이벤트 처리하기

학습 목표

- 안드로이드의 이벤트 처리를 이해한다.
- 버튼의 이벤트 처리에 대해 익힌다.
- 컴파운드 버튼의 종류와 이벤트 처리에 대해 익힌다.

목차

01

이벤트 처리 살펴보기

1. 이벤트 처리 살펴보기

이벤트 리스너와 이벤트 핸들러

- 이벤트 리스너(Event Listener) : View 클래스에 속한 인터페이스, 단일 콜백 메서드 가짐
- 이벤트 핸들러(Event Handler) : 실제로 수행될 동작을 포함하고 있는 메서드
- 이벤트에 대한 이벤트 리스너를 등록하면 해당 이벤트가 발생할 때마다 이벤트 리스너가 이벤트를 처리하는 메서드인 이벤트 핸들러 호출

표 5-1 주요 이벤트 리스너와 이벤트 핸들러

이벤트 리스너	이벤트 핸들러	설명
OnClickListener()	onClick()	화면의 항목을 클릭하거나 터치할 때, 트랙볼 또는 탐색 키로 항목에 포커스를 맞출 때 사용합니다.
OnLongClickListener()	onLongClick()	항목 또는 화면을 1초 이상 클릭할 때 사용합니다.
OnFocusChangeListener()	onFocusChange()	포커스가 맞춰진 항목에서 다른 곳으로 이동할 때 사용합니다.
OnKeyListener()	onKey()	항목에 포커스를 맞추고 클릭할 때 사용합니다.
OnTouchListener()	onTouch()	항목의 특정 범위를 제스처나 탭으로 터치할 때 사용합니다.

1. 이벤트 처리 살펴보기

이벤트 등록

- 이벤트가 발생했을 때 이벤트 핸들러를 호출하도록 이벤트 리스너에 이벤트 핸들러를 등록하는 과정
- 안드로이드 앱에서 이벤트를 등록하는 방법
  - 레이아웃 파일에 이벤트 핸들러를 직접 지정합니다.
  - 익명 클래스로 이벤트를 처리합니다.
  - 이벤트 리스너를 구현하는 Activity 클래스를 사용합니다.

02

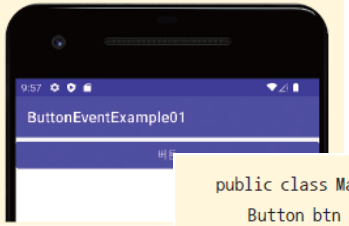
버튼 이벤트 처리하기

2. 버튼 이벤트 처리하기

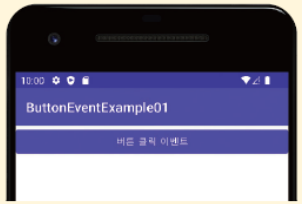
onClick 속성으로 이벤트 핸들러 등록

- 레이아웃 파일에 있는 위젯의 onClick 속성에 이벤트 처리 메서드를 직접 만드는 것
- onClick 속성으로 이벤트를 등록해 처리하는 방법
  - 레이아웃 파일에서 사용자와 상호작용할 위젯에 onClick 속성을 추가. 속성값에는 클릭 이벤트를 처리할 메서드 이름을 설정.
  - 레이아웃 파일과 연동하는 액티비티 클래스(소스 코드)에 이벤트 처리 메서드를 추가. 이때 메서드 이름을 레이아웃 파일에 설정한 onClick 속성값과 같아야 함.

```
<Button
    android:id="@+id/button_id"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="onClickButton" ❶
    android:text="버튼" />
```




```
public class MainActivity extends AppCompatActivity {
    Button btn ;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn = findViewById(R.id.button_id);
    }

    public void onClickButton(View view){ ❷
        btn.setText("버튼 클릭 이벤트");
    }
}
```



2. 버튼 이벤트 처리하기

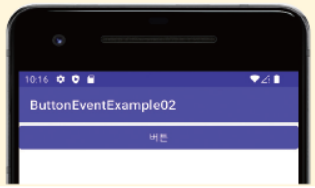
- onClick 속성을 이용해 이벤트를 처리하는 방법
  - 이벤트 개수가 적을 때
  - 간단한 클릭 이벤트 코드를 작성할 때
  - 이벤트 리스너를 별도로 작성하는 게 번거로울 때
  - 가장 간단하고 직관적인 방법을 선호할 때

2. 버튼 이벤트 처리하기

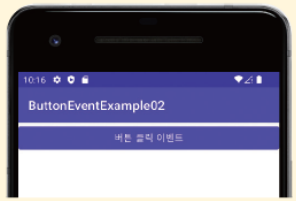
익명 클래스로 이벤트를 처리하는 이벤트 리스너 사용

- 익명 클래스를 사용하면 이벤트 처리 코드가 간결하고, 어디서 이벤트가 처리되는지 쉽게 확인할 수 있어서 자주 사용
- 익명 클래스(Anonymous Class)란 클래스 몸체는 정의되어 있지만 이름이 없는 클래스를 의미하므로 MainActivity와 같은 이름이 없음
- 익명 클래스로 이벤트를 처리하는 방법
  - 레이아웃 파일에 사용자와 상호작용할 위젯의 id 속성값을 추가
  - 레이아웃 파일에 추가한 위젯의 id 속성값을 액티비티 클래스로 가져옴. 이때 위젯의 id 속성값은 findViewById( ) 메서드를 호출해 가져와 객체 변수에 저장.
  - 액티비티 클래스에 setOnClickListener( ) 메서드로 익명 클래스 View.OnClickListener를 생성

```
<Button
        android:id="@+id/button_id" ❶
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="버튼" />
```



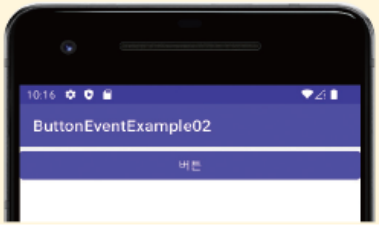

```
public class MainActivity extends AppCompatActivity {
    Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btn = findViewById(R.id.button_id); ❷
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                btn.setText("버튼 클릭 이벤트");
            }
        }); ❸
    }
}
```



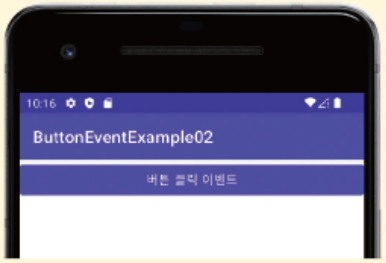
2. 버튼 이벤트 처리하기

■ 익명 클래스로 이벤트를 처리하는 이벤트 리스너 사용

```
<Button
        android:id="@+id/button_id" ❶
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="버튼" />
```




```
public class MainActivity extends AppCompatActivity {
    Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btn = findViewById(R.id.button_id); ❷
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
              public void onClick(View view) {
                  btn.setText("버튼 클릭 이벤트");
              }
         }); ❸
    }
}
```



2. 버튼 이벤트 처리하기

- 익명 클래스로 이벤트를 처리하는 방법
  - 이벤트가 일어나는 뷰의 개수가 적거나 이벤트가 일어나는 뷰 사이에 연관성이 적을 때
  - 이벤트 핸들러에서 익명 클래스 외부의 변수를 참조하지 않을 때
  - 간단한 뷰의 클릭 이벤트 테스트 코드를 작성할 때

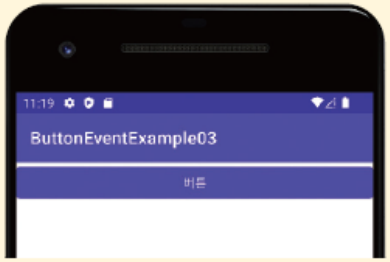
2. 버튼 이벤트 처리하기

이벤트 리스너 구현

- View 클래스에 있는 이벤트 리스너를 액티비티 클래스에서 상속받아 직접 구현하는 방법
  - 레이아웃 파일에 사용자와 상호작용할 위젯의 id 속성값을 추가
  - 액티비티 클래스에서 View 클래스에 있는 이벤트 리스너를 상속받음. Activity 클래스에 View.OnClickListener 인터페이스를 선언
  - 레이아웃 파일에 추가한 위젯의 id 속성값을 findViewById( ) 메서드를 호출해 액티비티 클래스로 가져오고 변수에 저장
  - 액티비티 클래스에서 이벤트 리스너 클래스를 참조(상속)하기 위해 setOnClick Listener( ) 메서드에 this로 이벤트를 전달
  - 액티비티에서 onClick( ) 메서드를 호출. 즉, 액티비티 클래스가 View.OnClick Listener를 상속하므로 액티비티 클래스에 onClick( ) 메서드를 작성

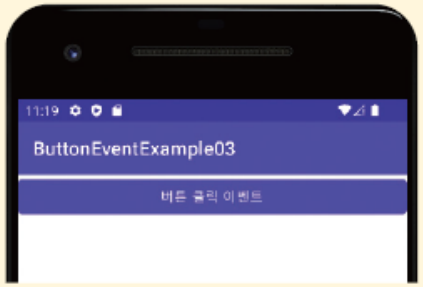
이벤트 리스너 구현

```
<Button
    android:id="@+id/button_id" ❶
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="버튼"
    />
```




```
public class MainActivity extends AppCompatActivity implements View.OnClickListener{
------------ ❷
   @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btn = (Button) findViewById(R.id.button_id); ❸
        btn.setOnClickListener(this); ❹
    }
   @Override
   public void onClick(View v) {
     btn.setText("버튼 클릭 이벤트");
   }
}
```

❺



2. 버튼 이벤트 처리하기

- 이벤트 리스너를 구현해 이벤트를 처리하는 방법
  - 이벤트 핸들러에서 많은 액티비티를 액세스해야 할 때
  - 액티비티 내부에 있는 뷰에 대한 클릭 이벤트를 한 곳에서 처리하고 싶을 때
  - 익명 클래스 또는 별도의 이벤트 리스너를 만들고 싶지 않을 때

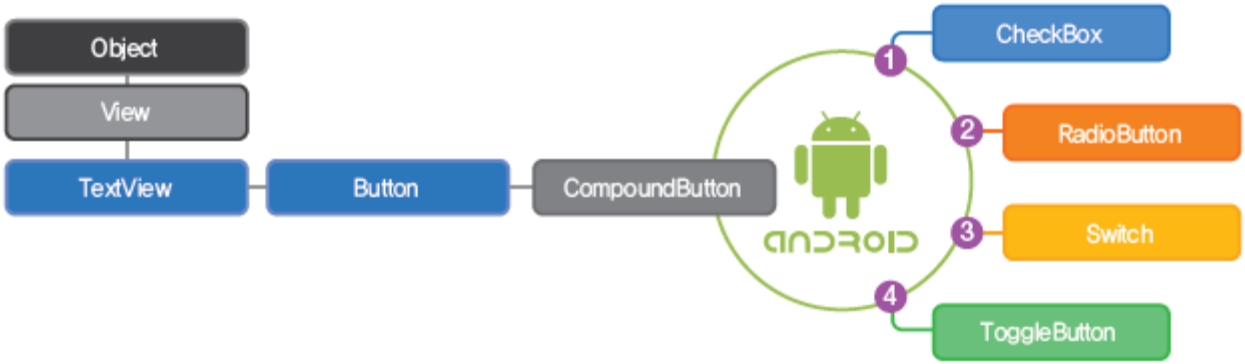
2. 버튼 이벤트 처리하기

- **[실습 예제 5-1] 버튼 클릭 이벤트로 맞는 그림 찾기**
  - 구현 내용 : 화면에 보이는 그림에 맞는 짝을 찾아 클릭했을 때 짝을 찾으면 '맞았습니다', 짝이 아니면 '틀렸습니다'를 화면에 출력합니다.

03
컴파운드 버튼 이벤트 처리 하기

3. 컴파운드 버튼 이벤트 처리하기

그림 5-1 컴파운드 버튼 구성



❶ 체크박스 ❷ 라디오 버튼 ❸ 스위치 ❹ 토글 버튼

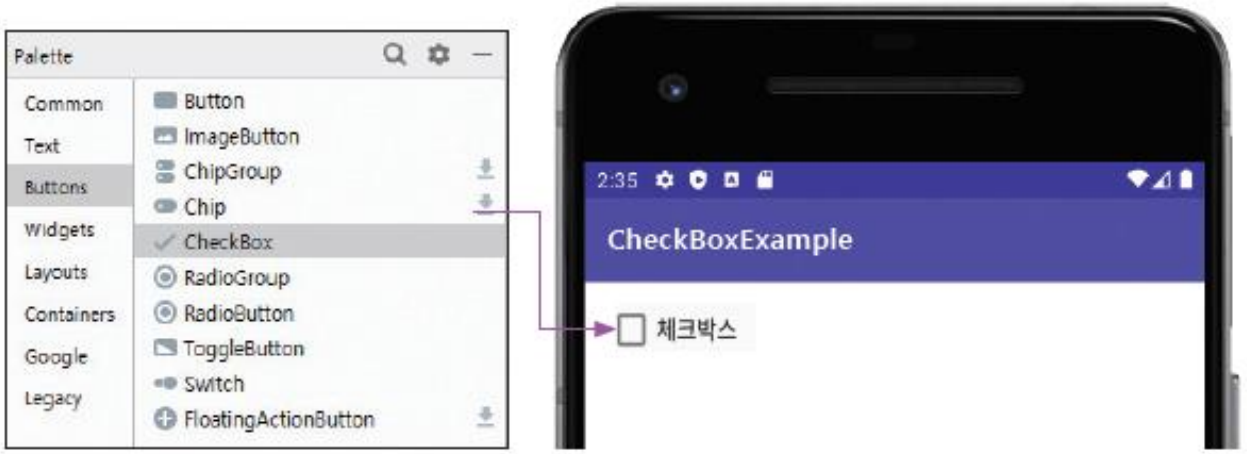
```
android:checked="true" // 기본값 false
```

3. 컴파운드 버튼 이벤트 처리하기

체크박스 이벤트 처리

- 두 가지 상태(ON/OFF)를 갖는 위젯
- 사용자는 요구사항에 따라 상태 전환
- 상호 배타적이지 않은 옵션들을 나타낼 때 사용. 예를 들어, 설문 조사에서 몇 가지 항목을 체크박스로 나열하고 선택하거나 로그인 화면에서 로그인 정보를 기억할지 말지 선택

그림 5-2 체크박스의 구현 예



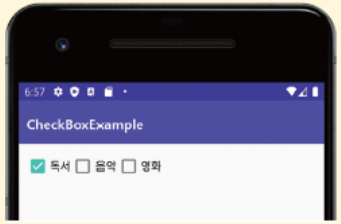
3. 컴파운드 버튼 이벤트 처리하기

■ 체크박스 이벤트 처리

표 5-3 체크박스의 속성 메서드

메서드	설명
boolean isChecked()	체크박스의 현재 상태를 가져옵니다.
void setChecked(boolean status)	체크박스의 상태를 설정합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout ...
    android:orientation="horizontal">

    <CheckBox
        android:id="@+id/checkBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="독서" />
```




```
    <CheckBox
        android:id="@+id/checkBox2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="음악" />

    <CheckBox
        android:id="@+id/checkBox3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="영화" />
</LinearLayout>
```

3. 컴파운드 버튼 이벤트 처리하기

[실습 예제 5-2] 체크박스 상태 가져오기

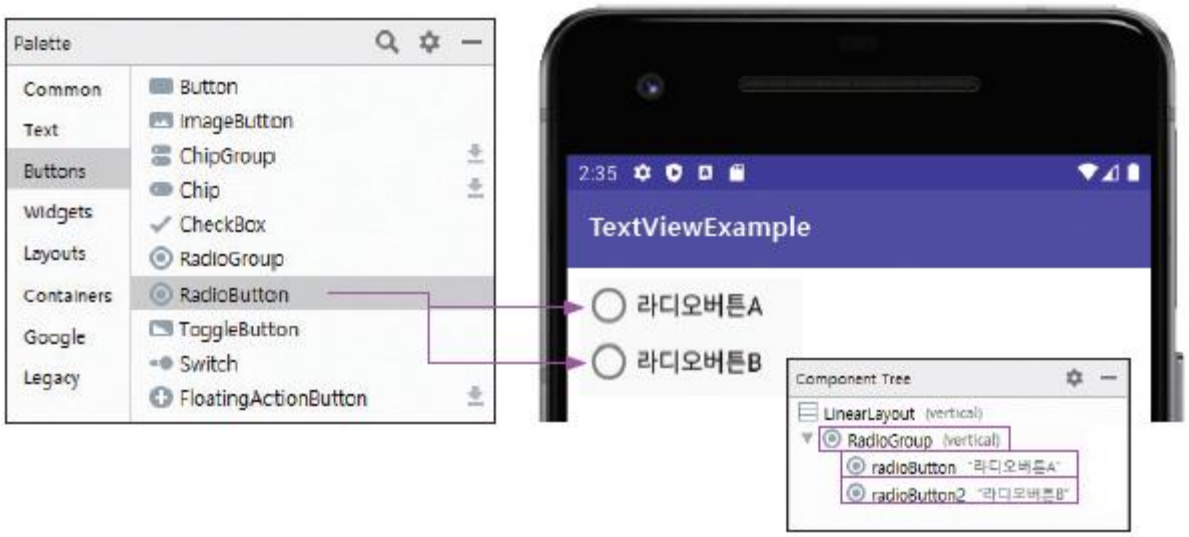
- 구현 내용 : 체크박스를 선택 또는 해제 상태로 설정하고 <결과보기> 버튼을 클릭하면 선택된 체크박스 상태를 가져와 토스트에 출력합니다.

3. 컴파운드 버튼 이벤트 처리하기

라디오 버튼 이벤트 처리

- 라디오 버튼(RadioButton)도 선택과 선택 해제의 두 가지 중 한 가지 상태를 나타내는 위젯
- 보통 라디오 버튼을 여러 개 묶어 라디오 그룹(RadioGroup)으로 그룹화해서 사용
- 라디오 버튼은 상호 배타적이어서 하나의 라디오 버튼을 선택하면 나머지 라디오 버튼은 모두 자동으로 선택이 해제. 즉, 동일한 라디오 그룹에 속한 라디오 버튼에서는 하나만 선택

그림 5-3 라디오 버튼의 구현 예



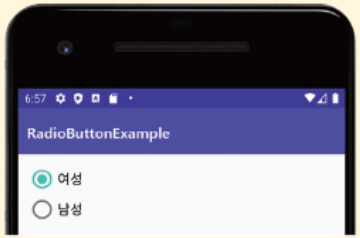
3. 컴파운드 버튼 이벤트 처리하기

■ 라디오 버튼 이벤트 처리

표 5-4 라디오 버튼의 속성 메서드

메서드	설명
boolean isChecked()	라디오 버튼의 현재 상태를 가져옵니다.
void setChecked(boolean status)	라디오 버튼의 상태를 설정합니다.

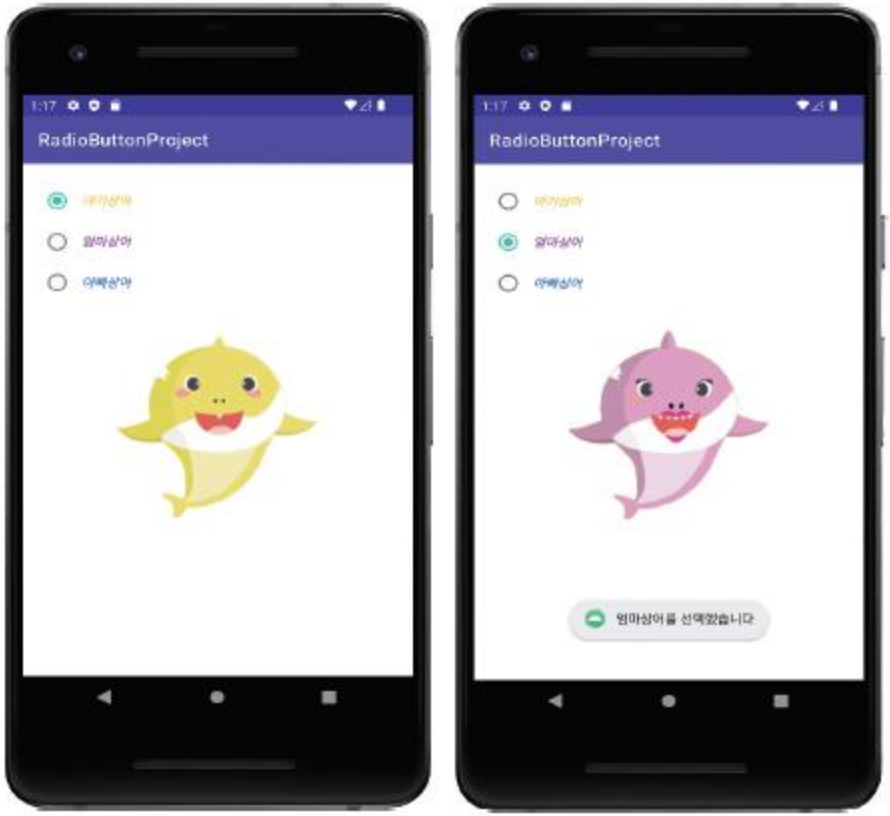
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout ...
    android:orientation="horizontal"
    tools:context=".MainActivity">

    <RadioGroup
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <RadioButton
            android:id="@+id/radioButton1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:checked="true"
            android:text="여성"
            />
```




```
        <RadioButton
            android:id="@+id/radioButton2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="남성"
            />

    </RadioGroup>

</LinearLayout>
```

3. 컴파운드 버튼 이벤트 처리하기

- **[실습 예제 5-3] 라디오 버튼의 상태 가져오기**
  - 구현 내용 : 라디오 버튼을 클릭하면 선택된 라디오 버튼의 상태를 가져와 토스트에 출력하고 해당 이미지를 출력합니다.

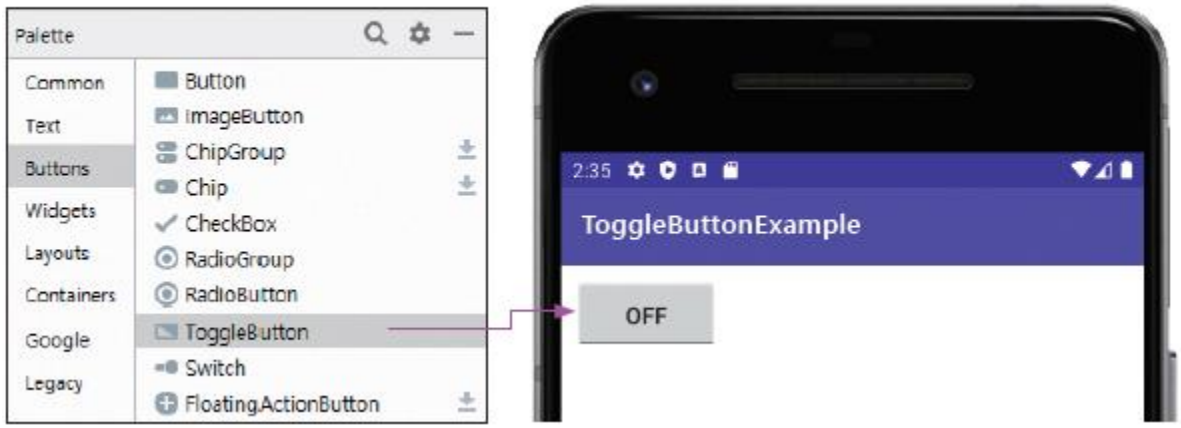


3. 컴파운드 버튼 이벤트 처리하기

토글 버튼 이벤트 처리

- 토글 버튼(ToggleButton) : 선택과 선택 해제 중 하나의 상태만 표시하는 위젯
  두 상태 중 하나만 표시하는 경우에 사용
  일반적으로 사운드, Wi-Fi, 블루투스 등을 켜거나 끄는 버튼으로 많이 사용

그림 5-4 토글 버튼의 구현 예



3. 컴파운드 버튼 이벤트 처리하기

표 5-5 토글 버튼의 속성

속성	설명
disabledAlpha	사용할 수 없을 때 버튼에 적용할 투명도입니다.
textOff	선택 해제(OFF) 상태일 때 버튼의 텍스트를 표시합니다.
textOn	선택(ON) 상태일 때 버튼의 텍스트를 표시합니다.

표 5-6 토글 버튼의 메서드

속성	설명
CharSequence getTextOff()	버튼이 선택 해제(OFF) 상태일 때의 텍스트를 돌려줍니다.
CharSequence getTextOn()	버튼이 선택(ON) 상태일 때의 텍스트를 돌려줍니다.
void setChecked(boolean checked)	버튼의 상태를 변경합니다.

토글 버튼 이벤트 처리

- textOn/textOf 속성

```
<ToggleButton
    android:id="@+id/toggleButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:text="ToggleButton"
    android:textOff="NO"
    android:textOn="YES" />
```

3. 컴파운드 버튼 이벤트 처리하기

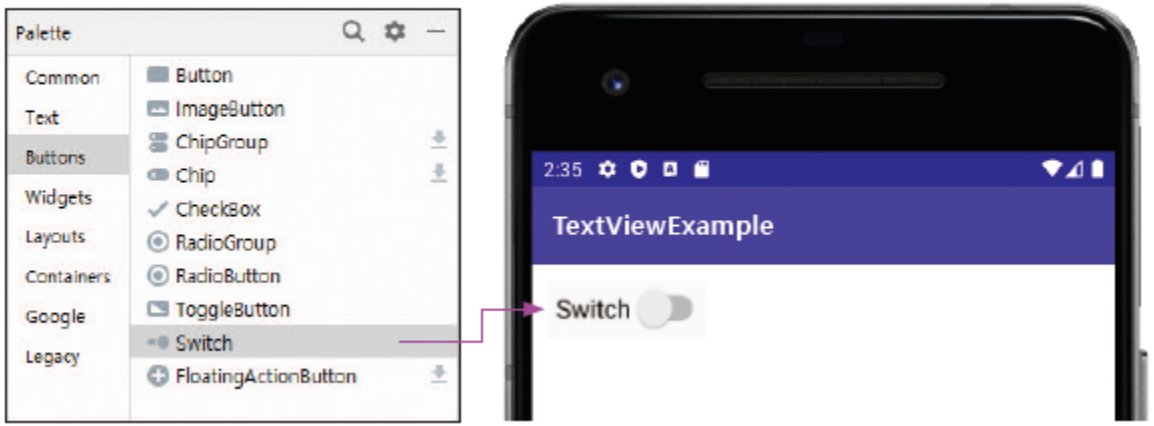
- **[실습 예제 5-4] 두 개의 전구 점등하기**
  - 구현 내용 : on/off를 선택하면 각각의 전구 불이 켜지거나 꺼지게 합니다.

3. 컴파운드 버튼 이벤트 처리하기

스위치 이벤트 처리

- 스위치(Switch) :토글 버튼의 새로운 형태로, 섬(thumb)이라고 하는 둥근 부분을 바 모양 트랙을 따라 앞뒤로 움직여(슬라이드) 상태를 선택
- 일반적으로 전등 스위치처럼 현재상태를 켜거나 끌 수 있음
- 또한 사운드, 블루투스, WiFi 등에서 On/Off를 선택할 때 토글 버튼과 마찬가지로 사용

그림 5-5 스위치의 구현 예



3. 컴파운드 버튼 이벤트 처리하기

■ 스위치 이벤트 처리

표 5-7 스위치의 속성

속성	설명
showText	섬에 텍스트를 표시할지 말지를 나타내는 속성입니다.
splitTrack	트랙을 분할하고 섬 이미지를 위한 간격을 둘지의 여부입니다.
switchMinWidth	스위치의 최소 너비입니다.
switchPadding	스위치와 스위치 캡션(텍스트) 사이의 최소 공간입니다.
switchTextAppearance	섬에 표시되는 텍스트 스타일을 설정합니다.
textOff	스위치가 선택 해제 또는 꺼짐 상태일 때 사용할 텍스트입니다.
textOn	스위치가 선택 또는 켜짐 상태일 때 사용합니다.
thumb	섬에 사용할 이미지를 설정합니다.
thumbTextPadding	섬에 표시되는 텍스트의 양쪽 패딩을 설정합니다.
thumbTint	섬의 색상을 정합니다.
thumbTintMode	섬의 색상을 적용하는 데 사용되는 혼합 모드입니다.
track	트랙으로 사용할 이미지를 설정합니다.

3. 컴파운드 버튼 이벤트 처리하기

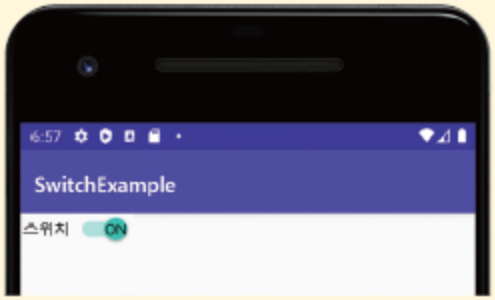
- **스위치 이벤트 처리**
  - showText 속성
  - switchMinWidth 속성
  - textOn/testOff 속성
  - thumb 속성
  - thumbTint 속성

3. 컴파운드 버튼 이벤트 처리하기

■ 스위치 이벤트 처리

- showText 속성

```
<Switch
    android:id="@+id/switch_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:showText="true"
    android:text="스위치" />
```

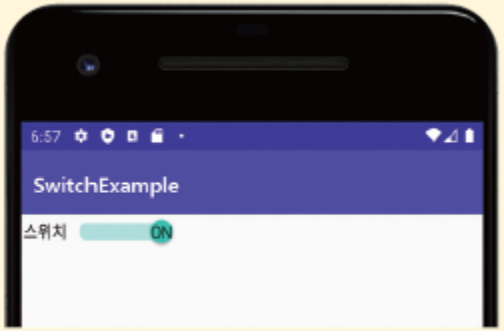


3. 컴파운드 버튼 이벤트 처리하기

스위치 이벤트 처리

- switchMinWidth 속성

```
<Switch
    android:id="@+id/switch_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:showText="true"
    android:switchMinWidth="100dp"
    android:text="스위치" />
```

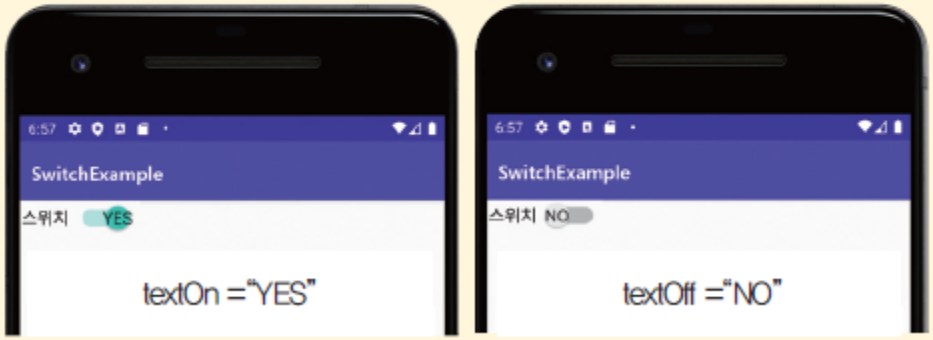


3. 컴파운드 버튼 이벤트 처리하기

■ 스위치 이벤트 처리

- textOn/testOff 속성

```
<Switch
    android:id="@+id/switch_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:showText="true"
    android:textOff="NO"
    android:textOn="YES"
    android:text="스위치" />
```

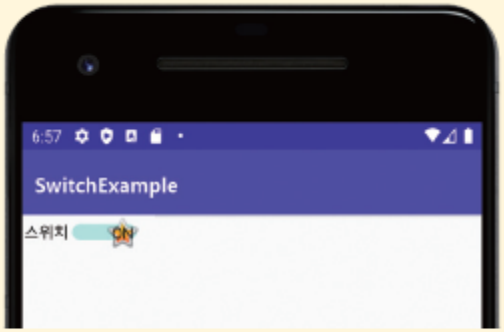


3. 컴파운드 버튼 이벤트 처리하기

■ 스위치 이벤트 처리

- thumb 속성

```
<Switch
    android:id="@+id/switch_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:showText="true"
    android:text="스위치"
    android:thumb="@android:drawable/btn_star_big_on" />
```

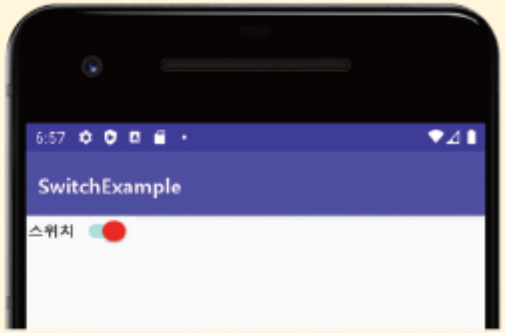


3. 컴파운드 버튼 이벤트 처리하기

스위치 이벤트 처리

- thumbTint 속성

```
<Switch
    android:id="@+id/switch_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:text="스위치"
    android:thumbTint="#ff0000" />
```



3. 컴파운드 버튼 이벤트 처리하기

- **[실습 예제 5-5] 스위치로 텍스트 변경하기**
  - 구현 내용 : 숨김, 굵게, 색상 3개의 스위치를 만들고 선택된 스위치 상태에 따라 텍스트 뷰에 다른 텍스트를 출력합니다.