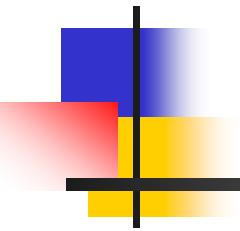


Button

배 희호 교수
경복대학교
소프트웨어융합과

Event-Driven Architecture(EDA)

- Event-Driven 방식은 System의 구성 요소들이 Event를 통해 상호 작용하는 Software Architecture Pattern
- 이 방식에서는 특정 Event가 발생했을 때 이를 감지하고 그에 맞는 동작을 수행하는 구조로 설계
- Event
  - System에서 발생하는 중요한 상태 변화나 작업 완료 등을 의미
  - 예) Button Click, Database Update, File Upload 등이 Event에 해당
- Event Source
  - Event를 생성하는 요소
  - 사용자 입력, System 상태 변화, Timer등 다양한 Source가 Event를 생성할 수 있음

Event-Driven Architecture(EDA)

- Event Listener
  - 특정 Event가 발생했을 때 이를 감지하고 처리하는 Code
  - Event Handler라고도 부름
- Event Bus
  - Event Source와 Event Listener간의 Message을 담당하는 중간 매개체
  - 이를 통해 느슨한 결합(Loose Coupling)을 유지할 수 있음

Event-Driven Architecture(EDA)

- 장점
  - 높은 응답성
    - Event가 발생할 때마다 즉각적으로 처리되므로 System의 응답성이 높음
  - 유연성 및 확장성
    - 새로운 Event 유형을 쉽게 추가할 수 있고, Event Handler를 독립적으로 개발 및 배포할 수 있어 확장성이 좋음
  - 느슨한 결합
    - Event Source와 Listener가 직접적으로 연결되지 않으므로, System 구성 요소들이 서로 독립적으로 동작할 수 있음(이는 유지보수성을 높임)
  - 비동기 처리
    - Event를 비동기적으로 처리할 수 있어, System 자원을 효율적으로 사용할 수 있음

Event-Driven 방식

- Application과 사용자간의 상호 작용
  - Program이 반응하도록 사용자가 만들어내는 동작을 발생하는 것을 말함
  - 예) SmartPhone 화면을 Touch하거나 Button을 Click하는 것
- Android Program은 시작부터 순차적으로 실행되는 것이 아니라 Event가 일어나기 전에는 다른 작업을 하고 있다가, Event가 발생하면 그 때 Application이 미리 지정해 둔 동작이 실행됨
- 이 처리 방식을 바로 Event-Driven 방식

Event-Driven 방식

- Event 처리 방법
  - View에서 발생하는 해당 Event 처리를 위해서는 각각의 Listener Interface를 구현하면서 CallBack 메소드에 처리하고자 하는 내용을 작성해야 함

- Android에서 Event를 처리하는 방법
  - XML File에 Event 처리 메소드를 등록하는 방법
  - Event 처리 객체를 생성하는 방법
  - View 클래스의 Event 처리 메소드를 재 정의하는 방법

Event-Driven 방식

- Call Back 메소드
  - 특정 Event 발생시 System에 의해 자동으로 호출되는 메소드
  - 이 메소드에 Code를 작성함으로써 Event 발생 시의 동작을 정의함
  - Event를 받는 가장 쉬운 방법은 해당 클래스를 상속받아 Call Back 메소드를 재 정의하는 것

Event-Driven 방식

- XML File에 Event 처리 메소드를 등록하는 방법
  - 유일하게 Click Event만 처리할 수 있는 방법
  - activity_main.xml File에서 TextView의 onClick 속성으로 “textViewClicked”라는 메소드를 지정해 줄 수 있음
  - 사용자가 TextView를 Click하면 위의 Layout을 사용하는 Activity인 MainActivity.java에는 textViewClicked()라는 이름의 메소드가 정의 되어 있어야 하며, 이 메소드가 Click Event를 처리함

Event-Driven 방식

- Event 처리(Listener) 객체를 생성하는 방법
  - 가장 일반적으로 사용되는 방법으로 Event를 처리하는 객체를 별도로 생성하여 Widget에 등록하는 방법
  - Event를 처리하는 객체는 Event를 처리하는 메소드를 가지고 있어야 하며, 이는 Event Listener라는 Interface를 사용하면 됨

- Event Listener란?
  - Event를 처리하는 메소드들이 정의된 Interface를 말함
  - Interface는 멤버 변수가 필요 없이 꼭 필요한 메소드를 강제로 구현하게 하도록 하는 추상 메소드의 집합
  - Interface는 멤버 변수가 없어서 클래스(객체)로서의 역할은 할 수 없지만, 이는 Listener 객체를 Modeling하기 위한 것이 아닌, 메소드 Library를 만들어 놓고 활용하기 위한 것

Event-Driven 방식

- Event 처리(Listener) 객체를 생성하는 방법
- Interface란?
  - Interface는 내용이 없는 메소드들의 형태만 구현해 놓은 추상 메소드의 집합이라고 할 수 있음
  - Interface는 일반 메소드나 일반 변수를 가질 수 없으며 변수의 형태는 static만 가능
  - 클래스와 가장 큰 차이점이 바로 생성자를 가질 수 없다는 것

Event-Driven 방식

- Event 처리(Listener) 객체를 생성하는 방법
  - Listener의 종류

리스너	콜백 메소드	설명
OnClickListener	onClick()	사용자가 어떤 항목을 터치하거나 내비게이션 키나 트랙볼로 항목으로 이동한 후에 엔터키를 눌러서 선택하면 호출
OnLongClickListener	onLongClick()	사용자가 항목을 터치한 상태로 일정 시간동안 그대로 누르고 있으면 발생
OnFocusChangeListener	onFocusChange()	사용자가 하나의 항목에서 다른항목으로 이동할 때 호출
OnKeyListener	onKey()	포커스를 가지고 있는 항목 위에서 키를 눌렀다가 놓았을 때 호출
OnTouchListener	onTouch()	사용자가 터치 이벤트로 간주되는 동작을 한 경우에 호출

Event-Driven 방식

- Listener 객체를 생성하는 방법
  - 내부 클래스
  - 익명 클래스
  - Activity 클래스 자체에 구현
  - 람다식

Event-Driven 방식

- View 클래스의 Event 처리 메소드를 재 정의하는 방법
  - Android의 Component들은 모두 View 클래스를 상속
  - Android 장치에서 사용자와 상호작용하는 객체는 View 클래스라는 것
  - Event가 발생하면 View 클래스에 정의되어 있는 콜백 메소드가 호출. 따라서 View에서 발생한 Event를 처리하기 위해서는 View 클래스에 정의된 Event 메소드를 재 정의하는 것이 가장 확실한 방법
  - 사용자 정의 클래스에서 이 방법을 사용하려면 반드시 View 클래스를 상속 받아야 함. 하지만 Event 처리 만을 위해서 View 클래스를 통째로 상속받는 것은 효율적이지 못함
  - 이 방법은 사용자가 자신만의 Custom Component를 만들고자 View 클래스를 상속 받았을 경우에 사용하면 동시에 Event 처리까지 할 수 있어서 좋음

Event-Driven 방식

- Activity 자체에 Interface를 구현하는 방법
  - Interface를 구현한다는 말은 클래스를 정의할 때 Interface를 implements 한다는 것
  - Activity 자체에 Listener를 implements할 수 있음

새로운 Activity 실습

- Activity 전환 방법

새로운 Activity 생성
1.Activity.xml 생성
2.새로운 Activity.java 생성

기존의 Activity.JAVA에서
새로운 Activity.JAVA 파일 로드

manifest.xml에서 Activity 추가 확인하기

Button

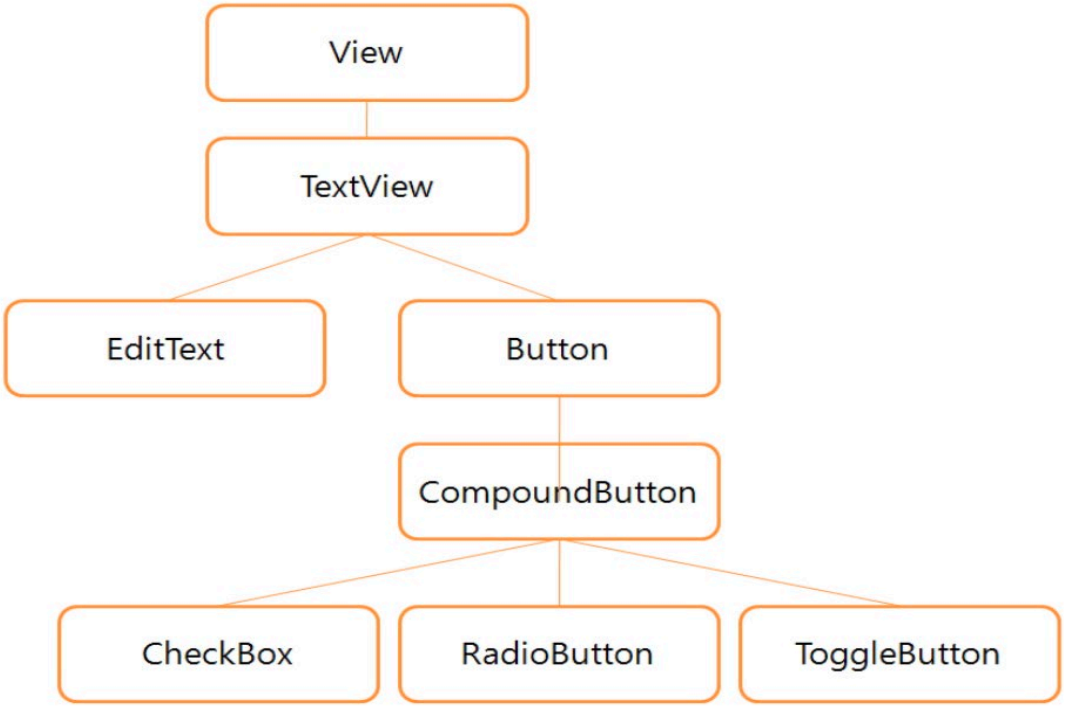
- Button은 View의 배경을 Button 모양을 취하는 것 이외에 특별한 기능은 없음
- Button의 주 기능은 사용자가 Button을 Touch하는 동작으로 명령을 내릴 수 있다는 점
- 이러한 기능은 Button의 기능이 아니라 최상위 View의 기능
  - View에서 파생된 모든 View들이 이 기능 사용할 수 있음
- Button은 TextView에서 파생되었고, TextView만의 기능을 제외하면 별다른 기능이 없음
- 다만 Button 모양의 배경 Image와 Button을 눌렀을 때 효과만이 TextView와 구별되는 기능

Button 계층도

```
java.lang.Object
    └ android.view.View
        └ android.widget.TextView
            └ android.widget.Button
```

Button

- View, TextView의 서브 클래스
- Button은 Android Application에서 사용자와 상호 작용을 하는 가장 기본적인 UI Widget으로 사용자가 Click하여 특정 작업을 Trigger할 수 있도록 하는 역할을 함



Button

- Android에서 Button은 사용자가 누르면 특정 동작을 수행할 수 있게 하는 Interface 요소
- 다양한 종류의 Button이 존재하며, 각각의 Button은 특정 용도나 사용자 경험을 제공하기 위해 설계되었음

Button

- Button의 기본적인 기능
  - 사용자 Interaction
    - Button은 사용자가 Click하거나 Touch할 수 있는 영역을 제공
    - Button을 누르면 해당 Button에 연결된 Event가 발생
  - Event 처리
    - Button은 보통 Click Event(onClickListener)를 통해 특정 기능을 실행
    - 예) Button을 Click하면 새로운 Activity를 시작하거나, Data를 저장하거나, 화면에 메시지를 표시할 수 있음
  - 상태 변화
    - Button은 다양한 상태(정상, 눌림, 비활성화 등)에 따라 시각적으로 변할 수 있음

Button

- Button의 종류
  - 기본 Button
    - 일반적인 Touch Button으로, Text나 Image를 포함할 수 있음
    - android:background, android:text, android:textColor 등을 통해 Style을 쉽게 변경할 수 있음
  - Image Button
    - Image를 Click할 수 있는 Button으로, src 속성에 Image를 설정하여 사용
    - 주로 Icon Button으로 사용
  - Toggle Button
    - 2가지 상태(켜짐/꺼짐)를 나타내는 Button
    - 상태에 따라 다른 Text나 Style을 설정할 수 있음
    - 예) Wi-Fi 설정의 On/Off Toggle로 사용될 수 있음

Button

- Button의 종류
  - Switch
    - Toggle Button과 비슷하지만, Switch처럼 왼쪽과 오른쪽으로 이동하여 상태를 변경
    - 주로 설정 화면에서 Option을 켜고 끄는 데 사용
  - RadioButton
    - 여러 Option 중 하나만 선택할 수 있게 하는 Button
    - RadioGroup내에서 사용되며, Group내에서 하나의 Button만 선택 가능하게 함
  - CheckBox
    - 하나 이상의 Option을 선택할 수 있게 하는 Button
    - 여러 개의 CheckBox를 독립적으로 선택하거나 해제할 수 있음

Button

- Button의 종류
  - FloatingActionButton
    - 화면의 특정 위치에 떠 있는 동그란 Button으로, 주로 가장 중요한 Action을 강조하는 데 사용’
    - 주로 Material Design Guideline에 따라 사용
  - 추가적인 Custom Button
    - 개발자는 다양한 XML 속성이나 Custom Drawable을 사용하여 Button의 모양과 동작을 Customizing할 수 있음
    - 예) GradientDrawable을 사용해 Gradation 배경을 가진 Button을 만들 수 있음

Button

- Button은 Style만 다른 TextView이며 Style은 결국 속성의 집합이므로 속성만 다른 TextView
- 사용자에게서 어떤 값을 입력 받기 위한 가장 기본적인 Widget으로 활용도가 높음

```
<Textview
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="나는 어떤 위젯일까요?"/>
```

```
<Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="나는 어떤 위젯일까요?"/>
```

Button

- Button은 일반적인 Push Button을 표현하며 손가락으로 눌러 명령을 내림
- Button은 문자열과 배경으로 구성되어짐
  - 문자열 : 내용, 크기, 색상 등을 선택 할 수 있음
  - 배경 : background 속성으로 지정
- 다음 메소드로 Code에서 배경을 변경할 수 있는데 단색, drawable, drawable resource의 ID를 전달 받음

```
void setBackgroundColor (int color)
void setBackgroundDrawable (Drawable d)
void setBackgroundResource (int resid)
```

- Button의 배경으로 사용할 Image를 drawable folder에 넣어두고 background 속성에 지정하면 Image가 Button의 배경에 나타나게 됨

Button

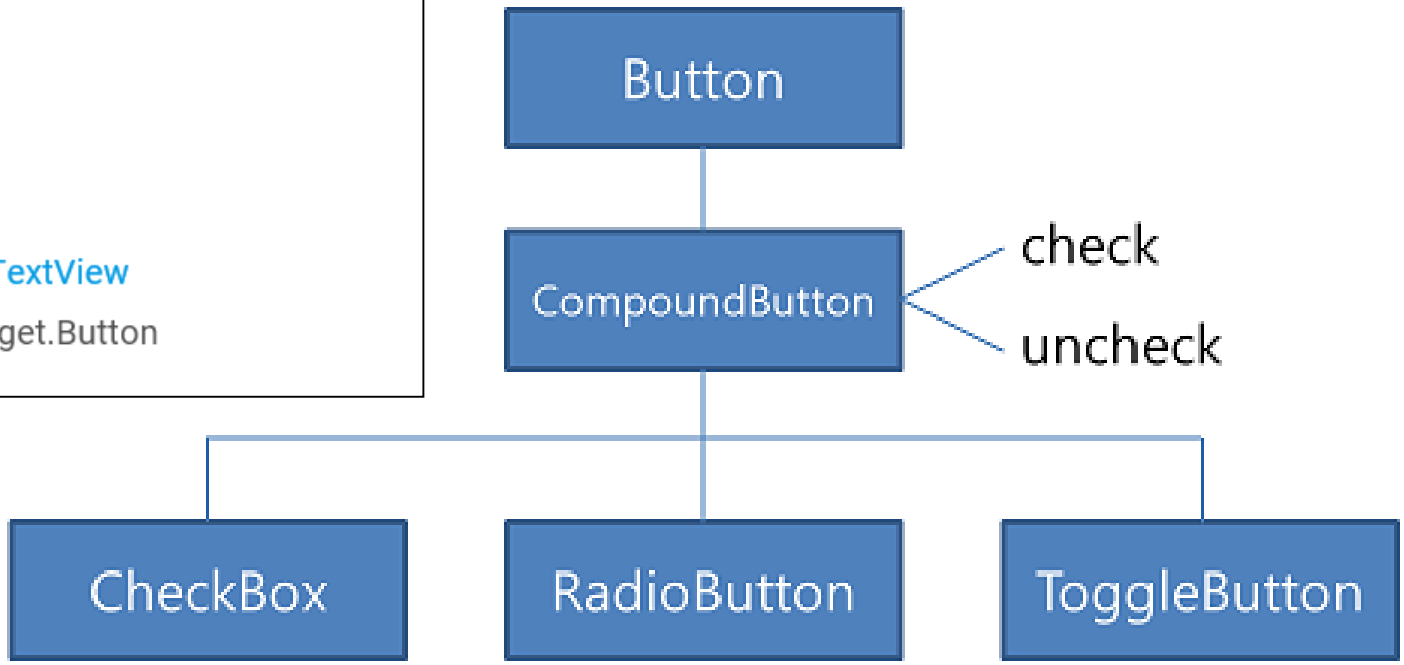
- Button은 사용자가 화면을 Touch했을 때 발생하는 Click Event를 처리하는 기능을 가진, Text 또는 Icon(또는 Text와 Icon 모두)으로 구성된 View Widget
- TextView와 마찬가지로 Android UI를 구성할 때 가장 많이 사용되는 Widget 중 하나
- Button은 기본적으로 사용자의 Touch Event를 처리할 수 있기 때문에 TextView와는 다른 목적으로 사용되지만, 거의 모든 기능을 TextView로부터 상속함
- Button 클래스의 부모 클래스는 TextView이며, TextView에서 정의된 대부분의 속성이나 기능들을 Button에서 그대로 사용할 수 있음

Button

Button

```
public class Button
extends TextView
```

java.lang.Object
↳ android.view.View
↳ android.widget.TextView
↳ android.widget.Button



- ✓ Switch
  - ✓ On/Off 상태 표시가 주 목적
  - ✓ Android 4.0(API 14)부터 지원

Button

- 속성

속성	내용
backgroundTint	Button의 배경 색상을 지정
textAllCaps	Button의 영어 Text를 전부 대문자로 할지 정함 false = Button에 대문자만 출력되는 것 방지
textColor	Button Text의 색상을 지정
text	Button에 보이는 Text를 설정
rotation	android 3.0 이상, API Level 11이상에서 지원 예) android:rotation="45" -> 시계 방향으로 45도만큼 회전

Button

- 속성

속성	내용
autoLink	✓none (0x00) 링크 적용 안함 (기본 값) ✓web (0x01) Web URL로 사용 (클릭 시, 웹 브라우저 앱 실행) ✓email (0x02) email 주소로 사용(클릭 시, 메일 클라이언트 실행) ✓phone (0x04) 전화번호로 사용 (클릭 시, 전화 걸기 앱 실행) ✓map (0x08) 지도 주소로 사용 (클릭 시, 지도 앱 실행) ✓all (0x0f) 모든 경우 사용 ("web | email | phone | map")

Button

- 메소드

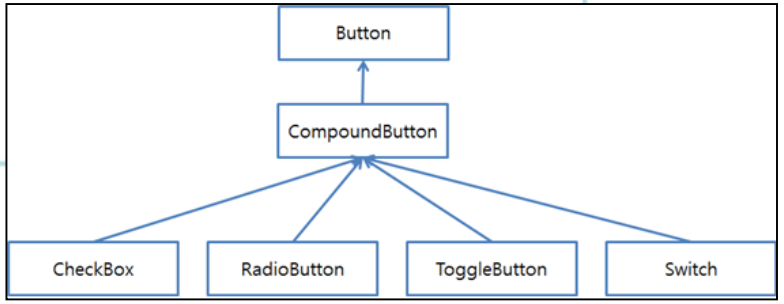
메소드	내용
setOnClickListener (View.OnClickListener l)	Button을 Click했을 때 호출될 콜백 Listener를 등록
setOnLongClickListener (View.OnLongClickListener l)	Button을 길게 Click했을 때 호출될 콜백 Listener를 등록
onClick(View v)	View를 Click했을 때 호출 v는 Click한 View를 의미함
onLongClick(View v)	View를 길게 Click했을 때 호출 v는 Click한 View를 의미함

CompoundButton

- Button 클래스의 하위 클래스
- CompoundButton 클래스 자체만으로는 사용되지 않음
- CompoundButton 클래스를 상속 받은 CheckBox, RadioButton, ToggleButton, Switch를 사용

CompundButton 계층도

```
java.lang.Object
    ㄴ android.view.View
        ㄴ android.widget.TextView
            ㄴ android.widget.Button
                ㄴ android.widget.CompundButton
                    ㄴ android.widget.CheckBox
                    ㄴ android.widget.RadioButton
                    ㄴ android.widget.Switch
                    ㄴ android.widget.ToggleButton
```



CompoundButton

- CompoundButton은 사용자의 선택을 받을 수 있는 Widget
- checked/unchecked 속성을 가지고 있음
  - 속성으로 checked를 사용
  - default 값 : false
- 메소드

속성	설명
public boolean isChecked()	Check 여부를 반환
public void setChecked(boolean checked)	Check 여부를 설정
public void toggle()	Check 상태를 반전

- Listener로 setOnCheckedChangedListener를 추가로 사용할 수 있음

CompoundButton

- Button으로부터 파생되는 서브 클래스
  - 중간의 한 단계를 거쳐 3개의 Widjet이 파생됨
  - CompoundButton은 Check, unCheck의 두 가지 상태를 가짐
  - Code에서 Check 상태를 변경 및 조사할 때는 다음 메소드를 사용
  - CheckBox뿐만 아니라 RadioButton, ToggleButton에도 똑같이 적용됨

```
setChecked(checked:Boolean)
toggle()
isChecked():Boolean
```

  - Check 상태 변경 시 onCheckedChangeListener의 메소드가 호출됨

```
onCheckedChanged (buttonView: CompoundButton,
isChecked:Boolean)
```

ImageButton

- Android에서 제공하는 기본 Button Widjet은 그 자체만으로 깔끔한 Design을 제공하지만 그 형태의 단순함으로 인해 사용자들의 다양한 취향을 만족시키기엔 다소 부족
- png 또는 jpg 등으로 만들어진 Image를 출력하는 Button
- 사용자 Click Event 처리 기능을 생각하면 ImageButton이 Button으로부터 상속받을 것 같지만 ImageButton의 부모는 ImageView
- ImageButton의 "src" 속성에 Image(png 등) Resource를 지정
- ImageButton에 Image를 출력하면 기본 Button 배경(회색) 위에 Image가 출력되는데 ImageButton이 가지고 있는 기본적인 padding 값이 적용되기 때문임
- Image가 ImageButton의 전체 영역에 표시되도록 만들기 위해 padding 값을 0으로 설정

ImageButton

- ImageButton의 상태(normal, pressed, focused)에 따라 다른 Image를 출력하려면 각 상태를 나타내는 Image File들을 Resource에 추가한 다음, 상태에 따라 표시될 Image를 각각 지정
- 상태 별 출력 Image 지정은 Drawable Resource 밑에 Resource File을 만들어서 사용

ImageButton

- xml 파일

```
<?xml version="1.0" encoding="utf-8"?>
<selector
   xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/imgbtn_pressed"
                                          android:state_pressed="true" />
    <item android:drawable="@drawable/imgbtn_focused"
                                        android:state_focused="true" />
    <item android:drawable="@drawable/imgbtn_default" />
</selector>
```

✓ 주의해야 할 점은, item의 순서가 결과에 영향을 미친다는 것으로 첫 번째 item부터 비교를 시작하여 현재 Button의 상태와 일치하는 아이템이 있다면 그 뒤에 있는 item은 더 이상 비교하지 않으며 또한 가장 마지막에 있는 상태 값이 사용되지 않은 item은 "default" 상태로 사용되는데 "pressed"도 아니고 "focused"도 아닌 경우에 출력하는 기본 item

Button

- RadioButton, CheckBox
  - RadioButton은 선택 가능한 여러 개의 값 중 하나를 입력 받을 때 사용
  - CheckBox는 다중 선택을 하고자 하는 경우 사용
  - RadioGroup은 LinearLayout의 서브 클래스이며 RadioGroup을 일렬로 배치
  - Group 내에 속한 RadioButton의 체크 상태를 변경할 때는 다음 메소드를 사용

```
check (id:Int)
clearCheck ()
getCheckedRadioButtonId():Int
```

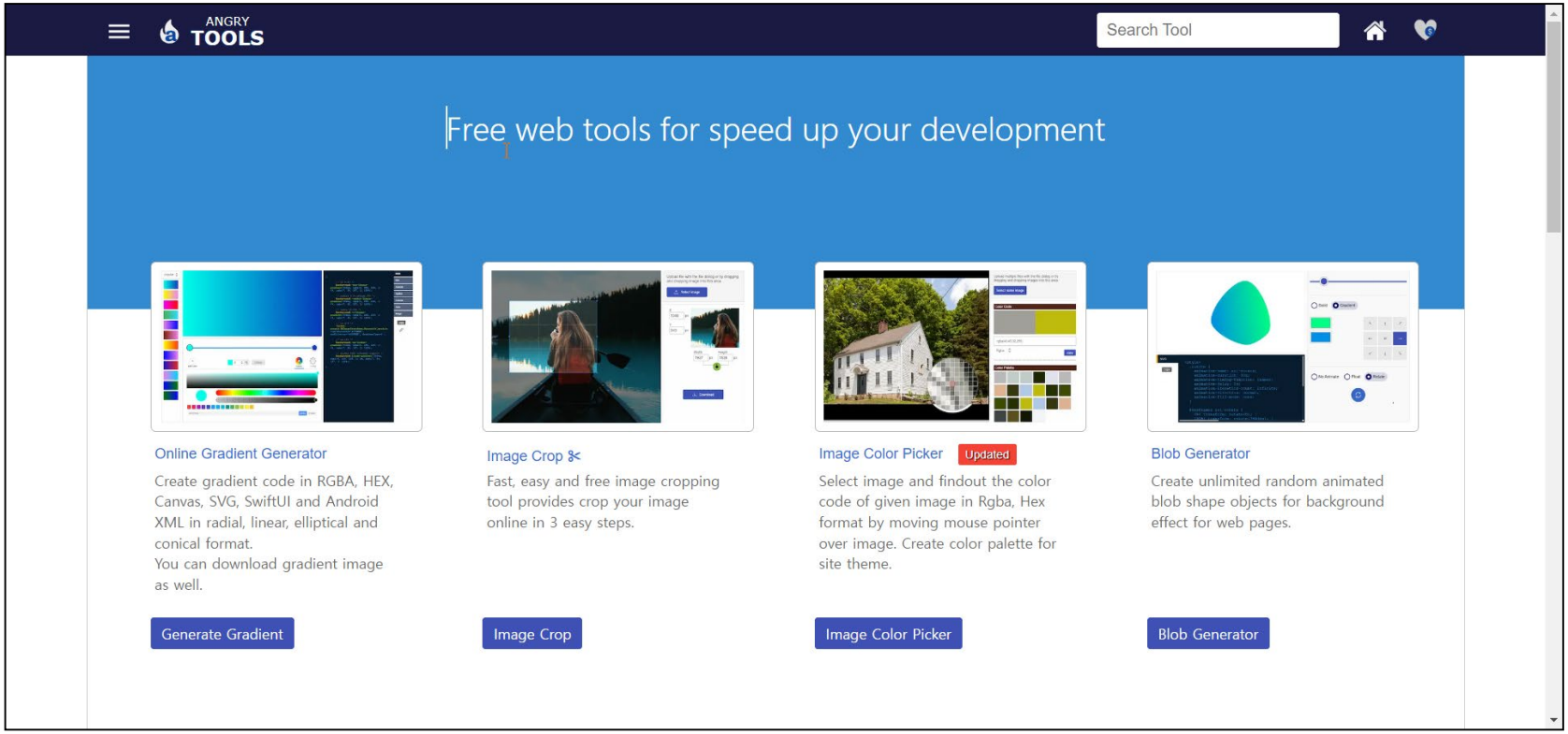
  - Button의 Check 상태가 바뀔 때는 OnCheckedChangeLisener Interface의 메소드가 호출

```
onCheckedChanged(group: RadioGroup, checkedId:Int)
```

Button Design

- 가장 간단한 Button Design 바꾸는 것은 기본이고 쉽지만 Preview를 보면서 간단하게 바꿔주는 Site

http://angrytools.com/android/button/



Widget의 Event 처리

- Android Event 처리 방식
  - GUI(Graphic User Interface)를 제공하는 Program은 화면과 사용자간의 상호 작용에 대해 처리하는 방식을 내부적으로 정의하고 있음
  - 이 상호작용을 Event라고 부르는데, Event에는 Touch Event, Click Event(LongClick Event), Key Event 등이 있음
  - Widget도 View이므로 Event를 처리하는 방법은 동일하지만, 다수의 Widget에 대한 Event를 한번에 처리할 수 있는 효율적인 방법이 존재함

Widget의 Event 처리

- Click Event 처리
  - Button은 자식 Widget이며 Button 클래스를 바로 사용하는 것이 보통이므로 상속을 받지 않고도 Event를 처리 할 수 있어야 함
  - Button의 가장 기본적인 기능은 Click Event를 처리하는 것
  - 사용자가 Button을 Click하면 지정된 동작을 수행하도록 Event Listener를 설정할 수 있음

```
Button button = findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // 수행할 작업
    }
});
```

Widget의 Event 처리

- Click Event 처리
  - OnClickListener 외에도 OnLongClickListener 등 다양한 Event Listener를 설정할 수 있음

```
button.setOnLongClickListener(
                new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View v) {
        // 길게 누를 때 수행할 작업
        return true;
    }
});
```

Button 처리

- Button 처리 5가지 방법
  - Layout Resource XML에서 Button의 onClick 속성을 이용하는 방법
  - Anonymous(익명) 클래스를 생성하여 Event Listener로 사용하기
  - 생성해 놓은 Anonymous 클래스의 참조를 Event Listener로 사용하기
  - Event Listener 인터페이스를 implements하는 Event Listener 클래스 생성하기
    - Inner 클래스 생성 방법
  - Activity에서 Event Listener implements해서 사용하기

Button 처리

- Event Listener를 사용하지 않고 XML의 onClick 속성을 이용하는 방법
  - 가장 간단한 방법이기는 하지만 다른 방법들에 비해 기능상의 제약이 있음
  - JAVA Code에 XML의 onClick에 입력한 이름의 메소드를 직접 작성하면 해당 Button Click시 이 메소드가 호출
  - 메소드의 형식은 반드시 “public void XXX(View v)”이어야 함

Button 처리

- OnClickListener를 Anonymous으로 새로 생성하여 해당 View 객체의 Event Listener로 등록해주는 방식
  - 생성자 new를 이용하여 OnClickListener를 만듦
  - 특정 Button의 Click Event가 어디서 처리되는지 직관적으로 확인할 수 있고, Code 작성이 간결하기 때문에 가장 기본적으로 사용되는 Event 처리 방식

  - 사용하는 경우
    - Button의 개수가 적거나, Button들 간의 연관성이 적은 경우
    - Event Handler 함수 내에서 Anonymous 클래스 외부의 변수를 참조하지 않는 경우
    - 간단한 Button Click Event Test Code를 작성하는 경우

Button 처리

- 생성해 놓은 Anonymous 클래스의 참조를 Event Listener로 사용하기
  - Anonymous 클래스 객체를 먼저 만들어 놓고 그 객체를 모든 Button의 Event Listener로 사용하는 것
  - Anonymous 클래스를 사용한다는 것은 앞의 방법과 비슷하지만 Anonymous 클래스 객체를 매번 생성하지 않는 차이가 있음
  - 각 Button들에 대한 Event 처리 Code가 Event Listener 객체 별로 구분되지 않고 Event Listener 안의 Handler 함수(onClick)에서 구분되는 점도 다름
  - Button이 여러 개 존재하고 Button들 간의 연관성이 많은 경우
  - Event Handler 함수 내에서 Anonymous 클래스 외부의 변수를 참조하지 않는 경우
  - 추후 또 다른 Button을 추가하여 사용할 가능성이 높은 경우

Button 처리

- Event Listener Interface를 implements하는 Event Listener 클래스 생성하기
  - 명시적으로 Event Listener Interface를 상속하여 만듦으로 Code의 가독성을 높이고 싶은 경우
  - 추후 또 다른 Button을 추가하여 사용할 가능성이 높은 경우
  - Inner 클래스 생성 방법도 있음

Button 처리

- JAVA Inner Class 혹은 Nested Class는 하나의 Class로, Class나 Interface 내부에서 선언
- Code를 더 읽기 쉽고, 더 오래 유지 하기 위해, 논리적인 Group과 Interface들에서 Inner Class를 사용
- Inner Class는 개인적인 Data Member와 Method를 포함하는 Outer Class의 모든 Member들에 접근할 수 있음
- JAVA Inner Class의 3가지 이점
  - Nested Class는 개인적인 것을 포함하는 Outer Class의 모든 Member(Data Member와 Method 등)에 접근할 수 있음
  - Nested Class는 논리 Group Class와 Interface 내부에 있기 때문에 더 읽기 쉽고, 유지 가능한 Code 개발에 사용
  - Code 최적화
    - 작성하는데 더 적은 Code가 요구

Button 처리

- OnClickListener를 implement하여 구현하는 방법
  - 이 방법을 사용하기 위해서는 반드시 MainActivity에 OnClickListener를 implement하여 구현하는 방법
  - 즉, implements View.OnClickListener부분이 추가되어야 함
  - OnClickListener의 인자로 현재 Activity(this)를 넘겨주고 OnClick(View v) 함수를 별도로 구현해주면 됨
  - 두 번째 방법에 비해 Code가 깔끔해서 많이 사용되는 방식

Button 예제 1

- Button의 종류를 만드는 Program을 만들어보자

Button 예제 1

- Button의 유형
  - Button
    - Image와 함께 사용
  - ImageButton
  - ToggleButton : 2가지 상태 (textOn, textOff 속성)
  - CheckBox
  - RadioButton (RadioGroup과 함께 사용)
  - Switch : 2가지 상태 (textOn, textOff, showText 속성)

- 사용 방법
  - Layout Design (layout/button_test.xml)
  - Event 처리
    - 각 Button에 대하여 Listener 작성

Button 예제 1

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#00ffff"
        android:gravity="center"
        android:padding="20dp"
        android:text="버튼의 종류"
        android:textSize="20dp" />
```

Button 예제 1

- 사용자 인터페이스

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:gravity="center"
    android:text=" 기본 버튼 "
    android:textSize="24dp"
    android:textStyle="bold" />
```

Button 예제 1

- 사용자 인터페이스

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:orientation="horizontal"
    android:paddingLeft="5dp"
    android:paddingRight="5dp">

    <RadioButton
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:checked="true"
        android:text="남성"
        android:textColor="#0002FF"
        android:textSize="24dp"
        android:textStyle="bold" />
```

Button 예제 1

- 사용자 인터페이스

```
    <RadioButton
        android:layout_width=“0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="여성"
        android:textColor="#0002FF"
        android:textSize="24dp"
        android:textStyle="bold" />
</RadioGroup>
```

Button 예제 1

- 사용자 인터페이스

```
<Switch
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginTop="10dp"
    android:gravity="end"
    android:showText="true"
    android:textOff="시작"
    android:textOn="끝"
    tools:ignore="UseSwitchCompatOrMaterialXml" />

<ToggleButton
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:textOff="시작"
    android:textOn="끝" />
```

Button 예제 1

- 사용자 인터페이스

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical|center_horizontal"
    android:paddingTop="10dp">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingRight="10dp"
        android:text="하루종일"
        android:textColor="#FF0000"
        android:textSize="24dp" />
    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true" />
</LinearLayout>
```

Button 예제 1

- 사용자 인터페이스

```
    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="10dp"
        android:src="@drawable/drawable_selector" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:drawableLeft="@drawable/ic_launcher"
        android:layout_gravity="center"
        android:text="My Image Button"/>
</LinearLayout>
```

Button 예제 1

- drawable/drawable_selector.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:drawable="@drawable/sad"
        android:state_pressed="true" />
    <item android:drawable="@drawable/happy"
        android:state_pressed="false" />
</selector>
```

- Image 준비

sad.png

happy.png

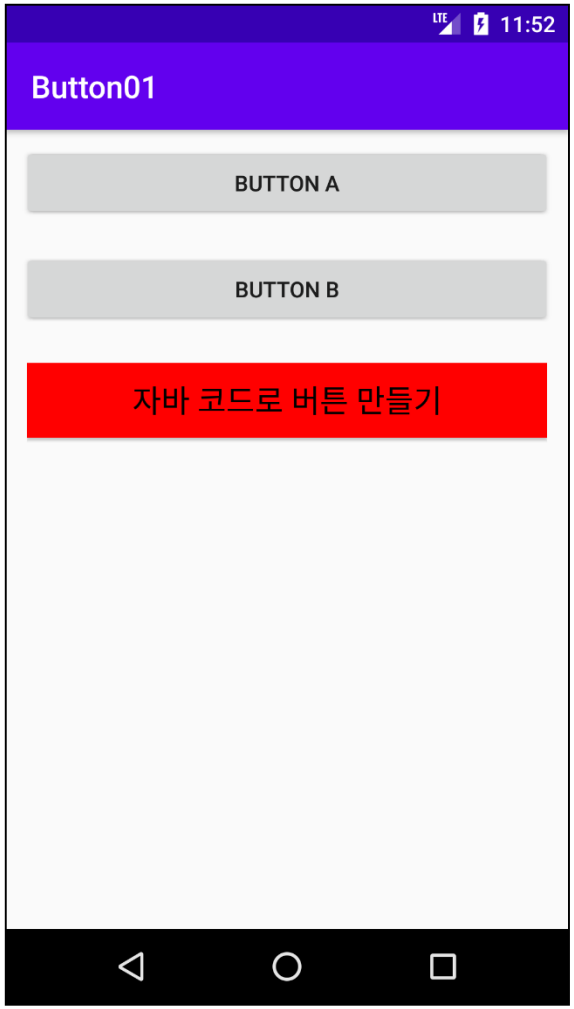
Button 예제 1

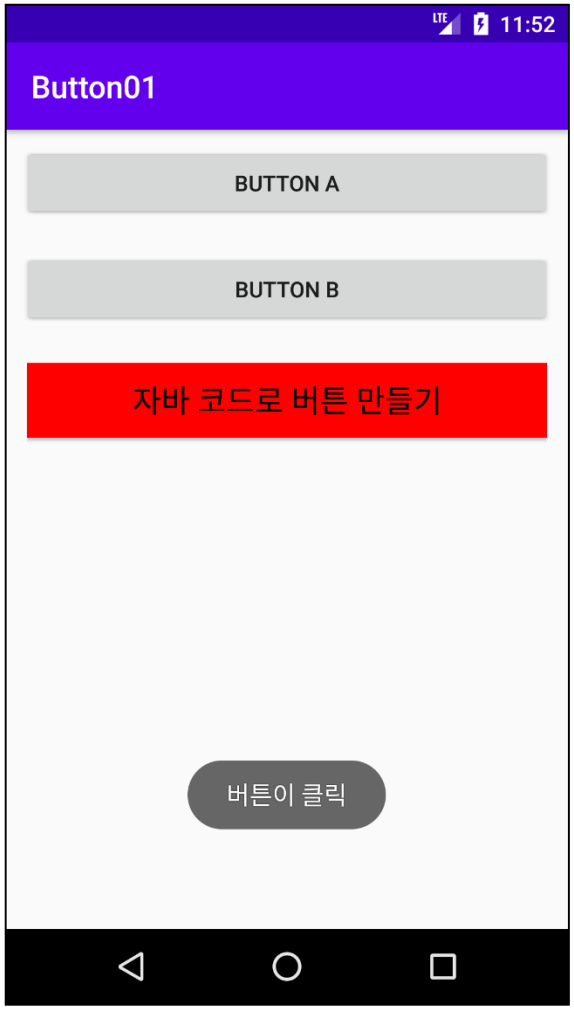
- ToggleButton은 기능상 CheckBox와 동일
  - 둘 중 하나의 값을 선택
  - Check 상태와 unCheck 상태를 명시적인 문자열로 분명히 표시 가능
  - Design 시에 XML 속성으로 지정 가능하며, 실행 중 메소드로도 변경 가능

XML 속성	메소드	설명
android:textOn	getTextOn, setTextOn	선택 상태일 때의 텍스트
android:textOff	getTextOff, setTextOff	선택 해제 상태일 때의 텍스트

Button 예제 2

- JAVA Code로만 Button을 만들어보자





Button 예제 2

- activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30px"
        android:text="Button A" />
</LinearLayout>
```

Button 예제 2

- MainActivity.JAVA

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main2);

    LinearLayout layout = findViewById(R.id.layout);

    Button button = new Button(this);
    LinearLayout.LayoutParams paramB = new LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.MATCH_PARENT,
            ViewGroup.LayoutParams.WRAP_CONTENT);
    paramB.setMargins(30, 30, 30, 30);
    button.setPadding(30,30,30,30);
    button.setLayoutParams(paramB);
    button.setText("BUTTON B");
    layout.addView(button);
```

Button 예제 2

- MainActivity.JAVA

```
LinearLayout linearLayout = new LinearLayout(this);
LinearLayout.LayoutParams layoutParams = new LinearLayout.LayoutParams(
                    ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT);
layoutParams.setMargins(30, 30, 30, 30);
linearLayout.setOrientation(LinearLayout.VERTICAL);
linearLayout.setLayoutParams(layoutParams);
linearLayout.setPadding(10, 10, 10, 10);
layout.addView(linearLayout);

Button button = new Button(this);
button.setText("자바 코드로 버튼 만들기");
button.setBackgroundColor(Color.RED);
button.setTextColor(getResources().getColor(R.color.black));
button.setTextSize(20);
linearLayout.addView(button);
```

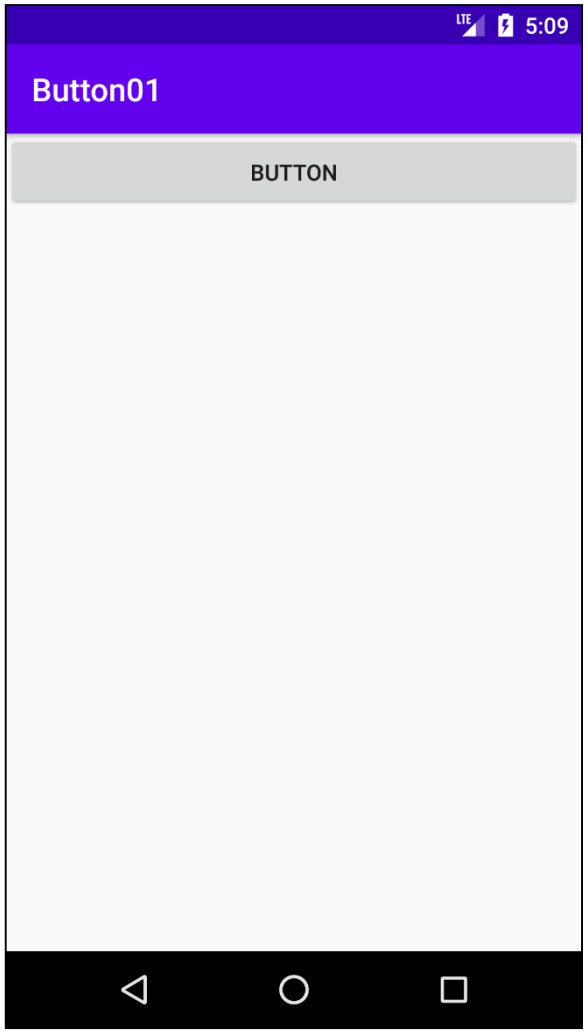
Button 예제 2

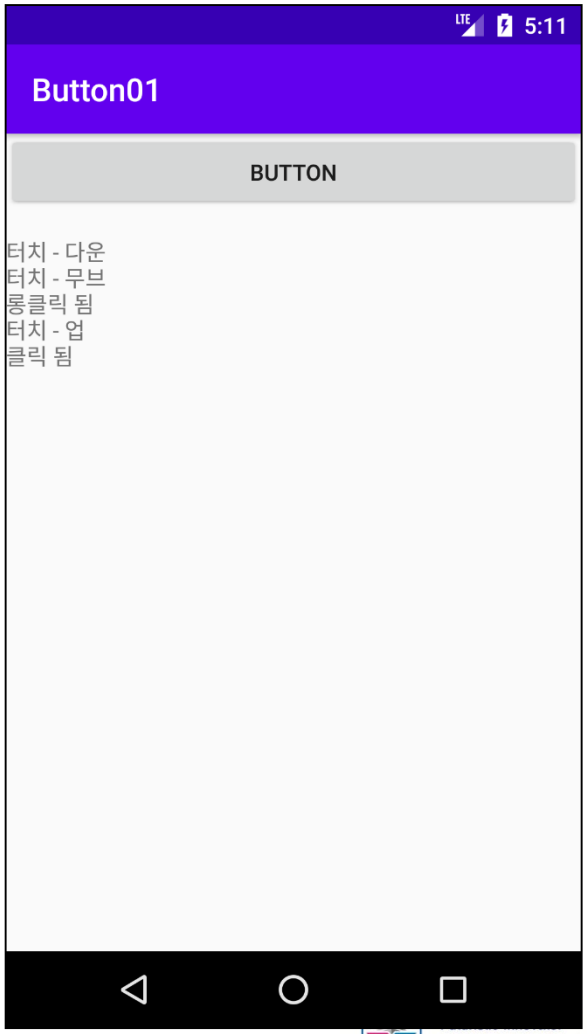
- MainActivity.JAVA

```
    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Toast.makeText(getBaseContext(), "버튼이 클릭",
                                                    Toast.LENGTH_LONG).show();
        }
    });
}
```

Button 예제 3

- Event 발생 순서를 알아보자





Button 예제 3

- Button Event
  - Click
  - LongClick
  - Touch

- Touch Event와 Click Event 그리고 LongClick Event는 어떠한 관계에 있을까?
  - 만약 사용자가 Button을 Click하면
    - Touch(down) -> Touch(up) -> OnClick 순으로 발생
  - 만약 사용자가 Button을 LongClick하면
    - Touch(down) -> OnLongClick -> Touch(up) -> OnClick 순으로 발생

Button 예제 3

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity10">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="button" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Button 예제 3

- MainActivity.JAVA

```
public class MainActivity10 extends AppCompatActivity {
    TextView textView;

    @SuppressLint("ClickableViewAccessibility")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main10);

        textView = findViewById(R.id.textView);
        Button button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                output("클릭 됨");
            }
        });
```

Button 예제 3

- MainActivity.JAVA

```
button.setOnTouchListener(new View.OnTouchListener() {
    public boolean onTouch(View v, MotionEvent event) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                output("터치 - 다운");
                break;
            case MotionEvent.ACTION_MOVE:
                output("터치 - 무브");
                break;
            case MotionEvent.ACTION_UP:
                output("터치 - 업  ");
        }
        return false;
    }
});
```

Button 예제 3

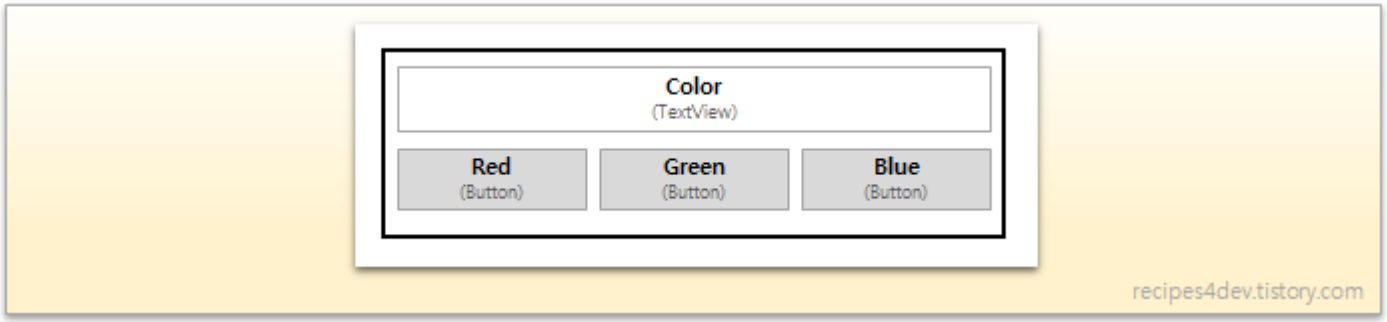
- MainActivity.JAVA

```
        button.setOnLongClickListener(new View.OnLongClickListener() {
            public boolean onLongClick(View v) {
                output("롱클릭 됨");
                return false;
            }
        });
    }

    void output(String str) {
        textView.setText(textView.getText().toString() + "₩n" + str);
    }
}
```

onTouch()와 onLongClick()에서 true를 반환하지 않고 false를 반환해야 다음에 발생되는 Event가 뭔지 알 수 있음

Button 예제 4

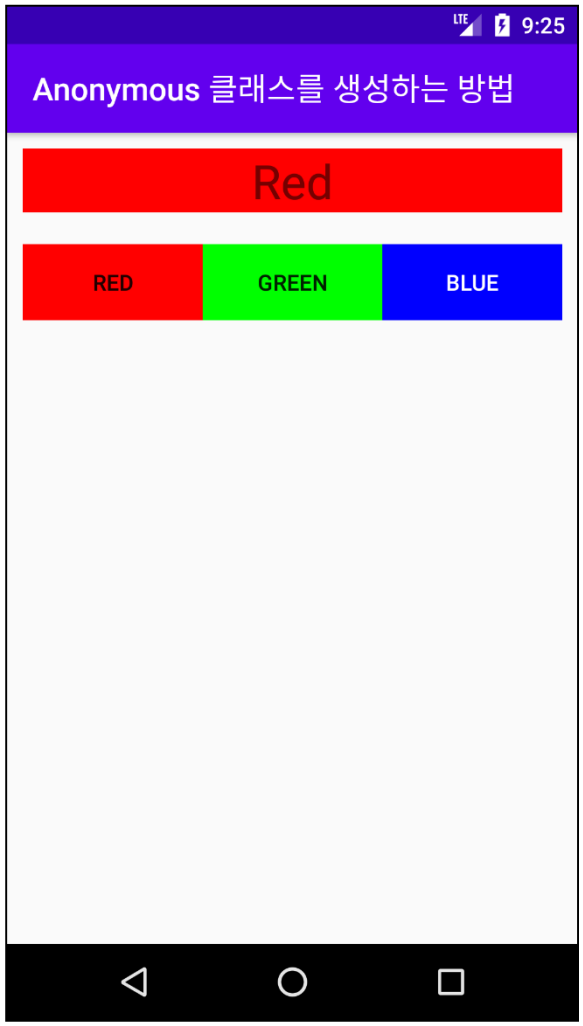
- 3개의 Button은 각각 Red, Green, Blue라고 하고, 각 Button이 Click되면 위쪽에 있는 TextView의 배경 색상을 변경하는 Program을 Button 처리방법 5개에 따라 각각 만들어보자

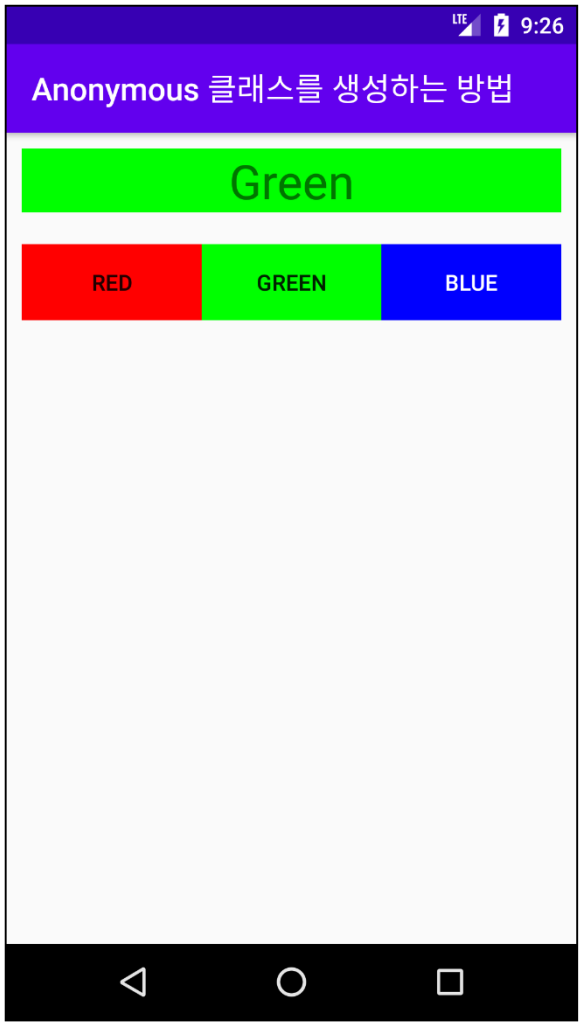


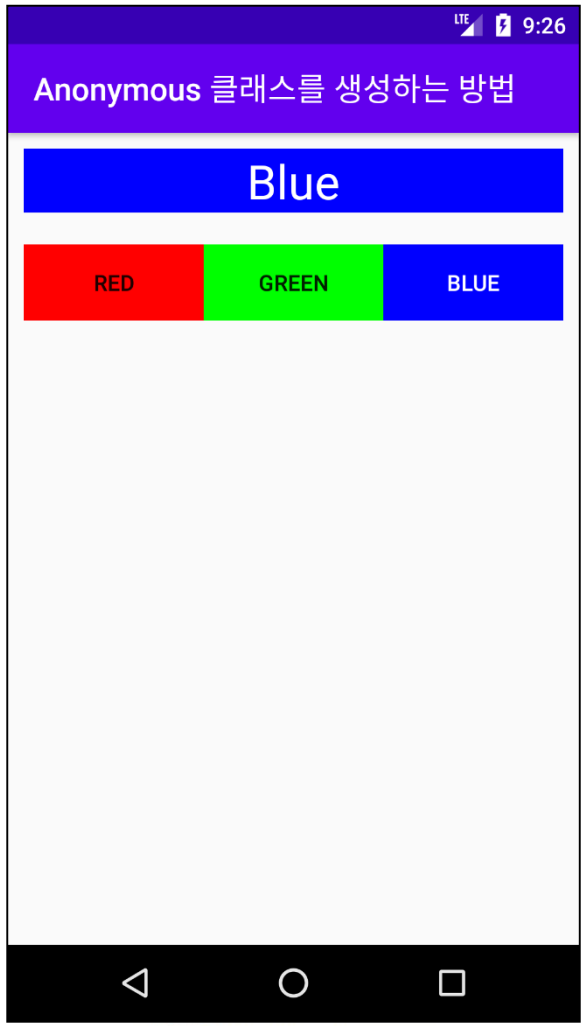
- App name에 Button 처리 방법을 안내할 것

Button 예제 4

- 실행 결과







Button 예제 4

- values/styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="Mystyle">
        <item name="android:layout_width">0dp</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:layout_weight">1</item>
    </style>
</resources>
```

Button 예제 4

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity4">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Color"
        android:textAlignment="center"
        android:textSize="30sp" />
```

Button 예제 4

- 사용자 인터페이스

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp">

    <Button
        android:id="@+id/buttonRed"
        style="@style/Mystyle"
        android:background="#FF0000"
        android:text="RED" />

    <Button
        android:id="@+id/buttonGreen"
        style="@style/Mystyle"
        android:background="#00FF00"
        android:text="GREEN" />
```

Button 예제 4

- 사용자 인터페이스

```
        <Button
            android:id="@+id/buttonBlue"
            style="@style/Mystyle"
            android:background="#0000FF"
            android:text="BLUE"
            android:textColor="@color/white" />
    </LinearLayout>
</LinearLayout>
```

Button 예제 4-1

- Anonymous 클래스를 생성하여 Event Listener로 사용하기

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main4);
    setTitle("Anonymous 클래스를 생성하는 방법");

    TextView textView = findViewById(R.id.textView);
    Button buttonRed = findViewById(R.id.buttonRed);
    buttonRed.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            textView.setText("Red");
            textView.setBackgroundColor(Color.rgb(255, 0, 0));
        }
    });
```

Button 예제 4-1

- Anonymous 클래스를 생성하여 Event Listener로 사용하기

```
    Button buttonGreen = findViewById(R.id.buttonGreen);
    buttonGreen.setOnClickListener(new Button.OnClickListener() {
        @Override
        public void onClick(View view) {
            textView.setText("Green");
            textView.setBackgroundColor(Color.rgb(0, 255, 0));
        }
    });
    Button buttonBlue = findViewById(R.id.buttonBlue);
    buttonBlue.setOnClickListener(new Button.OnClickListener() {
        @Override
        public void onClick(View view) {
            textView.setText("Blue");
            textView.setTextColor(Color.WHITE);
            textView.setBackgroundColor(Color.rgb(0, 0, 255));
        }
    });
    buttonRed.performClick();
}
```

Button 예제 4-1

- Anonymous 클래스를 생성하여 Event Listener로 사용하기
  - Button button;과 같이 Button을 선언
  - button = findViewById(R.id.버튼아이디) 메소드를 통해 View(Button)을 가져와 Button을 연결
  - button.setOnClickListener(new OnClickListener) 메소드를 통해 Click Listener를 등록
  - OnClickListener의 onClick(View view) 메소드를 재 구현(override)해서 Click Event 발생시의 처리를 함

```
Button button = findViewById(R.id.btn1);
button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
          // 이 부분에 Button Click할 때 작동 할 Code를 넣으면 됨

        }
} );
```

Button 예제 4-1

- Anonymous 클래스를 생성하여 Event Listener로 사용하기
  - 특정 Button의 Click Event가 어디서 처리되는지 직관적으로 확인할 수 있음
  - Code 작성이 간결하기 때문에 가장 자주 사용되는 방법
  - 단점
    - Button의 개수가 늘어나면 Button의 개수만큼 Anonymous 클래스 객체를 생성해야 하는 문제가 발생
    - Anonymous 클래스에 정의된 Event Handler에서 Anonymous 클래스 외부에 있는 변수를 사용하려면 변수를 선언할 때 반드시 Global 변수를 사용해야 함

Button 예제 4-1

- Anonymous 클래스를 생성하여 Event Listener로 사용하기
  - Button의 개수가 적거나, Button들 간의 연관성이 적은 경우
  - Event Handler 함수 내에서 Anonymous 클래스 외부의 변수를 참조하지 않는 경우
  - 간단한 Button Click Event Test Code를 작성하는 경우

Button 예제 4-2

- 생성된 Anonymous 클래스 참조 방법

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main4);

    setTitle("생성된 Anonymous 클래스 참조방법");
    TextView textView = findViewById(R.id.textView);
    Button.OnClickListener onClickListener = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            int check = view.getId();
            if (check == R.id.buttonRed) {
                textView.setText("Red");
                textView.setBackgroundColor(Color.rgb(255, 0, 0));
            } else if (check == R.id.buttonGreen) {
                textView.setText("Green");
                textView.setBackgroundColor(Color.rgb(0, 255, 0));
```

Button 예제 4-2

- 생성된 Anonymous 클래스 참조 방법

```
        } else if (check == R.id.buttonBlue) {
                textView.setText("Blue");
                textView.setTextColor(Color.WHITE);
                textView.setBackgroundColor(Color.rgb(0, 0, 255));
            }
        }
    };
    Button buttonRed = findViewById(R.id.buttonRed);
    buttonRed.setOnClickListener(onClickListener);
    Button buttonGreen = findViewById(R.id.buttonGreen);
    buttonGreen.setOnClickListener(onClickListener);
    Button buttonBlue = findViewById(R.id.buttonBlue);
    buttonBlue.setOnClickListener(onClickListener);
    buttonRed.performClick();
}
```

Button 예제 4-2

- 생성해 놓은 Anonymous 클래스의 참조를 Event Listener로 사용하기
  - Anonymous 클래스 객체를 먼저 만들어 놓고 그 객체를 모든 Button의 Event Listener로 사용하는 것
  - 기본적으로 Anonymous 클래스를 사용한다는 점에서 첫 번째 방법과 유사하지만 Anonymous 클래스 객체를 매번 생성하지 않는다는 차이가 있음
  - 각 Button들에 대한 Event 처리 Code가 Event Listener 객체 별로 구분되지 않고 Event Listener 안의 Handler 함수(onClick)에서 구분되는 점도 다름
  - 장점
    - 미리 만들어놓은 Anonymous 클래스 객체를 Event Listener로 사용하는 방법은 매번 객체를 새로 만들지 않아도 됨

Button 예제 4-2

- 생성해 놓은 Anonymous 클래스의 참조를 Event Listener로 사용하기
  - 단점
    - Event가 어디서 처리되는지 첫 번째 방법에 비해 직관적이지 않음
  - Button이 여러 개 존재하고 Button들 간의 연관성이 많은 경우
  - Event Handler 함수 내에서 Anonymous 클래스 외부의 변수를 참조하지 않는 경우
  - 추후 또 다른 Button을 추가하여 사용할 가능성이 높은 경우

Button 예제 4-3

- OnClickListener.JAVA

```
public class OnClickListener implements Button.OnClickListener {
    private TextView textView;

    public OnClickListener(TextView textView) {
        this.textView = textView;
    }

    @Override
    public void onClick(View view) {
        int check = view.getId();
        if (check == R.id.buttonRed) {

            textView.setText("Red");
            textView.setBackgroundColor(Color.rgb(255, 0, 0));
        } else if (check == R.id.buttonGreen) {
            textView.setText("Green");
            textView.setBackgroundColor(Color.rgb(0, 255, 0));
```

Button 예제 4-3

- OnClickListener.JAVA

```
        } else if (check == R.id.buttonBlue) {
            textView.setText("Blue");
            textView.setTextColor(Color.WHITE);
            textView.setBackgroundColor(Color.rgb(0, 0, 255));
        }
    }
}
```

Button 예제 4-3

- Listener를 implements하는 클래스 생성 방법

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main4);
    setTitle("Listener를 implements하는 클래스 생성 방법");

    TextView textView = findViewById(R.id.textView);
    OnClickListener onClickListener = new OnClickListener(textView);
    Button buttonRed = findViewById(R.id.buttonRed);
    buttonRed.setOnClickListener(onClickListener);
    Button buttonGreen = findViewById(R.id.buttonGreen);
    buttonGreen.setOnClickListener(onClickListener);
    Button buttonBlue = findViewById(R.id.buttonBlue);
    buttonBlue.setOnClickListener(onClickListener);
    buttonRed.performClick();
}
```

Button 예제 4-3

- Event Listener Interface를 implements하는 Event Listener 클래스 생성하기
  - Button.OnClickListener를 상속받는 새로운 클래스의 정의를 명시적으로 수행하는 방법
  - 이름이 부여된 Event Listener 클래스가 만들어짐으로써 Button Event가 어디서 처리되는지 조금 더 명확하게 구분됨을 확인할 수 있음
  - 명시적으로 Event Listener Interface를 상속하여 만듦으로 Code의 가독성을 높이고 싶은 경우
  - 추후 또 다른 Button을 추가하여 사용할 가능성이 높은 경우

Button 예제 4-4

- Activity에 Listener implements 하기

```
public class MainActivity7 extends AppCompatActivity
                                        implements View.OnClickListener {
    TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main4);
        setTitle("Activity에 Listener implements 하기");

        textView = findViewById(R.id.textView);
        Button buttonRed = findViewById(R.id.buttonRed);
        Button buttonGreen = findViewById(R.id.buttonGreen);
        Button buttonBlue = findViewById(R.id.buttonBlue);
        buttonRed.setOnClickListener(this);
        buttonGreen.setOnClickListener(this);
        buttonBlue.setOnClickListener(this);
        buttonRed.performClick();
    }
```

Button 예제 4-4

- Activity에 Listener implements 하기

```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.buttonRed:
            textView.setText("Red");
            textView.setBackgroundColor(Color.rgb(255, 0, 0));
            break;
        case R.id.buttonGreen:
            textView.setText("Green");
            textView.setBackgroundColor(Color.rgb(0, 255, 0));
            break;
        case R.id.buttonBlue:
            textView.setText("Blue");
            textView.setTextColor(Color.WHITE);
            textView.setBackgroundColor(Color.rgb(0, 0, 255));
    }
}
}
```

Button 예제 4-4

- Activity에서 Event Listener를 implements해서 사용하기
  - 어떤 클래스가 Button의 Click Event를 처리하는 Event Listener가 되기 위한 필요 조건은 새로운 클래스를 정의하거나 객체를 생성하는 것이 아니라, Button.OnClickListener Interface를 직접 implements하면 됨
  - OnClickListener Interface를 직접 구현 했으므로 Listener는 Activity 자신인 this를 사용
  - Listener를 칭할 수 있는 방법이 생겨서 Listener 등록 메소드로 this를 전달함
  - onClick( ) 메소드는 누구를 Click했는지 View의 v를 전달받으며 v의 getId()를 통해 Click된 Button을 알아냄

Button 예제 4-4

- Activity에서 Event Listener를 implements해서 사용하기
  - Event Handler 메소드에서 많은 수의 Activity Member를 액세스해야 하는 경우
  - Activity 내부의 Button들에 대한 Click Event를 한 곳에서 처리하고 싶은 경우
  - Anonymous 클래스 또는 별도의 Event Listener를 만들고 싶지 않은 경우

Button 예제 4-5

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity8">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Color"
        android:textAlignment="center"
        android:textSize="30sp" />
```

Button 예제 4-5

- 사용자 인터페이스

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp">

    <Button
        android:id="@+id/buttonRed"
        style="@style/Mystyle"
        android:background="#FF0000"
        android:onClick="buttonClicked"
        android:text="RED" />

    <Button
        android:id="@+id/buttonGreen"
        style="@style/Mystyle"
        android:background="#00FF00"
        android:onClick="buttonClicked"
        android:text="GREEN" />
```

Button 예제 4-5

- 사용자 인터페이스

```
        <Button
            android:id="@+id/buttonBlue"
            style="@style/Mystyle"
            android:background="#0000FF"
            android:onClick="buttonClicked"
            android:text="BLUE"
            android:textColor="@color/white" />
    </LinearLayout>
</LinearLayout>
```

Button 예제 4-5

- Button의 onClick 속성 방법

```
public class MainActivity8 extends AppCompatActivity {
    TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main8);
        setTitle("Button의 onClick 속성 방법");

        textView = findViewById(R.id.textView);
        Button button = findViewById(R.id.buttonRed);
        button.performClick();
    }
```

Button 예제 4-5

- Button의 onClick 속성 방법

```
public void buttonClicked(View view) {
        int check = view.getId();
        if (check == R.id.buttonRed) {
            textView.setText("Red");
            textView.setBackgroundColor(Color.rgb(255, 0, 0));
        } else if (check == R.id.buttonGreen) {
            textView.setText("Green");
            textView.setBackgroundColor(Color.rgb(0, 255, 0));
        } else if (check == R.id.buttonBlue) {
            textView.setText("Blue");
            textView.setTextColor(Color.WHITE);
            textView.setBackgroundColor(Color.rgb(0, 0, 255));
        }
    }
}
```

Button 예제 4-5

- Layout Resource XML에서 Button의 onClick 속성을 이용하는 방법
  - Activity의 Layout Resource XML에 Button을 추가할 때 onClick 속성을 사용하여 Event Handler 함수를 지정
  - onClick 속성에 지정한 Handler 함수와 똑같은 이름의 함수를 Activity의 멤버 함수로 추가
    - 함수의 형식은 "public void XXX(View view)"이어야 함
  - 장점
    - 간단하게 Event를 처리할 수 있음
  - 단점
    - Button의 onClick 속성을 사용하게 되면 Layout Resource XML에서 Event 처리에 관여하게 됨으로 역할 구분이 모호해지게 됨
    - Code양이 많아지고 복잡해지면 Button과 Event Handler의 관계를 파악하기가 용이하지 않음

Button 예제 4-5

- Layout Resource XML에서 Button의 onClick 속성을 이용하는 방법
  - Button의 개수가 적은 경우
  - 간단한 Button Click Event Task Code를 작성하는 경우
  - Event Listener를 별도로 작성하는게 번거롭게 느껴지는 경우
  - 가장 간단하고 직관적인 방법을 선호하는 경우

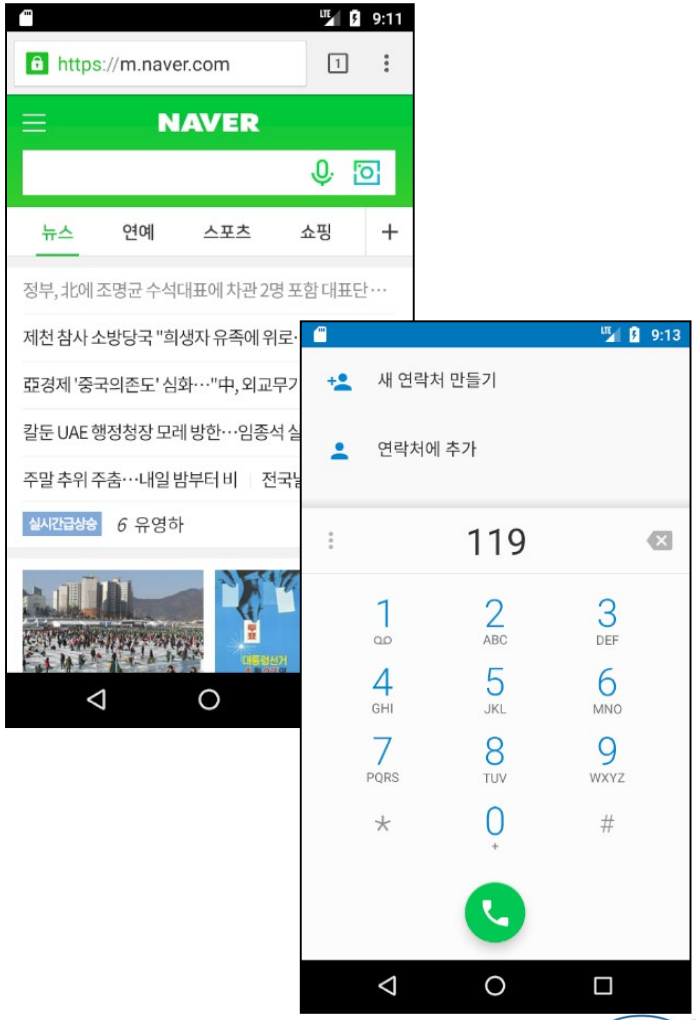
Button 예제 4-5

- Click/LongClick Event 강제 발생시키기
  - Code으로 Button의 Event를 강제로 발생시키는 방법으로 performClick(), performLongClick() 메소드를 사용

```
Button button = findViewById(R.id.button);
button.performClick();          // Click
button.performLongClick();  // Long Click
```

Button 예제 5

- 다음 그림처럼 4개의 Button에 각각의 Event를 연결해 보자



Button 예제 5

- activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity11">

    <Button
        android:id="@+id/buttonNaver"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:background="#ff0000"
        android:onClick="onButtonClicked"
        android:text="Naver 열기" />
```

Button 예제 5

- activity_main.xml

```
<Button
    android:id="@+id/buttonPhone"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:background="#00ff00"
    android:text="119 응급전화 걸기" />

<Button
    android:id="@+id/buttonGallary"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:background="#ff00ff"
    android:text="갤러리 열기" />
```

Button 예제 5

- activity_main.xml

```
    <Button
        android:id="@+id/buttonEnd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="10dp"
        android:background="#00ffff"
        android:text="끝내기" />
</LinearLayout>
```

Button 예제 5

- MainActivity.java

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main11);

        Button button119 = findViewById(R.id.buttonPhone);
        Button buttonGallary = findViewById(R.id.buttonGallary);
        Button buttonEnd = findViewById(R.id.buttonEnd);

        button119.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("tel:119"));
                startActivity(intent);
            }
        });
```

Intent라는 것을 사용하면 “http://...”나 “tel:...” 만으로도 웹 페이지 접속과 전화 걸기 가능

Button 예제 5

- MainActivity.java

```
        buttonGallary.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("content://media/internal/images/media"));
                startActivity(intent);
            }
        });
        buttonEnd.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                finish();
            }
        });
    }
    public void onButtonClicked(View view) {
        Intent intent = new Intent(Intent.ACTION_VIEW,
            Uri.parse("http://m.naver.com"));
        startActivity(intent);
    }
}
```

Button 예제 5

- AndroidMenifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.fivebutton">

    <uses-permission  android:name="android.permission.INTERNET“/>
                    ……..……
```

Button 예제 5

- 처리 방법
  - Naver 연결하기는 'OnClick' 속성을 등록하는 방법

```
Intent intent = new Intent(Intent.ACTION_VIEW,
                                    Uri.parse("http://m.naver.com"));
startActivity(intent);
```

  - 119에 전화 걸기

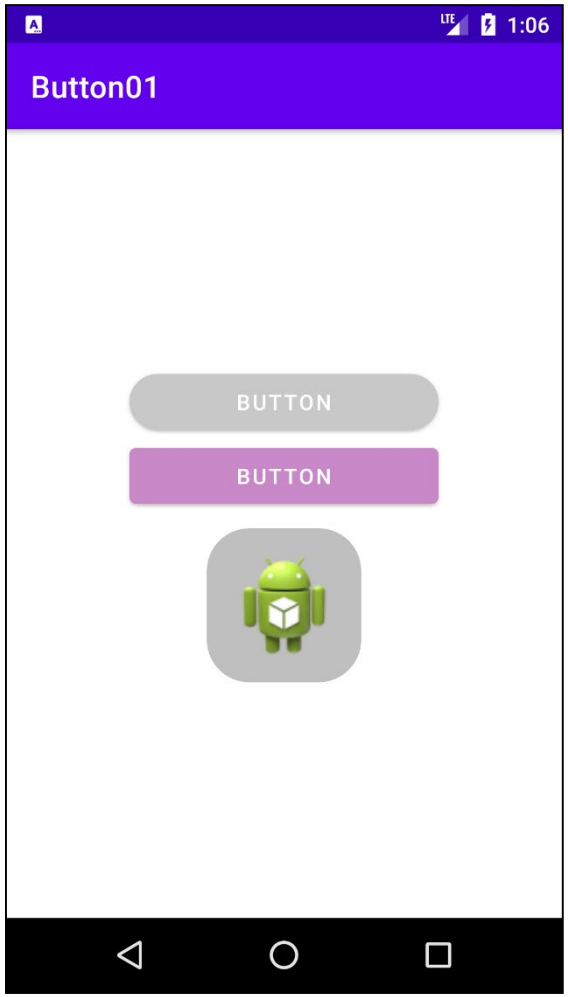
```
Intent intent = new Intent(Intent.ACTION_VIEW,
                                        Uri.parse("tel:119"));
startActivity(intent);
```

Ripple Effect

- Ripple Effect는 Button을 Click했을 때 물결이 퍼지는 듯한 효과가 나오는 기능을 말함
- Android 5.0 (API level 21) Material Design에 소개
- 5.0 미만에는 색상만 변경되고, 이상은 Ripple Drawable까지 적용

- Button에 Ripple Effect를 주는 2가지 방법
  - MaterialButton, MaterialCardView를 사용하는 방법 (Android Support Design Library에서 제공)
  - XML을 사용하는 방법

Ripple Effect



Ripple Effect

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity12">

    <com.google.android.material.button.MaterialButton
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:backgroundTint="#c8c8c8"
        android:text="Button"
        app:cornerRadius="55dp"
        app:rippleColor="#f00" />
```

Ripple Effect

- 사용자 인터페이스

```
<com.google.android.material.button.MaterialButton
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:backgroundTint="#c888c8"
    android:text="Button"
    app:rippleColor="#0ff" />
```

Ripple Effect

- 사용자 인터페이스

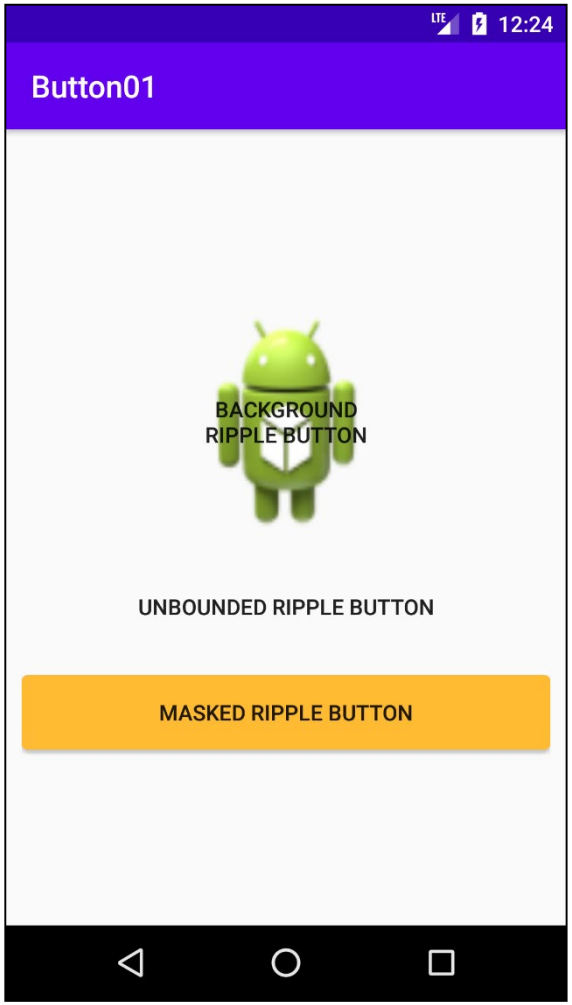
```
    <com.google.android.material.card.MaterialCardView
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_marginTop="10dp"
        android:backgroundTint="#40000000"
        android:clickable="true"
        app:cardCornerRadius="27.5dp"
        app:cardElevation="0dp"
        app:rippleColor="#f00">

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:src="@drawable/ic_launcher" />
    </com.google.android.material.card.MaterialCardView>
</LinearLayout>
```

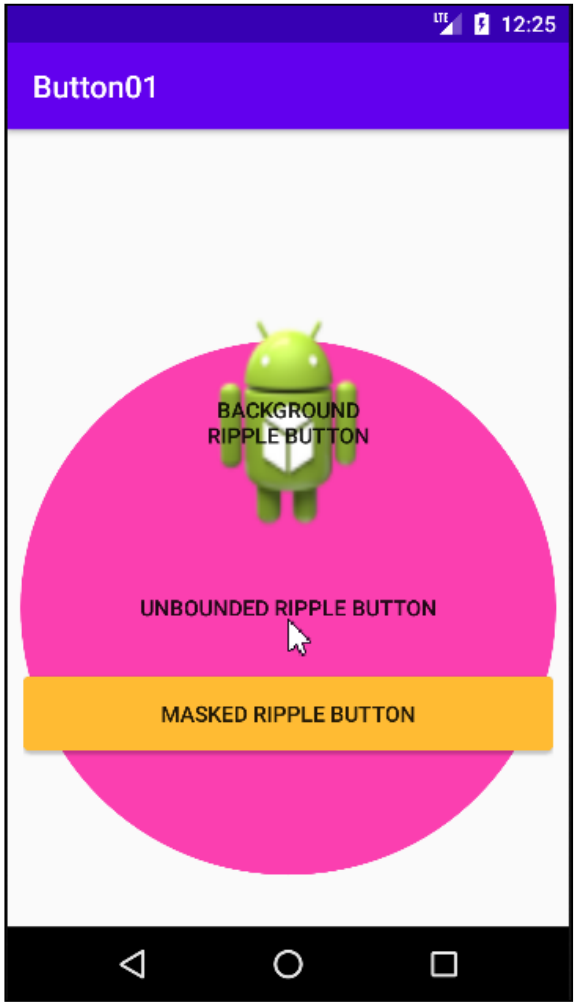
Ripple Effect

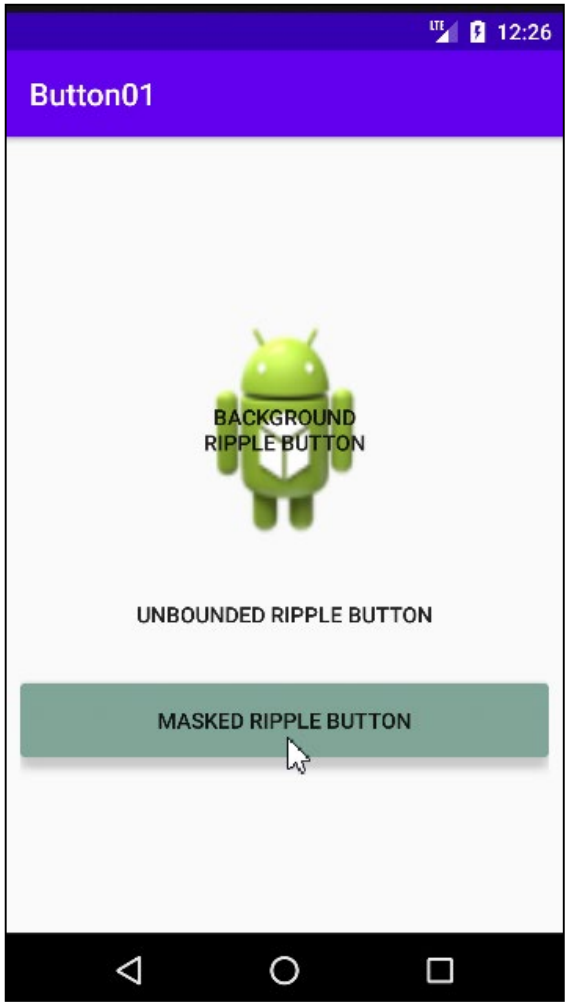
- XML을 사용하는 방법
  - 경계선 없는 Ripple 넣기
    - View의 크기를 벗어나 Ripple 효과가 생김
  - 다른 모양으로 Ripple 넣기
    - XML을 통해 Background로 사용될 drawable, shape 을 넣을 수 있음
    - Ripple 효과는 Background의 모습 안에서만 나타남
  - Mask가 있는 Ripple
    - Ripple 효과의 범위를 Background가 아닌 다른 것으로 설정할 수 있음

Ripple Effect

- 실행 결과







Ripple Effect

- 사용자 인터페이스

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".MainActivity9">

    <Button
        android:layout_width="120dp"
        android:layout_height="150dp"
        android:background="@drawable/ripple_custom1"
        android:text="Background Ripple Button" />
```

Ripple Effect

- 사용자 인터페이스

```
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:background="@drawable/ripple_custom2"
        android:text="UnBounded Ripple Button" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:background="@drawable/ripple_custom3"
        android:text="Masked Ripple Button" />
</LinearLayout>
```

Ripple Effect

- ripple_custom1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="@android:color/holo_orange_light">
    <item
        android:id="@android:id/background"
        android:drawable="@drawable/ic_launcher" />
</ripple>
```

- ripple_custom2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#FC0097">

</ripple>
```

Ripple Effect

ripple_custom3.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#008FFC">
    <item android:id="@android:id/mask">
        <shape android:shape="rectangle">
            <solid android:color="#008FFC" />
            <corners android:radius="4dp" />
        </shape>
    </item>
    <item android:id="@android:id/background">
        <shape android:shape="rectangle">
            <gradient
                android:angle="90"
                android:endColor="@android:color/holo_orange_light"
                android:startColor="@android:color/holo_orange_light"
                android:type="linear" />
            <corners android:radius="4dp" />
        </shape>
    </item>
</ripple>
```