8-1-1 顯示Toast訊息 - 說明

- Android應用程式除了使用TextView元件顯示訊息字串外，我們也可以使用Toast與Log類別來顯示訊息文字和偵錯資訊。
- Toast類別屬於android.widget套件，提供類別方法可以在行動裝置顯示一個彈跳訊息框，我們可以在此訊息框顯示一段訊息文字，而且只會保留一段時間，如下圖所示：

8-1-1 顯示Toast訊息

- Toast.makeText()方法：顯示Toast訊息
- 在第7-2節範例原來是在TextView元件顯示使用的選擇，我們可以改用Toast訊息，也就是當點選ListView元件的項目，使用Toast類別的makeText()類別方法建立一段訊息文字，可以顯示使用者選擇的項目名稱，如下所示：
  - Toast.makeText(this, "你是住在: " + cities[position],
  - Toast.LENGTH_SHORT).show();

8-1-2 顯示偵錯訊息

- Log類別方法
- Java語言提供try/catch例外處理程式敘述，可以配合android.util套件的Log類別建立記錄訊息，幫助我們進行Android應用程式的偵錯。Log類別常用的類別方法，如下表所示：

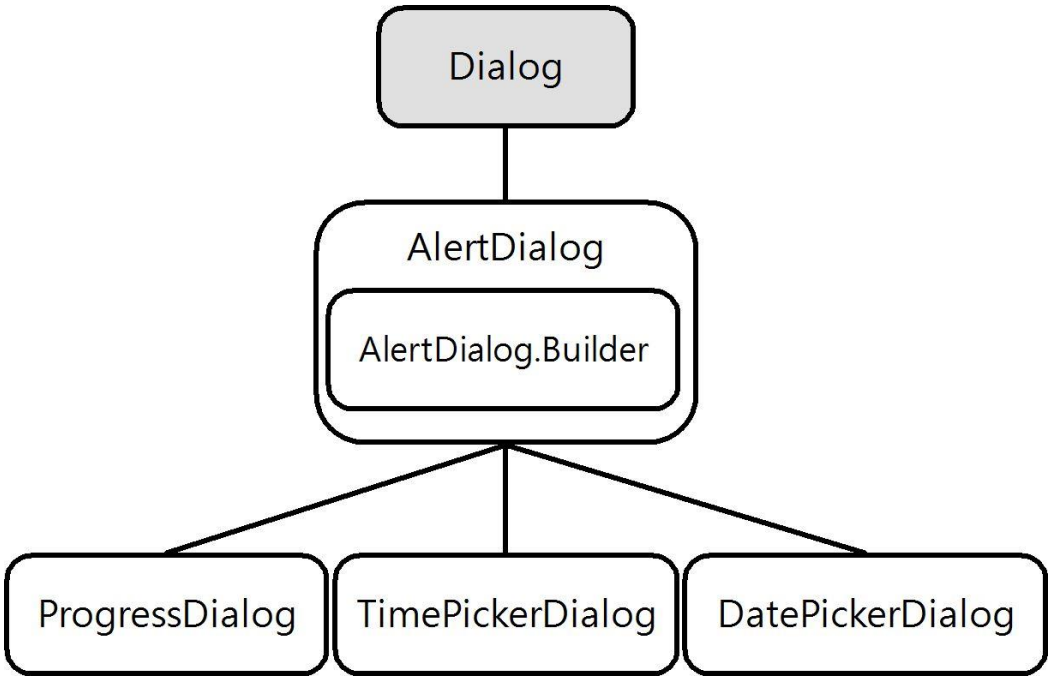
類別方法	說明
Log.e()	記錄錯誤訊息
Log.w()	記錄警告訊息
Log.i()	記錄一般資訊的訊息
Log.d()	記錄偵錯訊息
Log.v()	記錄詳細訊息

8-1-2 顯示偵錯訊息

- try/catch例外處理程式敘述
- try程式區塊：在try程式區塊的程式碼是用來檢查是否產生例外物件，當例外產生時，就會丟出指定例外類型的物件。
- catch程式區塊：當try程式區塊的程式碼丟出例外，程式需要準備一到多個catch程式區塊來處理不同類型的例外，傳入的參數e是例外類型的物件，可以使用相關方法取得進一步的例外資訊。
- finally程式區塊：finally程式區塊是一個可有可無的程式區塊，其主要目的是進行程式善後，不論例外是否產生，都會執行此程式區塊的程式碼。

8-2 對話方塊介紹 - 說明

- 對話方塊（dialog）是用來處理少量的使用者互動，每一種對話方塊都是設計用來取得使用者輸入的特定資料，例如：提供訊息、確認或選擇資料等。
- Android對話方塊是Dialog類別和其子類別，其類別架構如下圖所示：



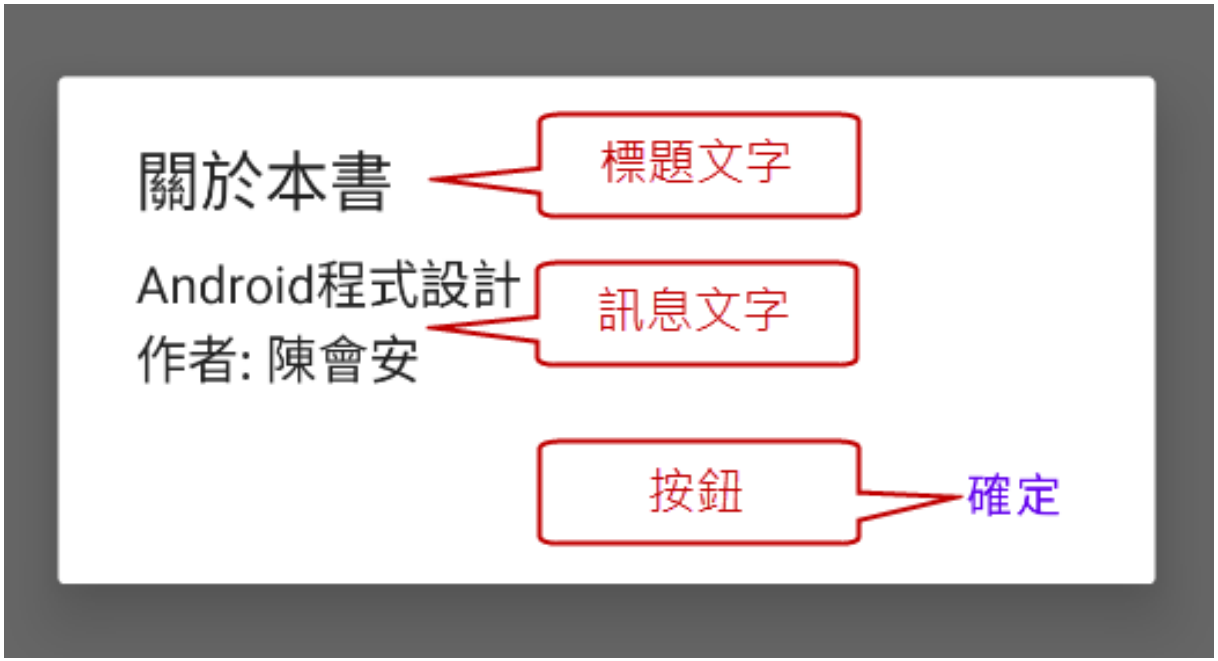
8-2 對話方塊介紹 - 種類

- 在AlertDialog類別的三個子類別可以建立特殊用途的對話方塊，其說明如下表所示：

Dialog類別	說明
ProgressDialog	顯示執行進度的對話方塊
DatePickerDialog	設定日期對話方塊，其內容是DatePicker元件，可以幫助我們設定日期
TimePickerDialog	設定時間對話方塊，其內容是TimePicker元件，可以幫助我們設定時間

8-3-1 訊息對話方塊 - 說明

- 在活動建立對話方塊是透過AlertDialog.Builder類別來建立與顯示訊息對話方塊，可以顯示一段訊息文字，例如：建立「關於本書」對話方塊顯示本書訊息，如下圖所示：



8-3-1 訊息對話方塊

- AlertDialog類別
- AlertDialog類別提供內建對話方塊元素，例如：標題、訊息文字、最多三個按鈕和可選擇的清單，可以讓我們建立訊息、複選和單選等不同功能的對話方塊，如下所示：
  - AlertDialog.Builder builder = new AlertDialog.Builder(this);
  - builder.setTitle("關於本書");
  - builder.setMessage("Android程式設計\n作者: 陳會安");
  - builder.setCancelable(true);
  - builder.show();

8-3-1 訊息對話方塊

- 在對話方塊新增【確定】按鈕
- AlertDialog.Builder物件提供三個方法來建立「確定」、「放棄」和「取消」按鈕，在對話方塊建立【確定】鈕，如下所示：
  - builder.setPositiveButton("確定", null);
- 上述setPositiveButton()方法的第1個參數是按鈕的標題文字，第2個參數是事件處理的傾聽者物件（其進一步說明，請參閱第8-3-2節），因為本節【確定】鈕只是關閉對話方塊，並沒有作什麼事，直接使用null即可。

8-3-2 確認對話方塊

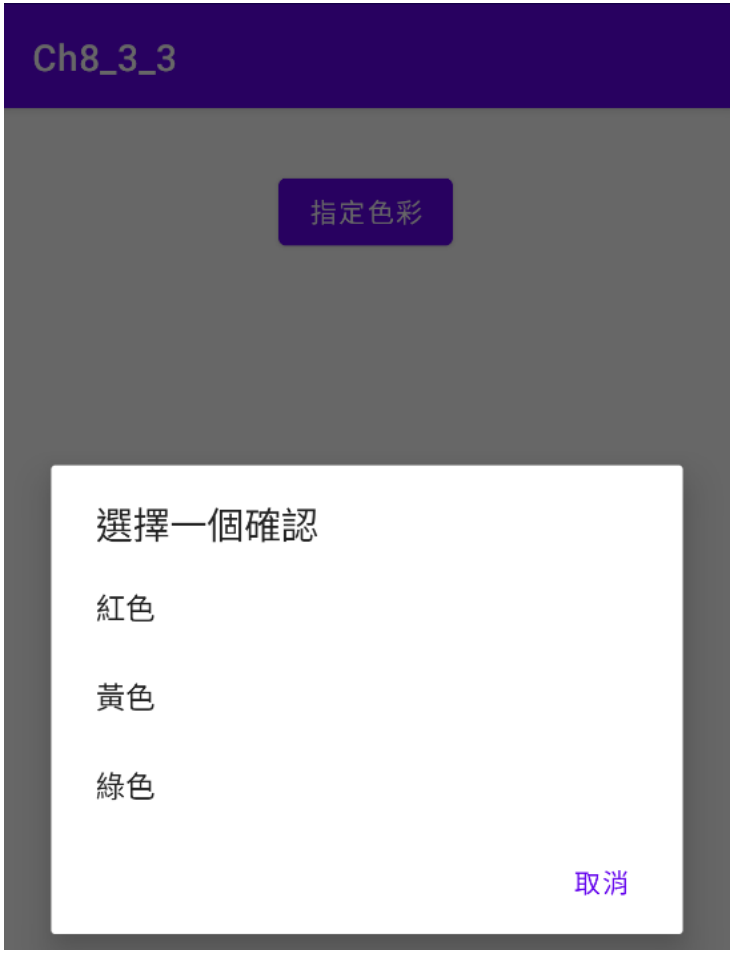
- 使用串連呼叫方法建立對話方塊
- 因為AlertDialog.Builder物件方法的傳回值都是AlertDialog.Builder物件，我們可以使用一種更簡單的方法，使用串連呼叫方法（method chaining）來建立對話方塊，如下所示：
  - builder.setTitle("確認")
  - .setMessage("確認結束本程式?")
  - .setPositiveButton("確定", this)
  - .setNegativeButton("取消", this)
  - .show();
- 上述setNegativeButton()方法是在對話方塊新增取消鈕，第2個參數的傾聽者物件是this，即活動自己。

8-3-2 確認對話方塊

- Android Studio專案：Ch8_3_2
- 在Android應用程式建立對話方塊來確認結束程式，請按【結束程式】鈕，可以顯示確認對話方塊，其執行結果如右圖所示：

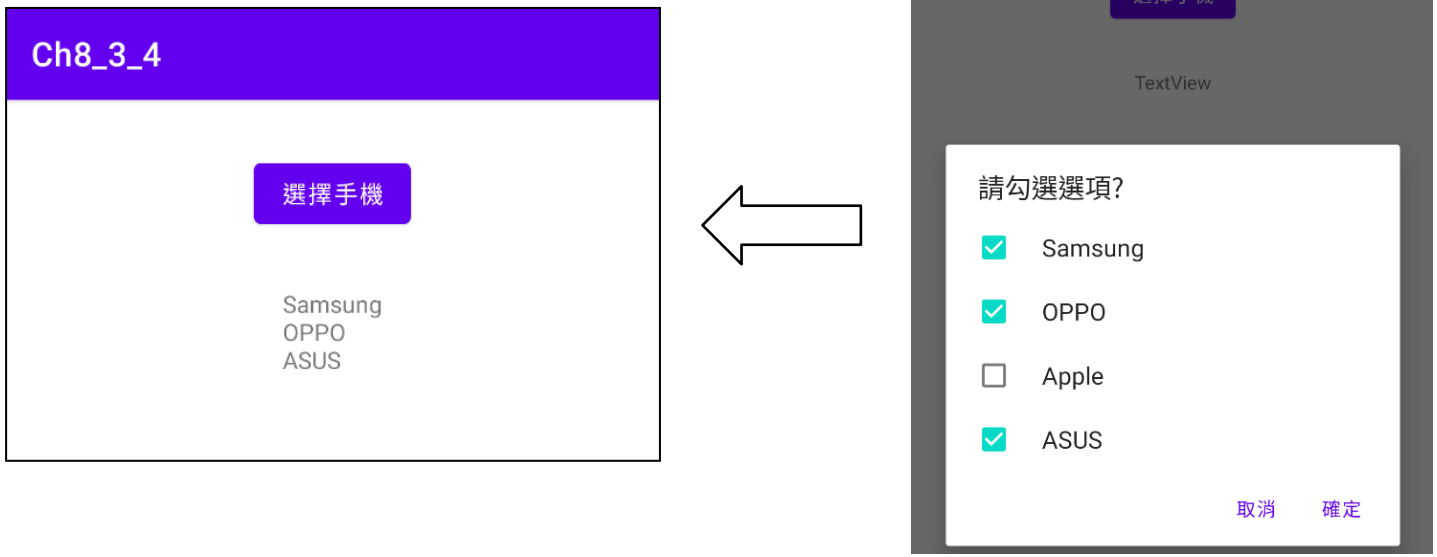
8-3-3 單選對話方塊

- Android Studio專案：Ch8_3_3
- 在Android應用程式建立色彩選擇對話方塊，可以選擇Button按鈕元件的背景色彩，其執行結果如右圖所示：

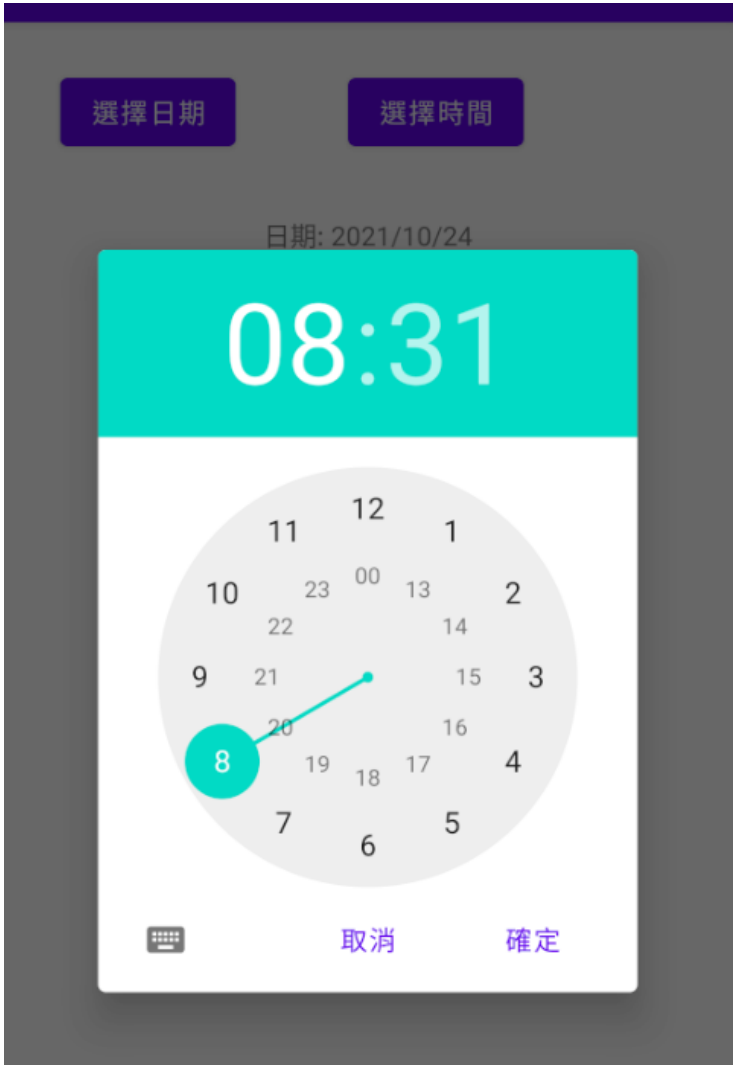


8-3-4 複選對話方塊

- Android Studio專案：Ch8_3_4
- 在Android應用程式建立複選對話方塊，可以選擇曾使用過的智慧型手機作業系統，請按【複選手機】鈕顯示複選對話方塊，其執行結果如下圖所示：




選擇日期
選擇時間
2021年
10月24日 週日
2021年10月
日 一 二 三 四 五 六
1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
取消
確定


選擇日期
選擇時間
日期: 2021/10/24
08:31
12 1 2 3 4 5 6 7 8 9 10 11
00 13 14 15 16 17 18 19 20 21 22 23
取消
確定

9-1-2 AndroidManifest.xml設定檔-說明

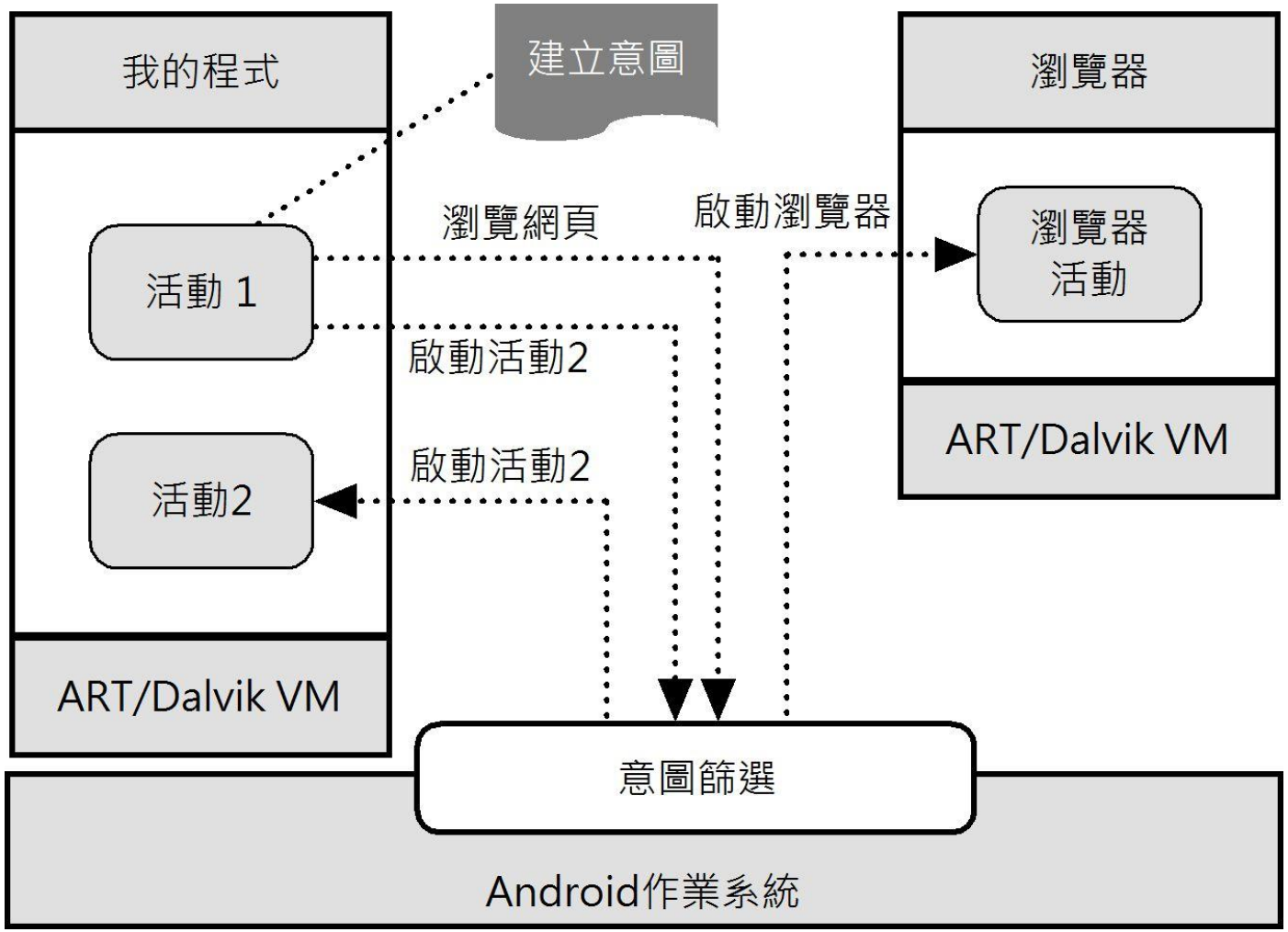
- AndroidManifest.xml設定檔一個十分重要的檔案，提供Android作業系統關於應用程式的資訊，一個功能清單。不同於Windows作業系統，Android作業系統需要透過AndroidManifest.xml檔案先認識這個應用程式，才能知道如何執行此應用程式。其主要提供的資訊有：
  - 應用程式的完整名稱（包含Java套件名稱），一個唯一的識別名稱，可以讓Android作業系統和Google Play找到此應用程式。
  - 應用程式包含的活動、內容提供者、廣播接收器和服務元件。
  - 宣告應用程式執行時需要的權限，例如：存取網路和GPS等。

9-2 意圖介紹 – 說明

- 意圖（intents）是一個啟動其他Android活動、服務和廣播接收器的系統訊息，一種抽象方式來描述希望執行的操作，可以告訴Android作業系統我想作什麼？執行什麼動作？例如：啟動其他活動、告訴指定服務可以啟動或停止與送出廣播。

9-2 意圖介紹

- **意圖與意圖篩選**
- Android應用程式送出意圖的訊息需要經過Android作業系統來判斷接收者是誰，它是使用意圖篩選（intent filters）找出有能力處理的活動或內建應用程式，然後才將訊息送給接收者，如下圖所示：


我的程式
建立意圖
瀏覽器
瀏覽網頁
啟動瀏覽器
活動 1
瀏覽器
活動
啟動活動2
ART/Dalvik VM
活動2
啟動活動2
ART/Dalvik VM
意圖篩選
Android作業系統

9-2 意圖介紹

- **意圖的種類**
- 在Android作業系統的意圖可以分為兩種，如下所示：
  - 明確意圖（explicit intent）：指明目標活動接收者名稱，即明確指明是送給誰，通常是使用在連接同一應用程式內部的多個活動，前述啟動活動2和第9-3~9-5節是明確意圖。
  - 隱含意圖（implicit intent）：意圖只有指出執行的動作型態和資料，並沒有目標接收者的確實活動名稱，Android作業系統任何可以完成此工作的應用程式都可以是接收者，即前述瀏覽網頁，在第10章是說明隱含意圖。

9-2 意圖介紹

- **使用意圖啟動的活動類型**
- Android應用程式的活動是一個佔滿行動裝置螢幕的視窗，在同一應用程式可以擁有多個活動，在活動中是一至多個介面元件建立的使用介面。Android應用程式的活動可以分成兩大類型，如下所示：
  - 獨立活動：一種沒有資料交換的活動，單純只是從一個螢幕轉換至下一個螢幕，在第9-3節使用意圖啟動的就是這種活動。
  - 相依活動：一種類似Web網頁之間資料傳遞的活動，在活動之間有資料交換，我們需要將資料傳遞至下一個活動，和取得回傳資料，在第9-4和9-5節啟動的就是這種活動。

9-3 使用意圖啟動其他活動實習

- **startActivity()方法：啟動新活動**
- 當Android應用程式擁有多個活動時，我們需要使用意圖來啟動其他活動。我們是呼叫startActivity()方法來啟動程式中的其他活動，參數的Intent意圖物件指明開啟的是哪一個活動，如下所示：

```
startActivity(intent);
```

9-3 使用意圖啟動其他活動實習

- **startActivity()方法：啟動新活動**
- 上述方法的參數是Intent意圖物件，在呼叫前，我們需要使用建構子來建立Intent物件，如下所示：

```
Intent intent = new Intent(this, SecondActivity.class);
```

- 上述建構子的第1個參數是活動自己this，第2個參數是欲開啟的活動類別SecondActivity.class，在「.」前是活動名稱，之後class表示是活動類別。

9-3 使用意圖啟動其他活動實習

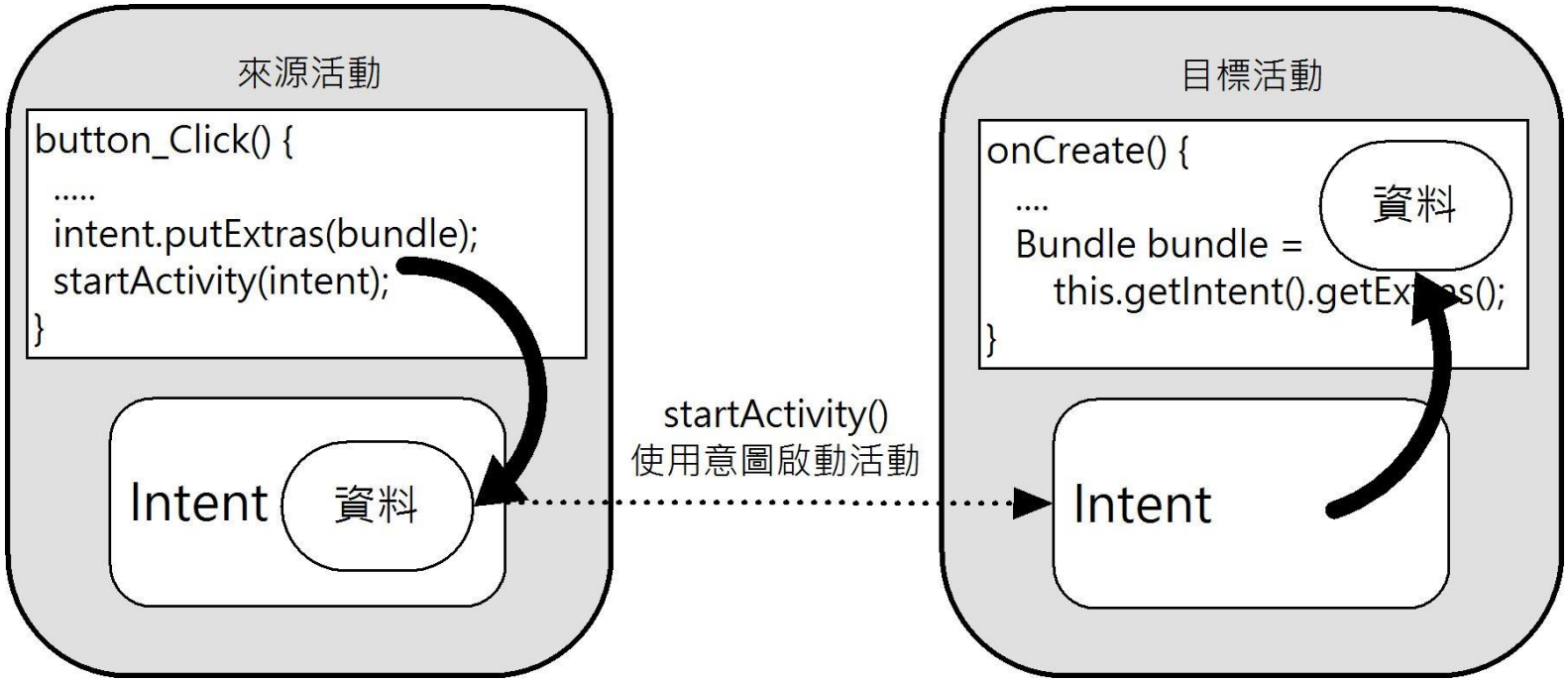
- **finish()方法：關閉活動**
- 在活動只需呼叫finish()方法，就可以關閉目前開啟的活動，如下所示：

```
finish();
```

- 請注意！如果目前活動是由來源活動所開啟，關閉活動就會回到來源活動。

9-4 傳遞資料給其他活動實習-說明

- Intent物件除了可以啟動活動，還可以攜帶資料，將這些資料一併傳遞給目標活動，例如：在主活動輸入攝氏溫度，然後將輸入值作為傳遞資料傳至目標活動來計算轉換結果的華氏溫度，如下圖所示：



9-4 傳遞資料給其他活動實習

- **建立Intent物件攜帶Bundle物件的傳遞資料**
- Intent物件攜帶資的料是Bundle物件，一種目錄物件來儲存字串型態鍵值對應的各種資料型態資料。首先建立Intent物件，如下所示：

```
Intent intent = new Intent(this, FActivity.class);
```

- 上述程式碼建立Intent物件後，我們需要先建立Bundle物件的大包包，然後將需要傳遞的資料都丟進去，如下所示：

```
Bundle bundle = new Bundle();
bundle.putString("TEMPC", txtC.getText().toString());
```

9-4 傳遞資料給其他活動實習

- **建立Intent物件攜帶Bundle物件的傳遞資料**
- 上述程式碼使用putString()方法新增字串資料，第1個參數是字串的鍵值"TEMPC"，第2個參數是值。常用的方法有putInt()放入整數、putDouble()放入浮點數和putByte()放入位元組資料等，如果資料不只一項，請重複呼叫put???()方法將資料一一放入Bundle物件。
- 接著使用Intent物件的putExtras()方法附加Bundle物件，此時的Intent物件不只有啟動活動的資訊，還攜帶有資料，然後就可以呼叫startActivity()方法啟動活動，如下所示：

```
intent.putExtras(bundle);
startActivity(intent);
```

9-4 傳遞資料給其他活動實習

- **取出Intent物件攜帶傳遞的Bundle物件資料**
- 在目標活動是呼叫活動物件的getIntent()方法取得Intent物件，然後呼叫Intent物件的getExtras()方法取得攜帶的Bundle物件，如下所示：

```
Bundle bundle = this.getIntent().getExtras();
if (bundle != null) {
  c = Integer.parseInt(bundle.getString("TEMPC"));
  ...
}
```

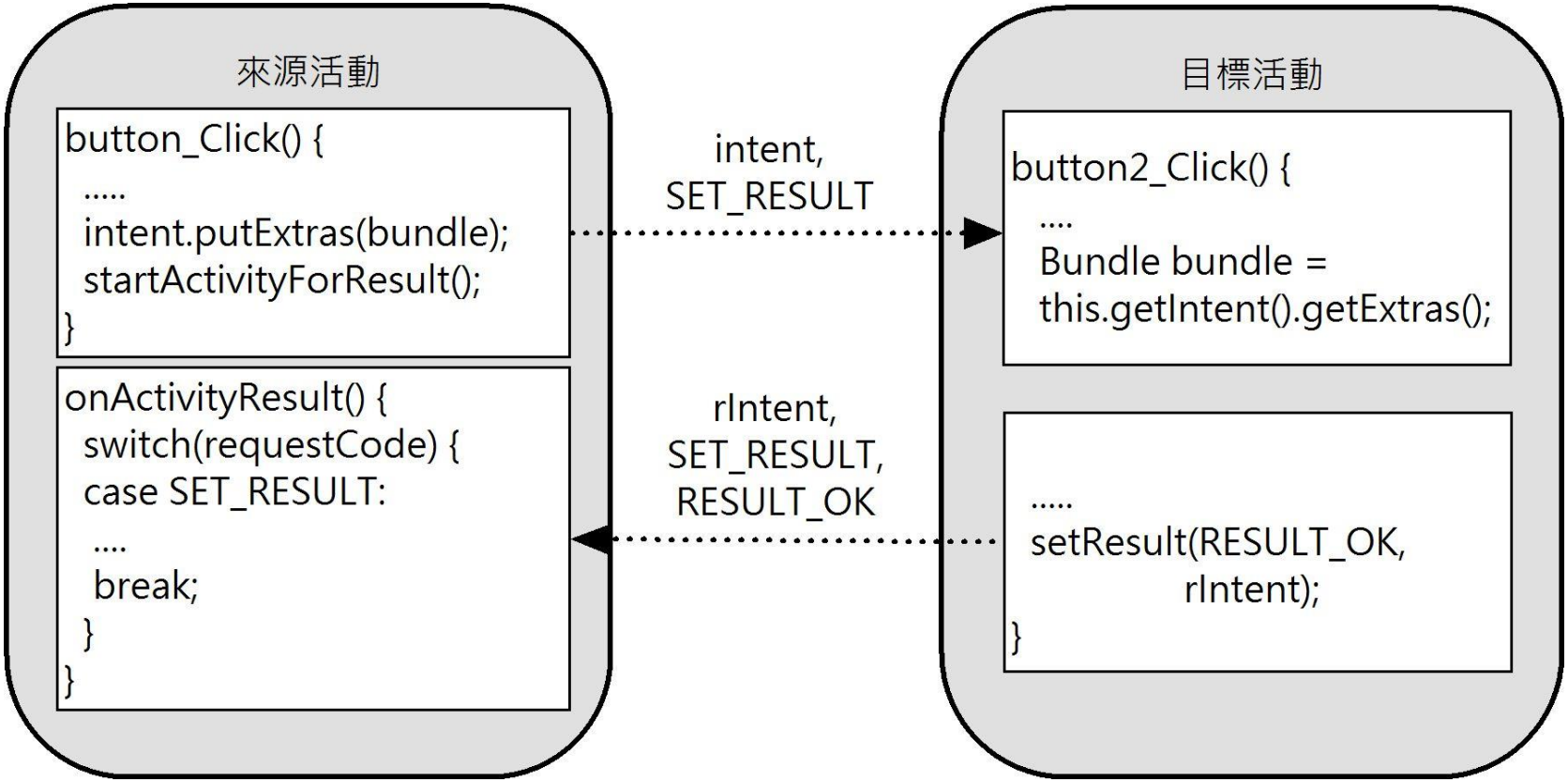
9-4 傳遞資料給其他活動實習

- **取出Intent物件攜帶傳遞的Bundle物件資料**
- 上述if條件判斷是否有攜帶資料，如果有，就使用getString()方法取出資料，參數是之前指定的字串鍵值"TEMPC"，分別對應之前的put????()方法，我們可以使用getInt()取出整數、getDouble()取出浮點數和getByte()取出位元組資料等。

9-5 取得活動的回傳資料實習-說明

- 在第9-3和9-4節的範例都是使用startActivity()方法啟動其他活動，我們只能將資料傳遞至目標活動，並不能取得活動的回傳資料，我們需要改用startActivityForResult()方法啟動活動，才能取得回傳資料。
- 本節範例是一個四則計算機，整個使用介面有2個活動，在第一個活動輸入2個運算元後，將輸入資料傳遞給第二個活動，讓使用者選擇運算子，在計算後，回傳給第一個活動來顯示。

9-5 取得活動的回傳資料實習-圖例



9-5 取得活動的回傳資料實習

- **建立Bundle物件的資料傳遞給需要回傳的活動**
- 如同第9-4節，我們需要建立Intent物件傳遞資料給需要回傳值的活動，如下所示：

```
Intent intent = new Intent(this, OpActivity.class);
Bundle bundle = new Bundle();
bundle.putString("OPERAND01",
txtOpd1.getText().toString());
bundle.putString("OPERAND02",
txtOpd2.getText().toString());
intent.putExtras(bundle);
```

- 上述程式碼建立Intent物件，建構子的第1個參數是活動自己，第2個參數是OpActivity.class，然後建立Bundle物件傳遞2個字串資料。

9-5 取得活動的回傳資料實習

- **startActivityForResult()方法：啟動需要回傳值的活動**
  - startActivityForResult()方法可以使用參數的意圖和請求碼來啟動新活動，可以取得啟動新活動的回傳值，如下所示：

    ```
    startActivityForResult(intent, SET_RESULT);
    ```

  - 上述方法的第1個參數是Intent物件，第2個參數是請求碼的整數值，此值是用來在活動中識別是哪一個活動的回傳資料（因為同一活動可能啟動多個有回傳值的活動）。

9-5 取得活動的回傳資料實習

- **在需要回傳值的活動取出傳遞的Bundle物件資料**
- 在回傳值的活動是呼叫getIntent()方法取得Intent物件後，即可取出傳遞的資料，如下所示：

```
Bundle bundle = this.getIntent().getExtras();
opd1 = Integer.parseInt(
        bundle.getString("OPERAND01"));
opd2 = Integer.parseInt(
        bundle.getString("OPERAND02"));
```

- 上述程式碼取得傳遞的2個字串，並且轉換成整數。

9-5 取得活動的回傳資料實習

- **setResult()方法：建立回傳資料回傳至來源活動**
- 在需要回傳值的活動結束前，需要建立回傳資料，使用的也是Intent物件，如下所示：

```
Intent rIntent = new Intent();
Bundle rbundle = new Bundle();
rbundle.putDouble("RESULT", result);
rIntent.putExtras(rbundle);
```

- 上述程式碼建立Bundle物件附加至Intent物件後，就可以設定結果碼（result code）來回傳資料，如下所示：

```
setResult(RESULT_OK, rIntent);
```

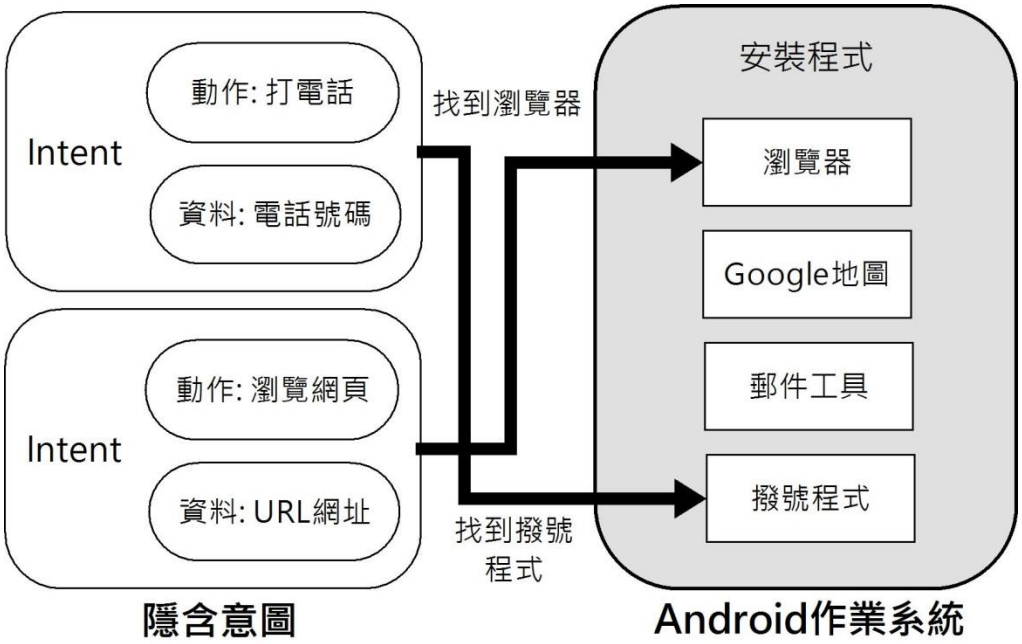
9-5 取得活動的回傳資料實習

- **onActivityResult()方法：在來源活動取出回傳的資料**
- 在呼叫的來源活動需要覆寫onActivityResult()方法取得回傳資料，方法的3個參數依序是請求碼、結果碼和Intent物件，如下所示：

```
@Override
protected void onActivityResult(int requestCode,
                int resultCode, Intent data) {
   super.onActivityResult(requestCode,resultCode,data);
   switch(requestCode) {
     case SET_RESULT:
      if (resultCode == RESULT_OK) {
        // 取得回傳值
      }
      break;
   }
}
```

10-1-1 使用隱含意圖啟動內建程式-說明

- 隱含意圖（implicit intent）只有指出執行的動作型態和資料，並沒有目標接收者的確實名稱（在第9章需指定目標的活動類別），Android作業系統安裝的程式之中，有任何1個可以完成此工作的程式都可以是接收者，如下圖所示：



10-1-2 建立隱含意圖的Intent物件

- **setAction()方法：指定意圖的動作類型**
- 意圖包含一些預先定義的動作類型，例如：ACTION_VIEW（即Intent-filter元素action子元素的android.intent.action.VIEW屬性值），這是使用在隱含意圖。Intent意圖物件可以在建構子的第1個參數指定動作類型，如下所示：

```
Intent i = new Intent(Intent.ACTION_VIEW);
```

- 上述建構子參數是動作類型常數。另一種方式是在建立Intent物件後，再呼叫setAction()方法指定動作類型，如下所示：

```
Intent i = new Intent();
i.setAction(Intent.ACTION_VIEW);
```

10-1-2 建立隱含意圖的Intent物件

- **setAction()方法：指定意圖的動作類型**
- 使用隱含意圖啟動內建程式的常用動作類型，如下表所示：

動作類型	說明
ACTION_VIEW	顯示資料給使用者檢視
ACTION_EDIT	顯示資料給使用者編輯
ACTION_DIAL	顯示撥號
ACTION_CALL	打電話
ACTION_PICK	選取URI目錄下的資料
ACTION_SENDTO	寄送電子郵件
ACTION_WEB_SEARCH	Web搜尋
ACTION_MAIN	啟動如同是程式進入點的主程式

10-1-2 建立隱含意圖的Intent物件

- **setData()方法：指定動作類別所需的資料**
- 在隱含意圖除了指定意圖使用的動作類型外，我們還需要指定目標的資料（Data）是誰，在建立Intent物件時，我們可以在第2個參數指定資料，如下所示：

```
Intent i = new Intent(Intent.ACTION_VIEW,
    Uri.parse("http://www.google.com.tw"));
```

10-1-2 建立隱含意圖的Intent物件

- **setData()方法：指定動作類別所需的資料**
- 上述建構子的第2個參數是URI（Universal Resource Identifier）字串，我們是將URL網址字串呼叫Uri.parse()方法建立成URI，這就是資料所在的位置索引。
- 如果在建構子沒有指明資料，我們可以使用Intent物件的setData()方法來指定，如下所示：

```
Intent i = new Intent(Intent.ACTION_VIEW);
i.setData(Uri.parse("http://www.google.com.tw"));
```

10-1-2 建立隱含意圖的Intent物件

- **URI（Universal Resource Identifier）**
- 萬用資源識別URI是用來定位Android系統的資源，幫助Intent意圖物件的動作取得或找到操作的資料。Android常用的URI，如下所示：
  - URL網址：URI可以直接使用URL網址，如下所示：

    http://www.google.com.tw/
  - 地圖位置：GPS定位的座標值（GeoPoint格式），如下所示：

    geo:25.046924371354 12,121.5161783959678

10-1-2 建立隱含意圖的Intent物件

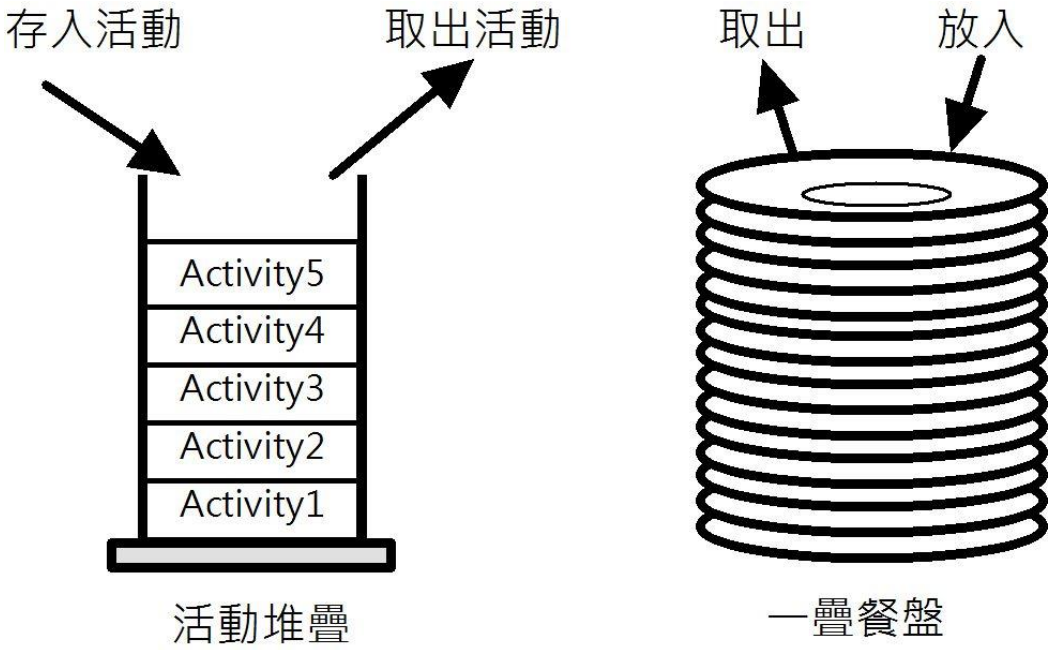
- **URI（Universal Resource Identifier）**
  - 電話號碼：指定撥打的電話號碼，如下所示：
    tel:+1234567
  - 寄送郵件：寄送郵件至指定的電子郵件地址，如下所示：
    mailto:hueyan@ms2.hinet.net

10-2 使用意圖啟動內建程式

- 10-2-1 啟動瀏覽器
- 10-2-2 啟動地圖、打電話和寄送電子郵件
- 10-2-3 輸入關鍵字執行Web搜尋
- 10-2-4 選取與顯示聯絡人資料

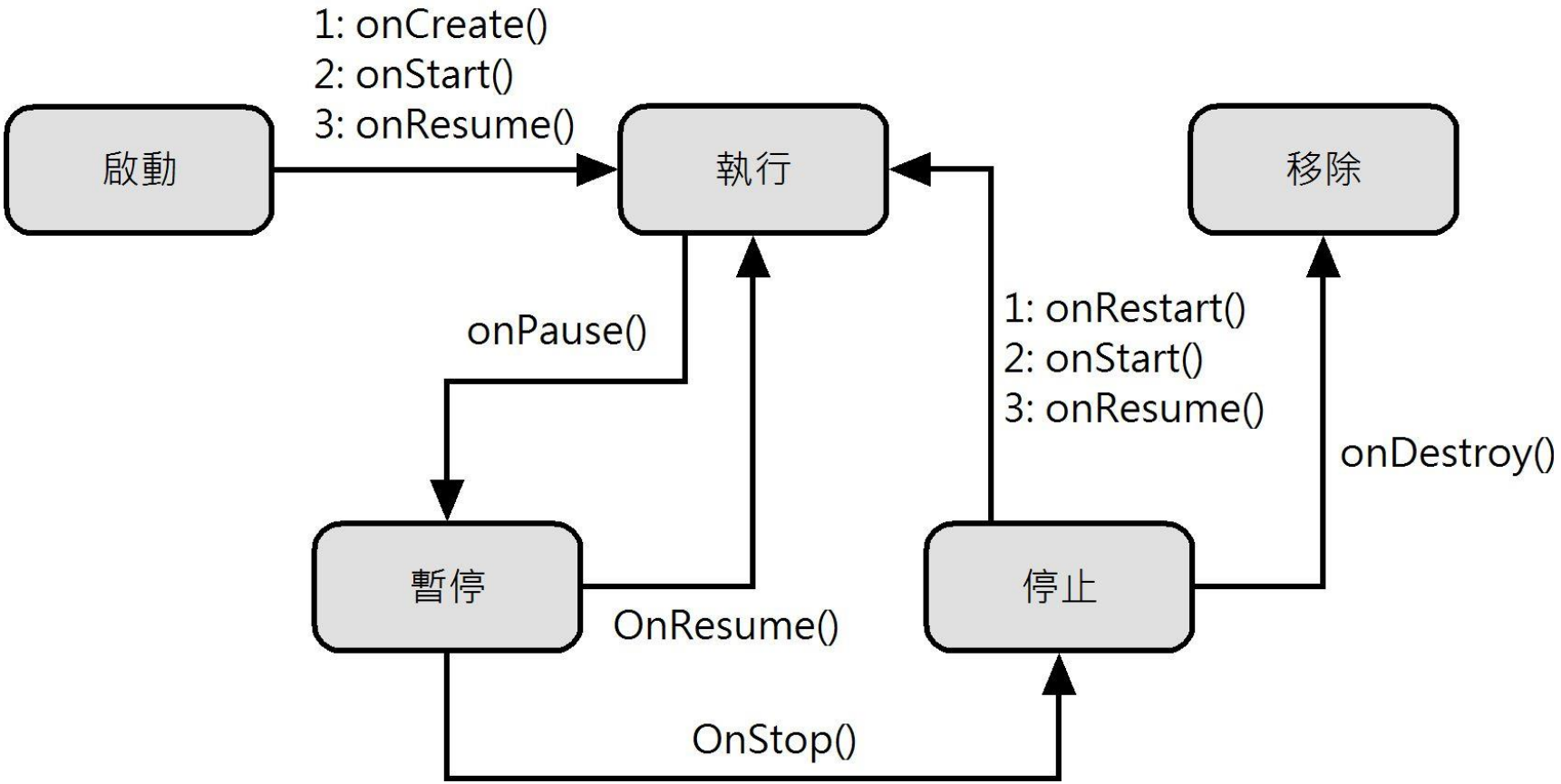
10-3-1 活動堆疊

- 對於在Android作業系統啟動中的眾多活動來說，系統是使用活動堆疊（activity stack）來管理這些活動。如同餐廳廚房的工人清洗餐盤，將洗好的餐盤疊在一起，每洗好一個餐盤就放在這疊餐盤的頂端，如下圖所示：



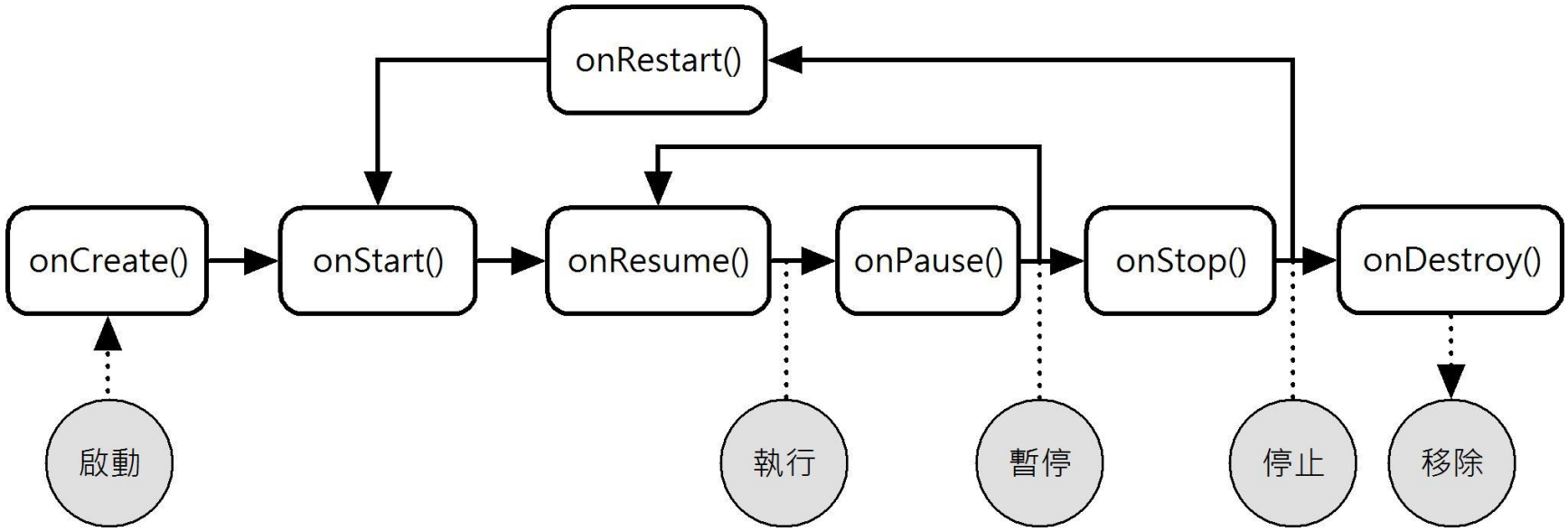
10-3-2 活動的生命周期-說明

- 活動的生命周期就是活動的狀態管理，而Android程式設計的主要工作就是撰寫程式碼來回應Android應用程式產生的狀態改變，所以，對於活動的生命周期來說，我們重視的是不同狀態之間的轉換，而不是目前位在哪一個狀態，如下圖所示：


1: onCreate()
2: onStart()
3: onResume()
啟動
執行
移除
onPause()
1: onRestart()
2: onStart()
3: onResume()
onDestroy()
暫停
停止
OnResume()
OnStop()

10-3-3 管理活動狀態

- **呼叫方法管理活動的狀態**
- 在活動整個生命周期共有7個方法會在活動的狀態轉換時呼叫，這些方法是開發者回應狀態改變，撰寫Java程式碼的地方，如下圖所示：



10-3-3 管理活動狀態-方法的簡單說明

方法	說明
onCreate()	在活動建立時呼叫，可以在此方法執行活動的初始化，即建立活動的使用介面元件，我們可以將此方法視為活動的進入點
onStart()	在使用者被看見時呼叫
onResume()	在與使用者互動時呼叫
onPause()	在暫停目前活動時呼叫，例如：顯示對話方塊，通常我們會在此方法儲存尚未儲存的資料，和任何使用者變更的資料
onStop()	在使用者看不見時呼叫，可能情況有三種：啟動新活動、之前活動返回螢幕或活動將被刪除
onRestart()	在活動重新返回螢幕時呼叫，例如：收到簡訊，停止目前的活動，等到閱讀完簡訊後，呼叫此方法返回之前的活動
onDestroy()	在刪除活動前呼叫

11-4 使用內建相機照相-說明

- 一般來說，Android行動裝置內建硬體相機，也有相機程式，我們可以使用Intent啟動相機來照相，然後在ImageView元件顯示照相結果的相片。

11-4 使用內建相機照相

- **使用Intent啟動相機程式**
- 我們是使用Intent啟動Android內建相機程式，如下所示：

```
Intent intent = new Intent(
                MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(intent, REQUEST_IMAGE);
```

- 上述程式碼的動作類型是ACTION_IMAGE_CAPTURE，然後呼叫startActivityForResult()方法開啟有回傳值的活動，回傳碼是REQUEST_IMAGE，可以開啟內建相機程式。

11-4 使用內建相機照相

- **onActivityResult()方法：顯示照相結果**
- 因為startActivityForResult()方法啟動有回傳值的相機程式，所以我們可以在onActivityResult()方法取得回傳相片，在建立成Bitmap物件後，顯示在ImageView元件，如下所示：

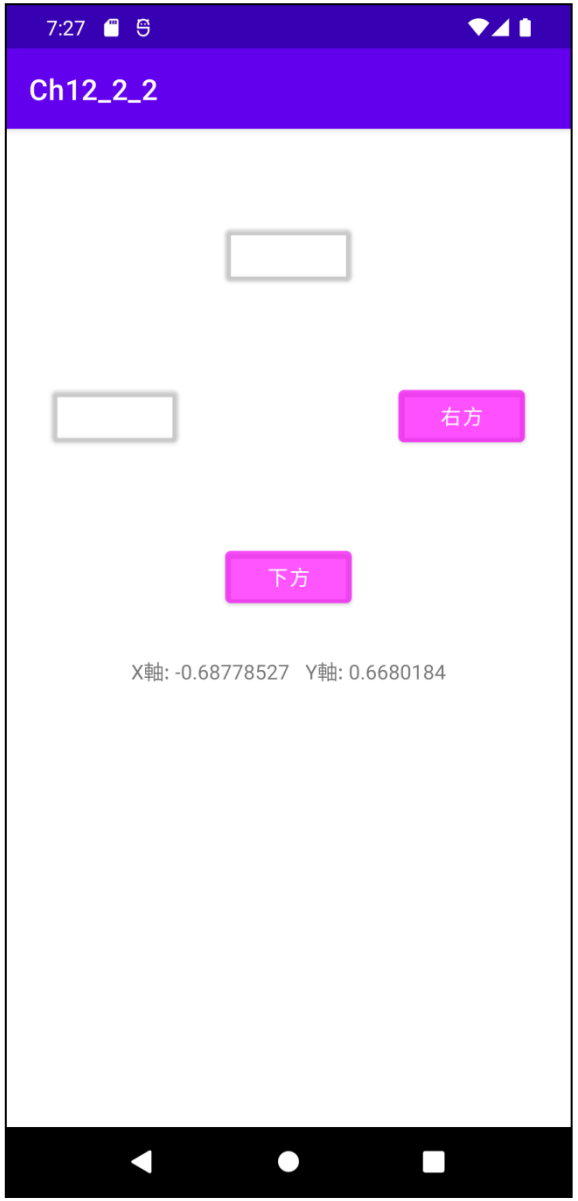
```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {
   if (requestCode == REQUEST_IMAGE &&
         resultCode == Activity.RESULT_OK) {
      Bitmap userImage = (Bitmap)
data.getExtras().get("data");
      image.setImageBitmap(userImage);
   }
}
```

11-4 使用內建相機照相

- **Android Studio專案：Ch11_4**
- Android應用程式是一個簡單的照相程式，使用Intent啟動行動裝置內建相機程式來照相，然後在下方ImageView元件顯示取得的相片，其執行結果如右圖所示：

12-2-2 行動裝置傾斜偵測

- **Android Studio專案：Ch12_2_2**
- 在Android應用程式取得加速感測器X和Y軸的值來判斷行動裝置是否有傾斜，以便在此方向的Button元件顯示不同深淺的背景顏色，其執行結果如右圖所示：



12-3 數位羅盤：指南針 – 說明

- 數位羅盤（digital compass）可以取得行動裝置指向的方向，即所謂的指南針，這是使用磁場感測器和加速感測器提供的資料來計算出裝置是指向哪一個方向。