Mobile Programming

Android App Basics

Android App Basics

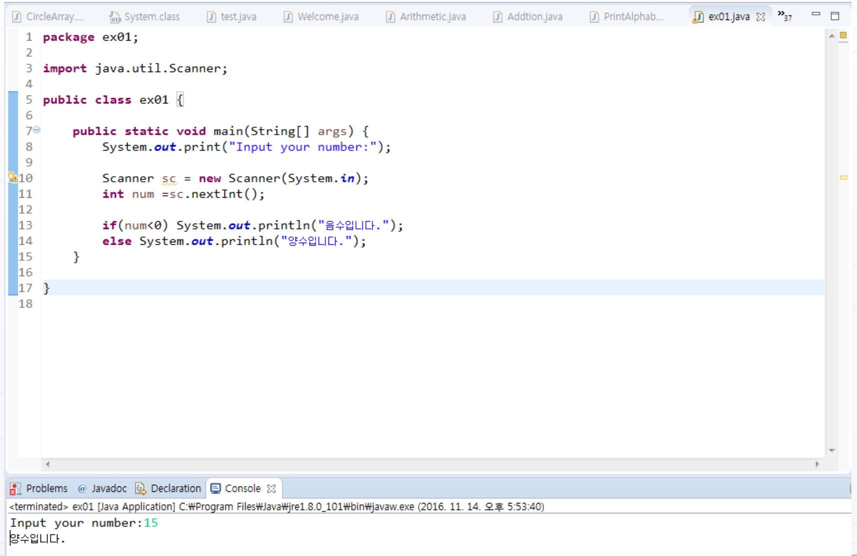
Activity Basics

Compared to Java Console Program? (1/3)

- Mobile application is…
  - Running on operating systems for a mobile device,
  - Utilizing various mobile device features
  - Accessing sensitive data/features based on the user's permission
  - Handling device fragmentations (versions/features/capabilities)
    - Different versions of the same application need to be maintained

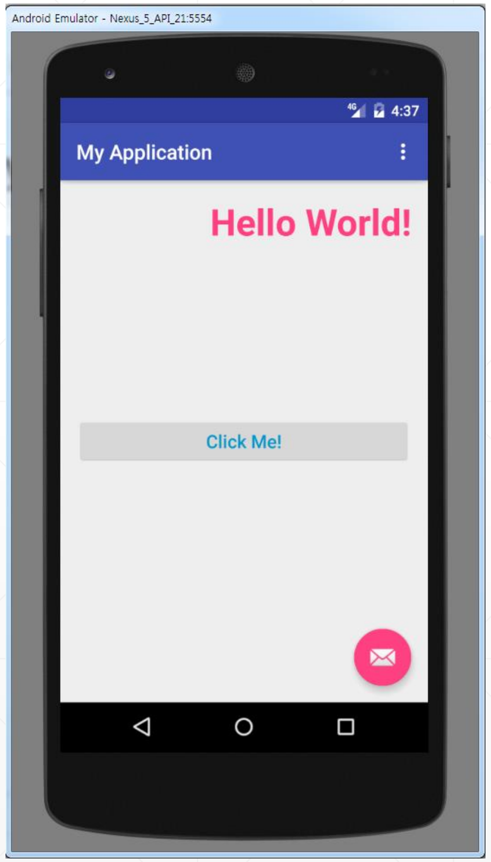
Compared to Java Console Program? (2/3)

- Mobile application is…
  - Operating on a screen with a limited size,
  - Focusing on the interaction with users / other programs,
  - May be terminated at any time by the user or the system,
  - Based on the Graphic user interface (GUI)

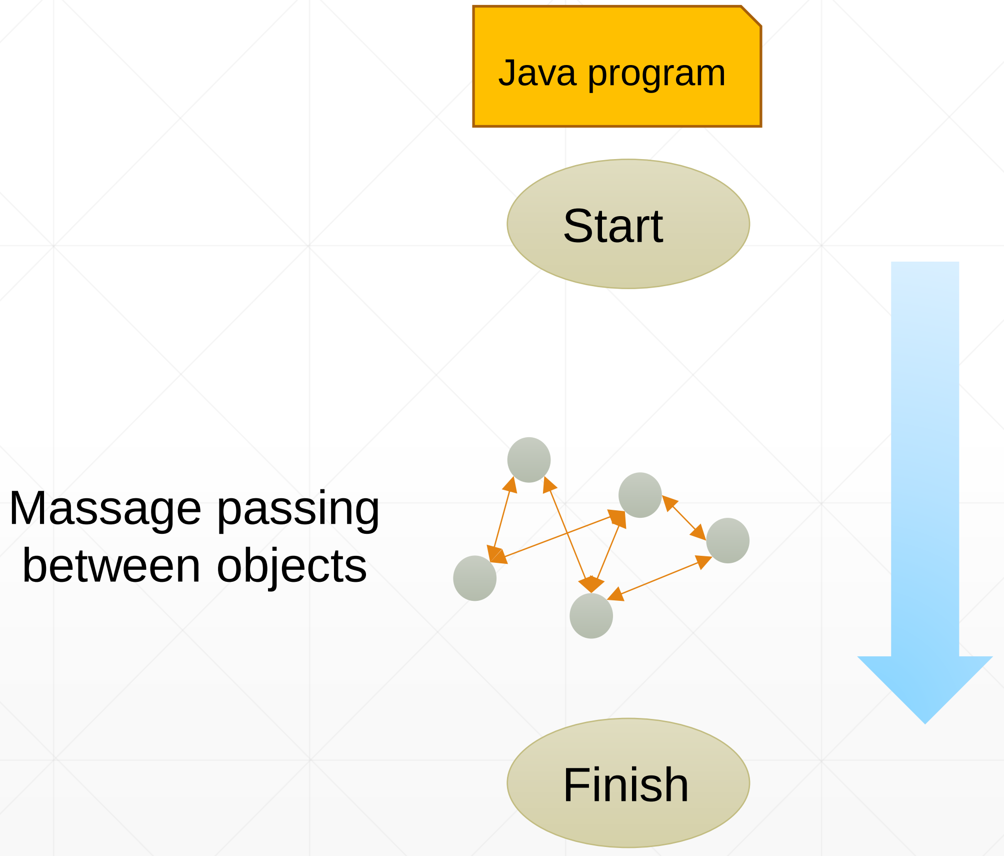


```java
package ex01;

import java.util.Scanner;

public class ex01 {

    public static void main(String[] args) {
        System.out.print("Input your number:");

        Scanner sc = new Scanner(System.in);
        int num =sc.nextInt();

        if(num<0) System.out.println("음수입니다.");
        else System.out.println("양수입니다.");
    }

}
```

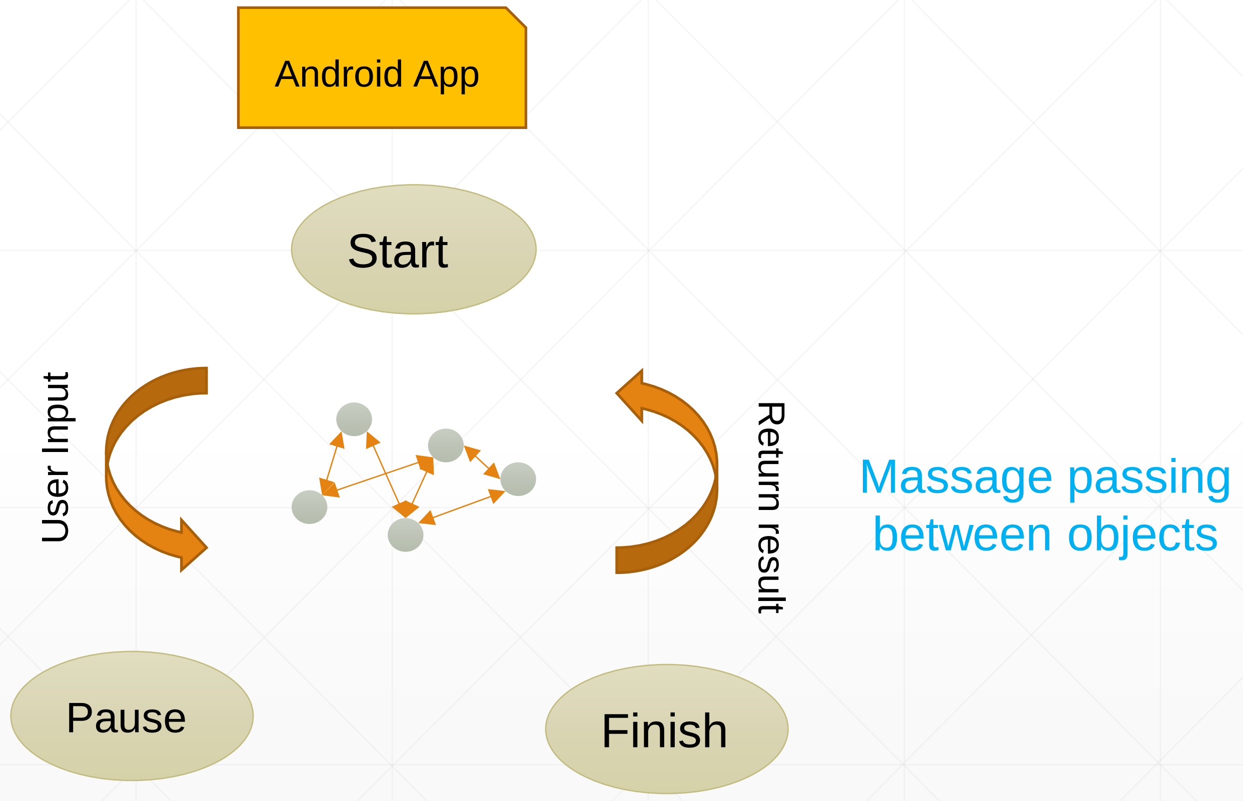
```
<terminated> ex01 [Java Application] C:₩Program Files₩Java₩jre1.8.0_101₩bin₩javaw.exe (2016. 11. 14. 오후 5:53:40)
Input your number:15
양수입니다.
```



Compared to Java Console Program? (3/3)



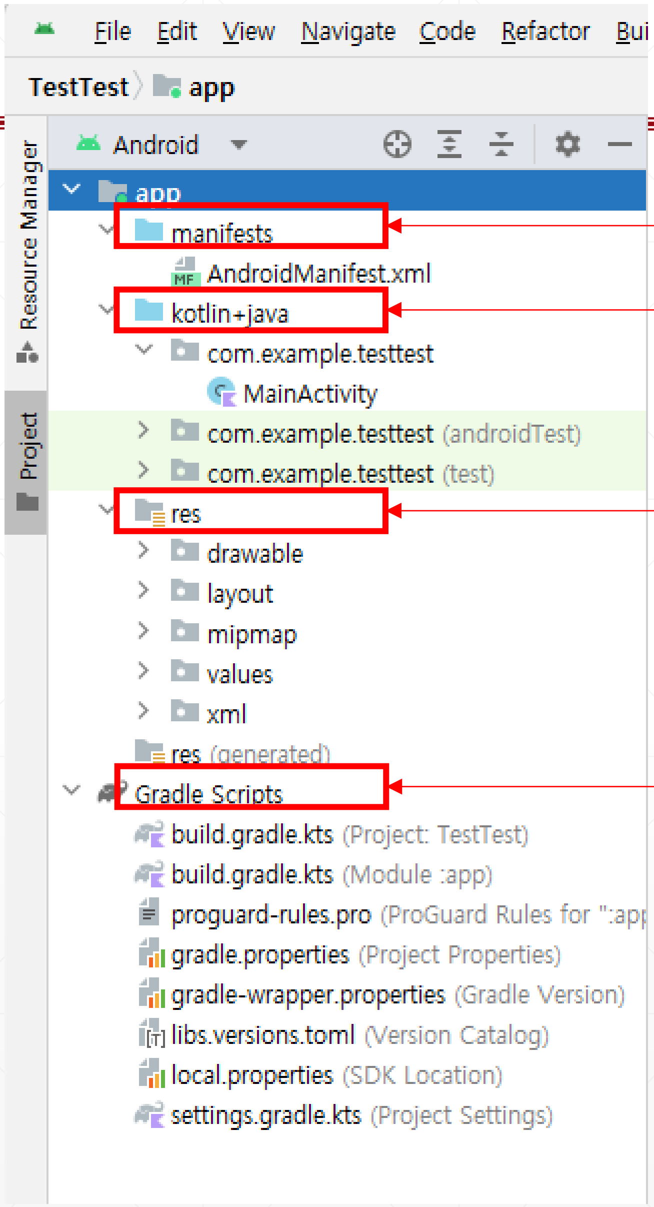
Object-oriented processing



Event-driven processing

Project Structure

- Project view



```
File  Edit  View  Navigate  Code  Refactor  Bui
TestTest 〉 app
Android
app
  manifests
    AndroidManifest.xml
  kotlin+java
    com.example.testtest
      MainActivity
    com.example.testtest (androidTest)
    com.example.testtest (test)
  res
    drawable
    layout
    mipmap
    values
    xml
  res (generated)
Gradle Scripts
  build.gradle.kts (Project: TestTest)
  build.gradle.kts (Module :app)
  proguard-rules.pro (ProGuard Rules for ":app
  gradle.properties (Project Properties)
  gradle-wrapper.properties (Gradle Version)
  libs.versions.toml (Version Catalog)
  local.properties (SDK Location)
  settings.gradle.kts (Project Settings)
```

Application information

Source

Resource (img, layout,etc)

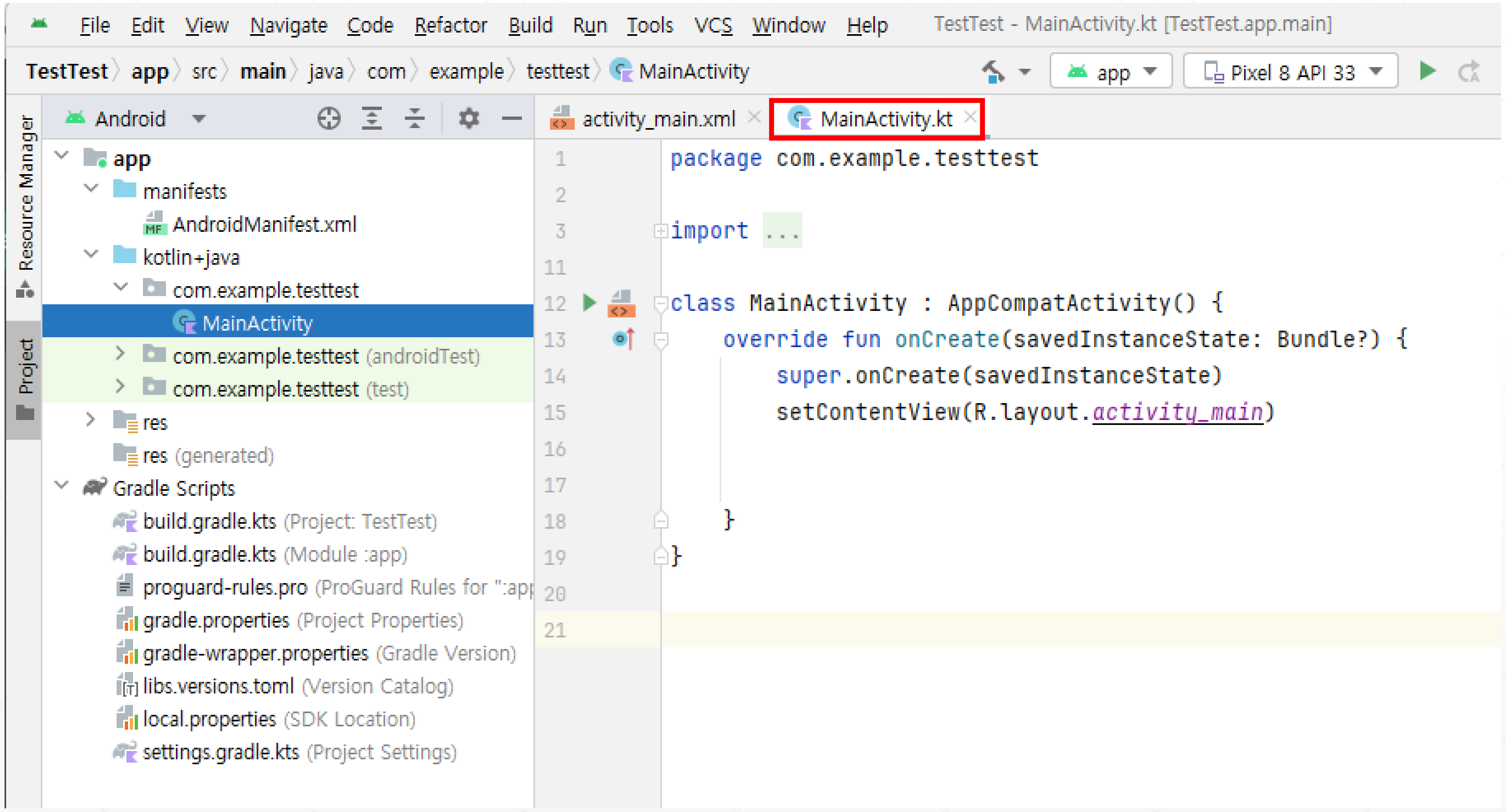
Build

Basic Steps to Build Mobile Applications

- Create a new project
- Edit a layout
- Connect the source code
- Launch your application!

Let’s Edit a Layout! (1/12)

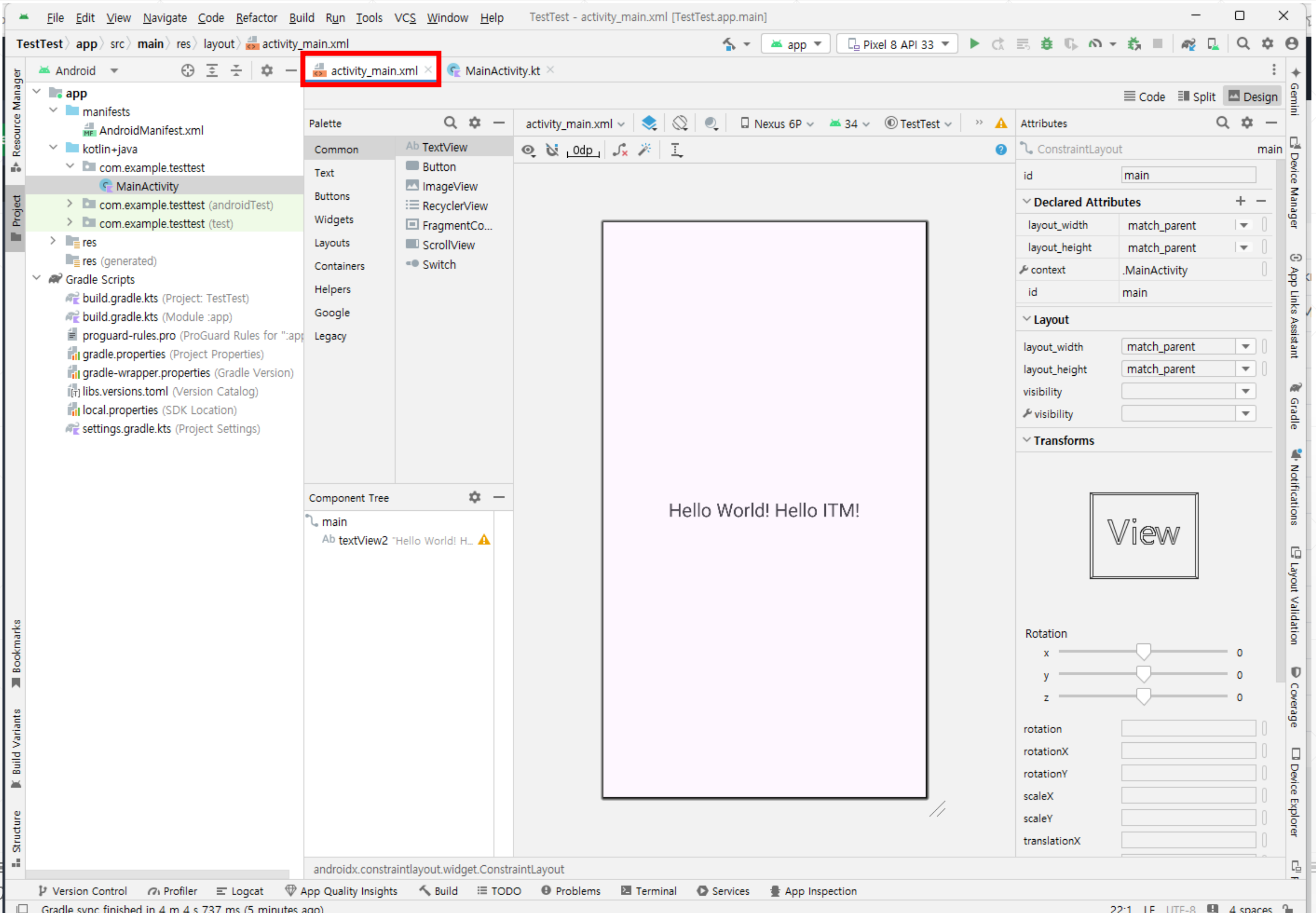
- Code editor for Kotlin files


File
Edit
View
Navigate
Code
Refactor
Build
Run
Tools
VCS
Window
Help
TestTest - MainActivity.kt [TestTest.app.main]
TestTest 〉 app 〉 src 〉 main 〉 java 〉 com 〉 example 〉 testtest 〉 MainActivity
app
Pixel 8 API 33
Android
Resource Manager
Project
app
manifests
AndroidManifest.xml
kotlin+java
com.example.testtest
MainActivity
com.example.testtest (androidTest)
com.example.testtest (test)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: TestTest)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
activity_main.xml
MainActivity.kt

```
package com.example.testtest

import ...

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


    }
}

```

Let's Edit a Layout! (2/12)

- Layout editor for layout xml files



```
app
  manifests
    AndroidManifest.xml
  kotlin+java
    com.example.testtest
      MainActivity
    com.example.testtest (androidTest)
    com.example.testtest (test)
  res
  res (generated)
Gradle Scripts
  build.gradle.kts (Project: TestTest)
  build.gradle.kts (Module :app)
  proguard-rules.pro (ProGuard Rules for ":ap
  gradle.properties (Project Properties)
  gradle-wrapper.properties (Gradle Version)
  libs.versions.toml (Version Catalog)
  local.properties (SDK Location)
  settings.gradle.kts (Project Settings)
```

```
main
  textView2 "Hello World! H...
```

Hello World! Hello ITM!

ConstraintLayout	main
id	main
Declared Attributes	
layout_width	match_parent
layout_height	match_parent
context	.MainActivity
id	main
Layout	
layout_width	match_parent
layout_height	match_parent
visibility	
visibility	
Transforms	
x	0
y	0
z	0
rotation	
rotationX	
rotationY	
scaleX	
scaleY	
translationX	

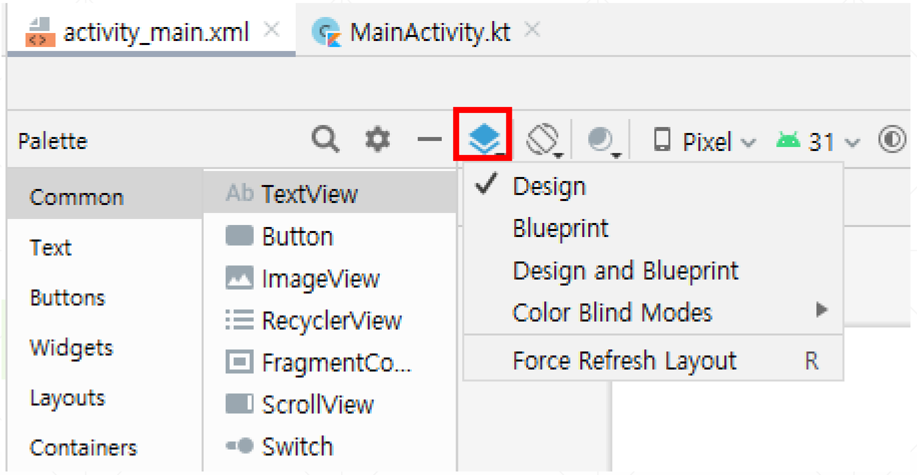
androidx.constraintlayout.widget.ConstraintLayout

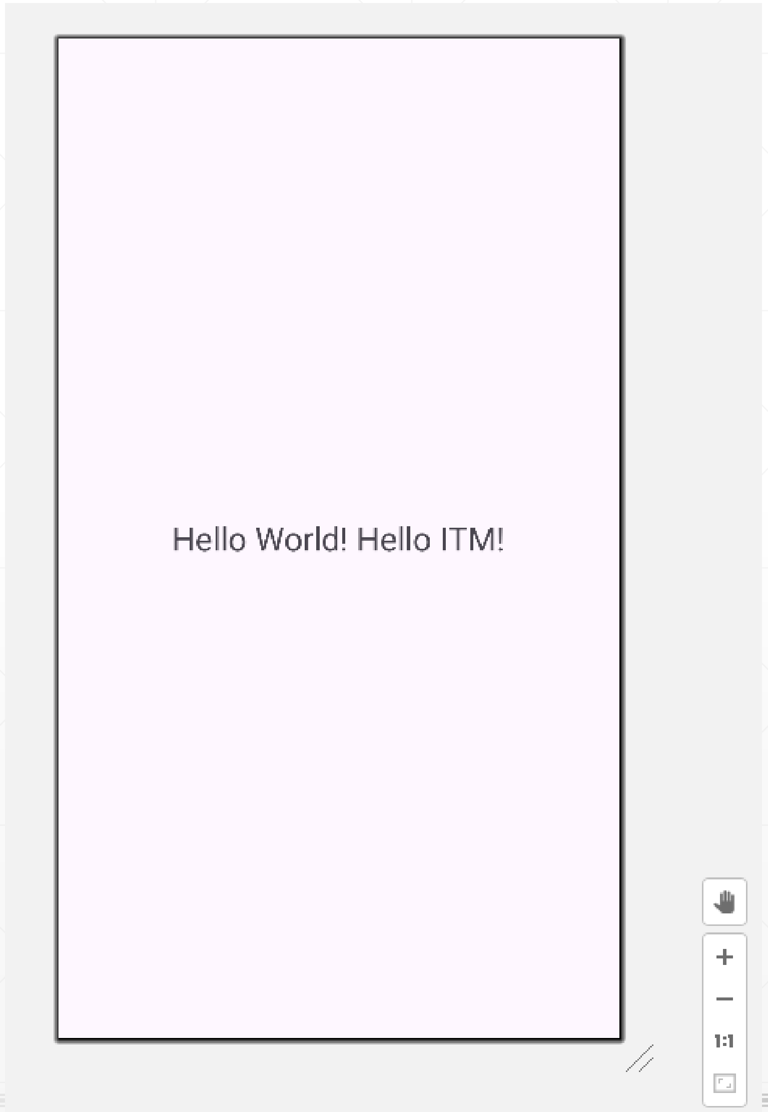
Gradle sync finished in 4 m 4 s 737 ms (5 minutes ago)

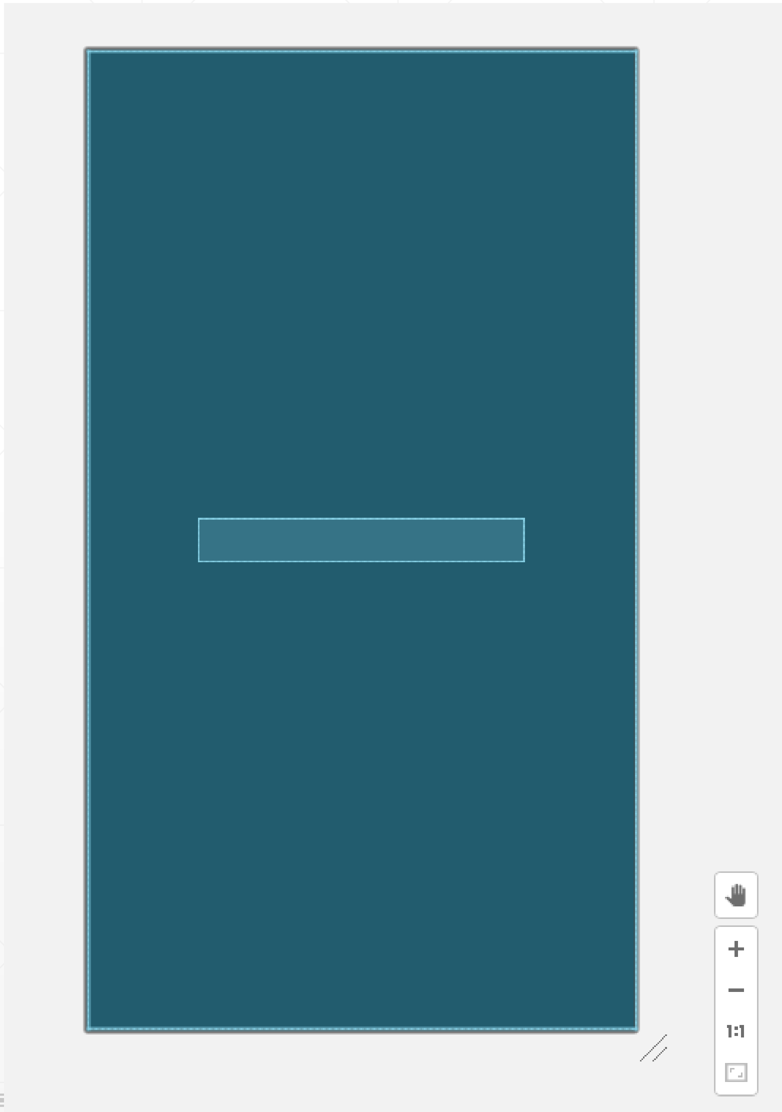
22:1 LF UTF-8 4 spaces

Let’s Edit a Layout! (3/12)

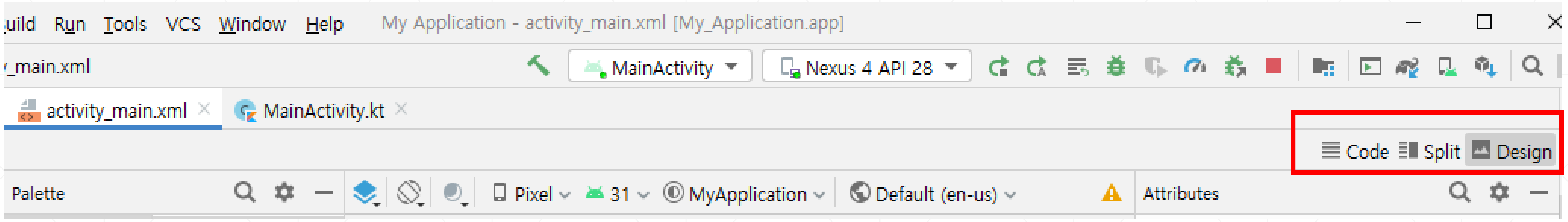
- Design surface

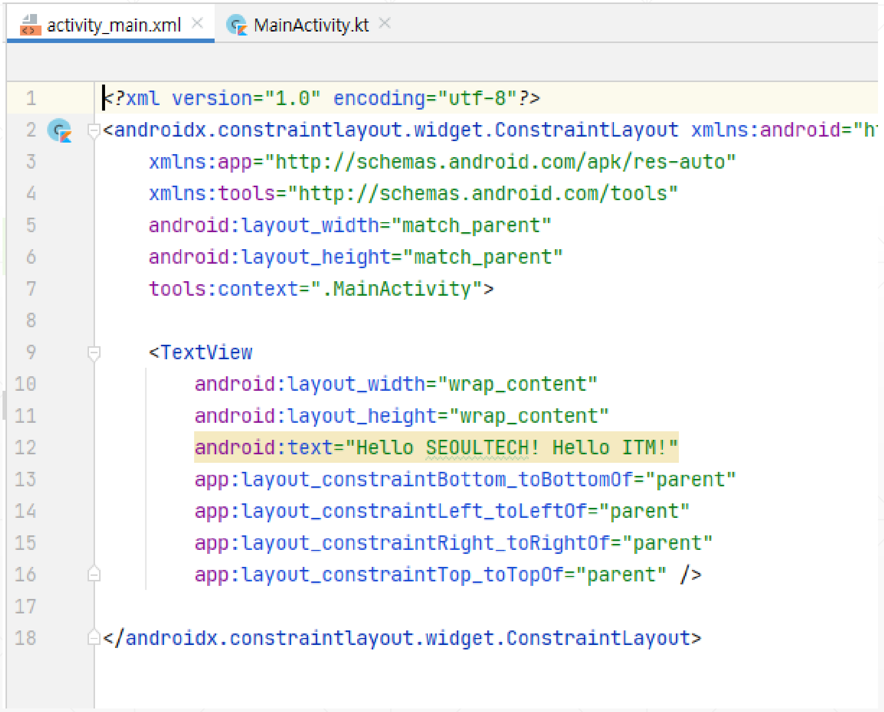




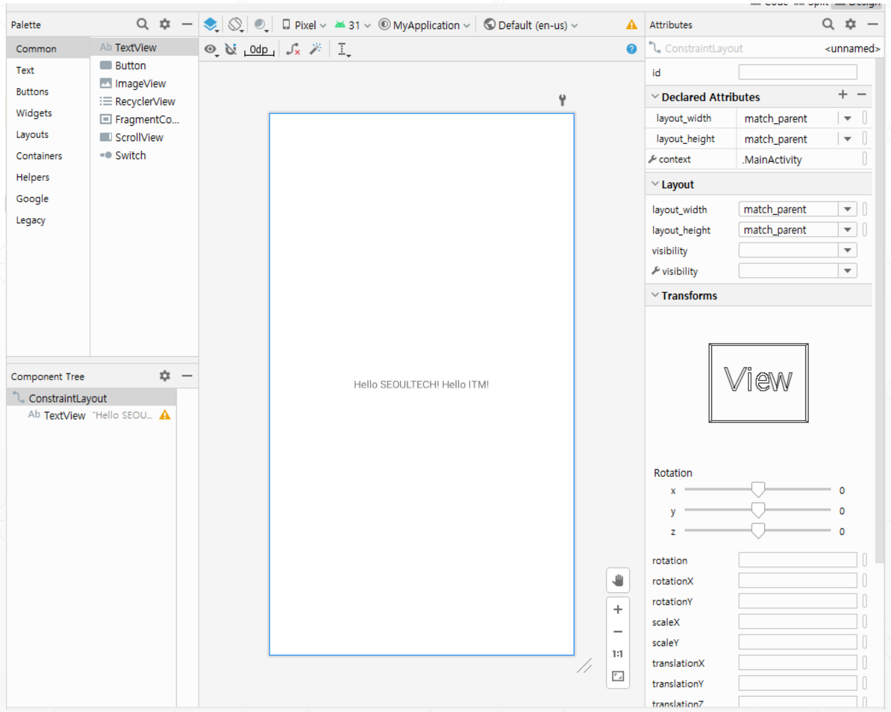


Let’s Edit a Layout! (4/12)





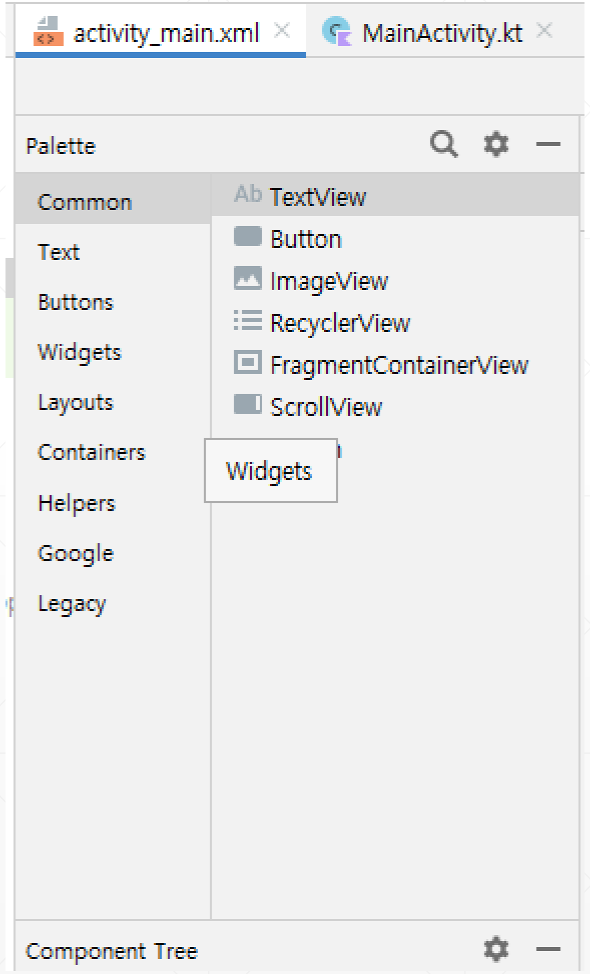
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="h
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello SEOULTECH! Hello ITM!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



layout_width match_parent
layout_height match_parent
context .MainActivity
layout_width match_parent
layout_height match_parent
visibility
visibility

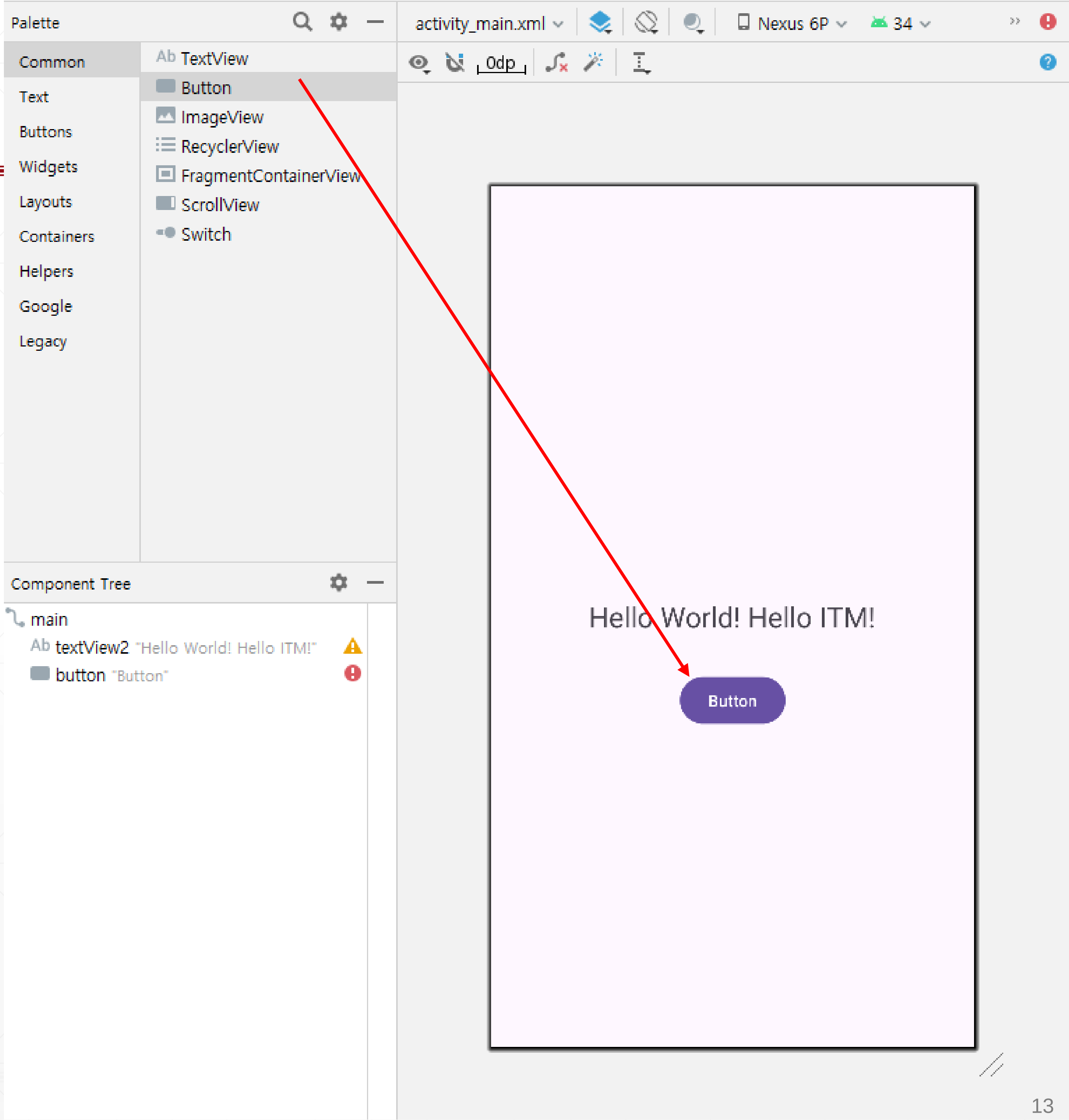
Let’s Edit a Layout! (5/12)

- UI components there~!



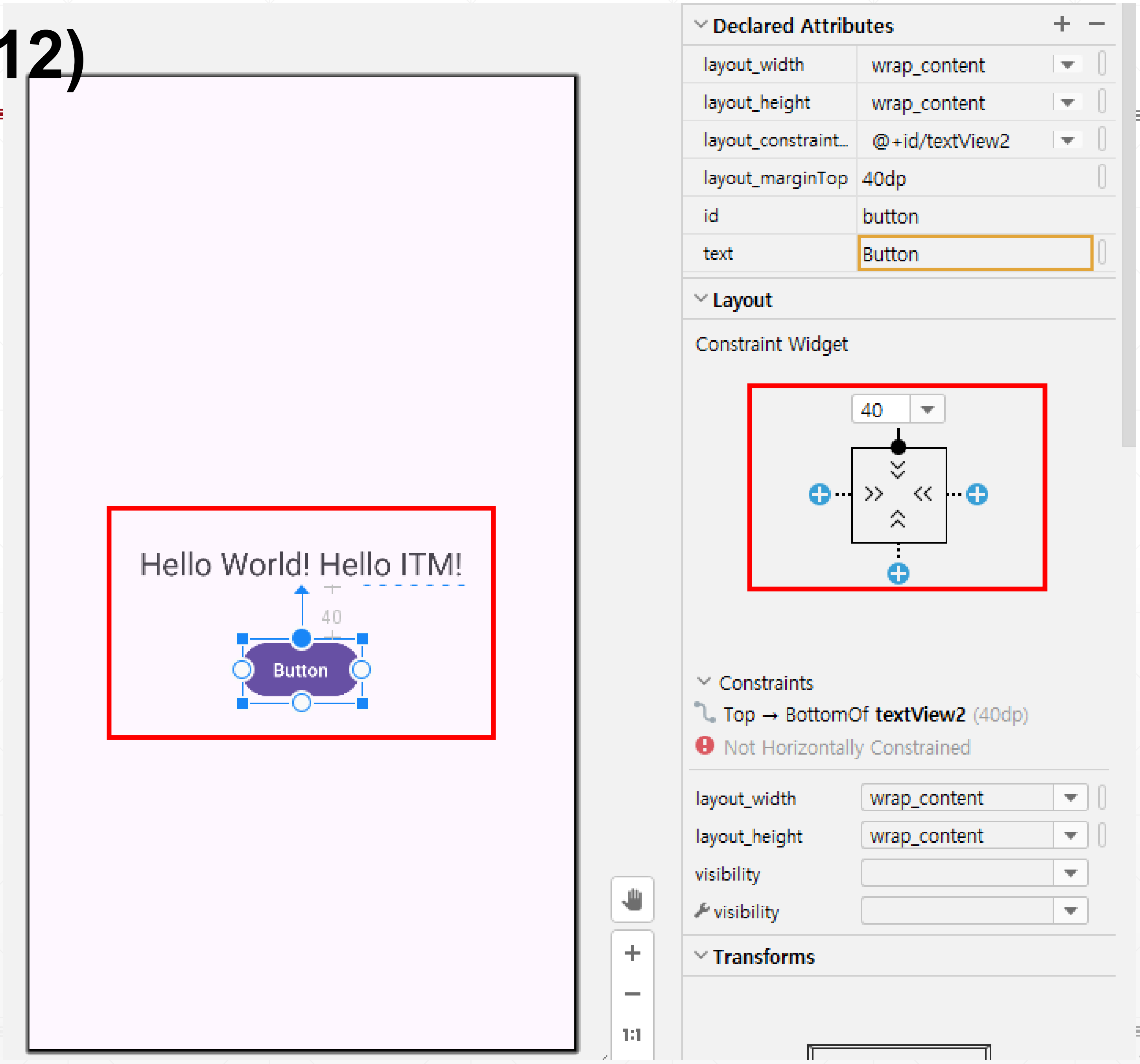
Let's Edit a Layout! (6/12)

- Create a single button
  - Run the application?



Let’s Edit a Layout! (7/12)

- Create a single button
- Connect top to the bottom of textView



Declared Attributes

layout_width	wrap_content
layout_height	wrap_content
layout_constraint…	@+id/textView2
layout_marginTop	40dp
id	button
text	Button

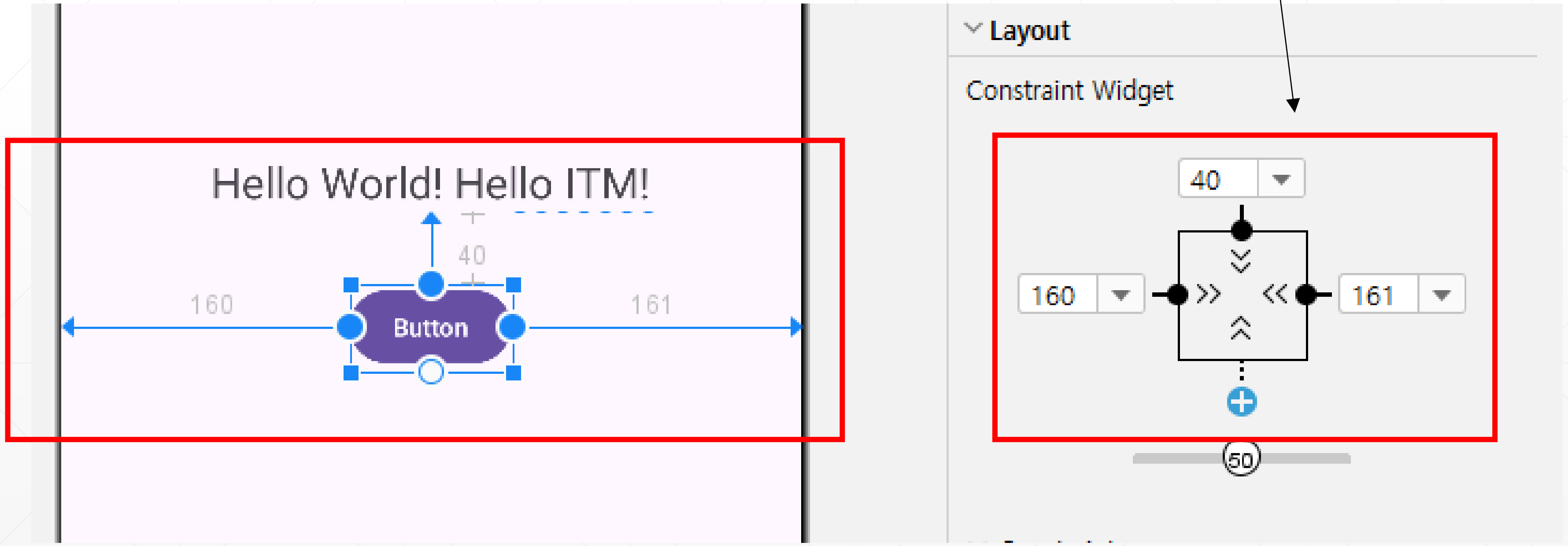
Constraints
Top → BottomOf textView2 (40dp)
Not Horizontally Constrained

layout_width	wrap_content
layout_height	wrap_content
visibility	
visibility	

Let’s Edit a Layout! (8/12)

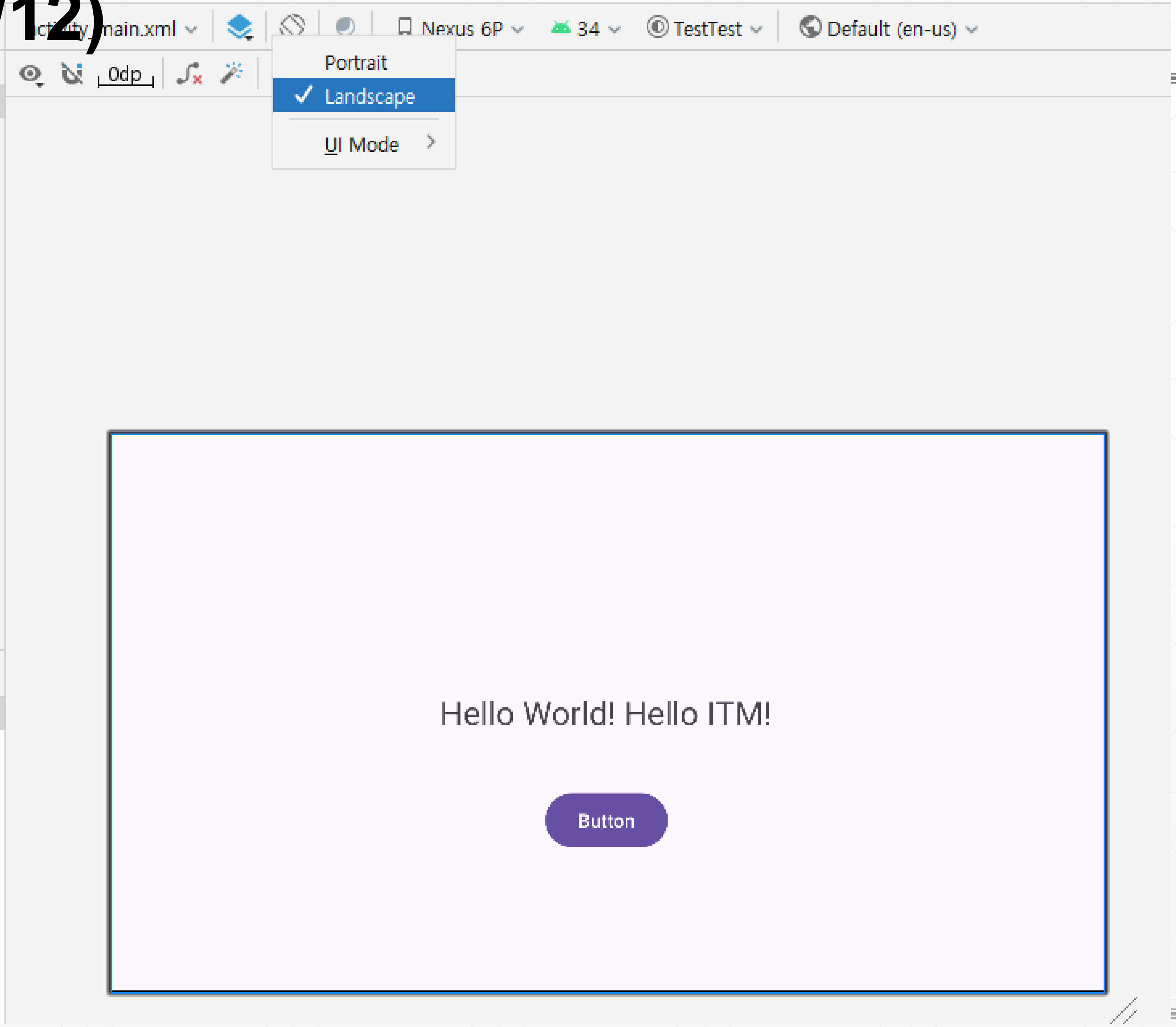
- Create a single button
- Connect top to the bottom of textView
- Connect start to the start of parent
- Connect end to the end of parent

Distance between the point and the connected component (40dp)



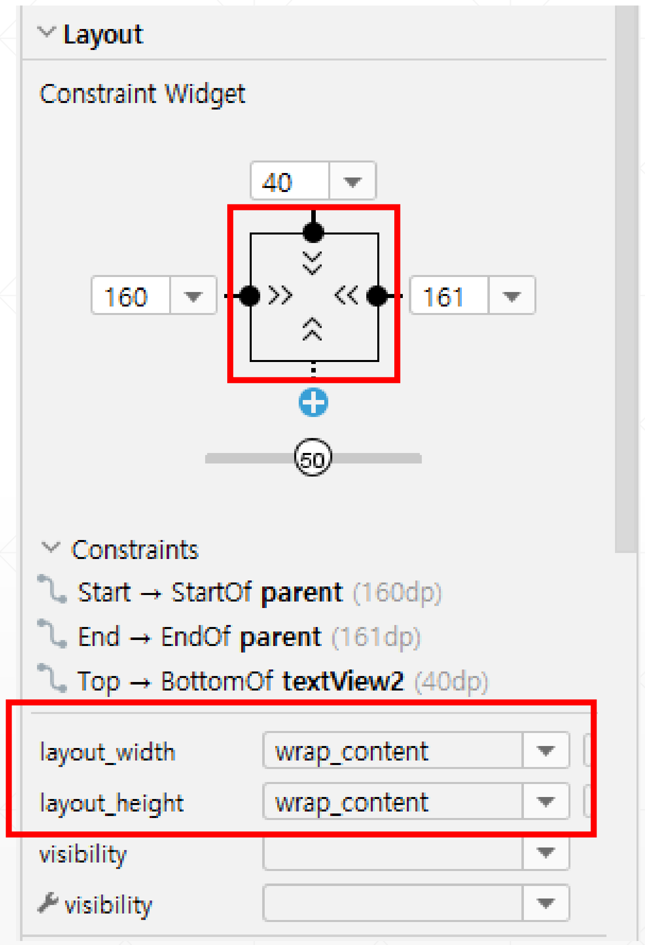
Let’s Edit a Layout! (9/12)

- You can see how they will look in a landscape mode!



Let's Edit a Layout! (10/12)

- Width/height of a widget
  - Fixed (use width & height value)
  - Wrap_content (takes only the space that is needed)
  - Match_constraint (takes all the available space)
    - Margin should be set to 0dp

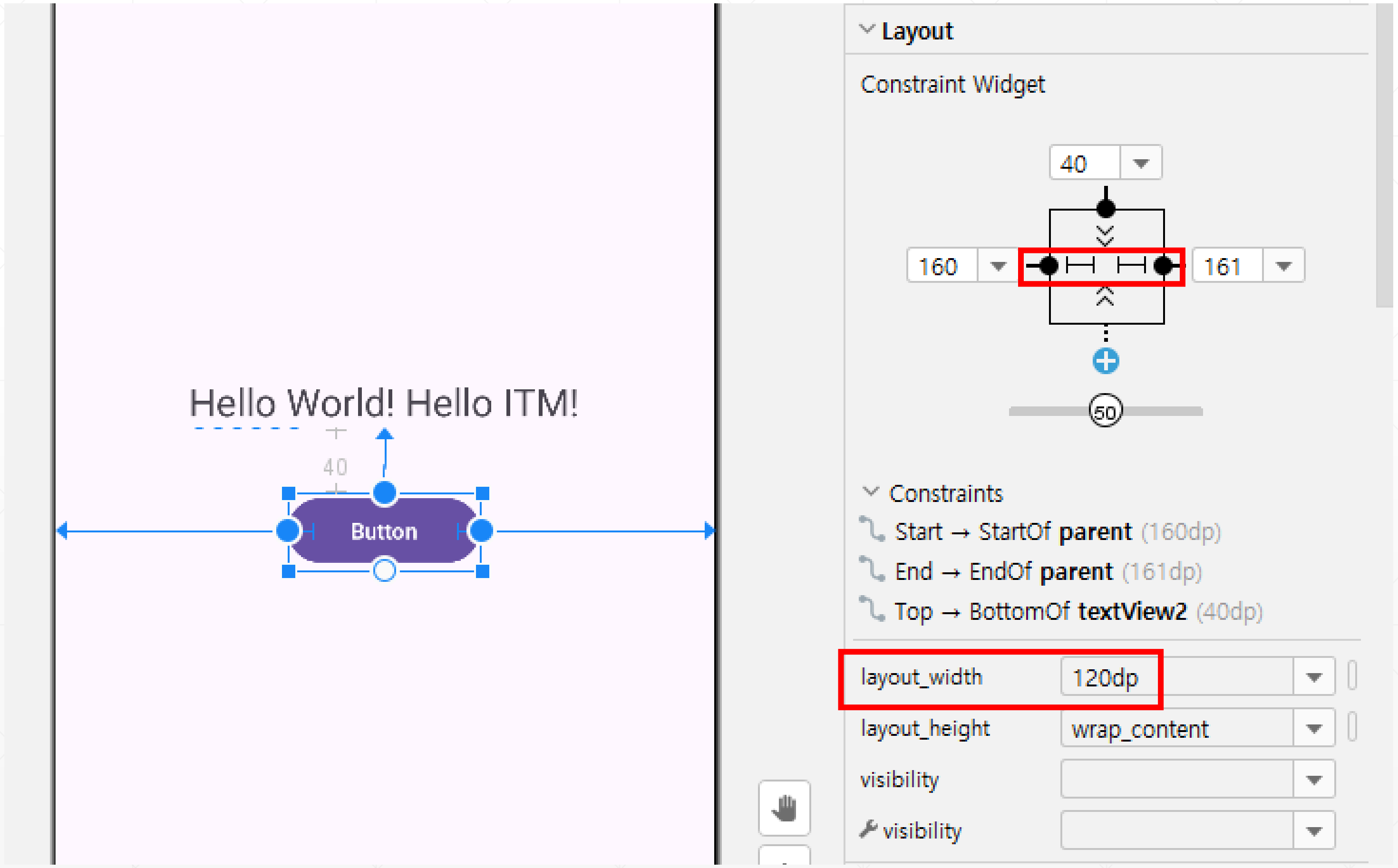


Constraints
Start → StartOf **parent** (160dp)
End → EndOf **parent** (161dp)
Top → BottomOf **textView2** (40dp)

layout_width	wrap_content
layout_height	wrap_content
visibility	
visibility	

Let's Edit a Layout! (11/12)

- Width/height of a widget
  - Fixed (use width & height value)

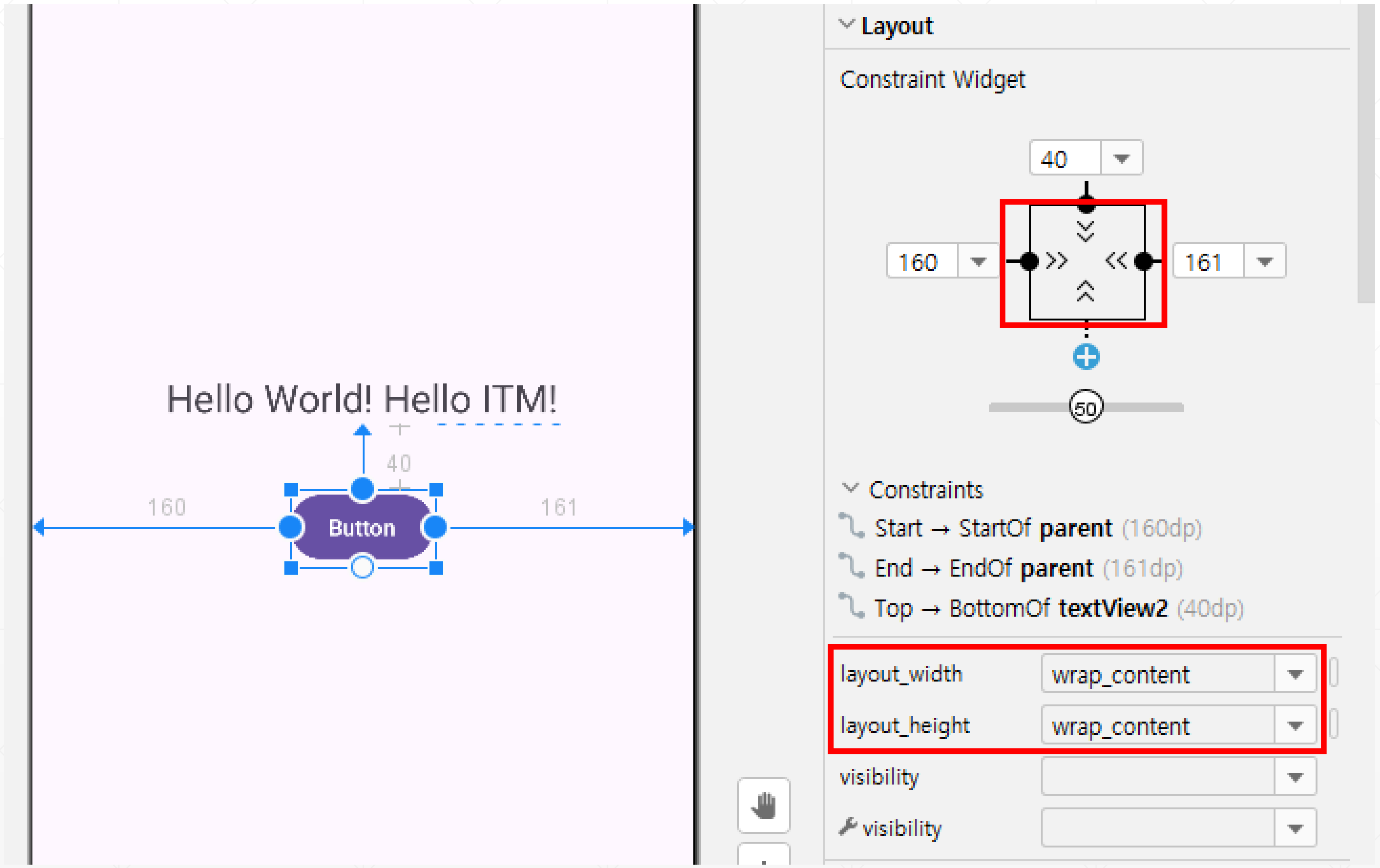


Constraints
- Start → StartOf **parent** (160dp)
- End → EndOf **parent** (161dp)
- Top → BottomOf **textView2** (40dp)

layout_width	120dp
layout_height	wrap_content
visibility	
visibility	

Let's Edit a Layout! (11/12)

- Width/height of a widget
  - Wrap_content (takes only the space that is needed)

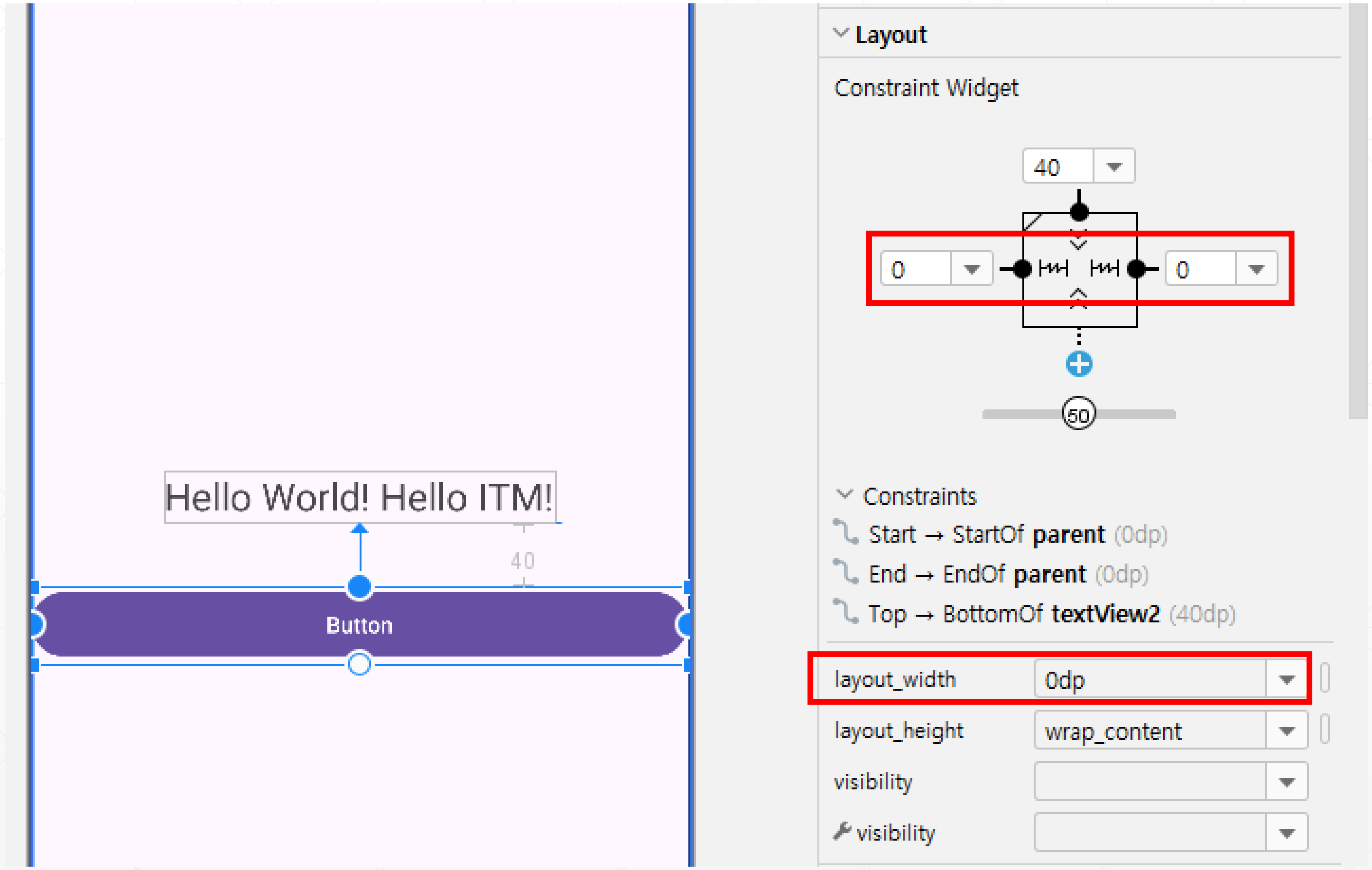


Constraints
Start → StartOf **parent** (160dp)
End → EndOf **parent** (161dp)
Top → BottomOf **textView2** (40dp)

layout_width	wrap_content
layout_height	wrap_content
visibility	
visibility	

Let’s Edit a Layout! (11/12)

- Width/height of a widget
  - Match_constraint (takes all the available space)

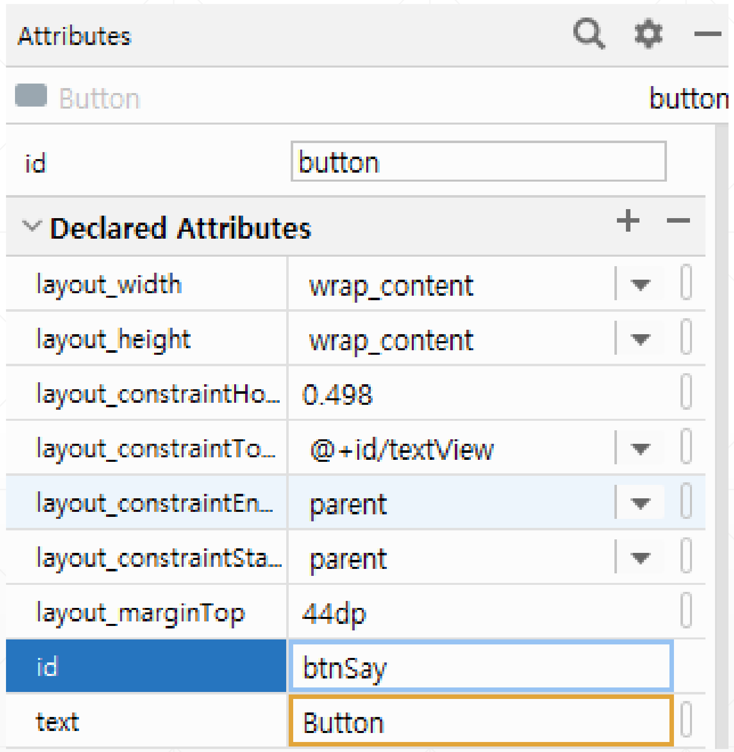


Constraints
Start → StartOf **parent** (0dp)
End → EndOf **parent** (0dp)
Top → BottomOf **textView2** (40dp)

layout_width	0dp
layout_height	wrap_content
visibility	
visibility	

Let’s Edit a Layout! (12/12)

- Rename the id and text of Button



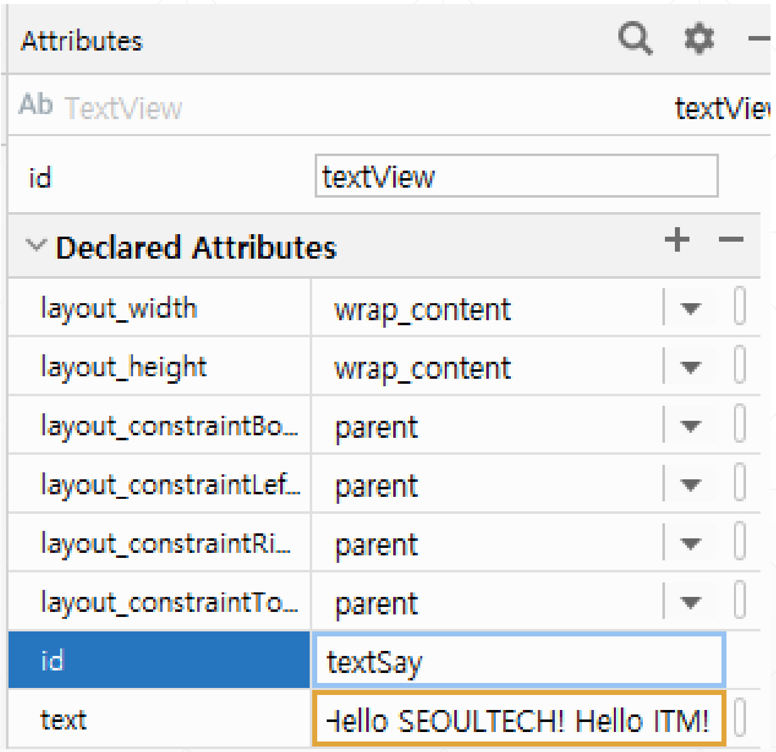
Attributes

Button button

id	button

Declared Attributes

layout_width	wrap_content
layout_height	wrap_content
layout_constraintHo...	0.498
layout_constraintTo...	@+id/textView
layout_constraintEn...	parent
layout_constraintSta...	parent
layout_marginTop	44dp
id	btnSay
text	Button



Attributes

Ab TextView textVie

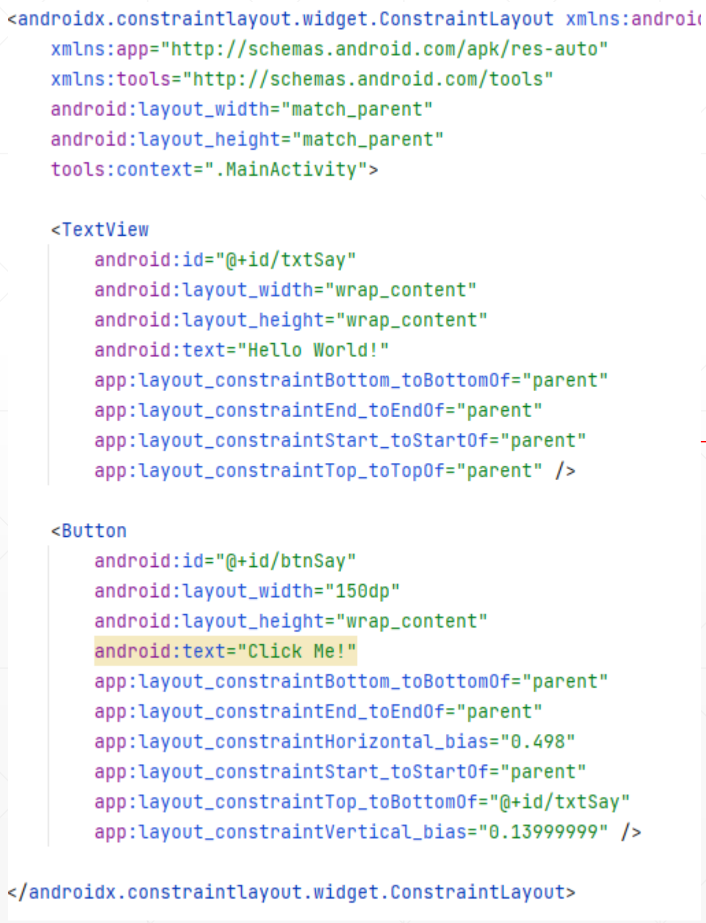
id	textView

Declared Attributes

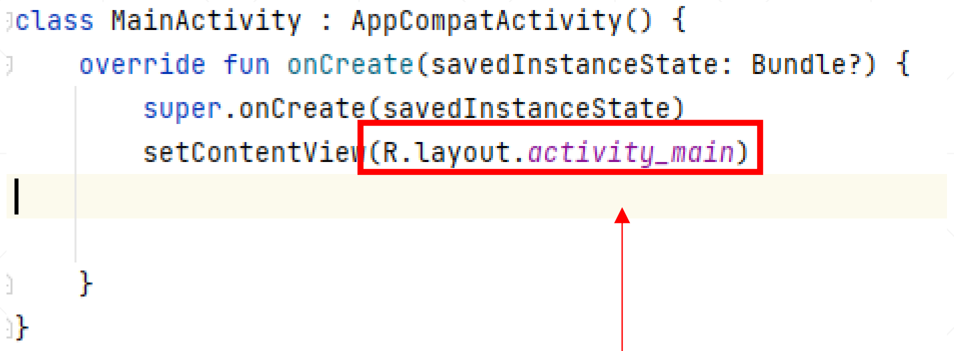
layout_width	wrap_content
layout_height	wrap_content
layout_constraintBo...	parent
layout_constraintLef...	parent
layout_constraintRi...	parent
layout_constraintTo...	parent
id	textSay
text	Hello SEOULTECH! Hello ITM!

UI Control with Code (1/5)

- setContentView()
  - Use an XML file to make a UI layout



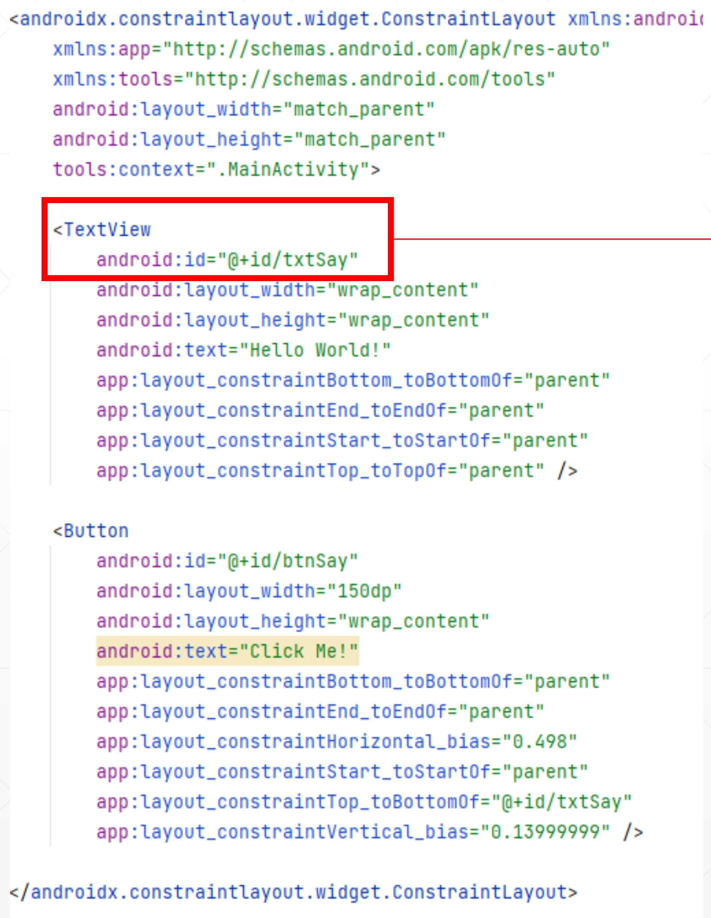
```xml
<androidx.constraintlayout.widget.ConstraintLayout xmlns:androi
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/txtSay"
        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" />

    <Button
        android:id="@+id/btnSay"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:text="Click Me!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtSay"
        app:layout_constraintVertical_bias="0.13999999" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



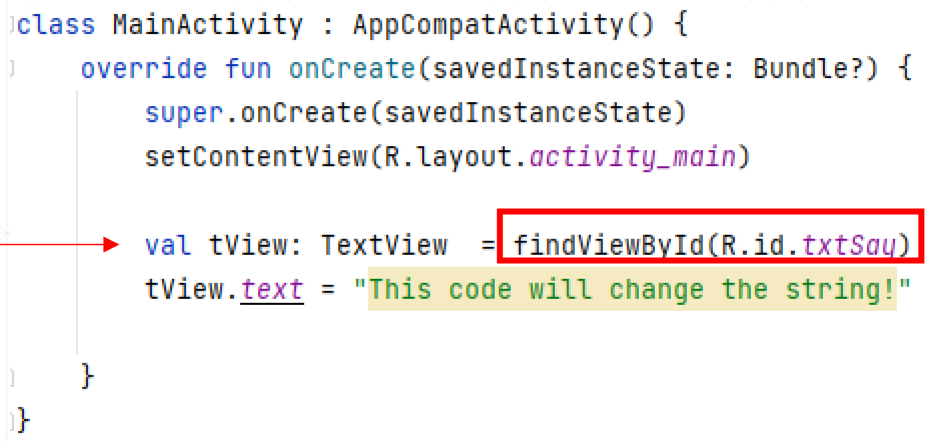
```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

    }
}
```

UI Control with Code (2/5)

- findViewById()
  - Get a View object for further manipulation



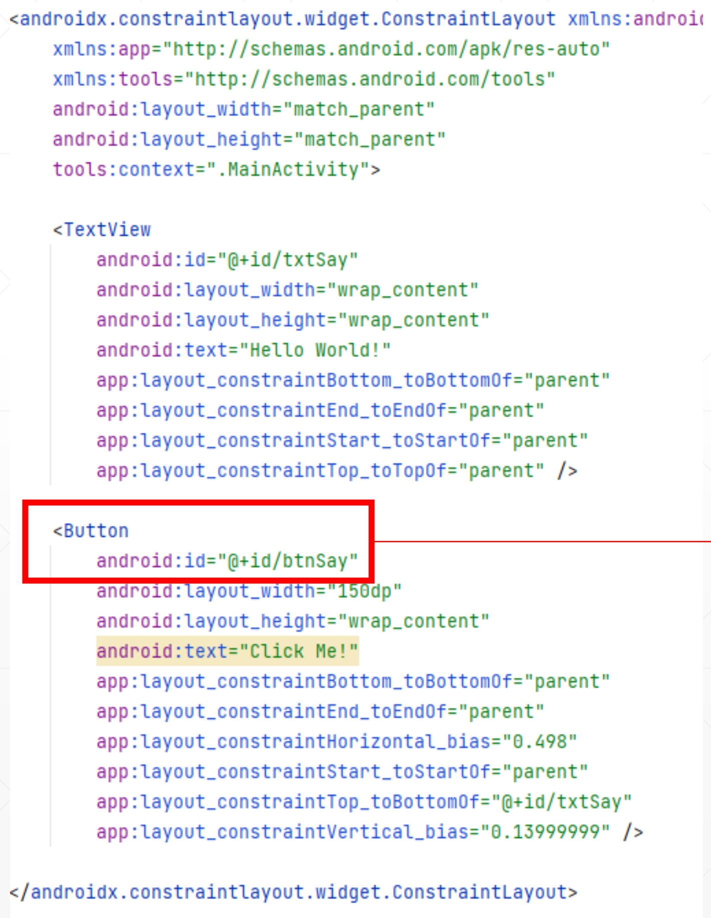
```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:androi
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/txtSay"
        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" />

    <Button
        android:id="@+id/btnSay"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:text="Click Me!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtSay"
        app:layout_constraintVertical_bias="0.13999999" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



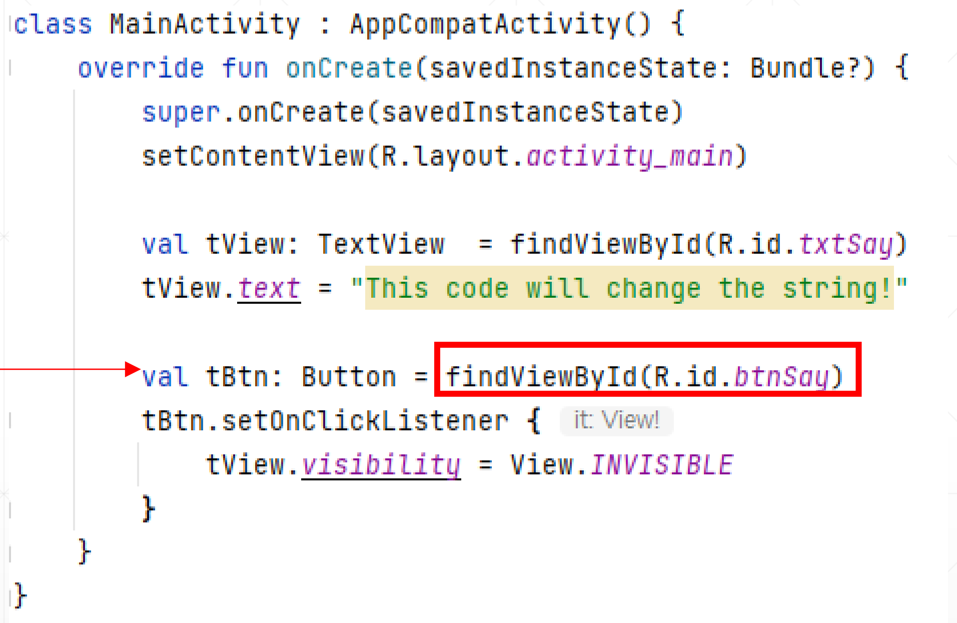
```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val tView: TextView  = findViewById(R.id.txtSay)
        tView.text = "This code will change the string!"

    }
}
```

UI Control with Code (3/5)

- findViewById()
  - Get a View object for further manipulation



```xml
<androidx.constraintlayout.widget.ConstraintLayout xmlns:androi
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/txtSay"
        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" />

    <Button
        android:id="@+id/btnSay"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:text="Click Me!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtSay"
        app:layout_constraintVertical_bias="0.13999999" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

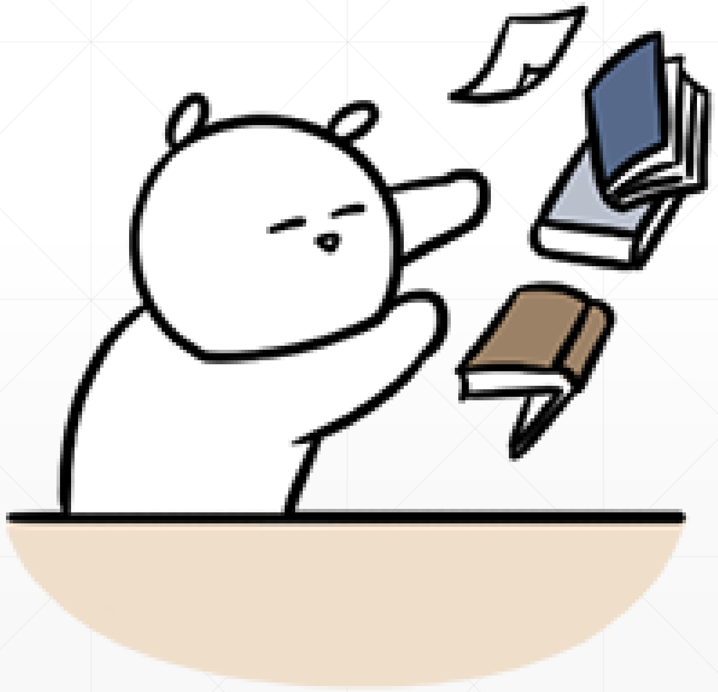


```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val tView: TextView  = findViewById(R.id.txtSay)
        tView.text = "This code will change the string!"

        val tBtn: Button = findViewById(R.id.btnSay)
        tBtn.setOnClickListener {  it: View!
            tView.visibility = View.INVISIBLE
        }
    }
}
```

How to toggle?

UI Control with Code (4/5)

- If you want to control your UI(view) in your code?
  - Then, you need to connect View and source codes
  - But, we don’t want a massive use of findViewById() call …



UI Control with Code (5/5)

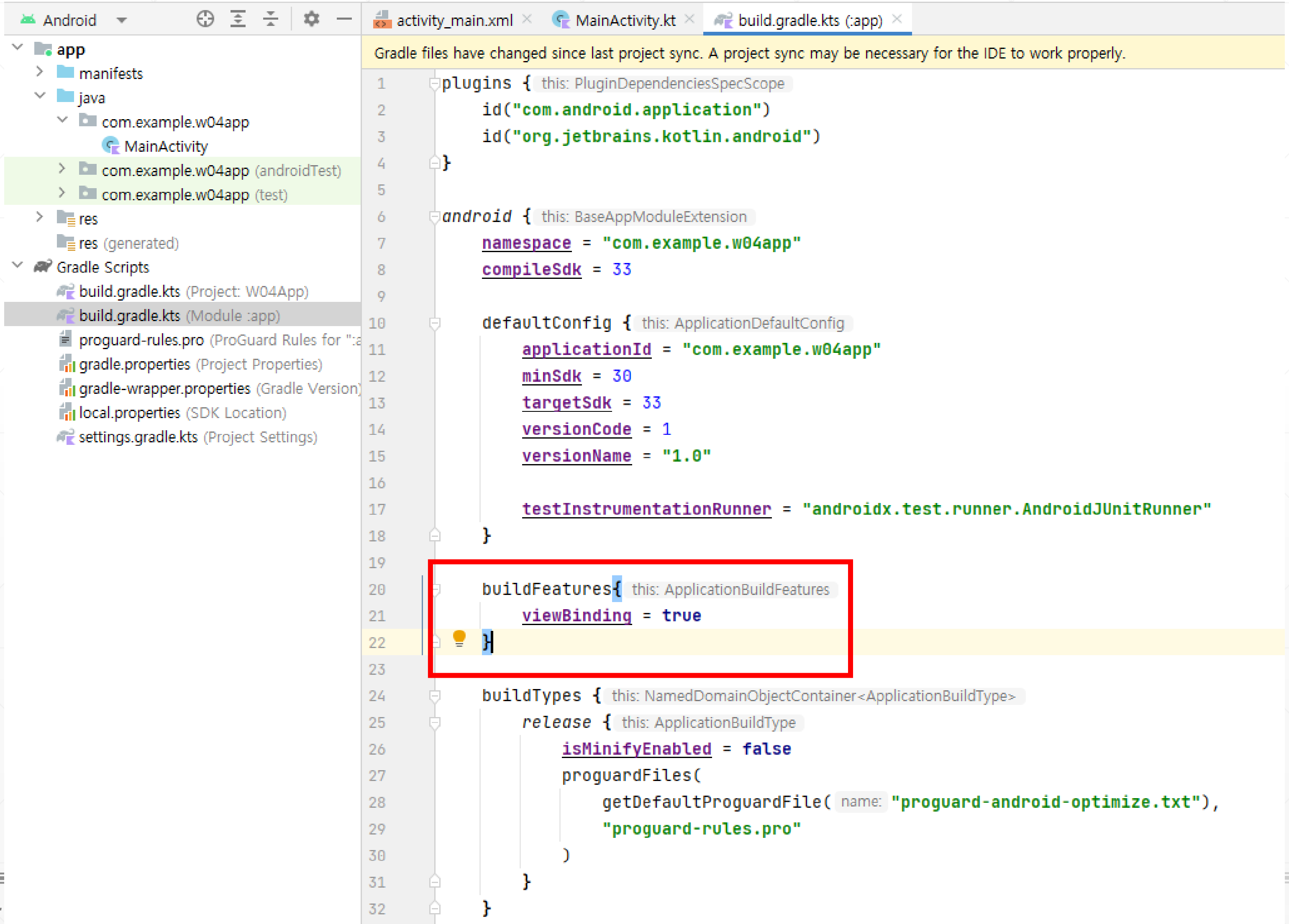
ViewBinding (1/7)

- If you want to control your UI(view) in your code?
  - Then, you need to connect View and source codes
  - But, we don't want a massive use of findViewById() call …

- How to setup viewbinding?
  - Set viewBinding true in build.gradle file
  - Click "Sync Now" for applying the update
  - Android will generate binding from the layout file
  - Initialize your binding and assign to the bindingVariable
  - Pass bindingVaiable.root to setContentView() method
  - Use bindingVariable.id to reference your view!

ViewBinding (2/7)

- Set viewBinding true in build.gradle file

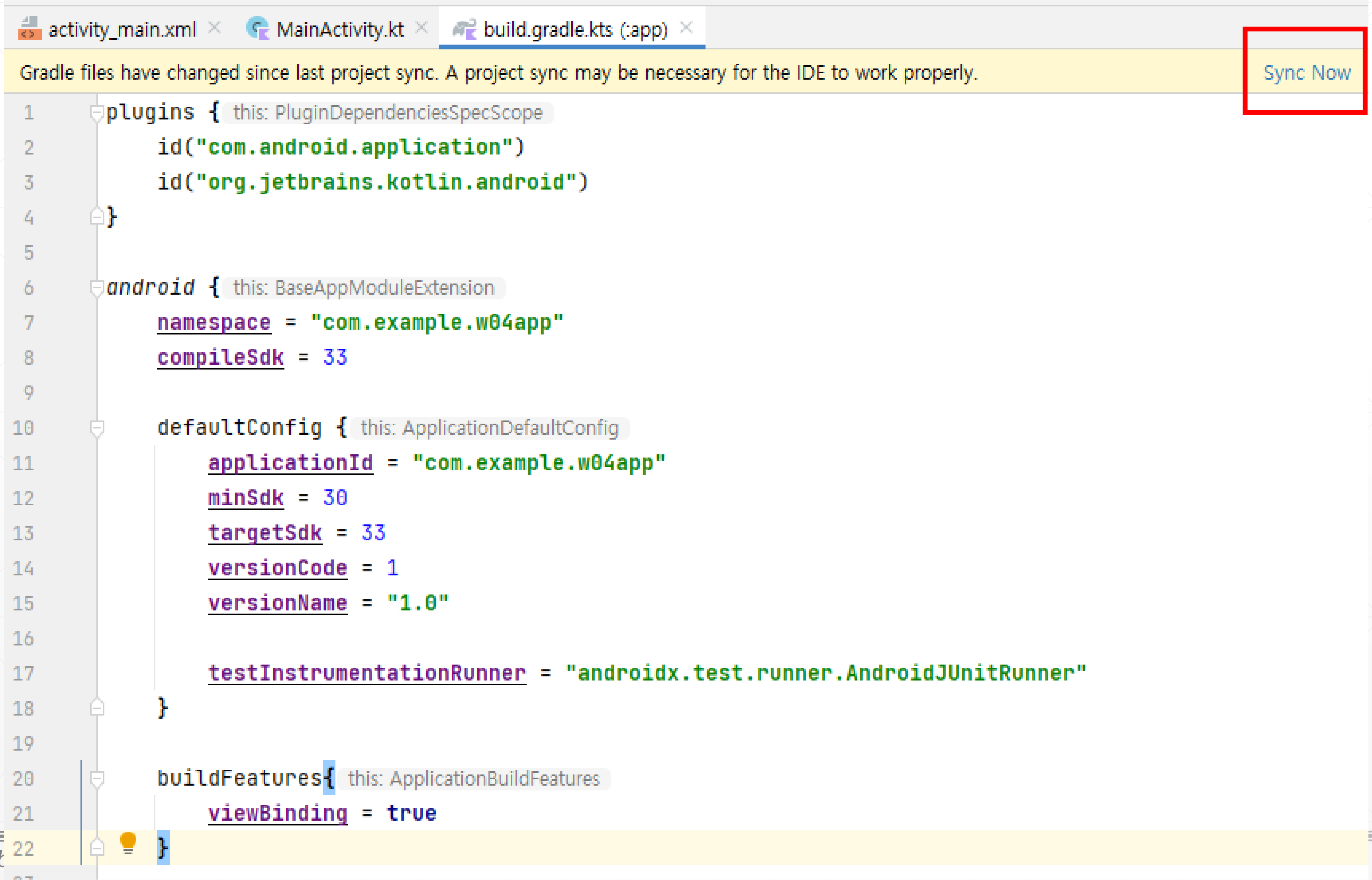

Android
app
manifests
java
com.example.w04app
MainActivity
com.example.w04app (androidTest)
com.example.w04app (test)
res
res (generated)
Gradle Scripts
build.gradle.kts (Project: W04App)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":a
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
activity_main.xml
MainActivity.kt
build.gradle.kts (:app)

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
plugins {
    id("com.android.application")
    id("org.jetbrains.kotlin.android")
}

android {
    namespace = "com.example.w04app"
    compileSdk = 33

    defaultConfig {
        applicationId = "com.example.w04app"
        minSdk = 30
        targetSdk = 33
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildFeatures{
        viewBinding = true
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
```

ViewBinding (3/7)

- Click "Sync Now" for applying the update



Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
plugins {  this: PluginDependenciesSpecScope
    id("com.android.application")
    id("org.jetbrains.kotlin.android")
}

android {  this: BaseAppModuleExtension
    namespace = "com.example.w04app"
    compileSdk = 33

    defaultConfig {  this: ApplicationDefaultConfig
        applicationId = "com.example.w04app"
        minSdk = 30
        targetSdk = 33
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildFeatures{  this: ApplicationBuildFeatures
        viewBinding = true
    }
```

ViewBinding (4/7)

- Android will generate binding from the layout file
  - If view binding is enabled for a module, a binding class is generated for each XML layout file that the module contains
  - The name of the binding class is generated by converting the name of the XML file to Pascal case and adding the word "Binding" to the end
    - activity_main.xml → ActivityMainBinding
    - result_profile.xml → ResultProfileBinding

ViewBinding (5/7)

- Initialize your binding and assign to the bindingVariable

1. Call the static inflate() method included in the generated binding class
   - This creates an instance of the binding class for the activity to use

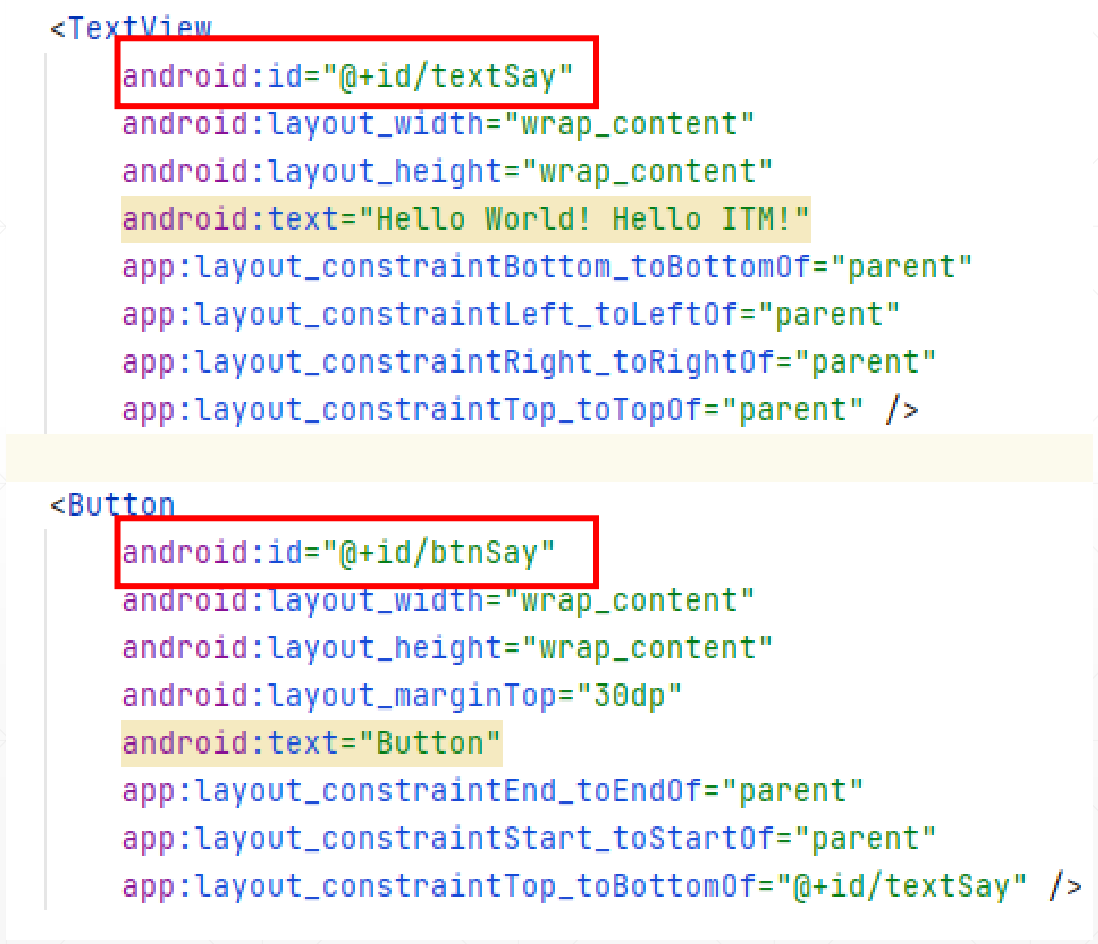
2. Get a reference to the root view by either calling the getRoot() method or using Kotlin property syntax

3. Pass the root view to setContentView() to make it the active view on the screen

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        val binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)


    }
}
```

ViewBinding (6/7)

- Use bindingVariable.id to reference your view!

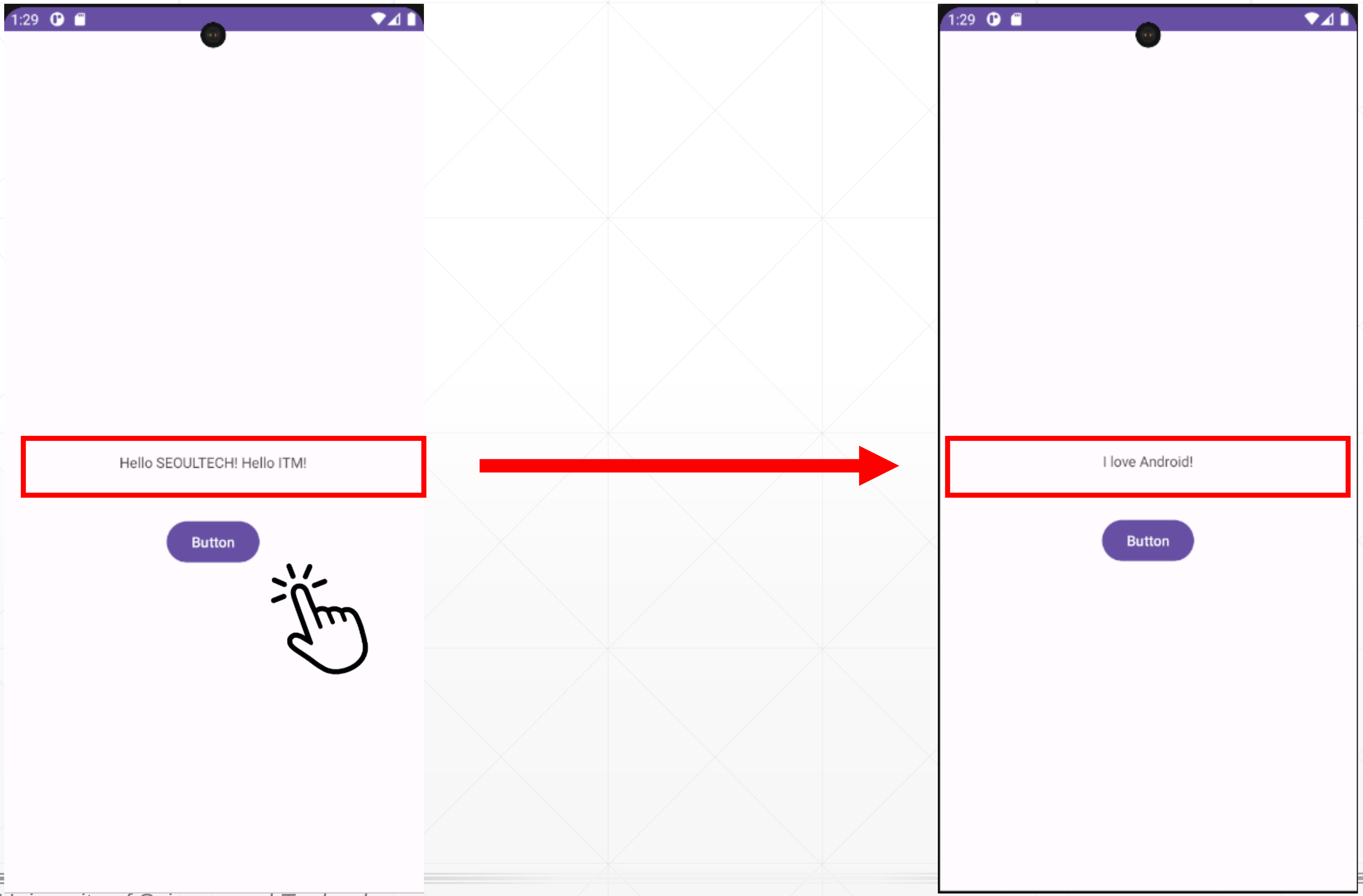


```
<TextView
    android:id="@+id/textSay"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World! Hello ITM!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/btnSay"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="30dp"
    android:text="Button"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textSay" />
```

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        val binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.btnSay.setOnClickListener{
            binding.textSay.text="I love Android!"
        }

    }
}
```

ViewBinding (7/7)

- Let’s see what happens!



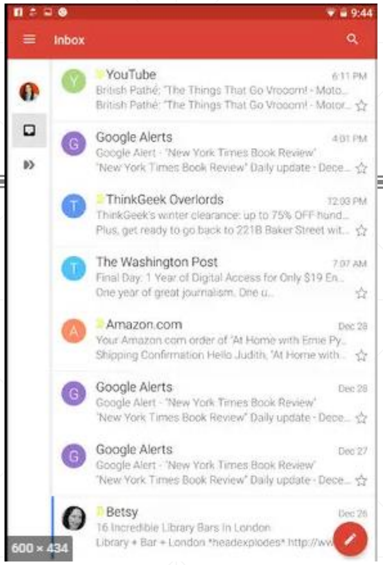
Hello SEOULTECH! Hello ITM!

I love Android!

Android App Basics

Activity Basics

Activity



YouTube 6:11 PM
British Pathé: 'The Things That Go Vrooom! - Moto...
British Pathé: 'The Things That Go Vrooom! - Motor...

Google Alerts 4:01 PM
Google Alert - 'New York Times Book Review'
'New York Times Book Review' Daily update - Dece...

ThinkGeek Overlords 12:03 PM
ThinkGeek's winter clearance: up to 75% OFF hund...
Plus, get ready to go back to 221B Baker Street wit...

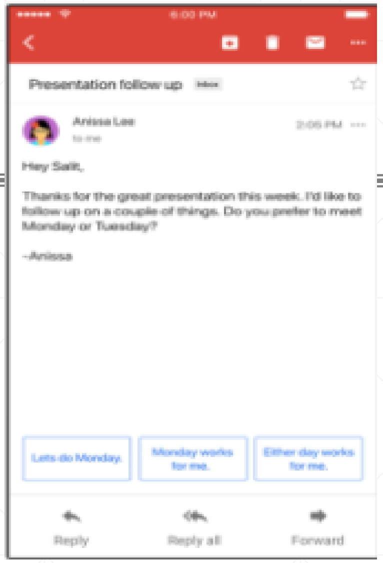
The Washington Post 7:07 AM
Final Day: 1 Year of Digital Access for Only $19 En...
One year of great journalism. One u...

Amazon.com Dec 28
Your Amazon.com order of 'At Home with Ernie Py...
Shipping Confirmation Hello Judith, 'At Home with...

Google Alerts Dec 28
Google Alert - 'New York Times Book Review'
'New York Times Book Review' Daily update - Dece...

Google Alerts Dec 27
Google Alert - 'New York Times Book Review'
'New York Times Book Review' Daily update - Dece...

Betsy Dec 26
16 Incredible Library Bars In London
Library + Bar + London *headexplodes* http://ww...



Presentation follow up

Anissa Lee 2:06 PM
to me

Hey Salit,

Thanks for the great presentation this week. I'd like to follow up on a couple of things. Do you prefer to meet Monday or Tuesday?

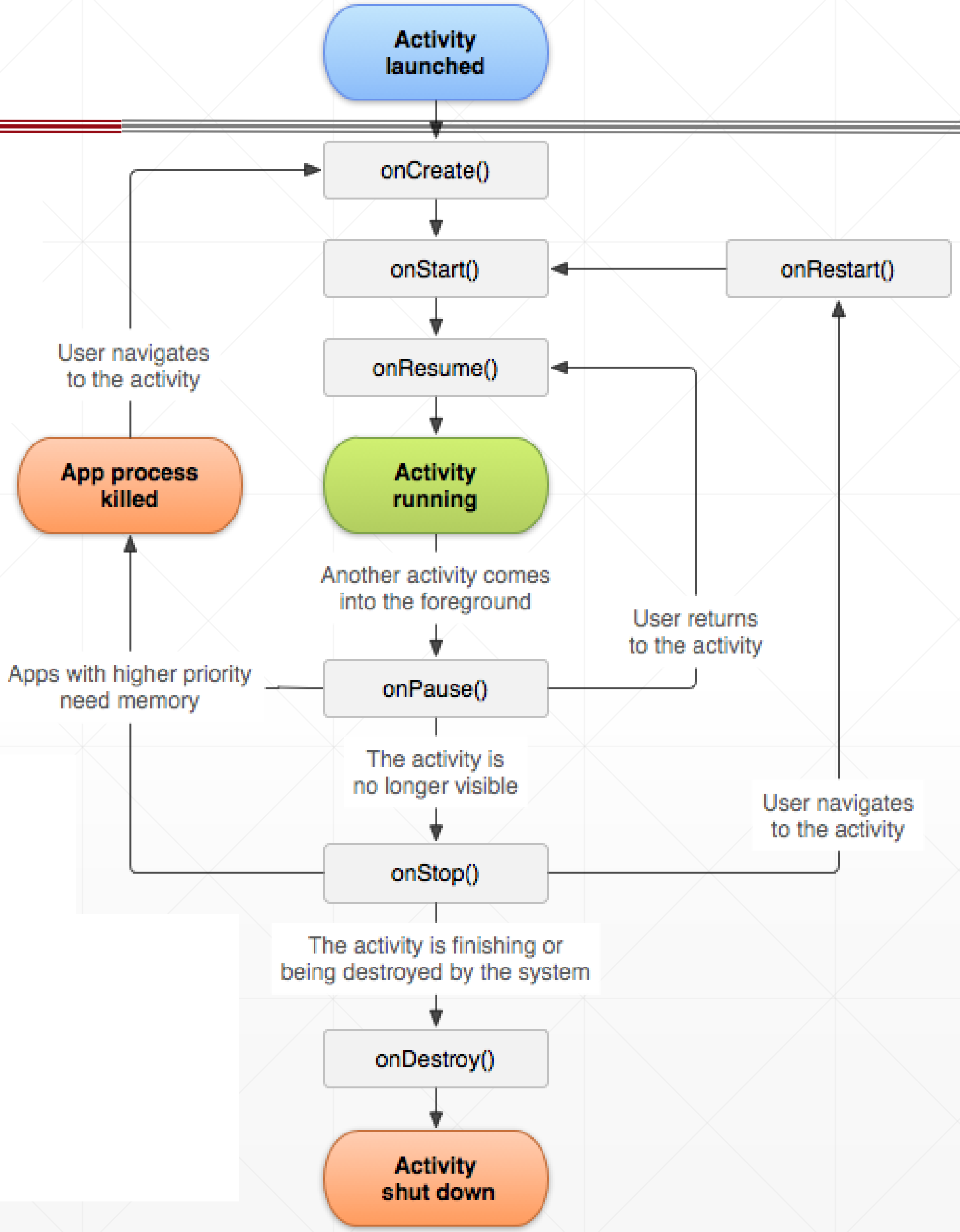
--Anissa

- A crucial component of an Android app
  - Serves as the entry point for an app's interaction with the use
  - Provides the window in which the app draws its UI

- Generally, one activity implements one screen in an app
  - Typically, one activity in an app is specified as the *main activity*, which is the first screen to appear when the user launches the app
  - Each activity can then start another activity in order to perform different actions

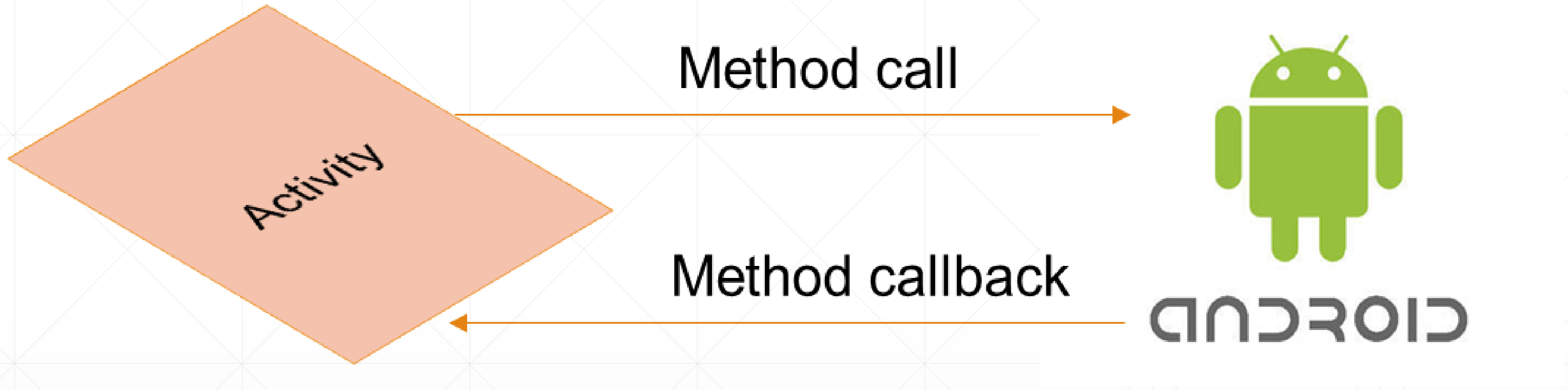
Activity: Lifecycle (1/10)

- Lifecycle
  - As a user navigates through, out of, and bacl the Activity instances in your app transition th **different states** in their lifecycle
- Activity class provides a number of callbacks that allow the activity to know that a state has changed
  - The system is creating, stopping, or resuming
  - The system is destroying the process in which


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Activity: Lifecycle (2/10)

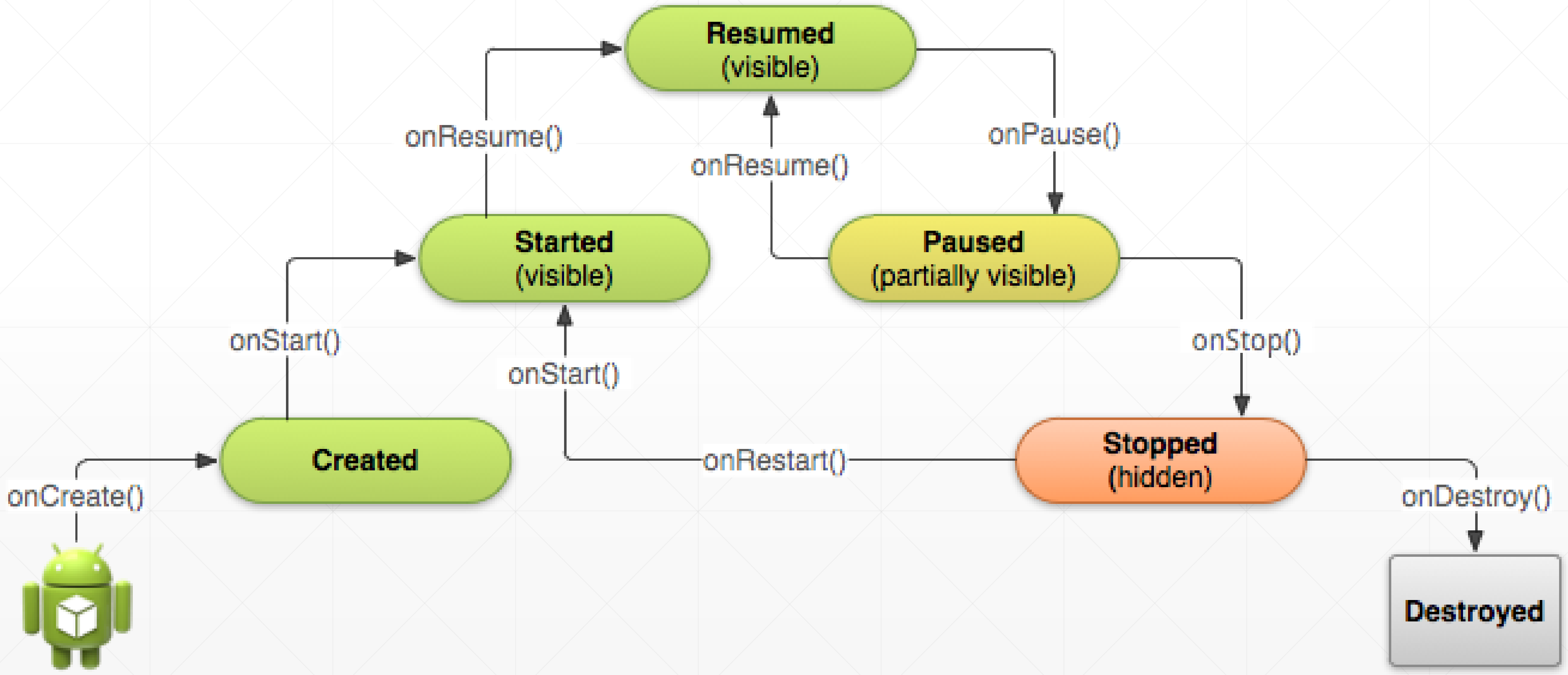
- Lifecycle callbacks



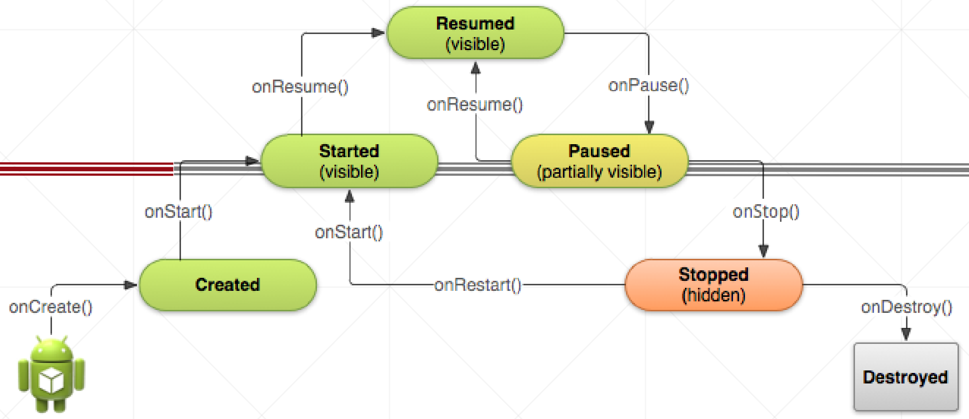
- Method call
  - User-program calls the methods provided by the system
- Method callback
  - System calls the methods provided by the user-program

Activity: Lifecycle (3/10)

- Lifecycle callbacks between state transition



Activity: Lifecycle (4/10)

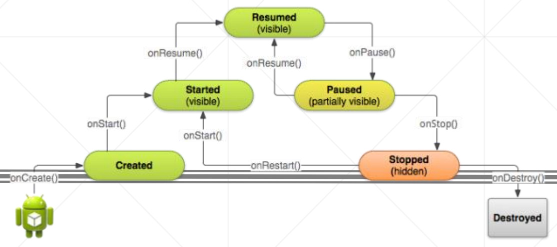


- onCreate()
  - Fires when the system first creates the activity
    - On activity creation, the activity enters the *Created* state
  - You need to perform a **basic application startup logic** that should happen only once for the entire life of the activity
    - Most importantly, this is where you must call setContentView() to define the layout for the activity's user interface

```kotlin
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    val binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)
```

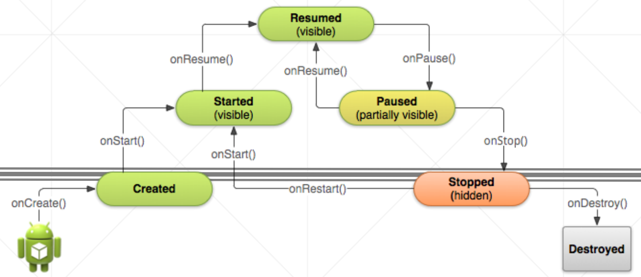
  - After the onCreate() method finishes execution, the activity enters the *Started* state, and the system calls the onStart() and onResume() methods in quick succession

Activity: Lifecycle (5/10)



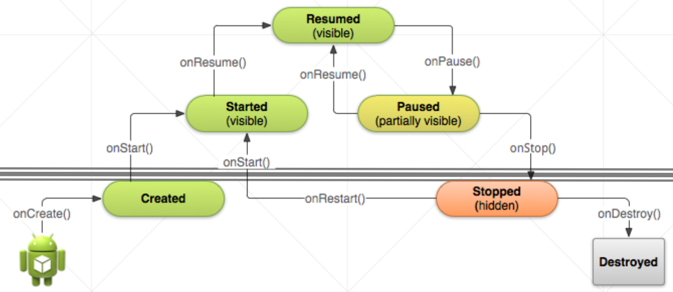
- onStart()
  - When the activity enters the *Started* state, the system invokes this callback
  - Makes the activity visible to the user, as the app prepares for the activity to enter the foreground and become interactive
  - Once this callback finishes, the activity enters the *Resumed* state, and the system invokes the onResume() method

Activity: Lifecycle (6/10)



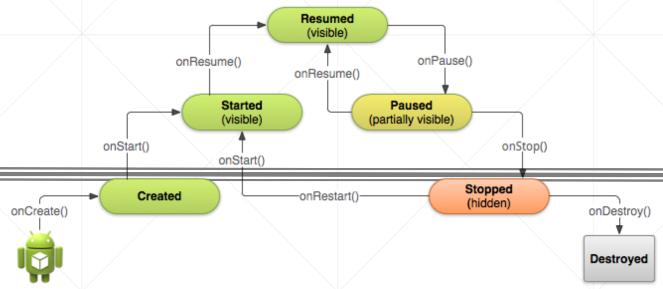
- onResume()
  - When the activity enters the *Resumed* state, it comes to the foreground, and then the system invokes the onResume() callback
  - The state in which the app interacts with the user
    - At this point, the activity is at the top of the activity stack, and captures all user input
    - The app stays in this state until something happens to take focus away from the app
  - When an interruptive event occurs, the activity enters the *Paused* state, and the system invokes the onPause() callback
  - If the activity returns to the *Resumed* state from the *Paused* state, the system once again calls onResume() method!

Activity: Lifecycle (7/10)



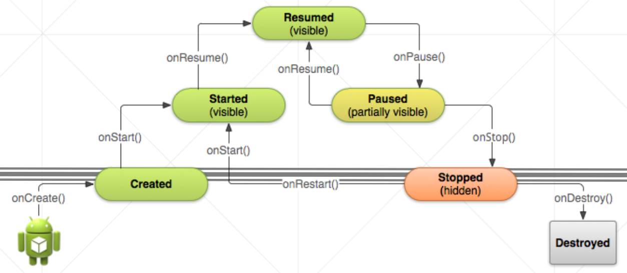
- onPause()
  - The first indication that the user is leaving your activity, indicating that the activity is no longer in the foreground
    - You can use the onPause() method to release system resources, handles to sensors (like GPS), or any resources that may affect battery life while your activity is paused
  - onPause() execution is very brief and does not necessarily afford enough time to perform save operations!
    - To save application or user data, make network calls, or execute database transactions is not recommended in this callback
    - Instead, perform heavy-load shutdown operations during onStop()

Activity: Lifecycle (8/10)



- onStop()
  - When your activity is no longer visible to the user, it has entered the *Stopped* state, and the system invokes the onStop() callback
  - The app should release or adjust resources that are not needed while the app is not visible to the user
    - E.g., To perform relatively CPU-intensive shutdown operations
  - If the activity comes back, the system invokes onRestart(). If the Activity is finished running, the system calls onDestroy()

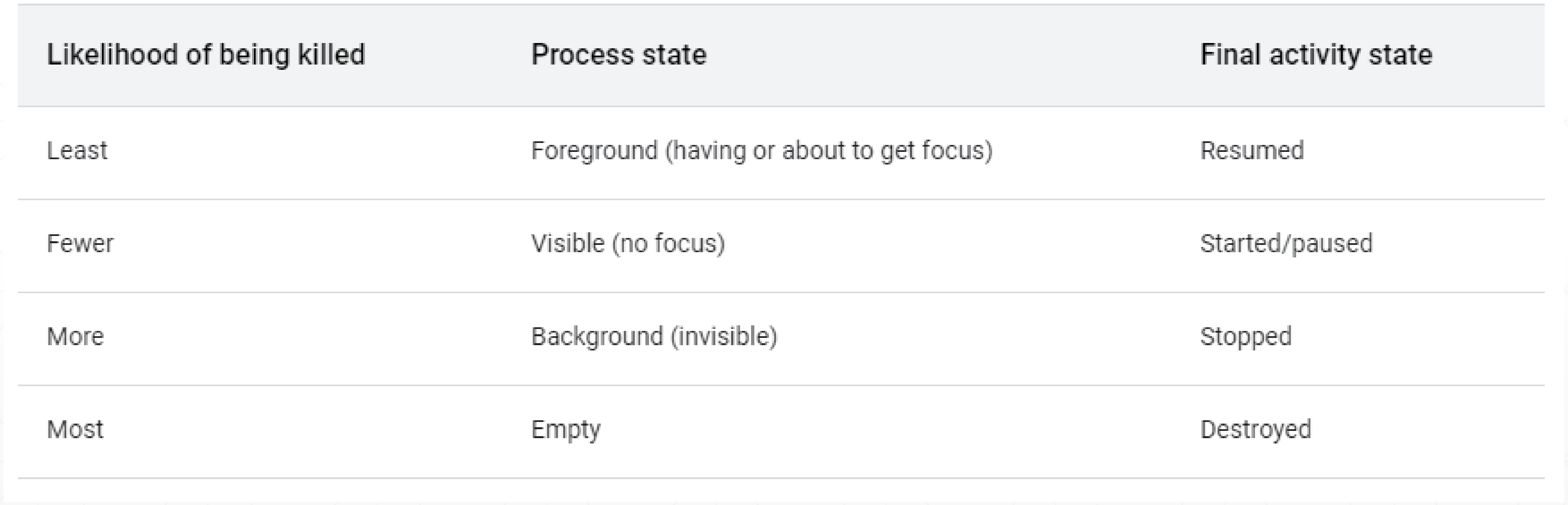
Activity: Lifecycle (9/10)



- onDestroy()
  - Called before the activity is destroyed
  - The system invokes this callback either because:
    - The activity is finishing (due to the user completely dismissing the activity or due to **finish()** being called on the activity)
    - The system is temporarily destroying the activity due to a configuration change (such as device rotation or multi-window mode)
  - Should release all resources that have not yet been released by earlier callbacks such as onStop()

Activity: Lifecycle (10/10)

- The system kills processes when it needs to free up RAM!



Likelihood of being killed	Process state	Final activity state
Least	Foreground (having or about to get focus)	Resumed
Fewer	Visible (no focus)	Started/paused
More	Background (invisible)	Stopped
Most	Empty	Destroyed

Lab: Activity Lifecycle

- Implement each lifecycle callback

```kotlin
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    val binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)
    binding.button.setOnClickListener {
        binding.textView.text = "I love Android!"
    }
    Log.d("ITM","onCreate() called!")
}

override fun onStart() {
    super.onStart()
    Log.d("ITM","onStart() called!")
}

override fun onResume() {
    super.onResume()
    Log.d("ITM","onResume() called!")
}
```

```kotlin
override fun onPause() {
    super.onPause()
    Log.d("ITM","onPause called()!")
}

override fun onStop() {
    super.onStop()
    Log.d("ITM","onStop called()!")
}

override fun onDestroy() {
    super.onDestroy()
    Log.d("ITM", "$isFinishing()")
    Log.d("ITM","onDestroy called()!")
}

override fun onRestart() {
    super.onRestart()
    Log.d("ITM","onRestart called()!")
}
```

Activity: Example of State Changes (1/3)

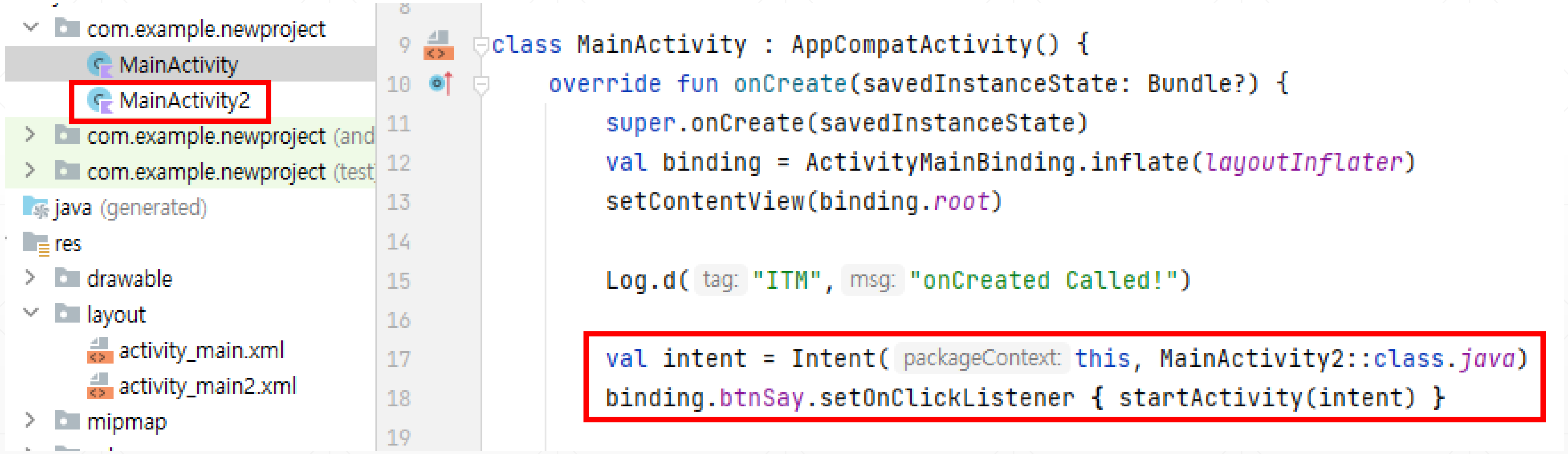
- Configuration changes
  - E.g., change between portrait and landscape orientations
- When a configuration change occurs, the activity is destroyed and recreated
  - The original activity instance will have the following callbacks triggered:
    - onPause()
    - onStop()
    - onDestroy()
  - A new instance of the activity will be created and have the following callbacks triggered:
    - onCreate()
    - onStart()
    - onResume()

Activity: Example of State Changes (2/3)

- New activity or dialog appears in foreground
  - If a new activity or dialog appears in the foreground, taking focus and partially covering the activity in progress, the covered activity loses focus and enters the *Paused* state
  - If a new activity or dialog appears in the foreground, taking focus and completely covering the activity in progress, the covered activity loses focus and enters the *Stopped* state
  - **Note:** When the user taps the Overview or Home button, the system behaves as if the current activity has been completely covered

Activity: Example of State Changes (2/3)

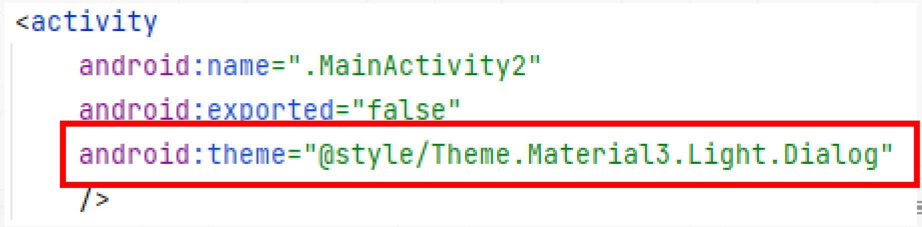
- New activity or dialog appears in foreground
  - Example)
    - 1) Add another activity!
    - 2) Add codelines to start the second activity



```
com.example.newproject
    MainActivity
    MainActivity2
com.example.newproject (and
com.example.newproject (test
java (generated)
res
  drawable
  layout
    activity_main.xml
    activity_main2.xml
  mipmap
```

```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        Log.d( tag: "ITM", msg: "onCreated Called!")

        val intent = Intent( packageContext: this, MainActivity2::class.java)
        binding.btnSay.setOnClickListener { startActivity(intent) }
```

- 3) Add dialog theme for the second activity



```xml
<activity
    android:name=".MainActivity2"
    android:exported="false"
    android:theme="@style/Theme.Material3.Light.Dialog"
    />
```

Activity: Example of State Changes (3/3)

- User presses or gestures Back
  - If an activity is in the foreground, and the user presses or gestures Back, the activity transitions through the onPause(), onStop(), and onDestroy() callbacks!

- Back press behavior for root launcher activities
  - Root launcher activities are activities that declare an Intent filter with both ACTION_MAIN and CATEGORY_LAUNCHER
    - These activities are unique because they act as entry points into your app from the app launcher!
  - On Android 11 and lower: the system finishes the activity
  - On Android 12 and higher: the system moves the activity to the background instead of finishing the activity