

**U**nity  
Thống nhất  
**E**xcellence  
Vượt trội  
**L**eadership  
Tiên phong

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT**



**PHÁT TRIỂN  
THƯƠNG MẠI DI ĐỘNG**

*GV: ThS Nguyễn Quang Phúc*

Phát triển thương mại di động:

# XÂY DỰNG ỨNG DỤNG DI ĐỘNG

ThS. Nguyễn Quang Phúc  
[phucnq@uel.edu.vn](mailto:phucnq@uel.edu.vn)

01

## GIỚI THIỆU VỀ LẬP TRÌNH ỨNG DỤNG DI ĐỘNG

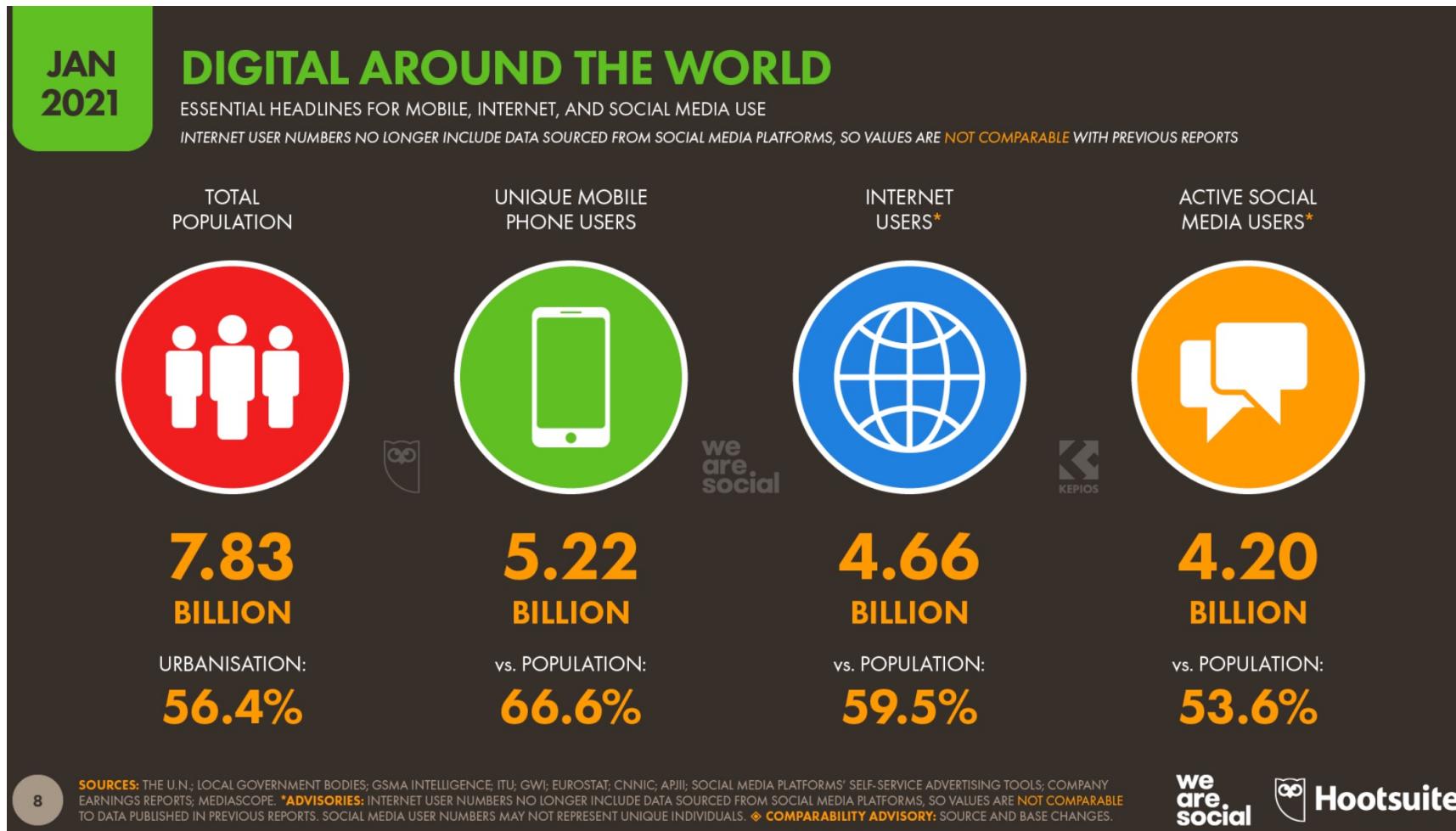
# NỘI DUNG

1. Xu thế phát triển ứng dụng di động
2. Các nền tảng lập trình di động
3. Sơ lược về hệ điều hành Android
4. Thiết lập môi trường lập trình ứng dụng Android
5. Giới thiệu công cụ Android Studio
6. Cách thức thực thi ứng dụng Android

# 1. Xu thế phát triển ứng dụng di động

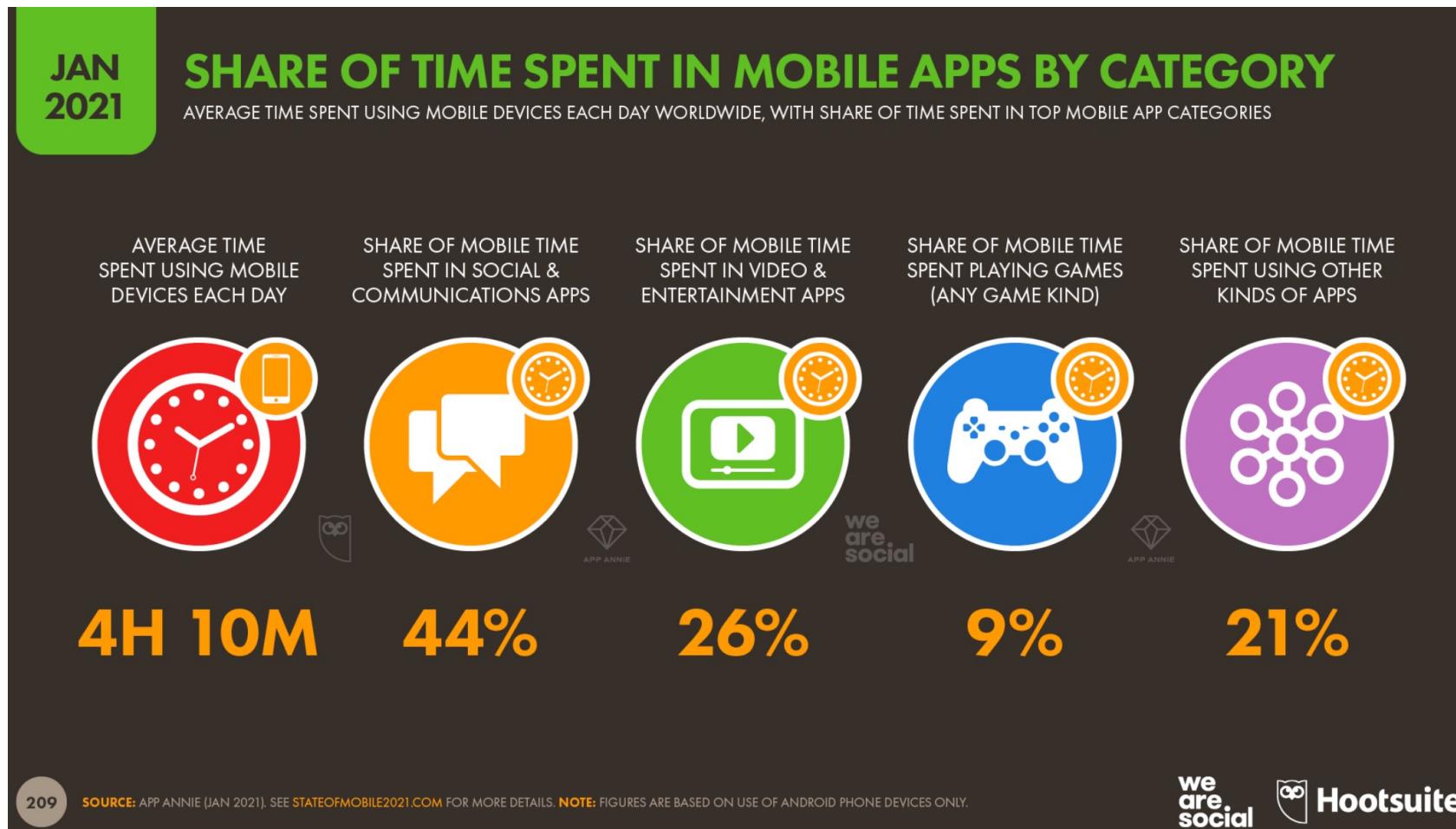
TẠI SAO NÊN HỌC LẬP TRÌNH  
ỨNG DỤNG DI ĐỘNG?

# 1. Xu thế phát triển ứng dụng di động



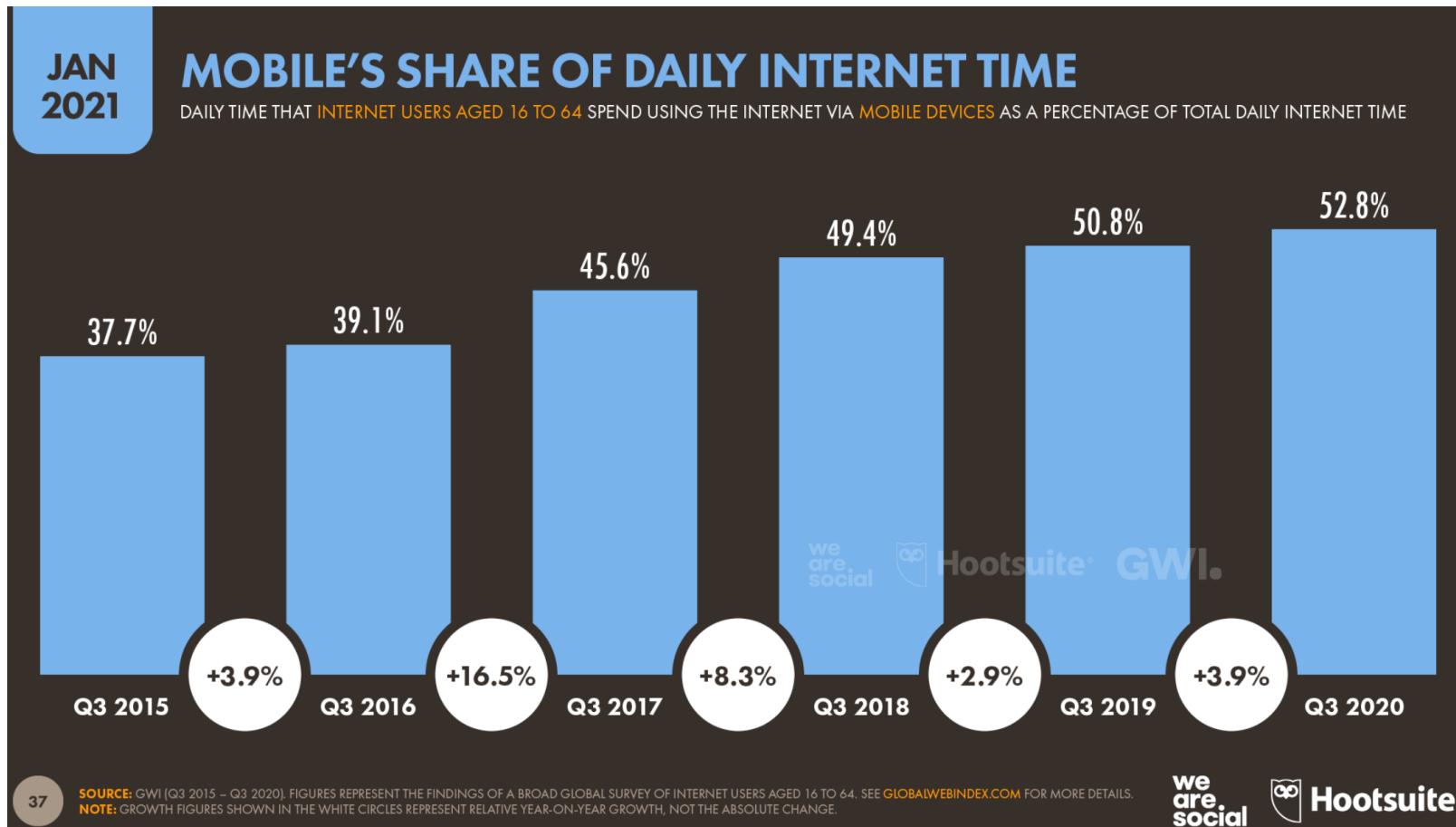
Nguồn: theo báo cáo Digital 2021 ([wearesocial.com](http://wearesocial.com))

# 1. Xu thế phát triển ứng dụng di động



Nguồn: theo báo cáo Digital 2021 ([wearesocial.com](http://wearesocial.com))

# 1. Xu thế phát triển ứng dụng di động



Nguồn: theo báo cáo Digital 2021 ([wearesocial.com](http://wearesocial.com))

→ Truy cập Internet trên thiết bị di động đang trở thành một xu hướng chung trên thế giới.

## 2. Các nền tảng lập trình di động

Objective-C / Swift



iOS

Android



Java / Kotlin

### 3. Sơ lược về hệ điều hành Android

#### »» Lịch sử

- Năm **2003**, Android Inc. được thành lập bởi Andy Rubin, Rich Miner, Nick Sears và Chris White tại California.
- Năm **2005**, Google mua lại dự án Android do thấy tiềm năng trong lĩnh vực di động.
- Năm **2007**, Google giới thiệu Android vào ngày 05/11 (ngày sinh nhật Android).
- Năm **2008**, thiết bị HTC Dream là phiên bản thử nghiệm đầu tiên hoạt động với hệ điều hành Android 1.0
- Năm **2010**, Google khởi đầu dòng thiết bị Nexus với thiết bị đầu tiên của HTC là Nexus One.

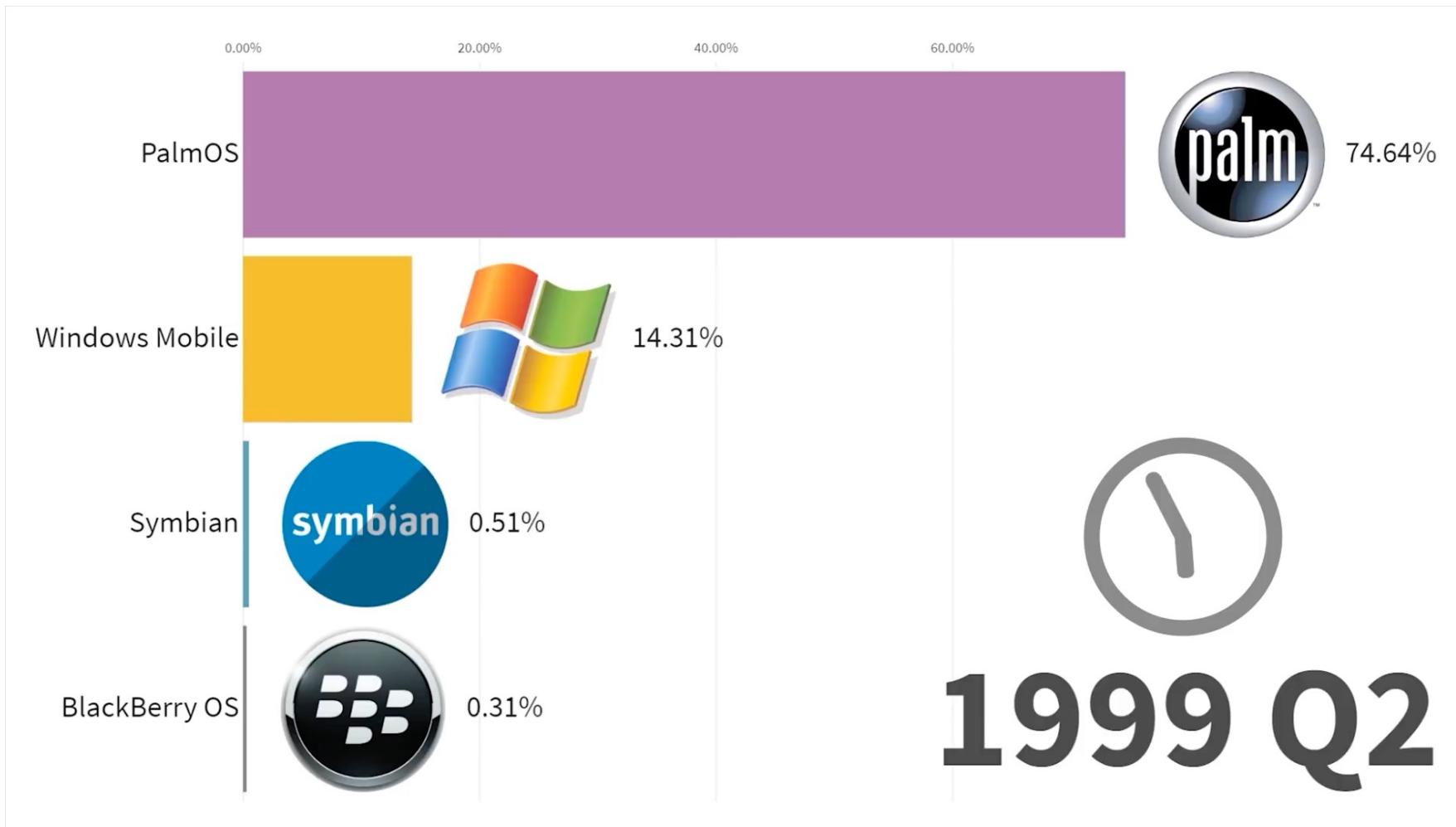
### 3. Sơ lược về hệ điều hành Android

#### ➤ Lịch sử

- Năm **2013**, ra mắt loạt thiết bị phiên bản GPE.
- Năm **2014**, hội nghị Google I/O diễn ra cực kỳ hoành tráng và thú vị với các công bố về Android Wear, Adroid TV hay Google Glass.
- Năm **2016**, một sự kiện quan trọng xảy ra khi một chiếc điện thoại Android hàng đầu - **Samsung Galaxy S7** chính thức đánh bại smartphone cao cấp - **iPhone 7** của Apple.
- Năm **2019**, Android chiếm 85,23% thị phần toàn cầu, theo sau là iOS với 10,63%, 4,13% cho KaiOS và 0,01% cho Windows.

### 3. Sơ lược về hệ điều hành Android

#### »» Lịch sử



### 3. Sơ lược về hệ điều hành Android

#### »» Lịch sử

Version	Name	Release
1.0		September 23, 2008
1.1		February 9, 2009
1.5	<b>Cupcake</b>	April 27, 2009
1.6	<b>Donut</b>	September 15, 2009
<b>2.0 - 2.1</b>	<b>Eclair</b>	October 26, 2009
<b>2.2 - 2.2.3</b>	<b>Froyo</b>	May 20, 2010
<b>2.3 - 2.3.7</b>	<b>Gingerbread</b>	December 6, 2010
<b>3.0 - 3.2.6</b>	<b>Honeycomb</b>	February 22, 2011
<b>4.0 - 4.0.4</b>	<b>Ice Cream Sandwich</b>	October 18, 2011
<b>4.1 - 4.3.1</b>	<b>Jelly Bean</b>	July 9, 2012
<b>4.4 - 4.4.4</b>	<b>KitKat</b>	October 31, 2013

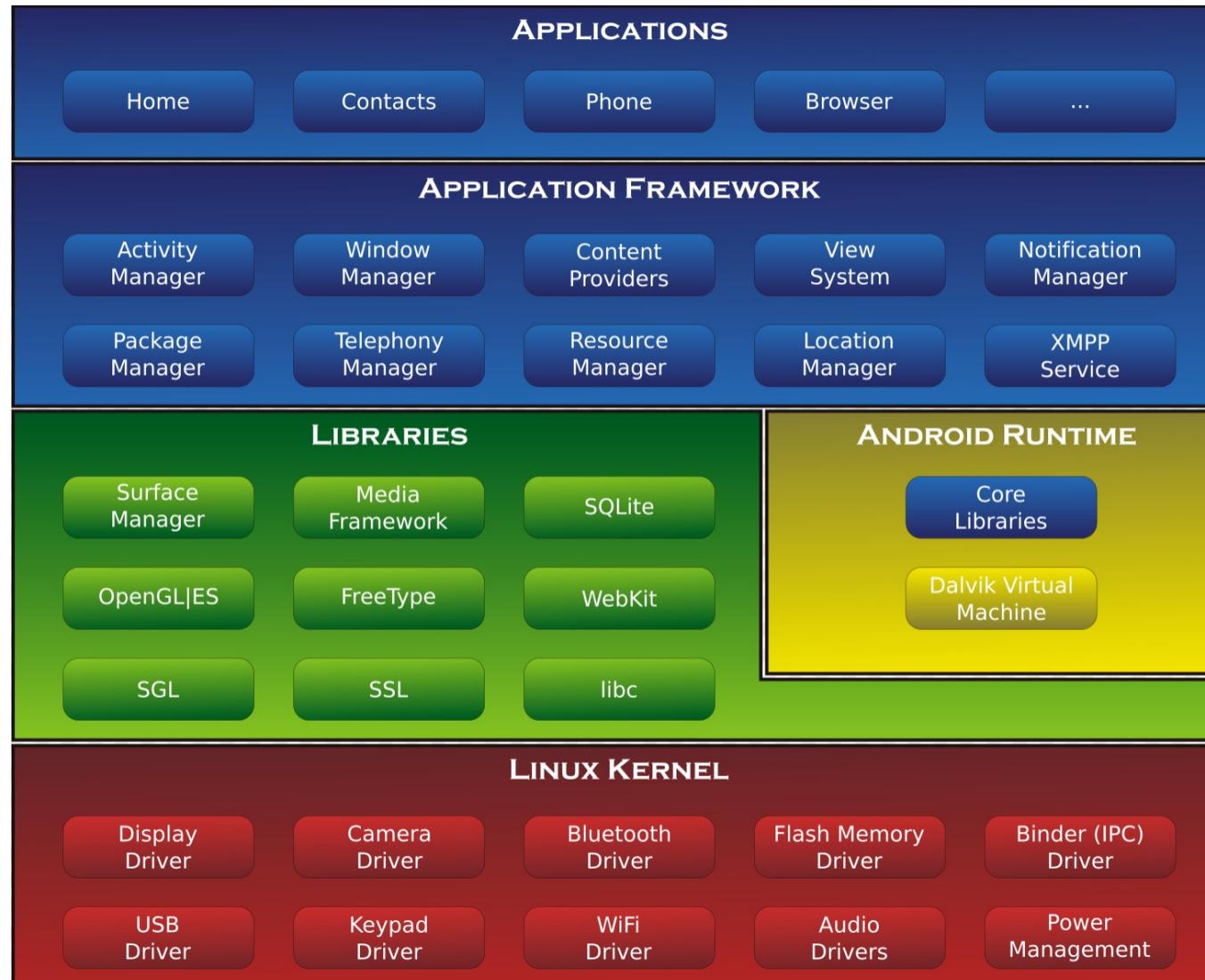
### 3. Sơ lược về hệ điều hành Android

#### »» Lịch sử

Version	Name	Release
5.0 - 5.1.1	Lollipop	November 12, 2014
6.0 - 6.0.1	Marshmallow	October 5, 2015
7.0 - 7.1.2	Nougat	August 22, 2016
8.0 - 8.1	Oreo	August 21, 2017
9.0	Pie	August 6, 2018
10.0	Q	September 3, 2019
11.0	R	September 2020
12.0	S	October 2021
13.0		August 2022

### 3. Sơ lược về hệ điều hành Android

#### »» Kiến trúc



### 3. Sơ lược về hệ điều hành Android

#### »» Kiến trúc

- Linux Kernel



➔ Chưa tất cả driver phần cứng cần thiết như camera, bàn phím, màn hình...

### 3. Sơ lược về hệ điều hành Android

#### »» Kiến trúc

##### ■ Libraries



➔ Thực thi trên tầng Linux Kernel bao gồm các thư viện hỗ trợ:

- ✓ Thư viện hỗ trợ phát các tập tin đa truyền thông,
- ✓ Bộ quản lý hiển thị
- ✓ Thư viện hỗ trợ đồ họa OpenGL 2D và 3D
- ✓ SQLite tiện lợi cho việc lưu trữ và chia sẻ dữ liệu,
- ✓ WebKit và SSL cho phép tương tác với trình duyệt và bảo mật Internet.

### 3. Sơ lược về hệ điều hành Android

#### »» Kiến trúc

- Android Runtime



- ✓ Chứa các thư viện lõi Android cung cấp hầu hết các chức năng chính có thể có trong thư viện Java cũng như thư viện riêng biệt của Android.
- ✓ Chứa Dalvik Virtual Machine (DVM) – một biến thể của Java Virtual Machine, được thiết kế và tối ưu hóa cho Android → giúp mỗi ứng dụng Android chạy trong chính tiến trình (process) của nó với một đại diện (instance) của DVM.

### 3. Sơ lược về hệ điều hành Android

#### »» Kiến trúc

- Application Framework



➔ Cung cấp nhiều dịch vụ cấp cao dưới dạng các lớp viết bằng Java (Java classes). Lập trình viên được phép sử dụng các lớp này để tạo ra các ứng dụng.

### 3. Sơ lược về hệ điều hành Android

#### »» Kiến trúc

- Application



→ Gồm các ứng dụng được tích hợp sẵn và các ứng dụng của hãng thứ ba.

## 4. Thiết lập môi trường lập trình Android

### »» Phản ứng

- Máy tính
  - ✓ **4 GB** RAM minimum, **8 GB** RAM recommended
  - ✓ **2 GB** of available disk space minimum, **4 GB** Recommended  
(**500 MB** for IDE + **1.5 GB** for Android SDK and emulator system image)
  - ✓ **1280 x 800** minimum screen resolution.

## 4. Thiết lập môi trường lập trình Android

### »» Phản ứng

- Điện thoại



# 4. Thiết lập môi trường lập trình Android

## »» Phản mềm

- Hệ điều hành

WINDOWS	MAC	LINUX
Windows® 7/8/10 (32- or 64-bit)  <i>The Android Emulator only supports 64-bit Windows</i>	Mac® OS X® 10.10 (Yosemite) or higher, up to 10.15 (macOS Catalina)	GNOME or KDE desktop

# 4. Thiết lập môi trường lập trình Android

## »» Phản mềm

- JDK – Java Development Kit



Link: <https://www.oracle.com/technetwork/java/javase/downloads/index.html>

## 4. Thiết lập môi trường lập trình Android

### »» Phản mềm

- JDK – Java Development Kit

**Java SE Development Kit 13.0.2**

You must accept the [Oracle Technology Network License Agreement for Oracle Java SE](#) to download this software.

Thank you for accepting the Oracle Technology Network License Agreement for Oracle Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux	155.72 MB	<a href="#"> jdk-13.0.2_linux-x64_bin.deb</a>
Linux	162.66 MB	<a href="#"> jdk-13.0.2_linux-x64_bin.rpm</a>
Linux	179.41 MB	<a href="#"> jdk-13.0.2_linux-x64_bin.tar.gz</a>
macOS	173.3 MB	<a href="#"> jdk-13.0.2_osx-x64_bin.dmg</a>
macOS	173.7 MB	<a href="#"> jdk-13.0.2_osx-x64_bin.tar.gz</a>
Windows	159.83 MB	<a href="#"> jdk-13.0.2_windows-x64_bin.exe</a>
Windows	178.99 MB	<a href="#"> jdk-13.0.2_windows-x64_bin.zip</a>

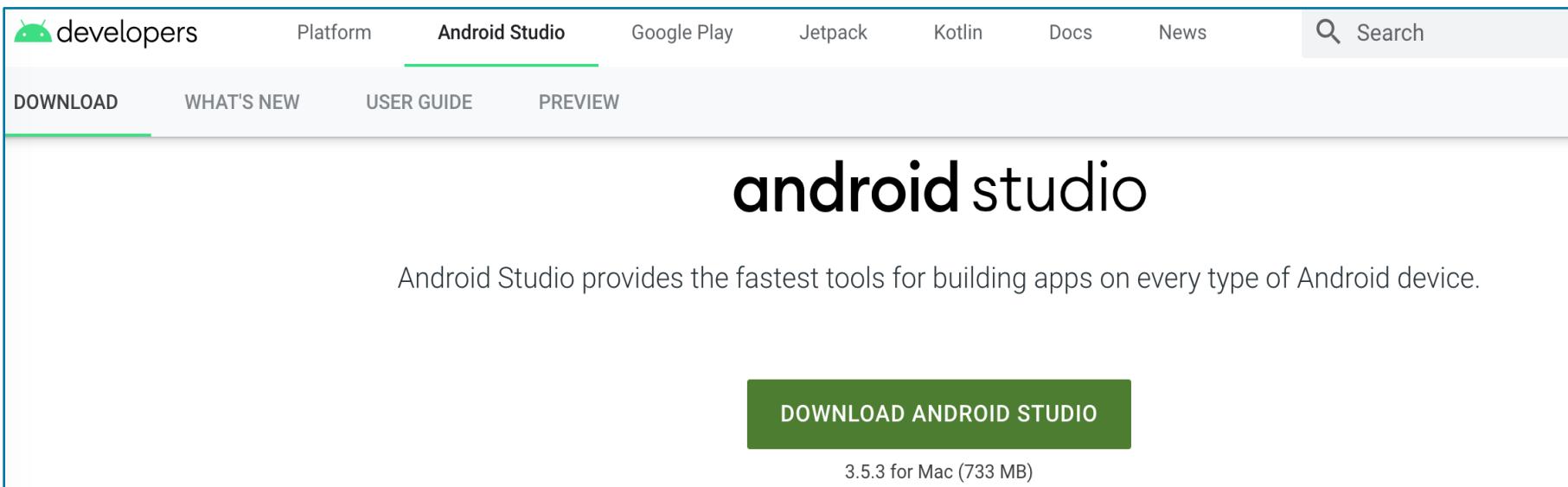
# 4. Thiết lập môi trường lập trình Android

## »» Phản mềm

- Android SDK và công cụ Android Studio



Android Studio



The screenshot shows the official Android Studio download page on the developer.android.com website. The top navigation bar includes links for Platform, Android Studio (which is underlined), Google Play, Jetpack, Kotlin, Docs, and News, along with a search bar. Below the navigation is a horizontal menu with DOWNLOAD, WHAT'S NEW, USER GUIDE, and PREVIEW. The main content area features the text "android studio" in large letters, followed by the subtext "Android Studio provides the fastest tools for building apps on every type of Android device." A prominent green button labeled "DOWNLOAD ANDROID STUDIO" is centered at the bottom, with the text "3.5.3 for Mac (733 MB)" underneath it.

*Link: <https://developer.android.com/studio#downloads>*

# 4. Thiết lập môi trường lập trình Android

## »» Phản mềm

- Android SDK và công cụ Android Studio



Android Studio

Platform	Android Studio package	Size
Windows (64-bit)	<a href="#">android-studio-ide-191.6010548-windows.exe</a> Recommended	718 MB
	<a href="#">android-studio-ide-191.6010548-windows.zip</a> No .exe installer	721 MB
Windows (32-bit)	<a href="#">android-studio-ide-191.6010548-windows32.zip</a> No .exe installer	721 MB
Mac (64-bit)	<a href="#">android-studio-ide-191.6010548-mac.dmg</a>	733 MB
Linux (64-bit)	<a href="#">android-studio-ide-191.6010548-linux.tar.gz</a>	738 MB
Chrome OS	<a href="#">android-studio-ide-191.6010548-cros.deb</a>	620 MB

# 4. Thiết lập môi trường lập trình Android

## »» Phản mềm

- Android SDK và công cụ Android Studio



Android Studio

These tools are included in Android Studio.

Platform	SDK tools package	Size
Windows	<a href="https://dl.google.com/dl/android/studio/4.3.3/4333796-windows.zip">sdk-tools-windows-4333796.zip</a>	148 MB
Mac	<a href="https://dl.google.com/dl/android/studio/4.3.3/4333796-mac.zip">sdk-tools-darwin-4333796.zip</a>	98 MB
Linux	<a href="https://dl.google.com/dl/android/studio/4.3.3/4333796-linux.zip">sdk-tools-linux-4333796.zip</a>	147 MB

Link: <https://developer.android.com/studio#downloads>

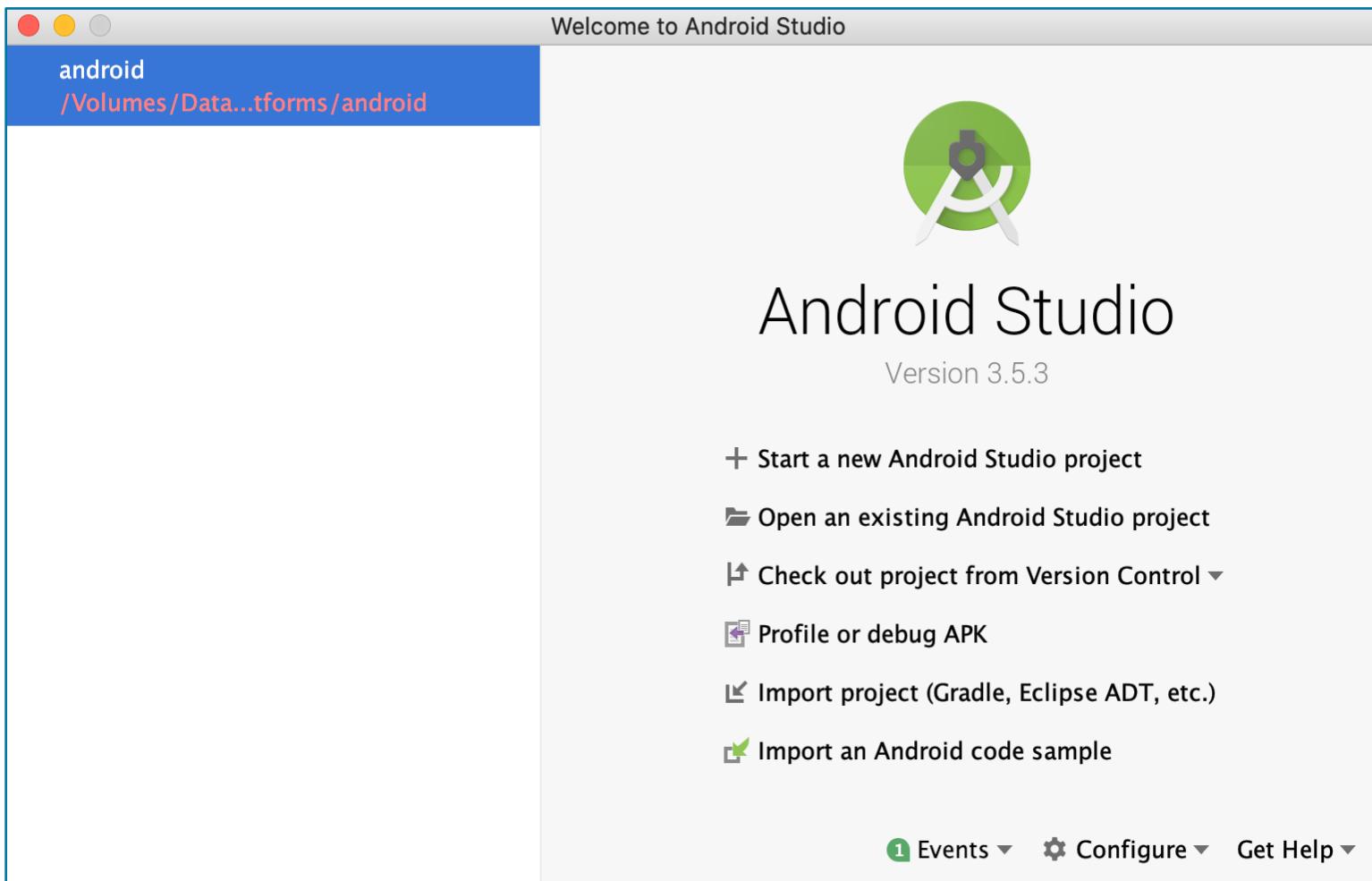
# 4. Thiết lập môi trường lập trình Android

## »» Phản mềm

- Android SDK và công cụ Android Studio

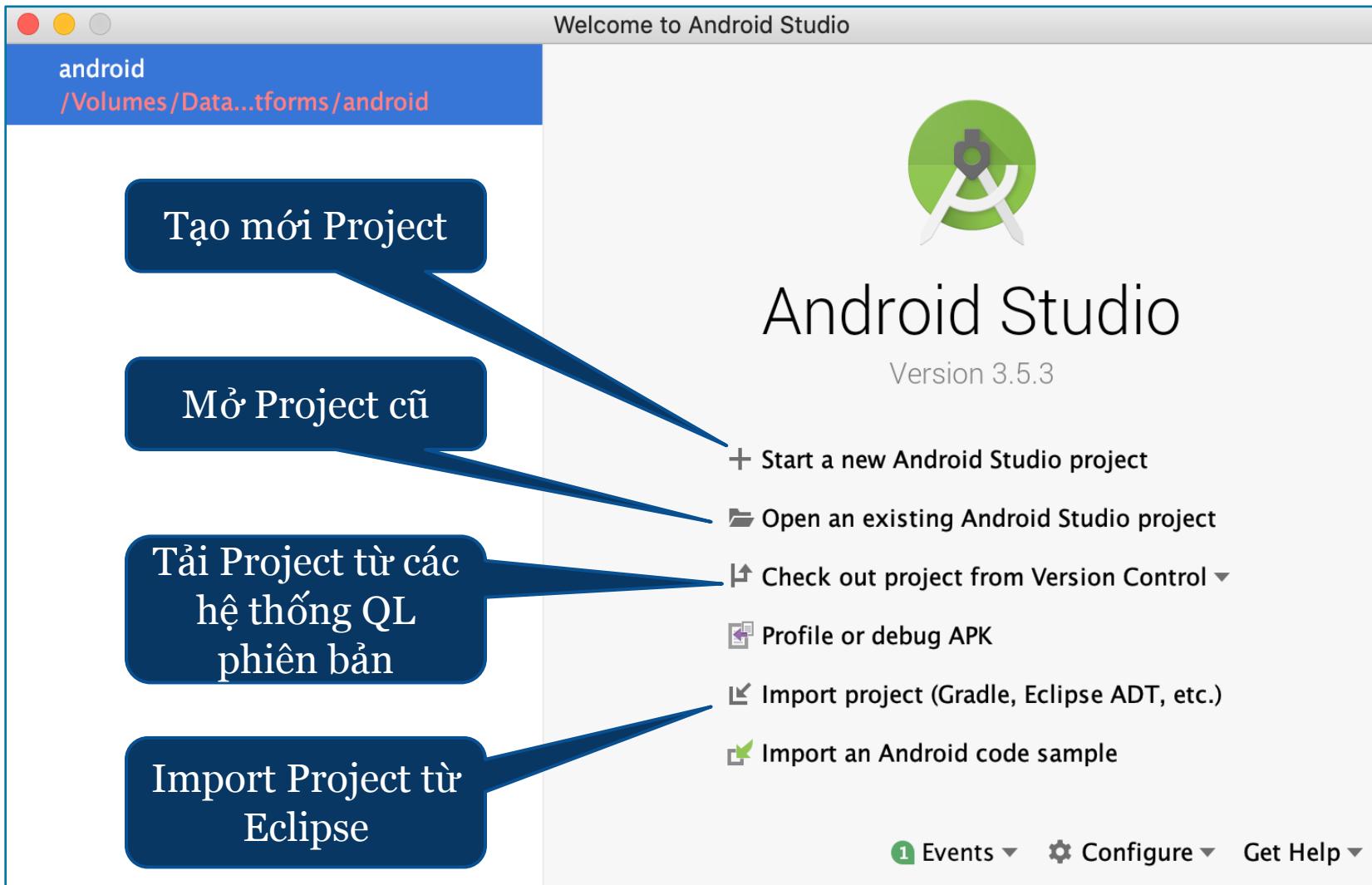


Android Studio



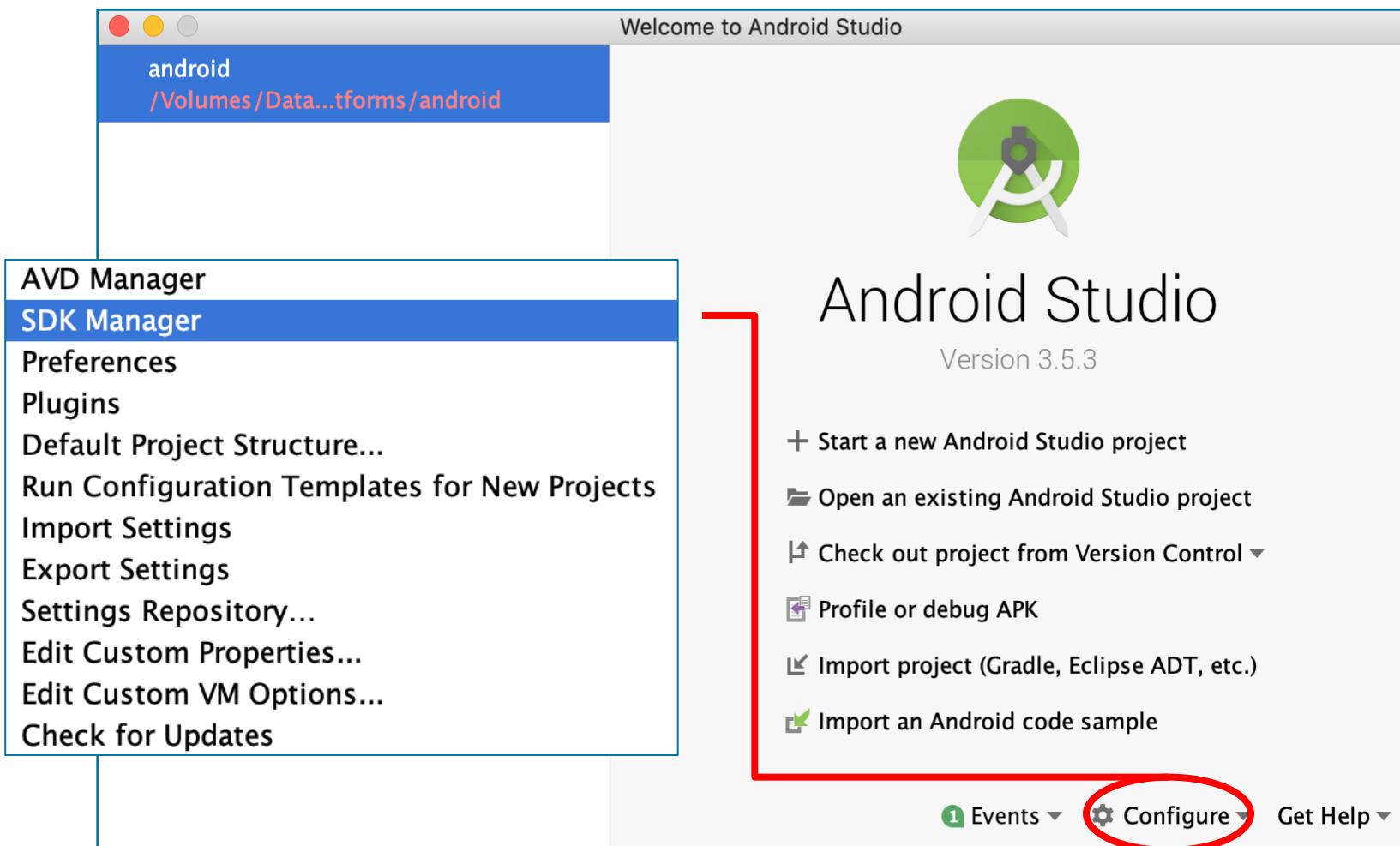
## 5. Công cụ Android Studio

### ➤ Các thành phần chức năng thường dùng



## 5. Công cụ Android Studio

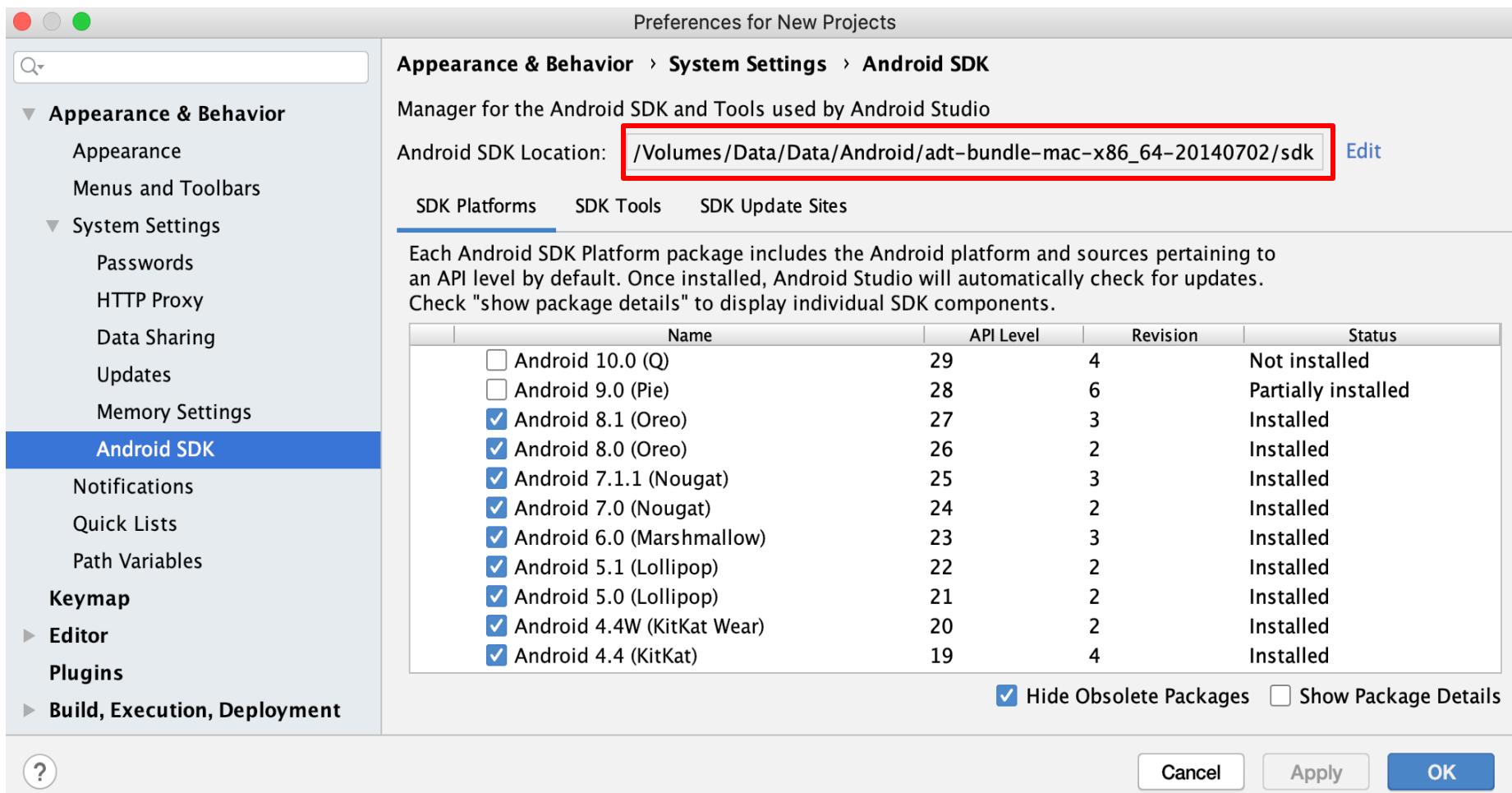
- Các thành phần cấu hình thường dùng
  - SDK Manager



# 5. Công cụ Android Studio

## ➤ Các thành phần cấu hình thường dùng

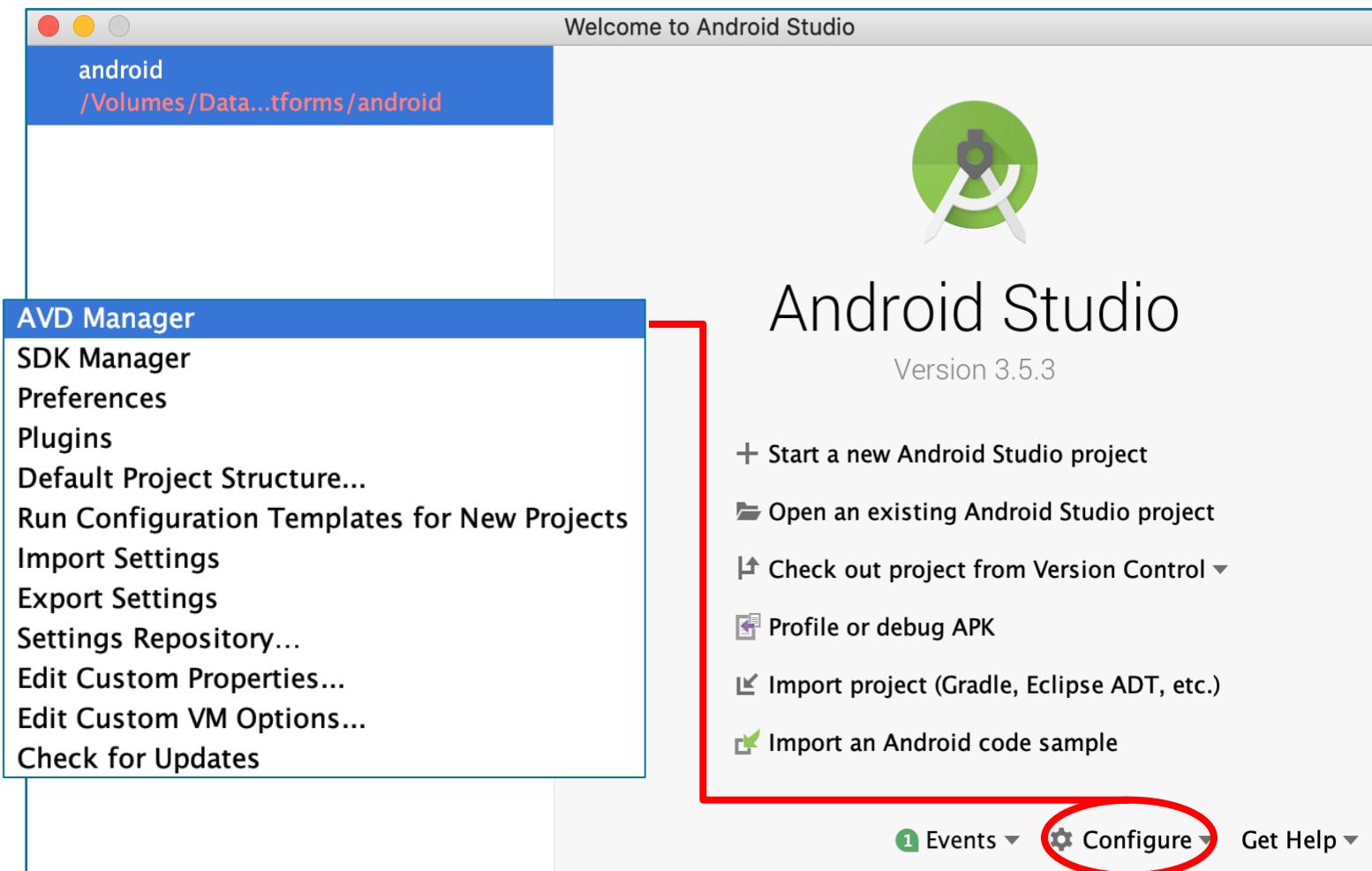
- SDK Manager



## 5. Công cụ Android Studio

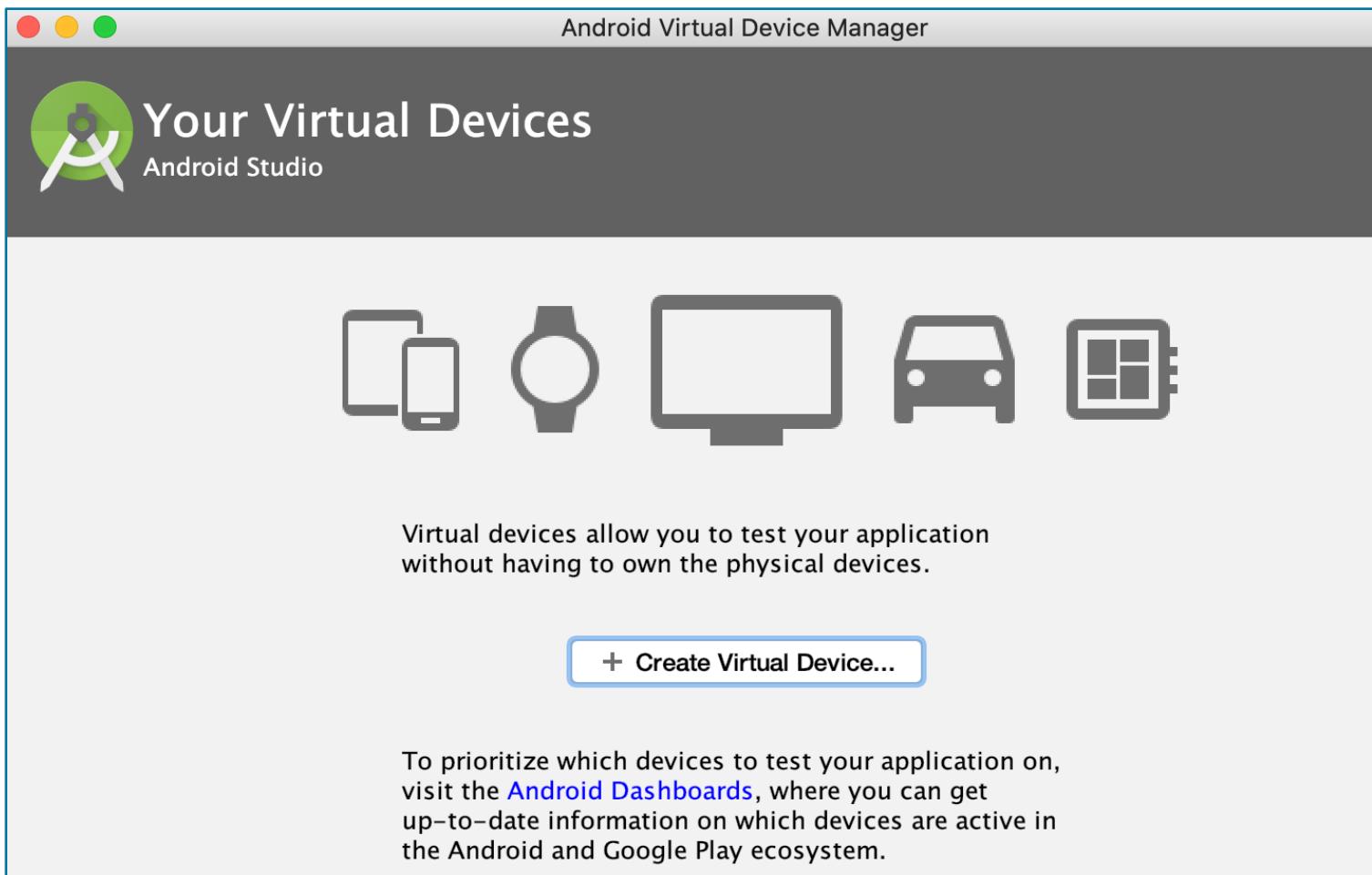
### ➤ Các thành phần cấu hình thường dùng

- AVD Manager



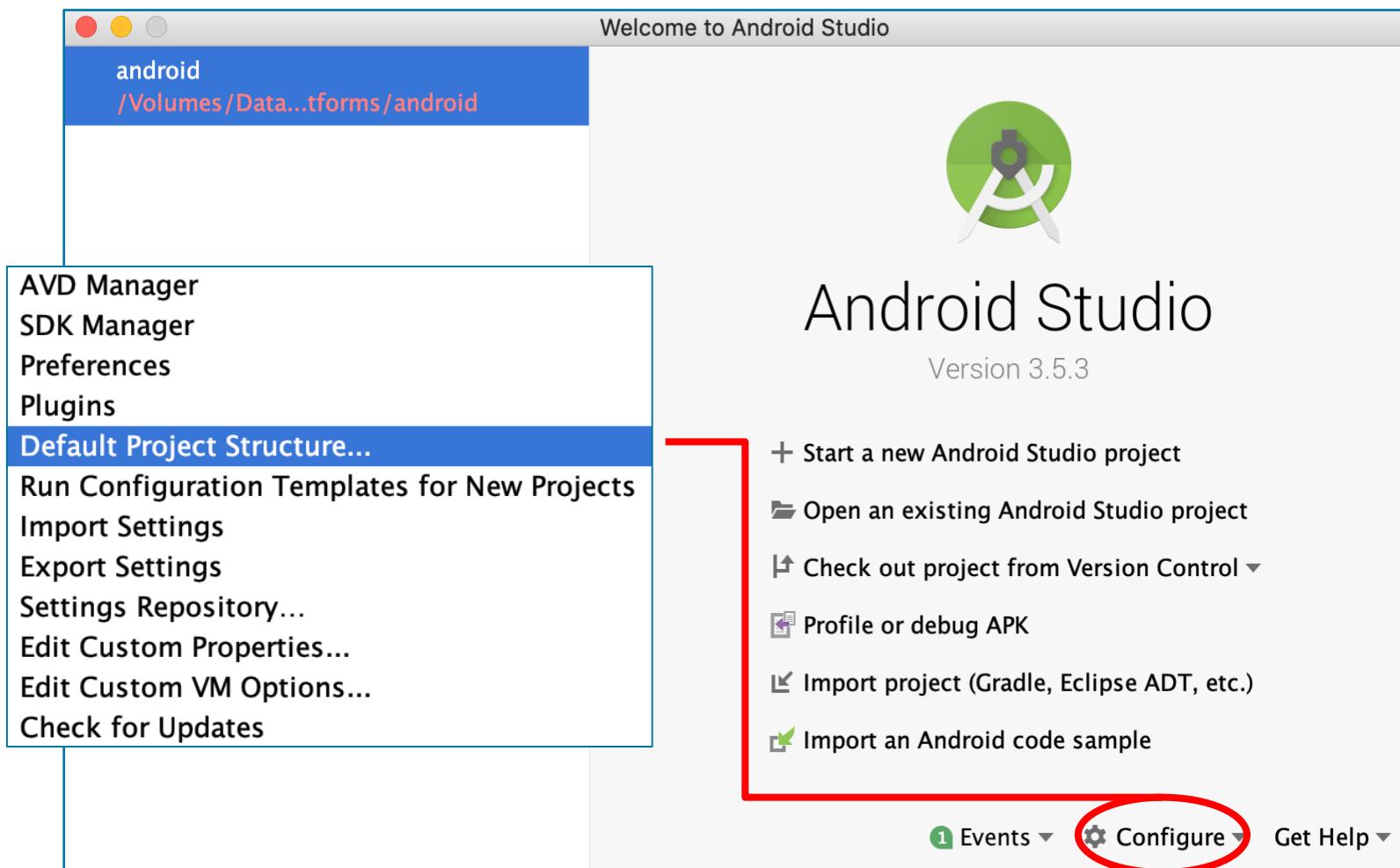
## 5. Công cụ Android Studio

- Các thành phần cấu hình thường dùng
  - AVD Manager



## 5. Công cụ Android Studio

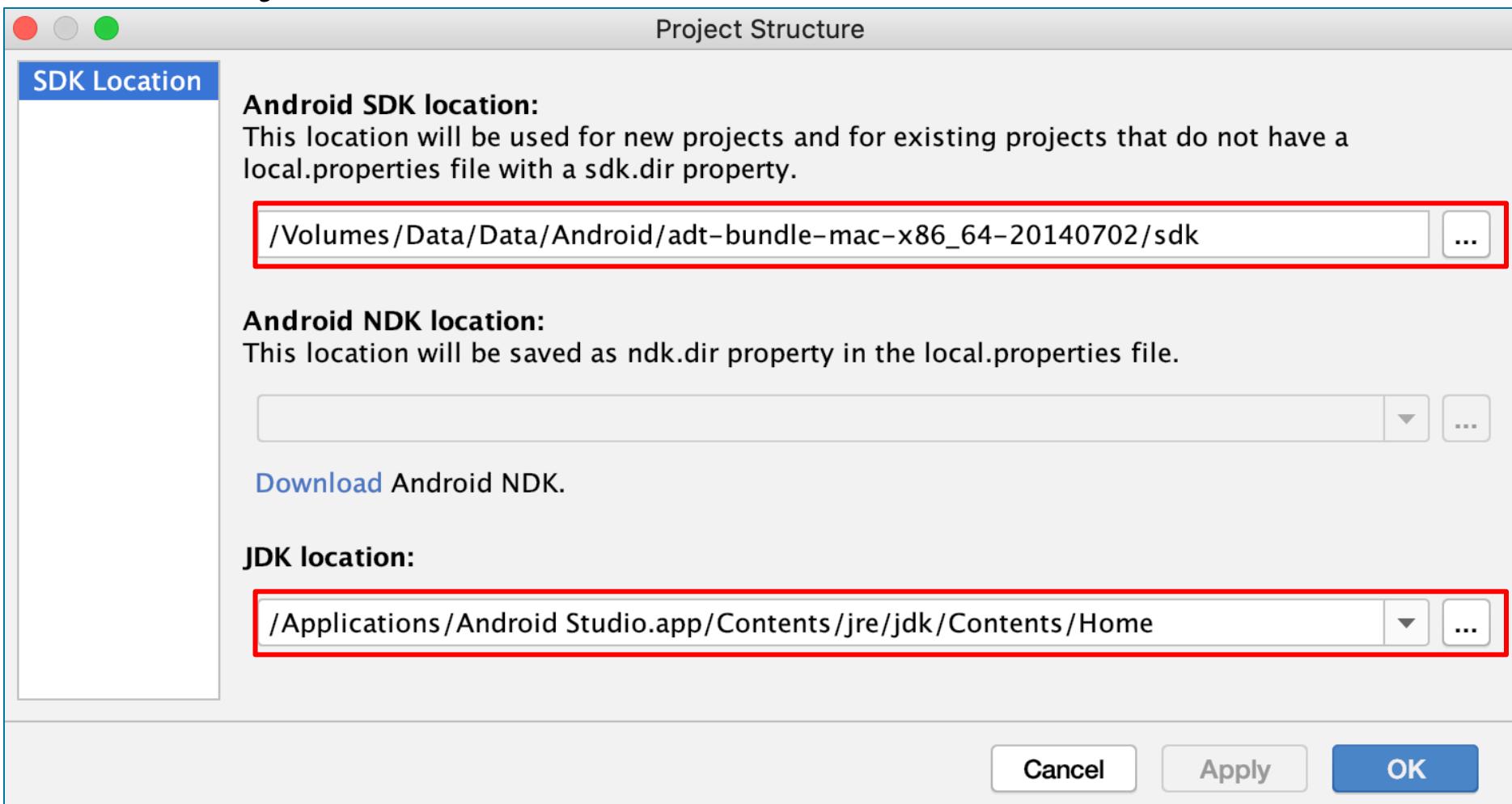
- Các thành phần cấu hình thường dùng
  - Default Project Structure...



## 5. Công cụ Android Studio

- Các thành phần cấu hình thường dùng

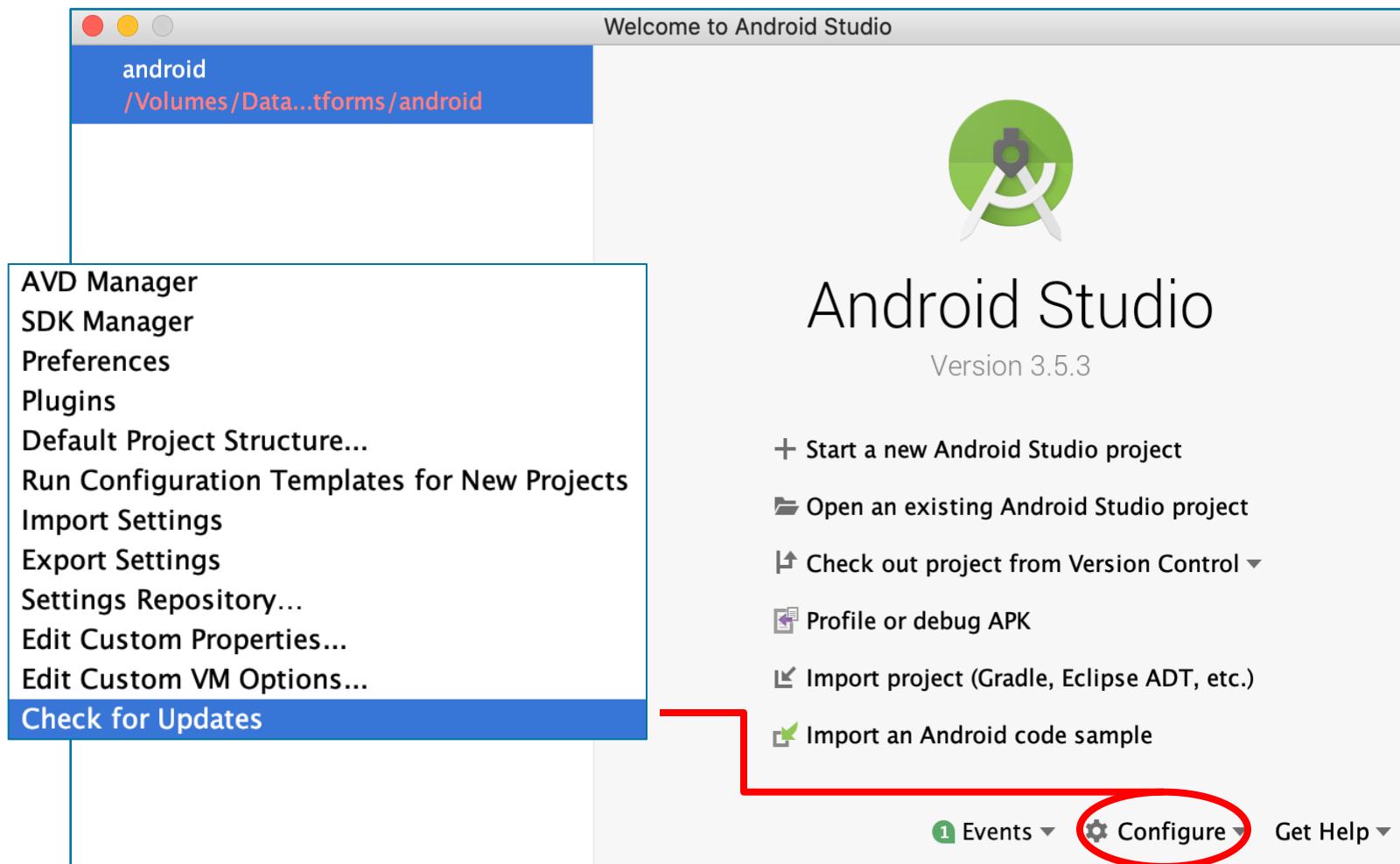
- Default Project Structure...



## 5. Công cụ Android Studio

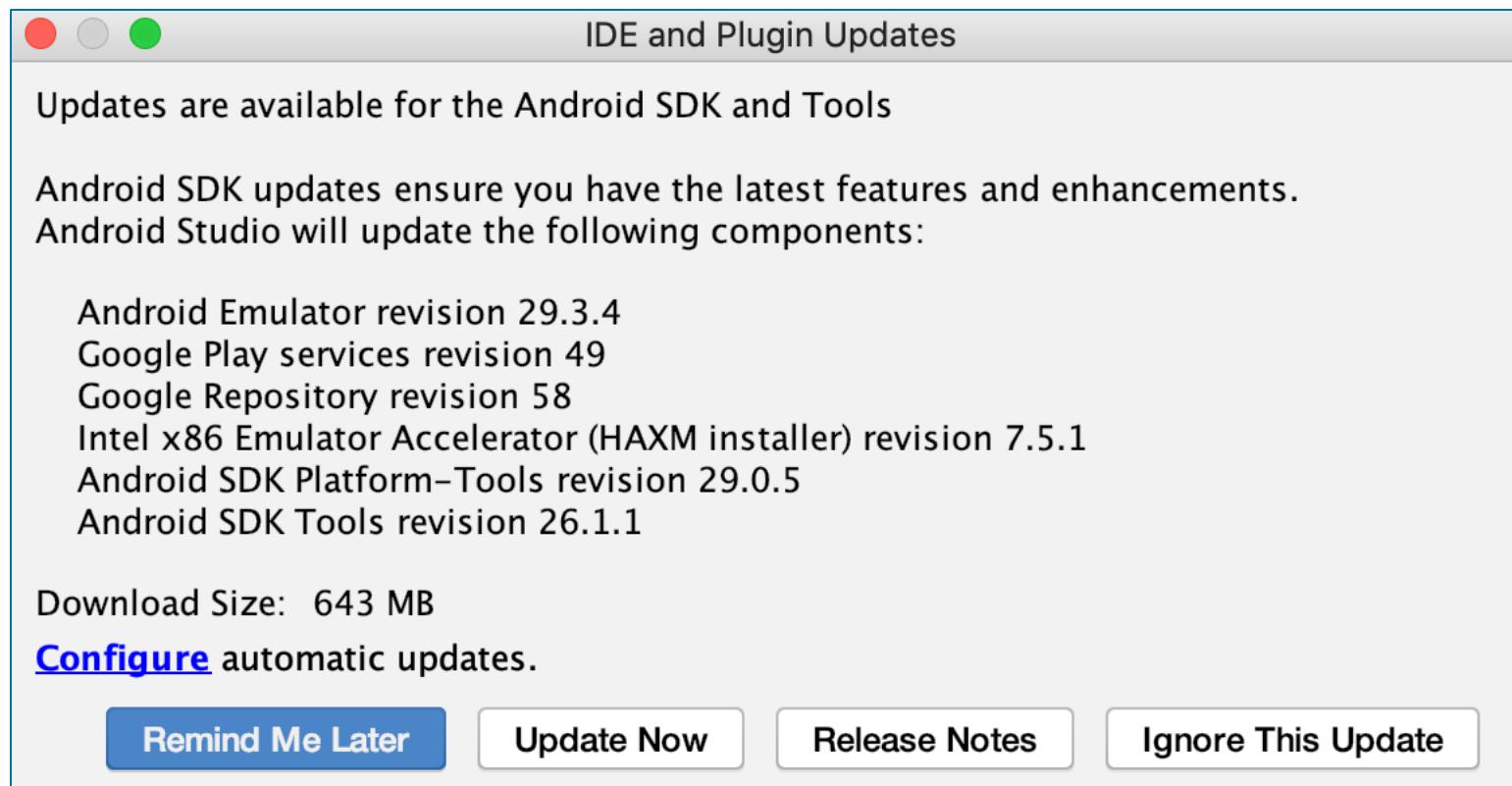
### ➤ Các thành phần cấu hình thường dùng

- Check for Updates



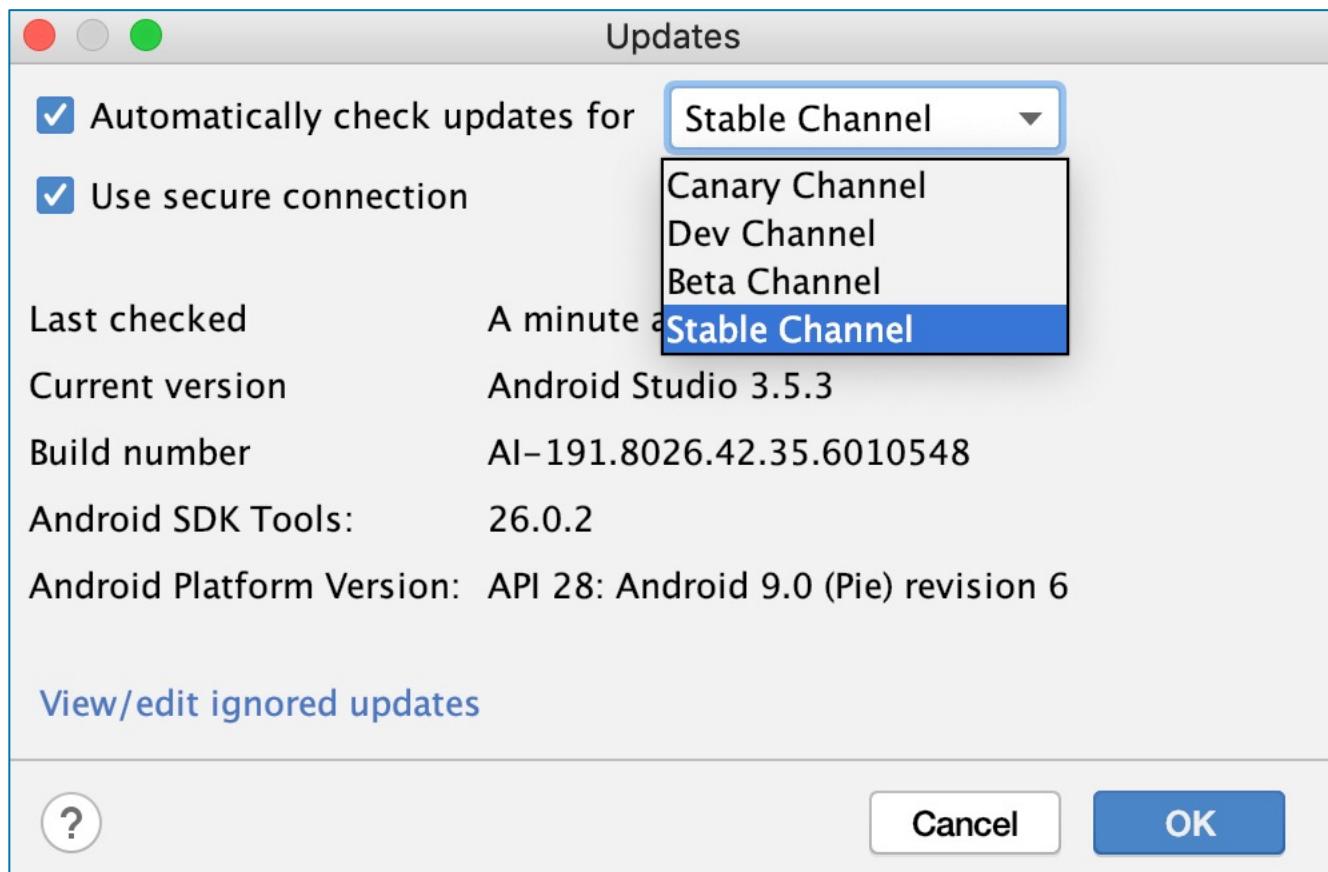
## 5. Công cụ Android Studio

- Các thành phần cấu hình thường dùng
  - Check for Updates



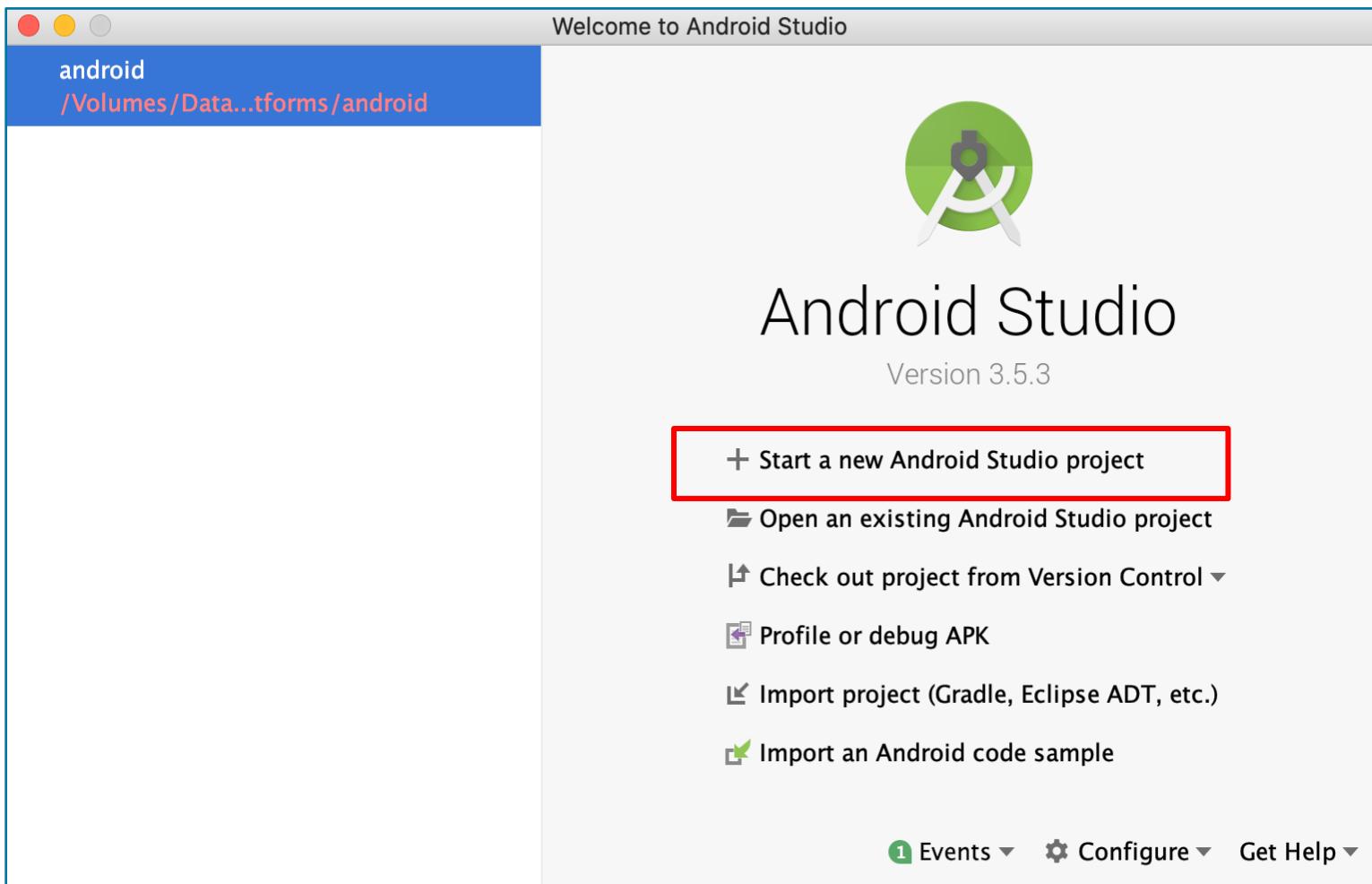
## 5. Công cụ Android Studio

- Các thành phần cấu hình thường dùng
  - Check for Updates



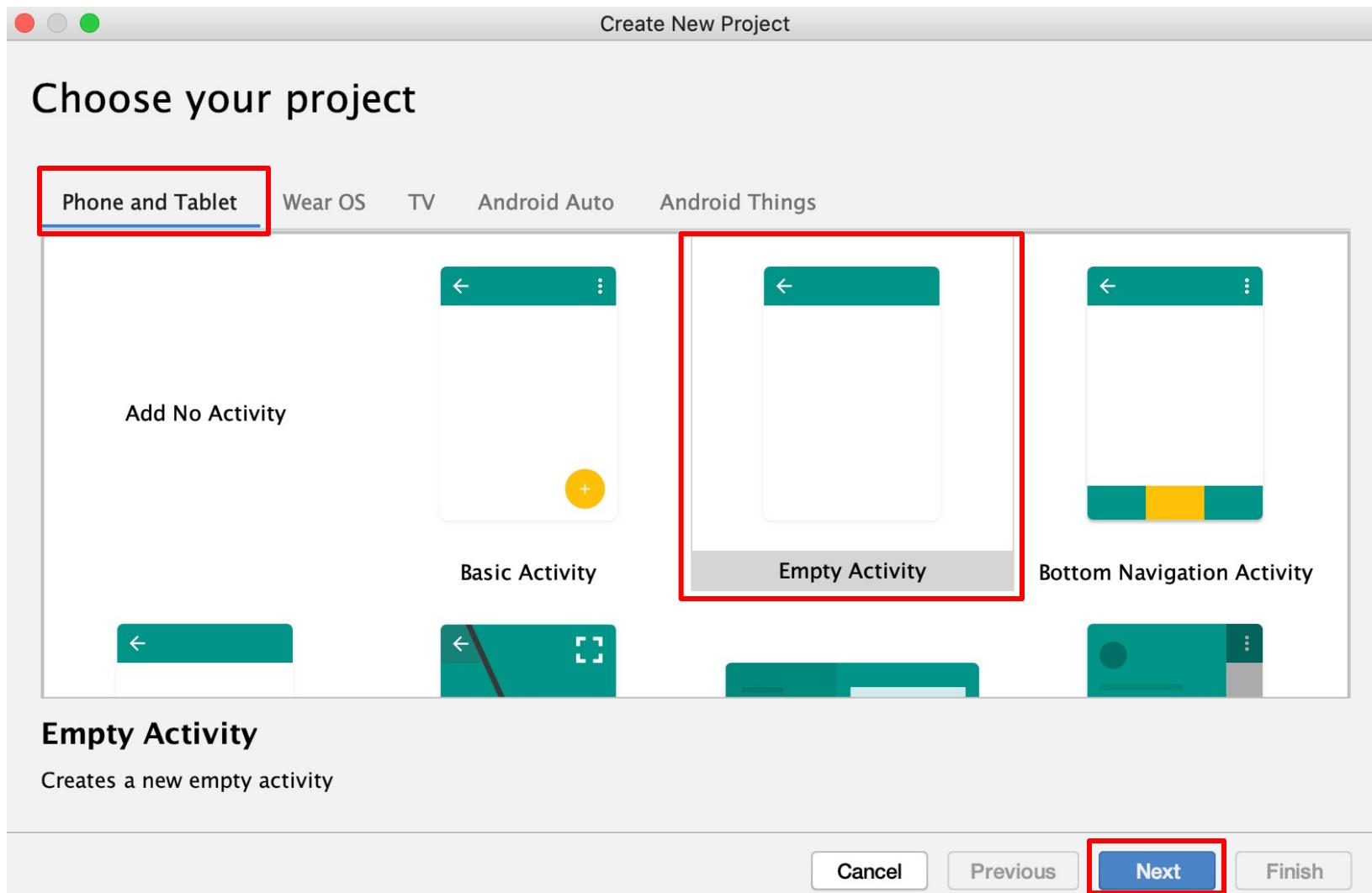
## 5. Công cụ Android Studio

### ➤ Tạo project



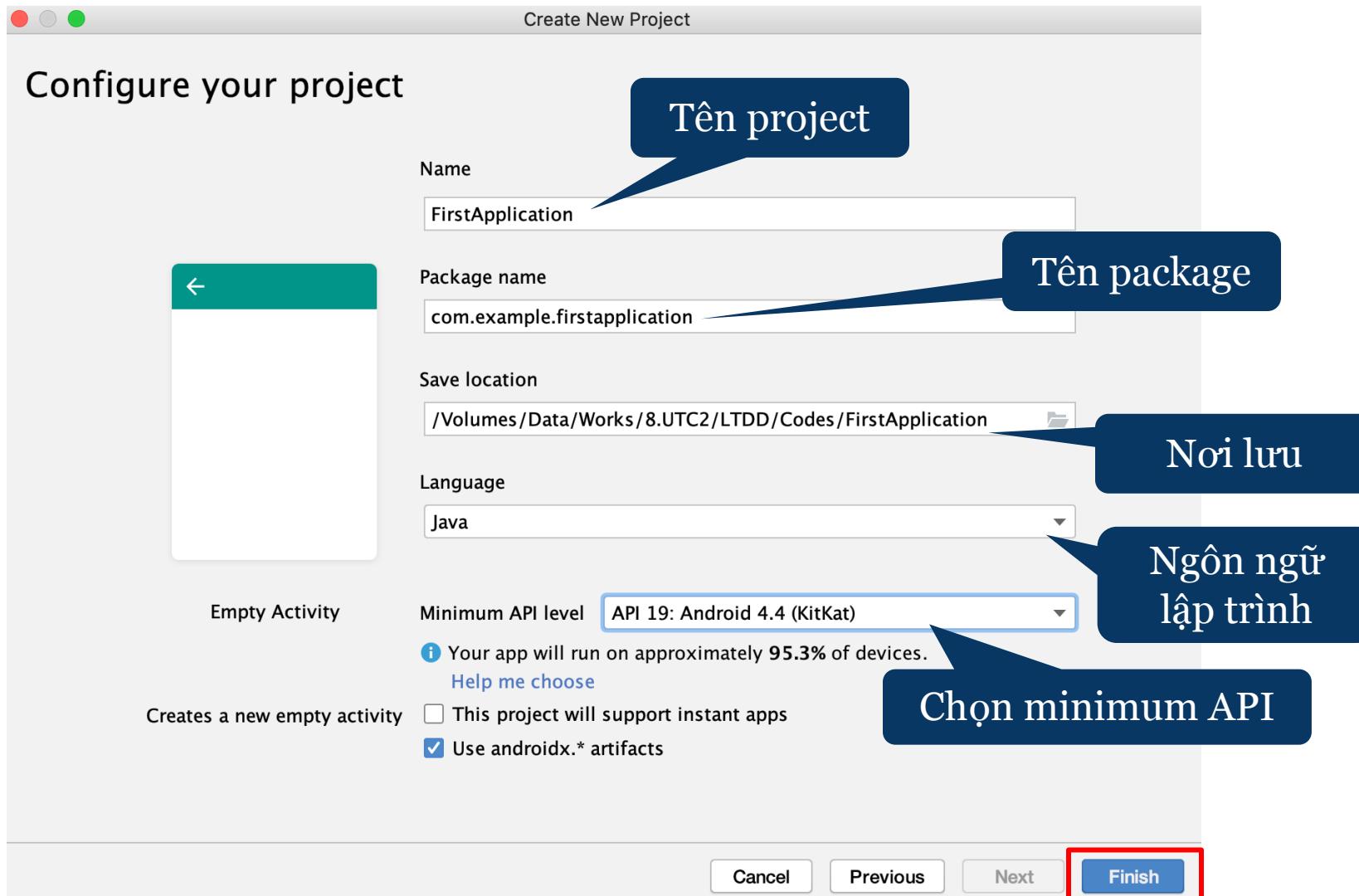
# 5. Công cụ Android Studio

## ➤ Tạo project



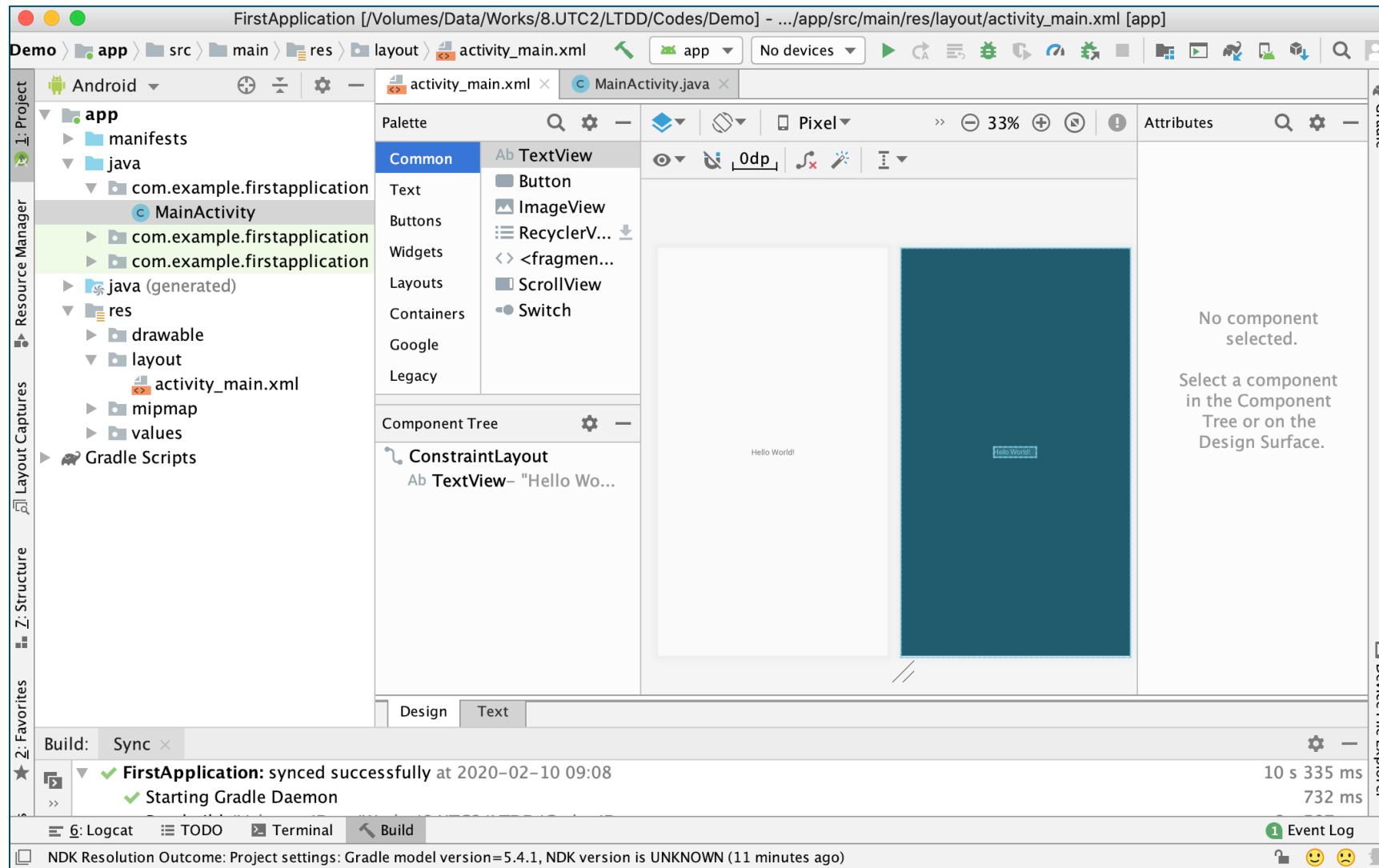
# 5. Công cụ Android Studio

## ➤ Tạo project



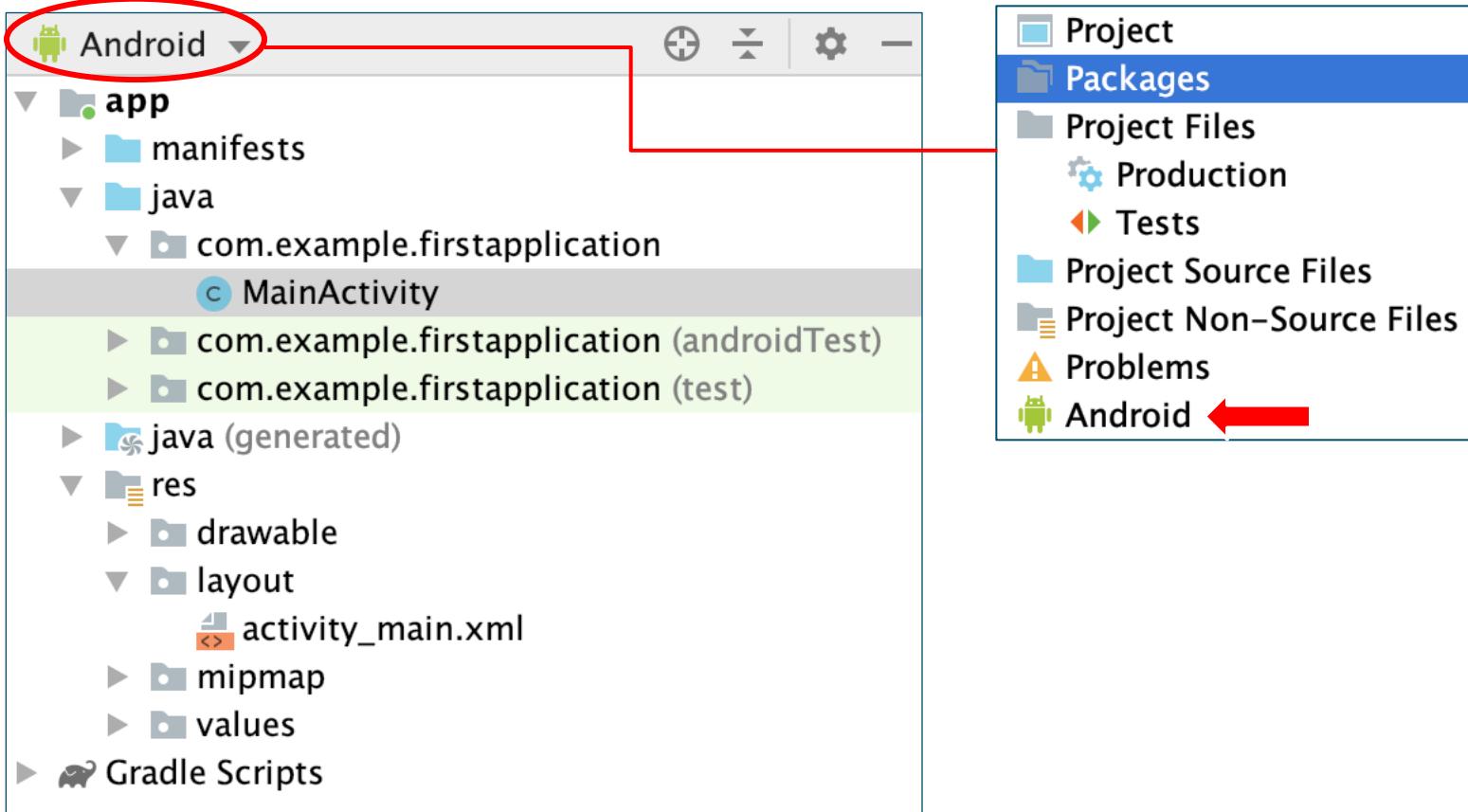
# 5. Công cụ Android Studio

## ➤ Tạo project



## 5. Công cụ Android Studio

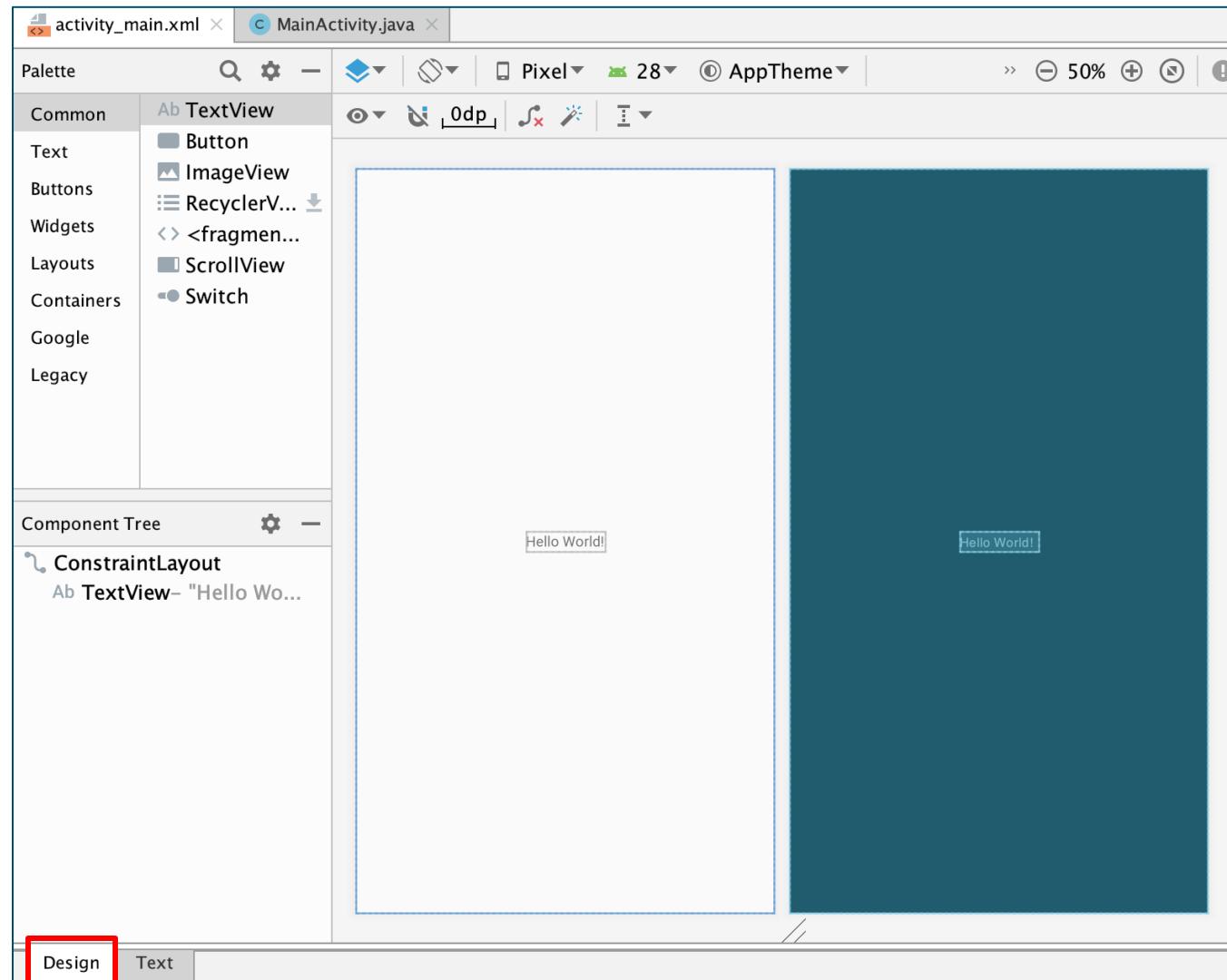
- Thành phần giao diện thường thao tác
  - Thay đổi layout xem cấu trúc project



## 5. Công cụ Android Studio

### ➤ Thành phần giao diện thường thao tác

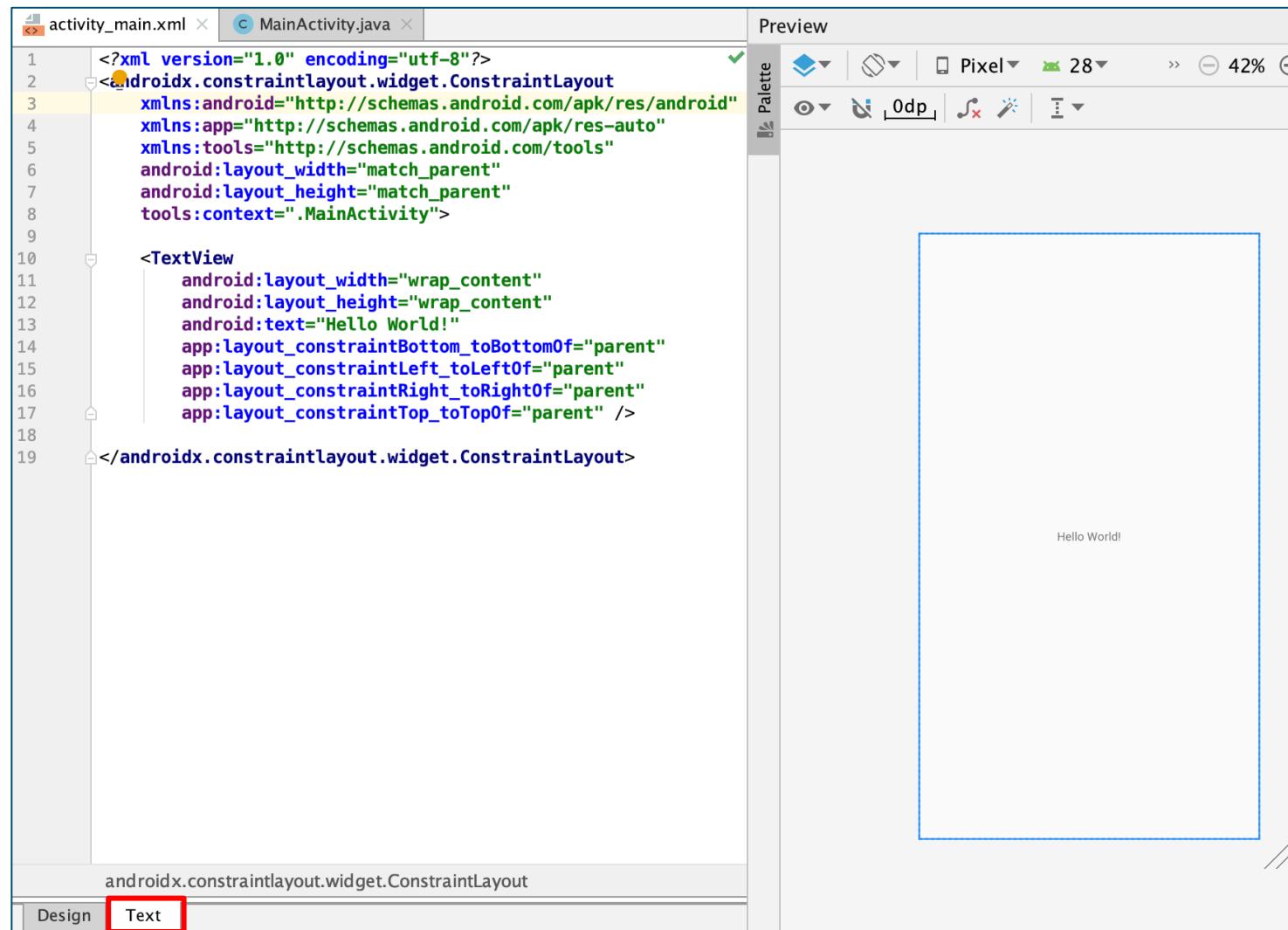
- Màn hình thiết kế giao diện



# 5. Công cụ Android Studio

## ➤ Thành phần giao diện thường thao tác

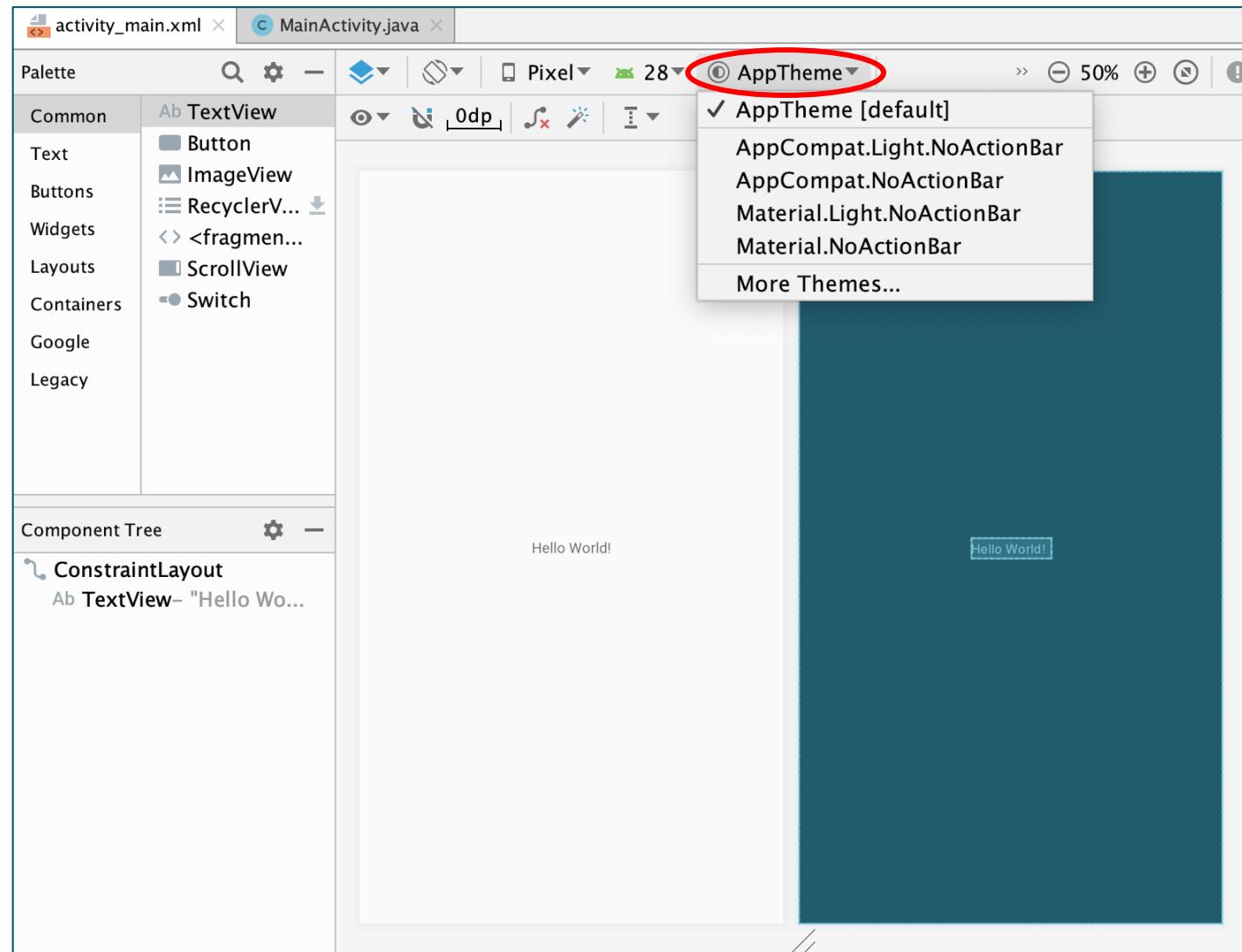
- Màn hình thiết kế giao diện



## 5. Công cụ Android Studio

- Thành phần giao diện thường thao tác

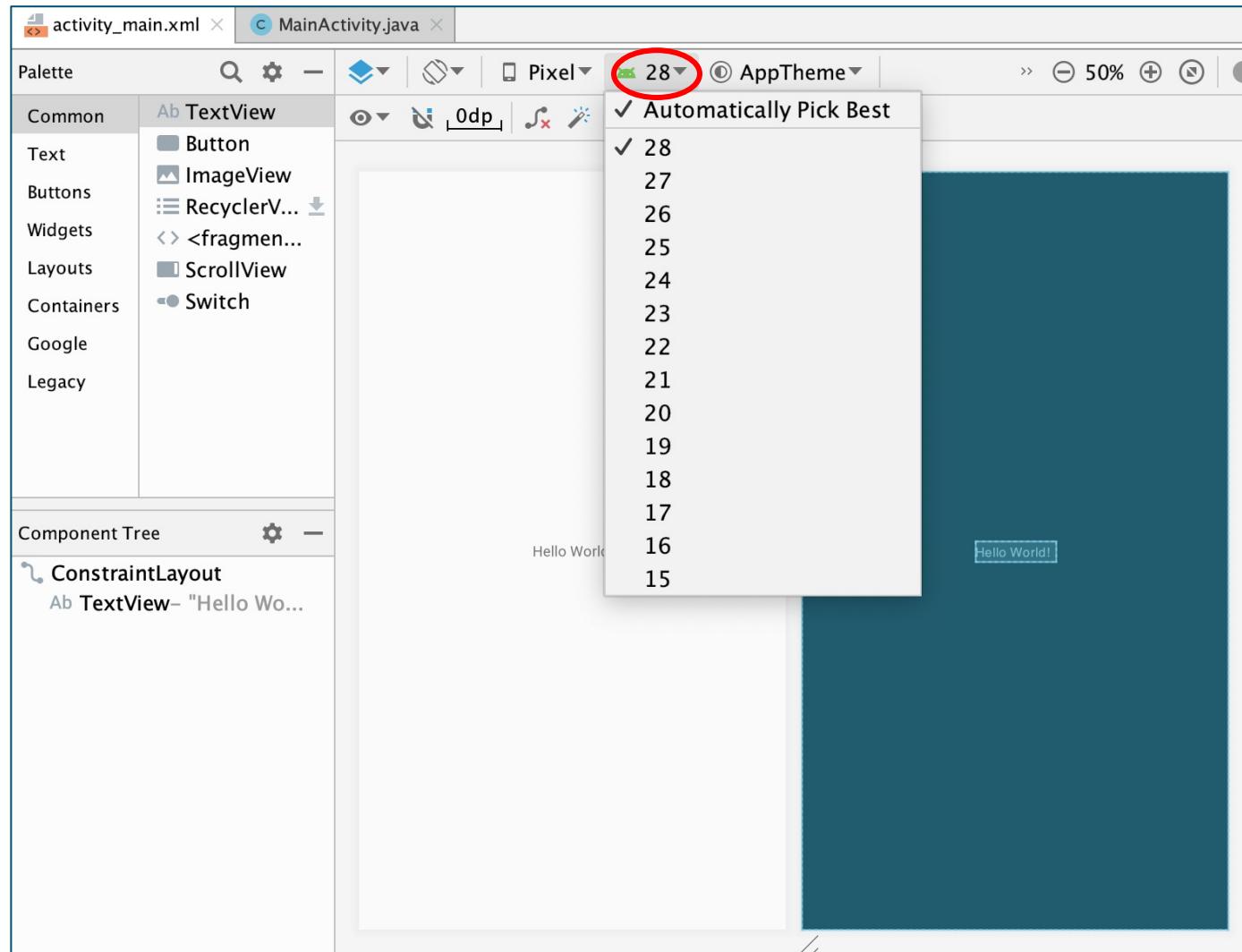
- Đổi theme



# 5. Công cụ Android Studio

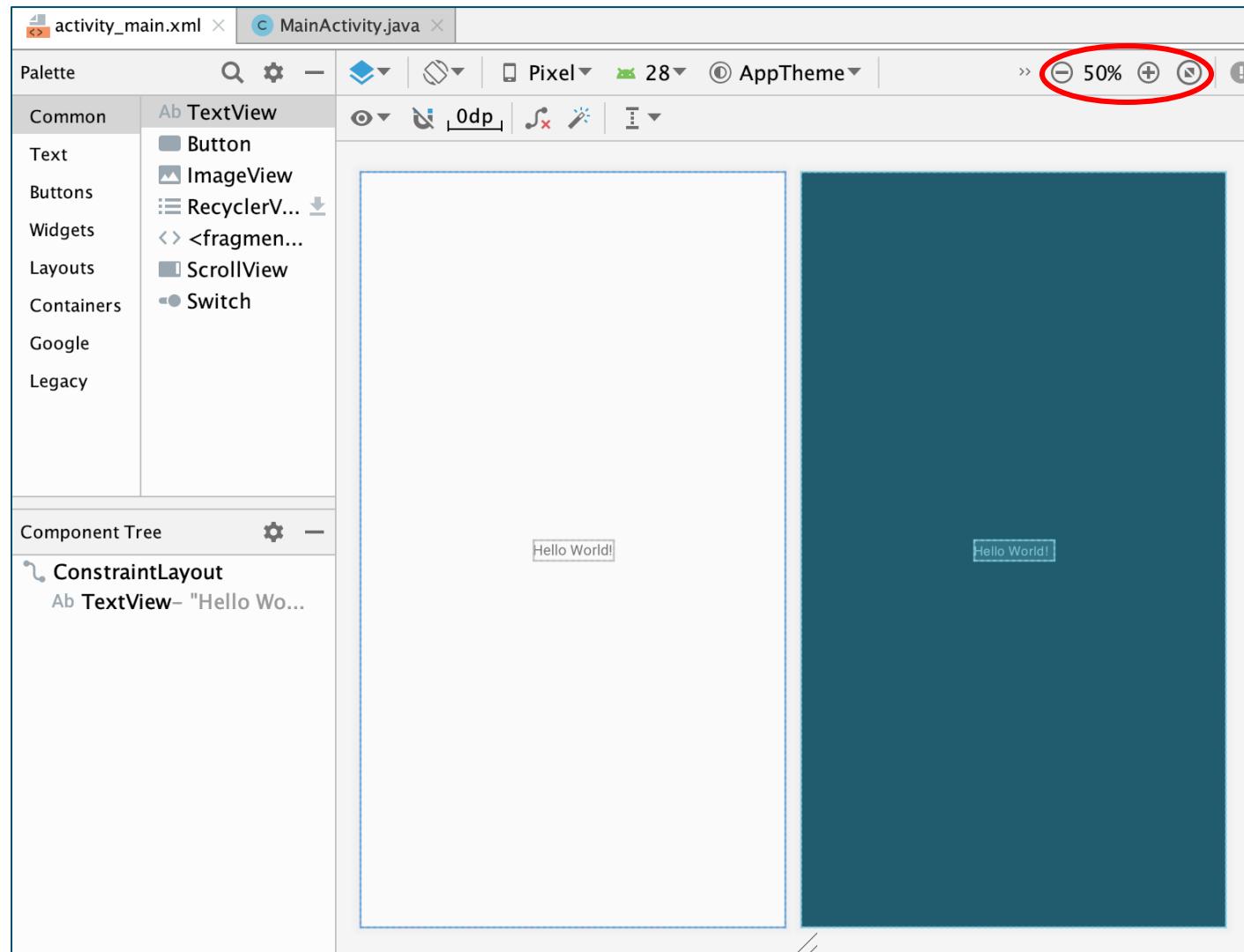
- Thành phần giao diện thường thao tác

- API version



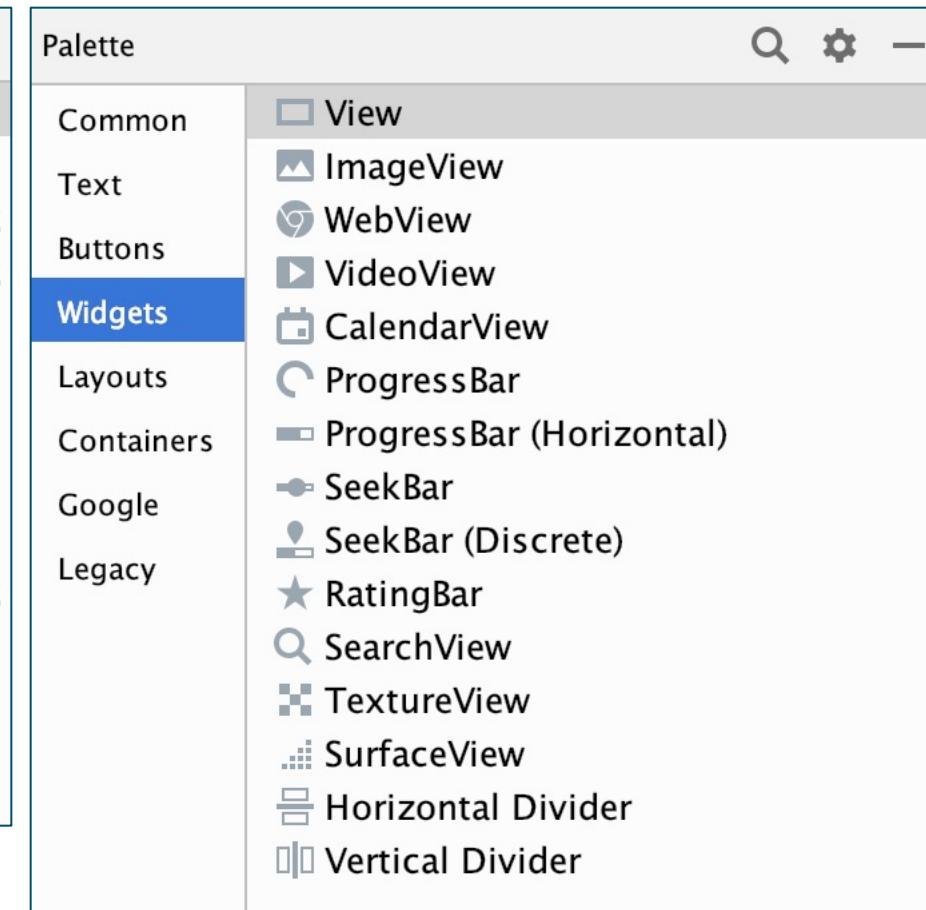
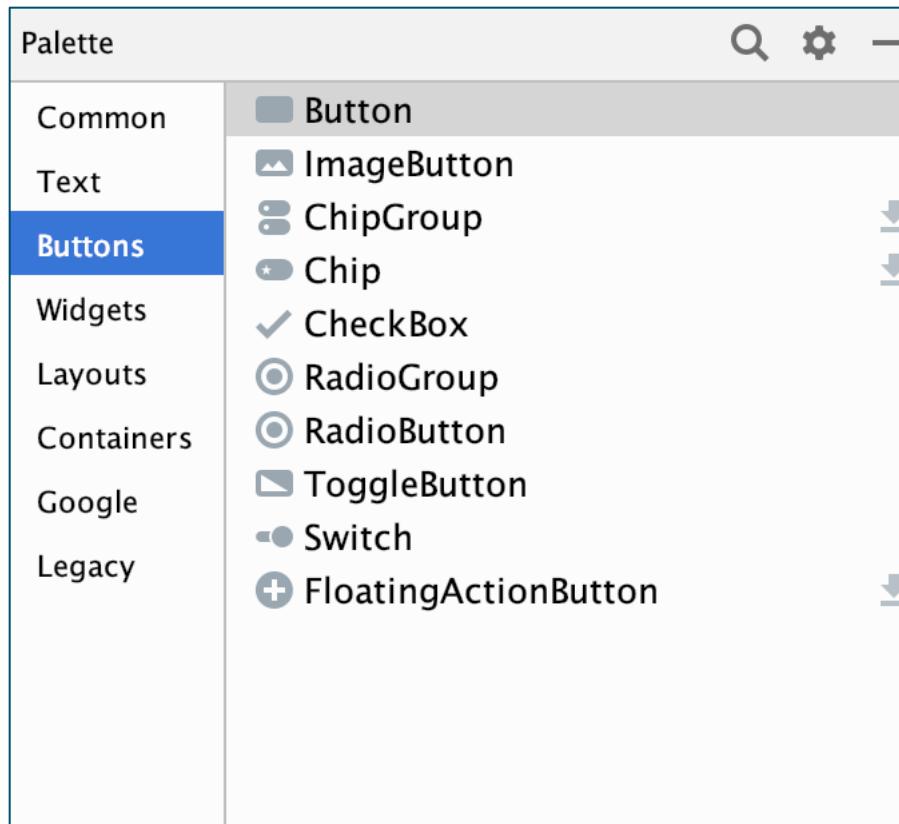
## 5. Công cụ Android Studio

- Thành phần giao diện thường thao tác
  - Zoom



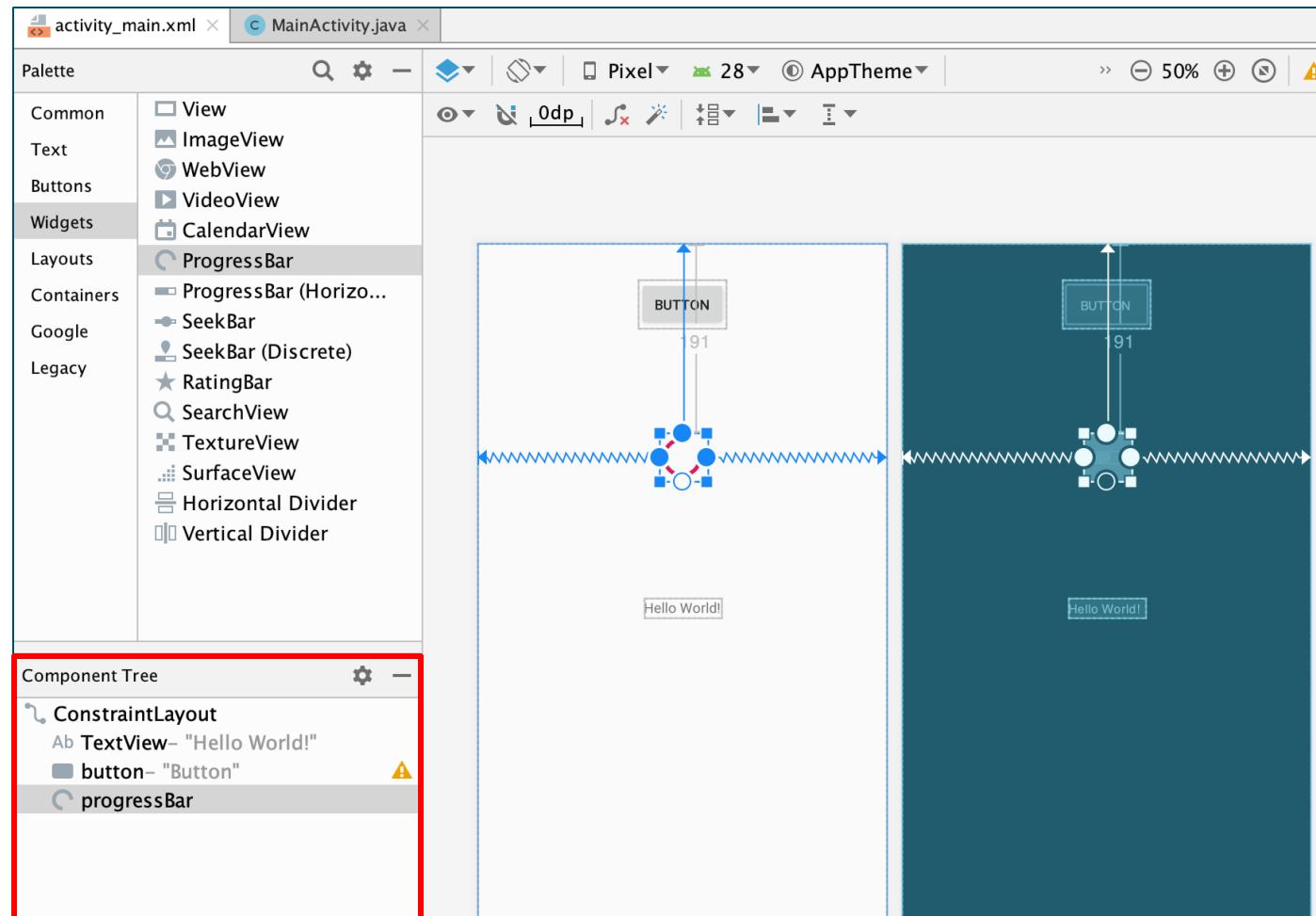
## 5. Công cụ Android Studio

- Thành phần giao diện thường thao tác
  - Palette: chọn control, layouts, ...



## 5. Công cụ Android Studio

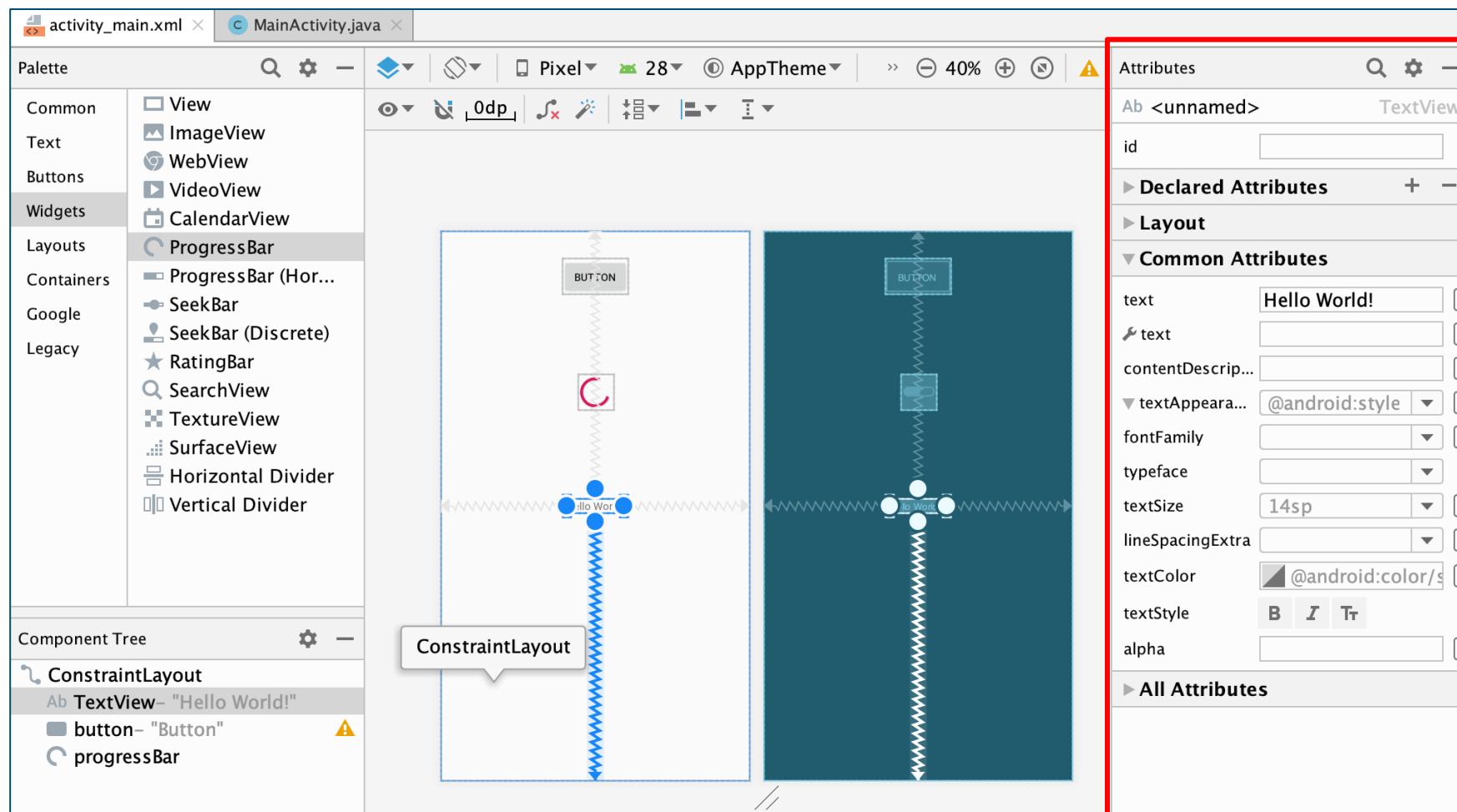
- Thành phần giao diện thường thao tác
  - Màn hình Component Tree



# 5. Công cụ Android Studio

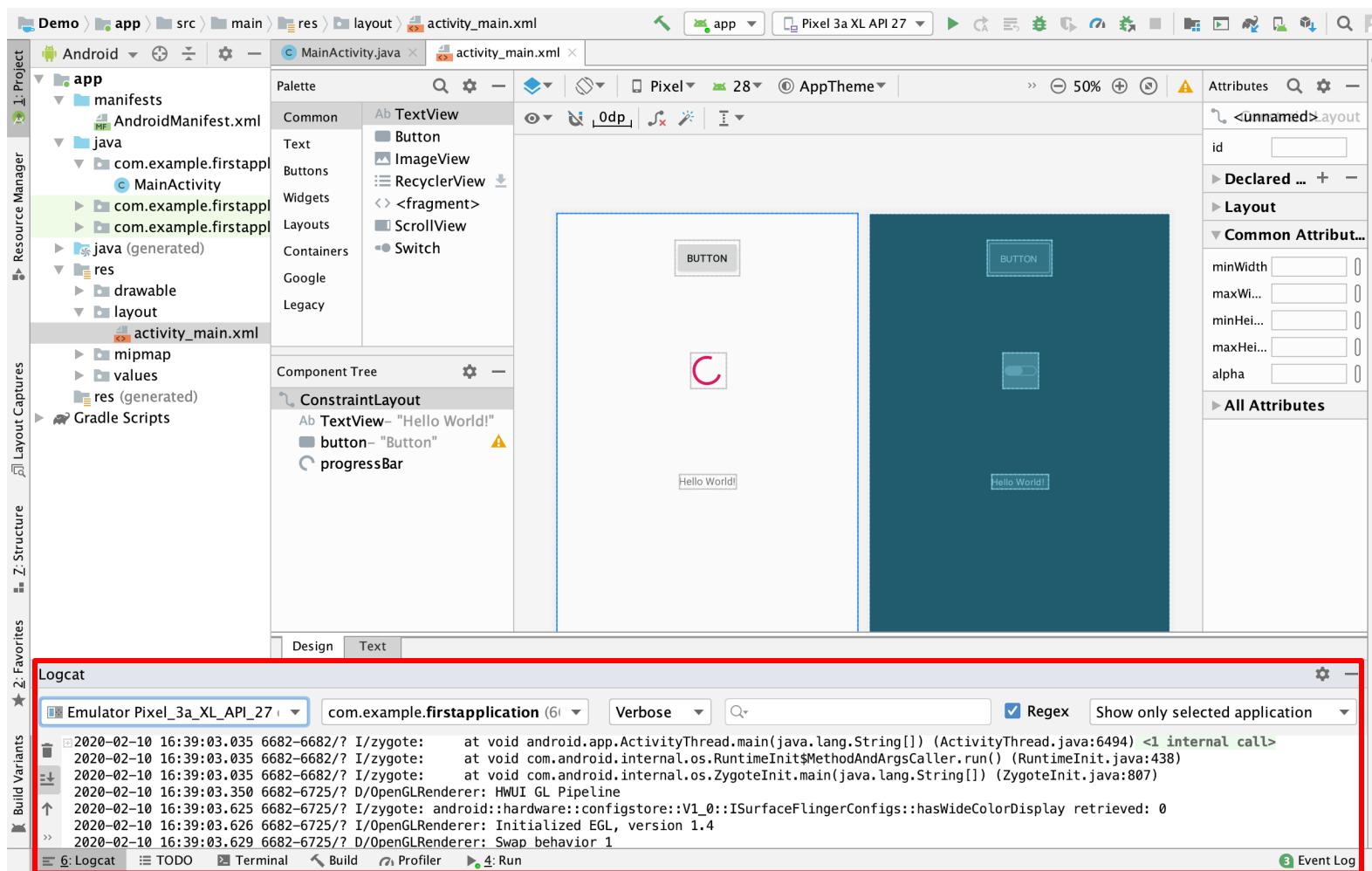
➤ Thành phần giao diện thường thao tác

- Attributes



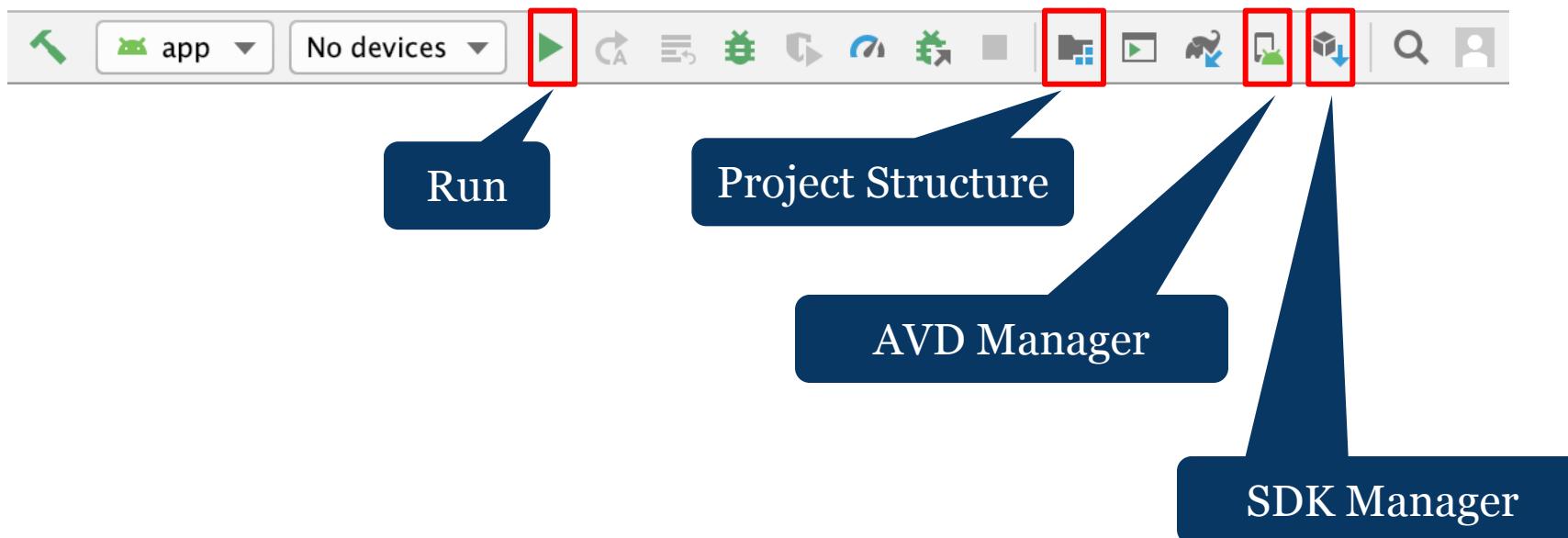
# 5. Công cụ Android Studio

- Thành phần giao diện thường thao tác
  - Logcat: hỗ trợ theo dõi lỗi trong quá trình lập trình



## 5. Công cụ Android Studio

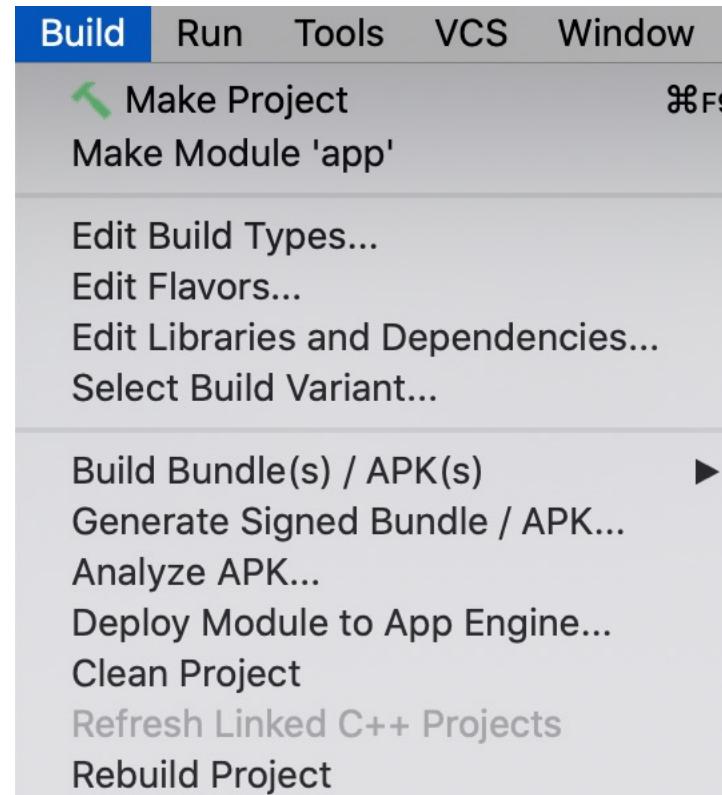
- Thành phần giao diện thường thao tác
  - Toolbar: Run, Project Structure, AVD Manager, SDK Manager



## 5. Công cụ Android Studio

### ➤ Thành phần giao diện thường thao tác

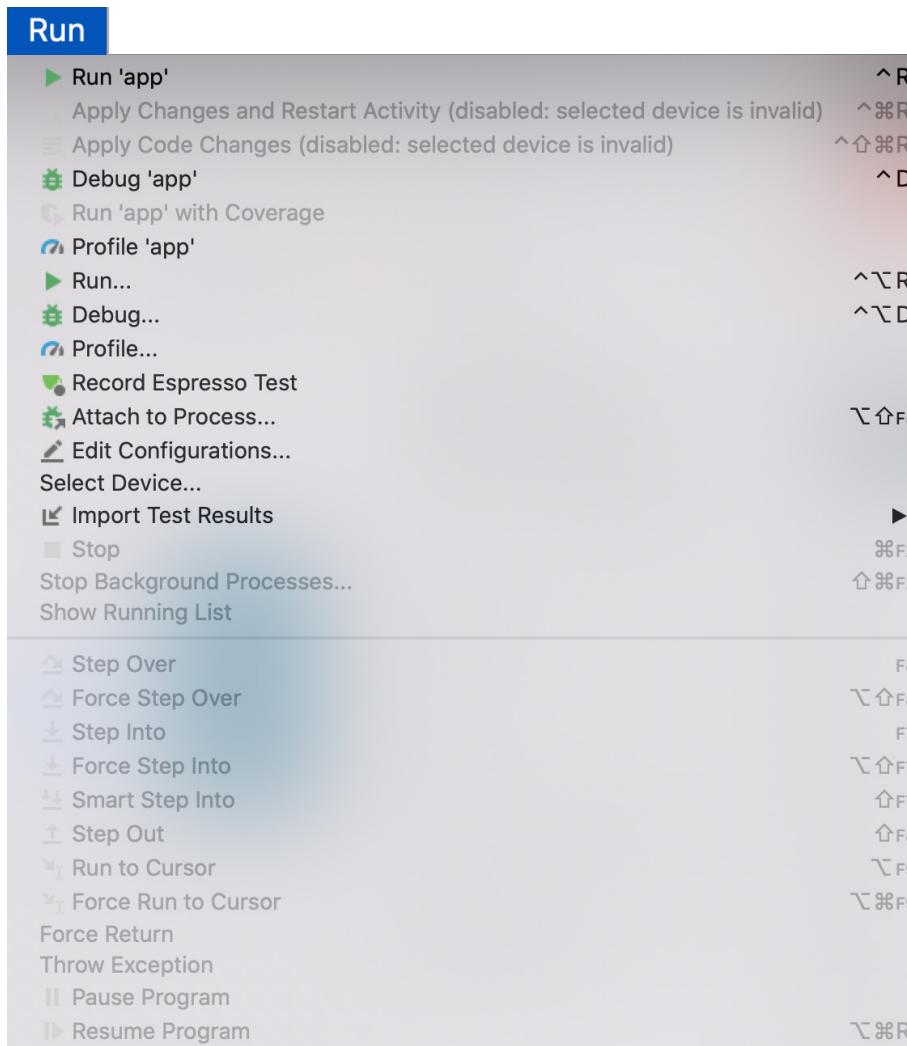
- Menu:
  - Build -> Clean project
  - Build -> Rebuild project
  - Build -> Bundle(s)/APK(s)
  - Build -> Generate Signed  
Bundle/APK



## 5. Công cụ Android Studio

### ➤ Thành phần giao diện thường thao tác

- Menu: Run



# 5. Công cụ Android Studio

## ➤ Cấu trúc Project

Android Manifest

Source code Java

Tài nguyên hình ảnh

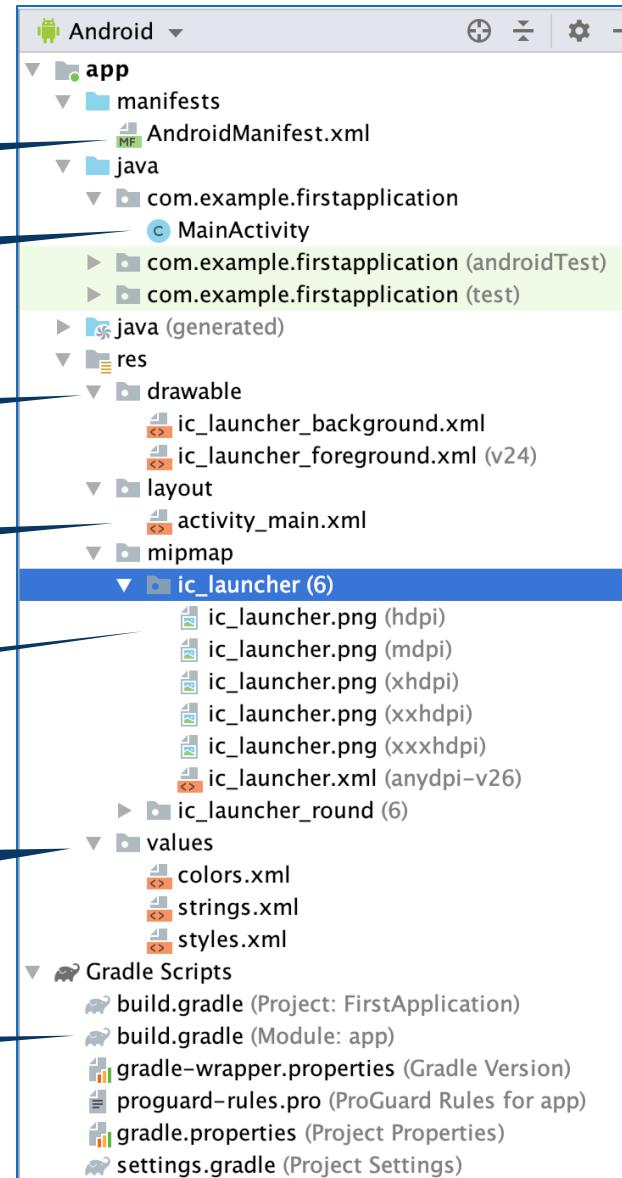
Layout

Hình ảnh

Định nghĩa các giá  
trị như: màu, chuỗi,

...

Cấu hình ứng dụng



# 5. Công cụ Android Studio

## ➤ Cấu trúc Project

- Hàm onCreate và setContentView

The screenshot shows the Android Studio interface. On the left, the Project Navigational Bar displays the project structure:

- app
- manifests
- java
  - com.example.firstapplication
  - MainActivity
- java (generated)
- res
  - drawable
  - layout
    - activity\_main.xml
  - mipmap
  - values
- Gradle Scripts

The Resource Manager panel is visible on the far left.

The main editor window shows the `MainActivity.java` file:

```
1 package com.example.firstapplication;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14
15 }
```

***onCreate*** là hàm mặc định được tạo ra khi ta tạo mới 1 Activity (màn hình) bên trong có ***setContentView*** để nạp XML layout.

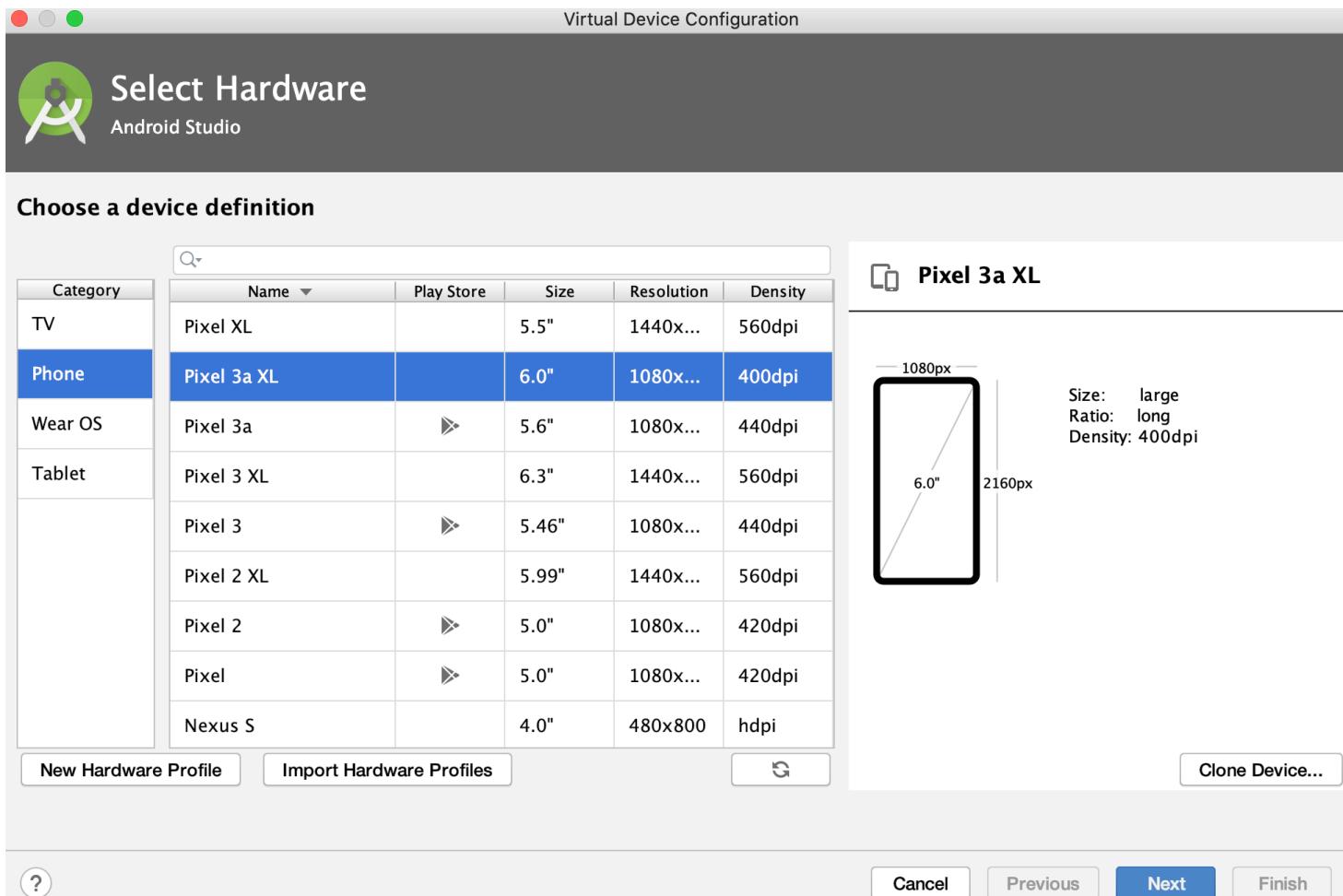
## 5. Công cụ Android Studio

- Thiết lập máy ảo Android Emulator
  - Tạo máy ảo



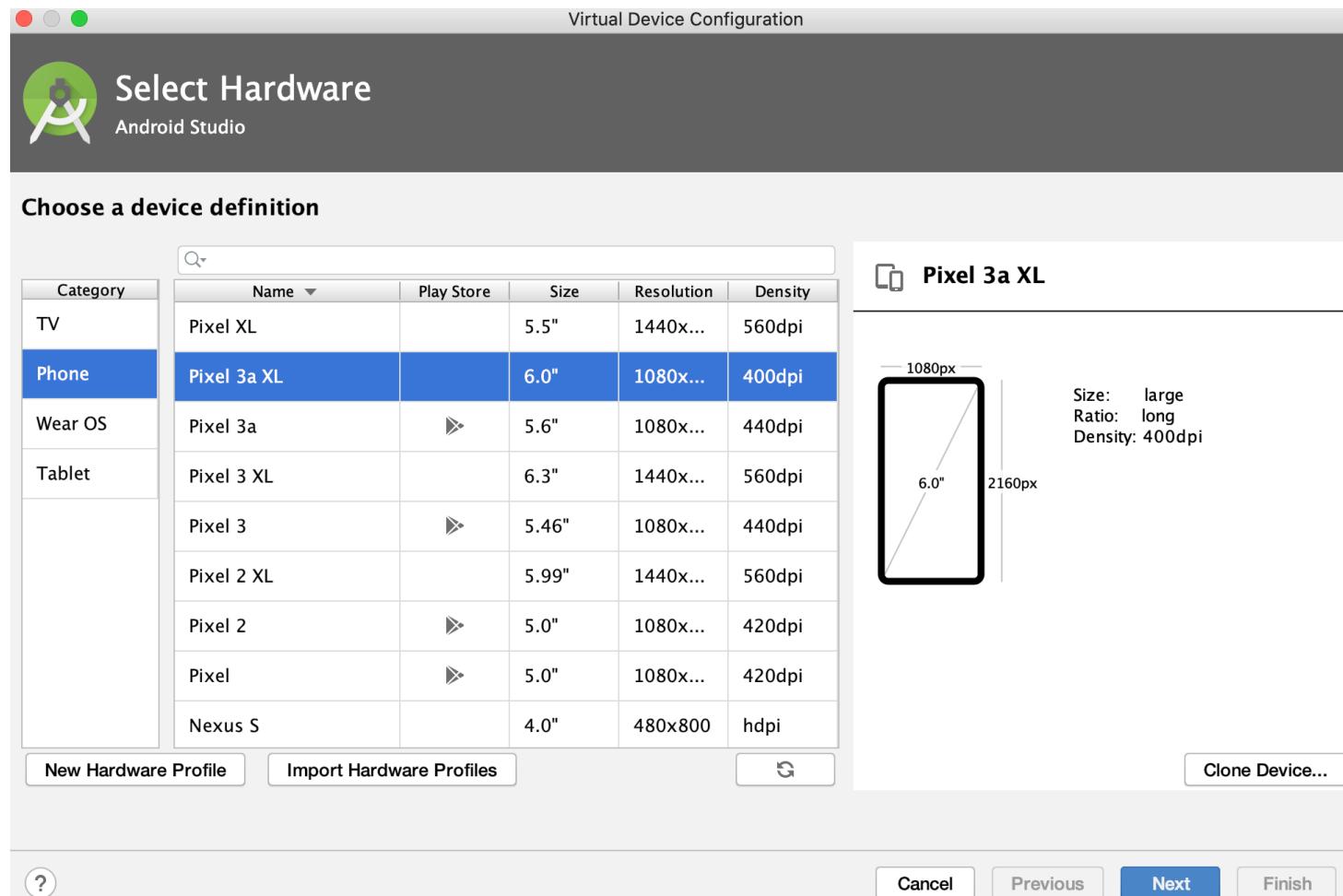
## 5. Công cụ Android Studio

- Thiết lập máy ảo Android Emulator
  - Chọn loại thiết bị



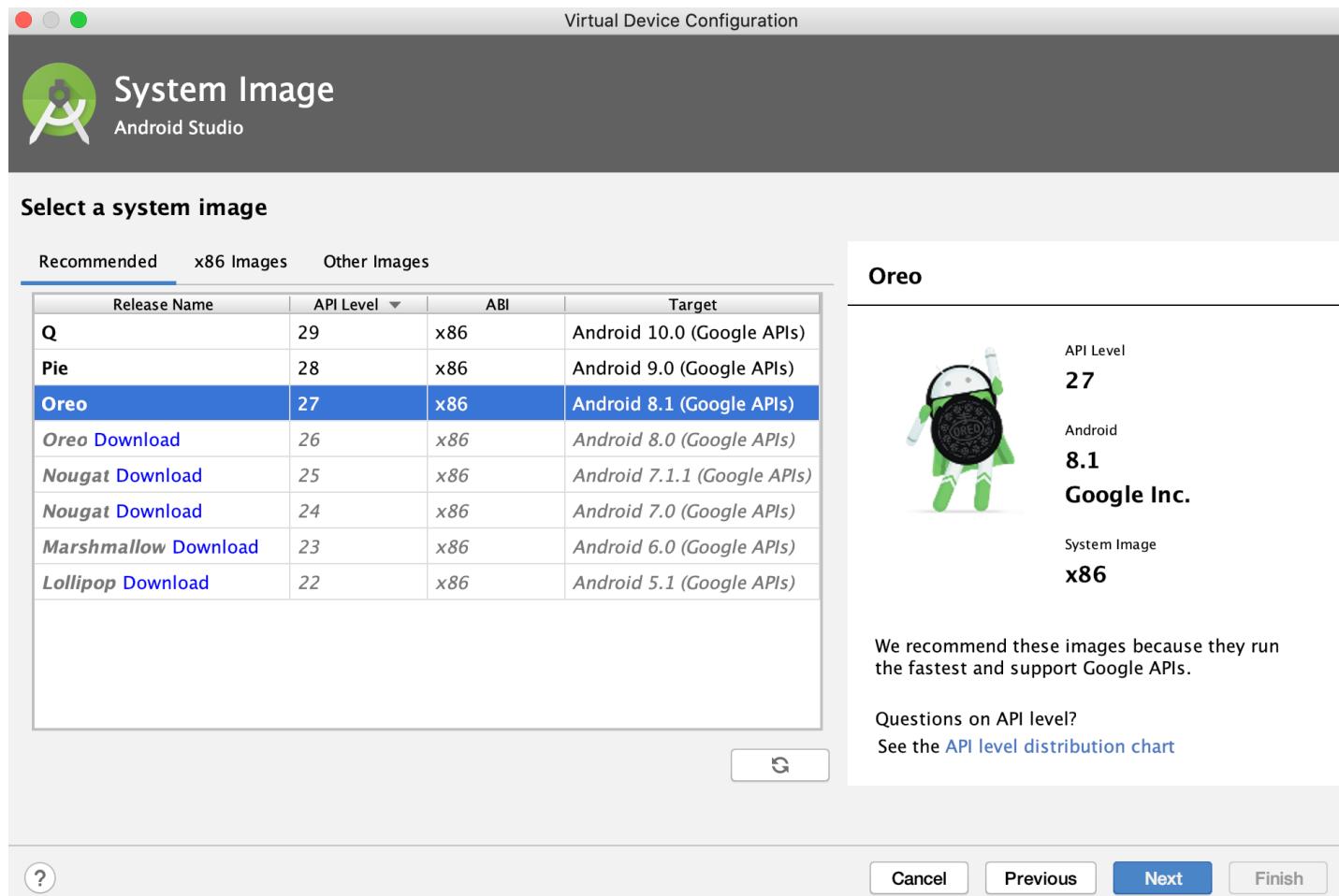
## 5. Công cụ Android Studio

- Thiết lập máy ảo Android Emulator
  - Chọn loại thiết bị



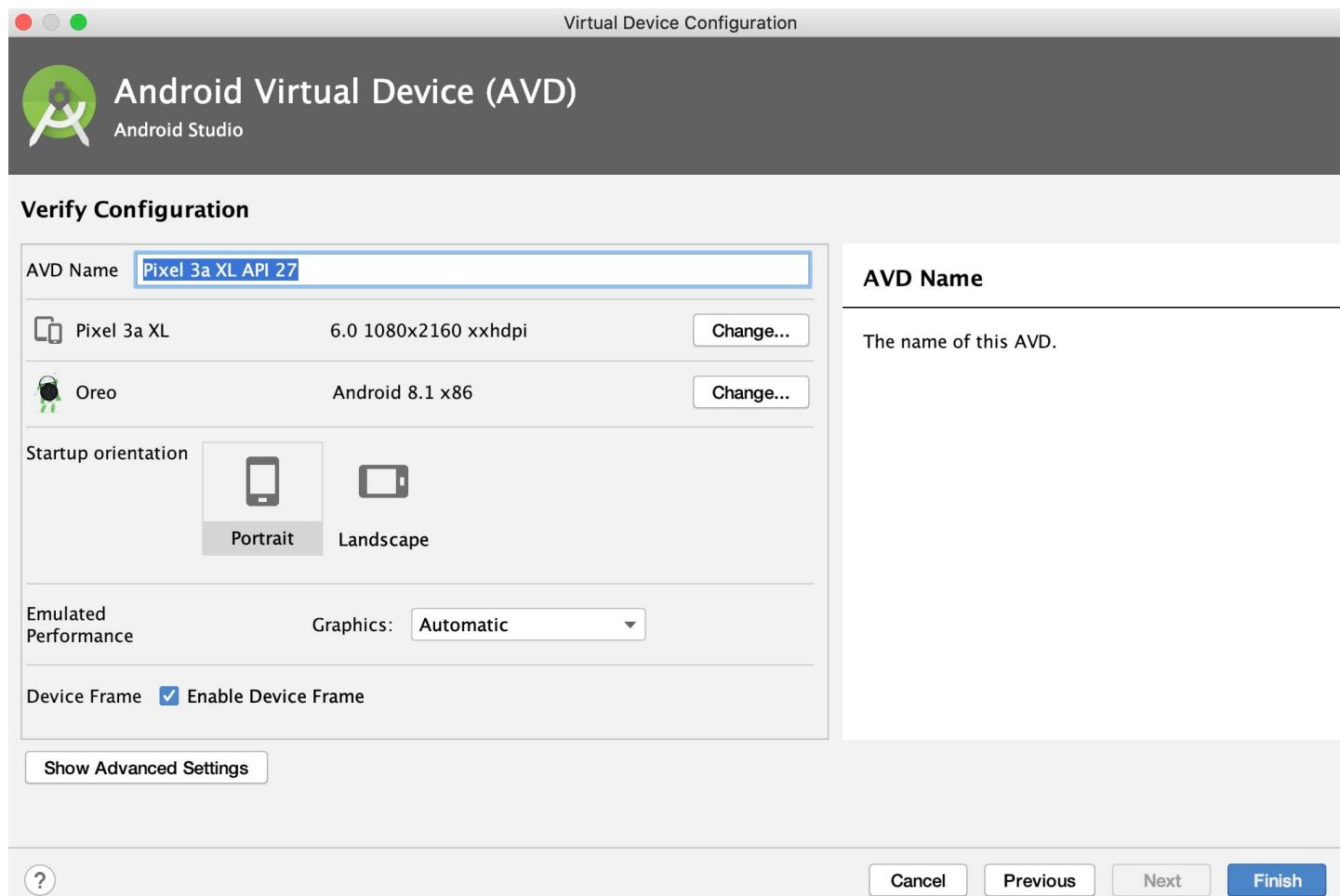
## 5. Công cụ Android Studio

- Thiết lập máy ảo Android Emulator
  - Chọn system image



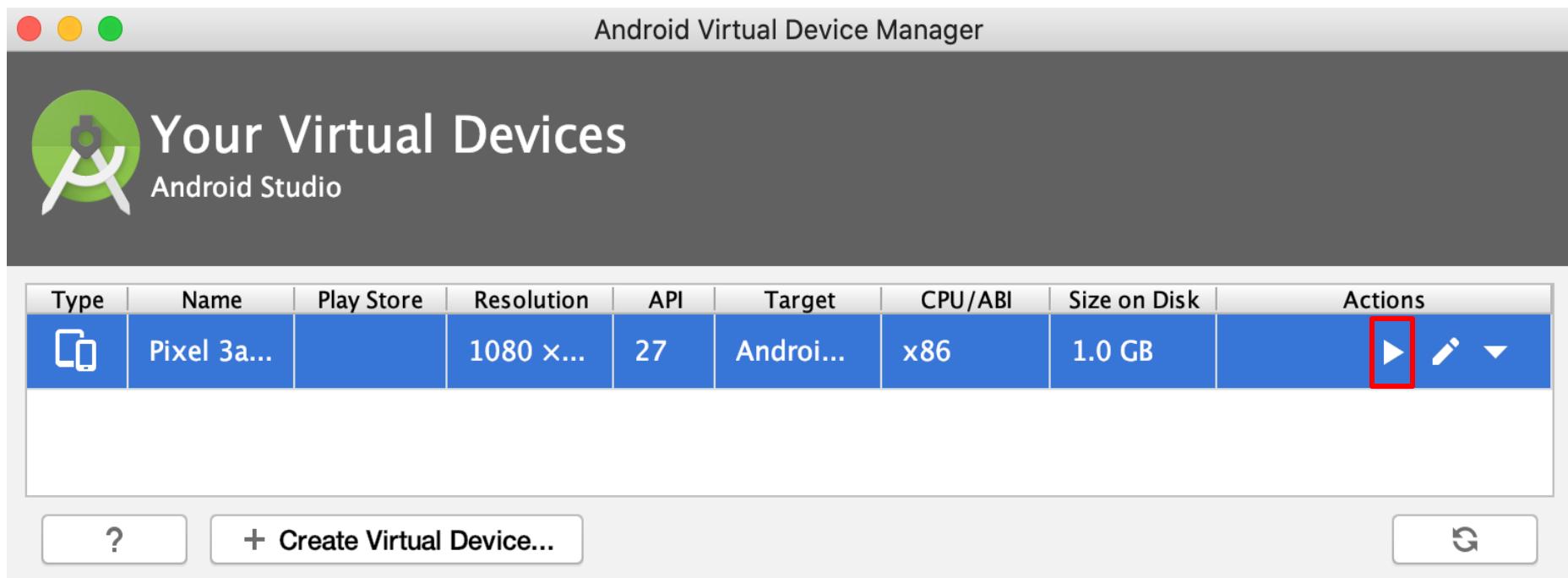
## 5. Công cụ Android Studio

- Thiết lập máy ảo Android Emulator
  - Đặt tên máy ảo



## 5. Công cụ Android Studio

- Thiết lập máy ảo Android Emulator
  - Start máy ảo



## 5. Công cụ Android Studio

- Thiết lập máy ảo Android Emulator
  - Start máy ảo



## 5. Công cụ Android Studio

- Thiết lập máy ảo Genymotion thay cho Android Emulator
  - Tạo tài khoản

The screenshot shows a web browser window titled "Account Creation - Genymotion" with the URL "genymotion.com/account/create/". The page is part of the GENYMOTION website. At the top, there are three circular steps: "1 Register", "2 Activate", and "3 Enjoy". Below this, the word "Register" is highlighted in pink. The form fields are arranged in two columns. The left column contains "Username\*" (with a placeholder "Username") and "Email address\*". The right column contains "Usage type\*" (with a dropdown menu showing "Select") and "Company type\*" (with a dropdown menu showing "Select"). Below the form, there is a section for "Country" with a dropdown menu showing "Vietnam". At the bottom, there are three checkboxes:

- I accept the [privacy policy](#) and acknowledge that my data will be collected and processed in compliance with it.
- I want the latest news and updates
- I accept the [Terms and Conditions](#).

## 5. Công cụ Android Studio

- Thiết lập máy ảo Genymotion thay cho Android Emulator
  - Tải Genymotion



For personal use only

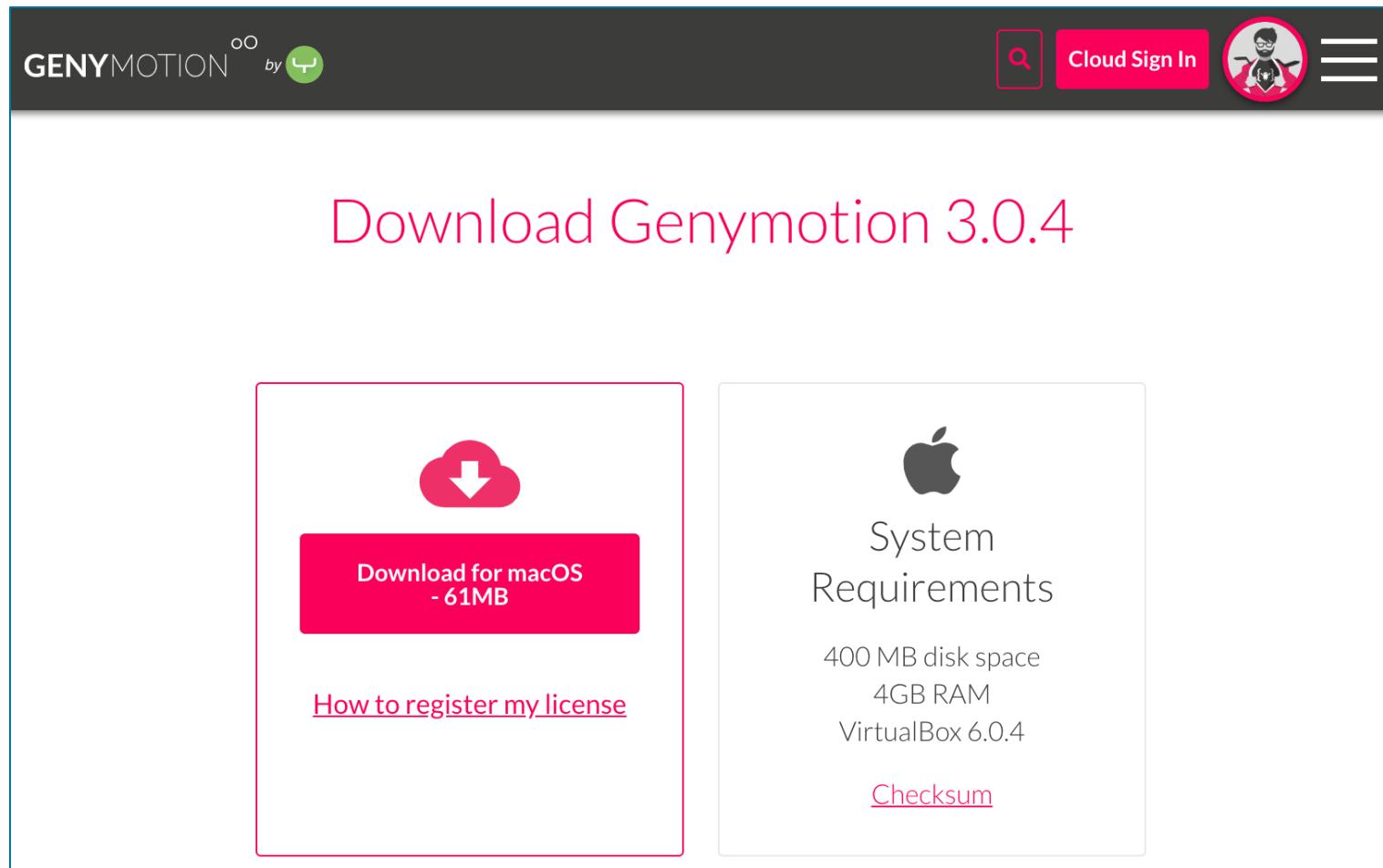
You want to enjoy Genymotion for personal Android app emulation or  
to play your favorite Android games on your computer?

[Download Genymotion  
Personal Edition](#)



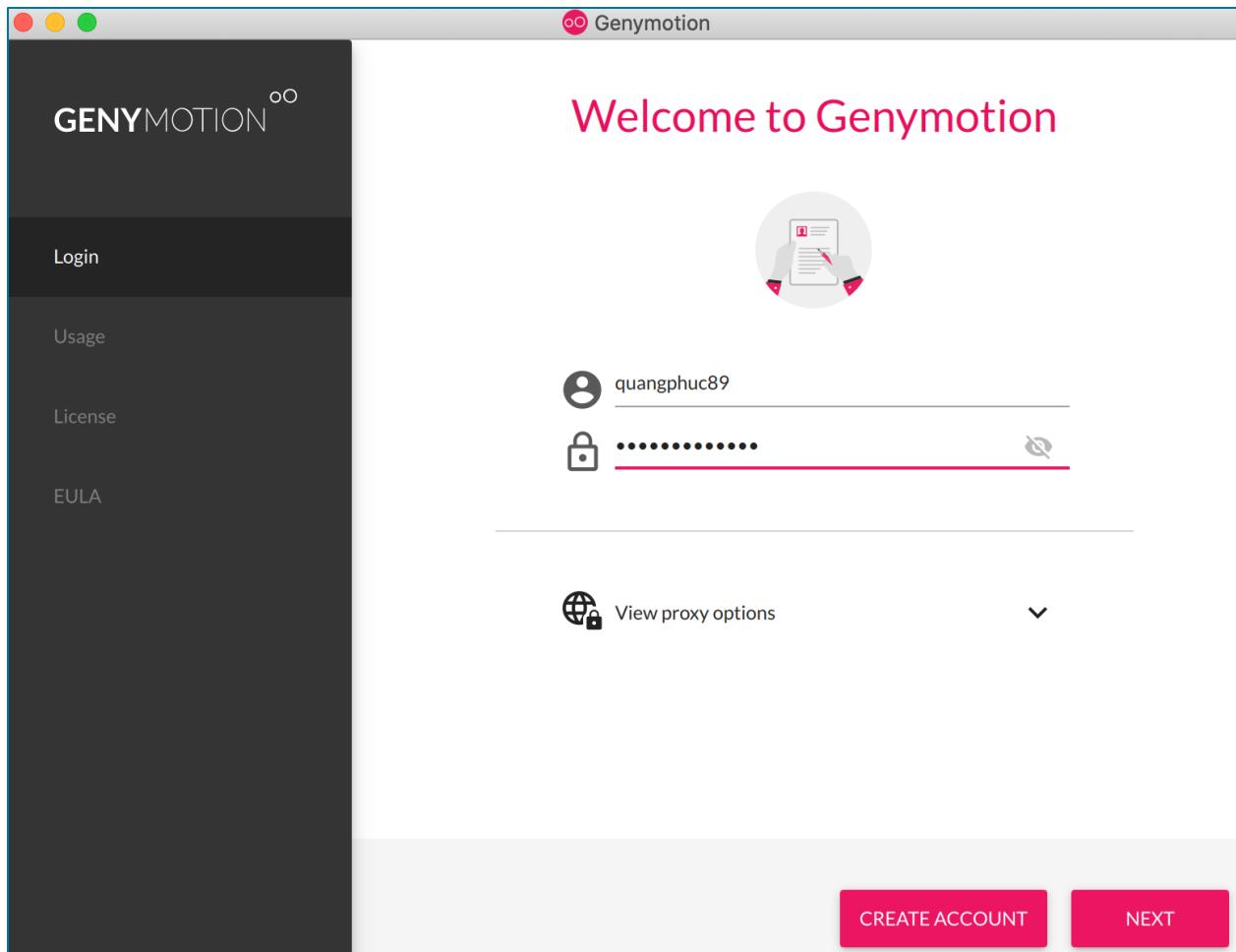
## 5. Công cụ Android Studio

- Thiết lập máy ảo Genymotion thay cho Android Emulator
  - Tải Genymotion



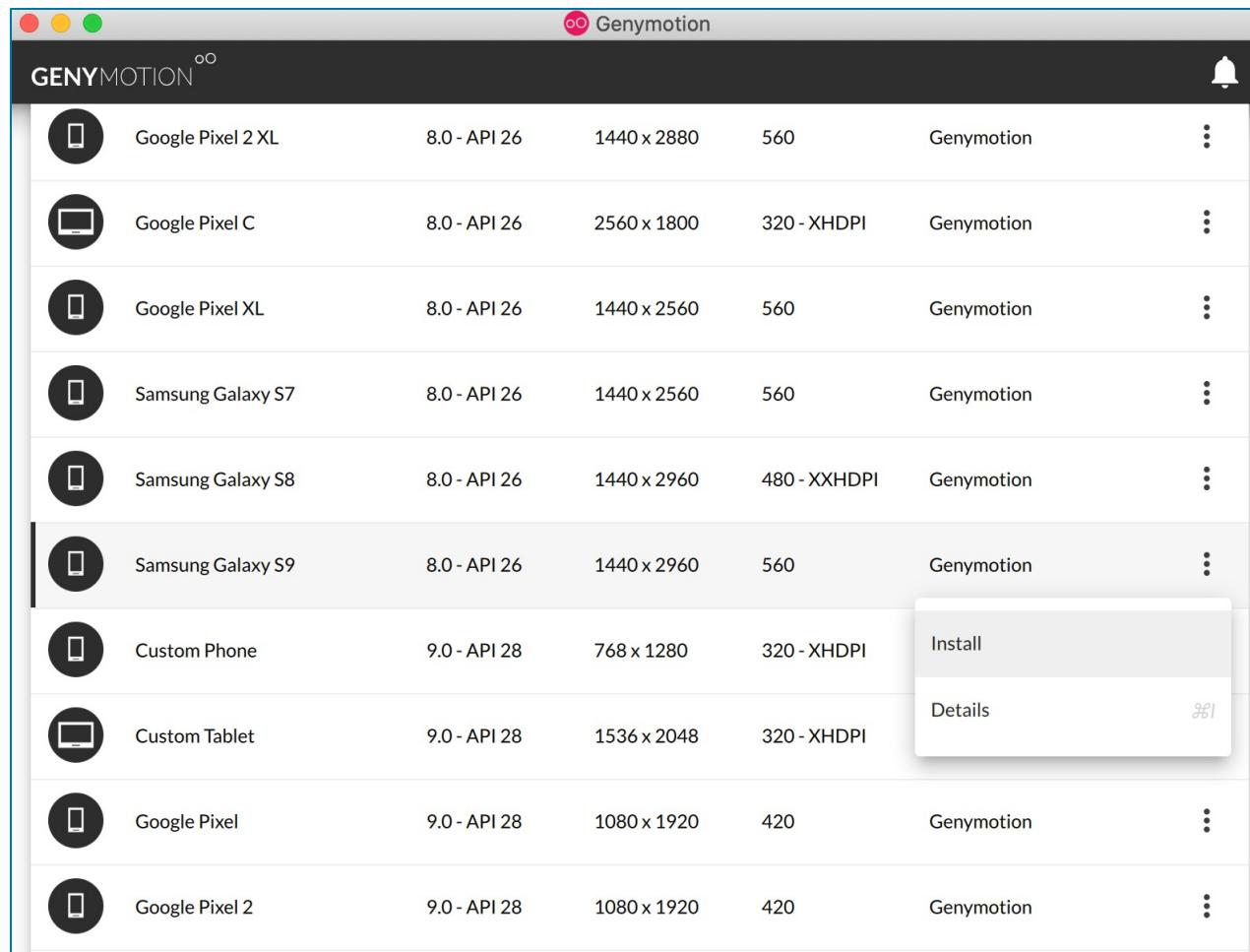
## 5. Công cụ Android Studio

- Thiết lập máy ảo Genymotion thay cho Android Emulator
  - Tạo máy ảo



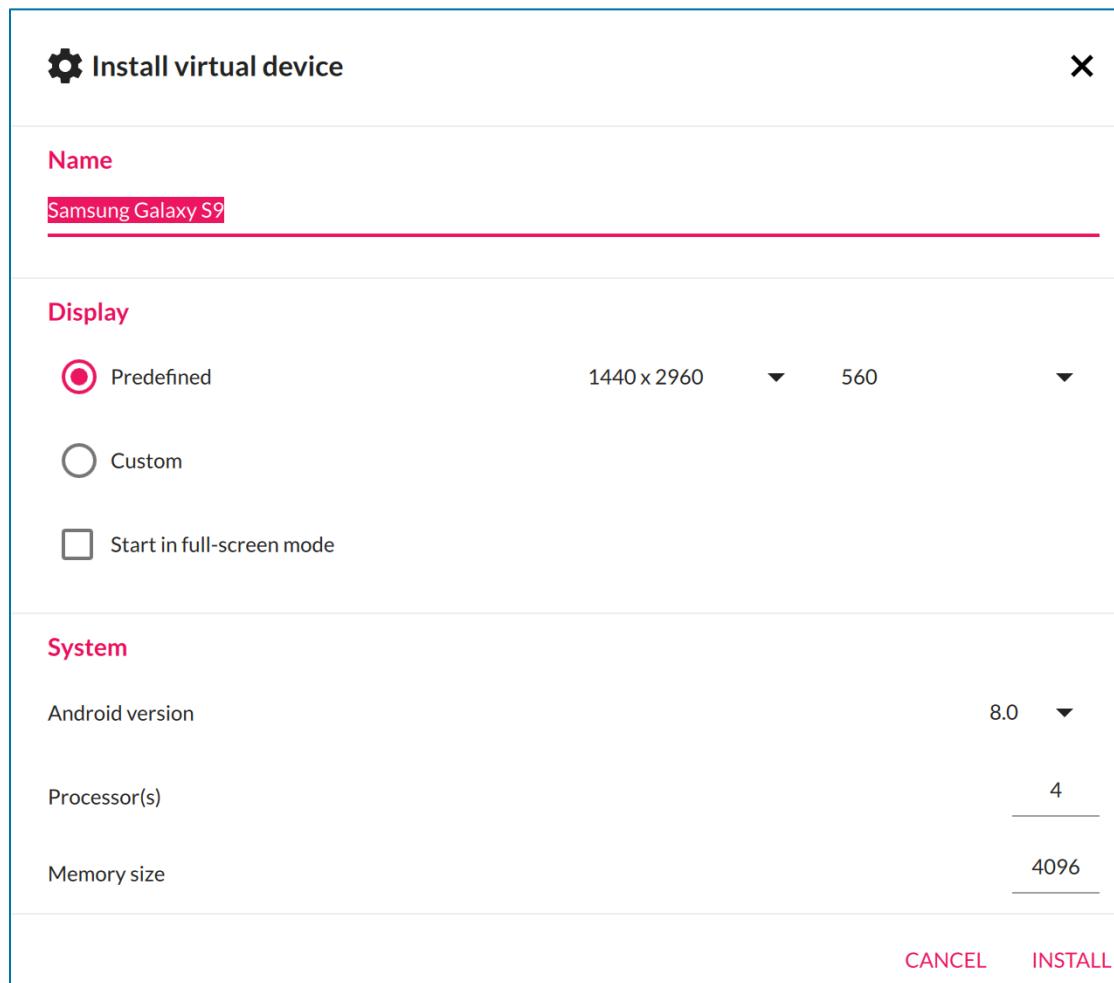
## 5. Công cụ Android Studio

- Thiết lập máy ảo Genymotion thay cho Android Emulator
  - Tạo máy ảo



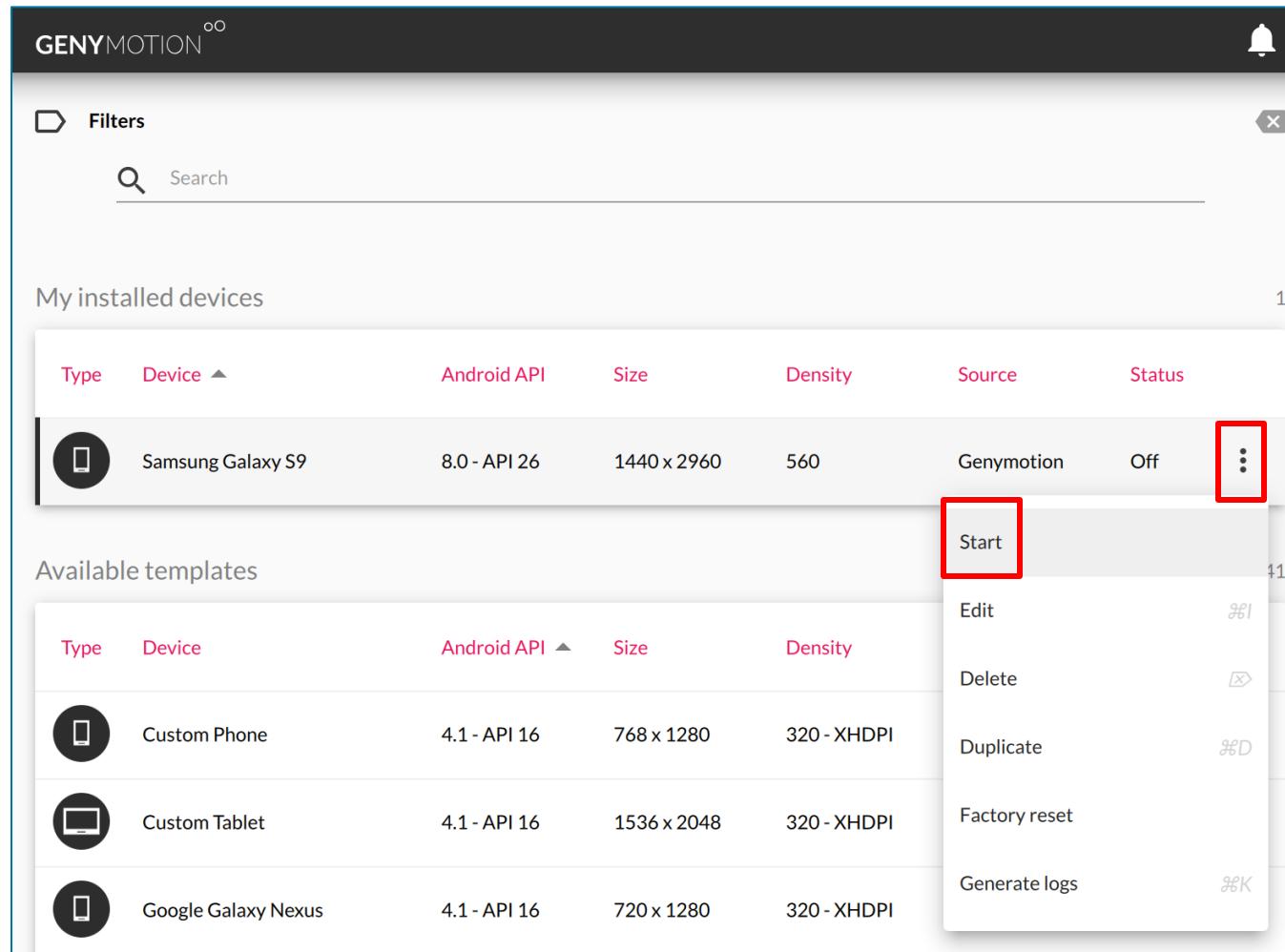
## 5. Công cụ Android Studio

- Thiết lập máy ảo Genymotion thay cho Android Emulator
  - Tạo máy ảo



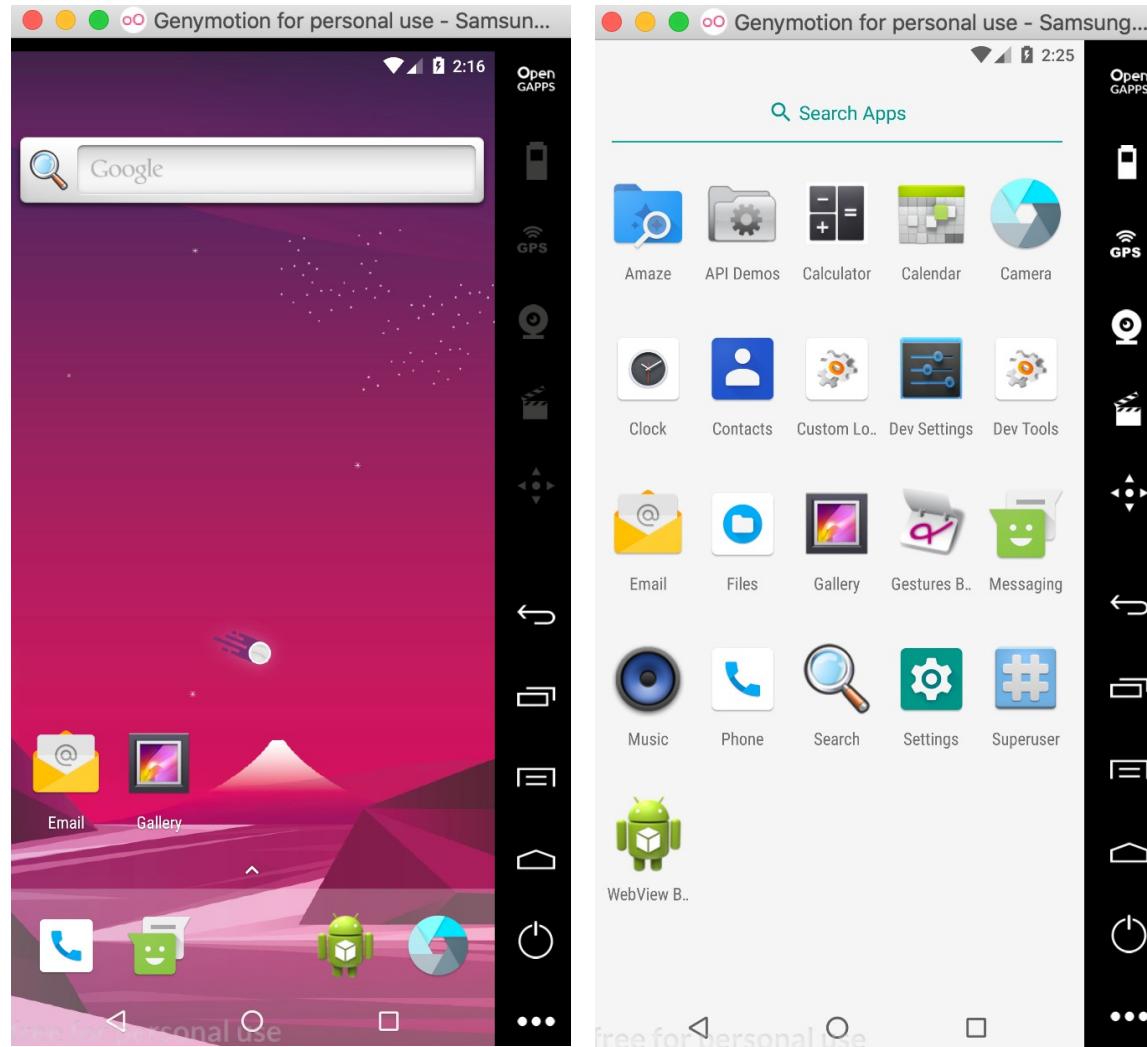
## 5. Công cụ Android Studio

- Thiết lập máy ảo Genymotion thay cho Android Emulator
  - Start máy ảo



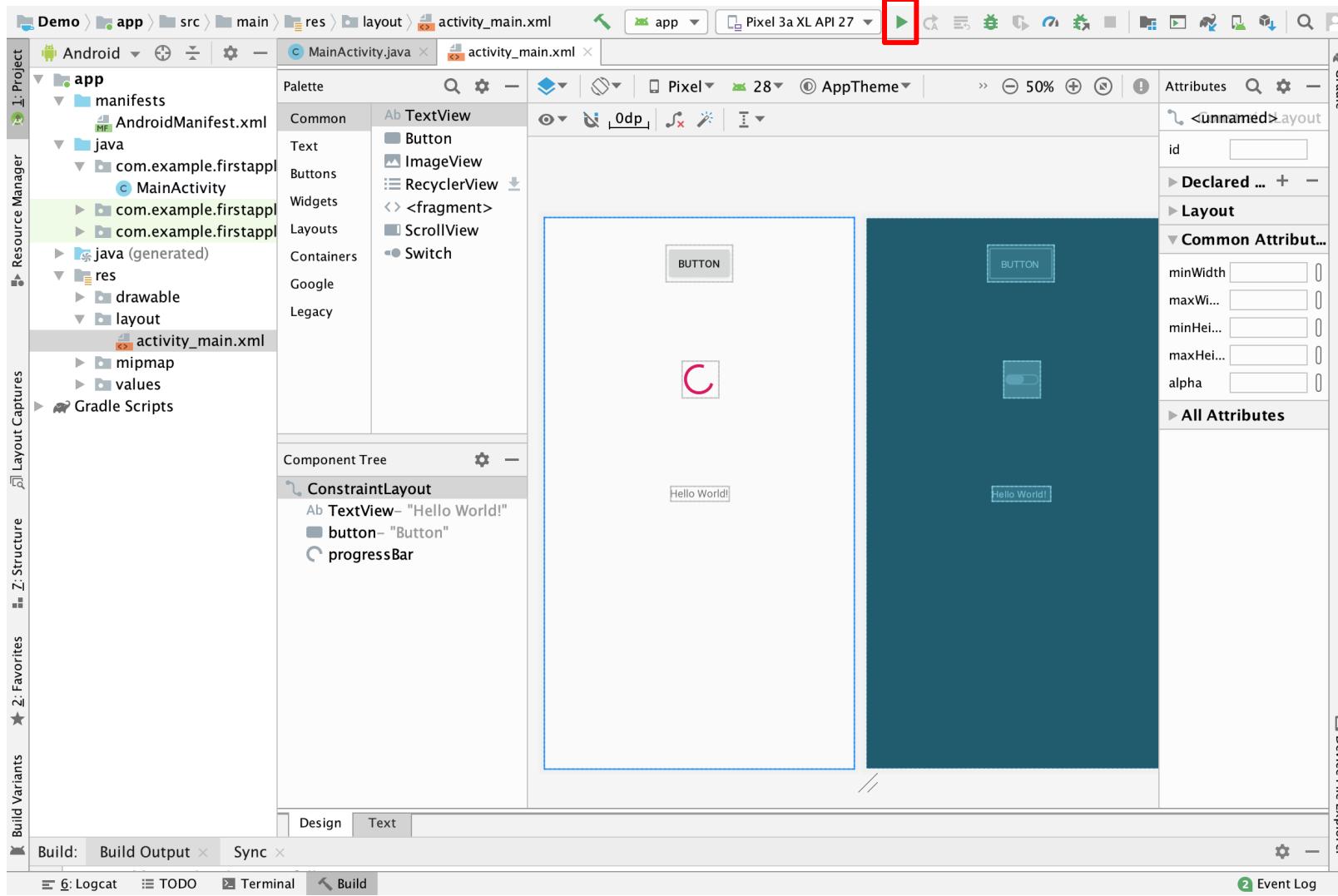
## 5. Công cụ Android Studio

- Thiết lập máy ảo Genymotion thay cho Android Emulator
  - Start máy ảo



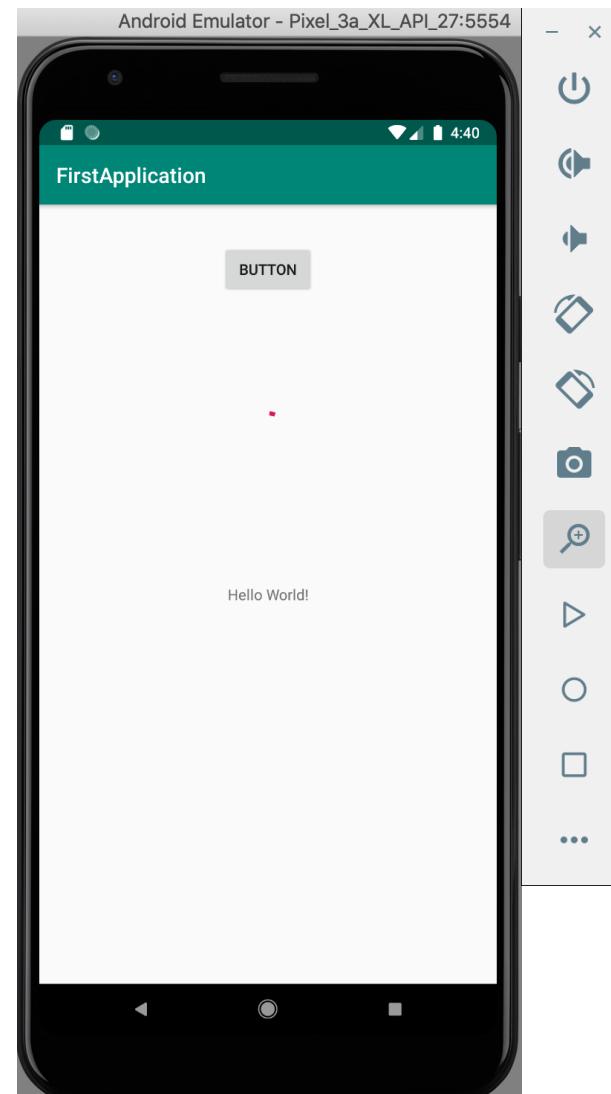
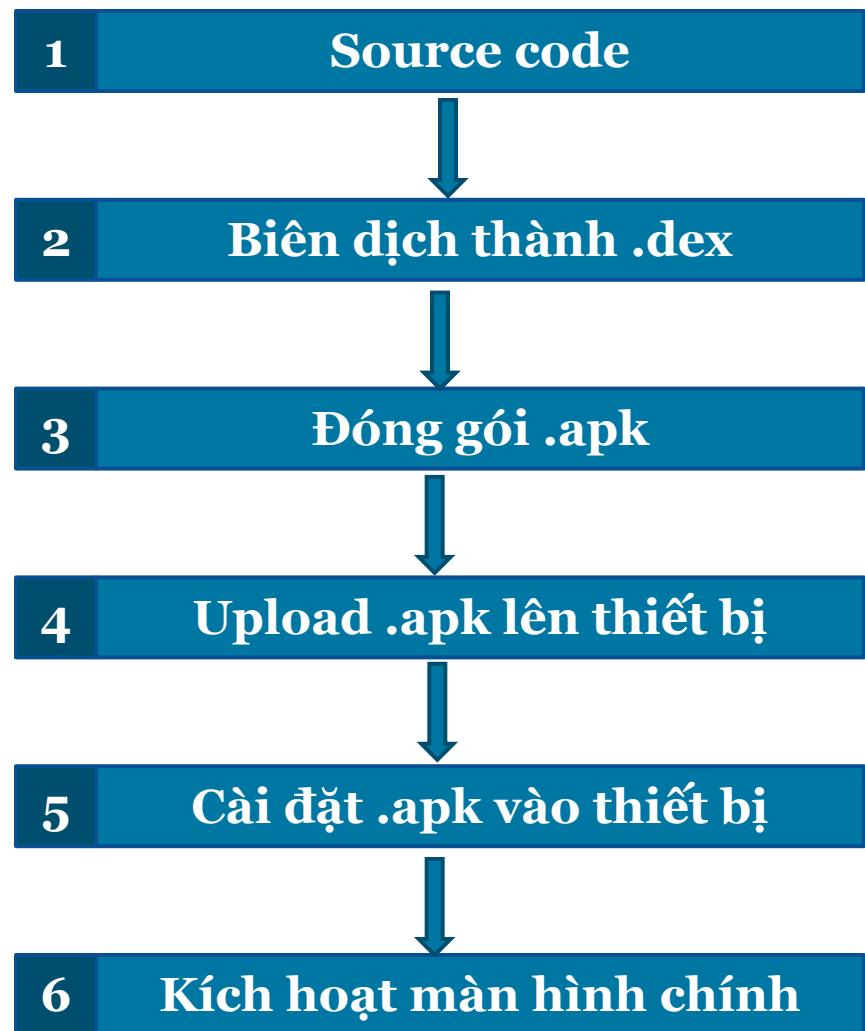
# 6. Thực thi ứng dụng Android

- Chạy ứng dụng trên máy ảo



## 6. Thực thi ứng dụng Android

- Chạy ứng dụng trên máy ảo

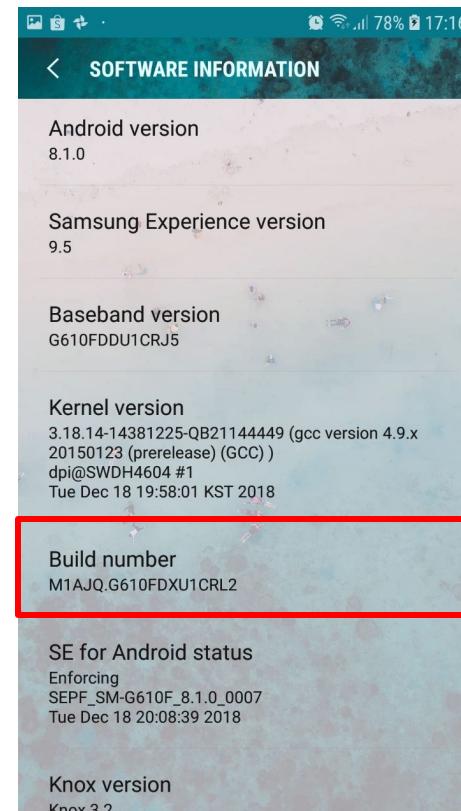


## 6. Thực thi ứng dụng Android

- Chạy ứng dụng trên máy thật

- Kết nối cable

*Settings -> About phone -> Software information -> Build number (chạm nhiều lần) ➔ Enable Developer options*

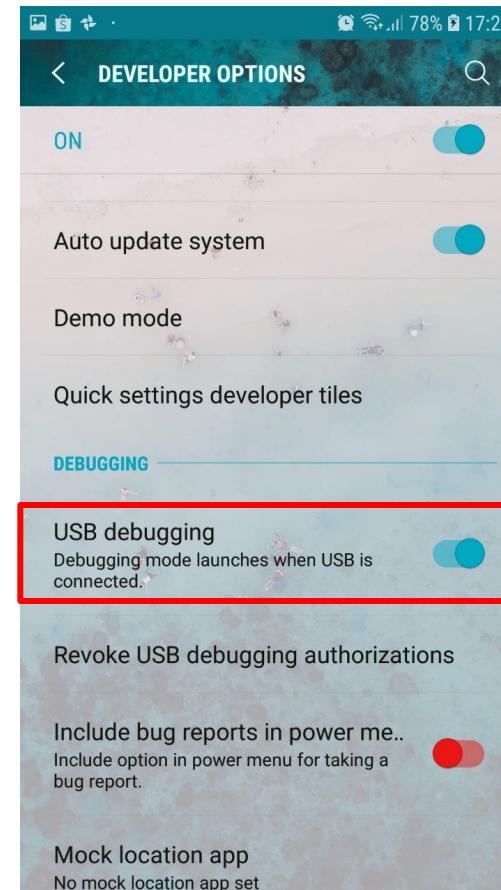
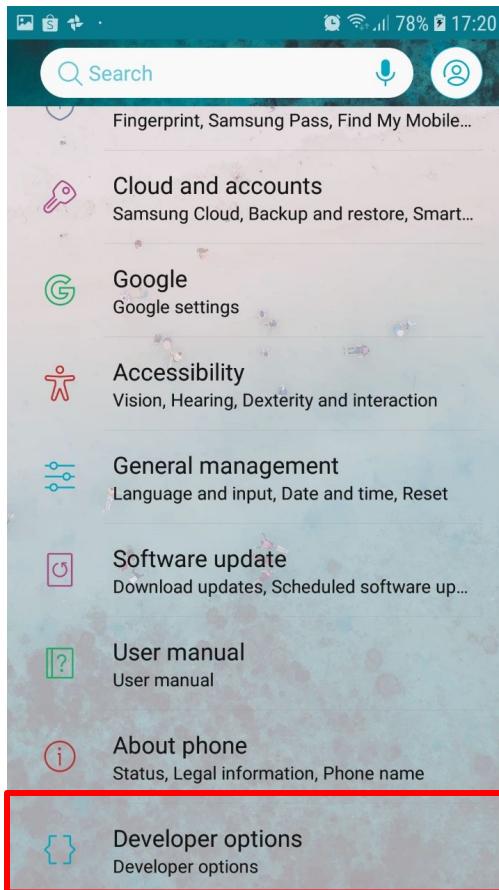


## 6. Thực thi ứng dụng Android

- Chạy ứng dụng trên máy thật

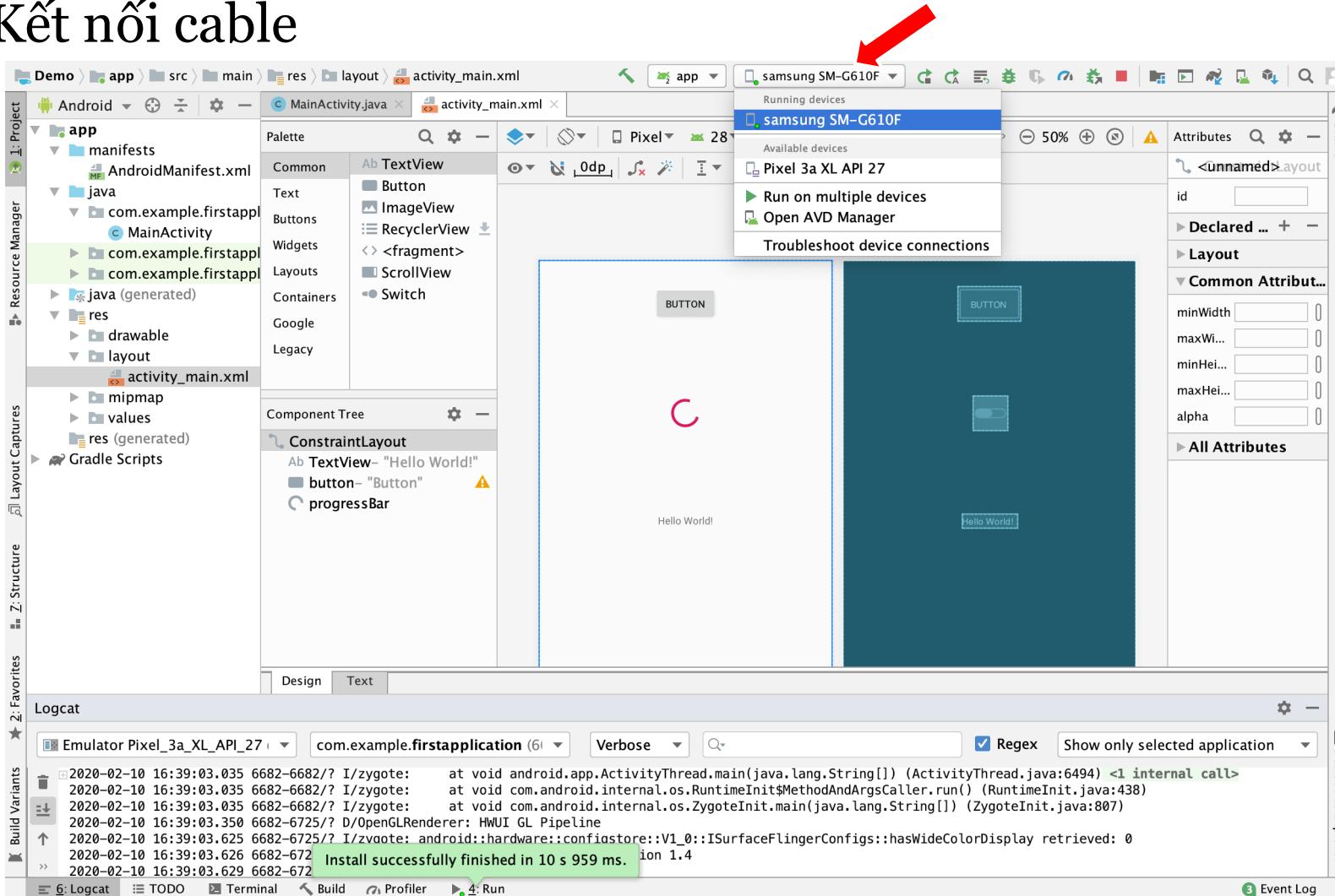
- Kết nối cable

*Settings -> Developer options -> Enable USB debugging*



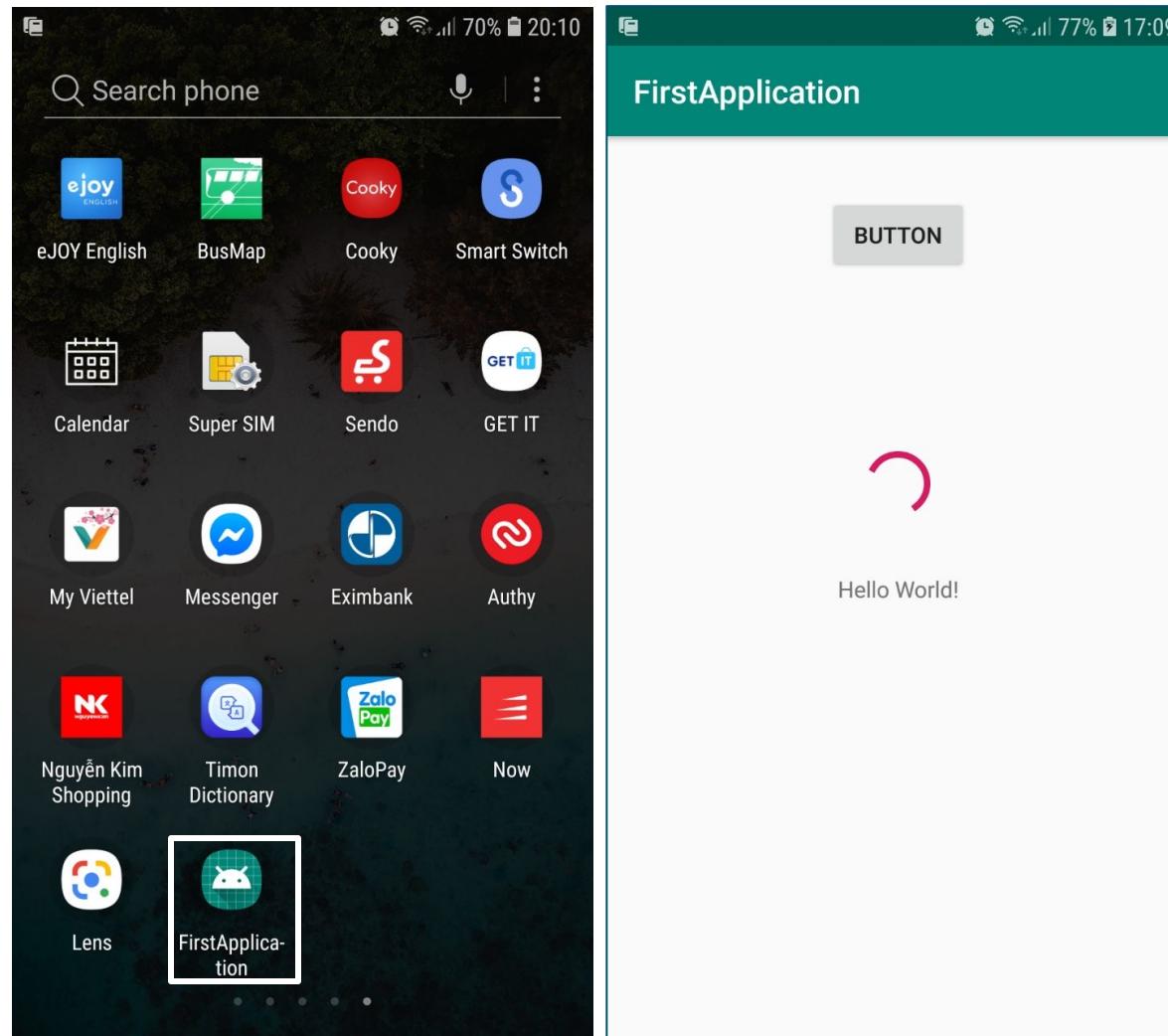
# 6. Thực thi ứng dụng Android

- Chạy ứng dụng trên máy thật
  - Kết nối cable



## 6. Thực thi ứng dụng Android

- Chạy ứng dụng trên máy thật
  - Kết nối cable



## 6. Thực thi ứng dụng Android

- Chạy ứng dụng trên máy thật
  - Upload apk lên **diawi**

The screenshot shows a web browser window with the URL [diawi.com](https://diawi.com). The main content is titled "Upload your app". It features a large rectangular area with rounded corners for dragging and dropping files, with the placeholder text "Drag&drop files here" and ".ipa or .zip(.app) .apk". Below this area is a button labeled "+ Add" with a red arrow pointing to it. At the bottom of this section are two checkboxes: "Allow your testers to find the app on Diawi's mobile web app using their UDID (iOS only)." and "Allow Diawi to display the app's icon on the wall of apps". To the right of these checkboxes is a link "More options...". At the very bottom is a large blue "Send" button.

**Stats**

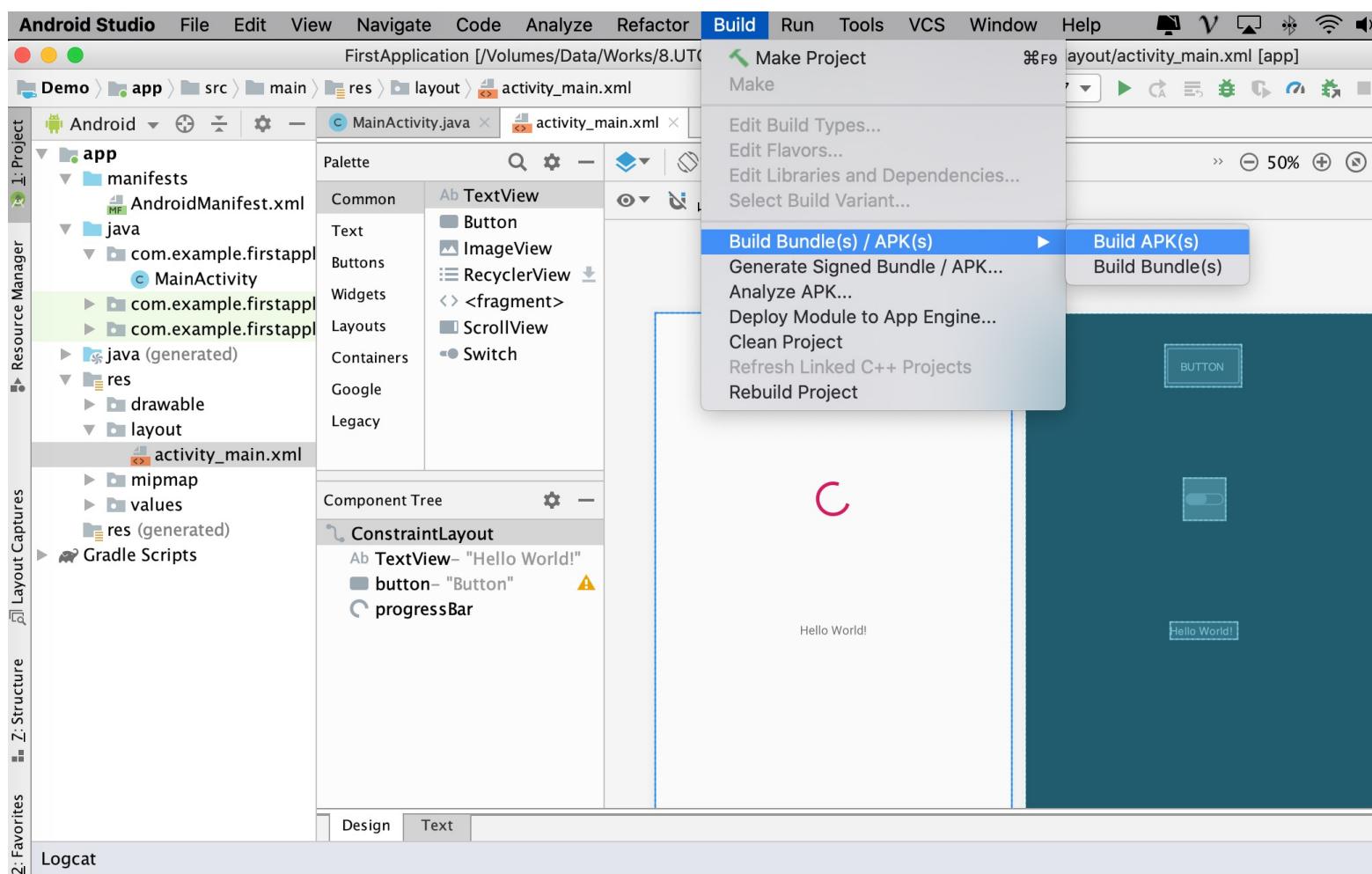
Android installations this week

Smartphone  
Phablet  
Tablet  
Others

9.x  
8.x  
7.x  
10.x  
6.x  
5.x  
4.4.x  
Others

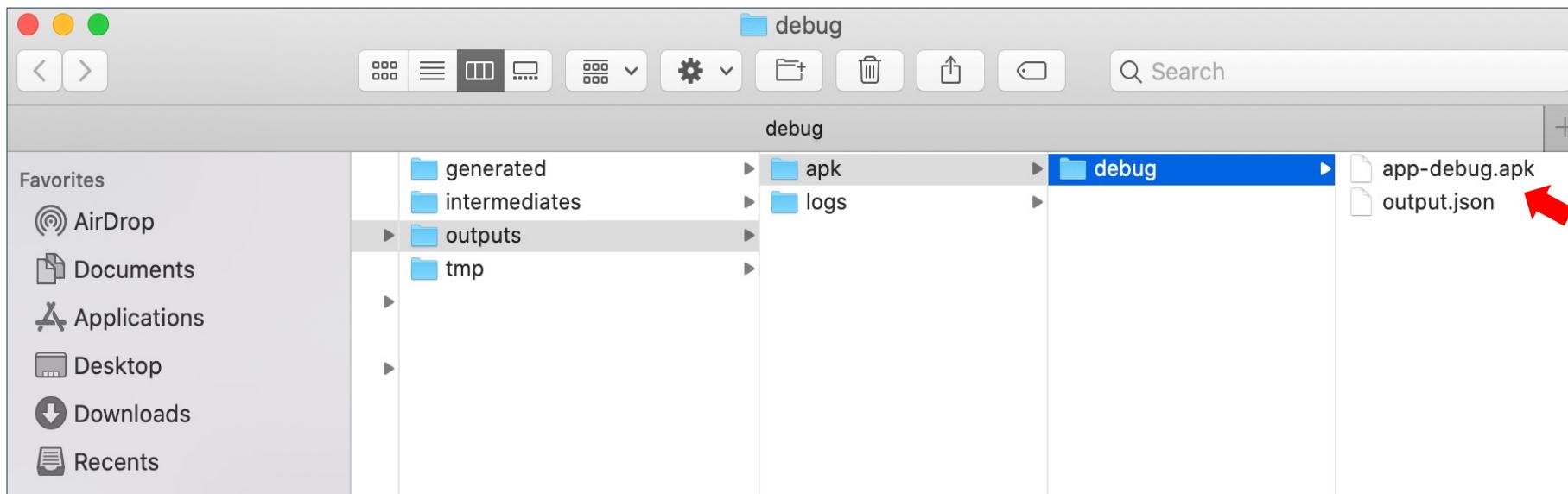
# 6. Thực thi ứng dụng Android

- Chạy ứng dụng trên máy thật
  - Upload apk lên **diawi**



## 6. Thực thi ứng dụng Android

- Chạy ứng dụng trên máy thật
  - Upload apk lên **diawi**



## 6. Thực thi ứng dụng Android

- Chạy ứng dụng trên máy thật
  - Upload apk lên **diawi**

The screenshot shows the Diawi web interface for uploading an APK file. The main area displays the QR code for the app 'Vmzn2n' with the URL <https://i.diawi.com/Vmzn2n>. Below the QR code is a link to copy the URL. To the left, it says 'Your app is ready'. To the right, there are two pie charts under the heading 'Stats' showing 'Android installations this week'. The first chart shows device types: Smartphone (blue), Phablet (green), Tablet (orange), and Others (yellow). The second chart shows Android versions: 9.x (blue), 8.x (green), 7.x (orange), 10.x (yellow), 6.x (purple), 5.x (red), 4.4.x (dark blue), and Others (light green).

Upload your app

Your app is ready

Stats

Android installations this week

Vmzn2n

<https://i.diawi.com/Vmzn2n>

Copy this link and send it to anyone.

9.x  
8.x  
7.x  
10.x  
6.x  
5.x  
4.4.x  
Others

9.x  
8.x  
7.x  
10.x  
6.x  
5.x  
4.4.x  
Others

➔ Dùng thiết bị thật vào link trực tiếp hoặc scan mã QR để cài đặt ứng dụng.

02

LAYOUT & VIEW (P1)

# NỘI DUNG

## 1. Layout

- *Frame Layout*
- *Linear Layout*
- *Table Layout*
- *Relative Layout*
- *Constraint Layout*

## 2. View cơ bản

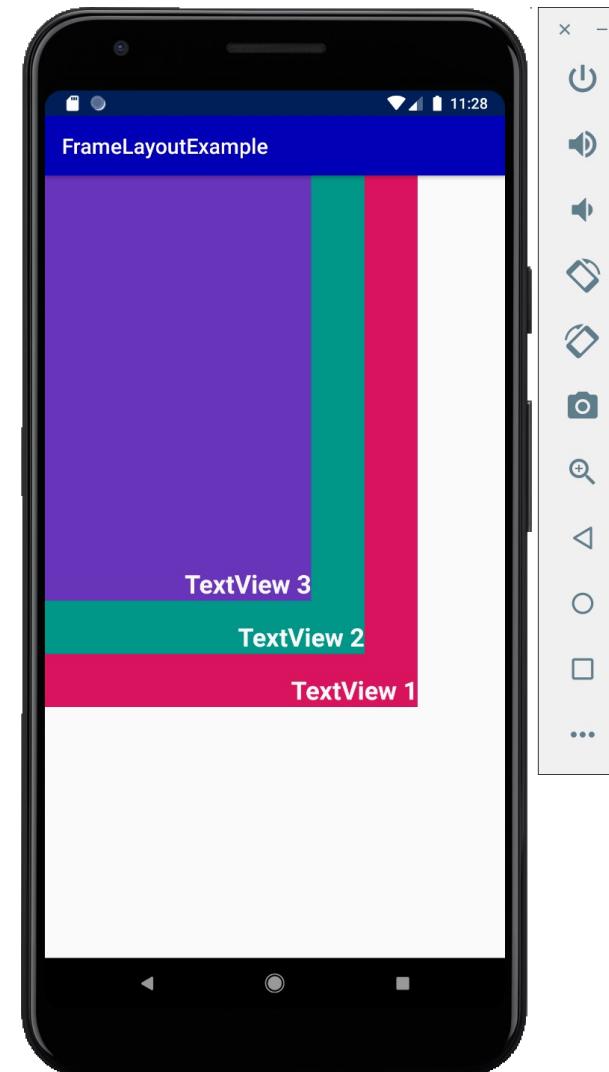
- *TextView*
- *EditText*
- *Button*
- *CheckBox, RadioButton*
- *ImageButton*
- *ImageView*

## 3. Bài tập

# 1. Thiết kế giao diện với Layout

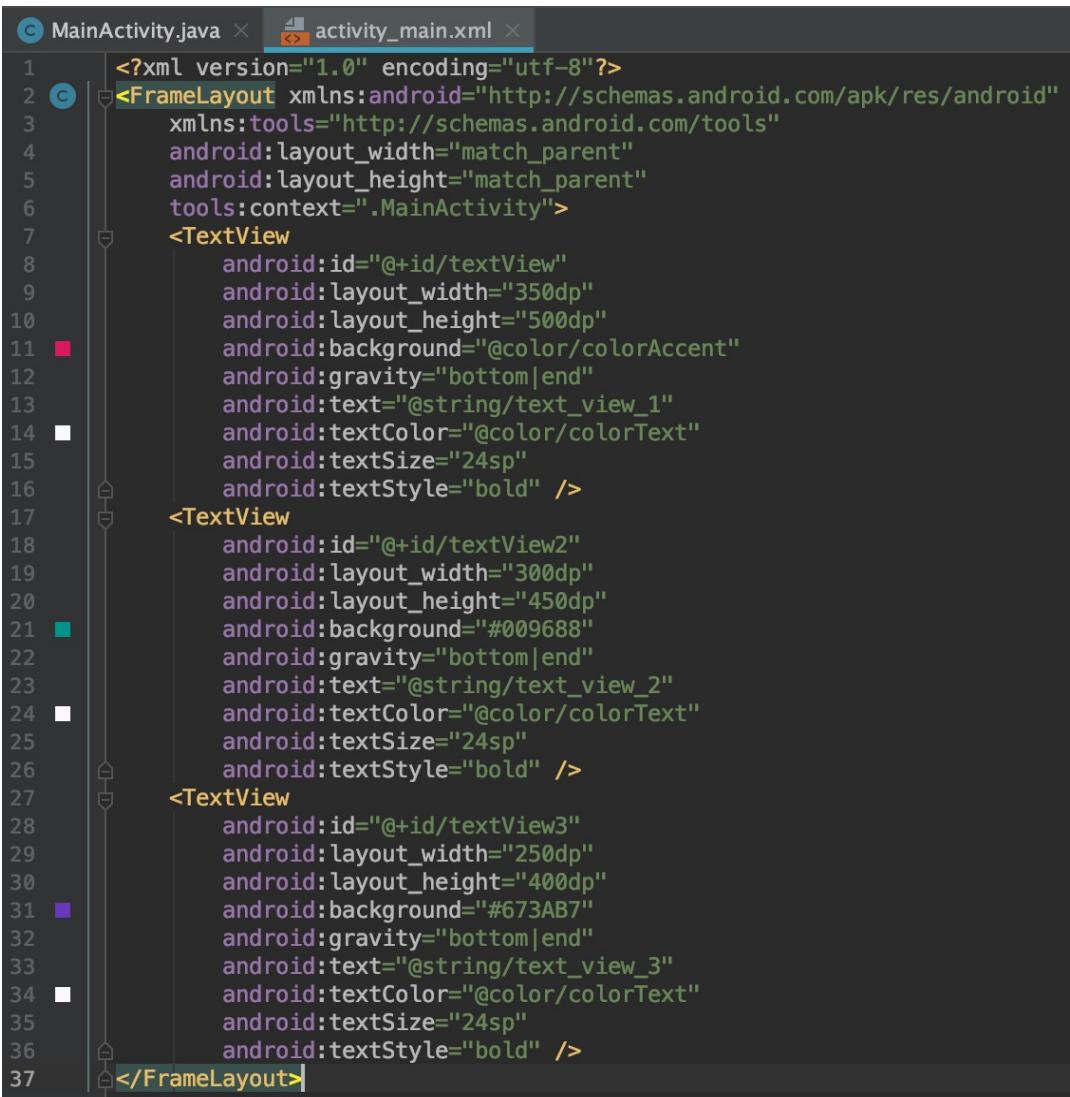
## ➤ Frame Layout

- ✓ Thường được sử dụng để xây dựng bố cục tổ chức hiển thị một đối tượng duy nhất.
- ✓ Các view được thêm vào FrameLayout sẽ mặc định được neo ở vị trí Top-Left, view thêm sau sẽ chồng lên view đã thêm trước đó.

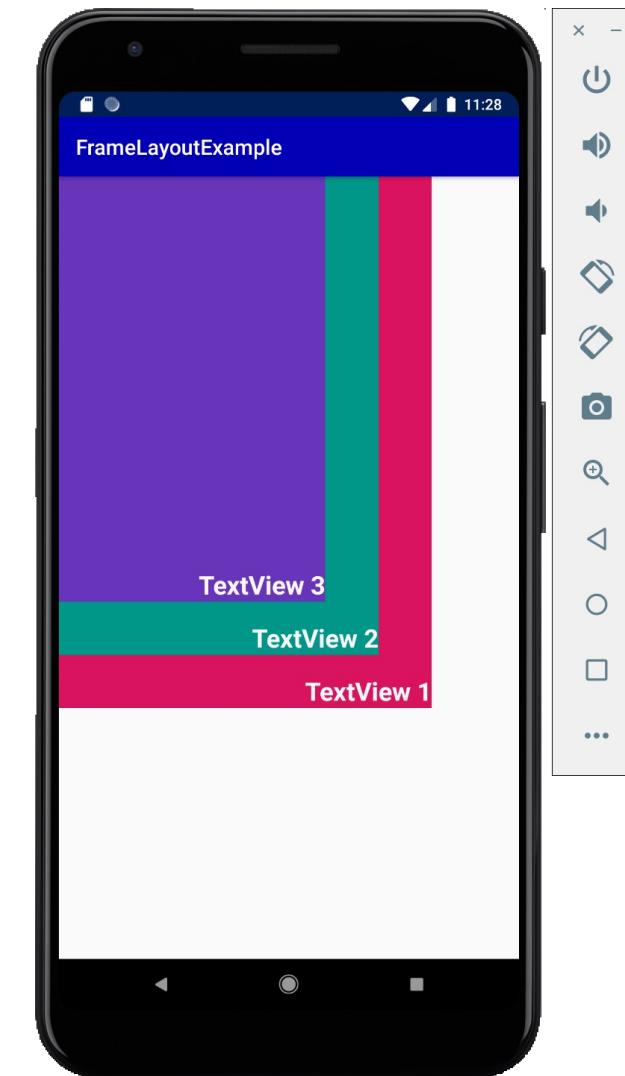


# 1. Thiết kế giao diện với Layout

## ➤ Frame Layout



```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="350dp"
        android:layout_height="500dp"
        android:background="@color/colorAccent"
        android:gravity="bottom|end"
        android:text="@string/text_view_1"
        android:textColor="@color/colorText"
        android:textSize="24sp"
        android:textStyle="bold" />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="300dp"
        android:layout_height="450dp"
        android:background="#009688"
        android:gravity="bottom|end"
        android:text="@string/text_view_2"
        android:textColor="@color/colorText"
        android:textSize="24sp"
        android:textStyle="bold" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="250dp"
        android:layout_height="400dp"
        android:background="#673AB7"
        android:gravity="bottom|end"
        android:text="@string/text_view_3"
        android:textColor="@color/colorText"
        android:textSize="24sp"
        android:textStyle="bold" />
</FrameLayout>
```



# 1. Thiết kế giao diện với Layout

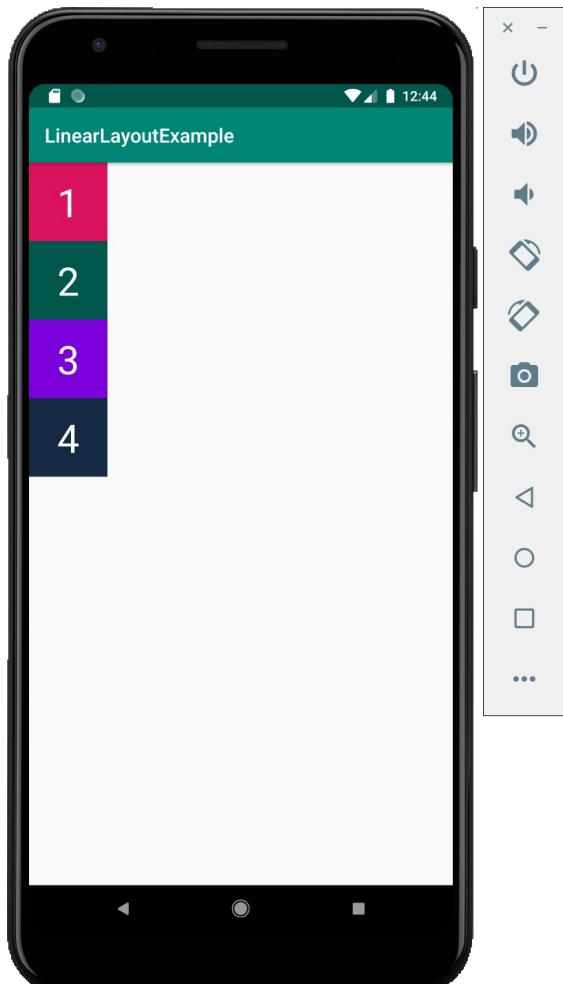
## ➤ Linear Layout

- ✓ Thường được sử dụng để xây dựng bố cục tổ chức hiển thị các đối tượng theo một chiều duy nhất (dọc hoặc ngang).
- ✓ Các view sẽ được bố trí theo dạng khối và không đè lên nhau, có thể sử dụng thuộc tính Gravity thiết lập lại vị trí.

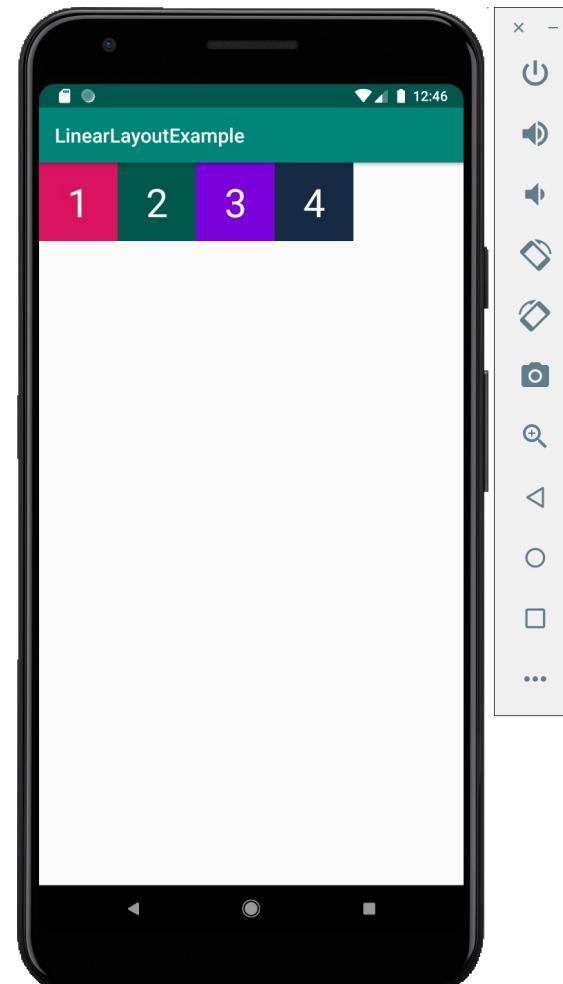
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".MainActivity"  
    android:orientation="vertical">  
  
</LinearLayout>
```

# 1. Thiết kế giao diện với Layout

## ➤ Linear Layout



Vertical



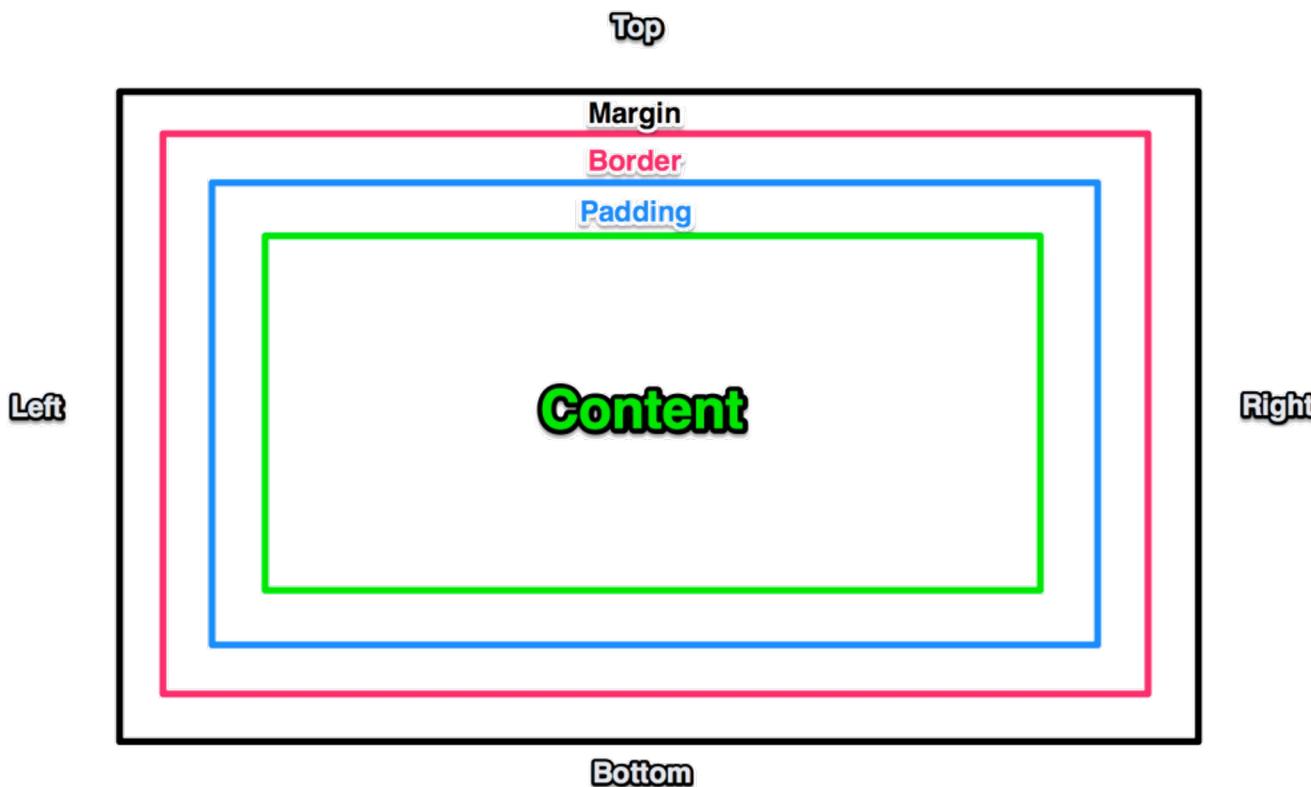
Horizontal

# 1. Thiết kế giao diện với Layout

## ➤ Linear Layout

- ✓ Một số thuộc tính hỗ trợ thiết kế: margin, padding, gravity, weight,

...



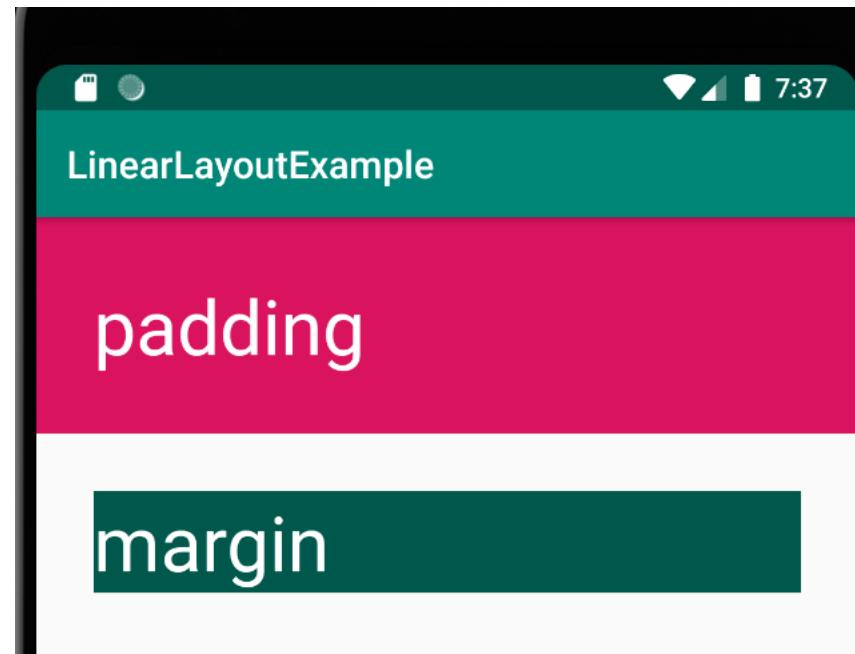
# 1. Thiết kế giao diện với Layout

## ➤ Linear Layout

✓ *Thuộc tính hỗ trợ thiết kế: margin, padding*

➔ Padding (internal spacing – khoảng cách giữa nội dung bên trong so với đường viền của control)

➔ Margin (external spacing – khoảng cách giữa control này với control khác)



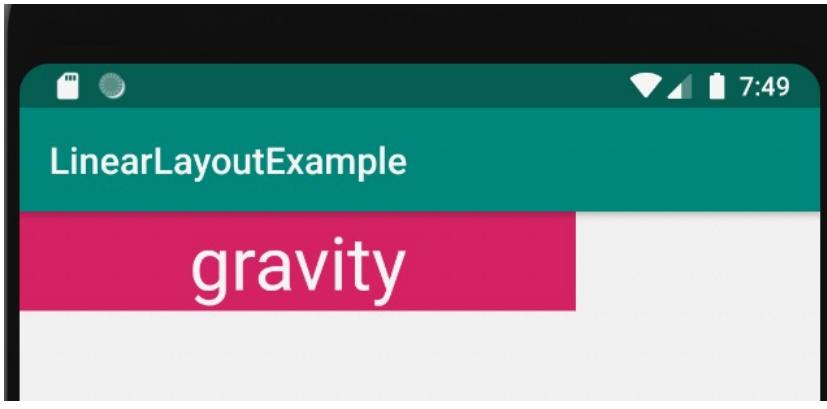
`android:padding="30dp"`

`android:layout_margin="30dp"`

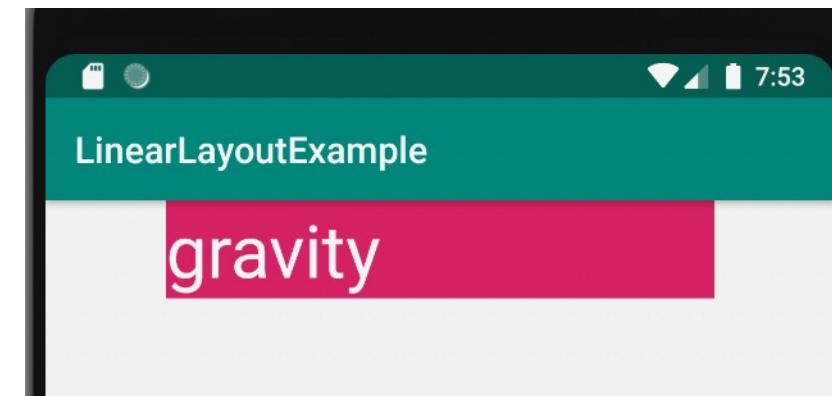
# 1. Thiết kế giao diện với Layout

## ➤ Linear Layout

- ✓ *Thuộc tính hỗ trợ thiết kế: gravity*



android:gravity="center"

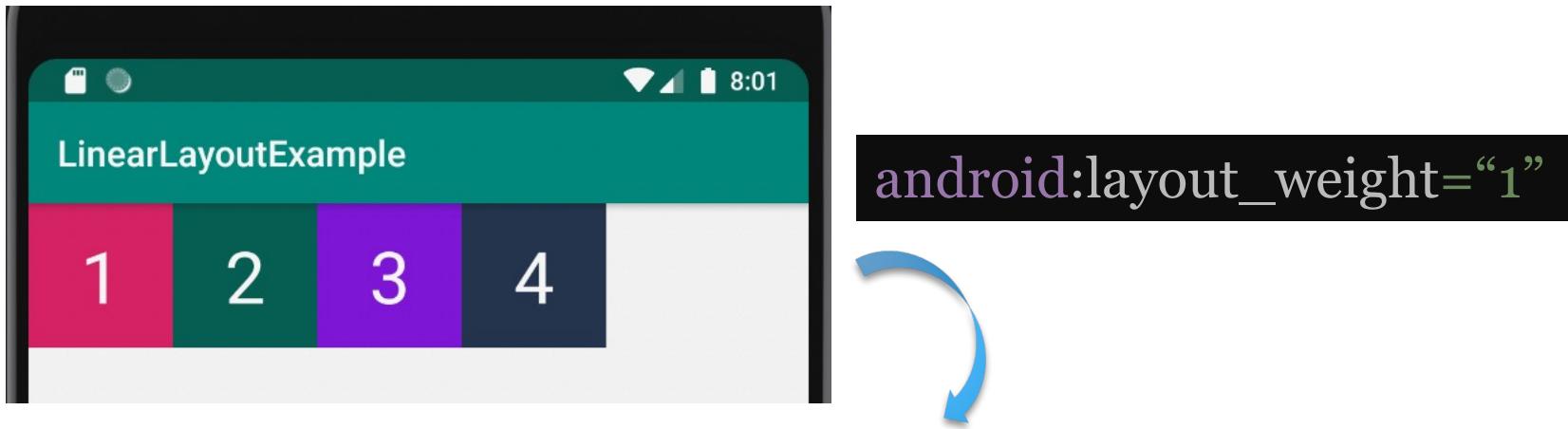


android:layout\_gravity="center"

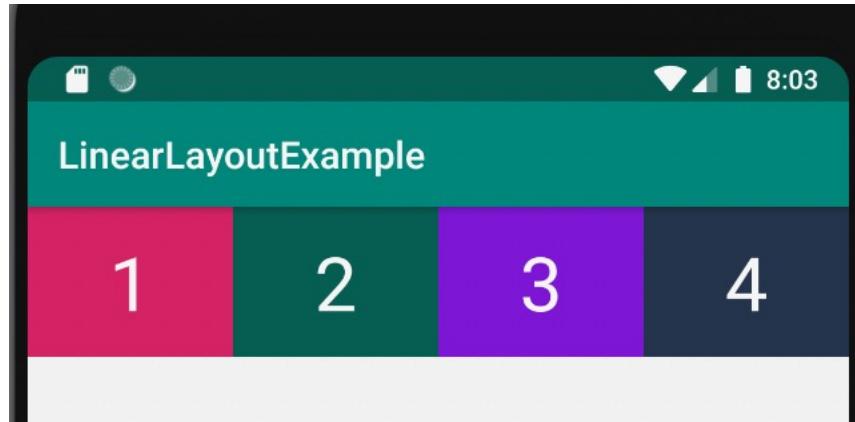
# 1. Thiết kế giao diện với Layout

## ➤ Linear Layout

- ✓ *Thuộc tính hỗ trợ thiết kế: weight*



➔ Thuộc tính “layout\_weight” giúp chia đều tỉ lệ hiển thị của các view.



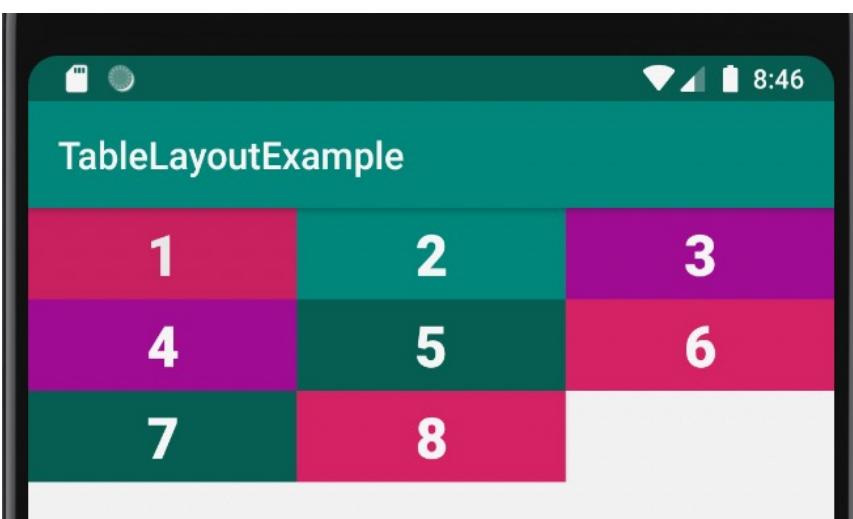
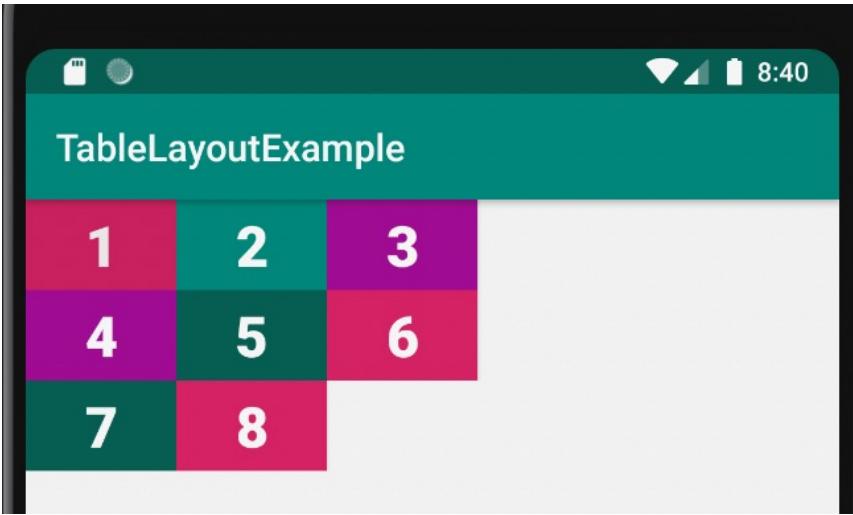
# 1. Thiết kế giao diện với Layout

## ➤ Table Layout

- ✓ *TableLayout kế thừa từ LinearLayout, cho phép hiển thị các đối tượng theo dạng bảng. Đối tượng view TableRow đại diện cho mỗi dòng, bên trong TableRow chứa các View con, mỗi View con này nằm ở vị trí một ô bảng (cell).*
- ✓ *Cột / dòng trong TableLayout tính từ vị trí số 0.*
- ✓ *TableLayout sẽ xem dòng nào có số lượng view nhiều nhất để xác định rằng nó có bao nhiêu cột.*

# 1. Thiết kế giao diện với Layout

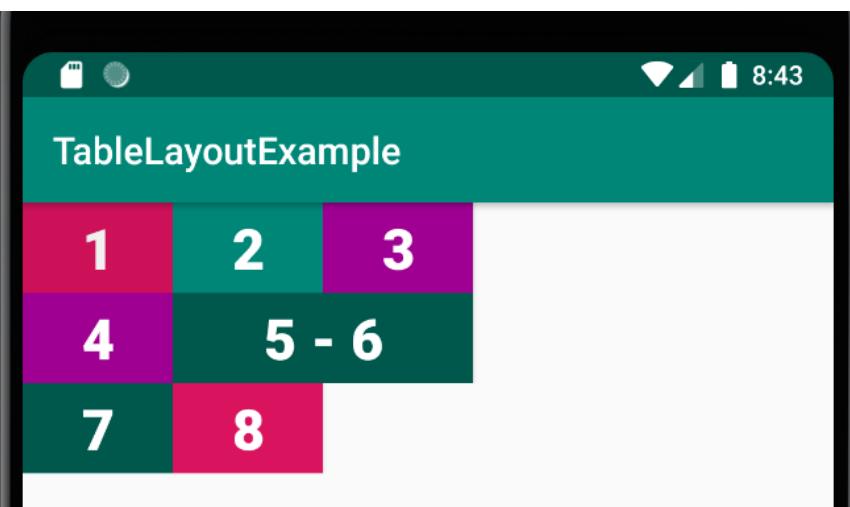
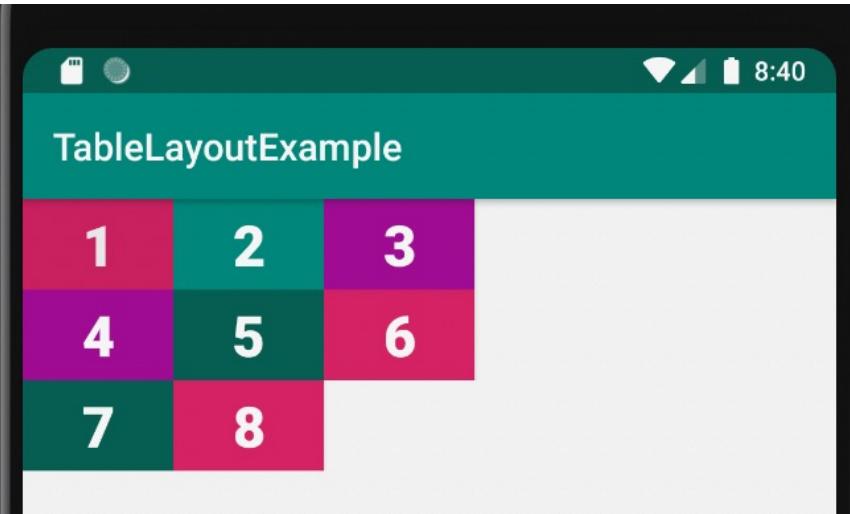
## ➤ Table Layout



```
<TableLayout  
    xmlns:android="http://schemas.android.  
    com/apk/res/android"  
    xmlns:tools="http://schemas.android.co  
    m/tools"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        tools:context=".MainActivity"  
        android:stretchColumns="*">  
  
</TableLayout>
```

# 1. Thiết kế giao diện với Layout

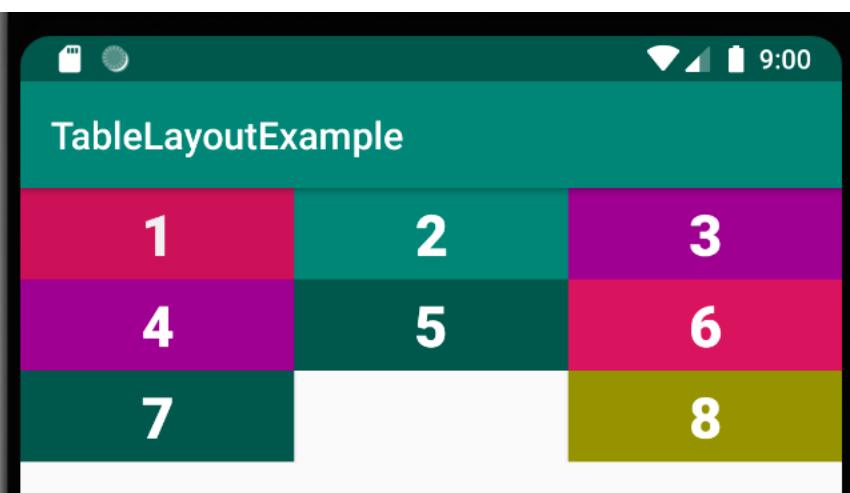
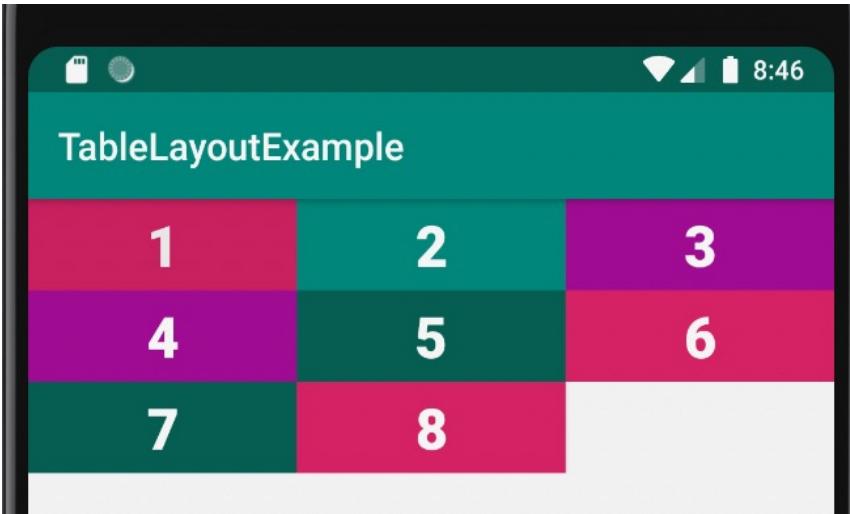
## ➤ Table Layout



```
<Button  
    android:id="@+id/button5"  
    android:layout_width="80dp"  
    android:layout_height="wrap_content"  
    android:background="@color/..."  
    android:text="5 - 6"  
    android:textColor="@color/mauTrang"  
    android:textSize="30sp"  
    android:textStyle="bold"  
    android:layout_span="2"/>
```

# 1. Thiết kế giao diện với Layout

## ➤ Table Layout

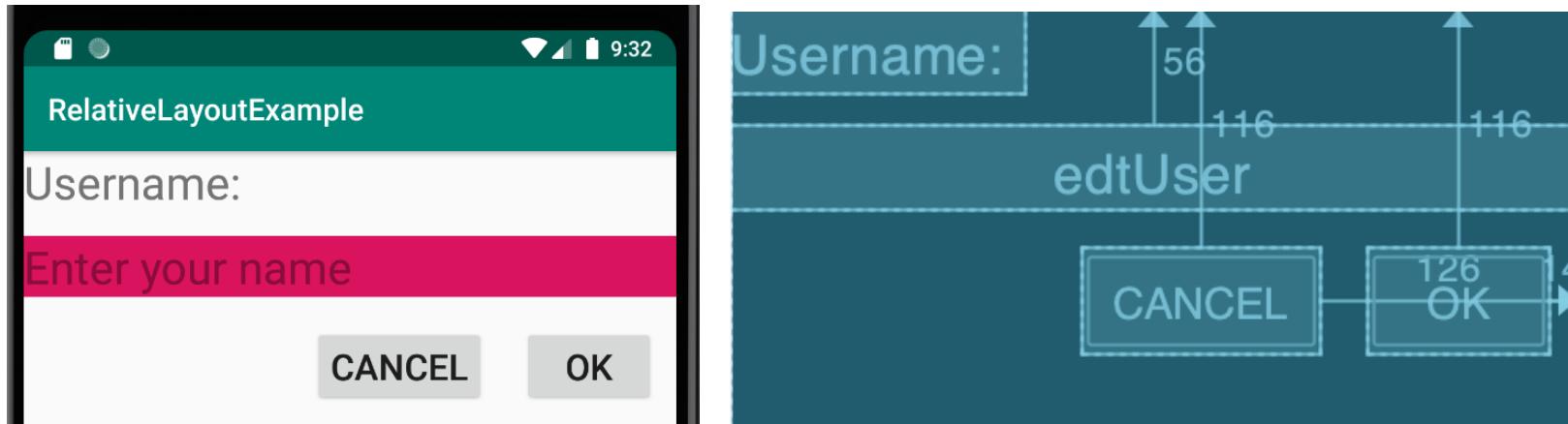


```
<Button  
    android:id="@+id/button8"  
    android:layout_width="80dp"  
    android:layout_height="wrap_content"  
    android:layout_column="2"  
    android:background="#959307" ↑  
    android:text="8"  
    android:textColor="@color/mauTrang"  
    android:textSize="30sp"  
    android:textStyle="bold" />
```

# 1. Thiết kế giao diện với Layout

## ➤ Relative Layout

- ✓ *RelativeLayout* là loại Layout mà trong đó vị trí của mỗi view con sẽ được xác định so với view khác hoặc so với thành phần cha của chúng thông qua ID.



```
<Button  
    android:id="@+id btnCancel"  
.....  
    android:layout_alignParentTop="true"  
    android:layout_alignParentEnd="true"  
    android:layout_marginTop="116dp"  
    android:layout_marginEnd="14dp"  
...../>
```

# 1. Thiết kế giao diện với Layout

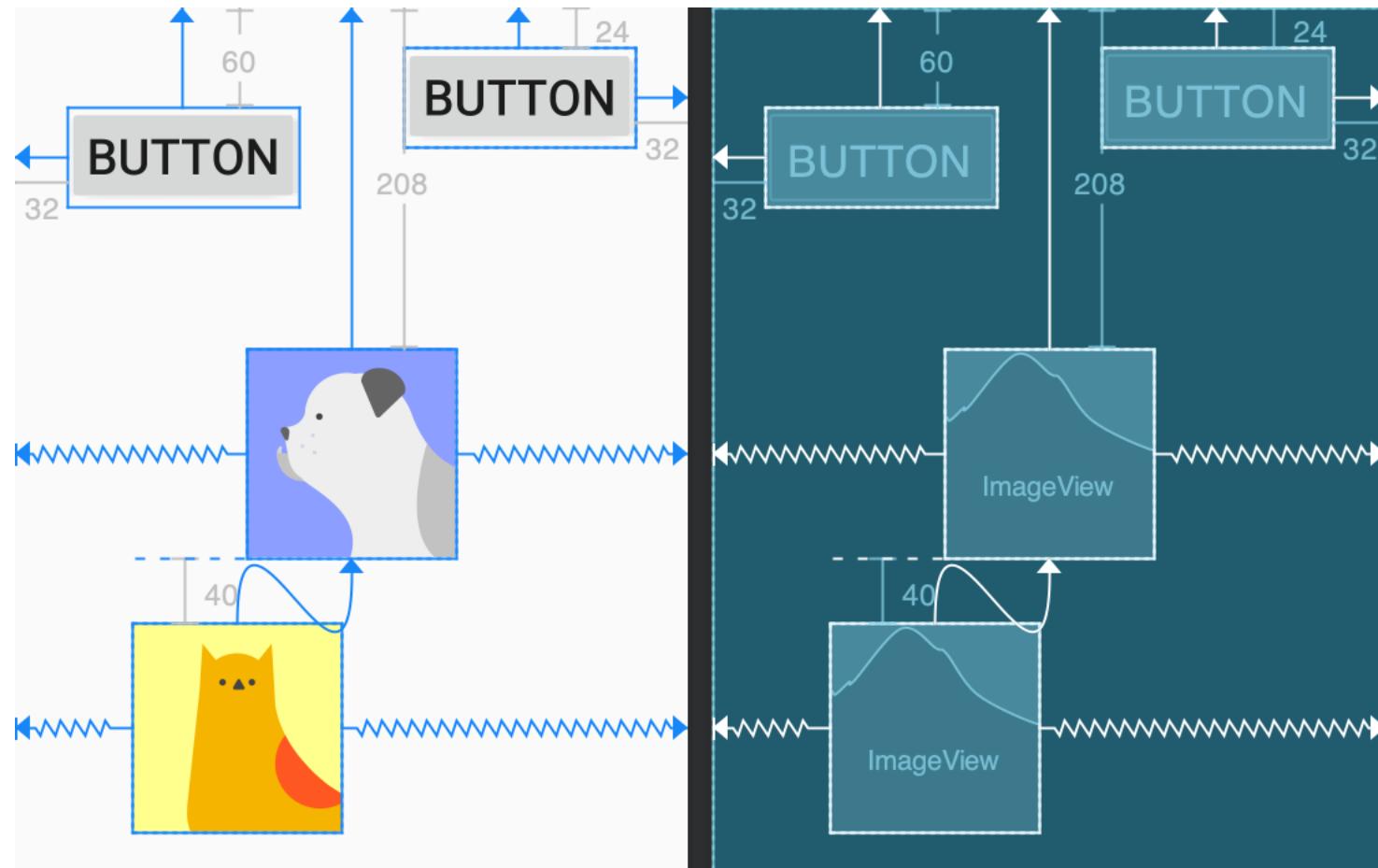
## ➤ Constraint Layout

- ✓ *ConstraintLayout giúp thiết kế giao diện với các ràng buộc, nguyên tắc liên kết nhất định giữa các thành phần (view).*
- ✓ *Mỗi một view phải có ít nhất một điểm neo theo chiều ngang và một điểm neo theo chiều dọc.*

# 1. Thiết kế giao diện với Layout

## ➤ Constraint Layout

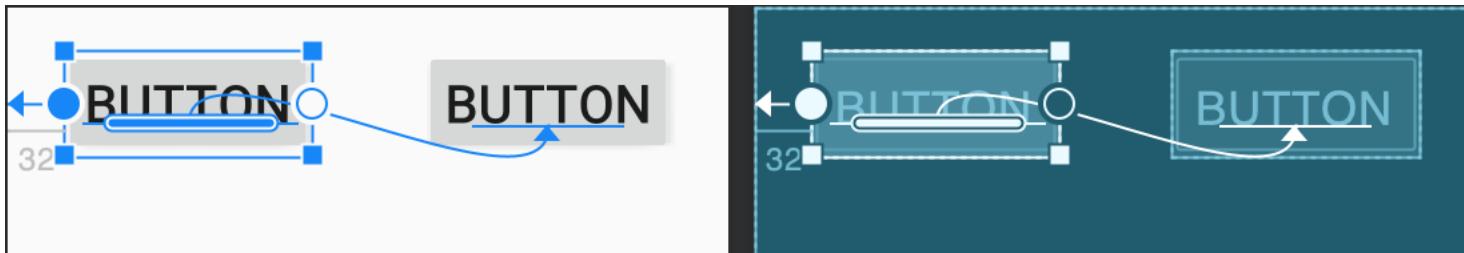
- ✓ Tạo constraint



# 1. Thiết kế giao diện với Layout

## ➤ Constraint Layout

- ✓ Tạo baseline constraint

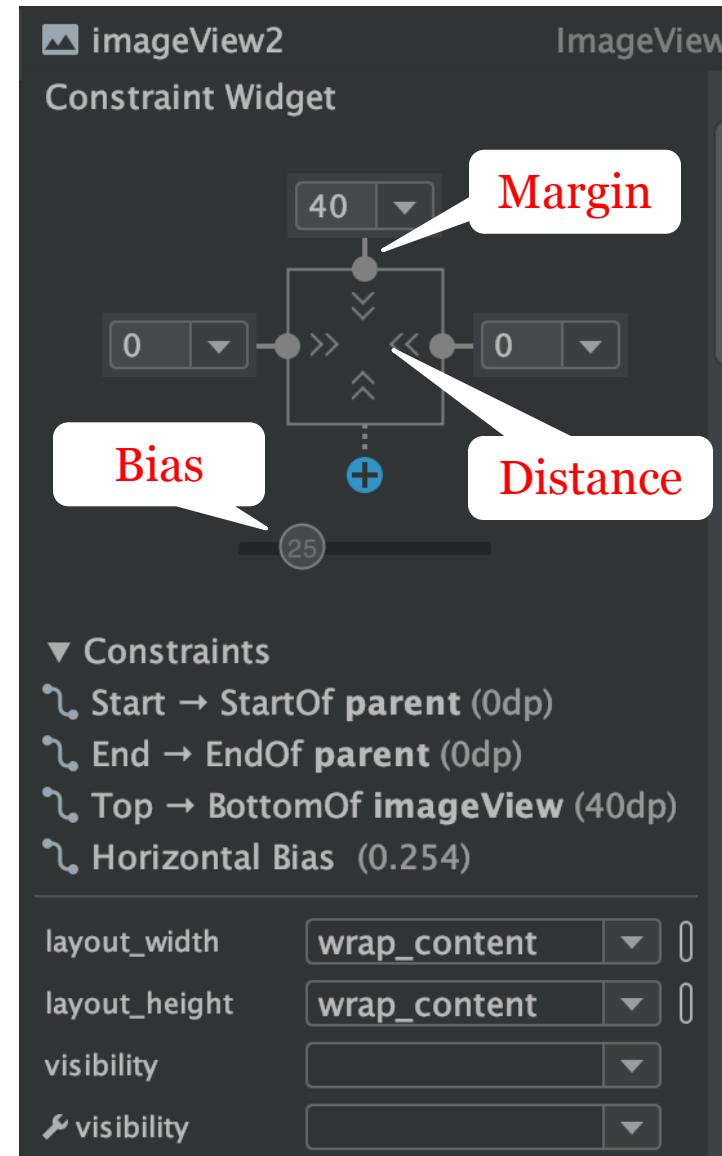
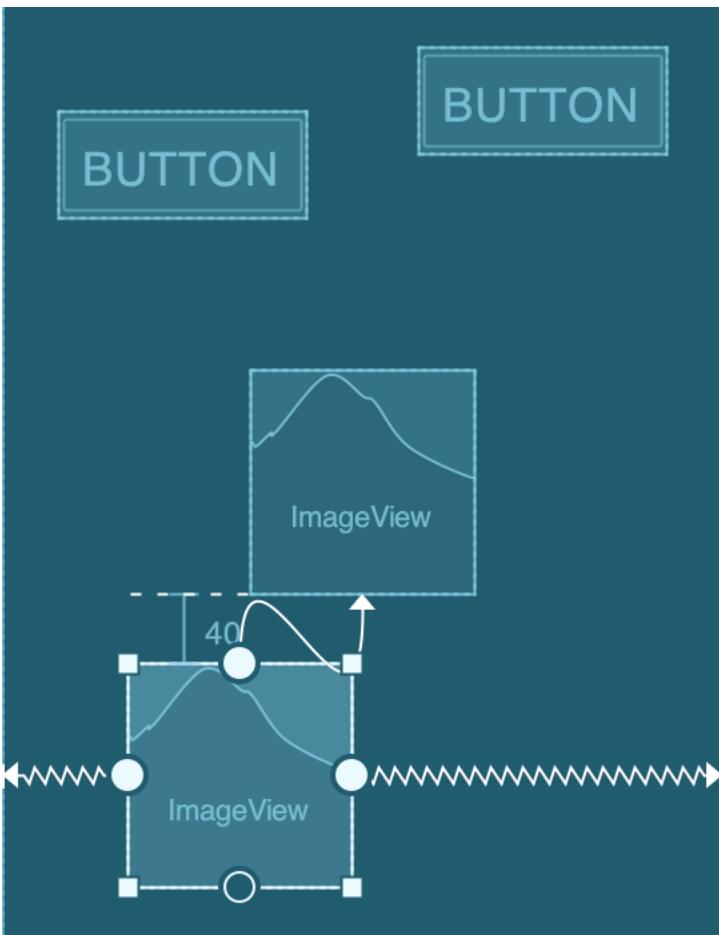


➔ Baseline constraint giúp căn chỉnh vị trí các view dựa theo text bên trong, phù hợp cho các view như: TextView, EditText, Button, ...

# 1. Thiết kế giao diện với Layout

## ➤ Constraint Layout

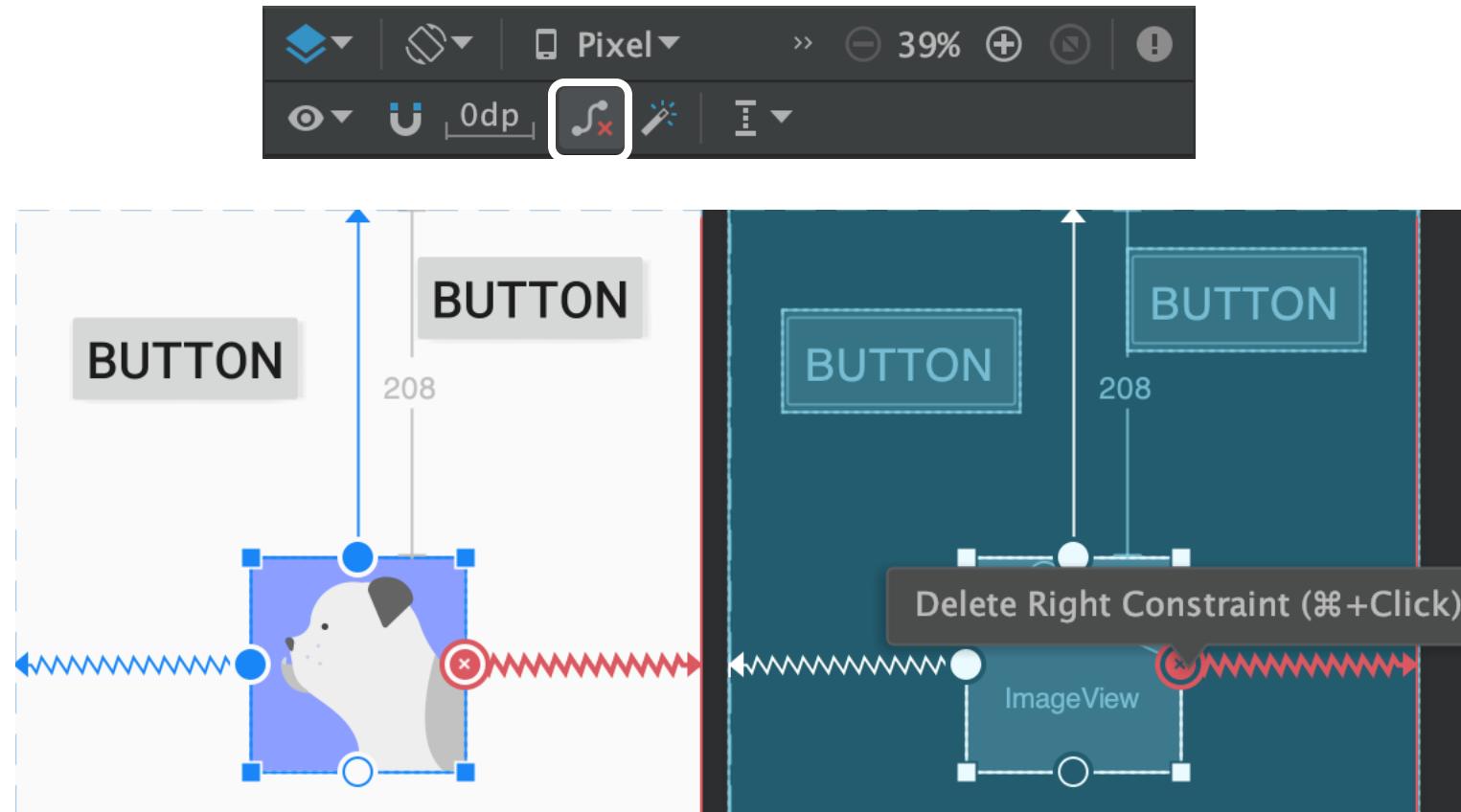
- ✓ Tạo constraint



# 1. Thiết kế giao diện với Layout

## ➤ Constraint Layout

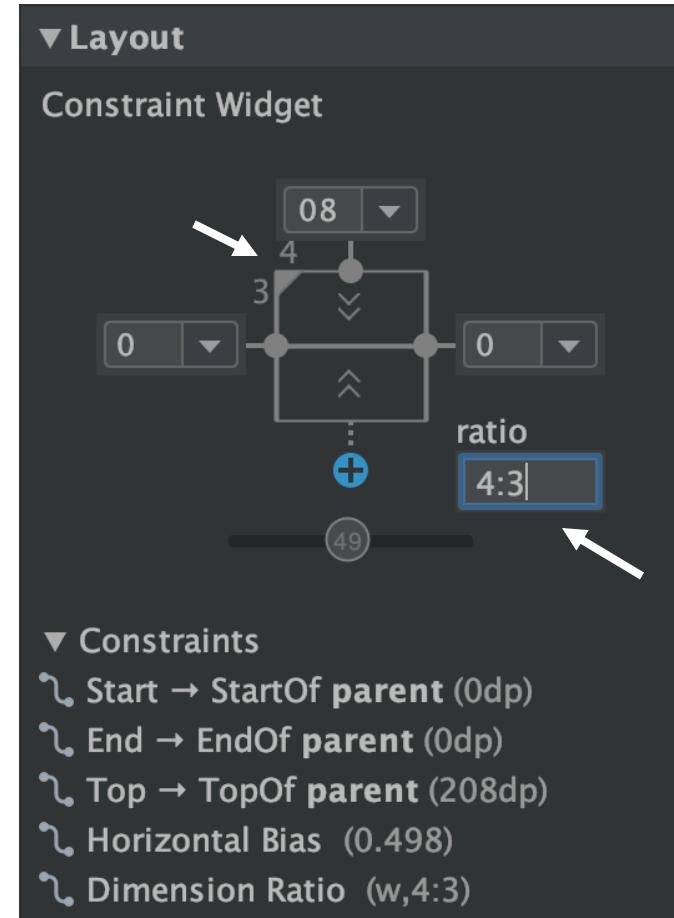
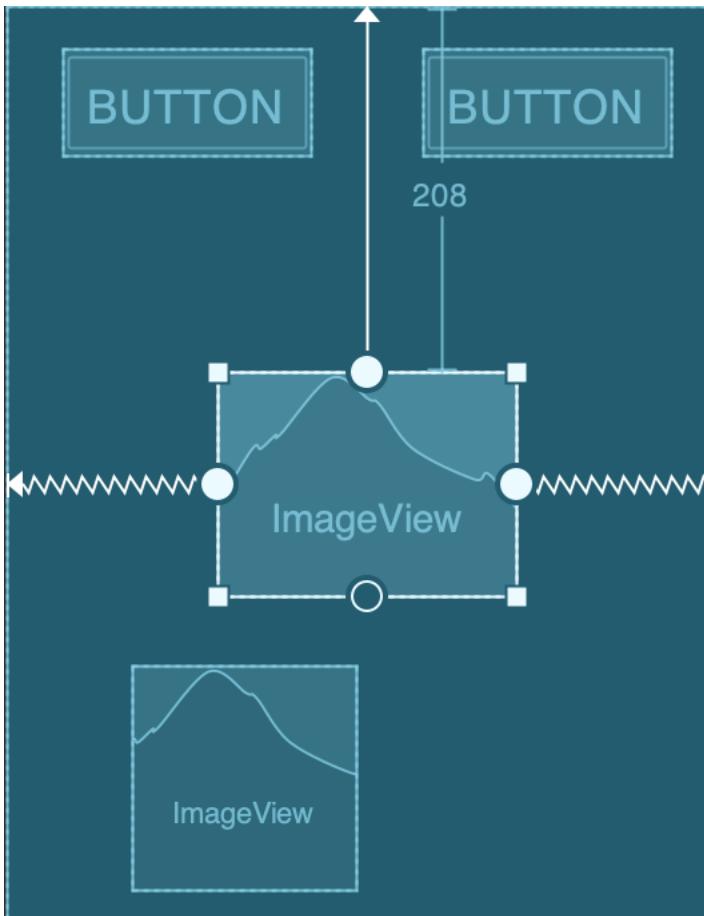
- ✓ Xóa constraint



# 1. Thiết kế giao diện với Layout

## ➤ Constraint Layout

- ✓ Ratio -> điều chỉnh kích thước view theo tỷ lệ

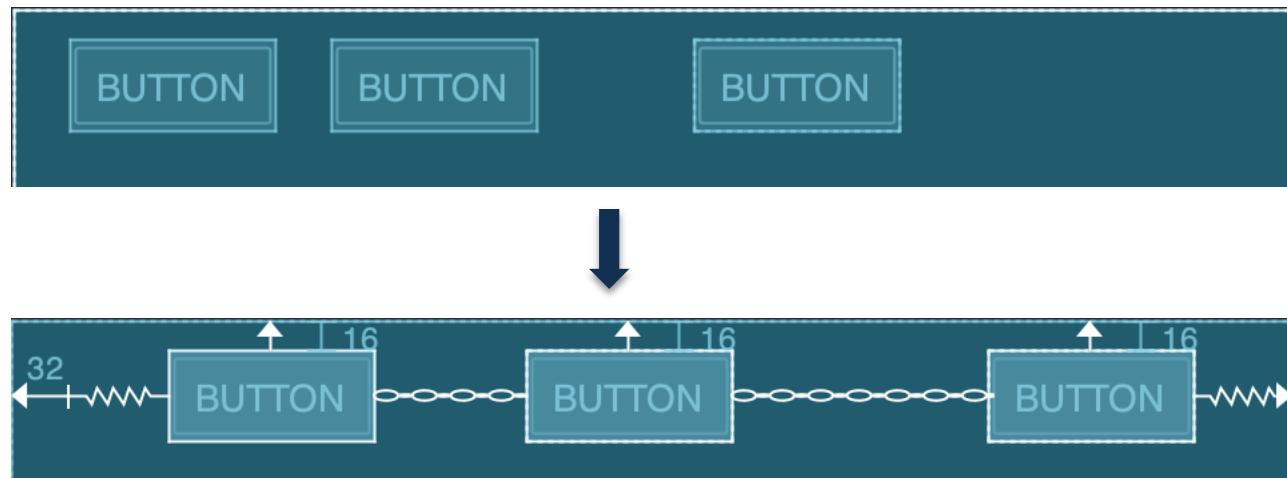


# 1. Thiết kế giao diện với Layout

## ➤ Constraint Layout

- ✓ Chain -> xâu chuỗi (xích) các view

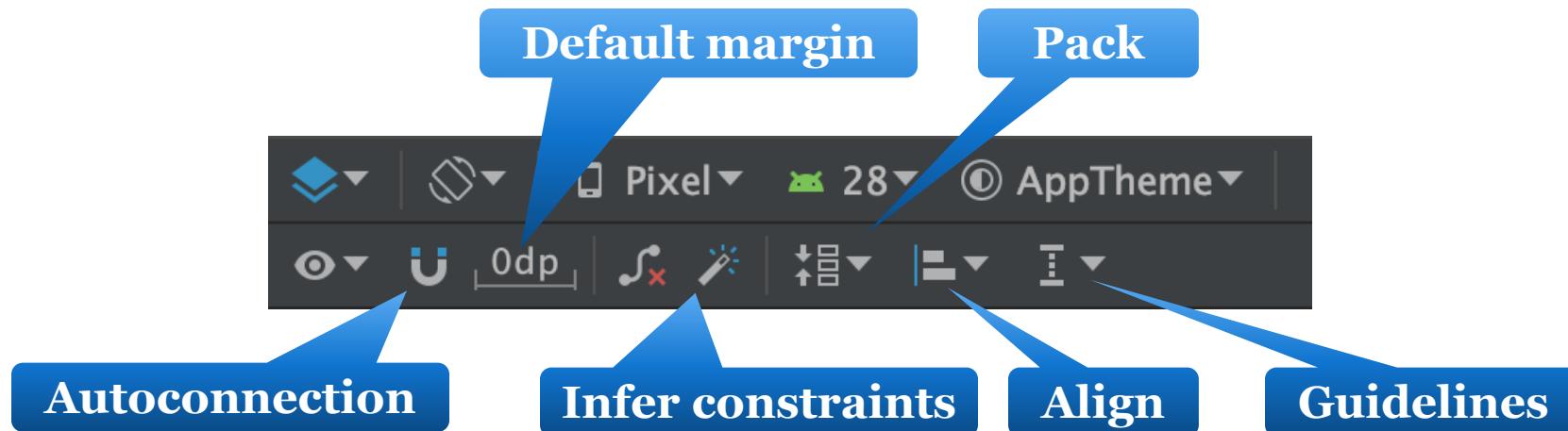
*Tạo chain: chọn tất cả các View → click phải chuột → Chains → Create Horizontal/Vertical Chain*



# 1. Thiết kế giao diện với Layout

## ➤ Constraint Layout

- ✓ Một số tính năng khác



## 2. Views

### ➤ TextView

- ✓ *Dùng hiển thị dữ liệu văn bản, không cho phép chỉnh sửa, thường được đặt tên với prefix txt.*
- ✓ *Thiết lập nội dung hiển thị:*
  - *Trong Java code (Activity):*

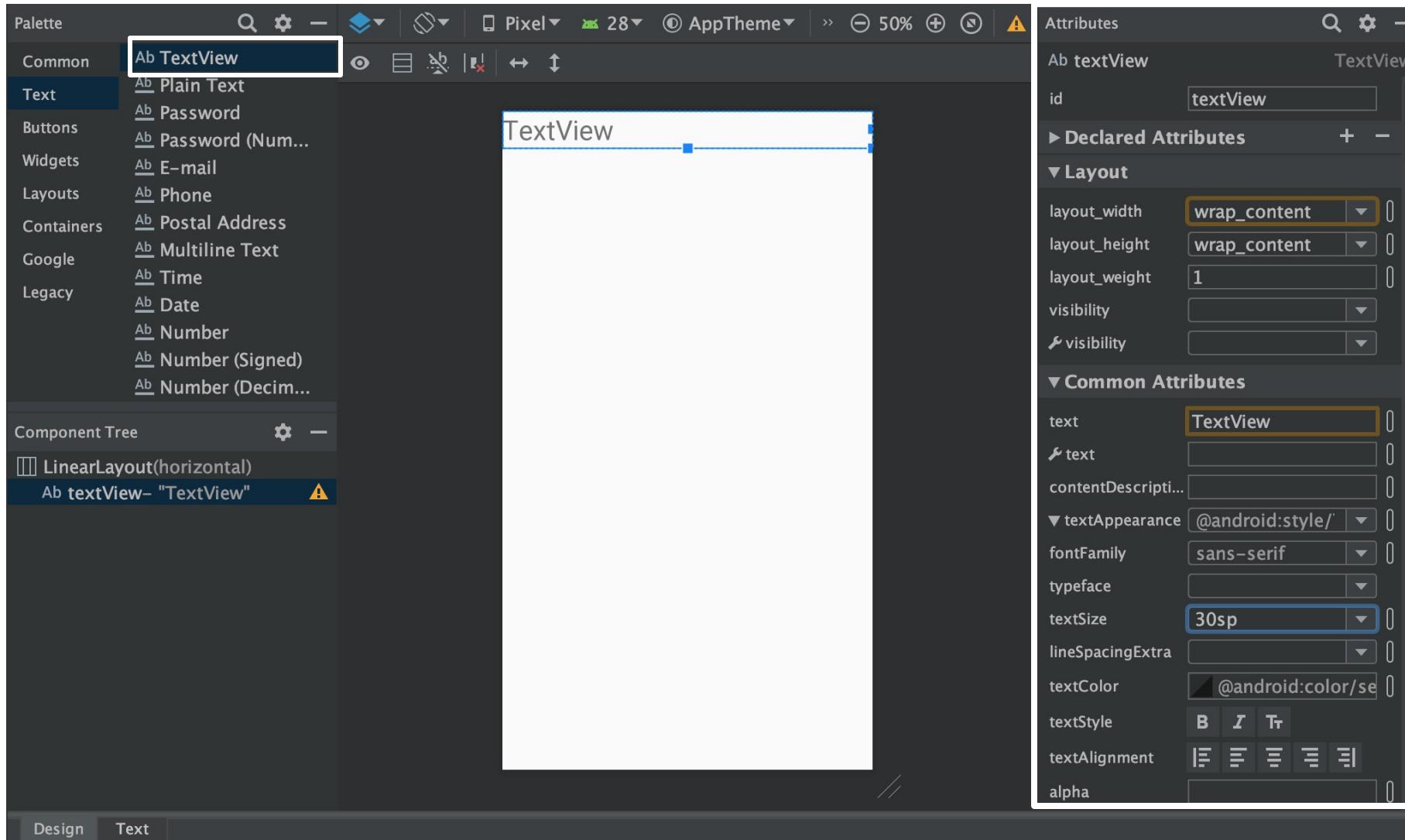
```
textView.setText("TextView");
```

- *Trong XML:*

```
android:text="TextView"
```

## 2. Views

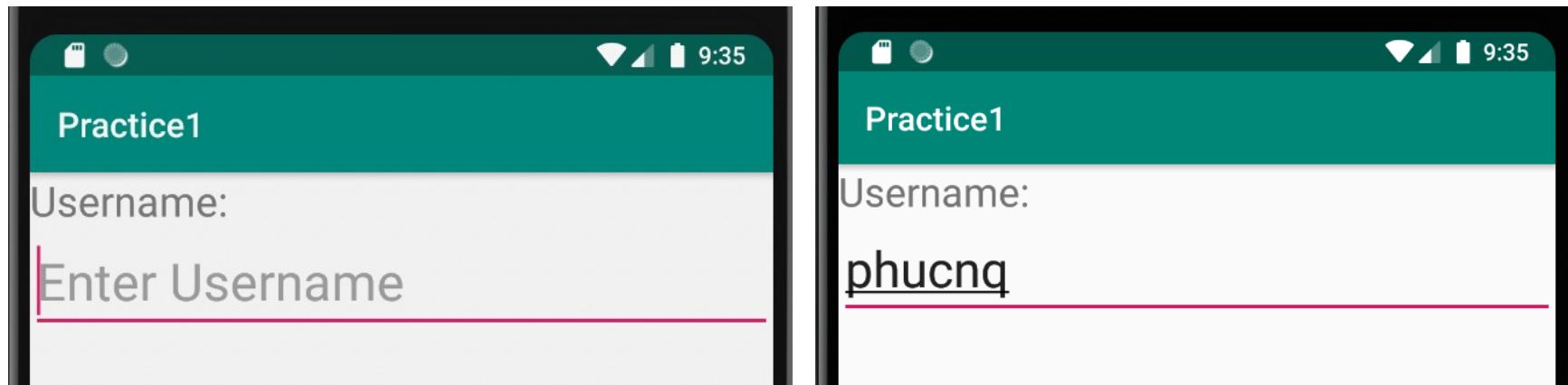
### ➤ TextView



## 2. Views

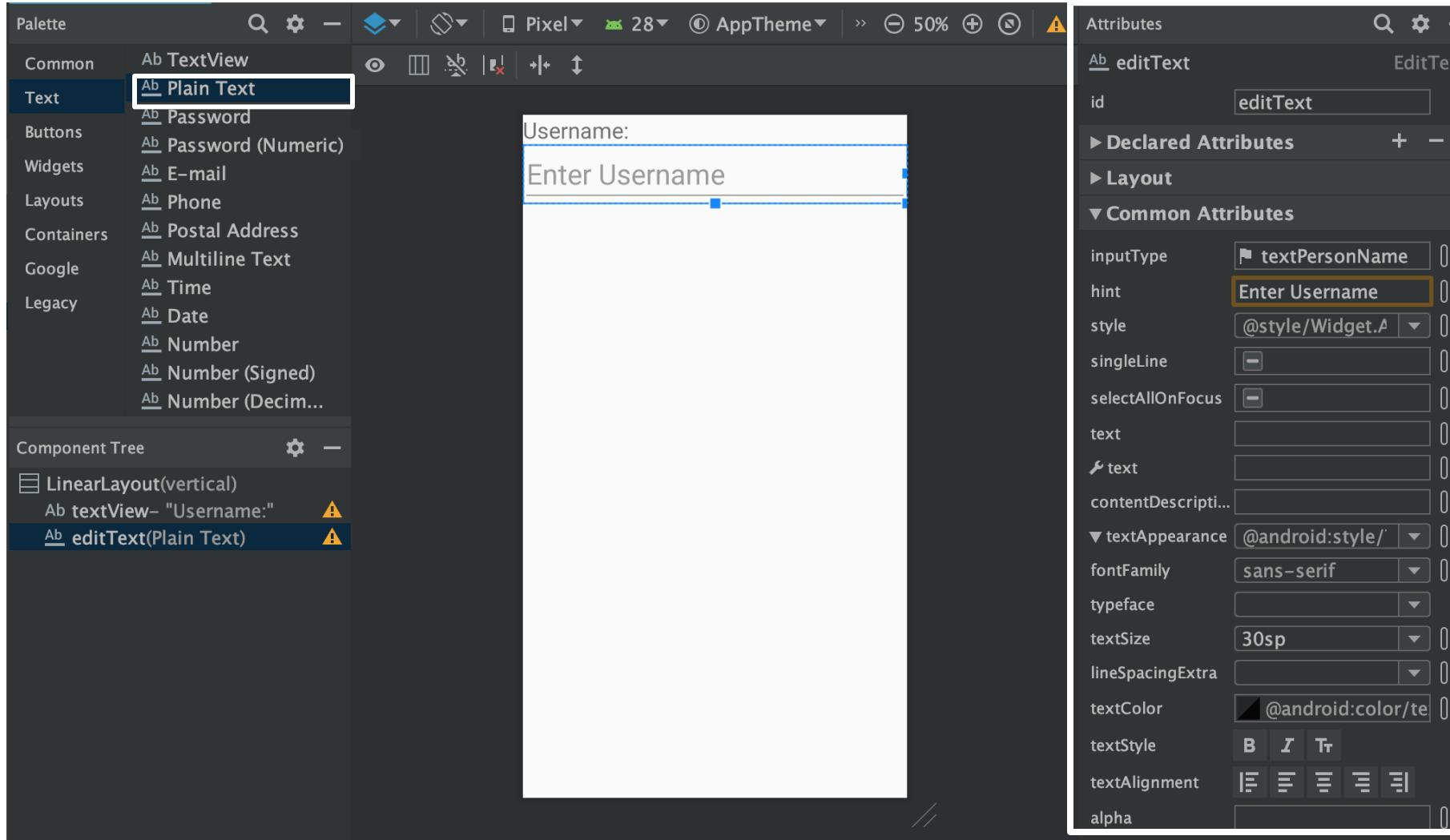
### ➤ EditText

- ✓ *Dùng hiển thị và cho phép thay đổi dữ liệu văn bản, thường được đặt tên với prefix **edt**.*
- ✓ *Có các thuộc tính tương tự như TextView.*



## 2. Views

### ➤ EditText



## 2. Views

### ➤ Button

- ✓ Cho phép hiển thị dữ liệu văn bản, hình ảnh; thường được đặt tên với prefix **btn**.
- ✓ Nhận và phản hồi tương tác “nhấn” từ người dùng.
- ✓ *Lắng nghe sự kiện:*
  - Trong Java code:

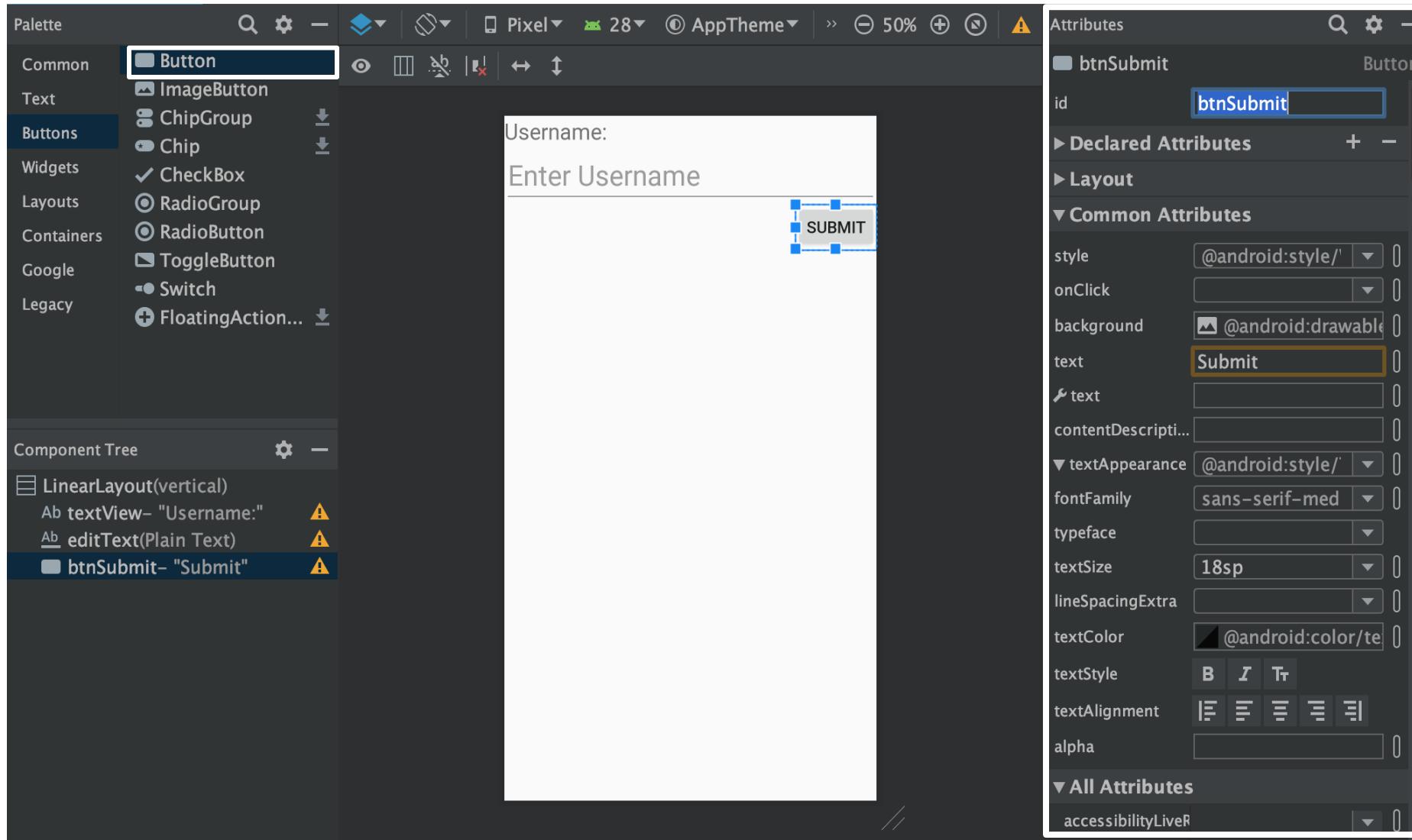
```
btnSubmit = findViewById(R.id.btnSubmit);
btnSubmit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Xử lý ...
    }
});
```

- Trong XML:

```
android:onClick="MethodName"
```

## 2. Views

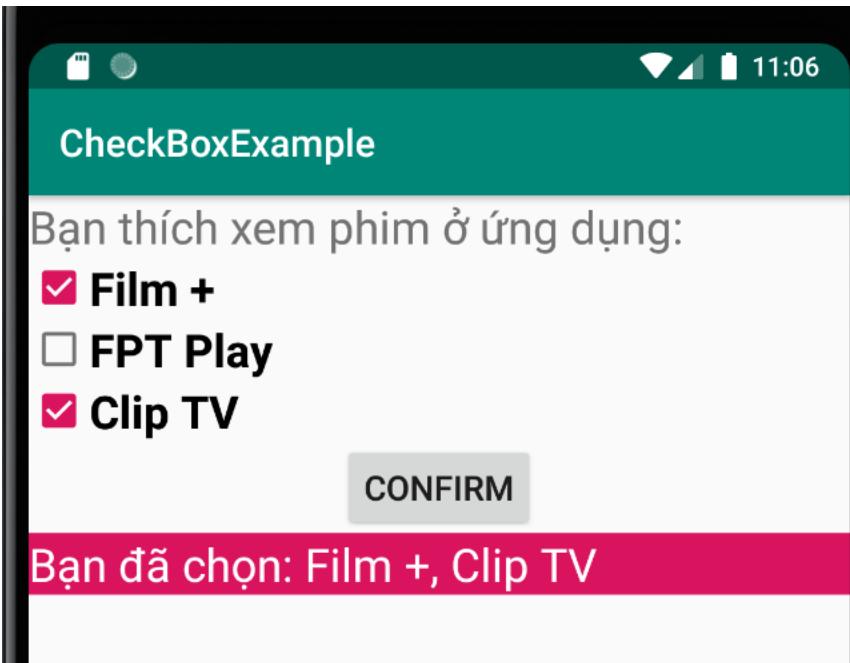
### ➤ Button



## 2. Views

### ➤ CheckBox

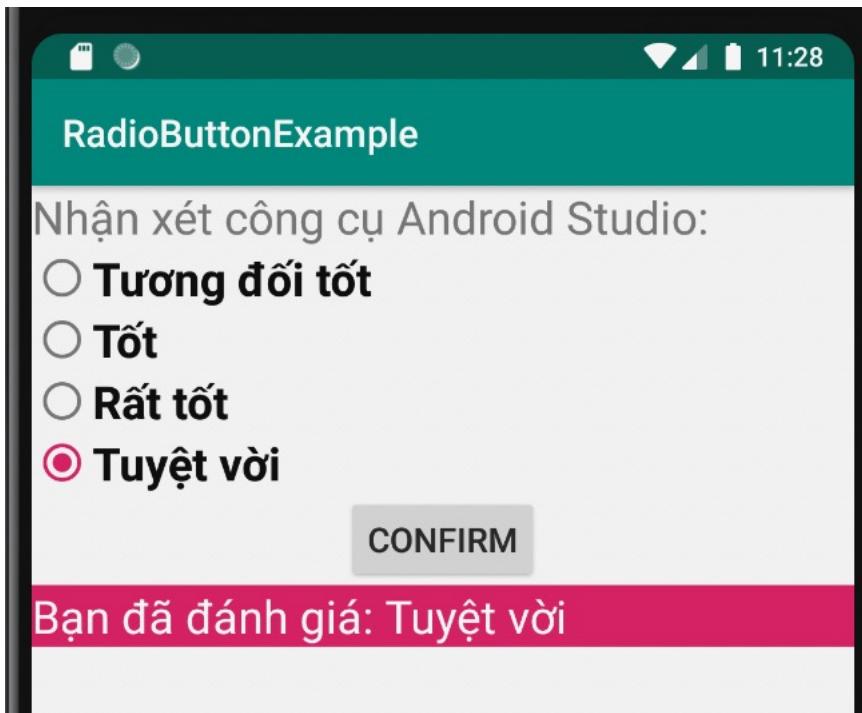
- ✓ Cho phép chọn nhiều tùy chọn, thường được đặt tên với prefix **chk**.



## 2. Views

### ➤ RadioButton

- ✓ Cho phép chọn duy nhất một tùy chọn, thường được đặt tên với prefix *rad*.



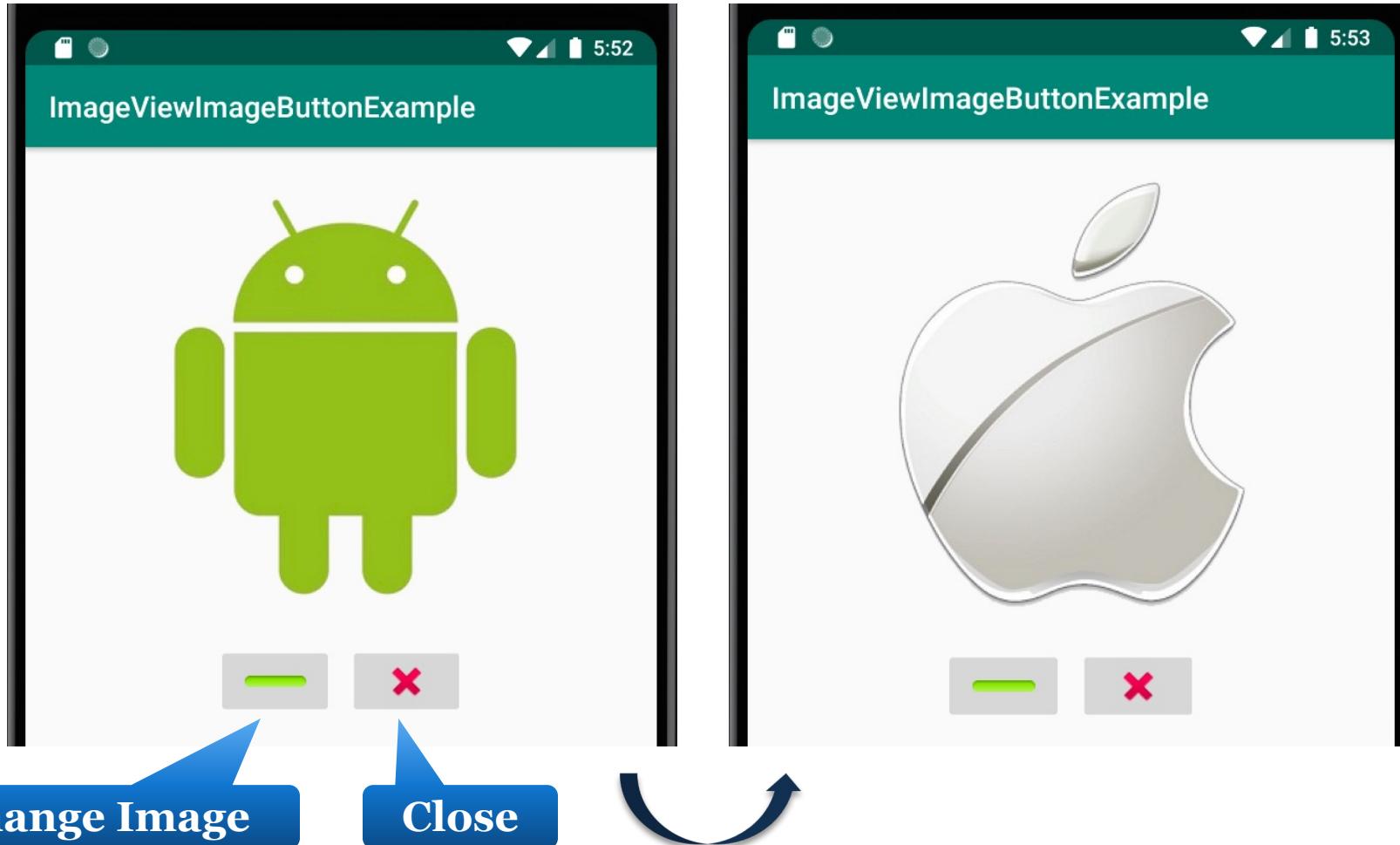
## 2. Views

### ➤ ImageButton, ImageView

- ✓ *ImageButton* là một button cho phép chứa hình ảnh, thường được đặt tên với prefix *btn*, *imgBtn*.
- ✓ *ImageView* dùng để hiển thị hình ảnh, cho phép tương tác; thường được đặt tên với prefix *imv*.

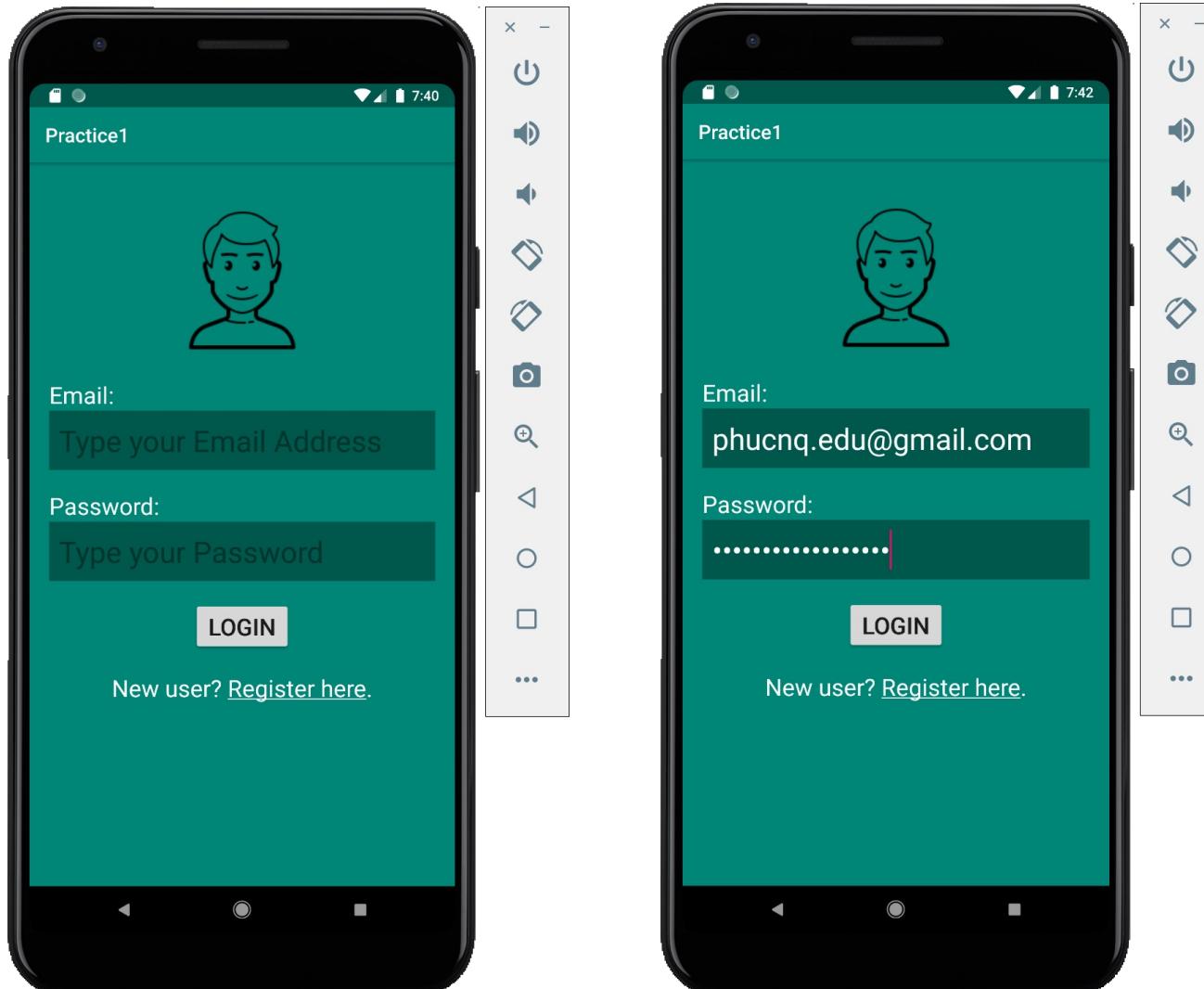
## 2. Views

### ➤ ImageButton, ImageView



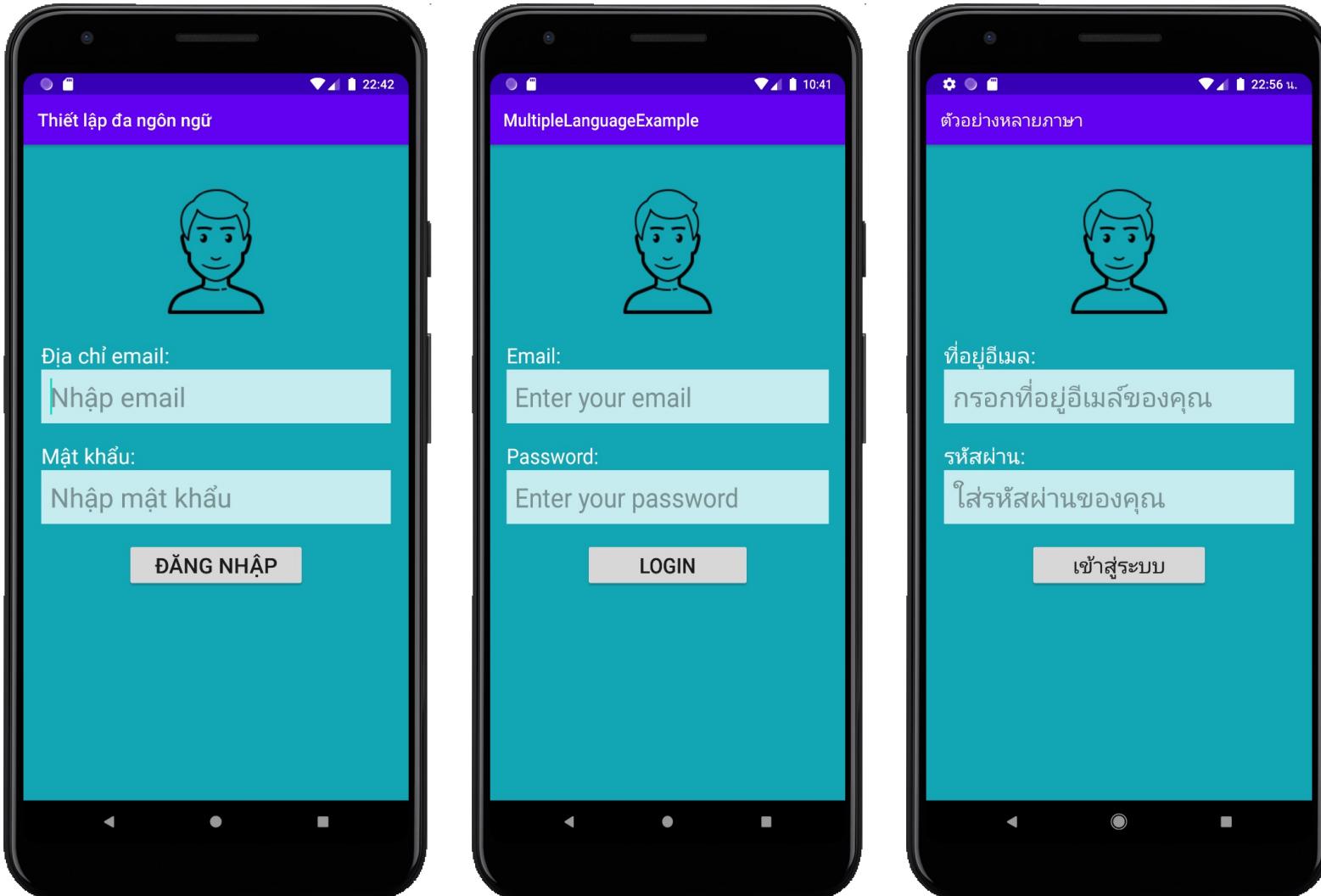
### 3. Bài tập

➤ **Bài tập 1** → Thiết kế màn hình đăng nhập



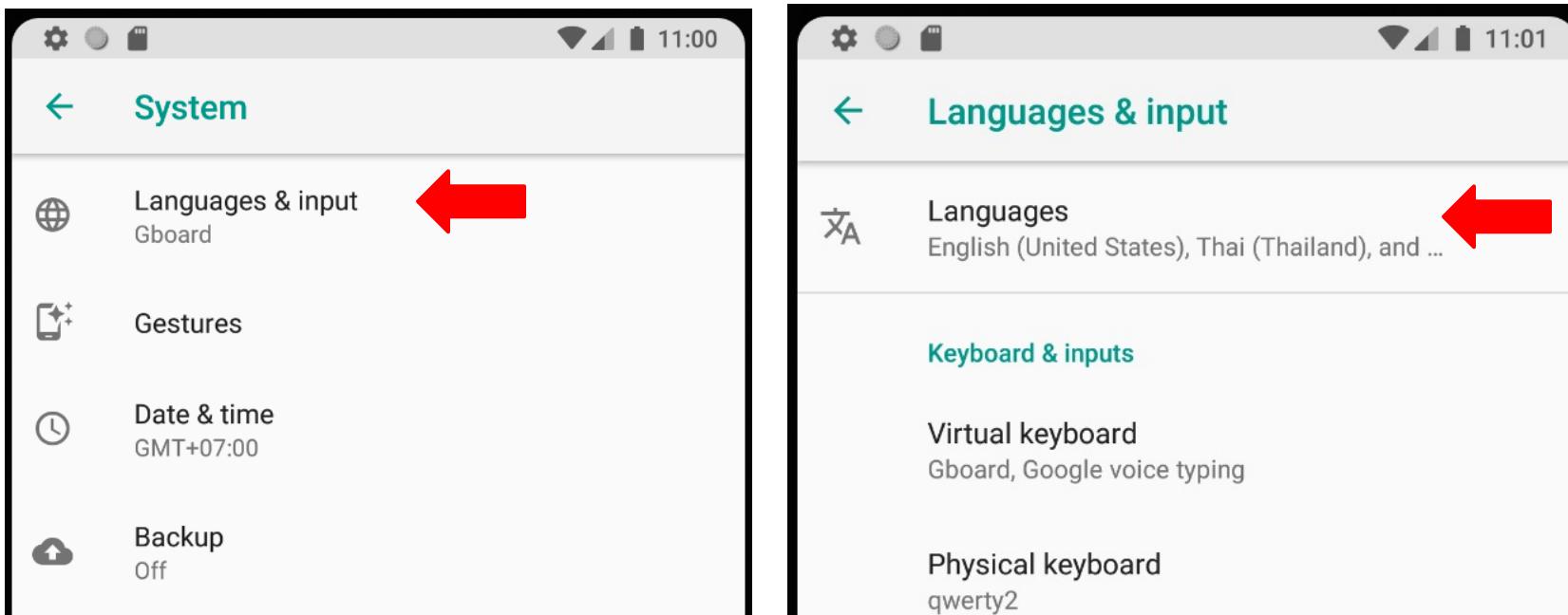
### 3. Bài tập

➤ **Bài tập 1** → Thiết lập đa ngôn ngữ cho ứng dụng



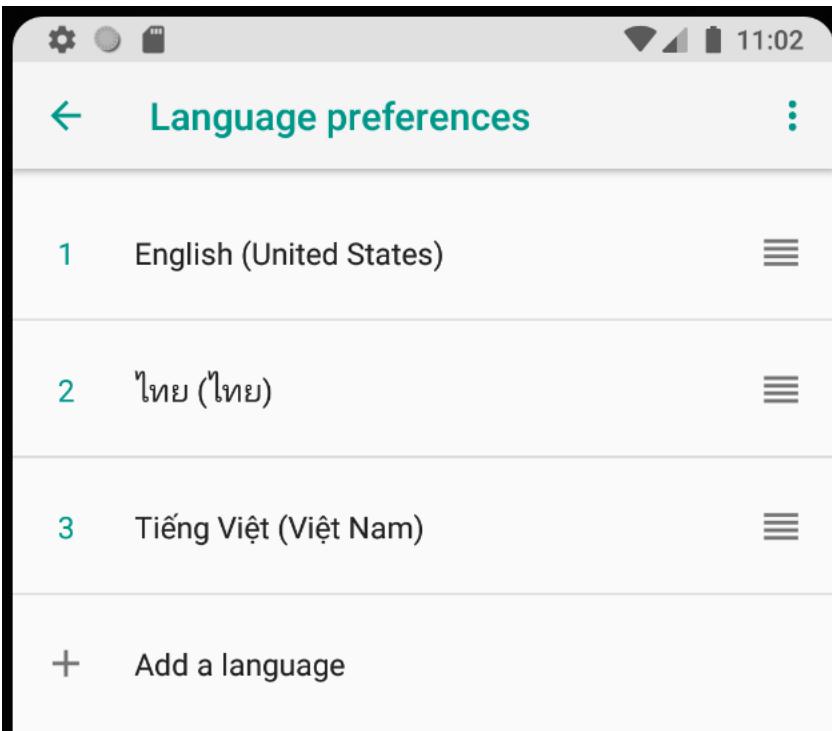
### 3. Bài tập

➤ **Bài tập 1** → Thiết lập đa ngôn ngữ cho ứng dụng



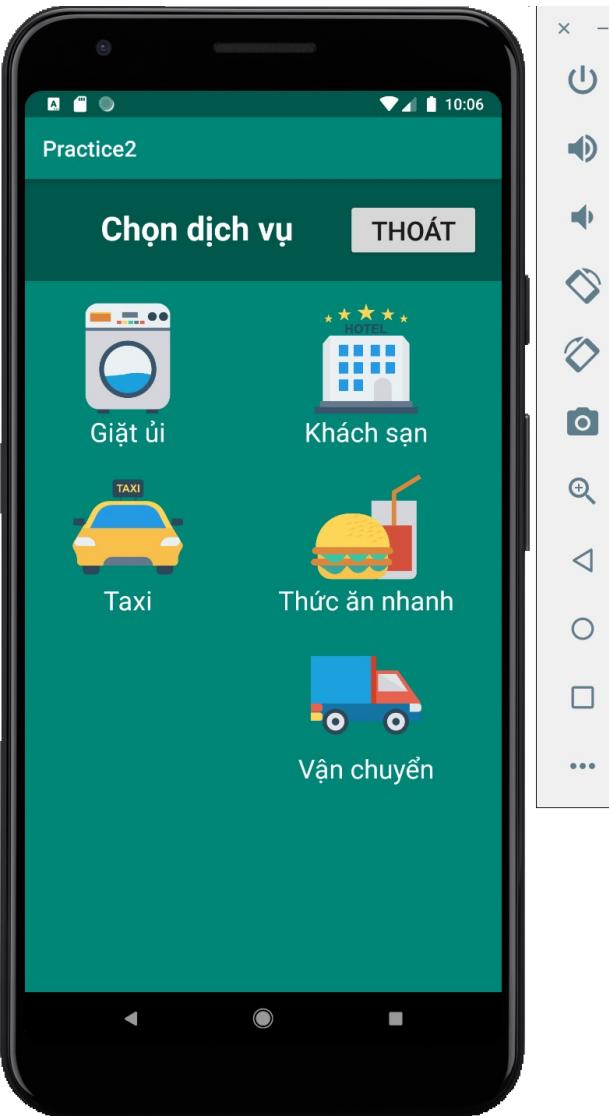
### 3. Bài tập

➤ **Bài tập 1** → Thiết lập đa ngôn ngữ cho ứng dụng



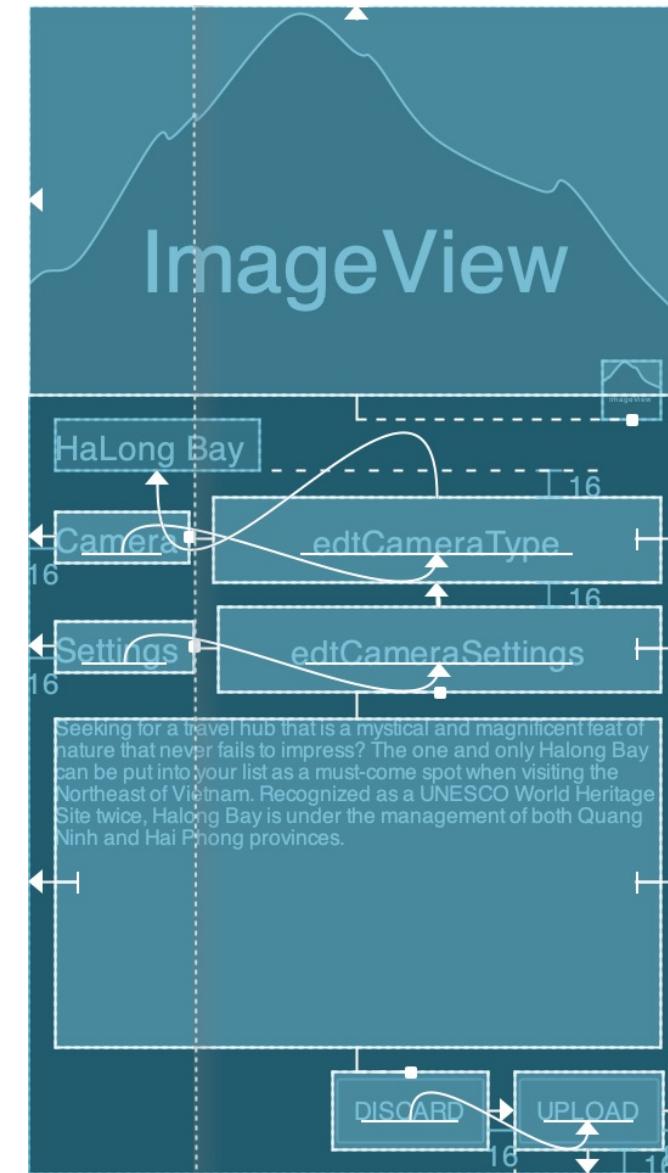
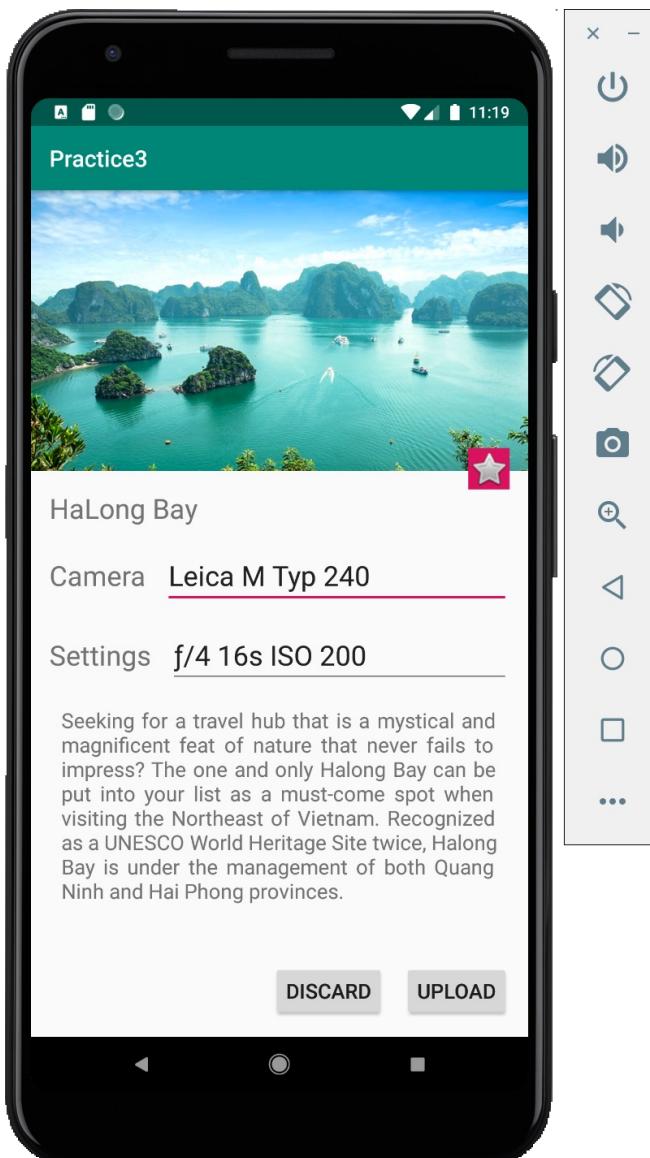
### 3. Bài tập rèn luyện

#### ➤ Bài tập 2



### 3. Bài tập rèn luyện

#### ➤ Bài tập 3



03

## EVENT & NOTIFICATION

# NỘI DUNG

## 1. Kỹ thuật gán sự kiện

- *Onclick XML*
- *Anonymous Listener*
- *Variable as Listener*
- *Activity as Listener*
- *Explicit class Listener*
- *View Subclassing*

## 2. Cơ chế thông báo

- *Toast*
- *Alert dialog*
- *Custom dialog*
- *Notification*

# 1. Kỹ thuật gán sự kiện

## ➤ Onclick XML

```
<ImageButton  
    android:id="@+id/imgChangeImage"  
    android:layout_width="80dp"  
    android:layout_height="50dp"  
    app:srcCompat="@android:drawable/button_onoff_indicator_on"  
    android:onClick="xuLyDoiAnh"  
    android:contentDescription="@string/todo" />
```

```
public void xuLyDoiAnh(View view) {  
    if(imageView.getTag() == null || imageView.getTag().equals("android")){  
        imageView.setImageResource(R.drawable.ios);  
        imageView.setTag("ios");  
    }else{  
        imageView.setImageResource(R.drawable.android);  
        imageView.setTag("android");  
    }  
}
```

# 1. Kỹ thuật gán sự kiện

## ➤ Anonymous Listener

```
btnGreen.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        txtColor.setBackgroundColor(Color.GREEN);
    }
});
```

```
btnBlue.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View view) {
        txtColor.setBackgroundColor(Color.BLUE);
        return false;
    }
});
```

# 1. Kỹ thuật gán sự kiện

## ➤ Variable as Listener

```
View.OnClickListener myClick = new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        if(view.getId() == R.id.btnHide){  
            imgPhoto.setVisibility(View.INVISIBLE);  
        }else if(view.getId() == R.id.btnShow){  
            imgPhoto.setVisibility(View.VISIBLE);  
        }  
    }  
};  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    addViews();  
    btnShow.setOnClickListener(myClick);  
    btnHide.setOnClickListener(myClick);  
}
```

# 1. Kỹ thuật gán sự kiện

## ➤ Activity as Listener

```
public class MainActivity extends AppCompatActivity implements  
View.OnClickListener {  
    ....  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ....  
        btnPic1.setOnClickListener(this);  
        btnPic2.setOnClickListener(this);  
    }  
    ....  
    @Override  
    public void onClick(View view) {  
        if(view.getId() == R.id.btnPic1){  
            imgPic.setImageResource(R.drawable.android_icon);  
        }else if(view.getId() == R.id.btnPic2){  
            imgPic.setImageResource((R.drawable.ios_icon));  
        }  
    }  
}
```

# 1. Kỹ thuật gán sự kiện

## ➤ Explicit class Listener

```
....  
btnMove.setOnClickListener(new MyEvents());  
....  
class MyEvents implements View.OnClickListener {  
    @Override  
    public void onClick(View view) {  
        if(view.equals(btnExit)){  
            finish();  
        }  
    }  
}
```

# 1. Kỹ thuật gán sự kiện

## ➤ View Subclassing

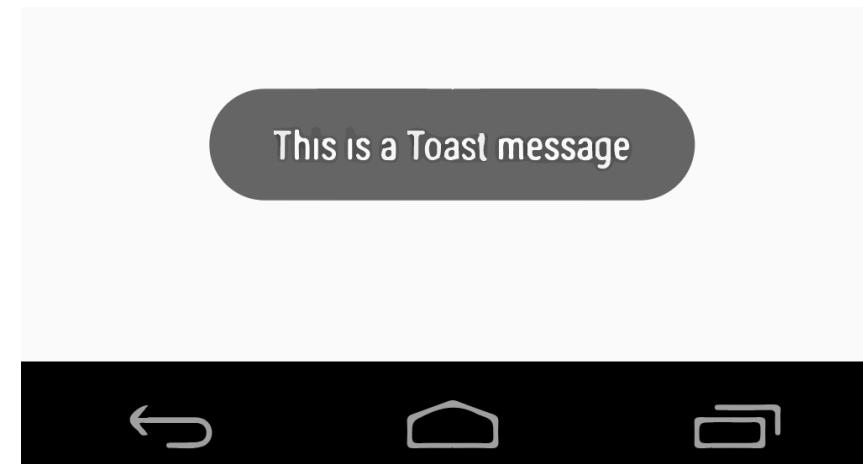
```
....  
Button btnBack = new Button(this){  
    @Override  
    public boolean performClick() {  
        //your code here;  
        return super.performClick();  
    }  
};  
btnBack.setText("Back");  
btnBack.setLayoutParams(params);  
linearLayout.addView(btnBack);  
....
```

## 2. Cơ chế thông báo

### »» Toast

#### ➤ Đặc điểm chức năng

- Toast có thể được tạo và hiển thị trong Activity hoặc trong Service.
- Không cho phép người sử dụng tương tác, hiển thị sau khoảng thời gian nào đó sẽ tự đóng lại.
- Có 2 giá trị mặc định:
  - Toast.LENGTH\_SHORT,
  - Toast.LENGTH\_LONG.



## 2. Cơ chế thông báo

### »» Toast

#### ➤ LENGTH\_SHORT, LENGTH\_LONG

```
Toast t = Toast.makeText(MainActivity.this,  
                            "Welcome to VietNam",  
                            Toast.LENGTH_SHORT);  
t.show();
```

```
Toast.makeText(MainActivity.this,  
                            "Welcome to VietNam",  
                            Toast.LENGTH_LONG).show();
```

Khi nào nên sử dụng Toast?

## 2. Cơ chế thông báo

### »» Alert dialog

#### ➤ Đặc điểm chức năng

- Hiển thị hộp thoại cho phép người dùng tương tác



```
AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
```

## 2. Cơ chế thông báo

### »» Alert dialog

#### ➤ Một số thuộc tính

- **setTitle**: thiết lập tiêu đề cho Dialog,
- **setMessage**: thiết lập nội dung cho Dialog,
- **setIcon**: thiết lập Icon,
- **setPositiveButton**, **setNegativeButton**: thiết lập hiển thị Nút chọn cho Dialog,
- **create()**: để tạo Dialog,
- **show()**: để hiển thị Dialog.

## 2. Cơ chế thông báo

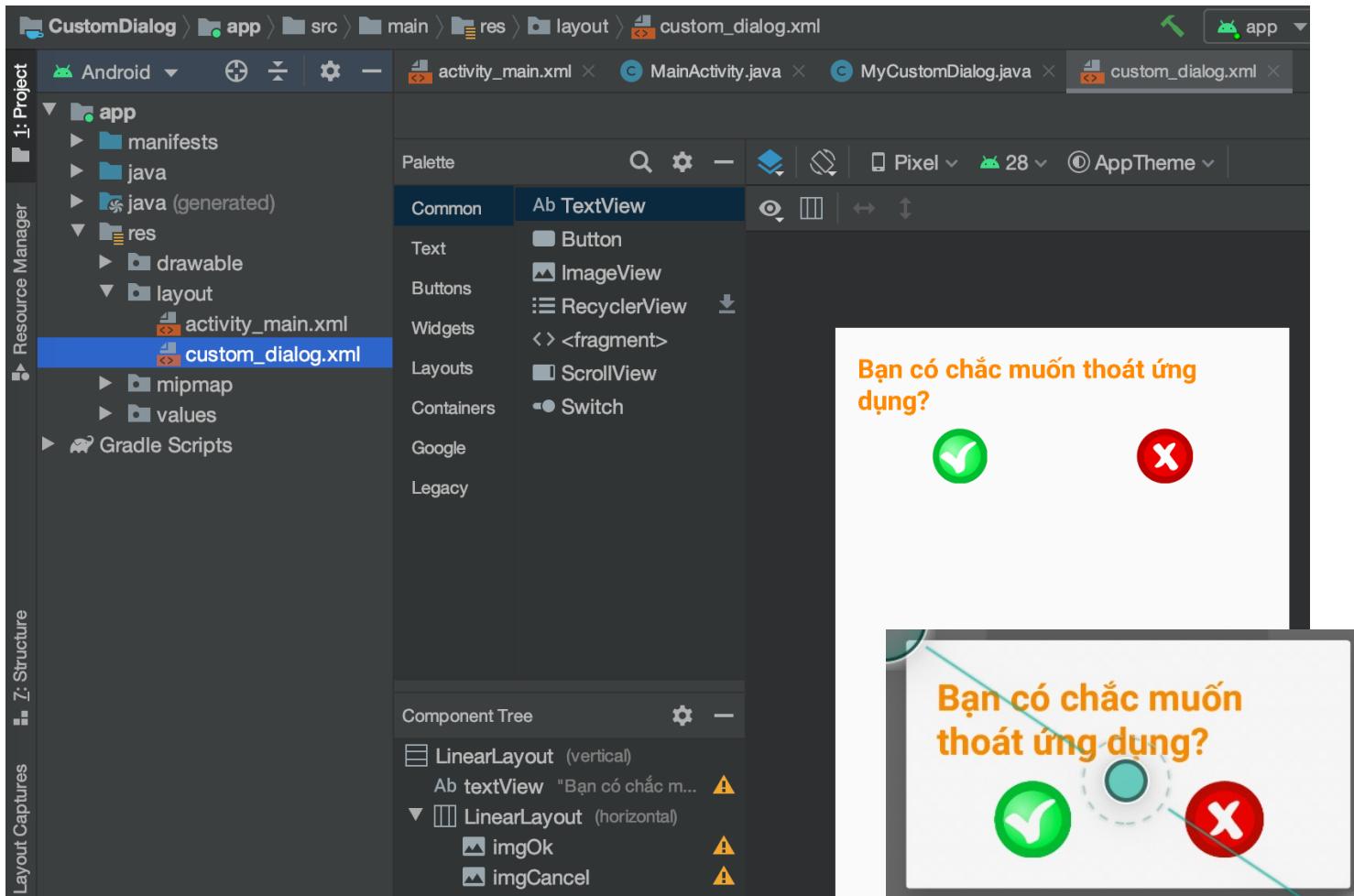
### »» Alert dialog

```
AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
builder.setTitle("Xác nhận thoát");
builder.setIcon(android.R.drawable.ic_dialog_info);
builder.setMessage("Bạn chắc chắn muốn đóng ứng dụng?");
builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        finish();
    }
});
builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        dialogInterface.dismiss();
    }
});
AlertDialog dialog = builder.create();
dialog.show();
```

## 2. Cơ chế thông báo

### »» Custom dialog

#### ➤ Thiết kế custom dialog



## 2. Cơ chế thông báo

### »» Custom dialog ➤ Hiển thị custom dialog (C1)

```
final Dialog dialog = new Dialog(MainActivity.this);
dialog.setContentView(R.layout.custom_dialog);
dialog.setCanceledOnTouchOutside(false);
ImageView imgOk = dialog.findViewById(R.id.imgOk);
imgOk.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        finish();
    }
});
ImageView imgCancel = dialog.findViewById(R.id.imgCancel);
imgCancel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        dialog.dismiss();
    }
});
dialog.show();
```

## 2. Cơ chế thông báo

### »» Custom dialog ➤ Hiển thị custom dialog (C2)

```
public class MyCustomDialog extends Dialog {  
  
    ImageView imgOk, imgCancel;  
    Activity context;  
  
    public MyCustomDialog(@NonNull Context context) {  
        super(context);  
        this.context = (Activity) context;  
        setContentView(R.layout.custom_dialog);  
        addViews();  
        setCanceledOnTouchOutside(false);  
        addEvents();  
    }  
}
```

## 2. Cơ chế thông báo

### »» Custom dialog ➤ Hiển thị custom dialog (C2)

```
private void addEvents() {
    imgOk.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            context.finish();
        }
    });
    imgCancel.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            dismiss();
        }
    });
}
private void addViews() {
    imgOk = findViewById(R.id.imgOk);
    imgCancel = findViewById(R.id.imgCancel);
}
```

## 2. Cơ chế thông báo

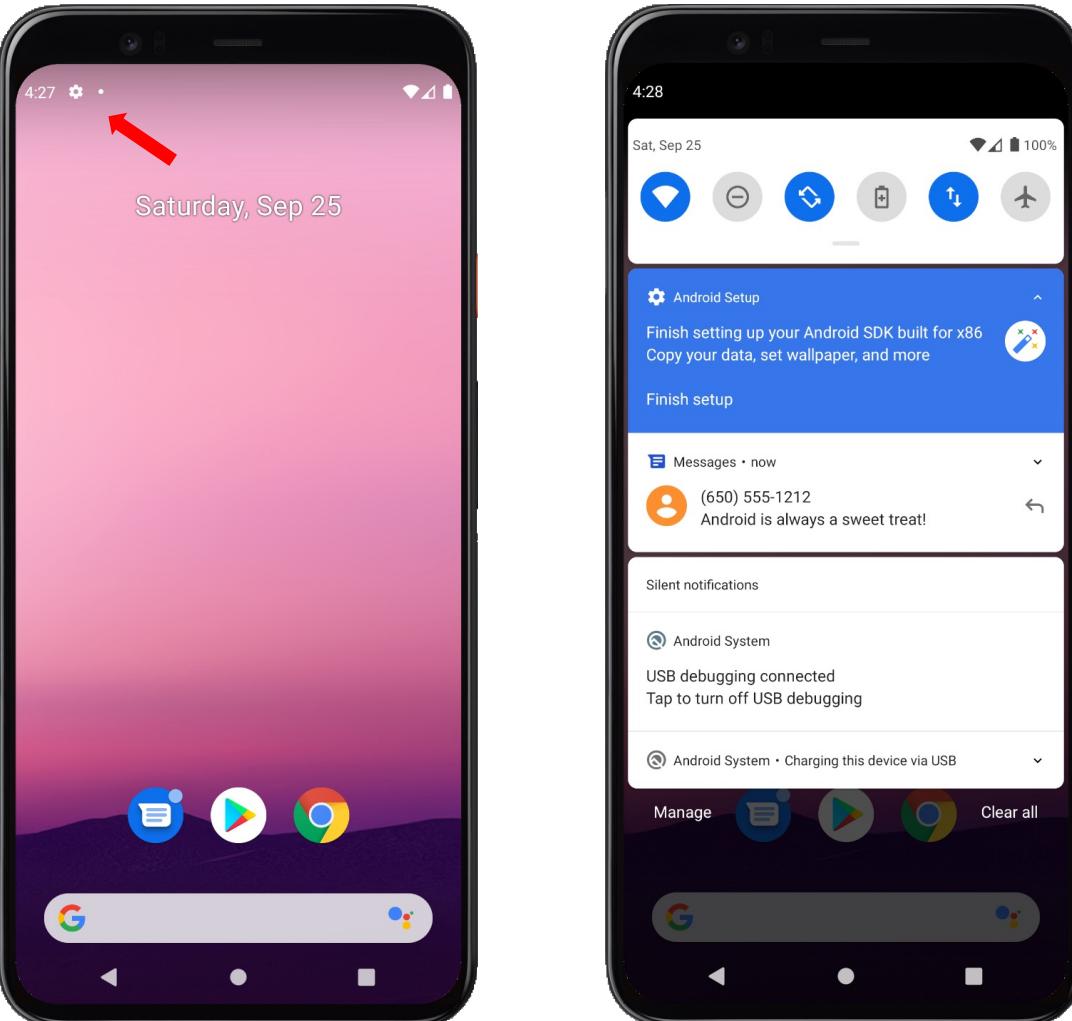
### »» Custom dialog ➤ Hiển thị custom dialog (C2)

```
MyCustomDialog myCustomDialog = new MyCustomDialog(MainActivity.this);  
myCustomDialog.show();
```



## 2. Cơ chế thông báo

### »» Notification



## 2. Cơ chế thông báo

### »» Notification ➤ Tạo notification

```
private void createNotificationChannels() {  
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){  
        NotificationChannel channel = new NotificationChannel(  
            CHANNEL_ID,  
            "MyChannel",  
            NotificationManager.IMPORTANCE_HIGH  
        );  
        channel.setDescription("This is My channel");  
  
        NotificationManager manager =  
            getSystemService(NotificationManager.class);  
        manager.createNotificationChannel(channel);  
    }  
}
```

\* Từ Android 8.0 (API level 26), notification phải được gửi theo Channel

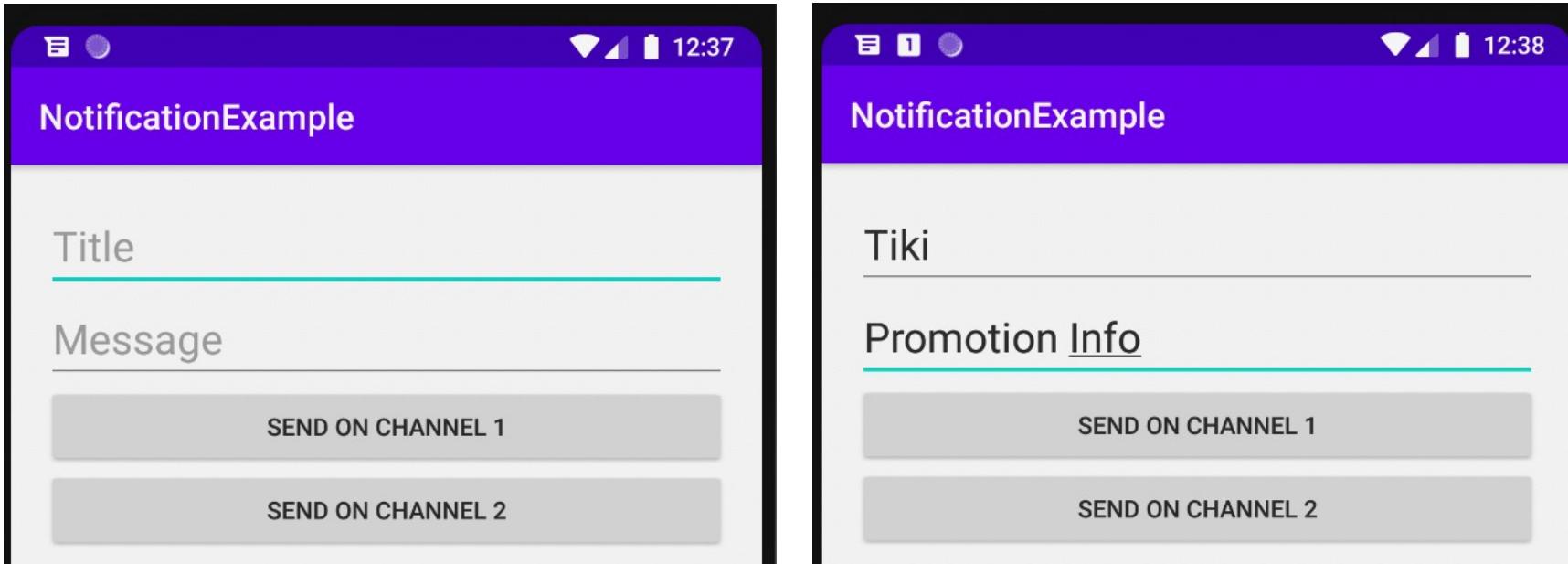
## 2. Cơ chế thông báo

### »» Notification ➤ Tạo notification

```
public void sendNotificationOnChannel(View view) {  
    Notification notification = new NotificationCompat.Builder(this,  
                    CHANNEL_ID)  
        .setSmallIcon(R.drawable.ic_one)  
        .setContentTitle("Tiêu đề")  
        .setContentText("Nội dung")  
        .setSound(RingtoneManager.getDefaultUri(  
                    RingtoneManager.TYPE_NOTIFICATION))  
        .setPriority(NotificationCompat.PRIORITY_HIGH)  
        .setCategory(NotificationCompat.CATEGORY_MESSAGE)  
        .build();  
  
    notificationManagerCompat.notify(1, notification);  
}
```

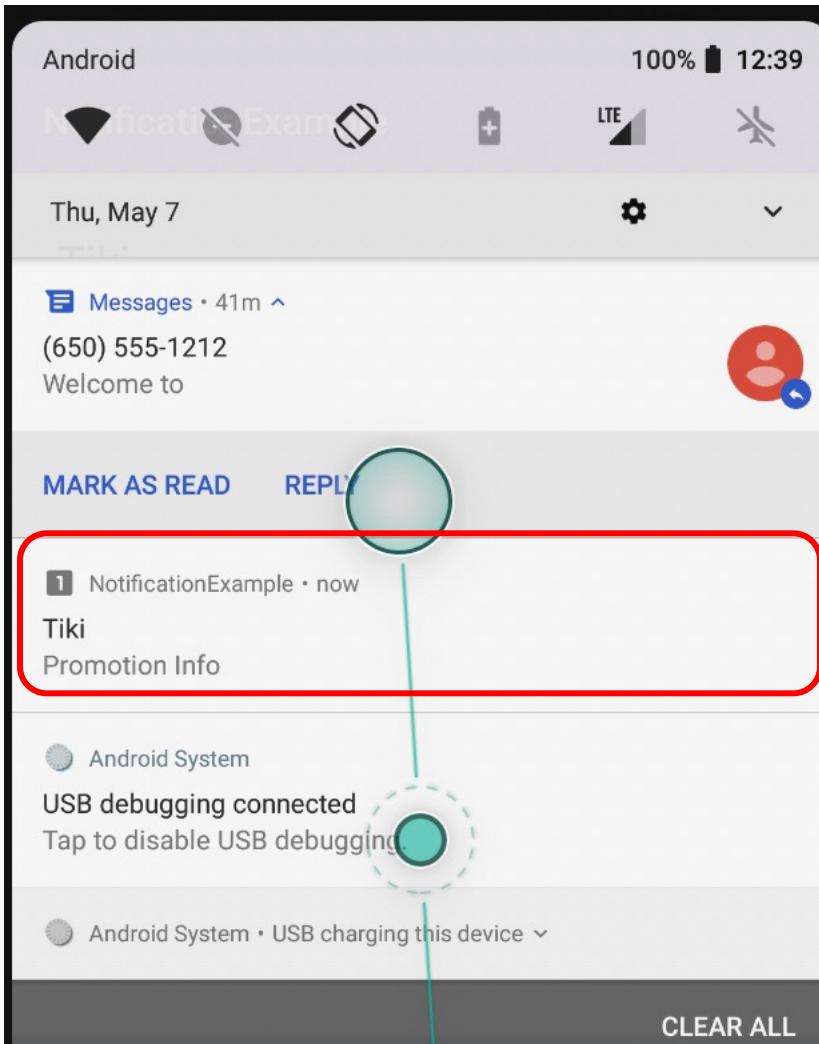
## 2. Cơ chế thông báo

### »» Notification ➤ Tạo notification



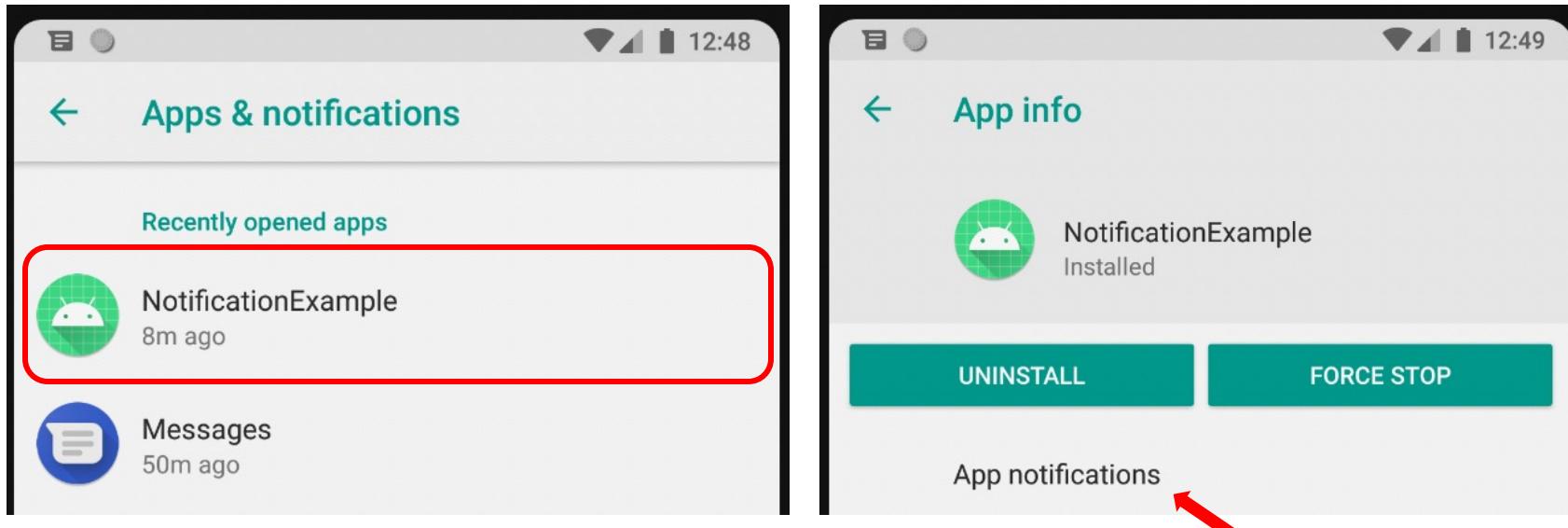
## 2. Cơ chế thông báo

### »» Notification ➤ Tạo notification



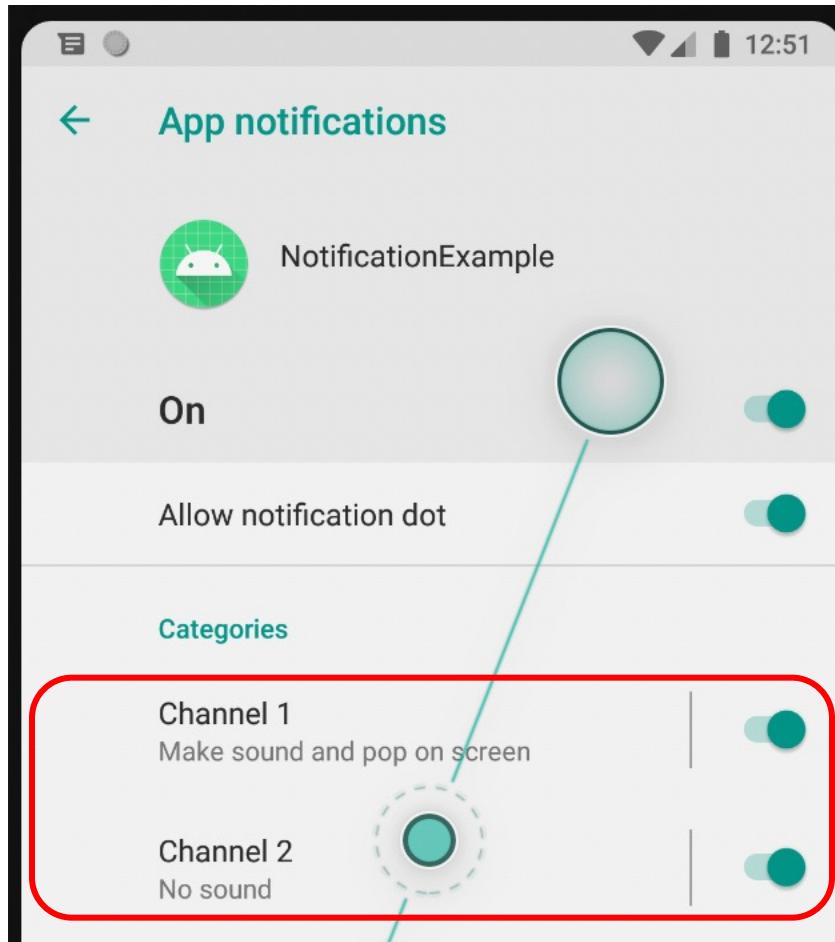
## 2. Cơ chế thông báo

»» Notification ➤ Thiết lập độ ưu tiên



## 2. Cơ chế thông báo

»» Notification ➤ Thiết lập độ ưu tiên



04

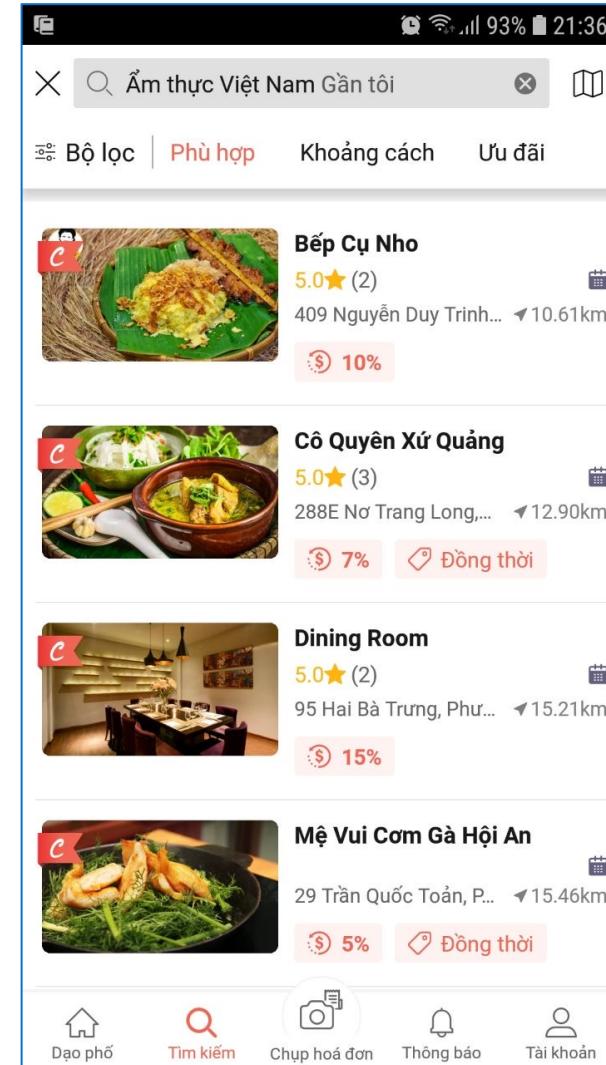
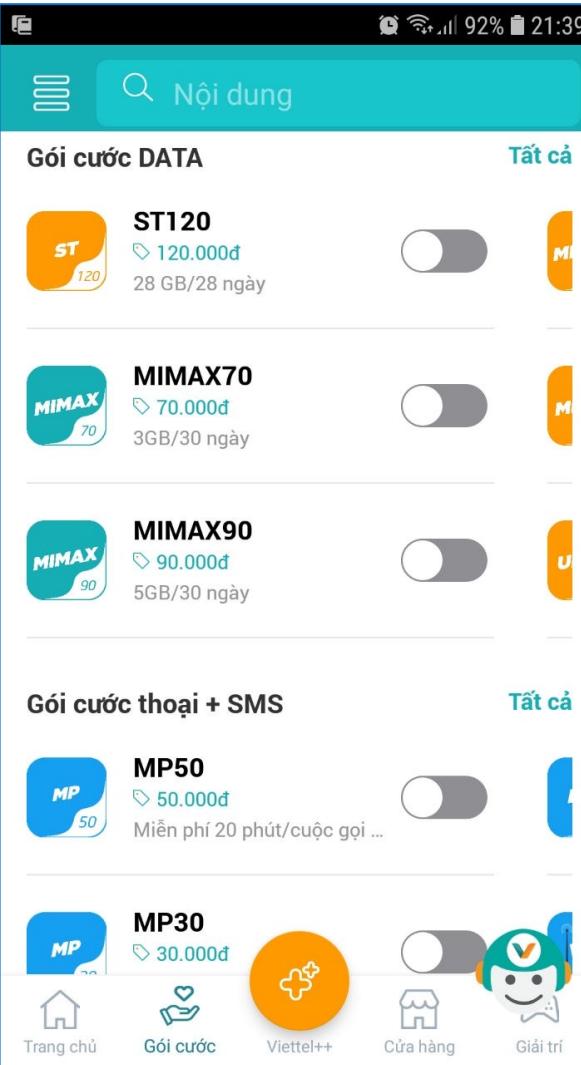
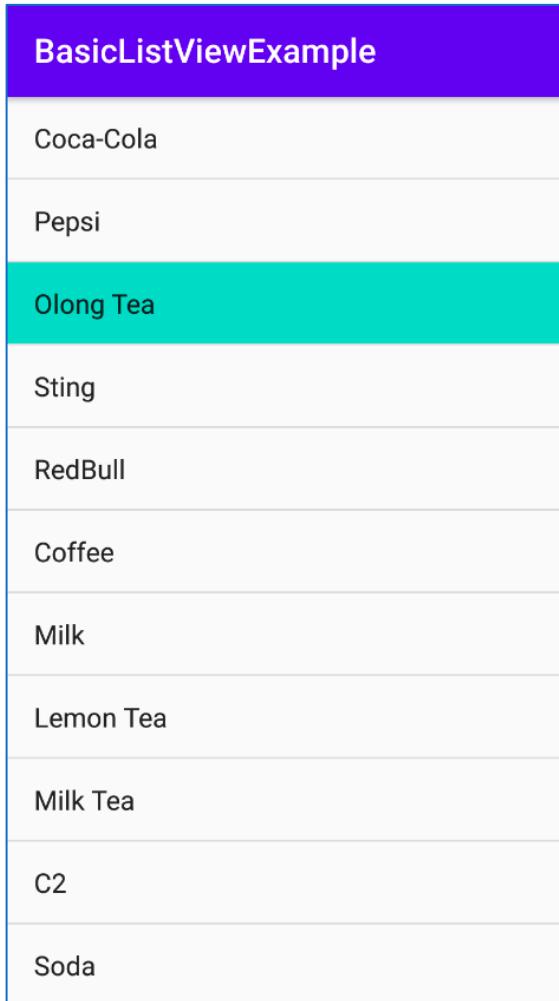
VIEW (P2)

# NỘI DUNG

- *ListView*
- *GridView*
- *RecyclerView*
- *Spinner*
- *AutocompleteTextView*
- *DatePicker, TimePicker*
- *TabHost*

# 1. ListView

✓ *ListView* là một dạng điều khiển cho phép hiển thị dữ liệu theo dạng dòng.



# 1. ListView

## ➤ Các bước sử dụng ListView

- ✓ **B1:** Tạo nguồn dữ liệu hiển thị trên ListView, có thể sử dụng ArrayList, mảng thông thường, mảng đối tượng, ...
- ✓ **B2:** Tạo ListView trên giao diện
- ✓ **B3:** Tạo đối tượng ArrayAdapter để liên kết giữa dữ liệu nguồn (mảng hoặc danh sách) và ListView

# 1. ListView

## ➤ Tạo ListView cơ bản

```
//1. Tạo nguồn dữ liệu cho ListView (data source)
String [] drinks = {"Coca-Cola", "Pepsi", "Olong Tea", "Sting", "RedBull",
    "Coffee", "Milk", "Lemon Tea", "Milk Tea", "C2", "Soda"};  
  
//2. Tham chiếu đến ListView trên giao diện qua Id
lvDrinks = findViewById(R.id.lvDrinks);  
  
//3. Gán data source vào đối tượng ArrayAdapter
arrayAdapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1, drinks);  
  
//4. Đưa dữ liệu lên ListView thông qua đối tượng ArrayAdapter
lvDrinks.setAdapter(arrayAdapter);
```

# 1. ListView

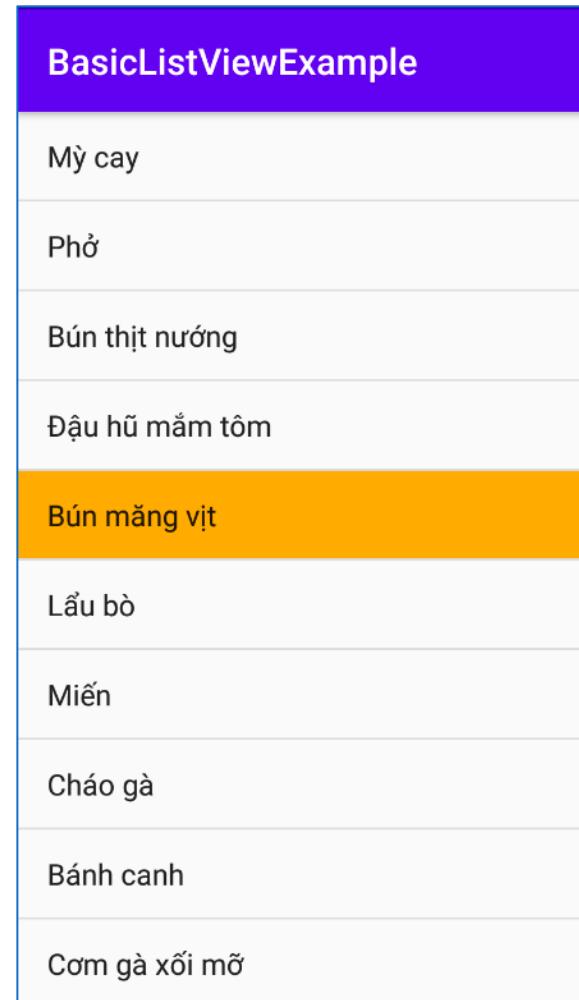
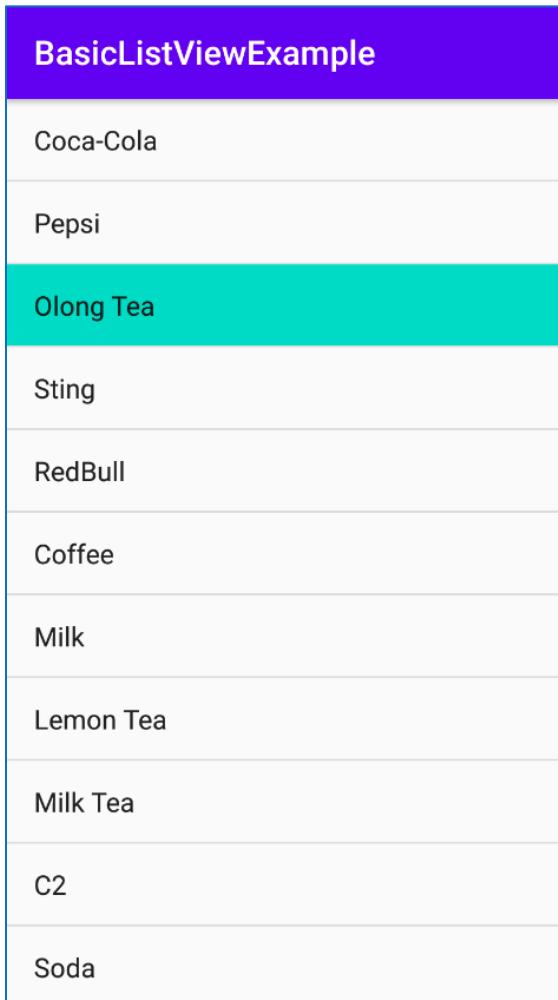
## ➤ Tạo ListView cơ bản

```
<resources>
    <string-array name="myFoods">
        <item>Mỳ cay</item>
        <item>Phở</item>
        <item>Bún thịt nướng</item>
        <item>Đậu hũ mắm tôm</item>
        <item>Bún măng vịt</item>
        <item>Lẩu bò</item>
        <item>Miến</item>
        <item>Cháo gà</item>
        <item>Bánh canh</item>
        <item>Com gà xối mỡ</item>
    </string-array>
</resources>
```

```
String [] foods = getResources().getStringArray(R.array.myFoods);
```

# 1. ListView

## ➤ Tạo ListView cơ bản



# 1. ListView

## ➤ Tạo ListView cơ bản (dữ liệu là ds đối tượng)

```
public class SanPham implements Serializable {  
    private String tenSP;  
    private String mauSac;  
    public SanPham() {}  
    public SanPham(String tenSP, String mauSac) {  
        this.tenSP = tenSP;  
        this.mauSac = mauSac;  
    }  
    public String getTenSP() { return tenSP; }  
    public void setTenSP(String tenSP) {this.tenSP = tenSP;}  
    public String getMauSac() {return mauSac;}  
    public void setMauSac(String mauSac) {this.mauSac = mauSac;}  
    @NotNull  
    @Override  
    public String toString() {  
        return this.tenSP + "\n" + this.mauSac;  
    }  
}
```

# 1. ListView

## ➤ Tạo ListView cơ bản (dữ liệu là ds đối tượng)

```
//Tham chiếu đến ListView trên giao diện qua Id  
lvSanPham = findViewById(R.id.lvSanPham);  
  
//Tạo đối tượng ArrayAdapter và gán dữ liệu đối tượng sản phẩm  
ArrayAdapter<SanPham> adapterSanPham;  
adapterSanPham = new ArrayAdapter<SanPham>(this,  
        android.R.layout.simple_list_item_1);  
adapterSanPham.add(new SanPham("T-Shirt", "Red"));  
adapterSanPham.add(new SanPham("Hat", "Blue"));  
adapterSanPham.add(new SanPham("Shoes", "Black"));  
  
//Nạp dữ liệu lên ListView thông qua đối tượng ArrayAdapter<SanPham>  
lvSanPham.setAdapter(adapterSanPham);
```

# 1. ListView

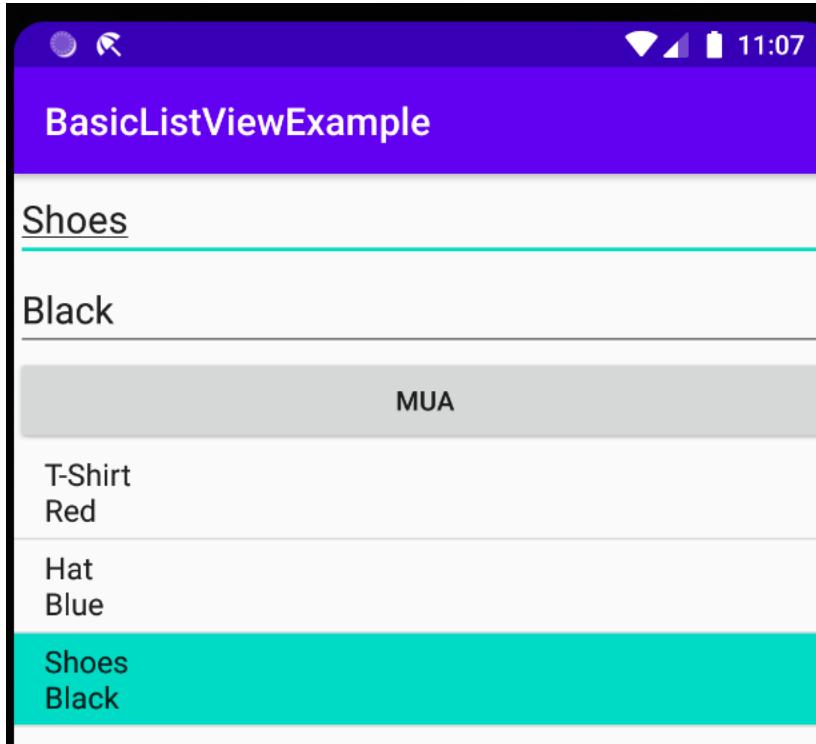
## ➤ Tạo ListView cơ bản (dữ liệu là ds đối tượng)

```
lvSanPham.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        SanPham sp = adapterSanPham.getItem(i);
        edtTenSP.setText(sp.getTenSP());
        edtMauSac.setText(sp.getMauSac());
    }
});

lvSanPham.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener()
{
    @Override
    public boolean onItemLongClick(AdapterView<?> adapterView, View view, int i, long l)
    {
        SanPham sp = adapterSanPham.getItem(i);
        adapterSanPham.remove(sp);
        return false;
    }
});
```

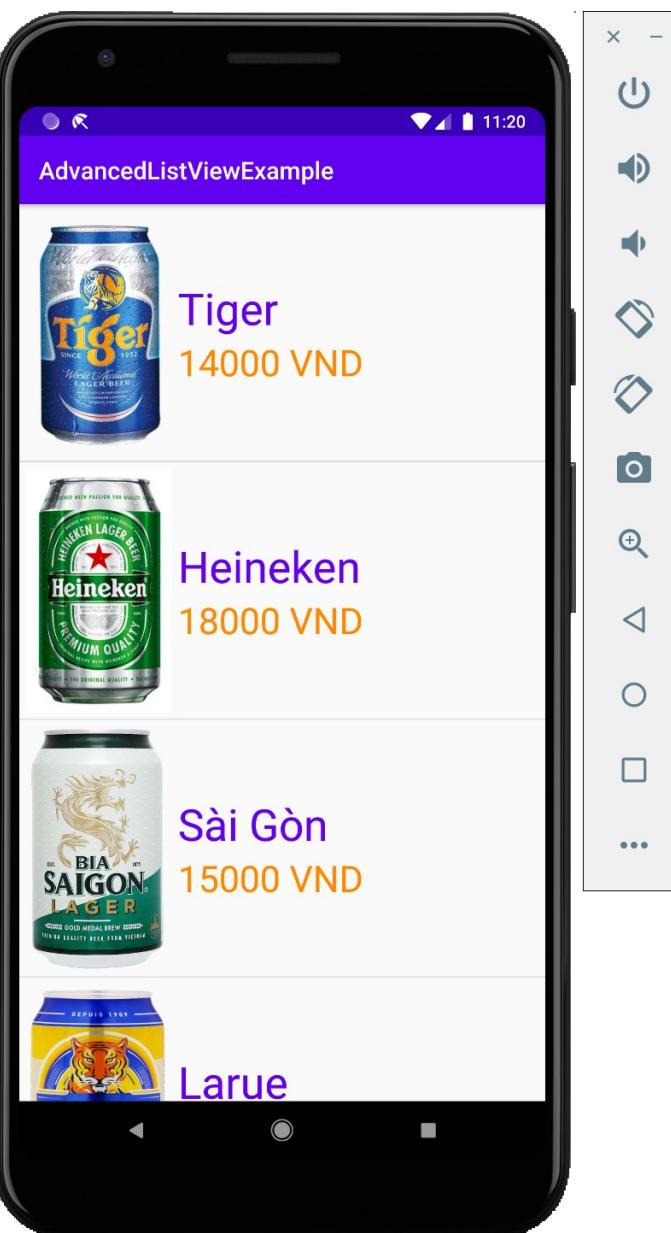
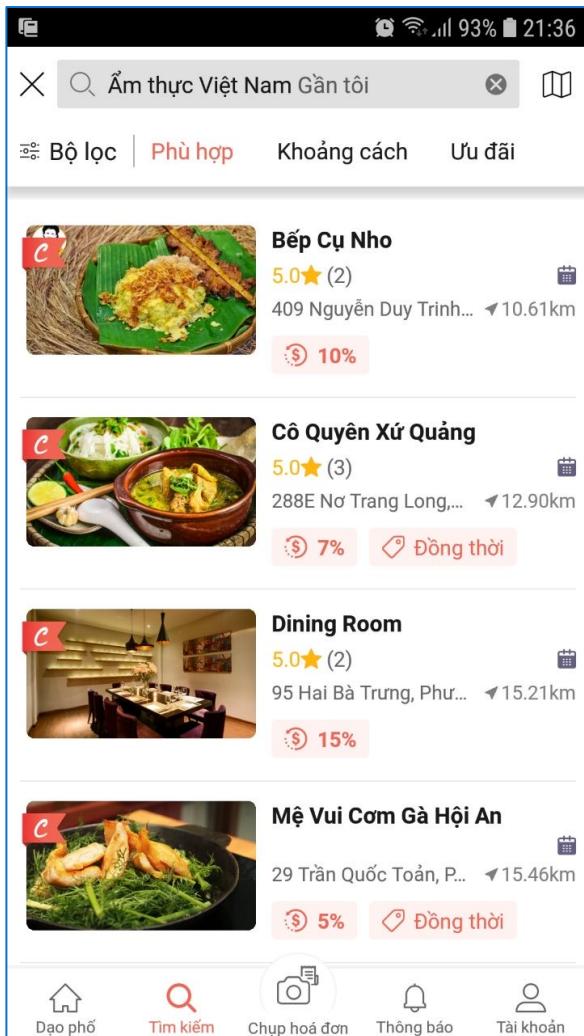
# 1. ListView

➤ Tạo ListView cơ bản (dữ liệu là ds đối tượng)



# 1. ListView

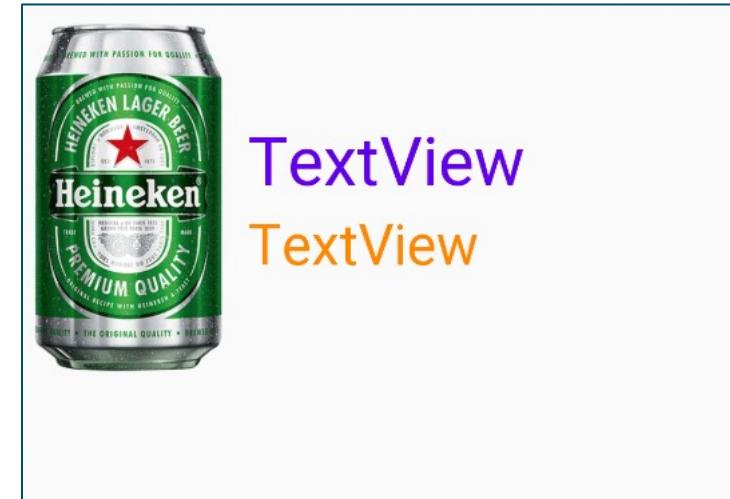
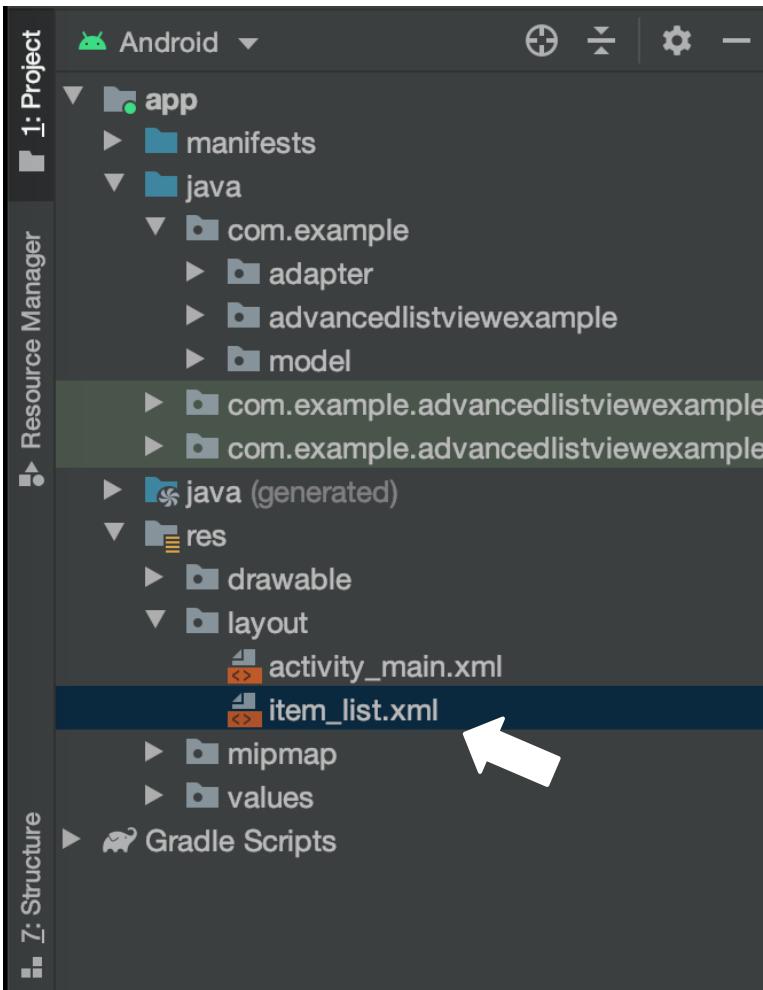
## ➤ Tạo ListView nâng cao



# 1. ListView

## ➤ Tạo ListView nâng cao

- ✓ Thiết kế layout cho một item của ListView



# 1. ListView

## ➤ Tạo ListView nâng cao

```
public class SanPham implements Serializable {  
    private int imageId;  
    private String ten;  
    private double gia;  
    public SanPham() {}  
    public SanPham(int imageId, String ten, double gia) {  
        this.imageId = imageId;  
        this.ten = ten;  
        this.gia = gia;  
    }  
    public int getImageId() {return imageId;}  
    public void setImageId(int imageId) {this.imageId = imageId;}  
    public String getTen() {return ten;}  
    public void setTen(String ten) {this.ten = ten;}  
    public double getGia() {return gia;}  
    public void setGia(double gia) {this.gia = gia;}  
}
```

# 1. ListView

## ➤ Tạo ListView nâng cao

- ✓ *Tạo lớp CustomAdapter kế thừa từ lớp ArrayAdapter*

```
public class SanPhamAdapter extends ArrayAdapter<SanPham> {  
    Activity context;  
    int resource;  
    public SanPhamAdapter(@NonNull Activity context, int resource) {  
        super(context, resource);  
        this.context = context;  
        this.resource = resource;  
    }  
    ...
```

# 1. ListView

## ➤ Tạo ListView nâng cao

- ✓ *Tạo lớp CustomAdapter kế thừa từ lớp ArrayAdapter*

```
@NonNull  
@Override  
public View getView(int position, @Nullable View convertView, @NonNull  
ViewGroup parent) {  
    LayoutInflator inflater = this.context.getLayoutInflater();  
    View customView = inflater.inflate(this.resource, null);  
  
    ImageView imgHinh = customView.findViewById(R.id.imgHinh);  
    TextView txtTen = customView.findViewById(R.id.txtTen);  
    TextView txtGia = customView.findViewById(R.id.txtGia);  
  
    SanPham sp = getItem(position);  
    imgHinh.setImageResource(sp.getImageId());  
    txtTen.setText(sp.getTen());  
    txtGia.setText(String.format("%,.0f", sp.getGia()) + " VND");  
  
    return customView;  
}
```

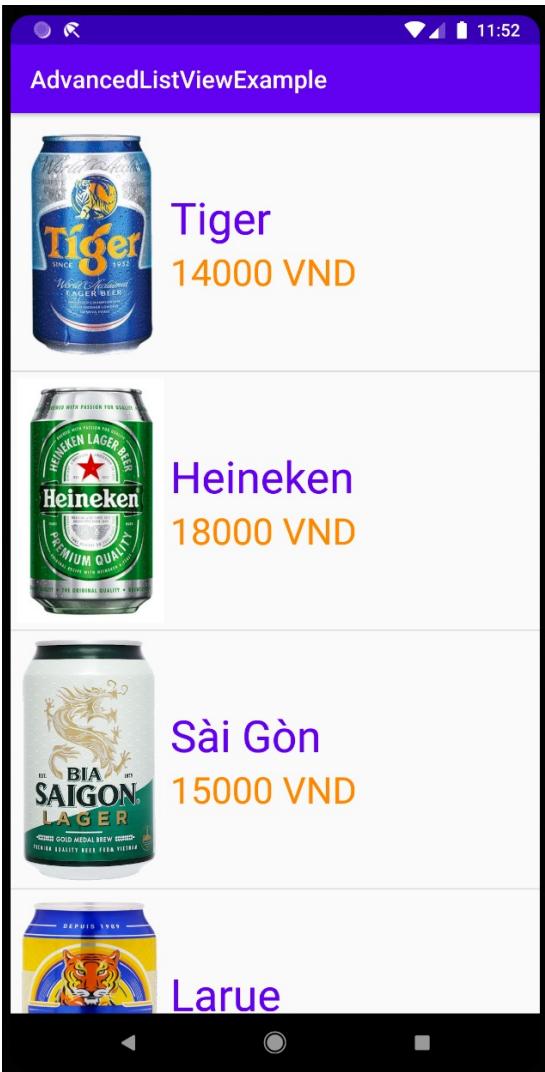
# 1. ListView

## ➤ Tạo ListView nâng cao

```
ListView lvSanPham;  
SanPhamAdapter sanPhamAdapter;  
  
lvSanPham = findViewById(R.id.lvSanPham);  
  
sanPhamAdapter = new SanPhamAdapter(MainActivity.this, R.layout.item_list);  
  
sanPhamAdapter.add(new SanPham(R.drawable.tiger, "Tiger", 14000));  
sanPhamAdapter.add(new SanPham(R.drawable.heiniken, "Heineken", 18000));  
sanPhamAdapter.add(new SanPham(R.drawable.saigon, "Sài Gòn", 15000));  
sanPhamAdapter.add(new SanPham(R.drawable.larue, "Larue", 12000));  
sanPhamAdapter.add(new SanPham(R.drawable.sapporo, "Sapporo", 17000));  
sanPhamAdapter.add(new SanPham(R.drawable.hanoi, "Hà Nội", 16000));  
sanPhamAdapter.add(new SanPham(R.drawable.beer333, "Bia 333", 15000));  
  
lvSanPham.setAdapter(sanPhamAdapter);
```

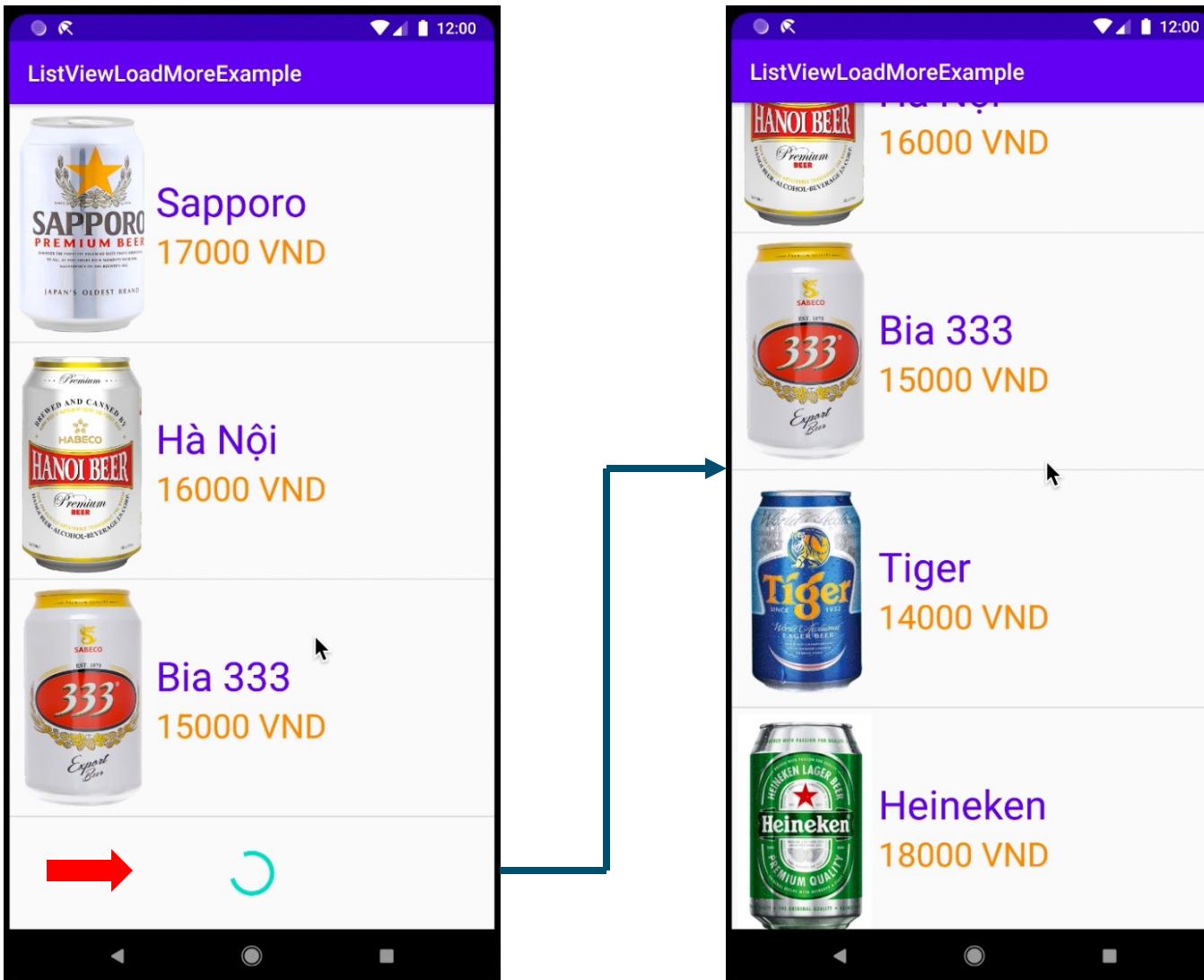
# 1. ListView

## ➤ Tạo ListView nâng cao



# 1. ListView

## ➤ Tạo ListView nâng cao – Load More



## 2. GridView

- ✓ *GridView là một dạng điều khiển cho phép hiển thị dữ liệu theo dạng bảng (dòng và cột).*
- ✓ *Cách thức tạo và sử dụng GridView hoàn toàn tương tự như ListView.*
- ✓ *Thiết lập số cột hiển thị trên GridView thông qua thuộc tính **numColumns**.*

```
<GridView  
    android:id="@+id/gvSanPham"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:listSelector="@android:color/holo_blue_bright"  
    android:numColumns="3" />
```

## 2. GridView

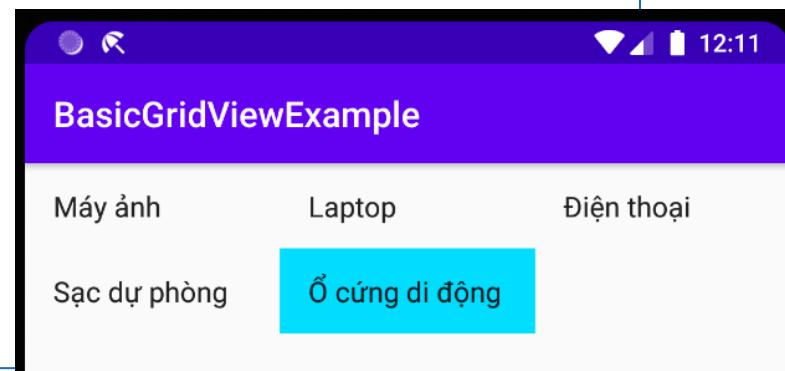
### ➤ Tạo GridView cơ bản

```
//1. Tạo nguồn dữ liệu cho GridView (data source)
ArrayList<String>dsSP = new ArrayList<>();
dsSP.add("Máy ảnh"); dsSP.add("Laptop"); dsSP.add("Điện thoại");
dsSP.add("Sạc dự phòng"); dsSP.add("Ổ cứng di động");

//2. Tham chiếu đến GridView trên giao diện qua Id
gvSanPham = findViewById(R.id.gvSanPham);

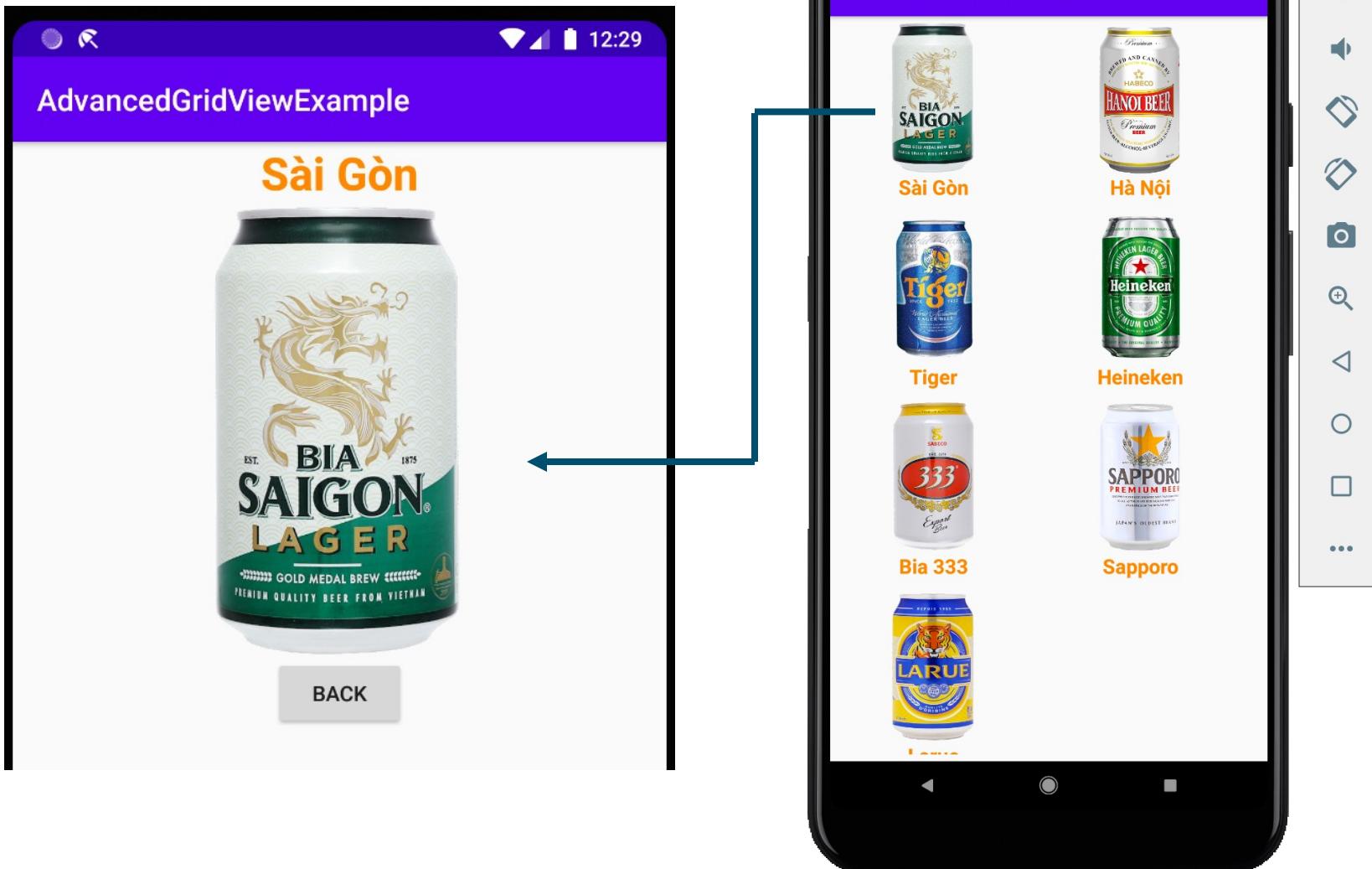
//3. Gán data source vào đối tượng ArrayAdapter
arrayAdapter = new ArrayAdapter<String>(MainActivity.this,
    android.R.layout.simple_list_item_1);
arrayAdapter.addAll(dsSP);

//4. Đưa dữ liệu lên GridView thông
// qua đối tượng ArrayAdapter
gvSanPham.setAdapter(arrayAdapter);
```



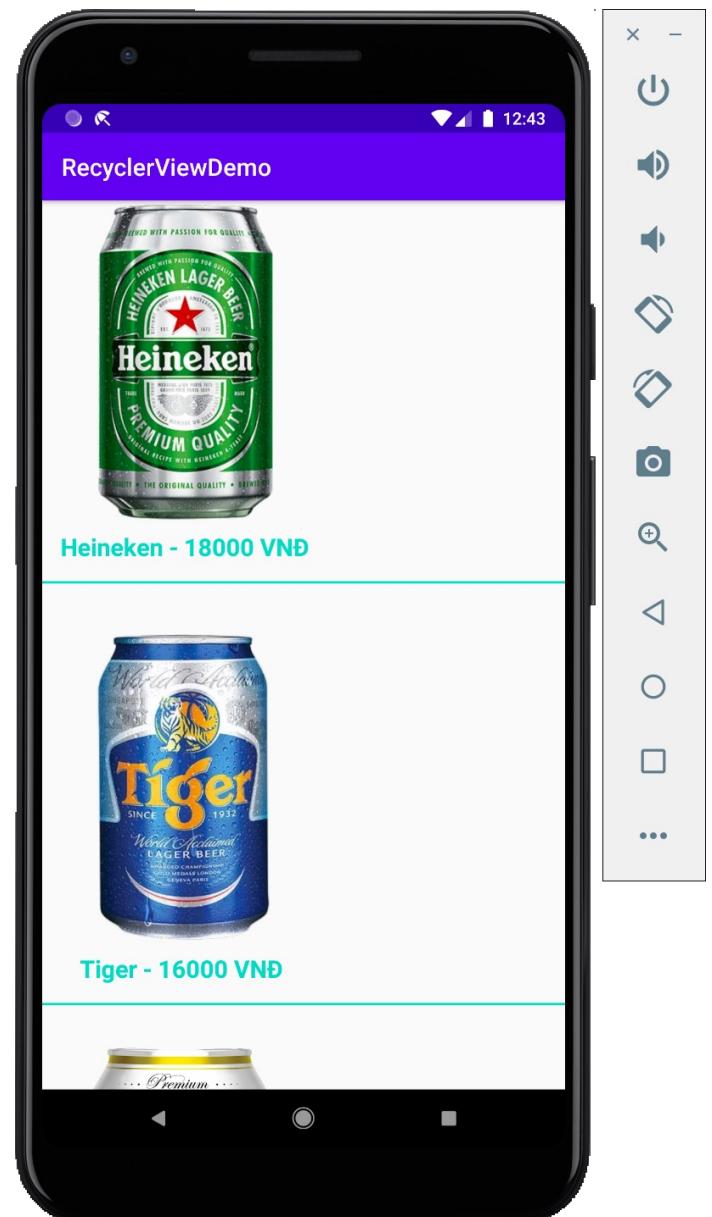
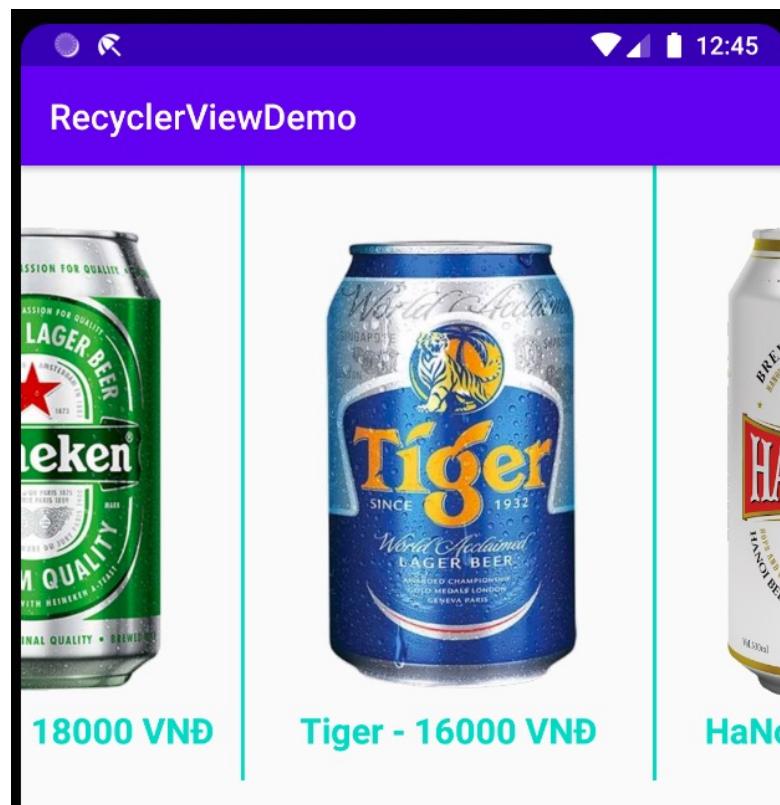
## 2. GridView

### ➤ Tạo GridView nâng cao



### 3. RecyclerView

- ✓ *RecyclerView* là một dạng điều khiển cho phép hiển thị dữ liệu dạng danh sách tương tự như *ListView* nhưng cho phép lập trình viên tùy biến sâu về giao diện cũng như cách thức hiển thị.



### 3. RecyclerView

#### ➤ Tạo RecyclerView

```
//1. Tạo CustomAdapter
public class SanPhamAdapter extends
RecyclerView.Adapter<SanPhamAdapter.ViewHolder>{ ... }

//2. Tạo ViewHolder tham chiếu các view trên giao diện cho một item
public class ViewHolder extends RecyclerView.ViewHolder{
    ImageView imgPhoto;
    TextView txtInfo;
    public ViewHolder(@NonNull View itemView) {
        super(itemView);
        imgPhoto = itemView.findViewById(R.id.imgPhoto);
        txtInfo = itemView.findViewById(R.id.txtInfo);
        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                RemoveItem(getAdapterPosition());
                Toast.makeText(context, "Remove " + txtInfo.getText(),
                    Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

### 3. RecyclerView

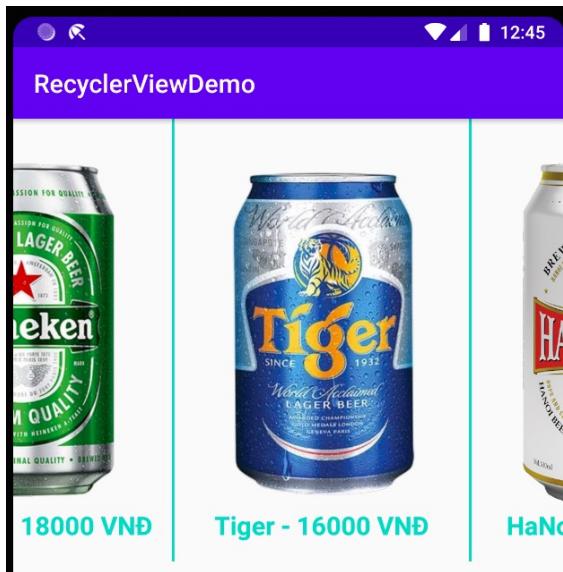
#### ➤ Tạo RecyclerView

```
//3. Nạp item layout lên ViewHolder  
@NonNull  
@Override  
public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
    LayoutInflator layoutInflater =  
        LayoutInflator.from(parent.getContext());  
    View customItemView =  
        layoutInflater.inflate(R.layout.custom_item, parent, false);  
    return new ViewHolder(customItemView);  
}
```

### 3. RecyclerView

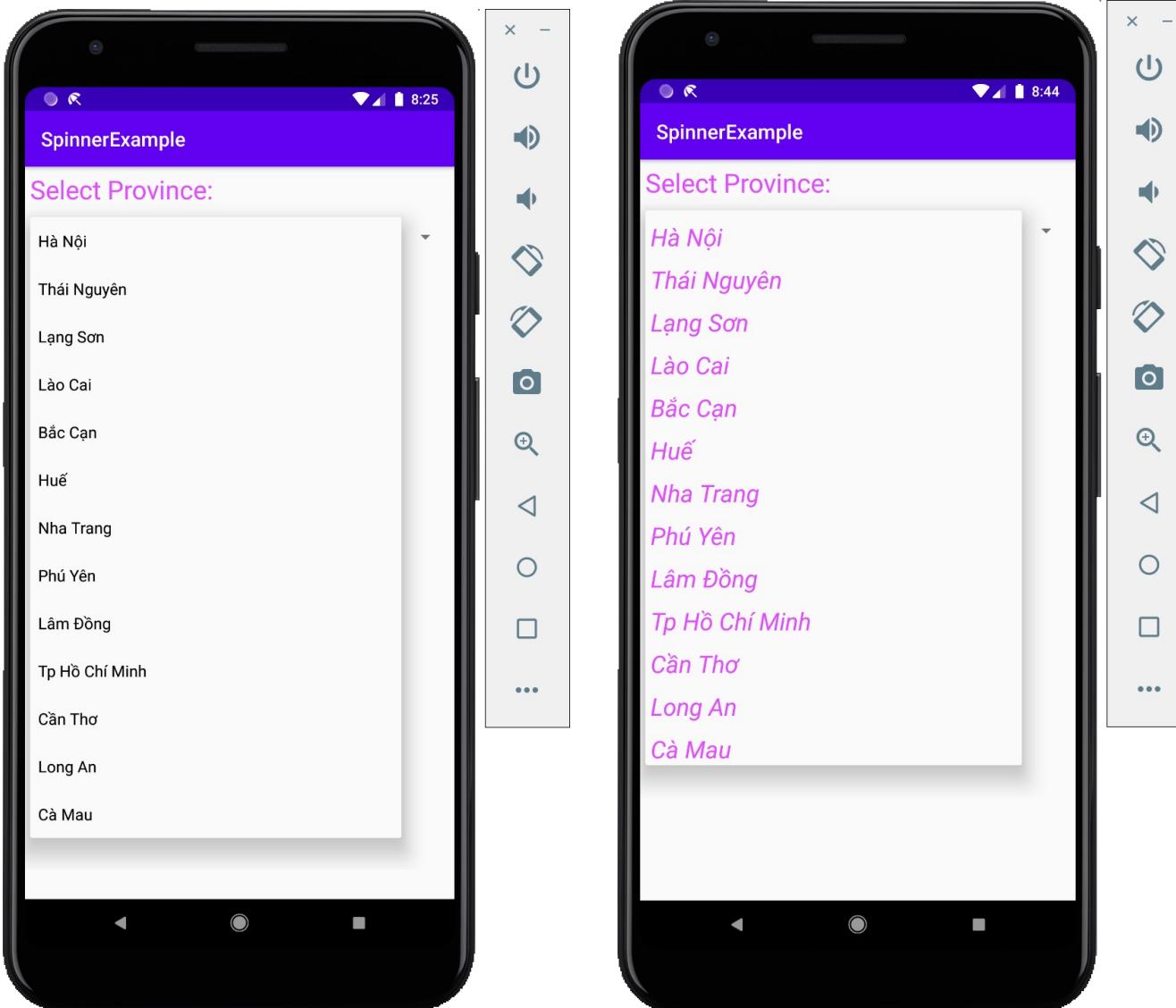
#### ➤ Tạo RecyclerView

```
//4. Binding dữ liệu lên ViewHolder  
@Override  
public void onBindViewHolder(@NonNull ViewHolder holder, int position)  
{  
    holder.imgPhoto.setImageResource(dsSanPham.get(position).getHinh());  
    holder.txtInfo.setText(dsSanPham.get(position).getTen() + " - " +  
        String.format("%.0f", dsSanPham.get(position).getGia()) + " VNĐ");  
}
```



## 4. Spinner

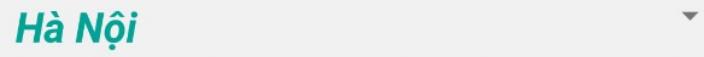
- ✓ *Spinner* là một dạng điều khiển tương tự như *ComboBox (C#)*, *JComboBox (Java)*



## 4. Spinner

### ➤ Tạo spinner

```
spinnerProvince = findViewById(R.id.spinnerProvince);
adapter = new ArrayAdapter<>(
    MainActivity.this, android.R.layout.spinner_item);
```

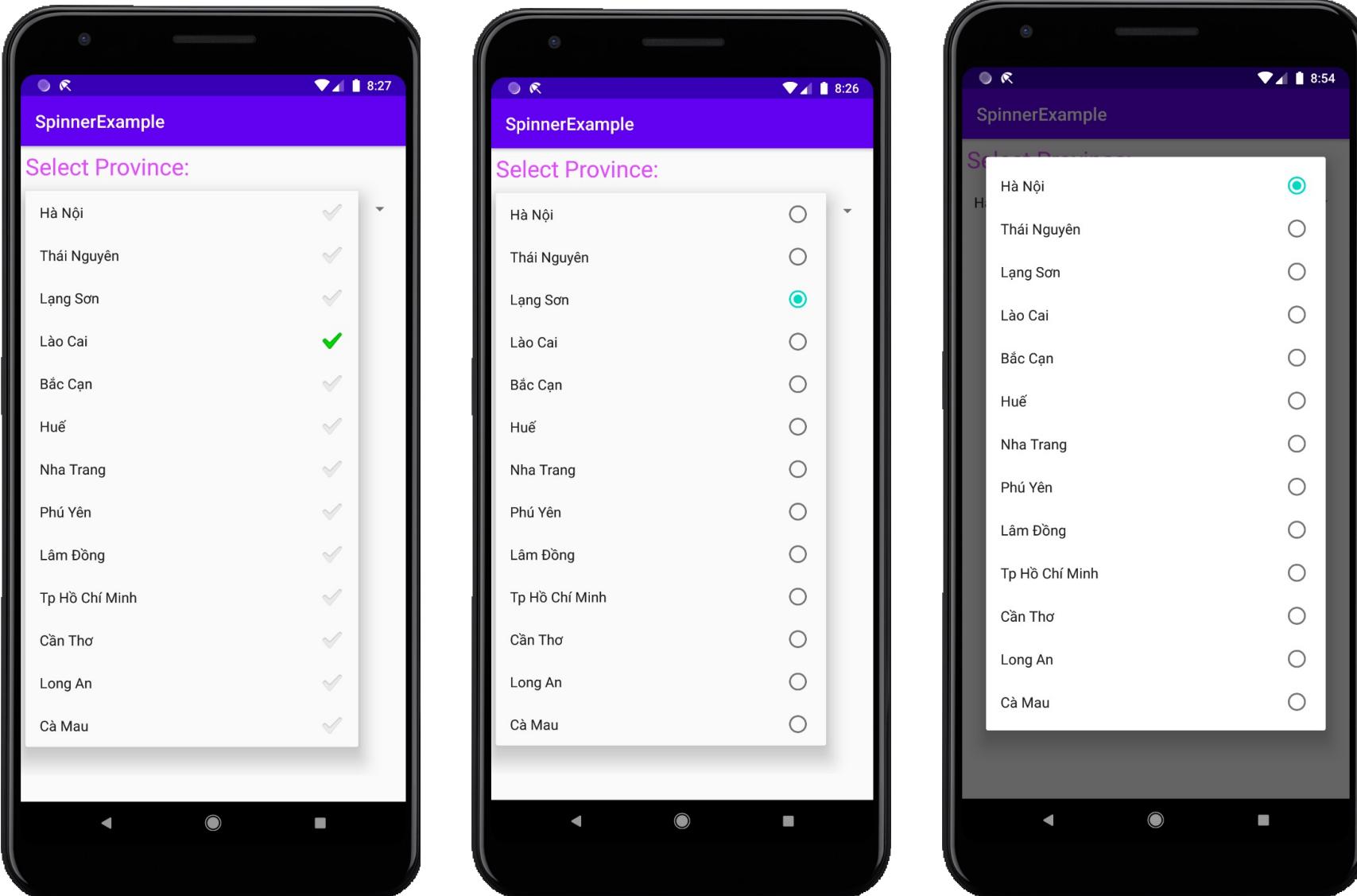


```
adapter.setDropDownViewResource(android.R.layout.spinner_list);
```



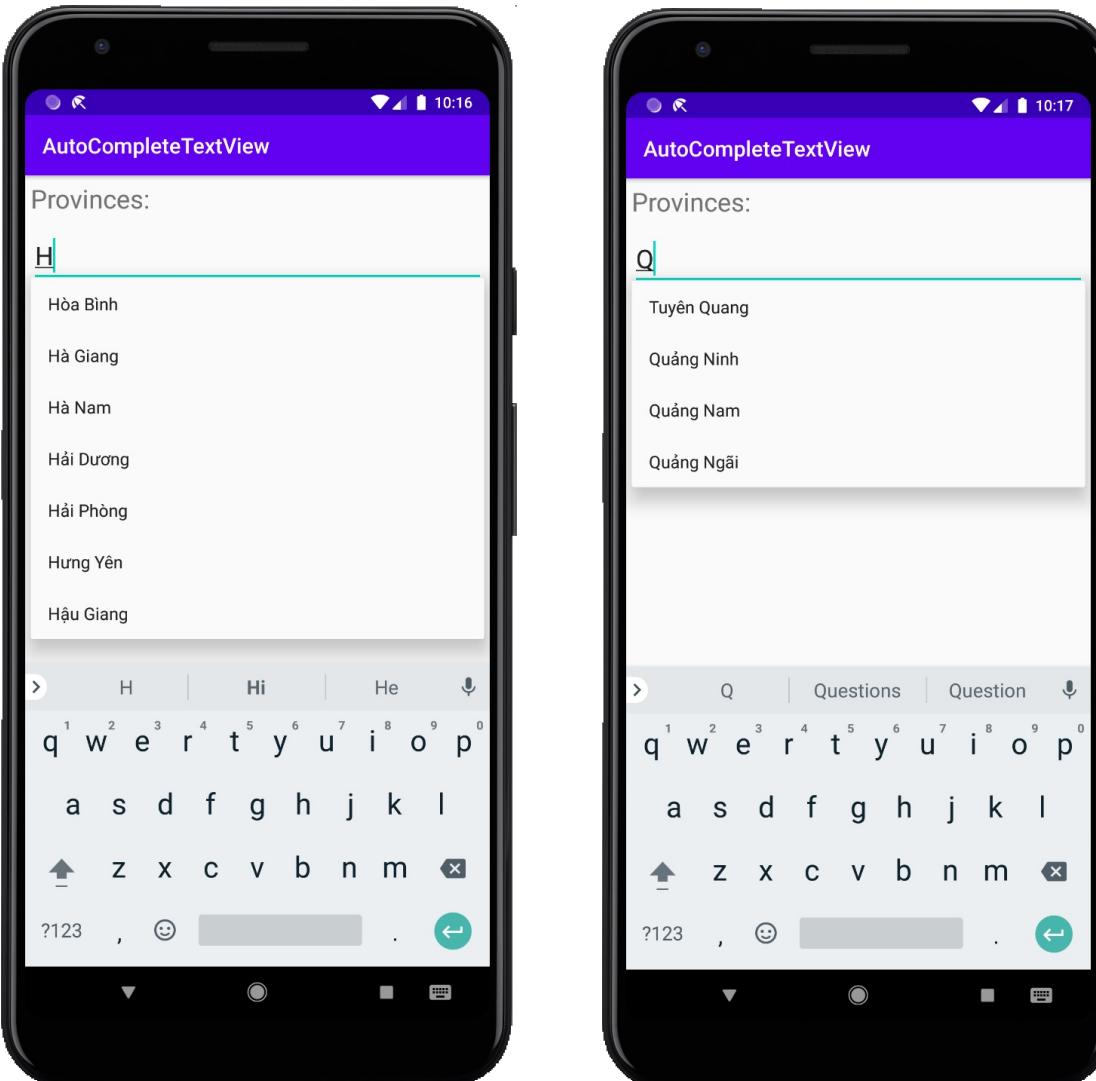
```
spinnerProvince.setAdapter(adapter);
```

## 4. Spinner



## 5. AutoCompleteTextView

- ✓ *AutoCompleteTextView* là một dạng điều khiển hỗ trợ gợi ý người dùng nhập liệu.



## 5. AutoCompleteTextView

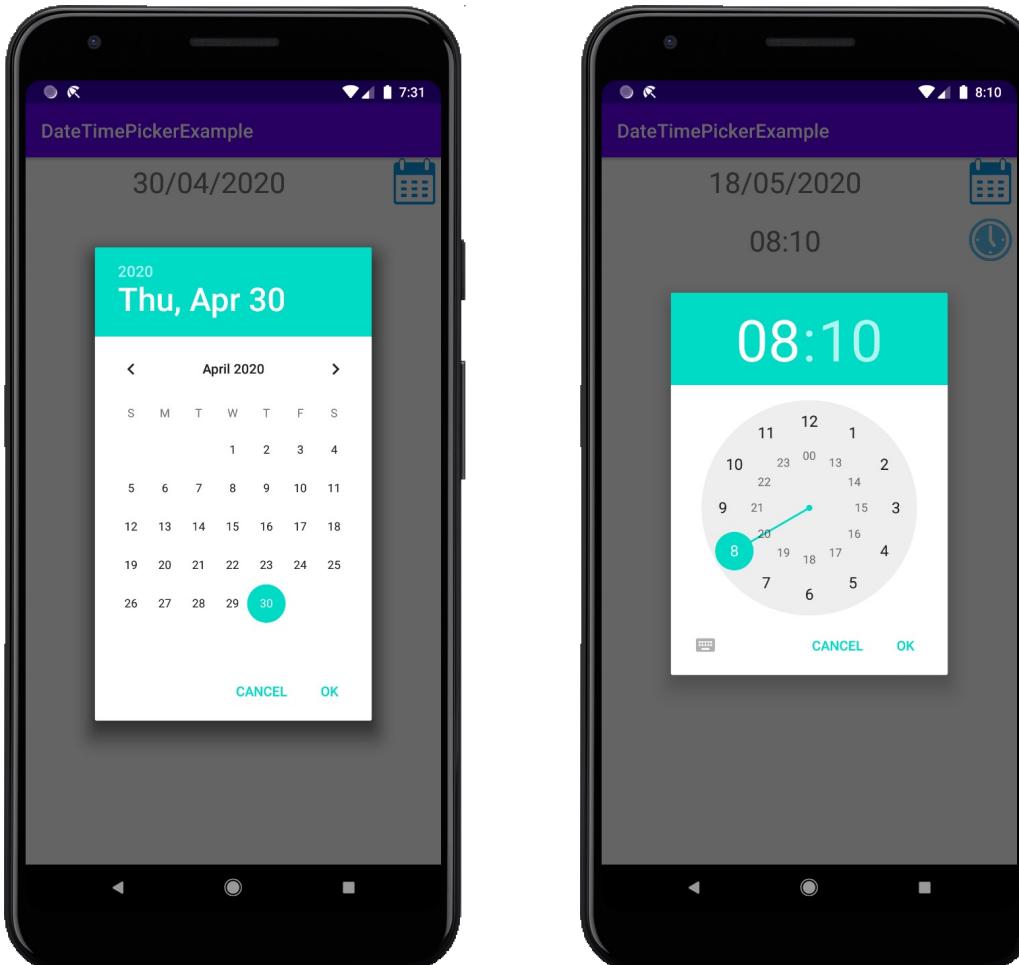
### ➤ Tạo AutoCompleteTextView

```
<AutoCompleteTextView  
    android:id="@+id/autoCompleteTextView"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:textSize="24sp"  
    android:layout_margin="5dp"  
    android:completionThreshold="1"/>
```

```
completeTextViewProvince =  
    findViewById(R.id.autoCompleteTextView);  
  
adapter = new ArrayAdapter<>(MainActivity.this,  
    android.R.layout.simple_list_item_1);  
adapter.addAll(getResources().getStringArray(R.array.provinces_name));  
  
completeTextViewProvince.setAdapter(adapter);
```

## 6. DatePicker, TimePicker

- ✓ **DatePicker** là điều khiển hỗ trợ người dùng chọn ngày tháng năm.
- ✓ **TimePicker** là điều khiển hỗ trợ người dùng chọn giờ, phút.



## 6. DatePicker, TimePicker

### ➤ Tạo DatePicker

```
Calendar calendarDate = Calendar.getInstance();
SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
DatePickerDialog.OnDateSetListener callBack = new
    DatePickerDialog.OnDateSetListener() {
    @Override
    public void onDateSet(DatePicker datePicker, int i, int i1, int i2) {
        calendarDate.set(Calendar.YEAR, i);
        calendarDate.set(Calendar.MONTH, i1);
        calendarDate.set(Calendar.DAY_OF_MONTH, i2);
        txtDate.setText(dateFormat.format(calendarDate.getTime()));
    }
};

DatePickerDialog datePickerDialog = new DatePickerDialog(
    MainActivity.this,
    callBack,
    calendarDate.get(Calendar.YEAR),
    calendarDate.get(Calendar.MONTH),
    calendarDate.get(Calendar.DAY_OF_MONTH));
datePickerDialog.show();
```

## 6. DatePicker, TimePicker

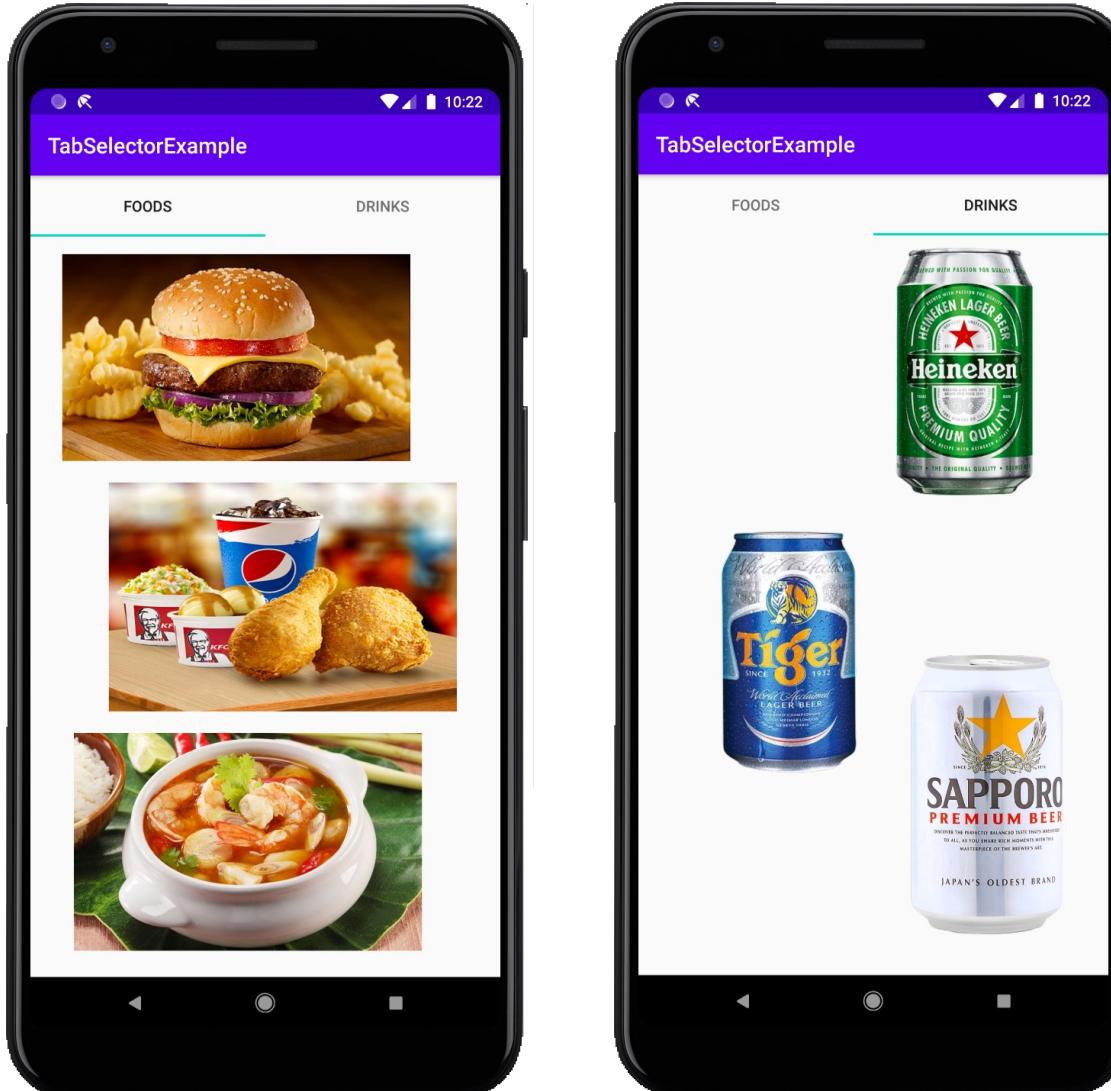
### ➤ Tạo TimePicker

```
Calendar calendarTime = Calendar.getInstance();
SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm");
TimePickerDialog.OnTimeSetListener callBack = new
    TimePickerDialog.OnTimeSetListener() {
    @Override
    public void onTimeSet(TimePicker timePicker, int i, int i1) {
        calendarTime.set(Calendar.HOUR, i);
        calendarTime.set(Calendar.MINUTE, i1);
        txtTime.setText(timeFormat.format(calendarTime.getTime()));
    }
};

TimePickerDialog timePickerDialog = new TimePickerDialog(
    MainActivity.this,
    callBack,
    calendarTime.get(Calendar.HOUR),
    calendarTime.get(Calendar.MINUTE),
    true);
timePickerDialog.show();
```

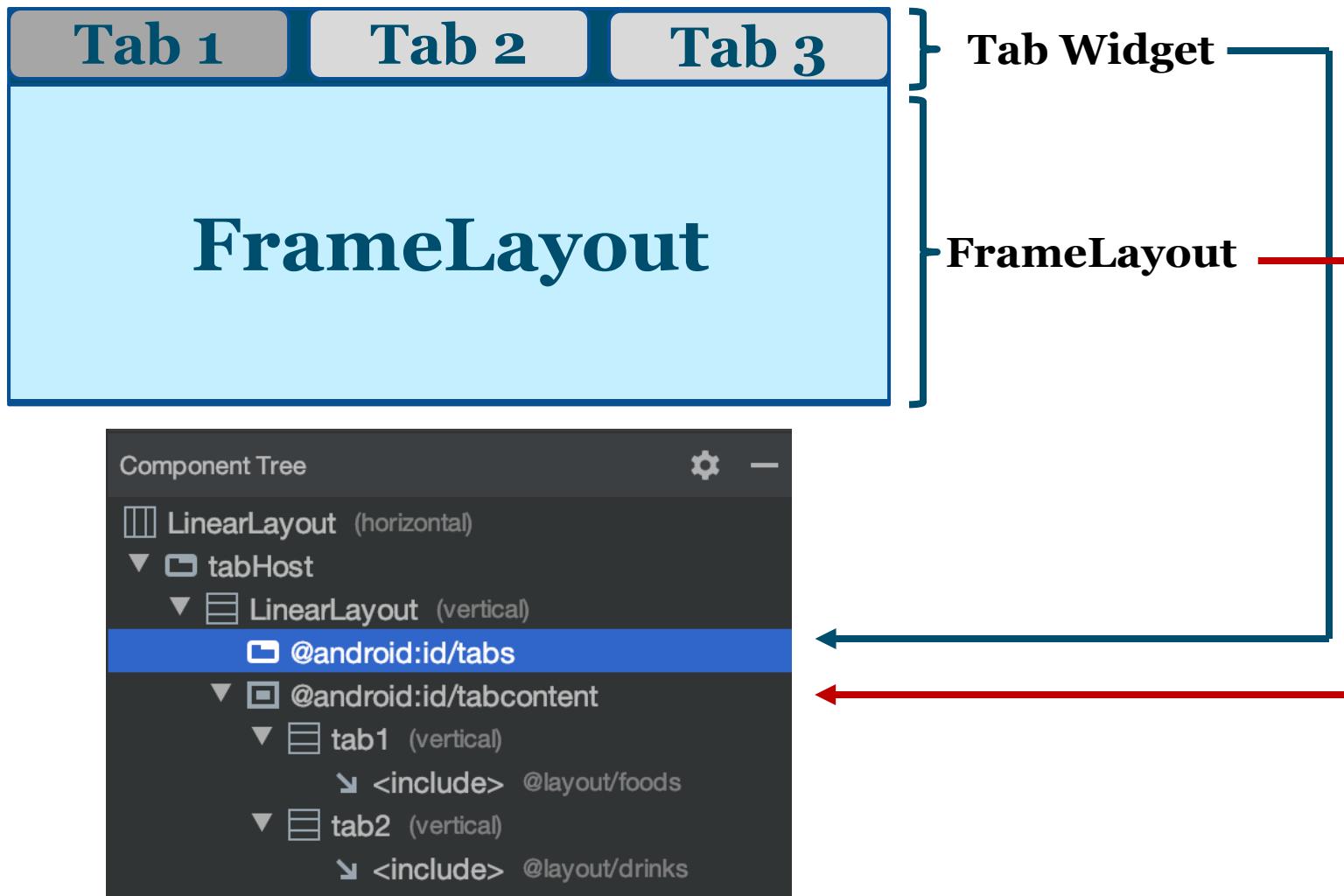
## 7. TabHost

- ✓ **TabHost** là dạng điều khiển tương tự Tab control trong C#, Java. TabHost cho phép hiển thị các giao diện khác nhau tại một vùng nhất định trên màn hình.



## 7. TabHost

### ➤ Cấu trúc TabHost



## 7. TabHost

### ➤ Tạo TabHost

```
tabHost = findViewById(R.id.tabHost);
tabHost.setup();

//Tạo tab 1
TabHost.TabSpec tab1 = tabHost.newTabSpec("tab1");
tab1.setContent(R.id.tab1);
tab1.setIndicator("Foods");
tabHost.addTab(tab1);

//Tạo tab 2
TabHost.TabSpec tab2 = tabHost.newTabSpec("tab2");
tab2.setContent(R.id.tab2);
tab2.setIndicator("Drinks");
tabHost.addTab(tab2);
```

## 7. TabHost

### ➤ Tạo TabHost

```
tabHost.setOnTabChangedListener(new
    TabHost.OnTabChangeListener() {
        @Override
        public void onTabChanged(String s) {
            Toast.makeText(MainActivity.this,
                "Tab " + s + " selected, index: " + tabHost.getCurrentTab(),
                Toast.LENGTH_LONG).show();
        }
    });
});
```

05

ACTIVITY & INTENT

# NỘI DUNG

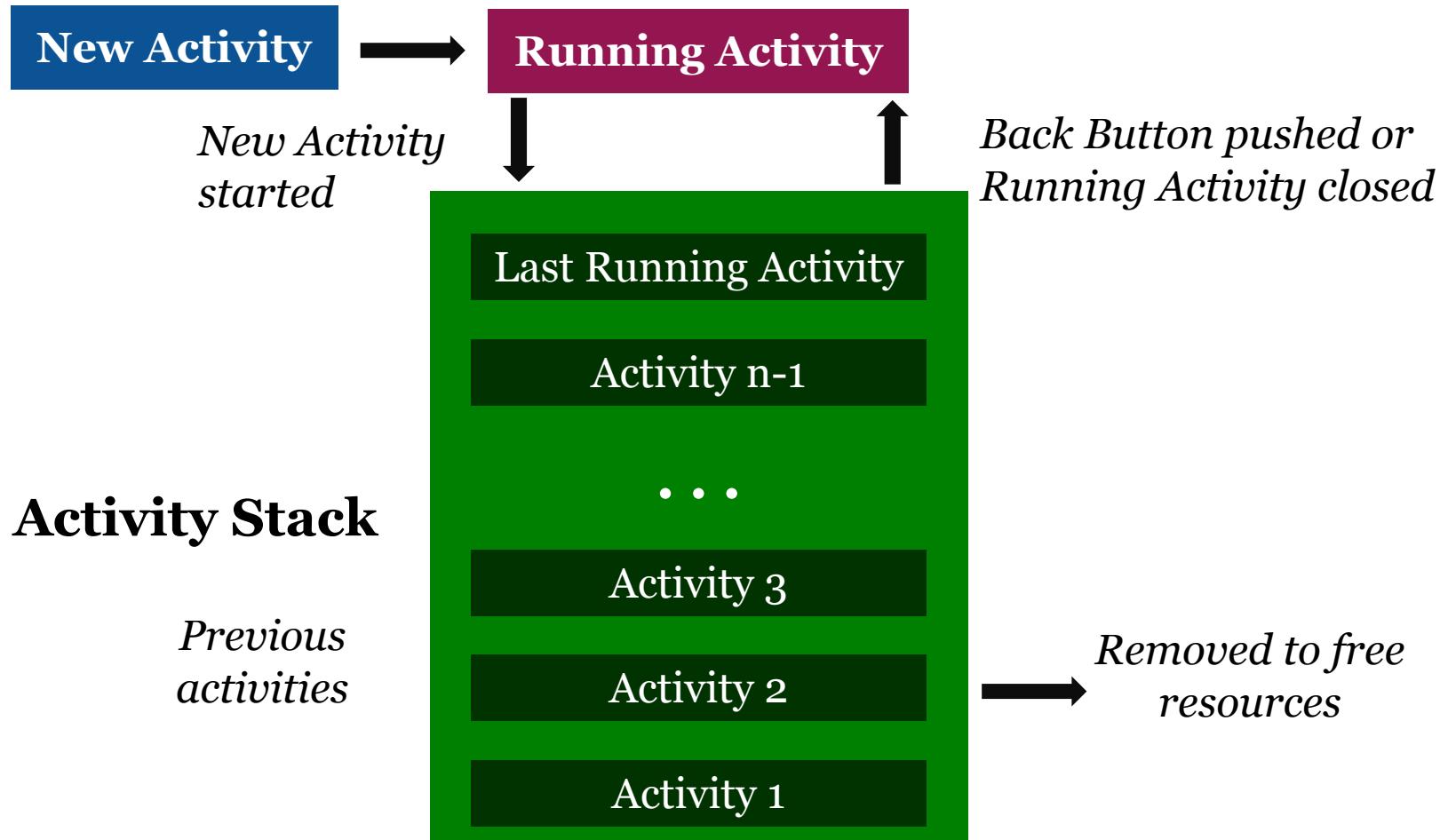
1. Activity
2. Intent
3. Cơ chế truyền và nhận dữ liệu giữa các Activity
4. Implicit intent

## 1. Activity

- Trong một ứng dụng có thể có nhiều Activity, mỗi Activity có thể hiểu như một màn hình ứng dụng.
- Mỗi Activity sẽ có vòng đời riêng độc lập hoàn toàn so với Activity khác.
- Mỗi Activity được triệu gọi trong ứng dụng *bắt buộc phải được khai báo trong Manifest*.

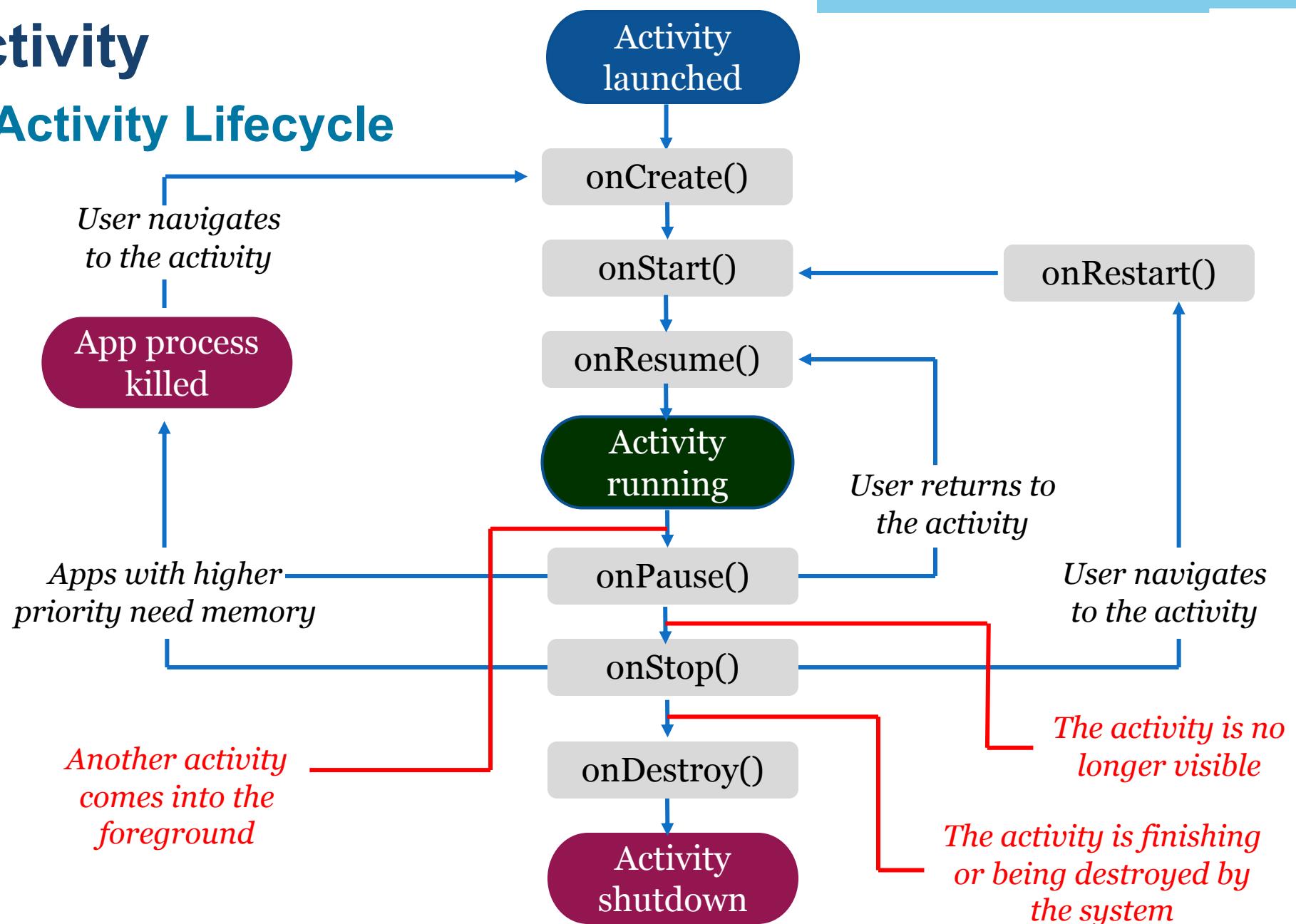
# 1. Activity

## »» Activity Stack



# 1. Activity

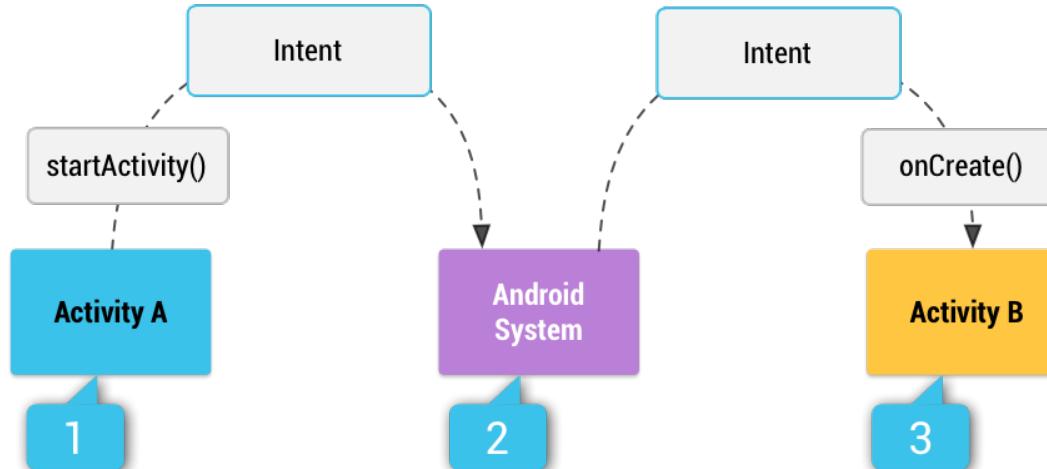
## »» Activity Lifecycle



## 2. Intent

### »» Khái niệm

- Intent được sử dụng để **truyền tải thông điệp, yêu cầu một hành động xử lý từ thành phần được gọi**.
- Intent được sử dụng trong ba trường hợp chính:
  - Khởi động Activity thông qua phương thức `startActivity`.
  - Khởi động Service thông qua phương thức `startService`.
  - Chuyển thông điệp đến BroadcastReceiver thông qua phương thức `sendBroadcast`.



## 2. Intent

### »» Phân loại Intent

- Intent được chia làm hai dạng:
  - **Explicit Intent:** chỉ định rõ thành phần xử lý thông qua tên lớp, thường được dùng để gọi đến các thành phần trong cùng ứng dụng.
  - **Implicit Intent:** không chỉ định rõ thành phần xử lý, thay vào đó bổ sung các thuộc tính như: mô tả hành động, dạng dữ liệu...

## 2. Intent

### »» Phân loại Intent

➤ **Explicit Intent:**

**Khai báo:**

Intent intent = new Intent(this, <**Component**>);

**Ví dụ:** khởi động Activity có tên SecondActivity từ MainActivity

Intent intent = new Intent(MainActivity.this,  
**SecondActivity.class**);

startActivity(intent);

## 2. Intent

### »» Phân loại Intent

#### ➤ Implicit Intent:

##### Khai báo:

```
Intent intent = new Intent(this, <Action>);
```

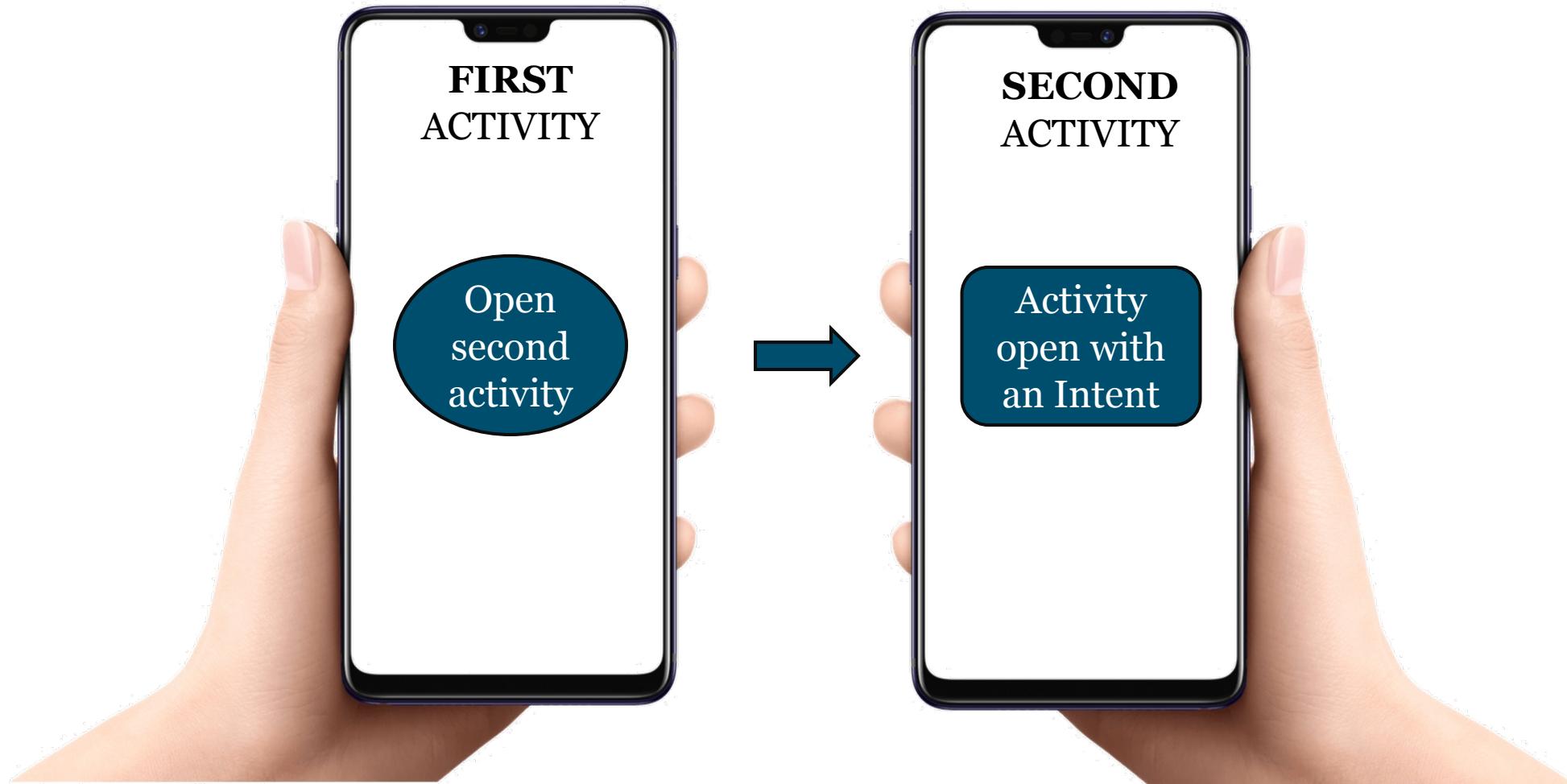
Ví dụ: khởi động Activity gọi điện thoại

```
Intent intent = new Intent(Intent.ACTION_CALL);
```

```
startActivity(intent);
```

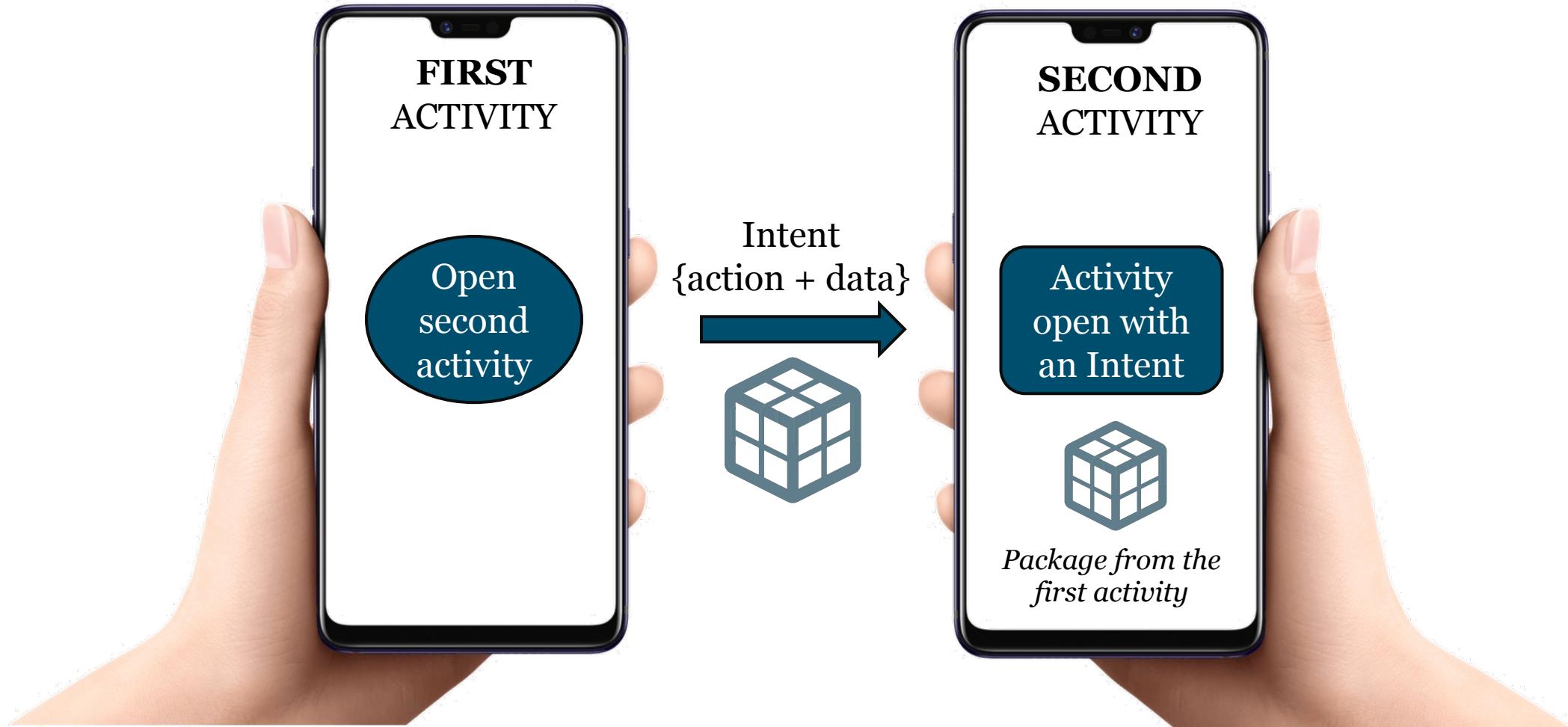
### 3. Truyền và nhận dữ liệu giữa các Activity

#### »» Mở một activity khác



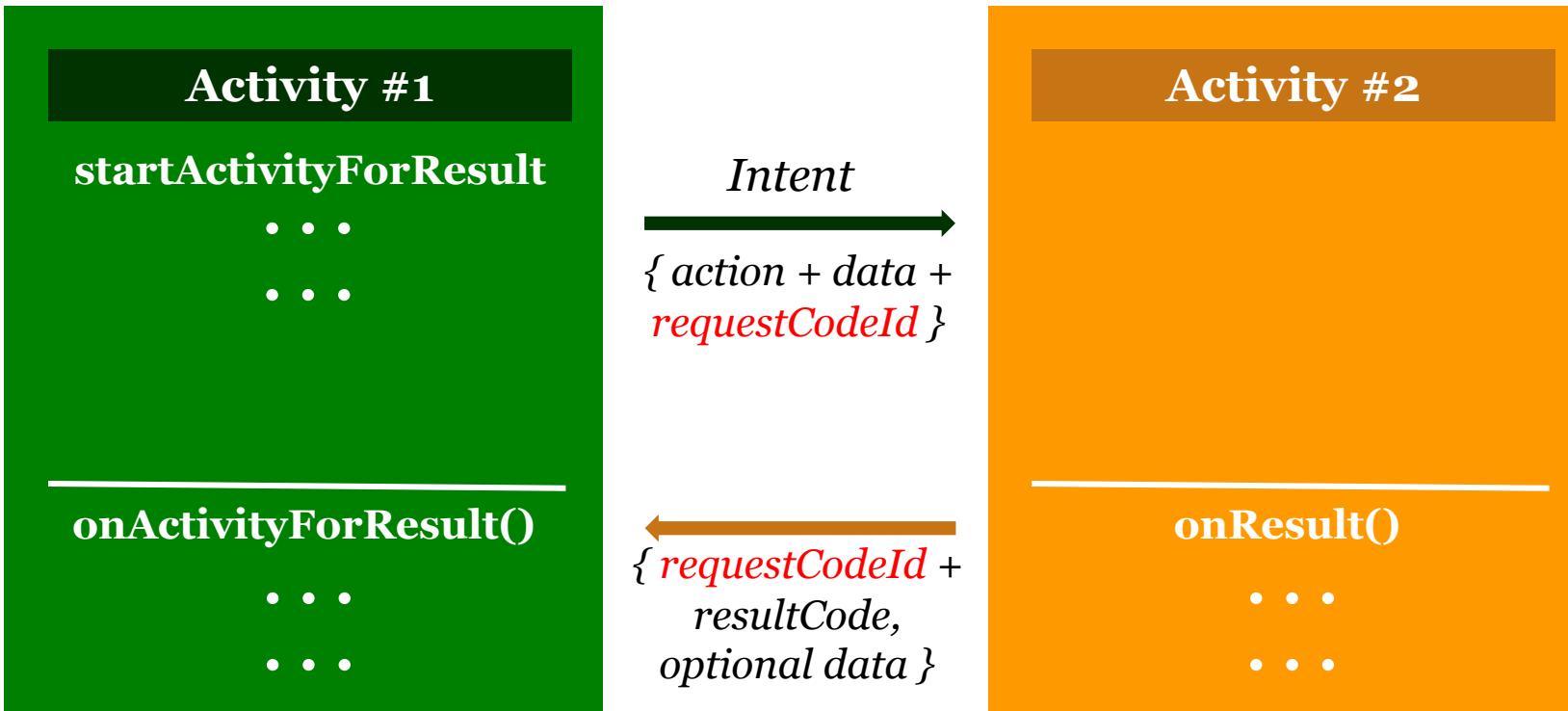
### 3. Truyền và nhận dữ liệu giữa các Activity

»» Mở activity khác → truyền dữ liệu qua



### 3. Truyền và nhận dữ liệu giữa các Activity

» Mở activity khác → truyền dữ liệu qua và xử lý dữ liệu trả về



## 4. Implicit Intent

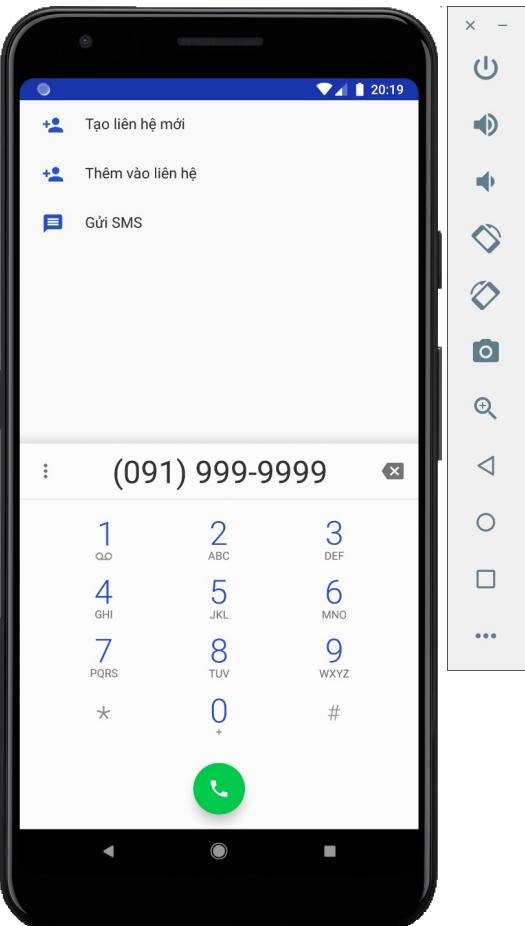
### »» Gọi Action thực hiện gọi điện thoại

```
//Thực hiện quay số  
//Intent intent = new Intent(Intent.ACTION_DIAL);  
//Thực hiện cuộc gọi  
Intent intent = new Intent(Intent.ACTION_CALL);  
Uri uri = Uri.parse("tel:" +  
edtPhone.getText().toString());  
intent.setData(uri);  
startActivity(intent);
```

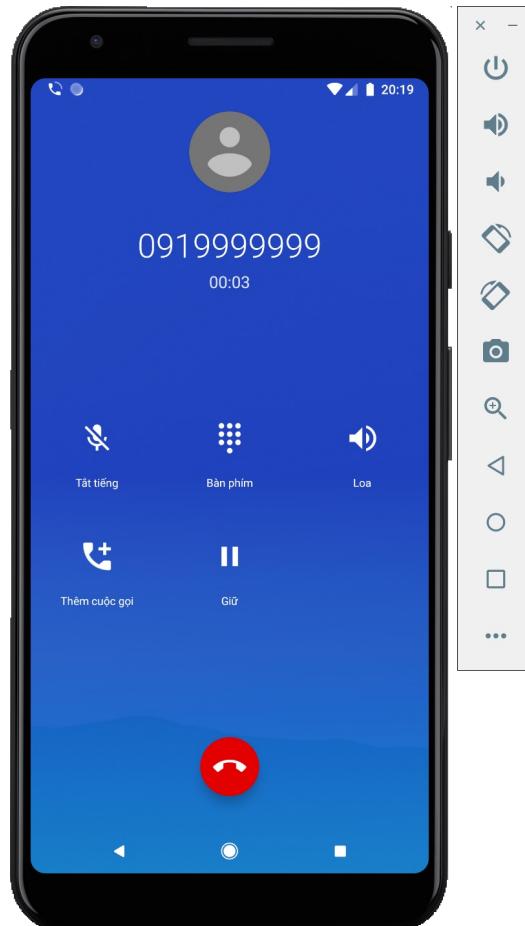
```
<uses-permission  
android:name="android.permission.CALL_PHONE"/>
```

## 4. Implicit Intent

»» Gọi Action thực hiện gọi điện thoại



ACTION\_DIAL



ACTION\_CALL

## 4. Implicit Intent

### » Gọi Action thực hiện gửi tin nhắn

```
SmsManager smsManager=SmsManager.getDefault();  
smsManager.sendTextMessage(  
    edtPhone.getText().toString(),  
    scAddress: null,  
    edtSms.getText().toString(),  
    sentIntent: null, deliveryIntent: null);
```

```
<uses-permission  
    android:name="android.permission.SEND_SMS"/>
```

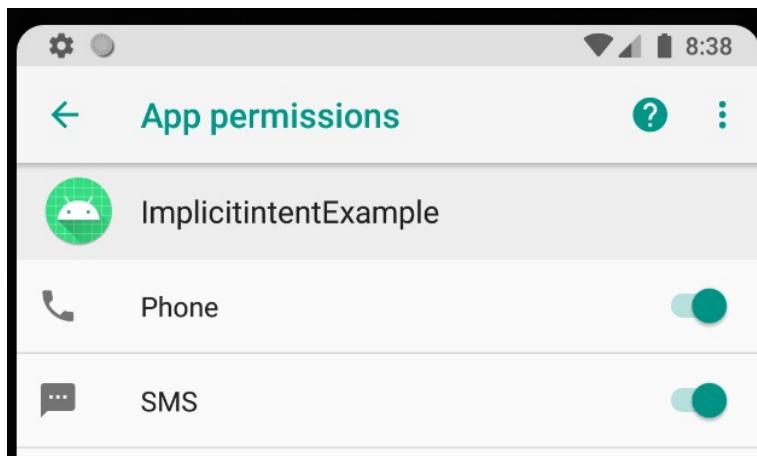
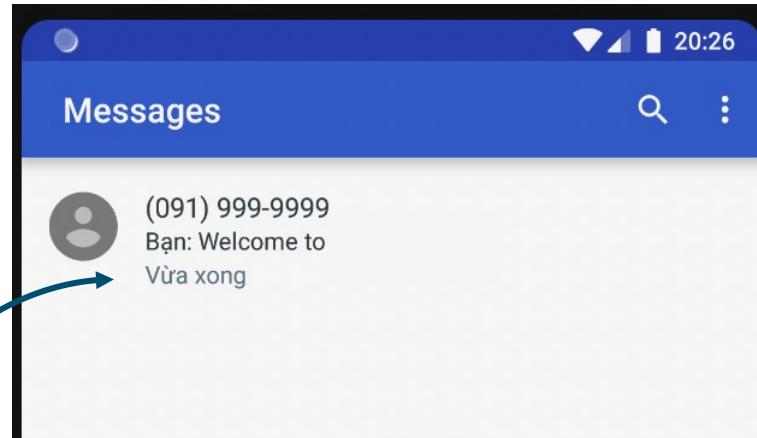
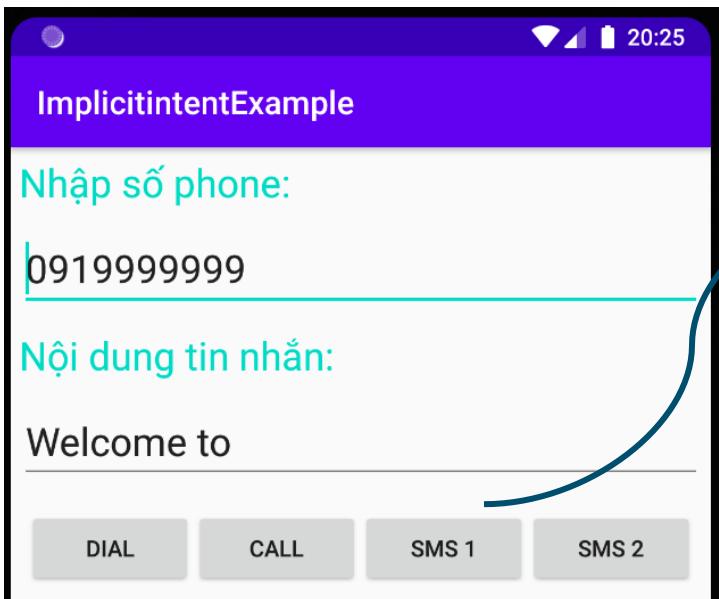
## 4. Implicit Intent

### »» Gọi Action thực hiện gửi tin nhắn

```
SmsManager smsManager=SmsManager.getDefault();
Intent msgSent=new Intent("ACTION_MSG_SENT");
PendingIntent pendingMsgSent =
PendingIntent.getBroadcast(MainActivity.this,o,msgSent,o);
registerReceiver(
    new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            int kq=getResultCode();
            if(kq== Activity.RESULT_OK)
                // Thông báo gửi tin nhắn thành công
            else
                // Thông báo gửi tin nhắn thất bại
        }
    },
    new IntentFilter("ACTION_MSG_SENT")
);
smsManager.sendTextMessage(
    edtPhone.getText().toString(), null
    edtSms.getText().toString(), pendingMsgSent,null);
```

## 4. Implicit Intent

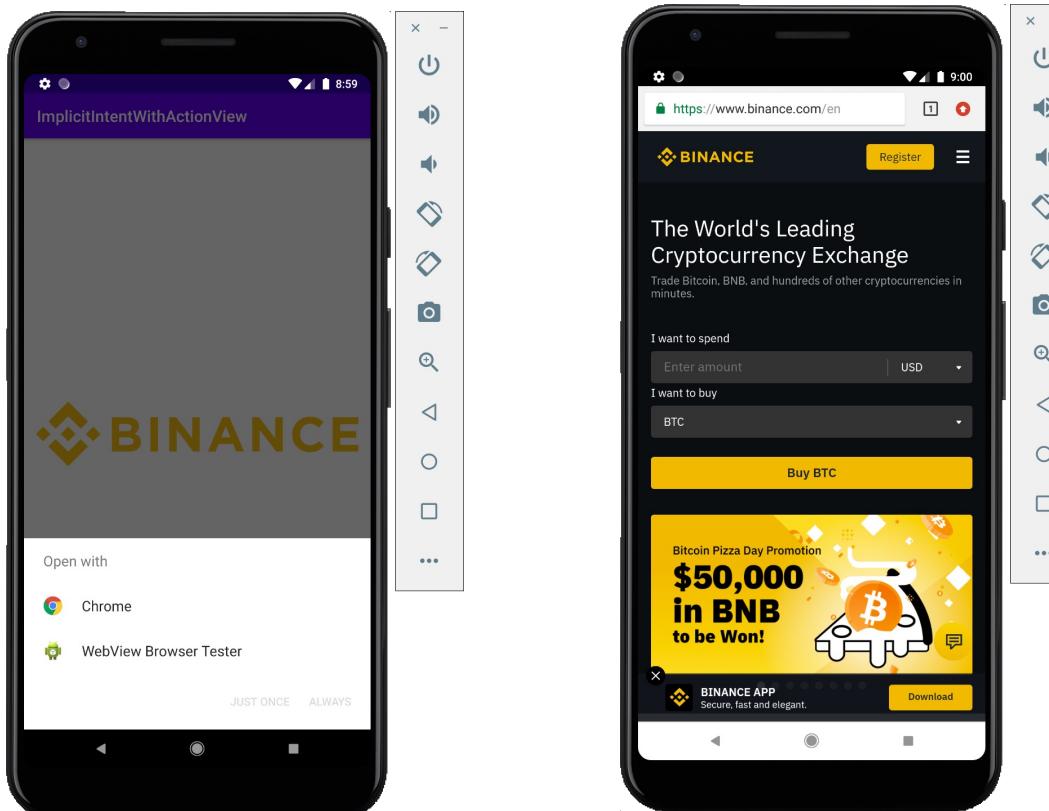
### »» Gọi Action thực hiện gửi tin nhắn



## 4. Implicit Intent

### »» Implicit intent với ACTION\_VIEW

```
Intent intent = new Intent();
intent.setAction(Intent.ACTION_VIEW);
intent.setData(Uri.parse("https://binance.com"));
startActivity(intent);
```



## 4. Implicit Intent

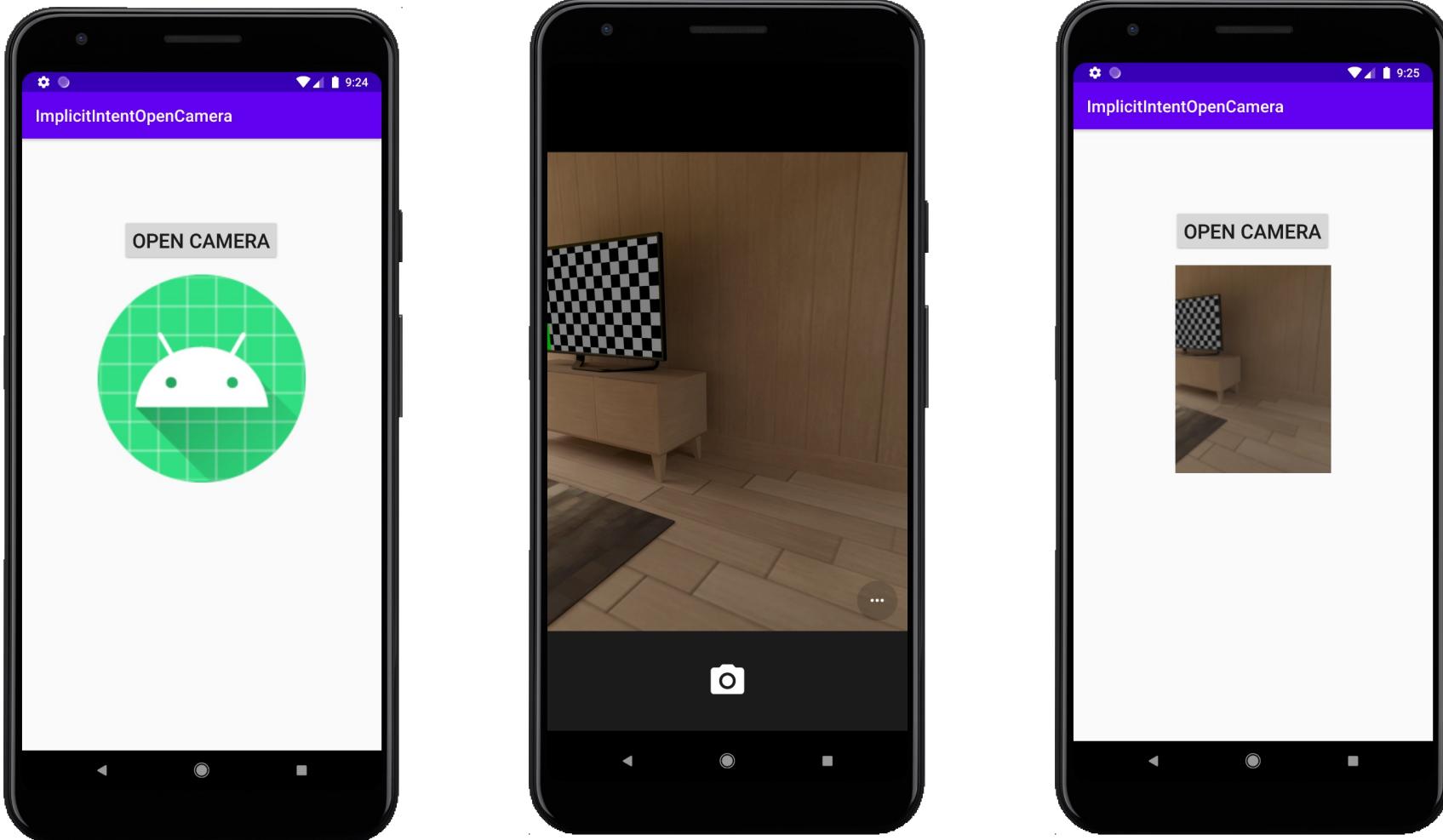
### »» Implicit intent mở Camera

```
public void openCamera(View view) {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    startActivityForResult(intent, REQUEST_CODE_ID);
}

@Override
protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode == REQUEST_CODE_ID && resultCode == RESULT_OK
        && data != null){
        Bitmap bitmap = (Bitmap) data.getExtras().get("data");
        imvPhoto.setImageBitmap(bitmap);
    }
}
```

## 4. Implicit Intent

### »» Implicit intent mở Camera



06 FRAGMENT

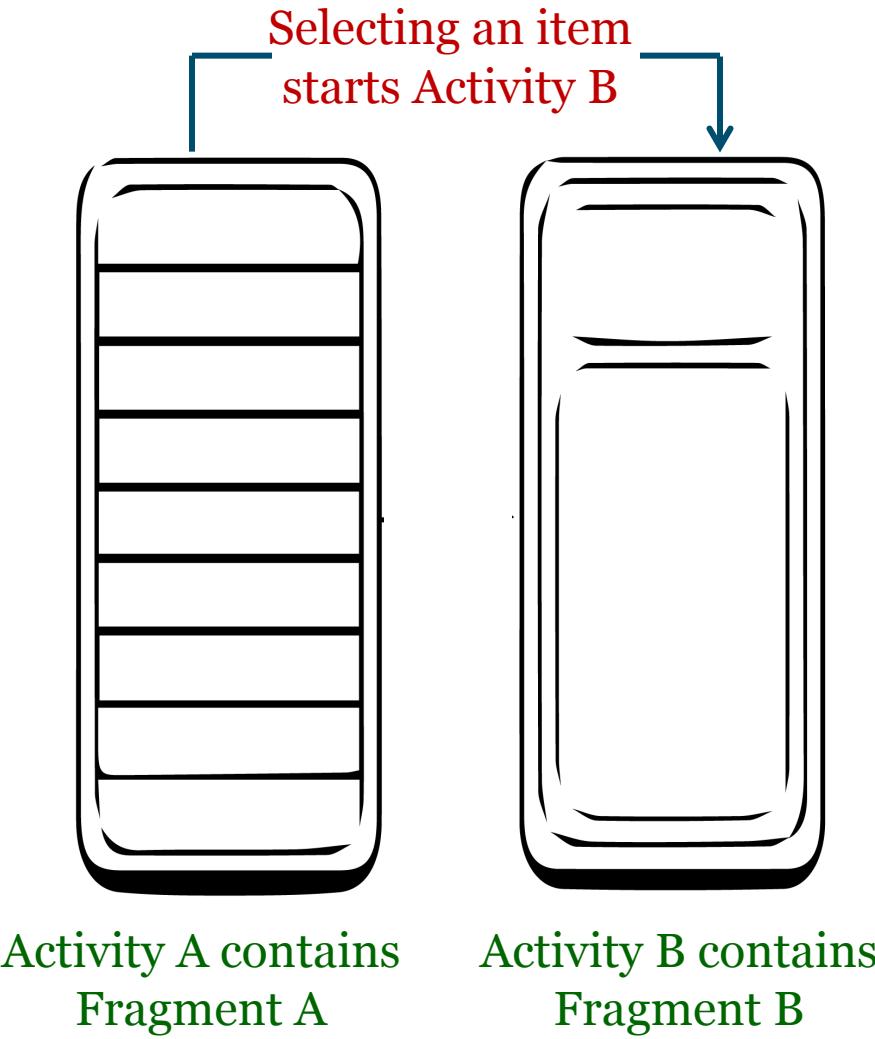
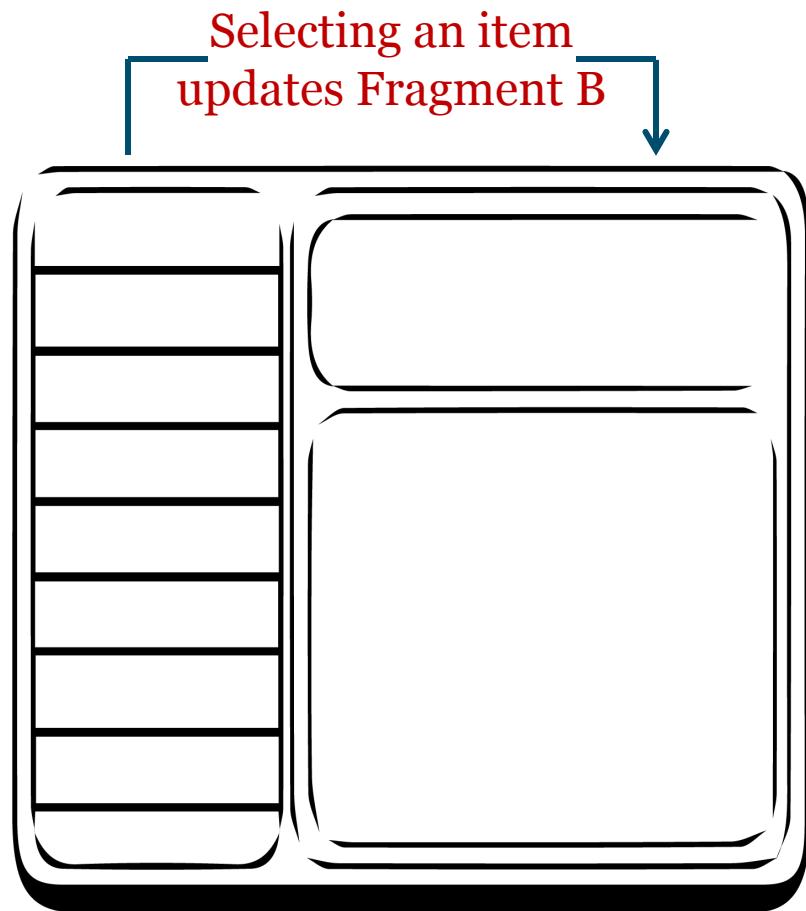
# NỘI DUNG

1. Giới thiệu về Fragment
2. Vòng đời Fragment
3. Xây dựng và sử dụng Fragment

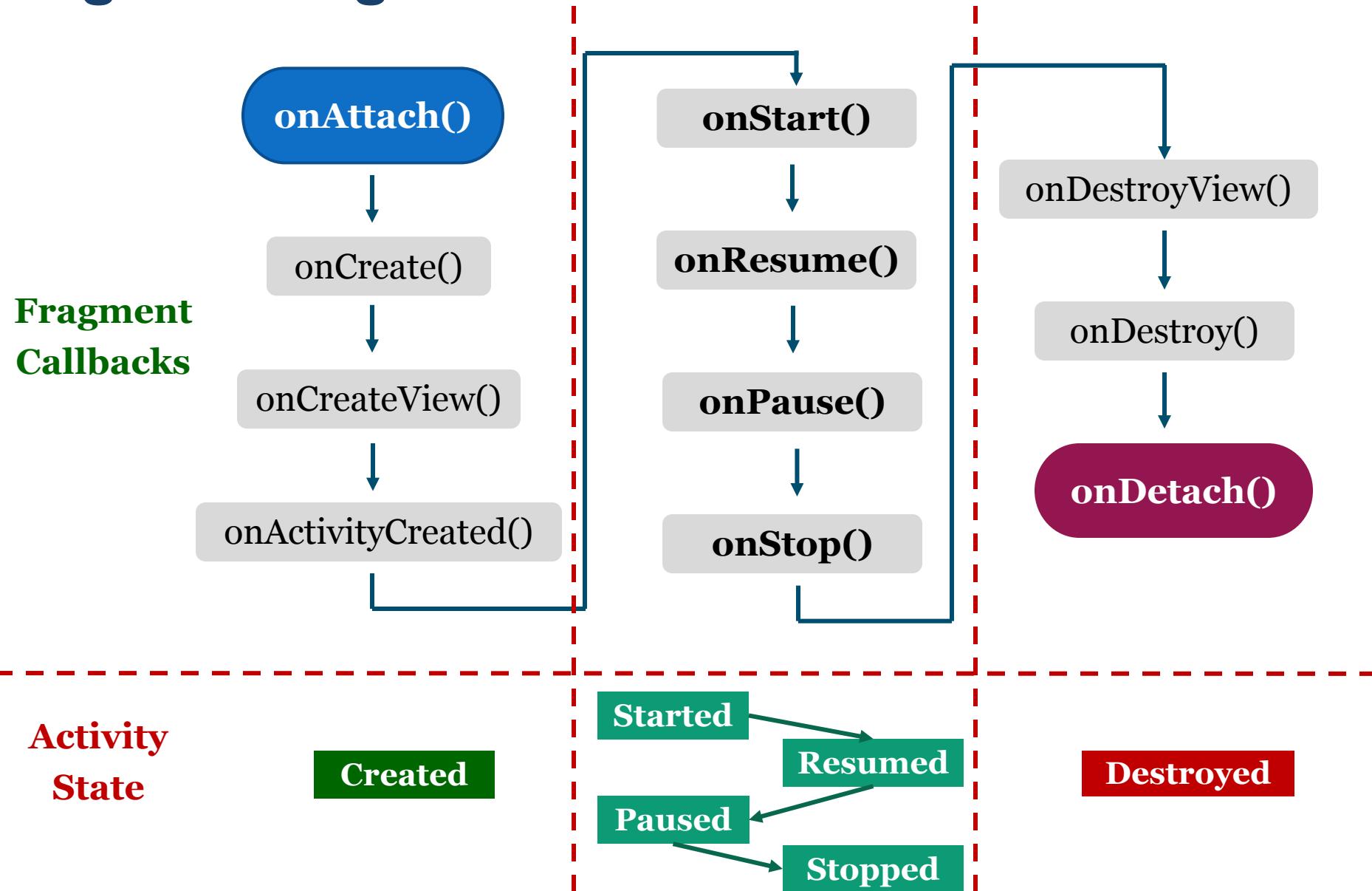
# 1. Giới thiệu về Fragment

- **Fragment** là một đối tượng biểu diễn một phần giao diện người dùng được nhúng trong **Activity**, cho phép thực hiện nhận tương tác, có vòng đời riêng và thực hiện trao đổi thông tin với **Activity** và các **Fragment khác**.
- Một số lưu ý khi sử dụng Fragment:
  - **Fragment** cũng có *layout riêng*, cũng có các *sự kiện* và *vòng đời riêng* → có thể thêm hoặc xóa fragment khi Activity đang mở.
  - Có thể kết hợp nhiều Fragment trong 1 Activity.
  - Một Fragment có thể được sử dụng ở nhiều Activity.

# 1. Giới thiệu về Fragment

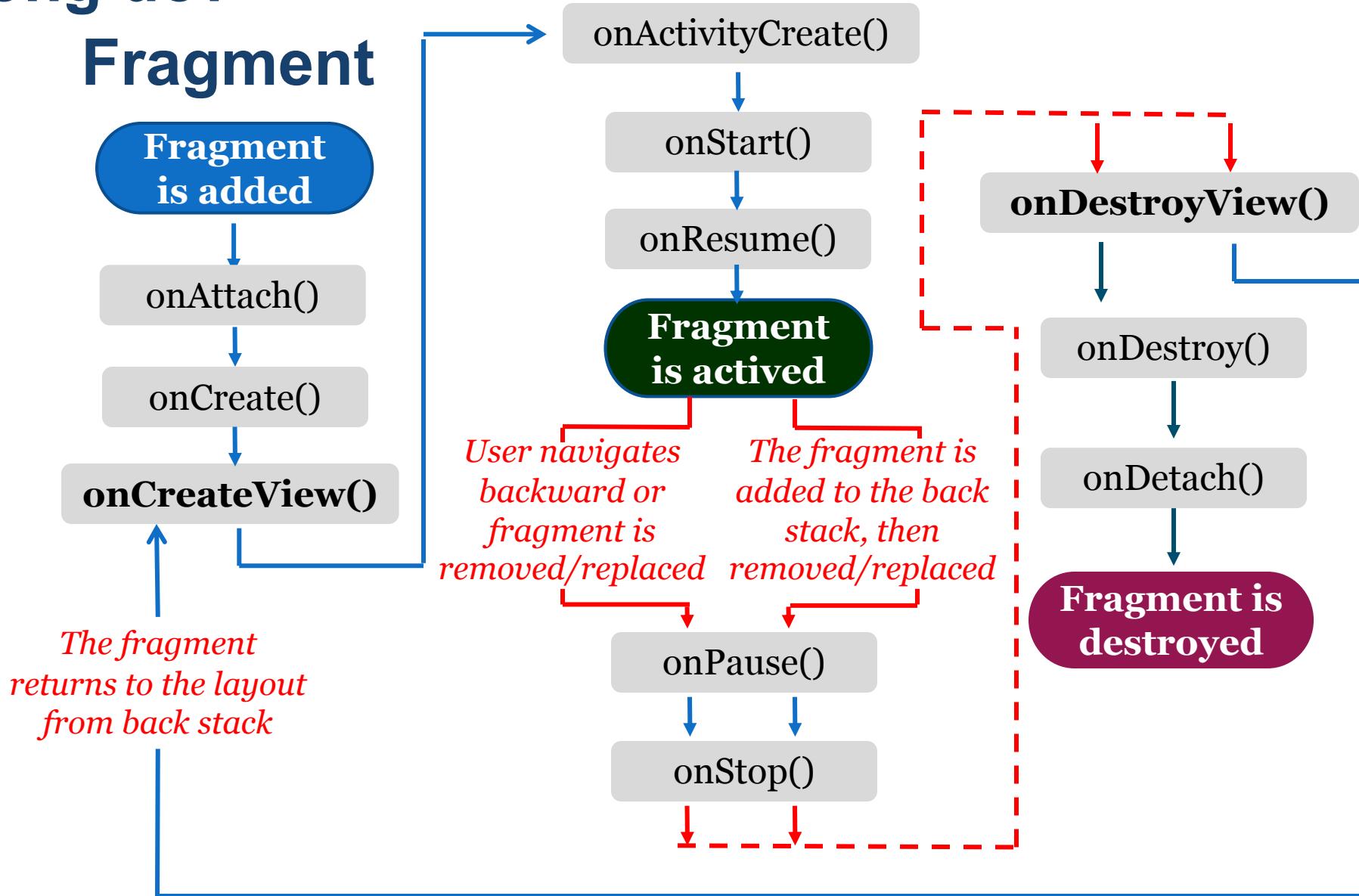


## 2. Vòng đời Fragment



## 2. Vòng đời

### Fragment



### 3. Xây dựng và sử dụng Fragment

#### »» Xây dựng Fragment

- **Bước 1:** Tạo lớp kế thừa từ lớp **Fragment**
- **Bước 2:** Override phương thức **onCreateView()** thực hiện nạp giao diện cho Fragment

```
public class MyFragment extends Fragment {  
    @Override  
    public View onCreateView(@NonNull LayoutInflater inflater,  
                             @Nullable ViewGroup container,  
                             @Nullable Bundle savedInstanceState) {  
        return inflater.inflate(R.layout.fragment_layout_1, container, false);  
    }  
}
```

### 3. Xây dựng và sử dụng Fragment

#### »» Sử dụng Fragment

- **Cách 1:** Thực hiện tham chiếu **Fragment** từ giao diện XML của Activity.
- **Cách 2:** Nhúng Fragment vào Activity thông qua đối tượng **FragmentManager** từ JavaCode.

### 3. Xây dựng và sử dụng Fragment

#### »» Sử dụng Fragment

- **Cách 1:** Thực hiện tham chiếu Fragment từ giao diện XML của Activity.

```
<fragment
    android:id="@+id/fragment1"
    android:name="com.example.fragmentdemo.MyFragment"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    tools:layout="@layout/fragment_layout_1" />
```

### 3. Xây dựng và sử dụng Fragment

#### »» Sử dụng Fragment

- **Cách 2:** Nhúng Fragment vào Activity thông qua đối tượng **FragmentManager**.

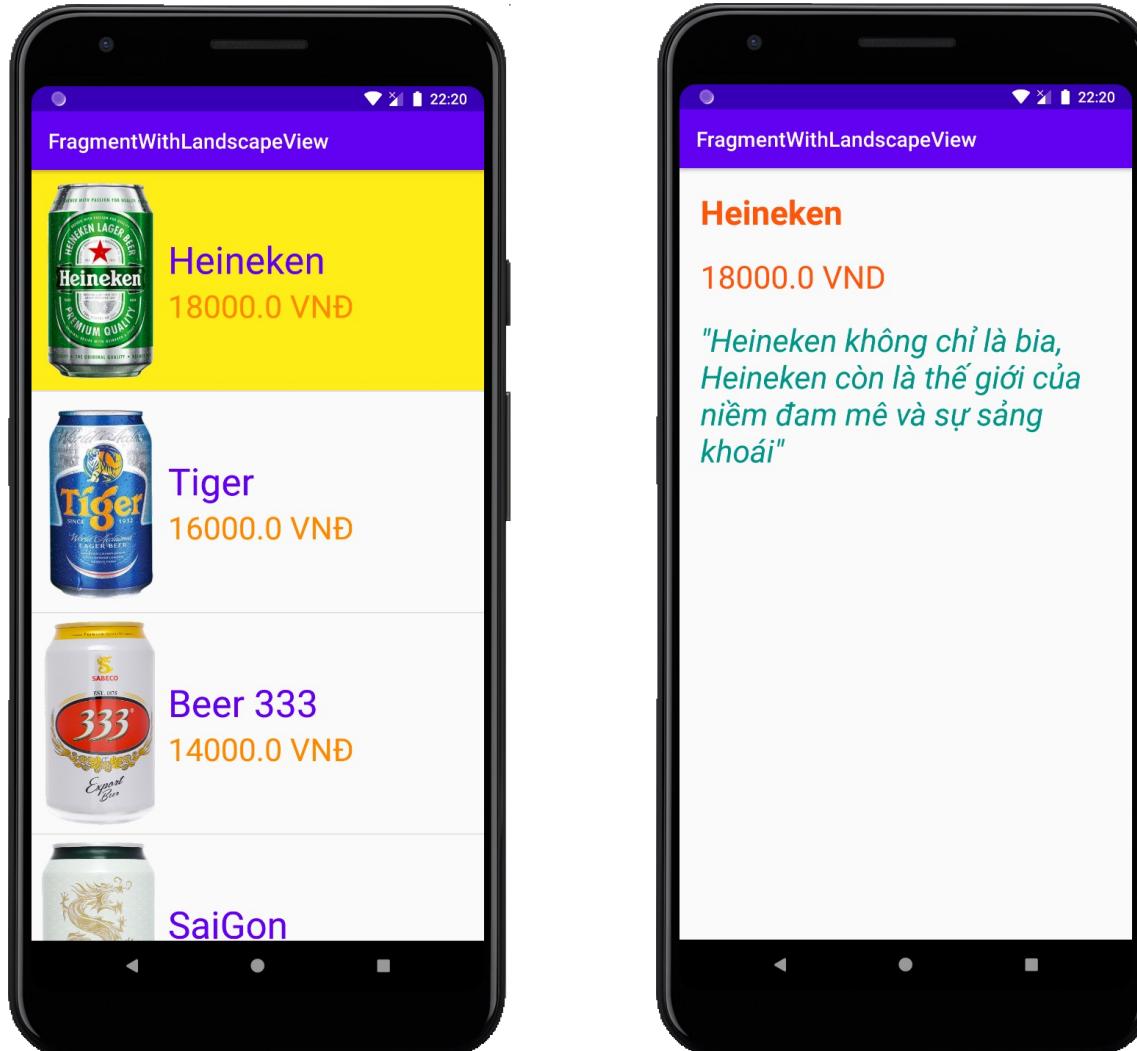
```
FragmentManager fragmentManager =  
    getSupportFragmentManager();  
  
FragmentTransaction fragmentTransaction =  
    fragmentManager.beginTransaction();  
  
Fragment fragment = new MyFragment();  
fragmentTransaction.add(R.id.layoutContainer, fragment);  
fragmentTransaction.commit();
```

Một số phương thức quan trọng của **FragmentTransaction**:

- |                              |                           |                          |
|------------------------------|---------------------------|--------------------------|
| ✓ <b>add()</b>               | ✓ <b>hide() – show()</b>  | ✓ <b>setTransition()</b> |
| ✓ <b>attach() – detach()</b> | ✓ <b>remove()</b>         | ✓ <b>Commit()</b>        |
| ✓ <b>replace()</b>           | ✓ <b>addToBackStack()</b> |                          |

### 3. Xây dựng và sử dụng Fragment

»» Bài tập:



### 3. Xây dựng và sử dụng Fragment

»» Bài tập:



07

## ASSETS & SHARED PREFERENCES

# NỘI DUNG

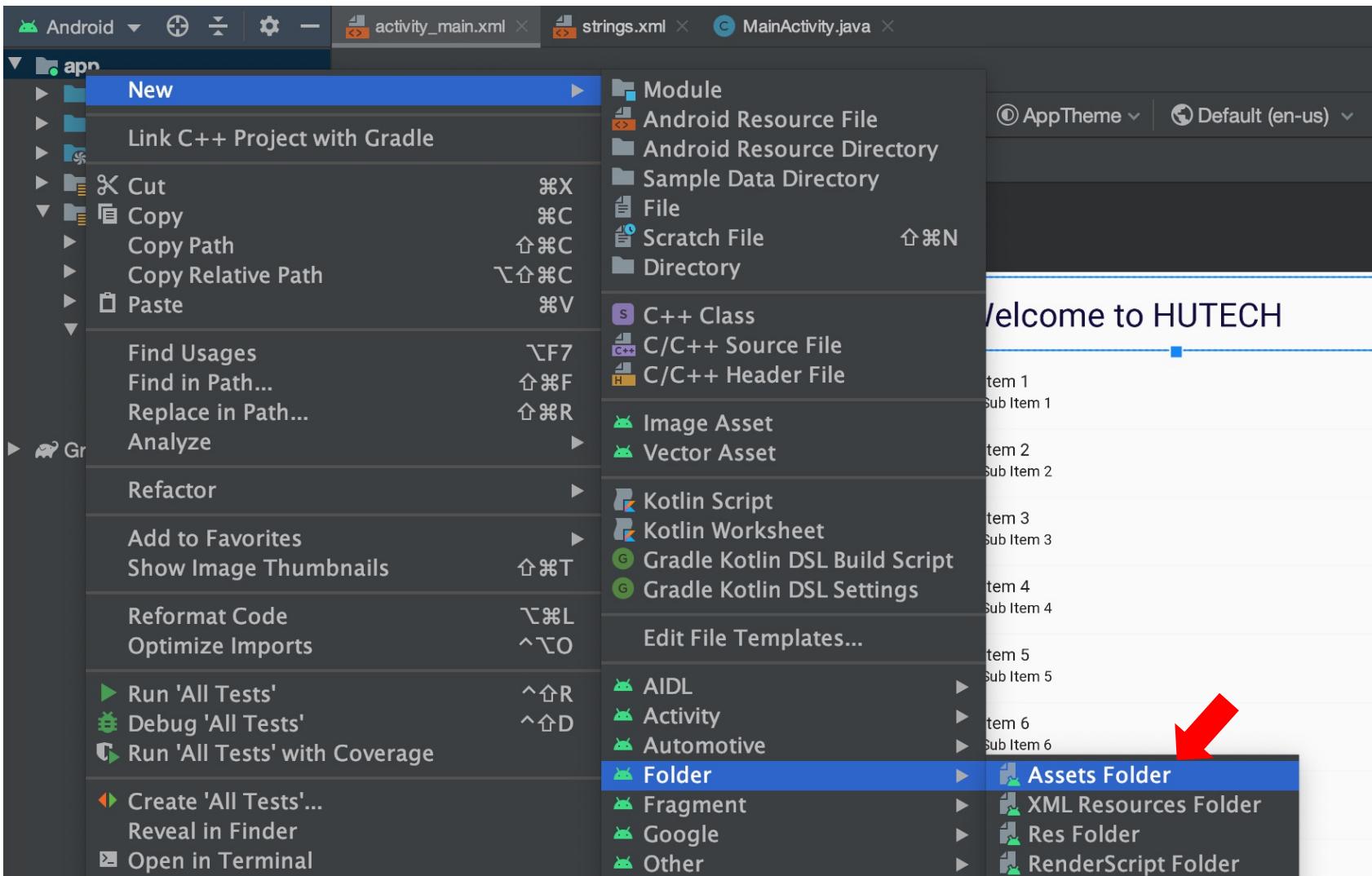
1. Giới thiệu về Assets trong Android
2. Cách thức truy xuất dữ liệu từ Assets
3. Giới thiệu về Shared Preferences
4. Sử dụng Shared Preferences

# 1. Giới thiệu về Assets trong Android

- **Assets** là thư mục chứa các tệp dữ liệu đầu vào cho ứng dụng: *âm thanh, hình ảnh, văn bản, tệp csdl, ...*. Những tệp này sẽ không được biên dịch khi ứng dụng được đóng gói.
- Các tệp trong **assets/** không được cấp ID tài nguyên → chỉ có thể đọc chúng bằng cách sử dụng **AssetManager**.

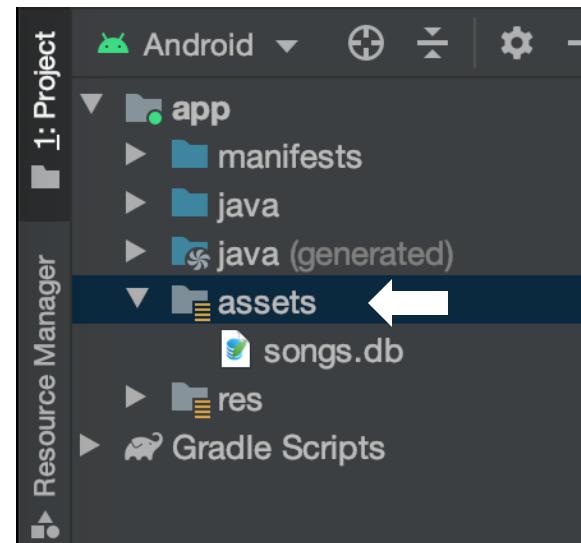
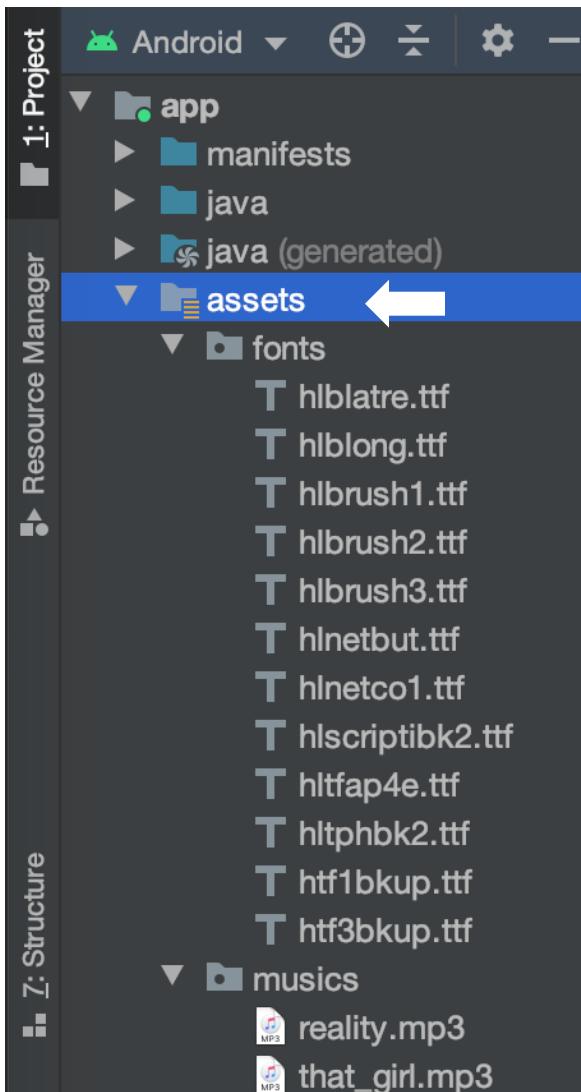
# 1. Giới thiệu về Assets trong Android

## »» Tạo thư mục Assets



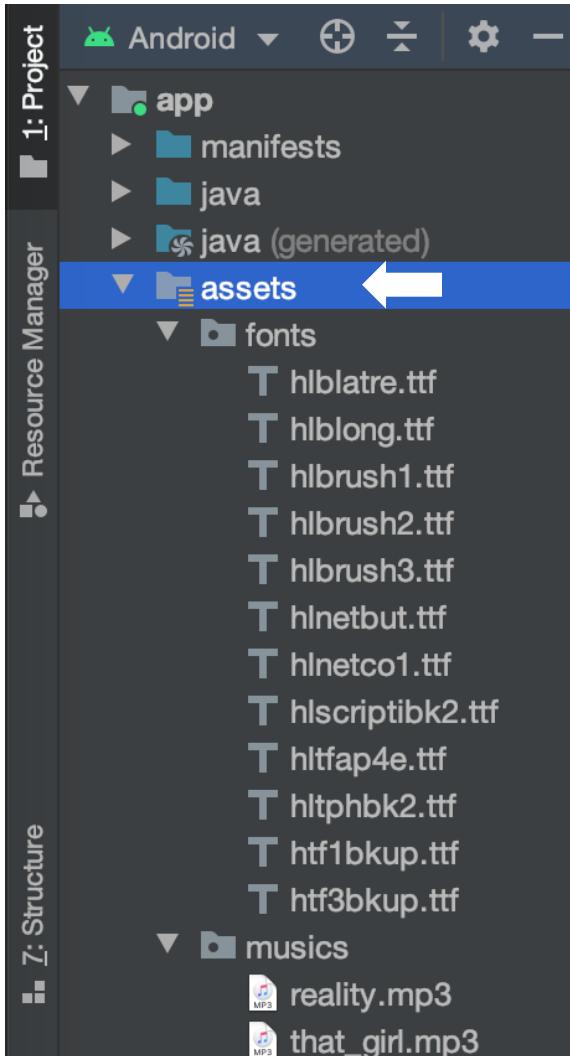
# 1. Giới thiệu về Assets trong Android

## »» Tạo thư mục Assets



## 2. Cách thức truy xuất dữ liệu từ Assets

» Vd: Truy xuất Font chữ



## 2. Cách thức truy xuất dữ liệu từ Assets

### »» Vd: Truy xuất Font chữ

```
lvFonts = findViewById(R.id.lvFonts);
fontAdapter = new ArrayAdapter<String>(MainActivity.this,
        android.R.layout.simple_list_item_1);
lvFonts.setAdapter(fontAdapter);
try{
    AssetManager assetManager = getAssets();
    String[] arrFonts = assetManager.list("fonts");
    fontAdapter.addAll(arrFonts);
} catch (Exception e){
    Log.e("Error: ", e.toString());
}
```

```
Typeface typeface = Typeface.createFromAsset(getAssets(),
        "fonts/" + fontAdapter.getItem(i));
textContent.setTypeface(typeface);
```

## 2. Cách thức truy xuất dữ liệu từ Assets

### » Vd: Truy xuất tệp âm thanh

```
private void playAudio() {  
    try{  
        AssetFileDescriptor assetFileDescriptor =  
            getAssets().openFd("musics/ting.mp3");  
        MediaPlayer mediaPlayer = new MediaPlayer();  
        mediaPlayer.setDataSource(  
            assetFileDescriptor.getFileDescriptor(),  
            assetFileDescriptor.getStartOffset(),  
            assetFileDescriptor.getLength());  
        assetFileDescriptor.close();  
        mediaPlayer.prepare();  
        mediaPlayer.start();  
    }catch (Exception e){  
        Log.e("Error: " , e.toString());  
    }  
}
```

### 3. Giới thiệu về Shared Preferences

- **Shared Preferences** cho phép lưu trữ dữ liệu ứng dụng bằng file XML.
- Dữ liệu được lưu trữ và truy xuất theo cặp khóa **key-value** với các kiểu dữ liệu như: *Boolean, Float, Int, Long, String*.
- Dữ liệu vẫn được bảo toàn ngay cả khi ứng dụng đóng hoàn toàn.

File list				
	Path	Permissions	Last modified	Size
▼	com.example.sharedpreferencesexample	drwx-----	2020-05-24 23:40	4 KB
►	cache	drwxrws--x	2020-05-24 23:40	4 KB
►	code_cache	drwxrws--x	2020-05-24 23:40	4 KB
▼	shared_prefs	drwxrwx--x	2020-05-25 20:46	4 KB
	my_data.xml	-rw-rw----	2020-05-25 20:46	406 B

## 4. Sử dụng Shared Preferences

### »» Lưu trữ dữ liệu

- **B1:** Khởi tạo đối tượng Shared Preferences

```
SharedPreferences preferences =  
    getSharedPreferences(PREFERENCES_NAME, MODE);
```

- **B2:** Tạo đối tượng Editor từ phương thức **edit()** của preferences

```
SharedPreferences.Editor editor = preferences.edit();
```

- **B3:** Thêm dữ liệu với phương thức **put<Type>** tùy theo kiểu dữ liệu

```
editor.putInt("int", 8);  
editor.putFloat("float", 6.8f);  
editor.putLong("long", 36);  
editor.putBoolean("boolean", true);  
editor.putString("string", "Android Sweets");
```

- **B4:** Lưu Shared Preferences với phương thức **apply()** hoặc **commit()**

```
editor.apply();
```

## 4. Sử dụng Shared Preferences

### »» Truy xuất dữ liệu

- Dữ liệu được truy xuất với phương thức **get<Type>** tùy theo kiểu dữ liệu:

```
get<Type>(Key, DefValue);
```

**DefValue**: là giá trị mặc định được trả về khi **get<Type>** không trả về được **value** ứng với **key** truyền vào.

- Ví dụ:

```
SharedPreferences preferences =  
    getSharedPreferences(PREFERENCES_NAME, MODE);  
int i = preferences.getInt("int", 0);  
float f = preferences.getFloat("float", 0.0f);  
long l = preferences.getLong("long", 0);  
boolean b = preferences.getBoolean("boolean", false);  
String s = preferences.getString("string", "");
```

## 08 SQLITE

# NỘI DUNG

1. Giới thiệu SQLite
2. Tạo CSDL SQLite
3. Thao tác trên CSDL SQLite
4. Làm việc với dữ liệu hình ảnh

# 1. Giới thiệu về SQLite

- **SQLite** là cơ sở dữ liệu mở được nhúng trực tiếp vào Android, hỗ trợ các đặc điểm về quan hệ chuẩn của csdl như: cú pháp câu lệnh, transaction, ...
- SQLite có kích thước nhỏ, gọn nhẹ với cấu hình đầy đủ không quá 300kb
- SQLite hỗ trợ các kiểu dữ liệu như: *Text, Integer, Real, Blob*

# 1. Giới thiệu về SQLite

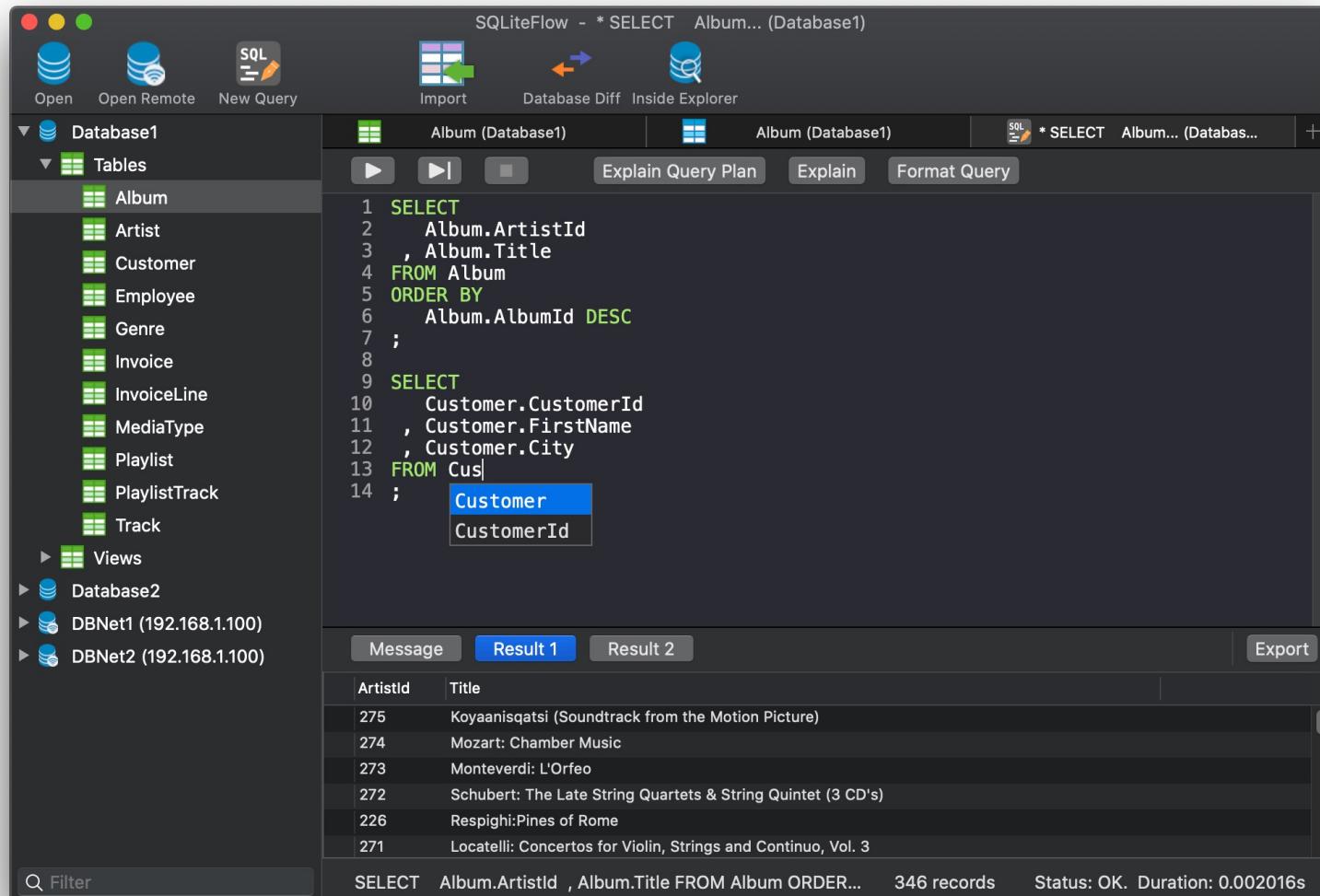
- Đường dẫn của csdl SQLite trên thiết bị:

**data/data/APP\_NAME/databases/FILE\_NAME**

→ CSDL SQLite được sử dụng cục bộ cho ứng dụng. Trong trường hợp cần cung cấp, chia sẻ dữ liệu với ứng dụng khác nên sử dụng **Content Provider**.

## 2. Tạo CSDL SQLite

- C1 - Sử dụng Tool: SQLite Studio, SQLiteFlow,



- C2 - Coding trực tiếp tạo csdl cho ứng dụng

## 2. Tạo CSDL SQLite

File List			
Path	Type	Last Modified	Size
data	drwxrwx--x	2020-05-18 22:31	4 KB
adb	drwx-----	2020-05-18 22:31	4 KB
anr	drwxrwxr-x	2020-06-06 19:27	4 KB
app	drwxrwx--x	2020-07-16 10:58	4 KB
app-asec	drwx-----	2020-05-18 22:31	4 KB
data	drwxrwx--x	2020-07-16 10:58	8 KB
android	drwx-----	2020-05-18 22:32	4 KB
com.android.backupconfirm	drwx-----	2020-05-18 22:32	4 KB
com.example.sqliteexample1	drwx-----	2020-05-29 13:27	4 KB
cache	drwxrws--x	2020-05-29 13:27	4 KB
code_cache	drwxrws--x	2020-05-29 13:27	4 KB
databases	drwxrwx--x	2020-05-29 13:27	4 KB
note_db.sqlite	-rw-rw----	2020-06-07 07:30	20 KB
note_db.sqlite-journal	-rw-rw----	2020-06-07 07:30	0 B
com.example.sqliteexample2	drwx-----	2020-06-03 23:27	4 KB
cache	drwxrws--x	2020-06-03 23:27	4 KB
code_cache	drwxrws--x	2020-06-03 23:27	4 KB
databases	drwxrwx--x	2020-06-03 23:27	4 KB
songs.db	-rw-rw----	2020-06-07 00:12	16 KB
songs.db-journal	-rw-rw----	2020-06-07 00:12	0 B

## 2. Tạo CSDL SQLite

### ➤ Sao chép csdl SQLite vào hệ thống (dùng cho C1)

```
public static final String DATABASE_NAME = "product_db.db";
public static final String DB_PATH_SUFFIX = "/databases/";
public static SQLiteDatabase database = null;

private void copyDataBase(){
    try{
        File dbFile = getDatabasePath(DATABASE_NAME);
        if(!dbFile.exists()){
            if(CopyDBFromAsset()){
                Toast.makeText(MainActivity.this,
                    "Copy database successful!", Toast.LENGTH_LONG).show();
            }else{
                Toast.makeText(MainActivity.this,
                    "Copy database fail!", Toast.LENGTH_LONG).show();
            }
        }
    }catch (Exception e){
        Log.e("Error: ", e.toString());
    }
}
```

## 2. Tạo CSDL SQLite

### ➤ Sao chép csdl SQLite vào hệ thống (dùng cho C1)

```
private boolean CopyDBFromAsset() {
    String dbPath = getApplicationInfo().dataDir + DB_PATH_SUFFIX +
DATABASE_NAME;
    try {
        InputStream inputStream = getAssets().open(DATABASE_NAME);
        File f = new File(getApplicationInfo().dataDir + DB_PATH_SUFFIX);
        if(!f.exists()){
            f.mkdir();
        }
        OutputStream outputStream = new FileOutputStream(dbPath);
        byte[] buffer = new byte[1024]; int length;
        while((length=inputStream.read(buffer))>0){
            outputStream.write(buffer,0, length);
        }
        outputStream.flush(); outputStream.close(); inputStream.close();
        return true;
    } catch (IOException e) {
        e.printStackTrace();
        return false;
    }
}
```

## 2. Tạo CSDL SQLite

### ➤ Coding trực tiếp (C2)

```
public class Databases extends SQLiteOpenHelper {  
  
    public Databases(@Nullable Context context, @Nullable String name,  
                    @Nullable SQLiteDatabase.CursorFactory factory, int version) {  
        super(context, name, factory, version);  
    }  
  
    //Truy vấn không trả kết quả (CREATE, INSERT, UPDATE, DELETE, ...)  
    public void QueryData(String sql){  
        SQLiteDatabase db = getWritableDatabase();  
        db.execSQL(sql);  
    }  
  
    //Truy vấn trả về kết quả (SELECT)  
    public Cursor GetData(String sql){  
        SQLiteDatabase db = getReadableDatabase();  
        return db.rawQuery(sql,null);  
    }  
}
```

## 2. Tạo CSDL SQLite

### ➤ Coding trực tiếp (C2)

```
private void PrepareDB() {
    //Create database
    databases = new Databases(this, "note_db.sqlite", null, 1);
    //Create table
    databases.QueryData(
        "CREATE TABLE IF NOT EXISTS Works(Id INTEGER PRIMARY
        KEY AUTOINCREMENT, WorkName VARCHAR(200))"
    );
    //Insert data
    databases.QueryData("INSERT INTO Works VALUES(null, 'Fix bugs')");
    databases.QueryData("INSERT INTO Works VALUES(null, 'Coding')");
    .....
}
```

### 3. Thao tác trên CSDL SQLite

#### ➤ Mở cơ sở dữ liệu

```
public static final String DATABASE_NAME = "songs.db";
public static final String TABLE_NAME = "song";

public static SQLiteDatabase database = null;
database = openOrCreateDatabase(DATABASE_NAME,
                                MODE_PRIVATE, null);
```

### 3. Thao tác trên CSDL SQLite

#### ➤ Truy vấn csdl với “rawQuery”

```
Cursor cursor = database.rawQuery("SELECT * FROM " + TABLE_NAME,  
                                selectionArgs: null);  
  
while(cursor.moveToNext()){  
    int id = cursor.getInt(0);  
    int songCode = cursor.getInt(1);  
    String songName = cursor.getString(2);  
    String singer = cursor.getString(3);  
    int favourite = cursor.getInt(4);  
    //To do something ....  
}  
  
cursor.close();
```



<b><u>Id</u></b>	<b><u>ProductCode</u></b>	<b><u>ProductName</u></b>
1	Ho1	Heineken
2	To1	Tiger

### 3. Thao tác trên CSDL SQLite

#### ➤ Truy vấn csdl với “rawQuery”

```
Cursor cursor = database.rawQuery(  
    "SELECT * FROM " + TABLE_NAME + " WHERE Favourite = ? AND  
    (SongCode LIKE ? OR SongName LIKE ? OR Singer LIKE ?)",  
    new String[]{"1", "%" + s + "%", "%" + s + "%", "%" + s + "%"});  
  
while(cursor.moveToNext()){  
    int id = cursor.getInt(0);  
    int songCode = cursor.getInt(1);  
    String songName = cursor.getString(2);  
    //To do something ....  
}  
cursor.close();
```

### 3. Thao tác trên CSDL SQLite

#### ➤ Truy vấn csdl với “query”

```
Cursor cursor = database.query("Product", columns: null,  
                                selection: null, selectionArgs: null,  
                                groupBy: null, having: null, orderBy: null);
```

```
Cursor cursor = database.rawQuery(  
    "SELECT * FROM Product WHERE ProductId = ? OR ProductId = ?",
    new String[] {"2", "3"});
```



```
Cursor cursor = database.query("Product", columns: null,  
                                selection: "ProductId = ? OR ProductId = ?",  
                                selectionArgs: new String[] {"2", "3"},  
                                groupBy: null, having: null, orderBy: null);
```

### 3. Thao tác trên CSDL SQLite

#### ➤ Thêm dữ liệu

```
ContentValues values = new ContentValues();
values.put("ProductName", pName);
values.put("ProductPrice", pPrice);
long flag = database.insert("Product", nullColumnHack: null, values);
if(flag > 0)
    Toast.makeText(AddProductActivity.this, "Add product successful!",
        Toast.LENGTH_LONG).show();
else
    Toast.makeText(AddProductActivity.this, "Add product fail!",
        Toast.LENGTH_LONG).show();
```

### 3. Thao tác trên CSDL SQLite

#### ➤ Thêm dữ liệu

```
//Truy vấn không trả kết quả (CREATE, INSERT, UPDATE, DELETE, ...)  
public void QueryData(String sql){  
    SQLiteDatabase db = getWritableDatabase();  
    db.execSQL(sql);  
}  
  
//Insert data  
databases.QueryData("INSERT INTO Product VALUES(  
    null, 'Laptop Dell XPS', 26500000);
```

### 3. Thao tác trên CSDL SQLite

#### ➤ Xóa dữ liệu

```
int flag = database.delete("Product", "ProductId=?", new String[]{"1"});  
if(flag > 0)  
    //Xóa thành công  
else  
    //Xóa thất bại
```

//Truy vấn không trả kết quả (CREATE, INSERT, UPDATE, DELETE, ...)

```
public void QueryData(String sql){  
    SQLiteDatabase db = getWritableDatabase();  
    db.execSQL(sql);  
}
```

//Delete data

```
databases.QueryData("DELETE FROM Product WHERE ProductId = 1");
```

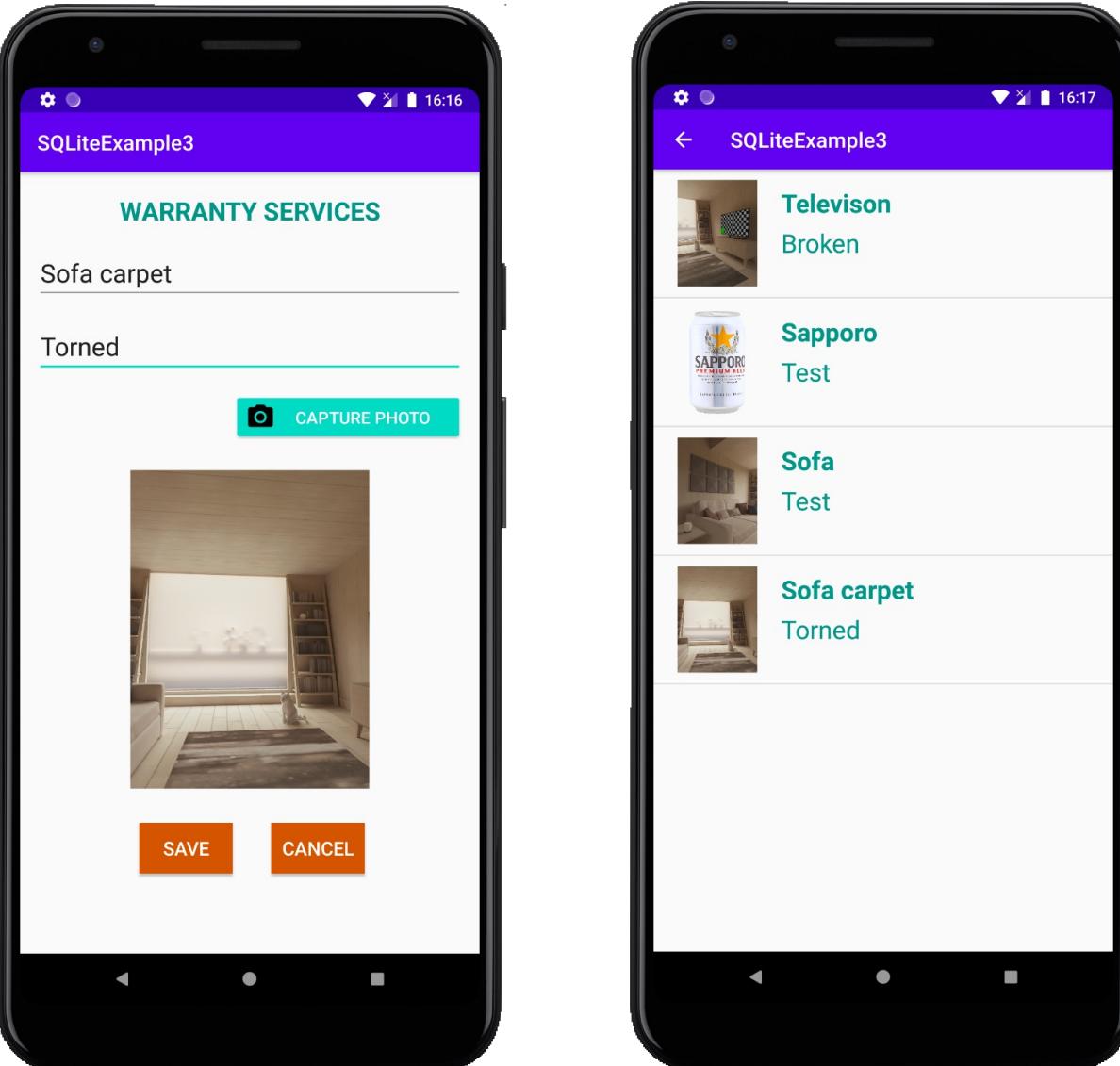
### 3. Thao tác trên CSDL SQLite

#### ➤ Sửa dữ liệu

```
ContentValues values = new ContentValues();
values.put("WorkName", "Fix bugs");
int flag = database.update("Works", values, "Id=?", new String[]{"1"});
if(flag > 0)
    //Cập nhật thành công
else
    //Cập nhật thất bại
```

```
//Truy vấn không trả kết quả (CREATE, INSERT, UPDATE, DELETE, ...)
public void QueryData(String sql){
    SQLiteDatabase db = getWritableDatabase();
    db.execSQL(sql);
}
//Update data
databases.QueryData("UPDATE Works SET WorkName = 'Fix bugs'
WHERE Id = 1");
```

## 4. Làm việc với dữ liệu hình ảnh



## 4. Làm việc với dữ liệu hình ảnh

```
public void insertData(String name, String des, byte[] photo){  
    SQLiteDatabase database = getWritableDatabase();  
    String sql = "INSERT INTO Product VALUES(null, ?, ?, ?)";  
  
    SQLiteStatement statement = database.compileStatement(sql);  
    statement.clearBindings();  
    statement.bindString(1, name);  
    statement.bindString(2, des);  
    statement.bindBlob(3, photo);  
  
    statement.executeInsert();  
}
```

## 4. Làm việc với dữ liệu hình ảnh

```
private byte[] convertPhoto() {  
    BitmapDrawable drawable = (BitmapDrawable) imgPhoto.getDrawable();  
    Bitmap bitmap = drawable.getBitmap();  
    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();  
    bitmap.compress(Bitmap.CompressFormat.PNG, 100, outputStream);  
    return outputStream.toByteArray();  
}
```

```
database.insertData("Sofa carpet", "Torned", convertPhoto());
```

09

## CONTENT PROVIDER

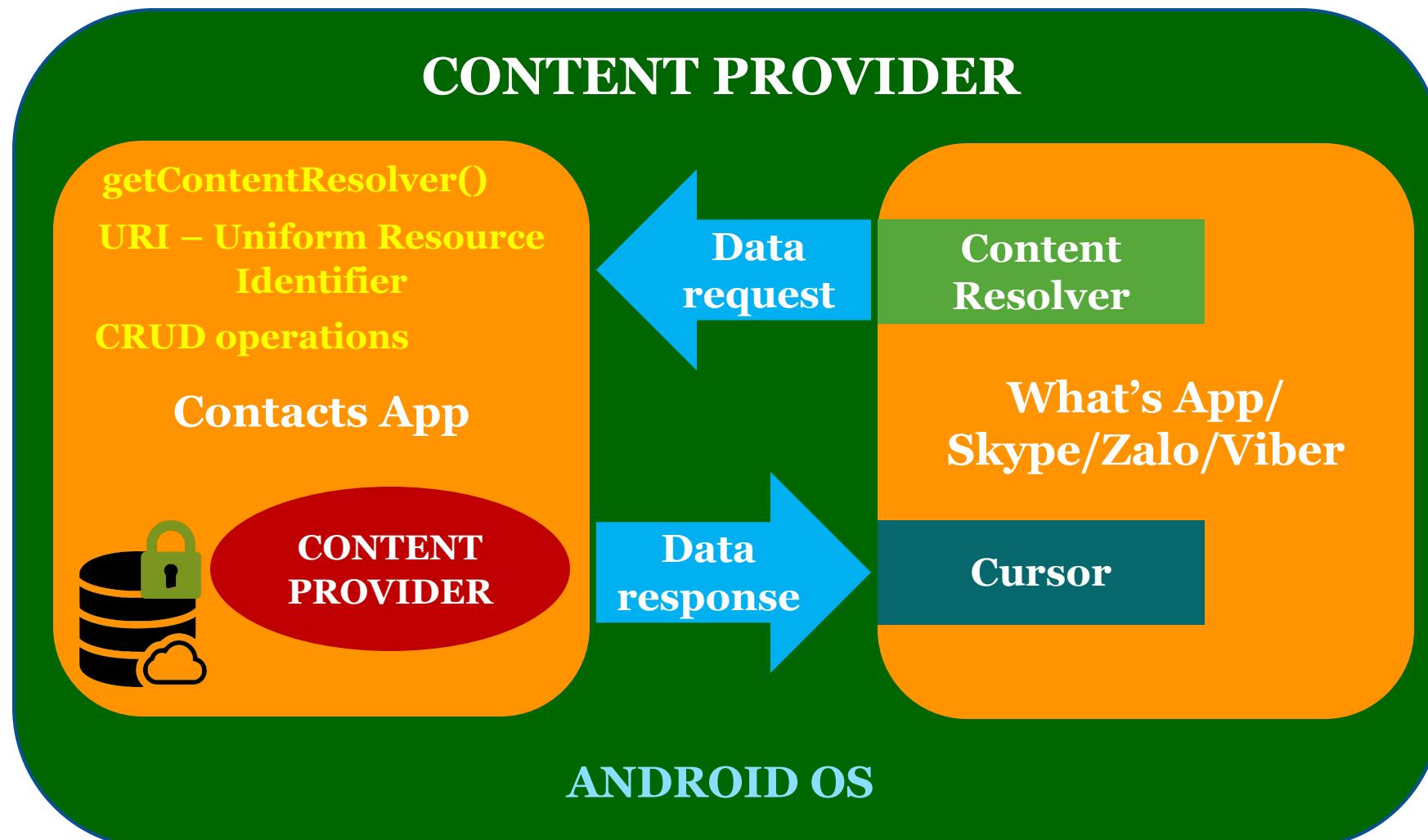
# NỘI DUNG

1. Giới thiệu về ContentProvider trong Android
2. Sử dụng ContentProvider
  - *Đọc và lưu danh bạ*
  - *Truy xuất dữ liệu nhật ký cuộc gọi*
  - *Truy xuất dữ liệu sms*
  - *Truy xuất dữ liệu Media*

# 1. Giới thiệu về Content Provider

- **ContentProvider** là một *thành phần quản lý truy cập dữ liệu*, cung cấp các phương thức khác nhau để các ứng dụng có thể truy cập dữ liệu từ những ứng dụng khác thông qua đối tượng *ContentResolver*.
- Một ứng dụng có thể quản lý quyền truy cập đến dữ liệu được lưu bởi ứng dụng đó thông qua ContentProvider.
- Dữ liệu được quản lý bởi ContentProvider giống như csdl quan hệ → các thao tác *truy vấn, thêm mới, cập nhật, xóa* được thực hiện qua các phương thức: *query()*, *insert()*, *update()*, *delete()*.

# 1. Giới thiệu về Content Provider



## 2. Sử dụng ContentProvider

» Vd: Đọc dữ liệu danh bạ điện thoại

Request  
permission

```
<uses-permission android:name="android.permission.READ_CONTACTS" />
```

```
Uri uri =  
    ContactsContract.CommonDataKinds.Phone.CONTENT_URI;  
Cursor cursor = getContentResolver().query(uri, projection: null,  
    selection: null, selectionArgs: null, sortOrder: null);
```

```
while (cursor.moveToNext()) {  
    int nameIndex =  
        cursor.getColumnIndex(ContactsContract.  
            Contacts.DISPLAY_NAME);  
    String name = cursor.getString(nameIndex); //Get Name  
    int phoneIndex =  
        cursor.getColumnIndex(ContactsContract.CommonDataKinds.  
            Phone.NUMBER);  
    String phone = cursor.getString(phoneIndex); //Get Phone Number  
    //Todo something ...  
}  
cursor.close();
```

## 2. Sử dụng ContentProvider

### »» Vd: Lưu danh bạ điện thoại

```
Uri addContactsUri = ContactsContract.Data.CONTENT_URI;  
//Add an empty contact and get the generated Id  
long rowContactId = getRawContactId();  
//Add contact name data  
insertContactDisplayName(addContactsUri, rowContactId, "Mr ....");  
//Add phone number  
insertContactPhoneNumber(addContactsUri, rowContactId,  
"0903778899");
```

```
private long getRawContactId() {  
    //Insert an empty contact  
    ContentValues values = new ContentValues();  
    Uri rawContactUri = getContentResolver().  
        insert(ContactsContract.RawContacts.CONTENT_URI, values);  
    return ContentUris.parseId(rawContactUri);  
}
```

## 2. Sử dụng ContentProvider

### »» Vd: Lưu danh bạ điện thoại

```
private void insertContactDisplayName(Uri addContactsUri, long  
rawContactId, String displayName)  
{  
    ContentValues values = new ContentValues();  
    values.put(ContactContract.Data.RAW_CONTACT_ID, rawContactId);  
    values.put(ContactContract.Data.MIMETYPE,  
        ContactContract.CommonDataKinds.  
        StructuredName.CONTENT_ITEM_TYPE);  
    values.put(ContactContract.CommonDataKinds.  
        StructuredName.GIVEN_NAME, displayName);  
    getContentResolver().insert(addContactsUri, values);  
}
```

## 2. Sử dụng ContentProvider

### »» Vd: Lưu danh bạ điện thoại

```
private void insertContactPhoneNumber(Uri addContactsUri, long  
rawContactId, String phoneNumber)  
{  
    ContentValues values = new ContentValues();  
    values.put(ContactContract.Data.RAW_CONTACT_ID, rawContactId);  
    values.put(ContactContract.Data.MIMETYPE,  
        ContactContract.CommonDataKinds.  
            Phone.CONTENT_ITEM_TYPE);  
    values.put(ContactContract.CommonDataKinds.Phone.NUMBER,  
        phoneNumber);  
    getContentResolver().insert(addContactsUri, values);  
}
```

## 2. Sử dụng ContentProvider

## » Vd: Đọc dữ liệu nhật ký cuộc gọi

# Request permission

```
<uses-permission android:name="android.permission.READ_CALL_LOG" />
```

## 2. Sử dụng ContentProvider

### »» Vd: Đọc dữ liệu nhật ký cuộc gọi

```
SimpleDateFormat dateFormat =  
    new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");  
  
while (cursor.moveToNext()) {  
    Date date = new Date(cursor.getLong(0));  
    String sDate = dateFormat.format(date);  
    String number = cursor.getString(1);  
    long duration = cursor.getLong(2);  
    //Todo something ...  
}  
  
cursor.close();
```

## 2. Sử dụng ContentProvider

» Vd: Đọc dữ liệu tin nhắn sms

Request  
permission

```
<uses-permission android:name="android.permission.READ_SMS" />
```

```
Uri uri = Uri.parse("content://sms/inbox");
Cursor cursor = getContentResolver().query(uri, projection: null,
selection: null, selectionArgs: null, sortOrder: null);
```

```
int phoneColumnIdx = cursor.getColumnIndex("address");
int timeColumnIdx = cursor.getColumnIndex("date");
int bodyColumnIdx = cursor.getColumnIndex("body");
while (cursor.moveToNext()){
    String phoneNumber = cursor.getString(phoneColumnIdx);
    Date date = new Date(cursor.getLong(timeColumnIdx));
    String content = cursor.getString(bodyColumnIdx);
    //Todo something ...
}
cursor.close();
```

## 2. Sử dụng ContentProvider

» Vd: Đọc dữ liệu Media

Request  
permission

```
<uses-permission  
    android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

```
String[] projection = {  
    MediaStore.MediaColumns.DISPLAY_NAME,  
    MediaStore.MediaColumns.DATE_MODIFIED,  
    MediaStore.MediaColumns.MIME_TYPE  
};  
CursorLoader cursorLoader = new CursorLoader(MainActivity.this,  
    MediaStore.Images.Media.EXTERNAL_CONTENT_URI,  
    projection, null, null, null);  
Cursor cursor = cursorLoader.loadInBackground();
```

## 2. Sử dụng ContentProvider

### »» Vd: Đọc dữ liệu Media

```
while (cursor.moveToNext()){  
    String name = cursor.getString(0);  
    Date date = new Date(cursor.getLong(1));  
    String type = cursor.getString(2);  
    //Todo something ...  
}  
cursor.close();
```

10

## BROADCAST RECEIVER

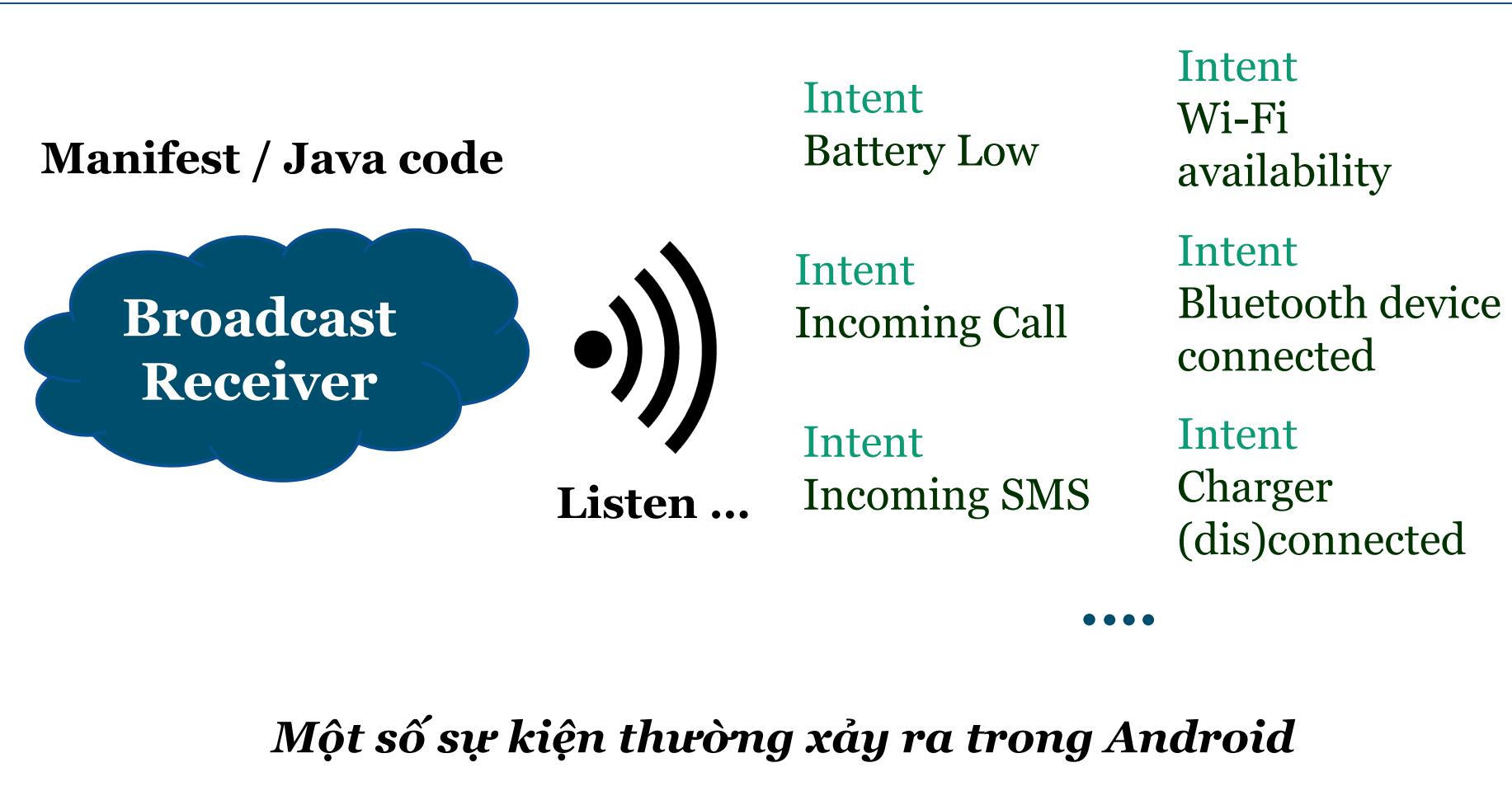
# NỘI DUNG

1. Giới thiệu Broadcast Receiver trong Android
2. Sử dụng Broadcast Receiver
3. Custom Broadcast Receiver

# 1. Giới thiệu Broadcast Receiver

- **Broadcast Receiver** là một *thành phần giúp lan truyền thông tin trong hệ thống* và chỉ những ứng dụng đã đăng ký mới nhận được thông tin.
- Broadcast Receiver giúp các ứng dụng được đăng ký *lắng nghe các sự kiện, trạng thái của hệ thống* (*trạng thái kết nối wifi, cảm nhận, tin nhắn đến, ...*) phát ra thông qua *Intent*.
- Broadcast Receiver có thể được tùy biến để truyền tải thông tin theo chủ đích của lập trình viên.

# 1. Giới thiệu Broadcast Receiver



## 2. Sử dụng BroadcastReceiver

### » Đăng ký

- **Cách 1:**  
*đăng ký trong Java code*

```
BroadcastReceiver receiver = new BroadcastReceiver() {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        //Todo something ....  
    }  
};
```

```
protected void onResume() {  
    super.onResume();  
    IntentFilter filter = new IntentFilter(ACTION);  
    registerReceiver(receiver, filter);  
}
```

```
protected void onPause() {  
    super.onPause();  
    unregisterReceiver(receiver);  
}
```

## 2. Sử dụng BroadcastReceiver

### » Đăng ký

- **Cách 2:** đăng ký trong Manifest

```
public class MyReceiver extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        //Todo something ....  
    }  
}
```

```
<receiver android:name=".MyReceiver">  
    <intent-filter>  
        <action android:name="android.provider.Telephony.  
                    SMS_RECEIVED"/>  
    </intent-filter>  
</receiver>
```

## 2. Sử dụng BroadcastReceiver

» Vd: Lắng nghe trạng thái Wi-Fi

Request  
permission

```
<uses-permission android:name="android.permission.  
ACCESS_NETWORK_STATE" />
```

```
@Override  
protected void onResume() {  
    super.onResume();  
    IntentFilter filter = new  
        IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);  
    registerReceiver(receiver, filter);  
}  
  
@Override  
protected void onPause() {  
    super.onPause();  
    unregisterReceiver(receiver);  
}
```

## 2. Sử dụng BroadcastReceiver

### » Vd: Lắng nghe trạng thái Wi-Fi

```
BroadcastReceiver receiver = new BroadcastReceiver() {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        ConnectivityManager connectivityManager = (ConnectivityManager)  
            context.getSystemService(CONNECTIVITY_SERVICE);  
        NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();  
        if(networkInfo != null && networkInfo.isConnected()) {  
            if(networkInfo.getType() == ConnectivityManager.TYPE_WIFI){  
                imvConnectionState.setImageResource(R.drawable.ic_action_wifi);  
                txtConnectionState.setText("Connected with WIFI");  
            } else if(networkInfo.getType() == ConnectivityManager.TYPE_MOBILE){  
                imvConnectionState.setImageResource(R.drawable.ic_action_cellular);  
                txtConnectionState.setText("Connected with Mobile Data");  
            }  
        } else {  
            imvConnectionState.setImageResource(R.drawable.ic_action_no);  
            txtConnectionState.setText("No internet connection");  
        }  
    }  
};
```

## 2. Sử dụng BroadcastReceiver

» Vd: Lắng nghe tin nhắn SMS

Request  
permission

```
<uses-permission android:name="android.permission.  
RECEIVE_SMS"/>  
<uses-permission android:name="android.permission.  
READ_SMS"/>
```

```
<receiver android:name=".SmsReceiver">  
    <intent-filter>  
        <action android:name="android.provider.Telephony.  
SMS_RECEIVED"/>  
    </intent-filter>  
</receiver>
```

## 2. Sử dụng BroadcastReceiver

### » Vd: Lắng nghe tin nhắn SMS

```
public class SmsReceiver extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        Bundle bundle = intent.getExtras();  
        Object[] arrMessages = (Object[]) bundle.get("pdus");  
        String phone, time, content;  
        Date date; byte[] bytes;  
        for(int i = 0; i<arrMessages.length; i++){  
            bytes = (byte[]) arrMessages[i];  
            SmsMessage message = SmsMessage.createFromPdu(bytes);  
            phone = message.getDisplayOriginatingAddress();  
            date = new Date(message.getTimestampMillis());  
            content = message.getMessageBody();  
            //Todo something ...  
        }  
    }  
}
```

### 3. Custom BroadcastReceiver

#### » TH 1: custom Broadcast & đăng ký Java code

##### App 1: Broadcast Sender

```
public void SendBroadCast(View view) {  
    Intent intent = new Intent("com.example.EXAMPLE_ACTION");  
    intent.putExtra("com.example.VALUE", "Broadcast Received");  
    sendBroadcast(intent);  
}
```

### 3. Custom BroadcastReceiver

#### » TH 1: custom Broadcast & đăng ký Java code

##### App 2: Broadcast Receiver

```
public class MyBroadcast extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        if("com.example.EXAMPLE_ACTION".  
                equals(intent.getAction())){  
            String receiverText =  
                intent.getStringExtra("com.example.VALUE");  
            Toast.makeText(context, receiverText,  
                Toast.LENGTH_LONG).show();  
        }  
    }  
}
```

### 3. Custom BroadcastReceiver

#### » TH 1: custom Broadcast & đăng ký Java code

##### App 2: Broadcast Receiver

```
public class MainActivity extends AppCompatActivity {  
    MyBroadcast receiver = new MyBroadcast();  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        IntentFilter filter = new  
            IntentFilter("com.example.EXAMPLE_ACTION");  
        registerReceiver(receiver, filter);  
    }  
    @Override  
    protected void onDestroy() {  
        super.onDestroy();  
        unregisterReceiver(receiver);  
    }  
}
```

### 3. Custom BroadcastReceiver

#### » TH 2: custom Broadcast & đăng ký Manifest

##### App 1: Broadcast Sender

```
Intent intent = new Intent();
ComponentName cn = new
ComponentName("com.example.custombroadcastreceiver",
"com.example.custombroadcastreceiver.
BroadcastReceiverExample");
intent.setComponent(cn);
sendBroadcast(intent);
```

##### App 2: Broadcast Receiver

```
<receiver android:name=".BroadcastReceiverExample"
android:exported="true"/>
```

### 3. Custom BroadcastReceiver

#### » TH 2: custom Broadcast & đăng ký Manifest

##### App 1: Broadcast Sender

```
Intent intent = new Intent("com.example.EXAMPLE_ACTION");
intent.setPackage("com.example.custombroadcastreceiver");
sendBroadcast(intent);
```

##### App 2: Broadcast Receiver

```
<receiver android:name=".BroadcastReceiverExample">
    <intent-filter>
        <action android:name="com.example.EXAMPLE_ACTION"/>
    </intent-filter>
</receiver>
```

### 3. Custom BroadcastReceiver

#### »» Sử dụng sendOrderedBroadcast

##### App 1: Broadcast Sender

```
Intent intent = new Intent("com.example.EXAMPLE_ACTION");
intent.setPackage("com.example.custombroadcastreceiver");
sendOrderedBroadcast(intent, receiverPermission: null);
```

### 3. Custom BroadcastReceiver

#### »» Sử dụng sendOrderedBroadcast

##### App 2: Broadcast Receiver

```
<receiver android:name=".OrderedReceiver1">
    <intent-filter android:priority="1"> ←
        <action android:name="com.example.EXAMPLE_ACTION"/>
    </intent-filter>
</receiver>

<receiver android:name=".OrderedReceiver2">
    <intent-filter android:priority="2"> ←
        <action android:name="com.example.EXAMPLE_ACTION"/>
    </intent-filter>
</receiver>
```

### 3. Custom BroadcastReceiver

#### »» Sử dụng sendOrderedBroadcast

##### App 2: Broadcast Receiver

```
public class OrderedReceiver1 extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        Log.i("Receiving app: ", "Ordered Receiver 1");  
    }  
}  
  
public class OrderedReceiver2 extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        Log.i("Receiving app: ", "Ordered Receiver 2");  
    }  
}
```

The results show in Logcat

I/Receiving app:: Ordered Receiver 2  
I/Receiving app:: Ordered Receiver 1

11

## MULTI-THREAD

# NỘI DUNG

1. Các khái niệm trong lập trình đa tuyến
2. Ngữ cảnh sử dụng đa tiến trình
3. Kỹ thuật lập trình đa tuyến trong Android
  - *Handler*
  - *AsyncTask*

# 1. Các khái niệm trong lập trình đa tuyến

## »» Process, Thread và Multiple Thread

- **Process** cung cấp tài nguyên cần thiết để thực thi chương trình và **phải có ít nhất 1 Thread** (tạm dịch là luồng) để thực thi.
- **Thread** là một thực thể trong Process, **1 thread được xem là 1 luồng để thực thi một chương trình**. Trong **1 process có thể có nhiều thread (multiple thread)** được thực thi đồng thời và mỗi thread đều có độ ưu tiên của nó.

→ ***Process = Program + State of all Threads executing in Program***

# 1. Các khái niệm trong lập trình đa tuyến

## »» Tạo Thread

- **Cách 1:** kế thừa (extends) từ class Thread

```
public class MyThread extends Thread {  
    @Override  
    public void run() {  
        super.run();  
        //TODO  
    }  
}
```

```
new MyThread().start();
```

# 1. Các khái niệm trong lập trình đa tuyến

## »» Tạo Thread

- **Cách 2:** thực thi (implements) interface Runnable

```
public class MyRunnable implements Runnable {  
    @Override  
    public void run() {  
        //TODO  
    }  
}
```

```
new Thread(new MyRunnable()).start();
```

# 1. Các khái niệm trong lập trình đa tuyến

## »» Main Thread và UI Thread, Worker Thread

- Trong android, khi ứng dụng được khởi chạy, hệ thống sẽ start một Thread ban đầu cùng với một Process, Thread này được gọi là **Main Thread**.
- Main Thread cũng thường được gọi là **UI Thread** vì thread này có nhiệm vụ xử lý *nạp giao diện ứng dụng* cũng như *tương tác với bộ công cụ Android UI* (Android UI Toolkit).
- **Worker Thread** là thread mà lập trình viên tạo thêm cho chương trình để thực thi một công việc nào đó không liên quan đến giao diện. Thread này cũng được gọi là **Background Thread**.

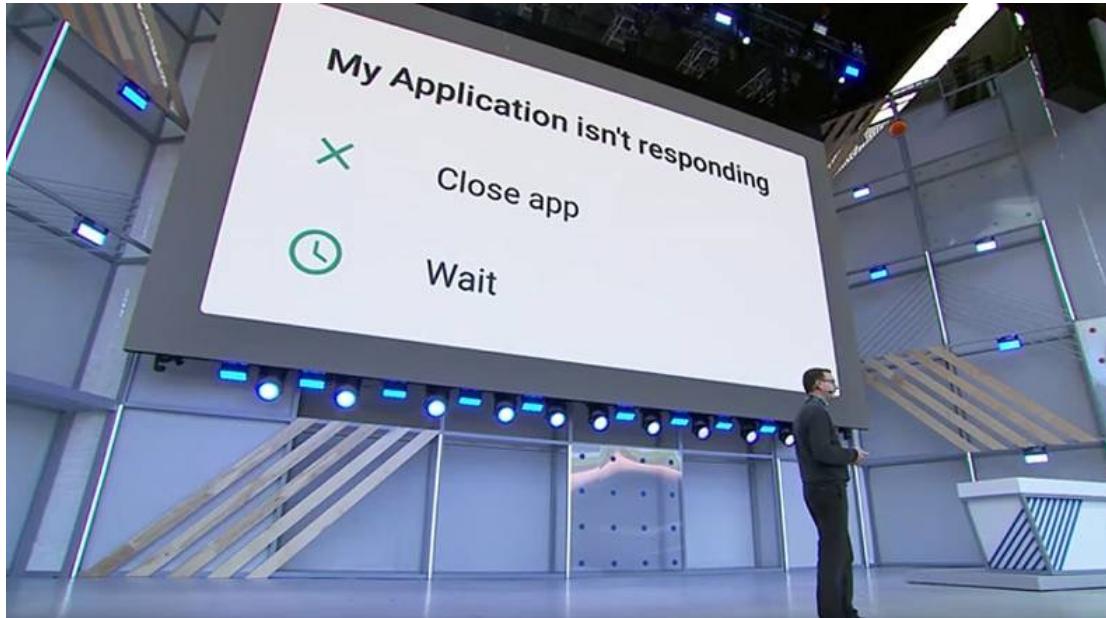
# 1. Các khái niệm trong lập trình đa tuyến

## »» Hiện tượng Application Not Responding - ANR

- Khi UI Thread thực thi nhiều công việc (task) và có 1 công việc đòi hỏi cần nhiều thời gian xử lý (kết nối Internet, truy vấn csdl, ...) thì khi đó UI sẽ bị block → xảy ra hiện tượng ANR (ứng dụng treo) nhưng thật ra chương trình vẫn đang được thực thi.
- Khi UI bị block khoảng vài giây (trung bình 5 giây) → hệ thống sẽ hiển thị hộp thoại thông báo cho phép người dùng *đóng ứng dụng* hoặc *chờ đợi*.

# 1. Các khái niệm trong lập trình đa tuyến

## »» Hiện tượng Application Not Responding - ANR



- Để hạn chế hiện tượng ANR, lập trình viên cần tuân thủ nguyên tắc sau:
  - ✓ *Không block UI Thread*
  - ✓ *Không được kết nối tới bộ công cụ Android UI từ một Thread không phải là UI Thread.*

## 2. Ngữ cảnh sử dụng đa tiến trình

- Các tác vụ đòi hỏi nhiều thời gian xử lý → **nên sử dụng đa tiến trình**. Ví dụ: kết nối và lấy dữ liệu từ Internet, truy vấn database, ....

➔ Không để người dùng phải đợi quá lâu (5 giây) khi tương tác 1 chức năng trên ứng dụng.

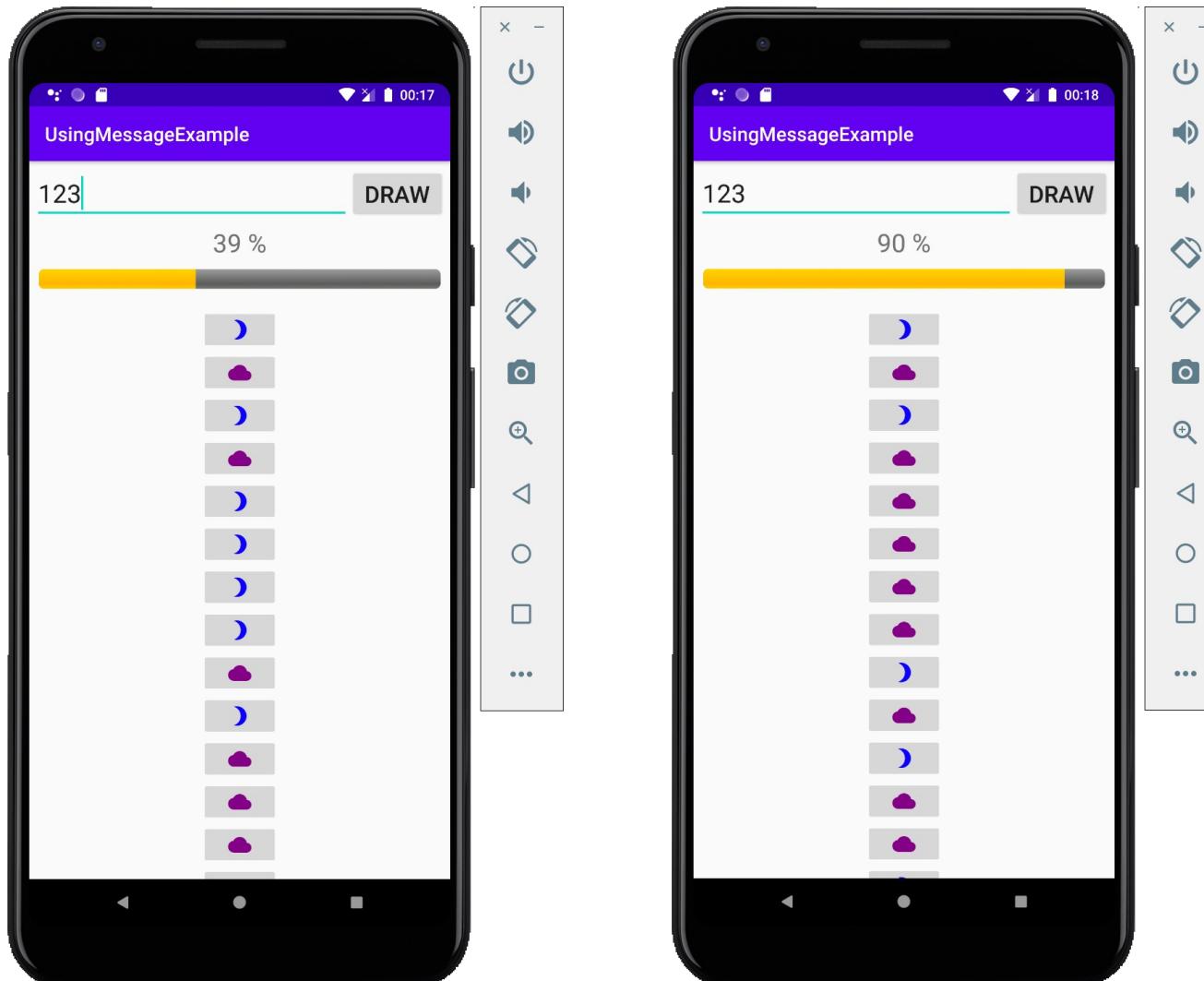
### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» Handler

- **Handler** là một đối tượng dùng để liên kết, trao đổi giữa các Thread với nhau, thường là Main Thread (UI Thread) với các Worker Thread (Background Thread).
- Các phương thức thường dùng khi sử dụng Handler:
  - ✓ ***sendMessage(Message)***
  - ✓ ***sendEmptyMessage(int)***
  - ✓ ***sendMessageAtTime(Message, long)***
  - ✓ ***sendMessageDelayed(Message, long)***
  - ✓ ***post(Runnable)***
  - ✓ ***postAtTime(Runnable, long)***
  - ✓ ***postDelayed(Runnable, Object, long)***

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» Handler



### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» Handler - sendMessage(Message) - 1

##### Main Thread / UI Thread

```
Handler handler = new Handler(new Handler.Callback() {
    @Override
    public boolean handleMessage(Message message) {
        int percent = message.arg1;
        if(percent == 100){
            Toast.makeText(MainActivity.this, "DONE",
                    Toast.LENGTH_LONG).show();
        }else{
            int value = (int) message.obj;
            //UPDATE UI ...
        }
        txtPercent.setText(percent + " %");
        pbPercent.setProgress(percent);
        return true;
    }
});
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» Handler - sendMessage(Message) - 1

##### Worker Thread / Brackground Thread

```
Thread thread = new Thread(new Runnable() {
    @Override
    public void run() {
        for(int i=1; i<=numb; i++){
            Message message = handler.obtainMessage();
            message.arg1 = i*100/numb; //Percent
            message.obj = random.nextInt(100);
            handler.sendMessage(message);
            SystemClock.sleep(100);
        }
    }
});  
thread.start();
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» Handler - sendMessage(Message) - 2

##### Main Thread / UI Thread

```
Handler handler = new Handler(Looper.getMainLooper()){
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what){
            case MSG_UPDATE_UI:
                int value = (int) msg.obj;
                //UPDATE UI ...
                txtPercent.setText(msg.arg1 + " %"); //percent
                pbPercent.setProgress(msg.arg1);
                break;
            case MSG_UPDATE_UI_DONE:
                //DONE
                break;
            default: break;
        }
    }
};
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» Handler - sendMessage(Message) - 2

##### Worker Thread / Brackground Thread

```
Thread thread = new Thread(new Runnable() {
    @Override
    public void run() {
        for(int i=1; i<=numb; i++){
            Message message = new Message();
            message.what = MSG_UPDATE_UI;
            message.arg1 = i*100/numb; //Percent
            message.obj = random.nextInt(100);
            handler.sendMessage(message);
            SystemClock.sleep(100);
        }
        handler.sendEmptyMessage(MSG_UPDATE_UI_DONE);
    }
});  
thread.start();
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» Handler - post(Runnable)

##### Main Thread / UI Thread

```
int numb = 0, percent, value;  
Handler handler = new Handler();  
Runnable foregroundThread = new Runnable() {  
    @Override  
    public void run() {  
        txtPercent.setText(percent + " %");  
        pbPercent.setProgress(percent);  
        //UPDATE UI ...  
    }  
};
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» Handler - post(Runnable)

##### Worker Thread / Brackground Thread

```
Thread backgroundThread = new Thread(new Runnable() {  
    @Override  
    public void run() {  
        for(int i=1; i<=numb; i++){  
            percent = i*100/numb; //Percent  
            value = random.nextInt(100);  
            handler.post(foregroundThread);  
            SystemClock.sleep(100);  
        }  
    }  
});  
backgroundThread.start();
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» **AsyncTask**

- **AsyncTask** là một lớp trừu tượng cho phép thực hiện khai báo các tiến trình xử lý ngầm và cập nhật giao diện một cách tự động.
- Các giai đoạn xử lý tác vụ trong AsyncTask được thực hiện thông qua các phương thức sau theo đúng trình tự:
  - ✓ **onPreExecute**: *tiến trình tiền xử lý*
  - ✓ **doInBackground**: *tiến trình xử lý ngầm*
  - ✓ **onProgressUpdate**: *tiến trình xử lý cập nhật*
  - ✓ **onPostExecute**: *tiến trình cập nhật UI → hoàn tất*

### 3. Kỹ thuật lập trình đa tuyến trong Android

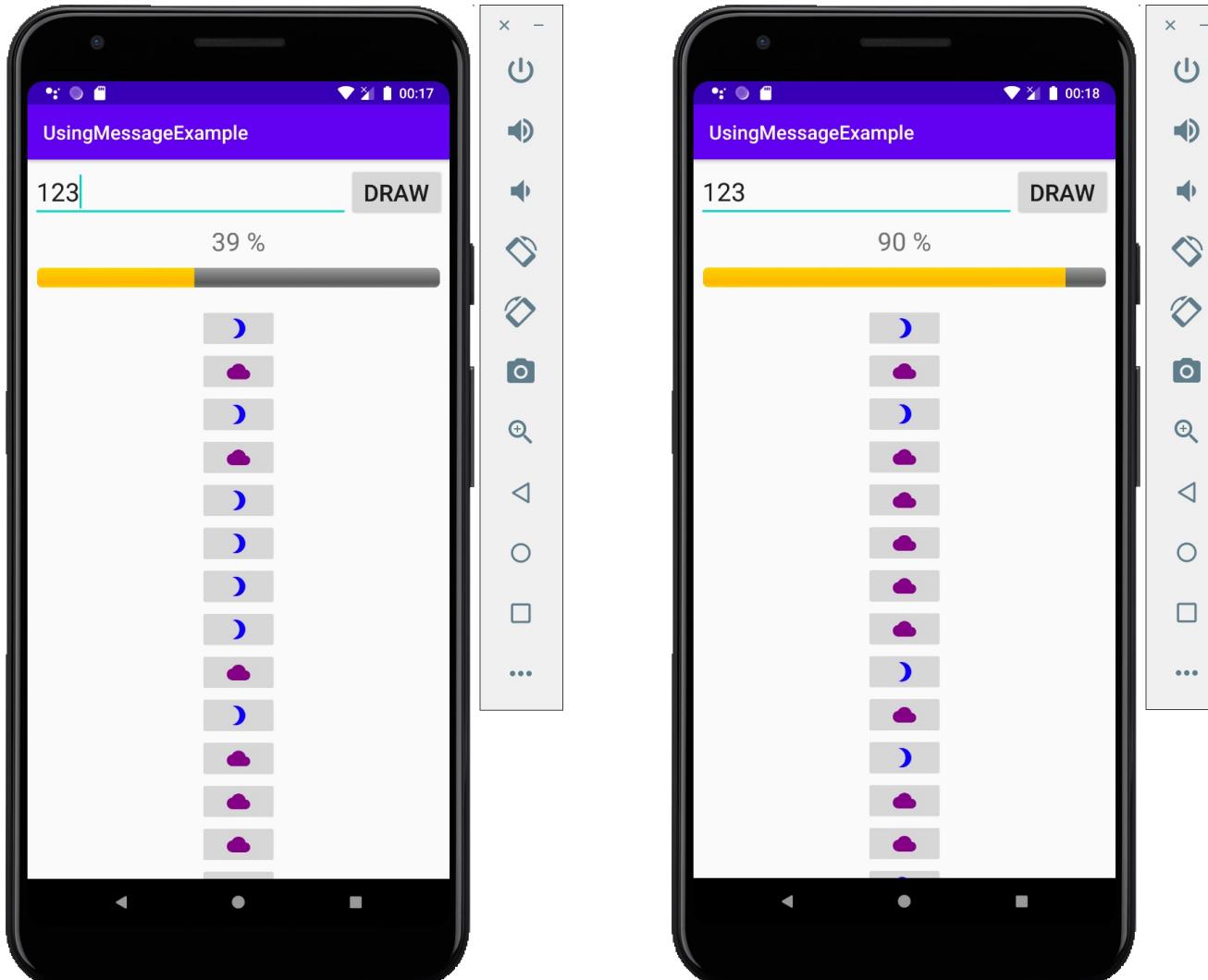
#### »» AsyncTask

```
public class MainActivity extends  
    AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle  
        savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        new MyTask().execute("string parameter");  
    }  
}
```

```
private class MyTask extends  
    AsyncTask<String, Integer, String>{  
  
    @Override  
    protected void onPreExecute() {  
        super.onPreExecute();  
    }  
  
    @Override  
    protected String doInBackground(String... strings) {  
        String myString = strings[0]; ←  
        int i = 0;  
        publishProgress(i); ←  
        return "result string";  
    }  
  
    @Override  
    protected void onProgressUpdate(Integer... values) {  
        super.onProgressUpdate(values);  
    }  
  
    @Override  
    protected void onPostExecute(String s) {  
        super.onPostExecute(s); ↑  
    }  
}
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» AsyncTask



### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» AsyncTask

```
private void drawUI() {  
    int num = Integer.parseInt(edtNumberOfControls.getText().toString());  
    MyAsyncTask myAsyncTask = new MyAsyncTask();  
    myAsyncTask.execute(num);  
}
```

```
class MyAsyncTask extends  
    AsyncTask<Integer, Integer, Void>{  
    @Override  
    protected void onPreExecute() {  
        super.onPreExecute();  
    }  
    @Override  
    protected Void doInBackground(Integer... integers) {  
        return null;  
    }  
    .....  
    @Override  
    protected void onProgressUpdate(Integer... values)  
    {  
        super.onProgressUpdate(values);  
    }  
    @Override  
    protected void onPostExecute(Void aVoid) {  
        super.onPostExecute(aVoid);  
    }  
}
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» AsyncTask

```
@Override  
protected void onPreExecute() {  
    super.onPreExecute();  
    txtPercent.setText("0 %");  
    layoutControls.removeAllViews();  
}  
  
@Override  
protected Void doInBackground(Integer... integers) {  
    int n = integers[0], percent, value;  
    for(int i=1; i<=n; i++){  
        percent = i*100/n;  
        value = random.nextInt(100);  
        publishProgress(percent, value);  
        SystemClock.sleep(100);  
    }  
    return null;  
}
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

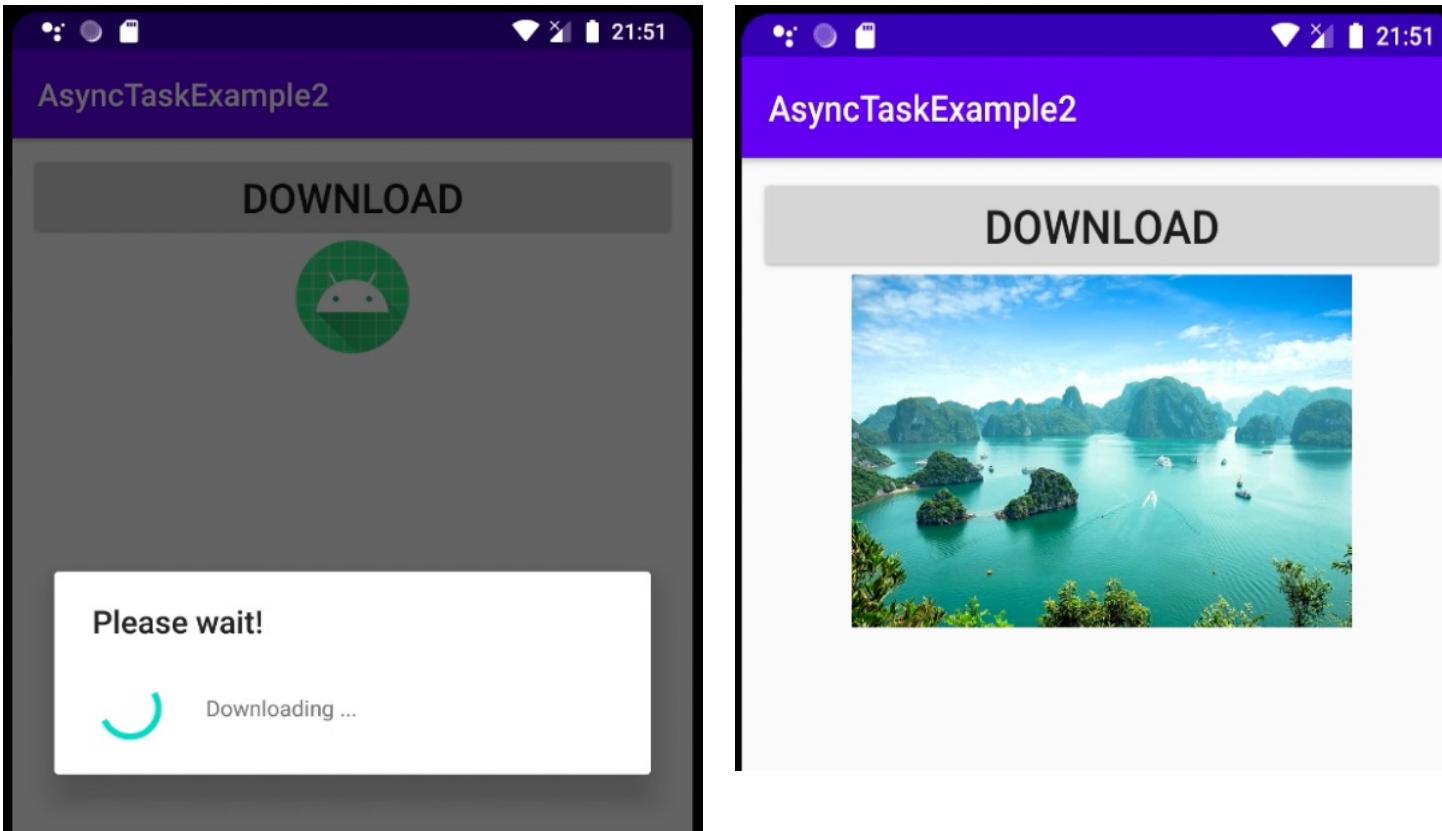
#### »» AsyncTask

```
@Override  
protected void onProgressUpdate(Integer... values) {  
    super.onProgressUpdate(values);  
    int percent = values[0];  
    int value = values[1];  
    txtPercent.setText(percent + " %");  
    pbPercent.setProgress(percent);  
    //UPDATE UI ...  
}  
  
@Override  
protected void onPostExecute(Void aVoid) {  
    super.onPostExecute(aVoid);  
    txtPercent.setText("DONE");  
}
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» AsyncTask

- **Ví dụ:** sử dụng AsyncTask load ảnh từ Internet cho ứng dụng



### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» AsyncTask

- **Ví dụ:** sử dụng AsyncTask load ảnh từ Internet cho ứng dụng

```
btnDownLoad.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        MyAsyncTask myAsyncTask = new MyAsyncTask();
        myAsyncTask.
            execute("https://media.worldnomads.com/explore/vietnam/"
            + "halong-bay-vietnam-from-above-gettyimages.jpg");
    }
});
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» AsyncTask

- **Ví dụ:** sử dụng AsyncTask load ảnh từ Internet cho ứng dụng

```
private class MyAsyncTask extends AsyncTask<String, Void, Bitmap>{  
  
    @Override  
    protected void onPreExecute() {  
        super.onPreExecute();  
    }  
    @Override  
    protected Bitmap doInBackground(String... strings) {  
        Bitmap bitmap = null;  
        ....  
        return bitmap;  
    }  
    @Override  
    protected void onProgressUpdate(Void... values) {  
        super.onProgressUpdate(values);  
    }  
    @Override  
    protected void onPostExecute(Bitmap bitmap) {  
        super.onPostExecute(bitmap);  
    }  
}
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» AsyncTask

- **Ví dụ:** sử dụng AsyncTask load ảnh từ Internet cho ứng dụng

```
@Override  
protected void onPreExecute() {  
    super.onPreExecute();  
    progressDialog.show();  
}  
  
@Override  
protected Bitmap doInBackground(String... strings) {  
    Bitmap bitmap = null;  
    try{  
        URL url = new URL(strings[0]);  
        HttpURLConnection connection = (HttpURLConnection)  
            url.openConnection();  
        bitmap = BitmapFactory.decodeStream(connection.getInputStream());  
    }catch (Exception e){  
        Log.e("Error: ", e.toString());  
    }  
    return bitmap;  
}
```

### 3. Kỹ thuật lập trình đa tuyến trong Android

#### »» AsyncTask

- **Ví dụ:** sử dụng AsyncTask load ảnh từ Internet cho ứng dụng

```
@Override  
protected void onProgressUpdate(Void... values) {  
    super.onProgressUpdate(values);  
}
```

```
@Override  
protected void onPostExecute(Bitmap bitmap) {  
    super.onPostExecute(bitmap);  
    progressDialog.dismiss();  
    imvPhoto.setImageBitmap(bitmap);  
}
```

