1. Mô tả phương pháp quản lý các trạng thái của Activity trong Android

Mỗi phương thức callback tương ứng với mỗi giai đoạn trong vòng đời của activity. Khi một activity thay đổi state, phương thức callback được liên kết sẽ được call. Để phản hồi các hành động thì ta overwrite các phương thức callback của activity.

1. Hãy mô tả các hàm callback được gọi khi Activity chuyển từ trạng thái này sang trạng thái khác trong lifecycle.

Các hàm callback bao gồm:

onCreate();

onStart();

onResume();

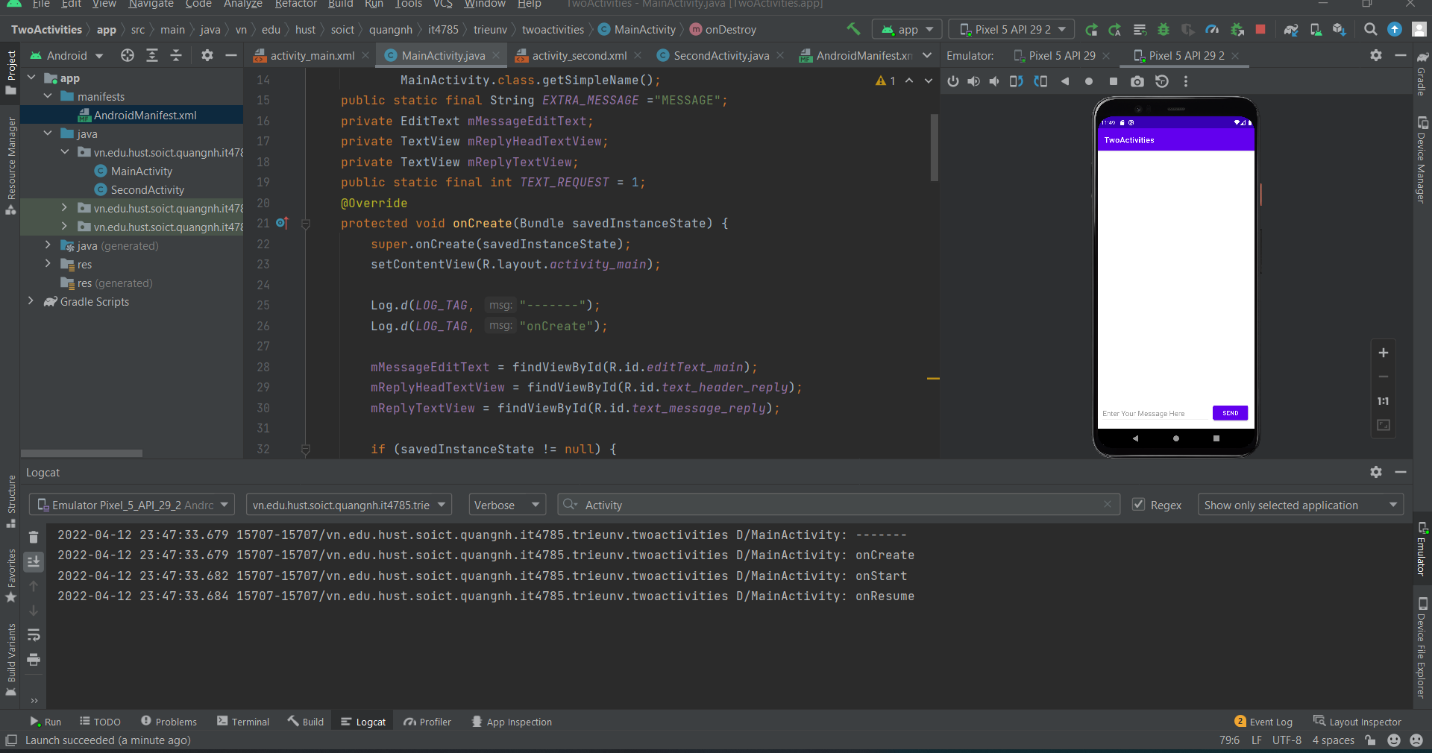
onRestart();

onPause();

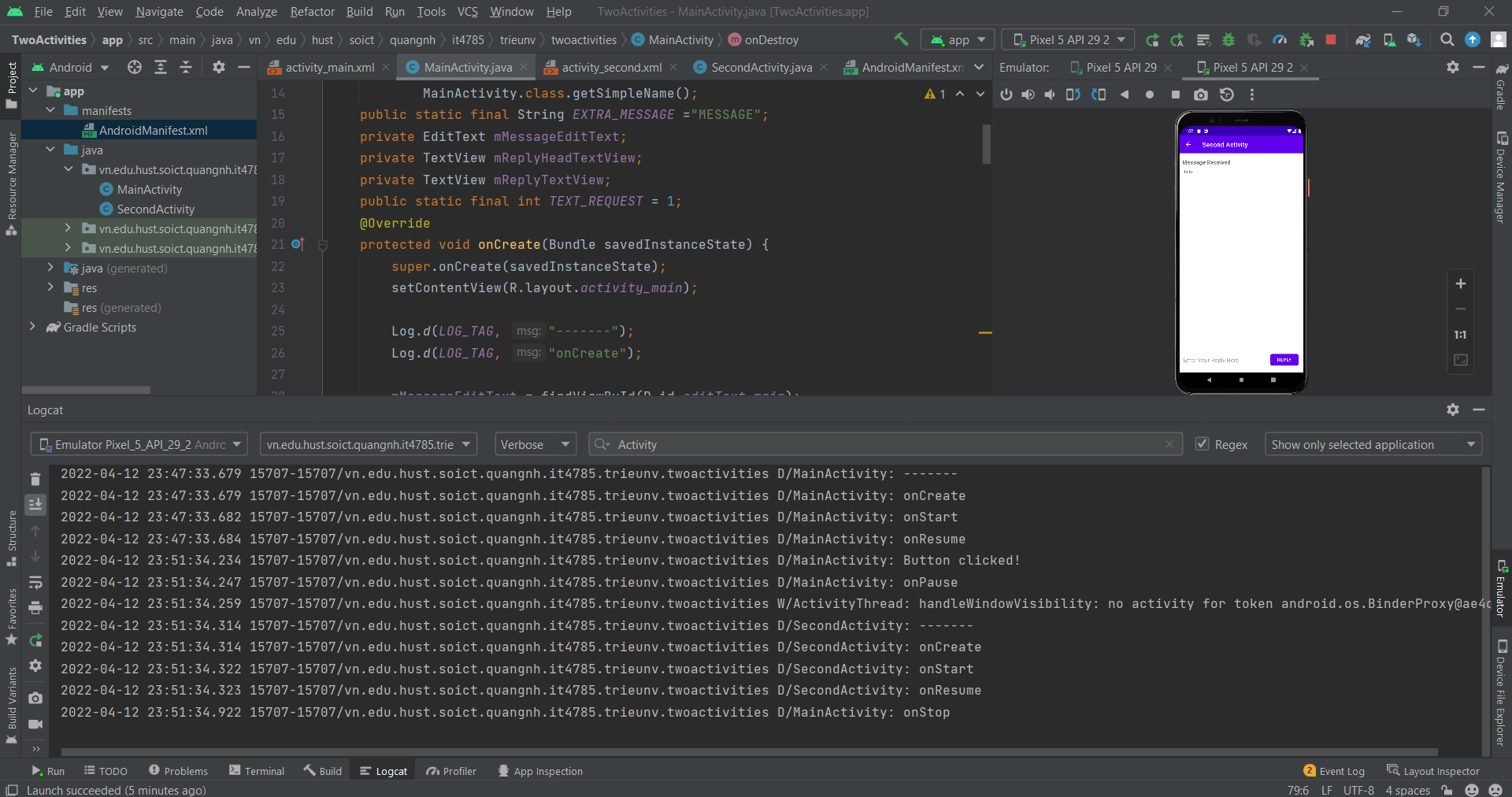
onStop();

onDestroy();

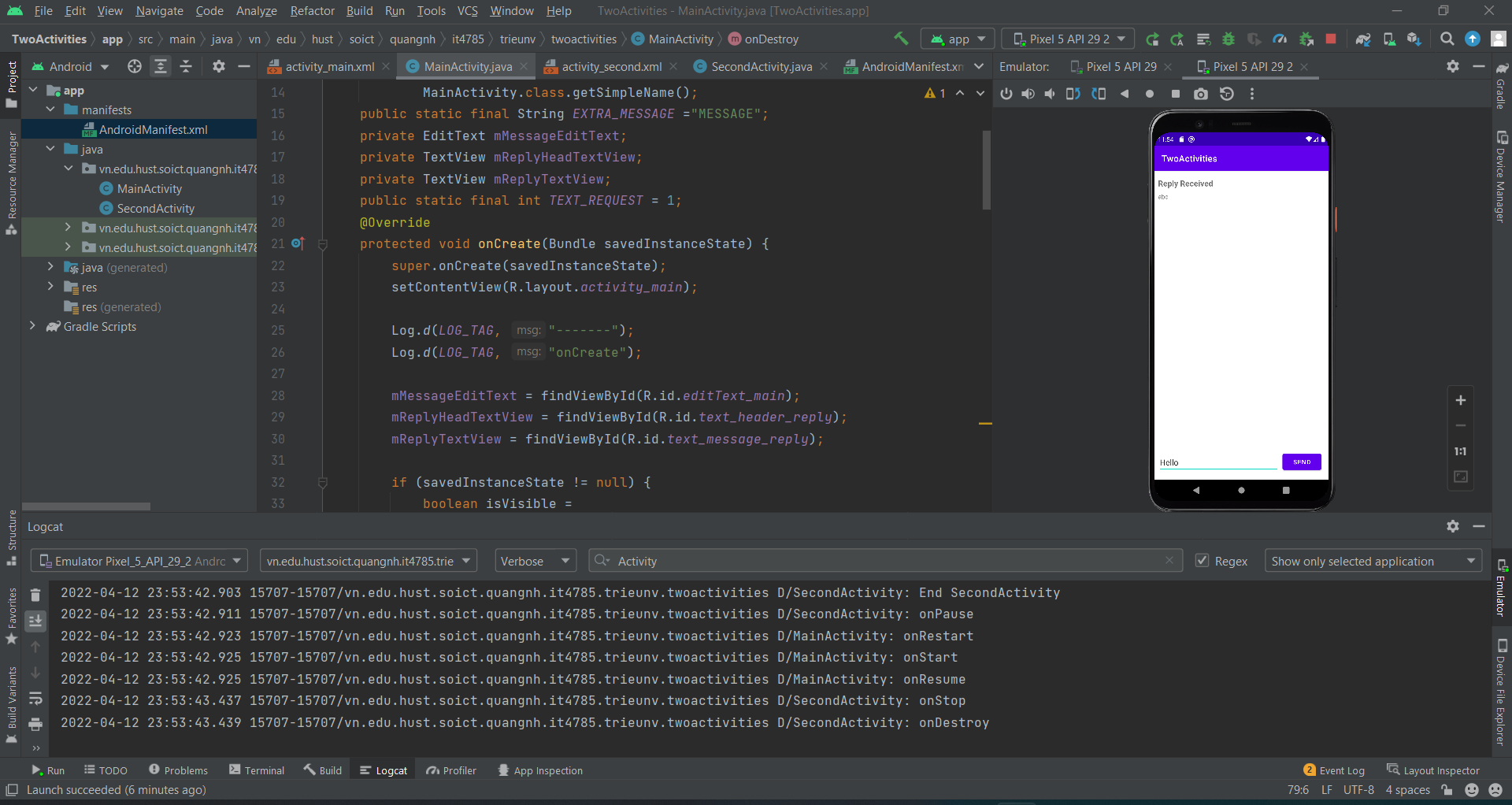
1. Hãy mô tả các hàm callback được gọi trong các trường hợp sau:
   * Use the app normally (send a message, reply with another message):
     + Khi ứng dụng bắt đầu chạy thì MainActivity được hiển thị đầu tiên: onCreate, onStart, onResume



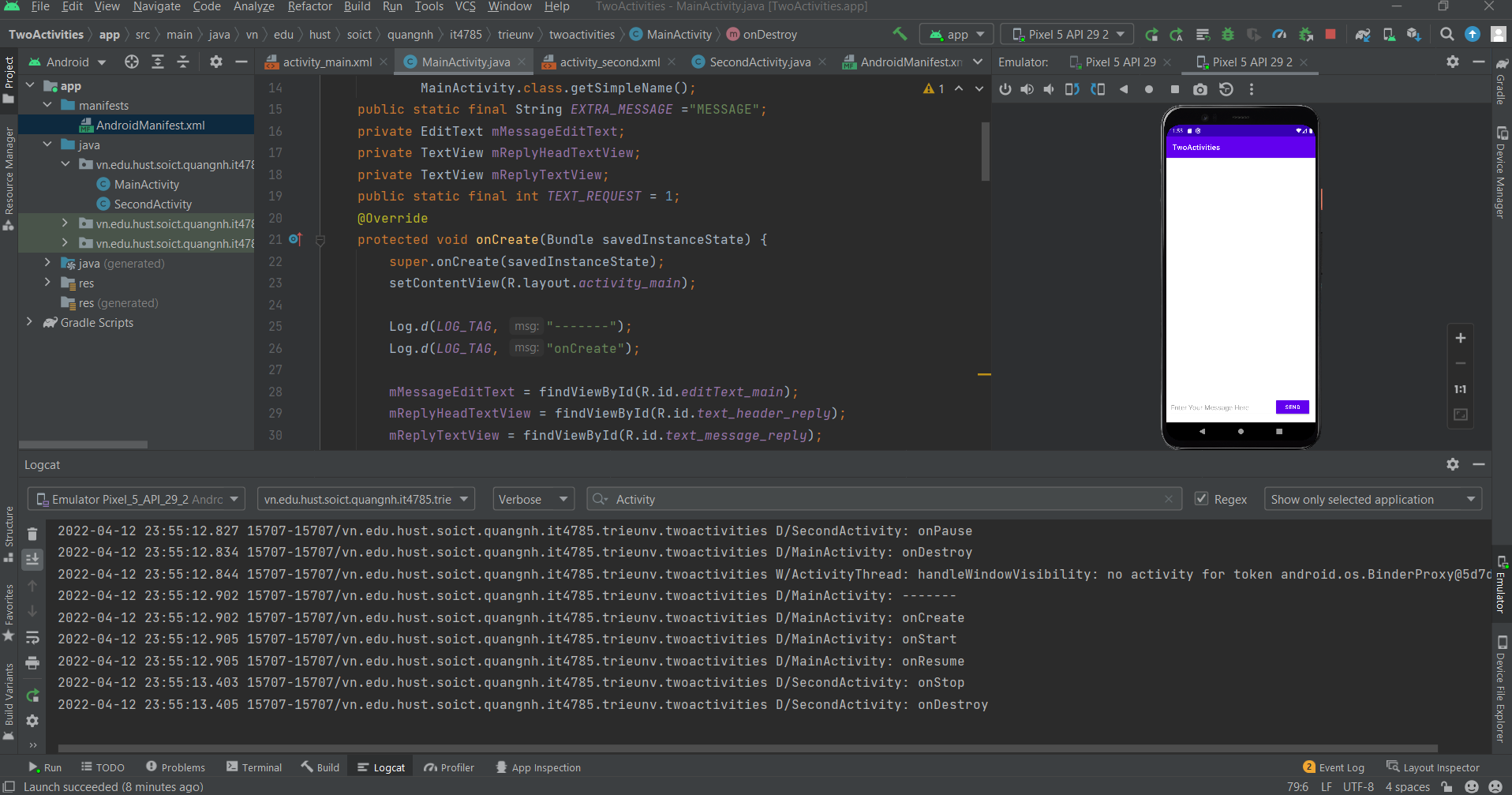
* + - Khi MainActivity chuyển quyền điều khiển sang SecondActivity: MainActivity sẽ gọi onPause, onStop, còn SecondActivity sẽ gọi onCreate, onStart và onResume.



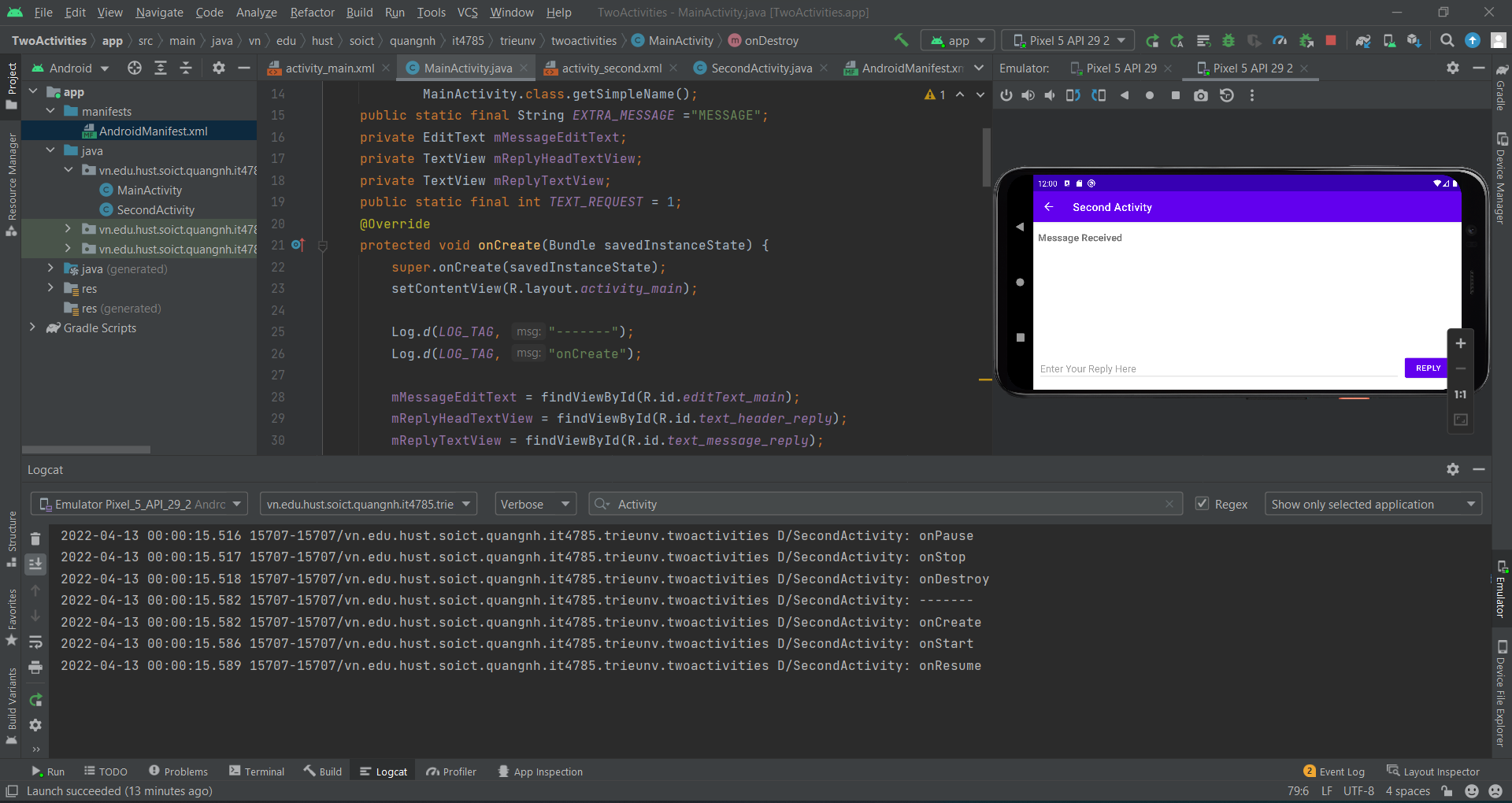
* + - Khi SecondActivity chuyển quyền điều khiển về cho MainActivity: SecondActivity sẽ gọi onResume, onStop, onDestroy; MainActivity sẽ gọi onRestart, onStart, onResume



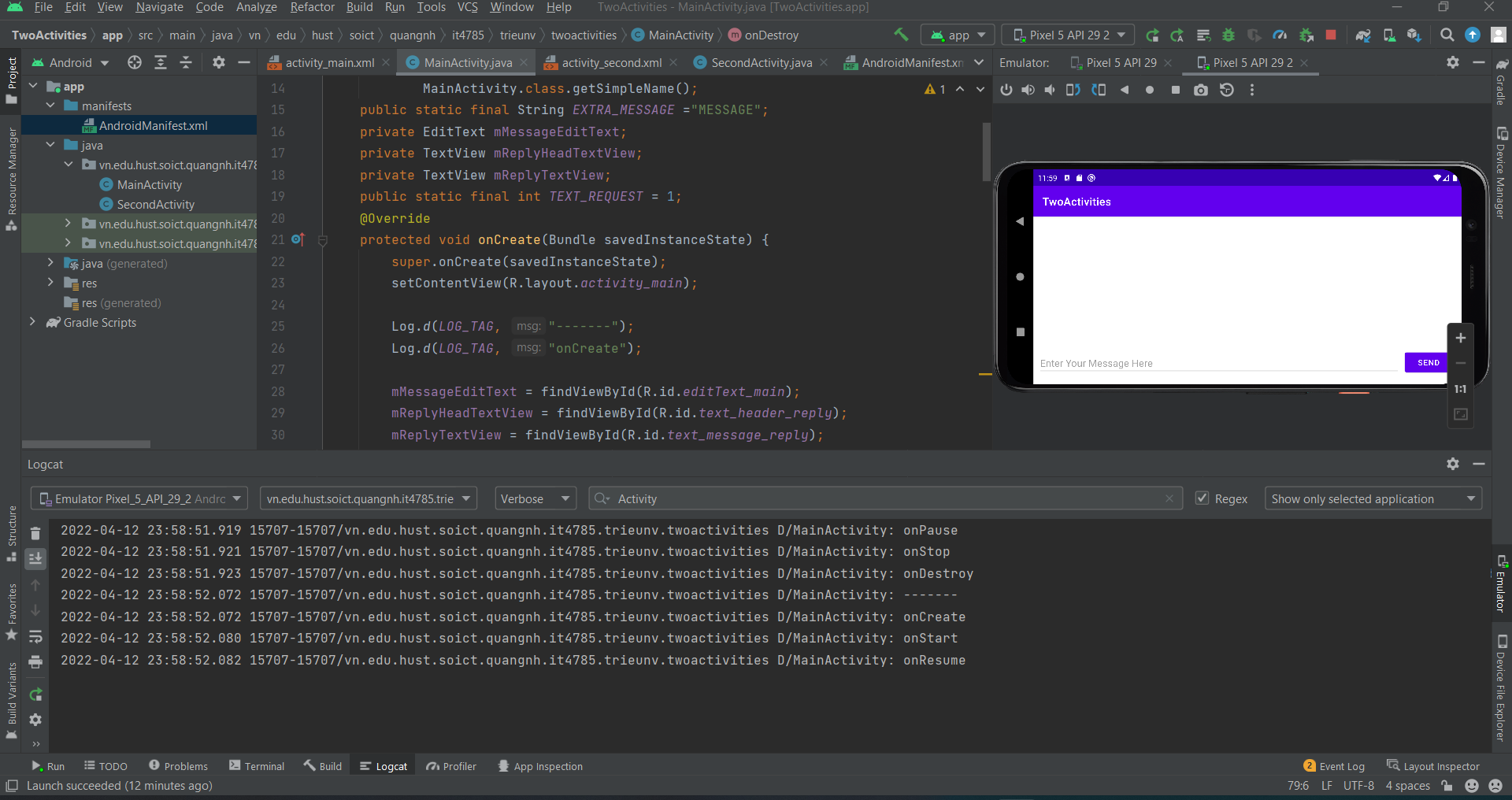
* + Use the Back button to go back from the second Activity to the main Activity: SecondActivity sẽ gọi onResume, onStop, onDestroy; MainActivity sẽ gọi onRestart, onStart, onResume => tương đương (về việc gọi các hàm callback của Activity’s lifecycle) với việc gọi hàm finish(). Tuy nhiên khi ấn nút Back: mã trả về là RESULT\_CANCELLED và Intent trả về là null.
  + Use the Up arrow in the app bar to go back from the second Activity to the main Activity, Khi người dùng ấn nút Up (mũi tên lùi) trên thanh công cụ của SecondActivity để chuyển quyền điều khiển về cho MainActivity: SecondActivity sẽ gọi onResume, onStop, onDestroy; MainActivity cũ sẽ bị hủy và sau đó được tạo lại mới, vì vậy các hàm callback được gọi: onDestroy, onCreate, onStart, onResume.



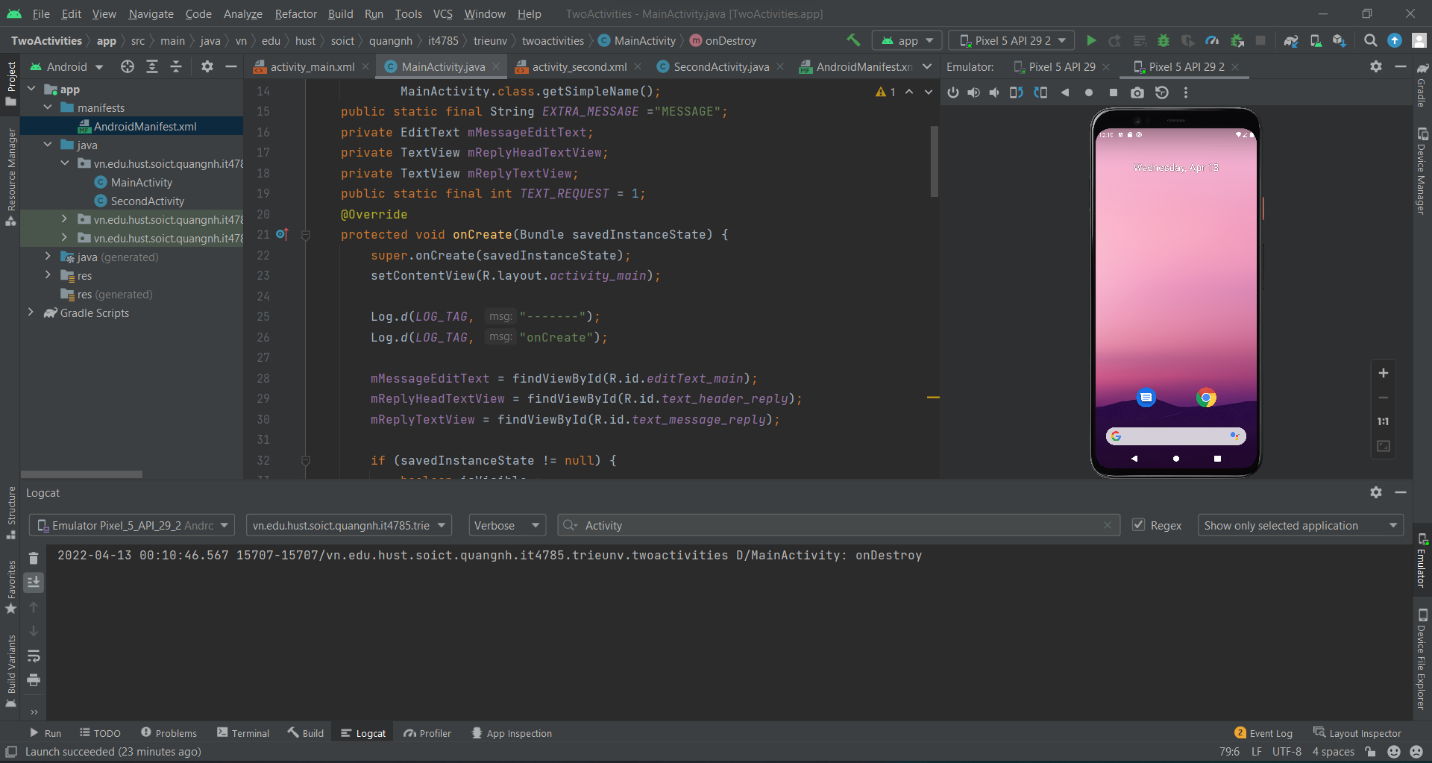
* + Rotate the device on both the main and second Activity at different times in your app and observe what happens in the log and on the screen. Khi xoay màn hình: Activity cũ bị hủy, và Activity mới được tạo ra
    - Khi xoay màn hình với SecondActivity: các hàm callback onPause, onStop và onDestroy, onCreate, onStart, onResume; dữ liệu được bảo toàn bao gồm: dữ liệu do MainActivity gửi sang (do đối tượng Intent không bị hủy) và dữ liệu do người dùng nhập vào ô EditText (trước khi Activity hiện tại bị hủy, Android sẽ tự động lưu một số dữ liệu cơ bản vào đối tượng Bundle, sau đó khi Activity mới được tạo ra thì đối tượng này được gửi đến cho Activity mới, Activity mới có thể truy cập đối tượng này trong hàm onCreate và tự động cập nhật lại các thành phần giao diện tương ứng).



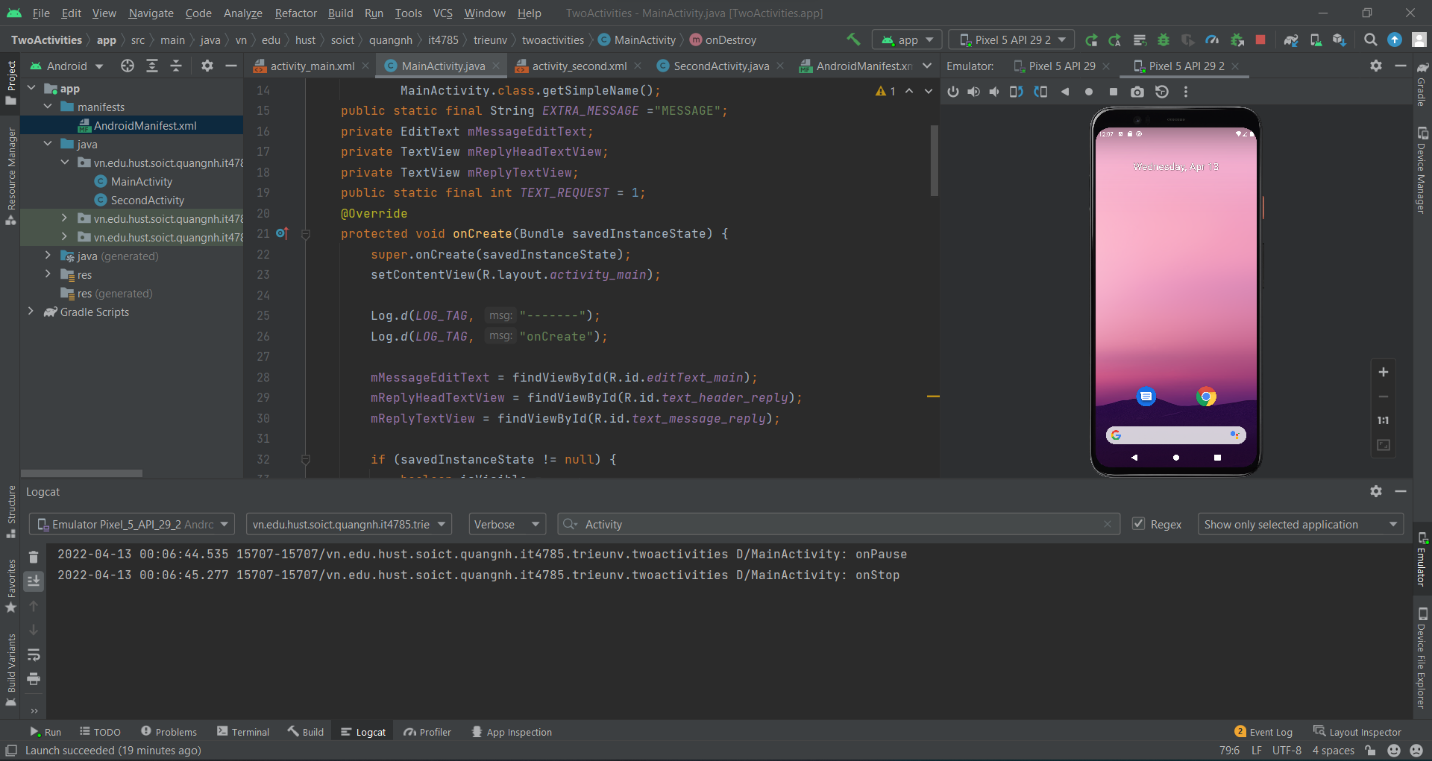
* + - Khi xoay màn hình với MainActivity: dữ liệu do người dùng nhập vào ô EditText vẫn được bảo toàn trong khi đó dữ liệu do SecondActivity gửi sang thì lại bị mất.

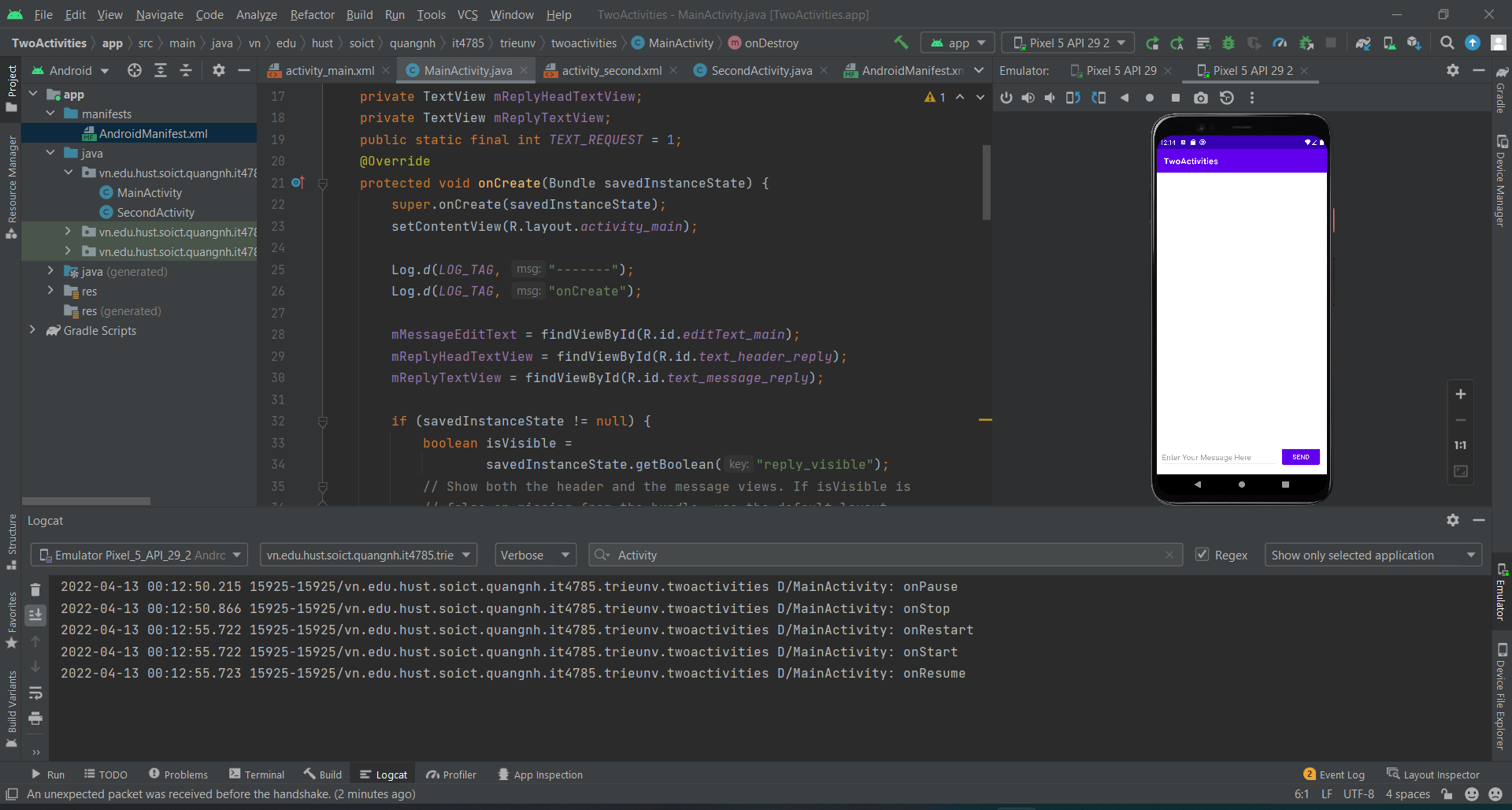


* + Press the overview button (the square button to the right of Home) and close the app (tap the X): Hàm callback onDestroy được gọi từ MainActivity



* + Return to the home screen and restart your app: hàm callback onPause, onStop, sau đó là onRestart, onStart, onResum được gọi từ MainActivity





1. Hãy mô tả phương pháp khôi phục lại các dữ liệu của Activity đã bị mất khi xoay màn hình: trước khi Activity cũ bị hủy thì sẽ lưu các dữ liệu này vào đối tượng Bundle, sau khi Activity mới được tạo ra thì trong hàm onCreate sẽ lấy các dữ liệu này trong đối tượng Bundle savedInstanceState và sử dụng để khôi phục lại các thành phần giao diện tương ứng.

@Override  
public void onSaveInstanceState(Bundle outState) {  
super.onSaveInstanceState(outState);  
}

Hàm onSaveInstanceState được Android gọi trước khi Activity cũ bị hủy. Chúng ta thực hiện lưu trữ lại các dữ liệu hiện tại của Activity vào đối tượng outState. Khi Activity mới được tạo ra thì đối tượng này có thể truy cập trong hàm onCreate với tên biến savedInstanceState.

@Override  
public void onSaveInstanceState(@NonNull Bundle outState) {  
super.onSaveInstanceState(outState);  
if (mReplyHeadTextView.getVisibility() == View.*VISIBLE*) {  
outState.putBoolean("reply\_visible", true);  
outState.putString("reply\_text",

mReplyTextView.getText().toString());  
}  
}

if (savedInstanceState != null) {  
boolean isVisible = savedInstanceState.getBoolean("reply\_visible");  
if (isVisible) {  
mReplyHeadTextView.setVisibility(View.*VISIBLE*);  
mReplyTextView.setVisibility(View.*VISIBLE*);  
String savedText =

savedInstanceState.getString("reply\_text");  
mReplyTextView.setText(savedText);  
}  
}