第一類

Android 基礎知識-UI 設計及語法應用

101. 整存整付計算機
102. 點餐系統
103. 選擇手機型號
104. 手機與平板 UI 佈局
105. 計算 BMI 值
106. 設計 CardView 樣式
107. 驗證 Activity
108. 動態密碼顯示
109. 電影租片清單
110. 自動收合的廣告版位

107. 驗證 Activity

題解說明

A.解題要項

- 熟悉新增 Activity 的關鍵步驟以及在清單檔案內的註冊方式
- 熟悉 Activity 的切換方法與參數傳遞的技巧
- 熟悉 Activity 回傳結果（含資料）的關鍵步驟與接收回傳結果的方法
- 利用 Log.i(...)體驗 Activity 生命週期的執行過程（方法）

B.重點描述

1. 當新增一個新 Activity 類別，通常會在 onCreate(...)內利用 setContentView(...)來指定特定的佈局檔案(R.layout.xxx)以作為該 Activity 的顯示內容；此外，也必須在清單檔案內(AndroidManifest.xml)宣告 Activity (<Activity...>)，否則在執行時會因為未宣告該 Activity 而導致錯誤、無法正常執行。
2. 當使用 Intent 進行 Activity 的切換，可以利用 Intent + Bundle 進行參數的傳遞；此外，若需要被呼叫的 Activity 回傳值行結果，則必須以 startActivityForResult(...) 來取代 startActivity(...)，並且覆寫 onActivityResult(...)來接收回傳的結果。
3. 被呼叫的 Activity 可以利用 setResult(...)來設定回傳資訊與執行結果；並且可利用 finish()來結束 Activity 本身的執行以返回前一個 Activity。以下是兩種常見的設定回傳結果的方法：

   (1) 設定回傳結果代碼、不帶資料：

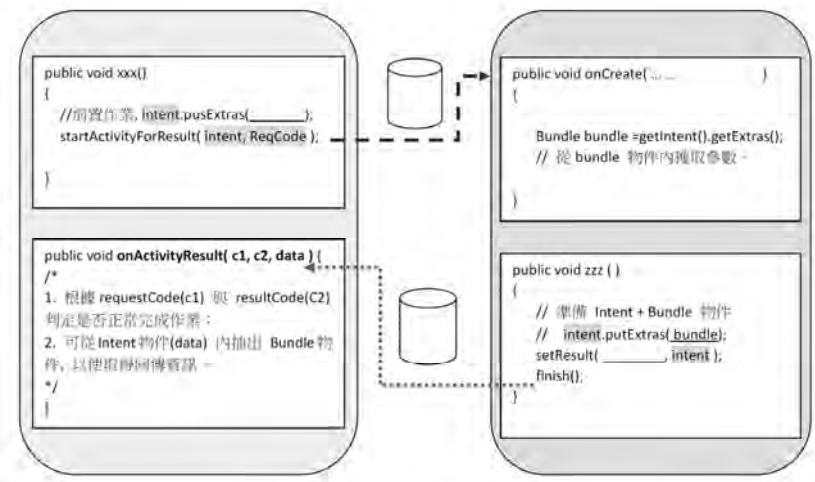
   ```
   setResult(resultCode);
   ```

   (2) 設定回傳結果代碼，並帶回傳資料(Intent + Bundle)：

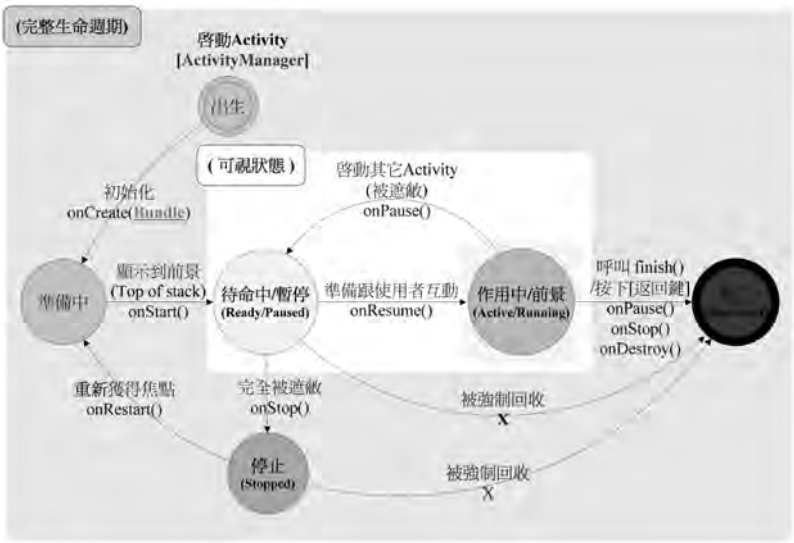
   ```
   setResult(resultCode, intent);
   ```

4. 熟悉 Activity 生命週期的各個過程（方法）是設計一個好的 Android App 必備的知識；在解題中，必須嘗試覆寫 Activity 生命週期中的幾個重要方法(onCreate、onStart、onResume、onPause、onStop、onRestart、onDestroy)，並利用 Log.i(...)顯示對應資訊，以便觀察 Activity 生命週期的執行過程。

為方便讀者理解與記憶，請參考以下圖例中的關鍵指令，以便熟悉 Activity 的切換過程：

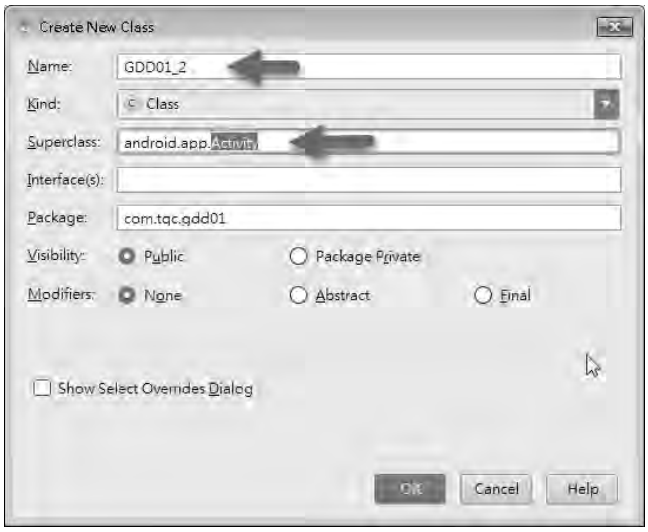


至於 Activity 生命週期的切換過程（狀態的改變與被呼叫的方法），請參考下圖：



【107. 驗證 Activity】

5. 根據題意，此題已經設計好佈局檔(mylayout.xml)，故只需新增一個新類別(GDD01_2)，並**讓該類別繼承 Activity**：以滑鼠右鍵點擊專案區的 Java 套件(com.tqc.gdd01)，選擇選單「New > Java Class」，接著輸入類別名稱(Name)為「GDD01_2」，並在 Superclass 輸入「android.app.Activity」；最後點擊「OK」按鈕即可完成。



然後覆寫 Activity 的 onCreate(...)方法，並且加入以下兩行關鍵指令：

```
super.onCreate(savedInstanceState);
setContentView( R.layout.mylayout);
```

其餘的事件處理與回傳結果的相關指令，請參考【程式實作】內容。

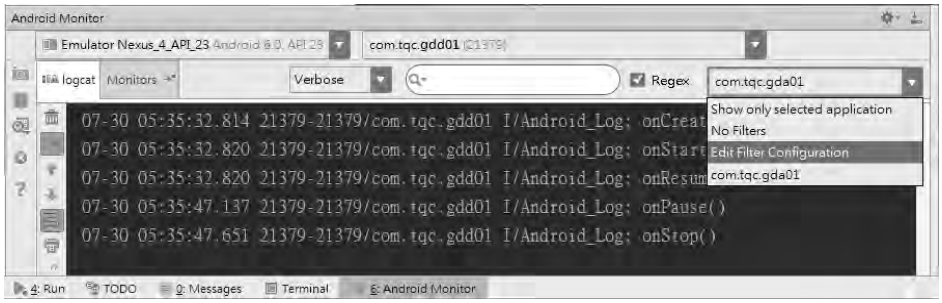
6. 再開啓 AndroidManifest.xml 清單檔案，在<Activity...></Activity>下方加入新 Activity (GDD01_2)的宣告：

```
<activity android:name=".GDD01_2"> </activity>
```

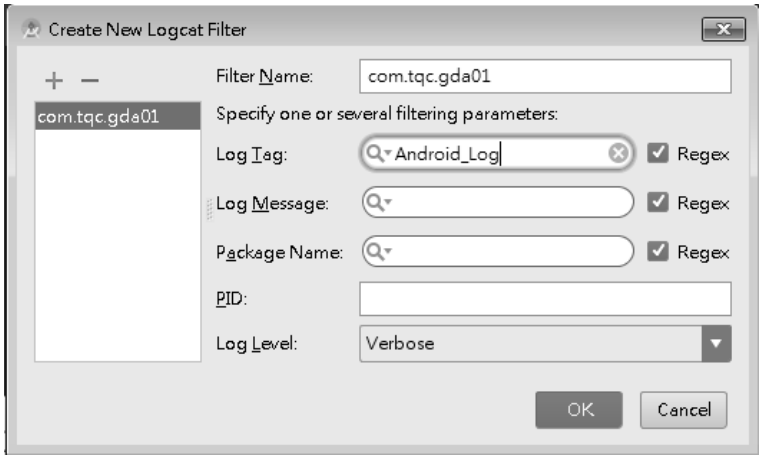
7. 至於主要 Activity(GDD01)如何利用 Intent 來啓動另一個 Activity(GDD01_2)，GDD01_2 如何回傳結果給 GDD01，其過程就如前面步驟 2~4 所解說；詳細指令請參考【程式實作】內容的說明。

8. 開啓 Logcat，以便觀察相關訊息、確認程式正常無誤：

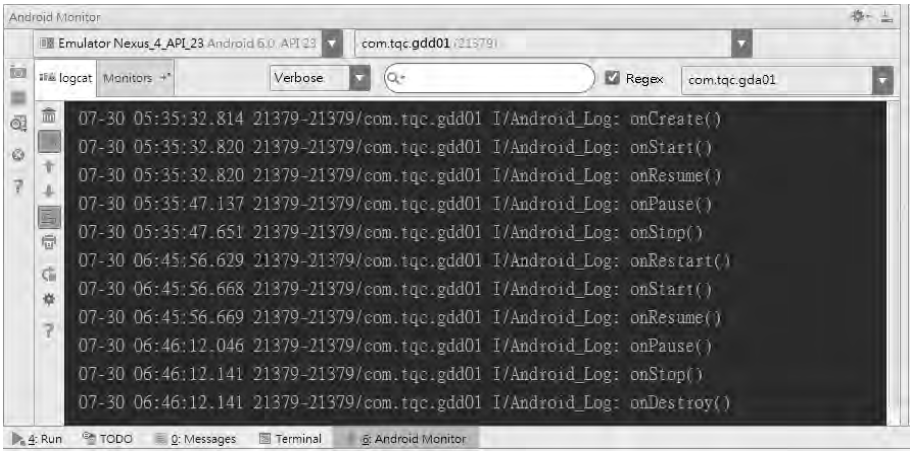
(1) 直接按下【Alt+6】→可開啓【Android Monitor】頁籤，接著點選右上方的下拉式選單，選擇「Edit Filter Configuration」：



(2) 然後新增過濾條件(Logcat Filter)：Filter Name 輸入「com.tqc.gda01」，Log Tag 則輸入「Android_Log」，最後點選「OK」按鈕即可完成設定。



(3) 執行所有過程後，從 Logcat 可觀察到的相關訊息如下圖所示：



第二類

資料儲存與解析

201. 公尺與英呎轉換

202. 匯率換算

203. 猜數字遊戲

204. 解析 XML 資料格式

205. 我是誰?

206. 資料庫讀取

207. 姓名清單

208. 簡單記事

209. 檔案下載管理員

210. 臺北捷運列車到站站名

202. 匯率換算

題解說明

A.解題要項

- 能利用 LinearLayout 與基本元件完成畫面設計
- 能利用 InputType 限制 EditText 的資料輸入類型
- 熟悉按鈕的事件註冊與處理機制
- 能利用 SharedPreferences 讀取與記錄資料

B.重點描述

此題的解題關鍵利用 SharedPreferences 進行簡易資料（匯率）的儲存與讀取；使用上與 Bundle 物件的方式很類似，都是利用 key 與 value 的對應關係，以下是使用 SharedPreferences 的關鍵步驟：

1. 先取得 SharedPreferences 控制物件：

```
SharedPreferences sp = Activity.getSharedPreferences("abc");
```

2. 資料讀取：通常會在適當的時機，針對不同資料型態、利用不同的方法來讀取資料。

```
String value = sp.getString("key", "default");
int level = sp.getInt("level", 0);
```

3. 資料儲存：通常會在適當的時機，針對不同資料型態、利用不同的方法來儲存資料。

```
SharedPreferences.Editor sped = sp.getEdit(); //  取得編輯物件
sped.putString("key", "newValue");
sped.putInteger("level", 1);
//  儲存資料最後一定要呼叫 commit()，才算完成寫入的動作
sped.commit();
```

Note 利用 SharedPreferences 進行簡易資料處理是常用的技巧之一，Android 是以 XML 的方式來記錄與讀取基本資料；以上面的例子而言，就會在該 App 的私有路徑內找到（生成）XML 檔案：

/data/data/<套件名稱>/shared_prefs/**abc.xml**

開發人員可以利用 DDMS 內的檔案總管進行該檔案的檢視與確認；通常模擬器可以直接查看該路徑，但是，一般手機/平板則必須有 root 權限方可查看。

4. 根據題意，畫面中所需的字串常數已經被定義在/res/values/strings.xml 資源檔案內；建議先開啓該檔案進行觀察名稱與對應字串內容：exchangerate、ntd、calc_btn、usd_result。

5. 接著，開啓/res/layout/main.xml 進行畫面設計；畫面中所顯示的文字均須參考事前定義的字串常數；此外，兩個 EditText 必須設定 InputType 為 numberDecimal，以限制只能輸入數值資料（含小數）；按鈕下方須有一個 TextView 用以顯示轉換後的匯率結果。

6. 關鍵元件 id 須設定為有意義的名稱，以下是建議修改的元件屬性值：

原名稱(id)	新名稱(id)	其他
editText(「匯率」下方)	etRate	限制輸入浮點數
editText2(「台幣」下方)	etNTD	限制輸入浮點數
button(「計算 台幣兌換美金」按鈕)	btnCompute	寬度設為 match_parent
textView3(按鈕下方)	tvResult	

7. 接著開啓主程式 GDD02.java，觀察後發現程式內已經宣告了三個成員變數(calcbutton、fieldExchangeRate、fieldNTD)；接著，請依序完成以下步驟：

   (1) 綁定畫面中的關鍵元件：
   calcbutton(btnCompute)、fieldExchangeRate(etRate)、fieldNTD(etNTD)。

(2) 利用匿名內部類別實作 OnClickListener 介面以進行按鈕的事件註冊處理；然後，覆寫抽象方法 onClick(View v)。

(3) 在 onClick(...)方法內部讀取使用者輸入的匯率與台幣資料，根據按鈕的不同，進行不同的公式計算，並在 TextView 顯示具兩位小數的運算結果。此題需要利用 Double.parseDouble("12.3")方法進行字串內的浮點數資料的解析。

8. 最後，處理匯率資料的儲存與顯示。

9. 匯率資料的儲存：觀察 GDD02 內的 onPause(...)方法，發覺該方法內部已經利用 SharedPreferences 完成匯率資料的儲存動作，故無需再加入任何程式碼；因為，當程式被關閉之前，一定會執行 onPause()方法。

10. 顯示前一次儲存的匯率資料：只需在程式初始化方法 onCreate(...)內部利用指令讀取前一次 SharedPreferences 所儲存的匯率資料，並顯示到對應的輸入元件內(EditText: etRate)即可。

程式實作

res/layout/main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="fill_parent"
5     android:layout_height="fill_parent"
6     android:orientation="vertical">
7
8        <TextView
9              android:layout_width="wrap_content"
10             android:layout_height="wrap_content"
11             android:textAppearance="?android:attr/textAppearanceMedium"
12             android:text="@string/exchangerate"
13             android:id="@+id/textView" />
14
15       <EditText
16             android:layout_width="match_parent"
```

```
17          android:layout_height="wrap_content"
18          android:inputType="numberDecimal"
19          android:ems="10"
20          android:id="@+id/etRate" />
21
22      <TextView
23          android:layout_width="wrap_content"
24          android:layout_height="wrap_content"
25          android:textAppearance="?android:attr/textAppearanceMedium"
26          android:text="@string/ntd"
27          android:id="@+id/textView2" />
28
29      <EditText
30          android:layout_width="match_parent"
31          android:layout_height="wrap_content"
32          android:inputType="numberDecimal"
33          android:ems="10"
34          android:id="@+id/etNTD" />
35
36      <Button
37          android:layout_width="match_parent"
38          android:layout_height="wrap_content"
39          android:text="@string/calc_btn"
40          android:id="@+id/btnCompute" />
41
42      <TextView
43          android:layout_width="wrap_content"
44          android:layout_height="wrap_content"
45          android:textAppearance="?android:attr/textAppearanceMedium"
46          android:text="@string/usd_result"
47          android:id="@+id/tvResult" />
48  </LinearLayout>
```

第三類

基礎服務應用

301. MP3 播放器

302. 畫布程式

303. 手機網路流量統計

304. 程式背景音樂

305. 接收 SMS

306. 經緯度查/反查地址

307. 判斷發送簡訊狀態

308. 接收簡訊

309. Android Wear 手錶錶面設計

310. 台北市運動中心網路連線 API 解析

309. Android Wear 手錶錶面設計

題解說明

A.解題要項

- 了解穿戴式 App 與傳統 App 的差異，並了解其基本操作方式
- 了解 Android Wear App 的基本設計與開發步驟
- 了解 buiid.gradle 模組檔案的架構，並能加入必要的程式庫參考
- 了解 WatchViewStub 針對不同錶面外觀（畫面）的參數設定方式
- 能修改 AndroidManifest.xml 以設定新的 Activity 與 Watch Face 服務模組
- 能利用 NotificationManager/Notification 顯示通知訊息
- 能利用 NotificationCompat.Action 與 PendingIntent 產生 Action 物件，並搭配 Notification 物件生成 Action Button 選項按鈕

B.重點描述

1. 設計 Android Wear App 時，開發環境必須先下載 Android Wear System Image，並且需要在專案內引用相關程式庫；以下三個是最基本的程式庫，前面兩個一定要加入、最後一個則視狀況加入：

   (1) Wearable Support Library：內含穿戴式設備的基本 UI 元件。

   (2) Google Play Services for Wearable：提供手機/手錶交換資料的基本框架與元件。

   (3) v4 或 v13 support library：同步通知訊息到手錶(Notification for Wearable App)；若只是利用 App 直接在手錶上顯示通知訊息，可選用 level 20 以上，就不需要引用 v4 或 v13 支援程式庫。

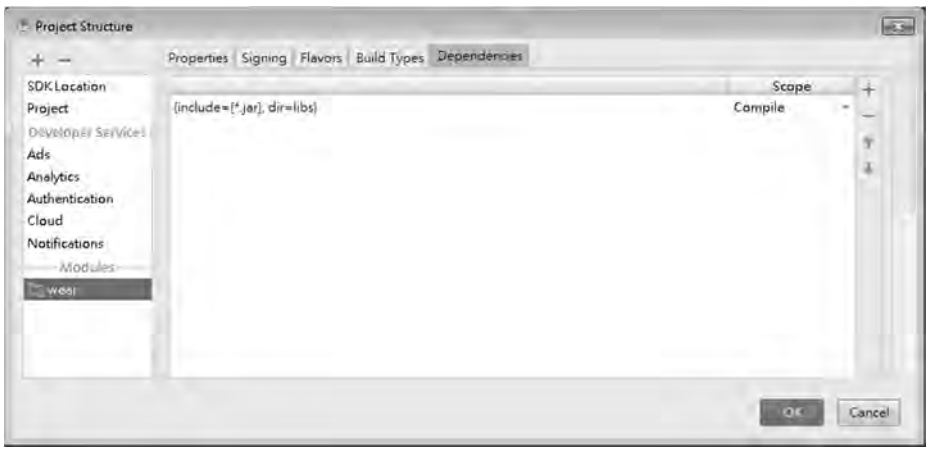
   此題專案開啓後是**無法立刻顯示正常的設計畫面**(main.xml)，是因為尚未引用必要的程式庫，請開啓 build.gradle(Module:wear)，在 dependencies{....} 內部加入以下參考程式庫：

   ```
   compile 'com.google.android.support:wearable:1.4.0'
   ```

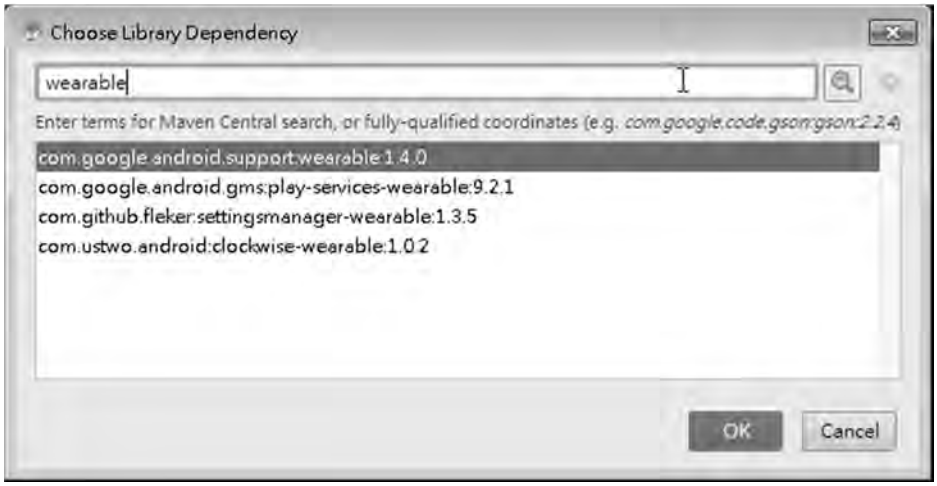
```
compile 'com.google.android.gms:play-services-wearable:9.2.1'
```

完成後，務必進行專案同步，以確保專案內容的即時與正確性。

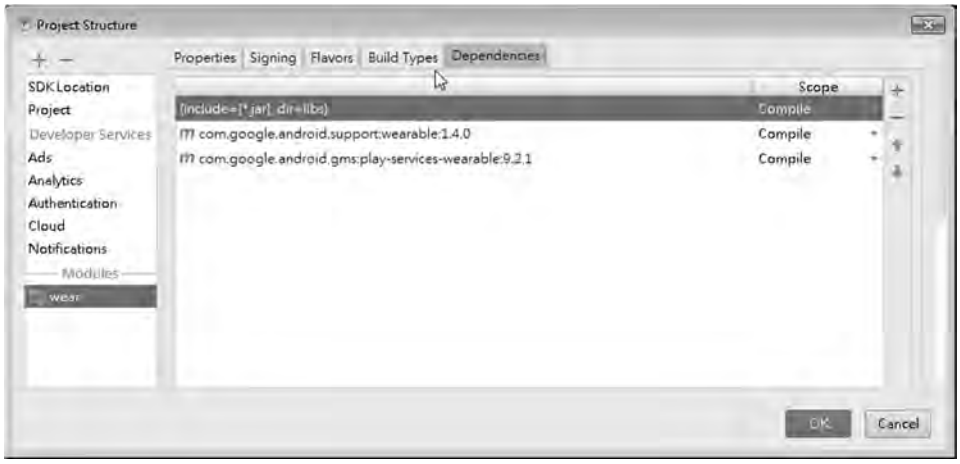
2. 除了手動之外，也可以利用 Studio 來協助完成相關程式庫的引用：請在左邊專案區內，以滑鼠右鍵點擊模組 wear，選擇「Open Module Settings」，接著會看到類似下面的視窗：



此時，請點選上方的【Dependencies】頁籤，先點擊右上角的「+」按鈕，接著選擇「Library dependency」，然後在畫面上方的輸入框內輸入 **wearable** 來篩選穿戴式相關的程式庫；接著選擇上述的兩個程式庫即可（一次只能選擇一個，須重複操作兩次），操作畫面類似下圖：

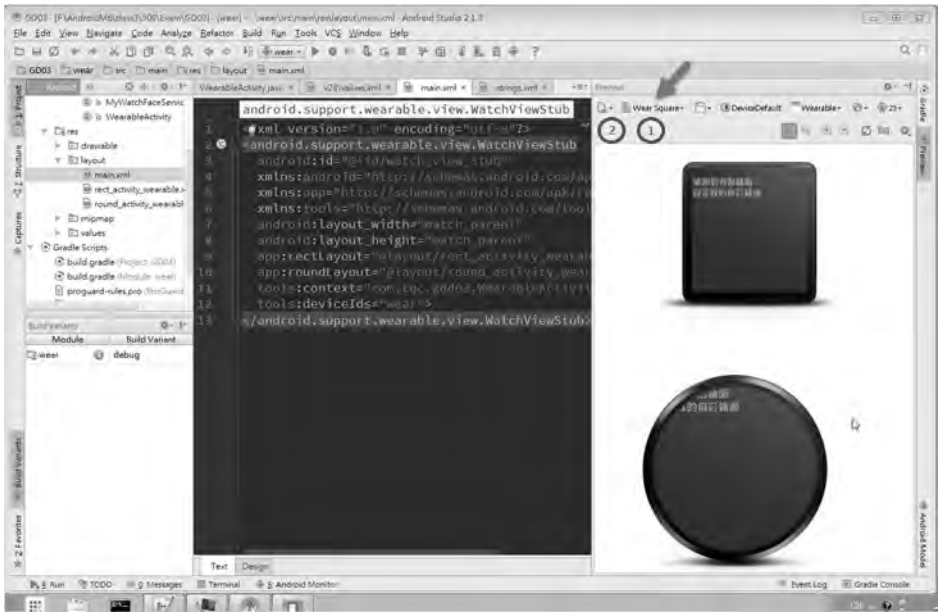


以下是操作完成後的畫面，可以看到兩個必要的程式庫已經出現：



點擊「OK」按鈕後，Studio 開發環境會**自動**進行組態更新，可能需要等待一段時間。必要時，也可以利用【Ctrl+Alt+Y】進行手動專案組態更新。完成後，也可以手動去開啓 wear 模組組態檔(build.gradle)，觀察其中的變化。

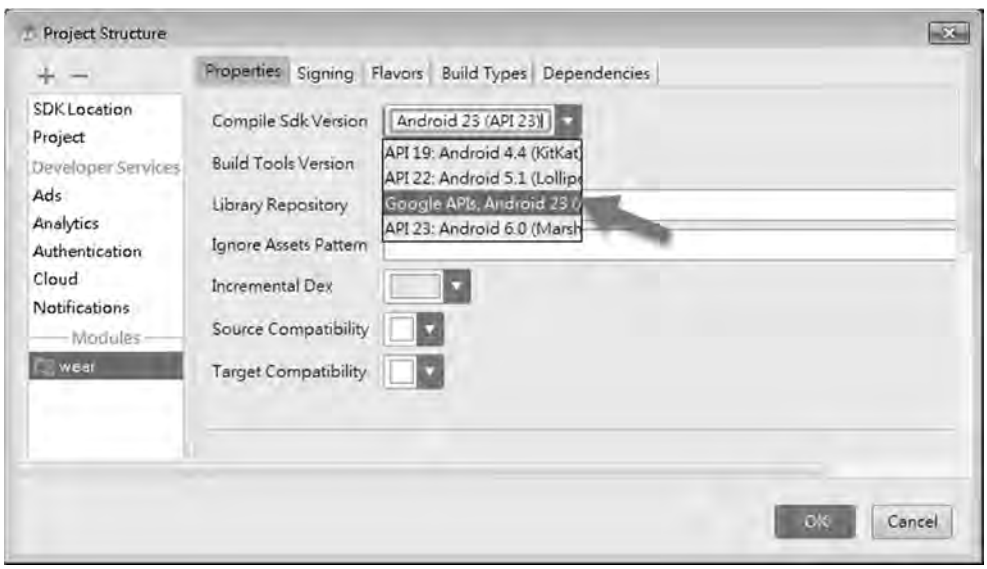
3. 接著請開啓 main.xml，先點選畫面中標示①的位置，選擇「Wear Square」切換到手錶錶面設計的顯示模式（方形）；若要同時看到方形與圓形錶面，請再點選②的位置，選擇「Preview All Screen Sizes」：



【309. Android Wear 手錶錶面設計】

4. 嘗試按下「執行」按鈕，選擇手錶模擬器之後（若沒有手錶模擬器，請先建立一個方形或圓形錶面的手錶模擬器），下方【Messages】頁籤內可能會出現類似以下的錯誤訊息：

主要是因為引用程式的編譯環境是使用了 Android 6.0 版本(Level 23)，但是本專案的編譯版本是 5.1(Level 22)，缺少部分參數而導致錯誤。類似前面第 2 步驟的操作，開啓模組設定畫面，修改編譯版本為具備 Google APIs Android 6.0 (Level 23)以上，該錯誤即會消失。



接著可以再次執行程式，發現還有程式碼中還有錯誤（原有 MyWatchFaceService.java 283~284 行指令)，那是因為程式(提示用)的不完整所導致的。此時，可以先將這兩行程式碼加上註解（不可刪除、稍後會使用到)；然後再次執行，若可以在手錶模擬上出現畫面，代表整個開發環境、參數均已完成設定。

Note 若程式碼發現還有**參考類別** R 相關的錯誤存在(R.xxx.xxx)，請依序進行以下動作：

[Build] → [Clean Project]

[File] → [Synchronize]

待專案資訊同步之後，R 檔案會被重新建立，相關錯誤應該就會消失。

5. 觀察此 Wearable App 專案，共有以下重要類別（檔案）：

(1) Constants：定義共用的常數資源(WATCH_ONLY_ID)。

(2) WearableActivity：手錶 App 的主要 Activity，以 main.xml 作為畫面內容，內含 WatchViewStub 元件；WatchViewStub 的相關重要屬性已經設定好（app: rectLayout, app:roundLayout），執行時會根據錶面樣式（圓形或方形）自動進行佈局檔案(畫面)的切換動作(rect_activity_wearable.xml, round_activity_wearable.xml)。解題時，此 Java 檔案與相關的三個佈局檔案均不需修改。

(3) MainActivity：根據題目要求，當點擊錶面的 Action Button 後，必須啓動此一 Activity；觀察既有程式碼，該 Activity 啓動後會立刻利用 startActivity(...)無條件切換到 WearableActivity。解題時，無須修改任何指令。但是，必須在清單檔案內宣告 MainActivity。

(4) MyWatchFaceService：為手錶錶面(Watch Face)的關鍵服務程式，一般會繼承 CanvasWatchFaceService、覆寫 onCreateEngine()方法，並回傳客製化的 CanvasWatchFaceService.Engine 物件；客製化的 CanvasWatchFaceService.Engine 必須負責處理錶面的內容顯示（覆寫 onDraw 方法）與互動處理（覆寫 onTapCommand 方法）；真正開發 Watch Face 時，也需要知道其他的重要方法，但是此時只需專注在這兩個方法即可。最後，也必須在清單檔案內將其宣告為服務模組<service>，加入三個必要的<meta-data>設定，並且利用<intent-filter>與<action><category>設定 Watch Face 的必要參數。

※詳參 Android 官網資訊：https://developer.android.com/training/wearables/watch-faces/service.html

Note <meta-data>其中一個（wall_paper）的設定必須利用 xml 檔案的方式來儲存相關設定資料，該 xml 檔案內容為：

<?xml version="1.0" encoding="UTF-8"?>

<**wallpaper**
xmlns:android="http://schemas.android.com/apk/res/android" />

6. 開啓 AndroidManifest.xml，依序完成以下宣告：

(1) 加入 WAKE_LOCK 使用權限宣告：

<uses-permission
android:name="android.permission.**WAKE_LOCK**" />

(2) 加入 activity 宣告：

<activity

android:name=".MainActivity"

android:label="@string/app_name">

</activity>

(3) 加入 Watch Face 服務宣告：詳細指令請參考【程式實作】內容。

(4) 在 res 內建立新資料夾「xml」，並在 xml 內部新增一個檔案「watch_face.xml」，並輸入以下內容：

<?xml version="1.0" encoding="UTF-8"?>

<**wallpaper**
xmlns:android="http://schemas.android.com/apk/res/android" />

7. 接著，開啓 MyWatchFaceService.java 程式，依序完成以下步驟：

(1) 找到 onDraw(...)方法，在// 取得日期時間附近的// TO DO 之下加入一行指令，以取得最新系統時間，以便顯示最新日期時間。

calendar = Calendar.getInstance();

(2) 找到 onTapCommand(...)方法，在 switch(tapType)內部（default:上方）加入以下指令處理點擊事件：根據題意，需要送出通知訊息；觀察程式內部既有模組，發覺可以直接呼叫內建方法

buildWearableOnlyNotiifcation(title, message)即可：

```
case TAP_TYPE_TOUCH:
    buildWearableOnlyNotification(
        getString(R.string.notification_title),
        getString(R.string.notification_content)
        );
    break;
```

(3) 完成 buildWearableOnlyNotiifcation(title, message)主要功能：要能顯示通知訊息（以快訊提示顯示於錶面下方），當手指往上拉時可顯示完整訊息內容，再向左滑動螢幕可以顯示**互動按鈕**，點擊該按鈕之後，即可開啓 MainActivity（自訂錶面首頁）。

解題關鍵：(a) 互動按鈕部分可利用 NotificationCompat.Action 與 PendingIntent 來設定按鈕圖示、文字與所要觸發的 Activity；(b) 通知訊息部分則可利用 Notification.Builder 或 NotificationCompat.Builder 生成通知訊息物件（必須搭配 Action 物件以生成 Action Button），再利用 NotificationManager 送出/顯示訊息內容。

```
// 建立按鈕事件開啓手錶應用程式頁面 MainActivity
Intent actionIntent =
        new Intent(MyWatchFaceService.this,
MainActivity.class);
PendingIntent pendingIntent =
PendingIntent.getActivity(MyWatchFaceService.this, 0,
actionIntent , 0);
```