

## Chương 2:

# ANDROID SKELETON AND ACTIVITY



**TS. Huỳnh Hữu Nghĩa**

luckerhuynhv@gmail.com

# Nội dung:

---

- Giới thiệu các thành phần của ứng dụng
- Vòng đời của ứng dụng Android
- Sử dụng các tài nguyên bên trong và bên ngoài
- Tạo mới Activities, kết nối các Activity dùng Intent
- Giới thiệu Fragments, làm việc với Notifications
- Trạng thái chuyển tiếp activity và lifecycle.



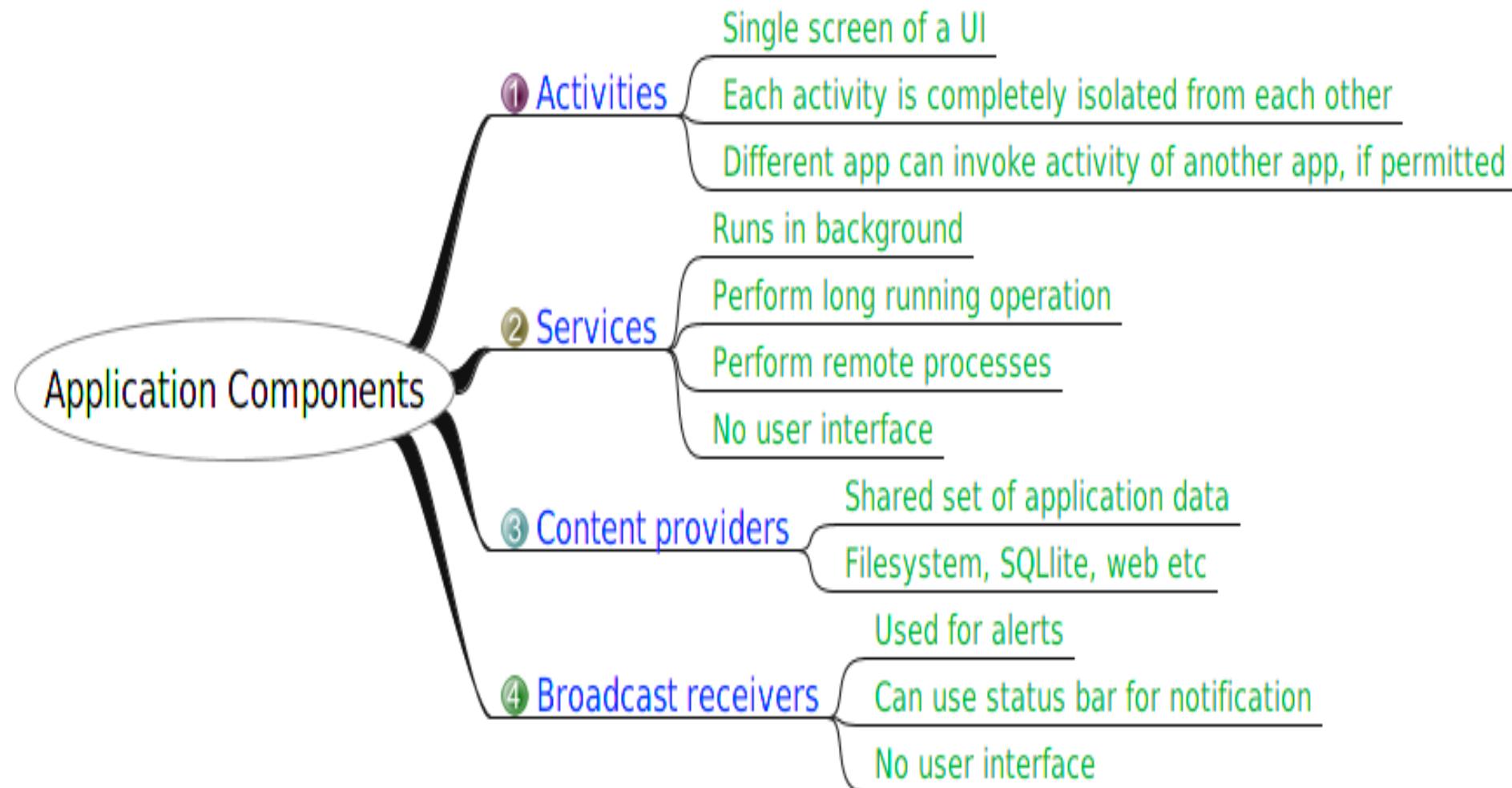
# Ứng dụng Android

## ❖ Ứng dụng (Application):

- Mỗi Android Project khi biên dịch thành công sẽ được đóng gói thành tập tin **.apk**, tập tin **.apk** gọi là một **Ứng dụng**.
- Một **Ứng dụng** có một hoặc nhiều **Activities** (*trong đó có một Activity chính gọi là MainActivity*)
- Mỗi một **Activity** sẽ có vòng đời riêng độc lập với những Activity khác.
- Một **Activity** muốn triệu gọi thì bắt buộc phải được khai báo trong **Manifest**.



# Các thành phần ứng dụng



# Các thành phần ứng dụng

## ❖ Android Activity:

- Một Activity có thể là một mô đun, thành phần chức năng độc lập của ứng dụng mà mỗi Activity thường tương ứng với một giao diện người dùng (UI) và các chức năng tương tác với người dùng.
- Các Activity có thể tái sử dụng, tương tác và chia sẻ giữa các ứng dụng.



# Các thành phần ứng dụng

## ❖ Android Activity:

- Một Activity được xây dựng dựa trên sự kế thừa lớp Activity, một Activity không thể gọi trực tiếp các phương thức hay truy cập dữ liệu của một Activity khác.
- Để các Activity có thể giao tiếp với nhau cần sử dụng thành phần trung gian Intent và Content Provider.

# Các thành phần ứng dụng

❖ **Android Services:** Là một thành phần chạy nền bên dưới để thực hiện các hoạt động chạy lâu dài mà không cần tương tác với người dùng và nó vẫn hoạt động ngay cả ứng dụng bị hủy.

Một service cơ bản có hai trạng thái:

- **Started.**
- **Bound.**

# Các thành phần ứng dụng

## ❖ Android Services:

- **Started:** Một service được **started** khi một thành phần ứng dụng (như 1 activity) khởi động nó bằng cách gọi `startService()`. Sau khi bắt đầu, một service có thể chạy vô hạn ở nền bên dưới, ngay cả khi thành phần khởi động bị hủy.

# Các thành phần ứng dụng

## ❖ Android Services:

- **Bound:** Một service được **bound** khi một thành phần ứng dụng liên kết với nó bằng cách gọi bindService(). Một dịch vụ bound cung cấp một giao diện client-server cho phép các thành phần tương tác với service, send requests, get results, thậm chí thực hiện xuyên các tiến trình với giao tiếp giữa các tiến trình.

# Các thành phần ứng dụng

## ❖ Android Content Provider:

- Thành phần content provider cung cấp dữ liệu từ một ứng dụng này đến ứng dụng khác theo yêu cầu. Những yêu cầu như vậy được xử lý bởi các phương thức của lớp ContentResolver.
- Một content provider có thể sử dụng nhiều cách khác nhau để lưu trữ dữ liệu và dữ liệu có thể được lưu trữ trong database, files, hoặc qua mạng.

# Các thành phần ứng dụng

## ❖ Android Content Provider:

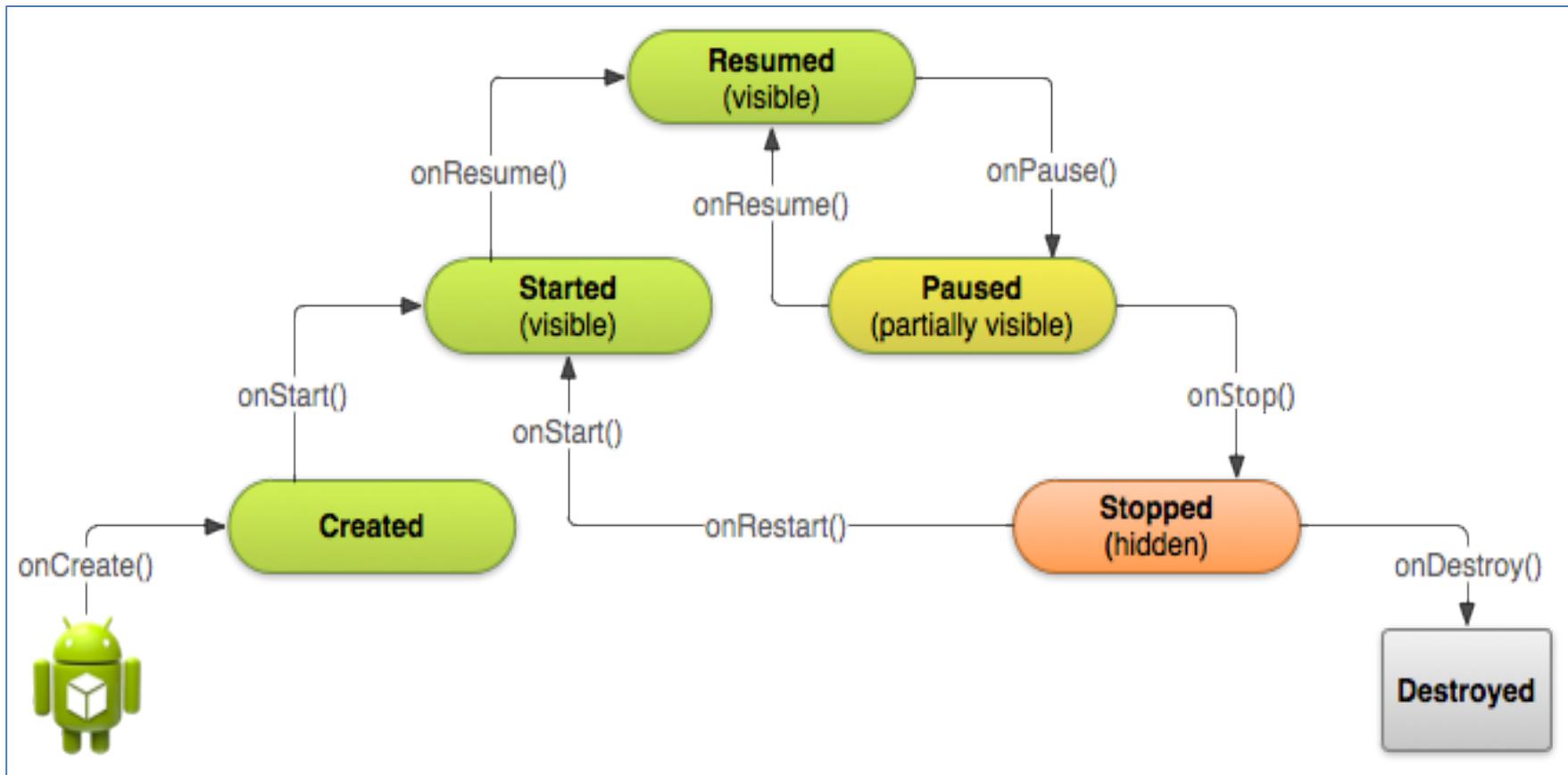


# Các thành phần ứng dụng

## ❖ Android Broadcast Receivers:

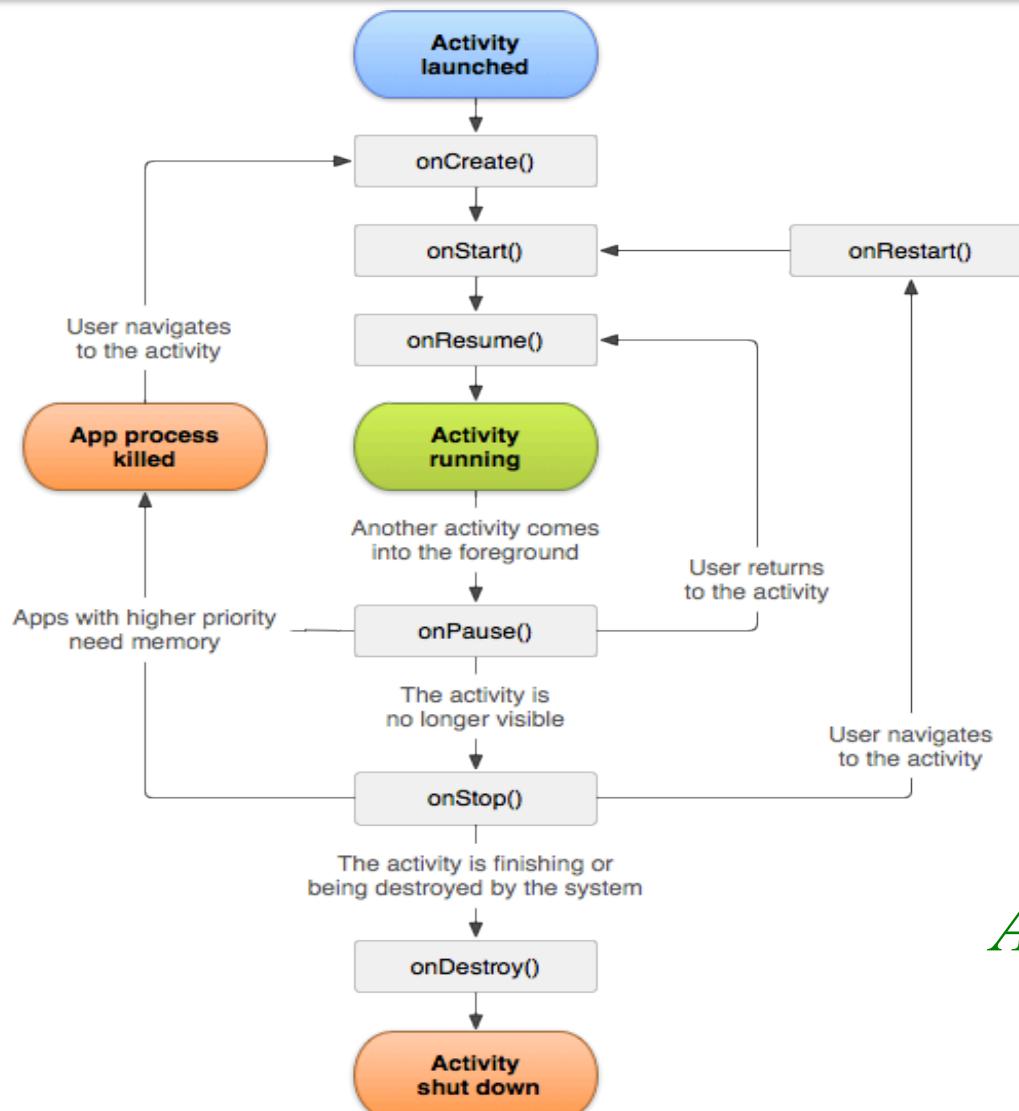
- Broadcast receivers chỉ đơn giản là trả lời các tin nhắn truyền thông từ các ứng dụng khác nhau hoặc từ chính hệ thống. Các tin nhắn này đôi khi được gọi là events hay intents.

# Vòng đời ứng dụng Android



*Android lifecycle*

# Vòng đời ứng dụng Android

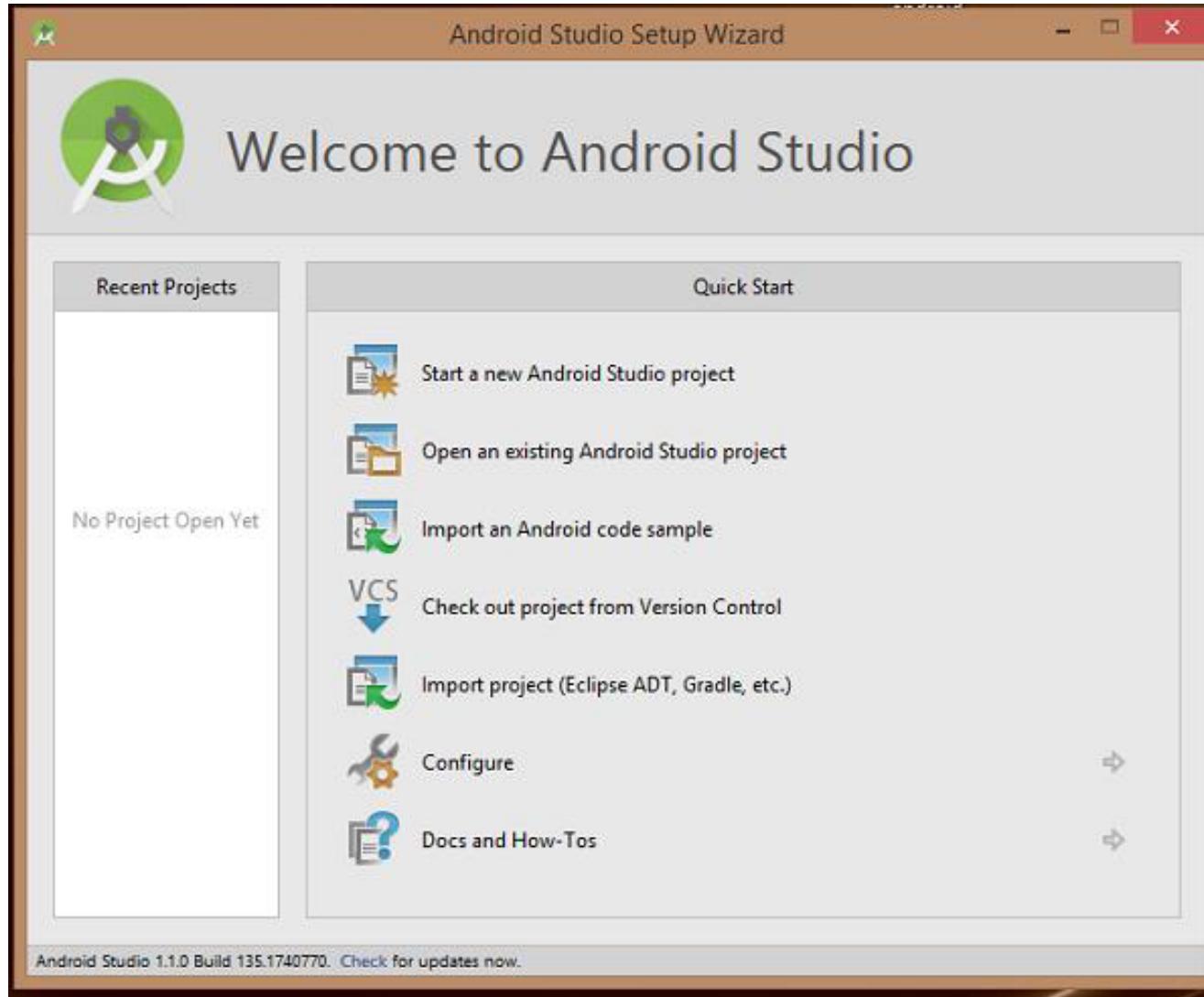


*Activity lifecycle*

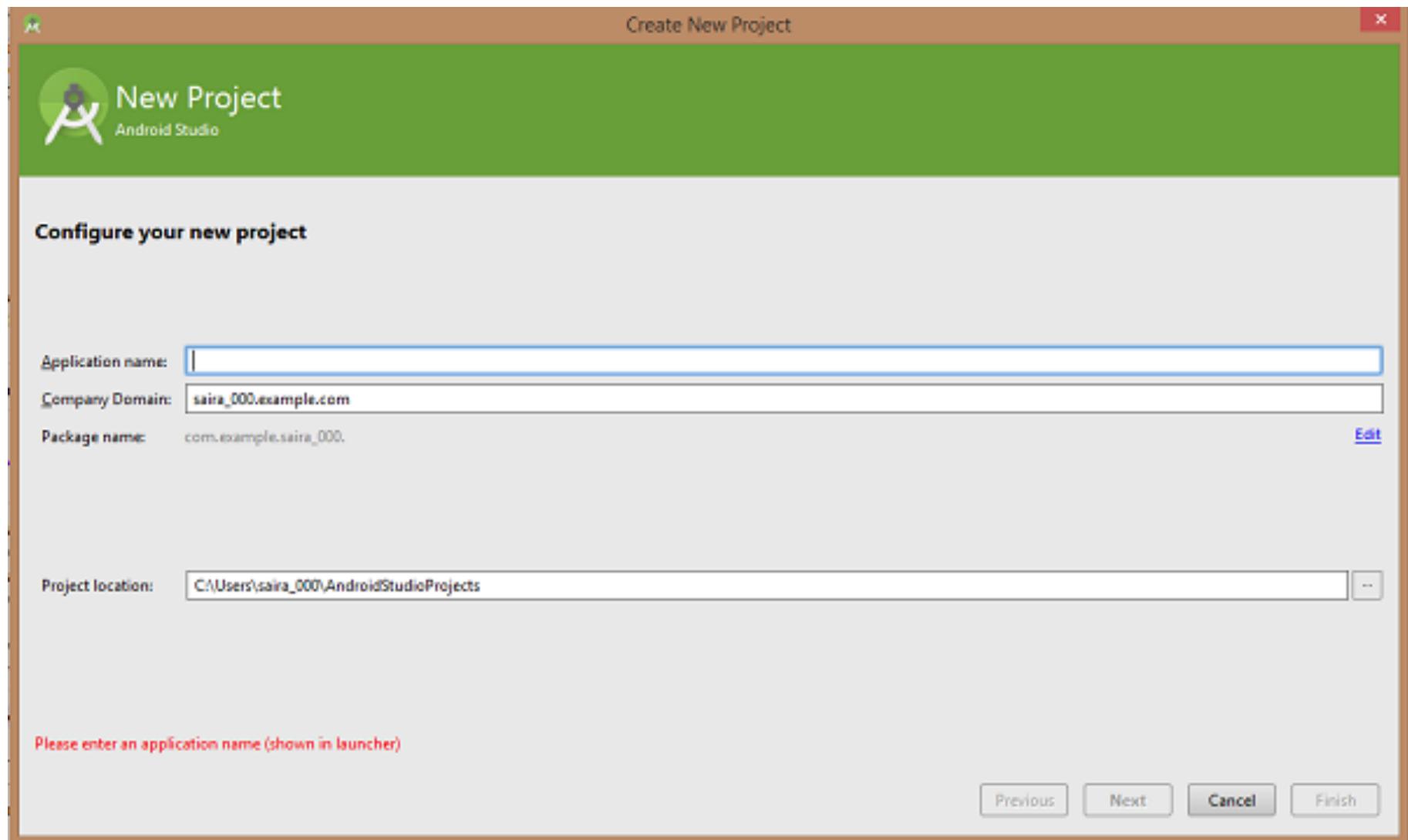
# Hello World Example



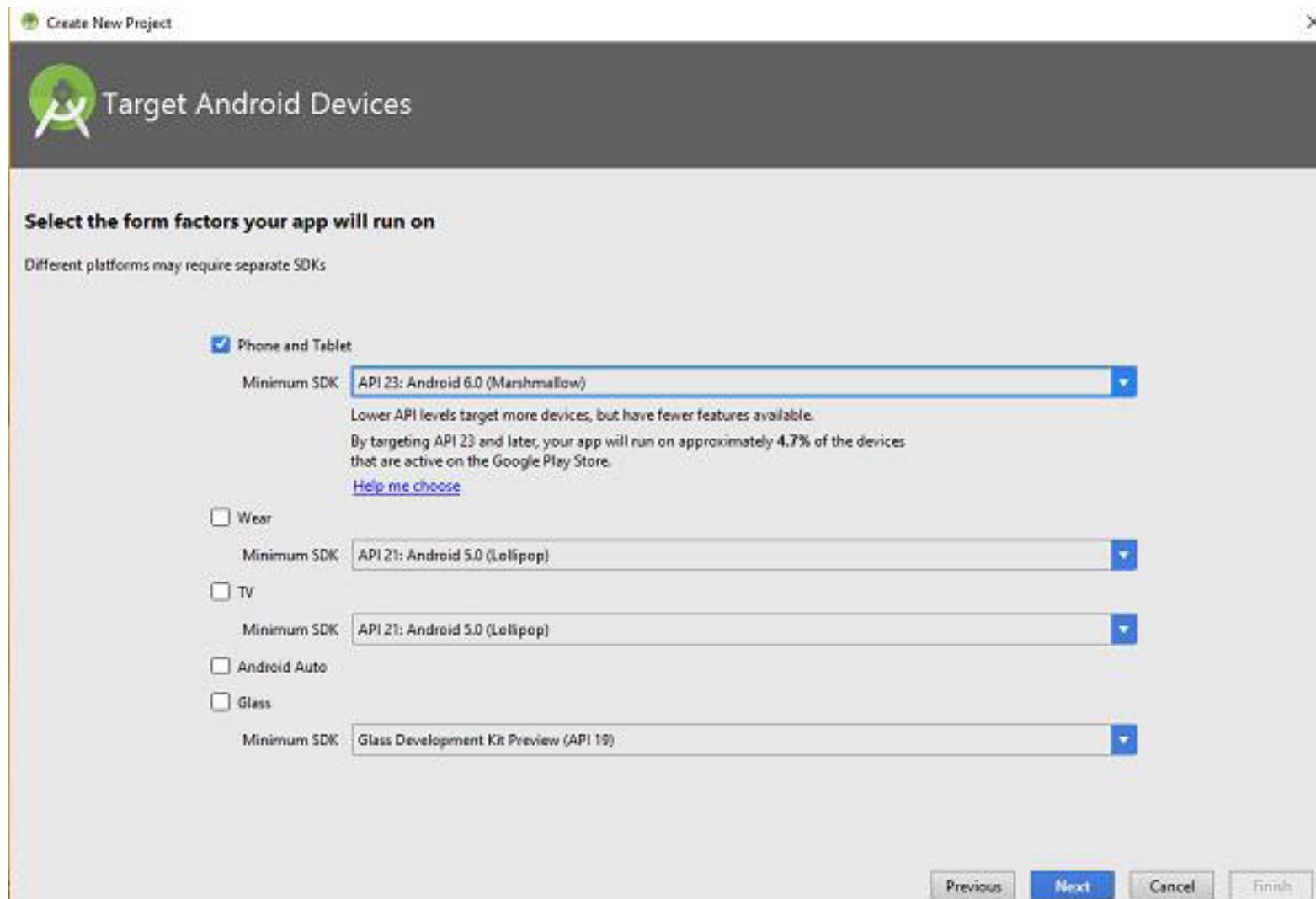
# Create Android Application



# Create Android Application



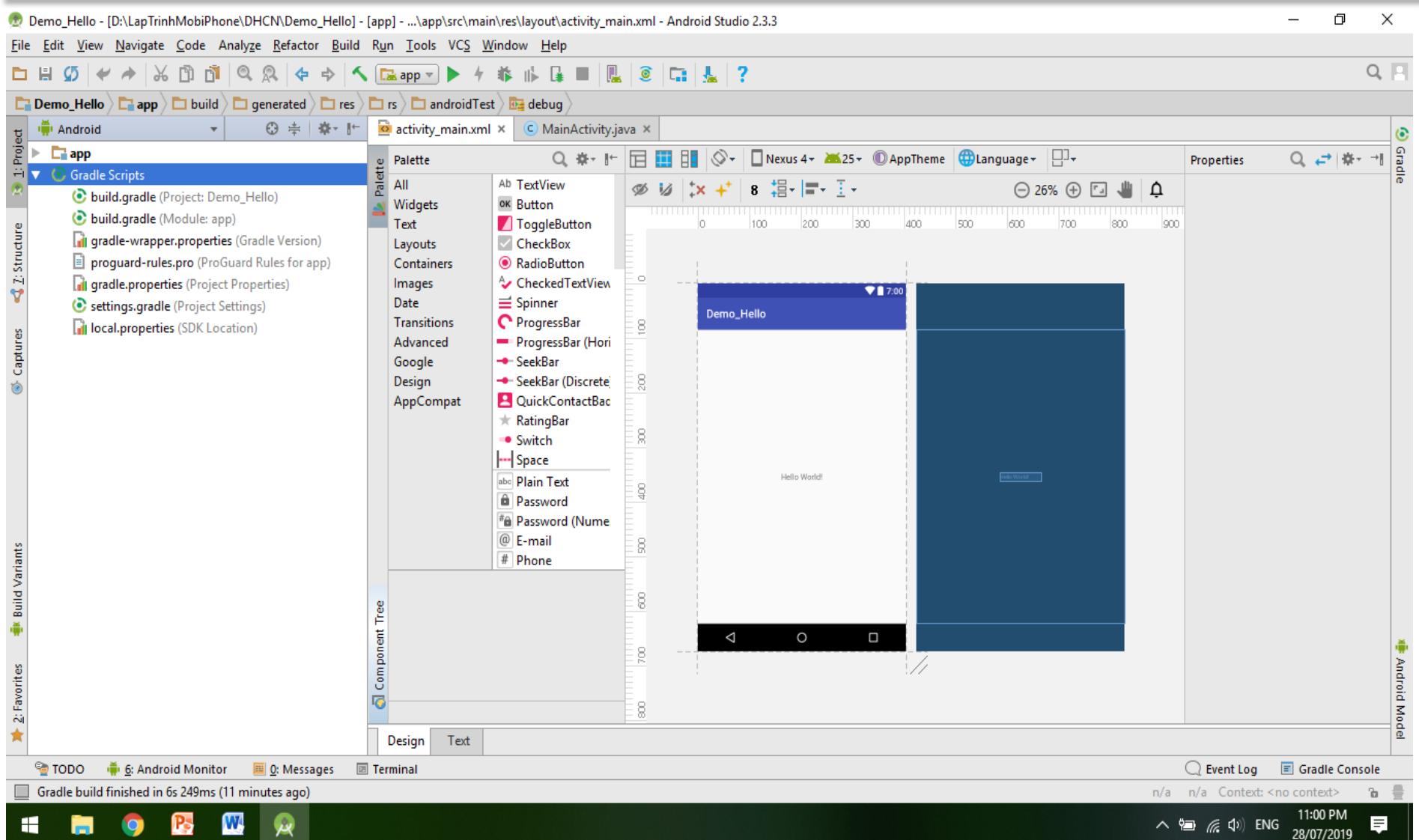
# Create Android Application



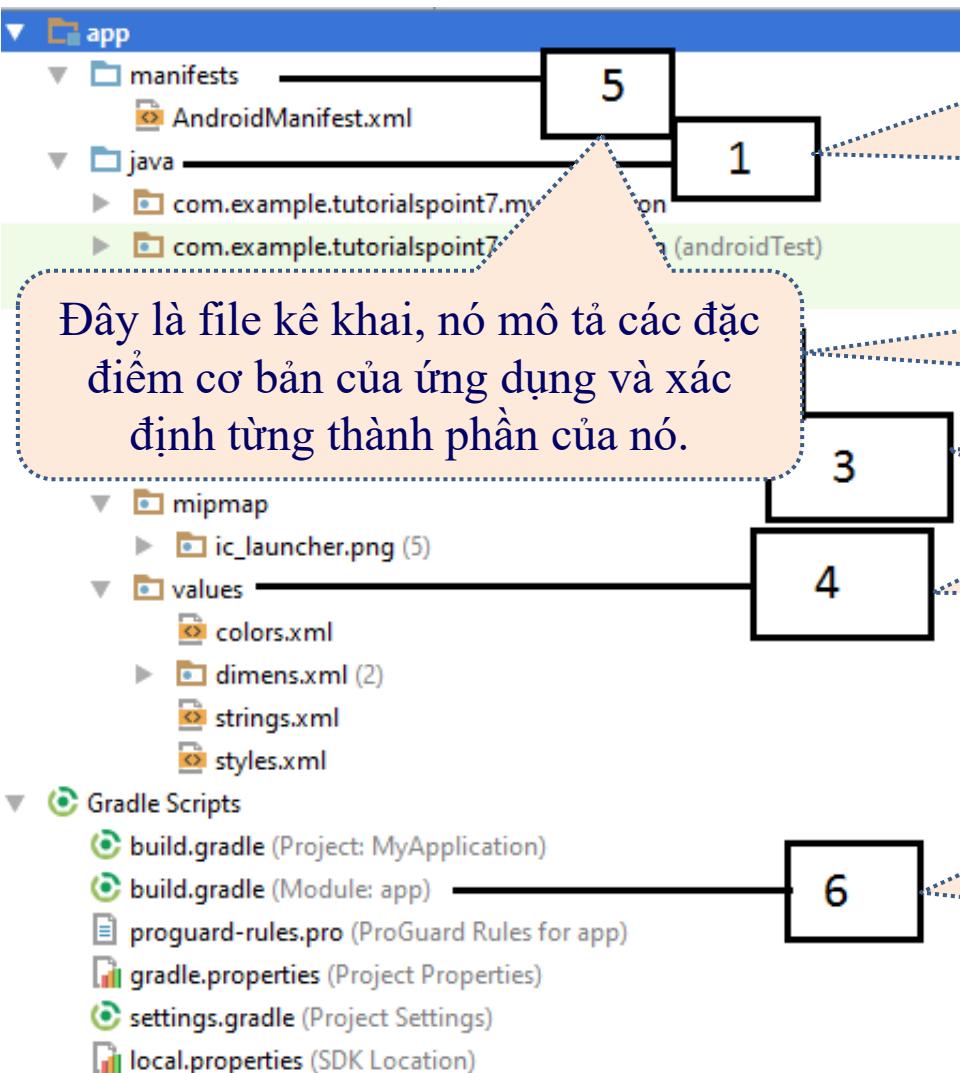
# Create Android Application



# Create Android Application



# Create Android Application



Đây là file khai báo, nó mô tả các đặc điểm cơ bản của ứng dụng và xác định từng thành phần của nó.

Phần này chứa các file nguồn cho dự án. Theo mặc định, nó bao gồm một file nguồn **MainActivity.java** có một

Đây là thư mục cho các đối tượng có thể vẽ ra màn hình nó được thiết kế

Đây là thư mục cho các file xác định giao diện người dùng của ứng dụng.

Đây là một thư mục cho các file XML khác nhau chứa tập nguồn tài nguyên, nhưng định nghĩa các chuỗi và các màu.

Đây là một file được sinh ra tự động, nó chứa compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode và versionName

# The Main Activity File

MainActivity.java

```
package com.example.helloworld;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

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



# The Manifest File

*manifest.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tutorialspoint7.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        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>
```



# The Strings File

strings.xml

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```

# The Layout File

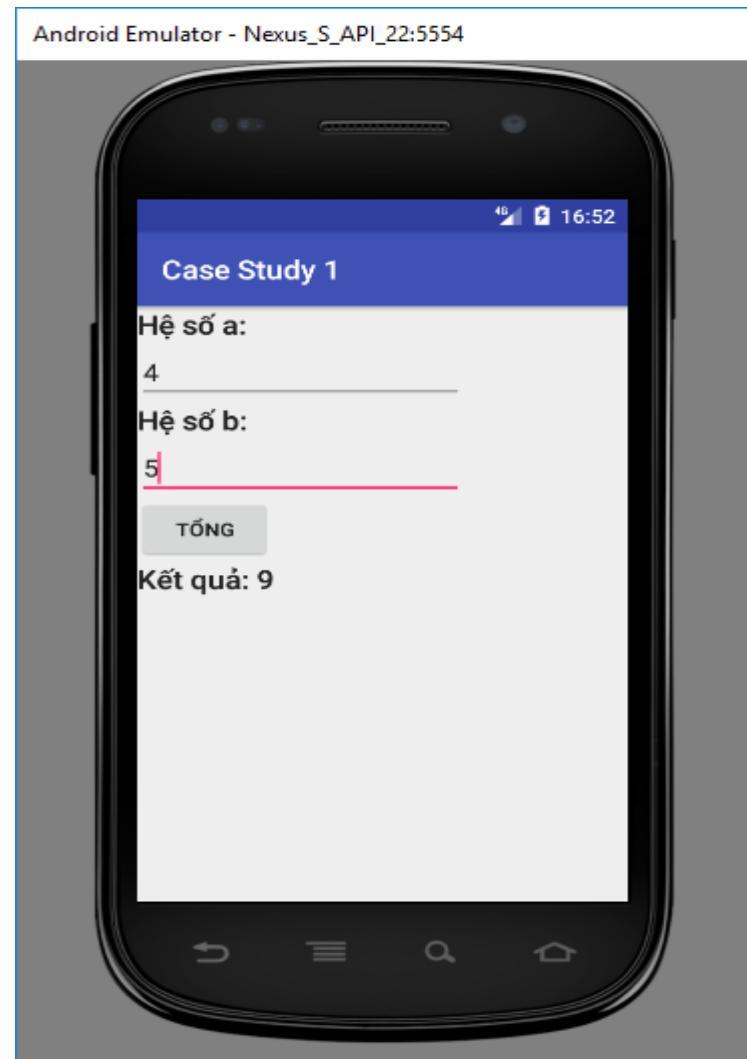
activity\_main.xml

```
<RelativeLayout 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">  
  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_centerHorizontal="true"  
        android:layout_centerVertical="true"  
        android:padding="@dimen/padding_medium"  
        android:text="@string/hello_world"  
        tools:context=".MainActivity" />  
  
</RelativeLayout>
```



# Case study 1: Tạo Activity

- ❖ Nhập 2 số a và b. Xuất ra tổng, hiệu, tích và thương.



# Kết nối các Activity - Intents



# Kết nối các Activity - Intents

## ❖ **Intents:**

- Được sử dụng để điều hướng từ một Activity này sang Activity khác.
- Có 2 kiểu:



# Kết nối các Activity - Intents

## ❖ Explicit Intents:

```
// Explicit Intent by specifying its class name  
Intent i = new Intent(FirstActivity.this, SecondActivity.class);  
// Starts TargetActivity  
startActivity(i);
```



# Kết nối các Activity - Intents

## ❖ Implicit Intents:

```
Intent sendIntent = new Intent(Intent.ACTION_SEND);
```

```
...
```

```
// Always use string resources for UI text.
```

```
// This says something like "Share this photo with"
```

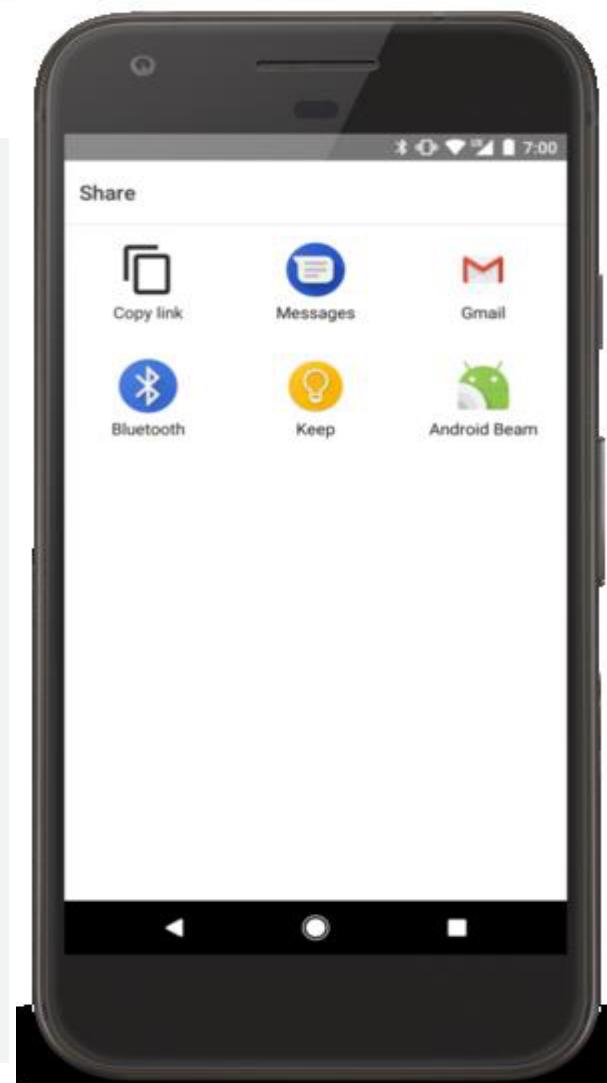
```
String title = getResources().getString(R.string.chooser_title);
```

```
// Create intent to show the chooser dialog
```

```
Intent chooser = Intent.createChooser(sendIntent, title);
```

```
// Verify the original intent will resolve to at least one activity
```

```
if (sendIntent.resolveActivity(getApplicationContext()) != null) {  
    startActivity(chooser);  
}
```



# Truyền dữ liệu giữa các Activity

Truyền dữ liệu từ Activity1 sang Activity2

## ❖ Activity1:

```
intent.putExtra("giatri1", new String("Hello"));
```

```
intent.putExtra("giatri2", new Integer(16));
```

## ❖ Activity2:

```
String gt1 = getIntent().getExtras().getString("giatri1");
```

```
int gt2 = getIntent().getExtras().getString("giatri2");
```

# Truyền dữ liệu giữa các Activity

Activity2 trả dữ liệu về cho Activity1

## ❖ Activity1:

```
Intent it = new Intent(arg0.getContext(),activity2.class);  
startActivityForResult(it,999);
```

## ❖ Activity2:

```
String age = et.getText().toString();  
Intent trave = new Intent();  
trave.putExtra("age", age);  
setResult(RESULT_OK, trave);
```



# Truyền dữ liệu giữa các Activity

Activity2 trả dữ liệu về cho Activity1

## ❖ Activity1:

```
// Override hàm onActivityResult  
protected void onActivityResult(int requestCode, int resultCode, Intent data){  
    super.onActivityResult(requestCode, resultCode, data);  
    if(requestCode == 999 && resultCode == RESULT_OK){  
        String result = data.getStringExtra("age").toString();  
        tv.setText("Your age is " + result);  
    }  
}
```



# Case study 2:

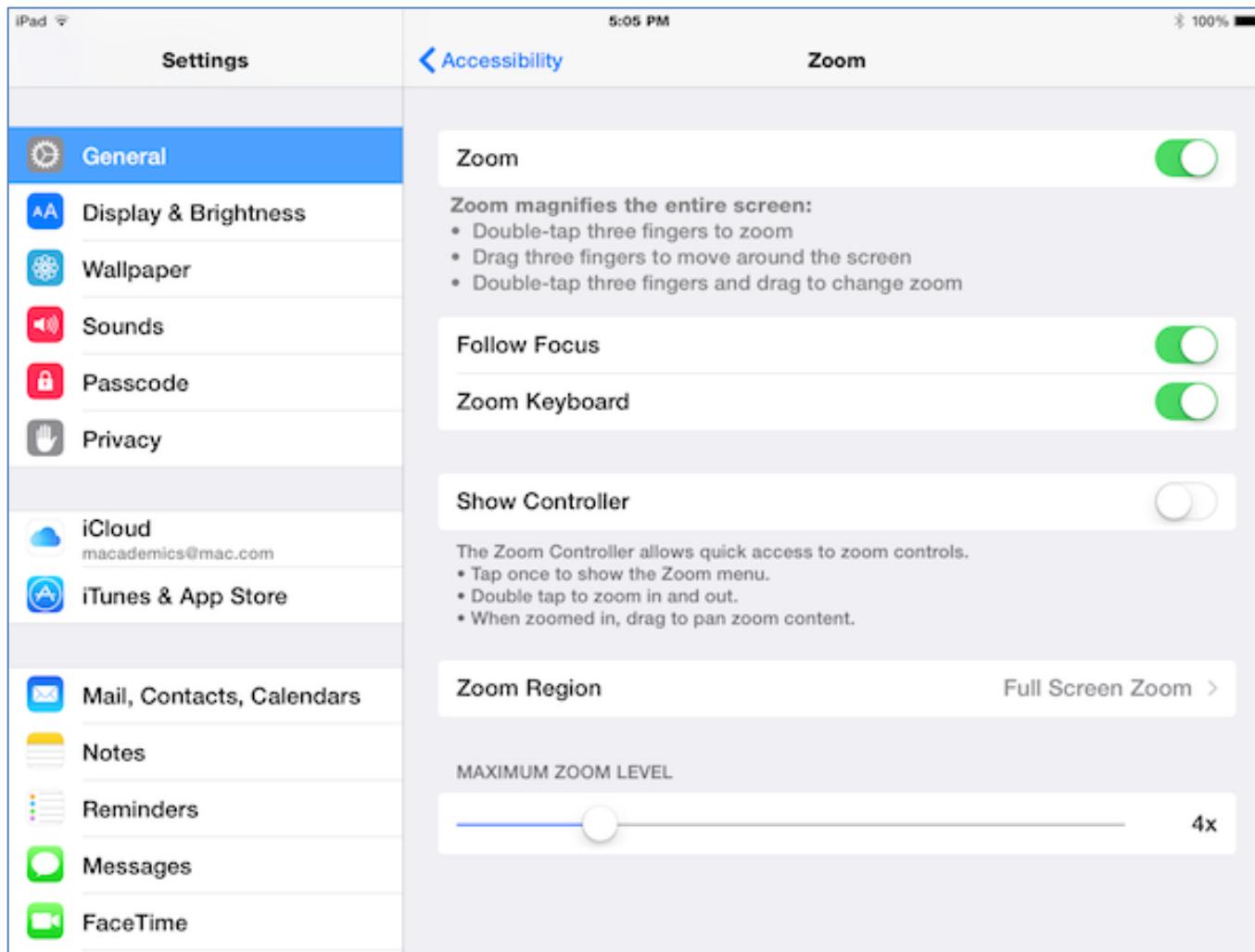
Xây dựng chương trình có 2 Activity. Activity1 cho phép nhập vào họ tên và năm sinh. Khi bấm nút “Submit” sẽ truyền dữ liệu qua Activity2. Activity2 sẽ nhận dữ liệu từ Activity1 và hiển thị: họ tên, năm sinh và tuổi.



# Case study 3:

Xây dựng chương trình có 2 Activity. Activity1 có nút button “Nhập thông tin” sẽ gọi Activity2 và nhận kết quả nhập trả về từ Activity2. Activity2 cho phép nhập vào họ tên và năm sinh, kết quả nhập trả về cho Activity1. Activity1 sẽ hiển thị: họ tên, năm sinh và tuổi.

# Fragments?



# Fragments?

- Để theo kịp Apple cũng như sức từ các “Nhà sản xuất thiết bị”, Google đã xây dựng một phiên bản Android đặc thù cho tablet, thư viện tập trung vào phát triển Fragments.
- **Fragment** là một thành phần dùng để hỗ trợ thiết kế giao diện người dùng nhằm mục tiêu tối ưu hóa không gian màn hình thiết bị.

# Fragments?



# Fragments?

- Để theo kịp Apple cũng như sức ép từ các “Nhà sản xuất thiết bị”, Google đã xây dựng một phiên bản Android đặc thù cho tablet, thư viện tập trung vào phát triển Fragments.
- Fragment đã phát triển rất nhanh từ Android 3.0 và thư viện support cho Fragment cũng đã xuất hiện và hỗ trợ ngược lại cho tới Android 1.6

# Fragments?

- Fragment là một thành phần Android độc lập, được sử dụng bởi một Activity, giống như một sub-activity.
- Fragment có vòng đời và giao diện riêng.
- Các Fragment thường có một file java đi kèm với file giao diện XML. Fragment không có file giao diện XML được gọi là headless Fragment.
- Một Fragment có thể được sử dụng trong nhiều Activity.

# Fragments?

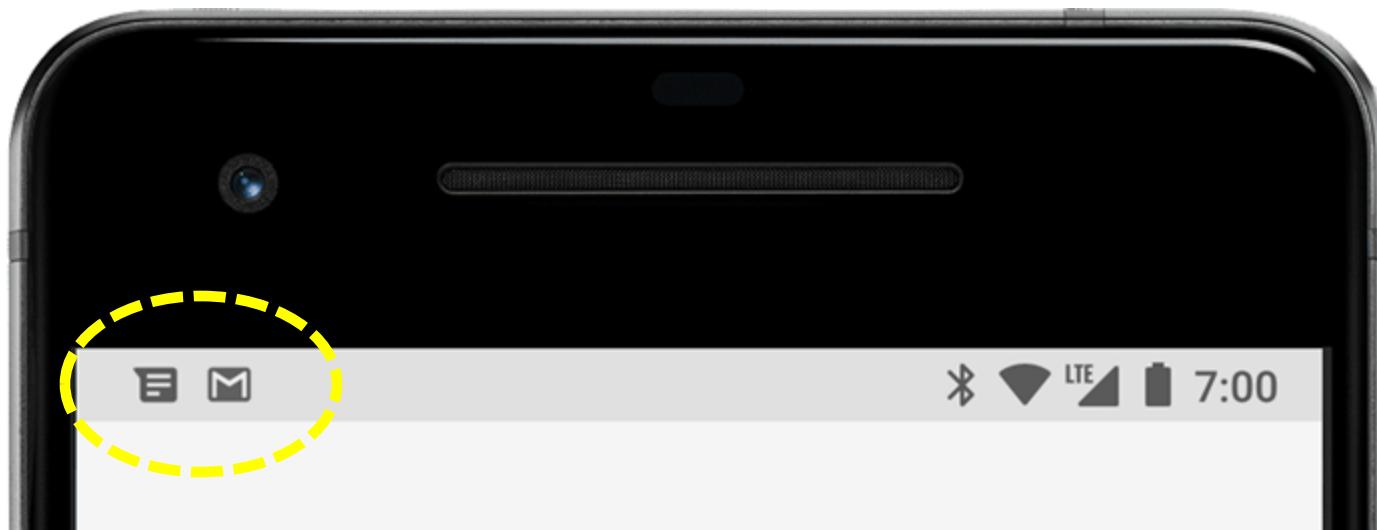
- Vòng đời của Fragment bị ảnh hưởng bởi vòng đời Activity chủ. Khi Activity bị tạm dừng hay bị hủy thì tất cả Fragment trong nó cũng bị dừng hay bị hủy theo.
- Fragment được định nghĩa trong file xml của Activity (static definition) hoặc có thể sửa đổi Fragment khi đang chạy (dynamic definition).

# Notifications

- Notification là một tin nhắn mà Android hiển thị bên ngoài UI của ứng dụng nhằm cung cấp người với những cái nhắc nhở, liên lạc từ người khác, hoặc thông tin kịp thời khác từ ứng dụng.
- Người dùng có thể bấm notification để mở ứng dụng hay thực hiện một hành động trực tiếp từ notification.

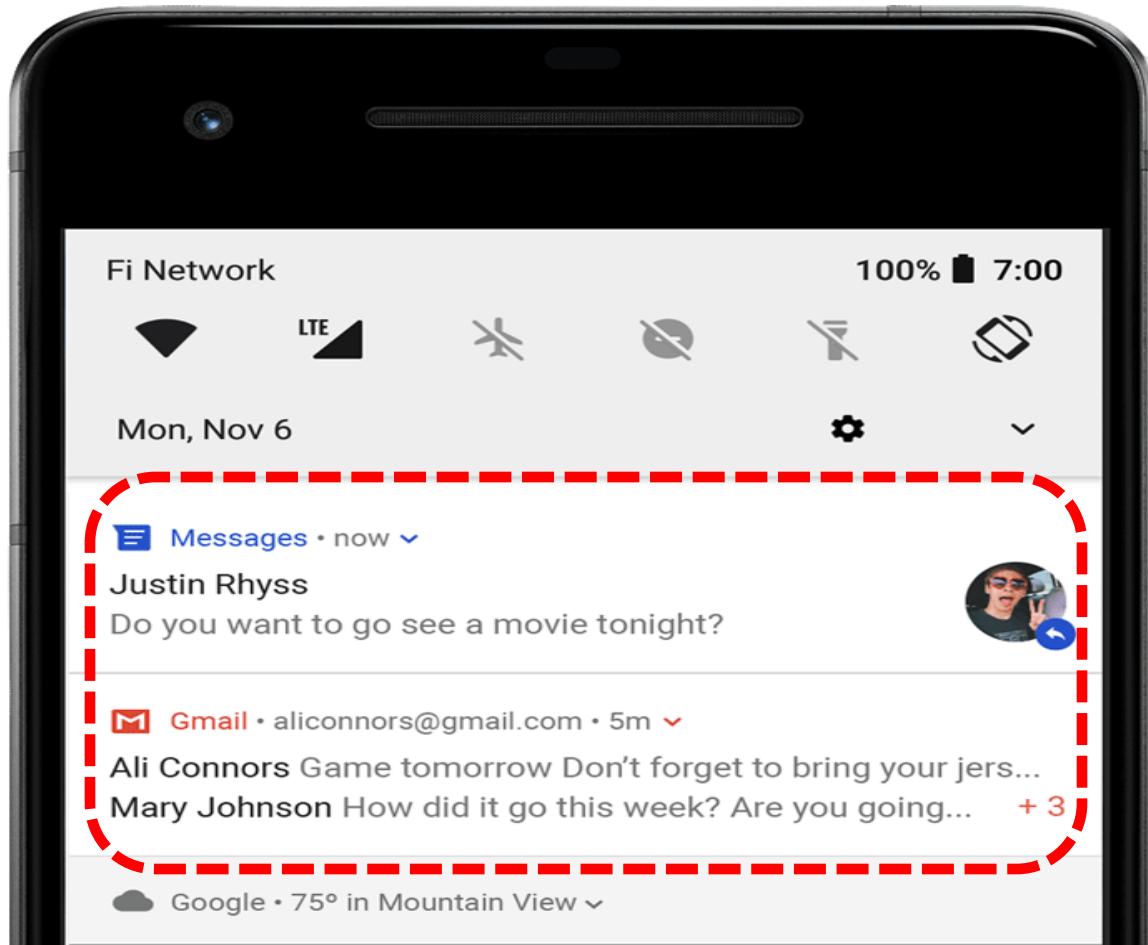
# Notifications xuất hiện trên thiết bị

- Status bar and notification drawer



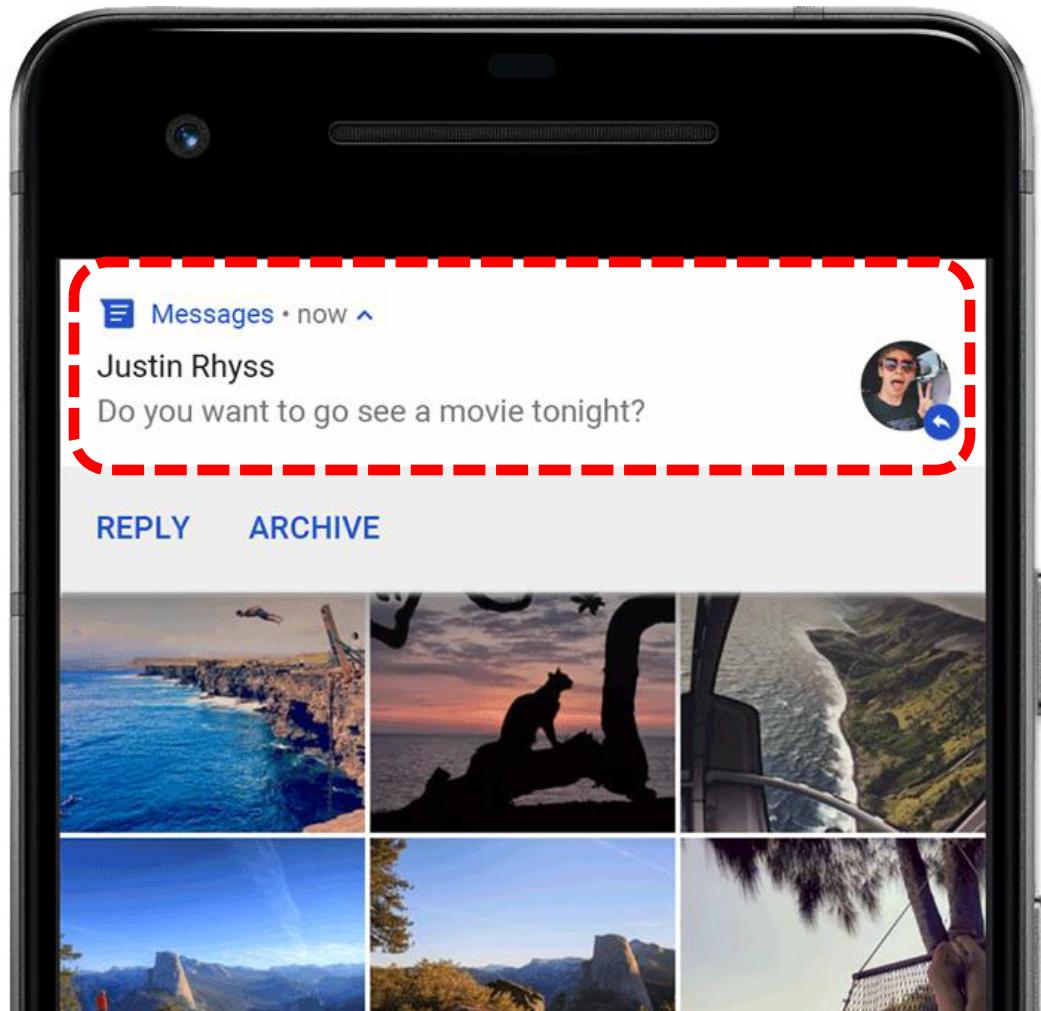
# Notifications xuất hiện trên thiết bị

- Status bar and notification drawer



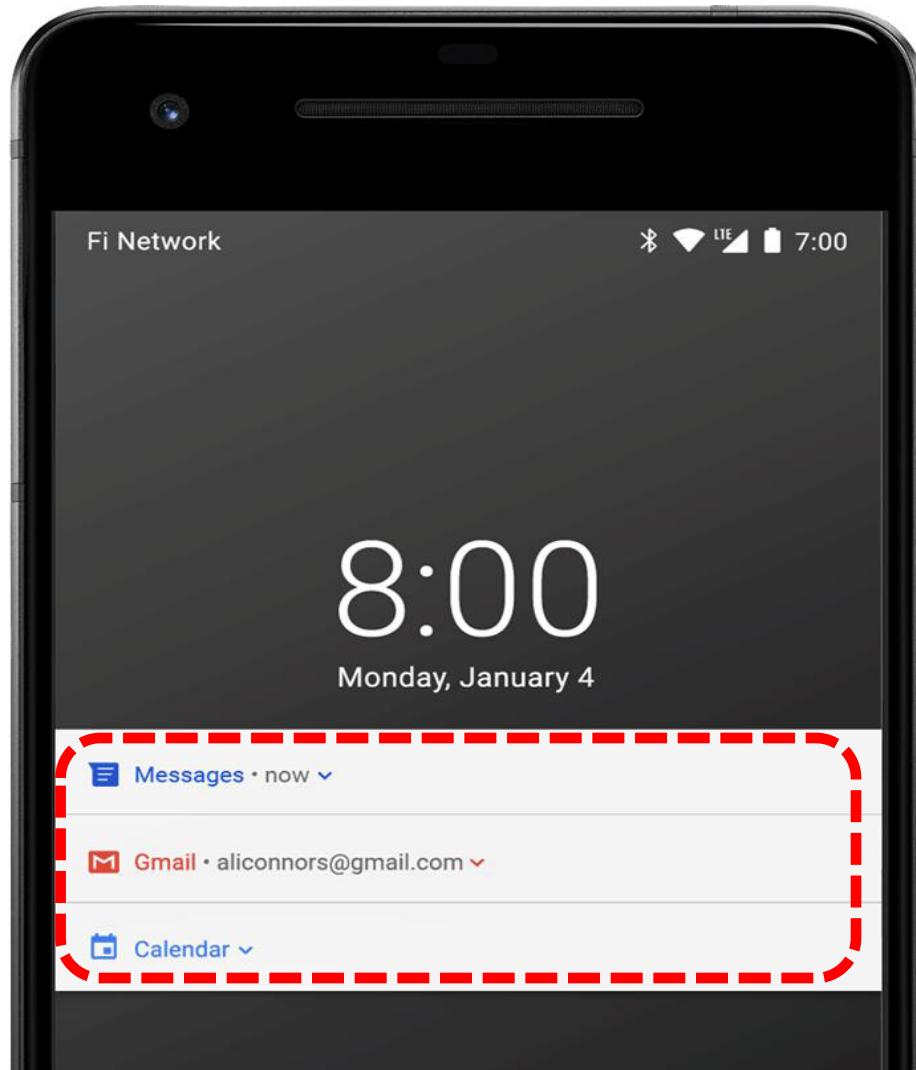
# Notifications xuất hiện trên thiết bị

- Heads-up notification



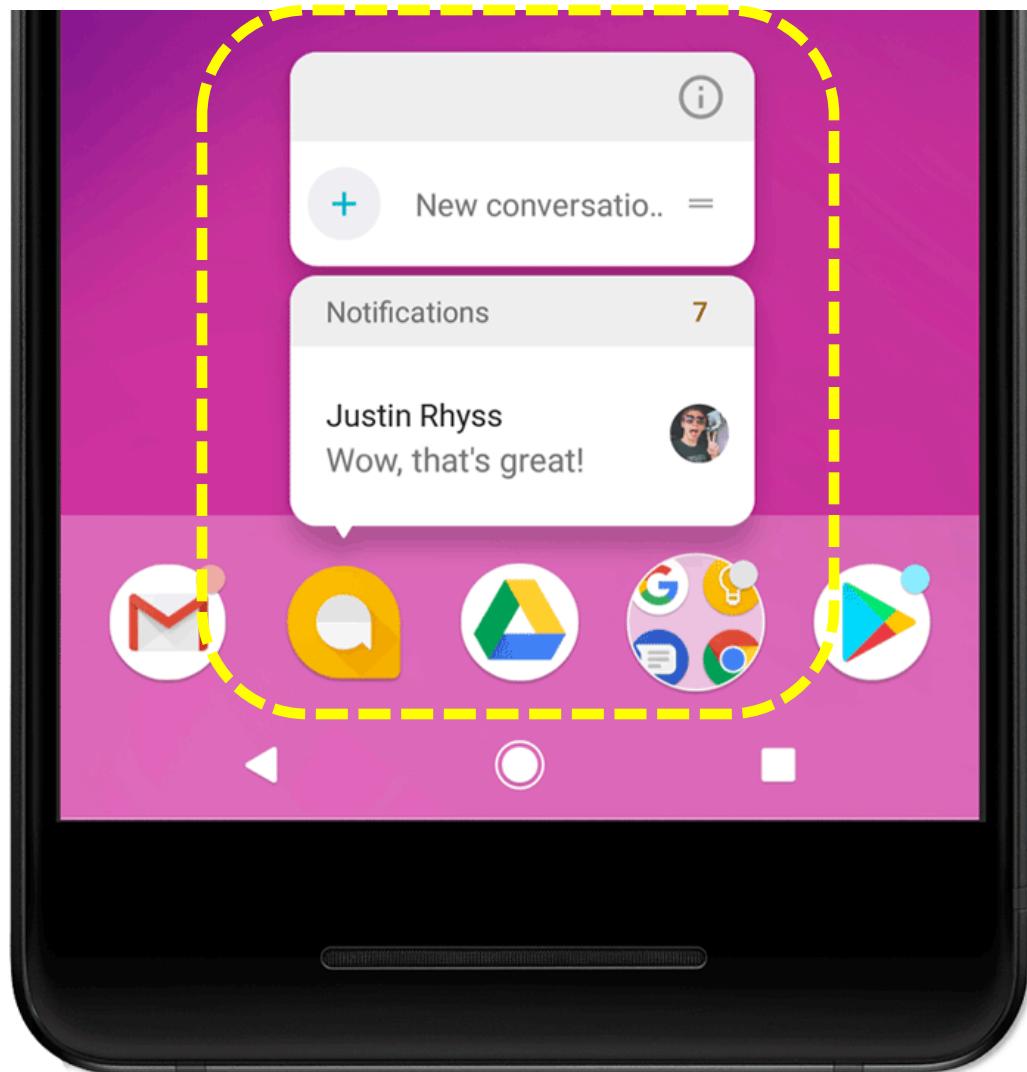
# Notifications xuất hiện trên thiết bị

## ➤ Lock screen



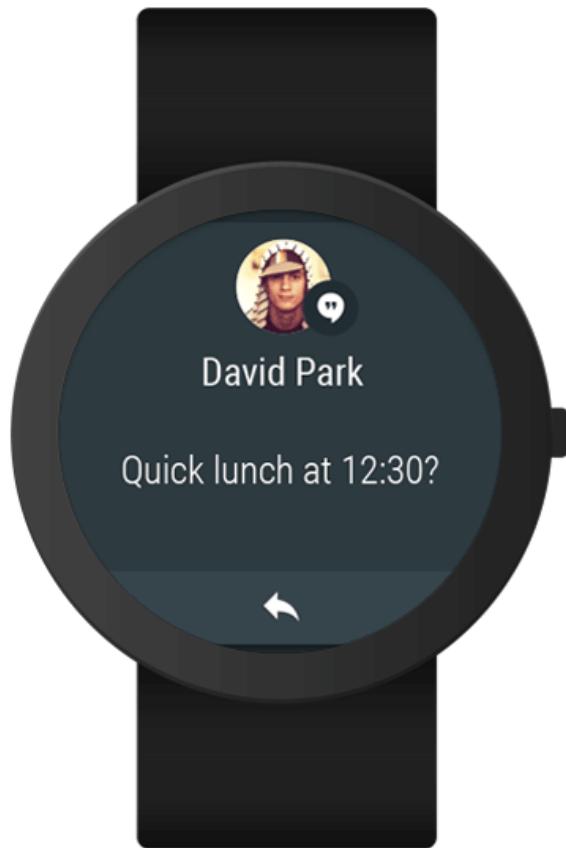
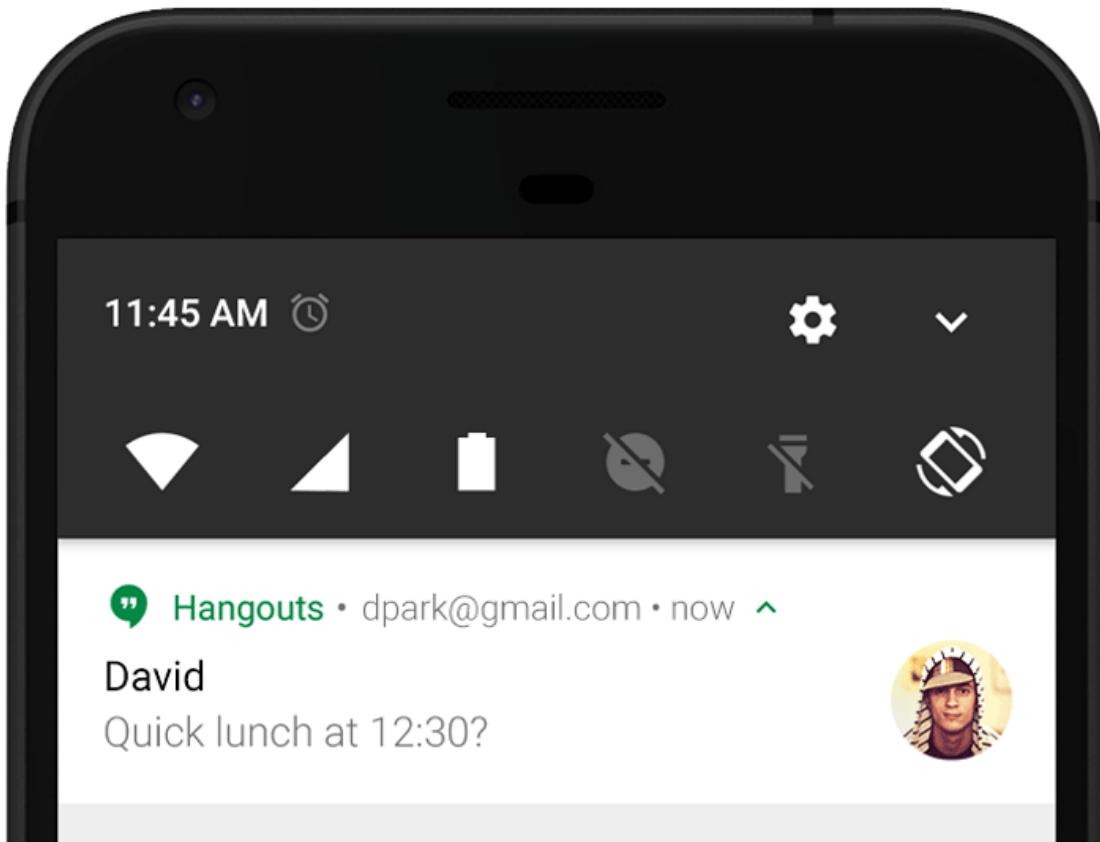
# Notifications xuất hiện trên thiết bị

- App icon badge



# Notifications xuất hiện trên thiết bị

- Wear OS devices



# Notification anatomy

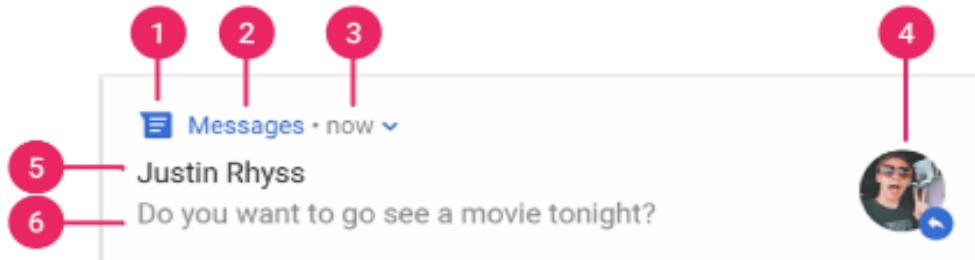


Figure 7. A notification with basic details

The most common parts of a notification are indicated in figure 7 as follows:

- ① Small icon: This is required and set with `setSmallIcon()`.
- ② App name: This is provided by the system.
- ③ Time stamp: This is provided by the system but you can override with `setWhen()` or hide it with `setShowWhen(false)`.
- ④ Large icon: This is optional (usually used only for contact photos; do not use it for your app icon) and set with `setLargeIcon()`.
- ⑤ Title: This is optional and set with `setContentTitle()`.
- ⑥ Text: This is optional and set with `setContentText()`.



# Sử dụng tài nguyên

- ❖ <https://viblo.asia/p/phan-3-cach-them-thu-vien-module-vao-android-project-yMnKMg2zl7P>

