

# TỔNG QUAN VỀ LẬP TRÌNH ANDROID

---

GV: TRƯƠNG BÁ THÁI

Email: [truongbathai@tdc.edu.vn](mailto:truongbathai@tdc.edu.vn)

ĐT: 0932.577.765

# I MỤC TIÊU THỰC HIỆN

- Trình bày tổng quan về lập trình android
- Cài đặt được môi trường phát triển
- Tạo được ứng dụng đầu tiên
- Biết cách debug trong android studio

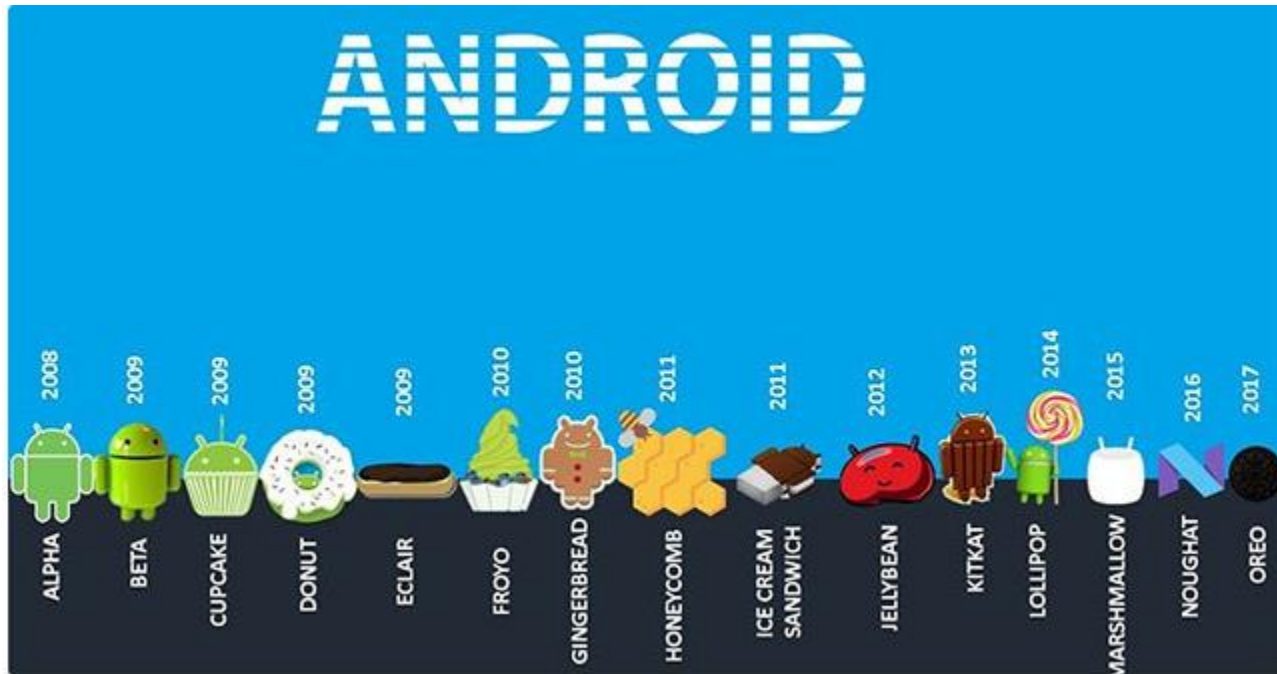


# Tổng quan về Android

# | Hệ điều hành Android

- Android là một Hệ điều hành mã nguồn mở và là một hệ điều hành dựa trên Linux cho các thiết bị mobile như Smartphone và máy tính bảng. Ban đầu Android được phát triển bởi Công ty Android với sự hỗ trợ tài chính từ Google, sau đó được Google mua lại vào năm 2005

# Các phiên bản của hệ điều hành Android





# Tại sao lập trình trên Android

# | Xu thế phát triển công nghệ di động

- Theo nhận định của nhiều chuyên gia công nghệ từ các hãng công nghệ hàng đầu như Microsoft, Google, IBM, ... Ba xu hướng tất trên toàn cầu hiện nay là: Social and Security (mạng xã hội và bảo mật), Mobility (công nghệ di động), Analytics Big Data (phân tích dữ liệu lớn), Cloud (Điện toán đám mây).

# | Thị trường thiết bị Android

- Trong tất cả các hệ điều hành dành cho di động hiện nay, có thể nói: Android đã mang lại một cuộc cách mạng thật sự cho các lập trình viên. Nổi bật với tính mở, đơn giản nhưng mạnh mẽ, không tốn phí cho bất cứ bản quyền nào và đặc biệt cộng đồng lập trình viên vô cùng lớn mạnh.



# Nhu cầu tuyển dụng lập trình viên Android

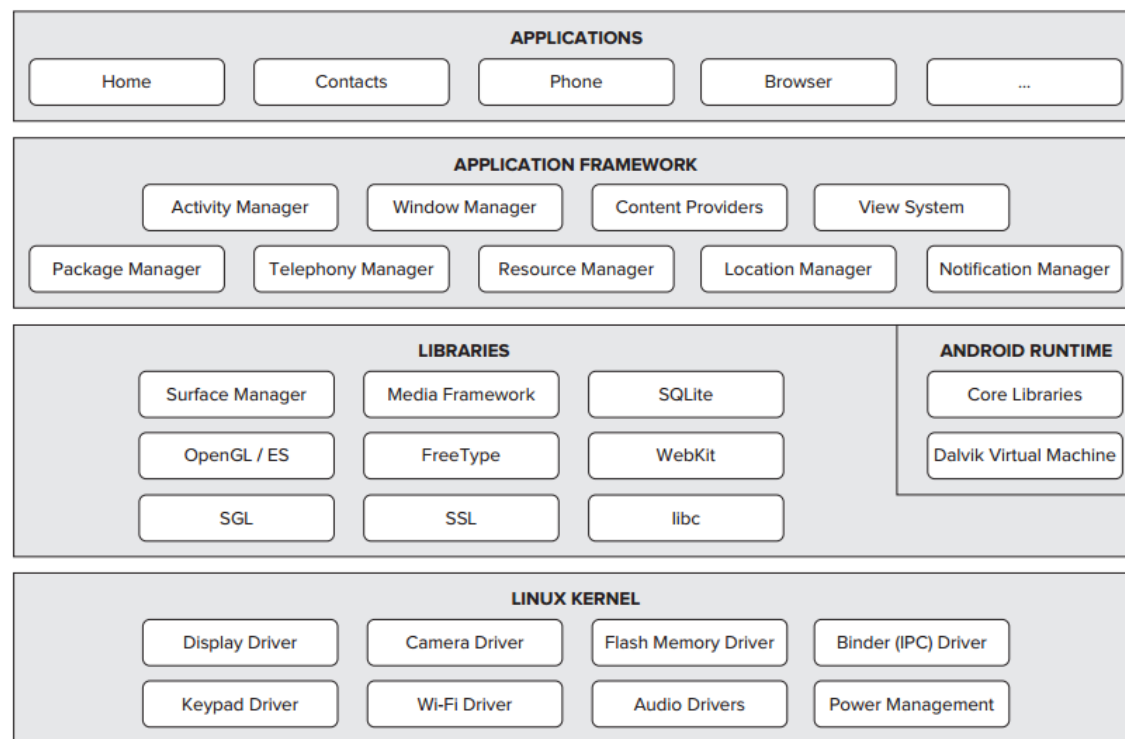
- Với xu thế phát triển công nghệ di động nhanh và mạnh như hiện nay, thị trường thiết bị Android chiếm vị trí cao nhất không chỉ ở Việt Nam mà trên toàn thế giới, thì nhu cầu sử dụng các ứng dụng cho các thiết bị Android là rất lớn. Vì vậy, nhu cầu tuyển dụng lập trình viên Android cũng rất lớn và sẽ tăng nhanh.



# Giới thiệu nền tảng phát triển Android

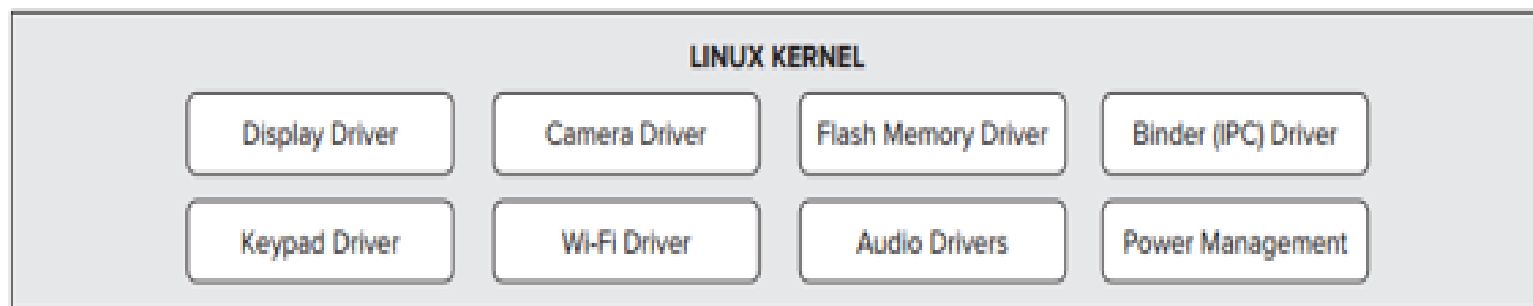
# Kiến trúc

- Hệ điều hành Android là một ngăn xếp của các thành phần phần mềm mà có thể đại khái phân chia thành 5 khu vực và 4 lớp chính.



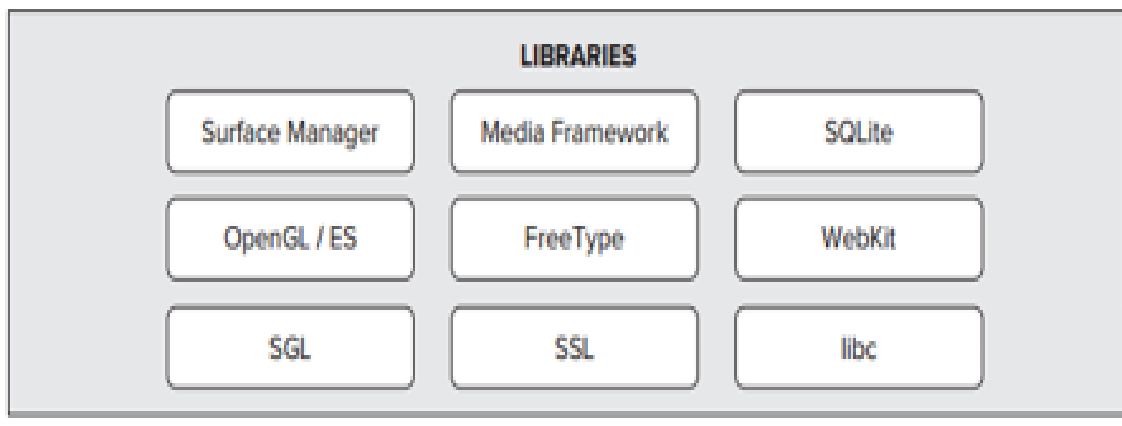
# Lớp Linux Kernel

- Linux Kernel là lớp thấp nhất. Nó cung cấp các chức năng cơ bản như quản lý tiến trình, quản lý bộ nhớ, quản lý thiết bị như: Camera, bàn phím, màn hình, ... Ngoài ra, nó còn quản lý mạng, driver của các thiết bị, điều này gỡ bỏ sự khó khăn về giao tiếp với các thiết bị ngoại vi



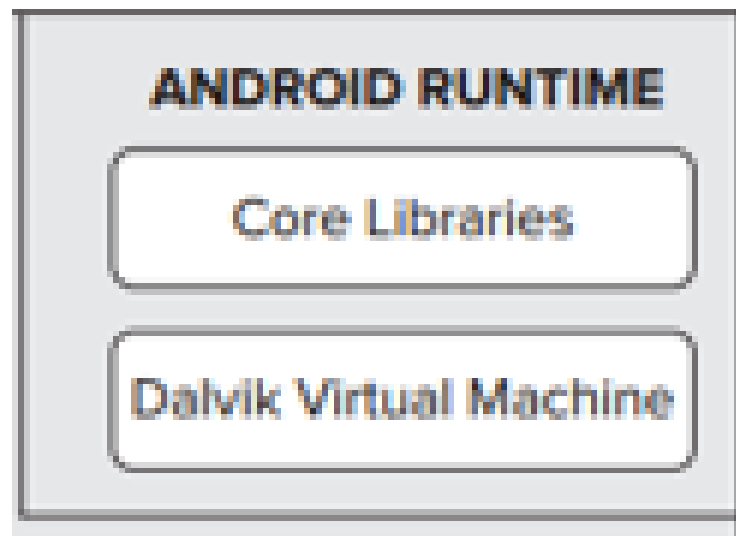
# Libraries trong Android

- Phía trên Linux Kernel là tập hợp các bộ thư viện mã nguồn mở WebKit, bộ thư viện nổi tiếng libc, cơ sở dữ liệu SQLite hữu ích cho việc lưu trữ và chia sẻ dữ liệu, bộ thư viện thẻ phát, ghi âm về âm thanh, hoặc video. Thư viện SSL chịu trách nhiệm cho bảo mật Internet



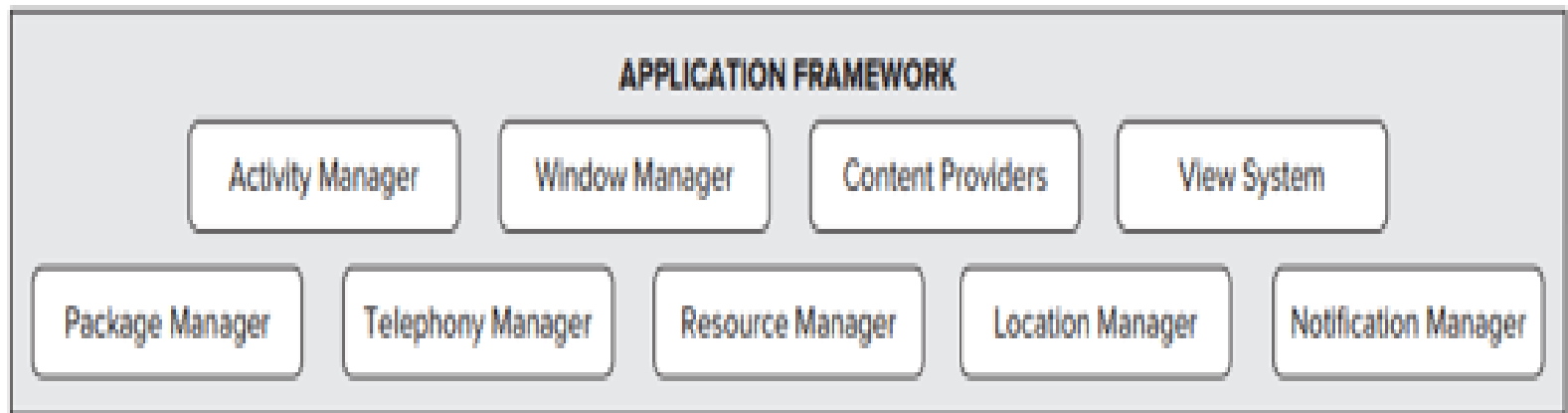
# | Android Runtime

- Đây là thành phần thứ 3 trong cấu trúc, thuộc về lớp 2 tính từ dưới lên. Phần này cung cấp một thành phần quan trọng gọi là Dalvik Virtual Machine là một máy ảo Java đặc biệt, được thiết kế tối ưu cho Android.



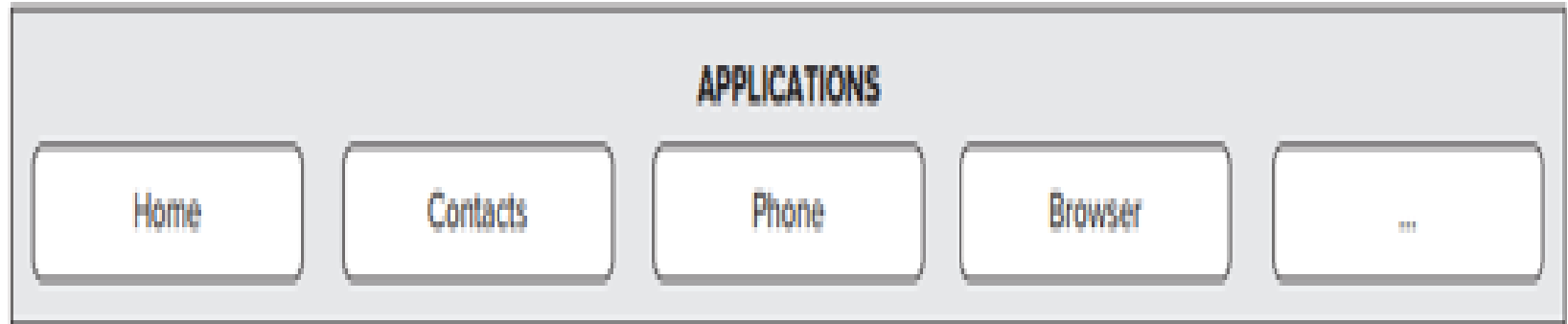
# | Application Framework

- Lớp Application Framework cung cấp nhiều dịch vụ cấp cao hơn cho các ứng dụng trong các lớp Java. Các lập trình viên cũng được phép sử dụng các dịch vụ này trong các ứng dụng của họ



# Application:

- Sẽ thấy tất cả các ứng dụng Android ở lớp trên cùng. Ứng dụng viết sẽ được cài đặt vào lớp này.
- Ví dụ của những ứng dụng này là Contacts Books, Browser, Games,







# Ngôn ngữ lập trình

# | Java

- Ngôn ngữ lập trình Java là một trong những ngôn ngữ ưa thích nhất khi phát triển ứng dụng Android. Một ngôn ngữ lập trình hướng đối tượng được phát triển tại Sun Microsystems (nay thuộc sở hữu của Oracle).

# | C++

- Đây là ngôn ngữ lập trình thích hợp và mạnh mẽ nhất khi xây dựng các ứng dụng di động cho Android và Windows chủ yếu dành cho lập trình cấp thấp.

# C#

- C # là một ngôn ngữ tuyệt vời. Microsoft đã nhìn thấy tiềm năng của Java và quyết định tạo một phiên bản tốt hơn của riêng họ.
- Sử dụng Xamarin.Android và Xamarin.iOS để tạo các ứng dụng di động bản địa với Visual Studio hoặc Xamarin Studio

# Kotlin

- Kotlin là một ngôn ngữ phát triển dựa vào Java Virtual Machine được phát triển bởi JetBrains5 Công ty phát triển IntelliJ IDE. Các tính năng thú vị của Kotlin đó là trực quan và dễ học, hầu hết các phần của Kotlin rất giống với những gì chúng ta đã biết, IDE Android studio đã được kết hợp Kotlin free.

# | HTML5 + CSS + JavaScript

- Ba ngôn ngữ lập trình này, ban đầu là trifecta cốt lõi cho việc phát triển front-end web, đã phát triển trở nên hữu dụng hơn. Bây giờ có thể thiết kế đa dạng nhiều loại apps, cả điện thoại di động và máy tính để bàn, chỉ cần sử dụng HTML5, CSS và JavaScript.
- Một sự lựa chọn khác là sử dụng React Native. Thư viện này có thể triển khai trên Android, iOS và nền tảng Windows chung. Nó được duy trì và sử dụng bởi Facebook.

# | Python

- Mặc dù Android không hỗ trợ phát triển Python bản địa nhưng vẫn có những công cụ cho phép tạo apps trên Python và sau đó chuyển đổi chúng thành các APK chạy thành công trên thiết bị Android.



# Môi trường phát triển ứng dụng



# | Giới thiệu Java JDK, Android SDK, Android Studio

- Android SDK (Software Development Kit) và JDK (Java Development Kit) là hai công cụ cần thiết để chúng ta có thể lập trình nên các ứng dụng Android.
- Android Studio được Google chính thức phát hành phiên bản đầu tiên Android Studio 0.1 vào tháng 5/2013.
- Là công cụ lập trình dựa trên nền IntelliJ.

# Các tính năng mạnh mẽ của Android Studio

- Hỗ trợ xây dựng dự án dạng Gradle.
- Hỗ trợ sửa lỗi nhanh và tái sử dụng cấu trúc phương thức.
- Cung cấp các công cụ kiểm tra tính khả dụng, khả năng hoạt động của ứng dụng, tương thích nền tảng...
- Hỗ trợ bảo mật mã nguồn và đóng gói ứng dụng.
- Trình biên tập giao diện cung cấp tổng quan giao diện ứng dụng và các thành phần, cho phép tùy chỉnh trên nhiều cấu hình khác nhau.
- Cho phép tương tác với nền Google Cloud.

# Thiết lập môi trường phát triển Android Studio

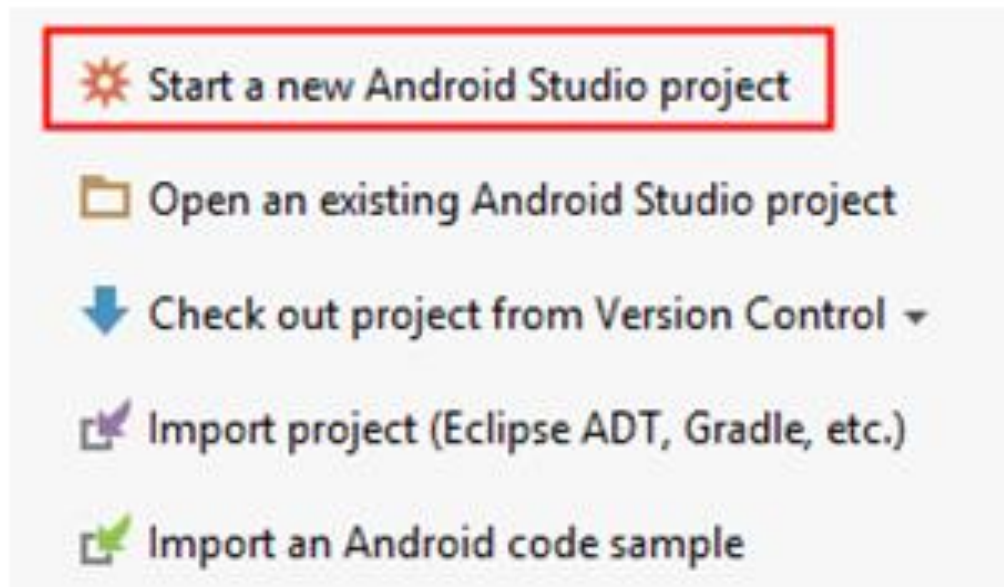
- Để bắt đầu viết ứng dụng với Android Studio, chúng ta cần tải và cài đặt hai bộ phần mềm sau:
- Java JDK:  
**<http://java.sun.com/javase/downloads/index.jsp>**  
(Cài đặt trước hết và nên chọn phiên bản mới nhất).
- Android Studio:  
**<http://developer.android.com/sdk/index.html>** - tải gói Android Studio.



Tạo ứng dụng đầu tiên

# | Khởi tạo dự án

- Bước 1: Tạo mới Project
- Chọn Start a new Android Studio project



# Đặt tên cho project

Create New Project

**New Project**  
Android Studio

**Configure your new project**

Application name:

Company Domain:

Package name:  [Edit](#)

☐ Include C++ Support

Project location:  [Browse](#)

[Previous](#) [Next](#) [Cancel](#) [Finish](#)

# | Bước 1.1: Đặt tên cho project

- **Application name:** Tên của ứng dụng, lưu ý phải viết HOA chữ cái đầu tiên của tên ứng dụng. Mặc định tên của ứng dụng cũng sẽ là tên Project.
- **Company Domain:** Tên domain của công ty. Dựa trên Application name và Company name, hệ thống sẽ tạo ra package name và thông tin này được sử dụng để đưa ứng dụng lên Google Play
- **Project location:** Đường dẫn trên máy dùng để lưu trữ ứng dụng.

# Bước 1.2: Chọn nền tảng để phát triển ứng dụng

Create New Project

## Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

☒ Phone and Tablet

Minimum SDK: API 19: Android 4.4 (KitKat)

Lower API levels target more devices, but have fewer features available.

By targeting API 19 and later, your app will run on approximately 73.9% of the devices that are active on the Google Play Store.

[Help me choose](#)

☐ Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ Android Auto

☐ Glass

Minimum SDK: Glass Development Kit Preview (API 19)

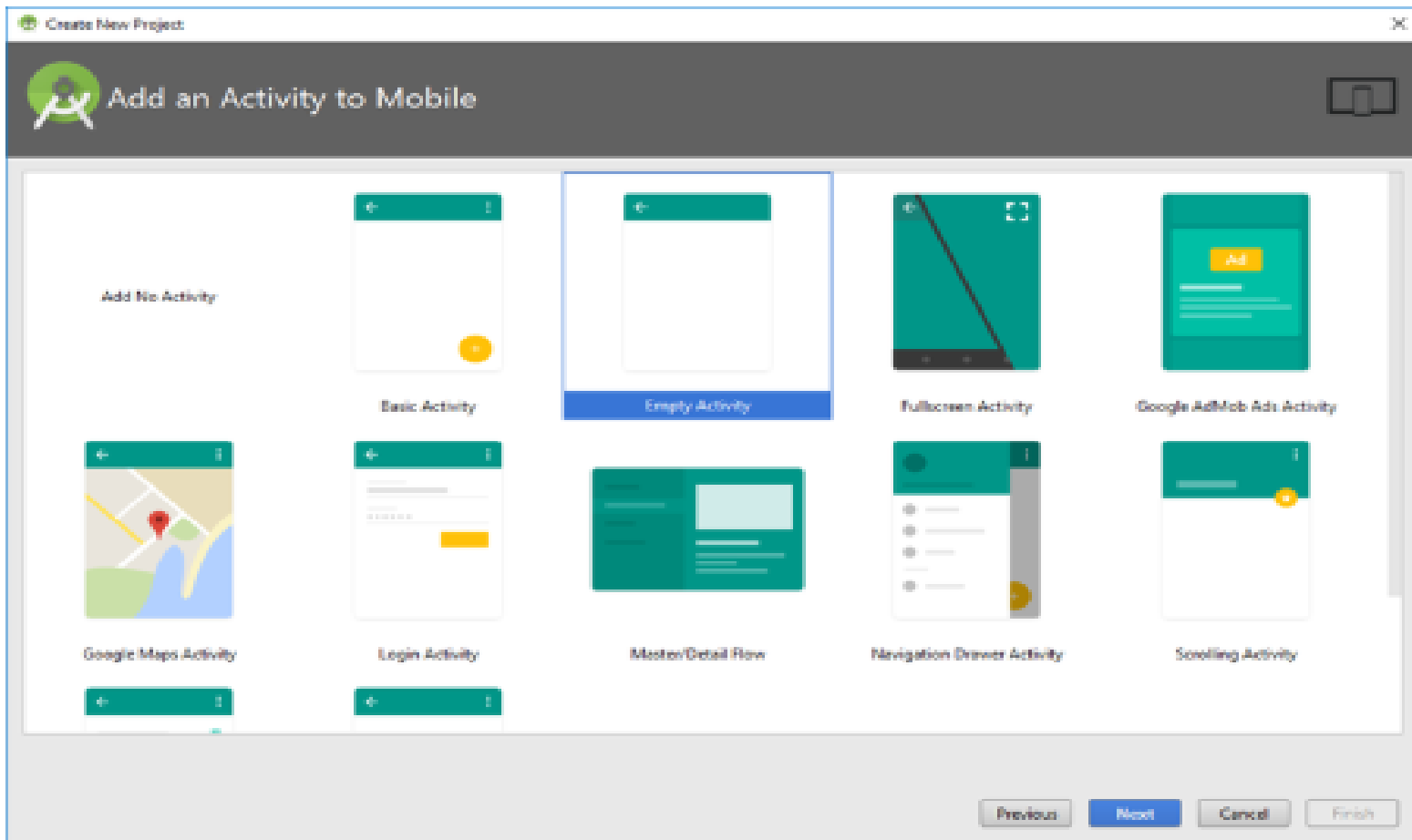
Previous Next Cancel Finish



## | Bước 1.2: Chọn nền tảng để phát triển ứng dụng

- **Phone and Tablet:** chọn mục này để xác định mình đang phát triển ứng dụng trên điện thoại và máy tính bảng. Sau đó chọn **Minimum SDK**, là phiên bản API thấp nhất mà ứng dụng có thể cài đặt.

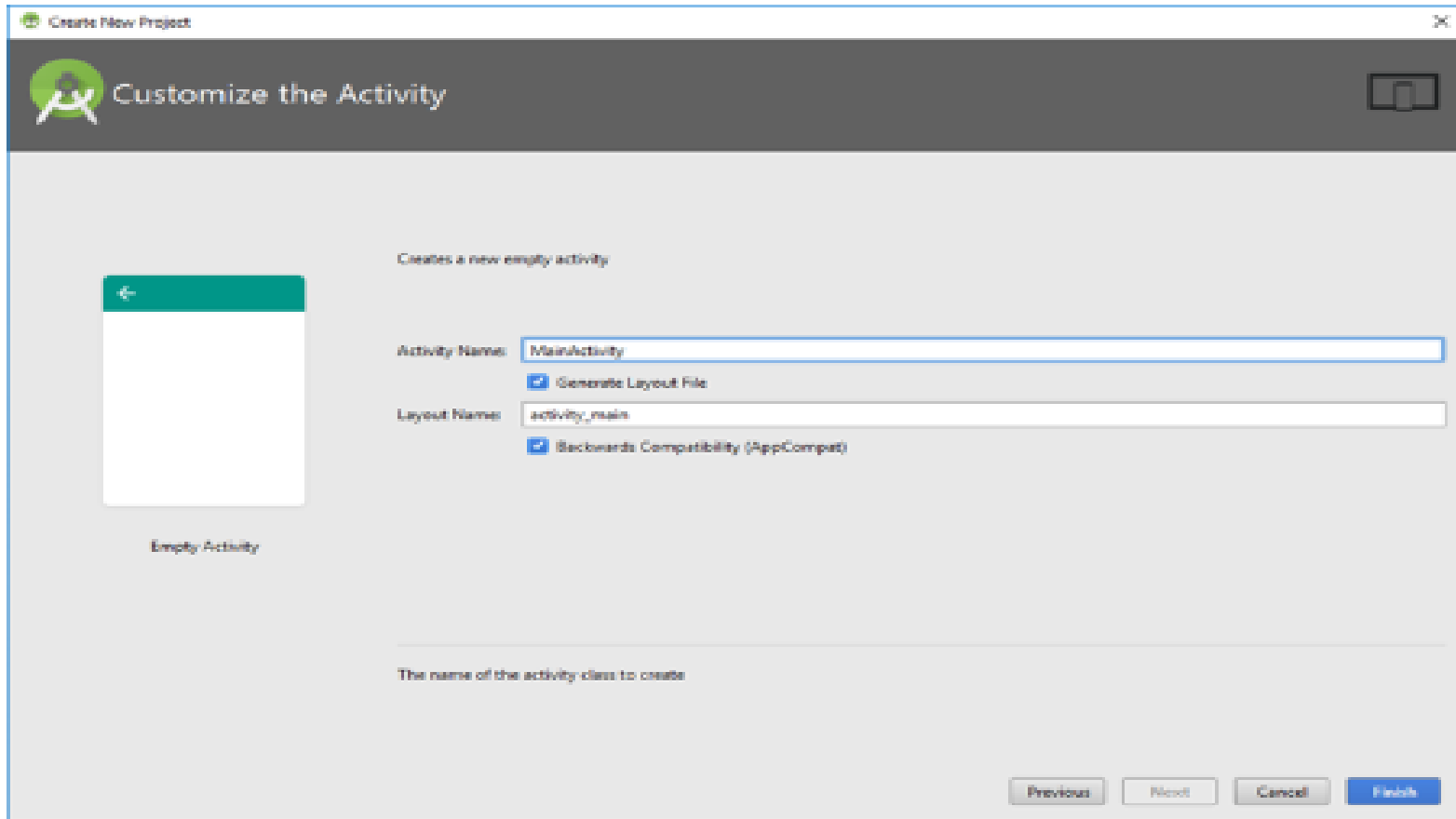
# Bước 1.3: Tạo mới và đưa Activity vào ứng dụng



## Bước 1.3: Tạo mới và đưa Activity vào ứng dụng

- Mỗi **Activity** là một màn hình giao diện người dùng, nơi người dùng tương tác, thực hiện một số thao tác tương ứng với chức năng của ứng dụng. Một ứng dụng có thể có nhiều **Activity** và sẽ có **Activity** hiển thị đầu tiên khi ứng dụng khởi động

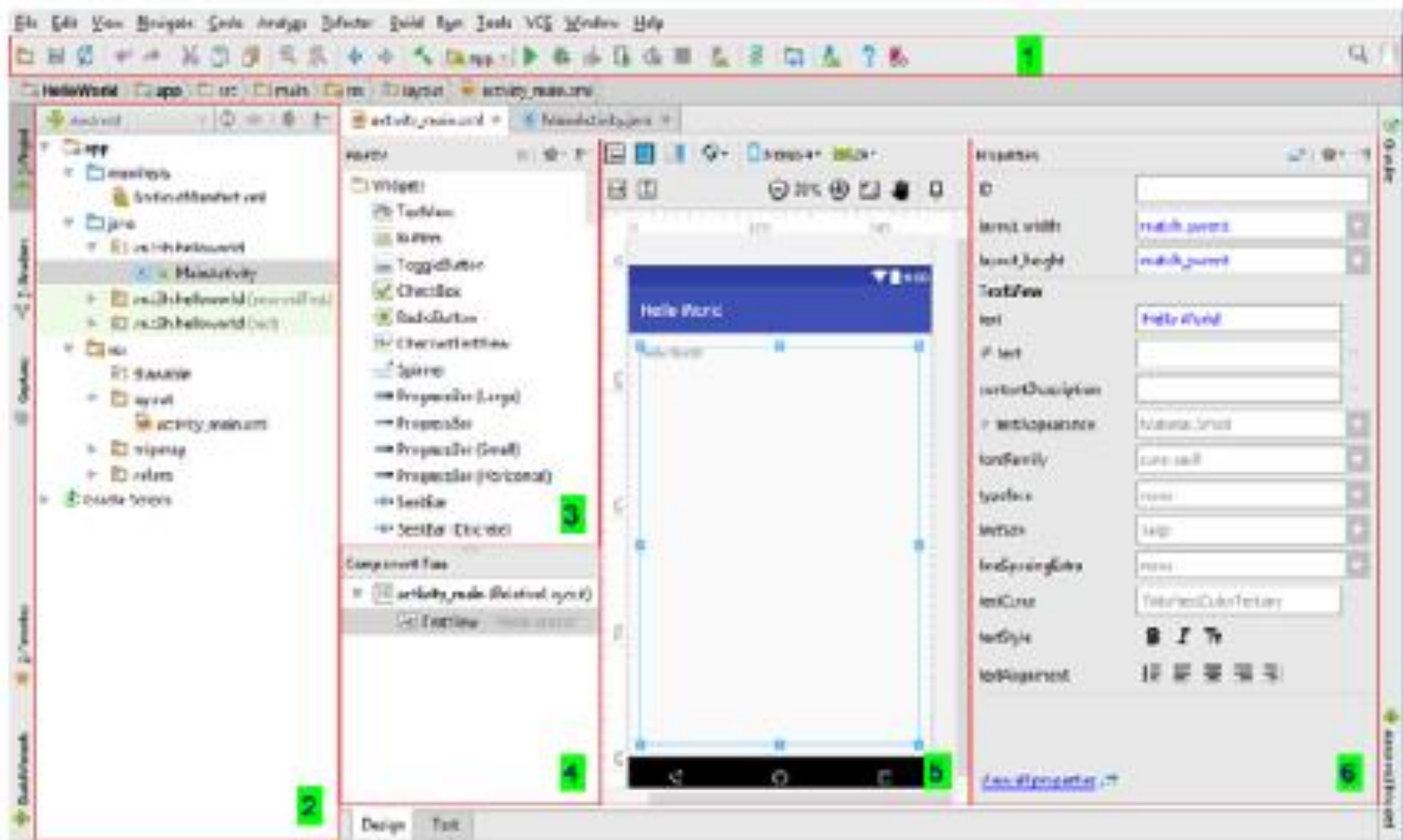
# Bước 1.4: Đặt tên cho Activity Name và Layout Name






## Bước 1.4: Đặt tên cho Activity Name và Layout Name

- Trong Android, tương ứng với mỗi **Activity** khi tạo ra sẽ có một tập tin lưu source code (**.java**) và một tập tin là mô tả giao diện của Activity (**.xml**). Trong trường hợp này, **Activity** của chúng ta là MainActivity nên hai tập tin đó là **MainActivity.java** và view layout sẽ có tên là **activity\_main.xml**. Nhấn nút **Finish** để hoàn tất các bước tạo ứng dụng đầu tiên.

# 1 Giao diện của Android Studio sẽ hiện ra như sau



# | *Vùng 1*

- Thanh công cụ giúp thao tác nhanh các chức năng thường dùng khi lập trình trong Android Studio.
- Chức năng Run 
- Debug ứng dụng 
- và quản lý máy ảo 

## Vùng 2: Cấu trúc hệ thống tài nguyên của ứng dụng

- **Thư mục manifests:** chứa thông tin cấu hình của ứng dụng
- **AndroidManifest.xml:** tập tin XML chứa tất cả các thông tin cấu hình dùng để build ứng dụng và các thành phần của ứng dụng (activity, service,...). Mỗi ứng dụng đều có một tập tin ***AndroidManifest.xml***. Trong ứng dụng, Activity nào muốn sử dụng đều bắt buộc phải có khai báo **AndroidManifest.xml**



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="vn.edu.tdc.chuong4vidu">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

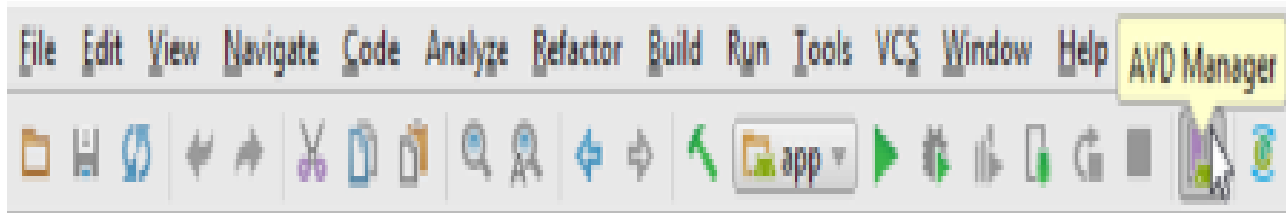
- **Thư mục java:** chứa tất cả các file mã nguồn .java của ứng dụng
- Tương ứng với mỗi **Activity** thì file mã nguồn sẽ chứa các xử lý trên Activity đó. Activity nào được khởi chạy đầu tiên khi ứng dụng hoạt động sẽ được khai báo đầu tiên trong tập tin **AndroidManifest.xml**.

- **Thư mục res:** chứa các tài nguyên của ứng dụng, bao gồm các tập tin hình ảnh, các thiết kế giao diện, thực đơn,... của ứng dụng

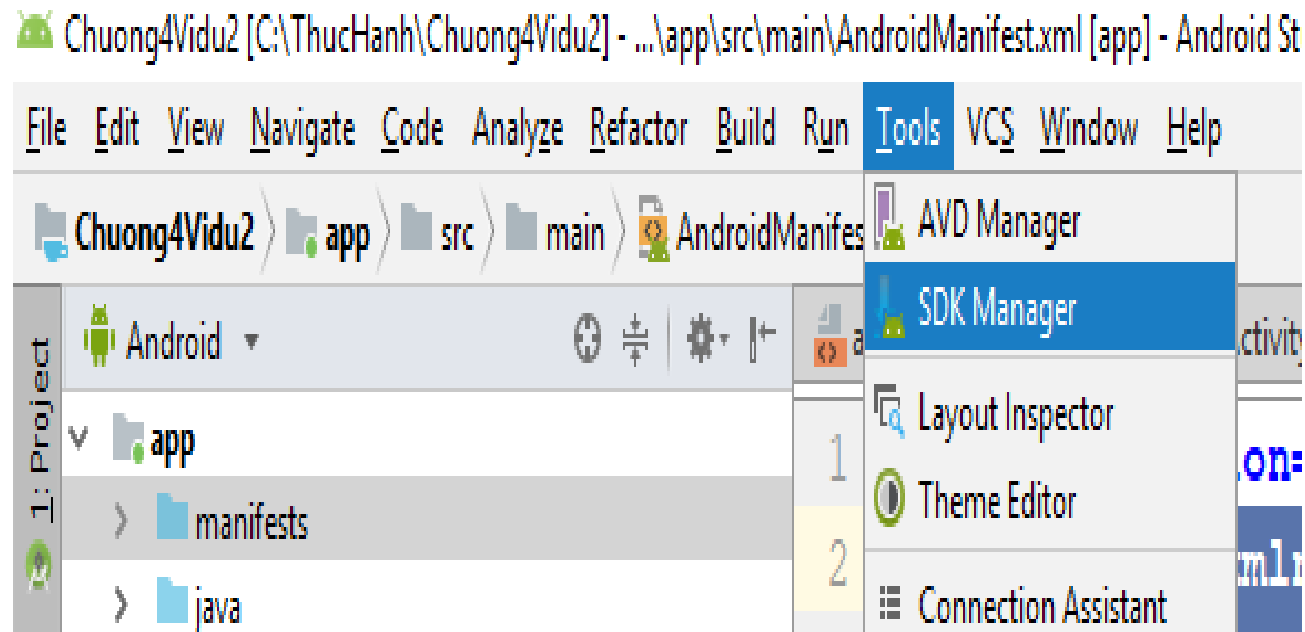
- Vùng 3: Danh sách các control mà Android Studio hỗ trợ
- **Vùng 4:** hiển thị giao diện theo cấu trúc cây giúp dễ dàng quan sát và lựa chọn control.
- **Vùng 5:** vùng giao diện của thiết bị cho phép kéo thả các control. Chúng ta có thể chọn cách hiển thị theo chiều nằm ngang, nằm đứng, phóng to, thu nhỏ, lựa chọn các loại thiết bị hiển thị...
- **Vùng 6:** Cửa sổ thuộc tính của control đang chọn, cho phép thiết lập các thuộc tính cần thiết.

## Bước 2: Tạo máy ảo

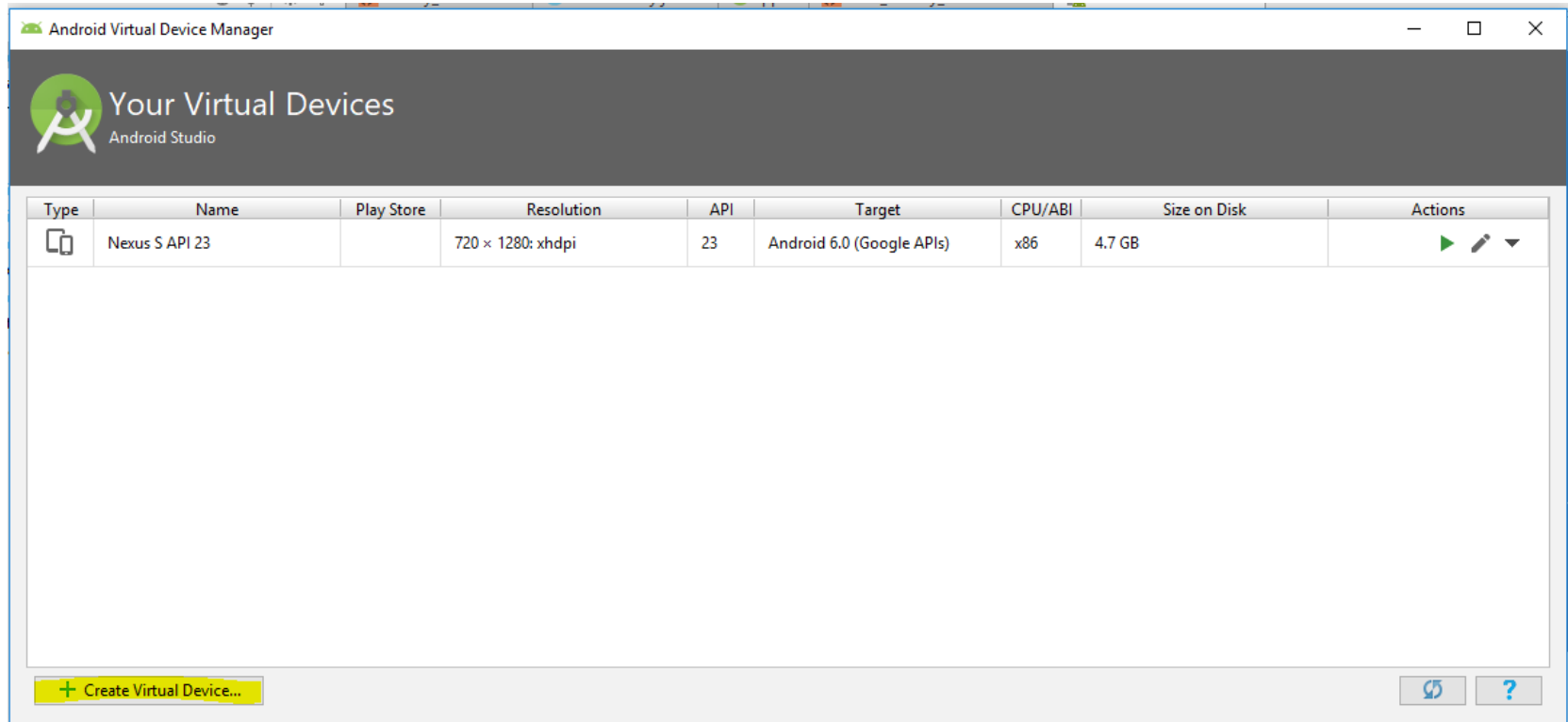
- Có 2 cách để tạo máy ảo
- Cách 1: Chọn biểu tượng AVD Manager trên thanh Toolbar



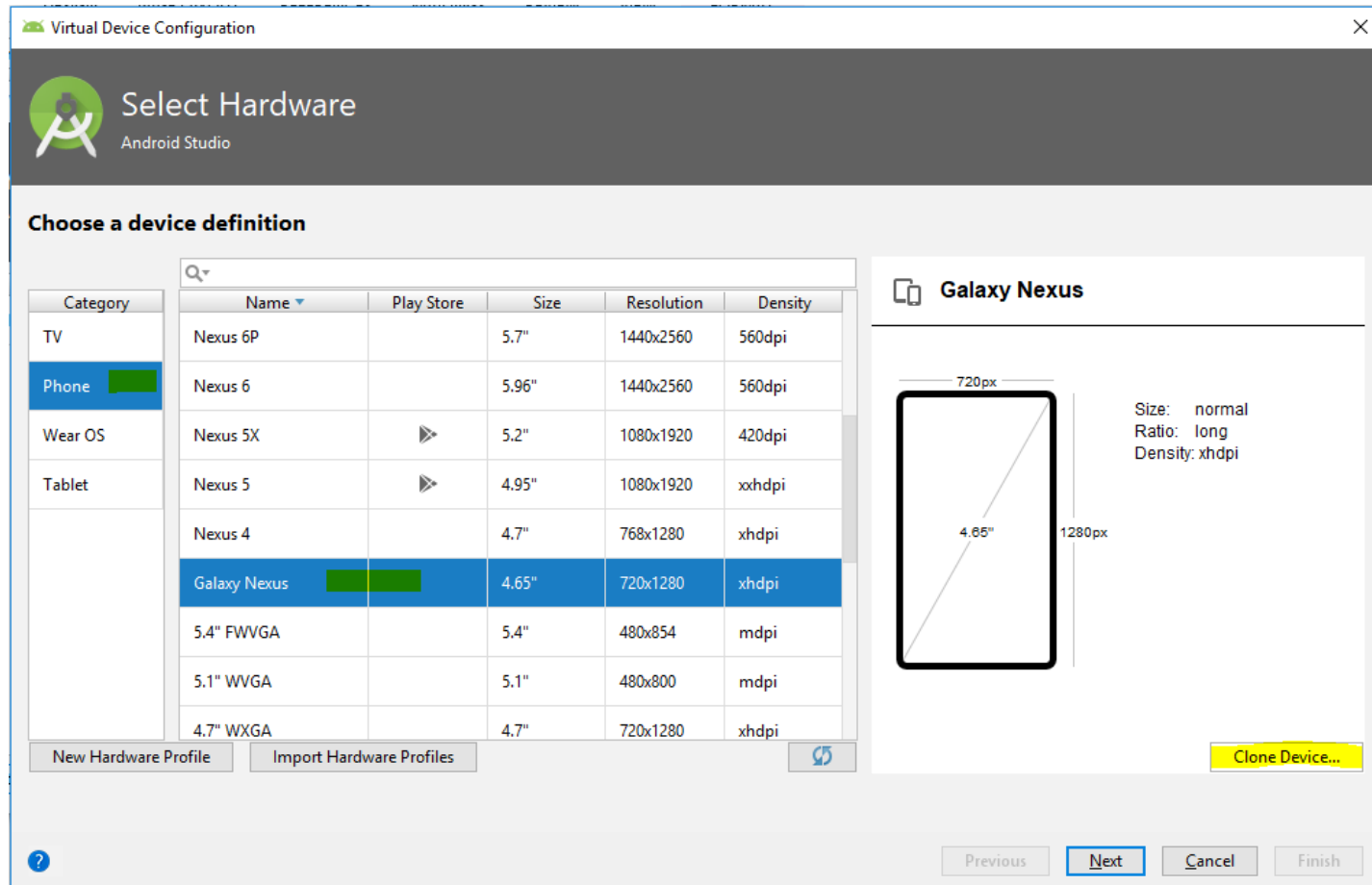
- *Cách 2:* Chọn tab Tools → Android → AVD Manager



# Bước 2.1: Chọn Create Virtual Device

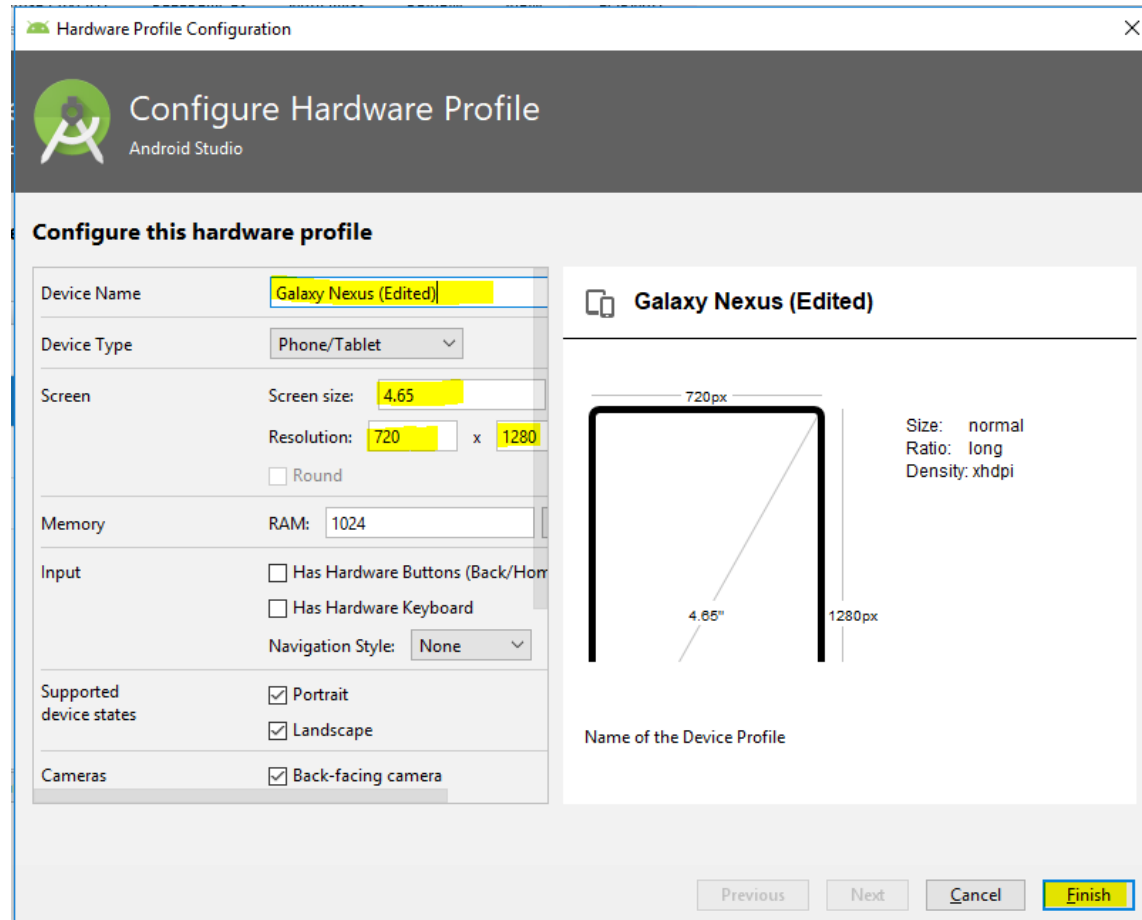


# Bước 2.2: Chọn máy ảo muốn tạo

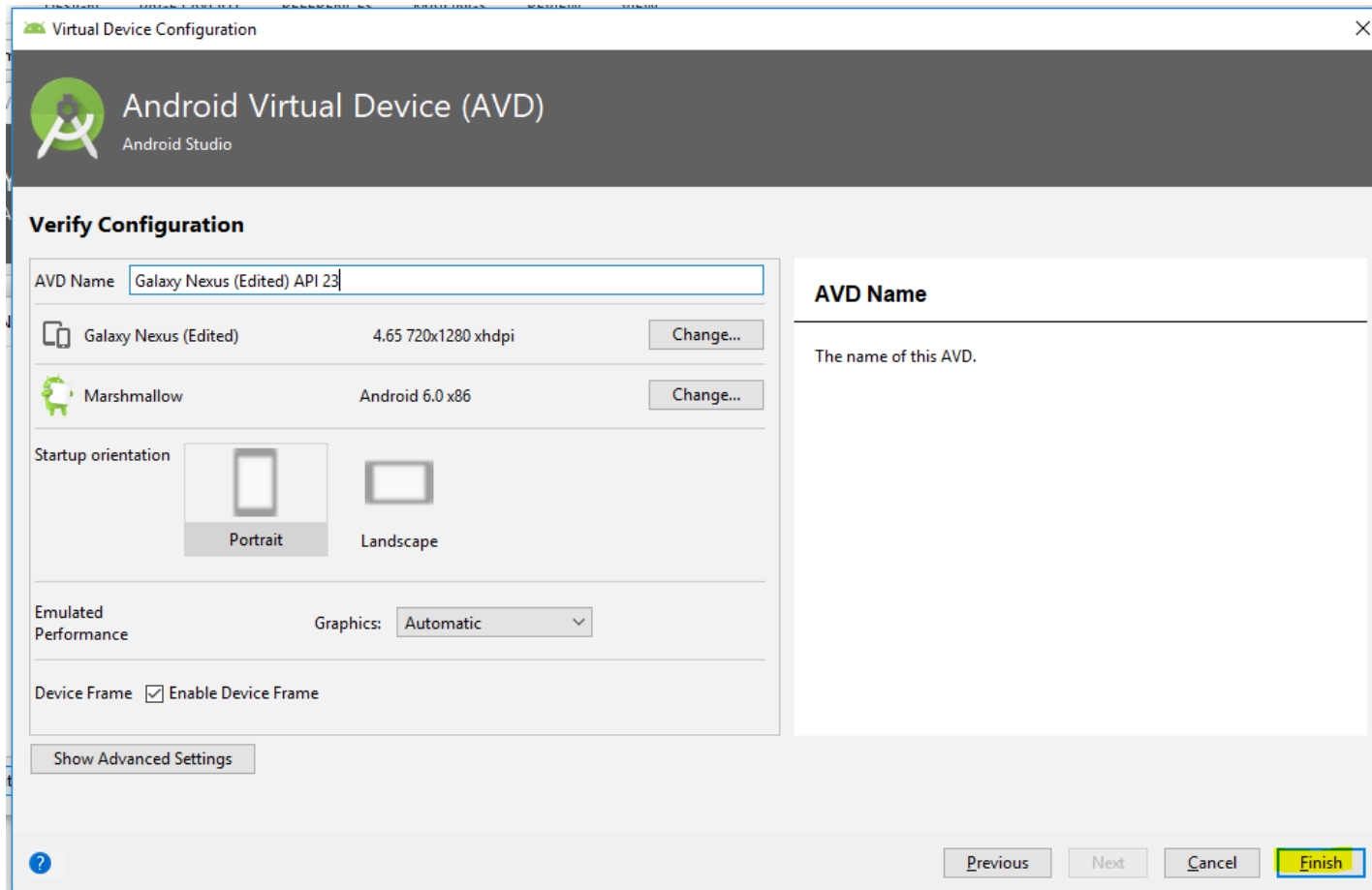




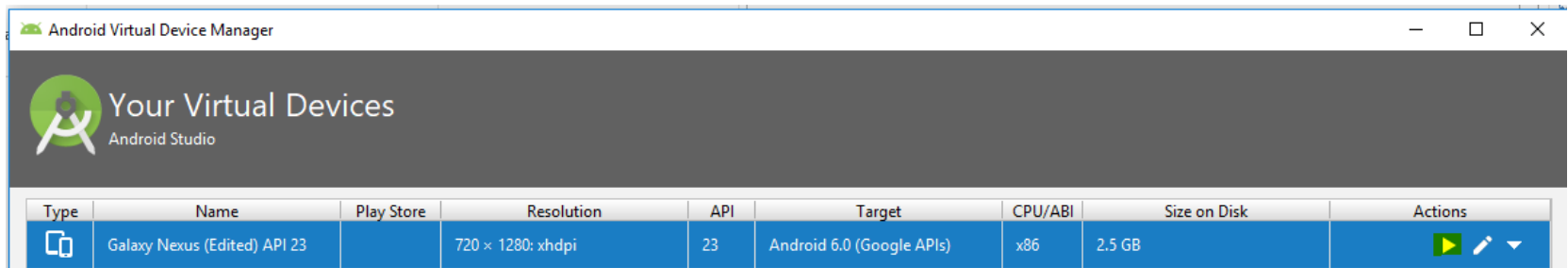
## Bước 2.3: Chọn hệ điều hành Android cho máy ảo



## Bước 2.4: Click vào Finish để hoàn thành quá trình tạo máy ảo



## Bước 2.5: Click vào biểu tượng để chạy máy ảo



## Bước 2.6: Máy ảo Android đã khởi động xong



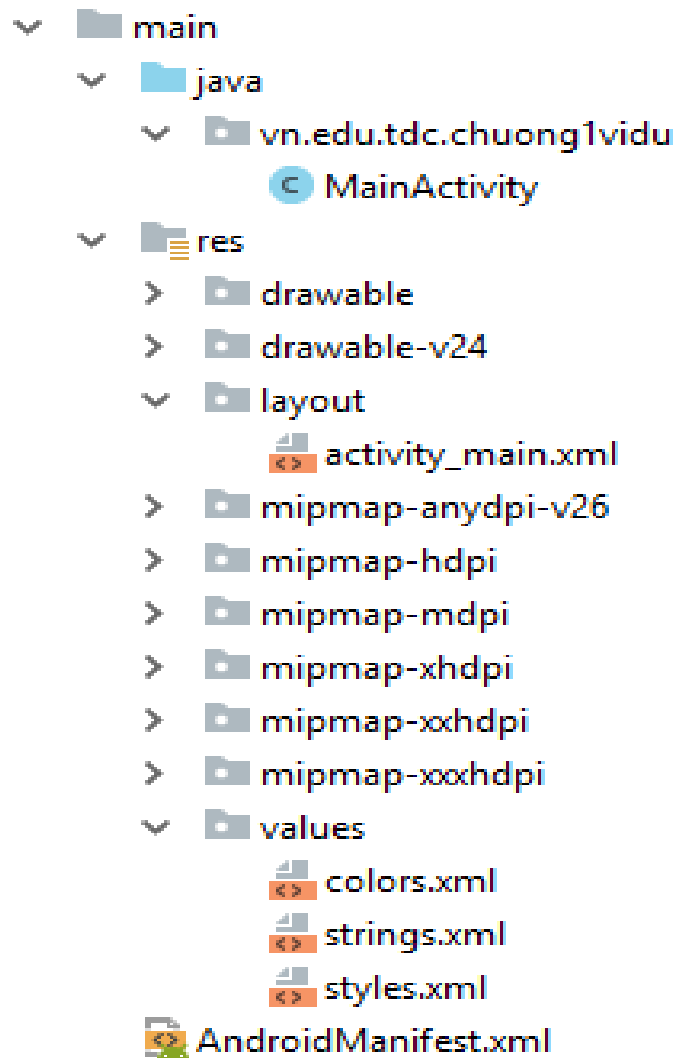
# | Bước 3: Build và thực thi ứng dụng



# | Kết quả



# Cấu trúc dự án



# | Main Activity code là một Java file với tên *MainActivity.java*

```
public class MainActivity extends AppCompatActivity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
```

```
    }
```

```
}
```



- ***strings.xml*** được đặt trong thư mục ***res/values*** và nó chứa tất cả text mà ứng dụng của sử dụng.

```
<resources>
```

```
    <string name="app_name">Chuong1ViDu</string>
```

```
</resources>
```

- *activity\_main.xml* là một layout file có sẵn trong thư mục *res/layout* mà được tham chiếu bởi ứng dụng của khi xây dựng giao diện.

```
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

# Resource trong Android

Thư mục	Kiểu Resource
<code>anim/</code>	Các XML file định nghĩa thuộc tính hiệu ứng. Chúng được lưu giữ trong thư mục <i>res/anim</i> và được truy cập từ lớp <b>R.anim</b> .
<code>color/</code>	Các XML file định nghĩa danh sách trạng thái của màu. Chúng được lưu giữ trong <i>res/color/</i> và được truy cập từ lớp <b>R.color</b> .
<code>drawable/</code>	Các Image file như <i>.png</i> , <i>.jpg</i> , <i>.gif</i> hoặc XML file mà được biên dịch vào trong bitmap, state list, shape, animation drawable. Chúng được lưu giữ trong <i>res/drawable/</i> và được truy cập từ lớp <b>R.drawable</b>

# | Resource trong Android

layout/	XML files that define a user interface layout. They are saved in <i>res/layout/</i> and accessed from the <i>R.layout</i> class.
menu/	XML file định nghĩa một UI layout. Chúng được lưu giữ trong <i>res/layout/</i> và được truy cập từ lớp <i>R.menu</i>
raw/	Các file riêng để lưu giữ trong dạng raw from. cần gọi <i>Resources.openRawResource()</i> với resource ID, mà là <i>R.raw.filename</i> để mở các raw file này

# | Resource trong Android

values/	<p>XML file chứa các giá trị đơn giản, ví dụ chuỗi, số nguyên và màu. Ví dụ, dưới đây là một số tên file qui ước cho các Resource có thể tạo trong thư mục này: –</p> <ul style="list-style-type: none"><li>✓ <i>arrays.xml</i> cho các mảng và được truy cập từ lớp <b>R.array</b></li><li>✓ <i>integers.xml</i> cho số nguyên và được truy cập từ lớp <b>R.integer</b></li><li>✓ <i>bools.xml</i> cho boolean, và được truy cập từ lớp <b>R.bool</b> class.</li><li>✓ <i>colors.xml</i> cho các giá trị màu, và được truy cập từ lớp <b>R.color</b></li><li>✓ <i>dimens.xml</i> cho các giá trị chiều, và được truy cập từ lớp <b>R.dimen</b></li><li>✓ <i>strings.xml</i> cho các giá trị chuỗi, và được truy cập từ lớp <b>R.string</b></li><li>✓ <i>styles.xml</i> cho các style, và được truy cập từ lớp <b>R.style</b></li></ul>
---------	---

# | Resource trong Android

xml/	Các XML file riêng có thể được đọc tại runtime bởi gọi phương thức <b>Resources.getXML()</b> . có thể lưu giữ các file cấu hình đa dạng tại đây, các file này sẽ được sử dụng tại runtime
------	---

# | AndroidManifest

- Tập tin ***AndroidManifest.xml*** chứa thông tin về package của ứng dụng, bao gồm các thành phần của ứng dụng như ***activities, services, broadcast receivers, content providers***
- Thực hiện một số nhiệm vụ khác nhau:
  - Cấp quyền một số phần trong ứng dụng
  - Khai báo các API mà ứng dụng sẽ sử dụng
  - Khai báo các thông tin về ứng dụng



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="vn.edu.tdc.chuong1vidu">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

# Các phân tử của tập tin *AndroidManifest.xml*

- **<manifest>:manifest** là phân tử đầu tin trong tập tin *AndroidManifest.xml*, có thuộc tính **package** mô tả tên **package** của class activity.
- **<application>: application** là phân tử con của manifest. Nó khai báo namespace, phân tử này chứa nhiều phân tử con được khai báo trong thành phần (component) của ứng dụng như: Activity ..

# Các thuộc tính của các phân tử thường được sử dụng

- **android:icon:** Thuộc tính này là nơi bạn thiết lập icon cho ứng dụng. Icon này sẽ xuất hiện trên màn hình chính của thiết bị. Hiện tại thì icon này đang được để mặc định là file .
- **android:label:** Thuộc tính này trỏ đến một giá trị *string* trong resource, hoàn toàn tương tự như cách bạn khai báo text cho *TextView* ở bài trước vậy. String này chính là tên ứng dụng

- **android:theme:** Thuộc tính này khá hay, nó giúp chúng ta định nghĩa “*giao diện chủ đề*” cho ứng dụng. Và vì nó hay quá nên mình sẽ không nói ở đây, mình dành hẳn một bài để nói về chủ đề này để cho các bạn hiểu rõ nhất về nó.
- ***ic\_launcher.png***: để trong thư mục ***res/mipmap/***. Cách sử dụng icon cho ứng dụng.

- **activity>: activity** là phần tử con của ứng dụng, một activity phải được định nghĩa trong tập tin AndroidManifest.xml. Nó có nhiều thuộc tính: **label, name, theme, launchMode**

- **android:label**: Tiêu đề của activity được hiển thị trên màn hình.
- **android:name**: Tên class của activity. Thuộc tính này bắt buộc.
- **<intent-filter>**: **intent-filter** là phần tử giúp cho hệ thống Android biết được ứng dụng của bạn có thể làm được những gì.
- **<action>**: Phần tử này một hành động cho intent-filter. intent-filter có ít nhất một action:
- **<category>**: Thêm tên category cho một intent-filter:



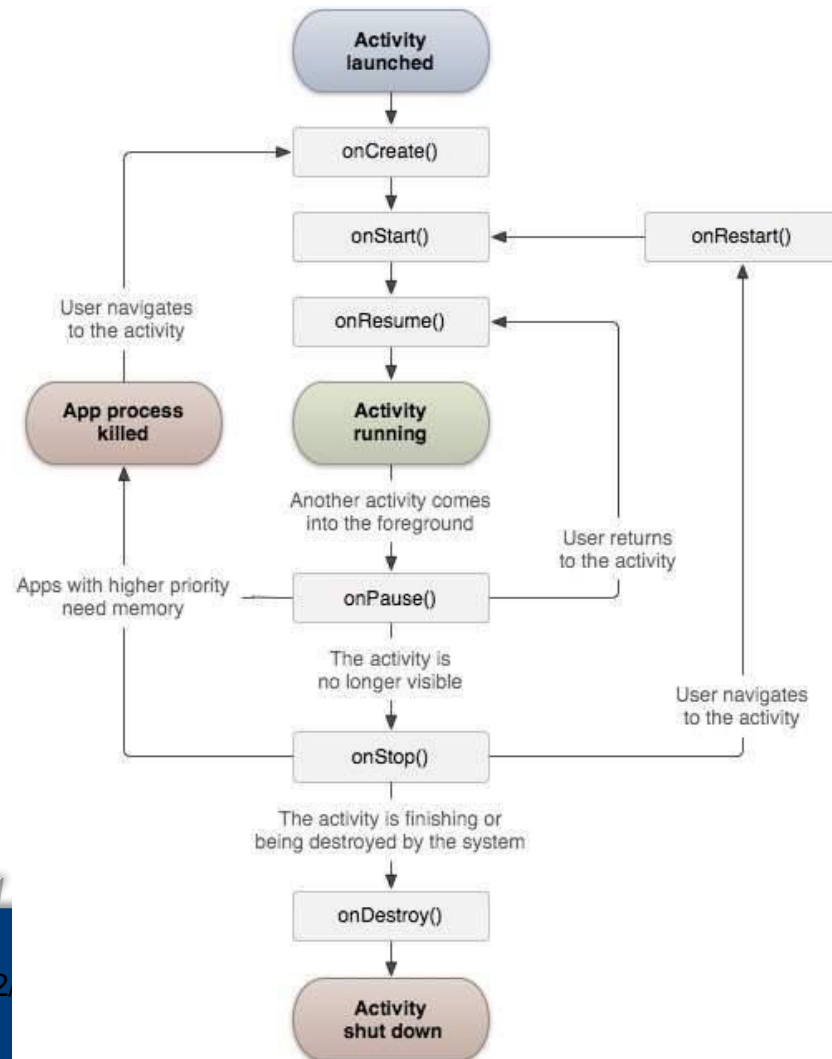
# Quản lý trạng thái Activity

# | Xây dựng Activity

- Một Activity biểu diễn một màn hình với một giao diện UI giống như Window hoặc Frame của Java.Android activity, mà là một lớp con của lớp ContextThemWrapper.
- Hệ điều hành Android khởi tạo chương trình của nó bên trong một Activity bắt đầu với một lời gọi trên phương thức callback là onCreate().



# Vòng đời của Activity



- Khi Activity được kích hoạt, và được hệ thống đẩy vào BackStack. Sau khi kích hoạt, lần lượt các callback onCreate(), onStart(), onResume() sẽ được hệ thống gọi đến

Callback	Miêu tả
onCreate()	Đây là phương thức callback đầu tiên và được gọi khi Activity được tạo đầu tiên
onStart()	Sau khi gọi đến onCreate(), hệ thống sẽ gọi đến onStart(). Hoặc hệ thống cũng sẽ gọi lại onStart() sau khi gọi onRestart() nếu trước đó nó bị che khuất bởi Activity nào khác (một màn hình khác hoặc một ứng dụng khác) che hoàn toàn và rơi vào onStop().
onResume()	Được gọi khi người dùng bắt đầu tương tác với ứng dụng
onPause()	Activity tạm dừng không nhận input từ người dùng và không thể thực thi bất cứ code nào và được gọi khi activity hiện tại đang được dừng và activity trước đó đang được phục hồi

onStop()	Callback này được gọi trước skhi activity bị hủy bởi hệ thống
onDestroy()	Callback này được gọi trước skhi activity bị hủy bởi hệ thống
onRestart()	Được gọi khi activity tái khởi động sau khi dừng nó





# Debug chương trình trong Android Studio

# | Sử dụng Log và Toast

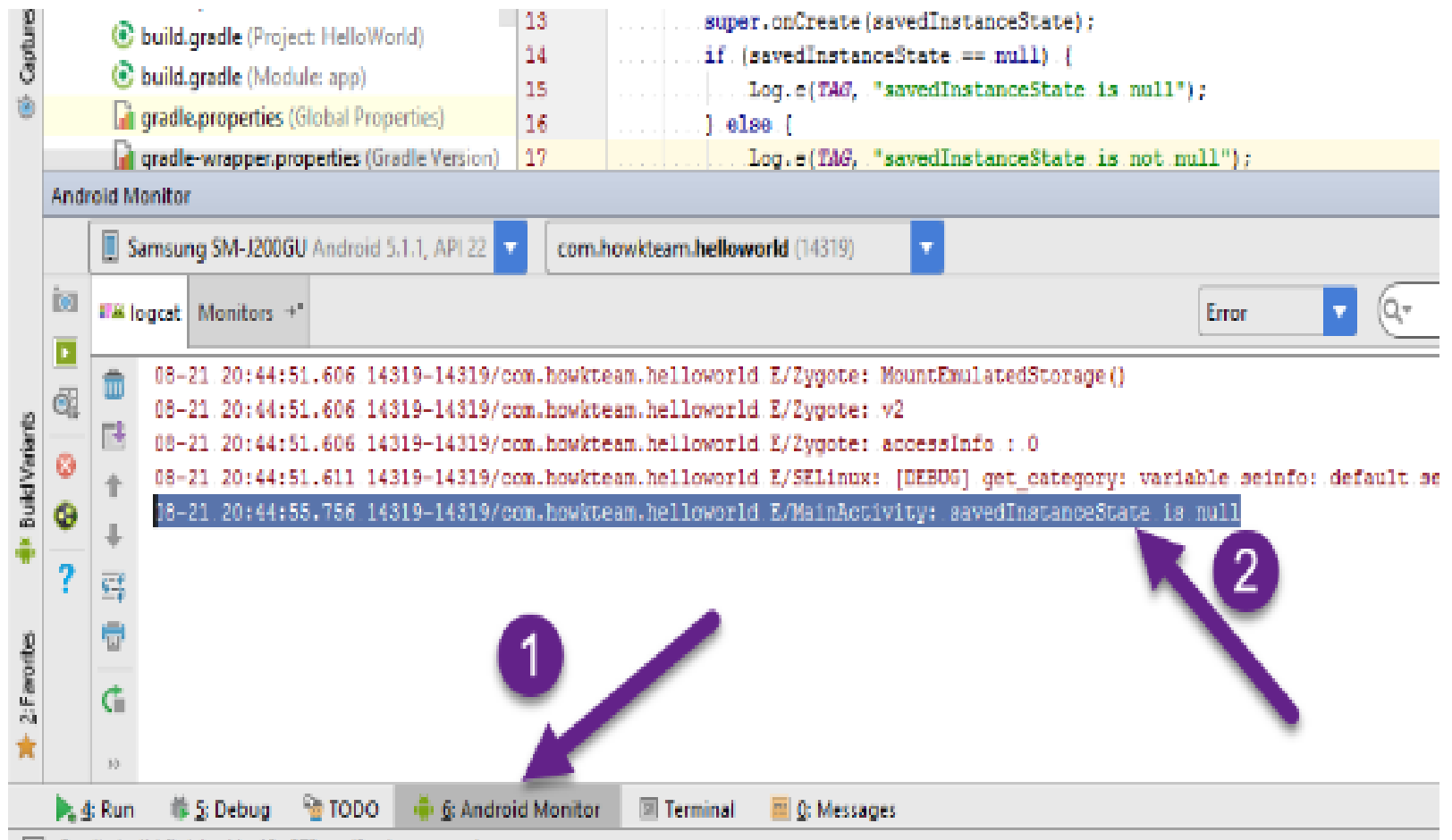
- Khi thực thi ứng dụng muốn kiểm tra xem đã xảy ra lỗi gì hay hệ thống có cảnh báo gì, hoặc chỉ là các thông tin thực thi bình thường chúng ta có thể sử dụng ***LogCat***.

# | Đặt Log trong code:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    Log.d("Test", "Tdc");  
}
```



# Đọc Log



# | Toast

- Toast dùng để hiển thị thông tin trong khoảng thời gian ngắn. nó giống như một thông báo nổi trên ứng dụng, không ngăn cản tương tác người dùng.



**Toast**



**Custom Toast With Image**

# | Các phương quan trọng của Toast

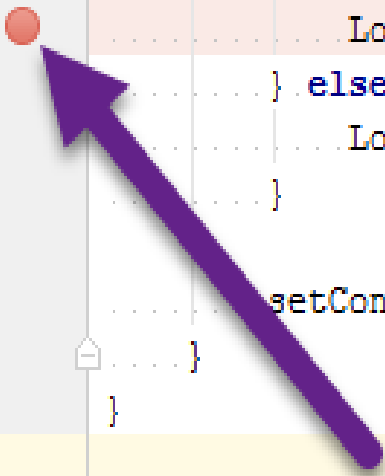
- ***makeText***(Context context, CharSequence text, int duration): Phương thức này thường sử dụng để hiển thị thông báo. Phương thức này có 3 tham số:
  - ***Context context***: thường là YourActivity.this của bạn vào nếu như bạn đang sử dụng ở Activity.
  - ***CharSequence text***: đây chính là nội dung bạn muốn show lên, ở đây là kiểu String
  - ***int duration***: Khoảng thời gian Toast cần hiển thị, nó là hằng số. Hằng số của Toast:

- ***show()***: phương thức này hiển thị thông báo ra màn hình. Phương pháp này hiển thị thông báo bằng cách sử dụng phương thức ***makeText()*** của Toast.

# | Sử dụng Debug

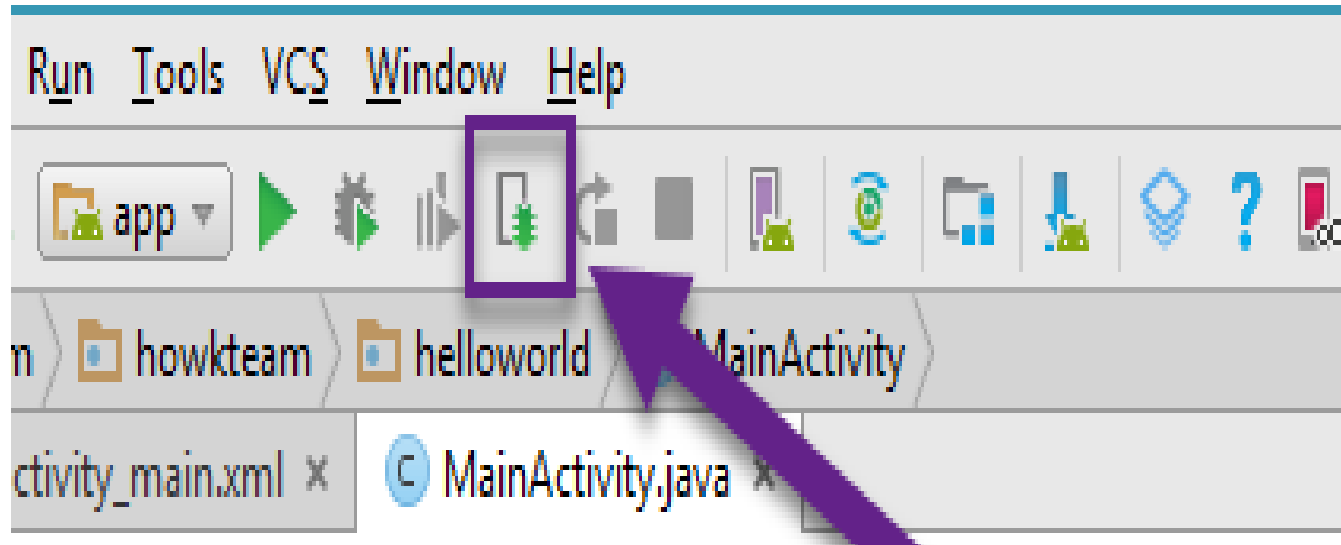
- Android Studio có cung cấp khả năng debug ứng dụng rất hiệu quả cho các ứng dụng chạy trên máy thật lẫn máy ảo
  - Đặt các breakpoint (điểm dừng) trong code.
  - Quan sát và kiểm tra các giá trị biến / biểu thức trong runtime.

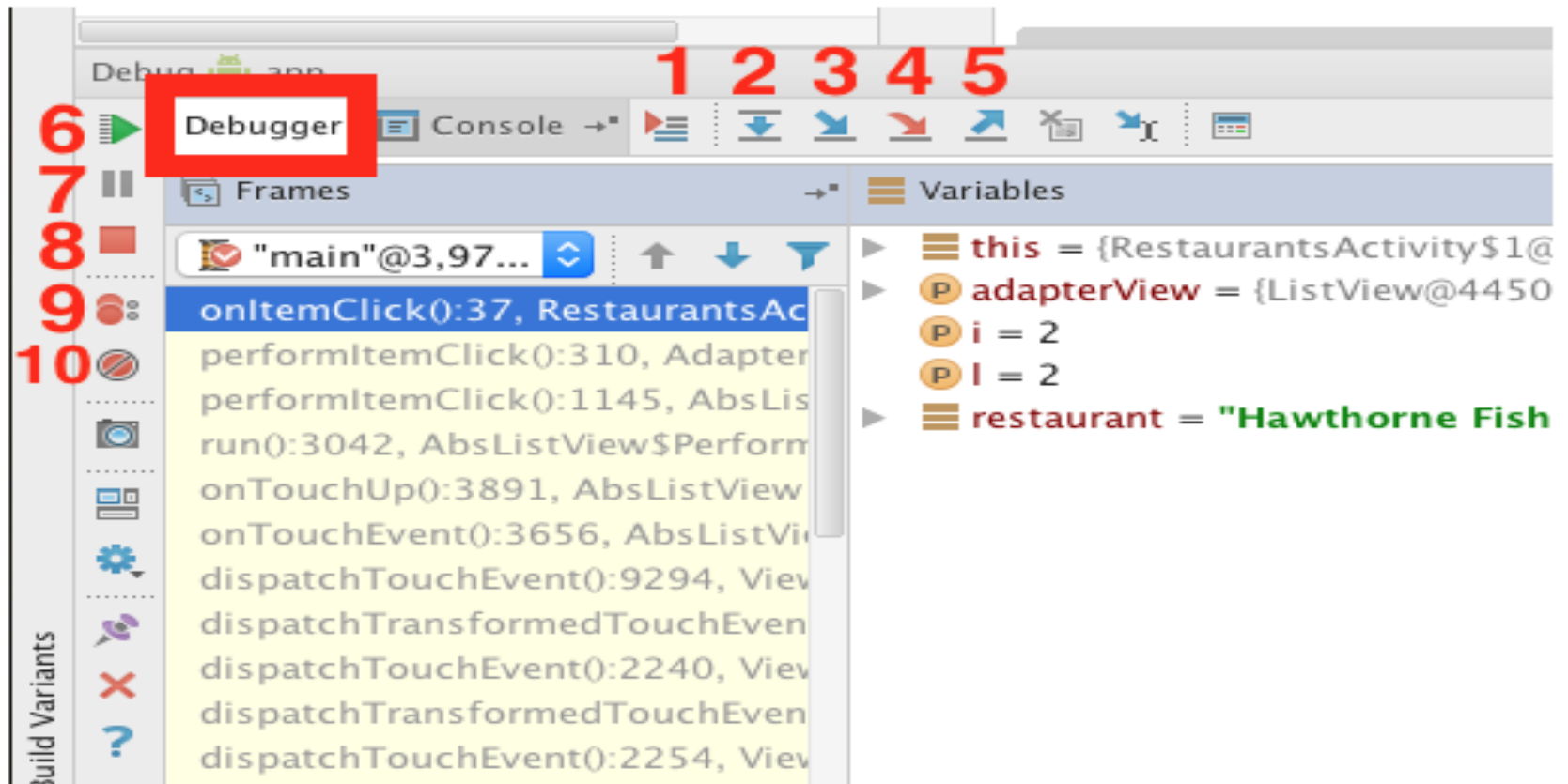
```
7 public class MainActivity extends AppCompatActivity {  
8  
9     ... public static final String TAG = MainActivity.class.getSimpleName()  
10  
11     ... @Override  
12     ... protected void onCreate(Bundle savedInstanceState) {  
13         ... super.onCreate(savedInstanceState);  
14         ... if (savedInstanceState == null) {  
15             ... Log.e(TAG, "savedInstanceState is null");  
16         } else {  
17             ... Log.e(TAG, "savedInstanceState is not null");  
18         }  
19  
20         ... setContentView(R.layout.activity_main);  
21     }  
22 }  
23
```



Click vào đây hoặc Ctrl+F8

# | Để bắt đầu debug,







- Số 1: Nút hiển thị Breakpoint đang active
- Số 2. Step Over, nút này sẽ giúp debug nhảy xuống dòng code tiếp theo
- Số 3. Step Into, nút này sẽ nhảy vào bên trong hàm
- Số 4. Force Step Into, nút này sẽ cho phép nhảy thẳng đến dòng đầu tiên bên trong của hàm được gọi
- Số 5. Thoát ra ngoài

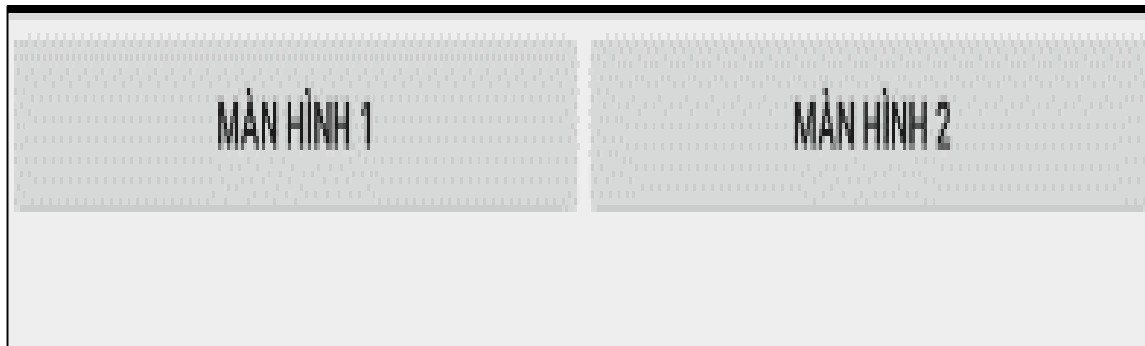
- Số 6. Tiếp tục chương trình(Resume Program), nút này sẽ tiếp tục chạy ứng dụng một cách bình thường. Tạm thời bỏ qua debug
- Số 7. Tạm dừng chương trình(Pause Program)
- Số 8. Dừng ứng dụng (Stop App)
- Số 9. Xem các Breakpoints
- 10. Mute Breakpoint, tắt tạm thời tất cả các breakpoint.



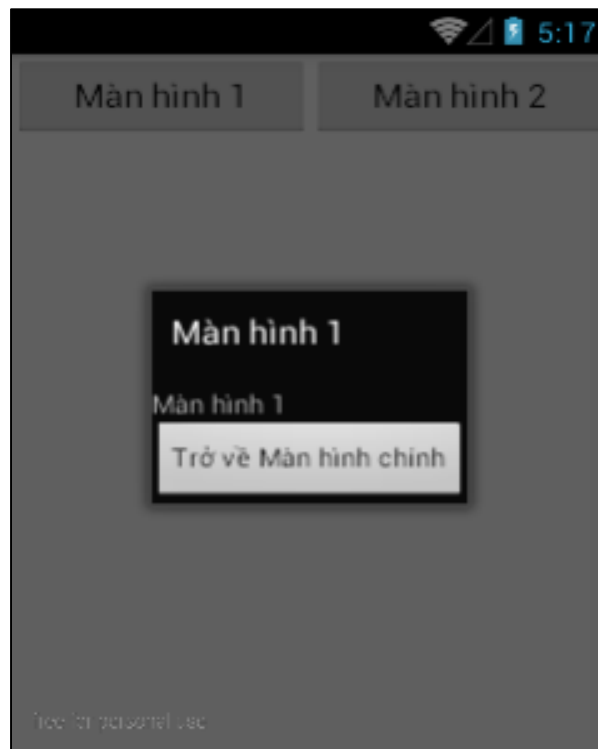
# Bài tập Chương 1

# Bài 1: Phân biệt reground Lifetime và Visible Lifetime

- Tạo 3 Activity đặt tên là: ManHinhChinh, ManHinh1, ManHinh2.
- Giao diện **màn hình chính** như sau:



- Khi ManHinh1 được kích hoạt thì nó sẽ nằm phía trên ManHinhChinh vẫn thấy được màn hình chính



- Khi ManHinh2 hiển thị thì nó sẽ chiếm toàn bộ màn hình không thể thấy được màn hình chính.
- Bắt các sự kiện khi Activity chuyển đổi trạng thái và thông báo lên màn hình Android.

# Bài 2: Sử dụng ConstraintLayout xây dựng ứng dụng

a = Nhập vào số a...

b = Nhập vào số b...

+ - \* /

Result: Kết Quả



**CẢM ƠN TẤT CẢ  
ĐÃ LẮNG NGHE**