Application Development for Mobile Computer

<Week 5>

Youn Kyu Lee

Layout

Layouts in Android

- Layouts are responsible for arranging UI elements on the screen
- When a new project is created, a layout file named activity_main is automatically generated
- Layout files are classified as resources, not source code
  - File names must be written in lowercase
  - The file type .xml is appended → activity_main.xml

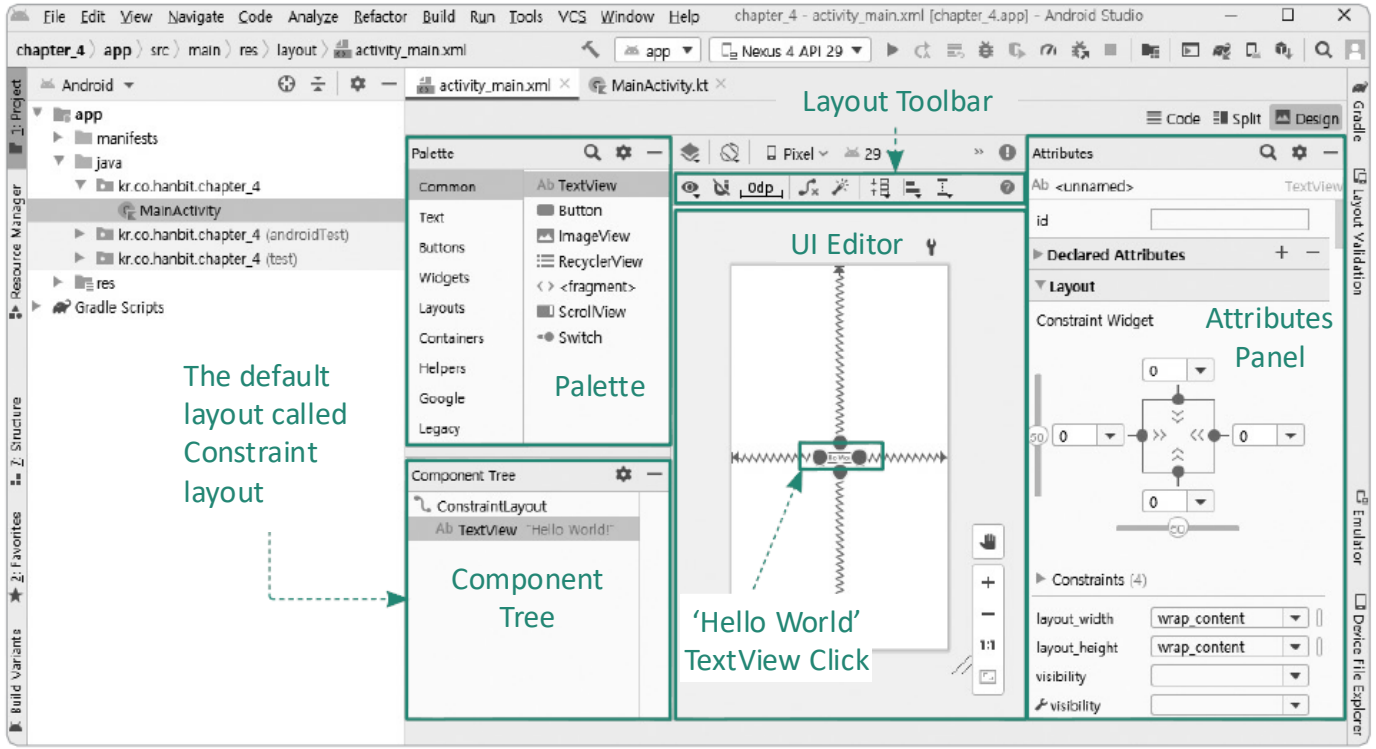
Types of Layouts

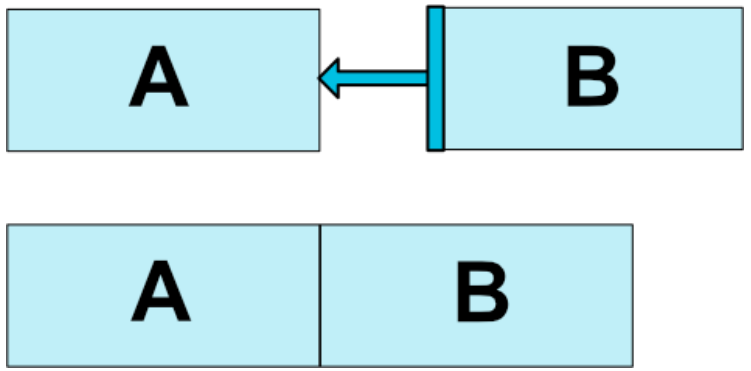
- ConstraintLayout
  - Allows flexible arrangement of UI elements with simple drag-and-drop
- LinearLayout
  - Places widgets in a single row or column (horizontal or vertical)
- FrameLayout
  - Used for overlapping widgets, rather than specifying exact positions

Constraint Layout

Constraint Layout

- Checking the Default Layout
  - Open the layout file by clicking activity_main.xml in the top tab
  - Check the default layout configuration

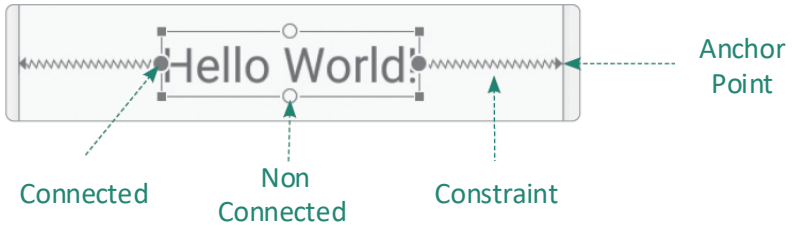


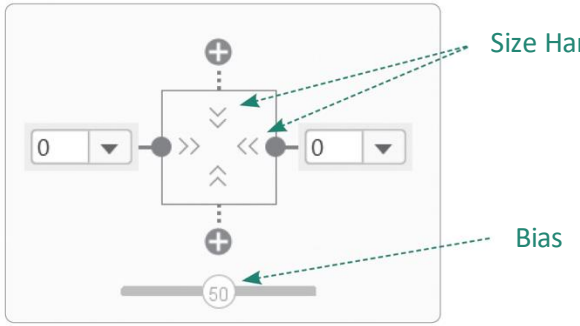


Constraint Layout

Constraint Layout

- Using Handlers
  - Constraint and Anchor Point



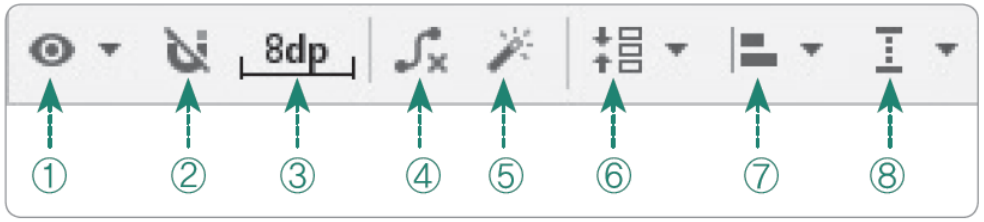


- Constraint Editor
  - When a widget inside a ConstraintLayout is selected, the Constraint Editor appears in the attributes panel
- Size Handler
  - Used when constraints are connected on both vertical or horizontal sides

Constraint Layout

Constraint Layout

- Bias
  - When constraints are connected vertically or horizontally, the bias adjustment button becomes active
- Aspect Ratio
  - If the size is set to match constraint, the aspect ratio feature is enabled
  - A small triangle appears in the upper-left corner of the rectangle
  - Click the triangle to set the width-to-height ratio
- Layout Toolbar
  - Depending on the selected layout, the toolbar for that layout is provided at the top of the UI editor

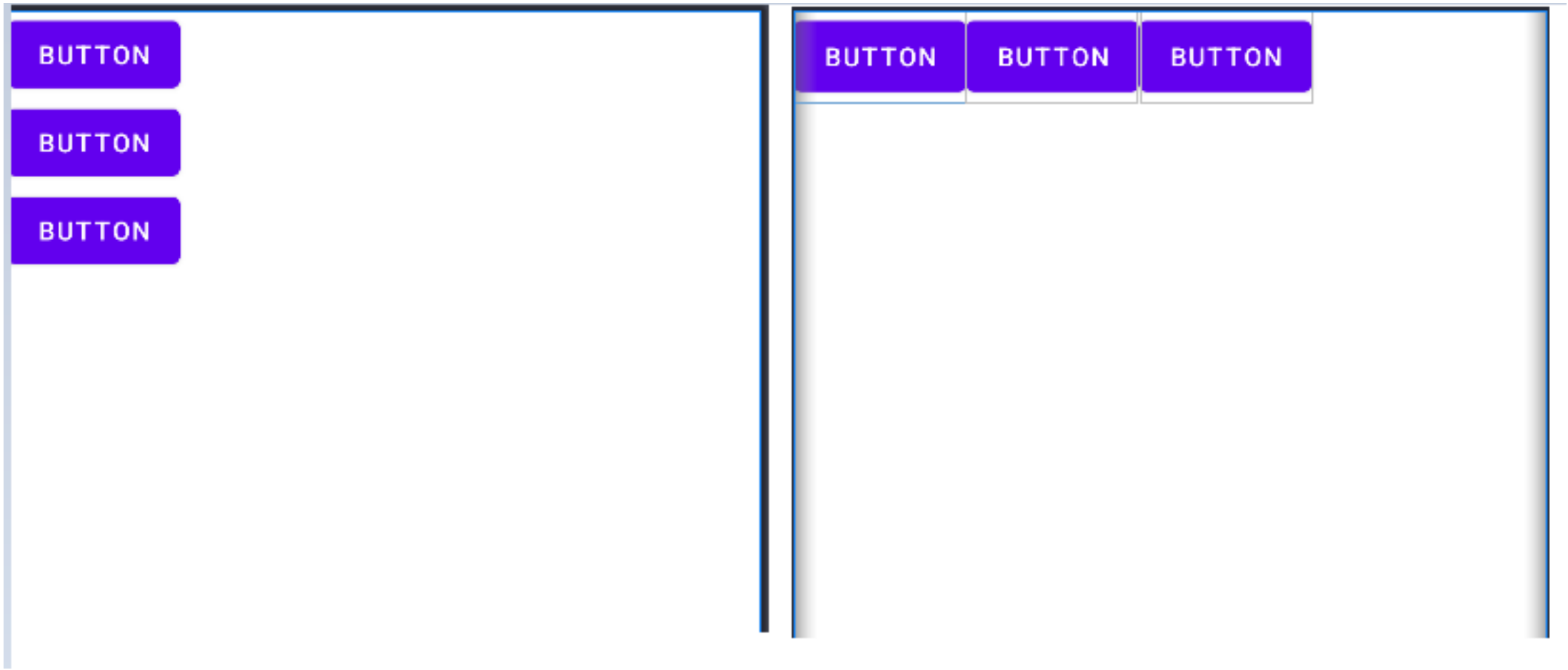


Linear Layout

LinearLayout

- A layout that arranges widgets in a single row or column
- By changing the orientation property to horizontal or vertical, existing widgets can also change direction
- Orientation Property
  - Since Android 3.1, if not specified, the default is horizontal
- layout_weight Property
  - Specifies the proportional size of widgets within a LinearLayout
  - Default value is 1 for each widget
- gravity Property
  - Aligns widgets inside the layout according to the direction specified in gravity
  - Multiple directions can be applied at the same time
- layout_gravity Property
  - Defines the position of a widget relative to its parent layout
- Space Tool
  - Space is a helper tool used to insert empty gaps in a layout
  - Commonly used in LinearLayouts to keep equal spacing between buttons

Linear Layout



FrameLayout

FrameLayout

- A layout used not for determining widget positions, but for overlapping widgets
- Commonly used in cases like game screens, where the background and player must move on different layers
- Among layouts, it has the fastest processing speed, making it optimal for simple cases such as displaying a single image
- Primarily used to stack other layouts or widgets on top of each other
- No mandatory properties specific to FrameLayout
- Alignment is handled by the layout_gravity property of the inserted widgets, not the FrameLayout itself

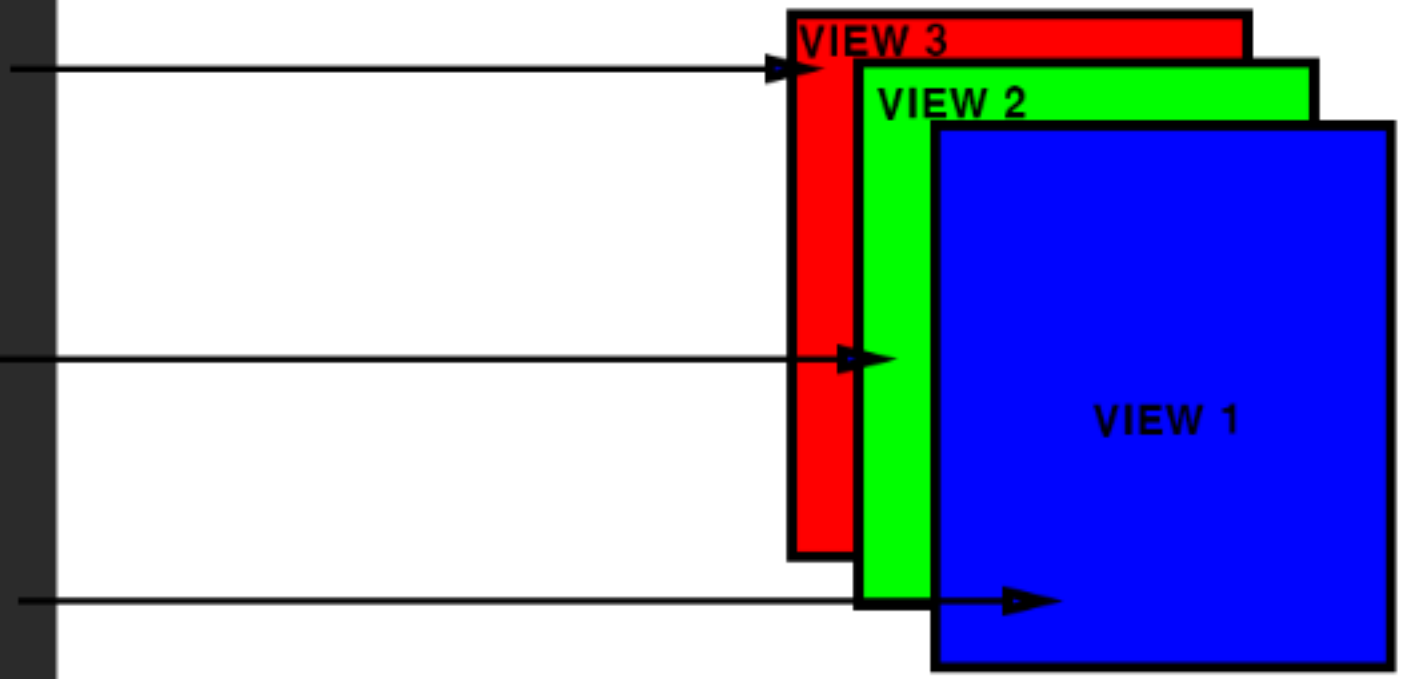
FrameLayout

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/colorAccent"
        android:text="TextView" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimary"
        android:text="TextView" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimaryDark"
        android:text="TextView" />
</FrameLayout>
```



FrameLayout

FrameLayout

- To check the XML code when using a FrameLayout, click the [Code mode icon] above the attribute panel
- Switching Back to Design Mode
  - Change the UI editor back to [Design] mode
  - Drag a button from the Palette into the UI editor
  - The button will then appear inside the editor layout

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

</FrameLayout>
```

FrameLayout

FrameLayout

- FrameLayout XML Structure
  - When the UI editor is switched to [Code] mode, the XML code appears as follows

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

In code, the Button widget is written as follows

- Building Layouts
  - Layouts can be created by drag-and-drop using the Palette
  - Or they can be built by directly writing XML code

Widget

Widget's Main Menu

- Widgets are UI design elements such as Button, TextView, and ImageView
  - The term “widget” is also used for items on a smartphone home screen — do not confuse the two
  - Until Android Studio 3.0, all widget menus were grouped together
  - Since version 3.1, they have been divided into: Text, Buttons, and Widgets
- Common
  - Menu that collects frequently used items such as Text, Button, and Layout
- Text
  - TextView: Displays text only
  - EditText: Receives text input
  - In the menu, the first TextView has an “Ab” icon without an underline
  - Items with an “Ab” icon with an underline are all EditText widgets
- Buttons
  - Widgets that receive click or touch events
  - Includes: Button, RadioButton, CheckBox, Switch, etc.
- Widgets
  - Collection of UI elements for displaying images, websites, ratings, progress status, etc.

Text View

Text View

- A widget that displays text on the screen (like Button, TextView, ImageView)
- Define and Apply Text: text
  - Text can be entered directly into the text attribute (not recommended)
  - Best practice: define text in strings.xml and reference it
  - Easier for multilingual support, text updates, and app management
- Text Color: textColor
  - Colors are defined using RGB (+ alpha for transparency) values
  - Each ranges from 0 to 255, expressed in hexadecimal (0–F)
  - Best practice: define colors in colors.xml and reference them
- Text Size: textSize
  - Typically defined in sp (Scale-independent Pixels)
  - Allows font scaling without affecting other widgets
  - Ensures accessibility: users with visual impairments can enlarge text without distorting layout
- Text Style: textStyle
  - System-provided styles: normal, bold, italic

Edit Text

Edit Text

- Can display text but is mainly used for user input
  - Common example: entering ID and password on a login screen
- Handling Input in Real Time
  - Capture text entered in EditText and print it to the log in real time
- Hint: hint
  - Displays placeholder text that disappears when clicked
  - Known as placeholder in other programming tools
- Keyboard Type: inputType
  - Changes the appearance of the keyboard depending on the option set in inputType
- Event Setting: imeOptions
  - Defines the event executed after input is completed
  - IME = Input Method Editor, refers to the text editor for input

Edit Text

Edit Text

- Setting Keyboard Type: inputType
  - The appearance of the keyboard changes depending on the option set in the inputType attribute

inputType	Option Value
textUri	URI format text input
textEmailAddress	Email address format text input
textPostalAddress	Postal code format text input
textPassword	Password input (masked)
textVisiblePassword	Password input (visible as plain text)
number	Numeric input
numberPassword	Numeric password input
phone	Phone number format input
date	Date format input

Edit Text

Edit Text

- Setting Keyboard Type: inputType
  - Defines how the input is displayed and what type of keyboard appears
  - After input is completed, an event can be set to execute

imeOptions Option		Option Value
	normal	No special function
	actionUnspecified	No specific action
	actionNone	Do not use any action
	actionGo	Navigate to a location (e.g., after entering a URL, go to that page)
	actionSearch	Search (e.g., Google, Naver, Daum)
	actionSend	Send (e.g., email, message)
	actionNext	Next (move to the next input field)
	actionDone	Done (complete input and hide the keyboard)
	actionPrevious	Previous (return to the previous input field)

Image Button

Image Button

- Both Button and ImageButton can have an image set through the background attribute
  - Button: displays text over a background image
  - ImageButton: can display an icon or image over the background image
- Using a Default Image
  - Open activity_main.xml
  - Drag an ImageButton from the palette into the UI editor
  - A dialog will appear to select the image to use
- Using a New Image
  1. Prepare the image file → click [Refactor] → save it in the drawable directory
  2. Drag an ImageButton from the palette into the UI editor → select the image in the popup dialog
  3. The selected image will appear in the editor

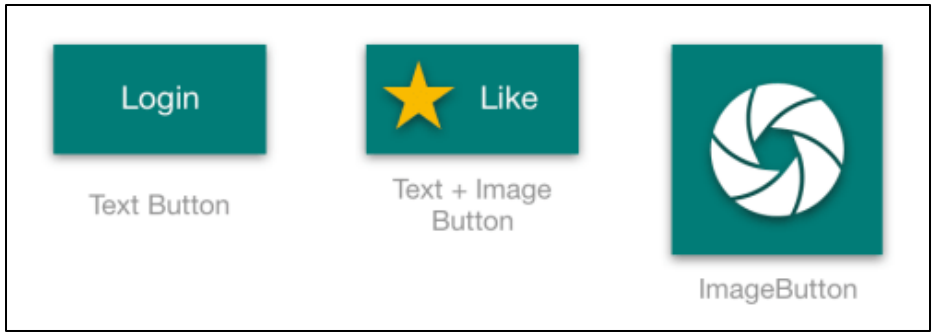


Image Button

Image Button

- Transparent Background
  - By default, an ImageButton includes a gray background
  - Apply @android:color/transparent to the background attribute to remove the gray area and make it transparent
- Image Size: scaleType
  - matrix: Displays the actual image starting from the top-left corner, cropped to the ImageButton size
  - fitXY: Stretches the image to match the width and height of the view
  - fitStart: Scales the image proportionally, aligned to the top-left corner
  - fitCenter: Scales the image proportionally, centered in the view
  - fitEnd: Scales the image proportionally, aligned to the bottom-right corner
  - center: Displays the image in its original size, centered; cropped if larger than the view
  - centerCrop: Scales based on the closest dimension and crops the other, filling the view (commonly used for thumbnails)
  - centerInside: If the image is larger, behaves like fitCenter; if smaller, places the image at the cente

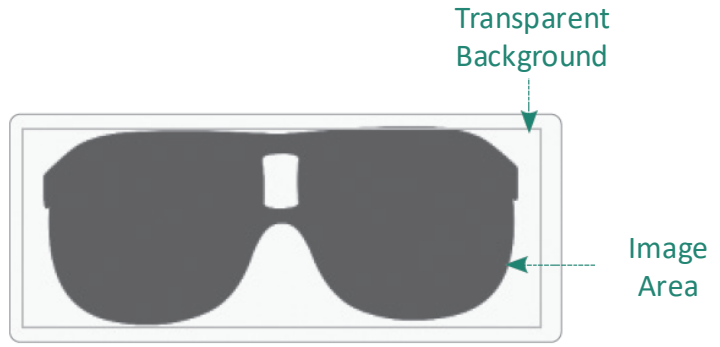
Image Button

Image Button

- Fill Image Area with Color: tint
  - The tint attribute fills the image area with a selected color
  - Use the [Eyedropper icon] to choose a color
  - Applied based on the transparency of the image
  - Commonly used for images with transparent backgrounds

▲ When purple is applied to a regular image

- Adjust Transparency: alpha
  - Accepts values from 1 to 0
  - 1 = fully opaque (not transparent)
  - 0 = fully transparent

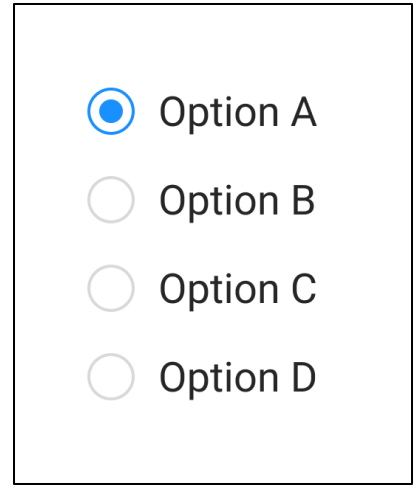


▲ When purple is applied to a glasses image with a transparent background

Radio Group and Radio Button

Radio Group and Radio Button

- RadioButton is used when only one option can be selected among multiple choices
- A RadioButton can be used alone, but it is easier to manage with a RadioGroup



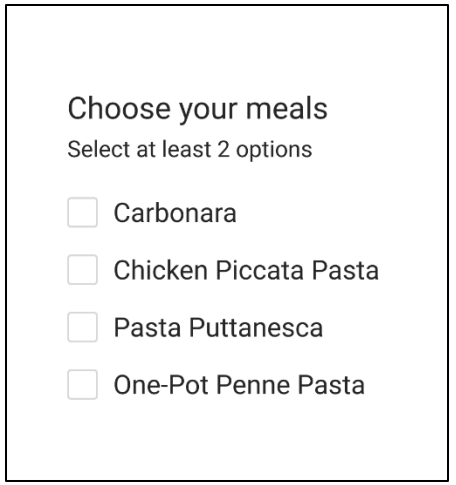
- RadioButton Attributes and Layout
  - RadioGroup is a type of layout similar to LinearLayout that holds RadioButtons
  - Use the orientation property to arrange RadioButtons either vertically or horizontally
  - Example: Changing orientation from vertical to horizontal arranges the buttons side by side
- Setting a Selected RadioButton: checkedButton
  - Used to define a pre-selected RadioButton
  - When opening the input field, the list of RadioButtons inside the RadioGroup is displayed

Check Box

Check Box

- CheckBox is similar to RadioButton in providing multiple options
  - Unlike RadioButton, multiple options can be selected at the same time
  - By default, each widget requires its own listener
  - However, a single shared listener can be implemented and used for all CheckBoxes

```
binding.checkApple.setOnCheckedChangeListener { checkBox, isChecked ->
    if (isChecked) Log.d("CheckBox", "Apple has been selected.")
    else Log.d("CheckBox", "Apple has been deselected.")
}
```

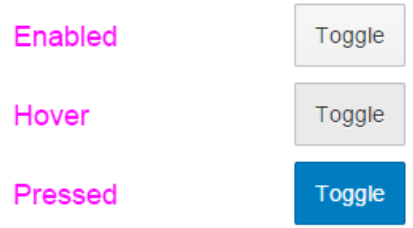


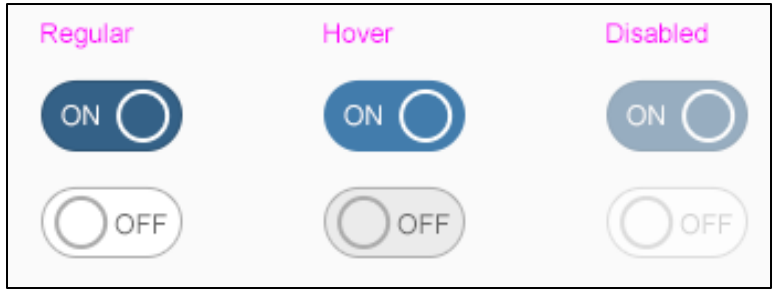
※ In actual projects, both check and uncheck states must be handled. Therefore, code for handling CheckBox unchecking should also be added

Toggle Button, Switch, Image View

Toggle Button, Switch, Image View

- ToggleButton functions the same as a CheckBox
  - Inherits from the parent class CompoundButton
  - Uses the same listener and implementation as CheckBox
  - Only the appearance is slightly different
- Switch's implementation is the same as CheckBox
  - Also inherits from CompoundButton
  - CheckBox, ToggleButton, and Switch all inherit from CompoundButton
  - Once you learn one, you can use the same listener to control all
- ImageView
  - Similar to ImageButton in usage
  - Can also receive click events with a listener
  - Recommended to use only for displaying images
  - Key attributes (src, background, scaleType) are the same as ImageButton

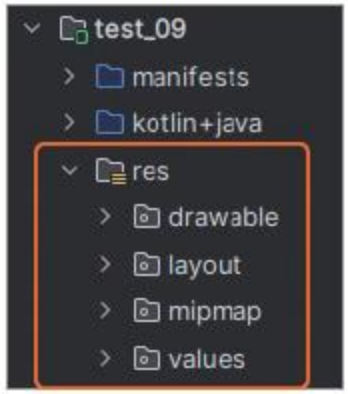




Types and Characteristics of Resources

Types and Characteristics of Resources

- Using App Resources
  - Resource directory names are fixed
  - Resource file names must follow Java naming rules (except files added under values)
  - Uppercase letters are not allowed in resource file names



Directory Name	Resource Type
animator	Property Animation XML
Anim	Twin Animation XML
color	Color State List XML
drawable	Color State List XML
mipmap	App Launch Icon Resource
layout	Layout XML
menu	Menu XML
raw	Raw Resource File
values	Value Resource
xml	Other XML Files Not in Specific Directories
font	Font Resource

Types and Characteristics of Resources

Types and Characteristics of Resources

- Layout resources → stored in the layout directory
- Image resources → stored in the drawable directory
  - Supported formats: PNG, JPG, GIF, 9.PNG
  - Images defined in XML are also possible

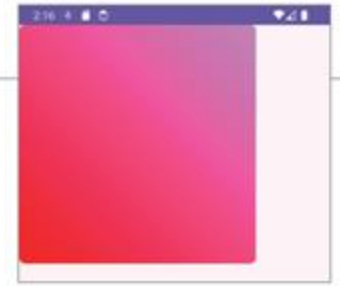
• Image defined in XML

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <gradient
        android:startColor="#FFFF0000"
        android:endColor="#80FF00FF"
        android:angle="45" />
    <corners android:radius="8dp" />
</shape>
```

• Example of using an XML Image

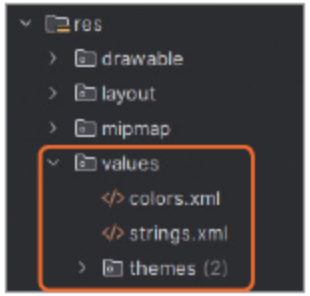
```
<ImageView
    android:layout_width="300dp"
    android:layout_height="300dp"
    android:src="@drawable/gradient_box" />
```

▶ Execution Result



Types and Characteristics of Resources

Types and Characteristics of Resources



- App icon resources → stored in the mipmap directory
- Value resources → stored in the values directory
  - Strings, colors, dimensions, styles, arrays, etc. are stored in XML
  - In the values directory, the name attribute of each XML tag is registered as the identifier, not the file name itself

• Register String Resource

```
<resources>
    <string name="app_name">Test9</string>
    <string name="txt_data1">Hello</string>
    <string name="txt_data2">World</string>
</resources>
```

• Use String Resource in XML

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/txt_data1" />
```

• Use String Resource in Code

```
binding.textView.text = getString(R.string.txt_data2)
```

Types and Characteristics of Resources

• Register Color Resource

```
<resources>
    <color name="txt_color">#FFFF00</color>
    <color name="txt_bg_color">#FF0000</color>
</resources>
```

• Register Dimension Resource

```
<resources>
    <dimen name="txt_size">20sp</dimen>
</resources>
```

• Use Color and Dimension Resources in XML

```
<TextView
    android:id="@+id/textView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/txt_data1"
    android:textColor="@color/txt_color"
    android:background="@color/txt_bg_color"
    android:textSize="@dimen/txt_size" />
```

• Use Color and Dimension Resources in Code

```
binding.textView.text = getString(R.string.txt_data2)
binding.textView.setTextColor(ResourcesCompat.getColor(resources, R.color.txt_color, null))
binding.textView.textSize = resources.getDimension(R.dimen.txt_size)
```

Types and Characteristics of Resources

Types and Characteristics of Resources

- Registered using the <style> tag
- Multiple view attributes can be defined in a style and applied all at once

• Register Style

```
<resources>
    <style name="MyTextStyle">
        <item name="android:textSize">@dimen/txt_size</item>
        <item name="android:textColor">@color/txt_color</item>
    </style>
    <style name="MyTextStyleSub" parent="MyTextStyle">
        <item name="android:textColor">#0000FF</item>
        <item name="android:background">@color/txt_bg_color</item>
    </style>
</resources>
```

• Use Style Resource

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@style/MyTextStyleSub"
    android:text="Hello World" />
```

Types and Characteristics of Resources

Types and Characteristics of Resources

- Stored in the color directory
  - Color resources define the state of a specific view and the colors applied to that state

• Define Color Resource

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
        android:color="#ffff0000" />
    <item android:state_focused="true"
        android:color="#ff0000ff" />
    <item android:color="#ff000000" />
</selector>
```

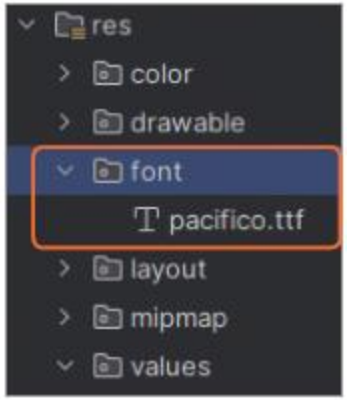
• Use Color Resource

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Click Me!!"
    android:textColor="@color/button_text" />
```

Types and Characteristics of Resources

Types and Characteristics of Resources

- Stored in the font directory
  - Used to save font resources



• Use Font Resource

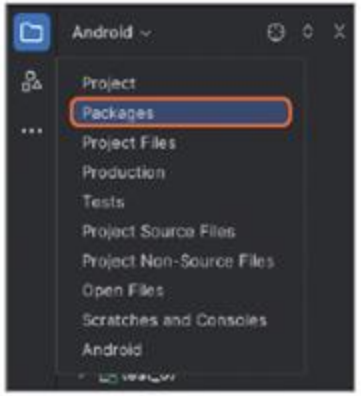
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="HelloWorld"
    android:textSize="20dp"
    android:fontFamily="@font/pacifico" />
```

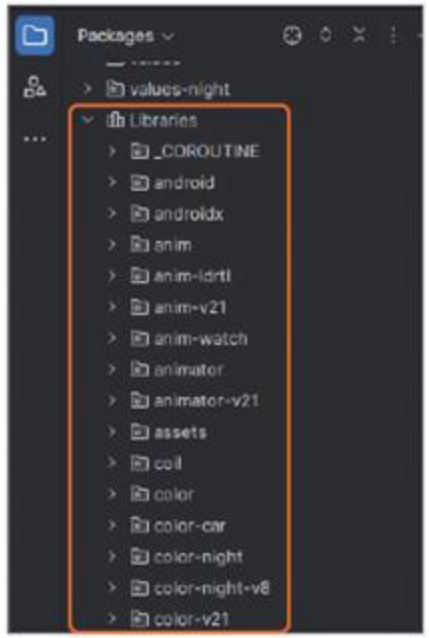
▶ Execution Result

Types and Characteristics of Resources

Types and Characteristics of Resources

- Using Platform Resources
  - Resources provided by the Android platform
  - Registered in the platform library android.R file





• Use Platform Resource in Code

```
binding.imageView.setImageDrawable(ResourcesCompat.getDrawable(resources,
    android.R.drawable.alert_dark_frame, null))
binding.textView.text=getString(android.R.string.emptyPhoneNumber)
```

• Use Platform Resource in XML

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@android:drawable/alert_dark_frame"/>

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@android:string/emptyPhoneNumber"/>
```

Drawable and Units

Drawable and Units

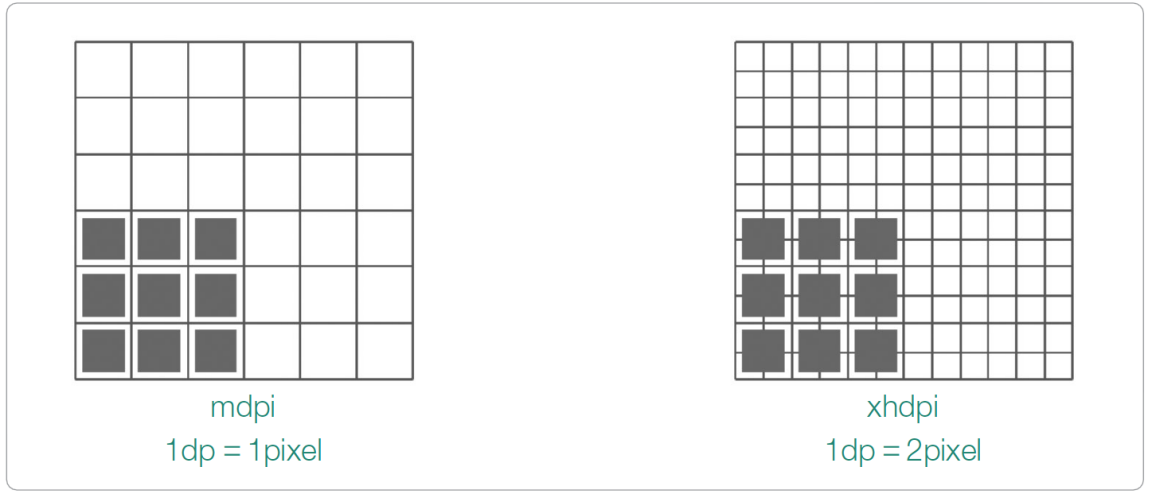
- Resource Usage
  - Drawable: image resources
  - Mipmap: used for app icons
  - Strings: for multilingual support
- dp (Density-independent Pixel)
  - Android devices have different numbers of horizontal and vertical pixels
  - To represent size, Android uses the virtual pixel unit dp
  - The actual pixel size changes depending on screen density (DPI)
- DPI (Dots Per Inch)
  - DPI: the number of pixels in a 1-inch (2.54 cm) square
  - Android uses 160 DPI as the default, called mdpi

Expression	Number of pixels per inch	-
ldpi	120	Not used
mdpi	160	1dp = 1pixel
hdpi	240	
xhdpi	320	1dp = 2pixel
xxhdpi	480	1dp = 3pixel
xxxhdpi	640	1dp = 4pixel

Drawable and Units

Drawable and Units

- DP
  - An independent unit used in Android to display the same size on the screen regardless of resolution
  - Example: A 3dp × 3dp square looks the same size on both an mdpi smartphone and an xhdpi smartphone

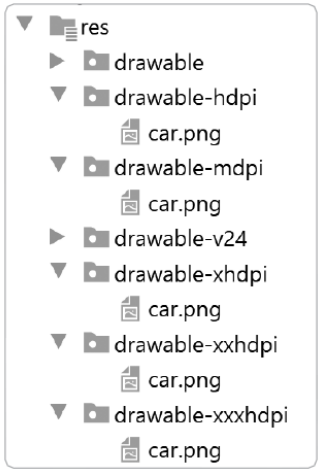


- SP
  - A unit used to represent text size
  - Using sp (Scale-independent Pixels) allows text to zoom in or out without affecting other widgets, so only the text size changes

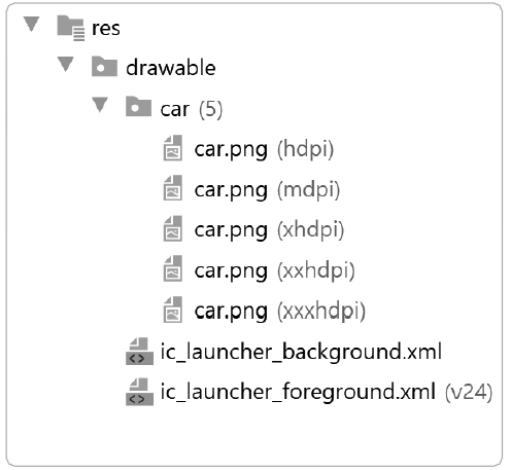
Drawable and Units

Drawable and Units

- Drawable Directory Structure
  - In examples, the drawable directory was used for convenience when explaining widgets
  - In practice, due to the DPI structure, images must be placed in the corresponding drawable directory for each resolution
  - In real projects, when collaborating with designers, images are provided in five different sizes, one for each directory
  - In Android Studio, setting the left navigation panel to Project view is more convenient for managing resources like images, since it shows the full directory structure



▲ Project View



▲ Android View

Q&A

aiclasscau@gmail.com