

# **Bài 02:**

# **GIỚI THIỆU CÁC THÀNH PHẦN ỨNG DỤNG CỦA ANDROID**

GVGD: ThS. Đặng Thế Hân

Biên soạn: ThS. Giang Hào Côn

# Mục tiêu

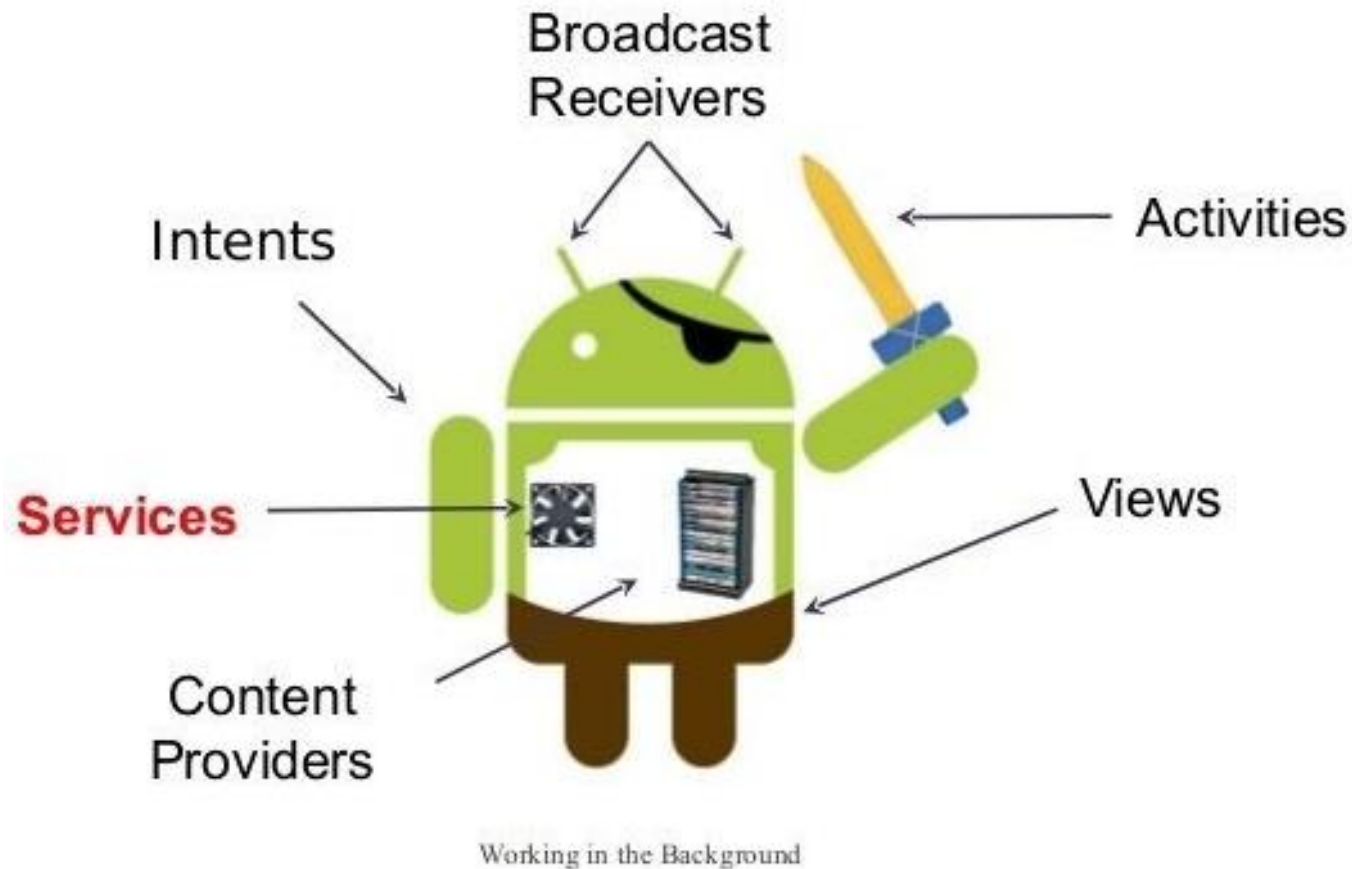
Cung cấp cho sinh viên kiến thức cơ bản về các thành phần ứng dụng của Android, và các kiểu lập trình sự kiện trong ngôn ngữ lập trình ứng dụng trên Android.

# Nội dung

- 1) Các thành phần cơ bản trong ứng dụng Android.
- 2) Ứng dụng Android và cơ chế hoạt động.
- 3) Activity và vòng đời của Activity.
- 4) Xử lý sự kiện trong ứng dụng Android.

## 2.1/ Cơ bản về ứng dụng Android

### ▪ Các thành phần cơ bản trong một ứng dụng Android



- 1) Views [?](#)
- 2) Activities [?](#)
- 3) Intents [?](#)
- 4) Broadcast Receivers [?](#)
- 5) Services [?](#)
- 6) Content Providers [?](#)

## 2.1/ Cơ bản về ứng dụng Android

- View biểu diễn một hình chữ nhật, trong đó nó hiển thị thông tin nào đó cho người dùng, và người dùng có thể tương tác với View. Nhưng loại View cơ bản cần tìm hiểu trước tiên đó là: TextView, ImageView, Button, ImageButton, EditText.
- Các thành phần giao diện xây dựng từ lớp cơ sở View (`android.view.View`) của Android, các thành phần này cung cấp sẵn khá đa dạng như Button, TextView, CheckBox ... tất cả chúng ta gọi nó là View. Sơ đồ các View được mô tả theo sơ đồ như hình dưới.



## 2.1/ Cơ bản về ứng dụng Android

- Activity là một thành phần của ứng dụng Android. Android Activity là nơi để ứng dụng tương tác trực tiếp với người dùng thông qua giao diện. Một ứng dụng có thể sẽ có nhiều màn hình và mỗi màn hình có thể là một Activity (nếu không sử dụng Fragment).
- Mỗi Activity thường hoạt động độc lập với nhau nhưng có thể tương tác và truyền dữ liệu qua nhau thông qua Intent. Chính vì Activity hoạt động độc lập nên sẽ có vòng đời riêng từ lúc được khởi tạo cho đến lúc được huỷ đi.



## 2.1/ Cơ bản về ứng dụng Android

- Intent cũng giúp liên lạc giữa các phần của một ứng dụng dễ dàng. Di chuyển từ một màn hình (Activity) sang một màn hình khác được thực hiện thông qua Intents.
- Trong Android, khả năng gửi tin nhắn đi xung quanh được thực hiện bởi đối tượng Intent. Với sự trợ giúp của Intents, các thành phần của Android có thể yêu cầu chức năng từ các thành phần khác. Khi bạn mở ứng dụng Instagram trên điện thoại của bạn và sử dụng nó để chụp ảnh, bạn chỉ cần sử dụng một Intent.



## 2.1/ Cơ bản về ứng dụng Android

1. Broadcast Receiver là một trong 4 component lớn trong Android, với mục đích là lắng nghe các sự kiện, trạng thái của hệ thống phát ra thông qua Intent nhờ đó mà các lập trình viên có thể xử lý được các sự kiện hệ thống ở bên trong ứng dụng của mình.
2. Broadcast Receiver có thể hoạt động được cả khi ứng dụng bị tắt đi, nghĩa là ở background chính vì vậy nó thường được sử dụng với service.





## 2.1/ Cơ bản về ứng dụng Android

- Một Service là một thành phần (component) có thể thực hiện các hoạt động lâu dài trong background và nó không cung cấp một giao diện người dùng.
- Một thành phần khác của ứng dụng có thể start nó, và nó tiếp tục chạy trong background ngay cả khi người dùng chuyển sang ứng dụng khác.
- Ví dụ: một Service có thể thực hiện các giao dịch mạng, chơi nhạc, ra vào file I/O hoặc tương tác với một content provider, tất cả đều từ background.



## 2.1/ Cơ bản về ứng dụng Android

Content provider là một thành phần để quản lý truy cập dữ liệu, nó cung cấp các phương thức khác nhau để các ứng dụng có thể truy cập dữ liệu từ một ứng dụng khác bằng cách sử dụng ContentResolver. Content Provider có thể giúp cho một ứng dụng quản lý quyền truy cập đến dữ liệu được lưu bởi ứng dụng đó, hoặc các ứng dụng khác, và đó là một cách để ta có thể chia sẻ dữ liệu cho các ứng dụng khác nhau. Hình dưới đây biểu diễn cho việc cách content providers quản lý việc truy cập tới bộ nhớ.



## 2.2/ Cơ chế hoạt động

- Ứng dụng Android được viết bằng ngôn ngữ Java và biên dịch, đóng gói cùng các tập tin tài nguyên thành tập tin \*.apk.
- Cài đặt trên thiết bị theo đường dẫn **data/app/<Tên đóng gói>**, được chứa trong **Sandbox** và được hiểu:
  - Mỗi ứng dụng là một dạng “người dùng” khác nhau.
  - Mỗi ứng dụng được cấp một ID, do đó chỉ duy nhất ứng dụng mới có thể truy xuất các tập tin liên quan đến ứng dụng đó.

## 2.2/ Cơ chế hoạt động

- Ứng dụng thực thi riêng biệt trên từng máy ảo.
- Tiến trình Linux được cấp phát khi bất cứ thành phần ứng dụng được gọi thực thi, và thu hồi khi chấm dứt hoạt động.
- Các ứng dụng có cùng ID và chứng chỉ (Certificate) có thể truy xuất tài nguyên của nhau, hoặc xin quyền nếu truy xuất hệ thống.

## 2.3/ Tìm hiểu Activity là gì ?

- **Activity** là một **thành phần cơ bản** của ứng dụng Android, có hiển thị giao diện và các xử lý tương tác với người sử dụng.
- Một ứng dụng Android có thể có một hoặc nhiều Activity, Activity được chạy đầu tiên khi khởi động ứng dụng gọi là **Activity chính (main-activity)**.
- Một lớp được gọi là Activity khi nó extend (kế thừa) từ những lớp cha như **Activity**, **AppCompatActivity** hay **FragmentActivity**.

## 2.3/ Tìm hiểu Activity là gì ?

- Mỗi Activity sẽ hoạt động độc lập với nhau nhưng có thể tương tác và truyền dữ liệu qua nhau.



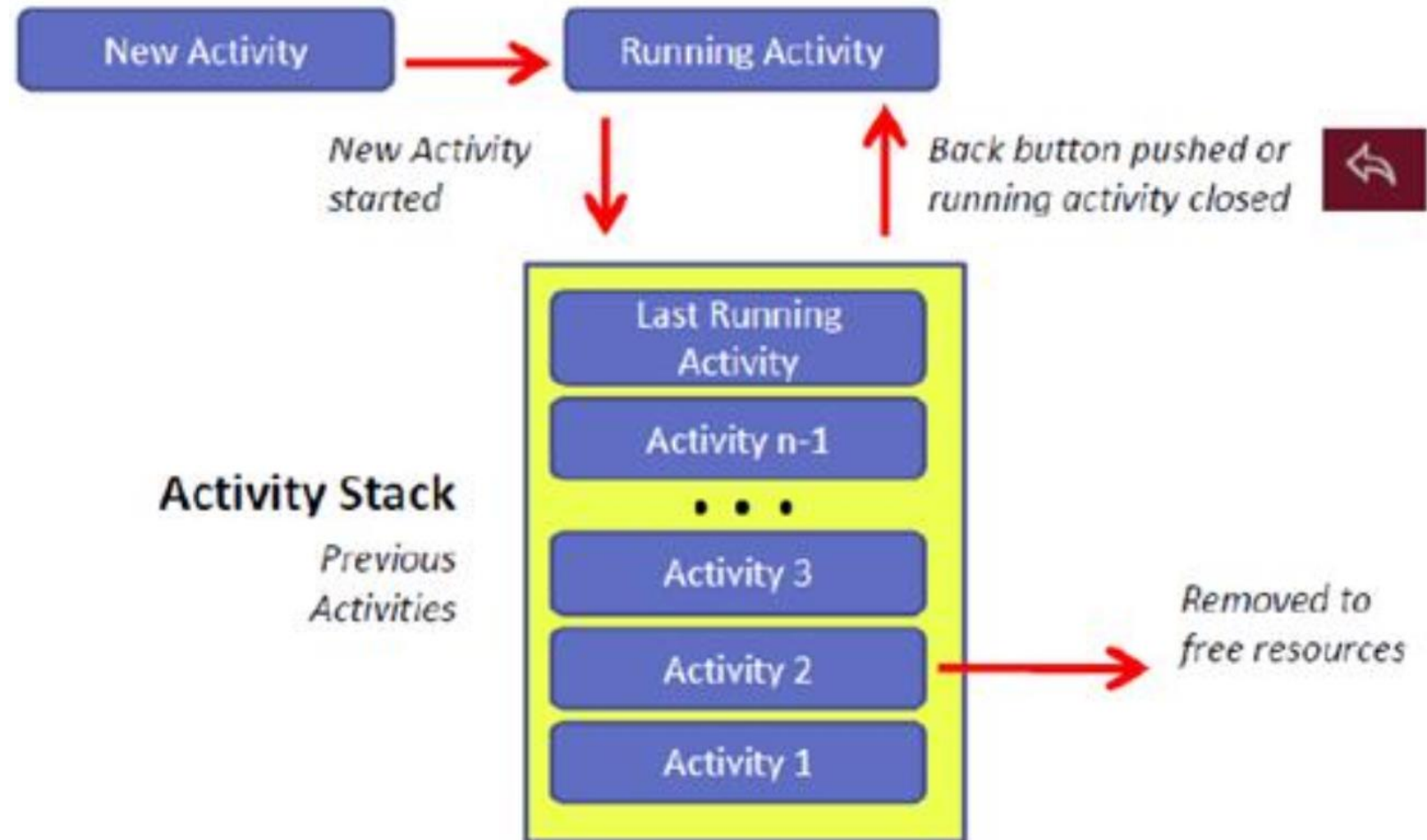
## 2.4/ Hoạt động của Activity

- Trong quá trình hoạt động, mỗi activity có thể khởi động một Activity khác để thực hiện các tác vụ khác.
- Khi một Activity mới được kích hoạt, Activity hiện hành sẽ bị tạm dừng và sẽ được đặt vào **ngăn xếp lùi (Back-Stack)**.
  - Back-stack hoạt động theo cơ chế LIFO.
  - Activity sẽ trải qua một số các trạng thái nhất định trong vòng đời của nó.



## 2.4/ Hoạt động của Activity

- Hình ảnh minh họa back-stack.





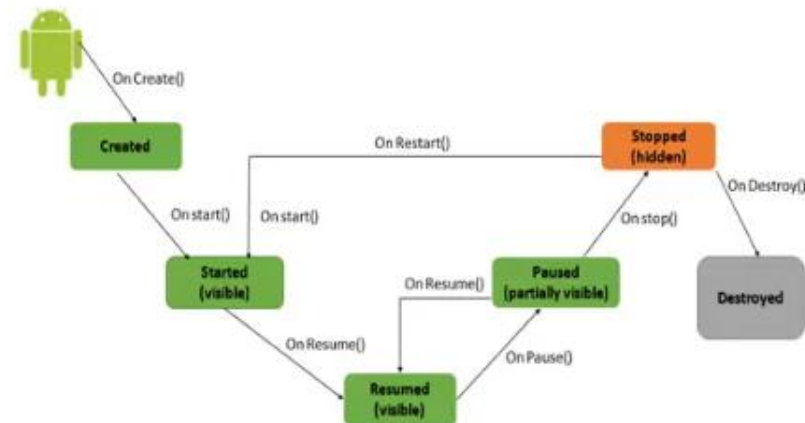
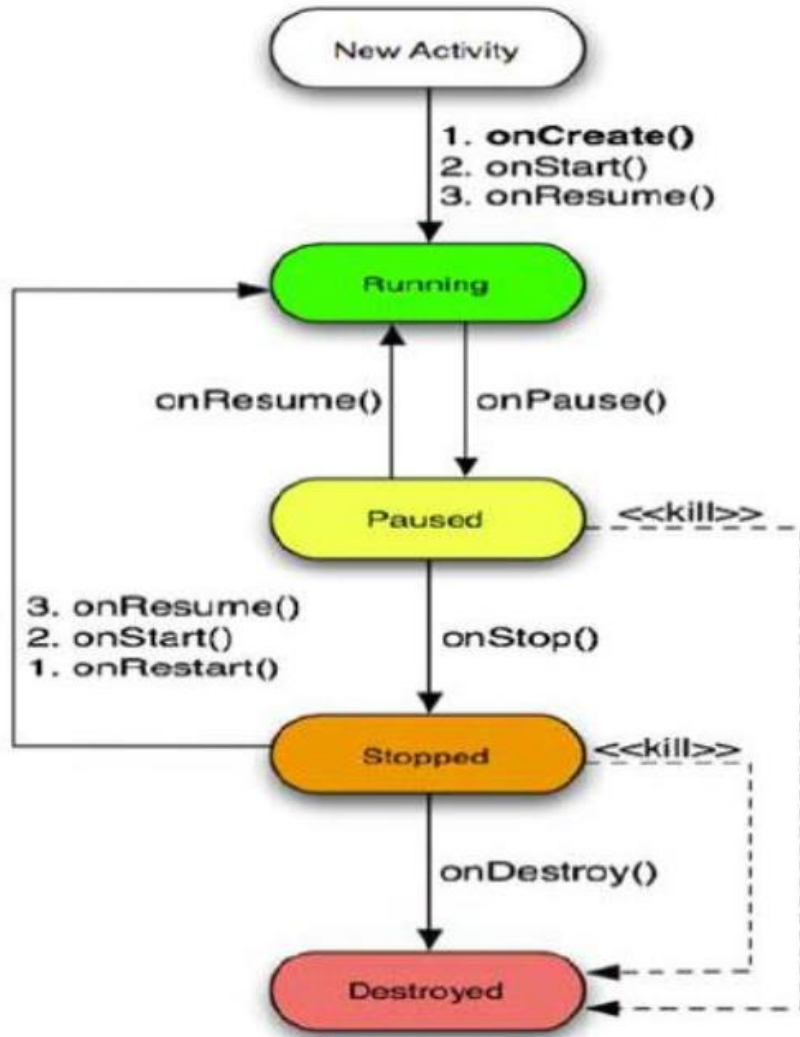
## 2.5/ Trạng thái của Activity

1) Running (Hoạt động) ?

2) Paused (tạm dừng) ?

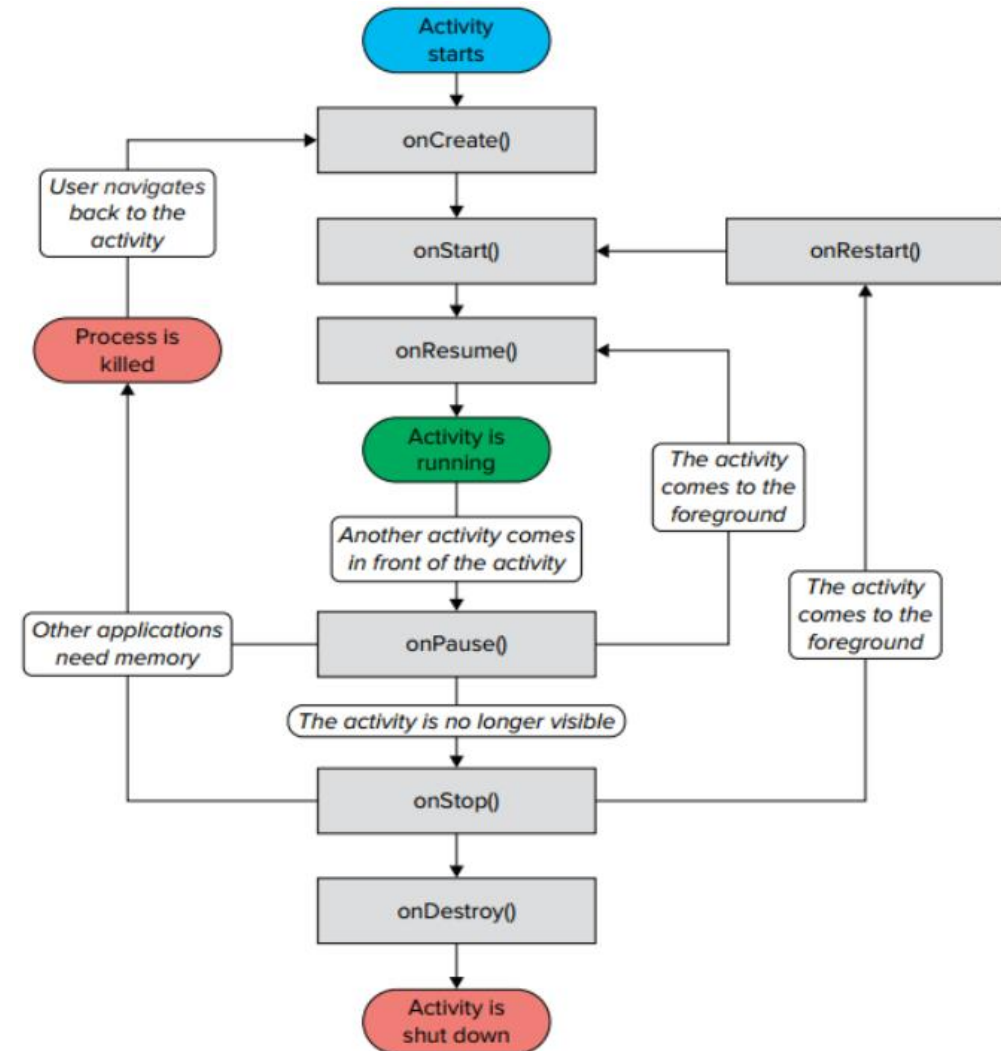
3) Stopped (bị dừng) ?

4) Destroyed (kết thúc) ?



## 2.6/ Vòng đời của Activity

- Khi một activity **chuyển đổi hoặc thoát khỏi các trạng thái khác nhau**, activity sẽ được thông báo qua nhiều phương thức callback khác nhau (**onCreate, onPause, onResume,...**)
- Vòng đời của Activity mô tả quá trình hoạt động và tương tác của một Activity kể từ khi nó bắt đầu chạy, cho tới khi kết thúc.



## 2.6/ Vòng đời của Activity

### 3 chu kỳ trong vòng đời của Activity

- 1) Từ khi gọi `onCreate()` cho tới `onDestroy()` – tức là từ lúc Activity được gọi ra cho đến lúc nó bị huỷ gọi là chu kỳ **Entire lifetime**.
- 2) Từ khi gọi `onStart()` cho tới lúc gọi `onStop()`, trong trường hợp này ta vẫn có thể thấy màn hình Activity gọi là chu kỳ **Visible lifetime**.
- 3) Từ khi gọi `onResume()` cho tới lúc gọi `onPause()`, quá trình này Activity luôn nằm ở foreground và ta có thể tương tác được với nó gọi là chu kỳ **Foreground lifetime**.

## 2.7/ Các phương thức callback

❑ **onCreate()** – được gọi khi activity được kích hoạt, trước khi hiển thị giao diện.

- Callback này chỉ được gọi một lần duy nhất.
- Dùng để khởi tạo cho activity như load giao diện, các lời gọi API, load database,...

❑ **onStart()** – được gọi khi Activity bắt đầu được hiện ra, trước khi nhận tương tác với người dùng.

- Callback này ít dùng trong lập trình

## 2.7/ Các phương thức callback

- ❑ **onResume()** – được gọi khi Activity đã nhìn thấy và nhận tương tác với người dùng.
  - Callback dùng để khôi phục hoạt động các tác vụ
- ❑ **onPause()** – được gọi khi activity khi có thành phần nào đó che một phần Activity hiện tại.
  - Thường dùng để thực hiện tạm dừng các tác vụ đang chạy, như tạm dừng sound, game pause...
- ❑ **onStop()** – được gọi khi Activity bị che khuất hoàn toàn hoặc nhấn nút Home.
  - Callback này ít dùng trong lập trình

## 2.7/ Các phương thức callback

❑ **onDestroy()** – được gọi khi Activity kết thúc, như bị hệ thống kill hoặc người tắt ứng dụng.

❑ Callback được dùng để giải phóng tài nguyên

Để hiểu rõ hơn về các hàm trong Activity chúng ta cùng đi sang phần thực hành để biết cách thức hoạt động của các hàm.

❑ Mở file MainActivity của project HelloAndroid, ghi đè các phương thức callback và thêm log hiển thị

❑ Dùng `Toast.makeText().show()`

❑ Hoặc: `Log.d()`, `Log.e()`, `Log.w()`, `Log.i()`

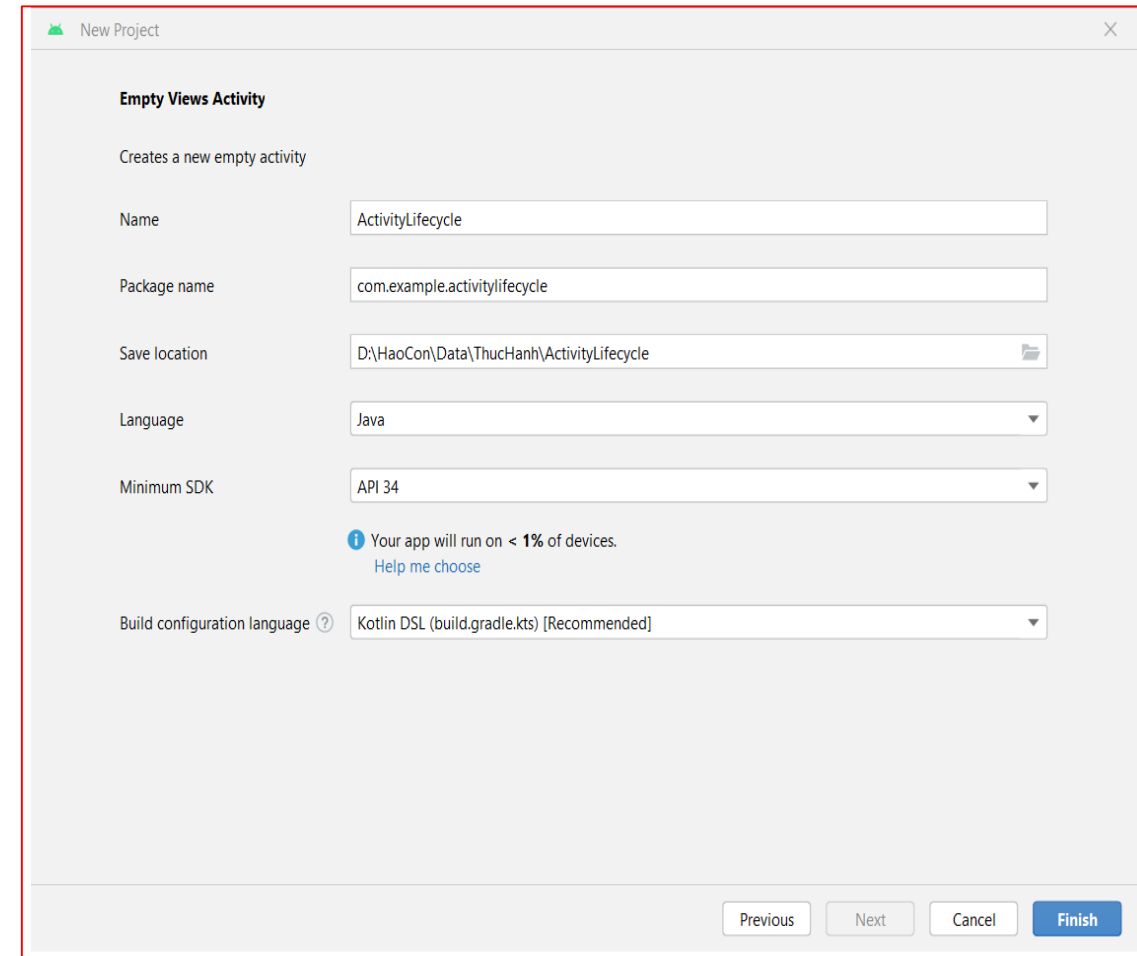
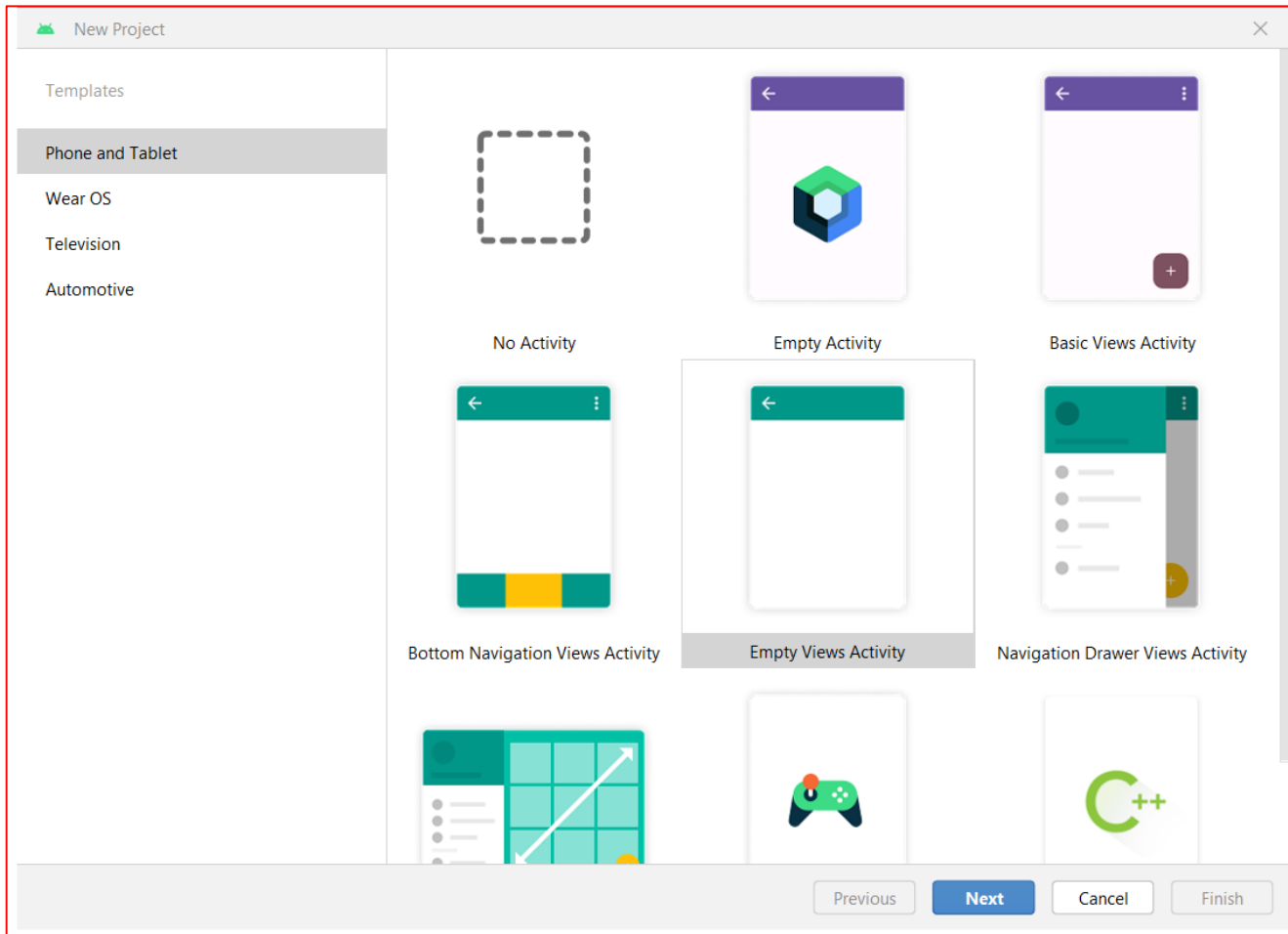
❑ Thực hiện các tác làm ảnh hưởng để hoạt động của Activity và cho nhận xét hiển thị ở cửa sổ logcat.

❑ Chạy ứng dụng khác, nhấn nút Back, nút Home,...



# 2.8/Thực hành làm quen với Activity

## ■ Tạo project như sau:



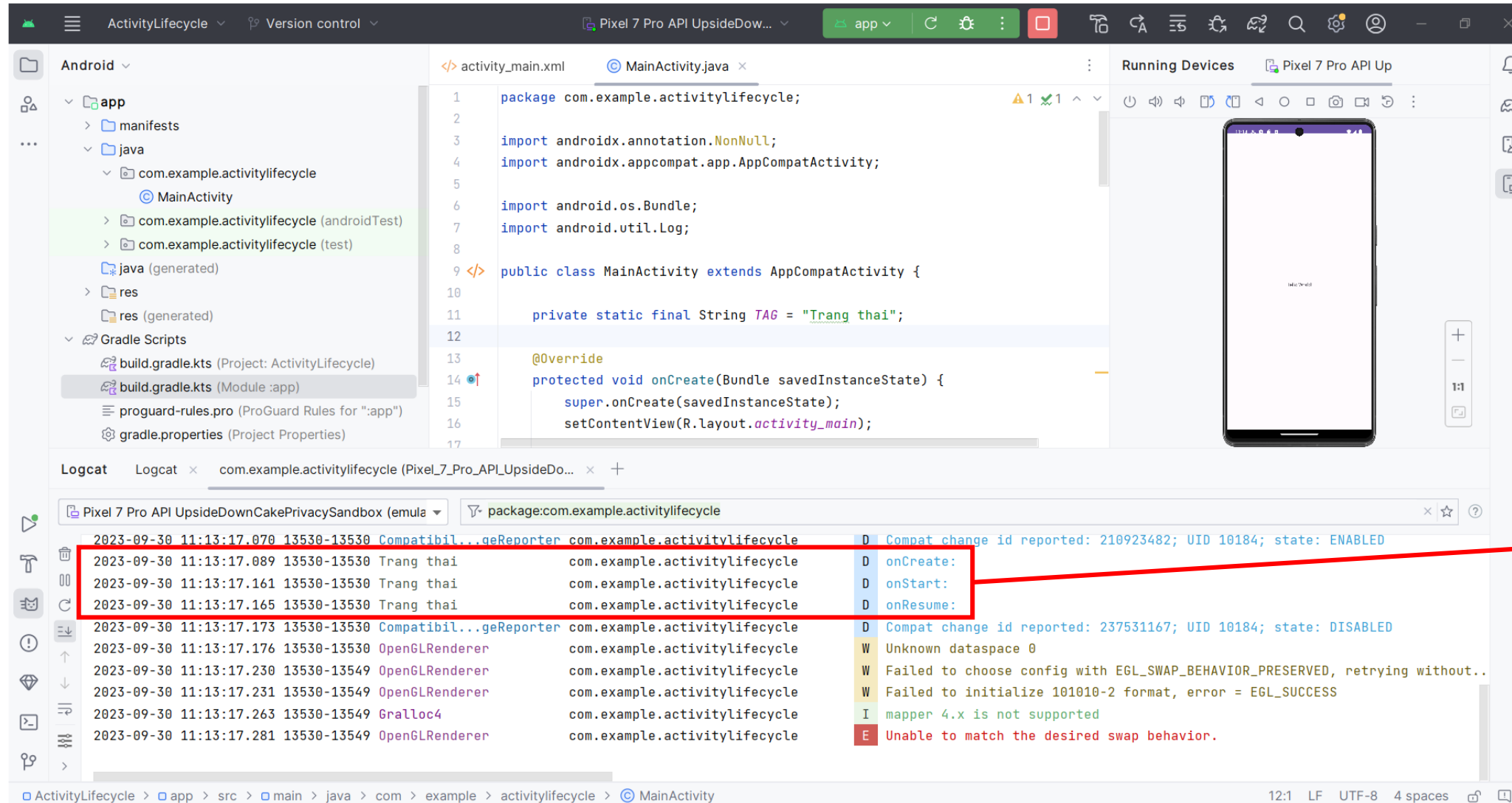
## 2.8/Thực hành làm quen với Activity

- Tiếp theo chúng ta thêm các hàm trạng thái của Activity để xem khi thay đổi trạng thái chương trình sẽ xử lý như thế nào.
- Ở đây chúng ta thêm đủ các hàm `onCreate`, `onRestart`, `onStop`, `onPause`, `onStart`, `onResume`, `onDestroy`, `onSaveInstanceState`, `onRestoreInstanceState`
- Và không quên đặt log bằng hàm `log.d()` để xem trạng thái ở cửa sổ `logcat`.

### Code demo



# 2.8/Thực hành làm quen với Activity



Android Studio interface showing the MainActivity.java file and the Logcat window.

The MainActivity.java file contains the following code:

```
1 package com.example.activitylifecycle;
2
3 import androidx.annotation.NonNull;
4 import androidx.appcompat.app.AppCompatActivity;
5
6 import android.os.Bundle;
7 import android.util.Log;
8
9 <?> public class MainActivity extends AppCompatActivity {
10
11     private static final String TAG = "Trang thai";
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17     }
18 }
```

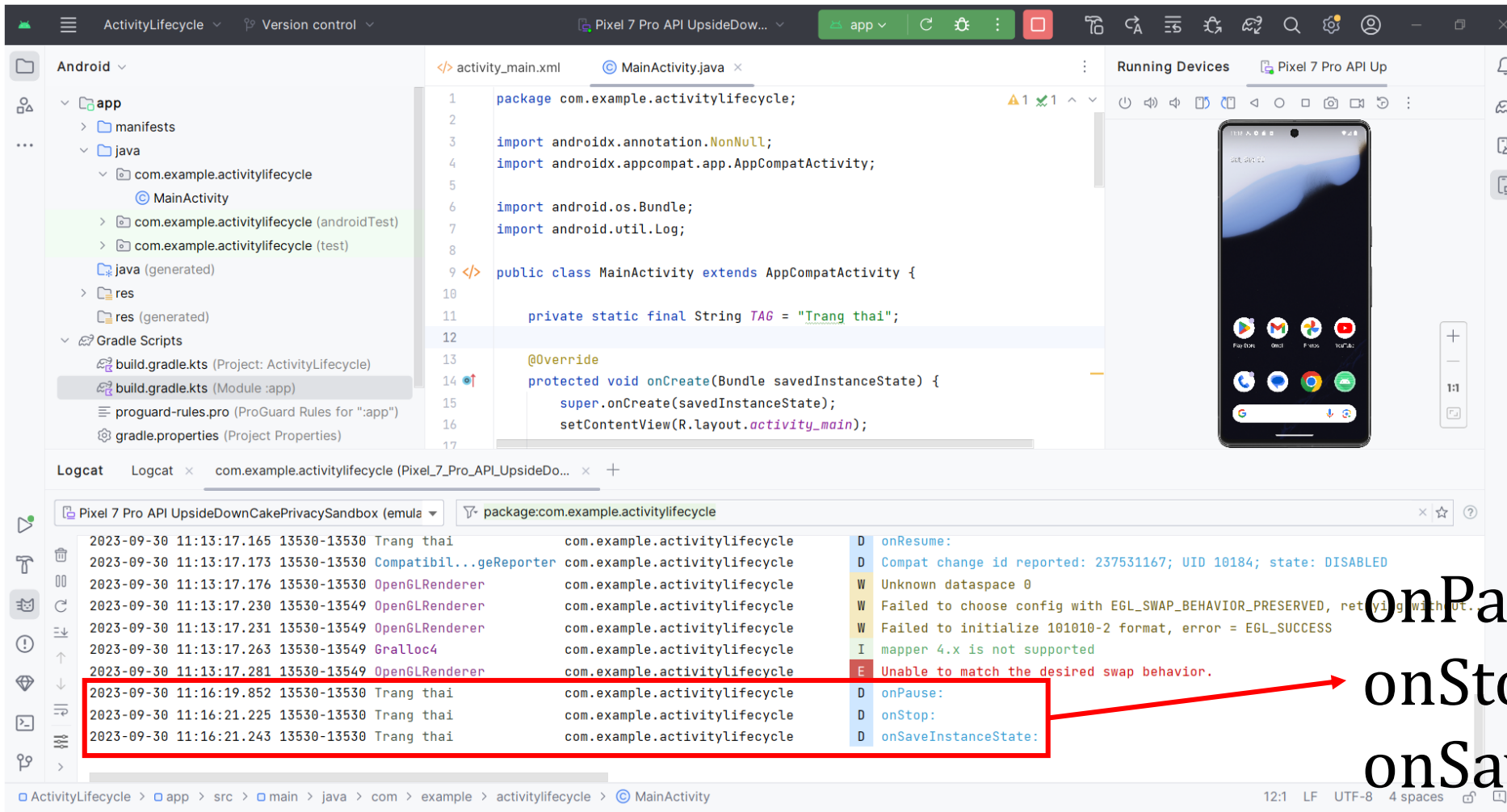
The Logcat window shows the following log entries:

Time	Priority	Tag	Message
2023-09-30 11:13:17.070	INFO	Compatibil...geReporter	Compat change id reported: 210923482; UID 10184; state: ENABLED
2023-09-30 11:13:17.089	INFO	Trang thai	onCreate:
2023-09-30 11:13:17.161	INFO	Trang thai	onStart:
2023-09-30 11:13:17.165	INFO	Trang thai	onResume:
2023-09-30 11:13:17.173	INFO	Compatibil...geReporter	Compat change id reported: 237531167; UID 10184; state: DISABLED
2023-09-30 11:13:17.176	WARN	OpenGLRenderer	Unknown dataspace 0
2023-09-30 11:13:17.230	WARN	OpenGLRenderer	Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without..
2023-09-30 11:13:17.231	WARN	OpenGLRenderer	Failed to initialize 101010-2 format, error = EGL_SUCCESS
2023-09-30 11:13:17.263	INFO	Gralloc4	mapper 4.x is not supported
2023-09-30 11:13:17.281	ERROR	OpenGLRenderer	Unable to match the desired swap behavior.

onCreate  
onStart  
onResume

# 2.8/Thực hành làm quen với Activity

Bây giờ chúng ta bấm nút home để thoát khỏi ứng dụng.



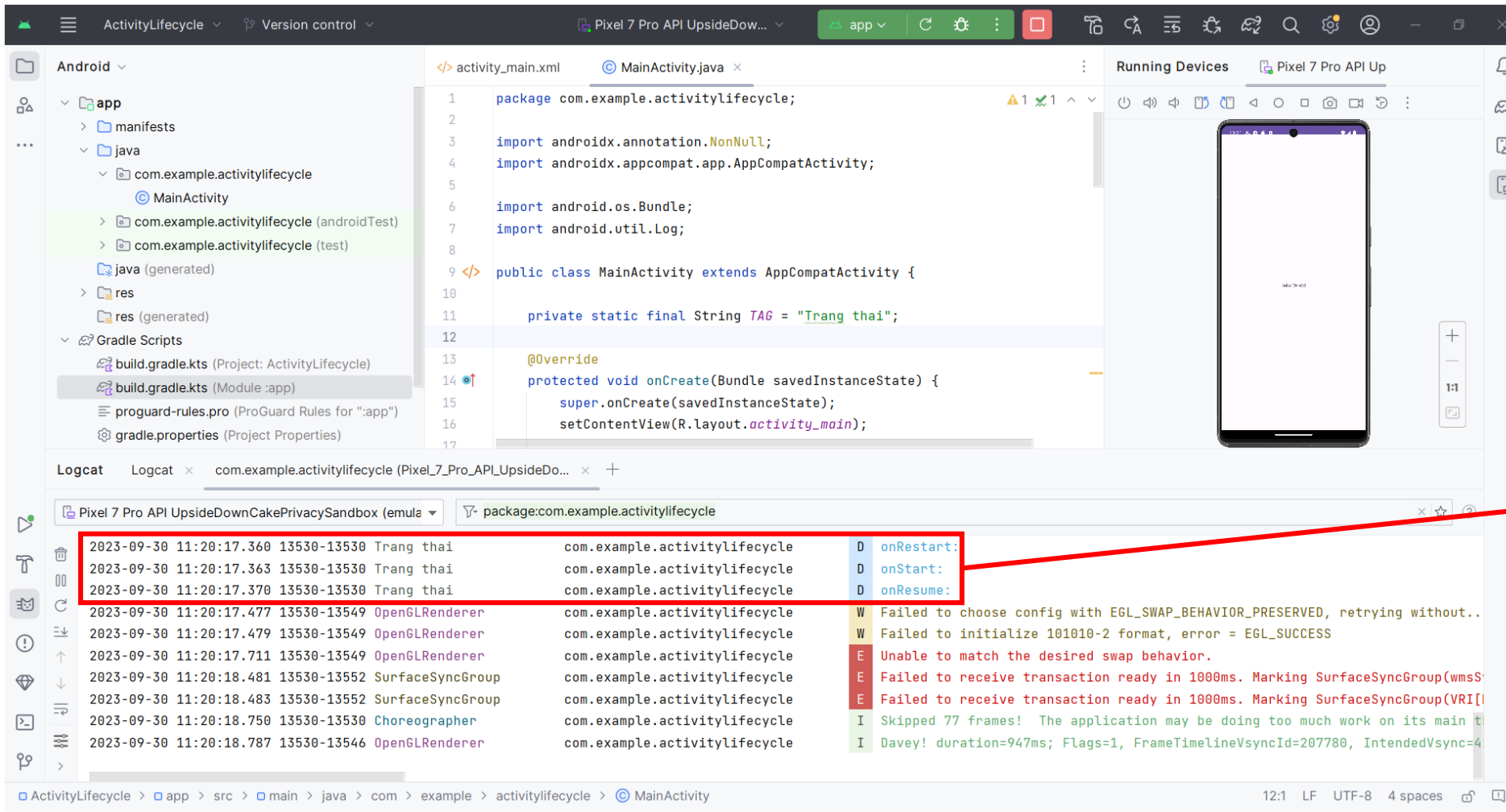
The screenshot shows the Android Studio interface. The main editor displays the `MainActivity.java` file, which is a simple `AppCompatActivity` subclass. The `onCreate` method is implemented, and the `setContentView` method is called. The `Logcat` window at the bottom shows a series of log messages. The last three messages are highlighted in red, indicating the sequence of events when the user presses the home button:

- `D onPause:`
- `D onStop:`
- `D onSaveInstanceState:`

A red arrow points from the text  `onPause, onStop, onSaveInstanceState` to the highlighted log messages.

# 2.8/Thực hành làm quen với Activity

## Tiếp tục mở lại ứng dụng.



```
package com.example.activitylifecycle;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "Trang thai";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

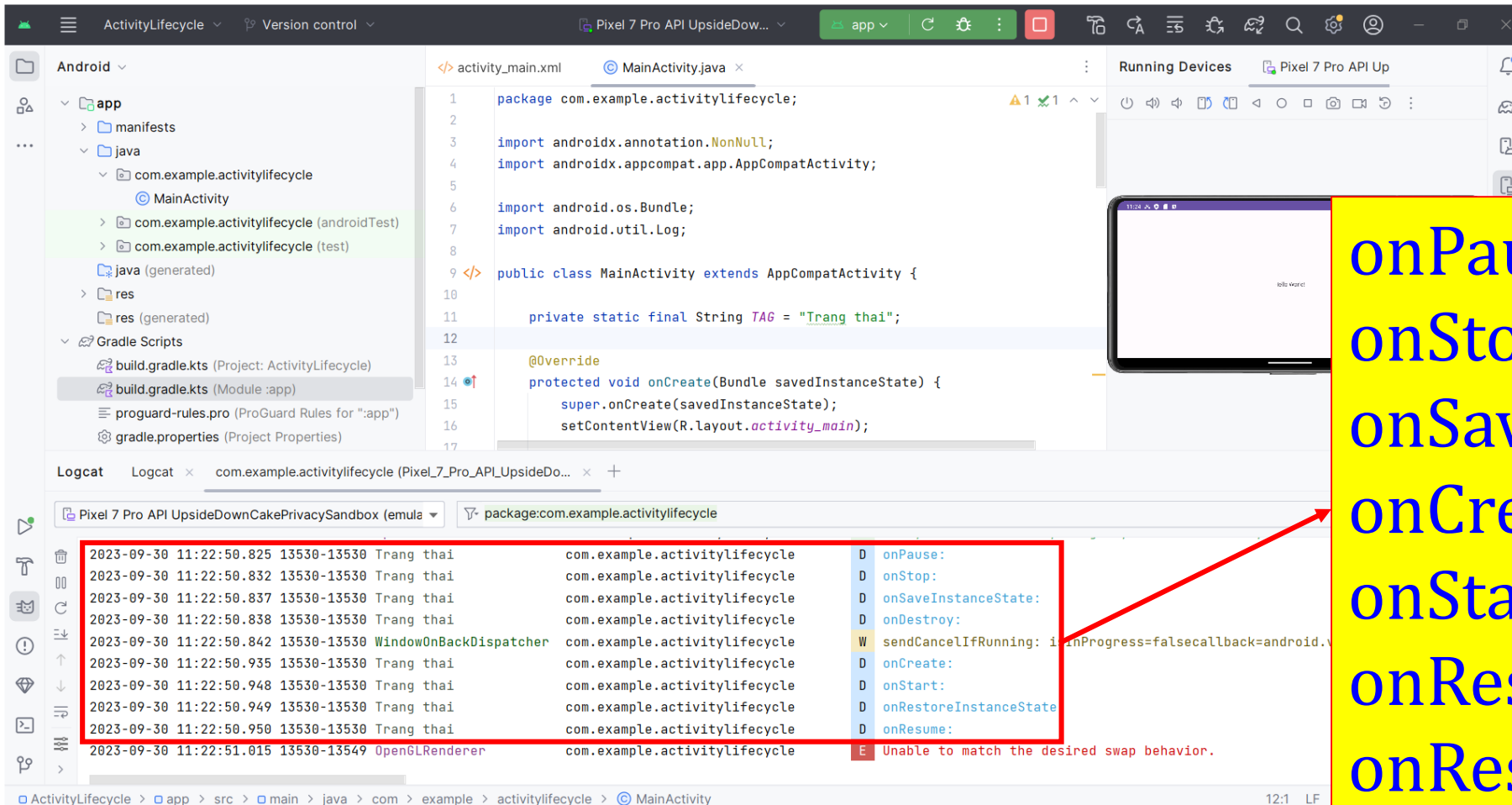
Logcat

Time	Level	Package	Class	Message
2023-09-30 11:20:17.360	D	com.example.activitylifecycle	onRestart:	
2023-09-30 11:20:17.363	D	com.example.activitylifecycle	onStart:	
2023-09-30 11:20:17.370	D	com.example.activitylifecycle	onResume:	
2023-09-30 11:20:17.477	W	com.example.activitylifecycle	Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without..	
2023-09-30 11:20:17.479	W	com.example.activitylifecycle	Failed to initialize 101010-2 format, error = EGL_SUCCESS	
2023-09-30 11:20:17.711	E	com.example.activitylifecycle	Unable to match the desired swap behavior.	
2023-09-30 11:20:18.481	E	com.example.activitylifecycle	Failed to receive transaction ready in 1000ms. Marking SurfaceSyncGroup(wmsS	
2023-09-30 11:20:18.483	E	com.example.activitylifecycle	Failed to receive transaction ready in 1000ms. Marking SurfaceSyncGroup(VRI	
2023-09-30 11:20:18.750	I	com.example.activitylifecycle	Skipped 77 frames! The application may be doing too much work on its main t	
2023-09-30 11:20:18.787	I	com.example.activitylifecycle	Davey! duration=947ms; Flags=1, FrameTimelineVsyncId=207780, IntendedVsync=4	

onRestart  
onStart  
onResume

# 2.8/Thực hành làm quen với Activity

Tiếp theo chúng ta xoay ngang màn hình.



The screenshot shows the Android Studio interface with the following components:

- Project Explorer:** Shows the project structure with the package `com.example.activitylifecycle` and the `MainActivity` class.
- MainActivity.java:** Contains the following code:
 

```

package com.example.activitylifecycle;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "Trang thai";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

```
- Logcat:** Shows a list of log messages. A red box highlights the following messages:
 

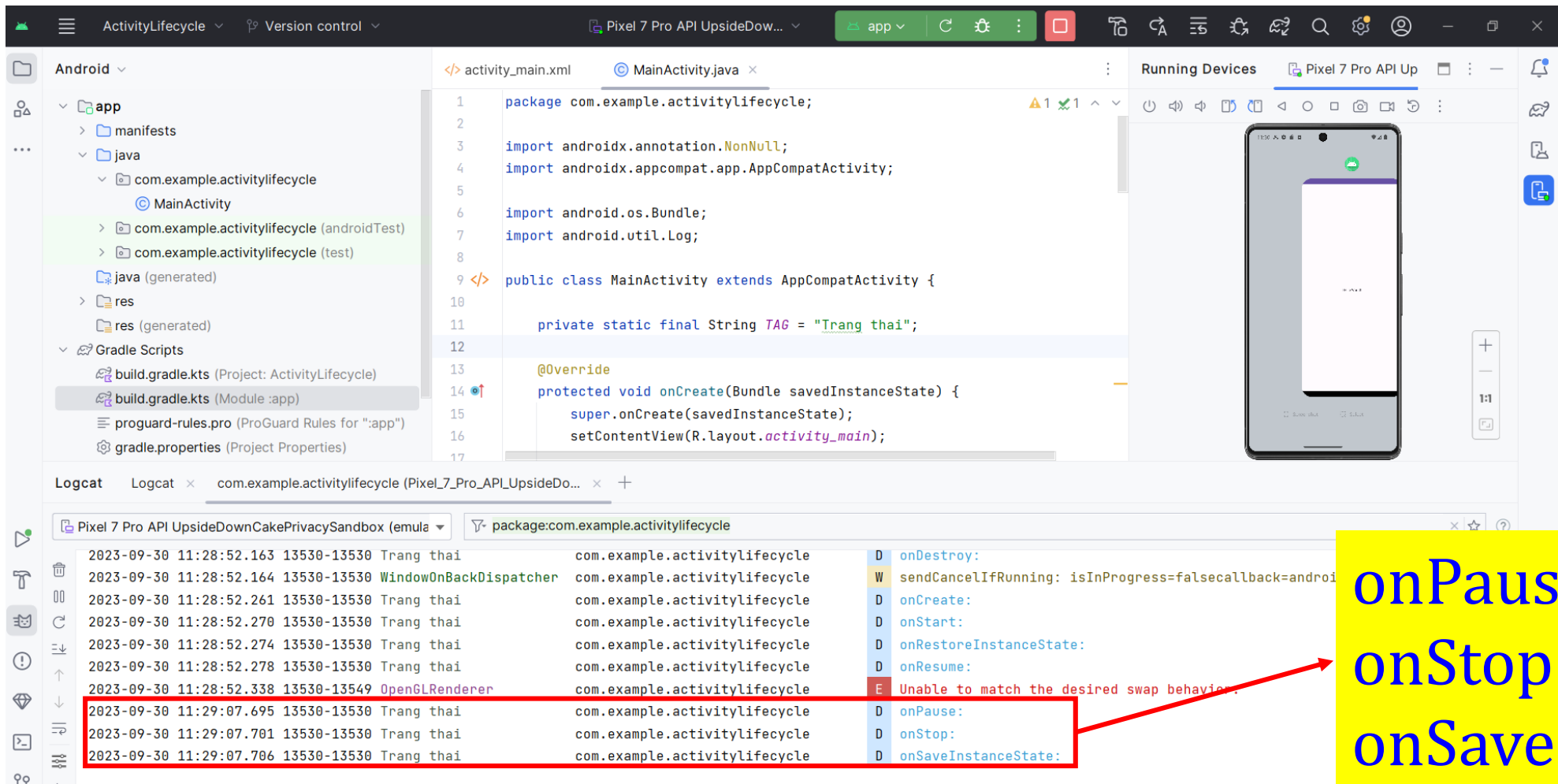
Time	Priority	Package	Message
2023-09-30 11:22:50.825	13530-13530	com.example.activitylifecycle	Trang thai
2023-09-30 11:22:50.832	13530-13530	com.example.activitylifecycle	Trang thai
2023-09-30 11:22:50.837	13530-13530	com.example.activitylifecycle	Trang thai
2023-09-30 11:22:50.838	13530-13530	com.example.activitylifecycle	Trang thai
2023-09-30 11:22:50.842	13530-13530	com.example.activitylifecycle	WindowOnBackDispatcher
2023-09-30 11:22:50.935	13530-13530	com.example.activitylifecycle	Trang thai
2023-09-30 11:22:50.948	13530-13530	com.example.activitylifecycle	Trang thai
2023-09-30 11:22:50.949	13530-13530	com.example.activitylifecycle	Trang thai
2023-09-30 11:22:50.950	13530-13530	com.example.activitylifecycle	Trang thai
2023-09-30 11:22:51.015	13530-13549	com.example.activitylifecycle	OpenGLRenderer

onPause  
onStop  
onSaveInstanceState  
onCreate  
onStart  
onRestoreInstanceState  
onResume



## 2.8/Thực hành làm quen với Activity

Tiếp theo bấm nút Overview và kéo màn hình sang phải.



The screenshot shows the Android Studio interface. The top bar indicates the project is 'ActivityLifecycle' and the device is 'Pixel 7 Pro API UpsideDown...'. The left sidebar shows the project structure with 'app' as the selected module. The main editor displays the 'activity\_main.xml' and 'MainActivity.java' files. The 'MainActivity.java' file contains the following code:

```
1 package com.example.activitylifecycle;
2
3 import androidx.annotation.NonNull;
4 import androidx.appcompat.app.AppCompatActivity;
5
6 import android.os.Bundle;
7 import android.util.Log;
8
9 <?> public class MainActivity extends AppCompatActivity {
10
11     private static final String TAG = "Trang thai";
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17     }
18 }
```

The bottom Logcat window shows the following log messages:

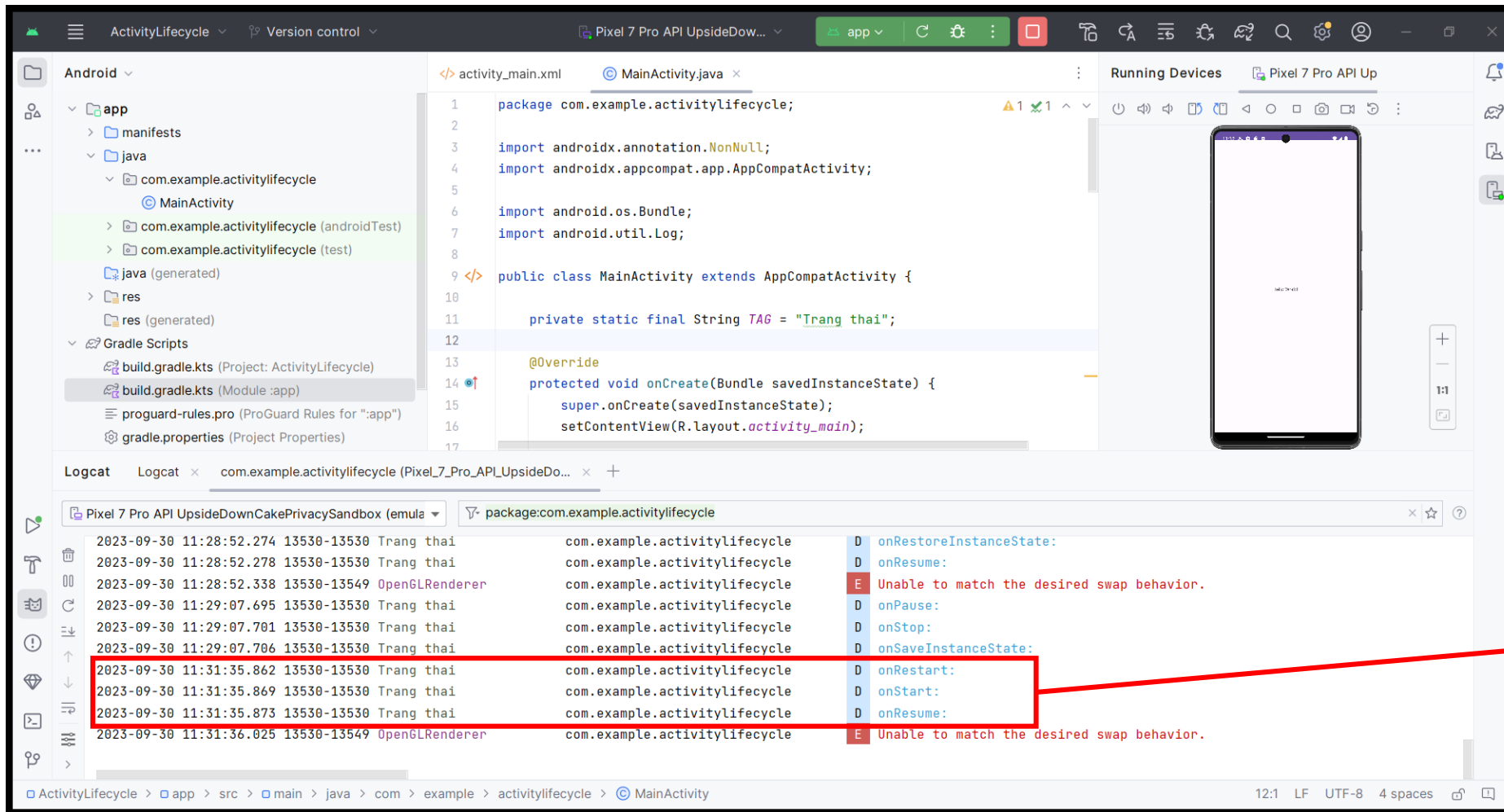
Time	Log ID	Message	Category
2023-09-30 11:28:52.163	13530-13530	Trang thai	com.example.activitylifecycle
2023-09-30 11:28:52.164	13530-13530	WindowOnBackPressedDispatcher	com.example.activitylifecycle
2023-09-30 11:28:52.261	13530-13530	Trang thai	com.example.activitylifecycle
2023-09-30 11:28:52.270	13530-13530	Trang thai	com.example.activitylifecycle
2023-09-30 11:28:52.274	13530-13530	Trang thai	com.example.activitylifecycle
2023-09-30 11:28:52.278	13530-13530	Trang thai	com.example.activitylifecycle
2023-09-30 11:28:52.338	13530-13549	OpenGLRenderer	com.example.activitylifecycle
2023-09-30 11:29:07.695	13530-13530	Trang thai	com.example.activitylifecycle
2023-09-30 11:29:07.701	13530-13530	Trang thai	com.example.activitylifecycle
2023-09-30 11:29:07.706	13530-13530	Trang thai	com.example.activitylifecycle

The 'onPause' message is highlighted in red in the original image. A red arrow points from this message to a yellow box containing the following text:

onPause  
onStop  
onSaveInstanceState

# 2.8/Thực hành làm quen với Activity

Tiếp tục quay trở lại ứng dụng.



The screenshot shows the Android Studio interface with the MainActivity.java file open. The Logcat window at the bottom displays a sequence of log messages for the MainActivity. A red box highlights the following messages:

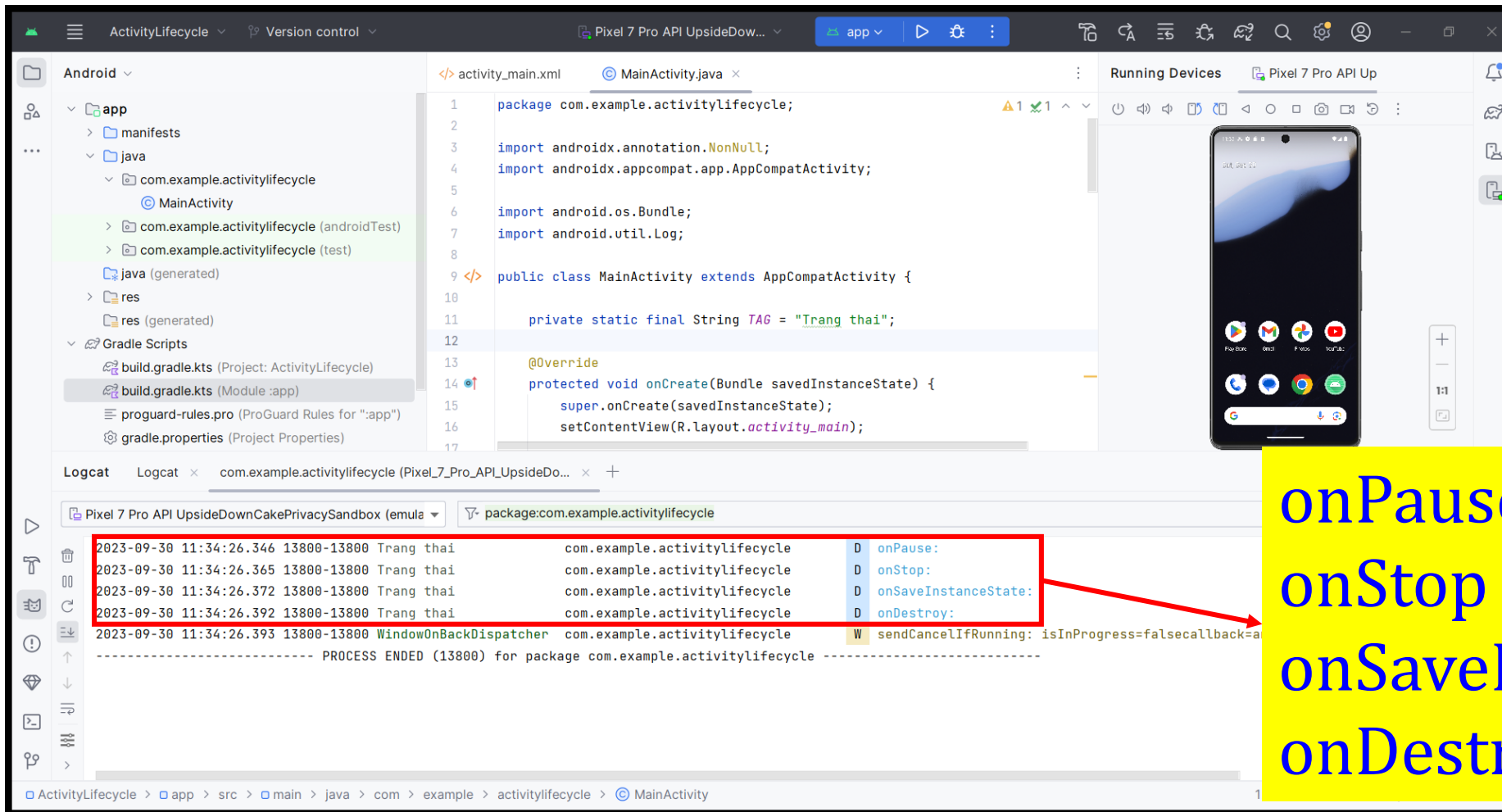
- 2023-09-30 11:31:35.862 13530-13530 Trang thai com.example.activitylifecycle D onRestart:
- 2023-09-30 11:31:35.869 13530-13530 Trang thai com.example.activitylifecycle D onStart:
- 2023-09-30 11:31:35.873 13530-13530 Trang thai com.example.activitylifecycle D onResume:

A red arrow points from the highlighted messages to the text "onRestart", "onStart", and "onResume" on the right side of the slide.

onRestart  
onStart  
onResume

# 2.8/Thực hành làm quen với Activity

Cuối cùng chúng ta thoát khỏi ứng dụng.



onPause  
onStop  
onSaveInstanceState  
onDestroy



**savedInstanceState** cũng là một trong các thành phần của trạng thái trong vòng đời của một Activity. Đây là :

- 1) Một loại dữ liệu không bền vững.
- 2) Không được lưu trữ cụ thể trong đâu ngoài bộ nhớ RAM.
- 3) Nó được sử dụng để truyền, phục hồi, lưu trạng thái của một Activity.
- 4) Dữ liệu trong savedInstanceState được lưu dưới dạng Bundle.
- 5) Được phục hồi khi phương thức **onCreate** và **onRestoreSavedInstanceState** được gọi.
- 6) Được lưu trước **onStop**, với phương thức **onSaveInstanceState**.



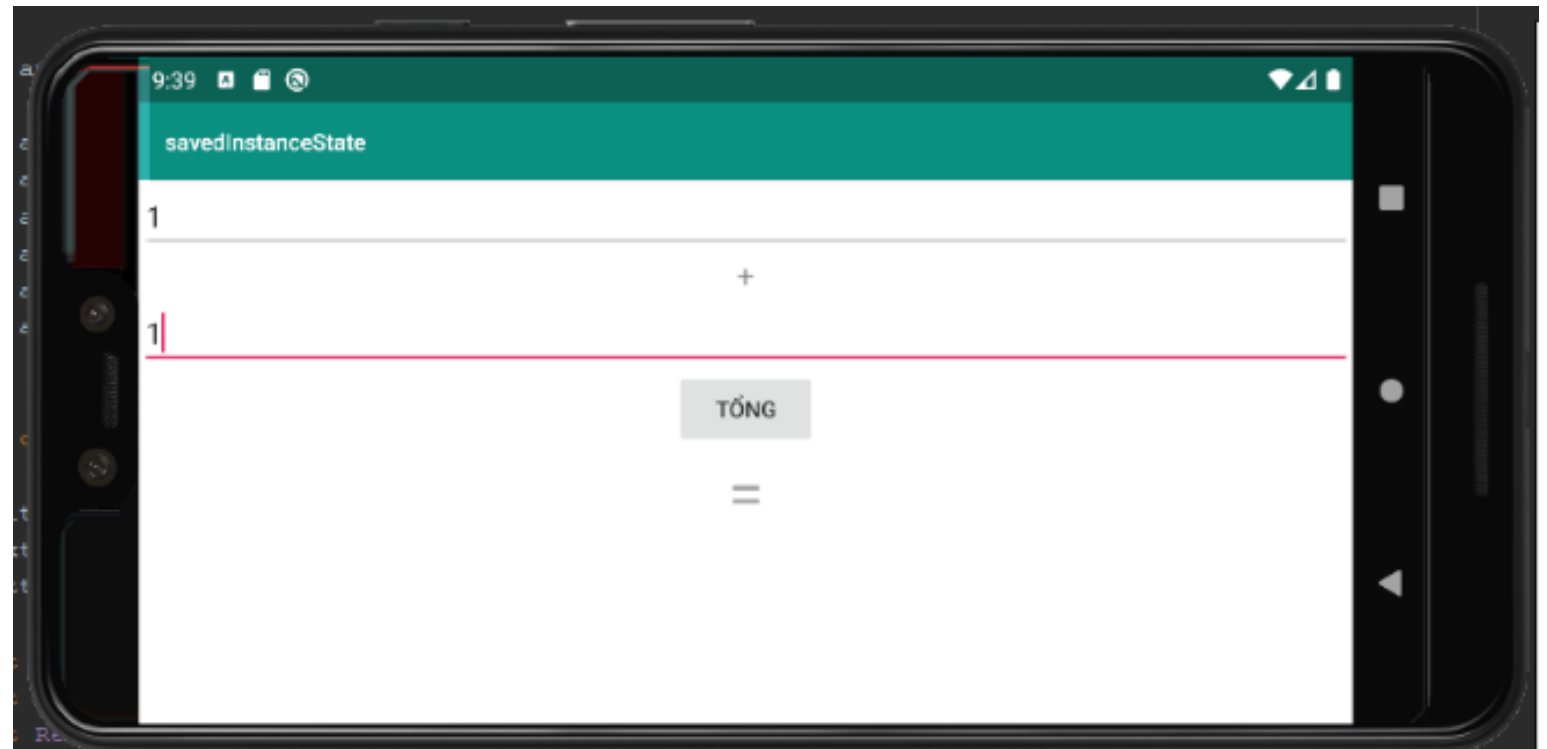
## 2.9/ Làm quen với savedInstanceState



Ví dụ:



Mọi thứ vẫn bình thường đến khi xoay ngang màn hình.



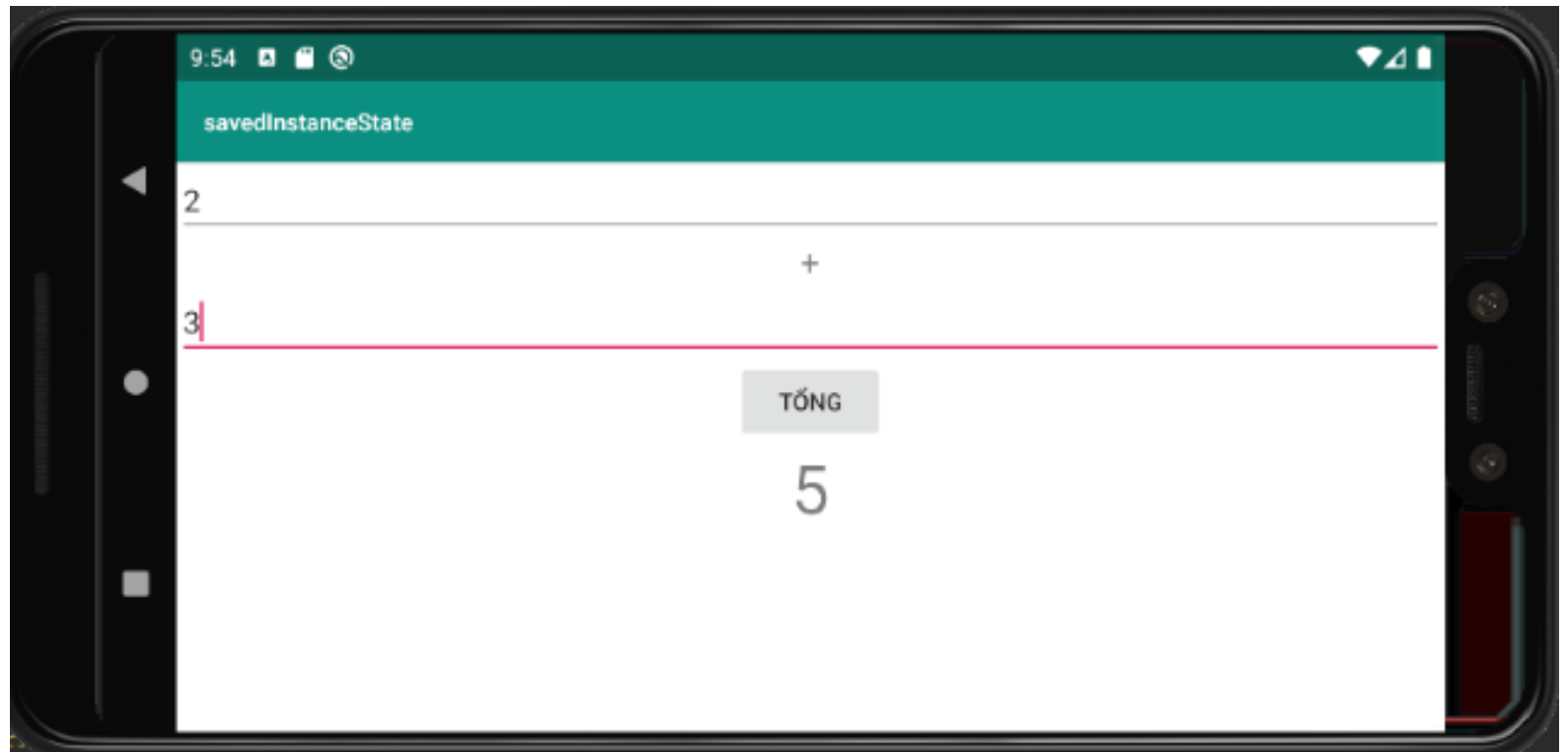
## 2.9/ Làm quen với savedInstanceState



Ví dụ:



Khi xoay ngang màn hình.



## 2.10/ Bắt sự kiện click của Button

- Button là một View nhưng chúng có thể cho phép ta tương tác bằng cách bấm vào nó. Nhưng trước khi bạn muốn button nhận biết sự kiện click thì các bạn phải bắt sự kiện click cho button đó.

### File .xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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">

    <Button
        android:id="@+id/bnt_click"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click" />

</LinearLayout>
```

### File .java

```
public class MainActivity extends AppCompatActivity {

    Button button; //Khai báo một button

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        button = findViewById(R.id.bnt_click); //Tìm lại button
    }
}
```

## 2.10/ Bắt sự kiện click của Button

### a) Bắt trực tiếp.

Để bắt trực tiếp chúng ta có thể dùng hàm `setOnClickListener()` bằng cách truyền vào một đối tượng `OnClickListener`.

```
public class MainActivity extends AppCompatActivity {  
  
    Button button; //Khai báo một button  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        button = findViewById(R.id.bnt_click); //Tìm lại button  
        button.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                Toast.makeText(MainActivity.this, "Bạn đã click", Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

## 2.10/ Bắt sự kiện click của Button

### a) Bắt trực tiếp.

Để bắt trực tiếp chúng ta có thể dùng hàm **setOnClickListener()** bằng cách truyền vào một đối tượng **OnClickListener**.

## Code giao diện

```
<?xml version="1.0" encoding="utf-8"?>
<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"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nguyenvanquan7826.tut2texteditbutton.MainActivity">

    <TextView
        android:id="@+id/tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />

    <EditText
        android:id="@+id/edit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter text here" />

    <Button
        android:id="@+id/btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click me" />

</LinearLayout>
```

## 2.10/ Bắt sự kiện click của Button

### a) Bắt trực tiếp.

Để bắt trực tiếp chúng ta có thể dùng hàm **setOnClickListener()** bằng cách truyền vào một đối tượng **OnClickListener**.

# Code trong file .java

```
public class MainActivity extends AppCompatActivity {  
  
    /*  
    * khai bao cac bien view  
    * */  
    private TextView tv;  
    private EditText edit;  
    private Button btn;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        // goi ham ket noi view  
        connectView();  
    }  
    /*  
    * ket noi cac thanh phan view  
    * */  
    private void connectView() {  
        tv = (TextView) findViewById(R.id.tv);  
        edit = (EditText) findViewById(R.id.edit);  
        btn = (Button) findViewById(R.id.btn);  
    }  
}
```



## 2.10/ Bắt sự kiện click của Button

### a) Bắt trực tiếp.

Để bắt trực tiếp chúng ta có thể dùng hàm **setOnClickListener()** bằng cách truyền vào một đối tượng **OnClickListener**.

# Code trong file .java

```
/*
 * ket noi cac thanh phan view
 */
private void connectView() {
    tv = (TextView) findViewById(R.id.tv);
    edit = (EditText) findViewById(R.id.edit);
    btn = (Button) findViewById(R.id.btn);

    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            doClickButton();
        }
    });
}

private void doClickButton() {
    String text = edit.getText().toString().trim();
    tv.setText(text);
    edit.setText("");
}
```

## 2.10/ Bắt sự kiện click của Button

### b) Bắt trong xml.

Ta có thể thêm sự kiện **onClick** trong chính file xml và trong file java ta thêm hàm thôngbao như sau:

#### File .xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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">

    <Button
        android:onClick="ThongBao"
        android:id="@+id/bnt_click"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click" />

</LinearLayout>
```

#### File .java

```
public void ThongBao(View view) {
}
```



## 2.10/ Bắt sự kiện click của Button

### b) Bắt sự kiện cho nhiều nút.

Muốn bắt sự kiện cho nhiều button ta chỉ cần **implements** interface

**View.OnClickListener**. Sau đó các bạn override lại phương thức `onClick` bằng cách nhấn `alt + enter` và chọn `Implement methods`:

# Code giao diện

```
<?xml version="1.0" encoding="utf-8"?>
<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"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.nguyenvanquan7826.tut2texteditbutton.MainActivity">

    <TextView
        android:id="@+id/tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />

    <EditText
        android:id="@+id/edit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter text here" />

    <Button
        android:id="@+id/btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click me" />

    <Button
        android:id="@+id/btnClear"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Clear" />

</LinearLayout>
```

## 2.10/ Bắt sự kiện click của Button

### b) Bắt sự kiện cho nhiều nút.

Muốn bắt sự kiện cho nhiều button ta chỉ cần **implements** interface

**View.OnClickListener**. Sau đó các bạn override lại phương thức `onClick` bằng cách nhấn `alt + enter` và chọn `Implement methods`:

# Code file .java

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {  
  
    /*  
    * khai bao cac bien view  
    * */  
    private TextView tv;  
    private EditText edit;  
    private Button btn;  
    private Button btnClear;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        // goi ham ket noi view  
        connectView();  
    }  
  
    /*  
    * ket noi cac thanh phan view  
    * */  
    private void connectView() {  
        tv = (TextView) findViewById(R.id.tv);  
        edit = (EditText) findViewById(R.id.edit);  
        btn = (Button) findViewById(R.id.btn);  
        btnClear = (Button) findViewById(R.id.btnClear);  
  
        // set on click  
        btn.setOnClickListener(this);  
        btnClear.setOnClickListener(this);  
    }  
}
```

## 2.10/ Bắt sự kiện click của Button

### b) Bắt sự kiện cho nhiều nút.

Muốn bắt sự kiện cho nhiều button ta chỉ cần **implements** interface

**View.OnClickListener**. Sau đó các bạn override lại phương thức `onClick` bằng cách nhấn `alt + enter` và chọn **Implement methods**:

```
@Override
public void onClick(View view) {
    switch (view.getId()) {
        case R.id.btn:
            doClickButton();
            break;
        case R.id.btnClear:
            doClickButtonClear();
            break;
    }
}

private void doClickButton() {
    String text = edit.getText().toString().trim();
    tv.setText(text);
    edit.setText("");
}

private void doClickButtonClear() {
    edit.setText("");
}
```

Code file .java

# Câu hỏi thảo luận

- 1) Activity là gì ? Trình bày vòng đời của Activity trong ứng dụng Android.
- 2) Trình bày các trạng thái của Activity ?
- 3) Trình bày các cách bắt sự kiện click cho nút lệnh trong ứng dụng Android