



CHAP 6. 이벤트 처리(1)

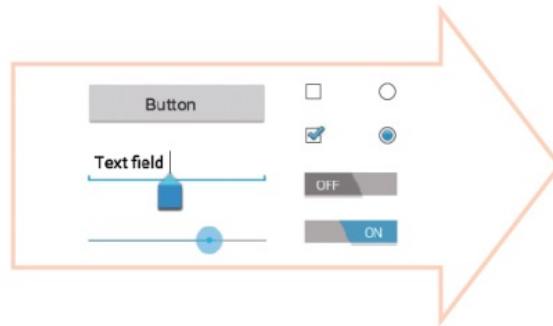
1

입력 위젯

- 버튼, 텍스트 필드, 시크 바, 체크 박스, 줌 버튼, 토글 버튼



사용자



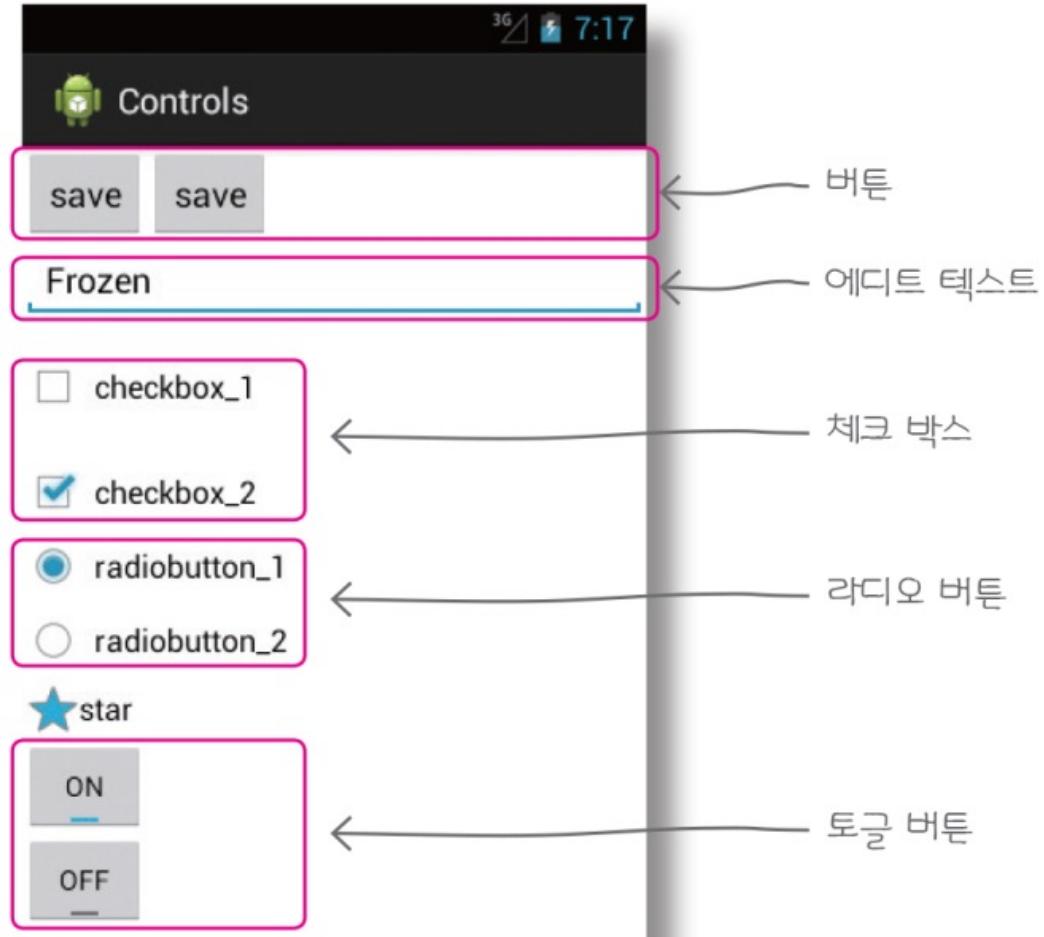
입력 위젯



입력 위젯의 종류

위젯	설명	관련 클래스
Button	어떤 동작을 수행하기 위하여 사용자가 누를 수 있고 클릭할 수 있는 푸시 버튼	Button
Text field	편집이 가능한 텍스트 필드. 자동 완성 기능을 제공하려면 autoCompleteTextView를 사용한다.	EditText, AutoCompleteTextView
Checkbox	사용자에 의하여 토글될 수 있는 on/off 스위치. 사용자가 그룹에서 여러 가지 옵션을 동시에 선택할 수 있게 하려면 체크 박스를 사용한다.	CheckBox
Radio button	체크 박스와 비슷하지만 그룹에서 하나의 옵션만 선택할 수 있다.	RadioGroup RadioButton
Toggle button	라이트 인디케이터가 있는 on/off 버튼	ToggleButton
Spinner	사용자가 여러 값 중에서 하나를 선택할 수 있는 드롭 다운 리스트	Spinner
Pickers	up/down 버튼이나 스와이프 제스처를 통하여 하나의 값을 선택하는 대화 상자. 날짜를 선택하려면 DatePicker를 사용한다. 시간을 선택하려면 TimePicker를 사용한다.	DatePicker TimePicker

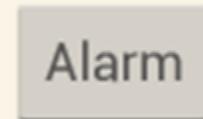
입력 위젯



버튼

○ 텍스트 버튼

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/button_text"  
    ... />
```



○ 이미지 버튼

```
<ImageButton  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/button_icon"  
    ... />
```

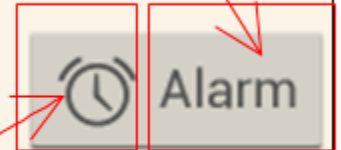
이미지 파일
이름



버튼

- 텍스트와 이미지를 동시에 가지는 버튼

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/button_text"  
    android:drawableLeft="@drawable/button_icon"  
    ... />
```

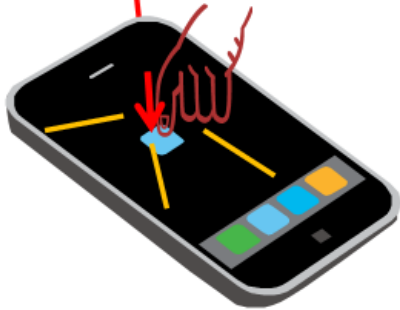


이미지 파일
이름

버튼의 이벤트 처리

```
<?xml version="1.0" encoding="utf-8"?>
<Button
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="sendMessage"
    android:text="@string/button_send" />
```

사용자가 클릭하면 호출된다.



XML 파일에 적었던 메소드를 구현해주면 된다.

```
public class MainActivity extends Activity {
    @Override
    public void onCreate(...) {
        //...
    }
    public void sendMessage(View view)
    {
        //...
    }
}
```

이벤트를 처리하는 가장 간단한 방법

- 레이아웃 안의 <Button> 요소에 **onClick** 속성을 추가

```
<?xml version="1.0" encoding="utf-8"?>
<Button xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:text="@string/button_send" />
```

onClick 속성에 이벤트를 처리하는 메소드 이름을 적는다.

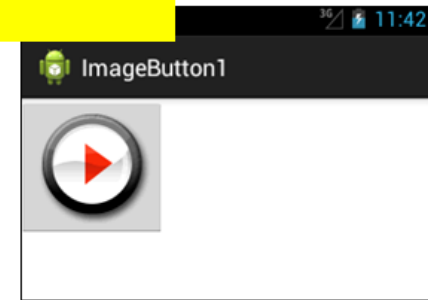
<IMAGEBUTTON> 태그를 사용한 이미지 버튼

main.xml

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

```
    <ImageButton
        android:id="@+id/imageButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:src="@drawable/mybutton" />
```

이미지 버튼을 정의한다.



XML로 이벤트 처리 메소드를 지정하자. 이것이 가장 간단하다.

```
</LinearLayout>
```

이벤트 처리 코드

ImageButtonActivity.java

```
package kr.co.company.imagebutton1;  
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.
```

```
public class ImageButtonActivity extends ActionBarActivity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.main);
```

```
    }
```

이미지 버튼이 눌리면 토스트 메시지를 출력한다.

```
    public void onClick(View target) {
```

```
        Toast.makeText(getApplicationContext(), "버튼이 눌러졌습니다",
```

```
            Toast.LENGTH_SHORT).show();
```

```
    }
```

```
}
```

이미지와 텍스트를 동시에 표시 (실습)

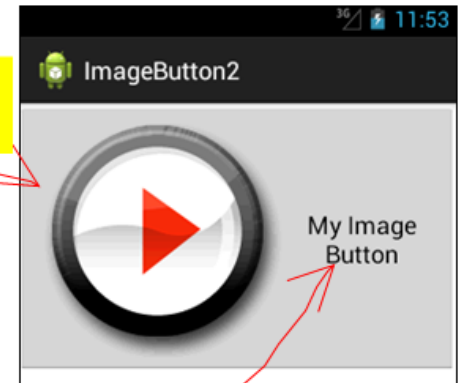
main.xml

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

```
<Button
    android:id="@+id/imageButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawableLeft="@drawable/mybutton"
    android:onClick="onClick"
    android:text="My Image Button" />
```

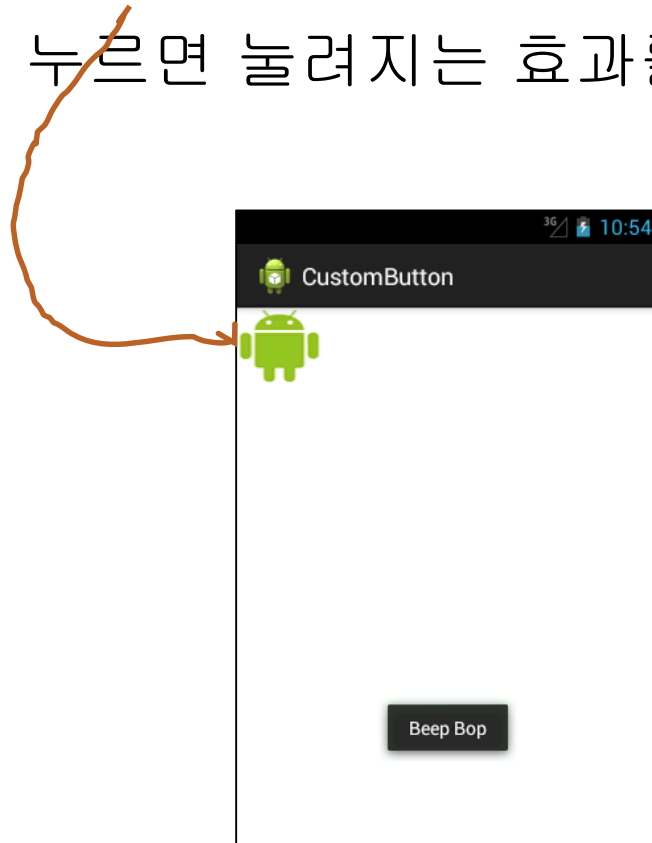
이미지가 왼쪽에 표시된다.

```
</LinearLayout>
```



커스텀 버튼


- 버튼 위에 텍스트 대신에 이미지가 그려져 있는 버튼
- 버튼을 누르면 눌려지는 효과를 발생하는 버튼



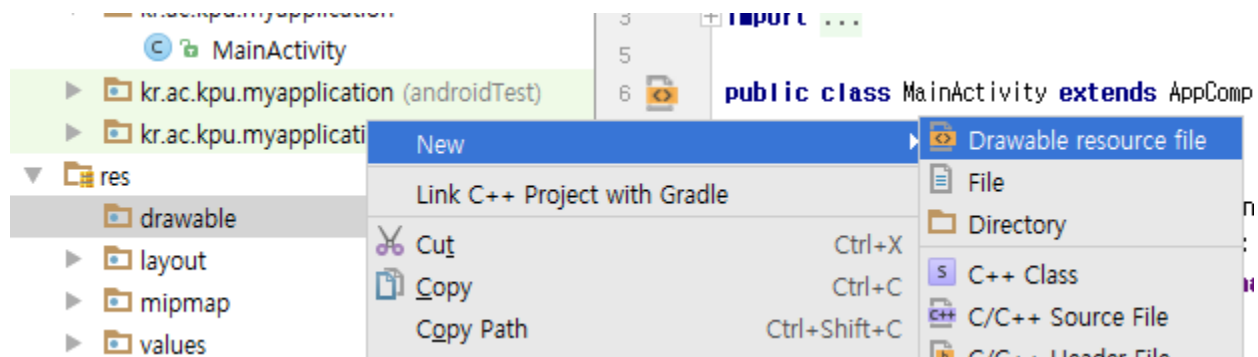
커스텀 버튼 정의

- XML로 버튼에 사용되는 이미지를 등록한다.

/res/drawable/android_button.xml



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



레이아웃 파일

- 레이아웃 파일에 버튼을 정의한다.

/res/layout/main.xml

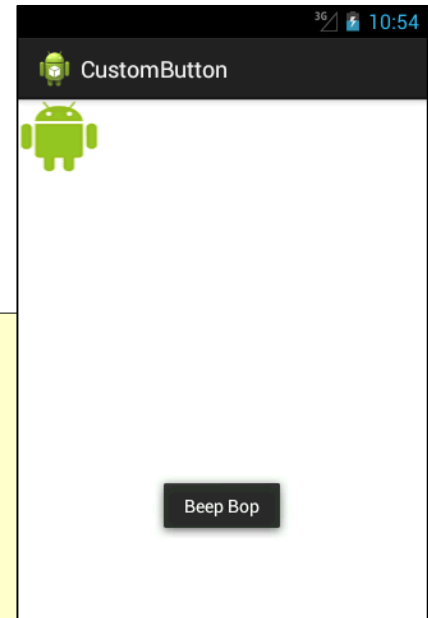
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/android_button"
        android:padding="10dp" />

</LinearLayout>
```

커스텀 버튼 (실습)

```
public class MainActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        final Button button = (Button) findViewById(R.id.button);  
        button.setOnClickListener(new OnClickListener() {  
            public void onClick(View v) {  
                Toast.makeText(getApplicationContext(), "Beep Bop",  
                    Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```



폴링과 이벤트 구동 방식



안드로이드에서의 이벤트 처리 방법

- XML 파일에 이벤트 처리 메소드를 등록하는 방법
 - 가장 쉬운 방법 <- 권장
 - 앞에서 살펴보았음!
- 이벤트 처리 객체를 생성하여 컴포넌트에 등록
 - 일반적인 방법
- 뷰 클래스의 이벤트 처리 메소드를 재정의
 - 커스텀 뷰를 작성하는 경우: (예) 게임

이벤트 처리 객체 사용



사용자가 클릭하면 호출된다.

```
Class MyClass
{
    class Listener implements OnClickListener {
        public void onClick(View v) {
            ...
        }
    }
    ...
    Listener lis = new Listener();
    button.setOnClickListener(lis);
    ...
}
```

버튼에 붙은 리스너 객체가 이벤트를 처리한다.

이벤트 리스너

}

```
class MyClass
```

```
{
```

```
    class Listener implements OnClickListener {
```

```
        public void onClick(View v){
```

```
            ...
```

```
        }
```

```
    }
```

```
    ...
```

```
    Listener lis = new Listener();
```

```
    button.setOnClickListener(lis);
```

```
    ...
```

```
}
```

리스너 인터페이스를
구현한 클래스 정의

이벤트 리스너
객체 생성

버튼에 이벤트
리스너 객체를 등록

리스너의 종류

리스너	콜백 메소드	설명
View.OnClickListener	<code>onClick()</code>	사용자가 어떤 항목을 터치하거나 내비게이션 키나 트랙볼로 항목으로 이동한 후에 엔터키를 눌러서 선택하면 호출된다.
View.OnLongClickListener	<code>onLongClick()</code>	사용자가 항목을 터치하여서 일정 시간 동안 그대로 누르고 있으면 발생한다.
View.OnFocusChangeListener	<code>onFocusChange()</code>	사용자가 하나의 항목에서 다른 항목으로 포커스를 이동할 때 호출된다.
View.OnKeyListener	<code>onKey()</code>	포커스를 가지고 있는 항목 위에서 키를 눌렀다가 놓았을 때 호출된다.
View.OnTouchListener	<code>onTouch()</code>	사용자가 터치 이벤트로 간주되는 동작을 한 경우에 호출된다.

리스너 객체를 생성하는 방법

- 리스너 클래스를 내부 클래스로 정의한다.
- 리스너 클래스를 무명 클래스로 정의한다.
- 리스너 인터페이스를 액티비티 클래스에 구현한다.

가장 많이 사용되는 방법!

무명 클래스로 이벤트를 처리하는 예제

ButtonEvent2Activity.java

```
package kr.co.company.buttonevent2;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class ButtonEvent2Activity extends ActionBarActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button button = (Button) findViewById(R.id.button);

        button.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {

                Toast.makeText(getApplicationContext(), "버튼이
                눌러졌습니다", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

클릭 리스너를 구현하는
무명 클래스를 정의하
고, 객체를 생성하여 버
튼에 등록한다.
한 곳에서 이벤트 처리
와 관련된 모든 코드가
작성되는 장점이 있다.

무명 클래스

- 클래스 몸체는 정의되지만 이름이 없는 클래스이다.
- 무명 클래스는 클래스를 정의하면서 동시에 객체를 생성하게 된다.

- 이름이 있는 클래스의 경우

```
class MyClass implements OnClickListener { ... }  
obj = new MyClass();
```

- 무명 클래스의 경우

```
obj = new OnClickListener() { ... };
```

코드 분석

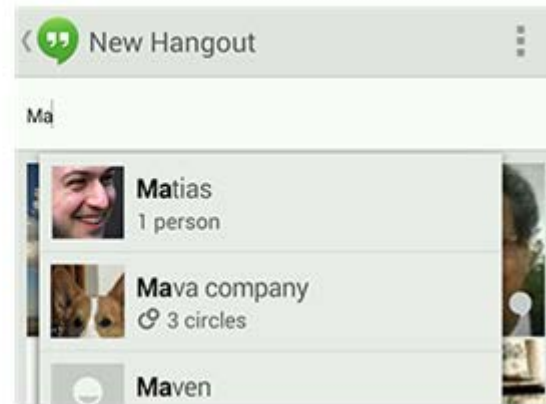
무명 클래스 사용

```
...
public class ButtonEvent2Activity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(), "버튼
                눌러졌습니다", Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```


텍스트 필드

- 텍스트 필드(text field)를 사용하면 사용자가 앱에 텍스트를 타이핑하여 입력할 수 있다.
- 단일 라인이거나 멀티 라인일 수 있다.

PHONE	
1 650-123	MOBILE
EMAIL	
Email	HOME
ADDRESS	
Address	HOME



키보드 종류 지정

<EditText

```
    android:id="@+id/email_address"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:hint="@string/email_hint"  
    android:inputType="textEmailAddress" />
```

이메일 형태의 입력을 받는다.



(textEmailAddress 입력 타입)



(phone 입력 타입)

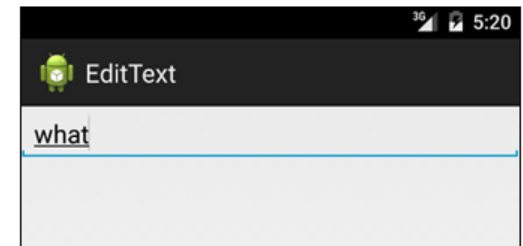
에디트 텍스트

main.xml

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

```
    <EditText
        android:id="@+id/search"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Search"
        android:imeOptions="actionSend"
        android:inputType="text" />
```

에디트 텍스트



```
</LinearLayout>
```

에디트 텍스트의 이벤트 처리

EditTextActivity.java

```
package kr.co.company.editttext;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class EditTextActivity extends ActionBarActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final EditText editText = (EditText) findViewById(R.id.search);
        editText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
                boolean handled = false;
                if (actionId == EditorInfo.IME_ACTION_SEND) {
                    Toast.makeText(getApplicationContext(), editText.getText(),
                        Toast.LENGTH_SHORT).show();
                    handled = true;
                }
                return handled;
            }
        });
    }
}
```

여기에 final이 붙는 이유는
참고사항에 있음!

사용자가 "Send" 버튼을
누르면 화면에 토스트
메시지를 표시한다.

변수에 final을 붙여서 상수로 만드는 이유

앞의 소스에서 editText 지역변수가 final로 선언된 이유는 내부의 무명 클래스에서 이 변수를 참조하기 때문이다. 내부 클래스에서 접근하는 외부 지역변수는 반드시 final이어야 한다. 아니면 그 변수가 가리키는 객체가 바뀌지 않는다는 보장이 있어야 한다. 간단한 예제로 설명하여 보자.

```
public void onCreate(...) {  
    EditText editText = (EditText) findViewById(R.id.edittext1);  
    ...  
    editText.setOnKeyListener(new OnKeyListener() {  
        public boolean onKey(...) {  
            s = edittext.getText();  
        }  
    });  
    editText = (EditText) findViewById(R.id.edittext2);  
}
```



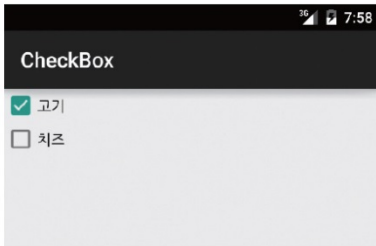
위와 같은 상황에서는 editText는 어떤 객체를 가리키는가? edittext1인가? 아니면 edittext2인가? 생성된 이벤트 리스너 객체는 onCreate()가 종료된 후에도 살아 있을 수 있다. 리스너 객체 안의 이벤트 처리 메소드는 이벤트가 발생하면 호출된다. 이런 혼란을 막기 위하여 editText는 반드시 final로 지정하도록 되어 있다. 만약 final로 지정하지 않으면 다음과 같은 오류 메시지가 표시된다.

Cannot refer to a non-final variable value inside an inner class defined in a different method

체크 박스

- XML로 체크 박스를 선언한다.

main.xml



체크 박스

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

    <CheckBox android:id="@+id/checkbox_meat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="고기"
        android:onClick="onCheckboxClicked" />

    <CheckBox android:id="@+id/checkbox_cheese"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="치즈"
        android:onClick="onCheckboxClicked" />

</LinearLayout>
```


체크박스의 이벤트 처리

```
public class CheckBoxActivity extends ActionBarActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        | super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
    public void onCheckboxClicked(View view) {  
        boolean checked = ((CheckBox) view).isChecked();  
  
        switch(view.getId()) {  
            case R.id.checkbox_meat:  
                if (checked)  
                    Toast.makeText(getApplicationContext(), "고기 선택",  
                                   Toast.LENGTH_SHORT).show();  
                else  
                    Toast.makeText(getApplicationContext(), "고기 선택 해제",  
                                   Toast.LENGTH_SHORT).show();  
                break;  
            case R.id.checkbox_cheese:  
                if (checked)  
                    Toast.makeText(getApplicationContext(), "치즈 선택",  
                                   Toast.LENGTH_SHORT).show();  
                else  
                    Toast.makeText(getApplicationContext(), "치즈 선택 해제",  
                                   Toast.LENGTH_SHORT).show();  
                break;  
        }  
    }  
}
```

클릭되면 먼저 `isChecked()` 을 호출하여서 체크 박스의 새로운 상태를 얻는다. 만약 체크 박스가 체크되었으면 "...선택"이라고 하는 토스트 메시지를 표시한다. 그렇지 않으면 "...선택해제"를 표시한다.
`onClick()`으로 전달된 매개 변수 `v`를 `CheckBox`로 형변환한 것에 유의하라. 왜냐하면 `isChecked()` 메소드는 부모 클래스인 `View`에는 없는 메소드이다. 체크 박스는 상태 변경을 스스로 처리하기 때문에 프로그래머는 단지 현재 상태만을 조사하면 된다.

라디오 버튼

- XML로 라디오 버튼을 정의한다.

main.xml

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

    <RadioGroup
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <RadioButton
            android:id="@+id/radio_red"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="onRadioButtonClicked"
            android:text="Red"/>
        <RadioButton
            android:id="@+id/radio_blue"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onClick="onRadioButtonClicked"
            android:text="Blue"/>
    </RadioGroup>
</LinearLayout>
```

라디오 그룹 안에 라디오 버튼을 2개 정의한다.



라디오 버튼의 이벤트 처리

```
public class RadioButtonActivity extends ActionBarActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
  
    public void onRadioButtonClicked(View view) {  
        boolean checked = ((RadioButton) view).isChecked();  
  
        switch(view.getId()) {  
            case R.id.radio_red:  
                if (checked)  
                    Toast.makeText(getApplicationContext(),  
                        ((RadioButton) view).getText(),  
                        Toast.LENGTH_SHORT).show();  
                break;  
            case R.id.radio_blue:  
                if (checked)  
                    Toast.makeText(getApplicationContext(),  
                        ((RadioButton) view).getText(),  
                        Toast.LENGTH_SHORT).show();  
                break;  
        }  
    }  
}
```

라디오 버튼의 이벤트를 처리한다.

버튼의 체크 여부를 얻는다.

토글 버튼

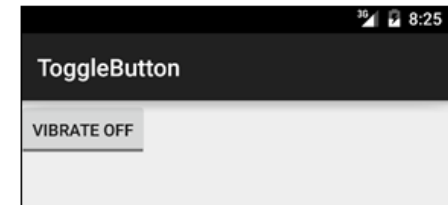
- XML로 라디오 버튼을 정의한다.

main.xml

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

    <ToggleButton
        android:id="@+id/togglebutton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textOn="Vibrate on"
        android:textOff="Vibrate off"
        android:onClick="onToggleClicked"
    />

</LinearLayout>
```



토글 버튼

토글 버튼

ToggleButtonActivity.java

```
package kr.co.company.togglebutton;
```

```
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.
```

```
public class ToggleButtonActivity extends ActionBarActivity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.main);
```

```
    }
```

```
    public void onToggleClicked(View view) {
```

```
        // Is the toggle on?
```

```
        boolean on = ((ToggleButton) view).isChecked();
```

```
        if (on) {
```

```
            Toast.makeText(getApplicationContext(), "Checked",
```

```
                Toast.LENGTH_SHORT).show();
```

```
        } else {
```

```
            Toast.makeText(getApplicationContext(), "Not checked",
```

```
                Toast.LENGTH_SHORT).show();
```

```
        }
```

```
    }
```

```
}
```

레이팅바

- XML로 라디오 버튼을 정의한다.

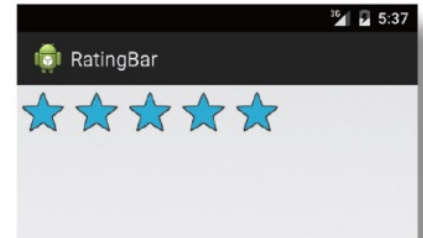
main.xml

<LinearLayout

```
xmlns:android="http://schemas.android.com/apk/res/android"  
android:layout_width="match_parent"  
android:layout_height="match_parent" >
```

<RatingBar

```
android:id="@+id/ratingbar"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:numStars="5"  
android:stepSize="1.0" />
```

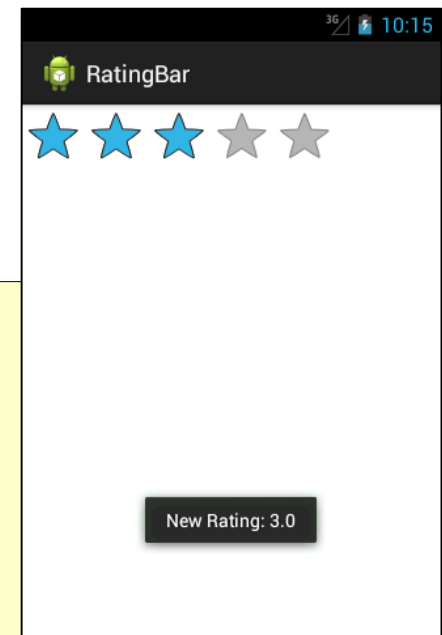


레이팅 바

</LinearLayout>

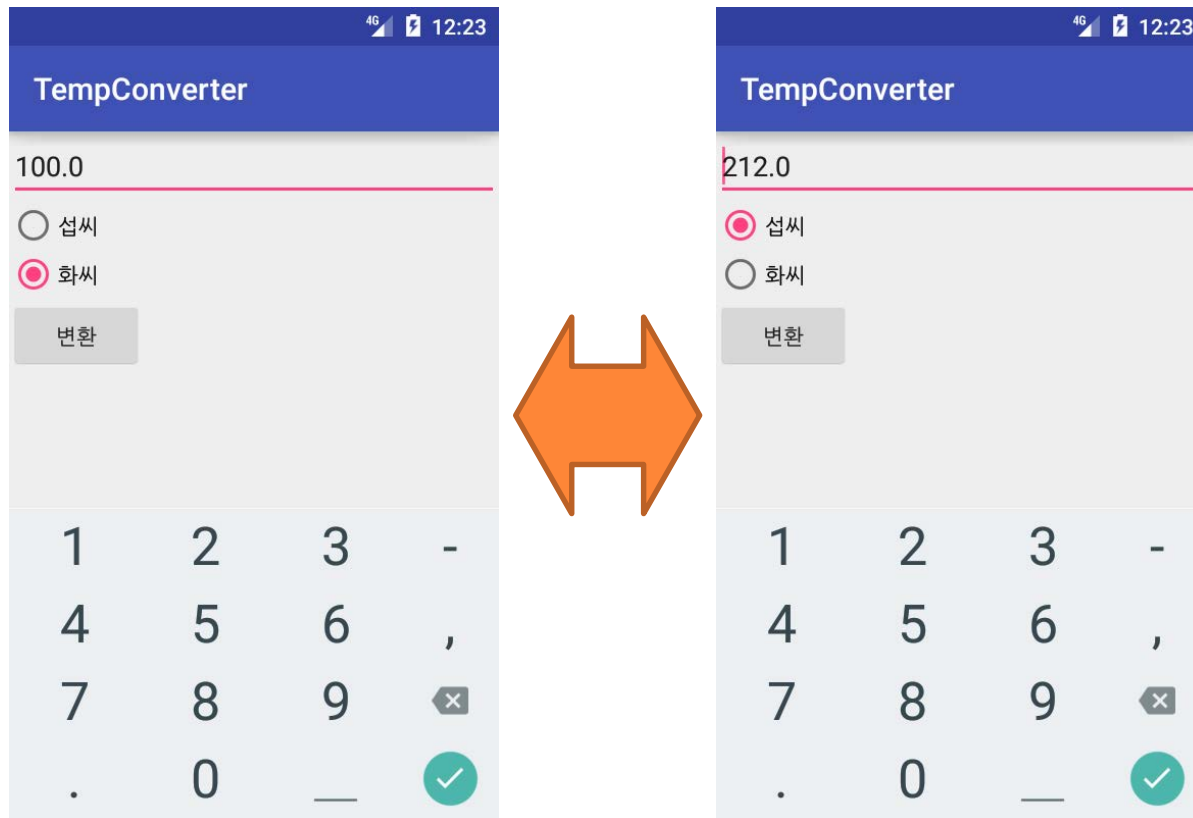
레이팅바 (심심)

```
...
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final RatingBar ratingbar = (RatingBar) findViewById(R.id.ratingbar);
        ratingbar.setOnRatingBarChangeListener(new OnRatingBarChangeListener() {
            public void onRatingChanged(RatingBar ratingBar, float rating,
                boolean fromUser) {
                Toast.makeText(getApplicationContext(),
                    "New Rating: " + rating, Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```



온도 변환기 앱 작성

뒷 페이지의 코드가 동작하도록 화면을 디자인 하시오.



```

public class MainActivity extends AppCompatActivity {
    private EditText text;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        text = (EditText) findViewById(R.id.edit_input);
    }

    public void onClicked(View view){
        switch (view.getId()){
            case R.id.btn_change:
                RadioButton celsiusButton = (RadioButton) findViewById(R.id.celsius);
                RadioButton fahrenheitButton = (RadioButton) findViewById(R.id.fahrenheit);
                if (text.getText().length() == 0){
                    Toast.makeText(this, "정확한 값을 입력하십시오.", Toast.LENGTH_LONG).show();
                    return;
                }

                float inputValue = Float.parseFloat(text.getText().toString());
                if (celsiusButton.isChecked()){
                    text.setText(String.valueOf(convertFahrenheitToCelsius(inputValue)));
                    celsiusButton.setChecked(false);
                    fahrenheitButton.setChecked(true);
                } else {
                    text.setText(String.valueOf(convertCelsiusToFahrenheit(inputValue)));
                    fahrenheitButton.setChecked(false);
                    celsiusButton.setChecked(true);
                }
            }
        }

        private float convertFahrenheitToCelsius(float fahrenheit){
            return ((fahrenheit - 32) * 5 / 9);
        }

        private float convertCelsiusToFahrenheit(float celsius){
            return ((celsius * 9) / 5) + 32;
        }
    }
}

```