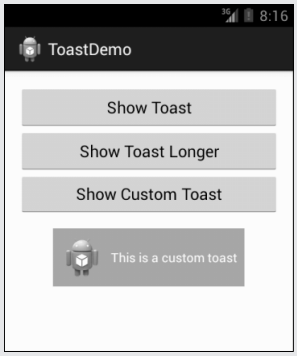
CHAPTER

01 Toast และ Activity

เนื้อหาในบทนี้

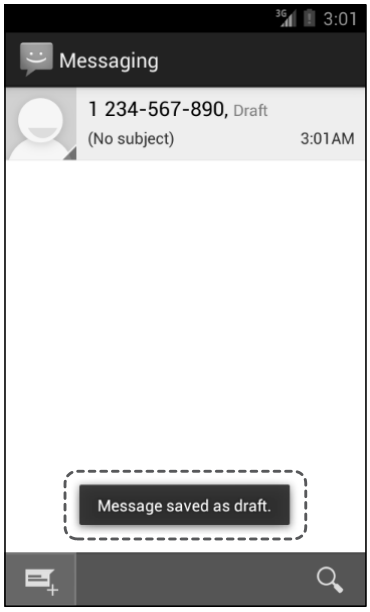
- การแสดง Toast
- การแสดง Toast ให้นานขึ้น
- การแสดง Toast ที่เราออกแบบหน้าตาเอง (Custom Toast)



- การรันแอคทิวิตีใหม่ขึ้นมา
- การส่งข้อมูลไปให้แอคทิวิตีปลายทาง
- การส่งออบเจ็คไปให้แอคทิวิตีปลายทาง
- การส่งข้อมูลกลับไปให้แอคทิวิตีต้นทาง

การแสดง Toast

Toast คือข้อความที่แสดงออกมาครู่หนึ่งแล้วหายไปเอง เราใช้ Toast แจ้งข่าวสารแก่ผู้ใช้ในเรื่องที่ไม่สำคัญมากซึ่งเป็นผลจากการทำงานหนึ่งๆ ยกตัวอย่างเช่น เมื่อผู้ใช้ออกจากหน้าจอพิมพ์ข้อความ SMS โดยที่ยังไม่ได้ส่งข้อความนั้น จะปรากฏ Toast ที่บอกให้รู้ว่าข้อความถูกบันทึกเก็บไว้ เพื่อให้ผู้ใช้กลับมาเขียนต่อและส่งได้ในภายหลัง ดังรูป



ตัวอย่าง

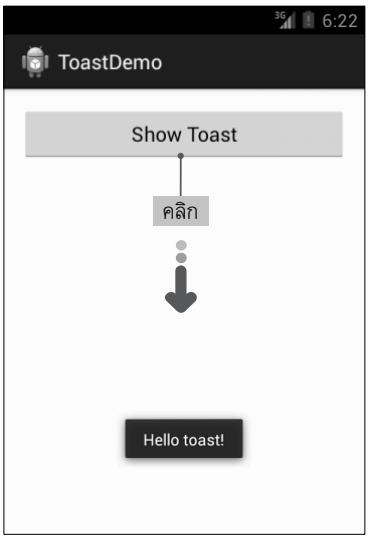
ตัวอย่างนี้จะแสดง Toast ในเมธอด onClick ของปุ่มหนึ่ง

โปรเจ็ค ToastDemo, ไฟล์ MainActivity.java

```
Button btnShowToast = (Button) findViewById(R.id.show_toast_button);
btnShowToast.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        Context context = getApplicationContext();
        CharSequence text = "Hello toast!";
        int duration = Toast.LENGTH_SHORT;

        Toast toast = Toast.makeText(context, text, duration);
        toast.show();
    }
});
```

ผลการรัน

เมื่อคลิก (แตะ) ปุ่มจะปรากฏ Toast ที่มีข้อความว่า “Hello toast!”



คำอธิบาย

การแสดง Toast นั้นก่อนอื่นเราต้องสร้างออบเจ็ค `Toast` โดยใช้เมธอด `makeText` ซึ่งเมธอดนี้มีพารามิเตอร์ 3 ตัว ได้แก่ คอนเท็กซ์ (Application Context), ข้อความที่ต้องการแสดง และระยะเวลาในการแสดง Toast

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
```

ระยะเวลาในการแสดง Toast สามารถระบุได้ 2 แบบคือ `Toast.LENGTH_SHORT` (ประมาณ 2 วินาที) และ `Toast.LENGTH_LONG` (ประมาณ 3.5 วินาที)

เมธอด `makeText` จะส่งคืนออบเจ็ค `Toast` กลับมาให้ เราจะเรียกเมธอด `show` ของออบเจ็คนี้เพื่อแสดง Toast ออกมาบนหน้าจอ

```
toast.show();
```

ถ้าคุณต้องการประหยัดการใช้ตัวแปร ก็สามารถยุบโค้ดทั้งหมดเหลือบรรทัดเดียวได้ ดังนี้

```
Toast.makeText(getApplicationContext, "Hello toast!",
               Toast.LENGTH_SHORT).show();
```

TIP

กำหนดตำแหน่งของ Toast

โดยปกติ Toast จะถูกแสดงบริเวณด้านล่างและกึ่งกลางจอในแนวนอน แต่คุณสามารถเปลี่ยนตำแหน่งในการแสดง Toast โดยใช้เมธอด setGravity เช่นหากต้องการแสดง Toast ที่มุมซ้ายบน ให้เรียกเมธอด setGravity ดังนี้

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

ถ้าหากต้องการขยับ Toast มาทางขวา ให้เพิ่มค่าพารามิเตอร์ตัวที่ 2 และหากต้องการขยับ Toast ลงมาให้เพิ่มค่าพารามิเตอร์ตัวที่ 3

การแสดง Toast ให้นานขึ้น

ดังที่อธิบายแล้วว่าเราสามารถกำหนดระยะเวลาในการแสดง Toast ได้ 2 แบบ คือแบบสั้นและแบบยาว ซึ่งจะแสดง Toast ประมาณ 2 วินาทีและ 3.5 วินาที ตามลำดับ

คำถามคือ ถ้าหากเราต้องการแสดง Toast นานกว่า 3.5 วินาที จะทำได้อย่างไร? ถึงแม้แอนดรอยด์จะไม่มีทางออกในเรื่องนี้ตรงๆ แต่เราสามารถใช้วิธีแสดง Toast เดิมออกมาซ้ำๆกันเรื่อยๆ จนดูเหมือน Toast ถูกแสดงค้างไว้บนหน้าจอเป็นเวลานานได้

ตัวอย่าง

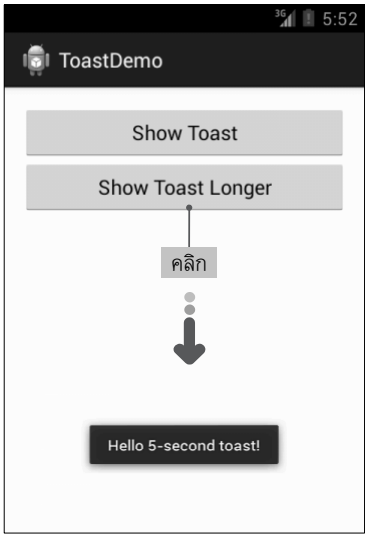
ตัวอย่างนี้จะแสดง Toast ประมาณ 5 วินาที

โปรเจ็ค ToastDemo, ไฟล์ MainActivity.java

```
Button btnShowToastLonger = (Button) findViewById(
                                        R.id. .show_toast_longer_button);
btnShowToastLonger.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        Context context = getApplicationContext();
        CharSequence text = "Hello 5-second toast!";
        int duration = Toast.LENGTH_SHORT;

        final Toast toast = Toast.makeText(context, text, duration);
        toast.show();

        new CountDownTimer(5000, 1000) {
            public void onTick(long millisUntilFinished) { toast.show(); }
            public void onFinish() { }
        }.start();
    }
});
```

ผลการรัน

เมื่อคลิกปุ่มจะปรากฏ Toast ที่มีข้อความว่า “Hello 5-second toast!” ซึ่งแสดงค้างบนหน้าจอประมาณ 5 วินาที



คำอธิบาย

เราสร้างออบเจ็ค `CountDownTimer` เพื่อให้เรียก `toast.show()` เป็นเวลา 5 วินาที ให้สังเกตว่าต้องประกาศตัวแปร `toast` เป็นแบบ `final` ด้วย มิฉะนั้นเราจะไม่สามารถเรียกใช้ตัวแปรนี้ในเมธอดของ Inner Class ได้ (คือเมธอด `onTick` ในที่นี้) ตามกฎเกณฑ์ของภาษาจาวา

การแสดง Toast ที่เราออกแบบหน้าตาเอง (Custom Toast)

นอกจาก Toast แบบมาตรฐาน แอนดรอยด์ยังมีวิธีให้เราแสดง Toast โดยใช้ Layout ที่เรากำหนดเองได้ด้วย หลักการคือ ให้สร้าง Layout File ที่กำหนดหน้าตาของ Toast แล้วนำอิลิเมนต์ที่เป็นรูทวิว (Root View) ของ Layout File นั้นไประบุให้เมธอด `setView` ในโค้ดจาวา

ตัวอย่าง

1 ออกแบบ Layout ของ Toast

โปรเจ็ค ToastDemo, ไฟล์ res\layout\toast_layout.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/toast_layout_root"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#80ff0066"
    android:orientation="horizontal"
```

```
    android:padding="8dp" >

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="8dp"
        android:src="@drawable/ic_launcher" />

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:textColor="#FFF" />

</LinearLayout>
```

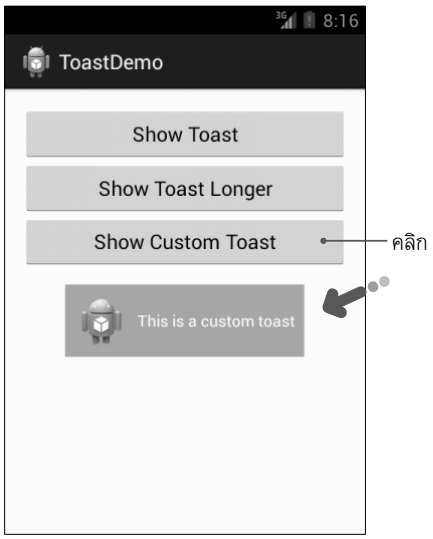
2 เขียนโค้ดในแอคทิวิตี

โปรเจ็ค ToastDemo, ไฟล์ MainActivity.java

```
Button btnShowCustomToast = (Button) findViewById(
                                        R.id.show_custom_toast_button);
btnShowCustomToast.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        LayoutInflater inflater = getLayoutInflater();
        View layout = inflater.inflate(R.layout.toast_layout,
                       (ViewGroup) findViewById(R.id.toast_layout_root));

        TextView text = (TextView) layout.findViewById(R.id.text);
        text.setText("This is a custom toast");

        Toast toast = new Toast(getApplicationContext());
        toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
        toast.setDuration(Toast.LENGTH_LONG);
        toast.setView(layout);
        toast.show();
    }
});
```

ผลการรัน

เมื่อคลิกปุ่ม Show Custom Toast จะปรากฏ Toast ที่เราออกแบบ Layout เอง ดังรูป



คำอธิบาย

เริ่มต้นเราเรียกเมธอด `getLayoutInflater` เพื่อเข้าถึงออบเจ็ค `LayoutInflater` จากนั้นจึงใช้เมธอด `inflate` ของออบเจ็คดังกล่าวสร้าง Layout ขึ้นมาจากไฟล์ XML โดยต้องระบุพารามิเตอร์ 2 ตัว ตัวแรกคือ Layout File ของ Toast และตัวที่สองคืออิลิเมนต์ที่เป็นรูทวิวใน Layout File นั้น

```
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.toast_layout,
              (ViewGroup) findViewById(R.id.toast_layout_root));
```

ต่อมากำหนดข้อความให้กับเท็กซ์วิวใน Layout

```
TextView text = (TextView) layout.findViewById(R.id.text);
text.setText("This is a custom toast");
```

สุดท้ายสร้างออบเจ็ค `Toast` ด้วยคีย์เวิร์ด `new` (ไม่ได้ใช้เมธอด `makeText` เหมือนตัวอย่างก่อนๆ) แล้วกำหนดคุณสมบัติต่างๆให้กับ Toast หนึ่งในนั้นคือการกำหนด Layout ของ Toast ด้วยเมธอด `setView` แล้วจึงแสดง Toast ออกมาด้วยเมธอด `show`

```
Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
toast.setView(layout);
toast.show();
```

การรันแอคทิวิตีใหม่ขึ้นมา

หน้าจอหนึ่งๆของแอพแอนดรอยด์ประกอบด้วย 2 ส่วนหลักๆ ได้แก่

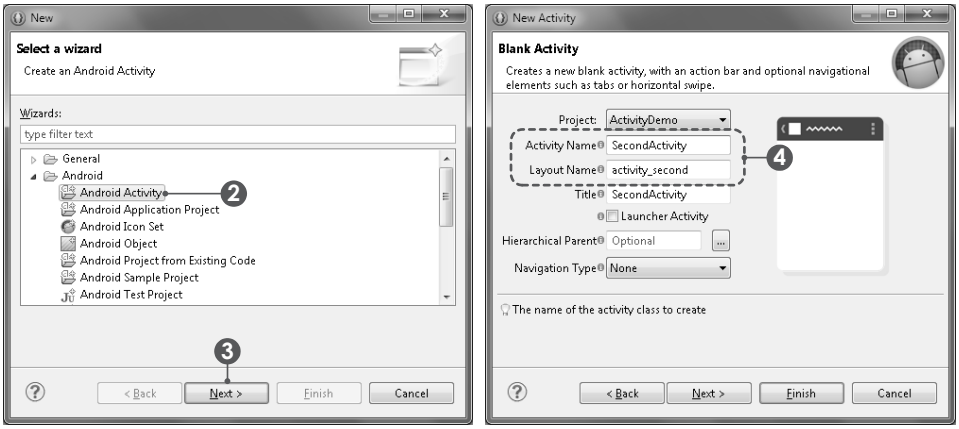
- **แอคทิวิตี (Activity)** คือจาวาคลาส (Java Class) ที่ควบคุมการทำงานของหน้าจอ เช่น การจัดการอีเวนต์ต่างๆที่เกิดขึ้นในหน้าจอ แอคทิวิตีจะเขียนด้วยภาษาจาวา และเก็บในไฟล์ที่มีนามสกุล .java
- **Layout File** ทำหน้าที่กำหนด Layout ของหน้าจอ ไฟล์นี้เขียนด้วยภาษา XML และมีนามสกุลของไฟล์เป็น .xml

โดยทั่วไป เราจะเรียกเมธอด `setContentView` ภายในเมธอด `onCreate` ของแอคทิวิตี เพื่อนำ Layout File มากำหนดเป็นส่วนติดต่อผู้ใช้หรือ User Interface (UI) ของแอคทิวิตี ดังรูป

เมื่อคุณใช้ Eclipse สร้างโปรเจ็คแอนดรอยด์ใหม่ Eclipse จะสร้างไฟล์ทั้งสองสำหรับหน้าจอหลักของแอพให้อัตโนมัติอยู่แล้ว แต่หากคุณต้องการให้แอพมีมากกว่า 1 หน้าจอ คุณจะต้องสร้างแอคทิวิตีและ Layout File สำหรับหน้าจออื่นๆเอง รวมถึงเชื่อมโยงการทำงานระหว่างหน้าจอเองด้วย

ตัวอย่าง

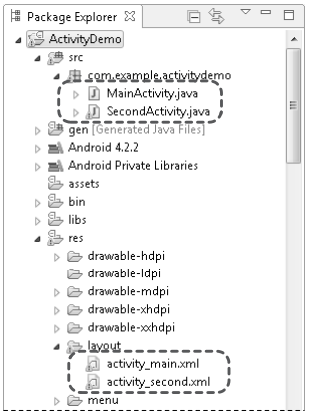
ตัวอย่างนี้จะแสดงการสร้างแอคทิวิตีใหม่ในโปรเจ็ค โดยตั้งชื่อว่า `SecondActivity` และมี Layout File ชื่อ activity_second.xml จากนั้นจะสร้างปุ่มใน `MainActivity` เพื่อเรียกไปยัง `SecondActivity`



1. ก่อนอื่นให้สร้างแอคทิวิตี (และ Layout File) ขึ้นมาใหม่ในโปรเจ็ค โดยคลิกเมนู **File ▸ New ▸ Other...**
2. ขยายหัวข้อ **Android** จากนั้นเลือก **Android Activity**
3. คลิก **Next** ไปเรื่อยๆจนถึงหน้า Blank Activity
4. ตั้งชื่อแอคทิวิตี (ชื่อคลาส) ว่า **SecondActivity** และตั้งชื่อ Layout ว่า **activity_second** แล้วคลิก **Finish**

   Eclipse จะสร้างแอคทิวิตี, Layout File และทำการประกาศแอคทิวิตีใน Manifest (ไฟล์ AndroidManifest.xml) ให้เสร็จสรรพ

ตอนนี้ในโปรเจ็คจะมีแอคทิวิตีและ Layout File อยู่ 2 ชุด คือ **MainActivity.java/activity_main.xml** (ที่ Eclipse สร้างไว้ให้ตั้งแต่แรก) กับ **SecondActivity.java/activity_second.xml** (ที่เราเพิ่งสร้างขึ้นมาเอง)



ต่อไปจะเชื่อมโยงหน้าจอทั้งสองเข้าด้วยกัน โดยสร้างปุ่มในหน้าจอหลัก สำหรับเรียกหน้าจอที่สองขึ้นมา

5 สร้างปุ่มใน Layout File ของหน้าจอหลัก

โปรเจ็ค ActivityDemo, ไฟล์ res\layout\activity__main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp" >

    <Button
        android:id="@+id/display_second_activity_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Display Second Activity" />

</LinearLayout>
```

6 Layout File ของหน้าจอที่สองมี `TextView` แสดงข้อความ “This is the second activity.”

โปรเจ็ค ActivityDemo, ไฟล์ res\layout\activity__second.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:text="This is the second activity." />
</LinearLayout>
```

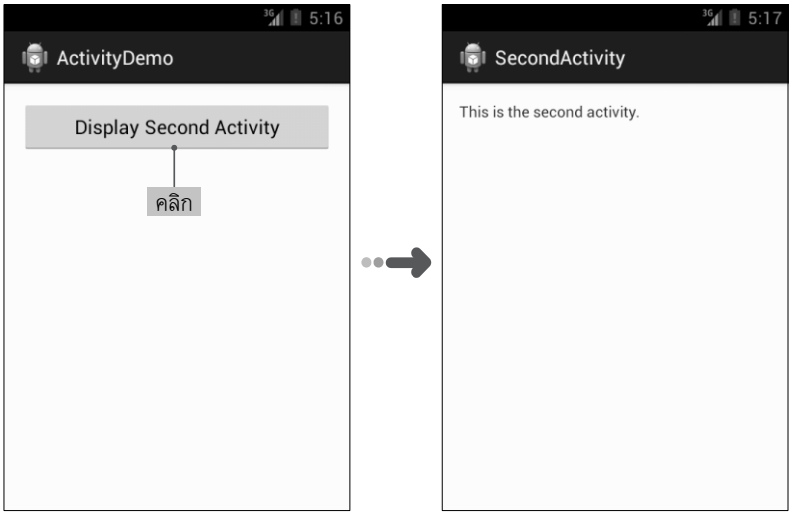
7 เขียนโค้ดควบคุมการคลิกปุ่มในแอคทิวิตีหลัก

โปรเจ็ค ActivityDemo, ไฟล์ MainActivity.java

```
Button btnDisplaySecondActivity = (Button) findViewById(
                                        R.id.display_second_activity_button);
btnDisplaySecondActivity.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        Intent i = new Intent(MainActivity.this, SecondActivity.class);
        startActivity(i);
    }
});
```

ผลการรัน

เมื่อคลิกปุ่มในแอคทิวิตีหลักจะทำให้แอคทิวิตีที่สองถูกรันขึ้นมาโดยซ้อนอยู่บนแอคทิวิตีหลัก ดังรูป ผู้ใช้สามารถกลับไปยังแอคทิวิตีหลักได้โดยกดหรือแตะปุ่ม Back ที่ตัวเครื่อง หรือเราอาจเรียกใช้เมธอด `finish` ในแอคทิวิตีที่สองเพื่อปิดแอคทิวิตีนี้และกลับไปยังแอคทิวิตีหลักก็ได้เช่นกัน (เช่น อาจสร้างปุ่มในแอคทิวิตีที่สอง สำหรับย้อนกลับไปยังแอคทิวิตีหลัก)



ให้สังเกตว่าข้อความบน Title bar ของแอคทิวิตีทั้งสองไม่เหมือนกัน ซึ่งข้อความบน Title bar ของแอคทิวิตีจะกำหนดด้วยแอตทริบิวต์ `android:label` ภายในการประกาศแอคทิวิตีในไฟล์ AndroidManifest.xml

```
...
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
        android:name="com.example.activitydemo.MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name="com.example.activitydemo.SecondActivity"
        android:label="@string/title_activity_second" >
    </activity>
</application>
...
```

ในที่นี้ข้อความบน Title bar ของ `MainActivity` คือค่าสตริง `app_name` ในรีซอร์ส ซึ่ง Eclipse กำหนดให้โดยใช้ชื่อแอพที่เราระบุตอนสร้างโปรเจ็ค ส่วนข้อความบน Title bar ของ `SecondActivity` คือค่าสตริง `title_activity_second` ในรีซอร์ส ซึ่ง Eclipse กำหนดให้อีกเช่นกันโดยใช้ชื่อแอคทิวิตีที่เราระบุตอนสร้างแอคทิวิตีใหม่ ถ้าหากคุณต้องการเปลี่ยนข้อความบน Title bar ก็สามารถแก้ไขได้ที่ค่าสตริงทั้งสองนี้

คำอธิบาย

การรันแอคทิวิตีอื่นขึ้นมาจะทำได้โดยใช้อินเทนต์ ซึ่งอินเทนต์ (Intent) คือออบเจ็คที่ทำหน้าที่เชื่อมโยงการทำงานระหว่างคอมโพเนนต์ต่างๆในระบบแอนดรอยด์ เช่น ระหว่างแอคทิวิตี 2 แอคทิวิตี หรือระหว่างแอคทิวิตีกับเซอร์วิส เป็นต้น

ในที่นี้เราสร้างอินเทนต์ด้วยโค้ด

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

พารามิเตอร์ตัวแรกคือคอนเท็กซ์ของแอพ เราระบุ `MainActivity.this` ซึ่งหมายถึงอินสแทนซ์ปัจจุบันของแอคทิวิตีนี้ (คลาส `MainActivity`) เนื่องจากคลาส `Activity` สืบทอดมาจากคลาส `Context` และคลาส `MainActivity` ของเราก็สืบทอดมาจากคลาส `Activity` อีกทีหนึ่ง ดังนั้น `MainActivity` จึงถือว่าเป็น `Context` ด้วย

NOTE

ในกรณีส่วนใหญ่เราระบุพารามิเตอร์ตัวแรกเป็น this เฉยๆก็ได้ แต่ในที่นี้ใช้ไม่ได้เพราะโค้ดข้างต้นอยู่ภายใน Inner Class ที่เรากำหนดเป็น OnClick Listener ให้กับปุ่ม ถ้าหากระบุ this เฉยๆจะหมายถึงอินสแทนซ์ของ Inner Class ดังกล่าว ไม่ใช่อินสแทนซ์ของแอคทิวิตี

สำหรับพารามิเตอร์ตัวที่สองจะระบุคลาสของคอมโพเนนต์ที่เราต้องการเรียกขึ้นมา ในที่นี้ก็คือแอคทิวิตีที่มีชื่อว่า SecondActivity

หลังจากสร้างอินเทนต์ที่ระบุคอนเท็กซ์และแอคทิวิตีปลายทางเรียบร้อยแล้ว ถัดไปให้เรียกเมธอด startActivity โดยระบุอินเทนต์นั้นเป็นพารามิเตอร์ เพื่อรันแอคทิวิตีขึ้นมาจริงๆ

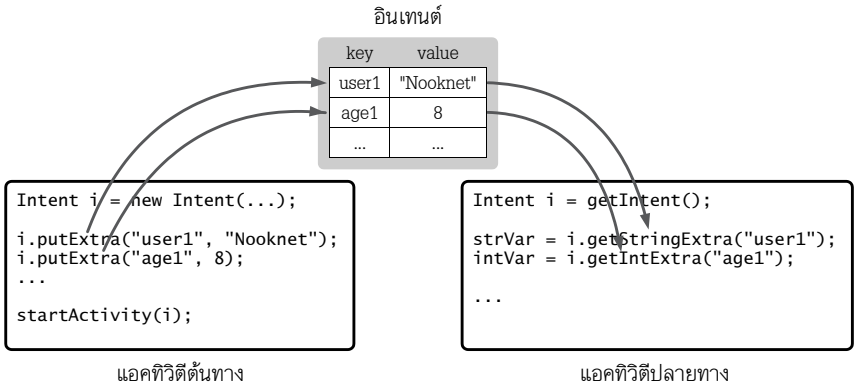
```
startActivity(i);
```

การรันแอคทิวิตีอื่น หรือถ้าพูดภาษาผู้ใช้ก็คือแสดงหน้าจออีกหน้าจอหนึ่งขึ้นมา จะเขียนโค้ดง่ายๆแค่ 2 บรรทัดข้างต้น อย่างไรก็ตาม ถ้าหากคุณต้องการส่งข้อมูลจากแอคทิวิตีต้นทางไปยังแอคทิวิตีปลายทางด้วย คุณจะต้องเขียนโค้ดมากกว่านั้น ซึ่งไม่ยากเลย และจะขออธิบายในหัวข้อต่อจากนี้

การส่งข้อมูลไปให้แอคทิวิตีปลายทาง

การเชื่อมโยงระหว่างแอคทิวิตีคงจะไม่ค่อยมีประโยชน์นักถ้าหากเราไม่สามารถส่งผ่านข้อมูลระหว่างแอคทิวิตีได้ แอนดรอยด์มีวิธีให้เราส่งข้อมูลทั้งสองทาง คือจากแอคทิวิตีต้นทางไปยังแอคทิวิตีปลายทาง และจากแอคทิวิตีปลายทางกลับมายังแอคทิวิตีต้นทาง ในหัวข้อนี้จะขออธิบายอย่างแรกก่อน

การส่งข้อมูลจากแอคทิวิตีต้นทางไปยังแอคทิวิตีปลายทาง มีหลักการคือ เราจะใส่ข้อมูลลงในอินเทนต์ที่แอคทิวิตีต้นทาง แล้วอ่านข้อมูลจากอินเทนต์มาใช้งานที่แอคทิวิตีปลายทาง ดังรูป



ตัวอย่าง

ตัวอย่างนี้จะมีปุ่มในแอคทิวิตีหลัก ซึ่งเมื่อคลิกจะรันแอคทิวิตีที่สองพร้อมทั้งส่งข้อมูลไปให้ แล้วแอคทิวิตีที่สองจะอ่านข้อมูลเหล่านั้นมาแสดงใน Toast

1 สร้างปุ่มใน Layout File ของแอคทิวิตีหลัก สำหรับรันและส่งข้อมูลไปให้แอคทิวิตีที่สอง

โปรเจ็ค ActivityDemo, ไฟล์ res\layout\activity__main.xml

```
<Button
    android:id="@+id/send_data_button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Send Data to Second Activity" />
```

2 เขียนโค้ดระบุการทำงานของปุ่มดังกล่าว

โปรเจ็ค ActivityDemo, ไฟล์ MainActivity.java

```
Button btnSendData = (Button) findViewById(R.id.send_data_button);
btnSendData.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        Intent i = new Intent(MainActivity.this, SecondActivity.class);

        i.putExtra("user1", "Nooknet");
        i.putExtra("age1", 8);
        i.putExtra("user2", "Numnahm");
        i.putExtra("age2", 5);

        Bundle bundle = new Bundle();
        bundle.putString("user3", "Tonyod");
        bundle.putInt("age3", 3);
        bundle.putString("user4", "Yui");
        bundle.putInt("age4", 41);

        startActivity(i);
    }
});
```

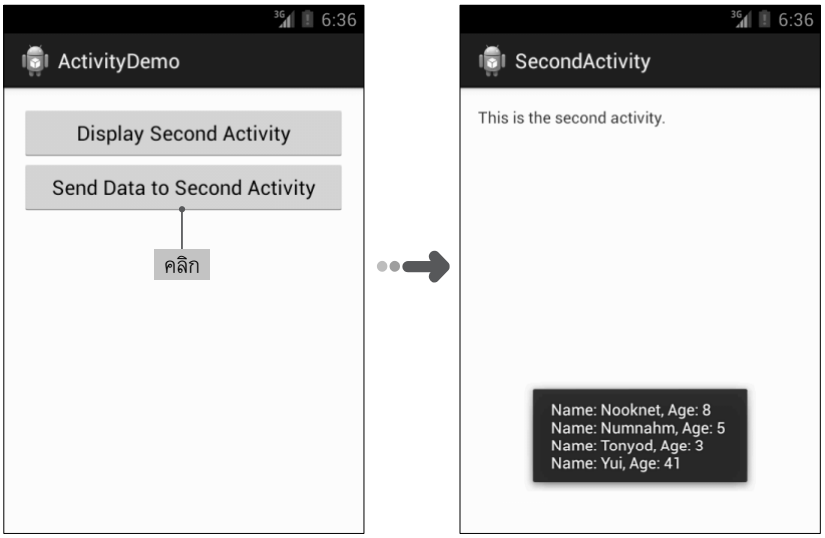
3 เขียนโค้ดในเมธอด onCreate ของแอคทิวิตีที่สอง เพื่ออ่านข้อมูลที่แอคทิวิตีหลักส่งมาให้และนำมาแสดงใน Toast

โปรเจ็ค ActivityDemo, ไฟล์ SecondActivity.java

```
String text;
Intent i = getIntent();
```

```
text = "Name: " + i.getStringExtra("user1");
text += ", Age: " + i.getIntExtra("age1", 0) + "\n";
text += "Name: " + i.getStringExtra("user2");
text += ", Age: " + i.getIntExtra("age2", 0) + "\n";

Bundle bundle = i.getExtras();

text += "Name: " + bundle.getString("user3");
text += ", Age: " + bundle.getInt("age3") + "\n";
text += "Name: " + bundle.getString("user4");
text += ", Age: " + bundle.getInt("age4");

Toast.makeText(this, text, Toast.LENGTH_LONG).show();
```

ผลการรัน



คำอธิบาย

การใส่ข้อมูลลงในอินเทนต์

เมื่อต้องการส่งข้อมูลไปให้แอคทิวิตีปลายทาง เราจะใส่ข้อมูลลงในอินเทนต์ด้วยเมธอด `putExtra`

```
i.putExtra("user1", "Nooknet");
i.putExtra("age1", 8);
i.putExtra("user2", "Numnahm");
i.putExtra("age2", 5);
```

ข้อมูลที่ใส่ลงในอินเทนต์จะมีรูปแบบเป็น “คีย์/ค่า” (key/value) โดย “คีย์” ก็คือชื่อที่ใช้อ้างถึงข้อมูลนั้นเวลาอ่านข้อมูลมาใช้ในแอคทิวิตีปลายทาง ส่วน “ค่า” สามารถระบุได้หลายชนิด เช่น ค่าสตริง ค่าจำนวนเต็ม ค่าทศนิยม ค่าตรรกะ รวมทั้งอาร์เรย์ของชนิดข้อมูลเหล่านี้ด้วย

โค้ด 4 บรรทัดข้างต้นเป็นการใส่ข้อมูล 4 ข้อมูลลงในอินเทนต์ โดยเป็นสตริง 2 ข้อมูล และเป็นเลขจำนวนเต็ม 2 ข้อมูล

นอกจากการใช้เมธอด `putExtra` เรายังสามารถเก็บข้อมูลไว้ในอินเทนต์ได้อีกวิธีหนึ่ง โดยใส่ข้อมูลลงในออบเจ็ค `Bundle` ก่อน แล้วค่อยนำออบเจ็ค `Bundle` ไปใส่ลงในอินเทนต์อีกที ก่อนอื่นต้องสร้างออบเจ็ค `Bundle` ขึ้นมา

```
Bundle bundle = new Bundle();
```

จากนั้นให้เรียกเมธอด เช่น `putString`, `putInt` เพื่อใส่ข้อมูลลงใน `Bundle`

```
bundle.putString("user3", "Tonyod");
bundle.putInt("age3", 3);
bundle.putString("user4", "Yui");
bundle.putInt("age4", 41);
```

ข้อมูลที่ใส่ลง `Bundle` จะมีรูปแบบเป็น “คีย์/ค่า” เช่นเดียวกัน เพียงแต่เราต้องเรียกใช้เมธอดให้เหมาะสมกับชนิดของข้อมูลที่จะใส่ลงไป

สุดท้ายต้องนำออบเจ็ค `Bundle` มาใส่ลงในอินเทนต์ด้วยเมธอด `putExtras` (สังเกตว่ามี s ท้ายชื่อเมธอด)

```
i.putExtras(bundle);
```

ตอนนี้อินเทนต์จะเก็บข้อมูลไว้ดังรูป และเมื่อเรียกเมธอด `startActivity(i)` อินเทนต์และข้อมูลทั้งหมดก็จะถูกส่งไปยังแอคทิวิตีปลายทาง

อินเทนต์

key	value
user1	"Nooknet"
age1	8
user2	"Numnahm"
age2	5

Bundle

key	value
user3	"Tonyod"
age3	3
user4	"Yui"
age4	41

การอ่านข้อมูลจากอินเทนต์

ที่แอคทิวิตีปลายทาง เราสามารถอ่านข้อมูลจากอินเทนต์มาใช้งานได้ โดยก่อนอื่นให้เรียกเมธอด `getIntent` เพื่อเข้าถึงอินเทนต์

```
Intent i = getIntent();
```

จากนั้นให้เรียกเมธอดต่างๆของอินเทนต์ เช่น getStringExtra, getIntExtra ฯลฯ เพื่ออ่านข้อมูลที่ต้องการ โดยต้องเลือกใช้เมธอดให้เหมาะสมกับชนิดของข้อมูลที่จะอ่าน

```
text = "Name: " + i.getStringExtra("user1");
text += ", Age: " + i.getIntExtra("age1", 0) + "\n";
text += "Name: " + i.getStringExtra("user2");
text += ", Age: " + i.getIntExtra("age2", 0) + "\n";
```

สำหรับออบเจ็ค Bundle ที่เก็บอยู่ในอินเทนต์ ให้เรียกเมธอด getExtras เพื่อเข้าถึงออบเจ็ค Bundle ก่อน

```
Bundle bundle = i.getExtras();
```

จากนั้นจึงเรียกเมธอดต่างๆของ Bundle เช่น getString, getInt ฯลฯ เพื่ออ่านข้อมูลที่ต้องการจาก Bundle โดยต้องเลือกใช้เมธอดให้เหมาะสมกับชนิดของข้อมูลที่จะอ่าน

```
text += "Name: " + bundle.getString("user3");
text += ", Age: " + bundle.getInt("age3") + "\n";
text += "Name: " + bundle.getString("user4");
text += ", Age: " + bundle.getInt("age4");
```

เราอ่านข้อมูลต่างๆจากอินเทนต์มาจัดรูปแบบแล้วเก็บลงตัวแปร text สุดท้ายจึงแสดงค่าของตัวแปรนี้ออกมาใน Toast

```
Toast.makeText(this, text, Toast.LENGTH_LONG).show();
```

การส่งออบเจ็คไปให้แอคทิวิตีปลายทาง

บางครั้งข้อมูลที่เราต้องการส่งไปให้แอคทิวิตีปลายทางอาจไม่ใช่ชนิดข้อมูลง่ายๆอย่างเช่นสตริงหรือเลขจำนวนเต็ม แต่อาจเป็นออบเจ็คทั้งออบเจ็ค ซึ่งแอนดรอยด์มีทางออกให้เราในเรื่องนี้ โดยหากเราใช้เมธอด putExtra ใส่ออบเจ็คลงในอินเทนต์ที่แอคทิวิตีต้นทาง แอนดรอยด์ก็ได้เตรียมเมธอด getSerializableExtra ไว้ให้เราอ่านออบเจ็คนั้นกลับขึ้นมาที่แอคทิวิตีปลายทาง

ตัวอย่าง

ออบเจ็คที่เราส่งไปให้แอคทิวิตีปลายทางมักเป็นออบเจ็คที่สร้างจากคลาสที่เรากำหนดขึ้นเอง (Custom Class) ดังนั้นตัวอย่างนี้จะมีการสร้างคลาส MyUser สำหรับเก็บข้อมูลผู้ใช้ ซึ่งในคลาสมีข้อมูล 2 อย่าง คือ ชื่อกับอายุ และมี Getter Method และ Setter Method สำหรับอ่านและกำหนดข้อมูลทั้งสองในแต่ละออบเจ็คที่สร้างจากคลาส ตามแนวทางของการเขียนโปรแกรมเชิงวัตถุหรือ OOP

1 สร้างคลาส MyUser

โปรเจ็ค ActivityDemo, ไฟล์ MyUser.java

```
package com.example.activitydemo;

import java.io.Serializable;

public class MyUser implements Serializable {
    private String m_name;
    private int m_age;

    public void setName(String name) {
        m_name = name;
    }

    public String getName() {
        return m_name;
    }

    public void setAge(int age) {
        m_age = age;
    }

    public int getAge() {
        return m_age;
    }
}
```

เมื่อต้องการใส่ออบเจ็คลงในอินเทนต์ คุณต้องแน่ใจว่าคลาสที่ใช้สร้างออบเจ็คนั้นทำการ Implement อินเทอร์เฟซ Serializable

2 เพิ่มปุ่มใน Layout File ของแอคทิวิตีหลัก

โปรเจ็ค ActivityDemo, ไฟล์ res\layout\activity__main.xml

```
<Button
    android:id="@+id/send_object_button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Send Object to Second Activity" />
```

3 เขียนโค้ดระบุการทำงานของปุ่มดังกล่าว

โปรเจ็ค ActivityDemo, ไฟล์ MainActivity.java

```
Button btnSendObject = (Button) findViewById(R.id.send_object_button);
btnSendObject.setOnClickListener(new View.OnClickListener() {
```

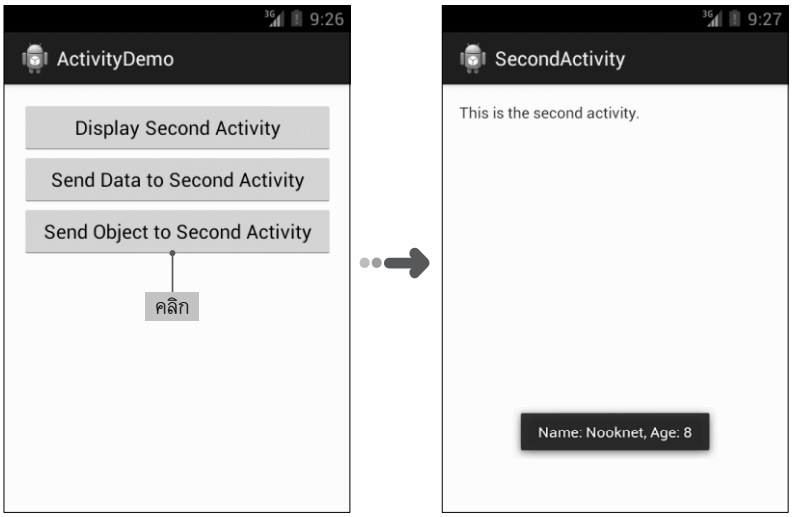
```
    @Override
    public void onClick(View v) {
        MyUser user = new MyUser();
        user.setName("Nooknet");
        user.setAge(8);

        Intent i = new Intent(MainActivity.this, SecondActivity.class);
        i.putExtra("someuser", user);
        startActivity(i);
    }
});
```

4 เขียนโค้ดในเมธอด onCreate ของแอคทิวิตีที่สอง เพื่ออ่านออบเจ็คกลับขึ้นมาจากอินเทนต์ แล้วอ่านข้อมูลผู้ใช้จากออบเจ็คนั้นมาแสดงใน Toast

โปรเจ็ค ActivityDemo, ไฟล์ SecondActivity.java

```
String text;
Intent i = getIntent();

MyUser user = (MyUser) i.getSerializableExtra("someuser");
text = "Name: " + user.getName();
text += ", Age: " + user.getAge();

Toast.makeText(this, text, Toast.LENGTH_LONG).show();
```

ผลการรัน

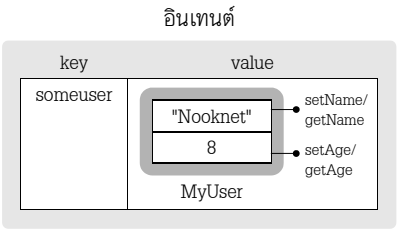


คำอธิบาย

การใส่ออบเจ็คลงในอินเทนต์จะใช้เมธอด `putExtra` เช่นเดียวกับชนิดข้อมูลพื้นฐาน เพียงแต่กรณีของออบเจ็คนั้นเราต้องสร้าง (Instantiate) มันขึ้นมาจากคลาส แล้วกำหนดข้อมูลต่างๆในออบเจ็คโดยใช้ Setter Method ที่คลาสเตรียมไว้ให้ (ได้แก่เมธอด `setName` และ `setAge` ในที่นี้)

```
MyUser user = new MyUser();
user.setName("Nooknet");
user.setAge(8);

Intent i = new Intent(MainActivity.this, SecondActivity.class);
i.putExtra("someuser", user);
```

ข้อมูลที่เก็บอยู่ในอินเทนต์จะมีลักษณะดังรูป



สำหรับการอ่านข้อมูลจากอินเทนต์มาใช้ในแอคทิวิตีปลายทาง หลังจากเรียกเมธอด `getIntent` เพื่อเข้าถึงอินเทนต์แล้ว ให้เรียกเมธอด `getSerializableExtra` เพื่ออ่านออบเจ็คขึ้นมาจากอินเทนต์ แล้วเรียกใช้ Getter Method เพื่ออ่านข้อมูลในออบเจ็คมาใช้งาน

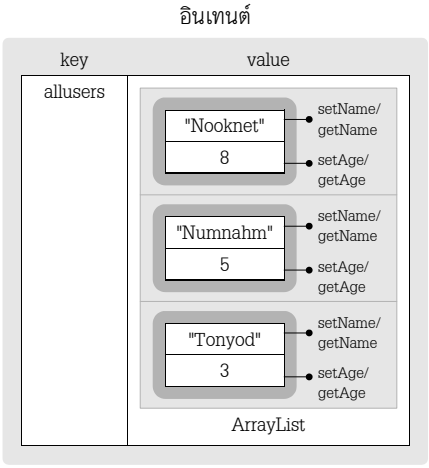
```
MyUser user = (MyUser) i.getSerializableExtra("someuser");
text = "Name: " + user.getName();
text += ", Age: " + user.getAge();

Toast.makeText(this, text, Toast.LENGTH_LONG).show();
```

การส่งหลายออบเจ็คพร้อมกัน

กรณีต้องการส่งข้อมูลผู้ใช้มากกว่า 1 คนไปให้แอคทิวิตีปลายทาง เราอาจใช้เมธอด `putExtra` ใส่ออบเจ็ค `MyUser` เพิ่มลงในอินเทนต์ตามจำนวนที่ต้องการ แต่วิธีที่ดีกว่าคือ ให้เก็บออบเจ็ค `MyUser` ทั้งหมดไว้ใน `ArrayList` ก่อน แล้วใส่ `ArrayList` ลงในอินเทนต์ (คลาส `ArrayList` นั้น Implement อินเทอร์เฟส `Serializable` เราจึงสามารถใส่ `ArrayList` ลงในอินเทนต์เพื่อส่งไปให้แอคทิวิตีปลายทางได้) ดังตัวอย่าง

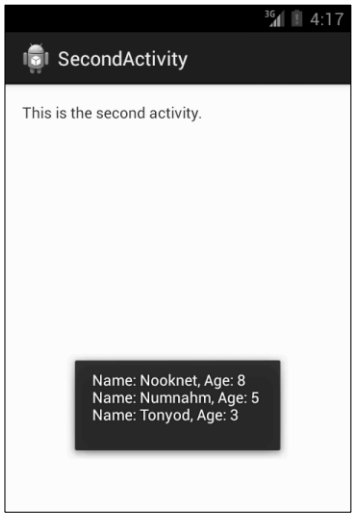
```
ArrayList<MyUser> list = new ArrayList<MyUser>();
MyUser user;

user = new MyUser();
user.setName("Nooknet");
user.setAge(8);
list.add(user);

user = new MyUser();
user.setName("Numnahm");
user.setAge(5);
list.add(user);

user = new MyUser();
user.setName("Tonyod");
user.setAge(3);
list.add(user);

Intent i = new Intent(MainActivity.this, SecondActivity.class);
i.putExtra("allusers", list);
startActivity(i);
```

ข้อมูลที่เก็บอยู่ในอินเทนต์จะมีลักษณะดังรูป



สำหรับการอ่านข้อมูลจากอินเทนต์ ก็ให้เรียกเมธอด `getSerializableExtra` เพื่ออ่าน `ArrayList` จากอินเทนต์ขึ้นมา จากนั้นวนลูปเพื่อเข้าถึงออบเจ็ค `MyUser` แต่ละออบเจ็คใน `ArrayList` แล้วจึงเรียกใช้ Getter Method เพื่ออ่านข้อมูลในออบเจ็คมาใช้งาน

```
String text = "";
Intent i = getIntent();

ArrayList<MyUser> userList = (ArrayList<MyUser>)
                             i.getSerializableExtra("allusers");
for (MyUser u : userList) {
    text += "Name: " + u.getName();
    text += ", Age: " + String.valueOf(u.getAge()) + "\n";
}
```



การส่งข้อมูลกลับไปให้แอคทิวิตีต้นทาง

ที่ผ่านมาได้อธิบายการส่งข้อมูลจากแอคทิวิตีต้นทางไปยังแอคทิวิตีปลายทาง สำหรับหัวข้อนี้จะตรงกันข้าม คือเราจะส่งข้อมูลจากแอคทิวิตีปลายทางกลับไปยังแอคทิวิตีต้นทาง ซึ่งข้อมูลนี้มักเป็นผลลัพธ์จากการทำงานของแอคทิวิตีปลายทาง

การส่งข้อมูลกลับไปยังแอคทิวิตีต้นทางจะใช้การใส่ข้อมูลลงในอินเทนต์เช่นเดียวกัน (แต่จะเป็นคนละออบเจ็คกับอินเทนต์ที่แอคทิวิตีต้นทางส่งมา) โดยเราต้องเรียกเมธอด `startActivityForResult` ที่แอคทิวิตีต้นทางเพื่อรันแอคทิวิตีปลายทาง แล้วเรียกใช้เมธอด `setResult` ที่แอคทิวิตีปลายทางเมื่อต้องการส่งข้อมูลกลับไปยังแอคทิวิตีต้นทาง

ตัวอย่าง

ตัวอย่างนี้จะมีแอคทิวิตีที่สองเป็นหน้าจอที่ให้ผู้ใช้ป้อนชื่อ ซึ่งเมื่อคลิก OK จะส่งชื่อที่ป้อนกลับไปยังแอคทิวิตีหลัก และแอคทิวิตีหลักจะแสดงชื่อนั้นออกมาใน Toast

1 Layout File ของแอคทิวิตีหลัก

โปรเจ็ค ActivityReturnDemo, ไฟล์ res\layout\activity__main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp" >

    <Button
        android:id="@+id/display_second_activity_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Display Second Activity" />

</LinearLayout>
```

2 ที่แอคทิวิตีหลัก ให้เพิ่มโค้ดในเมธอด `onCreate` เพื่อระบุการทำงานของปุ่ม และเพิ่มเมธอด `onActivityResult` เพื่ออ่านผลลัพธ์จากแอคทิวิตีปลายทางมาแสดงใน Toast

โปรเจ็ค ActivityReturnDemo, ไฟล์ MainActivity.java

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

    Button btn = (Button) findViewById(
                              R.id.display_second_activity_button);
    btn.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            Intent i = new Intent(MainActivity.this, SecondActivity.class);
            startActivityForResult(i, 123);
        }
    });
}
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
                                Intent data) {
    if (requestCode == 123) {
        if (resultCode == RESULT_OK) {
            String strName = data.getStringExtra("username");
            Toast.makeText(this, strName, Toast.LENGTH_LONG).show();
        }
    }
}
```

3 Layout File ของแอคทิวิตีที่สอง

โปรเจ็ค ActivityReturnDemo, ไฟล์ res\layout\activity__second.xml

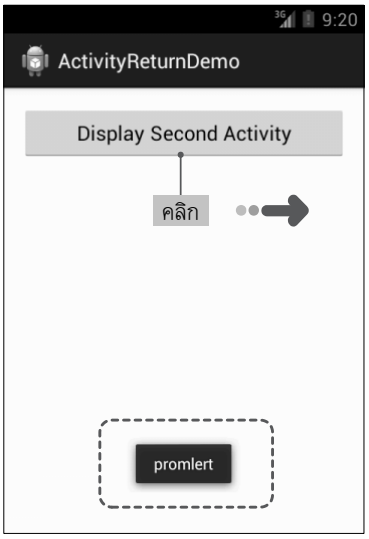
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Please enter your name:" />

    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/ok_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="OK" />

</LinearLayout>
```

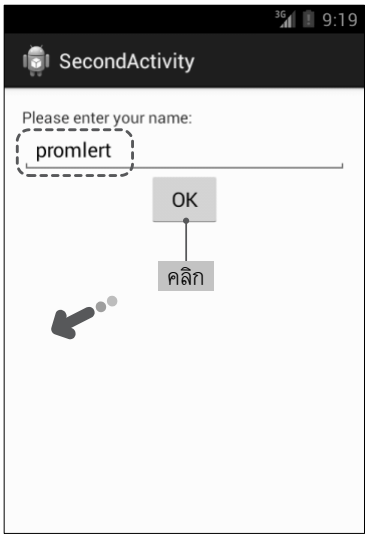
4 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตีที่สอง เพื่อระบุการทำงานของปุ่ม OK

โปรเจ็ค ActivityReturnDemo, ไฟล์ SecondActivity.java

```
Button btnOK = (Button) findViewById(R.id.ok_button);
btnOK.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        EditText etName = (EditText) findViewById(R.id.name);
        String strName = etName.getText().toString();

        Intent i = new Intent();
        i.putExtra("username", strName);

        setResult(RESULT_OK, i);
        finish();
    }
});
```

ผลการรัน





คำอธิบาย

MainActivity.java

```
package com.example.activityreturndemo;

import android.os.Bundle;

public class MainActivity extends Activity {

    static final int REQUEST_CODE = 123;

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

        Button btn = (Button) findViewById(R.id.display_second_activity_button);
        btn.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                Intent i = new Intent(MainActivity.this, SecondActivity.class);
                startActivityForResult(i, 123);
            }
        });
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == 123) {
            if (resultCode == RESULT_OK) {
                String strName = data.getStringExtra("username");
                Toast.makeText(this, strName, Toast.LENGTH_LONG).show();
            }
        }
    }
}
```

SecondActivity.java

```
package com.example.activityreturndemo;

import android.os.Bundle;

public class SecondActivity extends Activity {

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

        Button btnOK = (Button) findViewById(R.id.ok_button);
        btnOK.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                EditText etName = (EditText) findViewById(R.id.name);
                String strName = etName.getText().toString();

                Intent i = new Intent();
                i.putExtra("username", strName);

                setResult(RESULT_OK, i);
                finish();
            }
        });
    }
```

เมื่อคลิกปุ่มในแอคทิวิตีหลัก แอคทิวิตีที่สองจะถูกรันขึ้นมาด้วยโค้ดดังนี้ ❶

```
Intent i = new Intent(MainActivity.this, SecondActivity.class);
startActivityForResult(i, 123);
```

จะเห็นว่าโค้ดเหมือนกับหลายตัวอย่างที่ผ่านมา เพียงแต่เปลี่ยนจากเมธอด `startActivity` มาเป็น `startActivityForResult` เนื่องจากตัวอย่างนี้จะมีการส่งข้อมูลจากแอคทิวิตีปลายทางกลับมาให้

เมธอด `startActivityForResult` ต้องการพารามิเตอร์ 2 ตัว ตัวแรกคืออินเทนต์เช่นเดียวกับเมธอด `startActivity` ส่วนตัวที่ 2 ที่เพิ่มมาคือ Request Code ซึ่งจะระบุเป็นเลขจำนวนเต็มอะไรก็ได้ ในที่นี้ระบุเป็น 123 (ปกติมักกำหนดเป็นค่าคงที่ในคลาส แต่ในที่นี้ต้องการให้เข้าใจง่ายจึงขอระบุเป็นตัวเลขตรงๆ)

ความสำคัญของ Request Code ก็คือ เมื่อแอคทิวิตีปลายทางส่งข้อมูลกลับมา แอนดรอยด์จะเรียกกลับ (callback) มาที่เมธอด `onActivityResult` ในแอคทิวิตีต้นทาง โดยส่งผ่านค่า Request Code นี้มาที่พารามิเตอร์ตัวแรก เพื่อให้เราตรวจสอบได้ว่า callback ในครั้งนั้นสัมพันธ์กับการเรียก `startActivityForResult` ตรงจุดใดในแอคทิวิตีต้นทาง (อาจมีการเรียก `startActivityForResult` หลายครั้ง และแต่ละครั้งอาจรันแอคทิวิตีปลายทางที่แตกต่างกันไป ดังนั้นหากไม่มี Request Code เราก็จะไม่รู้ว่า callback นั้นเป็นผลจากแอคทิวิตีใด)

ในแอคทิวิตีที่สอง เมื่อปุ่ม OK ถูกคลิก เราจะอ่านค่าจาก `EditText` มาใส่ลงในอินเทนต์ ❷

```
EditText etName = (EditText) findViewById(R.id.name);
String strName = etName.getText().toString();

Intent i = new Intent();
i.putExtra("username", strName);
```

จากนั้นส่งอินเทนต์กลับไปยังแอคทิวิตีหลักด้วยเมธอด `setResult` แล้วจบการทำงานของแอคทิวิตีที่สองด้วยเมธอด `finish` ❸

```
setResult(RESULT_OK, i);
finish();
```

พารามิเตอร์ตัวแรกของเมธอด `setResult` คือ Result Code ซึ่งเป็นค่าที่บอกถึงความสำเร็จหรือความล้มเหลวในการทำงานของแอคทิวิตีปลายทาง โดยทั่วไปเราจะกำหนดเป็นค่าคงที่ `RESULT_OK` ถ้าทำงานสำเร็จ หรือค่าคงที่ `RESULT_CANCELED` ถ้าล้มเหลว

เมื่อแอคทิวิตีที่สองส่งผลลัพธ์ (อินเทนต์ที่มีข้อมูลอยู่) กลับมา แอนดรอยด์จะเรียกเมธอด `onActivityResult` ในแอคทิวิตีหลักให้อัตโนมัติ เราจึงเตรียมเมธอดดังกล่าวไว้เพื่ออ่านค่าจากอินเทนต์มาแสดงใน Toast

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
                                Intent data) {
    if (requestCode == 123) {
        if (resultCode == RESULT_OK) {
            String strName = data.getStringExtra("username");
            Toast.makeText(this, strName, Toast.LENGTH_LONG).show();
        }
    }
}
```

จากโค้ด ก่อนอื่นเราตรวจสอบว่า Request Code เป็น 123 และ Result Code เป็น `RESULT_OK` หรือไม่ ถ้าใช่จึงจะอ่านข้อมูลจากอินเทนต์ (ข้อมูลนี้ก็คือชื่อที่พิมพ์ลงใน `EditText` ในแอคทิวิตีที่สอง) มาแสดงใน Toast