

EDCAN 수업 자료

Android 기초 02

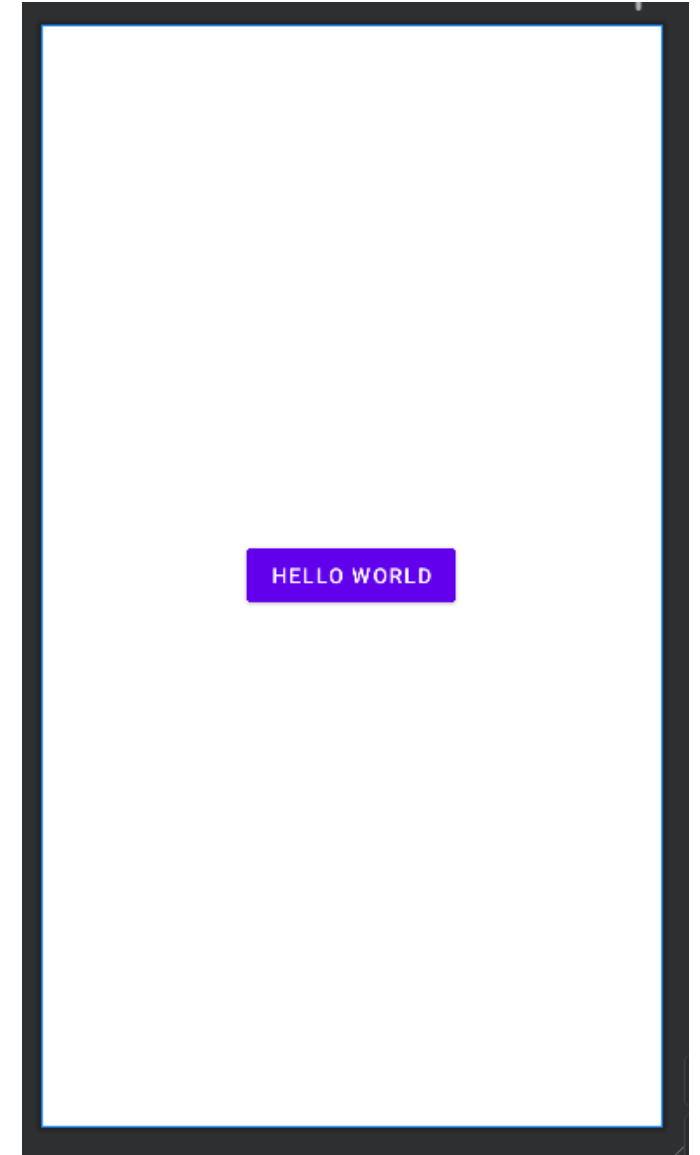
배우는 것들

**Button, Radio Button,
CheckBox, SeekBar, ImageView,
EditText, Drawable, 테마 설정,
Event, Activity, Intent,
RecyclerView**

Button

- 화면에 버튼을 추가한다.
- 안드로이드 내부적으로는 Button 클래스로 구현 돼있다.
- Button 클래스는 TextView 클래스를 상속받는다.
TextView에서 사용할 수 있는 모든 것을 사용할 수 있다
- XML에서 Button 태그를 추가하면 된다.

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
  
    android:text="Hello World"  
  
    app:layout_constraintTop_toTopOf="parent"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
>
```



Button - enabled속성

- 버튼의 활성화 여부를 표시
- 값으로 true가 들어오면 활성화 false가 들어오면 비활성화
- 기본값은 true

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

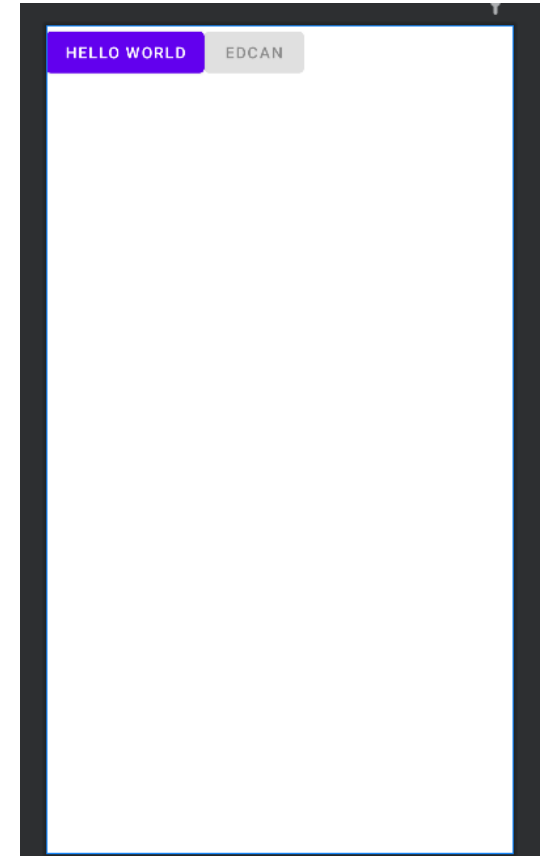
    android:text="Hello World"

/>

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:enabled="false"

    android:text="EDCAN"
/>
```



Button - textAllCaps 속성 (사실 TextView의 속성이다.)

- 안드로이드 기본 테마에서 모든 버튼의 내용은 대문자로 나온다.
- 모두 대문자를 활성화 할지 정하는 속성
- 값으로 true가 들어오면 활성화 false가 들어오면 비활성화
- 버튼에서 기본값은 true

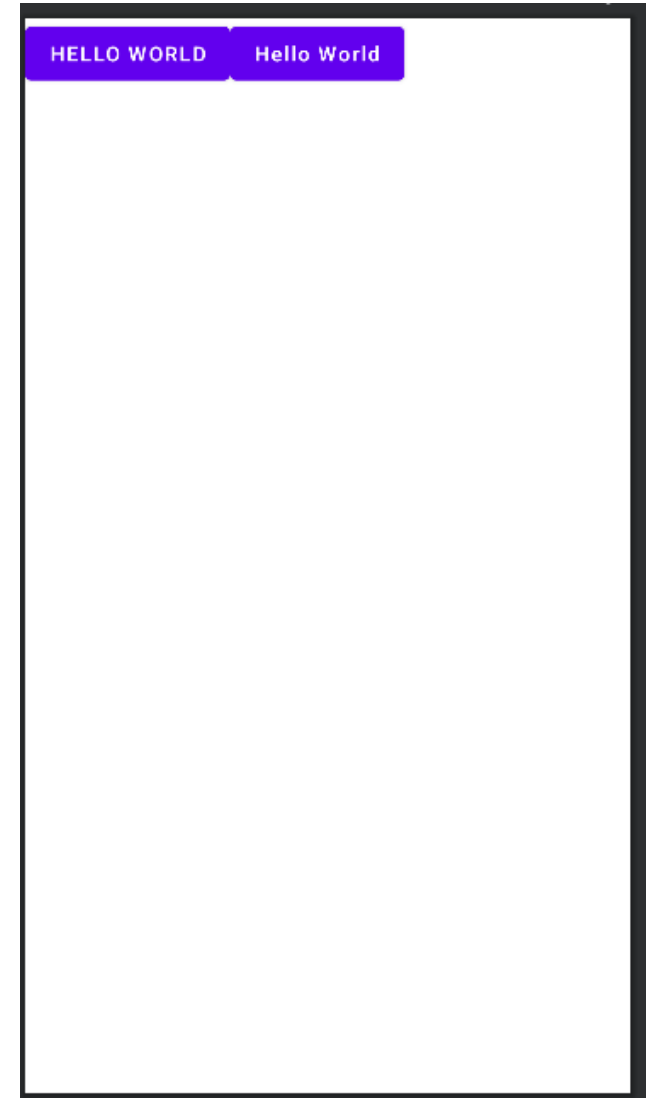
```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:text="Hello World"
/>

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:textAllCaps="false"

    android:text="Hello World"
/>
```



Button - textAllCaps 속성 (사실 TextView의 속성이다.)

- 안드로이드 기본 테마에서 모든 버튼의 내용은 대문자로 나온다.
- 모두 대문자를 활성화 할지 정하는 속성
- 값으로 true가 들어오면 활성화 false가 들어오면 비활성화
- 버튼에서 기본값은 true

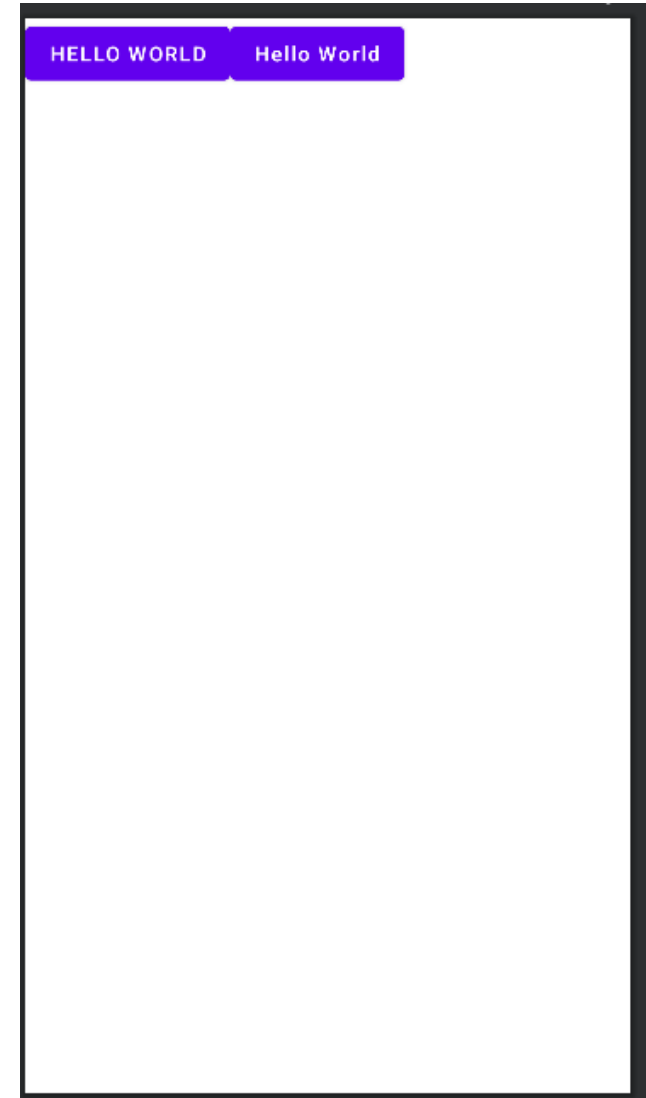
```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:text="Hello World"
/>

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:textAllCaps="false"

    android:text="Hello World"
/>
```



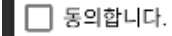
CheckBox

- 화면에 체크박스를 추가한다.
- Button 클래스를 상속 받아서 만들어진다.
- Button의 속성과 TextView의 속성 모두 사용 가능하다.
- 체크 여부는 `isChecked`을 통해서 Boolean형을 가져온다.

```
<CheckBox
    android:id="@+id/cb_main_consent"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:text="동의합니다."
/>
```

```
val 체크박스 = findViewById<CheckBox>(R.id.cb_main_consent)
val 동의했니 = 체크박스.isChecked
```



☐ 동의합니다.

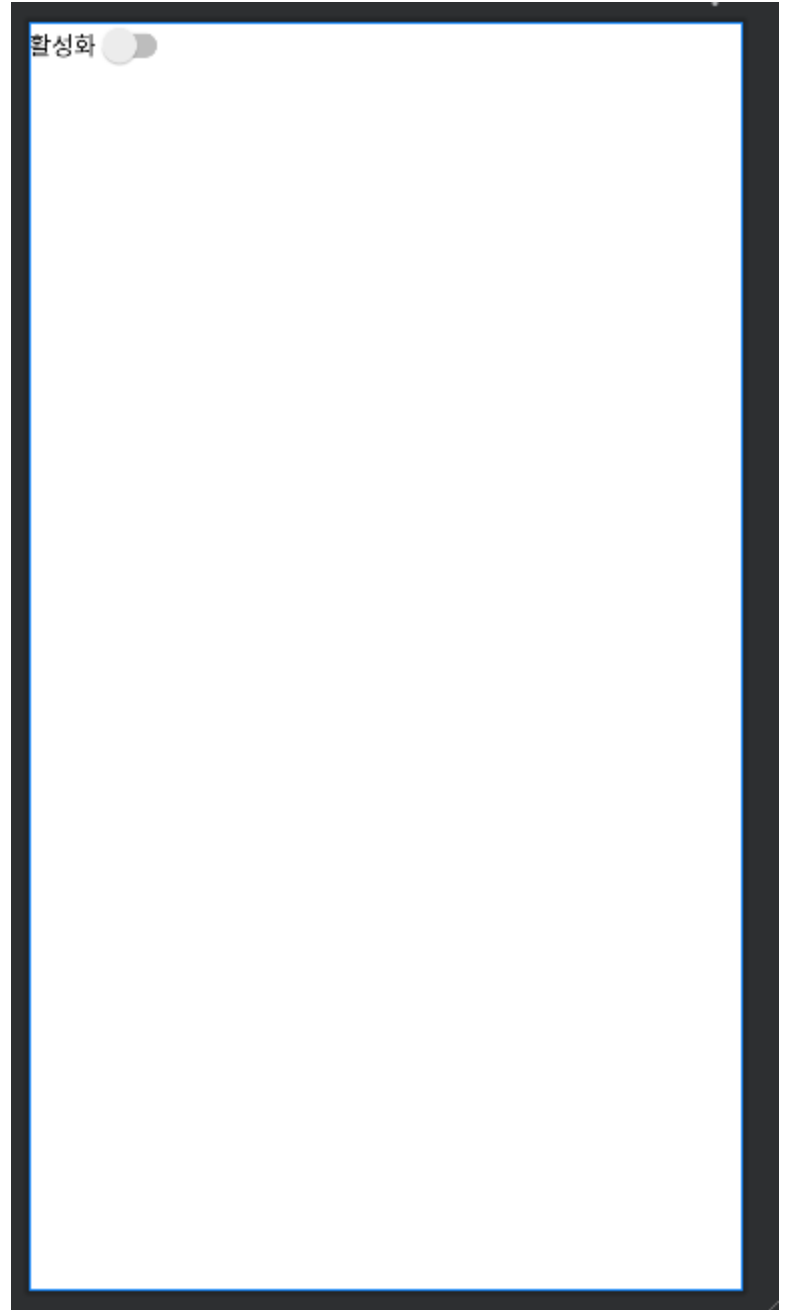
Switch

- 화면에 스위치 버튼을 추가한다.
- Button 클래스를 상속 받아서 만들어진다.
- Button의 속성과 TextView의 속성 모두 사용 가능하다.
- 체크 여부는 `isChecked`을 통해서 Boolean형을 가져온다.

```
<Switch
    android:id="@+id/sw_main_activate"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:text="활성화"
/>
```

```
val 스위치버튼 = findViewById<Switch>(R.id.sw_main_activate)
val 활성화했니 = 스위치버튼.isChecked
```



RadioButton & RadioGroup

- 화면에 라디오 버튼을 추가한다.
- RadioGroup은 LinearLayout을 상속받는다.
- RadioButton은 RadioGroup내부에 있어야 한다.
- 하나의 RadioGroup안에선 하나의 RadioButton 만 선택할 수 있다.
- checked 속성으로 기본적으로 선택을 시킬 수 있다.
- 선택 여부는 isChecked를 통해서 Boolean값을 가져온다.
- checkedRadioButtonId를 통해서 선택된 라디오 버튼의 id를 가져온다.



Check Box

Radio Button

Switch

RadioButton & RadioGroup

<RadioGroup

```
android:id="@+id/rg_main_choices"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content">
```

<RadioButton

```
android:id="@+id/rb_main_choice1"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"
```

```
android:text="선택지 1"  
>
```

<RadioButton

```
android:id="@+id/rb_main_choice2"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"
```

```
android:text="선택지 2"  
>
```

<RadioButton

```
android:id="@+id/rb_main_choice3"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"
```

```
android:text="선택지 3"  
>
```

```
</RadioGroup>
```

☐ 선택지 1

☐ 선택지 2

☐ 선택지 3

```
val 그룹 = findViewById<RadioGroup>(R.id.rg_main_choices)  
val 선택지1 = findViewById<RadioButton>(R.id.rb_main_choice1)  
val 선택지2 = findViewById<RadioButton>(R.id.rb_main_choice2)  
val 선택지3 = findViewById<RadioButton>(R.id.rb_main_choice3)
```

```
val 선택했니1 = 선택지1.isChecked  
val 선택했니2 = 선택지2.isChecked  
val 선택했니3 = 선택지3.isChecked
```

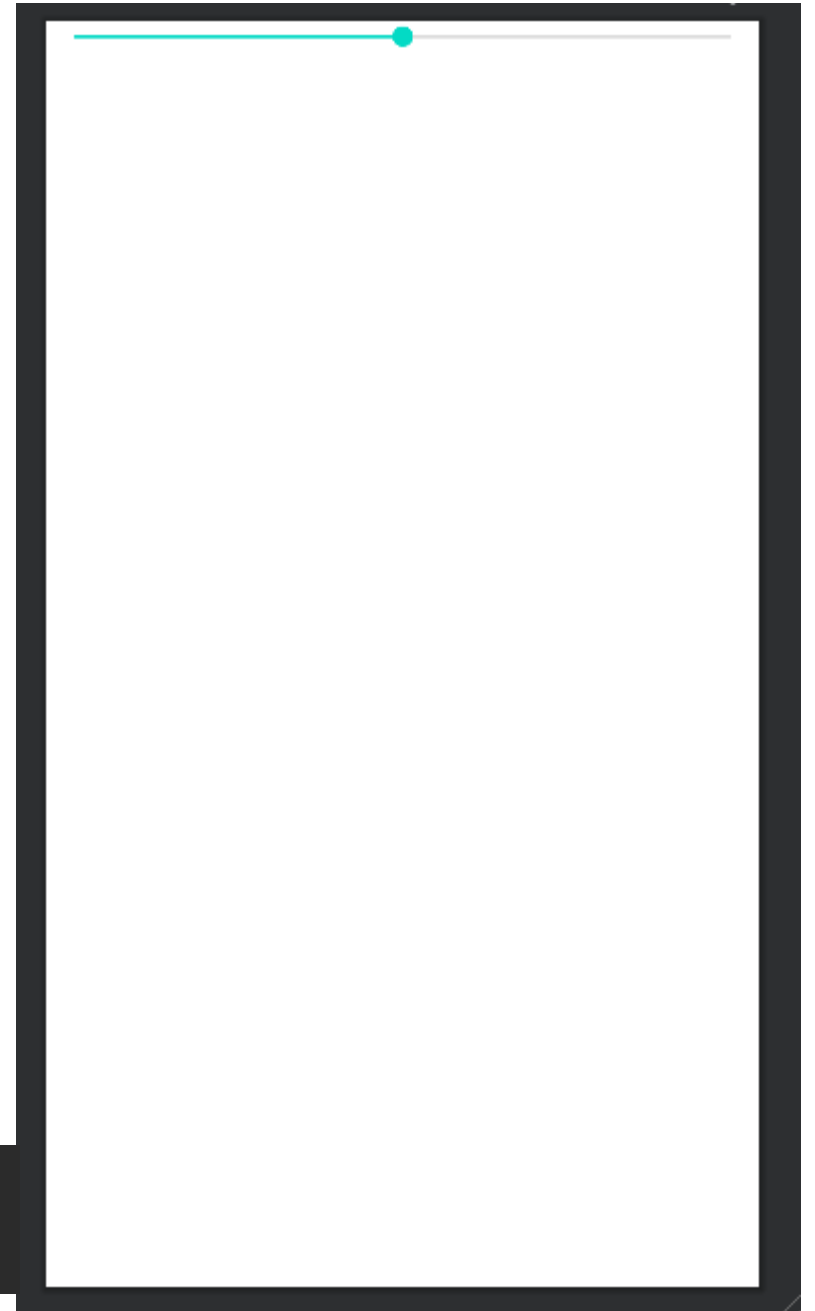
```
val 선택한것 = 그룹.checkedRadioButtonId
```

SeekBar

- 범위 선택 받는 요소를 화면에 추가한다
- min, max 속성으로 범위를 설정한다.
- progress 속성으로 현재 값을 설정한다.

```
<SeekBar  
    android:id="@+id/sb_main_score"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
  
    android:min="0"  
    android:max="100"  
    android:progress="50"  
>
```

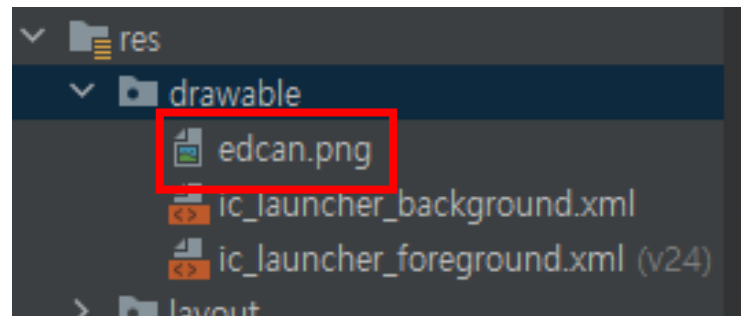
```
val 시크바 = findViewById<SeekBar>(R.id.sb_main_score)  
val 점수 = 시크바.progress
```



ImageView

- 화면에 이미지를 추가한다.
- src 속성으로 추가하려는 이미지를 지정한다.
- 이때 이미지는 res/drawable 폴더 내부에 있어야 한다.

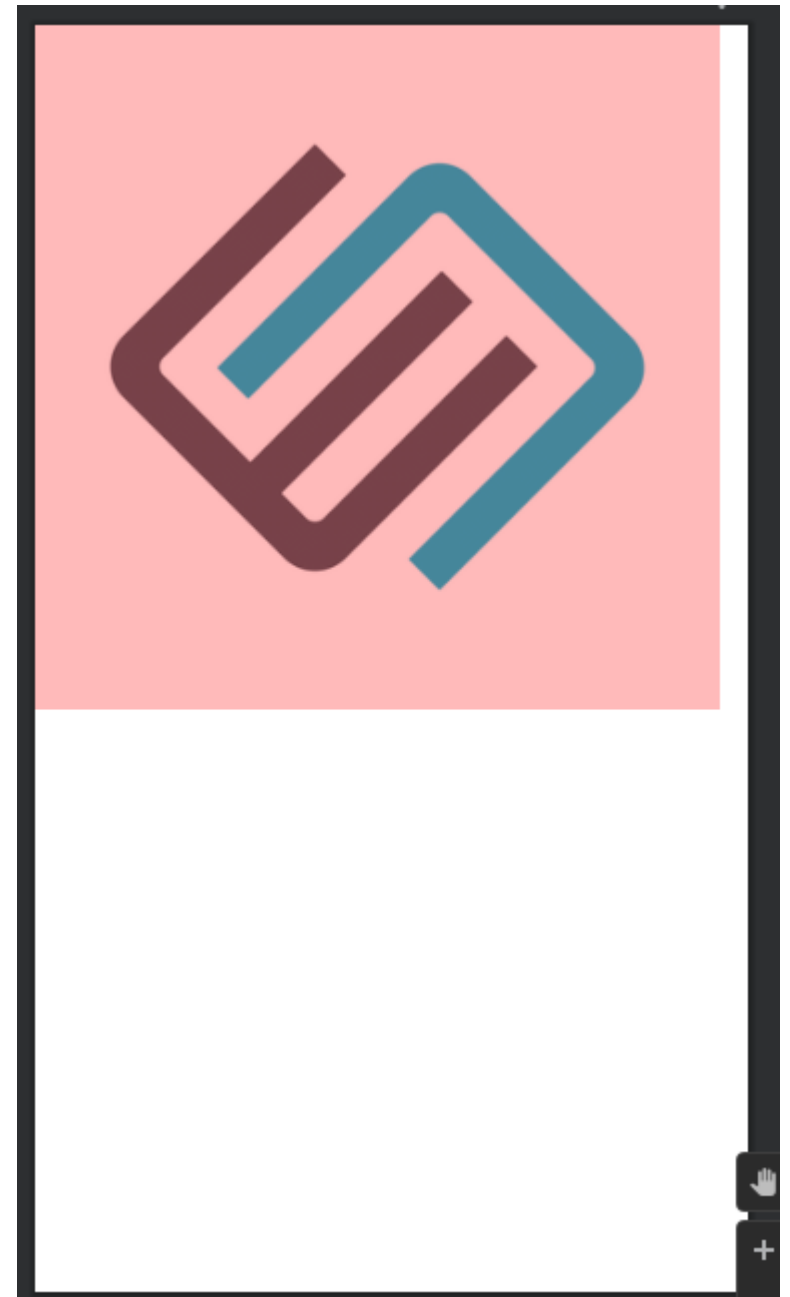
```
<ImageView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
  
    android:src="@drawable/edcan"  
/>
```



ImageView -tint 속성

- 이미지 위에 색상을 추가한다.

```
<ImageView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
  
    android:src="@drawable/edcan"  
  
    android:tint="#45ff0000"  
>
```



EditText

- 사용자에게 입력 받는 뷰
- `text` 속성으로 입력된 값을 가져올 수 있다.
- 입력된 값을 가져오면 `String`형이 아니다.
`String`형으로 변경해줘야 한다.

XML

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

Kotlin

```
val 이름입력창 = findViewById<EditText>(R.id.edt_main_nameInput)  
val 입력된이름 = 이름입력창.text.toString()
```

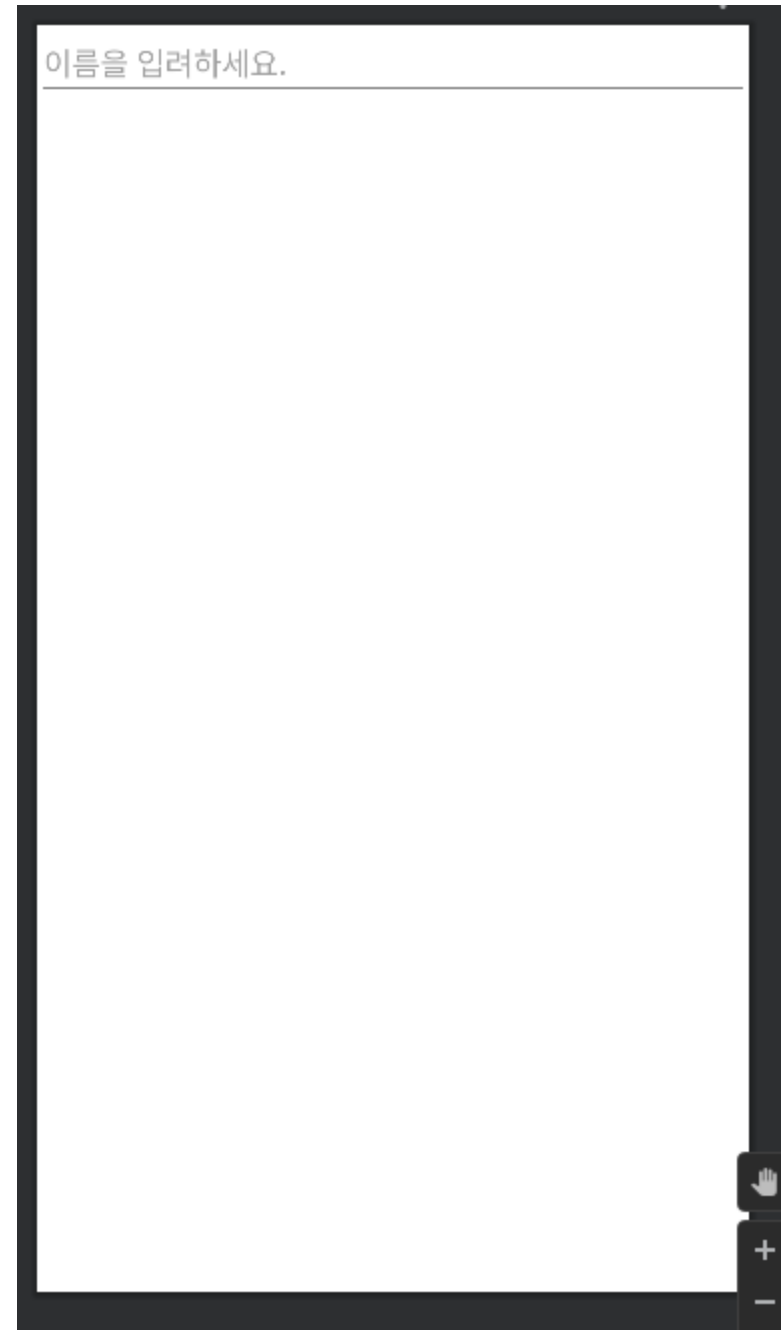


EditText – hint 속성

- 입력요소에 미리보기를 줄 수 있다.
- 입력된 값이 없을 때만 미리보기가 보인다.

XML

```
<EditText  
    android:id="@+id/edt_main_nameInput"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:hint="이름을 입력하세요."  
>
```

A screenshot of an Android application interface. It features a large, empty text input field with a light gray border. At the top left of the field, the text '이름을 입력하세요.' (Please enter your name.) is displayed in a small, light gray font, serving as a hint. The field is set against a dark background.

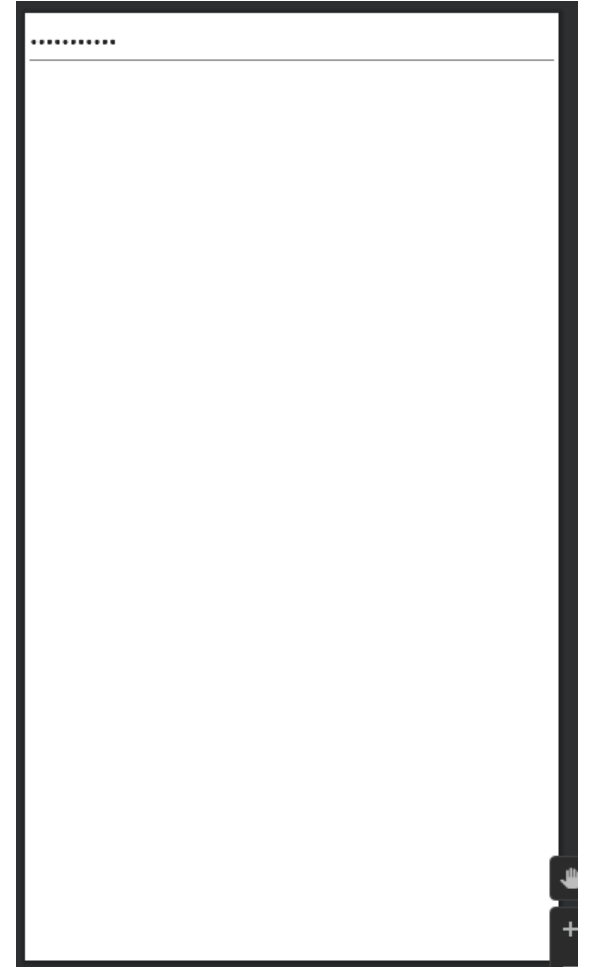
실행 화면

EditText – inputType 속성

- 입력요소의 형식을 지정한다.
- 형식에 따라서 키보드의 모양, 보여지는 입력된 값 등 변경된다.
- 이메일, 비밀번호, 숫자, URI, 전화번호 등 지정 가능하다

XML

```
<EditText  
    android:id="@+id/edt_main_nameInput"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
  
    android:inputType="textPassword"  
/>
```



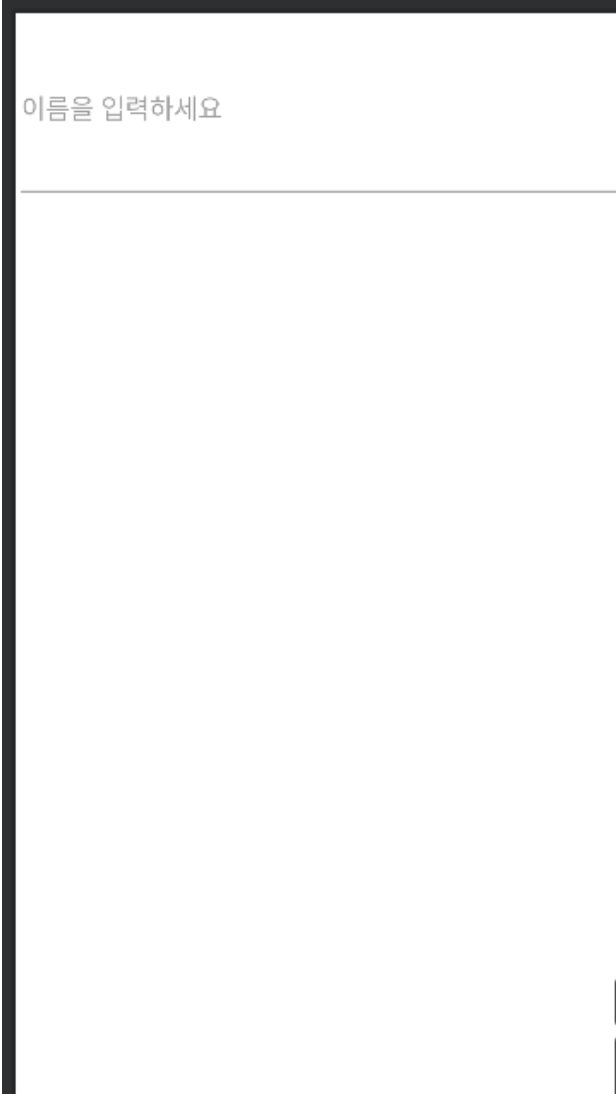
실행 화면

EditText – lines 속성

- 기본적으로 화면에 보여지는 줄 수
- 기본값은 가운데 정렬이 되어있다.
- gravity 속성을 주면 변경 가능하다.

```
<EditText  
    android:id="@+id/edt_main_nameInput"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
  
    android:hint="이름을 입력하세요"  
    android:lines="5"  
>
```

XML



실행 화면

EditText – maxLines

- 화면에 최대로 늘어나는 줄 수
- 화면에 보이는 줄 수만 지정하고 추가로 입력이 가능하다.

```
<EditText  
    android:id="@+id/edt_main_nameInput"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:maxLines="10"  
>
```

XML

EditText – maxLength 속성

- 입력되는 값의 최대 글자수를 지정한다.

```
<EditText  
    android:id="@+id/edt_main_nameInput"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
  
    android:maxLength="100"  
>
```

XML

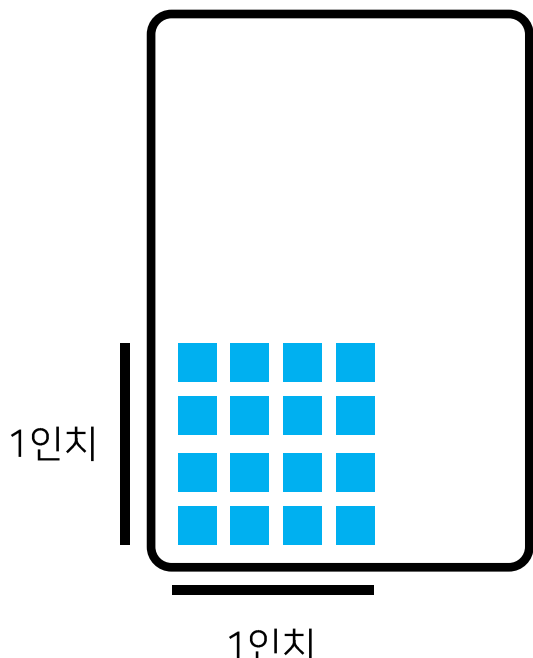
Drawable과 테마

Android에서 사용되는 단위

DPI : 가로세로1인치 크기의 정사각형에 들어있는 픽셀 수 (= 해상도)

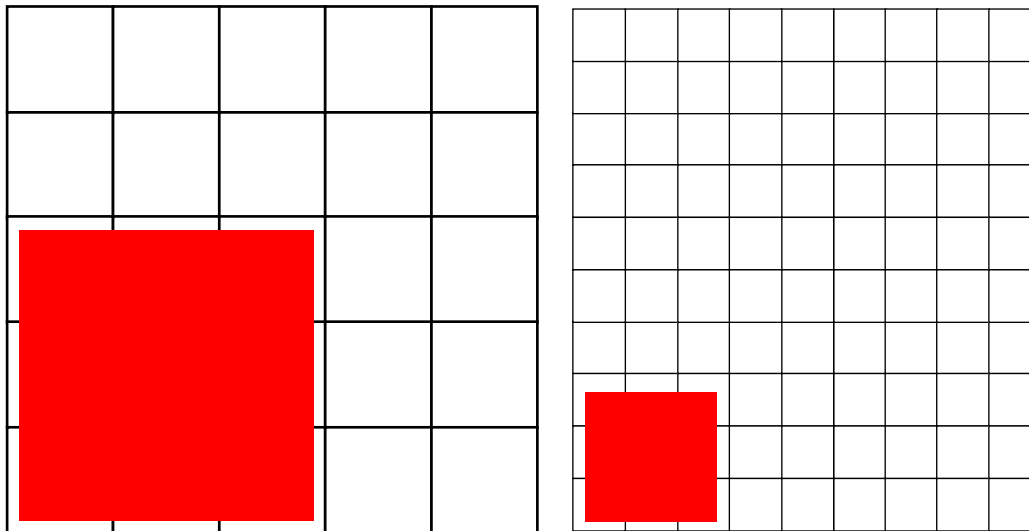
dp : 서로 다른 dpi를 가지고 있는 기기에서 같은 크기를 가지기 위해서 사용되는 단위

sp : 글자 크기를 지정할때 사용한다. 기본값은 dp와 같지만 안드로이드 설정에서 글자 크기를 변경할때 영향을 받는다

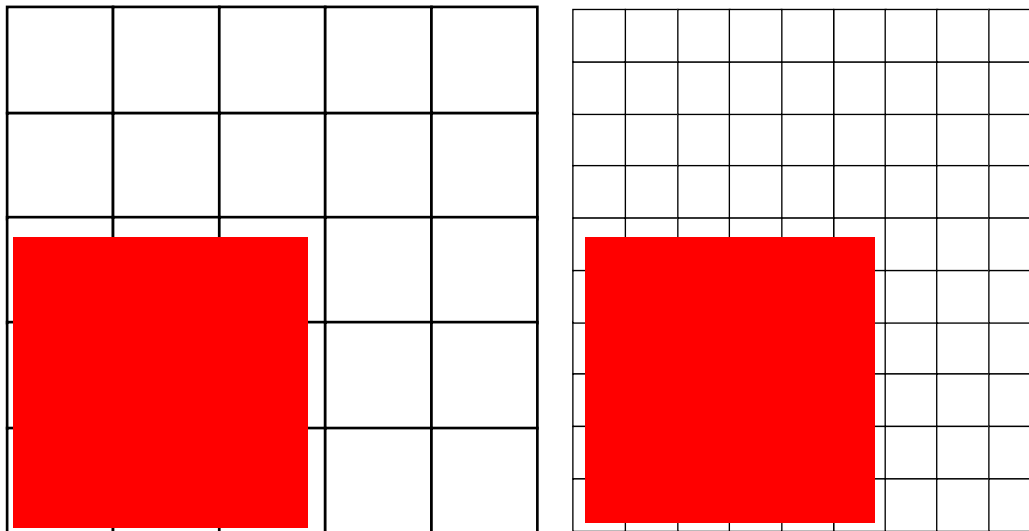


Android에서 사용되는 단위

3px의 정사각형을 만들었을때



3dp의 정사각형을 만들었을때

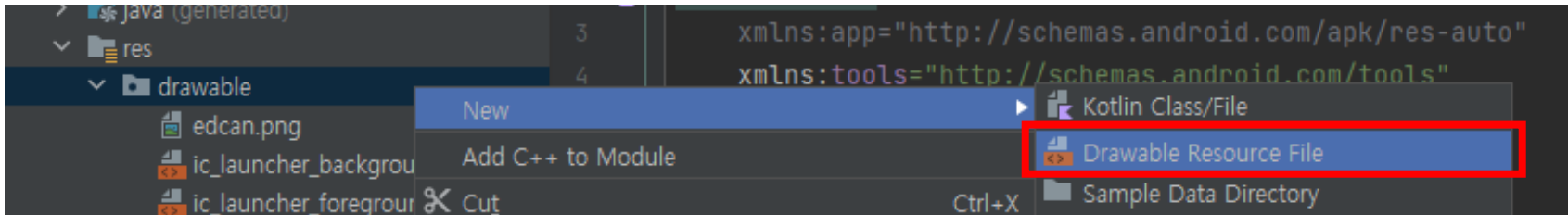


Shape Drawable

- Android에서 다양한 도형을 그릴 때 사용 된다.
- 테두리, 둥근 사각형, 원, 그라데이션 등을 만들 때 사용된다.
- XML파일에 정의한다.

Shape Drawable

1. res/drawable 파일에 XML파일을 만들어준다.



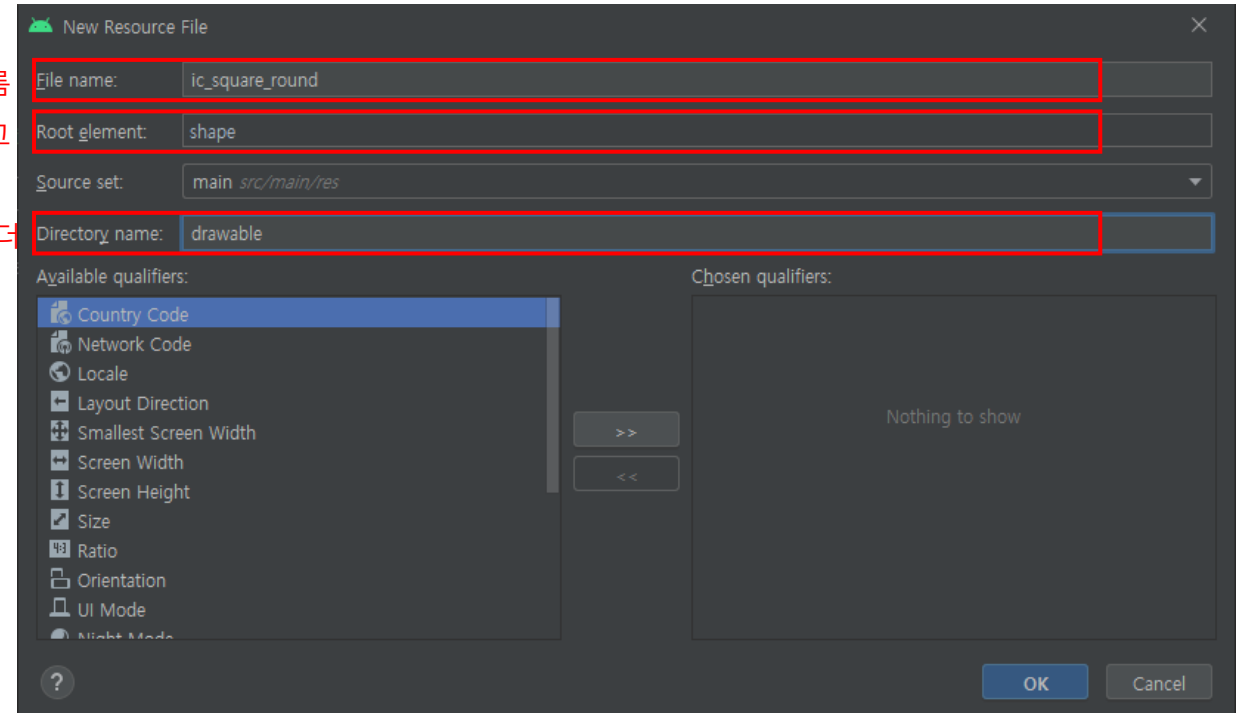
2. 만들 파일의 설정을 해준다.

이때 XML최상위 태그가 shape,
XML을 만들 폴더가 drawable인지 확인한다.

만들 파일의 이름

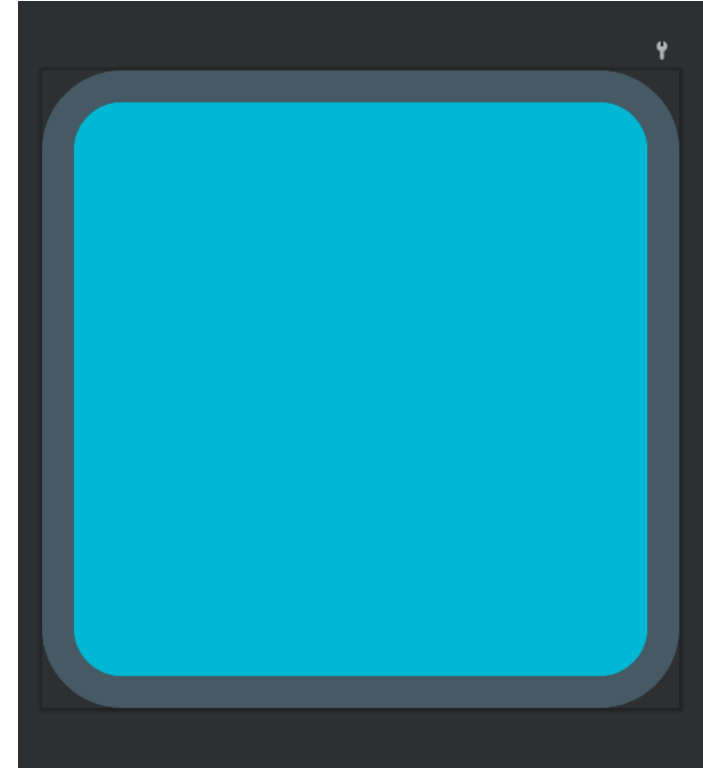
XML의 최상위 태그

XML을 만들 폴더



Shape Drawable

```
<shape xmlns:android="http://schemas.android.com/apk/res/android">  
  <size  
    android:width="100dp"  
    android:height="100dp"  
  />  
  <solid  
    android:color="#00b8d4"  
  />  
  <stroke  
    android:color="#455a64"  
    android:width="5dp"  
  />  
  <corners  
    android:radius="10dp"  
  />  
</shape>
```



size : 도형의 크기

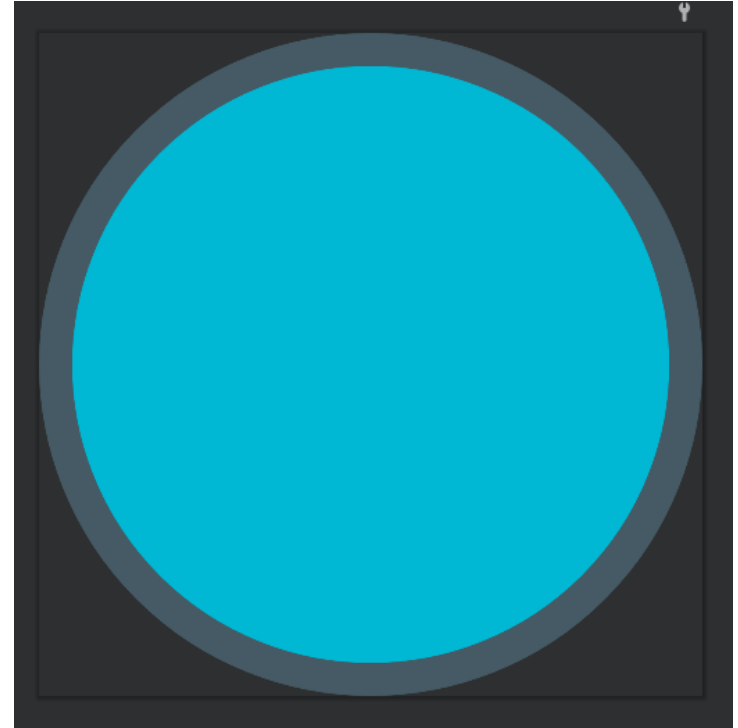
solid : 도형의 채우기

Stroke : 도형의 윤곽선 설정

corners : 도형의 둥근 테두리 설정 (각 모서리에 개별로 설정 가능하다.)

Shape Drawable

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval"
/>
    <size
        android:width="100dp"
        android:height="100dp"
    />
    <solid
        android:color="#00b8d4"
    />
    <stroke
        android:color="#455a64"
        android:width="5dp"
    />
</shape>
```



shape 태그에 shape 속성으로 만들 도형을 설정할 수 있다.

rectangle : 사각형 oval : 타원 line : 선

Shape Drawable

```
<View
```

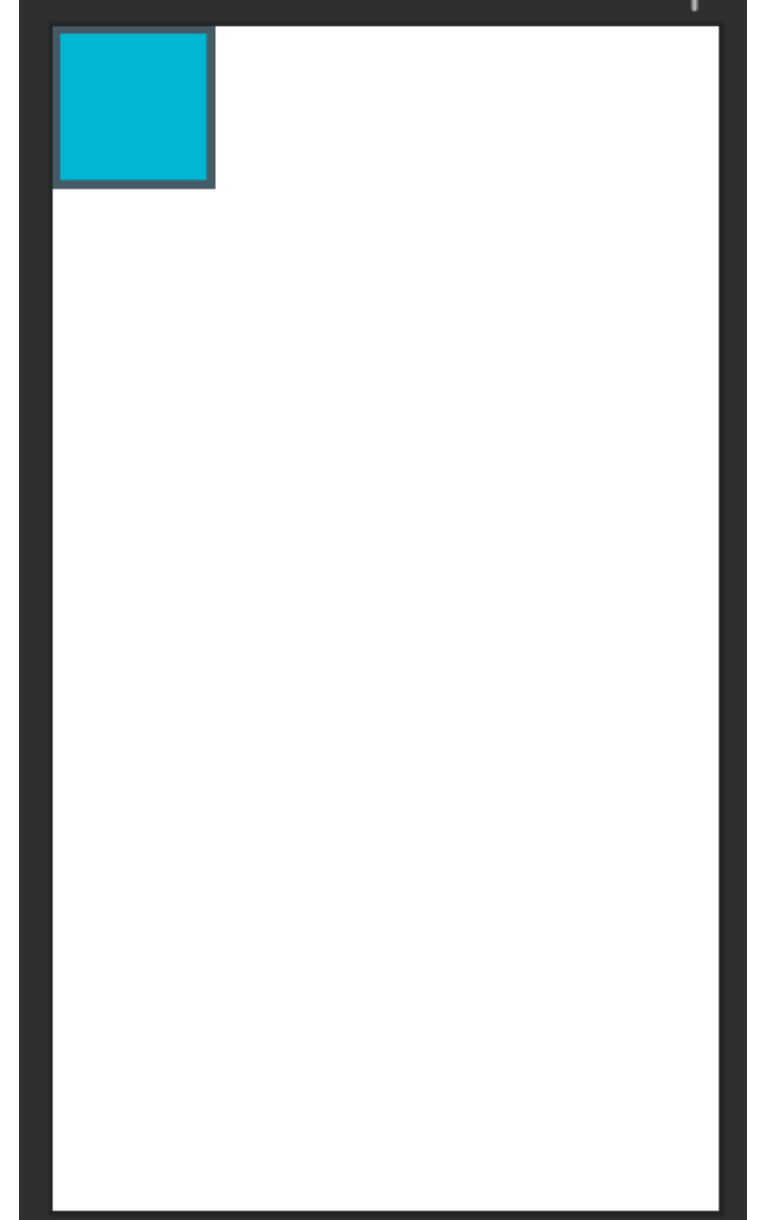
```
    android:layout_width="100dp"
```

```
    android:layout_height="100dp"
```

```
    android:background="@drawable/ic_square_round"
```

```
/>
```

뷰에서 Background 속성으로 만들어준 shape를 추가 해줄 수 있다.



테마를 설정하는 파일 - colors.xml

- 컬러 값에 이름을 지정하는 파일
- 컬러를 변경할 때 한번에 변경 할 수 있다.
- color 태그를 사용하고 name 속성에 컬러의 이름을 지정한다.
- 사용할 때는 @color/ 를 붙여준다.

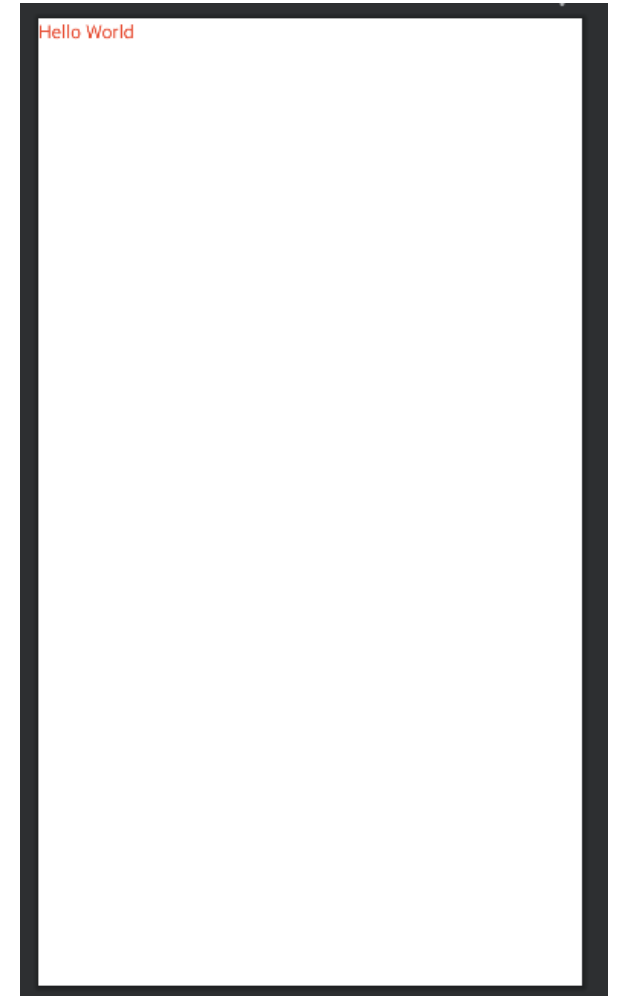
```
<resources>
  <color name="flatPurple">#6c63ff</color>
  <color name="flatBlue">#536dfe</color>
  <color name="flatLightBlue">#00b0ff</color>
  <color name="flatGreen">#1abc9c</color>
  <color name="flatYellow">#f1c40f</color>
  <color name="flatRed">#e74c32</color>
  <color name="flatBlack">#121212</color>
</resources>
```

colors.xml

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

  android:text="Hello World"
  android:textColor="@color/flatRed"
/>
```

XML



실행 화면

테마를 설정하는 파일 - theme.xml

- 테마를 지정하는 파일
- style 태그에서 앱의 주요색, 앱의 서브색, 상태바 컬러 등을 설정한다.
- style 태그안에서 값을 넣어주면 모든 xml파일에 속성으로 일괄 적용된다.

테마 이름

상속받는 테마

```
<style name="Theme.EDCANAndorid" parent="Theme.MaterialComponents.DayNight.DarkActionBar">
  <!-- Primary brand color. -->
  <item name="colorPrimary">@color/purple_500</item>
  <item name="colorPrimaryVariant">@color/purple_700</item>
  <item name="colorOnPrimary">@color/white</item>
  <!-- Secondary brand color. -->
  <item name="colorSecondary">@color/teal_200</item>
  <item name="colorSecondaryVariant">@color/teal_700</item>
  <item name="colorOnSecondary">@color/black</item>
  <!-- Status bar color. -->
  <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>
  <!-- Customize your theme here. -->

  <item name="android:textColor">@color/black</item>
</style>
```

앱의 주요 색

앱의 서브 색

앱의 위에 보이는 상태 바 컬러

모든 뷰의 textColor는 black으로 정의된다.

- 만들어준 테마는 Manifest 파일에서 적용 시켜준다.

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.EDCANAndorid">
```

폰트 적용하기

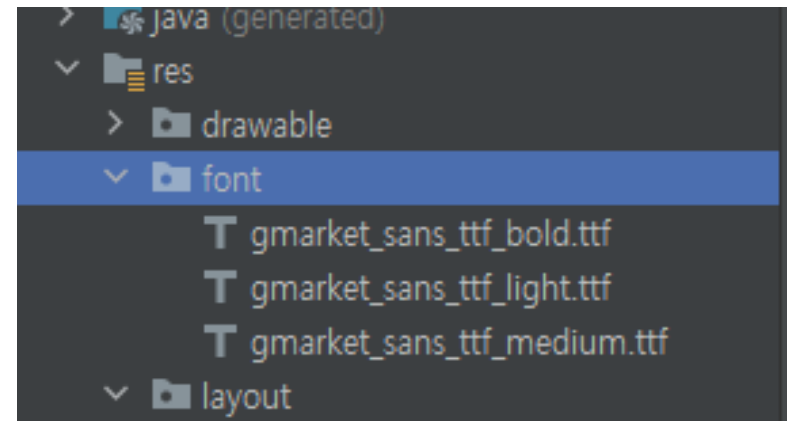
1. 적용할 폰트를 다운 받는다.

이때 파일명에는 소문자와 _(언더바)만 허용된다.



2. res/font 파일에 다운받은 폰트파일을 넣는다.

이때 font파일이 없다면 만들어준다.



3. xml에서 fontFamily에 적용한다.

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

    android:text="Hello World"
    android:textSize="20sp"

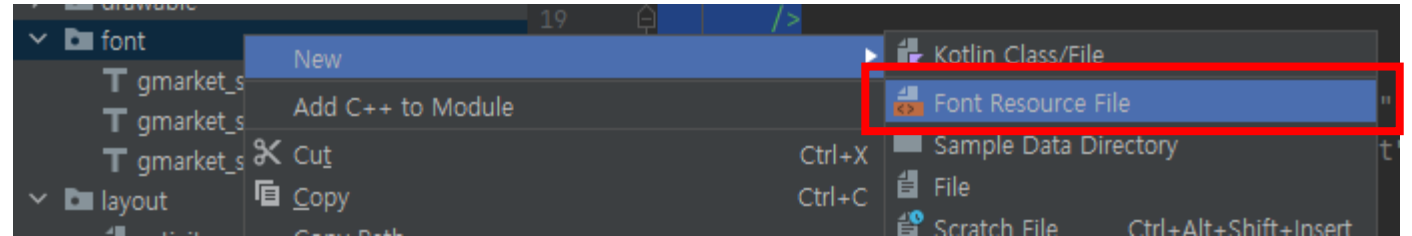
    android:fontFamily="@font/gmarket_sans_ttf_bold"
/>
```

Hello World

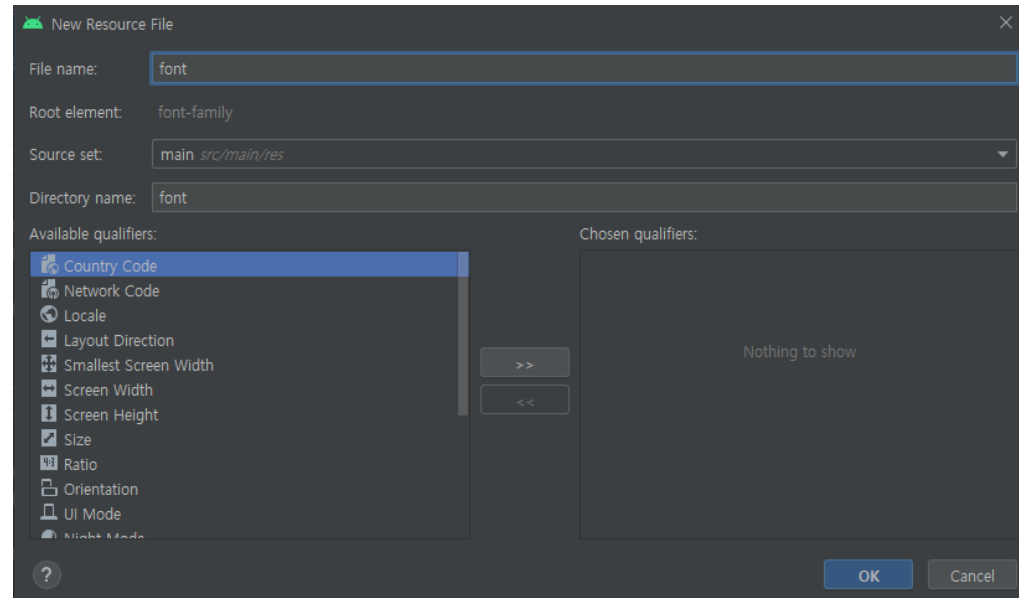
더 쉽게 폰트 적용하기

- 앞의 방법으로 하면 불편하다. ex) 폰트가 변경된 상황, 여러 두께의 폰트 관리가 어렵다.
- 폰트를 관리하는 XML 파일을 만들고 그 파일을 fontFamily에 적용한다.

1. res/font 에서 xml 파일을 만들어준다.



2. 만들 파일을 설정해준다.



더 쉽게 폰트 적용하기

폰트 관리 파일은 최상위 태그가 font-family이다.

```
<font-family xmlns:android="http://schemas.android.com/apk/res/android">
  <font
    android:font="@font/gmarket_sans_ttf_light"
    android:fontWeight="100"
  />
  <font
    android:font="@font/gmarket_sans_ttf_medium"
    android:fontWeight="300"
  />
  <font
    android:font="@font/gmarket_sans_ttf_bold"
    android:fontWeight="500"
  />
</font-family>
```

적용할 폰트
폰트를 어떤 두께일때 적용할지

font.xml

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

  android:text="Hello World"
  android:textSize="20sp"

  android:fontFamily="@font/gmarket_sans_ttf_bold"
/>
<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"

  android:text="Hello World"
  android:textSize="20sp"

  android:fontFamily="@font/gmarket_sans_ttf_medium"
/>
<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"

  android:text="Hello World"
  android:textSize="20sp"

  android:fontFamily="@font/gmarket_sans_ttf_light"
/>
```

XML

Hello World
Hello World
Hello World

실행 화면

Event

Android에서 이벤트를 정의하는 방법

1. 리스너 만들기

- Kotlin 파일에서 뷰에 접근해서 정의한다.
- 이때 이벤트는 람다식으로 정의한다.
- 대부분 뷰에 어떤 행동을 했을 때 실행된다.

```
val 버튼 = findViewById<Button>(R.id.btn)
버튼.setOnClickListener {
    Snackbar.make(it, "버튼을 클릭했습니다.", Snackbar.LENGTH_LONG).show()
}
```

클릭 이벤트 예시

2. 클래스 메소드 재정의

- 안드로이드 클래스의 메소드를 재정의
- 대부분 화면 또는 안드로이드 시스템에 어떤 행동을 했을 때 실행된다.

```
override fun onBackPressed() {
    Toast.makeText(this, "뒤로가기는 할 수 없습니다.", Toast.LENGTH_LONG).show()
}
```

뒤로 가기 버튼 눌렀을 때 실행되는 이벤트

Click Event

- 모든 뷰에서 사용 가능하다.
- 뷰를 클릭했을 때 실행되는 이벤트 이다.
- 뷰에 `setOnClickListener`을 사용해서 람다식으로 정의한다.
- 이벤트를 정의하는 람다식에서 `it`은 클릭된 버튼을 의미한다.

```
val 버튼 = findViewById<Button>(R.id.btn_main_mainButton)

버튼.setOnClickListener {
    Toast.makeText(this, "버튼을 클릭했습니다.", Toast.LENGTH_LONG).show()
}
```

버튼을 클릭했을 때 토스트 메시지가 나온다.

TextChanged Event

- EditText에서 입력된 값이 변경됐을 때 실행된다.
- 뷰를 클릭했을 때 실행되는 이벤트 이다.
- 이벤트를 정의하는 람다식에서 it은 입력한 값이다.

```
val 이름입력창 = findViewById<EditText>(R.id.edt_main_nameInput)
이름입력창.addTextChangedListener({
    Toast.makeText(this, it.toString(), Toast.LENGTH_SHORT).show()
})
```

입력창에 값을 입력하면 입력된 값이 토스트메시지라 출력된다.

checkedChange Event - CheckBox

- 체크박스에서 체크 여부가 변경 되었을 때 실행되는 이벤트
- 이벤트를 정의하는 람다식에서 첫번째 인자는 체크박스이고, 두번째 인자는 체크 여부가 Boolean 형으로 들어온다.
- 스위치 버튼에서도 동일하게 사용 가능하다.

```
val 체크박스 = findViewById<CheckBox>(R.id.ck_main_check)
체크박스.setOnCheckedChangeListener { compoundButton, isChecked ->
    if(isChecked){
        Toast.makeText(this, "체크됨", Toast.LENGTH_LONG).show()
    }
    else {
        Toast.makeText(this, "체크됨 해제됨", Toast.LENGTH_LONG).show()
    }
}
```

체크박스의 체크를 변경했을때 실행되는 이벤트

checkedChange Event – RadioButton

- 라디오 그룹에서 선택된 라디오 버튼이 변경될때 실행되는 이벤트
- 이벤트를 정의하는 람다식에서 첫번째 인자는 라디오 그룹이고, 두번째 인자는 선택된 라디오 버튼의 id가 들어온다.

```
<RadioGroup
    android:id="@+id/rg_main_choiceGender"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
>
    <RadioButton
        android:id="@+id/rb_main_choiceMale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:text="남자"
    />

    <RadioButton
        android:id="@+id/rb_main_choiceFemale"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:text="여자"
    />
</RadioGroup>
```

XML

```
val 성별선택그룹 = findViewById<RadioGroup>(R.id.rg_main_choiceGender)
성별선택그룹.setOnCheckedChangeListener { radioGroup, checkedId ->
    when(checkedId){
        R.id.rb_main_choiceMale -> {
            Toast.makeText(this, "남자 선택", Toast.LENGTH_LONG).show()
        }
        R.id.rb_main_choiceFemale -> {
            Toast.makeText(this, "여자 선택", Toast.LENGTH_LONG).show()
        }
    }
}
```

Kotlin

onBackPressed

- 뒤로가기 버튼을 클릭했을때 실행되는 이벤트
- 클래스의 메소드를 재정의 시켜준다.

```
override fun onBackPressed() {  
    Toast.makeText(this, "뒤로가기를 눌렀습니다.", Toast.LENGTH_LONG).show()  
}
```

그 밖의 이벤트...

**LongClick, Touth,
SeekBarChange,
keyDown, FocusChanged**

.....

필요할 때마다 구글에 검색 해보자

Activity와 Intent

Android의 구성 요소

- 안드로이드는 4개의 컴포넌트로 구성 돼있다. (= 안드로이드 4대 컴포넌트)
(Activity, Service, Broadcast Receiver, Content Provider)
- 각 컴포넌트는 하나의 독립된 형태로 존재하고 각자의 역할을 한다.
- 각 컴포넌트는 Intent를 통해서 통신한다.

Activity

- UI 화면을 담당하는 컴포넌트
- Activity는 class와 XML로 이루어져 있다.
- 모든 Activity class는 AppCompatActivity class를 상속받는다.
- Activity가 생성되는 가장 먼저 onCreate 메소드가 실행된다.
- onCreate 메소드에서는 화면에 보여줄 XML과 매칭 시켜줘야 한다.

AppCompatActivity 상속

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main) XML 파일 매칭  
    }  
}
```

Intent

- 안드로이드에서 동작하는 4대 컴포넌트간 통신을 위해 사용한다.
- 다른 앱에 있는 컴포넌트에도 접근 할 수 있다.
- 인텐트에는 액션과 데이터가 필요하다. 액션은 수행할 기능, 데이터는 수행할 기능의 데이터다.
- 다른 액티비티로 이동할 때 많이 사용한다.
- Intent 객체를 만들어 준 뒤 **startActivity** 함수를 호출해서 인텐트를 실행한다.

```
val intent = Intent(액션 Intent.ACTION_DIAL, 데이터 Uri.parse("tel:112"))  
startActivity(intent) 인텐트 실행
```

전화를 거는 인텐트 예제

```
val intent = Intent(액션 Intent.ACTION_VIEW, 데이터 Uri.parse("https://edcan.kr"))  
startActivity(intent) 인텐트 실행
```

웹사이트로 이동하는 인텐트 예제

Intent - 다른 액티비티로 이동하기

- 액티비티를 이동할 때는 원래 있던 액티비티와 이동하려는 액티비티 두개가 필요하다.

```
val intent = Intent(this, SubActivity::class.java)  
startActivity(intent)
```

SubActivity로 이동하는 예제

Intent – 액티비티 이동할 때 값 넘겨주기

- 액티비티를 이동할 때 원래 있던 액티비티에서 이동하려는 액티비티에 데이터를 넘겨줄 수 있다.

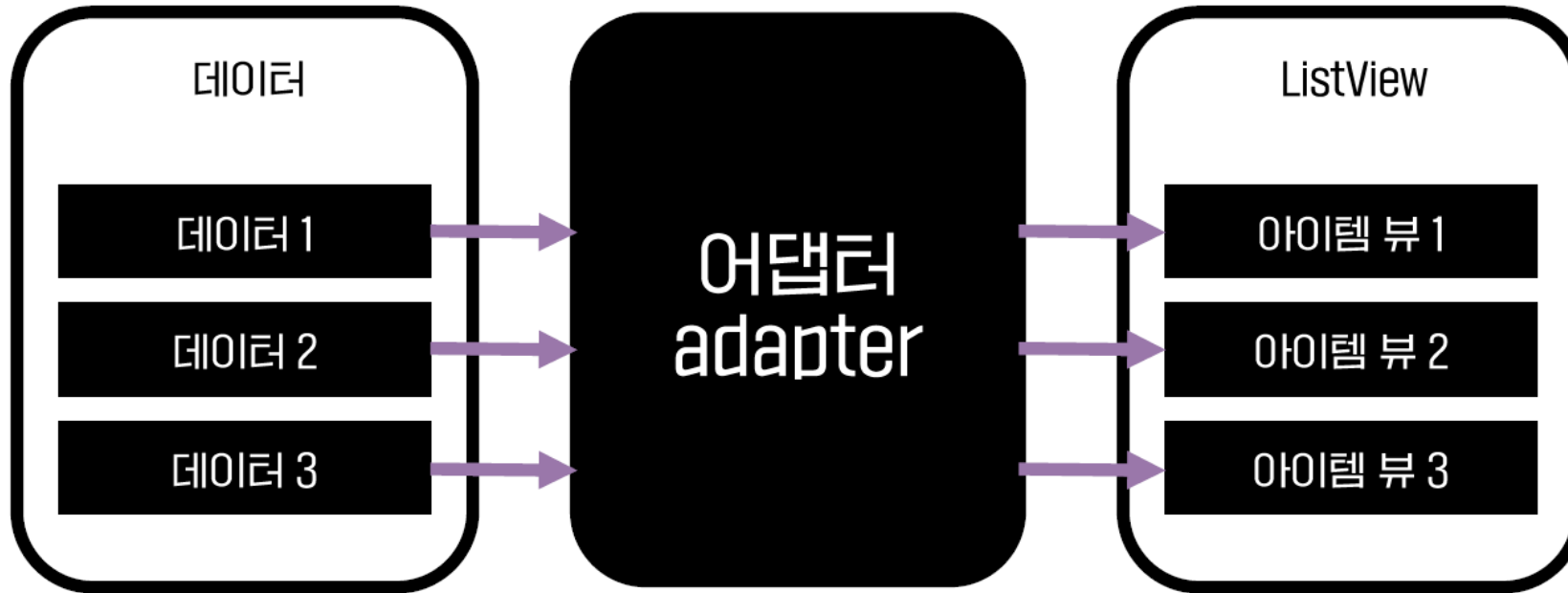
```
val intent = Intent(this, SubActivity::class.java)  
startActivity(intent)
```

SubActivity로 이동하는 예제

List View / Recycler View

ListView

- 화면에 데이터들을 리스트의 형태로 보여주는 View
- 데이터들을 리스트로 만들기 위해서는 어댑터를 만들어 줘야 한다.




```

<ListView
    android:id="@+id/list_main_club"
    android:layout_width="0dp"
    android:layout_height="0dp"

    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toTopOf="@+id/btn_main_addItem"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
/>

<Button
    android:id="@+id/btn_main_addItem"
    android:layout_width="0dp"
    android:layout_height="wrap_content"

    android:text="추가"

    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
/>

```

XML

```

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val dataList = mutableListOf<String>("Item 1", "Item 2", "Item 3", "Item 4", "Item 5")
        val adapter = ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, dataList)
        findViewById<ListView>(R.id.list_main_club).adapter = adapter

        findViewById<Button>(R.id.btn_main_addItem).setOnClickListener {
            adapter.add("Item ${dataList.size + 1}")
            adapter.notifyDataSetChanged()
        }
    }
}

```

Kotlin

EDCAN Andorid

Item 1

Item 2

Item 3

Item 4

Item 5

Item 6

Item 7

Item 8

Item 9

Item 10

Item 11

Item 12

Item 13

추가

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)
```

```
        val dataList = mutableListOf<String>("Item 1", "Item 2", "Item 3", "Item 4", "Item 5")
```

화면에 리스트로 보여줄 데이터

```
        val adapter = ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, dataList)
```

어댑터를 만들어준다

```
        findViewById<ListView>(R.id.list_main_club).adapter = adapter
```

만들어준 어댑터를 연결시켜준다

```
        findViewById<Button>(R.id.btn_main_addItem).setOnClickListener {
```

```
            adapter.add("Item ${dataList.size + 1}")
```

리스트에 아이템을 추가한다.

```
            adapter.notifyDataSetChanged()
```

리스트를 새로 그려준다.

```
        }
```

```
    }
```

```
}
```

ListView 어댑터 기능

`add()` : 데이터 추가

`clear()` : 데이터 전체 삭제

`remove(item)` : 데이터 삭제

`getPostion(item)` : 해당 아이템의 위치

`notifyDataSetChanged()` : 리스트 새로 그리기

ListView 단점

- 리스트 아이템 커스텀을 할 수 없다.
- 리스트 아이템이 많아지면 화면에 보여주기 위한 시간이 오래 걸린다.
- 구글이 ListView 보다는 RecyclerView를 쓰라고 한다.

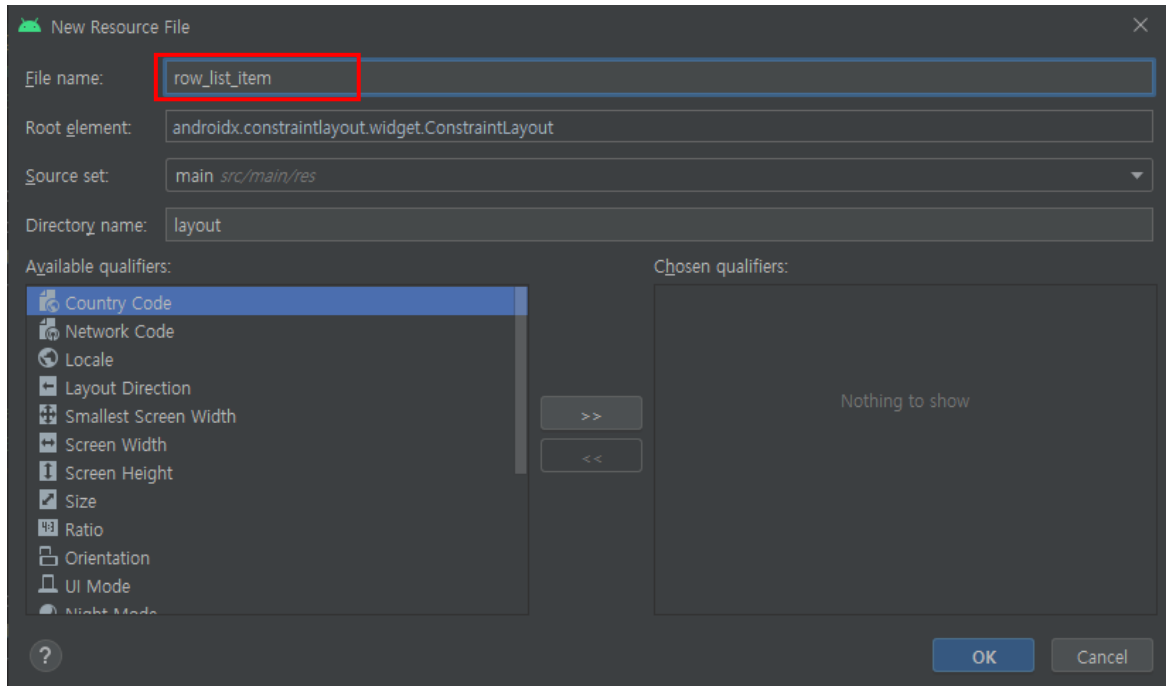
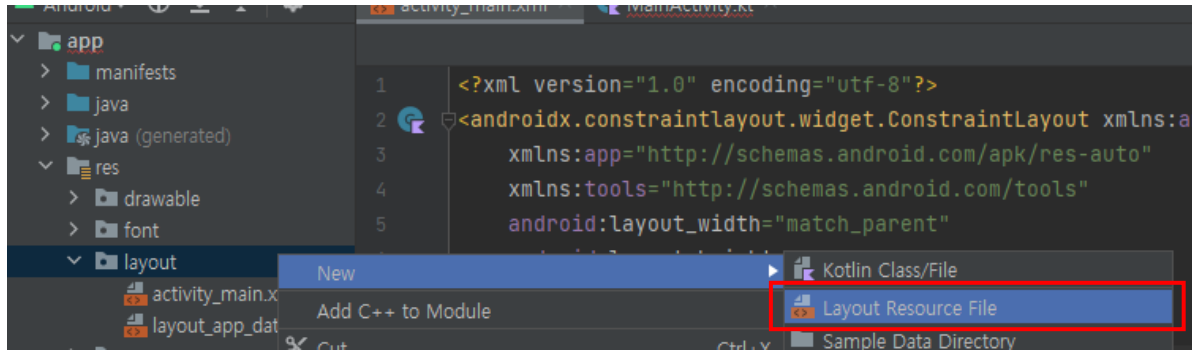
여러가지 이유 때문에 RecyclerView를 사용한다.

Recycler View

- 리스트 아이템 커스텀을 할 수 있다.
- 어댑터를 직접 만들어서 사용한다.
- ViewHolder를 만들어 줘야 한다.
- 리스트 아이템 레이아웃을 직접 만들어 줘야 한다.

Recycler View 사용법

1. 리스트 아이템 레이아웃 만들기



Recycler View 사용법

1. 리스트 아이템 레이아웃 만들기

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:paddingVertical="16dp">
    <TextView
        android:id="@+id/txt_listItem_number"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:text="기본 값"
        android:textSize="20dp"

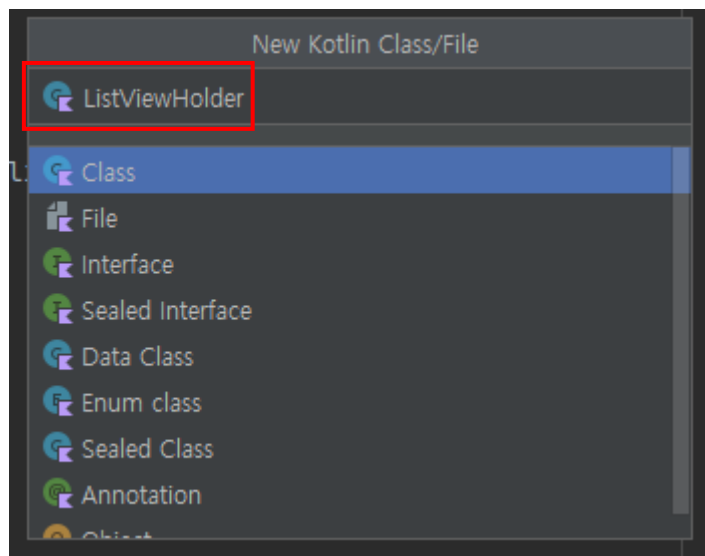
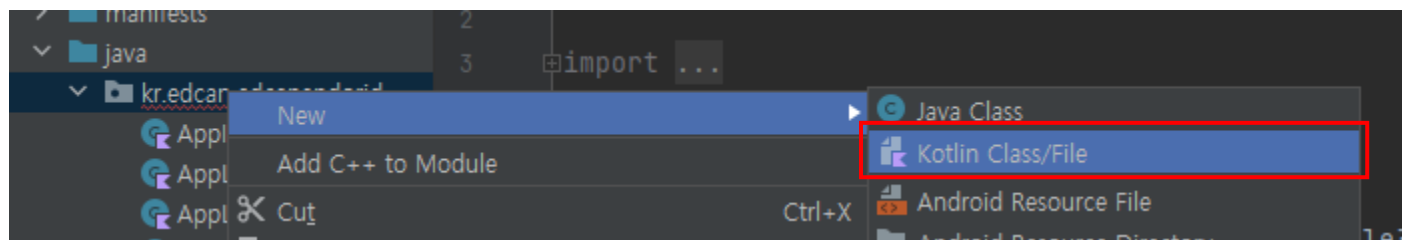
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
    />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Recycler View - ViewHolder

- 리스트에서 아이템 하나를 클래스로 표현한 것
- 하나의 리스트 아이템에 있는 뷰들이 모여있다.
- 모든 ViewHolder은 **RecyclerView.ViewHolder** 추상 클래스를 상속 받는다.
- 객체 생성시 View를 하나 인자로 받는다.

Recycler View 사용법

2. ViewHolder 만들기



Recycler View 사용법

2. ViewHolder 만들기

```
class ListViewHolder(val row : View) : RecyclerView.ViewHolder(row) {  
    val number = row.findViewById<TextView>(R.id.txt_listItem_number)  
}
```

리스트 아이템의 뷰들을 가져와준다.

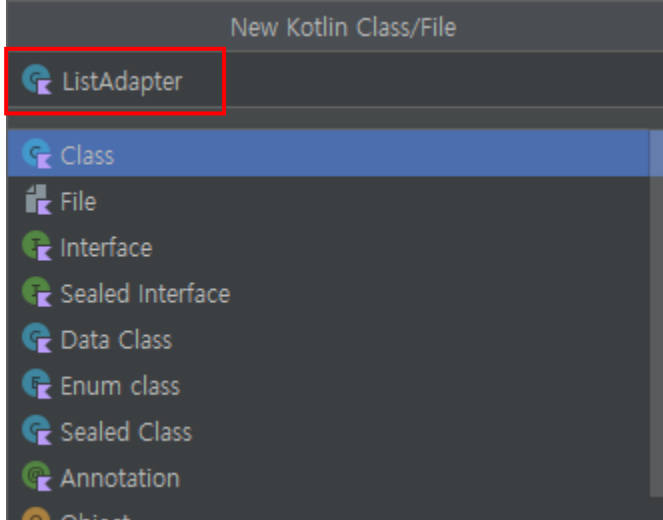
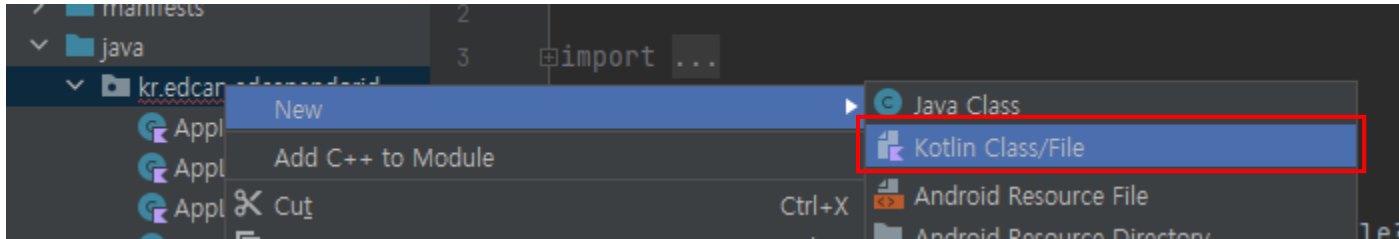
Recycler View - Adapter

- 데이터들을 각각의 리스트 아이템으로 만들어 준다.
- RecyclerView.Adapter 추상 클래스를 상속 받아준다.

이때 만들어준 ViewHolder클래스를 제네릭으로 넣어준다.

Recycler View 사용법

2. Adapter 만들기



Recycler View 사용법

2. Adapter 만들기

```
class ListAdapter(val items : List<Int>) : RecyclerView.Adapter<ListViewHolder>() {  
    }  
}
```

리스트로 보여줄 데이터들

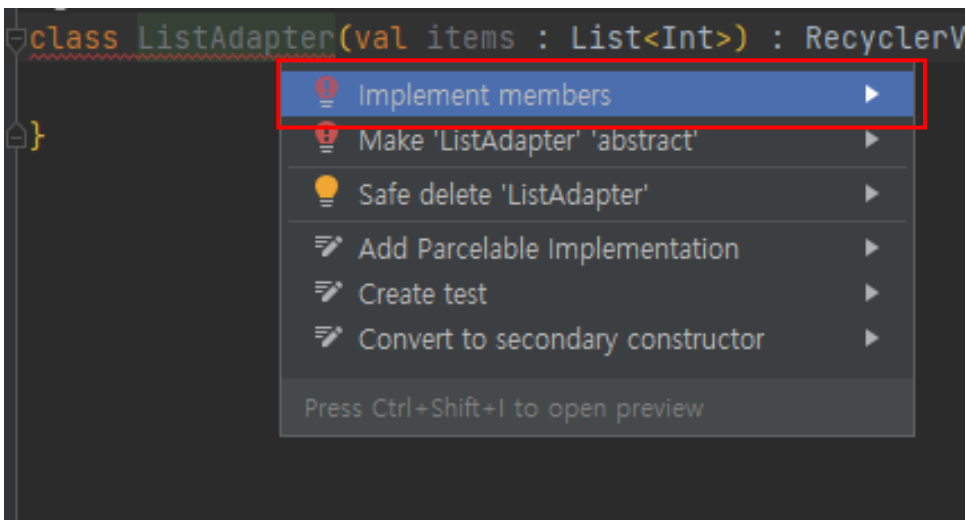
RecyclerView.Adapter를 상속

만들어준 ViewHolder

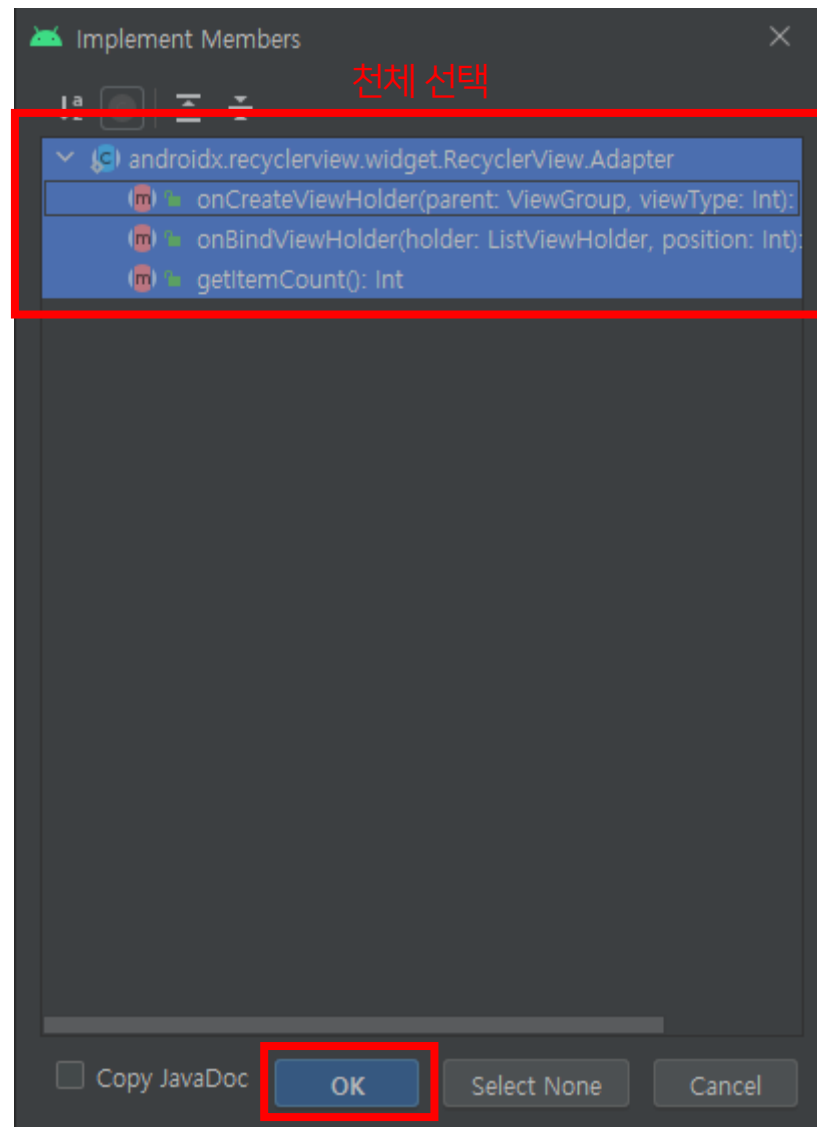
이 부분에서 에러가 난다.

Recycler View 사용법

2. 상속 받은 추상클래스 Implement 해주기



에러 난 부분에서 Alt + Enter



Recycler View 사용법

2. 상속 받은 추상클래스 Implement 해주기

```
class ListAdapter(val items : List<Int>) : RecyclerView.Adapter<ListViewHolder>() {  
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ListViewHolder {  
        TODO("Not yet implemented")  
    } 리스트 아이템별 ViewHolder 만드는 메서드  
  
    override fun onBindViewHolder(holder: ListViewHolder, position: Int) {  
        TODO("Not yet implemented")  
    } 리스트 아이템에 데이터를 채워넣어주는 메서드  
  
    override fun getItemCount(): Int {  
        TODO("Not yet implemented")  
    } 아이템의 전체 개수를 반환하는 메서드  
}
```

RecyclerView.Adapter 추상 클래스는 3개의 메서드를 구현 해줘야 한다.

Recycler View 사용법

2. 상속 받은 추상클래스 Implement 해주기

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {  
    val view = LayoutInflater.from(parent.context).inflate(R.layout.row_list_item, parent, false)  
    return ViewHolder(view)  
}
```

만들어준 레이아웃 파일

LayoutInflater는 뷰를 동적으로 만들어 주는 역할을 한다.

만들어준 뷰를 ViewHolder로 만들어서 반환한다.

Recycler View 사용법

2. 상속 받은 추상클래스 Implement 해주기

```
override fun onBindViewHolder(holder: ListViewHolder, position: Int) {  
    holder.number.text = items[position].toString()  
}
```

ViewHolder에 데이터를 채워 넣어 준다.

Recycler View 사용법

2. 상속 받은 추상클래스 Implement 해주기

```
override fun getItemCount(): Int {  
    return items.size  
}
```

데이터들의 전체 개수를 반환하다.

Recycler View 사용법

3. 리사이클러뷰와 어댑터를 연결해주기

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/list_main_club"
    android:layout_width="0dp"
    android:layout_height="0dp"

    tools:listitem="@layout/row_list_item"

    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
/>
```

activity_main.xml

```
val recyclerView : RecyclerView = findViewById(R.id.list_main_club)

val datas = listOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
val adapter = ListAdapter(datas)

recyclerView.adapter = adapter
recyclerView.layoutManager = LinearLayoutManager(this)
```

리스트가 보여지는 형식

EDCAN Andorid

1

2

3

4

5

6

7

8

9

10

Recycler View 사용법

리사이클러뷰 아이템 클릭 이벤트

```
class ListViewHolder(val row : View) : RecyclerView.ViewHolder(row) {  
    val number = row.findViewById<TextView>(R.id.txt_listItem_number)  
  
    init {  
        row.setOnClickListener {  
            Toast.makeText(row.context, "${number.text} 클릭", Toast.LENGTH_LONG).show()  
        }  
    }  
}
```

ViewHolder에서 들어온 view에 이벤트를 정의 해준다.

Recycler View 사용법

리사이클러뷰 데이터 추가하기

```
adapter.items.add(11)
```

 이때 items가 mutable list일 때 만 추가 가능하다.

```
adapter.notifyDataSetChanged()
```

 리스트를 새로 그린다.