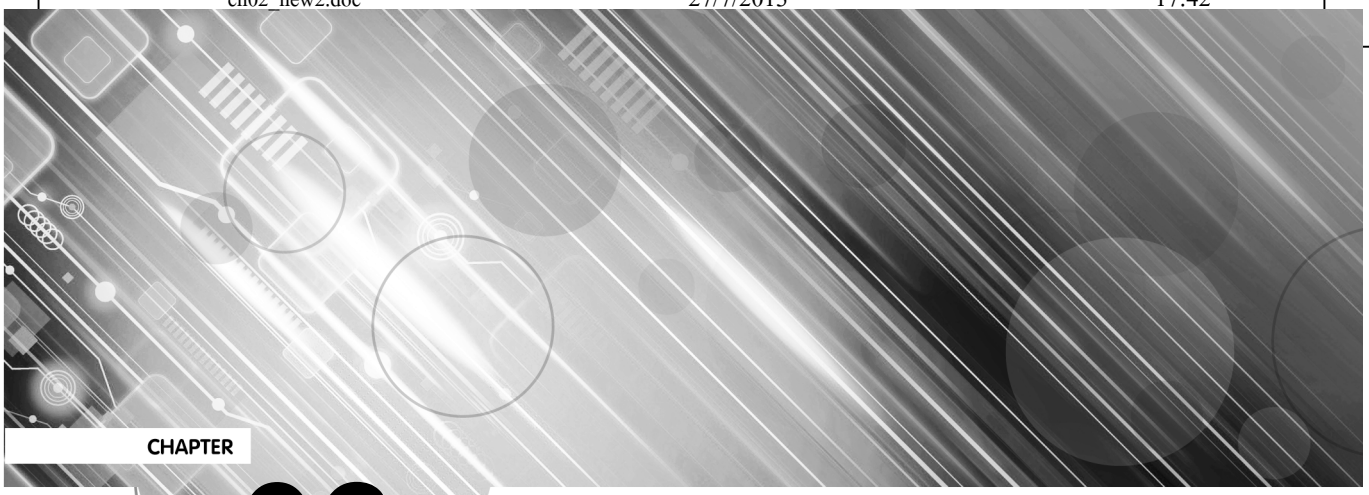


CHAPTER

02 Dialog และ Notification

เนื้อหาในบทนี้

- การแสดงไดอะล็อกอย่างง่าย
- การแสดงไดอะล็อกตัวเลือกแบบ Single-Choice List (วิธีที่ 1)
- การแสดงไดอะล็อกตัวเลือกแบบ Single-Choice List (วิธีที่ 2)
- การแสดงไดอะล็อกตัวเลือกแบบ Multiple-Choice List
- การแสดงไดอะล็อกที่เราออกแบบหน้าตาเอง (Custom Dialog)
- ไม่ให้ปิดไดอะล็อกจนกว่าจะป้อนข้อมูลที่เหมาะสม
- การแสดง Progress Dialog อย่างง่าย
- การแสดง Progress Dialog ที่บอกความคืบหน้า
- การแจ้งเตือนผู้ใช้ด้วย Notification
- การแสดง Notification ที่บอกความคืบหน้า

การแสดงไดอะล็อกอย่างง่าย

งานพื้นฐานอย่างหนึ่งในการเขียนแอพก็คือการแสดงไดอะล็อก (Dialog) เพื่อแจ้งข่าวสารแก่ผู้ใช้ หรือให้ผู้ใช้เลือกว่าจะทำอะไรต่อไป ไดอะล็อกคือหน้าต่างที่มีขนาดไม่เต็มจอ และจะแสดงอยู่บนหน้าจอของแอคทิวิตีที่เรียกมันขึ้นมา

คลาสพื้นฐาน (Base Class) สำหรับไดอะล็อกในแอนดรอยด์คือคลาส `Dialog` อย่างไรก็ตาม ในการแสดงไดอะล็อกทั่วไปเราไม่จำเป็นต้องใช้คลาสนี้ แต่ให้ใช้ซับคลาสของมัน เช่น `AlertDialog` ซึ่งสามารถแสดงไดอะล็อกได้หลายรูปแบบ

ตัวอย่าง

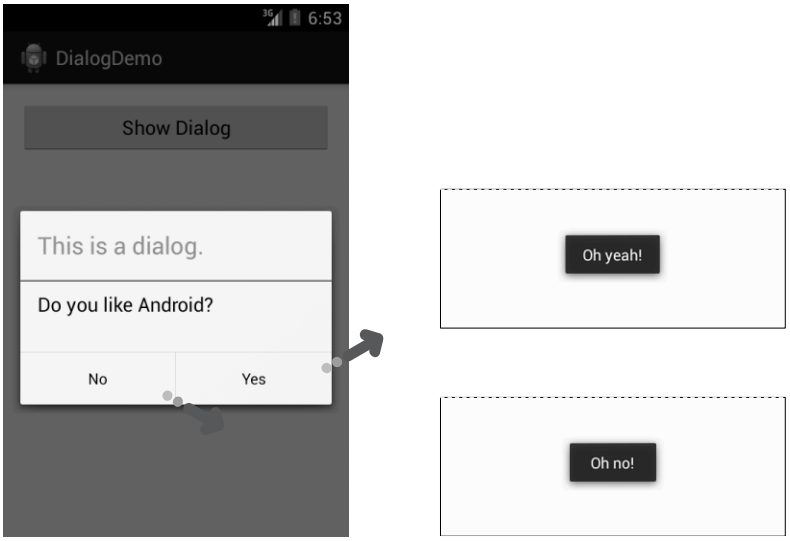
โค้ดต่อไปนี้จะแสดงไดอะล็อกที่มีปุ่ม Yes และ No

โปรเจ็ค DialogDemo, ไฟล์ MainActivity.java

```
AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);

builder.setTitle("This is a dialog.");
builder.setMessage("Do you like Android?");
builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which) {
        showToast("Oh yeah!");
    }
});
builder.setNegativeButton("No", new DialogInterface.OnClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which) {
        showToast("Oh no!");
    }
});

builder.show();
```

ผลการรัน

เมื่อคลิกปุ่ม Yes ของไดอะล็อกจะปรากฏ Toast ว่า “Oh yeah!” แต่ถ้าคลิกปุ่ม No จะปรากฏ Toast ว่า “Oh no!”



คำอธิบาย

การแสดงไดอะล็อกจะเริ่มจากการสร้างออบเจ็คของคลาส `AlertDialog.Builder` โดยระบุคอนเท็กซ์ (`MainActivity.this`) เป็นพารามิเตอร์

```
AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
```

ถัดไปเรากำหนดข้อความ Title และข้อความในไดอะล็อก

```
builder.setTitle("This is a dialog.");
builder.setMessage("Do you like Android?");
```

จากนั้นกำหนดปุ่มที่ต้องการให้มีในไดอะล็อก ในที่นี้กำหนดปุ่มประเภท Positive (ปุ่ม Yes) และปุ่มประเภท Negative (ปุ่ม No)

```
builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which) {
        showToast("Oh yeah!");
    }
});
```

```
builder.setNegativeButton("No", new DialogInterface.OnClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which) {
        showToast("Oh no!");
    }
});
```

ปุ่มในไดอะล็อกมี 3 ประเภท คือ Positive, Negative และ Neutral แอนดรอยด์ไม่ได้กำหนดความหมายของปุ่มทั้งสามไว้ แต่โดยทั่วไปเราจะใช้ปุ่ม Positive สำหรับให้ผู้ใช้ยอมรับหรือทำงานต่อ (เช่น ปุ่ม Yes หรือ OK), ปุ่ม Negative สำหรับปฏิเสธหรือยกเลิกการทำงาน (เช่น ปุ่ม No หรือ Cancel) และปุ่ม Neutral สำหรับกรณีที่ผู้ใช้ไม่ต้องการเลือกข้างใดข้างหนึ่ง แต่ต้องการให้ปิดไดอะล็อกไปเฉยๆ (เช่น ปุ่ม Remind me later หรือแปลให้เข้าใจง่ายๆว่า “เอาไว้คราวหน้าค่อยถามใหม่ละกัน ตอนนี้ขี้เกียจตอบ”)

การกำหนดปุ่มในไดอะล็อกจะใช้เมธอด `set`*`XXX`*`Button` ซึ่งมีพารามิเตอร์ 2 ตัว ตัวแรกคือข้อความบนปุ่ม และตัวที่สองคือ OnClick Listener ซึ่งระบุการทำงานเมื่อปุ่มถูกคลิก

สำหรับเมธอด `showToast` คือเมธอดที่ผู้เขียนสร้างขึ้นเองเพื่อความสะดวกในการแสดง Toast

```
private void showToast(String text) {
    Toast.makeText(this, text, Toast.LENGTH_SHORT).show();
}
```

ยุบโค้ดที่ใช้แสดงไดอะล็อก

เนื่องจากเมธอดต่างๆ ของ `AlertDialog.Builder` เช่น `setTitle`, `setMessage`, `set`*`XXX`*`Button` ฯลฯ จะส่งคืนออบเจ็ค `AlertDialog.Builder` นั้นๆกลับมา เราจึงสามารถเรียกใช้เมธอดเหล่านี้ต่อเนื่องกันไปได้เลย เช่น แทนที่จะเขียนว่า

```
builder.setTitle("title");
builder.setMessage("message");
```

ก็อาจเขียนแบบนี้แทน

```
builder.setTitle("title").setMessage("message");
```

หรือขึ้นบรรทัดใหม่แบบนี้ก็ได้เช่นกัน (สังเกตว่าบรรทัดแรกไม่มี semicolon ปิดท้าย ดังนั้น 2 บรรทัดนี้ถือเป็น 1 ประโยคคำสั่งในภาษาจาวา)

```
builder.setTitle("title")
       .setMessage("message");
```

และสำหรับตัวอย่างนี้ เราไม่จำเป็นต้องประกาศตัวแปร builder ขึ้นมาเก็บออบเจ็ค AlertDialog.Builder ก่อนอีกด้วย แต่สามารถเรียกเมธอดของ AlertDialog.Builder ต่อท้าย new AlertDialog.Builder(MainActivity.this) ได้เลย โค้ดทั้งหมดจึงสามารถเขียนได้ใหม่ว่า

```
new AlertDialog.Builder(MainActivity.this)
    .setTitle("This is a dialog.")
    .setMessage("Do you like Android?")
    .setPositiveButton("Yes",
            new DialogInterface.OnClickListener() {

                @Override
                public void onClick(DialogInterface dialog, int which) {
                    showToast("Oh yeah!");
                }
            })
    .setNegativeButton("No",
            new DialogInterface.OnClickListener() {

                @Override
                public void onClick(DialogInterface dialog, int which) {
                    showToast("Oh no!");
                }
            })
    .show();
```

และตอนนี้โค้ดทั้งหมดได้กลายเป็นประโยคคำสั่งเดียวในภาษาจาวาไปแล้ว (ไม่นับประโยคคำสั่งของ onClick ที่อยู่ใน Inner Class)

การแสดงไดอะล็อกตัวเลือกแบบ Single-Choice List (วิธีที่ 1)

ไดอะล็อกในแอนดรอยด์มีความยืดหยุ่นมาก นอกจากไดอะล็อกแจ้งข่าวสารที่มีปุ่มพื้นฐาน เช่น Yes/No หรือ OK/Cancel แล้ว เรายังสามารถแสดงไดอะล็อกที่เป็นรายการตัวเลือกได้ด้วย ในหัวข้อนี้จะแสดงการสร้างไดอะล็อกตัวเลือกที่ให้ผู้ใช้เลือกได้เพียงตัวเลือกเดียว (single-choice list)

ตัวอย่าง

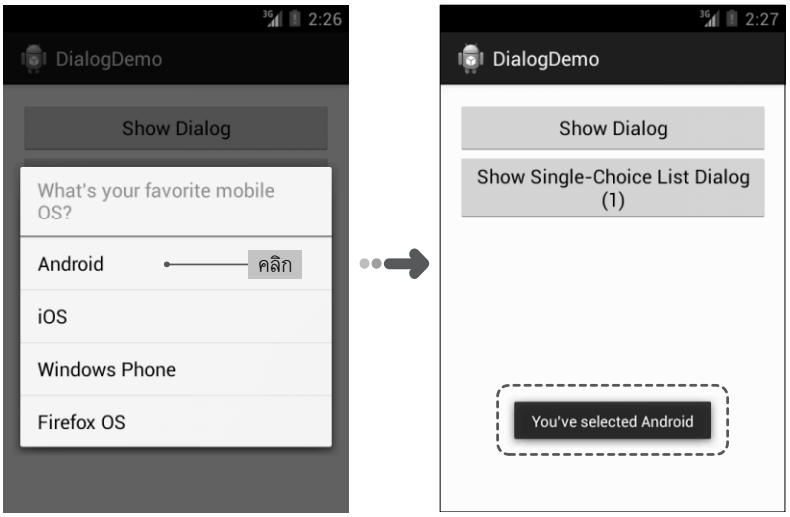
โค้ดต่อไปนี้จะแสดงไดอะล็อกที่มี 4 ตัวเลือก และไม่มีปุ่ม OK สำหรับปิดไดอะล็อก แต่ไดอะล็อกจะปิดไปเองเมื่อผู้ใช้เลือกตัวเลือกใดตัวเลือกหนึ่ง

โปรเจ็ค DialogDemo, ไฟล์ MainActivity.java

```
final String[] items = { "Android", "iOS", "Windows Phone", "Firefox OS" };

new AlertDialog.Builder(MainActivity.this)
```

```
    .setTitle("What's your favorite mobile OS")
    .setItems(items, new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            showToast("You've selected " + items[which]);
        }
    })
    .show();
```

ผลการรัน



คำอธิบาย

การกำหนดรายการตัวเลือกในไดอะล็อกจะใช้เมธอด `setItems` ของ `AlertDialog.Builder`

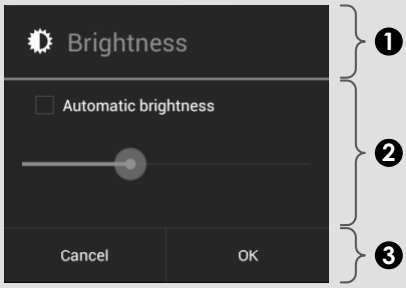
```
.setItems(items, new DialogInterface.OnClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which) {
        showToast("You've selected " + items[which]);
    }
})
```

เมธอด `setItems` ต้องการพารามิเตอร์ 2 ตัว ตัวแรกคืออาร์เรย์ของข้อความตัวเลือก และตัวที่สองคือ OnClick Listener ซึ่งระบุการทำงานเมื่อตัวเลือกในรายการถูกเลือก โดยแอนดรอยด์จะเรียกมายังเมธอด `onClick` ใน Listener นี้ พร้อมทั้งส่งผ่านค่า Index ของตัวเลือกที่ถูกเลือกมายังพารามิเตอร์ตัวที่ 2 (พารามิเตอร์ `which`) เช่น ถ้าเลือกตัวเลือกแรกในไดอะล็อก พารามิเตอร์ `which` ก็จะมีค่า 0

ในที่นี้เมื่อตัวเลือกหนึ่งๆในไดอะล็อกถูกเลือก เราจะนำค่า `which` ไปใช้เป็น Index ในการเข้าถึงอาร์เรย์ `items` เพื่ออ่านข้อความของตัวเลือกนั้นจากอาร์เรย์มาใช้งาน

NOTE

ไดอะล็อกในแอนดรอยด์ประกอบด้วย 3 ส่วนคือ ❶ ส่วนหัวเรื่อง (Title), ❷ ส่วนเนื้อหา (Content area) และ ❸ ส่วนของปุ่ม (Action buttons) ดังรูป



ส่วนของเนื้อหาอาจเป็นข้อความ (กำหนดด้วยเมธอด `setMessage`), รายการตัวเลือก หรือ Layout ที่เราออกแบบเอง อย่างใดอย่างหนึ่งเท่านั้น (ในรูปเป็น Layout ที่ออกแบบเอง) ด้วยเหตุนี้เมื่อแสดงรายการตัวเลือกในไดอะล็อกแล้ว คุณจะไม่สามารถใช้เมธอด `setMessage` แสดงข้อความภายในไดอะล็อกได้อีก คุณจะต้องแสดงคำอธิบายของไดอะล็อกโดยใช้หัวเรื่องแทน (เมธอด `setTitle`)

สำหรับส่วนของปุ่ม แอนดรอยด์อนุญาตให้มีปุ่มแต่ละประเภท (Positive, Negative, Neutral) ได้เพียงอย่างละปุ่ม ดังนั้นไดอะล็อกจะมีปุ่มในส่วนนี้ได้สูงสุด 3 ปุ่มเท่านั้น คือเป็น Positive, Negative และ Neutral อย่างละปุ่มนั่นเอง ถ้าหากคุณต้องการให้มีปุ่มมากกว่านี้จะต้องเตรียมปุ่มไว้ใน Layout ที่ออกแบบเองแล้วนำ Layout นั้นมากำหนดเป็นส่วนเนื้อหาของไดอะล็อก

การแสดงไดอะล็อกตัวเลือกแบบ Single-Choice List (วิธีที่ 2)

วิธีนี้จะแสดงตัวเลือกต่างๆเป็นปุ่มเรดิโอ ผู้ใช้สามารถเลือกได้เพียงตัวเลือกเดียวเช่นกัน แต่ไดอะล็อกจะไม่ปิดทันที ผู้ใช้ต้องคลิกปุ่มในไดอะล็อกจึงจะทำให้ไดอะล็อกถูกปิดไป

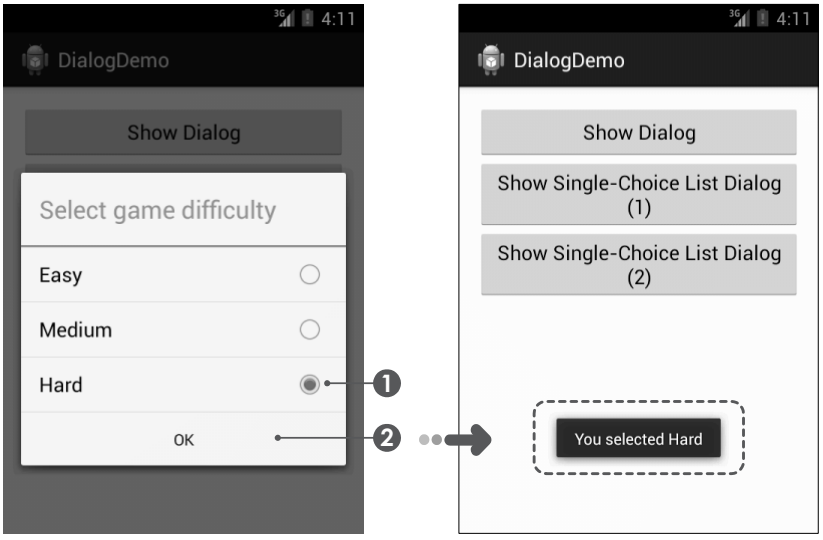
ตัวอย่าง

สมมติเรากำลังพัฒนาเกม และต้องการแสดงไดอะล็อกให้ผู้ใช้เลือกระดับความยากของเกมทุกครั้งที่เริ่มเกมใหม่ โค้ดต่อไปนี้จะแสดงไดอะล็อกดังกล่าว โดยมี 3 ตัวเลือกคือ Easy, Medium และ Hard และมีปุ่ม OK สำหรับปิดไดอะล็อก (ยืนยันการเลือกตัวเลือก และให้ดำเนินการต่อไป ซึ่งในตัวอย่างนี้จะแสดง Toast บอกว่าตัวเลือกใดถูกเลือก)

โปรเจ็ค DialogDemo, ไฟล์ MainActivity.java

```
final String[] items = { "Easy", "Medium", "Hard" };

new AlertDialog.Builder(MainActivity.this)
    .setTitle("Select game difficulty")
    .setSingleChoiceItems(items, 1, null)
    .setPositiveButton("OK", new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            ListView lv = ((AlertDialog) dialog).getListView();
            int selectedItem = lv.getCheckedItemPosition();
            showToast("You selected " + items[selectedItem]);
        }
    })
    .show();
```

ผลการรัน



คำอธิบาย

การแสดงรายการตัวเลือกแบบปุ่มเรดิโอในไดอะล็อก จะใช้เมธอด `setSingleChoiceItems`

```
.setSingleChoiceItems(items, 1, null)
```

พารามิเตอร์ตัวแรกคืออาร์เรย์ของข้อความตัวเลือก พารามิเตอร์ตัวที่สองคือค่าจำนวนเต็มที่กำหนดว่าตัวเลือกใดจะถูกเลือกไว้ตอนเริ่มต้น ในที่นี้ระบุค่า 1 ดังนั้นตัวเลือกที่สองคือ Medium จะถูกเลือกเป็นค่าดีฟอลต์ตอนที่ไดอะล็อกถูกแสดงออกมา แต่หากไม่ต้องการเลือกตัวเลือกใดไว้เลย ให้ระบุค่า -1

TIP

ประโยชน์ของค่าดีฟอลต์นี้ก็เช่น เราอาจจำระดับความยากที่ผู้ใช้เลือกไว้ล่าสุด แล้วกำหนดค่านั้นเป็นดีฟอลต์เมื่อเริ่มเกมใหม่ในครั้งถัดไป

สำหรับพารามิเตอร์ตัวที่สามคือ OnClick Listener เอาไว้ระบุโค้ดการทำงานเมื่อตัวเลือกในรายการถูกคลิก (ถึงแม้จะคลิกตัวเลือกที่ถูกเลือกไว้แล้วก็จะเกิดอีเวนต์เช่นกัน อนึ่ง การคลิกในที่นี้หมายถึงการแตะบนหน้าจอสัมผัส) อย่างไรก็ตาม ตัวอย่างนี้ไม่ได้สนใจตอนคลิกเลือกตัวเลือกต่างๆ แต่จะสนใจตอนคลิกปุ่ม OK เท่านั้น

เรากำหนดปุ่ม OK ของไดอะล็อกด้วยเมธอด `setPositiveButton`

```
.setPositiveButton("OK", new DialogInterface.OnClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which) {
        ListView lv = ((AlertDialog) dialog).getListView();
        int selectedItem = lv.getCheckedItemPosition();
        showToast("You selected " + items[selectedItem]);
    }
})
```

เมื่อปุ่ม OK ถูกคลิก แอนดรอยด์จะเรียกมายังเมธอด `onClick` ที่เราเตรียมไว้ ซึ่งเราเรียกเมธอด `getListView` เพื่อเข้าถึง `ListView` ของไดอะล็อก (รายการตัวเลือกที่เรากำหนดด้วยเมธอด `setSingleChoiceItems` นั้นจริงๆก็คือ `ListView`) จากนั้นเรียกเมธอด `getCheckedItemPosition` เพื่อหา Index ของตัวเลือกที่ถูกเลือก แล้วอ่านข้อความตัวเลือกจากอาร์เรย์ `items` มาแสดงใน Toast

การแสดงไดอะล็อกตัวเลือกแบบ Multiple-Choice List

บางครั้งเราอาจต้องการแสดงไดอะล็อกซึ่งมีรายการตัวเลือกที่อนุญาตให้ผู้ใช้เลือกได้หลายตัวเลือกพร้อมกัน (multiple-choice list) เราสามารถกำหนดรายการตัวเลือกแบบดังกล่าวได้โดยใช้เมธอด `setMultiChoiceItems` ของออบเจ็ค `AlertDialog.Builder`

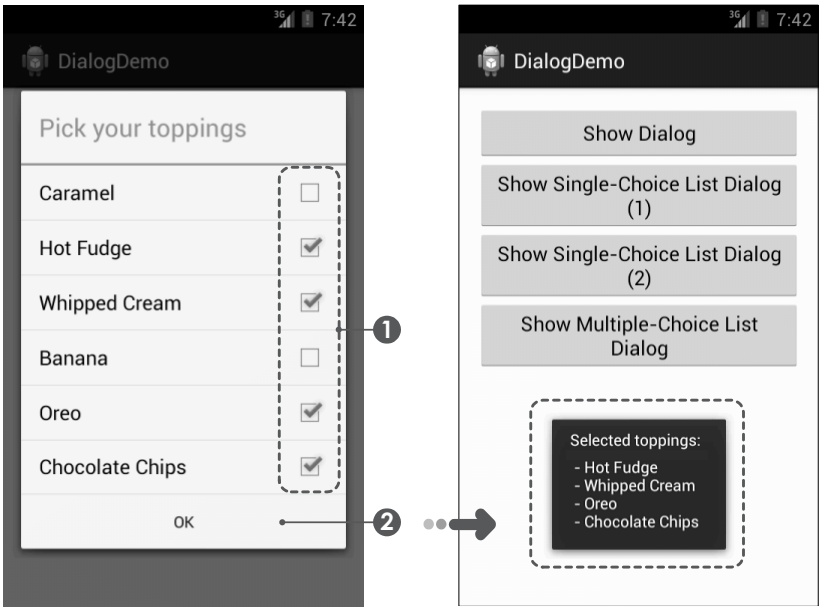
ตัวอย่าง

ไดอะล็อกนี้จะแสดงรายการท็อปปิ้งของไอศครีมให้ผู้ใช้เลือก

โปรเจ็ค DialogDemo, ไฟล์ MainActivity.java

```
final String[] toppings = { "Caramel", "Hot Fudge", "Whipped Cream",
                            "Banana", "Oreo", "Chocolate Chips" };
final ArrayList<Integer> selectedToppings = new ArrayList<Integer>();

new AlertDialog.Builder(MainActivity.this)
    .setTitle("Pick your toppings")
    .setMultiChoiceItems(toppings, null,
                         new DialogInterface.OnMultiChoiceClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which,
                            boolean isChecked) {
            if (isChecked) {
                selectedToppings.add(which);
            } else if (selectedToppings.contains(which)) {
                selectedToppings.remove(Integer.valueOf(which));
            }
        }
    })
    .setPositiveButton("OK", new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            String text = "Selected toppings: \n";
            for (int i : selectedToppings) {
                text += " - " + toppings[i] + "\n";
            }
            showToast(text);
        }
    })
    .show();
```

ผลการรัน



คำอธิบาย

เนื่องจากเราอนุญาตให้ผู้ใช้เลือกท็อปปิ้งไอศครีมได้มากกว่า 1 อย่าง เราจึงกำหนดตัวเลือกท็อปปิ้งเป็นแบบ multiple-choice list โดยใช้เมธอด `setMultiChoiceItems`

```
.setMultiChoiceItems(toppings, null,
                    new DialogInterface.OnMultiChoiceClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which,
                        boolean isChecked) {
        if (isChecked) {
            selectedToppings.add(which);
        } else if (selectedToppings.contains(which)) {
            selectedToppings.remove(Integer.valueOf(which));
        }
    }
})
```

แอนดรอยด์จะเรียกมายังเมธอด `onClick` ข้างต้นเมื่อตัวเลือกต่างๆถูกคลิก ไม่ว่าจะคลิกเพื่อเลือกหรือคลิกเพื่อยกเลิกการเลือกก็ตาม โดยพารามิเตอร์ตัวที่สอง (`which`) จะบอก Index ของตัวเลือกที่ถูกคลิก และพารามิเตอร์ตัวที่สาม (`isChecked`) จะบอกสถานะของตัวเลือกหลังจากถูกคลิกแล้ว กล่าวคือถ้าผู้ใช้คลิกเลือก ค่าของ `isChecked` ก็จะเป็น `true` แต่ถ้าคลิกยกเลิก ค่าของ `isChecked` ก็จะเป็น `false`

เราใช้ `ArrayList` (ตัวแปร `selectedToppings`) ในการเก็บ Index ของตัวเลือกที่ถูกเลือก โดยเมื่อผู้ใช้คลิกเลือกตัวเลือกใด เราจะเพิ่ม Index ของตัวเลือกนั้นเข้าไปใน `ArrayList` แต่ถ้าคลิกยกเลิกก็จะลบค่า Index ของตัวเลือกนั้นออกจาก `ArrayList` ดังนั้นข้อมูลใน `ArrayList` จึงเปลี่ยนแปลงทุกครั้งที่ผู้ใช้คลิกเลือกหรือคลิกยกเลิกตัวเลือกต่างๆ

สุดท้ายเมื่อผู้ใช้คลิก OK เราก็จะวนลูปอ่านข้อมูลจาก `ArrayList` แล้วแสดง Toast ออกมา

```
.setPositiveButton("OK", new DialogInterface.OnClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which) {
        String text = "Selected toppings: \n";
        for (int i : selectedToppings) {
            text += " - " + toppings[i] + "\n";
        }
        showToast(text);
    }
})
```

การแสดงไดอะล็อกที่เราออกแบบหน้าตาเอง (Custom Dialog)

แอนดรอยด์มีความยืดหยุ่นในเรื่องส่วนติดต่อผู้ใช้หรือ UI อย่างมาก ก่อนหน้านี้คุณเห็นแล้วว่าเราสามารถออกแบบหน้าตาของ Toast เองได้ และสำหรับไดอะล็อกก็เช่นเดียวกัน

การแสดงไดอะล็อกที่เราออกแบบหน้าตาเอง (Custom Dialog) นั้น ให้สร้าง Layout ที่ต้องการแล้วนำไปกำหนดให้ไดอะล็อกโดยใช้เมธอด `setView` ของ `AlertDialog.Builder`

ตัวอย่าง

1 ออกแบบ Layout ของไดอะล็อก

โปรเจ็ค DialogDemo, ไฟล์ res\layout\dialog__login.xml

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

```
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="72dp"
        android:background="#fe0000"
        android:scaleType="center"
        android:src="@drawable/header_logo" />

    <EditText
        android:id="@+id/username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="4dp"
        android:layout_marginTop="16dp"
        android:hint="Username"
        android:inputType="textEmailAddress" />

    <EditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="4dp"
        android:layout_marginTop="4dp"
        android:fontFamily="sans-serif"
        android:hint="Password"
        android:inputType="textPassword" />

</LinearLayout>
```

2 เขียนโค้ดในแอคทิวิตี

โปรเจ็ค DialogDemo, ไฟล์ MainActivity.java

```
LayoutInflater inflater = MainActivity.this.getLayoutInflater();

new AlertDialog.Builder(MainActivity.this)
    .setView(inflater.inflate(R.layout.dialog_login, null))
    .setPositiveButton("OK", new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            // ผู้ใช้คลิก OK
        }
    })
```

```
    .setNegativeButton("Cancel", new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            // ผู้ใช้คลิก Cancel
        }
    })
    .show();
```

ผลการรัน

คำอธิบาย

Layout ที่เรากำหนดจะถูกแสดงภายในส่วนเนื้อหา (Content area) ของไดอะล็อก ดังนั้นคุณสามารถเรียกเมธอด `setTitle` เพื่อกำหนด Title และเรียกเมธอด `set`*XXX*`Button` เพื่อกำหนดปุ่มต่างๆ ของไดอะล็อกได้ตามปกติ แต่ในที่นี้ไม่ได้เรียกเมธอด `setTitle` เนื่องจากภายใน Layout ของเรา เราได้เตรียมรูปภาพมาแสดงเป็น Title แล้ว

ไม่ให้ปิดไดอะล็อกจนกว่าจะป้อนข้อมูลที่เหมาะสม

คำถามยอดฮิตข้อหนึ่งสำหรับการเขียนแอพแอนดรอยด์ก็คือ จะยกเลิกการปิดไดอะล็อก (คือให้แสดงไดอะล็อกค้างไว้) ได้อย่างไร หลังจากผู้ใช้คลิกปุ่มใดปุ่มหนึ่งของไดอะล็อกแล้ว และเราตรวจสอบพบว่าข้อมูลที่กรอกในไดอะล็อกไม่ถูกต้อง?

ถ้าดูในเอกสารของแอนดรอยด์ เราจะไม่พบเมธอดหรือวิธีการใดๆที่จะตอบคำถามนี้ได้ แต่ที่จริงแล้วสามารถทำได้ไม่ยากเลย โดย Override การทำงานของปุ่มหลังจากที่ไดอะล็อกแสดงออกมาแล้ว

ตัวอย่าง

เราจะปรับปรุง Custom Dialog ในตัวอย่างที่แล้ว โดยเพิ่มการตรวจสอบชื่อและรหัสผ่านเมื่อผู้ใช้คลิก OK และจะไม่ปิดไดอะล็อกจนกว่าจะกรอกชื่อและรหัสผ่านถูกต้อง หรือคลิก Cancel

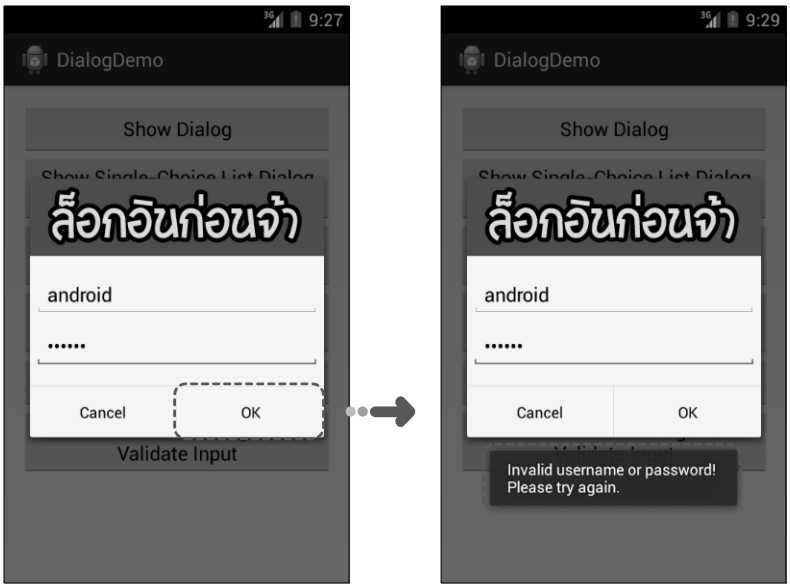
ให้เขียนโค้ดในแอคทิวิตีดังนี้

โปรเจ็ค DialogDemo, ไฟล์ MainActivity.java

```
LayoutInflater inflater = MainActivity.this.getLayoutInflater();

/* ประกาศตัวแปร loginDialog สำหรับเก็บ AlertDialog ที่สร้างขึ้น เนื่องจากโค้ดส่วนนี้จะสร้าง
   ไดอะล็อกแต่ยังไม่แสดงออกมา ต่างจากตัวอย่างอื่นๆก่อนหน้านี้ */
final AlertDialog loginDialog = new AlertDialog.Builder(MainActivity.this)
    .setView(inflater.inflate(R.layout.dialog_login, null))
    .setPositiveButton("OK", new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            // ไม่ต้องทำอะไรตรงนี้ เพราะเรา Override เมธอด onClick แล้ว (ดูข้างล่าง)
        }
    })
    .setNegativeButton("Cancel", new DialogInterface.OnClickListener() {

        @Override
        public void onClick(DialogInterface dialog, int which) {
            // ผู้ใช้คลิก Cancel
        }
    })
    .create(); // สร้างไดอะล็อกขึ้นมาในหน่วยความจำ แต่ยังไม่แสดงออกมา

// กำหนด OnShow Listener เพื่อระบุการทำงานตอนไดอะล็อกแสดงออกมา
loginDialog.setOnShowListener(new DialogInterface.OnShowListener() {

    @Override
    public void onShow(DialogInterface dialog) {
        // อ้างอิงไปยังปุ่ม OK แล้ว Override เมธอด onClick ของปุ่ม
```

```
        Button button = loginDialog.getButton(AlertDialog.BUTTON_POSITIVE);
        button.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // อ้างอิงไปยัง EditText ทั้งสอง (ช่อง Username และ Password)
                EditText etUsername = (EditText)
                        loginDialog.findViewById(R.id.username);
                EditText etPassword = (EditText)
                        loginDialog.findViewById(R.id.password);

                // อ่านค่าจาก EditText ทั้งสองมาเก็บลงตัวแปร
                String strUsername = etUsername.getText().toString();
                String strPassword = etPassword.getText().toString();

                /* ตรวจสอบชื่อและรหัสผ่านว่าถูกต้องหรือไม่ (ตัวอย่างนี้เป็นการสมมติ จึงเก็บชื่อ
                   และรหัสผ่านไว้ตายตัวในโค้ด แต่ในทางปฏิบัติจริงๆอาจเก็บชื่อและรหัสผ่านไว้ใน
                   ฐานข้อมูลหรือเครื่องเซิร์ฟเวอร์ในเน็ตเวิร์ก/อินเทอร์เน็ต) */
                if ((strUsername.equals("promlert")) &&
                        (strPassword.equals("123456"))) {
                    // ถ้าชื่อและรหัสผ่านถูกจะปิดไดอะล็อก และแสดงข้อความต้อนรับใน Toast
                    loginDialog.dismiss();

                    String msg = "Welcome user.";
                    Toast.makeText(MainActivity.this, msg,
                                   Toast.LENGTH_LONG).show();
                }
                else {
                    // ถ้าชื่อหรือรหัสผ่านผิดจะแจ้งให้ผู้ใช้ทราบใน Toast
                    String msg = "Invalid username or password!\n";
                    msg += "Please try again.";
                    Toast.makeText(MainActivity.this, msg,
                                   Toast.LENGTH_LONG).show();
                }
            }
        });
    }
});

loginDialog.show(); // แสดงไดอะล็อก
```

ผลการรัน



การแสดง Progress Dialog อย่างง่าย

เมื่อต้องทำงานที่ใช้เวลานาน เช่น การดาวน์โหลดข้อมูลจากเน็ตเวิร์ก หรือการคำนวณที่ซับซ้อน เราควรแสดงไดอะล็อกที่แจ้งให้ผู้ใช้ทราบถึงการทำงานนั้น เพื่อให้ผู้ใช้รอจนกว่าการทำงานจะจบลง แอนดรอยด์เรียกไดอะล็อกประเภทนี้ว่า Progress Dialog หรือไดอะล็อกแสดงความคืบหน้า อย่างไรก็ตาม รูปแบบที่ง่ายที่สุดของ Progress Dialog จะแสดงเพียงไอคอนที่มีการเคลื่อนไหว โดยไม่ได้บอกความคืบหน้าของงาน (รูปแบบนี้น่าจะเรียกว่า Please-wait Dialog มากกว่า)

ขั้นตอนการแสดง Progress Dialog อย่างง่าย

1 ประกาศตัวแปรเป็นชนิด `ProgressDialog` (มักประกาศไว้ที่ระดับคลาส)

```
ProgressDialog dialog;
```

2 เมื่อต้องการแสดง Progress Dialog ให้เรียกเมธอด `show` (เป็น Static Method) ของคลาส `ProgressDialog` เมธอดนี้จะส่งคืนออบเจ็ค `ProgressDialog` ที่แทนตัวไดอะล็อกกลับมาให้ ดังนั้นให้เก็บลงตัวแปรชนิด `ProgressDialog` ที่ประกาศไว้ข้างต้น เพื่อจะได้ปิดไดอะล็อกในภายหลัง

```
dialog = ProgressDialog.show(คอนเท็กซ์, หัวเรื่องของไดอะล็อก, ข้อความในไดอะล็อก);
```

3 เมื่อต้องการปิด Progress Dialog ให้เรียกเมธอด dismiss

```
dialog.dismiss();
```

ตัวอย่าง

ตัวอย่างนี้จะสร้างปุ่มขึ้นมาใน Layout 1 ปุ่ม สมมติให้เป็นปุ่มสำหรับดาวน์โหลดข้อมูล โดยเมื่อคลิกจะแสดง Progress Dialog และเมื่อ (สมมติว่า) ดาวน์โหลดข้อมูลเสร็จแล้ว Progress Dialog ก็จะถูกปิดลงไป

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค ProgressDialogDemo, ไฟล์ 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/download_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Download" />

</LinearLayout>
```

2 ที่แอคทิวิตี ให้เพิ่มตัวแปรระดับคลาส

โปรเจ็ค ProgressDialogDemo, ไฟล์ MainActivity.java

```
ProgressDialog dialog;
```

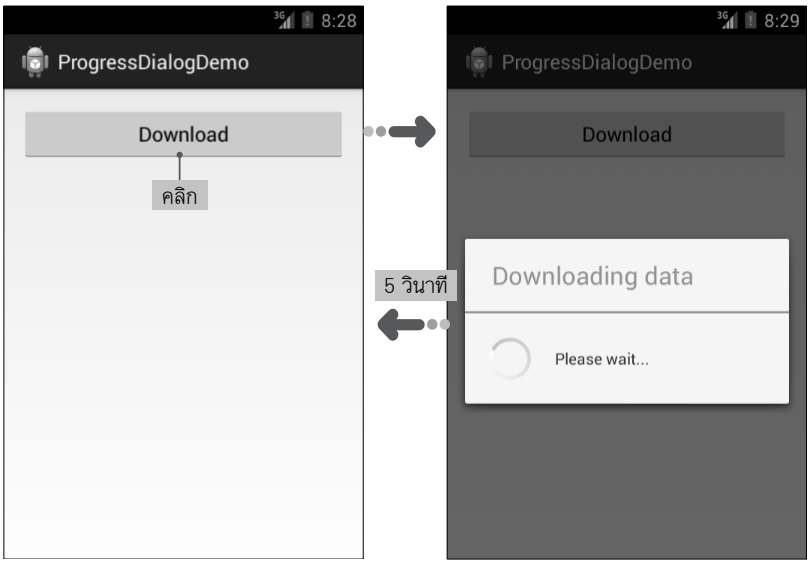
3 เพิ่มโค้ดในเมธอด onCreate

โปรเจ็ค ProgressDialogDemo, ไฟล์ MainActivity.java

```
Button btnDownload = (Button) findViewById(R.id.download_button);
btnDownload.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // แสดง Progress Dialog
        String title = "Downloading data";
        String msg = "Please wait...";
        dialog = ProgressDialog.show(MainActivity.this, title, msg);

        // สร้างเธรดใหม่สำหรับทำงานที่ใช้เวลานาน
        Thread thread = new Thread(new Runnable() {
```

```
            @Override
            public void run() {
                try {
                    /* ใส่โค้ดการทำงานไว้ตรงนี้ ในที่นี้จำลองการทำงานโดยสั่งให้เธรดหยุดพัก
                       5 วินาที */
                    Thread.sleep(5000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                // ปิดไดอะล็อกเมื่อการทำงานสิ้นสุด
                dialog.dismiss();
            }
        });

        thread.start(); // รันเธรด
    }
});
```

ผลการรัน



การแสดง Progress Dialog ที่บอกความคืบหน้า

Progress Dialog ที่บอกความคืบหน้าจะช่วยให้ผู้ใช้เห็นความก้าวหน้าของงานที่แอพทำอยู่ และคาดเดาได้ว่าจะต้องรออีกนานแค่ไหน

การเขียนโค้ดเพื่อแสดง Progress Dialog ในรูปแบบนี้จะคล้ายกับหัวข้อที่แล้ว ความแตกต่างหลักๆ คือ การสร้างออบเจ็ค `ProgressDialog` จะใช้การ Instantiate คลาส `ProgressDialog` ด้วยคีย์เวิร์ด `new` แทนที่จะใช้เมธอด `show` ของคลาส `ProgressDialog` อย่างหัวข้อที่แล้ว

ขั้นตอนการแสดง Progress Dialog ที่บอกความคืบหน้า

1 ประกาศตัวแปรเป็นชนิด `ProgressDialog` (มักประกาศไว้ที่ระดับคลาส)

```
ProgressDialog dialog;
```

2 สร้างออบเจ็ค `ProgressDialog` โดยใช้คีย์เวิร์ด `new` ซึ่ง Progress Dialog จะถูกสร้างขึ้นในหน่วยความจำ แต่ยังไม่แสดงออกมา

```
dialog = new ProgressDialog(คอนเท็กซ์);
```

3 กำหนดคุณสมบัติของ Progress Dialog ตามต้องการ เช่น ไอคอน, หัวเรื่อง, ค่าสูงสุดของตัวเลขความคืบหน้า, ค่าความคืบหน้าเริ่มต้น ฯลฯ

```
dialog.setIcon(...);
dialog.setTitle(...);
dialog.setProgressStyle(...);
dialog.setMax(...);
dialog.setProgress(...);
...
```

4 แสดง Progress Dialog ออกมาโดยเรียกเมธอด `show` (สังเกตว่าเรียกผ่านตัวแปรออบเจ็ค ไม่ได้เรียกจากคลาส `ProgressDialog` โดยตรงเหมือนหัวข้อที่แล้ว)

```
dialog.show();
```

5 เมื่อต้องการอัพเดทความคืบหน้าของงาน ให้เรียกเมธอด `setProgress` หรือ `incrementProgressBy` โดยเมธอดแรกจะกำหนดตัวเลขความคืบหน้าให้เท่ากับค่าที่ระบุ ส่วนเมธอดหลังจะเพิ่มตัวเลขความคืบหน้าขึ้นจากเดิมตามค่าที่ระบุ

```
dialog.setProgress(ค่า);
```

หรือ

```
dialog.incrementProgressBy(ค่า);
```

6 เมื่อต้องการปิด Progress Dialog ให้เรียกเมธอด `dismiss`

```
dialog.dismiss();
```

ตัวอย่าง

เราจะทำต่อจากตัวอย่างที่แล้ว โดยเพิ่มปุ่มอีกปุ่มหนึ่งใน Layout เพื่อแสดง Progress Dialog ที่บอกความคืบหน้า

1 เพิ่มปุ่มใน Layout

โปรเจ็ค ProgressDialogDemo, ไฟล์ activity__main.xml

```
<Button
    android:id="@+id/download_show_progress_button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Download and Show Progress" />
```

2 ที่แอคทิวิตี ให้ประกาศตัวแปร `cancel` เป็นตัวแปรระดับคลาส

โปรเจ็ค ProgressDialogDemo, ไฟล์ MainActivity.java

```
boolean cancel;
```

เราจะเตรียมปุ่ม Cancel ไว้ใน Progress Dialog เพื่อให้ผู้ใช้ยกเลิกการทำงานกลางคันได้ โดยเมื่อผู้ใช้คลิกปุ่มนี้ เราจะกำหนดตัวแปร `cancel` เป็นค่า `true` ซึ่งโค้ดการทำงานของเราจะคอยตรวจสอบค่าตัวแปรนี้อยู่เรื่อยๆ และจะยกเลิกการทำงานทันทีที่พบว่ามันมีค่า `true`

3 เพิ่มโค้ดในเมธอด `onCreate`

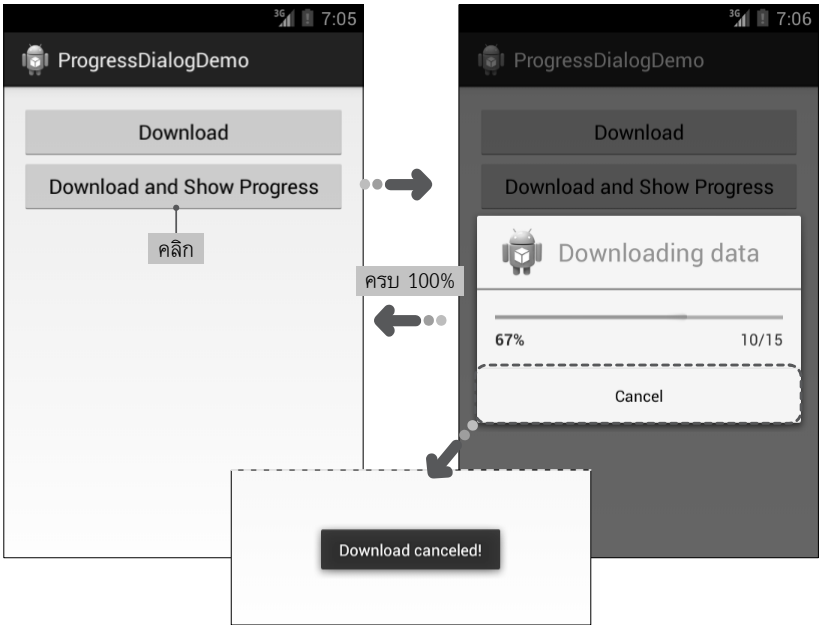
โปรเจ็ค ProgressDialogDemo, ไฟล์ MainActivity.java

```
Button btnDownloadShowProgress = (Button) findViewById(
                                    R.id.download_show_progress_button);
btnDownloadShowProgress.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        cancel = false;  // ตอนเริ่มต้นต้องกำหนด cancel เป็น false ไว้ก่อน

        // สร้าง Progress Dialog และกำหนดคุณสมบัติต่างๆ
        dialog = new ProgressDialog(MainActivity.this);
        dialog.setIcon(R.drawable.ic_launcher);
        dialog.setTitle("Downloading data");
        dialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
        dialog.setMax(15);              // กำหนดค่าสูงสุดของตัวเลขความคืบหน้า
        // กำหนดปุ่ม Cancel และการทำงานเมื่อปุ่มถูกคลิก
        dialog.setButton(DialogInterface.BUTTON_NEGATIVE, "Cancel",
            new DialogInterface.OnClickListener() {
```

```
            // เมื่อปุ่มถูกคลิกจะกำหนดตัวแปร cancel เป็นค่า true และแสดง Toast
            @Override
            public void onClick(DialogInterface dialog, int which) {
                cancel = true;
                Toast.makeText(MainActivity.this, "Download canceled!",
                               Toast.LENGTH_SHORT).show();
            }
        });
    dialog.setCancelable(false);  // ไม่ให้ยกเลิกไดอะล็อกด้วยการกด Back
    dialog.setProgress(0);        // กำหนดค่าความคืบหน้าตอนเริ่มต้น
    dialog.show();                // แสดงไดอะล็อก

    // สร้างเธรดใหม่สำหรับทำงานที่ใช้เวลานาน
    Thread thread = new Thread(new Runnable() {

        @Override
        public void run() {
            // จำลองการทำงานที่ประกอบด้วย 15 ขั้นตอน (เช่น ดาวน์โหลด 15 ไฟล์)
            for (int i = 1; i <= 15; i++) {
                // ถ้าผู้ใช้คลิก Cancel จะออกจากเมธอด run ทันที
                if (cancel == true) {
                    return;
                }

                try {
                    /* ใส่โค้ดการทำงานแต่ละขั้นตอนไว้ตรงนี้ ในที่นี้จำลองการทำงานโดยสั่ง
                       ให้เธรดหยุดพัก 1 วินาที */
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

                // เพิ่มค่าความคืบหน้าขึ้น 1
                dialog.incrementProgressBy(1);

                /* ********************************
                 *  หรือเรียก setProgress แบบนี้ก็ได้เช่นกัน:
                 *  dialog.setProgress(i);
                 * ********************************/

            } // ท้ายคำสั่ง for

            // ปิดไดอะล็อกเมื่อการทำงานสิ้นสุด
            dialog.dismiss();
        }
    });
```

```
        thread.start(); // รันเธรด
    }
});
```

ผลการรัน



การแจ้งเตือนผู้ใช้ด้วย Notification

แอนดรอยด์มีระบบ Notification ไว้ให้เราแจ้งเตือนผู้ใช้เกี่ยวกับเหตุการณ์บางอย่างที่เกิดขึ้นโดยที่แอพของเราไม่จำเป็นต้องแสดงผลอยู่ในขณะนั้น

ผู้ใช้สามารถดูรายการ Notification ทั้งหมดได้ โดยใช้นิ้วปาดแถบสถานะด้านบนลงมา (หรือกรณีของแท็บเล็ตให้แตะที่มุมขวาล่างของจอ) ซึ่งจะปรากฏหน้าแจ้งเตือนหรือ Notification Tray ที่รวบรวมการแจ้งเตือนไว้ และโดยทั่วไปเมื่อคลิกที่การแจ้งเตือนเหล่านี้ก็จะเปิดแอคทิวิตีที่แสดงรายละเอียดของการแจ้งเตือนนั้นๆออกมา

NOTE

Notification เป็นวิธีการแจ้งเตือนผู้ใช้สำหรับคอมโพเนนต์ที่มองไม่เห็น เช่น เซอร์วิส, Broadcast Receiver หรือแอคทิวิตีที่ไม่ได้แสดงผลออกมา แต่ในตัวอย่างนี้จะแสดง Notification เมื่อคลิกปุ่มในหน้าจอ เพื่อเน้นให้เห็นว่าจะเขียนโค้ดเพื่อแสดง Notification ได้อย่างไร

ตัวอย่าง

เราจะสร้างปุ่มไว้ในหน้าจอ ซึ่งเมื่อคลิกจะแสดง Notification ออกมาที่ Notification Tray

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค NotificationDemo, ไฟล์ 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/notify_user_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Notify User" />

</LinearLayout>
```

2 ที่แอคทิวิตี ให้เพิ่มค่าคงที่และตัวแปรระดับคลาส

โปรเจ็ค NotificationDemo, ไฟล์ MainActivity.java

```
static final int NOTIFY_ID = 0;     // กำหนด ID ของ Notification ที่จะแสดงออกมา
NotificationCompat.Builder mNotifyBuilder; // ตัวสร้าง Notification
NotificationManager mNotifyManager; // ตัวจัดการ Notification
```

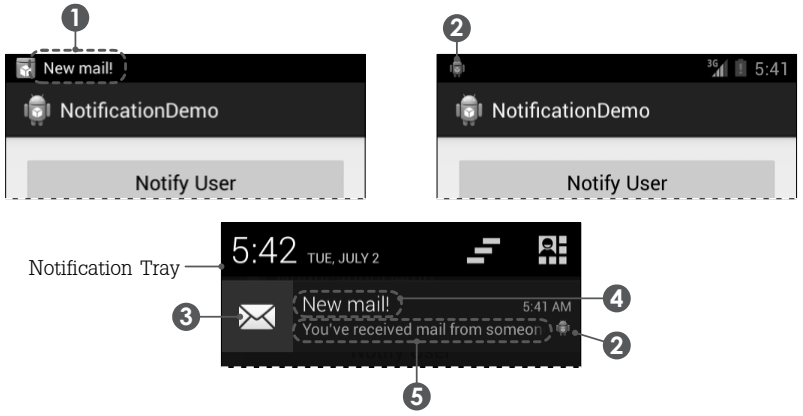
3 เพิ่มโค้ดในเมธอด onCreate

โปรเจ็ค NotificationDemo, ไฟล์ MainActivity.java

```
Button btnNotifyUser = (Button) findViewById(R.id.notify_user_button);
btnNotifyUser.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        /* สร้างอินเทนต์เพื่อระบุการกระทำ (action) เมื่อผู้ใช้คลิก Notification ของเรา ในที่นี้จะ
           ให้รันแอคทิวิตี NotificationActivity ขึ้นมา */
        Intent intent = new Intent(MainActivity.this,
                                   NotificationActivity.class);

        /* สร้าง PendingIntent ขึ้นจากอินเทนต์ข้างต้น เนื่องจากแอพของเราไม่ได้เรียกอินเทนต์นี้เอง
           แต่ตัวจัดการ Notification ของระบบจะเป็นคนเรียก (เพราะว่า Notification จะ
           แสดงอยู่ใน Notification Tray ของระบบ) ซึ่ง PendingIntent เป็นวิธีเตรียม
           อินเทนต์เพื่อให้แอพอื่นมาเรียกใช้ในนามของเรา */
        PendingIntent pIntent = PendingIntent.getActivity(
                                   MainActivity.this, 0, intent, 0);
```

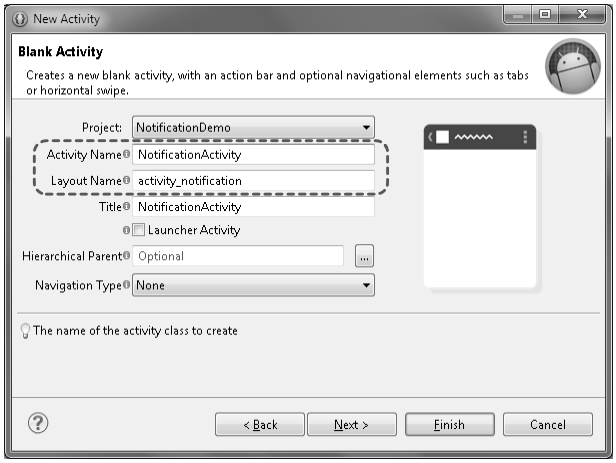
```
        //  เตรียมรูปภาพที่จะแสดงเป็นไอคอนขนาดใหญ่ใน Notification
        Bitmap largeIcon = BitmapFactory.decodeResource(
                getResources(), android.R.drawable.ic_dialog_email);

        // สร้างตัวสร้าง Notification และกำหนดคุณสมบัติของ Notification
        mNotifyBuilder = new NotificationCompat.Builder(MainActivity.this)
                .setTicker("New mail!") ❶
                .setSmallIcon(R.drawable.ic_launcher) ❷
                .setLargeIcon(largeIcon) ❸
                .setContentTitle("New mail!") ❹
                .setContentText("You've received mail from someone ;-)") ❺
                .setContentIntent(pIntent) ❻
                .setAutoCancel(true); ❼

        // สร้าง Notification โดยเรียกเมธอด build ของตัวสร้าง Notification
        Notification notification = mNotifyBuilder.build();

        // เข้าถึงตัวจัดการ Notification ของระบบ
        mNotifyManager = (NotificationManager)
                            getSystemService(NOTIFICATION_SERVICE);

        /* แสดง Notification ออกมา พร้อมทั้งกำหนด ID ของ Notification ซึ่งจะใช้
           ID นี้ในการอัพเดทหรือยกเลิก Notification ในภายหลัง */
        mNotifyManager.notify(NOTIFY_ID, mNotifyBuilder.build());
    }
});
```

เมธอดที่ใช้กำหนดคุณสมบัติของ Notification มีความหมายดังนี้



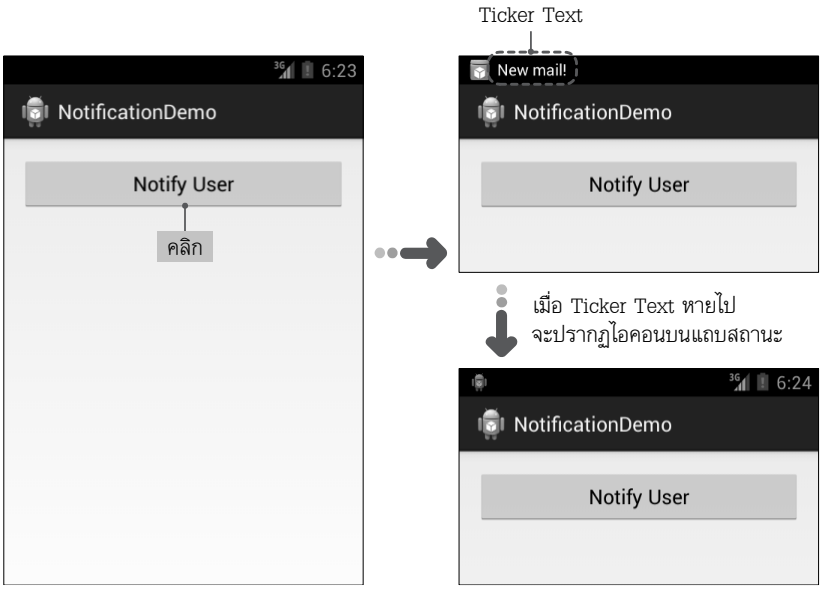
- **setTicker** ❶ กำหนดข้อความ (Ticker Text) ที่แสดงบนแถบสถานะตอนที่ Notification เริ่มปรากฏขึ้นมา
- **setSmallIcon** ❷ กำหนดไอคอนขนาดเล็กบนแถบสถานะ
- **setLargeIcon** ❸ กำหนดไอคอนขนาดใหญ่ใน Notification
- **setContentTitle** ❹ กำหนดข้อความหัวเรื่องของ Notification
- **setContentText** ❺ กำหนดข้อความของ Notification
- **setContentIntent** ❻ กำหนด `PendingIntent` ที่ระบุการกระทำเมื่อ Notification ถูกคลิก
- **setAutoCancel** ❼ กำหนดให้ลบ Notification ออกจาก Notification Tray โดยอัตโนมัติเมื่อถูกคลิก

4 สร้างแอคทิวิตีใหม่ โดยใช้เมนู **File ▸ New ▸ Other ▸ Android ▸ Android Activity**, ตั้งชื่อแอคทิวิตีว่า **NotificationActivity** และตั้งชื่อ Layout สำหรับแอคทิวิตีนี้ว่า **activity_notification**

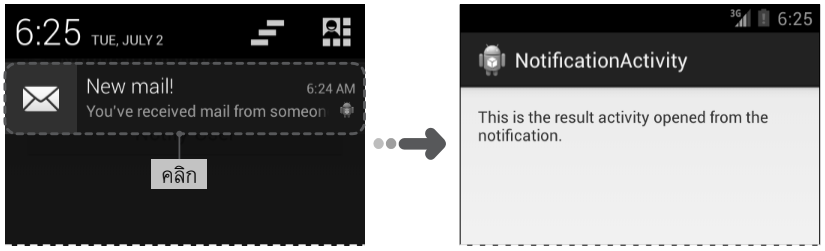


เราจะสมมติให้แอคทิวิตีนี้ทำหน้าที่แสดงรายละเอียดเพิ่มเติมเกี่ยวกับ Notification ของเรา เมื่อ Notification ถูกคลิก

ผลการรัน



เปิด Notification Tray โดยแตะลากแถบสถานะ
ลงมา จะเห็นรายละเอียดของ Notification



การอัพเดทและยกเลิก Notification

เราสามารถอัพเดทข้อมูลใน Notification ที่แสดงออกไปแล้วได้ เช่น ตัวอย่างนี้จะแก้ไขข้อความใน Notification

```
// กำหนดข้อความใหม่ โดยผ่านตัวสร้าง Notification
mNotifyBuilder.setContentText("This text is updated.");
// สร้าง Notification
Notification notification = mNotifyBuilder.build();
// แสดง Notification ออกมา โดยระบุ ID เดิมของ Notification ที่ต้องการอัพเดท
mNotifyManager.notify(NOTIFY_ID, notification);
```

สิ่งสำคัญในการอัพเดท Notification ก็คือ ตอนเรียกเมธอด `notify` ของตัวจัดการ Notification จะต้องระบุ ID เดิมของ Notification ที่เราต้องการอัพเดทข้อมูล

สำหรับการยกเลิก Notification (ลบ Notification ออกจาก Notification Tray) ให้ใช้เมธอด `cancel` ของตัวจัดการ Notification โดยระบุ ID ของ Notification ที่จะยกเลิก เช่น

```
mNotifyManager.cancel(NOTIFY_ID);
```

การแสดง Notification ที่บอกความคืบหน้า

เราสามารถแสดง Notification ที่มี Progress Bar สำหรับบอกความคืบหน้าในการทำงานหนึ่งๆ โดยให้เรียกเมธอด `setProgress` ของตัวสร้าง Notification ก่อนที่จะสร้างและแสดง Notification ออกมา

ตัวอย่าง

ตัวอย่างนี้จะมีปุ่มซึ่งเมื่อคลิกจะแสดง Notification ที่บอกความคืบหน้า และเพื่อเน้นโค้ดที่ใช้แสดงความคืบหน้าใน Notification ตัวอย่างนี้จะไม่ระบุการทำงานเมื่อ Notification ถูกคลิก

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค NotificationProgressDemo, ไฟล์ 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/notify_user_progress_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Notify User with Progress" />

</LinearLayout>
```

2 ที่แอคทิวิตี ให้เพิ่มค่าคงที่และตัวแปรระดับคลาส

โปรเจ็ค NotificationProgressDemo, ไฟล์ MainActivity.java

```
static final int NOTIFY_ID = 0;      // กำหนด ID ของ Notification ที่จะแสดงออกมา
NotificationCompat.Builder mNotifyBuilder; // ตัวสร้าง Notification
NotificationManager mNotifyManager; // ตัวจัดการ Notification
```

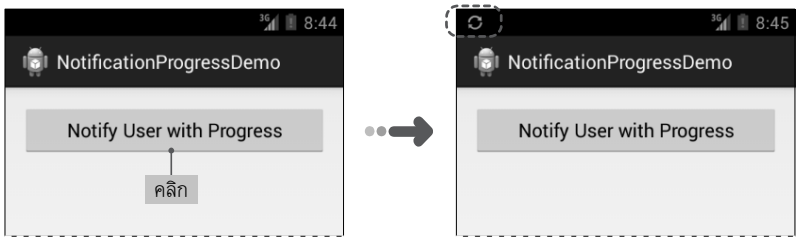
3 เพิ่มโค้ดในเมธอด onCreate

โปรเจ็ค NotificationProgressDemo, ไฟล์ MainActivity.java

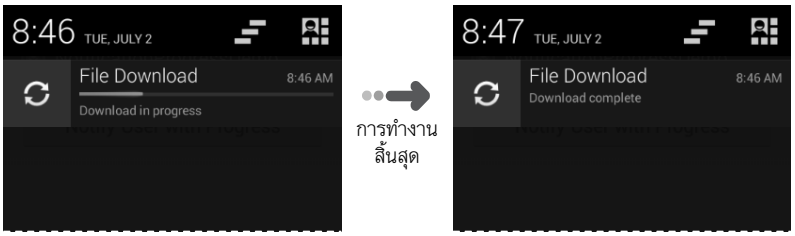
```
Button btnNotifyUser = (Button) findViewById(
                                    R.id.notify_user_progress_button);
btnNotifyUser.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // สร้างตัวสร้าง Notification และกำหนดคุณสมบัติของ Notification
        mNotifyBuilder = new NotificationCompat.Builder(MainActivity.this)
                .setContentTitle("File Download")
                .setContentText("Download in progress")
                .setSmallIcon(android.R.drawable.ic_popup_sync);

        // เข้าถึงตัวจัดการ Notification ของระบบ
        mNotifyManager = (NotificationManager)
                            getSystemService(NOTIFICATION_SERVICE);

        // สร้างเธรดใหม่สำหรับทำงานที่ใช้เวลานาน
        Thread thread = new Thread(new Runnable() {

            @Override
            public void run() {
                // จำลองการทำงานที่ใช้เวลานาน
                for (int i = 1; i <= 20; i++) {
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }

                    // ระบุค่าสูงสุดของความคืบหน้า (20) และความคืบหน้าขณะนี้ (i)
                    mNotifyBuilder.setProgress(20, i, false);
                    // สร้างและแสดง Notification
                    Notification notification = mNotifyBuilder.build();
                    mNotifyManager.notify(NOTIFY_ID, notification);
                }

                /* อัพเดท Notification และยกเลิกการแสดง Progress Bar เมื่อการทำงาน
                   สิ้นสุด */
                mNotifyBuilder.setContentText("Download complete")
                              .setProgress(0, 0, false);
                // สร้างและแสดง Notification
                Notification notification = mNotifyBuilder.build();
```

```
                mNotifyManager.notify(NOTIFY_ID, notification);
            }
        });

        thread.start(); // รันเธรด
    }
});
```

ผลการรัน



เปิด Notification Tray โดยแตะลากแถบสถานะลงมา



การแสดง Notification ที่บอกว่ากำลังดำเนินการอยู่

สำหรับการทำงานที่เราไม่สามารถแสดงปริมาณงานที่ทำไปแล้วเมื่อเทียบกับปริมาณงานทั้งหมดได้ ก็ให้บอกผู้ใช้ว่ากำลังดำเนินการอยู่ (Ongoing) โดยเรียกเมธอด `setProgress` และระบุค่า `true` เป็นพารามิเตอร์ตัวสุดท้าย เช่น

```
mNotifyBuilder.setProgress(0, 0, true);
```

จะได้ Notification ดังรูป