Lab7 Android 的非同步執行

一、本節目的

- 理解執行緒與非同步執行的觀念。
- 理解如何使用 Thread 類別。

二、觀念說明

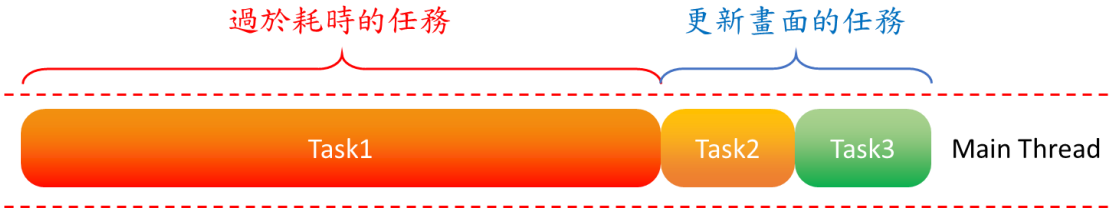
在前面章節中，我們知道應用程式的執行是在 Activity 之上，而 Activity 的運行好壞會直接影響到使用者的操作，假如說程式上設計不良，出現類似迴圈導致 Activity 執行時間過久呢？這時就會發生 ANR(應用程式沒有回應)。

要解決此方法的關鍵就是使用非同步執行的方式來執行程式，以下會做說明。

1　執行緒與非同步執行

在沒有特別設計下，所有的任務(Task)都會在 **Main Thread(或稱為 UI Thread)**上執行，如下圖所示：

Main Thread(或稱為 UI Thread)負責處理畫面更新的作業做更新，如果 Main Thread 其中一個任務(Task)非常耗時間，或是完成時間不可預期，如網路相關的動作、資料庫的動作、檔案操作或複雜的計算，使 Main Thread 無法執行更新畫面相關的操作，就會造成畫面卡住，甚至出現 ANR 情況，如下圖所示：



所以，我們需要把非常耗時間，或是完成時間不可預期的任務(Task)，使用**非同步的方式**來處理，透過非同步的方式，放到 **Background Thread** 來執行，這樣就可以避免 Main Thread 出現任務(Task)卡住的問題，如下圖所示

下面會介紹如何使用 Thread 類別來實現非同步執行，把過於耗時的任務(Task)放到 Background Thread，讓 Main Thread 的任務(Task)執行不會受到影響。

2 非同步執行方法

在 Android 中我們很常使用到非同步執行的方法，例如我們前面所教導的 Toast 就是很典型的非同步執行，他能在執行之後獨立運作，顯示期間 Activity 依然可以繼續的執行下去。

針對我們應用程式中的需求，我們可以產生出新的 Thread 去執行我們要實作的耗時作業，要在程式中實作一個新的 Thread 最簡單的方法可以用以下寫法：

```
new Thread(new Runnable() {

    @Override
    public void run() {
        //do something
    }
}).start();
```

產生一個 Thread 之後，我們需要用 Runnable()介面，Runnable()會提供 run()方法執行我們要跑的程式，因此要將要實作的程式碼寫在 run()之內，然後使用 Thread.start()方法將任務啟動。

由於 Thread 類別會產生出新的 Background Thread 去執行任務，但是 Background Thread 由於不是 Main Thread，無法操作畫面的更新，因此當 Background Thread 中的任務需要操作畫面時，就必須要嘗試與 Main Thread 溝通，透過 Main Thread 來對畫面更新，這時就會需要使用到 Handler 類別。

Handler 是一種跨 Thread 的溝通機制，可以在一個 Thread 內把訊息丟到 Message 類別中，再讓另外一個 Thread 從 Message 中取得訊息。在 Thread 類別中，我們需要額外加入以下程式碼：

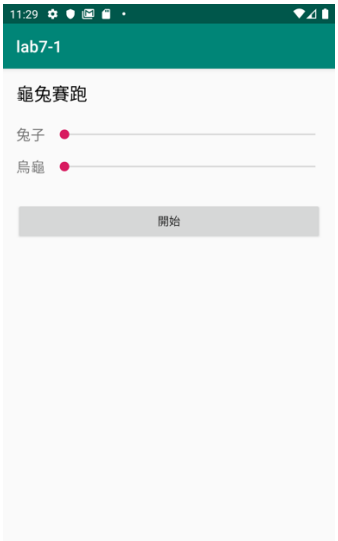
1) 首先，我們需要在建立一個 Handler 類別，在 Handler 中實現一個 handleMessage()方法，利用 handleMessage 這事件來監聽 Thread 是否有送 Message 出來，並在 Main Thread 執行對應的操作。

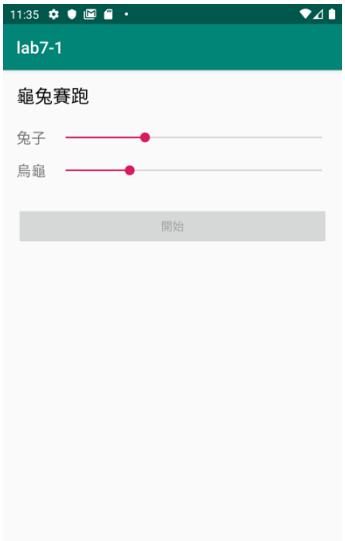
2) Thread 類別中需要建立一個對應的 Message 物件，Message 會用於傳遞至 handleMessage()，Message 的 what 屬性可以加入一組代號，handleMessage()可根據代號來判別要做什麼。

3) 當 Thread 類別要發出 Message 時使用 Handler.sendMessage()方法來觸發 handleMessage()，如此就能把畫面操作透過 Handler 來執行處理。

三、設計重點(龜兔賽跑專案)

- 畫面中會使用兩個進度條(Seekbar)來模擬烏龜與兔子的跑步路線。
- 由於烏龜與的兔子的移動要同時進行，且都為耗時的工作，因此烏龜與的兔子的移動使用 Thread 執行。
- 按下開始後，同時啟動 Thread，讓各自的進度條增加。
- 先抵達終點(進度條達到 100%)，則會用 Toast 顯示出贏家

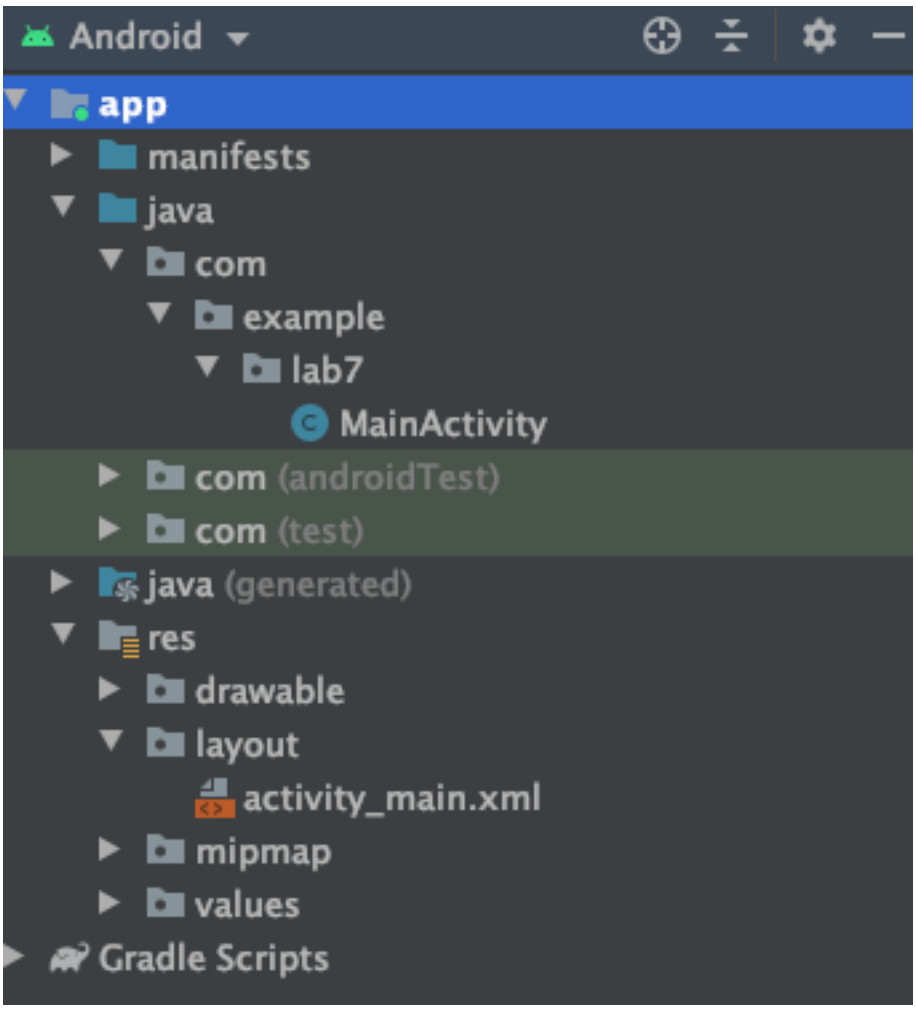




四、設計步驟(龜兔賽跑專案)

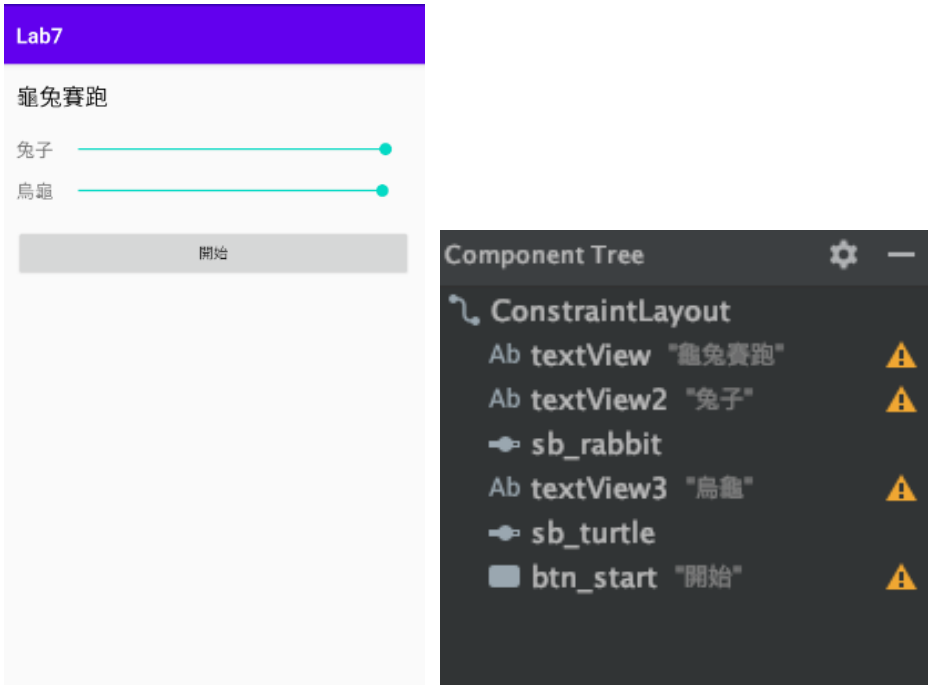
Step1

新增專案，以及對應的 java 檔和 xml 檔。



Step2

繪製 activity_main.xml 檔



對應的 xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="龜兔賽跑"
        android:textSize="22sp"
        android:textColor="@android:color/black"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="兔子"
        android:textSize="18sp"
        app:layout_constraintStart_toStartOf="@+id/textView"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

    <SeekBar
        android:id="@+id/sb_rabbit"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        app:layout_constraintBottom_toBottomOf="@+id/textView2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/textView2"
        app:layout_constraintTop_toTopOf="@+id/textView2" />
```

```
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="烏龜"
        android:textSize="18sp"
        app:layout_constraintStart_toStartOf="@+id/textView"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

    <SeekBar
        android:id="@+id/sb_turtle"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        app:layout_constraintBottom_toBottomOf="@+id/textView3"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/textView3"
        app:layout_constraintTop_toTopOf="@+id/textView3" />
```

```
    <Button
        android:id="@+id/btn_start"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="開始"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView3" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Step3

編寫 MainActivity，按下按鈕後，執行 runRabbit、runTurtle 副程式

```
public class MainActivity extends AppCompatActivity {

    //建立兩個數值，用於計算兔子與烏龜的進度
    private int progressRabbit = 0;
    private int progressTurtle = 0;

    private Button btn_start;
    private SeekBar sb_rabbit, sb_turtle;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btn_start = findViewById(R.id.btn_start);
        sb_rabbit = findViewById(R.id.sb_rabbit);
        sb_turtle = findViewById(R.id.sb_turtle);

        //開始按鈕監聽器
        btn_start.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                btn_start.setEnabled(false); //進行賽跑後使按鈕不可被操作
                progressRabbit = 0; //初始化兔子的賽跑進度
                progressTurtle = 0; //初始化烏龜的賽跑進度
                sb_rabbit.setProgress(0);
                sb_turtle.setProgress(0);
                runRabbit(); //執行runRabbit方法
                runTurtle(); //執行runTurtle方法
            }
        });
    }
```

```
//建立Handler物件接收訊息
private final Handler handler = new Handler(Looper.myLooper(), new Handler.Callback() {
    @Override
    public boolean handleMessage(@NonNull Message msg) {
        //判斷編號，並更新SeekBar的進度
        if (msg.what == 1)
            sb_rabbit.setProgress(progressRabbit);
        else if (msg.what == 2)
            sb_turtle.setProgress(progressTurtle);

        //判斷誰抵達終點
        if (progressRabbit >= 100 && progressTurtle < 100) {
            Toast.makeText( context: MainActivity.this,
                    text: "兔子勝利", Toast.LENGTH_SHORT).show(); //顯示兔子勝利
            btn_start.setEnabled(true); //按鈕可操作
        } else if (progressTurtle >= 100 && progressRabbit < 100) {
            Toast.makeText( context: MainActivity.this,
                    text: "烏龜勝利", Toast.LENGTH_SHORT).show(); //顯示烏龜勝利
            btn_start.setEnabled(true); //按鈕可操作
        }
        return false;
    }
});
```

Step4

runRabbit ()中編寫執行一個 Thread 來模擬兔子的移動。

```
    //模擬兔子移動
    private void runRabbit() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                //兔子有三分之二的機率會偷懶
                boolean[] sleepProbability = { true, true, false };

                while (progressRabbit <= 100 && progressTurtle < 100) {
                    try {
                        Thread.sleep( millis: 100); //延遲0.1秒更新賽況
                        //隨機產生0~2並取得兔子偷懶的機率
                        if (sleepProbability[(int)(Math.random()*3)])
                            Thread.sleep( millis: 300); //兔子偷懶0.3秒
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    progressRabbit += 3; //每次跑三步

                    Message msg = new Message(); //建立Message物件
                    msg.what = 1;  //加入編號
                    handler.sendMessage(msg); //傳送訊息
                }
            }
        }).start();
    }
```

Step5

runTurtle ()中編寫執行一個 Thread 來模擬烏龜的移動。

```
    //模擬烏龜移動
    private void runTurtle() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                while (progressTurtle <= 100 && progressRabbit < 100) {
                    try {
                        Thread.sleep( millis: 100); //延遲0.1秒更新賽況
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    progressTurtle += 1; //每次跑一步

                    Message msg = new Message(); //建立Message物件
                    msg.what = 2;  //加入編號
                    handler.sendMessage(msg); //傳送訊息
                }
            }
        }).start();
    }
}
```