Basic Layout

Mobile App Programming
Fall, 2024

What we learn today?

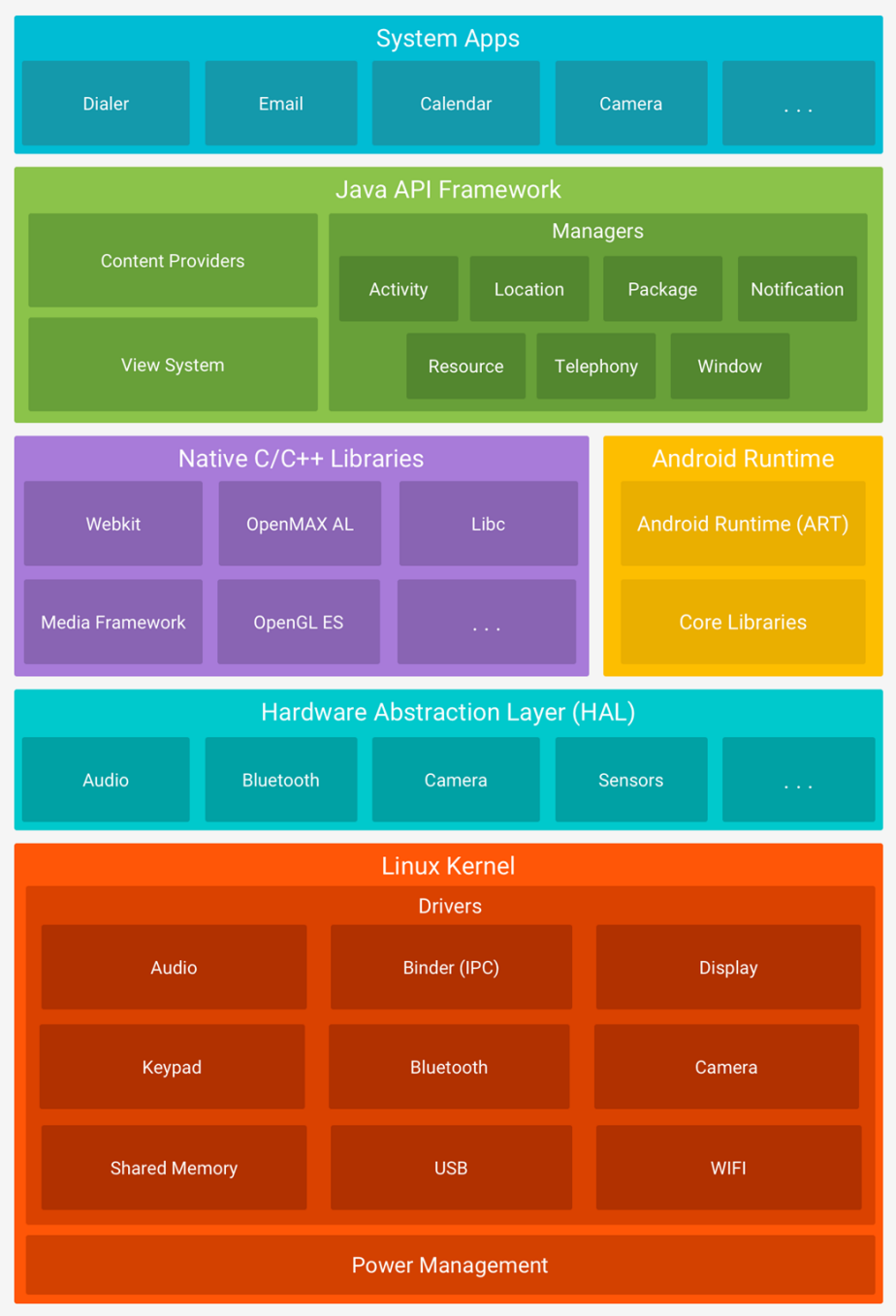
- Introduction to Android
- Android basic layout
  - View & Layout(ViewGroup)
  - Text, Image, Button
- Package Name : edu.skku.map.week3
  - Create project which contains empty activity
  - **Minimum SDK must be API 29 (Android 10)**

Introduction to Android

Android is..

- A mobile operating system developed by **Google**
- Based on a modified version of the **Linux kernel** and other **open source software.**
- Designed primarily for touchscreen mobile devices such as smartphones and tablets.
- Further developed for Android TV, Android Auto, and Android Wear
- Executing an application written in Java or **Kotlin** programming language

Android Platform Architecture



Glossary of Android

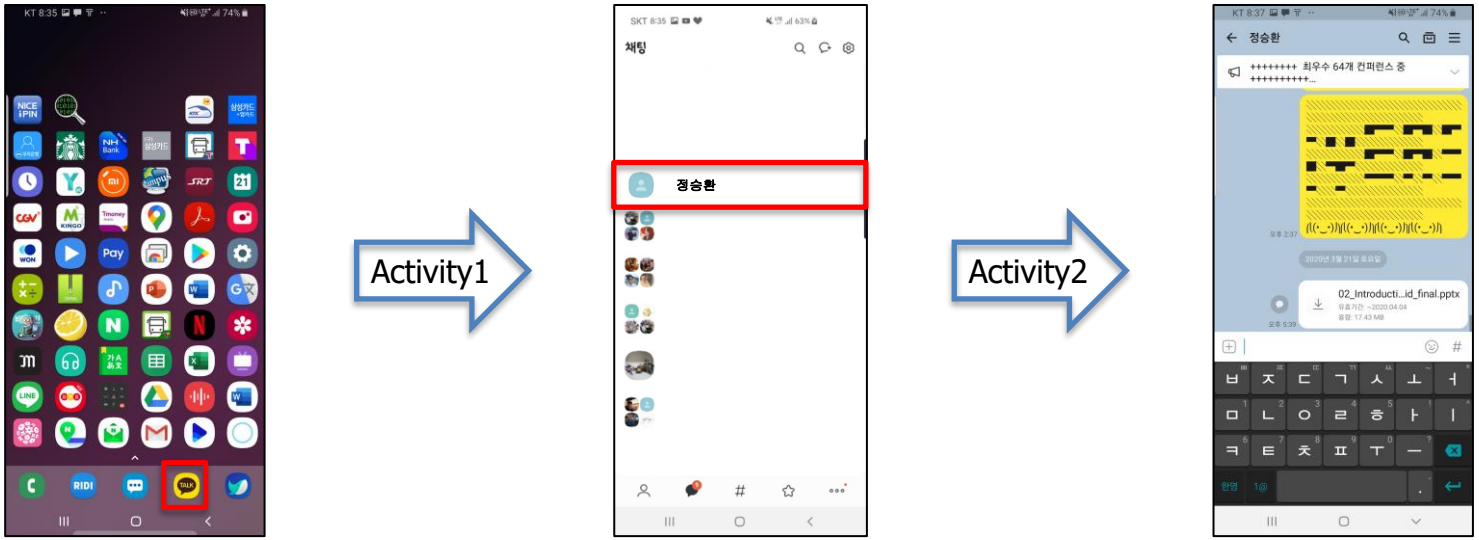
4 core components

- **Activity**
  - A single user-based task, usually, but not always, containing views
- **Service**
  - Background process responding local or remote application requests
- **Broadcast receiver**
  - Component receiving notifications from other activities
- **Content Provider**
  - A component that serves data to other application
- **Intent**
  - A messaging object to request an action from another app component
- **Context**
  - Object containing the global state of an application environment
- **Layout, View & View group**
  - Visual arrangement of views and view groups, UI

Glossary of Android - Activity

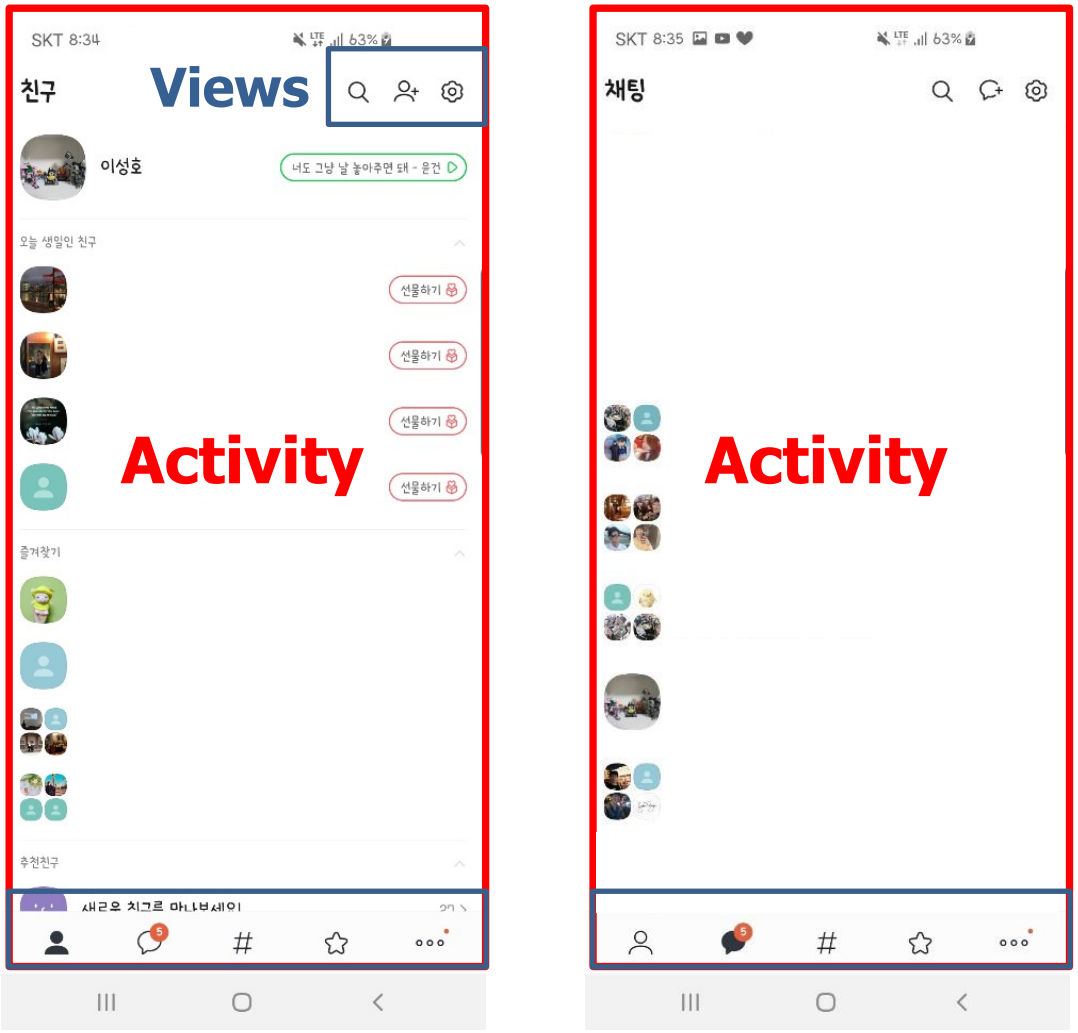
- **Activity**
  - An interface responsible for interacting with the user.
  - Android apps can consist of multiple activities, each of which means one screen.

When you want to chat with friend…



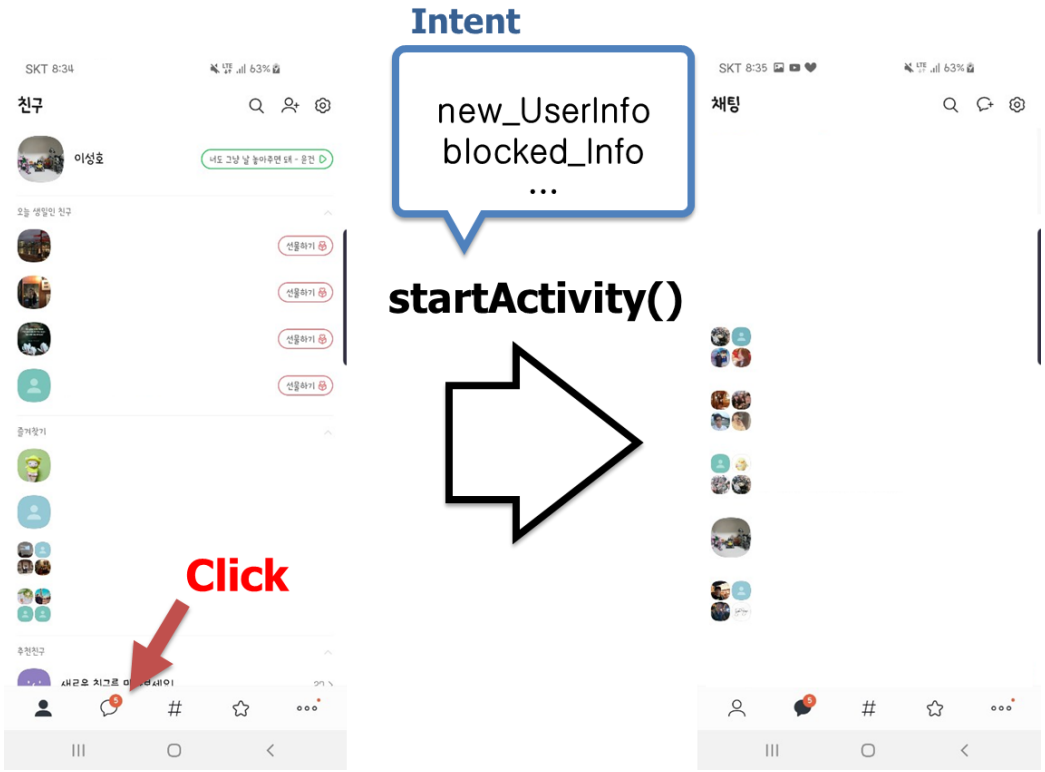
Click Icon at Home **Click your friend tab**

Glossary of Android - Activity



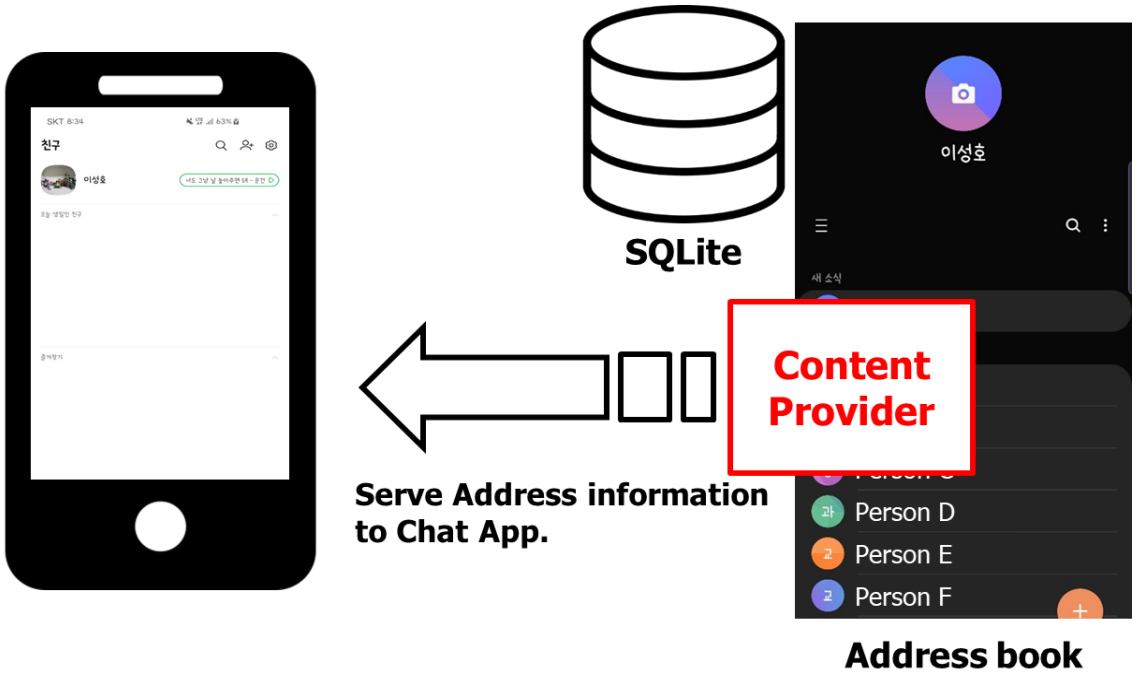
Glossary of Android - Intent

- **Intent**
  - A messaging object to request an action from another app component. (ex. Message to start next activity)



Glossary of Android - Content Provider

- **Content Provider**
  - Components that enable data sharing between apps
  - For example, if you import and use data from a contact app or gallery app, use a content provider.

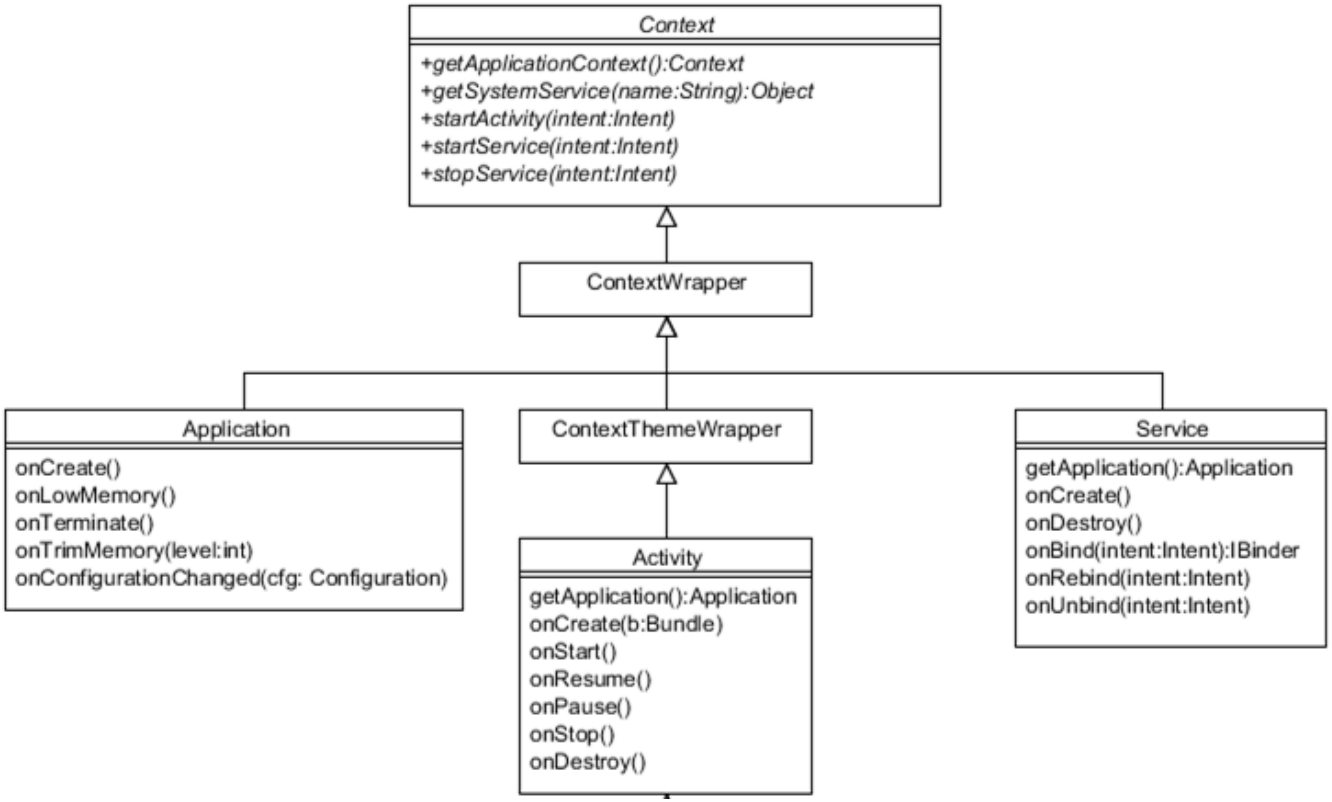


Glossary of Android - Broadcast Receiver

- **Broadcast Receiver**
  - A component that receives and processes broadcast messages from a system or other application.
  - Ex) System events (e.g., low battery, network connection status change) or events sent by apps (e.g., notification arrival)

Glossary of Android - Context

- **Context**
  - Object containing the environmental information of an application



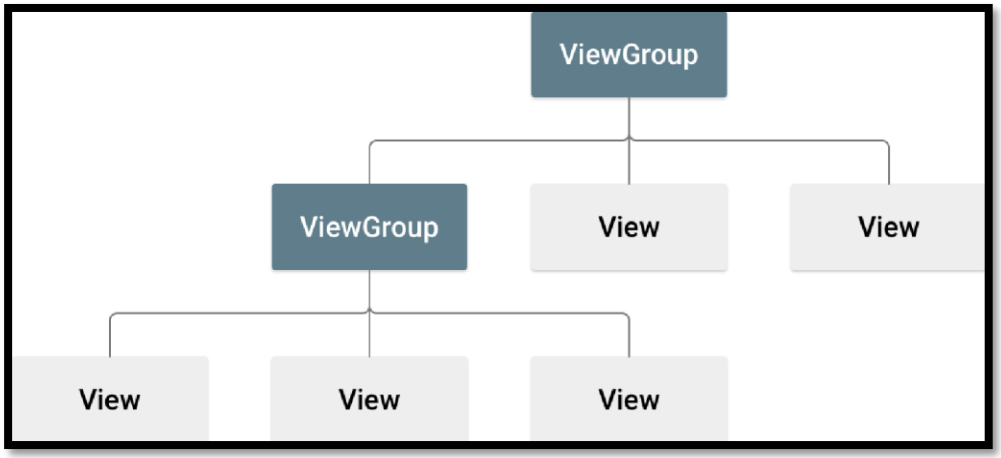
Glossary of Android - Gradle, Manifest

- **Gradle**
  - Gradle is an open-source build automation tool focused on flexibility and performance.

- **Manifest**
  - The manifest file describes essential information about your app to the Android build tools, the Android operating system.
  - Role:
    - Define permissions that the app needs to access protected parts of the system. (Internet, camera, sensors)
    - Define name, icons, themes, and minimum SDK version of application.
    - Define hardware and software features the app requires

Glossary of Android - Layout

- Layout declares UI elements in **XML file** (app → res → layout directory)
- All elements in the layout are built using a **hierarchy** of **View** and **ViewGroup** objects.
  - A **View** usually draws something the user can see and interact with
  - A **ViewGroup** is an invisible container that defines the layout structure for View and other ViewGroup objects



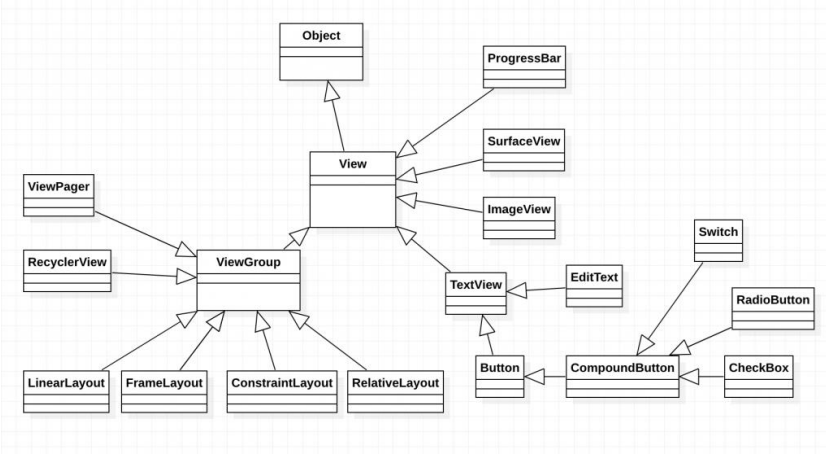
← **Layout**

Android Basic Layout

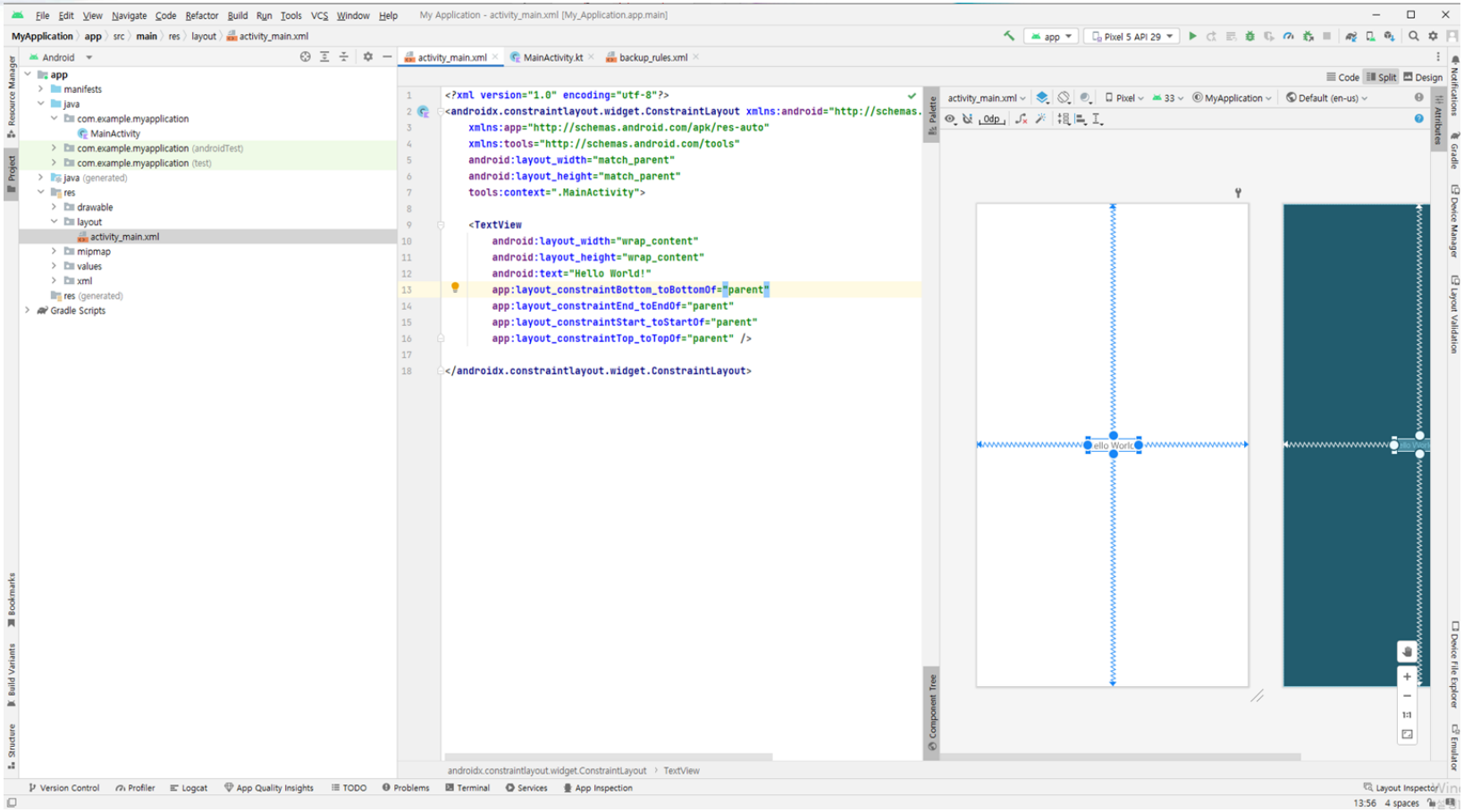
View and ViewGroup

- **View**
  - The single most basic element of the Android UI
  - Individual component responsible for interacting with users
  - Ex) TextView, Button, ImageView, EditText

- **ViewGroup**
  - Elements that can contain multiple Views and ViewGroups
  - Use to organize and place layout of the screen
  - Ex) LinearLayout, ConstraintLayout

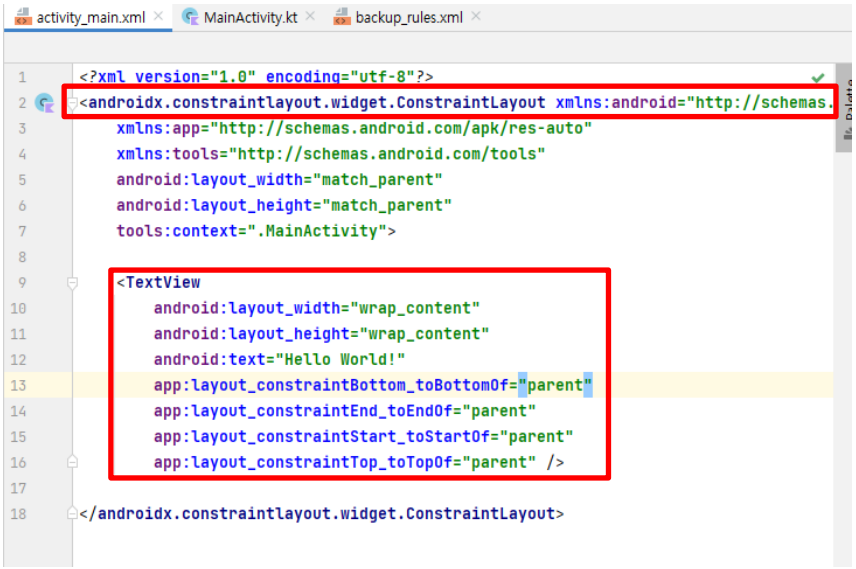


View and ViewGroup



XML file for Layout

- Make it easier to design UI Layouts and screen elements
- Each Layout must contain exactly one root element
- Root element must be a View or ViewGroup



Attributes of Layout

- **ID**
  - Any View object may have an integer ID
  - android:id="@+id/button"
    - @ symbol at the beginning of the string indicates that the XML parser should parse and expand the rest of ID string and identify it as an ID resource
    - + symbol means that this is a new resource name that must be created and added to our resources
  - Each View object can be access with ID
  - Ex) Button button = (Button)findViewById(R.id.button)

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />
```

Attributes of Layout

- **Layout Parameters**
  - Help defining how views are arranged and sized within a layout
  - Ex) width, height, margin, padding, ...

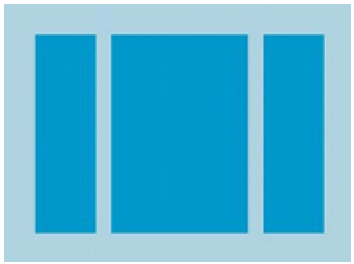
```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />
```

- **wrap_content**: as large as necessary to fit its content.
- **match_parent**: expand to match the size of its parent viewgroup.
- **dp (density independent pixel)** is an abstract unit that is based on the physical density of the screen. These units are relative to a 160 dpi mobile device, so **1 dp** is 1 pixel on a 160 dpi screen (on 640 dpi screen, 1 dp will be 4 pixels)
- **sp (scale independent pixels)** It is similar as dp unit but it is also scaled by the user's font size preference.

Attributes of Layout

- Each subclass of ViewGroup class provides a unique way to display the views
- Ex) LinearLayout, ConstraintLayout, RelativeLayout, etc.

Linear Layout



A layout that organizes its children into a single horizontal or vertical row.

Relative Layout

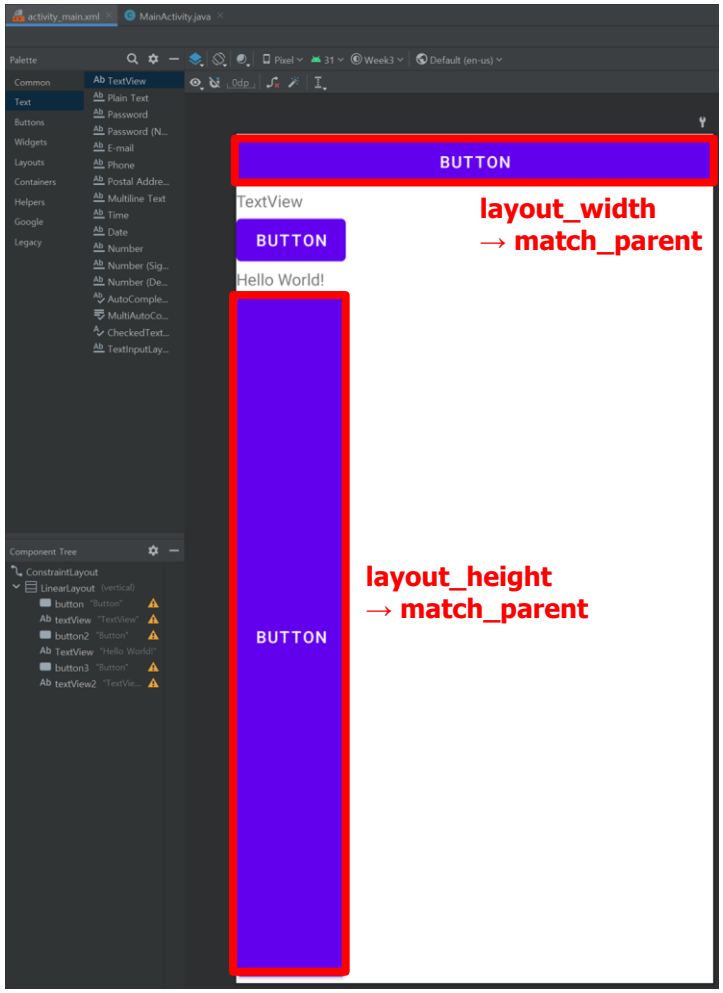
Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).

Web View

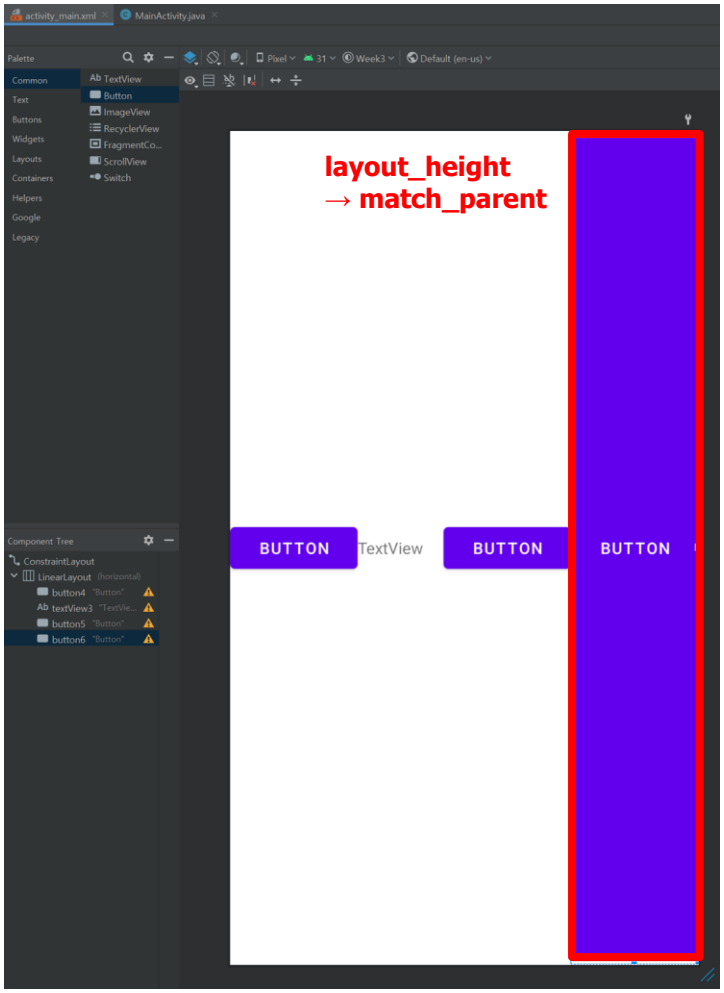
Displays web pages.

LinearLayout

LinearLayout (vertical)



LinearLayout (horizontal)



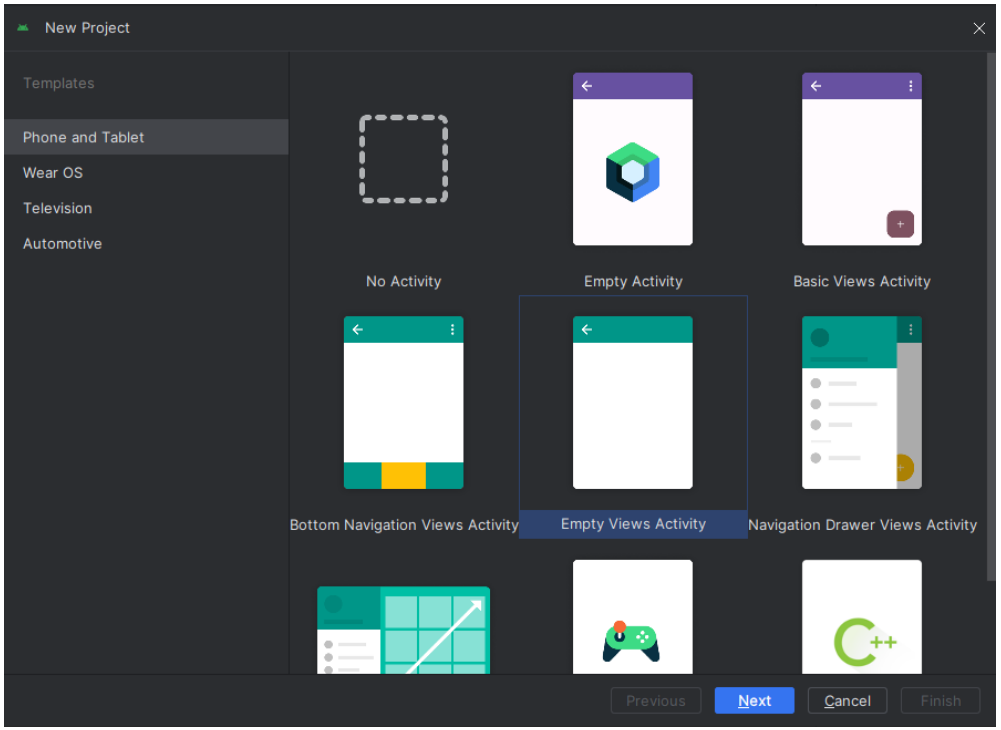
Event Driven Execution

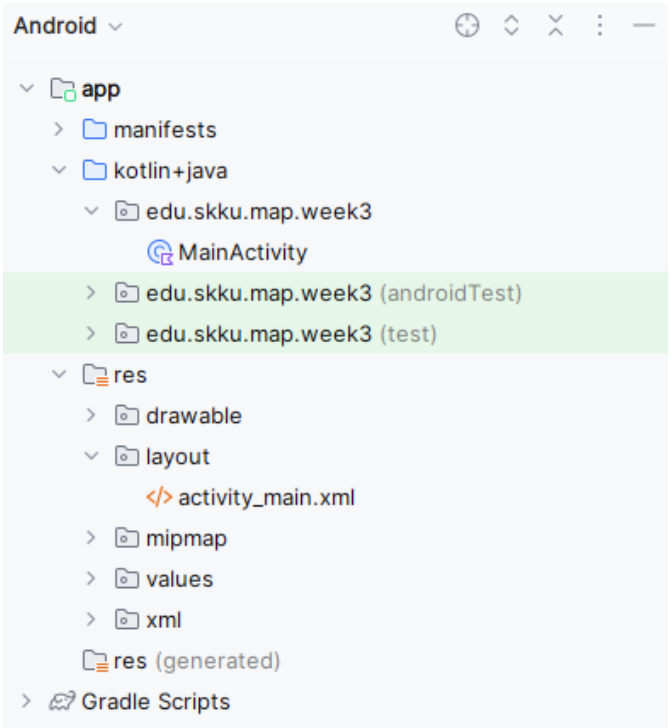
- Android uses **event-driven execution**, event occurs when input is received
- Programmer set a **Callback** method, that will be called when input is received through from **Listener**
- Ex) setOnClickListener: When the button is clicked, the registered Listener detects it and triggers the code inside

```
val myButton = findViewById<Button>(R.id.my_button)

myButton.setOnClickListener { it: View!
    Toast.makeText( context: this@MainActivity, text: "Button Clicked!", Toast.LENGTH_SHORT).show()
}
```

[Lab-Practice #3] Menu Application

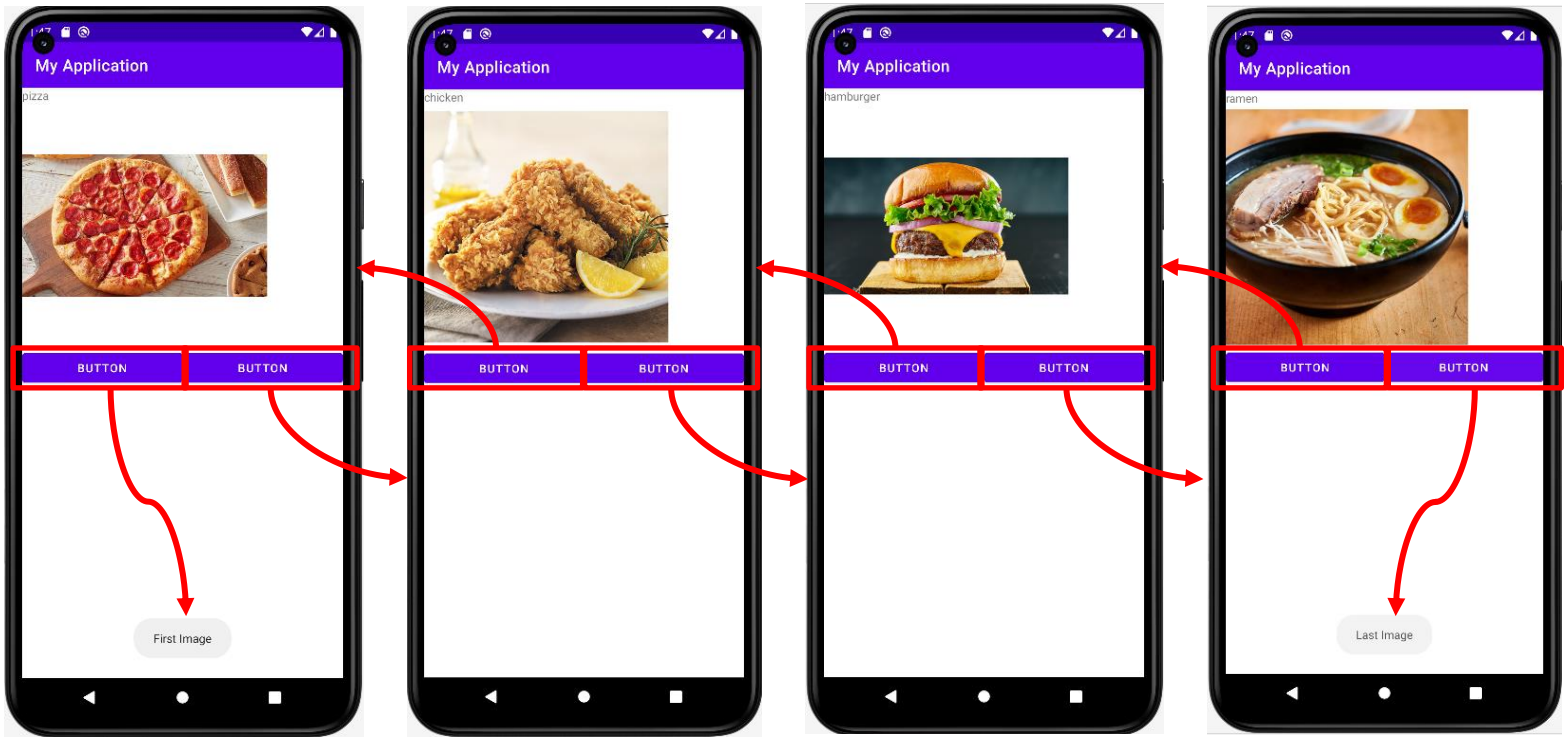
- Create new project with **Empty Views Activity.**
- In this example, you will modify 2 files.
  - App > java > edu.skku.map.week3 > MainActivity.kt
  - App > res > layout > activity_main.xml





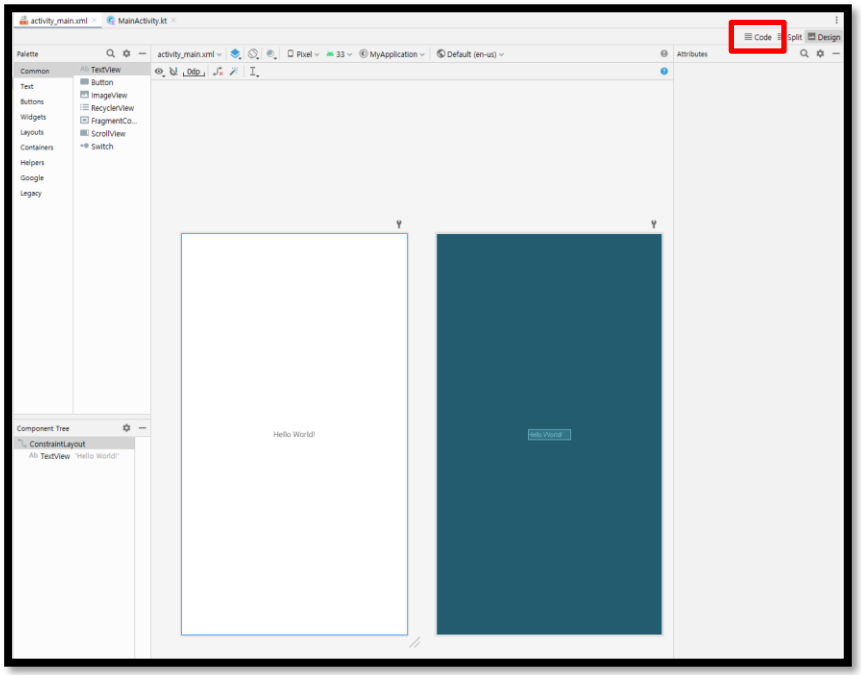
[Lab-Practice #3] Menu Application

- Print “Menu name” (TextView) and “Menu Image” (ImageView)
- Change the menu by clicking the “Left” or “Right” button
- If the menu is first or last one, print toast shortly
- Please use 4 images from ICAMPUS



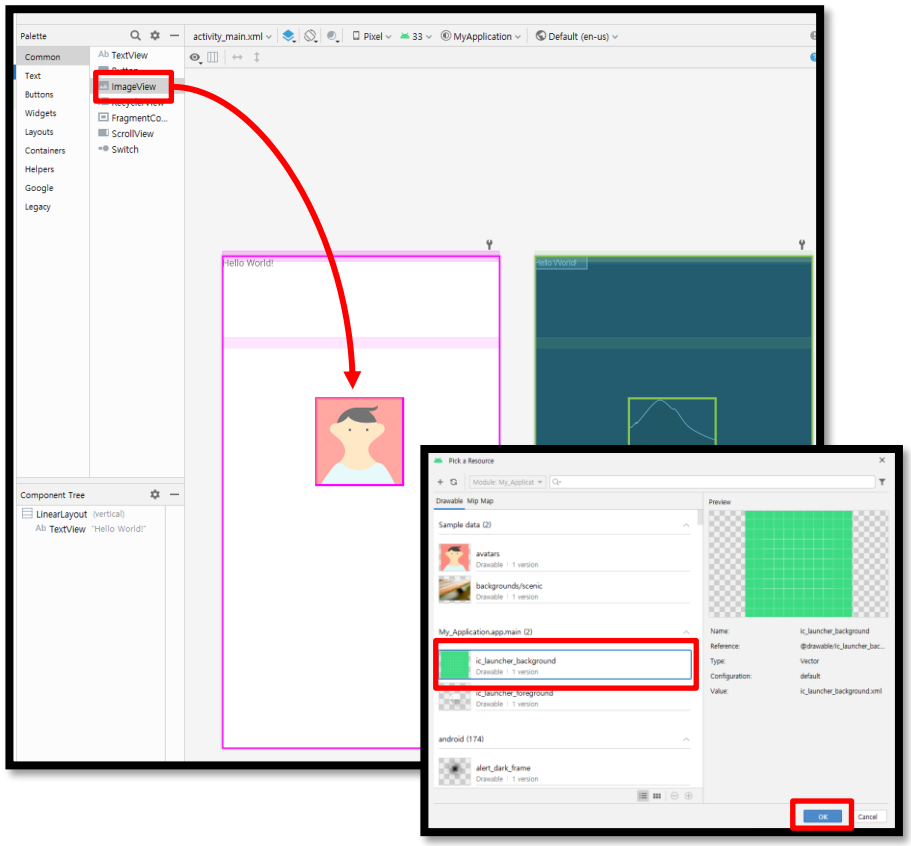
[Lab-Practice #3] Menu Application : Layout Design

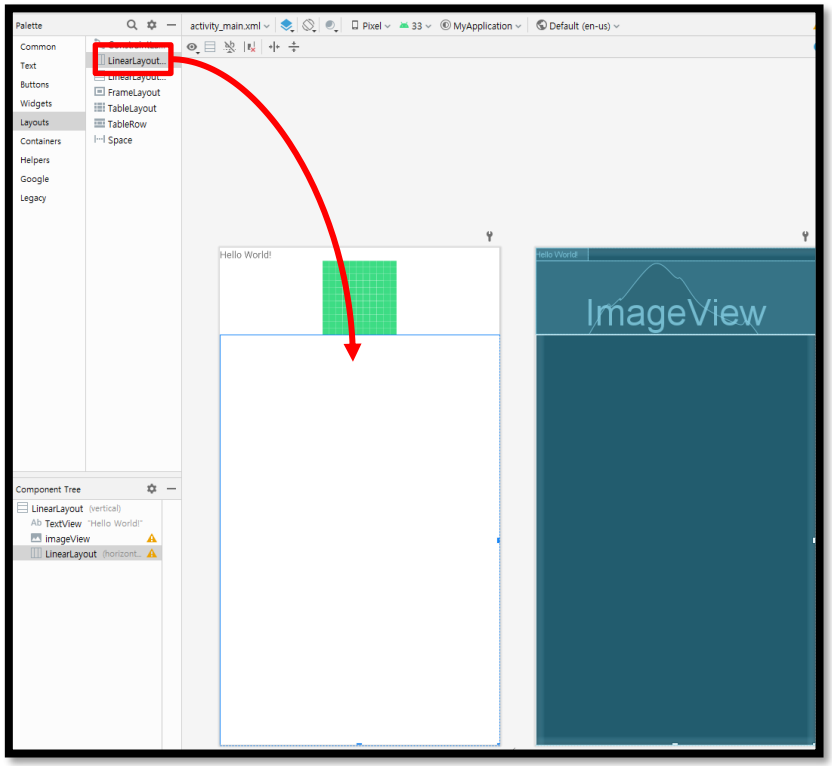
- Open “activity_main.xml” file
- Click the “**Code**” button, and change the layout to “LinearLayout”
- Add Orientation attributes to LinearLayout, set it “vertical”



[Lab-Practice #3] Menu Application : Layout Design

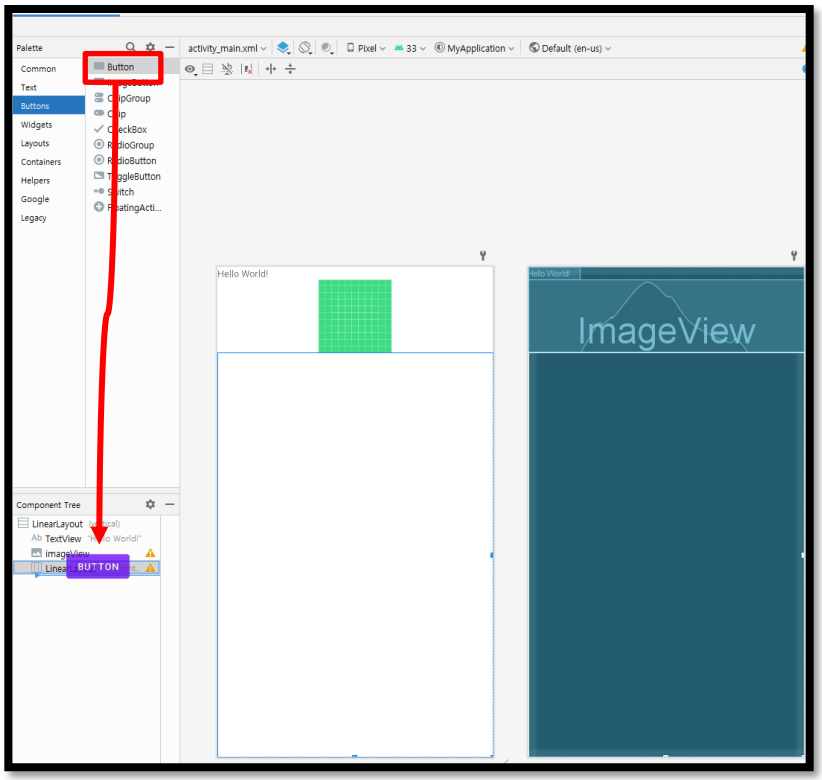
- Click the "**Design**" button (next to "**Code**" button)
- Drag ImageView, and LinearLayout (horizontal)

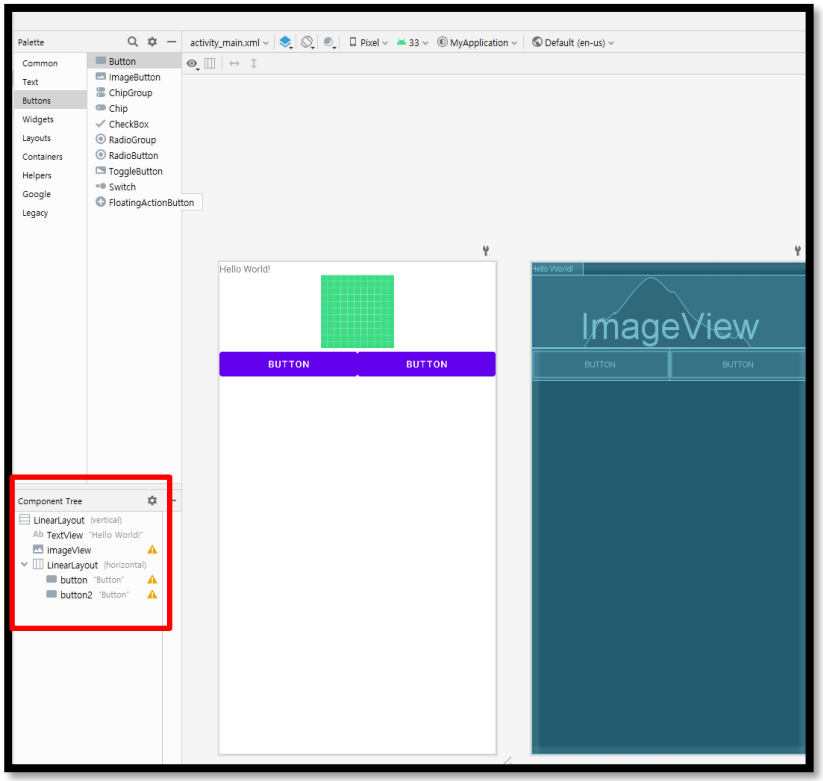




[Lab-Practice #3] Menu Application : Layout Design

- Drag Button, below the LinearLayout (horizontal), twice
- Check the Component Tree





[Lab-Practice #3] Menu Application : Layout Design

- Back to the "**Code**".

- Change the size of image view
  - layout_width → 300 dp
  - layout_height → 300 dp

- Change the size of Linear Layout
  - layout_width → match_parent
  - layout_height → wrap_content

[Lab-Practice #3] Menu Application : Kotlin

- Open the MainActivity.kt file
- To load view (which define in .xml file) in kotlin file, use findViewById<type>()
- At this time, you should check all view has **unique ID** attribute.

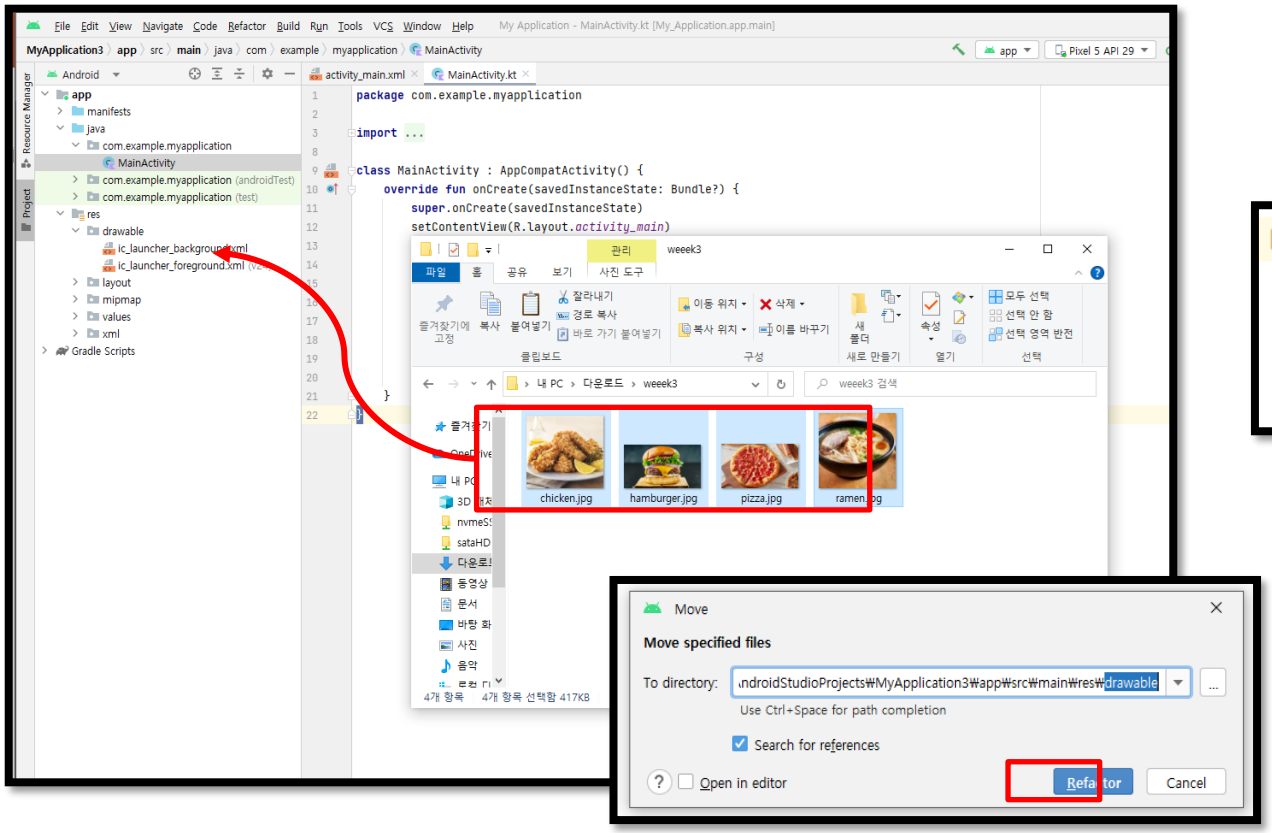
```kotlin
package com.example.myapplication

import ...

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


        var counter = 0
        val textview = findViewById<TextView>(R.id.textView)
        val imageview = findViewById<ImageView>(R.id.imageView)
        val left_btn = findViewById<Button>(R.id.button)
        val right_btn = findViewById<Button>(R.id.button2)

    }
}
```

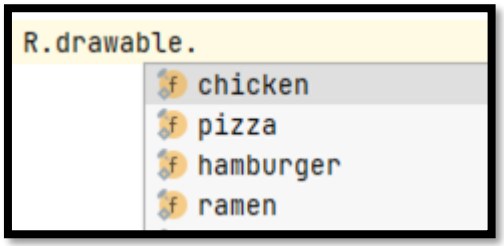
```xml
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<ImageView
    android:id="@+id/imageView"
    android:layout_width="300dp"
    android:layout_height="300dp"
    app:srcCompat="@drawable/ic_launcher_background" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Button" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Button" />
</LinearLayout>
```

```xml
android:id="@+id/textView"
```

[Lab-Practice #3] Menu Application : Kotlin

- To print the specific image in image View, first import image to project.
- Drag images to App > res > drawable directory.
- You can call image using R.drawable.<image_name>

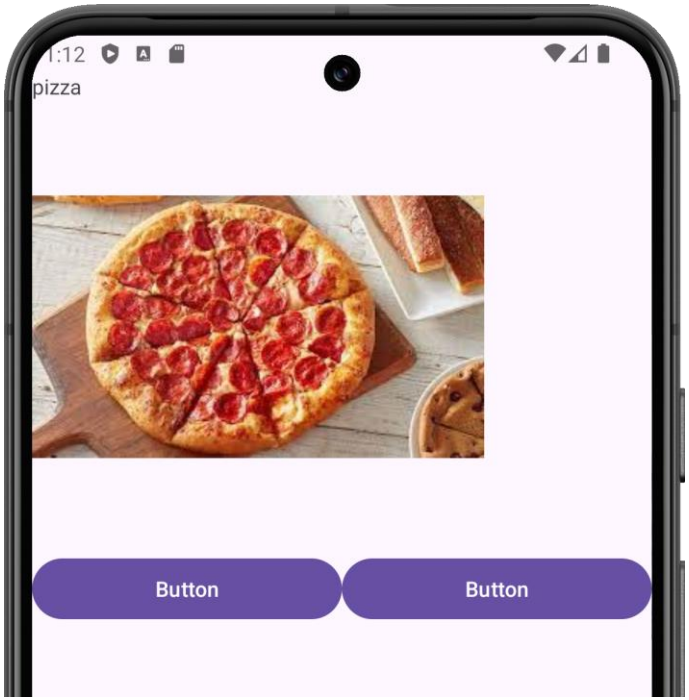




[Lab-Practice #3] Menu Application : Kotlin

- To modify the text, set text attribute of the textview instance to specific string.
- To change the image, call setImageResource( ) function with drawable id.

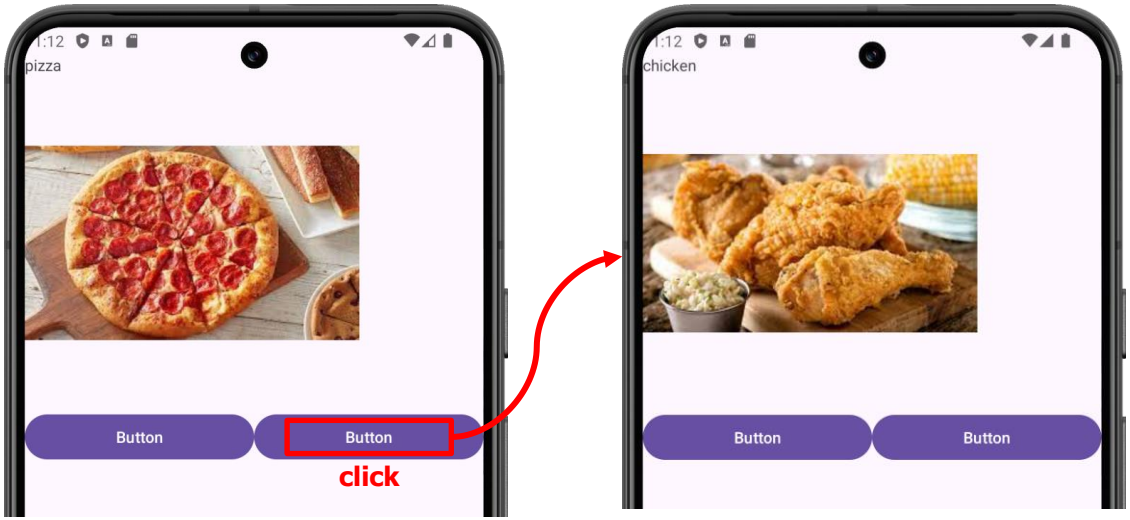
```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)


        var counter = 0
        val textview = findViewById<TextView>(R.id.textView)
        val imageview = findViewById<ImageView>(R.id.imageView)
        val left_btn = findViewById<Button>(R.id.button)
        val right_btn = findViewById<Button>(R.id.button2)

        textview.text = "pizza"
        imageview.setImageResource(R.drawable.pizza)

    }
}
```



[Lab-Practice #3] Menu Application : Kotlin

- Make event-driven execution, use btn.setOnClickListner() { } function
- If you click the "btn", the code in { } will be executed

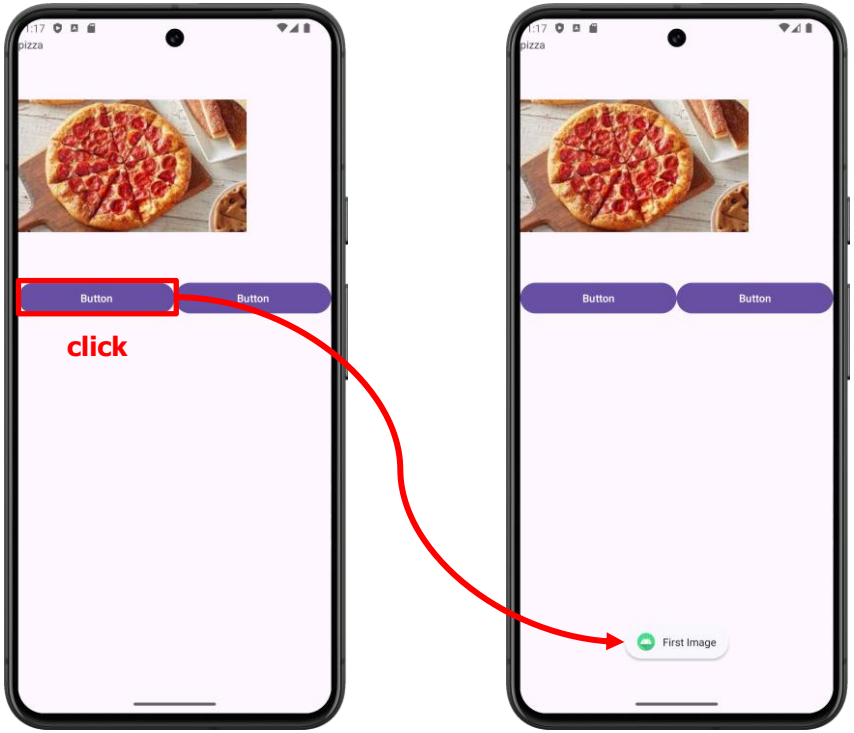
```
right_btn.setOnClickListener{ it: View!
    textview.text = "chicken"
    imageview.setImageResource(R.drawable.chicken)
}
```



[Lab-Practice #3] Menu Application : Kotlin

- Print the toast using Toast.makeText().show() function.
- As a parameter of makeText() you should put context, text which you want to print, and the time length.

```
Toast.makeText( context: this@MainActivity,  text: "First Image", Toast.LENGTH_LONG).show()
```



[Lab-Practice #3] Checking

- The Position of each view is not important
- If you click the button “left” or “right”, the menu text and image must changed
- If you click “left” button on first menu, print toast message “First Image”
- If you click “right” button on last menu, print toast message “Last Image”
- Before you leave the class, please check your example application.