
所謂的限制的話早期的設備是16MB  
後來的設備增加到24MB或32MB(依各家硬體設備)

硬體規格較新的設備上這個Crash是偶發性   
硬體規格較差的設備上Crash是100%會發生

1. Exception :

例外(Exception),指的是程式發生不正常的錯誤,而導致無法繼續執行的情形。例外處理(Exception Handling)顧名思義,就是當例外發生時的處理機制。(通常使用try….catch)的方式來寫。



在過去，程式開發人員常花時間在撰寫「檢查錯誤」的相關程式碼，最主要原因是，當程式碼執行過程中，若發生「錯誤（Error）」時，將會中斷整個軟體的執行，造成程式無法繼續往下執行。

現代語言以「**預先認定**」某些程式片段（或執行某特定方法）可能出現 Exception－例外，若事先因為認定其會發生例外，就要求在設計過程中一定要將處理例外情形的程式碼預先撰寫設計好，當執行過程中真的產生例外時，會按照事先設計的程式碼來處理例外，程式也能正常的繼續執行。

1. Material Design : 針對Android應用程式的畫面與操作，提供創新與統一的設計方式。應用程式開發人員可以設計具有3D空間感覺，與使用元件與畫面的轉換動畫效果，讓使用者在操作應用程式的時候更加順暢，畫面也更加美觀。

來源: <http://www.codedata.com.tw/mobile/atk0601/>

Q5.

Coroutine : 就是可以暫時中斷，之後再繼續執行的程序

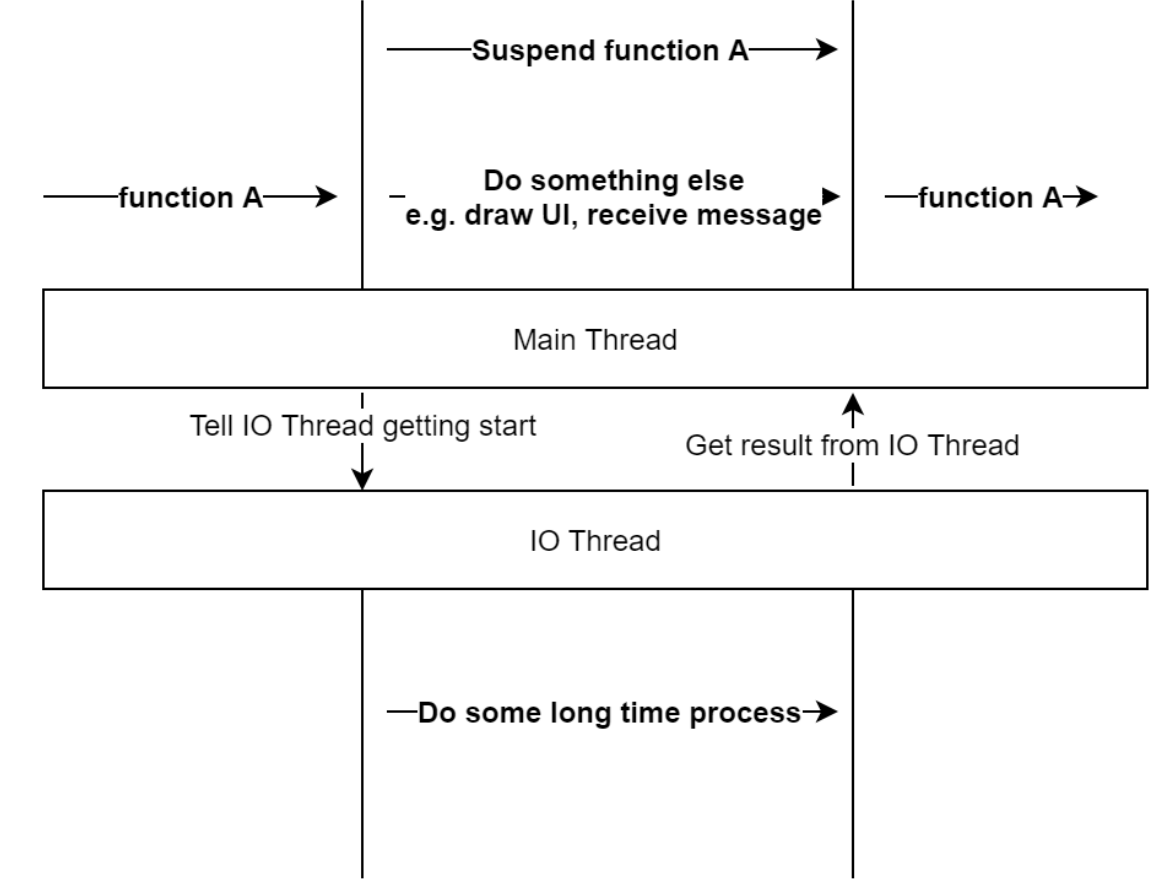
Coroutine ，這個單字會被標成錯字，理由是他其實是兩個單字合併而成的，分別是 cooperation + routine， cooperation 意指合作，routine 意指例行作業，照這裡直接翻譯就會是合作式例行作業。

那我們看到的翻譯多半會是協程、協作程序…這樣講沒啥前後感，誰協助程序？協助啥程序？

這裡 routine 指得是程序中被呼叫的 function、method ，也就是說，我們將 function 、method 協同其他更多的 function、method 共同作業這件事情稱為 Coroutine。

協同作業聽起來還是很抽象，具體協同甚麼呢？

這便是 Coroutine 最典型的特色，允許 method 被暫停( suspended)執行之後再回復(resumed)執行，而暫停執行的 method 狀態允許被保留，復原後再以暫停時的狀態繼續執行。



換句話說，就是我在 main thread 執行到 function A 需要等 IO thread 耗時處理的結果，那我先暫停 function A， 協調讓出 main thread 讓 main thread 去執行其他的事情，等到 IO thread 的耗時處理結束後得到結果後再回復 function A 繼續執行，已得到我要的結果，這便是 Coroutine 的概念。